eBook Gratuit

APPRENEZ Android

eBook gratuit non affilié créé à partir des **contributeurs de Stack Overflow.**

#android

Table des matières

À propos

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

Chapitre 1: Démarrer avec Android

Remarques

Si vous souhaitez en savoir plus sur le paramètre Android Gradle Plugin, consultez la documentation `android-gradle` .

Si vous êtes intéressé par les émulateurs alternatifs, vous pouvez regarder Genymotion . Il fournit un plan gratuit et nécessite moins de RAM.

Versions

Version	Niveau API	Code de version	Date de sortie
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	dix	`GINGERBREAD_MR1`	2011-02-09
3.0.x	11	`HONEYCOMB`	2011-02-22
3.1.x	12	`HONEYCOMB_MR1`	2011-05-10
3.2.x	13	`HONEYCOMB_MR2`	2011-07-15
4.0	14	`ICE_CREAM_SANDWICH`	2011-10-18
4.0.3	15	`ICE_CREAM_SANDWICH_MR1`	2011-12-16
4.1	16	`JELLY_BEAN`	2012-07-09
4.2	17	`JELLY_BEAN_MR1`	2012-11-13

Version	Niveau API	Code de version	Date de sortie
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` (guimauve)	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` (Aperçu du développeur 4)	2017-07-24

Examples

Configurer Android Studio

Android Studio est l'IDE de développement Android officiellement pris en charge et recommandé par Google. Android Studio est fourni avec Android SDK Manager , un outil permettant de télécharger les composants du `Android SDK` requis pour commencer à développer des applications.

Installation des outils Android Studio et `Android SDK` :

1. Téléchargez et installez Android Studio .
2. Téléchargez les outils SDK et les outils de plate-forme SDK les plus récents en ouvrant Android Studio, puis en suivant les instructions de mise à jour de l'outil SDK Android . Vous devez installer les derniers packages stables disponibles.

   Si vous avez besoin de travailler sur d'anciens projets construits à l'aide d'anciennes versions de SDK, vous devrez peut-être également télécharger ces versions

Depuis Android Studio 2.2, une copie de la dernière OpenJDK est fournie avec l'installation et est le JDK (Java Development Kit) recommandé pour tous les projets Android Studio. Cela supprime la nécessité d'installer le package JDK d'Oracle. Pour utiliser le SDK fourni, procédez comme suit:

1. Ouvrez votre projet dans Android Studio et sélectionnez **Fichier> Structure du projet** dans la barre de menus.
2. Dans la page **Emplacement** du **SDK** et sous **JDK** , cochez la case **Utiliser le JDK intégré** .
3. Cliquez sur **OK** .

Configurer Android Studio

Android Studio permet d'accéder à deux fichiers de configuration via le menu **Aide** :

- studio.vmoptions : personnalisez les options de la machine virtuelle Java (JVM) de Studio, telles que la taille du tas et la taille du cache. Notez que sur les machines Linux, ce fichier peut être nommé *studio64.vmoptions* , selon votre version d'Android Studio.
- idea.properties : personnalise les propriétés d'Android Studio, telles que le chemin du dossier des plug-ins ou la taille de fichier maximale prise en charge.

Changer / ajouter un thème

Vous pouvez le changer selon vos préférences. `File->Settings->Editor->Colors & Fonts->` et sélectionnez un thème.Vous pouvez également télécharger de nouveaux thèmes à partir de http://color-themes.com/ Une fois que vous avez téléchargé le fichier `.jar.zip` , allez dans `File -> Import Settings...` et choisissez le fichier téléchargé.

Compiler les applications

Créez un nouveau projet ou ouvrez un projet existant dans Android Studio et appuyez sur le bouton de lecture vert  dans la barre d'outils supérieure pour l'exécuter. S'il est gris, vous devez attendre une seconde pour permettre à Android Studio d'indexer correctement certains fichiers, dont la progression est visible dans la barre d'état inférieure.

Si vous souhaitez créer un projet à partir du shell, vérifiez que vous disposez d'un fichier `local.properties` créé automatiquement par Android Studio. Si vous devez créer le projet sans Android Studio, vous avez besoin d'une ligne commençant par `sdk.dir=` suivi du chemin d'accès à votre installation SDK.

Ouvrez un shell et accédez au répertoire du projet. Entrez `./gradlew aR` et appuyez sur Entrée. `aR` est un raccourci pour `assembleRelease` , qui télécharge toutes les dépendances pour vous et construit l'application. Le fichier APK final sera dans `ProjectName/ModuleName/build/outputs/apk` et s'appellera `ModuleName-release.apk` .

Créer un nouveau projet

Configurer Android Studio

Commencez par configurer Android Studio , puis ouvrez-le. Maintenant, vous êtes prêt à créer votre première application Android!

> **Remarque:** ce guide est basé sur Android Studio 2.2, mais le processus sur les autres versions est essentiellement le même.

Configurez votre projet

Configuration de base

Vous pouvez démarrer un nouveau projet de deux manières:

- Cliquez sur `Start a New Android Studio Project` partir de l'écran d'accueil.
- Accédez à `File` → `New Project` si un projet est déjà ouvert.

Ensuite, vous devez décrire votre application en remplissant certains champs:

1. **Nom de l'application** - Ce nom sera affiché à l'utilisateur.

   Exemple: `Hello World` . Vous pouvez toujours le modifier ultérieurement dans le fichier `AndroidManifest.xml` .

2. **Domaine de la société** - Il s'agit du qualificatif correspondant au nom du package de votre projet.

   Exemple: `stackoverflow.com` .

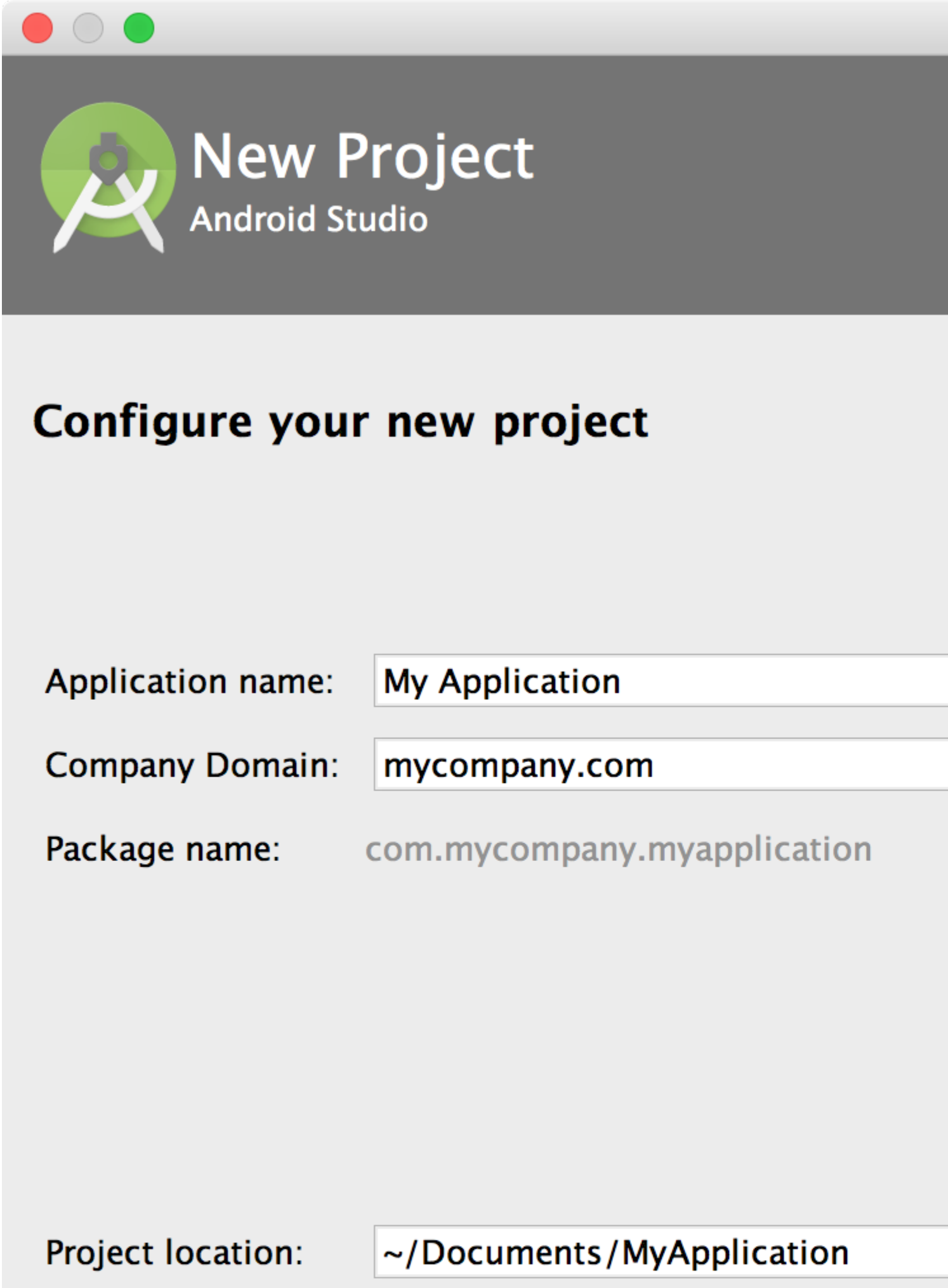
3. **Nom** du **package** (aka `applicationId`) - Ceci est le nom du package de projet *complet.*

   Il doit suivre la *notation inversée des noms de domaine* (*DNS inversé*): *domaine de premier niveau* **.** *Compagnie* ***Domaine.*** [*Segment de la société* **.**] *Nom de l'application* .

   Exemple: `com.stackoverflow.android.helloworld` ou `com.stackoverflow.helloworld` . Vous pouvez toujours modifier votre *application* en la remplaçant dans votre fichier de graduation .

   N'utilisez pas le préfixe par défaut "com.example" sauf si vous n'avez pas l'intention de soumettre votre application au Google Play Store. Le nom du package sera votre **application** unique dans Google Play.

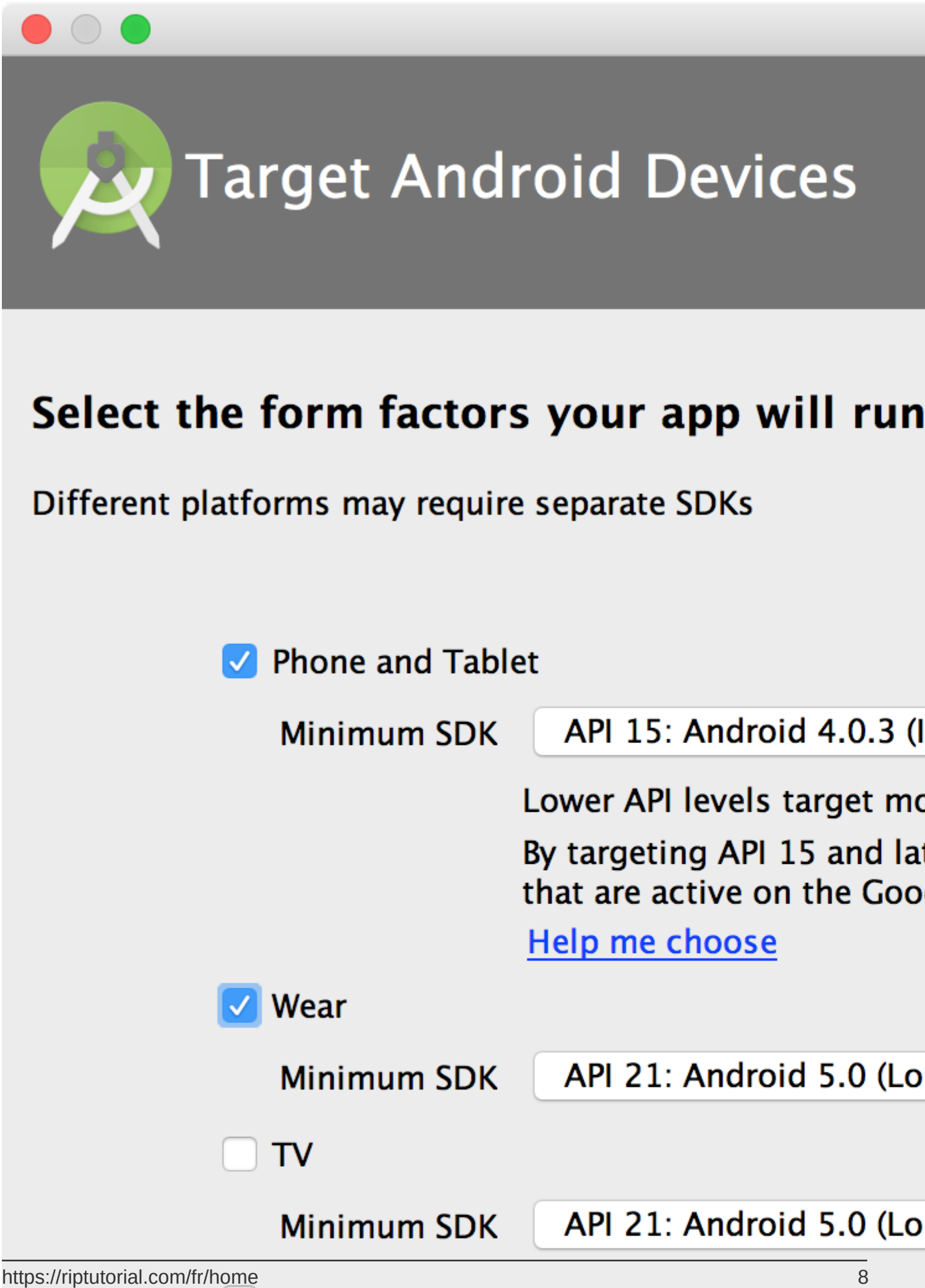
4. **Emplacement du projet** - C'est le répertoire dans lequel votre projet sera stocké.


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

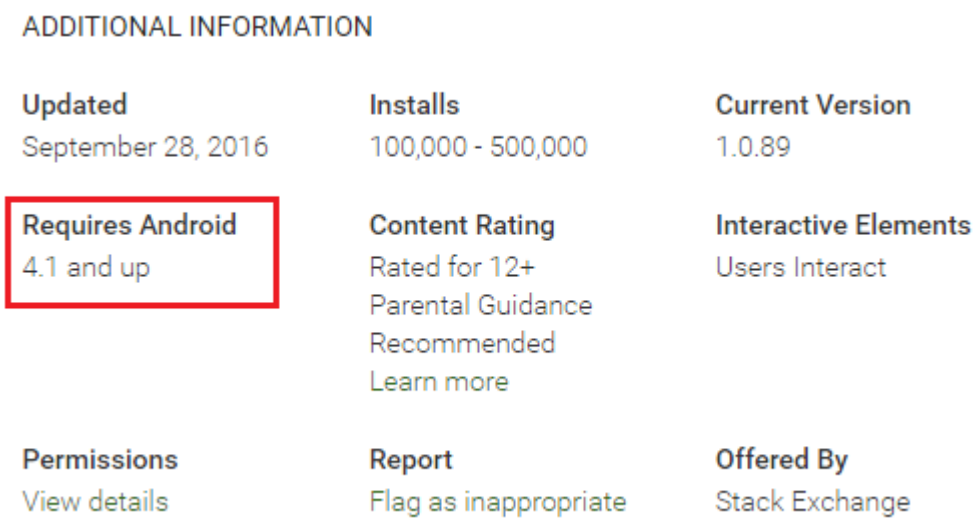
Graphique des distributions de versions Android actuelles, affichées lorsque vous cliquez sur Aidez-moi à choisir.

La fenêtre Distribution de plate-forme Android affiche la distribution des appareils mobiles exécutant chaque version d'Android, comme illustré à la Figure 2. Cliquez sur un niveau d'API pour afficher la liste des fonctionnalités introduites dans la version correspondante d'Android. Cela vous aide à choisir le niveau minimum d'API doté de toutes les fonctionnalités dont vos applications ont besoin pour pouvoir accéder à autant de périphériques que possible. Puis cliquez sur **OK** .

Maintenant, choisissez les plates-formes et la version du SDK Android que l'application prendra en charge.


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` pour déterminer les périphériques sur lesquels une application peut être installée. Par exemple, l'application Stack Exchange prend en charge Android 4.1+.



Android Studio vous indiquera (approximativement) quel pourcentage d'appareils sera pris en charge, compte tenu du SDK minimum spécifié.

> Les niveaux inférieurs de l'API ciblent davantage de périphériques mais disposent de moins de fonctionnalités.

Lorsque vous décidez du **kit SDK minimum** , vous devez tenir compte des statistiques de Dashboards , qui vous donneront des informations sur la version des appareils ayant visité le Google Play Store au cours de la dernière semaine.

De: Dashboards sur le site Web des développeurs Android.

Ajouter une activité

Nous allons maintenant sélectionner une activité par défaut pour notre application. Dans Android, une `Activity` est un écran unique qui sera présenté à l'utilisateur. Une application peut héberger plusieurs activités et naviguer entre elles. Pour cet exemple, choisissez `Empty Activity` et cliquez sur Suivant.

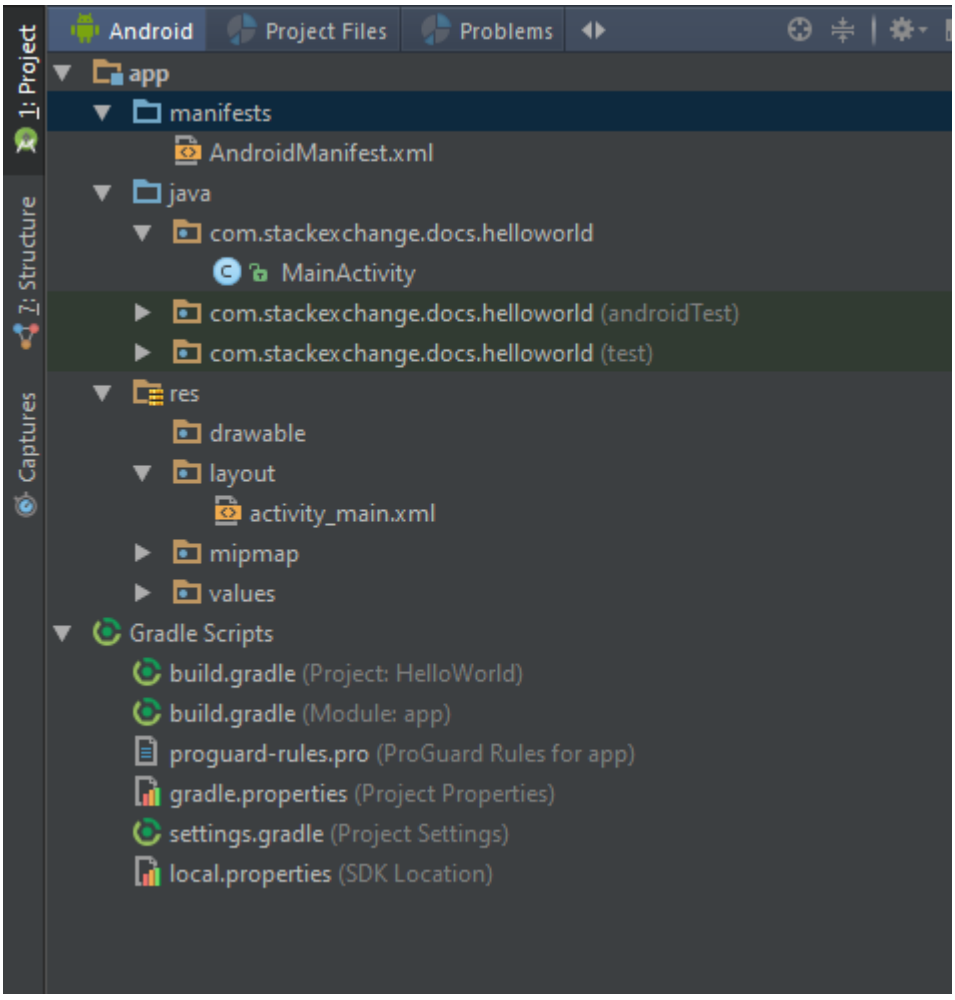
Ici, si vous le souhaitez, vous pouvez changer le nom de l'activité et de la mise en page. Une bonne pratique consiste à conserver l' `Activity` comme suffixe pour le nom de l'activité et `activity_` comme préfixe pour le nom de la mise en page. Si nous les laissons par défaut, Android Studio générera pour nous une activité appelée `MainActivity` et un fichier de mise en page appelé `activity_main` . Maintenant, cliquez sur `Finish` .

Android Studio créera et configurera notre projet, ce qui peut prendre un certain temps en fonction du système.

Inspection du projet

Pour comprendre comment fonctionne Android, jetons un coup d'œil à certains fichiers créés pour nous.

Sur le volet gauche d'Android Studio, nous pouvons voir la structure de notre application Android .



Tout d'abord, ouvrons `AndroidManifest.xml` en double-cliquant dessus. Le fichier manifeste Android décrit certaines des informations de base sur une application Android. Il contient la déclaration de nos activités, ainsi que des composants plus avancés.

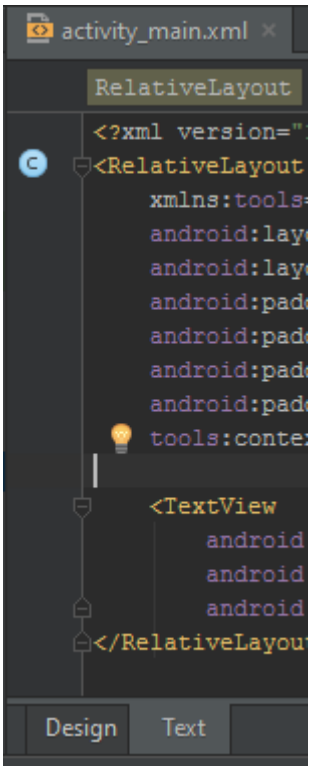
Si une application a besoin d'accéder à une fonctionnalité protégée par une autorisation, elle doit déclarer qu'elle requiert cette autorisation avec un élément `<uses-permission>` dans le manifeste.

Ensuite, lorsque l'application est installée sur le périphérique, le programme d'installation détermine s'il convient ou non d'accorder l'autorisation demandée en vérifiant les autorités ayant signé les certificats de l'application et, dans certains cas, en demandant à l'utilisateur. Une application peut également protéger ses propres composants (activités, services, récepteurs de diffusion et fournisseurs de contenu) avec des autorisations. Il peut utiliser l'une des autorisations définies par Android (répertoriées dans android.Manifest.permission) ou déclarées par d'autres applications. Ou il peut définir les siens.

```
<?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>
```

Ensuite, ouvrons `activity_main.xml` qui se trouve dans `app/src/main/res/layout/` . Ce fichier contient des déclarations pour les composants visuels de notre MainActivity. Vous verrez le concepteur visuel. Cela vous permet de faire glisser et de déposer des éléments sur la mise en page sélectionnée.

Vous pouvez également basculer vers le concepteur de mise en page XML en cliquant sur "Texte" en bas de Android Studio, comme illustré ici:




```
<?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>
```

Vous verrez un widget appelé `TextView` dans cette présentation, avec la propriété `android:text` définie sur "Hello World!". Ceci est un bloc de texte qui sera affiché à l'utilisateur lors de l'exécution de l'application.

Vous pouvez en savoir plus sur les mises en page et les attributs .

Ensuite, regardons `MainActivity` . C'est le code Java qui a été généré pour `MainActivity` .

```
public class MainActivity extends AppCompatActivity {

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

        // setContentView sets the Activity's layout to a specified XML layout
```

```
        // In our case we are using the activity_main layout
        setContentView(R.layout.activity_main);
    }
}
```

Comme défini dans notre manifeste Android, `MainActivity` sera lancé par défaut lorsqu'un utilisateur lance l'application `HelloWorld` .

Enfin, ouvrez le fichier nommé `build.gradle` situé dans `app/` .
Android Studio utilise le système de génération **Gradle** pour compiler et créer des applications et des bibliothèques Android.

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

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

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

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

Ce fichier contient des informations sur la version et la version de votre application. Vous pouvez également l'utiliser pour ajouter des dépendances à des bibliothèques externes. Pour l'instant, n'apportons aucune modification.

Il est conseillé de toujours sélectionner la dernière version disponible pour les dépendances:

- buildToolsVersion : 26.0.0
- com.android.support:appcompat-v7 : 26.0.0 (juillet 2017)

- base de feu : 11.0.4 (août 2017)

compileSdkVersion

`compileSdkVersion` est votre façon de dire à *Gradle* quelle version du SDK Android compiler votre application. L'utilisation du nouveau SDK Android est une exigence pour utiliser l'une des nouvelles API ajoutées à ce niveau.

Il convient de souligner que la modification de votre `compileSdkVersion` ne modifie pas le comportement à l'exécution. Bien que de nouveaux avertissements / erreurs du compilateur puissent être présents lors du changement de votre `compileSdkVersion` , votre `compileSdkVersion` n'est pas inclus dans votre APK: il est purement utilisé au moment de la compilation.

Par conséquent, il est fortement recommandé de toujours compiler avec le dernier SDK. Vous obtiendrez tous les avantages des nouvelles vérifications de compilation sur le code existant, éviterez les API récemment obsolètes et serez prêt à utiliser de nouvelles API.

minSdkVersion

Si `compileSdkVersion` définit les dernières API disponibles, `minSdkVersion` est la *limite inférieure* de votre application. `minSdkVersion` est l'un des signaux utilisés par Google Play Store pour déterminer sur quel périphérique d'un utilisateur une application peut être installée.

Il joue également un rôle important pendant le développement: par défaut, les linteaux contre votre projet vous avertissent lorsque vous utilisez des API au-dessus de votre `minSdkVersion` , vous aidant ainsi à éviter le problème d'exécution d'une API inexistante. La vérification de la version du système à l'exécution est une technique courante lors de l'utilisation des API uniquement sur les versions de plate-forme plus récentes.

targetSdkVersion

`targetSdkVersion` est la méthode principale `targetSdkVersion` Android pour `targetSdkVersion` la `targetSdkVersion` la `targetSdkVersion` n'appliquant pas les changements de comportement à moins que `targetSdkVersion` soit mis à jour. Cela vous permet d'utiliser de nouvelles API avant de modifier les comportements. La mise à jour pour cibler le dernier SDK devrait être une priorité élevée pour chaque application. Cela ne signifie pas que vous devez utiliser chaque nouvelle fonctionnalité introduite, ni mettre à jour aveuglément votre `targetSdkVersion` sans le tester.

targetSDKVersion est la version d'Android qui constitue la limite supérieure des outils disponibles. Si targetSDKVersion est inférieur à 23, l'application n'a pas besoin de demander des autorisations à l'exécution pour une instance, même si l'application est exécutée sur API 23+. TargetSDKVersion n'empêche **pas** les versions Android au-dessus de la version Android sélectionnée d'exécuter l'application.

Vous pouvez trouver plus d'informations sur le plugin Gradle:

- Un exemple de base
- Introduction au plugin Gradle pour Android et au wrapper
- Introduction à la configuration des méthodes build.gradle et DSL

Exécution de l'application

Maintenant, exécutons notre application HelloWorld. Vous pouvez soit exécuter un appareil virtuel Android (que vous pouvez configurer en utilisant le Gestionnaire AVD dans Android Studio, comme décrit dans l'exemple ci-dessous) ou connecter votre propre appareil Android via un câble USB.

Configurer un appareil Android

Pour exécuter une application à partir d'Android Studio sur votre appareil Android, vous devez activer le `USB Debugging` dans les `Developer Options` du `Developer Options` dans les paramètres de votre appareil.

`Settings > Developer options > USB debugging`

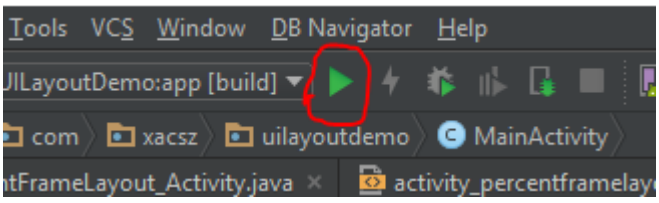
Si les `Developer Options` ne sont pas visibles dans les paramètres, accédez à `About Phone` et appuyez sur le `Build Number` sept fois. Cela permettra aux `Developer Options` de s'afficher dans vos paramètres.

`Settings > About phone > Build number`

Vous devrez peut-être également modifier la configuration de `build.gradle` pour `build.gradle` une version de votre périphérique.

Exécution depuis Android Studio

Cliquez sur le bouton vert `Run` dans la barre d'outils située en haut de Android Studio. Dans la fenêtre qui apparaît, sélectionnez le périphérique sur lequel vous souhaitez exécuter l'application (démarrez un périphérique virtuel Android si nécessaire ou reportez-vous à la section Configuration d'un périphérique virtuel Android si vous devez en configurer un), puis cliquez sur `OK` .



Sur les appareils fonctionnant sous Android 4.4 (KitKat) et éventuellement plus haut, une fenêtre contextuelle apparaîtra pour autoriser le débogage USB. Cliquez sur `OK` pour accepter.

L'application va maintenant installer et exécuter sur votre appareil Android ou émulateur.

Emplacement du fichier APK

Lorsque vous préparez votre application pour la publication, vous configurez, créez et testez une

version finale de votre application. Les tâches de configuration sont simples et impliquent des tâches de base de nettoyage du code et de modification du code qui permettent d'optimiser votre application. Le processus de génération est similaire au processus de génération de débogage et peut être effectué à l'aide des outils JDK et Android SDK. Les tâches de test servent de vérification finale, garantissant que votre application fonctionne comme prévu dans des conditions réelles. Lorsque vous avez terminé de préparer votre demande de publication, vous disposez d'un fichier APK signé, que vous pouvez distribuer directement aux utilisateurs ou distribuer via un marché d'applications tel que Google Play.

Studio Android

Puisque dans les exemples ci-dessus, Gradle est utilisé, l'emplacement du fichier APK généré est: `<Your Project Location>/app/build/outputs/apk/app-debug.apk`

IntelliJ

Si vous utilisez IntelliJ avant de passer à Studio et importez directement votre projet IntelliJ, rien n'a changé. L'emplacement de la sortie sera le même sous:

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

Note: ceci deviendra parfois obsolète autour de 1.0

Éclipse

Si vous importez directement le projet Android Eclipse, ne le faites pas! Dès que vous avez des dépendances dans votre projet (jars ou projets de bibliothèque), cela ne fonctionnera pas et votre projet ne sera pas correctement configuré. Si vous n'avez pas de dépendances, l'apk se trouverait au même endroit que vous le trouveriez dans Eclipse:

```
bin/...
```

Programmation Android sans IDE

Voici un exemple minimaliste de Hello World qui utilise uniquement les outils Android les plus élémentaires.

Exigences et hypothèses

- Oracle JDK 1.7 ou version ultérieure
- Android SDK Tools (juste les outils de ligne de commande)

Cet exemple suppose que Linux. Vous devrez peut-être ajuster la syntaxe de votre propre plate-forme.

Configuration du SDK Android

Après avoir déballé la version du SDK:

1. Installez des packages supplémentaires à l'aide du gestionnaire du SDK. N'utilisez pas la `android update sdk --no-ui` comme indiqué dans le fichier Readme.txt fourni; Il télécharge environ 30 Go de fichiers inutiles. Au lieu de cela, utilisez le gestionnaire de SDK interactif `android sdk` pour obtenir le minimum recommandé de paquets.

2. Ajoutez les répertoires JDK et SDK suivants à votre exécution PATH. Ceci est facultatif, mais les instructions ci-dessous l'assument.

    - JDK / bin
    - SDK / platform-tools
    - SDK / outils
    - SDK / build-tools / LATEST *(tel qu'installé à l'étape 1)*

3. Créez un périphérique virtuel Android. Utilisez le gestionnaire AVD interactif (`android avd`). Vous devrez peut-être bricoler un peu et chercher des conseils; les instructions sur site ne sont pas toujours utiles.

    (Vous pouvez également utiliser votre propre appareil)

4. Exécutez le périphérique:

    ```
    emulator -avd DEVICE
    ```

5. Si l'écran de l'appareil semble être verrouillé, faites-le glisser pour le déverrouiller.

    Laissez-le fonctionner pendant que vous codez l'application.

Coder l'application

6. Passez à un répertoire de travail vide.

7. Faites le fichier source:

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

    Contenu:

    ```
    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. Ajouter un manifeste:

```
touch AndroidManifest.xml
```

Contenu:

```
<?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. Créez un sous-répertoire pour les ressources déclarées:

```
mkdir res
```

Laissez le vide pour le moment.

Construire le code

10. Générez la source pour les déclarations de ressources. Remplacez ici le chemin d'accès correct à votre **SDK** et l' **API** installée pour la construire (par exemple "android-23"):

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

Les déclarations de ressources (décrites ci-dessous) sont facultatives. Pendant ce temps, l'appel ci-dessus ne fait rien si res / est toujours vide.

11. Compilez le code source en bytecode Java (.java → .class):

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

12. Traduire le bytecode de Java en Android (.class → .dex):

    En utilisant d'abord Jill (.class → .jayce):

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

    Puis Jack (.jayce → .dex):

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

    Android bytecode s'appelait auparavant "Dalvik executable code", et donc "dex".

    Vous pouvez remplacer les étapes 11 et 12 par un seul appel à Jack si vous le souhaitez; il peut compiler directement depuis le source Java (.java → .dex). Mais la compilation avec `javac` présente des avantages. C'est un outil mieux connu, mieux documenté et plus largement applicable.

13. Emballez les fichiers de ressources, y compris le manifeste:

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

    Cela se traduit par un fichier APK partiel (package d'application Android).

14. Créez le fichier APK complet à l'aide de l'outil `ApkBuilder` :

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

    Il avertit, "CET OUTIL EST PÉRIMÉ. Voir --help pour plus d'informations." Si `--help` échoue avec une `ArrayIndexOutOfBoundsException` , ne transmettez pas les arguments à la place:

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

    Il explique que l'interface de ligne de commande (`ApkBuilderMain`) est déconseillée au lieu d'appeler directement l'API Java (`ApkBuilder`). (Si vous savez comment faire cela depuis la ligne de commande, veuillez mettre à jour cet exemple.)

15. Optimiser l'alignement des données de l'APK (pratique recommandée):

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

Installation et exécution

16. Installez l'application sur l'appareil Android:

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

17. Démarrez l'application:

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

    Il devrait courir et dire bonjour.

C'est tout. C'est ce qu'il faut pour dire bonjour en utilisant les outils Android de base.

Déclaration d'une ressource

Cette section est facultative. Les déclarations de ressources ne sont pas nécessaires pour une application simple "bonjour monde". Si elles ne sont pas nécessaires pour votre application, vous pouvez simplifier quelque peu la construction en omettant l'étape 10 et en supprimant la référence au répertoire res / de l'étape 13.

Sinon, voici un bref exemple de la façon de déclarer une ressource et de la référencer.

18. Ajoutez un fichier de ressources:

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

    Contenu:

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

19. Référencez la ressource à partir du manifeste XML. Ceci est un style de référence déclaratif:

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

20. Référencez la même ressource de la source Java. Ceci est une référence impérative:

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

21. Testez les modifications ci-dessus en reconstruisant, en réinstallant et en réexécutant l'application (étapes 10 à 17).

    Il devrait redémarrer et dire: "Cette application s'appelle Dire bonjour".

Désinstallation de l'application

```
adb uninstall dom.domain
```

Voir également

- question originale - La question originale qui a incité cet exemple
- Exemple de travail - Un script de construction qui utilise les commandes ci-dessus

Principes de base de l'application

Les applications Android sont écrites en Java. Les outils du SDK Android compilent les fichiers de code, de données et de ressources dans un fichier APK (package Android). Généralement, un fichier APK contient tout le contenu de l'application.

Chaque application s'exécute sur sa propre machine virtuelle (VM) afin que l'application puisse s'exécuter isolée des autres applications. Le système Android fonctionne avec le principe du moindre privilège. Chaque application n'a accès qu'aux composants dont elle a besoin pour faire son travail, pas plus. Cependant, une application peut partager des données avec d'autres applications, par exemple en partageant l'ID utilisateur Linux entre les applications, ou les applications peuvent demander l'autorisation d'accéder aux données du périphérique telles que la carte SD, les contacts, etc.

Composants de l'application

Les composants de l'application sont les éléments constitutifs d'une application Android. Chaque composant joue un rôle spécifique dans une application Android qui sert un objectif distinct et a des cycles de vie distincts (le flux de comment et quand le composant est créé et détruit). Voici les quatre types de composants d'application:

1. **Activités:** Une activité représente un seul écran avec une interface utilisateur. Une application Android peut avoir plusieurs activités. (Par exemple, une application de messagerie peut avoir une activité pour répertorier tous les emails, une autre pour afficher le contenu de chaque email et une autre pour composer un nouvel email). Toutes les activités d'une application fonctionnent ensemble pour créer une expérience utilisateur.
2. **Services:** un service s'exécute en arrière-plan pour effectuer des opérations de longue durée ou pour exécuter des tâches pour des processus distants. Un service ne fournit aucune interface utilisateur, il s'exécute uniquement en arrière-plan avec l'entrée de l'utilisateur. (Par exemple, un service peut lire de la musique en arrière-plan lorsque

l'utilisateur se trouve dans une autre application ou il peut télécharger des données sur Internet sans bloquer l'interaction de l'utilisateur avec l'appareil Android.)

3. **Fournisseurs de contenu:** un fournisseur de contenu gère les données d'application partagée. Il existe quatre manières de stocker des données dans une application: elles peuvent être écrites dans un fichier et stockées dans le système de fichiers, insérées ou mises à jour dans une base de données SQLite, publiées sur le Web ou enregistrées dans tout autre emplacement de stockage persistant. . Grâce à des fournisseurs de contenu, d'autres applications peuvent interroger ou même modifier les données. (Par exemple, le système Android fournit un fournisseur de contenu qui gère les informations de contact de l'utilisateur afin que toute application autorisée puisse interroger les contacts.) Les fournisseurs de contenu peuvent également enregistrer les données privées de l'application pour une meilleure intégrité des données.
4. **Récepteurs de diffusion:** un récepteur de diffusion répond aux diffusions d'annonces diffusées à l'échelle du système (par exemple, une émission annonçant que l'écran est éteint, que la batterie est faible, etc.) ou d'applications (par exemple, téléchargé sur l'appareil et est disponible pour qu'ils puissent l'utiliser). Les récepteurs de diffusion n'ont pas d'interface utilisateur, mais ils peuvent afficher une notification dans la barre d'état pour alerter l'utilisateur. Les récepteurs de diffusion sont généralement utilisés comme passerelles vers d'autres composants de l'application, principalement des activités et des services.

Un aspect unique du système Android est que toute application peut démarrer le composant d'une autre application (par exemple, si vous souhaitez faire un appel, envoyer des SMS, ouvrir une page Web ou afficher une photo, il existe déjà une application l'utiliser, au lieu de développer une nouvelle activité pour la même tâche).

Lorsque le système démarre un composant, il lance le processus pour cette application (s'il n'est pas déjà lancé, c'est-à-dire qu'un seul processus par application peut s'exécuter à tout moment sur un système Android) et instancie les classes nécessaires pour ce composant. Ainsi, le composant s'exécute sur le processus de cette application à laquelle il appartient. Par conséquent, contrairement aux applications sur d'autres systèmes, les applications Android n'ont pas de point d'entrée unique (il n'y a pas de méthode `main()`).

Comme le système exécute chaque application dans un processus distinct, une application ne peut pas activer directement les composants d'une autre application, quel que soit le système Android. Ainsi, pour démarrer le composant d'une autre application, une application doit envoyer un message au système spécifiant l'intention de démarrer ce composant, puis le système démarrera ce composant.

Le contexte

Les instances de la classe `android.content.Context` fournissent la connexion au système Android qui exécute l'application. L'instance de contexte est nécessaire pour accéder aux ressources du projet et aux informations globales sur l'environnement de l'application.

Prenons un exemple facile à comprendre: Considérez que vous êtes dans un hôtel et que vous voulez manger quelque chose. Vous appelez le room-service et demandez-leur de vous apporter

des choses ou de nettoyer des choses pour vous. Maintenant, pensez à cet hôtel comme une application Android, vous-même en tant qu'activité et le room-service est alors votre contexte, ce qui vous permet d'accéder aux ressources de l'hôtel comme le room-service, les produits alimentaires, etc.

Encore un autre exemple, vous êtes dans un restaurant assis sur une table, chaque table a un accompagnateur, chaque fois que vous voulez commander des produits alimentaires, vous demandez au préposé de le faire. Le préposé place ensuite votre commande et vos produits alimentaires sont servis sur votre table. Encore une fois dans cet exemple, le restaurant est une application Android, les tables ou les clients sont des composants App, les produits alimentaires sont les ressources de votre application et le standardiste est votre contexte, ce qui vous permet d'accéder aux ressources comme des produits alimentaires.

L'activation de l'un des composants ci-dessus nécessite l'instance du contexte. Non seulement ce qui précède, mais presque toutes les ressources système: la création de l'interface utilisateur à l'aide de vues (abordées plus loin), la création d'une instance de services système, le démarrage de nouvelles activités ou services nécessitent tous un contexte.

Une description plus détaillée est écrite ici .

Configuration d'un AVD (Android Virtual Device)

TL; DR Il nous permet essentiellement de simuler des appareils réels et de tester nos applications sans appareil réel.

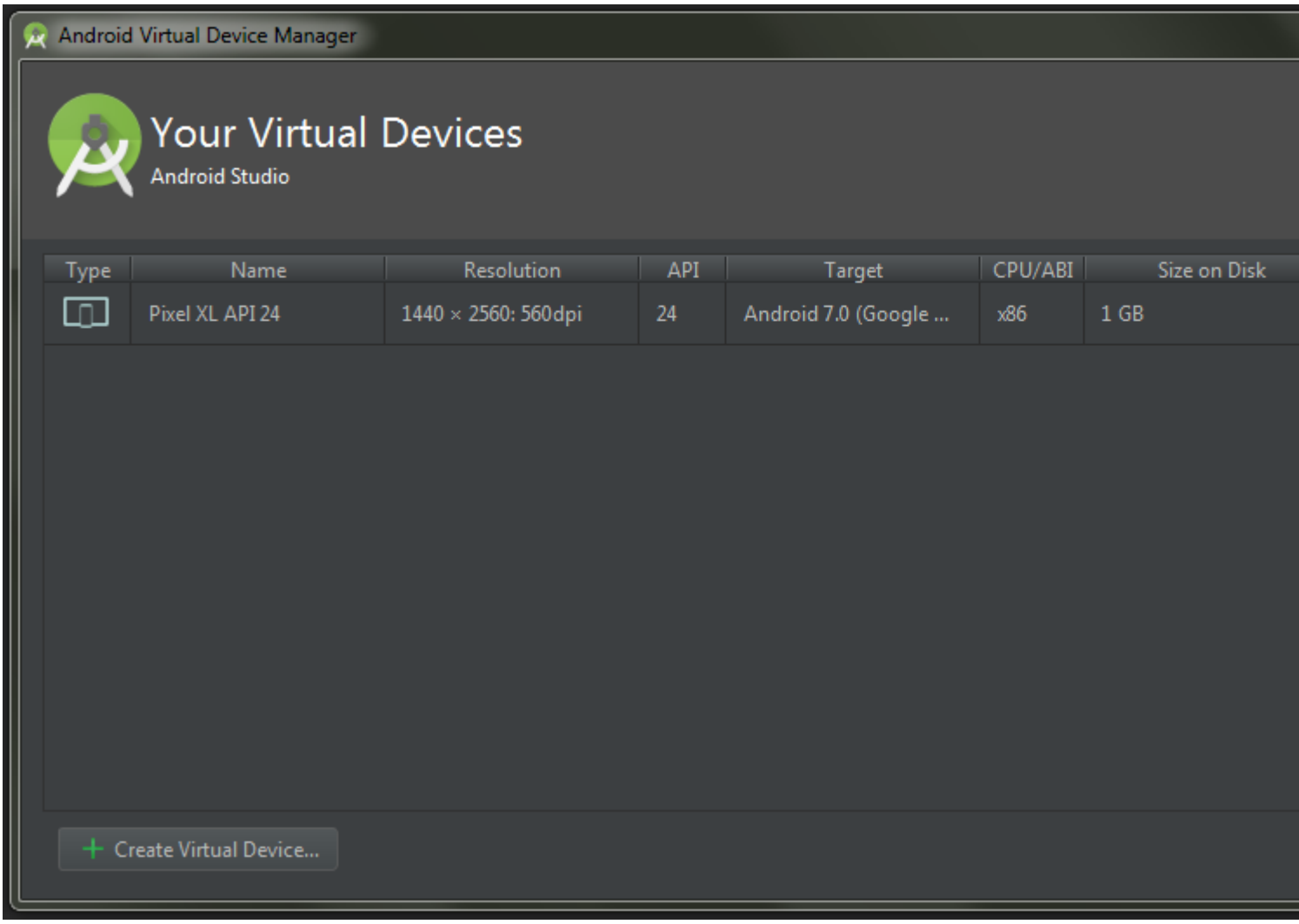
Selon Android Developer Documentation ,

> une *définition de* ***périphérique virtuel Android (AVD)*** vous permet de définir les caractéristiques d'un appareil Android Phone, Tablet, Android Wear ou Android TV que vous souhaitez simuler dans l'émulateur Android. Le gestionnaire AVD vous aide à créer et à gérer facilement des AVD.

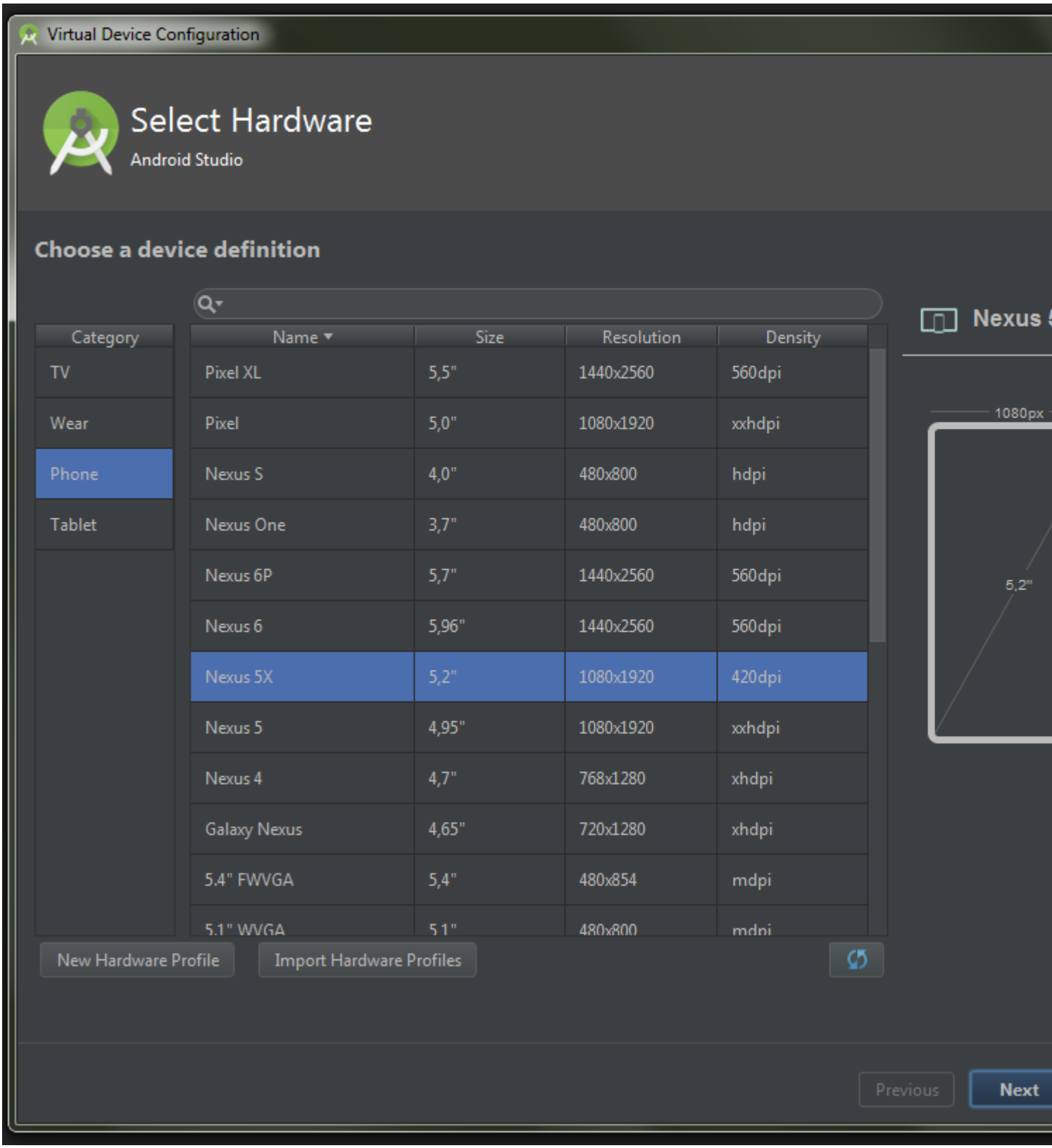
Pour configurer un AVD, procédez comme suit:

1. Cliquez sur ce bouton pour afficher le gestionnaire AVD:

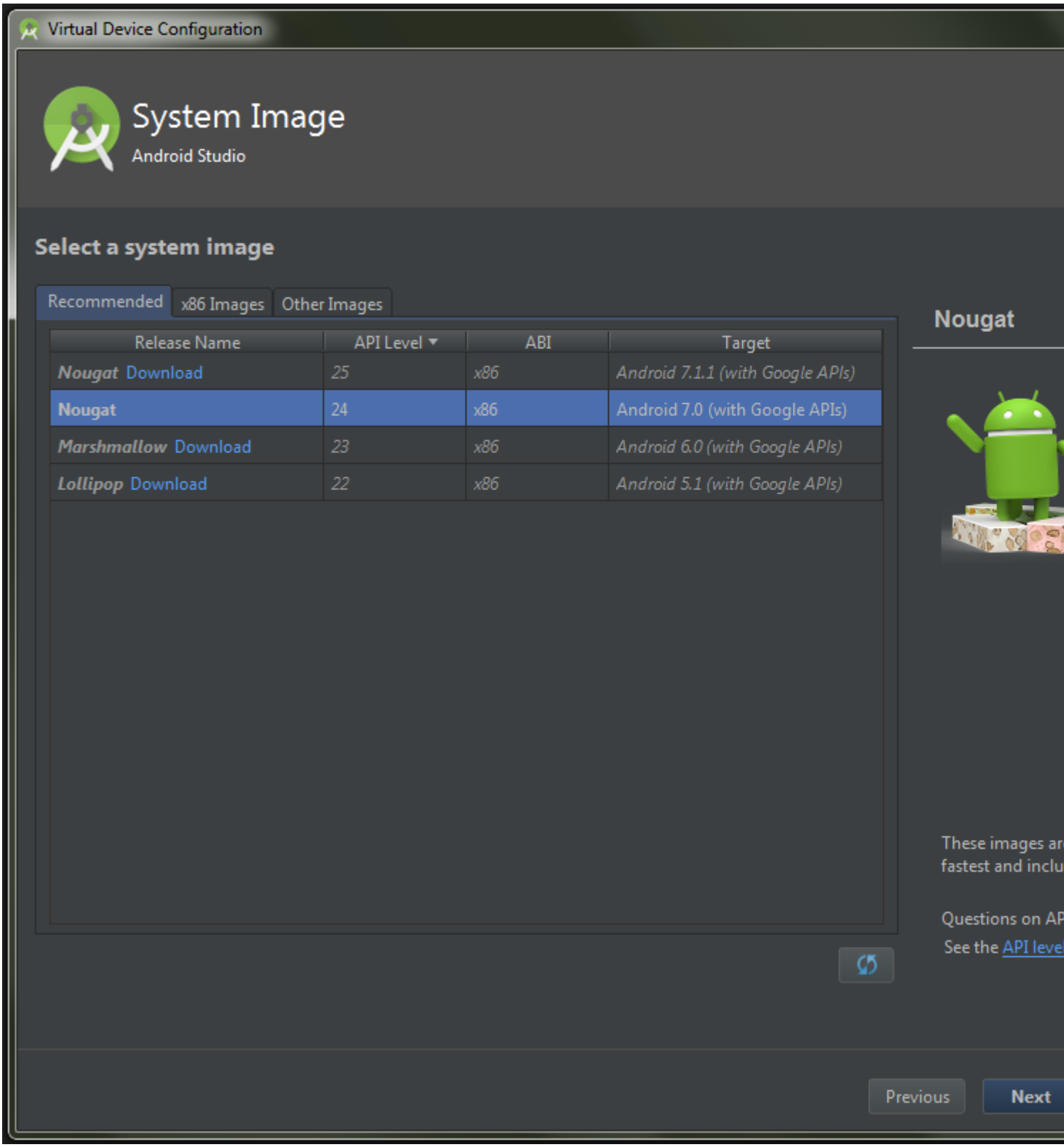
2. Vous devriez voir une boîte de dialogue comme celle-ci:



3. Cliquez maintenant sur le bouton `+ Create Virtual Device...` Cela fera apparaître la boîte de dialogue Virtual Device Configuration:



4. Sélectionnez n'importe quel périphérique de votre choix, puis cliquez sur `Next` :



5. Ici, vous devez choisir une version Android pour votre émulateur. Vous devrez peut-être aussi le télécharger en premier en cliquant sur `Download` . Après avoir choisi une version, cliquez sur `Next` .

6. Entrez ici un nom pour votre émulateur, l'orientation initiale et si vous souhaitez afficher un cadre autour de lui. Après avoir choisi tous ces éléments, cliquez sur `Finish` .

7. Vous avez maintenant un nouveau AVD prêt à lancer vos applications dessus.

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

Lire Démarrer avec Android en ligne: https://riptutorial.com/fr/android/topic/85/demarrer-avec-android

Chapitre 2: Accès aux bases de données SQLite à l'aide de la classe ContentValues

Examples

Insertion et mise à jour de lignes dans une base de données SQLite

Tout d'abord, vous devez ouvrir votre base de données SQLite, ce qui peut être fait comme suit:

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

Après avoir ouvert la base de données, vous pouvez facilement insérer ou mettre à jour des lignes à l'aide de la classe `ContentValues` . Les exemples suivants supposent qu'un prénom est donné par `str_edtfname` et un nom de famille par `str_edtlname` . Vous devez également remplacer `table_name` par le nom de votre table que vous souhaitez modifier.

Insérer des données

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

Mise à jour des données

```
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});
```

Lire Accès aux bases de données SQLite à l'aide de la classe ContentValues en ligne: https://riptutorial.com/fr/android/topic/10154/acces-aux-bases-de-donnees-sqlite-a-l-aide-de-la-classe-contentvalues

Chapitre 3: ACRA

Syntaxe

- android: nom = ". ACRAHandler"
- ACRA.init (this, config);
- la classe publique ACRAHandler étend l'application {

Paramètres

Paramètre	La description
@ReportCrashes	Définit les paramètres ACRA, tels que l'emplacement à signaler, le contenu personnalisé, etc.
formeUri	le chemin du fichier qui signale le plantage

Remarques

- ACRA ne prend plus en charge les formulaires Google, vous avez donc besoin d'un backend: https://github.com/ACRA/acra/wiki/Backends

Examples

ACRAHandler

Exemple de classe d'extension d'application pour gérer le reporting:

```
@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);


    }

}
```

Exemple manifeste

```
<?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'
```

Lire ACRA en ligne: https://riptutorial.com/fr/android/topic/1324/acra

Chapitre 4: Activité

Introduction

Une activité représente un seul écran avec une **interface utilisateur** . Une application Android peut avoir plusieurs activités. Par exemple, une application de messagerie peut avoir une activité pour répertorier tous les e-mails, une autre activité pour afficher le contenu des e-mails, une autre activité pour composer un nouvel e-mail. Toutes les activités d'une application fonctionnent ensemble pour créer une expérience utilisateur parfaite.

Syntaxe

- annuler onCreate (Bundle savedInstanceState) // Appelé au démarrage de l'activité.
- void onPostCreate (Bundle savedInstanceState) // Appelé à la fin du démarrage de l'activité (après onStart () et onRestoreInstanceState (Bundle) ont été appelés).
- void onStart () // Appelé après onCreate (Bundle) - ou après onRestart () lorsque l'activité a été arrêtée, mais est à nouveau affichée pour l'utilisateur.
- void onResume () // Appelé après onRestoreInstanceState (Bundle), onRestart () ou onPause (), pour que votre activité commence à interagir avec l'utilisateur.
- void onPostResume () // Appelé lorsque l'activité est terminée (après l'appel de onResume ()).
- void onRestart () // Appelé après onStop () lorsque l'activité en cours est affichée à nouveau pour l'utilisateur (l'utilisateur y est retourné).
- void onPause () // Appelé dans le cadre du cycle de vie d'une activité lorsqu'une activité est en arrière-plan, mais n'a pas encore été tué.
- void onStop () // Appelé lorsque l'utilisateur n'est plus visible.
- void onDestroy () // Effectue tout nettoyage final avant la destruction d'une activité.
- void onNewIntent (Intention intentionnelle) // Ceci est appelé pour les activités qui définissent launchMode sur "singleTop" dans leur package, ou si un client a utilisé l'indicateur FLAG_ACTIVITY_SINGLE_TOP lors de l'appel de startActivity (Intent).
- annulez onSaveInstanceState (Bundle outState) // Appelé pour récupérer l'état par instance d'une activité avant d'être tué pour que l'état puisse être restauré dans onCreate (Bundle) ou onRestoreInstanceState (Bundle) (le Bundle rempli par cette méthode sera transmis aux deux).
- void onRestoreInstanceState (Bundle savedInstanceState) // Cette méthode est appelée après onStart () lorsque l'activité est en cours de réinitialisation à partir d'un état

précédemment enregistré, donné ici dans savedInstanceState.

Paramètres

Paramètre	Détails
Intention	Peut être utilisé avec startActivity pour lancer une activité
Paquet	Un mappage des clés String vers diverses valeurs parcellables .
Le contexte	Interface vers des informations globales sur un environnement d'application.

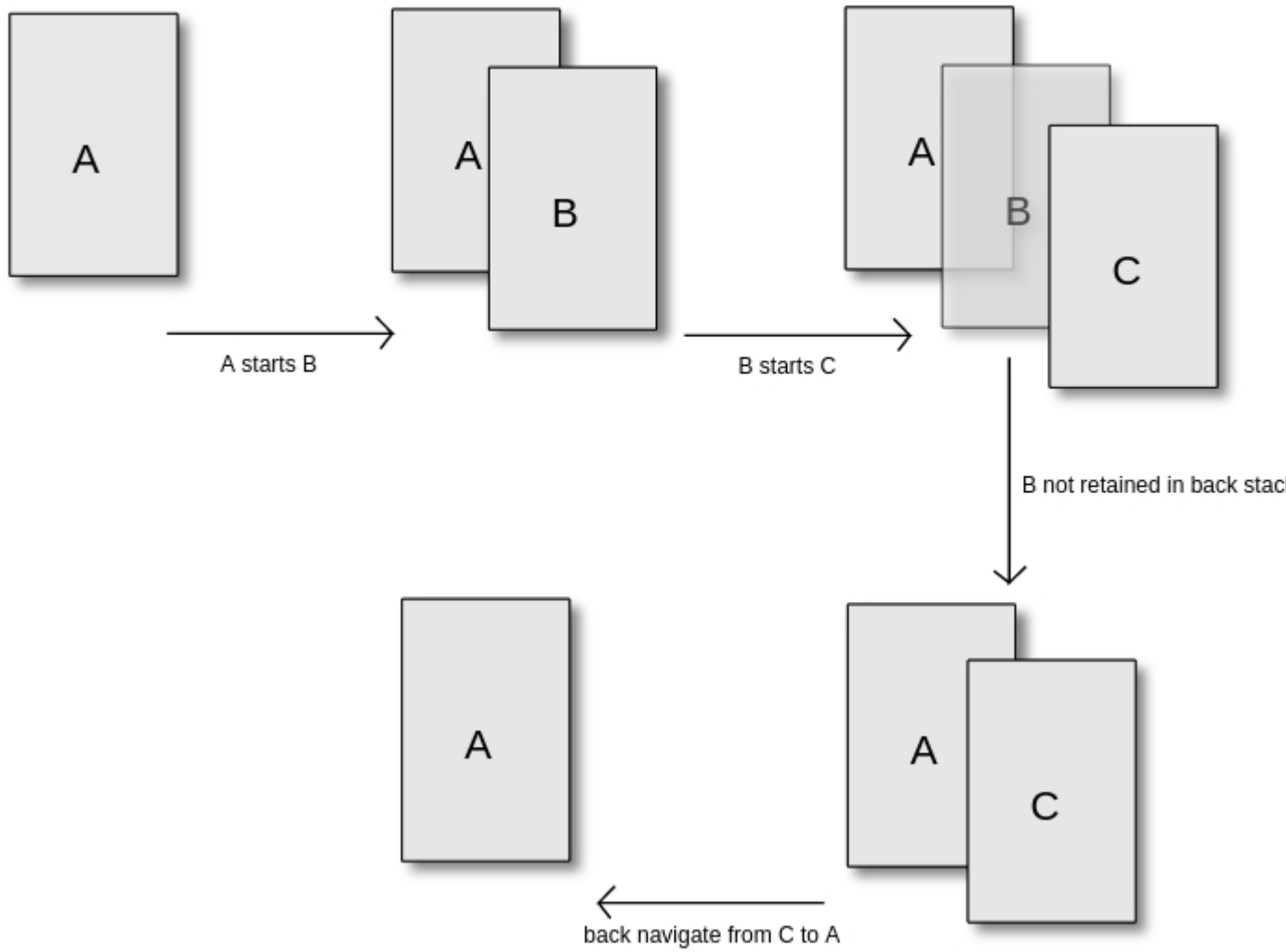
Remarques

Une activité est un composant d'application qui fournit un écran avec lequel les utilisateurs peuvent interagir pour faire quelque chose, comme composer le numéro du téléphone, prendre une photo, envoyer un courrier électronique ou afficher une carte. Chaque activité se voit attribuer une fenêtre dans laquelle dessiner son interface utilisateur. La fenêtre remplit généralement l'écran, mais peut être plus petite que l'écran et flotter au-dessus des autres fenêtres.

Examples

Exclure une activité de l'historique de la pile arrière

Que l'activité `B` puisse être ouverte et que d'autres activités puissent être lancées. Mais l'utilisateur ne devrait pas le rencontrer lors de la navigation dans les activités de tâche.



La solution la plus simple consiste à définir l'attribut `noHistory` sur `true` pour cette `<activity>` dans `AndroidManifest.xml` :

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

Ce même comportement est également possible à partir du code si `B` appelle `finish()` avant de démarrer l'activité suivante:

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

L'utilisation typique du drapeau `noHistory` est avec "Splash Screen" ou Login Activities.

Activité Android Explication de LifeCycle

Supposons une application avec une MainActivity qui puisse appeler l'activité suivante en cliquant

sur un bouton.

```
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);
    } }
```

et

```
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");
    } }
```

Lorsque l'application est créée pour la première fois
D / MainActivity: appel de onCreate à partir de MainActivity
D / MainActivity: appel de onStart depuis MainActivity
D / MainActivity: appel à onResume à partir de MainActivity
sont appelés

Quand l'écran dort
08: 11: 03.142 D / MainActivity: appel de onPause à partir de MainActivity
08: 11: 03.192 D / MainActivity: appel de onStop depuis MainActivity
sont appelés. Et encore quand il se réveille
08: 11: 55.922 D / MainActivity: appel de onRestart à partir de MainActivity
08: 11: 55.962 D / MainActivity: appel de onStart depuis MainActivity
08: 11: 55.962 D / MainActivity: appel de onResume à partir de MainActivity
sont appelés

Cas 1: quand l'activité suivante est appelée à partir de l'activité principale
D / MainActivity: appel à l'activité suivante
D / MainActivity: appel de onPause à partir de MainActivity

D / NextActivity: appeler onCreate à partir de l'activité suivante
D / NextActivity: appel de onStart à partir de l'activité suivante
D / NextActivity: appeler onResume à partir de l'activité suivante
D / MainActivity: appel de onStop depuis MainActivity

Lorsque vous revenez à l'activité principale à partir de l'activité suivante à l'aide du bouton retour
D / NextActivity: appeler onPause à partir de l'activité suivante
D / MainActivity: appeler onRestart à partir de MainActivity
D / MainActivity: appel de onStart depuis MainActivity
D / MainActivity: appel à onResume à partir de MainActivity
D / NextActivity: appeler onStop à partir de l'activité suivante
D / NextActivity: appeler onDestroy à partir de l'activité suivante

Cas 2: Lorsque l'activité est partiellement masquée (lorsque le bouton Vue d'ensemble est enfoncé) ou lorsque l'application passe en arrière-plan et qu'une autre application la masque complètement
D / MainActivity: appel de onPause à partir de MainActivity
D / MainActivity: appel de onStop depuis MainActivity
et lorsque l'application est de retour au premier plan prête à accepter les entrées utilisateur,
D / MainActivity: appeler onRestart à partir de MainActivity
D / MainActivity: appel de onStart depuis MainActivity
D / MainActivity: appel à onResume à partir de MainActivity
sont appelés

Cas 3: Lorsqu'une activité est appelée pour remplir une intention implicite et que l'utilisateur a effectué une sélection. Par exemple, lorsque vous appuyez sur le bouton Partager et que l'utilisateur doit sélectionner une application dans la liste des applications affichées
D / MainActivity: appel de onPause à partir de MainActivity

L'activité est visible mais pas active maintenant. Lorsque la sélection est terminée et que l'application est active
D / MainActivity: appel à onResume à partir de MainActivity
est appelé

Cas4:
Lorsque l'application est tuée en arrière-plan (pour libérer des ressources pour une autre application au premier plan), *onPause* (pour le périphérique pré-en nid d'abeilles) ou *onStop* (pour le périphérique en nid d'abeille) sera le dernier à être appelé avant la fin de l'application.

onCreate et onDestroy seront appelés au maximum une fois à chaque exécution de l'application. Mais onPause, onStop, onRestart, onStart, onResume peuvent être appelés plusieurs fois au cours du cycle de vie.

Mode de lancement d'activité

Le mode de lancement définit le comportement d'une activité nouvelle ou existante dans la tâche. Il existe des modes de lancement possibles:

- la norme
- unique
- singleTask
- seule instance

Il doit être défini dans le manifeste Android dans l'élément `<activity/>` tant `android:launchMode` .

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

La norme:

Valeur par défaut. Si ce mode est défini, une nouvelle activité sera toujours créée pour chaque nouvelle intention. Il est donc possible d'obtenir de nombreuses activités du même type. Une nouvelle activité sera placée en haut de la tâche. Il existe une différence pour les différentes versions d'Android: si l'activité commence à partir d'une autre application, sur les androïdes <= 4.4, elle sera placée dans la même tâche que l'application de démarrage, mais la nouvelle tâche> = 5.0 sera créée.

SingleTop:

Ce mode est presque identique à la `standard` . De nombreuses instances d'activité singleTop peuvent être créées. La différence est que, si une instance d'activité existe déjà en haut de la pile en cours, `onNewIntent()` sera appelé au lieu de créer une nouvelle instance.

SingleTask:

L'activité avec ce mode de lancement ne peut avoir qu'une seule instance **dans le système** . Une nouvelle tâche pour l'activité sera créée, si elle n'existe pas. Sinon, la tâche avec activité sera déplacée vers l'avant et `onNewIntent` sera appelée.

Seule instance:

Ce mode est similaire à `singleTask` . La différence est que la tâche qui détient une activité avec `singleInstance` ne peut avoir que cette activité et rien de plus. Lorsque l'activité d' `singleInstance` crée une autre activité, une nouvelle tâche sera créée pour placer cette activité.

Présentation de l'interface utilisateur avec setContentView

La classe d'activité prend soin de créer une fenêtre dans laquelle vous pouvez placer votre interface utilisateur avec `setContentView` .

Il existe trois méthodes `setContentView` :

- `setContentView(int layoutResID)` - Définit le contenu de l'activité à partir d'une ressource de présentation.
- `setContentView(View view)` - Définit le contenu de l'activité sur une vue explicite.
- `setContentView(View view, ViewGroup.LayoutParams params)` - Définit le contenu de l'activité sur une vue explicite avec les paramètres fournis.

Lorsque `setContentView` est appelée, cette vue est placée directement dans la hiérarchie de vues de l'activité. Elle peut elle-même être une hiérarchie de vues complexe.

Exemples

Définir le contenu du fichier de ressources:

Ajoutez le fichier de ressources (main.xml dans cet exemple) avec la hiérarchie de vue:

```
<?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>
```

Définissez-le comme contenu dans l'activité:

```
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);
    }
}
```

Définir le contenu à une vue explicite:

```
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);
    }
}
```

Effacer votre pile d'activité actuelle et lancer une nouvelle activité

Si vous souhaitez effacer votre pile d'activité actuelle et lancer une nouvelle activité (par exemple, vous déconnecter de l'application et lancer une activité de connexion), il semble qu'il y ait deux approches.

1. Cible (API> = 16)

Appeler `finishAffinity()` partir d'une activité

2. Cible (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();
```

Fin de l'application avec exclure des Récents

Définissez d'abord une ExitActivity dans AndroidManifest.xml

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

Après la classe ExitActivity

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

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

```
        } else {
            finish();
        }
    }

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

Up Navigation pour les activités

La navigation haut s'effectue dans Android en ajoutant `android:parentActivityName=""` dans Manifest.xml à la balise d'activité. Avec cette balise, vous indiquez au système l'activité parentale d'une activité.

Comment est-il fait?

```
<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>
```

Maintenant, quand je clique sur la flèche dans la barre d'outils de HomeActivity, cela me ramène à l'activité parent.

Code Java

Ici, je vais écrire le code Java approprié pour cette fonctionnalité.

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

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

Si vous exécutez ce code, vous verrez que lorsque vous appuyez sur le bouton retour, vous revenez à MainActivity. Pour une meilleure compréhension de Up Navigation, je vous recommande de lire docs

Vous pouvez plus personnaliser ce comportement selon vos besoins en remplaçant

```
@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);
}
```

Piratage simple

C'est un simple hack qui est principalement utilisé pour naviguer vers l'activité parent si le parent est en backstack. En appelant `onBackPressed()` si id est égal à `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);
    }
```

Lire Activité en ligne: https://riptutorial.com/fr/android/topic/1481/activite

Chapitre 5: ADB (Android Debug Bridge)

Introduction

ADB (Android Debug Bridge) est un outil de ligne de commande utilisé pour communiquer avec une instance d'émulateur ou un périphérique Android connecté.

Vue d'ensemble de la BAD

Une grande partie de ce sujet a été divisée en deux parties:

Remarques

Liste des exemples déplacés dans le shell adb :

- Octroi & révocation d'autorisations API 23+
- Envoyer du texte, des touches enfoncées et des événements tactiles au périphérique Android via ADB
- Liste des paquets
- Enregistrement de l'affichage
- Ouvrir les options du développeur
- Définir la date / heure via adb
- Modification des autorisations de fichier à l'aide de la commande chmod
- Génération d'une diffusion "Boot Complete"
- Imprimer les données d'application
- Afficher le contenu de stockage externe / secondaire
- http://stackoverflow.com/documentation/android/9408/adb-shell/29140/adb-shell
- tuer un processus dans un appareil Android

Examples

Imprimer la liste détaillée des appareils connectés

Pour obtenir une liste détaillée de tous les périphériques connectés à `adb` , écrivez la commande suivante dans votre terminal:

```
adb devices -l
```

Exemple de sortie

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

```
A50PL                  device usb:331592812X
```

- La première colonne est le numéro de série de l'appareil. Si elle commence par `emulator-` , cet appareil est un émulateur.
- `usb:` le chemin du périphérique dans le sous-système USB.
- `product:` le code produit de l'appareil. Ceci est très spécifique au fabricant et, comme vous pouvez le voir dans le cas de l'appareil Archos `A50PL` ci-dessus, il peut être vide.
- `model:` le modèle d'appareil. Comme le `product` , peut être vide.
- `device:` le code de l'appareil. Ceci est également très spécifique au fabricant et peut être vide.

Lire les informations sur l'appareil

Écrivez la commande suivante dans votre terminal:

```
adb shell getprop
```

Cela imprimera toutes les informations disponibles sous la forme de paires clé / valeur.

Vous pouvez simplement lire des informations spécifiques en ajoutant le nom d'une clé spécifique à la commande. Par exemple:

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

Voici quelques informations intéressantes que vous obtenez:

- `ro.product.model` : Nom du modèle du périphérique (par exemple, Nexus 6P)
- `ro.build.version.sdk` : Niveau API du périphérique (par exemple 23)
- `ro.product.brand` : `ro.product.brand` marque de l'appareil (par ex. Samsung)

Exemple complet de sortie

```
[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]
```

Connecter ADB à un périphérique via WiFi

La configuration ADB standard implique une connexion USB à un périphérique physique.
Si vous préférez, vous pouvez passer en mode TCP / IP et connecter ADB via WiFi.

Appareil non enraciné

1. **Obtenez sur le même réseau:**

    - Assurez-vous que votre appareil et votre ordinateur sont sur le même réseau.

2. **Connectez le périphérique à l'ordinateur hôte à l'aide d'un câble USB.**

3. **Connectez `adb` à un périphérique via le réseau:**

    Alors que votre appareil est connecté à `adb` via USB, `adb` la commande suivante pour écouter une connexion TCP / IP sur un port (par défaut 5555):

    - Tapez `adb tcpip <port>` (passez en mode TCP / IP).
    - Déconnectez le câble USB du périphérique cible.
    - Tapez `adb connect <ip address>:<port>` (le port est facultatif; par défaut 5555).

    Par exemple:

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

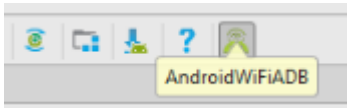
    Si vous ne connaissez pas l'adresse IP de votre appareil, vous pouvez:

    - vérifiez l'adresse IP dans les paramètres WiFi de votre appareil.
    - utiliser ADB pour découvrir l'IP (via USB):
        1. Connectez l'appareil à l'ordinateur via USB
        2. Dans une ligne de commande, tapez `adb shell ifconfig` et copiez l'adresse IP de votre périphérique

Pour **revenir** au débogage via **USB,** utilisez la commande suivante:

```
adb usb
```

Vous pouvez également connecter ADB via WiFi en installant un plug-in sur Android Studio. Pour ce faire, accédez à *Paramètres> Plug* -ins et Parcourir les référentiels, recherchez *ADB WiFi* , installez-le et rouvrez Android Studio. Vous verrez une nouvelle icône dans votre barre d'outils, comme indiqué dans l'image suivante. Connectez l'appareil à l'ordinateur hôte via USB et cliquez sur cette icône *AndroidWiFiADB* . Il affichera un message indiquant si votre appareil est connecté ou non. Une fois connecté, vous pouvez débrancher votre clé USB.



Appareil enraciné

Remarque: Certains appareils qui **sont enracinés** peuvent utiliser l'App WiFi ADB du Play Store pour activer cela de manière simple. En outre, pour certains périphériques (en particulier ceux dotés de ROM CyanogenMod), cette option est présente dans les options de développement parmi les paramètres. L'activer vous donnera l'adresse IP et le numéro de port requis pour vous connecter à `adb` en exécutant simplement `adb connect <ip address>:<port>` .

Lorsque vous avez un périphérique rooté mais que vous n'avez pas accès à un câble USB

Le processus est expliqué en détail dans la réponse suivante: http://stackoverflow.com/questions/2604727/how-can-i-connect-to-android-with-adb-over-tcp/3623727#3623727 Les commandes les plus importantes sont montrés ci-dessous.

Ouvrez un terminal dans l'appareil et tapez ce qui suit:

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

Par exemple:

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

Et sur votre ordinateur:

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

Par exemple:

```
adb connect 192.168.1.2:5555
```

Pour l'éteindre:

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

Éviter le délai d'attente

Par défaut, `adb` après 5000 ms. Cela peut arriver dans certains cas, comme le WiFi lent ou un grand APK.

Un simple changement dans la configuration de Gradle peut faire l'affaire:

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

Extraire (pousser) des fichiers de (vers) l'appareil

Vous pouvez extraire (télécharger) des fichiers du périphérique en exécutant la commande suivante:

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

Par exemple:

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

Vous pouvez également pousser (télécharger) des fichiers de votre ordinateur vers le périphérique:

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

Par exemple:

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

Exemple pour récupérer une base de données depuis un périphérique

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

Redémarrer le périphérique

Vous pouvez redémarrer votre appareil en exécutant la commande suivante:

```
adb reboot
```

Effectuez cette commande pour redémarrer dans bootloader:

```
adb reboot bootloader
```

Redémarrez en mode de récupération:

```
adb reboot recovery
```

Sachez que l'appareil ne s'arrêtera pas en premier!

Activer / désactiver le Wifi

Allumer:

```
adb shell svc wifi enable
```

Éteindre:

```
adb shell svc wifi disable
```

Voir les appareils disponibles

Commander:

```
 adb devices
```

Exemple de résultat:

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

Première colonne - numéro de série de l'appareil

Deuxième colonne - état de la connexion

Documentation Android

Connecter le périphérique par IP

Entrer les commandes dans les applications dispositif terminal

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

Après cela, vous pouvez utiliser **CMD** et **ADB** pour vous connecter en utilisant la commande suivante

```
adb connect 192.168.0.101.5555
```

Et vous pouvez le désactiver et renvoyer ADB sur USB avec

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

Depuis un ordinateur, si vous avez déjà un accès USB (aucune racine requise)

Il est encore plus facile de passer à l'utilisation du Wi-Fi si vous avez déjà une connexion USB. À partir d'une ligne de commande sur l'ordinateur sur lequel le périphérique est connecté via USB, émettez les commandes

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

Remplacez 192.168.0.101 par l'adresse IP du périphérique

Démarrer / arrêter adb

Démarrer ADB:

```
adb kill-server
```

Arrêtez ADB:

```
adb start-server
```

Voir logcat

Vous pouvez exécuter `logcat` tant que commande adb ou directement à l'invite du shell de votre émulateur ou périphérique connecté. Pour afficher la sortie du journal à l'aide d' `adb` , accédez à votre répertoire platform-tools / toolkit SDK et exécutez:

```
$ adb logcat
```

Vous pouvez également créer une connexion shell à un périphérique puis exécuter:

```
$ adb shell
```

```
$ logcat
```

Une commande utile est:

```
adb logcat -v threadtime
```

Cela affiche la date, l'heure d'appel, la priorité, la balise et le PID et le TID du thread émettant le message dans un format de message long.

Filtration

Les journaux Logcat ont été appelés niveaux de journalisation:

> **V** - Verbose, **D** - Debug, **I** - Info, **W** - Avertissement, **E** - Erreur, **F** - Fatal, **S** - Silencieux

Vous pouvez également filtrer logcat par niveau de journalisation. Par exemple, si vous souhaitez uniquement afficher le niveau de débogage:

```
adb logcat *:D
```

Logcat peut être filtré par un nom de package, bien sûr, vous pouvez le combiner avec le filtre de niveau de journalisation:

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

Vous pouvez également filtrer le journal à l'aide de grep (plus d'informations sur le filtrage des résultats de logcat ici):

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

Dans Windows, le filtre peut être utilisé avec findstr, par exemple:

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

Pour afficher un autre tampon de journal [main | events | radio], exécutez le `logcat` avec l'option `-b` :

```
adb logcat -b radio
```

Enregistrer la sortie dans le fichier:

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

Enregistrez la sortie dans le fichier tout en le regardant:

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

Nettoyage des journaux:

```
adb logcat -c
```

Commande ADB directe vers un périphérique spécifique dans un paramètre multi-périphérique

1. Ciblez un appareil par numéro de série

Utilisez l'option `-s` suivie d'un nom de périphérique pour sélectionner le périphérique sur lequel la commande `adb` doit s'exécuter. Les options `-s` doivent être en première ligne, **avant** la commande.

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

Exemple:

```
adb devices

List of devices attached
emulator-5554       device
02157df2d1faeb33    device

adb -s emulator-5554 shell
```

Exemple # 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. Ciblez un périphérique lorsque seul un type de périphérique est connecté

Vous pouvez cibler le seul émulateur en cours d'exécution avec -e

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

Ou vous pouvez cibler le seul périphérique USB connecté avec -d

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

Prendre une capture d'écran et une vidéo (pour kitkat uniquement) à partir d'un écran de périphérique

Capture d'écran: Option 1 (pure adb)

La commande `shell` adb nous permet d'exécuter des commandes à l'aide du shell intégré d'un périphérique. La commande shell `screencap` capture le contenu actuellement visible sur un périphérique et l'enregistre dans un fichier image donné, par exemple `/sdcard/screen.png` :

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

Vous pouvez ensuite utiliser la commande pull pour télécharger le fichier depuis le périphérique dans le répertoire actuel de votre ordinateur:

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

Capture d'écran: Option 2 (plus rapide)

Exécutez le one-liner suivant:

(Guimauve et plus tôt):

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

(Nougat et plus tard):

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

L' `-p` redirige la sortie de la commande `screencap` vers stdout. L'expression Perl dans laquelle elle est dirigée nettoie certains problèmes de fin de ligne sur Marshmallow et les versions antérieures. Le flux est ensuite écrit dans un fichier nommé `screen.png` dans le répertoire en cours. Voir cet article et cet article pour plus d'informations.

Vidéo

cela ne fonctionne que dans KitKat et via ADB uniquement. Cela ne fonctionne pas sous Kitkat Pour commencer à enregistrer l'écran de votre appareil, exécutez la commande suivante:

`adb shell screenrecord /sdcard/example.mp4` , Cette commande lance l'enregistrement de l'écran de votre appareil en utilisant les paramètres par défaut et enregistre la vidéo résultante dans un fichier dans le fichier `/sdcard/example.mp4` de votre appareil.

Une fois l'enregistrement terminé, appuyez sur Ctrl + C (z sous Linux) dans la fenêtre d'invite de commandes pour arrêter l'enregistrement à l'écran. Vous pouvez ensuite trouver le fichier d'enregistrement d'écran à l'emplacement que vous avez spécifié. Notez que l'enregistrement d'écran est enregistré dans la mémoire interne de votre appareil et non sur votre ordinateur.

Les paramètres par défaut consistent à utiliser la résolution d'écran standard de votre appareil, à encoder la vidéo à un débit de 4 Mbits / s et à définir la durée maximale d'enregistrement sur l'écran à 180 secondes. Pour plus d'informations sur les options de ligne de commande que vous pouvez utiliser, exécutez la commande suivante:

`adb shell screenrecord -help` , Cela fonctionne sans rooter le périphérique. J'espère que cela t'aides.

Effacer les données d'application

On peut effacer les données d'utilisateur d'une application spécifique en utilisant `adb` :

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

C'est la même chose que pour parcourir les paramètres du téléphone, sélectionnez l'application et appuyez sur le bouton Effacer les données.

- `pm` appelle le gestionnaire de paquets sur l'appareil
- `clear` supprime toutes les données associées à un package

Envoi de diffusion

Il est possible d'envoyer la diffusion à `BroadcastReceiver` avec `adb` .

Dans cet exemple, nous envoyons la diffusion avec l'action `com.test.app.ACTION` et la chaîne extra dans le bundle `'foo'='bar'` :

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

Vous pouvez mettre tout autre type pris en charge pour regrouper, pas seulement les chaînes:

--ez - booléen
--ei - entier
--el - long
--ef - float
--eu - uri
--eia - int array (séparé par ',')
--ela - tableau long (séparé par ',')
--efa - tableau flottant (séparé par ',')
--esa - tableau de chaînes (séparé par ',')

Pour envoyer l'intention à un paquet / classe spécifique, le paramètre `-n` ou `-p` peut être utilisé. Envoi à forfait:

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

Envoi d'un composant spécifique (`SomeReceiver` de classe en `com.test.app package`):

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

Exemples utiles:

- Envoi d'une diffusion "boot complete"
- Envoi d'une diffusion "heure changée" après mise à l'heure via la commande adb

Installer et exécuter une application

Pour installer un fichier APK , utilisez la commande suivante:

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

ou si l'application existe et que nous souhaitons la réinstaller

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

Pour désinstaller une application , il faut spécifier son package

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

Utilisez la commande suivante pour démarrer une application avec un nom de package fourni (ou une activité spécifique dans une application):

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

Par exemple, pour démarrer Waze:

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

Sauvegarde

Vous pouvez utiliser la commande de `adb backup` pour sauvegarder votre appareil.

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

`-f <filename>` spécifie le nom de fichier **par défaut:** *crée une sauvegarde.ab dans le répertoire en cours*

`-apk|noapk` activer / désactiver la sauvegarde de .apks eux-mêmes **par défaut:** *-noapk*

`-obb|noobb` activer / désactiver la sauvegarde des fichiers supplémentaires **par défaut:** *-noobb*

`-shared|noshared` partagé / contenu de la carte SD du périphérique de sauvegarde `-shared|noshared` **par défaut:** *-noshared*

`-all` sauvegarde tous les apllications installés

`-system|nosystem` inclut les applications système **par défaut:** *-system*

`<packages>` une liste de paquets à sauvegarder (ex: com.example.android.myapp) (non nécessaire si `-all` est spécifié)

Pour une sauvegarde complète du périphérique, y compris tout, utilisez

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

> **Remarque:** La sauvegarde complète peut prendre beaucoup de temps.

Pour restaurer une sauvegarde, utilisez

```
adb restore backup.ab
```

Installer ADB sur un système Linux

Comment installer Android Debugging Bridge (ADB) sur un système Linux avec le terminal utilisant les référentiels de votre distribution.

Installer sur le système Ubuntu / Debian via apt:

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

Installer sur le système Fedora / CentOS via yum:

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

Installer sur le système Gentoo avec portage:

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

Installer sur le système openSUSE avec zypper:

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

Installer sur le système Arch avec pacman:

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

Énumérer toutes les autorisations qui nécessitent une subvention d'exécution des utilisateurs sur Android 6.0

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

Afficher les données internes d'une application (données / données /) sur un appareil

Tout d'abord, assurez-vous que votre application peut être sauvegardée dans `AndroidManifest.xml` , à savoir `android:allowBackup` n'est pas `false` .

Commande de sauvegarde:

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

Créez un tar avec la commande dd:

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

Extraire le goudron:

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

Vous pouvez ensuite afficher le contenu extrait.

Afficher la pile d'activités

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

Très utile lorsqu'il est utilisé avec la commande `watch` unix:

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

Afficher et extraire les fichiers de cache d'une application

Vous pouvez utiliser cette commande pour répertorier les fichiers de votre propre apk déboguable:

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

Et ce script pour tirer du cache, cela copie le contenu à sdcard d'abord, le tire puis le supprime à la fin:

```
#!/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'"
```

Ensuite, vous pouvez extraire un fichier du cache comme ceci:

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

Obtenir le fichier de base de données via ADB

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

Lire ADB (Android Debug Bridge) en ligne: https://riptutorial.com/fr/android/topic/1051/adb--android-debug-bridge-

Chapitre 6: AdMob

Syntaxe

- compiler 'com.google.firebase: annonces firebase: 10.2.1' // REMARQUE: DÉFINIR LA NOUVELLE VERSION SI DISPONIBLE
- `<uses-permission android:name="android.permission.INTERNET" />` Obligatoire pour récupérer l'annonce
- AdRequest adRequest = new AdRequest.Builder (). Build (); // Bannière publicitaire
- AdView mAdView = (AdView) findViewById (R.id.adView) // Annonce publicitaire
- mAdView.loadAd (adRequest); // Bannière publicitaire

Paramètres

Param	Détails
annonces: adUnitId = "@ string / main_screen_ad"	L'ID de votre annonce. Obtenez votre identifiant sur le site admob. *"Bien que cela ne soit pas obligatoire, le stockage de vos valeurs d'identifiant de bloc d'annonces dans un fichier de ressources est une bonne pratique. Au fur et à mesure que votre application se développe fichier, vous ne devez jamais chercher dans votre code à leur recherche. "* [1]

Remarques

- Nécessite un compte Admob valide
- Lisez la politique d'admob . Assurez-vous de ne rien faire qui puisse suspendre votre compte d'admob

Examples

Exécution

Remarque: Cet exemple nécessite un compte Admob valide et un code d'annonce Admob valide.

Build.gradle au niveau de l'application

Passez à la dernière version si existante:

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

Manifeste

L'autorisation Internet est requise pour accéder aux données d'annonce. Notez qu'il n'est pas nécessaire de demander cette autorisation (en utilisant API 23+) car il s'agit d'une autorisation normale et non dangereuse:

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

XML

L'exemple XML suivant montre une bannière publicitaire:

```
<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" />
```

Pour le code d'autres types, reportez-vous à l' aide de Google AdMob .

Java

Le code suivant concerne l'intégration de bannières publicitaires. Notez que d'autres types d'annonces peuvent nécessiter une intégration différente:

```
// 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);
```

Ajoutez les méthodes de cycle de vie `AdView` dans les méthodes `onResume()` , `onPause()` et `onDestroy()` de votre activité:

```
@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();
}
```

Lire AdMob en ligne: https://riptutorial.com/fr/android/topic/5334/admob

Chapitre 7: Affichage

Introduction

Tout ce qui concerne la personnalisation de TextView dans le SDK Android

Syntaxe

- TextView (contexte contextuel)
- (TextView) findViewById (int id)
- void setText (int resid)
- void setText (texte CharSequence) // Vous pouvez utiliser String comme argument

Remarques

Essayez de l'utiliser dans la conception XML ou par programme.

Examples

Textview avec une taille de texte différente

Vous pouvez archiver différents Textsizes dans une Textview avec un Span

```
TextView textView = (TextView) findViewById(R.id.textView);
Spannable span = new SpannableString(textView.getText());
span.setSpan(new RelativeSizeSpan(0.8f), start, end, Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);
textView.setText(span)
```

Personnalisation de TextView

```
public class CustomTextView extends TextView {

    private float strokeWidth;
    private Integer strokeColor;
    private Paint.Join strokeJoin;
    private float strokeMiter;


    public CustomTextView(Context context) {
        super(context);
        init(null);
    }

    public CustomTextView(Context context, AttributeSet attrs) {
        super(context, attrs);
        init(attrs);
    }
```

```
    public CustomTextView(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
        init(attrs);
    }

    public void init(AttributeSet attrs) {

        if (attrs != null) {
            TypedArray a = getContext().obtainStyledAttributes(attrs,
R.styleable.CustomTextView);

            if (a.hasValue(R.styleable.CustomTextView_strokeColor)) {
                float strokeWidth =
a.getDimensionPixelSize(R.styleable.CustomTextView_strokeWidth, 1);
                int strokeColor = a.getColor(R.styleable.CustomTextView_strokeColor,
0xff000000);
                float strokeMiter =
a.getDimensionPixelSize(R.styleable.CustomTextView_strokeMiter, 10);
                Paint.Join strokeJoin = null;
                switch (a.getInt(R.styleable.CustomTextView_strokeJoinStyle, 0)) {
                    case (0):
                        strokeJoin = Paint.Join.MITER;
                        break;
                    case (1):
                        strokeJoin = Paint.Join.BEVEL;
                        break;
                    case (2):
                        strokeJoin = Paint.Join.ROUND;
                        break;
                }
                this.setStroke(strokeWidth, strokeColor, strokeJoin, strokeMiter);
            }
        }
    }

    public void setStroke(float width, int color, Paint.Join join, float miter) {
        strokeWidth = width;
        strokeColor = color;
        strokeJoin = join;
        strokeMiter = miter;
    }

    @Override
    public void onDraw(Canvas canvas) {
        super.onDraw(canvas);

        int restoreColor = this.getCurrentTextColor();
        if (strokeColor != null) {
            TextPaint paint = this.getPaint();
            paint.setStyle(Paint.Style.STROKE);
            paint.setStrokeJoin(strokeJoin);
            paint.setStrokeMiter(strokeMiter);
            this.setTextColor(strokeColor);
            paint.setStrokeWidth(strokeWidth);
            super.onDraw(canvas);
            paint.setStyle(Paint.Style.FILL);
            this.setTextColor(restoreColor);
        }
    }
}
```

Usage:

```
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);
    }
}
```

Disposition:

```
<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>
```

Utilisation par programme:

```
CustomTextView mtxt_name = (CustomTextView) findViewById(R.id.pager_title);
//then use
setStroke(float width, int color, Paint.Join join, float miter);
//method before setting
setText("Sample Text");
```

TextView Spannable

Un TextView pouvant être `TextView` peut être utilisé dans Android pour mettre en évidence une partie particulière du texte avec une couleur, un style, une taille et / ou un événement de clic différents dans un seul widget `TextView` .

Considérer que vous avez défini un `TextView` comme suit:

```
TextView textview=findViewById(R.id.textview);
```

Ensuite, vous pouvez appliquer différentes mises en évidence comme indiqué ci-dessous:

- **Couleur Spanable:** pour définir une couleur différente sur une partie du texte, vous pouvez utiliser `ForegroundColorSpan` , comme illustré dans l'exemple suivant:

  ```
  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 );
  ```

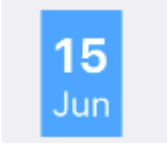
  Sortie créée par le code ci-dessus:

- **Police Spanable:** pour définir une taille de police différente sur une partie du texte, vous pouvez utiliser `RelativeSizeSpan` , comme illustré dans l'exemple suivant:

  ```
  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 );
  ```

  Sortie créée par le code ci-dessus:



- **Police de caractères Spanable:** pour définir une police de caractères différente sur une partie du texte, vous pouvez utiliser un `TypefaceSpan` personnalisé, comme illustré dans l'exemple suivant:

```
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 );
```

Cependant, pour que le code ci-dessus fonctionne, la classe `CustomTypefaceSpan` doit être dérivée de la classe `TypefaceSpan` . Cela peut être fait comme suit:

```
public class CustomTypefaceSpan extends TypefaceSpan {
    private final Typeface newType;

    public CustomTypefaceSpan(String family, Typeface type) {
        super(family);
        newType = type;
    }

    @Override
    public void updateDrawState(TextPaint ds) {
        applyCustomTypeFace(ds, newType);
    }

    @Override
    public void updateMeasureState(TextPaint paint) {
        applyCustomTypeFace(paint, newType);
    }

    private static void applyCustomTypeFace(Paint paint, Typeface tf) {
        int oldStyle;
        Typeface old = paint.getTypeface();
        if (old == null) {
            oldStyle = 0;
        } else {
            oldStyle = old.getStyle();
        }
        int fake = oldStyle & ~tf.getStyle();
        if ((fake & Typeface.BOLD) != 0) {
            paint.setFakeBoldText(true);
        }

        if ((fake & Typeface.ITALIC) != 0) {
            paint.setTextSkewX(-0.25f);
        }

        paint.setTypeface(tf);
    }
}
```

TextView avec image

Android permet aux programmeurs de placer des images aux quatre coins d'un `TextView` . Par exemple, si vous créez un champ avec un objet `TextView` et que vous souhaitez montrer que le champ est modifiable, les développeurs placent généralement une icône d'édition à proximité de ce champ. Android nous offre une option intéressante appelée **composé extractible** pour un `TextView` :

```
<TextView
        android:id="@+id/title"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:drawablePadding="4dp"
        android:drawableRight="@drawable/edit"
        android:text="Hello world"
        android:textSize="18dp" />
```

Vous pouvez définir le dessin sur n'importe quel côté de votre `TextView` comme suit:

```
android:drawableLeft="@drawable/edit"
android:drawableRight="@drawable/edit"
android:drawableTop="@drawable/edit"
android:drawableBottom="@drawable/edit"
```

Le paramétrage du dessin peut également être réalisé par programmation de la manière suivante:

```
yourTextView.setCompoundDrawables(leftDrawable, rightDrawable, topDrawable, bottomDrawable);
```

La définition de l'un des paramètres remis à `setCompoundDrawables()` sur `null` supprimera l'icône du côté correspondant de `TextView` .

Texte barré

Barré le texte entier

```
String sampleText = "This is a test strike";
textView.setPaintFlags(tv.getPaintFlags()| Paint.STRIKE_THRU_TEXT_FLAG);
textView.setText(sampleText);
```

Output: ~~Ceci est un test de frappe~~

Barré seulement des parties du texte

```
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);
```

Output: ~~Ceci~~ est un test de frappe

Personnalisation du thème et du style

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>
```

Rendre RelativeSizeSpan aligné sur le haut

Pour rendre un `RelativeSizeSpan` aligné sur le haut, une classe personnalisée peut être dérivée de la classe `SuperscriptSpan` . Dans l'exemple suivant, la classe dérivée est nommée `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" />
```

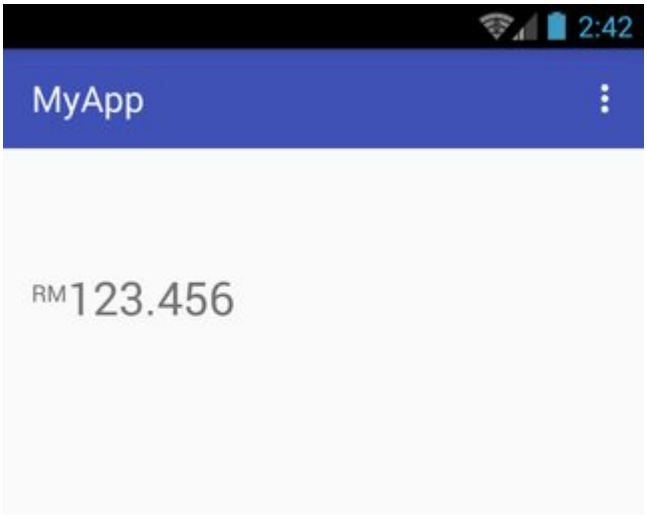
MainActivity.java:

```
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 );
    }
}
```

Capture d'écran de référence:



Pinchzoom sur TextView

activity_main.xml :

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

    <TextView
        android:id="@+id/mytv"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        android:text="This is my sample text for pinch zoom demo, you can zoom in and out
using pinch zoom, thanks" />

</RelativeLayout>
```

MainActivity.java :

```
import android.app.Activity;
import android.os.Bundle;
import android.view.MotionEvent;
import android.view.View;
import android.view.View.OnTouchListener;
import android.widget.TextView;

public class MyTextViewPinchZoomClass extends Activity implements OnTouchListener {

    final static float STEP = 200;
    TextView mytv;
    float mRatio = 1.0f;
    int mBaseDist;
    float mBaseRatio;
    float fontsize = 13;

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        mytv = (TextView) findViewById(R.id.mytv);
        mytv.setTextSize(mRatio + 13);
    }

    public boolean onTouchEvent(MotionEvent event) {
        if (event.getPointerCount() == 2) {
            int action = event.getAction();
            int pureaction = action & MotionEvent.ACTION_MASK;
            if (pureaction == MotionEvent.ACTION_POINTER_DOWN) {
                mBaseDist = getDistance(event);
                mBaseRatio = mRatio;
            } else {
                float delta = (getDistance(event) - mBaseDist) / STEP;
                float multi = (float) Math.pow(2, delta);
                mRatio = Math.min(1024.0f, Math.max(0.1f, mBaseRatio * multi));
```

```
                mytv.setTextSize(mRatio + 13);
            }
        }
        return true;
    }

    int getDistance(MotionEvent event) {
        int dx = (int) (event.getX(0) - event.getX(1));
        int dy = (int) (event.getY(0) - event.getY(1));
        return (int) (Math.sqrt(dx * dx + dy * dy));
    }

    public boolean onTouch(View v, MotionEvent event) {
        return false;
    }
}
```

Single TextView avec deux couleurs différentes

Le texte coloré peut être créé en transmettant le texte et un nom de couleur de police à la fonction suivante:

```
private String getColoredSpanned(String text, String color) {
    String input = "<font color=" + color + ">" + text + "</font>";
    return input;
}
```

Le texte en couleur peut alors être défini sur un `TextView` (ou même sur un `Button` , `EditText` , etc.) en utilisant l'exemple de code ci-dessous.

Tout d'abord, définissez un `TextView` comme suit:

```
TextView txtView = (TextView)findViewById(R.id.txtView);
```

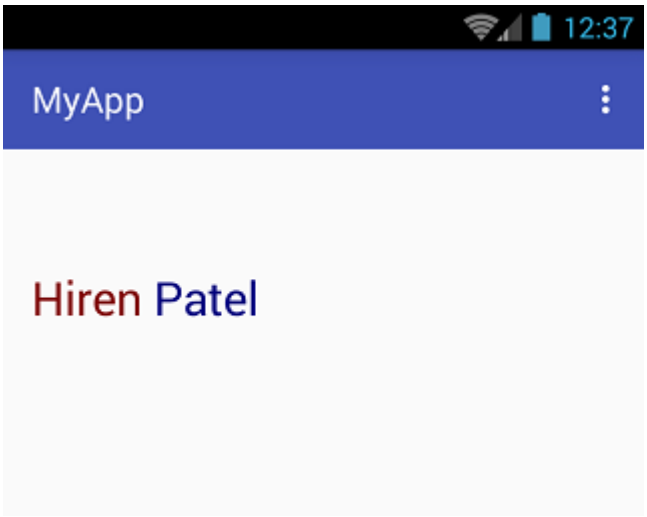
Ensuite, créez un texte de couleur différente et assignez-le aux chaînes:

```
String name = getColoredSpanned("Hiren", "#800000");
String surName = getColoredSpanned("Patel","#000080");
```

Enfin, définissez les deux chaînes de couleurs différentes sur le `TextView` :

```
txtView.setText(Html.fromHtml(name+" "+surName));
```

Capture d'écran de référence:



Lire Affichage en ligne: https://riptutorial.com/fr/android/topic/4212/affichage

Chapitre 8: Affichage des annonces Google

Examples

Configuration de base de l'annonce

Vous devrez ajouter les éléments suivants à vos dépendances:

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

puis mettez ceci dans le même fichier.

```
apply plugin: 'com.google.gms.google-services'
```

Ensuite, vous devrez ajouter des informations pertinentes dans votre fichier strings.xml.

```
<string name="banner_ad_unit_id">ca-app-pub-####/####</string>
```

Ensuite, placez un avis là où vous le souhaitez et modifiez-le comme n'importe quelle autre vue.

```
    <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>
```

Et last but not least, lancez ceci dans votre onCreate.

```
MobileAds.initialize(getApplicationContext(), "ca-app-pub-YOUR_ID");
AdView mAdView = (AdView) findViewById(R.id.adView);
    AdRequest adRequest = new AdRequest.Builder().build();
    mAdView.loadAd(adRequest);
```

Si vous copiez-collé exactement, vous devriez maintenant avoir une petite bannière publicitaire. Placez simplement plus de vues AdViews partout où vous en avez besoin pour plus.

Ajouter une annonce interstitielle

Les annonces interstitielles sont des annonces en plein écran qui couvrent l'interface de leur application hôte. Ils sont généralement affichés à des points de transition naturels dans le flux d'une application, par exemple entre les activités ou pendant la pause entre les niveaux d'un jeu.

Assurez-vous de disposer des autorisations nécessaires dans votre fichier `Manifest` :

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

1. Accédez à votre compte AdMob .
2. Cliquez sur l'onglet **Monétiser** .
3. Sélectionnez ou créez l'application et choisissez la plate-forme.
4. Sélectionnez Interstitial et indiquez un nom de bloc d'annonces.
5. Une fois le bloc d'annonces créé, vous pouvez remarquer l'ID du bloc d'annonces sur le tableau de bord. Par exemple: ca-app-pub-00000000000/000000000
6. Ajouter des dépendances

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

Celui-ci devrait être en bas.

```
apply plugin: 'com.google.gms.google-services'
```

Ajoutez votre **identifiant de bloc d'annonces** à votre `strings.xml`

```
<string name="interstitial_full_screen">ca-app-pub-00000000/00000000</string>
```

Ajoutez ConfigChanges et métadonnées à votre manifeste:

```
<activity
            android:name="com.google.android.gms.ads.AdActivity"

android:configChanges="keyboard|keyboardHidden|orientation|screenLayout|uiMode|screenSize|smallestScree

            android:theme="@android:style/Theme.Translucent" />
```

et

```
<meta-data
            android:name="com.google.android.gms.version"
            android:value="@integer/google_play_services_version" />
```

Activité:

```
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();
        }
    }

}
```

Cette AdActivity affichera maintenant une annonce en plein écran.

Lire Affichage des annonces Google en ligne:
https://riptutorial.com/fr/android/topic/5984/affichage-des-annonces-google

Chapitre 9: AIDL

Introduction

AIDL est un langage de définition d'interface Android.

Quelle? Pourquoi? Comment ?

Quelle? C'est un service limité. Ce service AIDL sera actif jusqu'à ce qu'au moins l'un des clients existe. Il fonctionne sur la base du concept de marshaling et unmarshaling.

Pourquoi? Les applications distantes peuvent accéder à votre service + Multi Threading (demande d'application distante).

Comment? Créez le fichier .aidl Implémentez l'interface Exposez l'interface aux clients

Examples

Service AIDL

ICalculator.aidl

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

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

AidlService.java

```
public class AidlService extends Service {

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

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

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

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

```
    }

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


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

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

}
```

Connexion de service

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

Lire AIDL en ligne: https://riptutorial.com/fr/android/topic/9504/aidl

Chapitre 10: Ajout d'un FuseView à un projet Android

Introduction

Exportez un Fuse.View depuis fusetools et utilisez-le dans un projet Android existant.

Notre objectif est d'exporter la totalité de l' application hikr et de l'utiliser dans une `Activity` .

Le travail final peut être trouvé @ lucamtudor / hikr-fuse-view

Examples

hikr app, juste un autre android.view.View

Conditions préalables

- vous devriez avoir un fusible installé (https://www.fusetools.com/downloads)
- vous devriez avoir fait le tutoriel d'introduction
- dans le terminal: `fuse install android`
- dans le terminal: `uno install Fuse.Views`

Étape 1

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

Étape 2 : Ajouter une référence de package à `Fuse.Views`

Recherchez le fichier `hikr.unoproj` dans le dossier racine du projet et ajoutez `"Fuse.Views"` au tableau `"Packages"` .

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

Étape 3 : Créer le composant `HikrApp` pour conserver l'application entière

3.1 Dans le dossier racine du projet, créez un nouveau fichier appelé `HikrApp.ux` et collez le contenu de `MainView.ux` .

HikrApp.ux

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

    <Router ux:Name="router" />

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

3.2 Dans `HikrApp.ux`

- remplacer les balises `<App>` par `<Page>`
- add `ux:Class="HikrApp"` à l'ouverture `<Page>`
- remove `<ClientPanel>` , nous n'avons plus à nous soucier de la barre d'état ou des boutons de navigation du bas

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 Utilisez le nouveau `HikrApp` composant à l' intérieur `MainView.ux`

Remplacez le contenu du fichier `MainView.ux` par:

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

Notre application est de retour à son comportement normal, mais nous l'avons maintenant extrait d'un composant distinct appelé `HikrApp`

Étape 4 Dans `MainView.ux` remplacez les balises `<App>` par `<ExportedViews>` et ajoutez `ux:Template="HikrAppView"` à `<HikrApp />`

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

Rappelez-vous le modèle `HikrAppView` , car nous en aurons besoin pour obtenir une référence à notre vue depuis Java.

Note À partir des documents de fusibles:

> `ExportedViews` se comportera comme une `App` lors de la `fuse preview` normale des `fuse preview` et lors de la `uno build`

Pas vrai. Vous obtiendrez cette erreur lors de la prévisualisation depuis Fuse Studio:

> Erreur: Impossible de trouver une balise App dans l'un des fichiers UX inclus. Avez-vous oublié d'inclure le fichier UX contenant la balise d'application?

Étape 5 `SplashPage.ux` le `<DockPanel>` dans un `<GraphicsView>`

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

    <JavaScript File="SplashPage.js" />

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

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

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

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

```
</Page>
```

Étape 6 Exportez le projet de fusible en tant que bibliothèque aar

- dans le terminal, dans le dossier du projet racine: `uno clean`
- dans le terminal, dans le dossier du projet racine: `uno build -t=android -DLIBRARY`

Étape 7 Préparez votre projet Android

- copier l'arar de `.../rootHikeProject/build/Android/Debug/app/build/outputs/aar/app-debug.aar` vers `.../androidRootProject/app/libs`
- Ajouter `flatDir { dirs 'libs' }` au fichier racine `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'
        }
    }
}

...
```

- ajouter `compile(name: 'app-debug', ext: 'aar')` aux dépendances dans `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'
}
```

- Ajoutez les propriétés suivantes à l'activité dans `AndroidManifest.xml`

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

Votre `AndroidManifest.xml` ressemblera à ceci:

```
<?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>
```

Étape 8 : Afficher le `Fuse.View HikrAppView` dans votre `Activity`

- Notez que votre `Activity` doit hériter de `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);
    }
}
```

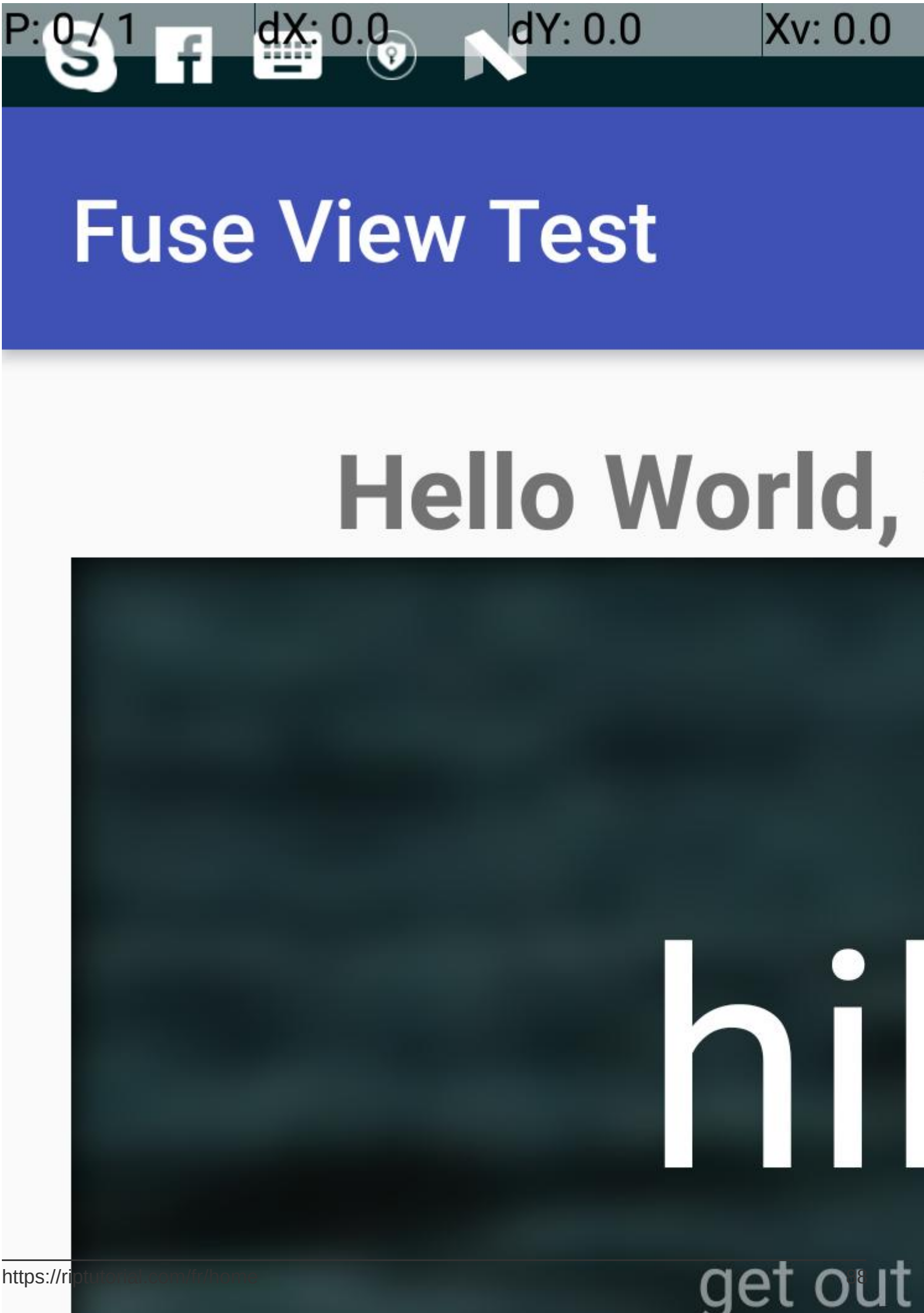
activity_main.xml

```
<?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>
```

Remarque
Lorsque vous appuyez sur le bouton de retour, sur Android, l'application se bloque. Vous pouvez suivre le problème sur le forum des fusibles .

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

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

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

Et le résultat final est quelque chose comme ça. Vous pouvez également trouver un court clip sur github .


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

https://riptutorial.com/fr/android/topic/10052/ajout-d-un-fuseview-a-un-projet-android

Chapitre 11: AlarmManager

Examples

Exécuter une intention ultérieurement

1. Créez un récepteur. Cette classe recevra l'intention et la traitera comme vous le souhaitez.

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

2. Donnez une intention à AlarmManager. Cet exemple déclenchera l'intention d'être envoyé à AlarmReceiver après 1 minute.

```
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 );
```

Comment annuler une alarme

Si vous souhaitez annuler une alarme et que vous n'avez pas de référence au PendingIntent d'origine utilisé pour définir l'alarme, vous devez recréer un PendingIntent exactement comme lors de sa création initiale.

Un Intent est considéré comme égal par le AlarmManager :

> si leur action, données, type, classe et catégories sont les mêmes. Cela ne compare aucune donnée supplémentaire incluse dans les intentions.

Le code de demande de chaque alarme est généralement défini comme une constante:

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

Donc, pour une simple alarme configurée comme ceci:

```
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);
```

Voici comment créer une nouvelle référence PendingIntent que vous pouvez utiliser pour annuler l'alarme avec une nouvelle référence AlarmManager:

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

Création d'alarmes exactes sur toutes les versions Android

Avec de plus en plus d'optimisations de la batterie dans le système Android au fil du temps, les méthodes du `AlarmManager` ont également considérablement changé (pour permettre une synchronisation plus souple). Cependant, pour certaines applications, il doit toujours être aussi précis que possible sur toutes les versions Android. L'assistant suivant utilise la méthode la plus précise disponible sur toutes les plates-formes pour planifier un `PendingIntent` :

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

Le mode API23 + Doze interfère avec AlarmManager

Android 6 (API23) a introduit le mode Doze qui interfère avec AlarmManager. Il utilise certaines fenêtres de maintenance pour gérer les alarmes. Même si vous avez utilisé `setExactAndAllowWhileIdle()` vous ne pouvez pas vous assurer que votre alarme se déclenche au moment voulu.

Vous pouvez désactiver ce comportement pour votre application en utilisant les paramètres de votre téléphone (`Settings/General/Battery & power saving/Battery usage/Ignore optimizations` ou similaire)

Dans votre application, vous pouvez vérifier ce paramètre ...

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

```
}
```

... et éventuellement afficher la boîte de dialogue des paramètres respectifs:

```
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);
```

Lire AlarmManager en ligne: https://riptutorial.com/fr/android/topic/1361/alarmmanager

Chapitre 12: Amélioration des dialogues d'alerte

Introduction

Cette rubrique concerne l'amélioration d'un `AlertDialog` avec des fonctionnalités supplémentaires.

Examples

Boîte de dialogue d'alerte contenant un lien cliquable

Pour afficher une boîte de dialogue contenant un lien pouvant être ouvert en cliquant dessus, vous pouvez utiliser le code suivant:

```
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
```

Lire Amélioration des dialogues d'alerte en ligne:
https://riptutorial.com/fr/android/topic/10163/amelioration-des-dialogues-d-alerte

Chapitre 13: Amélioration des performances Android à l'aide des polices Icon

Remarques

Les polices d'icône sont comme les types de police normaux qui ont des symboles au lieu de lettres. Il peut être utilisé dans votre application avec le plus de facilité.

Elles sont:

- Flexible
- Évolutive
- Vecteurs
- Processable Rapide
- Poids léger
- Accessible

Effet sur la taille

L'exportation d'une image de différentes tailles pour des appareils Android coûterait à votre application une taille d'actif supplémentaire d'environ 30 Ko par image. Tout en ajoutant un fichier de polices (.ttf) avec environ 36 icônes, cela ne coûterait que 9 Ko. Imaginez le cas si vous ajoutez 36 fichiers individuels de différentes configurations, ce serait environ 1000 Ko. C'est une quantité raisonnable d'espace que vous économiserez en utilisant des polices d'icônes.

Limites des polices Icon.

- Les polices d'icônes peuvent être utilisées dans le tiroir de navigation. Leur utilisation dans les vues de navigation en tant qu'icône des éléments de menu n'est pas possible car le fichier de menu ne peut pas être créé sans spécifier le titre. Il est donc conseillé d'utiliser les fichiers svg comme ressources pour ces icônes.
- Les polices d'icône ne peuvent pas être utilisées dans un bouton d'action flottant. comme ils n'ont pas d'attribut `setText()` .
- Les polices externes ne peuvent pas être appliquées à partir de XML. Ils doivent être spécifiés en utilisant le fichier java. Ou bien vous devez étendre la vue de base et créer une vue telle que spécifiée dans cet article.

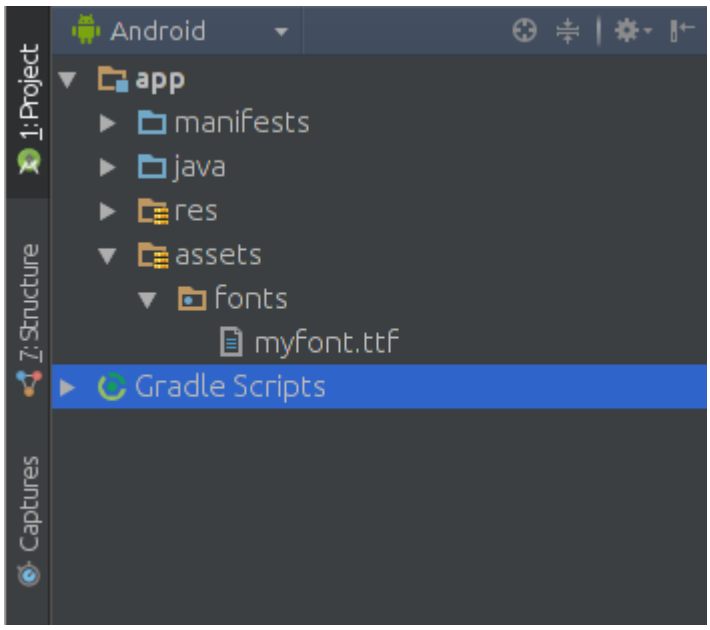
Examples

Comment intégrer les polices Icon

Pour utiliser les polices d'icônes, suivez simplement les étapes ci-dessous:

- **Ajouter le fichier de police à votre projet**

  Vous pouvez créer votre fichier d'icône de police à partir de sites Web en ligne tels que icomoon , où vous pouvez télécharger des fichiers SVG contenant les icônes requises, puis télécharger la police d'icône créée. Ensuite, placez le fichier de police *.ttf* dans un dossier nommé *polices* (nommez-le comme vous le souhaitez) dans le dossier assets:



- **Créer une classe d'assistance**

  Maintenant, créez la classe d'assistance suivante afin d'éviter de répéter le code d'initialisation de la police:

```
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);
  }
}
```

  Vous pouvez utiliser la classe `Typeface` pour sélectionner la police à partir des actifs. De cette façon, vous pouvez définir la police sur différentes vues, par exemple sur un bouton:

```
Button button=(Button) findViewById(R.id.button);
Typeface iconFont=FontManager.getTypeface(getApplicationContext());
button.setTypeface(iconFont);
```

  Maintenant, la police de caractères du bouton a été remplacée par la police d'icône nouvellement créée.

- **Ramassez les icônes que vous voulez**

  Ouvrez le fichier *styles.css* attaché à la police d'icône. Vous y trouverez les styles avec les caractères Unicode de vos icônes:

```
.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";
}
```

Ce fichier de ressources servira de dictionnaire, qui associe le caractère Unicode associé à une icône spécifique à un nom lisible par l'homme. Maintenant, créez les ressources de chaîne comme suit:

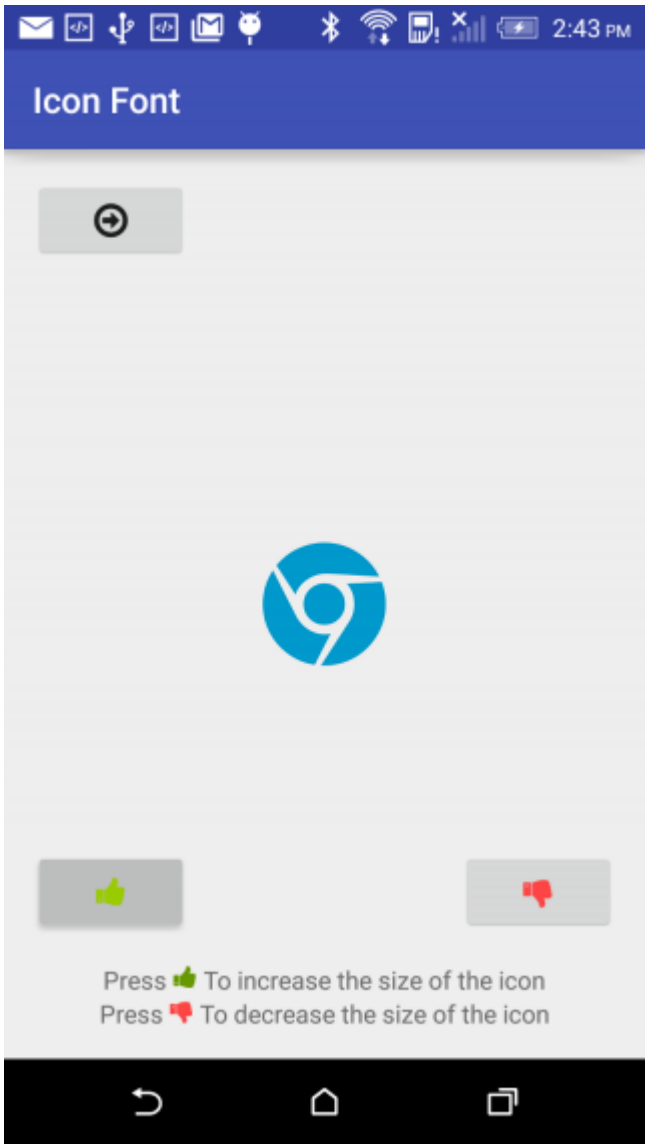
```
<resources>
  <! — Icon Fonts -->
  <string name="icon_arrow_circle_down">&#xe001; </string>
  <string name="icon_arrow_circle_left">&#xe002; </string>
  <string name="icon_arrow_circle-o_down">&#xe003; </string>
  <string name="icon_arrow_circle_o_left">&#xe004; </string>
</resources>
```

- **Utilisez les icônes dans votre code**

  Maintenant, vous pouvez utiliser votre police dans différentes vues, par exemple, comme suit:

  ```
  button.setText(getString(R.string.icon_arrow_circle_left))
  ```

  Vous pouvez également créer des vues de texte de bouton à l'aide des polices d'icônes:



TabLayout avec des polices d'icône

```
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;
}

}
```

- Dans cet exemple, myfont.ttf est dans le dossier Assets. Création du dossier Assets
- Dans votre classe d'activité

```
//..
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);
    //..
```

Lire Amélioration des performances Android à l'aide des polices Icon en ligne: https://riptutorial.com/fr/android/topic/3642/amelioration-des-performances-android-a-l-aide-des-polices-icon

Chapitre 14: Android Java Native Interface (JNI)

Introduction

JNI (Java Native Interface) est un outil puissant qui permet aux développeurs Android d'utiliser le NDK et d'utiliser du code natif C ++ dans leurs applications. Cette rubrique décrit l'utilisation de l'interface Java <-> C ++.

Examples

Comment appeler des fonctions dans une bibliothèque native via l'interface JNI

L' interface Java native (JNI) vous permet d'appeler des fonctions natives à partir du code Java et inversement. Cet exemple montre comment charger et appeler une fonction native via JNI, il n'entre pas dans l'accès aux méthodes et champs Java à partir du code natif à l'aide des fonctions JNI .

Supposons que vous ayez une bibliothèque native nommée `libjniexample.so` dans le dossier `project/libs/<architecture>` et que vous souhaitiez appeler une fonction de la classe Java `JNITest` dans le package `com.example.jniexample` .

Dans la classe JNITest, déclarez la fonction comme ceci:

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

Dans votre code natif, définissez la fonction comme ceci:

```
#include <jni.h>

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

L'argument `pEnv` est un pointeur sur l'environnement JNI que vous pouvez transmettre aux fonctions JNI pour accéder aux méthodes et aux champs d'objets et de classes Java. Le pointeur `thiz` est une référence de `jobject` à l'objet Java sur lequel la méthode native a été appelée (ou la classe s'il s'agit d'une méthode statique).

Dans votre code Java, dans `JNITest` , chargez la bibliothèque comme `JNITest` :

```
static{
```

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

Notez la `lib` au début, et le `.so` à la fin du nom de fichier est omis.

Appelez la fonction native à partir de Java comme ceci:

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

Comment appeler une méthode Java à partir du code natif

L'interface Java native (JNI) vous permet d'appeler des fonctions Java à partir de code natif. Voici un exemple simple de comment le faire:

Code Java:

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

Code natif:

```
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);
}
```

Signature de la méthode JNI vers le type Java:

Signature JNI	Type Java
Z	booléen
B	octet
C	carboniser
S	court

Signature JNI	Type Java
je	int
J	longue
F	flotte
ré	double
L classe entièrement qualifiée;	classe qualifiée
[type	type[]

Donc, pour notre exemple, nous avons utilisé (Z) I - ce qui signifie que la fonction obtient un booléen et renvoie un int.

Méthode utilitaire dans la couche JNI

Cette méthode aidera à obtenir la chaîne Java de la chaîne C ++.

```
jstring getJavaStringFromCPPString(JNIEnv *global_env, const char* cstring) {

        jstring nullString = global_env->NewStringUTF(NULL);

        if (!cstring) {
            return nullString;
        }

        jclass strClass = global_env->FindClass("java/lang/String");
        jmethodID ctorID = global_env->GetMethodID(strClass, "<init>",
                "([BLjava/lang/String;)V");
        jstring encoding = global_env->NewStringUTF("UTF-8");

        jbyteArray bytes = global_env->NewByteArray(strlen(cstring));
        global_env->SetByteArrayRegion(bytes, 0, strlen(cstring), (jbyte*) cstring);
        jstring str = (jstring) global_env->NewObject(strClass, ctorID, bytes,
                encoding);

        global_env->DeleteLocalRef(strClass);
        global_env->DeleteLocalRef(encoding);
        global_env->DeleteLocalRef(bytes);

        return str;
    }
```

Cette méthode vous aidera à convertir jbyteArray en char

```
char* as_unsigned_char_array(JNIEnv *env, jbyteArray array) {
    jsize length = env->GetArrayLength(array);
    jbyte* buffer = new jbyte[length + 1];

    env->GetByteArrayRegion(array, 0, length, buffer);
    buffer[length] = '\0';
```

```
    return (char*) buffer;
}
```

Lire Android Java Native Interface (JNI) en ligne:
https://riptutorial.com/fr/android/topic/8674/android-java-native-interface--jni-

Chapitre 15: Android Vk Sdk

Examples

Initialisation et connexion

1. Créer une nouvelle application ici: créer une application
2. Choisissez une application autonome et confirmez la création de l'application via SMS.
3. Remplissez le nom du **package pour Android en** tant que nom de votre package actuel. *Vous pouvez obtenir le nom de votre paquet dans le fichier manifeste Android, au tout début.*
4. Obtenez votre **empreinte de certificat** en exécutant cette commande dans votre shell / cmd:

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

Vous pouvez également obtenir cette empreinte par SDK lui-même:

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

5. Ajouter une empreinte digitale reçue dans votre **empreinte de certificat de signature pour Android:** dans les paramètres de l'application Vk (où vous avez entré le nom de votre package)

6. Ensuite, ajoutez ceci à votre fichier de gradle:

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

8. Initialisez le SDK au démarrage en utilisant la méthode suivante. Le meilleur moyen est de l'appeler dans la méthode Applications onCreate.

```
private static final int VK_ID = your_vk_id;
public static final String VK_API_VERSION = "5.52"; //current version
@Override
    public void onCreate() {
        super.onCreate();
        VKSdk.customInitialize(this, VK_ID, VK_API_VERSION);
}
```

C'est le meilleur moyen d'initialiser VKSdk. N'utilisez pas le methid où VK_ID doit être placé dans strings.xml car api ne fonctionnera pas correctement après.

9. La dernière étape consiste à vous connecter en utilisant vksdk.

```
    public static final String[] VK_SCOPES = new String[]{
            VKScope.FRIENDS,
```

```
            VKScope.MESSAGES,
            VKScope.NOTIFICATIONS,
            VKScope.OFFLINE,
            VKScope.STATUS,
            VKScope.STATS,
            VKScope.PHOTOS
    };

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

        someButtonForLogin.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
               VKSdk.login(this, VK_SCOPES);
            }
        });

    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        VKSdk.onActivityResult(requestCode, resultCode, data, new VKCallback<VKAccessToken>()
{
            @Override
            public void onResult(VKAccessToken res) {
                res.accessToken; //getting our token here.
            }

            @Override
            public void onError(VKError error) {
                Toast.makeText(SocialNetworkChooseActivity.this,
                        "User didn't pass Authorization", Toast.LENGTH_SHORT).show();
            }
        });
    }
```

Lire Android Vk Sdk en ligne: https://riptutorial.com/fr/android/topic/6046/android-vk-sdk

Chapitre 16: Android-x86 dans VirtualBox

Introduction

L'idée de cette section est de savoir comment installer et utiliser VirtualBox avec Android-x86 à des fins de débogage. C'est une tâche difficile car il existe des différences entre les versions. Pour le moment, je vais couvrir 6.0, celui avec lequel je devais travailler et ensuite nous devrons trouver des similitudes.

Il ne couvre pas VirtualBox ou un Linux en détail mais il montre les commandes que j'ai utilisées pour le faire fonctionner.

Examples

Configuration de la machine virtuelle

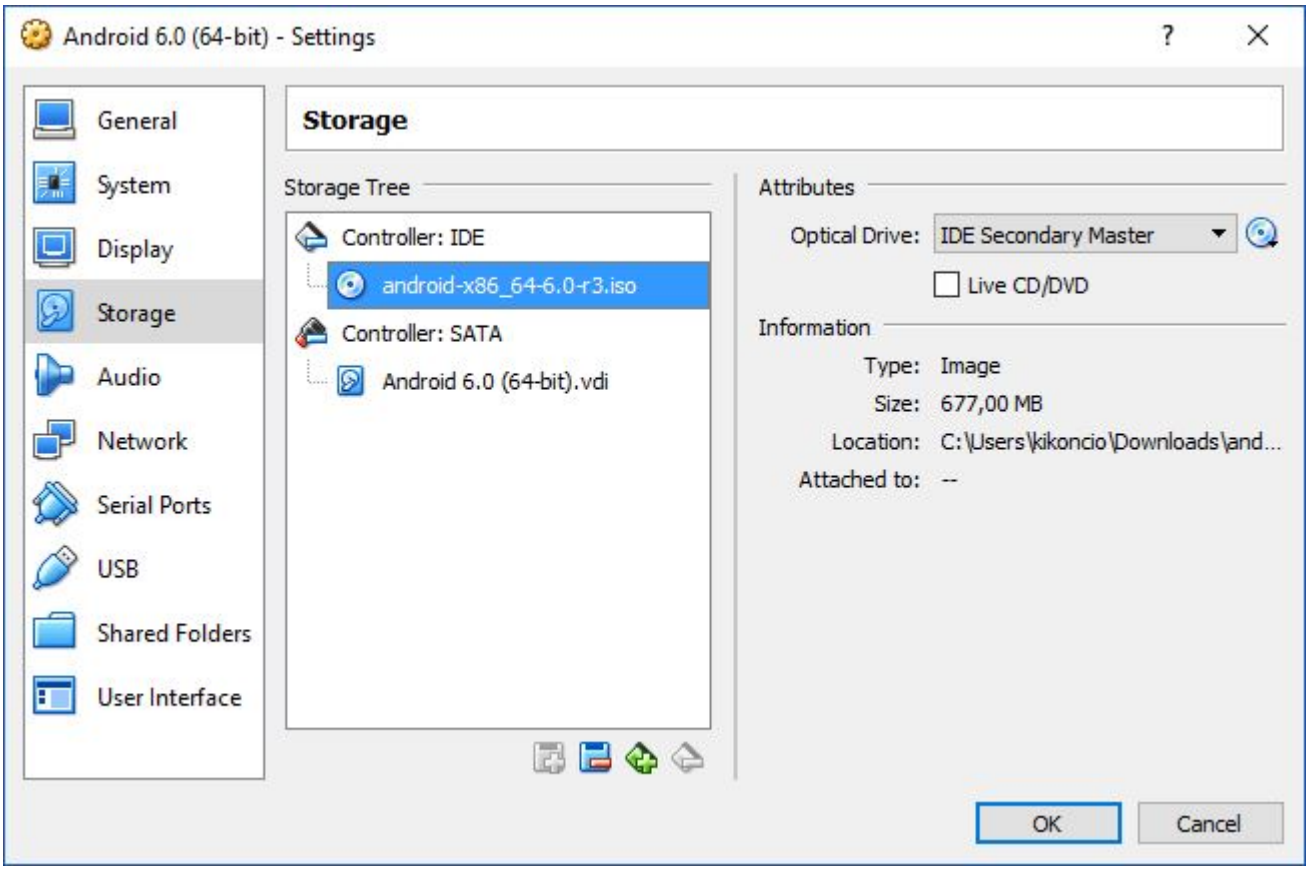
Ce sont mes paramètres VirtualBox:

- Type d'OS: Linux 2.6 (j'ai un utilisateur 64 bits car mon ordinateur peut le supporter)
- Taille du disque dur virtuel: 4 Go
- Mémoire Ram: 2048
- Mémoire vidéo: 8M
- Appareil audio: Sound Blaster 16.
- Périphérique réseau: PCnet-Fast III, connecté à NAT. Vous pouvez également utiliser un adaptateur ponté, mais vous avez besoin d'un serveur DHCP dans votre environnement.

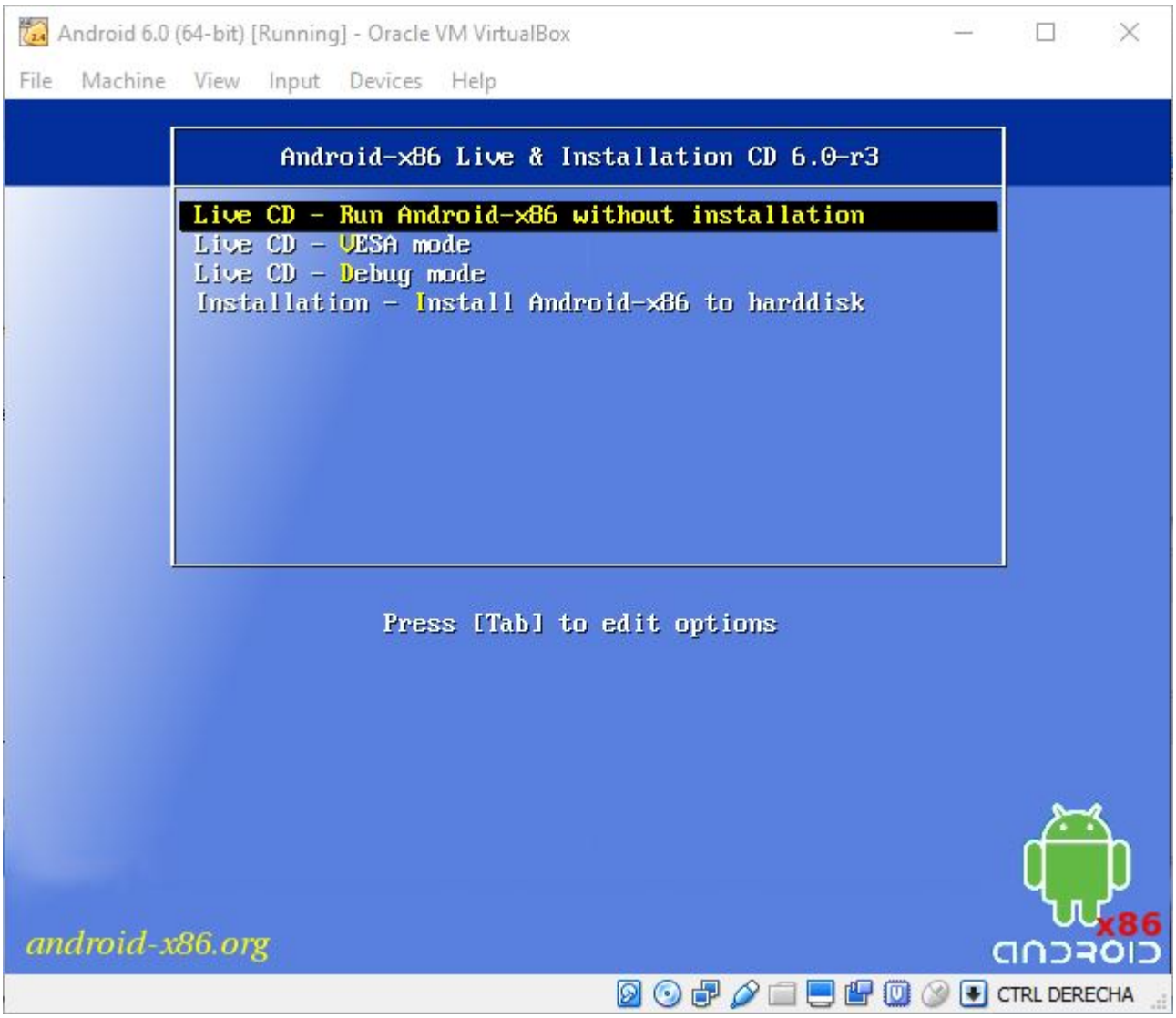
L'image utilisée avec cette configuration est android-x86_64-6.0-r3.iso (64 bits) téléchargée depuis http://www.android-x86.org/download . Je suppose que cela fonctionne aussi avec la version 32 bits.

Configuration du disque dur virtuel pour le support SDCARD

Avec le disque dur virtuel créé, démarrez la machine virtuelle avec l'image android-x86 dans le lecteur optique.



Une fois que vous démarrez, vous pouvez voir le menu grub du Live CD



Choisissez l'option de mode de débogage, alors vous devriez voir l'invite de shell. Ceci est un shell busybox. Vous pouvez obtenir plus de shell en basculant entre la console virtuelle Alt-F1 / F2 / F3.

Créez deux partitions par fdisk (d'autres versions utiliseraient cfdisk). Formatez-les en ext3. Puis redémarrez:

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

Puis tapez:

"n" (nouvelle partition)

"p" (partition primaire)

"1" (1ère partition)

"1" (premier cylindre)

"261" (choisissez un cylindre, nous laisserons 50% du disque pour une 2ème partition)

"2" (2ème partition)

"262" (262ème cylindre)

"522" (choisissez le dernier cylindre)

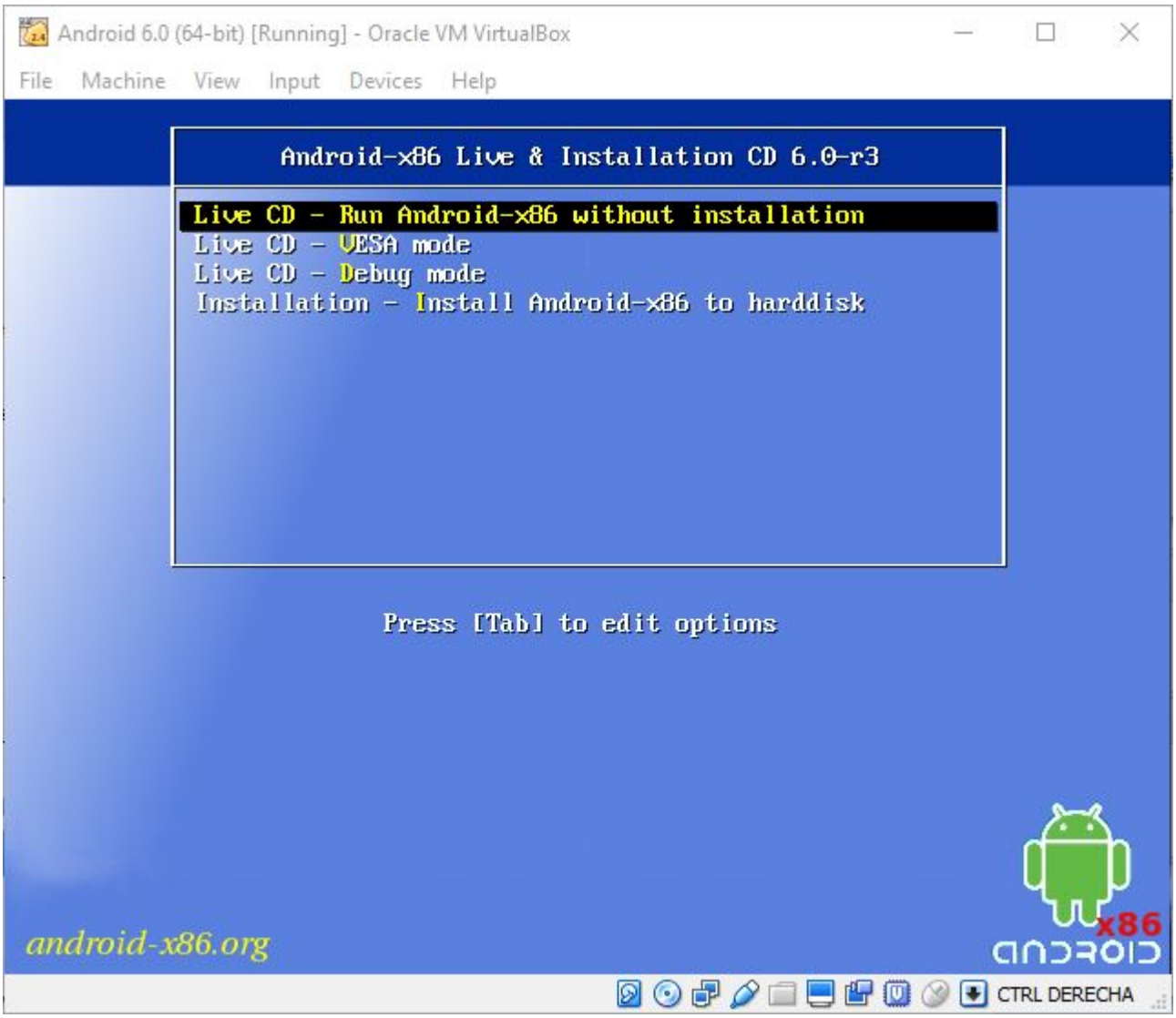
"w" (écrire la partition)

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

Lorsque vous redémarrez la machine virtuelle et que le menu grub apparaît et que vous pouvez modifier la ligne de démarrage du noyau, vous pouvez ajouter les options `DATA=sda1 SDCARD=sda2` pour indiquer la carte SD ou la partition de données.

Installation en partition

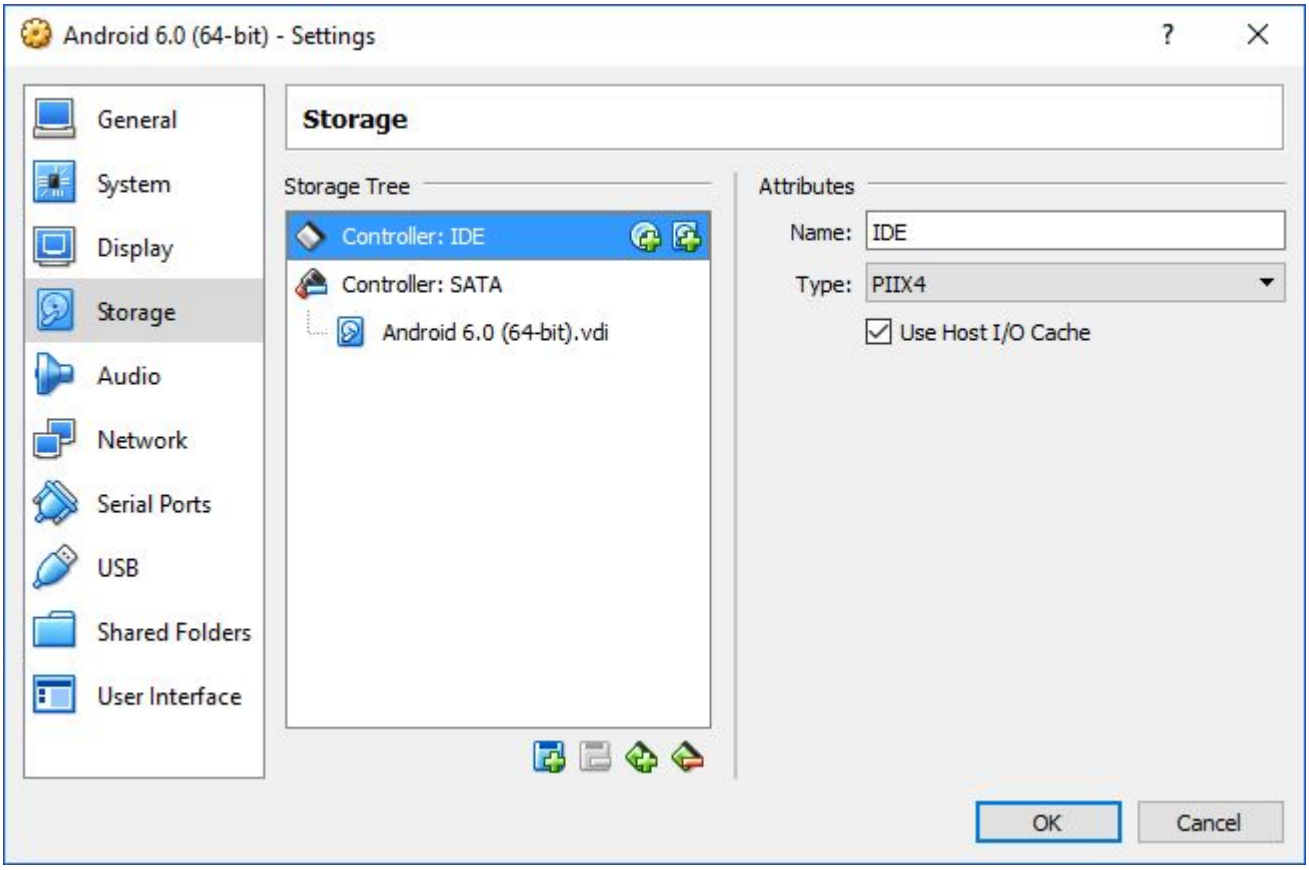
Avec le disque dur virtuel que vous venez de créer, démarrez la machine virtuelle avec l'image android-x86 en tant que lecteur optique.



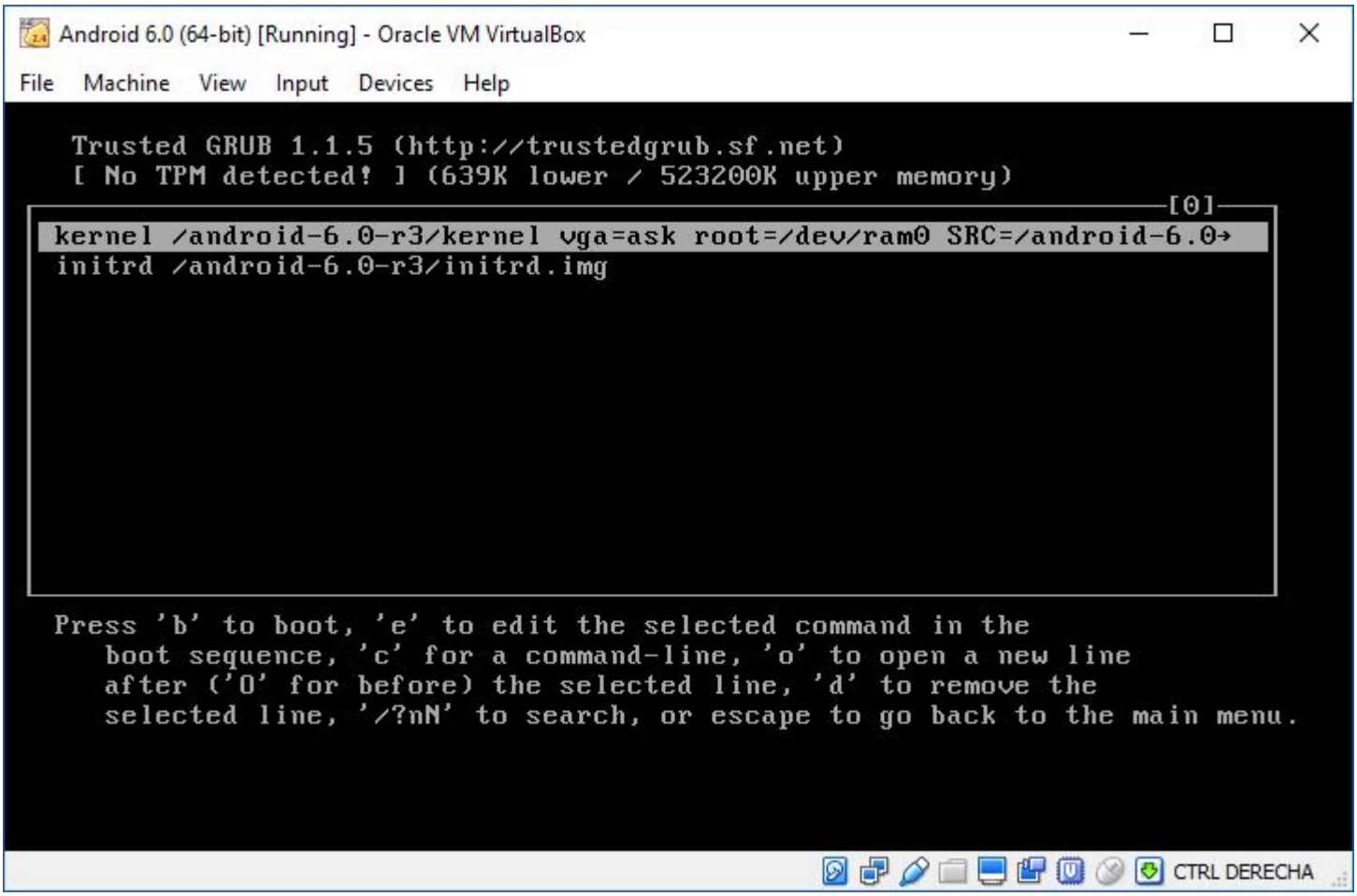
Dans les options de démarrage du Live CD, choisissez "Installation - Installer Android sur le disque dur"

Choisissez la partition sda1 et installez Android et nous installerons grub.

Redémarrez la machine virtuelle mais assurez-vous que l'image ne se trouve pas dans le lecteur optique pour pouvoir redémarrer à partir du disque dur virtuel.



Dans le menu grub, nous devons éditer le noyau comme dans l'option "Android-x86 6.0-r3", donc appuyez sur e.

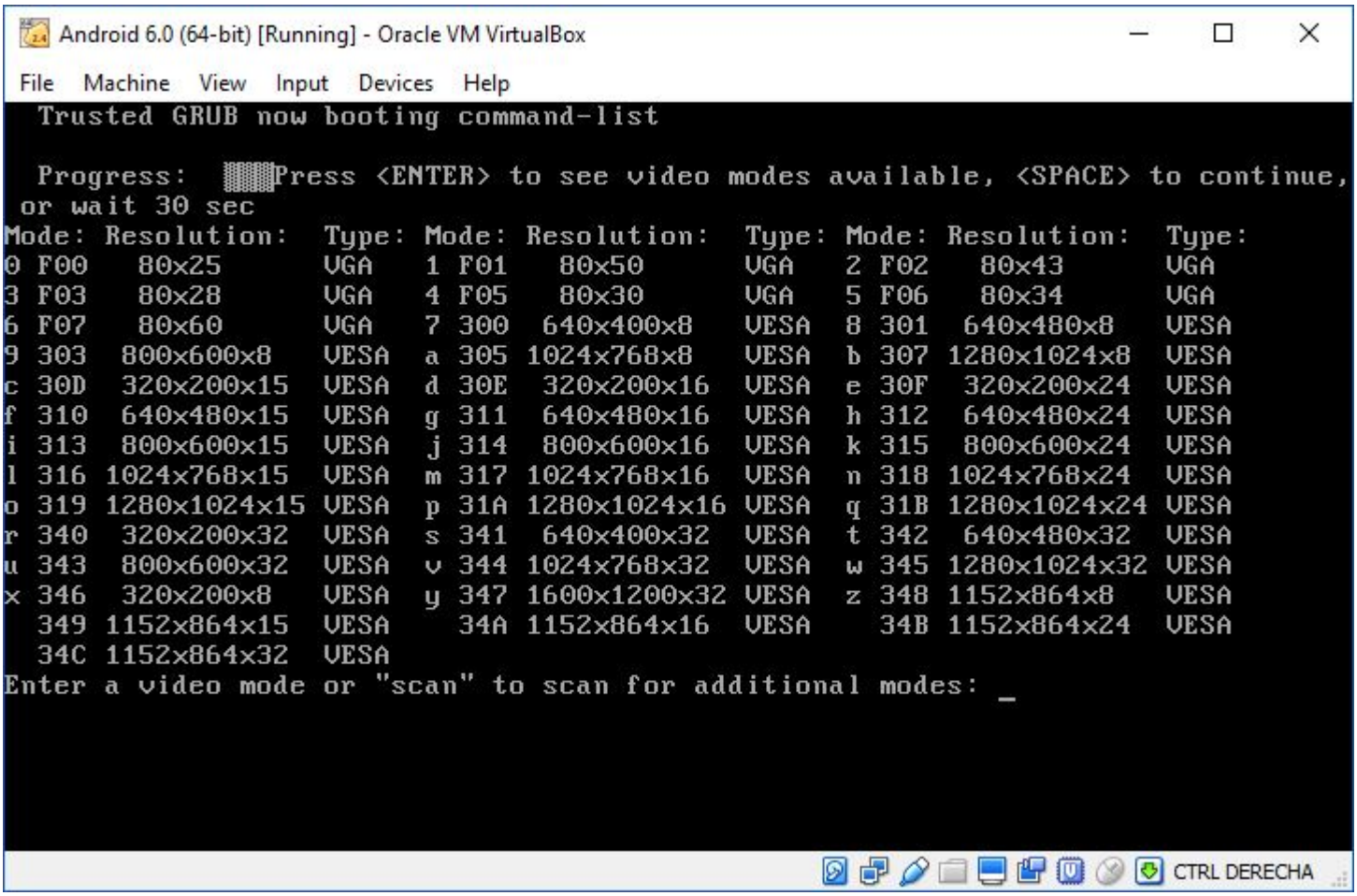


Ensuite, nous substituons "quiet" par "vga = ask" et ajoutons l'option "SDCARD = sda2"

Dans mon cas, la ligne du noyau ressemble à ceci après modification:

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

Appuyez sur b pour démarrer, vous pourrez alors choisir la taille de l'écran en appuyant sur ENTER (l'option `vga=ask`)



Une fois que l'assistant d'installation a démarré, choisissez la langue. Je pouvais choisir l'anglais (États-Unis) et l'espagnol (États-Unis) et j'ai eu du mal à en choisir un autre.

Lire Android-x86 dans VirtualBox en ligne: https://riptutorial.com/fr/android/topic/9903/android-x86-dans-virtualbox

Chapitre 17: Animateurs

Examples

Secouer l'animation d'une ImageView

Sous le dossier res, créez un nouveau dossier appelé "anim" pour stocker vos ressources d'animation et mettez-le dans ce dossier.

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" />
```

Créer une activité vide appelée 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>
```

Et la méthode pour animer l'imageview sur 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 de fondu entrant / sortant

Pour afficher ou `ObjectAnimator` lentement, utilisez un `ObjectAnimator` . Comme indiqué dans le code ci-dessous, définissez une durée à l'aide de `.setDuration(millis)` où le paramètre `millis` correspond à la durée (en millisecondes) de l'animation. Dans le code ci-dessous, les vues s'afficheront ou disparaîtront plus de 500 millisecondes ou 1/2 seconde. Pour démarrer le `ObjectAnimator` animation d », appelez `.start()` . Une fois l'animation terminée, on `onAnimationEnd(Animator animation)` . Voici un bon endroit pour définir la visibilité de votre vue sur `View.GONE` ou `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();
}
```

Animation TransitionDrawable

Cet exemple affiche une transaction pour une vue d'image avec seulement deux images (peut

utiliser plus d'images l'une après l'autre pour les positions du premier et du deuxième calque après chaque transaction en boucle).

- Ajouter un tableau d'images à `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` introduit un moyen simple d'animer une valeur (d'un type particulier, par exemple `int` , `float` , etc.).

La manière habituelle de l'utiliser est:

1. Créez un `ValueAnimator` qui animera une valeur de `min` à `max`
2. Ajoutez un `UpdateListener` dans lequel vous utiliserez la valeur animée calculée (que vous pouvez obtenir avec `getAnimatedValue()`)

Vous pouvez créer le `ValueAnimator` deux manières:

(l'exemple de code anime un `float` de `20f` à `40f` en `250ms`)

1. De `xml` (mettez le dans `/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. Du 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` est une sous-classe de `ValueAnimator` avec la possibilité d'ajouter la valeur calculée à la propriété d'une `View` `target` .

Tout comme dans `ValueAnimator` , il existe deux manières de créer `ObjectAnimator` :

(l'exemple de code anime un `alpha` d'un `View` de `0.4f` à `0.2f` dans `250ms`)

1. De `xml` (mettez-le dans le `/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. À partir du 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` est un moyen simplifié et optimisé d'animer les propriétés d'une `View` .

Chaque `View` a un objet `ViewPropertyAnimator` disponible via la méthode `animate()` . Vous pouvez l'utiliser pour animer plusieurs propriétés à la fois avec un simple appel. Chaque méthode d'un `ViewPropertyAnimator` spécifie la valeur **cible** d'un paramètre spécifique que `ViewPropertyAnimator` doit animer.

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

Remarque: L' appel de `start()` sur un objet `ViewPropertyAnimator` n'est **PAS** obligatoire. Si vous ne le faites pas, vous laissez simplement la plate-forme gérer le démarrage de l'animation au moment opportun (la prochaine manipulation de l'animation). Si vous le faites réellement (appelez `start()`), assurez-vous que l'animation est lancée immédiatement.

Développer et réduire l'animation de la vue

```
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);
        }
    }
```

Lire Animateurs en ligne: https://riptutorial.com/fr/android/topic/1829/animateurs

Chapitre 18: Annotations Typedef: @IntDef, @StringDef

Remarques

Le package d'annotations comprend un certain nombre d'annotations de métadonnées utiles avec lesquelles vous pouvez décorer votre propre code pour vous aider à détecter les bogues.

Ajoutez simplement la dépendance dans le fichier `build.gradle` .

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

Examples

IntDef Annotations

Cette annotation garantit que seules les constantes entières valides attendues sont utilisées. L'exemple suivant illustre les étapes pour créer une annotation:

```
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;
    }
```

```
}
```

Ils permettent également l'achèvement du code pour offrir automatiquement les constantes autorisées.
Lorsque vous générez ce code, un avertissement est généré si le paramètre type ne fait pas référence à l'une des constantes définies.

Combinaison de constantes avec des drapeaux

En utilisant l' `IntDef#flag()` défini sur `true` , plusieurs constantes peuvent être combinées.

En utilisant le même exemple dans cette rubrique:

```
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)

    .....

}
```

Les utilisateurs peuvent combiner les constantes autorisées avec un indicateur (tel que `|` , `&` , `^`).

Lire Annotations Typedef: @IntDef, @StringDef en ligne:
https://riptutorial.com/fr/android/topic/4505/annotations-typedef---intdef---stringdef

Chapitre 19: API Android Places

Examples

Exemple d'utilisation du sélecteur d'emplacement

Place Picker est un widget d'interface utilisateur très simple fourni par l'API Places. Il fournit une carte intégrée, l'emplacement actuel, les lieux à proximité, les capacités de recherche et la saisie semi-automatique.

Ceci est un exemple d'utilisation du widget de l'interface utilisateur de Picker Place.

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

```
}
```

Obtention des emplacements actuels à l'aide de l'API Places

Vous pouvez obtenir l'emplacement actuel et les lieux locaux de l'utilisateur à l'aide de l' API Google Adresses .

En premier lieu, vous devez appeler la méthode `PlaceDetectionApi.getCurrentPlace()` pour récupérer des entreprises locales ou d'autres lieux. Cette méthode retourne un objet `PlaceLikelihoodBuffer` qui contient une liste d'objets `PlaceLikelihood` . Ensuite, vous pouvez obtenir un objet `Place` en appelant la méthode `PlaceLikelihood.getPlace()` .

Important: Vous devez demander et obtenir l'autorisation `ACCESS_FINE_LOCATION` pour permettre à votre application d'accéder à des informations de localisation précises.

```
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());
}
```

Intégration de la saisie semi-automatique

La fonctionnalité de saisie semi-automatique de l'API Google Adresses pour Android fournit des prévisions de lieu à l'utilisateur. Alors que l'utilisateur tape dans la zone de recherche, la saisie semi-automatique indique les lieux en fonction des requêtes de l'utilisateur.

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"/>
```

Ajout de plus d'une activité complète de google auto.

```
 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.
           }
        }
    }
```

Définition des filtres de type de lieu pour PlaceAutocomplete

Dans certains scénarios, nous pouvons souhaiter limiter les résultats affichés par **PlaceAutocomplete** à un pays spécifique ou afficher uniquement les régions. Cela peut être réalisé en définissant un **filtre** de **saisie semi-automatique** sur l'intention. Par exemple, si je veux rechercher uniquement les lieux de type REGION et n'appartenant qu'à l'Inde, je ferais ce qui suit:

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>
```

Le **PlaceAutocomplete** se lancera automatiquement et vous pourrez alors sélectionner un endroit parmi les résultats qui seront uniquement du type **REGION** et appartiendront uniquement au pays spécifié. L'intention peut également être lancée d'un simple clic.

Lire API Android Places en ligne: https://riptutorial.com/fr/android/topic/4111/api-android-places

Chapitre 20: API Bluetooth et Bluetooth LE

Remarques

Bluetooth Classic est disponible à partir d'Android 2.0 (API niveau 5) et supérieur. Bluetooth LE est disponible à partir d'Android 4.3 (API niveau 18) et supérieur.

Examples

Autorisations

Ajoutez cette autorisation au fichier manifeste pour utiliser les fonctionnalités Bluetooth dans votre application:

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

Si vous devez lancer la découverte de périphérique ou manipuler les paramètres Bluetooth, vous devez également ajouter cette autorisation:

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

Cibler les niveaux 23 et supérieurs de l'API Android nécessitera un accès à l'emplacement:

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

* Voir également la rubrique Autorisations pour plus de détails sur la manière d'utiliser les autorisations de manière appropriée.

Vérifiez si Bluetooth est activé

```
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
        }
    }
}
```

Rendre l'appareil détectable

```
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
        }
    }
}
```

Trouver des périphériques Bluetooth à proximité

Déclarez d'abord un `BluetoothAdapter` .

```
BluetoothAdapter mBluetoothAdapter;
```

Maintenant, créez un `BroadcastReceiver` pour `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());
    }
  }
};
```

Enregistrer le `BroadcastReceiver`

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

Puis commencez à découvrir les périphériques Bluetooth à proximité en appelant `startDiscovery`

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

N'oubliez pas de désenregistrer le `BroadcastReceiver` dans `onDestroy`

```
unregisterReceiver(mReceiver);
```

Se connecter à un périphérique Bluetooth

Après avoir obtenu BluetoothDevice, vous pouvez communiquer avec lui. Ce type de communication est effectué en utilisant des flux d'entrée / sortie de socket:

Voici les étapes de base pour l'établissement de la communication Bluetooth:

1) Initialiser socket:

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

2) Connectez-vous à la prise:

```
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) Obtention des flux d'entrée \ sortie socket

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

Flux d'entrée - Utilisé comme canal de données entrant (recevoir des données du périphérique connecté)

Flux de sortie - Utilisé comme canal de données sortant (envoi de données au périphérique connecté)

Après avoir terminé la 3ème étape, nous pouvons recevoir et envoyer des données entre les deux appareils en utilisant des flux précédemment initialisés:

1) Réception de données (lecture depuis le flux d'entrée du socket)

```
byte[] buffer = new byte[1024];  // buffer (our data)
int bytesCount; // amount of read bytes

while (true) {
    try {
        //reading data from input stream
        bytesCount = _inStream.read(buffer);
        if(buffer != null && bytesCount > 0)
        {
            //Parse received bytes
        }
    } catch (IOException e) {
        //Error
    }
}
```

2) Envoi de données (écriture dans le flux de sortie)

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

- Bien entendu, les fonctionnalités de connexion, de lecture et d'écriture doivent être effectuées dans un thread dédié.
- Les sockets et les objets Stream doivent être

Trouver des appareils Bluetooth Low Energy à proximité

L'API BluetoothLE a été introduite dans l'API 18. Cependant, le mode d'analyse des périphériques a changé dans l'API 21. La recherche des périphériques doit commencer par la définition de l' UUID du service à analyser (des UUID 16 bits ou propriétaires) . Cet exemple illustre comment créer une méthode de recherche indépendante des API pour les périphériques BLE:

1. Créer un modèle d'appareil Bluetooth:

```
public class BTDevice {
    String address;
    String name;

    public String getAddress() {
        return address;
    }

    public void setAddress(String address) {
        this.address = address;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}
```

2. Définir l'interface de numérisation Bluetooth:

```
public interface ScanningAdapter {

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

3. Créer une classe de fabrique d'analyse:

```
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. Créer une implémentation d'usine pour chaque 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. Obtenez la liste des périphériques trouvés en appelant:

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

wait few seconds...

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

Lire API Bluetooth et Bluetooth LE en ligne: https://riptutorial.com/fr/android/topic/2462/api-bluetooth-et-bluetooth-le

Chapitre 21: API d'empreintes digitales dans Android

Remarques

voir également

Exemple de projet Github

Développeur Android blogspot

Site développeur Android

Examples

Ajout du scanner d'empreintes digitales dans une application Android

Android prend en charge l'API d'empreinte digitale d'Android 6.0 (Marshmallow) SDK 23

> Pour utiliser cette fonctionnalité dans votre application, ajoutez d'abord l'autorisation USE_FINGERPRINT dans votre manifeste.

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

Voici la procédure à suivre

> Vous devez d'abord créer une clé symétrique dans le magasin de clés Android à l'aide de KeyGenerator, qui ne peut être utilisée qu'après que l'utilisateur s'est authentifié avec l'empreinte digitale et passé un KeyGenParameterSpec.

```
KeyPairGenerator.getInstance(KeyProperties.KEY_ALGORITHM_EC, "AndroidKeyStore");
keyPairGenerator.initialize(
        new KeyGenParameterSpec.Builder(KEY_NAME,
                KeyProperties.PURPOSE_SIGN)
                .setDigests(KeyProperties.DIGEST_SHA256)
                .setAlgorithmParameterSpec(new ECGenParameterSpec("secp256r1"))
                .setUserAuthenticationRequired(true)
                .build());
keyPairGenerator.generateKeyPair();
```

> En définissant KeyGenParameterSpec.Builder.setUserAuthenticationRequired sur true, vous pouvez autoriser l'utilisation de la clé uniquement une fois que l'utilisateur l'a authentifiée, y compris lorsqu'elle est authentifiée avec l'empreinte digitale de l'utilisateur.

```
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);
```

> Commencez ensuite à écouter une empreinte digitale sur le capteur d'empreinte en appelant FingerprintManager.authenticate avec un chiffrement initialisé avec la clé symétrique créée. Vous pouvez également utiliser le mot de passe vérifié côté serveur en tant qu'authentificateur.

Créez et initialisez le `FingerprintManger` partir de `fingerprintManger.class`

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

Pour authentifier, utilisez `FingerprintManger` api et créez une sous-classe en utilisant

`FingerprintManager.AuthenticationCallback` et remplacer les méthodes

```
onAuthenticationError
onAuthenticationHelp
onAuthenticationSucceeded
onAuthenticationFailed
```

Commencer

Pour commencer à écouter la méthode d'authentification de l'appel d'événement fingerPrint avec crypto

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

Annuler

arrêter d'écouter l'appel du scanner

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

> Une fois que l'empreinte (ou le mot de passe) est vérifiée, le rappel FingerprintManager.AuthenticationCallback # onAuthenticationSucceeded () est appelé.

```
@Override

public void onAuthenticationSucceeded(AuthenticationResult result) {

            }
```

Comment utiliser Android Fingerprint API pour enregistrer les mots de passe des utilisateurs

Cet exemple de classe d'assistance interagit avec le gestionnaire d'empreintes digitales et effectue le chiffrement et le déchiffrement du mot de passe. Veuillez noter que la méthode utilisée pour le cryptage dans cet exemple est AES. Ce n'est pas la seule façon de chiffrer et d' autres exemples existent. Dans cet exemple, les données sont chiffrées et déchiffrées de la manière suivante:

Cryptage:

1. L'utilisateur donne à l'aide le mot de passe non crypté souhaité.
2. L'utilisateur doit fournir une empreinte digitale.
3. Une fois authentifié, l'assistant obtient une clé du `KeyStore` et chiffre le mot de passe à l'aide d'un `Cipher` .
4. Le mot de passe et le sel IV (IV est recréé pour chaque cryptage et n'est pas réutilisé) sont enregistrés dans les préférences partagées pour être utilisés ultérieurement dans le processus de décryptage.

Décryptage:

1. L'utilisateur demande à décrypter le mot de passe.
2. L'utilisateur doit fournir une empreinte digitale.
3. L'assistant construit un `Cipher` à l'aide de l'IV et, une fois l'utilisateur authentifié, le KeyStore obtient une clé du `KeyStore` et déchiffre le mot de passe.

```
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());
            }
```

```
        }
    }
}
```

Cette activité ci-dessous est un exemple très simple de la manière d'obtenir un mot de passe enregistré par l'utilisateur et d'interagir avec l'aide.

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

Lire API d'empreintes digitales dans Android en ligne:
https://riptutorial.com/fr/android/topic/7523/api-d-empreintes-digitales-dans-android

Chapitre 22: API de sensibilisation Google

Remarques

N'oubliez pas que l' API Snapshot est utilisée pour demander l'état actuel tandis que l' API Fence vérifie en permanence l'état spécifié et envoie des rappels lorsqu'une application n'est pas en cours d'exécution.

Dans l'ensemble, il existe quelques étapes de base pour utiliser l'API Snapshot ou l'API de clôture:

- Obtenir une clé API à partir de la console de développement Google
- Ajoutez les autorisations nécessaires et la clé API au manifeste:

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

- Initalise le `GoogleApiClient` quelque part, de préférence dans la méthode onCreate () de votre activité.

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

- Appelez l'API de votre choix
- Résultat de l'analyse

Un moyen simple de vérifier les autorisations requises est une méthode comme celle-ci:

```
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

Obtenir l'activité utilisateur actuelle à l'aide de l'API Snapshot

Pour les demandes ponctuelles non constantes relatives à l'activité physique d'un utilisateur, utilisez l'API Snapshot:

```
// Remember to initialize your client as described in the Remarks section
Awareness.SnapshotApi.getDetectedActivity(client)
    .setResultCallback(new ResultCallback<DetectedActivityResult>() {
        @Override
        public void onResult(@NonNull DetectedActivityResult detectedActivityResult) {
            if (!detectedActivityResult.getStatus().isSuccess()) {
                Log.e(getClass().getSimpleName(), "Could not get the current activity.");
                return;
            }
            ActivityRecognitionResult result = detectedActivityResult
                .getActivityRecognitionResult();
            DetectedActivity probableActivity = result.getMostProbableActivity();
            Log.i(getClass().getSimpleName(), "Activity received : " +
                probableActivity.toString());
        }
    });
```

Obtenir l'état du casque avec l'API Snapshot

```
// Remember to initialize your client as described in the Remarks section
Awareness.SnapshotApi.getHeadphoneState(client)
    .setResultCallback(new ResultCallback<HeadphoneStateResult>() {
        @Override
        public void onResult(@NonNull HeadphoneStateResult headphoneStateResult) {
            Log.i(TAG, "Headphone state connection state: " +
                headphoneStateResult.getHeadphoneState()
                .getState() == HeadphoneState.PLUGGED_IN));
        }
    });
```

Obtenir l'emplacement actuel à l'aide de Snapshot API

```
// Remember to intialize your client as described in the Remarks section
Awareness.SnapshotApi.getLocation(client)
    .setResultCallback(new ResultCallback<LocationResult>() {
        @Override
        public void onResult(@NonNull LocationResult locationResult) {
            Location location = locationResult.getLocation();
            Log.i(getClass().getSimpleName(), "Coordinates: "location.getLatitude() + "," +
                location.getLongitude() + ", radius : " + location.getAccuracy());
        }
    });
```

Obtenir des lieux à proximité en utilisant l'API Snapshot

```
// Remember to initialize your client as described in the Remarks section
Awareness.SnapshotApi.getPlaces(client)
    .setResultCallback(new ResultCallback<PlacesResult>() {
        @Override
```

```
        public void onResult(@NonNull PlacesResult placesResult) {
            List<PlaceLikelihood> likelihoodList = placesResult.getPlaceLikelihoods();
            if (likelihoodList == null || likelihoodList.isEmpty()) {
                Log.e(getClass().getSimpleName(), "No likely places");
            }
        }
    });
```

En ce qui concerne l'obtention des données à ces endroits, voici quelques options:

```
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()));
```

Obtenir la météo actuelle en utilisant l'API Snapshot

```
// Remember to initialize your client as described in the Remarks section
Awareness.SnapshotApi.getWeather(client)
    .setResultCallback(new ResultCallback<WeatherResult>() {
        @Override
        public void onResult(@NonNull WeatherResult weatherResult) {
            Weather weather = weatherResult.getWeather();
            if (weather == null) {
                Log.e(getClass().getSimpleName(), "No weather received");
            } else {
                Log.i(getClass().getSimpleName(), "Temperature is " +
                        weather.getTemperature(Weather.CELSIUS) + ", feels like " +
                        weather.getFeelsLikeTemperature(Weather.CELSIUS) +
                        ", humidity is " + weather.getHumidity());
            }
        }
    });
```

Obtenir des modifications dans l'activité de l'utilisateur avec l'API de clôture

Si vous souhaitez détecter quand votre utilisateur démarre ou termine une activité telle que la marche, l'exécution ou toute autre activité de la classe `DetectedActivityFence` , vous pouvez créer une clôture pour l'activité que vous souhaitez détecter et recevoir une notification lorsque votre utilisateur démarre / termine cette activité. En utilisant un `BroadcastReceiver` , vous obtiendrez une `Intent` avec des données contenant l'activité:

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

Maintenant, vous pouvez recevoir l'intention avec un `BroadcastReceiver` pour obtenir des rappels lorsque l'utilisateur modifie l'activité:

```
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;
        }
    }
}
```

Obtenir des modifications pour l'emplacement dans une certaine plage à l'aide de l'API de clôture

Si vous souhaitez détecter le moment où votre utilisateur entre un emplacement spécifique, vous

pouvez créer une clôture pour l'emplacement spécifique avec un rayon souhaité et être averti lorsque votre utilisateur entre ou quitte l'emplacement.

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

Maintenant, créez un BroadcastReciver pour recevoir les mises à jour à l'état utilisateur:

```
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;
            }
        }
    }
```

Lire API de sensibilisation Google en ligne: https://riptutorial.com/fr/android/topic/3361/api-de-sensibilisation-google

Chapitre 23: API Google Drive

Introduction

Google Drive est un service d'hébergement de fichiers créé par **Google** . Il fournit un service de stockage de fichiers et permet à l'utilisateur de télécharger des fichiers dans le cloud et de les partager avec d'autres personnes. Grâce à l'API Google Drive, nous pouvons synchroniser des fichiers entre un ordinateur ou un appareil mobile et Google Drive Cloud.

Remarques

Légal

Si vous utilisez l'API Google Drive Android dans votre application, vous devez inclure le texte d'attribution Google Play Services dans la section "Mentions légales" de votre application.

Il est recommandé d'inclure les avis juridiques en tant qu'élément de menu indépendant ou en tant qu'élément d'un élément de menu "À propos".

Vous pouvez appeler `GooglePlayServicesUtil.getOpenSourceSoftwareLicenseInfo()` pour obtenir le texte d'attribution à l'exécution.

Examples

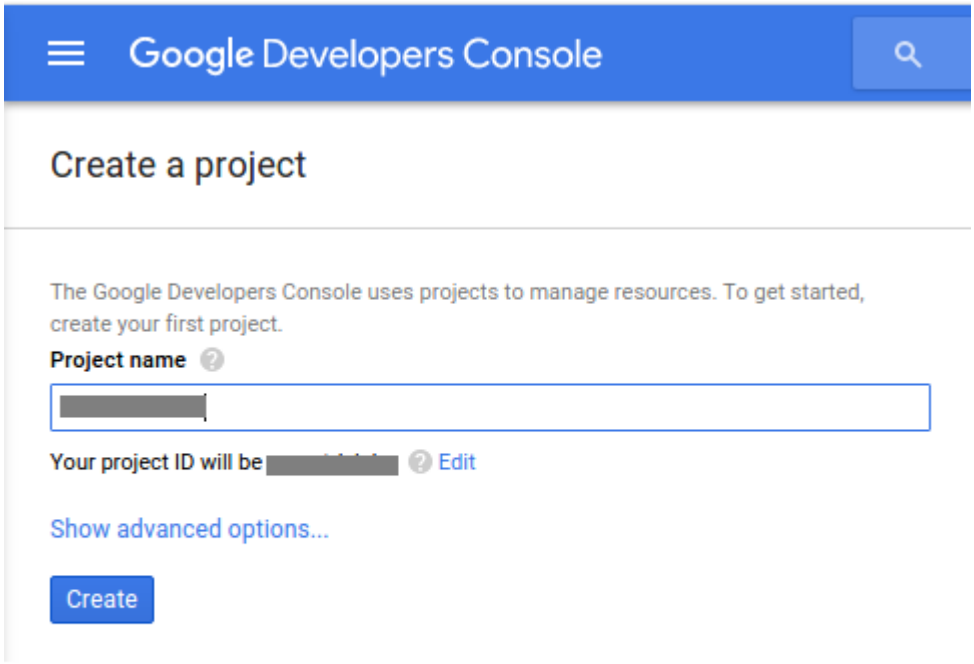
Intégrer Google Drive dans Android

Créer un nouveau projet sur Google Developer Console

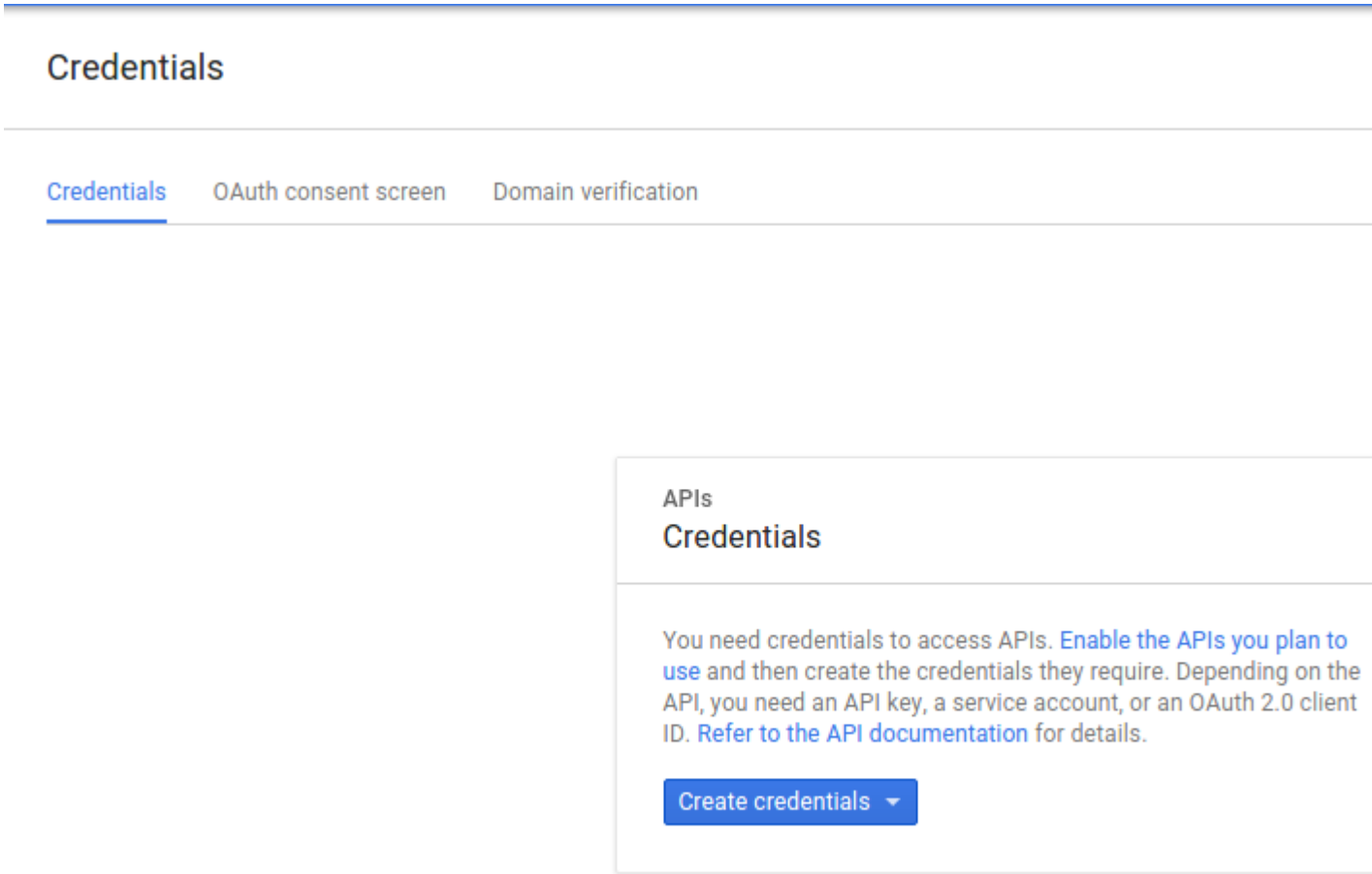
Pour intégrer une application Android à Google Drive, créez les informations d'identification du projet dans Google Developers Console. Nous devons donc créer un projet sur la console de développement Google.

Pour créer un projet sur Google Developer Console, procédez comme suit:

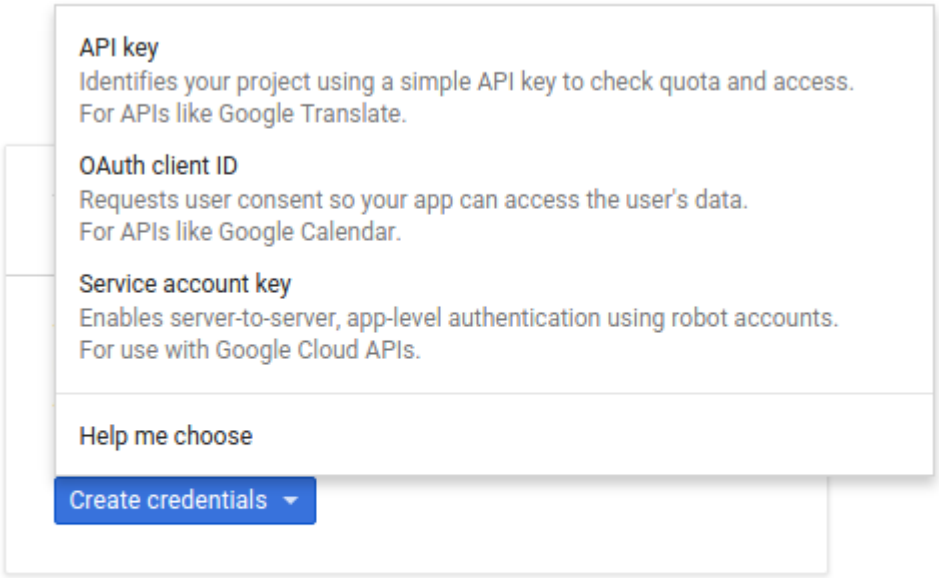
- Accédez à Google Developer Console pour Android. Remplissez votre **nom du projet** dans le champ de saisie et cliquez sur le bouton **Créer** pour créer un nouveau projet sur la console Google Developer.



- Nous devons créer des informations d'identification pour accéder à l'API. Alors, cliquez sur le bouton **Créer des identifiants** .

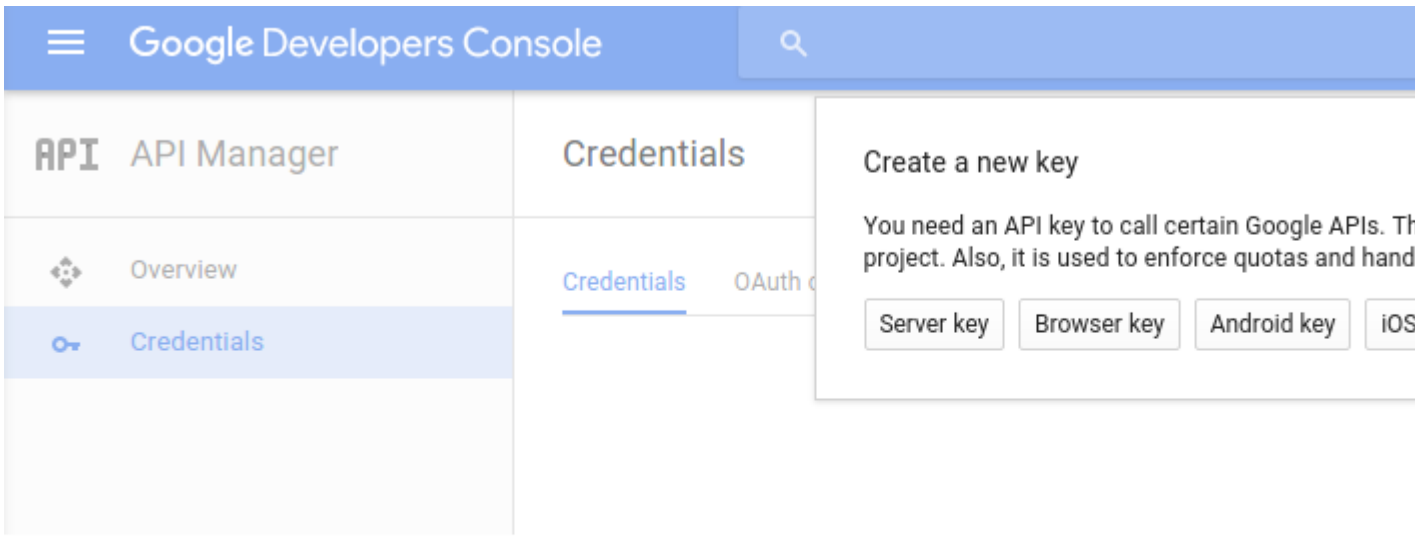


- Maintenant, une fenêtre pop s'ouvrira. Cliquez sur l'option **Clé API** dans la liste pour créer

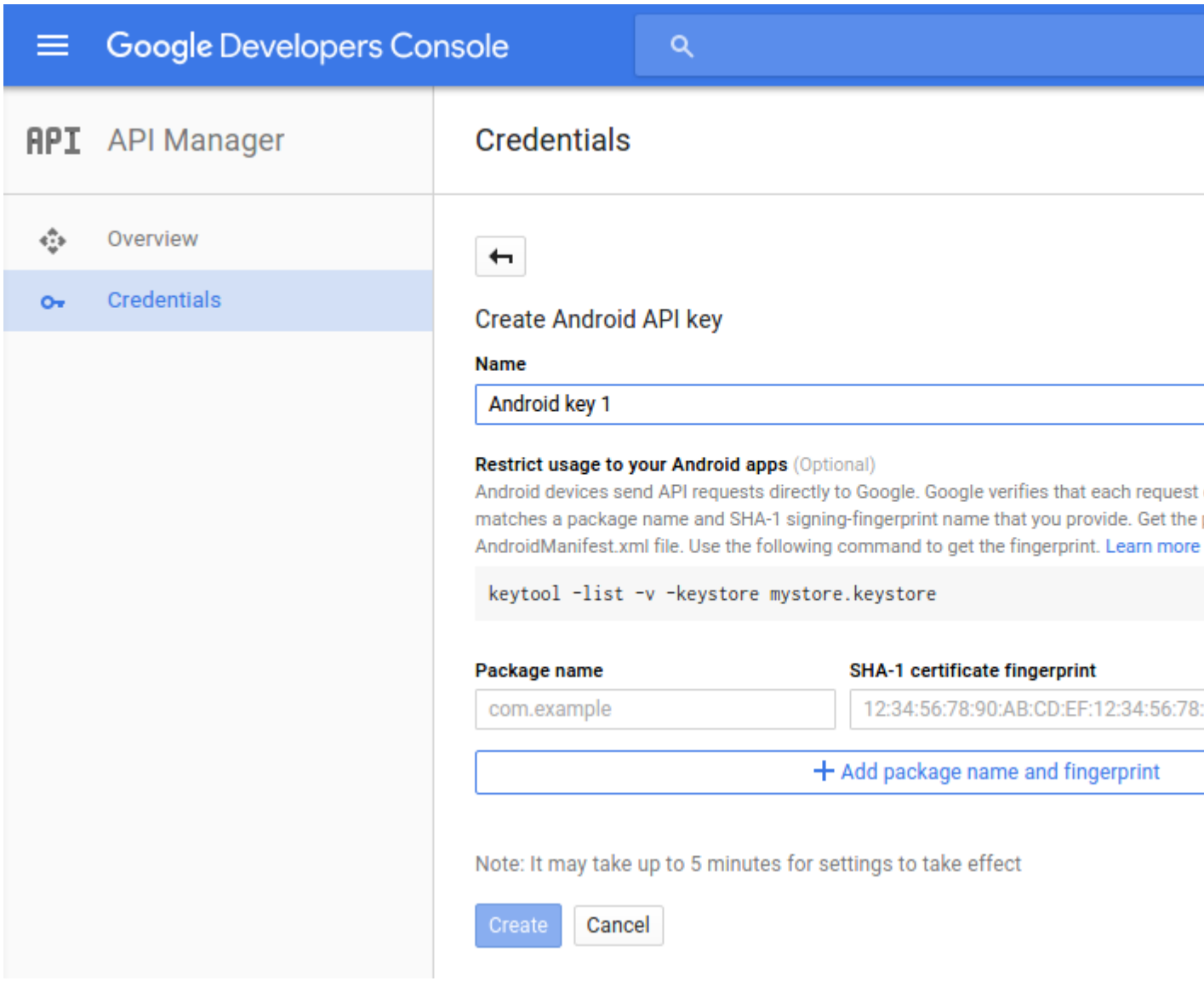


une clé API.

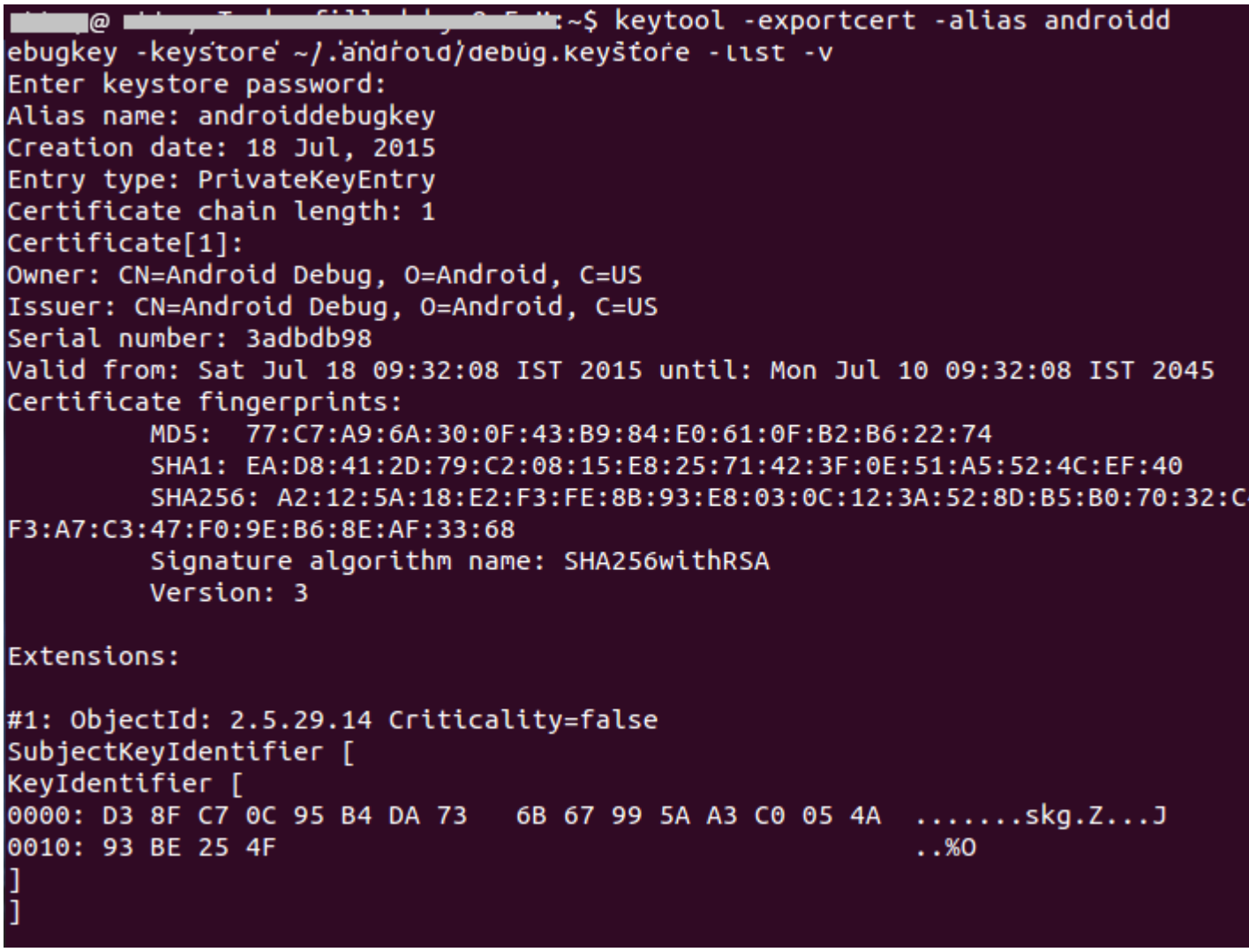
- Nous avons besoin d'une clé API pour appeler les API Google pour Android. Alors, cliquez sur la **touche Android** pour identifier votre projet Android.



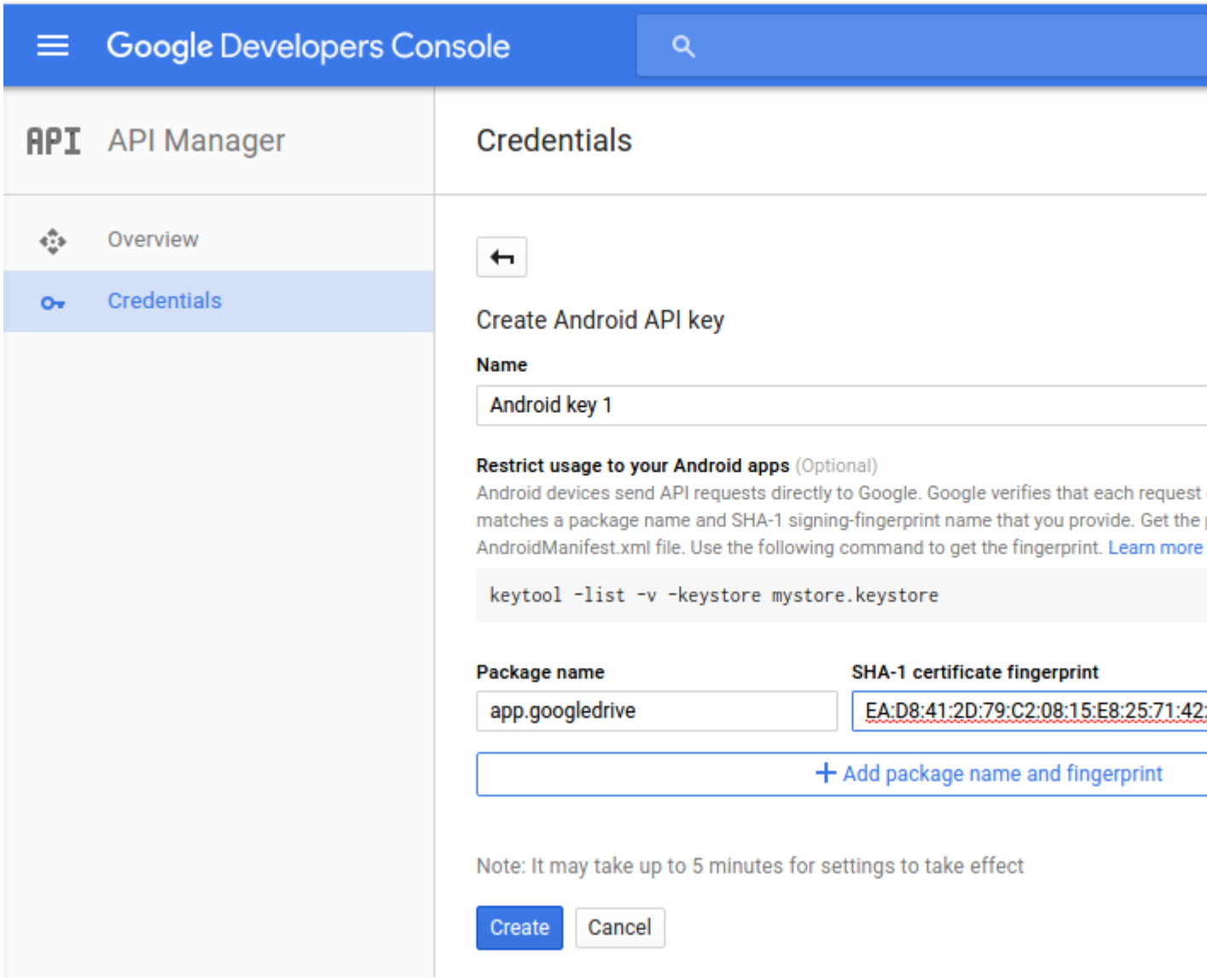
- Ensuite, nous devons ajouter le nom du package du projet Android et l' **empreinte SHA-1** dans les champs d'entrée pour créer une clé API.



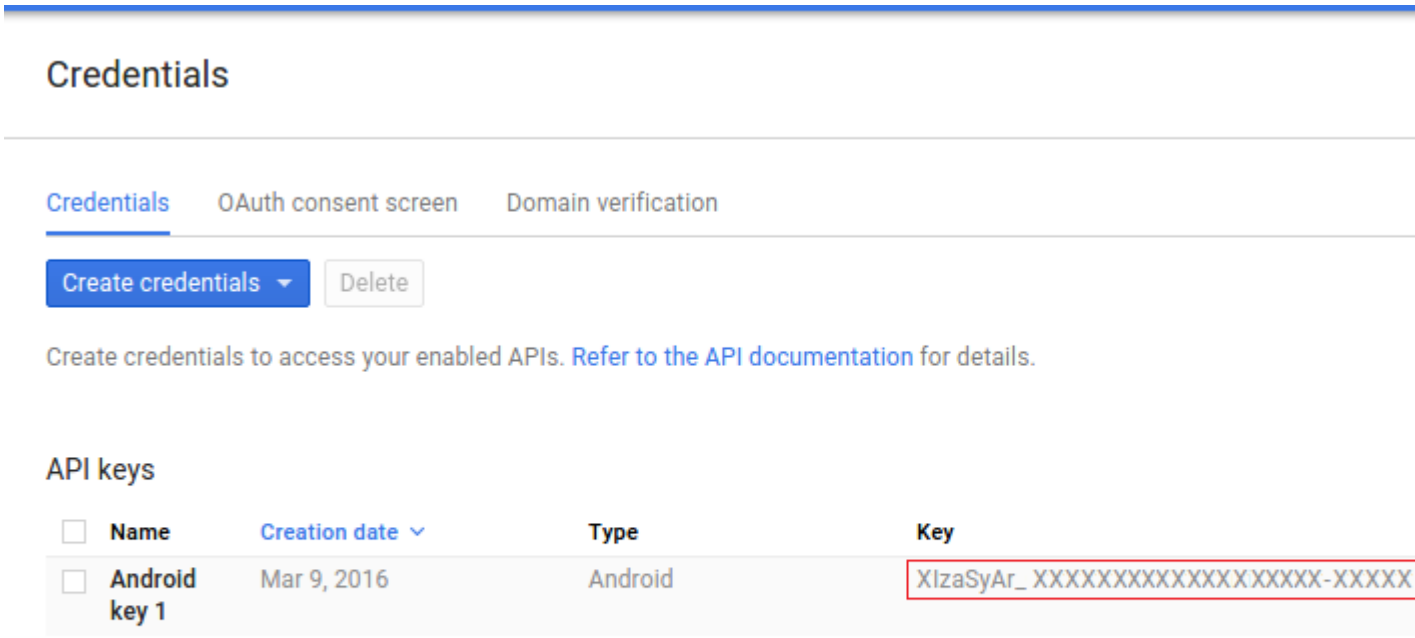
- Nous devons générer **une empreinte SHA-1** . Ouvrez donc votre terminal et lancez l' **utilitaire Keytool** pour obtenir l'empreinte SHA1. Lors de l'exécution de l'utilitaire Keytool, vous devez fournir un **mot de passe** pour le fichier de clés. Le mot de passe de développement par défaut est **«Android»** . `keytool -exportcert -alias androiddebugkey -keystore ~/.android/debug.keystore -list -v`



- Maintenant, ajoutez le **nom du package** et l' **empreinte SHA-1** dans les champs de saisie de la page d'informations d'identification. Enfin, cliquez sur le bouton Créer pour créer une clé API.



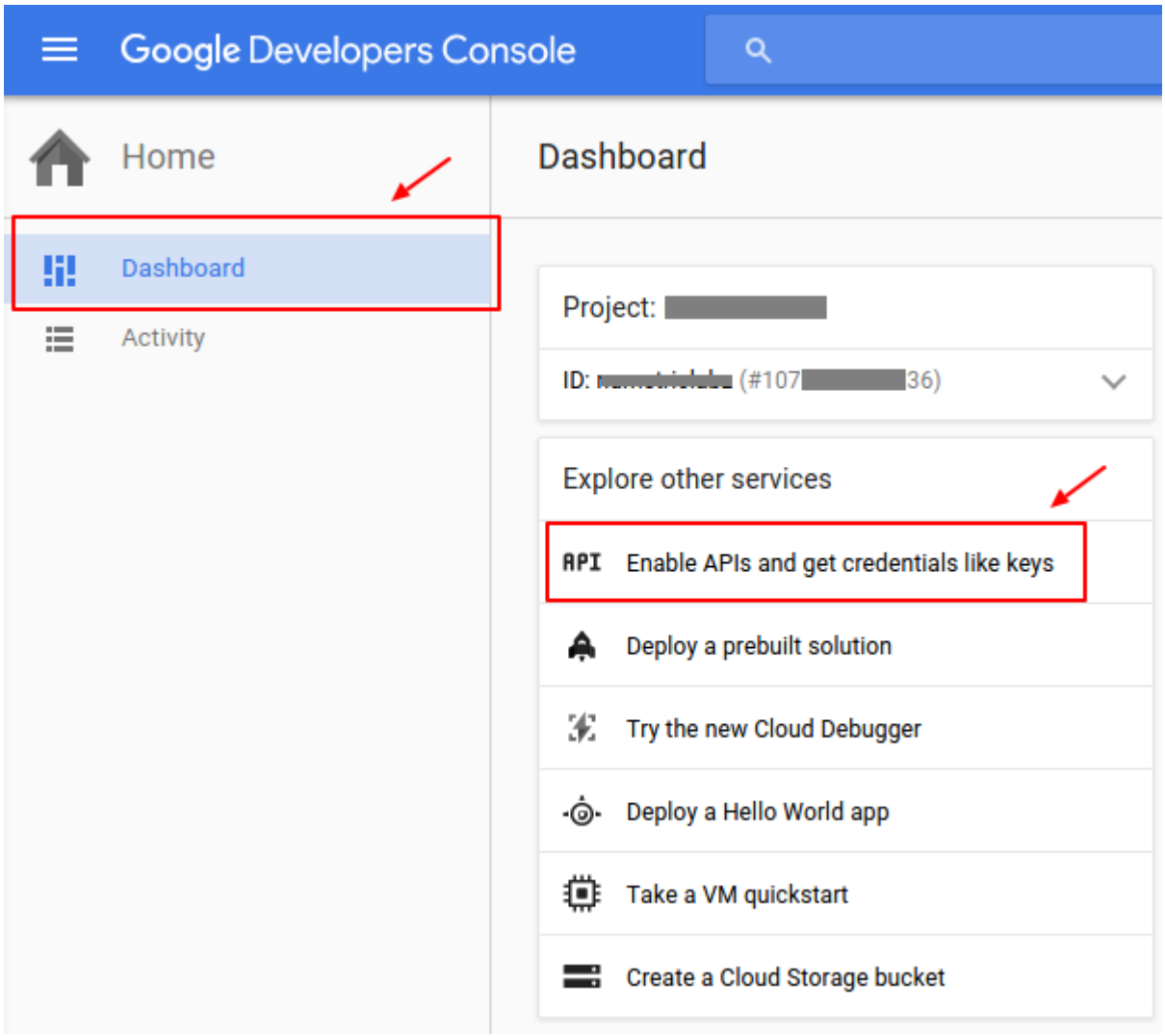
- Cela va créer une clé API pour Android. Nous utiliserons cette clé API pour intégrer l'application Android à Google Drive.



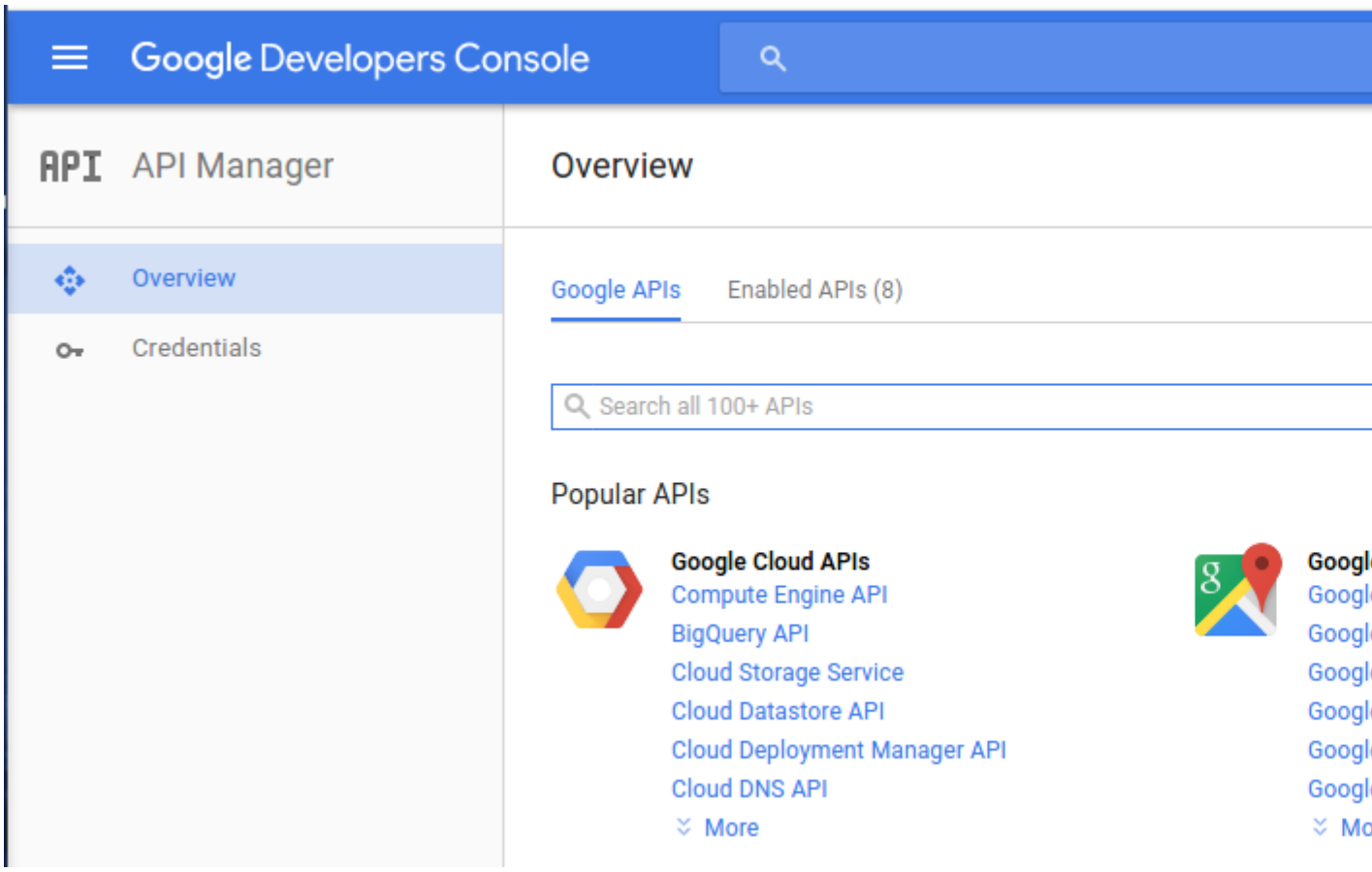
Activer l'API Google Drive

Nous devons activer Google Drive Api pour accéder aux fichiers stockés sur Google Drive depuis une application Android. Pour activer l'API Google Drive, suivez les étapes ci-dessous:

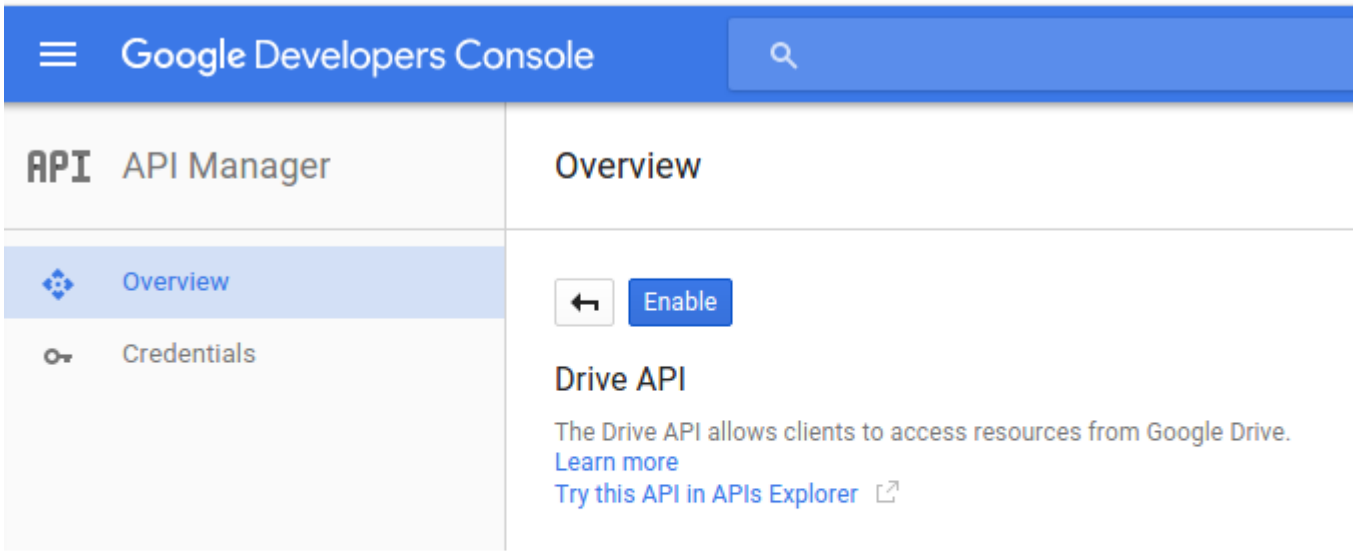
- Accédez à votre tableau de bord de la console de développeur Google et cliquez sur **Activer les API pour obtenir des informations d'identification telles que des clés.** Vous verrez alors les populaires API Google.



- Cliquez sur le lien **API du lecteur** pour ouvrir la page de présentation de l'API Google Drive.



- Cliquez sur le bouton Activer pour activer l'API Google Drive. Il permet un accès client à Google Drive.



Ajouter une autorisation Internet

App nécessite **un** accès **Internet** aux fichiers Google Drive. Utilisez le code suivant pour

configurer les autorisations Internet dans le fichier AndroidManifest.xml:

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

Ajouter des services Google Play

Nous utiliserons l' **API de services Google Play** qui comprend l' **API Google Drive Android** . Nous devons donc configurer les services de jeu Google SDK dans une application Android. Ouvrez votre `build.gradle` (module d'application) et ajoutez le SDK de services Google play en tant que dépendances.

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

Ajouter une clé API dans le fichier manifeste

Pour utiliser l'API Google dans une application Android, nous devons ajouter la clé API et la version du service Google Play dans le fichier AndroidManifest.xml. Ajoutez les balises de métadonnées correctes à l'intérieur de la balise du fichier AndroidManifest.xml.

Connectez-vous et autorisez l'API Google Drive Android

Nous devons nous authentifier et connecter l' **API Google Drive Android** à une application Android. L'autorisation de **Google Drive Android API** est gérée par **GoogleApiClient** . Nous utiliserons **GoogleApiClient** dans la méthode **onResume ()** .

```
/**
 * Called when the activity will start interacting with the user.
 * At this point your activity is at the top of the activity stack,
 * with user input going to it.
 */
@Override
protected void onResume() {
   super.onResume();
   if (mGoogleApiClient == null) {

       /**
        * Create the API client and bind it to an instance variable.
        * We use this instance as the callback for connection and connection failures.
        * Since no account name is passed, the user is prompted to choose.
        */
         mGoogleApiClient = new GoogleApiClient.Builder(this)
                 .addApi(Drive.API)
                 .addScope(Drive.SCOPE_FILE)
                 .addConnectionCallbacks(this)
                 .addOnConnectionFailedListener(this)
                 .build();
        }

        mGoogleApiClient.connect();
    }
```

Déconnecter l'API Android de Google Deive

Lorsque l'activité s'arrête, nous allons déconnecter la connexion API Google Drive Android avec l'application Android en appelant la méthode **disconnect () dans la** méthode **onStop () de l'activité** .

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

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

Implémenter des rappels de connexion et un écouteur avec échec de connexion

Nous implémenterons les callbacks de connexion et le listener d'échec de connexion du client API Google dans le fichier MainActivity.java pour connaître l'état de la connexion du client API Google. Ces écouteurs fournissent la **méthode onConnected (), onConnectionFailed (), onConnectionSuspended ()** pour gérer les problèmes de connexion entre l'application et le lecteur.

Si l'utilisateur a autorisé l'application, la méthode **onConnected ()** est appelée. Si l'utilisateur n'a pas autorisé l'application, la méthode **onConnectionFailed ()** est appelée et une boîte de dialogue s'affiche pour indiquer que votre application n'est pas autorisée à accéder à Google Drive. **Si la** connexion est suspendue, la méthode **onConnectionSuspended ()** est appelée.

Vous devez implémenter **ConnectionCallbacks** et **OnConnectionFailedListener** dans votre activité. Utilisez le code suivant dans votre fichier Java.

```
@Override
    public void onConnectionFailed(ConnectionResult result) {

        // Called whenever the API client fails to connect.
        Log.i(TAG, "GoogleApiClient connection failed:" + result.toString());

        if (!result.hasResolution()) {

            // show the localized error dialog.
            GoogleApiAvailability.getInstance().getErrorDialog(this, result.getErrorCode(),
0).show();
            return;
        }

        /**
         *  The failure has a resolution. Resolve it.
         *  Called typically when the app is not yet authorized, and an  authorization
         *  dialog is displayed to the user.
         */

        try {
```

```
            result.startResolutionForResult(this, REQUEST_CODE_RESOLUTION);

        } catch (SendIntentException e) {

            Log.e(TAG, "Exception while starting resolution activity", e);
        }
    }

   /**
    * It invoked when Google API client connected
    * @param connectionHint
    */
    @Override
    public void onConnected(Bundle connectionHint) {

        Toast.makeText(getApplicationContext(), "Connected", Toast.LENGTH_LONG).show();
    }

   /**
    * It invoked when connection suspended
    * @param cause
    */
    @Override
    public void onConnectionSuspended(int cause) {

        Log.i(TAG, "GoogleApiClient connection suspended");
    }
```

Créer un fichier sur Google Drive

Nous allons ajouter un fichier sur Google Drive. Nous allons utiliser la `createFile()` d'un objet `Drive` pour créer un fichier par programme sur Google Drive. Dans cet exemple nous ajoutons un nouveau fichier texte dans le dossier racine de l'utilisateur. Lorsqu'un fichier est ajouté, nous devons spécifier le jeu initial de métadonnées, le contenu du fichier et le dossier parent.

Nous devons créer une méthode de rappel `CreateMyFile()` et, au sein de cette méthode, utiliser l'objet `Drive` pour extraire une ressource `DriveContents` . Ensuite, nous passons le client API à l'objet `Drive` et appelons la méthode de rappel `driveContentsCallback` pour gérer le résultat de `DriveContents` .

Une ressource `DriveContents` contient une copie temporaire du flux binaire du fichier, uniquement disponible pour l'application.

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

Gestionnaire de résultat de DriveContents

La gestion de la réponse nécessite de vérifier si l'appel a réussi ou non. Si l'appel a réussi, nous

pouvons récupérer la ressource `DriveContents` .

Nous allons créer un gestionnaire de résultats de `DriveContents` . Dans cette méthode, nous appelons la méthode `CreateFileOnGoogleDrive()` et transmettons le résultat de `DriveContentsResult` :

```
/**
 * This is the Result result handler of Drive contents.
 * This callback method calls the CreateFileOnGoogleDrive() method.
 */
final ResultCallback<DriveContentsResult> driveContentsCallback =
         new ResultCallback<DriveContentsResult>() {
            @Override
            public void onResult(DriveContentsResult result) {
                if (result.getStatus().isSuccess()) {
                    if (fileOperation == true){
                        CreateFileOnGoogleDrive(result);
                    }
                }
            }
        };
```

Créer un fichier par programme

Pour créer des fichiers, nous devons utiliser un objet `MetadataChangeSet` . En utilisant cet objet, nous définissons le titre (nom du fichier) et le type de fichier. En outre, nous devons utiliser la `createFile()` de la classe `DriveFolder` et transmettre l'API cliente Google, l'objet `MetaDataChangeSet` et `driveContents` pour créer un fichier. Nous appelons le rappel du gestionnaire de résultats pour gérer le résultat du fichier créé.

Nous utilisons le code suivant pour créer un nouveau fichier texte dans le dossier racine de l'utilisateur:

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

Gérer le résultat du fichier créé

Le code suivant créera une méthode de rappel pour gérer le résultat du fichier créé:

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

Lire API Google Drive en ligne: https://riptutorial.com/fr/android/topic/10646/api-google-drive

Chapitre 24: API Twitter

Examples

Créer une connexion avec le bouton Twitter et y attacher un rappel

1. Dans votre mise en page, ajoutez un bouton Connexion avec le code suivant:

```
<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. Dans l'activité ou le fragment qui affiche le bouton, vous devez créer et joindre un rappel au bouton de connexion, comme suit:

```
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. Transmettez le résultat de l'activité d'authentification au bouton:

```
@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);
}
```

Remarque: Si vous utilisez le TwitterLoginButton dans un fragment, utilisez plutôt les étapes

suivantes:

```
 @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. Ajoutez les lignes suivantes à vos dépendances **build.gradle** :

```
 apply plugin: 'io.fabric'

 repositories {
     maven { url 'https://maven.fabric.io/public' }
 }

 compile('com.twitter.sdk.android:twitter:1.14.1@aar') {
     transitive = true;
 }
```

Lire API Twitter en ligne: https://riptutorial.com/fr/android/topic/4801/api-twitter

Chapitre 25: Applications compatibles avec la construction en amont

Examples

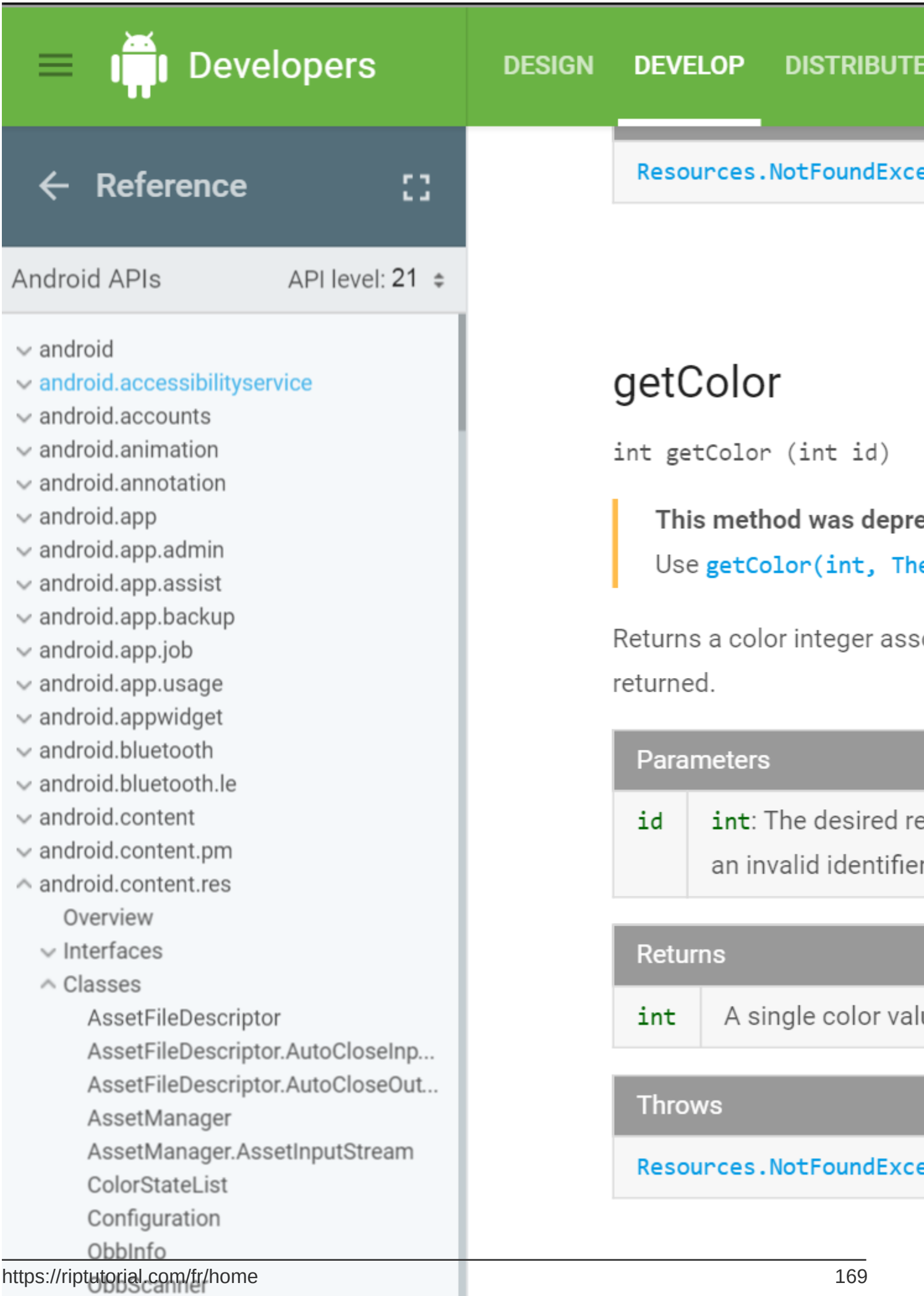
Comment gérer les API obsolètes

Il est peu probable qu'un développeur ne rencontre pas une API obsolète au cours d'un processus de développement. Un élément de programme obsolète est un élément que les programmeurs sont déconseillés d'utiliser, généralement parce qu'il est dangereux ou parce qu'il existe une meilleure alternative. Les compilateurs et les analyseurs (comme LINT) avertissent lorsqu'un élément de programme obsolète est utilisé ou remplacé dans un code non obsolète.

Une API obsolète est généralement identifiée dans Android Studio à l'aide d'un barré. Dans l'exemple ci-dessous, la méthode `.getColor(int id)` est obsolète:

```
getResources().~~getColor~~(R.color.colorAccent));
```

Si possible, les développeurs sont encouragés à utiliser des API et des éléments alternatifs. Il est possible de vérifier la compatibilité ascendante d'une bibliothèque en visitant la documentation Android de la bibliothèque et en consultant la section "Ajouté au niveau de l'API x":


Developers
DESIGN
DEVELOP
DISTRIBUTE
Reference
Android APIs
API level: 21
android
android.accessibilityservice
android.accounts
android.animation
android.annotation
android.app
android.app.admin
android.app.assist
android.app.backup
android.app.job
android.app.usage
android.appwidget
android.bluetooth
android.bluetooth.le
android.content
android.content.pm
android.content.res
Overview
Interfaces
Classes
AssetFileDescriptor
AssetFileDescriptor.AutoCloseInp...
AssetFileDescriptor.AutoCloseOut...
AssetManager
AssetManager.AssetInputStream
ColorStateList
Configuration
ObbInfo
ObbScanner
Resources.NotFoundExce
getColor
int getColor (int id)
This method was depre
Use getColor(int, The
Returns a color integer ass
returned.
Parameters
id
int: The desired re
an invalid identifier
Returns
int
A single color valu
Throws
Resources.NotFoundExce

https://riptutorial.com/fr/android/topic/4291/applications-compatibles-avec-la-construction-en-amont

Chapitre 26: Architecture MVP

Introduction

Cette rubrique fournira une architecture MVP (Model - View - Presenter) avec différents exemples.

Remarques

Il existe de nombreuses façons de concevoir une application Android. Mais tous ne sont pas testables et nous permettent de structurer notre code pour que l'application soit facile à tester. L'idée clé d'une architecture à tester est de séparer des parties de l'application, ce qui les rend plus faciles à entretenir, à étendre et à tester séparément les unes des autres.

Définition MVP

Modèle

Dans une application dotée d'une bonne architecture en couches, ce modèle ne serait que la passerelle vers la couche de domaine ou la logique métier. Voir comme le fournisseur des données que nous voulons afficher dans la vue.

Vue

La vue, généralement implémentée par une `Activity` ou un `Fragment` , contiendra une référence au *présentateur* . La seule chose à faire est d'appeler une méthode du présentateur chaque fois qu'il y a une action d'interface.

Présentateur

Le présentateur est responsable d'agir en tant qu'intermédiaire entre View et Model. Il récupère les données du modèle et les retourne au format de la vue. Mais contrairement au MVC typique, il décide également de ce qui se passe lorsque vous interagissez avec la vue.

* Définitions tirées de l'article d' Antonio Leiva.

Structure de l'application *recommandée* (non requise)

L'application doit être structurée par package *par fonctionnalité* . Cela améliore la lisibilité et modularise l'application de telle sorte que certaines parties peuvent être modifiées indépendamment les unes des autres. Chaque fonctionnalité clé de l'application se trouve dans son propre package Java.

Examples

Exemple de connexion dans le modèle MVP (Model View Presenter)

Voyons MVP en action en utilisant un simple écran de connexion. Il existe deux `Button` : un pour l'action de connexion et un autre pour un écran d'enregistrement; deux `EditText` : un pour le courrier électronique et l'autre pour le mot de passe.

LoginFragment (The View)

```
public class LoginFragment extends Fragment implements LoginContract.PresenterToView,
View.OnClickListener {

    private View view;
    private EditText email, password;
    private Button login, register;

    private LoginContract.ToPresenter presenter;

    @Nullable
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
        return inflater.inflate(R.layout.fragment_login, container, false);
    }

    @Override
    public void onViewCreated(View view, @Nullable Bundle savedInstanceState) {
        email = (EditText) view.findViewById(R.id.email_et);
        password = (EditText) view.findViewById(R.id.password_et);
        login = (Button) view.findViewById(R.id.login_btn);
        login.setOnClickListener(this);
        register = (Button) view.findViewById(R.id.register_btn);
        register.setOnClickListener(this);

        presenter = new LoginPresenter(this);

        presenter.isLoggedIn();

    }

    @Override
    public void onLoginResponse(boolean isLoginSuccess) {
        if (isLoginSuccess) {
            startActivity(new Intent(getActivity(), MapActivity.class));
            getActivity().finish();
        }
    }

    @Override
    public void onError(String message) {
        Toast.makeText(getActivity(), message, Toast.LENGTH_SHORT).show();
    }

    @Override
    public void isLoggedIn(boolean isLoggedIn) {
        if (isLoggedIn) {
```

```
            startActivity(new Intent(getActivity(), MapActivity.class));
            getActivity().finish();
        }
    }

    @Override
    public void onClick(View view) {
        switch (view.getId()) {
            case R.id.login_btn:
                LoginItem loginItem = new LoginItem();
                loginItem.setPassword(password.getText().toString().trim());
                loginItem.setEmail(email.getText().toString().trim());
                presenter.login(loginItem);
                break;
            case R.id.register_btn:
                startActivity(new Intent(getActivity(), RegisterActivity.class));
                getActivity().finish();
                break;
        }
    }
}
```

LoginPresenter (Le présentateur)

```
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);
    }
}
```

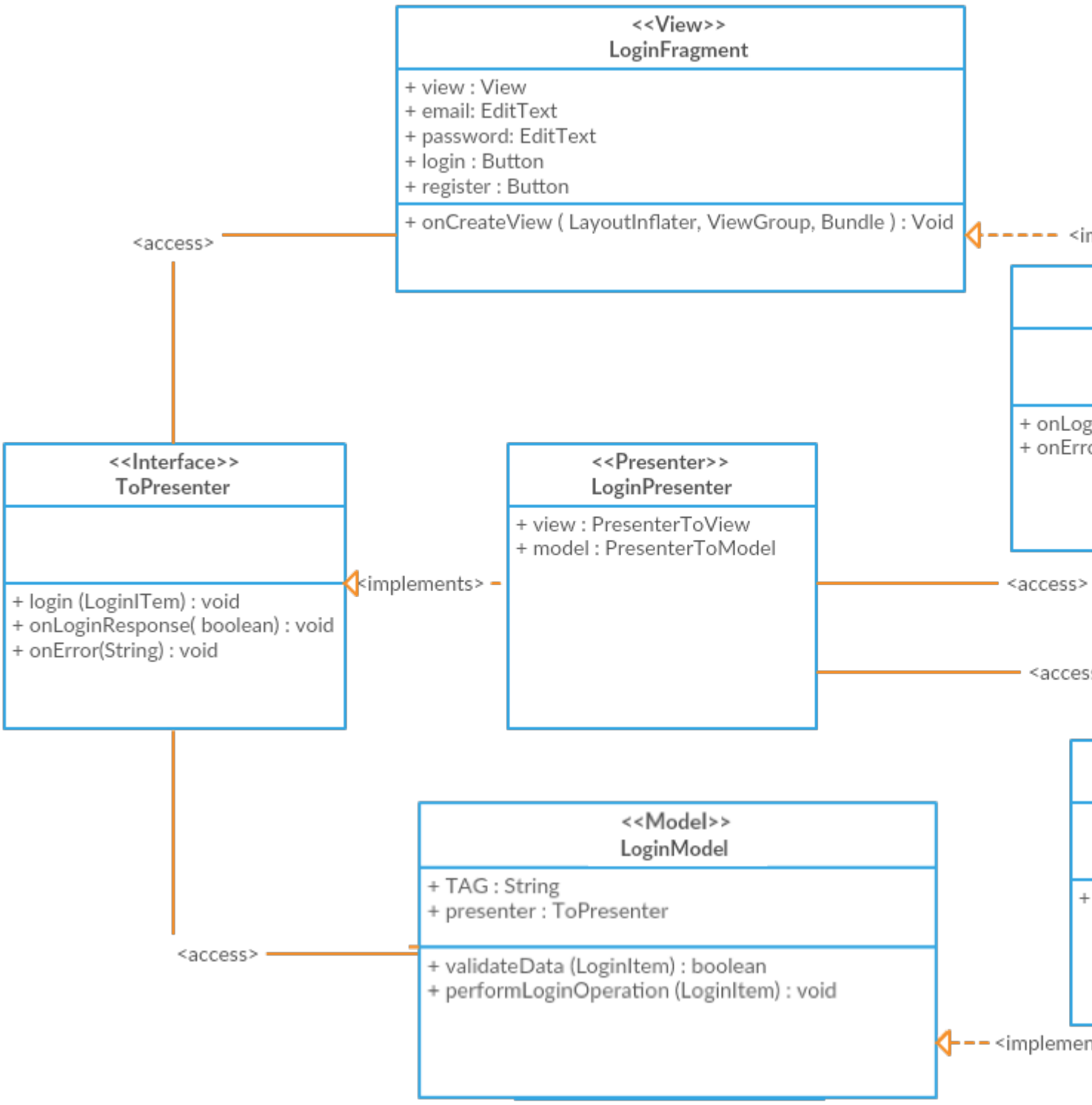
LoginModel (Le modèle)

```
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);
    }
}
```

Diagramme de classe

Voyons l'action sous la forme d'un diagramme de classes.

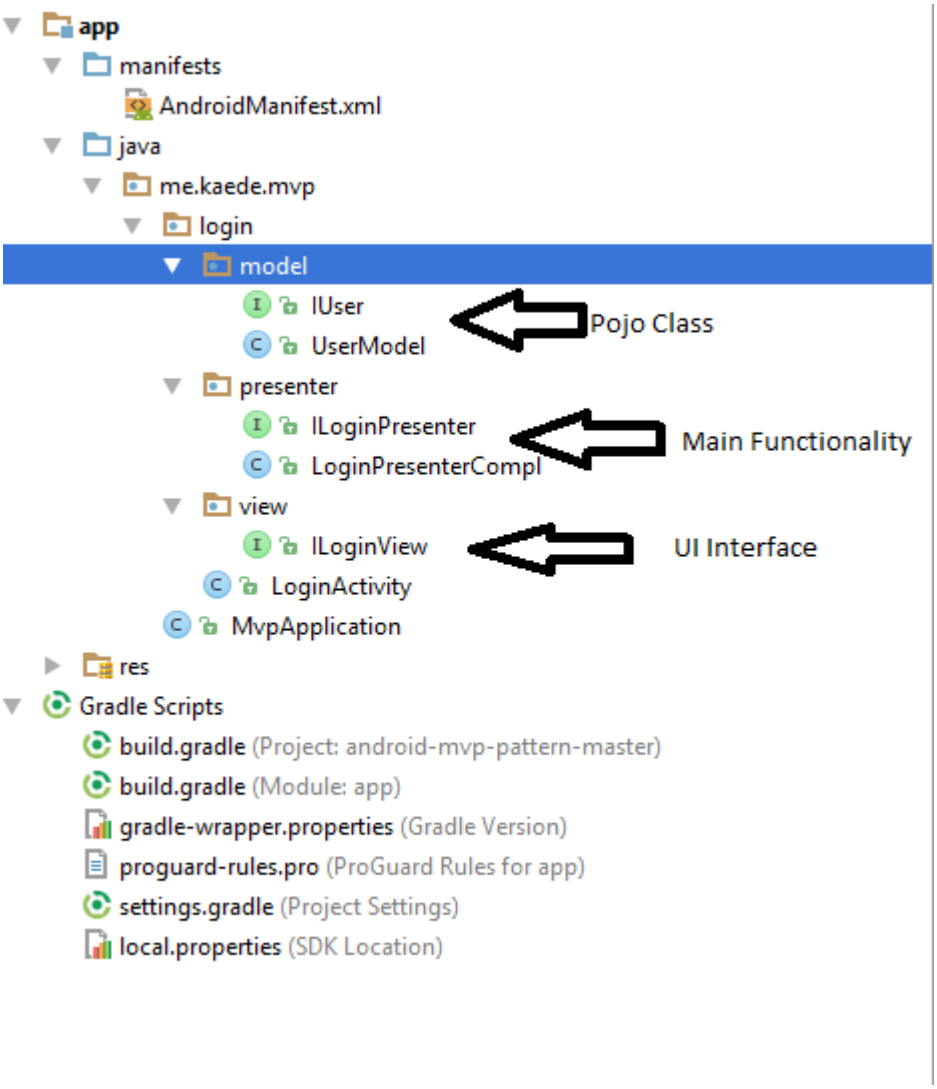


Remarques:

- Cet exemple utilise Volley pour la communication réseau, mais cette bibliothèque n'est pas requise pour MVP
- `UrlUtils` est une classe qui contient tous les liens pour mes points de terminaison API
- `ResponseErrorListener.ErrorListener` est une `interface` qui `ErrorResponse` erreurs dans `ErrorResponse` qui `implements` Volley's `Response.ErrorListener` ; ces classes ne sont pas incluses ici car elles ne font pas directement partie de cet exemple

Exemple de connexion simple dans MVP

Structure de colis requise



XML activity_login

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center_vertical"
    android:orientation="vertical"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin">

    <EditText
        android:id="@+id/et_login_username"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="USERNAME" />

    <EditText
```

```
        android:id="@+id/et_login_password"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="PASSWORD" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">

        <Button
            android:id="@+id/btn_login_login"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginRight="4dp"
            android:layout_weight="1"
            android:text="Login" />

        <Button
            android:id="@+id/btn_login_clear"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginLeft="4dp"
            android:layout_weight="1"
            android:text="Clear" />
    </LinearLayout>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="3dp"
        android:text="correct user: mvp, mvp" />

    <ProgressBar
        android:id="@+id/progress_login"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="40dp" />

</LinearLayout>
```

Classe d'activité 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);
    }
}
```

Créer une interface ILoginView

Créez une interface `ILoginView` pour les informations de mise à jour de Presenter sous le dossier d'affichage, comme suit:

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

Création d'une interface ILoginPresenter

Créez une interface `ILoginPresenter` afin de communiquer avec `LoginActivity` (Views) et créez la classe `LoginPresenterCompl` pour gérer la fonctionnalité de connexion et rendre compte de l'activité. La classe `LoginPresenterCompl` implémente l'interface `ILoginPresenter` :

ILoginPresenter.class

```
public interface ILoginPresenter {
    void clear();
    void doLogin(String name, String passwd);
    void setProgressBarVisiblity(int visiblity);
}
```

LoginPresenterCompl.class

```
public class LoginPresenterCompl implements ILoginPresenter {
    ILoginView iLoginView;
    IUser user;
    Handler handler;

    public LoginPresenterCompl(ILoginView iLoginView) {
        this.iLoginView = iLoginView;
        initUser();
        handler = new Handler(Looper.getMainLooper());
    }

    @Override
    public void clear() {
        iLoginView.onClearText();
    }

    @Override
    public void doLogin(String name, String passwd) {
        Boolean isLoginSuccess = true;
        final int code = user.checkUserValidity(name,passwd);
        if (code!=0) isLoginSuccess = false;
        final Boolean result = isLoginSuccess;
        handler.postDelayed(new Runnable() {
```

```
            @Override
            public void run() {
                    iLoginView.onLoginResult(result, code);
            }
        }, 5000);
    }

    @Override
    public void setProgressBarVisiblity(int visiblity){
        iLoginView.onSetProgressBarVisibility(visiblity);
    }

    private void initUser(){
        user = new UserModel("mvp","mvp");
    }
}
```

Créer un UserModel

Créez un `UserModel` qui ressemble à une classe Pojo pour `LoginActivity` . Créez une interface `IUser` pour les validations Pojo:

UserModel.class

```
public class UserModel implements IUser {
String name;
String passwd;

public UserModel(String name, String passwd) {
    this.name = name;
    this.passwd = passwd;
}

@Override
public String getName() {
    return name;
}

@Override
public String getPasswd() {
    return passwd;
}

@Override
public int checkUserValidity(String name, String passwd){
    if (name==null||passwd==null||!name.equals(getName())||!passwd.equals(getPasswd())){
        return -1;
    }
    return 0;
}
```

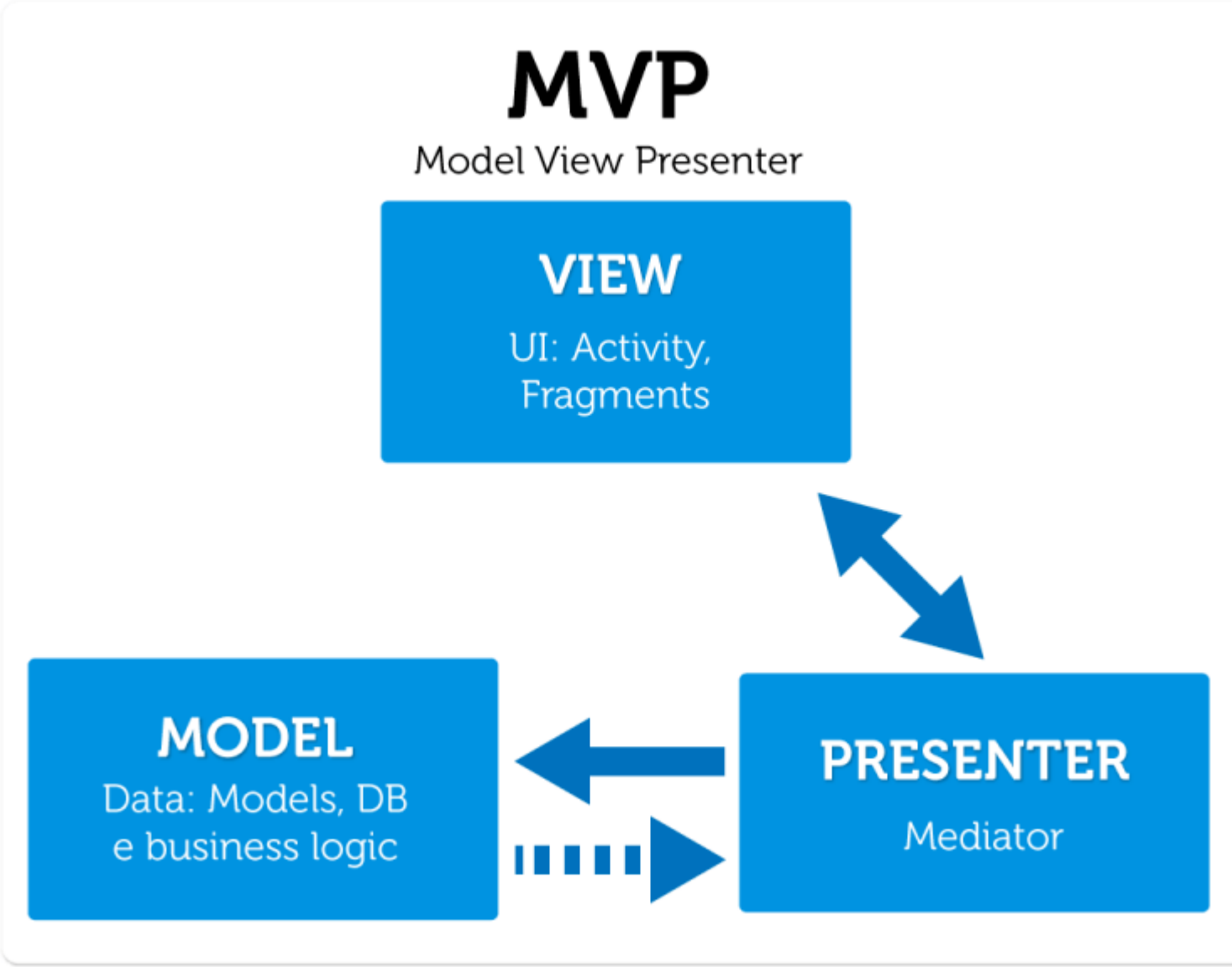
IUser.class

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

    String getPasswd();

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

MVP

Un modèle-vue-présentateur (MVP) est une dérivation du modèle architectural modèle – vue-contrôleur (MVC). Il est principalement utilisé pour créer des interfaces utilisateur et offre les avantages suivants:

- Les vues sont plus séparées des modèles. Le présentateur est le médiateur entre le modèle et la vue.
- Il est plus facile de créer des tests unitaires.
- En général, il existe une correspondance univoque entre View et Presenter, avec la possibilité d'utiliser plusieurs présentateurs pour des vues complexes.



Lire Architecture MVP en ligne: https://riptutorial.com/fr/android/topic/4615/architecture-mvp

Chapitre 27: AsyncTask

Paramètres

Paramètre	Détails
Params	le type des paramètres envoyés à la tâche lors de l'exécution.
Le progrès	le type des unités de progression publiées lors du calcul en arrière-plan
Résultat	le type du résultat du calcul en arrière-plan.

Examples

Utilisation de base

Dans Activités et services Android, la plupart des rappels sont exécutés sur le thread principal . Cela simplifie la mise à jour de l'interface utilisateur, mais l'exécution de tâches lourdes sur le processeur ou l'E / S sur le thread principal peut entraîner une pause de votre interface utilisateur et une absence de réponse (documentation officielle sur ce qui se passe alors).

Vous pouvez remédier à cela en plaçant ces tâches plus lourdes sur un thread d'arrière-plan.

Pour ce faire, vous pouvez utiliser une AsyncTask , qui fournit un cadre facilitant l'utilisation d'un thread en arrière-plan, et effectue également des tâches UI Thread avant, pendant et après la fin du thread Thread.

Méthodes pouvant être remplacées lors de l'extension d' `AsyncTask` :

- `onPreExecute()` : invoqué sur le **thread d'interface utilisateur** avant l' **exécution de** la tâche
- `doInBackground()` : invoqué sur **le thread d'arrière-plan** immédiatement après l' `onPreExecute()` de `onPreExecute()` .
- `onProgressUpdate()` : invoqué sur le **thread d'interface utilisateur** après un appel à `publishProgress(Progress...)` .
- `onPostExecute()` : invoqué sur le **thread d'interface utilisateur une** fois le calcul en arrière-plan terminé

Exemple

```
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.");
        }
    }

}
```

Usage:

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

ou simplement:

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

Remarque

Lors de la définition d'une `AsyncTask` nous pouvons passer trois types entre `< >` .
Défini comme `<Params, Progress, Result>` (voir la **section Paramètres**)

Dans l'exemple précédent, nous avons utilisé les types `<File, Void, String>` :

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

`Void` est utilisé lorsque vous souhaitez marquer un type comme non utilisé.

Notez que vous ne pouvez pas passer de types primitifs (c.-à-d. `int` , `float` et 6 autres) en tant que paramètres. Dans de tels cas, vous devez passer leurs classes wrapper , par exemple `Integer` au lieu de `int` , ou `Float` au lieu de `float` .

Le cycle de vie AsyncTask et Activity

AsyncTasks ne suit pas le cycle de vie des instances d'activité. Si vous démarrez une tâche AsyncTask dans une activité et que vous faites pivoter le périphérique, l'activité sera détruite et une nouvelle instance sera créée. Mais la AsyncTask ne mourra pas. Il continuera à vivre jusqu'à la fin.

Solution: AsyncTaskLoader

Une sous-classe de chargeurs est AsyncTaskLoader. Cette classe remplit la même fonction que AsyncTask, mais beaucoup mieux. Il peut gérer plus facilement les modifications de la configuration d'activité et se comporte au cours des cycles de vie des fragments et des activités. La bonne chose est que AsyncTaskLoader peut être utilisé dans toutes les situations où AsyncTask est utilisé. Chaque fois que des données doivent être chargées en mémoire pour que l'activité / le fragment puisse être géré, AsyncTaskLoader peut mieux faire le travail.

Annuler AsyncTask

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

Cela n'arrête pas votre tâche si elle était en cours, elle définit simplement le drapeau annulé qui peut être vérifié en vérifiant la valeur de retour de `isCancelled()` (en supposant que votre code est en cours d'exécution) en procédant comme `isCancelled()` :

```
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;
            }
        }
    }
}
```

Remarque

Si une AsyncTask est annulée alors que `doInBackground(Params... params)` est toujours en cours d'exécution, la méthode `onPostExecute(Result result)` ne sera **PAS** appelée après le `doInBackground(Params... params)` de `doInBackground(Params... params)` . AsyncTask appellera plutôt le `onCancelled(Result result)` pour indiquer que la tâche a été annulée pendant l'exécution.

Progrès de la publication

Parfois, nous devons mettre à jour la progression du calcul effectué par une `AsyncTask` . Cette progression peut être représentée par une chaîne, un entier, etc. Pour ce faire, nous devons utiliser deux fonctions. Tout d'abord, nous devons définir la fonction `onProgressUpdate` dont le type de paramètre est identique à celui du second paramètre de notre `AsyncTask` .

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

Deuxièmement, nous devons utiliser la fonction `publishProgress` nécessairement sur la fonction `doInBackground` , et c'est tout, la méthode précédente fera tout le travail.

```
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;
 }
```

Télécharger l'image en utilisant AsyncTask dans Android

Ce tutoriel explique comment télécharger Image en utilisant AsyncTask dans Android. L'exemple ci-dessous télécharge l'image tout en affichant la barre de progression pendant le téléchargement.

Comprendre Android AsyncTask

La tâche asynchrone vous permet d'implémenter MultiThreading sans que les mains soient sales dans les threads. AsyncTask permet une utilisation correcte et facile du thread de l'interface utilisateur. Il permet d'effectuer des opérations en arrière-plan et de transmettre les résultats sur le thread d'interface utilisateur. Si vous faites quelque chose d'isolement lié à l'interface utilisateur, par exemple en téléchargeant des données pour les présenter dans une liste, utilisez AsyncTask.

- AsyncTasks devrait idéalement être utilisé pour des opérations courtes (quelques secondes au maximum).
- Une tâche asynchrone est définie par 3 types génériques, appelés Params, Progress et Result, et 4 étapes, appelées `onPreExecute()` , `doInBackground()` , `onProgressUpdate()` et `onPostExecute()` .
- Dans `onPreExecute()` vous pouvez définir du code, qui doit être exécuté avant le traitement en arrière-plan.
- doInBackground a un code qui doit être exécuté en arrière-plan, ici dans `doInBackground()` nous pouvons envoyer des résultats à plusieurs fois au thread d'événement par la méthode publishProgress ().
- `onProgressUpdate()` méthode `onProgressUpdate()` reçoit les mises à jour de progression de la méthode `doInBackground()` , publiée via la méthode `publishProgress()` . Cette méthode peut utiliser cette mise à jour pour mettre à jour le thread d'événement.
- `onPostExecute()` méthode `onPostExecute()` gère les résultats renvoyés par la méthode `doInBackground()` .
- Les types génériques utilisés sont
  - Params, le type des paramètres envoyés à la tâche lors de l'exécution
  - Progress, le type des unités de progression publiées lors du calcul en arrière-plan.
  - Résultat, le type du résultat du calcul en arrière-plan.
- Si une tâche asynchrone n'utilise aucun type, elle peut alors être marquée comme type d'annulation.
- Une tâche asynchrone en cours d'exécution peut être annulée en appelant la méthode `cancel(boolean)` .

Téléchargement d'image en utilisant Android AsyncTask

votre mise en page .xml

```
<?xml version="1.0" encoding="utf-8"?>

<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

<Button
    android:id="@+id/downloadButton"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
```

```
    android:text="Click Here to Download" />

<ImageView
    android:id="@+id/imageView"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:contentDescription="Your image will appear here" />

</LinearLayout>
```

classe .java

```
package com.javatechig.droid;

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

public class ImageDownladerActivity extends Activity {

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

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

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

        imageDownloaderBtn.setOnClickListener(new OnClickListener() {

            @Override
            public void onClick(View v) {
                // TODO Auto-generated method stub
                new ImageDownloader().execute(downloadUrl);
            }

        });
    }

    private class ImageDownloader extends AsyncTask {

        @Override
```

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

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

        }

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

        }

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

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

                HttpResponse response = client.execute(getRequest);

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

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

                }

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

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

                        return bitmap;
                    } finally {
                        if (inputStream != null) {
                            inputStream.close();
                        }
                        entity.consumeContent();
                    }
                }
            } catch (Exception e) {
```

```
                // You Could provide a more explicit error message for IOException
                getRequest.abort();
                Log.e("ImageDownloader", "Something went wrong while" +
                        " retrieving bitmap from " + url + e.toString());
            }

            return null;
        }
    }
}
```

Comme il n'y a actuellement aucun champ de commentaire pour les exemples (ou que je ne l'ai pas trouvé ou que je n'ai pas l'autorisation de le faire), voici quelques commentaires à ce sujet:

C'est un bon exemple de ce que l'on peut faire avec AsyncTask.

Cependant, l'exemple a actuellement des problèmes avec

- fuites de mémoire possibles
- crash de l'application en cas de rotation de l'écran peu avant la fin de la tâche asynchrone.

Pour plus de détails voir:

- Passer l'activité en tant que WeakReference pour éviter les fuites de mémoire
- http://stackoverflow.com/documentation/android/117/asynctask/5377/possible-problems-with-inner-async-tasks
- Évitez les fuites d'activités avec AsyncTask

Passer l'activité en tant que WeakReference pour éviter les fuites de mémoire

Il est courant qu'un AsyncTask exige une référence à l'activité qui l'a appelé.

Si AsyncTask est une classe interne de l'activité, vous pouvez la référencer directement, ainsi que toutes les variables / méthodes membres.

Si, cependant, AsyncTask n'est pas une classe interne de l'activité, vous devrez transmettre une référence d'activité à la tâche asynchrone. Lorsque vous cela effectuez, un problème potentiel qui peut se produire est que AsyncTask conservera la référence de l'activité jusqu'à ce que AsyncTask ait terminé son travail dans son thread d'arrière-plan. Si l'activité est terminée ou supprimée avant que le thread d'arrière-plan d'AsyncTask ne soit terminé, AsyncTask aura toujours sa référence à l'activité et ne pourra donc pas être collectée.

Cela entraînera une fuite de mémoire.

Pour éviter cela, utilisez une référence WeakRefer dans AsyncTask au lieu d'avoir une référence directe à l'activité.

Voici un exemple d'AsyncTask utilisant une référence WeakRefer:

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

Appel de la tâche asynchrone à partir d'une activité:

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

Appel de la tâche Async depuis un fragment:

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

Ordre d'exécution

Lors de la première introduction, `AsyncTasks` ont été exécutés en série sur un seul thread d'arrière-plan. À partir de `DONUT` , cela a été changé en un pool de threads permettant à plusieurs tâches de fonctionner en parallèle. À partir de `HONEYCOMB` , les tâches sont exécutées sur un seul thread pour éviter les erreurs d'application courantes provoquées par une exécution en parallèle.

Si vous voulez vraiment une exécution parallèle, vous pouvez appeler `executeOnExecutor(java.util.concurrent.Executor, Object[])` avec `THREAD_POOL_EXECUTOR` .

> SERIAL_EXECUTOR -> Un exécuteur exécutant les tâches une par une dans un ordre série.
>
> THREAD_POOL_EXECUTOR -> Un exécuteur qui peut être utilisé pour exécuter des tâches en parallèle.

échantillon :

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

AsyncTask: exécution en série et exécution parallèle de la tâche

AsyncTask est une classe abstraite et n'hérite pas de la classe `Thread` . Il possède une méthode **abstraite** `doInBackground(Params... params)` , qui est remplacée pour effectuer la tâche. Cette méthode est appelée depuis `AsyncTask.call()` .

Les exécuteurs font partie du paquet `java.util.concurrent` .

De plus, AsyncTask contient 2 `Executor` s

THREAD_POOL_EXECUTOR

Il utilise des threads de travail pour exécuter les tâches en parallèle.

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

SERIAL_EXECUTOR

Il exécute la tâche en série, c'est-à-dire un par un.

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

Les deux `Executor` sont **statiques** , il existe donc un seul objet `THREAD_POOL_EXECUTOR` et un `SerialExecutor` objet `SerialExecutor` , mais vous pouvez créer plusieurs objets `AsyncTask` .

Par conséquent, si vous essayez d'effectuer plusieurs tâches en arrière-plan avec l'exécuteur par défaut (`SerialExecutor`), ces tâches seront mises en file d'attente et exécutées en série.

Si vous essayez d'effectuer plusieurs tâches en arrière-plan avec `THREAD_POOL_EXECUTOR` , elles seront exécutées en parallèle.

Exemple:

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

Effectuez un clic sur le bouton plusieurs fois pour démarrer une tâche et voir le résultat.

Tâche exécutée dans le pool de threads (1)

Chaque tâche dure 1000 ms.

À t = 36s, les tâches 2, 3 et 4 sont mises en file d'attente et ont commencé à s'exécuter également parce qu'elles s'exécutent en parallèle.

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

Commentaire `Task Executed in thread pool` (1) et décommenter `Task executed Serially` (2).

Effectuez un clic sur le bouton plusieurs fois pour démarrer une tâche et voir le résultat.

Il exécute la tâche en série, donc chaque tâche est démarrée après l'exécution de la tâche en cours. Par conséquent, lorsque l'exécution de la tâche 1 est terminée, seule la tâche 2 commence à s'exécuter en arrière-plan. Vice versa.

```
08-02 19:42:57.505: D/AsyncTaskExample(10299): Task = 1 Task Queued
08-02 19:42:57.505: D/AsyncTaskExample(10299): Task = 1 Task Running in Background
08-02 19:42:57.675: D/AsyncTaskExample(10299): Task = 2 Task Queued
08-02 19:42:57.835: D/AsyncTaskExample(10299): Task = 3 Task Queued
08-02 19:42:58.005: D/AsyncTaskExample(10299): Task = 4 Task Queued
08-02 19:42:58.155: D/AsyncTaskExample(10299): Task = 5 Task Queued
08-02 19:42:58.505: D/AsyncTaskExample(10299): Task = 1 Task Execution Completed
08-02 19:42:58.505: D/AsyncTaskExample(10299): Task = 2 Task Running in Background
08-02 19:42:58.755: D/AsyncTaskExample(10299): Task = 6 Task Queued
08-02 19:42:59.295: D/AsyncTaskExample(10299): Task = 7 Task Queued
08-02 19:42:59.505: D/AsyncTaskExample(10299): Task = 2 Task Execution Completed
08-02 19:42:59.505: D/AsyncTaskExample(10299): Task = 3 Task Running in Background
08-02 19:43:00.035: D/AsyncTaskExample(10299): Task = 8 Task Queued
08-02 19:43:00.505: D/AsyncTaskExample(10299): Task = 3 Task Execution Completed
08-02 19:43:**00.505**: D/AsyncTaskExample(10299): Task = 4 Task Running in Background
08-02 19:43:**01.505**: D/AsyncTaskExample(10299): Task = 4 Task Execution Completed
08-02 19:43:**01.515**: D/AsyncTaskExample(10299): Task = 5 Task Running in Background
08-02 19:43:**02.515**: D/AsyncTaskExample(10299): Task = 5 Task Execution Completed
08-02 19:43:**02.515**: D/AsyncTaskExample(10299): Task = 6 Task Running in Background
08-02 19:43:**03.515**: D/AsyncTaskExample(10299): Task = 7 Task Running in Background
08-02 19:43:**03.515**: D/AsyncTaskExample(10299): Task = 6 Task Execution Completed
08-02 19:43:04.515: D/AsyncTaskExample(10299): Task = 8 Task Running in Background
08-02 19:43:**04.515**: D/AsyncTaskExample(10299): Task = 7 Task Execution Completed
```

Lire AsyncTask en ligne: https://riptutorial.com/fr/android/topic/117/asynctask

Chapitre 28: AudioManager

Examples

Demande de focus audio transitoire

```
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);
        }
    }
}
```

Demande de focus audio

```
audioManager = (AudioManager) getSystemService(Context.AUDIO_SERVICE);

audioManager.requestAudioFocus(audioListener, AudioManager.STREAM_MUSIC,
AudioManager.AUDIOFOCUS_GAIN);

changedListener = new AudioManager.OnAudioFocusChangeListener() {
    @Override
    public void onAudioFocusChange(int focusChange) {
        if (focusChange == AudioManager.AUDIOFOCUS_GAIN) {
            // You now have the audio focus and may play sound.
        }
        else if (focusChange == AudioManager.AUDIOFOCUS_REQUEST_FAILED) {
            // Handle the failure.
        }
    }
}
```

Lire AudioManager en ligne: https://riptutorial.com/fr/android/topic/6798/audiomanager

Chapitre 29: Authentificateur Android

Examples

Service d'authentification de compte de base

Le système d'authentification de compte Android peut être utilisé pour authentifier le client avec un serveur distant. Trois informations sont requises:

- Un service, déclenché par `android.accounts.AccountAuthenticator` . Sa méthode `onBind` devrait renvoyer une sous-classe de `AbstractAccountAuthenticator` .
- Une activité pour demander à l'utilisateur des informations d'identification (activité de connexion)
- Un fichier de ressources xml pour décrire le compte

1. Le service:

Placez les autorisations suivantes dans votre fichier AndroidManifest.xml:

```
<uses-permission android:name="android.permission.GET_ACCOUNTS" />
<uses-permission android:name="android.permission.MANAGE_ACCOUNTS" />
<uses-permission android:name="android.permission.AUTHENTICATE_ACCOUNTS" />
<uses-permission android:name="android.permission.USE_CREDENTIALS" />
```

Déclarez le service dans le fichier manifeste:

```
<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>
```

Notez que `android.accounts.AccountAuthenticator` est inclus dans la balise `intent-filter` . La ressource xml (nommée `authenticator` ici) est spécifiée dans la balise `meta-data` .

La classe de service:

```
public class MyAuthenticationService extends Service {

    private static final Object lock = new Object();
    private MyAuthenticator mAuthenticator;

    public MyAuthenticationService() {
        super();
    }

    @Override
```

```
    public void onCreate() {
        super.onCreate();

        synchronized (lock) {
            if (mAuthenticator == null) {
                mAuthenticator = new MyAuthenticator(this);
            }
        }
    }

    @Override
    public IBinder onBind(Intent intent) {
        return mAuthenticator.getIBinder();
    }

}
```

2. La ressource xml:

```
<account-authenticator xmlns:android="http://schemas.android.com/apk/res/android"
    android:accountType="com.example.account"
    android:icon="@drawable/appicon"
    android:smallIcon="@drawable/appicon"
    android:label="@string/app_name" />
```

N'attribuez pas directement une chaîne à `android:label` ou attribuez des tirables manquants. Il va planter sans avertissement.

3. Étendez la classe AbstractAccountAuthenticator:

```
public class MyAuthenticator extends AbstractAccountAuthenticator {

    private Context mContext;

    public MyAuthenticator(Context context) {
        super(context);
        mContext = context;
    }

    @Override
    public Bundle addAccount(AccountAuthenticatorResponse response,
                             String accountType,
                             String authTokenType,
                             String[] requiredFeatures,
                             Bundle options) throws NetworkErrorException {

        Intent intent = new Intent(mContext, LoginActivity.class);
        intent.putExtra(AccountManager.KEY_ACCOUNT_AUTHENTICATOR_RESPONSE, response);

        Bundle bundle = new Bundle();
        bundle.putParcelable(AccountManager.KEY_INTENT, intent);

        return bundle;
    }

    @Override
    public Bundle confirmCredentials(AccountAuthenticatorResponse response, Account account,
Bundle options) throws NetworkErrorException {
```

```
        return null;
    }

    @Override
    public Bundle editProperties(AccountAuthenticatorResponse response, String accountType) {
        return null;
    }

    @Override
    public Bundle getAuthToken(AccountAuthenticatorResponse response, Account account, String
authTokenType, Bundle options) throws NetworkErrorException {
        return null;
    }

    @Override
    public String getAuthTokenLabel(String authTokenType) {
        return null;
    }

    @Override
    public Bundle hasFeatures(AccountAuthenticatorResponse response, Account account, String[]
features) throws NetworkErrorException {
        return null;
    }

    @Override
    public Bundle updateCredentials(AccountAuthenticatorResponse response, Account account,
String authTokenType, Bundle options) throws NetworkErrorException {
        return null;
    }
}
```

La méthode `addAccount()` de la classe `AbstractAccountAuthenticator` est importante car cette méthode est appelée lors de l'ajout d'un compte à partir de l'écran "Ajouter un compte" dans les paramètres sous. `AccountManager.KEY_ACCOUNT_AUTHENTICATOR_RESPONSE` est important, car il inclura l'objet AccountAuthenticatorResponse nécessaire pour renvoyer les clés du compte lors de la vérification de l'utilisateur.

Lire Authentificateur Android en ligne: https://riptutorial.com/fr/android/topic/6759/authentificateur-android

Chapitre 30: AutoCompleteTextView

Remarques

Si vous souhaitez proposer des suggestions à l'utilisateur lorsqu'il saisit un champ de texte modifiable, vous pouvez utiliser un `AutoCompleteTextView` . Il fournit des suggestions automatiquement lorsque l'utilisateur tape. La liste des suggestions est affichée dans un menu déroulant à partir duquel l'utilisateur peut en sélectionner un pour remplacer le contenu de la zone d'édition.

Examples

AutoCompleteTextView simple et codé en dur

Conception (XML de mise en page):

```
<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" />
```

Recherchez la vue dans le code après `setContentView()` (ou son équivalent de fragment ou de vue personnalisée):

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

Fournissez des données codées en dur via un adaptateur:

```
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);
```

Astuce: Bien que la méthode préférée soit de fournir des données via un `Loader` quelconque au lieu d'une liste codée comme celle-ci.

AutoComplete avec CustomAdapter, ClickListener et Filter

Mise en page principale: activity_main.xml

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

```
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <AutoCompleteTextView
        android:id="@+id/auto_name"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:completionThreshold="2"
        android:hint="@string/hint_enter_name" />
</LinearLayout>
```

Disposition des lignes row.xml

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

    <TextView
        android:id="@+id/lbl_name"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:paddingBottom="16dp"
        android:paddingLeft="8dp"
        android:paddingRight="8dp"
        android:paddingTop="16dp"
        android:text="Medium Text"
        android:textAppearance="?android:attr/textAppearanceMedium" />
</RelativeLayout>
```

strings.xml

```
<resources>
    <string name="hint_enter_name">Enter Name</string>
</resources>
```

MainActivity.java

```
public class MainActivity extends AppCompatActivity {
    AutoCompleteTextView txtSearch;
    List<People> mList;
    PeopleAdapter adapter;
    private People selectedPerson;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mList = retrievePeople();
        txtSearch = (AutoCompleteTextView) findViewById(R.id.auto_name);
        adapter = new PeopleAdapter(this, R.layout.activity_main, R.id.lbl_name, mList);
        txtSearch.setAdapter(adapter);
        txtSearch.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
```

```
            public void onItemClick(AdapterView<?> adapterView, View view, int pos, long id) {
                //this is the way to find selected object/item
                selectedPerson = (People) adapterView.getItemAtPosition(pos);
            }
        });
    }

    private List<People> retrievePeople() {
        List<People> list = new ArrayList<People>();
        list.add(new People("James", "Bond", 1));
        list.add(new People("Jason", "Bourne", 2));
        list.add(new People("Ethan", "Hunt", 3));
        list.add(new People("Sherlock", "Holmes", 4));
        list.add(new People("David", "Beckham", 5));
        list.add(new People("Bryan", "Adams", 6));
        list.add(new People("Arjen", "Robben", 7));
        list.add(new People("Van", "Persie", 8));
        list.add(new People("Zinedine", "Zidane", 9));
        list.add(new People("Luis", "Figo", 10));
        list.add(new People("John", "Watson", 11));
        return list;
    }
}
```

Classe de modèle: People.java

```
public class People {

    private String name, lastName;
    private int id;

    public People(String name, String lastName, int id) {
        this.name = name;
        this.lastName = lastName;
        this.id = id;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getlastName() {
        return lastName;
    }

    public void setlastName(String lastName) {
        this.lastName = lastName;
```

```
    }
}
```

Classe d'adaptateur: 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();
                }
            }
        }
    };
}
```

Lire AutoCompleteTextView en ligne:
https://riptutorial.com/fr/android/topic/5300/autocompletetextview

Chapitre 31: Autorisations d'exécution dans API-23 +

Introduction

Android Marshmallow a introduit le modèle d' autorisation d'exécution . Les autorisations sont classées en deux catégories, à savoir les autorisations normales et dangereuses . où les autorisations dangereuses sont désormais accordées par l'utilisateur au moment de l'exécution.

Remarques

Depuis sdk 23, Android requiert des autorisations d'exécution pour les autorisations sur les appareils fonctionnant sous Android 6.0 et les versions ultérieures, au sein de ce que l'on appelle les groupes d'autorisations dangereuses. Les groupes de permission dangereux sont ceux qui sont considérés comme compromettant la confidentialité et / ou la sécurité de l'utilisateur.

Voici une liste des groupes de permissions dangereux:

Groupes de permissions dangereux

Groupe de permission
CALENDRIER
CAMÉRA
CONTACTS
EMPLACEMENT
MICROPHONE
TÉLÉPHONE
CAPTEURS
SMS
ESPACE DE RANGEMENT

Toutes les autorisations de ces groupes requièrent la gestion des autorisations d'exécution pour les périphériques sous Android 6.0 et les versions ultérieures avec un SDK cible de 23 ou supérieur.

Autorisations normales

Voici une liste des autorisations normales. Ceux-ci ne sont pas considérés comme dangereux pour la confidentialité ou la sécurité de l'utilisateur et ne nécessitent donc pas d'autorisations d'exécution pour sdk 23 et versions ultérieures.

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
L'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
RÉGLER L'ALARME
SET_TIME_ZONE
DÉFINIR LE PAPIER PEINT
SET_WALLPAPER_HINTS
TRANSMIT_IR
UNINSTALL_SHORTCUT
USE_FINGERPRINT
VIBRER
VERROUILLAGE DE RÉVEIL
WRITE_SYNC_SETTINGS

Examples

Android 6.0 autorisations multiples

Cet exemple montre comment vérifier les autorisations à l'exécution dans Android 6 et versions ultérieures.

```
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;
        }
    }
}
```

Application des autorisations dans les diffusions, URI

Vous pouvez effectuer une vérification des autorisations lors de l'envoi d'une intention à un récepteur de diffusion enregistré. Les autorisations que vous envoyez sont vérifiées avec celles enregistrées sous la balise. Ils restreignent qui peut envoyer des émissions au récepteur associé.

Pour envoyer une demande de diffusion avec des autorisations, spécifiez l'autorisation en tant que chaîne dans l' `Context.sendBroadcast(Intent intent, String permission)` , mais gardez à l'esprit que l'application du destinataire **DOIT** avoir cette autorisation pour recevoir votre diffusion. Le récepteur doit être installé avant l'expéditeur.

La signature de la méthode est la suivante:

```
 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");
```

et vous pouvez spécifier dans votre manifeste que l'expéditeur de la diffusion doit inclure l'autorisation demandée envoyée via sendBroadcast:

```
<!--  Your special permission -->
<permission android:name="org.quadcore.mypermission"
   android:label="my_permission"
   android:protectionLevel="dangerous"></permission>
```

Déclarez également l'autorisation dans le manifeste de l'application censée recevoir cette diffusion:

```
<!--  I use the permission ! -->
<uses-permission android:name="org.quadcore.mypermission"/>
<!-- along with the receiver -->
<receiver android:name="Bcastreceiver" android:exported="true" />
```

Remarque: à la fois un récepteur et un diffuseur peuvent demander une autorisation et, lorsque cela se produit, les deux contrôles d'autorisation doivent être transmis pour que l'intention soit transmise à la cible associée. L'application qui définit l'autorisation doit être installée en premier.

Retrouvez la documentation complète ici sur les autorisations.

Autorisations d'exécution multiples à partir des mêmes groupes d'autorisations

Dans le manifeste, nous avons quatre autorisations d'exécution dangereuses provenant de deux groupes.

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

Dans l'activité où les autorisations sont requises. Notez qu'il est important de vérifier les autorisations dans toute activité nécessitant des autorisations, car les autorisations peuvent être révoquées lorsque l'application est en arrière-plan et l'application se bloque ensuite.

```
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();
    }
```

Il suffit de demander une autorisation pour l'un de ces groupes et toutes les autres autorisations de ce groupe sont accordées, sauf si l'utilisateur les révoque.

```
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;
}
```

Cela concerne le résultat de l'utilisateur autorisant ou non les autorisations. Dans cet exemple, si les autorisations ne sont pas autorisées, l'application est supprimée.

```
@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);
    }
}
```

Plus d'informations https://inthecheesefactory.com/blog/things-you-need-to-know-about-android-m-permission-developer-edition/en

Utiliser PermissionUtil

PermissionUtil est un moyen simple et pratique de demander des autorisations en contexte. Vous pouvez facilement fournir ce qui doit arriver dans le cas de toutes les autorisations demandées (`onAllGranted()`), toute demande a été refusée (`onAnyDenied()`) ou si un rationnel est nécessaire (`onRational()`).

N'importe où dans votre AppCompatActivity ou Fragment que vous souhaitez demander à l'utilisateur

```
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);
```

Et ajoutez ceci à `onRequestPermissionsResult`

```
if(mRequestObject!=null){
    mRequestObject.onRequestPermissionsResult(requestCode, permissions, grantResults);
}
```

Ajoutez également l'autorisation demandée à votre AndroidManifest.xml

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

Inclure tout le code lié aux permissions dans une classe de base abstraite et étendre l'activité de cette classe de base pour obtenir un code plus propre / réutilisable

```
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);
    }
}
```

Exemple d'utilisation dans l'activité

L'activité doit étendre la classe de base abstraite définie ci-dessus comme suit:

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

Lire Autorisations d'exécution dans API-23 + en ligne:
https://riptutorial.com/fr/android/topic/1525/autorisations-d-execution-dans-api-23-plus

Chapitre 32: Autosizing TextViews

Introduction

Un TextView qui redimensionne automatiquement le texte pour s'adapter parfaitement à ses limites.

Android O vous permet de demander à un TextView de laisser la taille du texte se développer ou se contracter automatiquement pour remplir sa mise en page en fonction des caractéristiques et des limites de TextView.

Vous pouvez configurer le redimensionnement automatique de TextView en code ou en XML.

Il existe deux manières de définir le redimensionnement automatique de TextView: **Granularité** et **tailles prédéfinies**

Examples

Granularité

En Java:

Appelez la méthode `setAutoSizeTextTypeUniformWithConfiguration()` :

```
setAutoSizeTextTypeUniformWithConfiguration(int autoSizeMinTextSize, int autoSizeMaxTextSize,
int autoSizeStepGranularity, int unit)
```

En XML:

Utilisez les `autoSizeMinTextSize` , `autoSizeMaxTextSize` et `autoSizeStepGranularity` pour définir les dimensions de dimensionnement automatique dans le fichier XML de présentation:

```
<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" />
```

Consultez le document AutosizingTextViews-Demo sur GitHub pour plus de détails.

Tailles prédéfinies

En Java:

Appelez la méthode `setAutoSizeTextTypeUniformWithPresetSizes()` :

```
setAutoSizeTextTypeUniformWithPresetSizes(int[] presetSizes, int unit)
```

En XML:

Utilisez l'attribut `autoSizePresetSizes` dans le fichier XML de présentation:

```
<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" />
```

Pour accéder au tableau en tant que ressource, définissez le tableau dans le fichier *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>
```

Consultez le document AutosizingTextViews-Demo sur GitHub pour plus de détails.

Lire Autosizing TextViews en ligne: https://riptutorial.com/fr/android/topic/9652/autosizing-textviews

Chapitre 33: Avertissements de peluches

Remarques

L' **outil Lint** vérifie les fichiers sources de votre projet Android à la recherche d'éventuels bogues et d'optimisations en termes d'exactitude, de sécurité, de performances, de facilité d'utilisation, d'accessibilité et d'internationalisation. Vous pouvez exécuter Lint depuis la ligne de commande ou depuis Android Studio.

Documentation officielle:

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

Examples

Utilisation des outils: ignore dans les fichiers xml

Les `tools:ignore` attribut `tools:ignore` peuvent être utilisés dans les fichiers xml pour `tools:ignore` avertissements de peluches.

MAIS rejeter les avertissements de peluches avec cette technique est la plupart du temps la mauvaise façon de procéder.

Un avertissement de peluches doit être compris et corrigé… il peut être ignoré si et seulement si vous avez une compréhension complète de sa signification et une forte raison de l'ignorer.

Voici un cas d'utilisation où il est légitime d'ignorer un avertissement de peluches:

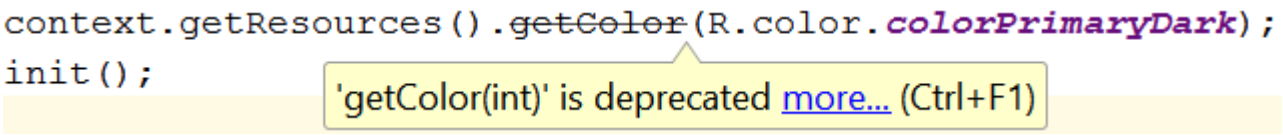
- Vous développez une application système (signée avec la clé du fabricant du périphérique)
- Votre application doit modifier la date du périphérique (ou toute autre action protégée)

Ensuite, vous pouvez le faire dans votre manifeste: (c'est-à-dire demander la permission protégée et ignorer l'avertissement de peluches parce que vous savez que dans votre cas la permission sera accordée)

```
<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"/>
```

Importation de ressources sans erreur "obsolète"

En utilisant l'API Android 23 ou supérieure, très souvent, une telle situation peut être observée:



Cette situation est due au changement structurel de l'API Android concernant l'obtention des ressources. Maintenant la fonction:

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

Devrait être utilisé. Mais la bibliothèque android.support.v4 a une autre solution.

Ajoutez la dépendance suivante au fichier build.gradle:

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

Ensuite, toutes les méthodes de la bibliothèque de support sont disponibles:

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

De plus, d'autres méthodes de la bibliothèque de support peuvent être utilisées:

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

Configurer LintOptions avec gradle

Vous pouvez configurer lint en ajoutant une section `lintOptions` dans le fichier `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
    }
}
```

Vous pouvez exécuter lint pour une variante spécifique (voir ci-dessous), par exemple `./gradlew lintRelease` , ou pour toutes les variantes (`./gradlew lint`).

Vérifiez ici la référence DSL pour toutes les options disponibles .

Comment configurer le fichier lint.xml

Vous pouvez spécifier vos préférences de vérification de `lint.xml` dans le fichier `lint.xml` . Si vous créez ce fichier manuellement, placez-le dans le répertoire racine de votre projet Android. Si vous configurez les préférences Lint dans Android Studio, le fichier lint.xml est automatiquement créé et ajouté à votre projet Android pour vous.

Exemple:

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

En définissant la valeur de l'attribut de gravité dans la balise, vous pouvez désactiver la vérification de la charpie pour détecter un problème ou modifier le niveau de gravité d'un problème.

L'exemple suivant montre le contenu d'un fichier `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>
```

Configuration de la vérification des peluches dans les fichiers source Java et XML

Vous pouvez désactiver la vérification de la charpie de vos fichiers source Java et XML.

Configuration de la vérification des peluches en Java

Pour désactiver la vérification de Lint spécifiquement pour une **classe** ou une méthode **Java** dans votre projet Android, ajoutez l' **annotation** `@SuppressLint` à ce code Java.

Exemple:

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

Pour désactiver la vérification de tous les problèmes de Lint:

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

Configuration de la vérification de peluches en XML

Vous pouvez utiliser les `tools:ignore` attribut Ignorer pour désactiver la vérification de la charpie des sections spécifiques de vos **fichiers XML** .

Par exemple:

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

Pour supprimer la vérification de tous les problèmes de Lint dans l'élément XML, utilisez

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

Mark Supprimer les avertissements

Il est recommandé de marquer des avertissements dans votre code. Par exemple, certaines méthodes obsolètes sont nécessaires pour vos tests ou votre ancienne version de support. Mais la vérification de peluches marque ce code avec des avertissements. Pour éviter ce problème, vous devez utiliser l'annotation @SuppressWarnings.

Par exemple, ajoutez ignorer les avertissements aux méthodes obsolètes. Vous devez également mettre la description des avertissements dans l'annotation:

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

En utilisant cette annotation, vous pouvez ignorer tous les avertissements, y compris Lint, Android et autres. En utilisant les avertissements de suppression, aide à comprendre le code correctement!

Lire Avertissements de peluches en ligne:
https://riptutorial.com/fr/android/topic/129/avertissements-de-peluches

Chapitre 34: Barre de progression

Remarques

Documentation Officielle: ProgressBar

Examples

ProgressBar Indéterminé

Un ProgressBar indéterminé montre une animation cyclique sans indication de progrès.

ProgressBar de base indéterminé (rouet)

```
<ProgressBar
    android:id="@+id/progressBar"
    android:indeterminate="true"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>
```

ProgressBar horizontal indéterminé (barre plate)

```
<ProgressBar
    android:id="@+id/progressBar"
    android:indeterminate="true"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    style="@android:style/Widget.ProgressBar.Horizontal"/>
```

Autres styles ProgressBar intégrés

```
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"
```

Pour utiliser la barre de progression indéterminée dans une activité

```
ProgressBar progressBar = (ProgressBar) findViewById(R.id.progressBar);
progressBar.setVisibility(View.VISIBLE);
progressBar.setVisibility(View.GONE);
```

Déterminer ProgressBar

Un ProgressBar déterminé indique la progression actuelle vers une valeur maximale spécifique.

ProgressBar à détermination horizontale

```
<ProgressBar
    android:id="@+id/progressBar"
    android:indeterminate="false"
    android:layout_width="match_parent"
    android:layout_height="10dp"
    style="@android:style/Widget.ProgressBar.Horizontal"/>
```

ProgressBar à détermination verticale

```
<ProgressBar
    android:id="@+id/progressBar"
    android:indeterminate="false"
    android:layout_width="10dp"
    android:layout_height="match_parent"
    android:progressDrawable="@drawable/progress_vertical"
    style="@android:style/Widget.ProgressBar.Horizontal"/>
```

res / drawable / progress_vertical.xml

```
<?xml version="1.0" encoding="utf-8"?>
<layer-list xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@android:id/background">
        <shape>
            <corners android:radius="3dp"/>
            <solid android:color="@android:color/darker_gray"/>
        </shape>
    </item>
    <item android:id="@android:id/secondaryProgress">
        <clip android:clipOrientation="vertical" android:gravity="bottom">
            <shape>
                <corners android:radius="3dp"/>
                <solid android:color="@android:color/holo_blue_light"/>
            </shape>
        </clip>
    </item>
    <item android:id="@android:id/progress">
        <clip android:clipOrientation="vertical" android:gravity="bottom">
            <shape>
                <corners android:radius="3dp"/>
                <solid android:color="@android:color/holo_blue_dark"/>
            </shape>
        </clip>
    </item>
</layer-list>
```

Anneau déterminé 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>
```

Pour utiliser la barre de progression déterminée dans une activité.

```
ProgressBar progressBar = (ProgressBar) findViewById(R.id.progressBar);
progressBar.setSecondaryProgress(100);
progressBar.setProgress(10);
progressBar.setMax(100);
```

Barre de progression personnalisée

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();

    }
}
```

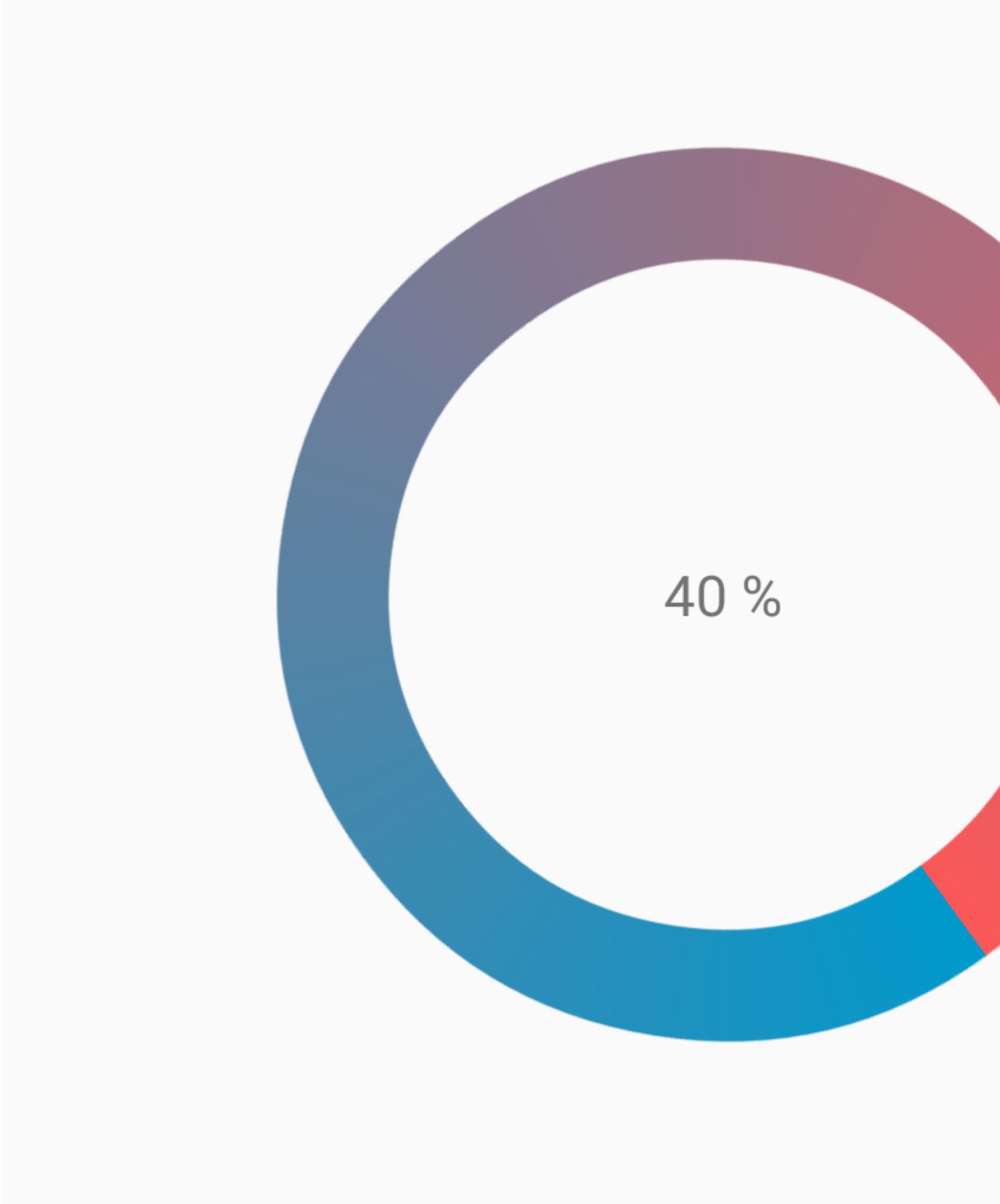
activity_custom_progressbar.xml :

```
<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>
```

Capture d'écran de référence:



Tinting ProgressBar

En utilisant un thème AppCompat, la couleur du `ProgressBar` sera le `colorAccent` vous avez défini.

5.0

Pour modifier la couleur `ProgressBar` sans modifier la couleur d'accentuation, vous pouvez utiliser l'attribut `android:theme` pour remplacer la couleur d'accentuation:

```
<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>
```

Pour colorer la `ProgressBar` vous pouvez utiliser dans le fichier xml les attributs `android:indeterminateTintMode` et `android:indeterminateTint`

```
<ProgressBar
    android:indeterminateTintMode="src_in"
    android:indeterminateTint="@color/my_color"
/>
```

Progressbar matériel linéaire

Selon la documentation matérielle :

> Un indicateur de progression linéaire devrait toujours remplir de 0% à 100% et ne jamais diminuer en valeur.
> Il doit être représenté par des barres sur le bord d'un en-tête ou d'une feuille qui apparaissent et disparaissent.

Pour utiliser une barre de progression linéaire, utilisez simplement dans votre fichier xml:

```
<ProgressBar
    android:id="@+id/my_progressBar"
    style="@style/Widget.AppCompat.ProgressBar.Horizontal"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>
```

Indéterminé

Pour créer ProgressBar **indéterminé,** définissez l'attribut `android:indeterminate` sur `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"/>
```

Déterminé

Pour créer **un** ProgressBar **déterminé,** définissez l'attribut `android:indeterminate` sur `false` et utilisez les attributs `android:max` et `android:progress` :

```
<ProgressBar
    android:id="@+id/my_progressBar"
    style="@style/Widget.AppCompat.ProgressBar.Horizontal"
    android:indeterminate="false"
    android:max="100"
    android:progress="10"/>
```

Utilisez simplement ce code pour mettre à jour la valeur:

```
ProgressBar progressBar = (ProgressBar) findViewById(R.id.my_progressBar);
progressBar.setProgress(20);
```

Tampon

Pour créer un effet **tampon** avec ProgressBar, définissez l'attribut `android:indeterminate` sur `false` et utilisez les attributs `android:max` , `android:progress` et `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"/>
```

La valeur du tampon est définie par `android:secondaryProgress` attribut `android:secondaryProgress` . Utilisez simplement ce code pour mettre à jour les valeurs:

```
ProgressBar progressBar = (ProgressBar) findViewById(R.id.my_progressBar);
progressBar.setProgress(20);
progressBar.setSecondaryProgress(50);
```

Indéterminé et Déterminé

Pour obtenir ce type de ProgressBar, utilisez simplement un ProgressBar indéterminé utilisant l'attribut `android:indeterminate` à true.

```
<ProgressBar
    android:id="@+id/progressBar"
    style="@style/Widget.AppCompat.ProgressBar.Horizontal"
```

```
    android:indeterminate="true"/>
```

Ensuite, lorsque vous devez passer d'une progression indéterminée à une progression déterminée, utilisez la méthode `setIndeterminate()` .

```
ProgressBar progressBar = (ProgressBar) findViewById(R.id.my_progressBar);
progressBar.setIndeterminate(false);
```

Création d'une boîte de dialogue de progression personnalisée

En créant une classe de dialogue de progression personnalisée, la boîte de dialogue peut être utilisée pour afficher une instance de l'interface utilisateur sans recréer la boîte de dialogue.

Commencez par créer une classe de dialogue de progression personnalisée.

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;
       }
   }
}
```

Maintenant créer la mise en page de progression personnalisée

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>
```

Ça y est. Maintenant, pour appeler le dialogue dans le 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();
```

Lire Barre de progression en ligne: https://riptutorial.com/fr/android/topic/3353/barre-de-progression

Chapitre 35: Bibliothèque Dagger 2: injection de dépendance dans les applications

Introduction

Dagger 2, comme expliqué sur GitHub , est une approche évolutive à la compilation de l'injection de dépendance. En prenant l'approche de Dagger 1.x, Dagger 2.x élimine toute réflexion et améliore la clarté du code en supprimant l' `ObjectGraph` / `Injector` traditionnel au profit des interfaces `@Component` spécifiées `@Component` utilisateur.

Remarques

1. Configuration de la bibliothèque dans l'application (pour les projets maven, gradle, java)
2. Avantages de l'utilisation de Dragger
3. Liens importants (pour la documentation et les démos)
4. Comment intégrer et utiliser des composants Dragger

Dagger 2 API:

Dagger 2 expose plusieurs annotations spéciales:

@Module pour les classes dont les méthodes fournissent des dépendances

@Provides pour les méthodes dans les classes @Module

@Inject pour demander une dépendance (un constructeur, un champ ou une méthode)

@Component est une interface de pont entre les modules et l'injection

Liens importants:

GitHub: https://github.com/google/dagger

UserGuide (Google): https://google.github.io/dagger/users-guide.html

Vidéos: https://google.github.io/dagger/resources.html

Tutoriel Vogella: http://www.vogella.com/tutorials/Dagger/article.html

Didacticiel Codepath: https://github.com/codepath/android_guides/wiki/Dependency-Injection-with-Dagger-2

Examples

Créer une annotation @Module Class et @Singleton pour 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());
    }
}
```

Chaque fournisseur (ou méthode) doit avoir l'annotation `@Provides` et la classe doit avoir l'annotation `@Module` . L'annotation `@Singleton` indique qu'il n'y aura qu'une seule instance de l'objet.

Demander des dépendances dans des objets dépendants

Maintenant que vous avez les fournisseurs de vos différents modèles, vous devez les demander. Tout comme `Vehicle` besoin de `Motor` , vous devez ajouter l'annotation `@Inject` dans le constructeur du `Vehicle` comme suit:

```
@Inject
public Vehicle(Motor motor){
    this.motor = motor;
}
```

Vous pouvez utiliser l'annotation `@Inject` pour demander des dépendances dans le constructeur, les champs ou les méthodes. Dans cet exemple, je garde l'injection dans le constructeur.

Connecter @Modules avec @Inject

La connexion entre le fournisseur de dépendances, `@Module` , et les classes les demandant via `@Inject` est effectuée à l'aide de `@Component` , qui est une interface:

```
import javax.inject.Singleton;
import dagger.Component;

@Singleton
@Component(modules = {VehicleModule.class})
public interface VehicleComponent {
    Vehicle provideVehicle();
}
```

Pour l'annotation `@Component` , vous devez spécifier les modules à utiliser. Dans cet exemple, `VehicleModule` est utilisé, ce qui est défini dans cet exemple . Si vous devez utiliser plus de modules, ajoutez-les simplement en utilisant une virgule comme séparateur.

Utilisation de l'interface @Component pour obtenir des objets

Maintenant que chaque connexion est prête, vous devez obtenir une instance de cette interface et appeler ses méthodes pour obtenir l'objet dont vous avez besoin:

```
VehicleComponent component = Dagger_VehicleComponent.builder().vehicleModule(new
VehicleModule()).build();
vehicle = component.provideVehicle();
Toast.makeText(this, String.valueOf(vehicle.getSpeed()), Toast.LENGTH_SHORT).show();
```

Lorsque vous essayez de créer un nouvel objet de l'interface avec l'annotation `@Component` , vous devez le faire en utilisant le préfixe `Dagger_<NameOfTheComponentInterface>` , dans ce cas `Dagger_VehicleComponent` , puis utilisez la méthode de générateur pour appeler tous les modules.

Lire Bibliothèque Dagger 2: injection de dépendance dans les applications en ligne: https://riptutorial.com/fr/android/topic/9079/bibliotheque-dagger-2--injection-de-dependance-dans-les-applications

Chapitre 36: Bibliothèque de liaison de données

Remarques

Installer

Avant d'utiliser la liaison de données, vous devez activer le plug-in dans votre `build.gradle` .

```
android {
    ....
    dataBinding {
        enabled = true
    }
}
```

Remarque: la liaison de données a été ajoutée au plug-in Android Gradle dans la version 1.5.0

Noms de classe de liaison

Le plug-in de liaison de données génère un nom de classe de liaison en convertissant le nom de fichier de votre mise en page en cas Pascal et en ajoutant "Liaison" à la fin. Ainsi, `item_detail_activity.xml` générera une classe nommée `ItemDetailActivityBinding` .

Ressources

- Documentation officielle

Examples

Liaison de texte de base

Gradle (Module: app) Configuration

```
android {
    ....
    dataBinding {
        enabled = true
    }
}
```

Modèle de données

```
public class Item {
    public String name;
    public String description;

    public Item(String name, String description) {
```

```
        this.name = name;
        this.description = description;
    }
}
```

XML de mise en page

La première étape consiste à encapsuler votre mise en page dans une `<layout>` , à ajouter un élément `<data>` et à ajouter un élément `<variable>` à votre modèle de données.

Vous pouvez ensuite lier des attributs XML à des champs du modèle de données à l'aide de `@{model.fieldname}` , où `model` est le nom de la variable et `fieldname` le champ `@{model.fieldname}` vous souhaitez accéder.

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>
```

Pour chaque fichier de disposition XML correctement configuré avec des liaisons, le plugin Android Gradle génère une classe correspondante: bindings. Comme nous avons une mise en page nommée *item_detail_activity* , la classe de liaison générée correspondante s'appelle `ItemDetailActivityBinding` .

Cette liaison peut alors être utilisée dans une activité comme celle-ci:

```
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);
```

```
    }
}
```

Liaison avec une méthode d'accesseur

Si votre modèle a des méthodes privées, la bibliothèque de liaison de données vous permet toujours d'y accéder dans votre vue sans utiliser le nom complet de la méthode.

Modèle de données

```
public class Item {
    private String name;

    public String getName() {
        return name;
    }
}
```

XML de mise en page

```
<?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>
```

Cours de référencement

Modèle de données

```
public class Item {
    private String name;

    public String getName() {
        return name;
    }
}
```

XML de mise en page

Vous devez importer les classes référencées, comme vous le feriez en Java.

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android">
   <data>
       <import type="android.view.View"/>
       <variable name="item" type="com.example.Item"/>
   </data>

   <LinearLayout
       android:orientation="vertical"
       android:layout_width="match_parent"
       android:layout_height="match_parent">

       <!-- We reference the View class to set the visibility of this TextView -->
       <TextView
           android:layout_width="wrap_content"
           android:layout_height="wrap_content"
           android:text="@{item.name}"
           android:visibility="@{item.name == null ? View.VISIBLE : View.GONE"/>

   </LinearLayout>
</layout>
```

Remarque: Le package `java.lang.*` Est importé automatiquement par le système. (La même chose est faite par `JVM` pour `Java`)

Liaison de données en fragment

Modèle de données

```
public class Item {
    private String name;

    public String getName() {
        return name;
    }

    public void setName(String name){
        this.name = name;
    }

}
```

XML de mise en page

```
<?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();
}
```

Liaison de données bidirectionnelle intégrée

La liaison de données bidirectionnelle prend en charge les attributs suivants:

Élément	Propriétés
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

Élément	Propriétés
ToggleButton	android:checked
Switch	android:checked

Usage

```
<layout ...>
    <data>
        <variable type="com.example.myapp.User" name="user"/>
    </data>
    <RelativeLayout ...>
        <EditText android:text="@={user.firstName}" .../>
    </RelativeLayout>
</layout>
```

Notez que l'expression Binding `@={}` **a un = supplémentaire** , nécessaire pour la **liaison bidirectionnelle** . Il n'est pas possible d'utiliser des méthodes dans des expressions de liaison bidirectionnelles.

Liaison de données dans l'adaptateur RecyclerView

Il est également possible d'utiliser la liaison de données dans votre adaptateur `RecyclerView` .

Modèle de données

```
public class Item {
    private String name;

    public String getName() {
        return name;
    }
}
```

Mise en page XML

```
<TextView
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:text="@{item.name}"/>
```

Classe d'adaptateur

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

Cliquez sur auditeur avec liaison

Créer une interface pour clickHandler

```
public interface ClickHandler {
    public void onButtonClick(View v);
}
```

XML de mise en page

```
<?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>
```

Manipuler l'événement dans votre activité

```
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();
    }
}
```

Evénement personnalisé à l'aide de l'expression lambda

Définir l'interface

```
public interface ClickHandler {
    public void onButtonClick(User user);
}
```

Créer une classe de modèle

```
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;
```

```
    }
}
```

XML de mise en page

```
<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>
```

Code d'activité:

```
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();
    }
}
```

Pour certains écouteurs de vue qui ne sont pas disponibles dans le code xml mais peuvent être définis dans du code java, ils peuvent être associés à une liaison personnalisée.

Classe personnalisée

```
public class BindingUtil {
    @BindingAdapter({"bind:autoAdapter"})
    public static void setAdapter(AutoCompleteTextView view, ArrayAdapter<String>
```

```
pArrayAdapter) {
        view.setAdapter(pArrayAdapter);
    }
    @BindingAdapter({"bind:onKeyListener"})
    public static void setOnKeyListener(AutoCompleteTextView view , View.OnKeyListener
pOnKeyListener)
    {
        view.setOnKeyListener(pOnKeyListener);
    }
}
```

Classe de gestionnaire

```
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>
```

Valeur par défaut dans la liaison de données

> Le volet Aperçu affiche les valeurs par défaut des expressions de liaison de données si elles sont fournies.

Par exemple :

```
android:layout_height="@{@dimen/main_layout_height, default=wrap_content}"
```

Il faudra `wrap_content` lors de la conception et agira comme un `wrap_content` dans le volet de prévisualisation.

Un autre exemple est

```
android:text="@{user.name, default=`Preview Text`}"
```

Il affichera le `Preview Text` d'aperçu dans le volet d'aperçu, mais lorsque vous l'exécuterez dans le périphérique / émulateur, le texte réel qui lui est lié sera affiché.

DataBinding avec des variables personnalisées (int, booléen)

Parfois, nous devons effectuer des opérations de base telles que masquer / afficher la vue en fonction d'une valeur unique, pour cette variable unique, nous ne pouvons pas créer de modèle ou ce n'est pas une bonne pratique de créer un modèle pour cela. DataBinding prend en charge les types de données de base pour effectuer ces opérations.

```
<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>
```

et définir sa valeur de classe java.

```
binding.setSelected(true);
```

Liaison de données dans un dialogue

```
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();
}
```

Passer le widget comme référence dans BindingAdapter

XML de mise en page

```
<?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>
```

Méthode BindingAdapter

```
@BindingAdapter({"imageUrl","progressbar"})
public static void loadImage(ImageView view, String imageUrl, ProgressBar progressBar){
    Glide.with(view.getContext()).load(imageUrl)
                        .listener(new RequestListener<String, GlideDrawable>() {
                    @Override
                    public boolean onException(Exception e, String model,
Target<GlideDrawable> target, boolean isFirstResource) {
                        return false;
                    }

                    @Override
                    public boolean onResourceReady(GlideDrawable resource, String model,
Target<GlideDrawable> target, boolean isFromMemoryCache, boolean isFirstResource) {
                        progressBar.setVisibility(View.GONE);
                        return false;
                    }
                }).into(view);
}
```

Lire Bibliothèque de liaison de données en ligne:
https://riptutorial.com/fr/android/topic/111/bibliotheque-de-liaison-de-donnees

Chapitre 37: Bluetooth Low Energy

Introduction

Cette documentation se veut une amélioration par rapport à la documentation originale et se concentrera sur la dernière API Bluetooth LE introduite dans Android 5.0 (API 21). Les rôles centraux et périphériques seront couverts, ainsi que le démarrage des opérations de numérisation et de publicité.

Examples

Recherche de périphériques BLE

Les autorisations suivantes sont requises pour utiliser les API Bluetooth:

```
android.permission.BLUETOOTH
android.permission.BLUETOOTH_ADMIN
```

Si vous ciblez des appareils avec Android 6.0 (**API niveau 23**) ou supérieur et que vous souhaitez effectuer des opérations d'analyse / de publicité, vous aurez besoin d'une autorisation de localisation:

```
android.permission.ACCESS_FINE_LOCATION
```

ou

```
android.permission.ACCESS_COARSE_LOCATION
```

Remarque.- Les services de localisation avec Android 6.0 (API niveau 23) ou plus doivent également avoir les services de localisation activés.

Un objet BluetoothAdapter est requis pour démarrer les opérations d'analyse / de publicité:

```
BluetoothManager bluetoothManager = (BluetoothManager)
context.getSystemService(Context.BLUETOOTH_SERVICE);
bluetoothAdapter = bluetoothManager.getAdapter();
```

La `startScan (ScanCallback callback)` de la classe BluetoothLeScanner est la méthode la plus simple pour démarrer une opération d'analyse. Un objet `ScanCallback` est requis pour recevoir les résultats:

```
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());
```

```
       }
    });
```

Connexion à un serveur GATT

Une fois que vous avez découvert un objet BluetoothDevice, vous pouvez vous y connecter en utilisant sa méthode `connectGatt()` qui prend en paramètre un objet Context, un booléen indiquant s'il faut se connecter automatiquement à l'appareil BLE et une référence BluetoothGattCallback où les événements de connexion et les opérations client les résultats seront livrés:

```
 if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
        device.connectGatt(context, false, bluetoothGattCallback,
BluetoothDevice.TRANSPORT_AUTO);
    } else {
        device.connectGatt(context, false, bluetoothGattCallback);
    }
```

Remplacez `onConnectionStateChange` dans BluetoothGattCallback pour recevoir une connexion des événements de déconnexion:

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

Écrire et lire à partir de caractéristiques

Une fois connecté à un serveur Gatt, vous allez interagir avec lui en écrivant et en lisant les caractéristiques du serveur. Pour ce faire, vous devez d’abord découvrir quels services sont disponibles sur ce serveur et quelles caractéristiques sont disponibles dans chaque service:

```
 @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
                    }
                }
              }
            }
```

Une opération d'écriture de base se déroule comme suit:

```
characteristic.setValue(newValue);
characteristic.setWriteType(BluetoothGattCharacteristic.WRITE_TYPE_DEFAULT);
gatt.writeCharacteristic(characteristic);
```

Lorsque le processus d'écriture est terminé, la méthode `onCharacteristicWrite` de votre `BluetoothGattCallback` sera appelée:

```
    @Override
    public void onCharacteristicWrite(BluetoothGatt gatt, BluetoothGattCharacteristic
characteristic, int status) {
        super.onCharacteristicWrite(gatt, characteristic, status);
        Log.d(TAG, "Characteristic " + characteristic.getUuid() + " written);
    }
```

Une opération d'écriture de base se déroule comme suit:

```
gatt.readCharacteristic(characteristic);
```

Lorsque le processus d'écriture est terminé, la méthode `onCharacteristicRead` de votre `BluetoothGattCallback` sera appelée:

```
@Override
public void onCharacteristicRead(BluetoothGatt gatt, BluetoothGattCharacteristic
characteristic, int status) {
  super.onCharacteristicRead(gatt, characteristic, status);
  byte[] value = characteristic.getValue();
  }
```

Abonnement aux notifications du serveur Gatt

Vous pouvez demander à être averti par le serveur Gatt lorsque la valeur d'une caractéristique a été modifiée:

```
gatt.setCharacteristicNotification(characteristic, true);
BluetoothGattDescriptor descriptor = characteristic.getDescriptor(
    UUID.fromString("00002902-0000-1000-8000-00805f9b34fb");
descriptor.setValue(BluetoothGattDescriptor.ENABLE_NOTIFICATION_VALUE);
mBluetoothGatt.writeDescriptor(descriptor);
```

Toutes les notifications du serveur seront reçues dans la méthode `onCharacteristicChanged` de votre BluetoothGattCallback:

```
    @Override
    public void onCharacteristicChanged(BluetoothGatt gatt, BluetoothGattCharacteristic
characteristic) {
        super.onCharacteristicChanged(gatt, characteristic);
        byte[] newValue = characteristic.getValue();
}
```

Publicité d'un appareil BLE

Vous pouvez utiliser la publicité Bluetooth LE pour diffuser des paquets de données sur tous les appareils à proximité sans avoir à établir au préalable une connexion. Gardez à l'esprit qu'il existe une limite stricte de 31 octets de données publicitaires. La publicité de votre appareil est également la première étape pour permettre aux autres utilisateurs de se connecter à vous.

Comme tous les appareils ne prennent pas en charge la publicité Bluetooth LE, la première étape consiste à vérifier que votre appareil dispose de toutes les conditions requises pour le prendre en charge. Ensuite, vous pouvez initialiser un objet `BluetoothLeAdvertiser` et avec cela, vous pouvez lancer des opérations publicitaires:

```
        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);
        }
```

Utiliser un serveur Gatt

Pour que votre appareil puisse agir en tant que périphérique, vous devez d'abord ouvrir un `BluetoothGattServer` et le remplir avec au moins un `BluetoothGattService` et un `BluetoothGattCharacteristic` :

```
BluetoothGattServer server=bluetoothManager.openGattServer(context,
bluetoothGattServerCallback);

BluetoothGattService service = new BluetoothGattService(SERVICE_UUID,
BluetoothGattService.SERVICE_TYPE_PRIMARY);
```

Ceci est un exemple d'une caractéristique BluetoothGattCharacter avec des autorisations d'écriture, de lecture et de notification complètes. Selon vos besoins, vous souhaiterez peut-être affiner les autorisations que vous accordez à cette caractéristique:

```
BluetoothGattCharacteristic characteristic = new
BluetoothGattCharacteristic(CHARACTERISTIC_UUID,
                    BluetoothGattCharacteristic.PROPERTY_READ |
BluetoothGattCharacteristic.PROPERTY_WRITE |
                            BluetoothGattCharacteristic.PROPERTY_NOTIFY,
                    BluetoothGattCharacteristic.PERMISSION_READ |
BluetoothGattCharacteristic.PERMISSION_WRITE);

characteristic.addDescriptor(new BluetoothGattDescriptor(UUID.fromString("00002902-0000-1000-
8000-00805f9b34fb"), BluetoothGattCharacteristic.PERMISSION_WRITE));

service.addCharacteristic(characteristic);

server.addService(service);
```

`BluetoothGattServerCallback` est chargé de recevoir tous les événements liés à votre `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);
                }
```

Chaque fois que vous recevez une demande d'écriture / lecture sur une caractéristique ou un descripteur, vous devez lui envoyer une réponse pour que la demande soit complétée avec succès:

```
@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);
}
```

Lire Bluetooth Low Energy en ligne: https://riptutorial.com/fr/android/topic/10020/bluetooth-low-energy

Chapitre 38: Boîte AlertDialog Animée

Introduction

Dialogue d'animation qui affiche certains effets d'animation. Vous pouvez obtenir des animations pour les boîtes de dialogue telles que Fadein, Slideleft, Slidetop, SlideBottom, Sliderright, Fall, Newspager, Fliph, Flipv, RotateBottom, RotateLeft, Slit, Shake, Sidefill application attrayante ..

Examples

Mettez le code ci-dessous pour la boîte de dialogue animée ...

```
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();
    }
}
```

Ajoutez les lignes ci-dessous dans votre build.gradle pour inclure le NifyBuilder (CustomView)

build.gradle

```
dependencies {

    compile 'com.nineoldandroids:library:2.4.0'

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

}
```

Lien de référence: https://github.com/sd6352051/NiftyDialogEffects

Lire Boîte AlertDialog Animée en ligne: https://riptutorial.com/fr/android/topic/10654/boite-alertdialog-animee

Chapitre 39: BottomNavigationView

Introduction

La vue de navigation du bas a été dans les directives de conception de matériel pendant un certain temps, mais il n'a pas été facile pour nous de l'implémenter dans nos applications.

Certaines applications ont créé leurs propres solutions, tandis que d'autres ont eu recours à des bibliothèques Open Source tierces pour faire le travail.

Maintenant que la bibliothèque d'aide à la conception voit l'ajout de cette barre de navigation inférieure, jetons un coup d'œil à la façon dont nous pouvons l'utiliser!

Remarques

Représente une barre de navigation inférieure standard pour l'application. C'est une implémentation de la navigation de fond de conception matérielle.

Liens:

- Javadoc officiel

Examples

Implémentation de base

Pour ajouter la `BottomNavigationView` procédez `BottomNavigationView` suit:

1. Ajoutez dans votre `build.gradle` la **dépendance** :

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

2. Ajoutez le `BottomNavigationView` dans **votre mise** en **page** :

```
<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. Créez le **menu** pour remplir la vue:

```
<!-- 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. Joignez **un écouteur** pour les événements de clic:

```
//Get the view
BottomNavigationView bottomNavigationView = (BottomNavigationView)
                findViewById(R.id.bottom_navigation);
//Attach the listener
bottomNavigationView.setOnNavigationItemSelectedListener(
        new BottomNavigationView.OnNavigationItemSelectedListener() {
            @Override
            public boolean onNavigationItemSelected(@NonNull MenuItem item) {
                switch (item.getItemId()) {

                    case R.id.my_action1:
                        //Do something...
                        break;

                    //...
                }
                return true;//returning false disables the Navigation bar animations
            }
        });
```

Checkout demo code à BottomNavigation-Demo

Personnalisation de BottomNavigationView

Remarque: Je suppose que vous savez comment utiliser ***`BottomNavigationView`*** *.*

Cet exemple explique comment ajouter un sélecteur pour **`BottomNavigationView`** . Ainsi, vous pouvez indiquer sur l'interface utilisateur des icônes et des textes.

Créez drawable `bottom_navigation_view_selector.xml` comme

```
<?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>
```

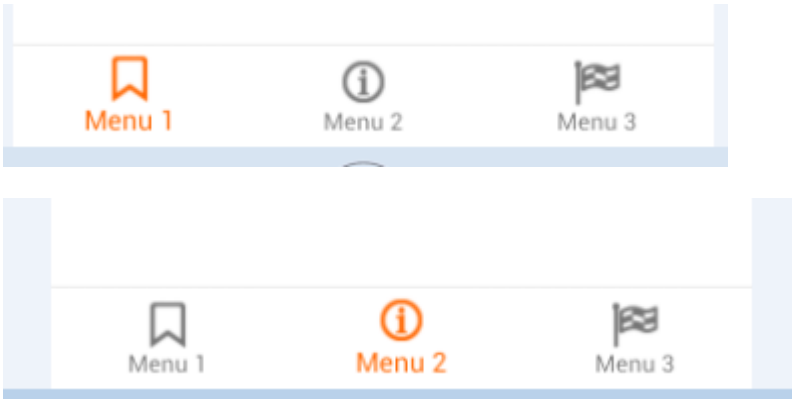
Et utiliser les attributs ci-dessous dans **`BottomNavigationView`** dans le fichier de disposition

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

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

> Dans l'exemple ci-dessus, j'ai utilisé le même sélecteur `bottom_navigation_view_selector` pour `app:itemIconTint` et `app:itemTextColor` fois pour conserver les couleurs du texte et des icônes. Mais si votre conception a une couleur différente pour le texte et l'icône, vous pouvez définir 2 sélecteurs différents et les utiliser.

La sortie sera similaire à ci-dessous



Gestion des états activés / désactivés

Créer un sélecteur pour activer / désactiver l'élément de menu.

selector.xml

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

design.xml

```
<android.support.design.widget.BottomNavigationView
    android:id="@+id/bottom_navigation"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    app:itemBackground="@color/colorPrimary"
    app:itemIconTint="@drawable/nav_item_color_state"
    app:itemTextColor="@drawable/nav_item_color_state"
    app:menu="@menu/bottom_navigation_main" />
```

Autoriser plus de 3 menus

Cet exemple est strictement une solution de contournement, car il n'existe actuellement aucun moyen de désactiver un comportement appelé ShiftMode.

Créez une fonction en tant que telle.

```
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);
    }
}
```

Cela désactive le comportement Shifting du menu lorsque le nombre d'éléments dépasse 3 ns.

USAGE

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

Proguard Problème : Ajouter le fichier de configuration de ligne suivant aussi, sinon cela ne fonctionnerait pas.

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

Vous pouvez également créer une classe et accéder à cette méthode à partir de là. Voir l'original Répondre ici

REMARQUE : Ceci est un **HOTFIX** basé sur **Reflection** , veuillez le mettre à jour une fois que la bibliothèque de support de Google est mise à jour avec un appel de fonction direct.

Lire BottomNavigationView en ligne:
https://riptutorial.com/fr/android/topic/7565/bottomnavigationview

Chapitre 40: Bouton

Syntaxe

- <Bouton ... />
- android: onClick = "methodname"
- button.setOnClickListener (new OnClickListener () {...});
- classname de classe publique implémente View.OnLongClickListener

Examples

en ligne surClickListener

Disons que nous avons un bouton (nous pouvons le créer par programmation ou le lier depuis une vue en utilisant findViewbyId (), etc ...)

```
Button btnOK = (...)
```

Maintenant, créez une classe anonyme et définissez-la en ligne.

```
btnOk.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // Do stuff here...
    }
});
```

Utiliser la mise en page pour définir une action de clic

Lorsque nous créons un bouton dans la présentation, nous pouvons utiliser l'attribut `android:onClick` pour référencer une méthode dans le code pour gérer les clics.

Bouton

```
<Button
    android:width="120dp"
    android:height="wrap_content"
    android:text="Click me"
    android:onClick="handleClick" />
```

Ensuite, dans votre activité, créez la méthode `handleClick` .

```
public void handleClick(View v) {
    // Do whatever.
}
```

Utiliser le même événement de clic pour une ou plusieurs vues dans le XML

Lorsque nous créons une vue dans la présentation, nous pouvons utiliser l'attribut android: onClick pour référencer une méthode dans l'activité ou le fragment associé pour gérer les événements de clic.

Mise en page XML

```
<Button android:id="@+id/button"
    ...
    // onClick should reference the method in your activity or fragment
    android:onClick="doSomething" />

// Note that this works with any class which is a subclass of View, not just Button
<ImageView android:id="@+id/image"
    ...
    android:onClick="doSomething" />
```

Code d'activité / fragment

Dans votre **code** , créez la méthode nommée, où v sera la vue touchée, et faites quelque chose pour chaque vue qui appelle cette méthode.

```
public void doSomething(View v) {
    switch(v.getId()) {
        case R.id.button:
            // Button was clicked, do something.
            break;
        case R.id.image:
            // Image was clicked, do something else.
            break;
    }
}
```

Si vous le souhaitez, vous pouvez également utiliser une méthode différente pour chaque vue (dans ce cas, vous n'avez bien sûr pas besoin de vérifier l'ID).

Écouter les événements de clic long

Pour attraper un clic long et l'utiliser, vous devez fournir un écouteur approprié au bouton:

```
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);
```

Définition d'un écouteur externe

Quand devrais-je l'utiliser

- Lorsque le code à l'intérieur d'un écouteur en ligne est trop gros et que votre méthode / classe devient moche et difficile à lire
- Vous souhaitez effectuer la même action dans différents éléments (vue) de votre application

---

Pour ce faire, vous devez créer une classe implémentant l'un des écouteurs de l' API View .

Par exemple, aidez lorsque vous cliquez longtemps sur un élément:

```
public class HelpLongClickListener implements View.OnLongClickListener
{
    public HelpLongClickListener() {
    }

    @Override
    public void onLongClick(View v) {
        // show help toast or popup
    }
}
```

Il vous suffit alors d'avoir un attribut ou une variable dans votre `Activity` pour l'utiliser:

```
HelpLongClickListener helpListener = new HelpLongClickListener(...);

button1.setOnClickListener(helpListener);
button2.setOnClickListener(helpListener);
label.setOnClickListener(helpListener);
button1.setOnClickListener(helpListener);
```

REMARQUE: la définition des écouteurs dans une classe séparée présente un inconvénient: elle ne peut pas accéder directement aux champs de classe. Vous devez donc transmettre des données (contexte, vue) via le constructeur, sauf si vous définissez des attributs publics ou définissez des geters.

Custom Click Listener pour empêcher plusieurs clics rapides

Afin d' **empêcher un bouton de se déclencher plusieurs fois dans un court laps de temps** (disons 2 clics en moins d'une seconde, ce qui peut causer de sérieux problèmes si le flux n'est pas contrôlé), on peut implémenter un ***SingleClickListener*** personnalisé.

Ce ClickListener définit un intervalle de temps spécifique comme seuil (par exemple, 1000 ms). Lorsque vous cliquez sur le bouton, une vérification est effectuée pour voir si le déclencheur a été exécuté au cours de la période de temps que vous avez définie et sinon, il le déclenchera.

```
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);

}
```

Et dans la classe, SingleClickListener est associé au Button en jeu

```
myButton = (Button) findViewById(R.id.my_button);
myButton.setOnClickListener(new SingleClickListener() {
    @Override
    public void performClick(View view) {
        // do stuff
    }
});
```

Personnalisation du style de bouton

Il existe de nombreux moyens de personnaliser l'apparence d'un bouton. Cet exemple présente plusieurs options:

Option 0: Utiliser ThemeOverlay (actuellement le moyen le plus simple et le plus rapide)

Créez un nouveau style dans votre fichier de styles:

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>
```

Ensuite, dans la disposition où vous placez votre bouton (par exemple, 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: Créez votre propre style de bouton

Dans values / styles.xml, créez un nouveau style pour votre bouton:

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>
```

Ensuite, dans la mise en page où vous placez votre bouton (par exemple dans 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: Attribuez un dessin pour chacun des états de vos boutons

Créez un fichier xml dans un dossier pouvant être dessiné appelé «mybuttondrawable.xml» pour définir la ressource pouvant être dessinée de chacun des états de votre bouton:

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>
```

Chacun de ces tirables peut être des images (par exemple, mybutton_disabled.png) ou des fichiers xml définis par vous et stockés dans le dossier drawables. Par exemple:

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>
```

Ensuite, dans la disposition où vous placez votre bouton (par exemple, 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: ajoutez votre style de bouton à votre thème d'application

Vous pouvez remplacer le style de bouton Android par défaut dans la définition du thème de votre application (dans values / styles.xml).

styles.xml

```
<resources>
     <style name="AppTheme" parent="android:Theme">
            <item name="colorPrimary">@color/colorPrimary</item>
            <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
            <item name="colorAccent">@color/colorAccent</item>
              <item name="android:button">@style/mybutton</item>
     </style>

     <style name="mybutton" parent="android:style/Widget.Button">
             <item name="android:gravity">center_vertical|center_horizontal</item>
                <item name="android:textColor">#FFFFFFFF</item>
                <item name="android:shadowColor">#FF000000</item>
                <item name="android:shadowDx">0</item>
```

```
                <item name="android:shadowDy">-1</item>
                <item name="android:shadowRadius">0.2</item>
                <item name="android:textSize">16dip</item>
                <item name="android:textStyle">bold</item>
                <item name="android:background">@drawable/anydrawable</item>
     </style>
</resources>
```

Option 4: superposer une couleur sur le style de bouton par défaut par programmation

Trouvez simplement votre bouton dans votre activité et appliquez un filtre de couleur:

```
Button mybutton = (Button) findViewById(R.id.mybutton);
mybutton.getBackground().setColorFilter(anycolor, PorterDuff.Mode.MULTIPLY)
```

Vous pouvez vérifier différents modes de mélange ici et de bons exemples ici .

Lire Bouton en ligne: https://riptutorial.com/fr/android/topic/5607/bouton

Chapitre 41: BroadcastReceiver

Introduction

BroadcastReceiver (récepteur) est un composant Android qui vous permet de vous inscrire à des événements système ou d'application. Tous les récepteurs enregistrés pour un événement sont notifiés par le moteur d'exécution Android une fois que cet événement se produit.

Par exemple, une émission annonçant que l'écran est éteint, que la batterie est faible ou qu'une photo a été capturée.

Les applications peuvent également lancer des diffusions, par exemple pour permettre aux autres applications de savoir que certaines données ont été téléchargées sur le périphérique et peuvent être utilisées.

Examples

Introduction au récepteur de diffusion

Un récepteur de diffusion est un composant Android qui vous permet de vous inscrire à des événements système ou d'application.

Un récepteur peut être enregistré via le fichier `AndroidManifest.xml` ou dynamiquement via la méthode `Context.registerReceiver()` .

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

Ici, j'ai pris un exemple de `ACTION_BOOT_COMPLETED` qui est déclenché par le système une fois que Android a terminé le processus de démarrage.

Vous pouvez enregistrer un récepteur dans un fichier manifeste comme ceci:

```
<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>
```

Maintenant que le périphérique est démarré, la méthode `onReceive()` sera appelée et vous pourrez alors faire votre travail (par exemple, démarrer un service, démarrer une alarme).

Notions de base sur BroadcastReceiver

Les BroadcastReceivers sont utilisés pour recevoir les intentions de diffusion envoyées par le système d'exploitation Android, d'autres applications ou dans la même application.

Chaque intention est créée avec un *filtre d'intention* , qui nécessite une *action* String. Des informations supplémentaires peuvent être configurées dans Intent.

De même, BroadcastReceivers s'enregistre pour recevoir des intentions avec un filtre d'intention particulier. Ils peuvent être enregistrés par programmation:

```
mContext.registerReceiver(new BroadcastReceiver() {
    @Override
   public void onReceive(Context context, Intent intent) {
      //Your implementation goes here.
   }
}, new IntentFilter("Some Action"));
```

ou dans le fichier `AndroidManifest.xml` :

```
<receiver android:name=".MyBroadcastReceiver">
    <intent-filter>
        <action android:name="Some Action"/>
    </intent-filter>
</receiver>
```

Pour recevoir l'intention, définissez l'action sur quelque chose documenté par Android OS, par une autre application ou API, ou dans votre propre application, en utilisant `sendBroadcast` :

```
mContext.sendBroadcast(new Intent("Some Action"));
```

En outre, Intent peut contenir des informations, telles que des chaînes, des primitives et des *parcelles* , qui peuvent être affichées dans `onReceive` .

Utiliser LocalBroadcastManager

LocalBroadcastManager est utilisé pour envoyer des Intentions de diffusion dans une application, sans les exposer à des écouteurs indésirables.

Utiliser *LocalBroadcastManager* est plus efficace et plus sûr que d'utiliser directement `context.sendBroadcast()` , car vous n'avez pas à vous soucier des diffusions simulées par d'autres applications, ce qui peut constituer un risque de sécurité.

Voici un exemple simple d'envoi et de réception d'émissions locales:

```
BroadcastReceiver receiver = new BroadcastReceiver() {
    @Override
```

```
    public void onReceive(Context context, Intent intent) {
        if (intent.getAction().equals("Some Action")) {
            //Do something
        }
    }
});

LocalBroadcastManager manager = LocalBroadcastManager.getInstance(mContext);
manager.registerReceiver(receiver, new IntentFilter("Some Action"));

// onReceive() will be called as a result of this call:
manager.sendBroadcast(new Intent("Some Action"));//See also sendBroadcastSync

//Remember to unregister the receiver when you are done with it:
manager.unregisterReceiver(receiver);
```

Récepteur de diffusion Bluetooth

ajouter une autorisation dans votre fichier manifeste

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

Dans votre fragment (ou activité)

- Ajouter la méthode du récepteur

```
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;
    }
};
```

Enregistrement de diffusion

- Appelez cette méthode sur onResume ()

```
private void registerBroadcastManager(){
    final LocalBroadcastManager manager = LocalBroadcastManager.getInstance(getActivity());
    manager.registerReceiver(mBluetoothStatusChangedReceiver, new
```

```
IntentFilter(Constants.BROADCAST_BLUETOOTH_STATE));
}
```

Annuler la diffusion

- Appelez cette méthode sur onPause ()

```
private void unregisterBroadcastManager(){
    final LocalBroadcastManager manager = LocalBroadcastManager.getInstance(getActivity());
    // Beacon□□□
    manager.unregisterReceiver(mBluetoothStatusChangedReceiver);
}
```

Activation et désactivation d'un récepteur de diffusion par programmation

Pour activer ou désactiver un `BroadcastReceiver` , nous devons obtenir une référence au `PackageManager` et nous avons besoin d'un objet `ComponentName` contenant la classe du récepteur que nous voulons activer / désactiver:

```
ComponentName componentName = new ComponentName(context, MyBroadcastReceiver.class);
PackageManager packageManager = context.getPackageManager();
```

Maintenant, nous pouvons appeler la méthode suivante pour activer le `BroadcastReceiver` :

```
packageManager.setComponentEnabledSetting(
    componentName,
    PackageManager.COMPONENT_ENABLED_STATE_ENABLED,
    PackageManager.DONT_KILL_APP);
```

Ou nous pouvons utiliser `COMPONENT_ENABLED_STATE_DISABLED` pour désactiver le récepteur:

```
packageManager.setComponentEnabledSetting(
    componentName,
    PackageManager.COMPONENT_ENABLED_STATE_DISABLED,
    PackageManager.DONT_KILL_APP);
```

BroadcastReceiver pour gérer les événements BOOT_COMPLETED

L'exemple ci-dessous montre comment créer un `BroadcastReceiver` capable de recevoir des événements `BOOT_COMPLETED` . De cette façon, vous pouvez démarrer un `Service` ou démarrer une `Activity` dès que le périphérique est mis sous tension.

En outre, vous pouvez utiliser les événements `BOOT_COMPLETED` pour restaurer vos alarmes car elles sont détruites lorsque le périphérique est mis hors tension.

REMARQUE: l'utilisateur doit avoir démarré l'application au moins une fois avant de pouvoir recevoir l'action `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
            }
        }
    }
}
```

Exemple de LocalBroadcastManager

Un BroadcastReceiver est essentiellement un mécanisme permettant de relayer les intentions via le système d'exploitation pour effectuer des actions spécifiques. Une définition classique étant

> "Un récepteur de diffusion est un composant Android qui vous permet d'enregistrer des événements système ou d'application."

LocalBroadcastManager est un moyen d'envoyer ou de recevoir des diffusions au sein d'un processus d'application. Ce mécanisme a beaucoup d'avantages

1. Comme les données restent à l'intérieur du processus d'application, les données ne peuvent pas être divulguées.
2. Les LocalBroadcasts sont résolus plus rapidement, car la résolution d'une diffusion normale se produit au moment de l'exécution du système d'exploitation.

Un exemple simple de LocalBroastManager est:

SenderActivity

```
  Intent intent = new Intent("anEvent");
  intent.putExtra("key", "This is an event");
  LocalBroadcastManager.getInstance(this).sendBroadcast(intent);
```

ReceiverActivity

1. Enregistrer un destinataire

```
LocalBroadcastManager.getInstance(this).registerReceiver(aLBReceiver,
              new IntentFilter("anEvent"));
```

2. Un objet concret pour effectuer une action lorsque le récepteur est appelé

```
private BroadcastReceiver aLBReceiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        // perform action here.
    }
};
```

3. désinscrire lorsque la vue n'est plus visible.

```
@Override
protected void onPause() {
    // Unregister since the activity is about to be closed.
    LocalBroadcastManager.getInstance(this).unregisterReceiver(aLBReceiver);
    super.onDestroy();
}
```

Communiquer deux activités via un récepteur de diffusion personnalisé

Vous pouvez communiquer deux activités pour que l'activité A puisse être informée d'un événement se produisant dans l'activité B.

Activité 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
    }
};
```

Activité B

```
final String eventName = "your.package.goes.here.EVENT";

private void launchEvent() {
    Intent eventIntent = new Intent(eventName);
    this.sendBroadcast(eventIntent);
}
```

Bien entendu, vous pouvez ajouter plus d'informations à la diffusion en ajoutant des extras à l'intention transmise entre les activités. Pas ajouté pour garder l'exemple aussi simple que possible.

Diffusion collante

Si nous utilisons la méthode sendStickyBroadcast (intention), l'intention correspondante est collante, ce qui signifie que l'intention que vous envoyez reste autour de la fin de la diffusion. Un StickyBroadcast, comme son nom l'indique, est un mécanisme permettant de lire les données d'une diffusion, une fois la diffusion terminée. Cela peut être utilisé dans un scénario où vous souhaiterez peut-être vérifier `Activity's onCreate()` la valeur d'une clé dans l'objet avant l'activité a été lancée.

```
Intent intent = new Intent("com.org.action");
intent.putExtra("anIntegerKey", 0);
sendStickyBroadcast(intent);
```

Utiliser les diffusions commandées

Les diffusions ordonnées sont utilisées lorsque vous devez spécifier une priorité pour les auditeurs de diffusion.

Dans cet exemple, `firstReceiver` recevra la diffusion toujours avant un `secondReceiver` :

```
final int highPriority = 2;
final int lowPriority = 1;
final String action = "action";

// intent filter for first receiver with high priority
final IntentFilter firstFilter = new IntentFilter(action);
first Filter.setPriority(highPriority);
final BroadcastReceiver firstReceiver = new MyReceiver();

// intent filter for second receiver with low priority
```

```
final IntentFilter secondFilter = new IntentFilter(action);
secondFilter.setPriority(lowPriority);
final BroadcastReceiver secondReceiver = new MyReceiver();

// register our receivers
context.registerReceiver(firstReceiver, firstFilter);
context.registerReceiver(secondReceiver, secondFilter);

// send ordered broadcast
context.sendOrderedBroadcast(new Intent(action), null);
```

De plus, le récepteur de diffusion peut interrompre la diffusion ordonnée:

```
@Override
public void onReceive(final Context context, final Intent intent) {
    abortBroadcast();
}
```

dans ce cas, tous les récepteurs de priorité inférieure ne recevront pas de message de diffusion.

État d'arrêt Android

Depuis Android 3.1, toutes les applications, lors de l'installation, sont arrêtées. En état d'arrêt, l'application ne s'exécutera pour aucune raison, sauf par un lancement manuel d'une activité ou par une intention **explicite** concernant une activité, un service ou une diffusion.

Lorsque vous écrivez une application système qui installe directement des APK, veuillez prendre en compte le fait que l'application nouvellement installée ne recevra aucune diffusion tant qu'elle n'a pas été arrêtée.

Un moyen simple d'activer une application consiste à envoyer une diffusion explicite à cette application. comme la plupart des applications implémentent `INSTALL_REFERRER` , nous pouvons l'utiliser comme point d'accrochage

Analysez le manifeste de l'application installée et envoyez une diffusion explicite à chaque récepteur:

```
Intent intent = new Intent();
intent.addFlags(Intent.FLAG_INCLUDE_STOPPED_PACKAGES);
intent.setComponent(new ComponentName(packageName, fullClassName));
sendBroadcast(intent);
```

Lire BroadcastReceiver en ligne: https://riptutorial.com/fr/android/topic/1460/broadcastreceiver

Chapitre 42: Cache Bitmap

Introduction

Mise en cache par bitmap efficace en mémoire: Cette fonctionnalité est particulièrement importante si votre application utilise des animations car celles-ci seront arrêtées lors du nettoyage du GC et rendront votre application trop lente pour l'utilisateur. Un cache permet de réutiliser des objets coûteux à créer. Si vous chargez un objet en mémoire, vous pouvez le considérer comme un cache pour l'objet. Travailler avec un bitmap dans android est délicat.

Syntaxe

- `LruCache<String, Bitmap> mMemoryCache;//declaration of LruCache object.`
- void addBitmapToMemoryCache (clé de chaîne, bitmap bitmap) {} // déclaration de la méthode générique ajoutant un bitmap à la mémoire cache
- Bitmap getBitmapFromMemCache (clé de chaîne) {} // déclaration de la méthode générique pour obtenir la bimap du cache.

Paramètres

Paramètre	Détails
clé	clé pour stocker le bitmap dans la mémoire cache
bitmap	valeur bitmap qui mettra en mémoire cache

Examples

Cache de bitmap utilisant le cache LRU

Cache LRU

L'exemple de code suivant illustre une implémentation possible de la classe LruCache pour la mise en cache des images.

```
private LruCache<String, Bitmap> mMemoryCache;
```

Ici, la valeur de la chaîne est la clé de la valeur bitmap.

```
// Get max available VM memory, exceeding this amount will throw an
// OutOfMemory exception. Stored in kilobytes as LruCache takes an
// int in its constructor.
final int maxMemory = (int) (Runtime.getRuntime().maxMemory() / 1024);
```

```
// Use 1/8th of the available memory for this memory cache.
final int cacheSize = maxMemory / 8;

mMemoryCache = new LruCache<String, Bitmap>(cacheSize) {
    @Override
    protected int sizeOf(String key, Bitmap bitmap) {
        // The cache size will be measured in kilobytes rather than
        // number of items.
        return bitmap.getByteCount() / 1024;
    }
};
```

Pour ajouter un bitmap au cache mémoire

```
public void addBitmapToMemoryCache(String key, Bitmap bitmap) {
if (getBitmapFromMemCache(key) == null) {
        mMemoryCache.put(key, bitmap);
    }
}
```

Pour obtenir un bitmap du cache mémoire

```
public Bitmap getBitmapFromMemCache(String key) {
    return mMemoryCache.get(key);
}
```

Pour charger un bitmap dans imageview, utilisez simplement **getBitmapFromMemCache ("Pass key").**

Lire Cache Bitmap en ligne: https://riptutorial.com/fr/android/topic/9901/cache-bitmap

Chapitre 43: Camera 2 API

Paramètres

Paramètre	Détails
`CameraCaptureSession`	Une session de capture configurée pour un `CameraDevice` , utilisée pour capturer des images de la caméra ou pour retraiter des images capturées depuis la caméra au cours de la même session.
`CameraDevice`	Une représentation d'une seule caméra connectée à un appareil Android
`CameraCharacteristics`	Les propriétés décrivant un CameraDevice. Ces propriétés sont corrigées pour un CameraDevice donné et peuvent être interrogées via l'interface `getCameraCharacteristics(String)` avec `getCameraCharacteristics(String)`
`CameraManager`	Un gestionnaire de services système pour détecter, caractériser et se connecter à `CameraDevices` . Vous pouvez obtenir une instance de cette classe en appelant `Context.getSystemService()`
`CaptureRequest`	Un ensemble immuable de paramètres et de sorties nécessaires pour capturer une seule image à partir de l'appareil photo. Contient la configuration du matériel de capture (capteur, objectif, flash), le pipeline de traitement, les algorithmes de contrôle et les tampons de sortie. Contient également la liste des surfaces cibles auxquelles envoyer les données d'image pour cette capture. Peut être créé en utilisant une instance de `CaptureRequest.Builder` , obtenue en appelant `createCaptureRequest(int)`
`CaptureResult`	Le sous-ensemble des résultats d'une capture d'image unique provenant du capteur d'image. Contient un sous-ensemble de la configuration finale du matériel de capture (capteur, objectif, flash), du pipeline de traitement, des algorithmes de contrôle et des tampons de sortie. Il est produit par un `CameraDevice` après avoir traité un `CaptureRequest`

Remarques

- Les API Camera2 sont disponibles dans API 21+ (Lollipop et au-delà)
- Même si un appareil Android a officiellement une ROM de 21+, il n'y a aucune garantie qu'il implémente les API Camera2, il appartient totalement au fabricant de l'implémenter ou non (Exemple: LG G2 a un support officiel de Lollipop, mais pas d'API Camera2)
- Avec Camera2, Camera ("Camera1") est obsolète

- Avec une grande puissance, vous avez une grande responsabilité: il est plus facile de la gâcher lorsque vous utilisez ces API.
- Rappelez-vous que si vous voulez seulement prendre une photo dans votre application et l'obtenir simplement, vous **n'avez pas** besoin d'implémenter Camera2, vous pouvez ouvrir l'application appareil photo via un Intent et la recevoir.

Examples

Prévisualiser la caméra principale dans un TextureView

Dans ce cas, la création par rapport à l'API 23, les autorisations sont également gérées.

Vous devez ajouter dans le manifeste l'autorisation suivante (quel que soit le niveau de l'API utilisé):

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

Nous allons créer une activité (Camera2Activity.java) qui remplit un `TextureView` avec l'aperçu de la caméra de l'appareil.

L'activité que nous allons utiliser est un AppCompatActivity typique:

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

Attributs (Vous devrez peut-être lire l'exemple complet pour en comprendre certains)

Le `MAX_PREVIEW_SIZE` garanti par l'API Camera2 est 1920x1080

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

`TextureView.SurfaceTextureListener` gère plusieurs événements de cycle de vie sur une `TextureView` . Dans ce cas, nous écoutons ces événements. Lorsque SurfaceTexture est prêt, nous initialisons la caméra. Lorsque la taille change, nous configurons l'aperçu provenant de la caméra en conséquence

```
private final TextureView.SurfaceTextureListener mSurfaceTextureListener
        = new TextureView.SurfaceTextureListener() {

    @Override
    public void onSurfaceTextureAvailable(SurfaceTexture texture, int width, int height) {
        openCamera(width, height);
    }

    @Override
    public void onSurfaceTextureSizeChanged(SurfaceTexture texture, int width, int height) {
        configureTransform(width, height);
    }

    @Override
```

```
    public boolean onSurfaceTextureDestroyed(SurfaceTexture texture) {
        return true;
    }

    @Override
    public void onSurfaceTextureUpdated(SurfaceTexture texture) {
    }

};
```

Un `CameraDevice` représente la caméra d'un périphérique physique. Dans cet attribut, nous sauvegardons l'ID du `CameraDevice` actuel

```
private String mCameraId;
```

C'est la vue (`TextureView`) que nous utiliserons pour "dessiner" l'aperçu de la caméra.

```
private TextureView mTextureView;
```

Le `CameraCaptureSession` pour la prévisualisation de la caméra

```
private CameraCaptureSession mCaptureSession;
```

Une référence au `CameraDevice` ouvert

```
private CameraDevice mCameraDevice;
```

La `Size` de l'aperçu de la caméra.

```
private Size mPreviewSize;
```

`CameraDevice.StateCallback` est appelé lorsque `CameraDevice` change d'état

```
private final CameraDevice.StateCallback mStateCallback = new CameraDevice.StateCallback() {

    @Override
    public void onOpened(@NonNull CameraDevice cameraDevice) {
        // This method is called when the camera is opened.  We start camera preview here.
        mCameraOpenCloseLock.release();
        mCameraDevice = cameraDevice;
        createCameraPreviewSession();
    }

    @Override
    public void onDisconnected(@NonNull CameraDevice cameraDevice) {
        mCameraOpenCloseLock.release();
        cameraDevice.close();
        mCameraDevice = null;
    }

    @Override
    public void onError(@NonNull CameraDevice cameraDevice, int error) {
        mCameraOpenCloseLock.release();
```

```
        cameraDevice.close();
        mCameraDevice = null;
        finish();
    }

};
```

Un thread supplémentaire pour l'exécution de tâches qui ne doivent pas bloquer l'interface utilisateur

```
private HandlerThread mBackgroundThread;
```

Un `Handler` pour exécuter des tâches en arrière-plan

```
private Handler mBackgroundHandler;
```

Un `ImageReader` qui gère la capture d'images fixes

```
private ImageReader mImageReader;
```

`CaptureRequest.Builder` pour l'aperçu de la caméra

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

`CaptureRequest` généré par `mPreviewRequestBuilder`

```
private CaptureRequest mPreviewRequest;
```

Un `Semaphore` pour empêcher l'application de quitter avant de fermer la caméra.

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

Identifiant constant de la demande d'autorisation

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

Méthodes de cycle de vie Android

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

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

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

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

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

Méthodes liées à Camera2

Ce sont des méthodes qui utilisent les API Camera2

```
private void openCamera(int width, int height) {
    if (ContextCompat.checkSelfPermission(this, Manifest.permission.CAMERA)
            != PackageManager.PERMISSION_GRANTED) {
        requestCameraPermission();
        return;
    }
    setUpCameraOutputs(width, height);
    configureTransform(width, height);
    CameraManager manager = (CameraManager) getSystemService(Context.CAMERA_SERVICE);
    try {
        if (!mCameraOpenCloseLock.tryAcquire(2500, TimeUnit.MILLISECONDS)) {
            throw new RuntimeException("Time out waiting to lock camera opening.");
        }
        manager.openCamera(mCameraId, mStateCallback, mBackgroundHandler);
    } catch (CameraAccessException e) {
        e.printStackTrace();
    } catch (InterruptedException e) {
        throw new RuntimeException("Interrupted while trying to lock camera opening.", e);
    }
}
```

Ferme la caméra actuelle

```
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();
    }
}
```

Configure les variables de membre liées à la caméra

```
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();
    }
}
```

Crée un nouveau `CameraCaptureSession` pour un aperçu de la caméra

```
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();
    }
}
```

Méthodes associées aux autorisations Pour 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);
    }
}
```

Méthodes de thread / gestionnaire d'arrière-plan

```
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();
    }
}
```

Méthodes utilitaires

En fonction des choix de `Size` pris en charge par une caméra, choisissez la plus petite taille au moins en tant que taille de vue de texture respective, aussi grande que la taille maximale respective et dont les proportions correspondent à la valeur spécifiée. Si n'existe pas, choisissez le plus grand qui est au maximum aussi grand que la taille maximale respective, et dont le rapport hauteur / largeur correspond à la valeur spécifiée

```
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];
```

```
    }
}
```

Cette méthode configure la transformation de `Matrix` `mTextureView` en `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);
}
```

Cette méthode compare deux `Size` basées sur leurs zones.

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

Pas grand chose à voir ici

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

Lire Camera 2 API en ligne: https://riptutorial.com/fr/android/topic/619/camera-2-api

Chapitre 44: Caméra et Galerie

Examples

Prendre une photo pleine grandeur à partir de l'appareil photo

Pour prendre une photo, nous devons d'abord déclarer les autorisations requises dans `AndroidManifest.xml` . Nous avons besoin de deux autorisations:

- `Camera` - pour ouvrir l'application de la caméra. Si l'attribut `required` est défini sur `true` vous ne pourrez pas installer cette application si vous n'avez pas de caméra matérielle.
- `WRITE_EXTERNAL_STORAGE` - Cette autorisation est nécessaire pour créer un nouveau fichier, dans lequel la photo capturée sera enregistrée.

AndroidManifest.xml

```
<uses-feature android:name="android.hardware.camera"
          android:required="true" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

L'idée principale en prenant des photos en taille réelle de l'appareil photo est que nous devons créer un nouveau fichier pour la photo, avant d'ouvrir l'application appareil photo et de capturer des photos.

```
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();
    }
}
```

Prendre une photo

Ajoutez une autorisation pour accéder à la caméra au fichier AndroidManifest:

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

Fichier Xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<SurfaceView android:id="@+id/surfaceView" android:layout_height="0dip"
android:layout_width="0dip"></SurfaceView>
<ImageView android:layout_width="wrap_content" android:layout_height="wrap_content"
android:id="@+id/imageView"></ImageView>
</LinearLayout>
```

Activité

```
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;
    }
}
```

Comment démarrer la caméra ou la galerie et enregistrer le résultat de la caméra dans le stockage

Tout d'abord, vous avez besoin des dossiers `Uri` et temporaire et des codes de demande:

```
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;
```

Puis lancez 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();
    }
}
```

Ouverture de la `Camera` et de la `Gallery` :

```
public void openCamera(){
    Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    try {
        mImageCaptureUri = null;
        String state = Environment.getExternalStorageState();
        if (Environment.MEDIA_MOUNTED.equals(state)) {
            mImageCaptureUri = Uri.fromFile(mFileTemp);

        } else {

            mImageCaptureUri = InternalStorageContentProvider.CONTENT_URI;

        }
        intent.putExtra(MediaStore.EXTRA_OUTPUT, mImageCaptureUri);
        intent.putExtra("return-data", true);
        startActivityForResult(intent, REQUEST_CODE_TAKE_PICTURE);
    } catch (Exception e) {

        Log.d("error", "cannot take picture", e);
    }
}

public void openGallery(){
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.JELLY_BEAN
            && ActivityCompat.checkSelfPermission(this,
Manifest.permission.READ_EXTERNAL_STORAGE)
            != PackageManager.PERMISSION_GRANTED) {
        requestPermission(Manifest.permission.READ_EXTERNAL_STORAGE,
                getString(R.string.permission_read_storage_rationale),
                REQUEST_STORAGE_READ_ACCESS_PERMISSION);
    } else {
        Intent intent = new Intent();
        intent.setType("image/*");
        intent.setAction(Intent.ACTION_GET_CONTENT);
        intent.addCategory(Intent.CATEGORY_OPENABLE);
        startActivityForResult(Intent.createChooser(intent, getString(R.string.select_image)),
REQUEST_SELECT_PICTURE);
    }

}
```

Puis dans la méthode `onActivityResult` :

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {

    if (resultCode != RESULT_OK) {
        return;
    }
    Bitmap bitmap;

    switch (requestCode) {

        case REQUEST_SELECT_PICTURE:
            try {
                Uri uri = data.getData();
                try {
                    bitmap = MediaStore.Images.Media.getBitmap(getContentResolver(), uri);
                    Bitmap bitmapScaled = Bitmap.createScaledBitmap(bitmap, 800, 800, true);
                    Drawable drawable=new BitmapDrawable(bitmapScaled);
```

```
                    mImage.setImageDrawable(drawable);
                    mImage.setVisibility(View.VISIBLE);
                } catch (IOException e) {
                    Log.v("act result", "there is an error : "+e.getContent());
                }
            } catch (Exception e) {
                Log.v("act result", "there is an error : "+e.getContent());
            }
            break;
        case REQUEST_CODE_TAKE_PICTURE:
            try{
               Bitmap bitmappicture = MediaStore.Images.Media.getBitmap(getContentResolver() ,
mImageCaptureUri);
               mImage.setImageBitmap(bitmappicture);
               mImage.setVisibility(View.VISIBLE);
           }catch (IOException e){
              Log.v("error camera",e.getMessage());
           }
           break;
    }
    super.onActivityResult(requestCode, resultCode, data);
}
```

Vous avez besoin de ces autorisations dans `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" />
```

Et vous devez gérer les autorisations d'exécution telles que le stockage externe en lecture / écriture, etc.

Je vérifie l'autorisation `READ_EXTERNAL_STORAGE` dans ma méthode `openGallery` :

Ma méthode `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);
    }
}
```

Ensuite, remplacez la méthode `onRequestPermissionsResult` :

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

```
@NonNull int[] grantResults) {
    switch (requestCode) {
        case REQUEST_STORAGE_READ_ACCESS_PERMISSION:
            if (grantResults[0] == PackageManager.PERMISSION_GRANTED) {
                handleGallery();
            }
            break;
        default:
            super.onRequestPermissionsResult(requestCode, permissions, grantResults);
    }
}
```

méthode `showAlertDialog` :

```
protected void showAlertDialog(@Nullable String title, @Nullable String message,
                               @Nullable DialogInterface.OnClickListener
onPositiveButtonClickListener,
                               @NonNull String positiveText,
                               @Nullable DialogInterface.OnClickListener
onNegativeButtonClickListener,
                               @NonNull String negativeText) {
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setTitle(title);
    builder.setMessage(message);
    builder.setPositiveButton(positiveText, onPositiveButtonClickListener);
    builder.setNegativeButton(negativeText, onNegativeButtonClickListener);
    mAlertDialog = builder.show();
}
```

Définir la résolution de la caméra

Définissez Haute résolution par programmation.

```
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);
```

Décoder le bitmap correctement pivoté à partir de l'URI récupéré avec l'intention

```
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;
}
```

Lire Caméra et Galerie en ligne: https://riptutorial.com/fr/android/topic/4789/camera-et-galerie

Chapitre 45: Canal de notification Android O

Introduction

Les canaux de notification nous permettent aux développeurs d'applications de regrouper nos notifications en groupes (canaux), l'utilisateur ayant la possibilité de modifier les paramètres de notification pour l'ensemble du canal.

Syntaxe

1. class NotificationUtils {} // Pour créer un canal de notification
2. createChannel () // Méthode générique de création de notification

Paramètres

Méthode	La description
IMPORTANCE_MAX	inutilisé
IMPORTANCE_HIGH	montre partout, fait du bruit et regarde
IMPORTANCE_DEFAULT	montre partout, fait du bruit, mais ne s'immisce pas visuellement
IMPORTANCE_LOW	montre partout, mais n'est pas intrusif
IMPORTANCE_MIN	montre seulement à l'ombre, au-dessous du pli
IMPORTANCE_NONE	une notification sans importance; ne se montre pas à l'ombre

Examples

Canal de notification

Que sont les canaux de notification?

Les canaux de notification nous permettent aux développeurs d'applications de regrouper nos notifications en groupes (canaux), l'utilisateur pouvant modifier les paramètres de notification pour l'ensemble du canal en même temps. Par exemple, pour chaque canal, les utilisateurs peuvent complètement bloquer toutes les notifications, remplacer le niveau d'importance ou autoriser l'affichage d'un badge de notification. Cette nouvelle fonctionnalité permet d'améliorer considérablement l'expérience utilisateur d'une application.

Créer des canaux de notification

```
 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;
}}
```

Dans le code ci-dessus, nous avons créé deux instances de NotificationChannel, en transmettant uniqueid un nom de canal, ainsi qu'un niveau d'importance dans son constructeur. Pour chaque canal de notification, nous avons appliqué les caractéristiques suivantes.

1. Du son
2. Lumières
3. Vibration
4. Notification à afficher sur l'écran de verrouillage.

Enfin, nous avons obtenu le NotificationManager du système, puis enregistré le canal en appelant la méthode createNotificationChannel (), en passant le canal que nous avons créé.

Nous pouvons créer plusieurs canaux de notification à la fois avec createNotificationChannels (), en transmettant une liste Java d'instances NotificationChannel. Vous pouvez obtenir tous les canaux de notification pour une application avec getNotificationChannels () et obtenir un canal spécifique avec getNotificationChannel (), en ne transmettant que l'ID du canal comme argument.

Niveau d'importance dans les canaux de notification

Méthode	La description
IMPORTANCE_MAX	inutilisé
IMPORTANCE_HIGH	montre partout, fait du bruit et regarde
IMPORTANCE_DEFAULT	montre partout, fait du bruit, mais ne s'immisce pas visuellement
IMPORTANCE_LOW	montre partout, mais n'est pas intrusif, la valeur est 0
IMPORTANCE_MIN	montre seulement à l'ombre, au-dessous du pli
IMPORTANCE_NONE	une notification sans importance; ne se montre pas à l'ombre

Créer une notification et publier sur le canal

Nous avons créé deux notifications en utilisant NotificationUtils une autre en utilisant 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);
}
```

Nous pouvons également définir NotificationChannel en utilisant Notification.Builder (). Pour cela, nous pouvons utiliser ***setChannel (String channelId)*** .

Mettre à jour les paramètres du canal de notification

Une fois que vous créez un canal de notification, l'utilisateur est responsable de ses paramètres et de son comportement. Vous pouvez appeler createNotificationChannel () à nouveau pour renommer un canal de notification existant ou mettre à jour sa description. L'exemple de code suivant décrit comment vous pouvez rediriger un utilisateur vers les paramètres d'un canal de notification en créant une intention de démarrer une activité. Dans ce cas, l'intention requiert des données étendues, notamment l'ID du canal de notification et le nom du package de votre application.

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

Fichier XML:

```
<!--...-->
<Button
    android:id="@+id/btn_android_notif_settings"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Notification Settings"/>
<!--...-->
```

Suppression du canal de notification

Vous pouvez supprimer les canaux de notification en appelant deleteNotificationChannel ().

```
NotificationManager mNotificationManager =
        (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);
// The id of the channel.
String id = "my_channel_01";
NotificationChannel mChannel = mNotificationManager.getNotificationChannel(id);
mNotificationManager.deleteNotificationChannel(mChannel);
```

Maintenant, créez MainActivity et 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

nous allons éditer notre MainActivity pour que nous puissions obtenir le titre et l'auteur des composants EditText, puis les envoyer au canal Android. Nous obtenons le Notification.Builder pour le canal Android que nous avons créé dans notre NotificationUtils, puis nous notifions le NotificationManager.

import android.app.Notification; importer android.os.Bundle; import android.support.v7.app.AppCompatActivity; import android.text.TextUtils; importer 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());
                }
            }
        });
    }
}
```

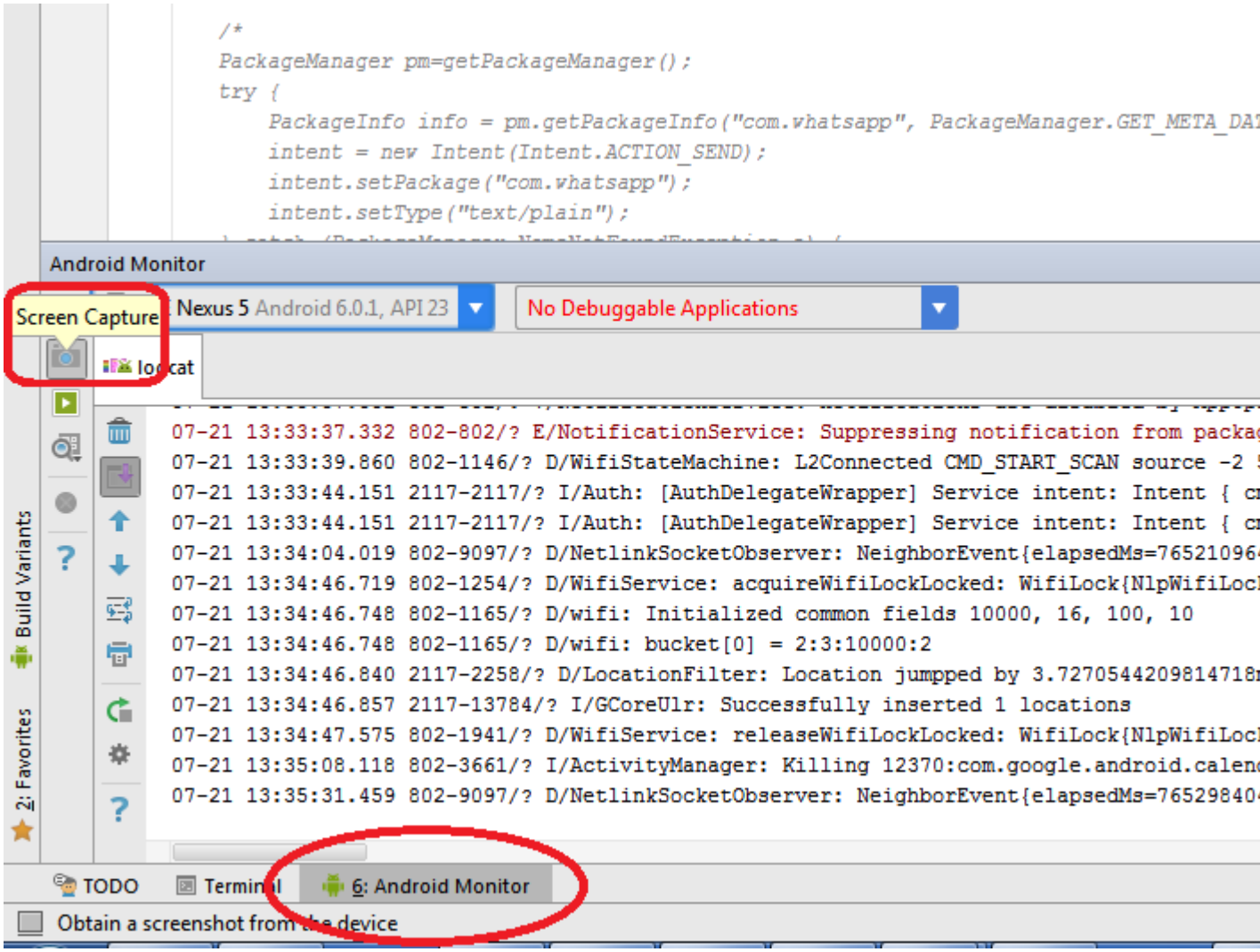
Lire Canal de notification Android O en ligne: https://riptutorial.com/fr/android/topic/10018/canal-de-notification-android-o

Chapitre 46: Capturer des captures d'écran

Examples

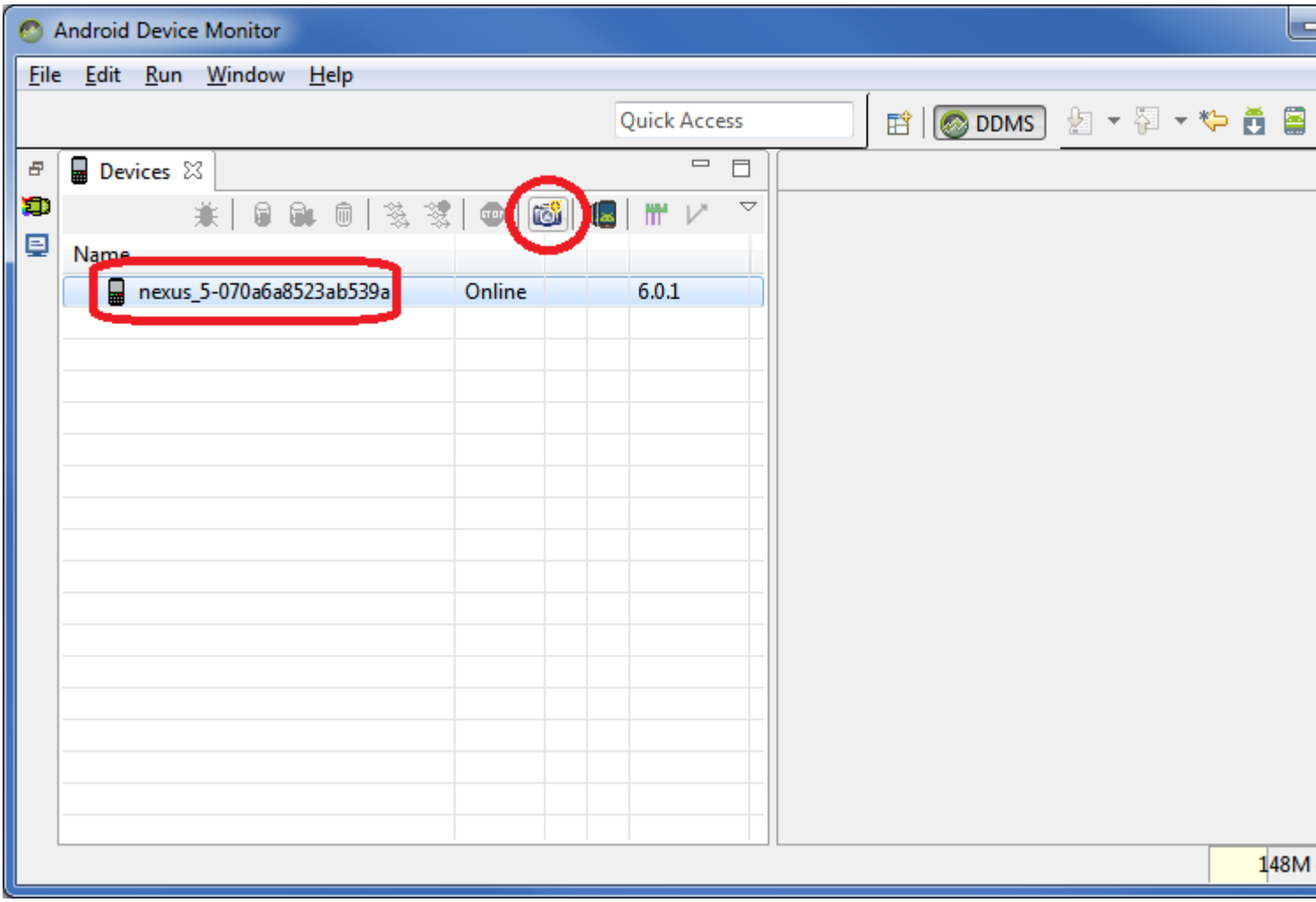
Capture d'écran via Android Studio

1. Ouvrir l'onglet Monitor Android
2. Cliquez sur le bouton de capture d'écran



Capture d'écran via Android Device Monitor

1. Ouvrez Android Device Monitor (*ex: C: <ANDROID_SDK_LOCATION> \ tools \ monitor.bat*)
2. Sélectionnez votre appareil
3. Cliquez sur le bouton de capture d'écran



Capture d'écran via ADB

L'exemple ci-dessous enregistre une capture d'écran sur le stockage interne des périphériques.

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

Capture d'écran via ADB et enregistrement directement sur votre PC

Si vous utilisez Linux (ou Windows avec Cygwin), vous pouvez exécuter:

```
adb shell screencap -p | sed 's/\r$//' > screenshot.png
```

Prendre une capture d'écran d'une vue particulière

Si vous souhaitez prendre une capture d'écran d'un View `v` particulier, vous pouvez utiliser le code suivant:

```
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();
}
```

Lire Capturer des captures d'écran en ligne: https://riptutorial.com/fr/android/topic/4506/capturer-des-captures-d-ecran

Chapitre 47: CardView

Introduction

Un FrameLayout avec un fond d'angle arrondi et une ombre.

CardView utilise la propriété d'altitude sur Lollipop pour les ombres et revient à une implémentation d'ombre émulée personnalisée sur les anciennes plates-formes.

En raison de la nature coûteuse du détourage des coins arrondis, sur les plates-formes antérieures à Lollipop, CardView ne coupe pas ses enfants qui se croisent avec des coins arrondis. Au lieu de cela, il ajoute un remplissage pour éviter une telle intersection (voir setPreventCornerOverlap (boolean) pour modifier ce comportement).

Paramètres

Paramètre	Détails
cardBackgroundColor	Couleur de fond pour CardView.
cardCornerRadius	Rayon d'angle pour CardView.
carteElevation	Élévation pour CardView.
cardMaxElevation	Altitude maximale pour CardView.
cardPreventCornerOverlap	Ajoutez un remplissage à CardView sur v20 et avant pour éviter les intersections entre le contenu de la carte et les coins arrondis.
cardUseCompatPadding	Ajouter un remplissage dans l'API v21 + aussi pour avoir les mêmes mesures avec les versions précédentes. Peut être une valeur booléenne, telle que "true" ou "false".
contentPadding	Rembourrage intérieur entre les bords de la carte et les enfants du CardView.
contentPaddingBottom	Rembourrage intérieur entre le bord inférieur de la carte et les enfants du CardView.
contentPaddingLeft	Rembourrage intérieur entre le bord gauche de la carte et les enfants du CardView.
contentPaddingRight	Élévation pour CardView.
carteElevation	Rembourrage intérieur entre le bord droit de la carte et les enfants du CardView.

Paramètre	Détails
contentPaddingTop	Rembourrage intérieur entre le bord supérieur de la carte et les enfants du CardView.

Remarques

`CardView` utilise des ombres réelles et dynamiques sur Lollipop (API 21) et au-delà. Cependant, avant que Lollipop `CardView` retombe dans une implémentation d'ombre programmatique.

Si vous essayez de faire un `ImageView` en forme dans les coins arrondis d'un `CardView` , vous remarquerez peut - être il ne semble pas correcte pré-Lollipop (API 21). Pour résoudre ce problème, vous devez appeler `setPreventCornerOverlap(false)` sur votre `CardView` ou ajouter `app:cardPreventCornerOverlap="false"` à votre mise en page.

Avant d'utiliser `CardView` vous devez ajouter la dépendance de la bibliothèque de support dans le fichier `build.gradle` :

```
dependencies{
    compile 'com.android.support:cardview-v7:25.2.0'
}
```

Un numéro de la dernière version peut être trouvé ici

Documentation officielle:

https://developer.android.com/reference/android/support/v7/widget/CardView.html
https://developer.android.com/training/material/lists-cards.html

Examples

Premiers pas avec CardView

`CardView` est un membre de la bibliothèque de support Android et fournit une disposition pour les cartes.

Pour ajouter `CardView` à votre projet, ajoutez la ligne suivante à vos dépendances `build.gradle` .

```
compile 'com.android.support:cardview-v7:25.1.1'
```

Un numéro de la dernière version peut être trouvé ici

Dans votre mise en page, vous pouvez ensuite ajouter les éléments suivants pour obtenir une carte.

```
<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>
```

Vous pouvez ensuite ajouter d'autres mises en page à l'intérieur de la carte et les inclure dans une carte.

En outre, CardView peut être rempli avec tout élément d'interface utilisateur et manipulé à partir du code .

```
<?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>
```

Personnalisation de CardView

CardView fournit un rayon d'élévation et d'angle par défaut afin que les cartes aient une apparence cohérente sur toutes les plates-formes.

Vous pouvez personnaliser ces valeurs par défaut en utilisant ces attributs dans le fichier xml:

1. `card_view:cardElevation` attribut `card_view:cardElevation` ajoute l'élévation dans CardView.
2. `card_view:cardBackgroundColor` attribut `card_view:cardBackgroundColor` est utilisé pour personnaliser la couleur d'arrière-plan de l'arrière-plan de CardView (vous pouvez donner n'importe quelle couleur).
3. `card_view:cardCornerRadius` attribut `card_view:cardCornerRadius` est utilisé pour courbe 4 bords de CardView
4. `card_view:contentPadding` attribut `card_view:contentPadding` ajoute un remplissage entre la carte et les enfants de la carte

Remarque: card_view est un espace de noms défini dans la vue de disposition parent la plus haute. xmlns: card_view = " http://schemas.android.com/apk/res-auto "

Voici un exemple:

```
<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>
```

Vous pouvez aussi le faire par programmation en utilisant:

```
card.setCardBackgroundColor(....);
card.setCardElevation(...);
card.setRadius(....);
card.setContentPadding();
```

Vérifiez le javadoc officiel pour des propriétés supplémentaires.

Ajout d'animation Ripple

Pour activer l'animation d'ondulation dans CardView, ajoutez les attributs suivants:

```
<android.support.v7.widget.CardView
  ...
  android:clickable="true"
  android:foreground="?android:attr/selectableItemBackground">
  ...
```

```
</android.support.v7.widget.CardView>
```

Utilisation d'images en tant qu'arrière-plan dans CardView (problèmes liés aux périphériques pré-Lollipop)

Lorsque vous utilisez Image / Couleur comme arrière-plan dans une vue CardView, vous risquez d'obtenir de légers bourrelets blancs (si la couleur par défaut de la carte est le blanc) sur les bords. Cela est dû aux coins arrondis par défaut dans la vue Carte. Voici comment éviter ces marges dans les dispositifs de pré-sucette.

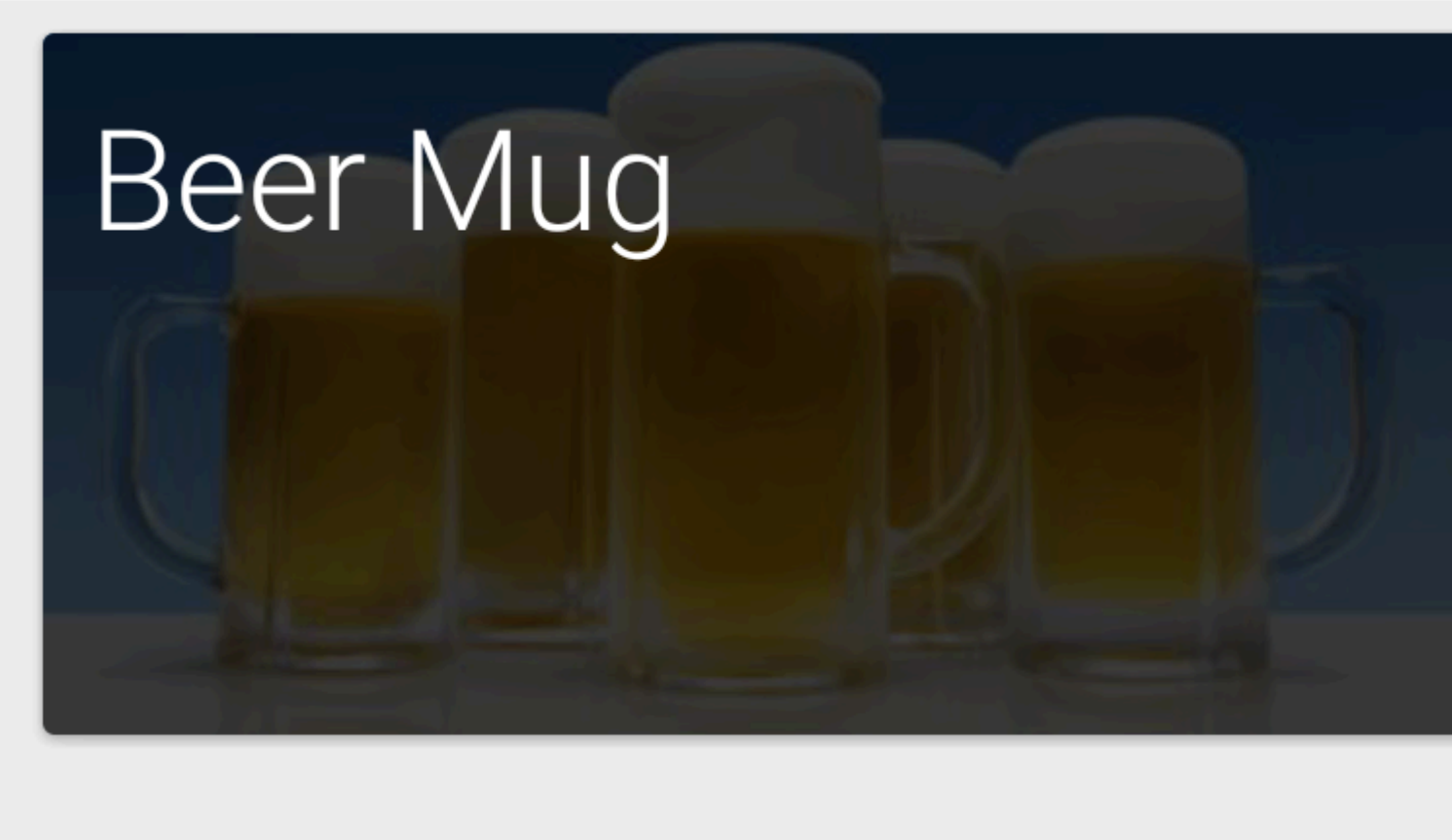
Nous devons utiliser un attribut `card_view:cardPreventCornerOverlap="false"` dans CardView. 1). En XML, utilisez l'extrait de code suivant.

```
<android.support.v7.widget.CardView
    xmlns:card_view="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    card_view:cardPreventCornerOverlap="false"
    android:layout_height="wrap_content">
      <ImageView
            android:id="@+id/row_wallet_redeem_img"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:adjustViewBounds="true"
            android:scaleType="centerCrop"
            android:src="@drawable/bg_image" />

</android.support.v7.widget.CardView>
```

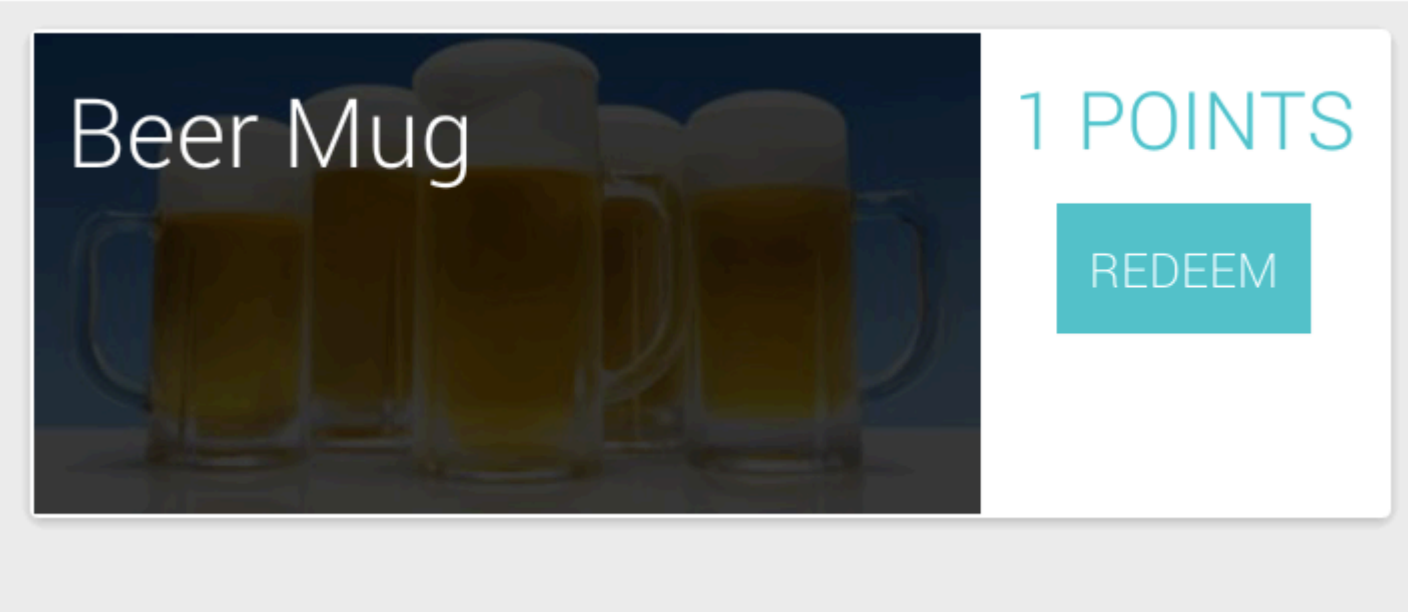
2. En Java comme ceci `cardView.setPreventCornerOverlap(false)` .

Cela supprime un bourrage indésirable sur les bords de la carte. Voici quelques exemples visuels liés à cette implémentation.

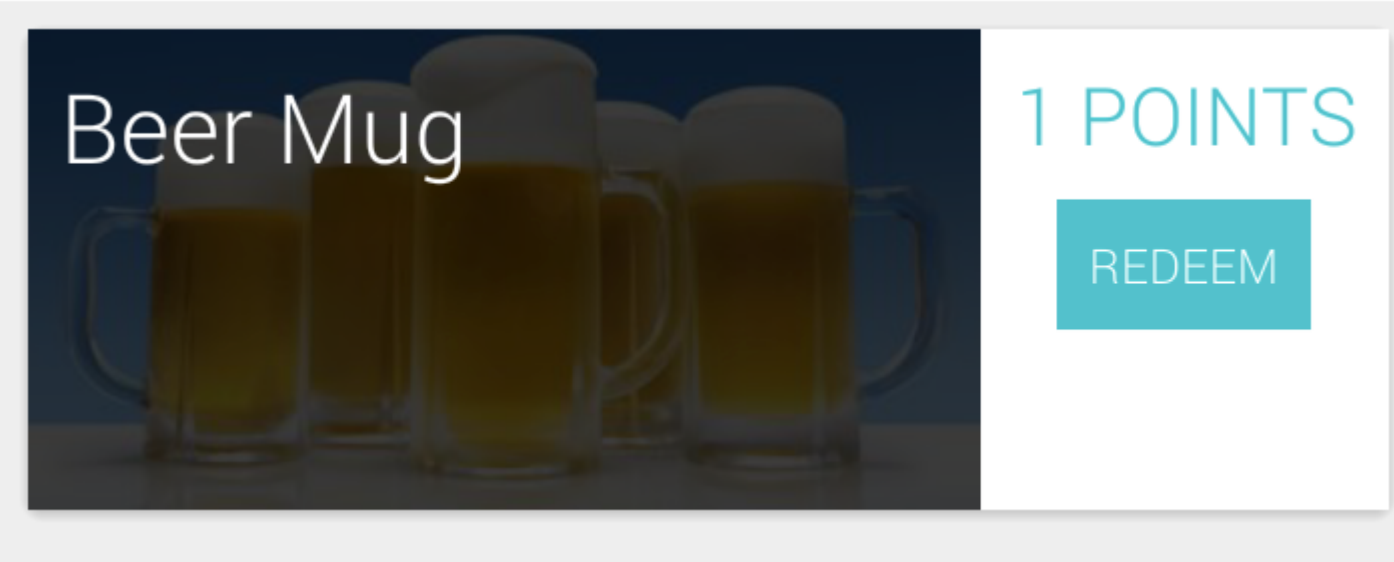
1 carte avec fond d'image dans API 21 (parfaitement bien)



2 carte avec fond d'image dans l'API 19 sans attribut (notez les rembourrages autour de l'image)



3 carte FIXED avec arrière-plan d'image dans l'API 19 avec l'attribut `cardView.setPreventCornerOverlap(false)` (problème désormais résolu)



Lisez aussi à ce sujet sur Documentation ici
Message SOF original ici

Animer la couleur d'arrière-plan de CardView avec 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);
}
```

Lire CardView en ligne: https://riptutorial.com/fr/android/topic/726/cardview

Chapitre 48: Carte à puce

Examples

Carte à puce envoyer et recevoir

Pour la connexion, voici un extrait pour vous aider à comprendre:

```
//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);
}
```

Maintenant, vous devez comprendre que dans Java, la communication a lieu en utilisant le package javax.smarcard qui n'est pas disponible pour Android, alors jetez un oeil ici pour avoir une idée de la façon de communiquer ou envoyer / recevoir APDU (commande smartcard).

Maintenant, comme indiqué dans la réponse mentionnée ci-dessus

Vous ne pouvez pas simplement envoyer une commande APDU (commande de carte à puce) sur le point de terminaison de la sortie groupée et espérer recevoir une unité APDU de réponse sur le point de terminaison en bloc. Pour obtenir les points de terminaison, consultez l'extrait de code ci-dessous:

```
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;
                }
```

```
            }
        }
    }
```

Maintenant, vous avez les points de terminaison en entrée et en sortie pour envoyer et recevoir des blocs de réponse APDU et de commande APDU:

Pour envoyer des commandes, consultez l'extrait de code ci-dessous:

```
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));
    }
}
```

Et pour recevoir / lire une réponse, voir l'extrait de code ci-dessous:

```
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;
}
```

Maintenant, si vous voyez cette réponse ici, la 1ère commande à envoyer est la suivante:

PC_to_RDR_IccPowerOn commande pour activer la carte.

que vous pouvez créer en lisant la section 6.1.1 du document Spécifications des classes de périphérique USB ici.

Prenons maintenant un exemple de cette commande comme celui-ci: `62000000000000000000`
Comment vous pouvez envoyer ceci:

```
write(connection, epOut, "62000000000000000000");
```

Maintenant, après avoir envoyé avec succès la commande APDU, vous pouvez lire la réponse en utilisant:

```
read(connection, epIn);
```

Et recevoir quelque chose comme

```
80 18000000 00 00 00 00 00 3BBF11008131FE45455041000000000000000000000000F1
```

Maintenant, la réponse reçue dans le code ici sera dans la variable `result` de la méthode `read()` du code

Lire Carte à puce en ligne: https://riptutorial.com/fr/android/topic/10945/carte-a-puce

Chapitre 49: Chaînes de formatage

Examples

Formater une ressource de chaîne

Vous pouvez ajouter des caractères génériques dans les ressources de chaîne et les remplir lors de l'exécution:

1. Modifier strings.xml

```
<string name="my_string">This is %1$s</string>
```

2. Format de chaîne si nécessaire

```
String fun = "fun";
context.getString(R.string.my_string, fun);
```

Mettre en forme un horodatage en chaîne

Pour une description complète des patterns, voir la référence SimpleDateFormat

```
Date now = new Date();
long timestamp = now.getTime();
SimpleDateFormat sdf = new SimpleDateFormat("MM/dd/yyyy", Locale.US);
String dateStr = sdf.format(timestamp);
```

Formatage des types de données en chaîne et vice versa

Types de données à mettre en forme de chaîne

Les types de données comme int, float, double, long, boolean peuvent être formatés en chaîne à l'aide de String.valueOf ().

```
String.valueOf(1); //Output -> "1"
String.valueOf(1.0); //Output -> "1.0"
String.valueOf(1.2345); //Output -> "1.2345"
String.valueOf(true); //Output -> "true"
```

À l'inverse de cela, formater une chaîne à un autre type de données

```
Integer.parseInt("1"); //Output -> 1
Float.parseFloat("1.2"); //Output -> 1.2
Boolean.parseBoolean("true"); //Output -> true
```

Lire Chaînes de formatage en ligne: https://riptutorial.com/fr/android/topic/1346/chaines-de-formatage

Chapitre 50: Changements d'orientation

Remarques

Référence: https://guides.codepath.com/android/Handling-Configuration-Changes#references

Examples

Enregistrement et restauration de l'état d'activité

À mesure que votre activité commence à s'arrêter, le système appelle `onSaveInstanceState()` afin que votre activité puisse enregistrer les informations d'état avec une collection de paires clé-valeur. L'implémentation par défaut de cette méthode enregistre automatiquement les informations sur l'état de la hiérarchie de vues de l'activité, telles que le texte d'un widget `EditText` ou la position de défilement d'un objet `ListView` .

Pour enregistrer des informations d'état supplémentaires pour votre activité, vous devez implémenter `onSaveInstanceState()` et ajouter des paires clé-valeur à l'objet Bundle. Par exemple:

```
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);
    }
}
```

Le système appellera cette méthode avant qu'une activité ne soit détruite. Ensuite, le système appellera `onRestoreInstanceState` où nous pouvons restaurer l'état du bundle:

```
@Override
protected void onRestoreInstanceState(Bundle savedInstanceState) {
    // Always call the superclass so it can restore the view hierarchy
    super.onRestoreInstanceState(savedInstanceState);
    // Restore state members from saved instance
    someIntValue = savedInstanceState.getInt(SOME_VALUE);
    someStringValue = savedInstanceState.getString(SOME_OTHER_VALUE);
}
```

L'état de l'instance peut également être restauré dans la méthode standard Activity # onCreate, mais il est pratique de le faire dans `onRestoreInstanceState` qui garantit que toutes les initialisations ont été effectuées et permet aux sous-classes de décider si elles doivent ou non être implémentées. Lisez ce post stackoverflow pour plus de détails.

Notez que `onSaveInstanceState` et `onRestoreInstanceState` ne sont pas garantis pour être appelés ensemble. Android appelle `onSaveInstanceState()` lorsqu'il y a une chance que l'activité soit détruite. Cependant, il existe des cas où `onSaveInstanceState` est appelé mais l'activité n'est pas détruite et par conséquent `onRestoreInstanceState` n'est pas `onRestoreInstanceState` .

Enregistrement et restauration de l'état des fragments

Les fragments ont également une méthode `onSaveInstanceState()` appelée lorsque leur état doit être enregistré:

```
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);
    }
}
```

Ensuite, nous pouvons extraire des données de cet état enregistré dans `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;
   }
}
```

Pour que l'état de fragment soit correctement enregistré, nous devons nous assurer que nous ne recréons pas inutilement le fragment lors des modifications de configuration. Cela signifie faire attention à ne pas réinitialiser les fragments existants lorsqu'ils existent déjà. Tout fragment initialisé dans une activité doit être recherché par balise après un changement de configuration:

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

Cela nous oblige à faire attention à inclure une balise pour la recherche chaque fois que l'on met un fragment dans l'activité dans une transaction:

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

Avec ce modèle simple, nous pouvons réutiliser correctement les fragments et restaurer leur état dans les modifications de configuration.

Conservation des fragments

Dans de nombreux cas, nous pouvons éviter les problèmes lorsqu'une activité est recréée en utilisant simplement des fragments. Si vos vues et votre état se trouvent dans un fragment, vous pouvez facilement conserver le fragment lorsque l'activité est recréée:

```
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;
    }
```

```
}
```

Cette approche empêche le fragment d'être détruit pendant le cycle de vie de l'activité. Ils sont conservés dans le gestionnaire de fragments. Voir les documents officiels Android pour plus d' informations .

Maintenant, vous pouvez vérifier si le fragment existe déjà par tag avant d'en créer un et le fragment conservera son état dans toutes les modifications de configuration. Voir le guide Handling Runtime Changes pour plus de détails .

Orientation de l'écran de verrouillage

Si vous souhaitez verrouiller le changement d'orientation de l'écran (activité) de votre application Android, il vous suffit de définir la propriété `android:screenOrientation` d'une `<activity>` dans le **fichier AndroidManifest.xml** :

```
<activity
    android:name="com.techblogon.screenorientationexample.MainActivity"
    android:screenOrientation="portrait"
    android:label="@string/app_name" >
    <!-- ... -->
</activity>
```

Maintenant, cette activité est forcée à toujours être affichée en mode " **portrait** ".

Gestion manuelle des modifications de configuration

Si votre application n'a pas besoin de mettre à jour les ressources lors d'un changement de configuration spécifique et que vous avez une limitation de performance qui vous oblige à éviter le redémarrage de l'activité, vous pouvez déclarer que votre activité gère elle-même le changement de configuration. activité.

Cependant, cette technique doit être considérée comme un dernier recours lorsque vous devez éviter les redémarrages en raison d'un changement de configuration et que cela n'est pas recommandé pour la plupart des applications. Pour adopter cette approche, nous devons ajouter le nœud `android:configChanges` à l'activité dans le **fichier AndroidManifest.xml** :

```
<activity android:name=".MyActivity"
          android:configChanges="orientation|screenSize|keyboardHidden"
          android:label="@string/app_name">
```

Maintenant, lorsque l'une de ces configurations change, l'activité ne redémarre pas mais reçoit un appel à `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();
    }
}
```

Voir la documentation sur la gestion du changement . Pour plus d'informations sur les modifications de configuration que vous pouvez gérer dans votre activité, consultez la documentation android: configChanges et la classe Configuration .

Gestion de la tâche asynchrone

Problème:

- Si, après le démarrage de `AsyncTask` il y a une rotation de l'écran, l'activité propriétaire est détruite et recréée.
- Lorsque `AsyncTask` termine, il souhaite mettre à jour l'interface utilisateur qui pourrait ne plus être valide.

Solution:

En utilisant des chargeurs , on peut facilement surmonter l'activité de destruction / récréation.

Exemple:

Activité principale:

```
public class MainActivity extends AppCompatActivity
        implements LoaderManager.LoaderCallbacks<Bitmap> {

    //Unique id for the loader
    private static final int MY_LOADER = 0;

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

        LoaderManager loaderManager = getSupportLoaderManager();

        if(loaderManager.getLoader(MY_LOADER) == null) {
            loaderManager.initLoader(MY_LOADER, null, this).forceLoad();
        }
    }

    @Override
```

```
    public Loader<Bitmap> onCreateLoader(int id, Bundle args) {
        //Create a new instance of your Loader<Bitmap>
        MyLoader loader = new MyLoader(MainActivity.this);
        return loader;
    }

    @Override
    public void onLoadFinished(Loader<Bitmap> loader, Bitmap data) {
        // do something in the parent activity/service
        // i.e. display the downloaded image
        Log.d("MyAsyncTask", "Received result: ");
    }

    @Override
    public void onLoaderReset(Loader<Bitmap> loader) {

    }
}
```

AsyncTaskLoader:

```
public class MyLoader extends AsyncTaskLoader<Bitmap> {
    private WeakReference<Activity> motherActivity;

    public MyLoader(Activity activity) {
        super(activity);
        //We don't use this, but if you want you can use it, but remember, WeakReference
        motherActivity = new WeakReference<>(activity);
    }

    @Override
    public Bitmap loadInBackground() {
        // Do work. I.e download an image from internet to be displayed in gui.
        // i.e. return the downloaded gui
        return result;
    }
}
```

Remarque:

Il est important d'utiliser la bibliothèque de compatibilité v4 ou non, mais n'utilisez pas une partie de l'une ou l'autre partie, car cela entraînerait des erreurs de compilation. Pour vérifier, vous pouvez consulter les importations pour `android.support.v4.content` et `android.content` (vous ne devriez pas avoir les deux).

Verrouiller la rotation de l'écran par programmation

Il est très courant que pendant le développement, on puisse trouver **très utile de verrouiller / déverrouiller l'écran de l'appareil pendant certaines parties du code** .

*Par exemple, tout en affichant une boîte de dialogue contenant des informations, le développeur peut vouloir **verrouiller** la rotation de l'écran pour empêcher le rejet de la boîte de dialogue et la reconstruction de l'activité en cours pour la **déverrouiller** .*

Même si nous pouvons réaliser un verrouillage de rotation à partir du manifeste en effectuant:

```
<activity
    android:name=".TheActivity"
    android:screenOrientation="portrait"
    android:label="@string/app_name" >
</activity>
```

On peut le faire aussi par programmation en procédant comme suit:

```
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);
        }
    }
}
```

Et puis appelant le suivant, pour verrouiller et déverrouiller respectivement la rotation de l'appareil

```
lockDeviceRotation(true)
```

et

```
lockDeviceRotation(false)
```

Lire Changements d'orientation en ligne: https://riptutorial.com/fr/android/topic/4621/changements-d-orientation

Chapitre 51: Chargement efficace de bitmaps

Introduction

Cette rubrique se concentre principalement sur le chargement efficace des images bitmap dans les appareils Android.

Lorsqu'il s'agit de charger un bitmap, la question vient d'où il est chargé. Ici, nous allons discuter de la façon de charger le bitmap à partir de ressources dans le périphérique Android. c'est-à-dire par exemple de la galerie.

Nous allons passer en revue ceci par exemple qui sont discutés ci-dessous.

Syntaxe

- `<uses-permission>` **->** Balise utilisée pour l'autorisation.
- `android:name` **->** Un attribut utilisé pour donner le nom de la permission que nous allons demander.
- `android.permission.READ_EXTERNAL_STORAGE` **->** Il s'agit d'autorisations système
- exemple "android.permission.CAMERA" ou "android.permission.READ_CONTACTS"

Examples

Chargez l'image à partir d'une ressource à partir d'un périphérique Android. Utiliser les intentions.

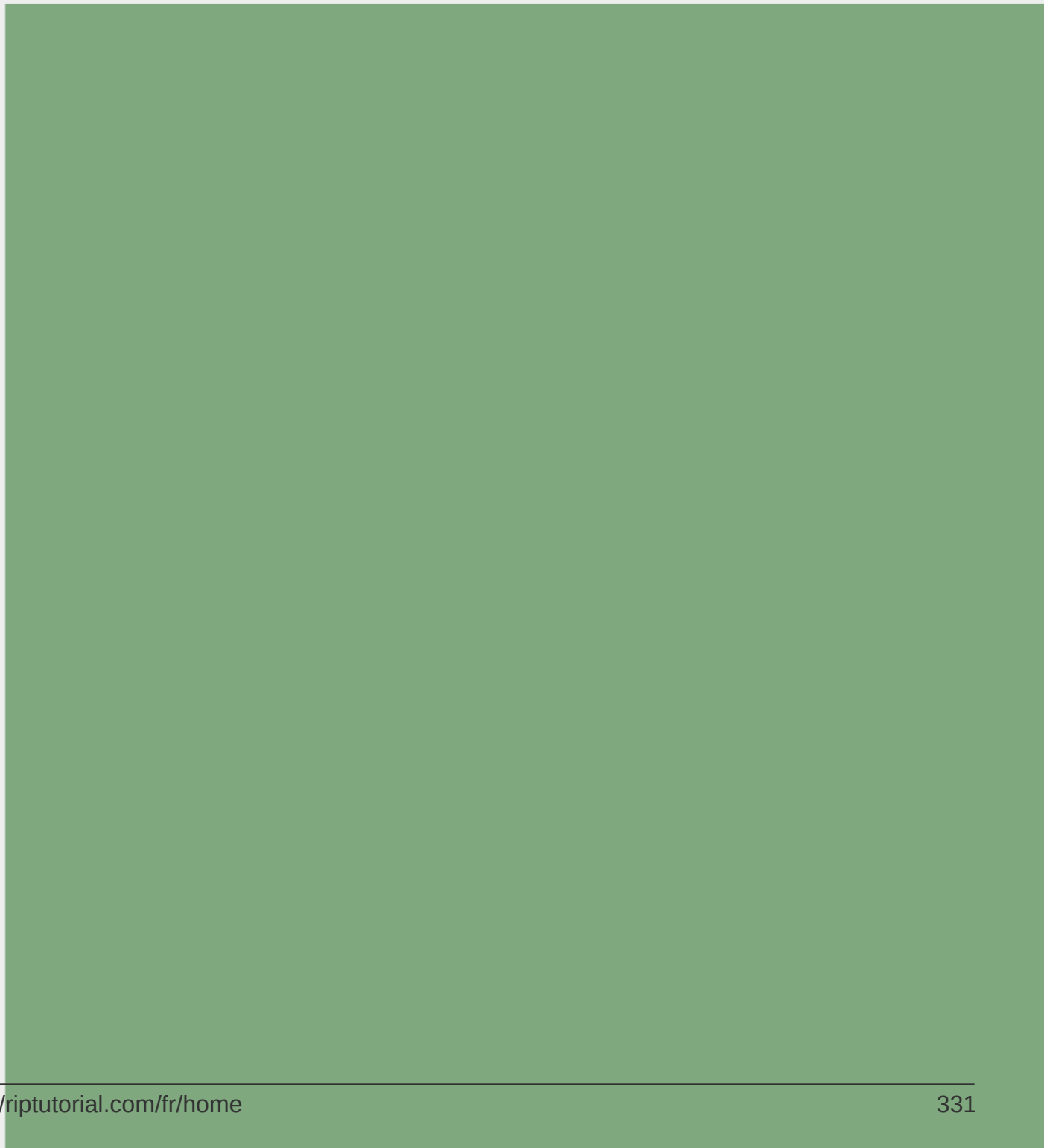
Utilisation des intentions pour charger l'image depuis la galerie.

1. Au départ, vous devez avoir la permission

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

2. Utilisez le code suivant pour que la mise en page soit conçue comme suit.

LoadImageFrmGallery



https://riptutorial.com/fr/android/topic/10902/chargement-efficace-de-bitmaps

Chapitre 52: Chargeur

Introduction

Loader est un bon choix pour éviter les fuites de mémoire si vous souhaitez charger des données en arrière-plan lorsque la méthode oncreate est appelée. Par exemple, lorsque nous exécutons Asynctask dans la méthode oncreate et que nous faisons pivoter l'écran pour que l'activité recrée exécute un autre AsyncTask à nouveau, deux Asyntask s'exécutant probablement en parallèle plutôt qu'un chargeur qui poursuivra le processus d'arrière-plan que nous avons exécuté auparavant.

Paramètres

Classe	La description
LoaderManager	Classe abstraite associée à une activité ou à un fragment pour gérer une ou plusieurs instances du chargeur.
LoaderManager.LoaderCallbacks	Une interface de rappel permettant à un client d'interagir avec le LoaderManager.
Chargeur	Une classe abstraite qui effectue un chargement asynchrone des données.
AsyncTaskLoader	Chargeur abstrait qui fournit une tâche AsyncTask pour effectuer le travail.
CursorLoader	Une sous-classe de AsyncTaskLoader qui interroge le ContentResolver et renvoie un curseur.

Remarques

Introduits dans Android 3.0, les chargeurs facilitent le chargement asynchrone de données dans une activité ou un fragment. Les chargeurs ont ces caractéristiques:

- Ils sont disponibles pour chaque activité et fragment .
- Ils fournissent un chargement asynchrone des données.
- Ils surveillent la source de leurs données et fournissent de nouveaux résultats lorsque le contenu change.
- Ils se reconnectent automatiquement au curseur du dernier chargeur lorsqu'ils sont recréés après un changement de configuration. Ainsi, ils n'ont pas besoin de ré-interroger leurs données.

Quand ne pas utiliser les chargeurs

Vous ne devez pas utiliser Loaders si vous avez besoin des tâches d'arrière-plan à effectuer. Android détruit les chargeurs avec les activités / fragments auxquels ils appartiennent. Si vous souhaitez effectuer certaines tâches, qui doivent être exécutées jusqu'à la fin, n'utilisez pas de chargeurs. Vous devriez utiliser les services pour ce genre de choses à la place.

Examples

AsyncTaskLoader de base

`AsyncTaskLoader` est un `Loader` abstrait qui fournit une `AsyncTask` pour effectuer le travail.

Voici quelques implémentations de base:

```
final class BasicLoader extends AsyncTaskLoader<String> {

    public BasicLoader(Context context) {
        super(context);
    }

    @Override
    public String loadInBackground() {
        // Some work, e.g. load something from internet
        return "OK";
    }

    @Override
    public void deliverResult(String data) {
        if (isStarted()) {
            // Deliver result if loader is currently started
            super.deliverResult(data);
        }
    }

    @Override
    protected void onStartLoading() {
        // Start loading
        forceLoad();
    }

    @Override
    protected void onStopLoading() {
        cancelLoad();
    }

    @Override
    protected void onReset() {
        super.onReset();

        // Ensure the loader is stopped
        onStopLoading();
    }
}
```

En règle générale, `Loader` est initialisé dans la méthode `onCreate()` l'activité ou dans le `onActivityCreated()` du fragment. En outre, l'activité ou le fragment implémente `LoaderManager.LoaderCallbacks` interface `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) {
    }
}
```

Dans cet exemple, lorsque le chargement est terminé, le toast avec le résultat sera affiché.

AsyncTaskLoader avec cache

Il est recommandé de mettre en cache les résultats chargés pour éviter le chargement multiple des mêmes données.
Pour invalider le cache `onContentChanged()` doit être appelé. Si loader a déjà été démarré, `forceLoad()` sera appelé, sinon (si loader à l'état arrêté), loader pourra comprendre le changement de contenu avec la `takeContentChanged()` de `takeContentChanged()` .

Remarque: `onContentChanged()` doit être appelé à partir du thread principal du processus.

Javadocs dit à propos de takeContentChanged ():

> Prenez l'indicateur actuel indiquant si le contenu du chargeur a été modifié pendant son arrêt. Si c'était le cas, true est renvoyé et le drapeau est effacé.

```
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
    }
}
```

Rechargement

Pour invalider vos anciennes données et redémarrer le chargeur existant, vous pouvez utiliser la méthode `restartLoader()` :

```
private void reload() {
    getLoaderManager().reastartLoader(LOADER_ID, Bundle.EMPTY, this);
}
```

Passer les paramètres à l'aide d'un ensemble

Vous pouvez passer des paramètres par Bundle:

```
Bundle myBundle = new Bundle();
myBundle.putString(MY_KEY, myValue);
```

Obtenez la valeur dans onCreateLoader:

```
@Override
public Loader<String> onCreateLoader(int id, final Bundle args) {
    final String myParam = args.getString(MY_KEY);
    ...
}
```

Lire Chargeur en ligne: https://riptutorial.com/fr/android/topic/4390/chargeur

Chapitre 53: Chargeur d'image universel

Remarques

Exemples d'URI acceptables:

```
"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

Initialiser Universal Image Loader

1. Ajoutez la dépendance suivante au fichier *build.gradle* :

   ```
   compile 'com.nostra13.universalimageloader:universal-image-loader:1.9.5'
   ```

2. Ajoutez les autorisations suivantes au fichier *AndroidManifest.xml* :

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

3. Initialisez Universal Image Loader. Cela doit être fait avant la première utilisation:

   ```
   ImageLoaderConfiguration config = new ImageLoaderConfiguration.Builder(this)
       // ...
       .build();
   ImageLoader.getInstance().init(config);
   ```

   Les options de configuration complètes peuvent être trouvées ici .

Utilisation de base

1. Chargez une image, décodez-la dans un bitmap et affichez le bitmap dans `ImageView` (ou toute autre vue qui implémente l'interface `ImageAware`):

   ```
   ImageLoader.getInstance().displayImage(imageUri, imageView);
   ```

2. Chargez une image, décodez-la dans un bitmap et renvoyez le bitmap à un rappel:

   ```
   ImageLoader.getInstance().loadImage(imageUri, new SimpleImageLoadingListener() {
   ```

```
    @Override
    public void onLoadingComplete(String imageUri, View view, Bitmap loadedImage) {
        // Do whatever you want with the bitmap.
    }
});
```

3. Chargez une image, décodez-la dans un bitmap et retournez le bitmap de manière synchrone:

```
Bitmap bmp = ImageLoader.getInstance().loadImageSync(imageUri);
```

Lire Chargeur d'image universel en ligne: https://riptutorial.com/fr/android/topic/2760/chargeur-d-image-universel

Chapitre 54: Chiffrement / déchiffrement des données

Introduction

Cette rubrique explique le fonctionnement du chiffrement et du déchiffrement sous Android.

Examples

Cryptage AES des données en utilisant un mot de passe de manière sécurisée

L'exemple suivant chiffre un bloc de données donné à l'aide d' AES . La clé de cryptage est dérivée de manière sécurisée (sel aléatoire, 1000 cycles de SHA-256). Le chiffrement utilise AES en mode CBC avec IV aléatoire.

Notez que les données stockées dans la classe `EncryptedData` (`salt` , `iv` et `encryptedData`) peuvent être concaténées en un tableau à un seul octet. Vous pouvez ensuite enregistrer les données ou les transmettre au destinataire.

```
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;
}
```

L'exemple de code suivant montre comment tester le chiffrement et le déchiffrement:

```
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();
}
```

Lire Chiffrement / déchiffrement des données en ligne:
https://riptutorial.com/fr/android/topic/3471/chiffrement---dechiffrement-des-donnees

Chapitre 55: Choses Android

Examples

Contrôle d'un servomoteur

Cet exemple suppose que vous avez un servo avec les caractéristiques suivantes, qui sont typiques:

- mouvement entre 0 et 180 degrés
- période d'impulsion de 20 ms
- longueur d'impulsion minimale de 0,5 ms
- longueur d'impulsion maximale de 2,5 ms

Vous devez vérifier si ces valeurs correspondent à votre matériel, car le forcer à sortir de sa plage de fonctionnement spécifiée peut endommager le servo. Un servo endommagé peut à son tour endommager votre appareil Android. L'exemple de classe `ServoController` comprend deux méthodes, `setup()` et `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();
```

```
        }
    }
}
```

Vous pouvez découvrir les noms de pin qui prennent en charge PWM sur votre appareil comme suit:

```
PeripheralManagerService service = new PeripheralManagerService();

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

Afin de rendre votre servo oscillant entre 80 et 100 degrés, vous pouvez simplement utiliser le code suivant:

```
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();
```

Vous pouvez compiler et déployer tout le code ci-dessus sans pour autant associer des servomoteurs au périphérique informatique. Pour le câblage, reportez-vous au tableau de brochage de votre ordinateur (par exemple, un tableau de brochage Raspberry Pi 3 est disponible ici).

Ensuite, vous devez connecter votre servo à Vcc, Gnd et signal.

Lire Choses Android en ligne: https://riptutorial.com/fr/android/topic/8938/choses-android

Chapitre 56: Clavier

Examples

Masquer le clavier lorsque l'utilisateur touche n'importe où ailleurs sur l'écran

Ajoutez du code dans votre **activité** .

Cela fonctionnerait aussi pour **Fragment** , **pas besoin** d'ajouter ce code dans **Fragment** .

```
@Override
public boolean dispatchTouchEvent(MotionEvent ev) {
    View view = getCurrentFocus();
    if (view != null && (ev.getAction() == MotionEvent.ACTION_UP || ev.getAction() ==
MotionEvent.ACTION_MOVE) && view instanceof EditText &&
!view.getClass().getName().startsWith("android.webkit.")) {
        int scrcoords[] = new int[2];
        view.getLocationOnScreen(scrcoords);
        float x = ev.getRawX() + view.getLeft() - scrcoords[0];
        float y = ev.getRawY() + view.getTop() - scrcoords[1];
        if (x < view.getLeft() || x > view.getRight() || y < view.getTop() || y >
view.getBottom())

((InputMethodManager)this.getSystemService(Context.INPUT_METHOD_SERVICE)).hideSoftInputFromWindow((this
0);
    }
    return super.dispatchTouchEvent(ev);
}
```

Enregistrer un rappel pour clavier ouvert et fermé

L'idée est de mesurer une mise en page avant et après chaque modification et s'il y a un changement significatif, vous pouvez être certain que c'est le clavier.

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

puis dans notre `onCreate` définir la valeur initiale pour `mLastContentHeight`

```
mLastContentHeight = findViewById(Window.ID_ANDROID_CONTENT).getHeight();
```

et ajouter l'auditeur

```
rootView.getViewTreeObserver().addOnGlobalLayoutListener(keyboardLayoutListener);
```

n'oubliez pas de retirer l'auditeur de `destroy`

```
rootView.getViewTreeObserver().removeOnGlobalLayoutListener(keyboardLayoutListener);
```

Lire Clavier en ligne: https://riptutorial.com/fr/android/topic/5606/clavier

Chapitre 57: CleverTap

Introduction

Hackings rapides pour le SDK d'analyse et d'engagement fourni par CleverTap - Android

Remarques

Obtenez vos informations d'identification CleverTap à partir de https://clevertap.com .

Examples

Obtenir une instance du SDK pour enregistrer des événements

```
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
}
```

Définition du niveau de débogage

Dans votre classe d'application personnalisée, remplacez la méthode `onCreate()` , ajoutez la ligne ci-dessous:

```
CleverTapAPI.setDebugLevel(1);
```

Lire CleverTap en ligne: https://riptutorial.com/fr/android/topic/9337/clevertap

Chapitre 58: Code à barres et lecture du code QR

Remarques

QRCodeReaderView

Zxing

Examples

Utiliser QRCodeReaderView (basé sur Zxing)

QRCodeReaderView implémente une vue Android qui montre la caméra et notifie quand il y a un code QR dans l'aperçu.

Il utilise la bibliothèque de traitement d'images à codes-barres 1D / 2D multi-formats et open source zxing .

Ajout de la bibliothèque à votre projet

Ajouter une dépendance QRCodeReaderView à votre build.gradle

```
dependencies{
    compile 'com.dlazaro66.qrcodereaderview:qrcodereaderview:2.0.0'
}
```

Première utilisation

- Ajouter à votre mise en page un `QRCodeReaderView`

```
<com.dlazaro66.qrcodereaderview.QRCodeReaderView
       android:id="@+id/qrdecoderview"
       android:layout_width="match_parent"
       android:layout_height="match_parent" />
```

- Créez une activité qui implémente `onQRCodeReadListener` et utilisez-la comme écouteur de `QrCodeReaderView` .
- Assurez-vous d'avoir des droits d'accès à la caméra pour pouvoir utiliser la bibliothèque. (https://developer.android.com/training/permissions/requesting.html)

Ensuite, dans votre activité, vous pouvez l'utiliser comme suit:

```
    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();
    }
}
```

Lire Code à barres et lecture du code QR en ligne:
https://riptutorial.com/fr/android/topic/6067/code-a-barres-et-lecture-du-code-qr

Chapitre 59: Comment stocker les mots de passe de manière sécurisée

Examples

Utilisation d'AES pour le chiffrement de mot de passe salé

Cet exemple utilise l'algorithme AES pour chiffrer les mots de passe. La longueur du sel peut atteindre 128 bits.

Nous utilisons la classe `SecureRandom` pour générer un sel, qui est combiné avec le mot de passe pour générer une clé secrète. Les classes utilisées existent déjà dans les packages Android `javax.crypto` et `java.security` .

Une fois qu'une clé est générée, nous devons conserver cette clé dans une variable ou la stocker. Nous le `S_KEY` parmi les préférences partagées dans la valeur `S_KEY` . Ensuite, un mot de passe est chiffré à l'aide de la méthode `doFinal` de la classe `Cipher` une fois qu'il est initialisé dans `ENCRYPT_MODE` . Ensuite, le mot de passe crypté est converti d'un tableau d'octets en chaîne et stocké parmi les préférences partagées. La clé utilisée pour générer un mot de passe chiffré peut être utilisée pour déchiffrer le mot de passe de la même manière:

```
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;
    }
```

```
}
```

Lire Comment stocker les mots de passe de manière sécurisée en ligne: https://riptutorial.com/fr/android/topic/9093/comment-stocker-les-mots-de-passe-de-maniere-securisee

Chapitre 60: Comment utiliser SparseArray

Introduction

Un `SparseArray` est une alternative à une `Map` . Une `Map` nécessite que ses clés soient des objets. Le phénomène de la sélection automatique se produit lorsque nous voulons utiliser une valeur primitive `int` comme clé. Le compilateur convertit automatiquement les valeurs primitives en leurs types encadrés (par exemple `int` à `Integer`). La différence de mémoire est notable: `int` utilise 4 octets, `Integer` utilise 16 octets. Un `SparseArray` utilise `int` comme valeur de clé.

Remarques

Avantage :

- Moins d'utilisation de la mémoire (à cause des clés primitives).
- Pas de boxe automatique.

Désavantage :

- SparseArray utilise la recherche binaire pour trouver la valeur (O (log n)), donc ce n'est peut-être pas la meilleure solution si vous devez travailler avec un grand nombre d'éléments (utilisez HashMap).

Il existe plusieurs variantes de la famille comme: -SparseArray <Integer, Object> - SparseBooleanArray <Entier, Booléen> -SparseIntArray <Entier, Entier> -SparseLongArray <Entier, Long> -LongSparseArray <Long, Objet> -LongSparseLongArray <Long, Long >

Opérations SparseArray

- Ajout de l'élément - put (int, x): Ajoute un mappage de la clé spécifiée à la valeur spécifiée, remplaçant le précédent mappage de la clé spécifiée s'il y en avait une. - append (int, x): place une paire clé / valeur dans le tableau, en optimisant le cas où la clé est supérieure à toutes les clés existantes du tableau. Vous devez utiliser append () en cas de clés séquentielles pour optimiser les performances. Sinon, put () va bien.

- Élément remove - delete (int): Supprime le mappage de la clé spécifiée, le cas échéant. - removeAt (int): Supprime le mappage à l'index donné. - removeAtRange (int, int): Supprime une plage de mappages en tant que lot.

- access element - get (int): Obtient l'int intéré à partir de la clé spécifiée, ou 0 si aucun mappage n'a été effectué. - get (int, E): Obtient l'int mappé depuis la clé spécifiée ou la valeur spécifiée si aucun mappage n'a été effectué. - valueAt (int): Étant donné un index compris dans la plage 0 ... size () - 1, renvoie la valeur du mappage valeur-clé indexth stocké par SparseIntArray. Les indices sont classés par ordre croissant.

- index / key search - keyAt (int): Étant donné un index compris entre 0 et size (), renvoie la

clé du mappage index-valeur indexée stocké par SparseIntArray. Les indices sont classés par ordre croissant. - valueAt (int): Étant donné un index compris dans la plage 0 ... size () - 1, renvoie la valeur du mappage valeur-clé indexth stocké par SparseIntArray. Les indices sont classés par ordre croissant. - indexOfKey (int): renvoie l'index pour lequel keyAt (int) renverrait la clé spécifiée ou un nombre négatif si la clé spécifiée n'est pas mappée. - indexOfValue (E): renvoie un index pour lequel valueAt (int) renverrait la clé spécifiée ou un nombre négatif si aucune clé ne correspond à la valeur spécifiée. Attention, il s'agit d'une recherche linéaire, contrairement aux recherches par clé, et que plusieurs clés peuvent être mappées à la même valeur et que celle-ci n'en trouvera qu'une. La différence de leur empreinte mémoire est notable: l’int utilise 4 octets, l’Integer utilise 16 octets.

Examples

Exemple basique utilisant 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);
```

Tutoriel sur YouTube

Lire Comment utiliser SparseArray en ligne: https://riptutorial.com/fr/android/topic/8824/comment-utiliser-sparsearray

Chapitre 61: Composants d'architecture Android

Introduction

Android Architecture Components est une nouvelle collection de bibliothèques qui vous permet de concevoir des applications robustes, testables et maintenables. Les parties principales sont: les cycles de vie, ViewModel, LiveData, Room.

Examples

Ajouter des composants d'architecture

Projet 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'
}
```

Application build 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"
```

Utilisation du cycle de vie dans AppCompatActivity

Prolongez votre activité de cette activité

```
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 avec transformations LiveData

```
public class BaseViewModel extends ViewModel {
    private static final int TAG_SEGMENT_INDEX = 2;
    private static final int VIDEOS_LIMIT = 100;

    // We save input params here
    private final MutableLiveData<Pair<String, String>> urlWithReferrerLiveData = new
MutableLiveData<>();

    // transform specific uri param to "tag"
    private final LiveData<String> currentTagLiveData =
Transformations.map(urlWithReferrerLiveData, pair -> {
        Uri uri = Uri.parse(pair.first);
        List<String> segments = uri.getPathSegments();
        if (segments.size() > TAG_SEGMENT_INDEX)
            return segments.get(TAG_SEGMENT_INDEX);
        return null;
    });

    // transform "tag" to videos list
    private final LiveData<List<VideoItem>> videoByTagData =
Transformations.switchMap(currentTagLiveData, tag -> contentRepository.getVideoByTag(tag,
VIDEOS_LIMIT));

    ContentRepository contentRepository;

    public BaseViewModel() {
        // some inits
    }

    public void setUrlWithReferrer(String url, String referrer) {
        // set value activates observers and transformations
        urlWithReferrerLiveData.setValue(new Pair<>(url, referrer));
    }

    public LiveData<List<VideoItem>> getVideoByTagData() {
        return videoByTagData;
    }
}
```

Quelque part dans l'interface utilisateur:

```
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);
        }
}
```

La pérennité de la chambre

Room requiert quatre parties: Classe de base de données, classes DAO, classes d'entité et classes de migration (vous ne pouvez désormais utiliser **que les méthodes DDL**):

Classes d'entités

```
// 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;
}
```

Classes DAO

```
@Dao
public interface VideoDao {
    // Create insert with custom conflict strategy
    @Insert(onConflict = OnConflictStrategy.REPLACE)
    void saveVideos(List<VideoItem> videos);

    // Simple update
    @Update
    void updateVideos(VideoItem... videos);

    @Query("DELETE FROM tags WHERE videoId = :videoId")
    void deleteTagsByVideoId(long videoId);

    // Custom query, you may use select/delete here
    @Query("SELECT v.* FROM tags t LEFT JOIN videos v ON v.articleId = t.videoId WHERE t.value
= :tag ORDER BY updatedAt DESC LIMIT :limit")
    LiveData<List<VideoItem>> getVideosByTag(String tag, int limit);
}
```

Classe de base de données

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

Migrations

```
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() {
    }
}
```

Utiliser dans la classe Application ou fournir via Dagger

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

Ecrivez votre référentiel:

```
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);
    }
}
```

Utiliser dans ViewModel:

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

LiveData personnalisé

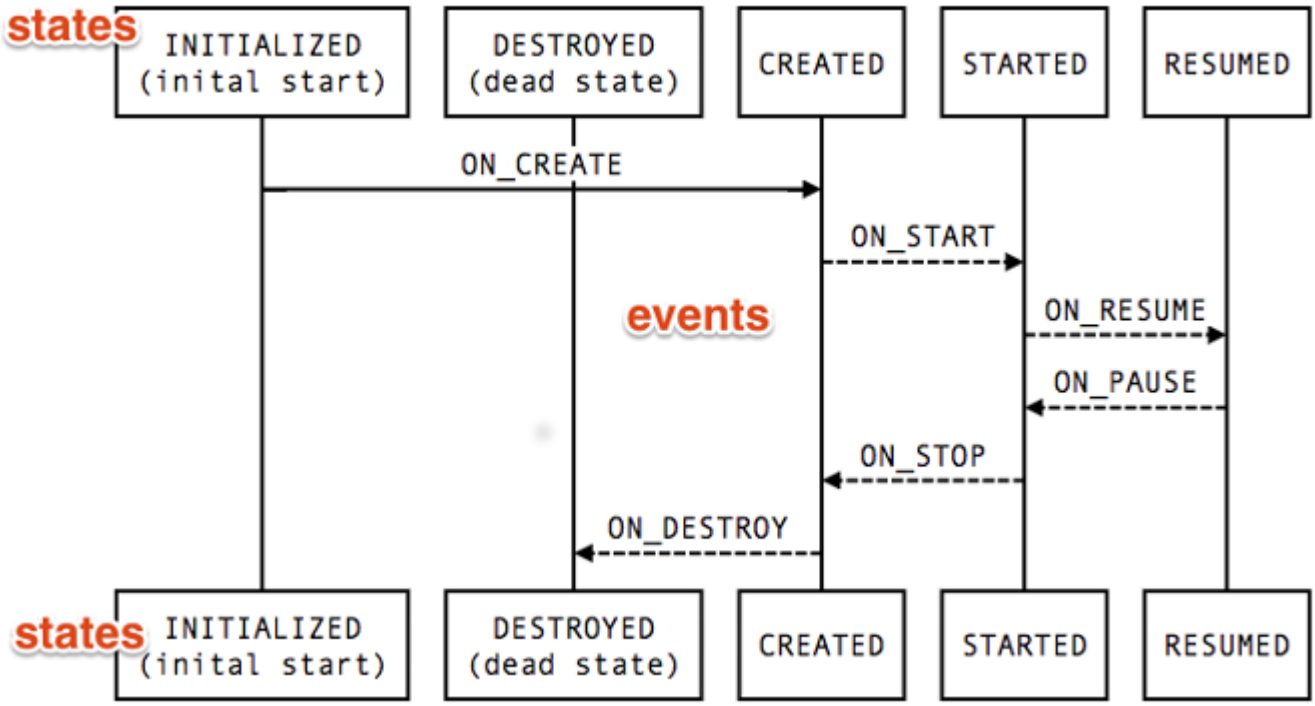
Vous pouvez écrire des données LiveData personnalisées si vous avez besoin d'une logique personnalisée.
N'écrivez pas de classe personnalisée si vous avez seulement besoin de transformer des données (utilisez la classe Transformations)

```
public class LocationLiveData extends LiveData<Location> {
    private LocationManager locationManager;

    private LocationListener listener = new LocationListener() {
        @Override
        public void onLocationChanged(Location location) {
            setValue(location);
        }

        @Override
        public void onStatusChanged(String provider, int status, Bundle extras) {
            // Do something
        }

        @Override
        public void onProviderEnabled(String provider) {
            // Do something
        }

        @Override
```

```
        public void onProviderDisabled(String provider) {
            // Do something
        }
    };

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

    @Override
    protected void onActive() {
        // We have observers, start working
        locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0, listener);
    }

    @Override
    protected void onInactive() {
        // We have no observers, stop working
        locationManager.removeUpdates(listener);
    }
}
```

Composant personnalisé prenant en compte le cycle de vie

Le cycle de vie de chaque composant de l'interface utilisateur a changé comme indiqué sur l'image.



Vous pouvez créer un composant qui sera notifié du changement d'état du cycle de vie:

```
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
    }
}
```

Lire Composants d'architecture Android en ligne:
https://riptutorial.com/fr/android/topic/10872/composants-d-architecture-android

Chapitre 62: Compression d'image

Examples

Comment compresser l'image sans changement de taille

Obtenir le **bitmap compressé** de la classe Singleton:

```
ImageView imageView = (ImageView)findViewById(R.id.imageView);
Bitmap bitmap = ImageUtils.getInstant().getCompressedBitmap("Your_Image_Path_Here");
imageView.setImageBitmap(bitmap);
```

ImageUtils.java :

```
public class ImageUtils {

    public static ImageUtils mInstant;

    public static ImageUtils getInstant(){
        if(mInstant==null){
            mInstant = new ImageUtils();
        }
        return mInstant;
    }

    public  Bitmap getCompressedBitmap(String imagePath) {
        float maxHeight = 1920.0f;
        float maxWidth = 1080.0f;
        Bitmap scaledBitmap = null;
        BitmapFactory.Options options = new BitmapFactory.Options();
        options.inJustDecodeBounds = true;
        Bitmap bmp = BitmapFactory.decodeFile(imagePath, options);

        int actualHeight = options.outHeight;
        int actualWidth = options.outWidth;
        float imgRatio = (float) actualWidth / (float) actualHeight;
        float maxRatio = maxWidth / maxHeight;

        if (actualHeight > maxHeight || actualWidth > maxWidth) {
            if (imgRatio < maxRatio) {
                imgRatio = maxHeight / actualHeight;
                actualWidth = (int) (imgRatio * actualWidth);
                actualHeight = (int) maxHeight;
            } else if (imgRatio > maxRatio) {
                imgRatio = maxWidth / actualWidth;
                actualHeight = (int) (imgRatio * actualHeight);
                actualWidth = (int) maxWidth;
            } else {
                actualHeight = (int) maxHeight;
                actualWidth = (int) maxWidth;

            }
        }

        options.inSampleSize = calculateInSampleSize(options, actualWidth, actualHeight);
```

```
        options.inJustDecodeBounds = false;
        options.inDither = false;
        options.inPurgeable = true;
        options.inInputShareable = true;
        options.inTempStorage = new byte[16 * 1024];

        try {
            bmp = BitmapFactory.decodeFile(imagePath, options);
        } catch (OutOfMemoryError exception) {
            exception.printStackTrace();

        }
        try {
            scaledBitmap = Bitmap.createBitmap(actualWidth, actualHeight,
Bitmap.Config.ARGB_8888);
        } catch (OutOfMemoryError exception) {
            exception.printStackTrace();
        }

        float ratioX = actualWidth / (float) options.outWidth;
        float ratioY = actualHeight / (float) options.outHeight;
        float middleX = actualWidth / 2.0f;
        float middleY = actualHeight / 2.0f;

        Matrix scaleMatrix = new Matrix();
        scaleMatrix.setScale(ratioX, ratioY, middleX, middleY);

        Canvas canvas = new Canvas(scaledBitmap);
        canvas.setMatrix(scaleMatrix);
        canvas.drawBitmap(bmp, middleX - bmp.getWidth() / 2, middleY - bmp.getHeight() / 2,
new Paint(Paint.FILTER_BITMAP_FLAG));

        ExifInterface exif = null;
        try {
            exif = new ExifInterface(imagePath);
            int orientation = exif.getAttributeInt(ExifInterface.TAG_ORIENTATION, 0);
            Matrix matrix = new Matrix();
            if (orientation == 6) {
                matrix.postRotate(90);
            } else if (orientation == 3) {
                matrix.postRotate(180);
            } else if (orientation == 8) {
                matrix.postRotate(270);
            }
            scaledBitmap = Bitmap.createBitmap(scaledBitmap, 0, 0, scaledBitmap.getWidth(),
scaledBitmap.getHeight(), matrix, true);
        } catch (IOException e) {
            e.printStackTrace();
        }
        ByteArrayOutputStream out = new ByteArrayOutputStream();
        scaledBitmap.compress(Bitmap.CompressFormat.JPEG, 85, out);

        byte[] byteArray = out.toByteArray();

        Bitmap updatedBitmap = BitmapFactory.decodeByteArray(byteArray, 0, byteArray.length);

        return updatedBitmap;
    }

    private int calculateInSampleSize(BitmapFactory.Options options, int reqWidth, int
reqHeight) {
```

```
        final int height = options.outHeight;
        final int width = options.outWidth;
        int inSampleSize = 1;

        if (height > reqHeight || width > reqWidth) {
            final int heightRatio = Math.round((float) height / (float) reqHeight);
            final int widthRatio = Math.round((float) width / (float) reqWidth);
            inSampleSize = heightRatio < widthRatio ? heightRatio : widthRatio;
        }
        final float totalPixels = width * height;
        final float totalReqPixelsCap = reqWidth * reqHeight * 2;

        while (totalPixels / (inSampleSize * inSampleSize) > totalReqPixelsCap) {
            inSampleSize++;
        }
        return inSampleSize;
    }
}
```

Les dimensions sont les mêmes après avoir compressé Bitmap .

Comment ai-je vérifié?

```
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());
```

Sortie:

```
Before Compress : Dimension: 1080-1452
After Compress : Dimension: 1080-1452
```

Lire Compression d'image en ligne: https://riptutorial.com/fr/android/topic/5588/compression-d-image

Chapitre 63: Compte à rebours

Paramètres

Paramètre	Détails
`long millisInFuture`	La durée totale de la minuterie, c'est-à-dire à quelle distance vous souhaitez que le minuteur se termine. En millisecondes.
`long countDownInterval`	L'intervalle auquel vous souhaitez recevoir les mises à jour du minuteur. En millisecondes.
`long millisUntilFinished`	Un paramètre fourni dans `onTick()` qui indique la durée restante de CountDownTimer. En millisecondes

Remarques

CountDownTimer est une classe plutôt simple - elle fait très bien une chose. Comme vous ne pouvez que démarrer / annuler un CountDownTimer, vous devez implémenter la fonctionnalité de pause / reprise, comme indiqué dans le deuxième exemple. Pour des fonctionnalités plus complexes ou pour spécifier une horloge qui doit s'exécuter indéfiniment, utilisez l'objet Timer .

Examples

Création d'un compte à rebours simple

CountDownTimer est utile pour effectuer plusieurs fois une action dans un intervalle régulier pendant une durée définie. Dans cet exemple, nous mettrons à jour une vue texte toutes les secondes pendant 30 secondes pour indiquer le temps restant. Ensuite, lorsque la minuterie se termine, nous allons définir le TextView pour dire "Terminé".

```
TextView textView = (TextView)findViewById(R.id.text_view);

CountDownTimer countDownTimer = new CountDownTimer(30000, 1000) {
    public void onTick(long millisUntilFinished) {
        textView.setText(String.format(Locale.getDefault(), "%d sec.", millisUntilFinished /
1000L));
    }

    public void onFinish() {
        textView.setText("Done.");
    }
}.start();
```

Un exemple plus complexe

Dans cet exemple, nous allons suspendre / reprendre le CountDownTimer basé sur le cycle de vie de l'activité.

```
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;
}
```

Lire Compte à rebours en ligne: https://riptutorial.com/fr/android/topic/6063/compte-a-rebours

Chapitre 64: Comptes et AccountManager

Examples

Comprendre les comptes / authentifications personnalisés

L'exemple suivant est une couverture de haut niveau des concepts clés et de la configuration squelettique de base: -

1. Collecte les informations d'identification de l'utilisateur (généralement à partir d'un écran de connexion que vous avez créé)
2. Authentifie les informations d'identification avec le serveur (stocke l'authentification personnalisée)
3. Stocke les informations d'identification sur le périphérique

Étendre un AbstractAccountAuthenticator *(principalement utilisé pour récupérer l'authentification et les ré-authentifier)*

```
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.
  }
}
```

Créer un service *(la structure Account Manager se connecte à AbstractAccountAuthenticator étendu via l'interface de service)*

```
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();
    }
}
```

Configuration XML de l'authentificateur *(la structure du gestionnaire de compte est requise. C'est ce que vous verrez dans Paramètres -> Comptes dans Android)*

```
<account-authenticator xmlns:android="http://schemas.android.com/apk/res/android"
    android:accountType="rename.with.your.applicationid"
    android:icon="@drawable/app_icon"
    android:label="@string/app_name"
    android:smallIcon="@drawable/app_icon" />
```

Modifications apportées à AndroidManifest.xml *(* **réunissez** *tous les concepts ci-dessus pour le rendre utilisable par programme via le AccountManager)*

```
<application
...>
    <service
        android:name=".authenticator.AccountAuthenticatorService"
        android:exported="false"
        android:process=":authentication">
```

```
        <intent-filter>
            <action android:name="android.accounts.AccountAuthenticator"/>
        </intent-filter>
        <meta-data
            android:name="android.accounts.AccountAuthenticator"
            android:resource="@xml/authenticator"/>
    </service>
</application>
```

L'exemple suivant contiendra comment utiliser cette configuration.

Lire Comptes et AccountManager en ligne: https://riptutorial.com/fr/android/topic/7003/comptes-et-accountmanager

Chapitre 65: Conception matérielle

Introduction

Material Design est un guide complet pour la conception visuelle, de mouvement et d'interaction entre plates-formes et appareils.

Remarques

Voir également le billet de blog Android original présentant la bibliothèque de support de conception

Documentation officielle

https://developer.android.com/design/material/index.html.

Directives pour la conception des matériaux

https://material.io/guidelines

Autres ressources de conception et bibliothèques

https://design.google.com/resources/

Examples

Appliquer un thème AppCompat

La bibliothèque de support AppCompat fournit des thèmes pour créer des applications avec la spécification Material Design . Un thème avec un parent de `Theme.AppCompat` est également requis pour qu'une activité étende `AppCompatActivity` .

La première étape consiste à personnaliser la palette de couleurs de votre thème pour coloriser automatiquement votre application.
Dans `res/styles.xml` votre application, vous pouvez définir:

```
<!-- 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>
```

Au lieu de `Theme.AppCompat` , qui a un arrière-plan sombre, vous pouvez également utiliser `Theme.AppCompat.Light` ou `Theme.AppCompat.Light.DarkActionBar` .

Vous pouvez personnaliser le thème avec vos propres couleurs. Les bons choix se trouvent dans le nuancier de spécifications de conception de matériau et dans la palette de matériaux . Les couleurs "500" sont de bons choix pour le primaire (bleu 500 dans cet exemple); choisissez "700" de la même teinte pour le noir; et une nuance d'une couleur différente de celle de l'accent. La couleur primaire est utilisée pour la barre d'outils de votre application et son entrée dans l'écran d'aperçu (applications récentes), la variante la plus sombre pour teinter la barre d'état et la couleur d'accentuation pour mettre en évidence certains contrôles.

Après avoir créé ce thème, appliquez-le à votre application dans `AndroidManifest.xml` et appliquez également le thème à une activité particulière. Ceci est utile pour appliquer un thème `AppTheme.NoActionBar` , qui vous permet d'implémenter des configurations de barre d'outils autres que celles par défaut.

```
<application android:theme="@style/AppTheme"
    ...>
    <activity
        android:name=".MainActivity"
        android:theme="@style/AppTheme" />
</application>
```

Vous pouvez également appliquer des thèmes à des vues individuelles à l'aide d' `android:theme` et d'un thème `ThemeOverlay` . Par exemple, avec une `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" />
```

ou un `Button` :

```
<Button
    style="@style/Widget.AppCompat.Button.Colored"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:theme="@style/MyButtonTheme"/>

<!-- res/values/themes.xml -->
<style name="MyButtonTheme" parent="ThemeOverlay.AppCompat.Light">
    <item name="colorAccent">@color/my_color</item>
</style>
```

Ajouter une barre d'outils

Une `Toolbar` est une généralisation d' `ActionBar` à utiliser dans les mises en page d'applications. Alors qu'un `ActionBar` fait traditionnellement partie du décor de fenêtre opaque d'une `Activity's` contrôlé par la structure, une `Toolbar` peut être placée à n'importe quel niveau d'imbrication arbitraire dans une hiérarchie de vues. Il peut être ajouté en procédant comme suit:

1. Assurez-vous que la dépendance suivante est ajoutée au fichier **build.gradle** de votre module (par exemple, celui de l'application) sous des dépendances:

```
compile 'com.android.support:appcompat-v7:25.3.1'
```

2. Définissez le thème de votre application sur celui qui ne possède **pas de** `ActionBar` . Pour ce faire, modifiez votre fichier **styles.xml** sous `res/values` et définissez un thème `Theme.AppCompat` .
   Dans cet exemple, nous utilisons `Theme.AppCompat.NoActionBar` tant que parent de votre `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>
```

   Vous pouvez également utiliser `Theme.AppCompat.Light.NoActionBar` ou `Theme.AppCompat.DayNight.NoActionBar` , ou tout autre thème n'ayant pas intrinsèquement un `ActionBar`

3. Ajoutez la `Toolbar` d' `Toolbar` à votre disposition d'activité:

```
<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"/>
```

   Sous la `Toolbar` d' `Toolbar` vous pouvez ajouter le reste de votre mise en page.

4. Dans votre `Activity` , définissez la `Toolbar` d' `Toolbar` comme `ActionBar` pour cette `Activity` .
   Si vous utilisez la bibliothèque *appcompat* et un `AppCompatActivity` , vous utiliseriez la méthode `setSupportActionBar()` :

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

    final Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);

    //...
}
```

Après avoir effectué les étapes ci-dessus, vous pouvez utiliser la méthode `getSupportActionBar()` pour manipuler la `Toolbar` définie comme `ActionBar` .

Par exemple, vous pouvez définir le titre comme indiqué ci-dessous:

```
getSupportActionBar().setTitle("Activity Title");
```

Par exemple, vous pouvez également définir la couleur du titre et de l'arrière-plan comme indiqué ci-dessous:

```
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)));
```

Ajouter un FloatingActionButton (FAB)

Dans la conception de matériau, un bouton d'action flottant représente l'action principale dans une activité.
Ils se distinguent par une icône encerclée flottant au-dessus de l'interface utilisateur et ont des comportements de mouvement qui incluent le morphing, le lancement et un point d'ancrage de transfert.

Assurez-vous que la dépendance suivante est ajoutée au fichier build.gradle de votre application sous les dépendances:

```
compile 'com.android.support:design:25.3.1'
```

Ajoutez maintenant le `FloatingActionButton` à votre fichier de mise en page:

```
<android.support.design.widget.FloatingActionButton
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_margin="16dp"
      android:src="@drawable/some_icon"/>
```

où l'attribut `src` référence à l'icône à utiliser pour l'action flottante.
Le résultat devrait ressembler à ceci (en supposant que votre couleur d'accentuation est Matière Rose):

Par défaut, la couleur d'arrière-plan de votre `FloatingActionButton` sera définie sur la couleur d'accent de votre thème. Notez également qu'un `FloatingActionButton` nécessite une marge pour fonctionner correctement. La marge recommandée pour le bas est `16dp` pour les téléphones et `24dp` pour les tablettes.

Voici les propriétés que vous pouvez utiliser pour personnaliser davantage le `FloatingActionButton` (en supposant que `xmlns:app="http://schemas.android.com/apk/res-auto` est déclaré comme espace de noms en haut de votre présentation):

- `app:fabSize` : peut être réglé sur `normal` ou `mini` pour basculer entre une taille normale ou une version plus petite.
- `app:rippleColor` : définit la couleur de l'effet d'ondulation de votre `FloatingActionButton` . Peut être une ressource de couleur ou une chaîne hexadécimale.
- `app:elevation` : peut être une chaîne, un entier, une valeur booléenne, une valeur de couleur, une virgule flottante, une valeur de dimension.
- `app:useCompatPadding` : Activer le remplissage de compat. Peut-être une valeur booléenne, telle que `true` ou `false` . Défini sur `true` pour utiliser le remplissage de compat sur api-21 et les versions ultérieures, afin de conserver une apparence cohérente avec les anciens niveaux d'API.

Vous pouvez trouver plus d'exemples sur FAB ici .

Boutons stylisés avec Material Design

La bibliothèque de prise en charge AppCompat définit plusieurs styles utiles pour les boutons , chacun d'entre eux `Widget.AppCompat.Button` style de base `Widget.AppCompat.Button` appliqué à tous les boutons par défaut si vous utilisez un thème `AppCompat` . Ce style permet de garantir que tous les boutons se ressemblent par défaut, conformément à la spécification de conception de matériau .

Dans ce cas, la couleur d'accent est rose.

1. Bouton simple: `@style/Widget.AppCompat.Button`


```
<Button
    style="@style/Widget.AppCompat.Button"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="16dp"
    android:text="@string/simple_button"/>
```

2. Bouton de couleur: `@style/Widget.AppCompat.Button.Colored`
   Le style `Widget.AppCompat.Button.Colored` étend le style `Widget.AppCompat.Button` et applique automatiquement la **couleur d'accentuation** sélectionnée dans le thème de votre application.


```
<Button
```

```
    style="@style/Widget.AppCompat.Button.Colored"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="16dp"
    android:text="@string/colored_button"/>
```

Si vous souhaitez personnaliser la couleur d'arrière-plan sans modifier la couleur d'accent de votre *thème principal,* vous pouvez créer un *thème personnalisé* (étendant le thème `ThemeOverlay`) pour votre `Button` et l'affecter à l'attribut `android:theme` du bouton:

```
<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"/>
```

Définissez le thème dans `res/values/themes.xml` :

```
 <style name="MyButtonTheme" parent="ThemeOverlay.AppCompat.Light">
      <item name="colorAccent">@color/my_color</item>
 </style>
```

3. Bouton sans bordure: `@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. Bouton de couleur sans bordure: `@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"/>
```

Comment utiliser TextInputLayout

Assurez-vous que la dépendance suivante est ajoutée au fichier `build.gradle` votre application sous les dépendances:

```
compile 'com.android.support:design:25.3.1'
```

Affiche l'indicateur d'un EditText sous la forme d'une étiquette flottante lors de la saisie d'une valeur.

```
<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>
```

Pour afficher l'icône de l'oeil du mot de passe avec TextInputLayout, nous pouvons utiliser le code suivant:

```
<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>
```

où `app:passwordToggleEnabled="true"` & `android:inputType="textPassword"` paramètres `android:inputType="textPassword"` sont requis.

`app` devrait utiliser l'espace de noms `xmlns:app="http://schemas.android.com/apk/res-auto"`

Vous pouvez trouver plus de détails et d'exemples dans la rubrique dédiée.

Ajouter un TabLayout

TabLayout fournit une disposition horizontale pour afficher les onglets, et est couramment utilisée en conjonction avec un ViewPager .

Assurez-vous que la dépendance suivante est ajoutée au fichier `build.gradle` votre application

sous les dépendances:

```
compile 'com.android.support:design:25.3.1'
```

Vous pouvez maintenant ajouter des éléments à un TabLayout dans votre mise en page à l'aide de la classe TabItem .

Par exemple:

```
<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>
```

Ajoutez un `OnTabSelectedListener` à notifier lorsqu'un onglet de l'onglet `TabLayout` est sélectionné / non sélectionné / resélectionné:

```
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) {

    }
});
```

Des onglets peuvent également être ajoutés / supprimés de `TabLayout` programmation.

```
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` a deux modes, fixe et défilable.

```
tabLayout.setTabMode(TabLayout.MODE_FIXED);
tabLayout.setTabMode(TabLayout.MODE_SCROLLABLE);
```

Celles-ci peuvent également être appliquées en XML:

```
<android.support.design.widget.TabLayout
    android:id="@+id/tabLayout"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:tabMode="fixed|scrollable" />
```

Remarque: les modes `TabLayout` sont mutuellement exclusifs, ce qui signifie qu'un seul peut être actif à la fois.

La couleur de l'indicateur d'onglet correspond à la couleur d'accent définie pour votre thème Conception de matériaux.
Vous pouvez remplacer cette couleur en définissant un style personnalisé dans `styles.xml` , puis en appliquant le style à votre onglet TabLayout:

```
<style name="MyCustomTabLayoutStyle" parent="Widget.Design.TabLayout">
    <item name="tabIndicatorColor">@color/your_color</item>
</style>
```

Vous pouvez ensuite appliquer le style à la vue en utilisant:

```
<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

L'effet tactile Ripple a été introduit avec la conception matérielle sous Android 5.0 (API niveau 21) et l'animation est implémentée par la nouvelle classe RippleDrawable .

> Dessiné qui montre un effet d'entraînement en réponse aux changements d'état. La position d'ancrage de l'ondulation pour un état donné peut être spécifiée en appelant `setHotspot(float x, float y)` avec l'identificateur d'attribut state correspondant.

5.0

En général, l'effet d'ondulation pour **les boutons standard fonctionne par défaut** dans l'API 21 et au-dessus, et pour d'autres vues accessibles, il peut être obtenu en spécifiant:

```
android:background="?android:attr/selectableItemBackground">
```

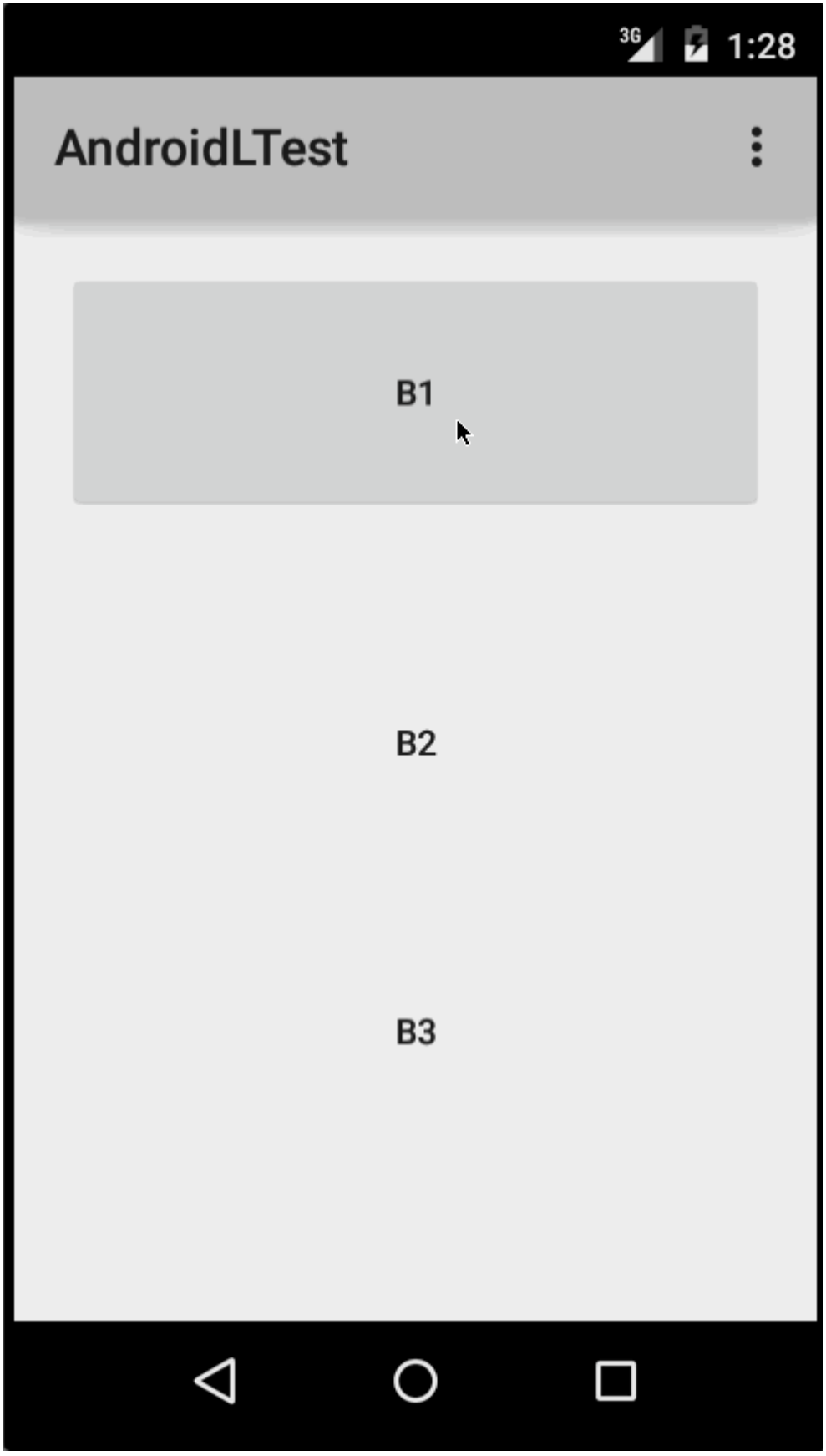
pour les ondulations contenues dans la vue ou:

```
android:background="?android:attr/selectableItemBackgroundBorderless"
```

pour les ondulations qui dépassent les limites de la vue.

Par exemple, dans l'image ci-dessous,

- B1 est un bouton sans arrière-plan,
- B2 est configuré avec `android:background="android:attr/selectableItemBackground"`
- B3 est configuré avec `android:background="android:attr/selectableItemBackgroundBorderless"`



(Courtoisie d'image: http://blog.csdn.net/a396901990/article/details/40187203)

Vous pouvez obtenir la même chose en code en utilisant:

```
int[] attrs = new int[]{R.attr.selectableItemBackground};
TypedArray typedArray = getActivity().obtainStyledAttributes(attrs);
```

```
int backgroundResource = typedArray.getResourceId(0, 0);
myView.setBackgroundResource(backgroundResource);
```

Les ondulations peuvent également être ajoutées à une vue en utilisant l'attribut `android:foreground` la même manière que ci-dessus. Comme son nom l' indique, dans le cas où l'ondulation est ajouté au premier plan, l'ondulation apparaîtra au- dessus de toute vue , il est ajouté à (par exemple `ImageView` , un `LinearLayout` contenant des vues multiples, etc.).

Si vous souhaitez personnaliser l'effet d'ondulation dans une vue, vous devez créer un nouveau fichier `XML` dans le répertoire pouvant être dessiné.

Voici quelques exemples:

Exemple 1 : une ondulation sans limite

```
<ripple xmlns:android="http://schemas.android.com/apk/res/android"
    android:color="#ffff0000" />
```

Exemple 2 : Ondulation avec masque et couleur d'arrière-plan

```
<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>
```

S'il existe une `view` avec un arrière-plan *déjà spécifié* avec une `shape` , des `corners` et d'autres balises, pour ajouter une ondulation à cette vue, utilisez un `mask layer` et définissez l'ondulation comme arrière-plan de la vue.

Exemple :

```
 <?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>
```

Exemple 3 : Ondulation sur une ressource pouvant être dessinée

```
<ripple xmlns:android="http://schemas.android.com/apk/res/android"
    android:color="#ff0000ff">
    <item android:drawable="@drawable/my_drawable" />
</ripple>
```

Utilisation: Pour associer votre fichier xml à une vue, définissez-le comme arrière-plan (en supposant que votre fichier d'ondulation s'appelle `my_ripple.xml`):

```
<View
    android:id="@+id/myViewId"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:background="@drawable/my_ripple" />
```

Sélecteur:

L'ondulation pouvant être utilisée à la place des sélecteurs d'état de couleur si votre version cible est v21 ou supérieure (vous pouvez également placer le sélecteur d'ondulation dans le dossier `drawable-v21`):

```
<!-- /drawable/button.xml: -->
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:state_pressed="true" android:drawable="@drawable/button_pressed"/>
    <item android:drawable="@drawable/button_normal"/>
</selector>

<!--/drawable-v21/button.xml:-->
<?xml version="1.0" encoding="utf-8"?>
<ripple xmlns:android="http://schemas.android.com/apk/res/android"
    android:color="?android:colorControlHighlight">
   <item android:drawable="@drawable/button_normal" />
</ripple>
```

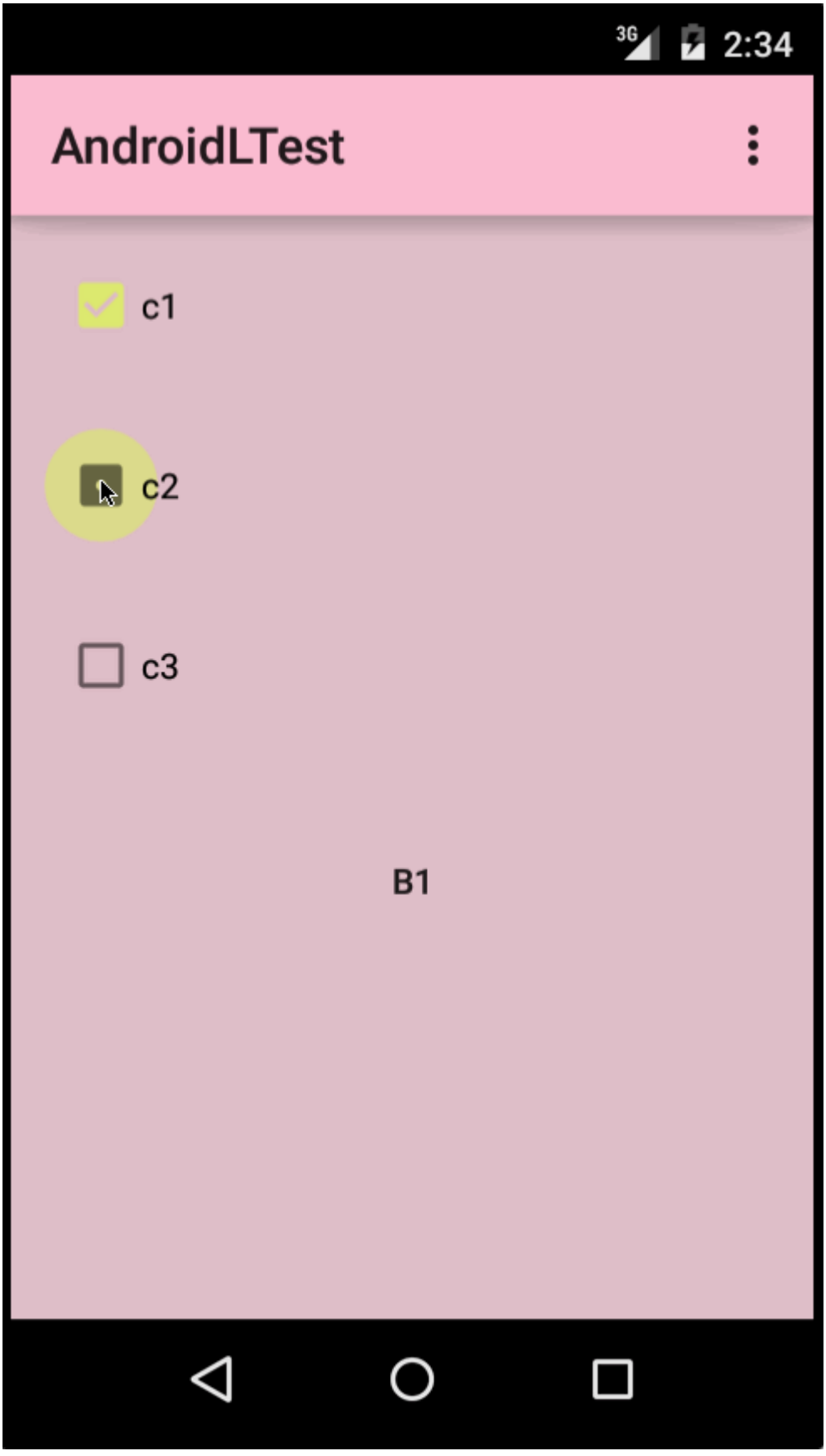
Dans ce cas, la couleur de l'état par défaut de votre vue serait blanche et l'état pressé indiquerait l'ondulation.

Point à noter: Using `?android:colorControlHighlight` donnera à la ondulation la même couleur que les ondulations intégrées dans votre application.

Pour modifier uniquement la couleur de l'ondulation, vous pouvez personnaliser la couleur `android:colorControlHighlight` dans votre thème comme `android:colorControlHighlight` :

```
<?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>
```

puis utilisez ce thème dans vos activités, etc. L'effet serait comme l'image ci-dessous:



(Courtoisie d'image: http://blog.csdn.net/a396901990/article/details/40187203)

Ajouter un tiroir de navigation

Les tiroirs de navigation permettent de naviguer vers des destinations de premier niveau dans une application.

Assurez-vous d'avoir ajouté la bibliothèque d'aide à la conception dans votre fichier `build.gradle` sous des dépendances:

```
 dependencies {
    // ...
    compile 'com.android.support:design:25.3.1'
}
```

Ensuite, ajoutez `DrawerLayout` et `NavigationView` dans votre fichier de ressources de disposition XML.
Le `DrawerLayout` est juste un conteneur sophistiqué qui permet à `NavigationView` , le tiroir de navigation proprement dit, de sortir de la gauche ou de la droite de l'écran. Remarque: pour les appareils mobiles, la taille de tiroir standard est de 320dp.

```
<!-- res/layout/activity_main.xml -->
<android.support.v4.widget.DrawerLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/navigation_drawer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"
    tools:openDrawer="start">
    <! -- You can use "end" to open drawer from the right side -->

    <android.support.design.widget.CoordinatorLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:fitsSystemWindows="true">

        <android.support.design.widget.AppBarLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:theme="@style/AppTheme.AppBarOverlay">

            <android.support.v7.widget.Toolbar
                android:id="@+id/toolbar"
                android:layout_width="match_parent"
                android:layout_height="?attr/actionBarSize"
                android:background="?attr/colorPrimary"
                app:popupTheme="@style/AppTheme.PopupOverlay" />

        </android.support.design.widget.AppBarLayout>

    </android.support.design.widget.CoordinatorLayout>

    <android.support.design.widget.NavigationView
        android:id="@+id/navigation_drawer"
        android:layout_width="320dp"
        android:layout_height="match_parent"
        android:layout_gravity="start"
        android:fitsSystemWindows="true"
        app:headerLayout="@layout/drawer_header"
        app:menu="@menu/navigation_menu" />

</android.support.v4.widget.DrawerLayout>
```

Maintenant, si vous le souhaitez, créez un **fichier d'en-tête** qui servira de haut de votre tiroir de navigation. Ceci est utilisé pour donner un aspect beaucoup plus élégant au tiroir.

```
<!-- 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>
```

Il est référencé dans la balise `NavigationView` dans l' `app:headerLayout="@layout/drawer_header"` . Cette `app:headerLayout` gonfle `app:headerLayout` la mise en page spécifiée dans l'en-tête. Cela peut aussi être fait à l'exécution avec:

```
// 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);
```

Pour remplir automatiquement votre tiroir de navigation avec les éléments de navigation compatibles avec la conception du matériau, créez un fichier de menu et ajoutez des éléments selon vos besoins. Remarque: bien que les icônes des éléments ne soient pas obligatoires, elles sont suggérées dans la spécification de conception du matériau .
Il est référencé dans la balise `NavigationView` dans l' `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>
```

Pour séparer les éléments en groupes, placez-les dans un `<menu>` imbriqué dans un autre `<item>` avec un attribut `android:title` ou enveloppez-les avec la `<group>` .

Maintenant que la mise en page est terminée, passez au code d' `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();
```

Vous pouvez maintenant faire ce que vous voulez dans la vue d'en-tête de `NavigationView`

```
View headerView = navigationView.getHeaderView();
TextView headerTextView = (TextView) headerview.findViewById(R.id.header_text_view);
ImageView headerImageView = (ImageView) headerview.findViewById(R.id.header_image);
// Set navigation header text
headerTextView.setText("User name");
// Set navigation header image
headerImageView.setImageResource(R.drawable.header_image);
```

La vue d'en-tête se comporte comme n'importe quelle autre `View` . Une fois que vous utilisez `findViewById()` et que vous ajoutez d'autres `View` à votre fichier de mise en page, vous pouvez définir les propriétés de tout `findViewById()` .

Vous pouvez trouver plus de détails et d'exemples dans la rubrique dédiée .

Feuilles de fond dans la bibliothèque d'aide à la conception

Les feuilles du bas glissent du bas de l'écran pour révéler plus de contenu.
Ils ont été ajoutés à la bibliothèque de support Android dans la version v25.1.0 et prennent en

charge toutes les versions.

Assurez-vous que la dépendance suivante est ajoutée au fichier build.gradle de votre application sous les dépendances:

```
compile 'com.android.support:design:25.3.1'
```

Feuilles de fond persistantes

Vous pouvez obtenir une feuille inférieure persistante associant un `BottomSheetBehavior` à un enfant. Vue d'un `CoordinatorLayout`

```
<android.support.design.widget.CoordinatorLayout >

    <!-- .....   -->

    <LinearLayout
       android:id="@+id/bottom_sheet"
       android:elevation="4dp"
       android:minHeight="120dp"
       app:behavior_peekHeight="120dp"
       ...
       app:layout_behavior="android.support.design.widget.BottomSheetBehavior">

           <!-- .....   -->

       </LinearLayout>

</android.support.design.widget.CoordinatorLayout>
```

Ensuite, dans votre code, vous pouvez créer une référence en utilisant:

```
// The View with the BottomSheetBehavior
View bottomSheet = coordinatorLayout.findViewById(R.id.bottom_sheet);
BottomSheetBehavior mBottomSheetBehavior = BottomSheetBehavior.from(bottomSheet);
```

Vous pouvez définir l'état de votre BottomSheetBehavior en utilisant la méthode setState () :

```
mBottomSheetBehavior.setState(BottomSheetBehavior.STATE_EXPANDED);
```

Vous pouvez utiliser l'un de ces états:

- `STATE_COLLAPSED` : cet état réduit est la valeur par défaut et ne montre qu'une partie de la disposition en bas. La hauteur peut être contrôlée avec l'attribut `app:behavior_peekHeight` (la valeur par défaut est 0)
- `STATE_EXPANDED` : l'état complètement développé de la feuille du bas, où soit la totalité de la feuille inférieure est visible (si sa hauteur est inférieure à celle du `CoordinatorLayout`), soit la totalité du `CoordinatorLayout` est remplie
- `STATE_HIDDEN`

: désactivé par défaut (et activé avec l'attribut `app:behavior_hideable`), ce qui permet aux utilisateurs de glisser le bas de la feuille pour masquer complètement la feuille du bas

Pour ouvrir ou fermer la feuille de calcul en cliquant sur une vue de votre choix, un bouton disons, voici comment basculer le comportement de la feuille et la vue de mise à jour.

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

Toutefois, le comportement de `BottomSheet` comporte également une fonctionnalité permettant à l'utilisateur d'interagir avec le balayage UP ou Down avec un mouvement DRAG. Dans un tel cas, nous pourrions ne pas être en mesure de mettre à jour la vue dépendante (comme le bouton ci-dessus) si l'état de la feuille a changé. Par ailleurs, vous souhaitez recevoir des rappels de modifications d'état, vous pouvez donc ajouter un `BottomSheetCallback` pour écouter les événements de glissement utilisateur:

```
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
   }
 });
```

Et si vous voulez seulement que votre feuille de fond soit visible uniquement en mode COLLAPSED et EXPANDED, et que vous n'utilisez jamais HIDE:

```
mBottomSheetBehavior2.setHideable(false);
```

DialogFragment de la feuille inférieure

Vous pouvez également afficher un BottomSheetDialogFragment à la place d'une vue dans la feuille du bas. Pour ce faire, vous devez d'abord créer une nouvelle classe qui étend BottomSheetDialogFragment.

Dans la méthode `setupDialog()` , vous pouvez gonfler un nouveau fichier de présentation et récupérer le BottomSheetBehavior de la vue du conteneur dans votre activité. Une fois que vous avez le comportement, vous pouvez créer et associer un BottomSheetCallback avec lui pour supprimer le fragment lorsque la feuille est masquée.

```
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);
        }
    }
}
```

Enfin, vous pouvez appeler show () sur une instance de votre fragment pour l'afficher dans la feuille du bas.

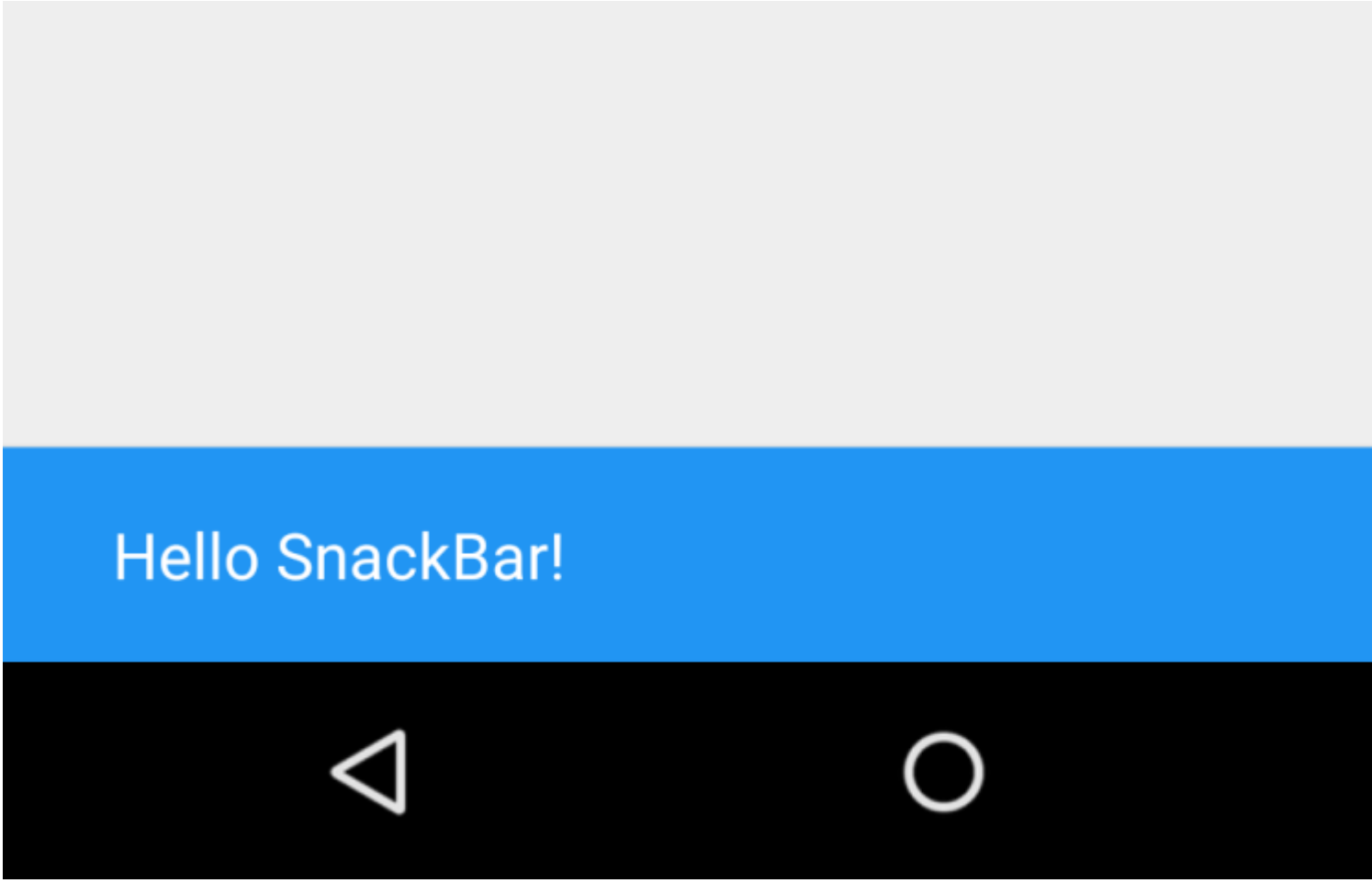
```
BottomSheetDialogFragment bottomSheetDialogFragment = new BottomSheetDialogFragmentExample();
bottomSheetDialogFragment.show(getSupportFragmentManager(),
bottomSheetDialogFragment.getTag());
```

Vous pouvez trouver plus de détails dans la rubrique dédiée

Ajouter un snack

L'une des principales caractéristiques de Material Design est l'ajout d'une `Snackbar` , qui remplace en théorie le `Toast` précédent. Selon la documentation Android:

> Les snackbars contiennent une seule ligne de texte directement liée à l'opération effectuée. Ils peuvent contenir une action de texte, mais pas d'icônes. Les toasts sont principalement utilisés pour la messagerie système. Ils s'affichent également en bas de l'écran, mais ne peuvent pas être déplacés hors écran.



Les toasts peuvent toujours être utilisés dans Android pour afficher des messages aux utilisateurs. Toutefois, si vous avez décidé d'utiliser la conception matérielle dans votre application, il est recommandé d'utiliser un snack. Au lieu d'être affichée en superposition sur votre écran, une `Snackbar` apparaît en bas.

Voici comment cela se fait:

```
Snackbar snackbar = Snackbar
        .make(coordinatorLayout, "Here is your new Snackbar", Snackbar.LENGTH_LONG);
snackbar.show();
```

En ce qui concerne la durée d' `Snackbar` la `Snackbar` , nous avons les options similaires à celles proposées par `Toast` ou nous pouvons définir une durée personnalisée en millisecondes:

- `LENGTH_SHORT`
- `LENGTH_LONG`
- `LENGTH_INDEFINITE`

- `setDuration()` (depuis la version `22.2.1`)

Vous pouvez également ajouter des fonctionnalités dynamiques à votre `Snackbar` telles que `ActionCallback` ou une couleur personnalisée. Toutefois, faites attention aux directives de conception offertes par Android lors de la personnalisation d'une `Snackbar` .

L'implémentation de la `Snackbar` présente toutefois une limitation. La disposition parente de la vue dans laquelle vous allez implémenter une `Snackbar` doit être un `CoordinatorLayout` . Cela permet de créer la fenêtre contextuelle du bas.

Voici comment définir un `CoordinatorLayout` dans votre fichier XML de présentation:

```
<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>
```

Le `CoordinatorLayout` doit alors être défini dans la méthode `onCreate` votre activité, puis utilisé lors de la création de la `Snackbar` elle-même.

Pour plus d'informations sur la `Snackbar` , consultez la documentation officielle ou la rubrique dédiée de la documentation.

Lire Conception matérielle en ligne: https://riptutorial.com/fr/android/topic/124/conception-materielle

Chapitre 66: Configuration de Jenkins CI pour les projets Android

Examples

Approche étape par étape pour configurer Jenkins pour Android

Ceci est un guide étape par étape pour configurer le processus de génération automatisé à l'aide de Jenkins CI pour vos projets Android. Les étapes suivantes supposent que vous avez un nouveau matériel avec n'importe quelle version de Linux installée. Il est également pris en compte que vous pourriez avoir une machine distante.

PARTIE I: Configuration initiale sur votre machine

1. Connectez-vous via *ssh* à votre machine Ubuntu:

   ssh username@xxx.xxx.xxx

2. Téléchargez une version du SDK Android sur votre machine:

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

3. Décompressez le fichier *tar* téléchargé:

   sudo apt-get install tar
   tar -xvf android-sdk_r24.4.1-linux.tgz

4. Maintenant, vous devez installer Java 8 sur votre machine Ubuntu, ce qui est une exigence pour les compilations Android sur Nougat. Jenkins vous demanderait d'installer JDK et JRE 7 en suivant les étapes ci-dessous:

   sudo apt-get installe python-software-properties
   sudo add-apt-repository ppa: webupd8team / java
   sudo apt-get update
   apt-get install openjdk-8-jdk

5. Maintenant, installez Jenkins sur votre machine Ubuntu:

   wget -q -O - https://pkg.jenkins.io/debian/jenkins-ci.org.key | sudo apt-key add -
   sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binaire />
   /etc/apt/sources.list.d/jenkins.list'
   sudo apt-get update
   sudo apt-get install jenkins

6. Téléchargez la dernière version de Gradle prise en charge pour votre configuration Android:

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

7. Configurez Android sur votre machine Ubuntu. Déplacez-vous d'abord vers le dossier des *outils du* dossier SDK Android téléchargé à l'étape 2:

   cd android-sdk-linux / tools **// répertorie le SDK disponible**
   mise à jour Android sdk --no-ui **// Mises à jour de la version du SDK**
   android list sdk -a | grep "SDK Build-tools" **// répertorie les outils de génération disponibles**
   mise à jour Android sdk -a -u -t 4 **// met à jour la version des outils de compilation à celle répertoriée comme 4 par prev. cmd.**
   mettre à jour java

8. Installez *Git* ou tout autre VCS sur votre machine:

   sudo apt-get install git

9. Connectez-vous maintenant à Jenkins en utilisant votre navigateur Internet. Tapez `ipAddress:8080` dans la barre d'adresse.

10. Afin de recevoir le mot de passe pour la première connexion, veuillez vérifier le fichier correspondant comme suit (vous aurez besoin des autorisations su pour accéder à ce fichier):

    cat / var / lib / jenkins / secrets / initialAdminPassword

PARTIE II: Configurer Jenkins pour construire des jobs Android

1. Une fois connecté, accédez au chemin suivant:

   Jenkins> Gérer Jenkins> Configuration de l'outil global

2. À cet emplacement, ajoutez `JAVA_HOME` avec les entrées suivantes:

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

3. Ajoutez également les valeurs suivantes à *Git* et enregistrez les variables d'environnement:

   Nom = par défaut
   / usr / bin / git

4. Maintenant, allez sur le chemin suivant:

Jenkins> Gérer Jenkins> Configuration

5. À cet emplacement, ajoutez `ANDROID_HOME` aux "propriétés globales":

   Nom = ANDROID_HOME
   Valeur = / home / nom d'utilisateur / android-sdk-linux

Partie III: Créer un Job Jenkins pour votre projet Android

1. Cliquez sur *Nouvel élément* dans l'écran d'accueil de Jenkins.
2. Ajoutez un *nom de projet* et une *description* .
3. Dans l'onglet *Général* , sélectionnez *Avancé* . Ensuite, sélectionnez *Utiliser un espace de travail personnalisé* :

   Répertoire / home / user / Code / ProjectFolder

4. Dans la gestion du code source, sélectionnez *Git* . J'utilise *Bitbucket* dans cet exemple:

   URL du référentiel = https: // nom d'utilisateur: password@bitbucket.org/project/projectname.git

5. Sélectionnez des comportements supplémentaires pour votre référentiel:

   Nettoyer avant la caisse
   Commander à un sous-répertoire. Sous-répertoire local pour repo / home / user / Code / ProjectFolder

6. Sélectionnez une branche que vous souhaitez construire:

   */maîtriser

7. Dans l'onglet *Générer* , sélectionnez *Exécuter le shell* dans *Ajouter une étape de construction* .
8. Dans le *shell Execute* , ajoutez la commande suivante:

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

9. Si vous souhaitez exécuter Lint sur le projet, ajoutez une autre étape de construction dans le *shell Execute* :

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

Maintenant, votre système est enfin configuré pour construire des projets Android en utilisant Jenkins. Cette configuration rend votre vie beaucoup plus facile pour la publication de versions

pour les équipes QA et UAT.

PS: Jenkins étant un utilisateur différent sur votre machine Ubuntu, vous devez lui donner le droit de créer des dossiers dans votre espace de travail en exécutant la commande suivante:

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

Lire Configuration de Jenkins CI pour les projets Android en ligne:
https://riptutorial.com/fr/android/topic/7830/configuration-de-jenkins-ci-pour-les-projets-android

Chapitre 67: Connexions Wi-Fi

Examples

Se connecter avec le cryptage WEP

Cet exemple se connecte à un point d'accès Wi-Fi avec cryptage WEP, à l'aide d'un SSID et du mot de passe.

```
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;
    }
}
```

Se connecter avec le cryptage WPA2

Cet exemple se connecte à un point d'accès Wi-Fi avec cryptage WPA2.

```
public boolean ConnectToNetworkWPA(String networkSSID, String password) {
```

```
  try {
    WifiConfiguration conf = new WifiConfiguration();
    conf.SSID = "\"" + networkSSID + "\"";   // Please note the quotes. String should contain
SSID in quotes

    conf.preSharedKey = "\"" + password + "\"";

    conf.status = WifiConfiguration.Status.ENABLED;
    conf.allowedGroupCiphers.set(WifiConfiguration.GroupCipher.TKIP);
    conf.allowedGroupCiphers.set(WifiConfiguration.GroupCipher.CCMP);
    conf.allowedKeyManagement.set(WifiConfiguration.KeyMgmt.WPA_PSK);
    conf.allowedPairwiseCiphers.set(WifiConfiguration.PairwiseCipher.TKIP);
    conf.allowedPairwiseCiphers.set(WifiConfiguration.PairwiseCipher.CCMP);

    Log.d("connecting", conf.SSID + " " + conf.preSharedKey);

    WifiManager wifiManager = (WifiManager)
this.getApplicationContext().getSystemService(Context.WIFI_SERVICE);
    wifiManager.addNetwork(conf);

    Log.d("after connecting", conf.SSID + " " + conf.preSharedKey);

    List<WifiConfiguration> list = wifiManager.getConfiguredNetworks();
    for( WifiConfiguration i : list ) {
      if(i.SSID != null && i.SSID.equals("\"" + networkSSID + "\"")) {
        wifiManager.disconnect();
        wifiManager.enableNetwork(i.networkId, true);
        wifiManager.reconnect();
        Log.d("re connecting", i.SSID + " " + conf.preSharedKey);

        break;
      }
    }

    //WiFi Connection success, return true
    return true;
  } catch (Exception ex) {
    System.out.println(Arrays.toString(ex.getStackTrace()));
    return false;
  }
}
```

Rechercher des points d'accès

Cet exemple analyse les points d'accès disponibles et les réseaux ad hoc. `btnScan` active un scan initié par la méthode `WifiManager.startScan()` . Après l'analyse, `WifiManager` appelle l'intention `SCAN_RESULTS_AVAILABLE_ACTION` et la classe `WifiScanReceiver` traite le résultat de l'analyse. Les résultats sont affichés dans un `TextView` .

```
public class MainActivity extends AppCompatActivity {

    private final static String TAG = "MainActivity";

    TextView txtWifiInfo;
    WifiManager wifi;
    WifiScanReceiver wifiReceiver;

    @Override
```

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

        wifi=(WifiManager)getSystemService(Context.WIFI_SERVICE);
        wifiReceiver = new WifiScanReceiver();

        txtWifiInfo = (TextView)findViewById(R.id.txtWifiInfo);
        Button btnScan = (Button)findViewById(R.id.btnScan);
        btnScan.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Log.i(TAG, "Start scan...");
                wifi.startScan();
            }
        });
    }

    protected void onPause() {
        unregisterReceiver(wifiReceiver);
        super.onPause();
    }

    protected void onResume() {
        registerReceiver(
            wifiReceiver,
            new IntentFilter(WifiManager.SCAN_RESULTS_AVAILABLE_ACTION)
        );
        super.onResume();
    }

    private class WifiScanReceiver extends BroadcastReceiver {
        public void onReceive(Context c, Intent intent) {
            List<ScanResult> wifiScanList = wifi.getScanResults();
            txtWifiInfo.setText("");
            for(int i = 0; i < wifiScanList.size(); i++){
                String info = ((wifiScanList.get(i)).toString());
                txtWifiInfo.append(info+"\n\n");
            }
        }
    }
}
```

Autorisations

Les autorisations suivantes doivent être définies dans *AndroidManifest.xml* :

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

`android.permission.ACCESS_WIFI_STATE` est nécessaire pour appeler `WifiManager.getScanResults()` . Sans `android.permission.CHANGE_WIFI_STATE` vous ne pouvez pas lancer une analyse avec `WifiManager.startScan()` .

Lors de la compilation du projet pour le niveau d'API 23 ou supérieur (Android 6.0 et supérieur), `android.permission.ACCESS_FINE_LOCATION` ou `android.permission.ACCESS_COARSE_LOCATION` doit être inséré. En outre, cette autorisation doit être demandée, par exemple dans la méthode `onCreate` de

votre activité principale:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    ...
    String[] PERMS_INITIAL={
            Manifest.permission.ACCESS_FINE_LOCATION,
    };
    ActivityCompat.requestPermissions(this, PERMS_INITIAL, 127);
}
```

Lire Connexions Wi-Fi en ligne: https://riptutorial.com/fr/android/topic/3288/connexions-wi-fi

Chapitre 68: ContrainteLayout

Introduction

`ConstraintLayout` est un `ViewGroup` qui vous permet de positionner et de dimensionner les widgets de manière flexible. Il est compatible avec Android 2.3 (API niveau 9) et supérieur.

Il vous permet de créer des mises en page volumineuses et complexes avec une hiérarchie de vues plates. Il est similaire à `RelativeLayout` en ce sens que toutes les vues sont disposées en fonction des relations entre les vues frères et la présentation parent, mais elles sont plus flexibles que `RelativeLayout` et plus faciles à utiliser avec l'éditeur de mise en page d'Android Studio.

Syntaxe

- **ContrainteLayout**
  - public void addView (Voir enfant, int index, ViewGroup.LayoutParams params)
  - public ConstraintLayout.LayoutParams generateLayoutParams (AttributeSet attrs)
  - public void onViewAdded (Voir la vue)
  - public void onViewRemoved (Voir la vue)
  - void public removeView (Voir la vue)
  - public void requestLayout ()
  - booléen protégé checkLayoutParams (paramètres de ViewGroup.LayoutParams)
  - protected ConstraintLayout.LayoutParams generateDefaultLayoutParams ()
  - protected ViewGroup.LayoutParams generateLayoutParams (paramètres de ViewGroup.LayoutParams)
  - void protégé surLayout (booléen modifié, int gauche, int top, int droite, int bottom)
  - protected onMeasure (int widthMeasureSpec, int heightMeasureSpec)
- **ContrainteLayout.LayoutParams**
  - public void resolveLayoutDirection (int layoutDirection)
  - annulation publique validate ()
  - protected void setBaseAttributes (TypedArray a, int widthAttr, int heightAttr)

Paramètres

Paramètre	Détails
enfant	La `View` à ajouter à la mise en page
indice	L'index de la `View` dans la hiérarchie de présentation
params	Le `LayoutParams` de la `View`
attrs	Le `AttributeSet` qui définit les `LayoutParams`
vue	La `View` qui a été ajoutée ou supprimée
modifié	Indique si cette `View` a changé de taille ou de position
la gauche	La position de gauche par rapport à la `View` parent
Haut	La position supérieure par rapport à la `View` parent
droite	La bonne position par rapport au parent `View`
bas	La position du bas par rapport à la `View` parent
widthMeasureSpec	L'espace horizontal requis par la `View` parent
heightMeasureSpec	Les exigences d'espace vertical imposées par la `View` parent
layoutDirection	-
une	-
widthAttr	-
heightAttr	-

Remarques

Sur Google IO 2016, Google a annoncé une nouvelle mise en page Android appelée ConstraintLayout.
Faites attention car actuellement, cette mise en page est une **version bêta** .

Pour en savoir plus sur la disposition des contraintes:

https://codelabs.developers.google.com/codelabs/constraint-layout/index.html

Examples

Ajouter ConstraintLayout à votre projet

Pour travailler avec ConstraintLayout, vous avez besoin d'Android Studio version 2.2 ou plus récent et avez au moins la version 32 (ou supérieure) du référentiel de support Android.

1. Ajoutez la bibliothèque de contrainte de contrainte en tant que dépendance dans votre fichier `build.gradle` :

```
dependencies {
   compile 'com.android.support.constraint:constraint-layout:1.0.2'
}
```

2. Projet de synchronisation

Pour ajouter une nouvelle disposition de contrainte à votre projet:

1. **Cliquez avec** le **bouton droit sur** le répertoire de disposition de votre module, puis cliquez sur `New > XML > Layout XML.`
2. Entrez un **nom** pour la mise en page et entrez `"android.support.constraint.ConstraintLayout"` pour la balise racine.
3. Cliquez sur **Terminer** .

Sinon, ajoutez simplement un fichier de mise en page:

```
<?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>
```

Chaînes

Depuis `ConstraintLayout` alpha 9, les **chaînes** sont disponibles. Une **chaîne** est un ensemble de vues à l'intérieur d'un `ConstraintLayout` qui sont connectées entre elles, c.-à-d. **A** connectée à **B** avec une contrainte et **B** connectée à **A** avec une autre contrainte.

Exemple:

```
<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>
```

Dans cet exemple, les deux vues sont positionnées les unes sur les autres et les deux sont centrées verticalement. Vous pouvez modifier la position verticale de ces vues en ajustant le **biais de** la chaîne. Ajoutez le code suivant au premier élément d'une chaîne:

```
app:layout_constraintVertical_bias="0.2"
```

Dans une chaîne verticale, le premier élément est la vue la plus haute et, dans une chaîne horizontale, la vue la plus à gauche. Le premier élément définit le comportement de toute la chaîne.

Les chaînes sont une nouvelle fonctionnalité et sont fréquemment mises à jour. Voici une documentation Android officielle sur les chaînes.

Lire ContrainteLayout en ligne: https://riptutorial.com/fr/android/topic/5076/contraintelayout

Chapitre 69: ContrainteSet

Introduction

Cette classe vous permet de définir par programmation un ensemble de contraintes à utiliser avec `ConstraintLayout` . Il vous permet de créer et d'enregistrer des contraintes et de les appliquer à une `ConstraintLayout` existante.

Examples

ConstraintSet avec ContraintLayout par programme

```
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
        }
    }
}
```

Lire ContrainteSet en ligne: https://riptutorial.com/fr/android/topic/9334/contrainteset

Chapitre 70: Conversion de la parole en texte

Examples

Discours à texte avec dialogue d'invite Google par défaut

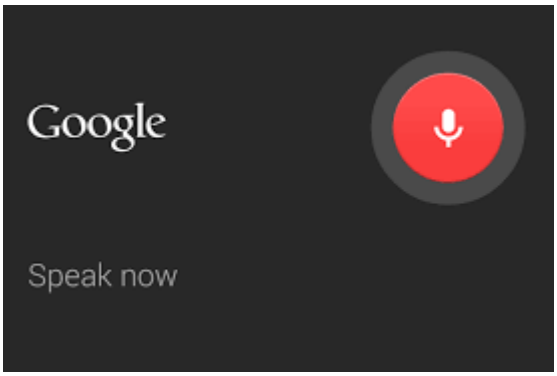
Déclencher la traduction du texte en texte

```
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();
    }
}
```

Obtenez les résultats traduits dans 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;
    }

    }
}
```

Sortie



Discours au texte sans dialogue

Le code suivant peut être utilisé pour déclencher une traduction vocale sans afficher de boîte de dialogue:

```
public void startListeningWithoutDialog() {
    // Intent to listen to user vocal input and return the result to the same activity.
    Intent intent = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);

    // Use a language model based on free-form speech recognition.
    intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,
            RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);
    intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE, Locale.getDefault());
    intent.putExtra(RecognizerIntent.EXTRA_MAX_RESULTS, 5);
    intent.putExtra(RecognizerIntent.EXTRA_CALLING_PACKAGE,
            appContext.getPackageName());

    // Add custom listeners.
    CustomRecognitionListener listener = new CustomRecognitionListener();
    SpeechRecognizer sr = SpeechRecognizer.createSpeechRecognizer(appContext);
    sr.setRecognitionListener(listener);
    sr.startListening(intent);
}
```

La classe d'écouteur personnalisée `CustomRecognitionListener` utilisée dans le code ci-dessus est implémentée comme suit:

```
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);
    }
}
```

Lire Conversion de la parole en texte en ligne:
https://riptutorial.com/fr/android/topic/6252/conversion-de-la-parole-en-texte

Chapitre 71: Convertir une chaîne vietnamienne en anglais

Examples

Exemple:

```
String myStr = convert("Lê Minh Thoại là người Việt Nam");
```

converti:

```
"Le Minh Thoai la nguoi Viet Nam"
```

Chuyển chuỗi Tiếng Việt

```
  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;
}
```

Lire Convertir une chaîne vietnamienne en anglais en ligne:
https://riptutorial.com/fr/android/topic/10946/convertir-une-chaine-vietnamienne-en-anglais

Chapitre 72: CoordinateurLayout et comportements

Introduction

Le CoordinatorLayout est un FrameLayout extrêmement puissant et l'objectif de ce groupe de vues est de coordonner les vues qu'il contient.

Le principal atout de CoordinatorLayout est sa capacité à coordonner les animations et les transitions des vues dans le fichier XML lui-même.

CoordinatorLayout est destiné à deux cas d'utilisation principaux:

: Comme un décor d'application de haut niveau ou une disposition chromée

: En tant que conteneur pour une interaction spécifique avec une ou plusieurs vues enfants

Remarques

Le `CoordinatorLayout` est un conteneur qui étend le `FrameLayout` .
En attachant un `CoordinatorLayout.Behavior` à un enfant direct de `CoordinatorLayout` , vous pouvez intercepter les événements tactiles, les encarts de fenêtre, les mesures, la mise en page et le défilement imbriqué.

En spécifiant les `Behaviors` pour les vues enfant d'un `CoordinatorLayout` vous pouvez fournir de nombreuses interactions différentes au sein d'un seul parent et ces vues peuvent également interagir les unes avec les autres. Les classes de vue peuvent spécifier un comportement par défaut lorsqu'elles sont utilisées comme enfant d'un `CoordinatorLayout` à l'aide de l'annotation `DefaultBehavior` .

Examples

Créer un comportement simple

Pour créer un `Behavior` simplement la classe `CoordinatorLayout.Behavior` .

Prolongez le CoordinatorLayout.Behavior

Exemple:

```
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);
    }
}
```

Ce comportement doit être associé à un enfant. Vue d'un `CoordinatorLayout` à appeler.

Attacher un comportement par programmation

```
MyBehavior myBehavior = new MyBehavior();
CoordinatorLayout.LayoutParams params = (CoordinatorLayout.LayoutParams)
view.getLayoutParams();
params.setBehavior(myBehavior);
```

Joindre un comportement en XML

Vous pouvez utiliser l'attribut `layout_behavior` pour associer le comportement en XML:

```
<View
  android:layout_height="...."
  android:layout_width="...."
  app:layout_behavior=".MyBehavior" />
```

Joindre un comportement automatiquement

Si vous travaillez avec une vue personnalisée, vous pouvez attacher le comportement à l'aide de l'annotation `@CoordinatorLayout.DefaultBehavior` :

```
@CoordinatorLayout.DefaultBehavior(MyBehavior.class)
public class MyView extends ..... {

}
```

Utiliser le SwipeDismissBehavior

`SwipeDismissBehavior` fonctionne sur n'importe quelle vue et implémente les fonctionnalités de swipe à supprimer dans nos mises en page avec un `CoordinatorLayout` .

Utilisez simplement:

```
        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);
```

Créer des dépendances entre les vues

Vous pouvez utiliser le `CoordinatorLayout.Behavior` pour créer des dépendances entre les vues. Vous pouvez ancrer une `View` à une autre `View` par:

- en utilisant l'attribut `layout_anchor` .
- créer un `Behavior` personnalisé et implémenter la méthode `layoutDependsOn` renvoyant `true` .

Par exemple, pour créer un `Behavior` permettant de déplacer un `ImageView` lorsqu'un autre est déplacé (exemple, barre d'outils), procédez comme suit:

- Créez le comportement personnalisé :

  ```
  public class MyBehavior extends CoordinatorLayout.Behavior<ImageView> {...}
  ```

- Remplacez la méthode `layoutDependsOn` renvoyant `true` . Cette méthode est appelée chaque fois qu'une modification est apportée à la mise en page:

  ```
  @Override
  public boolean layoutDependsOn(CoordinatorLayout parent,
          ImageView child, View dependency) {
      // Returns true to add a dependency.
      return dependency instanceof Toolbar;
  }
  ```

- Chaque fois que la méthode `layoutDependsOn` renvoie `true` la méthode `onDependentViewChanged` est appelée:

```
@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);
}
```

Lire CoordinateurLayout et comportements en ligne:
https://riptutorial.com/fr/android/topic/5714/coordinateurlayout-et-comportements

Chapitre 73: Couleurs

Examples

Manipulation de couleur

Pour manipuler les couleurs, nous allons modifier les valeurs argb (alpha, rouge, vert et bleu) d'une couleur.

D'abord, extrayez les valeurs RVB de votre couleur.

```
int yourColor = Color.parse("#ae1f67");

int red = Color.red(yourColor);
int green = Color.green(yourColor);
int blue = Color.blue(yourColor);
```

Vous pouvez maintenant réduire ou augmenter les valeurs rouge, vert et bleu et les combiner pour redevenir une couleur:

```
 int newColor = Color.rgb(red, green, blue);
```

Ou si vous souhaitez y ajouter de l'alpha, vous pouvez l'ajouter en créant la couleur:

```
 int newColor = Color.argb(alpha, red, green, blue);
```

Les valeurs alpha et RVB doivent être comprises entre 0 et 225].

Lire Couleurs en ligne: https://riptutorial.com/fr/android/topic/4986/couleurs

Chapitre 74: Couteau à beurre

Introduction

Butterknife est un outil de liaison de vues qui utilise des annotations pour générer du code standard pour nous. Cet outil est développé par Jake Wharton à Square et est essentiellement utilisé pour enregistrer des lignes de code répétitives telles que `findViewById(R.id.view)` lorsqu'il s'agit de vues, rendant ainsi notre code beaucoup plus propre.

Pour être clair, Butterknife n'est **pas une bibliothèque d'injection de dépendance** . Butterknife injecte du code au moment de la compilation. Il est très similaire au travail effectué par les annotations Android.

Remarques

Couteau à beurre

Liaison de champs et de méthodes pour les vues Android qui utilise le traitement des annotations pour générer du code standard pour vous.

- Éliminez les appels à findViewById en utilisant @BindView sur les champs.
- Regrouper plusieurs vues dans une liste ou un tableau. Utilisez-les tous en même temps avec des actions, des paramètres ou des propriétés.
- Éliminez les classes internes anonymes pour les auditeurs en annotant les méthodes avec @OnClick et autres.
- Éliminez les recherches de ressources en utilisant les annotations de ressources sur les champs.

Plus d'infos: http://jakewharton.github.io/butterknife/

Licence

Examples

Configurer ButterKnife dans votre projet

Configurez votre `build.gradle` au niveau du `build.gradle` pour inclure le plugin `android-apt` :

```
buildscript {
   repositories {
      mavenCentral()
   }

   dependencies {
      classpath 'com.jakewharton:butterknife-gradle-plugin:8.5.1'
   }
}
```

Ensuite, appliquez le plugin `android-apt` dans votre `build.gradle` niveau du `build.gradle` et ajoutez les dépendances ButterKnife:

```
apply plugin: 'android-apt'

android {
    ...
}

dependencies {
      compile 'com.jakewharton:butterknife:8.5.1'
      annotationProcessor 'com.jakewharton:butterknife-compiler:8.5.1'
}
```

Note: Si vous utilisez le nouveau compilateur Jack avec la version 2.2.0 ou plus récente, vous n'avez pas besoin du plug `android-apt` in `android-apt` et vous pouvez remplacer apt par `annotationProcessor` lors de la déclaration de la dépendance du compilateur.

Pour utiliser les annotations ButterKnife, vous ne devez pas oublier de les lier dans `onCreate()` de vos activités ou `onCreateView()` de vos fragments:

```
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;
    }

}
```

Les instantanés de la version de développement sont disponibles dans le référentiel d'instantanés de Sonatype .

Vous trouverez ci-dessous les étapes supplémentaires à suivre pour utiliser ButterKnife dans un projet de bibliothèque.

Pour utiliser ButterKnife dans un projet de bibliothèque, ajoutez le plug-in à votre `build.gradle` niveau du `build.gradle` :

```
buildscript {
    dependencies {
        classpath 'com.jakewharton:butterknife-gradle-plugin:8.5.1'
    }
}
```

… Et ensuite appliquer à votre module en ajoutant ces lignes en haut de votre `build.gradle` niveau de la `build.gradle` :

```
apply plugin: 'com.android.library'
// ...
apply plugin: 'com.jakewharton.butterknife'
```

Maintenant, assurez-vous d'utiliser `R2` au lieu de `R` dans toutes les annotations ButterKnife.

```
class ExampleActivity extends Activity {

    // Bind xml resource to their View
    @BindView(R2.id.user) EditText username;
    @BindView(R2.id.pass) EditText password;

    // Binding resources from drawable,strings,dimens,colors
    @BindString(R.string.choose) String choose;
    @BindDrawable(R.drawable.send) Drawable send;
    @BindColor(R.color.cyan) int cyan;
    @BindDimen(R.dimen.margin) Float generalMargin;

    // Listeners
    @OnClick(R.id.submit)
    public void submit(View view) {
    // TODO submit data to server...
    }

    // bind with butterknife in onCreate
    @Override
    public void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        ButterKnife.bind(this);
        // TODO continue
    }

}
```

Vues contraignantes à l'aide de ButterKnife

Nous pouvons annoter les champs avec `@BindView` et un identifiant de vue pour Butter Knife pour trouver et afficher automatiquement la vue correspondante dans notre présentation.

Vues contraignantes

Relier les vues dans l'activité

```
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...
  }
}
```

Vues contraignantes dans les fragments

```
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();
    }
```

```
}
```

Vues de liaison dans les dialogues

Nous pouvons utiliser `ButterKnife.findById` pour rechercher des vues sur une vue, une activité ou une boîte de dialogue. Il utilise des génériques pour déduire le type de retour et effectue automatiquement la conversion.

```
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);
```

Vues de liaison dans 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);
    }
  }
```

Ressources contraignantes

En plus d'être utile pour les vues de liaison, vous pouvez également utiliser ButterKnife pour lier des ressources telles que celles définies dans `strings.xml` , `drawables.xml` , `colors.xml` , `dimens.xml` , etc.

```
public class ExampleActivity extends Activity {

    @BindString(R.string.title) String title;
    @BindDrawable(R.drawable.graphic) Drawable graphic;
    @BindColor(R.color.red) int red; // int or ColorStateList field
    @BindDimen(R.dimen.spacer) Float spacer; // int (for pixel size) or float (for exact
value) field

    @Override
    public void onCreate(Bundle savedInstanceState) {

        // ...

        ButterKnife.bind(this);
    }

}
```

Listes de vue de reliure

Vous pouvez regrouper plusieurs vues dans une liste ou un tableau. Ceci est très utile lorsque vous devez effectuer une action sur plusieurs vues à la fois.

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

Fixations optionnelles

Par défaut, les `@Bind` et listener sont requises. Une exception est levée si la vue cible est introuvable. Mais si nous ne sommes pas sûrs qu'une vue soit présente ou non, nous pouvons ajouter une annotation `@Nullable` aux champs ou `@Optional` annotation `@Optional` aux méthodes permettant de supprimer ce comportement et créer une liaison facultative.

```
@Nullable
@BindView(R.id.might_not_be_there) TextView mightNotBeThere;

@Optional
@OnClick(R.id.maybe_missing)
void onMaybeMissingClicked() {
  // TODO ...
}
```

Relier les auditeurs à l'aide de ButterKnife

OnClick Listener:

```
@OnClick(R.id.login)
public void login(View view) {
  // Additional logic
}
```

Tous les arguments de la méthode écouteur sont facultatifs:

```
@OnClick(R.id.login)
public void login() {
   // Additional logic
}
```

Le type spécifique sera automatiquement lancé:

```
@OnClick(R.id.submit)
public void sayHi(Button button) {
  button.setText("Hello!");
}
```

Plusieurs ID dans une seule liaison pour la gestion des événements communs:

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

Les vues personnalisées peuvent se lier à leurs propres écouteurs en ne spécifiant pas d'ID:

```
public class CustomButton extends Button {
  @OnClick
  public void onClick() {
    // TODO
  }
}
```

Des vues indomptables dans ButterKnife

Les fragments ont un cycle de vie différent des activités. Lors de la liaison d'un fragment dans onCreateView, définissez les vues sur null dans onDestroyView. Butter Knife renvoie une instance Unbinder lorsque vous appelez bind pour le faire pour vous. Appelez sa méthode de détachement dans le rappel de cycle de vie approprié.

Un exemple:

```
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();
  }
}
```

> Remarque: L'appel de unbind () dans onDestroyView () n'est pas obligatoire, mais recommandé car il permet d'économiser un peu de mémoire si votre application présente un important backstack.

Android Studio ButterKnife Plugin

Android ButterKnife Zelezny

> Plugin pour générer des injections ButterKnife à partir de XML de disposition sélectionnés dans des activités / fragments / adaptateurs.

Remarque: Assurez-vous de faire un clic droit sur ***votre_xml_layou*** `(R.layout.your_xml_layou`), sinon le *menu Générer* ne contiendra pas d'option d'injecteur Butterknife.

```java
/**
 * Main UI for setting up GridWichterle.
 *
 * @author Michal Matl (michal.matl@inmite.eu)
 */
public class SettingsActivity extends FragmentActivity {

    private Config mConfig;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_settings);
        ButterKnife.inject(this);

        Intent intent = new Intent(this, GridOverlayService.class);
        startService(intent);

        setupViews();
    }
```

Lien: Plugin Jetbrains Android ButterKnife Zelezny

Lire Couteau à beurre en ligne: https://riptutorial.com/fr/android/topic/1072/couteau-a-beurre

Chapitre 75: Création de superposition Windows (toujours visible)

Examples

Popup overlay

Afin de placer votre vue au-dessus de chaque application, vous devez attribuer votre vue au gestionnaire de fenêtres correspondant. Pour cela, vous devez disposer de l'autorisation d'alerte système, qui peut être demandée en ajoutant la ligne suivante à votre fichier manifeste:

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

Remarque: Si votre application est détruite, votre vue sera supprimée du gestionnaire de fenêtres. Par conséquent, il est préférable de créer la vue et de l'affecter au gestionnaire de fenêtres par un service de premier plan.

Affectation d'une vue au WindowManager

Vous pouvez récupérer une instance du gestionnaire de fenêtres comme suit:

```
WindowManager mWindowManager = (WindowManager)
mContext.getSystemService(Context.WINDOW_SERVICE);
```

Pour définir la position de votre vue, vous devez créer des paramètres de disposition comme suit:

```
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;
```

Vous pouvez maintenant affecter votre vue avec les paramètres de mise en page créés à l'instance du gestionnaire de fenêtres comme suit:

```
mWindowManager.addView(yourView, mLayoutParams);
```

Voila! Votre vue a été placée avec succès sur toutes les autres applications.

Note: Votre vue ne sera pas placée sur le protège-clavier.

Accorder la permission SYSTEM_ALERT_WINDOW sur Android 6.0 et

supérieur

Depuis Android 6.0, cette autorisation doit accorder dynamiquement,

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

Lancer l'autorisation ci-dessous a refusé l'erreur sur 6.0,

```
Caused by: android.view.WindowManager$BadTokenException: Unable to add window
android.view.ViewRootImpl$W@86fb55b -- permission denied for this window type
```

Solution :-

Demander l'autorisation de superposition comme ci-dessous,

```
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);
}
```

Vérifier le résultat,

```
@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...
        }
    }
}
```

Lire Création de superposition Windows (toujours visible) en ligne:
https://riptutorial.com/fr/android/topic/6214/creation-de-superposition-windows--toujours-visible-

Chapitre 76: Création de vos propres bibliothèques pour les applications Android

Examples

Création d'un projet de bibliothèque

Pour créer une librairie, vous devez utiliser `File -> New -> New Module -> Android Library` . Ceci créera un projet de bibliothèque de base.

Lorsque cela est fait, vous devez avoir un projet configuré de la manière suivante:

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

Votre fichier `settings.gradle` doit contenir les éléments suivants:

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

Votre `[library root directory]` doit contenir les éléments suivants:

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

Votre fichier `build.gradle` "application" doit contenir les éléments suivants:

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

android {
    compileSdkVersion 23
    buildToolsVersion "23.0.2"

    defaultConfig {
        minSdkVersion 14
        targetSdkVersion 23
```

```
    }
}
```

Avec cela, votre projet devrait fonctionner correctement!

Utiliser la bibliothèque dans le projet en tant que module

Pour utiliser la bibliothèque, vous devez l'inclure en tant que dépendance avec la ligne suivante:

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

Créer une bibliothèque disponible sur Jitpack.io

Procédez comme suit pour créer la bibliothèque:

1. Créez un compte GitHub.
2. Créez un dépôt Git contenant votre projet de bibliothèque.
3. Modifiez le fichier `build.gradle` de votre projet de `build.gradle` en ajoutant le code suivant:

   ```
   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
   }
   ```

   Assurez-vous de valider / pousser les modifications ci-dessus sur GitHub.

4. Créez une version du code actuel sur Github.
5. Exécutez `gradlew install` sur votre code.

6. Votre bibliothèque est maintenant disponible par la dépendance suivante:

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

Lire Création de vos propres bibliothèques pour les applications Android en ligne: https://riptutorial.com/fr/android/topic/4118/creation-de-vos-propres-bibliotheques-pour-les-applications-android

Chapitre 77: Création de vues personnalisées

Examples

Création de vues personnalisées

Si vous avez besoin d'une vue totalement personnalisée, vous devrez sous-classer `View` (la super-classe de toutes les vues Android) et fournir vos `onMeasure(...)` dimensionnement personnalisées (`onMeasure(...)`) et de dessin (`onDraw(...)`):

1. **Créez votre squelette de vue personnalisée:** il s'agit essentiellement de la même chose pour chaque vue personnalisée. Ici, nous créons le squelette d'une vue personnalisée qui peut dessiner un smiley, appelé `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. **Initialiser vos peintures:** les objets `Paint` sont les pinceaux de votre canevas virtuel définissant la manière dont vos objets géométriques sont rendus (par exemple, style de couleur, de remplissage et de trait, etc.). Nous créons ici deux `Paint` , une peinture jaune remplie pour le cercle et une peinture noire pour les yeux et la bouche:

```
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. **Implémentez votre propre `onMeasure(...)` :** ceci est nécessaire pour que les mises en page parent (par exemple, `FrameLayout`) puissent aligner correctement votre vue personnalisée. Il fournit un ensemble de `measureSpecs` que vous pouvez utiliser pour déterminer la hauteur et la largeur de votre vue. Ici, nous créons un carré en nous assurant que la hauteur et la largeur sont les mêmes:

```
@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);
}
```

   Notez que `onMeasure(...)` doit contenir au moins un appel à `setMeasuredDimension(..)` sinon votre vue personnalisée tombera en panne avec une `IllegalStateException` .

4. **Implémentez votre propre `onSizeChanged(...)` :** cela vous permet de saisir la hauteur et la largeur actuelles de votre vue personnalisée pour ajuster correctement votre code de rendu. Ici, nous calculons simplement notre centre et notre rayon:

```
@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. **Implémentez votre propre `onDraw(...)` :** c'est là que vous implémentez le rendu réel de votre vue. Il fournit un objet `Canvas` lequel vous pouvez dessiner (voir la documentation officielle `Canvas` pour toutes les méthodes de dessin disponibles).

```
@Override
protected void onDraw(Canvas canvas) {
    // draw face
    canvas.drawCircle(mCenterX, mCenterY, mRadius, mCirclePaint);
    // draw eyes
    float eyeRadius = mRadius / 5f;
    float eyeOffsetX = mRadius / 3f;
    float eyeOffsetY = mRadius / 3f;
    canvas.drawCircle(mCenterX - eyeOffsetX, mCenterY - eyeOffsetY, eyeRadius,
mEyeAndMouthPaint);
    canvas.drawCircle(mCenterX + eyeOffsetX, mCenterY - eyeOffsetY, eyeRadius,
```

```
mEyeAndMouthPaint);
    // draw mouth
    float mouthInset = mRadius /3f;
    mArcBounds.set(mouthInset, mouthInset, mRadius * 2 - mouthInset, mRadius * 2 -
mouthInset);
    canvas.drawArc(mArcBounds, 45f, 90f, false, mEyeAndMouthPaint);
}
```

6. **Ajouter votre vue personnalisée à une mise en page:** la vue personnalisée peut désormais être incluse dans tous les fichiers de mise en page dont vous disposez. Ici, on l'enveloppe simplement dans un `FrameLayout` :

```
<FrameLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <com.example.app.SmileyView
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
</FrameLayout>
```

Notez qu'il est recommandé de créer votre projet une fois le code de la vue terminé. Sans le construire, vous ne pourrez pas voir la vue sur un écran d'aperçu dans Android Studio.

Après avoir tout réuni, vous devriez être accueilli avec l'écran suivant après le lancement de l'activité contenant la mise en page ci-dessus:

Ajout d'attributs à des vues

Les vues personnalisées peuvent également prendre des attributs personnalisés pouvant être

utilisés dans les fichiers de ressources de présentation Android. Pour ajouter des attributs à votre vue personnalisée, procédez comme suit:

1. **Définissez le nom et le type de vos attributs:** cela se fait à l'intérieur de `res/values/attrs.xml` (créez-le si nécessaire). Le fichier suivant définit un attribut de couleur pour la couleur du visage de notre smiley et un attribut enum pour l'expression du smiley:

```
<resources>
    <declare-styleable name="SmileyView">
        <attr name="smileyColor" format="color" />
        <attr name="smileyExpression" format="enum">
            <enum name="happy" value="0"/>
            <enum name="sad" value="1"/>
        </attr>
    </declare-styleable>
    <!-- attributes for other views -->
</resources>
```

2. **Utilisez vos attributs dans votre mise en page:** cela peut être fait à l'intérieur de tous les fichiers de mise en page qui utilisent votre vue personnalisée. Le fichier de mise en page suivant crée un écran avec un smiley jaune heureux:

```
<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>
```

   Conseil: Les attributs personnalisés ne fonctionnent pas avec les `tools:` préfixe dans Android Studio 2.1 et versions antérieures (et éventuellement dans les futures versions). Dans cet exemple, en remplaçant `app:smileyColor` par des `tools:smileyColor` `smileyColor` être défini pendant l'exécution ou au moment du design.

3. **Lisez vos attributs:** cela se fait à l'intérieur du code source de votre vue personnalisée. L'extrait suivant de `SmileyView` montre comment les attributs peuvent être extraits:

```
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. **(Facultatif) Ajoutez le style par défaut: pour** ce faire, ajoutez un style avec les valeurs par défaut et chargez-le dans votre vue personnalisée. Le style smiley par défaut suivant représente un style jaune heureux:

```
<!-- styles.xml -->
<style name="DefaultSmileyStyle">
    <item name="smileyColor">#ffff00</item>
    <item name="smileyExpression">happy</item>
</style>
```

   Ce qui est appliqué dans notre `SmileyView` en l'ajoutant comme dernier paramètre de l'appel à `obtainStyledAttributes` (voir le code à l'étape 3):

```
TypedArray a = context.obtainStyledAttributes(attrs, R.styleable.SmileyView,
defStyleAttr, R.style.DefaultSmileyViewStyle);
```

   Notez que toute valeur d'attribut définie dans le fichier de mise en forme gonflé (voir le code à l'étape 2) remplacera les valeurs correspondantes du style par défaut.

5. **(Facultatif) Fournissez des styles à l'intérieur des thèmes:** cela se fait en ajoutant un nouvel attribut de référence de style qui peut être utilisé dans vos thèmes et en fournissant un style pour cet attribut. Ici, nous `smileyStyle` simplement notre attribut de référence `smileyStyle` :

```
<!-- attrs.xml -->
<attr name="smileyStyle" format="reference" />
```

   Nous fournissons ensuite un style dans notre thème d'application (ici, nous réutilisons simplement le style par défaut de l'étape 4):

```
<!-- themes.xml -->
<style name="AppTheme" parent="AppBaseTheme">
    <item name="smileyStyle">@style/DefaultSmileyStyle</item>
</style>
```

Création d'une vue composée

Une **vue composée** est un `ViewGroup` personnalisé traité comme une vue unique par le code du programme environnant. Un tel ViewGroup peut être vraiment utile dans une conception de type DDD , car il peut correspondre à un agrégat, dans cet exemple, un contact. Il peut être réutilisé partout où ce contact est affiché.

Cela signifie que le code de contrôleur environnant, une activité, un fragment ou un adaptateur, peut simplement transmettre l'objet de données à la vue sans le séparer en plusieurs widgets d'interface utilisateur.

Cela facilite la réutilisation du code et permet une meilleure conception selon les principes SOLID .

La mise en page XML

C'est généralement là que vous commencez. Vous avez un bit XML existant que vous vous retrouvez en train de réutiliser, peut-être en tant que `<include/>` . Extrayez-le dans un fichier XML distinct et enveloppez-le dans un élément `<merge>` :

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

        <ImageView
            android:id="@+id/photo"
            android:layout_width="48dp"
            android:layout_height="48dp"
            android:layout_alignParentRight="true" />

        <TextView
            android:id="@+id/name"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_toLeftOf="@id/photo" />

        <TextView
            android:id="@+id/phone_number"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_below="@id/name"
            android:layout_toLeftOf="@id/photo" />
</merge>
```

Ce fichier XML fonctionne parfaitement dans l'éditeur de disposition d'Android Studio. Vous pouvez le traiter comme n'importe quelle autre mise en page.

Le composé ViewGroup

Une fois que vous avez le fichier XML, créez le groupe de vues personnalisé.

```
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);
        }
    }
}
```

La méthode `init(Context, AttributeSet)` permet de lire tous les attributs XML personnalisés, comme expliqué dans Ajout d'attributs aux vues .

Avec ces pièces en place, vous pouvez l'utiliser dans votre application.

Utilisation en XML

Voici un exemple `fragment_contact_info.xml` qui montre comment placer un seul ContactView au-dessus d'une liste de messages:

```
<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>
```

Utilisation dans le code

Voici un exemple `RecyclerView.Adapter` qui affiche une liste de contacts. Cet exemple illustre à quel point le code du contrôleur est plus propre lorsqu'il est totalement exempt de manipulation de View.

```
package myapp;

import android.content.Context;
import android.support.v7.widget.RecyclerView;
import android.view.ViewGroup;

public class ContactsAdapter extends RecyclerView.Adapter<ContactsViewHolder> {

    private final Context context;

    public ContactsAdapter(final Context context) {
        this.context = context;
    }

    @Override
```

```
    public ContactsViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        ContactView v = new ContactView(context); // <--- this
        return new ContactsViewHolder(v);
    }

    @Override
    public void onBindViewHolder(ContactsViewHolder holder, int position) {
        Contact contact = this.getItem(position);
        holder.setContact(contact);  // <--- this
    }

    static class ContactsViewHolder extends RecyclerView.ViewHolder {

        public ContactsViewHolder(ContactView itemView) {
            super(itemView);
        }

        public void setContact(Contact contact) {
            ((ContactView) itemView).setContact(contact); // <--- this
        }
    }
}
```

Conseils de performance CustomView

Ne pas allouer de nouveaux objets dans **onDraw**

```
@Override
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    Paint paint = new Paint(); //Do not allocate here
}
```

Au lieu de dessiner des tirables en toile ...

```
drawable.setBounds(boundsRect);

drawable.draw(canvas);
```

Utilisez un bitmap pour un dessin plus rapide:

```
canvas.drawBitmap(bitmap, srcRect, boundsRect, paint);
```

Ne redessinez pas la vue entière pour ne mettre à jour qu'une petite partie. Au lieu de cela, redessinez la partie spécifique de la vue.

```
invalidate(boundToBeRefreshed);
```

Si votre vue effectue une animation continue, par exemple une montre montrant chaque seconde, arrêtez au moins l'animation à `onStop()` de l'activité et redémarrez-la sur `onStart()` de l'activité.

Ne faites aucun calcul dans la méthode `onDraw` d'une vue, vous devriez plutôt terminer de dessiner avant d'appeler `invalidate()` . En utilisant cette technique, vous pouvez éviter de laisser tomber

des images dans votre vue.

Rotations

Les opérations de base d'une vue sont translater, faire pivoter, etc. Presque tous les développeurs ont rencontré ce problème lorsqu'ils utilisent des bitmap ou des dégradés dans leur vue personnalisée. Si la vue affiche une vue pivotée et que le bitmap doit être pivoté dans cette vue personnalisée, beaucoup d'entre nous penseront que cela va coûter cher. Beaucoup pensent que tourner un bitmap est très coûteux car pour ce faire, vous devez traduire la matrice de pixels du bitmap. Mais la vérité est que ce n'est pas si difficile! Au lieu de faire pivoter le bitmap, faites simplement pivoter la toile elle-même!

```
// 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.
```

Vue composée pour SVG / VectorDrawable comme drawableRight

Le motif principal pour développer cette vue composée est que, sous 5.0, les périphériques ne supportent pas svg dans drawable dans TextView / EditText. Un autre avantage est que nous pouvons définir la `height` et la `width` de `drawableRight` dans `EditText` . Je l'ai séparé de mon projet et créé dans un module séparé.

Nom du module: custom_edit_drawable (nom abrégé pour prefix- c_d_e)

"c_d_e_" préfixe à utiliser pour que les ressources du module d'application ne les remplacent pas par erreur. Exemple: le préfixe "abc" est utilisé par Google dans la bibliothèque de support.

build.gradle

```
dependencies {
   compile 'com.android.support:appcompat-v7:25.3.1'
}
```

utiliser AppCompat> = 23

Fichier de mise en page: 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>
```

Attributs personnalisés: 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();
        }
    }
}
```

Exemple: Comment utiliser la vue ci-dessus

Mise en page: 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>
```

Activité: 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);
    }
}
```

Répondre aux événements tactiles

De nombreuses vues personnalisées doivent accepter l'interaction de l'utilisateur sous la forme d'événements tactiles. Vous pouvez accéder aux événements tactiles en `onTouchEvent` . Vous pouvez filtrer plusieurs actions. Les principaux sont

- `ACTION_DOWN` : Ceci est déclenché une fois lorsque votre doigt touche pour la première fois la vue.
- `ACTION_MOVE` : Ceci est appelé chaque fois que votre doigt se déplace un peu sur la vue. Il est appelé à plusieurs reprises.
- `ACTION_UP` : Ceci est la dernière action à appeler lorsque vous retirez votre doigt de l'écran.

Vous pouvez ajouter la méthode suivante à votre vue, puis observer la sortie du journal lorsque vous touchez et déplacez votre doigt autour de votre vue.

```
@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;
}
```

Lectures complémentaires:

- Documentation officielle Android: Répondre aux événements tactiles

Lire Création de vues personnalisées en ligne:
https://riptutorial.com/fr/android/topic/1446/creation-de-vues-personnalisees

Chapitre 78: Créer des ROM personnalisées Android

Examples

Préparez votre machine pour la construction!

Avant de pouvoir construire quelque chose, vous devez préparer votre machine pour la construction. Pour cela, vous devez installer beaucoup de bibliothèques et de modules. La distribution Linux la plus recommandée est Ubuntu, donc cet exemple se concentrera sur l'installation de tout ce qui est nécessaire sur Ubuntu.

Installation de Java

Tout d'abord, ajoutez l'archive PPA (Personal Package Archive) suivante: `sudo apt-add-repository ppa:openjdk-r/ppa` .

Ensuite, mettez à jour les sources en exécutant: `sudo apt-get update` .

Installation de dépendances supplémentaires

Toutes les dépendances supplémentaires requises peuvent être installées à l'aide de la commande suivante:

```
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
```

Préparer le système pour le développement

Maintenant que toutes les dépendances sont installées, préparons le système pour le développement en exécutant:

```
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
```

Enfin, configurons le cache et le repo par les commandes suivantes:

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

Veuillez noter que nous pouvons également réaliser cette configuration en exécutant les scripts automatisés réalisés par Akhil Narang (*akhilnarang*), l'un des responsables de Resurrection Remix OS . Ces scripts peuvent être trouvés sur GitHub .

Lire Créer des ROM personnalisées Android en ligne:
https://riptutorial.com/fr/android/topic/9212/creer-des-rom-personnalisees-android

Chapitre 79: Créer un écran de démarrage

Remarques

Le premier exemple (un écran d'accueil de base) n'est pas le moyen le plus efficace de le gérer. En tant que tel, il s'agit d'un écran de base.

Examples

Un écran de base

Un écran de démarrage est comme toute autre activité, mais il peut gérer tous vos besoins de démarrage en arrière-plan. Exemple:

Manifeste:

```
<?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>
```

Maintenant, notre écran d'accueil sera appelé comme première activité.

Voici un exemple d'écran de démarrage qui gère également certains éléments d'application critiques:

```
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();
```

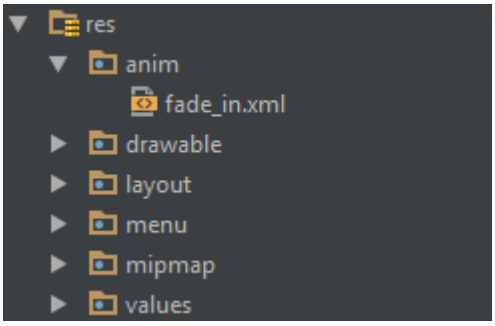
```
        finish();
    }

}
```

Écran de démarrage avec animation

Cet exemple montre un écran de démarrage simple mais efficace avec une animation pouvant être créée à l'aide d'Android Studio.

Etape 1: Créer une animation

Créez un nouveau répertoire nommé *anim* dans le répertoire *res* . Cliquez-droit dessus et créez un nouveau fichier de *ressources d'animation* nommé *fade_in.xml* :



Ensuite, placez le code suivant dans le fichier *fade_in.xml* :

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android" android:fillAfter="true" >
    <alpha
        android:duration="1000"
        android:fromAlpha="0.0"
        android:interpolator="@android:anim/accelerate_interpolator"
        android:toAlpha="1.0" />
</set>
```

Étape 2: Créer une activité

Créez une *activité vide à l'* aide d'Android Studio nommé `Splash` . Ensuite, placez-y le code suivant:

```
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);
    }
}
```

Ensuite, placez le code suivant dans le fichier de mise en page:

```
<?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>
```

Étape 3: Remplacez le lanceur par défaut

Transformez votre activité `Splash` en un lanceur en ajoutant le code suivant au fichier *AndroidManifest* :

```
<activity
    android:name=".Splash"
    android:theme="@style/AppTheme.NoActionBar">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
```

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

Ensuite, supprimez l'activité de lanceur par défaut en supprimant le code suivant du fichier *AndroidManifest* :

```
<intent-filter>
    <action android:name="android.intent.action.MAIN" />

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

Lire Créer un écran de démarrage en ligne: https://riptutorial.com/fr/android/topic/9316/creer-un-ecran-de-demarrage

Chapitre 80: Créer une classe Singleton pour le message Toast

Introduction

Les messages Toast constituent le moyen le plus simple de fournir un retour à l'utilisateur. Par défaut, Android fournit un message de couleur grise où nous pouvons définir le message et la durée du message. Si nous avons besoin de créer plus de messages de toast personnalisables et réutilisables, nous pouvons les implémenter nous-mêmes en utilisant une disposition personnalisée. Plus important encore, lorsque nous l'implémentons, l'utilisation du motif de conception Singelton facilitera la maintenance et le développement de la classe de message de toast personnalisée.

Syntaxe

- Toast Toast (Contexte Contexte)
- void setDuration (int duration)
- void setGravity (int gravity, int xOffset, int yOffset)
- void setView (Voir la vue)
- annuler le spectacle ()

Paramètres

Paramètre	détails
le contexte	Contexte pertinent qui doit afficher votre message de toast. Si vous l'utilisez dans l'activité, passez le mot-clé "this" ou Si vous utilisez la commande fring comme "getActivity ()".
vue	Créez une vue personnalisée et transmettez cet objet à cette vue.
la gravité	Passez la position de gravité du grille-pain. Toutes les positions ont été ajoutées sous la classe Gravity en tant que variables statiques. Les positions les plus courantes sont Gravity.TOP, Gravity.BOTTOM, Gravity.LEFT, Gravity.RIGHT.
xOffset	Décalage horizontal du message toast.
yOffset	Décalage vertical du message toast.
durée	Durée du spectacle de pain grillé. Nous pouvons définir Toast.LENGTH_SHORT ou Toast.LENGTH_LONG

Remarques

Le message Toast est un moyen simple de fournir une rétroaction à l'utilisateur à propos de quelque chose qui se passe. Si vous avez besoin d'un moyen plus avancé pour donner votre avis, vous pouvez utiliser des boîtes de dialogue ou un snack.

Pour plus de détails sur le message toast, consultez cette documentation.
https://developer.android.com/reference/android/widget/Toast.html

Examples

Créer sa propre classe de singleton pour les massages de pain grillé

Voici comment créer votre propre classe singleton pour les messages toast. Si votre application doit afficher les messages de réussite, d'avertissement et de danger pour différents cas d'utilisation, vous pouvez utiliser cette classe après l'avoir modifiée selon vos propres spécifications.

```
    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();
        }
    }
```

Lire Créer une classe Singleton pour le message Toast en ligne:
https://riptutorial.com/fr/android/topic/10843/creer-une-classe-singleton-pour-le-message-toast

Chapitre 81: Cueilleurs de date et d'heure

Examples

Matériau DatePicker

ajouter ci-dessous les dépendances au fichier `build.gradle` dans la section dépendances. (c'est une bibliothèque non officielle pour le sélecteur de date)

```
  compile 'com.wdullaer:materialdatetimepicker:2.3.0'
```

Maintenant, nous devons ouvrir `DatePicker` sur l'événement Click du bouton.

Donc, créez un bouton sur le fichier XML comme ci-dessous.

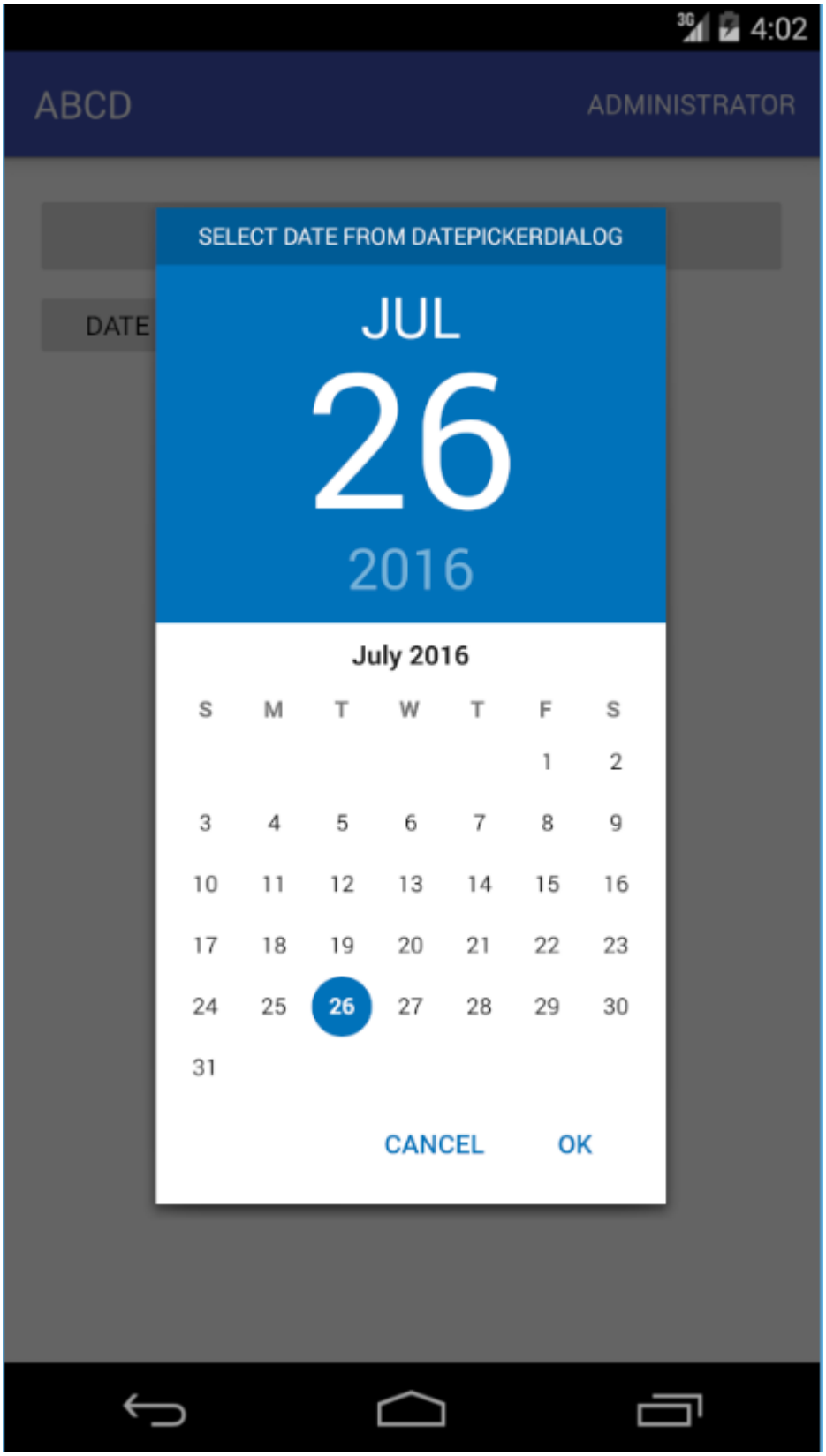
```
     <Button
            android:id="@+id/dialog_bt_date"
            android:layout_below="@+id/resetButton"
            android:layout_width="wrap_content"
            android:layout_height="40dp"
            android:textColor="#FF000000"
            android:gravity="center"
            android:text="DATE"/>
```

et dans MainActivity, utilisez cette méthode.

```
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;
    }

}
```

Sortie:



Dialogue de sélecteur de date

C'est une boîte de dialogue qui invite l'utilisateur à sélectionner la date à l'aide de `DatePicker` . La boîte de dialogue nécessite un contexte, une année initiale, un mois et un jour pour afficher la boîte de dialogue avec la date de début. Lorsque l'utilisateur sélectionne la date qu'il `DatePickerDialog.OnDateSetListener` via `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();

    }
```

Veuillez noter que le mois est un int commençant de 0 pour janvier à 11 pour décembre

Lire Cueilleurs de date et d'heure en ligne: https://riptutorial.com/fr/android/topic/2836/cueilleurs-de-date-et-d-heure

Chapitre 82: Cycle de vie de l'interface utilisateur

Examples

Enregistrement de données lors du découpage de la mémoire

```
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);
    }
}
```

Explication

Alors, qu'est-ce qui se passe ici?

Le système Android s'efforcera toujours d'effacer autant de mémoire que possible. Ainsi, si votre activité est en arrière-plan et qu'une autre activité de premier plan exige son partage, le système Android appellera `onTrimMemory()` sur votre activité.

Mais cela ne signifie pas que toutes vos propriétés devraient disparaître. Ce que vous devez faire est de les enregistrer dans un objet Bundle. Les objets groupés sont bien mieux gérés en termes de mémoire. À l'intérieur d'un ensemble, chaque objet est identifié par une séquence de texte unique - dans l'exemple ci-dessus, la variable de valeur entière `mArg` est maintenue sous le nom de référence `EXAMPLE_ARG` . Et lorsque l'activité est recréée, extrayez vos anciennes valeurs de l'objet Bundle au lieu de les recréer à partir de zéro

Lire Cycle de vie de l'interface utilisateur en ligne:
https://riptutorial.com/fr/android/topic/3440/cycle-de-vie-de-l-interface-utilisateur

Chapitre 83: Dague 2

Syntaxe

- @Module
- @Component (dépendances = {OtherComponent.class}, modules = {ModuleA.class, ModuleB.class})
- DaggerMyComponent.create ()
- DaggerMyComponent.builder (). MyModule (newMyModule ()). Create ()

Remarques

Ne pas confondre avec le poignard carré, le prédécesseur du poignard 2.

Examples

Configuration du composant pour l'injection d'application et d'activité

`AppComponent` base qui dépend d'un seul `AppModule` pour fournir des objets singleton à l'échelle de l'application.

```
@Singleton
@Component(modules = AppModule.class)
public interface AppComponent {

    void inject(App app);

    Context provideContext();

    Gson provideGson();
}
```

Un module à utiliser avec `AppComponent` qui fournira ses objets singleton, par exemple une instance de `Gson` à réutiliser dans toute l'application.

```
@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;
    }
}
```

Une application sous-classée pour configurer dagger et le composant singleton.

```
public class App extends Application {

    @Inject
    AppComponent mAppComponent;

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

        DaggerAppComponent.builder().appModule(new AppModule(this)).build().inject(this);
    }

    public AppComponent getAppComponent() {
        return mAppComponent;
    }
}
```

Maintenant, un composant de portée d'activité qui dépend de `AppComponent` pour accéder aux objets singleton.

```
@ActivityScope
@Component(dependencies = AppComponent.class, modules = ActivityModule.class)
public interface MainActivityComponent {

    void inject(MainActivity activity);
}
```

Et un `ActivityModule` réutilisable qui fournira des dépendances de base, comme un `FragmentManager`

```
@Module
public class ActivityModule {

    private final AppCompatActivity mActivity;

    public ActivityModule(AppCompatActivity activity) {
        mActivity = activity;
    }

    @ActivityScope
    public AppCompatActivity provideActivity() {
        return mActivity;
    }


    @ActivityScope
    public FragmentManager provideFragmentManager(AppCompatActivity activity) {
        return activity.getSupportFragmentManager();
```

```
    }
}
```

En mettant tout en place, nous sommes en mesure d'injecter notre activité et d'être sûr d'utiliser la même application `Gson` partout!

```
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);
    }
}
```

Scopes personnalisés

```
@Scope
@Documented
@Retention(RUNTIME)
public @interface ActivityScope {
}
```

Les portées ne sont que des annotations et vous pouvez en créer de nouvelles si nécessaire.

Constructeur Injection

Les classes sans dépendances peuvent facilement être créées par dagger.

```
public class Engine {

    @Inject // <-- Annotate your constructor.
    public Engine() {
    }
}
```

Cette classe peut être fournie par *n'importe quel* composant. Il n'a *pas de dépendance* et n'a *pas de portée* . Il n'y a pas d'autre code nécessaire.

Les dépendances sont déclarées comme paramètres dans le constructeur. Dagger appellera le constructeur et fournira les dépendances, tant que ces dépendances peuvent être fournies.

```
public class Car {
```

```
    private Engine engine;

    @Inject
    public Car(Engine engine) {
        this.engine = engine;
    }
}
```

Cette classe peut être fournie par chaque composant *si* ce composant peut également fournir toutes ses dépendances - `Engine` dans ce cas. Comme le `Engine` peut également être injecté par un constructeur, *tout* composant peut fournir une `Car` .

Vous pouvez utiliser l'injection de constructeur chaque fois que toutes les dépendances peuvent être fournies par le composant. Un composant peut fournir une dépendance, si

- il peut le créer en utilisant l'injection de constructeur
- un module du composant peut le fournir
- il peut être fourni par le composant parent (s'il s'agit d'un `@Subcomponent`)
- il peut utiliser un objet exposé par un composant dont il dépend (dépendances des composants)

Utiliser @Subcomponent au lieu de @Component (dépendances = {...})

```
@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);
    }
}
```

Comment ajouter Dagger 2 dans build.gradle

Depuis la sortie de Gradle 2.2, l'utilisation du plugin android-apt n'est plus utilisée. La méthode suivante pour configurer Dagger 2 doit être utilisée. Pour les anciennes versions de Gradle, utilisez la méthode précédente indiquée ci-dessous.

Pour 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'
}
```

Pour Gradle <2.2

Pour utiliser Dagger 2, il faut ajouter le plugin `android-apt` , ajouter ceci à la version build.gradle:

```
buildscript {
    dependencies {
        classpath 'com.android.tools.build:gradle:2.1.0'
        classpath 'com.neenbedankt.gradle.plugins:android-apt:1.8'
    }
}
```

Ensuite, le build.gradle du module d'application doit contenir:

```
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}"
}
```

Référence: https://github.com/codepath/android_guides/wiki/Dependency-Injection-with-Dagger-2

Création d'un composant à partir de plusieurs modules

Dagger 2 prend en charge la création d'un composant à partir de plusieurs modules. Vous pouvez créer votre composant de cette façon:

```
@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);
}
```

Les deux modules de référence `GeneralPurposeModule` et `SpecificModule` peuvent alors être implémentés comme suit:

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;
    }
}
```

Pendant la phase d'injection de dépendance, le composant récupérera les objets des deux modules en fonction des besoins.

Cette approche est très utile en termes de *modularité* . Dans l'exemple, il existe un module général utilisé pour instancier des composants tels que l'objet `Retrofit` (utilisé pour gérer la communication réseau) et un `PropertiesReader` (chargé de gérer les fichiers de configuration). Il existe également un module spécifique qui gère l'instanciation de contrôleurs et de classes de service spécifiques par rapport à ce composant d'application spécifique.

Lire Dague 2 en ligne: https://riptutorial.com/fr/android/topic/3088/dague-2

Chapitre 84: Date / heure localisée dans Android

Remarques

Il est recommandé d'utiliser des méthodes de la classe DateUtils afin de formater les dates qui prennent en compte les paramètres régionaux, c'est-à-dire qui prennent en compte les préférences de l'utilisateur (par exemple, formats d'horloge 12h / 24h). Ces méthodes sont les plus appropriées pour les dates affichées à l'utilisateur.

Pour les représentations de date entièrement personnalisées, il est recommandé d'utiliser la classe SimpleDateFormat , car elle permet de contrôler entièrement tous les éléments de date.

Examples

Format de date personnalisé personnalisé avec DateUtils.formatDateTime ()

DateUtils.formatDateTime () vous permet de fournir une heure, et en fonction des indicateurs que vous fournissez, il crée une chaîne datetime localisée. Les drapeaux vous permettent d'indiquer s'il faut inclure des éléments spécifiques (comme le jour de la semaine).

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

formatDateTime () s'occupe automatiquement des formats de date appropriés.

Formatage de date / heure standard dans Android

Formatez une date:

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

Formatez une date et une heure. La date est au format abrégé, l'heure est au format long:

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

Date et heure entièrement personnalisées

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

```
String localizedDate = df.format(date)
```

Modèles couramment utilisés:

- HH: heure (0-23)
- hh: heure (1-12)
- a: marqueur AM / PM
- mm: minute (0-59)
- ss: second
- dd: jour en mois (1-31)
- MM: mois
- aaaa: année

Lire Date / heure localisée dans Android en ligne: https://riptutorial.com/fr/android/topic/6057/date--heure-localisee-dans-android

Chapitre 85: Décompresser le fichier dans Android

Examples

Décompresser le fichier

```
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;}
```

Lire Décompresser le fichier dans Android en ligne:
https://riptutorial.com/fr/android/topic/3927/decompresser-le-fichier-dans-android

Chapitre 86: Décorations RecyclerView

Syntaxe

- RecyclerView addItemDecoration (décoration de RecyclerView.ItemDecoration)
- RecyclerView addItemDecoration (RecyclerView.ItemDecoration decoration, int index)

Paramètres

Paramètre	Détails
décoration	la décoration de l'article à ajouter au `RecyclerView`
indice	l'index dans la liste des décorations pour ce `RecyclerView` . Ceci est l'ordre dans lequel `getItemOffset` et `onDraw` sont appelés. Les appels ultérieurs peuvent dépasser les précédents.

Remarques

Les décorations sont statiques

Étant donné que les décorations sont uniquement dessinées, il n'est pas possible d'ajouter des écouteurs de clics ou d'autres fonctionnalités de l'interface utilisateur.

Décorations multiples

L'ajout de plusieurs décorations à `RecyclerView` fonctionnera dans certains cas, mais il n'existe actuellement aucune API publique pour prendre en compte d'autres décorations possibles lors de la mesure ou du dessin. Vous pouvez obtenir les limites de vue ou les limites décorées de vue, les bornes décorées étant la somme de tous les décalages de décoration appliqués.

Autres sujets connexes:

RecyclerView
RecyclerView onClickListeners

Javadoc officiel

https://developer.android.com/reference/android/support/v7/widget/RecyclerView.ItemDecoration.html

Examples

Dessiner un séparateur

Cela dessine une ligne au bas de chaque vue, mais la dernière à agir comme séparateur entre les éléments.

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

Marges par élément avec ItemDecoration

Vous pouvez utiliser `RecyclerView.ItemDecoration` pour placer des marges supplémentaires autour de chaque élément dans RecyclerView. Cela peut, dans certains cas, nettoyer à la fois l'implémentation de votre adaptateur et le code XML de votre vue d'élément.

```
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);
    }
}
```

Pour activer la décoration, ajoutez-la simplement à votre RecyclerView:

```
// in your onCreate()
RecyclerView rv = (RecyclerView) findItemById(R.id.myList);
```

```
rv.addItemDecoration(new MyItemDecoration(context));
```

Ajouter un diviseur à RecyclerView

Tout d'abord, vous devez créer une classe qui étend `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);
    }
}

}
```

Ajoutez `divider_blue.xml` à votre dossier pouvant être dessiné:

```
<?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>
```

Ensuite, utilisez-le comme:

```
recyclerView.addItemDecoration(new SimpleBlueDivider(context));
```

Le résultat sera comme:


28% 14:20
Stackoverflow :)
Stackoverflow :)
Stackoverflow :)
Stackoverflow :)
Stackoverflow :)
Stackoverflow :)
Stackoverflow :)

Cette image est juste un exemple du fonctionnement des diviseurs, si vous souhaitez suivre les spécifications de conception des matériaux lors de l'ajout de séparateurs, consultez ce lien: diviseurs et remerciements @Brenden Kromhout en fournissant un lien.

Comment ajouter des diviseurs en utilisant et diviserItemDecoration

Le `DividerItemDecoration` est un `RecyclerView.ItemDecoration` qui peut être utilisé comme séparateur entre les éléments.

```
DividerItemDecoration mDividerItemDecoration = new DividerItemDecoration(context,
             mLayoutManager.getOrientation());
recyclerView.addItemDecoration(mDividerItemDecoration);
```

Il prend en charge les deux orientations en utilisant `DividerItemDecoration.VERTICAL` et `DividerItemDecoration.HORIZONTAL` .

ItemOffsetDecoration pour GridLayoutManager dans RecycleView

L'exemple suivant aidera à donner un espace égal à un élément dans GridLayout.

ItemOffsetDecoration.java

```
public class ItemOffsetDecoration extends RecyclerView.ItemDecoration {

    private int mItemOffset;

    private int spanCount = 2;

    public ItemOffsetDecoration(int itemOffset) {
        mItemOffset = itemOffset;
    }

    public ItemOffsetDecoration(@NonNull Context context, @DimenRes int itemOffsetId) {
        this(context.getResources().getDimensionPixelSize(itemOffsetId));
    }

    @Override
    public void getItemOffsets(Rect outRect, View view, RecyclerView parent,
                               RecyclerView.State state) {
        super.getItemOffsets(outRect, view, parent, state);

        int position = parent.getChildLayoutPosition(view);

        GridLayoutManager manager = (GridLayoutManager) parent.getLayoutManager();

        if (position < manager.getSpanCount())
            outRect.top = mItemOffset;

        if (position % 2 != 0) {
            outRect.right = mItemOffset;
        }

        outRect.left = mItemOffset;
        outRect.bottom = mItemOffset;
    }
```

```
}
```

Vous pouvez appeler ItemDecoration comme ci-dessous le code.

```
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);
```

et exemple de décalage d'objet

```
<dimen name="item_offset">5dp</dimen>
```

Lire Décorations RecyclerView en ligne: https://riptutorial.com/fr/android/topic/506/decorations-recyclerview

Chapitre 87: Définir la valeur de pas (incrément) pour RangeSeekBar personnalisé

Introduction

Une personnalisation de la gamme Android RangeSeekBar proposée par Alex Florescu sur https://github.com/anothem/android-range-seek-bar

Il permet de définir une valeur de pas (incrément), lors du déplacement de la barre de recherche

Remarques

1- Ajouter l'attribut d'incrément dans attrs.xml

```
<attr name="increment" format="integer|float"/>
```

2- Définir une valeur par défaut dans RangeSeekBar.java et créer l'attribut également

```
private static final int DEFAULT_INCREMENT = 1;
private int increment;
```

3- Initier la valeur d'incrémentation dans le paramètre privé void init (Contexte contextuel, AttributeSet attrs)

```
if (attrs == null)
    increment = DEFAULT_INCREMENT;
else
    increment = a.getInt(R.styleable.RangeSeekBar_increment, DEFAULT_INCREMENT);
```

4- Définir la valeur d'incrémentation dans un void onDraw synchronisé protégé (@NonNull Canvas canvas)

Vous devrez remplacer la valeur minText et maxText. Donc au lieu de:

- minText = valueToString (getSelectedMinValue ());
- maxText = valueToString (getSelectedMaxValue ());

Vous aurez: 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 - Il ne vous reste plus qu'à l'utiliser. J'espère que cela aide

Examples

Définir une valeur de pas de 7

```
<RangeSeekBar
        android:id="@+id/barPrice"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        app:barHeight="0.2dp"
        app:barHeight2="4dp"
        app:increment="7"
        app:showLabels="false" />
```

Lire Définir la valeur de pas (incrément) pour RangeSeekBar personnalisé en ligne: https://riptutorial.com/fr/android/topic/8627/definir-la-valeur-de-pas--increment--pour-rangeseekbar-personnalise

Chapitre 88: Démarrer avec OpenGL ES 2.0+

Introduction

Cette rubrique concerne la configuration et l'utilisation d' **OpenGL ES 2.0+** sur Android. OpenGL ES est la norme pour les graphiques accélérés 2D et 3D sur les systèmes embarqués, y compris les consoles, les smartphones, les appareils et les véhicules.

Examples

Configuration de GLSurfaceView et OpenGL ES 2.0+

Pour utiliser OpenGL ES dans votre application, vous devez l'ajouter au manifeste:

```
<uses-feature android:glEsVersion="0x00020000" android:required="true"/>
```

Créez votre GLSurfaceView étendu:

```
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);
        }
    }
}
```

Ajoutez `MyGLSurfaceView` à votre mise en page:

```
<com.example.app.MyGLSurfaceView
    android:id="@+id/gles_renderer"
```

```
    android:layout_width="match_parent"
    android:layout_height="match_parent"/>
```

Pour utiliser une version plus récente d'OpenGL ES, changez simplement le numéro de version dans votre manifeste, dans l'importation statique et modifiez `setEGLContextClientVersion` .

Compiler et relier les shaders GLSL-ES à partir d'un fichier de ressources

Le dossier Assets est l'endroit le plus courant pour stocker vos fichiers shader GLSL-ES. Pour les utiliser dans votre application OpenGL ES, vous devez d'abord les charger dans une chaîne. Cette fonction crée une chaîne à partir du fichier de ressources:

```
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;
    }
}
```

Maintenant, vous devez créer une fonction qui compile un shader stocké dans un sting:

```
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;
}
```

Vous pouvez maintenant charger, compiler et lier vos shaders:

```
// Load shaders from file
String vertexShaderString = loadStringFromAssetFile(context, "your_vertex_shader.glsl");
```

```
String fragmentShaderString = loadStringFromAssetFile(context, "your_fragment_shader.glsl");

// Compile shaders
int vertexShader = compileShader(GL_VERTEX_SHADER, vertexShaderString);
int fragmentShader = compileShader(GL_FRAGMENT_SHADER, fragmentShaderString);

// Link shaders and create shader program
int shaderProgram = glCreateProgram();
glAttachShader(shaderProgram , vertexShader);
glAttachShader(shaderProgram , fragmentShader);
glLinkProgram(shaderProgram);

// Check for linking errors:
int linkStatus[] = new int[1];
glGetProgramiv(shaderProgram, GL_LINK_STATUS, linkStatus, 0);
if (linkStatus[0] != GL_TRUE) {
    String log = glGetProgramInfoLog(shaderProgram);

    Log.e(TAG,"Could not link shader program: ");
    Log.e(TAG, log);
}
```

S'il n'y a pas d'erreurs, votre programme shader est prêt à être utilisé:

```
glUseProgram(shaderProgram);
```

Lire Démarrer avec OpenGL ES 2.0+ en ligne:
https://riptutorial.com/fr/android/topic/8662/demarrer-avec-opengl-es-2-0plus

Chapitre 89: Des exceptions

Examples

NetworkOnMainThreadException

De la documentation :

> Exception levée lorsqu'une application tente d'effectuer une opération de mise en réseau sur son thread principal.
>
> Ceci est seulement lancé pour les applications ciblant le SDK Honeycomb ou supérieur. Les applications ciblant les versions antérieures du SDK sont autorisées à faire du réseautage sur leurs principaux threads de boucle d'événement, mais elles sont fortement déconseillées.

Voici un exemple de fragment de code qui peut provoquer cette exception:

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

Le code ci-dessus lancera `NetworkOnMainThreadException` pour les applications ciblant Honeycomb SDK (Android v3.0) ou supérieur car l'application tente d'effectuer une opération réseau sur le

thread principal.

Pour éviter cette exception, vos opérations réseau doivent toujours s'exécuter dans une tâche en arrière-plan via une tâche `AsyncTask` , `Thread` , `IntentService` , etc.

```
private class MyAsyncTask extends AsyncTask<String, Integer, Void> {

    @Override
    protected Void doInBackground(String[] params) {
        Uri.Builder builder = new Uri.Builder().scheme("http").authority("www.google.com");
        HttpURLConnection urlConnection = null;
        BufferedReader reader = null;
        URL url;
        try {
            url = new URL(builder.build().toString());
            urlConnection = (HttpURLConnection) url.openConnection();
            urlConnection.setRequestMethod("GET");
            urlConnection.connect();
        } catch (IOException e) {
            Log.e("TAG","Connection error", e);
        } finally{
            if (urlConnection != null) {
                urlConnection.disconnect();
            }
            if (reader != null) {
                try {
                    reader.close();
                } catch (final IOException e) {
                    Log.e("TAG", "Error closing stream", e);
                }
            }
        }

        return null;
    }
}
```

ActivityNotFoundException

C'est une `Exception` très commune. Cela provoque l'arrêt de votre application pendant le démarrage ou l'exécution de votre application. Dans le `LogCat` vous voyez le message:

```
android.content.ActivityNotFoundException : Unable to find explicit activity class;
have you declared this activity in your AndroidManifest.xml?
```

Dans ce cas, vérifiez si vous avez déclaré votre activité dans le fichier `AndroidManifest.xml` .

La manière la plus simple de déclarer votre `Activity` dans `AndroidManifest.xml` est la suivante:

```
<activity  android:name="com.yourdomain.YourStoppedActivity" />
```

OutOfMemoryError

Ceci est une erreur d'exécution qui se produit lorsque vous demandez une grande quantité de

mémoire sur le tas. Ceci est courant lors du chargement d'un bitmap dans un ImageView.

Vous avez des options:

1. Utilisez un grand tas d'applications

Ajoutez l'option "largeHeap" à la balise d'application dans votre fichier AndroidManifest.xml. Cela rendra plus de mémoire disponible pour votre application, mais ne résoudra probablement pas le problème.

```
<application largeHeap="true" ... >
```

2. Recyclez vos bitmaps

Après avoir chargé un bitmap, assurez-vous de le recycler et de libérer de la mémoire:

```
    if (bitmap != null && !bitmap.isRecycled())
       bitmap.recycle();
```

3. Charger des bitmaps échantillonnés en mémoire

Évitez de charger l'intégralité du bitmap en mémoire en échantillonnant une taille réduite, en utilisant BitmapOptions et inSampleSize.

Voir la documentation Android par exemple

DexException

```
com.android.dex.DexException: Multiple dex files define Lcom/example/lib/Class;
```

Cette erreur se produit car l'application, lors de la mise en package, trouve deux fichiers `.dex` qui définissent le même ensemble de méthodes.

Cela se produit généralement parce que l'application a accidentellement acquis 2 dépendances distinctes sur la même bibliothèque.

Par exemple, supposons que vous ayez un projet et que vous souhaitiez vous appuyer sur deux bibliothèques: `A` et `B` , chacune ayant ses propres dépendances. Si la bibliothèque `B` déjà une dépendance à la bibliothèque `A` , cette erreur sera renvoyée si la bibliothèque `A` est ajoutée au projet par elle-même. La bibliothèque de compilation `B` donnait déjà le jeu de code de `A` , donc lorsque le compilateur va regrouper la bibliothèque `A` , il trouve les méthodes de la bibliothèque `A` déjà empaquetées.

Pour résoudre ce problème, assurez-vous qu'aucune de vos dépendances ne soit ajoutée accidentellement deux fois de cette manière.

Exception non interceptée

Si vous souhaitez gérer des exceptions non interceptées, essayez de les récupérer toutes avec la méthode onCreate de votre classe Application:

```
public class MyApp extends Application {
    @Override
    public void onCreate() {
        super.onCreate();
        try {
            Thread
                .setDefaultUncaughtExceptionHandler(
                            new Thread.UncaughtExceptionHandler() {

                @Override
                public void uncaughtException(Thread thread, Throwable e) {
                    Log.e(TAG,
                            "Uncaught Exception thread: "+thread.getName()+"
                             "+e.getStackTrace());
                    handleUncaughtException (thread, e);
                }
            });
        } catch (SecurityException e) {
            Log.e(TAG,
                    "Could not set the Default Uncaught Exception Handler:"
                    +e.getStackTrace());
        }
    }

    private void handleUncaughtException (Thread thread, Throwable e){
        Log.e(TAG, "uncaughtException:");
        e.printStackTrace();
    }
}
```

Enregistrement de votre propre gestionnaire pour des exceptions inattendues

Voici comment vous pouvez réagir aux exceptions qui n'ont pas été détectées, comme le système standard "Application XYZ a planté"

```
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
    }
}
```

Enregistrez ensuite cette classe d'application dans votre fichier AndroidManifest.xml:

```
<application android:name="de.ioxp.arkmobile.MyApplication" >
```

Lire Des exceptions en ligne: https://riptutorial.com/fr/android/topic/112/des-exceptions

Chapitre 90: Des préférences partagées

Introduction

SharedPreferences permet de sauvegarder des données sur le disque sous la forme de paires **clé-valeur** .

Syntaxe

- **Méthode de contexte**
  - public SharedPreferences getSharedPreferences (nom de chaîne, mode int)
- **Méthode d'activité**
  - public SharedPreferences getPreferences ()
- **Méthodes SharedPreferences**
  - public SharedPreferences.Editor edit ()
  - booléen public contient ()
  - carte publique <String,?> getAll ()
  - boolean public getBoolean (clé de chaîne, booléen defValue)
  - float public getFloat (clé String, float defValue)
  - public int getInt (clé de chaîne, int defValue)
  - public long getLong (clé de chaîne, long defValue)
  - public String getString (clé de chaîne, String defValue)
  - public Set getStringSet (clé de chaîne, Set defValues)
  - public void registerOnSharedPreferenceChangeListener (auditeur SharedPreferences.OnSharedPreferenceChangeListener)
  - void public unregisterOnSharedPreferenceChangeListener (auditeur SharedPreferences.OnSharedPreferenceChangeListener)
- **Méthodes SharedPreferences.Editor**
  - annulation publique appliquer ()
  - commit booléen public ()
  - public SharedPreferences.Editor clear ()
  - public SharedPreferences.Editor putBoolean (clé de chaîne, valeur booléenne)
  - public SharedPreferences.Editor putFloat (clé de chaîne, valeur flottante)
  - public SharedPreferences.Editor putInt (clé de chaîne, valeur int)
  - public SharedPreferences.Editor putLong (clé de chaîne, valeur longue)
  - public SharedPreferences.Editor putString (clé de chaîne, valeur de chaîne)
  - public SharedPreferences.Editor putStringSet (clé de chaîne, valeurs définies)
  - public SharedPreferences.Editor remove (Clé de chaîne)

Paramètres

Paramètre	Détails
clé	Une `String` non nulle identifiant le paramètre. Il peut contenir des espaces ou des éléments non imprimables. Ceci n'est utilisé que dans votre application (et dans le fichier XML), il n'est donc pas nécessaire qu'il soit placé dans un répertoire, mais c'est une bonne idée de l'avoir comme constante dans votre code source. Ne le localisez pas.
defValue	Toutes les fonctions get prennent une valeur par défaut, qui est retournée si la clé donnée n'est pas présente dans les `SharedPreferences` . Il n'est pas retourné si la clé est présente mais la valeur a le mauvais type: dans ce cas, vous obtenez une `ClassCastException` .

Remarques

- `SharedPreferences` ne doit pas être utilisé pour stocker une grande quantité de données. A ces fins, il est préférable d'utiliser `SQLiteDatabase` .
- `SharedPreferences` est un processus unique uniquement, sauf si vous utilisez le mode obsolète `MODE_MULTI_PROCESS` . Donc, si votre application dispose de plusieurs processus, vous ne pourrez pas lire les `SharedPreferences` du processus principal dans un autre processus. Dans de tels cas, vous devez utiliser un autre mécanisme pour partager des données entre les processus, mais n'utilisez pas `MODE_MULTI_PROCESS` car il n'est pas fiable ni obsolète.
- Il est préférable d'utiliser l'instance `SharedPreferences` dans la classe `Singleton` pour accéder à l'ensemble du `context` l'application. Si vous voulez l'utiliser uniquement pour une activité particulière, rendez-vous sur `getPreferences()` .
- Évitez de stocker des informations sensibles en texte clair lorsque vous utilisez `SharedPreferences` car il peut être lu facilement.

Documentation officielle

https://developer.android.com/reference/android/content/SharedPreferences.html

Examples

Lire et écrire des valeurs dans 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()` est une méthode de la classe `Context` - dont l' `Activity` s'étend. Si vous devez accéder à la méthode `getSharedPreferences()` partir d'autres classes, vous pouvez utiliser `context.getSharedPreferences()` avec une référence d'objet de `Context` partir d'une `Activity` , d'une `View` ou d'une `Application` .

Enlever les clés

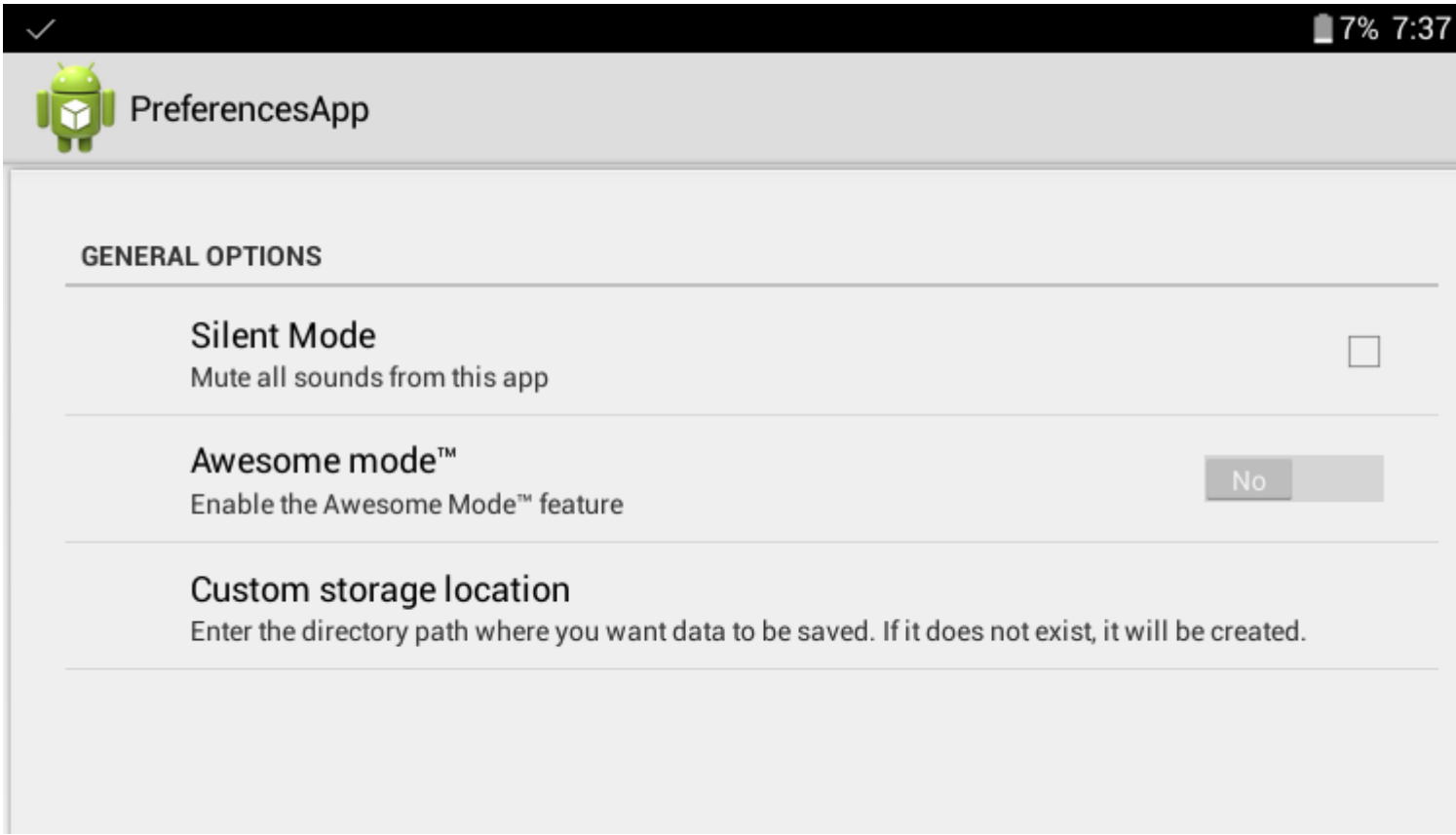
```
private static final String MY_PREF = "MyPref";

// ...
```

```
SharedPreferences prefs = ...;

// ...

SharedPreferences.Editor editor = prefs.edit();
editor.putString(MY_PREF, "value");
editor.remove(MY_PREF);
editor.apply();
```

Après le `apply()` , `prefs` contient "key" -> "value", en plus de ce qu'il contient déjà. Même s'il semble que j'ai ajouté "clé" et que je l'ai ensuite retiré, la suppression se produit en premier. Les modifications dans l' `Editor` sont toutes appliquées en une seule fois, et non dans l'ordre dans lequel vous les avez ajoutées. Toutes les suppressions ont lieu avant toutes les mises.

Implémentation d'un écran Paramètres en utilisant SharedPreferences

Une utilisation de `SharedPreferences` consiste à implémenter un écran "Settings" dans votre application, où l'utilisateur peut définir ses préférences / options. Comme ça:



Un `SharedPreferences` préférence enregistre les préférences de l'utilisateur dans les préférences `SharedPreferences` . Pour créer un écran de préférence, vous avez besoin de quelques éléments:

Un fichier XML pour définir les options disponibles:

Cela va dans `/res/xml/preferences.xml` , et pour l'écran de paramètres ci-dessus, cela ressemble à ceci:

```
<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>
```

Ceci définit les options disponibles dans l'écran des paramètres. Il existe de nombreux autres types de `Preference` répertoriés dans la documentation des développeurs Android sur la classe de préférence .

Ensuite, nous avons besoin d' **une activité pour héberger notre** interface utilisateur de **préférences** . Dans ce cas, c'est assez court et ressemble à ceci:

```
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);
    }
}
```

Il étend `PreferenceActivity` et fournit l'interface utilisateur pour l'écran des préférences. Il peut être lancé comme une activité normale, dans ce cas, avec quelque chose comme:

```
Intent i = new Intent(this, PreferencesActivity.class);
startActivity(i);
```

N'oubliez pas d'ajouter `PreferencesActivity` à votre `AndroidManifest.xml` .

Obtenir les valeurs des préférences dans votre application est assez simple, appelez

simplement `setDefaultValues()` pour définir les valeurs par défaut définies dans votre XML, puis obtenez les `SharedPreferences` par défaut. Un exemple:

```
//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", "");
```

Récupérer toutes les entrées stockées d'un fichier SharedPreferences particulier

La méthode `getAll()` récupère toutes les valeurs des préférences. Nous pouvons l'utiliser, par exemple, pour enregistrer le contenu actuel des `SharedPreferences` :

```
private static final String PREFS_FILE = "MyPrefs";

public static void logSharedPreferences(final Context context) {
    SharedPreferences sharedPreferences = context.getSharedPreferences(PREFS_FILE,
Context.MODE_PRIVATE);
    Map<String, ?> allEntries = sharedPreferences.getAll();
    for (Map.Entry<String, ?> entry : allEntries.entrySet()) {
        final String key = entry.getKey();
        final Object value = entry.getValue();
        Log.d("map values", key + ": " + value);
    }
}
```

La documentation vous avertit de la modification de la `Collection` retournée par `getAll` :

> Notez que vous ne devez pas modifier la collection renvoyée par cette méthode ou modifier son contenu. La cohérence de vos données stockées n'est pas garantie si vous le faites.

Écoute des modifications de SharedPreferences

```
SharedPreferences sharedPreferences = ...;
sharedPreferences.registerOnSharedPreferenceChangeListener(mOnSharedPreferenceChangeListener);


private final SharedPreferences.OnSharedPreferenceChangeListener
mOnSharedPreferenceChangeListener = new SharedPreferences.OnSharedPreferenceChangeListener() {
    @Override
    public void onSharedPreferenceChanged(SharedPreferences sharedPreferences, String key) {
        //TODO
    }
}
```

Notez s'il vous plaît:

- L'auditeur ne se déclenche que si la valeur a été ajoutée ou modifiée, le réglage de la même valeur ne l'appelera pas;
- Le programme d'écoute doit être enregistré dans une variable membre et **non** avec une classe anonyme, car `registerOnSharedPreferenceChangeListener` stocke avec une référence faible, de sorte qu'il soit récupéré.
- Au lieu d'utiliser une variable membre, elle peut également être directement implémentée par la classe, puis appeler `registerOnSharedPreferenceChangeListener(this);`
- N'oubliez pas de désenregistrer l'écouteur lorsqu'il n'est plus nécessaire d'utiliser `unregisterOnSharedPreferenceChangeListener` .

Lecture et écriture de données dans SharedPreferences with Singleton

Classe SharedPreferences Manager (Singleton) pour lire et écrire tous les types de données.

```
import android.content.Context;
import android.content.SharedPreferences;
import android.util.Log;

import com.google.gson.Gson;

import java.lang.reflect.Type;

/**
 * Singleton Class for accessing SharedPreferences,
 * should be initialized once in the beginning by any application component using static
 * method initialize(applicationContext)
 */
public class SharedPrefsManager {

    private static final String TAG = SharedPrefsManager.class.getName();
    private SharedPreferences prefs;
    private static SharedPrefsManager uniqueInstance;
    public static final String PREF_NAME = "com.example.app";

    private SharedPrefsManager(Context appContext) {
        prefs = appContext.getSharedPreferences(PREF_NAME, Context.MODE_PRIVATE);
    }

    /**
     * Throws IllegalStateException if this class is not initialized
     *
     * @return unique SharedPrefsManager instance
     */
    public static SharedPrefsManager getInstance() {
        if (uniqueInstance == null) {
            throw new IllegalStateException(
                    "SharedPrefsManager is not initialized, call
initialize(applicationContext) " +
                            "static method first");
        }
        return uniqueInstance;
    }

    /**
     * Initialize this class using application Context,
     * should be called once in the beginning by any application Component
     *
```

```
     * @param appContext application context
     */
    public static void initialize(Context appContext) {
        if (appContext == null) {
            throw new NullPointerException("Provided application context is null");
        }
        if (uniqueInstance == null) {
            synchronized (SharedPrefsManager.class) {
                if (uniqueInstance == null) {
                    uniqueInstance = new SharedPrefsManager(appContext);
                }
            }
        }
    }

    private SharedPreferences getPrefs() {
        return prefs;
    }

    /**
     * Clears all data in SharedPreferences
     */
    public void clearPrefs() {
        SharedPreferences.Editor editor = getPrefs().edit();
        editor.clear();
        editor.commit();
    }

    public void removeKey(String key) {
        getPrefs().edit().remove(key).commit();
    }

    public boolean containsKey(String key) {
        return getPrefs().contains(key);
    }

    public String getString(String key, String defValue) {
        return getPrefs().getString(key, defValue);
    }

    public String getString(String key) {
        return getString(key, null);
    }

    public void setString(String key, String value) {
        SharedPreferences.Editor editor = getPrefs().edit();
        editor.putString(key, value);
        editor.apply();
    }

    public int getInt(String key, int defValue) {
        return getPrefs().getInt(key, defValue);
    }

    public int getInt(String key) {
        return getInt(key, 0);
    }

    public void setInt(String key, int value) {
        SharedPreferences.Editor editor = getPrefs().edit();
        editor.putInt(key, value);
```

```
        editor.apply();
    }

    public long getLong(String key, long defValue) {
        return getPrefs().getLong(key, defValue);
    }

    public long getLong(String key) {
        return getLong(key, 0L);
    }

    public void setLong(String key, long value) {
        SharedPreferences.Editor editor = getPrefs().edit();
        editor.putLong(key, value);
        editor.apply();
    }

    public boolean getBoolean(String key, boolean defValue) {
        return getPrefs().getBoolean(key, defValue);
    }

    public boolean getBoolean(String key) {
        return getBoolean(key, false);
    }

    public void setBoolean(String key, boolean value) {
        SharedPreferences.Editor editor = getPrefs().edit();
        editor.putBoolean(key, value);
        editor.apply();
    }

    public boolean getFloat(String key) {
        return getFloat(key, 0f);
    }

    public boolean getFloat(String key, float defValue) {
        return getFloat(key, defValue);
    }

    public void setFloat(String key, Float value) {
        SharedPreferences.Editor editor = getPrefs().edit();
        editor.putFloat(key, value);
        editor.apply();
    }

    /**
     * Persists an Object in prefs at the specified key, class of given Object must implement
Model
     * interface
     *
     * @param key         String
     * @param modelObject Object to persist
     * @param <M>         Generic for Object
     */
    public <M extends Model> void setObject(String key, M modelObject) {
        String value = createJSONStringFromObject(modelObject);
        SharedPreferences.Editor editor = getPrefs().edit();
        editor.putString(key, value);
        editor.apply();
    }
```

```
    /**
     * Fetches the previously stored Object of given Class from prefs
     *
     * @param key                String
     * @param classOfModelObject Class of persisted Object
     * @param <M>                Generic for Object
     * @return Object of given class
     */
    public <M extends Model> M getObject(String key, Class<M> classOfModelObject) {
        String jsonData = getPrefs().getString(key, null);
        if (null != jsonData) {
            try {
                Gson gson = new Gson();
                M customObject = gson.fromJson(jsonData, classOfModelObject);
                return customObject;
            } catch (ClassCastException cce) {
                Log.d(TAG, "Cannot convert string obtained from prefs into collection of type
" +
                        classOfModelObject.getName() + "\n" + cce.getMessage());
            }
        }
        return null;
    }

    /**
     * Persists a Collection object in prefs at the specified key
     *
     * @param key            String
     * @param dataCollection Collection Object
     * @param <C>            Generic for Collection object
     */
    public <C> void setCollection(String key, C dataCollection) {
        SharedPreferences.Editor editor = getPrefs().edit();
        String value = createJSONStringFromObject(dataCollection);
        editor.putString(key, value);
        editor.apply();
    }

    /**
     * Fetches the previously stored Collection Object of given type from prefs
     *
     * @param key     String
     * @param typeOfC Type of Collection Object
     * @param <C>     Generic for Collection Object
     * @return Collection Object which can be casted
     */
    public <C> C getCollection(String key, Type typeOfC) {
        String jsonData = getPrefs().getString(key, null);
        if (null != jsonData) {
            try {
                Gson gson = new Gson();
                C arrFromPrefs = gson.fromJson(jsonData, typeOfC);
                return arrFromPrefs;
            } catch (ClassCastException cce) {
                Log.d(TAG, "Cannot convert string obtained from prefs into collection of type
" +
                        typeOfC.toString() + "\n" + cce.getMessage());
            }
        }
        return null;
    }
```

```
    public void registerPrefsListener(SharedPreferences.OnSharedPreferenceChangeListener
listener) {
        getPrefs().registerOnSharedPreferenceChangeListener(listener);
    }

    public void unregisterPrefsListener(

            SharedPreferences.OnSharedPreferenceChangeListener listener) {
        getPrefs().unregisterOnSharedPreferenceChangeListener(listener);
    }

    public SharedPreferences.Editor getEditor() {
        return getPrefs().edit();
    }

    private static String createJSONStringFromObject(Object object) {
        Gson gson = new Gson();
        return gson.toJson(object);
    }
}
```

`interface Model` qui est implémentée par les classes allant à `Gson` pour éviter le brouillage de proguard.

```
public interface Model {

}
```

Règles de progression pour l'interface du `Model` :

```
-keep interface com.example.app.Model
-keep class * implements com.example.app.Model { *;}
```

Différentes manières d'instancier un objet de SharedPreferences

Vous pouvez accéder à SharedPreferences de plusieurs manières:

Récupère le fichier SharedPreferences par défaut:

```
import android.preference.PreferenceManager;
SharedPreferences prefs = PreferenceManager.getDefaultSharedPreferences(this);
```

Obtenez un fichier SharedPreferences spécifique:

```
public static final String PREF_FILE_NAME = "PrefFile";
SharedPreferences prefs = getSharedPreferences(PREF_FILE_NAME, MODE_PRIVATE);
```

Obtenir des préférences partagées à partir d'une autre application:

```
// 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)`

renvoie les préférences enregistrées par `Activity's class name` comme décrit dans la documentation :

> Récupérer un objet SharedPreferences pour accéder aux préférences privées de cette activité. Cela appelle simplement la méthode getSharedPreferences (String, int) sous-jacente en transmettant le nom de classe de cette activité comme nom de préférences.

Lors de l'utilisation de la méthode getSharedPreferences (nom de chaîne, mode int) renvoie les préférences enregistrées sous le `name` donné. Comme dans les docs:

> Récupérez et conservez le contenu du fichier de préférences "nom", en renvoyant une valeur de SharedPreferences à travers laquelle vous pouvez récupérer et modifier ses valeurs.

Donc, si la valeur enregistrée dans les `SharedPreferences` doit être utilisée dans l'application, il faut utiliser `getSharedPreferences (String name, int mode)` avec un nom fixe. Comme, l'utilisation de `getPreferences(int)` renvoie / enregistre les préférences appartenant à l' `Activity` appelle.

Commit vs. Apply

La méthode `editor.apply()` est **asynchrone** , alors que `editor.commit()` est **synchrone** .

De toute évidence, vous devriez appeler `apply()` ou `commit()` .

2.3

```
SharedPreferences settings = getSharedPreferences(PREFS_FILE, MODE_PRIVATE);
SharedPreferences.Editor editor = settings.edit();
editor.putBoolean(PREF_CONST, true);
// This will asynchronously save the shared preferences without holding the current thread.
editor.apply();
```

```
SharedPreferences settings = getSharedPreferences(PREFS_FILE, MODE_PRIVATE);
SharedPreferences.Editor editor = settings.edit();
editor.putBoolean(PREF_CONST, true);
// This will synchronously save the shared preferences while holding the current thread until
done and returning a success flag.
boolean result = editor.commit();
```

`apply()` été ajouté dans 2.3 (API 9), il valide sans retourner un booléen indiquant le succès ou l'échec.

`commit()` renvoie true si la sauvegarde fonctionne, false sinon.

`apply()`

été ajouté, car l'équipe de développement Android a remarqué que presque personne ne prenait note de la valeur de retour, donc, appliquer est plus rapide car elle est asynchrone.

Contrairement à `commit()` , qui écrit ses préférences dans le stockage persistant de manière synchrone, `apply()` `SharedPreferences` immédiatement les modifications apportées à la `SharedPreferences` mémoire, mais lance une validation asynchrone sur le disque. Si un autre éditeur de cette `SharedPreferences` effectue un `commit()` normal `commit()` alors qu'un `apply()` est toujours en attente, le `commit()` bloquera jusqu'à ce que tous les commits asynchrones (s'appliquent) ainsi que tous les autres commits de synchronisation en attente.

Types de données pris en charge dans SharedPreferences

`SharedPreferences` vous permet de stocker uniquement les types de données primitifs (`boolean` , `float` , `long` , `int` , `String` et `string set`). Vous ne pouvez pas stocker des objets plus complexes dans `SharedPreferences` et, en tant que tel, vous souhaitez stocker des paramètres utilisateur ou similaires, ce n'est pas une base de données pour conserver les données utilisateur (par exemple enregistrer une liste de `SharedPreferences` par exemple).

Pour stocker quelque chose dans `SharedPreferences` vous utilisez une clé et une valeur. La clé est la manière dont vous pouvez référencer ce que vous avez enregistré plus tard et les données de valeur que vous souhaitez stocker.

```
    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();
```

Stocker, récupérer, supprimer et effacer les données des préférences partagées

Créer des préférences partagées BuyyaPref

```
SharedPreferences pref = getApplicationContext().getSharedPreferences("BuyyaPref",
MODE_PRIVATE);
Editor editor = pref.edit();
```

Stockage des données sous forme de paire KEY / VALUE

```
editor.putBoolean("key_name1", true);           // Saving boolean - true/false
editor.putInt("key_name2", 10);        // Saving integer
editor.putFloat("key_name3", 10.1f);    // Saving float
editor.putLong("key_name4", 1000);      // Saving long
editor.putString("key_name5", "MyString");  // Saving string

// Save the changes in SharedPreferences
editor.commit(); // commit changes
```

Obtenir des données SharedPreferences

Si la valeur de la clé n'existe pas, renvoyer la deuxième valeur de paramètre (dans ce cas, null, c'est comme la valeur par défaut)

```
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
```

Supprimer la valeur de clé de SharedPreferences

```
editor.remove("key_name3"); // will delete key key_name3
editor.remove("key_name4"); // will delete key key_name4

// Save the changes in SharedPreferences
editor.commit(); // commit changes
```

Effacer toutes les données de SharedPreferences

```
 editor.clear();
 editor.commit(); // commit changes
```

Pré-Honeycomb avec StringSet

Voici la classe d'utilitaire:

```
public class SharedPreferencesCompat {

    public static void putStringSet(SharedPreferences.Editor editor, String key, Set<String>
values) {
            if (Build.VERSION.SDK_INT >= 11) {
                while (true) {
                    try {
                        editor.putStringSet(key, values).apply();
                        break;
                    } catch (ClassCastException ex) {
                        // Clear stale JSON string from before system upgrade
                        editor.remove(key);
                    }
                }
            } else putStringSetToJson(editor, key, values);
    }

    public static Set<String> getStringSet(SharedPreferences prefs, String key, Set<String>
defaultReturnValue) {
        if (Build.VERSION.SDK_INT >= 11) {
            try {
                return prefs.getStringSet(key, defaultReturnValue);
            } catch (ClassCastException ex) {
                // If user upgraded from Gingerbread to something higher read the stale JSON
string
                return getStringSetFromJson(prefs, key, defaultReturnValue);
            }
```

```
        } else return getStringSetFromJson(prefs, key, defaultReturnValue);
    }

    private static Set<String> getStringSetFromJson(SharedPreferences prefs, String key,
Set<String> defaultReturnValue) {
        final String input = prefs.getString(key, null);
        if (input == null) return defaultReturnValue;

        try {
            HashSet<String> set = new HashSet<>();
            JSONArray json = new JSONArray(input);
            for (int i = 0, size = json.length(); i < size; i++) {
                String value = json.getString(i);
                set.add(value);
            }
            return set;
        } catch (JSONException e) {
            e.printStackTrace();
            return defaultReturnValue;
        }
    }

    private static void putStringSetToJson(SharedPreferences.Editor editor, String key,
Set<String> values) {
        JSONArray json = new JSONArray(values);
        if (Build.VERSION.SDK_INT >= 9)
            editor.putString(key, json.toString()).apply();
        else
            editor.putString(key, json.toString()).commit();
    }

    private SharedPreferencesCompat() {}
}
```

Un exemple pour enregistrer les préférences en tant que type de données StringSet est:

```
Set<String> sets = new HashSet<>();
sets.add("John");
sets.add("Nicko");
SharedPreferences preferences = PreferenceManager.getDefaultSharedPreferences(this);
SharedPreferencesCompat.putStringSet(preferences.edit(), "pref_people", sets);
```

Pour les récupérer:

```
Set<String> people = SharedPreferencesCompat.getStringSet(preferences, "pref_people", new
HashSet<String>());
```

Référence: Préférence de support Android

Ajouter un filtre pour EditTextPreference

Créez cette classe:

```
public class InputFilterMinMax implements InputFilter {

    private int min, max;
```

```
    public InputFilterMinMax(int min, int max) {
        this.min = min;
        this.max = max;
    }

    public InputFilterMinMax(String min, String max) {
        this.min = Integer.parseInt(min);
        this.max = Integer.parseInt(max);
    }

    @Override
    public CharSequence filter(CharSequence source, int start, int end, Spanned dest, int
dstart, int dend) {
        try {
            int input = Integer.parseInt(dest.toString() + source.toString());
            if (isInRange(min, max, input))
                return null;
        } catch (NumberFormatException nfe) { }
        return "";
    }

    private boolean isInRange(int a, int b, int c) {
        return b > a ? c >= a && c <= b : c >= b && c <= a;
    }
}
```

Utilisation :

```
EditText compressPic = ((EditTextPreference)
findPreference(getString("pref_key_compress_pic"))).getEditText();
compressPic.setFilters(new InputFilter[]{ new InputFilterMinMax(1, 100) });
```

Lire Des préférences partagées en ligne: https://riptutorial.com/fr/android/topic/119/des-preferences-partagees

Chapitre 91: Dessin sur toile avec SurfaceView

Remarques

Il est important de comprendre le concept de base de la vue de surface avant d'utiliser:

- C'est simplement un trou dans la fenêtre actuelle
- L'interface utilisateur native peut être placée dessus
- Le dessin est fait en utilisant un thread dédié non-interface utilisateur
- Le dessin n'est pas accéléré par le matériel
- Utilise deux tampons: L'un est actuellement affiché, l'autre est utilisé pour le dessin.
- `unlockCanvasAndPost()` échange les tampons.

Les blocages peuvent facilement se produire si les `lockCanvas()` et `unlockCanvasAndPost()` ne sont pas appelées dans le bon ordre.

Examples

SurfaceView avec fil de dessin

Cet exemple décrit comment créer un SurfaceView avec un thread de dessin dédié. Cette implémentation gère également les cas limites tels que la fabrication de problèmes spécifiques ainsi que le démarrage / arrêt du thread pour économiser du temps processeur.

```
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");
    }
}
```

Cette disposition ne contient que le SurfaceView personnalisé et le maximise à la taille de l'écran.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="sample.devcore.org.surfaceviewsample.MainActivity">

    <sample.devcore.org.surfaceviewsample.BaseSurface
        android:id="@+id/baseSurface"
        android:layout_width="match_parent"
        android:layout_height="match_parent"/>
</LinearLayout>
```

L'activité qui utilise SurfaceView est responsable du démarrage et de l'arrêt du thread de dessin. Cette approche permet d'économiser la batterie car le dessin est arrêté dès que l'activité est en arrière-plan.

```
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();
    }
}
```

Lire Dessin sur toile avec SurfaceView en ligne: https://riptutorial.com/fr/android/topic/3754/dessin-sur-toile-avec-surfaceview

Chapitre 92: Dessins vectoriels

Introduction

Comme son nom l'indique, les dessins vectoriels sont basés sur des graphiques vectoriels. Les graphiques vectoriels permettent de décrire des éléments graphiques à l'aide de formes géométriques. Cela vous permet de créer un dessin basé sur un graphique vectoriel XML. Désormais, il n'est plus nécessaire de concevoir des images de tailles différentes pour les formats mdpi, hdpi, xhdpi, etc. Cela ne permet pas non plus d'économiser de l'espace mais simplifie également la maintenance.

Paramètres

Paramètre	Détails
`<vector>`	Utilisé pour définir un dessin vectoriel
`<group>`	Définit un groupe de chemins ou de sous-groupes, ainsi que des informations de transformation. Les transformations sont définies dans les mêmes coordonnées que la fenêtre d'affichage. Et les transformations sont appliquées dans l'ordre d'échelle, en rotation puis en translation.
`<path>`	Définit les chemins à tracer.
`<clip-path>`	Définit le chemin pour être le clip en cours. Notez que le chemin d'accès du clip s'applique uniquement au groupe actuel et à ses enfants.

Remarques

Mettez à jour le fichier **build.gradle** .

```
dependencies {
    ...
   compile 'com.android.support:appcompat-v7:23.2.1'
}
```

Si vous utilisez la version **2.0 ou supérieure** du **plug** - **in Gradle** , ajoutez le code suivant.

```
// Gradle Plugin 2.0+
 android {
   defaultConfig {
     vectorDrawables.useSupportLibrary = true
    }
 }
```

Si vous utilisez la version **1.5 ou inférieure** du **plug** - **in Gradle** , ajoutez le code suivant.

```
// Gradle Plugin 1.5
 android {
   defaultConfig {
     generatedDensities = []
  }

  // This is handled for you by the 2.0+ Gradle Plugin
  aaptOptions {
    additionalParameters "--no-version-vectors"
  }
 }
```

Lisez les notes de publication d'Android Support Library 23.2 pour plus d'informations.

REMARQUE: Même avec *AppCompat* , Vector Drawables ne fonctionnera pas en dehors de votre application dans les anciennes versions Android. Par exemple, vous ne pouvez pas transmettre les fichiers vectoriels en tant qu'icônes de notification car ils sont gérés par le système et non par l'application. Voir cette réponse pour une solution de contournement.

Examples

Exemple d'utilisation de VectorDrawable

Voici un exemple d'actif vectoriel que nous utilisons actuellement dans AppCompat:

res / drawable / ic_search.xml

```
<vector xmlns:android="..."
        android:width="24dp"
        android:height="24dp"
        android:viewportWidth="24.0"
        android:viewportHeight="24.0"
        android:tint="?attr/colorControlNormal">

    <path
        android:pathData="..."
        android:fillColor="@android:color/white"/>

</vector>
```

En utilisant ce dessin, un exemple de déclaration `ImageView` serait:

```
<ImageView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    app:srcCompat="@drawable/ic_search"/>
```

Vous pouvez également le définir à l'exécution:

```
ImageView iv = (ImageView) findViewById(...);
iv.setImageResource(R.drawable.ic_search);
```

Le même attribut et les appels fonctionnent également pour `ImageButton` .

Exemple XML VectorDrawable

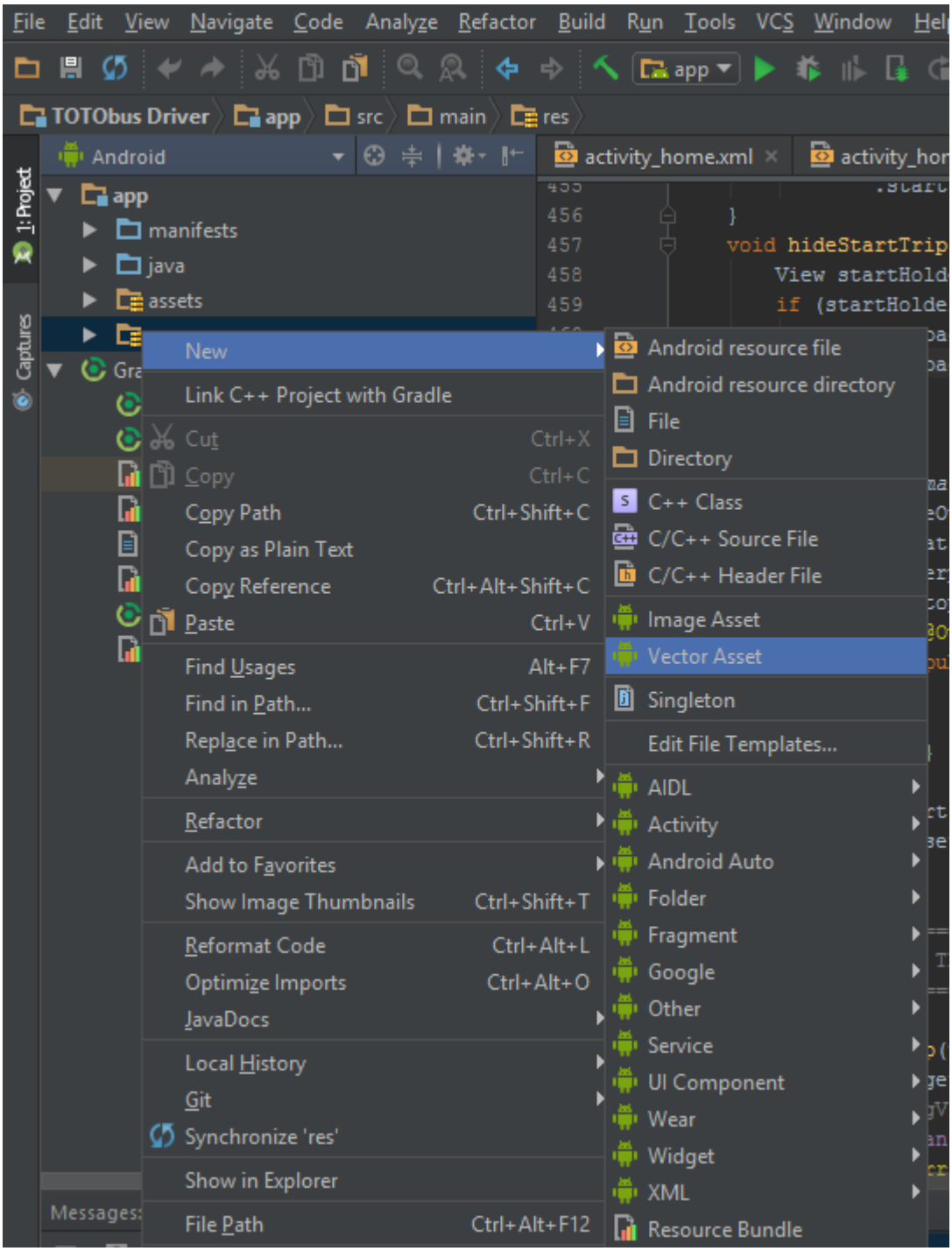
Voici un simple `VectorDrawable` dans ce fichier **vectordrawable.xml** .

```
<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>
```

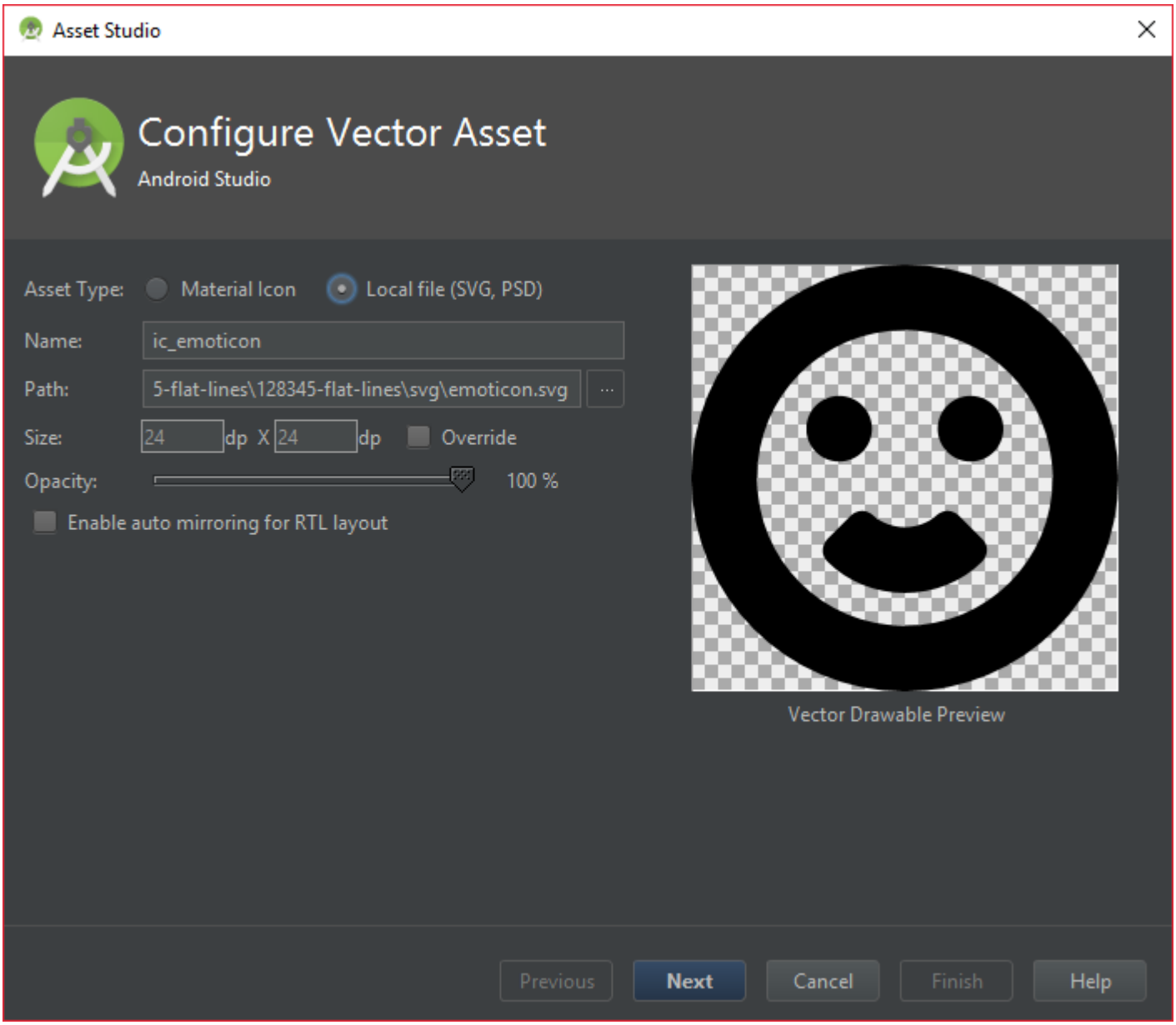
Importation d'un fichier SVG en tant que VectorDrawable

Vous pouvez importer un fichier *SVG* en tant que `VectorDrawable` dans Android Studio, procédez comme suit:

Cliquez avec le bouton droit de la souris sur le dossier `res` et sélectionnez **new** > **Vector Asset** .



Sélectionnez l'option **Fichier local** et accédez à votre fichier .svg. Changez les options à votre convenance et cliquez sur Suivant. Terminé.



Lire Dessins vectoriels en ligne: https://riptutorial.com/fr/android/topic/8194/dessins-vectoriels

Chapitre 93: Détecter l'événement Shake dans Android

Examples

Shake Detector in Android Exemple

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

Utilisation de la détection de secousses sismiques

Seismic est un périphérique Android de détection de bibliothèque de détection par Square. Pour l'utiliser, commencez simplement à écouter les événements de secousse émis par celui-ci.

```
@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();
}
```

Pour définir un seuil d'accélération différent, utilisez `sd.setSensitivity(sensitivity)` avec une `sensitivity` de `SENSITIVITY_LIGHT` , `SENSITIVITY_MEDIUM` , `SENSITIVITY_HARD` ou toute autre valeur entière raisonnable. Les valeurs par défaut données vont de *11* à *15* .

Installation

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

Lire Détecter l'événement Shake dans Android en ligne:
https://riptutorial.com/fr/android/topic/4501/detecter-l-evenement-shake-dans-android

Chapitre 94: Détection de geste

Remarques

Documentation officielle: Détection des gestes courants

Examples

Détection de balayage

```
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() {
    }

}
```

Appliqué à une vue ...

```
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");
    }

});
```

Détection de base du geste

```
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;
    }

}
```

Lire Détection de geste en ligne: https://riptutorial.com/fr/android/topic/4711/detection-de-geste

Chapitre 95: Développement de jeux Android

Introduction

Une brève introduction à la création d'un jeu sur la plateforme Android en utilisant Java

Remarques

- Le premier exemple couvre les bases: il n'y a pas d'objectifs, mais il montre comment vous créez une partie de base d'un jeu 2D à l'aide de SurfaceView.
- Veillez à enregistrer toutes les données importantes lorsque vous créez un jeu; tout le reste sera perdu

Examples

Jeu utilisant Canvas et SurfaceView

Cela couvre comment vous pouvez créer un jeu 2D de base en utilisant SurfaceView.

Tout d'abord, nous avons besoin d'une activité:

```
public class GameLauncher extends AppCompatActivity {

    private Game game;
    @Override
    public void onCreate(Bundle sis){
        super.onCreate(sis);
        game = new Game(GameLauncher.this);//Initialize the game instance
        setContentView(game);//setContentView to the game surfaceview
        //Custom XML files can also be used, and then retrieve the game instance using
findViewById.
    }

}
```

L'activité doit également être déclarée dans le manifeste Android.

Maintenant pour le jeu lui-même. Tout d'abord, nous commençons par implémenter un thread de jeu:

```
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");
    }
}
```

C'est la partie de base. Vous avez maintenant la possibilité de dessiner sur l'écran.

Maintenant, commençons par ajouter aux entiers:

```
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;
```

Pour cette prochaine partie, vous allez avoir besoin d'une image. Il devrait être d'environ 100x100 mais il peut être plus grand ou plus petit. Pour apprendre, un Rect peut également être utilisé (mais cela nécessite un changement de code un peu plus bas)

Maintenant, nous déclarons un bitmap:

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

Dans le rendu, nous devons dessiner ce bitmap.

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

AVANT DE LANCER, il reste encore des choses à faire

Nous avons d'abord besoin d'un booléen:

```
boolean up = false;
```

dans onTouchEvent, nous ajoutons:

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

Et en cochant cette case, vous devez déplacer le joueur:

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

et maintenant nous en avons besoin dans 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();
```

Et nous avons besoin de ces variables:

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

À ce stade, le jeu est exécutable. Cela signifie que vous pouvez le lancer et le tester.

Vous devez maintenant avoir une image de joueur ou un recto en haut et en bas de l'écran. Le lecteur peut être créé en tant que classe personnalisée si nécessaire. Ensuite, toutes les choses liées au lecteur peuvent être déplacées dans cette classe et utiliser une instance de cette classe pour déplacer, rendre et exécuter une autre logique.

Maintenant, comme vous l'avez probablement vu sous test, il quitte l'écran. Nous devons donc le limiter.

Tout d'abord, nous devons déclarer le Rect:

```
private Rect screen;
```

Dans init, après l'initialisation de la largeur et de la hauteur, nous créons un nouveau rect qui est l'écran.

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

Maintenant, nous avons besoin d'un autre rect sous la forme d'une méthode:

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

et en tick:

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

L'implémentation de gameOVer peut également être utilisée pour afficher le début d'un jeu.

Autres aspects d'un jeu à noter:

Enregistrement (actuellement manquant dans la documentation)

Lire Développement de jeux Android en ligne:
https://riptutorial.com/fr/android/topic/10011/developpement-de-jeux-android

Chapitre 96: Dialogue

Paramètres

Ligne	La description
montrer();	Affiche la boîte de dialogue
setContentView (R.layout.yourlayout)	définit le ContentView de la boîte de dialogue sur votre présentation personnalisée.
rejeter()	Ferme la boîte de dialogue

Remarques

- La boîte de dialogue du premier exemple (Dialog) n'a pas besoin d'appeler `show()` lors de sa création dans le constructeur.
- Les dialogues d'alerte doivent être construits via une nouvelle instance de la classe `AlertDialog.Builder()` . Après le modèle de générateur , tous les membres du composant AlertDialog.Builder peuvent être chaînés pour «construire» l'instance de dialogue.
- Le générateur Alert Dialog peut `show()` directement `show()` la boîte de dialogue - vous n'avez pas besoin d'appeler `create()` puis `show()` sur l'instance AlertDialog

Examples

Dialogue d'alerte

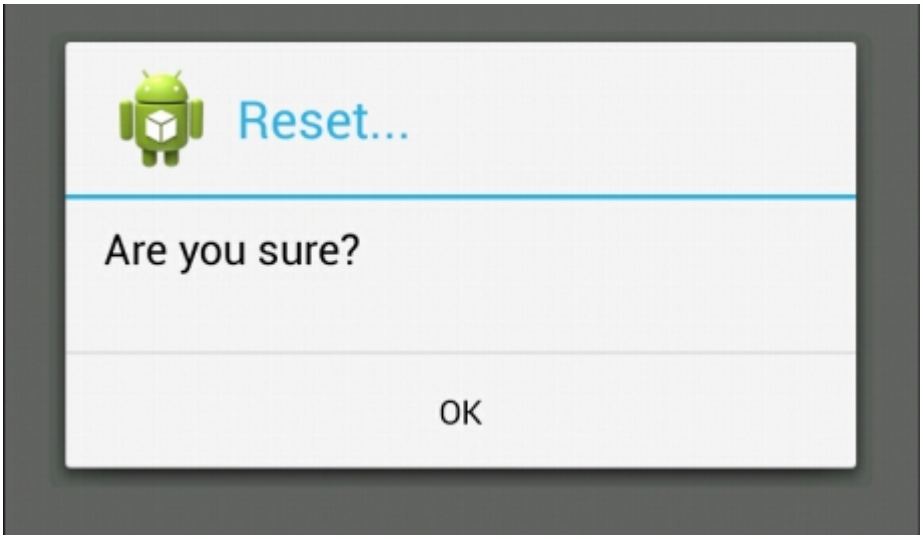
```
AlertDialog.Builder alertDialogBuilder = new AlertDialog.Builder(
                MainActivity.this);

        alertDialogBuilder.setTitle("Title Dialog");
        alertDialogBuilder
                .setMessage("Message Dialog")
                .setCancelable(true)
                .setPositiveButton("Yes",
                        new DialogInterface.OnClickListener() {

                            public void onClick(DialogInterface dialog, int arg1) {
                                // Handle Positive Button

                            }
                        })
                .setNegativeButton("No",
                        new DialogInterface.OnClickListener() {

                            public void onClick(DialogInterface dialog, int arg1) {
```

```
                                // Handle Negative Button
                                dialog.cancel();
                            }
                        });

        AlertDialog alertDialog = alertDialogBuilder.create();
        alertDialog.show();
```

Un dialogue d'alerte de base

```
    AlertDialog.Builder builder = new AlertDialog.Builder(context);
    //Set Title
    builder.setTitle("Reset...")
            //Set Message
            .setMessage("Are you sure?")
            //Set the icon of the dialog
            .setIcon(drawable)
            //Set the positive button, in this case, OK, which will dismiss the dialog and do
everything in the onClick method
            .setPositiveButton(android.R.string.ok, new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialogInterface, int i) {
                    // Reset
                }
            });
    AlertDialog dialog = builder.create();
    //Now, any time you can call on:
    dialog.show();
    //So you can show the dialog.
```

Maintenant ce code réalisera ceci:



(*Source de l'image: WikiHow*)

Sélecteur de date dans DialogFragment

xml du dialogue:

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

```
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent">

    <DatePicker
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/datePicker"
        android:layout_gravity="center_horizontal"
        android:calendarViewShown="false"/>

    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="ACCEPT"
        android:id="@+id/buttonAccept" />

</LinearLayout>
```

Classe de dialogue:

```
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;
    }

}
```

Activité appelant le dialogue:

```
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` est le moyen le plus simple d'utiliser `DatePicker` , car vous pouvez afficher la boîte de dialogue n'importe où dans votre application. Vous n'avez pas besoin d'implémenter votre propre disposition avec le widget `DatePicker` .

Comment afficher la boîte de dialogue:

```
DatePickerDialog datePickerDialog = new DatePickerDialog(context, listener, year, month, day);
datePickerDialog.show();
```

Vous pouvez obtenir le widget `DataPicker` dans la boîte de dialogue ci-dessus, pour accéder à plus de fonctions et, par exemple, définir une date minimale en millisecondes:

```
DatePicker datePicker = datePickerDialog.getDatePicker();
datePicker.setMinDate(System.currentTimeMillis());
```

Sélecteur de date

`DatePicker` permet à l'utilisateur de choisir la date. Lorsque nous créons une nouvelle instance de `DatePicker` , nous pouvons définir la date initiale. Si nous ne définissons pas la date initiale, la date actuelle sera définie par défaut.

Nous pouvons montrer `DatePicker` à l'utilisateur en utilisant `DatePickerDialog` ou en créant notre propre disposition avec le widget `DatePicker` .

Nous pouvons également limiter la plage de dates, que l'utilisateur peut choisir.

> En définissant la date minimale en millisecondes

```
//In this case user can pick date only from future
datePicker.setMinDate(System.currentTimeMillis());
```

> En définissant la date maximale en millisecondes

```
//In this case user can pick date only, before following week.
datePicker.setMaxDate(System.currentTimeMillis() + TimeUnit.DAYS.toMillis(7));
```

Pour recevoir des informations sur la date choisie par l'utilisateur, nous devons utiliser `Listener` .

Si nous utilisons `DatePickerDialog` , nous pouvons définir `OnDateSetListener` dans constructeur lorsque nous créons une nouvelle instance de `DatePickerDialog` :

Exemple d'utilisation de `DatePickerDialog`

```
public class SampleActivity extends AppCompatActivity implements
DatePickerDialog.OnDateSetListener {
```

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

    private void showDatePicker() {
        //We need calendar to set current date as initial date in DatePickerDialog.
        Calendar calendar = new GregorianCalendar(Locale.getDefault());
        int year = calendar.get(Calendar.YEAR);
        int month = calendar.get(Calendar.MONTH);
        int day = calendar.get(Calendar.DAY_OF_MONTH);

        DatePickerDialog datePickerDialog = new DatePickerDialog(this, this, year, month,
day);
        datePickerDialog.show();
    }

    @Override
    public void onDateSet(DatePicker datePicker, int year, int month, int day) {

    }
}
```

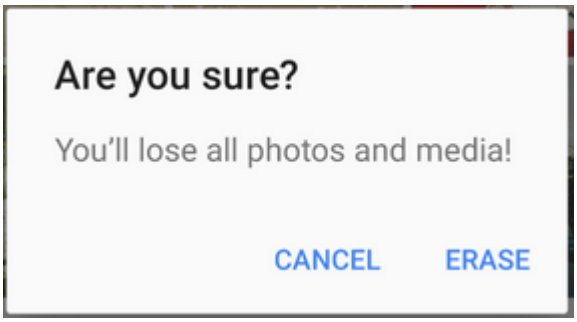
Sinon, si nous créons notre propre disposition avec le widget `DatePicker` , nous devons également créer notre propre écouteur tel qu'il a été montré dans d' autres exemples.

Ajouter AlertDialog à votre application en utilisant Appcompat

`AlertDialog` est une sous-classe de `Dialog` qui peut afficher un, deux ou trois boutons. Si vous souhaitez uniquement afficher une chaîne dans cette boîte de dialogue, utilisez la méthode `setMessage()` .

Le package `AlertDialog` partir de `android.app` s'affiche différemment selon les différentes versions du système d'exploitation Android.

La bibliothèque Android V7 Appcompat fournit une implémentation `AlertDialog` qui s'affichera avec Material Design sur toutes les versions de système d'exploitation Android prises en charge, comme indiqué ci-dessous:



Vous devez d'abord ajouter la bibliothèque V7 Appcompat à votre projet. vous pouvez le faire dans le fichier build.gradle au niveau de l'application:

```
dependencies {
```

```
    compile 'com.android.support:appcompat-v7:24.2.1'
    //........
}
```

Assurez-vous d'importer la classe correcte:

```
import android.support.v7.app.AlertDialog;
```

Ensuite, créez AlertDialog comme ceci:

```
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 dans AlertDialog

Nous pouvons toujours utiliser `ListView` ou `RecyclerView` pour la sélection dans la liste des éléments, mais si nous avons peu de choix et parmi ceux que nous souhaitons en sélectionner un, nous pouvons utiliser `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();
    }
```

Peut-être que si nous n'avons pas besoin d'un `ListView` particulier, nous pouvons utiliser un moyen de base:

```
AlertDialog.Builder builder = new AlertDialog.Builder(this);
builder.setTitle("Select an item")
       .setItems(R.array.your_array, new DialogInterface.OnClickListener() {
           public void onClick(DialogInterface dialog, int which) {
```

```
                // The 'which' argument contains the index position of the selected item
                Log.v(TAG, "Selected item on position " + which);
            }
        });
builder.create().show();
```

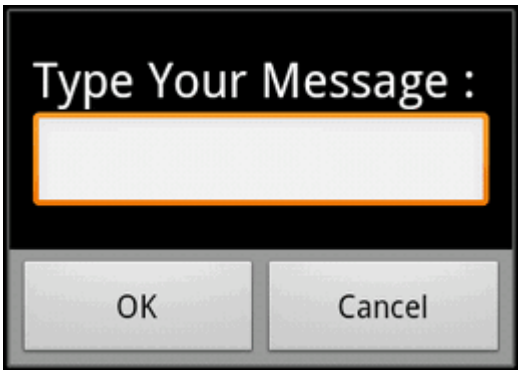
Dialogue d'alerte personnalisé avec EditText

```
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();
    }
```

Fichier Xml: res / layout / alert_dialog.xml

```
<TextView
    android:id="@+id/textView1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Type Your Message : "
    android:textAppearance="?android:attr/textAppearanceLarge" />

<EditText
    android:id="@+id/etUserInput"
    android:layout_width="match_parent"
```

```
    android:layout_height="wrap_content" >

    <requestFocus />

</EditText>
```



Dialogue personnalisé en plein écran sans arrière-plan et sans titre

dans `styles.xml` ajoutez votre style personnalisé:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="AppBaseTheme" parent="@android:style/Theme.Light.NoTitleBar.Fullscreen">
    </style>
</resources>
```

Créez votre disposition personnalisée pour la boîte de dialogue: `fullscreen.xml` :

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

</RelativeLayout>
```

Ensuite, dans un fichier java, vous pouvez l'utiliser pour une activité ou un dialogue, etc:

```
import android.app.Activity;
import android.app.Dialog;
import android.os.Bundle;

public class FullscreenActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //You can set no content for the activity.
        Dialog mDialog = new Dialog(this, R.style.AppBaseTheme);
        mDialog.setContentView(R.layout.fullscreen);
        mDialog.show();
    }
}
```

Dialogue d'alerte avec un titre multiligne

La méthode setCustomTitle () de AlertDialog.Builder vous permet de spécifier une vue arbitraire à utiliser pour le titre de la boîte de dialogue. Une utilisation courante de cette méthode consiste à créer une boîte de dialogue d'alerte comportant un titre long.

```
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();
```

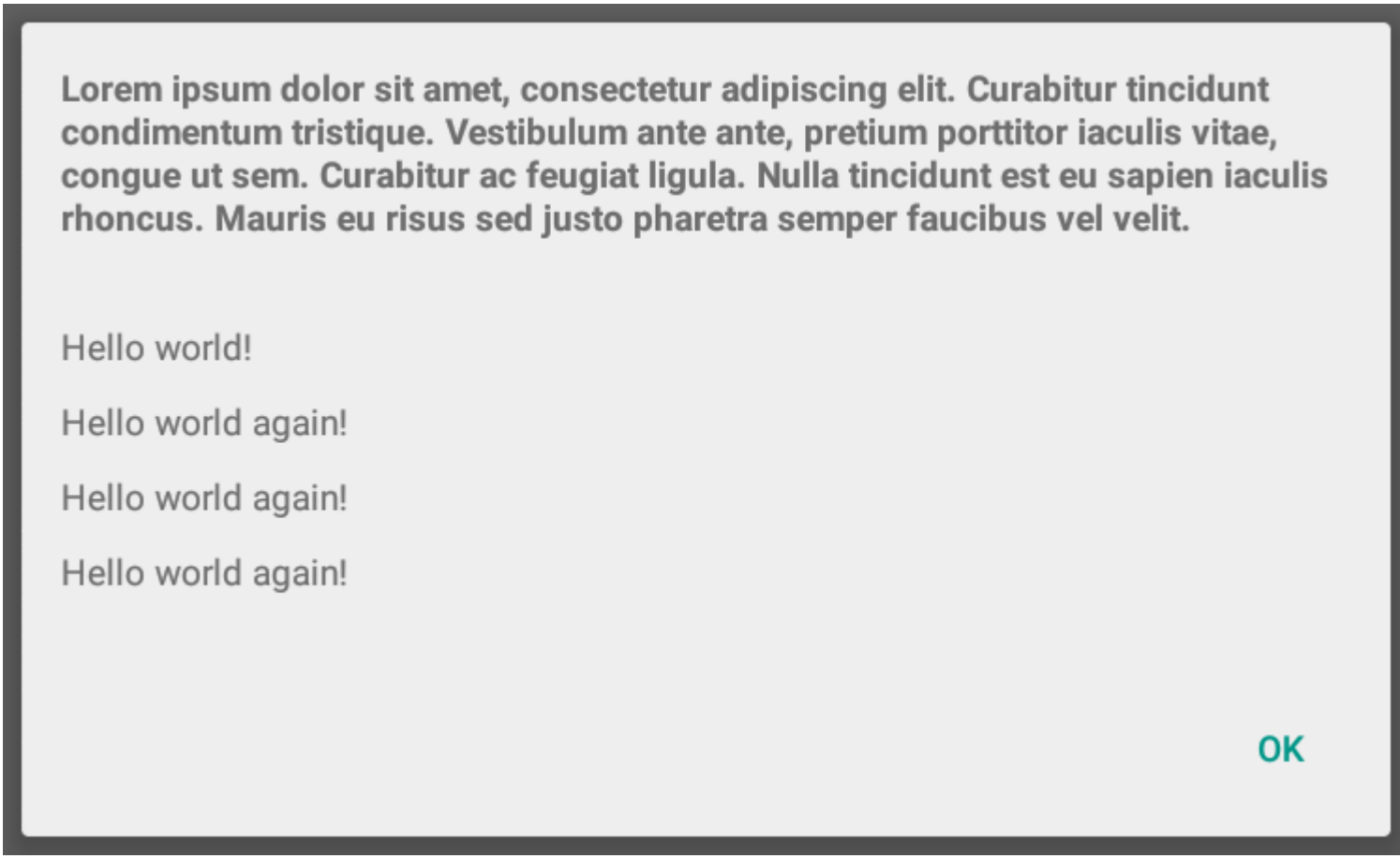
my_dialog_title.xml:

```
<?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>
```



Lire Dialogue en ligne: https://riptutorial.com/fr/android/topic/1225/dialogue

Chapitre 97: Directeur chargé d'emballage

Examples

Récupérer la version de l'application

```
public String getAppVersion() throws PackageManager.NameNotFoundException {
    PackageManager manager = getApplicationContext().getPackageManager();
    PackageInfo info = manager.getPackageInfo(
                            getApplicationContext().getPackageName(),
                            0);

    return info.versionName;
}
```

Nom de version et code de version

Pour obtenir `versionName` et `versionCode` de la `versionCode` actuelle de votre application, vous devez interroger le gestionnaire de packages Android.

```
try {
    // Reference to Android's package manager
    PackageManager packageManager = this.getPackageManager();

    // Getting package info of this application
    PackageInfo info = packageManager.getPackageInfo(this.getPackageName(), 0);

    // Version code
    info.versionCode

    // Version name
    info.versionName

} catch (NameNotFoundException e) {
    // Handle the exception
}
```

Heure d'installation et heure de mise à jour

Pour connaître l'heure à laquelle votre application a été installée ou mise à jour, vous devez interroger le gestionnaire de packages Android.

```
try {
    // Reference to Android's package manager
    PackageManager packageManager = this.getPackageManager();

    // Getting package info of this application
    PackageInfo info = packageManager.getPackageInfo(this.getPackageName(), 0);

    // Install time. Units are as per currentTimeMillis().
    info.firstInstallTime
```

```
    // Last update time. Units are as per currentTimeMillis().
    info.lastUpdateTime

} catch (NameNotFoundException e) {
    // Handle the exception
}
```

Méthode utilitaire utilisant PackageManager

Ici, nous pouvons trouver une méthode utile en utilisant PackageManager,

La méthode ci-dessous aidera à obtenir le nom de l'application en utilisant le nom du paquet

```
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;
}
```

La méthode ci-dessous aidera à obtenir l'icône de l'application en utilisant le nom du paquet,

```
private Drawable getAppIcon(String packageName, Context context) {
    Drawable appIcon = null;
    try {
        appIcon = context.getPackageManager().getApplicationIcon(packageName);
    } catch (PackageManager.NameNotFoundException e) {
    }

    return appIcon;
}
```

La méthode ci-dessous aidera à obtenir la liste des applications installées.

```
public static List<ApplicationInfo> getLaunchIntent(PackageManager packageManager) {

    List<ApplicationInfo> list =
packageManager.getInstalledApplications(PackageManager.GET_META_DATA);

    return list;
}
```

Note: la méthode ci-dessus donnera aussi l’application lanceur.

La méthode ci-dessous aidera à masquer l'icône de l'application du lanceur.

```
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);
    }
}
```

Remarque: Une fois l’appareil éteint et allumé, cette icône revient dans le lanceur.

Lire Directeur chargé d'emballage en ligne: https://riptutorial.com/fr/android/topic/4670/directeur-charge-d-emballage

Chapitre 98: Domaine

Introduction

Realm Mobile Database est une alternative à SQLite. Realm Mobile Database est beaucoup plus rapide qu'un ORM, et souvent plus rapide que SQLite brut.

Avantages

Fonctionnalités hors ligne, requêtes rapides, threading sécurisé, applications multiplates-formes, cryptage, architecture réactive.

Remarques

Lorsque vous utilisez Realm, vous devez vous rappeler que vous ne devez pas transmettre les instances RealmObjects, RealmResults et Realm entre les threads. Si vous avez besoin d'une requête sur un thread donné, ouvrez une instance Realm sur ce thread. À la fin du thread, vous devez fermer le royaume.

> ***NOTE JURIDIQUE*** : Vous comprenez que le Logiciel peut contenir des fonctions cryptographiques pouvant être soumises à des restrictions d'exportation, et vous déclarez et garantissez que **vous ne vous trouvez pas dans un pays** soumis à une restriction ou à un embargo des États-Unis, **notamment Cuba, Iran La Corée, le Soudan, la Syrie ou la région de Crimée** et que vous ne figurez pas sur la liste des personnes refusées, des parties non vérifiées du Département du commerce ou affiliées à une entité restreinte.

Examples

Ajouter un domaine à votre projet

Ajoutez la dépendance suivante à votre fichier `build.gradle` niveau du **projet** .

```
dependencies {
    classpath "io.realm:realm-gradle-plugin:3.1.2"
}
```

Ajoutez ce qui suit en haut de votre fichier `build.gradle` niveau de l' **application** .

```
apply plugin: 'realm-android'
```

Complétez une synchronisation progressive et vous avez maintenant Realm ajouté comme une dépendance à votre projet!

Realm nécessite un appel initial depuis 2.0.0 avant de l'utiliser. Vous pouvez le faire dans votre

classe `Application` ou dans la méthode `onCreate` de votre première activité.

```
Realm.init(this); // added in Realm 2.0.0
Realm.setDefaultConfiguration(new RealmConfiguration.Builder().build());
```

Modèles de domaine

Les modèles de domaine doivent étendre la classe de base `RealmObject` , ils définissent le schéma de la base de données sous-jacente.

Les types de champs pris en charge sont `boolean` , `byte` , caractères `short` , `int` , `long` , `float` , `double` , `String` , `Date` , `byte[]` , liens vers d'autres `RealmObject` `RealmList<T extends RealmModel>` et `RealmList<T extends RealmModel>` .

```
public class Person extends RealmObject {
    @PrimaryKey //primary key is also implicitly an @Index
                //it is required for `copyToRealmOrUpdate()` to update the object.
    private long id;

    @Index //index makes queries faster on this field
    @Required //prevents `null` value from being inserted
    private String name;

    private RealmList<Dog> dogs; //->many relationship to Dog

    private Person spouse; //->one relationship to Person

    @Ignore
    private Calendar birthday; //calendars are not supported but can be ignored

    // getters, setters
}
```

Si vous ajoutez (ou supprimez) un nouveau champ à votre RealmObject (ou si vous ajoutez une nouvelle classe RealmObject ou en supprimez un existant), une **migration** sera nécessaire. Vous pouvez définir `deleteIfMigrationNeeded()` dans votre `RealmConfiguration.Builder` ou définir la migration nécessaire. La migration est également requise lors de l'ajout (ou de la suppression) des `@Required` , `@Index` ou `@PrimaryKey` .

Les relations doivent être définies manuellement, elles ne sont PAS automatiques en fonction des clés primaires.

Depuis la version 0.88.0, il est également possible d'utiliser des champs publics au lieu de champs / getters / setters privés dans les classes RealmObject.

Il est également possible d'implémenter `RealmModel` au lieu d'étendre `RealmObject` , si la classe est également annotée avec `@RealmClass` .

```
@RealmClass
public class Person implements RealmModel {
    // ...
}
```

Dans ce cas, les méthodes telles que `person.deleteFromRealm()` ou `person.addChangeListener()` sont remplacées par `RealmObject.deleteFromRealm(person)` et `RealmObject.addChangeListener(person)` .

Les limitations sont que par un `RealmObject` , seul `RealmObject` peut être étendu, et il n'y a pas de prise en charge pour `transient` champs `final` , `volatile` et `transient` .

Il est important qu'une classe RealmObject *gérée* ne puisse être modifiée que dans une transaction. Un RealmObject *géré* ne peut pas être transmis entre les threads.

Liste de primitives (RealmList)

Realm ne prend actuellement pas en charge le stockage d'une liste de primitives. C'est sur leur liste de tâches (numéro GitHub n ° 575), mais pour le moment, voici une solution de contournement.

Créez une nouvelle classe pour votre type primitif, cela utilise Integer, mais changez-le pour ce que vous voulez stocker.

```
public class RealmInteger extends RealmObject {
    private int val;

    public RealmInteger() {
    }

    public RealmInteger(int val) {
        this.val = val;
    }

    // Getters and setters
}
```

Vous pouvez maintenant utiliser ceci dans votre `RealmObject` .

```
public class MainObject extends RealmObject {

    private String name;
    private RealmList<RealmInteger> ints;

    // Getters and setters
}
```

Si vous utilisez `GSON` pour remplir votre `RealmObject` , vous devrez ajouter un adaptateur de type personnalisé.

```
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();
```

essayer avec les ressources

```
try (Realm realm = Realm.getDefaultInstance()) {
    realm.executeTransaction(new Realm.Transaction() {
            @Override
            public void execute(Realm realm) {
                //whatever Transaction that has to be done
            }
    });
    //No need to close realm in try-with-resources
}
```

Le Try with resources ne peut être utilisé qu'à partir de KITKAT (minSDK 19)

Requêtes triées

Pour trier une requête, au lieu d'utiliser `findAll()` , vous devez utiliser `findAllSorted()` .

```
RealmResults<SomeObject> results = realm.where(SomeObject.class)
                                        .findAllSorted("sortField", Sort.ASCENDING);
```

Remarque:

`sort()` renvoie un tout nouveau RealmResults trié, mais une mise à jour de ce RealmResults le réinitialisera. Si vous utilisez `sort()` , vous devez toujours le trier de nouveau dans votre `RealmChangeListener` , supprimer le `RealmChangeListener` des anciens `RealmResults` et l'ajouter au nouveau `RealmResults` . L'utilisation de `sort()` sur un `RealmResults` renvoyé par une requête asynchrone non encore chargée échouera.

`findAllSorted()` retournera toujours les résultats triés par le champ, même s'il est mis à jour. Il est recommandé d'utiliser `findAllSorted()` .

Requêtes asynchrones

Chaque méthode de requête synchrone (telle que `findAll()` ou `findAllSorted()`) a un équivalent asynchrone (`findAllAsync()` / `findAllSortedAsync()`).

Les requêtes asynchrones déchargent l'évaluation de `RealmResults` sur un autre thread. Pour recevoir ces résultats sur le thread en cours, le thread en cours doit être un thread looper (lisez: les requêtes asynchrones ne fonctionnent généralement que sur le thread d'interface utilisateur).

```
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);
```

Utiliser le domaine avec RxJava

Pour les requêtes, Realm fournit la méthode `realmResults.asObservable()` . L'observation des résultats n'est possible que sur les threads de boucle (généralement le thread d'interface utilisateur).

Pour que cela fonctionne, votre configuration doit contenir les éléments suivants:

```
realmConfiguration = new RealmConfiguration.Builder(context)       //
                         .rxFactory(new RealmObservableFactory()) //
                            //...
                         .build();
```

Ensuite, vous pouvez utiliser vos résultats comme observable.

```
Observable<RealmResults<SomeObject>> observable = results.asObservable();
```

Pour les requêtes asynchrones, vous devez filtrer les résultats par `isLoaded()` afin de recevoir un événement uniquement lorsque la requête a été exécutée. Ce `filter()` n'est pas nécessaire pour les requêtes synchrones (`isLoaded()` renvoie toujours la valeur `true` pour les requêtes de synchronisation).

```
    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));
```

Pour les écritures, vous devez soit utiliser la méthode `executeTransactionAsync()` , soit ouvrir une instance Realm sur le thread d'arrière-plan, exécuter la transaction de manière synchrone, puis fermer l'instance Realm.

```
public Subscription loadObjectsFromNetwork() {
    return objectApi.getObjects()
        .subscribeOn(Schedulers.io())
        .subscribe(response -> {
            try(Realm realmInstance = Realm.getDefaultInstance()) {
                realmInstance.executeTransaction(realm ->
realm.insertOrUpdate(response.objects));
            }
        });
}
```

Utilisation de base

Mise en place d'une instance

Pour utiliser Realm, vous devez d'abord en obtenir une instance. Chaque instance de domaine correspond à un fichier sur le disque. La manière la plus simple d'obtenir une instance est la suivante:

```
// 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();
```

La méthode `Realm.getInstance()` crée le fichier de base de données s'il n'a pas été créé, sinon ouvre le fichier. L'objet `RealmConfiguration` contrôle tous les aspects de la création d'un domaine, qu'il s'agisse d'une base de données `inMemory()` , du nom du fichier Realm, si le domaine doit être effacé si une migration est nécessaire, des données initiales, etc.

Notez que les appels à `Realm.getInstance()` sont comptés (chaque appel incrémente un compteur) et le compteur est décrémenté lorsque `realm.close()` est appelé.

Fermer une instance

Sur les threads d'arrière-plan, il est *très important* de ***fermer*** les instances Realm une fois qu'elles ne sont plus utilisées (par exemple, la transaction est terminée et l'exécution du thread se

termine). Si vous ne fermez pas toutes les instances de Realm sur le thread d'arrière-plan, cela entraîne un épinglage de la version et peut entraîner une augmentation importante de la taille du fichier.

```
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
```

Il convient de noter qu'au-dessus du niveau 19 de l'API, vous pouvez remplacer ce code par ceci:

```
try(Realm realm = Realm.getDefaultInstance()) {
    // ...
}
```

Des modèles

La prochaine étape serait la création de vos modèles. Ici, une question pourrait être posée: "Qu'est-ce qu'un modèle?". Un modèle est une structure qui définit les propriétés d'un objet stocké dans la base de données. Par exemple, dans ce qui suit, nous modélisons un livre.

```
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;
    }
}
```

Notez que vos modèles doivent étendre la classe RealmObject. La clé primaire est également spécifiée par l'annotation `@PrimaryKey` . Les clés primaires peuvent être nulles, mais un seul élément peut avoir la valeur `null` tant que clé primaire. Vous pouvez également utiliser l'annotation `@Ignore` pour les champs à ne pas `@Ignore` sur le disque:

```
@Ignore
private String isbn;
```

Insérer ou mettre à jour des données

Pour stocker un objet livre dans votre instance de base de données Realm, vous pouvez d'abord créer une instance de votre modèle, puis la stocker dans la base de données via la méthode `copyToRealm` . Pour créer ou mettre à jour, vous pouvez utiliser `copyToRealmOrUpdate` . (Une alternative plus rapide est la `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);
    }
});
```

Notez que toutes les modifications apportées aux données doivent avoir lieu dans une transaction. Une autre façon de créer un objet consiste à utiliser le modèle suivant:

```
Book book = realm.createObject(Book.class, primaryKey);
...
```

Interroger la base de données

- Tous les livres:

```
RealmResults<Book> results = realm.where(Book.class).findAll();
```

- Tous les livres ayant un identifiant supérieur à 10:

```
RealmResults<Book> results = realm.where(Book.class)
                                  .greaterThan("id", 10)
                                  .findAll();
```

- Livres de `'Taylor Swift'` ou `'%Peter%'` :

```
RealmResults<Book> results = realm.where(Book.class)
                                  .beginGroup()
                                      .equalTo("author", "Taylor Swift")
                                      .or()
                                      .contains("author", "Peter")
                                  .endGroup().findAll();
```

Supprimer un objet

Par exemple, nous voulons supprimer tous les livres de Taylor Swift:

```
// Start of transaction
realm.executeTransaction(new Realm.Transaction() {
    @Override
    public void execute(Realm realm) {
        // First Step: Query all Taylor Swift books
        RealmResults<Book> results = ...

        // Second Step: Delete elements in Realm
        results.deleteAllFromRealm();
    }
});
```

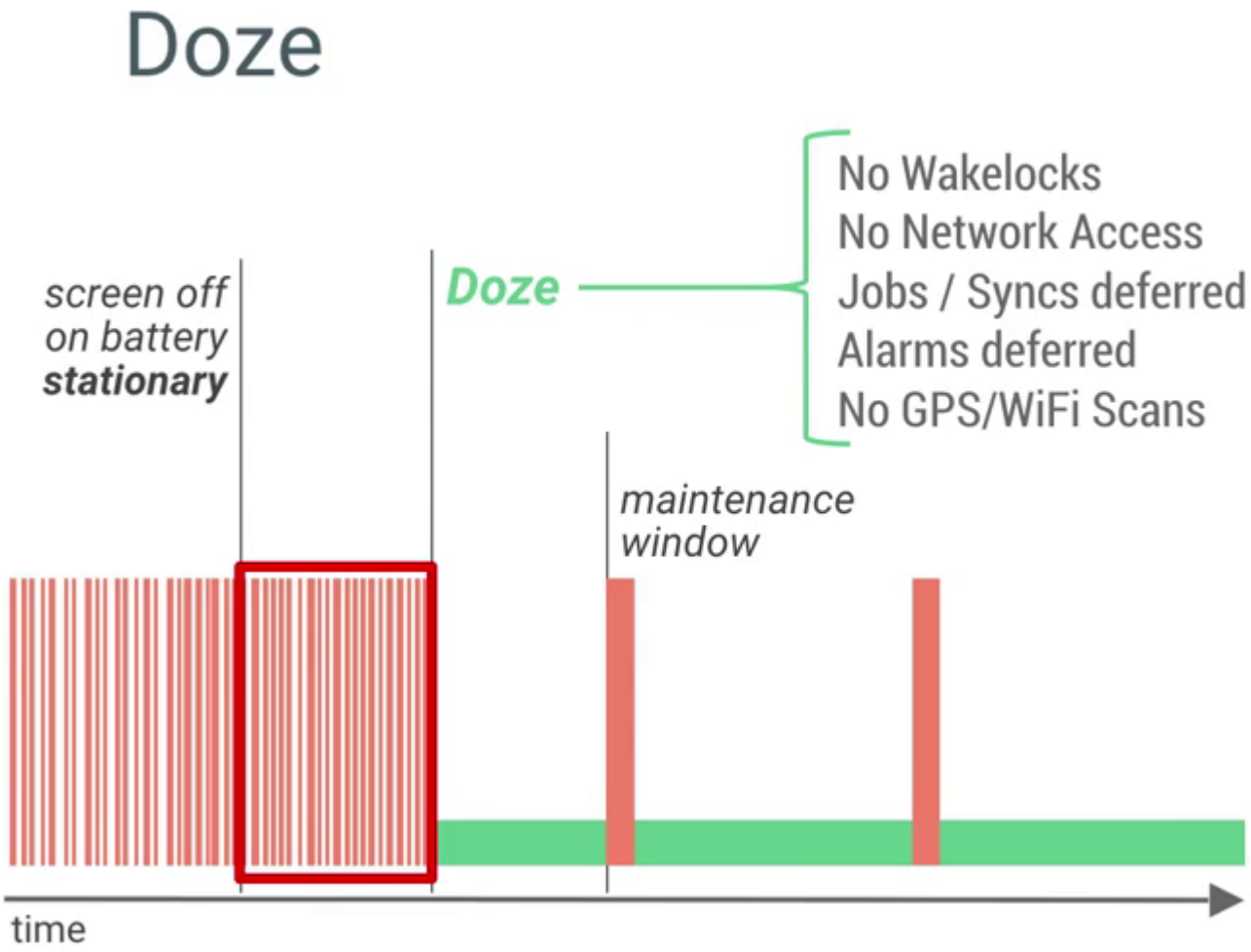
Lire Domaine en ligne: https://riptutorial.com/fr/android/topic/3187/domaine

Chapitre 99: Doze Mode

Remarques

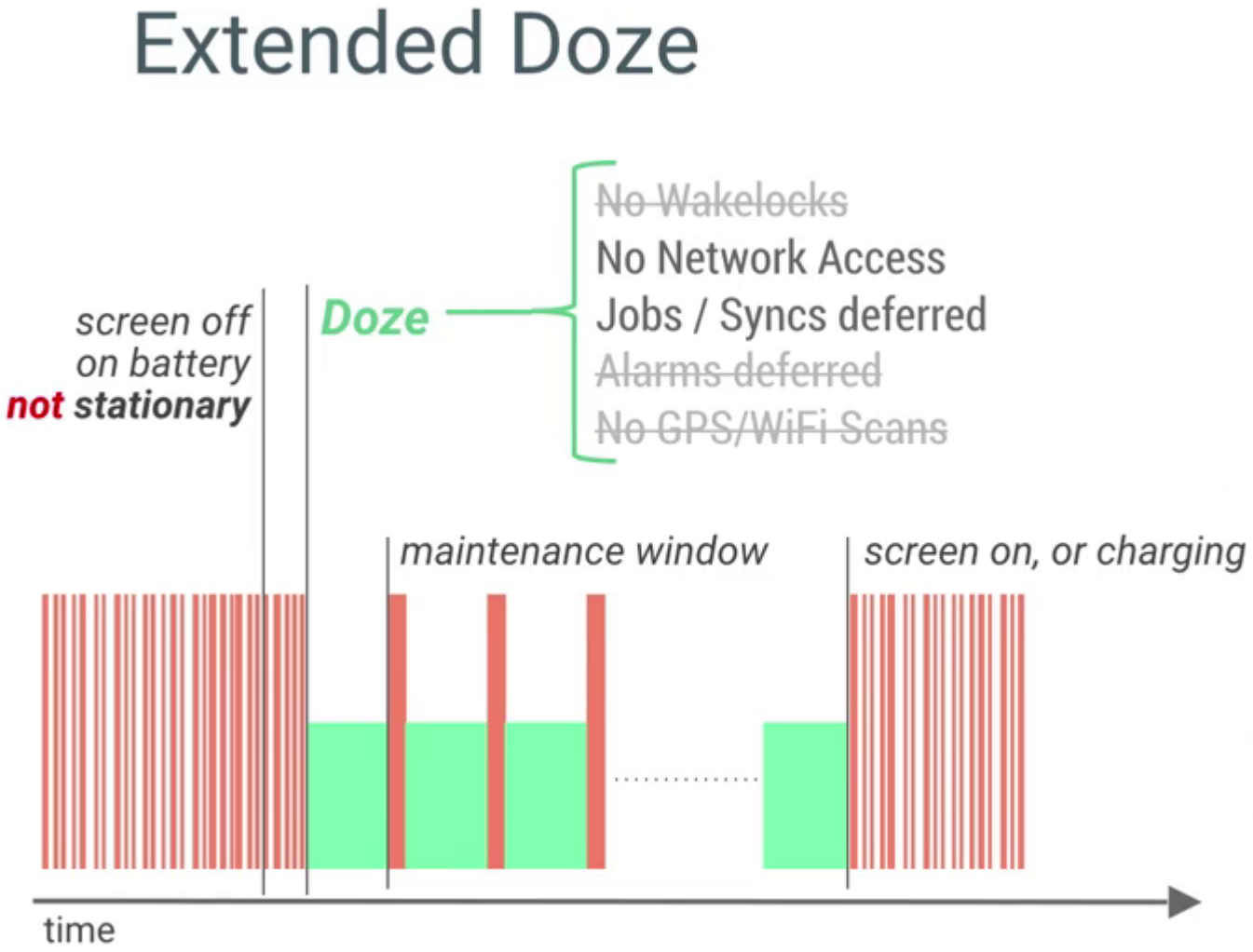
Le mode Doze est un ensemble de modifications et de règles qui mettent votre téléphone en veille lorsqu'il est inactif.

Sur Android 6.0 Marshmallow: le mode Doze est activé lorsque l'écran est éteint, l'appareil est à l'arrêt et fonctionne sur batterie.



Comme vous pouvez le voir dans le diagramme ci-dessus, lorsque le mode Doze est activé, l'appareil ne reçoit pas de wakelocks, d'accès réseau, de travaux / synchronisations, d'alarmes, de scans GPS / Wi-Fi.

Sur Android 7.0 Nougat: Imaginez que votre téléphone soit dans votre poche (l'écran est éteint, qu'il fonctionne sur batterie, mais qu'il ne soit pas immobile), vous pouvez également obtenir les fonctionnalités du mode Doze, non? C'est pourquoi Google a annoncé le mode Doze étendu: il fonctionne lorsque l'écran est éteint, mais pas immobile.



Comme vous pouvez le voir dans ce diagramme, seuls les accès réseau et les travaux / synchronisations sont désactivés. Notez que Doze étendu ne remplace pas le premier mode Doze. Ils travaillent ensemble, en fonction de l'état du téléphone (fixe ou non). Voici les distinctions:

	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

Les développeurs doivent être conscients que:

- Doze peut conserver un accès temporaire à wakelock et au réseau pour les messages GCM (Google Cloud Messaging) prioritaires (dans les cas où l'utilisateur a besoin d'une notification immédiate);
- Les services de premier plan (tels que la lecture de musique) continueront de fonctionner.

Vous pouvez trouver plus d'informations ici: https://developer.android.com/training/monitoring-device-state/doze-standby.html

Examples

Exclure l'application de l'utilisation du mode veille

1. Ouvrir les paramètres du téléphone
2. batterie ouverte
3. ouvrir le menu et sélectionner "optimisation de la batterie"
4. dans le menu déroulant, sélectionnez "toutes les applications"
5. sélectionnez l'application que vous souhaitez ajouter à la liste blanche
6. sélectionnez "ne pas optimiser"

Maintenant, cette application affichera sous des applications non optimisées.

Une application peut vérifier si elle est en liste blanche en appelant `isIgnoringBatteryOptimizations()`

Liste blanche d'une application Android par programmation

La liste blanche ne désactivera pas le mode veille pour votre application, mais vous pouvez le faire en utilisant les verrous de réseau et de veille.

La liste blanche d'une application Android par programmation peut être effectuée comme suit:

```
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);
}
```

Le résultat du démarrage de l'activité ci-dessus peut être vérifié par le code suivant:

```
@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
        }
    }
}
```

Lire Doze Mode en ligne: https://riptutorial.com/fr/android/topic/4719/doze-mode

Chapitre 100: Écran partagé / Activités multi-écrans

Examples

Split Screen introduit dans Android Nougat implémenté.

Définissez cet attribut dans votre manifeste ou élément pour activer ou désactiver l'affichage multi-fenêtre:

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

Si cet attribut est défini sur true, l'activité peut être lancée en modes écran partagé et forme libre. Si l'attribut est défini sur false, l'activité ne prend pas en charge le mode multi-fenêtre. Si cette valeur est fausse et que l'utilisateur tente de lancer l'activité en mode multi-fenêtre, l'activité prend la totalité de l'écran.

Si votre application cible le niveau 24 de l'API, mais que vous ne spécifiez pas de valeur pour cet attribut, la valeur de l'attribut est définie par défaut sur true.

Le code suivant montre comment spécifier la taille et l'emplacement par défaut d'une activité et sa taille minimale lorsque l'activité est affichée en mode libre:

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

Fonctions désactivées en mode multi-fenêtres

Certaines fonctionnalités sont désactivées ou ignorées lorsqu'un périphérique est en mode multi-fenêtres, car elles ne sont pas adaptées à une activité pouvant partager l'écran de l'appareil avec d'autres activités ou applications. Ces fonctionnalités incluent:

1. Certaines options de personnalisation de l'interface utilisateur du système sont désactivées. Par exemple, les applications ne peuvent pas masquer la barre d'état si elles ne s'exécutent pas en mode plein écran.

2. Le système ignore les modifications apportées à l'attribut **Android: screenOrientation** .

Si votre application cible le niveau 23 ou inférieur de l'API

Si votre application cible le niveau 23 ou inférieur de l'API et que l'utilisateur tente d'utiliser

l'application en mode multi-fenêtre, le système redimensionne de force l'application, sauf si l'application déclare une orientation fixe.

Si votre application ne déclare pas une orientation fixe, vous devez lancer votre application sur un appareil fonctionnant sous Android 7.0 ou version ultérieure et tenter de placer l'application en mode écran partagé. Vérifiez que l'expérience utilisateur est acceptable lorsque l'application est redimensionnée de force.

Si l'application déclare une orientation fixe, vous devez essayer de mettre l'application en mode multi-fenêtres. Vérifiez que l'application reste en mode plein écran.

Lire Écran partagé / Activités multi-écrans en ligne:
https://riptutorial.com/fr/android/topic/7130/ecran-partage---activites-multi-ecrans

Chapitre 101: Écrans de support avec différentes résolutions, tailles

Remarques

Termes et concepts

Taille de l'écran

Taille physique réelle, mesurée en diagonale de l'écran. Pour plus de simplicité, Android regroupe toutes les tailles d'écran réelles en quatre tailles généralisées: petite, normale, grande et extra-large.

Densité de l'écran

La quantité de pixels dans une zone physique de l'écran; généralement appelé dpi (points par pouce). Par exemple, un écran à faible densité présente moins de pixels dans une zone physique donnée, comparé à un écran à densité "normale" ou "élevée". Pour plus de simplicité, Android regroupe toutes les densités d'écran réelles en six densités généralisées: faible, moyenne, élevée, très élevée, extra-haute et extra-extra-haute.

Orientation

L'orientation de l'écran du point de vue de l'utilisateur. Ceci est soit paysage ou portrait, ce qui signifie que le format de l'écran est soit large ou grand, respectivement. Sachez que non seulement les différents périphériques fonctionnent par défaut dans des orientations différentes, mais que l'orientation peut changer à l'exécution lorsque l'utilisateur fait pivoter le périphérique. Résolution Le nombre total de pixels physiques sur un écran. Lorsque vous ajoutez un support pour plusieurs écrans, les applications ne fonctionnent pas directement avec la résolution. les applications ne devraient concerner que la taille et la densité de l'écran, comme spécifié par les groupes de taille et de densité généralisés. Pixel indépendant de la densité (dp) Unité de pixels virtuelle que vous devez utiliser lors de la définition de la disposition de l'interface utilisateur, pour exprimer les dimensions ou la position de la disposition de manière indépendante de la densité. Le pixel indépendant de la densité est équivalent à un pixel physique sur un écran de 160 ppp, qui est la densité de base supposée par le système pour un écran de densité "moyenne". Lors de l'exécution, le système gère de manière transparente toute mise à l'échelle des unités dp, en fonction des besoins, en fonction de la densité réelle de l'écran utilisé. La conversion des unités dp en pixels est simple: px = dp * (dpi / 160). Par exemple, sur un écran de 240 ppp, 1 dp correspond à 1,5 pixel physique. Vous devez toujours utiliser les unités dp lors de la définition de

l'interface utilisateur de votre application, afin de garantir un affichage correct de votre interface utilisateur sur des écrans présentant des densités différentes.

Unités

- ### px

  Pixels - correspond aux pixels réels à l'écran.

- ### dans

  Pouces - en fonction de la taille physique de l'écran. 1 pouce = 2,54 centimètres

- ### mm

  Millimètres - basé sur la taille physique de l'écran.

- ### pt

  Points - 1/72 de pouce basé sur la taille physique de l'écran.

- ### dp ou tremper

  Pixels indépendants de la densité - une unité abstraite basée sur la densité physique de l'écran. Ces unités sont relatives à un écran de 160 ppp, donc un pd est un pixel sur un écran de 160 ppp. Le rapport dp-pixel changera avec la densité de l'écran, mais pas nécessairement en proportion directe. Note: Le compilateur accepte à la fois "dip" et "dp", bien que "dp" soit plus compatible avec "sp".

- ### sp

  Pixels indépendants de l'échelle - c'est comme l'unité dp, mais elle est également adaptée à la taille de la police de l'utilisateur. Il est recommandé d'utiliser cet appareil pour spécifier les tailles de police afin qu'elles soient adaptées à la fois à la densité de l'écran et aux préférences de l'utilisateur. De comprendre l'indépendance de la densité dans Android:

Unité	La description	Unités par pouce physique	Densité Indépendante	Même taille physique sur chaque écran
px	Des pixels	Varie	Non	Non

Unité	La description	Unités par pouce physique	Densité Indépendante	Même taille physique sur chaque écran
dans	Pouces	1	Oui	Oui
mm	Millimètres	25.4	Oui	Oui
pt	Points	72	Oui	Oui
dp	Densité Indépendante Pixels	~ 160	Oui	Non
sp	Échelle des pixels indépendants	~ 160	Oui	Non

Les références:

- https://developer.android.com/guide/practices/screens_support.html
- http://developer.android.com/guide/topics/resources/more-resources.html

Examples

Utiliser des qualificateurs de configuration

Android prend en charge plusieurs qualificateurs de configuration qui vous permettent de contrôler la manière dont le système sélectionne vos ressources alternatives en fonction des caractéristiques de l'écran de périphérique actuel. Un qualificatif de configuration est une chaîne que vous pouvez ajouter à un répertoire de ressources dans votre projet Android et spécifie la configuration pour laquelle les ressources internes sont conçues.

Pour utiliser un qualificatif de configuration:

1. Créez un nouveau répertoire dans le répertoire res / de votre projet et nommez-le au format suivant: `<resources_name>-<qualifier>` . `<resources_name>` nom_ressources `<resources_name>` est le nom de ressource standard (tel que dessinable ou mise en page).
2. `<qualifier>` est un qualificatif de configuration, spécifiant la configuration d'écran pour laquelle ces ressources doivent être utilisées (telles que hdpi ou xlarge).

Par exemple, les répertoires de ressources d'application suivants proposent différents modèles de mise en page pour différentes tailles d'écran et différents tirables. Utilisez les `mipmap/` dossiers pour les icônes du lanceur.

```
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
```

Conversion de dp et sp en pixels

Lorsque vous avez besoin de définir une valeur de pixel pour quelque chose comme `Paint.setTextSize` mais que vous souhaitez tout de même la mettre à l'échelle en fonction du périphérique, vous pouvez convertir les valeurs dp et sp.

```
DisplayMetrics metrics = Resources.getSystem().getDisplayMetrics();
float pixels = TypedValue.applyDimension(TypedValue.COMPLEX_UNIT_SP, 12f, metrics);

DisplayMetrics metrics = Resources.getSystem().getDisplayMetrics();
float pixels = TypedValue.applyDimension(TypedValue.COMPLEX_UNIT_DIP, 12f, metrics);
```

Vous pouvez également convertir une ressource de dimension en pixels si vous avez un contexte dans lequel charger la ressource.

```
<?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);
```

Taille du texte et différentes tailles d'écran Android

Parfois, il vaut mieux n'avoir que trois options

```
 style="@android:style/TextAppearance.Small"
 style="@android:style/TextAppearance.Medium"
 style="@android:style/TextAppearance.Large"
```

Utilisez petit et grand pour différencier de la taille de l'écran normal.

```
<TextView
        android:id="@+id/TextViewTopBarTitle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        style="@android:style/TextAppearance.Small"/>
```

Pour la normale, vous n'avez rien à spécifier.

```
<TextView
        android:id="@+id/TextViewTopBarTitle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>
```

Grâce à cela, vous pouvez éviter de tester et de spécifier des dimensions pour différentes tailles d'écran.

Lire Écrans de support avec différentes résolutions, tailles en ligne: https://riptutorial.com/fr/android/topic/1086/ecrans-de-support-avec-differentes-resolutions--tailles

Chapitre 102: Ecriture des tests de l'interface utilisateur - Android

Introduction

L'objectif de ce document est de représenter les objectifs et les moyens d'écrire l'interface utilisateur et les tests d'intégration. Espresso et UIAutomator sont fournis par Google, vous devez donc vous concentrer sur ces outils et leurs enveloppes respectives, par exemple Appium, Spoon, etc.

Syntaxe

- **Ressource de ralenti**
- String getName () - Retourne le nom de la ressource inactive (utilisée pour la journalisation et l'idempotence de l'enregistrement).
- boolean isIdleNow () - Retourne true si la ressource est actuellement inactive.
- annulez registerIdleTransitionCallback (rappel de IdlingResource.ResourceCallback) - Enregistre le IDlingResource.ResourceCallback donné avec la ressource

Remarques

Règles JUnit:

Comme vous pouvez le voir dans l'exemple de MockWebServer et ActivityTestRule, ils entrent tous dans la catégorie des règles JUnit que vous pouvez créer vous-même et qui doivent ensuite être exécutés pour chaque test définissant son comportement @see: https://github.com/junit-team/junit4/ wiki / règles

Appium

Paramètres

Puisque les paramètres ont des problèmes pour les placer ici jusqu'à ce que le bug de la documentation soit résolu:

Paramètre	Détails
Activité de classeClass	quelle activité commencer
initialTouchMode	si l'activité est mise en mode tactile au démarrage: https://android-developers.blogspot.de/2008/12/touch-mode.html

Paramètre	Détails
launchActivity	true si l'activité doit être lancée une fois par méthode de test. Il sera lancé avant la première méthode Before et terminé après la dernière méthode After.

Examples

Exemple de MockWebServer

Dans le cas où vos activités, fragments et interface utilisateur requièrent un traitement en arrière-plan, une bonne chose à faire est un MockWebServer qui fonctionne localement sur un appareil Android qui apporte un environnement fermé et testable à votre interface utilisateur.

https://github.com/square/okhttp/tree/master/mockwebserver

La première étape consiste à inclure la dépendance graduelle:

```
testCompile 'com.squareup.okhttp3:mockwebserver:(insert latest version)'
```

Maintenant, les étapes pour exécuter et utiliser le serveur simulé sont les suivantes:

- créer un objet serveur de simulation
- lancez-le à une adresse et à un port spécifiques (généralement localhost: numéro de port)
- Réponses de mise en file d'attente pour des demandes spécifiques
- commence le test

Ceci est bien expliqué dans la page github du mockwebserver mais dans notre cas, nous voulons quelque chose de plus agréable et réutilisable pour tous les tests, et les règles de JUnit seront très utiles ici:

```
/**
 *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”);
        }
    }
}
```

Supposons maintenant que nous avons exactement la même activité que dans l'exemple précédent, mais dans ce cas, lorsque nous appuierons sur le bouton, l'application ira chercher quelque chose sur le réseau, par exemple: https://someapi.com/name

Cela renverrait une chaîne de texte qui serait concaténée dans le texte du snack, par exemple NAME + texte que vous avez tapé.

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

Je suggère que vous enveloppiez le répartiteur dans une sorte de générateur afin de pouvoir facilement enchaîner et ajouter de nouvelles réponses pour vos écrans. par exemple

```
 return newDispatcherBuilder()
            .withSerializedJSONBody("/authenticate", Mocks.getAuthenticationResponse())
            .withSerializedJSONBody("/getUserInfo", Mocks.getUserInfo())
            .withSerializedJSONBody("/checkNotBot", Mocks.checkNotBot());
```

IdlingResource

La puissance des ressources inactives réside dans le fait qu'il n'est pas nécessaire d'attendre que certains traitements de l'application (mise en réseau, calculs, animations, etc.) se terminent avec

`sleep()` , ce qui entraîne une certaine fragilité et / ou prolonge les tests. La documentation officielle est disponible ici .

la mise en oeuvre

Il y a trois choses à faire lors de l'implémentation de l'interface `IdlingResource` :

- **`getName()`** - Retourne le nom de votre ressource inactive.
- **`isIdleNow()`** - Vérifie si votre objet xyz, opération, etc. est inactif pour le moment.
- **`registerIdleTransitionCallback`** (callback `IdlingResource.ResourceCallback`) - Fournit un rappel que vous devez appeler lorsque votre objet passe au repos.

Vous devez maintenant créer votre propre logique et déterminer à quel moment votre application est inactive et non, car cela dépend de l'application. Vous trouverez ci-dessous un exemple simple, juste pour montrer comment cela fonctionne. Il existe d'autres exemples en ligne, mais la mise en œuvre d'applications spécifiques apporte des implémentations de ressources spécifiques au ralenti.

REMARQUES

- Il y a eu quelques exemples de Google où ils ont placé `IdlingResources` dans le code de l'application. **Ne faites pas cela.** Ils l'ont probablement placé juste pour montrer comment ils fonctionnent.
- Garder votre code propre et maintenir un principe de responsabilité unique dépend de vous!

Exemple

Disons que vous avez une activité qui fait des choses bizarres et que le chargement du fragment prend beaucoup de temps, ce qui fait que vos tests Espresso échouent car vous ne pouvez pas trouver les ressources de votre fragment (vous devez changer la façon dont votre activité est créée et quand pour l'accélérer). Mais dans tous les cas, pour rester simple, l'exemple suivant montre à quoi cela doit ressembler.

Notre exemple de ressource d'inactivité obtiendrait deux objets:

- La **balise** du fragment que vous devez trouver et en attente d'attacher à l'activité.
- Un objet **FragmentManager** utilisé pour trouver le fragment.

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

Maintenant que vous avez votre `IdlingResource` écrite, vous devez l'utiliser quelque part correctement?

Usage

Sautons toute la configuration de la classe de test et examinons à quoi ressemblerait un scénario de test:

```
@Test
public void testSomeFragmentText() {
    mActivityTestRule.launchActivity(null);

    //creating the idling resource
    IdlingResource fragmentLoadedIdlingResource = new
FragmentIdlingResource(mActivityTestRule.getActivity().getSupportFragmentManager(),
SomeFragmentText.TAG);
    //registering the idling resource so espresso waits for it
    Espresso.registerIdlingResources(idlingResource1);
    onView(withId(R.id.txtHelloWorld)).check(matches(withText(helloWorldText)));

    //lets cleanup after ourselves
    Espresso.unregisterIdlingResources(fragmentLoadedIdlingResource);
}
```

Combinaison avec la règle JUnit

Ce n'est pas difficile; Vous pouvez également appliquer la ressource inactive sous la forme d'une règle de test JUnit. Par exemple, disons que vous avez un SDK contenant Volley et que vous voulez qu'Espresso l'attende. Au lieu de parcourir chaque scénario de test ou de l'appliquer dans la configuration, vous pouvez créer une règle JUnit et écrire simplement:

```
@Rule
public final SDKIdlingRule mSdkIdlingRule = new
SDKIdlingRule(SDKInstanceHolder.getInstance());
```

Maintenant, puisque ceci est un exemple, ne le prenez pas pour acquis; tout le code ici est imaginaire et utilisé uniquement à des fins de démonstration:

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

Lire Ecriture des tests de l'interface utilisateur - Android en ligne:
https://riptutorial.com/fr/android/topic/3530/ecriture-des-tests-de-l-interface-utilisateur---android

Chapitre 103: Éditer le texte

Examples

Travailler avec EditTexts

EditText est le widget de saisie de texte standard dans les applications Android. Si l'utilisateur doit entrer du texte dans une application, c'est le principal moyen de le faire.

Éditer le texte

De nombreuses propriétés importantes peuvent être définies pour personnaliser le comportement d'un EditText. Plusieurs d'entre eux sont énumérés ci-dessous. Consultez le guide des champs de texte officiels pour encore plus de détails sur les champs de saisie.

Usage

Un EditText est ajouté à une présentation avec tous les comportements par défaut avec le code XML suivant:

```
<EditText
    android:id="@+id/et_simple"
    android:layout_height="wrap_content"
    android:layout_width="match_parent">
</EditText>
```

Notez qu'un EditText est simplement une mince extension de TextView et hérite de toutes les mêmes propriétés.

Récupération de la valeur

Pour obtenir la valeur du texte entré dans un EditText, procédez comme suit:

```
EditText simpleEditText = (EditText) findViewById(R.id.et_simple);
String strValue = simpleEditText.getText().toString();
```

Personnalisation de l'entrée supplémentaire

Nous pourrions vouloir limiter l'entrée à une seule ligne de texte (éviter les nouvelles lignes):

```
<EditText
  android:singleLine="true"
  android:lines="1"
/>
```

Vous pouvez limiter les caractères pouvant être saisis dans un champ en utilisant l'attribut digits:

```
<EditText
  android:inputType="number"
  android:digits="01"
/>
```

Cela limiterait les chiffres entrés à seulement "0" et "1". Nous pourrions vouloir limiter le nombre total de caractères avec:

```
<EditText
  android:maxLength="5"
/>
```

En utilisant ces propriétés, nous pouvons définir le comportement d'entrée attendu pour les champs de texte.

Réglage des couleurs

Vous pouvez ajuster la couleur d'arrière-plan de mise en évidence du texte sélectionné dans un EditText avec la propriété `android:textColorHighlight` :

```
<EditText
    android:textColorHighlight="#7cff88"
/>
```

Affichage des indicateurs d'espace réservé

Vous souhaiterez peut-être définir l'indicateur pour le contrôle EditText afin d'inviter un utilisateur pour une entrée spécifique avec:

```
<EditText
    ...
    android:hint="@string/my_hint">
</EditText>
```

Astuces

Changer la couleur de la ligne du bas

En supposant que vous utilisez la bibliothèque AppCompat, vous pouvez remplacer les styles colorControlNormal, colorControlActivated et colorControlHighlight:

```
<style name="Theme.App.Base" parent="Theme.AppCompat.Light.DarkActionBar">
    <item name="colorControlNormal">#d32f2f</item>
    <item name="colorControlActivated">#ff5722</item>
    <item name="colorControlHighlight">#f44336</item>
</style>
```

Si vous ne voyez pas ces styles appliqués dans un DialogFragment, il existe un bogue connu lors de l'utilisation de LayoutInflater passé dans la méthode onCreateView ().

Le problème a déjà été résolu dans la bibliothèque AppCompat v23. Voir ce guide sur la mise à

niveau. Une autre solution temporaire consiste à utiliser le composant inflater layout au lieu de celui passé dans la méthode onCreateView ():

```
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
        View view = getActivity().getLayoutInflater().inflate(R.layout.dialog_fragment,
container);
  }
```

Écoute de l'entrée EditText

Consultez les cliffnotes des écouteurs d'événement de base pour savoir comment écouter les modifications apportées à un EditText et effectuer une action lorsque ces modifications se produisent.

Affichage des commentaires sur les étiquettes flottantes

Traditionnellement, EditText masque le message de conseil (expliqué ci-dessus) après que l'utilisateur commence à taper. De plus, tout message d'erreur de validation devait être géré manuellement par le développeur.

Avec `TextInputLayout` vous pouvez configurer une étiquette flottante pour afficher les indicateurs et les messages d'erreur. Vous pouvez trouver plus de détails ici .

Personnalisation du InputType

Les champs de texte peuvent avoir différents types d'entrée, tels que le numéro, la date, le mot de passe ou l'adresse électronique. Le type détermine quels types de caractères sont autorisés dans le champ et peut inviter le clavier virtuel à optimiser sa disposition pour les caractères fréquemment utilisés.

Par défaut, tout contenu de texte dans un contrôle `EditText` est affiché en texte brut. En définissant l'attribut `inputType` , nous pouvons faciliter la saisie de différents types d'informations, tels que les numéros de téléphone et les mots de passe:

```
<EditText
    ...
    android:inputType="phone">
</EditText>
```

Les types d’entrée les plus courants sont les suivants:

Type	La description
textUri	Texte qui sera utilisé comme URI
textEmailAddress	Texte qui sera utilisé comme adresse e-mail
textPersonName	Texte qui est le nom d'une personne

Type	La description
textPassword	Texte qui est un mot de passe qui devrait être masqué
nombre	Un champ uniquement numérique
téléphone	Pour entrer un numéro de téléphone
rendez-vous amoureux	Pour entrer une date
temps	Pour entrer un temps
textMultiLine	Autoriser plusieurs lignes de texte dans le champ

L' `android:inputType` vous permet également de spécifier certains comportements de clavier, tels que la mise en majuscule de tous les nouveaux mots ou l'utilisation de fonctionnalités telles que la `android:inputType` automatique et les suggestions d'orthographe.
Voici certaines des valeurs de type d'entrée courantes qui définissent les comportements du clavier:

Type	La description
textCapSentences	Clavier de texte normal qui met en majuscule la première lettre de chaque nouvelle phrase
textCapWords	Clavier de texte normal qui capitalise chaque mot. Bon pour les titres ou les noms de personnes
textAutoCorrect	Clavier de texte normal qui corrige les mots mal orthographiés

Vous pouvez définir plusieurs attributs `inputType` si nécessaire (séparés par '|').
Exemple:

```
<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" />
```

Vous pouvez voir une liste de tous les types d'entrées disponibles ici .

Attribut `inputype`

attribut `inputype` dans le widget `EditText` : *(testé sur Android 4.4.3 et 2.3.3)*

```
<EditText android:id="@+id/et_test" android:inputType="?????"/>
```

textLongMessage = Clavier: alphabet / default. Bouton Entrée: Envoyer / Suivant. Emotion: oui. Cas: minuscule. Suggestion: oui. Ajouter. carbonise: **et.** et tout

textFilter = Clavier: alphabet / default. Bouton Entrée: Envoyer / Suivant. Emotion: oui. Cas: minuscule. **Suggestion: non** . Ajouter. carbonise: **et.** et tout

textCapWords = Clavier: alphabet / default. Bouton Entrée: Envoyer / Suivant. Emotion: oui. **Affaire: Affaire Camel** . Suggestion: oui. Ajouter. carbonise: **et.** et tout

textCapSentences = Clavier: alphabet / default. Bouton Entrée: Envoyer / Suivant. Emotion: oui. **Cas: cas de phrase** . Suggestion: oui. Ajouter. carbonise: **et.** et tout

time = Clavier: numérique. Bouton Entrée: Envoyer / Suivant. Emotion: non. Cas: -. **Suggestion: non** . Ajouter. caractères::

textMultiLine = Clavier: alphabet / default. **Bouton Enter: nextline** . Emotion: oui. Cas: minuscule. Suggestion: oui. Ajouter. carbonise: **et.** et tout

number = Clavier: numérique . Bouton Entrée: Envoyer / Suivant. Emotion: non. Cas: -. Suggestion: non. **Ajouter. caractères: rien**

textEmailAddress = Clavier: alphabet / default. Bouton Entrée: Envoyer / Suivant. **Emotion: non** . Cas: minuscule. **Suggestion: non** . Ajouter. caractères: @ et **.** et tout

(Aucun type) = Clavier: alphabet / default. **Bouton Enter: nextline** . Emotion: oui. Cas: minuscule. Suggestion: oui. Ajouter. carbonise: **et.** et tout

textPassword = Clavier: alphabet / default. Bouton Entrée: Envoyer / Suivant. Emotion: non. Cas: minuscule. **Suggestion: non** . Ajouter. carbonise: **et.** et tout

text = Clavier: Clavier: alphabet / default. Bouton Entrée: Envoyer / Suivant. Emotion: oui. Cas: minuscule. Suggestion: oui. Ajouter. carbonise: **et.** et tout

textShortMessage = Clavier: alphabet / default. **Bouton Entrée: émotion** . Emotion: oui. Cas: minuscule. Suggestion: oui. Ajouter. carbonise: **et.** et tout

textUri = Clavier: alphabet / default. Bouton Entrée: Envoyer / Suivant. Emotion: non. Cas: minuscule. **Suggestion: non** . Ajouter. caractères: **/** et **.** et tout

textCapCharacters = Clavier: alphabet / default. Bouton Entrée: Envoyer / Suivant. Emotion: oui. **Case: UPPERCASE** . Suggestion: oui. Ajouter. carbonise: **et.** et tout

phone = Clavier: numérique . Bouton Entrée: Envoyer / Suivant. Emotion: non. Cas: -. **Suggestion: non** . Ajouter. caractères: *** #. - / () WPN, + **

textPersonName = Clavier: alphabet / default. Bouton Entrée: Envoyer / Suivant. Emotion: oui. Cas: minuscule. Suggestion: oui. Ajouter. carbonise: **et.** et tout

Remarque: le paramètre de mise en `Auto-capitalization` modifiera le comportement par défaut.

Remarque 2: Dans le `Numeric keyboard` , TOUS les numéros sont en anglais 1234567890.

Remarque 3: `Correction/Suggestion` paramètre `Correction/Suggestion` changera le comportement par défaut.

Cacher SoftKeyboard

Le masquage du tableau de touches est généralement une **exigence de base** lorsque vous travaillez avec EditText. Le panneau de touches par *défaut* ne peut être fermé qu'en appuyant sur le bouton et la plupart des développeurs utilisent InputMethodManager pour forcer Android à masquer le clavier virtuel appelant hideSoftInputFromWindow et en transmettant le jeton de la fenêtre contenant votre vue focalisée. Le code pour effectuer les opérations suivantes:

```
public void hideSoftKeyboard()
{
        InputMethodManager inputMethodManager = (InputMethodManager)
getSystemService(Activity.INPUT_METHOD_SERVICE);
        inputMethodManager.hideSoftInputFromWindow(getCurrentFocus().getWindowToken(), 0);
}
```

Le code est direct, mais un autre problème majeur est que la fonction hide doit être appelée lorsqu'un événement se produit. Que faire lorsque vous avez besoin du tableau de touches lorsque vous appuyez ailleurs que sur votre EditText? Le code suivant donne une fonction ordonnée qui doit être appelée dans votre méthode onCreate () une seule fois.

```
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);
            }
        }
}
```

Icône ou bouton dans Custom Edit Text et son action et cliquez sur auditers.

Cet exemple aidera à avoir le texte d'édition avec l'icône sur le côté droit.

> Remarque: Dans ce cas, j'utilise setCompoundDrawablesWithIntrinsicBounds, donc si vous souhaitez modifier la position de l'icône, vous pouvez obtenir la même chose avec setCompoundDrawablesWithIntrinsicBounds dans setIcon.

```
public class MKEditText extends AppCompatEditText {

    public interface IconClickListener {
        public void onClick();
    }

    private IconClickListener mIconClickListener;

    private static final String TAG = MKEditText.class.getSimpleName();

    private final int EXTRA_TOUCH_AREA = 50;
    private Drawable mDrawable;
    private boolean touchDown;

    public MKEditText(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
    }

    public MKEditText(Context context) {
        super(context);
    }

    public MKEditText(Context context, AttributeSet attrs) {
        super(context, attrs);
    }

    public void showRightIcon() {
        mDrawable = ContextCompat.getDrawable(getContext(), R.drawable.ic_android_black_24dp);

        setIcon();
    }

    public void setIconClickListener(IconClickListener iconClickListener) {
        mIconClickListener = iconClickListener;
    }

    private void setIcon() {
        Drawable[] drawables = getCompoundDrawables();

        setCompoundDrawablesWithIntrinsicBounds(drawables[0], drawables[1], mDrawable,
drawables[3]);

        setInputType(InputType.TYPE_CLASS_TEXT | InputType.TYPE_TEXT_VARIATION_PASSWORD);
        setSelection(getText().length());
    }

    @Override
    public boolean onTouchEvent(MotionEvent event) {
        final int right = getRight();
        final int drawableSize = getCompoundPaddingRight();
        final int x = (int) event.getX();
        switch (event.getAction()) {
            case MotionEvent.ACTION_DOWN:
                if (x + EXTRA_TOUCH_AREA >= right - drawableSize && x <= right +
```

```
EXTRA_TOUCH_AREA) {
                    touchDown = true;
                    return true;
                }
                break;
            case MotionEvent.ACTION_UP:
                if (x + EXTRA_TOUCH_AREA >= right - drawableSize && x <= right +
EXTRA_TOUCH_AREA && touchDown) {
                    touchDown = false;
                    if (mIconClickListener != null) {
                        mIconClickListener.onClick();
                    }
                    return true;
                }
                touchDown = false;
                break;

        }
        return super.onTouchEvent(event);
    }
}
```

Si vous souhaitez modifier la zone tactile, vous pouvez modifier les valeurs EXTRA_TOUCH_AREA par défaut que j'ai données comme 50.

Et pour activer le bouton et cliquer sur auditeur, vous pouvez appeler à partir de votre activité ou fragment comme ceci,

```
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.

            }
        });
```

Lire Éditer le texte en ligne: https://riptutorial.com/fr/android/topic/5843/editer-le-texte

Chapitre 104: Emplacement

Introduction

Les API d'emplacement Android sont utilisées dans une grande variété d'applications à des fins différentes, telles que la localisation de l'utilisateur, la notification d'un utilisateur ayant quitté un espace général (Geofencing) et l'interprétation des activités des utilisateurs (marche, course, conduite, etc.).

Cependant, les API d'emplacement Android ne sont pas le seul moyen d'acquérir l'emplacement de l'utilisateur. Les exemples suivants expliquent comment utiliser `LocationManager` d'Android et d'autres bibliothèques d'emplacement courantes.

Remarques

Pour créer des applications prenant en compte l'emplacement dans Android, il existe deux chemins:

- Open Source natif d'Android `LocationManager`
- Google `FusedLocationProviderApi` , qui fait partie des services Google Play

LocationManager

Avantages

- Contrôle plus granulaire
- Disponible dans tous les appareils
- Partie du framework Android

Les inconvénients

- La perte de batterie est un problème, si elle n'est pas gérée correctement
- Nécessite une logique pour changer de fournisseur de localisation, si le périphérique ne parvient pas à trouver un emplacement (par exemple, mauvais GPS dans un bâtiment)

Caractéristiques

- Auditeur NMEA
- Écouteur d'état GPS
- Écoutez les changements de statut du fournisseur (par exemple, le GPS est désactivé par l'utilisateur)
- Liste de fournisseurs pour choisir la source de localisation de

Fournisseurs

GPS

- Autorisations requises:
  - `ACCESS_FINE_LOCATION`
- Précision: 10m - 100m
- Exigences de puissance: ÉLEVÉ
- Disponibilité: Dans le monde entier (avec une vue dégagée du ciel)
- **NOTES** :
  - Les mises à jour de localisation interviennent généralement une fois par seconde, mais dans les situations où le GPS n'a pas été utilisé depuis un certain temps et où A-GPS n'est pas disponible, il faut plusieurs minutes pour recevoir un emplacement.
  - Dans les cas où une vue dégagée du ciel est obstruée, les points GPS ne se regrouperont pas très bien (les points d'emplacement "sautent") et la précision peut induire en erreur dans certaines zones en raison de l'effet " Urban Canyon ".

Réseau

- Autorisations requises:
  - `ACCESS_COARSE_LOCATION` ou `ACCESS_FINE_LOCATION`
- Précision: 100m - 1000m +
- Exigences de puissance: BAS - MOYEN
- Disponibilité: À portée de la tour cellulaire ou du signal wifi
- **REMARQUES:**
  - Les mises à jour de localisation se produisent moins fréquemment que le GPS
  - Les mises à jour d'emplacement ne sont généralement pas bien groupées (les points d'emplacement "sautent") et la précision peut varier en fonction du nombre de facteurs différents (nombre de signaux wifi, puissance du signal, type de tour cellulaire, etc.)

Passif

- Autorisations requises:
  - `ACCESS_FINE_LOCATION`
- Précision: 10m - 1000m +
- Exigences d'alimentation: AUCUN
- Disponibilité: Uniquement lorsqu'une autre application reçoit un emplacement de GPS ou de réseau
- **REMARQUES:**
  - Ne comptez pas sur cela pour vous donner des mises à jour continues. Cela permet d'écouter passivement les autres applications qui effectuent des requêtes de localisation et de revenir à ces emplacements.
  - Ne renvoie pas les points générés par FusedLocationProviderApi, mais uniquement les points d'emplacement sous-jacents utilisés pour les générer.

FusedLocationProviderApi

Avantages

- Offre moins de vidange de la batterie "out of the box"
- Gère mal le GPS
- Obtient des mises à jour plus régulièrement

Les inconvénients

- Moins de contrôle granulaire sur le GPS
- Peut ne pas être disponible sur tous les appareils ou dans certains pays
- Nécessite une dépendance de bibliothèque tierce

Caractéristiques

- Utilisation bien gérée des fournisseurs de localisation pour des économies de pâte optimales
- Génère généralement des points plus précis que le fournisseur d'emplacement réseau
- Mises à jour plus fréquentes de la bibliothèque, permettant une plus grande amélioration
- Pas besoin de spécifier le type de fournisseur à utiliser

Niveaux de priorité d'emplacementRequest

PRIORITY_HIGH_ACCURACY

- Autorisations requises:
  - `ACCESS_FINE_LOCATION` pour un emplacement plus précis ou `ACCESS_COARSE_LOCATION` pour un emplacement moins précis
- Précision: 10m - 100m
- Exigences de puissance: ÉLEVÉ
- Disponibilité: partout où les services Google Play sont disponibles.
- **REMARQUES:**
  - Si `ACCESS_FINE_LOCATION` n'est pas utilisé, cela n'utilisera pas le GPS pour générer des mises à jour de localisation, mais trouvera toujours un point assez précis dans les bonnes conditions.
  - Si `ACCESS_FINE_LOCATION` est utilisé, il peut utiliser ou non le GPS pour générer des points de localisation, en fonction de la précision avec laquelle il peut actuellement suivre le périphérique en fonction des conditions environnementales.
  - Bien que cela puisse rapporter des mises à jour de localisation plus précises que les autres réglages, il reste sujet à l'effet « Urban Canyon ».

PRIORITY_BALANCED_POWER_ACCURACY

- Autorisations requises:
  - `ACCESS_FINE_LOCATION` pour un emplacement plus précis ou `ACCESS_COARSE_LOCATION` pour un emplacement moins précis
- Précision: 100m - 1000m +
- Exigences d'alimentation: MOYEN
- Disponibilité: partout où les services Google Play sont disponibles.
- **REMARQUES:**
  - Mêmes notes que `PRIORITY_HIGH_ACCURACY`
  - Bien que cela soit peu probable, ce paramètre peut toujours utiliser le GPS pour

générer un emplacement.

PRIORITY_LOW_POWER

- Autorisations requises:
  - `ACCESS_FINE_LOCATION` ou `ACCESS_COARSE_LOCATION`
- Précision: 100m - 1000m +
- Exigences de puissance: BAS
- Disponibilité: partout où les services Google Play sont disponibles.
- **REMARQUES:**
  - N'utilise probablement pas le GPS, mais pas encore testé.
  - Les mises à jour ne sont généralement pas très précises
  - Utilisé généralement pour détecter les changements importants de l'emplacement

PRIORITY_NO_POWER

- Autorisations requises:
  - `ACCESS_FINE_LOCATION` ou `ACCESS_COARSE_LOCATION`
- Précision: 10m - 1000m +
- Exigences d'alimentation: AUCUN
- Disponibilité: partout où les services Google Play sont disponibles.
- **REMARQUES:**
  - Fonctionne presque identique à `LocationManager PASSIVE_PROVIDER`
  - `PASSIVE_PROVIDER` mises à jour des services Google Play lorsqu'elles sont reçues, où `PASSIVE_PROVIDER` signale les mises à jour d'emplacement sous-jacentes utilisées

Dépannage

OnLocationChanged () jamais appelé

Étant donné que cela semble être un problème courant lors de l'obtention d'emplacement Android, je vais mettre en place une liste de vérification rapide des correctifs courants:

---

1. Vérifiez votre manifeste!

   L'un des problèmes les plus courants est que les autorisations appropriées n'ont jamais été données. Si vous utilisez le GPS (avec ou sans réseau), utilisez `<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>` , sinon utilisez `<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>` . FusedLocationApi de Google requiert `ACCESS_FINE_LOCATION` .

---

2. (Pour Android 6+) Vérifiez les autorisations d'exécution !

   Recherchez et demandez des autorisations! Si vous ne recevez jamais d'autorisations, vous vous retrouverez avec des accidents, ou pire (si vous attrapez toutes les exceptions), vous vous retrouverez sans aucune indication! Peu importe que l'utilisateur vous accorde

l'autorisation au début de l'application, vérifiez toujours si vous disposez des autorisations pour tous les appels. L'utilisateur peut facilement accéder à leurs paramètres et les révoquer.

---

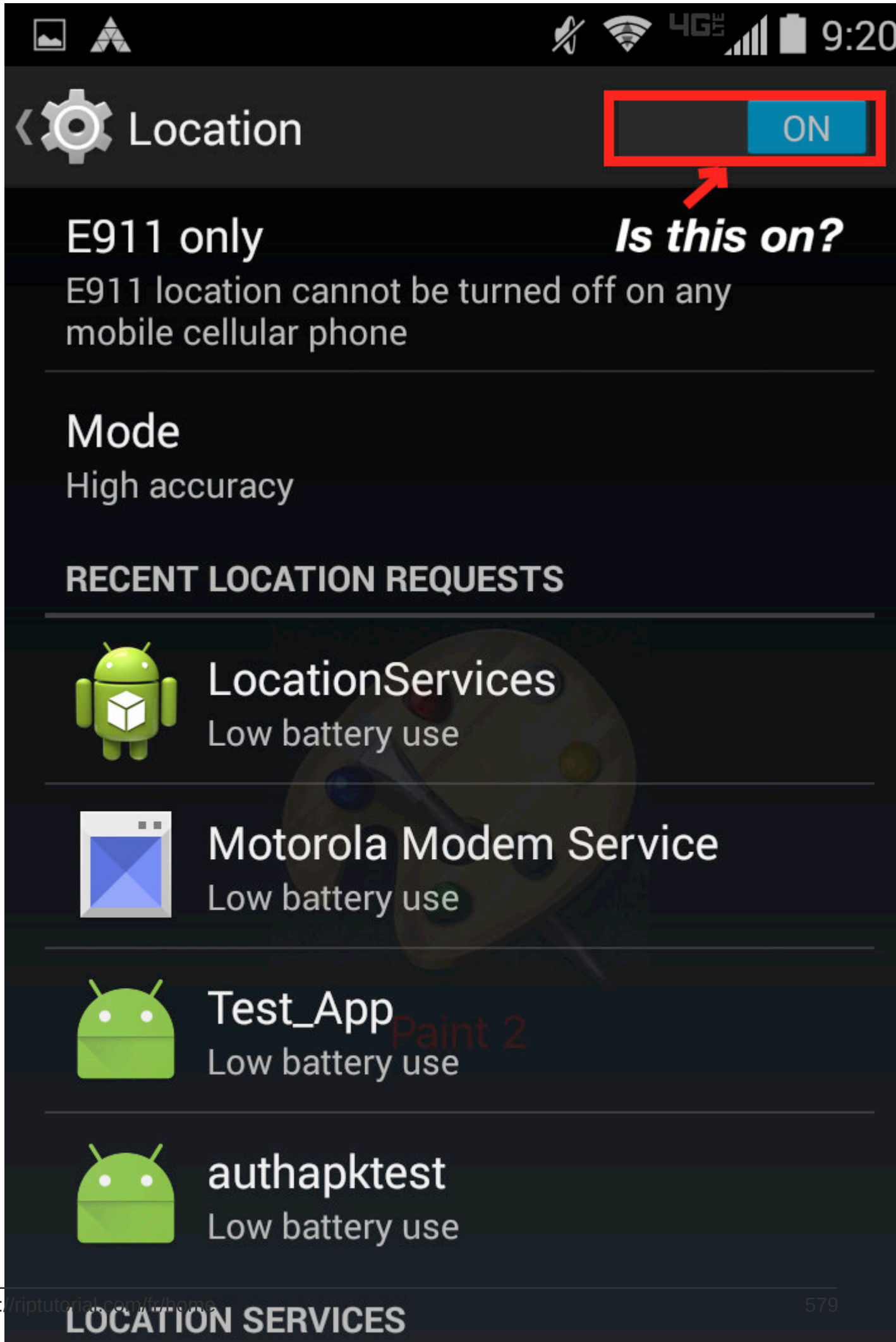
3. Vérifiez votre code!

   Êtes-vous sûr de passer le bon auditeur? Avez-vous ajouté ce `BroadcastReceiver` ou `IntentService` à votre manifeste? `PendingIntent.getService()` -vous `PendingIntent.getService()` dans une classe `BroadcastReceiver` ou `getBroadcast()` dans une classe `IntentService` ? Êtes-vous sûr de ne pas désinscrire votre auditeur ailleurs dans votre code immédiatement après avoir demandé?

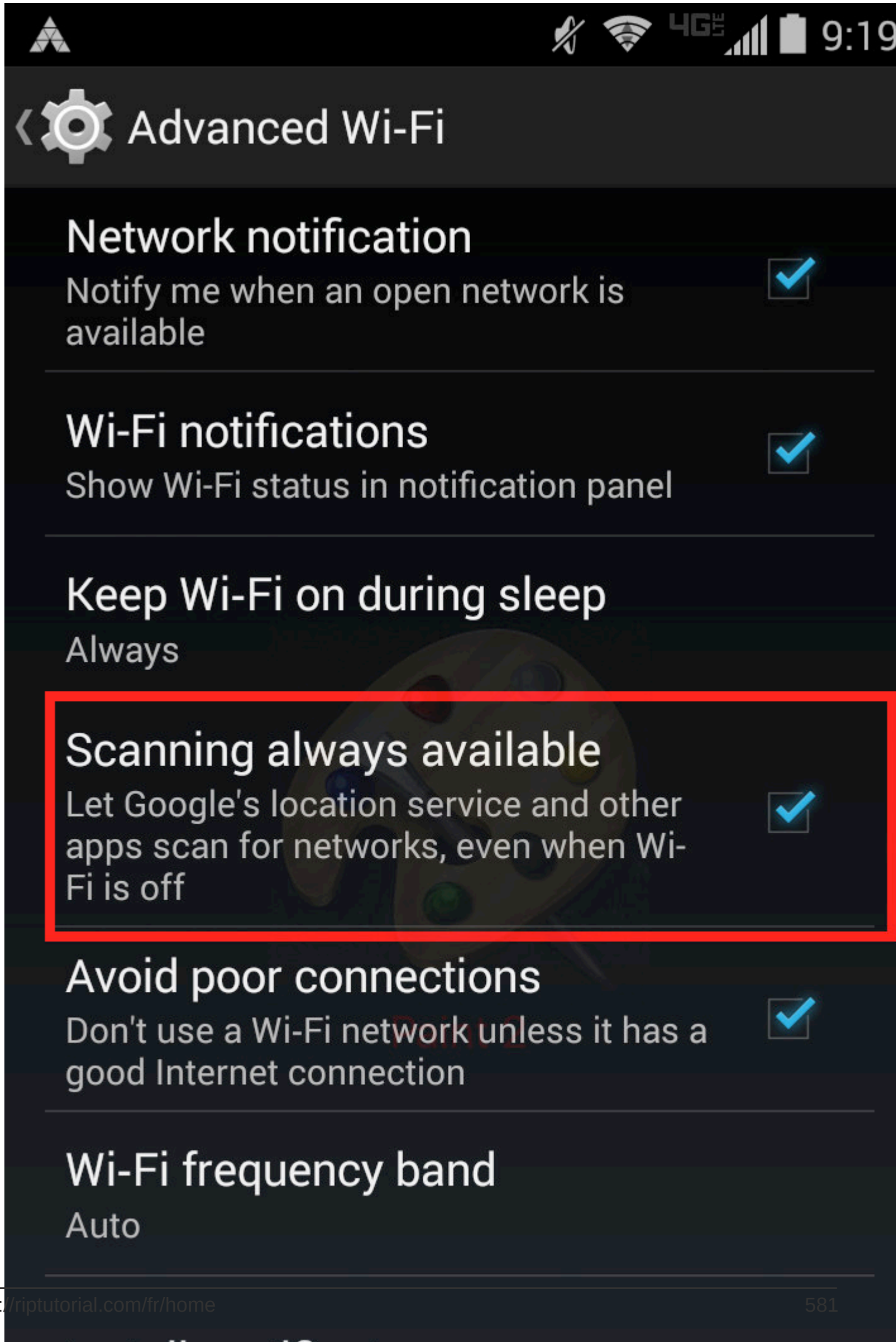
---

4. Vérifiez les paramètres de l'appareil!

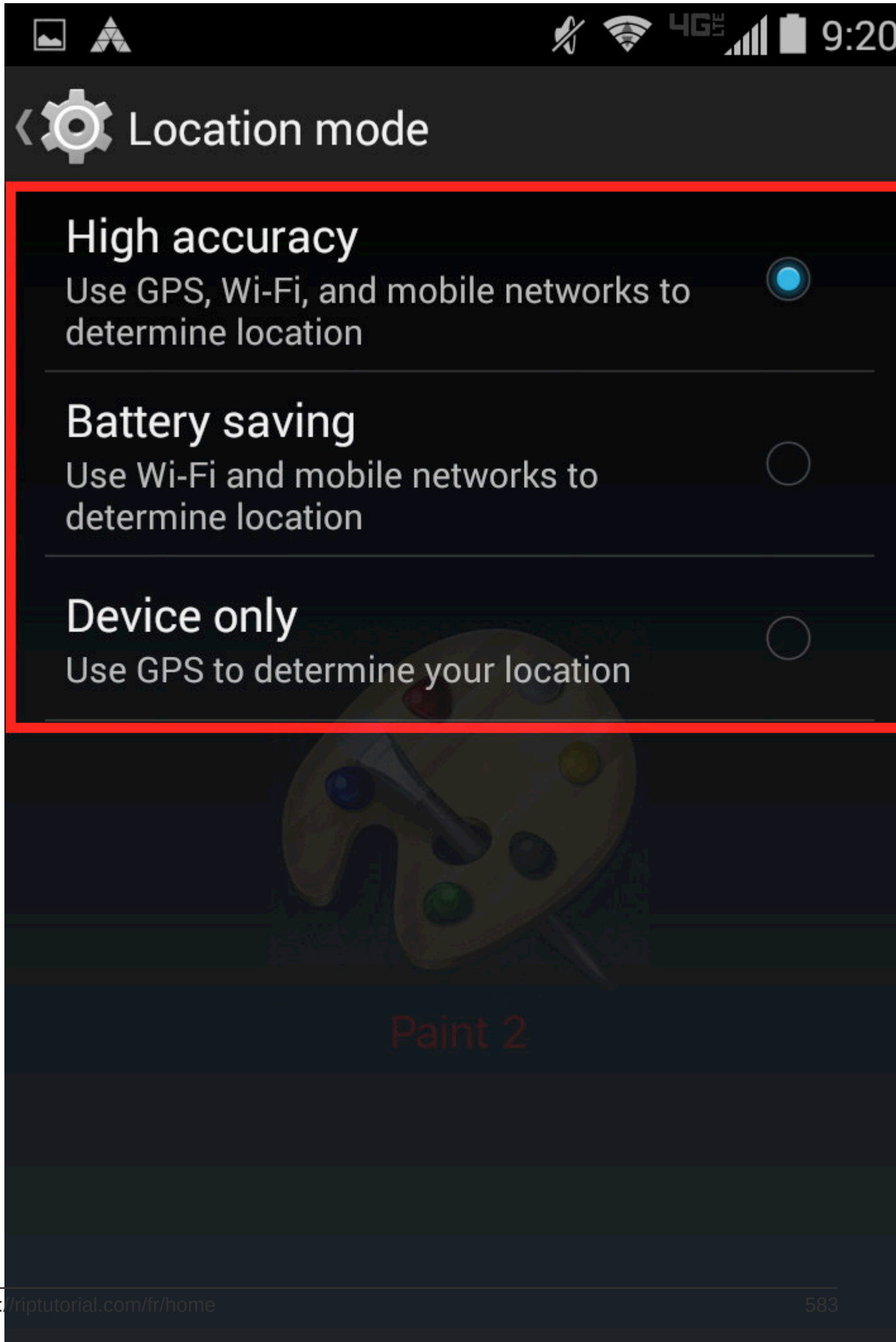
   De toute évidence, assurez-vous que les services de localisation sont activés.


9:20
Location
ON
Is this on?
E911 only
E911 location cannot be turned off on any mobile cellular phone
Mode
High accuracy
RECENT LOCATION REQUESTS
LocationServices
Low battery use
Motorola Modem Service
Low battery use
Test_App
Low battery use
authapktest
Low battery use
LOCATION SERVICES

Si vous utilisez les services réseau, avez-vous activé "Numérisation toujours disponible"? Votre mode de localisation est-il réglé sur "Meilleur" ("Haute précision") ou "Economie de batterie" ("Réseau uniquement")?


9:19
Advanced Wi-Fi
Network notification
Notify me when an open network is available
Wi-Fi notifications
Show Wi-Fi status in notification panel
Keep Wi-Fi on during sleep
Always
Scanning always available
Let Google's location service and other apps scan for networks, even when Wi-Fi is off
Avoid poor connections
Don't use a Wi-Fi network unless it has a good Internet connection
Wi-Fi frequency band
Auto

Si vous utilisez le GPS, avez-vous activé "Best" ("Haute précision") ou "Device only" en mode de localisation?


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

Oui, c'est ici deux fois. Avez-vous essayé d'utiliser un `LocationListener` au lieu d'un `PendingIntent` , ou vice-versa, pour vous assurer que vous avez implémenté `LocationManager` correctement? Êtes-vous sûr que la demande de localisation n'est pas supprimée dans une partie du cycle de vie de l'activité ou du service que vous ne vous attendiez pas?

`PendingIntent` , ou vice-versa, pour vous assurer que vous avez implémenté `LocationManager` correctement? Êtes-vous sûr que la demande de localisation n'est pas supprimée dans une partie du cycle de vie de l'activité ou du service que vous ne vous attendiez pas?

---

6. Vérifiez votre environnement!

   Êtes-vous en train de tester le GPS au premier étage d'un immeuble au milieu de San Francisco? Êtes-vous en train de tester des emplacements de réseau au milieu de nulle part? Travaillez-vous dans un bunker secret souterrain sans tous les signaux radio, en vous demandant pourquoi votre appareil ne trouve pas sa place? Vérifiez toujours votre environnement lorsque vous tentez de résoudre les problèmes de localisation!

Il pourrait y avoir beaucoup d'autres raisons moins évidentes pour lesquelles l'emplacement ne fonctionne pas, mais avant de rechercher ces corrections ésotériques, parcourez simplement cette liste de contrôle rapide.

Examples

API de localisation fusionnée

Exemple d'utilisation de l'activité w / LocationRequest

```
/*
 * This example is useful if you only want to receive updates in this
 * activity only, and have no use for location anywhere else.
 */
public class LocationActivity extends AppCompatActivity implements
        GoogleApiClient.ConnectionCallbacks, GoogleApiClient.OnConnectionFailedListener,
LocationListener {

    private GoogleApiClient mGoogleApiClient;
    private LocationRequest mLocationRequest;

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

        mGoogleApiClient = new GoogleApiClient.Builder(this)
                .addConnectionCallbacks(this)
                .addOnConnectionFailedListener(this)
```

```
                .addApi(LocationServices.API)
                .build();

        mLocationRequest = new LocationRequest()
                .setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY) //GPS quality location
points
                .setInterval(2000) //At least once every 2 seconds
                .setFastestInterval(1000); //At most once a second
    }

    @Override
    protected void onStart(){
        super.onStart();
        mGoogleApiClient.connect();
    }

    @Override
    protected void onResume(){
        super.onResume();
        //Permission check for Android 6.0+
        if(ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION)
== PackageManager.PERMISSION_GRANTED) {
            if(mGoogleApiClient.isConnected()) {
                LocationServices.FusedLocationApi.requestLocationUpdates(mGoogleApiClient,
mLocationRequest, this);
            }
        }
    }

    @Override
    protected void onPause(){
        super.onPause();
        //Permission check for Android 6.0+
        if(ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION)
== PackageManager.PERMISSION_GRANTED) {
            if(mGoogleApiClient.isConnected()) {
                LocationServices.FusedLocationApi.removeLocationUpdates(mGoogleApiClient,
this);
            }
        }
    }

    @Override
    protected void onStop(){
        super.onStop();
        mGoogleApiClient.disconnect();
    }

    @Override
    public void onConnected(@Nullable Bundle bundle) {
        if(ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION)
== PackageManager.PERMISSION_GRANTED) {
            LocationServices.FusedLocationApi.requestLocationUpdates(mGoogleApiClient,
mLocationRequest, this);
        }
    }

    @Override
    public void onConnectionSuspended(int i) {
        mGoogleApiClient.connect();
    }
```

```
    @Override
    public void onConnectionFailed(@NonNull ConnectionResult connectionResult) {
    }

    @Override
    public void onLocationChanged(Location location) {
        //Handle your location update code here
    }
}
```

Exemple d'utilisation du service w / PendingIntent et BroadcastReceiver

ExempleActivité

Lecture recommandée: 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
            }
        }
    }
}
```

Service de localisation

NOTE: N'oubliez pas d'inscrire ce service dans le manifeste!

```
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;
    }
}
```

EmplacementReceiver

REMARQUE: N'oubliez pas d'enregistrer ce récepteur dans le manifeste!

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

Demande de mises à jour d'emplacement à l'aide de LocationManager

Comme toujours, vous devez vous assurer que vous disposez des autorisations requises.

```
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) {

    }
}
```

Demande de mises à jour d'emplacement sur un thread distinct à l'aide de LocationManager

Comme toujours, vous devez vous assurer que vous disposez des autorisations requises.

```
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) {

    }
}
```

Enregistrez geofence

J'ai créé la classe **singleton** `GeoFenceObserversationService` .

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);
        }
    }
```

Où j'ai commencé le service? De la classe d'application

- `startService(new Intent(getApplicationContext(),GeoFenceObserversationService.class));`

Comment j'ai enregistré Geofences?

- `GeoFenceObserversationService.getInstant().addGeofences();`

Obtenir l'adresse de l'emplacement à l'aide de Geocoder

Après avoir obtenu l'objet `Location` de `FusedAPI` , vous pouvez facilement acquérir des informations d' `Address` partir de cet objet.

```
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;
}
```

Obtenir des mises à jour de localisation dans un BroadcastReceiver

Commencez par créer une classe BroadcastReceiver pour gérer les mises à jour d’emplacement entrantes:

```
public class LocationReceiver extends BroadcastReceiver implements Constants {

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

        if (LocationResult.hasResult(intent)) {
            LocationResult locationResult = LocationResult.extractResult(intent);
            Location location = locationResult.getLastLocation();
            if (location != null) {
                // Do something with your location
            } else {
                Log.d(LocationReceiver.class.getSimpleName(), "*** location object is null
***");
            }
```

```
        }
    }
}
```

Ensuite, lorsque vous vous connectez à GoogleApiClient dans le rappel onConnected:

```
@Override
public void onConnected(Bundle connectionHint) {
    Intent backgroundIntent = new Intent(this, LocationReceiver.class);
    mBackgroundPendingIntent = backgroundPendingIntent.getBroadcast(getApplicationContext(),
LOCATION_REUEST_CODE, backgroundIntent, PendingIntent.FLAG_CANCEL_CURRENT);
    mFusedLocationProviderApi.requestLocationUpdates(mLocationClient, mLocationRequest,
backgroundPendingIntent);
}
```

N'oubliez pas de supprimer l'intention de mise à jour d'emplacement dans le rappel de cycle de vie approprié:

```
@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();
}
```

Lire Emplacement en ligne: https://riptutorial.com/fr/android/topic/1837/emplacement

Chapitre 105: Émulateur

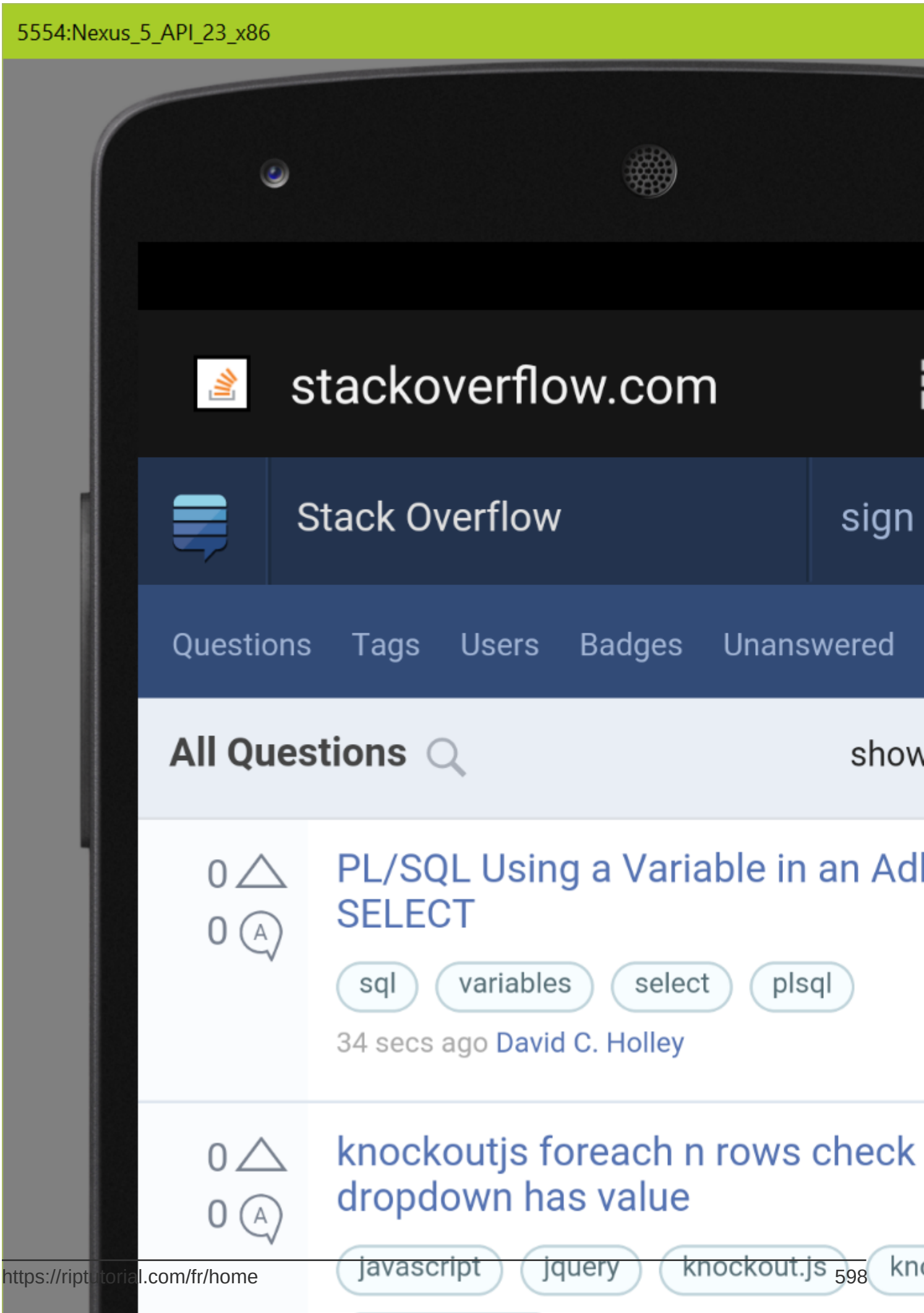
Remarques

AVD signifie *Android Virtual Device*

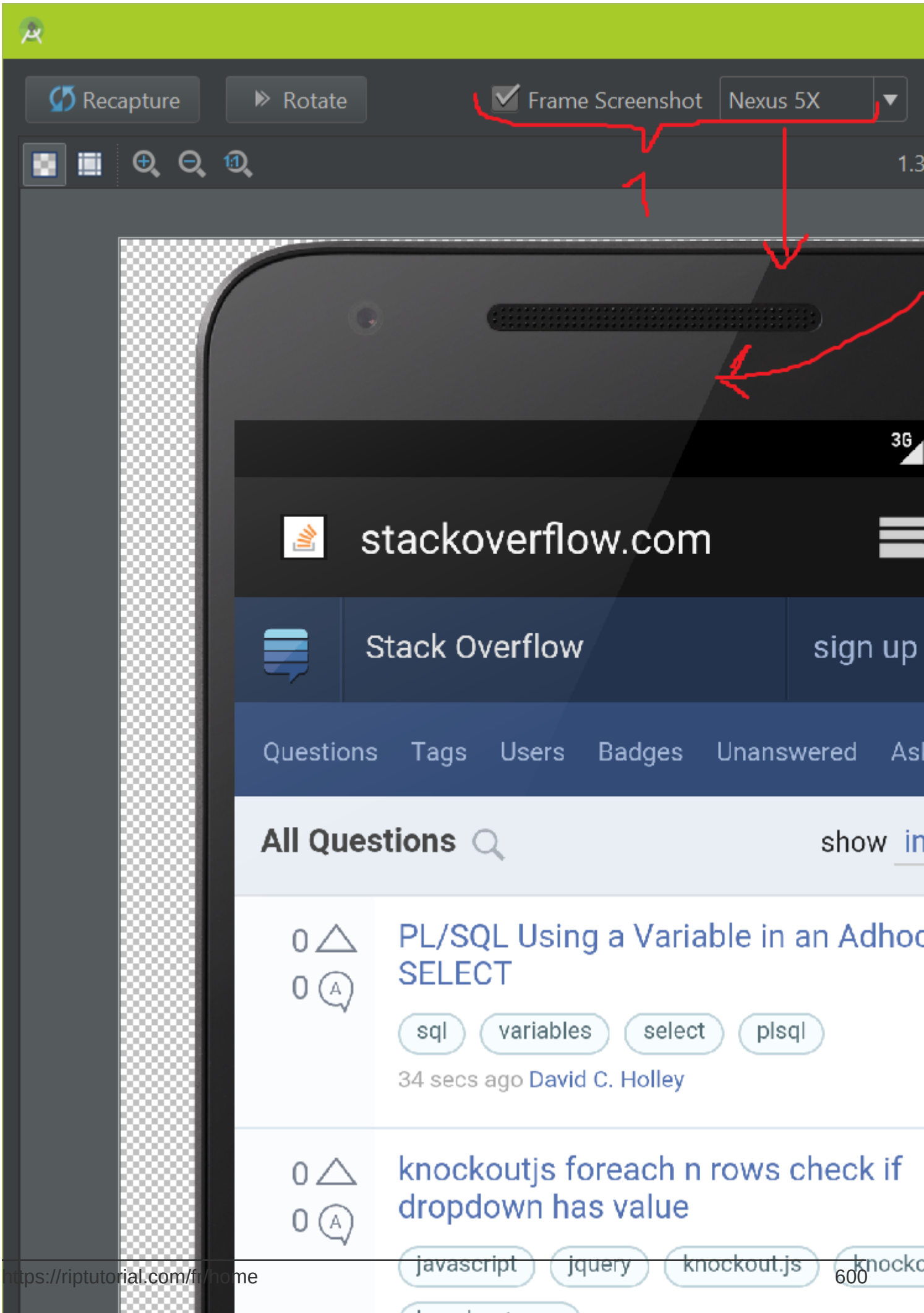
Examples

Prendre des captures d'écran

Si vous voulez faire une capture d'écran de l'émulateur Android (2.0), il vous suffit d'appuyer sur `Ctrl` + `S` ou de cliquer sur l'icône de la caméra sur la barre latérale:


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

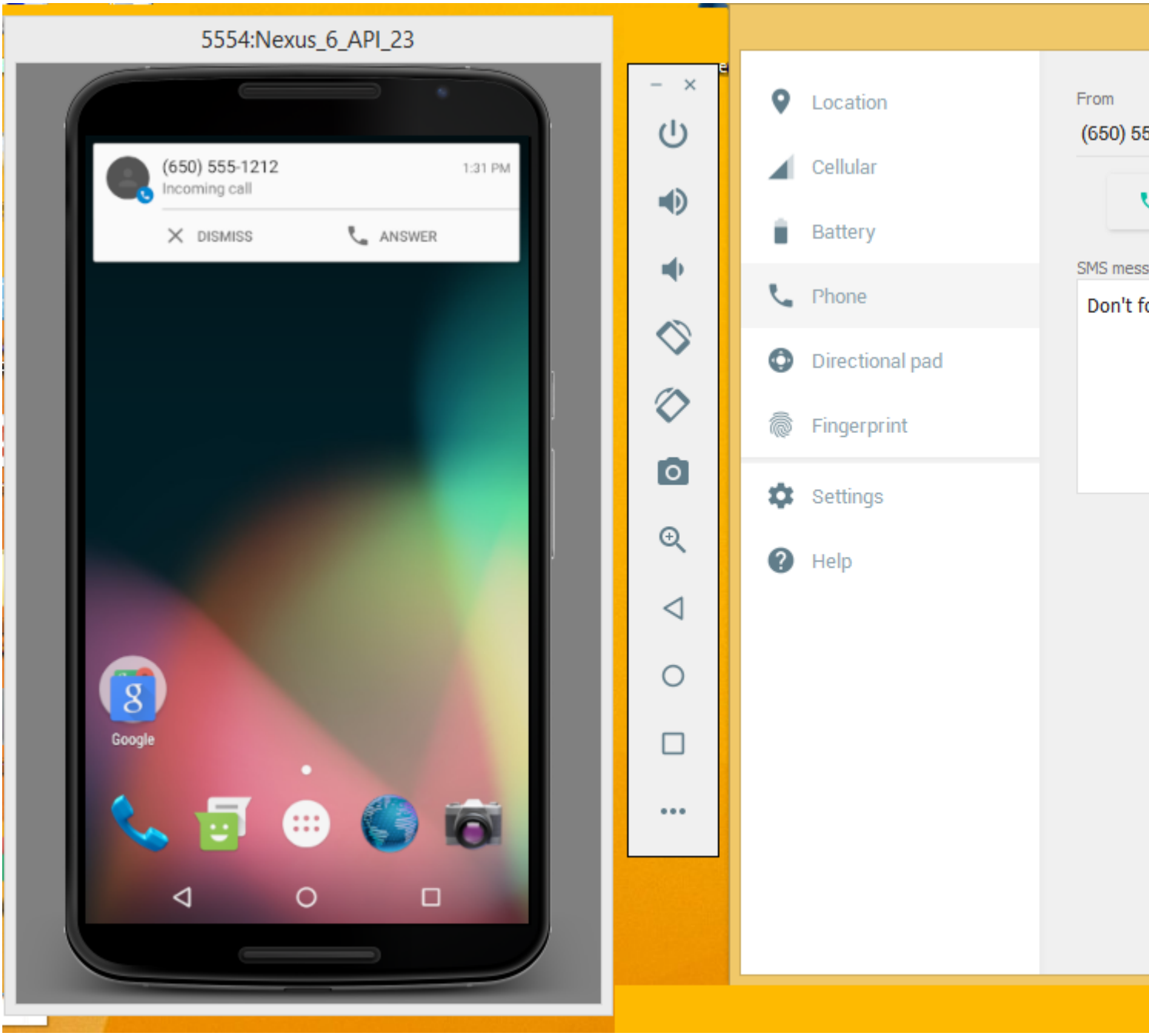
2. Une ombre portée sous le cadre de l'appareil.
3. Un reflet d'écran sur le cadre et la capture d'écran de l'appareil.


Recapture
Rotate
Frame Screenshot
Nexus 5X
3G
stackoverflow.com
Stack Overflow
sign up
Questions
Tags
Users
Badges
Unanswered
All Questions
show
0
0
PL/SQL Using a Variable in an Adhoc SELECT
sql
variables
select
plsql
34 secs ago David C. Holley
0
0
knockoutjs foreach n rows check if dropdown has value
javascript
jquery
knockout.js

`Android` > `AVD Manager` ou en cliquant sur l'icône du Gestionnaire AVD dans la barre d'outils qui est la deuxième de la capture d'écran ci-dessous.

Simuler un appel

Pour simuler un appel téléphonique, appuyez sur le bouton «Contrôles étendus» indiqué par trois points, choisissez «Téléphone» et sélectionnez «Appeler». Vous pouvez également éventuellement modifier le numéro de téléphone.



Résolution des erreurs lors du démarrage de l'émulateur

Tout d'abord, assurez-vous d'avoir activé la « **virtualisation»** dans la configuration de votre BIOS.

Démarrez le **SDK Manager Android** , sélectionnez **Extras** , puis sélectionnez **Intel Hardware Accelerated Execution Manager** et attendez la fin du téléchargement. Si cela ne fonctionne toujours pas, ouvrez votre dossier SDK et exécutez

`/extras/intel/Hardware_Accelerated_Execution_Manager/IntelHAXM.exe` .

Suivez les instructions à l'écran pour terminer l'installation.

Ou pour OS X, vous pouvez le faire sans les invites à l'écran comme ceci:
`/extras/intel/Hardware_Accelerated_Execution_Manager/HAXM\ installation`

> Si votre CPU ne prend pas en charge VT-x ou SVM, vous ne pouvez pas utiliser d'images Android basées sur x86. Veuillez utiliser des images basées sur ARM à la place.

Une fois l'installation terminée, vérifiez que le pilote de virtualisation fonctionne correctement en ouvrant une fenêtre d'invite de commandes et en exécutant la commande suivante: `sc query intelhaxm`

Pour exécuter un émulateur basé sur x86 avec une accélération de machine virtuelle: Si vous exécutez l'émulateur à partir de la ligne de commande, spécifiez simplement un AVD x86:
`emulator -avd <avd_name>`

Si vous suivez correctement toutes les étapes mentionnées ci-dessus, vous devriez pouvoir voir votre AVD avec HAXM arriver normalement.

Lire Émulateur en ligne: https://riptutorial.com/fr/android/topic/122/emulateur

Chapitre 106: Événements / Intentions du bouton matériel (PTT, LWP, etc.)

Introduction

Plusieurs appareils Android ont des boutons personnalisés ajoutés par le fabricant. Cela ouvre de nouvelles possibilités au développeur en matière de gestion de ces boutons, en particulier lors de la création d'applications ciblées pour les périphériques matériels.

Cette rubrique documente les boutons auxquels sont associées des intentions que vous pouvez écouter via des récepteurs d'intention.

Examples

Dispositifs Sonim

Les périphériques Sonim varient selon les modèles de nombreux boutons personnalisés:

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
```

Enregistrement des boutons

Pour recevoir ces informations, vous devrez attribuer les boutons à votre application dans les paramètres du téléphone. Sonim peut enregistrer automatiquement les boutons sur l'application lors de son installation. Pour ce faire, vous devrez les contacter et obtenir une clé spécifique au package à inclure dans votre manifeste comme ceci:

```
<meta-data
    android:name="app_key_green_data"
    android:value="your-key-here" />
```

RugGear Devices

Bouton PTT

```
android.intent.action.PTT.down
android.intent.action.PTT.up
```

Confirmé le: RG730, RG740A

Lire Événements / Intentions du bouton matériel (PTT, LWP, etc.) en ligne: https://riptutorial.com/fr/android/topic/10418/evenements---intentions-du-bouton-materiel--ptt--lwp--etc--

Chapitre 107: Exécution instantanée dans Android Studio

Remarques

Instant Run est un comportement étendu pour les commandes d'exécution et de débogage qui permet un débogage plus rapide en ne nécessitant pas une génération complète et une réinstallation pour toute modification effectuée dans le code de votre application.

> Introduit dans Android Studio 2.0, Instant Run est un comportement des commandes Run et Debug qui réduit considérablement le temps entre les mises à jour de votre application. Bien que votre première version puisse prendre plus de temps, Instant Run pousse les mises à jour ultérieures vers votre application sans créer de nouvel APK. Les modifications sont donc visibles beaucoup plus rapidement.
>
> Instant Run est pris en charge uniquement lorsque vous déployez la variante de génération de débogage, utilisez le plug-in Android pour Gradle version 2.0.0 ou ultérieure et définissez minSdkVersion sur 15 ou plus dans le fichier build.gradle de niveau application de votre application. Pour des performances optimales, définissez minSdkVersion sur 21 ou plus.
>
> Après avoir déployé une application, une petite icône représentant un coup de foudre jaune apparaît dans le bouton Exécuter (ou bouton Déboguer), indiquant qu'Instant Run est prêt à envoyer des mises à jour la prochaine fois que vous cliquerez sur le bouton. Au lieu de créer un nouvel APK, il ne fait que pousser ces nouveaux changements et, dans certains cas, l'application n'a même pas besoin de redémarrer, mais montre immédiatement l'effet de ces modifications de code.
>
> Instant Run pousse le code et les ressources mis à jour vers votre périphérique connecté ou émulateur en effectuant un échange à chaud, un échange à chaud ou un échange à froid. Il détermine automatiquement le type de swap à effectuer en fonction du type de modification effectué. La vidéo ci-dessus fournit des détails intéressants sur la façon dont tout cela fonctionne sous le capot. Pour un résumé rapide de la manière dont Instant Run se comporte lorsque vous transmettez certaines modifications de code à un périphérique cible, consultez le tableau suivant.

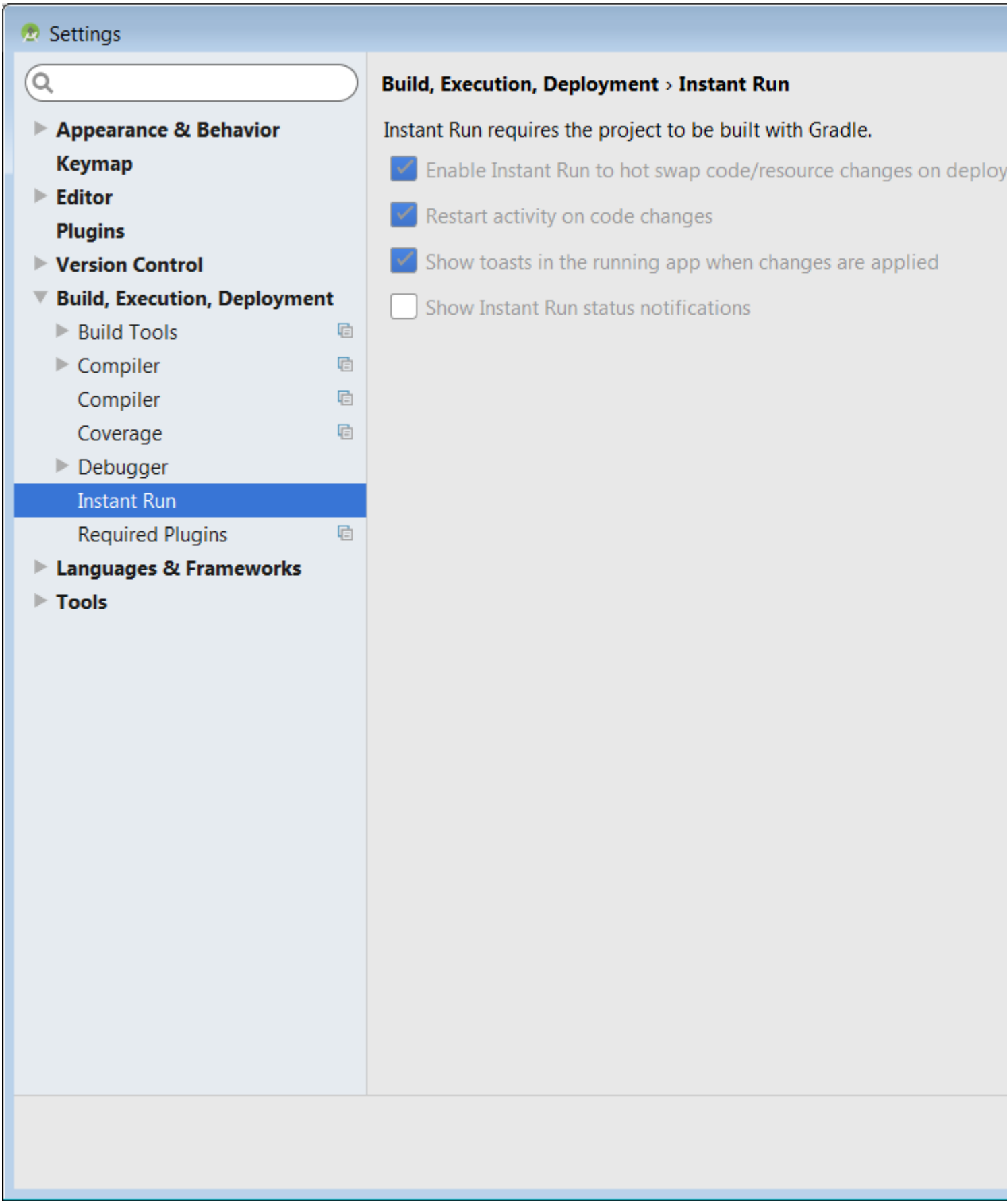
Documentation

Examples

Activation ou désactivation d'Instant Run

1. Ouvrez la boîte de dialogue Configuration ou Préférences:
    - Sous Windows ou Linux, sélectionnez `File` > `Settings` dans le menu principal.

    - Sur Mac OSX, sélectionnez `Android Studio` > `Preferences` dans le menu principal.
2. Naviguez vers la `Build, Execution, Deployment` > le `Compiler` .
3. Dans le champ de texte à côté des options de ligne de commande, entrez vos options de ligne de commande.
4. Cliquez sur OK pour enregistrer et quitter.



La première option est l'exécution instantanée. Cochez / décochez cette case.

Documentation

Types de swaps de code dans Instant Run

Il existe trois types de permutation de code qu'Instant run permet de prendre en charge un débogage et une exécution plus rapides de l'application à partir de votre code dans Android Studio.

- Échange à chaud
- Échange à chaud
- Échange de froid

Quand chacun de ces swaps est-il déclenché?

HOT SWAP est déclenché lorsque l'implémentation d'une méthode existante est modifiée.

WARM SWAP est déclenché lorsqu'une ressource existante est modifiée ou supprimée (n'importe quoi dans le dossier res)

COLD SWAP à chaque changement de code structurel dans le code de votre application, par exemple

1. Ajouter, supprimer ou modifier:

- une annotation
- un champ d'instance
- un champ statique
- une signature de méthode statique
- une signature de méthode d'instance

2. Modifier la classe parent dont la classe actuelle hérite
3. Changer la liste des interfaces implémentées
4. Changer l'initialiseur statique d'une classe
5. Réorganiser les éléments de présentation qui utilisent des ID de ressource dynamique

Que se passe-t-il lorsqu'un échange de code se produit?

Les modifications **HOT SWAP** sont visibles instantanément - dès que le prochain appel à la méthode dont l'implémentation est modifiée est effectué.

WARM SWAP redémarre l'activité en cours

COLD SWAP redémarre l'application entière (sans réinstaller)

Modifications de code non prises en charge lors de l'utilisation d'Instant Run

Il y a quelques changements où l'instant ne fera pas l'affaire et une version complète et la réinstallation de votre application se produiront comme si cela se produisait avant la naissance d'Instant Run.

1. Modifier le manifeste de l'application

2. Modifier les ressources référencées par le manifeste de l'application
3. Modifier un élément d'interface utilisateur de widget Android (nécessite un nettoyage et une réexécution)

Documentation

Lire Exécution instantanée dans Android Studio en ligne:
https://riptutorial.com/fr/android/topic/2119/execution-instantanee-dans-android-studio

Chapitre 108: ExoPlayer

Examples

Ajouter ExoPlayer au projet

Via jCenter

y compris les éléments suivants dans le fichier build.gradle de votre projet:

```
compile 'com.google.android.exoplayer:exoplayer:rX.X.X'
```

où rX.XX est votre version préférée. Pour la dernière version, voir les versions du projet. Pour plus de détails, voir le projet sur Bintray .

Utiliser ExoPlayer

Instanciez votre ExoPlayer:

```
exoPlayer = ExoPlayer.Factory.newInstance(RENDERER_COUNT, minBufferMs, minRebufferMs);
```

Pour lire de l'audio seulement, vous pouvez utiliser ces valeurs:

```
RENDERER_COUNT = 1 //since you want to render simple audio
minBufferMs = 1000
minRebufferMs = 5000
```

Les deux valeurs de tampon peuvent être modifiées en fonction de vos besoins.

Maintenant, vous devez créer un DataSource. Lorsque vous voulez diffuser du mp3, vous pouvez utiliser DefaultUriDataSource. Vous devez passer le contexte et un agent utilisateur. Pour rester simple, jouez un fichier local et transmettez null comme userAgent:

```
DataSource dataSource = new DefaultUriDataSource(context, null);
```

Ensuite, créez le sampleSource:

```
ExtractorSampleSource sampleSource = new ExtractorSampleSource(
                uri, dataSource, new Mp3Extractor(), RENDERER_COUNT, requestedBufferSize);
```

uri pointe vers votre fichier, en tant qu'extracteur, vous pouvez utiliser un simple Mp3Extractor par défaut si vous voulez lire des mp3. requestedBufferSize peut être modifié à nouveau selon vos besoins. Utilisez 5000 par exemple.

Vous pouvez maintenant créer votre rendu de piste audio en utilisant la source d'échantillon comme suit:

```
MediaCodecAudioTrackRenderer audioRenderer = new MediaCodecAudioTrackRenderer(sampleSource);
```

Enfin appelez-vous sur votre instance exoPlayer:

```
exoPlayer.prepare(audioRenderer);
```

Pour lancer l'appel de lecture:

```
exoPlayer.setPlayWhenReady(true);
```

Principales étapes pour lire de la vidéo et de l'audio à l'aide des implémentations TrackRenderer standard

```
// 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!
```

Lire ExoPlayer en ligne: https://riptutorial.com/fr/android/topic/6248/exoplayer

Chapitre 109: Facebook SDK pour Android

Syntaxe

- **newInstance** : Pour créer une instance unique de la classe d'assistance Facebook.
- **loginUser** : Pour vous connecter à l'utilisateur.
- **signOut** : pour déconnecter l'utilisateur.
- **getCallbackManager** : Obtenir un rappel pour Facebook.
- **getLoginCallback** : Pour obtenir un rappel pour la connexion.
- **getKeyHash** : pour générer un hachage de clé sur Facebook.

Paramètres

Paramètre	Détails
MARQUE	Une chaîne utilisée lors de la connexion
FacebookSignInHelper	Une référence statique à l'assistant facebook
CallbackManager	Un rappel pour les opérations facebook
Activité	Un contexte
PERMISSION_LOGIN	Un tableau contenant toutes les autorisations requises de facebook pour se connecter.
loginCallback	Un rappel pour la connexion facebook

Examples

Comment ajouter Facebook Login dans Android

Ajouter les dépendances ci-dessous à votre `build.gradle`

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

Ajoutez la classe d'assistance ci-dessous à votre paquet utilitaire:

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

Ajoutez le code ci-dessous dans votre activité:

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

Ajoutez le code ci-dessous à votre `OnActivityResult` :

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

Définition des autorisations d'accès aux données du profil Facebook

Si vous souhaitez récupérer les détails du profil Facebook d'un utilisateur, vous devez définir les autorisations pour le même:

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

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

Vous pouvez continuer à ajouter des autorisations comme la liste d'amis, les publications, les photos, etc. Il vous suffit de choisir la bonne autorisation et d'ajouter la liste ci-dessus.

Remarque: vous n'avez pas besoin de définir d'autorisations explicites pour accéder au profil public (prénom, nom, identifiant, sexe, etc.).

Créez votre propre bouton personnalisé pour la connexion Facebook

Une fois que vous avez ajouté la connexion / inscription Facebook, le bouton ressemble à:

La plupart du temps, cela ne correspond pas aux spécifications de conception de votre application. Et voici comment vous pouvez le personnaliser:

```
<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>
```

Il vous suffit de `com.facebook.login.widget.LoginButton` le `com.facebook.login.widget.LoginButton` origine dans un `FrameLayout` et de `FrameLayout` sa visibilité.

Ensuite, ajoutez votre bouton personnalisé dans le même `FrameLayout` . J'ai ajouté quelques exemples de spécifications. Vous pouvez toujours créer votre propre arrière-plan à dessiner pour le bouton facebook et le définir comme arrière-plan du bouton.

La dernière chose que nous faisons est de simplement convertir le clic sur mon bouton

personnalisé en un clic sur le bouton du facecbook:

```
//The original Facebook button
LoginButton loginButton = (LoginButton)findViewById(R.id.login_button);

//Our custom Facebook button
fb = (Button) findViewById(R.id.fb);

public void onClickFacebookButton(View view) {
    if (view == fb) {
        loginButton.performClick();
    }
}
```

Génial! Maintenant, le bouton ressemble à ceci:

Un guide minimaliste pour la connexion / inscription à Facebook

1. Vous devez configurer les prérequis .

2. Ajoutez l'activité Facebook au fichier *AndroidManifest.xml* :

   ```
   <activity
       android:name="com.facebook.FacebookActivity"
       android:configChanges= "keyboard|keyboardHidden|screenLayout|screenSize|orientation"
       android:theme="@android:style/Theme.Translucent.NoTitleBar"
       android:label="@string/app_name" />
   ```

3. Ajoutez le bouton de connexion à votre fichier XML de mise en page:

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

4. Maintenant, vous avez le bouton Facebook. Si l'utilisateur clique dessus, la boîte de dialogue de connexion Facebook apparaîtra en haut de l'écran de l'application. Ici, l'utilisateur peut remplir ses informations d'identification et appuyer sur le bouton *Connexion* . Si les informations d'identification sont correctes, la boîte de dialogue accorde les autorisations correspondantes et un rappel est envoyé à votre activité d'origine contenant le bouton. Le code suivant montre comment vous pouvez recevoir ce rappel:

   ```
   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.
    }
});
```

Déconnexion de Facebook

Facebook SDK 4.0 et versions ultérieures, voici comment nous nous déconnectons:

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

Pour les versions antérieures à 4.0, la déconnexion est supprimée en effaçant explicitement le jeton d'accès:

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

Lire Facebook SDK pour Android en ligne: https://riptutorial.com/fr/android/topic/3919/facebook-sdk-pour-android

Chapitre 110: Facturation dans l'application

Examples

Achats in-app consommables

Les produits gérés par consommables sont des produits qui peuvent être achetés plusieurs fois, tels que la monnaie du jeu, les vies de jeu, les bonus, etc.

Dans cet exemple, nous allons implémenter 4 produits **gérés** consommables différents `"item1", "item2", "item3", "item4"`.

Étapes en résumé:

1. Ajoutez la bibliothèque de facturation intégrée à votre projet (fichier AIDL).
2. Ajoutez l'autorisation requise dans le fichier `AndroidManifest.xml`.
3. Déployez un fichier apk signé sur Google Developers Console.
4. Définissez vos produits.
5. Implémentez le code.
6. Test de la facturation intégrée (facultatif).

Étape 1:

Tout d'abord, nous devrons ajouter le fichier AIDL à votre projet, comme l'explique clairement la documentation Google ici .

`IInAppBillingService.aidl` est un fichier AIDL (Android Interface Definition Language) qui définit l'interface avec le service In-App Billing Version 3. Vous utiliserez cette interface pour effectuer des demandes de facturation en appelant des appels de méthode IPC.

Étape 2:

Après avoir ajouté le fichier AIDL, ajoutez l'autorisation `AndroidManifest.xml` dans `AndroidManifest.xml` :

```
<!-- Required permission for implementing In-app Billing -->
<uses-permission android:name="com.android.vending.BILLING" />
```

Étape 3:

Générez un fichier apk signé et téléchargez-le sur Google Developers Console. Ceci est

nécessaire pour que nous puissions commencer à définir nos produits intégrés.

Étape 4:

Définissez tous vos produits avec différents productID et fixez un prix à chacun d'entre eux. Il existe 2 types de produits (produits gérés et abonnements). Comme nous l'avons déjà dit, nous allons mettre en œuvre 4 différents produits **gérés** consommables `"item1", "item2", "item3", "item4"` .

Étape 5:

Après avoir effectué toutes les étapes ci-dessus, vous êtes maintenant prêt à commencer à implémenter le code lui-même dans votre propre activité.

Activité principale:

```
public class MainActivity extends Activity {

    IInAppBillingService inAppBillingService;
    ServiceConnection serviceConnection;

    // productID for each item. You should define them in the Google Developers Console.
    final String item1 = "item1";
    final String item2 = "item2";
    final String item3 = "item3";
    final String item4 = "item4";

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

        // Instantiate the views according to your layout file.
        final Button buy1 = (Button) findViewById(R.id.buy1);
        final Button buy2 = (Button) findViewById(R.id.buy2);
        final Button buy3 = (Button) findViewById(R.id.buy3);
        final Button buy4 = (Button) findViewById(R.id.buy4);

        // setOnClickListener() for each button.
        // buyItem() here is the method that we will implement to launch the PurchaseFlow.
        buy1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                buyItem(item1);
            }
        });

        buy2.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                buyItem(item2);
            }
        });
```

```
        buy3.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                buyItem(item3);
            }
        });

        buy4.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                buyItem(item4);
            }
        });

        // Attach the service connection.
        serviceConnection = new ServiceConnection() {
            @Override
            public void onServiceDisconnected(ComponentName name) {
                inAppBillingService = null;
            }

            @Override
            public void onServiceConnected(ComponentName name, IBinder service) {
                inAppBillingService = IInAppBillingService.Stub.asInterface(service);
            }
        };

        // Bind the service.
        Intent serviceIntent = new
Intent("com.android.vending.billing.InAppBillingService.BIND");
        serviceIntent.setPackage("com.android.vending");
        bindService(serviceIntent, serviceConnection, BIND_AUTO_CREATE);

        // Get the price of each product, and set the price as text to
        // each button so that the user knows the price of each item.
        if (inAppBillingService != null) {
            // Attention: You need to create a new thread here because
            // getSkuDetails() triggers a network request, which can
            // cause lag to your app if it was called from the main thread.
            Thread thread = new Thread(new Runnable() {
                @Override
                public void run() {
                    ArrayList<String> skuList = new ArrayList<>();
                    skuList.add(item1);
                    skuList.add(item2);
                    skuList.add(item3);
                    skuList.add(item4);
                    Bundle querySkus = new Bundle();
                    querySkus.putStringArrayList("ITEM_ID_LIST", skuList);

                    try {
                        Bundle skuDetails = inAppBillingService.getSkuDetails(3,
getPackageName(), "inapp", querySkus);
                        int response = skuDetails.getInt("RESPONSE_CODE");

                        if (response == 0) {
                            ArrayList<String> responseList =
skuDetails.getStringArrayList("DETAILS_LIST");

                            for (String thisResponse : responseList) {
```

```
                                JSONObject object = new JSONObject(thisResponse);
                                String sku = object.getString("productId");
                                String price = object.getString("price");

                                switch (sku) {
                                    case item1:
                                        buy1.setText(price);
                                        break;
                                    case item2:
                                        buy2.setText(price);
                                        break;
                                    case item3:
                                        buy3.setText(price);
                                        break;
                                    case item4:
                                        buy4.setText(price);
                                        break;
                                }
                            }
                        }
                    } catch (RemoteException | JSONException e) {
                        e.printStackTrace();
                    }
                }
            });
            thread.start();
        }
    }

    // Launch the PurchaseFlow passing the productID of the item the user wants to buy as a
parameter.
    private void buyItem(String productID) {
        if (inAppBillingService != null) {
            try {
                Bundle buyIntentBundle = inAppBillingService.getBuyIntent(3, getPackageName(),
productID, "inapp", "bGoa+V7g/yqDXvKRqq+JTFn4uQZbPiQJo4pf9RzJ");
                PendingIntent pendingIntent = buyIntentBundle.getParcelable("BUY_INTENT");
                startIntentSenderForResult(pendingIntent.getIntentSender(), 1003, new
Intent(), 0, 0, 0);
            } catch (RemoteException | IntentSender.SendIntentException e) {
                e.printStackTrace();
            }
        }
    }

    // Unbind the service in onDestroy(). If you don't unbind, the open
    // service connection could cause your device's performance to degrade.
    @Override
    public void onDestroy() {
        super.onDestroy();
        if (inAppBillingService != null) {
            unbindService(serviceConnection);
        }
    }

    // Check here if the in-app purchase was successful or not. If it was successful,
    // then consume the product, and let the app make the required changes.
    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
```

```
        if (requestCode == 1003 && resultCode == RESULT_OK) {

            final String purchaseData = data.getStringExtra("INAPP_PURCHASE_DATA");

            // Attention: You need to create a new thread here because
            // consumePurchase() triggers a network request, which can
            // cause lag to your app if it was called from the main thread.
            Thread thread = new Thread(new Runnable() {
                @Override
                public void run() {
                    try {
                        JSONObject jo = new JSONObject(purchaseData);
                        // Get the productID of the purchased item.
                        String sku = jo.getString("productId");
                        String productName = null;

                        // increaseCoins() here is a method used as an example in a game to
                        // increase the in-game currency if the purchase was successful.
                        // You should implement your own code here, and let the app apply
                        // the required changes after the purchase was successful.
                        switch (sku) {
                            case item1:
                                productName = "Item 1";
                                increaseCoins(2000);
                                break;
                            case item2:
                                productName = "Item 2";
                                increaseCoins(8000);
                                break;
                            case item3:
                                productName = "Item 3";
                                increaseCoins(18000);
                                break;
                            case item4:
                                productName = "Item 4";
                                increaseCoins(30000);
                                break;
                        }

                        // Consume the purchase so that the user is able to purchase the same
product again.
                        inAppBillingService.consumePurchase(3, getPackageName(),
jo.getString("purchaseToken"));
                        Toast.makeText(MainActivity.this, productName + " is successfully
purchased. Excellent choice, master!", Toast.LENGTH_LONG).show();
                    } catch (JSONException | RemoteException e) {
                        Toast.makeText(MainActivity.this, "Failed to parse purchase data.",
Toast.LENGTH_LONG).show();
                        e.printStackTrace();
                    }
                }
            });
            thread.start();
        }
    }
}
```

Étape 6:

Après avoir implémenté le code, vous pouvez le tester en déployant votre apk sur le canal bêta / alpha, et laissez les autres utilisateurs tester le code pour vous. Cependant, les véritables achats intégrés ne peuvent pas être effectués en mode de test. Vous devez d'abord publier votre application / jeu sur Play Store pour que tous les produits soient entièrement activés.

Plus d'informations sur les tests La facturation intégrée aux applications est disponible ici .

(Tierce partie) Bibliothèque In-App v3

Étape 1: Tout d'abord, suivez ces deux étapes pour ajouter des fonctionnalités d'application:

1. Ajoutez la bibliothèque en utilisant:

```
repositories {
           mavenCentral()
       }
       dependencies {
          compile 'com.anjlab.android.iab.v3:library:1.0.+'
       }
```

2. Ajoutez une autorisation dans le fichier manifeste.

```
<uses-permission android:name="com.android.vending.BILLING" />
```

Étape 2: Initialisez votre processeur de facturation:

```
BillingProcessor bp = new BillingProcessor(this, "YOUR LICENSE KEY FROM GOOGLE PLAY CONSOLE
HERE", this);
```

et implémenter le gestionnaire de facturation: BillingProcessor.IBillingHandler qui contient 4 méthodes: a. onBillingInitialized (); b. onProductPurchased (String productId, détails TransactionDetails): vous devez gérer les actions à effectuer après un achat réussi c. onBillingError (int errorCode, erreur pouvant être lancée): Gère toute erreur survenue lors du processus d'achat d. onPurchaseHistoryRestored (): pour restaurer des achats dans l'application

Étape 3: Comment acheter un produit.

Pour acheter un produit géré:

```
bp.purchase(YOUR_ACTIVITY, "YOUR PRODUCT ID FROM GOOGLE PLAY CONSOLE HERE");
```

Et pour acheter un abonnement:

```
bp.subscribe(YOUR_ACTIVITY, "YOUR SUBSCRIPTION ID FROM GOOGLE PLAY CONSOLE HERE");
```

Étape 4: Consommer un produit.

Pour consommer un produit, appelez simplement la méthode consumePurchase.

bp.consumePurchase ("VOTRE IDENTIFIANT DE CONSOLE GOOGLE PLAY ICI");

Pour d'autres méthodes liées à l'application, visitez github

Lire Facturation dans l'application en ligne: https://riptutorial.com/fr/android/topic/2843/facturation-dans-l-application

Chapitre 111: Fastjson

Introduction

Fastjson est une bibliothèque Java pouvant être utilisée pour convertir des objets Java dans leur représentation JSON. Il peut également être utilisé pour convertir une chaîne JSON en un objet Java équivalent.

Fastjson Caractéristiques:

Fournir les meilleures performances côté client et client Android

Fournit des `toJSONString()` simples de `toJSONString()` et `parseObject()` pour convertir des objets Java en JSON et vice-versa

Autoriser les objets non modifiables préexistants à être convertis vers et depuis JSON

Prise en charge étendue de Java Generics

Syntaxe

- Analyse d'objet (texte de chaîne)
- JSONObject parseObject (texte de chaîne)
- T parseObject (texte de chaîne, classe <T> clazz)
- JSONArray parseArray (texte de chaîne)
- <T> Liste <T> parseArray (texte de chaîne, classe <T> clazz)
- String toJSONString (objet Object)
- String toJSONString (objet Object, boolean prettyFormat)
- Object toJSON (Object javaObject)

Examples

Analyse JSON avec Fastjson

Vous pouvez regarder l'exemple dans la bibliothèque Fastjson

Encoder

```
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);
```

Sortie

```
{"id":0,"name":"admin","users":[{"id":2,"name":"guest"},{"id":3,"name":"root"}]}
```

Décoder

```
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;
    }
}
```

Convertir les données de type Map en 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);
```

Sortie

```
{1:"No.1",2:"No.2",3:[{"id":2,"name":"Liu"},{"id":3,"name":"Yue"}]}
```

Lire Fastjson en ligne: https://riptutorial.com/fr/android/topic/10865/fastjson

Chapitre 112: Feuilles de fond

Introduction

Une feuille inférieure est une feuille qui glisse du bord inférieur de l'écran.

Remarques

Les feuilles du bas glissent du bas de l'écran pour révéler plus de contenu.
Ils ont été ajoutés à la bibliothèque de support Android en version v23.2.0.

Examples

BottomSheetBehavior comme Google maps

2.1.x

Cet exemple dépend de la bibliothèque de support 23.4.0. +.

BottomSheetBehavior se caractérise par:

1. Deux barres d'outils avec des animations qui répondent aux mouvements de la feuille du bas.
2. Un FAB qui se cache près de la "barre d'outils modale" (celle qui apparaît lorsque vous glissez vers le haut).
3. Une image de fond derrière la feuille de fond avec un effet de parallaxe.
4. Un titre (TextView) dans la barre d'outils qui apparaît lorsque la feuille du bas l'atteint.
5. La barre de notification satus peut transformer son arrière-plan en couleur transparente ou pleine.
6. Un comportement de feuille de fond personnalisé avec un état "ancre".

Maintenant vérifions-les un par un:

Barres d'outils
Lorsque vous ouvrez cette vue dans Google Maps, vous pouvez voir une barre d'outils dans laquelle vous pouvez effectuer une recherche, la seule que je ne fais pas exactement comme Google Maps, car je voulais le faire plus générique. Quoi qu'il en soit, `ToolBar` trouve à l'intérieur d'un `AppBarLayout` et il a été masqué lorsque vous avez commencé à faire glisser la BottomSheet et il apparaît à nouveau lorsque la feuille de fond atteint l'état `COLLAPSED` .
Pour y parvenir, vous devez:

- créer un `Behavior` et l'étendre depuis `AppBarLayout.ScrollingViewBehavior`
- `onDependentViewChanged` méthodes `layoutDependsOn` et `onDependentViewChanged` . En faisant cela,

vous écouterez les mouvements bottomSheet.
- créer des méthodes pour masquer et afficher la barre d'outils AppBarLayout / ToolBar avec des animations.

Voici comment je l'ai fait pour la première barre d'outils ou ActionBar:

```
@Override
public boolean layoutDependsOn(CoordinatorLayout parent, View child, View dependency) {
    return dependency instanceof NestedScrollView;
}

@Override
public boolean onDependentViewChanged(CoordinatorLayout parent, View child,
                                      View dependency) {

    if (mChild == null) {
        initValues(child, dependency);
        return false;
    }

    float dVerticalScroll = dependency.getY() - mPreviousY;
    mPreviousY = dependency.getY();

    //going up
    if (dVerticalScroll <= 0 && !hidden) {
        dismissAppBar(child);
        return true;
    }

    return false;
}

private void initValues(final View child, View dependency) {

    mChild = child;
    mInitialY = child.getY();

    BottomSheetBehaviorGoogleMapsLike bottomSheetBehavior =
BottomSheetBehaviorGoogleMapsLike.from(dependency);
    bottomSheetBehavior.addBottomSheetCallback(new
BottomSheetBehaviorGoogleMapsLike.BottomSheetCallback() {
        @Override
        public void onStateChanged(@NonNull View bottomSheet,
@BottomSheetBehaviorGoogleMapsLike.State int newState) {
            if (newState == BottomSheetBehaviorGoogleMapsLike.STATE_COLLAPSED ||
                    newState == BottomSheetBehaviorGoogleMapsLike.STATE_HIDDEN)
                showAppBar(child);
        }

        @Override
        public void onSlide(@NonNull View bottomSheet, float slideOffset) {

        }
    });
}

private void dismissAppBar(View child){
    hidden = true;
    AppBarLayout appBarLayout = (AppBarLayout)child;
    mToolbarAnimation =
```

```
appBarLayout.animate().setDuration(mContext.getResources().getInteger(android.R.integer.config_shortAn

    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();
}
```

Voici le fichier complet si vous en avez besoin

La deuxième barre d'outils ou barre d'outils "Modal":
Vous devez remplacer les mêmes méthodes, mais dans celle-ci vous devez prendre en compte plus de comportements:

- afficher / masquer la barre d'outils avec des animations
- changer la couleur de la barre d'état / fond
- afficher / masquer le titre de la feuille de fond dans la barre d'outils
- fermer la bottomSheet ou l'envoyer à l'état réduit

Le code pour celui-ci est un peu long, alors je vais laisser le lien

Le FAB

Ceci est un comportement personnalisé également, mais s'étend de `FloatingActionButton.Behavior`. Dans `onDependentViewChanged` vous devez regarder quand il atteint le "offSet" ou le point où vous voulez le cacher. Dans mon cas, je veux le cacher quand il est proche de la deuxième barre d'outils, alors je creuse dans le parent FAB (un CoordinatorLayout) à la recherche de AppBarLayout contenant la barre d'outils, puis j'utilise la position ToolBar comme `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;
}
```

Lien Complet sur le Comportement FAB personnalisé

L'image derrière la feuille de fond avec effet de parallaxe :
Comme les autres, c'est un comportement personnalisé, la seule chose "compliquée" dans celui-ci est le petit algorithme qui garde l'image ancrée dans la feuille de fond et évite que l'image ne s'effondre comme l'effet de parallaxe par défaut:

```
@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;
}
```

Le fichier complet pour l'image de fond avec effet de parallaxe

Maintenant pour la fin: **le comportement de BottomSheet personnalisé**
Pour réaliser les 3 étapes, vous devez d'abord comprendre que BottomSheetBehavior par défaut comporte 5 états: `STATE_DRAGGING, STATE_SETTLING, STATE_EXPANDED, STATE_COLLAPSED, STATE_HIDDEN` et pour le comportement de Google Maps, vous devez ajouter un état intermédiaire entre `STATE_ANCHOR_POINT` .
J'ai essayé de prolonger le fichier bottomSheetBehavior par défaut sans succès, alors je viens de copier tout le code collé et de modifier ce dont j'ai besoin.
Pour réaliser ce dont je parle, suivez les étapes suivantes:

1. Créez une classe Java et étendez-la à partir de `CoordinatorLayout.Behavior<V>`
2. Copiez le code de `BottomSheetBehavior` fichier `BottomSheetBehavior` par défaut vers le nouveau.
3. Modifiez la méthode `clampViewPositionVertical` avec le code suivant:

```
@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. Ajouter un nouvel état

   public static final int STATE_ANCHOR_POINT = X;

5. Modifiez les méthodes suivantes: `onLayoutChild` , `onStopNestedScroll` , `BottomSheetBehavior<V> from(V view)` et `setState` (facultatif)

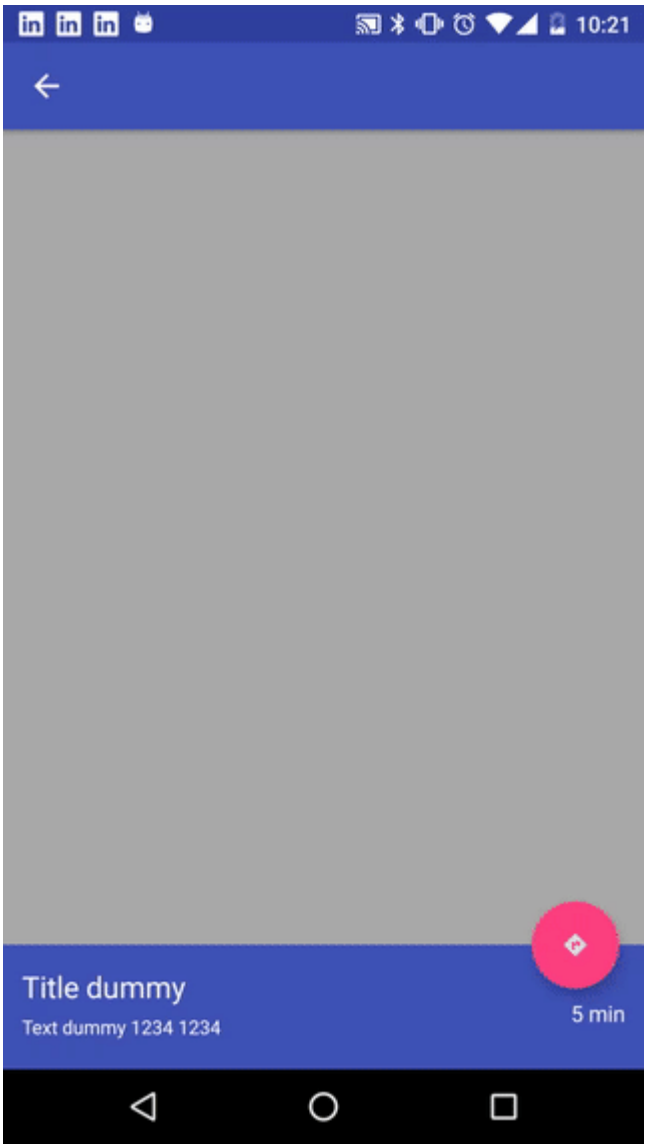
```
public boolean onLayoutChild(CoordinatorLayout parent, V child, int layoutDirection) {
    // First let the parent lay it out
    if (mState != STATE_DRAGGING && mState != STATE_SETTLING) {
        if (ViewCompat.getFitsSystemWindows(parent) &&
                !ViewCompat.getFitsSystemWindows(child)) {
            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;
}
```

Lien vers l'ensemble du projet où vous pouvez voir tous les comportements personnalisés

Et voici à quoi ça ressemble:
[



]

Installation rapide

Assurez-vous que la dépendance suivante est ajoutée au fichier build.gradle de votre application sous les dépendances:

```
compile 'com.android.support:design:25.3.1'
```

Vous pouvez ensuite utiliser la feuille inférieure en utilisant ces options:

- `BottomSheetBehavior` à utiliser avec `CoordinatorLayout`
- `BottomSheetDialog` qui est un dialogue avec un comportement de feuille inférieure
- `BottomSheetDialogFragment` qui est une extension de `DialogFragment` , qui crée un `BottomSheetDialog` au lieu d'une boîte de dialogue standard.

Feuilles de fond persistantes

Vous pouvez obtenir une feuille inférieure persistante associant un `BottomSheetBehavior` à un enfant. Vue d'un `CoordinatorLayout`

```
<android.support.design.widget.CoordinatorLayout >

    <!-- .....   -->

    <LinearLayout
       android:id="@+id/bottom_sheet"
       android:elevation="4dp"
       android:minHeight="120dp"
       app:behavior_peekHeight="120dp"
       ...
       app:layout_behavior="android.support.design.widget.BottomSheetBehavior">

           <!-- .....   -->

       </LinearLayout>

</android.support.design.widget.CoordinatorLayout>
```

Ensuite, dans votre code, vous pouvez créer une référence en utilisant:

```
 // The View with the BottomSheetBehavior
 View bottomSheet = coordinatorLayout.findViewById(R.id.bottom_sheet);
 BottomSheetBehavior mBottomSheetBehavior = BottomSheetBehavior.from(bottomSheet);
```

Vous pouvez définir l'état de votre BottomSheetBehavior en utilisant la méthode setState () :

```
mBottomSheetBehavior.setState(BottomSheetBehavior.STATE_EXPANDED);
```

Vous pouvez utiliser l'un de ces états:

- `STATE_COLLAPSED` : cet état réduit est la valeur par défaut et ne montre qu'une partie de la disposition en bas. La hauteur peut être contrôlée avec l'attribut `app:behavior_peekHeight` (la valeur par défaut est 0)
- `STATE_EXPANDED` : l'état complètement développé de la feuille du bas, où soit la totalité de la feuille inférieure est visible (si sa hauteur est inférieure à celle du `CoordinatorLayout`), soit la totalité du `CoordinatorLayout` est remplie
- `STATE_HIDDEN` : désactivé par défaut (et activé avec l'attribut `app:behavior_hideable`), ce qui permet aux utilisateurs de glisser le bas de la feuille pour masquer complètement la feuille du bas

Si vous souhaitez recevoir des rappels de modifications d'état, vous pouvez ajouter un `BottomSheetCallback` :

```
mBottomSheetBehavior.setBottomSheetCallback(new BottomSheetCallback() {
    @Override
    public void onStateChanged(@NonNull View bottomSheet, int newState) {
      // React to state change
    }
      @Override
      public void onSlide(@NonNull View bottomSheet, float slideOffset) {
       // React to dragging events
   }
```

```
});
```

Feuilles de fond modales avec BottomSheetDialogFragment

Vous pouvez réaliser des feuilles de fond modales en utilisant un `BottomSheetDialogFragment` .

Le `BottomSheetDialogFragment` est une feuille de fond modale.
Ceci est une version de `DialogFragment` qui affiche une feuille inférieure en utilisant `BottomSheetDialog` au lieu d'une boîte de dialogue flottante.

Définissez simplement le 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);
    }
}
```

Ensuite, utilisez ce code pour afficher le fragment:

```
MyBottomSheetDialogFragment mySheetDialog = new MyBottomSheetDialogFragment();
FragmentManager fm = getSupportFragmentManager();
mySheetDialog.show(fm, "modalSheetDialog");
```

Ce fragment va créer un `BottomSheetDialog` .

Feuilles de fond modales avec BottomSheetDialog

Le `BottomSheetDialog` est une boîte de dialogue conçue comme une feuille inférieure

Utilisez simplement:

```
//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();
```

Dans ce cas, vous n'avez pas besoin d'attacher un comportement BottomSheet.

Ouvrez BottomSheet DialogFragment en mode étendu par défaut.

BottomSheet DialogFragment s'ouvre dans `STATE_COLLAPSED` par défaut. Qui peut être forcé d'ouvrir à `STATE_EXPANDED` et prendre l'écran complet du périphérique avec l'aide du modèle de code suivant.

@NonNull @Override Dialogue public 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;
}
```

Bien que l'animation de dialogue soit légèrement perceptible, la tâche d'ouvrir le DialogFragment en plein écran est très bonne.

Lire Feuilles de fond en ligne: https://riptutorial.com/fr/android/topic/5702/feuilles-de-fond

Chapitre 113: Fichier Zip dans Android

Examples

Fichier Zip sur 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();
        }

      }

    }
```

Lire Fichier Zip dans Android en ligne: https://riptutorial.com/fr/android/topic/8137/fichier-zip-dans-android

Chapitre 114: Fil

Examples

Exemple de fil avec sa description

Lors du lancement d'une application, tout d'abord, le thread principal est exécuté. Ce thread principal gère tout le concept d'application de l'interface utilisateur. Si nous voulons exécuter longtemps la tâche dans laquelle nous n'avons pas besoin de l'interface utilisateur, nous utilisons le thread pour exécuter cette tâche en arrière-plan.

Voici l'exemple de Thread qui décrit le coup:

```
new Thread(new Runnable() {
    public void run() {
        for(int i = 1; i < 5;i++) {
            System.out.println(i);
        }
    }
}).start();
```

Nous pouvons créer un thread en créant l'objet de Thread qui a la méthode `Thread.run()` pour exécuter le thread. Ici, la méthode `run()` est appelée par la méthode `start()` .

Nous pouvons également exécuter les threads multiples de manière indépendante, appelée MultiThreading. Ce thread a également la fonctionnalité de veille par laquelle le thread en cours d'exécution s'endort (arrête temporairement l'exécution) pour le nombre de fois spécifié. Mais le sommeil lance l'InterruptedException Donc, nous devons le gérer en utilisant try / catch comme ça.

```
try{Thread.sleep(500);}catch(InterruptedException e){System.out.println(e);}
```

Mise à jour de l'interface utilisateur à partir d'un thread d'arrière-plan

Il est courant d'utiliser un thread d'arrière-plan pour effectuer des opérations réseau ou des tâches longues, puis mettre à jour l'interface utilisateur avec les résultats si nécessaire.

Cela pose un problème, car seul le thread principal peut mettre à jour l'interface utilisateur.

La solution consiste à utiliser la méthode `runOnUiThread()` , car elle vous permet de lancer l'exécution de code sur le thread d'interface utilisateur à partir d'un thread d'arrière-plan.

Dans cet exemple simple, un thread est démarré lors de la création de l'activité, s'exécute jusqu'à ce que le nombre magique de `42` soit généré de manière aléatoire, puis utilise la méthode `runOnUiThread()` pour mettre à jour l'interface une fois cette condition remplie.

```
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();
    }
}
```

Lire Fil en ligne: https://riptutorial.com/fr/android/topic/7131/fil

Chapitre 115: FileIO avec Android

Introduction

La lecture et l'écriture de fichiers sous Android ne sont pas différentes de la lecture et de l'écriture de fichiers en Java standard. Même package `java.io` peut être utilisé. Cependant, il existe des éléments spécifiques liés aux dossiers que vous êtes autorisé à écrire, des autorisations en général et des solutions de contournement MTP.

Remarques

Android fournit des moyens pour partager le fichier entre plusieurs applications, comme indiqué ici . Cela n'est pas obligatoire si une seule application crée et utilise le fichier.

Android fournit des options de stockage alternatives telles que les préférences partagées et privées, les bundles enregistrés et la base de données intégrée. Dans certains cas, ils constituent un meilleur choix que l'utilisation de fichiers simples.

L'activité Android a peu de méthodes spécifiques qui ressemblent à des remplacements des méthodes File IO standard Java. Par exemple, au lieu de `File.delete()` vous pouvez appeler `Context.deleteFile()` et au lieu d'appliquer récursivement `File.listFiles()` vous pouvez appeler `Context.fileList()` pour obtenir la liste de tous les fichiers spécifiques de votre application avec un peu moins. code. Cependant, ils ne fournissent pas de fonctionnalités supplémentaires au-delà du package `java.io` standard.

Examples

Obtenir le dossier de travail

Vous pouvez obtenir votre dossier de travail en appelant la méthode `getFilesDir()` sur votre activité (Activity est la classe centrale de votre application qui hérite de Context. Voir ici). La lecture n'est pas différente. Seule votre application aura accès à ce dossier.

Votre activité peut contenir le code suivant, par exemple:

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

Écrire un tableau brut d'octets

```
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()
```

Il n'y a rien Android spécifique à ce code. Si vous écrivez souvent beaucoup de petites valeurs, utilisez BufferedOutputStream pour réduire l'usure du SSD interne du périphérique.

Sérialiser l'objet

L'ancienne sérialisation d'objet Java est disponible pour vous dans Android. vous pouvez définir des classes sérialisables comme:

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

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

puis écrivez ensuite dans ObjectOutputStream:

```
File myFile = new File(getFilesDir(), "myObjects.bin");
FileOutputStream out = new FileOutputStream(myFile);
ObjectOutputStream oout = new ObjectOutputStream(new BufferedOutputStream(out));

oout.writeObject(new Circle(10, "One"));
oout.writeObject(new Circle(12, "Two"));

oout.close()
```

La sérialisation des objets Java peut être un choix parfait ou très mauvais, en fonction de ce que vous voulez en faire - en dehors de la portée de ce tutoriel et parfois des opinions. Lisez d'abord le contrôle de version si vous décidez de l'utiliser.

Ecriture sur stockage externe (carte SD)

Vous pouvez également lire et écrire depuis / vers une carte mémoire (carte SD) présente sur de nombreux appareils Android. Les fichiers de cet emplacement sont accessibles par d'autres programmes, également directement par l'utilisateur après la connexion du périphérique au PC via le câble USB et l'activation du protocole MTP.

La localisation de la carte SD est un peu plus problématique. La classe Environment contient des méthodes statiques pour obtenir des "répertoires externes" qui devraient normalement se trouver dans la carte SD, également des informations si la carte SD existe et est accessible en écriture. Cette question contient des réponses précieuses sur la manière de s'assurer que le bon emplacement sera trouvé.

L'accès au stockage externe nécessite des autorisations dans votre manifeste Android:

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

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

Pour les anciennes versions d'Android, il est suffisant de placer ces autorisations dans un manifeste (l'utilisateur doit approuver lors de l'installation). Toutefois, à partir d'Android 6.0, Android demande l'approbation de l'utilisateur au moment du premier accès, et vous devez prendre en charge cette nouvelle approche. Sinon, l'accès est refusé quel que soit votre manifeste.

Dans Android 6.0, vous devez d'abord vérifier l'autorisation, puis, s'il n'est pas accordé, le demander. Les exemples de code se trouvent dans cette question SO .

Résoudre le problème "Invisible MTP files".

Si vous créez des fichiers à exporter via un câble USB vers un ordinateur de bureau à l'aide du protocole MTP, il se peut que les nouveaux fichiers créés ne soient pas immédiatement visibles dans l'explorateur de fichiers exécuté sur le PC de bureau connecté. Pour rendre les nouveaux fichiers visibles, vous devez appeler MediaScannerConnection :

```
File file = new File(Environment.getExternalStoragePublicDirectory(
        Environment.DIRECTORY_DOCUMENTS), "theDocument.txt");
FileOutputStream out = new FileOutputStream(file)

... (write the document)

out.close()
MediaScannerConnection.scanFile(this, new String[] {file.getPath()}, null, null);
    context.sendBroadcast(new Intent(Intent.ACTION_MEDIA_SCANNER_SCAN_FILE,
            Uri.fromFile(file)));
```

Ce code d'appel MediaScannerConnection fonctionne uniquement pour les fichiers, pas pour les répertoires. Le problème est décrit dans ce rapport de bogue sur Android . Cela peut être corrigé pour une version future ou sur certains appareils.

Travailler avec de gros fichiers

Les petits fichiers sont traités en une fraction de seconde et vous pouvez les lire / écrire à la place du code où vous en avez besoin. Cependant, si le fichier est plus gros ou plus lent à traiter, vous devrez peut-être utiliser AsyncTask dans Android pour utiliser le fichier en arrière-plan:

```
    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
        }
    }
}
```

et alors

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

Cette question SO contient l'exemple complet sur la façon de créer et d'appeler AsyncTask. Consultez également la question sur la gestion des erreurs pour gérer les exceptions IOExceptions et autres erreurs.

Lire FileIO avec Android en ligne: https://riptutorial.com/fr/android/topic/8689/fileio-avec-android

Chapitre 116: FileProvider

Examples

Partage d'un fichier

Dans cet exemple, vous allez apprendre à partager un fichier avec d'autres applications. Nous allons utiliser un fichier pdf dans cet exemple, bien que le code fonctionne également avec tous les autres formats.

La feuille de route:

Spécifiez les répertoires dans lesquels les fichiers que vous souhaitez partager sont placés

Pour partager des fichiers, nous utiliserons FileProvider, une classe permettant le partage de fichiers sécurisé entre les applications. Un FileProvider ne peut partager que des fichiers dans des répertoires prédéfinis, définissons-les.

1. Créez un nouveau fichier XML qui contiendra les chemins, par exemple *res / xml / filepaths.xml*

2. Ajouter les chemins

```
<paths xmlns:android="http://schemas.android.com/apk/res/android">
    <files-path name="pdf_folder" path="documents/"/>
</paths>
```

Définir un FileProvider et le lier aux chemins de fichiers

Ceci est fait dans le manifeste:

```
<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>
```

Génère l'URI du fichier

Pour partager le fichier, nous devons fournir un identifiant pour le fichier. Cela se fait en utilisant un URI (Uniform Resource Identifier).

```
// We assume the file we want to load is in the documents/ subdirectory
// of the internal storage
File documentsPath = new File(Context.getFilesDir(), "documents");
File file = new File(documentsPath, "sample.pdf");
// This can also in one line of course:
// File file = new File(Context.getFilesDir(), "documents/sample.pdf");

Uri uri = FileProvider.getUriForFile(getContext(), "com.mydomain.fileprovider", file);
```

Comme vous pouvez le voir dans le code, nous créons d'abord une nouvelle classe File représentant le fichier. Pour obtenir une URI, nous demandons à FileProvider de nous en obtenir un. Le second argument est important: il passe l'autorité d'un FileProvider. Il doit être égal à l'autorité du FileProvider défini dans le manifeste.

Partager le fichier avec d'autres applications

Nous utilisons ShareCompat pour partager le fichier avec d'autres applications:

```
Intent intent = ShareCompat.IntentBuilder.from(getContext())
    .setType("application/pdf")
    .setStream(uri)
    .setChooserTitle("Choose bar")
    .createChooserIntent()
    .addFlags(Intent.FLAG_GRANT_READ_URI_PERMISSION);

Context.startActivity(intent);
```

Un sélecteur est un menu à partir duquel l'utilisateur peut choisir avec quelle application il souhaite partager le fichier. L'indicateur Intent.FLAG_GRANT_READ_URI_PERMISSION est nécessaire pour accorder une autorisation d'accès en lecture temporaire à l'URI.

Lire FileProvider en ligne: https://riptutorial.com/fr/android/topic/6266/fileprovider

Chapitre 117: Fileur

Examples

Ajouter un spinner à votre activité

Dans /res/values/strings.xml:

```
<string-array name="spinner_options">
    <item>Option 1</item>
    <item>Option 2</item>
    <item>Option 3</item>
</string-array>
```

Dans le format XML:

```
<Spinner
    android:id="@+id/spinnerName"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:entries="@array/spinner_options" />
```

En activité:

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

Exemple de base de spinner

Spinner C'est un type de liste déroulante. Tout d'abord dans la mise en page

```
  <Spinner
    android:id="@+id/spinner"     <!-- id to refer this spinner from JAVA-->
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

</Spinner>
```

Maintenant, remplissez les valeurs dans spinner Il existe principalement deux manières de renseigner les valeurs dans `spinner` .

1. À partir de XML même, créez un **tableau array.xml** dans le répertoire de **valeurs** sous **res** . Créer ce `array`

```
  <string-array name="defaultValue">
    <item>--Select City Area--</item>
    <item>--Select City Area--</item>
    <item>--Select City Area--</item>
</string-array>
```

Maintenant, ajoutez cette ligne dans XML sppiner

```
            android:entries="@array/defaultValue"
```

2. Vous pouvez également ajouter des valeurs via JAVA

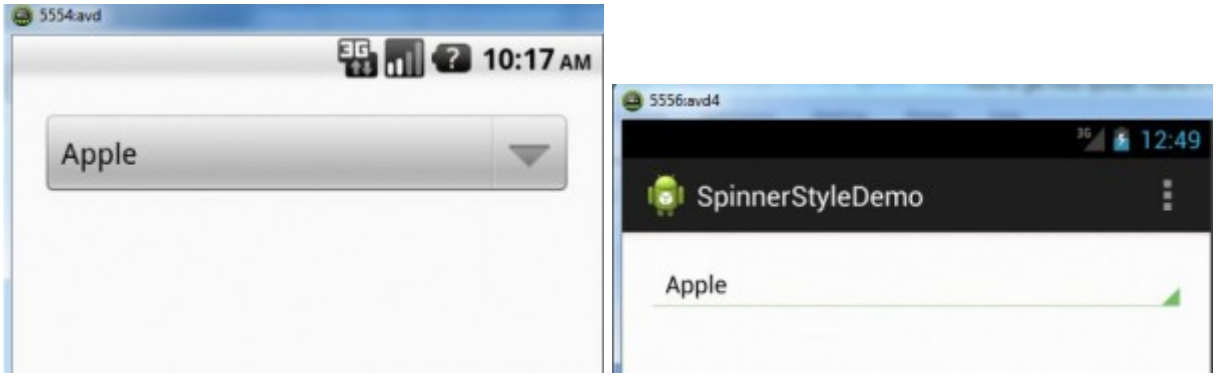
Si vous utilisez l' `activity` cityArea = (Spinner) findViewById (R.id.cityArea); sinon si vous utilisez dans le `fragment`

```
cityArea = (Spinner) findViewById(R.id.cityArea);
```

Maintenant, créez un `arrayList` de `Strings`

```
ArrayList<String> area = new ArrayList<>();
//add values in area arrayList
cityArea.setAdapter(new ArrayAdapter<String>(context
                            , android.R.layout.simple_list_item_1, area));
```

Cela ressemblera à



Selon le périphérique version Android, il rendra le style

Voici quelques-uns des thèmes par défaut

Si une application ne demande pas explicitement un thème dans son manifeste, le système Android déterminera le thème par défaut en fonction de la cible targetSdkVersion de l'application pour conserver les attentes d'origine de l'application:

Version du SDK Android	Thème par défaut
Version <11	@ android: style / thème
Version entre 11 et 13	@ android: style / theme.holo

Version du SDK Android	Thème par défaut
14 et plus	@ android: style / Theme.DeviceDefault

`Spinner` peut être facilement personnalisé à l'aide de xml, par exemple

```
 android:background="@drawable/spinner_background"

 android:layout_margin="16dp"

 android:padding="16dp"
```

Créez un arrière-plan personnalisé en XML et utilisez-le.

obtenir facilement la position et d'autres détails de l'élément sélectionné dans spinner

```
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) {

        }
    });
```

Modifier la couleur du texte de l'élément sélectionné dans spinner

Cela peut être fait de deux manières en XML

```
<item android:state_activated="true" android:color="@color/red"/>
```

Cela changera la couleur de l'élément sélectionné dans la fenêtre contextuelle.

et de JAVA faire ceci (dans le setOnItemSelectedListener (...))

```
 @Override
            public void onItemSelected(AdapterView<?> parent, View view, int position, long
id) {
                 ((TextView) parent.getChildAt(0)).setTextColor(0x00000000);
// similarly change `background color` etc.
            }
```

Lire Fileur en ligne: https://riptutorial.com/fr/android/topic/3459/fileur

Chapitre 118: Firebase

Introduction

Firebase est une plate-forme d'application mobile et Web avec des outils et une infrastructure conçus pour aider les développeurs à créer des applications de haute qualité.

Caractéristiques

Firebase Cloud Messaging, Firebase Auth, Base de données en temps réel, Firebase Storage, Firebase Hosting, Firebase Test Lab pour Android, Firebase Crash Reporting.

Remarques

Firebase - Documentation étendue:

Il y a une autre balise où vous pouvez trouver plus de sujets et d'exemples sur l'utilisation de Firebase.

Autres sujets connexes:

- Firebase Realtime DataBase
- Indexation des applications Firebase
- Rapport d'incident de Firebase
- Firebase Cloud Messaging

Examples

Créer un utilisateur Firebase

```
public class SignUpActivity extends BaseAppCompatActivity {

    @BindView(R.id.tIETSignUpEmail)
    EditText mEditEmail;
    @BindView(R.id.tIETSignUpPassword)
    EditText mEditPassword;

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
    }

    @OnClick(R.id.btnSignUpSignUp)
    void signUp() {

        FormValidationUtils.clearErrors(mEditEmail, mEditPassword);
```

```
        if (FormValidationUtils.isBlank(mEditEmail)) {
            mEditEmail.setError("Please enter email");
            return;
        }

        if (!FormValidationUtils.isEmailValid(mEditEmail)) {
            mEditEmail.setError("Please enter valid email");
            return;
        }

        if (TextUtils.isEmpty(mEditPassword.getText())) {
            mEditPassword.setError("Please enter password");
            return;
        }

        createUserWithEmailAndPassword(mEditEmail.getText().toString(),
mEditPassword.getText().toString());
    }

    private void createUserWithEmailAndPassword(String email, String password) {
        DialogUtils.showProgressDialog(this, "", getString(R.string.str_creating_account),
false);
        mFirebaseAuth
                .createUserWithEmailAndPassword(email, password)
                .addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {
                    @Override
                    public void onComplete(@NonNull Task<AuthResult> task) {
                        if (!task.isSuccessful()) {
                            Toast.makeText(SignUpActivity.this,
task.getException().getMessage(),
                                    Toast.LENGTH_SHORT).show();
                            DialogUtils.dismissProgressDialog();
                        } else {
                            Toast.makeText(SignUpActivity.this,
R.string.str_registration_successful, Toast.LENGTH_SHORT).show();
                            DialogUtils.dismissProgressDialog();
                            startActivity(new Intent(SignUpActivity.this,
HomeActivity.class));
                        }
                    }
                });
    }

    @Override
    protected int getLayoutResourceId() {
        return R.layout.activity_sign_up;
    }
}
```

Se connecter utilisateur Firebase avec email et mot de passe

```
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));
    }
}
```

Envoyer un e-mail de réinitialisation du mot de passe Firebase

```
public class ForgotPasswordActivity extends AppCompatActivity {

    @BindView(R.id.tIETForgotPasswordEmail)
    EditText mEditEmail;
    private FirebaseAuth mFirebaseAuth;
    private FirebaseAuth.AuthStateListener mAuthStateListener;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_forgot_password);
        ButterKnife.bind(this);

        mFirebaseAuth = FirebaseAuth.getInstance();

        mAuthStateListener = new FirebaseAuth.AuthStateListener() {
            @Override
            public void onAuthStateChanged(@NonNull FirebaseAuth firebaseAuth) {
                FirebaseUser firebaseUser = firebaseAuth.getCurrentUser();
                if (firebaseUser != null) {
                    // Do whatever you want with the UserId by firebaseUser.getUid()
                } else {

                }
            }
        };
    }

    @Override
    protected void onStart() {
        super.onStart();
        mFirebaseAuth.addAuthStateListener(mAuthStateListener);
    }

    @Override
    protected void onStop() {
        super.onStop();
        if (mAuthStateListener != null) {
            mFirebaseAuth.removeAuthStateListener(mAuthStateListener);
        }
    }

    @OnClick(R.id.btnForgotPasswordSubmit)
    void onSubmitClick() {
```

```
        if (FormValidationUtils.isBlank(mEditEmail)) {
            FormValidationUtils.setError(null, mEditEmail, "Please enter email");
            return;
        }

        if (!FormValidationUtils.isEmailValid(mEditEmail)) {
            FormValidationUtils.setError(null, mEditEmail, "Please enter valid email");
            return;
        }

        DialogUtils.showProgressDialog(this, "", "Please wait...", false);
        mFirebaseAuth.sendPasswordResetEmail(mEditEmail.getText().toString())
                .addOnCompleteListener(new OnCompleteListener<Void>() {
                    @Override
                    public void onComplete(@NonNull Task<Void> task) {
                        DialogUtils.dismissProgressDialog();
                        if (task.isSuccessful()) {
                            Toast.makeText(ForgotPasswordActivity.this, "An email has been
sent to you.", Toast.LENGTH_SHORT).show();
                            finish();
                        } else {
                            Toast.makeText(ForgotPasswordActivity.this,
task.getException().getMessage(), Toast.LENGTH_SHORT).show();
                        }
                    }
                });
    }
}
```

Mise à jour du courrier électronique d'un utilisateur Firebase

```
public class ChangeEmailActivity extends BaseAppCompatActivity implements
ReAuthenticateDialogFragment.OnReauthenticateSuccessListener {

    @BindView(R.id.et_change_email)
    EditText mEditText;
    private FirebaseUser mFirebaseUser;

    @OnClick(R.id.btn_change_email)
    void onChangeEmailClick() {

        FormValidationUtils.clearErrors(mEditText);

        if (FormValidationUtils.isBlank(mEditText)) {
            FormValidationUtils.setError(null, mEditText, "Please enter email");
            return;
        }

        if (!FormValidationUtils.isEmailValid(mEditText)) {
            FormValidationUtils.setError(null, mEditText, "Please enter valid email");
            return;
        }

        changeEmail(mEditText.getText().toString());
    }

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

```
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        mFirebaseUser = mFirebaseAuth.getCurrentUser();
    }

    private void changeEmail(String email) {
        DialogUtils.showProgressDialog(this, "Changing Email", "Please wait...", false);
        mFirebaseUser.updateEmail(email)
                .addOnCompleteListener(new OnCompleteListener<Void>() {
                    @Override
                    public void onComplete(@NonNull Task<Void> task) {
                        DialogUtils.dismissProgressDialog();
                        if (task.isSuccessful()) {
                            showToast("Email updated successfully.");
                            return;
                        }

                        if (task.getException() instanceof
FirebaseAuthRecentLoginRequiredException) {
                            FragmentManager fm = getSupportFragmentManager();
                            ReAuthenticateDialogFragment reAuthenticateDialogFragment = new
ReAuthenticateDialogFragment();
                            reAuthenticateDialogFragment.show(fm,
reAuthenticateDialogFragment.getClass().getSimpleName());
                        }
                    }
                });
    }

    @Override
    protected int getLayoutResourceId() {
        return R.layout.activity_change_email;
    }

    @Override
    public void onReauthenticateSuccess() {
        changeEmail(mEditText.getText().toString());
    }
}
```

Changer le mot de passe

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

Ré-authentifier l'utilisateur Firebase

```
public class ReAuthenticateDialogFragment extends DialogFragment {

    @BindView(R.id.et_dialog_reauthenticate_email)
    EditText mEditTextEmail;
    @BindView(R.id.et_dialog_reauthenticate_password)
    EditText mEditTextPassword;
    private OnReauthenticateSuccessListener mOnReauthenticateSuccessListener;

    @OnClick(R.id.btn_dialog_reauthenticate)
    void onReauthenticateClick() {

        FormValidationUtils.clearErrors(mEditTextEmail, mEditTextPassword);

        if (FormValidationUtils.isBlank(mEditTextEmail)) {
```

```
            FormValidationUtils.setError(null, mEditTextEmail, "Please enter email");
            return;
        }

        if (!FormValidationUtils.isEmailValid(mEditTextEmail)) {
            FormValidationUtils.setError(null, mEditTextEmail, "Please enter valid email");
            return;
        }

        if (TextUtils.isEmpty(mEditTextPassword.getText())) {
            FormValidationUtils.setError(null, mEditTextPassword, "Please enter password");
            return;
        }

        reauthenticateUser(mEditTextEmail.getText().toString(),
mEditTextPassword.getText().toString());
    }

    private void reauthenticateUser(String email, String password) {
        DialogUtils.showProgressDialog(getActivity(), "Re-Authenticating", "Please wait...",
false);
        FirebaseUser firebaseUser = FirebaseAuth.getInstance().getCurrentUser();
        AuthCredential authCredential = EmailAuthProvider.getCredential(email, password);
        firebaseUser.reauthenticate(authCredential)
                .addOnCompleteListener(new OnCompleteListener<Void>() {
                    @Override
                    public void onComplete(@NonNull Task<Void> task) {
                        DialogUtils.dismissProgressDialog();
                        if (task.isSuccessful()) {
                            mOnReauthenticateSuccessListener.onReauthenticateSuccess();
                            dismiss();
                        } else {
                            ((BaseAppCompatActivity)
getActivity()).showToast(task.getException().getMessage());
                        }
                    }
                });
    }

    @Override
    public void onAttach(Context context) {
        super.onAttach(context);
        mOnReauthenticateSuccessListener = (OnReauthenticateSuccessListener) context;
    }

    @OnClick(R.id.btn_dialog_reauthenticate_cancel)
    void onCancelClick() {
        dismiss();
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.dialog_reauthenticate, container);
        ButterKnife.bind(this, view);
        return view;
    }

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

```
        Window window = getDialog().getWindow();
        window.setLayout(WindowManager.LayoutParams.MATCH_PARENT,
WindowManager.LayoutParams.WRAP_CONTENT);
    }

    interface OnReauthenticateSuccessListener {
        void onReauthenticateSuccess();
    }
}
```

Opérations de stockage Firebase

Avec cet exemple, vous pourrez effectuer les opérations suivantes:

1. Se connecter au stockage Firebase
2. Créer un répertoire nommé "images"
3. Télécharger un fichier dans le répertoire images
4. Télécharger un fichier depuis le répertoire images
5. Supprimer un fichier du répertoire images

```
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();
    }
}
```

Par défaut, les règles de stockage Firebase appliquent la restriction d'authentification. Si l'utilisateur est authentifié, alors seulement, il peut effectuer des opérations sur Firebase Storage, sinon il ne le peut pas. J'ai désactivé la partie d'authentification dans cette démonstration en mettant à jour les règles de stockage. Auparavant, les règles ressemblaient à:

```
 service firebase.storage {
   match /b/**something**.appspot.com/o {
     match /{allPaths=**} {
       allow read, write: if request.auth != null;
     }
   }
 }
```

Mais j'ai changé pour ignorer l'authentification:

```
service firebase.storage {
  match /b/**something**.appspot.com/o {
    match /{allPaths=**} {
      allow read, write;
    }
  }
}
```

Firebase Cloud Messaging

Tout d'abord, vous devez configurer votre projet en ajoutant Firebase à votre projet Android en suivant les étapes décrites dans cette rubrique .

Configurer Firebase et le SDK FCM

Ajoutez la dépendance FCM à votre fichier `build.gradle` niveau de l' `build.gradle`

```
dependencies {
 compile 'com.google.firebase:firebase-messaging:11.0.4'
}
```

Et tout en bas (c'est important) ajouter:

```
// ADD THIS AT THE BOTTOM
apply plugin: 'com.google.gms.google-services'
```

Modifier le manifeste de votre application

Ajoutez les éléments suivants au manifeste de votre application:

- Un service qui étend `FirebaseMessagingService` . Ceci est nécessaire si vous souhaitez effectuer une gestion des messages au-delà de la réception des notifications sur les applications en arrière-plan.
- Un service qui étend `FirebaseInstanceIdService` pour gérer la création, la rotation et la mise à jour des jetons d'inscription.

Par exemple:

```
    <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>
```

Voici des implémentations simples des 2 services.

Pour récupérer le jeton d'enregistrement actuel, `onTokenRefresh()` la classe `FirebaseInstanceIdService` et remplacez la méthode `onTokenRefresh()` :

```
public class MyInstanceIdListenerService extends FirebaseInstanceIdService {

    // Called if InstanceID token is updated. Occurs if the security of the previous token had
been
    // compromised. This call is initiated by the InstanceID provider.
    @Override
    public void onTokenRefresh() {
        // Get updated InstanceID token.
        String refreshedToken = FirebaseInstanceId.getInstance().getToken();

        // Send this token to your server or store it locally
    }
}
```

Pour recevoir des messages, utilisez un service qui étend `FirebaseMessagingService` et remplacez la méthode `onMessageReceived` .

```
public class MyFcmListenerService extends FirebaseMessagingService {

    /**
     * Called when message is received.
     *
     * @param remoteMessage Object representing the message received from Firebase Cloud
Messaging.
     */
    @Override
    public void onMessageReceived(RemoteMessage remoteMessage) {
        String from = remoteMessage.getFrom();

        // Check if message contains a data payload.
        if (remoteMessage.getData().size() > 0) {
            Log.d(TAG, "Message data payload: " + remoteMessage.getData());
            Map<String, String> data = remoteMessage.getData();
        }

        // Check if message contains a notification payload.
        if (remoteMessage.getNotification() != null) {
            Log.d(TAG, "Message Notification Body: " +
remoteMessage.getNotification().getBody());
        }

        // do whatever you want with this, post your own notification, or update local state
    }
```

Dans **Firebase, vous** pouvez regrouper les utilisateurs en fonction de leur comportement, par exemple "AppVersion, utilisateur gratuit, utilisateur d’achat ou règles spécifiques", puis envoyer une notification à un groupe spécifique en envoyant une fonctionnalité de **rubrique** dans FireBase.

enregistrer l'utilisateur dans l'utilisation du sujet

```
FirebaseMessaging.getInstance().subscribeToTopic("Free");
```

puis, dans la console fireBase, envoyer une notification par nom de sujet

Plus d'informations dans la rubrique dédiée Firebase Cloud Messaging .

Ajouter Firebase à votre projet Android

Voici des étapes simplifiées (basées sur la documentation officielle) requises pour créer un projet Firebase et le connecter à une application Android.

Ajouter Firebase à votre application

1. Créez un projet Firebase dans la console Firebase et cliquez sur **Créer un nouveau projet** .
2. Cliquez sur **Ajouter Firebase à votre application Android** et suivez les étapes de configuration.
3. Lorsque vous y êtes invité, entrez **le nom du package de** votre **application** .
   Il est important de saisir le nom complet du package utilisé par votre application. Cela ne peut être défini que lorsque vous ajoutez une application à votre projet Firebase.
4. À la fin, vous allez télécharger un fichier `google-services.json` . Vous pouvez télécharger ce fichier à tout moment.
5. Si vous ne l'avez pas déjà fait, copiez le fichier `google-services.json` dans le dossier du module de votre projet, généralement `app/` .

L'étape suivante consiste à ajouter le SDK pour intégrer les bibliothèques Firebase dans le projet.

Ajouter le SDK

Pour intégrer les bibliothèques Firebase dans un de vos propres projets, vous devez effectuer quelques tâches de base pour préparer votre projet Android Studio. Vous l'avez peut-être déjà fait dans le cadre de l'ajout de Firebase à votre application.

1. Ajoutez des règles à votre fichier `build.gradle` niveau `build.gradle` pour inclure le **plug-in google-services** :

```
buildscript {
    // ...
    dependencies {
        // ...
        classpath 'com.google.gms:google-services:3.1.0'
    }
```

```
}
```

Ensuite, dans le fichier Gradle de votre module (généralement `app/build.gradle`), ajoutez la ligne de plug-in apply au bas du fichier pour activer le plug-in Gradle:

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

android {
  // ...
}

dependencies {
  // ...
  compile 'com.google.firebase:firebase-core:11.0.4'
}

// ADD THIS AT THE BOTTOM
apply plugin: 'com.google.gms.google-services'
```

La dernière étape consiste à ajouter les dépendances du kit SDK Firebase en utilisant une ou plusieurs **bibliothèques disponibles** pour les différentes fonctionnalités Firebase.

Ligne de dépendance Gradle	Un service
com.google.firebase: firebase-core: 11.0.4	Analytique
com.google.firebase: base de données firebase: 11.0.4	Base de données en temps réel
com.google.firebase: firebase-storage: 11.0.4	Espace de rangement
com.google.firebase: crash-firebase: 11.0.4	Rapport de collision
com.google.firebase: firebase-auth: 11.0.4	Authentification
com.google.firebase: Messagerie-firebase: 11.0.4	Cloud Messaging / Notifications
com.google.firebase: firebase-config: 11.0.4	Configuration à distance
com.google.firebase: invite-firebase: 11.0.4	Invites / Liens dynamiques
com.google.firebase: annonces firebase: 11.0.4	AdMob
com.google.android.gms: play-services-appindexing: 11.0.4	Indexation des applications

Firebase Realtime Database: comment définir / obtenir des données

Remarque: configurons une authentification anonyme pour l'exemple

```
{
```

```
  "rules": {
    ".read": "auth != null",
    ".write": "auth != null"
  }
}
```

Une fois cela fait, créez un enfant en modifiant votre adresse de base de données. Par exemple:

https://your-project.firebaseio.com/ à **https://your-project.firebaseio.com/chat**

Nous allons mettre des données à cet endroit à partir de notre appareil Android. Vous **n'avez pas besoin de** créer la structure de la base de données (onglets, champs, etc.), elle sera automatiquement créée lorsque vous enverrez un objet Java à Firebase!

Créez un objet Java contenant tous les attributs que vous souhaitez envoyer à la base de données:

```
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;
    }
}
```

Puis dans votre activité:

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

Pour envoyer une valeur:

```
ChatMessage msg = new ChatMessage("user1", "Hello World!");
reference.push().setValue(msg);
```

Pour recevoir les modifications qui se produisent dans la base de données:

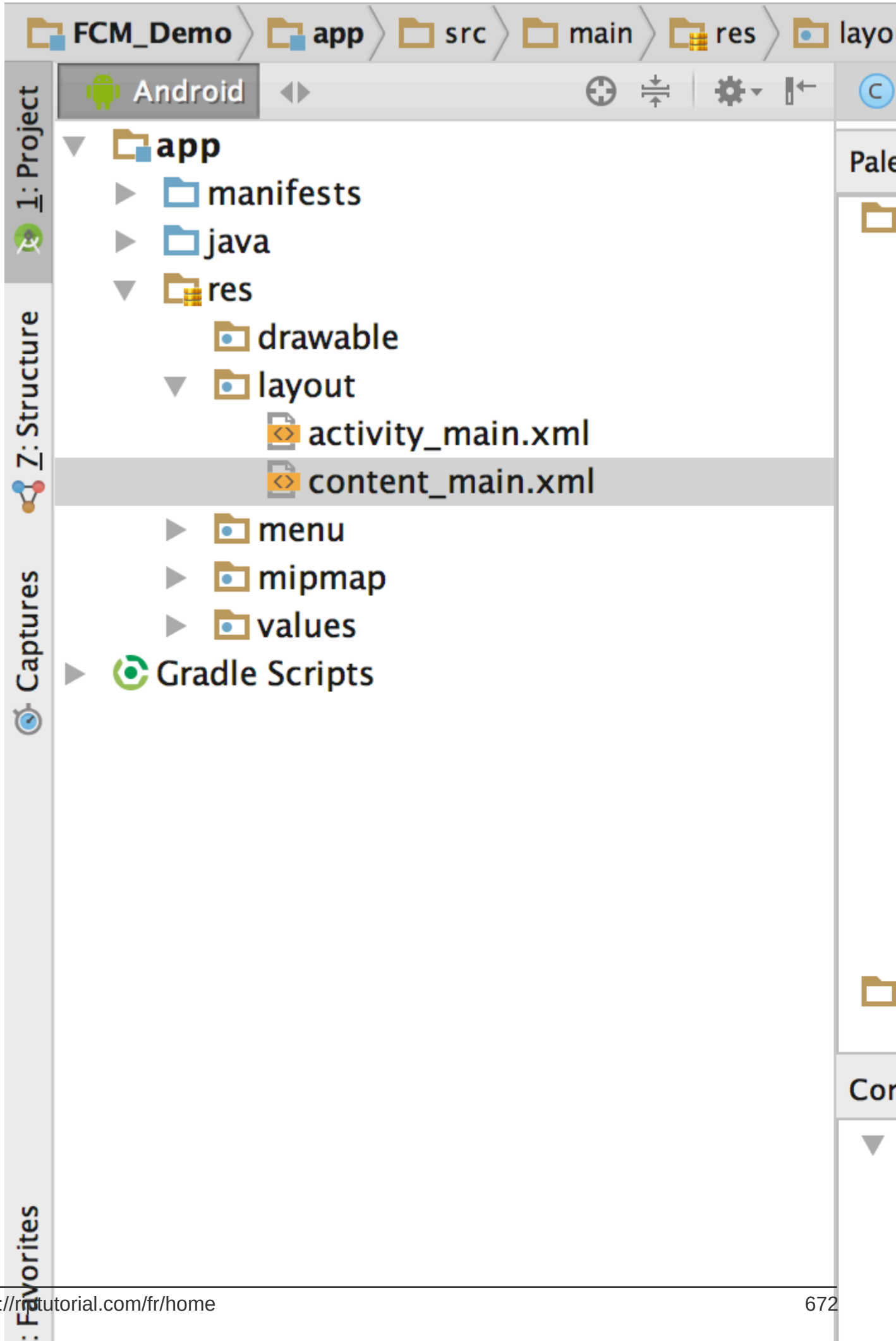
```
reference.addChildEventListener(new ChildEventListener() {
    @Override
    public void onChildAdded(DataSnapshot dataSnapshot, String s) {
        ChatMessage msg = dataSnapshot.getValue(ChatMessage.class);
        Log.d(TAG, msg.getUsername()+" "+msg.getMessage());
    }

    public void onChildChanged(DataSnapshot dataSnapshot, String s) {}
    public void onChildRemoved(DataSnapshot dataSnapshot) {}
    public void onChildMoved(DataSnapshot dataSnapshot, String s) {}
    public void onCancelled(DatabaseError databaseError) {}
});
```

Démo des notifications basées sur FCM

Cet exemple montre comment utiliser la plate-forme Firebase Cloud Messaging (FCM). FCM est un successeur de Google Cloud Messaging (GCM). Il ne nécessite pas les autorisations C2D_MESSAGE des utilisateurs de l'application.

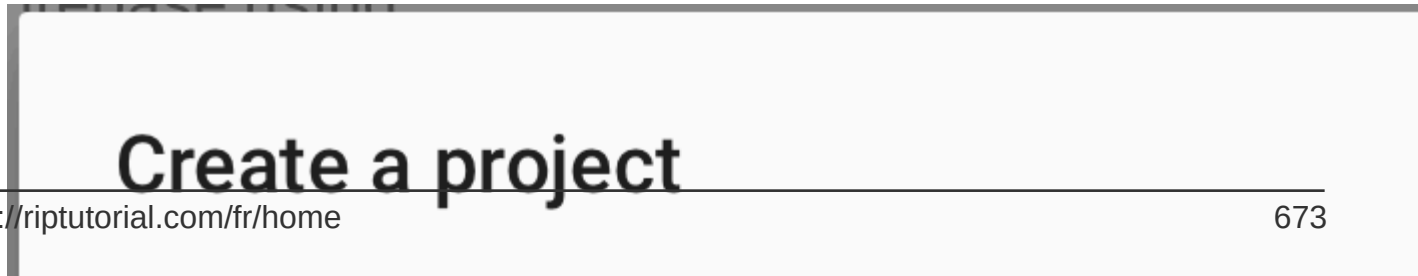
Les étapes pour intégrer le FCM sont les suivantes.

1. Créer un exemple de projet hello world dans Android Studio Votre écran de studio Android ressemblerait à l'image suivante.


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. L'étape suivante consiste à configurer le projet Firebase. Rendez-vous sur https://console.firebase.google.com et créez un projet avec un nom identique afin de pouvoir le suivre facilement.

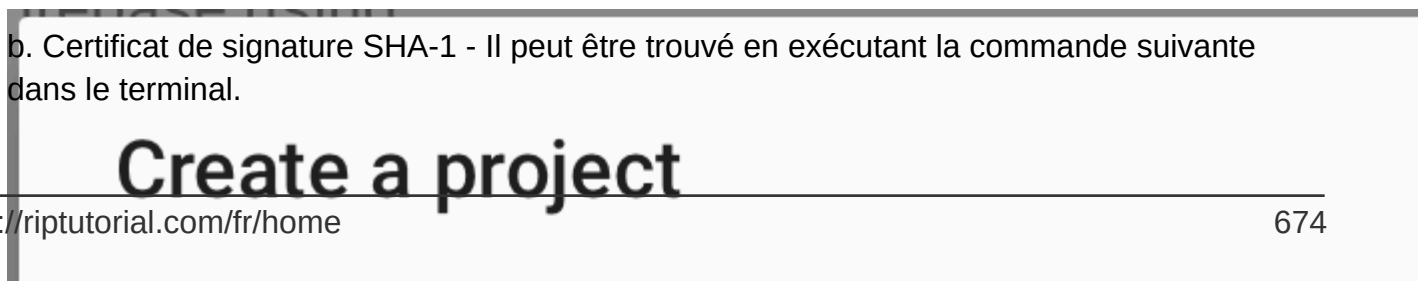
https://console.firebase.google.com et créez un projet avec un nom identique afin de pouvoir le suivre facilement.



3. Il est maintenant temps d'ajouter Firebase à votre exemple de projet Android que vous venez de créer. Vous aurez besoin du nom du package de votre projet et du certificat de signature de débogage SHA-1 (facultatif).

   une. Nom du paquet - Il peut être trouvé à partir du fichier XML du manifeste android.

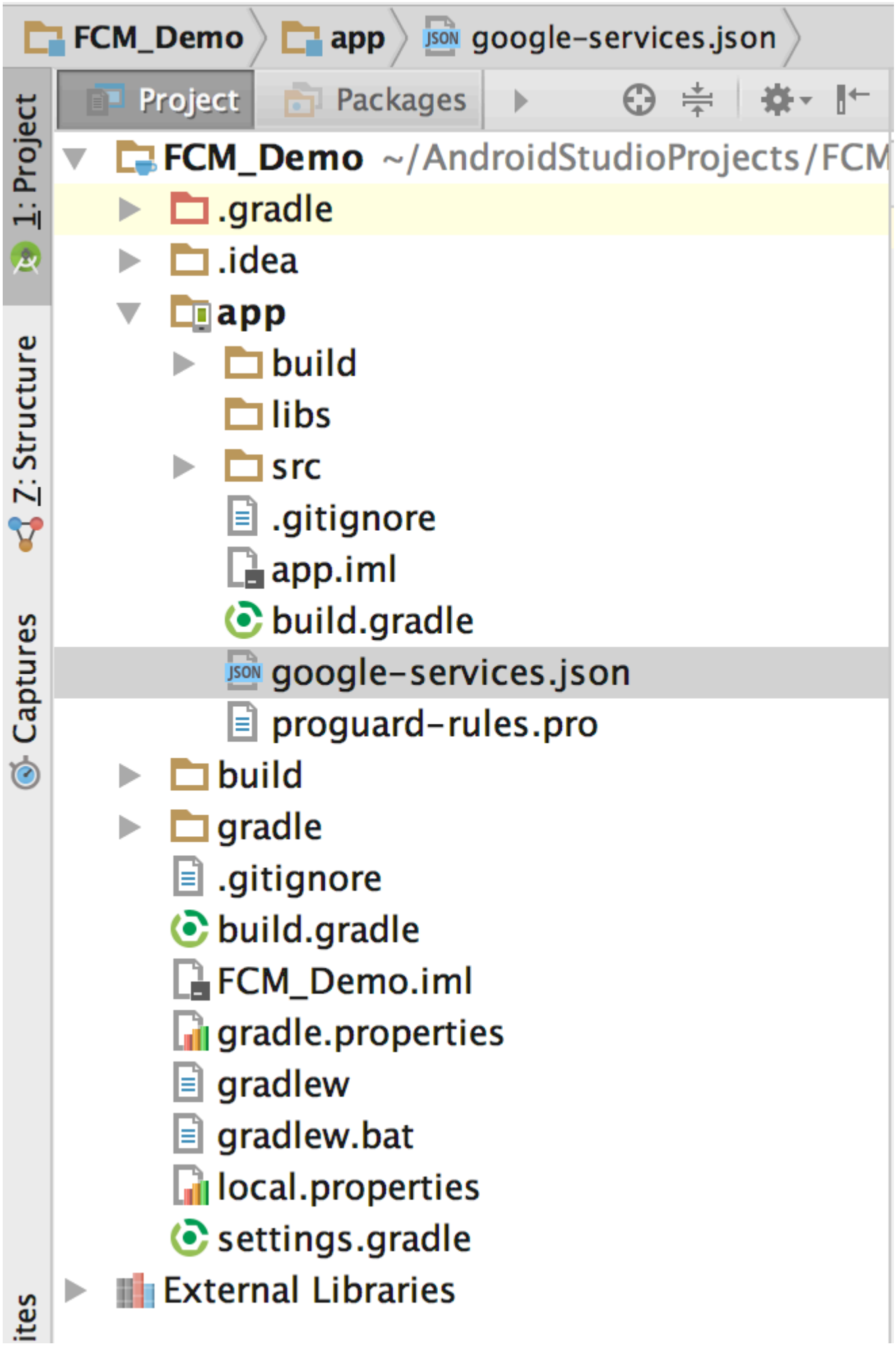
   b. Certificat de signature SHA-1 - Il peut être trouvé en exécutant la commande suivante dans le terminal.



```
keytool -list -v -keystore ~/.android/debug.keystore -alias androiddebugkey -storepass android -keypass android
```

Entrez ces informations dans la console firebase et ajoutez l'application au projet firebase. Une fois que vous cliquez sur le bouton Ajouter une application, votre navigateur télécharge automatiquement un fichier JSON nommé "google-services.json".

4. Copiez maintenant le fichier google-services.json que vous venez de télécharger dans le répertoire racine de votre module d'application Android.


FCM_Demo
app
google-services.json
Project
Packages
1: Project
Z: Structure
Captures
FCM_Demo ~/AndroidStudioProjects/FCM
.gradle
.idea
app
build
libs
src
.gitignore
app.iml
build.gradle
google-services.json
proguard-rules.pro
build
gradle
.gitignore
build.gradle
FCM_Demo.iml
gradle.properties
gradlew
gradlew.bat
local.properties
settings.gradle
External Libraries

5. Suivez les instructions données sur la console Firebase pendant que vous avancez. une. Ajoutez la ligne de code suivante à votre niveau de projet build.gradle

```
dependencies{ classpath 'com.google.gms:google-services:3.1.0' .....
```

b. Ajoutez la ligne de code suivante à la fin de votre niveau d'application build.gradle.

```
    //following are the dependencies to be added
    compile 'com.google.firebase:firebase-messaging:11.0.4'
    compile 'com.android.support:multidex:1.0.1'
}
// this line goes to the end of the file
apply plugin: 'com.google.gms.google-services'
```

c. Le studio Android vous demanderait de synchroniser le projet. Cliquez sur Synchroniser maintenant.

6. La tâche suivante consiste à ajouter deux services. une. Une extension FirebaseMessagingService avec intent-filter comme suit

```
    <intent-filter>
        <action android:name="com.google.firebase.MESSAGING_EVENT"/>
    </intent-filter>
```

b. Une extension FirebaseInstanceIDService.

```
<intent-filter>
    <action android:name="com.google.firebase.INSTANCE_ID_EVENT"/>
</intent-filter>
```

7. Le code FirebaseMessagingService devrait ressembler à ceci.

```
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 devrait ressembler à ceci.

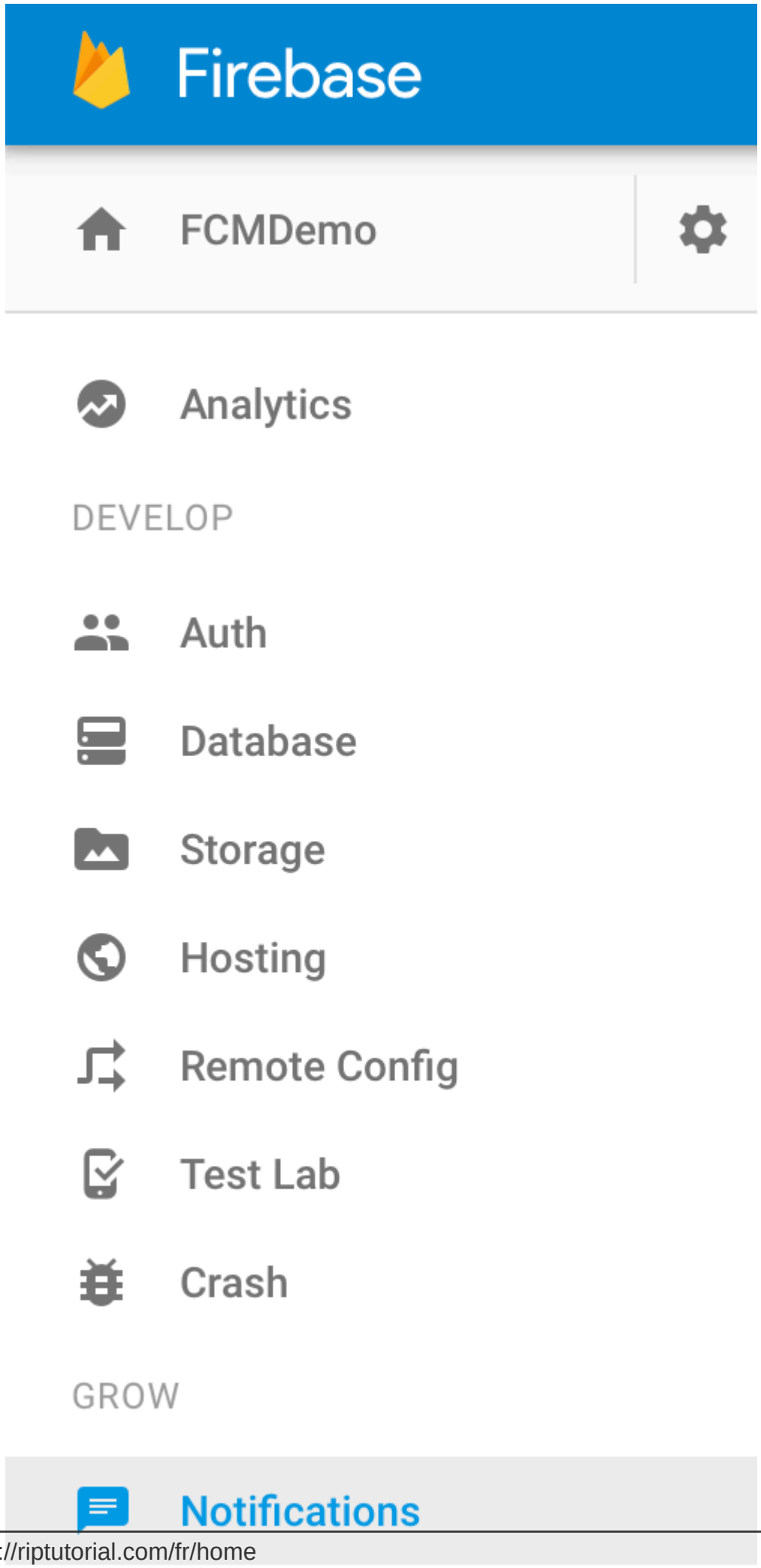
```
import android.app.Service;
import android.content.Intent;
import android.os.IBinder;

import com.google.firebase.iid.FirebaseInstanceIdService;

public class MyFirebaseInstanceIDService extends FirebaseInstanceIdService {
    public MyFirebaseInstanceIDService() {
    }
```

```
}
```

9. Il est maintenant temps de capturer le jeton d'enregistrement du périphérique. Ajoutez la ligne de code suivante à la méthode onCreate de MainActivity.

```
String token = FirebaseInstanceId.getInstance().getToken();
Log.d("FCMAPP", "Token is "+token);
```

10. Une fois que nous avons le jeton d'accès, nous pouvons utiliser la console firebase pour envoyer la notification. Exécutez l'application sur votre combiné Android.


Firebase
FCMDemo
Analytics
DEVELOP
Auth
Database
Storage
Hosting
Remote Config
Test Lab
Crash
GROW
Notifications

Chapitre 119: Firebase Cloud Messaging

Introduction

Firebase Cloud Messaging (FCM) est une solution de messagerie multiplate-forme qui vous permet de diffuser des messages de manière fiable et sans frais.

En utilisant FCM, vous pouvez notifier à une application cliente qu'un nouvel email ou d'autres données sont disponibles pour la synchronisation. Vous pouvez envoyer des messages de notification pour activer le réengagement et la rétention des utilisateurs. Pour les cas d'utilisation tels que la messagerie instantanée, un message peut transférer jusqu'à 4 `4KB` de charge utile vers une application client.

Examples

Configurer une application client Firebase Cloud Messaging sur Android

1. Complétez la partie Installation et configuration pour connecter votre application à Firebase. Cela créera le projet dans Firebase.

2. Ajoutez la dépendance de Firebase Cloud Messaging à votre fichier `build.gradle` niveau du `build.gradle` :

```
dependencies {
     compile 'com.google.firebase:firebase-messaging:10.2.1'
}
```

Vous êtes maintenant prêt à travailler avec le FCM sous Android.

Les clients FCM nécessitent des périphériques exécutant `Android 2.3` ou version ultérieure, ainsi que l'application Google Play Store ou un émulateur exécutant Android 2.3 avec les API Google.

Modifier votre fichier `AndroidManifest.xml`

```
<service
    android:name=".MyFirebaseMessagingService">
    <intent-filter>
        <action android:name="com.google.firebase.MESSAGING_EVENT"/>
    </intent-filter>
</service>

<service
    android:name=".MyFirebaseInstanceIDService">
    <intent-filter>
        <action android:name="com.google.firebase.INSTANCE_ID_EVENT"/>
    </intent-filter>
</service>
```

Jeton d'inscription

Au démarrage initial de votre application, le kit SDK FCM génère un jeton d'enregistrement pour l'instance d'application cliente.
Si vous souhaitez cibler des périphériques uniques ou créer des groupes de périphériques, vous devez accéder à ce jeton en étendant `FirebaseInstanceIdService` .

Le rappel `onTokenRefresh` déclenche chaque fois qu'un nouveau jeton est généré et vous pouvez utiliser la méthode `FirebaseInstanceID.getToken()` pour récupérer le jeton actuel.

Exemple:

```
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);

    }

}
```

Ce code que j'ai implanté dans mon application pour repousser l'image, le message et aussi le lien pour ouvrir dans votre webView

Ceci est mon 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;

    }
}}
```

Et ceci est MainActivity pour ouvrir le lien dans mon WebView ou autre navigateur dépand sur vos besoins par le biais d'intents.

```
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();
    }}
```

Recevoir des messages

Pour recevoir des messages, utilisez un service qui étend `FirebaseMessagingService` et remplacez la méthode `onMessageReceived` .

```
public class MyFcmListenerService extends FirebaseMessagingService {

    /**
     * Called when message is received.
     *
     * @param remoteMessage Object representing the message received from Firebase Cloud
Messaging.
     */
    @Override
    public void onMessageReceived(RemoteMessage message) {
        String from = message.getFrom();

        // Check if message contains a data payload.
        if (remoteMessage.getData().size() > 0) {
            Log.d(TAG, "Message data payload: " + remoteMessage.getData());
            Map<String, String> data = message.getData();
        }

        // Check if message contains a notification payload.
        if (remoteMessage.getNotification() != null) {
            Log.d(TAG, "Message Notification Body: " +
remoteMessage.getNotification().getBody());
        }

        //.....
    }
```

Lorsque l'application est en arrière-plan, Android dirige les messages de notification vers la barre d'état système. Un utilisateur touche la notification pour ouvrir le lanceur d'applications par défaut.

Cela inclut les messages contenant à la fois une notification et des données utiles (et tous les messages envoyés depuis la console Notifications). Dans ces cas, la notification est transmise à la barre d'état système du périphérique et la charge de données est fournie dans les extras du but de votre activité de lancement.

Voici un bref récapitulatif:

État App	Notification	Les données	Tous les deux
Premier plan	`onMessageReceived`	`onMessageReceived`	`onMessageReceived`
Contexte	Barre d'état système	`onMessageReceived`	Notification: barre d'état système
			Données: dans les extras de l'intention.

S'abonner à un sujet

Les applications client peuvent s'abonner à tout sujet existant ou créer un nouveau sujet. Lorsqu'une application client s'abonne à un nouveau nom de sujet, un nouveau sujet portant ce nom est créé dans FCM et tout client peut ensuite s'y abonner.

Pour vous abonner à une rubrique, utilisez la méthode `subscribeToTopic()` spécifiant le nom de la rubrique:

```
FirebaseMessaging.getInstance().subscribeToTopic("myTopic");
```

Lire Firebase Cloud Messaging en ligne: https://riptutorial.com/fr/android/topic/8826/firebase-cloud-messaging

Chapitre 120: Firebase Realtime DataBase

Remarques

Autres sujets connexes:

- Firebase

Examples

Gestionnaire d'événements Firebase Realtime DataBase

Première initialisation de la base de données Firebase:

```
FirebaseDatabase database = FirebaseDatabase.getInstance();
```

Écrivez dans votre base de données:

```
// Write a message to the database
FirebaseDatabase database = FirebaseDatabase.getInstance();
DatabaseReference myRef = database.getReference("message");

myRef.setValue("Hello, World!");
```

Lire depuis votre base de données:

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

Récupérer des données sur des événements Android:

```
ChildEventListener childEventListener = new ChildEventListener() {
    @Override
    public void onChildAdded(DataSnapshot dataSnapshot, String previousChildName) {
        Log.d(TAG, "onChildAdded:" + dataSnapshot.getKey());
```

```
    }

    @Override
    public void onChildChanged(DataSnapshot dataSnapshot, String previousChildName) {
        Log.d(TAG, "onChildChanged:" + dataSnapshot.getKey());
    }

    @Override
    public void onChildRemoved(DataSnapshot dataSnapshot) {
        Log.d(TAG, "onChildRemoved:" + dataSnapshot.getKey());

    }

    @Override
    public void onChildMoved(DataSnapshot dataSnapshot, String previousChildName) {
        Log.d(TAG, "onChildMoved:" + dataSnapshot.getKey());

    }

    @Override
    public void onCancelled(DatabaseError databaseError) {
        Log.w(TAG, "postComments:onCancelled", databaseError.toException());
        Toast.makeText(mContext, "Failed to load comments.",
                Toast.LENGTH_SHORT).show();
    }
};
ref.addChildEventListener(childEventListener);
```

Installation rapide

1. Complétez la partie Installation et configuration pour connecter votre application à Firebase. Cela créera le projet dans Firebase.

2. Ajoutez la dépendance pour la base de données Firebase Realtime à votre fichier `build.gradle` niveau du `build.gradle` :

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

3. Configurer les règles de base de données Firebase

Vous êtes maintenant prêt à travailler avec la base de données Realtime sous Android.

Par exemple, vous écrivez un message `Hello World` dans la base de données sous la clé de `message` .

```
// Write a message to the database
FirebaseDatabase database = FirebaseDatabase.getInstance();
DatabaseReference myRef = database.getReference("message");

myRef.setValue("Hello, World!");
```

Concevoir et comprendre comment récupérer des données en temps réel à partir de la base de données Firebase

Cet exemple suppose que vous avez déjà configuré une base de données Firebase Realtime. Si vous êtes un démarreur, alors s'il vous plaît informez-vous ici sur la façon d'ajouter Firebase à votre projet Android.

Tout d'abord, ajoutez la dépendance de la base de données Firebase au fichier *build.gradle au* niveau de l'application:

```
compile 'com.google.firebase:firebase-database:9.4.0'
```

Maintenant, laissez-nous créer une application de chat qui stocke les données dans la base de données Firebase.

Étape 1: Créer une classe nommée Chat

Il suffit de créer une classe avec certaines variables de base requises pour le chat:

```
public class Chat{
    public String name, message;
}
```

Étape 2: créer des données JSON

Pour envoyer / récupérer des données vers / depuis la base de données Firebase, vous devez utiliser JSON. Supposons que certaines discussions sont déjà stockées au niveau racine dans la base de données. Les données de ces chats peuvent ressembler à ceci:

```
[
    {
        "name":"John Doe",
        "message":"My first Message"
    },
    {
        "name":"John Doe",
        "message":"Second Message"
    },
    {
        "name":"John Doe",
        "message":"Third Message"
    }
]
```

Étape 3: Ajouter les auditeurs

Il existe trois types d'auditeurs. Dans l'exemple suivant, nous allons utiliser le `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();
    }
});
```

Étape 4: Ajouter des données à la base de données

Créez simplement un objet de classe Chat et ajoutez les valeurs comme suit:

```
Chat chat=new Chat();
chat.name="John Doe";
chat.message="First message from android";
```

Maintenant, obtenez une référence au nœud de chats comme lors de la session de récupération:

```
DatabaseReference chatDb = FirebaseDatabase.getInstance().getReference().child("chats");
```

Avant de commencer à ajouter des données, gardez à l'esprit que vous avez besoin d'une référence de plus, car un nœud de discussion a plusieurs autres nœuds et l'ajout d'un nouveau chat implique l'ajout d'un nouveau nœud contenant les détails de la discussion. Nous pouvons générer un nom nouveau et unique du noeud à l'aide de la fonction `push()` sur l'objet `DatabaseReference` , qui renverra une autre `DatabaseReference` , qui à son tour pointe sur un noeud nouvellement formé pour insérer les données de discussion.

Exemple

```
// The parameter is the chat object that was newly created a few lines above.
chatDb.push().setValue(chat);
```

La fonction `setValue()` s'assurera que toutes les fonctions `onDataChanged` de l'application sont appelées (y compris le même périphérique), qui se trouve être l'écouteur attaché du nœud "chats".

Dénormalisation: Structure de base de données plate

La dénormalisation et une structure de base de données plate sont nécessaires pour télécharger efficacement des appels séparés. Avec la structure suivante, il est également possible de maintenir des relations bidirectionnelles. L'inconvénient de cette approche est que vous devez toujours mettre à jour les données à plusieurs endroits.

Par exemple, imaginez une application qui permet à l'utilisateur de stocker des messages pour lui-même (mémos).

Structure de base de données à plat souhaitée:

```
|--database
  |-- memos
     |-- memokey1
       |-- title: "Title"
       |-- content: "Message"
     |-- memokey2
       |-- title: "Important Title"
       |-- content: "Important Message"
  |-- users
     |-- userKey1
       |-- name: "John Doe"
       |-- memos
         |-- memokey1 : true //The values here don't matter, we only need the keys.
```

```
         |-- memokey2 : true
     |-- userKey2
       |-- name: "Max Doe"
```

La classe de mémo utilisée

```
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;
    }
}
```

Récupérer les mémos d'un utilisateur

```
//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) { }
    }
```

Créer un mémo

```
//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);
```

Après le push ou la base de données ressemble à ceci:

```
|--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"
```

Comprendre la base de données JSON Firebase

Avant de nous salir le code, j'estime qu'il est nécessaire de comprendre comment les données sont stockées dans Firebase. Contrairement aux bases de données relationnelles, Firebase stocke les données au format JSON. Considérez chaque ligne d'une base de données relationnelle comme un objet JSON (qui est essentiellement une paire clé-valeur non ordonnée). Ainsi, le nom de la colonne devient clé et la valeur stockée dans cette colonne pour une ligne particulière est la valeur. De cette manière, la ligne entière est représentée comme un objet JSON et une liste de ceux-ci représente une table de base de données entière. L'avantage immédiat que je vois pour cette modification du schéma devient une opération beaucoup moins chère par rapport aux anciens SGBDR. Il est plus facile d'ajouter quelques attributs supplémentaires à un JSON que de modifier une structure de table.

Voici un exemple de JSON pour montrer comment les données sont stockées dans la base de données:

```
{
 "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
  }
```

Cela montre clairement comment les données que nous stockions dans des bases de données relationnelles peuvent être stockées au format JSON. Ensuite, voyons comment lire ces données sur les appareils Android.

Récupération de données à partir de Firebase

Je suppose que vous savez déjà comment ajouter des dépendances progressives à la base de feu dans le studio Android. Si vous ne suivez pas simplement le guide d' ici . Ajoutez votre application dans la console firebase, synchronisez votre studio Android après avoir ajouté des dépendances. Toutes les dépendances ne sont pas nécessaires, mais uniquement la base de données firebase et l'authentification firebase.

Maintenant que nous savons comment les données sont stockées et comment ajouter des dépendances progressives, voyons comment utiliser le SDK Android Firebase importé pour récupérer des données.

créer une référence de base de données firebase

```
DatabaseReference userDBRef = FirebaseDatabase.getInstance().getReference();
// above statement point to base tree
userDBRef = DatabaseReference.getInstance().getReference().child("user_base")
// points to user_base table JSON (see previous section)
```

à partir de là, vous pouvez chaîner plusieurs appels de méthode child () pour pointer vers les données qui vous intéressent. Par exemple, si des données sont stockées comme décrit dans la section précédente et que vous souhaitez indiquer l'utilisateur Bruce Wayne, vous pouvez utiliser:

```
DatabaseReference bruceWayneRef = userDBRef.child("371298");
// 371298 is key of bruce wayne user in JSON structure (previous section)
```

Ou passez simplement la référence entière à l'objet JSON:

```
DatabaseReference bruceWayneRef = DatabaseReference.getInstance().getReference()
     .child("user_base/371298");
// deeply nested data can also be referenced this way, just put the fully
// qualified path in pattern shown in above code "blah/blah1/blah1-2/blah1-2-3..."
```

Maintenant que nous avons la référence des données que nous voulons récupérer, nous pouvons utiliser des écouteurs pour récupérer des données dans des applications Android. Contrairement aux appels traditionnels où vous lancez des appels d'API REST en utilisant retrofit ou volley, un simple écouteur de rappel est nécessaire pour obtenir les données. Firebase sdk appelle les méthodes de rappel et vous avez terminé.

Il existe essentiellement deux types d'écouteurs que vous pouvez attacher, l'un est ValueEventListener et l'autre est ChildEventListener (décrit dans la section suivante). Pour tout

changement de données sous le nœud auquel nous avons des références et des écouteurs ajoutés, les écouteurs d'événement de valeur renvoient la structure JSON complète et l'écouteur d'événement enfant retourne un enfant spécifique où la modification s'est produite. Les deux sont utiles à leur manière. Pour extraire les données de Firebase, nous pouvons ajouter un ou plusieurs écouteurs à une référence de base de données firebase (listez userDBRef que nous avons créé précédemment).

Voici un exemple de code (code explicatif après 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
    });
```

Avez-vous remarqué le type de classe passé? DataSnapshot peut convertir les données JSON dans nos POJO définis, en utilisant simplement le bon type de classe.

Si votre cas d'utilisation ne nécessite pas l'intégralité des données (dans notre cas la table user_base) chaque fois qu'un petit changement survient ou que vous souhaitez ***récupérer les données une seule fois*** , vous pouvez utiliser la méthode ***addListenerForSingleValueEvent ()*** de la référence de base de données. Cela déclenche le rappel une seule fois.

```
userDBRef.addListenerForSingleValueEvent(new ValueEventListener() {
    @Override
    public void onDataChange(DataSnapshot dataSnapshot) {
        // Do something
    }

    @Override
    public void onCancelled(DatabaseError databaseError) {
        // Do something about the error
    });
```

Les exemples ci-dessus vous donneront la valeur du noeud JSON. Pour obtenir la clé, appelez simplement:

```
String myKey = dataSnapshot.getKey();
```

Écoute des mises à jour d'enfants

Prenez un cas d'utilisation, comme une application de chat ou une application de liste d'épicerie collaborative (qui nécessite essentiellement une liste d'objets à synchroniser entre les utilisateurs). Si vous utilisez une base de données firebase et ajoutez un écouteur d'événement de valeur au nœud parent ou au nœud parent de liste d'épicerie, vous finirez avec toute la structure de

discussion depuis le début du chat (chaque fois qu'un nœud de discussion est ajouté). c.-à-d. que quelqu'un dit bonjour). Que nous ne voulons pas faire, ce qui nous intéresse, c'est seulement le nouveau nœud ou seulement l'ancien nœud qui a été supprimé ou modifié, les autres ne doivent pas être retournés.

Dans ce cas, nous pouvons utiliser ChildEvenListener . Sans plus tarder, voici un exemple de code (voir sections prev pour un exemple de données JSON):

```
userDBRef.addChildEventListener(new ChildEventListener() {
    @Override
    public void onChildAdded(DataSnapshot dataSnapshot, String s) {
    }

    @Override
    public void onChildChanged(DataSnapshot dataSnapshot, String s) {
    }

    @Override
    public void onChildRemoved(DataSnapshot dataSnapshot) {
    }

    @Override
    public void onChildMoved(DataSnapshot dataSnapshot, String s) {
        //If not dealing with ordered data forget about this
    }

    @Override
    public void onCancelled(DatabaseError databaseError) {
    });
```

Les noms de méthode sont explicites. Comme vous pouvez le voir à chaque fois qu'un nouvel utilisateur est ajouté ou qu'une propriété d'un utilisateur existant est modifiée ou qu'un utilisateur est supprimé ou supprimé, la méthode de rappel appropriée est appelée avec les données pertinentes. Donc, si vous conservez l'interface utilisateur actualisée, par exemple pour une application de chat, obtenez le fichier JSON de l'analyse onChildAdded () dans POJO et placez-le dans votre interface utilisateur. Rappelez-vous juste de supprimer votre auditeur lorsque l'utilisateur quitte l'écran.

onChildChanged () donne la valeur enfant entière avec les propriétés modifiées (nouvelles).

onChiledRemoved () renvoie le nœud enfant supprimé.

Récupération de données avec pagination

Lorsque vous avez une énorme base de données JSON, l'ajout d'un écouteur d'événement de valeur n'a pas de sens. Il retournera l'énorme JSON et l'analyse prendra beaucoup de temps. Dans de tels cas, nous pouvons utiliser la pagination et récupérer une partie des données et les afficher ou les traiter. Un peu comme le chargement paresseux ou comme chercher d'anciennes discussions lorsque l'utilisateur clique sur un ancien chat. Dans ce cas, Query peut être utilisé.

Prenons notre ancien exemple dans les sections précédentes. La base d'utilisateurs contient 3 utilisateurs, si le nombre d'utilisateurs est supérieur à trois cent mille et que vous souhaitez

récupérer la liste d'utilisateurs par lots de 50:

```
// class level
final int limit = 50;
int start = 0;

// event level
Query userListQuery = userDBRef.orderByChild("email").limitToFirst(limit)
        .startAt(start)
userListQuery.addValueEventListener(new ValueEventListener() {
    @Override
        public void onDataChange(DataSnapshot dataSnapshot) {
        // Do something
        start += (limit+1);
    }

    @Override
    public void onCancelled(DatabaseError databaseError) {
        // Do something about the error
    });
```

Ici, la valeur ou les événements enfants peuvent être ajoutés et écoutés. Appelez à nouveau la requête pour récupérer les 50 suivants. **Assurez-vous d'ajouter la méthode orderByChild ()** , cela ne fonctionnera pas sans cela. Firebase doit connaître l'ordre dans lequel vous paginez.

Lire Firebase Realtime DataBase en ligne: https://riptutorial.com/fr/android/topic/5511/firebase-realtime-database

Chapitre 121: FloatingActionButton

Introduction

Le bouton d'action flottant est utilisé par défaut pour un type spécial d'action promue, qu'il anime sur l'écran en tant que matériau en expansion. L'icône à l'intérieur peut être animée, et FAB peut également se déplacer différemment des autres éléments de l'interface utilisateur en raison de leur importance relative. Un bouton d'action flottant représente l'action principale dans une application qui peut simplement déclencher une action ou naviguer quelque part.

Paramètres

Paramètre	Détail
`android.support.design:elevation`	Valeur d'élévation pour le FAB. Peut être une référence à une autre ressource, sous la forme "@ [+] [package:] type / name" ou un attribut de thème sous la forme "? [Package:] type / name".
`android.support.design:fabSize`	Taille pour le FAB.
`android.support.design:rippleColor`	Couleur d'ondulation pour le FAB.
`android.support.design:useCompatPadding`	Activer le remplissage de compat.

Remarques

Les boutons d'action flottants sont utilisés pour un type particulier d'action promue. Ils se distinguent par une icône encerclée flottant au-dessus de l'interface utilisateur et ont des comportements de mouvement particuliers liés au morphing, au lancement et au point d'ancrage de transfert.

Un seul bouton d'action flottant est recommandé par écran pour représenter l'action la plus courante.

Avant d'utiliser le `FloatingActionButton` vous devez ajouter la dépendance de bibliothèque du support de conception dans le fichier `build.gradle` :

```
dependencies {
    compile 'com.android.support:design:25.1.0'
}
```

Documentation officielle:

https://developer.android.com/reference/android/support/design/widget/FloatingActionButton.html

Spécifications de conception matérielle:

https://material.google.com/components/buttons-floating-action-button.html

Examples

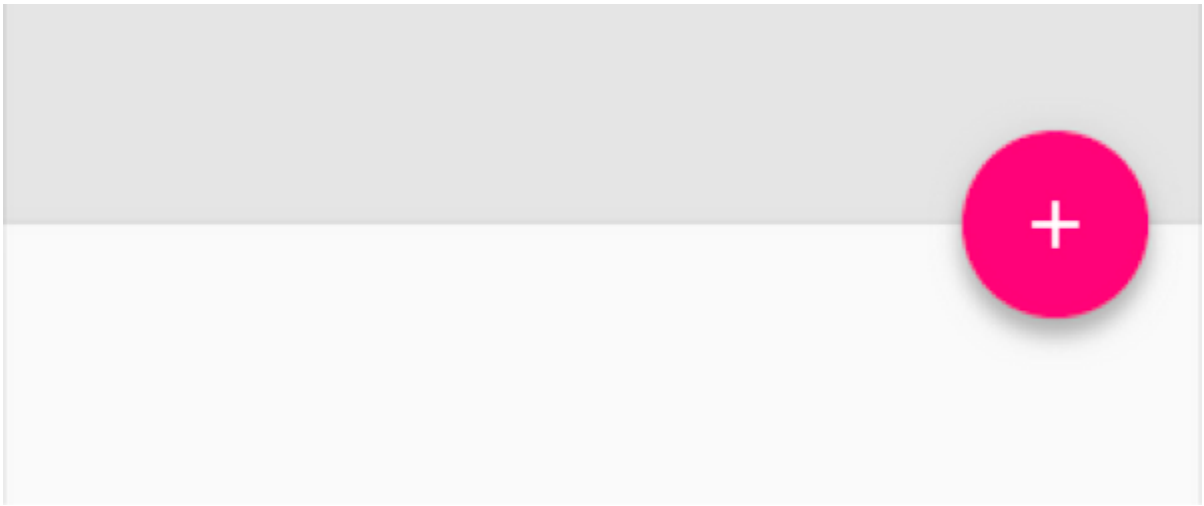
Comment ajouter le FAB à la mise en page

Pour utiliser un FloatingActionButton, ajoutez simplement la dépendance dans le fichier `build.gradle` comme décrit dans la section Remarques.

Puis ajoutez à la mise en page:

```
<android.support.design.widget.FloatingActionButton
       android:id="@+id/fab"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:layout_gravity="bottom|end"
       android:layout_margin="@dimen/fab_margin"
       android:src="@drawable/my_icon" />
```

Un exemple:



Couleur

La couleur d'arrière-plan de cette vue correspond par défaut au colorAccent de votre thème.

Dans l'image ci-dessus, si le `src` ne pointe que sur l'icône + (par défaut 24x24 dp), pour obtenir la *couleur d'arrière - plan* du cercle complet, vous pouvez utiliser
`app:backgroundTint="@color/your_colour"`

Si vous souhaitez changer la couleur dans le code, vous pouvez utiliser,

```
myFab.setBackgroundTintList(ColorStateList.valueOf(your color in int));
```

Si vous voulez changer la couleur de FAB dans l'utilisation de l'état pressé

```
mFab.setRippleColor(your color in int);
```

Positionnement

Il est recommandé de placer un minimum de 16 dB à partir du bord sur le mobile et de 24 dB minimum sur la tablette / le bureau.

Remarque : une fois que vous avez défini un src sauf pour couvrir toute la zone de `FloatingActionButton` assurez-vous de disposer de la bonne taille pour obtenir le meilleur résultat.

La taille de cercle par défaut est de 56 x 56dp

Taille de cercle mini: 40 x 40dp

Si vous ne voulez changer que l'icône Intérieur, utilisez une icône 24 x 24dp pour la taille par défaut.

Afficher et masquer FloatingActionButton sur Swipe

Pour afficher et masquer un `FloatingActionButton` avec l'animation par défaut, il suffit d'appeler les méthodes `show()` et `hide()` . Il est recommandé de conserver un `FloatingActionButton` dans la disposition des activités au lieu de le placer dans un fragment, cela permet aux animations par défaut de fonctionner lors de l'affichage et du masquage.

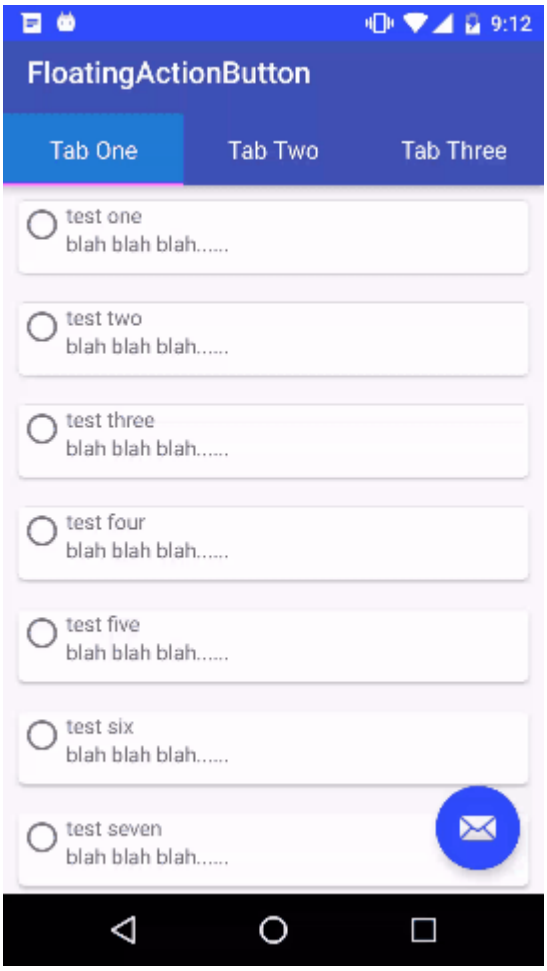
Voici un exemple avec un `ViewPager` :

- Trois onglets
- Afficher `FloatingActionButton` pour le premier et le troisième onglet
- Masquer le `FloatingActionButton` au milieu de l'onglet

```
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();
                }
            }
        });

    }
}
```

Résultat:



Afficher et masquer FloatingActionButton sur le défilement

À partir de la version 22.2.1 de Support Library, il est possible d'afficher et de masquer un FloatingActionButton contre un comportement de défilement à l'aide d'une sous-classe `FloatingActionButton.Behavior` qui tire parti des méthodes `show()` et `hide()` .

Notez que cela ne fonctionne qu'avec un CoordinatorLayout associé à des vues internes prenant en charge le défilement imbriqué, telles que RecyclerView et NestedScrollView .

Cette classe `ScrollAwareFABBehavior` provient des Guides Android sur Codepath (cc-wiki avec attribution requise)

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

Dans le `app:layout_behavior` XML de présentation FloatingActionButton, spécifiez l' `app:layout_behavior` avec le nom de classe qualifié complet de `ScrollAwareFABBehavior` :

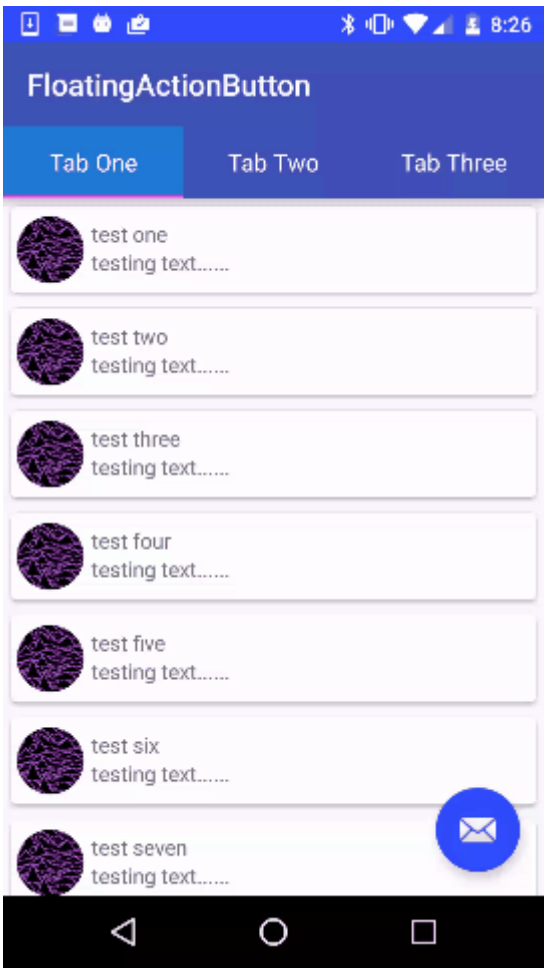
```
app:layout_behavior="com.example.app.ScrollAwareFABBehavior"
```

Par exemple avec cette mise en page:

```
<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>
```

Voici le résultat:



Définition du comportement de FloatingActionButton

Vous pouvez définir le comportement du FAB en XML.

Par exemple:

```
<android.support.design.widget.FloatingActionButton
   app:layout_behavior=".MyBehavior" />
```

Ou vous pouvez définir par programmation en utilisant:

```
CoordinatorLayout.LayoutParams p = (CoordinatorLayout.LayoutParams) fab.getLayoutParams();
p.setBehavior(xxxx);
fab.setLayoutParams(p);
```

Lire FloatingActionButton en ligne: https://riptutorial.com/fr/android/topic/2979/floatingactionbutton

Chapitre 122: Formatage des numéros de téléphone avec motif.

Introduction

Cet exemple vous montre comment formater des numéros de téléphone avec un motif

Vous aurez besoin de la bibliothèque suivante dans votre dossier.

compile 'com.googlecode.libphonenumber: libphonenumber: 7.2.2'

Examples

Patterns + 1 (786) 1234 5678

Étant donné un numéro de téléphone normalisé tel que +178612345678, nous obtiendrons un numéro formaté avec le motif fourni.

```
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;
}
```

Lire Formatage des numéros de téléphone avec motif. en ligne: https://riptutorial.com/fr/android/topic/9083/formatage-des-numeros-de-telephone-avec-motif-

Chapitre 123: Fournisseur de contenu

Remarques

Les fournisseurs de contenu gèrent l'accès à un ensemble structuré de données. Ils encapsulent les données et fournissent des mécanismes pour définir la sécurité des données. Les fournisseurs de contenu sont l'interface standard qui connecte les données dans un processus avec du code exécuté dans un autre processus.

Lorsque vous souhaitez accéder aux données d'un fournisseur de contenu, vous utilisez l'objet `ContentResolver` dans le `Context` votre application pour communiquer avec le fournisseur en tant que client. L'objet `ContentResolver` communique avec l'objet fournisseur, une instance d'une classe qui implémente `ContentProvider` . L'objet fournisseur reçoit les demandes de données des clients, effectue l'action demandée et renvoie les résultats.

Vous n'avez pas besoin de développer votre propre fournisseur si vous n'avez pas l'intention de partager vos données avec d'autres applications. Cependant, vous avez besoin de votre propre fournisseur pour fournir des suggestions de recherche personnalisées dans votre propre application. Vous avez également besoin de votre propre fournisseur si vous souhaitez copier et coller des données ou des fichiers complexes depuis votre application vers d'autres applications.

Android lui-même comprend des fournisseurs de contenu qui gèrent des données telles que des données audio, vidéo, des images et des informations de contact personnelles. Vous pouvez voir certains d'entre eux répertoriés dans la documentation de référence du package `android.provider` . Avec certaines restrictions, ces fournisseurs sont accessibles à toute application Android.

Examples

Implémentation d'une classe de fournisseur de contenu de base

1) Créez une classe de contrat

Une classe de contrat définit des constantes qui aident les applications à utiliser les URI de contenu, les noms de colonne, les actions d'intention et les autres fonctionnalités d'un fournisseur de contenu. Les classes de contrat ne sont pas incluses automatiquement avec un fournisseur; le développeur du fournisseur doit les définir et les mettre à la disposition des autres développeurs.

Un fournisseur a généralement une autorité unique, qui sert de nom interne à Android. Pour éviter les conflits avec d'autres fournisseurs, utilisez une autorité de contenu unique. Étant donné que cette recommandation est également vraie pour les noms de package Android, vous pouvez définir votre autorité de fournisseur comme une extension du nom du package contenant le fournisseur. Par exemple, si votre nom de package Android est `com.example.appname` , vous devez donner à votre fournisseur l'autorité `com.example.appname.provider` .

```
public class MyContract {
    public static final String CONTENT_AUTHORITY = "com.example.myApp";
```

```
    public static final String PATH_DATATABLE = "dataTable";
    public static final String TABLE_NAME = "dataTable";
}
```

Un URI de contenu est un URI qui identifie des données dans un fournisseur. Les URI de contenu incluent le nom symbolique du fournisseur complet (son autorité) et un nom qui pointe vers une table ou un fichier (un chemin). La partie facultative id pointe sur une ligne individuelle dans une table. Chaque méthode d'accès aux données de ContentProvider a un URI de contenu comme argument; Cela vous permet de déterminer la table, la ligne ou le fichier auquel accéder. Définissez-les dans la classe de contrat.

```
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) Créer la classe d'assistance

Une classe d'assistance gère la création de bases de données et la gestion des versions.

```
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) Créer une classe qui étend la classe ContentProvider

```
public class MyProvider extends ContentProvider {

public DatabaseHelper dbHelper;

public static final UriMatcher matcher = buildUriMatcher();
public static final int DATA_TABLE = 100;
public static final int DATA_TABLE_DATE = 101;
```

Un UriMatcher mappe une autorité et un chemin d'accès à une valeur entière. La méthode `match()` renvoie une valeur entière unique pour un URI (il peut s'agir d'un nombre quelconque, à condition qu'il soit unique). Une instruction de commutateur choisit d'interroger la table entière et d'interroger un seul enregistrement. Notre UriMatcher renvoie 100 si l'URI est l'URI du contenu de la table et 101 si l'URI pointe sur une ligne spécifique de cette table. Vous pouvez utiliser le caractère générique `#` pour correspondre à n'importe quel nombre et `*` pour correspondre à n'importe quelle chaîne.

```
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;
}
```

IMPORTANT: la commande d' `addURI()` importante! Le UriMatcher regardera dans l'ordre séquentiel du premier ajouté à la dernière. Comme les caractères génériques tels que `#` et `*` sont gourmands, vous devez vous assurer que vous avez correctement commandé vos URI. Par exemple:

```
uriMatcher.addURI(CONTENT_AUTHORITY, "/example", 1);
uriMatcher.addURI(CONTENT_AUTHORITY, "/*",       2);
```

est le bon ordre, car le matcher cherchera d'abord `/example` avant de recourir à la correspondance `/*` . Si ces appels de méthode ont été inversés et que vous avez appelé `uriMatcher.match("/example")` , le UriMatcher cessera de rechercher des correspondances une fois qu'il rencontrera le chemin `/*` et renverra le mauvais résultat!

Vous devrez alors remplacer ces fonctions:

onCreate () : Initialise votre fournisseur. Le système Android appelle cette méthode immédiatement après la création de votre fournisseur. Notez que votre fournisseur n'est pas créé jusqu'à ce qu'un objet ContentResolver tente d'y accéder.

```
@Override
public boolean onCreate() {
    dbhelper = new DatabaseHelper(getContext());
    return true;
}
```

getType () : Retourne le type MIME correspondant à un URI de contenu

```
@Override
public String getType(Uri uri) {
    final int match = matcher.match(uri);
    switch (match) {
        case DATA_TABLE:
            return ContentResolver.CURSOR_DIR_BASE_TYPE + "/" + MyContract.CONTENT_AUTHORITY +
"/" + MyContract.PATH_DATATABLE;
        case DATA_TABLE_DATE:
            return ContentResolver.ANY_CURSOR_ITEM_TYPE + "/" + MyContract.CONTENT_AUTHORITY +
```

```
"/" + MyContract.PATH_DATATABLE;
        default:
            throw new UnsupportedOperationException("Unknown Uri: " + uri);

    }
}
```

query () : récupère les données de votre fournisseur. Utilisez les arguments pour sélectionner la table à interroger, les lignes et les colonnes à renvoyer et l'ordre de tri du résultat. Renvoie les données en tant qu'objet Cursor.

```
@Override
public Cursor query(Uri uri, String[] projection, String selection, String[] selectionArgs,
String sortOrder) {
    Cursor retCursor = dbHelper.getReadableDatabase().query(
        MyContract.TABLE_NAME, projection, selection, selectionArgs, null, null, sortOrder);
    retCursor.setNotificationUri(getContext().getContentResolver(), uri);
    return retCursor;
}
```

Insérez une nouvelle ligne dans votre fournisseur. Utilisez les arguments pour sélectionner la table de destination et obtenir les valeurs de colonne à utiliser. Renvoie un URI de contenu pour la ligne nouvellement insérée.

```
@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 () : supprime les lignes de votre fournisseur. Utilisez les arguments pour sélectionner la table et les lignes à supprimer. Renvoie le nombre de lignes supprimées.

```
@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 () : Mettez à jour les lignes existantes dans votre fournisseur. Utilisez les arguments pour sélectionner la table et les lignes à mettre à jour et pour obtenir les nouvelles valeurs de colonne. Renvoie le nombre de lignes mises à jour.

```
@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) Mettre à jour le fichier manifeste

```
<provider
        android:authorities="com.example.myApp"
        android:name=".DatabaseProvider"/>
```

Lire Fournisseur de contenu en ligne: https://riptutorial.com/fr/android/topic/3075/fournisseur-de-contenu

Chapitre 124: Fragments

Introduction

Introduction sur les fragments et leur mécanisme d'intercommunication

Syntaxe

- void onActivityCreated (Bundle savedInstanceState) // Appelée lorsque l'activité du fragment a été créée et la hiérarchie de vue de ce fragment instanciée.
- void onActivityResult (int requestCode, int resultCode, Intent data) // Reçoit le résultat d'un précédent appel à startActivityForResult (Intent, int).
- void onAttach (activité d'activité) // Cette méthode est obsolète au niveau 23 de l'API. Utilisez plutôt onAttach (Context).
- void onAttach (Contexte contextuel) // Appelé lorsqu'un fragment est attaché pour la première fois à son contexte.
- void onAttachFragment (Fragment childFragment) // Appelé lorsqu'un fragment est attaché en tant qu'enfant de ce fragment.
- void onConfigurationChanged (Configuration newConfig) // Appelé par le système lorsque la configuration du périphérique change pendant que votre composant est en cours d'exécution.
- void onCreate (Bundle savedInstanceState) // Appelé pour faire la création initiale d'un fragment.
- Voir onCreateView (inflater LayoutInflater, conteneur ViewGroup, Bundle savedInstanceState) // Appelé pour que le fragment instancie sa vue d'interface utilisateur.
- void onDestroy () // Appelé lorsque le fragment n'est plus utilisé.
- void onDestroyView () // Appelé lorsque la vue précédemment créée par onCreateView (LayoutInflater, ViewGroup, Bundle) a été détachée du fragment.
- void onDetach () // Appelé lorsque le fragment n'est plus attaché à son activité.
- Annulez onInflate (Activité, Attributs AttributeSet, Bundle savedInstanceState) // Cette méthode est obsolète au niveau 23 de l'API. Utilisez plutôt onInflate (Context, AttributeSet, Bundle).
- Annulation de onInflate (Contexte contextuel, AttributeSet attrs, Bundle savedInstanceState) // Appelée lorsqu'un fragment est en cours de création dans le cadre d'une inflation de la disposition de la vue, généralement lors de la définition de la vue du contenu d'une activité.

- void onPause () // Appelé lorsque le fragment n'est plus repris.
- void onResume () // Appelé lorsque le fragment est visible par l'utilisateur et en cours d'exécution.
- void onSaveInstanceState (Bundle outState) // Appelé pour demander au fragment de sauvegarder son état dynamique actuel, il peut ensuite être reconstruit dans une nouvelle instance de son processus est redémarré.
- void onStart () // Appelé lorsque le fragment est visible par l'utilisateur.
- void onStop () // Appelé lorsque le fragment n'est plus démarré.
- void onViewStateRestored (Bundle savedInstanceState) // Appelé lorsque tout l'état enregistré a été restauré dans la hiérarchie de vue du fragment.

Remarques

Un fragment représente un comportement ou une partie de l'interface utilisateur dans une activité. Vous pouvez combiner plusieurs fragments dans une même activité pour créer une interface utilisateur à plusieurs volets et réutiliser un fragment dans plusieurs activités. Vous pouvez considérer un fragment comme une section modulaire d'une activité, qui possède son propre cycle de vie, reçoit ses propres événements d'entrée et que vous pouvez ajouter ou supprimer pendant l'exécution de l'activité (un peu comme une "sous-activité"). réutiliser dans différentes activités).

Constructeur

Chaque fragment doit avoir un **constructeur vide** , de sorte qu'il peut être instancié lors de la restauration de l'état de son activité. Il est fortement recommandé que les sous-classes n'aient pas d'autres constructeurs avec des paramètres, puisque ces constructeurs ne seront pas appelés lorsque le fragment sera ré-instancié; à la place, les arguments peuvent être fournis par l'appelant avec setArguments (Bundle) et récupérés plus tard par Fragment with getArguments ().

Examples

Le pattern newInstance ()

Bien qu'il soit possible de créer un constructeur de fragments avec des paramètres, Android appelle en interne le constructeur sans argument lors de la recréation de fragments (par exemple, s'ils sont restaurés après avoir été tués pour des raisons propres à Android). Pour cette raison, il est déconseillé de faire appel à un constructeur doté de paramètres.

Pour vous assurer que vos arguments de fragment attendus sont toujours présents, vous pouvez utiliser une `newInstance()` statique `newInstance()` pour créer le fragment et placer les paramètres souhaités dans un ensemble qui sera disponible lors de la création d'une nouvelle instance.

```
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);
    }
  }
}
```

Maintenant, dans l'activité:

```
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();
```

Ce modèle est une bonne pratique pour garantir que tous les arguments nécessaires seront transmis aux fragments lors de la création. Notez que lorsque le système détruit le fragment et le recrée plus tard, il restaure automatiquement son état - mais vous devez lui fournir une `onSaveInstanceState(Bundle)` .

Navigation entre des fragments à l'aide d'un backstack et d'un motif de tissu statique

Tout d'abord, nous devons ajouter notre premier `Fragment` au début, nous devrions le faire dans la méthode `onCreate()` de notre activité:

```
if (null == savedInstanceState) {
    getSupportFragmentManager().beginTransaction()
      .addToBackStack("fragmentA")
      .replace(R.id.container, FragmentA.newInstance(), "fragmentA")
      .commit();
}
```

Ensuite, nous devons gérer notre backstack. La manière la plus simple consiste à utiliser une fonction ajoutée dans notre activité et utilisée pour toutes les FragmentTransactions.

```
public void replaceFragment(Fragment fragment, String tag) {
    //Get current fragment placed in container
    Fragment currentFragment = getSupportFragmentManager().findFragmentById(R.id.container);

    //Prevent adding same fragment on top
    if (currentFragment.getClass() == fragment.getClass()) {
        return;
    }

    //If fragment is already on stack, we can pop back stack to prevent stack infinite growth
    if (getSupportFragmentManager().findFragmentByTag(tag) != null) {
        getSupportFragmentManager().popBackStack(tag,
FragmentManager.POP_BACK_STACK_INCLUSIVE);
    }

    //Otherwise, just replace fragment
    getSupportFragmentManager()
            .beginTransaction()
            .addToBackStack(tag)
            .replace(R.id.container, fragment, tag)
            .commit();
}
```

Enfin, nous devrions remplacer `onBackPressed()` pour quitter l'application en revenant du dernier fragment disponible dans le backstack.

```
@Override
public void onBackPressed() {
    int fragmentsInStack = getSupportFragmentManager().getBackStackEntryCount();
    if (fragmentsInStack > 1) { // If we have more than one fragment, pop back stack
        getSupportFragmentManager().popBackStack();
    } else if (fragmentsInStack == 1) { // Finish activity, if only one fragment left, to
prevent leaving empty screen
        finish();
    } else {
        super.onBackPressed();
    }
}
```

Exécution en activité:

```
replaceFragment(FragmentB.newInstance(), "fragmentB");
```

Exécution en dehors de l'activité (en supposant que `MainActivity` est notre activité):

```
((MainActivity) getActivity()).replaceFragment(FragmentB.newInstance(), "fragmentB");
```

Passer des données d'Activité à Fragment en utilisant Bundle

Tous les fragments doivent avoir un constructeur vide (c.-à-d. Une méthode constructeur sans arguments d'entrée). Par conséquent, pour transmettre vos données au fragment en cours de création, vous devez utiliser la méthode `setArguments()` . Cette méthode obtient un ensemble dans lequel vous stockez vos données et stocke le bundle dans les arguments. Par la suite, cet ensemble peut être récupéré dans les `onCreate()` et `onCreateView()` du fragment.

Activité:

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

Envoi d'événements à une activité avec une interface de rappel

Si vous devez envoyer des événements de fragment en activité, l'une des solutions possibles consiste à définir l'interface de rappel et à exiger que l'activité hôte l'implémente.

Exemple

Envoyer un rappel à une activité lorsque le bouton du fragment est cliqué

Tout d'abord, définissez l'interface de rappel:

```
public interface SampleCallback {
    void onButtonClicked();
}
```

L'étape suivante consiste à assigner ce rappel dans fragment:

```
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;
    }
}
```

Et enfin, implémentez le rappel dans l'activité:

```
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
    }
}
```

Animer la transition entre les fragments

Pour animer la transition entre les fragments ou pour animer le processus d'affichage ou de masquage d'un fragment, utilisez le `FragmentManager` pour créer une `FragmentTransaction` .

Pour une seule `FragmentTransaction` , il existe deux manières différentes d'effectuer des animations: vous pouvez utiliser une animation standard ou fournir vos propres animations personnalisées.

Les animations standard sont spécifiées en appelant `FragmentTransaction.setTransition(int transit)` et en utilisant l'une des constantes prédéfinies disponibles dans la classe `FragmentTransaction` . Au moment de la rédaction, ces constantes sont les suivantes:

```
FragmentTransaction.TRANSIT_NONE
FragmentTransaction.TRANSIT_FRAGMENT_OPEN
FragmentTransaction.TRANSIT_FRAGMENT_CLOSE
FragmentTransaction.TRANSIT_FRAGMENT_FADE
```

La transaction complète pourrait ressembler à ceci:

```
getSupportFragmentManager()
    .beginTransaction()
    .setTransition(FragmentTransaction.TRANSIT_FRAGMENT_FADE)
    .replace(R.id.contents, new MyFragment(), "MyFragmentTag")
    .commit();
```

Les animations personnalisées sont spécifiées en appelant soit `FragmentTransaction.setCustomAnimations(int enter, int exit)` ou `FragmentTransaction.setCustomAnimations(int enter, int exit, int popEnter, int popExit)` .

Les animations d' `enter` et de `exit` seront jouées pour les `FragmentTransaction` qui n'impliquent pas l'extinction de fragments de la pile arrière. Les animations `popEnter` et `popExit` seront lues lors de l'extraction d'un fragment de la pile arrière.

Le code suivant montre comment remplacer un fragment en faisant glisser un fragment et en glissant l'autre à sa place.

```
getSupportFragmentManager()
    .beginTransaction()
    .setCustomAnimations(R.anim.slide_in_left, R.anim.slide_out_right)
    .replace(R.id.contents, new MyFragment(), "MyFragmentTag")
    .commit();
```

Les définitions d'animation XML utiliseraient la balise `objectAnimator` . Un exemple de **slide_in_left.xml** pourrait ressembler à ceci:

```
<?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>
```

Communication entre fragments

Toutes les communications entre les fragments doivent passer par une activité. Les fragments **NE PEUVENT PAS** communiquer entre eux sans une activité.

Ressources additionnelles

- Comment implémenter OnFragmentInteractionListener
- Android | Communiquer avec d'autres fragments

Dans cet exemple, nous avons une `MainActivity` qui héberge deux fragments, `SenderFragment` et `ReceiverFragment` , respectivement pour envoyer et recevoir un `message` (une chaîne simple dans ce cas).

Un bouton dans `SenderFragment` lance le processus d'envoi du message. Un TextView dans le `ReceiverFragment` est mis à jour lorsque le message est reçu par celui-ci.

Voici l'extrait de code pour MainActivity avec des commentaires expliquant les lignes de code importantes:

```
// Our MainActivity implements the interface defined by the SenderFragment. This enables
// communication from the fragment to the activity
public class MainActivity extends AppCompatActivity implements
SenderFragment.SendMessageListener {

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

/**
 * This method is called when we click on the button in the SenderFragment
 * @param message The message sent by the SenderFragment
 */
@Override
public void onSendMessage(String message) {
    // Find our ReceiverFragment using the SupportFragmentManager and the fragment's id
    ReceiverFragment receiverFragment = (ReceiverFragment)
            getSupportFragmentManager().findFragmentById(R.id.fragment_receiver);

    // Make sure that such a fragment exists
    if (receiverFragment != null) {
        // Send this message to the ReceiverFragment by calling its public method
        receiverFragment.showMessage(message);
    }
}
}
```

Le fichier de disposition de `MainActivity` héberge deux fragments à l'intérieur d'un LinearLayout:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
```

```
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.naru.fragmentcommunication.MainActivity">

<fragment
    android:id="@+id/fragment_sender"
    android:name="com.naru.fragmentcommunication.SenderFragment"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="1"
    tools:layout="@layout/fragment_sender" />

<fragment
    android:id="@+id/fragment_receiver"
    android:name="com.naru.fragmentcommunication.ReceiverFragment"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="1"
    tools:layout="@layout/fragment_receiver" />
</LinearLayout>
```

`SenderFragment` expose une interface `SendMessageListener` qui aide le `MainActivity` savoir quand l' `SenderFragment` a cliqué sur le bouton dans `SenderFragment` .

Voici l'extrait de code pour le `SenderFragment` expliquant les lignes de code importantes:

```
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;
}
}
```

Le fichier de disposition pour le `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` est simple et expose une méthode publique simple pour mettre à jour son TextView. Lorsque `MainActivity` reçoit le message du `SenderFragment` il appelle cette méthode publique de `ReceiverFragment`

Voici l'extrait de code pour `ReceiverFragment` avec des commentaires expliquant les lignes de code importantes:

```
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);
}
}
```

Le fichier de mise en page pour `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>
```

Lire Fragments en ligne: https://riptutorial.com/fr/android/topic/1396/fragments

Chapitre 125: Fresque

Introduction

Fresco est un système puissant pour afficher des images dans les applications Android.

Dans Android 4.x et versions ultérieures, Fresco place les images dans une région particulière de la **mémoire Android** (appelée ashmem). Cela permet à votre application de s'exécuter plus rapidement et de faire face à la redoutable OutOfMemoryError beaucoup moins souvent.

Fresco prend également en charge la diffusion de fichiers JPEG.

Remarques

Comment configurer des dépendances dans le fichier build.gradle au niveau de l'application:

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

Plus d'informations peuvent être trouvées ici .

Examples

Premiers pas avec Fresco

Tout d'abord, ajoutez Fresco à votre `build.gradle` comme indiqué dans la section Remarques:

Si vous avez besoin de fonctionnalités supplémentaires, telles que le support GIF ou WebP animé, vous devez également ajouter les artefacts de fresques correspondants.

La fresque doit être initialisée. Vous ne devriez le faire qu'une seule fois, il est donc judicieux de placer l'initialisation dans votre `Application` . Un exemple pour cela serait:

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

Si vous souhaitez charger des images distantes depuis un serveur, votre application a besoin de l'autorisation internt. Ajoutez-le simplement à votre `AndroidManifest.xml` :

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

Ensuite, ajoutez un `SimpleDraweeView` à votre disposition XML. Fresco ne prend pas en charge `wrap_content` pour les dimensions de l'image, car vous pouvez avoir plusieurs images de dimensions différentes (image de substitution, image d'erreur, image réelle, ...).

Vous pouvez donc soit ajouter un `SimpleDraweeView` avec des dimensions fixes (ou `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" />
```

Ou fournissez un *ratio d'aspect* pour votre image:

```
<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" />
```

Enfin, vous pouvez définir votre URI d'image en Java:

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

C'est tout! Votre espace réservé doit pouvoir être dessiné jusqu'à ce que l'image réseau ait été récupérée.

Utiliser OkHttp 3 avec Fresco

Tout d'abord, en plus de la dépendance normale de Fresco Gradle, vous devez ajouter la dépendance OkHttp 3 à votre `build.gradle` :

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

Lorsque vous initialisez Fresco (généralement dans votre implémentation d' `Application` personnalisée), vous pouvez maintenant spécifier votre client OkHttp:

```
OkHttpClient okHttpClient = new OkHttpClient(); // Build on your own OkHttpClient.

Context context = ... // Your Application context.
ImagePipelineConfig config = OkHttpImagePipelineConfigFactory
        .newBuilder(context, okHttpClient)
        .build();
Fresco.initialize(context, config);
```

Streaming JPEG avec Fresco en utilisant DraweeController

Cet exemple suppose que vous avez déjà ajouté Fresco à votre application (voir cet exemple):

```
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.
```

Lire Fresque en ligne: https://riptutorial.com/fr/android/topic/5217/fresque

Chapitre 126: Fuite de fuite

Introduction

Leak Canary est une librairie Android et Java utilisée pour détecter les fuites dans l'application

Remarques

Vous pouvez voir l'exemple dans le lien ci-dessous

https://github.com/square/leakcanary

Examples

Implémenter une fuite canarienne dans une application Android

Dans votre *build.gradle,* vous devez ajouter les dépendances ci-dessous:

```
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'
```

Dans votre classe d' `Application` , vous devez ajouter le code ci-dessous dans votre `onCreate()` :

```
LeakCanary.install(this);
```

C'est tout ce que vous devez faire pour *LeakCanary* , il affichera automatiquement les notifications quand il y a une fuite dans votre build.

Lire Fuite de fuite en ligne: https://riptutorial.com/fr/android/topic/10041/fuite-de-fuite

Chapitre 127: Fuites de mémoire

Examples

Fuites de mémoire communes et comment les corriger

1. Fixez vos contextes:

Essayez d'utiliser le contexte approprié: par exemple, comme Toast peut être vu dans de nombreuses activités plutôt que dans une seule, utilisez `getApplicationContext()` pour les toasts, et comme les services peuvent continuer à fonctionner même si une activité est terminée, lancez un service avec:

```
Intent myService = new Intent(getApplicationContext(), MyService.class);
```

Utilisez ce tableau comme guide rapide pour savoir quel contexte est approprié:

	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	

Article original sur le contexte ici .

2. Référence statique au contexte

Une erreur de fuite de mémoire grave conserve une référence statique à `View` . Chaque `View` a une référence interne au `Context` . Ce qui signifie qu'une ancienne activité avec toute sa hiérarchie de vues ne sera pas récupérée avant la fin de l'application. Vous aurez votre application deux fois en mémoire lors de la rotation de l'écran.

Assurez-vous qu'il n'y a absolument aucune référence statique à View, Context ou à l'un de leurs descendants.

3. Vérifiez que vous terminez réellement vos services.

Par exemple, j'ai un intentService qui utilise l'API du service de localisation Google. Et j'ai oublié d'appeler `googleApiClient.disconnect();` :

```
//Disconnect from API onDestroy()
if (googleApiClient.isConnected()) {
    LocationServices.FusedLocationApi.removeLocationUpdates(googleApiClient,
GoogleLocationService.this);
    googleApiClient.disconnect();
}
```

4. Vérifiez l'utilisation des images et des bitmaps:

Si vous utilisez la **bibliothèque Picasso de** Square, j'ai constaté que je `.fit()` la mémoire en n'utilisant pas le `.fit()` , ce qui réduisait considérablement mon empreinte mémoire de 50 Mo en moyenne à moins de 19 Mo:

```
Picasso.with(ActivityExample.this)                    //Activity context
                .load(object.getImageUrl())
                .fit()                                 //This avoided the OutOfMemoryError
                .centerCrop()                          //makes image to not stretch
                .into(imageView);
```

5. Si vous utilisez des récepteurs de diffusion, désinscrivez-les.

6. Si vous utilisez `java.util.Observer` (modèle Observer):

Veillez à utiliser `deleteObserver(observer);`

Évitez les fuites d'activités avec AsyncTask

> **Un mot d'avertissement** : AsyncTask a beaucoup de pièges à part la fuite de mémoire décrite ici. Alors faites attention à cette API, ou évitez-la complètement si vous ne comprenez pas complètement les implications. Il existe de nombreuses alternatives (Thread, EventBus, RxAndroid, etc.).

Une erreur commune avec `AsyncTask` est de capturer une référence forte à l' `Activity` hôte (ou `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!
  }
}
```

Cela pose un problème car `AsyncTask` peut facilement survivre à l' `Activity` parente, par exemple si une modification de configuration se produit pendant l'exécution de la tâche.

La bonne façon de faire est de faire de votre tâche une classe `static` , qui ne capture pas le parent, et contenant une référence faible à l' `Activity` de l'hôte:

```
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
    }
  }
}
```

Rappel anonyme dans les activités

Chaque fois que vous créez une classe anonyme, elle conserve une référence implicite à sa classe parente. Donc, quand vous écrivez:

```
public class LeakyActivity extends Activity
{

...

    foo.registerCallback(new BarCallback()
    {
        @Override
        public void onBar()
        {
            // do something
        }
    });
}
```

En fait, vous envoyez une référence à votre instance LeakyActivity à foo. Lorsque l'utilisateur navigue hors de votre LeakyActivity, cette référence peut empêcher l'instance LeakyActivity d'être récupérée. Il s'agit d'une fuite sérieuse car les activités ont une référence à la hiérarchie de vue entière et sont donc des objets plutôt volumineux en mémoire.

Comment éviter cette fuite:

Vous pouvez bien sûr éviter d'utiliser des rappels anonymes dans les activités. Vous pouvez également annuler l'enregistrement de tous vos rappels en fonction du cycle de vie de l'activité. ainsi:

```
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);
    }
}
```

Contexte d'activité dans les classes statiques

Souvent, vous voudrez envelopper certaines classes d'Android dans des classes utilitaires plus faciles à utiliser. Ces classes d'utilitaires nécessitent souvent un contexte pour accéder au système d'exploitation Android ou aux ressources de vos applications. Un exemple courant de ceci est un wrapper pour la classe SharedPreferences. Pour accéder aux préférences partagées d'Androids, il faut écrire:

```
context.getSharedPreferences(prefsName, mode);
```

Et on peut être tenté de créer la classe suivante:

```
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);
    }
}
```

maintenant, si vous appelez `init()` avec votre contexte d'activité, le LeakySharedPrefsWrapper conservera une référence à votre activité, l'empêchant d'être nettoyé.

Comment éviter:

Lorsque vous appelez des fonctions d'assistance statique, vous pouvez envoyer le contexte d'application à l'aide de `context.getApplicationContext();`

Lors de la création de fonctions d'assistance statique, vous pouvez extraire le contexte d'application du contexte qui vous est fourni (Calling getApplicationContext () sur le contexte d'application renvoie le contexte d'application). Donc, le correctif à notre wrapper est simple:

```
public static void init(Context context)
{
    sContext = context.getApplicationContext();
}
```

Si le contexte d'application n'est pas adapté à votre cas d'utilisation, vous pouvez inclure un paramètre Context dans chaque fonction d'utilitaire, vous devez éviter de conserver des références à ces paramètres de contexte. Dans ce cas, la solution devrait ressembler à ceci:

```
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);
}
```

Détecter les fuites de mémoire avec la bibliothèque LeakCanary

LeakCanary est une bibliothèque Java Open Source pour détecter les fuites de mémoire dans vos versions de débogage.

Ajoutez simplement les dépendances dans le `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'
 }
```

Ensuite, dans votre classe d' `Application` :

```
public class ExampleApplication extends Application {

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

    if (LeakCanary.isInAnalyzerProcess(this)) {
      // This process is dedicated to LeakCanary for heap analysis.
      // You should not init your app in this process.
      return;
    }

    LeakCanary.install(this);
  }
}
```

Maintenant, LeakCanary affichera automatiquement une notification lorsqu'une fuite de mémoire d'activité est détectée dans votre version de **débogage** .

Remarque: le code de version ne contiendra aucune référence à LeakCanary autre que les deux classes vides qui existent dans la `leakcanary-android-no-op` .

Évitez les fuites d'activités avec les auditeurs

Si vous implémentez ou créez un écouteur dans une activité, faites toujours attention au cycle de vie de l'objet sur lequel le programme d'écoute est enregistré.

Considérons une application dans laquelle plusieurs activités / fragments différents sont intéressés lorsqu'un utilisateur est connecté ou déconnecté. Une façon de procéder serait d'avoir une instance singleton de `UserController` laquelle vous pouvez vous abonner pour être averti lorsque l'état de l'utilisateur change:

```
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();
    }
}
```

Ensuite, il y a deux activités, `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();
    }
}
```

Et `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();
    }
}
```

Ce qui se passe avec cet exemple, c'est que chaque fois que l'utilisateur se connecte puis se déconnecte, une instance de `MainActivity` fuit. La fuite se produit car il existe une référence à l'activité dans les `UserController#listeners` .

S'il vous plaît noter: Même si nous utilisons une classe interne anonyme en tant qu'auditeur, l'activité fuirait toujours:

```
...
this.userController.registerUserStateChangeListener(new UserController.StateListener() {
    @Override
    public void userLoggedIn() {
        showUserAccount();
    }

    @Override
    public void userLoggedOut() {
        finish();
    }
});
...
```

L'activité fuirait toujours, car la classe interne anonyme a une référence implicite à la classe externe (dans ce cas, l'activité). C'est pourquoi il est possible d'appeler des méthodes d'instance dans la classe externe à partir de la classe interne. En fait, les seules classes internes qui n'ont pas de référence à la classe externe sont **les** classes internes **statiques** .

En bref, toutes les instances de classes internes non statiques contiennent une référence implicite à l'instance de la classe externe qui les a créées.

Il existe deux approches principales pour résoudre ce problème, soit en ajoutant une méthode pour supprimer un écouteur des écouteurs `UserController#listeners` ou en utilisant une `WeakReference` pour contenir la référence des écouteurs.

Alternative 1: Supprimer les auditeurs

Commençons par créer une nouvelle méthode `removeUserStateChangeListener(StateListener listener)` :

```
public class UserController {
```

```
    ...

    public void registerUserStateChangeListener(StateListener listener) {
        listeners.add(listener);
    }

    public void removeUserStateChangeListener(StateListener listener) {
        listeners.remove(listener);
    }

    ...
}
```

`onDestroy` ensuite cette méthode dans la méthode `onDestroy` l'activité:

```
public class MainActivity extends Activity implements UserController.StateListener{
    ...

    @Override
    protected void onDestroy() {
        super.onDestroy();
        userController.removeUserStateChangeListener(this);
    }
}
```

Avec cette modification, les instances de `MainActivity` ne fuient plus lorsque l'utilisateur se connecte et se déconnecte. Cependant, si la documentation n'est pas claire, il est probable que le prochain développeur qui commence à utiliser `UserController` risque de ne plus pouvoir enregistrer l'auditeur lorsque l'activité est détruite, ce qui nous amène à la deuxième méthode pour éviter ces types de fuites.

Alternative 2: Utiliser des références faibles

Tout d'abord, commençons par expliquer ce qu'est une référence faible. Une référence faible, comme son nom l'indique, contient une référence faible à un objet. Par rapport à un champ d'instance normal, qui est une référence forte, une référence faible n'arrête pas le ramasse-miettes, GC, de supprimer les objets. Dans l'exemple ci-dessus, cela permettrait à `MainActivity` d'être récupéré après avoir été détruit si `UserController` utilisait `WeakReference` pour référencer les écouteurs.

En bref, une référence faible indique au GC que si personne d'autre ne fait référence à cet objet, retirez-le.

`WeakReference` le `UserController` pour utiliser une liste de `WeakReference` pour suivre ses écouteurs:

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

Avec cette modification, peu importe que les écouteurs soient supprimés ou non, car `UserController` ne contient aucune référence forte à aucun des écouteurs. Cependant, écrire ce code passe-partout à chaque fois est encombrant. `WeakCollection` une classe générique appelée `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);
        }
    }
}
```

Maintenant, laissez-nous réécrire `UserController` pour utiliser `WeakCollection<T>` place:

```
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();
        }
```

```
    }

    ...
}
```

Comme indiqué dans l'exemple de code ci-dessus, `WeakCollection<T>` supprime tout le code passe-`WeakReference` nécessaire pour utiliser `WeakReference` au lieu d'une liste normale. Pour couronner le tout: Si un appel à `UserController#removeUserStateChangeListener(StateListener)` est manquant, le listener et tous les objets qu'il référence ne vont pas fuir.

Évitez les fuites de mémoire avec la classe anonyme, le gestionnaire, la tâche de minuterie, le thread

Dans Android, chaque développeur utilise la `Anonymous Class` (Runnable) au moins une fois dans un projet. Toute `Anonymous Class` a une référence à son parent (activité). Si nous effectuons une tâche de longue durée, l'activité parent ne sera pas détruite tant que la tâche n'est pas terminée. L'exemple utilise le gestionnaire et la classe anonyme `Runnable` . La mémoire va fuir lorsque nous quitterons l'activité avant que le `Runnable` soit terminé.

```
  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...
```

Comment pouvons-nous le résoudre?

1. Ne faites pas fonctionner longtemps avec la `Anonymous Class` ou nous avons besoin d'une `Static class` pour cela et y passer `WeakReference` (comme l'activité, la vue ...). `Thread` est le même avec la `Anonymous Class` .
2. Annuler le `Handler` , `Timer` lorsque l'activité est détruite.

Lire Fuites de mémoire en ligne: https://riptutorial.com/fr/android/topic/2687/fuites-de-memoire

Chapitre 128: Genymotion pour Android

Introduction

Genymotion est un émulateur tiers rapide qui peut être utilisé à la place de l'émulateur Android par défaut. Dans certains cas, c'est aussi bon ou meilleur que de développer sur des appareils réels!

Examples

Installer Genymotion, la version gratuite

Étape 1 - Installation de `VirtualBox`

Téléchargez et installez VirtualBox selon votre système d'exploitation. , il faut faire tourner `Genymotion` .

Étape 2 - télécharger `Genymotion`

Accédez à la page de téléchargement de `Genymotion` et téléchargez `Genymotion` fonction de votre système d'exploitation.

Remarque: vous devrez créer un nouveau compte OU vous connecter avec votre compte.

Étape 3 - Installation de `Genymotion`

Si sous `Linux` reportez-vous à cette réponse pour installer et exécuter un fichier `.bin` .

Étape 4 - Installation des émulateurs de `Genymotion`

- exécuter `Genymotion`
- Appuyez sur le bouton Ajouter (dans la barre supérieure).
- Connectez-vous avec votre compte et vous pourrez parcourir les émulateurs disponibles.
- sélectionnez et installez ce dont vous avez besoin.

Étape 5 - Intégrer `genymotion` avec `Android Studio`

`Genymotion` , peut être intégré à `Android Studio` via un plugin, voici les étapes pour l'installer dans `Android Studio`

- aller dans Fichier / Paramètres (pour Windows et Linux) ou sur Android Studio / Préférences

(pour Mac OS X)
- Sélectionnez Plugins et cliquez sur Parcourir les référentiels.
- Cliquez avec le bouton droit sur Genymotion et cliquez sur Télécharger et installer.

Vous devriez maintenant pouvoir voir l'icône du plugin, voir cette image

Notez que vous souhaiterez peut-être afficher la barre d'outils en cliquant sur Affichage> Barre d'outils.

Étape 6 - `Genymotion` depuis `Android Studio`

- aller dans Fichier / Paramètres (pour Windows et Linux) ou sur Android Studio / Préférences (pour Mac OS X)
- allez dans Autres paramètres / Genymotion et ajoutez le chemin du dossier de `Genymotion's` et appliquez vos modifications.

Maintenant, vous devriez pouvoir lancer `Genymotion's` émulateur `Genymotion's` en appuyant sur l'icône du plug-in et en sélectionnant un émulateur installé, puis appuyez sur le bouton Démarrer!

Google framework sur Genymotion

Si les développeurs veulent tester Google Maps ou tout autre service Google tel que Gmail, Youtube, Google Drive, etc., ils doivent d'abord installer Google Framework sur Genymotion. Voici les étapes: -

4.4 Kitkat
5.0 Sucette
5.1 Sucette
6,0 guimauve
7.0 Nougat
7.1 Nougat (patch webview)

1. Télécharger depuis le lien ci-dessus
2. Il suffit de glisser-déposer le fichier zip téléchargé sur genymotion et de le redémarrer
3. Ajoutez un compte Google et téléchargez "Google Play Music" et Exécuter.

Référence:-
Question de débordement de pile sur ce sujet

Lire Genymotion pour Android en ligne: https://riptutorial.com/fr/android/topic/9245/genymotion-pour-android

Chapitre 129: Gestion des événements tactiles et animés

Introduction

Un résumé de quelques-uns des systèmes de manipulation tactile / mouvements de base de l'API Android.

Paramètres

Auditeur	Détails
onTouchListener	Gère les touches simples pour les boutons, les surfaces et plus encore
onTouchEvent	Un auditeur qui peut être trouvé dans les surfaces (par exemple, SurfaceView). N'a pas besoin d'être défini comme les autres écouteurs (par exemple, onTouchListener)
onLongTouch	Semblable à onTouch, mais écoute de longues pressions sur les boutons, les surfaces et plus encore.

Examples

Boutons

Les événements tactiles liés à un `Button` peuvent être vérifiés comme suit:

```
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;
    }
}
```

Surface

Gestionnaire d'événements tactiles pour les surfaces (par exemple, SurfaceView, GLSurfaceView et autres):

```
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;
    }
}
```

Ou alternativement (en surface):

```
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;
    }
}
```

Gestion du multitouch dans une surface

```
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.
    }
}
```

Lire Gestion des événements tactiles et animés en ligne:
https://riptutorial.com/fr/android/topic/9315/gestion-des-evenements-tactiles-et-animes

Chapitre 130: Gestionnaire de raccourcis

Examples

Raccourcis du lanceur dynamique

```
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));
```

Nous pouvons supprimer tous les raccourcis dynamiques facilement en appelant: -

```
 shortcutManager.removeAllDynamicShortcuts();
```

Nous pouvons mettre à jour les Dynamic Shorcuts existants en utilisant

```
shortcutManager.updateShortcuts(Arrays.asList(shortcut);
```

Notez que `setDynamicShortcuts(List)` est utilisé pour redéfinir la liste complète des raccourcis dynamiques, `addDynamicShortcuts(List)` est utilisé pour ajouter des raccourcis dynamiques à la liste existante des raccourcis dynamiques.

Lire Gestionnaire de raccourcis en ligne: https://riptutorial.com/fr/android/topic/7661/gestionnaire-de-raccourcis

Chapitre 131: Glisse

Introduction

**** AVERTISSEMENT Cette documentation n'est pas maintenue et souvent inexacte ****

La documentation officielle de Glide est une bien meilleure source:

Pour Glide v4, voir http://bumptech.github.io/glide/ . Pour Glide v3, voir https://github.com/bumptech/glide/wiki .

Remarques

Glide est une **infrastructure** rapide et efficace de gestion de médias open source et de chargement d'images pour Android qui intègre le décodage des supports, la mémoire et la mise en cache des disques, ainsi que la mise en pool des ressources dans une interface simple et facile à utiliser.

Glide prend en charge l'extraction, le décodage et l'affichage d'images fixes vidéo, d'images et de GIF animés. Glide comprend une API flexible qui permet aux développeurs de se connecter à presque toutes les piles du réseau.
Par défaut, Glide utilise une pile personnalisée basée sur HttpUrlConnection , mais inclut également des bibliothèques d'utilitaires connectées au projet Volley de Google ou à la bibliothèque OkHttp de Square .

L'objectif principal de Glide est de rendre le défilement de toutes les listes d'images aussi fluide et rapide que possible, mais Glide est également efficace dans presque tous les cas où vous devez extraire, redimensionner et afficher une image distante.

Le code source et la documentation supplémentaire sont disponibles sur GitHub: https://github.com/bumptech/glide

Examples

Ajouter Glide à votre projet

De la documentation officielle :

Avec 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'
}
```

Avec 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>
```

Selon votre configuration et utilisation de ProGuard (DexGuard), vous devrez peut-être également inclure les lignes suivantes dans votre proguard.cfg (voir le wiki de Glide pour plus d'informations):

```
-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
```

Chargement d'une image

ImageView

Pour charger une image à partir d'une URL, d'un Uri, d'un ID de ressource ou de tout autre modèle spécifié dans une `ImageView` :

```
ImageView imageView = (ImageView) findViewById(R.id.imageView);
String yourUrl = "http://www.yoururl.com/image.png";

Glide.with(context)
    .load(yourUrl)
    .into(imageView);
```

Pour Uris, remplacez votre `yourUrl` par votre Uri (`content://media/external/images/1`). Pour Drawables, remplacez votre `yourUrl` par votre identifiant de ressource (`R.drawable.image`).

RecyclerView et ListView

Dans ListView ou RecyclerView, vous pouvez utiliser exactement les mêmes lignes:

```
@Override
public void onBindViewHolder(RecyclerView.ViewHolder viewHolder, int position) {
    MyViewHolder myViewHolder = (MyViewHolder) viewHolder;
    String currentUrl = myUrls.get(position);

    Glide.with(context)
        .load(currentUrl)
        .into(myViewHolder.imageView);
}
```

Si vous ne voulez pas démarrer un chargement dans `onBindViewHolder` , assurez-vous que vous `clear()` tout `ImageView` Glide en cours de gestion avant de modifier `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);
    }
}
```

Transformation du cercle de glissement (Charger l'image dans une image circulaire)

Créez une image circulaire avec la glisse.

```
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();
    }
}
```

Usage:

```
Glide.with(context)
    .load(yourimageurl)
    .transform(new CircleTransform(context))
    .into(userImageView);
```

Transformations par défaut

Glide comprend deux transformations par défaut, le centre d'ajustement et le recadrage central.

Centre d'ajustement:

```
Glide.with(context)
    .load(yourUrl)
    .fitCenter()
    .into(yourView);
```

Le centre d'ajustement effectue la même transformation que ScaleType.FIT_CENTER d'Android.

Centre de récolte:

```
Glide.with(context)
    .load(yourUrl)
    .centerCrop()
    .into(yourView);
```

Centrage central effectue la même transformation que ScaleType.CENTER_CROP pour Android.

Pour plus d'informations, voir le wiki de Glide .

Image des coins arrondis avec la cible Glide personnalisée

Commencez par créer une classe utilitaire ou utilisez cette méthode en classe

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

Chargement de l'image:

```
Glide.with(context)
     .load(imageUrl)
     .asBitmap()
     .into(UIUtils.getRoundedImageTarget(context, imageView, radius));
```

Comme vous utilisez asBitmap (), les animations seront supprimées. Vous pouvez utiliser votre propre animation à cet endroit en utilisant la méthode animate ().

Exemple avec un fondu similaire à celui de l'animation par défaut

```
Glide.with(context)
     .load(imageUrl)
     .asBitmap()
     .animate(R.anim.abc_fade_in)
     .into(UIUtils.getRoundedImageTarget(context, imageView, radius));
```

Veuillez noter que cette animation est la ressource privée de la bibliothèque de support - il est déconseillé de l'utiliser car elle peut changer ou même être supprimée.

Notez que vous devez également avoir une bibliothèque de support pour utiliser RoundedBitmapDrawableFactory

Préchargement d'images

Pour précharger des images distantes et vous assurer que l'image n'est téléchargée qu'une seule fois:

```
Glide.with(context)
    .load(yourUrl)
    .diskCacheStrategy(DiskCacheStrategy.SOURCE)
    .preload();
```

Alors:

```
Glide.with(context)
    .load(yourUrl)
    .diskCacheStrategy(DiskCacheStrategy.SOURCE) // ALL works here too
    .into(imageView);
```

Pour précharger des images locales et vous assurer qu'une copie transformée se trouve dans le cache du disque (et peut-être dans le cache de la mémoire):

```
Glide.with(context)
    .load(yourFilePathOrUri)
    .fitCenter() // Or whatever transformation you want
    .preload(200, 200); // Or whatever width and height you want
```

Alors:

```
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);
```

Placeholder et gestion des erreurs

Si vous souhaitez ajouter un objet pouvant être dessiné pendant le chargement, vous pouvez ajouter un espace réservé:

```
Glide.with(context)
    .load(yourUrl)
    .placeholder(R.drawable.placeholder)
    .into(imageView);
```

Si vous souhaitez afficher un Drawable si le chargement échoue pour une raison quelconque:

```
Glide.with(context)
    .load(yourUrl)
    .error(R.drawable.error)
    .into(imageView);
```

Si vous voulez qu'un Drawable soit affiché si vous fournissez un modèle nul (URL, Uri, chemin de fichier, etc.):

```
Glide.with(context)
    .load(maybeNullUrl)
    .fallback(R.drawable.fallback)
    .into(imageView);
```

Charger l'image dans un ImageView circulaire sans transformations personnalisées

Créez un objet BitmapImageViewTarget personnalisé pour charger l'image dans:

```
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);
    }
}
```

Usage:

```
Glide
    .with(context)
    .load(yourimageidentifier)
    .asBitmap()
    .into(new CircularBitmapImageViewTarget(context, imageView));
```

Échec du chargement de l'image Glide de manipulation

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

Ici, à `SET_YOUR_DEFAULT_IMAGE` place, vous pouvez définir n'importe quel `Drawable` par défaut. Cette image sera affichée si le chargement de l'image a échoué.

Lire Glisse en ligne: https://riptutorial.com/fr/android/topic/1091/glisse

Chapitre 132: Glisser pour rafraîchir

Syntaxe

1. **setColorSchemeResources** définit les couleurs de l' **indicateur** SwipeToRefreshLayout
2. **setOnRefreshListener** définit ce qu'il faut faire lorsque la disposition est balayée
3. **app: layout_behavior = "@ string / appbar_scrolling_view_behavior"** Si vous avez une barre d'outils avec votre mise en page, ajoutez-la avec les scrollFlags dans la barre d'outils et la barre d'outils glissera vers le haut et glissera vers le haut.

Examples

Balayer pour actualiser avec RecyclerView

Pour ajouter une disposition **Balayer à rafraîchir** avec **RecyclerView,** ajoutez ce qui suit à votre fichier de mise en page Activité / Fragment:

```
<android.support.v4.widget.SwipeRefreshLayout
    android:id="@+id/refresh_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    app:layout_behavior="@string/appbar_scrolling_view_behavior">

    <android.support.v7.widget.RecyclerView
        android:id="@+id/recycler_view"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:scrollbars="vertical" />

</android.support.v4.widget.SwipeRefreshLayout>
```

Dans votre activité / fragment, ajoutez ce qui suit pour initialiser le **SwipeToRefreshLayout** :

```
    SwipeRefreshLayout mSwipeRefreshLayout = (SwipeRefreshLayout)
findViewById(R.id.refresh_layout);
    mSwipeRefreshLayout.setColorSchemeResources(R.color.green_bg,
            android.R.color.holo_green_light,
            android.R.color.holo_orange_light,
            android.R.color.holo_red_light);

    mSwipeRefreshLayout.setOnRefreshListener(new SwipeRefreshLayout.OnRefreshListener() {
        @Override
        public void onRefresh() {
            // Execute code when refresh layout swiped
        }
    });
```

Comment ajouter Swipe-to-Refresh à votre application

Assurez-vous que la dépendance suivante est ajoutée au fichier `build.gradle` votre application sous les dépendances:

```
compile 'com.android.support:support-core-ui:24.2.0'
```

Ajoutez ensuite le `SwipeRefreshLayout` dans votre mise en page:

```
<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>
```

Enfin, implémentez l'écouteur `SwipeRefreshLayout.OnRefreshListener` .

```
mSwipeRefreshLayout = (SwipeRefreshLayout) findViewById(R.id.swipe_refresh_layout);
mSwipeRefreshLayout.setOnRefreshListener(new OnRefreshListener() {
    @Override
    public void onRefresh() {
         // your code
    }
});
```

Lire Glisser pour rafraîchir en ligne: https://riptutorial.com/fr/android/topic/5241/glisser-pour-rafraichir

Chapitre 133: Google Maps API v2 pour Android

Paramètres

Paramètre	Détails
Google Map	`GoogleMap` est un objet reçu sur un `onMapReady()`
MarkerOptions	`MarkerOptions` est la classe de générateur d'un `Marker` et permet d'ajouter un marqueur à une carte.

Remarques

Exigences

1. Google Play Services SDK installé.
2. Un compte Google Console.
3. Une clé API Google Maps obtenue dans Google Console.

Examples

Activité Google Map par défaut

Ce code d'activité fournira des fonctionnalités de base pour inclure une carte Google Map utilisant un SupportMapFragment.

L'API Google Maps V2 comprend un tout nouveau moyen de charger des cartes.

Les activités doivent maintenant implémenter l'interface **OnMapReadyCallBack,** qui vient avec une substitution de la méthode **onMapReady ()** qui est à chaque fois exécuté , nous courons **SupportMapFragment. getMapAsync (OnMapReadyCallback)** ; et l'appel est terminé avec succès.

Les cartes utilisent des **marqueurs** , des **polygones** et des **lignes linéaires** pour afficher des informations interactives à l'utilisateur.

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));
    }
}
```

Notez que le code ci-dessus gonfle une mise en page, qui a un SupportMapFragment imbriqué dans la présentation du conteneur, défini avec un ID de `R.id.map` . Le fichier de disposition est illustré ci-dessous:

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>
```

Styles Google Map personnalisés

Style de carte

Google Maps est fourni avec un ensemble de styles différents à appliquer à l'aide de ce code:

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

Les différents styles de carte sont les suivants:

Ordinaire

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

Carte routière typique. Les routes, certaines caractéristiques créées par l'homme et d'importantes caractéristiques naturelles telles que les rivières sont représentées. Les étiquettes de route et de caractéristique sont également visibles.

Hybride

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

Données de photographie satellite avec cartes routières ajoutées. Les étiquettes de route et de caractéristique sont également visibles.

Satellite

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

Données de photographie par satellite. Les étiquettes de route et de caractéristique ne sont pas visibles.

Terrain

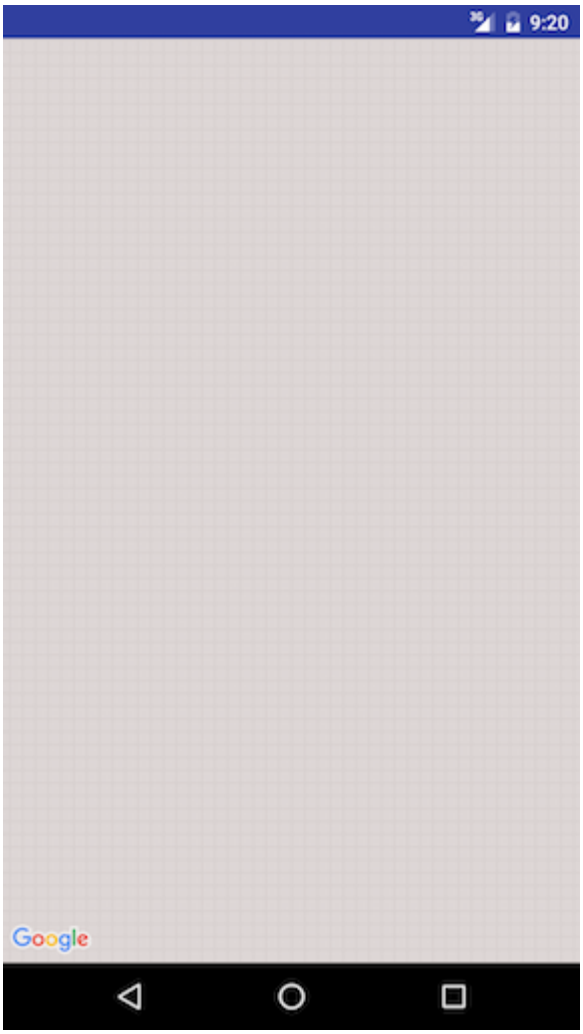
```
map.setMapType(GoogleMap.MAP_TYPE_TERRAIN);
```

Données topographiques. La carte comprend les couleurs, les lignes de contour et les étiquettes, ainsi que les ombres de perspective. Certaines routes et étiquettes sont également visibles.

Aucun

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

Pas de tuiles La carte sera rendue comme une grille vide sans tuiles chargées.



AUTRES OPTIONS DE STYLE

Cartes intérieures

À des niveaux de zoom élevés, la carte affichera les plans d'étage des espaces intérieurs. Celles-ci sont appelées cartes d'intérieur et ne sont affichées que pour les types de carte «normale» et «satellite».

pour activer ou désactiver les cartes intérieures, voici comment cela se passe:

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

Nous pouvons ajouter des styles personnalisés aux cartes.

Dans la méthode onMapReady, ajoutez l'extrait de code suivant

```
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);
    }
```

sous le dossier *res* , créez un nom de dossier *brut* et ajoutez le fichier styles json. Exemple de fichier style.json

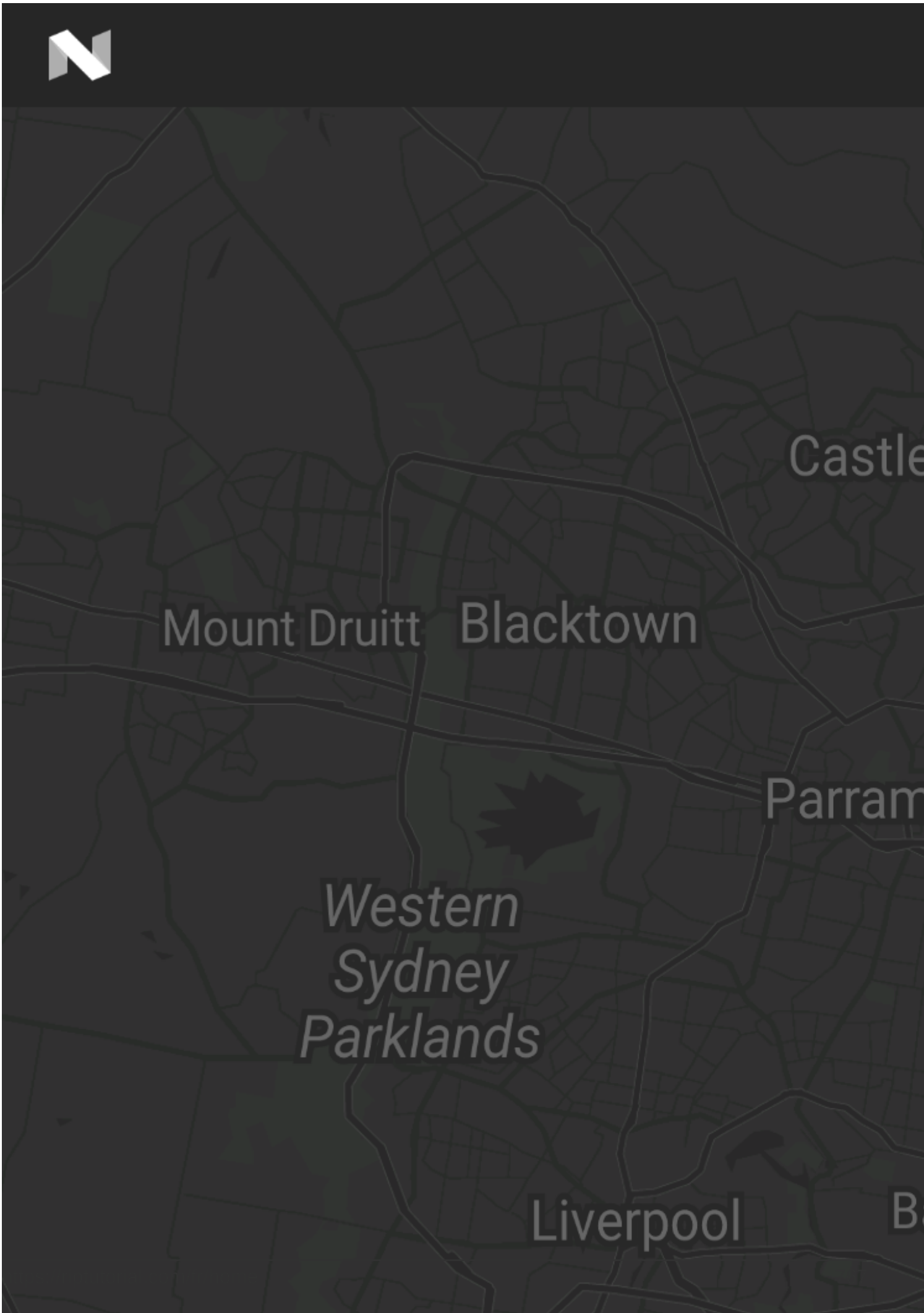
```
    [
  {
    "featureType": "all",
    "elementType": "geometry",
    "stylers": [
      {
        "color": "#242f3e"
      }
    ]
  },
  {
    "featureType": "all",
    "elementType": "labels.text.stroke",
    "stylers": [
      {
        "lightness": -80
      }
    ]
  },
  {
    "featureType":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

Pour générer des fichiers json styles, cliquez sur ce lien


Castle
Mount Druitt
Blacktown
Parram
Western
Sydney
Parklands
Liverpool

`MyLocation` , nous pouvons le faire de cette manière.

La classe de titulaire `MyLocation` :

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

Voici une méthode qui prendrait une liste d'objets `MyLocation` et placerait un marqueur pour chacun d'eux:

```
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));
 }
}
```

Remarque: Pour les besoins de cet exemple, `mMap` est une variable membre de classe de l'activité, dans laquelle nous l'avons affectée à la référence de carte reçue dans le `onMapReady()` .

MapView: incorporation d'un GoogleMap dans une mise en page existante

Il est possible de traiter un GoogleMap comme une vue Android si nous utilisons la classe MapView fournie. Son utilisation est très similaire à MapFragment.

Dans votre mise en page, utilisez MapView comme suit:

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

Votre activité doit implémenter l'interface OnMapReadyCallback pour fonctionner:

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

Afficher la position actuelle dans une carte Google

Voici une classe d'activité complète qui place un marqueur à l'emplacement actuel et déplace également la caméra vers la position actuelle.

Il se passe quelque chose dans l'ordre ici:

- Vérifier l'autorisation de localisation
- Une fois l'autorisation d'emplacement accordée, appelez `setMyLocationEnabled()` , créez GoogleApiClient et connectez-le.
- Une fois que GoogleApiClient est connecté, demandez des mises à jour de l'emplacement

```
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.
```

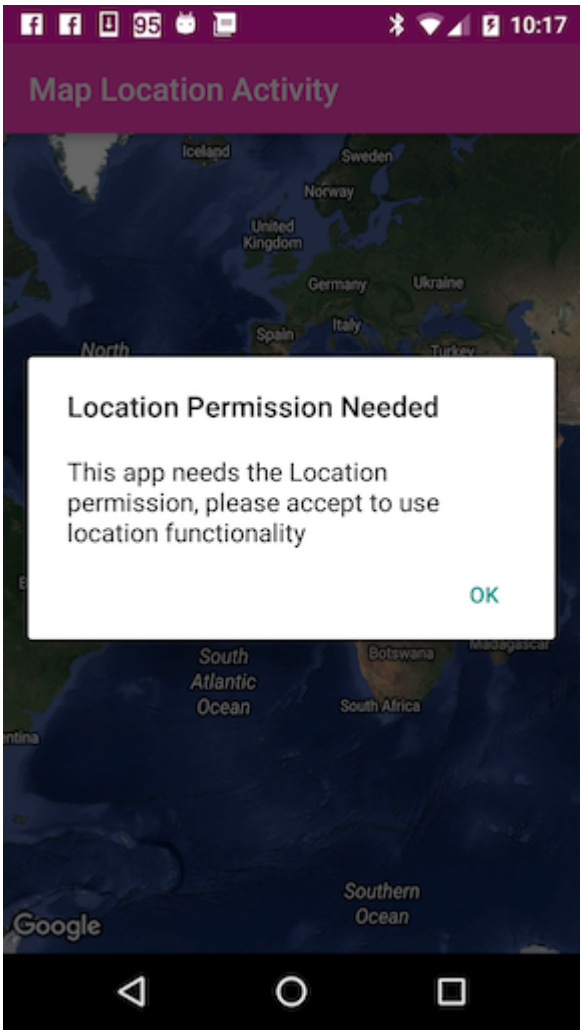
```
                if (grantResults.length > 0
                        && grantResults[0] == PackageManager.PERMISSION_GRANTED) {

                    // 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:

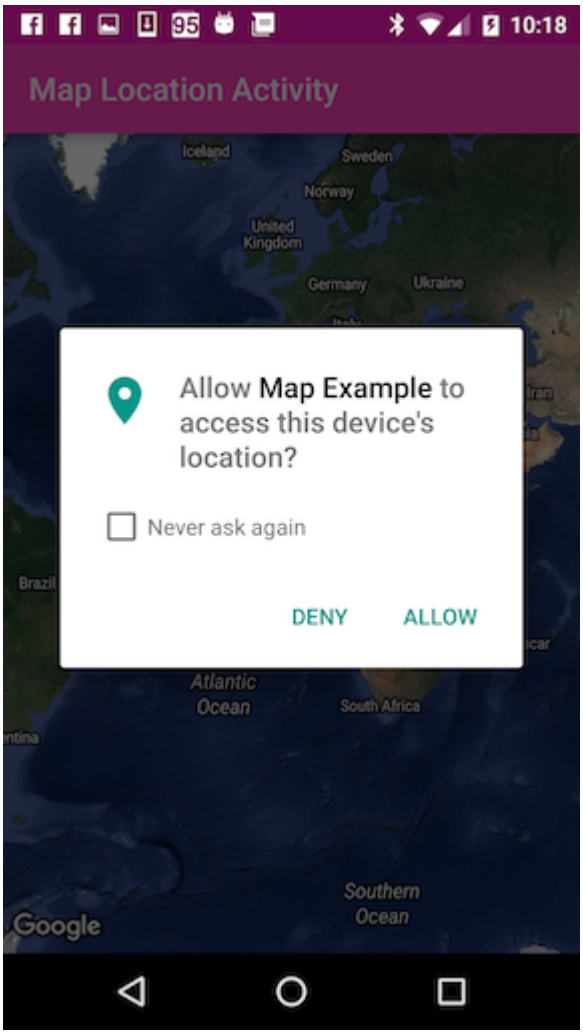
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent">

    <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>
```

Résultat:

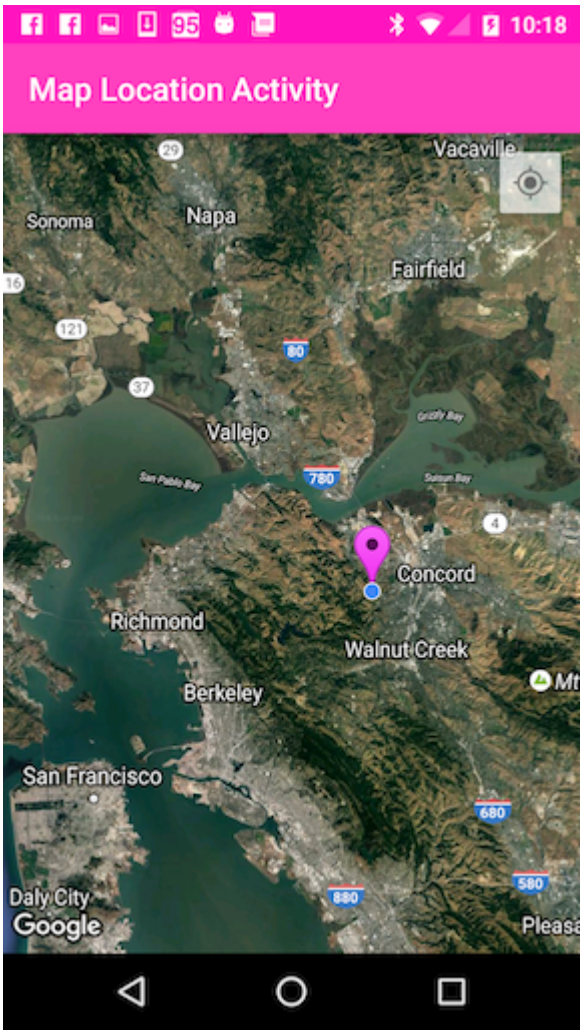
Affiche des explications si nécessaire sur Marshmallow et Nougat à l'aide d'un AlertDialog (ce cas se produit lorsque l'utilisateur a précédemment refusé une demande d'autorisation, ou lui a accordé l'autorisation et l'a ensuite révoqué dans les paramètres):



Invitez l'utilisateur à obtenir l'autorisation Location sur Marshmallow et Nougat en appelant `ActivityCompat.requestPermissions()` :



Déplacez la caméra vers l'emplacement actuel et placez le marqueur lorsque l'autorisation de localisation est accordée:



Obtention de l'empreinte SH1 de votre fichier de clés de certificat

Pour obtenir une clé API Google Maps pour votre certificat, vous devez fournir à la console API l'empreinte SH1 de votre fichier de clés de débogage / édition.

Vous pouvez obtenir le fichier de clés à l'aide du programme **keytool du JDK,** comme décrit ici dans les documents.

Une autre approche consiste à obtenir l'empreinte digitale par programmation en exécutant cet extrait de code avec votre application signée avec le certificat de débogage / édition et en imprimant le hachage dans le journal.

```
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());
}
```

Ne lancez pas Google Maps lorsque la carte est cliquée (mode lite)

Lorsqu'une carte Google est affichée en mode allégé, un clic sur une carte ouvre l'application Google Maps. Pour désactiver cette fonctionnalité, vous devez appeler `setClickable(false)` sur `MapView` , *par exemple* :

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

UISettings

En utilisant les `UISettings` , l'apparence de Google Map peut être modifiée.

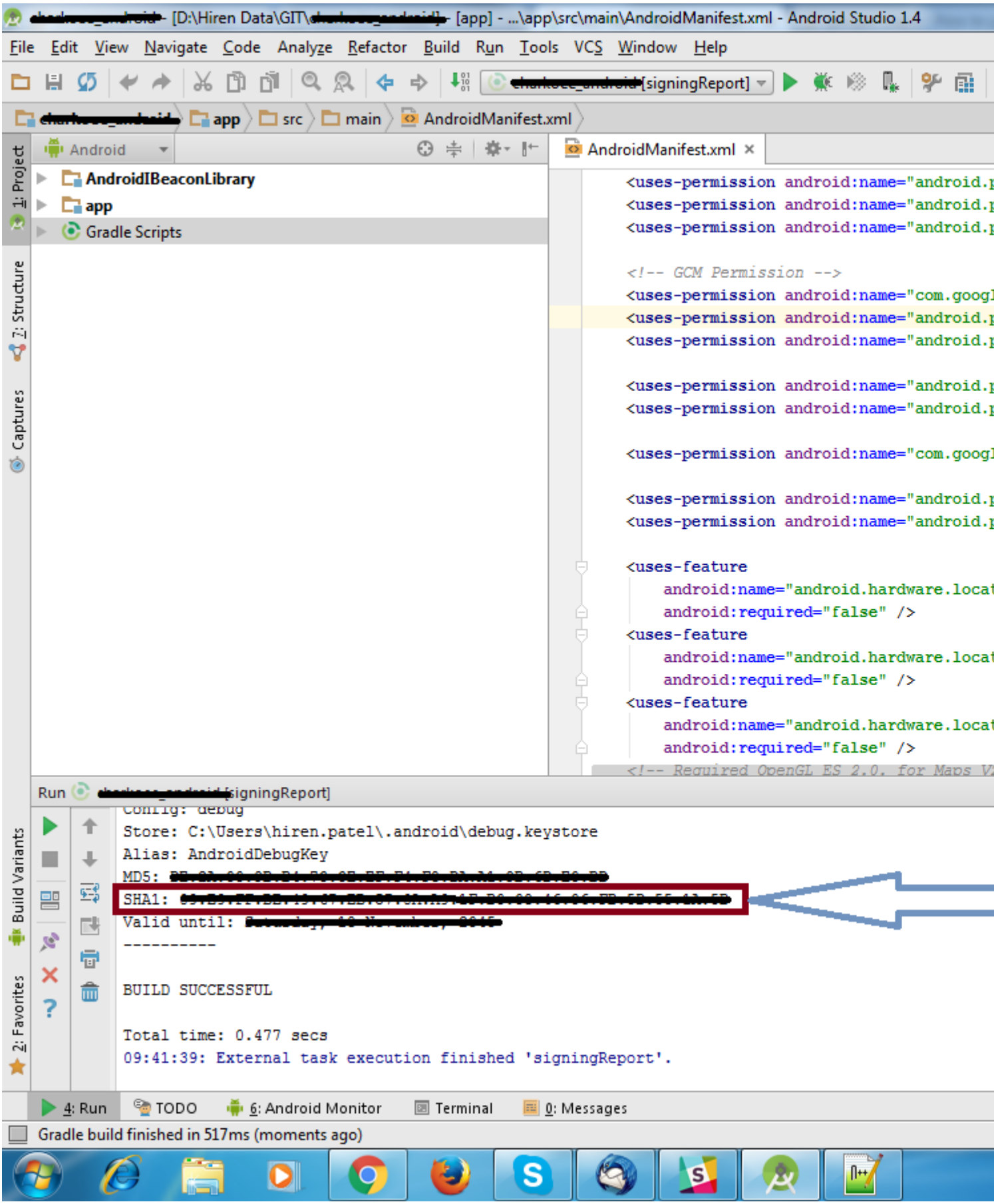
Voici un exemple de paramètres communs:

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

Résultat:

Obtenir l'empreinte SHA1 du débogage

1. Ouvrir Android Studio
2. Ouvrez votre projet
3. Cliquez sur Gradle (du panneau latéral droit, vous verrez **Barre Gradle**)
4. Cliquez sur Actualiser (Cliquez sur Actualiser dans **Gradle Bar** , vous verrez les scripts de **liste** de votre projet)
5. Cliquez sur votre projet (Votre formulaire Nom du projet **Liste** (racine))
6. Cliquez sur les tâches
7. Cliquez sur Android
8. Cliquez deux fois sur la signatureSignaler (vous obtiendrez **SHA1** et **MD5** dans la **barre d'** exécution)



InfoWindow Click Listener

Voici un exemple de la façon de définir une action différente pour chaque événement de clic

InfoWindow du marqueur.

Utilisez un HashMap dans lequel l'ID du marqueur est la clé et la valeur correspond à l'action correspondante à effectuer lorsque l'utilisateur clique sur InfoWindow.

Ensuite, utilisez un `OnInfoWindowClickListener` pour gérer l'événement d'un utilisateur qui clique sur InfoWindow et utilisez HashMap pour déterminer l'action à entreprendre.

Dans cet exemple simple, nous allons ouvrir une activité différente en fonction de laquelle l'utilisateur a cliqué sur InfoWindow.

Déclarez le HashMap comme variable d'instance de l'activité ou du fragment:

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

Ensuite, chaque fois que vous ajoutez un marqueur, effectuez une entrée dans HashMap avec l'ID de marqueur et l'action à effectuer lorsque l'utilisateur clique sur InfoWindow.

Par exemple, ajouter deux marqueurs et définir une action à entreprendre pour chacun:

```
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");
```

Dans InfoWindow, cliquez sur le port d'écoute, obtenez l'action de HashMap et ouvrez l'activité correspondante en fonction de l'action du marqueur:

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

Remarque Si le code est dans un fragment, remplacez MainActivity.this par getActivity ().

Changer le décalage

En modifiant les valeurs de *x* et *y de* mappoint, vous pouvez modifier la possibilité de décalage de google map. Par défaut, elle sera au centre de la vue de carte. Appelez ci-dessous la méthode où vous voulez le changer! Mieux vaut l'utiliser dans votre `onLocationChanged` comme `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

        }
```

Lire Google Maps API v2 pour Android en ligne: https://riptutorial.com/fr/android/topic/170/google-maps-api-v2-pour-android

Chapitre 134: Google Play Store

Examples

Ouvrez Google Play Store Listing pour votre application

L'extrait de code suivant montre comment ouvrir la liste Google Play Store de votre application en toute sécurité. Habituellement, vous voulez l'utiliser lorsque vous demandez à l'utilisateur de laisser un commentaire pour votre application.

```
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);
}
```

Remarque : le code ouvre le Google Play Store si l'application est installée. Sinon, il suffit d'ouvrir le navigateur Web.

Ouvrez Google Play Store avec la liste de toutes les applications de votre compte d'éditeur

Vous pouvez ajouter un bouton "Parcourir nos autres applications" dans votre application, répertoriant toutes vos applications (d'éditeur) dans l'application Google Play Store.

```
String urlApp = "market://search?q=pub:Google+Inc.";
String urlWeb = "http://play.google.com/store/search?q=pub:Google+Inc.";
try {
    Intent i = new Intent(Intent.ACTION_VIEW, Uri.parse(urlApp));
    setFlags(i);
    startActivity(i);
} catch (android.content.ActivityNotFoundException anfe) {
    Intent i = new Intent(Intent.ACTION_VIEW, Uri.parse(urlWeb)));
    setFlags(i);
```

```
    startActivity(i);
}


@SuppressWarnings("deprecation")
public void setFlags(Intent i) {
    i.addFlags(Intent.FLAG_ACTIVITY_NO_HISTORY);
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP)  {
        i.addFlags(Intent.FLAG_ACTIVITY_NEW_DOCUMENT);
    }
    else  {
        i.addFlags(Intent.FLAG_ACTIVITY_CLEAR_WHEN_TASK_RESET);
    }
}
```

Lire Google Play Store en ligne: https://riptutorial.com/fr/android/topic/10900/google-play-store

Chapitre 135: Gradle pour Android

Introduction

Gradle est un système de génération basé sur JVM qui permet aux développeurs d'écrire des scripts de haut niveau pouvant être utilisés pour automatiser le processus de compilation et de production d'applications. C'est un système flexible basé sur un plug-in, qui vous permet d'automatiser divers aspects du processus de construction. y compris la compilation et la signature d'un `.jar` , le téléchargement et la gestion des dépendances externes, l'injection de champs dans `AndroidManifest` ou l'utilisation de versions spécifiques du SDK.

Syntaxe

- `apply plugin` : Les plugins qui devraient normalement être utilisés juste `'com.android.application'` ou `'com.android.library'` .
- `android` : la configuration principale de votre application
    - `compileSdkVersion` : La version du SDK de compilation
    - `buildToolsVersion` : La version des outils de construction
    - `defaultConfig` : les paramètres par défaut qui peuvent être écrasés par les saveurs et les types de construction
        - `applicationId` : l'identifiant de l'application que vous utilisez, par exemple dans le PlayStore, est le même que le nom de votre paquet
        - `minSdkVersion` : la version minimale requise du SDK
        - `targetSdkVersion` : La version du SDK sur laquelle vous compilez (devrait toujours être la nouvelle)
        - `versionCode` : Le numéro de version interne qui doit être plus grand à chaque mise à jour
        - `versionName` : le numéro de version que l'utilisateur peut voir dans la page de détails de l'application
    - `buildTypes` : Voir ailleurs (TODO)
- `dependencies` : les `dependencies` maven ou locales de votre application
    - `compile` une seule dépendance
    - `testCompile` : une dépendance pour l'unité ou les tests d'intégration

Remarques

Voir également

- La page d'accueil officielle
- Comment configurer les builds de gradle
- Le plugin android pour gradle

- Android Gradle DSL

Gradle pour Android - Documentation étendue:

Il y a une autre balise où vous pouvez trouver plus de sujets et d'exemples sur l'utilisation de gradle dans Android.
http://www.riptutorial.com/topic/2092

Examples

Un fichier de base build.gradle

Voici un exemple de fichier `build.gradle` par défaut dans un module.

```
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 (langage spécifique au domaine)

Chaque bloc du fichier ci-dessus est appelé un `DSL` (langage spécifique au domaine).

Plugins

La première ligne, `apply plugin: 'com.android.application'` , applique le plug-in Android pour Gradle à la construction et rend le bloc `android {}` disponible pour déclarer les options de construction spécifiques à Android.

Pour une **application Android** :

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

Pour une **bibliothèque Android** :

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

Comprendre les DSL dans l'exemple ci-dessus

La deuxième partie, Le bloc `android {...}` , est la `DSL` Android qui contient des informations sur votre projet.

Par exemple, vous pouvez définir le `compileSdkVersion` qui spécifie le niveau d'API Android, qui doit être utilisé par Gradle pour compiler votre application.
Le sous-bloc `defaultConfig` contient les valeurs par défaut pour votre manifeste. Vous pouvez les `override` par des saveurs de produit .

Vous pouvez trouver plus d'informations dans ces exemples:

- DSL pour le module d'application
- Types de construction
- Saveurs du produit
- Paramètres de signature

Les dépendances

Le bloc de `dependencies` est défini en dehors du bloc `android {...}` : cela signifie qu'il n'est pas

défini par le plugin Android mais qu'il est standard.
Le bloc de `dependencies` spécifie les bibliothèques externes (généralement les bibliothèques Android, mais les bibliothèques Java sont également valides) que vous souhaitez inclure dans votre application. Gradle téléchargera automatiquement ces dépendances pour vous (s'il n'y a pas de copie locale disponible), il vous suffit d'ajouter des lignes de `compile` similaires lorsque vous souhaitez ajouter une autre bibliothèque.

Regardons l'une des lignes présentes ici:

```
compile 'com.android.support:design:25.3.1'
```

Cette ligne dit essentiellement

> Ajouter une dépendance sur la bibliothèque de conception du support Android à mon projet.

Gradle s'assurera que la bibliothèque est téléchargée et présente afin que vous puissiez l'utiliser dans votre application, et son code sera également inclus dans votre application.

Si vous êtes familier avec Maven, cette syntaxe est *GroupId* , deux points, *ArtifactId* , un autre deux-points, puis la version de la dépendance que vous souhaitez inclure, vous donnant un contrôle total sur le versionnage.

Bien qu'il soit possible de spécifier des versions d'artefacts en utilisant le signe plus (+), la meilleure pratique consiste à éviter de le faire; Cela peut entraîner des problèmes si la bibliothèque est mise à jour à votre insu à l'aide de modifications, ce qui pourrait entraîner des pannes dans votre application.

Vous pouvez ajouter différents types de dépendances:

- dépendances binaires locales
- dépendances des modules
- dépendances distantes

Une attention particulière devrait être consacrée aux dépendances à plat .

Vous pouvez trouver plus de détails dans cette rubrique.

Note sur le **-v7 dans *appcompat-v7***

```
compile 'com.android.support:appcompat-v7:25.3.1'
```

Cela signifie simplement que cette **bibliothèque** (`appcompat`) est compatible avec l'API Android de niveau 7 et ultérieur.

Note sur le **junit: junit: 4.12**

Ceci est la dépendance de test pour les tests unitaires.

Spécification des dépendances spécifiques aux différentes configurations de construction

Vous pouvez spécifier qu'une dépendance ne doit être utilisé pour une certaine configuration de construction ou vous pouvez définir des dépendances pour les types de construction ou les saveurs des produits (par exemple, le débogage, de test ou libération) en utilisant `debugCompile` , `testCompile` ou `releaseCompile` au lieu de l'habituel `compile` .

Ceci est utile pour garder les dépendances liées au test et au débogage hors de votre version, ce qui maintiendra votre version `APK` aussi mince que possible et vous assurera que les informations de débogage ne peuvent pas être utilisées pour obtenir des informations internes sur votre application.

---

signatureConfig

Le `signingConfig` vous permet de configurer votre Gradle pour inclure `keystore` informations et faire en sorte que l'APK construit en utilisant ces configurations sont signés et prêts à jouer la libération de magasin.

Ici vous pouvez trouver un sujet dédié .

Remarque : Il n'est pas recommandé de conserver les informations d'identification de signature dans votre fichier Gradle. Pour supprimer les configurations de signature, omettez simplement la partie `signingConfigs` .
Vous pouvez les spécifier de différentes manières:

- stocker dans un fichier externe
- les stocker dans la définition des variables d'environnement .

Voir cette rubrique pour plus de détails: Signer APK sans exposer le mot de passe du fichier de clés .

Vous trouverez d'autres informations sur Gradle pour Android dans le sujet dédié à Gradle .

Définition des arômes du produit

Les `build.gradle` produit sont définies dans le fichier `build.gradle` à l'intérieur du bloc `android { ... }` , comme `build.gradle` ci-dessous.

```
...
```

```
android {
    ...
    productFlavors {
        free {
            applicationId "com.example.app.free"
            versionName "1.0-free"
        }
        paid {
            applicationId "com.example.app.paid"
            versionName "1.0-paid"
        }
    }
}
```

Ce faisant, nous avons maintenant deux nouvelles saveurs de produits: `free` et `paid` . Chacun peut avoir sa propre configuration et ses propres attributs. Par exemple, nos deux nouvelles versions ont un `versionName` d' `applicationId` et un nom de `versionName` distincts de ceux de notre `versionName` `main` existante (disponible par défaut, donc non affichée ici).

Ajout de dépendances spécifiques au produit

Des dépendances peuvent être ajoutées pour une saveur de produit spécifique, de la même manière que vous pouvez les ajouter pour des configurations de construction spécifiques.

Pour cet exemple, supposons que nous ayons déjà défini deux variantes de produits appelées `free` et `paid` (plus sur la définition des saveurs ici).
Nous pouvons alors ajouter la dépendance AdMob pour la version `free` et la bibliothèque Picasso pour la version `paid` comme ceci:

```
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'
}
...
```

Ajout de ressources spécifiques au produit

Des ressources peuvent être ajoutées pour une saveur de produit spécifique.

Pour cet exemple, supposons que nous avons déjà défini deux versions de produits appelées `free` et `paid` . Afin d'ajouter des ressources spécifiques aux produits, nous créons des dossiers de ressources supplémentaires à côté du dossier `main/res` , auquel nous pouvons ensuite ajouter des ressources comme d'habitude. Pour cet exemple, nous allons définir une chaîne, `status` , pour chaque version de produit:

*/ src / **main** /res/values/strings.xml*

```
<resources>
    <string name="status">Default</string>
</resources>
```

*/ src / **free** /res/values/strings.xml*

```
<resources>
    <string name="status">Free</string>
</resources>
```

*/ src / **paid** /res/values/strings.xml*

```
<resources>
    <string name="status">Paid</string>
</resources>
```

Les chaînes de `status` spécifiques à l'arôme du produit remplacent la valeur de `status` dans l'arôme `main` .

Définir et utiliser les champs de configuration de la construction

BuildConfigField

Gradle permet `buildConfigField` lignes `buildConfigField` de définir des constantes. Ces constantes seront accessibles au moment de l'exécution en tant que champs statiques de la classe `BuildConfig` . Cela peut être utilisé pour créer des arômes en définissant tous les champs dans le bloc `defaultConfig` , puis en les `defaultConfig` des `defaultConfig` build individuelles, selon les besoins.

Cet exemple définit la date de construction et marque la génération pour la production plutôt que de tester:

```
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'
        }
    }
}
```

Le <package_name> généré automatiquement. `BuildConfig` .java dans le dossier gen contient les champs suivants basés sur la directive ci-dessus:

```
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";
}
```

Les champs définis peuvent désormais être utilisés dans l'application lors de l'exécution en accédant à la classe `BuildConfig` générée:

```
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

La `resValue` dans le `productFlavors` crée une valeur de ressource. Cela peut être n'importe quel type de ressource (`string` , `dimen` , `color` , etc.). Cela revient à définir une ressource dans le fichier approprié: par exemple, définir une chaîne dans un `strings.xml` . L'avantage étant que celui défini dans gradle peut être modifié en fonction de votre productFlavor / buildVariant. Pour accéder à la valeur, écrivez le même code que si vous accédiez à un fichier res à partir du fichier de

ressources:

```
getResources().getString(R.string.app_name)
```

L'important est que les ressources définies de cette manière ne puissent pas modifier les ressources existantes définies dans les fichiers. Ils ne peuvent créer que de nouvelles valeurs de ressource.

Certaines bibliothèques (telles que l'API Google Maps Android) nécessitent une clé API fournie dans le manifeste sous la forme d'une balise de `meta-data` . Si des clés différentes sont nécessaires pour le débogage et les générations de production, spécifiez un espace réservé de manifeste rempli par Gradle.

Dans votre fichier `AndroidManifest.xml` :

```
<meta-data
    android:name="com.google.android.geo.API_KEY"
    android:value="${MAPS_API_KEY}"/>
```

Et puis définissez le champ en conséquence dans votre fichier `build.gradle` :

```
android {
    defaultConfig {
        ...
        // Your development key
        manifestPlaceholders = [ MAPS_API_KEY: "AIza..." ]
    }

    productFlavors {
        prod {
            // Your production key
            manifestPlaceholders = [ MAPS_API_KEY: "AIza..." ]
        }
    }
}
```

Le système de génération Android génère automatiquement un certain nombre de champs et les place dans `BuildConfig.java` . Ces champs sont:

Champ	La description
`DEBUG`	un `Boolean` indiquant si l'application est en mode debug ou release
`APPLICATION_ID`	une `String` contenant l'ID de l'application (par exemple, `com.example.app`)
`BUILD_TYPE`	une `String` contenant le type de construction de l'application (généralement `debug` ou `release`)
`FLAVOR`	une `String` contenant la saveur particulière de la construction
`VERSION_CODE`	un `int` contenant le numéro de version (build).

Champ	La description
	Ceci est le même que `versionCode` dans `build.gradle` ou `versionCode` dans `AndroidManifest.xml`
`VERSION_NAME`	une `String` contenant le nom de la version (build). Ceci est le même que `versionName` dans `build.gradle` ou `versionName` dans `AndroidManifest.xml`

En plus de ce qui précède, si vous avez défini plusieurs dimensions de saveur, chaque dimension aura sa propre valeur. Par exemple, si vous avez deux dimensions de `color` pour la `color` et la `size` vous aurez également les variables suivantes:

Champ	La description
`FLAVOR_color`	une `String` contenant la valeur de la saveur 'color'.
`FLAVOR_size`	une `String` contenant la valeur de la saveur "taille".

Centraliser les dépendances via le fichier "dependencies.gradle"

Lorsque vous travaillez avec des projets multi-modules, il est utile de centraliser les dépendances dans un seul emplacement plutôt que de les répartir sur de nombreux fichiers de construction, en particulier pour les bibliothèques communes telles que les bibliothèques de support Android et les bibliothèques Firebase .

Une méthode recommandée consiste à séparer les fichiers de construction Gradle, avec un `build.gradle` par module, ainsi qu'un dans la racine du projet et un autre pour les dépendances, par exemple:

```
root
  +- gradleScript/
  |     dependencies.gradle
  +- module1/
  |     build.gradle
  +- module2/
  |     build.gradle
  +- build.gradle
```

Ensuite, toutes vos dépendances peuvent être situées dans `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}",
    ];
}
```

Ce qui peut alors être appliqué à partir de ce fichier dans le fichier de niveau supérieur `build.gradle` comme ceci:

```
// Load dependencies
apply from: 'gradleScript/dependencies.gradle'
```

et dans le `module1/build.gradle` comme ceci:

```
// Module build file
dependencies {
    // ...
    compile supportDependencies.appCompat
    compile supportDependencies.design
    compile firebaseDependencies.crash
}
```

Une autre approche

Une approche moins détaillée pour centraliser les versions de dépendances de bibliothèque peut être obtenue en déclarant une fois le numéro de version comme variable et en l'utilisant partout.

Dans la racine de l'espace de travail, `build.gradle` ajoute ceci:

```
ext.v = [
    supportVersion:'24.1.1',
]
```

Et dans chaque module qui utilise la même bibliothèque, ajoutez les bibliothèques nécessaires

```
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}"
```

Structure de répertoire pour les ressources spécifiques aux saveurs

Les différentes versions des applications peuvent contenir des ressources différentes. Pour créer une ressource spécifique à une saveur, créez un répertoire avec le nom en minuscule de votre saveur dans le répertoire `src` et ajoutez vos ressources de la même manière que vous le feriez normalement.

Par exemple, si vous aviez un `Development` saveur et que vous souhaitiez lui fournir une icône de lanceur distincte, vous `src/development/res/drawable-mdpi` un répertoire `src/development/res/drawable-mdpi` et créez un fichier `ic_launcher.png` avec votre icône de développement.

La structure de répertoire ressemblera à ceci:

```
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'
```

(Bien sûr, dans ce cas, vous créeriez également des icônes pour drawable-hdpi, drawable-xhdpi, *etc.*).

Pourquoi y a-t-il deux fichiers build.gradle dans un projet Android Studio?

`<PROJECT_ROOT>\app\build.gradle` est spécifique au **module d'application** .

`<PROJECT_ROOT>\build.gradle` est un **"fichier de construction de premier niveau" dans** lequel vous pouvez ajouter des options de configuration communes à tous les sous-projets / modules.

Si vous utilisez un autre module dans votre projet, en tant que bibliothèque locale, vous auriez un autre fichier `build.gradle` : `<PROJECT_ROOT>\module\build.gradle`

Dans le fichier de niveau supérieur, vous pouvez spécifier des propriétés communes en tant que bloc de génération ou certaines propriétés communes.

```
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"
}
```

Dans l' `app\build.gradle` vous ne définissez que les propriétés du module:

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


android {
    compileSdkVersion rootProject.ext.compileSdkVersion
    buildToolsVersion rootProject.ext.buildToolsVersion
}

dependencies {
    //.....
}
```

Exécuter un script shell à partir de gradle

Un script shell est un moyen très polyvalent d'étendre votre build à tout ce que vous pouvez imaginer.

En guise d'exemple, voici un script simple pour compiler des fichiers protobuf et ajouter les fichiers Java de résultat au répertoire source pour une compilation supplémentaire:

```
def compilePb() {
    exec {
        // NOTICE: gradle will fail if there's an error in the protoc file...
        executable "../pbScript.sh"
    }
}

project.afterEvaluate {
    compilePb()
}
```

Le script shell 'pbScript.sh' pour cet exemple, situé dans le dossier racine du projet:

```
#!/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
```

Déboguer vos erreurs Gradle

Ce qui suit est un extrait de Gradle - Quelle est une valeur de sortie non nulle et comment puis-je la corriger? , voir pour la discussion complète.

Disons que vous développez une application et que vous obtenez une erreur de Gradle qui apparaît généralement comme telle.

```
: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
```

Vous recherchez ici votre problème sur StackOverflow, et les gens disent de nettoyer et de reconstruire votre projet ou d'activer MultiDex , et lorsque vous essayez cela, cela ne résout tout simplement pas le problème.

Il existe des moyens d'obtenir plus d'informations , mais la sortie de Gradle elle-même doit indiquer l'erreur réelle dans les quelques lignes au-dessus du message entre: `module:someTask FAILED` et le dernier `:module:someOtherTask` qui a réussi. Par conséquent, si vous posez une question sur votre erreur, veuillez modifier vos questions pour inclure plus de contexte à l'erreur.

Donc, vous obtenez une "valeur de sortie non nulle". Eh bien, ce nombre est un bon indicateur de ce que vous devriez essayer de corriger. En voici quelques-unes les plus fréquentes.

- `1` est juste un code d'erreur général et l'erreur est probable dans la sortie Gradle
- `2` semble être lié à des dépendances qui se chevauchent ou à une mauvaise configuration du projet.
- `3` semble être d'inclure trop de dépendances, ou un problème de mémoire.

Les solutions générales pour ce qui précède (après avoir tenté un nettoyage et une reconstruction du projet) sont les suivantes:

- `1` - Résoudre l'erreur mentionnée. En général, il s'agit d'une erreur de compilation, ce qui signifie qu'une partie de code de votre projet n'est pas valide. Cela inclut à la fois XML et Java pour un projet Android.
- `2` & `3` - Plusieurs réponses vous indiquent d'activer le multidex . Bien qu'il puisse résoudre le problème, il s'agit probablement d'une solution de contournement. Si vous ne comprenez pas pourquoi vous l'utilisez (voir le lien), vous n'en avez probablement pas besoin. Les solutions générales impliquent de réduire la surutilisation des dépendances de bibliothèque (comme tous les services Google Play, par exemple, lorsque vous n'utilisez qu'une seule bibliothèque, par exemple Maps ou Sign-In).

Spécification de différents ID d'application pour les types de construction et les variantes de produit

Vous pouvez spécifier différents ID d'application ou noms de package pour chaque `buildType` ou `productFlavor` utilisant l'attribut de configuration **applicationIdSuffix** :

Exemple de suffixe de l' `applicationId` pour chaque `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"
    }
}
```

Nos `applicationIds` résultantes seraient désormais:

- com.package.android pour `release`
- com.package.android. **dev** pour le `development`
- com.package.android. **qa** pour `testing`

Cela peut aussi être fait pour `productFlavors` :

```
productFlavors {
    free {
        applicationIdSuffix ".free"
    }
    paid {
        applicationIdSuffix ".paid"
    }
}
```

L' `applicationIds` résultante serait:

- com.package.android. **gratuit** pour la saveur `free`
- com.package.android. **payé** pour la saveur `paid`

Signer APK sans exposer le mot de passe du magasin de clés

Vous pouvez définir la configuration de signature pour signer le fichier `build.gradle` dans le fichier `build.gradle` en utilisant ces propriétés:

- `storeFile` : le fichier de `storeFile`
- `storePassword` : le mot de passe du `storePassword`

- `keyAlias` : un nom d'alias de clé
- `keyPassword` : un mot de passe alias de clé

Dans de nombreux cas, vous devrez peut-être éviter ce genre d'informations dans le fichier `build.gradle` .

Méthode A: Configurer la signature de version à l'aide d'un fichier keystore.properties

Il est possible de configurer le `build.gradle` votre application pour qu'il lise vos informations de configuration de signature à partir d'un fichier de propriétés tel que `keystore.properties` .

Mettre en place une telle signature est bénéfique car:

- Vos informations de configuration de signature sont distinctes de votre fichier `build.gradle`
- Vous n'avez pas besoin d'intervenir pendant le processus de signature pour fournir des mots de passe pour votre fichier de clés.
- Vous pouvez facilement exclure le fichier `keystore.properties` du contrôle de version

Tout d'abord, créez un fichier appelé `keystore.properties` à la racine de votre projet avec un contenu comme celui-ci (en remplaçant les valeurs par les vôtres):

```
storeFile=keystore.jks
storePassword=storePassword
keyAlias=keyAlias
keyPassword=keyPassword
```

Maintenant, dans le fichier `build.gradle` votre application, configurez le bloc `signingConfigs` comme suit:

```
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']
            }
        }
    }
}
```

C'est tout ce qu'il y a à faire, **mais n'oubliez pas d'exclure votre fichier de clés et votre fichier `keystore.properties` du contrôle de version** .

Quelques points à noter:

- Le chemin `storeFile` spécifié dans le fichier `keystore.properties` doit être relatif au fichier `build.gradle` votre application. Cet exemple suppose que le fichier de clés se trouve dans le même répertoire que le fichier `build.gradle` l'application.
- Cet exemple contient le fichier `keystore.properties` à la racine du projet. Si vous le placez ailleurs, veillez à changer la valeur dans `rootProject.file('keystore.properties')` à l'emplacement de votre compte, par rapport à la racine de votre projet.

Méthode B: En utilisant une variable d'environnement

La même chose peut être réalisée sans fichier de propriétés, ce qui rend le mot de passe plus difficile à trouver:

```
android {

  signingConfigs {
    release {
        storeFile file('/your/keystore/location/key')
        keyAlias 'your_alias'
        String ps = System.getenv("ps")
        if (ps == null) {
             throw new GradleException('missing ps env variable')
        }
        keyPassword ps
        storePassword ps
    }
}
```

La variable d'environnement `"ps"` peut être globale, mais une approche plus sûre peut être en l'ajoutant uniquement au shell d'Android Studio.
Dans Linux, cela peut être fait en éditant `Desktop Entry` Android Studio

```
Exec=sh -c "export ps=myPassword123 ; /path/to/studio.sh"
```

Vous pouvez trouver plus de détails dans cette rubrique .

Gestion des versions de votre build via le fichier "version.properties"

Vous pouvez utiliser Gradle pour incrémenter automatiquement la version de votre package à chaque fois que vous la créez. Pour ce faire, créez un fichier `version.properties` dans le même répertoire que votre `build.gradle` avec le contenu suivant:

```
VERSION_MAJOR=0
```

```
VERSION_MINOR=1
VERSION_BUILD=1
```

(Changer les valeurs pour majeur et mineur comme bon vous semble). Ensuite, dans votre `build.gradle` ajoutez le code suivant à la section `android` :

```
// Read version information from local file and increment as appropriate
def versionPropsFile = file('version.properties')
if (versionPropsFile.canRead()) {
  def Properties versionProps = new Properties()

  versionProps.load(new FileInputStream(versionPropsFile))

  def versionMajor = versionProps['VERSION_MAJOR'].toInteger()
  def versionMinor = versionProps['VERSION_MINOR'].toInteger()
  def versionBuild = versionProps['VERSION_BUILD'].toInteger() + 1

  // Update the build number in the local file
  versionProps['VERSION_BUILD'] = versionBuild.toString()
  versionProps.store(versionPropsFile.newWriter(), null)

  defaultConfig {
    versionCode versionBuild
    versionName "${versionMajor}.${versionMinor}." + String.format("%05d", versionBuild)
  }
}
```

Les informations sont accessibles en Java sous forme de chaîne `BuildConfig.VERSION_NAME` pour le `BuildConfig.VERSION_NAME` complet {major}. { `BuildConfig.VERSION_NAME` }. { `BuildConfig.VERSION_CODE` } et en tant `BuildConfig.VERSION_CODE` pour le numéro de build uniquement.

Modification du nom apk de la sortie et ajout du nom de la version:

C'est le code pour changer le nom du fichier d'application de sortie (.apk). Le nom peut être configuré en affectant une valeur différente à `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 + "]");
        }
    }
}
```

Désactiver la compression d'image pour une taille de fichier APK plus petite

Si vous optimisez toutes les images manuellement, désactivez APT Cruncher pour une taille de fichier APK plus petite.

```
android {

    aaptOptions {
        cruncherEnabled = false
    }
}
```

Activer Proguard en utilisant gradle

Pour activer les configurations Proguard pour votre application, vous devez l'activer dans votre fichier de gradation au niveau du module. Vous devez définir la valeur de `minifyEnabled` sur `true` .

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

Le code ci-dessus appliquera vos configurations Proguard contenues dans le SDK Android par défaut combiné au fichier "proguard-rules.pro" de votre module à votre apk publiée.

Activer le support expérimental du plug-in NDK pour Gradle et AndroidStudio

Activez et configurez le plug-in expérimental Gradle pour améliorer le support NDK d'AndroidStudio. Vérifiez que vous remplissez les conditions suivantes:

- Gradle 2.10 (pour cet exemple)
- Android NDK r10 ou supérieur
- SDK Android avec les outils de compilation v19.0.0 ou version ultérieure

Configurer le fichier MyApp / build.gradle

Editez la ligne dependencies.classpath dans build.gradle à partir par exemple

```
classpath 'com.android.tools.build:gradle:2.1.2'
```

à

```
classpath 'com.android.tools.build:gradle-experimental:0.7.2'
```

(v0.7.2 était la dernière version au moment de l'écriture. Vérifiez la dernière version vous-même et adaptez votre ligne en conséquence)

Le fichier build.gradle doit ressembler à ceci:

```
buildscript {
    repositories {
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle-experimental:0.7.2'
    }
}

allprojects {
    repositories {
        jcenter()
    }
}

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

Configurez le fichier MyApp / app / build.gradle

Modifiez le fichier build.gradle pour qu'il ressemble à l'exemple suivant. Vos numéros de version peuvent être différents.

```
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'])
}
```

Synchronisez et vérifiez qu'il n'y a pas d'erreurs dans les fichiers Gradle avant de continuer.

Tester si le plugin est activé

Assurez-vous d'abord d'avoir téléchargé le module Android NDK. Créez ensuite une nouvelle application dans AndroidStudio et ajoutez les éléments suivants au fichier ActivityMain:

```
public class MainActivity implements Activity {
    onCreate() {
        // Pregenerated code. Not important here
    }
    static {
        System.loadLibrary("myLib");
    }
    public static native String getString();
}
```

La partie `getString()` doit être surlignée en rouge pour indiquer que la fonction JNI correspondante est introuvable. Passez votre souris sur l'appel de fonction jusqu'à ce qu'une ampoule rouge apparaisse. Cliquez sur l'ampoule et sélectionnez `create function JNI_...` Cela devrait générer un fichier myLib.c dans le répertoire myApp / app / src / main / jni avec l'appel de fonction JNI correct. Cela devrait ressembler à ceci:

```
#include <jni.h>

JNIEXPORT jstring JNICALL
Java_com_example_mydomain_myapp_MainActivity_getString(JNIEnv *env, jobject instance)
{
    // TODO

    return (*env)->NewStringUTF(env, returnValue);
}
```

Si cela ne ressemble pas à ceci, alors le plugin n'a pas été correctement configuré ou le NDK n'a pas été téléchargé

Afficher toutes les tâches du projet de graduation

```
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
```

Supprimer automatiquement "non aligné" apk

Si vous n'avez pas besoin de fichiers apk générés automatiquement avec un suffixe `unaligned` (que vous n'avez probablement pas), vous pouvez ajouter le code suivant au fichier `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()
        }
    }
  }
}
```

D' ici

Ignorer la variante de construction

Pour certaines raisons, vous pouvez ignorer vos variantes de build. Par exemple: vous avez une saveur de produit «fictive» et vous l'utilisez uniquement à des fins de débogage, comme les tests d'unité / d'instrumentation.

Ignorons la variante **mockRelease** de notre projet. Ouvrez le fichier **build.gradle** et écrivez:

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

Voir arbre de dépendance

Utilisez les dépendances de tâches. En fonction de la configuration de vos modules, cela peut être soit des `./gradlew dependencies` soit des dépendances de l'application du module `./gradlew :app:dependencies`

L'exemple suivant du fichier build.gradle

```
dependencies {
    compile 'com.android.support:design:23.2.1'
    compile 'com.android.support:cardview-v7:23.1.1'

    compile 'com.google.android.gms:play-services:6.5.87'
}
```

produira le graphique suivant:

```
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 (*)

. . .
```

Ici, vous pouvez voir que le projet inclut directement `com.android.support:design` version 23.2.1, qui apporte lui-même `com.android.support:support-v4` avec la version 23.2.1. Cependant, `com.google.android.gms:play-services` lui `com.google.android.gms:play-services` même a une dépendance sur le même `support-v4` mais avec une ancienne version 21.0.0, qui est un conflit détecté par gradle.

`(*)` sont utilisés lorsque gradle ignore le sous-arbre car ces dépendances étaient déjà répertoriées précédemment.

Utilisez gradle.properties pour les versions centralisées de versionnumber / build

Vous pouvez définir les informations de configuration centrales dans

- un fichier d'inclusion de graduation séparé Centraliser les dépendances via le fichier "dependencies.gradle"
- un fichier de propriétés autonome Contrôle de version de vos builds via le fichier "version.properties"

ou faites-le avec le fichier racine `gradle.properties`

la structure du projet

```
root
  +- module1/
  |     build.gradle
  +- module2/
  |     build.gradle
  +- build.gradle
  +- gradle.properties
```

paramètre global pour tous les sous-modules dans 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
```

utilisation dans un sous-module

```
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 {
    ...
}
```

Remarque: Si vous souhaitez publier votre application dans le magasin d'applications F-Droid, vous devez utiliser des chiffres magiques dans le fichier de graduation, sinon le robot f-droid ne peut pas lire le versionnumner actuel pour détecter / vérifier les modifications de version.

Afficher les informations de signature

Dans certaines circonstances (par exemple, l'obtention d'une clé API Google), vous devez rechercher votre empreinte digitale de fichier de clés. Gradle a une tâche pratique qui affiche toutes les informations de signature, y compris les empreintes de clés:

```
./gradlew signingReport
```

Ceci est un exemple de sortie:

```
: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
----------
```

Définition des types de construction

Vous pouvez créer et configurer des types de construction dans le fichier `build.gradle` niveau du module `build.gradle` dans le bloc `android {}` .

```
    android {
        ...
        defaultConfig {...}

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

            debug {
                applicationIdSuffix ".debug"
            }
        }
    }
```

Lire Gradle pour Android en ligne: https://riptutorial.com/fr/android/topic/95/gradle-pour-android

Chapitre 136: GreenDAO

Introduction

GreenDAO est une bibliothèque de mappage objet-relationnel pour aider les développeurs à utiliser les bases de données SQLite pour le stockage local persistant.

Examples

Méthodes d'aide pour les requêtes SELECT, INSERT, DELETE, UPDATE

Cet exemple montre une classe d'assistance qui contient des méthodes utiles lors de l'exécution des requêtes pour les données. Chaque méthode utilise Java Generic afin d'être très flexible.

```
public <T> List<T> selectElements(AbstractDao<T, ?> dao) {
    if (dao == null) {
        return null;
    }
    QueryBuilder<T> qb = dao.queryBuilder();
    return qb.list();
}

public <T> void insertElements(AbstractDao<T, ?> absDao, List<T> items) {
    if (items == null || items.size() == 0 || absDao == null) {
        return;
    }
    absDao.insertOrReplaceInTx(items);
}

public <T> T insertElement(AbstractDao<T, ?> absDao, T item) {
    if (item == null || absDao == null) {
        return null;
    }
    absDao.insertOrReplaceInTx(item);
    return item;
}

public <T> void updateElements(AbstractDao<T, ?> absDao, List<T> items) {
    if (items == null || items.size() == 0 || absDao == null) {
        return;
    }
    absDao.updateInTx(items);
}

public <T> T selectElementByCondition(AbstractDao<T, ?> absDao,
                                      WhereCondition... conditions) {
    if (absDao == null) {
        return null;
    }
    QueryBuilder<T> qb = absDao.queryBuilder();
    for (WhereCondition condition : conditions) {
        qb = qb.where(condition);
    }
    List<T> items = qb.list();
```

```
    return items != null && items.size() > 0 ? items.get(0) : null;
}

public <T> List<T> selectElementsByCondition(AbstractDao<T, ?> absDao,
                                             WhereCondition... conditions) {
    if (absDao == null) {
        return null;
    }
    QueryBuilder<T> qb = absDao.queryBuilder();
    for (WhereCondition condition : conditions) {
        qb = qb.where(condition);
    }
    List<T> items = qb.list();
    return items != null ? items : null;
}

public <T> List<T> selectElementsByConditionAndSort(AbstractDao<T, ?> absDao,
                                                    Property sortProperty,
                                                    String sortStrategy,
                                                    WhereCondition... conditions) {
    if (absDao == null) {
        return null;
    }
    QueryBuilder<T> qb = absDao.queryBuilder();
    for (WhereCondition condition : conditions) {
        qb = qb.where(condition);
    }
    qb.orderCustom(sortProperty, sortStrategy);
    List<T> items = qb.list();
    return items != null ? items : null;
}

public <T> List<T> selectElementsByConditionAndSortWithNullHandling(AbstractDao<T, ?> absDao,
                                                                    Property sortProperty,
                                                                    boolean handleNulls,
                                                                    String sortStrategy,
                                                                    WhereCondition...
conditions) {
    if (!handleNulls) {
        return selectElementsByConditionAndSort(absDao, sortProperty, sortStrategy,
conditions);
    }
    if (absDao == null) {
        return null;
    }
    QueryBuilder<T> qb = absDao.queryBuilder();
    for (WhereCondition condition : conditions) {
        qb = qb.where(condition);
    }
    qb.orderRaw("(CASE WHEN " + "T." + sortProperty.columnName + " IS NULL then 1 ELSE 0
END)," + "T." + sortProperty.columnName + " " + sortStrategy);
    List<T> items = qb.list();
    return items != null ? items : null;
}

public <T, V extends Class> List<T> selectByJoin(AbstractDao<T, ?> absDao,
                                                 V className,
                                                 Property property, WhereCondition
whereCondition) {
    QueryBuilder<T> qb = absDao.queryBuilder();
    qb.join(className, property).where(whereCondition);
```

```
    return qb.list();
}

public <T> void deleteElementsByCondition(AbstractDao<T, ?> absDao,
                                          WhereCondition... conditions) {
    if (absDao == null) {
        return;
    }
    QueryBuilder<T> qb = absDao.queryBuilder();
    for (WhereCondition condition : conditions) {
        qb = qb.where(condition);
    }
    List<T> list = qb.list();
    absDao.deleteInTx(list);
}

public <T> T deleteElement(DaoSession session, AbstractDao<T, ?> absDao, T object) {
    if (absDao == null) {
        return null;
    }
    absDao.delete(object);
    session.clear();
    return object;
}

public <T, V extends Class> void deleteByJoin(AbstractDao<T, ?> absDao,
                                              V className,
                                              Property property, WhereCondition
whereCondition) {
    QueryBuilder<T> qb = absDao.queryBuilder();
    qb.join(className, property).where(whereCondition);
    qb.buildDelete().executeDeleteWithoutDetachingEntities();
}

public <T> void deleteAllFromTable(AbstractDao<T, ?> absDao) {
    if (absDao == null) {
        return;
    }
    absDao.deleteAll();
}

public <T> long countElements(AbstractDao<T, ?> absDao) {
    if (absDao == null) {
        return 0;
    }
    return absDao.count();
}
```

Création d'une entité avec GreenDAO 3.X qui possède une clé primaire composite

Lors de la création d'un modèle pour une table dotée d'une clé primaire composite, un travail supplémentaire est requis sur l'objet pour que l'entité modèle respecte ces contraintes.

L'exemple suivant de table SQL et d'entité illustre la structure permettant de stocker une révision laissée par un client pour un élément dans un magasin en ligne. Dans cet exemple, nous voulons que les colonnes `customer_id` et `item_id` soient une clé primaire composite, ce qui ne permet qu'une seule révision entre un client et un article spécifiques.

Table SQL

```
CREATE TABLE review (
    customer_id STRING NOT NULL,
    item_id STRING NOT NULL,
    star_rating INTEGER NOT NULL,
    content STRING,
    PRIMARY KEY (customer_id, item_id)
);
```

Habituellement, nous utiliserions les annotations `@Id` et `@Unique` au-dessus des champs respectifs de la classe d'entités, mais pour une clé primaire composite, nous procédons comme suit:

1. Ajoutez l'annotation `@Index` à l'intérieur de l'annotation `@Entity` niveau de la classe. La propriété value contient une liste séparée par des virgules des champs qui constituent la clé. Utilisez la propriété `unique` comme indiqué pour appliquer l'unicité à la clé.

2. GreenDAO exige que chaque entité ait un objet `long` ou `Long` tant que clé primaire. Nous devons toujours ajouter ceci à la classe Entity, mais nous n'avons pas besoin de l'utiliser ou de nous en préoccuper pour notre implémentation. Dans l'exemple ci-dessous, il s'appelle `localID`

Entité

```
@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() {}
}
```

Démarrer avec GreenDao v3.X

Après avoir ajouté la dépendance à la bibliothèque GreenDao et le plug-in Gradle, vous devez d'abord créer un objet entité.

Entité

Une entité est un objet *POJO (* Plain Old Java Object *)* qui modélise certaines données dans la base de données. GreenDao utilisera cette classe pour créer une table dans la base de données SQLite et générer automatiquement des classes d'assistance que nous pouvons utiliser pour accéder et stocker des données sans avoir à écrire d'instructions SQL.

```
@Entity
public class Users {

    @Id(autoincrement = true)
    private Long id;

    private String firstname;
    private String lastname;

    @Unique
    private String email;

    // Getters and setters for the fields...

}
```

Configuration unique de GreenDao

Chaque fois qu'une application est lancée, GreenDao doit être initialisé. GreenDao suggère de conserver ce code dans une classe Application ou quelque part il ne sera exécuté qu'une seule fois.

```
DaoMaster.DevOpenHelper helper = new DaoMaster.DevOpenHelper(this, "mydatabase", null);
db = helper.getWritableDatabase();
DaoMaster daoMaster = new DaoMaster(db);
DaoSession daoSession = daoMaster.newSession();
```

Classes d'assistance GreenDao

Une fois l'objet entité créé, GreenDao crée automatiquement les classes d'assistance utilisées pour interagir avec la base de données. Celles-ci sont nommées de la même manière que le nom de l'objet entité créé, suivi de `Dao` et récupérées à partir de l'objet `daoSession` .

```
UsersDao usersDao = daoSession.getUsersDao();
```

De nombreuses actions de base de données typiques peuvent maintenant être effectuées à l'aide de cet objet Dao avec l'objet entité.

Question

```
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();
```

Insérer

```
Users newUser = new User("John","Doe","jdoe@example.com");
usersDao.insert(newUser);
```

Mettre à jour

```
// Modify a previously retrieved user object and update
user.setLastname("Dole");
usersDao.update(user);
```

Effacer

```
// Delete a previously retrieved user object
usersDao.delete(user);
```

Lire GreenDAO en ligne: https://riptutorial.com/fr/android/topic/1345/greendao

Chapitre 137: GreenRobot EventBus

Syntaxe

- @Subscribe (threadMode = ThreadMode.POSTING) public void onEvent (événement EventClass) {}

Paramètres

Mode filetage	La description
`ThreadMode.POSTING`	Sera appelé sur le même thread que l'événement a été posté sur. C'est le mode par défaut.
`ThreadMode.MAIN`	Sera appelé sur le thread principal de l'interface utilisateur.
`ThreadMode.BACKGROUND`	Sera appelé sur un fil de fond. Si le thread de publication n'est pas le thread principal, il sera utilisé. S'il est posté sur le thread principal, `EventBus` possède un seul thread d'arrière-plan qu'il utilisera.
`ThreadMode.ASYNC`	Sera appelé sur son propre fil.

Examples

Création d'un objet événement

Pour envoyer et recevoir des événements, nous avons d'abord besoin d'un objet Event. Les objets événement sont en fait de simples POJO.

```
public class ArbitaryEvent{
    public static final int TYPE_1 = 1;
    public static final int TYPE_2 = 2;
    private int eventType;
    public ArbitaryEvent(int eventType){
        this.eventType = eventType;
    }

    public int getEventType(){
        return eventType;
    }
}
```

Recevoir des événements

Pour recevoir des événements, vous devez enregistrer votre classe sur le `EventBus` .

```
@Override
public void onStart() {
    super.onStart();
    EventBus.getDefault().register(this);
}

@Override
public void onStop() {
   EventBus.getDefault().unregister(this);
   super.onStop();
}
```

Et puis abonnez-vous aux événements.

```
@Subscribe(threadMode = ThreadMode.MAIN)
public void handleEvent(ArbitaryEvent event) {
    Toast.makeText(getActivity(), "Event type: "+event.getEventType(),
Toast.LENGTH_SHORT).show();
}
```

Envoi d'événements

L'envoi d'événements est aussi simple que la création de l'objet Event et sa publication.

```
EventBus.getDefault().post(new ArbitaryEvent(ArbitaryEvent.TYPE_1));
```

Passer un événement simple

La première chose à faire est d'ajouter EventBus au fichier de dégradé de notre module:

```
dependencies {
    ...
    compile 'org.greenrobot:eventbus:3.0.0'
    ...
}
```

Maintenant, nous devons créer un modèle pour notre événement. Il peut contenir tout ce que nous voulons transmettre. Pour l'instant, nous allons simplement créer une classe vide.

```
public class DeviceConnectedEvent
{
}
```

Maintenant, nous pouvons ajouter le code à notre `Activity` qui sera enregistré avec EventBus et abonné à l'événement.

```
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
    }
}
```

Dans cette `Activity` nous obtenons une instance de `EventBus` dans la méthode `onCreate()` . Nous enregistrons / `onStart()` pour les événements dans `onStart()` / `onStop()` . Il est important de vous rappeler de vous désinscrire lorsque votre auditeur perd de la portée ou que vous risquez de perdre votre `Activity` .

Enfin, nous définissons la méthode que nous voulons appeler avec l'événement. L'annotation `@Subscribe` indique à EventBus les méthodes qu'il peut rechercher pour gérer les événements. Vous devez avoir au moins une méthode annotée avec `@Subscribe` pour vous inscrire à EventBus ou une exception. Dans l'annotation, nous définissons le mode thread. Ceci indique à EventBus quel thread appelle la méthode. C'est un moyen très pratique de transmettre des informations d'un thread d'arrière-plan au thread d'interface utilisateur! C'est exactement ce que nous faisons ici. `ThreadMode.MAIN` signifie que cette méthode sera appelée sur le thread d'interface utilisateur principal d'Android, de sorte qu'il est possible d'effectuer des manipulations d'interface utilisateur ici. Le nom de la méthode n'a pas d'importance. Les seuls pensent, à part que l'annotation `@Subscribe` , que EventBus recherche est le type de l'argument. Tant que le type correspond, il sera appelé lorsqu'un événement est publié.

La dernière chose à faire pour poster un événement. Ce code sera dans notre `Service` .

```
EventBus.getDefault().post(new DeviceConnectedEvent());
```

C'est tout ce qu'on peut en dire! EventBus prend cet événement DeviceConnectedEvent et examine ses écouteurs enregistrés, examine les méthodes auxquelles ils ont souscrit et recherche ceux qui prennent un DeviceConnectedEvent comme argument et les appelle sur le thread auquel ils souhaitent être appelés.

Lire GreenRobot EventBus en ligne: https://riptutorial.com/fr/android/topic/3551/greenrobot-eventbus

Chapitre 138: Gson

Introduction

Gson est une bibliothèque Java pouvant être utilisée pour convertir des objets Java en leur représentation JSON. Gson considère ces deux objectifs comme des objectifs de conception très importants.

Caractéristiques de Gson:

Fournit des `toJson()` simples de `toJson()` et `fromJson()` pour convertir des objets Java en JSON et vice-versa

Autoriser les objets non modifiables préexistants à être convertis vers et depuis JSON

Prise en charge étendue de Java Generics

Prise en charge d'objets arbitrairement complexes (avec des hiérarchies d'héritage profondes et une utilisation extensive des types génériques)

Syntaxe

- Excluder excluder ()
- Champ FieldNamingStrategyNamingStrategy ()
- <T> T fromJson (JsonElement json, Class <T> classOfT)
- <T> T fromJson (JsonElement json, type typeOfT)
- <T> T fromJson (lecteur JsonReader, type typeOfT)
- <T> T fromJson (Reader json, Class <T> classOfT)
- <T> T fromJson (Reader json, type typeOfT)
- <T> T fromJson (String json, Class <T> classOfT)
- <T> T fromJson (String json, type typeOfT)
- <T> TypeAdapter <T> getAdapter (type Class <T>)
- <T> TypeAdapter <T> getAdapter (type TypeToken <T>)
- <T> TypeAdapter <T> getDelegateAdapter (typeAdapterFactory skipPast, type TypeToken <T>)
- JsonReader newJsonReader (lecteur de lecteur)
- JsonWriter newJsonWriter (Writer writer)
- JsonElement toJsonTree (objet src)
- JsonElement toJsonTree (objet src, type typeOfSrc)
- boolean serializeNulls ()
- boolean htmlSafe ()
- String toJson (JsonElement jsonElement)
- String toJson (Object src)
- String toJson (Object src, Type typeOfSrc)
- String toString ()

- void toJson (objet src, type typeOfSrc, écrivain appendable)
- void toJson (Object src, Type typeOfSrc, JsonWriter writer)
- void toJson (JsonElement jsonElement, écrivain appendable)
- void toJson (JsonElement jsonElement, JsonWriter writer)
- annuler toJson (objet src, écrivain appendable)

Examples

Analyse JSON avec Gson

L'exemple montre l'analyse d'un objet JSON à l'aide de la bibliothèque Gson de Google .

Objets d'analyse:

```
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 = "";
    }


}
```

Ensuite, lorsque l'analyse doit avoir lieu, utilisez les éléments suivants:

```
String robotJson = "{
                        \"version\": \"JellyBean\",
                        \"age\": 3,
                        \"robotName\": \"Droid\"
                    }";

Gson gson = new Gson();
Robot robot = gson.fromJson(robotJson, Robot.class);
```

Analyse d'une liste:

Lors de la récupération d'une liste d'objets JSON, vous souhaiterez souvent les analyser et les convertir en objets Java.

La chaîne JSON que nous allons essayer de convertir est la suivante:

```
{
  "owned_dogs": [
    {
      "name": "Ron",
      "age": 12,
      "breed": "terrier"
    },
    {
      "name": "Bob",
      "age": 4,
      "breed": "bulldog"
    },
    {
      "name": "Johny",
      "age": 3,
      "breed": "golden retriever"
    }
  ]
}
```

Ce tableau JSON particulier contient trois objets. Dans notre code Java, nous souhaitons mapper ces objets aux objets `Dog` . Un objet Chien ressemblerait à ceci:

```
private class Dog {
    public String name;
    public int age;

    @SerializedName("breed")
    public String breedName;
}
```

Pour convertir le tableau JSON en un `Dog[]` :

```
Dog[] arrayOfDogs = gson.fromJson(jsonArrayString, Dog[].class);
```

Conversion d'un `Dog[]` en chaîne JSON:

```
String jsonArray = gson.toJson(arrayOfDogs, Dog[].class);
```

Pour convertir le tableau JSON en `ArrayList<Dog>` nous pouvons procéder comme suit:

```
Type typeListOfDogs = new TypeToken<List<Dog>>(){}.getType();
List<Dog> listOfDogs = gson.fromJson(jsonArrayString, typeListOfDogs);
```

L'objet Type `typeListOfDogs` définit à quoi ressemblerait une liste d'objets `Dog` . GSON peut utiliser cet objet de type pour mapper le tableau JSON sur les bonnes valeurs.

Vous pouvez également convertir un `List<Dog>` en un tableau JSON de la même manière.

```
String jsonArray = gson.toJson(listOfDogs, typeListOfDogs);
```

Analyse de la propriété JSON pour énumérer avec Gson

Si vous voulez analyser une chaîne enum avec Gson:

> {"status": "open"}

```
public enum Status {
    @SerializedName("open")
    OPEN,
    @SerializedName("waiting")
    WAITING,
    @SerializedName("confirm")
    CONFIRM,
    @SerializedName("ready")
    READY
}
```

Analyse d'une liste avec Gson

Méthode 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());
```

Ceci est utile pour la plupart des classes de conteneurs génériques, car vous ne pouvez pas obtenir la classe d'un type paramétré (c'est-à-dire que vous ne pouvez pas appeler `List<String>.class`).

Méthode 2

```
public class StringList extends ArrayList<String> { }

...

List<String> members = gson.fromJson(json, StringList.class);
```

Alternativement, vous pouvez toujours sous-classer le type que vous voulez, puis passer dans cette classe. Cependant, ce n'est pas toujours la meilleure pratique, car cela vous renverra un objet de type `StringList` ;

Sérialisation / désérialisation JSON avec AutoValue et Gson

Importer dans votre fichier racine gradle

```
classpath 'com.neenbedankt.gradle.plugins:android-apt:1.8'
```

Importer dans votre fichier d'application Gradle

```
apt 'com.google.auto.value:auto-value:1.2'
apt 'com.ryanharter.auto.value:auto-value-gson:0.3.1'
provided 'com.jakewharton.auto.value:auto-value-annotations:1.2-update1'
provided 'org.glassfish:javax.annotation:10.0-b28'
```

Créer un objet avec autovalue:

```
@AutoValue public abstract class SignIn {
    @SerializedName("signin_token") public abstract String signinToken();
    public abstract String username();

    public static TypeAdapter<SignIn> typeAdapter(Gson gson) {
        return new AutoValue_SignIn.GsonTypeAdapter(gson);
    }

    public static SignIn create(String signin, String username) {
        return new AutoValue_SignIn(signin, username);
    }
}
```

Créez votre convertisseur Gson avec votre GsonBuilder

```
Gson gson = new GsonBuilder()
                .registerTypeAdapterFactory(
                    new AutoValueGsonTypeAdapterFactory())
                .create());
```

Désérialiser

```
String myJsonData = "{
    \"signin_token\": \"mySigninToken\",
    \"username\": \"myUsername\" }";
SignIn signInData = gson.fromJson(myJsonData, Signin.class);
```

Sérialiser

```
Signin myData = SignIn.create("myTokenData", "myUsername");
String myJsonData = gson.toJson(myData);
```

L'utilisation de Gson est un excellent moyen de simplifier le code de sérialisation et de désérialisation à l'aide d'objets POJO. L'effet secondaire est que la réflexion est coûteuse en termes de performances. C'est pourquoi l'utilisation d'AutoValue-Gson pour générer CustomTypeAdapter permet d'éviter ce coût de réflexion tout en restant très simple à mettre à jour en cas de changement d'API.

Analyse JSON en objet de classe générique avec Gson

Supposons que nous ayons une chaîne JSON:

```
["first","second","third"]
```

Nous pouvons analyser cette chaîne JSON dans un tableau `String` :

```
Gson gson = new Gson();
String jsonArray = "[\"first\",\"second\",\"third\"]";
String[] strings = gson.fromJson(jsonArray, String[].class);
```

Mais si nous voulons l'analyser dans un objet `List<String>` , nous devons utiliser `TypeToken` .

Voici l'échantillon:

```
Gson gson = new Gson();
String jsonArray = "[\"first\",\"second\",\"third\"]";
List<String> stringList = gson.fromJson(jsonArray, new TypeToken<List<String>>()
{}.getType());
```

Supposons que nous avons deux classes ci-dessous:

```
public class Outer<T> {
    public int index;
    public T data;
}

public class Person {
    public String firstName;
    public String lastName;
}
```

et nous avons une chaîne JSON qui doit être analysée dans un objet `Outer<Person>` .

Cet exemple montre comment analyser cette chaîne JSON avec l'objet de classe générique associé:

```
String json = "......";
Type userType = new TypeToken<Outer<Person>>(){}.getType();
Result<User> userResult = gson.fromJson(json,userType);
```

Si la chaîne JSON doit être analysée dans un objet `Outer<List<Person>>` :

```
Type userListType = new TypeToken<Outer<List<Person>>>(){}.getType();
Result<List<User>> userListResult = gson.fromJson(json,userListType);
```

Ajouter Gson à votre projet

```
dependencies {
    compile 'com.google.code.gson:gson:2.8.1'
}
```

Utiliser la dernière version de Gson

La ligne ci-dessous compilera la dernière version de la bibliothèque gson chaque fois que vous compilerez, vous n'avez pas à changer de version.

Avantages: Vous pouvez utiliser les dernières fonctionnalités, la vitesse et moins de bugs.
Inconvénients: Cela pourrait rompre la compatibilité avec votre code.

```
compile 'com.google.code.gson:gson:+'
```

Utiliser Gson pour charger un fichier JSON à partir du disque.

Cela va charger un fichier JSON à partir du disque et le convertir au type donné.

```
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);
}
```

Ajout d'un convertisseur personnalisé à Gson

Parfois, vous devez sérialiser ou désérialiser certains champs dans un format souhaité, par exemple votre backend peut utiliser le format "AAAA-MM-jj HH: mm" pour les dates et vous voulez que votre POJOS utilise la classe DateTime dans Joda Time.

Pour convertir automatiquement ces chaînes en objet DateTimes, vous pouvez utiliser un convertisseur personnalisé.

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

```
}
```

Pour que Gson utilise le convertisseur nouvellement créé, vous devez l'assigner lors de la création de l'objet Gson:

```
    DateTimeConverter dateTimeConverter = new DateTimeConverter();
    Gson gson = new GsonBuilder().registerTypeAdapter(DateTime.class, dateTimeConverter)
            .create();

    String s = gson.toJson(DateTime.now());
    // this will show the date in the desired format
```

Pour désérialiser la date dans ce format, il suffit de définir un champ au format DateTime:

```
public class SomePojo {
    private DateTime someDate;
}
```

Lorsque Gson rencontre un champ de type DateTime, il appelle votre convertisseur afin de désérialiser le champ.

Utiliser Gson comme sérialiseur avec Retrofit

Tout d'abord, vous devez ajouter `GsonConverterFactory` à votre fichier build.gradle

```
compile 'com.squareup.retrofit2:converter-gson:2.1.0'
```

Ensuite, vous devez ajouter la fabrique de convertisseurs lors de la création du service Retrofit:

```
Gson gson = new GsonBuilder().create();
new Retrofit.Builder()
        .baseUrl(someUrl)
        .addConverterFactory(GsonConverterFactory.create(gson))
        .build()
        .create(RetrofitService.class);
```

Vous pouvez ajouter des convertisseurs personnalisés lors de la création de l'objet Gson que vous transmettez à la fabrique. Vous permettant de créer des conversions de types personnalisées.

Analyse du tableau json en classe générique à l'aide de Gson

Supposons que nous ayons un json:

```
{
  "total_count": 132,
  "page_size": 2,
  "page_index": 1,
  "twitter_posts": [
    {
      "created_on": 1465935152,
```

```
      "tweet_id": 210462857140252672,
      "tweet": "Along with our new #Twitterbird, we've also updated our Display Guidelines",
      "url": "https://twitter.com/twitterapi/status/210462857140252672"
    },
    {
      "created_on": 1465995741,
      "tweet_id": 735128881808691200,
      "tweet": "Information on the upcoming changes to Tweets is now on the developer site",
      "url": "https://twitter.com/twitterapi/status/735128881808691200"
    }
  ]
}
```

Nous pouvons analyser ce tableau dans un objet Custom Tweets (conteneur de liste de tweets) manuellement, mais il est plus facile de le faire avec la méthode `fromJson` :

```
Gson gson = new Gson();
String jsonArray = "....";
Tweets tweets = gson.fromJson(jsonArray, Tweets.class);
```

Supposons que nous avons deux classes ci-dessous:

```
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;
}
```

et si vous avez juste besoin d'analyser un tableau json, vous pouvez utiliser ce code dans votre analyse:

```
String tweetsJsonArray = "[{.....},{.....}]"
List<Tweet> tweets = gson.fromJson(tweetsJsonArray, new TypeToken<List<Tweet>>()
{}.getType());
```

Deserializer JSON personnalisé avec Gson

Imaginez que vous avez toutes les dates dans toutes les réponses dans un format personnalisé,

par exemple `/Date(1465935152)/` et que vous souhaitez appliquer une règle générale pour désérialiser toutes les dates Json en instances de `Date` Java. Dans ce cas, vous devez implémenter Json Deserializer personnalisé.

Exemple de json:

```
{
  "id": 1,
  "created_on": "Date(1465935152)",
  "updated_on": "Date(1465968945)",
  "name": "Oleksandr"
}
```

Supposons que nous ayons cette classe ci-dessous:

```
class User {
    @SerializedName("id")
    long id;
    @SerializedName("created_on")
    Date createdOn;
    @SerializedName("updated_on")
    Date updatedOn;
    @SerializedName("name")
    String name;
}
```

Désérialiseur personnalisé:

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

Et l'utilisation:

```
Gson gson = new GsonBuilder()
                .registerTypeAdapter(Date.class, new DateDeSerializer())
                .create();
String json = "....";
User user = gson.fromJson(json, User.class);
```

Sérialiser et désérialiser les chaînes Jackson JSON avec les types de date

Cela s'applique également au cas où vous souhaitez rendre la conversion de date de Gson compatible avec Jackson, par exemple.

Jackson a l'habitude de sérialiser Date à "millisecondes depuis l'époque" alors que Gson utilise un format lisible comme `Aug 31, 2016 10:26:17` pour représenter Date. Cela conduit à JsonSyntaxExceptions dans Gson lorsque vous essayez de désérialiser une date au format Jackson.

Pour contourner ce problème, vous pouvez ajouter un sérialiseur personnalisé et un désérialiseur personnalisé:

```
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();
```

Utiliser Gson avec héritage

Gson ne supporte pas l'héritage par défaut.

Disons que nous avons la hiérarchie de classes suivante:

```
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;
    }
}
```

Et maintenant, nous voulons sérialiser une instance de `DerivedClass1` en une chaîne JSON

```
DerivedClass1 derivedClass1 = new DerivedClass1();
derivedClass1.b = 5;
derivedClass1.a = 10;

Gson gson = new Gson();
String derivedClass1Json = gson.toJson(derivedClass1);
```

Maintenant, à un autre endroit, nous recevons cette chaîne json et souhaitons la désérialiser - mais au moment de la compilation, nous savons seulement qu'elle est supposée être une instance de `BaseClass` :

```
BaseClass maybeDerivedClass1 = gson.fromJson(derivedClass1Json, BaseClass.class);
System.out.println(maybeDerivedClass1.getInt());
```

Mais GSON ne sait pas que: `derivedClass1Json` était à l'origine une instance de `DerivedClass1` , donc cela imprimera 10.

Comment résoudre ce problème?

Vous devez créer votre propre `JsonDeserializer` , qui gère de tels cas. La solution n'est pas parfaitement propre, mais je n'ai pas pu en trouver une meilleure.

Tout d'abord, ajoutez le champ suivant à votre classe de base

```
@SerializedName("type")
private String typeName;
```

Et l'initialiser dans le constructeur de la classe de base

```
public BaseClass() {
    typeName = getClass().getName();
}
```

Ajoutez maintenant la classe suivante:

```
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);
 }
}
```

Tout ce qui reste à faire est de tout brancher -

```
GsonBuilder builder = new GsonBuilder();
 builder
 .registerTypeAdapter(BaseClass.class, new JsonDeserializerWithInheritance<BaseClass>());
 Gson gson = builder.create();
```

Et maintenant, en exécutant le code suivant

```
 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());
```

Imprimera 5.

Lire Gson en ligne: https://riptutorial.com/fr/android/topic/4158/gson

Chapitre 139: HttpURLConnection

Syntaxe

- débranchement de vide abstrait ()
- booléen abstrait utilisantProxy ()
- booléen statique getFollowRedirects ()
- setFollowRedirects vide statique (ensemble booléen)
- String getHeaderField (int n)
- String getHeaderFieldKey (int n)
- String getRequestMethod ()
- String getResponseMessage ()
- int getResponseCode ()
- long getHeaderFieldDate (nom de la chaîne, par défaut long)
- boolean getInstanceFollowRedirects ()
- Permission getPermission ()
- InputStream getErrorStream ()
- void setChunkedStreamingMode (int chunklen)
- void setFixedLengthStreamingMode (int contentLength)
- void setFixedLengthStreamingMode (long contentLength)
- void setInstanceFollowRedirects (boolean followRedirects)
- void setRequestMethod (méthode String)

Remarques

`HttpURLConnection` est le client HTTP standard pour Android, utilisé pour envoyer et recevoir des données sur le Web. C'est une implémentation concrète d'URLConnection pour HTTP (RFC 2616).

Examples

Créer un HttpURLConnection

Pour créer un nouveau client HTTP Android `HttpURLConnection` , appelez `openConnection()` sur une instance d'URL. Comme `openConnection()` renvoie une `URLConnection` , vous devez explicitement `URLConnection` la valeur renvoyée.

```
URL url = new URL("http://example.com");
HttpURLConnection connection = (HttpURLConnection) url.openConnection();
// do something with the connection
```

Si vous créez une nouvelle `URL` , vous devez également gérer les exceptions associées à l'analyse des URL.

```
try {
    URL url = new URL("http://example.com");
    HttpURLConnection connection = (HttpURLConnection) url.openConnection();
    // do something with the connection
} catch (MalformedURLException e) {
    e.printStackTrace();
}
```

Une fois que le corps de la réponse a été lu et que la connexion n'est plus requise, la connexion doit être fermée en appelant `disconnect()` .

Voici un exemple:

```
URL url = new URL("http://example.com");
HttpURLConnection connection = (HttpURLConnection) url.openConnection();
try {
    // do something with the connection
} finally {
    connection.disconnect();
}
```

Envoi d'une requête HTTP GET

```
URL url = new URL("http://example.com");
HttpURLConnection connection = (HttpURLConnection) url.openConnection();

try {
    BufferedReader br = new BufferedReader(new
InputStreamReader(connection.getInputStream()));

    // read the input stream
    // in this case, I simply read the first line of the stream
    String line = br.readLine();
    Log.d("HTTP-GET", line);

} finally {
    connection.disconnect();
}
```

Veuillez noter que les exceptions ne sont pas traitées dans l'exemple ci-dessus. Un exemple complet, y compris la gestion des exceptions (trivial), serait:

```
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();
    }
}
```

Lecture du corps d'une requête HTTP GET

```
URL url = new URL("http://example.com");
HttpURLConnection connection = (HttpURLConnection) url.openConnection();

try {
    BufferedReader br = new BufferedReader(new
InputStreamReader(connection.getInputStream()));

    // use a string builder to bufferize the response body
    // read from the input strea.
    StringBuilder sb = new StringBuilder();
    String line;
    while ((line = br.readLine()) != null) {
        sb.append(line).append('\n');
    }

    // use the string builder directly,
    // or convert it into a String
    String body = sb.toString();

    Log.d("HTTP-GET", body);

} finally {
    connection.disconnect();
}
```

Veuillez noter que les exceptions ne sont pas traitées dans l'exemple ci-dessus.

Utiliser HttpURLConnection pour multipart / form-data

Créer une classe personnalisée pour appeler la requête HttpURLConnection multipart / form-data

MultipartUtility.java

```
public class MultipartUtility {
        private final String boundary;
        private static final String LINE_FEED = "\r\n";
        private HttpURLConnection httpConn;
        private String charset;
        private OutputStream outputStream;
        private PrintWriter writer;

        /**
         * This constructor initializes a new HTTP POST request with content type
         * is set to multipart/form-data
         *
         * @param requestURL
```

```
     * @param charset
     * @throws IOException
     */
    public MultipartUtility(String requestURL, String charset)
            throws IOException {
        this.charset = charset;

        // creates a unique boundary based on time stamp
        boundary = "===" + System.currentTimeMillis() + "===";
        URL url = new URL(requestURL);
        httpConn = (HttpURLConnection) url.openConnection();
        httpConn.setUseCaches(false);
        httpConn.setDoOutput(true);    // indicates POST method
        httpConn.setDoInput(true);
        httpConn.setRequestProperty("Content-Type",
                "multipart/form-data; boundary=" + boundary);
        outputStream = httpConn.getOutputStream();
        writer = new PrintWriter(new OutputStreamWriter(outputStream, charset),
                true);
    }

    /**
     * Adds a form field to the request
     *
     * @param name  field name
     * @param value field value
     */
    public void addFormField(String name, String value) {
        writer.append("--" + boundary).append(LINE_FEED);
        writer.append("Content-Disposition: form-data; name=\"" + name + "\"")
                .append(LINE_FEED);
        writer.append("Content-Type: text/plain; charset=" + charset).append(
                LINE_FEED);
        writer.append(LINE_FEED);
        writer.append(value).append(LINE_FEED);
        writer.flush();
    }

    /**
     * Adds a upload file section to the request
     *
     * @param fieldName  name attribute in <input type="file" name="..." />
     * @param uploadFile a File to be uploaded
     * @throws IOException
     */
    public void addFilePart(String fieldName, File uploadFile)
            throws IOException {
        String fileName = uploadFile.getName();
        writer.append("--" + boundary).append(LINE_FEED);
        writer.append(
                "Content-Disposition: form-data; name=\"" + fieldName
                        + "\"; filename=\"" + fileName + "\"")
                .append(LINE_FEED);
        writer.append(
                "Content-Type: "
                        + URLConnection.guessContentTypeFromName(fileName))
                .append(LINE_FEED);
        writer.append("Content-Transfer-Encoding: binary").append(LINE_FEED);
        writer.append(LINE_FEED);
        writer.flush();
```

```
            FileInputStream inputStream = new FileInputStream(uploadFile);
            byte[] buffer = new byte[4096];
            int bytesRead = -1;
            while ((bytesRead = inputStream.read(buffer)) != -1) {
                outputStream.write(buffer, 0, bytesRead);
            }
            outputStream.flush();
            inputStream.close();
            writer.append(LINE_FEED);
            writer.flush();
        }

        /**
         * Adds a header field to the request.
         *
         * @param name  - name of the header field
         * @param value - value of the header field
         */
        public void addHeaderField(String name, String value) {
            writer.append(name + ": " + value).append(LINE_FEED);
            writer.flush();
        }

        /**
         * Completes the request and receives response from the server.
         *
         * @return a list of Strings as response in case the server returned
         * status OK, otherwise an exception is thrown.
         * @throws IOException
         */
        public List<String> finish() throws IOException {
            List<String> response = new ArrayList<String>();
            writer.append(LINE_FEED).flush();
            writer.append("--" + boundary + "--").append(LINE_FEED);
            writer.close();

            // checks server's status code first
            int status = httpConn.getResponseCode();
            if (status == HttpURLConnection.HTTP_OK) {
                BufferedReader reader = new BufferedReader(new InputStreamReader(
                        httpConn.getInputStream()));
                String line = null;
                while ((line = reader.readLine()) != null) {
                    response.add(line);
                }
                reader.close();
                httpConn.disconnect();
            } else {
                throw new IOException("Server returned non-OK status: " + status);
            }
            return response;
        }
    }
```

Utilisez-le (manière asynchrone)

```
    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;
                }
```

Envoi d'une requête HTTP POST avec des paramètres

Utilisez un HashMap pour stocker les paramètres à envoyer au serveur via les paramètres POST:

```
HashMap<String, String> params;
```

Une fois les `params` HashMap renseignés, créez le StringBuilder qui sera utilisé pour les envoyer au serveur:

```
    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++;
    }
```

Ensuite, créez le HttpURLConnection, ouvrez la connexion et envoyez les paramètres POST:

```
try{
    String url = "http://www.example.com/test.php";
    URL urlObj = new URL(url);
    HttpURLConnection conn = (HttpURLConnection) urlObj.openConnection();
    conn.setDoOutput(true);
    conn.setRequestMethod("POST");
    conn.setRequestProperty("Accept-Charset", "UTF-8");
```

```
    conn.setReadTimeout(10000);
    conn.setConnectTimeout(15000);

    conn.connect();

    String paramsString = sbParams.toString();

    DataOutputStream wr = new DataOutputStream(conn.getOutputStream());
    wr.writeBytes(paramsString);
    wr.flush();
    wr.close();
} catch (IOException e) {
  e.printStackTrace();
}
```

Ensuite, recevez le résultat renvoyé par le serveur:

```
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();
    }
}
```

Fichier de téléchargement (POST) en utilisant HttpURLConnection

Il est souvent nécessaire d'envoyer / télécharger un fichier sur un serveur distant, par exemple une image, une vidéo, un fichier audio ou une sauvegarde de la base de données de l'application sur un serveur privé distant. En supposant que le serveur attend une requête POST avec le contenu, voici un exemple simple de la façon de réaliser cette tâche dans Android.

Les téléchargements de fichiers sont envoyés à l'aide de requêtes POST `multipart/form-data` . C'est très facile à mettre en œuvre:

```
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
}
```

Bien sûr, les exceptions devront être détectées ou déclarées comme étant lancées. Quelques points à noter à propos de ce code:

1. `postTarget` est l'URL de destination du POST; `oauthToken` est le jeton d'authentification; `fileDescription` est la description du fichier, qui est envoyée en tant que valeur de la `description` du champ; `file` est le fichier à envoyer - il est de type `java.io.File` - si vous avez le chemin du fichier, vous pouvez utiliser `new File(filePath)` place.
2. Il définit l'en-tête d' `Authorization` pour une authentification oAuth
3. Il utilise Apache Common `FileUtil` pour lire le fichier dans un tableau d'octets - si vous avez déjà le contenu du fichier dans un tableau d'octets ou d'une autre manière en mémoire, il n'est pas nécessaire de le lire.

Une classe HttpURLConnection polyvalente pour gérer tous les types de requêtes HTTP

La classe suivante peut être utilisée en tant que classe unique capable de gérer les requêtes `GET` , `POST` , `PUT` , `PATCH` et autres:

```
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();
    }
}
```

Usage

Utilisez l'un des constructeurs donnés de la classe selon que vous devez envoyer des données `POST` ou des en-têtes supplémentaires.

La méthode `onComplete()` sera appelée lorsque la récupération de données sera terminée. Les données sont renvoyées sous la forme d'un objet de la classe `APIResponseObject` , qui comporte un code d'état indiquant le code d'état HTTP de la demande et une chaîne contenant la réponse. Vous pouvez analyser cette réponse dans votre classe, à savoir XML ou JSON.

Appelez `execute()` sur l'objet de la classe pour exécuter la demande, comme illustré dans l'exemple suivant:

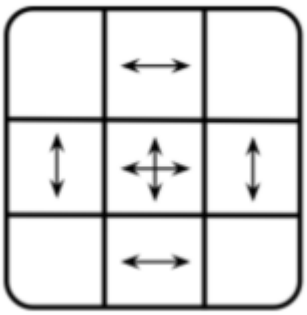
```
class MainClass {
    String url = "https://example.com./api/v1/ex";
    String method = "POST";
    List<Pair<String,String>> postData = new ArrayList<>();

    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();
}
```

Lire HttpURLConnection en ligne: https://riptutorial.com/fr/android/topic/781/httpurlconnection

Chapitre 140: Images 9-Patch

Remarques

Un fichier image à *9 patchs* est un fichier spécialement formaté pour qu'Android sache quelles zones / parties de l'image peuvent ou ne peuvent pas être mises à l'échelle. Il divise votre image en une grille 3x3. Les coins ne sont pas calibrés, les côtés sont mis à l'échelle dans une direction et le centre est redimensionné dans les deux dimensions.



Une image Nine Patch (9-Patch) est une image bitmap avec une bordure large de pixels autour de l'image. Ignorer les 4 pixels dans les coins de l'image. Cette bordure fournit des métadonnées pour le bitmap lui-même. Les limites sont marquées par des lignes noires pleines.

Une image Nine Patch est stockée avec l'extension `.9.png` .

La bordure supérieure indique les zones qui s'étendent horizontalement. La bordure gauche indique les zones qui s'étendent verticalement.

La bordure inférieure indique un remplissage horizontal. La bordure droite indique un remplissage vertical.

Les bordures de remplissage sont généralement utilisées pour déterminer où le texte doit être dessiné.

Il existe un excellent outil fourni par Google qui simplifie *grandement* la création de ces fichiers.

Situé dans le SDK Android: `android-sdk\tools\lib\draw9patch.jar`

Examples

Coins arrondis basiques

La clé pour étirer correctement est dans le bord supérieur et gauche.

La bordure supérieure contrôle l'étirement horizontal et la bordure gauche contrôle l'étirement

vertical.

Cet exemple crée des coins arrondis adaptés à un pain grillé.

Les parties de l'image situées sous la *bordure supérieure* et à droite de la *bordure gauche* se développent pour remplir tout l'espace inutilisé.

Cet exemple s'étendra à toutes les combinaisons de tailles, comme indiqué ci-dessous:

Spinner de base

Le `Spinner` peut être réaménagé selon vos propres exigences de style en utilisant un Nine Patch.

A titre d'exemple, voir ce neuf patch:

Comme vous pouvez le voir, il y a 3 zones d'étirement extrêmement petites.

La bordure supérieure ne comporte plus que l'icône marquée. Cela indique que je souhaite que le côté gauche (transparence totale) du dessinable remplisse la vue `Spinner` jusqu'à ce que l'icône soit atteinte.

La bordure gauche comporte des segments transparents marqués en haut et en bas de l'icône. Cela indique que le haut et le bas vont s'étendre à la taille de la vue `Spinner` . Cela laissera l'icône elle-même centrée verticalement.

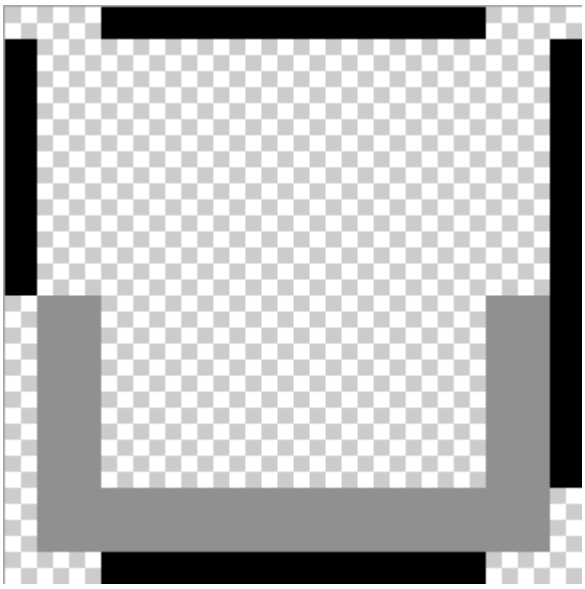
Utilisation de l'image sans métadonnées Nine Patch:

Utilisation de l'image avec les métadonnées Nine Patch:

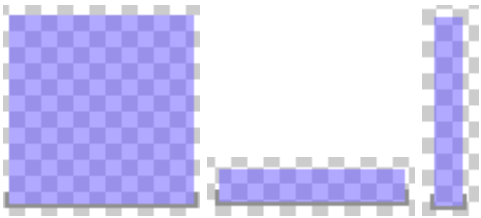
Lignes de remplissage facultatives

Les images à neuf patch permettent une définition facultative des lignes de remplissage dans l'image. Les lignes de remplissage sont les lignes à droite et en bas.

Si une vue définit l'arrière-plan de l'image à 9 patchs, les lignes de remplissage permettent de définir l'espace pour le contenu de la vue (par exemple, l'entrée de texte dans un `EditText`). Si les lignes de remplissage ne sont pas définies, les lignes gauche et supérieure sont utilisées à la place.



La zone de contenu de l'image étirée ressemble alors à ceci:



Lire Images 9-Patch en ligne: https://riptutorial.com/fr/android/topic/461/images-9-patch

Chapitre 141: ImageView

Introduction

ImageView (`android.widget.ImageView`) est une vue permettant d'afficher et de manipuler des ressources d'image, telles que Drawables et Bitmaps.

Certains effets, abordés dans cette rubrique, peuvent être appliqués à l'image. La source d'image peut être définie dans un fichier XML (dossier de `layout`) ou par programmation dans le code Java.

Syntaxe

- La méthode `setImageResource(int resId)` définit un dessin comme contenu de cette `ImageView` .
- **Utilisation:** `imageView.setImageResource(R.drawable.anyImage)`

Paramètres

Paramètre	La description
`resId`	votre nom de fichier image dans le dossier `res` (généralement dans `drawable` dossier `drawable`)

Examples

Définir une ressource image

```
<ImageView
 android:id="@+id/imgExample"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 ...
 />
```

définir un dessin comme contenu de `ImageView` utilisant l'attribut XML:

```
android:src="@drawable/android2"
```

définir un dessinable par programme:

```
 ImageView imgExample = (ImageView) findViewById(R.id.imgExample);
 imgExample.setImageResource(R.drawable.android2);
```

Définir alpha

"alpha" est utilisé pour spécifier l'opacité d'une image.

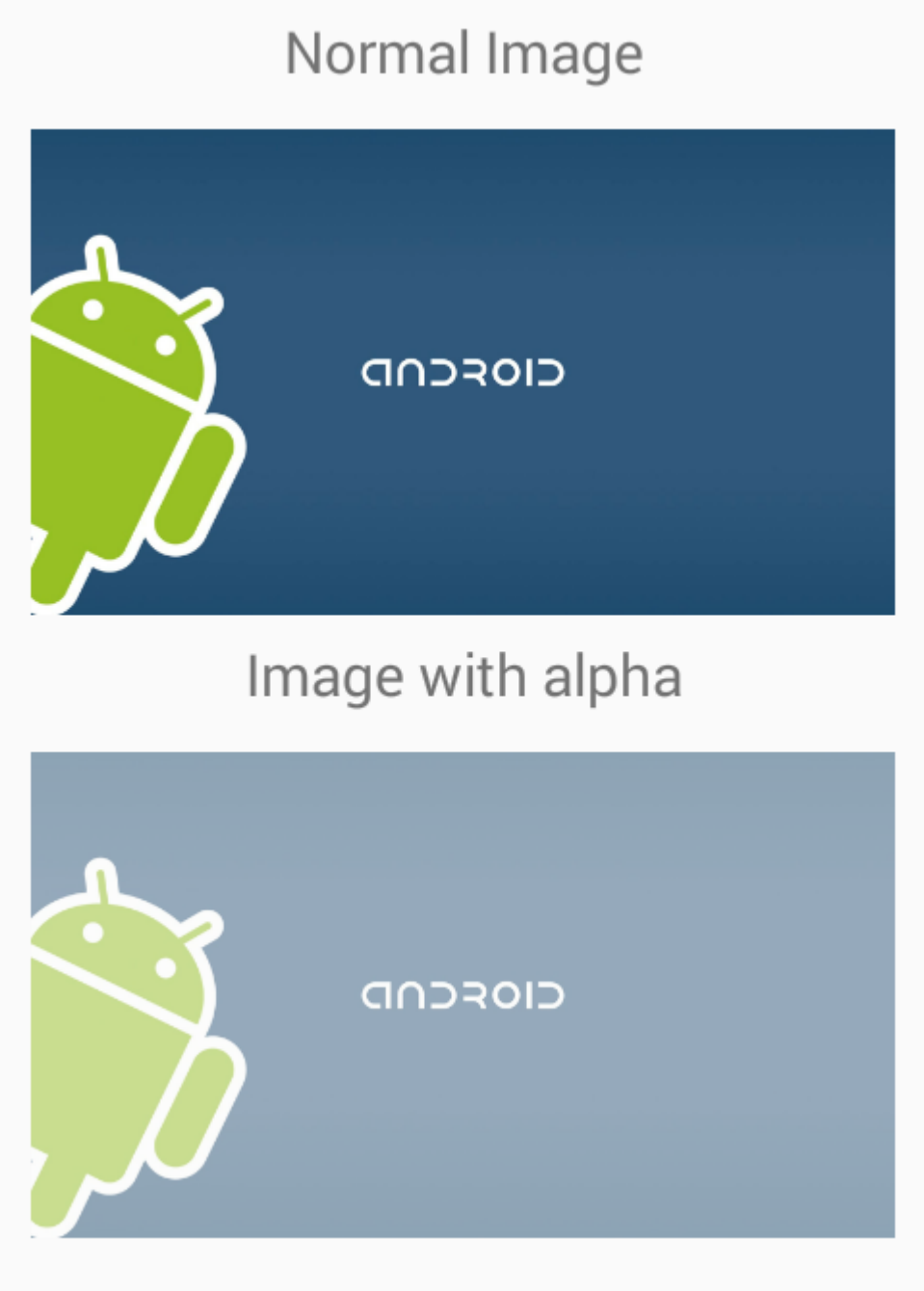
définir alpha en utilisant l'attribut XML:

```
android:alpha="0.5"
```

Remarque: prend la valeur flottante de 0 (transparent) à 1 (entièrement visible)

définir alpha par programmation:

```
imgExample.setAlpha(0.5f);
```



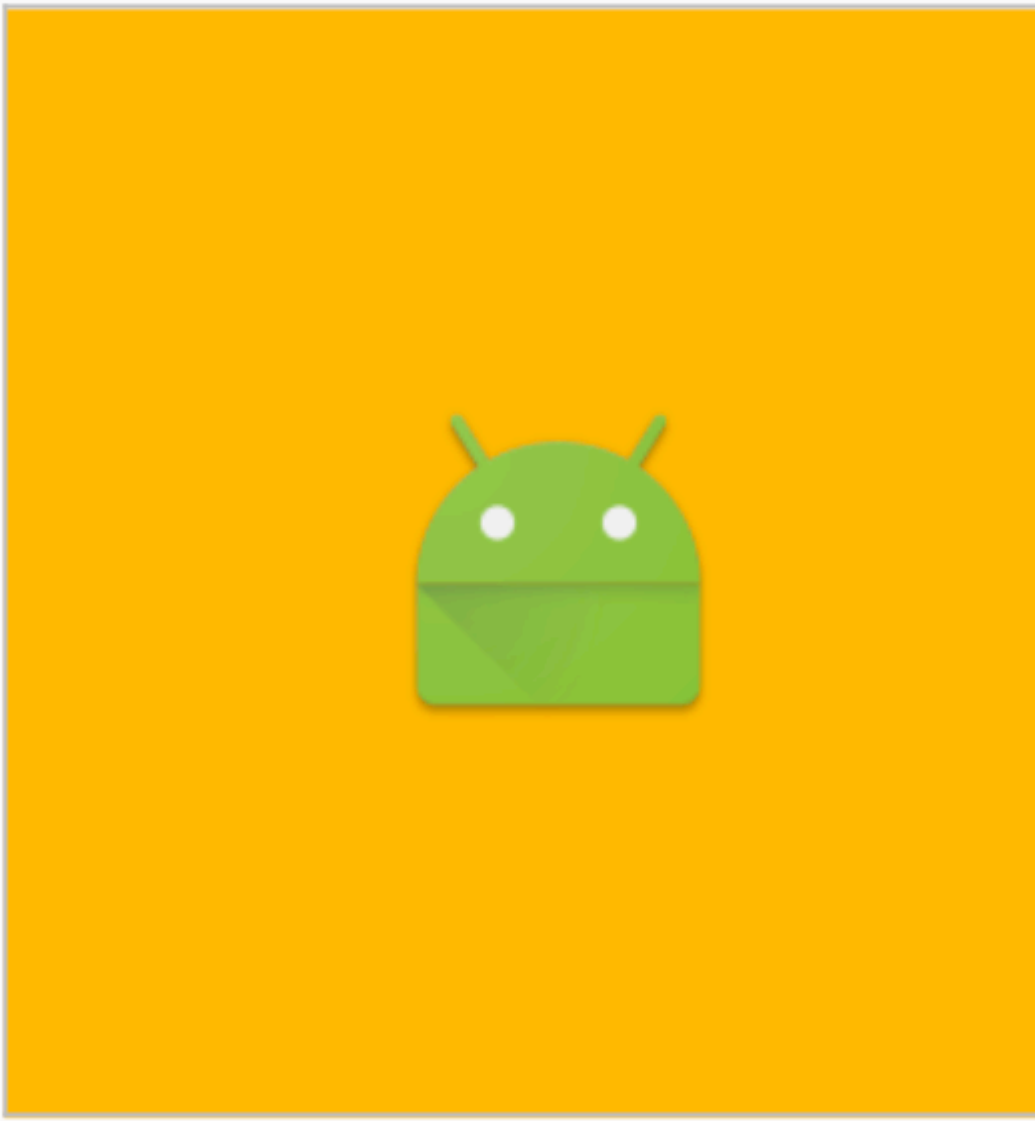
ImageView ScaleType - Centre

L'image contenue dans ImageView peut ne pas correspondre exactement à la taille donnée au conteneur. Dans ce cas, le framework vous permet de redimensionner l'image de plusieurs manières.

Centre

```
    <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"/>
```

Cela ne redimensionnera pas l'image et la centrera dans le conteneur *(Orange = conteneur)*

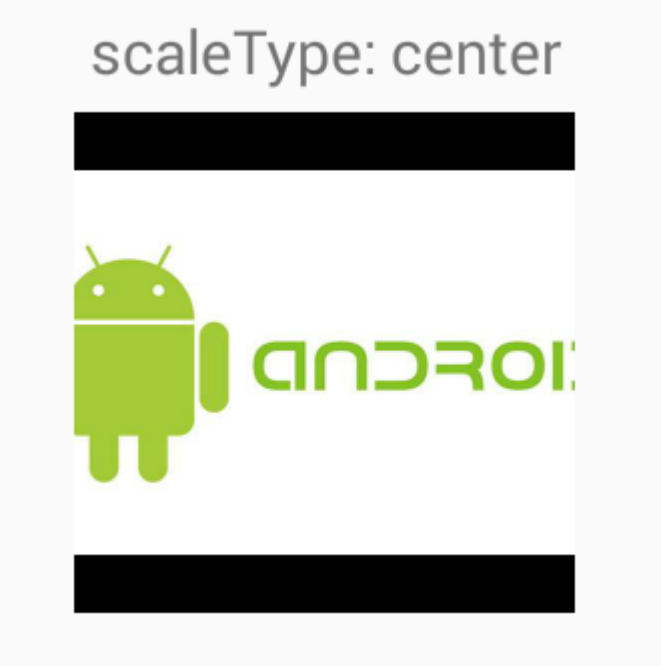


carré qui a un arrière-plan noir et nous voulons afficher un dessin rectangulaire en fond blanc dans `ImageView` .

```
<ImageView
 android:id="@+id/imgExample"
 android:layout_width="200dp"
 android:layout_height="200dp"
 android:background="#000"
 android:src="@drawable/android2"
 android:scaleType="..."/>
```

scaleType doit être l'une des valeurs suivantes:

1. `center` : centrez l'image dans la vue, mais n'effectuez aucune mise à l'échelle.



2. `centerCrop` : redimensionne l'image uniformément (conserve le format de l'image) pour que les dimensions (largeur et hauteur) de l'image soient égales ou supérieures à la dimension correspondante de la vue (moins le remplissage). L'image est ensuite centrée dans la vue.

3. `centerInside` : redimensionne l'image uniformément (conserve le format de l'image) afin que les dimensions (largeur et hauteur) de l'image soient égales ou inférieures à la dimension correspondante de la vue (moins le remplissage). L'image est ensuite centrée dans la vue.

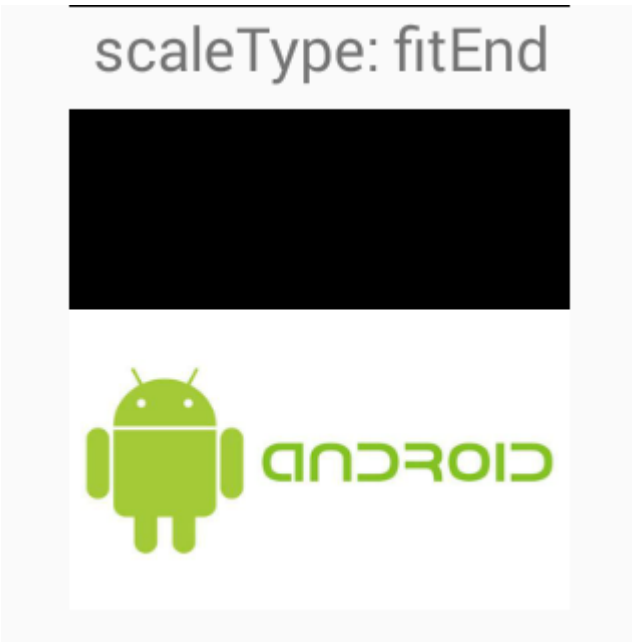
4. `matrix` : mettre à l'échelle en utilisant la matrice d'image lors du dessin.

5. `fitXY` : redimensionne l'image en utilisant FILL .

6. `fitStart` : redimensionne l'image à l'aide de START .

7. `fitCenter` : redimensionne l'image à l'aide de CENTER .

8. `fitEnd` : redimensionne l'image en utilisant END .



Mettre la teinte

Définissez une couleur de teinte pour l'image. Par défaut, la teinte se fondra en utilisant le mode `SRC_ATOP` .

définir la teinte en utilisant l'attribut XML:

```
android:tint="#009c38"
```

Note: Doit être une valeur de couleur, sous la forme `"#rgb"` , `"#argb"` , `"#rrggbb"` ou `"#aarrggbb"` .

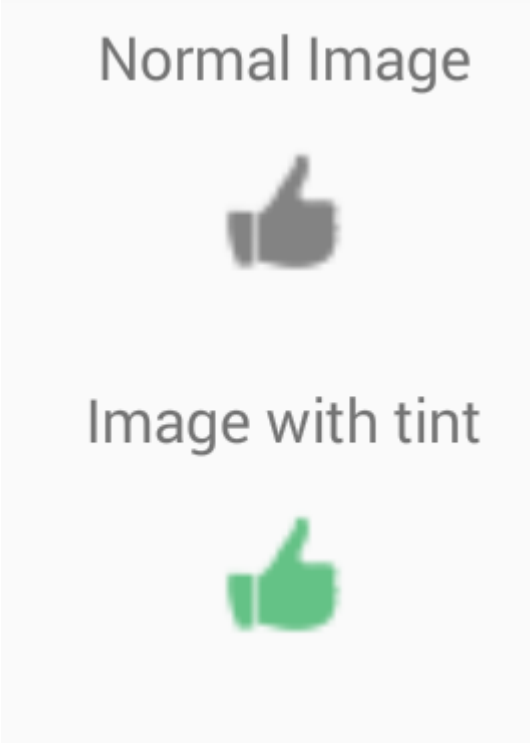
définir la teinte par programmation:

```
imgExample.setColorFilter(Color.argb(255, 0, 156, 38));
```

et vous pouvez effacer ce filtre de couleur:

```
imgExample.clearColorFilter();
```

Exemple:



MLRoundedImageView.java

Copiez et collez la classe suivante dans votre package:

```
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;
    }
}
```

Utilisez cette classe en XML avec le nom du package au lieu de `ImageView`

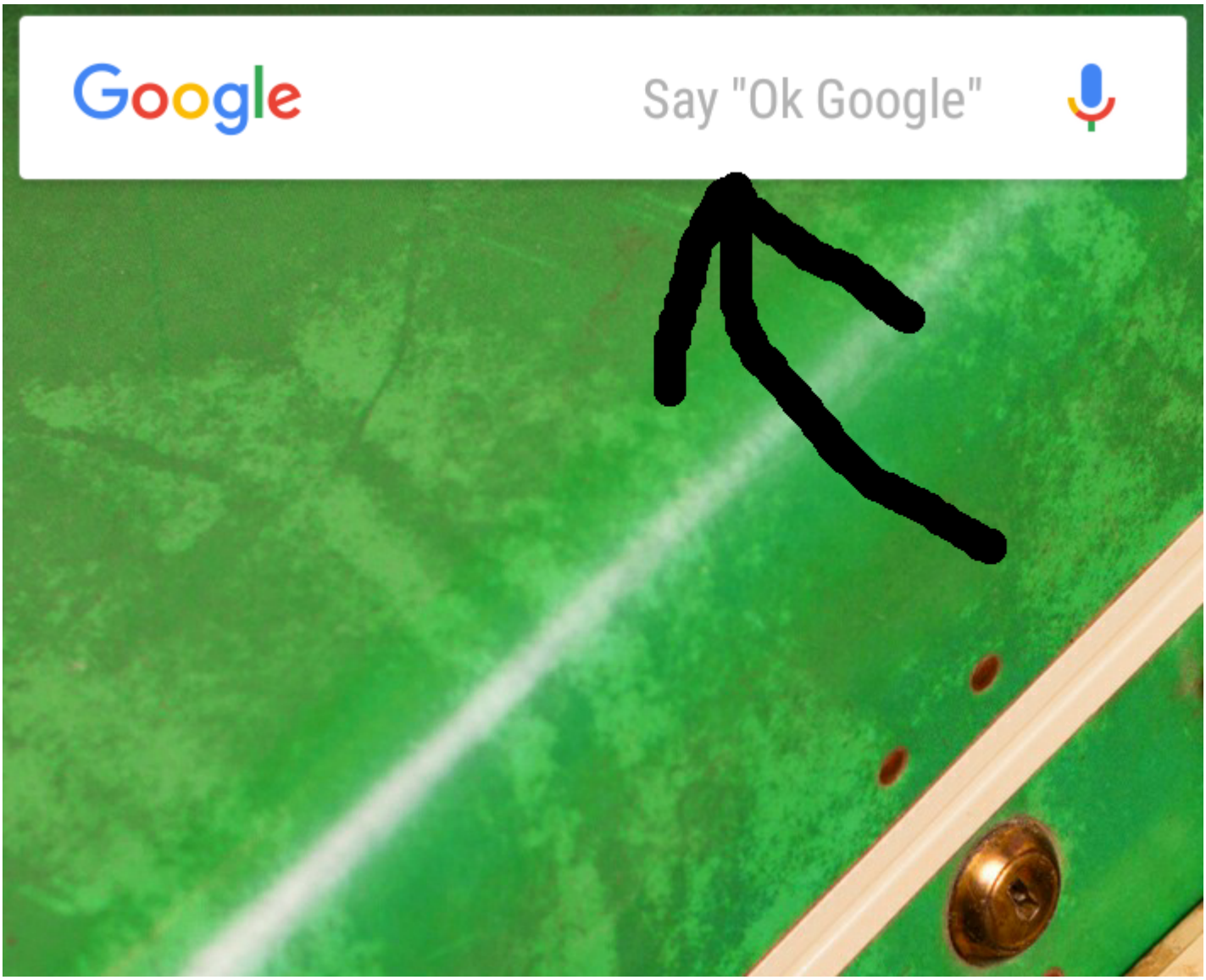
```
<com.androidbuts.example.MLRoundedImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@mipmap/ic_launcher" />
```

Lire ImageView en ligne: https://riptutorial.com/fr/android/topic/4709/imageview

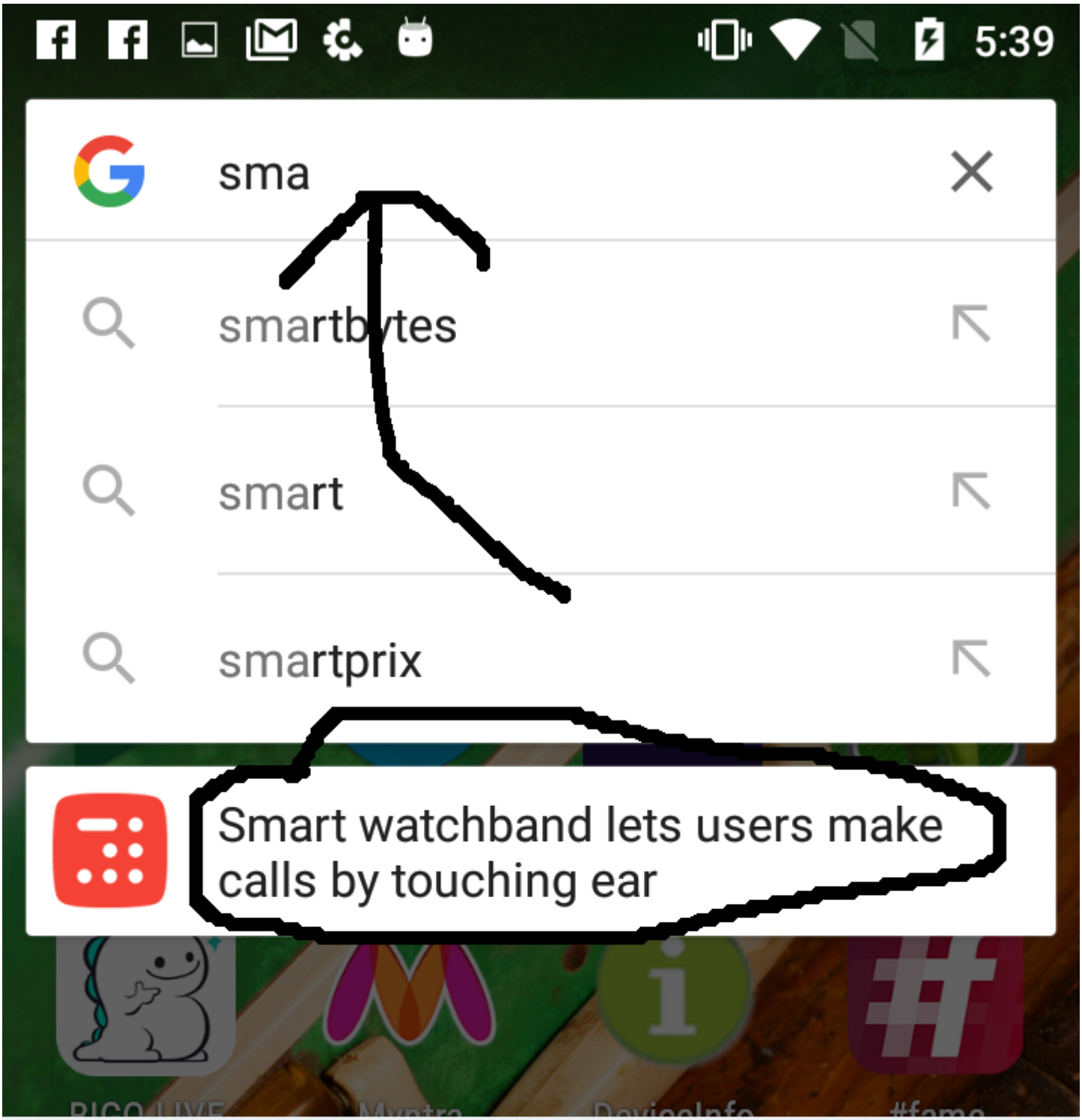
Chapitre 142: Indexation des applications Firebase

Remarques

- Lorsque vous choisissez d’implémenter l’indexation des applications, vous pouvez trouver de nombreux blogs, des documents qui peuvent vous induire en erreur, dans ce cas, je vous suggère de vous en tenir aux documents officiels fournis par Firebase-Google. Même si vous souhaitez utiliser un tiers pour ce faire, essayez d'abord de suivre cette documentation car cela vous donnera une idée claire de la manière dont les choses fonctionnent.
- Google prendra environ 24 heures pour indexer votre contenu. Alors soyez patient. Vous pouvez faire des tests pour que tout se passe bien de votre côté.
- Le premier exemple vous permet de prendre en charge l'URL HTTP de votre site Web pour la rediriger dans votre application. Cela fonctionnera, par exemple, vous avez recherché une requête dans la recherche google, les résultats montrent une des URL de votre site Web, dont les liens d'application sont présents dans votre application qui est déjà installée. En cliquant sur cette URL, il vous redirigera directement dans votre écran d'application correspondant à ce résultat de recherche. C'est ce que j'ai découvert pour cela.
- L'ajout de l'API AppIndexing indexe votre contenu et est utilisé dans les complétions automatiques dans la barre de recherche Google. Prenons l'exemple de l'application InShorts pour chaque page, il y a un titre et une petite description. Après avoir lu 2 ou 3 titres, fermez l'application et passez à Google SearchBar.



Essayez d'entrer le titre que vous venez de parcourir, vous obtiendrez une suggestion de page d'application avec ce titre comme titre. Ceci est différent des suggestions App que vous obtenez lors de la recherche d'applications. Cela se produit car vous avez écrit le code API AppIndexing pour cette page particulière et le titre est identique à celui que vous avez initialisé dans `onCreate()` .


5:39
sma
smartbytes
smart
smartprix
Smart watchband lets users make calls by touching ear

Spécifiez l'action d'intention ACTION_VIEW afin que le filtre d'intention soit accessible à partir de Google Search.

<data> Ajoutez un ou plusieurs tags, chaque tag représentant un format d'URI correspondant à l'activité. Au minimum, la balise doit inclure l'attribut android: scheme. Vous pouvez ajouter des attributs supplémentaires pour affiner davantage le type d'URI que l'activité accepte. Par exemple, vous pouvez avoir plusieurs activités qui acceptent des URI similaires, mais qui diffèrent simplement en fonction du nom du chemin. Dans ce cas, utilisez l'attribut android: path ou ses variantes (pathPattern ou pathPrefix) pour différencier l'activité que le système doit ouvrir pour différents chemins URI.

<catégorie> Inclure la catégorie BROWABLE. La catégorie BROWABLE est nécessaire pour que le filtre d'intention soit accessible à partir d'un navigateur Web. Sans cela, cliquer sur un lien dans un navigateur ne peut pas être résolu dans votre application. La catégorie DEFAULT est facultative, mais recommandée. Sans cette catégorie, l'activité peut être démarrée uniquement avec une intention explicite, en utilisant le nom du composant de votre application.

Étape 4: - Gérer les URL entrantes

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       setContentView(R.layout.activity_schedule);
       onNewIntent(getIntent());
   }

    protected void onNewIntent(Intent intent) {
    String action = intent.getAction();
    Uri data = intent.getData();
    if (Intent.ACTION_VIEW.equals(action) && data != null) {
       articleId = data.getLastPathSegment();
       TextView linkText = (TextView)findViewById(R.id.link);
       linkText.setText(data.toString());
    }
```

}

Étape 5: - Vous pouvez tester cela en utilisant la commande Android Debug Bridge ou les configurations de studio. Commande Adb: - Lancez votre application puis lancez cette commande: -

```
adb shell am start -a android.intent.action.VIEW -d "{URL}" < package name >
```

Configurations d'Android Studio: - Studio **Android> Build> Modifier la configuration> Options de lancement> Sélectionner l'URL> puis saisissez votre URL ici> Appliquer** et tester.Lancez votre application si la fenêtre «Exécuter» affiche une erreur. Les applinks mentionnés dans le manifeste autrement s'exécuteront correctement, et redirigeront vers la page mentionnant votre URL si spécifié.

Ajouter l'API AppIndexing

Pour l'ajouter au projet, vous pouvez trouver facilement un document officiel, mais dans cet exemple, je vais mettre en évidence certains des domaines clés à prendre en compte.

Étape 1: - Ajouter un service Google

```
dependencies {
      ...
      compile 'com.google.android.gms:play-services-appindexing:9.4.0'
      ...
    }
```

Étape 2: - Importer des classes

```
import com.google.android.gms.appindexing.Action;
import com.google.android.gms.appindexing.AppIndex;
import com.google.android.gms.common.api.GoogleApiClient;
```

Étape 3: - Ajouter des appels d'API d'indexation d'application

```
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();
}
```

Pour tester cela, suivez simplement l'étape 4 de la section Remarques ci-dessous.

Lire Indexation des applications Firebase en ligne:
https://riptutorial.com/fr/android/topic/5957/indexation-des-applications-firebase

Chapitre 143: Installation d'applications avec ADB

Examples

Installer une application

Écrivez la commande suivante dans votre terminal:

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

Notez que vous devez transmettre un fichier qui se trouve sur votre ordinateur et non sur votre appareil.

Si vous ajoutez `-r` à la fin, tous les apks en conflit existants seront écrasés. Sinon, la commande se fermera avec une erreur.

`-g` accordera immédiatement toutes les autorisations d'exécution.

`-d` permet de rétrograder le code de la version (uniquement applicable sur les paquets pouvant être débogués).

Utilisez `-s` pour installer l'application sur la carte SD externe.

`-t` permettra d'utiliser des applications de test.

Désinstaller une application

Écrivez la commande suivante dans votre terminal pour désinstaller une application avec un nom de paquet fourni:

```
adb uninstall <packagename>
```

Installer tous les fichiers apk dans le répertoire

Les fenêtres :

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

Linux:

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

Lire Installation d'applications avec ADB en ligne:
https://riptutorial.com/fr/android/topic/5301/installation-d-applications-avec-adb

Chapitre 144: Intégration de Google Signin sur Android

Introduction

Ce sujet est basé sur la façon d'intégrer la connexion google, sur les applications Android

Examples

Intégration de Google Auth dans votre projet. (Obtenir un fichier de configuration)

Commencez par obtenir le fichier de configuration pour la connexion à partir de

Ouvrir le lien ci-dessous

[https://developers.google.com/identity/sign-in/android/start-integrating]née1]

cliquez sur get Un fichier de configuration

- Entrez le nom de l'application et le nom du package, puis cliquez sur choisir et configurer les services
- fournir SHA1 Activer Google SIGNIN et générer des fichiers de configuration

Téléchargez le fichier de configuration et placez le fichier dans le dossier app / de votre projet

1. Ajoutez la dépendance à votre build.gradle au niveau du projet:

   classpath 'com.google.gms: google-services: 3.0.0'

2. Ajoutez le plugin à votre build.gradle au niveau de l'application: (en bas)

   appliquer le plugin: 'com.google.gms.google-services'

3. ajoutez cette dépendance à votre fichier d'application graduel

   dépendances {compiler 'com.google.android.gms: play-services-auth: 9.8.0'}

Mise en œuvre du code Google SignIn

- Dans la méthode onCreate de votre activité de connexion, configurez Google Sign-In pour demander les données utilisateur requises par votre application.

```
 GoogleSignInOptions gso = new
GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
        .requestEmail()
```

```
        .build();
```

- Créez un objet GoogleApiClient avec accès à l'API Google Sign-In et aux options que vous avez spécifiées.

```
mGoogleApiClient = new GoogleApiClient.Builder(this)
        .enableAutoManage(this /* FragmentActivity */, this /* OnConnectionFailedListener */)
        .addApi(Auth.GOOGLE_SIGN_IN_API, gso)
        .build();
```

- Maintenant, lorsque l'utilisateur clique sur le bouton Google Signin, appelez cette fonction.

```
private void signIn() {
Intent signInIntent = Auth.GoogleSignInApi.getSignInIntent(mGoogleApiClient);
startActivityForResult(signInIntent, RC_SIGN_IN);
}
```

- implémenter OnActivityResult pour obtenir la réponse.

```
@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);
    }
}
```

- Dernière étape Gérer le résultat et obtenir des données utilisateur

```
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);
}
}
```

Lire Intégration de Google Signin sur Android en ligne:
https://riptutorial.com/fr/android/topic/9960/integration-de-google-signin-sur-android

Chapitre 145: Intégration de la passerelle Android Paypal

Remarques

Paypal nous fournit sa propre bibliothèque pour le paiement, il est donc maintenant beaucoup plus sécurisé et facile à implémenter dans notre application. Vous trouverez ci-dessous l'étape importante à suivre.

Examples

Installez paypal dans votre code Android

1) Commencez par parcourir le site Web du développeur Paypal et créez une application.

2) Maintenant, ouvrez votre fichier manifeste et donnez les autorisations ci-dessous

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

3) Et certaines activités et services requis-

```
 <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) Ouvrez votre classe d'activité et définissez la configuration de votre application.

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

5) Maintenant, définissez l'ID client à partir du compte de développeur Paypal - private static final String CONFIG_CLIENT_ID = "METTEZ VOTRE ID CLIENT"; 6) Dans la méthode onCreate, appelez le service Paypal - Intention intentionnelle = nouvelle intention (ceci, PayPalService.class); intent.putExtra (PayPalService.EXTRA_PAYPAL_CONFIGURATION, config); startService (intention);

7) Maintenant, vous êtes prêt à effectuer un paiement juste sur le bouton appuyez sur appeler l'activité de paiement-

```
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) Et enfin, sur onActivityResult, obtenez la réponse de paiement-

```
 @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.");
                   }
               }


           }
```

Lire Intégration de la passerelle Android Paypal en ligne:
https://riptutorial.com/fr/android/topic/5895/integration-de-la-passerelle-android-paypal

Chapitre 146: Intégrer Google Connexion

Syntaxe

- newInstance () - Pour créer une instance unique de Google Helper
- initGoogleSignIn () - Pour initialiser la connexion Google
- getGoogleAccountDetails () - Pour vous connecter
- signOut () - Pour déconnecter l'utilisateur
- getGoogleClient () - Pour utiliser GoogleApiClient

Paramètres

Paramètre	Détail
MARQUE	Une chaîne utilisée lors de la connexion
GoogleSignInHelper	Une référence statique pour assistant
AppCompatActivité	Une référence d'activité
GoogleApiClient	Une référence de GoogleAPIClient
RC_SIGN_IN	Un entier représente la constante du résultat de l'activité
isLoggingOut	Un booléen pour vérifier si la tâche de déconnexion est en cours ou non

Examples

Connexion à Google avec classe d'assistance

Ajoutez ci-dessous à votre `build.gradle` de la balise `android` :

```
// Apply plug-in to app.
apply plugin: 'com.google.gms.google-services'
```

Ajoutez la classe d'assistance ci-dessous à votre paquet util:

```
/**
 * Created by Andy
 */
public class GoogleSignInHelper implements GoogleApiClient.OnConnectionFailedListener,
        GoogleApiClient.ConnectionCallbacks {
    private static final String TAG = GoogleSignInHelper.class.getSimpleName();

    private static GoogleSignInHelper googleSignInHelper;
```

```
    private AppCompatActivity mActivity;
    private GoogleApiClient mGoogleApiClient;
    public static final int RC_SIGN_IN = 9001;
    private boolean isLoggingOut = false;

    public static GoogleSignInHelper newInstance(AppCompatActivity mActivity) {
        if (googleSignInHelper == null) {
            googleSignInHelper = new GoogleSignInHelper(mActivity, fireBaseAuthHelper);
        }
        return googleSignInHelper;
    }

    public GoogleSignInHelper(AppCompatActivity mActivity) {
        this.mActivity = mActivity;
        initGoogleSignIn();
    }


    private void initGoogleSignIn() {
        // [START config_sign_in]
        // Configure Google Sign In
        GoogleSignInOptions gso = new
GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
                .requestIdToken(mActivity.getString(R.string.default_web_client_id))
                .requestEmail()
                .build();
        // [END config_sign_in]

        mGoogleApiClient = new GoogleApiClient.Builder(mActivity)
                .enableAutoManage(mActivity /* FragmentActivity */, this /*
OnConnectionFailedListener */)
                .addApi(Auth.GOOGLE_SIGN_IN_API, gso)
                .addConnectionCallbacks(this)
                .build();

    }

    @Override
    public void onConnectionFailed(@NonNull ConnectionResult connectionResult) {
        // An unresolvable error has occurred and Google APIs (including Sign-In) will not
        // be available.
        Log.d(TAG, "onConnectionFailed:" + connectionResult);
        Toast.makeText(mActivity, "Google Play Services error.", Toast.LENGTH_SHORT).show();
    }

    public void getGoogleAccountDetails(GoogleSignInResult result) {
        // Google Sign In was successful, authenticate with FireBase
        GoogleSignInAccount account = result.getSignInAccount();
        // You are now logged into Google
    }
    public void signOut() {

        if (mGoogleApiClient.isConnected()) {

            // Google sign out
            Auth.GoogleSignInApi.signOut(mGoogleApiClient).setResultCallback(
                    new ResultCallback<Status>() {
                        @Override
                        public void onResult(@NonNull Status status) {
                            isLoggingOut = false;
                        }
```

```
                    });
        } else {
            isLoggingOut = true;
        }
    }

    public GoogleApiClient getGoogleClient() {
        return mGoogleApiClient;
    }

    @Override
    public void onConnected(@Nullable Bundle bundle) {
        Log.w(TAG, "onConnected");
        if (isLoggingOut) {
            signOut();
        }
    }

    @Override
    public void onConnectionSuspended(int i) {
        Log.w(TAG, "onConnectionSuspended");
    }
}
```

Ajoutez le code ci-dessous à votre fichier `OnActivityResult` dans le fichier d'activité:

```
 // [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]
```

Lire Intégrer Google Connexion en ligne: https://riptutorial.com/fr/android/topic/2837/integrer-google-connexion

Chapitre 147: Intégrer OpenCV dans Android Studio

Remarques

Les bibliothèques de CV ouvert peuvent être trouvées sur le Web en utilisant un moteur de recherche.

Les **Gotchas** :

- Si vous abaissez votre plate-forme cible sous KitKat, certaines des bibliothèques OpenCV ne fonctionneront plus, en particulier les classes liées à *org.opencv.android.Camera2Renderer* et à d'autres classes associées. Vous pouvez probablement contourner ce problème en supprimant simplement les fichiers .java OpenCV appropriés.
- Si vous élevez votre plate-forme cible à Lollipop ou au-dessus, mon exemple de chargement d'un fichier peut ne pas fonctionner car l'utilisation de chemins d'accès absolus est mal vue. Donc, vous devrez peut-être changer l'exemple pour charger un fichier de la galerie ou ailleurs. Il existe de nombreux exemples qui circulent.

Examples

Instructions

Testé avec AS v1.4.1 mais devrait également fonctionner avec les nouvelles versions.

1. Créez un nouveau projet Android Studio à l'aide de l'assistant de projet (Menu: / Fichier / Nouveau projet):

   - Appelez ça " **cvtest1** "
   - Facteur de forme: **API 19, Android 4.4 (KitKat)**
   - **Activité vide** nommée **MainActivity**

   Vous devriez avoir un répertoire *cvtest1* où ce projet est stocké. (la barre de titre du studio Android vous indique où cvtest1 se trouve lorsque vous ouvrez le projet)

2. Vérifiez que votre application s'exécute correctement. Essayez de changer quelque chose comme le texte "Hello World" pour confirmer que le cycle de construction / test est correct pour vous. (Je teste avec un émulateur d'un périphérique API 19).

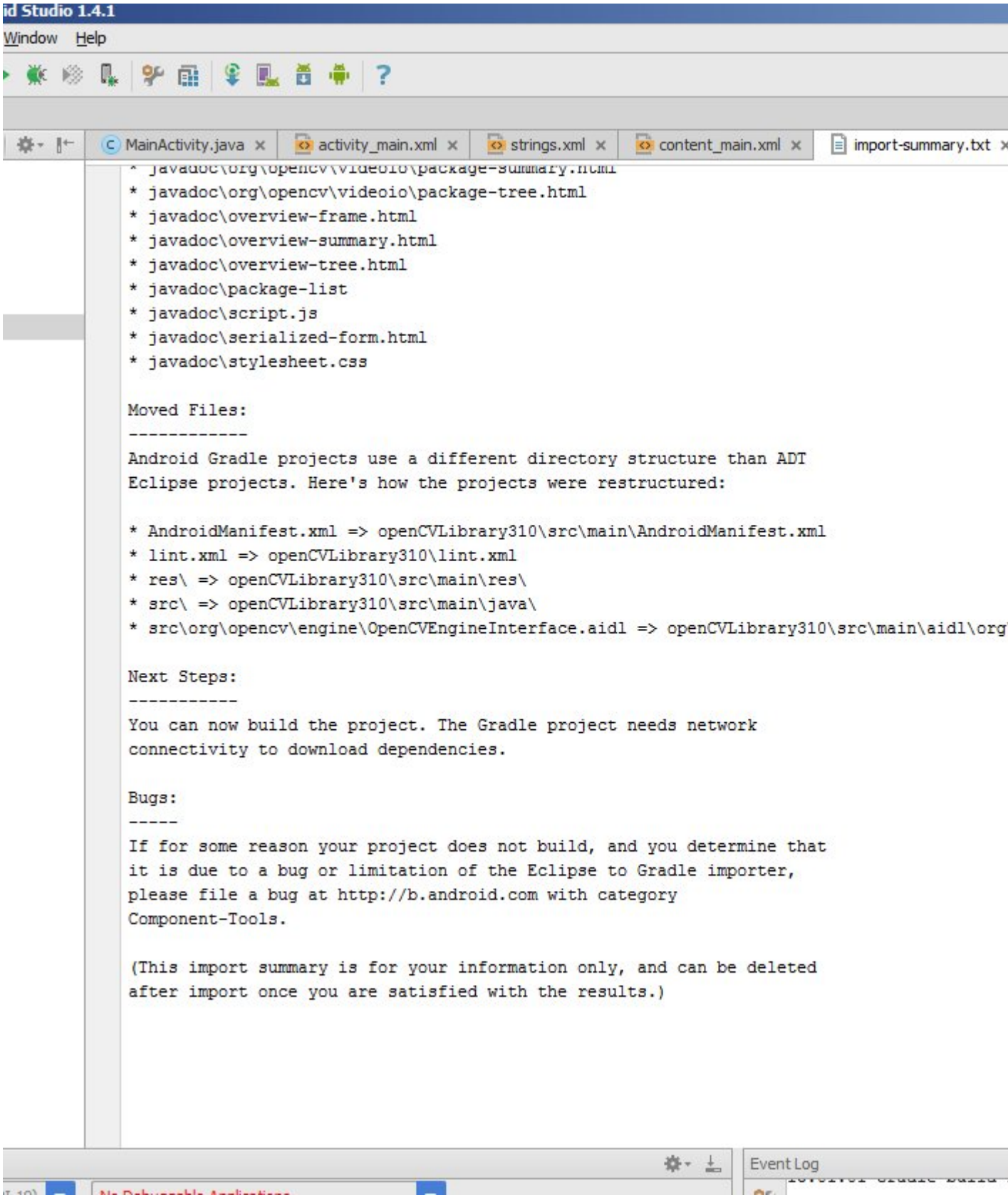
3. Téléchargez le package OpenCV pour Android v3.1.0 et décompressez-le dans un répertoire temporaire quelque part. (Assurez-vous que c'est le paquet spécifiquement pour Android et pas seulement le paquetage OpenCV pour Java.) J'appellerai ce répertoire " **unzip-dir** " Sous **unzip-dir,** vous devriez avoir un **répertoire sdk / native / libs** avec des sous-répertoires commençant par des choses comme **bras** ..., **mips** ... et **x86** ... (un pour

chaque type "d'architecture" Android s'exécute)

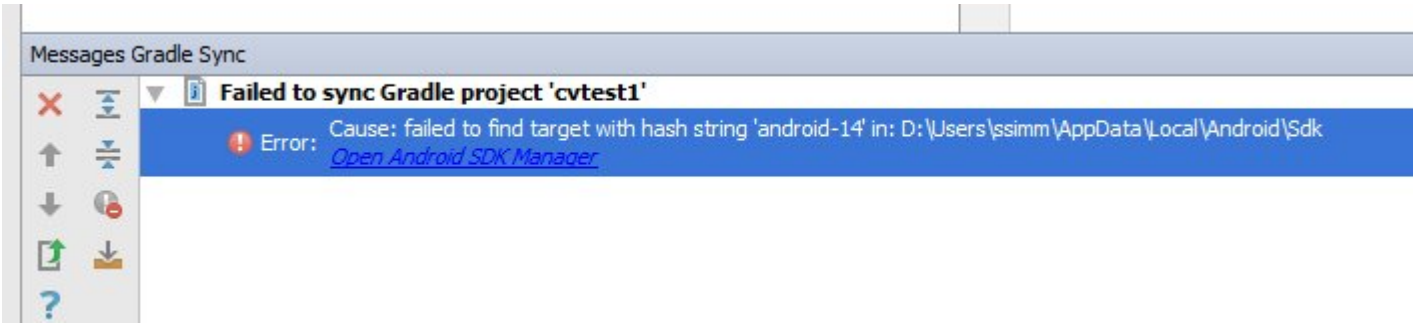
4. Depuis Android Studio, importez OpenCV dans votre projet en tant que module: **Menu: / File / New / Import_Module** :

    - Répertoire-source: **{unzip-dir} / sdk / java**
    - Nom du module: Android studio remplit automatiquement ce champ avec **openCVLibrary310** (le nom exact n'a probablement pas d'importance mais nous allons y aller).
    - Cliquez sur **suivant** . Vous obtenez un écran avec trois cases à cocher et des questions sur les pots, les bibliothèques et les options d'importation. Tous les trois doivent être vérifiés. Cliquez sur **Terminer.**

    Android Studio commence à importer le module et vous obtenez un fichier **import-summary.txt** contenant une liste de ce qui n'a pas été importé (principalement des fichiers javadoc) et d'autres informations.



Mais vous obtenez également un message d'erreur indiquant que vous n'avez **pas réussi à trouver la cible avec la chaîne de hachage "android-14" ....** Cela se produit parce que le fichier build.gradle du fichier zip OpenCV que vous avez téléchargé dit de compiler à l'aide d'Android API version 14, que vous n'avez pas par défaut avec Android Studio v1.4.1.

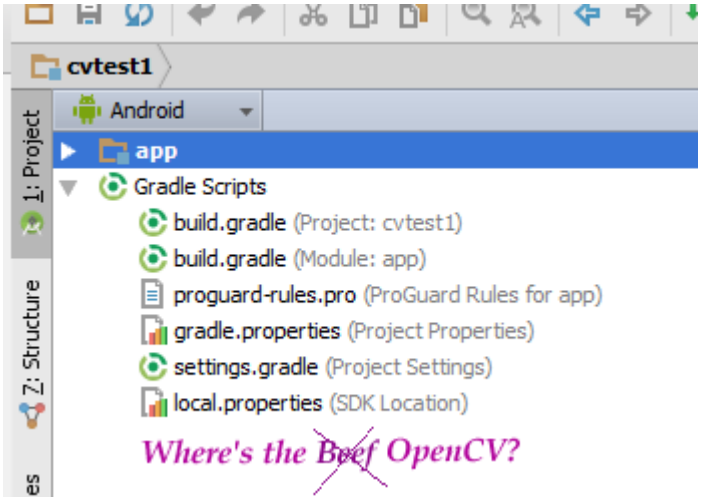


5. Ouvrez le dialogue de structure du projet (**Menu: / File / Project_Structure**). Sélectionnez le module "app", cliquez sur l'onglet **Dépendances** et ajoutez **: openCVLibrary310 en** tant que dépendance de module. Lorsque vous sélectionnez **Add / Module_Dependency,** il doit apparaître dans la liste des modules que vous pouvez ajouter. Il apparaîtra désormais comme une dépendance, mais vous obtiendrez quelques erreurs de **non-recherche-Android-14** dans le journal des événements.

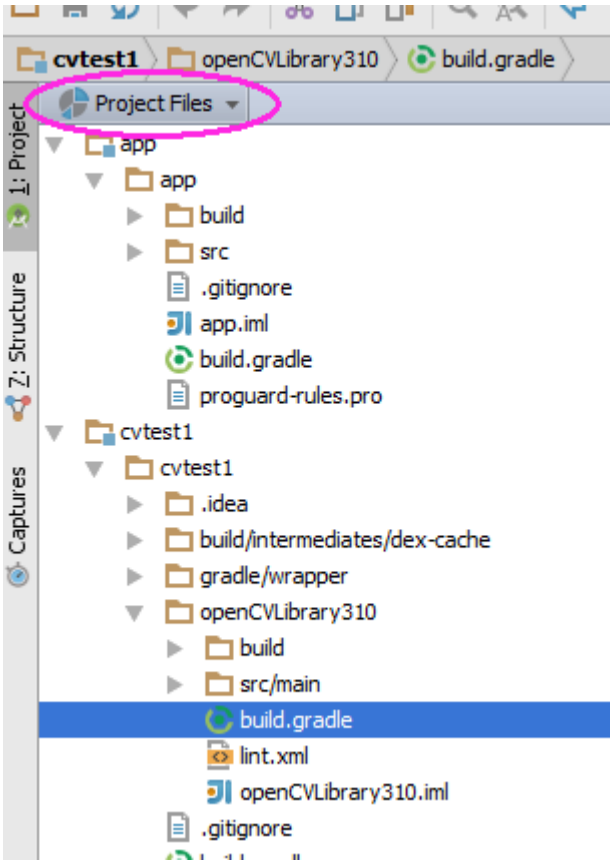
6. Recherchez dans le fichier **build.gradle** votre module d'application. Il existe plusieurs fichiers build.gradle dans un projet Android. Celui que vous voulez se trouve dans le **répertoire cvtest1 / app** et dans la vue du projet, il ressemble à **build.gradle (Module: app)** . Notez les valeurs de ces quatre champs:

    - compileSDKVersion (le mien dit 23)
    - buildToolsVersion (le mien dit 23.0.2)
    - minSdkVersion (le mien dit 19)
    - targetSdkVersion (le mien dit 23)

7. Votre projet a maintenant un **répertoire cvtest1 / OpenCVLibrary310** mais il n'est pas visible dans la vue du projet:

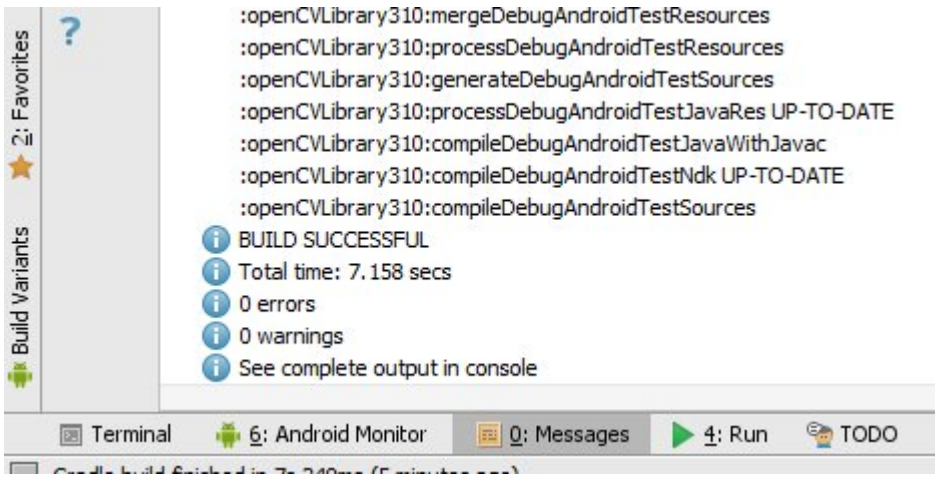


Utilisez un autre outil, tel qu'un gestionnaire de fichiers, et accédez à ce répertoire. Vous pouvez également basculer la vue du projet d' **Android** vers **Fichiers de projet** et vous pouvez trouver ce répertoire comme indiqué dans cette capture d'écran:



A l'intérieur, il y a un autre fichier **build.gradle** (il est mis en évidence dans la capture d'écran ci-dessus). Mettez à jour ce fichier avec les quatre valeurs de l'étape 6.

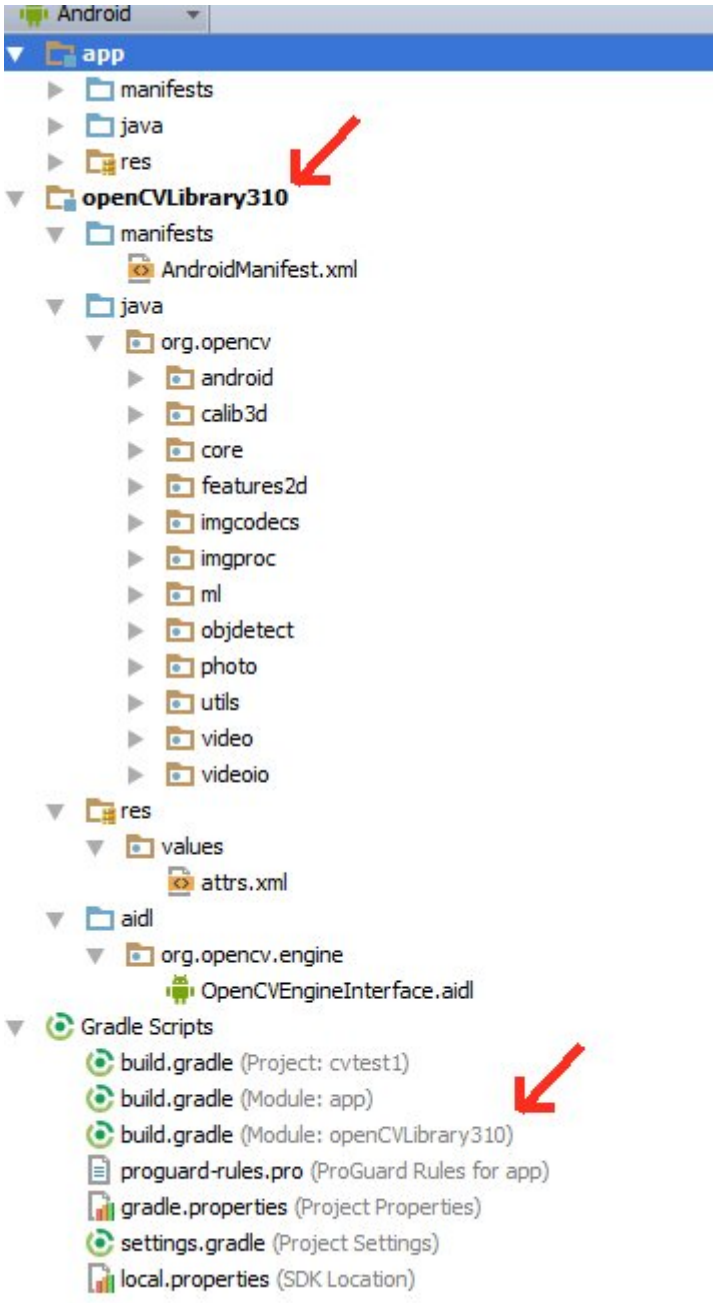
8. Resynchronisez votre projet, puis nettoyez / reconstruisez-le. (**Menu: / Build / Clean_Project**) Il devrait nettoyer et construire sans erreurs et vous devriez voir de nombreuses références à **: openCVLibrary310** dans l'écran **0: Messages** .



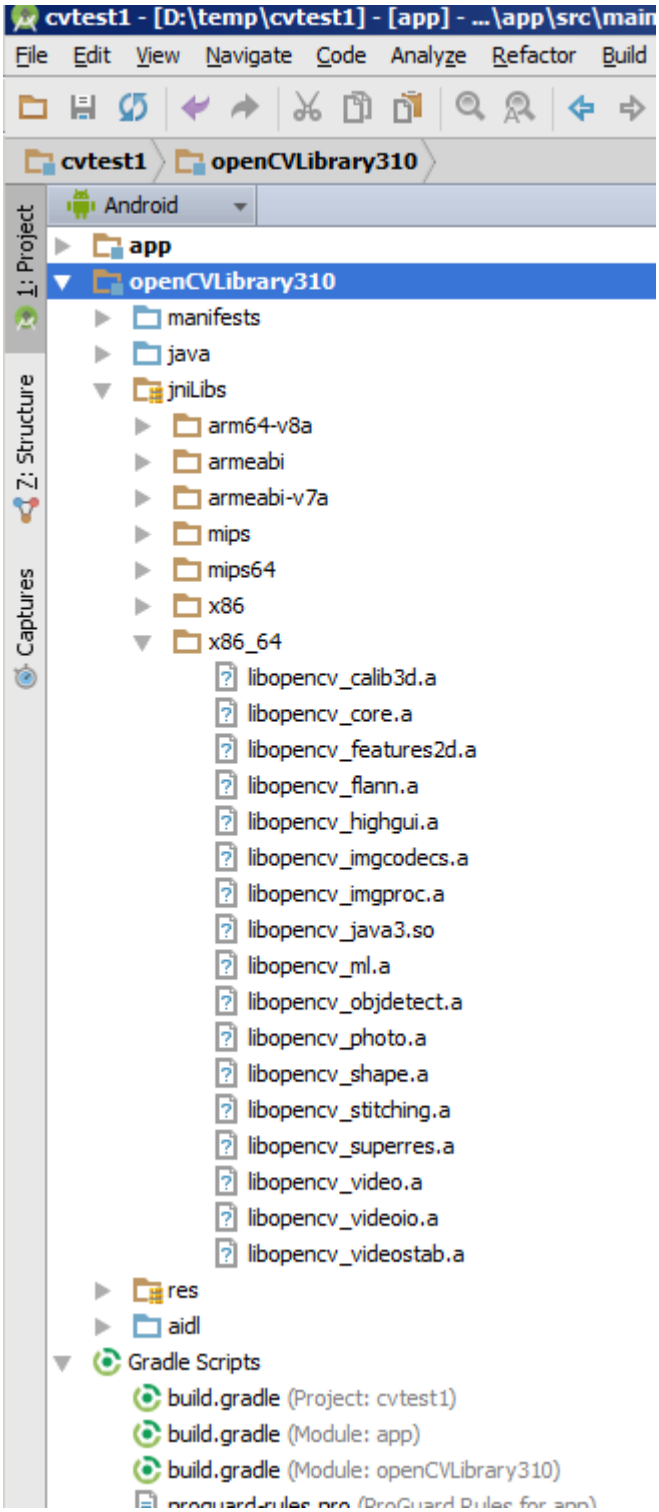
   À ce stade, le module devrait apparaître dans la hiérarchie du projet en tant que **openCVLibrary310** , tout comme **app** . (Notez que dans ce petit menu déroulant, je suis passé de la **vue projet à la vue Android**). Vous devriez également voir un fichier **build.gradle** supplémentaire sous "Gradle Scripts", mais je trouve que l'interface d'Android Studio est un peu complexe et qu'elle ne le fait parfois pas tout de suite. Essayez donc de resynchroniser, de nettoyer, voire de redémarrer Android Studio.

   Vous devriez voir le module openCVLibrary310 avec toutes les fonctions OpenCV sous java

comme dans cette capture d'écran:



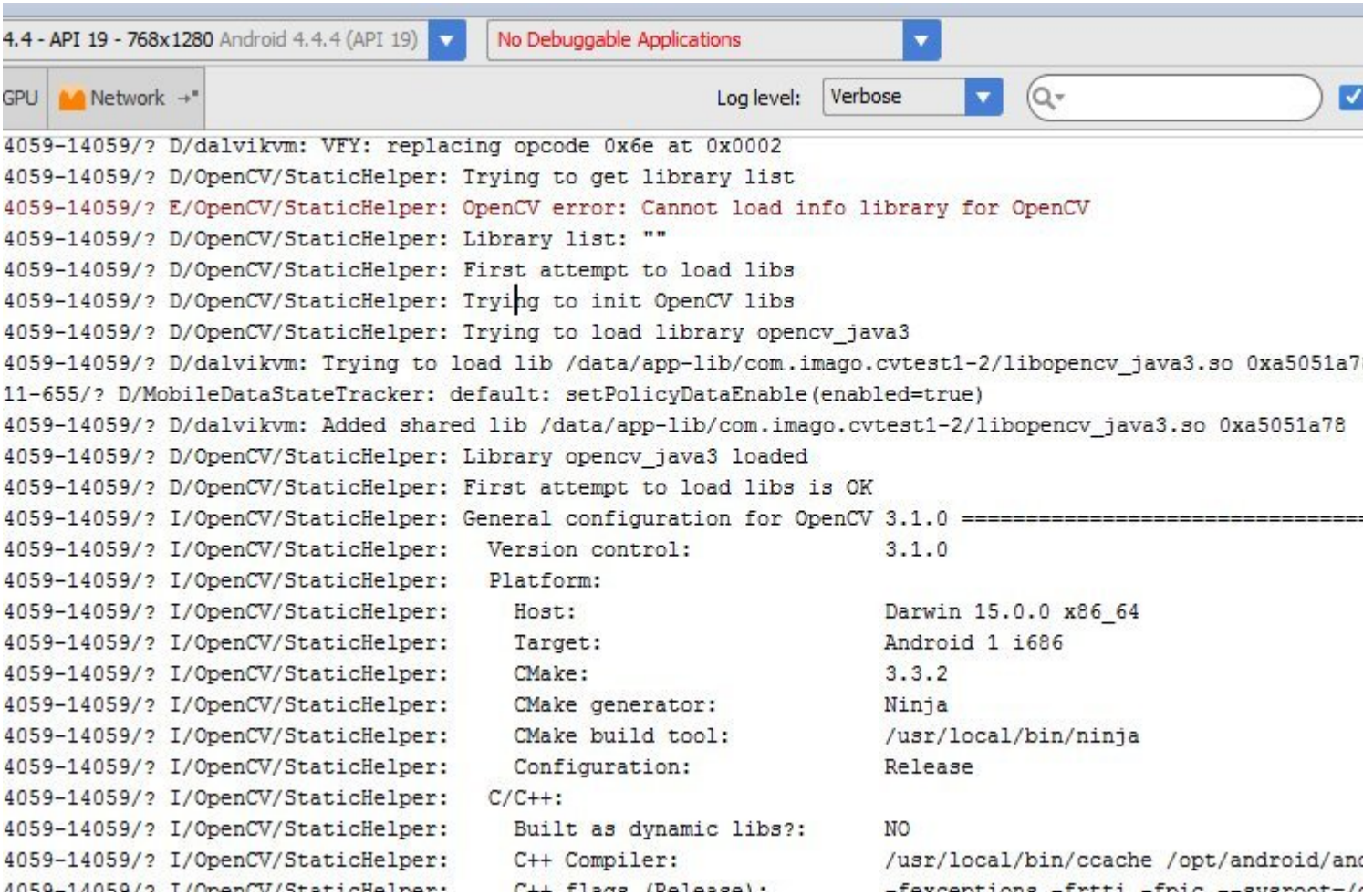
9. Copiez le répertoire **{unzip-dir} / sdk / native / libs** (et tout ce qu'il **contient**) dans votre projet Android, dans **cvtest1 / OpenCVLibrary310 / src / main /** , puis renommez votre copie de **libs** en **jniLibs** . Vous devriez maintenant avoir un **répertoire cvtest1 / OpenCVLibrary310 / src / main / jniLibs** . Resynchronisez votre projet et ce répertoire devrait maintenant apparaître dans la vue du projet sous **openCVLibrary310** .



10. Accédez à la méthode onCreate de *MainActivity.java* et ajoutez ce code:

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

Ensuite, lancez votre application. Vous devriez voir des lignes comme celles-ci dans le moniteur Android:



(Je ne sais pas pourquoi cette ligne avec le message d'erreur est là)

11. Maintenant, essayez d'utiliser un code openCV. Dans l'exemple ci-dessous, j'ai copié un fichier .jpg dans le répertoire cache de l'application cvtest1 sur l'émulateur Android. Le code ci-dessous charge cette image, exécute l'algorithme de détection de bord canny, puis écrit les résultats dans un fichier .png dans le même répertoire.

```
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. Exécutez votre application. Votre émulateur doit créer une image "bord" en noir et blanc. Vous pouvez utiliser le moniteur de périphérique Android pour récupérer la sortie ou écrire une activité pour l'afficher.

Lire Intégrer OpenCV dans Android Studio en ligne:
https://riptutorial.com/fr/android/topic/7068/integrer-opencv-dans-android-studio

Chapitre 148: Intention

Introduction

Une intention est un petit message transmis autour du système Android. Ce message peut contenir des informations sur notre intention d'effectuer une tâche.

Il s'agit essentiellement d'une structure de données passive contenant une description abstraite d'une action à effectuer.

Syntaxe

- Intention Intention ()
- Intention Intention (intention intentionnelle)
- Intention Intent (Action de chaîne)
- Intention Intent (Action de cordes, Uri uri)
- Intention Intent (Context packageContext, Class <?> Cls)
- Intent Intent (Action de chaîne, Uri uri, Context packageContext, Class <?> Cls)
- void startActivity (Intention intentionnelle)
- void startActivity (Intention intentionnelle, Options de regroupement)
- void startActivityForResult (Intention intent, int requestCode)
- void startActivityForResult (Intention intent, int requestCode, Options du bundle)
- Intention putExtra (nom de la chaîne, valeur double [])
- Intention putExtra (nom de la chaîne, valeur int)
- Intention putExtra (Nom de la chaîne, valeur CharSequence)
- Intention putExtra (nom de la chaîne, valeur du caractère)
- Intention putExtra (Nom de la chaîne, valeur du lot)
- Intention putExtra (Nom de la chaîne, valeur Parcelable [])
- Intention putExtra (nom de la chaîne, valeur sérialisable)
- Intention putExtra (nom de la chaîne, valeur int [])
- Intention putExtra (Nom de la chaîne, valeur flottante)
- Intention putExtra (nom de la chaîne, valeur de l'octet [])
- Intention putExtra (nom de la chaîne, valeur long [])
- Intention putExtra (nom de la chaîne, valeur parcelable)
- Intention putExtra (nom de la chaîne, valeur float [])
- Intention putExtra (Nom de la chaîne, valeur longue)
- Intention putExtra (Nom de la chaîne, valeur String [])
- Intention putExtra (Nom de la chaîne, valeur booléenne)
- Intention putExtra (nom de la chaîne, valeur booléenne [])
- Intention putExtra (Nom de la chaîne, valeur courte)
- Intention putExtra (Nom de la chaîne, valeur double)
- Intention putExtra (nom de la chaîne, valeur courte [])
- Intention putExtra (nom de la chaîne, valeur de la chaîne)
- Intention putExtra (Nom de la chaîne, valeur d'octet)

- Intention putExtra (Nom de la chaîne, valeur char [])
- Intention putExtra (nom de la chaîne, valeur CharSequence [])

Paramètres

Paramètre	Détails
intention	L'intention de commencer
code requis	Numéro unique pour identifier la demande
options	Options supplémentaires pour le démarrage de l'activité
prénom	Le nom des données supplémentaires
valeur	La valeur des données supplémentaires
CHOOSE_CONTACT_REQUEST_CODE	le code de la requête, pour l'identifier sur la méthode `onActivityResult`
action	Toute action à effectuer via cette intention, par exemple: Intent.ACTION_VIEW
uri	données uri à utiliser pour effectuer une action spécifiée
packageContext	Contexte à utiliser pour initialiser l'intention
cls	Classe à utiliser par cette intention

Remarques

Mises en garde concernant l'utilisation d'une intention implicite

Lors de l'appel d'une intention implicite, il est toujours utile de vérifier si le système le permet.

Cela peut être fait en vérifiant l'utilisation de `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
}
```

Activité de démarrage qui est une `singleTask` ou `singleTop`

Lorsque le mode de lancement de l'activité est `singleTask` ou `singleTop` , `onActivityResult` sera appelé dès que l'activité sera démarrée avec une donnée null. Pour éviter cela, utilisez `Intent.setFlags(0)` pour réinitialiser les indicateurs par défaut.

Examples

Commencer une activité

Cet exemple démarrera `DestinationActivity` depuis `OriginActivity` .

Ici, le constructeur `Intent` prend deux paramètres:

1. Un contexte comme premier paramètre (utilisé car la classe Activity est une sous-classe de Context)
2. La classe du composant de l'application à laquelle le système doit fournir l'intention (dans ce cas, l'activité à démarrer)

```
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.
    }
}
```

Une autre façon de créer `Intent` pour ouvrir `DestinationActivity` consiste à utiliser le constructeur par défaut pour `Intent` et à utiliser la méthode `setClass()` pour lui indiquer quelle activité ouvrir:

```
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
```

Transmission de données entre activités

Cet exemple illustre l'envoi d'une `String` avec la valeur `"Some data!"` de `OriginActivity` à

`DestinationActivity` .

REMARQUE: C'est le moyen le plus simple d'envoyer des données entre deux activités. Voir l'exemple d'utilisation du modèle de démarrage pour une implémentation plus robuste.

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);
    }
}
```

Il est également possible de passer d' autres `primitive` types de données, ainsi que `arrays` , `Bundle` et `Parcelable` données. Passer `Serializable` est également possible, mais devrait être évité car il est plus de trois fois plus lent que `Parcelable` .

Serializable est une `interface` Java `interface` . Vous marquez simplement une classe comme `Serializable` en implémentant l' `interface Serializable` et Java le sérialisera automatiquement lors des situations requises.

Parcelable est une `interface` spécifique à Android qui peut être implémentée sur des types de données personnalisés (c.-à-d. Vos propres objets / objets POJO), elle permet d'aplatir votre objet et de se reconstruire sans que la destination ait besoin de faire quoi que ce soit. Il existe un exemple de documentation permettant de rendre un objet parcelable .

Une fois que vous avez un objet `parcelable` , vous pouvez l'envoyer comme un type primitif, avec un objet intentionnel:

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

Ou dans un bundle / comme argument pour un fragment:

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

puis lisez-le également à l'intention de destination à l'aide de getParcelableExtra:

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

Ou lors de la lecture d'un fragment d'un bundle:

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

Une fois que vous avez un objet `Serializable` , vous pouvez le placer dans un objet intentionnel:

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

puis lisez-le également à partir de l'objet intentionnel à la destination, comme indiqué ci-dessous:

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

Envoyer des emails

```
// 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
}
```

Cela pré-remplira un email dans une application de messagerie du choix de l'utilisateur.

Si vous devez ajouter une pièce jointe, vous pouvez utiliser `Intent.ACTION_SEND` au lieu de `Intent.ACTION_SENDTO` . Pour plusieurs pièces jointes, vous pouvez utiliser `ACTION_SEND_MULTIPLE`

Un mot d'avertissement: tous les périphériques ne disposent pas d'un fournisseur pour `ACTION_SENDTO` , et l'appel de `startActivity()` sans vérifier avec `resolveActivity()` peut d'abord `resolveActivity()` une ActivityNotFoundException.

Obtenir un résultat d'une autre activité

En utilisant `startActivityForResult(Intent intent, int requestCode)` vous pouvez démarrer une autre `Activity` , puis recevoir un résultat de cette `Activity` dans la `onActivityResult(int requestCode, int resultCode, Intent data)` . Le résultat sera renvoyé en tant `Intent` . Une intention peut contenir des données via un ensemble

---

Dans cet exemple, `MainActivity` lancera une `DetailActivity` et attendra un résultat. Chaque type de requête doit avoir son propre code de demande `int` , de sorte que dans la `onActivityResult(int requestCode, int resultCode, Intent data)` *substituée* `onActivityResult(int requestCode, int resultCode, Intent data)` de `MainActivity` , il soit possible de déterminer la demande de traitement en comparant les valeurs de `requestCode` et `REQUEST_CODE_EXAMPLE` Par exemple, il n'y en a qu'un).

Activité principale:

```
public class MainActivity extends Activity {

    // Use a unique request code for each use case
    private static final int REQUEST_CODE_EXAMPLE = 0x9345;

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

        // Create a new instance of Intent to start DetailActivity
        final Intent intent = new Intent(this, DetailActivity.class);

        // Start DetailActivity with the request code
        startActivityForResult(intent, REQUEST_CODE_EXAMPLE);
    }

    // onActivityResult only get called
    // when the other Activity previously started using startActivityForResult
    @Override
    public void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);

        // First we need to check if the requestCode matches the one we used.
        if(requestCode == REQUEST_CODE_EXAMPLE) {

            // The resultCode is set by the DetailActivity
            // By convention RESULT_OK means that whatever
            // DetailActivity did was executed successfully
```

```
            if(resultCode == Activity.RESULT_OK) {
                // Get the result from the returned Intent
                final String result = data.getStringExtra(DetailActivity.EXTRA_DATA);

                // Use the data - in this case, display it in a Toast.
                Toast.makeText(this, "Result: " + result, Toast.LENGTH_LONG).show();
            } else {
                // setResult wasn't successfully executed by DetailActivity
                // Due to some error or flow of control. No data to retrieve.
            }
        }
    }
}
```

DetailActivity:

```
public class DetailActivity extends Activity {

    // Constant used to identify data sent between Activities.
    public static final String EXTRA_DATA = "EXTRA_DATA";

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

        final Button button = (Button) findViewById(R.id.button);
        // When this button is clicked we want to return a result
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                // Create a new Intent object as container for the result
                final Intent data = new Intent();

                // Add the required data to be returned to the MainActivity
                data.putExtra(EXTRA_DATA, "Some interesting data!");

                // Set the resultCode as Activity.RESULT_OK to
                // indicate a success and attach the Intent
                // which contains our result data
                setResult(Activity.RESULT_OK, data);

                // With finish() we close the DetailActivity to
                // return back to MainActivity
                finish();
            }
        });
    }

    @Override
    public void onBackPressed() {
        // When the user hits the back button set the resultCode
        // as Activity.RESULT_CANCELED to indicate a failure
        setResult(Activity.RESULT_CANCELED);
        super.onBackPressed();
    }
}
```

Quelques points à connaître:

- Les données ne sont renvoyées qu'une fois que vous appelez `finish()` . Vous devez appeler `setResult()` avant d'appeler `finish()` , sinon aucun résultat ne sera renvoyé.
- Assurez-vous que votre `Activity` n'utilise pas `android:launchMode="singleTask"` , `android:launchMode="singleTask"` l' `Activity` sera exécutée dans une tâche distincte et vous ne recevrez donc aucun résultat. Si votre `Activity` utilise `singleTask` comme mode de lancement, elle appellera `onActivityResult()` immédiatement avec un code de résultat `Activity.RESULT_CANCELED` .
- Soyez prudent lorsque vous utilisez `android:launchMode="singleInstance"` . Sur les appareils avant Lollipop (Android 5.0, API niveau 21), les activités ne renverront aucun résultat.
- Vous pouvez utiliser des intentions explicites ou implicites lorsque vous appelez `startActivityForResult()` . Lorsque vous démarrez l'une de vos activités pour recevoir un résultat, vous devez utiliser une intention explicite pour vous assurer de recevoir le résultat attendu. Une `intent` explicite est toujours transmise à sa cible, peu importe ce qu'elle contient; le `filter` n'est pas consulté. Mais une intention implicite est transmise à un composant uniquement si elle peut traverser l'un des filtres du composant.

Ouvrir une URL dans un navigateur

Ouverture avec le navigateur par défaut

Cet exemple montre comment vous pouvez ouvrir une URL par programmation dans le navigateur Web intégré plutôt que dans votre application. Cela permet à votre application d'ouvrir une page Web sans avoir besoin d'inclure la permission `INTERNET` dans votre fichier manifeste.

```
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);
    }
}
```

Demander à l'utilisateur de sélectionner un navigateur

Notez que cet exemple utilise la méthode `Intent.createChooser()` :

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

}
```

Dans certains cas, l'URL peut commencer par **"www"** . Si tel est le cas, vous obtiendrez cette exception:

> `android.content.ActivityNotFoundException` : Aucune activité détectée pour gérer l'intention

L'URL doit toujours commencer par **"http: //"** ou **"https: //"** . Votre code doit donc le vérifier, comme illustré dans l'extrait de code suivant:

```
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);
}
```

Les meilleures pratiques

Vérifiez s'il n'y a pas d'applications sur le périphérique pouvant recevoir l'intention implicite. Sinon, votre application va planter lorsqu'elle appelle `startActivity()` . Pour vérifier d'abord qu'une application existe pour recevoir l'intention, appelez `resolveActivity()` sur votre objet Intent. Si le résultat est non nul, il existe au moins une application capable de gérer l'intention et d'appeler `startActivity()` toute sécurité. Si le résultat est null, vous ne devez pas utiliser l'intention et, si possible, vous devez désactiver la fonctionnalité invoquant l'intention.

Effacement d'une pile d'activités

Parfois, vous voudrez peut-être commencer une nouvelle activité tout en supprimant les activités précédentes de la pile arrière, de sorte que le bouton Précédent ne vous les renvoie pas. Par exemple, vous pouvez lancer une application sur l'activité de connexion pour accéder à l'activité principale de votre application, mais lorsque vous vous déconnectez, vous souhaitez être redirigé vers Login sans avoir à revenir en arrière. Dans un cas comme celui-ci, vous pouvez définir l'indicateur `FLAG_ACTIVITY_CLEAR_TOP` pour l'intention, ce qui signifie que l'activité en cours d'exécution est déjà en cours dans la tâche en cours (LoginActivity), au lieu de lancer une nouvelle instance de cette activité. il sera fermé et cette intention sera livrée à l'ancienne activité (désormais au-dessus) en tant que nouvelle intention.

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

Il est également possible d'utiliser les indicateurs `FLAG_ACTIVITY_NEW_TASK` avec `FLAG_ACTIVITY_CLEAR_TASK` si vous souhaitez effacer toutes les activités de la pile arrière:

```
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);
```

Intention URI

Cet exemple montre comment démarrer intentionnellement à partir du navigateur:

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

Cette intention démarre l'application avec le package `com.sample.test` ou ouvre Google Play avec ce package.

En outre, cette intention peut être lancée avec JavaScript:

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

En activité, cet hôte et ce chemin peuvent être obtenus à partir des données d'intention:

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

Syntaxe d'URI d'intention:

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

Diffusion de messages vers d'autres composants

Les intentions peuvent être utilisées pour diffuser des messages vers d'autres composants de votre application (tels qu'un service d'arrière-plan en cours d'exécution) ou vers l'ensemble du système Android.

Pour envoyer une diffusion **dans votre application** , utilisez la classe `LocalBroadcastManager` :

```
Intent intent = new Intent("com.example.YOUR_ACTION"); // the intent action
intent.putExtra("key", "value"); // data to be passed with your broadcast

LocalBroadcastManager manager = LocalBroadcastManager.getInstance(context);
manager.sendBroadcast(intent);
```

Pour envoyer une diffusion à des composants extérieurs à votre application, utilisez la méthode `sendBroadcast()` sur un objet `Context` .

```
Intent intent = new Intent("com.example.YOUR_ACTION"); // the intent action
intent.putExtra("key", "value"); // data to be passed with your broadcast

context.sendBroadcast(intent);
```

Vous trouverez des informations sur la *réception des* émissions ici: Récepteur de diffusion

CustomTabsIntent pour les onglets personnalisés Chrome

4.0.3

À l'aide d'un `CustomTabsIntent` , il est désormais possible de configurer les onglets personnalisés de Chrome afin de personnaliser les composants clés de l'interface utilisateur dans le navigateur ouvert à partir de votre application.

C'est une bonne alternative à l'utilisation de WebView dans certains cas. Il permet de charger une page Web avec une intention, avec la possibilité d'ajouter une certaine apparence à votre application dans le navigateur.

Voici un exemple d'ouverture d'une URL à l'aide de `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));
```

Remarque:

Pour utiliser des onglets personnalisés, vous devez ajouter cette dépendance à votre build.gradle

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

Partage de plusieurs fichiers par intention

La liste de chaînes transmise en tant que paramètre à la méthode `share()` contient les chemins

d'accès de tous les fichiers que vous souhaitez partager.

Il parcourt les chemins, les ajoute à Uri et lance l'activité qui peut accepter des fichiers de ce type.

```
  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);
    }
```

Motif de départ

Ce modèle est une approche plus stricte pour démarrer une `Activity` . Son objectif est d'améliorer la lisibilité du code tout en réduisant la complexité du code, les coûts de maintenance et le couplage de vos composants.

L'exemple suivant implémente le modèle de démarrage, qui est généralement implémenté en tant que méthode statique sur l' `Activity` elle-même. Cette méthode statique accepte tous les paramètres requis, construit une `Intent` valide à partir de ces données, puis démarre l' `Activity` .

Un `Intent` est un objet qui fournit une liaison d'exécution entre des composants distincts, tels que deux activités. L'intention représente "l'intention de faire quelque chose" d'une application. Vous pouvez utiliser des intentions pour une grande variété de tâches, mais ici, votre intention commence une autre activité.

```
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);
    }
}
```

Ce modèle vous permet également de forcer la transmission de données supplémentaires avec l'intention.

L' `ExampleActivity` peut alors être démarré comme ceci, où le `context` est un contexte d'activité:

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

Démarrer le service non lié à l'aide d'une intention

Un service est un composant qui s'exécute en arrière-plan (sur le thread d'interface utilisateur) sans interaction directe avec l'utilisateur. Un service non lié vient d'être démarré et n'est lié au cycle de vie d'aucune activité.

Pour démarrer un service, vous pouvez faire comme indiqué dans l'exemple ci-dessous:

```
// 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);
```

Vous pouvez utiliser des extras de l'intention en utilisant un `onStartCommand()` :

```
public class MyService extends Service {
    public MyService() {
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startId)
    {
        if (intent != null) {
            Bundle extras = intent.getExtras();
            String key1 = extras.getString("KEY1", "");
            if (key1.equals("Value to be used by the service")) {
                //do something
            }
        }
        return START_STICKY;
    }

    @Nullable
    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }

}
```

Partager l'intention

Partagez des informations simples avec différentes applications.

```
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)));
```

Partagez une image avec différentes applications.

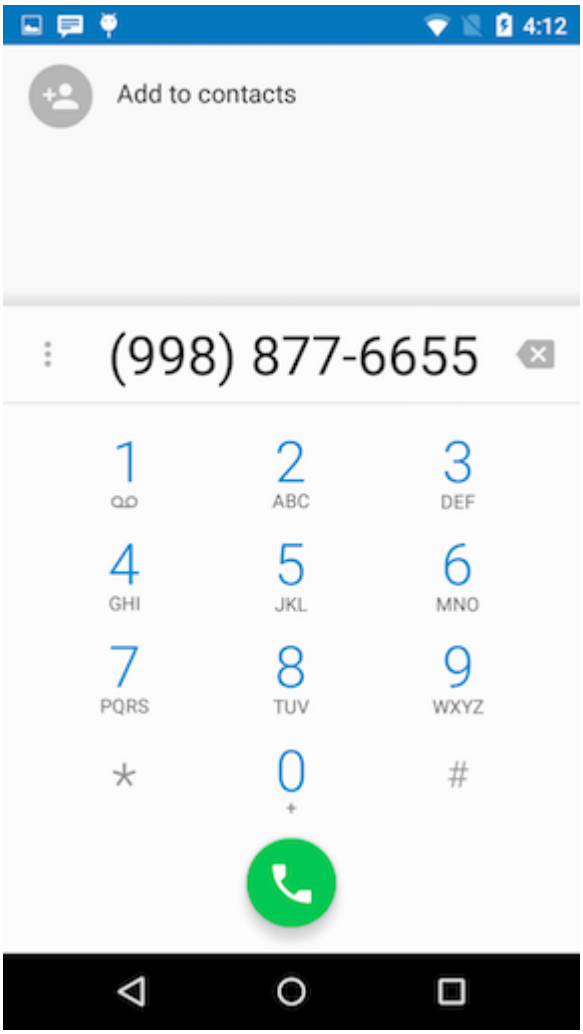
```
Intent shareIntent = new Intent();
shareIntent.setAction(Intent.ACTION_SEND);
shareIntent.putExtra(Intent.EXTRA_STREAM, uriToImage);
shareIntent.setType("image/jpeg");
startActivity(Intent.createChooser(shareIntent, getResources().getText(R.string.send_to)));
```

Démarrer le numéroteur

Cet exemple montre comment ouvrir un composeur par défaut (une application qui effectue des appels réguliers) avec un numéro de téléphone fourni:

```
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);
```

Résultat de l'exécution du code ci-dessus:



Ouvrir la carte Google avec la latitude et la longitude spécifiées

Vous pouvez transmettre la latitude, la longitude de votre application à Google map en utilisant Intent

```
String uri = String.format(Locale.ENGLISH, "http://maps.google.com/maps?q=loc:%f,%f",
28.43242324,77.8977673);
Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse(uri));
startActivity(intent);
```

Transmission de données différentes via une intention dans l'activité

1. Passer des données entières:

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. Passer des données doubles:

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. Données de chaîne de transmission:

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");
```

ou

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

4. Transmission des données ArrayList:

SenderActivity

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

ReceiverActivity

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

5. Passer des données d'objet:

SenderActivity

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

ReceiverActivity

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

> **Remarque:** gardez à l'esprit que votre classe personnalisée doit implémenter l'interface `Serializable` .

6. Transmission des données HashMap <String, String>:

SenderActivity

HashMap <String, String> hashMap;

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

ReceiverActivity

```
Intent mIntent = getIntent();
HashMap<String, String> hashMap = (HashMap<String, String>)
mIntent.getSerializableExtra("hashMap");
```

7. Transmission de données Bitmap:

SenderActivity

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

ReceiverActivity

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

Affichage d'un sélecteur de fichier et lecture du résultat

Démarrer une activité de sélecteur de fichiers

```
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);
}
```

Lecture du résultat

```
@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;
}
```

Passer objet personnalisé entre les activités

Il est également possible de transmettre votre objet personnalisé à d'autres activités en utilisant la classe `Bundle` .

Il y a deux manières:

- Interface `Serializable` - pour Java et Android
- Interface `Parcelable` efficace pour la mémoire, uniquement pour Android (recommandé)

Parcelable

Le traitement parcellaire est beaucoup plus rapide que le traitement sérialisable. Une des raisons à cela est que nous sommes explicites sur le processus de sérialisation au lieu d'utiliser la réflexion pour l'inférer. Il va de soi que le code a été fortement optimisé à cette fin.

```
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 d'activité d'envoi

```
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);
```

Recevoir l'objet dans l'activité de destination.

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

Vous pouvez passer Arraylist d'objet Parceble comme ci-dessous

```
//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");
```

> **Note:** Il existe des plugins Android Studio tels que celui-ci disponibles pour générer du code Parcelable

Sérialisable

Code d'activité d'envoi

```
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);
```

Recevoir l'objet dans l'activité de destination.

```
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` d'objet Serializable: identique au passage d'un objet unique

L'objet personnalisé doit implémenter l'interface `Serializable` .

Obtenir un résultat de l'activité au fragment

Comme pour obtenir un résultat d'une autre activité, vous devez appeler la méthode `Fragment`

`startActivityForResult(Intent intent, int requestCode)` . Notez que vous ne devez pas appeler `getActivity().startActivityForResult()` car cela ramènera le résultat à l' `Activity` parente du `Fragment` .

La réception du résultat peut être effectuée en utilisant la méthode de `Fragment` `onActivityResult()` . Vous devez vous assurer que l'activité parente du fragment remplace également `onActivityResult()` et appelle sa `super` implémentation.

Dans l'exemple suivant, `ActivityOne` contient `FragmentOne` , qui démarrera `ActivityTwo` et attend un résultat.

ActivityOne

```
public class ActivityOne extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_one);
    }

    // You must override this method as the second Activity will always send its results to
this Activity and then to the Fragment
    @Override
    public void onActivityResult(int requestCode, int resultCode, Intent data) {
       super.onActivityResult(requestCode, resultCode, data);
    }
}
```

activity_one.xml

```
<fragment android:name="com.example.FragmentOne"
    android:id="@+id/fragment_one"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

FragmentOne

```
public class FragmentOne extends Fragment {
    public static final int REQUEST_CODE = 11;
    public static final int RESULT_CODE = 12;
    public static final String EXTRA_KEY_TEST = "testKey";

    // Initializing and starting the second Activity
    private void startSecondActivity() {
        Intent intent = new Intent(getActivity(), ActivityTwo.class);
        startActivityForResult(REQUEST_CODE, intent);
    }

    @Override
    public void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (requestCode == REQUEST_CODE && resultCode == RESULT_CODE) {
            String testResult = data.getStringExtra(EXTRA_KEY_TEST);
            // TODO: Do something with your extra data
        }
```

```
    }
}
```

ActivityTwo

```
public class ActivityTwo extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_two);
    }

    private void closeActivity() {
        Intent intent = new Intent();
        intent.putExtra(FragmentOne.EXTRA_KEY_TEST, "Testing passing data back to
ActivityOne");
        setResult(FragmentOne.RESULT_CODE, intent); // You can also send result without any
data using setResult(int resultCode)
        finish();
    }
}
```

Lire Intention en ligne: https://riptutorial.com/fr/android/topic/103/intention

Chapitre 149: Intentions implicites

Syntaxe

- Intention()
- Intention (Intention o)
- Intention (action de chaîne)
- Intention (action de cordes, Uri uri)
- Intention (Contexte packageContext, Class <?> Cls)
- Intention (Action de chaîne, Uri uri, Context packageContext, Class <?> Cls)

Paramètres

Paramètres	Détails
o	Intention
action	`String`: action d'intention, telle que `ACTION_VIEW`.
uri	`Uri`: L'URI de données d'intention.
packageContext	`Context`: Contexte du package d'application implémentant cette classe.
cls	`Class`: classe de composant à utiliser pour l'intention.

Remarques

- En savoir plus sur l' intention
- En savoir plus sur les types d'intention

Examples

Intentions implicites et explicites

Une intention explicite est utilisée pour démarrer une activité ou un service dans le même package d'application. Dans ce cas, le nom de la classe visée est explicitement mentionné:

```
Intent intent = new Intent(this, MyComponent.class);
startActivity(intent);
```

Toutefois, une intention implicite est envoyée sur le système pour toute application installée sur le périphérique de l'utilisateur et capable de gérer cette intention. Ceci est utilisé pour partager des informations entre différentes applications.

```
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
}
```

Vous trouverez plus de détails sur les différences dans les documents Android Developer: Intent Resolution

Intentions implicites

Les intentions implicites ne nomment pas un composant spécifique, mais déclarent une action générale à effectuer, ce qui permet à un composant d'une autre application de le gérer.

Par exemple, si vous souhaitez montrer à l'utilisateur un emplacement sur une carte, vous pouvez utiliser une intention implicite pour demander qu'une autre application capable affiche un emplacement spécifié sur une carte.

Exemple:

```
// 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);
}
```

Lire Intentions implicites en ligne: https://riptutorial.com/fr/android/topic/5336/intentions-implicites

Chapitre 150: IntentService

Syntaxe

4. <service android: name = ". UploadS3IntentService" android: exporté = "false" />

Remarques

Un `IntentService` fournit un moyen simple de décharger le travail sur un thread d'arrière-plan. Il gère tout ce qui concerne la réception des demandes, leur mise en attente, leur arrêt, etc. pour vous. Il est également facile à mettre en œuvre, ce qui en fait la solution idéale lorsque vous avez des opérations fastidieuses à effectuer qui n'appartiennent pas au thread principal (UI).

Examples

Créer un IntentService

Pour créer un IntentService, créez une classe qui étend `IntentService` , et en son sein, une méthode qui remplace `onHandleIntent` :

```
package com.example.myapp;
public class MyIntentService extends IntentService {
    @Override
     protected void onHandleIntent (Intent workIntent) {
         //Do something in the background, based on the contents of workIntent.
     }
}
```

Exemple de service d'intention

Voici un exemple de `IntentService` qui prétend charger des images en arrière-plan. Tout ce que vous devez faire pour implémenter un `IntentService` est de fournir un constructeur qui appelle le constructeur `super(String)` , et vous devez implémenter la `onHandleIntent(Intent)` .

```
public class ImageLoaderIntentService extends IntentService {

    public static final String IMAGE_URL = "url";

    /**
     * Define a constructor and call the super(String) constructor, in order to name the
worker
     * thread - this is important if you want to debug and know the name of the thread upon
     * which this Service is operating its jobs.
     */
    public ImageLoaderIntentService() {
        super("Example");
    }
```

```
    @Override
    protected void onHandleIntent(Intent intent) {
        // This is where you do all your logic - this code is executed on a background thread

        String imageUrl = intent.getStringExtra(IMAGE_URL);

        if (!TextUtils.isEmpty(imageUrl)) {
            Drawable image = HttpUtils.loadImage(imageUrl); // HttpUtils is made-up for the
example
        }

        // Send your drawable back to the UI now, so that you can use it - there are many ways
        // to achieve this, but they are out of reach for this example
    }
}
```

Pour lancer `IntentService` , vous devez lui envoyer une `Intent` . Vous pouvez le faire à partir d'une `Activity` , par exemple. Bien sûr, vous n'êtes pas limité à cela. Voici un exemple de la façon dont vous invoqueriez votre nouveau `Service` partir d'une classe d' `Activity` .

```
Intent serviceIntent = new Intent(this, ImageLoaderIntentService.class); // you can use 'this'
as the first parameter if your class is a Context (i.e. an Activity, another Service, etc.),
otherwise, supply the context differently
serviceIntent.putExtra(IMAGE_URL, "http://www.example-site.org/some/path/to/an/image");
startService(serviceIntent); // if you are not using 'this' in the first line, you also have
to put the call to the Context object before startService(Intent) here
```

`IntentService` traite les données de ses `Intent` manière séquentielle, de sorte que vous pouvez envoyer plusieurs `Intent` sans vous soucier de savoir si elles entrent en collision. Seule une `Intent` à la fois est traitée, les autres vont dans une file d'attente. Lorsque tous les travaux sont terminés, `IntentService` s'arrête automatiquement.

Exemple de base IntentService

La classe abstraite `IntentService` est une classe de base pour les services, qui s'exécutent en arrière-plan sans aucune interface utilisateur. Par conséquent, pour mettre à jour l'interface utilisateur, nous devons utiliser un récepteur, qui peut être un `BroadcastReceiver` ou un `ResultReceiver` :

- Un `BroadcastReceiver` doit être utilisé si votre service doit communiquer avec plusieurs composants souhaitant écouter les communications.
- Un `ResultReceiver` : doit être utilisé si votre service doit communiquer uniquement avec l'application parente (c'est-à-dire votre application).

Dans `IntentService` , nous avons une méthode clé, `onHandleIntent()` , dans laquelle nous allons effectuer toutes les actions, par exemple, préparer des notifications, créer des alarmes, etc.

Si vous souhaitez utiliser votre propre `IntentService` , vous devez l'étendre comme suit:

```
public class YourIntentService extends IntentService {
    public YourIntentService () {
        super("YourIntentService ");
```

```
    }

    @Override
    protected void onHandleIntent(Intent intent) {
        // TODO: Write your own code here.
    }
}
```

L'appel / démarrage de l'activité peut se faire comme suit:

```
Intent i = new Intent(this, YourIntentService.class);
startService(i);  // For the service.
startActivity(i); // For the activity; ignore this for now.
```

Semblable à toute activité, vous pouvez lui transmettre des informations supplémentaires, telles que des données de regroupement, comme suit:

```
Intent passDataIntent = new Intent(this, YourIntentService.class);
msgIntent.putExtra("foo","bar");
startService(passDataIntent);
```

Supposons maintenant que nous avons transmis des données à la classe `YourIntentService` . Sur la base de ces données, une action peut être effectuée comme suit:

```
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.
        }
    }
}
```

Le code ci-dessus montre également comment gérer les contraintes dans la méthode `OnHandleIntent()` .

Lire IntentService en ligne: https://riptutorial.com/fr/android/topic/5319/intentservice

Chapitre 151: Interfaces

Examples

Auditeur personnalisé

Définir une interface

```
//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);
}
```

Créer un auditeur

Dans l'étape suivante, nous devons définir une variable d'instance dans l'objet qui enverra un rappel via `MyCustomListener` . Et ajouter setter pour notre auditeur.

```
public class SampleClassB {
    private MyCustomListener listener;

    public void setMyCustomListener(MyCustomListener listener) {
        this.listener = listener;
    }
}
```

Mettre en œuvre l'auditeur

Maintenant, dans une autre classe, nous pouvons créer une instance de `SampleClassB` .

```
public class SomeActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        SampleClassB sampleClass = new SampleClassB();
    }
}
```

Ensuite, nous pouvons définir notre auditeur, pour `sampleClass` , de deux manières:

par implémente `MyCustomListener` dans notre classe:

```
public class SomeActivity extends Activity implements MyCustomListener {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        SampleClassB sampleClass = new SampleClassB();
        sampleClass.setMyCustomListener(this);
    }

    @Override
    public void onSuccess(List<Bitmap> bitmapList) {

    }

    @Override
    public void onFailure(String error) {

    }
}
```

ou juste instancier une classe interne anonyme:

```
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) {

            }
        });
    }
}
```

Auditeur de déclenchement

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

Auditeur de base

Le modèle «écouteur» ou «observateur» est la stratégie la plus courante pour créer des rappels asynchrones dans le développement Android.

```
public class MyCustomObject {

  //1 - Define the interface
  public interface MyCustomObjectListener {
      public void onAction(String action);
  }

  //2 - Declare your listener object
  private MyCustomObjectListener listener;

  // and initialize it in the costructor
  public MyCustomObject() {
    this.listener = null;
 }

 //3 - Create your listener setter
 public void setCustomObjectListener(MyCustomObjectListener listener) {
    this.listener = listener;
 }

 // 4 - Trigger listener event
 public void makeSomething(){
    if (this.listener != null){
       listener.onAction("hello!");
 }
}
```

Maintenant sur votre activité:

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

Lire Interfaces en ligne: https://riptutorial.com/fr/android/topic/1785/interfaces

Chapitre 152: Internationalisation et localisation (I18N et L10N)

Introduction

L'internationalisation (i18n) et la localisation (L10n) permettent d'adapter les logiciels en fonction des différences de langue, des différences régionales et du public cible.

Internationalisation: processus de planification pour la localisation future, c.-à-d. Rendre la conception du logiciel flexible dans une mesure telle qu'elle peut s'adapter et s'adapter aux futurs efforts de localisation.

Localisation: processus d'adaptation du logiciel à une région / un pays / un marché particulier (paramètres régionaux).

Remarques

Pour tester un périphérique en vue de sa localisation, le périphérique ou l'émulateur peut être redémarré dans un environnement local spécifique en utilisant `adb` comme suit:

1. Exécutez adb à l'aide de la commande: `adb shell`
2. Exécutez la commande suivante à l'invite de commande adb: `setprop persist.sys.locale [BCP-47 language tag];stop;sleep 5;start` [code de langue BCP-47] le code spécifique à la langue décrit ici: codes BCP47

Par exemple, pour vérifier la localisation en japonais dans l'application, utilisez la commande: `setprop persist.sys.locale ja-JP;stop;sleep 5;start`

Examples

Planification de la localisation: activez le support RTL dans Manifest

La prise en charge de RTL (de droite à gauche) est un élément essentiel de la planification pour i18n et L10n. Contrairement à la langue anglaise écrite de gauche à droite, de nombreuses langues comme l'arabe, le japonais, l'hébreu, etc. sont écrites de droite à gauche. Pour attirer un public plus global, il est judicieux de planifier vos mises en page de manière à ce qu'elles prennent en charge ces langages dès le début du projet, de manière à faciliter l'ajout de la localisation par la suite.

RTL support peut être activé dans une application Android en ajoutant la `supportsRtl` tag dans le `AndroidManifest` , comme suit:

```
<application
    ...
    android:supportsRtl="true"
```

```
    ...>
...
</application>
```

Planification de la localisation: ajout du support RTL dans les mises en page

À partir du SDK 17 (Android 4.2), le support RTL a été ajouté dans les dispositions Android et constitue une partie essentielle de la localisation. À l'avenir, la notation `left/right` dans les mises en page doit être remplacée par la notation de `start/end` . Si, toutefois, votre projet a une valeur `minSdk` inférieure à `17` , vous devez utiliser les deux notations `left/right` et `start/end` .

Pour les mises en page relatives, `alignParentStart` et `alignParentEnd` , comme ceci:

```
<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>
```

Pour spécifier la gravité et la gravité de la disposition, une notation similaire doit être utilisée, comme ceci:

```
    <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"/>
```

Les rembourrages et les marges doivent également être spécifiés en conséquence, comme ceci:

```
<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"/>
```

Planification de la localisation: test des dispositions pour RTL

Pour tester si les mises en page créées sont compatibles RTL, procédez comme suit:

> Allez dans Paramètres -> Options du développeur -> Dessin -> Force la direction de la mise en page RTL

L'activation de cette option obligerait le périphérique à utiliser les paramètres régionaux RTL et vous pouvez facilement vérifier toutes les parties de l'application pour la prise en charge RTL. Notez que vous n'avez pas besoin d'ajouter de nouveaux paramètres linguistiques / locaux jusqu'à ce stade.

Codage pour la localisation: création de chaînes et de ressources par défaut

La première étape du codage de la localisation consiste à créer des ressources par défaut. Cette étape est tellement implicite que de nombreux développeurs n'y pensent même pas. Cependant, la création de ressources par défaut est importante car si le périphérique s'exécute sur des paramètres régionaux non pris en charge, il charge toutes ses ressources à partir des dossiers par défaut. Si même une des ressources est manquante dans les dossiers par défaut, l'application se bloquerait simplement.

L'ensemble de chaînes par défaut doit être placé dans le dossier suivant à l'emplacement spécifié:

```
res/values/strings.xml
```

Ce fichier doit contenir les chaînes dans la langue que la majorité des utilisateurs de l'application sont censés parler.

En outre, les ressources par défaut de l'application doivent être placées dans les dossiers et emplacements suivants:

```
res/drawable/
res/layout/
```

Si votre application nécessite des dossiers tels que `anim` ou `xml` , les ressources par défaut doivent être ajoutées aux dossiers et emplacements suivants:

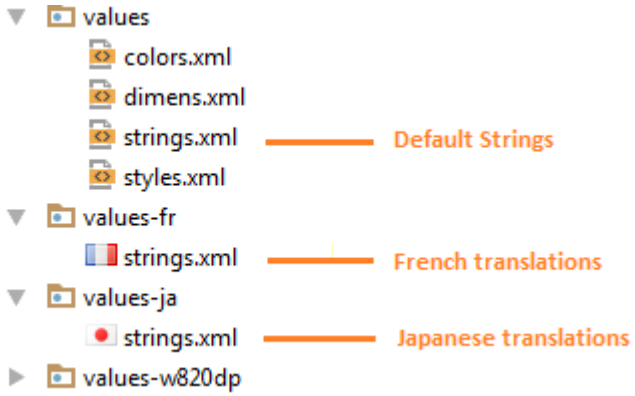
```
res/anim/
res/xml/
res/raw/
```

Codage pour la localisation: fournir des chaînes alternatives

Pour fournir des traductions dans d'autres langues (locales), nous devons créer un `strings.xml` dans un dossier distinct en respectant la convention suivante:

```
res/values-<locale>/strings.xml
```

Un exemple pour la même chose est donné ci-dessous:



Dans cet exemple, nous avons des chaînes en anglais par défaut dans le fichier `res/values/strings.xml` , des traductions en français sont fournies dans le dossier `res/values-fr/strings.xml` et des traductions en japonais sont fournies dans le dossier `res/values-ja/strings.xml`

D'autres traductions pour d'autres paramètres régionaux peuvent également être ajoutées à l'application.

> Une liste complète des codes de paramètres régionaux peut être trouvée ici: codes ISO 639

Chaînes non traduisibles:

Votre projet peut avoir certaines chaînes à ne pas traduire. Les chaînes qui sont utilisées comme clés pour SharedPreferences ou des chaînes de caractères utilisées comme symboles font partie de cette catégorie. Ces chaînes ne doivent être stockées que dans le `strings.xml` par défaut `strings.xml` et doivent être marquées avec un attribut `translatable="false"` . par exemple

```
<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>
```

Cet attribut est important car les traductions sont souvent effectuées par des professionnels bilingues. Cela permettrait à ces personnes impliquées dans la traduction d'identifier les chaînes à ne pas traduire, économisant ainsi du temps et de l'argent.

Codage pour la localisation: fournir des mises en page alternatives

La création de dispositions spécifiques à une langue est souvent inutile si vous avez spécifié la

notation de `start/end` correcte, comme décrit dans l'exemple précédent. Cependant, il peut arriver que les mises en page par défaut ne fonctionnent pas correctement dans certaines langues. Parfois, les mises en page de gauche à droite peuvent ne pas être traduites pour les langages RTL. Il est nécessaire de fournir les dispositions correctes dans de tels cas.

Pour fournir une optimisation complète des mises en page RTL, nous pouvons utiliser des fichiers de mise en page entièrement séparés en utilisant le `ldrtl` ressource `ldrtl` (`ldrtl` signifie layout-direction-right-left}). Par exemple, nous pouvons enregistrer vos fichiers de mise en page par défaut dans `res/layout/` et nos mises en page optimisées RTL dans `res/layout-ldrtl/` .

Le qualificatif `ldrtl` est idéal pour les ressources pouvant être dessinées, de sorte que vous pouvez fournir des graphiques orientés dans la direction correspondant à la direction de lecture.

Voici un excellent article décrivant la priorité des dispositions `ldrtl` : `ldrtl` page spécifiques à la langue

Lire Internationalisation et localisation (I18N et L10N) en ligne: https://riptutorial.com/fr/android/topic/8796/internationalisation-et-localisation--i18n-et-l10n-

Chapitre 153: Jackson

Introduction

Jackson est une bibliothèque Java polyvalente pour le traitement de JSON. Jackson se veut la meilleure combinaison possible de rapidité, de correction, de légèreté et d'ergonomie pour les développeurs.

Caractéristiques de Jackson:

Mode multi-traitement et très bonne collaboration

Non seulement les annotations, mais aussi les annotations mixtes

Prise en charge complète des types génériques

Soutenir les types polymorphes

Examples

Exemple de liaison de données complète

Données JSON

```
{
  "name" : { "first" : "Joe", "last" : "Sixpack" },
  "gender" : "MALE",
  "verified" : false,
  "userImage" : "keliuyue"
}
```

Il faut deux lignes de Java pour en faire une instance utilisateur:

```
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;
    }
}
```

Le retour au JSON est tout aussi simple:

```
mapper.writeValue(new File("user-modified.json"), user);
```

Lire Jackson en ligne: https://riptutorial.com/fr/android/topic/10878/jackson

Chapitre 154: Java sur Android

Introduction

Android prend en charge toutes les fonctionnalités de langage Java 7 et un sous-ensemble de fonctionnalités de langage Java 8 qui varient selon la version de la plate-forme. Cette page décrit les nouvelles fonctionnalités linguistiques que vous pouvez utiliser, comment configurer correctement votre projet pour les utiliser et tout problème connu.

Examples

Fonctionnalités Java 8 sous-ensemble avec Retrolambda

Retrolambda vous permet d'exécuter du code Java 8 avec des expressions lambda, des références de méthode et des instructions try-with-resources sous Java 7, 6 ou 5. Pour ce faire, il transforme votre bytecode compilé Java 8 afin qu'il puisse s'exécuter sur une ancienne version de Java.

Fonctionnalités de langue Backported:

- Les expressions lambda sont exportées en les convertissant en classes internes anonymes. Cela inclut l'optimisation de l'utilisation d'une instance de singleton pour les expressions lambda sans état afin d'éviter une allocation d'objet répétée. Les références de méthode sont simplement du sucre de syntaxe pour les expressions lambda et elles sont de la même manière.

- Les instructions try-with-resources sont backportées en supprimant les appels à `Throwable.addSuppressed` si la version du bytecode cible est inférieure à Java 7. Si vous souhaitez que les exceptions supprimées soient consignées au lieu d'être avalées, créez une demande de fonctionnalité et nous la ferons configurable

- `Objects.requireNonNull` appels `Objects.requireNonNull` sont remplacés par des appels à `Object.getClass` si la version du bytecode cible est inférieure à Java 7. Les vérifications null synthétiques générées par JDK 9 utilisent `Objects.requireNonNull` , alors que les versions JDK antérieures utilisaient `Object.getClass` .

- Éventuellement aussi:

    1. Les méthodes par défaut sont répliquées en copiant les méthodes par défaut dans une classe associée (nom d'interface + "$") en tant que méthodes statiques, en remplaçant les méthodes par défaut de l'interface par des méthodes abstraites et en ajoutant les implémentations de méthode nécessaires à toutes les classes implémentant cette interface .

    2. Les méthodes statiques sur les interfaces sont répliquées en déplaçant les méthodes statiques vers une classe associée (nom d'interface + "$") et en modifiant tous les

appels de méthodes pour appeler le nouvel emplacement de méthode.

Limitations Connues:

- Ne transfère pas les API Java 8
- Le backporting des méthodes par défaut et des méthodes statiques sur les interfaces nécessite que toutes les interfaces backportées et toutes les classes les implémentent ou appellent leurs méthodes statiques à être backportées ensemble, avec une exécution de Retrolambda. En d'autres termes, vous devez toujours faire une construction propre. De plus, les méthodes par défaut de backporting ne fonctionneront pas sur les limites de module ou de dépendance.
- Peut se briser si un futur build JDK 8 cesse de générer une nouvelle classe pour chaque appel `invokedynamic` . Retrolambda fonctionne de sorte qu'il capture le bytecode que `java.lang.invoke.LambdaMetafactory` génère dynamiquement, de sorte que les optimisations à ce mécanisme peuvent casser Retrolambda.

Retrolambda gradle plugin va automatiquement construire votre projet Android avec Retrolambda. La dernière version peut être trouvée sur la page des versions .

Usage:

1. Téléchargez et installez jdk8
2. Ajoutez ce qui suit à votre `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
    }
}
```

Problèmes connus:

- Lint échoue sur les fichiers java contenant des lambdas. La charpie d'Android ne comprend

pas la syntaxe Java 8 et échouera silencieusement ou bruyamment. Il y a maintenant un fork expérimental qui corrige le problème.

- L'utilisation de Google Play Services entraîne l'échec de Retrolambda. La version 5.0.77 contient un bytecode incompatible avec Retrolambda. Cela devrait être résolu dans les nouvelles versions des services de lecture, si vous pouvez mettre à jour, cela devrait être la solution préférée. Pour contourner ce problème, vous pouvez utiliser une version antérieure telle que 4.4.52 ou ajouter `-noverify` aux `-noverify` jvm.

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

Lire Java sur Android en ligne: https://riptutorial.com/fr/android/topic/9223/java-sur-android

Chapitre 155: JCodec

Examples

Commencer

Vous pouvez obtenir JCodec automatiquement avec maven. Pour cela, ajoutez simplement l'extrait ci-dessous à votre fichier pom.xml.

```
<dependency>
    <groupId>org.jcodec</groupId>
    <artifactId>jcodec-javase</artifactId>
    <version>0.1.9</version>
</dependency>
```

Obtenir un cadre de film

Obtenir une seule image à partir d'un film (prend en charge uniquement AVC, H.264 dans MP4, ISO BMF, conteneur Quicktime):

```
int frameNumber = 150;
BufferedImage frame = FrameGrab.getFrame(new File("filename.mp4"), frameNumber);
ImageIO.write(frame, "png", new File("frame_150.png"));
```

Obtenir une séquence d'images à partir d'un film (prend en charge uniquement AVC, H.264 dans MP4, ISO BMF, conteneur Quicktime):

```
double startSec = 51.632;
FileChannelWrapper ch = null;
try {
    ch = NIOUtils.readableFileChannel(new File("filename.mp4"));
    FrameGrab fg = new FrameGrab(ch);
    grab.seek(startSec);
    for (int i = 0; i < 100; i++) {
        ImageIO.write(grab.getFrame(), "png",
            new File(System.getProperty("user.home"), String.format("Desktop/frame_%08d.png",
i)));
    }
} finally {
    NIOUtils.closeQuietly(ch);
}
```

Lire JCodec en ligne: https://riptutorial.com/fr/android/topic/9948/jcodec

Chapitre 156: Journalisation et utilisation de Logcat

Syntaxe

- `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)`

Paramètres

Option	La description
-b (tampon)	Charge un autre tampon de journal pour l'affichage, tel que des événements ou la radio. Le tampon principal est utilisé par défaut. Voir Affichage des tampons de journal alternatifs.
-c	Efface (vidange) le journal entier et quitte.
-ré	Vide le journal à l'écran et quitte.
-f (nom de fichier)	Écrit le résultat du journal dans (nomfichier). La valeur par défaut est stdout.
-g	Imprime la taille du tampon de journal et des exits spécifiés.
-n (compte)	Définit le nombre maximal de journaux pivotés sur (count). La valeur par défaut est 4. Nécessite l'option -r.
-r (kbytes)	Fait pivoter le fichier journal tous les kilo-octets de sortie. La valeur par défaut est 16. Nécessite l'option -f.
-s	Définit la spécification de filtre par défaut sur silence.
-v (format)	Définit le format de sortie pour les messages de journal. Le format par défaut est bref.

Remarques

Définition

Logcat est un outil de ligne de commande qui affiche un journal des messages système, y compris des traces de pile lorsque le périphérique génère une erreur et des messages que vous avez écrits à partir de votre application avec la classe Log .

Quand utiliser

Si vous envisagez d'utiliser les méthodes System.out de Java pour imprimer sur la console au lieu d'utiliser l'une des méthodes Log d'Android, vous devez savoir qu'elles fonctionnent essentiellement de la même manière. Cependant, il est préférable d'éviter d'utiliser les méthodes de Java car les informations et la mise en forme supplémentaires fournies par les méthodes Log d'Android sont plus bénéfiques. De plus, les méthodes d'impression System.out sont redirigées vers la méthode `Log.i()` .

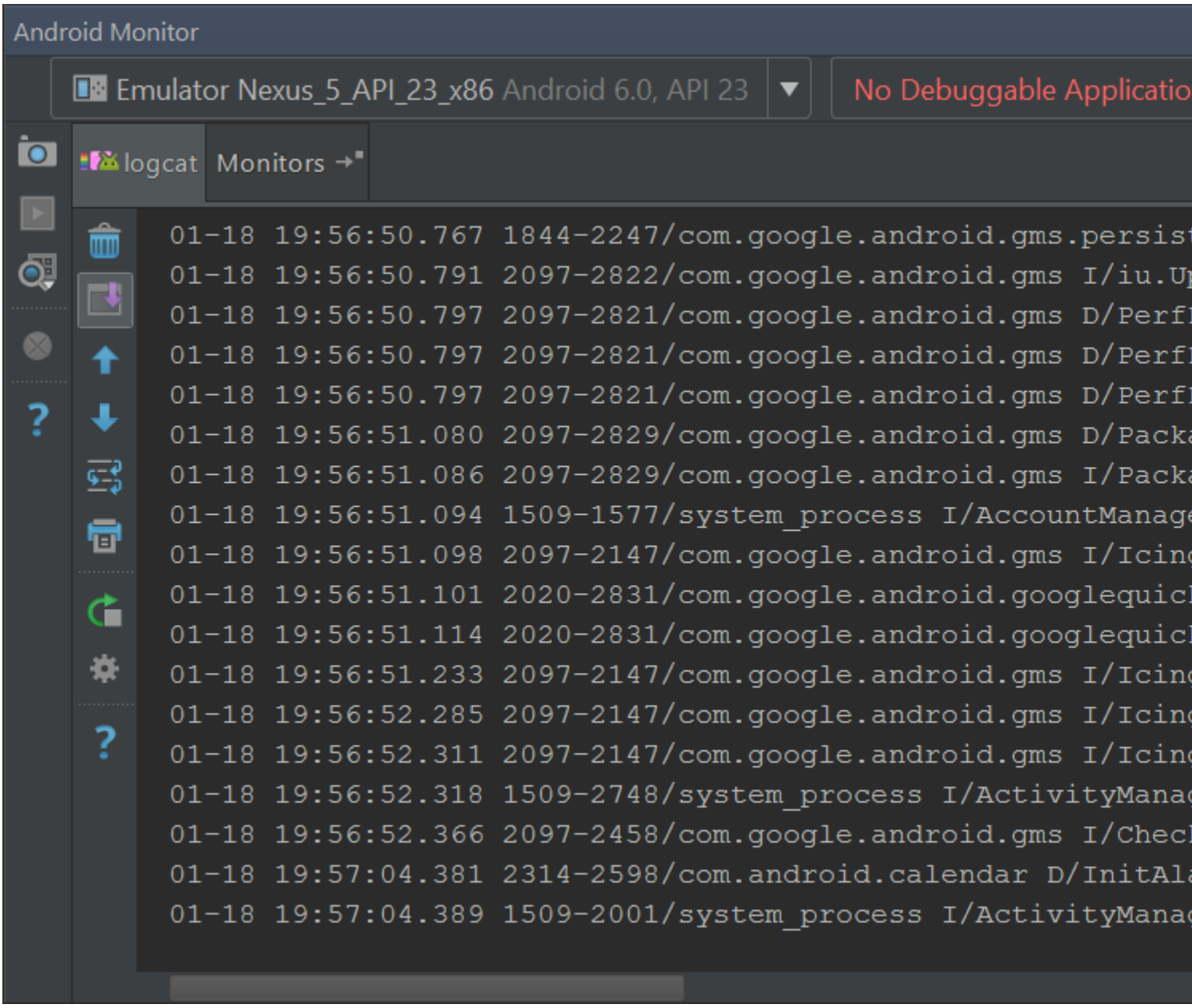
Liens utiles

- Documentation officielle du développeur Android pour Log et logcat .
- Stackoveflow Question: Android Log.v (), Log.d (), Log.i (), Log.w (), Log.e () - Quand utiliser chacun d'eux?

Examples

Filtrage de la sortie logcat

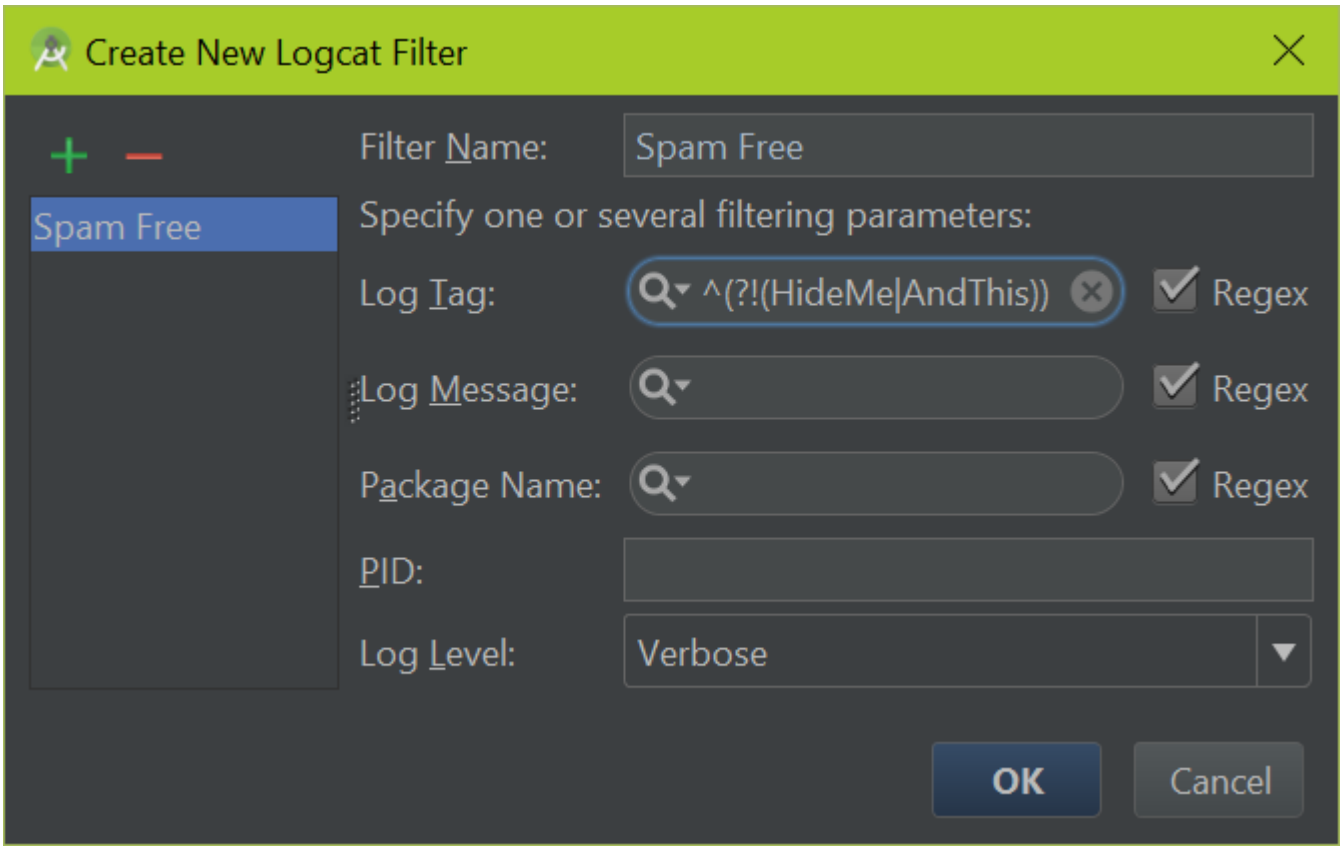
Il est utile de filtrer la sortie logcat car il existe de nombreux messages sans intérêt. Pour filtrer la sortie, ouvrez le "moniteur Android" et cliquez sur le menu déroulant en haut à droite et sélectionnez *Modifier la configuration du filtre*



Vous pouvez maintenant ajouter des filtres personnalisés pour afficher les messages qui vous intéressent, ainsi que filtrer les lignes de journal connues qui peuvent être ignorées en toute sécurité. Pour ignorer une partie de la sortie, vous pouvez définir une expression régulière . Voici un exemple d'exclusion de balises correspondantes:

```
^(?!(HideMe|AndThis))
```

Cela peut être saisi en suivant cet exemple:



Ce qui précède est une expression régulière qui exclut les entrées. Si vous souhaitez ajouter un autre tag à la *liste noire* , ajoutez-le après un tube `|` personnage. Par exemple, si vous souhaitez mettre en liste noire "GC", vous utiliseriez un filtre comme celui-ci:

```
^(?!(HideMe|AndThis|GC))
```

Pour plus de documentation et d'exemples, consultez la section Journalisation et utilisation de Logcat.

Enregistrement

Toute application Android de qualité gardera une trace de ce qu'elle fait via les journaux d'application. Ces journaux facilitent le débogage pour aider le développeur à diagnostiquer ce qui se passe avec l'application. La documentation complète d'Android peut être trouvée ici , mais un résumé suit:

Enregistrement de base

La classe `Log` est la principale source d'écriture des journaux de développement, en spécifiant une `tag` et un `message` . La balise est ce que vous pouvez utiliser pour filtrer les messages de journal en identifiant les lignes provenant de votre activité particulière. Il suffit d'appeler

```
Log.v(String tag, String msg);
```

Et le système Android écrira un message au logcat:

```
07-28 12:00:00.759 24812-24839/my.packagename V/MyAnimator: Some log messages
 └ time stamp             |  app.package┘     |    └ any tag  |
    process & thread ids ┘          log level┘               └ the log message
```

> POINTE:
> Notez l'ID de processus et l'ID de thread. S'ils sont identiques, le journal provient du thread principal / de l'interface utilisateur!

N'importe quelle balise peut être utilisée, mais il est courant d'utiliser le nom de la classe en tant que balise:

```
public static final String tag = MyAnimator.class.getSimpleName();
```

Niveaux de journal

Le logger Android dispose de 6 niveaux différents, chacun servant un objectif spécifique:

- `ERROR` : `Log.e()`
  - Utilisé pour indiquer une défaillance critique, il s'agit du niveau imprimé lors du lancement d'une `Exception` .
- `WARN` : `Log.w()`
  - Utilisé pour indiquer un avertissement, principalement pour les défaillances récupérables
- `INFO` : `Log.i()`
  - Utilisé pour indiquer des informations de niveau supérieur sur l'état de l'application
- `DEBUG` : `Log.d()`
  - Utilisé pour enregistrer des informations qui seraient utiles lors du débogage de l'application, mais qui entraveraient l'exécution de l'application
- `VERBOSE` : `Log.v()`
  - Utilisé pour enregistrer des informations qui reflètent les petits détails sur l'état de l'application
- `ASSERT` : `Log.wtf()`
  - Utilisé pour enregistrer des informations sur une condition qui ne devrait jamais se produire.
  - *wtf* signifie "What a Terrible Failure".

Motivation pour l'enregistrement

La motivation pour la journalisation est de trouver facilement les erreurs, les avertissements et autres informations en parcourant la chaîne des événements de l'application. Par exemple, imaginez une application qui lit les lignes d'un fichier texte, mais suppose à tort que le fichier ne sera jamais vide. La trace du journal (d'une application qui ne se connecte pas) ressemblerait à ceci:

```
E/MyApplication: Process: com.example.myapplication, PID: 25788
                         com.example.SomeRandomException: Expected string, got 'null' instead
```

Suivi par un tas de traces de pile qui mèneraient éventuellement à la ligne incriminée, où passer en revue avec un débogueur finirait par poser le problème

Cependant, la trace de journal d'une application avec la journalisation activée pourrait ressembler à ceci:

```
V/MyApplication: Looking for file myFile.txt on the SD card
D/MyApplication: Found file myFile.txt at path <path>
V/MyApplication: Opening file myFile.txt
D/MyApplication: Finished reading myFile.txt, found 0 lines
V/MyApplication: Closing file myFile.txt
...
E/MyApplication: Process: com.example.myapplication, PID: 25788
                         com.example.SomeRandomException: Expected string, got 'null' instead
```

Un rapide coup d’œil aux journaux et il est évident que le fichier était vide.

Choses à considérer lors de la journalisation:

Bien que la journalisation soit un outil puissant permettant aux développeurs Android de mieux comprendre le fonctionnement interne de leur application, la journalisation présente certains inconvénients.

La lisibilité du journal:

Il est courant que les applications Android exécutent plusieurs journaux de manière synchrone. En tant que tel, il est très important que chaque journal soit facilement lisible et ne contienne que des informations pertinentes et nécessaires.

Performance:

La journalisation nécessite une petite quantité de ressources système. En général, cela ne pose pas de problème, cependant, en cas de surutilisation, la journalisation peut avoir un impact négatif sur les performances des applications.

Sécurité:

Récemment, plusieurs applications Android ont été ajoutées au marché Google Play pour permettre à l'utilisateur de consulter les journaux de toutes les applications en cours d'exécution. Cet affichage involontaire de données peut permettre aux utilisateurs d'afficher des informations confidentielles. En règle générale, supprimez toujours les journaux contenant des données non publiques *avant de* publier votre application sur le marché.

Conclusion:

La journalisation est une partie essentielle d'une application Android, en raison de la puissance qu'elle donne aux développeurs. La possibilité de créer une trace de journal utile est l'un des aspects les plus difficiles du développement logiciel, mais la classe Log d'Android facilite grandement la tâche.

Pour plus de documentation et d'exemples, consultez la section Journalisation et utilisation de Logcat.

Journal avec un lien vers la source directement depuis Logcat

C'est un bon truc pour ajouter un lien au code, il sera donc facile de passer directement au code qui a généré le journal.

Avec le code suivant, cet appel:

```
MyLogger.logWithLink("MyTag","param="+param);
```

Aura pour résultat:

```
07-26...012/com.myapp D/MyTag: MyFrag:onStart(param=3)  (MyFrag.java:2366) // << logcat
converts this to a link to source!
```

Voici le code (dans une classe appelée 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());
}
```

Ceci est un exemple élémentaire, il peut être facilement étendu pour créer un lien vers l'appelant (conseil: la pile sera [4] au lieu de [3]), et vous pouvez également ajouter d'autres informations pertinentes.

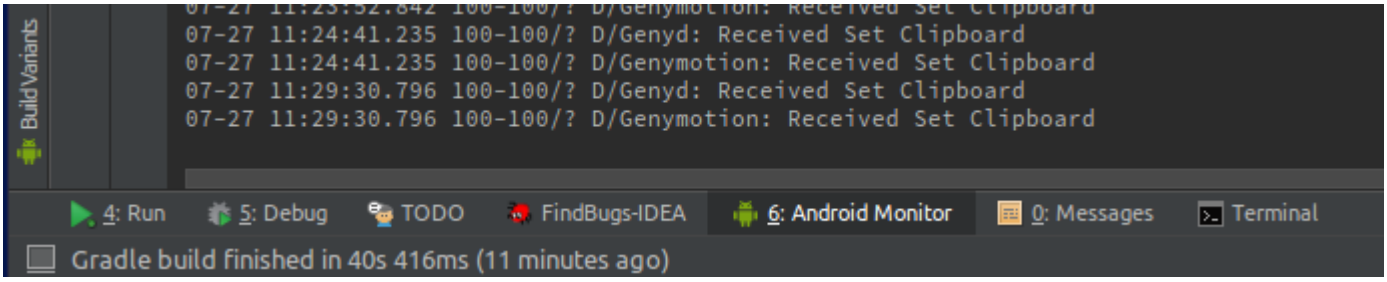
Utiliser le logcat

Logcat est un outil de ligne de commande qui affiche un journal des messages système, y compris des traces de pile lorsque le périphérique génère une erreur et des messages que vous avez écrits à partir de votre application avec la classe Log.

La sortie Logcat peut être affichée dans le moniteur Android d'Android Studio ou avec la ligne de commande adb.

Dans Android Studio

Afficher en cliquant sur l'icône "Android Monitor":



Ou en appuyant sur `Alt` + `6` sous Windows / Linux ou `CMD` + `6` sur Mac.

via la ligne de commande:

Utilisation simple:

```
$ adb logcat
```

Avec timestamps:

```
$ adb logcat -v time
```

Filtrer sur un texte spécifique:

```
$ adb logcat -v time | grep 'searchtext'
```

Il existe de nombreuses options et filtres disponibles pour *logcat en ligne de commande* , documentés ici .
Un exemple simple mais utile est l'expression de filtre suivante qui affiche tous les messages de journal avec le niveau de priorité "erreur", sur toutes les balises:

```
$ adb logcat *:E
```

Générer un code de journalisation

`Live templates Android Studio` peuvent offrir plusieurs raccourcis pour une journalisation rapide. Pour utiliser des modèles Live, il vous suffit de commencer à taper le nom du modèle et d' `TAB` sur `TAB` ou d'entrer pour insérer l'instruction.

Exemples:

- `logi` → devient → `android.util.Log.i(TAG, "$METHOD_NAME$: $content$");`
  - `$METHOD_NAME$` sera automatiquement remplacé par le nom de votre méthode et le curseur attendra que le contenu soit rempli.
- `loge` → même, pour erreur
- etc. pour le reste des niveaux de journalisation.

La liste complète des modèles se trouve dans `Android Studio` paramètres d' `Android Studio` (`ALT` + `s` et tapez "live"). Et il est également possible d'ajouter vos modèles personnalisés.

Si vous trouvez `Android Studio Live templates Android Studio` insuffisants pour vos besoins, vous pouvez envisager le plugin Android Postfix

Ceci est une bibliothèque très utile qui vous aide à éviter d'écrire manuellement la ligne de journalisation.

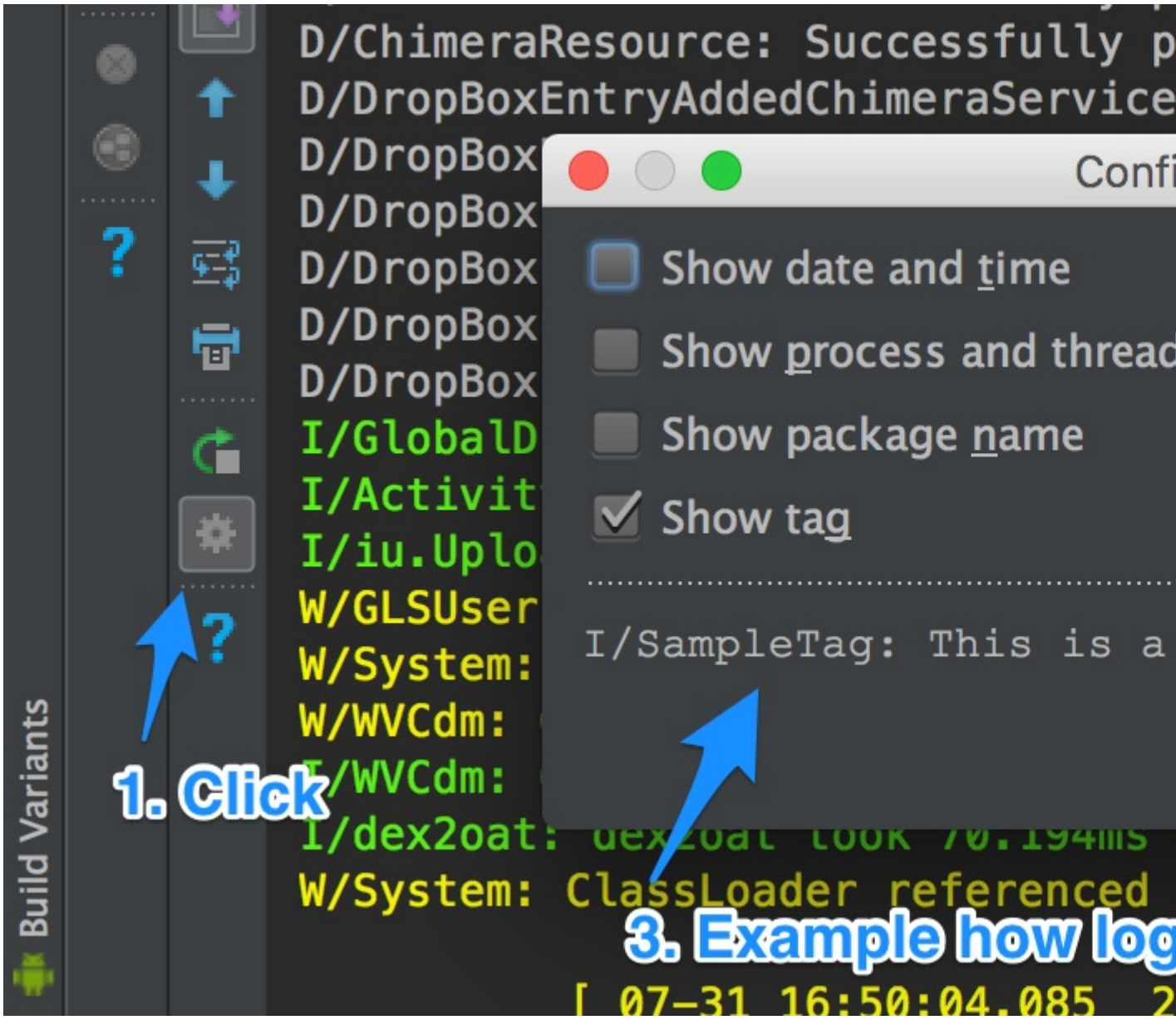
La syntaxe est absolument simple:

.log - Journalisation. S'il existe une variable constante "TAG", il utilise "TAG". Sinon, utilisez le nom de la classe.

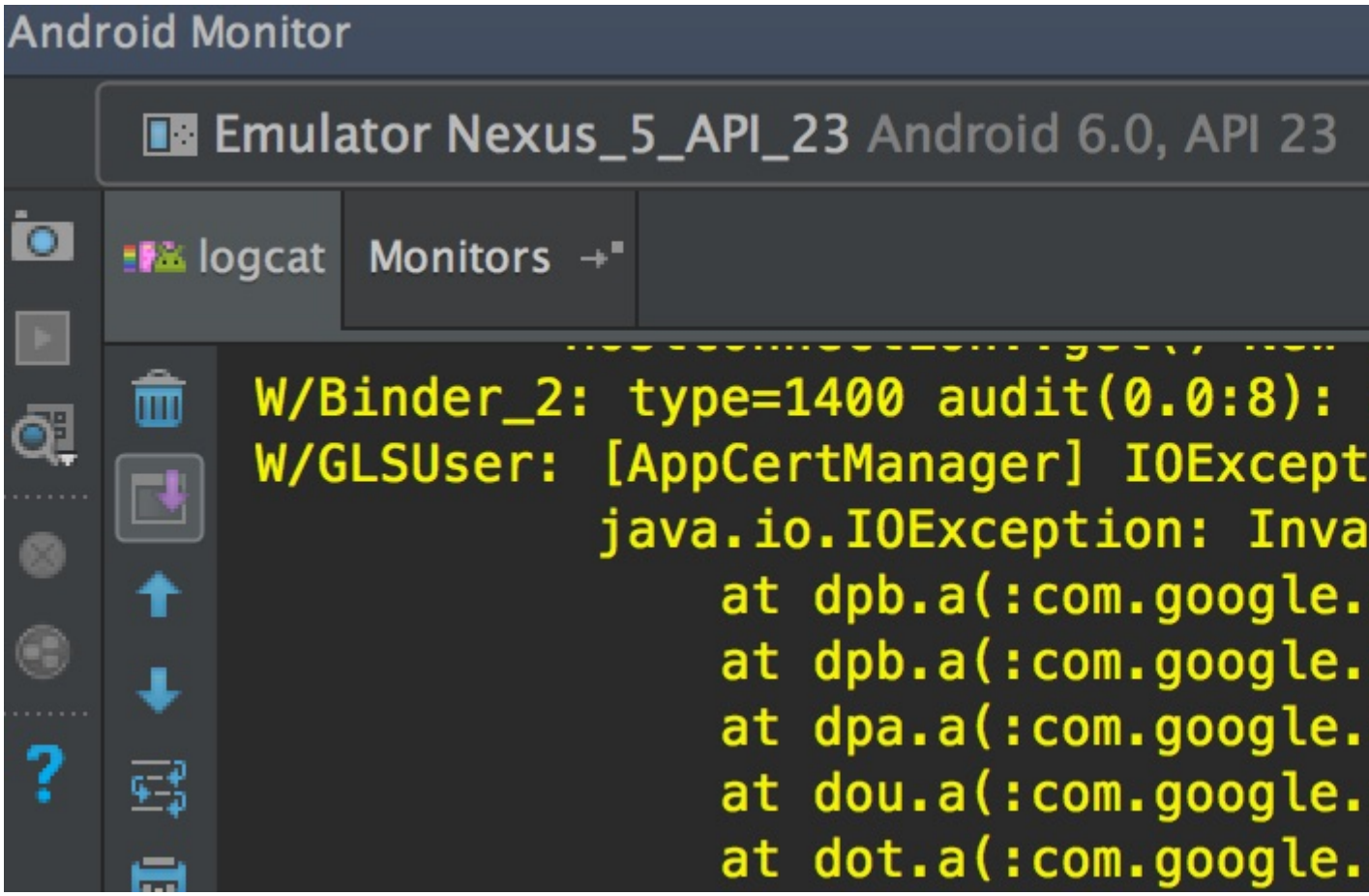
```
ublic class MyActivity extends Activity {
    static final String T= "test";
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        new View.OnClickListener() {
            @Override
            public void onClick(View view) {

            }
        };
    }
```

Utilisation d'Android Studio

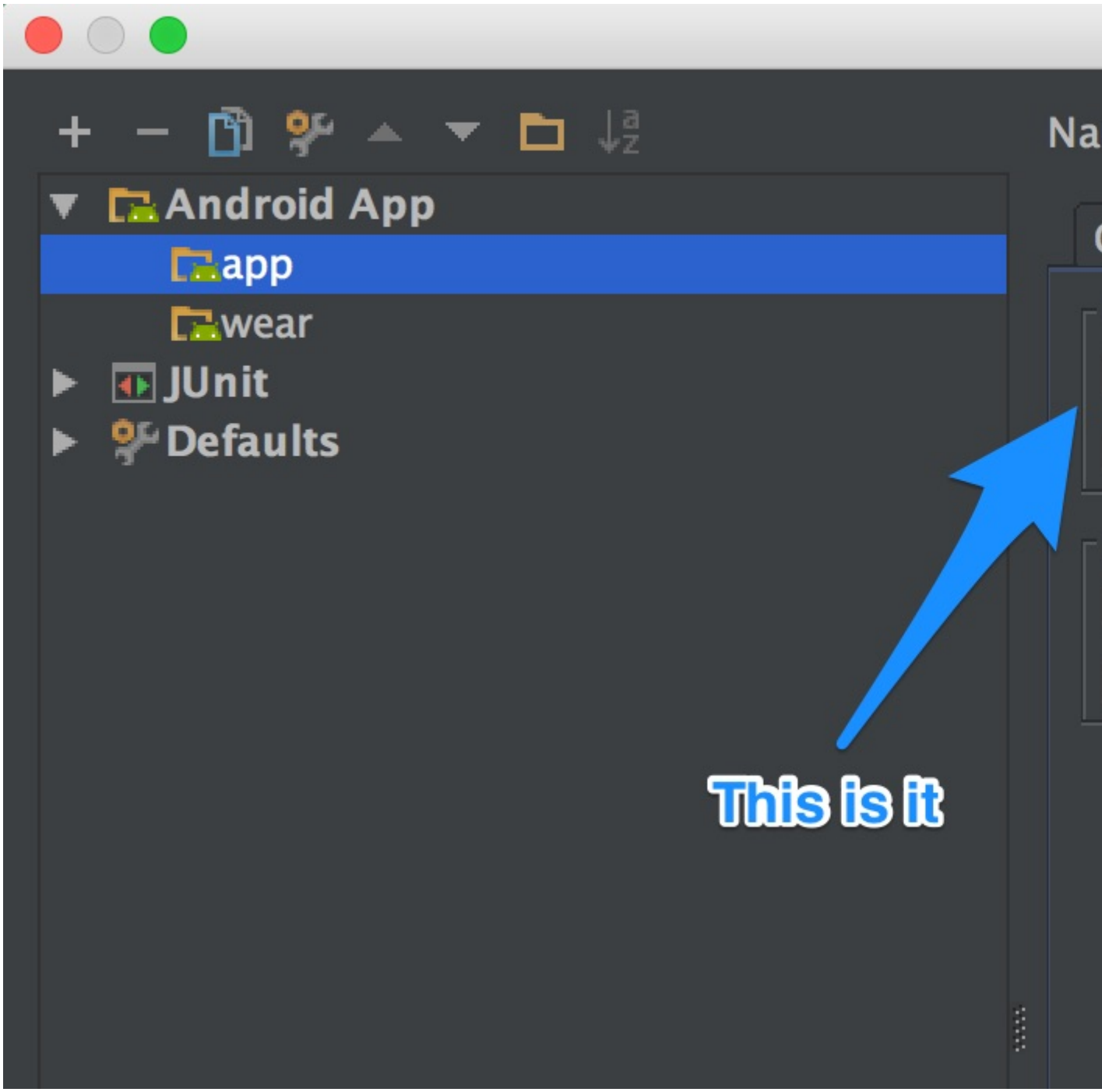
1. Masquer / afficher les informations imprimées:



2. Contrôle de la verbosité de la journalisation:



3. Désactiver / activer l'ouverture de la fenêtre de journal au démarrage de l'application run / debug



Effacer les journaux

Pour effacer (rincer) le journal entier:

```
adb logcat -c
```

Lire Journalisation et utilisation de Logcat en ligne:
https://riptutorial.com/fr/android/topic/1552/journalisation-et-utilisation-de-logcat

Chapitre 157: JSON dans Android avec org.json

Syntaxe

- **Object** : un objet est un ensemble non ordonné de paires nom / valeur. Un objet commence par {(accolade gauche) et se termine par} (accolade droite). Chaque nom est suivi de: (deux points) et les paires nom / valeur sont séparées par (virgule).
- **Tableau** : un tableau est une collection ordonnée de valeurs. Un tableau commence par [(crochet gauche) et se termine par] (crochet droit). Les valeurs sont séparées par, (virgule).
- **Valeur** : Une valeur peut être une chaîne entre guillemets, ou un nombre, ou vrai ou faux ou null, ou un objet ou un tableau. Ces structures peuvent être imbriquées.
- **Chaîne** : une chaîne est une séquence de zéro ou plusieurs caractères Unicode, entourés de guillemets doubles, à l'aide de barres obliques inverses. Un caractère est représenté par une chaîne de caractères unique. Une chaîne ressemble beaucoup à une chaîne C ou Java.
- **Number** : Un nombre ressemble beaucoup à un numéro C ou Java, sauf que les formats octal et hexadécimal ne sont pas utilisés.

Remarques

Cette rubrique concerne l'utilisation du package `org.json` inclus dans le SDK Android.

Examples

Analyser un objet JSON simple

Considérez la chaîne JSON suivante:

```
{
  "title": "test",
  "content": "Hello World!!!",
  "year": 2016,
  "names" : [
        "Hannah",
        "David",
        "Steve"
   ]
}
```

Cet objet JSON peut être analysé à l'aide du code suivant:

```
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());
}
```

Voici un autre exemple avec un JSONArray imbriqué dans JSONObject:

```
{
    "books":[
      {
        "title":"Android JSON Parsing",
        "times_sold":186
      }
    ]
}
```

Cela peut être analysé avec le code suivant:

```
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");
```

Créer un objet JSON simple

Créez l'objet `JSONObject` à l'aide du constructeur vide et ajoutez des champs à l'aide de la méthode `put()` , qui est surchargée pour pouvoir être utilisée avec différents types:

```
try {
    // Create a new instance of a JSONObject
    final JSONObject object = new JSONObject();

    // With put you can add a name/value pair to the JSONObject
    object.put("name", "test");
    object.put("content", "Hello World!!!1");
    object.put("year", 2016);
    object.put("value", 3.23);
    object.put("member", true);
    object.put("null_value", JSONObject.NULL);

    // Calling toString() on the JSONObject returns the JSON in string format.
    final String json = object.toString();

} catch (JSONException e) {
    Log.e(TAG, "Failed to create JSONObject", e);
}
```

La chaîne `JSON` résultante ressemble à ceci:

```
{
   "name":"test",
   "content":"Hello World!!!1",
   "year":2016,
   "value":3.23,
   "member":true,
   "null_value":null
}
```

Ajouter JSONArray à JSONObject

```
// Create a new instance of a JSONArray
JSONArray array = new JSONArray();

// With put() you can add a value to the array.
array.put("ASDF");
array.put("QWERTY");

// Create a new instance of a JSONObject
JSONObject obj = new JSONObject();

try {
    // Add the JSONArray to the JSONObject
    obj.put("the_array", array);
} catch (JSONException e) {
    e.printStackTrace();
}

String json = obj.toString();
```

La chaîne JSON résultante ressemble à ceci:

```
{
   "the_array":[
      "ASDF",
      "QWERTY"
   ]
}
```

Créez une chaîne JSON avec une valeur nulle.

Si vous devez produire une chaîne JSON avec une valeur `null` comme ceci:

```
{
   "name":null
}
```

Ensuite, vous devez utiliser la constante spéciale JSONObject.NULL .

Exemple de fonctionnement:

```
jsonObject.put("name", JSONObject.NULL);
```

Travailler avec une chaîne vide lors de l'analyse syntaxique de json

```
{
    "some_string": null,
    "ather_string": "something"
}
```

Si nous allons utiliser cette façon:

```
JSONObject json = new JSONObject(jsonStr);
String someString = json.optString("some_string");
```

Nous aurons une sortie:

```
someString = "null";
```

Nous devons donc fournir cette solution de contournement:

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

Et puis appelez:

```
JSONObject json = new JSONObject(jsonStr);
String someString = optNullableString(json, "some_string");
String someString2 = optNullableString(json, "some_string", "");
```

Et nous aurons la sortie comme prévu:

```
someString = null; //not "null"
someString2 = "";
```

Utiliser JsonReader pour lire JSON depuis un flux

`JsonReader` lit une valeur codée JSON sous forme de flux de jetons.

```
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);
  }
```

Créer un objet JSON imbriqué

Pour produire un objet JSON imbriqué, vous devez simplement ajouter un objet JSON à un autre:

```
JSONObject mainObject = new JSONObject();            // Host object
JSONObject requestObject = new JSONObject();         // Included object

try {
    requestObject.put("lastname", lastname);
    requestObject.put("phone", phone);
    requestObject.put("latitude", lat);
    requestObject.put("longitude", lon);
    requestObject.put("theme", theme);
    requestObject.put("text", message);

    mainObject.put("claim", requestObject);
} catch (JSONException e) {
    return "JSON Error";
}
```

`mainObject` contient `mainObject` une clé appelée `claim` avec la totalité de `requestObject` comme valeur.

Gestion de la clé dynamique pour la réponse JSON

Ceci est un exemple sur la façon de gérer la clé dynamique pour la réponse. Ici, `A` et `B` sont des clés dynamiques, cela peut être n'importe quoi

Réponse

```
{
  "response": [
    {
      "A": [
        {
          "name": "Tango"
```

```
        },
        {
          "name": "Ping"
        }
      ],
      "B": [
        {
          "name": "Jon"
        },
        {
          "name": "Mark"
        }
      ]
    }
  ]
}
```

Code Java

```
// ResponseData is raw string of response
JSONObject responseDataObj = new JSONObject(responseData);
JSONArray responseArray = responseDataObj.getJSONArray("response");
for (int i = 0; i < responseArray.length(); i++) {
    // Nodes ArrayList<ArrayList<String>> declared globally
    nodes = new ArrayList<ArrayList<String>>();
    JSONObject obj = responseArray.getJSONObject(i);
    Iterator keys = obj.keys();
    while(keys.hasNext()) {
       // Loop to get the dynamic key
       String currentDynamicKey = (String)keys.next();
       // Get the value of the dynamic key
       JSONArray currentDynamicValue = obj.getJSONArray(currentDynamicKey);
       int jsonArraySize = currentDynamicValue.length();
       if(jsonArraySize > 0) {
           for (int ii = 0; ii < jsonArraySize; ii++) {
                // NameList ArrayList<String> declared globally
                nameList = new ArrayList<String>();
               if(ii == 0) {
                JSONObject nameObj = currentDynamicValue.getJSONObject(ii);
                String name = nameObj.getString("name");
                System.out.print("Name = " + name);
                // Store name in an array list
                nameList.add(name);
              }
           }
       }
     nodes.add(nameList);
    }
}
```

Vérifier l'existence de champs sur JSON

Parfois, il est utile de vérifier si un champ est présent ou absent sur votre JSON pour éviter une `JSONException` sur votre code.

Pour cela, utilisez la `JSONObject#has(String)` ou la méthode, comme dans l'exemple suivant:

Exemple de JSON

```
{
   "name":"James"
}
```

Code Java

```
String jsonStr = " { \"name\":\"James\" }";
JSONObject json = new JSONObject(jsonStr);
// Check if the field "name" is present
String name, surname;

// This will be true, since the field "name" is present on our JSON.
if (json.has("name")) {
    name = json.getString("name");
}
else {
    name = "John";
}
// This will be false, since our JSON doesn't have the field "surname".
if (json.has("surname")) {
    surname = json.getString("surname");
}
else {
    surname = "Doe";
}

// Here name == "James" and surname == "Doe".
```

Mise à jour des éléments dans le JSON

échantillon json pour mettre à jour

```
 {
 "student":{"name":"Rahul", "lastname":"sharma"},
 "marks":{"maths":"88"}
 }
```

Pour mettre à jour la valeur des éléments dans json, nous devons affecter la valeur et la mise à jour.

```
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);
}
```

valeur actualisée

```
 {
 "student":{"name":"Kumar", "lastname":"sharma"},
 "marks":{"maths":"88"}
 }
```

Lire JSON dans Android avec org.json en ligne: https://riptutorial.com/fr/android/topic/106/json-dans-android-avec-org-json

Chapitre 158: Le contexte

Introduction

Selon la documentation Google: "Interface vers des informations globales sur un environnement d'application. Elle permet d'accéder à des ressources et des classes spécifiques à l'application, ainsi qu'à des appels pour des opérations telles que des activités de lancement, de diffusion et de réception, etc."

Plus simplement, le contexte est l'état actuel de votre application. Il vous permet de fournir des informations aux objets afin qu'ils puissent être au courant de ce qui se passe dans d'autres parties de votre application.

Syntaxe

- `getApplicationContext()`
- `getBaseContext()`
- `getContext()`
- `this`

Remarques

Cette page StackOverflow contient plusieurs explications complètes et bien écrites du concept de contexte:

Qu'est-ce que le contexte?

Examples

Exemples de base

Utilisation standard dans l'activité:

```
Context context = getApplicationContext();
```

Utilisation standard en fragment:

```
Context context = getActivity().getApplicationContext();
```

`this` (dans une classe qui s'étend du contexte, comme les classes Application, Activity, Service et IntentService)

```
TextView textView = new TextView(this);
```

un autre `this` exemple:

```
Intent intent = new Intent(this, MainActivity.class);
startActivity(intent);
```

Lire Le contexte en ligne: https://riptutorial.com/fr/android/topic/9774/le-contexte

Chapitre 159: Le fichier manifeste

Introduction

Le manifeste est un fichier obligatoire nommé exactement "AndroidManifest.xml" et situé dans le répertoire racine de l'application. Il spécifie le nom de l'application, l'icône, le nom du package Java, la version, la déclaration des activités, les services, les autorisations d'application et d'autres informations.

Examples

Déclaration de composants

La tâche principale du manifeste consiste à informer le système des composants de l'application. Par exemple, un fichier manifeste peut déclarer une activité comme suit:

```
<?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>
```

Dans l'élément `<application>` , l'attribut `android:icon` pointe vers les ressources pour une icône identifiant l'application.

Dans l'élément, l'attribut `android:name` spécifie le nom de classe complet de la sous-classe Activity et l'attribut android: label spécifie une chaîne à utiliser comme étiquette visible par l'utilisateur pour l'activité.

Vous devez déclarer tous les composants de l'application de cette manière:

- éléments `<activity>` pour activités

- éléments `<service>` pour les services

- Éléments `<receiver>` pour récepteurs de diffusion

- Éléments `<provider>` pour les fournisseurs de contenu

Les activités, services et fournisseurs de contenu que vous incluez dans votre source mais ne déclarent pas dans le manifeste ne sont pas visibles pour le système et ne peuvent donc jamais être exécutés. Toutefois, les récepteurs de diffusion peuvent être soit déclarés dans le manifeste, soit créés dynamiquement dans le code (objets `BroadcastReceiver`) et enregistrés avec le système en appelant `registerReceiver()` .

Pour plus d'informations sur la structure du fichier manifeste pour votre application, voir la documentation sur le fichier AndroidManifest.xml.

Déclaration des autorisations dans votre fichier manifeste

Toute autorisation requise par votre application pour accéder à une partie protégée de l'API ou pour interagir avec d'autres applications doit être déclarée dans votre fichier `AndroidManifest.xml` . Ceci est fait en utilisant la `<uses-permission />` .

Syntaxe

```
<uses-permission android:name="string"
   android:maxSdkVersion="integer"/>
```

android: name: c'est le nom de l'autorisation requise

android: maxSdkVersion: le niveau d'API le plus élevé auquel cette autorisation doit être accordée pour votre application. La définition de cette autorisation est facultative et ne doit être définie que si l'autorisation requise par votre application n'est plus nécessaire à un certain niveau d'API.

Exemple d'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>
```

* Voir aussi la rubrique Autorisations .

Lire Le fichier manifeste en ligne: https://riptutorial.com/fr/android/topic/1848/le-fichier-manifeste

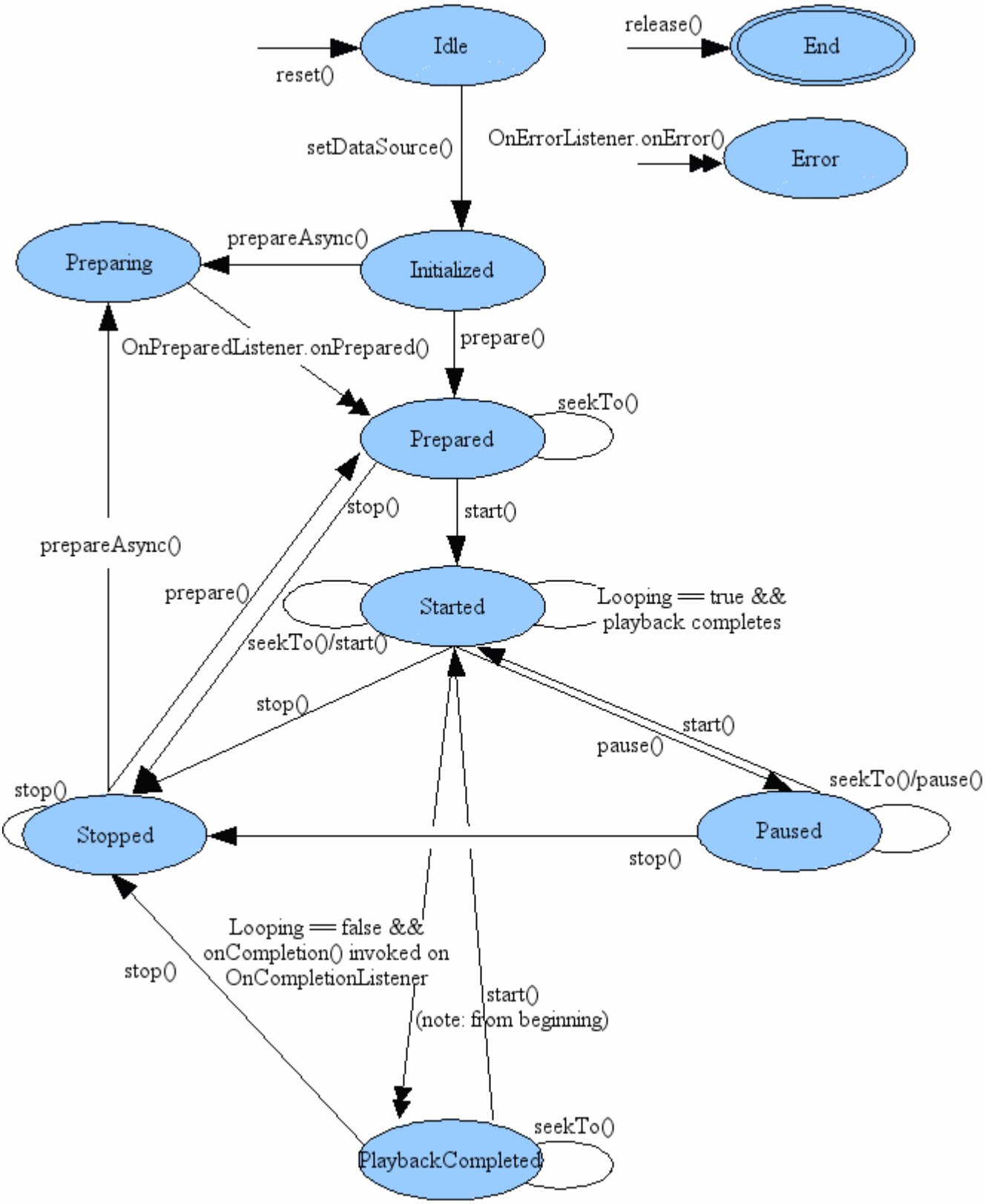
Chapitre 160: Lecteur multimédia

Syntaxe

- void setAudioStreamType (int streamtype)
- void setDataSource (Contexte contextuel, Uri uri)
- annuler préparer ()
- annuler prepareAsync ()
- départ nul ()
- annuler l'annulation ()

Remarques

L'utilisation de `MediaPlayer` est principalement basée sur le diagramme d'état:



Cela signifie que pour jouer de l'audio / vidéo, une séquence d'action définie doit avoir lieu dans un ordre spécifique. Il indique également quelles actions peuvent être effectuées dans quel état .

L'API MediaPlayer manque de flexibilité (ajout d'un décodeur et d'une logique de rendu personnalisés) et ne prend pas en charge sSupport Dynamic Streaming Adaptative sur HTTP (SmashStreaming). Pour cela, regardez dans ExoPlayer .

Examples

Création de base et jeu

La classe MediaPlayer peut être utilisée pour contrôler la lecture des fichiers audio / vidéo et des flux.

La création de l'objet MediaPlayer peut être de trois types:

1. Média de ressource locale

```
MediaPlayer mediaPlayer = MediaPlayer.create(context, R.raw.resource);
mediaPlayer.start(); // no need to call prepare(); create() does that for you
```

2. De l'URI local (obtenu à partir de ContentResolver)

```
Uri myUri = ....; // initialize Uri here
MediaPlayer mediaPlayer = new MediaPlayer();
mediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
mediaPlayer.setDataSource(getApplicationContext(), myUri);
mediaPlayer.prepare();
mediaPlayer.start();
```

3. À partir d'une URL externe

```
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 préparer

Le `MediaPlayer$prepare()` est un appel bloquant qui gèle l'interface utilisateur jusqu'à la fin de l'exécution. Pour résoudre ce problème, `MediaPlayer$prepareAsync()` peut être utilisé.

```
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
```

Sur les opérations synchrones, les erreurs sont normalement signalées par une exception ou un code d'erreur, mais chaque fois que vous utilisez des ressources asynchrones, vous devez vous assurer que votre application est correctement informée des erreurs. Pour 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
        }
});
```

Obtenir des sonneries système

Cet exemple montre comment récupérer l'URI des sonneries système (`RingtoneManager.TYPE_RINGTONE`):

```
private List<Uri> loadLocalRingtonesUris() {
        List<Uri> alarms = new ArrayList<>();
        try {
            RingtoneManager ringtoneMgr = new RingtoneManager(getActivity());
            ringtoneMgr.setType(RingtoneManager.TYPE_RINGTONE);

            Cursor alarmsCursor = ringtoneMgr.getCursor();
            int alarmsCount = alarmsCursor.getCount();
            if (alarmsCount == 0 && !alarmsCursor.moveToFirst()) {
                alarmsCursor.close();
                return null;
            }

            while (!alarmsCursor.isAfterLast() && alarmsCursor.moveToNext()) {
                int currentPosition = alarmsCursor.getPosition();
                alarms.add(ringtoneMgr.getRingtoneUri(currentPosition));
            }

        } catch (Exception ex) {
            ex.printStackTrace();
        }

        return alarms;
    }
```

La liste dépend des types de sonneries demandés. Les possibilités sont les suivantes:

- `RingtoneManager.TYPE_RINGTONE`
- `RingtoneManager.TYPE_NOTIFICATION`
- `RingtoneManager.TYPE_ALARM`
- `RingtoneManager.TYPE_ALL = TYPE_RINGTONE | TYPE_NOTIFICATION | TYPE_ALARM`

Pour obtenir les sonneries comme `android.media.Ringtone` chaque `Uri` doit être résolu par le `RingtoneManager` :

```
android.media.Ringtone osRingtone = RingtoneManager.getRingtone(context, uri);
```

Pour jouer le son, utilisez la méthode:

```
public void setDataSource(Context context, Uri uri)
```

depuis `android.media.MediaPlayer` . `MediaPlayer` doit être initialisé et préparé conformément au diagramme d'état

Obtenir et définir le volume du système

Types de flux audio

Il existe différents profils de flux de sonnerie. Chacun d'eux a son volume différent.

Chaque exemple ici est écrit pour le type de flux `AudioManager.STREAM_RING` . Cependant, ce n'est pas le seul. Les types de flux disponibles sont les suivants:

- `STREAM_ALARM`
- `STREAM_DTMF`
- `STREAM_MUSIC`
- `STREAM_NOTIFICATION`
- `STREAM_RING`
- `STREAM_SYSTEM`
- `STREAM_VOICE_CALL`

Réglage du volume

Pour obtenir le volume du profil spécifique, appelez:

```
AudioManager audio = (AudioManager) getActivity().getSystemService(Context.AUDIO_SERVICE);
int currentVolume = audioManager.getStreamVolume(AudioManager.STREAM_RING);
```

Cette valeur est très peu utile lorsque la valeur maximale du flux n'est pas connue:

```
AudioManager audio = (AudioManager) getActivity().getSystemService(Context.AUDIO_SERVICE);
int streamMaxVolume = audioManager.getStreamMaxVolume(AudioManager.STREAM_RING);
```

Le rapport de ces deux valeurs donnera un volume relatif (0 <volume <1):

```
float volume = ((float) currentVolume) / streamMaxVolume
```

Réglage du volume d'un pas

Pour augmenter le volume du flux d'un pas, appelez:

```
AudioManager audio = (AudioManager) getActivity().getSystemService(Context.AUDIO_SERVICE);
audio.adjustStreamVolume(AudioManager.STREAM_RING, AudioManager.ADJUST_RAISE, 0);
```

Pour réduire le volume du flux d'un pas, appelez:

```
AudioManager audio = (AudioManager) getActivity().getSystemService(Context.AUDIO_SERVICE);
audio.adjustStreamVolume(AudioManager.STREAM_RING, AudioManager.ADJUST_LOWER, 0);
```

Définition de MediaPlayer pour utiliser un type de flux spécifique

Pour ce faire, il existe une fonction d'assistance de la classe `MediaPlayer` .
Appelez simplement `void setAudioStreamType(int streamtype)` :

```
MediaPlayer mMedia = new MediaPlayer();
mMedia.setAudioStreamType(AudioManager.STREAM_RING);
```

Media Player avec progression de la mémoire tampon et position de lecture

```
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();
```

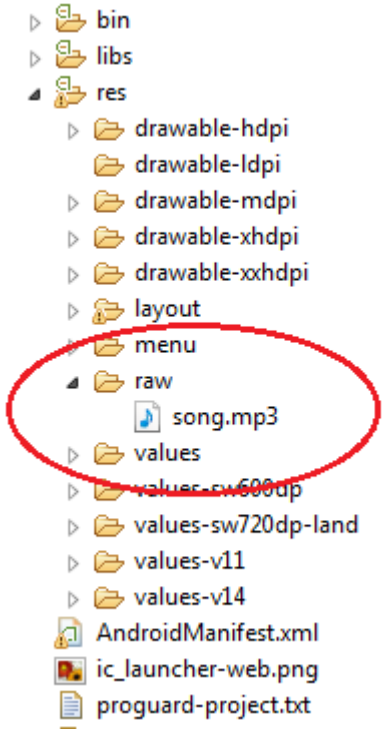
```
                mediaPlayer.reset();
            }
        });
        mediaPlayer.setOnBufferingUpdateListener(new MediaPlayer.OnBufferingUpdateListener() {
            @Override
            public void onBufferingUpdate(MediaPlayer mp, int percent) {
                progress_bar.setSecondaryProgress(percent);
            }
        });
        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>
```

Importer de l'audio dans androidstudio et le lire

Voici un exemple pour obtenir un fichier audio que vous avez déjà sur votre PC / ordinateur portable. Créez d'abord un nouveau répertoire sous res et nommez-le comme tel



Copiez le fichier audio que vous souhaitez lire dans ce dossier. Il peut s'agir d'un fichier .mp3 ou .wav.

Maintenant, par exemple sur le clic sur le bouton, vous voulez jouer ce son, voici comment cela se fait

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

Ceci jouera la chanson seulement une fois quand le bouton est cliqué, si vous voulez rejouer la chanson à chaque bouton cliquez écrivez le code comme ceci

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

Lire Lecteur multimédia en ligne: https://riptutorial.com/fr/android/topic/1851/lecteur-multimedia

Chapitre 161: Les notifications

Examples

Créer une notification simple

Cet exemple montre comment créer une notification simple qui démarre une application lorsque l'utilisateur clique dessus.

Spécifiez le contenu de la notification:

```
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
```

Créez l'intention de tirer sur le clic:

```
Intent intent = new Intent(this, MainActivity.class);
PendingIntent pi = PendingIntent.getActivity(this, 0, intent, Intent.FLAG_ACTIVITY_NEW_TASK);
mBuilder.setContentIntent(pi);
```

Enfin, créez la notification et affichez-la

```
NotificationManager mNotificationManager =
(NotificationManager)getSystemService(Context.NOTIFICATION_SERVICE);
mNotificationManager.notify(0, mBuilder.build());
```

Notification Heads Up avec Ticker pour les anciens appareils

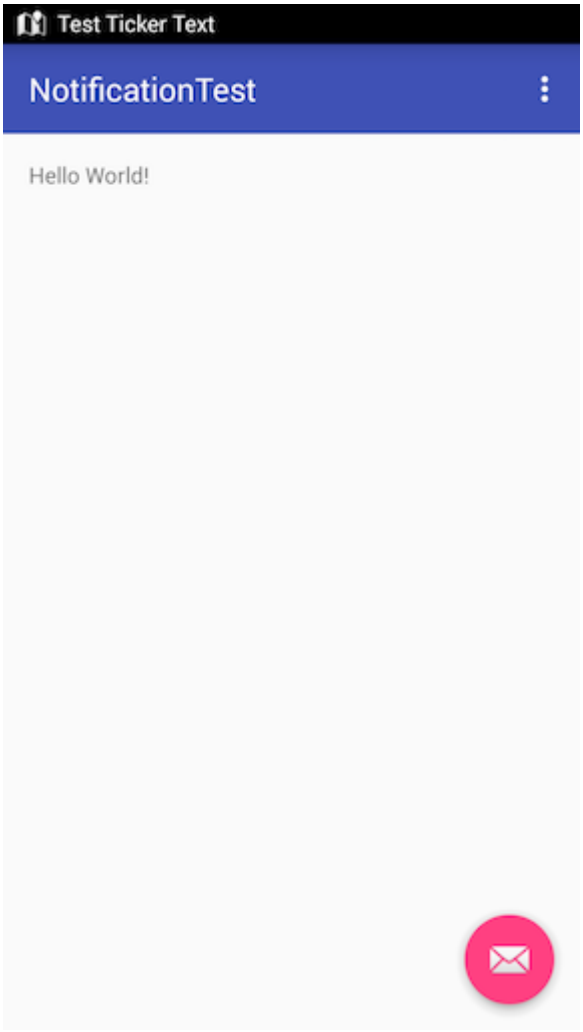
Voici comment créer une notification Heads Up pour les périphériques compatibles et utiliser un ticker pour les anciens appareils.

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

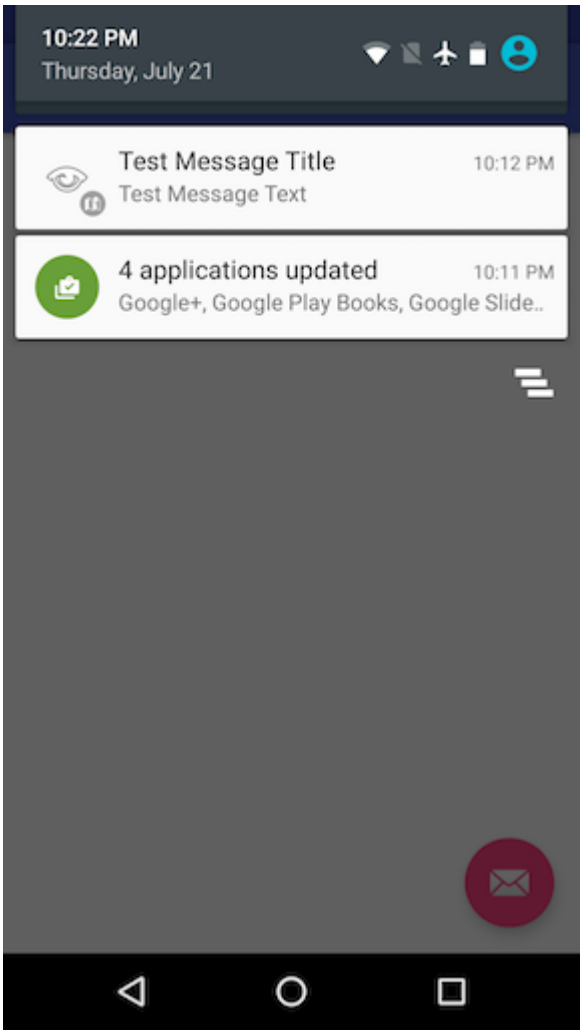
Voici à quoi il ressemble sur Android Marshmallow avec la notification Heads Up:

Voici à quoi ça ressemble sur Android KitKat avec le Ticker:

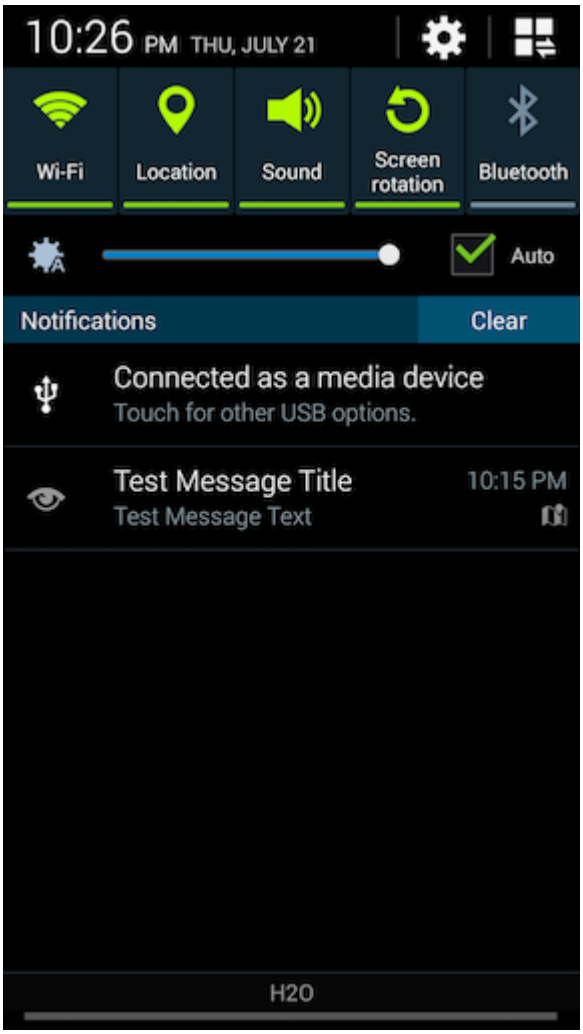


Sur toutes les versions Android, la `Notification` est affichée dans le tiroir de notification.

Android 6.0 Guimauve:



Android 4.4.x KitKat:



Définition de différentes priorités dans la notification

```
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);
```

> Lorsque la notification contient une image et que vous souhaitez étendre automatiquement l'image lorsque la notification reçue utilise "PRIORITY_MAX", vous pouvez utiliser d'autres niveaux de priorité selon les exigences

Différents niveaux de priorité Info:

PRIORITY_MAX - Utilisé pour les notifications critiques et urgentes qui alertent l'utilisateur sur une condition critique dans le temps ou devant être résolue avant de pouvoir poursuivre une tâche particulière.

PRIORITY_HIGH - Utilisé principalement pour les communications importantes, telles que les messages ou les événements de discussion avec un contenu particulièrement intéressant pour l'utilisateur. Les notifications de haute priorité déclenchent l'affichage de la notification tête haute.

PRIORITY_DEFAULT - Utilisé pour toutes les notifications qui ne correspondent à aucune des priorités décrites ici.

PRIORITY_LOW - Utilisez cette option pour les notifications dont vous souhaitez que l'utilisateur soit informé, mais qui sont moins urgentes. Les notifications de faible priorité ont tendance à apparaître au bas de la liste, ce qui en fait un bon choix pour des choses telles que les mises à jour sociales publiques ou non dirigées: l'utilisateur a demandé à être averti, mais ces notifications communication directe.

PRIORITY_MIN - Utilisé pour des informations contextuelles ou d'arrière-plan telles que des informations météorologiques ou des informations de localisation contextuelles. Les notifications de priorité minimale n'apparaissent pas dans la barre d'état. L'utilisateur les découvre lors de l'extension de la nuance de notification.

Références: Lignes directrices pour la conception des matériaux - notifications

Notifications de planification

Parfois, il est nécessaire d'afficher une notification à un moment précis, une tâche qui n'est malheureusement pas triviale sur le système Android, car il n'y a pas de méthode `setTime()` ou similaire pour les notifications. Cet exemple décrit les étapes nécessaires pour planifier des notifications à l'aide du `AlarmManager` :

1. **Ajoutez un `BroadcastReceiver`** qui écoute les `Intent` diffusées par le `AlarmManager` Android.

C'est l'endroit où vous créez votre notification en fonction des extras fournis avec l' `Intent` :

```
   public class NotificationReceiver extends BroadcastReceiver {
       @Override
       public void onReceive(Context context, Intent intent) {
           // Build notification based on Intent
           Notification notification = new NotificationCompat.Builder(context)
               .setSmallIcon(R.drawable.ic_notification_small_icon)
               .setContentTitle(intent.getStringExtra("title", ""))
               .setContentText(intent.getStringExtra("text", ""))
               .build();
           // Show notification
           NotificationManager manager = (NotificationManager)
context.getSystemService(Context.NOTIFICATION_SERVICE);
           manager.notify(42, notification);
       }
   }
```

2. **Enregistrez le `BroadcastReceiver`** dans votre fichier `AndroidManifest.xml` (sinon, le récepteur ne recevra aucune `Intent` du `AlarmManager`):

   ```
   <receiver
       android:name=".NotificationReceiver"
       android:enabled="true" />
   ```

3. **Planifier une notification** en transmettant un `PendingIntent` pour votre `BroadcastReceiver` avec les extras `Intent` nécessaires au système `AlarmManager` . Votre `BroadcastReceiver`

recevra l' `Intent` une fois l'heure donnée et affichera la notification. La méthode suivante permet de planifier une notification:

```
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);
}
```

S'il vous plaît noter que le `42` ci-dessus doit être unique pour chaque notification planifiée, sinon les `PendingIntent` s se remplaceront mutuellement provoquant des effets indésirables!

4. **Annulez une notification** en reconstruisant le `PendingIntent` associé et en l'annulant sur le `AlarmManager` du système. La méthode suivante annule une notification:

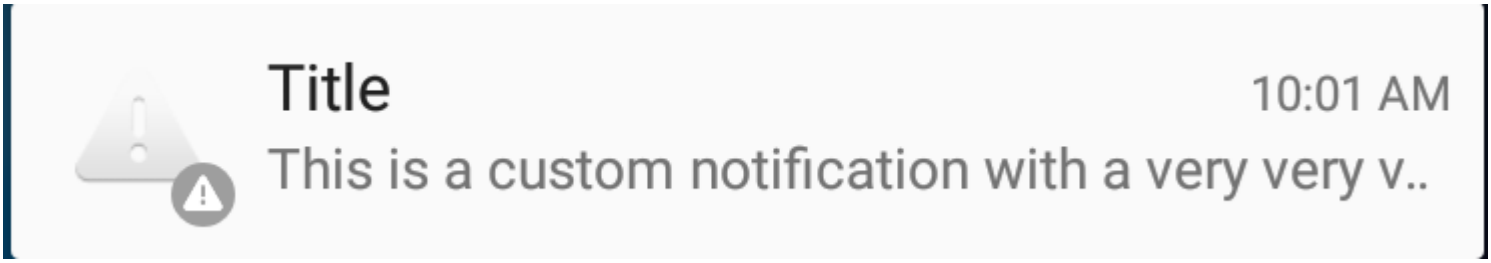
```
public static void cancelNotification(Context context, String title, String text) {
    Intent intent = new Intent(context, NotificationReceiver.class);
    intent.putExtra("title", title);
    intent.putExtra("text", text);
    PendingIntent pending = PendingIntent.getBroadcast(context, 42, intent,
PendingIntent.FLAG_UPDATE_CURRENT);
    // Cancel notification
    AlarmManager manager = (AlarmManager)
context.getSystemService(Context.ALARM_SERVICE);
    manager.cancel(pending);
}
```

Notez que le `42` ci-dessus doit correspondre au numéro de l'étape 3!

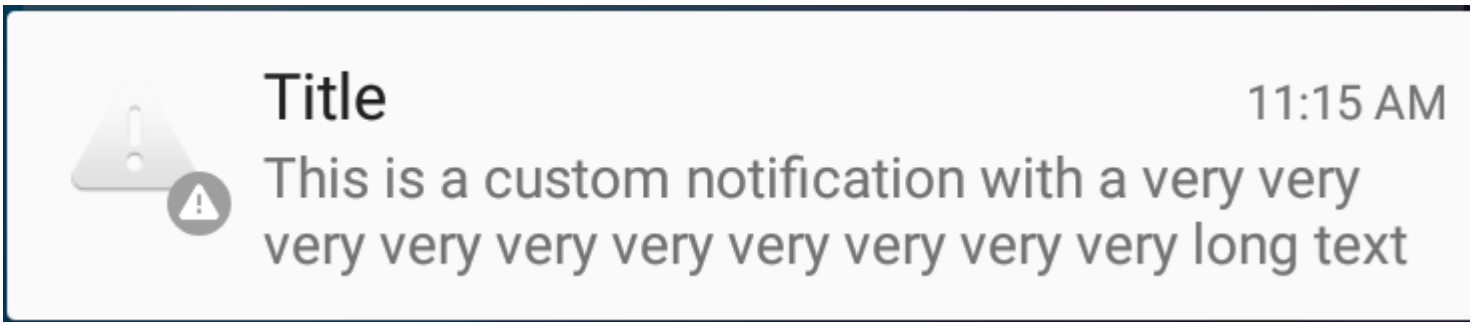
Définir une notification personnalisée - affiche le texte intégral du contenu

Si vous voulez avoir un long texte à afficher dans le contexte, vous devez définir un contenu personnalisé.

Par exemple, vous avez ceci:



Mais vous souhaitez que votre texte soit entièrement montré:



Tout ce que vous avez à faire est d' **ajouter un style** à votre contenu comme ci-dessous:

```
  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);
    }
```

Définir l'icône de notification personnalisée en utilisant la bibliothèque `Picasso`.

```
    PendingIntent pendingIntent = PendingIntent.getActivity(context,
    uniqueIntentId, intent, PendingIntent.FLAG_CANCEL_CURRENT);

    final RemoteViews remoteViews = new RemoteViews(context.getPackageName(),
R.layout.remote_view_notification);
    remoteViews.setImageViewResource(R.id.remoteview_notification_icon,
R.mipmap.ic_navigation_favorites);

    Uri defaultSoundUri = RingtoneManager.getDefaultUri(RingtoneManager.TYPE_NOTIFICATION);
    NotificationCompat.Builder notificationBuilder =
            new NotificationCompat.Builder(context)
                    .setSmallIcon(R.mipmap.ic_navigation_favorites) //just dummy icon
                    .setContent(remoteViews) // here we apply our view
                    .setAutoCancel(true)
                    .setContentIntent(pendingIntent)
                    .setPriority(NotificationCompat.PRIORITY_DEFAULT);
```

```
    final Notification notification = notificationBuilder.build();

    if (android.os.Build.VERSION.SDK_INT >= 16) {
        notification.bigContentView = remoteViews;
    }

    NotificationManager notificationManager =
            (NotificationManager) context.getSystemService(Context.NOTIFICATION_SERVICE);

    notificationManager.notify(uniqueIntentId, notification);


    //don't forget to include picasso to your build.gradle file.
    Picasso.with(context)
            .load(avatar)
            .into(remoteViews, R.id.remoteview_notification_icon, uniqueIntentId,
notification);
```

Et puis définissez une mise en page dans votre dossier de mises en page:

```
<?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>
```

Obtenir dynamiquement la taille de pixel correcte pour la grande icône

Si vous créez une image, décodez une image ou redimensionnez une image pour l'adapter à la grande zone d'image de notification, vous pouvez obtenir les dimensions de pixel correctes comme suit:

```
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);
```

Notification en cours avec le bouton Action

```
// 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);
```

Enregistrez un récepteur de diffusion pour la même action pour gérer l'événement de clic du bouton d'action.

Lire Les notifications en ligne: https://riptutorial.com/fr/android/topic/1347/les-notifications

Chapitre 162: ListView

Introduction

ListView est un groupe de vues qui regroupe plusieurs éléments d'une source de données telle qu'un tableau ou une base de données et les affiche dans une liste déroulante. Les données sont liées à listview à l'aide d'une classe d'adaptateur.

Remarques

`ListView` est un groupe de vues qui affiche une liste d'éléments défilables.
Les éléments de la liste sont automatiquement insérés dans la liste à l'aide d'un `Adapter` qui extrait le contenu d'une source telle qu'un tableau ou une requête de base de données et convertit chaque résultat d'élément en une vue placée dans la liste.

Lorsque le contenu de votre mise en page est dynamique ou non prédéterminé, vous pouvez utiliser une présentation qui sous-classe `AdapterView` pour remplir la présentation avec des vues à l'exécution. Une sous-classe de la classe `AdapterView` utilise un `Adapter` pour lier des données à sa disposition.

Avant d'utiliser `ListView` vous devriez également vérifier les exemples `RecyclerView` .

Examples

Filtrage avec 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);
     }
});
```

Supposons que votre requête s'exécute à chaque fois que l'utilisateur tape dans un `EditText` :

```
    EditText queryText = (EditText) findViewById(R.id.textQuery);
    queryText.addTextChangedListener(new TextWatcher() {
        @Override
        public void beforeTextChanged(final CharSequence s, final int start, final int count,
final int after) {

        }

        @Override
        public void onTextChanged(final CharSequence s, final int start, final int before,
final int count) {
            // This is the filter in action
            adapter.getFilter().filter(s.toString());
            // Don't forget to notify the adapter
            adapter.notifyDataSetChanged();
        }

        @Override
        public void afterTextChanged(final Editable s) {

        }
    });
```

Custom ArrayAdapter

Par défaut, la classe ArrayAdapter crée une vue pour chaque élément du tableau en appelant `toString()` sur chaque élément et en plaçant le contenu dans un TextView.

Pour créer une vue complexe pour chaque élément (par exemple, si vous voulez un ImageView pour chaque élément de tableau), étendez la classe ArrayAdapter et remplacez la méthode `getView()` pour renvoyer le type de vue souhaité pour chaque élément.

Par exemple:

```
public class MyAdapter extends ArrayAdapter<YourClassData>{

    private LayoutInflater inflater;

    public MyAdapter (Context context, List<YourClassData> data){
        super(context, 0, data);
        inflater = LayoutInflater.from(context);
    }

    @Override
    public long getItemId(int position)
    {
        //It is just an example
        YourClassData data = (YourClassData) getItem(position);
        return data.ID;
    }

    @Override
    public View getView(int position, View view, ViewGroup parent)
    {
        ViewHolder viewHolder;
```

```
        if (view == null) {
            view = inflater.inflate(R.layout.custom_row_layout_design, null);
            // Do some initialization

            //Retrieve the view on the item layout and set the value.
            viewHolder = new ViewHolder(view);
            view.setTag(viewHolder);
         }
         else {
             viewHolder = (ViewHolder) view.getTag();
         }

        //Retrieve your object
        YourClassData data = (YourClassData) getItem(position);

        viewHolder.txt.setTypeface(m_Font);
        viewHolder.txt.setText(data.text);
        viewHolder.img.setImageBitmap(BitmapFactory.decodeFile(data.imageAddr));

        return view;

    }

    private class ViewHolder
    {
         private final TextView txt;
         private final ImageView img;

         private ViewHolder(View view)
         {
             txt = (TextView) view.findViewById(R.id.txt);
             img = (ImageView) view.findViewById(R.id.img);
         }
    }
}
```

Un ListView de base avec un ArrayAdapter

Par défaut, `ArrayAdapter` crée une vue pour chaque élément du tableau en appelant `toString()` sur chaque élément et en plaçant le contenu dans un `TextView` .

Exemple:

```
ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
        android.R.layout.simple_list_item_1, myStringArray);
```

où `android.R.layout.simple_list_item_1` est la mise en page qui contient un `TextView` pour chaque chaîne du tableau.

Ensuite, appelez simplement `setAdapter()` sur votre `ListView` :

```
ListView listView = (ListView) findViewById(R.id.listview);
listView.setAdapter(adapter);
```

Pour utiliser autre chose que TextViews pour l'affichage du tableau, par exemple, ImageViews, ou

pour que certaines données en dehors des résultats de `toString()` remplissent les vues, substituez `getView(int, View, ViewGroup)` pour renvoyer le type de vue souhaité. Vérifiez cet exemple .

Lire ListView en ligne: https://riptutorial.com/fr/android/topic/4226/listview

Chapitre 163: Localisation avec des ressources sous Android

Examples

Devise

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

Ajout de la traduction sur votre application Android

Vous devez créer un `strings.xml` différent pour chaque nouvelle langue.

1. Cliquez avec le bouton droit sur le dossier *res*
2. Choisissez *Nouveau* → *Fichier de ressources Valeurs*
3. Sélectionnez un paramètre régional parmi les qualificateurs disponibles
4. Cliquez sur le bouton *Suivant* (>>)
5. Sélectionnez une langue
6. Nommez le fichier *strings.xml*

strings.xml

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

strings.xml (salut)

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

Définir la langue par programmation:

```
public void setLocale(String locale) // Pass "en","hi", etc.
{
    myLocale = new Locale(locale);
    // Saving selected locale to session - SharedPreferences.
    saveLocale(locale);
    // Changing locale.
    Locale.setDefault(myLocale);
    android.content.res.Configuration config = new android.content.res.Configuration();
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.N) {
```

```
        config.setLocale(myLocale);
    } else {
        config.locale = myLocale;
    }
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.JELLY_BEAN_MR1) {
        getBaseContext().createConfigurationContext(config);
    } else {
        getBaseContext().getResources().updateConfiguration(config,
getBaseContext().getResources().getDisplayMetrics());
    }
}
```

La fonction ci-dessus modifiera les champs de texte référencés depuis *strings.xml* . Par exemple, supposons que vous ayez les deux vues de texte suivantes:

```
<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"/>
```

Ensuite, après avoir modifié les paramètres régionaux, les chaînes de langue ayant les identifiants `app_name` et `hello` seront modifiées en conséquence.

Type de répertoires de ressources sous le dossier "res"

Lors de la localisation de différents types de ressources sont nécessaires, chacun ayant sa propre maison dans la structure du projet Android. Voici les différents répertoires que nous pouvons placer sous le répertoire `\res` . Les types de ressources placés dans chacun de ces répertoires sont expliqués dans le tableau ci-dessous:

Annuaire	Type de ressource
animateur/	Fichiers XML définissant des animations de propriétés.
anim /	Fichiers XML définissant les animations Tween. (Les animations de propriétés peuvent également être enregistrées dans ce répertoire, mais l'animateur / répertoire est préférable pour les animations de propriétés afin de distinguer les deux types.)
Couleur/	Fichiers XML définissant une liste d'états de couleurs. Voir la liste des états des couleurs
dessinable /	"Fichiers bitmap (.png, .9.png, .jpg, .gif) ou fichiers XML compilés dans les sous-types de ressources pouvant être dessinés suivants: `Bitmap files - Nine-Patches (re-sizable bitmaps) - State lists - Shapes - Animation drawables - Other drawables - "`
mipmap /	Fichiers pouvant être dessinés pour différentes densités d'icône de lanceur.

Annuaire	Type de ressource
	Pour plus d'informations sur la gestion des icônes de lanceur avec mipmap / dossiers, voir Gestion de la vue d'ensemble des projets.
disposition/	Fichiers XML définissant une interface utilisateur. Voir Ressource de mise en page.
menu/	Fichiers XML qui définissent des menus d'application, tels qu'un menu d'options, un menu contextuel ou un sous-menu. Voir la ressource de menu.
brut/	Fichiers arbitraires à enregistrer sous leur forme brute. Pour ouvrir ces ressources avec un InputStream brut, appelez Resources.openRawResource () avec l'ID de ressource, à savoir R.raw.filename.
	Cependant, si vous avez besoin d'accéder aux noms de fichiers d'origine et à la hiérarchie de fichiers, vous pouvez envisager de sauvegarder certaines ressources dans le répertoire assets / (au lieu deres / raw /). Les fichiers contenus dans assets / ne reçoivent pas d'ID de ressource, vous pouvez donc les lire uniquement à l'aide de AssetManager.
valeurs/	Fichiers XML contenant des valeurs simples, telles que des chaînes, des entiers et des couleurs, ainsi que des styles et des thèmes
xml /	Fichiers XML arbitraires pouvant être lus lors de l'exécution en appelant Resources.getXML (). Différents fichiers de configuration XML doivent être enregistrés ici, par exemple une configuration interrogeable.

Types de configuration et noms de qualificateur pour chaque dossier sous le répertoire "res"

Chaque répertoire de ressources sous le dossier `res` (répertorié dans l'exemple ci-dessus) peut avoir différentes variantes des ressources contenues dans un répertoire portant le même nom, avec des `qualifier-values` de `qualifier-values` différentes pour chaque `configuration-type` .

Exemple de variantes de répertoire avec différentes valeurs de qualificatif suffixées que l'on voit souvent dans nos projets Android:

- dessinable /
- drawable-fr /
- drawable-fr-rCA /
- drawable-en-port /
- drawable-en-notouch-12key /
- drawable-port-ldpi /
- drawable-port-notouch-12key /

Liste exhaustive de tous les différents types de

configuration et de leurs valeurs de qualificatif pour les ressources Android:

Configuration	Valeurs de qualification
MCC et MNC	Exemples:
	mcc310
	mcc310-mnc004
	mcc208-mcc00
	etc.
Langue et région	Exemples:
	en
	fr
	en-rUS
	fr-rFR
	fr-rCA
Direction de la mise en page	ldrtl
	ldltr
plus petite largeur	swdp
	Exemples:
	sw320dp
	sw600dp
	sw720dp
Largeur disponible	wdp
	w720dp
	w1024dp
Hauteur disponible	HDP
	h720dp

Configuration	Valeurs de qualification
	h1024dp
Taille de l'écran	petit
	Ordinaire
	grand
	xlarge
Aspect de l'écran	longue
	pas longtemps
Écran rond	rond
	pas du tout
Orientation de l'écran	Port
	terre
Mode interface utilisateur	voiture
	bureau
	télévision
	appliancewatch
Mode nuit	nuit
	nuit
Densité des pixels de l'écran (dpi)	ldpi
	mdpi
	hdpi
	xhdpi
	xxhdpi
	xxxhdpi
	nodpi
	tvdpi

Configuration	Valeurs de qualification
	anydpi
Type d'écran tactile	pas touche
	doigt
Disponibilité du clavier	keysexposed
	keyhidden
	clés
Méthode de saisie de texte primaire	nokeys
	qwerty
	12key
Disponibilité des clés de navigation	navexposed
	navhidden
Méthode de navigation principale non tactile	nonav
	dpad
	boule de commande
	roue
Version de plate-forme (niveau API)	Exemples:
	v3
	v4
	v7

Changer la locale de l'application Android par programmation

Dans les exemples ci-dessus, vous comprenez comment localiser les ressources de l'application. L'exemple suivant explique comment modifier les paramètres régionaux de l'application dans l'application, et non à partir du périphérique. Pour modifier uniquement les paramètres régionaux de l'application, vous pouvez utiliser ci-dessous des paramètres régionaux.

```
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();
    }
}
```

Initialiser les paramètres régionaux de préférence par l'utilisateur, à partir de la classe Application.

```
public class LocaleApp extends Application{

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

        LocaleUtils.setLocale(new Locale(LocaleUtils.getPrefLangCode(this),
LocaleUtils.getPrefCountryCode(this)));
        LocaleUtils.updateConfiguration(this, getResources().getConfiguration());
    }
}
```

Vous devez également créer une activité de base et étendre cette activité à toute autre activité afin de pouvoir modifier les paramètres régionaux de l'application d'un seul endroit, comme suit:

```
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);
    }

}
```

Remarque: Toujours initialiser les paramètres régionaux dans le constructeur.

Maintenant, vous pouvez utiliser LocalizationActivity comme suit.

```
public class MainActivity extends LocalizationActivity {

    @Override
```

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

    }
}
```

> Remarque: Lorsque vous modifiez les paramètres régionaux de l'application par programmation, vous devez redémarrer votre activité pour modifier les paramètres régionaux. Pour fonctionner correctement avec cette solution, vous pouvez utiliser les paramètres régionaux des préférences partagées au démarrage de l'application `android:name=".LocaleApp"` in vous Manifest.xml.

Parfois, invite du vérificateur de peluches à créer la version de publication. Pour résoudre ce problème, suivez les options ci-dessous.

Premier:

Si vous souhaitez désactiver la traduction pour certaines chaînes uniquement, ajoutez l'attribut suivant à la chaîne par défaut string.xml

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

Seconde:

Ignorez toutes les traductions manquantes à partir de l'attribut d'ajout de fichier de ressources L'attribut ignore de l'espace de noms des outils dans votre fichier de chaînes, comme suit:

```
<?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>
```

Troisième:

Une autre façon de désactiver la chaîne non traduisible

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

Si vous avez beaucoup de ressources qui ne doivent pas être traduites, vous pouvez les placer dans un fichier nommé donottranslate.xml et lint prendra en compte toutes les ressources non traduisibles.

Quatrième:

Vous pouvez également ajouter des paramètres régionaux dans le fichier de ressources

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

Vous pouvez également désactiver le contrôle de traduction manquant pour les peluches depuis app / build.gradle

```
lintOptions {

        disable 'MissingTranslation'
    }
```

Lire Localisation avec des ressources sous Android en ligne:
https://riptutorial.com/fr/android/topic/3345/localisation-avec-des-ressources-sous-android

Chapitre 164: Looper

Introduction

Un `Looper` est une classe Android utilisée pour exécuter une boucle de message pour un thread, dont aucun ne leur est généralement associé.

Le `Looper` le plus commun dans Android est la boucle principale, également connue sous le nom de thread principal. Cette instance est unique pour une application et peut être accédée de manière statique avec `Looper.getMainLooper()` .

Si un `Looper` est associé au thread en cours, il peut être récupéré avec `Looper.myLooper()` .

Examples

Créer un LooperThread simple

Un exemple typique de l'implémentation d'un thread `Looper` donné par la documentation officielle utilise `Looper.prepare()` et `Looper.loop()` et associe un `Handler` à la boucle entre ces appels.

```
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();
    }
}
```

Exécuter une boucle avec un HandlerThread

Un `HandlerThread` peut être utilisé pour démarrer un thread avec un `Looper` . Ce looper peut alors être utilisé pour créer un `Handler` pour les communications avec ce dernier.

```
HandlerThread thread = new HandlerThread("thread-name");
thread.start();
Handler handler = new Handler(thread.getLooper());
```

Lire Looper en ligne: https://riptutorial.com/fr/android/topic/10593/looper

Chapitre 165: LruCache

Remarques

Vous devez utiliser Lru Cache dans les applications où des charges de ressources répétitives affectent un comportement d'application fluide. Par exemple une galerie de photos avec de grandes vignettes (128x128).

Soyez toujours attentif à la taille du cache Lru car sa définition trop élevée peut affecter l'application.

Une fois que le cache Lru n'est plus utile, évitez de lui attribuer des références pour permettre au Garbage Collector de le nettoyer de la mémoire.

Pour des performances optimales, n'oubliez pas de charger des ressources telles que les images bitmap en utilisant les meilleures pratiques, comme la sélection d'une taille inSampleSize appropriée avant de l'ajouter au cache Lru.

Examples

Initialiser le cache

Le Lru Cache stockera toutes les ressources (valeurs) ajoutées pour un accès rapide jusqu'à ce qu'il atteigne une limite de mémoire, auquel cas il supprimera la ressource (valeur) la moins utilisée pour stocker la nouvelle.

Pour initialiser le cache Lru, vous devez fournir une valeur de mémoire maximale. Cette valeur dépend des exigences de votre application et de son importance pour conserver une utilisation fluide des applications. Une valeur recommandée pour une galerie d'images, par exemple, serait 1/8 de votre mémoire disponible maximale.

Notez également que le cache Lru fonctionne sur la base de la valeur clé. Dans l'exemple suivant, la clé est une `String` et la valeur est une image `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();
    }
};
```

Ajout d'un bitmap (ressource) au cache

Pour ajouter une ressource au cache, vous devez fournir une clé et la ressource. Assurez-vous d'abord que la valeur n'est pas déjà dans le cache

```
public void addResourceToMemoryCache(String key, Bitmap resource) {
    if (memoryCache.get(key) == null)
        memoryCache.put(key, resource);
}
```

Obtenir un bitmap (Resouce) du cache

Pour obtenir une ressource du cache, transmettez simplement la clé de votre ressource (String dans cet exemple)

```
public Bitmap getResourceFromMemoryCache(String key) {
    memoryCache.get(key);
}
```

Lire LruCache en ligne: https://riptutorial.com/fr/android/topic/7709/lrucache

Chapitre 166: Manipulation de liens profonds

Introduction

Les liens profonds sont des URL qui permettent aux utilisateurs d'accéder directement à un contenu spécifique de votre application. Vous pouvez configurer des liens profonds en ajoutant des filtres d'intention et en extrayant des données des intentions entrantes afin de diriger les utilisateurs vers le bon écran de votre application.

Paramètres

`<data>` Attribut	Détails
schème	La partie *schéma* d'un URI (sensible à la casse). Exemples: `http` , `https` , `ftp`
hôte	La partie *hôte* d'un URI (sensible à la casse). Exemples: `google.com` , `example.org`
Port	La partie *port* d'un URI. Exemples: `80` , `443`
chemin	La partie *chemin* d'un URI. Doit commencer par `/` . Exemples: `/` , `/about`
pathPrefix	Un préfixe pour la partie *chemin* d'un URI. Exemples: `/item` , `/article`
pathPattern	Un modèle à faire correspondre à la partie *chemin* d'un URI. Exemples: `/item/.*` , `/article/[0-9]*`
mimeType	Un type de mime correspondant. Exemples: `image/jpeg` , `audio/*`

Remarques

Le `<intent-filter>`

Cette combinaison d'éléments `<action>` et `<category>` est ce qui indique au système Android qu'une activité spécifique doit être lancée lorsque l'utilisateur clique sur un lien dans une autre application.

```
<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>
```

Plusieurs balises `<data>`

L'ensemble des liens profonds pris en charge par votre `<intent-filter>` est le produit croisé de tous les éléments `<data>` que vous définissez dans ce filtre d'intention. Les exemples de domaines multiples, de chemins multiples et de schémas multiples en sont la preuve.

Ressources

- Activation des liens profonds pour le contenu de l'application (developer.android.com)
- `<intent-filter>` (developer.android.com

Examples

Lien profond simple

AndroidManifest.xml:

```
<activity android:name="com.example.MainActivity" >

    <intent-filter>
        <action android:name="android.intent.action.VIEW" />
        <category android:name="android.intent.category.DEFAULT" />
        <category android:name="android.intent.category.BROWSABLE" />

        <data android:scheme="http"
              android:host="www.example.com" />

    </intent-filter>

</activity>
```

Cela acceptera tout lien commençant par `http://www.example.com` comme un lien profond pour démarrer votre `MainActivity` .

Plusieurs chemins sur un seul domaine

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>
```

Cela lancera votre `MainActivity` lorsque l'utilisateur `MainActivity` l'un de ces liens:

- `http://www.example.com/`
- `http://www.example.com/about`
- `http://www.example.com/map`

Plusieurs domaines et plusieurs chemins

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>
```

Cela lancera votre MainActivity lorsque l'utilisateur cliquera sur l'un de ces liens:

- `http://www.example.com/`
- `http://www.example2.com/`
- `http://www.example.com/map`
- `http://www.example2.com/map`

Les deux http et https pour le même domaine

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>
```

Cela lancera votre MainActivity lorsque l'utilisateur cliquera sur l'un de ces liens:

- `http://www.example.com/`
- `https://www.example.com/`
- `http://www.example.com/map`
- `https://www.example.com/map`

Récupération des paramètres de requête

```
public class MainActivity extends Activity {

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

        Intent intent = getIntent();
        Uri data = intent.getData();

        if (data != null) {
            String param1 = data.getQueryParameter("param1");
            String param2 = data.getQueryParameter("param2");
        }
    }

}
```

Si l'utilisateur clique sur un lien vers `http://www.example.com/map?param1=FOO&param2=BAR` , alors `param1` ici aura la valeur `"FOO"` et `param2` aura la valeur `"BAR"` .

Utiliser pathPrefix

AndroidManifest.xml:

```
<activity android:name="com.example.MainActivity" >

    <intent-filter>
        <action android:name="android.intent.action.VIEW" />
        <category android:name="android.intent.category.DEFAULT" />
        <category android:name="android.intent.category.BROWSABLE" />

        <data android:scheme="http"
              android:host="www.example.com"
              android:path="/item" />

    </intent-filter>
```

```
</activity>
```

Cela lancera votre mainActivity lorsque l'utilisateur cliquera sur un lien commençant par `http://www.example.com/item` , tel que:

- `https://www.example.com/item`
- `http://www.example.com/item/1234`
- `https://www.example.com/item/xyz/details`

Lire Manipulation de liens profonds en ligne:
https://riptutorial.com/fr/android/topic/3716/manipulation-de-liens-profonds

Chapitre 167: Manutentionnaire

Remarques

Un gestionnaire peut facilement être utilisé pour exécuter du code après un laps de temps retardé. Il est également utile pour exécuter du code à plusieurs reprises après un délai spécifié en appelant à nouveau la méthode Handler.postDelayed () à partir de la méthode run () de Runnable.

Examples

Utiliser un gestionnaire pour exécuter du code après un laps de temps retardé

Exécution du code après 1,5 seconde:

```
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
```

Exécuter le code à plusieurs reprises toutes les 1 secondes:

```
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 et communication entre les threads

Comme `Handler` s sont utilisés pour envoyer un `Message` s et `Runnable` au message d'un fil de file d' attente , il est facile à mettre en œuvre une communication à base d'événements entre plusieurs threads. Chaque fil doté d'un `Looper` peut recevoir et traiter des messages. Un `HandlerThread` est un thread qui implémente un tel `Looper` , par exemple le thread principal (UI Thread) implémente les fonctionnalités d'un `HandlerThread` .

Créer un gestionnaire pour le thread en cours

```
Handler handler = new Handler();
```

Création d'un gestionnaire pour le thread principal (thread

d'interface utilisateur)

```
Handler handler = new Handler(Looper.getMainLooper());
```

Envoyer un runnable d'un autre thread au thread principal

```
new Thread(new Runnable() {
    public void run() {
        // this is executed on another Thread

        // create a Handler associated with the main Thread
        Handler handler = new Handler(Looper.getMainLooper());

        // post a Runnable to the main Thread
        handler.post(new Runnable() {
            public void run() {
                // this is executed on the main Thread
            }
        });
    }
}).start();
```

Créer un gestionnaire pour un autre HandlerThread et lui envoyer des événements

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

Arrêter le gestionnaire d'exécution

Pour arrêter l'exécution du gestionnaire, supprimez le rappel qui lui est associé à l'aide du fichier exécutable exécuté à l'intérieur:

```
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);
}
```

Utilisez Handler pour créer un minuteur (similaire à javax.swing.Timer)

Cela peut être utile si vous écrivez un jeu ou quelque chose qui doit exécuter un morceau de code toutes les quelques secondes.

```
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 ();
    }
}
```

Exemple d'utilisation:

```
Timer timer = new Timer(new Runnable() {
    public void run() {
        System.out.println("Hello");
    }
}, 1000, true)
```

Ce code affichera "Hello" toutes les secondes.

Lire Manutentionnaire en ligne: https://riptutorial.com/fr/android/topic/1425/manutentionnaire

Chapitre 168: MediaSession

Syntaxe

- void mediaSessionCompat.setFlags (int flags)
- void mediaSessionCompat.setMediaButtonReceiver (PendingIntent mbr)
- void mediaSessionCompat.setCallback (rappel MediaSessionCompat.Callback)
- void mediaSessionCompat.setActive (booléen actif)
- MediaSessionCompat.Token mediaSessionCompat.getSessionToken ()
- void mediaSessionCompat.release ()
- void mediaSessionCompat.setPlaybackState (état PlaybackStateCompat)
- void mediaSessionCompat.setMetadata (métadonnées MediaMetadataCompat)

Remarques

Pour de meilleures pratiques, utilisez la bibliothèque **media-compat** . La bibliothèque prend en charge la rétrocompatibilité en traduisant les méthodes de session multimédia en méthodes équivalentes sur les anciennes versions de plate-forme lorsqu'elles sont disponibles.

Examples

Réception et traitement des événements de bouton

Cet exemple crée un objet `MediaSession` lors du démarrage d'un `Service` . L'objet `MediaSession` est libéré lorsque le `Service` est détruit:

```
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();
    }
}
```

La méthode suivante instancie et configure les `MediaSession` bouton `MediaSession` :

```
private void configureMediaSession {
    s_mediaSession = new MediaSession(this, "MyMediaSession");

    // Overridden methods in the MediaSession.Callback class.
```

```
    s_mediaSession.setCallback(new MediaSession.Callback() {
        @Override
        public boolean onMediaButtonEvent(Intent mediaButtonIntent) {
            Log.d(TAG, "onMediaButtonEvent called: " + mediaButtonIntent);
            KeyEvent ke = mediaButtonIntent.getParcelableExtra(Intent.EXTRA_KEY_EVENT);
            if (ke != null && ke.getAction() == KeyEvent.ACTION_DOWN) {
                int keyCode = ke.getKeyCode();
                Log.d(TAG, "onMediaButtonEvent Received command: " + ke);
            }
            return super.onMediaButtonEvent(mediaButtonIntent);
        }

        @Override
        public void onSkipToNext() {
            Log.d(TAG, "onSkipToNext called (media button pressed)");
            Toast.makeText(getApplicationContext(), "onSkipToNext called",
Toast.LENGTH_SHORT).show();
            skipToNextPlaylistItem(); // Handle this button press.
            super.onSkipToNext();
        }

        @Override
        public void onSkipToPrevious() {
            Log.d(TAG, "onSkipToPrevious called (media button pressed)");
            Toast.makeText(getApplicationContext(), "onSkipToPrevious called",
Toast.LENGTH_SHORT).show();
            skipToPreviousPlaylistItem(); // Handle this button press.
            super.onSkipToPrevious();
        }

        @Override
        public void onPause() {
            Log.d(TAG, "onPause called (media button pressed)");
            Toast.makeText(getApplicationContext(), "onPause called",
Toast.LENGTH_SHORT).show();
            mpPause(); // Pause the player.
            super.onPause();
        }

        @Override
        public void onPlay() {
            Log.d(TAG, "onPlay called (media button pressed)");
            mpStart(); // Start player/playback.
            super.onPlay();
        }

        @Override
        public void onStop() {
            Log.d(TAG, "onStop called (media button pressed)");
            mpReset(); // Stop and/or reset the player.
            super.onStop();
        }
    });

    s_mediaSession.setFlags(MediaSession.FLAG_HANDLES_MEDIA_BUTTONS |
MediaSession.FLAG_HANDLES_TRANSPORT_CONTROLS);
    s_mediaSession.setActive(true);
}
```

La méthode suivante envoie des métadonnées (stockées dans un `HashMap`) au périphérique à

l'aide de A2DP:

```
void sendMetaData(@NonNull final HashMap<String, String> hm) {
    // Return if Bluetooth A2DP is not in use.
    if (!((AudioManager) getSystemService(Context.AUDIO_SERVICE)).isBluetoothA2dpOn()) return;

    MediaMetadata metadata = new MediaMetadata.Builder()
            .putString(MediaMetadata.METADATA_KEY_TITLE, hm.get("Title"))
            .putString(MediaMetadata.METADATA_KEY_ALBUM, hm.get("Album"))
            .putString(MediaMetadata.METADATA_KEY_ARTIST, hm.get("Artist"))
            .putString(MediaMetadata.METADATA_KEY_AUTHOR, hm.get("Author"))
            .putString(MediaMetadata.METADATA_KEY_COMPOSER, hm.get("Composer"))
            .putString(MediaMetadata.METADATA_KEY_WRITER, hm.get("Writer"))
            .putString(MediaMetadata.METADATA_KEY_DATE, hm.get("Date"))
            .putString(MediaMetadata.METADATA_KEY_GENRE, hm.get("Genre"))
            .putLong(MediaMetadata.METADATA_KEY_YEAR, tryParse(hm.get("Year")))
            .putLong(MediaMetadata.METADATA_KEY_DURATION, tryParse(hm.get("Raw Duration")))
            .putLong(MediaMetadata.METADATA_KEY_TRACK_NUMBER, tryParse(hm.get("Track
Number")))
            .build();

    s_mediaSession.setMetadata(metadata);
}
```

La méthode suivante définit l'état de `PlaybackState` . Il définit également les actions de bouton auxquelles `MediaSession` répondra:

```
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);
}
```

Lire MediaSession en ligne: https://riptutorial.com/fr/android/topic/6250/mediasession

Chapitre 169: MediaStore

Examples

Récupérer des fichiers audio / MP3 à partir d'un dossier spécifique de l'appareil ou récupérer tous les fichiers

Tout d'abord, ajoutez les autorisations suivantes au manifeste de votre projet afin d'activer l'accès au stockage du périphérique:

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

Ensuite, créez le fichier *AudioModel.class* et placez-y la classe de modèle suivante pour permettre l'obtention et la définition des éléments de liste:

```
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;
    }
}
```

Ensuite, utilisez la méthode suivante pour lire tous les fichiers MP3 d'un dossier de votre appareil ou pour lire tous les fichiers de votre appareil:

```
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;
}
```

Le code ci-dessus renverra une liste de tous les fichiers MP3 avec le nom de la musique, son chemin, son artiste et son album. Pour plus de détails, reportez-vous à la documentation Media.Store.Audio .

Pour lire les fichiers d'un dossier spécifique, utilisez la requête suivante (vous devez remplacer le nom du dossier):

```
Cursor c = context.getContentResolver().query(uri,
    projection,
    MediaStore.Audio.Media.DATA + " like ? ",
    new String[]{"%yourFolderName%"}, // Put your device folder / file location here.
    null);
```

Si vous souhaitez récupérer tous les fichiers de votre appareil, utilisez la requête suivante:

```
Cursor c = context.getContentResolver().query(uri,
    projection,
    null,
    null,
    null);
```

Remarque: N'oubliez pas d'activer les autorisations d'accès au stockage.

Il ne vous reste plus qu'à appeler la méthode ci-dessus pour obtenir les fichiers MP3:

```
getAllAudioFromDevice(this);
```

Exemple avec activité

```
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;
    }
```

```
}
```

Lire MediaStore en ligne: https://riptutorial.com/fr/android/topic/7136/mediastore

Chapitre 170: Menu

Syntaxe

- inflater.inflate (R.menu.your_xml_file, menu);

Paramètres

Paramètre	La description
`inflate(int menuRes, Menu menu)`	Gonflez une hiérarchie de menus à partir de la ressource XML spécifiée.
`getMenuInflater ()`	Retourne un `MenuInflater` avec ce contexte.
`onCreateOptionsMenu (Menu menu)`	Initialiser le contenu du menu d'options standard de l'activité. Vous devez placer vos éléments de menu dans le menu.
`onOptionsItemSelected (MenuItem item)`	Cette méthode est appelée chaque fois qu'un élément de votre menu d'options est sélectionné

Remarques

Pour en savoir plus sur les **menus** , lisez ceci . J'espère que cela aide!

Examples

Menu d'options avec séparateurs

Dans Android, il existe un menu d'options par défaut, qui peut prendre un certain nombre d'options. Si un plus grand nombre d'options doit être affiché, il est judicieux de regrouper ces options afin de conserver la clarté. Les options peuvent être regroupées en mettant des séparateurs (lignes horizontales) entre eux. Pour permettre des séparateurs, le thème suivant peut être utilisé:

```
<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>
```

En modifiant le thème, des séparateurs peuvent être ajoutés à un menu.

Appliquer une police personnalisée à Menu

```
public static void applyFontToMenu(Menu m, Context mContext){
    for(int i=0;i<m.size();i++) {
        applyFontToMenuItem(m.getItem(i),mContext);
    }
}
public static void applyFontToMenuItem(MenuItem mi, Context mContext) {
    if(mi.hasSubMenu())
        for(int i=0;i<mi.getSubMenu().size();i++) {
            applyFontToMenuItem(mi.getSubMenu().getItem(i),mContext);
        }
    Typeface font = Typeface.createFromAsset(mContext.getAssets(),
"fonts/yourCustomFont.ttf");
    SpannableString mNewTitle = new SpannableString(mi.getTitle());
    mNewTitle.setSpan(new CustomTypefaceSpan("", font, mContext), 0, mNewTitle.length(),
Spannable.SPAN_INCLUSIVE_INCLUSIVE);
    mi.setTitle(mNewTitle);
}
```

et ensuite dans l'activité:

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.main, menu);
    applyFontToMenu(menu,this);
    return true;
}
```

Créer un menu dans une activité

Pour définir votre propre menu, créez un fichier XML dans le répertoire `res/menu/` votre projet et créez le menu avec les éléments suivants:

- `<menu>` : définit un menu contenant tous les éléments du menu.
- `<item>` : crée un MenuItem, qui représente un seul élément dans un menu. Nous pouvons également créer un élément imbriqué afin de créer un sous-menu.

Étape 1:

Créez votre propre fichier xml comme suit:

Dans `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>
```

Étape 2:

Pour spécifier le menu d'options, remplacez `onCreateOptionsMenu()` dans votre *activité* .

Dans cette méthode, vous pouvez gonfler votre ressource de menu (définie dans votre fichier XML, à savoir `res/menu/main_menu.xml`)

```
 @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.main_menu, menu);
        return true;
    }
```

Lorsque l'utilisateur sélectionne un élément dans le menu d'options, le système appelle la méthode `onOptionsItemSelected()` votre *activité* .

- Cette méthode passe le menuItem sélectionné.
- Vous pouvez identifier l'élément en appelant `getItemId()` , qui renvoie l'identifiant unique de l'élément de menu (défini par l' `android:id attribute` dans la ressource de menu - `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);
    }
}
```

Emballer!

Votre code d' `Activity` devrait ressembler à ceci:

```
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);
        }
    }
}
```

Capture d'écran de l'apparence de votre propre menu:

Lire Menu en ligne: https://riptutorial.com/fr/android/topic/2028/menu

Chapitre 171: Métriques d'affichage du périphérique

Examples

Obtenir les dimensions des pixels des écrans

Pour retrouver la largeur et la hauteur des écrans en pixels, nous pouvons utiliser les métriques d'affichage de WindowManagers .

```
// Get display metrics
DisplayMetrics metrics = new DisplayMetrics();
context.getWindowManager().getDefaultDisplay().getMetrics(metrics);
```

Ces DisplayMetrics contiennent une série d'informations sur l'écran des périphériques, comme sa densité ou sa taille:

```
// Get width and height in pixel
Integer heightPixels = metrics.heightPixels;
Integer widthPixels = metrics.widthPixels;
```

Obtenir la densité de l'écran

Pour obtenir la densité des écrans, nous pouvons également utiliser le Windowmanagers DisplayMetrics . Ceci est un exemple rapide:

```
// Get density in dpi
DisplayMetrics metrics = new DisplayMetrics();
context.getWindowManager().getDefaultDisplay().getMetrics(metrics);
int densityInDpi =  metrics.densityDpi;
```

Formule px à dp, dp à px conversation

DP to Pixel:

```
private int dpToPx(int dp)
{
    return (int) (dp * Resources.getSystem().getDisplayMetrics().density);
}
```

Pixel to DP:

```
private int pxToDp(int px)
{
    return (int) (px / Resources.getSystem().getDisplayMetrics().density);
}
```

Lire Métriques d'affichage du périphérique en ligne:
https://riptutorial.com/fr/android/topic/4207/metriques-d-affichage-du-peripherique

Chapitre 172: Mises en page

Introduction

Une mise en page définit la structure visuelle d'une interface utilisateur, telle qu'une activité ou un widget.

Une mise en page est déclarée en XML, y compris les éléments d'écran qui y apparaîtront. Vous pouvez ajouter du code à l'application pour modifier l'état des objets à l'écran lors de l'exécution, y compris ceux déclarés en XML.

Syntaxe

- android: gravity = "top | bottom | left | right | centre_vertical | fill_vertical | center_horizontal | fill_horizontal | center | fill | clip_vertical | clip_horizontal | début | fin"
- android: layout_gravity = "haut | bas | gauche | droite | center_vertical | fill_vertical | center_horizontal | fill_horizontal | center | fill | clip_vertical | clip_horizontal | début | fin"

Remarques

LayoutParams et Layout_ Attributes

Layout Attributes

+CoordinatorLayout

layout_behavior

+FrameLayout

layout_gravity

+LinearLayout

layout_weight

+RelativeLayout

layout_above layout_below

layout_alignLeft/Top/Right/Bottom

layout_alignParentLeft/etc

layout_toLeftOf/etc

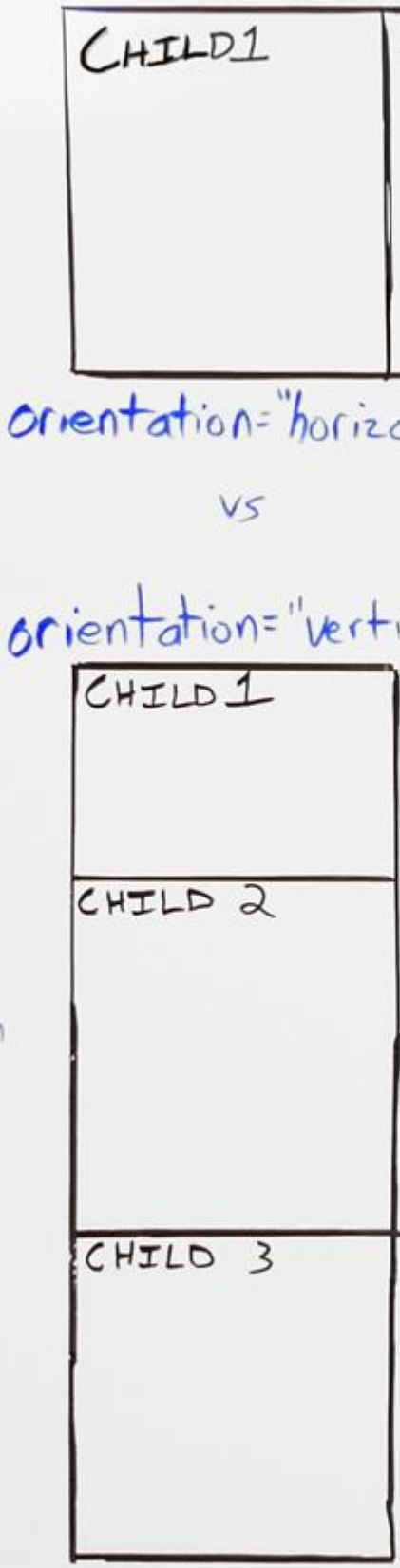
layout_alignBaseline

layout_centerInParent

+AbsoluteLayout NO

Please don't

Linear



nécessite le rendu correct de deux passes de présentation. Pour les hiérarchies de vues complexes, cela peut avoir un impact significatif sur les performances. L'imbrication de `RelativeLayouts` aggrave ce problème, car chaque `RelativeLayout` entraîne une augmentation du nombre de passages de la présentation.

Examples

LinearLayout

LinearLayout est un `ViewGroup` qui organise ses enfants dans une seule colonne ou une seule ligne. L'orientation peut être définie en appelant la méthode `setOrientation()` ou en utilisant l'attribut xml `android:orientation` .

1. **Orientation verticale** : `android:orientation="vertical"`

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

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/app_name" />

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@android:string/cancel" />

</LinearLayout>
```

Voici une capture d'écran à quoi cela ressemblera:

2. **Orientation horizontale** : `android:orientation="horizontal"`

```
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/app_name" />

<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@android:string/cancel" />
```

`LinearLayout` prend également en charge l'attribution d'un poids à des enfants individuels avec l'attribut `android:layout_weight` .

Disposition relative

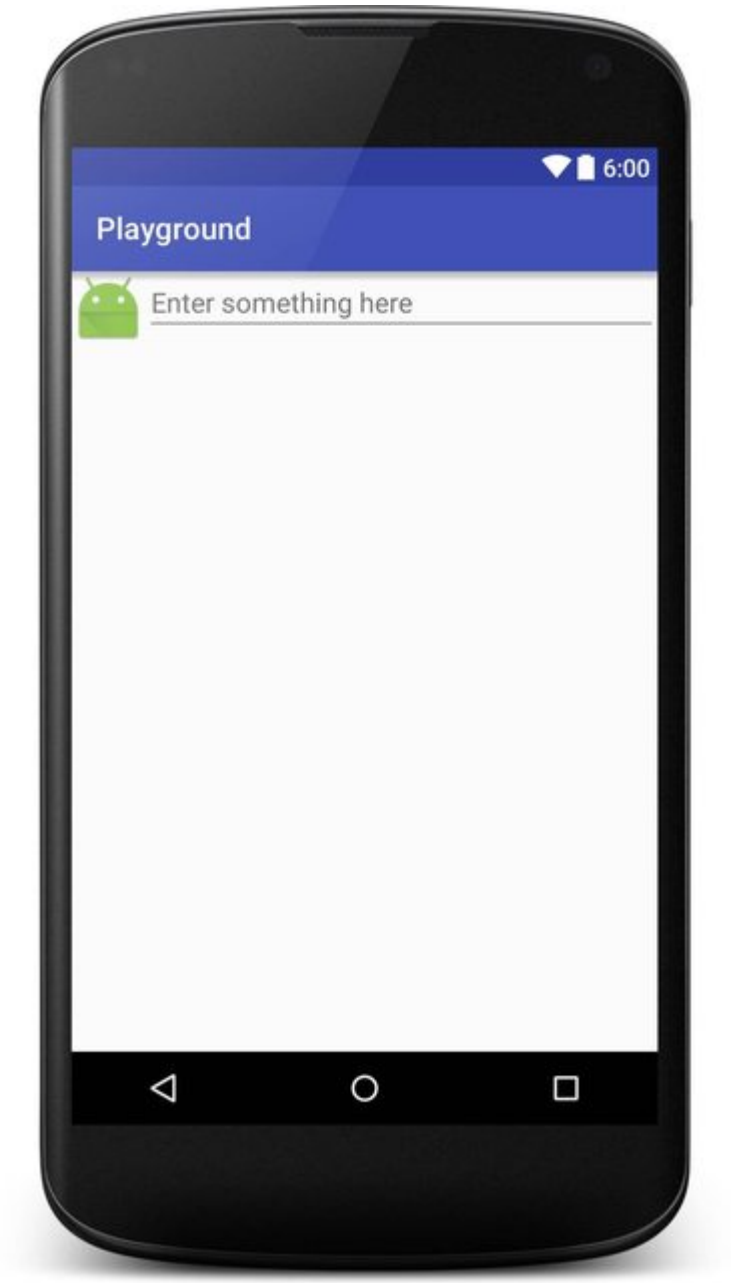
`RelativeLayout` est un `ViewGroup` qui affiche les vues enfants dans des positions relatives. Par

défaut, toutes les vues enfants sont dessinées en haut à gauche de la présentation. Vous devez donc définir la position de chaque vue à l'aide des différentes propriétés de mise en page disponibles dans `RelativeLayout.LayoutParams` . La valeur de chaque propriété de mise en page est soit une valeur booléenne permettant d'activer une position de mise en page par rapport au parent RelativeLayout, soit un ID faisant référence à une autre vue de la mise en page sur laquelle la vue doit être positionnée.

Exemple:

```
<?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>
```

Voici une capture d'écran à quoi cela ressemblera:



Gravité et disposition gravitationnelle

android: layout_gravity

- `android:layout_gravity` est utilisé pour définir la position d'un élément dans son parent (par exemple, une `View` enfant dans une `Layout`).
- Pris en charge par LinearLayout et FrameLayout

android: gravité

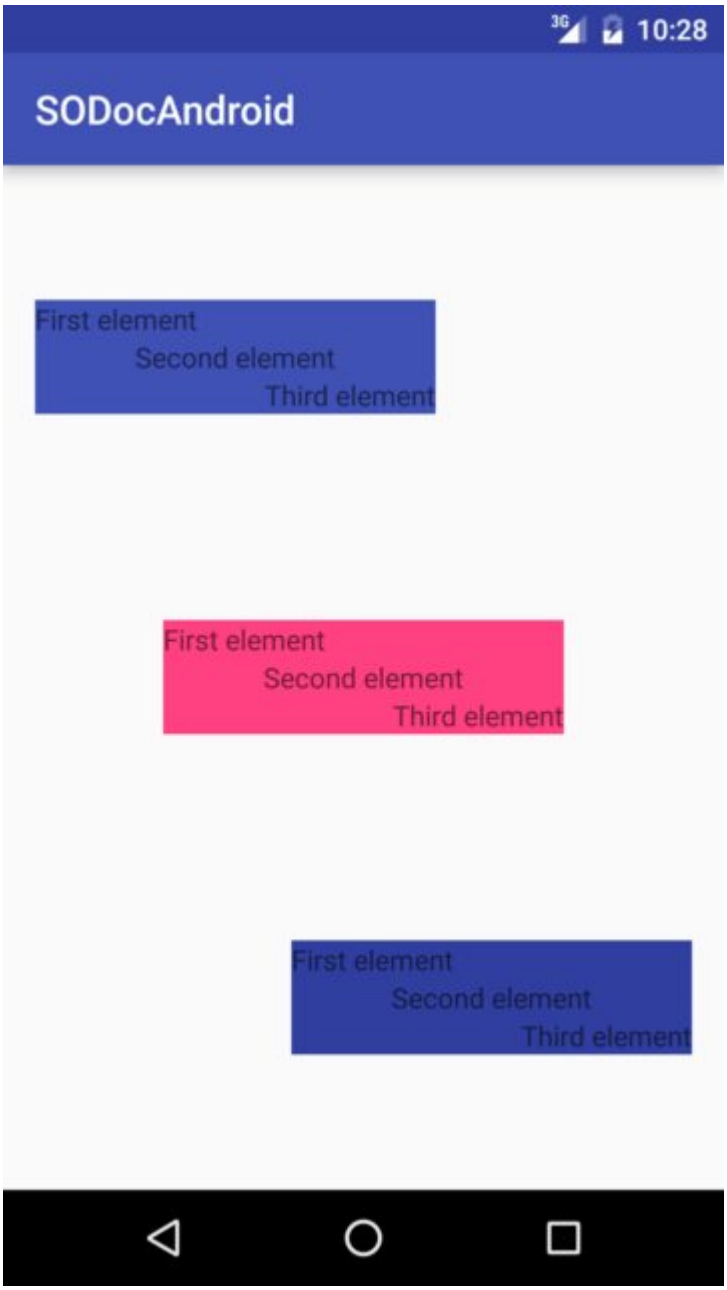
- `android:gravity` est utilisée pour définir la position du contenu à l'intérieur d'un élément (par exemple, un texte dans un `TextView`).

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

```
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:orientation="vertical">

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:orientation="vertical"
        android:layout_gravity="left"
        android:gravity="center_vertical">

        <TextView
            android:layout_width="@dimen/fixed"
            android:layout_height="wrap_content"
            android:text="@string/first"
            android:background="@color/colorPrimary"
            android:gravity="left"/>

        <TextView
            android:layout_width="@dimen/fixed"
            android:layout_height="wrap_content"
            android:text="@string/second"
            android:background="@color/colorPrimary"
            android:gravity="center"/>

        <TextView
            android:layout_width="@dimen/fixed"
            android:layout_height="wrap_content"
            android:text="@string/third"
            android:background="@color/colorPrimary"
            android:gravity="right"/>

    </LinearLayout>

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:orientation="vertical"
        android:layout_gravity="center"
        android:gravity="center_vertical">

        <TextView
            android:layout_width="@dimen/fixed"
            android:layout_height="wrap_content"
            android:text="@string/first"
            android:background="@color/colorAccent"
            android:gravity="left"/>

        <TextView
            android:layout_width="@dimen/fixed"
            android:layout_height="wrap_content"
            android:text="@string/second"
            android:background="@color/colorAccent"
            android:gravity="center"/>

        <TextView
```

```
            android:layout_width="@dimen/fixed"
            android:layout_height="wrap_content"
            android:text="@string/third"
            android:background="@color/colorAccent"
            android:gravity="right"/>

    </LinearLayout>

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:orientation="vertical"
        android:layout_gravity="right"
        android:gravity="center_vertical">

        <TextView
            android:layout_width="@dimen/fixed"
            android:layout_height="wrap_content"
            android:text="@string/first"
            android:background="@color/colorPrimaryDark"
            android:gravity="left"/>

        <TextView
            android:layout_width="@dimen/fixed"
            android:layout_height="wrap_content"
            android:text="@string/second"
            android:background="@color/colorPrimaryDark"
            android:gravity="center"/>

        <TextView
            android:layout_width="@dimen/fixed"
            android:layout_height="wrap_content"
            android:text="@string/third"
            android:background="@color/colorPrimaryDark"
            android:gravity="right"/>

    </LinearLayout>

</LinearLayout>
```

Qui est rendu comme suit:

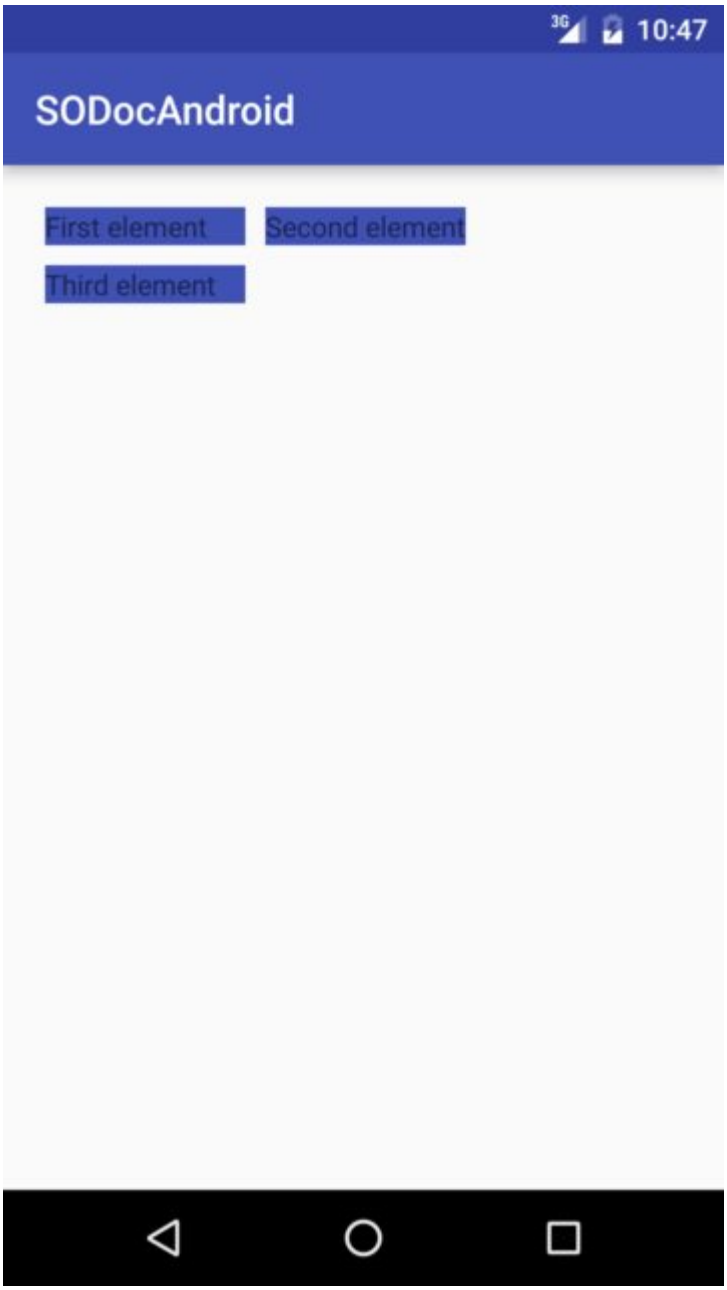


Disposition de la grille

GridLayout, comme son nom l'indique est une mise en page utilisée pour organiser les vues dans une grille. Un GridLayout se divise en colonnes et en lignes. Comme vous pouvez le voir dans l'exemple ci-dessous, la quantité de colonnes et / ou de lignes est spécifiée par les propriétés `columnCount` et `rowCount` . L'ajout de vues à cette disposition ajoute la première vue à la première colonne, la deuxième vue à la deuxième colonne et la troisième à la première colonne de la deuxième ligne.

```
<?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>
```



Mise en page en pourcentage

2.3

La bibliothèque de pourcentage de prise en charge fournit `PercentFrameLayout` et `PercentRelativeLayout` , deux groupes ViewGroup qui permettent de spécifier facilement les **dimensions et les marges de** View en termes de **pourcentage** de la taille globale.

Vous pouvez utiliser la bibliothèque de pourcentage de support en ajoutant les éléments suivants à vos dépendances.

```
compile 'com.android.support:percent:25.3.1'
```

Si vous souhaitez afficher une vue qui remplit l'écran horizontalement mais que la moitié de l'écran verticalement, vous feriez ce qui suit.

```
<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>
```

Vous pouvez également définir les pourcentages dans un fichier XML distinct avec un code tel que:

```
<fraction name="margin_start_percent">25%</fraction>
```

Et se référer à eux dans vos mises en page avec `@fraction/margin_start_percent` .

Ils contiennent également la possibilité de définir un **format d'image** personnalisé via l' `app:layout_aspectRatio` .
Cela vous permet de définir une seule dimension, par exemple uniquement la largeur, et la hauteur sera automatiquement déterminée en fonction des proportions que vous avez définies, que ce soit 4: 3 ou 16: 9 ou même un carré 1: 1. ratio d'aspect.

Par exemple:

```
 <ImageView
    app:layout_widthPercent="100%"
    app:layout_aspectRatio="178%"
    android:scaleType="centerCrop"
    android:src="@drawable/header_background"/>
```

FrameLayout

`FrameLayout` est conçu pour bloquer une zone sur l'écran pour afficher un seul élément. Vous pouvez cependant ajouter plusieurs enfants à un FrameLayout et contrôler leur position dans FrameLayout en attribuant une gravité à chaque enfant, en utilisant l'attribut android: layout_gravity .

En règle générale, `FrameLayout` est utilisé pour contenir une vue enfant unique. Les cas d'utilisation courants sont la création d'espaces `Fragments` au gonflage de `Fragments` dans l' `Activity` , de superpositions de vues ou de l'application d'un premier plan sur les vues.

Exemple:

```
<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>
```

Il ressemblera à ceci:

Coordinateur

2.3

Le `CoordinatorLayout` est un conteneur quelque peu similaire à `FrameLayout` mais avec des

fonctionnalités supplémentaires, il est appelé `FrameLayout` super-puissant dans la documentation officielle.

En attachant un `CoordinatorLayout.Behavior` à un enfant direct de CoordinatorLayout, vous pouvez intercepter les événements tactiles, les encarts de fenêtre, les mesures, la mise en page et le défilement imbriqué.

Pour l'utiliser, vous devrez d'abord ajouter une dépendance pour la bibliothèque de support dans votre fichier de gradle:

```
compile 'com.android.support:design:25.3.1'
```

Le numéro de la dernière version de la bibliothèque peut être trouvé ici

Un cas d'utilisation pratique de `CoordinatorLayout` est la création d'une vue avec un `FloatingActionButton` . Dans ce cas spécifique, nous allons créer un `RecyclerView` avec un `SwipeRefreshLayout` et un `FloatingActionButton` en plus. Voici comment procéder:

```
<?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>
```

Remarquez comment le FloatingActionButton est ancré dans le CoordinatorLayout avec `app:layout_anchor="@id/coord_layout"`

Comportement du défilement du coordinateur

2.3-2.3.2

Vous pouvez utiliser un `CoordinatorLayout` englobant pour obtenir des effets de défilement de conception de matériau lors de l'utilisation de mises en page internes prenant en charge le défilement imbriqué, telles que `NestedScrollView` ou `RecyclerView` .

Pour cet exemple:

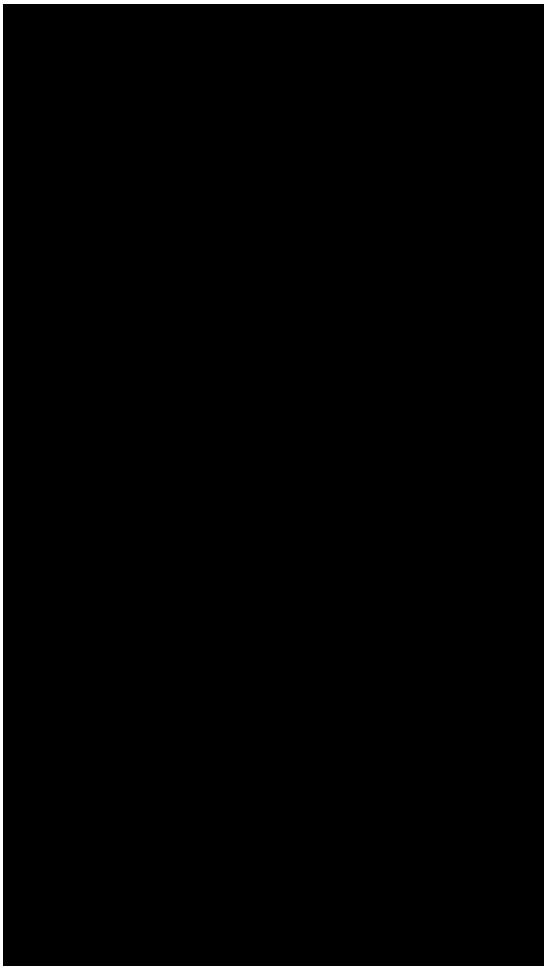
- `app:layout_scrollFlags="scroll|enterAlways"` est utilisé dans les propriétés de la barre d'outils
- `app:layout_behavior="@string/appbar_scrolling_view_behavior"` est utilisé dans les propriétés de ViewPager
- Un RecyclerView est utilisé dans les fragments ViewPager

Voici le fichier XML de présentation utilisé dans une activité:

```
<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>
```

Résultat:



Voir le poids

L'un des attributs les plus utilisés pour LinearLayout est le poids de ses vues enfants. Le poids définit la quantité d'espace consommée par une vue par rapport aux autres vues d'un LinearLayout.

Le poids est utilisé lorsque vous souhaitez donner un espace d'écran spécifique à un composant par rapport à un autre.

Propriétés clés :

- `weightSum` est la somme globale des poids de toutes les vues enfants. Si vous ne spécifiez

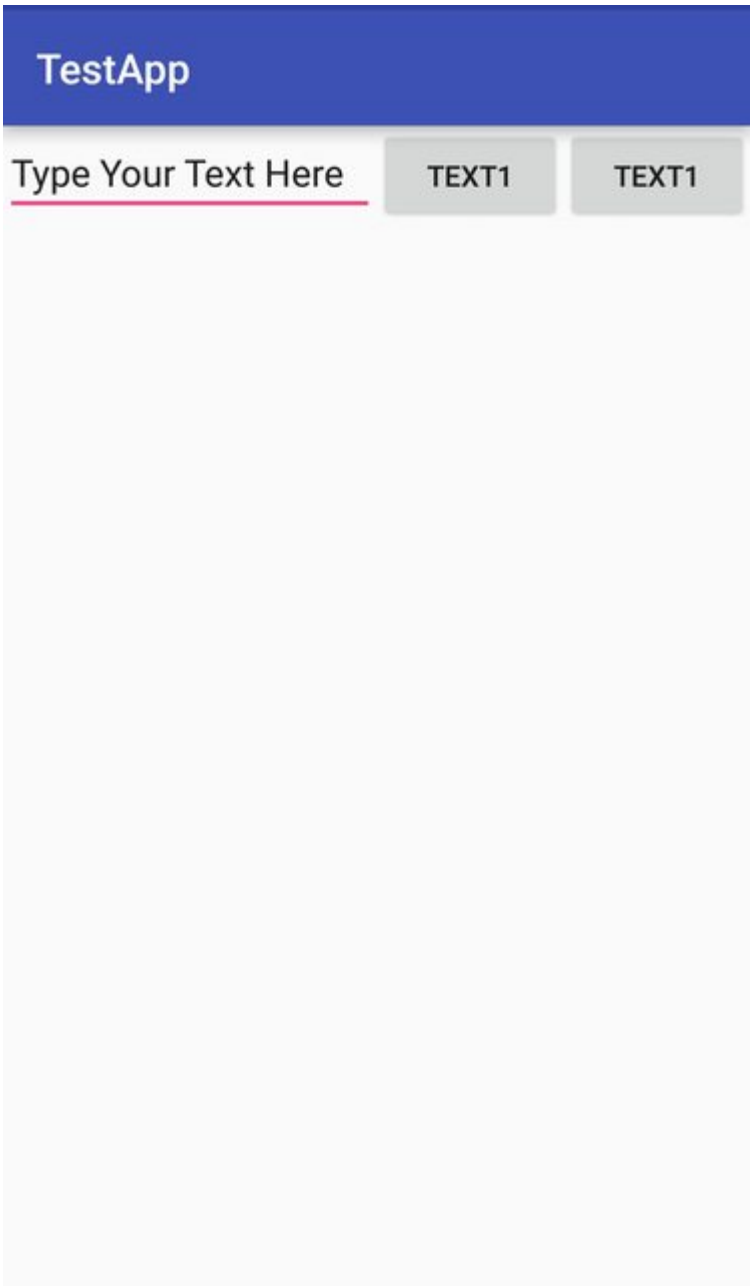
pas le `weightSum` , le système calculera lui-même la somme de tous les poids.

- `layout_weight` spécifie la quantité d'espace hors de la somme de poids totale que le widget va occuper.

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>
```

La sortie est la suivante:



Maintenant, même si la taille de l'appareil est plus grande, le EditText prendra 2/4 de l'espace de l'écran. Par conséquent, l'apparence de votre application est visible sur tous les écrans.

Note: Ici, la `layout_width` est conservée `0dp` car l'espace du widget est divisé horizontalement. Si les widgets doivent être alignés verticalement, `layout_height` sera défini sur `0dp` . Ceci est fait pour augmenter l'efficacité du code car au moment de l'exécution, le système ne tentera pas de calculer la largeur ou la hauteur respectivement, car celle-ci est gérée par le poids. Si vous utilisez plutôt `wrap_content` le système essaiera de calculer d'abord la largeur / hauteur avant d'appliquer l'attribut de pondération qui provoque un autre cycle de calcul.

Créer LinearLayout par programmation

Hiérarchie

```
- 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

Chaque groupe `ViewGroup` (par exemple, `LinearLayout` , `RelativeLayout` , `CoordinatorLayout` , etc.) doit stocker des informations sur les propriétés de ses enfants. À propos de la façon dont ses enfants sont disposés dans le `ViewGroup` . Ces informations sont stockées dans des objets d'une classe d'encapsuleur `ViewGroup.LayoutParams` .

Pour inclure des paramètres spécifiques à un type de `ViewGroups` particulier, `ViewGroups` utilise des sous-classes de la classe `ViewGroup.LayoutParams` .

Par exemple pour

- `LinearLayout` c'est `LinearLayout.LayoutParams`
- `RelativeLayout` c'est `RelativeLayout.LayoutParams`
- `CoordinatorLayout` en `CoordinatorLayout.LayoutParams`
- ...

La plupart des `ViewGroups` réutilisent la possibilité de définir des `margins` pour leurs enfants, afin de ne pas sous- `ViewGroup.LayoutParams` directement, mais de sous- `ViewGroup.MarginLayoutParams` (qui est elle-même une sous-classe de `ViewGroup.LayoutParams`).

`LayoutParams` en xml

`LayoutParams`

objets `LayoutParams` sont créés sur la base du fichier `xml` mise en page gonflé.

```
<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>
```

Tous les paramètres commençant par `layout_` spécifient le fonctionnement de la mise en page **englobante** . Lorsque la disposition est gonflée, ces paramètres sont `LayoutParams` dans un objet `LayoutParams` approprié, qui sera ensuite utilisé par la `Layout` pour positionner correctement une `View` particulière dans le `ViewGroup` . Les autres attributs d'une `View` sont directement liés à la `View` et sont traités par la `View` elle-même.

Pour `TextView` :

- `layout_width` , `layout_height` et `layout_gravity` seront stockés dans un objet `LinearLayout.LayoutParams` et utilisés par `LinearLayout`
- `gravity` , `text` et `textColor` seront utilisés par le `TextView` lui-même

Pour `ImageView` :

- `layout_width` , `layout_height` et `layout_weight` seront stockés dans un objet `LinearLayout.LayoutParams` et utilisés par `LinearLayout`
- `background` , `scaleType` et `src` seront utilisés par `ImageView`

Obtenir l'objet `LayoutParams`

`getLayoutParams` est une méthode `View's` qui permet de récupérer un objet `LayoutParams` cours.

`LayoutParams` objet `LayoutParams` étant directement lié au `ViewGroup` **englobant** , cette méthode renvoie une valeur non nulle uniquement lorsque `View` est associé au `ViewGroup` . Vous devez garder à l'esprit que cet objet peut ne pas être présent à tout moment. Surtout vous ne devriez pas dépendre de l'avoir à l'intérieur `View's` constructeur `View's` .

```
public class ExampleView extends View {

    public ExampleView(Context context) {
        super(context);
        setupView(context);
    }

    public ExampleView(Context context, AttributeSet attrs) {
        super(context, attrs);
        setupView(context);
    }

    public ExampleView(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
        setupView(context);
    }

    private void setupView(Context context) {
        if (getLayoutParams().height == 50){  // DO NOT DO THIS!
                                              // This might produce NullPointerException
            doSomething();
        }
    }

    //...
}
```

Si vous voulez dépendre de l'objet `LayoutParams` , vous `onAttachedToWindow` plutôt utiliser la méthode `onAttachedToWindow` .

```
public class ExampleView extends View {

    public ExampleView(Context context) {
        super(context);
    }

    public ExampleView(Context context, AttributeSet attrs) {
        super(context, attrs);
    }

    public ExampleView(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
    }

    @Override
    protected void onAttachedToWindow() {
        super.onAttachedToWindow();
        if (getLayoutParams().height == 50) { // getLayoutParams() will NOT return null here
            doSomething();
        }
    }

    //...
}
```

Objet `LayoutParams`

Vous devrez peut-être utiliser des fonctionnalités spécifiques à un `ViewGroup` particulier (par

exemple, vous pouvez modifier les règles d'un `RelativeLayout`). Pour cela, vous devez savoir comment `ViewGroup.LayoutParams` correctement l'objet `ViewGroup.LayoutParams` .

Cela peut être un peu compliqué lors de l'obtention d'un objet `LayoutParams` pour un enfant `View` qui est en réalité un autre `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>
```

IMPORTANT: le type d'objet `LayoutParams` est directement lié au type du groupe de **vues** `ViewGroup` .

Casting ***incorrect*** :

```
FrameLayout innerLayout = (FrameLayout)findViewById(R.id.inner_layout);
FrameLayout.LayoutParams par = (FrameLayout.LayoutParams) innerLayout.getLayoutParams();
                                        // INCORRECT! This will produce ClassCastException
```

Coulée correcte :

```
FrameLayout innerLayout = (FrameLayout)findViewById(R.id.inner_layout);
LinearLayout.LayoutParams par = (LinearLayout.LayoutParams) innerLayout.getLayoutParams();
                                       // CORRECT! the enclosing layout is a LinearLayout
```

Lire Mises en page en ligne: https://riptutorial.com/fr/android/topic/94/mises-en-page

Chapitre 173: Mode PorterDuff

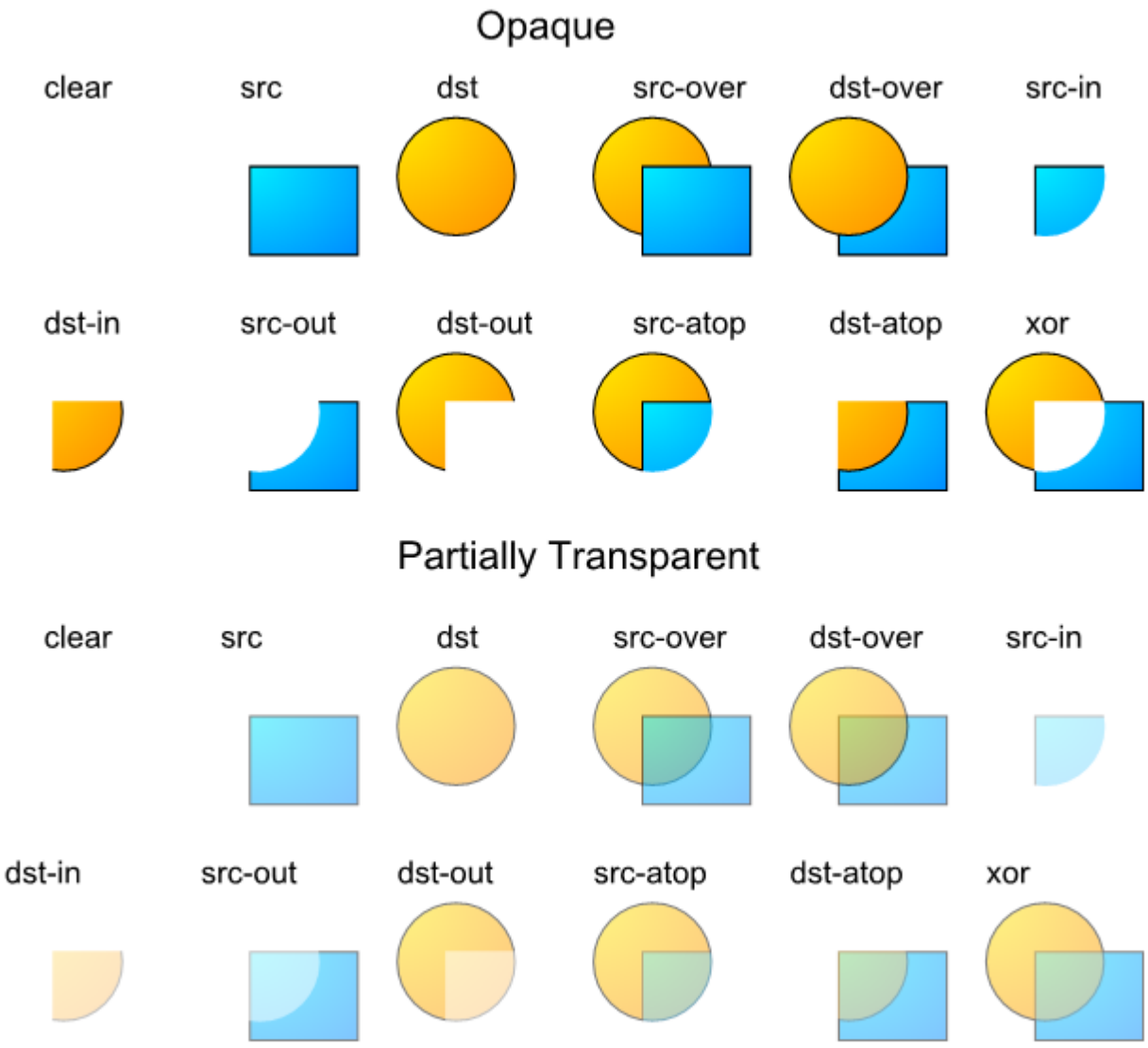
Introduction

PorterDuff est décrit comme un moyen de combiner des images comme s'il s'agissait de «morceaux de carton de forme irrégulière» superposés les uns sur les autres, ainsi que d'un schéma de mélange des parties superposées.

Remarques

"Porter Duff" est en soi une technique de composition alpha nommée d'après un article de Thomas Porter et Tom Duff.

En résumé, la technique prend deux images avec un canal alpha et génère l'image de sortie en combinant les valeurs de pixels de deux images. Les différents modes de combinaison produisent une image de sortie différente. Par exemple, dans l'image suivante, la forme bleue (source, pixels existants) est combinée avec la forme jaune (destination, nouveaux pixels) dans différents modes:



Examples

Créer un filtre de couleur PorterDuff

`PorterDuff.Mode` est utilisé pour créer un `PorterDuffColorFilter` . Un filtre de couleur modifie la couleur de chaque pixel d'une ressource visuelle.

```
ColorFilter filter = new PorterDuffColorFilter(Color.BLUE, PorterDuff.Mode.SRC_IN);
```

Le filtre ci-dessus colorera les pixels non transparents en couleur bleue.

Le filtre de couleur peut être appliqué à un `Drawable` :

```
drawable.setColorFilter(filter);
```

Il peut être appliqué à un `ImageView` :

```
imageView.setColorFilter(filter);
```

En outre, il peut être appliqué à une `Paint` , de sorte que la couleur dessinée à l'aide de cette peinture soit modifiée par le filtre:

```
paint.setColorFilter(filter);
```

Créer un PorterDuff XferMode

Un `Xfermode` (pensez au mode "transfer") fonctionne comme une étape de transfert dans le dessin du pipeline. Lorsqu'un `Xfermode` est appliqué à une `Paint` , les pixels dessinés avec la peinture sont combinés avec les pixels sous-jacents (déjà dessinés) selon le mode:

```
paint.setColor(Color.BLUE);
paint.setXfermode(new PorterDuffXfermode(PorterDuff.Mode.SRC_IN));
```

Maintenant, nous avons une peinture bleue. Toute forme dessinée colorera les pixels bleus déjà transparents existants dans la zone de la forme.

Appliquer un masque radial (vignette) à un bitmap à l'aide de PorterDuffXfermode

```
/**
 * Apply a radial mask (vignette, i.e. fading to black at the borders) to a bitmap
 * @param imageToApplyMaskTo Bitmap to modify
 */
public static void radialMask(final Bitmap imageToApplyMaskTo) {
    Canvas canvas = new Canvas(imageToApplyMaskTo);

    final float centerX = imageToApplyMaskTo.getWidth() * 0.5f;
    final float centerY = imageToApplyMaskTo.getHeight() * 0.5f;
    final float radius = imageToApplyMaskTo.getHeight() * 0.7f;

    RadialGradient gradient = new RadialGradient(centerX, centerY, radius,
            0x00000000, 0xFF000000, android.graphics.Shader.TileMode.CLAMP);

    Paint p = new Paint();
    p.setShader(gradient);
    p.setColor(0xFF000000);
    p.setXfermode(new PorterDuffXfermode(PorterDuff.Mode.DST_OUT));
    canvas.drawRect(0, 0, imageToApplyMaskTo.getWidth(), imageToApplyMaskTo.getHeight(), p);
}
```

Lire Mode PorterDuff en ligne: https://riptutorial.com/fr/android/topic/377/mode-porterduff

Chapitre 174: Modèles de conception

Introduction

Les modèles de conception sont des meilleures pratiques formalisées que le programmeur peut utiliser pour résoudre des problèmes courants lors de la conception d'une application ou d'un système.

Les modèles de conception peuvent accélérer le processus de développement en fournissant des paradigmes de développement éprouvés et éprouvés.

La réutilisation de modèles de conception évite les problèmes subtils pouvant causer des problèmes majeurs et améliore la lisibilité du code pour les codeurs et les architectes familiarisés avec les modèles.

Examples

Exemple de classe Singleton

Motif Singleton Java

Pour implémenter le modèle Singleton, nous avons différentes approches, mais toutes ont des concepts communs suivants.

- Constructeur privé pour restreindre l'instanciation de la classe à partir d'autres classes.
- Variable statique privée de la même classe qui est la seule instance de la classe.
- Méthode statique publique qui retourne l'instance de la classe, c'est l'accès global
- point pour le monde extérieur pour obtenir l'instance de la classe singleton.

```
/**
 * Singleton class.
 */
public final class Singleton {

  /**
   * Private constructor so nobody can instantiate the class.
   */
  private Singleton() {}

  /**
   * Static to class instance of the class.
   */
  private static final Singleton INSTANCE = new Singleton();

  /**
   * To be called by user to obtain instance of the class.
   *
   * @return instance of the singleton.
   */
  public static Singleton getInstance() {
```

```
    return INSTANCE;
  }
}
```

Motif d'observateur

Le modèle d'observateur est un modèle courant, largement utilisé dans de nombreux contextes. Un exemple concret peut être tiré de YouTube: lorsque vous aimez une chaîne et que vous souhaitez obtenir toutes les nouvelles et regarder les nouvelles vidéos de cette chaîne, vous devez vous abonner à cette chaîne. Ensuite, chaque fois que cette chaîne publie des informations, vous (et tous les autres abonnés) recevrez une notification.

Un observateur aura deux composants. L'un est un diffuseur (canal) et l'autre est un récepteur (vous ou tout autre abonné). Le radiodiffuseur traitera toutes les instances de récepteur qui y ont souscrit. Lorsque le diffuseur déclenche un nouvel événement, il l'annonce à toutes les instances du récepteur. Lorsque le récepteur reçoit un événement, il doit réagir à cet événement, par exemple en allumant YouTube et en lisant la nouvelle vidéo.

Implémenter le motif de l'observateur

1. Le radiodiffuseur doit fournir des méthodes permettant aux destinataires de s'abonner et de se désabonner. Lorsque le radiodiffuseur déclenche un événement, les abonnés doivent être avertis qu'un événement s'est produit:

```
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. Le récepteur doit implémenter une méthode qui gère l'événement depuis le diffuseur:

```
interface Subscriber {
    void doSubscribe(Channel channel);
    void doUnsubscribe(Channel channel);
    void handleEvent();  // Process the new event.
}
```

Lire Modèles de conception en ligne: https://riptutorial.com/fr/android/topic/9949/modeles-de-conception

Chapitre 175: Moshi

Introduction

Moshi est une bibliothèque JSON moderne pour Android et Java. Il est facile d'analyser JSON en objets Java et Java en JSON.

Remarques

N'oubliez pas de toujours lire le README !

Examples

JSON dans 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);
```

sérialiser les objets Java en JSON

```
BlackjackHand blackjackHand = new BlackjackHand(
    new Card('6', SPADES),
    Arrays.asList(new Card('4', CLUBS), new Card('A', HEARTS)));

Moshi moshi = new Moshi.Builder().build();
JsonAdapter<BlackjackHand> jsonAdapter = moshi.adapter(BlackjackHand.class);

String json = jsonAdapter.toJson(blackjackHand);
System.out.println(json);
```

Adaptateurs intégrés

Moshi a un support intégré pour lire et écrire les principaux types de données Java:

- Primitives (int, float, char ...) et leurs contreparties encadrées (Integer, Float, Character ...).
- Tableaux
- Collections
- Des listes
- Ensembles
- Cartes Cordes Enums

Il prend en charge vos classes de modèle en les écrivant champ par champ. Dans l'exemple ci-

dessus, Moshi utilise ces classes:

```
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"
    }
  ]
}
```

Lire Moshi en ligne: https://riptutorial.com/fr/android/topic/8744/moshi

Chapitre 176: Moyen rapide pour configurer Retrolambda sur un projet Android.

Introduction

Retrolambda est une bibliothèque qui permet d'utiliser les expressions lambda Java, les références de méthode et les instructions try-with-resources sous Java 7, 6 ou 5.

Le plug-in Gradle Retrolambda permet d'intégrer Retrolambda dans une version basée sur Gradle. Cela permet par exemple d'utiliser ces constructions dans une application Android, car le développement Android standard ne prend pas encore en charge Java 8.

Examples

Configuration et exemple d'utilisation:

Étapes d'installation:

1. Téléchargez et installez jdk8.
2. Ajoutez ce qui suit au fichier build.gradle principal de votre projet.

```
buildscript {
    repositories {
        mavenCentral()
    }

    dependencies {
        classpath 'me.tatarka:gradle-retrolambda:3.2.3'
    }
}
```

3. Ajoutez maintenant ceci à votre build.gradle de module d'application

```
apply plugin: 'com.android.application' // or apply plugin: 'java'
apply plugin: 'me.tatarka.retrolambda'
```

4. Ajoutez ces lignes au build.gradle de votre module d'application pour informer l'EDI du niveau de langue:

```
android {
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
}
```

Exemple:

Donc des choses comme ça:

```
button.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        log("Clicked");
    }
});
```

Deviens ça:

```
button.setOnClickListener(v -> log("Clicked"));
```

Lire Moyen rapide pour configurer Retrolambda sur un projet Android. en ligne: https://riptutorial.com/fr/android/topic/8822/moyen-rapide-pour-configurer-retrolambda-sur-un-projet-android-

Chapitre 177: Multidex et la limite de méthode Dex

Introduction

DEX désigne les fichiers de code-octet exécutables de l'application Android (APK) sous la forme de fichiers Dalvik Executable (DEX), qui contiennent le code compilé utilisé pour exécuter votre application.

La spécification Dalvik Executable limite le nombre total de méthodes pouvant être référencées dans un seul fichier DEX à 65 536 (64 Ko), y compris les méthodes d'infrastructure Android, les méthodes de bibliothèque et les méthodes de votre propre code.

Pour surmonter cette limite, vous devez configurer le processus de génération de votre application afin de générer plusieurs fichiers DEX, appelés Multidex.

Remarques

Qu'est ce que le dex?

Dex est le nom du format de fichier et du codage auquel le code Java Android est compilé. Les premières versions d'Android chargeraient et exécuteraient des binaires `dex` directement dans une machine virtuelle nommée Dalvik. Les versions plus récentes d'Android utilisent Android Runtime (ART), qui traite les fichiers `dex` comme une représentation intermédiaire et effectue de nouvelles compilations avant d'exécuter l'application.

Dex est un format de fichier très ancien, en termes de durée de vie des smartphones, conçu pour les périphériques dont la mémoire principale était mesurée en dizaines de mégaoctets. Les limites de conception de ces jours sont restées avec nous à ce jour.

Le problème:

Le format de fichier `dex` encode une limite au nombre de méthodes pouvant être référencées dans un seul fichier binaire. Étant donné que la partie du format de fichier qui stocke le nombre de références comporte deux octets, le nombre maximal de références de méthode est `0xFFFF` ou 65535. Si une application contient plus de ce nombre de références de méthode, la compilation échouera.

Que faire à ce sujet:

Google a fourni un moyen de contourner ce problème, appelé Multidex. Il comporte des composants à la compilation et à l'exécution. Comme son nom l'indique, au moment de la compilation, il divisera le code entre un ou plusieurs fichiers `dex` . Lors de l'exécution, le

`ClassLoader` par défaut `ClassLoader` à rechercher les classes à partir de ces fichiers.

Cette approche fonctionne bien sur les nouveaux périphériques, mais présente des inconvénients importants. Il peut augmenter le temps de démarrage de l' application de façon spectaculaire, et sur les appareils plus anciens peuvent causer `Application Not Responding` des échecs.

Multidex, bien qu'efficace, devrait être évité si possible.

Comment éviter la limite:

Avant de configurer votre application pour permettre l'utilisation de références de méthode 64 Ko ou plus, vous devez prendre des mesures pour réduire le nombre total de références appelées par votre code d'application, y compris les méthodes définies par votre code d'application ou les bibliothèques incluses. Les stratégies suivantes peuvent vous aider à éviter d'atteindre la limite de référence dex:

- **Examinez les dépendances directes et transitives de votre application** - Assurez-vous que toute dépendance de bibliothèque importante que vous incluez dans votre application est utilisée d'une manière qui dépasse la quantité de code ajoutée à l'application. Un anti-pattern commun est d'inclure une très grande bibliothèque car quelques méthodes utilitaires étaient utiles. La réduction des dépendances du code de votre application peut souvent vous aider à éviter la limite de référence dex.
- **Supprimez le code inutilisé avec ProGuard** - Configurez les paramètres ProGuard pour que votre application exécute ProGuard et assurez-vous que la réduction des créations est activée. L'activation du rétrécissement garantit que vous ne livrez pas de code inutilisé avec vos APK.

Le premier point nécessite une diligence et une discipline de la part du développeur. Lors de l'intégration de bibliothèques tierces, il faut tenir compte de la taille de la bibliothèque. Par exemple, deux bibliothèques JSON populaires sont Jackson et Gson. Fonctionnellement, ils sont assez similaires, mais Gson a tendance à voir une plus grande utilisation dans Android. L'une des raisons est que Jackson pèse environ 9 000 méthodes, alors que Gson en contribue 1 900.

Plusieurs outils sont disponibles pour aider les développeurs à suivre la taille de leur application:

- dexcount-gradle-plugin indique le nombre de références de méthode dans votre APK ou AAR sur chaque version
- dex-method-count est un outil en ligne de commande qui compte le nombre de références de méthode dans un APK
- www.methodscount.com est un service Web qui comptera les références de méthode dans tout fichier APK que vous téléchargez.

Examples

Multidex en utilisant MultiDexApplication directement

Utilisez cette option si vous n'avez pas besoin d'une sous-classe d' `Application` .

C'est l'option la plus simple, mais de cette façon, vous ne pouvez pas fournir votre propre sous-classe d' `Application` . Si une sous-classe d' `Application` est nécessaire, vous devrez passer à l'une des autres options pour le faire.

Pour cette option, spécifiez simplement le nom de classe complet `android.support.multidex.MultiDexApplication` pour la propriété `android:name` de la balise d' `application` dans le fichier AndroidManifest.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.android.multidex.myapplication">
    <application
        ...
        android:name="android.support.multidex.MultiDexApplication">
        ...
    </application>
</manifest>
```

Multidex en étendant l'application

Utilisez cette option si votre projet nécessite une sous-classe `Application` .

Spécifiez cette sous-classe d' `Application` à l'aide de la propriété `android:name` dans le fichier manifeste de la balise d' `application` .

Dans la sous-classe `Application` , ajoutez la méthode `attachBaseContext()` et, dans cette méthode, appelez `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);
    }
}
```

Assurez-vous que la sous-classe `Application` est spécifiée dans la balise d' `application` de votre AndroidManifest.xml:

```
<application
    android:name="com.example.MyApplication"
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name">
</application>
```

Activation de Multidex

Pour activer une configuration multidex, vous avez besoin:

- changer votre configuration de build Gradle
- utiliser `MultiDexApplication` ou activer MultiDex dans votre classe `Application`

Configuration de Gradle

Dans `app/build.gradle` ajoutez ces parties:

```
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'
}
```

Activer MultiDex dans votre application

Ensuite, effectuez l'une des trois options suivantes:

- Multidex en étendant l'application
- Multidex en étendant MultiDexApplication
- Multidex en utilisant MultiDexApplication directement

Lorsque ces paramètres de configuration sont ajoutés à une application, les outils de génération Android construisent un dex (classes.dex) principal et prennent en charge (classes2.dex, classes3.dex) selon les besoins.
Le système de compilation les intégrera ensuite dans un fichier APK pour la distribution.

Méthode de comptage Références sur chaque version (Dexcount Gradle Plugin)

Le plug-in dexcount compte les méthodes et le nombre de ressources de classe après une génération réussie.

Ajoutez le plugin dans l' `app/build.gradle` :

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

buildscript {
    repositories {
        mavenCentral() // or jcenter()
    }

    dependencies {
        classpath 'com.getkeepsafe.dexcount:dexcount-gradle-plugin:0.5.5'
    }
}
```

Appliquez le plugin dans le fichier `app/build.gradle` :

```
apply plugin: 'com.getkeepsafe.dexcount'
```

Recherchez les données de sortie générées par le plug-in dans:

../app/build/outputs/dexcount

Le graphique .html est particulièrement utile dans:

../app/build/outputs/dexcount/debugChart/index.html

Multidex en étendant MultiDexApplication

Cela ressemble beaucoup à l'utilisation d'une sous-classe `Application` et au `attachBaseContext()` méthode `attachBaseContext()` .

Cependant, en utilisant cette méthode, vous n'avez pas besoin de remplacer `attachBaseContext()` car cela est déjà fait dans la super-classe `MultiDexApplication` .

Étendez `MultiDexApplication` au lieu de `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()

     //.........
}
```

Ajoutez cette classe à votre AndroidManifest.xml exactement comme si vous étendiez l'application:

```
<application
    android:name="com.example.MyApplication"
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name">
</application>
```

Lire Multidex et la limite de méthode Dex en ligne:
https://riptutorial.com/fr/android/topic/1887/multidex-et-la-limite-de-methode-dex

Chapitre 178: MVVM (Architecture)

Remarques

Syntaxe quirks avec DataBinding

Lors de la liaison d'une fonction viewModel à une propriété dans xml, certains préfixes de fonctions tels que `get` ou `is` sont supprimés. Par exemple. `ViewModel::getFormattedText` sur le ViewModel deviendra `@{viewModel.formattedText}` lors de la liaison à une propriété au format XML. De même avec `ViewModel::isContentVisible` -> `@{viewModel.contentVisible}` (notation Java Bean)

Les classes de liaison générées, telles que `ActivityMainBinding` portent le nom du fichier XML pour lequel elles créent des liaisons, et non la classe Java.

Fixations personnalisées

Dans le fichier activity_main.xml, je définis l'attribut textColor sur l' `app` et non l'espace de noms `android` . Pourquoi donc? Étant donné qu'un attribut personnalisé est défini pour l'attribut `textColor` qui résout un ID de ressource ColorRes envoyé par ViewModel à une couleur réelle.

```
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);
    }

}
```

Pour plus d'informations sur son fonctionnement, consultez DataBinding Library: Custom Setters

Attendez ... il y a de la logique dans votre xml !!!

Vous pourriez faire valoir que les choses que je fais dans XML pour `android:visibility` et `app:textColor` sont fausses / anti-patterns dans le contexte MVVM car il y a une logique de vue à mon avis. Cependant, je pense qu'il est plus important pour moi de garder les dépendances Android hors de mon ViewModel pour des raisons de test.

En outre, que fait vraiment `app:textColor` ? Il ne résout qu'un pointeur de ressource sur la couleur

réelle qui lui est associée. Donc, le ViewModel décide toujours quelle couleur est affichée en fonction de certaines conditions.

En ce qui concerne l' `android:visibility` me semble liée à la manière dont la méthode est nommée. Il est donc normal d'utiliser l'opérateur ternaire ici. En raison du nom `isLoadingVisible` et `isContentVisible` il n'y a vraiment aucun doute sur la résolution de chaque résultat dans la vue. Je pense donc qu'il s'agit plutôt d'exécuter une commande donnée par ViewModel que de faire de la logique de vue.

D'un autre côté, je suis d'accord que l'utilisation de `viewModel.isLoading ? View.VISIBLE : View.GONE` serait une mauvaise chose à faire car vous faites des suppositions dans la vue de ce que cet état signifie pour la vue.

Matériel utile

Les ressources suivantes m'ont beaucoup aidé à comprendre ce concept:

- Jeremy Likness - Explication de Model-View-ViewModel (MVVM) (C #) (08.2010)
- Shamlia Shukkur - Comprendre les bases du modèle de conception MVVM (C #) (03.2013)
- Frode Nilsen - Liaison de données Android: Goodbye Presenter, Hello ViewModel! (07.2015)
- Joe Birch - Approcher Android avec MVVM (09.2015)
- Florina Muntenescu - Modèles d'architecture Android Partie 3: Model-View-ViewModel (10.2016)

Examples

Exemple MVVM utilisant la bibliothèque DataBinding

L'intérêt de MVVM est de séparer les couches contenant la logique de la couche de vue.

Sur Android, nous pouvons utiliser la bibliothèque DataBinding pour nous aider avec ceci et rendre la plus grande partie de notre logique testable sans nous soucier des dépendances Android.

Dans cet exemple, je vais montrer les composants centraux d'une application simple et stupide qui effectue les opérations suivantes:

- Au démarrage, simulez un appel réseau et affichez un compteur de chargement
- Afficher une vue avec un compteur de clics TextView, un message TextView et un bouton pour incrémenter le compteur
- Sur le bouton cliquez sur le compteur de mise à jour et mettez à jour la couleur du compteur et le texte du message si le compteur atteint un certain nombre

Commençons par la couche de vue:

`activity_main.xml` :

Si vous ne connaissez pas le fonctionnement de DataBinding, vous devriez probablement prendre 10 minutes pour vous familiariser avec celui-ci. Comme vous pouvez le voir, tous les champs que

vous mettrez à jour avec setters sont liés à des fonctions de la variable viewModel.

Si vous avez une question à propos de l' `android:visibility` propriétés `android:visibility` ou `app:textColor` , consultez la section "Remarques".

```
<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>
```

Suivant la couche modèle. J'ai ici:

- deux champs qui représentent l'état de l'application
- les getters à lire le nombre de clics et l'état d'excitation

- une méthode pour incrémenter mon nombre de clics
- une méthode pour restaurer un état précédent (important pour les changements d'orientation)

Je définis également ici un «état d'excitation» qui dépend du nombre de clics. Cela sera utilisé plus tard pour mettre à jour la couleur et le message sur la vue.

Il est important de noter qu'il n'y a pas d'hypothèses dans le modèle sur la manière dont l'état peut être affiché pour l'utilisateur!

`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;
        }
    }
}
```

Suivant le ViewModel.

Cela déclenchera des modifications sur le modèle et formatera les données du modèle pour les afficher dans la vue. Notez que c'est ici que nous évaluons quelle représentation graphique est appropriée pour l'état donné par le modèle (`resolveCounterColor` et `resolveLabelText`). Ainsi, nous pourrions par exemple facilement implémenter un `UnderachieverClickerModel` qui présente des seuils inférieurs pour l’état d’excitation sans toucher au code de viewModel ou de la vue.

Notez également que ViewModel ne contient aucune référence pour afficher les objets. Toutes les propriétés sont liées via les annotations `@Bindable` et mises à jour lorsque `notifyChange()` (signale que toutes les propriétés doivent être mises à jour) ou `notifyPropertyChanged(BR.propertyName)` (signale que ces propriétés doivent être mises à jour).

`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;
        }
    }

}
```

Lier tout ensemble dans l'activité!

Nous voyons ici la vue initialisant le viewModel avec toutes les dépendances dont il pourrait avoir besoin, qui doivent être instanciées à partir d'un contexte android.

Une fois le viewModel initialisé, il est lié à la mise en page XML par le biais de DataBindingUtil (veuillez vérifier la section "Syntaxe" pour la dénomination des classes générées).

Notez que les abonnements sont abonnés à cette couche car nous devons les désabonner lorsque l'activité est suspendue ou détruite pour éviter les fuites de mémoire et les NPE. La persistance et le rechargement de la vueState on OrientationChanges se déclenchent ici

`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;
    }
}
```

Pour voir tout ce qui se passe, consultez cet exemple de projet .

Lire MVVM (Architecture) en ligne: https://riptutorial.com/fr/android/topic/7549/mvvm--architecture-

Chapitre 179: NavigationView

Remarques

NavigationView représente un menu de navigation standard pour l'application. Le contenu du menu peut être rempli par un fichier de ressources de menu.

Avant d'utiliser `NavigationView` vous devez ajouter la dépendance de bibliothèque du support de conception dans le fichier build.gradle:

```
 dependencies {
    compile 'com.android.support:design:24.2.0'
}
```

Documentation officielle:

https://developer.android.com/reference/android/support/design/widget/NavigationView.html

Spécifications de conception matérielle:

https://material.google.com/patterns/navigation-drawer.html#navigation-drawer-content

Examples

Comment ajouter le NavigationView

Pour utiliser un NavigationView, ajoutez simplement la dépendance dans le fichier `build.gradle` comme décrit dans la section Remarques.

Puis ajoutez le `NavigationView` dans la mise en page

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v4.widget.DrawerLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/drawer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"
    tools:openDrawer="start">

    <include
        layout="@layout/app_bar_main"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

    <android.support.design.widget.NavigationView
```

```
        android:id="@+id/nav_view"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_gravity="start"
        app:headerLayout="@layout/nav_header_main"
        app:menu="@menu/activity_main_drawer" />

</android.support.v4.widget.DrawerLayout>
```

res/layout/nav_header_main.xml : La vue qui sera affichée en haut du tiroir

```
<?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 Une couche d'abstraction pour que la barre d'outils le sépare du contenu:

```
<?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 Le contenu réel de l'activité juste pour la démonstration, ici vous mettriez votre disposition XML normale:

```
<?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>
```

Définissez votre fichier de menu comme *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
```

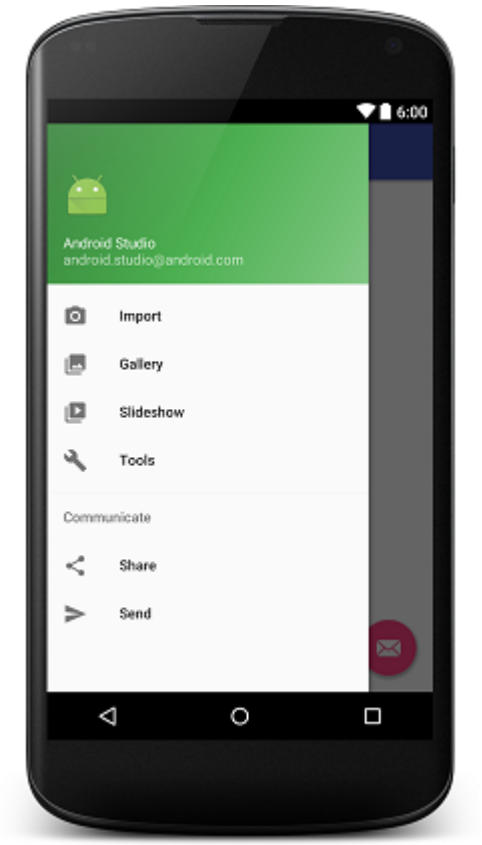
```
            android:id="@+id/nav_camera"
            android:icon="@drawable/ic_menu_camera"
            android:title="Import" />
        <item
            android:id="@+id/nav_gallery"
            android:icon="@drawable/ic_menu_gallery"
            android:title="Gallery" />
        <item
            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>
```

Et enfin le `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;
    }
}
```

Il ressemblera à ceci:

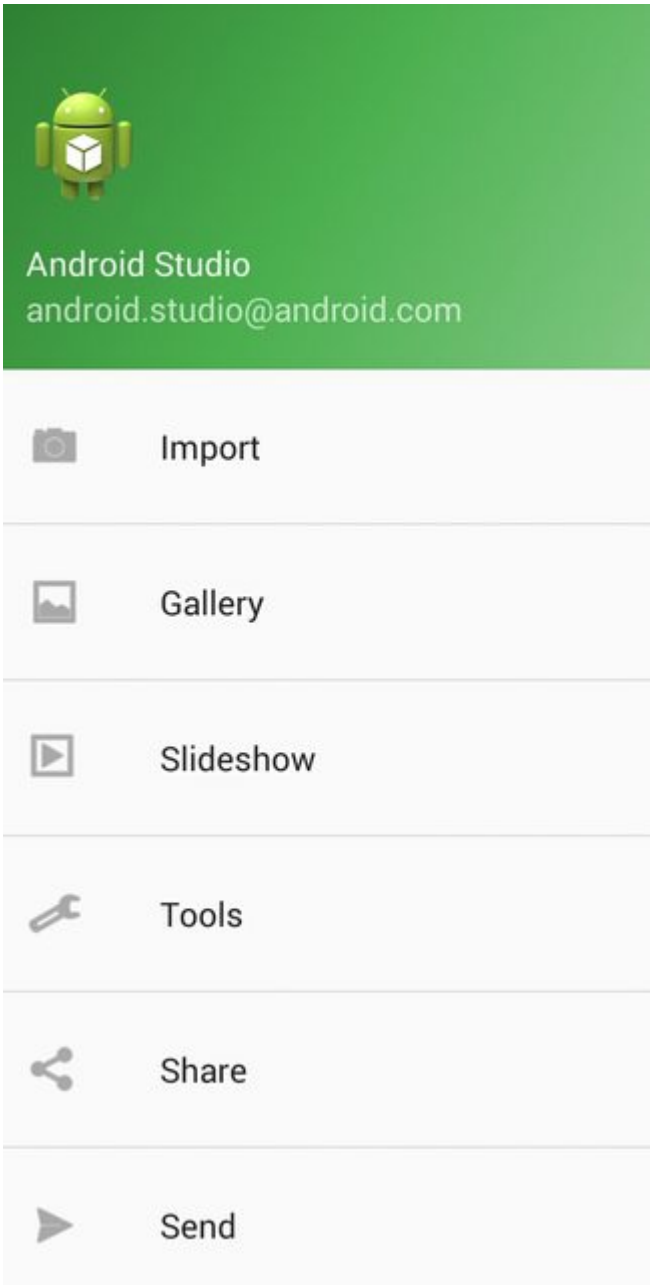


Ajouter un soulignement dans les éléments de menu

Chaque groupe se termine par un séparateur de ligne. Si chaque élément de votre menu possède son propre groupe, vous obtiendrez la sortie graphique souhaitée. Cela fonctionnera seulement si vos différents groupes ont `android:id` différent `android:id` . Aussi, dans `menu.xml` n'oubliez pas de mentionner `android:checkable="true"` pour un seul élément et `android:checkableBehavior="single"` pour un groupe d'éléments.

```
<?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>
```



Ajouter des séparateurs au menu

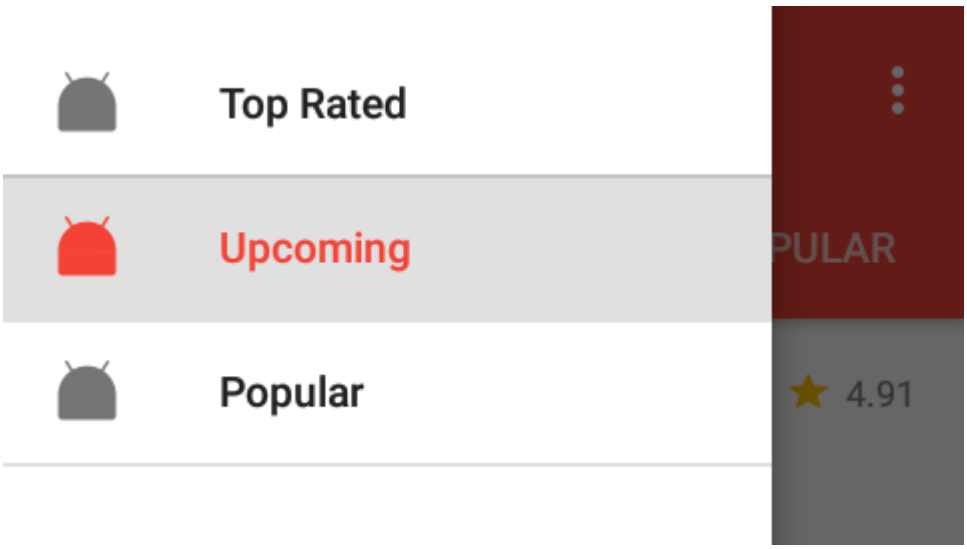
Accédez à RecyclerView dans NavigationView et ajoutez-y ItemDecoration .

```
NavigationView navigationView = (NavigationView) findViewById(R.id.nav_view);
NavigationMenuView navMenuView = (NavigationMenuView) navigationView.getChildAt(0);
navMenuView.addItemDecoration(new DividerItemDecoration(this));
```

Code pour 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);
        }
    }
}
```

Aperçu:



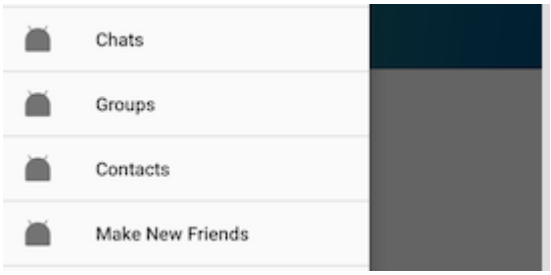
Ajouter un menu Divider en utilisant par défaut DividerItemDecoration.

Utilisez simplement la classe DividerItemDecoration par défaut:

```
NavigationView navigationView = (NavigationView) findViewById(R.id.navigation);
```

```
NavigationMenuView navMenuView = (NavigationMenuView) navigationView.getChildAt(0);
navMenuView.addItemDecoration(new
DividerItemDecoration(context,DividerItemDecoration.VERTICAL));
```

Aperçu :



Lire NavigationView en ligne: https://riptutorial.com/fr/android/topic/97/navigationview

Chapitre 180: NDK Android

Examples

Construire des exécutables natifs pour Android

projet / jni / main.c

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

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

projet / jni / Android.mk

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

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

projet / jni / Application.mk

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

Si vous souhaitez prendre en charge les périphériques exécutant des versions Android inférieures à 5.0 (API 21), vous devez compiler votre fichier binaire avec `APP_PLATFORM` défini sur une ancienne API, par exemple `android-8` . Ceci est une conséquence de la mise en place de *binaires de positionnement indépendants* (PIE) par Android 5.0, alors que les anciens appareils ne prennent pas nécessairement en charge les PIE. Par conséquent, vous devez utiliser le PIE ou le non-PIE, selon la version du périphérique. Si vous souhaitez utiliser le binaire depuis votre application Android, vous devez vérifier le niveau de l'API et extraire le binaire correct.

`APP_ABI` peut être changé en plates-formes spécifiques telles que `armeabi` pour construire le binaire pour ces architectures uniquement.

Dans le pire des cas, vous aurez à la fois un binaire PIE et un binaire non-PIE pour chaque architecture (environ 14 binaires différents utilisant ndk-r10e).

Pour construire l'exécutable:

```
cd project
ndk-build
```

Vous trouverez les binaires à `project/libs/<architecture>/hello_world` . Vous pouvez les utiliser via `ADB` (`push` et `chmod` avec l'autorisation exécutable) ou depuis votre application (extrayez et `chmod` le avec l'autorisation de l'exécutable).

Pour déterminer l'architecture de la CPU, récupérez la propriété de construction `ro.product.cpu.abi` pour l'architecture principale ou `ro.product.cpu.abilist` (sur les nouveaux périphériques) pour obtenir une liste complète des architectures prises en charge. Vous pouvez le faire en utilisant la classe `android.os.Build` depuis votre application ou en utilisant `getprop <name>` via ADB.

Comment nettoyer la construction

Si vous devez nettoyer la version:

```
ndk-build clean
```

Comment utiliser un makefile autre que Android.mk

ndk-build NDK_PROJECT_PATH = PROJECT_PATH APP_BUILD_SCRIPT = MyAndroid.mk

Comment se connecter à ndk

Assurez-vous d'abord de créer un lien avec la bibliothèque de journalisation dans votre fichier `Android.mk` :

```
LOCAL_LDLIBS := -llog
```

Ensuite, utilisez l'un des appels `__android_log_print()` :

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

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

Ou utilisez-les de manière plus pratique en définissant les macros correspondantes:

```
#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__)
```

Exemple :

```
int x = 42;
```

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

Lire NDK Android en ligne: https://riptutorial.com/fr/android/topic/492/ndk-android

Chapitre 181: Obtenir les noms de police du système et utiliser les polices

Introduction

Les exemples suivants montrent comment récupérer les noms par défaut des polices système stockées dans le répertoire */ system / fonts /* et comment utiliser une police système pour définir la police d'un élément `TextView` .

Examples

Obtenir les noms de police système

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

Application d'une police système à un TextView

Dans le code suivant, vous devez remplacer `fontsname` par le nom de la police que vous souhaitez utiliser:

```
TextView lblexample = (TextView) findViewById(R.id.lblexample);
lblexample.setTypeface(Typeface.createFromFile("/system/fonts/" + "fontsname" + ".ttf"));
```

Lire Obtenir les noms de police du système et utiliser les polices en ligne: https://riptutorial.com/fr/android/topic/10930/obtenir-les-noms-de-police-du-systeme-et-utiliser-les-polices

Chapitre 182: OkHttp

Examples

Intercepteur de journalisation

`Interceptors` sont utilisés pour intercepter les appels `OkHttp` . La raison de l'interception pourrait être de surveiller, réécrire et réessayer les appels. Il peut être utilisé pour les demandes sortantes ou les réponses entrantes.

```
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;
  }
}
```

Réécriture des réponses

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

Exemple d'utilisation de base

J'aime envelopper mon `OkHttp` dans une classe appelée `HttpClient` par exemple, et dans cette classe, j'ai des méthodes pour chacun des principaux verbes HTTP, `post` , `get` , `put` et `delete` , le plus souvent. (J'inclus généralement une interface, afin de la conserver pour l'implémenter, afin de pouvoir facilement passer à une implémentation différente, le cas échéant):

```
public class HttpClient implements HttpClientInterface{

private static final String TAG = OkHttpClient.class.getSimpleName();
public static final MediaType JSON
```

```
        = MediaType.parse("application/json; charset=utf-8");

OkHttpClient httpClient = new OkHttpClient();

@Override
public String post(String url, String json) throws IOException {
    Log.i(TAG, "Sending a post request with body:\n" + json + "\n to URL: " + url);

    RequestBody body = RequestBody.create(JSON, json);
    Request request = new Request.Builder()
            .url(url)
            .post(body)
            .build();
    Response response = httpClient.newCall(request).execute();
    return response.body().string();
}
```

La syntaxe est la même pour `put` , `get` et `delete` sauf pour 1 mot (`.put(body)`). L'utilisation est assez simple, il suffit d'appeler la méthode appropriée sur une `url` avec un payload `json` et la méthode retournera une chaîne en résultat que vous pourrez ensuite utiliser et analyser. Supposons que la réponse soit un `json` , nous pouvons facilement créer un objet `JSONObject` :

```
String response = httpClient.post(MY_URL, JSON_PAYLOAD);
JSONObject json = new JSONObject(response);
// continue to parse the response according to it's structure
```

Appel synchrone

```
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());
  }
```

Asynchrone Get Call

```
private final OkHttpClient client = new OkHttpClient();

  public void run() throws Exception {
    Request request = new Request.Builder()
        .url(yourUrl)
        .build();

    client.newCall(request).enqueue(new Callback() {
      @Override public void onFailure(Call call, IOException e) {
        e.printStackTrace();
```

```
      }

      @Override
      public void onResponse(Call call, Response response) throws IOException {
        if (!response.isSuccessful()) throw new IOException("Unexpected code " + response);

        Headers responseHeaders = response.headers();

        System.out.println(response.body().string());
      }
    });
  }
```

Affichage des paramètres du formulaire

```
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());
  }
```

Publication d'une requête en plusieurs parties

```
 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());
  }
```

Configurer OkHttp

Prenez via Maven:

```
<dependency>
  <groupId>com.squareup.okhttp3</groupId>
  <artifactId>okhttp</artifactId>
  <version>3.6.0</version>
</dependency>
```

ou Gradle:

```
compile 'com.squareup.okhttp3:okhttp:3.6.0'
```

Lire OkHttp en ligne: https://riptutorial.com/fr/android/topic/3625/okhttp

Chapitre 183: Okio

Examples

Télécharger / Implémenter

Téléchargez le dernier JAR ou récupérez-le via Maven:

```
<dependency>
    <groupId>com.squareup.okio</groupId>
    <artifactId>okio</artifactId>
    <version>1.12.0</version>
</dependency>
```

ou Gradle:

```
compile 'com.squareup.okio:okio:1.12.0'
```

Décodeur PNG

Le décodage des morceaux d'un fichier PNG montre Okio en pratique.

```
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);
```

```
    }
}
```

ByteString et Buffers

ByteString et Buffers

Okio est construit autour de deux types qui intègrent beaucoup de fonctionnalités dans une API simple:

ByteString est une séquence immuable d'octets. Pour les données de caractères, String est fondamental. ByteString est le frère perdu depuis longtemps de String, ce qui facilite le traitement des données binaires en tant que valeur. Cette classe est ergonomique: elle sait encoder et se décoder en hex, base64 et UTF-8.

La mémoire tampon est une séquence de bits modifiable. Comme ArrayList, vous n'avez pas besoin de dimensionner votre tampon à l'avance. Vous lisez et écrivez des tampons en tant que file d'attente: écrivez les données à la fin et lisez-les à partir du recto. Il n'y a aucune obligation de gérer les positions, les limites ou les capacités.

En interne, `ByteString` et `Buffer` font des choses intelligentes pour économiser le processeur et la mémoire. Si vous encodez une chaîne UTF-8 en tant que ByteString, elle met en cache une référence à cette chaîne afin que, si vous la décodez plus tard, cela ne fonctionne plus.

`Buffer` est implémenté en tant que liste de segments liée. Lorsque vous déplacez des données d'un tampon à un autre, il réaffecte la propriété des segments plutôt que de copier les données. Cette approche est particulièrement utile pour les programmes multithread: un thread qui communique avec le réseau peut échanger des données avec un thread de travail sans aucune copie ni cérémonie.

Lire Okio en ligne: https://riptutorial.com/fr/android/topic/9952/okio

Chapitre 184: Optimisation des performances

Introduction

La performance de vos applications est un élément crucial de l'expérience utilisateur. Essayez d'éviter les problèmes de performances, tels que le travail sur le thread d'interface utilisateur et apprenez comment écrire des applications rapides et réactives.

Examples

Enregistrer les recherches avec le modèle ViewHolder

Surtout dans `ListView` , vous pouvez rencontrer des problèmes de performances en faisant trop d' `findViewById()` pendant le défilement. En utilisant le modèle `ViewHolder` , vous pouvez enregistrer ces recherches et améliorer vos performances `ListView` .

Si votre élément de liste contient un seul objet `TextView` , créez une classe `ViewHolder` pour stocker l'instance:

```
static class ViewHolder {
    TextView myTextView;
}
```

Lors de la création de votre élément de liste, `ViewHolder` un objet `ViewHolder` à l'élément de liste:

```
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;
}
```

Grâce à ce modèle, `findViewById()` sera uniquement appelé lors de la création d'une nouvelle `View` et le `ListView` pourra recycler vos vues beaucoup plus efficacement.

Lire Optimisation des performances en ligne:
https://riptutorial.com/fr/android/topic/8711/optimisation-des-performances

Chapitre 185: Optimisation du noyau Android

Examples

Configuration de RAM faible

Android prend désormais en charge les appareils avec 512 Mo de RAM. Cette documentation est destinée à aider les équipementiers à optimiser et à configurer Android 4.4 pour les périphériques à faible mémoire. Plusieurs de ces optimisations sont suffisamment génériques pour pouvoir être appliquées aux versions précédentes.

Activer le drapeau du périphérique à faible RAM

Nous introduisons une nouvelle API appelée ActivityManager.isLowRamDevice () pour les applications afin de déterminer si elles doivent désactiver des fonctionnalités spécifiques gourmandes en mémoire qui fonctionnent mal sur les périphériques à faible mémoire.

Pour les périphériques de 512 Mo, cette API doit renvoyer: "true" Elle peut être activée par la propriété système suivante dans le fichier Make du périphérique.

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

Désactiver JIT

L'utilisation de la mémoire JIT à l'échelle du système dépend du nombre d'applications en cours d'exécution et de l'empreinte de code de ces applications. Le JIT établit la taille maximale du cache de code traduit et touche les pages qu'il contient. JIT coûte entre 3 et 6 millions de dollars sur un système courant.

Les grandes applications ont tendance à maximiser le cache de code assez rapidement (ce qui est par défaut 1M). En moyenne, l'utilisation du cache JIT se situe entre 100 Ko et 200 Ko par application. La réduction de la taille maximale du cache peut aider quelque peu avec l'utilisation de la mémoire, mais si elle est trop faible, le JIT passera en mode thrashing. Pour les périphériques à très faible mémoire, nous recommandons que le JIT soit entièrement désactivé.

Cela peut être réalisé en ajoutant la ligne suivante au makefile du produit:

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

Comment ajouter un gouverneur de processeur

Le gouverneur du processeur lui-même n'est constitué que d'un fichier C, qui se trouve dans kernel_source / drivers / cpufreq /, par exemple: cpufreq_smartass2.c. Vous êtes responsable de trouver le gouverneur (regardez dans un dépôt de noyau existant pour votre appareil). Mais pour pouvoir appeler et compiler ce fichier avec succès dans votre noyau, vous devrez apporter les modifications suivantes:

1. Copiez votre fichier gouverneur (cpufreq_govname.c) et naviguez jusqu'à kernel_source / drivers / cpufreq, collez-le maintenant.
2. et ouvrez Kconfig (c'est l'interface de la disposition du menu de configuration) lorsque vous ajoutez un noyau, vous voulez qu'il apparaisse dans votre configuration. Vous pouvez le faire en ajoutant le choix du gouverneur.

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

par exemple, pour smartassV2.

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

Outre l'ajout du choix, vous devez également déclarer la possibilité que le gouverneur soit choisi comme gouverneur par défaut.

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

par exemple, pour smartassV2.

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

- Vous ne trouvez pas le bon endroit pour le mettre? Recherchez simplement `"CPU_FREQ_GOV_CONSERVATIVE"` et placez le code ci-dessous, la même chose compte pour `"CPU_FREQ_DEFAULT_GOV_CONSERVATIVE"`

Maintenant que Kconfig est terminé, vous pouvez enregistrer et fermer le fichier.

3. Toujours dans le dossier `/drivers/cpufreq` , ouvrez Makefile. Dans Makefile, ajoutez la ligne correspondant au gouverneur de votre CPU. par exemple:

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

Notez que vous n'appelez pas le fichier C natif, mais le fichier O! qui est le fichier C compilé. Enregistrez le fichier.

4. Déplacer vers: `kernel_source/includes/linux` . Maintenant, ouvrez `cpufreq.h` Faites défiler jusqu'à ce que vous voyiez quelque chose comme:

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

(d'autres gouverneurs de processeurs y sont également répertoriés)

Ajoutez maintenant votre entrée avec le gouverneur de CPU sélectionné, exemple:

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

Enregistrez le fichier et fermez-le.

La configuration initiale du gouverneur CPU est maintenant terminée. Lorsque vous avez terminé toutes les étapes avec succès, vous devriez pouvoir choisir votre gouverneur dans le menu (`menuconfig` , `xconfig` , `gconfig` , `nconfig`). Une fois coché dans le menu, il sera inclus dans le noyau.

Commit qui est presque identique aux instructions ci-dessus: Ajouter la validation smartassV2 et lulzactive gouverneur

I / O Schedulers

Vous pouvez améliorer votre noyau en ajoutant de nouveaux ordonnanceurs d'E / S si nécessaire. Globalement, les gouverneurs et les ordonnanceurs sont les mêmes; ils fournissent tous deux un moyen de faire fonctionner le système. Cependant, pour les ordonnanceurs, il s'agit uniquement du flux de données d'entrée / sortie, à l'exception des paramètres du processeur. Les ordonnanceurs d'E / S décident de la planification d'une activité d'E / S à venir. Les ordonnanceurs standard tels que *noop* ou *cfq* fonctionnent très raisonnablement.

Les ordonnanceurs d'E / S peuvent être trouvés dans *kernel_source / block* .

1. Copiez le fichier du planificateur d'E / S (par exemple, *sio-iosched.c*) et accédez à *kernel_source / block* . Collez le fichier du planificateur ici.

2. Maintenant, ouvrez *Kconfig.iosched* et ajoutez votre choix à *Kconfig* , par exemple pour *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. Ensuite, définissez l'option de choix par défaut comme suit:

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

   Enregistrez le fichier.

4. Ouvrez le *Makefile* dans *kernel_source* / *block* / et ajoutez simplement la ligne suivante pour *SIO* :

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

   Enregistrez le fichier et vous avez terminé! Les ordonnanceurs d'E / S doivent maintenant apparaître dans la configuration du menu.

Commit similaire sur GitHub: ajout du programmateur Simple I / O.

Lire Optimisation du noyau Android en ligne:
https://riptutorial.com/fr/android/topic/9106/optimisation-du-noyau-android

Chapitre 186: ORMLite dans Android

Examples

Exemple Android OrmLite sur SQLite

ORMLite est un package de mappage relationnel objet qui fournit des fonctionnalités simples et légères pour la persistance des objets Java dans les bases de données SQL, tout en évitant la complexité et la surcharge de packages ORM plus standard.

Parlant pour Android, OrmLite est implémenté sur la base de données prise en charge prête à l'emploi, SQLite. Il appelle directement l'API pour accéder à SQLite.

Installation de Gradle

Pour commencer, vous devez inclure le package dans la version de construction.

```
 // https://mvnrepository.com/artifact/com.j256.ormlite/ormlite-android
compile group: 'com.j256.ormlite', name: 'ormlite-android', version: '5.0'
POJO configuration
```

Ensuite, vous devez configurer un POJO à conserver dans la base de données. Ici, il faut faire attention aux annotations:

- Ajoutez l'annotation @DatabaseTable au début de chaque classe. Vous pouvez également utiliser @Entity.
- Ajoutez l'annotation @DatabaseField juste avant chaque champ à conserver. Vous pouvez également utiliser @Column et autres.
- Ajoutez un constructeur sans argument à chaque classe avec au moins la visibilité du package.

```
@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;
    }
}
```

Dans l'exemple ci-dessus, il y a une table (form_model) avec 4 champs.

Le champ id est un index généré automatiquement.

nom d'utilisateur est un index de la base de données.

Vous trouverez plus d'informations sur l'annotation sur la documentation officielle .

Assistant de base de données

Pour continuer avec, vous devrez créer une classe d'assistance de base de données qui devrait étendre la classe OrmLiteSqliteOpenHelper.

Cette classe crée et met à niveau la base de données lorsque votre application est installée et peut également fournir les classes DAO utilisées par vos autres classes.

DAO signifie Data Access Object et fournit toutes les fonctionnalités de scrum et se spécialise dans la gestion d'une seule classe persistante.

La classe d'assistance doit implémenter les deux méthodes suivantes:

- onCreate (SQLiteDatabase sqliteDatabase, ConnectionSource connectionSource);

  onCreate crée la base de données lors de la première installation de votre application

- onUpgrade (base de données SQLiteDatabase, ConnectionSource connectionSource, int oldVersion, int newVersion);

  onUpgrade gère la mise à niveau des tables de base de données lorsque vous mettez à niveau votre application vers une nouvelle version

Exemple de classe d'assistance de base de données:

```
  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;
        }
```

```
    }
```

Objet persistant à SQLite

Enfin, la classe qui conserve l'objet dans la base de données.

```
     public class ReviewPresenter {
    Dao<FormModel, Long> simpleDao;


    public ReviewPresenter(Application application) {
        this.application = (GaiaApplication) application;
        simpleDao = this.application.getHelper().getDao();
    }

    public void storeFormToSqLite(FormModel form) {

        try {
            simpleDao.create(form);
        } catch (SQLException e) {
            e.printStackTrace();
        }
        List<FormModel> list = null;
        try {
// query for all of the data objects in the database
            list = simpleDao.queryForAll();
        } catch (SQLException e) {
            e.printStackTrace();
        }
// our string builder for building the content-view
        StringBuilder sb = new StringBuilder();
        int simpleC = 1;
        for (FormModel simple : list) {
            sb.append('#').append(simpleC).append(":
").append(simple.getUsername()).append('\n');
            simpleC++;
        }
        System.out.println(sb.toString());
    }

    //Query to database to get all forms by username
    public List<FormModel> getAllFormsByUsername(String username) {
        List<FormModel> results = null;
        try {
            results = simpleDao.queryBuilder().where().eq("username",
PreferencesManager.getInstance().getString(Constants.USERNAME)).query();
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return results;
    }
}
```

L'accesseur du DOA du constructeur de la classe ci-dessus est défini comme suit:

```
 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;
}
```

Lire ORMLite dans Android en ligne: https://riptutorial.com/fr/android/topic/7571/ormlite-dans-android

Chapitre 187: Otto Event Bus

Remarques

Otto est déconseillé en faveur de `RxJava` et `RxAndroid` . Ces projets permettent d'utiliser le même modèle de programmation événementiel qu'Otto, mais ils sont plus performants et offrent un meilleur contrôle du threading.

Examples

Passer un événement

Cet exemple décrit la réussite d'un événement à l'aide du bus d'événements Otto .

Pour utiliser le bus d'événements Otto dans **Android Studio,** vous devez insérer l'instruction suivante dans le fichier de graduation de vos modules:

```
dependencies {
    compile 'com.squareup:otto:1.3.8'
}
```

L'événement que nous aimerions transmettre est un simple objet Java:

```
public class DatabaseContentChangedEvent {
    public String message;

    public DatabaseContentChangedEvent(String message) {
        this.message = message;
    }
}
```

Nous avons besoin d'un bus pour envoyer des événements. C'est typiquement un singleton:

```
import com.squareup.otto.Bus;

public final class BusProvider {
    private static final Bus mBus = new Bus();

    public static Bus getInstance() {
        return mBus;
    }

    private BusProvider() {
    }
}
```

Pour envoyer un événement, nous avons uniquement besoin de notre méthode BusProvider et de sa méthode `post` . Ici, nous envoyons un événement si l'action d'un AsyncTask est terminée:

```
public abstract class ContentChangingTask extends AsyncTask<Object, Void, Void> {

    ...

    @Override
    protected void onPostExecute(Void param) {
        BusProvider.getInstance().post(
            new DatabaseContentChangedEvent("Content changed")
        );
    }
}
```

Recevoir un événement

Pour recevoir un événement, il est nécessaire d'implémenter une méthode avec le type d'événement en tant que paramètre et de l'annoter à l'aide de `@Subscribe` . De plus, vous devez enregistrer / désenregistrer l'instance de votre objet sur le `BusProvider` (voir exemple *Envoi d'un événement*):

```
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);
    }
}
```

Important: pour recevoir cet événement, une instance de la classe doit exister. Ce n'est généralement pas le cas lorsque vous souhaitez envoyer un résultat d'une activité à une autre. Donc, vérifiez votre cas d'utilisation pour le bus d'événements.

Lire Otto Event Bus en ligne: https://riptutorial.com/fr/android/topic/6068/otto-event-bus

Chapitre 188: Outils Attributs

Remarques

Android possède un espace de noms XML dédié aux outils pour pouvoir enregistrer des informations dans un fichier XML.

L'URI de l'espace de noms est:

`http://schemas.android.com/tools` et est généralement lié aux `tools:` préfixe.

Examples

Attributs de conception

Ces attributs sont utilisés lorsque la présentation est rendue dans Android Studio, mais n'ont aucun impact sur le moteur d'exécution.

En général, vous pouvez utiliser n'importe quel attribut d'infrastructure Android, simplement en utilisant les `tools:` namespace plutôt que l' `android:` namespace pour la prévisualisation de la mise en page. Vous pouvez ajouter à la fois l'attribut `android:` namespace (utilisé lors de l'exécution) et les `tools:` correspondants `tools:` attribute (qui ne remplacent que l'attribut runtime dans l'aperçu de la présentation).

Définissez simplement l'espace de noms des outils comme décrit dans la section Remarques.

Par exemple, l'attribut `text` :

```
<EditText
      tools:text="My Text"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content" />
```

Ou l'attribut `visibility` pour annuler l'affichage d'une prévisualisation:

```
<LinearLayout
        android:id="@+id/ll1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        tools:visibility="gone" />
```

Ou l'attribut `context` pour associer la disposition à une activité ou à un fragment

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context=".MainActivity" >
```

Ou l'attribut `showIn` pour voir et inclure l'aperçu de la mise en page dans une autre mise en page

```
<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" />
```

Lire Outils Attributs en ligne: https://riptutorial.com/fr/android/topic/1676/outils-attributs

Chapitre 189: Outils de rapport d'incident

Remarques

Le meilleur wiki complet est disponible ici dans github .

Examples

Tissu - Crashlytics

Fabric est une plate-forme mobile modulaire qui fournit des kits utiles que vous pouvez mélanger pour créer votre application. **Crashlytics** est un outil de reporting des **incidents** et des problèmes fourni par Fabric qui vous permet de suivre et de surveiller vos applications en détail.

Comment configurer Fabric-Crashlytics

Etape 1: Changez votre `build.gradle` :

Ajoutez le repo plugin et le plugin gradle:

```
 buildscript {
  repositories {
    maven { url 'https://maven.fabric.io/public' }
  }

  dependencies {
    // The Fabric Gradle plugin uses an open ended version to react
    // quickly to Android tooling updates
    classpath 'io.fabric.tools:gradle:1.+'
  }
}
```

Appliquez le plugin:

```
apply plugin: 'com.android.application'
//Put Fabric plugin after Android plugin
apply plugin: 'io.fabric'
```

Ajoutez le dépôt Fabric:

```
repositories {
  maven { url 'https://maven.fabric.io/public' }
}
```

Ajoutez le kit Crashlyrics:

```
dependencies {

  compile('com.crashlytics.sdk.android:crashlytics:2.6.6@aar') {
    transitive = true;
  }
}
```

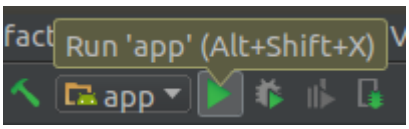
Étape 2: Ajouter votre **clé API** et l'autorisation **INTERNET** dans `AndroidManifest.xml`

```
<?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>
```

Étape 3: Initiez le kit à l'exécution dans votre code, par exemple:

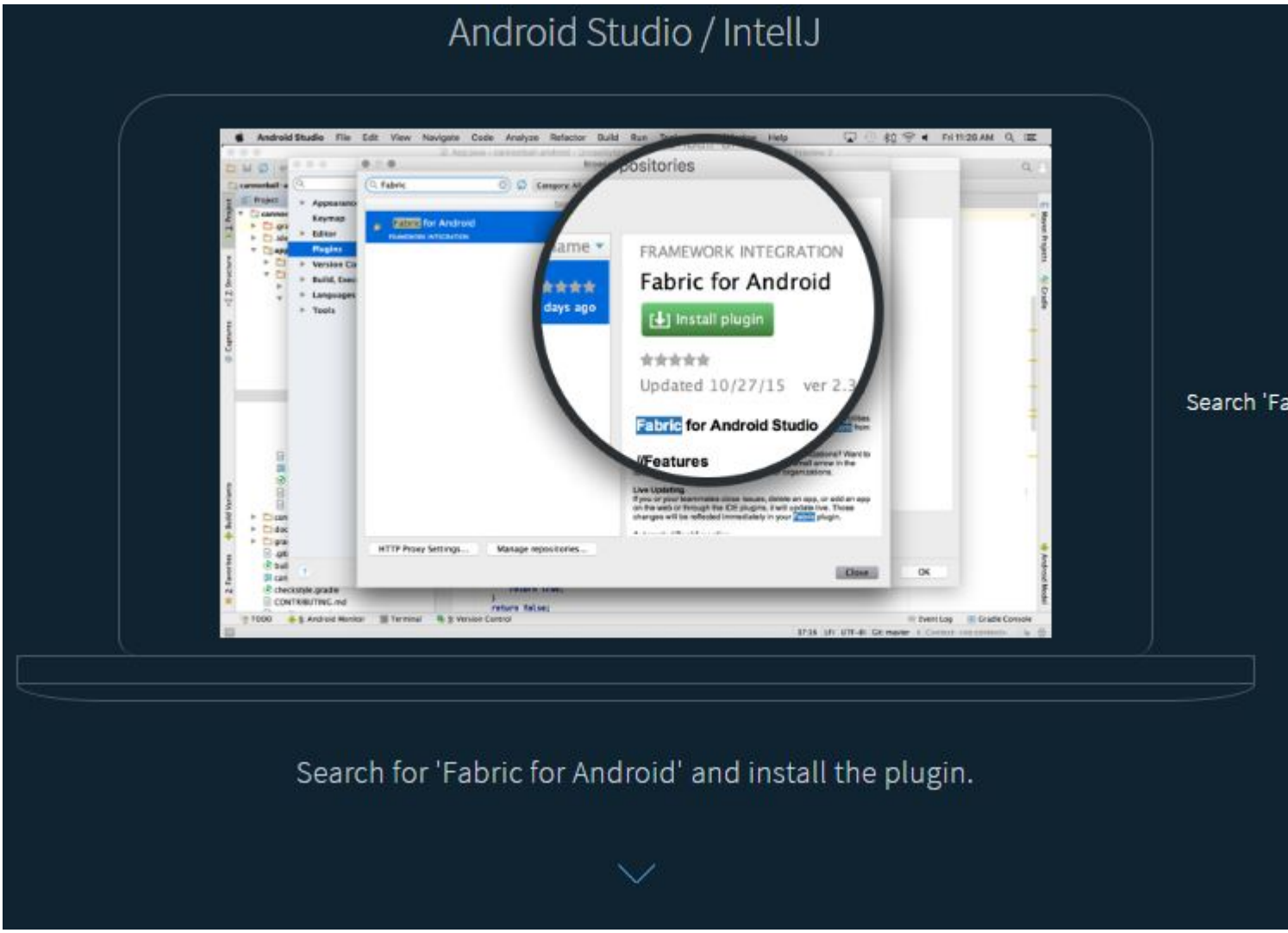
```
public class MainActivity extends ActionBarActivity {

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

      //Init the KIT
      Fabric.with(this, new Crashlytics());

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

Étape 4: Construire le projet. Pour construire et exécuter:



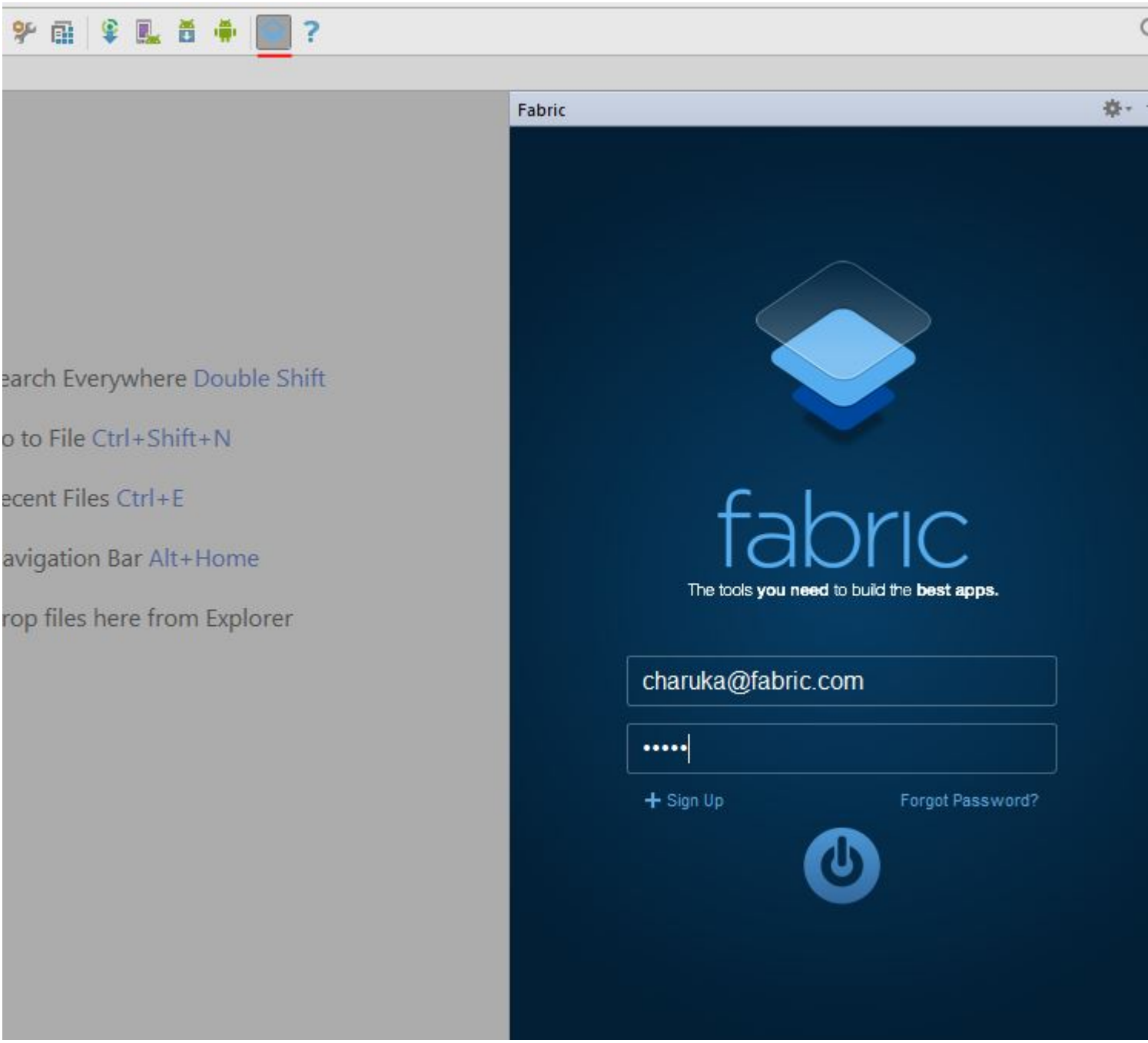
Utilisation du plug-in Fabric IDE

Les kits peuvent être installés à l'aide du plug-in Fabric IDE pour Android Studio ou IntelliJ en suivant ce lien.



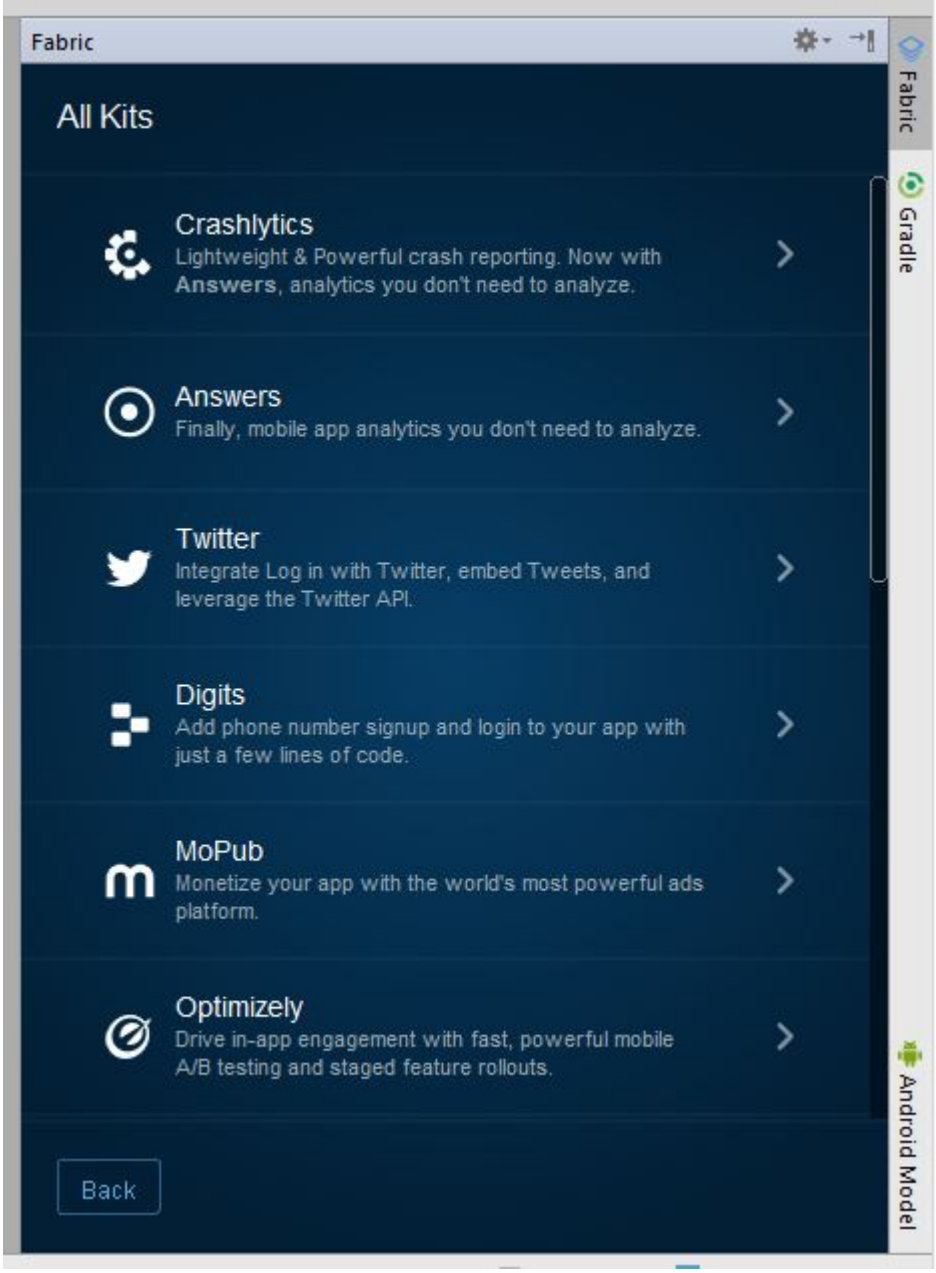
Après avoir installé le plug-in, **redémarrez** Android Studio et **connectez-** vous avec votre compte à l'aide d' **Android Studio** .

(touche courte> `CTRL + L`)

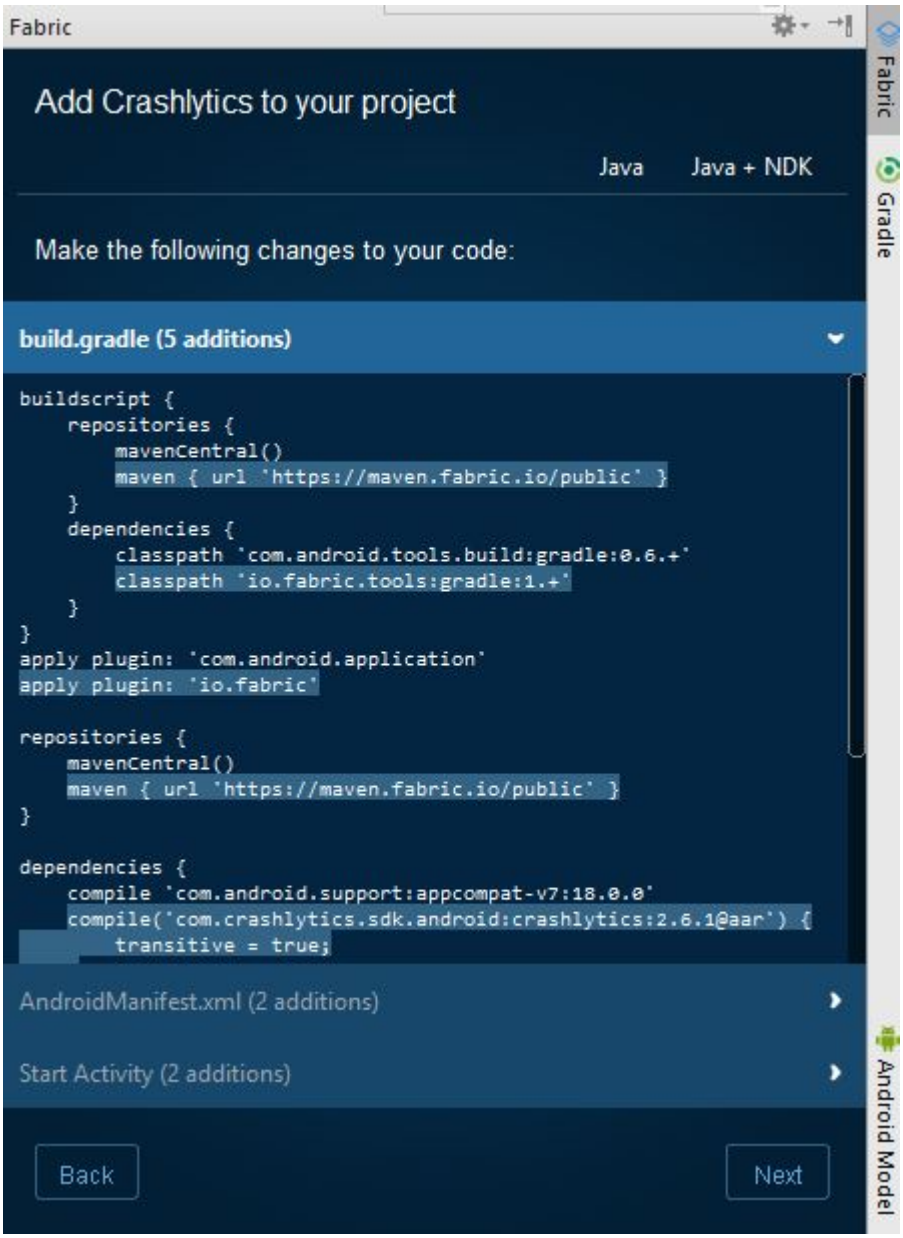


Ensuite, il montrera les projets que vous avez / le projet que vous avez ouvert, sélectionnez celui dont vous avez besoin et cliquez sur suivant .. suivant.

Sélectionnez le kit que vous souhaitez ajouter, par exemple **Crashlytics** :



Puis appuyez sur `Install` . Vous n'avez pas besoin d'ajouter manuellement cette fois comme ci-dessus **gradle plug** - **in,** au lieu qu'il construira pour vous.



Terminé!

Rapport de collision avec ACRA

Etape 1: Ajoutez la dépendance du dernier ACAR AAR à votre application (build.gradle).

Étape 2: Dans votre classe d'application (la classe qui étend Application, sinon créez-la) Ajoutez une annotation `@ReportsCrashes` et remplacez la méthode `attachBaseContext()` .

Étape 3: Initialiser la classe ACRA dans votre classe d'application

```
@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);
            }
    }
```

Où AUTH_USERNAME et AUTH_PASSWORD sont les informations d'identification de vos backends souhaités.

Étape 4: Définir la classe d'application dans AndroidManifest.xml

```
<application
       android:name=".MyApplication">
       <service></service>
       <activity></activity>
       <receiver></receiver>
</application>
```

Étape 5: Assurez-vous de disposer de `internet` autorisation `internet` pour recevoir le rapport de l'application en panne

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

Si vous souhaitez envoyer le rapport silencieux au serveur, utilisez simplement la méthode ci-dessous pour y parvenir.

```
ACRA.getErrorReporter().handleSilentException(e);
```

Force un test de crash avec le tissu

Ajoutez un bouton sur lequel vous pouvez appuyer pour déclencher un crash. Collez ce code dans votre mise en page où vous souhaitez que le bouton apparaisse.

```
<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" />
```

Lancer une exceptionExécution

```
public void forceCrash(View view) {
    throw new RuntimeException("This is a crash");
}
```

Exécutez votre application et appuyez sur le nouveau bouton pour provoquer un plantage. Dans une minute ou deux, vous devriez pouvoir voir le crash sur votre tableau de bord Crashlytics et vous recevrez un mail.

Capture se bloque avec Sherlock

Sherlock capture tous vos accidents et les signale en tant que notification. Lorsque vous appuyez sur la notification, une activité s'ouvre avec tous les détails de la panne, ainsi que des informations sur le périphérique et l'application.

Comment intégrer Sherlock avec votre application?

Il vous suffit d'ajouter Sherlock comme une dépendance progressive dans votre projet.

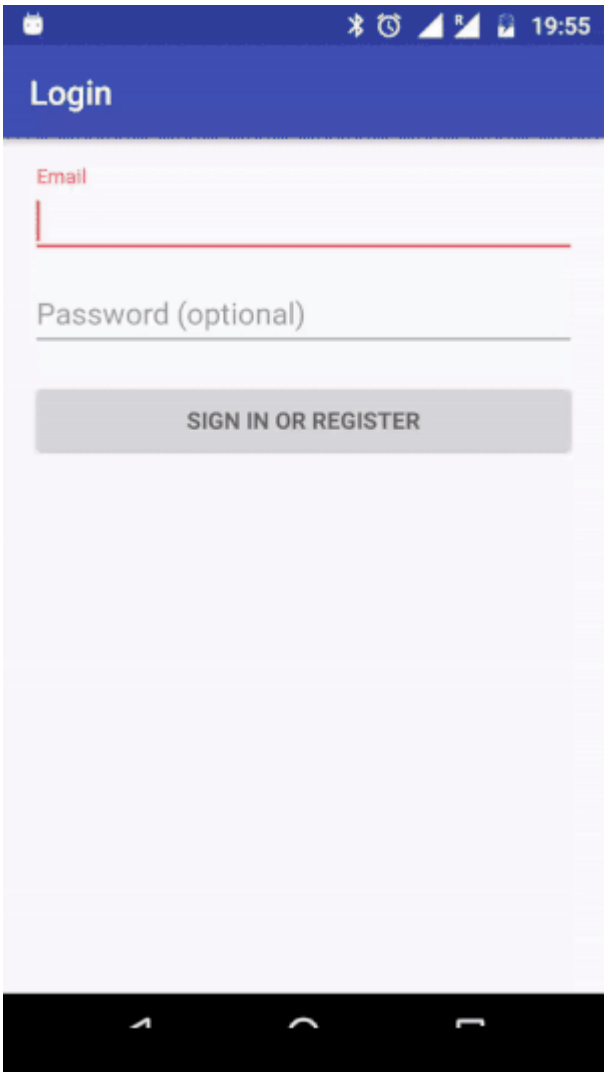
```
dependencies {
    compile('com.github.ajitsing:sherlock:1.0.1@aar') {
        transitive = true
    }
}
```

Après avoir synchronisé votre studio Android, initialisez Sherlock dans votre classe Application.

```
package com.singhajit.login;

import android.app.Application;

import com.singhajit.sherlock.core.Sherlock;

public class SampleApp extends Application {
  @Override
  public void onCreate() {
    super.onCreate();
    Sherlock.init(this);
  }
}
```

C'est tout ce que vous devez faire. Sherlock fait beaucoup plus que simplement signaler un crash. Pour vérifier toutes ses fonctionnalités, consultez cet article .

Démo



Lire Outils de rapport d'incident en ligne: https://riptutorial.com/fr/android/topic/3871/outils-de-rapport-d-incident

Chapitre 190: Pagination dans RecyclerView

Introduction

La pagination est un problème courant pour de nombreuses applications mobiles qui doivent gérer des listes de données. La plupart des applications mobiles commencent maintenant à utiliser le modèle de "page sans fin", où le défilement se charge automatiquement dans un nouveau contenu. CWAC Endless Adapter, il est très facile d'utiliser ce modèle dans les applications Android

Examples

MainActivity.java

```
import android.os.Bundle;
import android.os.Handler;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.support.v7.widget.Toolbar;
import android.util.Log;
import android.widget.TextView;

import com.android.volley.Request;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.VolleyLog;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class MainActivity extends AppCompatActivity {

    private static final String TAG = "MainActivity";
    private Toolbar toolbar;

    private TextView tvEmptyView;
    private RecyclerView mRecyclerView;
    private DataAdapter mAdapter;
    private LinearLayoutManager mLayoutManager;
    private int mStart=0,mEnd=20;
    private List<Student> studentList;
    private List<Student> mTempCheck;
    public static int pageNumber;
    public int total_size=0;
```

```
protected Handler handler;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    pageNumber = 1;
    toolbar = (Toolbar) findViewById(R.id.toolbar);
    tvEmptyView = (TextView) findViewById(R.id.empty_view);
    mRecyclerView = (RecyclerView) findViewById(R.id.my_recycler_view);
    studentList = new ArrayList<>();
    mTempCheck=new ArrayList<>();
    handler = new Handler();
    if (toolbar != null) {
        setSupportActionBar(toolbar);
        getSupportActionBar().setTitle("Android Students");

    }

    mRecyclerView.setHasFixedSize(true);
    mLayoutManager = new LinearLayoutManager(this);
    mRecyclerView.setLayoutManager(mLayoutManager);
    mAdapter = new DataAdapter(studentList, mRecyclerView);
    mRecyclerView.setAdapter(mAdapter);
    GetGroupData("" + mStart, "" + mEnd);
    mAdapter.setOnLoadMoreListener(new OnLoadMoreListener() {
        @Override
        public void onLoadMore() {
            if( mTempCheck.size()> 0) {
                studentList.add(null);
                mAdapter.notifyItemInserted(studentList.size() - 1);
                int start = pageNumber * 20;
                start = start + 1;
                ++ pageNumber;
                mTempCheck.clear();
                GetData("" + start,""+ mEnd);
            }
        }
    });
}

public void GetData(final String LimitStart, final String LimitEnd) {
    Map<String, String> params = new HashMap<>();
    params.put("LimitStart", LimitStart);
    params.put("Limit", LimitEnd);
    Custom_Volly_Request jsonObjReq = new Custom_Volly_Request(Request.Method.POST,
            "Your php file link", params,
            new Response.Listener<JSONObject>() {
                @Override
                public void onResponse(JSONObject response) {
                    Log.d("ResponseSuccess",response.toString());
                    // handle the data from the servoce
                }
            }, new Response.ErrorListener() {

        @Override
        public void onErrorResponse(VolleyError error) {
            VolleyLog.d("ResponseErrorVolly: " + error.getMessage());

        }));
```

```
    }
    // load initial data
    private void loadData(int start,int end,boolean notifyadapter) {
        for (int i = start; i <= end; i++) {
            studentList.add(new Student("Student " + i, "androidstudent" + i + "@gmail.com"));
            if(notifyadapter)
                mAdapter.notifyItemInserted(studentList.size());
        }
    }
}
```

OnLoadMoreListener.java

interface publique 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);
        }
    }

}
```

Lire Pagination dans RecyclerView en ligne:
https://riptutorial.com/fr/android/topic/9243/pagination-dans-recyclerview

Chapitre 191: Pain grillé

Introduction

Un toast fournit des commentaires simples sur une opération dans une petite fenêtre contextuelle et disparaît automatiquement après un délai d'attente. Il ne remplit que la quantité d'espace requise pour le message et l'activité en cours reste visible et interactive.

Syntaxe

- Toast makeText (Contexte contextuel, texte CharSequence, durée int)
- Toast makeText (Contexte contextuel, int resId, int duration)
- void setGravity (int gravity, int xOffset, int yOffset)
- annuler le spectacle ()

Paramètres

Paramètre	Détails
le contexte	Le contexte d'affichage de votre Toast dans. `this` est couramment utilisé dans une activité et `getActivity()` est couramment utilisé dans un fragment.
texte	Un CharSequence qui spécifie quel texte sera affiché dans le Toast. Tout objet qui implémente CharSequence peut être utilisé, y compris une chaîne
resId	Un ID de ressource pouvant être utilisé pour fournir une chaîne de ressources à afficher dans le Toast
durée	Drapeau entier représentant la durée d'affichage du Toast. Les options sont `Toast.LENGTH_SHORT` et `Toast.LENGTH_LONG`
la gravité	Entier spécifiant la position ou "gravité" du pain grillé. Voir les options ici
xOffset	Spécifie le décalage horizontal pour la position Toast
yOffset	Spécifie le décalage vertical pour la position Toast

Remarques

Un toast fournit des commentaires simples sur une opération dans une petite fenêtre contextuelle. Il ne remplit que la quantité d'espace requise pour le message et l'activité en cours reste visible et interactive.

SnackBar est une alternative plus récente à Toast. SnackBar offre un style visuel mis à jour et

permet à l'utilisateur de supprimer le message ou de prendre des mesures supplémentaires. Voir la documentation SnackBar pour plus de détails.

Documentation officielle:

https://developer.android.com/reference/android/widget/Toast.html

Examples

Définir la position d'un pain grillé

Une notification de toast standard apparaît en bas de l'écran, alignée au centre horizontal. Vous pouvez changer cette position avec `setGravity(int, int, int)` . Cela accepte trois paramètres: une constante de gravité, un décalage de la position x et un décalage de la position y.

Par exemple, si vous décidez que le toast doit apparaître dans le coin supérieur gauche, vous pouvez définir la gravité comme ceci:

```
toast.setGravity(Gravity.TOP|Gravity.LEFT, 0, 0);
```

Afficher un message de pain grillé

Dans Android, un Toast est un élément d'interface utilisateur simple qui peut être utilisé pour donner un retour contextuel à un utilisateur.

Pour afficher un simple message Toast, nous pouvons procéder comme suit.

```
// 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();
```

Ou, pour afficher un toast en ligne, sans conserver l'objet Toast, vous pouvez:

```
Toast.makeText(context, "Ding! Your Toast is ready.", Toast.LENGTH_SHORT).show();
```

IMPORTANT: assurez-vous que la méthode `show()` est appelée à partir du thread d'interface utilisateur. Si vous essayez d'afficher un `Toast` partir d'un thread différent, vous pouvez par exemple utiliser la méthode `runOnUiThread` d'une `Activity` .

Ne pas le faire, c'est-à-dire essayer de modifier l'interface utilisateur en créant un Toast, lancera

une `RuntimeException` qui ressemblera à ceci:

```
java.lang.RuntimeException: Can't create handler inside thread that has not called
Looper.prepare()
```

La manière la plus simple de gérer cette exception est d'utiliser simplement runOnUiThread: la syntaxe est indiquée ci-dessous.

```
   runOnUiThread(new Runnable() {
        @Override
        public void run() {
            // Your code here
        }
    });
```

Création d'un toast personnalisé

Si vous ne souhaitez pas utiliser la vue Toast par défaut, vous pouvez fournir la vôtre à l'aide de la `setView(View)` sur un objet `Toast` .

Tout d'abord, créez la disposition XML que vous souhaitez utiliser dans votre Toast.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/toast_layout_root"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="8dp"
    android:background="#111">

    <TextView android:id="@+id/title"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="#FFF"/>

    <TextView android:id="@+id/description"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="#FFF"/>

</LinearLayout>
```

Ensuite, lors de la création de votre Toast, gonflez votre vue personnalisée à partir de XML et appelez `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();
```

Fil de manière sécurisée pour afficher 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();
                }
            }
        });
    }
```

N'oubliez pas d'ajouter `MainApplication` au `manifest` .

Appelez-le maintenant depuis n'importe quel thread pour afficher un message de toast.

```
MainApplication.getApp().showToast("Some message", Toast.LENGTH_LONG);
```

Afficher le message Toast ci-dessus

Par défaut, Android affichera les messages Toast en bas de l'écran même si le clavier est affiché. Cela affichera un message Toast juste au-dessus du clavier.

```
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();
}
```

Un moyen sûr d'afficher un message Toast (pour AsyncTask)

Si vous ne souhaitez pas étendre l'application et garder vos messages de toast sécurisés, assurez-vous de les afficher dans la section post-exécution de vos AsyncTasks.

```
public class MyAsyncTask extends AsyncTask <Void, Void, Void> {

    @Override
    protected Void doInBackground(Void... params) {
        // Do your background work here
    }

    @Override
    protected void onPostExecute(Void aVoid) {
        // Show toast messages here
        Toast.makeText(context, "Ding! Your Toast is ready.",   Toast.LENGTH_SHORT).show();
    }

}
```

Lire Pain grillé en ligne: https://riptutorial.com/fr/android/topic/1741/pain-grille

Chapitre 192: Parcelable

Introduction

Parcelable est une interface spécifique à Android où vous implémentez vous-même la sérialisation. Il a été créé pour être beaucoup plus efficace que Serializable, et pour contourner certains problèmes avec le schéma de sérialisation Java par défaut.

Remarques

Il est important de garder à l'esprit que l'ordre dans lequel vous écrivez des champs dans une parcelle DOIT ÊTRE LE MÊME ORDRE que vous les lisez dans la parcelle lors de la construction de votre objet personnalisé.

L'interface parcellaire a une limite de taille stricte de 1 Mo. Cela signifie que tout objet ou combinaison d'objets que vous mettez dans une parcelle occupant plus de 1 Mo d'espace sera corrompu de l'autre côté. Cela peut être difficile à découvrir, alors gardez à l'esprit le type d'objets que vous envisagez de rendre parcellable. S'ils disposent de grands arbres de dépendance, envisagez un autre moyen de transmettre des données.

Examples

Rendre un objet personnalisé Parcelable.

```
/**
 * Created by Alex Sullivan on 7/21/16.
 */
public class Foo implements Parcelable
{
    private final int myFirstVariable;
    private final String mySecondVariable;
    private final long myThirdVariable;

    public Foo(int myFirstVariable, String mySecondVariable, long myThirdVariable)
    {
        this.myFirstVariable = myFirstVariable;
        this.mySecondVariable = mySecondVariable;
        this.myThirdVariable = myThirdVariable;
    }

    // Note that you MUST read values from the parcel IN THE SAME ORDER that
    // values were WRITTEN to the parcel! This method is our own custom method
    // to instantiate our object from a Parcel. It is used in the Parcelable.Creator variable
we declare below.
    public Foo(Parcel in)
    {
        this.myFirstVariable = in.readInt();
        this.mySecondVariable = in.readString();
        this.myThirdVariable = in.readLong();
    }
```

```
    // The describe contents method can normally return 0. It's used when
    // the parceled object includes a file descriptor.
    @Override
    public int describeContents()
    {
        return 0;
    }

    @Override
    public void writeToParcel(Parcel dest, int flags)
    {
        dest.writeInt(myFirstVariable);
        dest.writeString(mySecondVariable);
        dest.writeLong(myThirdVariable);
    }

    // Note that this seemingly random field IS NOT OPTIONAL. The system will
    // look for this variable using reflection in order to instantiate your
    // parceled object when read from an Intent.
    public static final Parcelable.Creator<Foo> CREATOR = new Parcelable.Creator<Foo>()
    {
        // This method is used to actually instantiate our custom object
        // from the Parcel. Convention dictates we make a new constructor that
        // takes the parcel in as its only argument.
        public Foo createFromParcel(Parcel in)
        {
            return new Foo(in);
        }

        // This method is used to make an array of your custom object.
        // Declaring a new array with the provided size is usually enough.
        public Foo[] newArray(int size)
        {
            return new Foo[size];
        }
    };
}
```

Objet parcellable contenant un autre objet parcellable

Un exemple de classe contenant une classe parcellable à l'intérieur:

```
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;
    }
}
```

Le propriétaire est juste une classe parcelable normale.

Utiliser Enums avec 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
}
```

Ceci est préférable à (par exemple) en utilisant un ordinal, car l'insertion de nouvelles valeurs dans votre enum n'affectera pas les valeurs stockées précédemment

Lire Parcelable en ligne: https://riptutorial.com/fr/android/topic/1849/parcelable

Chapitre 193: Peindre

Introduction

Une peinture est l'un des quatre objets nécessaires pour dessiner, avec un canevas (contient des appels de dessin), un bitmap (contient les pixels) et une primitive de dessin (rect, chemin, bitmap ...)

Examples

Créer une peinture

Vous pouvez créer une nouvelle peinture avec l'un de ces 3 constructeurs:

- `new Paint()` Créer avec les paramètres par défaut
- `new Paint(int flags)` Créer avec des drapeaux
- `new Paint(Paint from)` Copier les paramètres d'une autre peinture

Il est généralement suggéré de ne jamais créer un objet paint, ou tout autre objet dans onDraw (), car cela peut entraîner des problèmes de performances. (Android Studio vous avertira probablement) Au lieu de cela, rendez-le global et initialisez-le dans votre constructeur de classes comme suit:

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

Configuration de Paint pour le texte

Paramètres de dessin de texte

- `setTypeface(Typeface typeface)` Définit la police. Voir police de caractères
- `setTextSize(int size)` Définit la taille de la police, en pixels.
- `setColor(int color)` Définit la couleur du dessin de peinture, y compris la couleur du texte.

Vous pouvez également utiliser `setARGB(int a, int r, int g, int b` et `setAlpha(int alpha)`

- `setLetterSpacing(float size)` Définit l'espacement entre les caractères, dans ems. La valeur par défaut est 0, une valeur négative resserre le texte, tandis qu'une valeur positive la développe.
- `setTextAlign(Paint.Align align)` Définit l'alignement du texte par rapport à son origine. `Paint.Align.LEFT` le dessinera à droite de l'origine, `RIGHT` le dessinera à gauche et `CENTER` le dessinera centré sur l'origine (horizontalement)
- `setTextSkewX(float skewX)` Cela pourrait être considéré comme un faux italique. SkewX représente le décalage horizontal du fond du texte. (utilisez -0.25 pour l'italique)
- `setStyle(Paint.Style style)` Remplissez texte `FILL` , le texte Stroke `STROKE` , ou les deux `FILL_AND_STROKE`

Notez que vous pouvez utiliser `TypedValue.applyDimension(TypedValue.COMPLEX_UNIT_SP, size, getResources().getDisplayMetrics())` pour convertir de SP ou DP en pixels.

Texte de mesure

- `float width = paint.measureText(String text)` Mesurer la largeur du texte
- `float height = paint.ascent()` Mesure la hauteur du texte
- `paint.getTextBounds(String text, int start, int end, Rect bounds` Stocke les dimensions du texte. Vous avez alloué le Rect, il ne peut pas être nul:

```
String text = "Hello world!";
Rect bounds = new Rect();
paint.getTextBounds(text, 0, text.length(), bounds);
```

Il existe d'autres méthodes de mesure, mais ces trois méthodes devraient convenir à la plupart des objectifs.

Mise en place de la peinture pour dessiner des formes

- `setStyle(Paint.Style style)` forme remplie `FILL` , la forme de maladies `STROKE` , ou les deux `FILL_AND_STROKE`
- `setColor(int color)` Définit la couleur du dessin de peinture. Vous pouvez également utiliser `setARGB(int a, int r, int g, int b` et `setAlpha(int alpha)`
- `setStrokeCap(Paint.Cap cap)` fixer des plafonds de ligne, soit `ROUND` , `SQUARE` , ou `BUTT` (none) Voir ce .
- `setStrokeJoin(Paint.Join join)` Définit les jointures de ligne, soit `MITER` (pointu), `ROUND` ou `BEVEL` . Voir ça
- `setStrokeMiter(float miter)` Définit la limite de jointure de l'onglet. Cela peut empêcher la jointure d'onglet de se poursuivre indéfiniment, en la transformant en jointure en biseau après x pixels. Voir ça
- `setStrokeWidth(float width)` Définit la largeur du trait. `0` va dessiner en mode hairline, indépendamment de la matrice de canevas. (toujours 1 pixel)

Réglage des drapeaux

Vous pouvez définir les indicateurs suivants dans le constructeur ou avec `setFlags(int flags)`

- `Paint.ANTI_ALIAS_FLAG` Active l'anticrénelage, lisse le dessin.
- `Paint.DITHER_FLAG` Active le dithering. Si la précision des couleurs est supérieure à celle du périphérique, cela se produira .
- `Paint.EMBEDDED_BITMAP_TEXT_FLAG` Permet l'utilisation de polices bitmap.
- `Paint.FAKE_BOLD_TEXT_FLAG` va dessiner un texte avec un faux effet, peut être utilisé au lieu d'utiliser une police de caractères en gras. Certaines polices ont un style audacieux, les faux audibles
- `Paint.FILTER_BITMAP_FLAG` Affecte l'échantillonnage des bitmaps lors de la transformation.
- `Paint.HINTING_OFF` , `Paint.HINTING_ON` Active / `Paint.HINTING_ON` police, voir ceci
- `Paint.LINEAR_TEXT_FLAG` Désactive la mise à l'échelle des polices, les opérations de dessin sont mises à l'échelle à la place
- `Paint.SUBPIXEL_TEXT_FLAG` texte sera calculé en utilisant la précision du sous-pixel.
- `Paint.STRIKE_THRU_TEXT_FLAG` texte dessiné sera rayé
- `Paint.UNDERLINE_TEXT_FLAG` texte dessiné sera souligné

Vous pouvez ajouter un indicateur et supprimer des indicateurs comme ceci:

```
Paint paint = new Paint();
paint.setFlags(paint.getFlags() | Paint.FLAG);   // Add flag
paint.setFlags(paint.getFlags() & ~Paint.FLAG);  // Remove flag
```

Essayer de supprimer un drapeau qui n'est pas là ou ajouter un drapeau qui est déjà là ne changera rien. Notez également que la plupart des indicateurs peuvent également être définis avec `set<Flag>(boolean enabled)` , par exemple `setAntialias(true)` .

Vous pouvez utiliser `paint.reset()` pour réinitialiser la peinture à ses paramètres par défaut. Le seul indicateur par défaut est `EMBEDDED_BITMAP_TEXT_FLAG` . Il sera défini même si vous utilisez une `new Paint(0)` , vous aurez

Lire Peindre en ligne: https://riptutorial.com/fr/android/topic/9141/peindre

Chapitre 194: Picasso

Introduction

Picasso est une bibliothèque d'images pour Android. Il est créé et entretenu par Square . Il simplifie le processus d'affichage des images à partir d'emplacements externes. La bibliothèque gère chaque étape du processus, de la requête HTTP initiale à la mise en cache de l'image. Dans de nombreux cas, seules quelques lignes de code sont nécessaires pour implémenter cette bibliothèque soignée.

Remarques

Picasso est une puissante bibliothèque de téléchargement et de mise en cache d'images pour Android.
Suivez cet exemple pour ajouter la bibliothèque à votre projet.

Sites Internet:

- La source
- Doc
- Changer le journal

Examples

Ajout de la bibliothèque Picasso à votre projet Android

De la documentation officielle :

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>
```

Placeholder et gestion des erreurs

Picasso prend en charge les espaces réservés pour le téléchargement et les erreurs en tant que fonctionnalités facultatives. Il fournit également des rappels pour gérer le résultat du téléchargement.

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

Une demande sera réessayée trois fois avant que l'espace réservé aux erreurs ne soit affiché.

Redimensionnement et rotation

```
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);
```

Avatars circulaires avec Picasso

Voici un exemple de classe de transformation de cercle Picasso basée sur l'original , avec l'ajout d'une bordure mince, et comprend également des fonctionnalités pour un séparateur facultatif à empiler:

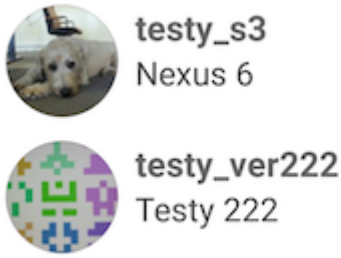
```
import android.graphics.Bitmap;
import android.graphics.BitmapShader;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.Paint.Style;

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";
    }
}
```

Voici comment l'utiliser lors du chargement d'une image (en supposant `this` s'agit d'un contexte d'activité et que l' `url` est une chaîne avec l'URL de l'image à charger):

```
ImageView ivAvatar = (ImageView) itemView.findViewById(R.id.avatar);
Picasso.with(this).load(url)
    .fit()
    .transform(new CircleTransform())
    .into(ivAvatar);
```

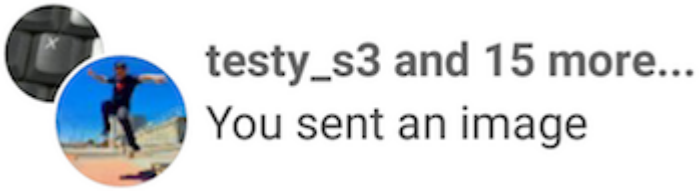
Résultat:



Pour utiliser avec le séparateur, attribuez la valeur `true` au constructeur de l'image supérieure:

```
ImageView ivAvatar = (ImageView) itemView.findViewById(R.id.avatar);
Picasso.with(this).load(url)
    .fit()
    .transform(new CircleTransform(true))
    .into(ivAvatar);
```

Résultat (deux ImageViews dans un FrameLayout):



Désactiver le cache dans Picasso

```
Picasso.with(context)
.load(uri)
.networkPolicy(NetworkPolicy.NO_CACHE)
.memoryPolicy(MemoryPolicy.NO_CACHE)
.placeholder(R.drawable.placeholder)
.into(imageView);
```

Chargement de l'image à partir d'un stockage externe

```
String filename = "image.png";
String imagePath = getExternalFilesDir() + "/" + filename;

Picasso.with(context)
    .load(new File(imagePath))
    .into(imageView);
```

Téléchargement de l'image en tant que bitmap à l'aide de Picasso

Si vous souhaitez télécharger l'image sous forme de `Bitmap` aide de `Picasso` code suivant vous aidera à:

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

Annulation de demandes d'images à l'aide de Picasso

Dans certains cas, nous devons annuler une demande de téléchargement d'image dans Picasso avant la fin du téléchargement.

Cela peut se produire pour diverses raisons, par exemple si la vue parente est passée à une autre vue avant que le téléchargement de l'image puisse être terminé.

Dans ce cas, vous pouvez annuler la demande de téléchargement d'image à l'aide de la méthode `cancelRequest()` :

```
ImageView imageView;

//......

Picasso.with(imageView.getContext()).cancelRequest(imageView);
```

Utiliser Picasso comme ImageGetter pour Html.fromHtml

Utiliser Picasso comme ImageGetter pour Html.fromHtml

```
public class PicassoImageGetter implements Html.ImageGetter {

private TextView textView;

private Picasso picasso;

public PicassoImageGetter(@NonNull Picasso picasso, @NonNull TextView textView) {
    this.picasso = picasso;
    this.textView = textView;
}
```

```
@Override
public Drawable getDrawable(String source) {
    Log.d(PicassoImageGetter.class.getName(), "Start loading url " + source);

    BitmapDrawablePlaceHolder drawable = new BitmapDrawablePlaceHolder();

    picasso
            .load(source)
            .error(R.drawable.connection_error)
            .into(drawable);

    return drawable;
}

private class BitmapDrawablePlaceHolder extends BitmapDrawable implements Target {

    protected Drawable drawable;

    @Override
    public void draw(final Canvas canvas) {
        if (drawable != null) {
            checkBounds();
            drawable.draw(canvas);
        }
    }

    public void setDrawable(@Nullable Drawable drawable) {
        if (drawable != null) {
            this.drawable = drawable;
            checkBounds();
        }
    }

    private void checkBounds() {
        float defaultProportion = (float) drawable.getIntrinsicWidth() / (float)
drawable.getIntrinsicHeight();
        int width = Math.min(textView.getWidth(), drawable.getIntrinsicWidth());
        int height = (int) ((float) width / defaultProportion);

        if (getBounds().right != textView.getWidth() || getBounds().bottom != height) {

            setBounds(0, 0, textView.getWidth(), height); //set to full width

            int halfOfPlaceHolderWidth = (int) ((float) getBounds().right / 2f);
            int halfOfImageWidth = (int) ((float) width / 2f);

            drawable.setBounds(
                    halfOfPlaceHolderWidth - halfOfImageWidth, //centering an image
                    0,
                    halfOfPlaceHolderWidth + halfOfImageWidth,
                    height);

            textView.setText(textView.getText()); //refresh text
        }
    }

    //------------------------------------------------------------------//

    @Override
    public void onBitmapLoaded(Bitmap bitmap, Picasso.LoadedFrom from) {
```

```
        setDrawable(new BitmapDrawable(Application.getContext().getResources(), bitmap));
    }

    @Override
    public void onBitmapFailed(Drawable errorDrawable) {
        setDrawable(errorDrawable);
    }

    @Override
    public void onPrepareLoad(Drawable placeHolderDrawable) {
        setDrawable(placeHolderDrawable);
    }

    //------------------------------------------------------------------//

}
}
```

L'utilisation est simple:

```
Html.fromHtml(textToParse, new PicassoImageGetter(picasso, textViewTarget), null);
```

Essayez d'abord le cache disque hors connexion, puis connectez-vous et récupérez l'image

ajoutez d'abord l'OkHttp au fichier de construction graduel du module d'application

```
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'
```

Ensuite, créez une application qui étend la classe

```
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);

    }
}
```

ajoutez-le au fichier manifeste comme suit:

```
<application
        android:name=".Global"
        .. >

</application>
```

Utilisation normale

```
Picasso.with(getActivity())
.load(imageUrl)
.networkPolicy(NetworkPolicy.OFFLINE)
.into(imageView, new Callback() {
    @Override
    public void onSuccess() {
        //Offline Cache hit
    }

    @Override
    public void onError() {
        //Try again online if cache failed
        Picasso.with(getActivity())
                .load(imageUrl)
                .error(R.drawable.header)
                .into(imageView, new Callback() {
            @Override
            public void onSuccess() {
                 //Online download
            }

            @Override
            public void onError() {
                Log.v("Picasso","Could not fetch image");
            }
        });
    }
});
```

Lien vers la réponse originale

Lire Picasso en ligne: https://riptutorial.com/fr/android/topic/2172/picasso

Chapitre 195: Ping ICMP

Introduction

La requête ICMP Ping peut être effectuée dans Android en créant un nouveau processus pour exécuter la requête ping. Le résultat de la demande peut être évalué à la fin de la requête ping depuis son processus.

Examples

Effectue un seul ping

Cet exemple tente une seule requête Ping. La commande ping à l'intérieur de l' `runtime.exec` méthode `runtime.exec` peut être modifiée pour n'importe quelle commande ping valide que vous pourriez effectuer vous-même dans la ligne de commande.

```
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();
}
```

Lire Ping ICMP en ligne: https://riptutorial.com/fr/android/topic/9434/ping-icmp

Chapitre 196: Piste audio

Examples

Générer le ton d'une fréquence spécifique

Pour jouer un son avec une sonorité spécifique, il faut d'abord créer un son sinusoïdal. Ceci se fait de la manière suivante.

```
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
}
```

Maintenant, nous devons configurer AudioTrack pour jouer en fonction du tampon généré. Cela se fait de la manière suivante

```
AudioTrack audioTrack = new AudioTrack(AudioManager.STREAM_MUSIC,
                    sampleRate, AudioFormat.CHANNEL_OUT_MONO,
                    AudioFormat.ENCODING_PCM_16BIT, buffer.length,
                    AudioTrack.MODE_STATIC);
```

Ecrire le tampon généré et jouer la piste

```
audioTrack.write(buffer, 0, buffer.length);
audioTrack.play();
```

J'espère que cela t'aides :)

Lire Piste audio en ligne: https://riptutorial.com/fr/android/topic/9155/piste-audio

Chapitre 197: Planification du travail

Remarques

Attention à exécuter beaucoup de code ou à faire un gros travail dans votre `JobService` , par exemple dans `onStartJob()` . Le code s'exécutera sur le thread **principal / UI** et peut donc entraîner une interface utilisateur bloquée, ne répondant plus à l'application ou même un plantage de votre application!

À cause de cela, vous devez décharger le travail, par exemple en utilisant un `Thread` ou un `AsyncTask` .

Examples

Utilisation de base

Créer un nouveau JobService

Cela se fait en étendant la classe `JobService` et en implémentant / `onStartJob()` méthodes requises `onStartJob()` et `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;
    }
}
```

Ajoutez le nouveau JobService à votre AndroidManifest.xml

L'étape suivante est *obligatoire* , sinon vous ne pourrez pas exécuter votre travail:

Déclarez votre classe `MyJobService` tant que nouvel élément `<service>` entre `<application> </application>` dans votre *fichier 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>
```

Configurer et exécuter le travail

Après avoir implémenté un nouveau JobService et l'a ajouté à votre *AndroidManifest.xml* , vous pouvez continuer les étapes finales.

- `onButtonClick_startJob()` prépare et exécute un travail périodique. Outre les tâches périodiques, `JobInfo.Builder` permet de spécifier de nombreux autres paramètres et contraintes. Par exemple, vous pouvez définir qu'un *chargeur branché* ou une *connexion réseau* est nécessaire pour exécuter le travail.
- `onButtonClick_stopJob()` annule tous les travaux en cours d'exécution

```
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
    }
}
```

Après avoir appelé `onButtonClick_startJob()` , le travail sera exécuté approximativement à intervalles de 10 secondes, même lorsque l'application est en *pause* (le bouton d'accueil de l'utilisateur et l'application ne sont plus visibles).

Au lieu d'annuler tous les travaux en cours d'exécution dans `onButtonClick_stopJob()` , vous pouvez également appeler `jobScheduler.cancel()` pour annuler un travail spécifique en fonction de son ID de travail.

Lire Planification du travail en ligne: https://riptutorial.com/fr/android/topic/6907/planification-du-travail

Chapitre 198: Polices Personnalisées

Examples

Mettre une police personnalisée dans votre application

1. Allez dans le (dossier du projet)
2. Alors app -> src -> main.
3. Créez le dossier 'assets -> fonts' dans le dossier principal.
4. Mettez votre "fontfile.ttf" dans le dossier des polices.

Initialisation d'une police

```
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");
}
```

Utiliser une police personnalisée dans un TextView

```
public void setFont(TextView textView) {
    textView.setTypeface(myFont);
}
```

Appliquer la police sur TextView par xml (code Java non requis)

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: (Où placer les **res / values**)

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <declare-styleable name="TextViewPlus">
        <attr name="customFont" format="string"/>
    </declare-styleable>
</resources>
```

Comment utiliser:

```
<?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>
```

Police personnalisée dans le texte de toile

Dessiner du texte sur la toile avec votre police à partir d'éléments.

```
Typeface typeface = Typeface.createFromAsset(getAssets(), "fonts/SomeFont.ttf");
Paint textPaint = new Paint();
textPaint.setTypeface(typeface);
canvas.drawText("Your text here", x, y, textPaint);
```

Chargement de police efficace

Le chargement de polices personnalisées peut entraîner de mauvaises performances. Je recommande fortement d'utiliser cette petite aide qui sauvegarde / charge vos polices déjà utilisées dans une table de hachage.

```
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;
}
```

}

Usage:

```
Typeface typeface = TypefaceUtils.getTypeface(context, "RobotoSlab-Bold.ttf");
setTypeface(typeface);
```

Police personnalisée pour toute l'activité

```
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();
    }
}
```

Ensuite, dans votre activité, dans la méthode `onCreate()` :

```
// Put your font to assets folder...

ReplaceFont.changeDefaultFont(getApplication(), "DEFAULT", "LinLibertine.ttf");
```

Travailler avec des polices dans Android O

Android O change la façon de travailler avec les polices.

Android O introduit une nouvelle fonctionnalité, appelée *Fonts in XML* , qui vous permet d'utiliser des polices en tant que ressources. Cela signifie qu'il n'est pas nécessaire de regrouper les polices en tant qu'actifs. Les polices sont maintenant compilées dans un fichier *R* et sont automatiquement disponibles dans le système en tant que ressource.

Pour ajouter une nouvelle **police** , vous devez effectuer les opérations suivantes:

- Créez un nouveau répertoire de ressources: `res/font` .
- Ajoutez vos fichiers de polices dans ce dossier de polices. Par exemple, en ajoutant `myfont.ttf` , vous pourrez utiliser cette police via `R.font.myfont` .

Vous pouvez également créer votre propre **famille de polices** en ajoutant le fichier XML suivant dans le répertoire `res/font` :

```
<?xml version="1.0" encoding="utf-8"?>
<font-family xmlns:android="http://schemas.android.com/apk/res/android">
    <font
        android:fontStyle="normal"
        android:fontWeight="400"
        android:font="@font/lobster_regular" />
    <font
        android:fontStyle="italic"
        android:fontWeight="400"
        android:font="@font/lobster_italic" />
</font-family>
```

Vous pouvez utiliser le fichier de **polices** et le fichier de **famille de polices** de la même manière:

- **Dans un fichier XML,** en utilisant l'attribut `android:fontFamily` , par exemple comme ceci:

  ```
  <TextView
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:fontFamily="@font/myfont"/>
  ```

  Ou comme ça:

  ```
  <style name="customfontstyle" parent="@android:style/TextAppearance.Small">
      <item name="android:fontFamily">@font/myfont</item>
  </style>
  ```

- **Dans votre code** , en utilisant les lignes de code suivantes:

```
Typeface typeface = getResources().getFont(R.font.myfont);
textView.setTypeface(typeface);
```

Lire Polices Personnalisées en ligne: https://riptutorial.com/fr/android/topic/3358/polices-personnalisees

Chapitre 199: Port Mapping en utilisant la bibliothèque Cling dans Android

Examples

Ajout du support Cling à votre projet Android

build.gradle

```
repositories {
  maven { url 'http://4thline.org/m2' }
}

dependencies {

// Cling
compile 'org.fourthline.cling:cling-support:2.1.0'

//Other dependencies required by Cling
compile 'org.eclipse.jetty:jetty-server:8.1.18.v20150929'
compile 'org.eclipse.jetty:jetty-servlet:8.1.18.v20150929'
compile 'org.eclipse.jetty:jetty-client:8.1.18.v20150929'
compile 'org.slf4j:slf4j-jdk14:1.7.14'

}
```

Mapper un port NAT

```
String myIp = getIpAddress();
int port = 55555;

//creates a port mapping configuration with the external/internal port, an internal host IP,
the protocol and an optional description
PortMapping[] desiredMapping = new PortMapping[2];
desiredMapping[0] = new PortMapping(port,myIp, PortMapping.Protocol.TCP);
desiredMapping[1] = new PortMapping(port,myIp, PortMapping.Protocol.UDP);

//starting the UPnP service
UpnpService upnpService = new UpnpServiceImpl(new AndroidUpnpServiceConfiguration());
RegistryListener registryListener = new PortMappingListener(desiredMapping);
upnpService.getRegistry().addListener(registryListener);
upnpService.getControlPoint().search();



//method for getting local ip
private String getIpAddress() {
    String ip = "";
    try {
        Enumeration<NetworkInterface> enumNetworkInterfaces = NetworkInterface
                .getNetworkInterfaces();
        while (enumNetworkInterfaces.hasMoreElements()) {
            NetworkInterface networkInterface = enumNetworkInterfaces
```

```
                .nextElement();
        Enumeration<InetAddress> enumInetAddress = networkInterface
                .getInetAddresses();
        while (enumInetAddress.hasMoreElements()) {
            InetAddress inetAddress = enumInetAddress.nextElement();

            if (inetAddress.isSiteLocalAddress()) {
                ip +=inetAddress.getHostAddress();
            }
        }
    }
 } catch (SocketException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
    ip += "Something Wrong! " + e.toString() + "\n";
 }
 return ip;
}
```

Lire Port Mapping en utilisant la bibliothèque Cling dans Android en ligne: https://riptutorial.com/fr/android/topic/6208/port-mapping-en-utilisant-la-bibliotheque-cling-dans-android

Chapitre 200: Processeur d'annotation

Introduction

Le processeur d'annotations est un outil intégré à javac pour analyser et traiter les annotations au moment de la compilation.

Les annotations sont une classe de métadonnées pouvant être associées à des classes, méthodes, champs et même à d'autres annotations. Il existe deux manières d'accéder à ces annotations lors de l'exécution via réflexion et au moment de la compilation via des processeurs d'annotation.

Examples

@NonNull Annotation

```
public class Foo {
    private String name;
    public Foo(@NonNull String name){...};
    ...
}
```

Ici, @NonNull est une annotation qui est traitée au moment de la compilation par le studio Android pour vous avertir que la fonction particulière nécessite un paramètre non nul.

Types d'annotations

Il existe trois types d'annotations.

1. **Annotation des marqueurs** - annotation sans méthode

   ```
   @interface CustomAnnotation {}
   ```

2. **Annotation à valeur unique** - annotation comportant une méthode

   ```
   @interface CustomAnnotation {
       int value();
   }
   ```

3. **Annotation à valeurs multiples** - annotation comportant plusieurs méthodes

   ```
   @interface CustomAnnotation{
       int value1();
       String value2();
       String value3();
   }
   ```

Création et utilisation d'annotations personnalisées

Pour créer des annotations personnalisées, nous devons décider

- Cible - sur laquelle ces annotations fonctionneront comme le niveau du champ, le niveau de la méthode, le niveau de type, etc.
- Rétention - à quel niveau l'annotation sera disponible.

Pour cela, nous avons intégré des annotations personnalisées. Découvrez ces plus utilisés:

@Cible

Element Types	Where the a
TYPE	class, interfac
FIELD	fields
METHOD	methods
CONSTRUCTOR	constructors
LOCAL_VARIABLE	local variable
ANNOTATION_TYPE	annotation ty
PARAMETER	parameter

@Rétention

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

Création d'annotations personnalisées

```
@Retention(RetentionPolicy.SOURCE) // will not be available in compiled class
@Target(ElementType.METHOD) // can be applied to methods only
@interface CustomAnnotation{
      int value();
}
```

Utilisation de l'annotation personnalisée

```
class Foo{
  @CustomAnnotation(value = 1)  // will be used by an annotation processor
  public void foo(){..}
}
```

la valeur fournie dans `@CustomAnnotation` sera consommée par un Annotationprocessor pour générer du code au moment de la compilation, etc.

Lire Processeur d'annotation en ligne: https://riptutorial.com/fr/android/topic/10726/processeur-d-annotation

Chapitre 201: Programmation Android avec Kotlin

Introduction

Utiliser Kotlin avec Android Studio est une tâche facile, car Kotlin est développé par JetBrains. Il s'agit de la même entreprise qui soutient IntelliJ IDEA - un IDE de base pour Android Studio. C'est pourquoi il n'y a presque aucun problème avec la compatibilité.

Remarques

Si vous voulez en savoir plus sur le langage de programmation Kotlin, consultez Documentation .

Examples

Installer le plugin Kotlin

Tout d'abord, vous devrez installer le plug-in Kotlin.

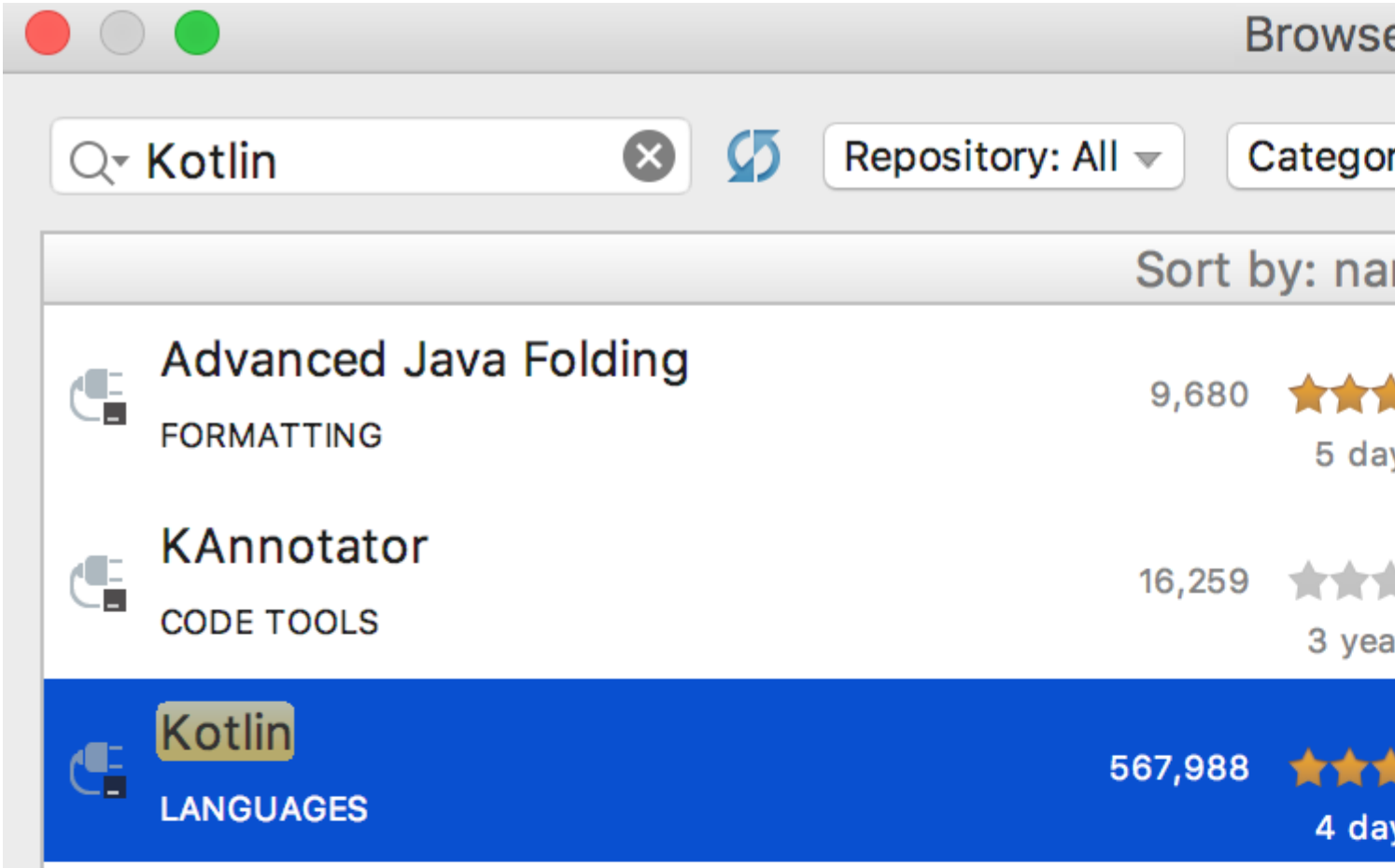
Pour les fenêtres:

- Accédez à `File` → `Settings` → `Plugins Install JetBrains plugin` → `Install JetBrains plugin`

Pour Mac:

- Naviguez vers `Android Studio` → `Preferences` → `Plugins` → `Install JetBrains plugin`

Et puis recherchez et installez Kotlin. Vous devrez redémarrer l'IDE une fois cette opération terminée.


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

, puis y ajouter le support Kotlin ou modifier votre projet existant. Pour ce faire, vous devez:

1. **Ajouter une dépendance à un fichier de graduation racine** - vous devez ajouter la dépendance pour le `kotlin-android` in `kotlin-android` à un fichier racine `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. **Appliquez le plug** - `apply plugin: 'kotlin-android'` **Android Kotlin** - ajoutez simplement le `apply plugin: 'kotlin-android'` à un fichier `build.gradle` module.

3. **Ajoutez une dépendance à Kotlin stdlib** - ajoutez la dépendance à `'org.jetbrains.kotlin:kotlin-stdlib:1.1.2'` à la section de dépendance dans un fichier `build.gradle` module.

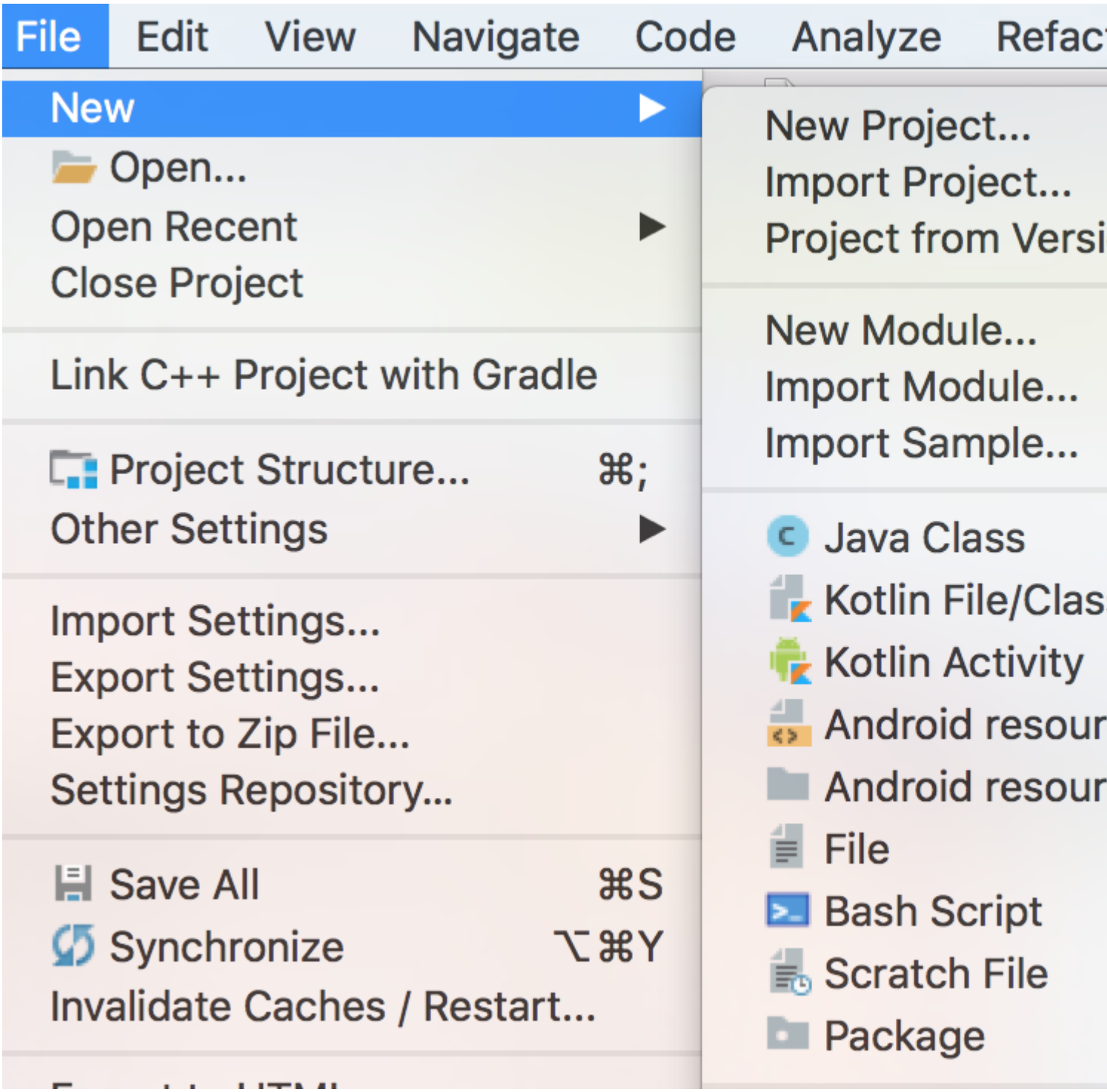
Pour un nouveau projet, le fichier `build.gradle` pourrait ressembler à ceci:

```
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'
}
```

Créer une nouvelle activité Kotlin

1. Cliquez sur `File` → `New` → `Kotlin Activity` .
2. Choisissez un type d'activité.
3. Sélectionnez un nom et un autre paramètre pour l'activité.
4. Terminer.

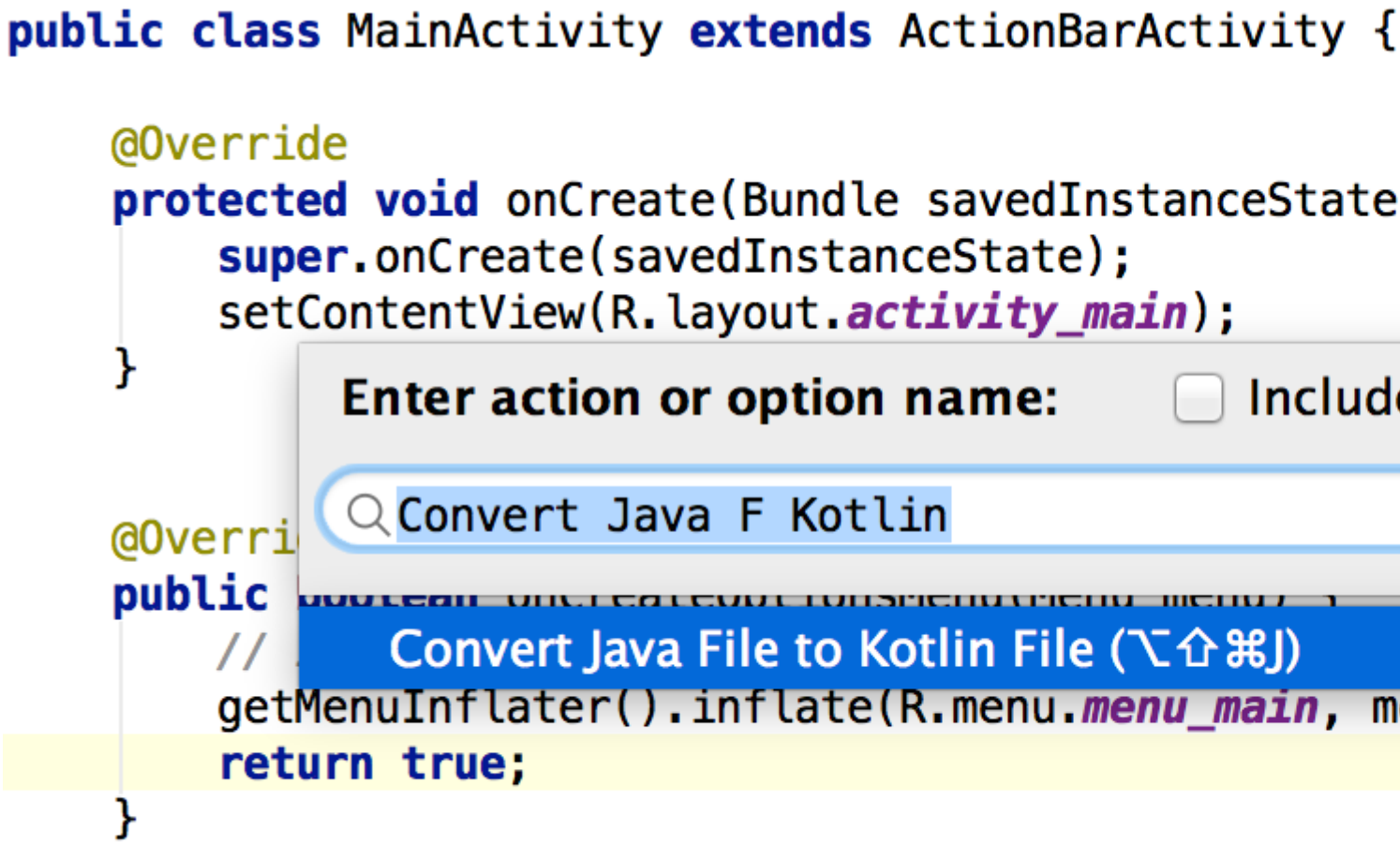


La classe finale pourrait ressembler à ceci:

```
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)
    }
}
```

Conversion du code Java existant en Kotlin

Kotlin Plugin pour Android Studio prend en charge la conversion des fichiers Java existants en fichiers Kotlin. Choisissez un fichier Java et appelez l'action Convertir un fichier Java en fichier Kotlin:



Commencer une nouvelle activité

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

Vous pouvez ajouter des extras à l'intention comme en Java.

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

Lire Programmation Android avec Kotlin en ligne:
https://riptutorial.com/fr/android/topic/9623/programmation-android-avec-kotlin

Chapitre 202: ProGuard - Obscurcir et réduire votre code

Examples

Règles pour certaines des bibliothèques les plus utilisées

Actuellement, il contient des règles pour les bibliothèques suivantes: -

1. Couteau à beurre
2. RxJava
3. Bibliothèque de support Android
4. Bibliothèque de support de conception Android
5. Rénovation
6. Gson et Jackson
7. Otto
8. Crashlitycs
9. Picasso
10. Volée
11. OkHttp3
12. Parcelable

```
#Butterknife
-keep class butterknife.** { *; }
-keepnames class * { @butterknife.Bind *;}

-dontwarn butterknife.internal.**
-keep class **$$ViewBinder { *; }

-keepclasseswithmembernames class * {
    @butterknife.* <fields>;
}

-keepclasseswithmembernames class * {
    @butterknife.* <methods>;
}

# rxjava
-keep class rx.schedulers.Schedulers {
    public static <methods>;
}
-keep class rx.schedulers.ImmediateScheduler {
    public <methods>;
}
-keep class rx.schedulers.TestScheduler {
    public <methods>;
}
-keep class rx.schedulers.Schedulers {
    public static ** test();
}
-keepclassmembers class rx.internal.util.unsafe.*ArrayQueue*Field* {
```

```
    long producerIndex;
    long consumerIndex;
}
-keepclassmembers class rx.internal.util.unsafe.BaseLinkedQueueProducerNodeRef {
    long producerNode;
    long consumerNode;
}

# Support library
-dontwarn android.support.**
-dontwarn android.support.v4.**
-keep class android.support.v4.** { *; }
-keep interface android.support.v4.** { *; }
-dontwarn android.support.v7.**
-keep class android.support.v7.** { *; }
-keep interface android.support.v7.** { *; }

# support design
-dontwarn android.support.design.**
-keep class android.support.design.** { *; }
-keep interface android.support.design.** { *; }
-keep public class android.support.design.R$* { *; }

# retrofit
-dontwarn okio.**
-keepattributes Signature
-keepattributes *Annotation*
-keep class com.squareup.okhttp.** { *; }
-keep interface com.squareup.okhttp.** { *; }
-dontwarn com.squareup.okhttp.**

-dontwarn rx.**
-dontwarn retrofit.**
-keep class retrofit.** { *; }
-keepclasseswithmembers class * {
    @retrofit.http.* <methods>;
}

-keep class sun.misc.Unsafe { *; }
#your package path where your gson models are stored
-keep class com.abc.model.** { *; }

# Keep these for GSON and Jackson
-keepattributes Signature
-keepattributes *Annotation*
-keepattributes EnclosingMethod
-keep class sun.misc.Unsafe { *; }
-keep class com.google.gson.** { *; }

#keep otto
-keepattributes *Annotation*
-keepclassmembers class ** {
    @com.squareup.otto.Subscribe public *;
    @com.squareup.otto.Produce public *;
}

# Crashlitycs 2.+
-keep class com.crashlytics.** { *; }
-keep class com.crashlytics.android.**
-keepattributes SourceFile, LineNumberTable, *Annotation*
# If you are using custom exceptions, add this line so that custom exception types are skipped
```

```
during obfuscation:
-keep public class * extends java.lang.Exception
# For Fabric to properly de-obfuscate your crash reports, you need to remove this line from
your ProGuard config:
# -printmapping mapping.txt

# Picasso
-dontwarn com.squareup.okhttp.**

# Volley
-keep class com.android.volley.toolbox.ImageLoader { *; }

# OkHttp3
-keep class okhttp3.** { *; }
-keep interface okhttp3.** { *; }
-dontwarn okhttp3.**

# Needed for Parcelable/SafeParcelable Creators to not get stripped
-keepnames class * implements android.os.Parcelable {
    public static final ** CREATOR;
}
```

Activer ProGuard pour votre build

Pour activer les configurations `ProGuard` pour votre application, vous devez l'activer dans votre fichier de gradation au niveau du module. vous devez définir la valeur de `minifyEnabled true` .

Vous pouvez également activer `shrinkResources true` ce qui supprimera les ressources `ProGuard` utilisées par ProGuard.

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

Le code ci-dessus appliquera vos configurations ProGuard contenues dans `proguard-rules.pro` ("proguard-project.txt" dans Eclipse) à votre apk publié.

Pour vous permettre de déterminer ultérieurement la ligne sur laquelle une exception s'est produite dans une trace de pile, "proguard-rules.pro" doit contenir les lignes suivantes:

```
-renamesourcefileattribute SourceFile
-keepattributes SourceFile,LineNumberTable
```

Pour activer Proguard dans Eclipse, ajoutez `proguard.config=${sdk.dir}/tools/proguard/proguard-android.txt:proguard-project.txt` à "project.properties".

Supprimer les instructions de journalisation de trace (et autres) au moment de la génération

Si vous voulez supprimer les appels à certaines méthodes, en supposant qu'elles soient vides et sans effets secondaires (comme les appeler ne modifie aucune valeur système, argument de référence, statique, etc.), vous pouvez demander à ProGuard de les supprimer. sortie après la construction est terminée.

Par exemple, je trouve cela utile pour supprimer les instructions de journalisation / débogage utiles au débogage, mais il est inutile de générer les chaînes pour la production.

```
# 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(...);
}
```

Remarque: Si `-dontoptimize` est utilisé dans une configuration ProGuard afin de ne pas minimiser / supprimer le code inutilisé, cela ne supprimera pas les instructions. (Mais qui ne voudrait pas supprimer le code inutilisé, non?)

Note 2: cet appel supprime l'appel à enregistrer, mais ne vous protège pas du code. Les chaînes resteront dans l'apk généré. Lire plus dans cet article .

Protéger votre code contre les pirates

L'obscurcissement est souvent considéré comme une solution magique pour la protection du code, en rendant votre code plus difficile à comprendre s'il est compilé par des pirates.

Mais si vous pensez que la suppression de `Log.x(..)` supprime réellement les informations dont les pirates ont besoin, vous aurez une mauvaise surprise.

Supprimer tous vos appels de journal avec:

```
-assumenosideeffects class android.util.Log {
    public static *** d(...);
    ...etc
}
```

va effectivement supprimer l'appel de journal lui-même, mais généralement *pas* les chaînes que vous y mettez.

Si, par exemple, dans votre journal, vous tapez un message de journal commun tel que: `Log.d(MyTag,"Score="+score);` , le compilateur convertit le `+` en 'new StringBuilder ()' en dehors de l'appel du journal. ProGuard ne change pas ce nouvel objet.

Votre code décompilé aura toujours un `StringBuilder` pour `"Score="` , ajouté à la version obscurcie de la variable `score` (supposons qu'il ait été converti en `b`).
Maintenant, le pirate sait ce qu'est le `b` et donne un sens à votre code.

Une bonne pratique pour supprimer réellement ces résidus de votre code est de ne pas les y

placer en premier lieu (utilisez plutôt le formateur de chaînes avec des règles proguard pour les supprimer), ou envelopper vos appels de `Log` avec:

```
    if (BuildConfig.DEBUG) {
        Log.d(TAG,".."+var);
    }
```

Pointe:

Testez la protection de votre code obscur en la décompilant vous-même!

1. dex2jar - convertit l'apk en jar
2. jd - décompile le fichier jar et l'ouvre dans un éditeur graphique

Activation de ProGuard avec un fichier de configuration d'obscurcissement personnalisé

ProGuard permet au développeur de masquer, réduire et optimiser son code.

 # 1 La première étape de la procédure consiste à activer le proguard sur la construction .

Cela peut être fait en **définissant la commande 'minifyEnabled' sur true** sur la construction de votre choix.

 # 2 La deuxième étape consiste à spécifier les fichiers proguard que nous utilisons pour la construction donnée

Cela peut être fait en **définissant la ligne 'proguardFiles' avec les noms de fichiers appropriés**

```
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 Le développeur peut alors éditer son fichier proguard avec les règles qu'il souhaite.

Cela peut être fait en éditant le fichier (par exemple «proguard-rules-tests.pro») et en ajoutant les contraintes souhaitées. *Le fichier suivant sert d'exemple de fichier proguard*

```
// default & basic optimization configurations
```

```
-optimizationpasses 5
-dontpreverify
-repackageclasses ''
-allowaccessmodification
-optimizations !code/simplification/arithmetic
-keepattributes *Annotation*

-verbose

-dump obfuscation/class_files.txt
-printseeds obfuscation/seeds.txt
-printusage obfuscation/unused.txt // unused classes that are stripped out in the process
-printmapping obfuscation/mapping.txt // mapping file that shows the obfuscated names of the
classes after proguad is applied

// the developer can specify keywords for the obfuscation (I myself use fruits for obfuscation
names once in a while :-) )
-obfuscationdictionary obfuscation/keywords.txt
-classobfuscationdictionary obfuscation/keywords.txt
-packageobfuscationdictionary obfuscation/keywords.txt
```

Enfin, chaque fois que le développeur exécute et / ou génère son nouveau fichier .APK, les configurations proguard personnalisées seront appliquées, répondant ainsi aux exigences.

Lire ProGuard - Obscurcir et réduire votre code en ligne:
https://riptutorial.com/fr/android/topic/4500/proguard---obscurcir-et-reduire-votre-code

Chapitre 203: Publier sur Play Store

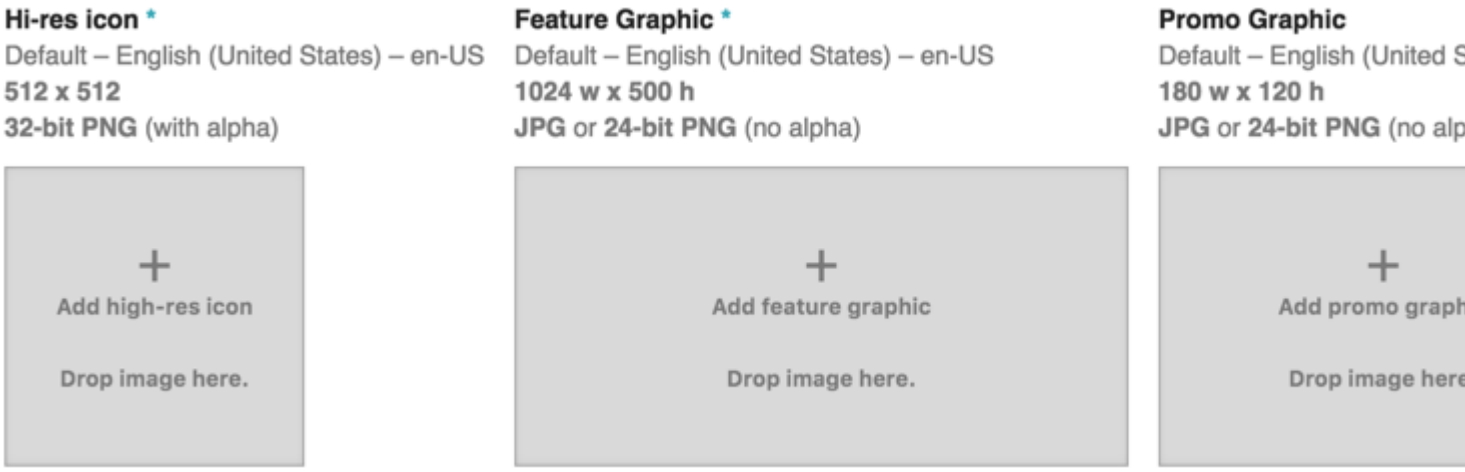
Examples

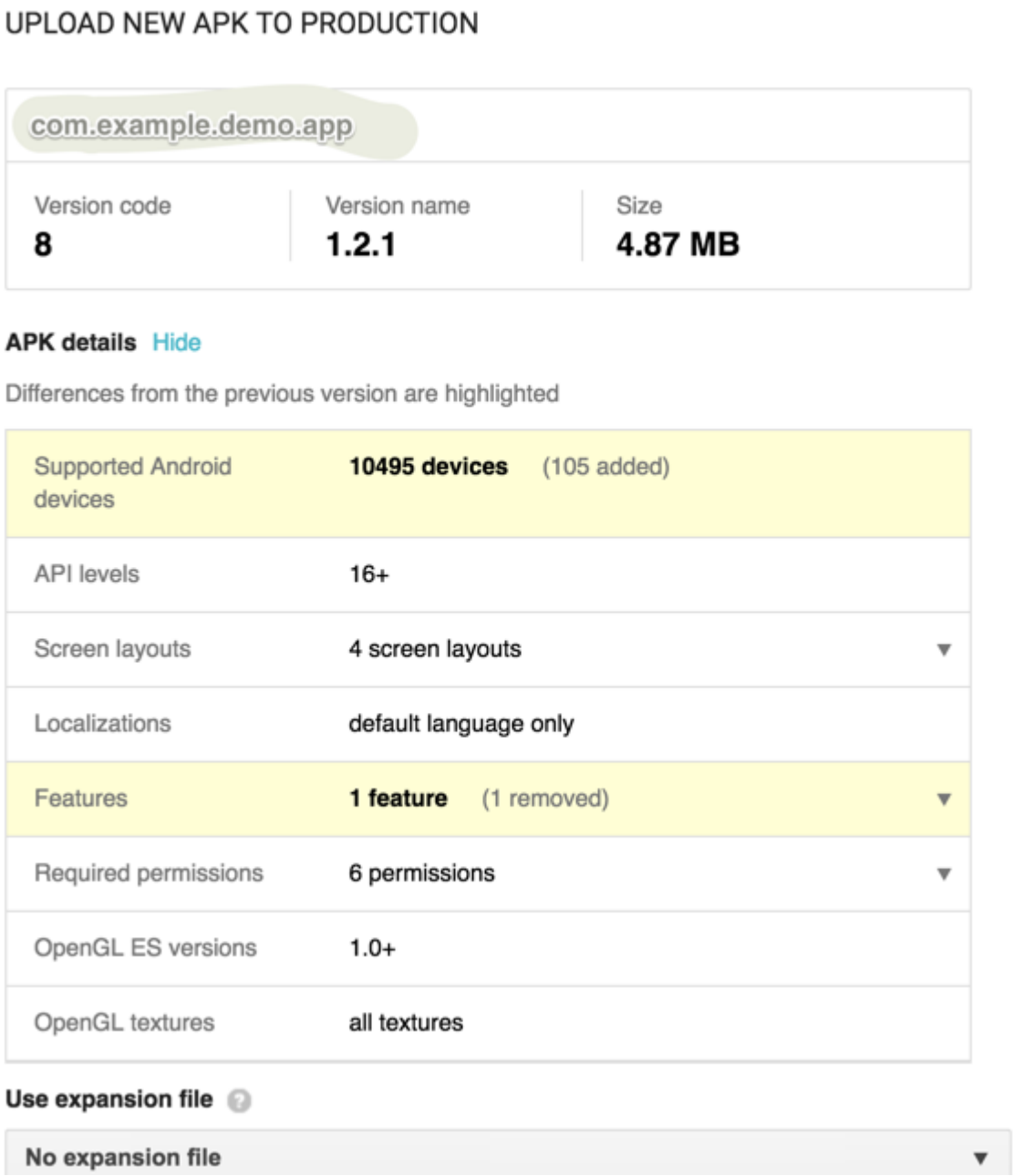
Guide de soumission de l'application minimale

Exigences:

- Un compte développeur
- Un apk déjà construit et signé avec une clé non-debug
- Une application gratuite sans facturation intégrée
- no Firebase Cloud Messaging ou Game Services

1. Rendez-vous sur https://play.google.com/apps/publish/
   1a) Créez votre compte de développeur si vous n'en avez pas
2. Cliquez sur le bouton `Create new Application`
3. Cliquez sur soumettre APK
4. Remplissez tous les champs obligatoires du formulaire, y compris certains éléments qui seront affichés sur le Play Store (voir l'image ci-dessous)
5. Lorsque satisfait `Publish app` bouton `Publish app`





En savoir plus sur la connexion dans Configurer les paramètres de signature

Lire Publier sur Play Store en ligne: https://riptutorial.com/fr/android/topic/5369/publier-sur-play-store

Chapitre 204: Publier un fichier .aar sur Apache Archiva avec Gradle

Examples

Exemple d'implémentation simple

```
apply plugin: 'com.android.library'
apply plugin: 'maven'
apply plugin: 'maven-publish'
android {
compileSdkVersion 21
buildToolsVersion "21.1.2"


repositories {
    mavenCentral()
}

defaultConfig {
    minSdkVersion 9
    targetSdkVersion 21
    versionCode 1
    versionName "1.0"
}

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


dependencies {
   compile fileTree(include: ['*.jar'], dir: 'libs')
   provided 'com.android.support:support-v4:21.0.3'
   provided 'com.android.support:appcompat-v7:21.0.3'
}

task sourceJar(type: Jar) {
   classifier "source"
}

publishing {
   publications {

       repositories.maven {
           url 'myurl/repositories/myrepo'
           credentials {
               username "user"
               password "password"
           }
       }
```

```
        maven(MavenPublication) {
            artifacts {
                groupId 'com.mycompany'
                artifactId 'mylibrary'
                version '1.0'
                artifact 'build/outputs/aar/app-release.aar'
            }
        }
    }

}
```

Lire Publier un fichier .aar sur Apache Archiva avec Gradle en ligne:
https://riptutorial.com/fr/android/topic/6453/publier-un-fichier--aar-sur-apache-archiva-avec-gradle

Chapitre 205: Publier une bibliothèque dans les référentiels Maven

Examples

Publier un fichier .aar sur Maven

Pour publier dans un référentiel au format Maven, le plug-in «maven-publish» pour gradle peut être utilisé.

Le plug-in doit être ajouté au fichier `build.gradle` dans le module bibliothèque.

```
apply plugin: 'maven-publish'
```

Vous devez également définir la publication et ses attributs d'identité dans le fichier `build.gradle` . Ces attributs d'identité seront affichés dans le fichier pom généré et vous les utiliserez à l'avenir pour importer cette publication.Vous devez également définir les artefacts que vous souhaitez publier, par exemple, je souhaite simplement publier le fichier .aar généré après la construction de la bibliothèque. .

```
publishing {
    publications {
        myPulication(MavenPublication) {
            groupId 'com.example.project'
            version '1.0.2'
            artifactId 'myProject'
            artifact("$buildDir/outputs/aar/myProject.aar")
        }
    }
}
```

Vous devrez également définir votre URL de référentiel

```
publishing{
    repositories {
        maven {
            url "http://www.myrepository.com"
        }
    }
}
```

Voici le fichier `build.gradle` bibliothèque `build.gradle`

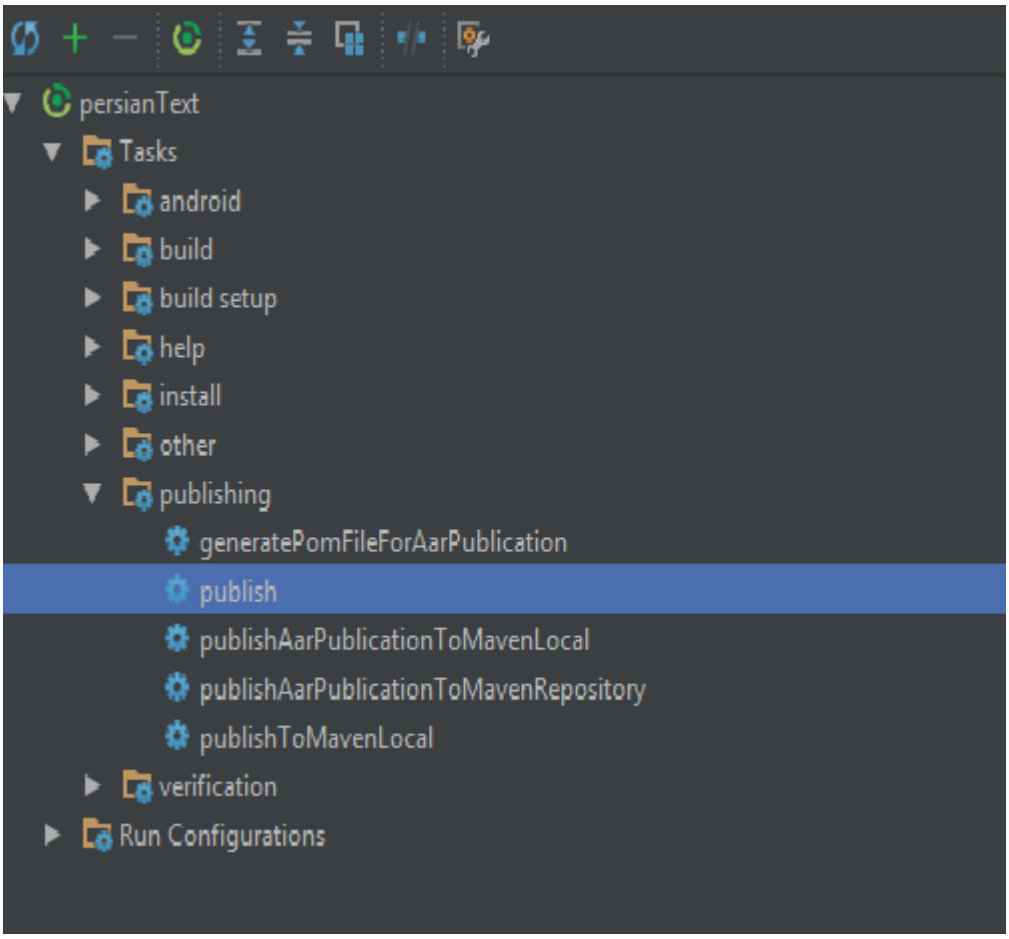
```
apply plugin: 'com.android.library'
apply plugin: 'maven-publish'

buildscript {
   ...
}
```

```
android {
    ...
}
publishing {
    publications {
        myPulication(MavenPublication) {
            groupId 'com.example.project'
            version '1.0.2'
            artifactId 'myProject'
            artifact("$buildDir/outputs/aar/myProject.aar")
        }
    }
    repositories {
        maven {
            url "http://www.myrepository.com"
        }
    }
}
```

Pour la publication, vous pouvez exécuter la commande gradle console

```
gradle publish
```

ou vous pouvez exécuter à partir du panneau des tâches de gradle



Lire Publier une bibliothèque dans les référentiels Maven en ligne:
https://riptutorial.com/fr/android/topic/9359/publier-une-bibliotheque-dans-les-referentiels-maven

Chapitre 206: Qu'est-ce que ProGuard? Qu'est-ce que l'utilisation dans Android?

Introduction

Proguard est un outil de rétraction, optimiseur, obfuscateur et pré-générateur de fichiers Java gratuit. Il détecte et supprime les classes, champs, méthodes et attributs inutilisés. Il optimise le bytecode et supprime les instructions inutilisées. Il renomme les classes, champs et méthodes restants en utilisant des noms courts sans signification.

Examples

Réduisez votre code et vos ressources avec proguard

Pour rendre votre fichier APK aussi petit que possible, vous devez activer la réduction pour supprimer le code et les ressources inutilisés dans votre version. Cette page décrit comment procéder et comment spécifier le code et les ressources à conserver ou à supprimer lors de la génération.

La réduction du code est disponible avec ProGuard, qui détecte et supprime les classes, champs, méthodes et attributs inutilisés de votre application packagée, y compris ceux des bibliothèques de codes incluses (ce qui en fait un outil précieux pour contourner la limite de référence de 64 Ko). ProGuard optimise également le bytecode, supprime les instructions de code inutilisées et masque les autres classes, champs et méthodes avec des noms courts. Le code brouillé rend votre APK difficile à désosser, ce qui est particulièrement utile lorsque votre application utilise des fonctionnalités de sécurité, telles que la vérification des licences.

La réduction des ressources est disponible avec le plug-in Android pour Gradle, qui supprime les ressources inutilisées de votre application packagée, y compris les ressources inutilisées des bibliothèques de codes. Cela fonctionne en conjonction avec le code réduit de telle sorte qu'une fois le code inutilisé a été supprimé, toutes les ressources qui ne sont plus référencées peuvent également être supprimées en toute sécurité.

Réduire votre code

Pour activer la réduction du code avec `ProGuard` , ajoutez `minifyEnabled` au type de génération approprié dans votre fichier `build.gradle` .

Sachez que la réduction du code ralentit le temps de compilation, vous devez donc éviter de l'utiliser si possible dans votre version de débogage. Cependant, il est important que vous activiez le code rétréci sur votre APK final utilisé pour les tests, car cela pourrait introduire des bogues si vous ne personnalisez pas suffisamment le code à conserver.

Par exemple, l'extrait de code suivant d'un fichier `build.gradle` permet de réduire le code pour la version `build.gradle` :

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

Outre la propriété `minifyEnabled` , la propriété `proguardFiles` définit les `ProGuard rules` :

La méthode getDefaultProguardFile ('proguard-android.txt') obtient les paramètres ProGuard par défaut du dossier `tools/proguard/ folder` SDK Android. Conseil: pour réduire encore plus le code, essayez le `proguard-android-optimize.txt` situé au même emplacement. Il inclut les mêmes règles ProGuard, mais avec d'autres optimisations qui effectuent des analyses au niveau du bytecode (à l'intérieur des méthodes et entre celles-ci) pour réduire davantage la taille de l'APK et accélérer son exécution. Le fichier `proguard-rules.pro` est l'endroit où vous pouvez ajouter des règles ProGuard personnalisées. Par défaut, ce fichier est situé à la racine du module (à côté du fichier build.gradle). Pour ajouter d'autres règles ProGuard spécifiques à chaque variante de construction, ajoutez une autre propriété proguardFiles dans le bloc `productFlavor` correspondant. Par exemple, le fichier Gradle suivant ajoute flavour2-rules.pro à l'arôme du produit flavour2. À présent, flavour2 utilise les trois règles ProGuard, car celles du bloc de publication sont également appliquées.

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

Lire Qu'est-ce que ProGuard? Qu'est-ce que l'utilisation dans Android? en ligne: https://riptutorial.com/fr/android/topic/9205/qu-est-ce-que-proguard--qu-est-ce-que-l-utilisation-dans-android-

Chapitre 207: Rapport d'incident de Firebase

Examples

Comment ajouter Firebase Crash Reporting à votre application

Pour ajouter *Firebase Crash Reporting* à votre application, procédez comme suit:

- Créez une application sur la *console Firebase* ici .
- Copiez le fichier `google-services.json` de votre projet dans votre répertoire `app/` .
- Ajoutez les règles suivantes à votre fichier *build.gradle de* niveau *racine* pour inclure le plug `google-services` in `google-services` :

```
buildscript {
    // ...
    dependencies {
        // ...
        classpath 'com.google.gms:google-services:3.0.0'
    }
}
```

- Dans le fichier Gradle de votre module, ajoutez la ligne de `apply plugin` au bas du fichier pour activer le plug-in Gradle:

```
apply plugin: 'com.google.gms.google-services'
```

- Ajoutez la dépendance de *Crash Reporting* à votre fichier *build.gradle au* niveau de l' *application* :

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

- Vous pouvez ensuite déclencher une exception personnalisée à partir de votre application en utilisant la ligne suivante:

```
FirebaseCrash.report(new Exception("Non Fatal Error logging"));
```

  Toutes vos exceptions fatales seront signalées sur votre *console Firebase* .

- Si vous souhaitez ajouter des journaux personnalisés à une console, vous pouvez utiliser le code suivant:

```
FirebaseCrash.log("Level 2 completed.");
```

Pour plus d'informations, s'il vous plaît visitez:

- Documentation officielle
- Sujet dédié Stack Overflow

Comment signaler une erreur

Firebase Crash Reporting génère automatiquement des rapports pour les erreurs fatales (ou les exceptions non interceptées).

Vous pouvez créer votre rapport personnalisé en utilisant:

```
FirebaseCrash.report(new Exception("My first Android non-fatal error"));
```

Vous pouvez archiver le journal lorsque FirebaseCrash a initialisé le module:

> 07–20 08: 57: 24.442 D / FirebaseCrashApiImpl: **API de génération de rapports FirebaseCrash initialisée** 07–20 08: 57: 24.442 I / FirebaseCrash: **initialisation des rapports FirebaseCrash** d com.google.firebase.crash.internal.zzg@3333d325 07–20 08: 57: 24.442 D / FirebaseApp: **classe initialisée com.google.firebase.crash.FirebaseCrash.**

Et puis quand il a envoyé l'exception:

> 07–20 08: 57: 47.052 D / FirebaseCrashApiImpl: **Possibilité de lancer java.lang.Exception: Ma première erreur non fatale Android** 07–20 08: 58: 18.822 D / FirebaseCrashSenderServiceImpl: **Code de réponse: 200** 07–20 08: 58: 18.822 D / FirebaseCrashSenderServiceImpl: **Rapport envoyé**

Vous pouvez ajouter des journaux personnalisés à votre rapport avec

```
FirebaseCrash.log("Activity created");
```

Lire Rapport d'incident de Firebase en ligne: https://riptutorial.com/fr/android/topic/5965/rapport-d-incident-de-firebase

Chapitre 208: RechercherView

Examples

Appcompat SearchView avec l'observateur RxBindings

build.gradle :

```
dependencies {
    compile 'com.android.support:appcompat-v7:23.3.0'
    compile 'com.jakewharton.rxbinding:rxbinding-appcompat-v7:0.4.0'
}
```

menu / menu.xml :

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">

    <item android:id="@+id/action_search" android:title="Search"
        android:icon="@android:drawable/ic_menu_search"
        app:actionViewClass="android.support.v7.widget.SearchView"
        app:showAsAction="always"/>
</menu>
```

MainActivity.java :

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.menu, menu);

    MenuItem searchMenuItem = menu.findItem(R.id.action_search);
    setupSearchView(searchMenuItem );

    return true;
}

private void setupSearchView(MenuItem searchMenuItem) {
    SearchView searchView = (SearchView) searchMenuItem.getActionView();
    searchView.setQueryHint(getString(R.string.search_hint)); // your hint here

    SearchAdapter searchAdapter = new SearchAdapter(this);
    searchView.setSuggestionsAdapter(searchAdapter);

    // optional: set the letters count after which the search will begin to 1
    // the default is 2
    try {
        int autoCompleteTextViewID =
getResources().getIdentifier("android:id/search_src_text", null, null);
        AutoCompleteTextView searchAutoCompleteTextView = (AutoCompleteTextView)
searchView.findViewById(autoCompleteTextViewID);
        searchAutoCompleteTextView.setThreshold(1);
    } catch (Exception e) {
        Logs.e(TAG, "failed to set search view letters threshold");
```

```
    }

    searchView.setOnSearchClickListener(v -> {
        // optional actions to search view expand
    });
    searchView.setOnCloseListener(() -> {
        // optional actions to search view close
        return false;
    });

    RxSearchView.queryTextChanges(searchView)
            .doOnEach(notification -> {
                CharSequence query = (CharSequence) notification.getValue();
                searchAdapter.filter(query);
            })
            .debounce(300, TimeUnit.MILLISECONDS) // to skip intermediate letters
            .flatMap(query -> MyWebService.search(query)) // make a search request
            .retry(3)
            .subscribe(results -> {
                searchAdapter.populateAdapter(results);
            });

    //optional: collapse the searchView on close
    searchView.setOnQueryTextFocusChangeListener((view, queryTextFocused) -> {
        if (!queryTextFocused) {
            collapseSearchView();
        }
    });
}
```

SearchAdapter.java

```
public class SearchAdapter extends CursorAdapter {
    private List<SearchResult> items = Collections.emptyList();

    public SearchAdapter(Activity activity) {
        super(activity, null, CursorAdapter.FLAG_REGISTER_CONTENT_OBSERVER);
    }

    public void populateAdapter(List<SearchResult> items) {
        this.items = items;
        final MatrixCursor c = new MatrixCursor(new String[]{BaseColumns._ID});
        for (int i = 0; i < items.size(); i++) {
            c.addRow(new Object[]{i});
        }
        changeCursor(c);
        notifyDataSetChanged();
    }

    public void filter(CharSequence query) {
        final MatrixCursor c = new MatrixCursor(new String[]{BaseColumns._ID});
        for (int i = 0; i < items.size(); i++) {
            SearchResult result = items.get(i);
            if (result.getText().startsWith(query.toString())) {
                c.addRow(new Object[]{i});
            }
        }
        changeCursor(c);
        notifyDataSetChanged();
    }
```

```
    @Override
    public void bindView(View view, Context context, Cursor cursor) {
        ViewHolder holder = (ViewHolder) view.getTag();
        int position = cursor.getPosition();
        if (position < items.size()) {
            SearchResult result = items.get(position);
            // bind your view here
        }
    }

    @Override
    public View newView(Context context, Cursor cursor, ViewGroup parent) {
        LayoutInflater inflater = (LayoutInflater) context
                .getSystemService(Context.LAYOUT_INFLATER_SERVICE);

        View v = inflater.inflate(R.layout.search_list_item, parent, false);
        ViewHolder holder = new ViewHolder(v);

        v.setTag(holder);
        return v;
    }

    private static class ViewHolder {
        public final TextView text;

        public ViewHolder(View v) {
            this.text= (TextView) v.findViewById(R.id.text);
        }
    }
}
```

SearchView dans la barre d'outils avec 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);
    }
}
```

Capture d'écran de référence:

Hello Android

partir du `menu.xml` , nous avons besoin de comprendre qu'il dépend complètement du style appliqué à la barre d'outils sous-jacente. Pour réaliser le thème, appliquez les étapes suivantes.

Créer un style dans le `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>
```

Appliquez le style à la barre d'outils.

```
<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" />
```

Cela donne la couleur souhaitée à toutes les vues correspondant à la barre d'outils (bouton retour, icônes de menu et SearchView).

Lire RechercherView en ligne: https://riptutorial.com/fr/android/topic/4786/rechercherview

Chapitre 209: Reconnaissance d'activité

Introduction

La reconnaissance d'activité est la détection de l'activité physique d'un utilisateur afin d'effectuer certaines actions sur le périphérique, comme prendre des points lorsqu'un disque est détecté, désactiver le wifi lorsqu'un téléphone est immobile ou mettre le volume de sonnerie au maximum lorsque l'utilisateur est en marchant.

Examples

Google Play ActivityRecognitionAPI

Voici un exemple simple d'utilisation d'ActivityRecognitionApi de GooglePlay Service. Bien qu'il s'agisse d'une excellente bibliothèque, elle ne fonctionne pas sur les appareils sur lesquels Google Play Services n'est pas installé.

Docs pour l'API ActivityRecognition

Manifeste

```
<!-- This is needed to use Activity Recognition! -->
<uses-permission android:name="com.google.android.gms.permission.ACTIVITY_RECOGNITION" />

<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">

    <activity android:name=".MainActivity">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

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

    <receiver android:name=".ActivityReceiver" />
</application>
```

MainActivity.java

```
public class MainActivity extends AppCompatActivity implements
GoogleApiClient.ConnectionCallbacks, GoogleApiClient.OnConnectionFailedListener {

    private GoogleApiClient apiClient;
    private LocalBroadcastManager localBroadcastManager;
    private BroadcastReceiver localActivityReceiver;
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        apiClient = new GoogleApiClient.Builder(this)
                .addApi(ActivityRecognition.API)
                .addConnectionCallbacks(this)
                .addOnConnectionFailedListener(this)
                .build();

        //This just gets the activity intent from the ActivityReceiver class
        localBroadcastManager = LocalBroadcastManager.getInstance(this);
        localActivityReceiver = new BroadcastReceiver() {
            @Override
            public void onReceive(Context context, Intent intent) {
                ActivityRecognitionResult recognitionResult =
ActivityRecognitionResult.extractResult(intent);
                TextView textView = (TextView) findViewById(R.id.activityText);

                //This is just to get the activity name. Use at your own risk.

textView.setText(DetectedActivity.zzkf(recognitionResult.getMostProbableActivity().getType()));

            }
        };
    }

    @Override
    protected void onResume() {
        super.onResume();

        //Register local broadcast receiver
        localBroadcastManager.registerReceiver(localActivityReceiver, new
IntentFilter("activity"));

        //Connect google api client
        apiClient.connect();
    }

    @Override
    protected void onPause() {
        super.onPause();

        //Unregister for activity recognition
        ActivityRecognition.ActivityRecognitionApi.removeActivityUpdates(apiClient,
PendingIntent.getBroadcast(this, 0, new Intent(this, ActivityReceiver.class),
PendingIntent.FLAG_UPDATE_CURRENT));

        //Disconnects api client
        apiClient.disconnect();

        //Unregister local receiver
        localBroadcastManager.unregisterReceiver(localActivityReceiver);
    }

    @Override
    public void onConnected(@Nullable Bundle bundle) {
        //Only register for activity recognition if google api client has connected
        ActivityRecognition.ActivityRecognitionApi.requestActivityUpdates(apiClient, 0,
PendingIntent.getBroadcast(this, 0, new Intent(this, ActivityReceiver.class),
```

```
PendingIntent.FLAG_UPDATE_CURRENT));
    }

    @Override
    public void onConnectionSuspended(int i) {
    }

    @Override
    public void onConnectionFailed(@NonNull ConnectionResult connectionResult) {

    }
}
```

ActivityReceiver

```
public class ActivityReceiver extends BroadcastReceiver {

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

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

Reconnaissance d'activité PathSense

La reconnaissance des activités PathSense est une autre bonne bibliothèque pour les appareils qui ne disposent pas des services Google Play, car ils ont créé leur propre modèle de reconnaissance d'activité, mais exigent que les développeurs s'inscrivent sur http://developer.pathsense.com pour obtenir une clé API et un identifiant client. .

Manifeste

```
<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);
    }
}
```

Lire Reconnaissance d'activité en ligne:
https://riptutorial.com/fr/android/topic/9831/reconnaissance-d-activite

Chapitre 210: RecyclerView

Introduction

RecyclerView est une version plus avancée de l'affichage en liste avec des performances améliorées et des fonctionnalités supplémentaires.

Paramètres

Paramètre	Détail
Adaptateur	Une sous-classe de RecyclerView.Adapter chargée de fournir des vues représentant des éléments dans un ensemble de données
Position	La position d'une donnée dans un adaptateur
Indice	Index d'une vue enfant attachée utilisé dans un appel à getChildAt (int). Contraste avec la position
Contraignant	Processus de préparation d'une vue enfant pour afficher des données correspondant à une position dans l'adaptateur
Recycler (voir)	Une vue précédemment utilisée pour afficher des données pour une position d'adaptateur spécifique peut être placée dans un cache pour une réutilisation ultérieure afin d'afficher à nouveau le même type de données ultérieurement. Cela peut considérablement améliorer les performances en sautant l'inflation ou la construction initiale de la mise en page
Scrap (voir)	Une vue enfant entrée dans un état temporairement détaché lors de la mise en page. Les vues Scrap peuvent être réutilisées sans se détacher complètement du parent RecyclerView, que ce soit sans modification si aucune liaison n'est requise ou modifiée par l'adaptateur si la vue a été considérée comme sale
Sale (voir)	Une vue enfant qui doit être rebondie par l'adaptateur avant d'être affichée

Remarques

`RecyclerView` est une vue flexible pour fournir une fenêtre limitée dans un grand ensemble de données.

Avant d'utiliser `RecyclerView` vous devez ajouter la dépendance de la bibliothèque de support dans le fichier `build.gradle` :

```
dependencies {
```

```
    // Match the version of your support library dependency
    compile 'com.android.support:recyclerview-v7:25.3.1'
}
```

Vous pouvez trouver le dernier numéro de version de recyclerview sur le site officiel.

Autres sujets connexes:

Il y a d'autres sujets qui décrivent les composants `RecyclerView` :

- RecyclerView LayoutManagers
- RecyclerView ItemDecorations
- RecyclerView onClickListeners

Documentation officielle

http://developer.android.com/reference/android/support/v7/widget/RecyclerView.html

Versions plus anciennes:

```
  //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

Ajouter un RecyclerView

Ajoutez la dépendance décrite dans la section Remarque, puis ajoutez `RecyclerView` à votre mise en page:

```
<android.support.v7.widget.RecyclerView
    android:id="@+id/my_recycler_view"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"/>
```

Une fois que vous avez ajouté un widget `RecyclerView` à votre mise en page, obtenez un handle pour l'objet, connectez-le à un gestionnaire de disposition et connectez un adaptateur pour que les données soient affichées:

```
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);
```

Ou simplement configurer le gestionnaire de disposition à partir de XML en ajoutant ces lignes:

```
xmlns:app="http://schemas.android.com/apk/res-auto"
app:layoutManager="android.support.v7.widget.LinearLayoutManager"
```

Si vous savez que les modifications apportées au contenu de `RecyclerView` ne modifieront pas la taille de mise en page de `RecyclerView` , utilisez le code suivant pour améliorer les performances du composant. Si RecyclerView a une taille fixe, il sait que RecyclerView lui-même ne sera pas redimensionné en raison de ses enfants. Il ne fait que gérer le changement lui-même. Si vous invalidez le parent, le coordinateur, la mise en page, etc. (vous pouvez utiliser cette méthode avant même de définir `LayoutManager` et `Adapter`):

```
mRecyclerView.setHasFixedSize(true);
```

`RecyclerView` fournit ces gestionnaires de disposition intégrés à utiliser. Vous pouvez donc créer une liste, une grille et une grille échelonnée en utilisant `RecyclerView` :

1. LinearLayoutManager affiche les éléments dans une liste de défilement verticale ou horizontale.
2. GridLayoutManager affiche les éléments dans une grille.
3. StaggeredGridLayoutManager affiche les éléments dans une grille échelonnée.

Chargement plus fluide des articles

Si les éléments de votre `RecyclerView` chargent des données du réseau (généralement des images) ou effectuent d'autres traitements, cela peut prendre un temps considérable et vous pouvez vous retrouver avec des éléments à l'écran mais pas complètement chargés. Pour éviter cela, vous pouvez étendre le `LinearLayoutManager` existant pour précharger un certain nombre d'éléments avant qu'ils ne soient visibles à l'écran:

```
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;
  }
}
```

Glisser-Déposer et Glisser avec RecyclerView

Vous pouvez implémenter les fonctionnalités de glisser-déplacer et de glisser-déposer avec RecyclerView sans utiliser de bibliothèques tierces.
Utilisez simplement la classe `ItemTouchHelper` incluse dans la bibliothèque de support RecyclerView.

Instanciez `ItemTouchHelper` avec le rappel `SimpleCallback` et, selon la fonctionnalité que vous prenez en charge, vous devez remplacer `onMove(RecyclerView, ViewHolder, ViewHolder)` et / ou `onSwiped(ViewHolder, int)` et attacher enfin à votre `RecyclerView` .

```
ItemTouchHelper.SimpleCallback simpleItemTouchCallback = new ItemTouchHelper.SimpleCallback(0,
ItemTouchHelper.LEFT | ItemTouchHelper.RIGHT) {

    @Override
    public void onSwiped(RecyclerView.ViewHolder viewHolder, int swipeDir) {
        // remove item from adapter
    }

    @Override
    public boolean onMove(RecyclerView recyclerView, RecyclerView.ViewHolder viewHolder,
RecyclerView.ViewHolder target) {
            final int fromPos = viewHolder.getAdapterPosition();
            final int toPos = target.getAdapterPosition();
            // move item in `fromPos` to `toPos` in adapter.
            return true;// true if moved, false otherwise
        }

};

ItemTouchHelper itemTouchHelper = new ItemTouchHelper(simpleItemTouchCallback);
itemTouchHelper.attachToRecyclerView(recyclerView);
```

Il convient de mentionner que le constructeur de `SimpleCallback` applique la même stratégie de balayage à tous les éléments de `RecyclerView` . Dans tous les cas, il est possible de mettre à jour la direction de balayage par défaut pour des éléments spécifiques en `getSwipeDirs(RecyclerView, ViewHolder)` simplement la méthode `getSwipeDirs(RecyclerView, ViewHolder)` .

Supposons par exemple que notre `RecyclerView` inclue un `HeaderViewHolder` et que nous ne souhaitons évidemment pas lui appliquer de glissement. Il suffira de remplacer `getSwipeDirs` comme suit:

```
@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);
}
```

Ajouter en-tête / pied de page à un RecyclerView

Ceci est un exemple de code d'adaptateur.

```
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
    }
}
}
```

Voici une bonne lecture de l'implémentation de `RecyclerView` avec en-tête et pied de page.

Méthode alternative:

Bien que la réponse ci-dessus fonctionne, vous pouvez également utiliser cette approche en utilisant une vue de recycleur utilisant un `NestedScrollView` Vous pouvez ajouter une mise en page pour l'en-tête en utilisant l'approche suivante:

```
<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>
```

Ou vous pouvez également utiliser un `LinearLayout` avec un alignement vertical dans votre `NestedScrollView` .

Note: Cela ne fonctionnera qu'avec `RecyclerView` au-dessus de **23.2.0**

```
compile 'com.android.support:recyclerview-v7:23.2.0'
```

Utiliser plusieurs ViewHolders avec ItemViewType

Parfois, un RecyclerView doit utiliser plusieurs types de vues pour être affiché dans la liste

affichée dans l'interface utilisateur, et chaque vue nécessite un fichier XML de mise en page différent.

Pour ce problème, vous pouvez utiliser différents ViewHolders dans un seul adaptateur, en utilisant une méthode spéciale dans RecyclerView - `getItemViewType(int position)` .

Voici un exemple d'utilisation de deux ViewHolders:

1. Un ViewHolder pour afficher les entrées de la liste
2. Un ViewHolder pour afficher plusieurs vues d'en-tête

```
@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.
    }
}
```

Filtrer les éléments dans RecyclerView avec un SearchView

Ajouter une méthode de `filter` dans `RecyclerView.Adapter` :

```
public void filter(String text) {
        if(text.isEmpty()){
            items.clear();
            items.addAll(itemsCopy);
        } else{
            ArrayList<PhoneBookItem> result = new ArrayList<>();
            text = text.toLowerCase();
            for(PhoneBookItem item: itemsCopy){
                //match by name or phone
                if(item.name.toLowerCase().contains(text) ||
item.phone.toLowerCase().contains(text)){
                    result.add(item);
                }
            }
            items.clear();
            items.addAll(result);
        }
        notifyDataSetChanged();
    }
```

`itemsCopy` est initialisé dans le constructeur de l'adaptateur comme `itemsCopy.addAll(items)` .

Si vous le faites, appelez simplement le `filter` de `OnQueryTextListener` partir de `SearchView` :

```
searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
    @Override
    public boolean onQueryTextSubmit(String query) {
        adapter.filter(query);
        return true;
    }

    @Override
    public boolean onQueryTextChange(String newText) {
        adapter.filter(newText);
        return true;
    }
});
```

Menu contextuel avec recyclerView

mettre ce code dans votre ViewHolder

note: Dans ce code, j'utilise `btnExpand` click-event, pour tout événement click de `recyclerview` , vous pouvez définir l'écouteur sur l'objet itemView.

```
public class MyViewHolder extends RecyclerView.ViewHolder{
```

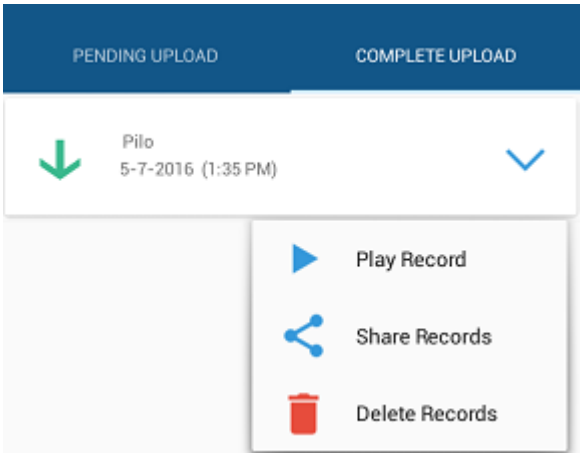
```
        CardView cv;
        TextView recordName, visibleFile, date, time;
        Button btnIn, btnExpand;

        public MyViewHolder(final View itemView) {
            super(itemView);

            cv = (CardView)itemView.findViewById(R.id.cardview);
            recordName = (TextView)itemView.findViewById(R.id.tv_record);
            visibleFile = (TextView)itemView.findViewById(R.id.visible_file);
            date = (TextView)itemView.findViewById(R.id.date);
            time = (TextView)itemView.findViewById(R.id.time);
            btnIn = (Button)itemView.findViewById(R.id.btn_in_out);

            btnExpand = (Button) itemView.findViewById(R.id.btn_expand);

            btnExpand.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    PopupMenu popup = new PopupMenu(btnExpand.getContext(), itemView);

                    popup.setOnMenuItemClickListener(new PopupMenu.OnMenuItemClickListener() {
                        @Override
                        public boolean onMenuItemClick(MenuItem item) {
                            switch (item.getItemId()) {
                                case R.id.action_delete:
                                    moveFile(recordName.getText().toString(),
getAdapterPosition());
                                    return true;
                                case R.id.action_play:
                                    String valueOfPath = recordName.getText().toString();
                                    Intent intent = new Intent();
                                    intent.setAction(android.content.Intent.ACTION_VIEW);
                                    File file = new File(valueOfPath);
                                    intent.setDataAndType(Uri.fromFile(file), "audio/*");
                                    context.startActivity(intent);
                                    return true;
                                case R.id.action_share:
                                    String valueOfPath = recordName.getText().toString();
                                    File filee = new File(valueOfPath);
                                    try {
                                        Intent sendIntent = new Intent();
                                        sendIntent.setAction(Intent.ACTION_SEND);
                                        sendIntent.setType("audio/*");
                                        sendIntent.putExtra(Intent.EXTRA_STREAM,
Uri.fromFile(filee));
                                        context.startActivity(sendIntent);
                                    } catch (NoSuchMethodError | IllegalArgumentException |
NullPointerException e) {
                                        e.printStackTrace();
                                    } catch (Exception e) {
                                        e.printStackTrace();
                                    }
                                    return true;
                                default:
                                    return false;
                            }
                        }
                    });
                    // here you can inflate your menu
                    popup.inflate(R.menu.my_menu_item);
```

```
                    popup.setGravity(Gravity.RIGHT);

                    // if you want icon with menu items then write this try-catch block.
                    try {
                        Field mFieldPopup=popup.getClass().getDeclaredField("mPopup");
                        mFieldPopup.setAccessible(true);
                        MenuPopupHelper mPopup = (MenuPopupHelper) mFieldPopup.get(popup);
                        mPopup.setForceShowIcon(true);
                    } catch (Exception e) {

                    }
                    popup.show();
                }
            });

        }
    }
```

autre manière de montrer des icônes dans le menu

```
try {
    Field[] fields = popup.getClass().getDeclaredFields();
    for (Field field : fields) {
        if ("mPopup".equals(field.getName())) {
            field.setAccessible(true);
            Object menuPopupHelper = field.get(popup);
            Class<?> classPopupHelper = Class.forName(menuPopupHelper
                .getClass().getName());
            Method setForceIcons = classPopupHelper.getMethod(
                "setForceShowIcon", boolean.class);
            setForceIcons.invoke(menuPopupHelper, true);
            break;
        }
    }
} catch (Exception e) {

}
```

Voici la sortie:



Animer le changement de données

`RecyclerView` effectuera une animation pertinente si l'une des méthodes "notify" est utilisée à

l'exception de `notifyDataSetChanged` ; cela inclut `notifyItemChanged` , `notifyItemInserted` , `notifyItemMoved` , `notifyItemRemoved` , etc.

L'adaptateur doit étendre cette classe à la place de `RecyclerView.Adapter` .

```
import android.support.annotation.NonNull;
import android.support.v7.widget.RecyclerView;

import java.util.List;

public abstract class AnimatedRecyclerAdapter<T, VH extends RecyclerView.ViewHolder>
        extends RecyclerView.Adapter<VH> {
    protected List<T> models;

    protected AnimatedRecyclerAdapter(@NonNull List<T> models) {
        this.models = models;
    }

    //Set new models.
    public void setModels(@NonNull final List<T> models) {
        applyAndAnimateRemovals(models);
        applyAndAnimateAdditions(models);
        applyAndAnimateMovedItems(models);
    }

    //Remove an item at position and notify changes.
    private T removeItem(int position) {
        final T model = models.remove(position);
        notifyItemRemoved(position);
        return model;
    }

    //Add an item at position and notify changes.
    private void addItem(int position, T model) {
        models.add(position, model);
        notifyItemInserted(position);
    }

    //Move an item at fromPosition to toPosition and notify changes.
    private void moveItem(int fromPosition, int toPosition) {
        final T model = models.remove(fromPosition);
        models.add(toPosition, model);
        notifyItemMoved(fromPosition, toPosition);
    }

    //Remove items that no longer exist in the new models.
    private void applyAndAnimateRemovals(@NonNull final List<T> newTs) {
        for (int i = models.size() - 1; i >= 0; i--) {
            final T model = models.get(i);
            if (!newTs.contains(model)) {
                removeItem(i);
            }
        }
    }

    //Add items that do not exist in the old models.
    private void applyAndAnimateAdditions(@NonNull final List<T> newTs) {
        for (int i = 0, count = newTs.size(); i < count; i++) {
            final T model = newTs.get(i);
            if (!models.contains(model)) {
```

```
                addItem(i, model);
            }
        }
    }

    //Move items that have changed their position.
    private void applyAndAnimateMovedItems(@NonNull final List<T> newTs) {
        for (int toPosition = newTs.size() - 1; toPosition >= 0; toPosition--) {
            final T model = newTs.get(toPosition);
            final int fromPosition = models.indexOf(model);
            if (fromPosition >= 0 && fromPosition != toPosition) {
                moveItem(fromPosition, toPosition);
            }
        }
    }
}
```

> Vous ne devez **PAS** utiliser la même `List` pour `setModels` et `List` dans l'adaptateur.

Vous déclarez des `models` tant que variables globales. `DataModel` est une classe factice uniquement.

```
private List<DataModel> models;
private YourAdapter adapter;
```

Initialisez les `models` avant de les transmettre à l'adaptateur. `YourAdapter` est l'implémentation de `AnimatedRecyclerAdapter` .

```
models = new ArrayList<>();
//Add models
models.add(new DataModel());
//Do NOT pass the models directly. Otherwise, when you modify global models,
//you will also modify models in adapter.
//adapter = new YourAdapter(models); <- This is wrong.
adapter = new YourAdapter(new ArrayList(models));
```

Appelez ceci après avoir mis à jour vos `models` globaux.

```
adapter.setModels(new ArrayList(models));
```

> Si vous ne remplacez pas la valeur `equals` , toute la comparaison est comparée par référence.

Exemple utilisant SortedList

Android a introduit la classe `SortedList` peu après l'introduction de `RecyclerView` . Cette classe gère tous les appels de méthode 'notify' au `RecyclerView.Adapter` pour garantir une animation correcte, et permet même le traitement par lots de plusieurs modifications, de sorte que les animations ne changent pas.

```
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 avec DataBinding

Voici une classe ViewHolder générique que vous pouvez utiliser avec n'importe quelle disposition

de DataBinding. Ici, une instance de classe `ViewDataBinding` particulière est créée à l'aide de l'objet `View` gonflé et de la classe d'utilitaire `DataBindingUtil` .

```
import android.databinding.DataBindingUtil;
import android.support.v7.widget.RecyclerView;
import android.view.View;

public class BindingViewHolder<T> extends RecyclerView.ViewHolder{

    private final T binding;

    public BindingViewHolder(View itemView) {
        super(itemView);
        binding = (T)DataBindingUtil.bind(itemView);
    }

    public T getBinding() {
        return binding;
    }
}
```

Après avoir créé cette classe, vous pouvez utiliser `<layout>` dans votre fichier de mise en page pour activer la liaison de données pour cette mise en page, comme ceci:

`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>
```

et voici votre exemple de dataModel:

```
public class ItemModel {
    public String itemLabel;
}
```

Par défaut, la bibliothèque Android Data Binding génère une classe `ViewDataBinding` basée sur le nom du fichier de mise en page, en la convertissant en cas Pascal et en y ajoutant le suffixe "Binding". Pour cet exemple, ce serait `MyItemBinding` pour le fichier de présentation `my_item.xml` . Cette classe Binding aurait également une méthode setter pour définir l'objet défini en tant que

données dans le fichier de disposition (`ItemModel` pour cet exemple).

Maintenant que nous avons toutes les pièces, nous pouvons implémenter notre adaptateur comme ceci:

```
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();
    }
}
```

Défilement sans fin dans Recycleview.

Ici, j'ai partagé un extrait de code pour implémenter le défilement sans fin dans la vue de recyclage.

Étape 1: Créez d'abord une méthode abstraite dans l'adaptateur Recycleview, comme ci-dessous.

```
public abstract class ViewAllCategoryAdapter extends
RecyclerView.Adapter<RecyclerView.ViewHolder> {
    public abstract void load();
}
```

Étape 2: Remplacez maintenant la méthode onBindViewHolder et `getItemCount()` de la classe ViewAllCategoryAdapter et appelez la méthode `Load()` comme ci-dessous.

```
@Override
public void onBindViewHolder(RecyclerView.ViewHolder holder, final int position) {
    if ((position >= getItemCount() - 1)) {
        load();
    }
}

@Override
public int getItemCount() {
    return YOURLIST.size();
```

```
}
```

Etape 3: Maintenant que chaque logique de backend est terminée, il est temps d'exécuter cette logique. C'est simple, vous pouvez remplacer la méthode de chargement où vous créez l'objet de votre adaptateur.

```
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);
```

La méthode `load()` appelle maintenant automatiquement lorsque l'utilisateur fait défiler à la fin de la liste.

Meilleure chance

Afficher la vue par défaut jusqu'à ce que les éléments se chargent ou quand les données ne sont pas disponibles

Capture d'écran

Classe d'adaptateur

```
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>
```

J'utilise FontAwesome avec Iconics Library pour les images. Ajoutez ceci à votre fichier build.gradle au niveau de l'application.

```
compile 'com.mikepenz:fontawesome-typeface:4.6.0.3@aar'
compile 'com.mikepenz:iconics-core:2.8.1@aar'
```

Ajouter des lignes de séparation aux éléments RecyclerView

Ajoutez simplement ces lignes à l'initialisation

```
RecyclerView mRecyclerView = (RecyclerView) view.findViewById(recyclerView);
mRecyclerView.setLayoutManager(new LinearLayoutManager(getActivity()));
mRecyclerView.addItemDecoration(new DividerItemDecoration(getActivity(),
DividerItemDecoration.VERTICAL));
```

Ajoutez un `adapter` et appelez `.notifyDataSetChanged();` comme d'habitude !
Ce n'est pas une fonctionnalité intégrée de Recyclerview, mais ajoutée aux bibliothèques de support. Donc, n'oubliez pas d'inclure ceci dans votre fichier build.gradle au niveau de l'application

```
compile "com.android.support:appcompat-v7:25.3.1"
```

```
compile "com.android.support:recyclerview-v7:25.3.1"
```

Plusieurs ItemDecorations peuvent être ajoutés à un seul RecyclerView.

Changer la couleur du séparateur :

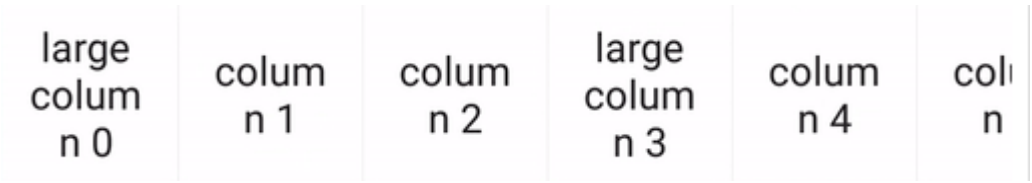
Il est assez facile de définir une couleur pour un objetDecoration.

1. l'étape est la suivante: créer un fichier `divider.xml` qui se trouve dans `drawable` dossier `drawable`

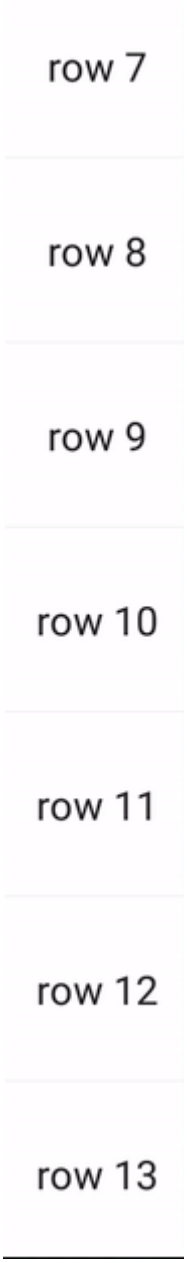
```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
       android:shape="line">
    <size
        android:width="1px"
        android:height="1px"/>
    <solid android:color="@color/divider_color"/>
</shape>
```

2. l'étape est la suivante: mise en dessin

```
    // 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);
```



Lire RecyclerView en ligne: https://riptutorial.com/fr/android/topic/169/recyclerview

Chapitre 211: RecyclerView et LayoutManagers

Examples

GridLayoutManager avec comptage dynamique

Lors de la création d'une recyclerview avec un gestionnaire de disposition gridlayout, vous devez spécifier le comptage de port dans le constructeur. Le nombre de plages correspond au nombre de colonnes. Ceci est assez encombrant et ne prend pas en compte les grandes tailles d'écran ou l'orientation de l'écran. Une approche consiste à créer plusieurs dispositions pour les différentes tailles d'écran. Une autre approche plus dynamique peut être vue ci-dessous.

Tout d'abord, nous créons une classe RecyclerView personnalisée comme suit:

```
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);
        }
    }
}
```

Cette classe détermine le nombre de colonnes pouvant être intégrées à la recyclerview. Pour l'utiliser, vous devrez le mettre dans votre layout.xml comme suit:

```
<?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"
    />
```

Notez que nous utilisons l'attribut columnWidth. Recyclerview en aura besoin pour déterminer le nombre de colonnes pouvant être insérées dans l'espace disponible.

Dans votre activité / fragment, vous obtenez une référence à la recylerview et définissez-y un adaptateur (ainsi que les décorations ou animations que vous souhaitez ajouter). **NE DÉFINISSEZ PAS DE GESTIONNAIRE DE MISE EN PAGE**

```
RecyclerView recyclerView = (RecyclerView) findViewById(R.id.auto_fit_recycler_view);
recyclerView.setAdapter(new MyAdapter());
```

(où MyAdapter est votre classe d'adaptateur)

Vous avez maintenant une liste de recyclage qui ajustera le compte (c.-à-d. Les colonnes) en fonction de la taille de l'écran. En dernier lieu, vous voudrez peut-être centrer les colonnes dans la recyclerview (par défaut, elles sont alignées sur layout_start). Vous pouvez le faire en modifiant un peu la classe AutofitRecyclerView. Commencez par créer une classe interne dans la recyclerview. Ce sera une classe qui s'étend de GridLayoutManager. Il ajoutera suffisamment de remplissage à gauche et à droite pour centrer les lignes:

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

Ensuite, lorsque vous définissez le LayoutManager dans AutofitRecyclerView, utilisez le CenteredGridLayoutManager comme suit:

```
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);
}
```

Et c'est tout! Vous disposez d'un compte dynamique, centré sur la grille, basé sur le centre de grille.

Sources:

- Blog de Chiu-Ki Chan's Square Island
- StackOverflow

Ajout de la vue d'en-tête à recyclerview avec le gestionnaire gridlayout

Pour ajouter un en-tête à une recyclerview avec un gridlayout, il faut d'abord indiquer à l'adaptateur que la vue d'en-tête est la première position, plutôt que la cellule standard utilisée pour le contenu. Ensuite, le gestionnaire de disposition doit être informé que la première position doit avoir une portée égale au nombre * span de la liste entière. *

Prenez une classe RecyclerView.Adapter régulière et configurez-la comme suit:

```
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
    }
}
```

Puis dans l'activité / 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);
```

La même approche peut être utilisée en ajoutant un pied de page en plus ou au lieu d'un en-tête.

Source: blog Square Chiu-Ki Chan

Liste simple avec LinearLayoutManager

Cet exemple ajoute une liste des lieux avec l' image et le nom en utilisant une `ArrayList` de la coutume `Place` des objets comme jeu de données.

Disposition de l'activité

La disposition de l'activité / du fragment ou de l'emplacement où RecyclerView est utilisé doit uniquement contenir RecyclerView. Il n'y a pas de ScrollView ou une disposition spécifique nécessaire.

```
<?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>
```

Définir le modèle de données

Vous pouvez utiliser n'importe quel type de données de classe ou primitive en tant que modèle, comme `int` , `String` , `float[]` ou `CustomObject` . RecyclerView fera référence à une `List` de ces objets / primitives.

Lorsqu'un élément de liste fait référence à différents types de données tels que du texte, des nombres, des images (comme dans cet exemple avec des espaces), il est souvent utile d'utiliser un objet personnalisé.

```
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;
    }
}
```

Mise en page des éléments de la liste

Vous devez spécifier un fichier de présentation xml qui sera utilisé pour chaque élément de la liste. Dans cet exemple, un `ImageView` est utilisé pour l'image et un `TextView` pour le nom. Le `LinearLayout` positionne le `ImageView` à gauche et le `TextView` droit à l'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>
```

Créez un adaptateur RecyclerView et ViewHolder

Ensuite, vous devez hériter de `RecyclerView.Adapter` et du `RecyclerView.ViewHolder` . Une structure de classe habituelle serait:

```
public class PlaceListAdapter extends RecyclerView.Adapter<PlaceListAdapter.ViewHolder> {
    // ...

    public class ViewHolder extends RecyclerView.ViewHolder {
        // ...
```

```
    }
}
```

Tout d'abord, nous implémentons le `ViewHolder` . Il hérite uniquement du constructeur par défaut et enregistre les vues nécessaires dans certains champs:

```
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);
    }
}
```

Le constructeur de l'adaptateur définit le jeu de données utilisé:

```
public class PlaceListAdapter extends RecyclerView.Adapter<PlaceListAdapter.ViewHolder> {
    private List<Place> mPlaces;

    public PlaceListAdapter(List<Place> contacts) {
        mPlaces = contacts;
    }

    // ...
}
```

Pour utiliser notre disposition d'élément de liste personnalisée, nous `onCreateViewHolder(...)` la méthode `onCreateViewHolder(...)` . Dans cet exemple, le fichier de disposition est appelé `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);
    }

    // ...
}
```

Dans le `onBindViewHolder(...)` , nous définissons réellement le contenu des vues. Nous obtenons le modèle utilisé en le trouvant dans la `List` à la position donnée, puis nous définissons l'image et le nom sur les `ViewHolder` du `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());
    }

    // ...
}
```

Nous devons également mettre en œuvre `getItemCount()` , qui renvoie simplement la `List` taille de.

```
public class PlaceListAdapter extends RecyclerView.Adapter<PlaceListAdapter.ViewHolder> {
    // ...

    @Override
    public int getItemCount() {
        return mPlaces.size();
    }

    // ...
}
```

(Générer des données aléatoires)

Pour cet exemple, nous allons générer des endroits aléatoires.

```
@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;
}
```

Connectez le RecyclerView à la PlaceListAdapter et à l'ensemble de données

Connecter un `RecyclerView` à un adaptateur est très simple. Vous devez définir `LinearLayoutManager` comme gestionnaire de disposition pour obtenir la disposition de liste.

```
@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));
}
```

Terminé!

StaggeredGridLayoutManager

1. Créez votre RecyclerView dans votre fichier XML de mise en page:

```
<android.support.v7.widget.RecyclerView
            android:id="@+id/recycleView"
            android:layout_width="match_parent"
            android:layout_height="match_parent" />
```

2. Créez votre classe de modèle pour conserver vos données:

```
     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. Créez un fichier de disposition pour contenir les éléments RecyclerView:

```
<ImageView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
```

```
    android:adjustViewBounds="true"
    android:scaleType="centerCrop"
    android:id="@+id/imageView"/>
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:id="@+id/name"
        android:layout_gravity="center"
        android:textColor="@android:color/white"/>
```

4. Créez la classe d'adaptateur pour 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. Instanciez le RecyclerView dans votre activité ou 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() );
```

N'oubliez pas d'ajouter la dépendance Picasso dans votre fichier build.gradle:

```
compile 'com.squareup.picasso:picasso:2.5.2'
```

Lire RecyclerView et LayoutManagers en ligne:
https://riptutorial.com/fr/android/topic/6772/recyclerview-et-layoutmanagers

Chapitre 212: RecyclerView onClickListeners

Examples

Nouvel exemple

```
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 et RxJava exemple

Premier exemple réimplémenté dans Kotlin et utilisant RxJava pour une interaction plus propre.

```
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.support.v7.widget.RecyclerView
import rx.subjects.PublishSubject

public class SampleAdapter(private val items: Array<String>) :
RecyclerView.Adapter<SampleAdapter.ViewHolder>() {

    // change to different subjects from rx.subjects to get different behavior
    // BehaviorSubject for example allows to receive last event on subscribe
    // PublishSubject sends events only after subscribing on the other hand which is desirable
for clicks
    public val itemClickStream: PublishSubject<View> = PublishSubject.create()

    override fun getItemCount(): Int {
        return items.size
    }

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ViewHolder? {
        val v = LayoutInflater.from(parent.getContext()).inflate(R.layout.text_row_item,
parent, false);
        return ViewHolder(view)
    }

    override fun onBindViewHolder(holder: ViewHolder, position: Int) {
        holder.bind(items[position])
    }

    public inner class ViewHolder(view: View) : RecyclerView.ViewHolder(view) {
        private val textView: TextView by lazy { view.findViewById(R.id.textView) as TextView
}

        init {
```

```
            view.setOnClickListener { v -> itemClickStream.onNext(v) }
        }

        fun bind(text: String) {
            textView.text = text
        }
    }
}
```

L'utilisation est assez simple alors. Il est possible de s'inscrire sur des threads distincts en utilisant les fonctionnalités de RxJava.

```
val adapter = SampleAdapter(arrayOf("Hello", "World"))
adapter.itemClickStream.subscribe { v ->
    if (v.id == R.id.textView) {
        // do something
    }
}
```

Easy OnLongClick et OnClick Exemple

Tout d'abord, implémentez votre support d'affichage:

```
implements View.OnClickListener, View.OnLongClickListener
```

Enregistrez ensuite les écouteurs comme suit:

```
itemView.setOnClickListener(this);
itemView.setOnLongClickListener(this);
```

Ensuite, remplacez les écouteurs comme suit:

```
@Override
public void onClick(View v) {
    onclicklistner.onItemClick(getAdapterPosition(), v);
}

@Override
public boolean onLongClick(View v) {
    onclicklistner.onItemLongClick(getAdapterPosition(), v);
    return true;
}
```

Et enfin, ajoutez le code suivant:

```
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);
}
```

Démo de l'adaptateur

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

L'exemple de code ci-dessus peut être appelé par le code suivant:

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

Item Cliquez sur les auditeurs

Pour implémenter un écouteur de clic d'élément et / ou un écouteur de clic long, vous pouvez créer une interface dans votre adaptateur:

```
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;
    }

}
```

Vous pouvez ensuite définir vos écouteurs de clic après avoir créé une instance de l'adaptateur:

```
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;
    }
});
```

Une autre façon d'implémenter un écouteur d'élément

Une autre façon d'implémenter un écouteur de clic d'éléments consiste à utiliser l'interface avec plusieurs méthodes, dont le nombre est égal au nombre de vues cliquables, et à utiliser des programmes d'écoute de clics comme vous pouvez le voir ci-dessous. Cette méthode est plus flexible, car vous pouvez définir des écouteurs de clic sur différentes vues et contrôler facilement la logique des clics séparément pour chacun.

```
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);
        }
    }
}
```

Et lorsque vous avez une instance d'adaptateur, vous pouvez définir votre écouteur de clic qui écoute en cliquant sur chacune des vues:

```
customAdapter.setClickInterface(new CustomAdapter.ClickInterface {
    @Override
    public void clickEventOne(Object obj) {
        // Your implementation here
    }
    @Override
    public void clickEventTwo(Object obj1, Object obj2) {
        // Your implementation here
    }
});
```

RecyclerView Click auditeur

```
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);
    }
}
```

Dans MainActivity

```
RecyclerView recyclerView =(RecyclerView) findViewById(R.id.recyclerview);
 recyclerView.addOnItemTouchListener(new RecyclerTouchListener(getActivity(),recyclerView, new
RecyclerTouchListener.ClickListener() {
      @Override
      public void onLongClick(View child, int childPosition) {
```

```
      }

      @Override
      public void onClick(View child, int childPosition) {


     }
    }));
```

Lire RecyclerView onClickListeners en ligne:
https://riptutorial.com/fr/android/topic/96/recyclerview-onclicklisteners

Chapitre 213: RenderScript

Introduction

RenderScript est un langage de script qui vous permet d'écrire un rendu graphique haute performance et un code de calcul brut. Il fournit un moyen d'écrire du code critique de performance que le système compile ultérieurement en code natif pour le processeur sur lequel il peut être exécuté. Cela pourrait être le CPU, un CPU multi-core, ou même le GPU. Cela dépend de nombreux facteurs qui ne sont pas facilement accessibles au développeur, mais dépend également de l'architecture prise en charge par le compilateur interne.

Examples

Commencer

RenderScript est un framework permettant des calculs parallèles hautes performances sur Android. Les scripts que vous écrivez seront exécutés en parallèle sur tous les processeurs disponibles (par ex. CPU, GPU, etc.), ce qui vous permettra de vous concentrer sur la tâche que vous souhaitez réaliser au lieu de la planifier et de l'exécuter.

Les scripts sont écrits dans un langage C99 (C99 étant une ancienne version du standard du langage de programmation C). Pour chaque script, une classe Java est créée qui vous permet d'interagir facilement avec RenderScript dans votre code Java.

Mise en place de votre projet

Il existe deux manières différentes d'accéder à RenderScript dans votre application, avec les bibliothèques Android Framework ou la bibliothèque de support. Même si vous ne souhaitez pas cibler les périphériques avant le niveau 11 de l'API, vous devez toujours utiliser l'implémentation de la bibliothèque de support car elle garantit la compatibilité des périphériques sur de nombreux périphériques différents. Pour utiliser l'implémentation de la bibliothèque de support, vous devez utiliser au moins les outils de construction version `18.1.0` !

Configurez maintenant le fichier build.gradle de votre application:

```
android {
    compileSdkVersion 24
    buildToolsVersion '24.0.1'

    defaultConfig {
        minSdkVersion 8
        targetSdkVersion 24

        renderscriptTargetApi 18
        renderscriptSupportModeEnabled true
    }
```

```
}
```

- `renderscriptTargetApi` : Cela doit être défini sur le niveau de l'API le plus ancien de la version qui fournit toutes les fonctionnalités RenderScript dont vous avez besoin.
- `renderscriptSupportModeEnabled` : active l'utilisation de l'implémentation RenderScript de la bibliothèque de support.

Comment fonctionne RenderScript

Un script Render typique se compose de deux éléments: les noyaux et les fonctions. Une fonction est exactement ce que cela ressemble - elle accepte une entrée, fait quelque chose avec cette entrée et renvoie une sortie. Un noyau est l'endroit d'où vient la puissance réelle de RenderScript.

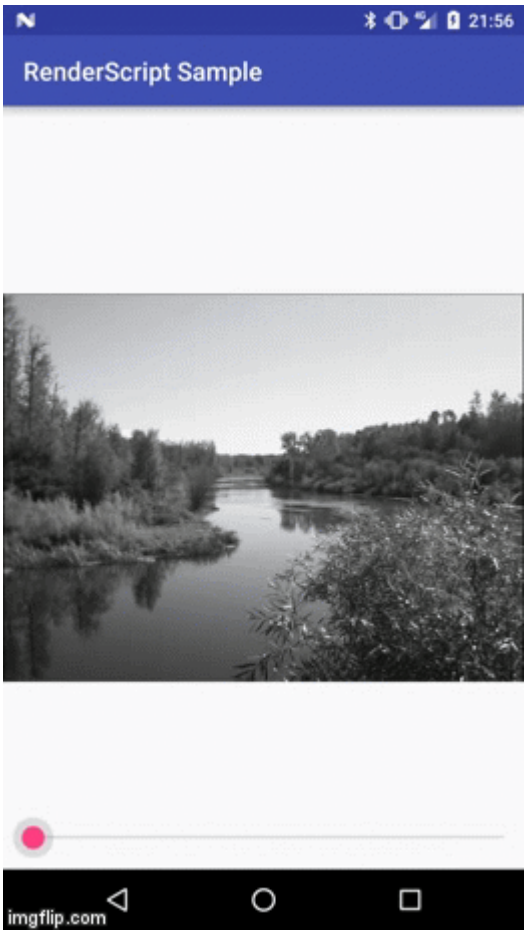
Un noyau est une fonction exécutée sur chaque élément d'une `Allocation` . Une `Allocation` peut être utilisée pour transmettre des données comme un `Bitmap` ou un tableau d' `byte` à un `RenderScript` et elles sont également utilisées pour obtenir un résultat d'un noyau. Les noyaux peuvent soit prendre une `Allocation` en entrée et une autre en sortie, soit modifier les données dans une seule `Allocation` .

Vous pouvez écrire vos noyaux, mais il existe également de nombreux noyaux prédéfinis que vous pouvez utiliser pour effectuer des opérations courantes telles qu'un flou d'image gaussien.

Comme déjà mentionné pour chaque fichier RenderScript, une classe est générée pour interagir avec elle. Ces classes commencent toujours par le préfixe `ScriptC_` suivi du nom du fichier RenderScript. Par exemple, si votre fichier RenderScript est appelé `example` la classe Java générée s'appellera `ScriptC_example` . Tous les scripts prédéfinis commencent par le préfixe `Script` - par exemple, le script gaussien de l'image est appelé `ScriptIntrinsicBlur` .

Écrire votre premier script RenderScript

L'exemple suivant est basé sur un exemple sur GitHub. Il effectue une manipulation d'image de base en modifiant la saturation d'une image. Vous pouvez trouver le code source **ici** et le vérifier si vous voulez jouer avec lui-même. Voici un aperçu rapide de ce à quoi le résultat est censé ressembler:



RenderScript Boilerplate

Les fichiers RenderScript résident dans le dossier `src/main/rs` de votre projet. Chaque fichier a l'extension de fichier `.rs` et doit contenir deux instructions `#pragma` en haut:

```
#pragma version(1)
#pragma rs java_package_name(your.package.name)
```

- `#pragma version(1)` : Ceci peut être utilisé pour définir la version de RenderScript que vous utilisez. Actuellement, il n'y a que la version 1.

- `#pragma rs java_package_name(your.package.name)` : Ceci peut être utilisé pour définir le nom du paquet de la classe Java générée pour interagir avec ce RenderScript particulier.

Il y a un autre `#pragma` vous devriez normalement définir dans chacun de vos fichiers RenderScript et il est utilisé pour définir la précision en virgule flottante. Vous pouvez définir la précision en virgule flottante sur trois niveaux différents:

- `#pragma rs_fp_full` : C'est le paramètre le plus strict avec la plus grande précision et c'est aussi la valeur par défaut si rien n'est spécifié. Vous devriez l'utiliser si vous avez besoin d'une précision élevée en virgule flottante.
- `#pragma rs_fp_relaxed` : Ceci garantit non seulement une précision en virgule flottante aussi élevée, mais sur certaines architectures cela permet un tas d'optimisations qui peuvent accélérer l'exécution de vos scripts.

- `#pragma rs_fp_imprecise` : Ceci garantit encore moins de précision et devrait être utilisé si la précision en virgule flottante n'a pas vraiment d'importance pour votre script.

La plupart des scripts peuvent simplement utiliser `#pragma rs_fp_relaxed` sauf si vous avez vraiment besoin d'une précision élevée en virgule flottante.

Variables globales

Maintenant, tout comme en code C, vous pouvez définir des variables globales ou des constantes:

```
const static float3 gMonoMult = {0.299f, 0.587f, 0.114f};

float saturationLevel = 0.0f;
```

La variable `gMonoMult` est de type `float3` . Cela signifie que c'est un vecteur composé de 3 nombres flottants. L'autre variable `float` appelée `saturationValue` n'est pas constante, vous pouvez donc la définir à l'exécution à une valeur qui vous plait. Vous pouvez utiliser des variables comme celle-ci dans vos noyaux ou fonctions et elles constituent donc un autre moyen de fournir des entrées ou des sorties à vos scripts Render. Pour chaque variable non constante, une méthode getter et setter sera générée sur la classe Java associée.

Graines

Mais maintenant, commençons à implémenter le noyau. Pour les besoins de cet exemple, je ne vais pas expliquer les calculs utilisés dans le noyau pour modifier la saturation de l'image, mais plutôt comment implémenter un noyau et comment l'utiliser. À la fin de ce chapitre, j'expliquerai rapidement ce que fait le code dans ce noyau.

Noyaux en général

Regardons d'abord le code source:

```
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);
}
```

Comme vous pouvez le voir, cela ressemble à une fonction C normale avec une exception: le `__attribute__((kernel))` entre le type de retour et le nom de la méthode. C'est ce qui dit à RenderScript que cette méthode est un noyau. Une autre chose que vous pouvez remarquer est que cette méthode accepte un paramètre `uchar4` et renvoie une autre valeur `uchar4` . `uchar4` est - comme la variable `float3` nous avons parlé dans le chapitre précédent - un vecteur. Il contient 4 valeurs `uchar` qui ne sont que des valeurs d'octet comprises entre 0 et 255.

Vous pouvez accéder à ces valeurs individuelles de différentes manières, par exemple, `in.r` renverrait l'octet qui correspond au canal rouge d'un pixel. Nous utilisons un `uchar4` puisque

chaque pixel est composé de 4 valeurs - `r` pour le rouge, `g` pour le vert, `b` pour le bleu et `a` pour alpha - et vous pouvez y accéder avec ce raccourci. RenderScript vous permet également de prendre n'importe quel nombre de valeurs d'un vecteur et de créer un autre vecteur avec elles. Par exemple, `in.rgb` renverrait une valeur `uchar3` contenant uniquement les parties rouge, verte et bleue du pixel sans la valeur alpha.

A l'exécution, RenderScript appellera cette méthode du noyau pour chaque pixel d'une image, ce qui explique pourquoi la valeur de retour et le paramètre ne représentent qu'une seule valeur `uchar4` . RenderScript exécutera plusieurs de ces appels en parallèle sur tous les processeurs disponibles, raison pour laquelle RenderScript est si puissant. Cela signifie également que vous n'avez pas à vous soucier des threads ou de la sécurité des threads, vous pouvez simplement implémenter ce que vous voulez faire pour chaque pixel et RenderScript se charge du reste.

Lorsque vous appelez un noyau en Java, vous devez fournir deux variables d' `Allocation` , l'une qui contient les données en entrée et l'autre qui reçoit la sortie. Votre méthode du noyau sera appelée pour chaque valeur de l' `Allocation` entrée et écrira le résultat à l' `Allocation` sortie.

Méthodes de RenderScript Runtime API

Dans le noyau ci-dessus quelques méthodes sont utilisées qui sont fournies hors de la boîte. RenderScript fournit beaucoup de ces méthodes et elles sont essentielles pour presque tout ce que vous allez faire avec RenderScript. Parmi celles-ci, il y a des méthodes pour effectuer des opérations mathématiques comme `sin()` et des méthodes auxiliaires comme `mix()` qui mélange deux valeurs selon d'autres valeurs. Mais il existe également des méthodes pour des opérations plus complexes avec des vecteurs, des quaternions et des matrices.

La **référence** officielle de **RenderScript Runtime API** est la meilleure ressource disponible si vous souhaitez en savoir plus sur une méthode particulière ou si vous recherchez une méthode spécifique qui effectue une opération commune telle que le calcul du produit scalaire d'une matrice. Vous pouvez trouver cette documentation **ici** .

Implémentation du noyau

Voyons maintenant les spécificités de ce que fait le noyau. Voici la première ligne du noyau:

```
float4 f4 = rsUnpackColor8888(in);
```

La première ligne appelle la méthode **`rsUnpackColor8888()`** qui transforme la valeur `uchar4` valeur `float4` . Chaque canal de couleur est également transformé dans la plage `0.0f - 1.0f` où `0.0f` correspond à une valeur d'octet de `0` et `1.0f` à `255` . Le but principal de cette opération est de simplifier tous les calculs dans ce noyau.

```
float3 dotVector = dot(f4.rgb, gMonoMult);
```

Cette ligne suivante utilise la méthode intégrée **`dot()`** pour calculer le produit scalaire de deux vecteurs. `gMonoMult` est une valeur constante que nous avons définie ci-dessus. Puisque les deux vecteurs doivent avoir la même longueur pour calculer le produit scalaire et aussi parce que nous

voulons simplement affecter les canaux de couleur et non le canal alpha d'un pixel, nous utilisons le raccourci `.rgb` pour obtenir un nouveau vecteur `float3` canaux de couleur rouge, vert et bleu. Ceux d'entre nous qui se souviennent encore de la façon dont fonctionne le produit scalaire remarqueront rapidement que le produit scalaire ne devrait renvoyer qu'une valeur et non un vecteur. Pourtant, dans le code ci-dessus, nous `float3` le résultat à un vecteur `float3` . Ceci est encore une fonctionnalité de RenderScript. Lorsque vous attribuez un numéro à une dimension à un vecteur, tous les éléments du vecteur seront définis sur cette valeur. Par exemple, l'extrait suivant attribue `2.0f` à chacune des trois valeurs du vecteur `float3` :

```
float3 example = 2.0f;
```

Ainsi, le résultat du produit scalaire ci-dessus est attribué à chaque élément du vecteur `float3` ci-dessus.

Maintenant vient la partie dans laquelle nous utilisons effectivement la variable globale `saturationLevel` pour modifier la saturation de l'image:

```
float3 newColor = mix(dotVector, f4.rgb, saturationLevel);
```

Cela utilise la méthode intégrée **`mix()`** pour mélanger la couleur originale avec le vecteur de produit scalaire que nous avons créé ci-dessus. La façon dont ils sont mélangés est déterminée par la variable globale `saturationLevel` . Ainsi, un `saturationLevel` de `0.0f` ne donnera aucune partie des valeurs de couleur d'origine à la couleur résultante et consistera uniquement en des valeurs dans le `dotVector` ce qui `dotVector` une image en noir et blanc ou grisée. Une valeur de `1.0f` fera que la couleur résultante sera complètement composée des valeurs de couleur d'origine et que les valeurs supérieures à `1.0f` multiplieront les couleurs d'origine pour les rendre plus lumineuses et plus intenses.

```
return rsPackColorTo8888(newColor);
```

C'est la dernière partie du noyau. **`rsPackColorTo8888()`** transforme le vecteur `float3` en une valeur `uchar4` qui est ensuite renvoyée. Les valeurs d'octet résultantes sont bloquées dans une plage comprise entre 0 et 255; les valeurs flottantes supérieures à `1.0f` entraînent une valeur d'octet de 255 et des valeurs inférieures à `0.0` entraînent une valeur d'octet de `0` .

Et c'est toute l'implémentation du noyau. Il ne reste qu'une partie: comment appeler un noyau en Java.

Appeler RenderScript en Java

Les bases

Comme cela a déjà été mentionné ci-dessus pour chaque fichier RenderScript, une classe Java est générée qui vous permet d'interagir avec vos scripts. Ces fichiers ont le préfixe `ScriptC_` suivi du nom du fichier RenderScript. Pour créer une instance de ces classes, vous devez d'abord

disposer d’une instance de la classe `RenderScript` :

```
final RenderScript renderScript = RenderScript.create(context);
```

La méthode statique `create()` peut être utilisée pour créer une occurrence `RenderScript` partir d'un `Context` . Vous pouvez ensuite instancier la classe Java générée pour votre script. Si vous appelez le fichier RenderScript `saturation.rs` alors la classe s'appellera `ScriptC_saturation` :

```
final ScriptC_saturation script = new ScriptC_saturation(renderScript);
```

Sur cette classe, vous pouvez maintenant définir le niveau de saturation et appeler le noyau. Le setter qui a été généré pour la variable `saturationLevel` aura le préfixe `set_` suivi du nom de la variable:

```
script.set_saturationLevel(1.0f);
```

Il y a aussi un getter préfixé par `get_` qui vous permet d'obtenir le niveau de saturation actuellement défini:

```
float saturationLevel = script.get_saturationLevel();
```

Les noyaux que vous définissez dans votre script RenderScript sont préfixés par `forEach_` suivi du nom de la méthode du noyau. Le noyau que nous avons écrit attend une `Allocation` entrée et une `Allocation` sortie comme paramètres:

```
script.forEach_saturation(inputAllocation, outputAllocation);
```

L' `Allocation` entrée doit contenir l'image d'entrée et, une fois la méthode `forEach_saturation` terminée, l'allocation de sortie contiendra les données d'image modifiées.

Une fois que vous avez une instance `Allocation` , vous pouvez copier des données depuis et vers ces `Allocations` en utilisant les méthodes `copyFrom()` et `copyTo()` . Par exemple, vous pouvez copier une nouvelle image dans votre entrée `Allocation en appelant:

```
inputAllocation.copyFrom(inputBitmap);
```

De la même façon, vous pouvez récupérer l'image de résultat en appelant `copyTo()` sur la sortie `Allocation` :

```
outputAllocation.copyTo(outputBitmap);
```

Création d'instances d'allocation

Il existe plusieurs façons de créer une `Allocation` . Une fois que vous avez une instance `Allocation` , vous pouvez copier les nouvelles données depuis et vers ces `Allocations` avec `copyTo()` et `copyFrom()` comme expliqué ci-dessus, mais pour les créer initialement, vous devez savoir avec

quel type de données vous travaillez. Commençons par l' `Allocation` entrée:

Nous pouvons utiliser la méthode statique `createFromBitmap()` pour créer rapidement notre `Allocation` entrée à partir d'un `Bitmap` :

```
final Allocation inputAllocation = Allocation.createFromBitmap(renderScript, image);
```

Dans cet exemple, l'image d'entrée ne change jamais, nous n'avons donc jamais besoin de modifier l' `Allocation` entrée. Nous pouvons le réutiliser à chaque fois que `saturationLevel` change pour créer un nouveau `Bitmap` sortie.

Création de la sortie L' `Allocation` est un peu plus complexe. Nous devons d'abord créer ce qu'on appelle un `Type` . Un `Type` est utilisé pour indiquer à une `Allocation` avec quel type de données il a affaire. Généralement, on utilise la classe `Type.Builder` pour créer rapidement un `Type` approprié. Regardons d'abord le code:

```
final Type outputType = new Type.Builder(renderScript, Element.RGBA_8888(renderScript))
        .setX(inputBitmap.getWidth())
        .setY(inputBitmap.getHeight())
        .create();
```

Nous travaillons avec un `Bitmap` normal de 32 bits (ou 4 octets) par pixel avec 4 canaux de couleur. C'est pourquoi nous choisissons `Element.RGBA_8888` pour créer le `Type` . Ensuite, nous utilisons les méthodes `setX()` et `setY()` pour définir la largeur et la hauteur de l'image de sortie sur la même taille que l'image d'entrée. La méthode `create()` crée alors le `Type` avec les paramètres que nous avons spécifiés.

Une fois que nous avons le bon `Type` nous pouvons créer la sortie `Allocation` avec la méthode statique `createTyped()` :

```
final Allocation outputAllocation = Allocation.createTyped(renderScript, outputType);
```

Maintenant, nous avons presque terminé. Nous avons également besoin d'un `Bitmap` sortie dans lequel nous pouvons copier les données de l' `Allocation` sortie. Pour ce faire, nous utilisons la méthode statique `createBitmap()` pour créer un nouveau `Bitmap` vide ayant la même taille et la même configuration que le `Bitmap` entrée.

```
final Bitmap outputBitmap = Bitmap.createBitmap(
        inputBitmap.getWidth(),
        inputBitmap.getHeight(),
        inputBitmap.getConfig()
);
```

Et avec cela, nous avons toutes les pièces du puzzle pour exécuter notre script Render.

Exemple complet

Maintenant, mettons tout cela ensemble dans un exemple:

```
// 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);
```

Conclusion

Avec cette introduction, vous devriez être prêt à écrire vos propres noyaux RenderScript pour une manipulation simple des images. Cependant, il y a quelques choses à garder à l'esprit:

- **RenderScript ne fonctionne que dans les projets d'application** : Actuellement, les fichiers RenderScript ne peuvent pas faire partie d'un projet de bibliothèque.
- **Attention à la mémoire** : RenderScript est très rapide, mais il peut également nécessiter beaucoup de mémoire. Il ne devrait jamais y avoir plus d'une instance de `RenderScript` à tout moment. Vous devriez également réutiliser autant que possible. Normalement, il vous suffit de créer vos instances d' `Allocation` une fois et de les réutiliser ultérieurement. Il en va de même pour les `Bitmaps` sortie ou vos instances de script. Réutiliser autant que possible.
- **Faites votre travail en arrière-plan** : à nouveau, RenderScript est très rapide, mais pas instantané. Tout noyau, en particulier les noyaux complexes, devrait être exécuté à partir du thread d'interface utilisateur dans un `AsyncTask` ou quelque chose de similaire. Cependant, la plupart du temps, vous n'avez pas à vous soucier des fuites de mémoire. Toutes les classes liées à RenderScript utilisent uniquement le `Context` application et ne provoquent donc pas

de fuite de mémoire. Mais vous devez toujours vous inquiéter des choses habituelles comme la fuite de `View` , `Activity` ou de toute instance de `Context` que vous utilisez vous-même!

- **Utilisez des éléments intégrés** : il existe de nombreux scripts prédéfinis qui effectuent des tâches telles que le flou, la fusion, la conversion et le redimensionnement des images. Et il y a beaucoup d'autres méthodes intégrées qui vous aident à implémenter vos noyaux. Les chances sont que si vous voulez faire quelque chose, il existe soit un script ou une méthode qui fait déjà ce que vous essayez de faire. Ne réinventez pas la roue.

Si vous voulez rapidement commencer à jouer avec du code, je vous recommande de regarder l'exemple de projet GitHub qui implémente l'exemple exact dont il est question dans ce tutoriel. Vous pouvez trouver le projet **ici** . Amusez-vous avec RenderScript!

Flou une image

Cet exemple montre comment utiliser l'API Renderscript pour flouter une image (en utilisant Bitmap). Cet exemple utilise ScriptInstrinsicBlur fourni par l'API Android Renderscript (API> = 17).

```
public class BlurProcessor {

    private RenderScript rs;
    private Allocation inAllocation;
    private Allocation outAllocation;
    private int width;
    private int height;

    private ScriptIntrinsicBlur blurScript;

    public BlurProcessor(RenderScript rs) {
        this.rs = rs;
    }

    public void initialize(int width, int height) {
        blurScript = ScriptIntrinsicBlur.create(rs, Element.U8_4(rs));
        blurScript.setRadius(7f); // Set blur radius. 25 is max

        if (outAllocation != null) {
            outAllocation.destroy();
            outAllocation = null;
        }

        // Bitmap must have ARGB_8888 config for this type
        Type bitmapType = new Type.Builder(rs, Element.RGBA_8888(rs))
            .setX(width)
            .setY(height)
            .setMipmaps(false) // We are using MipmapControl.MIPMAP_NONE
            .create();

        // Create output allocation
        outAllocation = Allocation.createTyped(rs, bitmapType);

        // Create input allocation with same type as output allocation
        inAllocation = Allocation.createTyped(rs, bitmapType);
    }

    public void release() {
```

```
        if (blurScript != null) {
            blurScript.destroy();
            blurScript = null;
        }

        if (inAllocation != null) {
            inAllocation.destroy();
            inAllocation = null;
        }

        if (outAllocation != null) {
            outAllocation.destroy();
            outAllocation = null;
        }
    }

    public Bitmap process(Bitmap bitmap, boolean createNewBitmap) {
        if (bitmap.getWidth() != width || bitmap.getHeight() != height) {
            // Throw error if required
            return null;
        }

        // Copy data from bitmap to input allocations
        inAllocation.copyFrom(bitmap);

        // Set input for blur script
        blurScript.setInput(inAllocation);

        // process and set data to the output allocation
        blurScript.forEach(outAllocation);

        if (createNewBitmap) {
            Bitmap returnVal = Bitmap.createBitmap(width, height, Bitmap.Config.ARGB_8888);
            outAllocation.copyTo(returnVal);
            return returnVal;
        }

        outAllocation.copyTo(bitmap);
        return bitmap;
    }
}
```

Chaque script a un noyau qui traite les données et il est généralement `forEach` via la méthode `forEach` .

```
public class BlurActivity extends AppCompatActivity {
    private BlurProcessor blurProcessor;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        // setup layout and other stuff


        blurProcessor = new BlurProcessor(Renderscript.create(getApplicationContext()));
    }

    private void loadImage(String path) {
        // Load image to bitmap
        Bitmap bitmap = loadBitmapFromPath(path);
```

```
        // Initialize processor for this bitmap
        blurProcessor.release();
        blurProcessor.initialize(bitmap.getWidth(), bitmap.getHeight());

        // Blur image
        Bitmap blurImage = blurProcessor.process(bitmap, true); // Use newBitamp as false if
you don't want to create a new bitmap
    }
}
```

Ceci a conclu l'exemple ici. Il est conseillé de faire le traitement dans un thread d'arrière-plan.

Flou une vue

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;
    }
}
```

Usage:

```
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);
```

Lire RenderScript en ligne: https://riptutorial.com/fr/android/topic/5214/renderscript

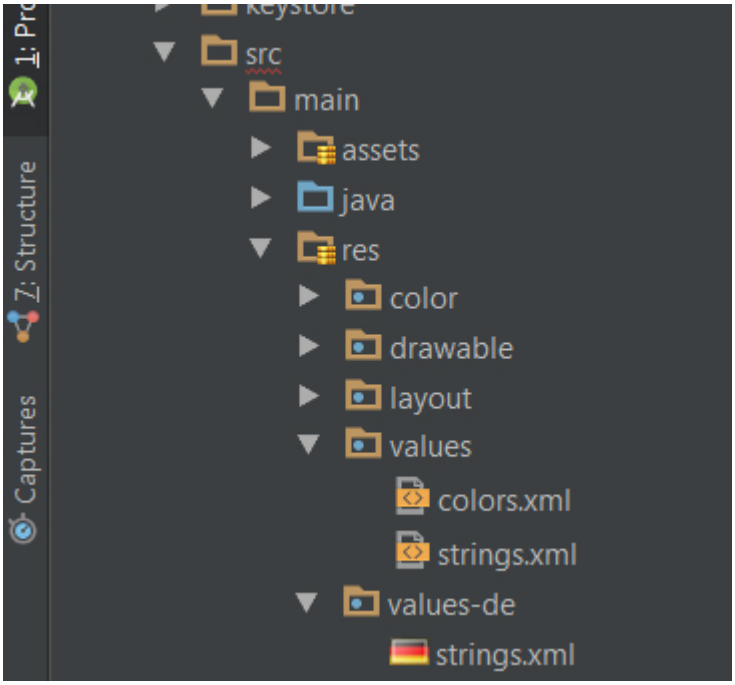
Chapitre 214: Ressources

Examples

Traduire une chaîne

Les chaînes peuvent être internationalisées en définissant un fichier strings.xml différent pour chaque langue prise en charge.

Vous ajoutez une nouvelle langue en créant un nouveau répertoire de valeurs avec le code de langue ISO comme suffixe. Par exemple, lors de l'ajout d'un ensemble allemand, votre structure peut ressembler à ceci:



Lorsque le système recherche la chaîne demandée, il vérifie d'abord le fichier XML spécifique à la langue. S'il n'est pas trouvé, la valeur du fichier strings.xml par défaut est renvoyée. La clé reste la même pour chaque langue et seule la valeur change.

Exemple de contenu:

/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>
```

Définir des chaînes

Les chaînes sont généralement stockées dans le fichier de ressources `strings.xml` . Ils sont définis à l'aide d'un élément XML `<string>` .

Strings.xml a pour but de permettre l'internationalisation. Vous pouvez définir un fichier strings.xml pour chaque code iso de la langue. Ainsi, lorsque le système recherche la chaîne 'nom_application', il vérifie d'abord le fichier xml correspondant à la langue actuelle et, s'il n'est pas trouvé, recherche l'entrée dans le fichier strings.xml par défaut. Cela signifie que vous pouvez choisir de ne localiser que certaines de vos chaînes alors que d'autres ne le sont pas.

/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>
```

Une fois qu'une chaîne est définie dans un fichier de ressources XML, elle peut être utilisée par d'autres parties de l'application.

Les fichiers de projet XML d'une application peuvent utiliser un élément `<string>` en faisant référence à `@string/string_name` . Par exemple, le fichier de manifeste d' une application (/manifests/AndroidManifest.xml) inclut la ligne suivante par défaut dans Android Studio:

```
android:label="@string/app_name"
```

Cela dit à Android de rechercher une ressource `<string>` appelée "app_name" à utiliser comme nom de l'application lorsqu'elle est installée ou affichée dans un lanceur.

Une autre fois, vous utiliseriez une ressource `<string>` partir d'un fichier XML dans android dans un fichier de mise en page. Par exemple, ce qui suit représente un TextView qui affiche la chaîne `hello_world` définie précédemment:

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/hello_world"/>
```

Vous pouvez également accéder aux ressources `<string>` partir de la partie Java de votre application. Pour rappeler notre même chaîne `hello_world` de dessus dans une classe Activity, utilisez:

```
String helloWorld = getString(R.string.hello_world);
```

Définir un tableau de chaînes

Pour définir un tableau de chaînes, écrivez dans un fichier de ressources

res / values / filename.xml

```
<string-array name="string_array_name">
    <item>text_string</item>
    <item>@string/string_id</item>
</string-array>
```

par exemple

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>
```

et l'utiliser de java comme

```
String[] strings = getResources().getStringArray(R.array.string_array_example;
Log.i("TAG",Arrays.toString(strings)));
```

Sortie

```
I/TAG: [HelloWorld, Hello World!]
```

Définir les dimensions

Les dimensions sont généralement stockées dans un fichier de ressources `dimens.xml` . Ils sont définis à l'aide d'un élément `<dimen>` .

res / values / dimens.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <dimen name="small_padding">5dp</dimen>
    <dimen name="medium_padding">10dp</dimen>
    <dimen name="large_padding">20dp</dimen>

    <dimen name="small_font">14sp</dimen>
    <dimen name="medium_font">16sp</dimen>
    <dimen name="large_font">20sp</dimen>
</resources>
```

Vous pouvez utiliser différentes unités:

- **sp:** Pixels indépendants de l'échelle. Pour les polices
- **dp:** Pixels indépendants de la densité. Pour tout le reste.
- **pt:** Points
- **px:** Pixels
- **mm:** millimètres
- **im:** pouces

Les dimensions peuvent maintenant être référencées en XML avec la syntaxe `@dimen/name_of_the_dimension` .

Par exemple:

```
<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>
```

Définir des entiers

Les entiers sont généralement stockés dans un fichier de ressources nommé `integers.xml` , mais le nom du fichier peut être choisi de manière arbitraire. Chaque entier est défini à l'aide d'un élément `<integer>` , comme indiqué dans le fichier suivant:

res / values / integers.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <integer name="max">100</integer>
</resources>
```

Les entiers peuvent maintenant être référencés en XML avec la syntaxe `@integer/name_of_the_integer` , comme illustré dans l'exemple suivant:

```
<ProgressBar
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:max="@integer/max"/>
```

Définir un tableau d'entiers

Pour définir un tableau entier, écrivez dans un fichier de ressources

res / values / filename.xml

```
<integer-array name="integer_array_name">
    <item>integer_value</item>
    <item>@integer/integer_id</item>
```

```
</integer-array>
```

par exemple

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>
```

et l'utiliser de java comme

```
int[] values = getResources().getIntArray(R.array.fibo);
Log.i("TAG",Arrays.toString(values)));
```

Sortie

```
I/TAG: [0, 1, 1, 2, 3, 5]
```

Définir les couleurs

Les couleurs sont généralement stockées dans un fichier de ressources nommé `colors.xml` dans le dossier `/res/values/` .

Ils sont définis par `<color>` éléments `<color>` :

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#3F51B5</color>
    <color name="colorPrimaryDark">#303F9F</color>
    <color name="colorAccent">#FF4081</color>

    <color name="blackOverlay">#66000000</color>
</resources>
```

Les couleurs sont représentées par des valeurs de couleur hexadécimales pour chaque canal de couleur `(0 - FF)` dans l'un des formats:

- `#RGB`
- `#ARGB`
- `#RRGGBB`
- `#AARRGGBB`

Légende

- A - canal alpha - la valeur 0 est totalement transparente, la valeur FF est opaque
- R - canal rouge
- G - canal vert
- B - canal bleu

Les couleurs définies peuvent être utilisées en XML avec la syntaxe suivante
`@color/name_of_the_color`

Par exemple:

```
<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/blackOverlay">
```

Utiliser des couleurs dans le code

Ces exemples supposent `this` s'agit d'une référence d'activité. Une référence de contexte peut également être utilisée à sa place.

1.6

```
int color = ContextCompat.getColor(this, R.color.black_overlay);
view.setBackgroundColor(color);
```

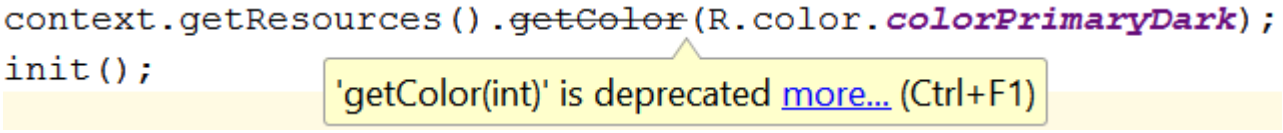
6,0

```
int color = this.getResources().getColor(this, R.color.black_overlay);
view.setBackgroundColor(color);
```

Dans la déclaration ci-dessus, `colorPrimary` , `colorPrimaryDark` et `colorAccent` sont utilisés pour définir les couleurs de conception de matériaux qui seront utilisées pour définir un thème Android personnalisé dans `styles.xml` . Ils sont automatiquement ajoutés lorsqu'un nouveau projet est créé avec Android Studio.

Obtenir des ressources sans avertissements "obsolètes"

En utilisant l'API Android 23 ou supérieure, très souvent, une telle situation peut être observée:



Cette situation est due au changement structurel de l'API Android concernant l'obtention des ressources.
Maintenant la fonction:

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

Devrait être utilisé. Mais la bibliothèque `android.support.v4` a une autre solution.

Ajoutez la dépendance suivante au fichier `build.gradle` :

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

Ensuite, toutes les méthodes de la bibliothèque de support sont disponibles:

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

De plus, d'autres méthodes de la bibliothèque de support peuvent être utilisées:

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

Définir une ressource de menu et l'utiliser à l'intérieur d'Activité / Fragment

Définir un menu dans **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>
```

Pour plus d'options de configuration, reportez-vous à: Ressource de menu

`Activity` intérieure:

```
@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);
}
```

Pour appeler les méthodes ci-dessus pendant l'affichage de la vue, appelez `getActivity().invalidateOptionsMenu();`

Inside **`Fragment`** un appel supplémentaire est nécessaire:

```
@Nullable
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
    setHasOptionsMenu(true);
    super.onCreateView(inflater, container, savedInstanceState);
}
```

Formatage de chaîne dans strings.xml

La définition de chaînes dans le fichier strings.xml permet également le formatage des chaînes. Le seul inconvénient est que la chaîne devra être traitée dans le code ci-dessous, par opposition à la simple fixation à une mise en page.

```
<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);
```

Dans l'exemple ci-dessus,
% 1 $ s
'%' se sépare des caractères normaux,
«1» indique le premier paramètre,
'$' est utilisé comme séparateur entre le numéro de paramètre et le type,
's' dénote le type de chaîne ('d' est utilisé pour l'entier)

Notez que `getString()` est une méthode de `Context` ou `Resources` , c'est-à-dire que vous pouvez

l'utiliser directement dans une instance `Activity` , ou bien utiliser `getActivity().getString()` ou `getContext().getString()` respectivement.

Définir une liste d'états de couleurs

Les listes d'états de couleurs peuvent être utilisées comme couleurs, mais changeront en fonction de l'état de la vue pour laquelle elles sont utilisées.

Pour en définir un, créez un fichier de ressources dans `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>
```

Les éléments sont évalués dans l'ordre où ils sont définis et le premier élément dont les états spécifiés correspondent à l'état actuel de la vue est utilisé. Il est donc recommandé de spécifier un catch-all à la fin, sans aucun sélecteur d'état spécifié.

Chaque élément peut utiliser un littéral de couleur ou faire référence à une couleur définie ailleurs.

Définir des cordes à cordes

Pour différencier les chaînes plurielles et singulières, vous pouvez définir un pluriel dans votre *fichier strings.xml* et répertorier les différentes quantités, comme illustré dans l'exemple ci-dessous:

```
<?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>
```

Cette définition est accessible à partir du code Java à l'aide de la méthode `getQuantityString()` de la classe `Resources` , comme illustré dans l'exemple suivant:

```
getResources().getQuantityString(R.plurals.hello_people, 3, 3);
```

Ici, le premier paramètre `R.plurals.hello_people` est le nom de la ressource. Le deuxième paramètre (`3` dans cet exemple) est utilisé pour sélectionner la chaîne de `quantity` correcte. Le troisième paramètre (également `3` dans cet exemple) est l'argument de format qui sera utilisé pour remplacer le spécificateur de format `%d` .

Les valeurs de quantité possibles (répertoriées par ordre alphabétique) sont les suivantes:

```
few
```

```
many
one
other
two
zero
```

Il est important de noter que tous les paramètres régionaux ne prennent pas en charge chaque dénomination de `quantity` . Par exemple, la langue chinoise n'a pas de concept d' `one` élément. L'anglais n'a pas d'élément `zero` , car il est grammaticalement identique aux `other` . Les instances de `quantity` non prises en charge seront signalées par l'EDI en tant qu'avertissements Lint, mais ne provoqueront pas d'erreurs de complication si elles sont utilisées.

Importer un tableau d'objets définis dans les ressources

Il existe des cas où des objets personnalisés doivent être créés et définis dans les ressources de l'application. De tels objets peuvent être composés de types `Java` simples, par exemple `Integer` , `Float` , `String` .

Voici l'exemple de l'importation d'un objet défini dans les ressources de l'application. L'objet `Category` contient 3 propriétés de la catégorie:

- ID
- Couleur
- prénom

Ce `POJO` a son équivalent dans le fichier `categories.xml` , où chaque tableau a les mêmes propriétés définies pour chaque catégorie.

1. **Créez un modèle pour votre objet:**

```
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. **Créez le fichier dans le dossier `res/values` :**

```
categories.xml
```

3. **Composez chaque modèle composé de ressources:**

```
<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. **Définissez un tableau dans le fichier de ressources:**

```
<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. **Créez une fonction pour les importer:**

```
@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 patchs

9 Les patchs sont des images **extensibles** dans lesquelles les zones pouvant être étirées sont définies par des marqueurs noirs sur une bordure transparente.

Il y a un excellent tutoriel ici .
En dépit d'être si vieux, c'est toujours si précieux et cela a aidé beaucoup d'entre nous à comprendre profondément les 9 équipements de patch.

Malheureusement, cette page a été récemment mise de côté (elle est actuellement à nouveau disponible).

Par conséquent, la nécessité d'avoir une copie physique de cette page pour les développeurs Android sur notre serveur fiable / s.

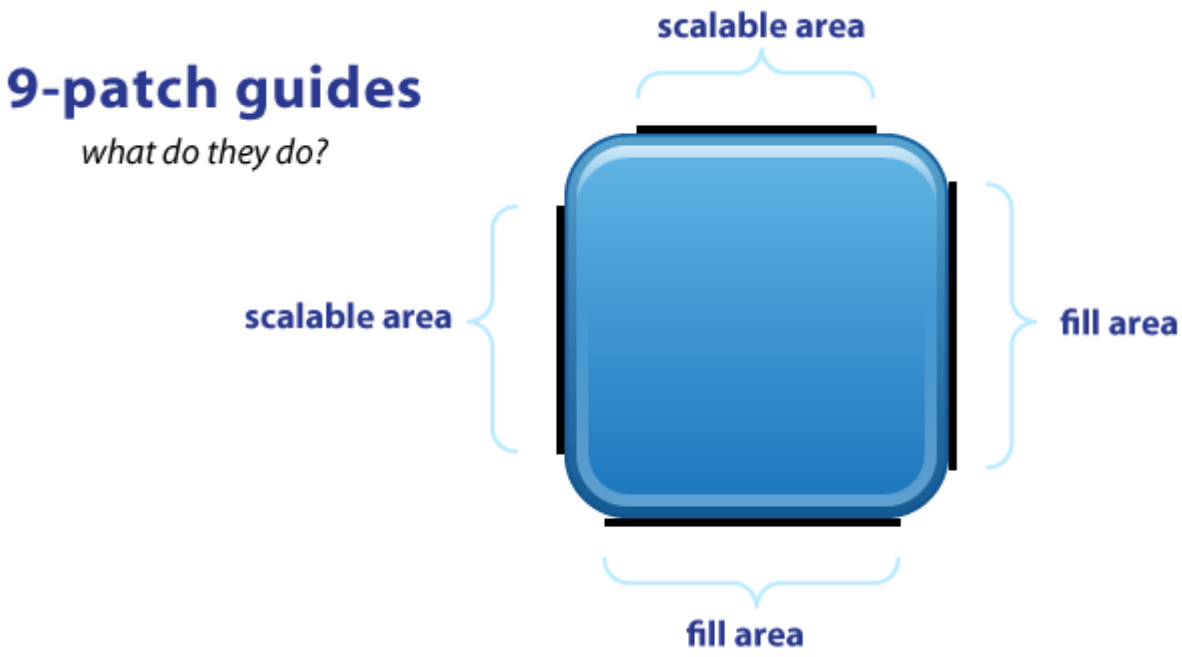
C'est ici.

UN GUIDE SIMPLE À L'APPLICATION 9-PATCH POUR ANDROID UI 18 mai 2011

Alors que je travaillais sur ma première application Android, j'ai trouvé 9-patch (aka 9.png) confus et mal documenté. Après un petit moment, j'ai finalement compris comment cela fonctionnait et j'ai décidé de rassembler quelque chose pour aider les autres à le comprendre.

Fondamentalement, 9-patch utilise la transparence png pour faire une forme avancée de 9-tranche ou échelle9. Les guides sont des lignes droites de 1 pixel dessinées sur le bord de votre image qui définissent la mise à l'échelle et le remplissage de votre image. En nommant votre nom de fichier image.9.png, Android reconnaîtra le format 9.png et utilisera les guides noirs pour mettre à l'échelle et remplir vos images bitmap.

Voici un guide de base:



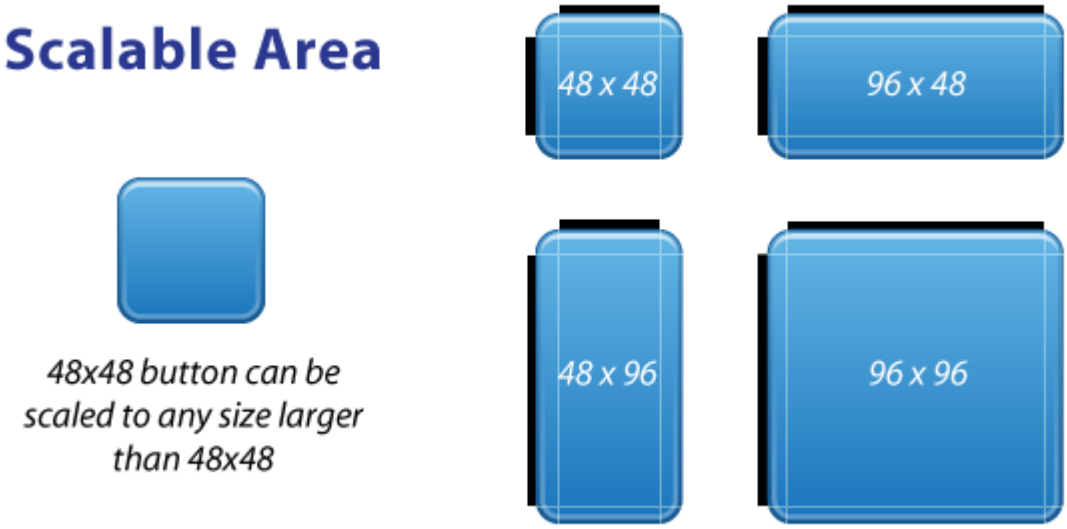
Comme vous pouvez le voir, vous avez des guides de chaque côté de votre image. Les guides TOP et LEFT permettent de mettre à l'échelle votre image (c.-à-d. 9 tranches), tandis que les guides RIGHT et BOTTOM définissent la zone de remplissage.

Les lignes de guidage noires sont coupées / supprimées de votre image - elles n'apparaîtront pas dans l'application. Les guides ne doivent pas avoir plus d'un pixel de largeur, donc si vous voulez un bouton 48 × 48, votre png sera en réalité de 50 × 50. Tout ce qui est plus épais qu'un pixel restera dans votre image. (Mes exemples ont des guides de 4 pixels de large pour une meilleure visibilité. Ils ne doivent vraiment être que de 1 pixel).

Vos guides doivent être en noir (# 000000). Même une légère différence de couleur (# 000001) ou alpha entraînera une défaillance et un étirement normal. Cet échec ne sera pas évident non plus *, il échoue silencieusement! Oui. Vraiment. Maintenant tu sais.

Vous devez également garder à l'esprit que la zone restante du contour d'un pixel doit être complètement transparente. Cela inclut les quatre coins de l'image - ceux-ci devraient toujours être clairs. Cela peut être un problème plus grave que celui que vous réalisez. Par exemple, si vous redimensionnez une image dans Photoshop, elle ajoutera des pixels anti-aliasés pouvant inclure des pixels presque invisibles, ce qui entraînera également son échec *. Si vous devez évoluer dans Photoshop, utilisez le paramètre Voisin le plus proche dans le menu déroulant Rééchantillonnage d'image (en bas du menu contextuel Taille de l'image) pour conserver des bords nets sur vos repères.

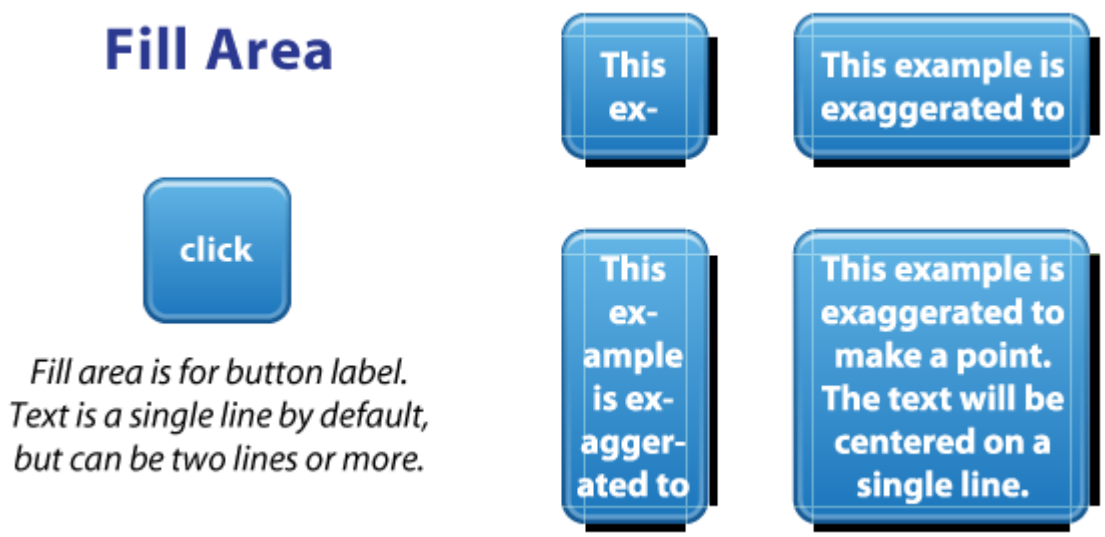
* (mis à jour 1/2012) Il s'agit en fait d'un «correctif» dans le dernier kit de développement. Auparavant, il se manifestait sous la forme d'une rupture soudaine de toutes vos autres images et ressources, et non de l'image de 9 patchs réellement cassée.



Les guides TOP et LEFT sont utilisés pour définir la partie évolutive de votre image - GAUCHE pour la hauteur de mise à l'échelle, TOP pour la largeur de mise à l'échelle. En utilisant une image de bouton comme exemple, cela signifie que le bouton peut s'étendre horizontalement et verticalement dans la partie noire et que tout le reste, comme les coins, restera de la même taille. Cela vous permet d'avoir des boutons qui peuvent s'adapter à n'importe quelle taille et conserver un aspect uniforme.

Il est important de noter que les images à 9 patchs ne sont pas réduites - elles ne font qu'augmenter. Il est donc préférable de commencer le plus petit possible.

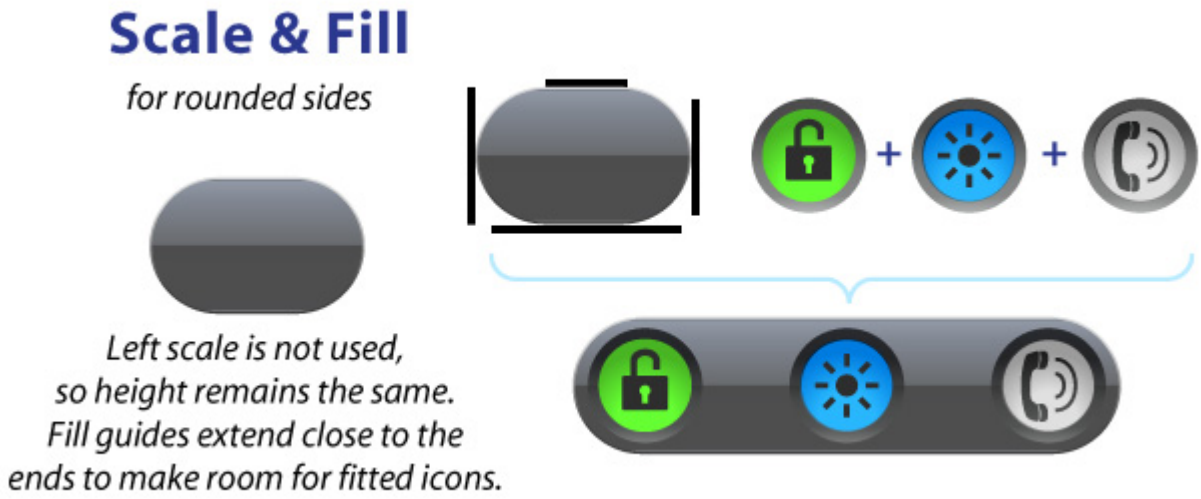
En outre, vous pouvez omettre des parties au milieu de la ligne d'échelle. Ainsi, par exemple, si vous avez un bouton avec un bord brillant au centre, vous pouvez laisser quelques pixels au milieu du guide LEFT. L'axe horizontal central de votre image ne sera pas mis à l'échelle, mais uniquement les parties situées au-dessus et au-dessous de celui-ci, de sorte que votre brillance ne deviendra pas anti-aliasée ou floue.



Les guides de zone de remplissage sont facultatifs et permettent de définir la zone pour des éléments tels que votre étiquette de texte. Fill détermine la quantité d'espace disponible dans votre image pour placer du texte, une icône ou d'autres choses. 9-patch ne concerne pas uniquement les boutons, il fonctionne également pour les images de fond.

L'exemple ci-dessus de bouton et d'étiquette est exagéré simplement pour expliquer l'idée de remplissage - l'étiquette n'est pas complètement exacte. Pour être honnête, je n'ai pas vu comment Android fait des étiquettes à plusieurs lignes, car une étiquette de bouton est généralement une seule ligne de texte.

Enfin, voici une bonne démonstration de la manière dont les guides d'échelle et de remplissage peuvent varier, tels que LinearLayout avec une image d'arrière-plan et des côtés entièrement arrondis:



Avec cet exemple, le guide LEFT n'est pas utilisé mais nous avons toujours besoin d'un guide. L'image d'arrière-plan n'est pas mise à l'échelle verticalement. il ne fait qu'échelonner horizontalement (basé sur le guide TOP). En regardant les guides de remplissage, les guides DROIT et BAS s'étendent au-delà de l'endroit où ils rencontrent les bords incurvés de l'image. Cela me permet de placer mes boutons ronds près des bords du fond pour un look ajusté et ajusté.

Alors c'est tout. 9-patch est super facile, une fois que vous l'obtenez. Ce n'est pas un moyen idéal de procéder à la mise à l'échelle, mais les repères d'échelle des zones de remplissage et des lignes multiples offrent une plus grande flexibilité que les modèles traditionnels à 9 et 9 tranches9. Essayez-le et vous le découvrirez rapidement.

Niveau de transparence des couleurs (alpha)

Valeurs de l'opacité hexadécimale

Alpha(%)	Hex Value
100%	FF
95%	F2
90%	E6
85%	D9
80%	CC
75%	BF
70%	B3
65%	A6
60%	99
55%	8C
50%	80
45%	73
40%	66
35%	59
30%	4D
25%	40
20%	33
15%	26

```
   |     10%     |      1A        |
   |      5%     |      0D        |
   |      0%     |      00        |
    -------------------------------
```

Si vous souhaitez définir la couleur rouge sur 45%.

```
<color name="red_with_alpha_45">#73FF0000</color>
```

Valeur hexadécimale pour le rouge - # FF0000

Vous pouvez ajouter 73 pour une opacité de 45% dans le préfixe - # 73FF0000

Travailler avec le fichier strings.xml

Une ressource de chaîne fournit des chaînes de texte pour votre application avec un style de texte et une mise en forme facultatifs. Il existe trois types de ressources pouvant fournir des chaînes à votre application:

Chaîne

```
XML resource that provides a single string.
```

Syntaxe:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="string_name">text_string</string>
</resources>
```

Et pour utiliser cette chaîne dans la mise en page:

```
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/string_name" />
```

Tableau de chaînes

```
XML resource that provides an array of strings.
```

Syntaxe:

```
    <resources>
<string-array name="planets_array">
    <item>Mercury</item>
    <item>Venus</item>
    <item>Earth</item>
    <item>Mars</item>
</string-array>
```

Usage

```
Resources res = getResources();
String[] planets = res.getStringArray(R.array.planets_array);
```

Quantité Strings (Plurals)

```
XML resource that carries different strings for pluralization.
```

Syntaxe:

```
<?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>
```

Usage:

```
int count = getNumberOfsongsAvailable();
Resources res = getResources();
String songsFound = res.getQuantityString(R.plurals.plural_name, count, count);
```

Lire Ressources en ligne: https://riptutorial.com/fr/android/topic/108/ressources

Chapitre 215: Retrofit2

Introduction

La page officielle de rénovation se décrit comme

Un client REST de type sécurisé pour Android et Java.

Retrofit transforme votre API REST en une interface Java. Il utilise des annotations pour décrire les requêtes HTTP, le remplacement des paramètres URL et la prise en charge des paramètres de requête est intégrée par défaut. En outre, il fournit des fonctionnalités pour le téléchargement de requêtes et de fichiers en plusieurs parties.

Remarques

Dépendances pour la bibliothèque de rattrapage:

De la documentation officielle :

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

Une demande GET simple

Nous allons montrer comment faire une requête `GET` vers une API qui répond avec un objet `JSON` ou un tableau `JSON` . La première chose à faire est d'ajouter les dépendances Retrofit et `GSON` Converter au fichier de dégradé de notre module.

Ajoutez les dépendances pour la bibliothèque de rattrapage comme décrit dans la section Remarques.

Exemple d'objet JSON attendu:

```
{
    "deviceId": "56V56C14SF5B4SF",
    "name": "Steven",
    "eventCount": 0
}
```

Exemple de tableau JSON:

```
[
    {
        "deviceId": "56V56C14SF5B4SF",
        "name": "Steven",
        "eventCount": 0
    },
    {
        "deviceId": "35A80SF3QDV7M9F",
        "name": "John",
        "eventCount": 2
    }
]
```

Exemple de classe de modèle correspondante:

```
public class Device
{
    @SerializedName("deviceId")
    public String id;

    @SerializedName("name")
    public String name;

    @SerializedName("eventCount")
    public int eventCount;
}
```

Les annotations `@SerializedName` proviennent de la bibliothèque `GSON` et nous permettent de `serialize` et `deserialize` cette classe en `JSON` utilisant le nom sérialisé comme clé. Maintenant, nous pouvons créer l'interface pour l'API qui va récupérer les données du serveur.

```
public interface DeviceAPI
{
    @GET("device/{deviceId}")
    Call<Device> getDevice (@Path("deviceId") String deviceID);

    @GET("devices")
    Call<List<Device>> getDevices();
}
```

Il se passe beaucoup de choses ici dans un espace assez compact, alors nous allons le décomposer:

- L'annotation `@GET` provient de Retrofit et indique à la bibliothèque que nous définissons une

requête GET.
- Le chemin entre parenthèses est le point final que notre requête GET doit atteindre (nous définirons l'URL de base un peu plus tard).
- Les accolades nous permettent de remplacer des parties du chemin au moment de l'exécution pour que nous puissions passer des arguments.
- La fonction que nous définissons s'appelle `getDevice` et prend l'identifiant de périphérique que nous voulons comme argument.
- L'annotation `@PATH` indique à Retrofit que cet argument doit remplacer l'espace réservé "deviceId" dans le chemin.
- La fonction renvoie un objet `Call` de type `Device` .

Création d'une classe wrapper:

Nous allons maintenant créer une petite classe wrapper pour notre API afin de conserver le code d'initialisation Retrofit bien intégré.

```
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);
    }
}
```

Cette classe crée une instance GSON pour pouvoir analyser la réponse JSON, crée une instance Retrofit avec notre URL de base et un GSONConverter, puis crée une instance de notre API.

Appel de l'API:

```
// Getting a JSON object
Call<Device> callObject = api.getDevice(deviceID);
callObject.enqueue(new Callback<Response<Device>>()
{
    @Override
    public void onResponse (Call<Device> call, Response<Device> response)
    {
        if (response.isSuccessful())
        {
           Device device = response.body();
        }
    }

    @Override
    public void onFailure (Call<Device> call, Throwable t)
    {
        Log.e(TAG, t.getLocalizedMessage());
    }
```

```
});

// Getting a JSON array
Call<List<Device>> callArray = api.getDevices();
callArray.enqueue(new Callback<Response<List<Device>>()
{
    @Override
    public void onResponse (Call<List<Device>> call, Response<List<Device>> response)
    {
        if (response.isSuccessful())
        {
           List<Device> devices = response.body();
        }
    }

    @Override
    public void onFailure (Call<List<Device>> call, Throwable t)
    {
        Log.e(TAG, t.getLocalizedMessage());
    }
});
```

Cela utilise notre interface API pour créer un objet `Call<Device>` et pour créer un `Call<List<Device>>` respectivement. L'appel de `enqueue` indique à Retrofit de faire cet appel sur un thread d'arrière-plan et de renvoyer le résultat au rappel que nous créons ici.

Remarque: L' analyse d'un tableau JSON d'objets primitifs (tels que *String, Integer, Boolean* et *Double*) est similaire à l'analyse d'un tableau JSON. Cependant, vous n'avez pas besoin de votre propre classe de modèle. Vous pouvez obtenir le tableau de chaînes par exemple en utilisant le type de retour de l'appel comme `Call<List<String>>` .

Ajouter une journalisation à Retrofit2

Les demandes de modification peuvent être enregistrées à l'aide d'un intercepteur. Plusieurs niveaux de détails sont disponibles: AUCUN, BASIQUE, HEADERS, BODY. Voir le projet Github ici .

1. Ajoutez une dépendance à build.gradle:

```
compile 'com.squareup.okhttp3:logging-interceptor:3.8.1'
```

2. Ajouter un intercepteur de journalisation lors de la création de la modification:

```
  HttpLoggingInterceptor loggingInterceptor = new HttpLoggingInterceptor();
  loggingInterceptor.setLevel(LoggingInterceptor.Level.BODY);
  OkHttpClient okHttpClient = new OkHttpClient().newBuilder()
          .addInterceptor(loggingInterceptor)
          .build();
  Retrofit retrofit = new Retrofit.Builder()
          .baseUrl("http://example.com/")
          .client(okHttpClient)
          .addConverterFactory(GsonConverterFactory.create(gson))
          .build();
```

L'exposition des journaux dans le terminal (moniteur Android) doit être évitée dans la version finale, car cela peut entraîner une exposition indésirable d'informations critiques telles que des jetons d'authentification, etc.

Pour éviter que les journaux ne soient exposés lors de l'exécution, vérifiez la condition suivante

```
if(BuildConfig.DEBUG){
     //your interfector code here
    }
```

Par exemple:

```
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();
```

Télécharger un fichier via Multipart

Déclarez votre interface avec les annotations Retrofit2:

```
public interface BackendApiClient {
    @Multipart
    @POST("/uploadFile")
    Call<RestApiDefaultResponse> uploadPhoto(@Part("file\"; filename=\"photo.jpg\" ")
RequestBody photo);
}
```

Où `RestApiDefaultResponse` est une classe personnalisée contenant la réponse.

Construire l'implémentation de votre API et mettre l'appel en file d'attente:

```
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>);
```

Rénovation avec un intercepteur OkHttp

Cet exemple montre comment utiliser un intercepteur de requête avec OkHttp. Cela a de nombreux cas d'utilisation tels que:

- Ajout d'un en- `header` universel à la requête. Par exemple, authentifier une demande
- Débogage des applications en réseau
- Récupération de la `response` brute
- Enregistrement de transaction réseau, etc.
- Définir l'agent utilisateur personnalisé

```
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);
```

Voir la rubrique OkHttp pour plus de détails.

En-tête et corps: un exemple d'authentification

Les annotations `@Header` et `@Body` peuvent être placées dans les signatures de méthode et Retrofit les crée automatiquement en fonction de vos modèles.

```
public interface MyService {
     @POST("authentication/user")
     Call<AuthenticationResponse> authenticateUser(@Body AuthenticationRequest request,
@Header("Authorization") String basicToken);
}
```

AuthenticaionRequest est notre modèle, un POJO, contenant les informations requises par le serveur. Pour cet exemple, notre serveur souhaite la clé client et le secret.

```
public class AuthenticationRequest {
     String clientKey;
```

```
     String clientSecret;
}
```

Notez que dans `@Header("Authorization")` nous `@Header("Authorization")` que nous `@Header("Authorization")` l'en-tête Authorization. Les autres en-têtes seront remplis automatiquement car Retrofit peut déduire ce qu'ils sont basés sur le type d'objets que nous envoyons et attendons en retour.

Nous créons notre service Retrofit quelque part. Nous nous assurons d'utiliser HTTPS.

```
Retrofit retrofit = new Retrofit.Builder()
            .baseUrl("https:// some example site")
            .client(client)
            .build();
MyService myService = retrofit.create(MyService.class)
```

Ensuite, nous pouvons utiliser notre service.

```
AuthenticationRequest request = new AuthenticationRequest();
request.setClientKey(getClientKey());
request.setClientSecret(getClientSecret());
String basicToken = "Basic " + token;
myService.authenticateUser(request, basicToken);
```

Télécharger plusieurs fichiers à l'aide de l'option Rénovation en plusieurs parties

Une fois que vous avez configuré l'environnement Retrofit dans votre projet, vous pouvez utiliser l'exemple suivant qui montre comment télécharger plusieurs fichiers à l'aide de Retrofit:

```
private void mulipleFileUploadFile(Uri[] fileUri) {
    OkHttpClient okHttpClient = new OkHttpClient();
    OkHttpClient clientWith30sTimeout = okHttpClient.newBuilder()
            .readTimeout(30, TimeUnit.SECONDS)
            .build();

    Retrofit retrofit = new Retrofit.Builder()
            .baseUrl(API_URL_BASE)
            .addConverterFactory(new MultiPartConverter())
            .client(clientWith30sTimeout)
            .build();

    WebAPIService service = retrofit.create(WebAPIService.class); //here is the interface
which you have created for the call service
    Map<String, okhttp3.RequestBody> maps = new HashMap<>();

    if (fileUri!=null && fileUri.length>0) {
        for (int i = 0; i < fileUri.length; i++) {
            String filePath = getRealPathFromUri(fileUri[i]);
            File file1 = new File(filePath);

            if (filePath != null && filePath.length() > 0) {
                if (file1.exists()) {
                    okhttp3.RequestBody requestFile =
```

```
okhttp3.RequestBody.create(okhttp3.MediaType.parse("multipart/form-data"), file1);
                    String filename = "imagePath" + i; //key for upload file like : imagePath0
                    maps.put(filename + "\"; filename=\"" + file1.getName(), requestFile);
                }
            }
        }
    }

    String descriptionString = " string request";//
    //hear is the your json request
    Call<String> call = service.postFile(maps, descriptionString);
    call.enqueue(new Callback<String>() {
        @Override
        public void onResponse(Call<String> call,
                               Response<String> response) {
            Log.i(LOG_TAG, "success");
            Log.d("body==>", response.body().toString() + "");
            Log.d("isSuccessful==>", response.isSuccessful() + "");
            Log.d("message==>", response.message() + "");
            Log.d("raw==>", response.raw().toString() + "");
            Log.d("raw().networkResponse()", response.raw().networkResponse().toString() +
"");
        }

        @Override
        public void onFailure(Call<String> call, Throwable t) {
            Log.e(LOG_TAG, t.getMessage());
        }
    });
}

public String getRealPathFromUri(final Uri uri) { // function for file path from uri,
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.KITKAT &&
DocumentsContract.isDocumentUri(mContext, uri)) {
        // ExternalStorageProvider
        if (isExternalStorageDocument(uri)) {
            final String docId = DocumentsContract.getDocumentId(uri);
            final String[] split = docId.split(":");
            final String type = split[0];

            if ("primary".equalsIgnoreCase(type)) {
                return Environment.getExternalStorageDirectory() + "/" + split[1];
            }
        }
        // DownloadsProvider
        else if (isDownloadsDocument(uri)) {

            final String id = DocumentsContract.getDocumentId(uri);
            final Uri contentUri = ContentUris.withAppendedId(
                    Uri.parse("content://downloads/public_downloads"), Long.valueOf(id));

            return getDataColumn(mContext, contentUri, null, null);
        }
        // MediaProvider
        else if (isMediaDocument(uri)) {
            final String docId = DocumentsContract.getDocumentId(uri);
            final String[] split = docId.split(":");
            final String type = split[0];

            Uri contentUri = null;
            if ("image".equals(type)) {
```

```
                contentUri = MediaStore.Images.Media.EXTERNAL_CONTENT_URI;
            } else if ("video".equals(type)) {
                contentUri = MediaStore.Video.Media.EXTERNAL_CONTENT_URI;
            } else if ("audio".equals(type)) {
                contentUri = MediaStore.Audio.Media.EXTERNAL_CONTENT_URI;
            }

            final String selection = "_id=?";
            final String[] selectionArgs = new String[]{
                    split[1]
            };

            return getDataColumn(mContext, contentUri, selection, selectionArgs);
        }
    }
    // MediaStore (and general)
    else if ("content".equalsIgnoreCase(uri.getScheme())) {

        // Return the remote address
        if (isGooglePhotosUri(uri))
            return uri.getLastPathSegment();

        return getDataColumn(mContext, uri, null, null);
    }
    // File
    else if ("file".equalsIgnoreCase(uri.getScheme())) {
        return uri.getPath();
    }

    return null;
}
```

Voici l'interface

```
public interface WebAPIService {
    @Multipart
    @POST("main.php")
    Call<String> postFile(@PartMap Map<String,RequestBody> Files, @Part("json") String
description);
}
```

Télécharger un fichier à partir du serveur à l'aide de Retrofit2

Déclaration d'interface pour le téléchargement d'un fichier

```
public interface ApiInterface {
    @GET("movie/now_playing")
    Call<MovieResponse> getNowPlayingMovies(@Query("api_key") String apiKey, @Query("page")
int page);

    // option 1: a resource relative to your base URL
    @GET("resource/example.zip")
    Call<ResponseBody> downloadFileWithFixedUrl();

    // option 2: using a dynamic URL
    @GET
    Call<ResponseBody> downloadFileWithDynamicUrl(@Url String fileUrl);
}
```

L'option 1 est utilisée pour télécharger un fichier à partir du serveur ayant une URL fixe. et l'option 2 est utilisée pour transmettre une valeur dynamique en tant qu'URL complète pour demander un appel. Cela peut être utile lors du téléchargement de fichiers, qui dépendent de paramètres tels que l'utilisateur ou l'heure.

Configuration de la mise à niveau pour faire des appels à l'API

```
public class ServiceGenerator {

    public static final String API_BASE_URL = "http://your.api-base.url/";

    private static OkHttpClient.Builder httpClient = new OkHttpClient.Builder();

    private static Retrofit.Builder builder =
            new Retrofit.Builder()
            .baseUrl(API_BASE_URL)
            .addConverterFactory(GsonConverterFactory.create());

    public static <S> S createService(Class<S> serviceClass){
        Retrofit retrofit = builder.client(httpClient.build()).build();
        return retrofit.create(serviceClass);
    }

}
```

Maintenant, implémentez api pour télécharger le fichier du serveur

```
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) {

            }
        });
    }
```

Et après avoir reçu une réponse dans le rappel, codez une IO standard pour enregistrer le fichier sur le disque. Voici le 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;
        }
    }
```

Notez que nous avons spécifié **ResponseBody** comme type de retour, sinon Retrofit essaiera de l'analyser et de le convertir, ce qui n'a aucun sens lorsque vous téléchargez un fichier.

Si vous souhaitez en savoir plus sur les améliorations Retrofit, accédez à ce lien car il est très utile. [1]: https://futurestud.io/blog/retrofit-getting-started-and-android-client

Déboguer avec Stetho

Ajoutez les dépendances suivantes à votre application.

```
compile 'com.facebook.stetho:stetho:1.5.0'
```

```
compile 'com.facebook.stetho:stetho-okhttp3:1.5.0'
```

Dans la méthode `onCreate` votre classe d' `onCreate` , appelez les éléments suivants.

```
Stetho.initializeWithDefaults(this);
```

Lors de la création de votre instance `Retrofit` , créez une instance OkHttp personnalisée.

```
OkHttpClient.Builder clientBuilder = new OkHttpClient.Builder();
clientBuilder.addNetworkInterceptor(new StethoInterceptor());
```

Ensuite, définissez cette instance OkHttp personnalisée dans l'instance Retrofit.

```
Retrofit retrofit = new Retrofit.Builder()
    // ...
    .client(clientBuilder.build())
    .build();
```

Connectez maintenant votre téléphone à votre ordinateur, lancez l'application et tapez `chrome://inspect` dans votre navigateur Chrome. Les appels réseau de mise à niveau doivent maintenant apparaître pour que vous puissiez les inspecter.

Retrofit 2 Convertisseur Xml personnalisé

Ajout de dépendances dans le fichier build.gradle.

```
dependencies {
    ....
    compile 'com.squareup.retrofit2:retrofit:2.1.0'
    compile ('com.thoughtworks.xstream:xstream:1.4.7') {
        exclude group: 'xmlpull', module: 'xmlpull'
    }
    ....
}
```

Puis créez 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);
    }
}
```

créer une classe pour gérer la requête du corps.

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

créer une classe pour gérer la réponse du corps.

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

Donc, ce point, nous pouvons envoyer et recevoir du XML, nous avons juste besoin de créer des annotations XStream pour les entités.

Ensuite, créez une instance Retrofit:

```
XStream xs = new XStream(new DomDriver());
xs.autodetectAnnotations(true);

Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com/")
    .addConverterFactory(XStreamXmlConverterFactory.create(xs))
    .client(client)
    .build();
```

Une simple requête POST avec GSON

Exemple de JSON:

```
{
    "id": "12345",
    "type": "android"
}
```

Définissez votre demande:

```
public class GetDeviceRequest {
```

```
    @SerializedName("deviceId")
    private String mDeviceId;

    public GetDeviceRequest(String deviceId) {
        this.mDeviceId = deviceId;
    }

    public String getDeviceId() {
        return mDeviceId;
    }

}
```

Définissez votre service (points d'extrémité à atteindre):

```
public interface Service {

    @POST("device")
    Call<Device> getDevice(@Body GetDeviceRequest getDeviceRequest);

}
```

Définissez votre instance singleton du client réseau:

```
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;
    }

}
```

Définissez un objet modèle simple pour le périphérique:

```
public class Device {

    @SerializedName("id")
```

```
    private String mId;

    @SerializedName("type")
    private String mType;

    public String getId() {
        return mId;
    }

    public String getType() {
        return mType;
    }

}
```

Définir le contrôleur pour gérer les demandes pour le périphérique

```
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
            }

        });
```

Lecture de l'URL du formulaire XML avec Retrofit 2

Nous utiliserons retrofit 2 et SimpleXmlConverter pour obtenir des données XML à partir de l'URL et de l'analyse dans la classe Java.

Ajoutez une dépendance au script Gradle:

```
compile 'com.squareup.retrofit2:retrofit:2.1.0'
compile 'com.squareup.retrofit2:converter-simplexml:2.1.0'
```

Créer une interface

Créez également un wrapper de classe xml dans notre cas Rss class

```
    public interface ApiDataInterface{

        // path to xml link on web site

        @GET (data/read.xml)

        Call<Rss> getData();

}
```

Fonction de lecture Xml

```
private void readXmlFeed() {
        try {

            // base url - url of web site
            Retrofit retrofit = new Retrofit.Builder()
                    .baseUrl(http://www.google.com/)
                    .client(new OkHttpClient())
                    .addConverterFactory(SimpleXmlConverterFactory.create())
                    .build();

            ApiDataInterface apiService = retrofit.create(ApiDataInterface.class);

            Call<Rss> call = apiService.getData();
            call.enqueue(new Callback<Rss>() {

                @Override
                public void onResponse(Call<Rss> call, Response<Rss> response) {

                    Log.e("Response success", response.message());

                }

                @Override
                public void onFailure(Call<Rss> call, Throwable t) {
                    Log.e("Response fail", t.getMessage());
                }
            });

        } catch (Exception e) {
            Log.e("Exception", e.getMessage());
        }


    }
```

Ceci est un exemple de classe Java avec des annotations SimpleXML

En savoir plus sur les annotations 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;
```

Lire Retrofit2 en ligne: https://riptutorial.com/fr/android/topic/1132/retrofit2

Chapitre 216: Retrofit2 avec RxJava

Examples

Retrofit2 avec RxJava

Tout d'abord, ajoutez les dépendances pertinentes dans le fichier build.gradle.

```
dependencies {
    ....
    compile 'com.squareup.retrofit2:retrofit:2.3.0'
    compile 'com.squareup.retrofit2:converter-gson:2.3.0'
    compile 'com.squareup.retrofit2:adapter-rxjava:2.3.0'
    ....
}
```

Créez ensuite le modèle que vous souhaitez recevoir:

```
public class Server {
    public String name;
    public String url;
    public String apikey;
    public List<Site> siteList;
}
```

Créez une interface contenant les méthodes utilisées pour échanger des données avec le serveur distant:

```
public interface ApiServerRequests {

    @GET("api/get-servers")
    public Observable<List<Server>> getServers();
}
```

Ensuite, créez une instance `Retrofit` :

```
public ApiRequests DeviceAPIHelper ()
{
    Gson gson = new GsonBuilder().create();

    Retrofit retrofit = new Retrofit.Builder()
            .baseUrl("http://example.com/")
            .addConverterFactory(GsonConverterFactory.create(gson))
            .addCallAdapterFactory(RxJavaCallAdapterFactory.create())
            .build();

    api = retrofit.create(ApiServerRequests.class);
    return api;
}
```

Ensuite, n'importe où depuis le code, appelez la méthode:

```
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
       }
   });
```

Mise à niveau avec RxJava pour récupérer les données de manière asynchrone

À partir du dépôt GitHub de RxJava, *RxJava est une implémentation Java de Reactive Extensions: une bibliothèque pour composer des programmes asynchrones et basés sur des événements en utilisant des séquences observables. Il étend le motif de l'observateur pour prendre en charge des séquences de données / événements et ajoute des opérateurs qui vous permettent de composer des séquences de manière déclarative tout en évitant les problèmes tels que le threading de bas niveau, la synchronisation, la sécurité des threads et les structures de données simultanées.*

Retrofit est un client HTTP de type sécurisé pour Android et Java. Grâce à cela, les développeurs peuvent simplifier tous les aspects du réseau. Par exemple, nous allons télécharger des fichiers JSON et les afficher dans RecyclerView sous forme de liste.

Commencer:

Ajoutez les dépendances RxJava, RxAndroid et Retrofit dans votre fichier build.gradle au niveau de l'application: `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"
```

Définir ApiClient et ApiInterface pour échanger des données du serveur

```
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;
}}
```

interface publique ApiInterface {

```
@GET("repos/{org}/{repo}/issues")
Observable<List<Issue>> getIssues(@Path("org") String organisation,
                                  @Path("repo") String repositoryName,
                                  @Query("page") int pageNumber);}
```

Notez que getRepos () renvoie un objet Observable et pas seulement une liste de problèmes.

Définir les modèles

Un exemple pour cela est montré. Vous pouvez utiliser des services gratuits comme JsonSchema2Pojo ou ceci.

```
public class Comment {

@SerializedName("url")
@Expose
private String url;
@SerializedName("html_url")
@Expose
private String htmlUrl;

//Getters and Setters
}
```

Créer une instance de modification

```
ApiInterface apiService = ApiClient.getInstance().create(ApiInterface.class);
```

Ensuite, utilisez cette instance pour extraire des données du serveur

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

Vous avez maintenant récupéré avec succès des données à partir d'un serveur à l'aide de Retrofit et de RxJava.

Exemple de requêtes imbriquées: plusieurs demandes, combiner les résultats

Supposons que nous ayons une API qui nous permette d'obtenir des métadonnées d'objet dans une requête unique (`getAllPets`), et une autre requête qui possède des données complètes de ressource unique (`getSinglePet`). Comment pouvons-nous tous les interroger dans une seule chaîne?

```
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;
    }
}
```

}

Lire Retrofit2 avec RxJava en ligne: https://riptutorial.com/fr/android/topic/7632/retrofit2-avec-rxjava

Chapitre 217: RoboGuice

Examples

Exemple simple

RoboGuice est un framework qui apporte la simplicité et la facilité de Dependency Injection à Android, en utilisant la propre bibliothèque Guice de Google.

```
@ContentView(R.layout.main)
class RoboWay extends RoboActivity {
    @InjectView(R.id.name)              TextView name;
    @InjectView(R.id.thumbnail)         ImageView thumbnail;
    @InjectResource(R.drawable.icon)    Drawable icon;
    @InjectResource(R.string.app_name)  String myName;
    @Inject                             LocationManager loc;

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        name.setText( "Hello, " + myName );
    }
}
```

Installation pour les projets Gradle

Ajoutez le pom suivant à la section dépendances de votre fichier de construction gradle:

```
project.dependencies {
    compile 'org.roboguice:roboguice:3.+'
    provided 'org.roboguice:roboblender:3.+'
}
```

Annotation @ContentView

L'annotation @ContentView peut être utilisée pour améliorer davantage le développement des activités et remplacer l'instruction setContentView:

```
@ContentView(R.layout.myactivity_layout)
public class MyActivity extends RoboActivity {
    @InjectView(R.id.text1) TextView textView;

    @Override
    protected void onCreate( Bundle savedState ) {
        textView.setText("Hello!");
    }
}
```

@InjectResource annotation

Vous pouvez injecter tout type de ressource, Strings, Animations, Drawables, etc.

Pour injecter votre première ressource dans une activité, vous devez:

- Hériter de RoboActivity
- Annotez vos ressources avec @InjectResource

Exemple

```
@InjectResource(R.string.app_name) String name;

@InjectResource(R.drawable.ic_launcher) Drawable icLauncher;

@InjectResource(R.anim.my_animation) Animation myAnimation;
```

Annotation @InjectView

Vous pouvez injecter n'importe quelle vue à l'aide de l'annotation @InjectView:

Vous devrez:

- Hériter de RoboActivity
- Définir votre vue de contenu
- Annotez vos vues avec @InjectView

Exemple

```
@InjectView(R.id.textView1) TextView textView1;

@InjectView(R.id.textView2) TextView textView2;

@InjectView(R.id.imageView1) ImageView imageView1;
```

Introduction à RoboGuice

`RoboGuice` est un framework qui apporte la simplicité et la facilité de Dependency Injection à Android, en utilisant la propre bibliothèque Guice de Google.

RoboGuice 3 réduit votre code d'application. Moins de code signifie moins d'opportunités pour les bogues. Il rend également votre code plus facile à suivre - votre code n’est plus encombré des mécanismes de la plate-forme Android, mais il peut désormais se concentrer sur la logique propre à votre application.

Pour vous donner une idée, jetez un coup d’œil à cet exemple simple d’une `Activity` Android typique:

```
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 );
        }
    }
```

Cet exemple est composé de 19 lignes de code. Si vous essayez de lire `onCreate()` , vous devez ignorer 5 lignes d'initialisation standard pour trouver la seule qui compte vraiment: `name.setText()` . Et les activités complexes peuvent aboutir à beaucoup plus de ce type de code d'initialisation.

Comparez cela à la même application, écrite à l'aide de `RoboGuice` :

```
@ContentView(R.layout.main)
   class RoboWay extends RoboActivity {
        @InjectView(R.id.name)             TextView name;
        @InjectView(R.id.thumbnail)        ImageView thumbnail;
        @InjectResource(R.drawable.icon)   Drawable icon;
        @InjectResource(R.string.app_name) String myName;
        @Inject                            LocationManager loc;

        public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            name.setText( "Hello, " + myName );
        }
    }
```

L'objectif de RoboGuice est de faire en sorte que votre code concerne votre application, plutôt que d'être à propos de tout le code d'initialisation et de cycle de vie que vous devez généralement conserver dans Android.

Annotations:

Annotation @ContentView:

L'annotation @ContentView peut être utilisée pour améliorer davantage le développement des activités et remplacer l'instruction setContentView:

```
@ContentView(R.layout.myactivity_layout)
    public class MyActivity extends RoboActivity {
        @InjectView(R.id.text1) TextView textView;

        @Override
        protected void onCreate( Bundle savedState ) {
            textView.setText("Hello!");
        }
    }
```

Annotation @InjectResource:

Tout d'abord, vous avez besoin d'une activité qui hérite de RoboActivity. Ensuite, en supposant que vous ayez une animation my_animation.xml dans votre dossier res / anim, vous pouvez maintenant la référencer avec une annotation:

```
public class MyActivity extends RoboActivity {
    @InjectResource(R.anim.my_animation) Animation myAnimation;
    // the rest of your code
}
```

Annotation @Inject:

Assurez-vous que votre activité s'étend à partir de RoboActivity et annotez votre membre du service système avec @Inject. Roboguice fera le reste.

```
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();
```

En plus des vues, des ressources, des services et d'autres éléments spécifiques à Android, RoboGuice peut injecter des objets Java simples. Par défaut, Roboguice appelle un constructeur sans argument sur votre POJO.

```
class MyActivity extends RoboActivity {
    @Inject Foo foo; // this will basically call new Foo();
}
```

Lire RoboGuice en ligne: https://riptutorial.com/fr/android/topic/2563/roboguice

Chapitre 218: Robolectric

Introduction

Le test unitaire consiste à prendre un morceau de code et à le tester indépendamment, sans aucune autre dépendance ou partie du système (par exemple la base de données).

Robolectric est un framework de test unitaire qui défigure le jar Android SDK pour que vous puissiez tester le développement de votre application Android. Les tests s'exécutent à l'intérieur de la JVM sur votre poste de travail en quelques secondes.

Les combiner vous permet d'effectuer des tests rapides sur le JVN en utilisant toujours les API Android.

Examples

Test Robolectric

```
@RunWith(RobolectricTestRunner.class)
public class MyActivityTest {

  @Test
  public void clickingButton_shouldChangeResultsViewText() throws Exception {
    MyActivity activity = Robolectric.setupActivity(MyActivity.class);

    Button button = (Button) activity.findViewById(R.id.button);
    TextView results = (TextView) activity.findViewById(R.id.results);

    button.performClick();
    assertThat(results.getText().toString()).isEqualTo("Robolectric Rocks!");
  }
}
```

Configuration

Pour configurer la méthode robotique, ajoutez `@Config` annotation `@Config` à une classe ou à une méthode de test.

Exécuter avec une classe d'application personnalisée

```
@RunWith(RobolectricTestRunner.class)
@Config(application = MyApplication.class)
public final class MyTest {
}
```

Définir le SDK cible

```
@RunWith(RobolectricTestRunner.class)
@Config(sdk = Build.VERSION_CODES.LOLLIPOP)
public final class MyTest {
}
```

Exécuter avec un manifeste personnalisé

Une fois spécifié, Robolectric ressemblera au répertoire en cours. La valeur par défaut est `AndroidManifest.xml`

Les ressources et les actifs seront chargés par rapport au manifeste.

```
@RunWith(RobolectricTestRunner.class)
@Config(manifest = "path/AndroidManifest.xml")
public final class MyTest {
}
```

Utilisez des qualificatifs

Des qualificatifs possibles peuvent être trouvés dans les documents Android .

```
@RunWith(RobolectricTestRunner.class)
public final class MyTest {

    @Config(qualifiers = "sw600dp")
    public void testForTablet() {
    }
}
```

Lire Robolectric en ligne: https://riptutorial.com/fr/android/topic/8743/robolectric

Chapitre 219: Secure SharedPreferences

Introduction

Les préférences partagées sont **des fichiers XML basés sur des valeurs-clés** . Il se trouve sous `/data/data/package_name/shared_prefs/<filename.xml>` .

Ainsi, un utilisateur disposant des privilèges root peut accéder à cet emplacement et peut modifier ses valeurs. Si vous souhaitez protéger des valeurs dans vos préférences partagées, vous pouvez écrire un mécanisme simple de chiffrement et de déchiffrement.

Vous devez savoir que les préférences partagées n'ont jamais été conçues pour être sécurisées, mais simplement pour conserver les données.

Syntaxe

1. public static String encrypt (entrée de chaîne);
2. public static String decrypt (entrée de chaîne);

Paramètres

Paramètre	Définition
contribution	Valeur de chaîne pour chiffrer ou déchiffrer.

Remarques

Les préférences partagées n'ont jamais été conçues pour être sécurisées, c'est juste un moyen simple de conserver les données.

Ce n'est pas une bonne idée d'utiliser les préférences partagées pour stocker des informations critiques telles que les informations d'identification de l'utilisateur. Pour enregistrer les informations d'identification de l'utilisateur (telles que les mots de passe), vous devez utiliser d'autres méthodes telles que `AccountManager` d'Android.

Examples

Sécuriser une préférence partagée

Codec Simple

Pour illustrer le principe de fonctionnement, nous pouvons utiliser le cryptage et le décryptage simples comme suit.

```
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));
}
```

Technique d'implémentation

```
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);
```

Lire Secure SharedPreferences en ligne: https://riptutorial.com/fr/android/topic/9887/secure-sharedpreferences

Chapitre 220: Secure SharedPreferences

Introduction

Les préférences partagées sont **des fichiers XML basés sur des valeurs-clés** . Il se trouve sous / data / data / nom_package / shared_prefs / <filename.xml>.

Ainsi, un utilisateur disposant des privilèges root peut accéder à cet emplacement et peut modifier ses valeurs. Si vous souhaitez protéger des valeurs dans vos préférences partagées, vous pouvez écrire un mécanisme simple de chiffrement et de déchiffrement.

Vous devez savoir que les préférences partagées n'ont jamais été conçues pour être sécurisées, mais simplement pour conserver les données.

Syntaxe

1. public static String encrypt (entrée de chaîne);
2. public static String decrypt (entrée de chaîne);

Paramètres

Paramètre	Définition
contribution	Valeur de chaîne pour chiffrer ou déchiffrer.

Remarques

Les préférences partagées n'ont jamais été conçues pour être sécurisées, c'est juste un moyen simple de conserver les données.

Ce n'est pas une bonne idée d'utiliser les préférences partagées pour stocker des informations critiques telles que les informations d'identification de l'utilisateur. Pour enregistrer les informations d'identification de l'utilisateur (telles que les mots de passe), vous devez utiliser d'autres méthodes telles que `AccountManager` d'Android.

Examples

Sécuriser une préférence partagée

Codec Simple

Pour illustrer le principe de fonctionnement, nous pouvons utiliser le cryptage et le décryptage simples comme suit.

```
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));
}
```

Technique d'implémentation

```
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);
```

Lire Secure SharedPreferences en ligne: https://riptutorial.com/fr/android/topic/9890/secure-sharedpreferences

Chapitre 221: Sécurité

Examples

Vérification de la signature de l'application - Détection de sabotage

Cette technique explique comment garantir que votre certificat .apk a été signé avec votre certificat de développeur et exploite le fait que le certificat reste cohérent et que vous seul y avez accès. Nous pouvons casser cette technique en 3 étapes simples:

- Trouvez votre signature de certificat de développeur.
- Intégrez votre signature dans une constante String dans votre application.
- Vérifiez que la signature à l'exécution correspond à notre signature de développeur intégrée.

Voici l'extrait de code:

```
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;

}
```

Lire Sécurité en ligne: https://riptutorial.com/fr/android/topic/4664/securite

Chapitre 222: SensorManager

Examples

Récupération des événements de capteur

Récupération des informations de capteur depuis les capteurs embarqués:

```
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
    }
}
```

Transformation du capteur en système de coordonnées mondial

Les valeurs de capteur renvoyées par Android correspondent respectivement au système de coordonnées du téléphone (par exemple, + Y pointe vers le haut du téléphone). Nous pouvons transformer ces valeurs de capteur en un système de coordonnées mondial (par exemple, + Y pointe vers le nord magnétique, tangent au sol) en utilisant la matrice de rotation des gestionnaires de capteurs

Tout d'abord, vous devez déclarer et initialiser les matrices / tableaux où les données seront stockées (vous pouvez le faire dans la méthode `onCreate` , par exemple):

```
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];
```

Ensuite, nous devons détecter les changements dans les valeurs des capteurs, les stocker dans les tableaux correspondants (si nous voulons les utiliser plus tard / ailleurs), puis calculer la matrice de rotation et la transformation résultante en coordonnées mondiales:

```
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];

}
```

Décidez si votre appareil est statique ou non, en utilisant l'accéléromètre

Ajoutez le code suivant à la `onCreate()` / `onResume()` :

```
SensorManager sensorManager;
Sensor mAccelerometer;
final float movementThreshold = 0.5f;  // You may have to change this value.
boolean isMoving = false;
float[] prevValues = {1.0f, 1.0f, 1.0f};
float[] currValues = new float[3];

sensorManager = (SensorManager)getSystemService(SENSOR_SERVICE);
mAccelerometer = sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
sensorManager.registerListener(this, mAccelerometer, SensorManager.SENSOR_DELAY_NORMAL);
```

Vous devrez peut-être ajuster la sensibilité en adaptant le seuil de `movementThreshold` par essais et erreurs. Ensuite, remplacez la méthode `onSensorChanged()` comme suit:

```
@Override
public void onSensorChanged(SensorEvent event) {
    if (event.sensor == mAccelerometer) {
        System.arraycopy(event.values, 0, currValues, 0, event.values.length);
        if ((Math.abs(currValues[0] - prevValues[0]) > movementThreshold) ||
                (Math.abs(currValues[1] - prevValues[1]) > movementThreshold) ||
                (Math.abs(currValues[2] - prevValues[2]) > movementThreshold)) {
            isMoving = true;
        } else {
            isMoving = false;
        }
        System.arraycopy(currValues, 0, prevValues, 0, currValues.length);
    }
}
```

Si vous souhaitez empêcher l'installation de votre application sur des appareils ne disposant pas d'accéléromètre, vous devez ajouter la ligne suivante à votre manifeste:

```
<uses-feature android:name="android.hardware.sensor.accelerometer" />
```

Lire SensorManager en ligne: https://riptutorial.com/fr/android/topic/3344/sensormanager

Chapitre 223: shell adb

Introduction

`adb shell` ouvre un shell Linux dans un périphérique ou un émulateur cible. C'est le moyen le plus puissant et le plus polyvalent de contrôler un appareil Android via `adb` .

Ce sujet a été séparé d' ADB (Android Debug Bridge) en raison de la limite des exemples, dont beaucoup impliquaient la commande `adb shell` .

Syntaxe

- shell adb [-e échapper] [-n] [-Tt] [-x] [commande]

Paramètres

Paramètre	Détails
-e	choisissez le caractère d'échappement, ou "aucun"; par défaut '~'
-n	ne lis pas de stdin
-T	désactiver l'allocation PTY
-t	forcer l'allocation PTY
-X	désactiver les codes de sortie distants et la séparation stdout / stderr

Examples

Envoyer du texte, des touches enfoncées et des événements tactiles au périphérique Android via ADB

exécuter la commande suivante pour insérer le texte dans une vue avec un focus (s'il prend en charge la saisie de texte)

6,0

Envoyer du texte sur le SDK 23+

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

Si déjà connecté à votre appareil via `adb` :

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

6,0

Envoyer du texte avant le SDK 23

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

Les espaces ne sont pas acceptés comme entrée, remplacez-les par% s.

Envoyer des événements

Pour simuler une pression sur la touche d'alimentation matérielle

```
adb shell input keyevent 26
```

Ou bien

```
adb shell input keyevent POWER
```

Même si vous n'avez pas de clé matérielle, vous pouvez toujours utiliser un `keyevent` pour effectuer l'action équivalente

```
adb shell input keyevent CAMERA
```

Envoyer un événement tactile en entrée

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

Envoyer un événement swipe en entrée

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

* DURÉE est facultative, par défaut = 300ms. la source

Obtenez des points X et Y en activant l'emplacement du pointeur dans l'option développeur.

Exemple de script shell ADB

> Pour exécuter un script dans Ubuntu, créez un script.sh avec le bouton droit de la souris sur le fichier et ajoutez une autorisation en lecture / écriture et cochez **autoriser l'exécution du fichier en tant que programme** .
>
> Ouvrez l'émulateur de terminal et exécutez la commande ./script.sh

Script.sh

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

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

Pour une liste complète des numéros d'événement

- liste restreinte de plusieurs événements intéressants ADB Shell Input Events
- documentation de référence https://developer.android.com/reference/android/view/KeyEvent.html#KEYCODE_POWER .

Liste des paquets

Imprime tous les paquets, éventuellement uniquement ceux dont le nom du paquet contient le texte dans <FILTER>.

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

All <FILTER>

adb shell pm list packages
```

Les attributs:

`-f` pour voir leur fichier associé.

`-i` Voir l'installateur pour les paquets.

`-u` pour inclure également les paquets désinstallés.

`-u` Inclure également les paquets désinstallés.

Attributs qui filtrent:

`-d` pour les paquets désactivés.

`-e` pour les paquets activés.

`-s` pour les packages système.

`-3` pour les packages tiers.

`--user <USER_ID>` pour un espace utilisateur spécifique à interroger.

Octroi & révocation d'autorisations API 23+

Un one-liner qui aide à accorder ou à révoquer des autorisations vulnérables.

- **octroi**

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

- **révoquer**

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

- **Accorder toutes les autorisations d'exécution à la fois sur l'installation (-g)**

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

Imprimer les données d'application

Cette commande imprime toutes les données d'application pertinentes:

- code de version
- nom de la version
- autorisations accordées (Android API 23+)
- etc..

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

Enregistrement de l'affichage

4.4

Enregistrement de l'affichage des appareils fonctionnant sous Android 4.4 (API niveau 19) et supérieur:

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

(appuyez sur Ctrl-C pour arrêter l'enregistrement)

Téléchargez le fichier depuis l'appareil:

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

> Remarque: Arrêtez l'enregistrement d'écran en appuyant sur Ctrl-C, sinon l'enregistrement s'arrête automatiquement à trois minutes ou à la limite de temps définie par `--time-limit` .

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

Règle la taille de la vidéo: 1280x720. La valeur par défaut est la résolution d'affichage native du périphérique (si elle est prise en charge), sinon 1280x720. Pour de meilleurs résultats, utilisez une taille prise en charge par l'encodeur AVC (Advanced Video Coding) de votre appareil.

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

Définit le débit binaire vidéo pour la vidéo, en mégabits par seconde. La valeur par défaut est 4 Mbps. Vous pouvez augmenter le débit binaire pour améliorer la qualité vidéo, mais cela entraîne des fichiers vidéo plus volumineux. L'exemple suivant définit le débit binaire d'enregistrement sur 5 Mbits / s:

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

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

Définit la durée d'enregistrement maximale, en secondes. La valeur par défaut et maximale est 180 (3 minutes).

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

Fait pivoter la sortie de 90 degrés. Cette fonctionnalité est expérimentale.

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

Affiche les informations du journal sur l'écran de la ligne de commande. Si vous ne définissez pas cette option, l'utilitaire n'affiche aucune information pendant l'exécution.

> Remarque: Cela peut ne pas fonctionner sur certains appareils.

4.4

La commande d'enregistrement d'écran n'est pas compatible avec les versions Android pré 4.4

> La commande screenrecord est un utilitaire shell permettant d'enregistrer l'affichage des périphériques exécutant Android 4.4 (niveau 19 de l'API) et supérieur. L'utilitaire enregistre l'activité de l'écran dans un fichier MPEG-4.

Modification des autorisations de fichier à l'aide de la commande chmod

Notez que pour changer les prmissions de fichiers, votre périphérique doit être rooté, *`su` binary ne vient pas avec les périphériques livrés en usine!*

Convention:

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

Autorisation numérique construite à partir de sections utilisateur, groupe et monde.

Par exemple, si vous souhaitez modifier le fichier pour qu'il soit lisible, accessible en écriture et exécutable par tous, ce sera votre commande:

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

Ou

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

si vous avez l'intention de lui refuser des autorisations.

1er chiffre - spécifie la permission de l'utilisateur, **2e chiffre** - spécifie l'autorisation de groupe, **3e chiffre** - spécifie la permission du monde (autres).

Autorisations d'accès:

```
--- :   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)
```

Définir la date / heure via adb

6,0

Le format SET par défaut est `MMDDhhmm[[CC]YY][.ss]` , c'est-à-dire (2 chiffres chacun)

Par exemple, pour définir le 17 juillet à 10h10, sans modifier l'année en cours, tapez:

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

Astuce 1: le changement de date ne sera pas répercuté immédiatement, et un changement notable ne se produira qu'une fois que l'horloge du système passera à la minute suivante. Vous pouvez forcer une mise à jour en attachant une `TIME_SET` intention `TIME_SET` à votre appel, comme ceci:

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

Astuce 2: synchroniser l'horloge d'Android avec votre machine locale:

Linux:

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

Windows (PowerShell):

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

Les deux conseils ensemble:

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

6,0

Le format SET par défaut est 'AAAAMMJJ.HHmmss'

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

Astuce: pour synchroniser l'horloge d'Android avec votre ordinateur local (basé sur Linux):

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

Ouvrir les options du développeur

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

Navigation dans votre appareil / émulateur vers la section `Developer Options` du `Developer Options` .

Génération d'une diffusion "Boot Complete"

Ceci est pertinent pour les applications qui implémentent un `BootListener` . Testez votre application en tuant votre application, puis testez avec:

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

(remplacez votre `your.package/your.app.BootListener` par des valeurs correctes).

Afficher le contenu de stockage externe / secondaire

Afficher le contenu:

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

Chemin de vue:

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

tuer un processus dans un appareil Android

Parfois, le logcat d'Android s'exécute à l'infini avec des erreurs provenant de certains processus que vous ne possédez pas, en vidant la batterie ou en rendant difficile le débogage de votre code.

Un moyen pratique de résoudre le problème sans redémarrer le périphérique consiste à localiser et à supprimer le processus à l'origine du problème.

De Logcat

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

remarquez le numéro de processus: 1550

Maintenant, nous pouvons ouvrir un shell et tuer le processus. Notez que nous ne pouvons pas tuer `root` processus `root` .

```
adb shell
```

à l'intérieur de la coquille, nous pouvons vérifier plus sur le processus en utilisant

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

et tue le si on veut:

```
kill -9 1550
```

Lire shell adb en ligne: https://riptutorial.com/fr/android/topic/9408/shell-adb

Chapitre 224: Signez votre application Android pour publication

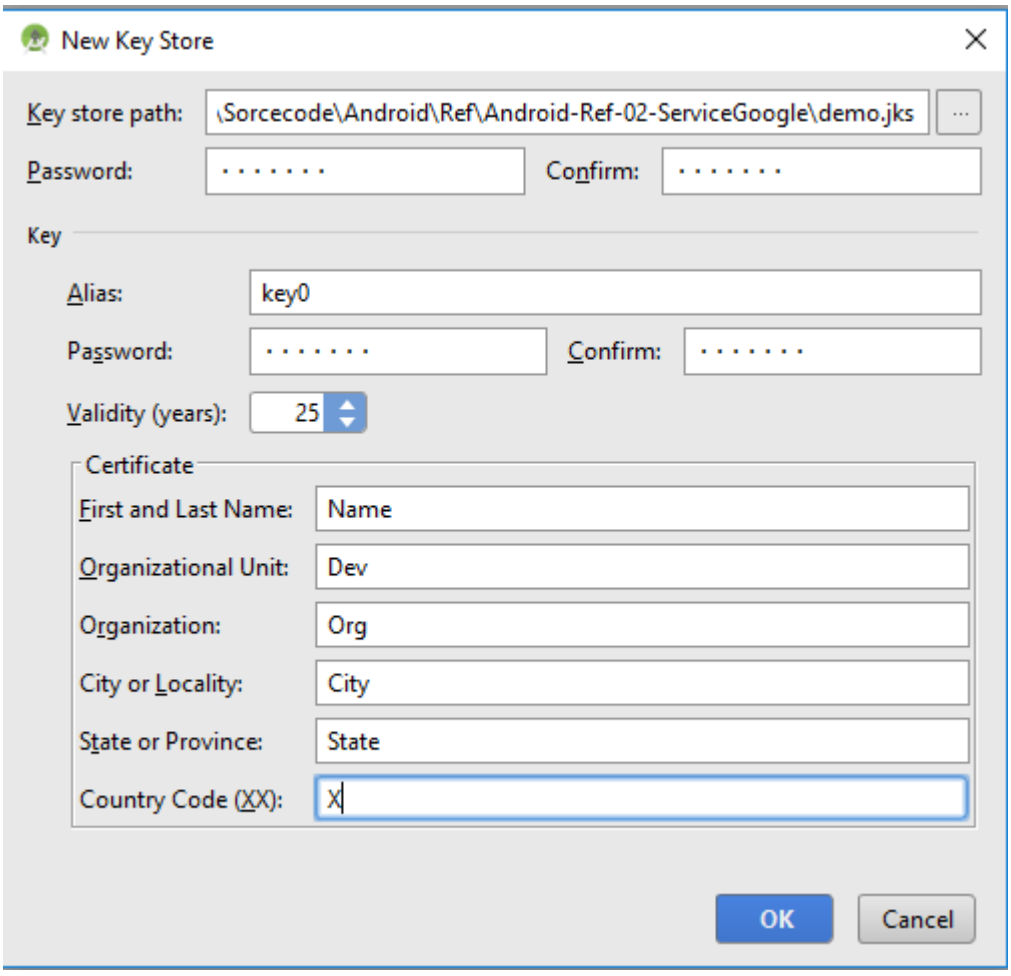
Introduction

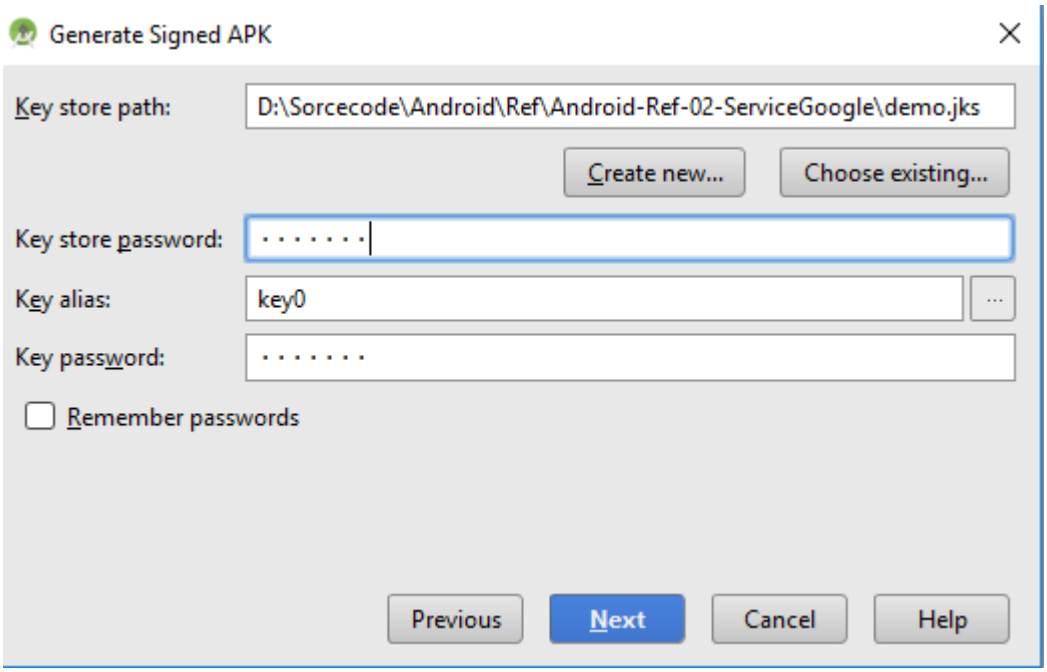
Android exige que tous les fichiers APK soient signés pour publication.

Examples

Signer votre application

1. Dans la barre de menus, cliquez sur Build> Generate Signed APK.
2. Sélectionnez le module que vous souhaitez libérer du menu déroulant et cliquez sur Suivant.
3. Pour créer un nouveau fichier de clés, cliquez sur Créer un nouveau fichier. Maintenant, remplissez les informations requises et appuyez sur OK dans New Key Store.





4. Dans les champs Generate Signed APK Wizard, vous êtes déjà renseigné si vous venez de créer un nouveau fichier de clés, sinon remplissez-le et cliquez sur next.

5. Dans la fenêtre suivante, sélectionnez une destination pour l'APK signé, sélectionnez le type de construction et cliquez sur Terminer.

Configurez le build.gradle avec la configuration de signature

Vous pouvez définir la configuration de signature pour signer le fichier `build.gradle` dans le fichier `build.gradle` .

Vous pouvez définir:

- `storeFile` : le fichier de `storeFile`
- `storePassword` : le mot de passe du `storePassword`
- `keyAlias` : un nom d'alias de clé
- `keyPassword` : un mot de passe alias de clé

Vous devez **définir** le bloc `signingConfigs` pour créer une configuration de signature:

```
android {
    signingConfigs {

        myConfig {
            storeFile file("myFile.keystore")
            storePassword "xxxx"
            keyAlias "xxxx"
            keyPassword "xxxx"
        }
    }
    //....
}
```

Vous pouvez ensuite l' **assigner** à un ou plusieurs types de construction.

```
android {

    buildTypes {
        release {
            signingConfig signingConfigs.myConfig
        }
    }
}
```

Lire Signez votre application Android pour publication en ligne:
https://riptutorial.com/fr/android/topic/9721/signez-votre-application-android-pour-publication

Chapitre 225: Snackbar

Syntaxe

- Snackbar make (Vue de la vue, texte CharSequence, durée int)
- Snackbar make (Voir la vue, int resId, int duration)

Paramètres

Paramètre	La description
vue	Affichage: vue permettant de rechercher un parent.
texte	CharSequence: Le texte à afficher. Peut être du texte formaté.
resId	int: Identifiant de la ressource de chaîne à utiliser. Peut être du texte formaté.
durée	int: combien de temps pour afficher le message. Cela peut être LENGTH_SHORT, LENGTH_LONG ou LENGTH_INDEFINITE

Remarques

Snackbar fournit des commentaires légers sur une opération. Il affiche un bref message au bas de l'écran sur le mobile et en bas à gauche sur les plus gros appareils. Les snackbars apparaissent au dessus de tous les autres éléments à l'écran et un seul peut être affiché à la fois.

Ils disparaissent automatiquement après une temporisation ou après une interaction de l'utilisateur ailleurs sur l'écran, en particulier après des interactions qui invoquent une nouvelle surface ou une nouvelle activité. Snackbar peut être glissé hors de l'écran.

Avant d'utiliser `SnackBar` vous devez ajouter la dépendance de bibliothèque du support de conception dans le fichier `build.gradle` :

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

Documentation officielle

https://developer.android.com/reference/android/support/design/widget/Snackbar.html

Examples

Créer un Snackbar simple

La création d'une `Snackbar` peut se faire comme suit:

```
Snackbar.make(view, "Text to display", Snackbar.LENGTH_LONG).show();
```

La `view` est utilisée pour trouver un parent approprié à utiliser pour afficher la `Snackbar` . En général, ce serait un `CoordinatorLayout` que vous avez défini dans votre XML, ce qui permet d'ajouter des fonctionnalités telles que swipe pour ignorer et déplacer automatiquement d'autres widgets (par exemple `FloatingActionButton`). S'il n'y a pas de `CoordinatorLayout` la vue du contenu du décor de fenêtre est utilisée.

Très souvent, nous ajoutons également une action au `Snackbar` . Un cas d'utilisation courant serait une action "Annuler".

```
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();
```

Vous pouvez créer une `Snackbar` et l'afficher plus tard:

```
Snackbar snackbar = Snackbar.make(view, "Text to display", Snackbar.LENGTH_LONG);
snackbar.show();
```

Si vous voulez changer la couleur du texte de la `Snackbar` :

```
Snackbar snackbar = Snackbar.make(view, "Text to display", Snackbar.LENGTH_LONG);
View view = snackbar .getView();
TextView textView = (TextView) view.findViewById(android.support.design.R.id.snackbar_text);
textView.setTextColor(Color.parseColor("#FF4500"));
snackbar.show();
```

Par défaut, `Snackbar` rejette sur son coup droit.Cet exemple montre comment rejeter le snackBar sur son balayage à gauche .

Snack Bar Personnalisé

Fonction pour personnaliser le snack

```
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;

    }
```

Appelez la fonction à partir de fragment ou d'activité

```
 SnackBar.makeText(MyActivity.this, "Please Locate your address at Map",
Snackbar.LENGTH_SHORT).show();
```

Snackbar avec rappel

Vous pouvez utiliser Snackbar.Callback pour écouter si le snack a été rejeté par l'utilisateur ou le délai d'attente.

```
Snackbar.make(getView(), "Hi snackbar!", Snackbar.LENGTH_LONG).setCallback( new
Snackbar.Callback() {
                @Override
                public void onDismissed(Snackbar snackbar, int event) {
                    switch(event) {
                        case Snackbar.Callback.DISMISS_EVENT_ACTION:
                            Toast.makeText(getActivity(), "Clicked the action",
Toast.LENGTH_LONG).show();
                            break;
                        case Snackbar.Callback.DISMISS_EVENT_TIMEOUT:
                            Toast.makeText(getActivity(), "Time out",
Toast.LENGTH_LONG).show();
                            break;
                    }
                }

                @Override
                public void onShown(Snackbar snackbar) {
                    Toast.makeText(getActivity(), "This is my annoying step-brother",
Toast.LENGTH_LONG).show();
                }
            }).setAction("Go!", new View.OnClickListener() {
                @Override
                public void onClick(View v) {

                }
            }).show();
```

Snackbar personnalisé

Cet exemple montre une Snackbar blanche avec l'icône Annuler personnalisée.

```
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: Lequel dois-je utiliser?

Les toasts sont généralement utilisés lorsque nous souhaitons afficher des informations sur l'utilisateur concernant une action qui a réussi (ou non) et que cette action ne nécessite aucune autre action de la part de l'utilisateur. Comme lorsqu'un message a été envoyé, par exemple:

```
Toast.makeText(this, "Message Sent!", Toast.LENGTH_SHORT).show();
```

Les snackbars sont également utilisés pour afficher une information. Mais cette fois, nous pouvons donner à l'utilisateur la possibilité de prendre des mesures. Par exemple, supposons que l'utilisateur ait supprimé une image par erreur et qu'il souhaite la récupérer. Nous pouvons fournir un Snackbar avec l'action "Annuler". Comme ça:

```
Snackbar.make(getCurrentFocus(), "Picture Deleted", Snackbar.LENGTH_SHORT)
        .setAction("Undo", new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                //Return his picture
            }
        })
        .show();
```

Conclusion: Les toasts sont utilisés lorsque nous n'avons pas besoin d'interaction avec l'utilisateur. Les snackbars sont utilisés pour autoriser les utilisateurs à effectuer une autre action ou à annuler une précédente.

Snackbar personnalisé (vue inutile)

Créer une Snackbar sans la vue de passage nécessaire à Snackbar, toutes les mises en page créent dans Android dans android.R.id.content.

```
public class CustomSnackBar {

    public static final int STATE_ERROR = 0;
    public static final int STATE_WARNING = 1;
    public static final int STATE_SUCCESS = 2;
    public static final int VIEW_PARENT = android.R.id.content;

    public CustomSnackBar(View view, String message, int actionType) {
        super();

        Snackbar snackbar = Snackbar.make(view, message, Snackbar.LENGTH_LONG);
        View sbView = snackbar.getView();
        TextView textView = (TextView)
sbView.findViewById(android.support.design.R.id.snackbar_text);
        textView.setTextColor(Color.parseColor("#ffffff"));
        textView.setTextSize(TypedValue.COMPLEX_UNIT_SP, 14);
        textView.setGravity(View.TEXT_ALIGNMENT_CENTER);
        textView.setLayoutDirection(View.LAYOUT_DIRECTION_RTL);

        switch (actionType) {
            case STATE_ERROR:
                snackbar.getView().setBackgroundColor(Color.parseColor("#F12B2B"));
                break;
            case STATE_WARNING:
                snackbar.getView().setBackgroundColor(Color.parseColor("#000000"));
                break;
            case STATE_SUCCESS:
                snackbar.getView().setBackgroundColor(Color.parseColor("#7ED321"));
                break;
        }
        snackbar.show();
    }
}
```

pour la classe d'appel

new CustomSnackBar (findViewById (CustomSnackBar.VIEW_PARENT), "message", CustomSnackBar.STATE_ERROR);

Lire Snackbar en ligne: https://riptutorial.com/fr/android/topic/1500/snackbar

Chapitre 226: Son et média Android

Examples

Comment choisir l'image et la vidéo pour api> 19

Voici un code testé pour l'image et la vidéo. Il fonctionnera pour toutes les API de moins de 19 ans et de plus de 19 ans.

Image:

```
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);
                    }
```

Vidéo:

```
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();
        }
```

Jouer des sons via SoundPool

```
public class PlaySound extends Activity implements OnTouchListener {
    private SoundPool soundPool;
    private int soundID;
    boolean loaded = false;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.main);
            View view = findViewById(R.id.textView1);
            view.setOnTouchListener(this);
            // Set the hardware buttons to control the music
            this.setVolumeControlStream(AudioManager.STREAM_MUSIC);
            // Load the sound
            soundPool = new SoundPool(10, AudioManager.STREAM_MUSIC, 0);
            soundPool.setOnLoadCompleteListener(new OnLoadCompleteListener() {
                    @Override
                    public void onLoadComplete(SoundPool soundPool, int sampleId,
                                    int status) {
                            loaded = true;
                    }
            });
            soundID = soundPool.load(this, R.raw.sound1, 1);

    }

    @Override
    public boolean onTouch(View v, MotionEvent event) {
            if (event.getAction() == MotionEvent.ACTION_DOWN) {
                    // Getting the user sound settings
                    AudioManager audioManager = (AudioManager)
getSystemService(AUDIO_SERVICE);
                    float actualVolume = (float) audioManager
                                    .getStreamVolume(AudioManager.STREAM_MUSIC);
                    float maxVolume = (float) audioManager
                                    .getStreamMaxVolume(AudioManager.STREAM_MUSIC);
                    float volume = actualVolume / maxVolume;
                    // Is the sound loaded already?
                    if (loaded) {
                            soundPool.play(soundID, volume, volume, 1, 0, 1f);
                            Log.e("Test", "Played sound");
```

```
                    }
                }
                return false;
    }
}
```

Lire Son et média Android en ligne: https://riptutorial.com/fr/android/topic/4730/son-et-media-android

Chapitre 227: SpannableString

Syntaxe

- `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 (objet quoi)
- `void setSpan (Object what, int start, int end, int flags)`
- `CharSequence subSequence (int start, int end)`
- `String toString ()`
- `SpannableString valueOf (CharSequence source)`

Examples

Ajouter des styles à un TextView

Dans l'exemple suivant, nous créons une activité pour afficher un seul TextView.

Le TextView utilisera un `SpannableString` comme contenu, qui illustrera certains des styles disponibles.

Voici ce que nous allons faire avec le texte:

- Agrandir
- Audacieux
- Souligner
- Mettre en italique
- Barré
- Coloré
- A souligné
- Montrer en exposant
- Montrer en indice
- Montrer comme lien
- Rendez-le cliquable.

```
@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);

}
```

Et le résultat ressemblera à ceci:

Multi string, avec multi couleur

Méthode: **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;
}
```

Usage:

```
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`
```

Lire SpannableString en ligne: https://riptutorial.com/fr/android/topic/10553/spannablestring

Chapitre 228: SQLite

Introduction

SQLite est un système de gestion de base de données relationnelle écrit en C. Pour commencer à utiliser des bases de données SQLite dans le cadre Android, définissez une classe qui étend SQLiteOpenHelper et personnalisez-la selon vos besoins.

Remarques

La classe `SQLiteOpenHelper` définit les `onCreate()` et `onUpgrade()` statiques. Ces méthodes sont appelées dans les méthodes correspondantes d'une sous-classe `SQLiteOpenHelper` que vous personnalisez avec vos propres tables.

Examples

Utilisation de la classe SQLiteOpenHelper

```
public class DatabaseHelper extends SQLiteOpenHelper {
    private static final String DATABASE_NAME = "Example.db";
    private static final int DATABASE_VERSION = 3;

    // For all Primary Keys _id should be used as column name
    public static final String COLUMN_ID = "_id";

    // Definition of table and column names of Products table
    public static final String TABLE_PRODUCTS = "Products";
    public static final String COLUMN_NAME = "Name";
    public static final String COLUMN_DESCRIPTION = "Description";
    public static final String COLUMN_VALUE = "Value";

    // Definition of table and column names of Transactions table
    public static final String TABLE_TRANSACTIONS = "Transactions";
    public static final String COLUMN_PRODUCT_ID = "ProductId";
    public static final String COLUMN_AMOUNT = "Amount";

    // Create Statement for Products Table
    private static final String CREATE_TABLE_PRODUCT = "CREATE TABLE " + TABLE_PRODUCTS + "
(" +
            COLUMN_ID + " INTEGER PRIMARY KEY, " +
            COLUMN_DESCRIPTION + " TEXT, " +
            COLUMN_NAME + " TEXT, " +
            COLUMN_VALUE + " REAL" +
            ");";

    // Create Statement for Transactions Table
    private static final String CREATE_TABLE_TRANSACTION = "CREATE TABLE " +
TABLE_TRANSACTIONS + "  (" +
            COLUMN_ID + " INTEGER PRIMARY KEY," +
            COLUMN_PRODUCT_ID + " INTEGER," +
            COLUMN_AMOUNT + " INTEGER," +
            " FOREIGN KEY (" + COLUMN_PRODUCT_ID + ") REFERENCES " + TABLE_PRODUCTS + "(" +
```

```
COLUMN_ID + ")" +
            ");";

    public DatabaseHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        // onCreate should always create your most up to date database
        // This method is called when the app is newly installed
        db.execSQL(CREATE_TABLE_PRODUCT);
        db.execSQL(CREATE_TABLE_TRANSACTION);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        // onUpgrade is responsible for upgrading the database when you make
        // changes to the schema. For each version the specific changes you made
        // in that version have to be applied.
        for (int version = oldVersion + 1; version <= newVersion; version++) {
            switch (version) {

                case 2:
                    db.execSQL("ALTER TABLE " + TABLE_PRODUCTS + " ADD COLUMN " +
COLUMN_DESCRIPTION + " TEXT;");
                    break;

                case 3:
                    db.execSQL(CREATE_TABLE_TRANSACTION);
                    break;
            }
        }
    }
}
```

Insérer des données dans la base de données

```
// 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
);
```

méthode onUpgrade ()

`SQLiteOpenHelper` est une classe d'assistance pour gérer la création de base de données et la gestion des versions.

Dans cette classe, la méthode `onUpgrade()` est responsable de la mise à niveau de la base de données lorsque vous modifiez le schéma. Il est appelé lorsque le fichier de base de données existe déjà, mais sa version est inférieure à celle spécifiée dans la version actuelle de l'application. Pour chaque version de base de données, les modifications spécifiques que vous avez apportées doivent être appliquées.

```
@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;
        }
    }
}
```

Lecture des données d'un curseur

Voici un exemple de méthode qui vivrait dans une sous-classe `SQLiteOpenHelper` . Il utilise la chaîne `searchTerm` pour filtrer les résultats, itérer le contenu du curseur et renvoie ces contenus dans une `List` d'objets de `Product` .

Tout d'abord, définissez la classe de `Product` POJO qui sera le conteneur pour chaque ligne extraite de la base de données:

```
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;
  }
}
```

Ensuite, définissez la méthode qui interrogera la base de données et renverrez une `List` d'objets de `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();
    }
}
```

Créer un contrat, une aide et un fournisseur pour SQLite dans 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;
    }

}
```

Comment utiliser:

```
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;
}
```

Mise à jour d'une ligne dans une table

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

Effectuer une transaction

Les transactions peuvent être utilisées pour apporter plusieurs modifications à la base de données de manière atomique. Toute transaction normale suit ce modèle:

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

L'appel de `beginTransaction()` intérieur d'une transaction active n'a aucun effet.

Supprimer les lignes de la table

Pour supprimer toutes les lignes de la table

```
//get writable database
```

```
SQLiteDatabase db = openHelper.getWritableDatabase();

db.delete(TABLE_NAME, null, null);
db.close();
```

Pour supprimer toutes les lignes de la table et obtenir le nombre de lignes supprimées dans la valeur de retour

```
//get writable database
SQLiteDatabase db = openHelper.getWritableDatabase();

int numRowsDeleted = db.delete(TABLE_NAME, String.valueOf(1), null);
db.close();
```

Pour supprimer une ou plusieurs lignes avec la condition `WHERE`

```
//get writable database
SQLiteDatabase db = openHelper.getWritableDatabase();

String whereClause = KEY_NAME + " = ?";
String[] whereArgs = new String[]{String.valueOf(KEY_VALUE)};

//for multiple condition, join them with AND
//String whereClause = KEY_NAME1 + " = ? AND " + KEY_NAME2 + " = ?";
//String[] whereArgs = new String[]{String.valueOf(KEY_VALUE1), String.valueOf(KEY_VALUE2)};

int numRowsDeleted = db.delete(TABLE_NAME, whereClause, whereArgs);
db.close();
```

Stocker l'image dans SQLite

Configuration de la base de données

```
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);
    }
}
```

Insérer dans la base de données:

```
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 );
}
```

Récupération de données :

```
 byte[] image = cursor.getBlob(1);
```

Remarque:

1. Avant de l'insérer dans une base de données, vous devez d'abord convertir votre image Bitmap en tableau d'octets, puis l'appliquer à l'aide d'une requête de base de données.
2. Lors de la récupération à partir d'une base de données, vous disposez certainement d'un tableau d'octets d'image. Vous devez donc convertir le tableau d'octets en image d'origine. Donc, vous devez utiliser BitmapFactory pour décoder.

Vous trouverez ci-dessous un cours utilitaire qui, je l’espère, pourrait vous aider:

```
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);
    }
}
```

Créer une base de données à partir du dossier des ressources

Placez votre fichier dbname.sqlite ou dbname.db dans le dossier assets de votre projet.

```
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) {

        }
    }
```

Voici comment accéder à un objet de base de données pour votre activité.

```
// Create Databasehelper class object in your activity.
   private Databasehelper db;
```

Ensuite, dans la méthode onCreate, initialisez-la et appelez la méthode createDatabase () comme indiqué ci-dessous.

```
db = new Databasehelper(MainActivity.this);
        try {
            db.createDataBase();
        } catch (Exception e) {
            e.printStackTrace();
        }
```

Effectuez toutes les opérations d'insertion, de mise à jour, de suppression et de sélection comme indiqué ci-dessous.

```
String query = "select Max(Id) as Id from " + TABLE_NAME;
        db.openDataBase();
        int count = db.getId(query);
        db.close();
```

Exportation et importation d'une base de données

Vous voudrez peut-être importer et exporter votre base de données pour bacukups par exemple. Ne pas oublier les autorisations.

```
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();
    }
}
```

Insert en vrac

Voici un exemple d'insertion de gros morceaux de données en même temps. Toutes les données que vous souhaitez insérer sont regroupées dans un tableau ContentValues.

```
@Override
public int bulkInsert(Uri uri, ContentValues[] values) {
    int count = 0;
    String table = null;

    int uriType = IChatContract.MessageColumns.uriMatcher.match(uri);
    switch (uriType) {
        case IChatContract.MessageColumns.MESSAGES:
            table = IChatContract.MessageColumns.TABLE_NAME;
            break;
    }
    mDatabase.beginTransaction();
    try {
        for (ContentValues cv : values) {
            long rowID = mDatabase.insert(table, " ", cv);
            if (rowID <= 0) {
```

```
                throw new SQLException("Failed to insert row into " + uri);
            }
        }
        mDatabase.setTransactionSuccessful();
        getContext().getContentResolver().notifyChange(uri, null);
        count = values.length;
    } finally {
        mDatabase.endTransaction();
    }
    return count;
}
```

Et voici un exemple d'utilisation:

```
ContentResolver resolver = mContext.getContentResolver();
ContentValues[] valueList = new ContentValues[object.size()];
//add whatever you like to the valueList
resolver.bulkInsert(IChatContract.MessageColumns.CONTENT_URI, valueList);
```

Lire SQLite en ligne: https://riptutorial.com/fr/android/topic/871/sqlite

Chapitre 229: Stockage de fichiers dans le stockage interne et externe

Syntaxe

- FileOutputStream openFileInput (nom de la chaîne)
- FileOutputStream openFileOutput (Nom de la chaîne, mode int)
- Fichier (répertoire du fichier, nom de la chaîne)
- Fichier (chemin de chaîne)
- Fichier getExternalStoragePublicDirectory (type de chaîne)
- Fichier getExternalFilesDir (type de chaîne)

Paramètres

Paramètre	Détails
prénom	Le nom du fichier à ouvrir. REMARQUE: ne peut pas contenir de séparateurs de chemin
mode	Mode de fonctionnement. Utilisez `MODE_PRIVATE` pour l'opération par défaut et `MODE_APPEND` pour ajouter un fichier existant. Les autres modes incluent `MODE_WORLD_READABLE` et `MODE_WORLD_WRITEABLE` , tous deux obsolètes dans l'API 17.
dir	Répertoire du fichier pour créer un nouveau fichier dans
chemin	Chemin pour spécifier l'emplacement du nouveau fichier
type	Type de répertoire de fichiers à récupérer. Peut être `null` ou l'un des suivants: `DIRECTORY_MUSIC` , `DIRECTORY_PODCASTS` , `DIRECTORY_RINGTONES` , `DIRECTORY_ALARMS` , `DIRECTORY_NOTIFICATIONS` , `DIRECTORY_PICTURES` ou `DIRECTORY_MOVIES`

Examples

Utilisation du stockage interne

Par défaut, tous les fichiers que vous enregistrez dans Internal Storage sont privés pour votre application. Ils ne sont pas accessibles par d'autres applications, ni par l'utilisateur dans des circonstances normales. **Ces fichiers sont supprimés lorsque l'utilisateur désinstalle l'application** .

Pour écrire du texte dans un fichier

```
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();
}
```

Pour ajouter du texte à un fichier existant

Utilisez `Context.MODE_APPEND` pour le paramètre de mode de `openFileOutput`

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

Utilisation du stockage externe

Le stockage "externe" est un autre type de stockage que nous pouvons utiliser pour enregistrer des fichiers sur le périphérique de l'utilisateur. Il présente certaines différences clés par rapport au stockage "interne", à savoir:

- Il n'est pas toujours disponible. Dans le cas d'un support amovible (carte SD), l'utilisateur peut simplement supprimer le stockage.
- Ce n'est pas privé. L'utilisateur (et les autres applications) ont accès à ces fichiers.
- Si l'utilisateur désinstalle l'application, les fichiers que vous enregistrez dans le répertoire extrait avec `getExternalFilesDir()` seront supprimés.

Pour utiliser le stockage externe, vous devez d'abord obtenir les autorisations appropriées. Vous devrez utiliser:

- `android.permission.WRITE_EXTERNAL_STORAGE` pour lire et écrire
- `android.permission.READ_EXTERNAL_STORAGE` pour juste lire

Pour accorder ces autorisations, vous devrez les identifier dans votre `AndroidManifest.xml` tant que tel.

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

> REMARQUE: Étant donné qu'il s'agit d' autorisations dangereuses si vous utilisez l' **API niveau 23** ou supérieur, vous devrez demander les **autorisations à l'exécution** .

Avant d'essayer d'écrire ou de lire à partir du stockage externe, vous devez toujours vérifier que le support de stockage est disponible.

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

```
        state.equals(Environment.MEDIA_MOUNTED_READ_ONLY)) {
    // Available to at least read
}
```

Lorsque vous écrivez des fichiers sur le stockage externe, vous devez décider si le fichier doit être reconnu comme public ou privé. Bien que ces deux types de fichiers soient toujours accessibles à l'utilisateur et à d'autres applications sur le périphérique, il existe une distinction essentielle entre eux.

Les fichiers publics doivent rester sur le périphérique lorsque l'utilisateur désinstalle l'application. Un exemple de fichier devant être enregistré en tant que public serait les photos prises via votre application.

Les fichiers privés doivent tous être supprimés lorsque l'utilisateur désinstalle l'application. Ces types de fichiers seraient spécifiques à l'application et ne seraient pas utiles à l'utilisateur ou à d'autres applications. Ex. fichiers temporaires téléchargés / utilisés par votre application.

Voici comment accéder au répertoire `Documents` pour les fichiers publics et privés.

Publique

```
// 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();
```

Privé

```
// 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: Stockage interne et externe - Clarification de la terminologie

Les développeurs Android (principalement les débutants) ont été confus en ce qui concerne la terminologie de stockage interne et externe. Il y a beaucoup de questions sur Stackoverflow à propos de la même chose. Cela est principalement dû au fait que la *terminologie* selon la documentation Google / Android officielle est très différente de celle de l'utilisateur normal du système d'exploitation Android. J'ai donc pensé que documenter cela aiderait.

Ce que nous pensons - Terminologie de l'utilisateur (UT)

Stockage interne (UT)	Stockage externe (UT)
Mémoire interne intégrée au téléphone	carte SD (Secure Digital) amovible ou stockage micro SD

Stockage interne (UT)	Stockage externe (UT)
Exemple: mémoire interne de 32 Go de Nexus 6P.	**Exemple:** espace de stockage dans des cartes SD amovibles fournies par des fournisseurs tels que samsung, sandisk, strontium, transcend et autres

Mais, **selon Android Documentation / Guide - Terminologie de Google (GT)**

Stockage interne (GT):

> Par défaut, les fichiers enregistrés dans le stockage interne sont privés pour votre application et les autres applications ne peuvent pas y accéder (ni l'utilisateur).

Stockage externe (GT):

> Cela peut être un support de stockage amovible (comme une carte SD) ou un stockage interne (non amovible).

Le stockage externe (GT) peut être classé en deux types:

Stockage externe primaire	Stockage externe secondaire ou stockage amovible (GT)
Ceci est identique à la mémoire interne intégrée du téléphone (ou) au stockage interne (UT)	Ceci est identique au stockage sur carte micro SD amovible (ou) Stockage externe (UT)
Exemple: mémoire interne de 32 Go de Nexus 6P.	**Exemple:** espace de stockage dans des cartes SD amovibles fournies par des fournisseurs tels que samsung, sandisk, strontium, transcend et autres
Ce type de stockage est accessible sur Windows PC en connectant votre téléphone au PC via un câble USB et en sélectionnant *Camera (PTP)* dans la notification des options USB.	Ce type de stockage est accessible sur Windows PC en connectant votre téléphone au PC via un câble USB et en sélectionnant *Transfert de fichiers* dans la notification des options USB.

En un mot,

Stockage externe (GT) = Stockage interne (UT) et stockage externe (UT)

Stockage amovible (GT) = Stockage externe (UT)

Le stockage interne (GT) n'a pas de terme en UT.

Laissez-moi vous expliquer clairement,

Stockage interne (GT): par défaut, les fichiers enregistrés sur le stockage interne sont privés de

votre application et les autres applications ne peuvent pas y accéder. Votre utilisateur de l'application ne peut pas non plus y accéder à l'aide du gestionnaire de fichiers. même après avoir activé l'option "Afficher les fichiers cachés" dans le gestionnaire de fichiers. Pour accéder aux fichiers dans le stockage interne (GT), vous devez rooter votre téléphone Android. De plus, lorsque l'utilisateur désinstalle votre application, ces fichiers sont supprimés / supprimés.

Donc, le stockage interne (GT) n'est **pas** ce que nous pensons comme la mémoire interne 32/64 Go de Nexus 6P

En règle générale, **l'emplacement du stockage interne (GT)** serait:
`/data/data/your.application.package.appname/someDirectory/`

Stockage externe (GT):

> Chaque appareil compatible Android prend en charge un «stockage externe» partagé que vous pouvez utiliser pour enregistrer des fichiers. Les fichiers enregistrés sur le stockage externe sont lisibles dans le monde entier et peuvent être modifiés par l'utilisateur lorsqu'ils permettent le stockage de masse USB pour transférer des fichiers sur un ordinateur.

Emplacement de stockage externe (GT): il peut se trouver *n'importe où* dans votre stockage interne ou dans votre stockage amovible, par exemple une carte micro SD. Cela dépend de l'OEM de votre téléphone et aussi de la version du système d'exploitation Android.

Pour lire ou écrire des fichiers sur le stockage externe (GT), votre application doit acquérir les autorisations système `READ_EXTERNAL_STORAGE` ou `WRITE_EXTERNAL_STORAGE` .

Par exemple:

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

> Si vous devez à la fois lire et écrire des fichiers, vous devez demander uniquement l'autorisation `WRITE_EXTERNAL_STORAGE` , car cela nécessite implicitement un accès en lecture.

Dans le **stockage externe (GT)** , vous pouvez également enregistrer des fichiers qui sont **app-private**

Mais,

> Lorsque l'utilisateur désinstalle votre application, ce répertoire et tout son contenu sont supprimés.

Quand avez-vous besoin de sauvegarder des fichiers qui sont **app-private** dans **External Storage (GT)** ?

> Si vous gérez des fichiers qui ne sont pas destinés à d'autres applications (par

exemple, des textures graphiques ou des effets sonores utilisés uniquement par votre application), vous devez utiliser un répertoire de stockage privé sur le stockage externe.

Depuis Android 4.4, la lecture ou l'écriture de fichiers dans les répertoires privés de votre application ne nécessite pas les autorisations `READ_EXTERNAL_STORAGE` ou `WRITE_EXTERNAL_STORAGE` . Vous pouvez donc déclarer que l'autorisation doit être demandée uniquement sur les versions inférieures d'Android en ajoutant l'attribut `maxSdkVersion` :

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

Méthodes à stocker dans le stockage interne (GT):

Ces deux méthodes sont présentes dans la classe Context

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

File getFilesDir ()
```

Méthodes à stocker dans le stockage externe primaire, c'est-à-dire le stockage interne (UT):

```
File getExternalStorageDirectory ()

File getExternalFilesDir (String type)

File getExternalStoragePublicDirectory (String type)
```

Au début, tout le monde utilisait Environment.getExternalStorageDirectory () , qui indiquait la **racine** du **stockage externe primaire** . En conséquence, le stockage externe principal était rempli de contenu aléatoire.

Plus tard, ces deux méthodes ont été ajoutées:

1. Dans la classe `Context` , ils ont ajouté getExternalFilesDir () , pointant vers un **répertoire spécifique** à l' **application** sur le stockage externe principal. Ce répertoire et son contenu **seront supprimés** lors de la désinstallation de l'application.

2. Environment.getExternalStoragePublicDirectory () pour les emplacements centralisés permettant de stocker des types de fichiers connus, tels que des photos et des films. Ce répertoire et son contenu **ne seront PAS supprimés** lors de la désinstallation de l'application.

Méthodes de stockage dans le stockage amovible (GT), c.-à-d. Carte micro SD

Avant le **niveau 19 de l'API** , il n'y avait **pas de moyen officiel** de stocker sur une carte SD.

Mais, beaucoup pourraient le faire en utilisant des bibliothèques non officielles ou des API.

Officiellement, une méthode a été introduite dans la classe `Context` au niveau 19 de l'API (Android version 4.4 - Kitkat).

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

> Il renvoie des chemins d'accès absolus aux répertoires spécifiques à l'application sur tous les périphériques de stockage partagés / externes sur lesquels l'application peut placer des fichiers persistants qu'elle possède. Ces fichiers sont internes à l'application et ne sont généralement pas visibles par l'utilisateur en tant que support.

Cela signifie qu'il renverra des chemins vers les **deux** types de stockage externe (GT) - mémoire interne et carte micro SD. Généralement, le **deuxième chemin** serait le chemin de stockage de la carte micro SD (mais pas toujours). Vous devez donc le vérifier en exécutant le code avec cette méthode.

Exemple avec l'extrait de code:

J'ai créé un nouveau projet Android avec une activité vide, écrit le code suivant à l'intérieur

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

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

    File external_m1 =  Environment.getExternalStorageDirectory();

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

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

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

Après avoir exécuté le code ci-dessus,

Sortie:

```
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
```

Remarque: J'ai connecté mon téléphone à un PC Windows; activé les deux options de développeur, le débogage USB et ensuite exécuté ce code. Si vous **ne connectez pas votre téléphone** mais au lieu de cela exécuter sur l' **émulateur Android** , votre sortie peut varier. Mon modèle de téléphone est Coolpad Note 3 - fonctionnant sur Android 5.1

Emplacements de stockage sur mon téléphone:

Emplacement de stockage Micro SD : `/storage/sdcard1`

Emplacement de stockage interne (UT) : `/storage/sdcard0` .

Notez que `/sdcard` & `/storage/emulated/0` pointe également vers le stockage interne (UT). Mais ce sont des liens symboliques vers `/storage/sdcard0` .

Pour bien comprendre les différents chemins de stockage sous Android, veuillez parcourir cette réponse

Disclaimer: Tous les chemins de stockage mentionnés ci-dessus sont des chemins sur **mon** téléphone. Vos fichiers **ne** peuvent **pas** être stockés sur les mêmes chemins de stockage. Parce que les emplacements de stockage / chemins peuvent varier sur les autres téléphones mobiles en fonction de votre fournisseur, du fabricant et des différentes versions du système d'exploitation Android.

Enregistrer la base de données sur la carte SD (Backup DB on 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;
    }
}
```

appelle cette méthode de cette façon:

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

Récupérer le répertoire du périphérique:

Tout d'abord, ajoutez l'autorisation de stockage pour lire / récupérer le répertoire du périphérique.

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

Créer une classe de modèle

```
//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;
    }
}
```

Créer une liste en utilisant le modèle de répertoire pour ajouter des données de répertoire.

```
//define list to show directory

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

Récupère le répertoire en utilisant la méthode suivante.

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

Imprimer la liste des répertoires dans le journal.

```
//print directory list in logs

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

Usage

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

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

Pour récupérer les fichiers / dossiers internes d'un répertoire spécifique, utilisez la même méthode, changez simplement d'argument, transmettez le chemin sélectionné dans l'argument et gérez la réponse pour le même.

Pour obtenir une extension de fichier:

```
private String getExtension(String filename) {

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

    return extension;
}
```

Lire Stockage de fichiers dans le stockage interne et externe en ligne:
https://riptutorial.com/fr/android/topic/150/stockage-de-fichiers-dans-le-stockage-interne-et-externe

Chapitre 230: Stratégie de mode strict: outil permettant de détecter le bogue lors de la compilation.

Introduction

Strict Mode est une classe spéciale introduite dans Android 2.3 pour le débogage. Ces outils de développement détectent les choses accidentellement et les portent à notre attention afin que nous puissions les corriger. Il est le plus souvent utilisé pour intercepter le disque ou l'accès réseau accidentel sur le thread principal des applications, où les opérations d'interface utilisateur sont reçues et les animations ont lieu. StrictMode est fondamentalement un outil pour attraper le bug dans le mode Compile Time.

Remarques

StrictMode est fondamentalement un outil pour attraper le bug dans le mode Compile Time. Grâce à cela, nous pouvons éviter les fuites de mémoire dans nos applications.

Examples

Le code ci-dessous sert à configurer le StrictMode for Thread Policies. Ce code doit être défini aux points d'entrée de notre application.

```
StrictMode.setThreadPolicy(new StrictMode.ThreadPolicy.Builder()
    .detectDiskWrites()
    .penaltyLog() //Logs a message to LogCat
    .build())
```

Le code ci-dessous traite des fuites de mémoire, comme il détecte lorsque dans SQLLite la finalisation est appelée ou non.

```
StrictMode.setVmPolicy(new StrictMode.VmPolicy.Builder()
    .detectActivityLeaks()
    .detectLeakedClosableObjects()
    .penaltyLog()
    .build());
```

Lire Stratégie de mode strict: outil permettant de détecter le bogue lors de la compilation. en ligne: https://riptutorial.com/fr/android/topic/8756/strategie-de-mode-strict--outil-permettant-de-detecter-le-bogue-lors-de-la-compilation-

Chapitre 231: Studio Android

Examples

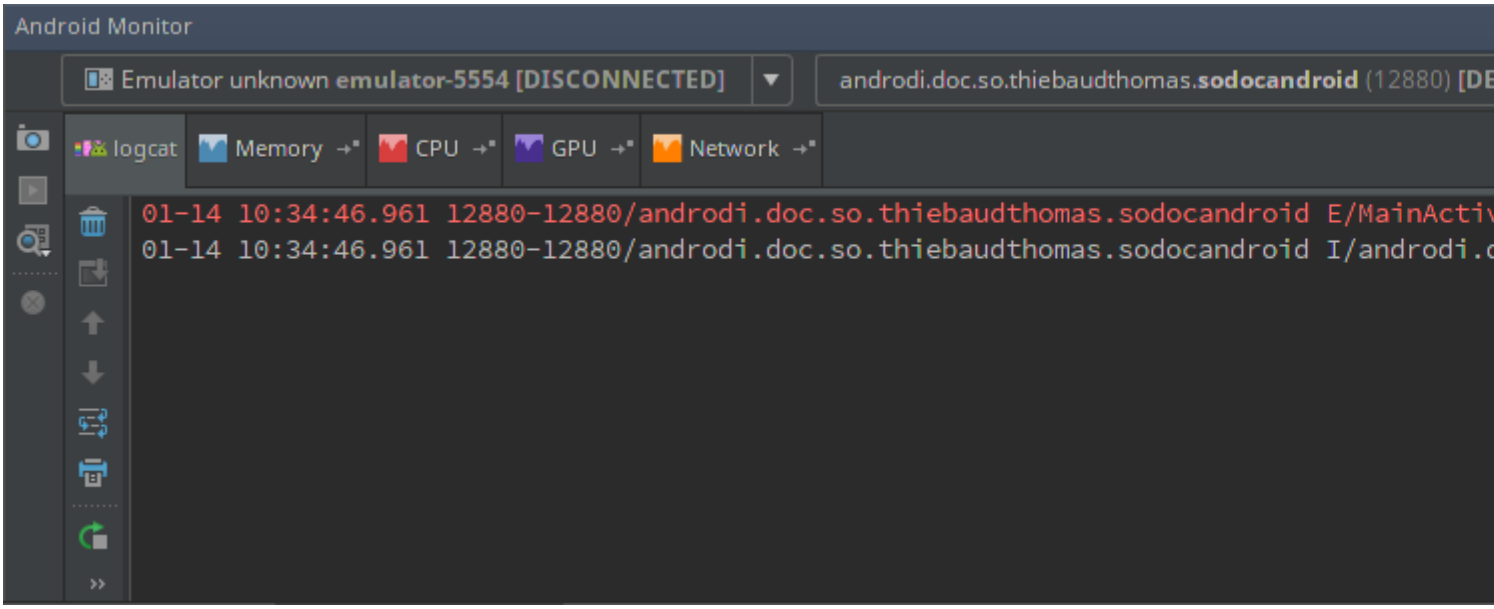
Filtrer les journaux de l'interface utilisateur

Les journaux Android peuvent être filtrés directement depuis l'interface utilisateur. En utilisant ce code

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

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

Si j'utilise le regex `TAG1|TAG2` et le niveau `verbose` je reçois

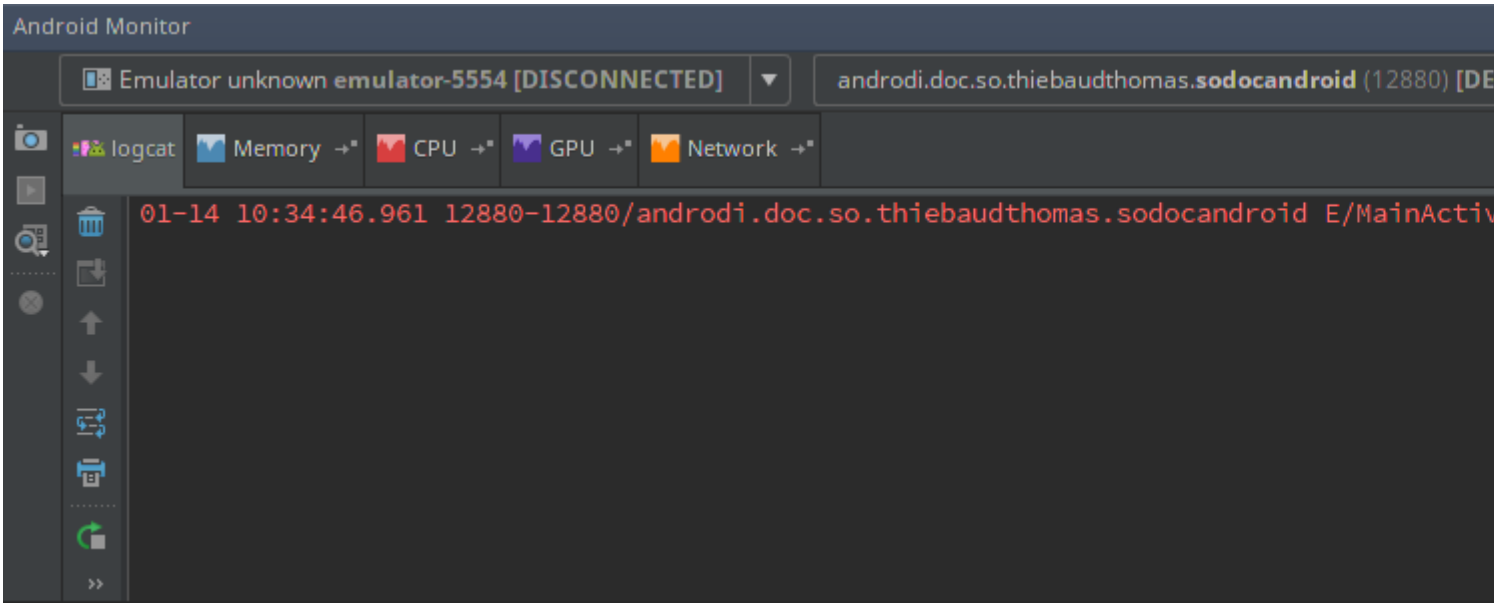
```
01-14 10:34:46.961 12880-12880/android.doc.so.thiebaudthomas.sodocandroid E/MainActivity: Log
from onCreate method with TAG1
01-14 10:34:46.961 12880-12880/android.doc.so.thiebaudthomas.sodocandroid
I/androdi.doc.so.thiebaudthomas.sodocandroid.MainActivity: Log from onCreate method with TAG2
```



Le niveau peut être défini pour obtenir des journaux avec un niveau donné et au-dessus. Par exemple, le niveau `verbose` intercepte les journaux `verbose, debug, info, warn, error and assert` .

En utilisant le même exemple, si je mets le niveau sur `error` , je ne reçois que

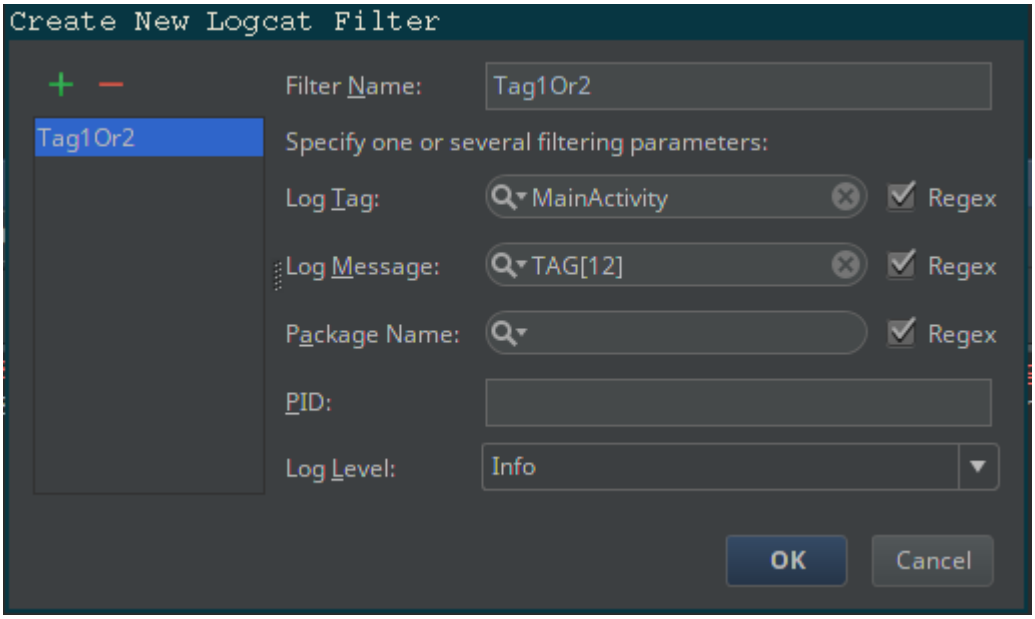
```
01-14 10:34:46.961 12880-12880/androdi.doc.so.thiebaudthomas.sodocandroid E/MainActivity: Log from onCreate method with TAG1
```



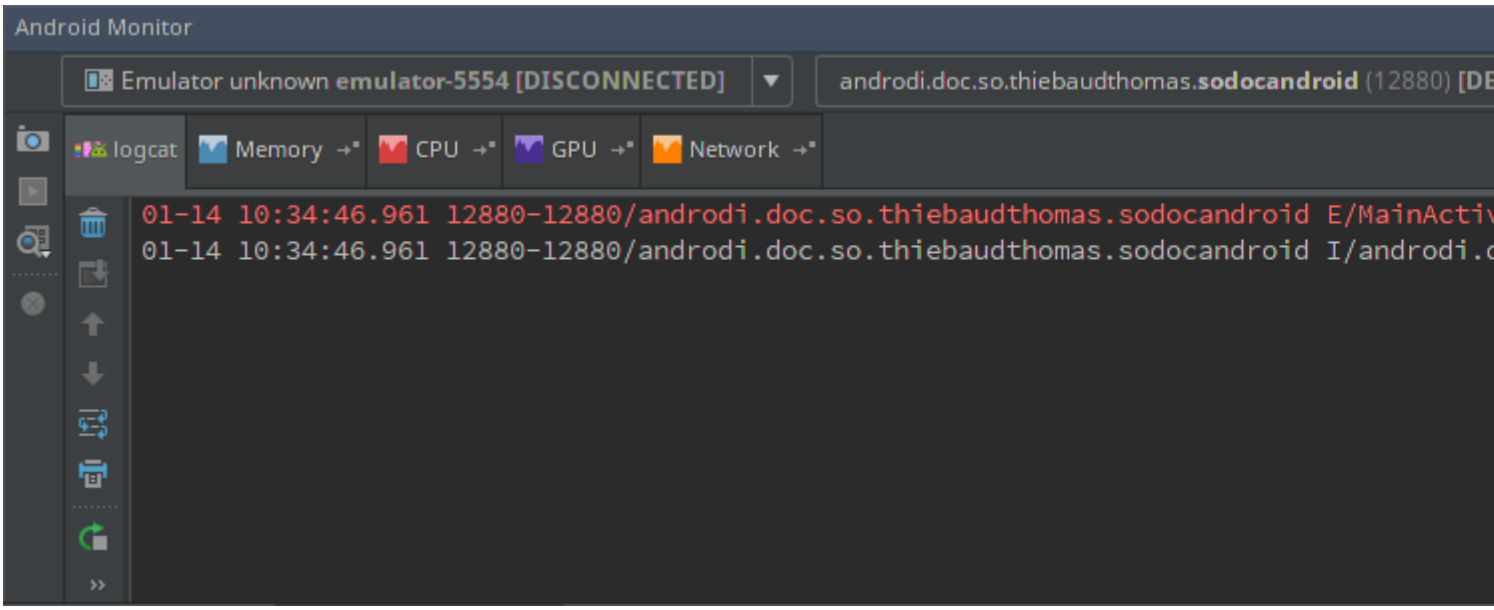
Créer une configuration de filtres

Des filtres personnalisés peuvent être définis et enregistrés à partir de l'interface utilisateur. Dans l'onglet `AndroidMonitor` , cliquez sur la liste déroulante de droite (doit contenir `Show only selected application` ou `No filters`) et sélectionner `Edit filter configuration` .

Entrez le filtre que vous voulez

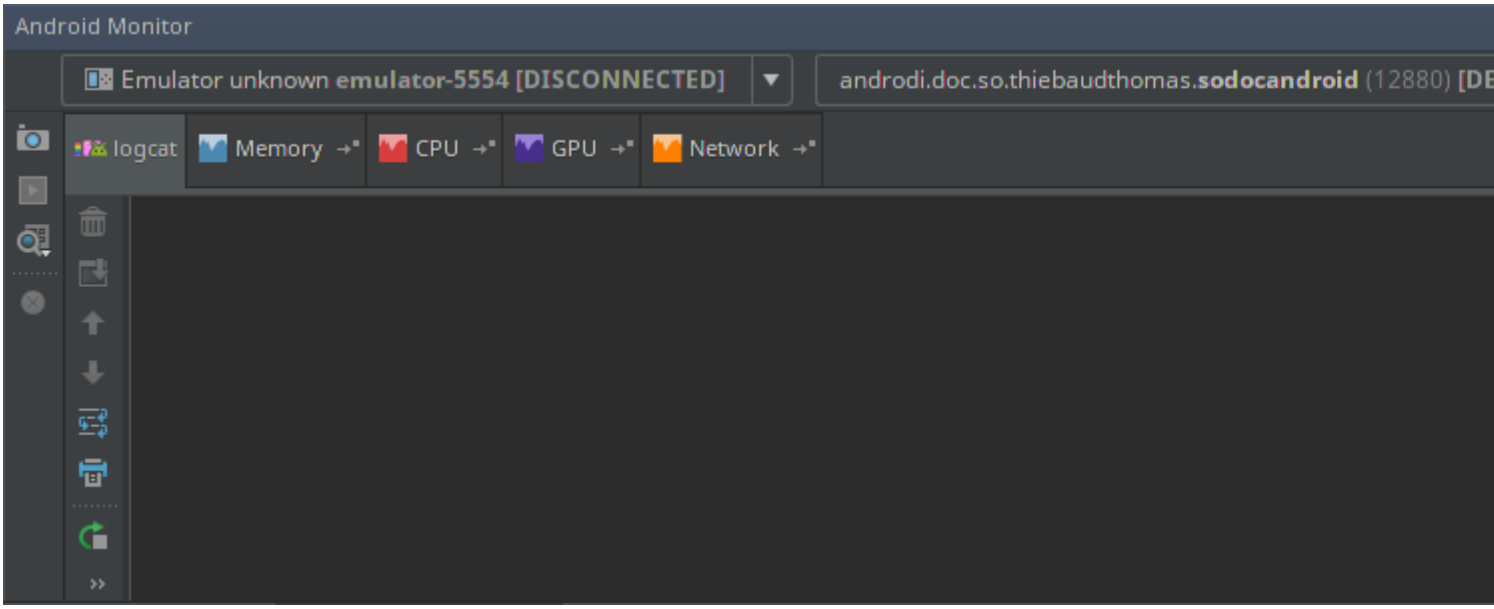


Et l'utiliser (vous pouvez le sélectionner à partir du même menu déroulant)

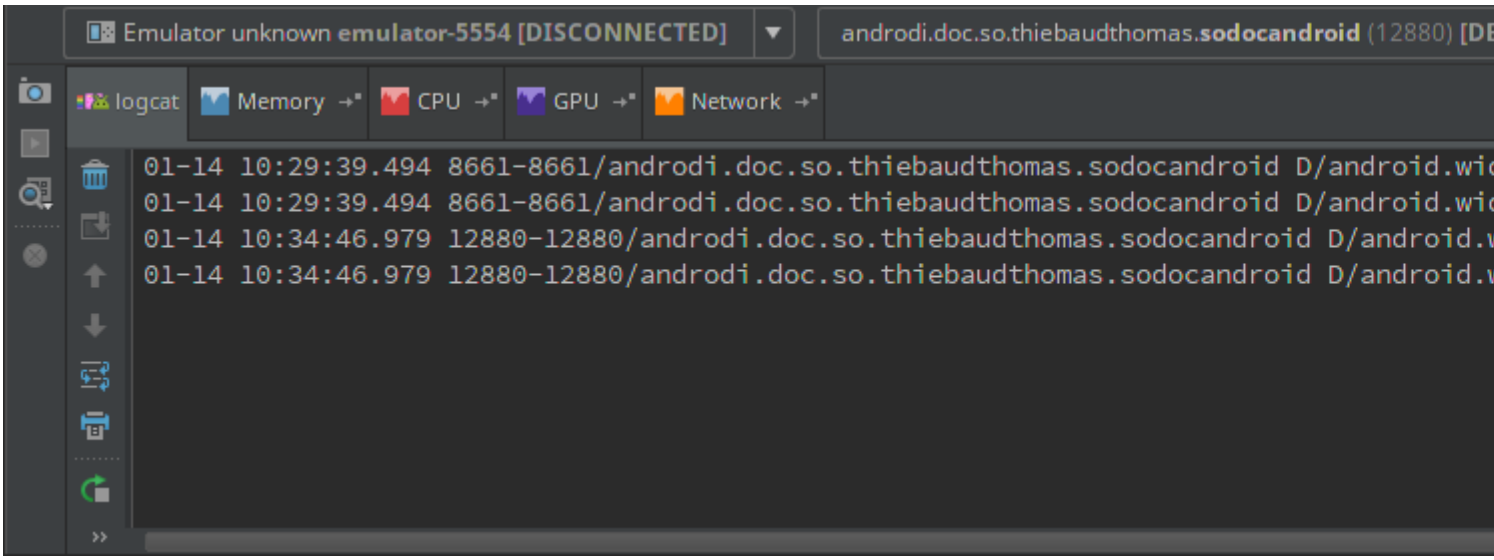


Important Si vous ajoutez une entrée dans la barre de filtre, android studio tiendra compte à la fois de votre filtre et de votre saisie.

Avec l'entrée et le filtre, il n'y a pas de sortie



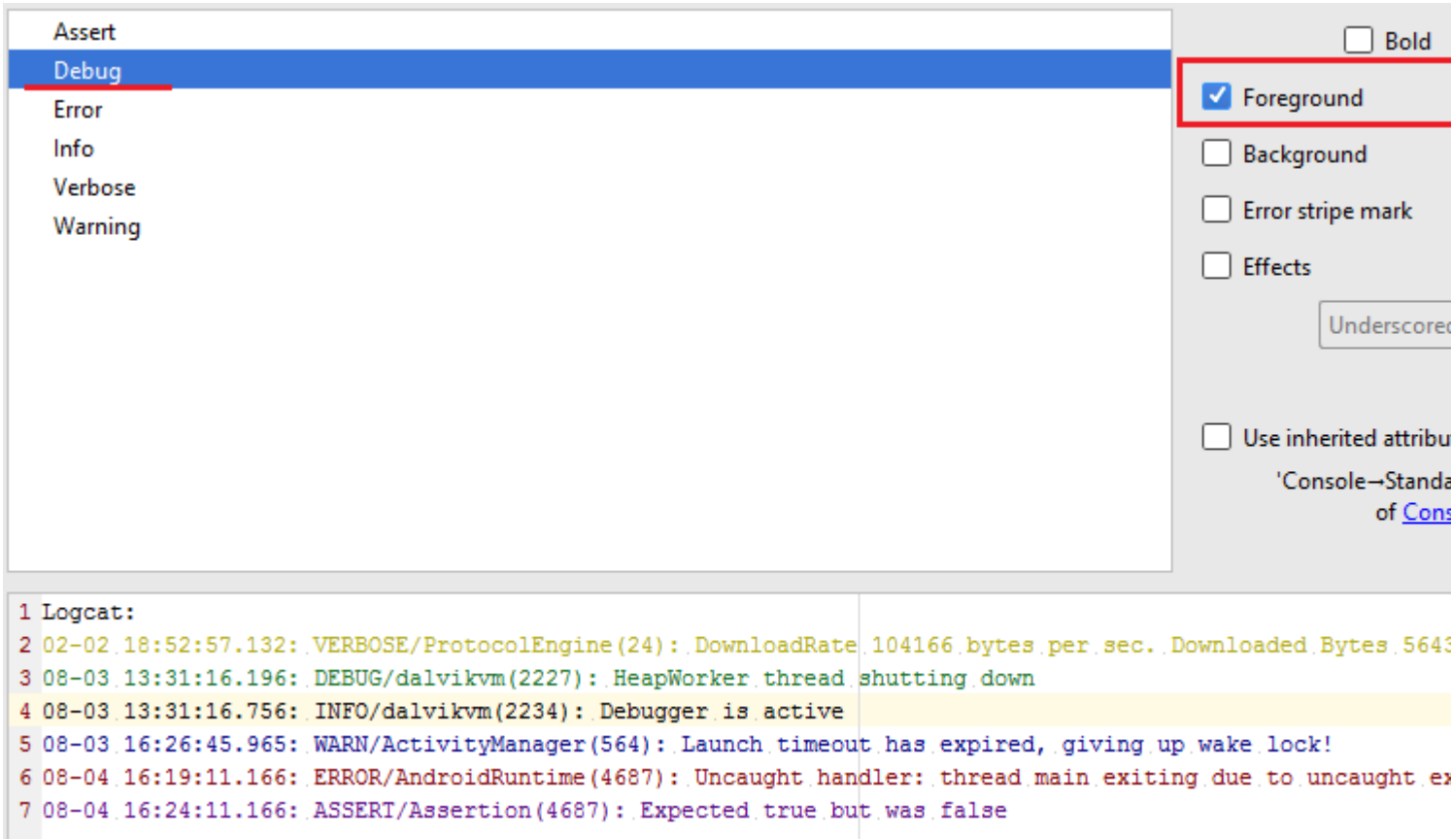
Sans filtre, il y a des sorties



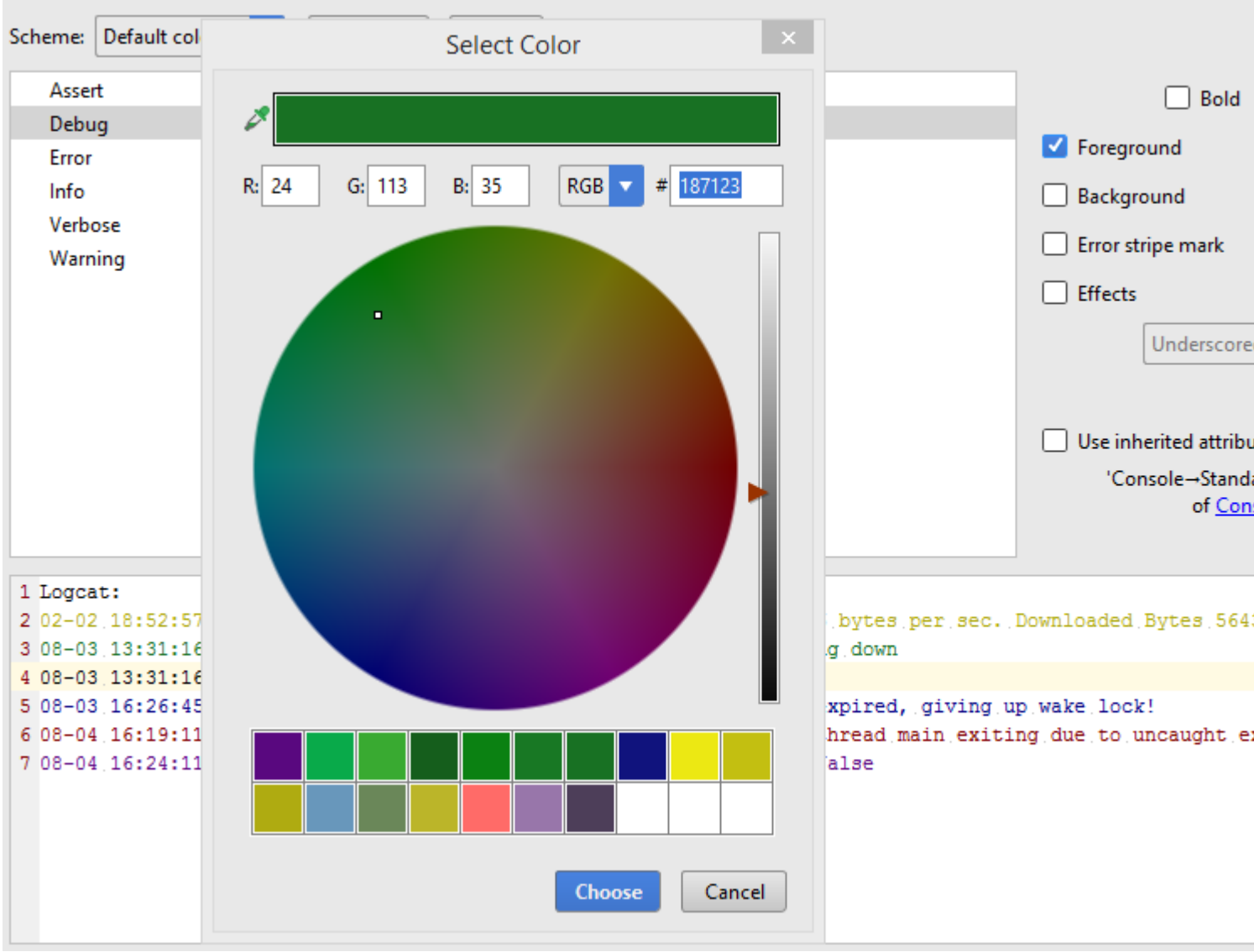
Couleurs personnalisées du message logcat en fonction de l'importance du message

Allez dans Fichier -> Paramètres -> Editeur -> Couleurs et polices -> Android Logcat

Changez les couleurs selon vos besoins:

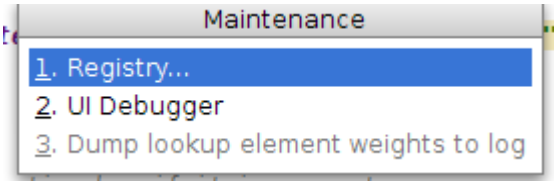


Choisissez la couleur appropriée:

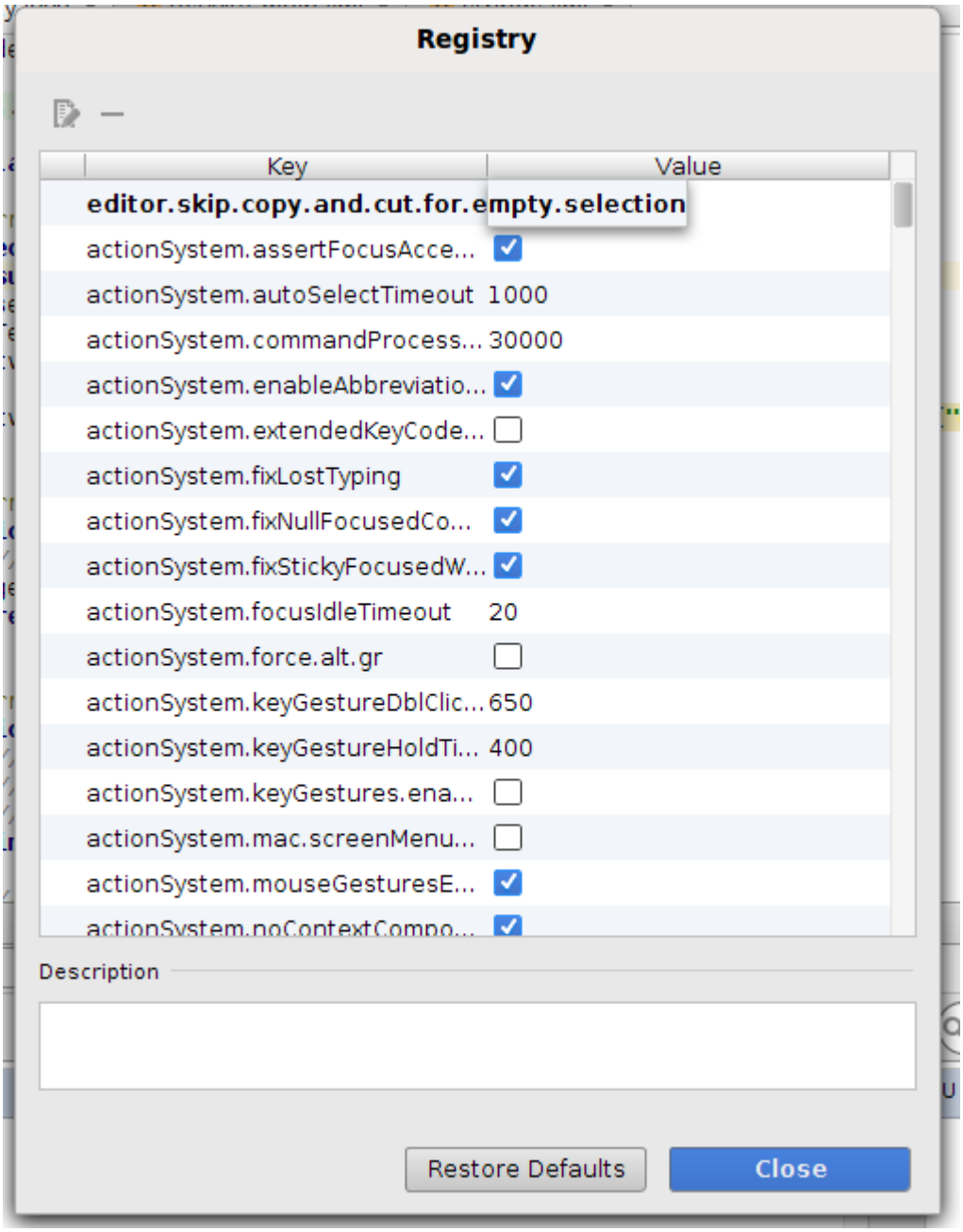


Activer / désactiver la copie de ligne vide

`ctrl + alt + shift + /` (`cmd + alt + shift + /` sur `MacOS`) devrait vous montrer la boîte de dialogue suivante:



En cliquant sur le `Registry` vous obtiendrez



La clé que vous souhaitez activer / désactiver est

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

Testé sur `Linux Ubuntu` et `MacOS` .

Raccourcis utiles pour Android Studio

Voici quelques-uns des raccourcis les plus courants / utiles.

Ceux-ci sont basés sur la carte de raccourci IntelliJ par défaut. Vous pouvez passer à d'autres raccourcis IDE courants via `File -> Settings -> Keymap -> <Choose Eclipse/Visual Studio/etc from Keymaps dropdown>`

action	Raccourci
Code de format	CTRL + ALT + L
Ajouter des méthodes non implémentées	CTRL + I
Afficher logcat	ALT + 6
Construire	CTRL + F9
Construire et courir	CTRL + F10
Trouver	CTRL + F
Trouver dans le projet	CTRL + MAJ + F
Trouver et remplacer	CTRL + R
Trouver et remplacer dans le projet	CTRL + MAJ + R
Remplacer les méthodes	CTRL + O
Montrer projet	ALT + 1
Cacher le projet - logcat	SHIFT + ESC
Tout réduire	CTRL + MAJ + NumPad +
Afficher les points de débogage	CTRL + MAJ + F8
Développer tout	CTRL + MAJ + NumPad -
Ouvrir les paramètres	ALT + s
Sélectionner la cible (ouvrir le fichier en cours dans la vue Projet)	ALT + F1 → ENTER
Rechercher partout	SHIFT → SHIFT (double décalage)
Code | Surround avec	CTRL → ALT + T
Créer un code de méthode de formulaire sélectionné	ALT + CTRL

Refactor:

action	Raccourci
Refactor This (menu / sélecteur pour toutes les actions de refactor applicables de l'élément en cours)	Mac CTRL + T - Win / Linux CTRL + ALT + T

action	Raccourci
Renommer	`SHIFT` + `F6`
Méthode d'extraction	Mac `CMD` + `ALT` + `M` - Win / Linux `CTRL` + `ALT` + `M`
Extraire le paramètre	Mac `CMD` + `ALT` + `P` - Win / Linux `CTRL` + `ALT` + `P`
Extraire la variable	Mac `CMD` + `ALT` + `V` - Win / Linux `CTRL` + `ALT` + `V`

Android Studio Améliorez les performances

Activer le travail hors ligne:

1. Cliquez sur Fichier -> Paramètres. Recherchez "gradle" et cliquez dans la zone de `Offline work` .
2. Accédez au compilateur (dans la même boîte de dialogue de paramètres, juste en dessous de `Gradle`) et ajoutez `--offline` à la zone de texte `Command-line Options` de `Command-line Options` .

Améliorer la performance de Gradle

Ajoutez les deux lignes de code suivantes dans votre fichier gradle.properties.

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

Augmenter la valeur de `-Xmx` et `-Xms` dans le fichier `studio.vmoptions`

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

Fenêtre

> % USERPROFILE%. {NOM FOLDER} \ studio.exe.vmoptions et / ou% USERPROFILE%. {NOM FOLDER} \ studio64.exe.vmoptions

Mac

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

Linux

> ~ /. {NOM DU DOSSIER} /studio.vmoptions et / ou ~ /. {NOM FOLDER}

/studio64.vmoptions

Configurer Android Studio

Configuration requise

- Microsoft® Windows® 8/7 / Vista / 2003 (32 ou 64 bits).
- Mac® OS X® 10.8.5 ou supérieur, jusqu'à 10.9 (Mavericks)
- Bureau GNOME ou KDE

Installation

Fenêtre

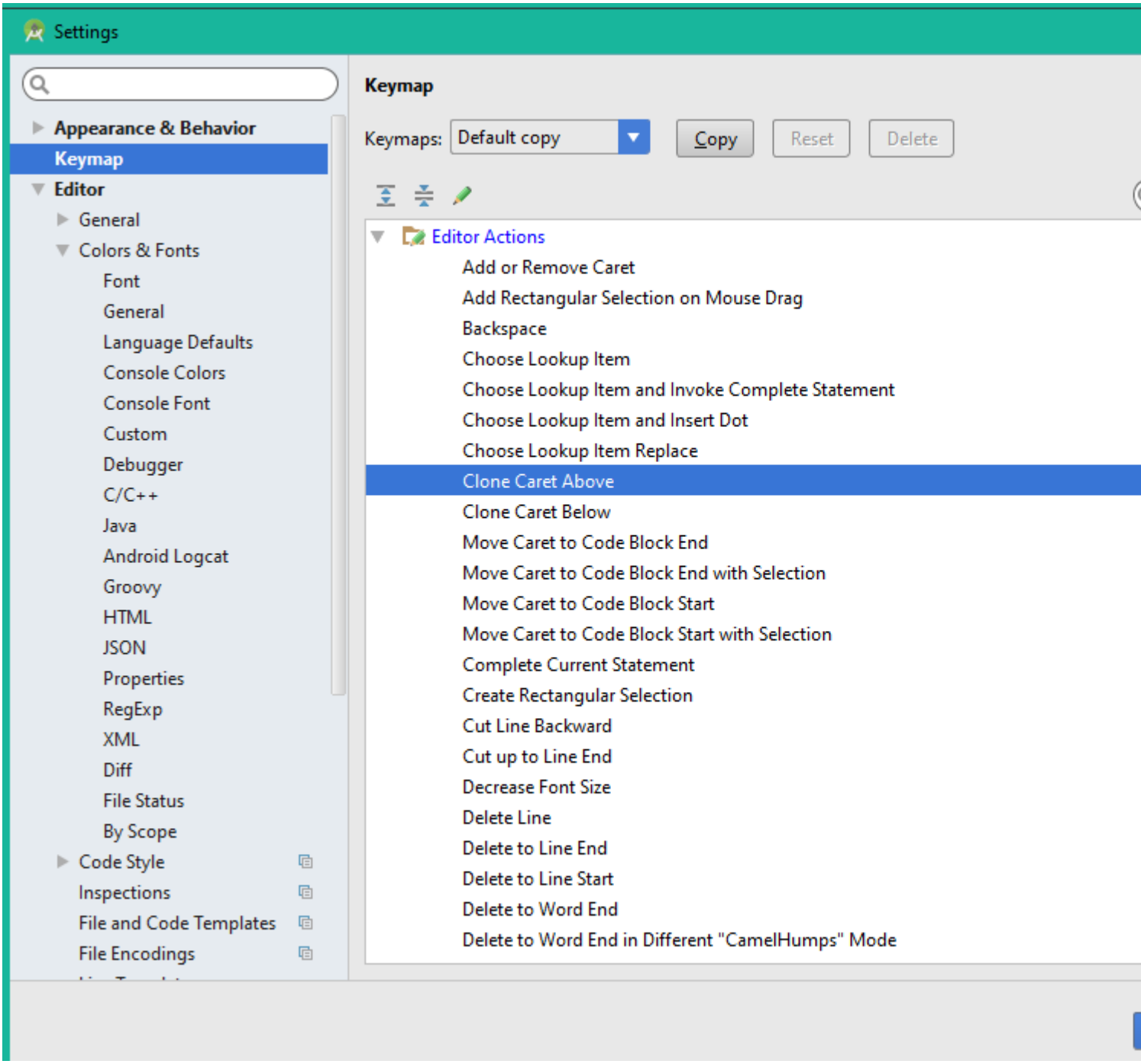
1. Téléchargez et installez JDK (Java Development Kit) version 8
2. Télécharger Android Studio
3. Lancez `Android Studio.exe` puis mentionnez le chemin JDK et téléchargez le dernier SDK

Linux

1. Téléchargez et installez JDK (Java Development Kit) version 8
2. Télécharger Android Studio
3. Extraire le fichier zip
4. Ouvrez le terminal, cd vers le dossier extrait, cd vers bin (exemple `cd android-studio/bin`)
5. Exécuter ./studio.sh

Afficher et ajouter des raccourcis dans Android Studio

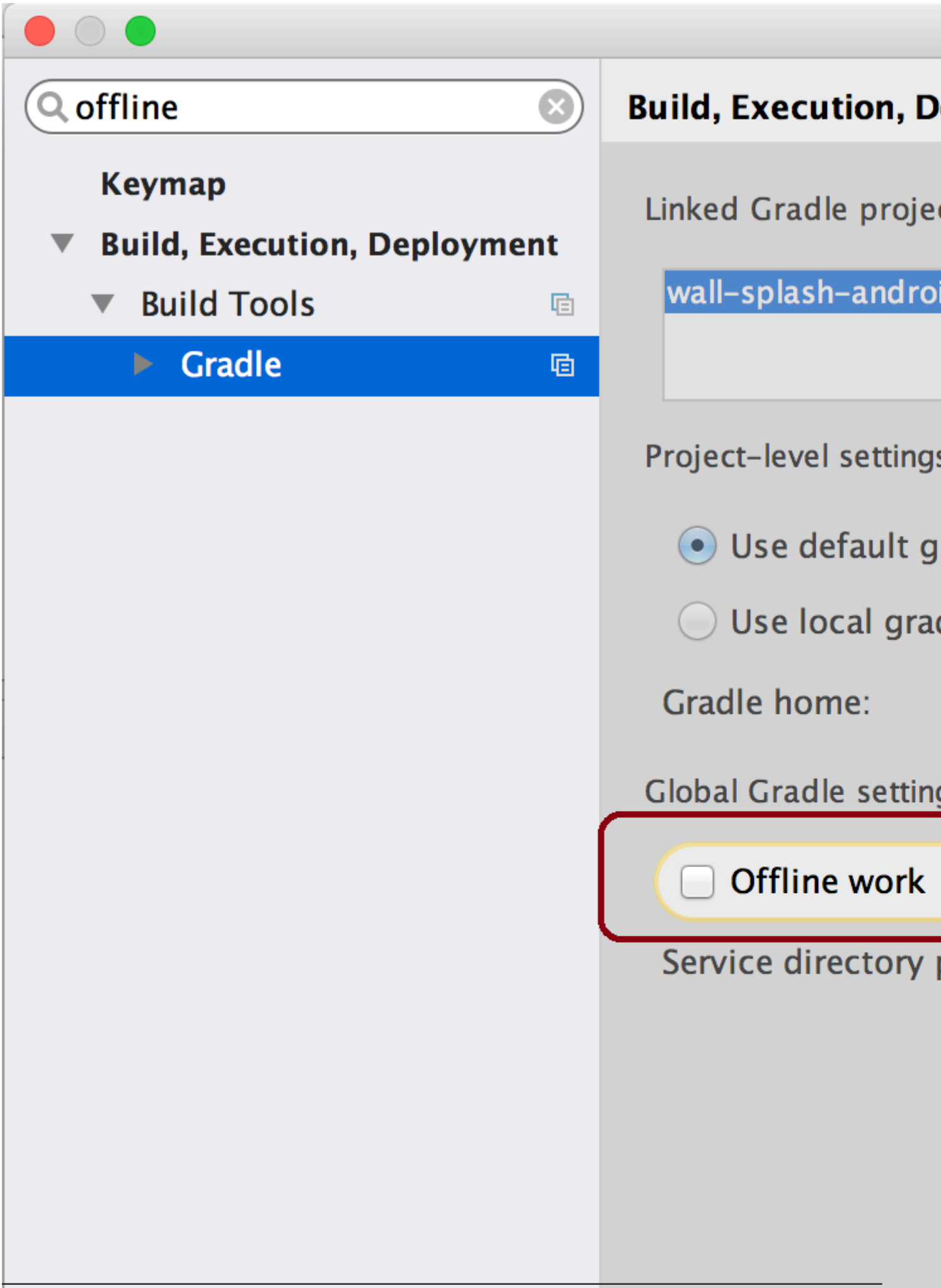
En allant dans Paramètres >> Keymap Une fenêtre apparaîtra affichant Toutes les `Editor Actions` l' `Editor Actions` avec leur nom et leurs raccourcis. Certaines `Editor Actions` de l' `Editor Actions` n'ont pas de raccourcis. Alors faites un clic droit dessus et ajoutez un nouveau raccourci à cela. Vérifiez l'image ci-dessous



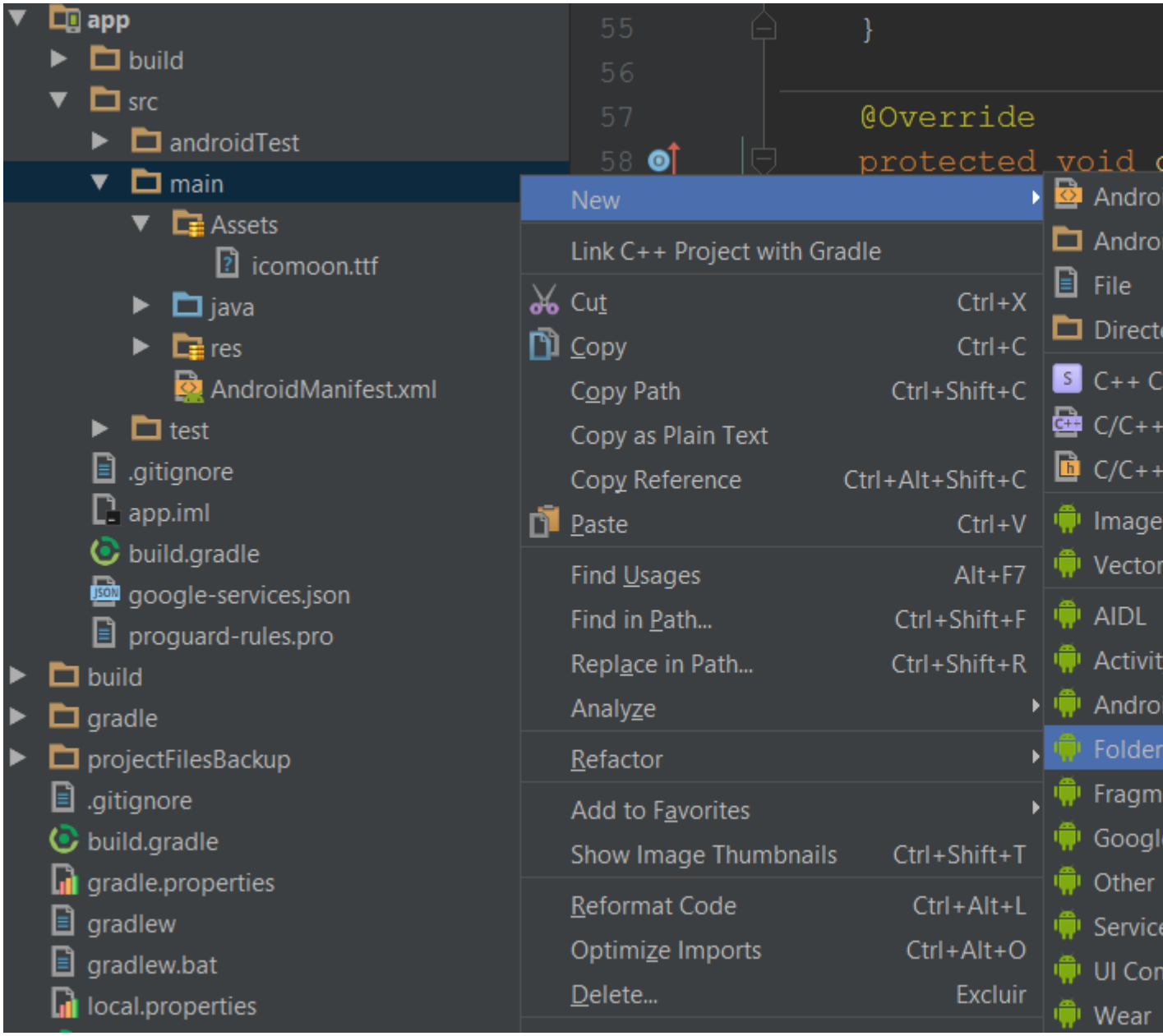
Le projet de construction de Gradle prend une éternité

Android Studio -> **Préférences** -> **Gradle** -> Cochez **Offline work** , puis redémarrez votre studio Android.

Capture d'écran de référence:


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

- Le dossier Assets sera sous le dossier MAIN avec le même symbole que le dossier RES.
- Dans cet exemple, je mets un fichier de police.



Lire Studio Android en ligne: https://riptutorial.com/fr/android/topic/107/studio-android

Chapitre 232: SyncAdapter avec régulièrement synchroniser les données

Introduction

Le composant adaptateur de synchronisation de votre application encapsule le code des tâches qui transfèrent des données entre le périphérique et un serveur. En fonction de la planification et des déclencheurs que vous fournissez dans votre application, l'infrastructure d'adaptateur de synchronisation exécute le code dans le composant de l'adaptateur de synchronisation.

Récemment, j'ai travaillé sur SyncAdapter, je souhaite partager mes connaissances avec d'autres, cela pourrait aider les autres.

Examples

Adaptateur de synchronisation avec chaque minute demandant la valeur du serveur.

```
<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>
```

Ce code doit être ajouté dans le fichier manifeste

Dans le code ci-dessus, nous avons le syncservice et conteprovider et authenticatorservice.

Dans l'application, nous devons créer le package xml pour ajouter des fichiers xml syncadpter et authentique. **authentique.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"
    />
```

adaptateur syncad

```
<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"/>
```

Authentificateur

```
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;
    }
}
```

Activité principale

```
     public class MainActivity extends AppCompatActivity {

        private static final String ACCOUNT_TYPE="sample.map.com.ipsyncadapter";
        private static final String AUTHORITY="sample.map.com.ipsyncadapter";
        private static final String ACCOUNT_NAME="Sync";

        public TextView mIp,mCountryCod,mCountryName,mCity,mLatitude,mLongitude;
        CursorAdapter cursorAdapter;
        Account mAccount;
        private String TAG=this.getClass().getCanonicalName();
        ListView mListView;
        public SharedPreferences mSharedPreferences;
        @Override
        protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_main);
            mListView = (ListView) findViewById(R.id.list);
            mIp=(TextView)findViewById(R.id.txt_ip);
            mCountryCod=(TextView)findViewById(R.id.txt_country_code);
            mCountryName=(TextView)findViewById(R.id.txt_country_name);
            mCity=(TextView)findViewById(R.id.txt_city);
            mLatitude=(TextView)findViewById(R.id.txt_latitude);
            mLongitude=(TextView)findViewById(R.id.txt_longitude);
            mSharedPreferences=getSharedPreferences("MyIp",0);

//Using shared preference iam displaying values in text view.
            String txtIp=mSharedPreferences.getString("ipAdr","");
            String txtCC=mSharedPreferences.getString("CCode","");
            String txtCN=mSharedPreferences.getString("CName","");
            String txtC=mSharedPreferences.getString("City","");
            String txtLP=mSharedPreferences.getString("Latitude","");
            String txtLN=mSharedPreferences.getString("Longitude","");

            mIp.setText(txtIp);
            mCountryCod.setText(txtCC);
            mCountryName.setText(txtCN);
            mCity.setText(txtC);
            mLatitude.setText(txtLP);
            mLongitude.setText(txtLN);

            mAccount=createSyncAccount(this);
//In this code i am using content provider to save data.
           /* Cursor
cursor=getContentResolver().query(MyIPContentProvider.CONTENT_URI,null,null,null,null);
            cursorAdapter=new SimpleCursorAdapter(this,R.layout.list_item,cursor,new String
[]{"ip","country_code","country_name","city","latitude","longitude"},
                                                                new int[]
{R.id.txt_ip,R.id.txt_country_code,R.id.txt_country_name,R.id.txt_city,R.id.txt_latitude,R.id.txt_longi


            mListView.setAdapter(cursorAdapter);

getContentResolver().registerContentObserver(MyIPContentProvider.CONTENT_URI,true,new
StockContentObserver(new Handler()));
    */
            Bundle settingBundle=new Bundle();
            settingBundle.putBoolean(ContentResolver.SYNC_EXTRAS_MANUAL,true);
            settingBundle.putBoolean(ContentResolver.SYNC_EXTRAS_EXPEDITED,true);
            ContentResolver.requestSync(mAccount,AUTHORITY,settingBundle);
```

```
            ContentResolver.setSyncAutomatically(mAccount,AUTHORITY,true);
            ContentResolver.addPeriodicSync(mAccount,AUTHORITY,Bundle.EMPTY,60);
        }

        private Account createSyncAccount(MainActivity mainActivity) {
            Account account=new Account(ACCOUNT_NAME,ACCOUNT_TYPE);
            AccountManager
accountManager=(AccountManager)mainActivity.getSystemService(ACCOUNT_SERVICE);
            if(accountManager.addAccountExplicitly(account,null,null))
            {

            }else
            {

            }
            return account;
        }


        private class StockContentObserver extends ContentObserver {
            @Override
            public void onChange(boolean selfChange, Uri uri) {
                Log.d(TAG, "CHANGE OBSERVED AT URI: " + uri);

cursorAdapter.swapCursor(getContentResolver().query(MyIPContentProvider.CONTENT_URI, null,
null, null, null));
            }

            public StockContentObserver(Handler handler) {
                super(handler);

            }
        }
        @Override
        protected void onResume() {
            super.onResume();
            registerReceiver(syncStaredReceiver, new IntentFilter(SyncAdapter.SYNC_STARTED));
            registerReceiver(syncFinishedReceiver, new
IntentFilter(SyncAdapter.SYNC_FINISHED));
        }

        @Override
        protected void onPause() {
            super.onPause();
            unregisterReceiver(syncStaredReceiver);
            unregisterReceiver(syncFinishedReceiver);
        }
        private BroadcastReceiver syncFinishedReceiver = new BroadcastReceiver() {

            @Override
            public void onReceive(Context context, Intent intent) {
                Log.d(TAG, "Sync finished!");
                Toast.makeText(getApplicationContext(), "Sync Finished",
Toast.LENGTH_SHORT).show();
            }
        };
        private BroadcastReceiver syncStaredReceiver = new BroadcastReceiver() {

            @Override
            public void onReceive(Context context, Intent intent) {
                Log.d(TAG, "Sync started!");
```

```
                Toast.makeText(getApplicationContext(), "Sync started...",
Toast.LENGTH_SHORT).show();
            }
        };
    }
```

MyIPContentProvider

```
public class MyIPContentProvider extends ContentProvider {

public static final int IP_DATA=1;
private static final String AUTHORITY="sample.map.com.ipsyncadapter";
private static final String TABLE_IP_DATA="ip_data";
public static final Uri CONTENT_URI=Uri.parse("content://" + AUTHORITY + '/' + TABLE_IP_DATA);
private static final UriMatcher URI_MATCHER= new UriMatcher(UriMatcher.NO_MATCH);

static
{
    URI_MATCHER.addURI(AUTHORITY,TABLE_IP_DATA,IP_DATA);
}

private IpDataDBHelper myDB;

@Override
public boolean onCreate() {
    myDB=new IpDataDBHelper(getContext(),null,null,1);
    return false;
}

@Nullable
@Override
public Cursor query(Uri uri, String[] strings, String s, String[] strings1, String s1) {
    int uriType=URI_MATCHER.match(uri);
    Cursor cursor=null;
    switch (uriType)
    {
        case IP_DATA:
            cursor=myDB.getAllIpData();
            break;
        default:
            throw new IllegalArgumentException("UNKNOWN URL");
    }
    cursor.setNotificationUri(getContext().getContentResolver(), uri);
    return cursor;
}

@Nullable
@Override
public String getType(Uri uri) {
    return null;
}

@Nullable
@Override
public Uri insert(Uri uri, ContentValues contentValues) {
    int uriType=URI_MATCHER.match(uri);
    long id=0;
    switch (uriType)
    {
        case IP_DATA:
```

```
            id=myDB.AddIPData(contentValues);
            break;
        default:
            throw new IllegalArgumentException("UNKNOWN URI :" +uri);
    }
    getContext().getContentResolver().notifyChange(uri,null);
    return Uri.parse(contentValues + "/" + id);
}

@Override
public int delete(Uri uri, String s, String[] strings) {
    int uriType=URI_MATCHER.match(uri);
    int rowsDeleted=0;

    switch (uriType)
    {
        case IP_DATA:
            rowsDeleted=myDB.deleteAllIpData();
            break;
        default:
            throw new IllegalArgumentException("UNKNOWN URI :" +uri);
    }
    getContext().getContentResolver().notifyChange(uri,null);
    return rowsDeleted;
}

@Override
public int update(Uri uri, ContentValues contentValues, String s, String[] strings) {
    return 0;
}
```

}

SyncAdapter

```
public class SyncAdapter extends AbstractThreadedSyncAdapter {
ContentResolver mContentResolver;
Context mContext;
public static final String SYNC_STARTED="Sync Started";
public static final String SYNC_FINISHED="Sync Finished";
private static final String TAG=SyncAdapter.class.getCanonicalName();
public SharedPreferences mSharedPreferences;

public SyncAdapter(Context context, boolean autoInitialize) {
    super(context, autoInitialize);
    this.mContext=context;
    mContentResolver=context.getContentResolver();
    Log.i("SyncAdapter","SyncAdapter");
}

@Override
public void onPerformSync(Account account, Bundle bundle, String s, ContentProviderClient
contentProviderClient, SyncResult syncResult) {

    Intent intent = new Intent(SYNC_STARTED);
    mContext.sendBroadcast(intent);

    Log.i(TAG, "onPerformSync");

    intent = new Intent(SYNC_FINISHED);
```

```
    mContext.sendBroadcast(intent);
    mSharedPreferences =mContext.getSharedPreferences("MyIp",0);
    SharedPreferences.Editor editor=mSharedPreferences.edit();

    mContentResolver.delete(MyIPContentProvider.CONTENT_URI,null,null);

    String data="";

    try {
        URL url =new URL("https://freegeoip.net/json/");
        Log.d(TAG, "URL :"+url);
        HttpURLConnection connection=(HttpURLConnection)url.openConnection();
        Log.d(TAG,"Connection :"+connection);
        connection.connect();
        Log.d(TAG,"Connection 1:"+connection);
        InputStream inputStream=connection.getInputStream();
        data=getInputData(inputStream);
        Log.d(TAG,"Data :"+data);

        if (data != null || !data.equals("null")) {
            JSONObject jsonObject = new JSONObject(data);


            String ipa = jsonObject.getString("ip");
            String country_code = jsonObject.getString("country_code");
            String country_name = jsonObject.getString("country_name");
            String region_code=jsonObject.getString("region_code");
            String region_name=jsonObject.getString("region_name");
            String zip_code=jsonObject.getString("zip_code");
            String time_zone=jsonObject.getString("time_zone");
            String metro_code=jsonObject.getString("metro_code");

            String city = jsonObject.getString("city");
            String latitude = jsonObject.getString("latitude");
            String longitude = jsonObject.getString("longitude");
           /* ContentValues values = new ContentValues();
            values.put("ip", ipa);
            values.put("country_code", country_code);
            values.put("country_name", country_name);
            values.put("city", city);
            values.put("latitude", latitude);
            values.put("longitude", longitude);*/
            //Using cursor adapter for results.
            //mContentResolver.insert(MyIPContentProvider.CONTENT_URI, values);

            //Using Shared preference for results.
            editor.putString("ipAdr",ipa);
            editor.putString("CCode",country_code);
            editor.putString("CName",country_name);
            editor.putString("City",city);
            editor.putString("Latitude",latitude);
            editor.putString("Longitude",longitude);
            editor.commit();

        }
        }catch(Exception e){
            e.printStackTrace();
        }

}
```

```
private String getInputData(InputStream inputStream) throws IOException {
    StringBuilder builder=new StringBuilder();
    BufferedReader bufferedReader=new BufferedReader(new InputStreamReader(inputStream));
    //String data=null;
    /*Log.d(TAG,"Builder 2:"+ bufferedReader.readLine());
    while ((data=bufferedReader.readLine())!= null);
    {
        builder.append(data);
        Log.d(TAG,"Builder :"+data);
    }
    Log.d(TAG,"Builder 1 :"+data);
    bufferedReader.close();*/
    String data=bufferedReader.readLine();
    bufferedReader.close();
    return data.toString();
}
```

}

SyncService

```
public class SyncService extends Service {
private static SyncAdapter syncAdapter=null;
private static final Object syncAdapterLock=new Object();

@Override
public void onCreate() {
    synchronized (syncAdapterLock)
    {
        if(syncAdapter==null)
        {
            syncAdapter =new SyncAdapter(getApplicationContext(),true);
        }
    }
}

@Nullable
@Override
public IBinder onBind(Intent intent) {
    return syncAdapter.getSyncAdapterBinder();
}
```

}

Lire SyncAdapter avec régulièrement synchroniser les données en ligne: https://riptutorial.com/fr/android/topic/10774/syncadapter-avec-regulierement-synchroniser-les-donnees

Chapitre 233: Synchronisation des données avec l'adaptateur de synchronisation

Examples

Adaptateur de synchronisation factice avec fournisseur de stub

SyncAdapter

```
/**
 * Define a sync adapter for the app.
 * <p/>
 * <p>This class is instantiated in {@link SyncService}, which also binds SyncAdapter to the
system.
 * SyncAdapter should only be initialized in SyncService, never anywhere else.
 * <p/>
 * <p>The system calls onPerformSync() via an RPC call through the IBinder object supplied by
 * SyncService.
 */
class SyncAdapter extends AbstractThreadedSyncAdapter {
    /**
     * Constructor. Obtains handle to content resolver for later use.
     */
    public SyncAdapter(Context context, boolean autoInitialize) {
        super(context, autoInitialize);
    }

    /**
     * Constructor. Obtains handle to content resolver for later use.
     */
    public SyncAdapter(Context context, boolean autoInitialize, boolean allowParallelSyncs) {
        super(context, autoInitialize, allowParallelSyncs);
    }

    @Override
    public void onPerformSync(Account account, Bundle extras, String authority,
                              ContentProviderClient provider, SyncResult syncResult) {
        //Jobs you want to perform in background.
        Log.e("" + account.name, "Sync Start");
    }

}
```

Service de synchronisation

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

Authentificateur

```
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();
    }
}
```

Service d'authentification

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

Ajouts à 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;
    }
}
```

Appelez cette fonction en cas de connexion réussie pour créer un compte avec l'ID utilisateur connecté

```
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;
    }
```

Forcer une synchronisation

```
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);
```

Lire Synchronisation des données avec l'adaptateur de synchronisation en ligne: https://riptutorial.com/fr/android/topic/1944/synchronisation-des-donnees-avec-l-adaptateur-de-synchronisation

Chapitre 234: TabLayout

Examples

Utiliser un TabLayout sans ViewPager

La plupart du temps, un `TabLayout` est utilisé avec un ViewPager , afin d'obtenir la fonctionnalité de balayage qui l'accompagne.

Il est possible d'utiliser un `TabLayout` sans `ViewPager` en utilisant un `TabLayout.OnTabSelectedListener` .

Tout d'abord, ajoutez un `TabLayout` au fichier XML de votre activité:

```
<android.support.design.widget.TabLayout
    android:layout_height="wrap_content"
    android:layout_width="match_parent"
    android:id="@+id/tabLayout" />
```

Pour la navigation dans une `Activity` , remplissez manuellement l'interface utilisateur en fonction de l'onglet sélectionné.

```
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) {

    }
});
```

Lire TabLayout en ligne: https://riptutorial.com/fr/android/topic/7601/tablayout

Chapitre 235: Temps Utils

Examples

Convertir le format de date en millisecondes

Pour convertir votre date au format jj / MM / aaaa en millisecondes, vous appelez cette fonction avec des données sous forme de chaîne

```
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();
}
```

Cette méthode convertit les millisecondes en date de format d'horodatage:

```
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;
}
```

Cette méthode convertira un jour, un mois et une année donnés en millisecondes. Ce sera très utile lorsque vous utilisez `Timpicker` ou `Datepicker`

```
public static long getTimeInMillis(int day, int month, int year) {
    Calendar calendar = Calendar.getInstance();
    calendar.set(year, month, day);
    return calendar.getTimeInMillis();
}
```

Il retournera des millisecondes à partir de la date

```
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;
}
```

Il retournera la date actuelle

```
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;
}
```

Remarque: Java Fournit le nombre de formats de date pris en charge Modèle de date

Pour vérifier dans une période

Cet exemple aidera à vérifier que l'heure donnée est comprise dans une période ou non.

*Pour vérifier l'heure, nous pouvons utiliser la classe **DateUtils***

```
boolean isToday = DateUtils.isToday(timeInMillis);
```

Pour vérifier l'heure est dans une semaine,

```
private static boolean isWithinWeek(final long millis) {
    return System.currentTimeMillis() - millis <= (DateUtils.WEEK_IN_MILLIS -
DateUtils.DAY_IN_MILLIS);
}
```

Pour vérifier l'heure est dans un an,

```
private static boolean isWithinYear(final long millis) {
    return System.currentTimeMillis() - millis <= DateUtils.YEAR_IN_MILLIS;
}
```

Pour vérifier l'heure est dans un nombre de jour de jour, y compris aujourd'hui,

```
public static boolean isWithinDay(long timeInMillis, int day) {
    long diff = System.currentTimeMillis() - timeInMillis;

    float dayCount = (float) (diff / DateUtils.DAY_IN_MILLIS);

    return dayCount < day;
}
```

Remarque: DateUtils est **android.text.format.DateUtils**

GetCurrentRealTime

Cela calcule l'heure actuelle de l'appareil et ajoute / soustrait la différence entre l'heure réelle et l'heure de l'appareil

```
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;
}
```

obtenir le fuseau horaire basé sur UTC.

```
public static TimeZone getUTCTimeZone() {
    return TimeZone.getTimeZone("GMT");
}
```

Lire Temps Utils en ligne: https://riptutorial.com/fr/android/topic/7138/temps-utils

Chapitre 236: TensorFlow

Introduction

TensorFlow a été conçu pour les plates-formes mobiles et intégrées. Nous avons un exemple de code et de support de construction que vous pouvez essayer maintenant pour ces plates-formes:

Android iOS Raspberry Pi

Remarques

Un travail apprécié de MindRocks

Examples

Comment utiliser

Installez Bazel d' ici . Bazel est le principal système de compilation de TensorFlow. Maintenant, éditez le WORKSPACE, nous pouvons trouver le fichier WORKSPACE dans le répertoire racine du TensorFlow que nous avons cloné plus tôt.

```
# 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)
```

Comme ci-dessous avec notre chemin sdk et ndk:

```
android_sdk_repository(
    name = "androidsdk",
    api_level = 23,
    build_tools_version = "25.0.1",
    # Replace with path to Android SDK on your system
    path = "/Users/amitshekhar/Library/Android/sdk/",
)
android_ndk_repository(
    name="androidndk",
    path="/Users/amitshekhar/Downloads/android-ndk-r13/",
    api_level=14)
```

Lire TensorFlow en ligne: https://riptutorial.com/fr/android/topic/9991/tensorflow

Chapitre 237: Test d'interface utilisateur inter-app avec UIAutomator

Syntaxe

- Instrumentation getInstrumentation ()
- UIDevice UiDevice.getInstance (Instrumentation d'instrumentation)
- booléen UIDevice.pressHome ()
- booléen UIDevice.pressBack ()
- booléen UIDevice.pressRecentApps ()
- annuler UIDevice.wakeUp ()
- booléen UIDevice.swipe (int startX, int startY, int endX, int endY, int étapes)
- booléen UIDevice.drag (int startX, int startY, int endX, int endY, int étapes)
- UIObject2 UIDevice.findObject (By.desc (String contentDesc))
- booléen UIObject2.click ()

Remarques

UIAutomator est particulièrement utile pour tester les user stories. Vous rencontrez des problèmes si les éléments de vue ne possèdent ni un *identifiant de ressource* unique ni un *contenu-desc* . Dans la plupart des cas, il existe un moyen de terminer le test de toute façon, ce qui prend beaucoup de temps. Si vous pouvez influencer le code de votre application, UIAutomator peut être votre outil de test.

Examples

Préparez votre projet et écrivez le premier test UIAutomator

Ajoutez les bibliothèques requises dans la section dépendances du build.gradle de votre module Android:

```
android {
...
  defaultConfig {
    ...
    testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
  }
}

dependencies {
  ...
  androidTestCompile 'com.android.support.test:runner:0.5'
  androidTestCompile 'com.android.support.test:rules:0.5'
  androidTestCompile 'com.android.support.test.uiautomator:uiautomator-v18:2.1.2'
  androidTestCompile 'com.android.support:support-annotations:23.4.0'
}
```

Notez bien entendu que les versions peuvent différer dans le même temps.

Après cette synchronisation avec les modifications.

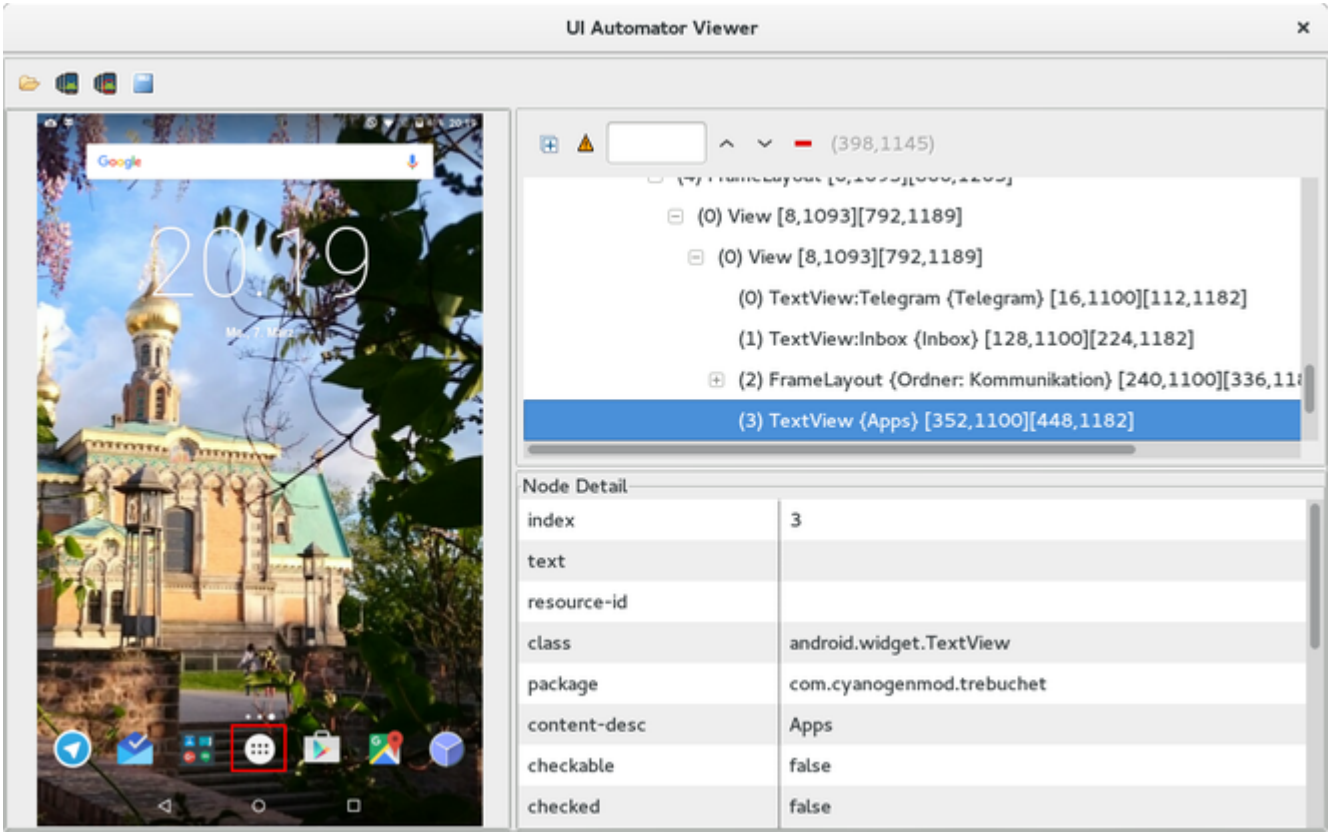
Ajoutez ensuite une nouvelle classe Java dans le dossier androidTest:

```
public class InterAppTest extends InstrumentationTestCase {

  private UiDevice device;

  @Override
  public void setUp() throws Exception {
     device = UiDevice.getInstance(getInstrumentation());
  }

  public void testPressHome() throws Exception {
    device.pressHome();
  }
}
```

En faisant un clic droit sur l'onglet de la classe et sur "Exécuter" InterAppTest "exécute ce test.

Rédiger des tests plus complexes à l'aide de UIAutomatorViewer

Pour permettre l'écriture de tests d'interface utilisateur plus complexes, *UIAutomatorViewer* est nécessaire. L'outil situé dans */ tools /* fait une capture d'écran en plein écran incluant les dispositions des vues actuellement affichées. Voir la photo suivante pour avoir une idée de ce qui est montré:



Pour les tests de l'interface utilisateur, nous recherchons un *identifiant* de *ressource* , un *contenu-*

desc ou quelque chose d'autre pour identifier une vue et l'utiliser dans nos tests.

Le *uiautomatorviewer* est exécuté via le terminal.

Si, par exemple, nous voulons maintenant cliquer sur le bouton des applications, puis ouvrir une application et la balayer, voici à quoi la méthode de test peut ressembler:

```
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)
}
```

Création d'une suite de tests de tests UIAutomator

La combinaison des tests UIAutomator à une suite de tests est rapide:

```
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 {}
```

Exécuter un test similaire à un seul test en cliquant à droite et exécutez la suite.

Lire Test d'interface utilisateur inter-app avec UIAutomator en ligne:
https://riptutorial.com/fr/android/topic/6249/test-d-interface-utilisateur-inter-app-avec-uiautomator

Chapitre 238: Test de l'interface utilisateur avec Espresso

Remarques

Expresso

La feuille de triche expresso vous aidera à écrire vos tests et ce que vous voulez tester:

https://google.github.io/android-testing-support-library/docs/espresso/cheatsheet/

La documentation officielle est toujours un bon endroit pour référence:

https://google.github.io/android-testing-support-library/docs/espresso/index.html

Suggestions avancées de vidéos expresso par Google:
https://www.youtube.com/watch?v=isihPOY2vS4

Dépannage

- Lorsque vous essayez de faire défiler, veillez à fermer le clavier en premier:

Watchout: ne pas utiliser la version "Espresso" ne fera rien en dehors de ViewAction. Cela peut ne pas être évident si vous avez une importation sur la version de ViewAction, car ils ont exactement le même nom de méthode.

```
ViewActions.closeSoftKeyboard;
Espresso.closeSoftKeyboard();
```

- Lorsque vous exécutez des tests ensemble dans une suite plutôt qu'individuellement, sachez que l'activité du test précédent est peut-être toujours en cours d'exécution. Ne vous fiez pas à l'appel du test précédent onDestroy () avant les tests en cours onResume (). **Il s'avère que c'est en fait un bogue** : http://b.android.com/201513

Examples

Configurer Espresso

Dans le fichier `build.gradle` de votre module d'application Android, ajoutez les prochaines dépendances:

```
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'
}
```

Spécifiez `AndroidJUnitRunner` pour le paramètre `testInstrumentationRunner` dans le fichier `build.gradle` .

```
android {

  defaultConfig {
    testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
  }

}
```

En outre, ajoutez cette dépendance pour fournir une prise en charge intentionnelle

```
androidTestCompile 'com.android.support.test.espresso:espresso-intents:2.2.2'
```

Et ajoutez celui-ci pour la prise en charge des tests WebView

```
// Espresso-web for WebView support
androidTestCompile 'com.android.support.test.espresso:espresso-web:2.2.2'
```

Créer une classe de test expresso

Placez la classe java suivante dans **src / androidTest / java** et lancez-la.

```
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
  }

}
```

Ouvrir Fermer 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);
      }
    };
  }

}
```

Test de l'interface utilisateur simple expresso

Outils de test de l'interface utilisateur

Appium et Espresso sont les deux principaux outils actuellement utilisés pour les tests d’interface utilisateur.

Appium	Expresso
test de blackbox	test de boîte blanche / grise
ce que vous voyez est ce que vous pouvez tester	peut modifier le fonctionnement interne de l'application et le préparer pour le test, par exemple enregistrer certaines données dans la base de données ou les préférences partagées avant de lancer le test
Utilisé principalement pour des tests d'intégration de bout en bout et des flux d'utilisateurs	tester la fonctionnalité d'un écran et / ou d'un flux

Appium	Expresso
entiers	
peut être abstraite afin que le test écrit puisse être exécuté sur iOS et Android	Android uniquement
Bien soutenu	Bien soutenu
prend en charge les tests parallèles sur plusieurs périphériques avec grille de sélénium	Pas de tests parallèles prêts à l'emploi, des plug-ins comme Spoon existent jusqu'à ce que le véritable support de Google apparaisse

Comment ajouter du café au projet

```
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'
}
```

REMARQUE Si vous utilisez les dernières bibliothèques de support, les annotations, etc., vous devez exclure les anciennes versions de l'espresso pour éviter les collisions:

```
    // 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'
    }
```

Outre ces importations, il est nécessaire d'ajouter le testeur d'instrumentation Android à build.gradle android.defaultConfig:

```
testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
```

Configuration de l'appareil

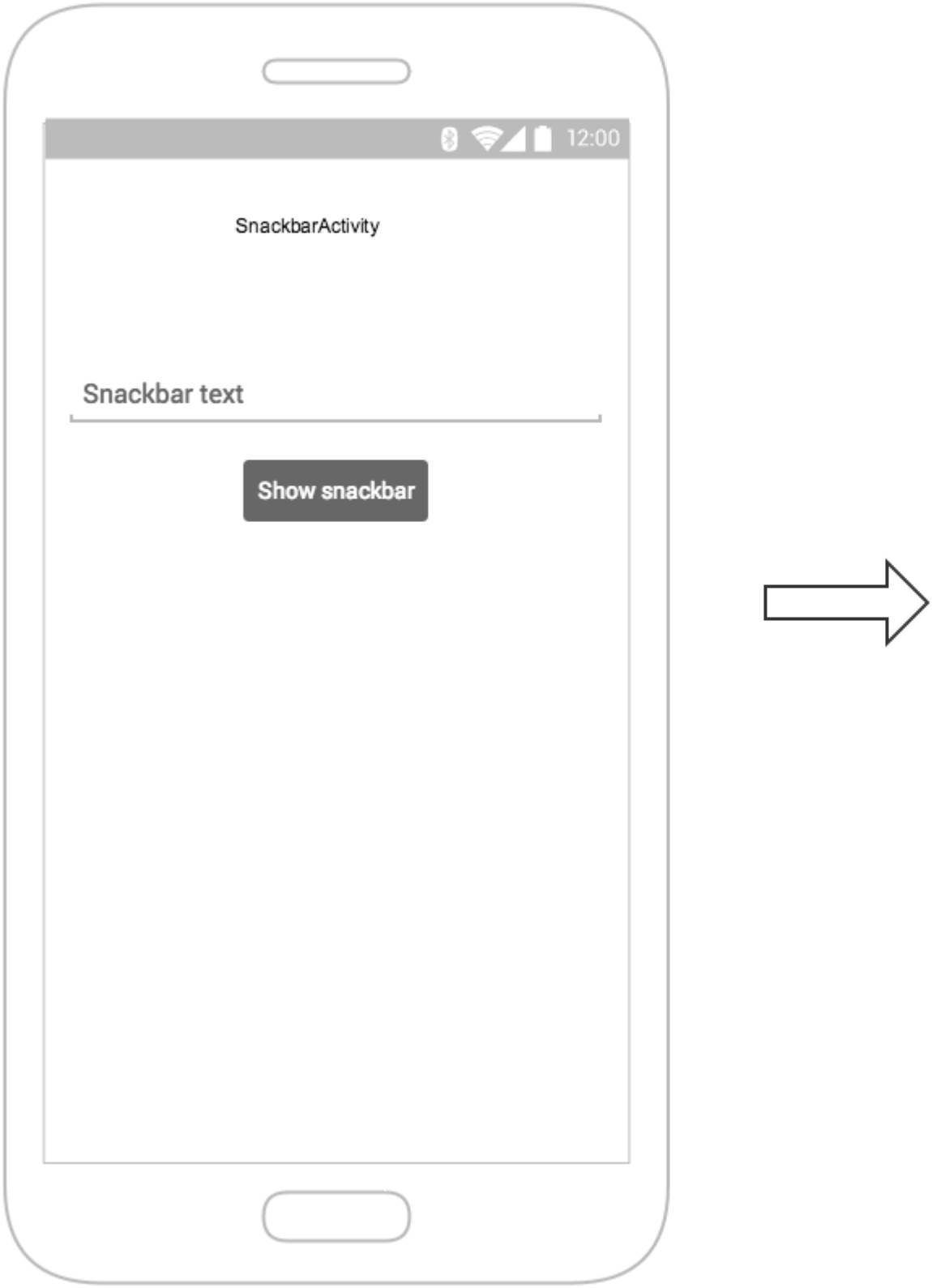
Pour un test non feuilleté, il est recommandé de définir les paramètres suivants sur vos appareils:

- Options pour les développeurs / Désactiver les animations - réduit la fragilité des tests
- Options pour les développeurs / Restez éveillé - si vous avez des appareils dédiés aux tests, cela est utile
- Options du développeur / Taille du tampon de l'enregistreur - définissez un nombre plus élevé si vous exécutez des suites de test très volumineuses sur votre téléphone
- Délai d'accessibilité / Touch & Hold - long pour éviter les problèmes de tapotement dans l'espresso

Assez une configuration du monde réel ha? Eh bien, maintenant, quand c'est hors de portée permet de jeter un oeil à la configuration d'un petit test

Écrire le test

Supposons que nous avons l'écran suivant:



L'écran contient:

- champ de saisie de texte - **R.id.textEntry**
- bouton qui montre le snack avec le texte tapé quand cliqué - **R.id.shownSnackbarBtn**

- snackbar qui devrait contenir le texte saisi par l'utilisateur - **android.support.design.R.id.snackbar_text**

Maintenant, créons une classe qui testera notre flux:

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

Comme vous l'avez remarqué, il y a 3 ou 4 choses que vous remarquerez souvent:

onView (withXYZ) <- viewMatchers avec eux, vous pouvez trouver des éléments à l'écran

effectuer (click ()) <- viewActions, vous pouvez exécuter des actions sur des éléments trouvés

précédemment

check (correspond à (isDisplayed ())) <- viewAssertions, vérifications à effectuer sur les écrans précédemment trouvés

Tous ces éléments et bien d'autres peuvent être trouvés ici: https://google.github.io/android-testing-support-library/docs/espresso/cheatsheet/index.html

Thats it, maintenant vous pouvez exécuter le test soit avec un clic droit sur le nom de la classe / test et en sélectionnant Exécuter le test ou avec la commande:

```
./gradlew connectedFLAVORNAMEAndroidTest
```

La navigation

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

Notez qu'il s'agit d'une solution de contournement et qu'elle entrera en conflit avec d'autres vues ayant la même description de contenu.

Effectuer une action sur une vue

Il est possible d'effectuer des `ViewActions` sur une vue en utilisant la méthode perform.
La classe `ViewActions` fournit des méthodes d'assistance pour les actions les plus courantes, telles que:

```
ViewActions.click()
ViewActions.typeText()
ViewActions.clearText()
```

Par exemple, pour cliquer sur la vue:

```
onView(...).perform(click());
onView(withId(R.id.button_simple)).perform(click());
```

Vous pouvez exécuter plusieurs actions avec un seul appel:

```
onView(...).perform(typeText("Hello"), click());
```

Si la vue avec laquelle vous travaillez est située dans un `ScrollView` (vertical ou horizontal), considérez les actions précédentes qui nécessitent l'affichage de la vue (comme `click()` et

`typeText()`) avec `scrollTo()` . Cela garantit que la vue est affichée avant de passer à l'autre action:

```
onView(...).perform(scrollTo(), click());
```

Trouver une vue avec onView

Avec `ViewMatchers` vous pouvez voir la vue dans la hiérarchie de la vue actuelle.

Pour rechercher une vue, utilisez la méthode `onView()` avec un observateur de vue qui sélectionne la vue correcte. Les méthodes `onView()` renvoient un objet de type `ViewInteraction` .

Par exemple, trouver une vue par son `R.id` est aussi simple que:

```
onView(withId(R.id.my_view))
```

Trouver une vue avec un texte:

```
onView(withText("Hello World"))
```

Espresso personnalisés

Espresso par défaut a beaucoup de correspondances qui vous aident à trouver des vues dont vous avez besoin pour faire des vérifications ou des interactions avec eux.

Les plus importants peuvent être trouvés dans la feuille de triche suivante:

https://google.github.io/android-testing-support-library/docs/espresso/cheatsheet/

Voici quelques exemples d’appariеurs:

- avecId (R.id.ID_of_object_you_are_looking_for);
- withText ("Du texte que vous attendez d'un objet");
- isDisplayed () <- check est la vue visible
- doesNotExist () <- vérifie que la vue n'existe pas

Tous ces éléments sont très utiles pour un usage quotidien, mais si vous avez des vues plus complexes, les personnaliseurs peuvent rendre les tests plus lisibles et les réutiliser à différents endroits.

Il y a 2 types les plus communs d'allumeurs que vous pouvez étendre: **TypeSafeMatcher BoundedMatcher**

L'implémentation de TypeSafeMatcher nécessite que vous vérifiiez l'instance de la vue à laquelle vous vous engagez, si le type correct correspond à certaines de ses propriétés par rapport à une valeur que vous avez fournie à un analyseur.

Par exemple, le gestionnaire de type sécurisé qui valide une vue d'image peut être dessiné correctement:

```
public class DrawableMatcher extends TypeSafeMatcher<View> {

    private @DrawableRes final int expectedId;
    String resourceName;

    public DrawableMatcher(@DrawableRes int expectedId) {
        super(View.class);
        this.expectedId = expectedId;
    }

    @Override
    protected boolean matchesSafely(View target) {
        //Type check we need to do in TypeSafeMatcher
        if (!(target instanceof ImageView)) {
            return false;
        }
        //We fetch the image view from the focused view
        ImageView imageView = (ImageView) target;
        if (expectedId < 0) {
            return imageView.getDrawable() == null;
        }
        //We get the drawable from the resources that we are going to compare with image view
source
        Resources resources = target.getContext().getResources();
        Drawable expectedDrawable = resources.getDrawable(expectedId);
        resourceName = resources.getResourceEntryName(expectedId);

        if (expectedDrawable == null) {
            return false;
        }
        //comparing the bitmaps should give results of the matcher if they are equal
        Bitmap bitmap = ((BitmapDrawable) imageView.getDrawable()).getBitmap();
        Bitmap otherBitmap = ((BitmapDrawable) expectedDrawable).getBitmap();
        return bitmap.sameAs(otherBitmap);
    }


    @Override
    public void describeTo(Description description) {
        description.appendText("with drawable from resource id: ");
        description.appendValue(expectedId);
        if (resourceName != null) {
            description.appendText("[");
            description.appendText(resourceName);
            description.appendText("]");
        }
    }
}
```

L'utilisation du matcher pourrait être comme ceci:

```
  public static Matcher<View> withDrawable(final int resourceId) {
    return new DrawableMatcher(resourceId);
}

 onView(withDrawable(R.drawable.someDrawable)).check(matches(isDisplayed()));
```

Les corrélateurs liés sont similaires mais vous n'avez pas à faire de vérification de type mais, puisque cela se fait automatiquement pour vous:

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

Plus sur les matchers peuvent être lus sur:

http://hamcrest.org/

https://developer.android.com/reference/android/support/test/espresso/matcher/ViewMatchers.html

Espresso globale

Configurer Espresso:

```
androidTestCompile 'com.android.support.test.espresso:espresso-core:2.2.2'
androidTestCompile 'com.android.support.test:runner:0.5'
```

ViewMatchers - Une collection d'objets qui implémentent `Matcher<? super View>` interface. Vous pouvez en transmettre une ou plusieurs à la méthode `onView` pour localiser une vue dans la hiérarchie de vues actuelle.

ViewActions - Collection de `ViewActions` pouvant être transmise à la méthode `ViewInteraction.perform()`

(par exemple, `click()`).

ViewAssertions - Une collection de `ViewAssertions` pouvant être passée à la méthode `ViewInteraction.check()` . La plupart du temps, vous utiliserez l'assertion de correspondances, qui utilise un analyseur de vues pour déterminer l'état de la vue actuellement sélectionnée.

Feuillet d'espresso par google

```
onView(ViewMatcher)
    .perform(ViewAction)
    .check(ViewAssertion);
```

View Matchers

USER PROPERTIES

```
withId(...)
withText(...)
withTagKey(...)
withTagValue(...)
hasContentDescription(...)
withContentDescription(...)
withHint(...)
withSpinnerText(...)
hasLinks()
hasEllipsizedText()
hasMultilineTest()
```

UI PROPERTIES

```
isDisplayed()
isCompletelyDisplayed()
isEnabled()
hasFocus()
isClickable()
isChecked()
isNotChecked()
withEffectiveVisibility(...)
isSelected()
```

OBJECT MATCHER

```
allOf(Matchers)
anyOf(Matchers)
is(...)
```

HIERARCHY

```
withParent(Matcher)
withChild(Matcher)
hasDescendant(Matcher)
isDescendantOfA(Matcher)
hasSibling(Matcher)
isRoot()
```

INPUT

```
supportsInputMethods(...)
hasIMEAction(...)
```

CLASS

```
isAssignableFrom(...)
withClassName(...)
```

ROOT MATCHERS

```
isFocusable()
isTouchable()
isDialog()
withDecorView()
isPlatformPopup()
```

SEE ALSO

Preference matchers

https://riptutorial.com/fr/android/topic/3485/test-de-l-interface-utilisateur-avec-espresso

Chapitre 239: Tests unitaires sous Android avec JUnit

Remarques

- Vogella: Tests unitaires avec JUnit
- Annotations Junit: java2novice.com
- Classe Assert : junit.org
- JUnit Api: tutorialspoint.com
- Anroid testing Medium.com posts

Examples

Créer des tests unitaires locaux

Placez vos classes de test ici: `/src/test/<pkg_name>/`

Exemple de classe de test

```
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
    }
}
```

Panne

```
public class ExampleUnitTest {
  ...
}
```

La classe de test, vous pouvez créer plusieurs classes de test et les placer dans le package de test.

```
@Test
public void addition_isCorrect() {
   ...
}
```

La méthode de test, plusieurs méthodes de test peuvent être créées dans une classe de test.

Notez l'annotation `@Test` .

> L'annotation Test indique à JUnit que la méthode vide publique à laquelle elle est attachée peut être exécutée en tant que scénario de test.

Il existe plusieurs autres annotations utiles comme `@Before` , `@After` etc. Cette page serait un bon point de départ.

```
assertEquals(9, c); // This test passes
assertEquals(10, c); //Test fails
```

Ces méthodes sont membres de la classe `Assert` . Quelques autres méthodes utiles sont `assertFalse()` , `assertNotNull()` , `assertTrue` etc. Voici une explication détaillée.

Informations d'annotation pour le test JUnit:

@Test: L'annotation Test indique à JUnit que la méthode vide publique à laquelle elle est attachée peut être exécutée en tant que **scénario de** test. Pour exécuter la méthode, JUnit construit d'abord une nouvelle instance de la classe, puis appelle la méthode annotée.

@Before: Lors de l'écriture des tests, il est courant de constater que plusieurs tests nécessitent des objets similaires créés avant de pouvoir être exécutés. Annoter une méthode vide publique avec `@Before` provoque l' `@Before` cette méthode avant la méthode Test.

@Après: Si vous **allouez** des ressources externes dans une méthode Before, vous devez les libérer après le test. Annoter une méthode vide publique avec `@After` provoque l' `@After` cette méthode après la méthode Test. Toutes les méthodes `@After` sont garanties pour s'exécuter même si une méthode Before ou Test émet une exception

---

Astuce Créez rapidement des classes de test dans Android Studio

- Placez le curseur sur le nom de la classe pour laquelle vous souhaitez créer une classe de test.
- Appuyez sur Alt + Entrée (Windows).
- Sélectionnez Créer un test, appuyez sur Retour.
- Sélectionnez les méthodes pour lesquelles vous souhaitez créer des méthodes de test, cliquez sur OK.
- Sélectionnez le répertoire dans lequel vous souhaitez créer la classe de test.
- Vous avez terminé, ce que vous obtenez est votre premier test.

Astuce Exécuter facilement des tests dans Android Studio

- Faites un clic droit pour tester le paquet.
- Sélectionnez Run 'Tests in ...

- Tous les tests du package seront exécutés immédiatement.

Déplacement de la logique métier hors des composants Android

Une grande partie de la valeur des tests unitaires JVM locaux provient de la façon dont vous concevez votre application. Vous devez le concevoir de telle sorte que vous puissiez découpler votre logique métier de vos composants Android. Voici un exemple d'utilisation du modèle Model-View-Presenter . Permet de s'exercer en mettant en place un écran d'inscription de base qui ne prend qu'un nom d'utilisateur et un mot de passe. Notre application Android est chargée de valider que le nom d'utilisateur fourni par l'utilisateur n'est pas vide et que le mot de passe comporte au moins huit caractères et au moins un chiffre. Si le nom d'utilisateur / mot de passe est valide, nous effectuons notre appel api d'inscription, sinon nous affichons un message d'erreur.

Exemple où la logique métier est fortement couplée au composant Android.

```
public class LoginActivity extends Activity{
    ...
    private void onSubmitButtonClicked(){
        String username = findViewById(R.id.username).getText().toString();
        String password = findViewById(R.id.password).getText().toString();
        boolean isUsernameValid = username != null && username.trim().length() != 0;
        boolean isPasswordValid = password != null && password.trim().length() >= 8 &&
password.matches(".*\\d+.*");
        if(isUsernameValid && isPasswordValid){
            performSignUpApiCall(username, password);
        } else {
            displayInvalidCredentialsErrorMessage();
        }
    }
}
```

Exemple où la logique métier est découplée du composant Android.

Nous définissons ici dans une seule classe, LoginContract, les différentes interactions entre nos différentes classes.

```
public interface LoginContract {
    public interface View {
        performSignUpApiCall(String username, String password);
        displayInvalidCredentialsErrorMessage();
    }
    public interface Presenter {
        void validateUserCredentials(String username, String password);
    }
}
```

Notre LoginActivity est pour la plupart identique, sauf que nous avons supprimé la responsabilité de savoir comment valider le formulaire d'inscription d'un utilisateur (notre logique métier). LoginActivity s'appuiera désormais sur notre nouveau LoginPresenter pour effectuer la validation.

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

Votre logique métier réside désormais dans votre nouvelle classe LoginPresenter.

```
public class LoginPresenter implements LoginContract.Presenter{
    private LoginContract.View view;

    public LoginPresenter(LoginContract.View view){
        this.view = view;
    }

    public void validateUserCredentials(String username, String password){
        boolean isUsernameValid = username != null && username.trim().length() != 0;
        boolean isPasswordValid = password != null && password.trim().length() >= 8 &&
password.matches(".*\\d+.*");
        if(isUsernameValid && isPasswordValid){
            view.performSignUpApiCall(username, password);
        } else {
            view.displayInvalidCredentialsErrorMessage();
        }
    }
}
```

Et maintenant, nous pouvons créer des tests unitaires JVM locaux par rapport à votre nouvelle classe LoginPresenter.

```
public class LoginPresenterTest {

    @Mock
    LoginContract.View view;

    private LoginPresenter presenter;

    @Before
    public void setUp() throws Exception {
        MockitoAnnotations.initMocks(this);
        presenter = new LoginPresenter(view);
    }

    @Test
    public void test_validateUserCredentials_userDidNotEnterUsername_displayErrorMessage()
throws Exception {
        String username = "";
        String password = "kingslayer1";
        presenter.validateUserCredentials(username, password);
```

```
        Mockito.verify(view). displayInvalidCredentialsErrorMessage();
    }

    @Test
    public void
test_validateUserCredentials_userEnteredFourLettersAndOneDigitPassword_displayErrorMessage()
throws Exception {
        String username = "Jaime Lanninster";
        String password = "king1";
        presenter.validateUserCredentials(username, password);
        Mockito.verify(view). displayInvalidCredentialsErrorMessage();
    }

    @Test
    public void
test_validateUserCredentials_userEnteredNineLettersButNoDigitsPassword_displayErrorMessage()
throws Exception {
        String username = "Jaime Lanninster";
        String password = "kingslayer";
        presenter.validateUserCredentials(username, password);
        Mockito.verify(view). displayInvalidCredentialsErrorMessage();
    }

    @Test
    public void
test_validateUserCredentials_userEnteredNineLettersButOneDigitPassword_performApiCallToSignUpUser()
throws Exception {
        String username = "Jaime Lanninster";
        String password = "kingslayer1";
        presenter.validateUserCredentials(username, password);
        Mockito.verify(view).performSignUpApiCall(username, password);
    }
}
```

Comme vous pouvez le constater, lorsque nous avons extrait notre logique métier de LoginActivity et l'avons placée dans le POJO LoginPresenter. Nous pouvons maintenant créer des tests unitaires JVM locaux en fonction de notre logique métier.

Il convient de noter que notre changement d'architecture a plusieurs autres implications, comme l'adhésion à chaque classe ayant une responsabilité unique, des classes supplémentaires, etc. Ce ne sont que des effets secondaires de la manière dont je choisis découplage via le style MVP. MVP n'est qu'une façon d'y parvenir, mais vous pouvez également envisager d'autres solutions, telles que MVVM . Il vous suffit de choisir le meilleur système qui fonctionne pour vous.

Démarrer avec JUnit

Installer

Pour lancer l'unité de test de votre projet Android à l'aide de JUnit, vous devez ajouter la dépendance JUnit à votre projet et créer un ensemble de sources de test qui contiendra le code source des tests unitaires. Les projets créés avec Android Studio incluent souvent déjà la dépendance JUnit et le jeu de sources de test

Ajoutez la ligne suivante à votre fichier `build.gradle` module dans les dépendances `Closure` :

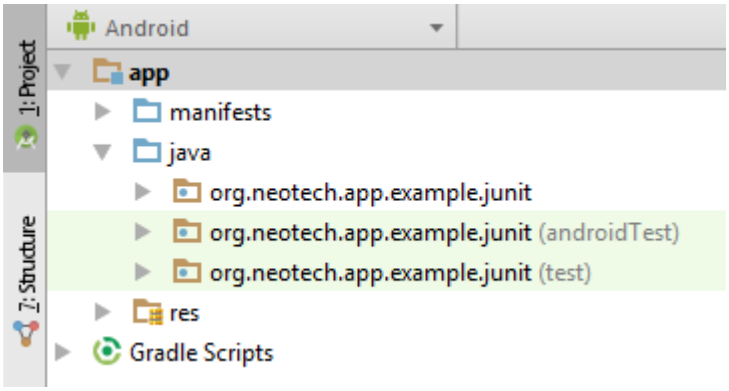
```
testCompile 'junit:junit:4.12'
```

Les classes de test JUnit se trouvent dans un ensemble de sources nommé `test` . Si cet ensemble de sources n'existe pas, vous devez créer un nouveau dossier vous-même. La structure de dossiers d'un projet Android Studio (basé sur Gradle) par défaut ressemble à ceci:

```
<project-root-folder>
    /app (module root folder)
        /build
        /libs
        /src
            /main (source code)
            /test (unit test source code)
            /androidTest (instrumentation test source code)
        build.gradle (module gradle file)
    /build
    /gradle
    build.gradle (project gradle file)
    gradle.properties
    gradlew
    gradlew.bat
    local.properties
    settings.gradle (gradle settings)
```

Si votre projet ne dispose pas du dossier `/app/src/test` , vous devez le créer vous-même. Dans le dossier de `test` , vous avez également besoin d'un dossier `java` (créez-le s'il n'existe pas). Le dossier Java dans le jeu de sources de `test` doit contenir la même structure de package que votre jeu de sources `main` .

Si la configuration est correcte, la structure de votre projet (dans la vue Android d'Android Studio) devrait ressembler à ceci:



Remarque: Vous n'avez pas nécessairement besoin du `androidTest` sources `androidTest` , cet ensemble de sources se trouve souvent dans les projets créés par Android Studio et est inclus ici pour référence.

Écrire un test

1. **Créez une nouvelle classe dans le jeu de sources de `test` .**
   Cliquez avec le bouton droit de la souris sur le jeu de sources de test dans la vue du projet,

choisissez `New` > `Java class` .
Le modèle de nommage le plus utilisé consiste à utiliser le nom de la classe que vous allez tester avec `Test` ajouté. Donc, `StringUtilities` devient `StringUtilitiesTest` .

2. **Ajouter l'annotation `@RunWith`**
   L'annotation `@RunWith` est nécessaire pour que JUnit exécute les tests que nous allons définir dans notre classe de test. Le runner JUnit par défaut (pour JUnit 4) est le `BlockJUnit4ClassRunner` mais, au lieu d'utiliser directement cette exécution, il est plus pratique d'utiliser l'alias `JUnit4` qui est un raccourci pour le runner JUnit par défaut.

```
@RunWith(JUnit4.class)
public class StringUtilitiesTest {

}
```

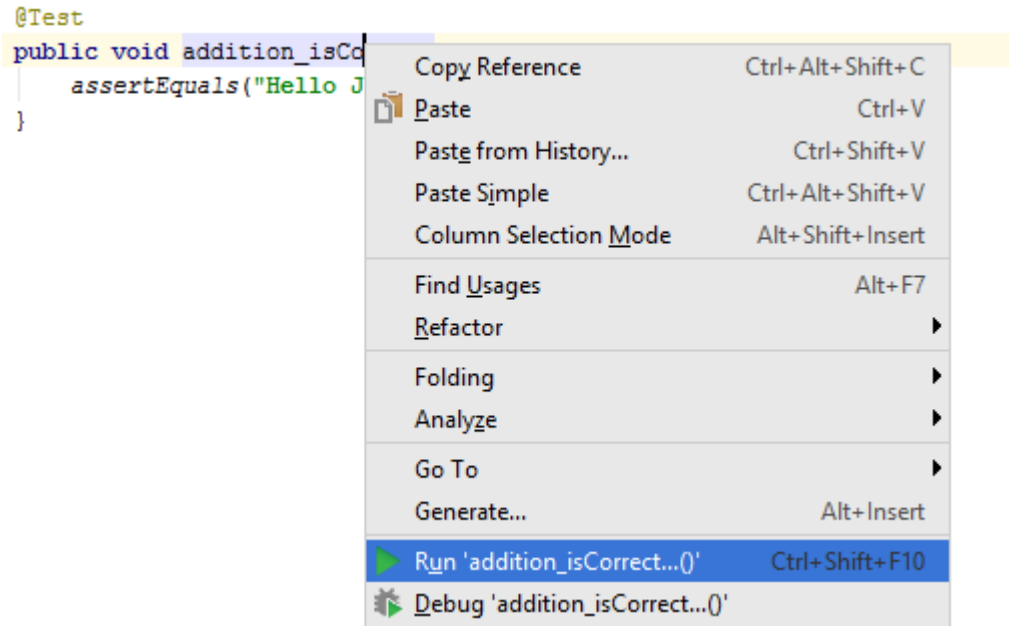
3. **Créer un test**
   Un test unitaire est essentiellement une méthode qui, dans la plupart des cas, ne doit pas échouer si elle est exécutée. En d'autres termes, il ne devrait pas lancer d'exception. Dans une méthode de test, vous trouverez presque toujours des assertions qui vérifient si des conditions spécifiques sont remplies. Si une assertion échoue, elle génère une exception qui entraîne l'échec de la méthode / du test. Une méthode de test est toujours annotée avec l'annotation `@Test` . Sans cette annotation, JUnit ne lancera pas automatiquement le test.

```
@RunWith(JUnit4.class)
public class StringUtilitiesTest {

    @Test
    public void addition_isCorrect() throws Exception {
        assertEquals("Hello JUnit", "Hello" + " " + "JUnit");
    }
}
```

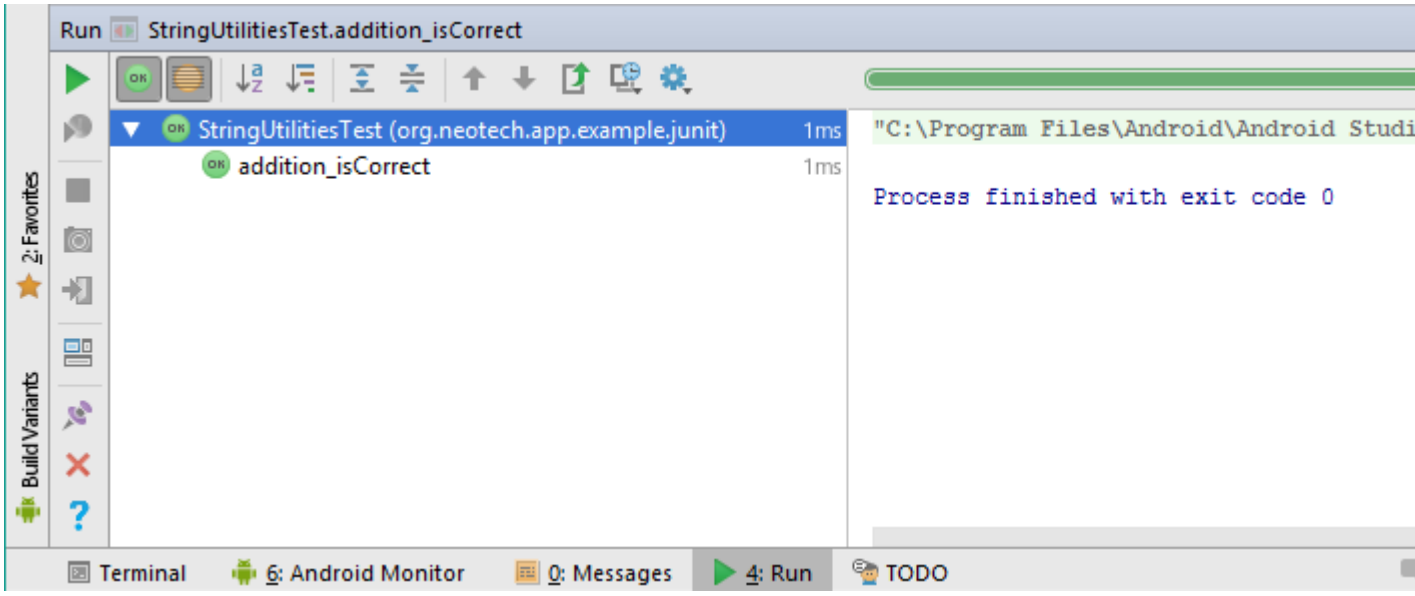
Remarque: contrairement à la méthode Java standard, les noms de méthode de test des unités de convention contiennent souvent des traits de soulignement.

Lancer un test

1. **Méthode**
   Pour exécuter une méthode de test unique, vous pouvez cliquer avec le bouton droit sur la méthode et cliquer sur `Run 'addition_isCorrect()'` ou utiliser le raccourci clavier `ctrl+shift+f10` .



Si tout est configuré correctement, JUnit commence à exécuter la méthode et vous devriez voir l'interface suivante dans Android Studio:



2. **Classe**
   Vous pouvez également exécuter tous les tests définis dans une seule classe, en cliquant avec le bouton droit sur la classe dans la vue du projet et en cliquant sur `Run 'StringUtilitiesTest '` ou utilisez le raccourci clavier `ctrl+shift+f10` si vous avez sélectionné la classe dans la vue du projet.

3. **Package (tout)**
   Si vous ne voulez pas exécuter tous les tests définis dans le projet ou dans un package, vous pouvez cliquer avec le bouton droit sur le package et cliquer sur `Run ...` comme si vous exécutiez tous les tests définis dans une seule classe.

Des exceptions

JUnit peut également être utilisé pour tester si une méthode renvoie une exception spécifique pour

une entrée donnée.

Dans cet exemple, nous allons tester si la méthode suivante renvoie vraiment une exception si le format booléen (input) n'est pas reconnu / inconnu:

```
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);
    }
}
```

En ajoutant le paramètre `expected` à l'annotation `@Test` , vous pouvez définir quelle exception est susceptible d'être lancée. Le test unitaire échouera si cette exception ne se produit pas et réussit si l'exception est bien levée:

```
@Test(expected = IllegalArgumentException.class)
public void parseBoolean_parsesInvalidFormat_throwsException(){
    StringUtilities.parseBoolean("Hello JUnit");
}
```

Cela fonctionne bien, cependant, cela vous limite à un seul cas de test dans la méthode. Parfois, vous pouvez vouloir tester plusieurs cas dans une même méthode. Une technique souvent utilisée pour surmonter cette limitation consiste à utiliser `try-catch` blocs `try-catch` et la méthode `Assert.fail()` :

```
@Test
public void parseBoolean_parsesInvalidFormats_throwsException(){
    try {
        StringUtilities.parseBoolean("Hello!");
        fail("Expected IllegalArgumentException");
    } catch(IllegalArgumentException e){
    }

    try {
        StringUtilities.parseBoolean("JUnit!");
        fail("Expected IllegalArgumentException");
    } catch(IllegalArgumentException e){
    }
}
```

Remarque: certaines personnes considèrent que tester un seul cas à l'intérieur d'un test unitaire est une mauvaise pratique.

Import statique

JUnit définit quelques méthodes `assertEquals` au moins une pour chaque type de primitive et une autre pour les objets. Ces méthodes ne sont pas directement disponibles par défaut pour être

appelées et doivent être appelées comme ceci: `Assert.assertEquals` . Mais comme ces méthodes sont utilisées, les utilisateurs utilisent souvent une importation statique, de sorte que la méthode peut être utilisée directement comme si elle faisait partie de la classe elle-même.

Pour ajouter une importation statique pour la méthode `assertEquals` , utilisez l'instruction d'importation suivante:

```
import static org.junit.Assert.assertEquals;
```

Vous pouvez également importer de manière statique toutes les méthodes d'assert, y compris `assertArrayEquals` , `assertNotNull` et `assertFalse` utilisant l'importation statique suivante:

```
import static org.junit.Assert.*;
```

Sans importation statique:

```
@Test
public void addition_isCorrect(){
    Assert.assertEquals(4 , 2 + 2);
}
```

Avec importation statique:

```
@Test
public void addition_isCorrect(){
    assertEquals(4 , 2 + 2);
}
```

Lire Tests unitaires sous Android avec JUnit en ligne:
https://riptutorial.com/fr/android/topic/3205/tests-unitaires-sous-android-avec-junit

Chapitre 240: Text to Speech (TTS)

Examples

Base texte-parole

layout_text_to_speech.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp">

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter text here!"
        android:id="@+id/textToSpeak"/>
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_below="@id/textToSpeak"
        android:id="@+id/btnSpeak"/>

</RelativeLayout>
```

AndroidTextToSpeechActivity.java

```
public class AndroidTextToSpeechActivity extends Activity implements
        TextToSpeech.OnInitListener {

    EditText textToSpeak = null;
    Button btnSpeak = null;
    TextToSpeech tts;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        textToSpeak = findViewById(R.id.textToSpeak);
        btnSpeak = findViewById(R.id.btnSpeak);
        btnSpeak.setEnabled(false);
        tts = new TextToSpeech(this, this);
        btnSpeak.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    speakOut();
                }
            });
    }

    @Override
    public void onDestroy() {
        // Don't forget to shutdown tts!
```

```
        if (tts != null) {
            tts.stop();
            tts.shutdown();
        }
        super.onDestroy();
    }

    @Override
    public void onInit(int status) {
        if (status == TextToSpeech.SUCCESS) {
            int result = tts.setLanguage(Locale.US);

            if (result == TextToSpeech.LANG_MISSING_DATA
                    || result == TextToSpeech.LANG_NOT_SUPPORTED) {
                Log.e("TTS", "This Language is not supported");
            } else {
                btnSpeak.setEnabled(true);
                speakOut();
            }
        } else {
            Log.e("TTS", "Initilization Failed!");
        }
    }

    private void speakOut() {
        String text = textToSpeak.getText().toString();
        if(text == null || text.isEmpty())
            return;

        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP) {
            String utteranceId=this.hashCode() + "";
            tts.speak(text, TextToSpeech.QUEUE_FLUSH, null, utteranceId);
        } else {
            tts.speak(text, TextToSpeech.QUEUE_FLUSH, null);
        }
    }
}
```

La langue à parler peut être définie en fournissant un `setLanguage() Locale` à la méthode `setLanguage()` :

```
tts.setLanguage(Locale.CHINESE); // Chinese language
```

Le nombre de langues prises en charge varie entre les niveaux Android. La méthode `isLanguageAvailable()` peut être utilisée pour vérifier si une langue `isLanguageAvailable()` est prise en charge:

```
tts.isLanguageAvailable(Locale.CHINESE);
```

Le niveau de hauteur de parole peut être défini à l'aide de la méthode `setPitch()` . Par défaut, la valeur de la hauteur est 1.0. Utilisez des valeurs inférieures à 1,0 pour diminuer le niveau de hauteur ou des valeurs supérieures à 1,0 pour augmenter le niveau de hauteur:

```
tts.setPitch(0.6);
```

Le débit de parole peut être défini à l'aide de `setSpeechRate()` . Le taux de parole par défaut est 1.0. Le taux de parole peut être doublé en le réglant sur 2.0 ou sur la moitié en le réglant sur 0.5:

```
tts.setSpeechRate(2.0);
```

Implémentation de TextToSpeech sur les API

L'implémentation observable à froid, émise lorsque le moteur TTS a fini de parler, commence à s'exprimer lorsqu'il est abonné. Notez que le niveau 21 de l'API introduit une manière différente d'exercer la parole:

```
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;
}
```

}

Lire Text to Speech (TTS) en ligne: https://riptutorial.com/fr/android/topic/3381/text-to-speech--tts-

Chapitre 241: TextInputLayout

Introduction

TextInputLayout a été introduit pour afficher l'étiquette flottante sur EditText. EditText doit être encapsulé par TextInputLayout afin d'afficher l'étiquette flottante.

Remarques

`TextInputLayout` est une mise en page qui `EditText` un `EditText` (ou un descendant) pour afficher une étiquette flottante lorsque le conseil est masqué en raison de la saisie de texte par l'utilisateur. De plus, `TextInputLayout` vous permet d'afficher un message d'erreur sous le `EditText` .

Assurez-vous que la dépendance suivante est ajoutée au fichier `build.gradle` votre application sous les dépendances:

```
compile 'com.android.support:design:25.3.1'
```

Examples

Utilisation de base

C'est l'utilisation de base de `TextInputLayout` .
Veillez à ajouter la dépendance dans le fichier `build.gradle` comme décrit dans la section remarques.

Exemple:

```
<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>
```

Gestion des erreurs

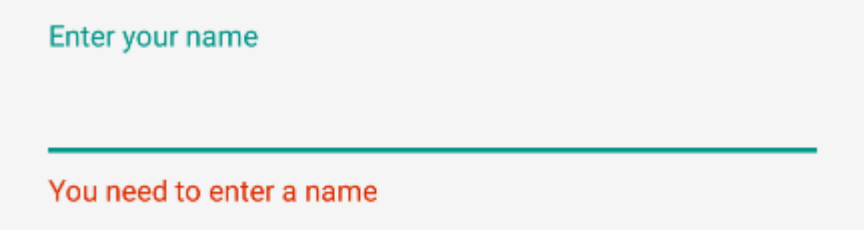
Vous pouvez utiliser `TextInputLayout` pour afficher les messages d'erreur conformément aux directives de conception de matériaux à l'aide des méthodes `setError` et `setErrorEnabled` .

Pour afficher l'erreur sous l'EditText, utilisez:

```
TextInputLayout til = (TextInputLayout) findViewById(R.id.username);
til.setErrorEnabled(true);
til.setError("You need to enter a name");
```

Pour activer l'erreur dans `TextInputLayout` vous pouvez utiliser `app:errorEnabled="true"` dans xml ou `til.setErrorEnabled(true);` Comme montré ci-dessus.

Vous obtiendrez:



Ajouter un compte de caractères

Le TextInputLayout a un compteur de caractères pour un EditText défini dans celui-ci.
Le compteur sera rendu en dessous du EditText.

Utilisez simplement les `setCounterEnabled()` et `setCounterMaxLength` :

```
TextInputLayout til = (TextInputLayout) findViewById(R.id.username);
til.setCounterEnabled(true);
til.setCounterMaxLength(15);
```

ou les `app:counterEnabled` et `app:counterMaxLength` dans le `app:counterMaxLength` XML.

```
<android.support.design.widget.TextInputLayout
    app:counterEnabled="true"
    app:counterMaxLength="15">

    <EditText/>

</android.support.design.widget.TextInputLayout>
```

La visibilité du mot de passe bascule

Avec un type de mot de passe en entrée, vous pouvez également activer une icône pouvant afficher ou masquer l'intégralité du texte à l'aide de l'attribut `passwordToggleEnabled` .

Vous pouvez également personnaliser la même valeur par défaut en utilisant ces attributs:

- `passwordToggleDrawable` : pour changer l'icône d'oeil par défaut
- `passwordToggleTint` : pour appliquer une teinte à l'index de visibilité du mot de passe.
- `passwordToggleTintMode` : pour spécifier le mode de fusion utilisé pour appliquer la teinte de fond.

Exemple:

```
<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

Le `TextInputEditText` est un `EditText` avec un correctif supplémentaire pour afficher un indice dans l'IME en mode 'extract' .

Le **mode d'extraction** est le mode sur lequel l'éditeur de clavier bascule lorsque vous cliquez sur un EditText lorsque l'espace est trop petit (par exemple, paysage sur un smartphone).
Dans ce cas, en utilisant un `EditText` pendant que vous éditez le texte, vous pouvez voir que l'IME ne vous donne pas une idée de ce que vous modifiez.

`TextInputEditText` résout ce problème en fournissant un indice lorsque l'IME de l'appareil de l'utilisateur est en mode Extract.

Exemple:

```
<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>
```

Personnalisation de l'apparence de TextInputLayout

Vous pouvez personnaliser l'apparence de `TextInputLayout` et de son `EditText` incorporé en définissant des styles personnalisés dans votre `styles.xml` . Les styles définis peuvent être ajoutés en tant que styles ou thèmes à votre `TextInputLayout` .

Exemple pour personnaliser l'apparence de l'indice:

`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>
```

Pour appliquer le style, mettez à jour votre TextInputLayout et EditText comme suit

```
<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>
```

Exemple pour personnaliser la couleur d'accent de `TextInputLayout` . La couleur d'accentuation affecte la couleur de la ligne de base du texte `EditText` et la couleur du texte du conseil flottant:

`styles.xml` :

```
<style name="TextInputLayoutWithPrimaryColor" parent="Widget.Design.TextInputLayout">
    <item name="colorAccent">@color/primary</item>
</style>
```

fichier de mise en page:

```
<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>
```

Lire TextInputLayout en ligne: https://riptutorial.com/fr/android/topic/5652/textinputlayout

Chapitre 242: Thème DayNight (AppCompat v23.2 / API 14+)

Examples

Ajout du thème DayNight à une application

Le thème DayNight permet à une application de changer de schéma de couleurs en fonction de l'heure et du dernier emplacement connu de l'appareil.

Ajoutez ce qui suit à votre `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>
```

Les thèmes à partir desquels vous pouvez ajouter une fonctionnalité de changement de thème de nuit sont les suivants:

- `"Theme.AppCompat.DayNight"`
- `"Theme.AppCompat.DayNight.NoActionBar"`
- `"Theme.AppCompat.DayNight.DarkActionBar"`

Outre `colorPrimary` , `colorPrimaryDark` et `colorAccent` , vous pouvez également ajouter d'autres couleurs que vous souhaitez changer, par exemple `textColorPrimary` ou `textColorSecondary` . Vous pouvez également ajouter les couleurs personnalisées de votre application à ce `style` .

Pour que le changement de thème fonctionne, vous devez définir un `colors.xml` par défaut dans le répertoire `res/values` et un `colors.xml` dans le `colors.xml res/values-night` et définir les couleurs jour / nuit de manière appropriée.

Pour changer de thème, appelez la `AppCompatDelegate.setDefaultNightMode(int)` partir de votre code Java. (Cela changera la palette de couleurs pour l'application entière, et pas seulement pour une activité ou un fragment.) Par exemple:

```
AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE_NIGHT_NO);
```

Vous pouvez passer l'un des trois suivants selon votre choix:

- `AppCompatDelegate.MODE_NIGHT_NO` : cela définit le thème par défaut pour votre application et prend les couleurs définies dans le répertoire `res/values` . Il est recommandé d'utiliser des couleurs claires pour ce thème.
- `AppCompatDelegate.MODE_NIGHT_YES` : définit un thème de nuit pour votre application et prend les couleurs définies dans le répertoire `res/values-night` . Il est recommandé d'utiliser des

couleurs sombres pour ce thème.

- `AppCompatDelegate.MODE_NIGHT_AUTO` : cette `AppCompatDelegate.MODE_NIGHT_AUTO` change automatiquement les couleurs de l'application en fonction de l'heure du jour et des couleurs que vous avez définies dans `values-night` répertoires de `values` et de `values-night` .

Il est également possible d'obtenir le statut actuel du mode nuit à l'aide de la méthode `getDefaultNightMode()` . Par exemple:

```
int modeType = AppCompatDelegate.getDefaultNightMode();
```

Veuillez noter, cependant, que le changement de thème ne persistera pas si vous tuez l'application et la rouvrez. Si vous faites cela, le thème reviendra à `AppCompatDelegate.MODE_NIGHT_AUTO` , qui est la valeur par défaut. Si vous souhaitez que le changement de thème persiste, assurez-vous de stocker la valeur dans les préférences partagées et de charger la valeur stockée à chaque ouverture de l'application après sa destruction.

Lire Thème DayNight (AppCompat v23.2 / API 14+) en ligne:
https://riptutorial.com/fr/android/topic/7650/theme-daynight--appcompat-v23-2---api-14plus-

Chapitre 243: Thème, Style, Attribut

Examples

Utiliser un thème personnalisé à l'échelle mondiale

Dans themes.xml:

```
<style name="AppTheme" parent="Theme.AppCompat">
    <!-- Theme attributes here -->
</style>
```

Dans AndroidManifest.xml:

```
<application
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme">

    <!-- Activity declarations here -->

</application>
```

Définir les couleurs primaires, primaires et d'accentuation

Vous pouvez personnaliser la palette de couleurs de votre thème .

Utilisation des API de **framework**

5.0

```
<style name="AppTheme" parent="Theme.Material">
    <item name="android:colorPrimary">@color/primary</item>
    <item name="android:colorPrimaryDark">@color/primary_dark</item>
    <item name="android:colorAccent">@color/accent</item>
</style>
```

Utilisation de la **bibliothèque de support Appcompat** (et `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>
```

Utiliser un thème personnalisé par activité

Dans themes.xml:

```
<style name="MyActivityTheme" parent="Theme.AppCompat">
    <!-- Theme attributes here -->
</style>
```

Dans 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>
```

Couleur Overscroll (API 21+)

```
<style name="AppTheme" parent="Theme.AppCompat">
    <item name="android:colorEdgeEffect">@color/my_color</item>
</style>
```

Couleur d'ondulation (API 21+)

5.0

L'animation d' ondulation est affichée lorsque l'utilisateur appuie sur des vues cliquables.

Vous pouvez utiliser la même couleur d'ondulation que celle utilisée par votre application pour attribuer le `?android:colorControlHighlight` à vos vues. Vous pouvez personnaliser cette couleur en modifiant l'attribut `android:colorControlHighlight` dans votre thème:

Cette couleur d'effet peut être modifiée:

```
<style name="AppTheme" parent="Theme.AppCompat">
    <item name="android:colorControlHighlight">@color/my_color</item>
</style>
```

Ou, si vous utilisez un thème matériel:

```
<style name="AppTheme" parent="android:Theme.Material.Light">
    <item name="android:colorControlHighlight">@color/your_custom_color</item>
</style>
```

Barre d'état de la lumière (API 23+)

Cet attribut peut changer l'arrière-plan des icônes de la barre d'état (en haut de l'écran) en blanc.

```
<style name="AppTheme" parent="Theme.AppCompat">
    <item name="android:windowLightStatusBar">true</item>
</style>
```

Barres de navigation et d'état translucides (API 19+)

La barre de navigation (en bas de l'écran) peut être transparente. Voici le moyen d'y parvenir.

```
<style name="AppTheme" parent="Theme.AppCompat">
    <item name="android:windowTranslucentNavigation">true</item>
</style>
```

La barre d'état (en haut de l'écran) peut être rendue transparente en appliquant cet attribut au style:

```
<style name="AppTheme" parent="Theme.AppCompat">
    <item name="android:windowTranslucentStatus">true</item>
</style>
```

Couleur de la barre de navigation (API 21+)

5.0

Cet attribut est utilisé pour modifier la barre de navigation (une, qui contient le bouton Précédent, Accueil récent). Il est généralement noir, mais sa couleur peut être modifiée.

```
<style name="AppTheme" parent="Theme.AppCompat">
    <item name="android:navigationBarColor">@color/my_color</item>
</style>
```

Héritage thématique

Lors de la définition des thèmes, on utilise généralement le thème fourni par le système, puis les modifications modifient l'apparence pour l'adapter à sa propre application. Par exemple, `Theme.AppCompat` comment le thème `Theme.AppCompat` est hérité:

```
<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>
```

Ce thème a maintenant toutes les propriétés du thème `Theme.AppCompat` standard, sauf celles que nous avons explicitement modifiées.

Il y a aussi un raccourci lors de l'héritage, généralement utilisé lorsque l'on hérite de son propre thème:

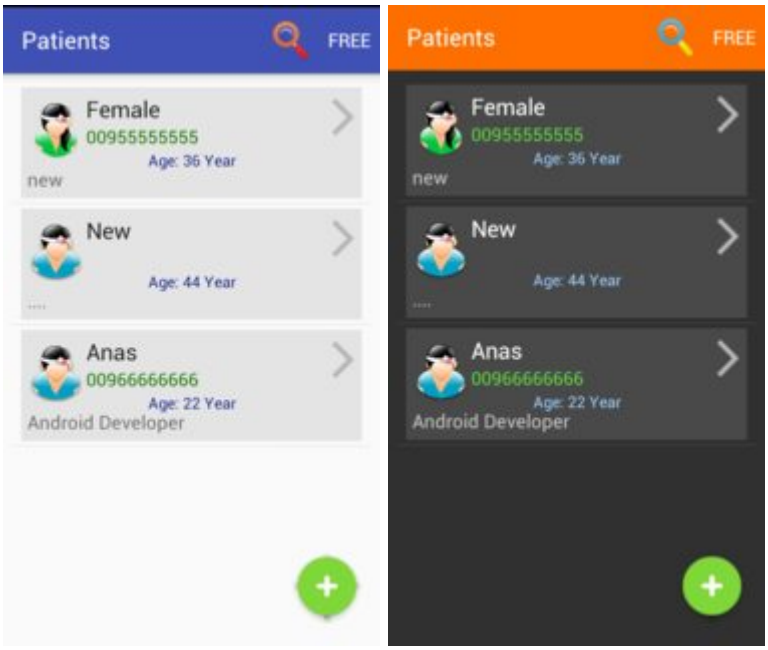
```
<style name="AppTheme.Red">
```

```
    <item name="colorAccent">@color/red</item>
</style>
```

Depuis, il a déjà `AppTheme.` au début de son nom, il en hérite automatiquement, sans avoir à définir le thème `parent` . Ceci est utile lorsque vous devez créer des styles spécifiques pour une pièce (par exemple, une seule activité) de votre application.

Plusieurs thèmes dans une application

En utilisant plus d'un thème dans votre application Android, vous pouvez ajouter des couleurs personnalisées à chaque thème pour qu'elles se présentent comme suit:



Tout d'abord, nous devons ajouter nos thèmes à `style.xml` comme ceci:

```
<style name="OneTheme" parent="Theme.AppCompat.Light.DarkActionBar">

</style>

<!--  -->
<style name="TwoTheme" parent="Theme.AppCompat.Light.DarkActionBar" >

</style>
......
```

Vous pouvez voir **ci-** dessus **OneTheme** et **TwoTheme** .

Maintenant, allez à votre `AndroidManifest.xml` et ajoutez cette ligne: `android:theme="@style/OneTheme"` à votre balise d' *application* , cela fera de **OneTheme** le thème par défaut:

```
<application
        android:theme="@style/OneTheme"
        ...>
```

Créez un nouveau fichier xml nommé `attrs.xml` et ajoutez ce code:

```
<?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) -->
```

Revenez à `style.xml` et ajoutez ces couleurs avec ses valeurs pour chaque thème:

```
<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>
```

Maintenant, vous avez des couleurs personnalisées pour chaque thème, ajoutons ces couleurs à nos vues.

Ajoutez la couleur **custom_blue** au TextView en utilisant "? Attr /":

Accédez à votre imageView et ajoutez cette couleur:

```
<TextView>
    android:id="@+id/txte_view"
    android:textColor="?attr/custom_blue" />
```

Mow, nous pouvons changer le thème par simple ligne `setTheme(R.style.TwoTheme);` cette ligne doit être avant `setContentView()` Procédé `onCreate()` méthode, comme celle - `Activity.java` :

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setTheme(R.style.TwoTheme);
    setContentView(R.layout.main_activity);
    ....
}
```

---

changer de thème pour toutes les activités à la fois

Si nous voulons changer le thème pour toutes les activités, nous devons créer une nouvelle classe nommée MyActivity `AppCompatActivity` (ou `Activity` class) et ajouter line `setTheme(R.style.TwoTheme);` à la méthode **onCreate ()** :

```
public class MyActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        if (new MySettings(this).isDarkTheme())
            setTheme(R.style.TwoTheme);
    }
}
```

Enfin, accédez à toutes vos activités, ajoutez toutes à la classe de base **MyActivity** :

```
public class MainActivity extends MyActivity {
    ....
}
```

Pour changer de thème, il suffit d'aller sur **MyActivity** et de modifier `R.style.TwoTheme` sur votre thème (`R.style.OneTheme` , `R.style.ThreeTheme` ....).

Lire Thème, Style, Attribut en ligne: https://riptutorial.com/fr/android/topic/1843/theme--style--attribut

Chapitre 244: Tiroirs

Examples

Teinte à dessiner

Un tirable peut être teinté d'une certaine couleur. Cela est utile pour prendre en charge différents thèmes dans votre application et réduire le nombre de fichiers de ressources pouvant être dessinés.

Utilisation des API du framework sur le SDK 21+:

```
Drawable d = context.getDrawable(R.drawable.ic_launcher);
d.setTint(Color.WHITE);
```

Utilisation de la bibliothèque android.support.v4 sur le 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
```

Veuillez ne pas que la `int color` **ne se** réfère **pas** à une ressource couleur, mais vous n'êtes pas limité aux couleurs définies dans la classe 'Couleur'. Lorsque vous avez une couleur définie dans votre XML que vous souhaitez utiliser, vous devez d'abord obtenir sa valeur.

Vous pouvez remplacer les utilisations de `Color.WHITE` utilisant les méthodes ci-dessous

Lorsque vous ciblez les anciennes API:

```
getResources().getColor(R.color.your_color);
```

Ou sur des cibles plus récentes:

```
ContextCompat.getColor(context, R.color.your_color);
```

Make View avec des coins arrondis

Créez **un** fichier **pouvant être dessiné** nommé avec **custom_rectangle.xml** dans le dossier

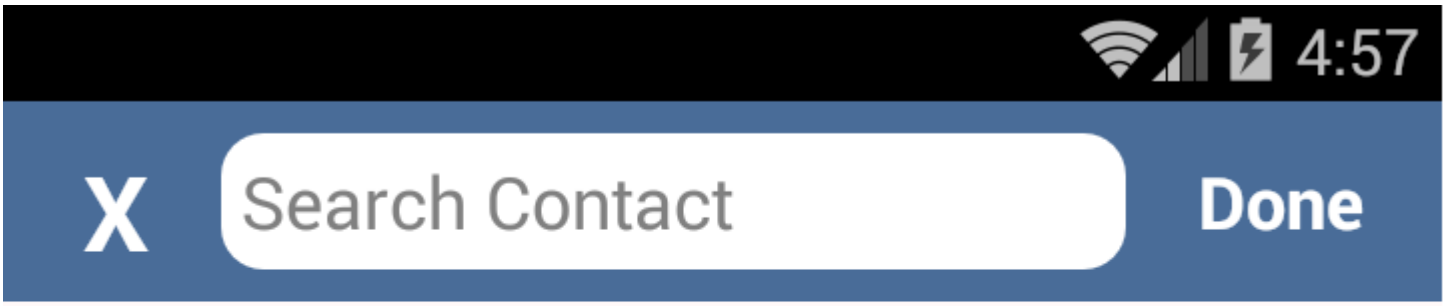
pouvant être dessiné :

```
<?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>
```

Maintenant, appliquez le **fond de rectangle** sur **View** :

```
mView.setBackGround(R.drawlable.custom_rectangle);
```

Capture d'écran de référence:



Vue circulaire

Pour une vue circulaire (dans ce cas, `TextView`), créez un fichier **round_view.xml** dans **un** dossier **drawble** :

```
<?xml version="1.0" encoding="utf-8"?>
<shape
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="oval">
    <solid android:color="#FAA23C" />
    <stroke android:color="#FFF" android:width="2dp" />
</shape>
```

Attribuez le dessin à la vue:

```
    <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" />
```

Maintenant, il devrait ressembler au cercle orange:

Personnalisable

Étendez votre classe avec Drawable et remplacez ces méthodes

```
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;
    }
}
```

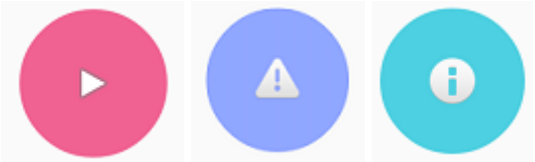
Déclarez une image dans votre mise en page

```
<ImageView
   android:layout_width="80dp"
   android:id="@+id/imageView"
   android:layout_height="80dp" />
```

Définissez votre dessin personnalisé sur ImageView

```
 IconDrawable iconDrawable=new
IconDrawable(ContextCompat.getDrawable(this,android.R.drawable.ic_media_play),ContextCompat.getColor(th

 imageView.setImageDrawable(iconDrawable);
```

Capture d'écran



Lire Tiroirs en ligne: https://riptutorial.com/fr/android/topic/4841/tiroirs

Chapitre 245: Touch Events

Examples

Comment varier entre les événements tactiles des groupes de vues enfants et parents

1. `onTouchEvents()` pour les groupes de vues imbriqués peut être géré par le `boolean` onInterceptTouchEvent .

La valeur par défaut pour `OnInterceptTouchEvent` est false.

Le `onTouchEvent` du `onTouchEvent` est reçu avant l'enfant. Si `OnInterceptTouchEvent` renvoie false, il envoie l'événement motion en aval de la chaîne au gestionnaire `OnTouchEvent` l'enfant. Si elle retourne vrai, le parent va gérer l'événement tactile.

Cependant, il peut y avoir des cas où nous voulons que certains éléments enfants gèrent `OnTouchEvent` s et que certains soient gérés par la vue parent (ou éventuellement le parent du parent).

Cela peut être géré de plusieurs façons.

2. Une manière de protéger un élément enfant du `OnInterceptTouchEvent` du `OnInterceptTouchEvent` consiste à implémenter le requestDisallowInterceptTouchEvent .

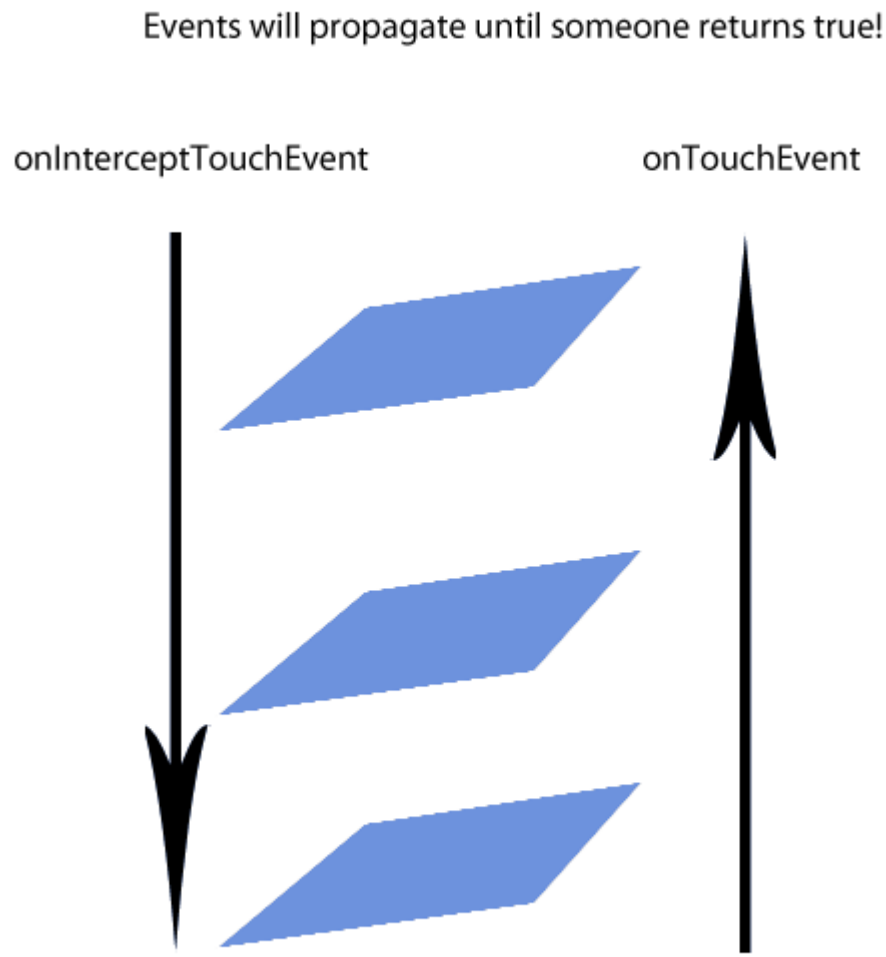
   public void requestDisallowInterceptTouchEvent (booléen disallowIntercept)

Cela empêche toute vue parent de gérer le `OnTouchEvent` pour cet élément si l'élément possède des gestionnaires d'événements activés.

3.

Si le `OnInterceptTouchEvent` est false, l'élément `OnTouchEvent` l'élément enfant sera évalué. Si vous avez des méthodes dans les éléments enfants qui gèrent les différents événements tactiles, tous les gestionnaires d'événements liés qui sont désactivés renverront le OnTouchEvent au parent.

Cette réponse:
Une visualisation de la manière dont la propagation des événements tactiles passe:
`parent -> child|parent -> child|parent -> child views.`



Courtoisie d'ici

4. Une autre méthode consiste à renvoyer des valeurs variables à partir du `OnInterceptTouchEvent` pour le parent.

Cet exemple, tiré de la gestion des événements tactiles dans un groupe ViewGroup , montre comment intercepter le `OnTouchEvent` l'enfant lorsque l'utilisateur fait défiler.

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;
}
```

Ceci est un code du même lien montrant comment créer les paramètres du rectangle autour de votre élément:

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);
}
```

Lire Touch Events en ligne: https://riptutorial.com/fr/android/topic/7167/touch-events

Chapitre 246: TransitionDrawable

Examples

Ajouter une transition ou un fondu enchaîné entre deux images.

Étape 1: Créer une transition pouvant être dessinée en XML

Enregistrez ce fichier `transition.xml` dans le dossier `res/drawable` de votre projet.

```
<transition xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:drawable="@drawable/image1"/>
    <item android:drawable="@drawable/image2"/>
</transition>
```

Les images1 et image2 sont les deux images que nous souhaitons transformer et elles doivent également être placées dans votre dossier `res/drawable` .

Étape 2: Ajoutez du code pour ImageView dans votre mise en page XML pour afficher le dessin ci-dessus.

```
<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>
```

Étape 3: Accédez à la transition XML pouvant être dessinée dans la méthode onCreate () de votre activité et commencez la transition dans l'événement 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);
        }
    });
}
```

Animer la couleur d'arrière-plan des vues (couleur de commutation) avec 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);
}
```

Lire TransitionDrawable en ligne: https://riptutorial.com/fr/android/topic/6088/transitiondrawable

Chapitre 247: Transitions d'éléments partagés

Introduction

Vous trouverez ici des exemples de transition entre les `Activities` ou les `Fragments` aide d'un élément partagé. Un exemple de ce comportement est l'application Google Play Store, qui traduit l'icône d'une application de la liste en vue détaillée de l'application.

Syntaxe

- transaction.addSharedElement (sharedElementView, "targetTransitionName");
- fragment.setSharedElementEnterTransition (new CustomTransaction ());

Examples

Transition d'éléments partagés entre deux fragments

Dans cet exemple, l'un des deux `ImageViews` différents doit être traduit du `ChooserFragment` au `DetailFragment` .

Dans la disposition `ChooserFragment` , nous avons besoin des attributs uniques `transitionName` :

```
<ImageView
    android:id="@+id/image_first"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/ic_first"
    android:transitionName="fistImage" />

<ImageView
    android:id="@+id/image_second"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/ic_second"
    android:transitionName="secondImage" />
```

Dans la classe `ChooserFragments` , nous devons transmettre la `View` laquelle vous `ChooserFragments` cliqué et un ID à l' `Activity` parente qui gère le remplacement des fragments (nous avons besoin de l'ID pour savoir quelle ressource d'image afficher dans le `DetailFragment`). Comment transmettre des informations à une activité parent en détail est sûrement couvert dans une autre documentation.

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

Dans le `DetailFragment` , l' `ImageView` de l'élément partagé nécessite également l'attribut unique `transitionName` .

```
<ImageView
    android:id="@+id/image_shared"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:transitionName="sharedImage" />
```

Dans la méthode `onCreateView()` du `DetailFragment` , nous devons décider quelle ressource d'image doit être affichée (si nous ne le faisons pas, l'élément partagé disparaîtra après la transition).

```
public static DetailFragment newInstance(Bundle args) {
    DetailFragment fragment = new DetailFragment();
    fragment.setArguments(args);
    return fragment;
}

@Nullable
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
    super.onCreateView(inflater, container, savedInstanceState);
    View view = inflater.inflate(R.layout.fragment_detail, container, false);

    ImageView sharedImage = (ImageView) view.findViewById(R.id.image_shared);

    // Check which resource should be shown.
    int type = getArguments().getInt("type");

    // Show image based on the type.
    switch (type) {
        case 1:
            sharedImage.setBackgroundResource(R.drawable.ic_first);
            break;

        case 2:
            sharedImage.setBackgroundResource(R.drawable.ic_second);
            break;
    }

    return view;
}
```

L' `Activity` parent reçoit les rappels et gère le remplacement des fragments.

```
@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();
}
```

Ne pas oublier - la `Transition` elle-même. Cet exemple déplace et met à l'échelle l'élément partagé.

```
@TargetApi(Build.VERSION_CODES.LOLLIPOP)
public class DetailsTransition extends TransitionSet {

    public DetailsTransition() {
        setOrdering(ORDERING_TOGETHER);
        addTransition(new ChangeBounds()).
            addTransition(new ChangeTransform()).
            addTransition(new ChangeImageTransform());
    }

}
```

Lire Transitions d'éléments partagés en ligne:
https://riptutorial.com/fr/android/topic/8933/transitions-d-elements-partages

Chapitre 248: Un service

Introduction

Un service s'exécute en **arrière-plan** pour effectuer des opérations de longue durée ou pour exécuter des tâches pour des processus distants. Un service ne fournit aucune interface utilisateur, il s'exécute uniquement en arrière-plan avec les entrées de l'utilisateur. Par exemple, un service peut lire de la musique en arrière-plan lorsque l'utilisateur se trouve dans une autre application ou il peut télécharger des données sur Internet sans bloquer l'interaction de l'utilisateur avec l'appareil Android.

Remarques

Si vous n'avez pas défini votre service dans votre fichier AndroidManifest.xml, vous recevrez une exception `ServiceNotFoundException` lorsque vous tentez de le démarrer.

Remarque:

Pour plus d'informations sur `IntentService` , voir ici: IntentService Example

Examples

Démarrer un service

Démarrer un service est très simple, il suffit d'appeler `startService` avec une intention, à partir d'une activité:

```
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.
```

Cycle de vie d'un service

Le cycle de vie des services comporte les rappels suivants

- `onCreate()` :

Exécuté lors de la création du service afin de configurer les configurations initiales dont vous pourriez avoir besoin. Cette méthode est exécutée uniquement si le service n'est pas déjà en cours d'exécution.

- `onStartCommand()` :

Exécuté chaque fois que `startService()` est `startService()` par un autre composant, comme une

activité ou un BroadcastReceiver. Lorsque vous utilisez cette méthode, le service s'exécute jusqu'à ce que vous `stopSelf()` ou `stopService()` . Notez que peu importe le nombre de fois que vous appelez `onStartCommand()` , les méthodes `stopSelf()` et `stopService()` doivent être appelées une seule fois pour arrêter le service.

- `onBind()` :

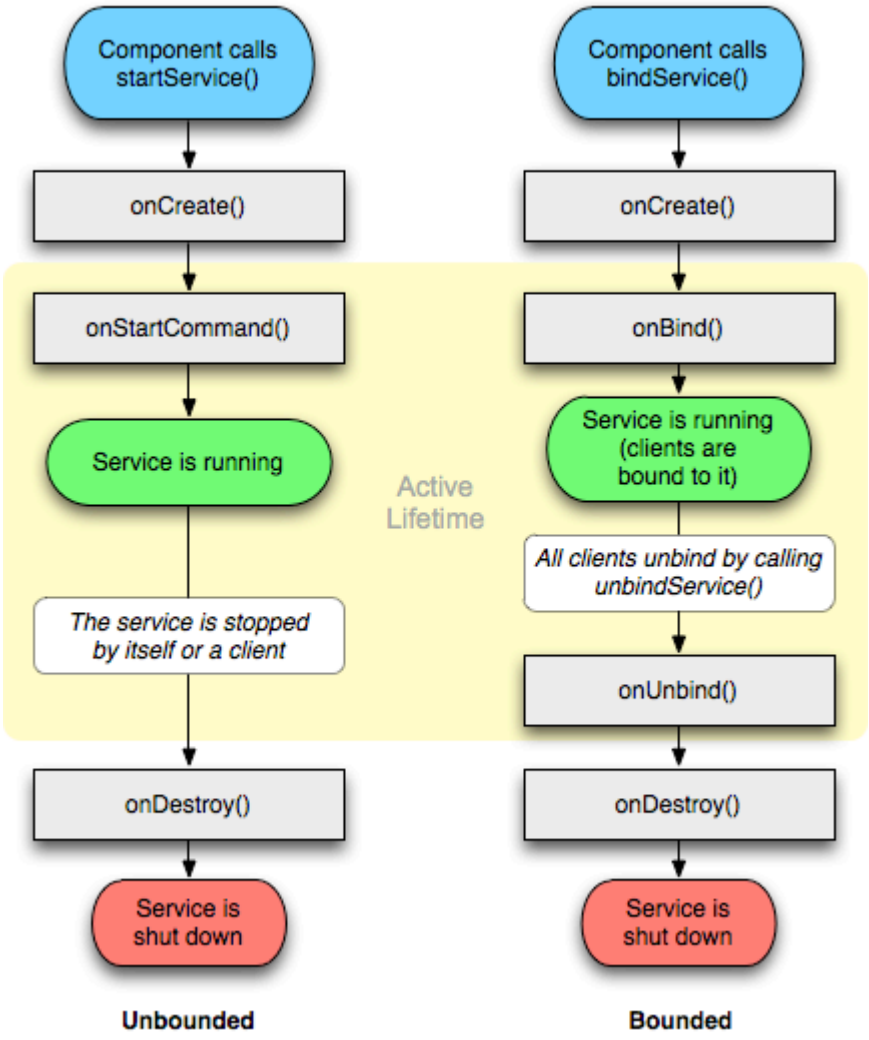
Exécuté lorsqu'un composant appelle `bindService()` et renvoie une instance de IBInder, fournissant un canal de communication au service. Un appel à `bindService()` maintiendra le service en cours d'exécution tant que des clients y sont liés.

- `onDestroy()` :

Exécuté lorsque le service n'est plus utilisé et permet la mise au rebut des ressources allouées.

Il est important de noter qu'au cours du cycle de vie d'un service, d'autres rappels peuvent être `onConfigurationChanged()` tels que `onConfigurationChanged()` et `onLowMemory()`

https://developer.android.com/guide/components/services.html



Définir le processus d'un service

Le champ `android:process` définit le nom du processus sur lequel le service doit s'exécuter.

Normalement, tous les composants d'une application s'exécutent dans le processus par défaut créé pour l'application. Toutefois, un composant peut remplacer la valeur par défaut par son propre attribut de processus, ce qui vous permet de répartir votre application sur plusieurs processus.

Si le nom attribué à cet attribut commence par deux points (':'), le service s'exécutera dans un processus distinct.

```
<service
  android:name="com.example.appName"
  android:process=":externalProcess" />
```

Si le nom du processus commence par un caractère minuscule, le service s'exécutera dans un processus global de ce nom, à condition qu'il soit autorisé à le faire. Cela permet aux composants de différentes applications de partager un processus, réduisant ainsi l'utilisation des ressources.

Création d'un service lié avec l'aide du classeur

Créez une classe qui étend la classe `Service` et, dans la méthode `onBind` renvoyez votre instance de classeur locale:

```
public class LocalService extends Service {
    // Binder given to clients
    private final IBinder mBinder = new LocalBinder();

    /**
     * Class used for the client Binder.  Because we know this service always
     * runs in the same process as its clients, we don't need to deal with IPC.
     */
    public class LocalBinder extends Binder {
        LocalService getService() {
            // Return this instance of LocalService so clients can call public methods
            return LocalService.this;
        }
    }

    @Override
    public IBinder onBind(Intent intent) {
        return mBinder;
    }
}
```

Ensuite, dans votre activité, associez-vous au service dans le rappel `onStart` , en utilisant l'instance `ServiceConnection` et en vous `onStop` dans `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;
        }
    };
}
```

Créer un service à distance (via AIDL)

Décrivez votre interface d'accès au service via le fichier `.aidl` :

```
// IRemoteService.aidl
package com.example.android;

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

/** Example service interface */
interface IRemoteService {
    /** Request the process ID of this service, to do evil things with it. */
    int getPid();
}
```

Maintenant, après l'application de construction, les outils sdk `.java` fichier `.java` approprié. Ce fichier contiendra la classe `Stub` qui implémente notre interface aidl et que nous devons étendre:

```
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;
    }
}
```

Puis en activité:

```
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;
    }
```

```
}
```

Créer un service non lié

La première chose à faire est d'ajouter le service à `AndroidManifest.xml` , à l'intérieur de la `<application>` :

```
<application ...>

    ...

    <service
        android:name=".RecordingService"
        <!--"enabled" tag specifies Whether or not the service can be instantiated by the
system – "true" -->
        <!--if it can be, and "false" if not. The default value is "true".-->
        android:enabled="true"
        <!--exported tag specifies Whether or not components of other applications can invoke
the -->
        <!--service or interact with it – "true" if they can, and "false" if not. When the
value-->
        <!--is "false", only components of the same application or applications with the same
user -->
        <!--ID can start the service or bind to it.-->
        android:exported="false" />

</application>
```

Si vous avez l'intention de gérer votre classe de service dans un package séparé (par exemple: .AllServices.RecordingService), vous devrez spécifier l'emplacement de votre service. Donc, dans le cas ci-dessus, nous modifierons:

```
android:name=".RecordingService"
```

à

```
android:name=".AllServices.RecordingService"
```

ou le moyen le plus simple est de spécifier le nom complet du package.

Ensuite, nous créons la classe de service actuelle:

```
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();
    }

}
```

Tout ce que fait ce service est de montrer une notification quand il est en cours d'exécution, et il peut afficher des toasts lorsque sa méthode `doSomething()` est appelée.

Comme vous le remarquerez, il est implémenté en tant que singleton , en gardant une trace de sa propre instance, mais sans la méthode habituelle de fabrique de singleton, car les services sont naturellement conçus et créés par intents. L'instance est utile à l'extérieur pour obtenir un "handle" au service lorsqu'il est en cours d'exécution.

Enfin, nous devons démarrer et arrêter le service d’une activité:

```
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);
    }
}
```

Dans cet exemple, le service est démarré et arrêté par la même méthode, en fonction de son état actuel.

Nous pouvons également invoquer la méthode `doSomething()` de notre activité:

```
public void makeServiceDoSomething(){
    if( RecordingService.isRunning )
        RecordingService.instance.doSomething();
}
```

Lire Un service en ligne: https://riptutorial.com/fr/android/topic/137/un-service

Chapitre 249: URL de rappel

Examples

Exemple d'URL de rappel avec Instagram OAuth

Un des cas d'utilisation des *URL* de *rappel* est OAuth. Faisons-le avec une connexion Instagram: Si l'utilisateur entre ses informations d'identification et clique sur le bouton *Connexion* , Instagram valide les informations d'identification et retourne un `access_token` . Nous avons besoin de cet `access_token` dans notre application.

Pour que notre application puisse écouter de tels liens, nous devons ajouter une URL de rappel à notre `Activity` . Nous pouvons le faire en ajoutant un `<intent-filter/>` à notre `Activity` , qui réagira à cette URL de rappel. Supposons que notre URL de rappel soit `appSchema://appName.com` . Ensuite, vous devez ajouter les lignes suivantes à l' `Activity` souhaitée dans le fichier *Manifest.xml* :

```
<action android:name="android.intent.action.VIEW" />
<category android:name="android.intent.category.BROWSABLE"/>
<data android:host="appName.com" android:scheme="appSchema"/>
```

Explication des lignes ci-dessus:

- `<category android:name="android.intent.category.BROWSABLE"/>` permet à l'activité cible d'être lancée par un navigateur Web pour afficher les données référencées par un lien.
- `<data android:host="appName.com" android:scheme="appSchema"/>` spécifie notre schéma et l'hôte de notre URL de rappel.
- Tous ensemble, ces lignes entraînent l'ouverture de l' `Activity` spécifique chaque fois que l'URL de rappel est appelée dans un navigateur.

Maintenant, pour obtenir le contenu de l'URL dans votre `Activity` , vous devez remplacer la méthode `onResume()` comme suit:

```
@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.
}
```

Vous avez maintenant récupéré le `access_token` d'Instagram, qui est utilisé dans divers points de terminaison d'API d'Instagram.

Lire URL de rappel en ligne: https://riptutorial.com/fr/android/topic/4790/url-de-rappel

Chapitre 250: Validation du courrier électronique

Examples

Validation de l'adresse email

Ajoutez la méthode suivante pour vérifier si une adresse électronique est valide ou non:

```
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();
}
```

La méthode ci-dessus peut facilement être vérifiée en convertissant le texte d'un widget `EditText` en `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();
}
```

Validation de l'adresse e-mail à l'aide de Patterns

```
  if (Patterns.EMAIL_ADDRESS.matcher(email).matches()){
       Log.i("EmailCheck","It is valid");
  }
```

Lire Validation du courrier électronique en ligne:
https://riptutorial.com/fr/android/topic/5605/validation-du-courrier-electronique

Chapitre 251: VectorDrawable et AnimatedVectorDrawable

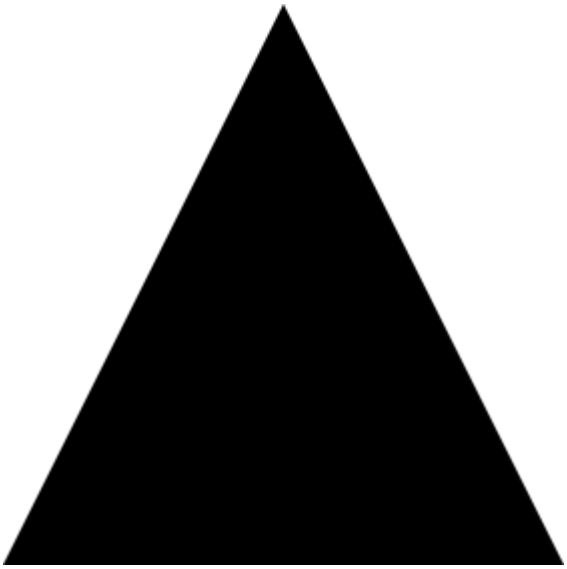
Examples

Basic VectorDrawable

Un `VectorDrawable` doit comporter au moins une `<path>` définissant une forme

```
<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>
```

Cela produirait un triangle noir:



En utilisant

Un `<clip-path>` définit une forme qui agit comme une fenêtre, permettant uniquement à des parties d'un `<path>` de montrer si elles se trouvent dans la forme `<clip-path>` et de couper le reste.

```
<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>
```

Dans ce cas, le `<path>` produit un triangle noir, mais le `<clip-path>` définit une forme carrée plus petite, ne permettant qu'une partie du triangle:

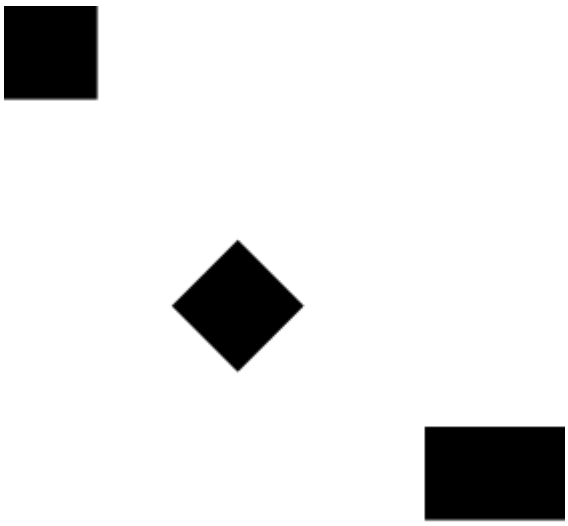
Mots clés

Une `<group>` permet d'ajuster, de modifier et de positionner un ou plusieurs éléments d'un objet `VectorDrawable` :

```
<vector xmlns:android="http://schemas.android.com/apk/res/android"
android:width="24dp"
android:height="24dp"
android:viewportWidth="24.0"
android:viewportHeight="24.0">
    <path
        android:pathData="M0,0 h4 v4 h-4 z"
        android:fillColor="#FF000000"/>

    <group
        android:name="middle square group"
        android:translateX="10"
        android:translateY="10"
        android:rotation="45">
        <path
            android:pathData="M0,0 h4 v4 h-4 z"
            android:fillColor="#FF000000"/>
    </group>

    <group
        android:name="last square group"
        android:translateX="18"
        android:translateY="18"
```

```
        android:scaleX="1.5">
        <path
            android:pathData="M0,0 h4 v4 h-4 z"
            android:fillColor="#FF000000"/>
    </group>
</vector>
```

L'exemple de code ci-dessus contient trois balises `<path>` identiques, toutes décrivant des carrés noirs. Le premier carré n'est pas ajusté. Le deuxième carré est enveloppé dans une `<group>` qui le déplace et le fait pivoter de 45 °. Le troisième carré est entouré d'une `<group>` qui le déplace et l'étire horizontalement de 50%. Le résultat est le suivant:



Une `<group>` peut contenir plusieurs balises `<path>` et `<clip-path>` . Il peut même contenir un autre `<group>` .

AnimatedVectorDrawable de base

Un `AnimatedVectorDrawable` nécessite au moins 3 composants:

- Un `VectorDrawable` qui sera manipulé
- Un `objectAnimator` qui définit quelle propriété à modifier et comment
- `AnimatedVectorDrawable` lui `AnimatedVectorDrawable` même qui connecte `objectAnimator` à `VectorDrawable` pour créer l'animation

Ce qui suit crée un triangle qui transforme sa couleur du noir au rouge.

Le `VectorDrawable` , nom du fichier: `triangle_vector_drawable.xml`

```
<vector xmlns:android="http://schemas.android.com/apk/res/android"
    android:width="24dp"
    android:height="24dp"
    android:viewportWidth="24.0"
    android:viewportHeight="24.0">

    <path
        android:name="triangle"
        android:fillColor="@android:color/black"
```

```
        android:pathData="M0,24 l12,-24 l12,24 z"/>

</vector>
```

`objectAnimator` , filename: `color_change_animator.xml`

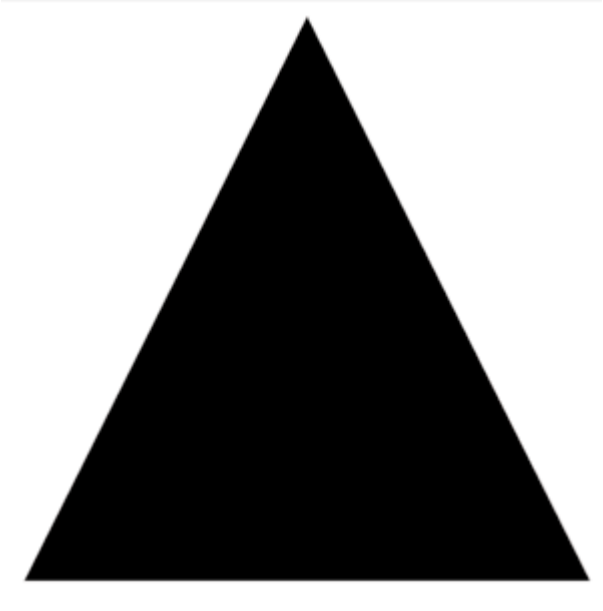
```
<objectAnimator xmlns:android="http://schemas.android.com/apk/res/android"
    android:propertyName="fillColor"
    android:duration="2000"
    android:repeatCount="infinite"
    android:valueFrom="@android:color/black"
    android:valueTo="@android:color/holo_red_light"/>
```

`AnimatedVectorDrawable` , nom du fichier: `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>
```

Notez que `<target>` spécifie `android:name="triangle"` qui correspond au `<path>` dans le `VectorDrawable` . Un objet `VectorDrawable` peut contenir plusieurs éléments et la propriété `android:name` est utilisée pour définir l'élément qui est ciblé.

Résultat:



Utiliser des traits

L'utilisation du trait SVG facilite la création d'un dessin vectoriel avec une longueur de trait unifiée, conformément aux directives de conception du matériau :

> Des poids de trait constants sont essentiels pour unifier la famille des icônes du système. Conservez une largeur de 2dp pour toutes les instances de trait, y compris les courbes, les angles et les courses intérieures et extérieures.

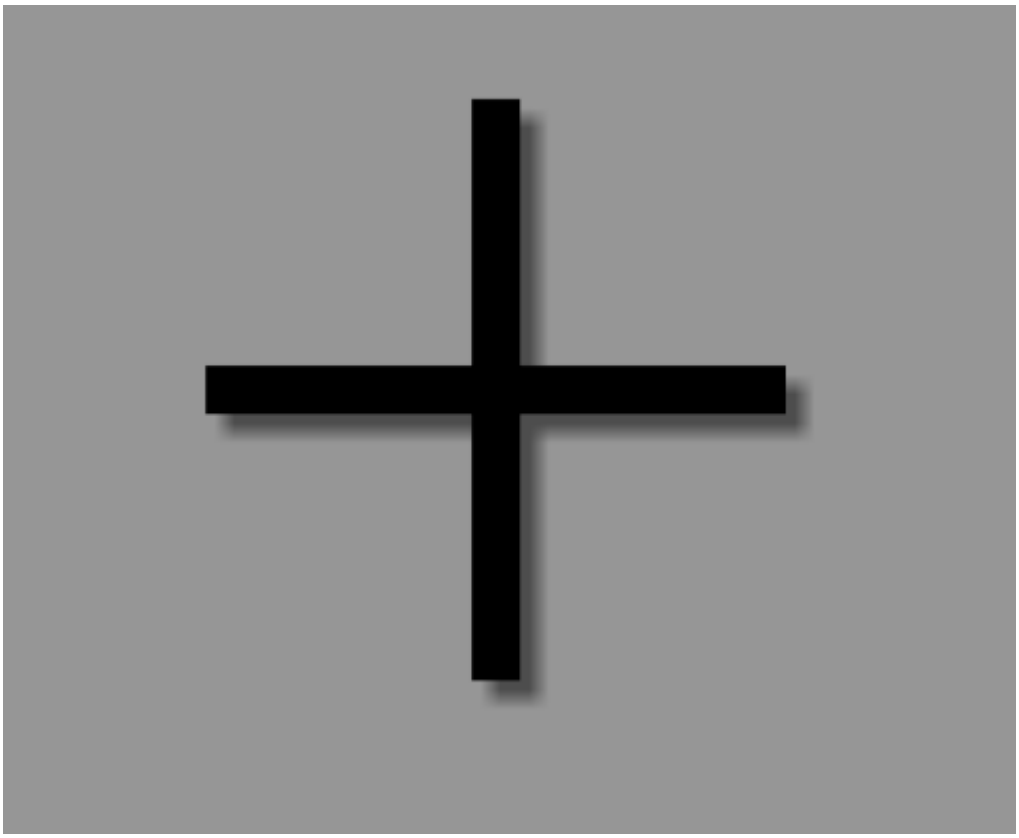
Ainsi, par exemple, vous créez un signe "plus" en utilisant des traits:

```
<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` définit la couleur du trait.
- `strokeWidth` définit la largeur (en dp) du trait (2dp dans ce cas, comme suggéré par les directives).
- `pathData` est l'endroit où nous décrivons notre image SVG:
- `M12,0` déplace le "curseur" sur la position 12,0
- `V24` crée une ligne verticale à la position 12, 24

etc., consultez la documentation SVG et ce tutoriel utile "Chemin SVG" de w3schools pour en savoir plus sur les commandes de chemin spécifiques.

En conséquence, nous avons eu ce signe plus simple:



Ceci est **particulièrement utile** pour créer un `AnimatedVectorDrawable` , puisque vous utilisez maintenant un seul trait avec une longueur unifiée, au lieu d'un chemin compliqué.

Compatibilité vectorielle via AppCompat

Quelques pré-requis dans `build.gradle` pour que les vecteurs fonctionnent jusqu'à API 7 pour VectorDrawables et API 13 pour AnimatedVectorDrawables (avec quelques réserves actuellement):

```
//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'
}
```

Dans votre `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" />
```

Lire VectorDrawable et AnimatedVectorDrawable en ligne:
https://riptutorial.com/fr/android/topic/1627/vectordrawable-et-animatedvectordrawable

Chapitre 252: Vérifier la connexion de données

Examples

Vérifier la connexion de données

Cette méthode consiste à vérifier la connexion de données en exécutant une commande ping sur certains IP ou noms de domaine.

```
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;
}
```

Vérifier la connexion à l'aide de ConnectivityManager

```
    public static boolean isConnectedNetwork (Context context) {

        ConnectivityManager cm = (ConnectivityManager)
context.getSystemService(Context.CONNECTIVITY_SERVICE);
        return cm.getActiveNetworkInfo () != null && cm.getActiveNetworkInfo
().isConnectedOrConnecting ();

    }
```

Utiliser les intentions du réseau pour effectuer des tâches tant que les données sont autorisées

Lorsque votre appareil se connecte à un réseau, une intention est envoyée. De nombreuses applications ne vérifient pas ces intentions, mais pour que votre application fonctionne correctement, vous pouvez écouter les modifications de réseau qui vous indiqueront quand la communication est possible. Pour vérifier la connectivité réseau, vous pouvez, par exemple, utiliser la clause suivante:

```
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
```

```
}
```

Lire Vérifier la connexion de données en ligne: https://riptutorial.com/fr/android/topic/8670/verifier-la-connexion-de-donnees

Chapitre 253: Vérifiez la connectivité Internet

Introduction

Cette méthode est utilisée pour vérifier si la connexion Wi-Fi est connectée ou non.

Syntaxe

- isNetworkAvailable (): pour vérifier si Internet est disponible sur le périphérique

Paramètres

Paramètre	Détail
Le contexte	Une référence du contexte d'activité

Remarques

Si internet connecté alors la méthode retournera vrai ou faux.

Examples

Vérifiez si l'appareil est connecté à Internet

Ajoutez les autorisations réseau requises au fichier manifeste de l'application:

```
<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;
}
```

Comment vérifier la force du réseau dans 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
            }
        }
```

Comment vérifier la force du réseau

Pour vérifier la force exacte en décibels, utilisez ceci-

```
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
        }
    }
```

Pour vérifier le type de réseau, utilisez cette classe.

```
 public class Connectivity {
        /*
         * These constants aren't yet available in my API level (7), but I need to
         * handle these cases if they come up, on newer versions
         */
        public static final int NETWORK_TYPE_EHRPD = 14; // Level 11
        public static final int NETWORK_TYPE_EVDO_B = 12; // Level 9
        public static final int NETWORK_TYPE_HSPAP = 15; // Level 13
        public static final int NETWORK_TYPE_IDEN = 11; // Level 8
        public static final int NETWORK_TYPE_LTE = 13; // Level 11

    /**
     * Check if there is any connectivity
     *
     * @param context
     * @return
     */
    public static boolean isConnected(Context context) {
        ConnectivityManager cm = (ConnectivityManager) context
                .getSystemService(Context.CONNECTIVITY_SERVICE);
        NetworkInfo info = cm.getActiveNetworkInfo();
        return (info != null && info.isConnected());
    }

    /**
     * Check if there is fast connectivity
     *
     * @param context
     * @return
     */
    public static String isConnectedFast(Context context) {
        ConnectivityManager cm = (ConnectivityManager) context
                .getSystemService(Context.CONNECTIVITY_SERVICE);
        NetworkInfo info = cm.getActiveNetworkInfo();

        if ((info != null && info.isConnected())) {
            return Connectivity.isConnectionFast(info.getType(),
                    info.getSubtype());
        } else
            return "No NetWork Access";

    }
```

```
/**
 * Check if the connection is fast
 *
 * @param type
 * @param subType
 * @return
 */
public static String isConnectionFast(int type, int subType) {
    if (type == ConnectivityManager.TYPE_WIFI) {
        System.out.println("CONNECTED VIA WIFI");
        return "CONNECTED VIA WIFI";
    } else if (type == ConnectivityManager.TYPE_MOBILE) {
        switch (subType) {
        case TelephonyManager.NETWORK_TYPE_1xRTT:
            return "NETWORK TYPE 1xRTT"; // ~ 50-100 kbps
        case TelephonyManager.NETWORK_TYPE_CDMA:
            return "NETWORK TYPE CDMA (3G) Speed: 2 Mbps"; // ~ 14-64 kbps
        case TelephonyManager.NETWORK_TYPE_EDGE:

            return "NETWORK TYPE EDGE (2.75G) Speed: 100-120 Kbps"; // ~
                                                                    // 50-100
                                                                    // kbps
        case TelephonyManager.NETWORK_TYPE_EVDO_0:
            return "NETWORK TYPE EVDO_0"; // ~ 400-1000 kbps
        case TelephonyManager.NETWORK_TYPE_EVDO_A:
            return "NETWORK TYPE EVDO_A"; // ~ 600-1400 kbps
        case TelephonyManager.NETWORK_TYPE_GPRS:
            return "NETWORK TYPE GPRS (2.5G) Speed: 40-50 Kbps"; // ~ 100
                                                                 // kbps
        case TelephonyManager.NETWORK_TYPE_HSDPA:
            return "NETWORK TYPE HSDPA (4G) Speed: 2-14 Mbps"; // ~ 2-14
                                                               // Mbps
        case TelephonyManager.NETWORK_TYPE_HSPA:
            return "NETWORK TYPE HSPA (4G) Speed: 0.7-1.7 Mbps"; // ~
                                                                 // 700-1700
                                                                 // kbps
        case TelephonyManager.NETWORK_TYPE_HSUPA:
            return "NETWORK TYPE HSUPA (3G) Speed: 1-23 Mbps"; // ~ 1-23
                                                               // Mbps
        case TelephonyManager.NETWORK_TYPE_UMTS:
            return "NETWORK TYPE UMTS (3G) Speed: 0.4-7 Mbps"; // ~ 400-7000
                                                               // kbps
            // NOT AVAILABLE YET IN API LEVEL 7
        case Connectivity.NETWORK_TYPE_EHRPD:
            return "NETWORK TYPE EHRPD"; // ~ 1-2 Mbps
        case Connectivity.NETWORK_TYPE_EVDO_B:
            return "NETWORK_TYPE_EVDO_B"; // ~ 5 Mbps
        case Connectivity.NETWORK_TYPE_HSPAP:
            return "NETWORK TYPE HSPA+ (4G) Speed: 10-20 Mbps"; // ~ 10-20
                                                                // Mbps
        case Connectivity.NETWORK_TYPE_IDEN:
            return "NETWORK TYPE IDEN"; // ~25 kbps
        case Connectivity.NETWORK_TYPE_LTE:
            return "NETWORK TYPE LTE (4G) Speed: 10+ Mbps"; // ~ 10+ Mbps
            // Unknown
        case TelephonyManager.NETWORK_TYPE_UNKNOWN:
            return "NETWORK TYPE UNKNOWN";
        default:
            return "";
        }
```

```
        } else {
            return "";
        }
    }

}
```

Lire Vérifiez la connectivité Internet en ligne: https://riptutorial.com/fr/android/topic/3918/verifiez-la-connectivite-internet

Chapitre 254: Versions Android

Remarques

prénom	Version Android	Date de sortie	Niveau API	Build.VERSION_CODES
Angel Cake (Alpha)	1.0	23 septembre 2008	1	BASE
Battenberg (Beta)	1.1	9 février 2009	2	BASE_1_1
Petit gâteau	1,5	30 avril 2009	3	CUPCAKE
Beignet	1.6	15 septembre 2009	4	BEIGNET
Eclair	2.0	26 octobre 2009	5	ECLAIR
	2.0.1	3 décembre 2009	6	ECLAIR_0_1
	2.1	12 janvier 2010	7	ECLAIR_MR1
Froyo	2.2	20 mai 2010	8	FROYO
pain d'épice	2.3	6 décembre 2010	9	PAIN D'ÉPICE
	2.3.3	9 février 2011	dix	GINGERBREAD_MR1
Rayon de miel	3.0	22 février 2011	11	RAYON DE MIEL
	3.1	10 mai 2011	12	HONEYCOMB_MR2
	3.2	15 juillet 2011	13	HONEYCOMB_MR1
Sandwich à la crème glacée	4.0	19 octobre 2011	14	SANDWICH À LA CRÈME GLACÉE

prénom	Version Android	Date de sortie	Niveau API	Build.VERSION_CODES
	4.0.3	16 décembre 2011	15	ICE_CREAM_SANDWICH_MR1
Dragée	4.1	9 juillet 2012	16	DRAGÉE
	4.2	13 novembre 2012	17	JELLY_BEAN_MR1
	4.3	24 juillet 2013	18	JELLY_BEAN_MR2
KitKat	4.4	31 octobre 2013	19	KITKAT
		25 juillet 2014	20	KITKAT_WATCH
Sucette	5.0	17 octobre 2014	21	SUCETTE
	5.1	9 mars 2015	22	LOLLIPOP_MR1
Guimauve	6,0	5 octobre 2015	23	M
Nougat	7.0	22 août 2016	24	N
	7.1.1	5 décembre 2016	25	N_MR1

Examples

Vérification de la version Android sur l'appareil au moment de l'exécution

`Build.VERSION_CODES` est une énumération des codes de version du SDK actuellement connus.

Afin d'exécuter le code de manière conditionnelle en fonction de la version Android du périphérique, utilisez l'annotation `TargetApi` pour éviter les erreurs de Lint et vérifiez la version de génération avant d'exécuter le code spécifique au niveau de l'API.

Voici un exemple d'utilisation d'une classe introduite dans API-23, dans un projet qui prend en charge des niveaux d'API inférieurs à 23:

```
@Override
@TargetApi(23)
public void onResume() {
```

```
    super.onResume();
    if (android.os.Build.VERSION.SDK_INT <= Build.VERSION_CODES.M) {
        //run Marshmallow code
        FingerprintManager fingerprintManager =
this.getSystemService(FingerprintManager.class);
        //.....................
    }
}
```

Lire Versions Android en ligne: https://riptutorial.com/fr/android/topic/3264/versions-android

Chapitre 255: Versions du SDK de projet

Introduction

Une application Android doit s'exécuter sur tous les types d'appareils. Chaque appareil peut avoir une version différente sur Android fonctionnant dessus.

Désormais, chaque version d'Android peut ne pas prendre en charge toutes les fonctionnalités requises par votre application. Ainsi, lors de la création d'une application, vous devez garder à l'esprit la version minimale et maximale d'Android.

Paramètres

Paramètre	Détails
Version du SDK	La version SDK de chaque champ est le nombre entier de niveau API SDK de la version Android. Par exemple, Froyo (Android 2.2) correspond au niveau 8 de l'API. Ces entiers sont également définis dans `Build.VERSION_CODES` .

Remarques

Il existe quatre versions de SDK pertinentes dans chaque projet:

- `targetSdkVersion` est la dernière version d'Android que vous avez testée.

  Le framework utilisera `targetSdkVersion` pour déterminer quand activer certains comportements de compatibilité. Par exemple, le niveau 23 ou supérieur de l'API de ciblage vous permet d' accéder au modèle d'autorisations d'exécution .

- `minSdkVersion` est la version minimale d'Android prise en charge par votre application. Les utilisateurs exécutant une version d'Android antérieure à cette version ne pourront pas installer votre application ou la voir dans le Play Store.

- `maxSdkVersion` est la version maximale d'Android prise en charge par votre application. Les utilisateurs exécutant une version d'Android plus récente que cette version ne pourront pas installer votre application ou la voir dans le Play Store. Cela ne devrait généralement pas être utilisé car la plupart des applications fonctionneront sur les nouvelles versions d'Android sans aucun effort supplémentaire.

- `compileSdkVersion` est la version du SDK Android avec lequel votre application sera compilée. Il devrait généralement s'agir de la dernière version d'Android publiée. Ceci définit les API auxquelles vous pouvez accéder lors de l'écriture de votre code. Vous ne pouvez pas appeler des méthodes introduites dans le niveau 23 de l'API si votre `compileSdkVersion` est défini sur 22 ou moins.

Examples

Définition des versions du SDK du projet

Dans votre fichier `build.gradle` du module principal (**app**), définissez votre numéro de version minimum et cible.

```
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
    }
}
```

Lire Versions du SDK de projet en ligne: https://riptutorial.com/fr/android/topic/162/versions-du-sdk-de-projet

Chapitre 256: Vibration

Examples

Démarrer avec Vibration

Accorder une autorisation de vibration

avant de commencer le code de l'outil, vous devez ajouter une autorisation dans le manifeste Android:

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

Importer une bibliothèque de vibrations

```
import android.os.Vibrator;
```

Obtenir une instance de vibrateur à partir du contexte

```
Vibrator vibrator = (Vibrator) getSystemService(Context.VIBRATOR_SERVICE);
```

Vérifier le dispositif a vibrateur

```
void boolean isHaveVibrate(){
    if (vibrator.hasVibrator()) {
        return true;
    }
    return false;
}
```

Vibrer indéfiniment

en utilisant le *vibreur (modèle long [], répétition int)*

```
Vibrator vibrator = (Vibrator) getSystemService(Context.VIBRATOR_SERVICE);

// Start time delay
// Vibrate for 500 milliseconds
// Sleep for 1000 milliseconds
long[] pattern = {0, 500, 1000};

// 0 meaning is repeat indefinitely
vibrator.vibrate(pattern, 0);
```

Motifs de vibration

Vous pouvez créer des motifs de vibration en passant un tableau de longs, chacun représentant une durée en millisecondes. Le premier numéro est la temporisation de début. Chaque entrée de

tableau alterne alors entre vibrer, dormir, vibrer, dormir, etc.

L'exemple suivant illustre ce modèle:

- vibrer 100 millisecondes et dormir 1000 millisecondes
- vibrer 200 millisecondes et dormir 2000 millisecondes

```
long[] pattern = {0, 100, 1000, 200, 2000};
```

Pour que le motif se répète, indiquez l'index dans le tableau de modèles auquel lancer la répétition ou `-1` pour désactiver la répétition.

```
Vibrator vibrator = (Vibrator) getSystemService(Context.VIBRATOR_SERVICE);
vibrator.vibrate(pattern, -1); // does not repeat
vibrator.vibrate(pattern,  0); // repeats forever
```

Arrêter Vibrer

Si vous voulez arrêter de vibrer, veuillez appeler:

```
vibrator.cancel();
```

Vibrer pendant une fois

en utilisant le *vibreur (longues millisecondes)*

```
Vibrator vibrator = (Vibrator) getSystemService(Context.VIBRATOR_SERVICE);
vibrator.vibrate(500);
```

Lire Vibration en ligne: https://riptutorial.com/fr/android/topic/3359/vibration

Chapitre 257: VideoView

Examples

VideoView Créer

Recherchez VideoView in Activity et ajoutez-y une vidéo.

```
VideoView videoView = (VideoView) .findViewById(R.id.videoView);
videoView.setVideoPath(pathToVideo);
```

Commencez à lire la vidéo.

```
videoView.start();
```

Définissez VideoView dans le fichier de disposition XML.

```
<VideoView
    android:id="@+id/videoView"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_gravity="center" />
```

Lire une vidéo à partir d'une URL à l'aide de VideoView

```
 videoView.setVideoURI(Uri.parse("http://example.com/examplevideo.mp4"));
 videoView.requestFocus();

 videoView.setOnCompletionListener(new MediaPlayer.OnCompletionListener() {
      @Override
      public void onCompletion(MediaPlayer mediaPlayer) {
      }
 });

 videoView.setOnPreparedListener(new MediaPlayer.OnPreparedListener() {
     @Override
     public void onPrepared(MediaPlayer mediaPlayer) {
          videoView.start();
          mediaPlayer.setOnVideoSizeChangedListener(new
MediaPlayer.OnVideoSizeChangedListener() {
                @Override
                public void onVideoSizeChanged(MediaPlayer mp, int width, int height) {
                      MediaController mediaController = new
MediaController(ActivityName.this);
                      videoView.setMediaController(mediaController);
                      mediaController.setAnchorView(videoView);
                }
          });
     }
 });

 videoView.setOnErrorListener(new MediaPlayer.OnErrorListener() {
```

```
        @Override
        public boolean onError(MediaPlayer mediaPlayer, int i, int i1) {
             return false;
        }
});
```

Lire VideoView en ligne: https://riptutorial.com/fr/android/topic/8962/videoview

Chapitre 258: VideoView optimisé

Introduction

La lecture d'une vidéo à l'aide d'un `VideoView` qui étend `SurfaceView` intérieur d'une ligne de `ListView` semble fonctionner en premier, jusqu'à ce que l'utilisateur essaie de faire défiler la liste. Dès que la liste commence à défiler, la vidéo devient noire (affiche parfois en blanc). Il continue de jouer en arrière-plan mais vous ne pouvez plus le voir car il affiche le reste de la vidéo sous forme de boîte noire. Avec le VideoView optimisé personnalisé, les vidéos seront lues sur Scroll dans le `ListView` tout comme notre Instagram, Facebook et Twitter.

Examples

VideoView optimisée dans ListView

C'est la `VideoView` personnalisée dont vous avez besoin dans votre paquet.

Disposition de VideoView personnalisée:

```
<your.packagename.VideoView
    android:id="@+id/video_view"
    android:layout_width="300dp"
    android:layout_height="300dp" />
```

Code pour la `VideoView` optimisée `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);
    }
}
```

Aide de ce dépôt gitub . Bien qu'il y ait quelques problèmes comme il a été écrit il y a 3 ans, j'ai réussi à les réparer par moi-même, comme indiqué ci-dessus.

Lire VideoView optimisé en ligne: https://riptutorial.com/fr/android/topic/10638/videoview-optimise

Chapitre 259: ViewFlipper

Introduction

Un `ViewFlipper` est un `ViewAnimator` qui permet de basculer entre plusieurs vues ajoutées. Un seul enfant est montré à la fois. Si `ViewFlipper` , le `ViewFlipper` peut automatiquement basculer entre chaque enfant à intervalles réguliers.

Examples

ViewFlipper avec image glissant

Fichier XML:

```
<ViewFlipper
    android:id="@+id/viewflip"
    android:layout_width="match_parent"
    android:layout_height="250dp"
    android:layout_weight="1"
    />
```

Code JAVA:

```
public class BlankFragment extends Fragment{
    ViewFlipper viewFlipper;
    FragmentManager fragmentManager;
    int gallery_grid_Images[] = {drawable.image1, drawable.image2, drawable.image3,
            drawable.image1, drawable.image2, drawable.image3, drawable.image1,
            drawable.image2, drawable.image3, drawable.image1
    };

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState){
        View rootView = inflater.inflate(fragment_blank, container, false);
        viewFlipper = (ViewFlipper)rootView.findViewById(R.id.viewflip);
        for(int i=0; i<gallery_grid_Images.length; i++){
            // This will create dynamic image views and add them to the ViewFlipper.
            setFlipperImage(gallery_grid_Images[i]);
        }
        return rootView;
    }

    private void setFlipperImage(int res) {
        Log.i("Set Filpper Called", res+"");
        ImageView image = new ImageView(getContext());
        image.setBackgroundResource(res);
        viewFlipper.addView(image);
        viewFlipper.setFlipInterval(1000);
        viewFlipper.setAutoStart(true);
    }
}
```

Lire ViewFlipper en ligne: https://riptutorial.com/fr/android/topic/9032/viewflipper

Chapitre 260: ViewPager

Introduction

ViewPager est un gestionnaire de mise en page qui permet à l'utilisateur de parcourir les pages de données à gauche et à droite. Il est le plus souvent utilisé conjointement avec Fragment, qui est un moyen pratique de fournir et de gérer le cycle de vie de chaque page.

Remarques

Une chose importante à noter à propos de l'utilisation de ViewPager est qu'il existe deux versions différentes de `FragmentPagerAdapter` et de `FragmentStatePagerAdapter` .

Si vous utilisez des fragments natifs `android.app.Fragment` avec un FragmentPagerAdapter ou un FragmentStatePagerAdapter, vous devez utiliser les versions de la bibliothèque de support v13 de l'adaptateur, à savoir `android.support.v13.app.FragmentStatePagerAdapter` .

Si vous utilisez la bibliothèque de support `android.support.v4.app.Fragment` avec un fragment FragmentPagerAdapter ou FragmentStatePagerAdapter, vous devez utiliser les versions de bibliothèque de support v4 de l'adaptateur, à savoir `android.support.v4.app.FragmentStatePagerAdapter` .

Examples

Utilisation de base de ViewPager avec des fragments

Un `ViewPager` permet d'afficher plusieurs fragments dans une activité qui peut être parcourue en basculant à gauche ou à droite. Un `ViewPager` doit être alimenté par Views ou Fragments à l'aide d'un `PagerAdapter` .

Il existe cependant deux implémentations plus spécifiques que vous trouverez plus utiles dans le cas de l'utilisation de Fragments, à savoir `FragmentPagerAdapter` et `FragmentStatePagerAdapter` . Lorsqu'un fragment doit être instancié pour la première fois, `getItem(position)` sera appelé pour chaque position nécessitant une instanciation. La méthode `getCount()` renvoie le nombre total de pages afin que `ViewPager` sache combien de fragments doivent être affichés.

`FragmentPagerAdapter` et `FragmentStatePagerAdapter` conservent tous deux un cache des fragments que `ViewPager` devra afficher. Par défaut, `ViewPager` essaiera de stocker un maximum de 3 fragments correspondant au fragment actuellement visible, et ceux situés à droite et à gauche. Aussi `FragmentStatePagerAdapter` gardera l'état de chacun de vos fragments.

Sachez que les deux implémentations supposent que vos fragments conserveront leurs positions, donc si vous conservez une liste des fragments au lieu d'en avoir un nombre statique comme vous pouvez le voir dans la méthode `getItem()` , vous devrez créer une sous-classe de `PagerAdapter` et remplacer au moins `instantiateItem()` , `destroyItem()` et `getItemPosition()`

méthodes.

Ajoutez simplement un ViewPager dans votre mise en page comme décrit dans l' exemple de base :

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout>
    <android.support.v4.view.ViewPager
        android:id="@+id/vpPager">
    </android.support.v4.view.ViewPager>
</LinearLayout>
```

Définissez ensuite l'adaptateur qui déterminera le nombre de pages existantes et le fragment à afficher pour chaque page de l'adaptateur.

```
public class MyViewPagerActivity extends AppCompatActivity {
    private static final String TAG = MyViewPagerActivity.class.getName();

    private MyPagerAdapter mFragmentAdapter;
    private ViewPager mViewPager;

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

        //Apply the Adapter
        mFragmentAdapter = new MyPagerAdapter(getSupportFragmentManager());
        mViewPager = (ViewPager) findViewById(R.id.view_pager);
        mViewPager.setAdapter(mFragmentAdapter);
    }

    private class MyPagerAdapter extends FragmentPagerAdapter{

        public MyPagerAdapter(FragmentManager supportFragmentManager) {
            super(supportFragmentManager);
        }

        // Returns the fragment to display for that page
        @Override
        public Fragment getItem(int position) {
            switch(position) {
                case 0:
                    return new Fragment1();

                case 1:
                    return new Fragment2();

                case 2:
                    return new Fragment3();

                default:
                    return null;
            }
        }

        // Returns total number of pages
        @Override
```

```
        public int getCount() {
            return 3;
        }

    }
}
```

3.2.x

Si vous utilisez `android.app.Fragment` vous devez ajouter cette dépendance:

```
compile 'com.android.support:support-v13:25.3.1'
```

Si vous utilisez `android.support.v4.app.Fragment` vous devez ajouter cette dépendance:

```
compile 'com.android.support:support-fragment:25.3.1'
```

ViewPager avec TabLayout

Un `TabLayout` peut être utilisé pour faciliter la navigation.
Vous pouvez définir les onglets pour chaque fragment de votre adaptateur à l'aide de la méthode `TabLayout.newTab()` , mais il existe une autre méthode plus pratique et plus simple pour cette tâche: `TabLayout.setupWithViewPager()` .

Cette méthode se synchronise en créant et en supprimant des onglets en fonction du contenu de l'adaptateur associé à votre `ViewPager` chaque fois que vous l'appelez.
En outre, il va définir un rappel chaque fois que l'utilisateur retourne la page, l'onglet correspondant sera sélectionné.

Il suffit de définir une mise en page

```
<?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>
```

Ensuite, implémentez `FragmentPagerAdapter` et appliquez-le à `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 avec PréférenceFragment

Jusqu'à récemment, l'utilisation de `android.support.v4.app.FragmentPagerAdapter` empêchait l'utilisation de `PreferenceFragment` comme l'un des fragments utilisés dans FragmentPagerAdapter.

Les dernières versions de la bibliothèque de support v7 incluent désormais la classe `PreferenceFragmentCompat` , qui fonctionnera avec un ViewPager et la version v4 de FragmentPagerAdapter.

Exemple de fragment qui étend `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) {

    }
}
```

Vous pouvez maintenant utiliser ce fragment dans une sous-classe `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;
        }
    }

    // .......

}
```

Ajouter un ViewPager

Assurez-vous que la dépendance suivante est ajoutée au fichier `build.gradle` votre application sous les dépendances:

```
compile 'com.android.support:support-core-ui:25.3.0'
```

Ajoutez ensuite le `ViewPager` à votre disposition d'activité:

```
<android.support.v4.view.ViewPager
    android:id="@+id/viewpager"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    />
```

Définissez ensuite votre `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;
    }
}
```

Enfin, configurez le `ViewPager` dans votre activité:

```
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 avec un indicateur de points

Tout ce dont nous avons besoin sont: ViewPager , TabLayout et 2 tirables pour les points sélectionnés et par défaut.

Tout d'abord, nous devons ajouter `TabLayout` à notre disposition d'écran et le connecter à `ViewPager`

. Nous pouvons le faire de deux manières:

Onglet imbriquéMise en forme dans 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>
```

> Dans ce cas, `TabLayout` sera automatiquement connecté à `ViewPager` , mais `TabLayout` sera à côté de `ViewPager` , pas sur lui.

TabLayout séparé

```
<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"/>
```

> Dans ce cas, nous pouvons mettre `TabLayout` n'importe où, mais nous devons connecter `TabLayout` avec `ViewPager` programmation

```
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);
```

Une fois que nous avons créé notre mise en page, nous devons préparer nos points. Nous créons donc trois fichiers: `selected_dot.xml` , `default_dot.xml` et `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>
```

Maintenant, nous devons ajouter seulement 3 lignes de code à `TabLayout` dans notre présentation XML et vous avez terminé.

```
app:tabBackground="@drawable/tab_selector"
app:tabGravity="center"
app:tabIndicatorHeight="0dp"
```

Configuration de OnPageChangeListener

Si vous devez écouter les modifications apportées à la page sélectionnée, vous pouvez implémenter l'écouteur `ViewPager.OnPageChangeListener` sur ViewPager:

```
viewPager.addOnPageChangeListener(new OnPageChangeListener() {
```

```
    // This method will be invoked when a new page becomes selected. Animation is not
necessarily complete.
    @Override
    public void onPageSelected(int position) {
         // Your code
    }

    // This method will be invoked when the current page is scrolled, either as part of
    // a programmatically initiated smooth scroll or a user initiated touch scroll.
    @Override
    public void onPageScrolled(int position, float positionOffset, int positionOffsetPixels) {
         // Your code
    }

    // Called when the scroll state changes. Useful for discovering when the user begins
    // dragging, when the pager is automatically settling to the current page,
    // or when it is fully stopped/idle.
    @Override
    public void onPageScrollStateChanged(int state) {
         // Your code
    }
});
```

Lire ViewPager en ligne: https://riptutorial.com/fr/android/topic/692/viewpager

Chapitre 261: voie rapide

Remarques

fastlane est un outil pour les développeurs iOS, Mac et Android qui permet d'automatiser les tâches fastidieuses telles que la génération de captures d'écran, le traitement des profils d'approvisionnement et la publication de votre application.

Docs: https://docs.fastlane.tools/

Code source: https://github.com/fastlane/fastlane

Examples

Fastfile pour créer et télécharger plusieurs versions de Beta par Crashlytics

Ceci est un exemple de configuration de **Fastfile** pour une application multi-saveur. Cela vous donne la possibilité de créer et de déployer toutes les saveurs ou une seule saveur. Après le déploiement, il signale à **Slack** le statut du déploiement et envoie une notification aux testeurs de la version bêta du groupe de testeurs Crashlytics.

Pour créer et déployer toutes les saveurs, utilisez:

```
fastlane android beta
```

Pour créer un seul APK et déployer, utilisez:

```
fastlane android beta app:flavorName
```

En utilisant un seul fichier Fastlane, vous pouvez gérer les applications iOS, Android et Mac. Si vous utilisez ce fichier, une seule `platform` - `platform` applications n'est pas requise.

Comment ça marche

1. `android` argument `android` dit fastlane que nous allons utiliser `:android` plateforme `:android` .
2. À l'intérieur `:android` plateforme `:android` , vous pouvez avoir plusieurs voies. Actuellement, je n'ai que `:beta` lane. Le second argument de la commande ci-dessus spécifie la voie que nous voulons utiliser.
3. `options[:app]`
4. Il y a deux tâches **Gradle** . Tout d'abord, il fonctionne `gradle clean` . Si vous avez fourni une saveur avec une clé d' `app` , fastfile exécute `gradle assembleReleaseFlavor` . Sinon, il exécute `gradle assembleRelease` pour générer toutes les `gradle assembleRelease` de build.
5. Si nous construisons pour tous les goûts, un tableau de noms de fichiers APK générés est stocké dans `SharedValues::GRADLE_ALL_APK_OUTPUT_PATHS` . Nous l'utilisons pour parcourir les fichiers générés et les déployer sur **Beta par Crashlytics** . `notifications` champs de

`notifications` et de `groups` sont facultatifs. Ils sont utilisés pour informer les testeurs enregistrés pour l'application sur la **version bêta de Crashlytics** .

6. Si vous connaissez Crashlytics, vous savez peut-être que pour activer une application sur le portail, vous devez l'exécuter sur un périphérique et l'utiliser en premier. Sinon, Crashlytics assumera l'inactivité de l'application et générera une erreur. Dans ce scénario, je le capture et le signale à **Slack** comme un échec. Vous saurez ainsi quelle application est inactive.
7. Si le déploiement réussit, **fastlane** enverra un message de réussite à **Slack** .
8. `#{/([^\/]*)$/.match(apk)}` cette regex est utilisée pour obtenir le nom de saveur du chemin APK. Vous pouvez le supprimer si cela ne fonctionne pas pour vous.
9. `get_version_name` et `get_version_code` sont deux plugins **Fastlane** permettant de récupérer le nom et le code de la version de l'application. Vous devez installer ces gemmes si vous voulez utiliser, ou vous pouvez les supprimer. En savoir plus sur les plugins ici.
10. L'instruction `else` sera exécutée si vous construisez et déployez un seul APK. Nous n'avons pas à fournir `apk_path` à Crashlytics car nous n'avons qu'une seule application.
11. `error do` bloc à la fin est utilisé pour être averti si quelque chose ne va pas pendant l'exécution.

Remarque

N'oubliez pas de remplacer `SLACK_URL` , `API_TOKEN` , `GROUP_NAME` et `BUILD_SECRET` par vos propres informations d'identification.

```
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
```

Ligne Fastfile pour créer et installer toutes les versions d'un type de construction donné sur un périphérique

Ajoutez cette voie à votre **Fastfile** et exécutez `fastlane installAll type:{BUILD_TYPE}` en ligne de commande. Remplacez `BUILD_TYPE` par le type de construction que vous souhaitez générer.

Par exemple: `fastlane installAll type:Debug`

Cette commande va générer toutes les versions d'un type donné et l'installer sur votre appareil. Actuellement, cela ne fonctionne pas si vous avez plusieurs périphériques connectés. Assurez-vous que vous n'en avez qu'un. À l'avenir, je prévois d'ajouter une option pour sélectionner le périphérique cible.

```
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
```

Lire voie rapide en ligne: https://riptutorial.com/fr/android/topic/8215/voie-rapide

Chapitre 262: Voir Calculs Dimensions

Remarques

Notez qu'une instance de `ViewTreeObserver` associée à une instance de `View` peut devenir invalide tant que cette `View` est toujours active. A partir du `View.getViewTreeObserver` View.getViewTreeObserver:

```
// The returned ViewTreeObserver observer is not guaranteed to remain
// valid for the lifetime of this View. If the caller of this method keeps
// a long-lived reference to ViewTreeObserver, it should always check for
// the return value of {@link ViewTreeObserver#isAlive()}.
```

Ainsi, si vous avez précédemment ajouté un écouteur à une instance de `ViewTreeObserver` et souhaitez maintenant le supprimer, il est plus facile d'appeler à nouveau `getViewTreeObserver` sur l'instance `View` correspondante pour recevoir une nouvelle instance de `ViewTreeObserver` . (Vérifier `isAlive` sur une instance existante est plus de travail pour un petit bénéfice; si le `ViewTreeObserver` n'est plus en vie, vous allez quand même aller chercher cette nouvelle référence!)

Examples

Calcul des dimensions de vue initiales dans une activité

```
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);
    }

}
```

Lire Voir Calculs Dimensions en ligne: https://riptutorial.com/fr/android/topic/115/voir-calculs-dimensions

Chapitre 263: Volée

Introduction

Volley est une bibliothèque HTTP Android introduite par Google pour simplifier les appels réseau. Par défaut, tous les appels réseau Volley sont effectués de manière asynchrone, gérant tout ce qui se trouve dans un thread d'arrière-plan et renvoyant les résultats au premier plan en utilisant des rappels. Comme l'extraction de données sur un réseau est l'une des tâches les plus courantes effectuées dans n'importe quelle application, la bibliothèque Volley a été conçue pour faciliter le développement d'applications Android.

Syntaxe

- RequestQueue queue = Volley.newRequestQueue (context); // configure la file d'attente
- Request request = new SomeKindOfRequestClass (Request.Method, String url, Response.Listener, Response.ErrorListener); // configure une sorte de requête, le type exact et les arguments changent pour chaque type de requête
- queue.add (demande); // ajoute la requête à la file d'attente; l'auditeur de réponse approprié sera appelé une fois la demande terminée (ou terminée pour quelque raison que ce soit)

Remarques

Installation

Vous pouvez créer Volley à partir du code source officiel de Google . Pendant un moment, c'était la seule option. Ou en utilisant l'une des versions pré-construites tierces. Cependant, Google a finalement publié un paquet maven officiel sur jcenter.

Dans votre fichier `build.gradle` niveau `build.gradle` , ajoutez ceci à votre liste de dépendances:

```
dependencies {
    ...
    compile 'com.android.volley:volley:1.0.0'
}
```

Assurez-vous que l'autorisation `INTERNET` est définie dans le manifeste de votre application:

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

Documentation officielle

Google n'a pas fourni de documentation très complète sur cette bibliothèque et elle ne l'a pas

touchée depuis des années. Mais ce qui est disponible peut être trouvé à:

https://developer.android.com/training/volley/index.html

Il existe une documentation non officielle hébergée sur GitHub, mais il devrait y avoir un meilleur emplacement pour l'accueillir à l'avenir:

https://pablobaxter.github.io/volley-docs/

Examples

Basic StringRequest en utilisant la méthode GET

```
final TextView mTextView = (TextView) findViewById(R.id.text);
...

// Instantiate the RequestQueue.
RequestQueue queue = Volley.newRequestQueue(this);
String url ="http://www.google.com";

// Request a string response from the provided URL.
StringRequest stringRequest = new StringRequest(Request.Method.GET, url,
            new Response.Listener<String>() {
    @Override
    public void onResponse(String response) {
        // Display the first 500 characters of the response string.
        mTextView.setText("Response is: "+ response.substring(0,500));
    }
}, new Response.ErrorListener() {
    @Override
    public void onErrorResponse(VolleyError error) {
        mTextView.setText("That didn't work!");
    }
});
// Add the request to the RequestQueue.
queue.add(stringRequest);
```

Annuler une demande

```
// 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);
```

Ajout d'attributs de conception personnalisés à NetworkImageView

Il existe plusieurs attributs supplémentaires que Volley `NetworkImageView` ajoute à la version standard d' `ImageView` . Cependant, ces attributs ne peuvent être définis que dans le code. Voici un exemple de création d'une classe d'extension qui récupérera les attributs de votre fichier de disposition XML et les appliquera à l'instance `NetworkImageView` .

Dans votre répertoire `~/res/xml` , ajoutez un fichier nommé `attrx.xml` :

```
<resources>
    <declare-styleable name="MoreNetworkImageView">
        <attr name="defaultImageResId" format="reference"/>
        <attr name="errorImageResId" format="reference"/>
    </declare-styleable>
</resources>
```

Ajoutez un nouveau fichier de classe à votre projet:

```
package my.namespace;

import android.content.Context;
import android.content.res.TypedArray;
import android.support.annotation.NonNull;
import android.util.AttributeSet;

import com.android.volley.toolbox.NetworkImageView;

public class MoreNetworkImageView extends NetworkImageView {
    public MoreNetworkImageView(@NonNull final Context context) {
        super(context);
    }

    public MoreNetworkImageView(@NonNull final Context context, @NonNull final AttributeSet
attrs) {
        this(context, attrs, 0);
    }

    public MoreNetworkImageView(@NonNull final Context context, @NonNull final AttributeSet
attrs, final int defStyle) {
        super(context, attrs, defStyle);

        final TypedArray attributes = context.obtainStyledAttributes(attrs,
R.styleable.MoreNetworkImageView, defStyle, 0);

        // load defaultImageResId from XML
        int defaultImageResId =
attributes.getResourceId(R.styleable.MoreNetworkImageView_defaultImageResId, 0);
        if (defaultImageResId > 0) {
            setDefaultImageResId(defaultImageResId);
        }

        // load errorImageResId from XML
        int errorImageResId =
attributes.getResourceId(R.styleable.MoreNetworkImageView_errorImageResId, 0);
        if (errorImageResId > 0) {
            setErrorImageResId(errorImageResId);
        }
```

```
    }
}
```

Un exemple de fichier de mise en page montrant l'utilisation des attributs personnalisés:

```
<?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>
```

Demander JSON

```
final TextView mTxtDisplay = (TextView) findViewById(R.id.txtDisplay);
ImageView mImageView;
String url = "http://ip.jsontest.com/";

final JsonObjectRequest jsObjRequest = new JsonObjectRequest
        (Request.Method.GET, url, null, new Response.Listener<JSONObject>() {
    @Override
    public void onResponse(JSONObject response) {
        mTxtDisplay.setText("Response: " + response.toString());
    }
}, new Response.ErrorListener() {
    @Override
    public void onErrorResponse(VolleyError error) {
        // ...
    }
});

requestQueue.add(jsObjRequest);
```

Ajouter des en-têtes personnalisés à vos requêtes [par exemple pour l'authentification de base]

Si vous devez ajouter des en-têtes personnalisés à vos demandes de volley, vous ne pouvez pas le faire après l'initialisation, car les en-têtes sont enregistrés dans une variable privée.

Au lieu de cela, vous devez remplacer la méthode `getHeaders()` de `Request.class` tant que telle:

```
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;
    }
};
```

Explication des paramètres:

- `REQUEST_METHOD` - L'une des constantes `Request.Method.*` .
- `REQUEST_URL` - L'URL complète à laquelle envoyer votre demande.
- `REQUEST_BODY` - Un objet `JSONObject` contenant le POST-Body à envoyer (ou null).
- `RESP_LISTENER` - Un objet `Response.Listener<?>` , Dont la `onResponse(T data)` est appelée après son achèvement.
- `ERR_LISTENER` - Objet `Response.ErrorListener` dont la `onErrorResponse(VolleyError e)` est appelée sur une requête ayant échoué.

Si vous souhaitez créer une demande personnalisée, vous pouvez également y ajouter les en-têtes suivants:

```
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;
    }
    ...
}
```

Classe d'assistance pour gérer les erreurs de volley

```
public class VolleyErrorHelper {
        /**
         * Returns appropriate message which is to be displayed to the user
         * against the specified error object.
         *
         * @param error
         * @param context
         * @return
```

```
         */

        public static String getMessage (Object error , Context context){
            if(error instanceof TimeoutError){
                return context.getResources().getString(R.string.timeout);
            }else if (isServerProblem(error)){
                return handleServerError(error ,context);

            }else if(isNetworkProblem(error)){
                return context.getResources().getString(R.string.nointernet);
            }
            return context.getResources().getString(R.string.generic_error);

        }

        private static String handleServerError(Object error, Context context) {

            VolleyError er = (VolleyError)error;
            NetworkResponse response = er.networkResponse;
            if(response != null){
                switch (response.statusCode){

                    case 404:
                    case 422:
                    case 401:
                        try {
                            // server might return error like this { "error": "Some error
occured" }
                            // Use "Gson" to parse the result
                            HashMap<String, String> result = new Gson().fromJson(new
String(response.data),
                                    new TypeToken<Map<String, String>>() {
                                    }.getType());

                            if (result != null && result.containsKey("error")) {
                                return result.get("error");
                            }

                        } catch (Exception e) {
                            e.printStackTrace();
                        }
                        // invalid request
                        return ((VolleyError) error).getMessage();

                    default:
                        return context.getResources().getString(R.string.timeout);
                }
            }

            return context.getResources().getString(R.string.generic_error);
        }

        private static boolean isServerProblem(Object error) {
            return (error instanceof ServerError || error instanceof AuthFailureError);
        }

        private static boolean isNetworkProblem (Object error){
            return (error instanceof NetworkError || error instanceof NoConnectionError);
        }
```

Authentification du serveur distant à l'aide de StringRequest via la méthode POST

Pour cet exemple, supposons que nous ayons un serveur pour traiter les requêtes POST que nous allons faire depuis notre application Android:

```
// User input data.
String email = "my@email.com";
String password = "123";

// Our server URL for handling POST requests.
String URL = "http://my.server.com/login.php";

// When we create a StringRequest (or a JSONRequest) for sending
// data with Volley, we specify the Request Method as POST, and
// the URL that will be receiving our data.
StringRequest stringRequest =
    new StringRequest(Request.Method.POST, URL,
    new Response.Listener<String>() {
        @Override
        public void onResponse(String response) {
            // At this point, Volley has sent the data to your URL
            // and has a response back from it. I'm going to assume
            // that the server sends an "OK" string.
            if (response.equals("OK")) {
                // Do login stuff.
            } else {
                // So the server didn't return an "OK" response.
                // Depending on what you did to handle errors on your
                // server, you can decide what action to take here.
            }
        }
    },
    new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {
            // This is when errors related to Volley happen.
            // It's up to you what to do if that should happen, but
            // it's usually not a good idea to be too clear as to
            // what happened here to your users.
        }
    }) {
        @Override
        protected Map<String, String> getParams() throws AuthFailureError {
            // Here is where we tell Volley what it should send in
            // our POST request. For this example, we want to send
            // both the email and the password.

            // We will need key ids for our data, so our server can know
            // what is what.
            String key_email = "email";
            String key_password = "password";

            Map<String, String> map = new HashMap<String, String>();
            // map.put(key, value);
            map.put(key_email, email);
            map.put(key_password, password);
            return map;
        }
```

```
    };

    // This is a policy that we need to specify to tell Volley, what
    // to do if it gets a timeout, how many times to retry, etc.
    stringRequest.setRetryPolicy(new RetryPolicy() {
            @Override
            public int getCurrentTimeout() {
                // Here goes the timeout.
                // The number is in milliseconds, 5000 is usually enough,
                // but you can up or low that number to fit your needs.
                return 50000;
            }
            @Override
            public int getCurrentRetryCount() {
                // The maximum number of attempts.
                // Again, the number can be anything you need.
                return 50000;
            }
            @Override
            public void retry(VolleyError error) throws VolleyError {
                // Here you could check if the retry count has gotten
                // to the maximum number, and if so, send a VolleyError
                // message or similar. For the sake of the example, I'll
                // show a Toast.
                Toast.makeText(getContext(), error.toString(), Toast.LENGTH_LONG).show();
            }
    });

    // And finally, we create a Volley Queue. For this example, I'm using
    // getContext(), because I was working with a Fragment. But context could
    // be "this", "getContext()", etc.
    RequestQueue requestQueue = Volley.newRequestQueue(getContext());
    requestQueue.add(stringRequest);

} else {
    // If, for example, the user inputs an email that is not currently
    // on your remote DB, here's where we can inform the user.
    Toast.makeText(getContext(), "Wrong email", Toast.LENGTH_LONG).show();
}
```

Utiliser Volley pour les requêtes HTTP

Ajoutez la dépendance de gradle dans build.gradle au niveau de l'application

```
compile 'com.android.volley:volley:1.0.0'
```

Ajoutez également l'autorisation android.permission.INTERNET au manifeste de votre application.

** Créez le singleton d'instance Volley RequestQueue dans votre application **

```
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;
}
}
```

Maintenant, vous pouvez utiliser l'instance de volley en utilisant la méthode getInstance () et ajouter une nouvelle requête dans la file d'attente en utilisant
`InitApplication.getInstance().addToQueue(request);`

Un exemple simple pour demander JsonObject à partir du serveur est

```
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));
```

Pour gérer les délais d'attente de Volley, vous devez utiliser `RetryPolicy` . Une stratégie de nouvelle tentative est utilisée dans le cas où une demande ne peut pas être complétée en raison d'une défaillance du réseau ou d'autres cas.

Volley fournit un moyen simple d'implémenter votre `RetryPolicy` pour vos demandes. Par défaut, Volley définit tous les délais de connexion et de socket sur 5 secondes pour toutes les requêtes. `RetryPolicy` est une interface où vous devez implémenter votre logique sur la manière dont vous souhaitez réessayer une requête particulière en cas de dépassement de délai.

Le constructeur prend les trois paramètres suivants:

- `initialTimeoutMs` - Spécifie le délai d'expiration du socket en millisecondes pour chaque tentative de nouvelle tentative.
- `maxNumRetries` - Le nombre de tentatives est tenté.
- `backoffMultiplier` - Multiplicateur utilisé pour déterminer le temps exponentiel défini sur socket pour chaque tentative de nouvelle tentative.

Réponse booléenne variable du serveur avec demande json dans volley

vous pouvez personnaliser la classe en dessous d'un

```
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;
        }
    }
}
```

utilisez ceci avec votre activité

```
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();
    }
```

Utiliser JSONArray comme corps de requête

Les requêtes par défaut intégrées dans volley ne permettent pas de passer un objet `JSONArray` tant

que corps de requête dans une requête `POST` . Au lieu de cela, vous pouvez uniquement transmettre un objet `JSON` tant que paramètre.

Toutefois, au lieu de transmettre un objet `JSON` tant que paramètre au constructeur de la demande, vous devez remplacer la méthode `getBody()` de la `Request.class` . Vous devez également transmettre `null` comme troisième paramètre:

```
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;
        }
    }
};
```

Explication des paramètres:

- `REQUEST_URL` - L'URL complète à laquelle envoyer votre demande.
- `RESP_LISTENER` - Un objet `Response.Listener<?>` , Dont la `onResponse(T data)` est appelée après son achèvement.
- `ERR_LISTENER` - Objet `Response.ErrorListener` , dont la `onErrorResponse(VolleyError e)` est appelée lors d'une requête infructueuse.

Lire Volée en ligne: https://riptutorial.com/fr/android/topic/2800/volee

Chapitre 264: WebView

Introduction

WebView est une vue qui affiche des pages Web dans votre application. De cette manière, vous pouvez ajouter votre propre URL.

Remarques

S'il vous plaît n'oubliez pas d'ajouter la permission dans votre fichier manifeste Android

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

Examples

Dialogues d'alerte JavaScript dans WebView - Comment les faire fonctionner

Par défaut, WebView n'implémente pas les boîtes de dialogue d'alerte JavaScript, c.-à-d. `alert()` ne fera rien. Pour que vous deviez activer JavaScript (évidemment ..), puis définir un `WebChromeClient` pour gérer les demandes de boîtes de dialogue d'alerte à partir de la page:

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

Ici, nous `onJsAlert` , puis nous appelons la super implémentation, ce qui nous donne une boîte de dialogue Android standard. Vous pouvez également utiliser le message et l'URL vous-même, par exemple si vous souhaitez créer une boîte de dialogue personnalisée ou si vous souhaitez les enregistrer.

Communication de Javascript vers Java (Android)

Activité Android

```
package com.example.myapp;

import android.os.Bundle;
import android.app.Activity;
import android.webkit.WebView;

public class WebViewActivity extends Activity {
    @Override
```

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

        WebView webView = new WebView(this);
        setContentView(webView);

       /*
        *   Note the label Android, this is used in the Javascript side of things
        *   You can of course change this.
        */
        webView.addJavascriptInterface(new JavascriptHandler(), "Android");

        webView.loadUrl("http://example.com");
    }
}
```

Java Javascript Handler

```
import android.webkit.JavascriptInterface;

public class JavascriptHandler {

    /**
     *  Key point here is the annotation @JavascriptInterface
     *
     */
    @JavascriptInterface
    public void jsCallback() {
       // Do something
    }

    @JavascriptInterface
    public void jsCallbackTwo(String dummyData) {
       // Do something
    }
}
```

Page Web, appel Javascript

```
<script>
...
Android.jsCallback();
...
Android.jsCallback('hello test');
...
</script>
```

Conseil supplémentaire

En passant dans une structure de données complexe, une solution possible est d'utiliser JSON.

```
Android.jsCallback('{ "fake-var" : "fake-value", "fake-array" : [0,1,2] }');
```

Du côté Android, utilisez votre analyseur JSON préféré, à savoir: JSONObject

Communication de Java à Javascript

Exemple de base

```
package com.example.myapp;

import android.os.Bundle;
import android.app.Activity;
import android.webkit.WebView;

public class WebViewActivity extends Activity {

    private Webview webView;

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

       webView = new WebView(this);
       webView.getSettings().setJavaScriptEnabled(true);

       setContentView(webView);

       webView.loadUrl("http://example.com");

       /*
        * Invoke Javascript function
        */
       webView.loadUrl("javascript:testJsFunction('Hello World!')");
    }

   /**
    * Invoking a Javascript function
    */
   public void doSomething() {
     this.webView.loadUrl("javascript:testAnotherFunction('Hello World Again!')");
   }
}
```

Ouvrir un exemple de numéroteur

Si la page Web contient un numéro de téléphone, vous pouvez passer un appel à l'aide du composeur de votre téléphone. Ce code vérifie l'URL qui commence par tel: alors faites une intention d'ouvrir le composeur et vous pouvez appeler le numéro de téléphone cliqué:

```
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;
}
```

Dépannage de WebView en imprimant des messages de console ou par le débogage à distance

Impression des messages de la console Webview dans logcat

Pour gérer `console` messages de `console` à partir de la page Web, vous pouvez remplacer `onConsoleMessage` dans `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;
    }
}
```

Et définissez-le dans votre activité ou fragment:

```
webView.setWebChromeClient(new ChromeClient());
```

Donc, cet exemple de page:

```
<html>
<head>
    <script type="text/javascript">
        console.log('test message');
    </script>
</head>
<body>
</body>
</html>
```

écrira log 'test message' à logcat:

> WebView: exemple de message de test.html: 4

`console.info()` , `console.warn()` et `console.error()` sont également supportés par chrome-client.

Débogage à distance des appareils Android avec Chrome

Vous pouvez déboguer une application basée sur Webview à partir de Google Chrome.

Activer le débogage USB sur votre appareil Android

Sur votre appareil Android, ouvrez Paramètres, recherchez la section Options de développeur et

activez le débogage USB.

Connectez et découvrez votre appareil Android

Ouvrir la page en chrome Page suivante: chrome: // inspect / # devices

Dans la boîte de dialogue Inspecter les périphériques, sélectionnez votre appareil et appuyez sur **inspecter** . Une nouvelle instance de Chrome DevTools s'ouvre sur votre machine de développement.

Des instructions et une description plus détaillées de DevTools sont disponibles sur developers.google.com.

Ouvrir un fichier local / Créer un contenu dynamique dans Webview

Layout.xml

```
            <WebView
            android:id="@+id/WebViewToDisplay"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:layout_gravity="center"
            android:fadeScrollbars="false" />
```

Charger des données dans 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");
```

Lire WebView en ligne: https://riptutorial.com/fr/android/topic/153/webview

Chapitre 265: Widgets

Remarques

SDv

Examples

Déclaration Manifeste -

Déclarez la classe `AppWidgetProvider` dans le fichier `AndroidManifest.xml` votre application. Par exemple:

```
<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>
```

Métadonnées

Ajoutez les métadonnées AppWidgetProviderInfo dans `res/xml` :

```
<appwidget-provider xmlns:android="http://schemas.android.com/apk/res/android"
    android:minWidth="40dp"
    android:minHeight="40dp"
    android:updatePeriodMillis="86400000"
    android:previewImage="@drawable/preview"
    android:initialLayout="@layout/example_appwidget"
    android:configure="com.example.android.ExampleAppWidgetConfigure"
    android:resizeMode="horizontal|vertical"
    android:widgetCategory="home_screen">
</appwidget-provider>
```

Classe AppWidgetProvider

Le rappel `AppWidgetProvider` plus important est **`onUpdate()`** . Il est appelé à chaque fois qu'un appwidget est ajouté.

```
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()` est appelée lorsque le widget est placé ou redimensionné.

`onDeleted(Context, int[])` est appelé lorsque le widget est supprimé.

Deux widgets avec une déclaration de mise en page différente

1. Déclarez deux récepteurs dans un fichier manifeste:

```
<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. Créer deux mises en page
    - `@xml/widget_1x1`
    - `@xml/widget_2x2`
3. Déclarez la sous-classe `UVMateWidget2x2` de la classe `UVMateWidget` avec un comportement étendu:

```
package au.com.aershov.uvmate;
```

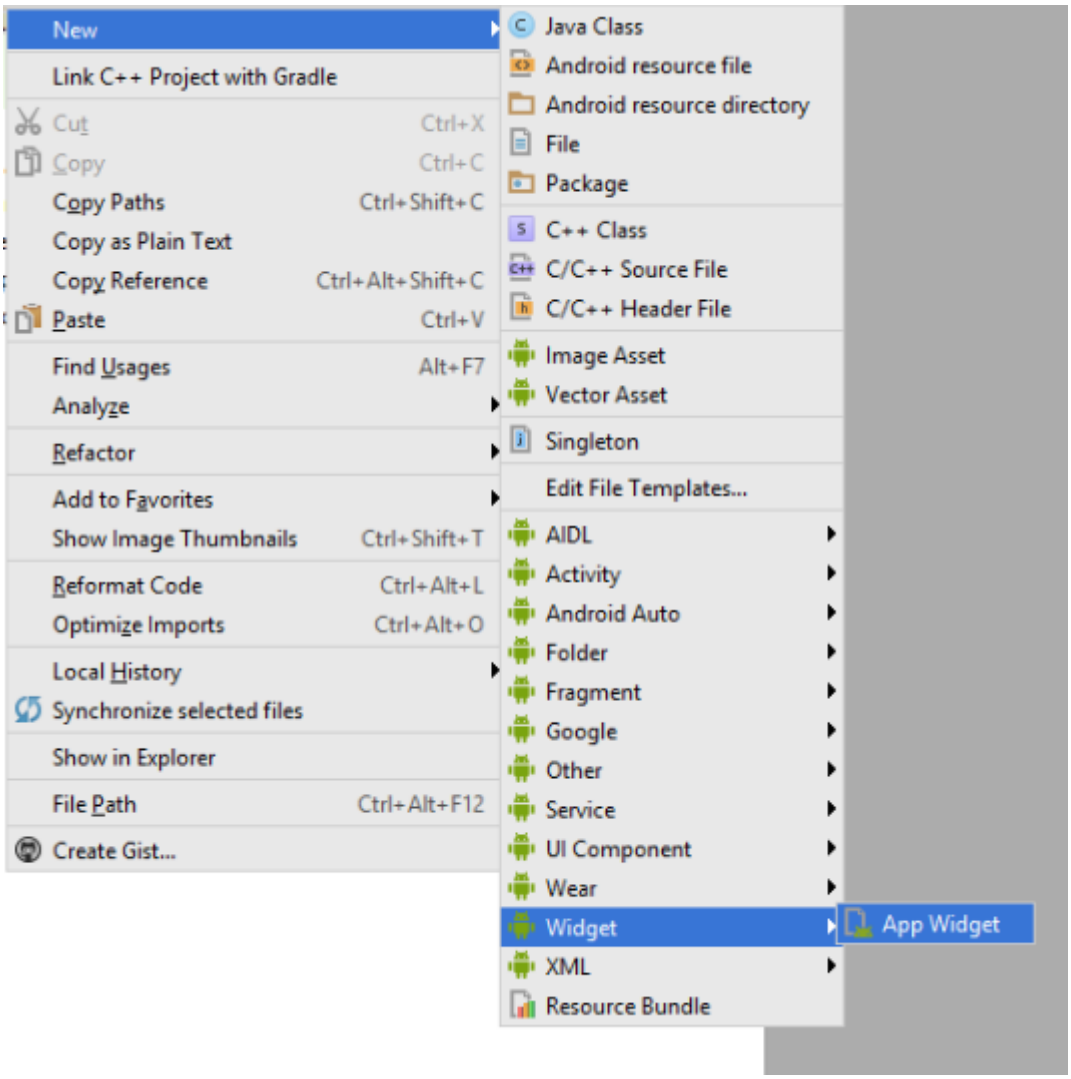
```
import android.content.Context;
import android.widget.RemoteViews;

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);
    }
}
```

Créer / intégrer un widget de base en utilisant Android Studio

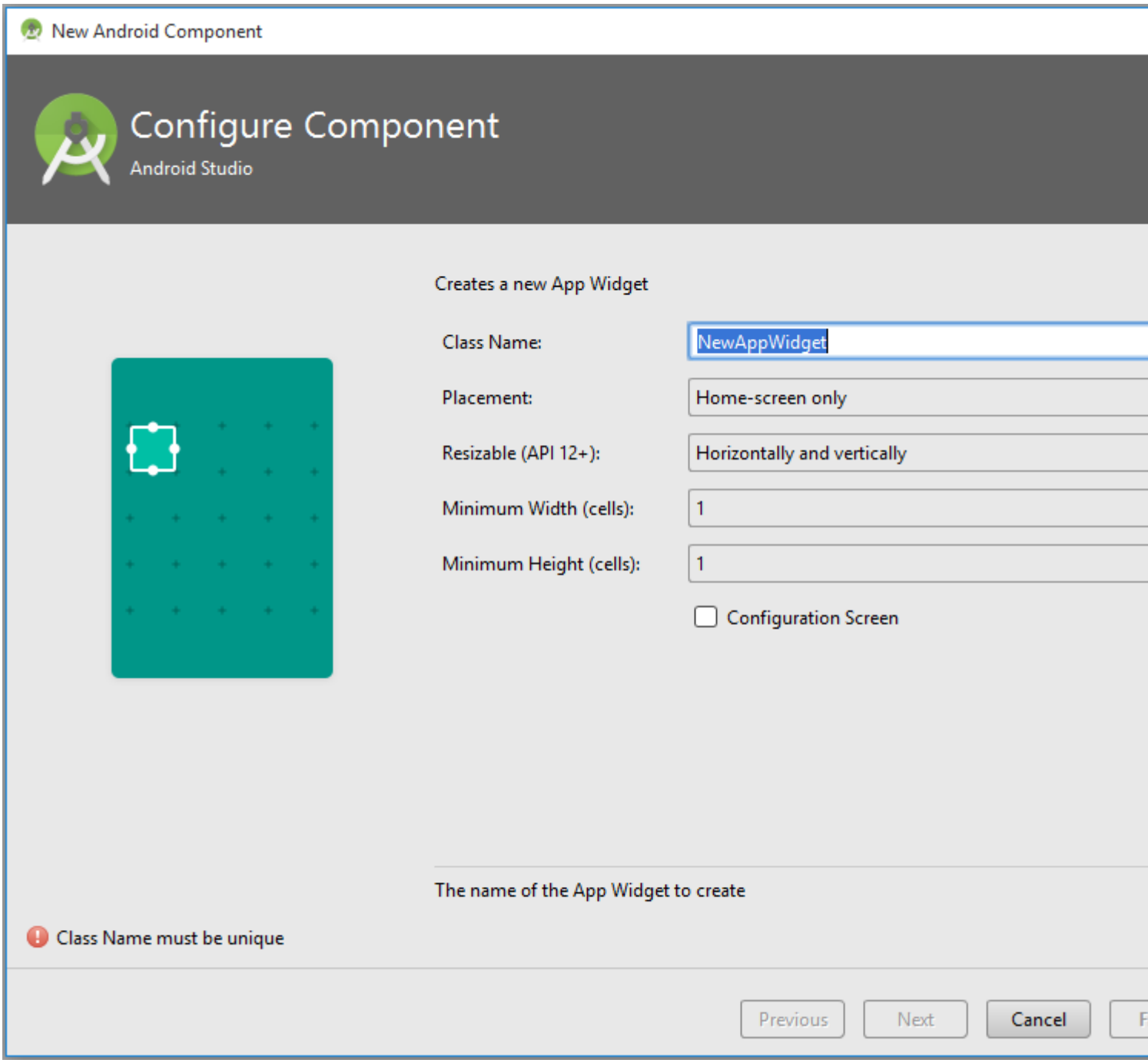
Dernier Android Studio va créer et intégrer un widget de base à votre application en 2 étapes.

Droit sur votre application ==> Nouveau ==> Widget ==> Widget App

.



Il affichera un écran comme ci-dessous et remplira les champs



C'est fait.

Il va **créer et intégrer un widget HelloWorld de base** (y compris un fichier de mise en page, un fichier de métadonnées, une déclaration dans un fichier manifeste, etc.) à votre application.

Lire Widgets en ligne: https://riptutorial.com/fr/android/topic/2812/widgets

Chapitre 266: XMPP s'inscrire à la session et à chatter exemple simple

Examples

Exemple d'enregistrement de base et de chat d'enregistrement XMPP

Installez openfire ou tout serveur de discussion sur votre système ou sur le serveur. Pour plus de détails, cliquez ici.

Créer un projet Android et ajouter ces bibliothèques dans Gradle:

```
compile 'org.igniterealtime.smack:smack-android:4.2.0'
compile 'org.igniterealtime.smack:smack-tcp:4.2.0'
compile 'org.igniterealtime.smack:smack-im:4.2.0'
compile 'org.igniterealtime.smack:smack-android-extensions:4.2.0'
```

Ensuite, créez une classe xmpp à partir de la connexion xmpp:

```
public class XMPP {

public static final int PORT = 5222;
private static XMPP instance;
private XMPPTCPConnection connection;
private static String TAG = "XMPP-EXAMPLE";
public static final String ACTION_LOGGED_IN = "liveapp.loggedin";
private String HOST = "192.168.0.10";

private XMPPTCPConnectionConfiguration buildConfiguration() throws XmppStringprepException {
    XMPPTCPConnectionConfiguration.Builder builder =
            XMPPTCPConnectionConfiguration.builder();


    builder.setHost(HOST);
    builder.setPort(PORT);
    builder.setCompressionEnabled(false);
    builder.setDebuggerEnabled(true);
    builder.setSecurityMode(ConnectionConfiguration.SecurityMode.disabled);
    builder.setSendPresence(true);

    if (Build.VERSION.SDK_INT >= 14) {
        builder.setKeystoreType("AndroidCAStore");
        // config.setTruststorePassword(null);
        builder.setKeystorePath(null);
    } else {
        builder.setKeystoreType("BKS");
        String str = System.getProperty("javax.net.ssl.trustStore");
        if (str == null) {
            str = System.getProperty("java.home") + File.separator + "etc" + File.separator +
"security"
                    + File.separator + "cacerts.bks";
        }
        builder.setKeystorePath(str);
```

```
    }
    DomainBareJid serviceName = JidCreate.domainBareFrom(HOST);
    builder.setServiceName(serviceName);


    return builder.build();
}

private XMPPTCPConnection getConnection() throws XMPPException, SmackException, IOException,
InterruptedException {
    Log.logDebug(TAG, "Getting XMPP Connect");
    if (isConnected()) {
        Log.logDebug(TAG, "Returning already existing connection");
        return this.connection;
    }

    long l = System.currentTimeMillis();
    try {
        if(this.connection != null){
            Log.logDebug(TAG, "Connection found, trying to connect");
            this.connection.connect();
        }else{
            Log.logDebug(TAG, "No Connection found, trying to create a new connection");
            XMPPTCPConnectionConfiguration config = buildConfiguration();
            SmackConfiguration.DEBUG = true;
            this.connection = new XMPPTCPConnection(config);
            this.connection.connect();
        }
    } catch (Exception e) {
        Log.logError(TAG,"some issue with getting connection :" + e.getMessage());

    }

    Log.logDebug(TAG, "Connection Properties: " + connection.getHost() + " " +
connection.getServiceName());
    Log.logDebug(TAG, "Time taken in first time connect: " + (System.currentTimeMillis() -
l));
    return this.connection;
}

public static XMPP getInstance() {
    if (instance == null) {
        synchronized (XMPP.class) {
            if (instance == null) {
                instance = new XMPP();
            }
        }
    }
    return instance;
}

public void close() {
    Log.logInfo(TAG, "Inside XMPP close method");
    if (this.connection != null) {
        this.connection.disconnect();
    }
}

private XMPPTCPConnection connectAndLogin(Context context) {
    Log.logDebug(TAG, "Inside connect and Login");
    if (!isConnected()) {
```

```
        Log.logDebug(TAG, "Connection not connected, trying to login and connect");
        try {
           // Save username and password then use here
            String username = AppSettings.getUser(context);
            String password = AppSettings.getPassword(context);
            this.connection = getConnection();
            Log.logDebug(TAG, "XMPP username :" + username);
            Log.logDebug(TAG, "XMPP password :" + password);
            this.connection.login(username, password);
            Log.logDebug(TAG, "Connect and Login method, Login successful");
            context.sendBroadcast(new Intent(ACTION_LOGGED_IN));
        } catch (XMPPException localXMPPException) {
            Log.logError(TAG, "Error in Connect and Login Method");
            localXMPPException.printStackTrace();
        } catch (SmackException e) {
            Log.logError(TAG, "Error in Connect and Login Method");
            e.printStackTrace();
        } catch (IOException e) {
            Log.logError(TAG, "Error in Connect and Login Method");
            e.printStackTrace();
        } catch (InterruptedException e) {
            Log.logError(TAG, "Error in Connect and Login Method");
            e.printStackTrace();
        } catch (IllegalArgumentException e) {
            Log.logError(TAG, "Error in Connect and Login Method");
            e.printStackTrace();
        } catch (Exception e) {
            Log.logError(TAG, "Error in Connect and Login Method");
            e.printStackTrace();
        }
    }
    Log.logInfo(TAG, "Inside getConnection - Returning connection");
    return this.connection;
}

public boolean isConnected() {
    return (this.connection != null) && (this.connection.isConnected());
}

public EntityFullJid getUser() {
    if (isConnected()) {
        return connection.getUser();
    } else {
        return null;
    }
}

public void login(String user, String pass, String username)
        throws XMPPException, SmackException, IOException, InterruptedException,
PurplKiteXMPPConnectException {
    Log.logInfo(TAG, "inside XMPP getlogin Method");
    long l = System.currentTimeMillis();
    XMPPTCPConnection connect = getConnection();
    if (connect.isAuthenticated()) {
        Log.logInfo(TAG, "User already logged in");
        return;
    }

    Log.logInfo(TAG, "Time taken to connect: " + (System.currentTimeMillis() - l));

    l = System.currentTimeMillis();
```

```
    try{
        connect.login(user, pass);
    }catch (Exception e){
        Log.logError(TAG, "Issue in login, check the stacktrace");
        e.printStackTrace();
    }

    Log.logInfo(TAG, "Time taken to login: " + (System.currentTimeMillis() - l));

    Log.logInfo(TAG, "login step passed");

    PingManager pingManager = PingManager.getInstanceFor(connect);
    pingManager.setPingInterval(5000);

}

public void register(String user, String pass) throws XMPPException,
SmackException.NoResponseException, SmackException.NotConnectedException {
    Log.logInfo(TAG, "inside XMPP register method, " + user + " : " + pass);
    long l = System.currentTimeMillis();
    try {
        AccountManager accountManager = AccountManager.getInstance(getConnection());
        accountManager.sensitiveOperationOverInsecureConnection(true);
        accountManager.createAccount(Localpart.from(user), pass);
    } catch (SmackException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    } catch (InterruptedException e) {
        e.printStackTrace();
    } catch (PurplKiteXMPPConnectException e) {
        e.printStackTrace();
    }
    Log.logInfo(TAG, "Time taken to register: " + (System.currentTimeMillis() - l));
}


public void addStanzaListener(Context context, StanzaListener stanzaListener){
    XMPPTCPConnection connection =  connectAndLogin(context);
    connection.addAsyncStanzaListener(stanzaListener, null);
}

public void removeStanzaListener(Context context, StanzaListener stanzaListener){
    XMPPTCPConnection connection =  connectAndLogin(context);
    connection.removeAsyncStanzaListener(stanzaListener);
}

public void addChatListener(Context context, ChatManagerListener chatManagerListener){
    ChatManager.getInstanceFor(connectAndLogin(context))
            .addChatListener(chatManagerListener);
}

public void removeChatListener(Context context, ChatManagerListener chatManagerListener){

ChatManager.getInstanceFor(connectAndLogin(context)).removeChatListener(chatManagerListener);
}

public void getSrvDeliveryManager(Context context){
    ServiceDiscoveryManager sdm = ServiceDiscoveryManager
            .getInstanceFor(XMPP.getInstance().connectAndLogin(
                    context));
```

```
    //sdm.addFeature("http://jabber.org/protocol/disco#info");
    //sdm.addFeature("jabber:iq:privacy");
    sdm.addFeature("jabber.org/protocol/si");
    sdm.addFeature("http://jabber.org/protocol/si");
    sdm.addFeature("http://jabber.org/protocol/disco#info");
    sdm.addFeature("jabber:iq:privacy");

}

public String getUserLocalPart(Context context){
   return  connectAndLogin(context).getUser().getLocalpart().toString();
}

public EntityFullJid getUser(Context context){
    return  connectAndLogin(context).getUser();
}

public Chat getThreadChat(Context context, String party1, String party2){
    Chat chat = ChatManager.getInstanceFor(
            XMPP.getInstance().connectAndLogin(context))
            .getThreadChat(party1 + "-" + party2);
    return chat;
}

public Chat createChat(Context context, EntityJid jid, String party1, String party2,
ChatMessageListener messageListener){
    Chat chat = ChatManager.getInstanceFor(
            XMPP.getInstance().connectAndLogin(context))
            .createChat(jid, party1 + "-" + party2,
                    messageListener);
    return chat;
}

public void sendPacket(Context context, Stanza packet){
    try {
        connectAndLogin(context).sendStanza(packet);
    } catch (SmackException.NotConnectedException e) {
        e.printStackTrace();
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
}

}
```

Enfin, ajoutez cette activité:

```
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();
    }

}
```

Assurez-vous que l'autorisation Internet est ajoutée dans votre fichier manifeste.

Lire XMPP s'inscrire à la session et à chatter exemple simple en ligne: https://riptutorial.com/fr/android/topic/6747/xmpp-s-inscrire-a-la-session-et-a-chatter-exemple-simple

Chapitre 267: Xposed

Examples

Créer un module Xposed

Xposed est un framework qui vous permet d'accrocher des appels de méthode d'autres applications. Lorsque vous effectuez une modification en décompilant un APK, vous pouvez insérer / modifier des commandes directement où vous voulez. Cependant, vous devrez recompiler / signer l'APK par la suite et vous ne pourrez distribuer que le package complet. Avec Xposed, vous pouvez injecter votre propre code avant ou après les méthodes, ou remplacer complètement des méthodes entières. Malheureusement, vous ne pouvez installer Xposed que sur des appareils rootés. Vous devez utiliser Xposed chaque fois que vous souhaitez manipuler le comportement d'autres applications ou du système Android principal et ne pas avoir à vous soucier de la décompilation, de la recompilation et de la signature d'APK.

Tout d'abord, vous créez une application standard sans activité dans Android Studio.

Ensuite, vous devez inclure le code suivant dans votre *build.gradle* :

```
repositories {
    jcenter();
}
```

Après cela, vous ajoutez les dépendances suivantes:

```
provided 'de.robv.android.xposed:api:82'
provided 'de.robv.android.xposed:api:82:sources'
```

Maintenant, vous devez placer ces balises dans la balise d' *application* trouvée dans le *fichier AndroidManifest.xml* afin que Xposed reconnaisse votre module:

```
<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" />
```

REMARQUE: remplacez toujours **82** par la dernière version de Xposed .

Accrocher une méthode

Créez une nouvelle classe implémentant `IXposedHookLoadPackage` et implémentez la méthode

`handleLoadPackage` :

```
public class MultiPatcher implements IXposedHookLoadPackage
{
    @Override
    public void handleLoadPackage(XC_LoadPackage.LoadPackageParam loadPackageParam) throws
Throwable
    {

    }
}
```

À l'intérieur de la méthode, cochez `loadPackageParam.packageName` pour le nom de package de l'application que vous souhaitez raccorder:

```
@Override
public void handleLoadPackage(XC_LoadPackage.LoadPackageParam loadPackageParam) throws
Throwable
{
    if (!loadPackageParam.packageName.equals("other.package.name"))
    {
        return;
    }
}
```

Maintenant, vous pouvez accrocher votre méthode et la manipuler avant l'exécution du code ou après:

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

Lire Xposed en ligne: https://riptutorial.com/fr/android/topic/4627/xposed

Chapitre 268: Youtube-API

Remarques

1. Tout d'abord, vous devez télécharger le dernier fichier jar ci-dessous: https://developers.google.com/youtube/android/player/downloads/
2. Vous devez inclure ce pot dans votre projet. Copiez et collez ce fichier jar dans le dossier libs et n'oubliez pas de l'ajouter dans les dépendances des fichiers graduels {compiler les fichiers ('libs / YouTubeAndroidPlayerApi.jar')}
3. Vous avez besoin d'une clé api pour accéder à api youtube. Suivez ce lien: https://developers.google.com/youtube/android/player/register pour générer votre clé API.
4. Nettoyez et construisez votre projet. Maintenant, vous êtes prêt à utiliser YoutubeAndroidPlayerApi Pour lire une vidéo sur youtube, vous devez avoir un identifiant vidéo correspondant afin de pouvoir le lire sur youtube. Par exemple: https://www.youtube.com/watch?v=B08iLAtS3AQ , B08iLAtS3AQ est l'identifiant vidéo que vous devez lire sur YouTube.

Examples

Lancer StandAlonePlayerActivity

1. Lancer une activité de joueur autonome

```
     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);
```

Activité prolongeant YouTubeBaseActivity

```
public class CustomYouTubeActivity extends YouTubeBaseActivity implements
YouTubePlayer.OnInitializedListener, YouTubePlayer.PlayerStateChangeListener {

    private YouTubePlayerView mPlayerView;
    private YouTubePlayer mYouTubePlayer;
    private String mVideoId = "B08iLAtS3AQ";
    private String mApiKey;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        mApiKey = Config.YOUTUBE_API_KEY;
        mPlayerView = new YouTubePlayerView(this);
        mPlayerView.initialize(mApiKey, this); // setting up OnInitializedListener
```

```
        addContentView(mPlayerView, new LayoutParams(LayoutParams.MATCH_PARENT,
                LayoutParams.MATCH_PARENT)); //show it in full screen
    }

    //Called when initialization of the player succeeds.
    @Override
    public void onInitializationSuccess(YouTubePlayer.Provider provider,
                                        YouTubePlayer player,
                                        boolean wasRestored) {

        player.setPlayerStateChangeListener(this); // setting up the player state change
listener
        this.mYouTubePlayer = player;
        if (!wasRestored)
            player.cueVideo(mVideoId);
    }

        @Override
    public void onInitializationFailure(YouTubePlayer.Provider provider,
                                        YouTubeInitializationResult errorReason) {

        Toast.makeText(this, "Error While initializing", Toast.LENGTH_LONG).show();
    }

    @Override
    public void onAdStarted() {
    }

    @Override
    public void onLoaded(String videoId) { //video has been loaded
        if(!TextUtils.isEmpty(mVideoId) && !this.isFinishing() && mYouTubePlayer != null)
            mYouTubePlayer.play(); // if we dont call play then video will not auto play, but
user still has the option to play via play button
    }

    @Override
    public void onLoading() {
    }

    @Override
    public void onVideoEnded() {
    }

    @Override
    public void onVideoStarted() {

    }

    @Override
    public void onError(ErrorReason reason) {
        Log.e("onError", "onError : " + reason.name());
    }

}
```

YoutubePlayerFragment dans Portrait Activty

Le code suivant implémente un simple YoutubePlayerFragment. La mise en page de l'activité est verrouillée en mode portrait et lorsque l'orientation change ou que l'utilisateur clique sur plein

écran sur YoutubePlayer, il devient un écran lansscape avec YoutubePlayer remplissant l'écran. Le YoutubePlayerFragment n'a pas besoin d'étendre une activité fournie par la bibliothèque Youtube. Il doit implémenter YouTubePlayer.OnInitializedListener pour que YoutubePlayer soit initialisé. La classe de notre activité est donc la suivante

```
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.util.Log;
import android.widget.Toast;

import com.google.android.youtube.player.YouTubeInitializationResult;
import com.google.android.youtube.player.YouTubePlayer;
import com.google.android.youtube.player.YouTubePlayerFragment;

public class MainActivity extends AppCompatActivity implements
YouTubePlayer.OnInitializedListener {

    public static final String API_KEY ;
    public static final String VIDEO_ID = "B08iLAtS3AQ";

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

        YouTubePlayerFragment youTubePlayerFragment = (YouTubePlayerFragment)
getFragmentManager()
                .findFragmentById(R.id.youtubeplayerfragment);

        youTubePlayerFragment.initialize(API_KEY, this);

    }

    /**
     *
     * @param provider The provider which was used to initialize the YouTubePlayer
     * @param youTubePlayer A YouTubePlayer which can be used to control video playback in the
provider.
     * @param wasRestored Whether the player was restored from a previously saved state, as
part of the YouTubePlayerView
     *                    or YouTubePlayerFragment restoring its state. true usually means
playback is resuming from where
     *                    the user expects it would, and that a new video should not be loaded
     */
    @Override
    public void onInitializationSuccess(YouTubePlayer.Provider provider, YouTubePlayer
youTubePlayer, boolean wasRestored) {

youTubePlayer.setFullscreenControlFlags(YouTubePlayer.FULLSCREEN_FLAG_CONTROL_ORIENTATION |
                    YouTubePlayer.FULLSCREEN_FLAG_ALWAYS_FULLSCREEN_IN_LANDSCAPE);


        if(!wasRestored) {
            youTubePlayer.cueVideo(VIDEO_ID);
        }
    }

    /**
     *
     * @param provider The provider which failed to initialize a YouTubePlayer.
```

```
     * @param error The reason for this failure, along with potential resolutions to this
failure.
     */
    @Override
    public void onInitializationFailure(YouTubePlayer.Provider provider,
YouTubeInitializationResult error) {

        final int REQUEST_CODE = 1;

        if(error.isUserRecoverableError()) {
            error.getErrorDialog(this,REQUEST_CODE).show();
        } else {
            String errorMessage = String.format("There was an error initializing the
YoutubePlayer (%1$s)", error.toString());
            Toast.makeText(this, errorMessage, Toast.LENGTH_LONG).show();
        }
    }
}
```

Un YoutubePlayerFragment peut être ajouté à la disposition xaml de l'activité comme suit

```
<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>
```

Enfin, vous devez ajouter les attributs suivants dans votre fichier manifeste à l'intérieur du tag de l'activité

```
android:configChanges="keyboardHidden|orientation|screenSize"
android:screenOrientation="portrait"
```

API du lecteur YouTube

Obtenir la clé de l'API Android:

Tout d'abord, vous devez obtenir l'empreinte SHA-1 sur votre machine à l'aide de l'outil clé Java. Exécutez la commande ci-dessous dans cmd / terminal pour obtenir l'empreinte SHA-1.

```
keytool -list -v -keystore ~/.android/debug.keystore -alias androiddebugkey -storepass android
-keypass android
```

MainActivity.java

```
public class Activity extends YouTubeBaseActivity implements
YouTubePlayer.OnInitializedListener {

    private static final int RECOVERY_DIALOG_REQUEST = 1;

    // YouTube player view
    private YouTubePlayerView youTubeView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
                WindowManager.LayoutParams.FLAG_FULLSCREEN);

        setContentView(R.layout.activity_main);

        youTubeView = (YouTubePlayerView) findViewById(R.id.youtube_view);

        // Initializing video player with developer key
        youTubeView.initialize(Config.DEVELOPER_KEY, this);

    }

    @Override
    public void onInitializationFailure(YouTubePlayer.Provider provider,
                                        YouTubeInitializationResult errorReason) {
        if (errorReason.isUserRecoverableError()) {
            errorReason.getErrorDialog(this, RECOVERY_DIALOG_REQUEST).show();
        } else {
            String errorMessage = String.format(
                    getString(R.string.error_player), errorReason.toString());
            Toast.makeText(this, errorMessage, Toast.LENGTH_LONG).show();
        }
    }

    @Override
    public void onInitializationSuccess(YouTubePlayer.Provider provider,
                                        YouTubePlayer player, boolean wasRestored) {
        if (!wasRestored) {

            // loadVideo() will auto play video
            // Use cueVideo() method, if you don't want to play it automatically
            player.loadVideo(Config.YOUTUBE_VIDEO_CODE);

            // Hiding player controls
            player.setPlayerStyle(YouTubePlayer.PlayerStyle.CHROMELESS);
        }
    }

    @Override
```

```
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        if (requestCode == RECOVERY_DIALOG_REQUEST) {
            // Retry initialization if user performed a recovery action
            getYouTubePlayerProvider().initialize(Config.DEVELOPER_KEY, this);
        }
    }

    private YouTubePlayer.Provider getYouTubePlayerProvider() {
        return (YouTubePlayerView) findViewById(R.id.youtube_view);
    }
}
```

Maintenant, créez le fichier `Config.java` . Ce fichier contient la clé de développeur de l'API Google Console et l'identifiant de la vidéo YouTube.

Config.java

```
public class Config {

    // Developer key
    public static final String DEVELOPER_KEY = "AIzaSyDZtE10od_hXM5aXYEh6Zn7c6brV9ZjKuk";

    // YouTube video id
    public static final String YOUTUBE_VIDEO_CODE = "_oEA18Y8gM0";
}
```

fichier xml

```
<com.google.android.youtube.player.YouTubePlayerView
                android:id="@+id/youtube_view"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginBottom="30dp" />
```

Consommer YouTube Data API sur Android

Cet exemple vous guidera pour obtenir des données de playlist en utilisant l'API YouTube Data sur Android.

Empreinte digitale SHA-1

Vous devez d'abord obtenir une empreinte SHA-1 pour votre machine. Il existe différentes méthodes pour le récupérer. Vous pouvez choisir n'importe quelle méthode fournie dans cette Q & R.

Console Google API et clé YouTube pour Android

Maintenant que vous avez une empreinte SHA-1, ouvrez la console Google API et créez un projet. Accédez à cette page et créez un projet à l'aide de cette clé SHA-1 et activez l'API YouTube Data. Maintenant, vous aurez une clé. Cette clé sera utilisée pour envoyer des requêtes à partir d'Android et récupérer des données.

Gradle part

Vous devrez ajouter les lignes suivantes à votre fichier Gradle pour l'API YouTube Data:

```
compile 'com.google.apis:google-api-services-youtube:v3-rev183-1.22.0'
```

Pour utiliser le client natif de YouTube pour envoyer des requêtes, nous devons ajouter les lignes suivantes dans Gradle:

```
compile 'com.google.http-client:google-http-client-android:+'
compile 'com.google.api-client:google-api-client-android:+'
compile 'com.google.api-client:google-api-client-gson:+'
```

La configuration suivante doit également être ajoutée dans Gradle pour éviter les conflits:

```
configurations.all {
    resolutionStrategy.force 'com.google.code.findbugs:jsr305:3.0.2'
}
```

Ci-dessous, il est montré à quoi *ressemblerait* le *gradle.build* .

build.gradle

```
apply plugin: 'com.android.application'
android {
    compileSdkVersion 25
    buildToolsVersion "25.0.2"
    defaultConfig {
        applicationId "com.aam.skillschool"
        minSdkVersion 19
        targetSdkVersion 25
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
    configurations.all {
        resolutionStrategy.force 'com.google.code.findbugs:jsr305:3.0.2'
    }
}

dependencies {
    compile fileTree(include: ['*.jar'], dir: 'libs')
    androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2', {
        exclude group: 'com.android.support', module: 'support-annotations'
    })
    compile 'com.google.apis:google-api-services-youtube:v3-rev183-1.22.0'
    compile 'com.android.support:appcompat-v7:25.3.1'
    compile 'com.android.support:support-v4:25.3.1'
    compile 'com.google.http-client:google-http-client-android:+'
    compile 'com.google.api-client:google-api-client-android:+'
    compile 'com.google.api-client:google-api-client-gson:+'
}
```

Maintenant vient la partie Java. Puisque nous utiliserons `HttpTransport` pour la mise en réseau et `GsonFactory` pour convertir JSON en POJO, nous n'avons besoin d'aucune autre bibliothèque pour envoyer des requêtes.

Maintenant, je veux montrer comment obtenir des listes de lecture via l'API YouTube en fournissant les ID de liste de lecture. Pour cette tâche, je vais utiliser `AsyncTask` . Pour comprendre comment nous demandons des paramètres et pour comprendre le flux, veuillez consulter l' API YouTube Data .

```
public class GetPlaylistDataAsyncTask extends AsyncTask<String[], Void, PlaylistListResponse>
{
    private static final String YOUTUBE_PLAYLIST_PART = "snippet";
    private static final String YOUTUBE_PLAYLIST_FIELDS = "items(id,snippet(title))";

    private YouTube mYouTubeDataApi;

    public GetPlaylistDataAsyncTask(YouTube api) {
        mYouTubeDataApi = api;
    }

    @Override
    protected PlaylistListResponse doInBackground(String[]... params) {

        final String[] playlistIds = params[0];

        PlaylistListResponse playlistListResponse;
        try {
            playlistListResponse = mYouTubeDataApi.playlists()
                    .list(YOUTUBE_PLAYLIST_PART)
                    .setId(TextUtils.join(",", playlistIds))
                    .setFields(YOUTUBE_PLAYLIST_FIELDS)
                    .setKey(AppConstants.YOUTUBE_KEY) //Here you will have to provide the keys
                    .execute();
        } catch (IOException e) {
            e.printStackTrace();
            return null;
        }

        return playlistListResponse;
    }
}
```

La tâche asynchrone ci-dessus renvoie une instance de `PlaylistListResponse` qui est une classe intégrée du SDK YouTube. Il a tous les champs obligatoires, donc nous n'avons pas à créer des POJO nous-mêmes.

Enfin, dans notre `MainActivity` nous devrons faire ce qui suit:

```
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);
    }
}
```

Lire Youtube-API en ligne: https://riptutorial.com/fr/android/topic/7587/youtube-api

Crédits

S. No	Chapitres	Contributeurs
1	Démarrer avec 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, ושי אהוא בתוך
2	Accès aux bases de données SQLite à l'aide de la classe ContentValues	Adil Saiyad, emecas, honk
3	ACRA	Zarul Izham, Zoe
4	Activité	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
5	ADB (Android Debug Bridge)	3VYZkz7t, adao7000, Ahmad Aghazadeh, Amit Thakkar, AndroidMechanic, Anirudh Sharma, Anup Kulkarni, auval, Barend, Blackbelt, Burak Day, Charuක, Chris Stratton, Da-Jin C, Dale, Daniel Nugent, David Cheung, Erik, Fabio, fyfyone Google, g4s8, Gabriele Mariotti, grebulon, Hannoun Yassir, Hi I'm Frogatto, hichris123, honk, jim,

		Kashyap Jha, Laurel, MCeley, Menasheh, Natali, Nemus, Pavel Durov, Piyush, R. Zagórski, RishbhSharma, stkent, Sudip Bhandari, sukumar, theFunkyEngineer, thiagolr, Tien , Xaver Kapeller, Yassie, younes zeboudj, Yury Fedorov
6	AdMob	Carlos Borau, honk, RamenChef, Sukrit Kumar, Zarul Izham, Zoe
7	Affichage	Beena, Daniel Nugent, Eyad Mhanna, gaara87, Gabriele Mariotti, Hiren Patel, honk, keno, Michele, Sohail Zahid, Sujith Niraikulathan, sun-solar-arrow
8	Affichage des annonces Google	Egek92, RamenChef, ReverseCold, Stephen Leppik, Zarul Izham
9	AIDL	Krishnakanth
10	Ajout d'un FuseView à un projet Android	Tudor Luca
11	AlarmManager	Daniel Nugent, devnull69, Greg T, honk, TR4Android
12	Amélioration des dialogues d'alerte	Adil Saiyad, honk
13	Amélioration des performances Android à l'aide des polices Icon	Beto Caldas, honk, Neeraj
14	Android Java Native Interface (JNI)	Doron Yakovlev-Golani, Muthukrishnan Rajendran, samgak
15	Android Vk Sdk	alexey polusov
16	Android-x86 dans VirtualBox	Daniel Nugent, Enrique de Miguel
17	Animateurs	Aryan, Bartek Lipinski, Blundering Philosopher, Brenden Kromhout, Charuක, Daniel Nugent, Eixx, Hiren Patel, Lewis McGeary, Piyush, TR4Android, Uriel Carrillo, Yury Fedorov
18	Annotations Typedef: @IntDef, @StringDef	Gabriele Mariotti, hardik m, mmBs, Pongpat
19	API Android Places	busradeniz, honk, Karan Razdan, Murali
20	API Bluetooth et Bluetooth LE	antonio, Jon Adams, Lukas, Myon, Pavel Durov, R. Zagórski, Reaz Murshed, V-PTR, WMios

21	API d'empreintes digitales dans Android	Doron Yakovlev-Golani, RamenChef, user01232
22	API de sensibilisation Google	Dus, honk, Willie Chalmers III
23	API Google Drive	Christlin Joseph, honk
24	API Twitter	Mahmoud Ibrahim
25	Applications compatibles avec la construction en amont	Jon Adams, mnoronha, RamenChef, SoroushA
26	Architecture MVP	Atif Farrukh, Harish Gyanani, honk, Jon Adams, Magesh Pandian, N J, zmingchun
27	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
28	AudioManager	honk, Nicolai Weitkemper
29	Authentificateur Android	4444, kRiZ
30	AutoCompleteTextView	Harish Gyanani, Jon Adams, Ricardo Vieira, Vivek Mishra
31	Autorisations d'exécution dans 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
32	Autosizing TextViews	honk, Priyank Patel
33	Avertissements de peluches	ben75, Daniel Nugent, Gabriele Mariotti, GensaGames, R. Zagórski, rekire, SuperBiasedMan
34	Barre de progression	Gabriele Mariotti, Hiren Patel, mpkuth, Sanoop, shtolik

35	Bibliothèque Dagger 2: injection de dépendance dans les applications	Er. Kaushik Kajavadara, honk
36	Bibliothèque de liaison de données	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
37	Bluetooth Low Energy	Roberto Betancourt
38	Boîte AlertDialog Animée	krunal patel, Thomas Easo
39	BottomNavigationView	Abdul Wasae, Daniel Nugent, Gabriele Mariotti, guik, Pankaj Kumar, Pratik Butani, Priyank Patel, RamenChef, rciovati, Stephen Leppik, sud007
40	Bouton	Aleksandar Stefanović, BlitzKraig, Carlos Borau, Community, Daniel Nugent, Gabriele Mariotti, James_Parsons, Jordi Castilla, Mauro Frezza, Michael Spitsin, Muhammed Refaat, Nick Cardoso, Nougat Lover, r3flss ExlUtr, RamenChef, Ricardo Vieira, sun-solar-arrow, webo80
41	BroadcastReceiver	0x0000eWan, Adarsh Ashok, anupam_kamble, Daniel Nugent, g4s8, Hiren Patel, Ichthyocentaurs, Jon Adams, Joscandreu, Kirill Kulakov, Lazy Ninja, Leo.Han, Medusalix , param, Phil, Rajesh, Squidward, W0rmH0le
42	Cache Bitmap	Lokesh Desai
43	Camera 2 API	ChemicalFlash, devnull69, RamenChef, webo80
44	Caméra et Galerie	Ahmad Aghazadeh, carvaq, Daniel Nugent, Hiren Patel, johnrao07, RediOne1, Squidward, Yasin Kaçmaz
45	Canal de notification Android O	Lokesh Desai
46	Capturer des captures d'écran	Ayush Bansal, Daniel Nugent, honk, Onik, sushant kumar, W0rmH0le
47	CardView	Carlos, Dan, Er. Kaushik Kajavadara, Gabriele Mariotti, Kaushik, Nougat Lover, RamenChef, S.R, Sneh Pandya, Somesh Kumar, Stephen Leppik, sud007, WarrenFaith, Yury Fedorov

48	Carte à puce	shadygoneinsane
49	Chaînes de formatage	Beena, Daniel Nugent, gaara87, Greg T, Michele, RamenChef, Suresh Kumar
50	Changements d'orientation	EmmanuelMess, k3b, Ricardo Vieira, y.feizi
51	Chargement efficace de bitmaps	iDevRoids
52	Chargeur	chandsie, g4s8, jefry jacky, Marcus Becker, RamenChef, Stephen Leppik
53	Chargeur d'image universel	Greg T, honk, Jon Adams, priyankvex, Stephen Leppik
54	Chiffrement / déchiffrement des données	honk, HoseinIT, Robert
55	Choses Android	Fabio, honk
56	Clavier	Hiren Patel, Kayvan N
57	CleverTap	Jordan, judepereira
58	Code à barres et lecture du code QR	FlyingPumba
59	Comment stocker les mots de passe de manière sécurisée	honk, Jaggs
60	Comment utiliser SparseArray	honk, Robert Banyai
61	Composants d'architecture Android	DeKaNszn
62	Compression d'image	Hiren Patel, mnoronha
63	Compte à rebours	privatestaticint
64	Comptes et AccountManager	gaara87, systemovich
65	Conception matérielle	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
66	Configuration de Jenkins CI pour les projets Android	honk, Ichthyocentaurs
67	Connexions Wi-Fi	4444, AndroidMechanic, Daniel Nugent, gus27
68	ContrainteLayout	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
69	ContrainteSet	Pratik Butani
70	Conversion de la parole en texte	Hitesh Sahu, honk, RamenChef, Stephen Leppik
71	Convertir une chaîne vietnamienne en anglais	1SStorm
72	CoordinateurLayout et comportements	Adarsh Ashok, Gabriele Mariotti, honk, RamenChef, Stephen Leppik
73	Couleurs	Carlos Borau, Dalija Prasnikar, Daniel Nugent, Erfan Mowlaei, Jon Adams, N J, Sujith Niraikulathan
74	Couteau à beurre	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
75	Création de superposition Windows (toujours visible)	honk, mnoronha, NitZRobotKoder, Rupali, Sujith Niraikulathan

76	Création de vos propres bibliothèques pour les applications Android	EpicPandaForce, honk, mnoronha
77	Création de vues personnalisées	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
78	Créer des ROM personnalisées Android	honk, Pradumn Kumar Mahanta
79	Créer un écran de démarrage	honk, Kiran Benny Joseph, Zoe
80	Créer une classe Singleton pour le message Toast	Emad, Ishan Fernando
81	Cueilleurs de date et d'heure	adalPaRi, Brenden Kromhout, Daniel Nugent, Harish Gyanani, Ironman, Milad Nouri, RediOne1, Rohan Arora
82	Cycle de vie de l'interface utilisateur	Daniel Nugent, Dinesh Choudhary, Floern, Lewis McGeary, orelzion, R. Zagórski, Sergey Glotov
83	Dague 2	Aurasphere, Cabezas, David Medenjak, EpicPandaForce, honk, mattfred, Tomik
84	Date / heure localisée dans Android	Geert, honk, mnoronha
85	Décompresser le fichier dans Android	Arth Tilva, Daniel Nugent, mnoronha
86	Décorations RecyclerView	Barend, David Medenjak, Gabriele Mariotti, Muthukrishnan Rajendran, Peter Gordon, RamenChef, Stephen Leppik, Yasin Kaçmaz
87	Définir la valeur de pas (incrément) pour RangeSeekBar personnalisé	Romu Dizzy
88	Démarrer avec OpenGL ES 2.0+	MarGenDo
89	Des exceptions	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
90	Des préférences partagées	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
91	Dessin sur toile avec SurfaceView	davidgiga1993
92	Dessins vectoriels	Priyank Patel, ShahiM
93	Détecter l'événement Shake dans Android	N-JOY, tynn, Xiaozou
94	Détection de geste	mpkuth
95	Développement de jeux Android	Zoe
96	Dialogue	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
97	Directeur chargé d'emballage	FredMaggiowski, Hi I'm Frogatto, Muthukrishnan Rajendran, Piyush, Squidward
98	Domaine	bdash, Dan, EpicPandaForce, Hi I'm Frogatto, iurysza, null pointer, RamenChef, Stephen Leppik, sukumar
99	Doze Mode	Daniel Nugent, Fabio, honk, NitZRobotKoder, RamenChef, Rosário Pereira Fernandes, Rupali
100	Écran partagé / Activités multi-écrans	Vishal Puri
101	Écrans de support avec différentes résolutions, tailles	Eduardo, Guilherme Torres Castro, kalan, mpkuth, Onur, ppeterka

102	Ecriture des tests de l'interface utilisateur - Android	Atif Farrukh, Daniel Nugent, Gabriele Mariotti, honk, Jon Adams, originx
103	Éditer le texte	Daniel Nugent, Gabriele Mariotti, Kaushik NP, Muthukrishnan Rajendran, Rubin Nellikunnathu, Yousha Aleayoub
104	Emplacement	Alex Chengalan, Aryan, BadCash, Daniel Nugent, Hiren Patel, Mahmoud Ibrahim, MidasLefko, Pablo Baxter, RamenChef, Stephen Leppik
105	Émulateur	Ahmad Aghazadeh, Dan Hulme, fyfyone Google, honk, rekire, Rubin Nellikunnathu, ThomasThiebaud
106	Événements / Intentions du bouton matériel (PTT, LWP, etc.)	JensV
107	Exécution instantanée dans Android Studio	AndroidMechanic, Daniel Nugent, ridsatrio, Zoe
108	ExoPlayer	Hamed Gh
109	Facebook SDK pour Android	Akeshwar Jha, AndiGeeky, Community, Daniel Nugent, honk, Zarul Izham
110	Facturation dans l'application	Hussein El Feky, Pro Mode, Zoe
111	Fastjson	KeLiuyue
112	Feuilles de fond	Daniel Nugent, Gabriele Mariotti, Magesh Pandian, MiguelHincapieC, RamenChef, Stephen Leppik, sud007, Zarul Izham
113	Fichier Zip dans Android	Adnan
114	Fil	Daniel Nugent, PRIYA PARASHAR, RamenChef
115	FileIO avec Android	h22, sun-solar-arrow
116	FileProvider	Joost Verbraeken, pedros
117	Fileur	AndroidMechanic, Anonsage, Daniel Nugent, Vishwesh Jainkuniya
118	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
119	Firebase Cloud Messaging	Gabriele Mariotti, shikhar bansal, Shubham Shukla, Zarul Izham
120	Firebase Realtime DataBase	Aawaz Gyawali, drulabs, Gabriele Mariotti, honk, Md. Ali Hossain, RamenChef, Sneh Pandya, Stephen Leppik, yennsarah, Zarul Izham
121	FloatingActionButton	Ahmad Aghazadeh, Charuක, Daniel Nugent, Gabriele Mariotti, mattfred, RamenChef, Shinil M S, Stephen Leppik
122	Formatage des numéros de téléphone avec motif.	Pedro Varela
123	Fournisseur de contenu	Andrew Siplas, cdeange, Daniel Nugent, Dinesh Choudhary, Lewis McGeary, RamenChef
124	Fragments	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
125	Fresque	Alexander Oprisnik, Daniel Nugent, honk, Nilesh Singh, Zarul Izham
126	Fuite de fuite	Rakshit Nawani, tynn
127	Fuites de mémoire	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
128	Genymotion pour Android	Atef Hares, Harish Gyanani
129	Gestion des événements tactiles et animés	honk, Zoe
130	Gestionnaire de raccourcis	g4s8, Sukrit Kumar
131	Glisse	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
132	Glisser pour rafraîchir	Chirag SolankI, Daniel Nugent, Gabriele Mariotti, Malek Hijazi, RamenChef, Stephen Leppik
133	Google Maps API v2 pour 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
134	Google Play Store	dakshbhatt21, Daniel Nugent, reVerse
135	Gradle pour 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
136	GreenDAO	Allan Pereira, Carl Poole, Grundy, MiguelHincapieC, R. Zagórski, RamenChef, Stephen Leppik
137	GreenRobot EventBus	CaseyB, Daniel Nugent, Hamed Momeni, RamenChef
138	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
139	HttpURLConnection	Aleks G, Daniel Nugent, Duan Bressan, honk, KDeogharkar, marshmallow, Shantanu Paul, Simone Carletti
140	Images 9-Patch	Knossos, Nissim R, Tomik

141	ImageView	Ahmad Aghazadeh, Ali Sherafat, Chip, Daniel Nugent, DanielDiSu, Dinesh, Gabriele Mariotti, Harish Gyanani, kit, lax1089, Pratik Butani, Squidward, Sup
142	Indexation des applications Firebase	shalini, tynn
143	Installation d'applications avec ADB	Ahmad Aghazadeh, fyfyone Google, Laurel, Xaver Kapeller
144	Intégration de Google Signin sur Android	jagapathi
145	Intégration de la passerelle Android Paypal	A-Droid Tech
146	Intégrer Google Connexion	AndiGeeky, RamenChef, Tot Zam
147	Intégrer OpenCV dans Android Studio	MashukKhan, RamenChef, ssimm
148	Intention	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, noɥʎdʎzɐɔO, 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
149	Intentions implicites	Blundering Philosopher, Daniel Nugent, mnoronha, Pratik Butani, SoroushA
150	IntentService	Anax, Daniel Nugent, honk, JonasCz, TRINADH KOYA, Yashaswi Maharshi
151	Interfaces	appersiano, Brenden Kromhout, Daniel Nugent, RediOne1
152	Internationalisation et localisation (I18N et L10N)	Ankur Aggarwal
153	Jackson	KeLiuyue
154	Java sur Android	Eugen Martynov
155	JCodec	Adhikari Bishwash
156	Journalisation et utilisation de Logcat	Adam Ratzman, akshay, Alexander Mironov, alexey polusov, Anand Singh, AndroidMechanic, astuter, auval, Daniel Nugent, Eugen Martynov, faranjit, FromTheSeventhSky, gattsbr, Jeeter, Jon Adams, Laurel, LaurentY, Manan Sharma, Mario Lenci, Piyush, pRaNaY, Pratik Butani, rekire, russt, Sujith Niraikulathan, TDG, thiagolr, Yury Fedorov, Zachary David Saunders
157	JSON dans Android avec 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
158	Le contexte	Will Evers
159	Le fichier manifeste	John Snow, Jon Adams, kit, mayojava, Menasheh
160	Lecteur multimédia	Ahmad Aghazadeh, Carlos Vázquez Losada, hello_world, Makille, R. Zagórski, Redman

161	Les notifications	alexey polusov, bricklore, Da-Jin C, Daniel Nugent, Dus, gbansal, Jeeter, piotrek1543, RediOne1, Rupali, TR4Android, weston
162	ListView	A.A., brainless, Daniel Nugent, Diti, Douglas Drumond, Fabian Tamp, Gabriele Mariotti, Hiren Patel, Mr.7, Ruben Pirotte, Saeed-rz, shaonAshraf, Squidward
163	Localisation avec des ressources sous Android	AndroidMechanic, electroid, Fabio, Gubbel, Harish Gyanani, honk, Jinesh Francis, mpkuth, RamenChef, USKMobility
164	Looper	tynn
165	LruCache	Daniel Nugent, honk, LordSidious, RamenChef, Stephen Leppik
166	Manipulation de liens profonds	Doron Yakovlev-Golani, Harsh Sharma, mnoronha, Tanis.7x
167	Manutentionnaire	Daniel Nugent, Floern, Hasif Seyd, Lewis McGeary, Mike Scamell, Muhammed Refaat, Sweeper, Tomik, TR4Android
168	MediaSession	Disk Crasher, honk, KuroObi, RamenChef
169	MediaStore	Daniel Nugent, honk, RamenChef, Uttam Panchasara
170	Menu	Bhargavi Yamanuri, Chip, Daniel Nugent, Hi I'm Frogatto, honk, Iman Hamidi
171	Métriques d'affichage du périphérique	Daniel Nugent, Hiren Patel, Talha, W3hri
172	Mises en page	a.ch., Adarsh Ashok, Adinia, AesSedai101, Ahmad Aghazadeh, Aleksandar Stefanović, ankit dassor, Aurasphere, Bartek Lipinski, Björn Kechel, bjrne, Brenden Kromhout, Charuක, Dan Hulme, Daniel Nugent, devnull69, Floern, Gabriele Mariotti, Gaurav Jindal, Gurgen Hakobyan , Infinite Recursion, Kaushik NP, Knossos, Lewis McGeary, Michael Spitsin, MiguelHincapieC, Mr.7, Nepster, Patrick Dattilio, Phan Van Linh, Rajesh, rciovati, rekire, Sir SC, Sneh Pandya, Talha Mir, ThomasThiebaud, Tim Kranen, Trilarion, ubuntudroid, Vasily Kabunov, Yury Fedorov
173	Mode PorterDuff	Adarsh Ashok, AndroidMechanic, Knossos, PhilLab, S.D., Vasily Kabunov
174	Modèles de conception	Adhikari Bishwash, honk, Steve.P
175	Moshi	Blundell

176	Moyen rapide pour configurer Retrolambda sur un projet Android.	anatoli, Md. Ali Hossain
177	Multidex et la limite de méthode Dex	Adarsh Ashok, Ben, bigbaldy, cdeange, Daniel Nugent, Gabriele Mariotti, Mike, Pongpat, R. Zagórski, Shirane85
178	MVVM (Architecture)	Daniel W., RamenChef, Stephen Leppik
179	NavigationView	Adam Lear, akshay, Charuක, Daniel Nugent, Gabriele Mariotti, Kedar Tendolkar, petrumo, RamenChef, rekire, SANAT, Sevle, Stephen Leppik, sud007
180	NDK Android	Alex, astuter, Doron Yakovlev-Golani, Flayn, Onik, samgak , still_learning, Täg, thiagolr
181	Obtenir les noms de police du système et utiliser les polices	Adil Saiyad, honk
182	OkHttp	A-Droid Tech, Daniel Nugent, Gabriele Mariotti, Gubbel, noob, Rohit Arya, Vucko, Zarul Izham
183	Okio	Adhikari Bishwash
184	Optimisation des performances	honk, Jonas Köritz
185	Optimisation du noyau Android	honk, Sneh Pandya
186	ORMLite dans Android	Manos
187	Otto Event Bus	gus27, tynn
188	Outils Attributs	Dalija Prasnikar, Gabriele Mariotti, Harsh Sharma, Kayvan N, TR4Android
189	Outils de rapport d'incident	Ajit Singh, Charuක, Ekin, Gabriele Mariotti, Ishita Sinha, Jason Bourne, Madhukar Hebbar, pRaNaY
190	Pagination dans RecyclerView	Muhammad Younas
191	Pain grillé	Adam Ratzman, adao7000, Aida Isay, Amit, Andrew Brooke, AndroidMechanic, Avijit Karmakar, Bartek Lipinski, cdeange, Charuක, Daniel Nugent, Erik Ghonyan, Gabriele Mariotti, Lewis McGeary, LordSidious, Lukas, mpkuth, MrSalmon, RamenChef, Rohit Arya, Sammy T, saurav, SoroushA, sukumar, Vicky, Vucko

192	Parcelable	Alex Sullivan, Andrei T, HoseinIT, Nick Cardoso
193	Peindre	Nicolas Maltais
194	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
195	Ping ICMP	Carl Poole
196	Piste audio	Ayush Bansal
197	Planification du travail	RamenChef, reflective_mind
198	Polices Personnalisées	Daniel Nugent, Erik Ghonyan, Gabriele Mariotti, Hiren Patel, honk, kit, Nougat Lover, Simon Schubert, Stanojkovic, Sujith Niraikulathan
199	Port Mapping en utilisant la bibliothèque Cling dans Android	Shinil M S
200	Processeur d'annotation	krishan
201	Programmation Android avec Kotlin	Gian Patrick Quintana, Govinda Paliwal, Oknesif, Zarul Izham
202	ProGuard - Obscurcir et réduire votre code	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
203	Publier sur Play Store	Carlos Borau, Fabio, mnoronha, Zoe
204	Publier un fichier .aar sur Apache Archiva avec Gradle	Marian Klühspies
205	Publier une bibliothèque dans les référentiels Maven	Farid
206	Qu'est-ce que ProGuard? Qu'est-ce que l'utilisation dans Android?	Ayush Bansal, Daniel Nugent, Ghanshyam Sharma, Pratik Butani
207	Rapport d'incident de	AndiGeeky, Gabriele Mariotti, honk, RamenChef, Stephen

	Firebase	Leppik, Zarul Izham
208	RechercherView	Daniel Nugent, Dmide, Hiren Patel, RamenChef, Stephen Leppik, sud007
209	Reconnaissance d'activité	Pablo Baxter
210	RecyclerView	Abhishek Jain, Abilash, Adinia, Ahmad Aghazadeh, Akash Patel, Alex Bonel, Alok Omkar, anatoli, Andrii Abramov, AndroidMechanic, Anirudh Sharma, BalaramNayak, Barend, Bartek Lipinski, Bryan, cascal, Charuज, Chirag SolankI, Daniel Nugent, Fahad Al-malki, Felix Edelmann, FromTheSeventhSky, Gabriele Mariotti, GensaGames, humazed, Ironman, Jacob, jgm, Joel Mathew, Jon Adams, Joshua, Kayvan N, keineantwort, Kevin DiTraglia, Knossos , kyp, MathaN, MidasLefko, MKJParekh, mklimek, Pablo Baxter, Patrick Dattilio, Piyush, raktale, RamenChef, rciovati, Reaz Murshed, Rohan Arora, Sagar Chavada, Sanket Berde, Sasank Sunkavalli, Sneh Pandya, Stephen Leppik, sukumar, Sunday G Akinsete, thetonrifles, Tot Zam , Uttam Panchasara, V. Kalyuzhnyu, Vasily Kabunov, Xaver Kapeller, Yasin Kaçmaz, Yura Ivanov, Yury Fedorov, Zilk
211	RecyclerView et LayoutManagers	4444, BalaramNayak, Felix Edelmann, Gabriele Mariotti, Kayvan N, MidasLefko, RamenChef, Stephen Leppik
212	RecyclerView onClickListeners	abhishesh, Braj Bhushan Singh, Bryan, FromTheSeventhSky, fuwaneko, Gabriele Mariotti, honk, RamenChef, Smit.Satodia, Stephen Leppik
213	RenderScript	Ankit Popli, Dalija Prasnikar, Froyo, honk, Rucha Bhatt, Xaver Kapeller
214	Ressources	biddulph.r, Brenden Kromhout, Charuज, 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
215	Retrofit2	Adarsh Ashok, Adnan, Anderson K, AndroidMechanic, AndyRoid, aquib23, arcticwhite, CaseyB, Cassio Landim, Dan, Daniel Nugent, DanielDiSu, devnull69, Dhaval Solanki, FiN, Greg T, Kamran Ahmed, KATHYxx, Kaushik, mrtuovinen, NashHorn, Omar Al Halabi, param, Pavneet_Singh, Pinaki Acharya, R. Zagórski, RamenChef, SKen, Sneh Pandya, Stephen Leppik, xdk78, Zarul Izham

216	Retrofit2 avec RxJava	Anand Singh, gaara87, GurpreetSK95, Lukas, mrtuovinen, R. Zagórski, Zarul Izham
217	RoboGuice	AndroidRuntimeException, Lewis McGeary, Rajesh
218	Robolectric	Blundell, g4s8
219	Secure SharedPreferences	Christlin Joseph
220	Sécurité	xDragonZ
221	SensorManager	honk, Simon, TDG
222	shell adb	3VYZkz7t, Ahmad Aghazadeh, auval, Burak Day, Fabio, fyfyone Google, Hannoun Yassir, Natali, Pavel Durov, R. Zagórski, sukumar, Yury Fedorov
223	Signez votre application Android pour publication	Gabriele Mariotti, M M
224	Snackbar	AndroidRuntimeException, Charuක, Daniel Nugent, Gabriele Mariotti, Harsh Pandey, Jinesh Francis, Lithimlin, marshmallow, Mike Scamell, miss C, Mochamad Taufik Hidayat, Patrick Dattilio, Piyush, RamenChef, Rasoul Miri, Rosário Pereira Fernandes, Sneh Pandya, Stephen Leppik , Zarul Izham
225	Son et média Android	johnrao07, Muhammad Umair Shafique, Squidward
226	SpannableString	S.R
227	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
228	Stockage de fichiers dans le stockage interne et externe	Amit Vaghela, Andrew Brooke, AnV, Daniel Nugent, Gabriele Mariotti, Nickan B, Uttam Panchasara
229	Stratégie de mode strict: outil permettant de détecter le bogue lors de la compilation.	Shekhar

230	Studio Android	AndroidMechanic, auval, Blackbelt, Charuක, Daniel Nugent, Gabriele Mariotti, Hiren Patel, Inzimam Tariq IT, Jon Adams, N J, Phan Van Linh, R. Zagórski, Squidward, Sujith Niraikulathan, ThomasThiebaud
231	SyncAdapter avec régulièrement synchroniser les données	Bhargavi Yamanuri
232	Synchronisation des données avec l'adaptateur de synchronisation	Arpit Gandhi, mnoronha
233	TabLayout	Daniel Nugent, Willie Chalmers III
234	Temps Utils	Burhanuddin Rashid, Mukesh Kumar Swami, Muthukrishnan Rajendran
235	TensorFlow	Pratik Butani
236	Test d'interface utilisateur inter-app avec UIAutomator	Timo Bähr
237	Test de l'interface utilisateur avec Espresso	Daniel Nugent, Gabriele Mariotti, Jason Robinson, Michael Vescovo, Milad Faridnia, N J, RamenChef, Víctor Albertos
238	Tests unitaires sous Android avec JUnit	abhi, Andre Perkins, AndroidMechanic, Eugen Martynov, honk, Lewis McGeary, N J, Namnodorel, Patrick Dattilio, Rolf ツ
239	Text to Speech (TTS)	Ahmad Aghazadeh, honk, Jordan, Lukas, nibarius, Peter Taylor, RamenChef, Stephen Leppik
240	TextInputLayout	Adarsh Ashok, BrickTop, Gabriele Mariotti, Hi I'm Frogatto, RamenChef, Shashanth, Sneh Pandya, Stephen Leppik
241	Thème DayNight (AppCompat v23.2 / API 14+)	Ishita Sinha
242	Thème, Style, 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
243	Tiroirs	alanv, B001, Daniel Nugent, Greg T, Hiren Patel, Jinesh

		Francis, Nick Cardoso, TR4Android
244	Touch Events	Yvette Colomb
245	TransitionDrawable	S.R, Yogesh Umesh Vaity
246	Transitions d'éléments partagés	noongiya95
247	Un service	adao7000, AndroidMechanic, Apoorv Parmar, BadCash, B-GangsteR, Daniel Nugent, g4s8, Hiren Patel, JonasCz, Lazai, Lucas Paolillo, Michael Spitsin, Nougat Lover, rakeshdas, Vinícius Barros
248	URL de rappel	Atif Farrukh, honk, RamenChef, Stephen Leppik
249	Validation du courrier électronique	Hiren Patel, honk, iravul, Nicolas Maltais
250	VectorDrawable et AnimatedVectorDrawable	Ahmad Aghazadeh, Aleksandar Stefanović, gaara87, honk, Lewis McGeary, RamenChef, Stephen Leppik
251	Vérifier la connexion de données	sukumar, Suresh Kumar
252	Vérifiez la connectivité Internet	AndiGeeky, Bill, Daniel Nugent, gbansal, Ichigo Kurosaki, Jon Adams, sukumar, TameHog, Yousha Aleayoub
253	Versions Android	4444, AndroidMechanic, athor, BooleanCheese, Dalija Prasnikar, Daniel Nugent, Fildor, Gabriele Mariotti, H. Pauwelyn, Matt, RediOne1, tynn
254	Versions du SDK de projet	Arnav M., Ranveer, Tanis.7x
255	Vibration	cdeange, Zerntrino
256	VideoView	iravul, Sashabrava
257	VideoView optimisé	Chip
258	ViewFlipper	Anita Kunjir, Daniel Nugent, honk
259	ViewPager	Adarsh Ashok, Adrián Pérez, Daniel Nugent, Gabriele Mariotti, Moustachauve, RamenChef, RediOne1, Rucha Bhatt, Sneh Pandya, Stephen Leppik, Usman, ZeroOne
260	voie rapide	Gokhan Arik
261	Voir Calculs Dimensions	mnoronha, stkent

262	Volée	2943, Ankur Aggarwal, Endzeit, Harsh Dalwadi, herrmartell , honk, Jon Adams, Pablo Baxter, RamenChef, Rubin Nellikunnathu, Rucha Bhatt, sameera lakshitha, Stephen Leppik, VISHWANATH N P
263	WebView	Amod Gokhale, Daniel Nugent, g4s8, j2ko, jasonlam604, JonasCz, Mohammad Yahia, ppeterka, Prakash Bala, shtolik, Squidward, Sukrit Kumar, sukumar
264	Widgets	4444, Alex Ershov, Daniel Nugent, Don Chakkappan, Imdad, nenofite, sun-solar-arrow
265	XMPP s'inscrire à la session et à chatter exemple simple	4444, RamenChef, Saveen
266	Xposed	Medusalix
267	Youtube-API	abhishesh, Giannis, honk, MashukKhan, Zarul Izham, Zeeshan Shabbir