



Développement Android

Partie 1

Plan

- Généralités
- La plateforme Android
- L'architecture Android
- Les outils de développement
 - Interface et ressources
 - Applications et activités
 - SQLite



Évaluation

- ▶ **Volume horaire :**
 - ▶ 12h cours (3 séances)
 - ▶ 16h TP (4 séances de TP * 2 groupes)
- ▶ **Modules d'évaluation :**
 - ▶ Évaluation écrite (Examen, Contrôle Continu, Exposés, Rapport)
 - ▶ Évaluation pratique (TP, Activités pratiques)

Développement Mobile

- ▶ Dans le monde de mobile y a trois manière de développer une application :
 - ▶ Développer une application en Natif
 - ▶ Développer une WebApp (qui correspond à une application web)
 - ▶ Développer en Hybride

Applications natives

- **Une application native** est conçue spécialement pour les appareils mobiles et dans un langage de programmation spécifique à un système d'exploitation



Applications natives

▶ Inconvénients

- ▶ **Coûts de développement** (*reprogrammer l'application pour chaque OS*)
- ▶ **Maintenance**

▶ Avantages

- ▶ **Applications performantes**
- ▶ **Expérience utilisateur**
- ▶ **Accès plus facile** aux fonctionnalités relatives aux différents composants de l'appareil (*GPS, micro, appareil photo, accéléromètre ...*)
- ▶ **Utilisable hors-ligne** (*pas besoins forcément accès à internet*)
- ▶ **Directement téléchargeable** « *Google Play Store ou App store* »
- ▶ **Notifications** (*alerter vos utilisateurs et d'attirer leur attention chaque fois que vous le souhaitez, que ce soit pour du nouveau contenu ou une offre promotionnelle*)
- ▶ **Material design** (*développer spécifiquement pour chaque OS, adapter ton design selon l'expérience utilisateur*)

Applications web (*Webapp*)

- ▶ **Une application web** est une application mobile exécutable via le navigateur internet dans l'appareil mobile et les tablettes
- ▶ **Avantages :**
 - ▶ **Entièrement en HTML, CSS et JavaScript**, donc un apprentissage du code bien moins long et coûteux que pour les applications natives.
 - ▶ **Pas d'installation** (*pas de stockage, gain de mémoire, ...*)
 - ▶ **Faible coûts** de développement (*une application web coûte jusqu'à trois fois moins cher qu'une application native*)
- ▶ **Inconvénients :**
 - ▶ Besoins « toujours » **d'internet**
 - ▶ **Accès impossible au disque** de l'utilisateur depuis un navigateur (*pour des raisons de sécurités*)
 - ▶ **Plus lente et moins adaptée**
 - ▶ **Fidélisation presque absente** : pas de store

Applications hybrides (*cross plateforme*)

- ▶ Le développement hybride est un mélange des deux premiers types d'applications.
- ▶ Développer une application en une seule fois et sera disponible sur l'ensemble des plateformes existantes
- ▶ **Avantages :**
 - ▶ peuvent constituer une **alternative intéressante** aux applications natives (*plus facile et plus rapide à développer qu'une application native*)
 - ▶ **Coûts de développement** (*moins coûteux par rapport au natives*)
 - ▶ **Maintenance** (*plus facile puisqu'il n'y a qu'une seule version à revoir pour plusieurs plateformes.*)
 - ▶ **téléchargeable**
- ▶ **Inconvénients :**
 - ▶ **Expérience utilisateur** (spécificité de chaque OS)
 - ▶ **Moins performantes** que les natives (*moins bonnes et moins stables*)
 - ▶ **Une ergonomie** pas toujours optimisée (*le système est moins adapté à chaque plateforme*)

Librairies et Frameworks pour le développement mobile

HTML5 - CSS3 - javascript

Typescript

Librairies :



Frameworks :



Frameworks



RhoMobile est un framework basé sur le langage Ruby qui permet de créer des applications natives qui seront compatibles avec un grand nombre d'OS, notamment Android, iOS, Windows Mobile, Symbian, etc.



Apache Cordova ou plus anciennement Apache Callback ou PhoneGap, est un framework open-source développé par la Fondation Apache. Il permet de créer des applications pour différentes plateformes en HTML, CSS et JavaScript



Ionic est un framework open-source créé en 2013 par Max Lynch, Ben Sperry, et Adam Bradley. Deux versions distinctes sont disponibles, incompatibles entre elles : la première version, 1.3.3, se base sur AngularJS 1.5.3 tandis que la version 3.5.0 se base sur Angular 4.1.3 et TypeScript

Frameworks



Onsen UI est un framework d'interface utilisateur open-source et des composants pour le développement d'applications mobiles hybrides HTML5, basés sur PhoneGap / Cordova. Il permet aux développeurs de créer des applications mobiles utilisant des technologies Web telles que CSS, HTML5 et JavaScript



Appcelerator Titanium est un kit de construction de logiciels destinés aux téléphones mobiles utilisant Android ou iOS, distribué par Appcelerator Inc depuis décembre 2008



Xamarin permet de développer une application multiplateforme en partageant toute une couche business et fonctionnelle. Les interfaces graphiques restent par contre propres à chaque plateforme. Tous les développements se font en C#,

Gestionnaires de bibliothèques

- ▶ Les gestionnaires de bibliothèques sont très pratiques pour gérer toutes vos bibliothèques de manière propre et organisée.
- ▶ Les plus populaires sont **Cocoapods** sur **iOS**, et le plugin **Gradle** pour Android Studio sur **Android**.



Plateformes de développement d'applications web et hybrides



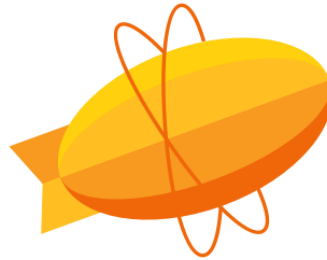
Plateformes de développement d'applications web et hybrides permettant de créer des applications sans aucune connaissance en programmation

Outils pour des applications web et hybrides



Outils de création de wireframes qui peut servir à designer vos applications

Outils pour des applications web et hybrides



ZEPLIN

Il s'agit d'un excellent outil de collaboration entre webdesigners et développeurs qui vous permettront de façonner des applications exactement telles que vous vous les imaginez

Outils de développement



Intelij est un outil pour la programmation d'applications Android développé par JetBrains



Android Studio est un environnement de développement pour développer des applications mobiles Android. Il est basé sur IntelliJ IDEA et utilise le moteur de production Gradle.



Corona est un kit de développement logiciel développé par Corona Labs Inc. au milieu de 2009, qui permet aux programmeurs de logiciels de créer des applications mobiles 2D pour iOS, Android et Kindle, des applications de bureau pour Windows et OS X et des applications de télévision connectée pour Apple TV, Fire



Xcode est un environnement de développement pour macOS, ainsi que pour iOS, watchOS et tvOS. L'API Cocoa permet de programmer avec les langages suivants : Objective-C Ruby Swift



Android

Android - Présentation



- Android est un système d'exploitation open source basé sur le noyau Linux, conçu pour smartphones, PDA, tablettes tactiles: systèmes légers.
- Utilisé, après, dans les objets connectés, glass, gear ...
- née en 2004 et rachetée par Google en août 2005,
- 5 novembre 2007, création de l'OHA (Open Handset Alliance)
= un consortium créé à l'initiative de Google réunissant des entreprises opérateurs mobiles, constructeurs et éditeurs logiciels
- But de l'OHA est de favoriser l'innovation sur les appareils mobiles en fournissant une plate-forme véritablement ouverte, complète et gratuite.

Android et Google

- Conçu pour intégrer des applications Google, Google Maps, Google Agenda, YouTube et la géolocalisation, ...
- Les différentes versions ont des noms de dessert (qui suivent l'ordre alphabétique, de A à Z, depuis la sortie de la version 1.5) qui sont sculpté et affiché devant le siège social de Google (Mountain View)



Android versions



ANDROID
1.5
CUPCAKE



ANDROID
1.6
DONUT



ANDROID
2.0-2.1
ECLAIR



ANDROID
2.6.32
FROYO



ANDROID
2.6.35
GINGERBREAD



ANDROID
2.6.36
HONEYCOMB



ANDROID
4.0-4.0.4
ICE CREAM
SANDWICH



ANDROID
4.1-4.3.1
JELLY BEAN



ANDROID
4.4-4.4.4
KITKAT



ANDROID
5.0-5.1.1
LOLLIPOP



ANDROID
6.0-6.0.1
MARSHMALLOW



ANDROID
7.0-7.1.2
NOUGAT



ANDROID
8.0-8.1
OREO



ANDROID
9.0
PIE



10: Android Q

Android - Smartphone

- Android tire partie des particularité des smartphones :
 - Interface homme machine adaptée (tactile, widget)
 - Divers modes : vibreur, sonnerie, silencieux, alarme
 - Notifications (d'applications, d'emails, de SMS, d'appels en instance)
 - De boussole, accéléromètre, GPS
 - Divers capteurs (gyroscope, gravité, accélération linéaire, baromètre)
 - NFC, RFID
 - Téléphonie (GSM) et réseau 3G, ...
- En plus de ce qu'on peut avoir sur un ordinateur : navigateur, bibliothèques graphiques 2D, 3D (Open GL), base de données (SQLite), applications de rendu multimédia (audio, vidéo, image) de divers formats, réseau Bluetooth et Wi-Fi, caméra.

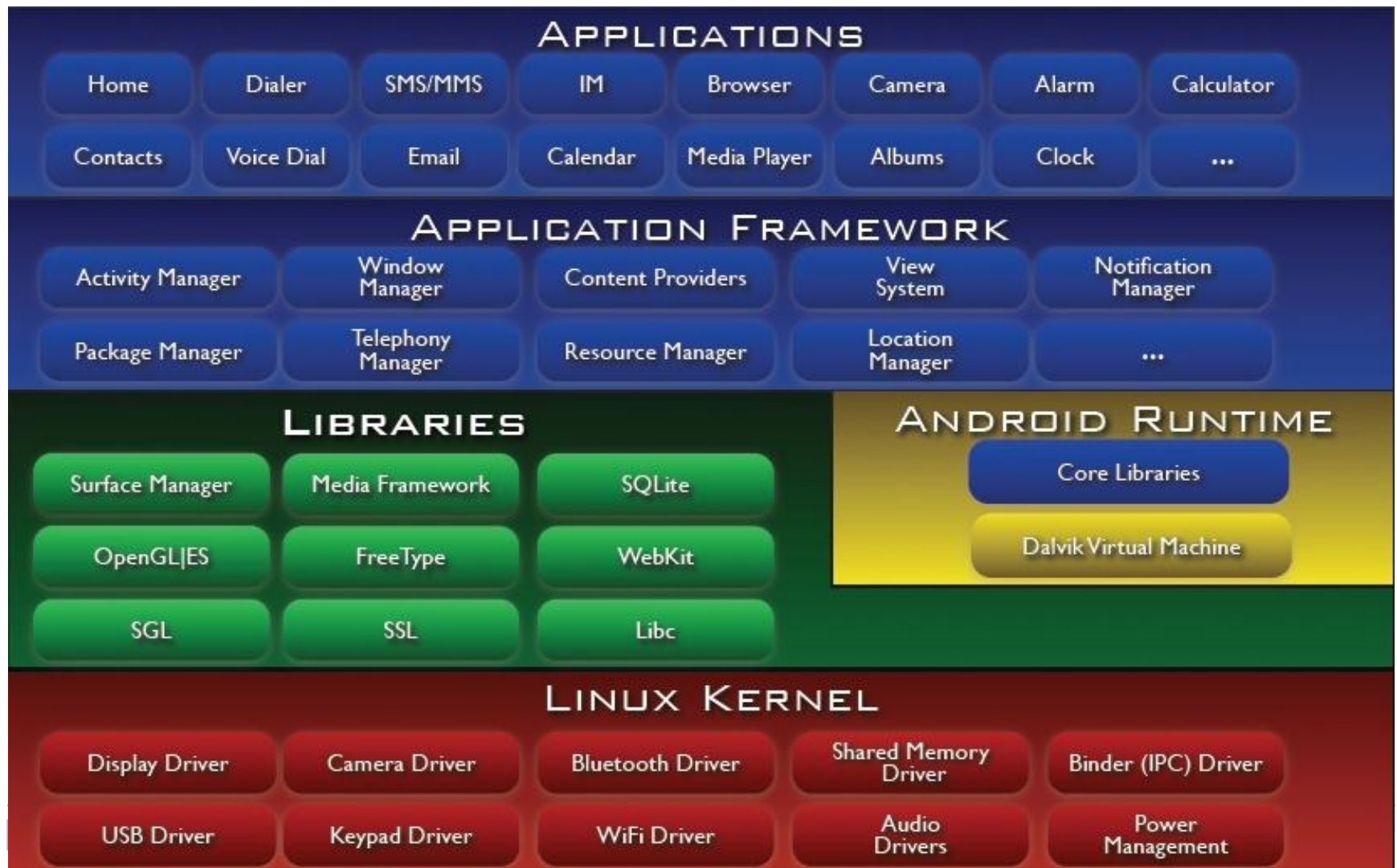
Android – Architecture

- La plate-forme Android est composée de différentes couches :
 - Un noyau Linux qui lui confère notamment des caractéristiques multitâches;
 - des bibliothèques graphiques, multimédias;
 - Une machine virtuelle Java adaptée : la Dalvik Virtual Machine
 - Un Framework applicatif proposant des fonctionnalités de gestion de fenêtres, de téléphonie, de gestion de contenu...;
 - Des applications dont un navigateur web, une gestion des contacts, un calendrier ...

Les composants majeurs de la plate-forme Android sont résumés sur le schéma suivant

Android – Architecture (1/6)

- Architecture en « pile logicielle »



Android – Architecture (2/6)



- La couche « *Applications* » :
 - *Android* est utilisé dans un ensemble contenant déjà des **applications natives** comme, un client de mail, des programmes pour envoyer des SMS, d'agenda, de navigateur web, de contacts personnels.

Android – Architecture (3/6)



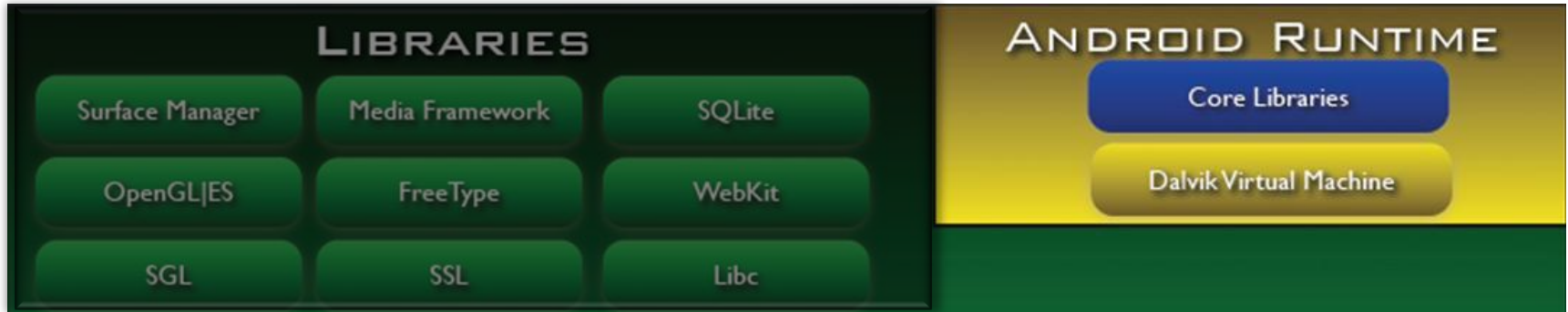
- La couche « *Application Framework* » : cette couche permet au programmeur de **construire** de nouvelles applications. Cette couche fournit la gestion :
 - des **View System** : un jeu extensible de vues utilisé pour créer des interfaces utilisateur d'application (=IHM)
 - des **ContentProviders** (Fournisseurs de contenu) = Permet aux applications de publier et de partager des données avec d'autres applications (ex : les contacts)
 - Des **Resources Manager** = Fournit l'accès à des ressources non intégrées au code telles que les chaînes, les paramètres de couleur et les mises en page de l'interface utilisateur.
 - des **Notifications Manager** : Permet aux applications d'afficher des alertes et des notifications à l'utilisateur
 - des **activity Manager** = l'enchaînement des écrans

Android – Architecture (4/6)



- La couche « *Librairies* » (bibliothèques) = couche logicielle basse pour utiliser :
 - Les **formats multimédia** : images, audio et vidéo
 - Les **dessins 2D et 3D**, bitmap et vectoriel
 - Une **base de données léger SQL** (SQLite)

Android – Architecture (5/6)



- L'environnement d'exécution (*Android Runtime*) : Toute application est **exécutée** dans son **propre processus**, dans sa **propre DVM** « *Dalvik virtual machine* ».

Android – Architecture (6/6)



- Le **noyau Linux** sur lequel la *Dalvik virtual machine* s'appuie pour **gérer le multithreading**, la **mémoire**.
- Le noyau Linux **apporte** les **services de sécurité**, la **gestion des processus**, etc.

Noyau Android

- ▶ Le noyau Android est une version modifiée du noyau Linux et représente le cœur du système.
- ▶ Parmi les modifications notables apportées dans le noyau Android, nous pouvons citer les mécanismes :
 - ▶ binder,
 - ▶ ashmem,
 - ▶ wakelock,
 - ▶ low memory killer,
 - ▶ logger,
 - ▶ RAM console
 - ▶ Paranoid Networking

Noyau Android

▶ Binder

- ▶ est un mécanisme de **communication** entre processus et d'appel de **méthodes distantes**.
- ▶ L'appel de méthodes distantes, appelé *Remote Procedure Call*, consiste à faire exécuter une méthode par une entité distante.

▶ Ashmem pour *Anonymous Shared Memory*

- ▶ est un mécanisme de **partage de mémoire** similaire à celui du noyau Linux shm.
- ▶ Il est utilisé pour partager les données entre applications Android.
- ▶ Parmi les nouveautés apportées par **ashmem** il y a par exemple l'**usage de compteur** pour connaître le **nombre de processus faisant référence à une zone de mémoire partagée** et **éviter les fuites de mémoire**.

Noyau Android

▶ Wakelock

- ▶ est un mécanisme servant à **notifier le noyau de ne pas se mettre en veille**.
- ▶ Contrairement aux systèmes Linux, le système Android essaie par défaut de se mettre en veille étant donné qu'il est destiné à tourner sur des appareils à ressources limitées.
- ▶ Lors d'exécution de code ne devant être interrompu, le wakelock est ainsi utilisé pour dire au système de rester éveillé.

▶ Low memory killer

- ▶ est un mécanisme utilisé par le noyau pour libérer de la mémoire lorsqu'il ne reste plus assez de mémoire.

▶ Logger

- ▶ est un mécanisme de journalisation qui écrit les évènements du système uniquement dans des zones allouées en mémoire.
- ▶ les évènements écrits dans le journal du système ne sont ainsi jamais écrits dans un fichier (Contrairement aux systèmes Linux traditionnels).

Noyau Android

▶ RAM Console

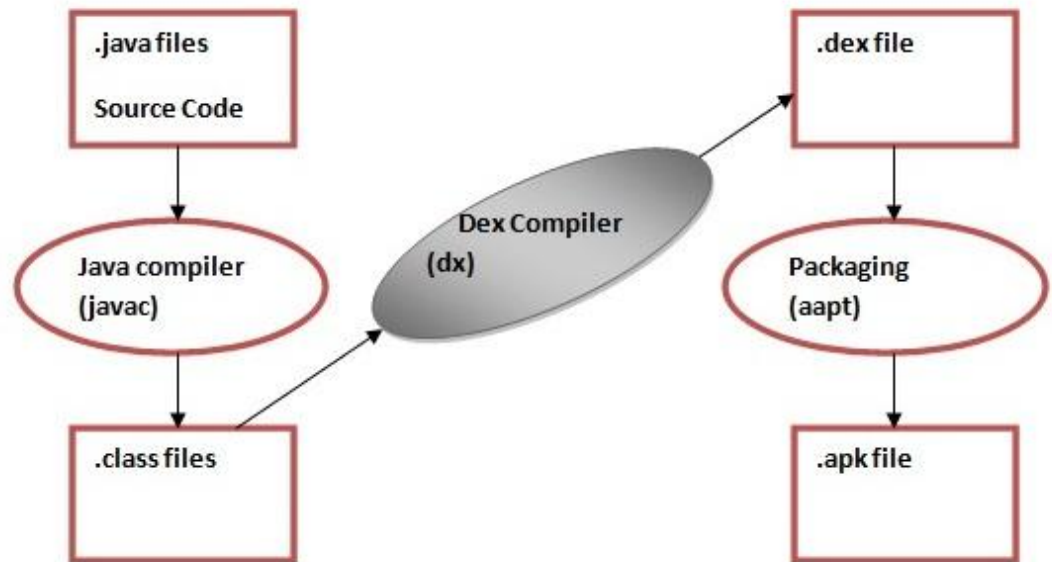
- ▶ est un mécanisme qui préserve en mémoire le contenu des événements systèmes ajoutés par du code noyau (via la fonction `printk`) lors de la précédente exécution du système.
- ▶ Son contenu est accessible via le fichier `/proc/last_kmsg`.
- ▶ Sous Linux, le contenu du journal des événements systèmes sont stockés de manière persistante dans les fichiers sous le répertoire `/var/log/`.
- ▶ Ce n'est pas le cas sous Android et en cas de crash du système par exemple, il devient impossible de récupérer les événements qui ont amené au crash.
- ▶ RAM Console est ainsi été créé pour résoudre cette limitation du système de journalisation sous Android

▶ Paranoid Network

- ▶ est un mécanisme contrôlant l'accès des applications au réseau.
- ▶ Seule les applications avec une autorisation explicite sont autorisées à créer des sockets et communiquer sur le réseau.

Android – Dalvik Virtual Machine (DVM)

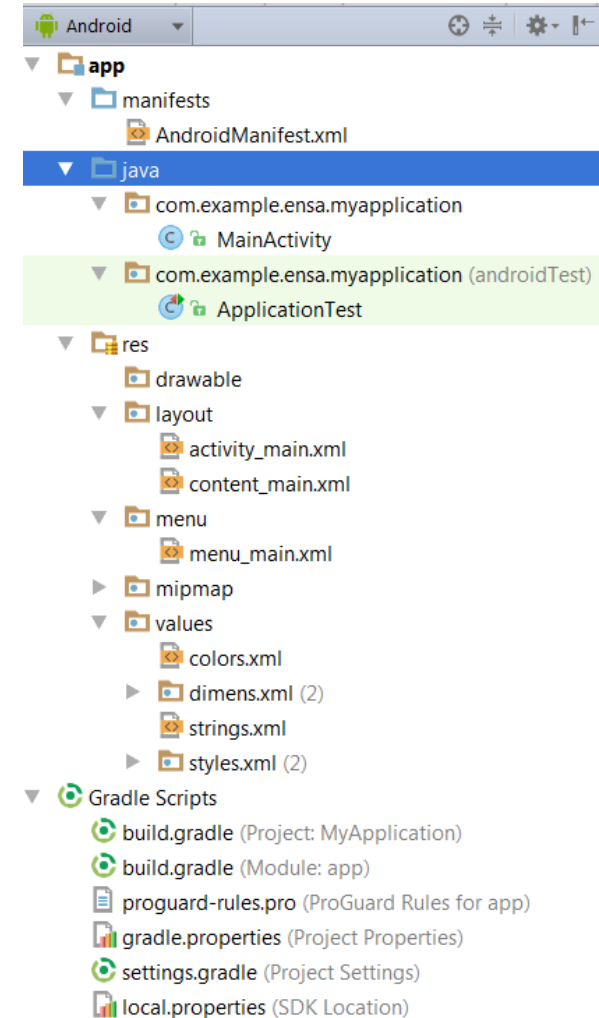
- ▶ DVM est la machine virtuelle Java pour les applications Android
- ▶ Conçu pour exécuter le code Java des systèmes ayant des contraintes de place mémoire et rapidité d'exécution
- ▶ Exécute du code .dex (Dalvik executable) = des .class adaptés à l'environnement Android



- ▶ Écrit par Dan Bornstein d'où le nom (= village islandais dont sont originaires certains de ces ancêtres)

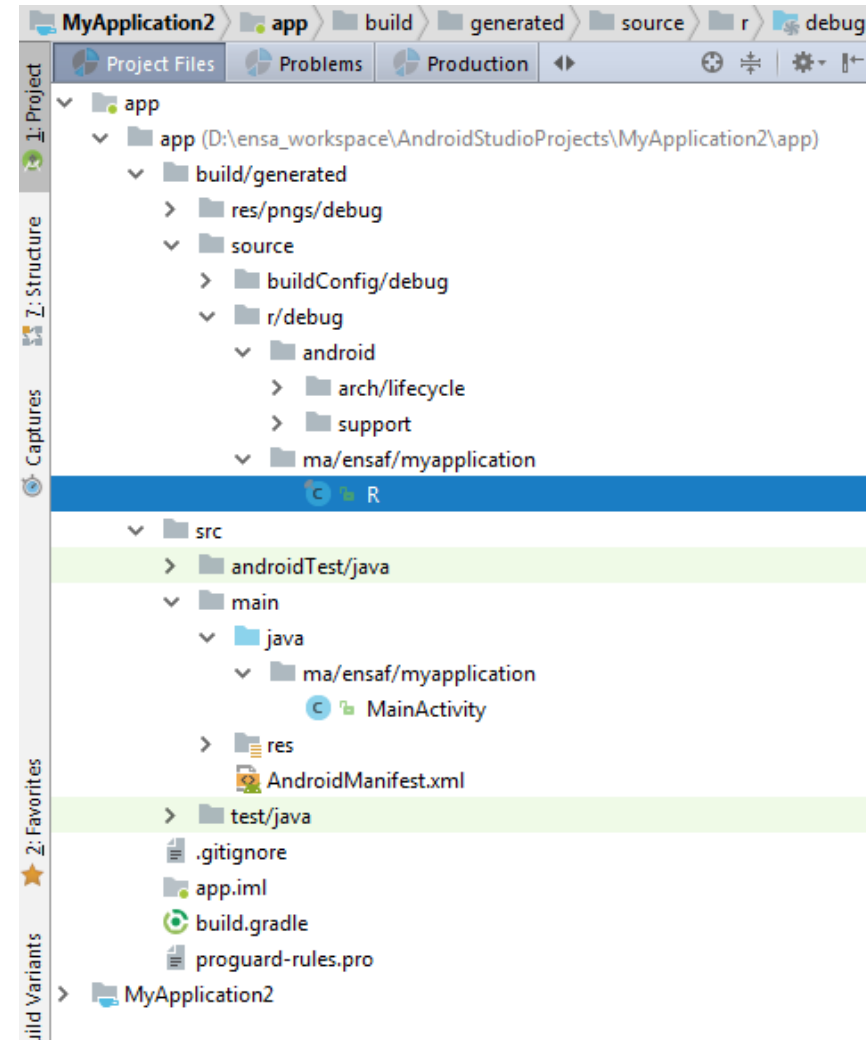
Composition d'une application Android

- Une application Android est un assemblage de composants liés grâce à un fichier de configuration.
- On trouve parmi autres :
 - Les activités (Activity)
 - Les vues et contrôles (et leurs mise en pages)
 - Les ressources
 - Le fichier de configuration appelé également *Manifeste*, et contient :
 - une description du contenu du logiciel ;
 - des fichiers présents dans l'archive ;
 - des demandes d'autorisations ;
 - des signature des fichiers, durée de validité, etc.

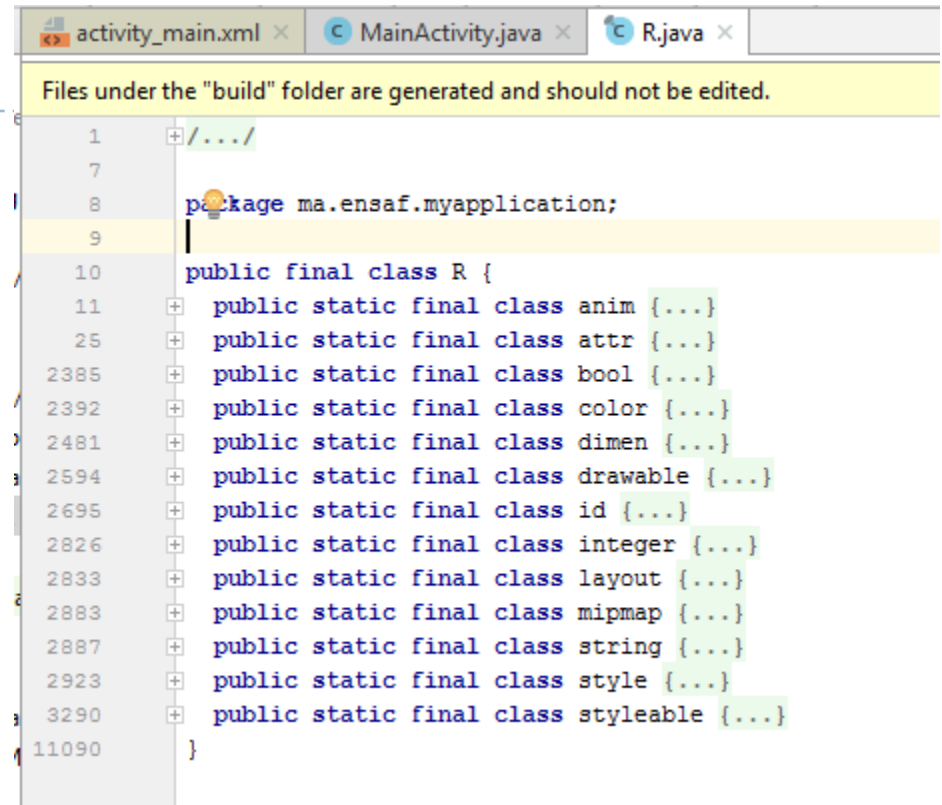


Composition d'une application Android

- ▶ Visiblement on utilise le R.java
- ▶ Remarque :
 - ▶ R.java est généré par l'environnement (parfois à la première compilation) et ne devra pas être édité, ...



Le R.java généré



```
activity_main.xml x MainActivity.java x R.java x
Files under the "build" folder are generated and should not be edited.
1  + /.../
7
8  package ma.ensaf.myapplication;
9
10 public final class R {
11  + public static final class anim {...}
25  + public static final class attr {...}
2385 + public static final class bool {...}
2392 + public static final class color {...}
2481 + public static final class dimen {...}
2594 + public static final class drawable {...}
2695 + public static final class id {...}
2826 + public static final class integer {...}
2833 + public static final class layout {...}
2883 + public static final class mipmap {...}
2887 + public static final class string {...}
2923 + public static final class style {...}
3290 + public static final class styleable {...}
11090 }
```

- ▶ Des constantes dont :
 - ▶ La constante main (R.layout.main)
 - ▶ app_name (R.string.app_name)
 - ▶ hello (R.string.hello)

Remarques sur le R.java (1 / 4)

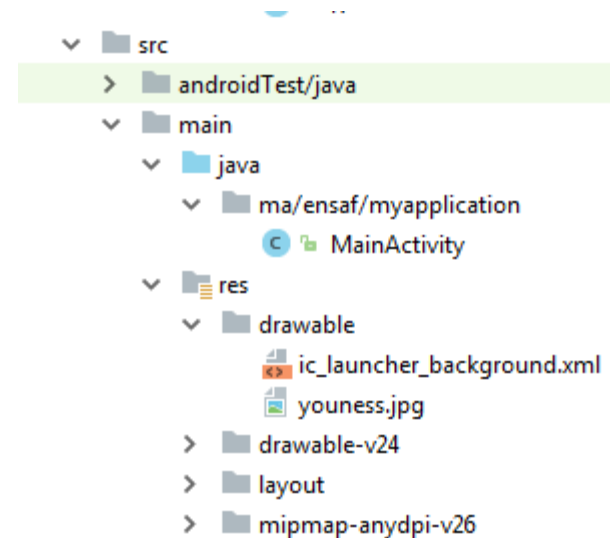
```
Files under the "build" folder are generated and should not be edited.

1  /* AUTO-GENERATED FILE. DO NOT MODIFY.
2
3  * This class was automatically generated by the
4  * aapt tool from the resource data it found. It
5  * should not be modified by hand.
6  */
7
8  package ma.ensaf.myapplication;
9
```

- ▶ Comme indiqué dans l'entête, il est automatiquement généré (le forcer à se générer à la première construction du projet)
- ▶ Il y a correspondance entre les noms dans le fichier R.java et les noms utilisés dans le programme Android. Les identificateurs dans R.java font référence à des fichiers se trouvent dans le répertoire **res**. Par exemple **R.layout.main** indique le fichier **main.xml** se trouvant dans le répertoire **layout** (sous répertoire de **res**)
- ▶ Comme **main.xml** décrit une interface graphique, on affecte cette IHM à une activité par setContentView (R.layout.main);

Remarques sur le R.java (2/4)

```
10 public final class R {
11     + public static final class anim {...}
25     + public static final class attr {...}
2385    + public static final class bool {...}
2392    + public static final class color {...}
2481    + public static final class dimen {...}
2594    + public static final class drawable {
2693        public static final int tooltip_frame_light=0x7f060063;
2694        public static final int youness =0x7f060064;
2695    }
2696    + public static final class id {...}
2827    + public static final class integer {...}
2834    + public static final class layout {...}
2884    + public static final class mipmap {...}
2888    + public static final class string {...}
2924    + public static final class style {...}
3291    + public static final class styleable {...}
11091 }
```



- ▶ Les images sont accessibles par `R.drawable.nomImage` et correspondent à des fichiers images dans le répertoire `res/drawable`.
- ▶ Le fichier `R.java` permet d'associer un `int` à ce nom (de fichier)

Remarques sur le R.java (3/4)

```
2827 public static final class integer {...}
2834 public static final class layout {...}
2884 public static final class mipmap {...}
2888 public static final class string {
2889     public static final int abc_action_bar_home_description=0x7f
2920     public static final int app_name=0x7f0b001f;
2921     public static final int bouton1=0x7f0b0020;
2922     public static final int hello=0x7f0b0021;
2923     public static final int search_menu_title=0x7f0b0022;
2924     public static final int status_bar_notification_info_overflow
2925 }
2926 public static final class style {...}
3293 public static final class styleable {...}
11093 }
```

activity_main.xml x MainActivity.java x strings.xml x

Edit translations for all locales in the translations editor.

```
1 <resources>
2     <string name="app_name">My Application</string>
3     <string name="hello">Hello World</string>
4     <string name="bouton1">Valider</string>
5 </resources>
6
```

- ▶ Les noms sont accessibles par `R.string.nom` et correspondent à des chaînes de caractères dans le fichier `strings.xml` (avec un `s` !) dans le répertoire `res/values`. On récupère ces noms dans le code source par `getResources().getString(R.string.nom)`;
- ▶ Le fichier `R.java` permet d'associer un `int` à ce nom.

Remarques sur le R.java (4/4)

- ▶ Les composants graphiques ont un identifiant dans le fichier xml qui les contient (par exemple le main.xml).
- ▶ Cet identifiant est la valeur de l'attribut android:id et est de la forme "*@+id/leId*".
- ▶ On récupère, dans le code java, le composant graphique grâce à cet identifiant par l'appel `findViewById(R.id.leId)`;
- ▶ Par exemple `Button leBouton = (Button)findViewById(R.id.leBelId)`;

Structure d'un fichier XML

► Un fichier XML :

- nœuds racine,
- éléments,
- attributs,
- valeurs,
- texte.

```
<?xml version="1.0" encoding="utf-8"?>
<racine>
  <!-- commentaire -->
  <element attribut1="valeur1"
           attribut2="valeur2">
    <feuille1 attribut3="valeur3"/>
    <feuille2>texte</feuille2>
  </element>
  texte en vrac
</racine>
```

Espaces de nommages dans un fichier XML

- Dans le cas d'Android, il y a un grand nombre d'éléments et d'attributs normalisés. Pour les distinguer, ils ont été regroupés dans le namespace android. Dans la norme XML, le namespace par défaut n'est jamais appliqué aux attributs, donc il faut mettre le préfix (**android:**) sur chacun d'eux.

```
<menu
  xmlns:android= "http://schemas.android.com/apk/res/android">
  <item
    android:id="@+id/action_settings"
    android:orderInCategory="100"
    android:showAsAction="never"
    android:title="Configuration"/>
</menu>
```

- Voir : http://www.w3schools.com/xml/xml_namespaces.asp

Création d'un écran

- Chaque écran est géré par une instance d'une sous-classe perso de **Activity**. Sa méthode `onCreate` définit, entre autres, ce qui doit être affiché sur l'écran :

```
public class MainActivity extends Activity {  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.main);  
    }  
}
```

- La méthode **setContentView** spécifie l'identifiant de l'interface à afficher dans l'écran : `R.layout.main`. C'est un entier, identifiant d'une disposition de vues : un *layout*.

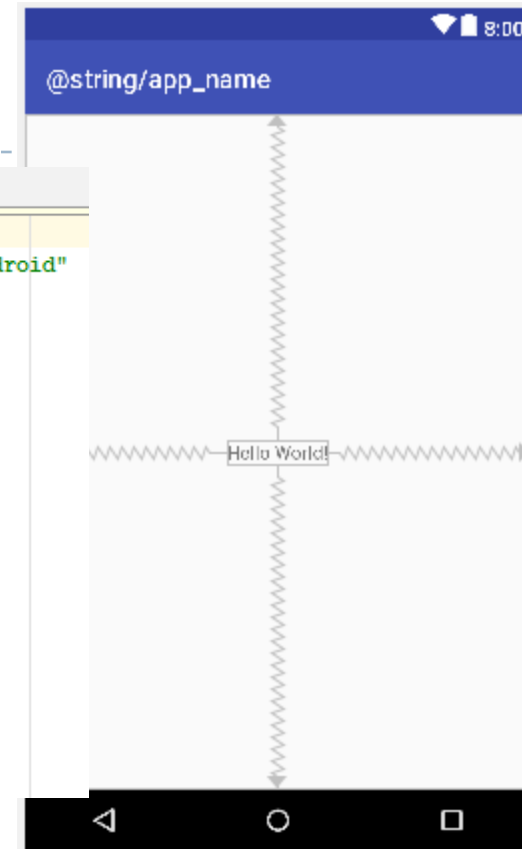
Layout



```
<?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"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</android.support.constraint.ConstraintLayout>
```



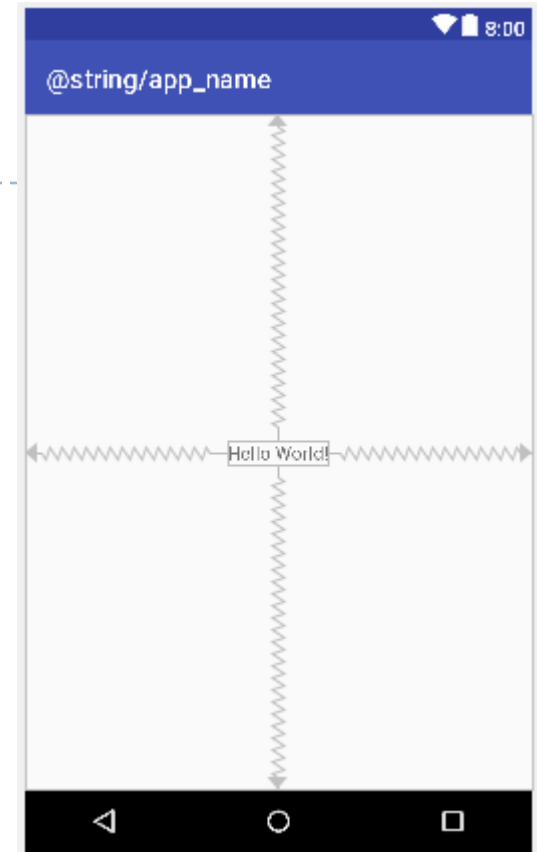
- ▶ Layout = le placement des composants graphiques dans l'IHM
- ▶ Avec Android, il peut être décrit dans un fichier XML
- ▶ Par exemple le « activity_main.xml » dans « res/layout/activity_main.xml »

Layout : activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas
  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">

  <TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/hello"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

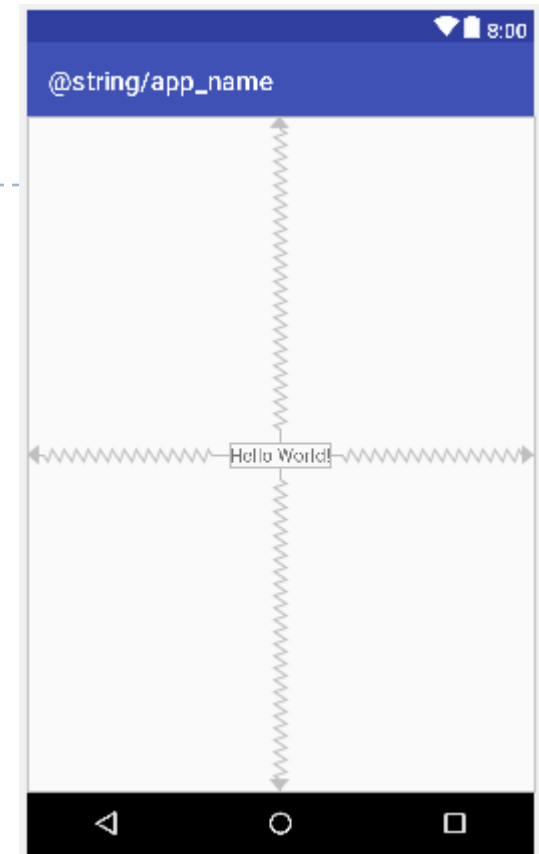
</android.support.constraint.ConstraintLayout>
```



- ▶ Un **ConstraintLayout** a été utilisé contenant un **TextView**
- ▶ Le texte affiché par le TextView (= une zone de texte ~ Label de AWT) est la chaîne (@string) de l'élément hello du fichier strings.xml
 - ▶ **android:text="@string/hello"**

string.xml

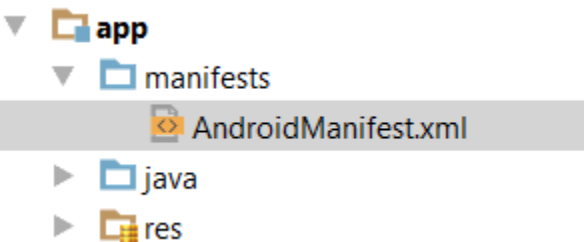
```
activity_main.xml x MainActivity.java x strings.xml x
Translations for all locales in the translations editor.
<resources>
    <string name="app_name">My Application</string>
    <string name="hello">Hello World</string>
    <string name="bouton1">Valider</string>
</resources>
```



- ▶ L'élément « hello » du fichier `strings.xml` a pour corps « Hello World »
- ▶ C'est la chaîne affichée dans le TextField à l'exécution

Android – le Manifest de l'application

Le fichier **AndroidManifest.xml** déclare l'ensemble des éléments de l'application.



Fichier obligatoire qui décrit les caractéristiques de l'Application :

- requirements : par ex. camera indispensable
- que faire au démarrage
- permissions
- icônes
- etc.

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

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

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

</manifest>
```

Pour qu'une activité soit disponible depuis le lanceur d'applications, elle doit inclure un **Intent Filter** à l'écoute de l'action **MAIN** ainsi une catégorie **LAUNCHER**

Création d'une interface par programme

- Il est possible de créer une interface en java par programme, mais c'est assez compliqué:

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    Context ctx = getApplicationContext();  
    TextView tv = new TextView(ctx);  
    tv.setText("ENSAF!");  
    RelativeLayout rl = new RelativeLayout(ctx);  
    LayoutParams lp = new LayoutParams();  
    lp.width = LayoutParams.MATCH_PARENT;  
    lp.height = LayoutParams.MATCH_PARENT;  
    rl.addView(tv, lp);  
    setContentView(rl);  
}
```


Programme et ressources

- Il est donc préférable de stocker l'interface dans un fichier `res/layout/main.xml` :

```
<RelativeLayout ... >
    <TextView android:text="ENSAF !" ... />
</RelativeLayout>
```

- Qui est référencé par son identifiant `R.layout.nom_du_fichier` dans le programme Java :

```
protected void onCreate(Bundle bundle) {
    super.onCreate(bundle);
    setContentView(R.layout.main);
}
```

Ressources de type chaînes

- Dans `res/values/strings.xml`, on place les chaînes de l'application, au lieu de les mettre en constantes dans le code source :

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">HelloWorld</string>
    <string name="main_menu">Menu principal</string>
    <string name="action_settings">Configuration</string>
    <string name="bonjour">Bonjour ENSAF ! </string>
</resources>
```

- Intérêt : pouvoir traduire une application sans la recompiler.

Les ressources – l'internationalisation

- Le système de ressources permet de gérer très facilement l'internationalisation d'une application.
- Il suffit de créer des répertoires **values-XX** où **XX** est le code de la langue que l'on souhaite implanter.
- On place alors dans ce sous répertoire le fichier xml **strings.xml** contenant les chaines traduites associées aux même clefs que dans **values/strings.xml**.
- On obtient par exemple pour les langues *es* et *fr* l'arborescence :
- Android chargera le fichier de ressources approprié en fonction de la langue du système.

```
MyProject/  
  res/  
    values/  
      strings.xml  
    values-es/  
      strings.xml  
    values-fr/  
      strings.xml
```

Référencement des ressources texte

- Voici comment affecter une ressource chaîne à une vue en Java :

```
TextView tv = new TextView(ctx);  
tv.setText(R.string.bonjour);
```

- Voici comment spécifier un titre de label dans un layout.xml :

```
<RelativeLayout>  
    <TextView android:text="@string/bonjour" />  
</RelativeLayout>
```

- @string/nom signifie la chaîne de res/values/strings.xml ayant ce nom.

Identifiants et vues

- La méthode `setContentView` fait afficher le formulaire défini par l'identifiant `R.layout` indiqué. Lorsque l'application veut manipuler l'une de ses vues, elle doit faire utiliser `R.id.symbole`, ex :

```
TextView tv = (TextView) findViewById(R.id.message);
```

- (remarquez la conversion de type), avec la définition suivante:

```
<RelativeLayout>  
<TextView android:id="@+id/message"  
android:text="@string/bonjour" />  
</RelativeLayout>
```

- La notation `@+id/nom` fait créer ou utiliser `R.id.nom`.

Identifiants et vues

- Il y a les deux notations :
 - @id/nom pour référencer un identifiant déjà défini (ailleurs)
 - @+id/nom pour définir (créer) cet identifiant
- Exemple :

```
<RelativeLayout xmlns:android="..." ... >
  <TextView ...
    android:id="@+id/titre"
    android:text="@string/titre" />
  <Button ...
    android:id="@+id/btn"
    android:layout_below="@id/titre"
    android:text="@string/ok" />
</RelativeLayout>
```

ImageView

- De la même façon, les images placées dans res/drawable et res/mipmaps-* sont référençables :

```
<ImageView  
    android:src="@drawable/velo"  
    android:contentDescription="@string/mon_velo" />
```

- La notation @drawable/nom référence l'image res/drawable/nom
- N.B : les dossiers res/mipmaps-* contiennent la même image à des définitions différentes, pour correspondre à différents téléphones et tablettes.

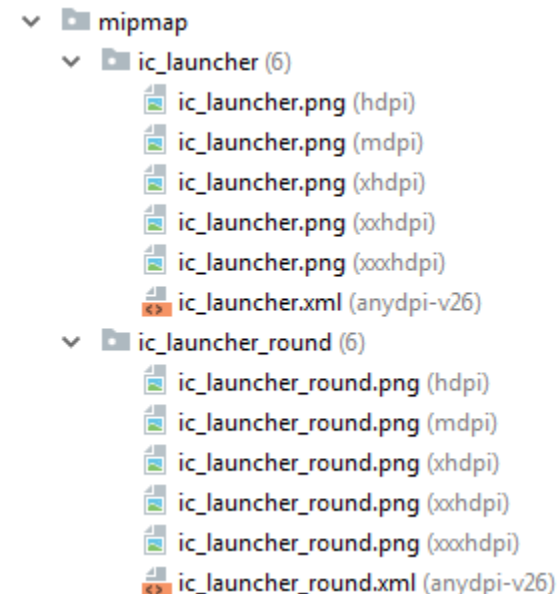


Tableau de chaîne : R.array.nom

- Exemple du fichier res/strings/arrays.xml :

```
<resources >
  <string-array name="planetes">
    <item>Mercure</item>
    <item>Venus</item>
    <item>Terre</item>
    <item>Mars</item>
  </string-array>
</resources>
```

- Dans le programme Java, il est possible de faire :

```
Resources res = getResources();
String[] planetes = res.getStringArray(R.array.planetes);
```


Google Play Store

<https://play.google.com/apps/publish/signup/>



Google Play Console



Connectez-vous à votre
compte Google.

2

Accepter le contrat du
développeur

3

Réglez vos frais
d'inscription

4

Indiquez infos sur
votre compte

Vous êtes connecté en tant que...



ykhamlichi@gmail.com

Ce compte Google sera associé à votre Developer Console

Si vous souhaitez utiliser un autre compte, vous pouvez choisir parmi les propositions ci-dessous. Si vous représentez une organisation, vous pouvez envisager d'enregistrer un nouveau compte Google, plutôt que d'utiliser un compte personnel.

[SE CONNECTER AVEC UN AUTRE COMPTE](#)

[CRÉER UN COMPTE GOOGLE](#)

Etape 1 :

Avant de poursuivre...



Accepter le contrat du développeur

Lisez et acceptez le [contrat relatif à la distribution sur Google Play \(pour les développeurs\)](#).



J'accepte et j'autorise Google à associer l'inscription de mon compte au contrat relatif à la distribution sur Google Play (pour les développeurs). Je confirme également que je suis âgé d'au moins 18 ans.



Examiner les pays de distribution

Reportez-vous à la liste des pays dans lesquels vous pouvez distribuer et vendre des applications.

[En savoir plus](#)

Si vous envisagez de vendre des applications ou des produits intégrés à l'application, vérifiez que vous pouvez créer un compte marchand dans votre pays. [En savoir plus](#)



Carte de paiement

Préparez votre carte de paiement pour régler les frais d'inscription de 25 USD à la prochaine étape.

POURSUIVRE ET PAYER

Etape 2 :

Terminer votre achat

Developer Registration Fee 25,00 \$U

Ajouter une carte de crédit ou de débit

Numéro de carte
Vous devez indiquer un numéro de carte.

Nom et prénom du titulaire
Youness Idrissi Khamlichi

Adresse de facturation

En continuant, vous créez un compte de paiement Google et acceptez les conditions suivantes : [Conditions d'utilisation de Google Payments](#) et [Avis de confidentialité](#).

ACHETER

#

Numéro de carte

MM / AA

CVC

Vous devez indiquer un numéro de carte.

Nom et prénom du titulaire

Youness Idrissi Khamlichi



Maroc (MA)

Ligne d'adresse 1

Le champ "Ligne d'adresse 1" est obligatoire.

Ligne d'adresse 2

Code postal

Ville

En continuant, vous créez un compte de paiement Google et acceptez les conditions suivantes : [Conditions d'utilisation de Google Payments](#) et [Avis de confidentialité](#).

ACHETER

Etape 3:



Votre paiement a bien été traité

Un reçu vous sera envoyé par e-mail.

[POURSUIVRE L'INSCRIPTION](#)



Connectez-vous à votre
compte Google.



Accepter le contrat du
développeur



Réglez vos frais
d'inscription



Indiquez infos sur
votre compte

Vous avez bientôt terminé...

Il vous suffit de renseigner les détails suivants. Si besoin, vous pourrez modifier ces informations ultérieurement dans les paramètres de votre compte.

Profil du développeur

Vous devez renseigner les champs marqués d'un * avant d'enregistrer.

Nom du développeur *

0/50

Le nom du développeur s'affiche sous celui de votre application. Toute modification apportée à ce nom fait l'objet d'un examen par Google et la validation peut prendre jusqu'à sept jours.

Adresse e-mail *

Site Web



Site Web

Numéro de téléphone *

Saisissez le signe plus, ainsi que l'indicatif du pays et de la région, par exemple "+1-800-555-0199".
Pourquoi vous demandons-nous votre numéro de téléphone ?

ID de compte de développeur

4792365791599607350

ID : 4792365791599607350

Mentionnez cet ID si vous devez contacter l'assistance pour les développeurs Google Play.

Préférences relatives aux e-mails

- ☐ Je souhaite être informé des nouvelles fonctionnalités et recevoir des conseils pour améliorer mes applications.
- ☐ Je souhaite envoyer des commentaires pour contribuer à l'amélioration de la console Google Play.

En accédant à la console Play, conformément aux [Règles de confidentialité de Google](#) et au [Contrat relatif à la distribution \(pour les développeurs\)](#), vous acceptez de partager avec Google des informations concernant votre utilisation de la console Play. Ces données seront utilisées pour le développement de fonctionnalités et la personnalisation de la console. Vous pouvez désactiver le partage des données concernant votre utilisation de la console dans Paramètres > préférences. [En savoir plus](#)

FINALISER L'INSCRIPTION



Toutes les applications

Services de jeux

Gestion des commandes



Télécharger des rapports

Alertes

Paramètres



PUBLIER UNE APPLICATION ANDROID SUR GOOGLE PLAY

Pour en savoir plus sur la procédure détaillée, consultez le [Guide de démarrage](#).



UTILISER LES SERVICES DE JEUX GOOGLE PLAY


Ajoutez des fonctionnalités de réseaux sociaux à vos jeux sur Android, iOS et le Web. [En savoir plus](#)



Travaillez-vous en équipe ?
[Invitez des collègues à utiliser la console Play.](#)




Si vous souhaitez créer des applications payantes ou des produits intégrés à une application, vous devez [configurer un compte marchand](#).


 **Google Payments** <payments-noreply@google.com>
À moi ▾


01:19 (il y a 2 minutes)





  Gmail

 Nouveau message

 **Boîte de réception**


 En attente

 Important

 Messages envoyés

 **Brouillons** 127

 **Spam** 79

 Youness ▾ +

merci

Bonjour,

Vous avez effectué u

Google

29 septembre 2019 à 17:13:10 UTC-7

Article	Quantité	Prix
Developer Registration Fee	1	25,00 \$US
		Taxe 0,00 \$US

25,00 \$US

Mode de paiement

Visa ••••

Numéro de commande Google

PDS.3804-2577-7896-33402

[DÉTAILS DE LA TRANSACTION](#)