Java Avancé – Java pour terminaux mobiles

Emmanuel Conchon
(emmanuel.conchon@unilim.fr)

Objectifs et modalités

- Présentation des technologies Java dans un environnement mobile
  - Focus sur Android
- Développement d'applications interactives et sécurisées
- Répartition
  - 12h CM (3h en présentiel, le reste à distance), 18h TP
- Mode d'évaluation
  - Session 1: 50% TP-Projet + 50% Examen écrit
  - Session 2: 100% Examen écrit

La plateforme Android

Bibliographie

- L'essentiel du cours s'appuie sur http://developer.android.com/guide

Qu'est ce qu'Android

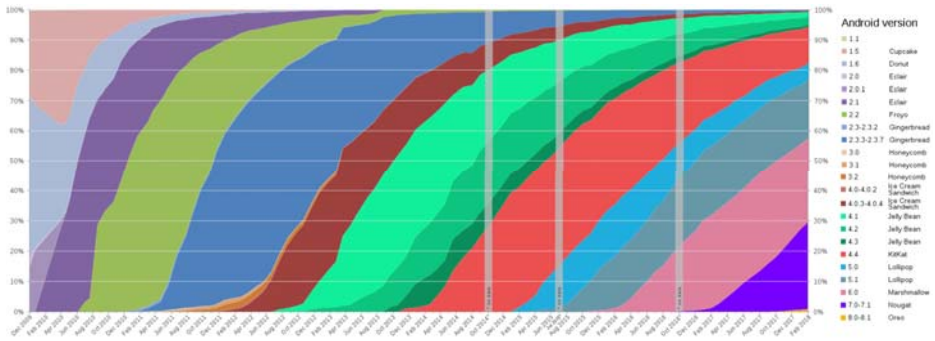
- Système d'exploitation mobile développé et maintenu par Google
  - Fonctionne sur une base linux au niveau noyau
  - Basé sur Java pour le développement d'applications
- Système mobile numéro 1 dans le monde en Aout 2020
  - Il représente 70,74% du marché des systèmes d'exploitation mobiles (source: netmarketshare.com)
- Disponible pour les téléphones mais aussi pour
  - les TV connectées (AndroidTV)
  - les voitures (Android Auto)
  - les montres connectés (AndroidWear)
  - etc.

Qu'est ce qu'Android

- Android est un nom générique qui regroupe de très nombreuses versions connues chacune sous un nom de code différent
  - Cupcake (1.5)
  - Donut (1.6)
  - Eclair (2.0–2.1)
  - Froyo (2.2–2.2.3)
  - Gingerbread (2.3–2.3.7)
  - Honeycomb (3.0–3.2.6)
  - Ice Cream Sandwich (4.0–4.0.4)
  - Jelly Bean (4.1–4.3.1)
  - KitKat (4.4–4.4.4, 4.4W–4.4W.2)
  - Lollipop (5.x)
  - Marshmallow (6.x)
  - Nougat (7.x)
  - Oreo (8.x)
  - Pie (9.x)
  - Q (10.x)

Qu'est ce qu'Android

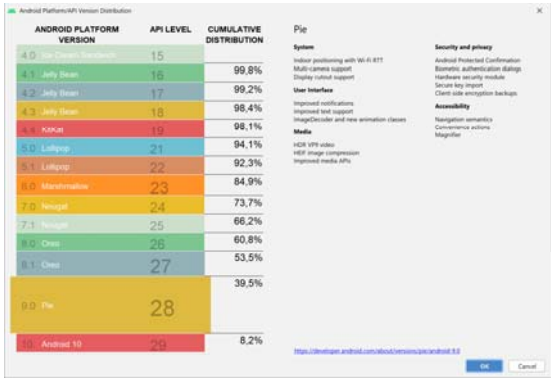
- Fragmentation d'Android en Février 2018



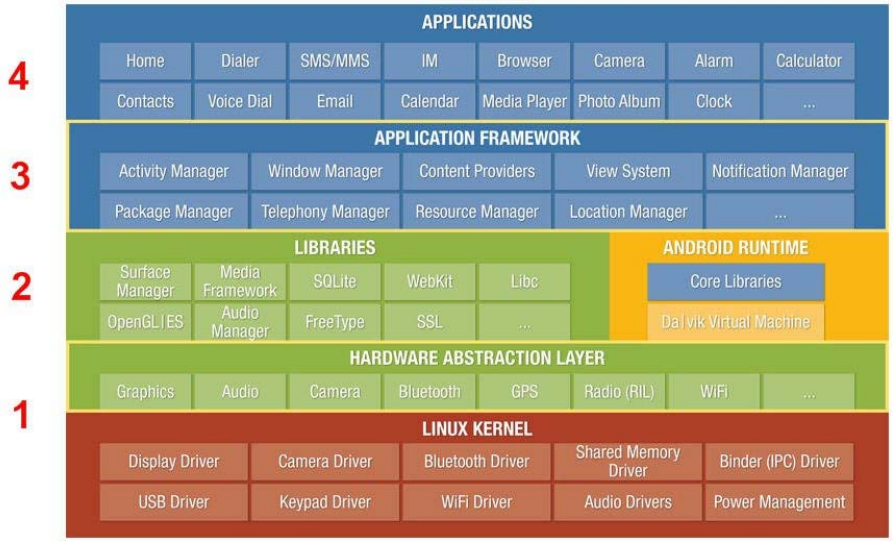
Souce:Wikipedia

Qu'est ce qu'Android

- Répartition des versions d'Android en 2020
  - Cette fragmentation peut rendre complexe le déploiement d'applications
- Chaque version d'Android propose une API différente
  - On parle de niveau d'API
  - Il y a une compatibilité ascendante entre les niveaux



Architecture globale

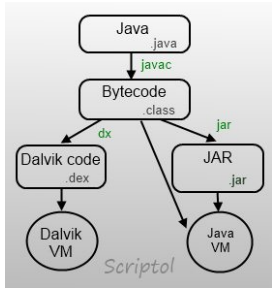


Architecture globale

- Repose sur un noyau Linux 2.6 pour
  - La gestion des périphériques
  - La gestion de la mémoire, des processus…
  - La sécurité
- Fournit un lot d'application standards (navigateur web, contacts, téléphone, bureau…)
- Repose sur une machine virtuelle de type Java avec les bibliothèques associées
  - Les applications se développent en langage Java
  - Bibliothèque proche de l'édition standard de Java à l'exception des bibliothèques graphiques (Swing et AWT)
  - Jusqu'à Android 5.0 machine virtuelle Dalvik, ART depuis

La machine virtuelle : Dalvik puis ART

- Dalvik est une machine virtuelle JIT (Just In Time)
  - Le bytecode est compilé en temps réel et produit un .dex à chaque lancement d'une application
  - Utilise une librairie Java basée sur la librairie Harmony de la fondation Apache

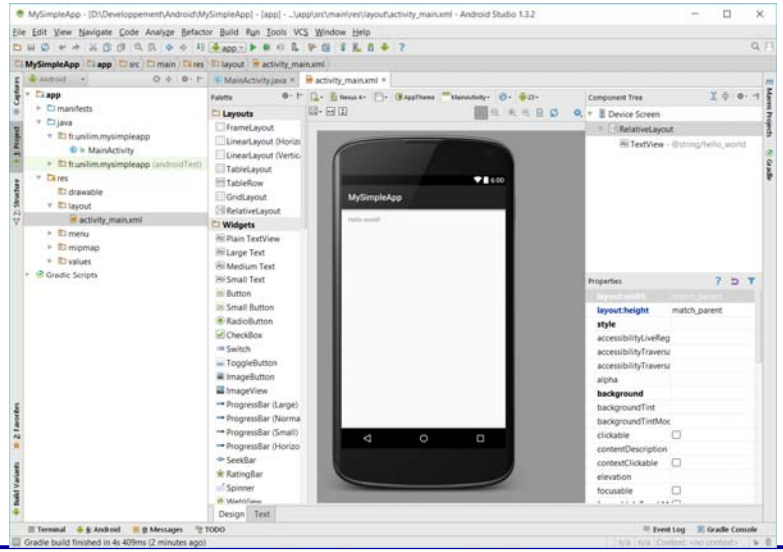


- ART (Android RunTime) utilise un compilateur AOT (Ahead of Time)
  - Traduction du Bytecode en .dex au moment de l'installation d'une application

Le développement Android

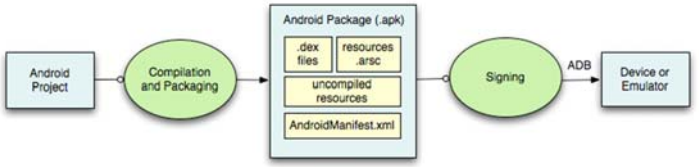
- Repose sur Android SDK (multi-plateformes) qui offre plusieurs outils
  - Un environnement de développement intégré (IDE): Android Studio
    - Eclipse + le plugin ADT (Android Development Tools) jusqu'en 2014
  - Les librairies du langage, la documentation, des exemples de codes et un tutoriel
  - Un débuggeur
  - Un émulateur basé sur QEmu : AVD (Android Virtual Device)
  - Android Debug Bridge
    - Suite d'outils en ligne de commande qui permette de dialoguer avec un émulateur ou avec un terminal connecté à l'ordinateur
- Nécessite d'avoir installé au préalable un JDK (Java Development Kit) sur la machine

IDE pour les TPs: IntelliJ IDEA

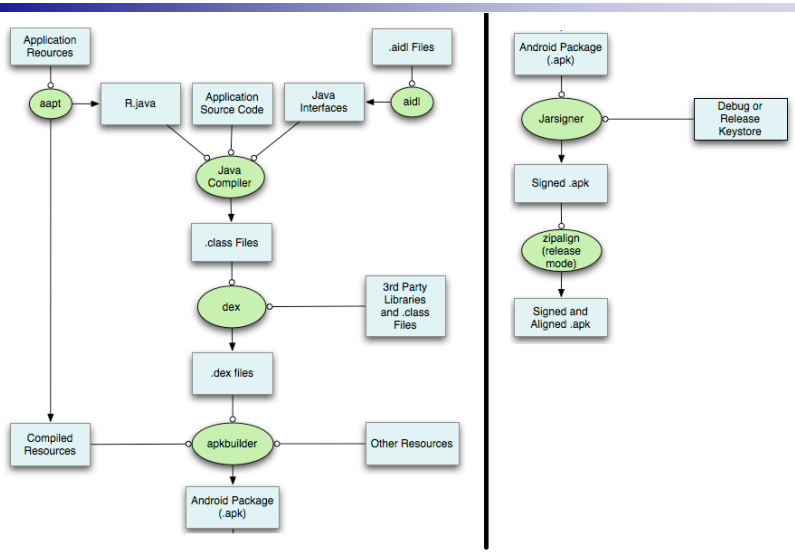


Le développement Android

- Compilation et déploiement d'une application



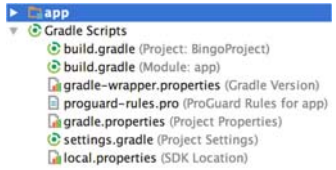
Processus de compilation d'une application Android



Gradle

- Moteur de production de référence pour Android
  - Alternative à Maeven, Ant (il reprend le meilleur des deux)
  - Permet de construire des programmes en ramenant et en compilant les librairies nécessaires de manière automatisée
  - Propose un système de gestion de dépendances pour les librairies externes

- Sur Android, un projet Gradle repose sur deux fichiers
  - local.properties qui définit les emplacements des SDK
  - build.gradle qui va contenir la description des dépendances
    - En pratique 2 fichiers, un général et un spécifique au projet



Gradle

- Exemple de fichier build.gradle

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

android {
    compileSdkVersion 23
    buildToolsVersion '25.0.0'

    defaultConfig {
        applicationId "fr.unilim.mysimpleapp"
        minSdkVersion 19
        targetSdkVersion 23
        versionCode 1
        versionName "1.0"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}

dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile 'com.android.support:appcompat-v7:23.0.1'
}
```


Exécution d'applications Android

- Le système Android repose sur le principe du sandboxing
  - Chaque application s'exécute dans son bac à sable sécurisé
  - Le système Android gère chaque application comme un utilisateur Linux différent
  - Chaque application dispose d'un identifiant unique permettant de gérer des permissions différentes d'une application à l'autre
  - Principe du moindre privilège
- Une application se lance dans un processus dédié
  - Le système lance le processus dès qu'un composant de l'application a besoin d'être exécuté
  - De la même manière c'est le système qui gère l'arrêt des processus et la libération de la mémoire
- Chaque processus utilise sa propre VM
  - Isolation de l'exécution du code d'une application

Exécution d'applications Android

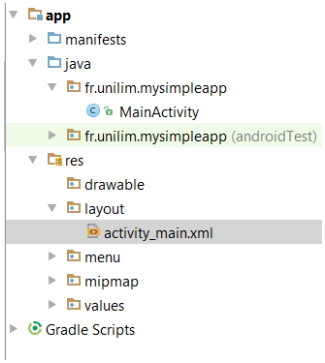
- Il est tout de même possible de partager des informations entre applications en affectant le même identifiant à deux applications
  - Dans ce cas elles partagent la même VM et donc le même processus
  - Ce n'est possible que si les deux applications sont signées par le même certificat
- Une autre solution de partage de ressources et la possibilité pour l'application de demander des permissions complémentaires lors de l'installation
  - Doivent être acceptées par l'utilisateur
  - Avant Android 6
    - Acceptation ou refus total
      - ★ *Pas de panachage possible*
  - Depuis Android 6
    - Révocation possible
    - Les permissions sont vérifiées à chaque lancement

Les principaux composants Android

- Une applications Android peut disposer de plusieurs types de composants
  - Des activités
    - Correspond à un écran de l'application
  - Des services
    - Composant fonctionnant en tache de fond sans interfaces graphique
  - Des content providers
    - Permettent le partage d'informations depuis l'application
      - ★ *Exemple: l'application de contact Android fournit un content provider*
  - Des broadcast receivers
    - Permettent de réagir à des sollicitations du système
      - ★ *Exemple: Réagir lorsque l'écran se verrouille ou lorsque la batterie est faible*
- Chaque application Android peut faire appel à un composant d'une autre application
  - Evite de redévelopper des fonctionnalités existantes (Prise de photo…)

Structure d'un projet Android

- **AndroidManifest.xml**
  - Fichier de configuration générale du projet
- **src/ ou java/…**
  - Stockage du code source en Java de l'application
- **res/**…: *Fichiers de ressources*
  - drawable/ = images
  - layout/ = descriptions des interfaces graphiques
  - menu/ = options des menus de l'application
  - values/ = stockage des constantes
    - strings.xml = données de localisation (internationalisation)
    - Styles.xml = apparence générale de l'application



Le fichier de Manifeste

- Le fichier AndroidManifest.xml permet de définir le nom et les composants de l'application
  - Liste des activités impliquées et de leurs points d'entrée
  - Liste des permissions nécessaires au fonctionnement de l'application
    - Accès internet
    - NFC
    - Accès aux contacts du téléphone
    - …
  - Définit la version minimum d'Android nécessaire au bon fonctionnement de l'application
  - Définit les fonctionnalités matérielles requises
    - Appareil photo
    - Bluetooth
    - …

Exemple de fichier Manifeste

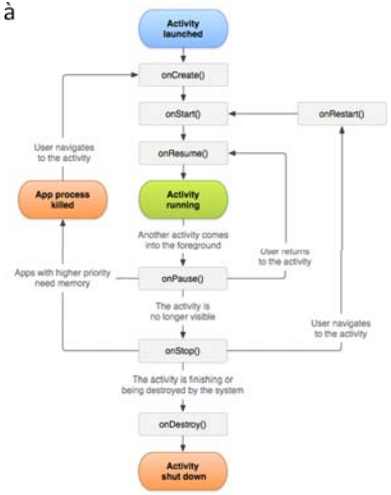
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.isis.imageflux"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="17"
        android:targetSdkVersion="19" />

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

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_mylauncher"
        android:label="MyProject"
        android:largeHeap="true"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
             android:configChanges="orientation|keyboardHidden|screenSize"
            android:label="Imageflux"
            android:windowSoftInputMode="stateHidden" >
            <meta-data
                android:name="android.support.PARENT_ACTIVITY"
                android:value="com.isis.imageflux.MainActivity" />
            <!-- android:theme="@style/FullscreenTheme" > -->
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
```


Le cœur de l'application: les activités (Activity)

- Une application Android se compose d'une ou plusieurs activités (Activity)
  - Les activités sont relativement indépendantes les unes des autres
  - Chaque activité correspond à un écran de l'application
    - Pour chaque activité il peut y avoir une interface utilisateur différentes spécifiée dans un layout
- Une activité doit hériter d'une classe prédéfinie
  - **AppCompatActivity** depuis l'API 22.1
  - ActionBarActivity sur les versions précédentes
- Exemple: Dans le cas d'une application de gestion de mails, il y aura au moins trois activités
  - Une activité principale avec la liste des mails
  - Une activité pour lire les mails
  - Une activité pour écrire les mails

Le cœur de l'application: les activités (Activity)

- Une activité fonctionne suivant une machine à états
  - 5 états
    - Starting: en train de démarrer
    - Running: en fonctionnement
    - Paused: Obscurcie ou hors champs mais pas arrêtée
    - Stopped: Arrêtée mais toujours ne mémoire
    - Destroyed: Eteinte et plus en mémoire
  - On passe d'un état à un autre à l'aide de transitions représentées par des évènements
    - On peut réagir à ces évènements dans le code dans des méthodes dédiée
      - ★ *onCreate(), on Pause(), onResume(), onStop(), onDestroy()…*



Le cœur de l'application: les activités (Activity)

- La méthode onCreate()
  - Joue un rôle proche de celui d'un constructeur Java
  - Permet d'initialiser l'objet Activity, de charger les ressources nécessaires
    - Layout
    - Images
    - Menus
    - …
  - A la fin de cette méthode, l'objet Activity existe
  - Exemple:

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);      // Toujours appeler super
        setContentView(R.layout.activity_main); // Définit le layout à utiliser
    }
```


Le cœur de l'application: les activités (Activity)

- La méthode onPause ()
  - Est appelé lorsque l'activité est encore partiellement visible
  - Cette pause peut être temporaire ou la première étape d'un arrêt définitif
- L'objectif de cette méthode est de se préparer à un éventuel arrêt
  - En arrêtant des animations
  - En sauvegardant l'état courant de l'activité
  - En libérant les ressources qui consomme de la batterie
  - …

Le cœur de l'application: les activités (Activity)

- La méthode onResume()
  - Est appelée pour sortir d'un état de pause
  - Cette méthode est également appelée lors du premier lancement d'une activité
- Elle permet
  - D'initialiser les ressources qui devront être libérer par onPause()
  - De démarrer les animations et/ou les actions à effectuer lorsque l'activité devient visible

Le cœur de l'application: les activités (Activity)

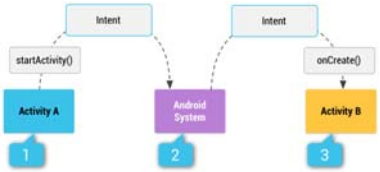
- La méthode onStop()
  - Est appelée lorsqu'une activité n'est plus visible à l'écran
    - Lorsque l'utilisateur lance une nouvelle activité de l'application
    - Lorsque l'utilisateur change d'application
    - Lorsqu'une autre application prend la main (appel par exemple)
- L'application peut continuer à fonctionner mais l'activité se termine
- Cette méthode est toujours appelée après un appel à onResume()
  - On effectue les opérations lourdes dans cette méthode
    - Sauvegarde dans une base de donnée par exemple

Le cœur de l'application: les activités (Activity)

- La méthode onStart()
  - Est appelée lors du premier démarrage d'une activité (et uniquement lors de celui-ci) puis après chaque appel à onRestart()
- La méthode onRestart()
  - Est appelée lorsqu'une activité a été stoppée et est rappelée plus tard
  - L'objectif est de ré-allouer toutes les ressources libérées par onStop()
  - Méthode assez peu utilisée, il vaut mieux lui préférer onResume()

Le cœur de l'application: les Intents et Intents Filters

- Une activité est atteignable à l'aide d'un point d'entrée (Intent)
  - Elle peut en avoir plusieurs
- Un Intent et un objet message utilisé pour demander une action à un composant Android comme une activité par exemple
  - Une activité peut en déclencher une autre à l'aide d'un Intent explicite en fournissant le nom de l'activité à lancer (le nom de la classe)
  - Il existe également des Intents implicites auxquels n'est fournit qu'une description de l'action à effectuer
    - Le système se charge de trouver l'activité permettant de réaliser l'action



Les interfaces graphiques

- Dans une application Android les interfaces graphiques sont décrites à l'aide de fichiers XML
  - Stockés dans le répertoire Layouts du projet
- L'utilisation des interfaces repose sur la notion de programmation évènementielle
  - Un évènement est un stimulus auquel votre programme doit répondre
  - Repose sur la notion de Listener d'évènements
    - Un Listener définit le comportement à avoir lorsqu'un stimulus survient
    - Un Listener est attaché à un widget de l'interface graphique
  - Technique très répandue pour la programmation d'interfaces graphiques

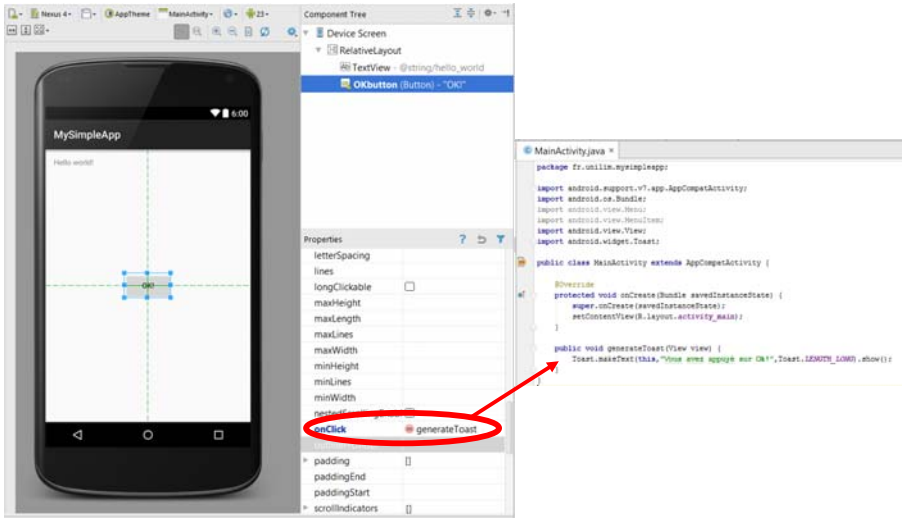
Exemple simple

- Dans cet exemple, nous allons créer une application dotée d'un bouton situé au milieu de la fenêtre
  - Lorsque l'on clique sur le bouton, un message apparait
- Manifeste:

AndroidManifest.xml

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

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="MySimpleApp"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            android:label="MySimpleApp" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```


Exemple simple



Exemple simple

activity_main.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:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    android:paddingBottom="16dp" tools:context=".MainActivity">

    <TextView android:text="@string/hello_world" android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="OK!"
        android:id="@+id/OKbutton"
        android:layout_centerVertical="true"
        android:layout_centerHorizontal="true"
        android:onClick="generateToast" />
</RelativeLayout>
```


Exemple simple

- Lorsque l'on appuie sur ok, un message (Toast) s'affiche puis disparait