ANDROID - Environnement de développement

Université Virtuelle de Côte d'Ivoire (UVCI)

Table des matières

Introduction

Ce cours est basé sur *« Programmation Android »* de Pierre NERZIC, Université de Rennes 1, IUT LANNION

INTRODUCTION

I

1. Un petit peu d'histoire

Android est né en 2004

Racheté par Google en 2005

Il est publie en 2007, sous sa version 1.5. De nombreuses versions verront le jour plus tard.

En 2018 , Google a sorti "Android Pie" qui est la version 9.x.

Logo Android Pie

2. Qu'est-ce qu'Android ?

Système complet pour smartphones et tablettes

- *Gestion matérielle* : système d'exploitation Linux sous-jacent
- *API de programmation* : interfaces utilisateur, outils. . .
- *Applications* : navigateur, courrier. . .

Évolution et obsolescence très rapides (c'est voulu)

- Ce que vous allez apprendre sera rapidement dépassé (1 an)
    - syntaxiquement (méthodes, paramètres, classes, ressources. . .)
    - mais pas les grands concepts (principes, organisation. . .)

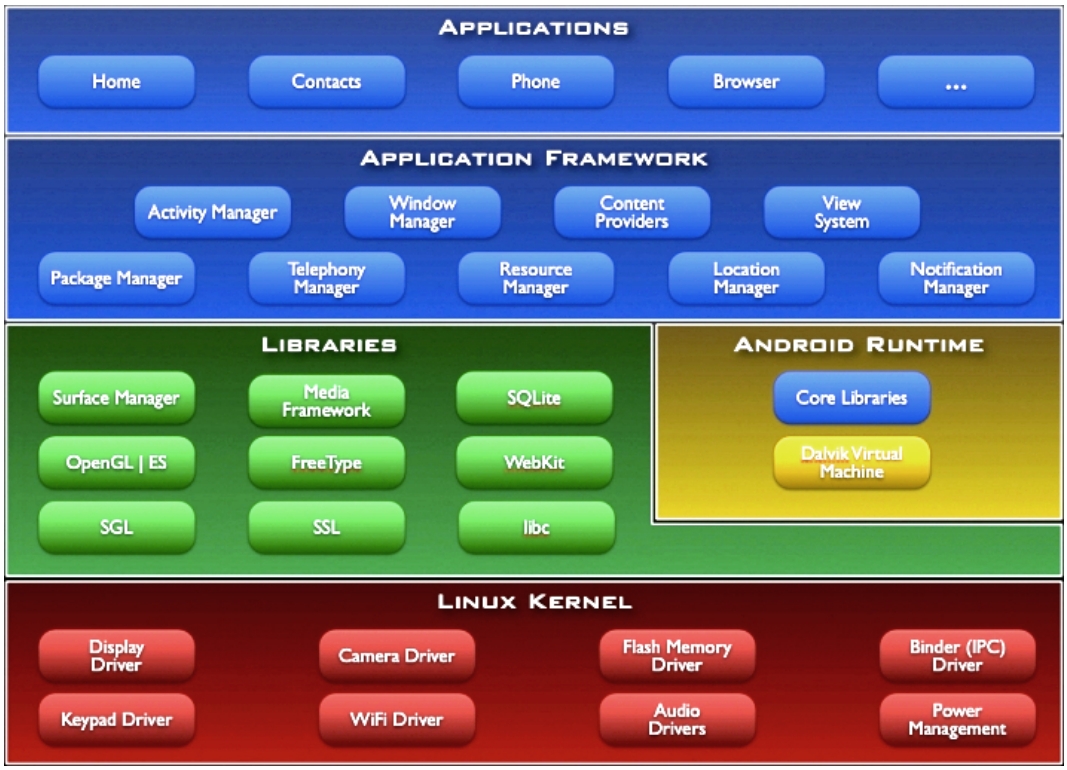
- Vous êtes condamné(e) à une autoformation permanente, mais c'est le lot des informaticiens

Définition : API

une interface de programmation applicative (souvent désignée par le terme API pour application programming interface) est un ensemble normalisé de classes, de méthodes ou de fonctions qui sert de façade par laquelle un logiciel offre des services à d'autres logiciels. Wikipedia

3. Composants d'Android

Plusieurs couches comme un système Unix :



Composants Android

4. Programmation d'applications

Une application Android est composée de :

- *Sources Java* (ou Kotlin) compilés pour une machine virtuelle « Dalvik » (versions ? 4.4) ou « ART » depuis la version 5
- *ressources* , c'est l'ensemble des fichiers aux :
  - format *XML* : interface, textes. . .
  - format *PNG* : icônes, images. . .
- *Manifeste* = description du contenu du logiciel
  - fichiers présents dans l'archive
  - demandes d'autorisations
  - signature des fichiers, durée de validité, etc.

Tout cet ensemble est géré à l'aide d'un *IDE* (environnement de développement) appelé *Android Studio* qui s'appuie sur un ensemble logiciel (bibliothèques, outils) appelé *SDK Android*.

SDK Android et Android Studio

II

1. SDK et Android Studio

Le SDK contient :

- les *librairies Java* pour créer des logiciels
- les outils de *mise en boîte des logiciels*
- *AVD* : Acronyme Android Virtual Devicede un émulateur de tablettes pour tester les applications
- *ADB* : Acronyme de Android Debug Bridge , un outil de communication avec les vraies tablettes

Android Studio offre :

- un éditeur de *sources* et de *ressources*
- des outils de compilation : *gradle*
- des outils de *test* et de *mise au point*

2. Android Studio

Pour commencer, il faut installer Android Studio selon la procédure expliquée sur le lien ci-dessous.

Suivre la procédure d'installation ici : *https://developer.android.com/studio/install*

Pour le SDK, vous avez le choix, soit de l'installer automatiquement avec Studio, soit de faire une installation personnalisée.

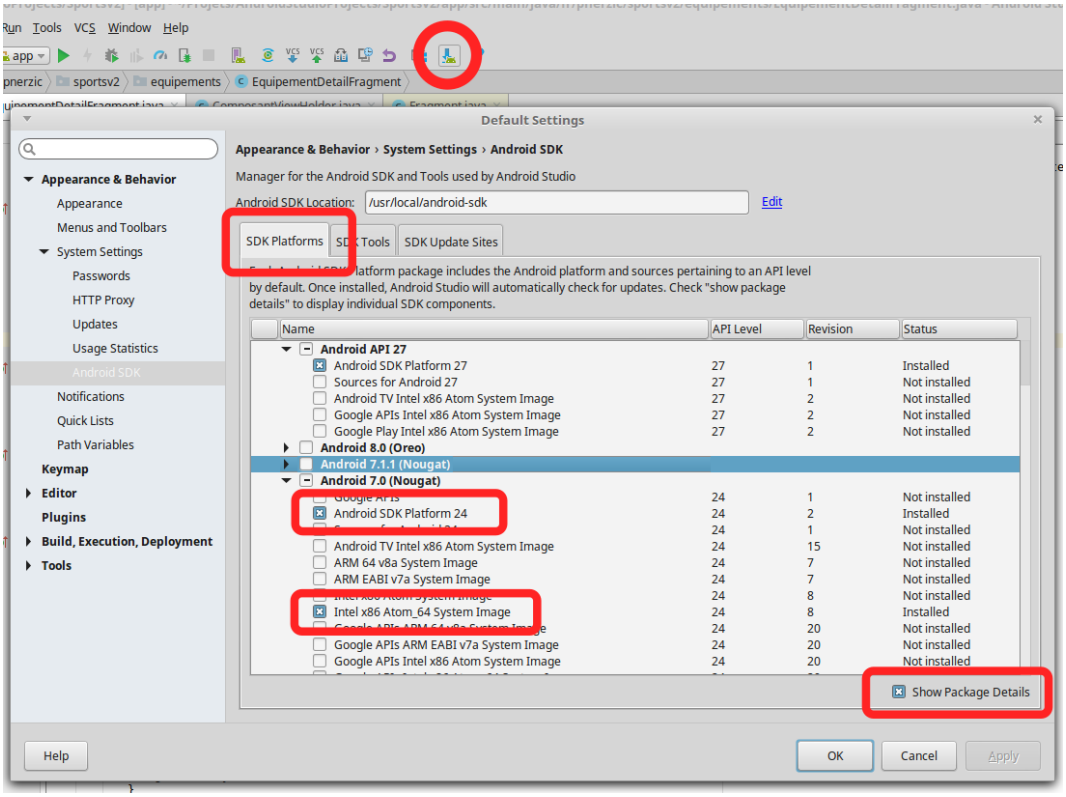
En général, vous pouvez choisir ce que vous voulez ajouter au SDK :

- version des librairies,
- versions des émulateurs de smarphones,

à l'aide du *SDK Manager*.

3. SDK Manager

C'est le gestionnaire du SDK, une application qui permet de choisir les composants à installer et mettre à jour.



Gestionnaire de paquets Android

4. Choix des éléments du SDK

Le gestionnaire permet de choisir les versions à installer

Exemple

- Android 8.1 (API 27)
- Android 8.0 (API 26)
- Android 7.0 (API 24)
- Android 6 (API 23)
- . . .

Choisir celles qui correspondent aux tablettes qu'on vise.

Il faut cocher *"Show Package Details"*, puis choisir élément par élément.

Les suivants sont indispensables :

- *Android SDK Platform*
- *Intel x86 Atom_64 System Image*

Remarque

Le reste est facultatif (*Google APIs*, *sources*, *exemples* et *docs*).

5. Dossiers du SDK

Le gestionnaire installe plusieurs fichiers dans différents sous-dossiers :

- *SDK Tools* : indispensable, contient le gestionnaire,
- *SDK Platform-tools* : indispensable, contient adb,
- *SDK Platform* : indispensable, contient les librairies,
- *System images* : pour créer des AVD,
- *Android Support* : divers outils pour créer des applications,
- Exemples et sources.

Création d'une première Application

III

Cette activite vous permettra :

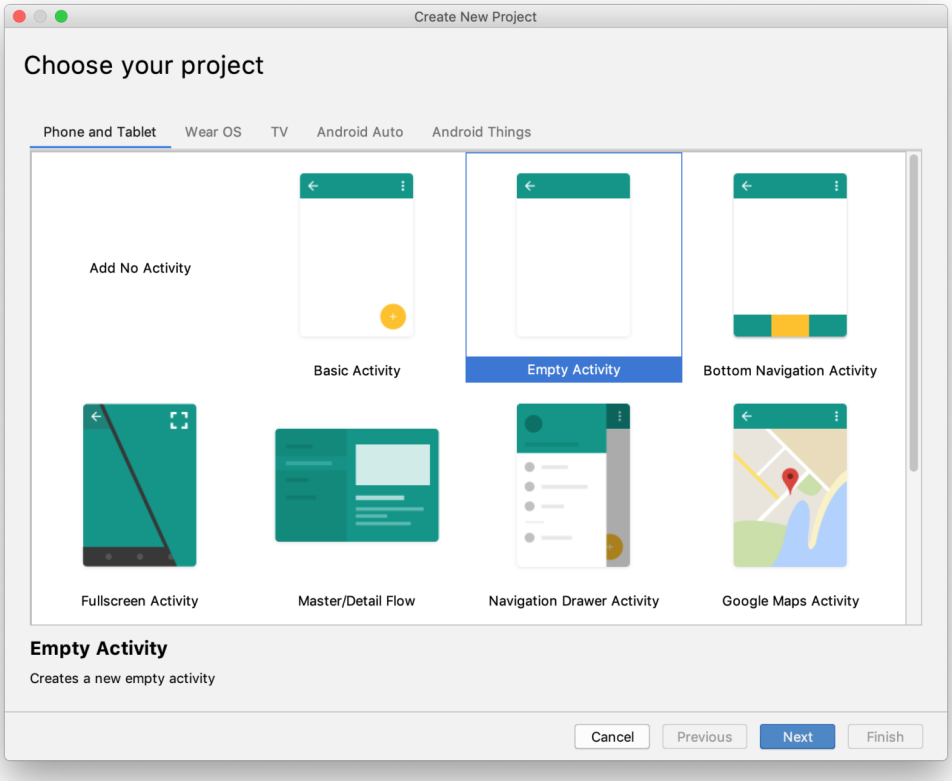
- Création d'une application « HelloWorld » avec un assistant,
- Tour du propriétaire,
- Exécution de l'application,
- Mise sous forme d'un paquet

1. Choix du type d'application

Android Studio contient un assistant de création d'applications :

Pour un premier essai, on se limite au plus simple, *Blank Activity*, comme le montre la figure ci dessous.

Si un projet est ouvert, pour créer un nouveau projet , selectionnez a partir du menu principal : *File* > *New* > *New Project*



Ce écran de l'assistant , permet de choisir le type de projet que vous souhaitez créer

Une fois la sélection effectuée, cliquez sur *next*.

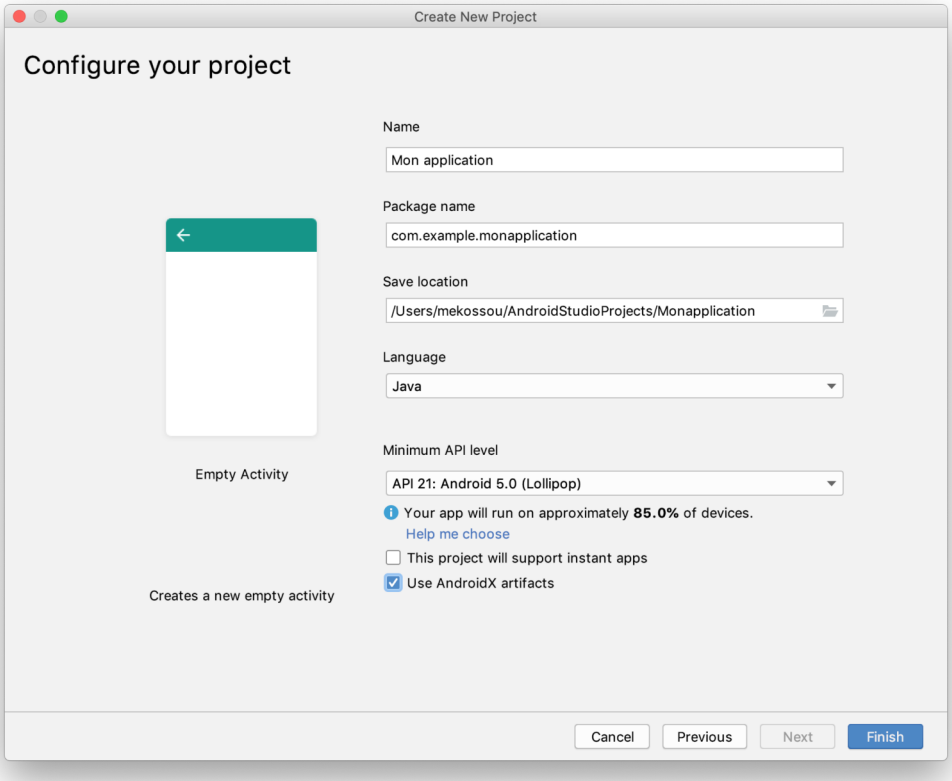
⚠ *Attention*

les copies écran ne correspondent pas exactement à ce qui est disponible, à cause des changements rapides de versions et de l'ordinateur.

2. Configurer le projet

L'assistant demande ensuite plusieurs informations :

1. *Name* : Spécifie le nom du projet
2. *Package name* : Spécifie le paquetage
3. *Save location* : Spécifie le répertoire de sauvegarde du projet
4. *Language* : Permet de choir le type de langage de développement
5. *Minimum API level* ; Permet de choisir le niveau minimum de L'API pris en charge . Sélectionner un API de niveau bas, permet à votre application de fonctionner sur 100% de smartphones android. En revanche, choisir un niveau API plus élevé, couvrira moins de smartphone android.
   Pour comprendre le fonctionnement des APIs , cliquez sur *help me choose*
6. Si vous souhaitez utiliser les librairies androidX , cochez *Use AndroidX artifacts*. cliquez *ici* pour plus de détails sur AndroidX
7. Pour terminer la creation du projet cliquer sur *Finish*



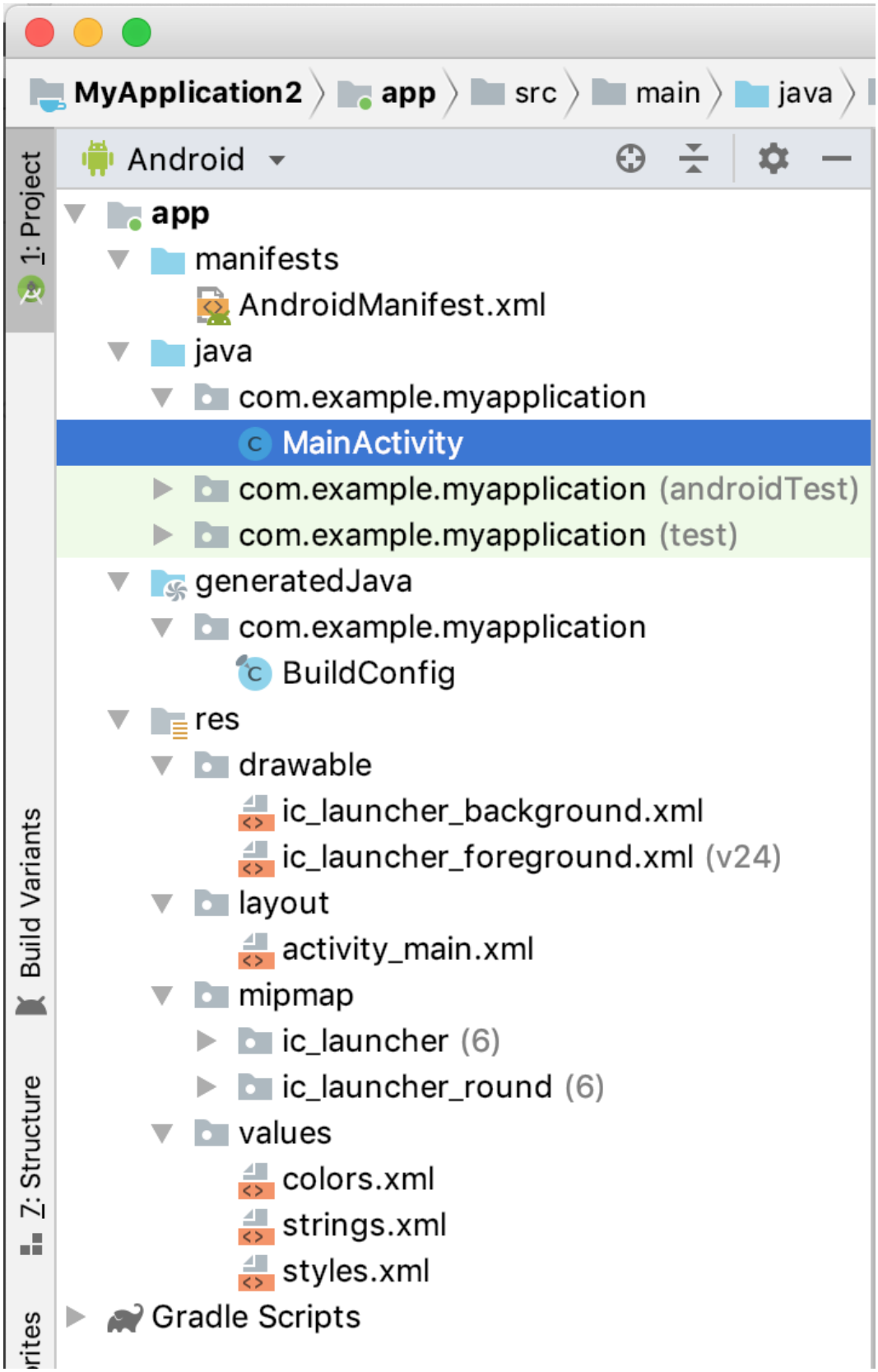
Configurez votre nouveau projet avec quelques paramètrages

3. Résultat de l'assistant

L'assistant a créé de nombreux éléments visibles dans la colonne de gauche de l'IDE :

- *manifests* : description et liste des classes de l'application
- *java* : les sources, rangés par paquetage,
- *res : ressources* = fichiers XML et images de l'interface, il y a des sous-dossiers :
    - *layout* : interfaces (disposition des vues sur les écrans)
    - *menu* : menus contextuels ou d'application
    - *mipmap et drawable* : images, icônes de l'interface
    - *values* : valeurs de configuration, textes. . .
- *Gradle* scripts : c'est l'outil de compilation du projet.

4. Fenêtre du projet



Les Fichiers du projet

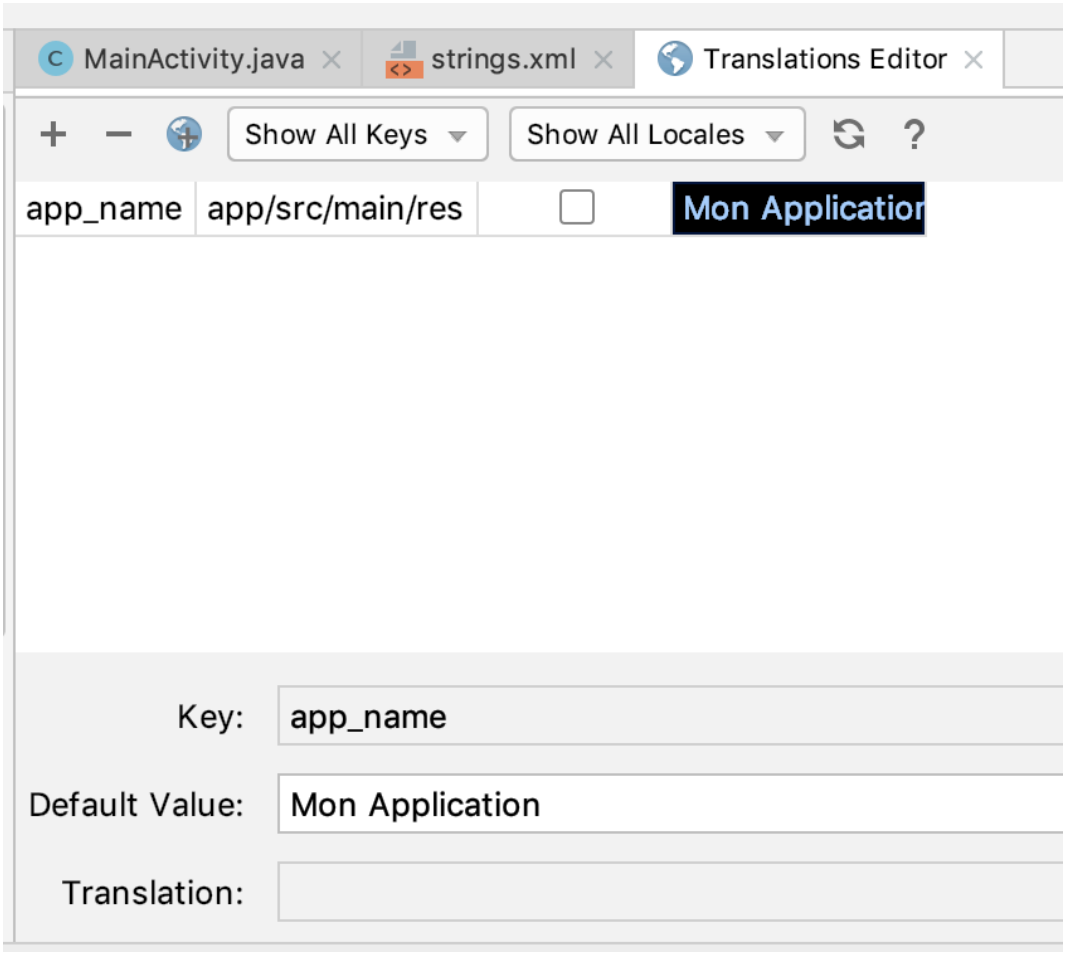
5. Éditeurs spécifiques

Les ressources (disposition des vues dans les interfaces, menus, images vectorielles, textes. . .) sont définies à l'aide de fichiers XML

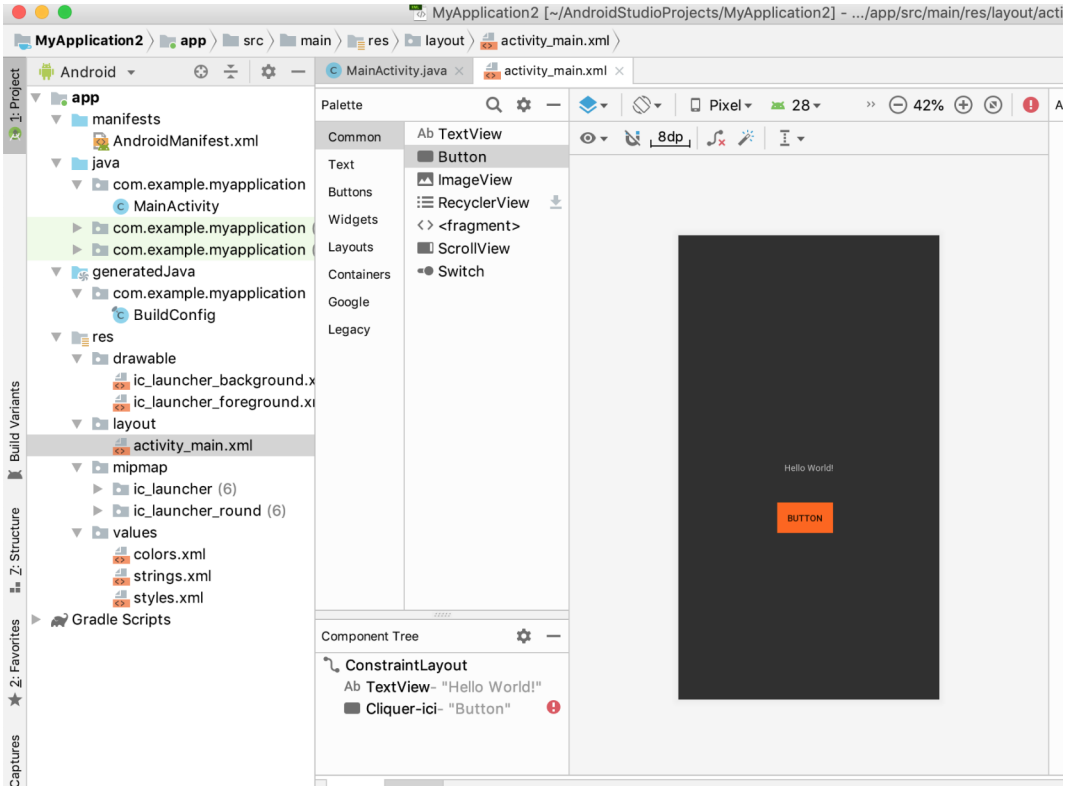
Studio fournit des éditeurs spécialisés pour ces fichiers, par exemple :

- *Formulaires* pour :
    - *res/values/strings.xml* : textes de l'interface.
- *Éditeurs graphiques* pour :
    - *res/layout/*.xml* : disposition des contrôles sur l'interface.

6. Exemple res/values/strings.xml



7. Exemple res/layout/activity_main.xml



Exemple de "res/layout/activity_main.xml"

8. Source XML sous-jacent

Ces éditeurs sont beaucoup plus confortables que le XML brut, mais ne permettent pas de tout faire (widgets custom et plantages).

Assez souvent, il faut éditer le source XML directement :

```
1
2 <?xml version="1.0" encoding="utf-8"?>
3 <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.
  android.com/apk/res/android"
4     xmlns:app="http://schemas.android.com/apk/res-auto"
5     xmlns:tools="http://schemas.android.com/tools"
6     android:layout_width="match_parent"
7     android:layout_height="match_parent"
8     tools:context=".MainActivity">
9
10    <TextView
11        android:layout_width="wrap_content"
12        android:layout_height="wrap_content"
13        android:text="Hello World!"
14        app:layout_constraintBottom_toBottomOf="parent"
15        app:layout_constraintLeft_toLeftOf="parent"
16        app:layout_constraintRight_toRightOf="parent"
17        app:layout_constraintTop_toTopOf="parent" />
18
```

```
19     <Button
20         android:id="@+id/Cliquer-ici"
21         style="@style/Widget.AppCompat.Button.Colored"
22         android:layout_width="wrap_content"
23         android:layout_height="wrap_content"
24         android:background="#ff6600"
25         android:text="Button"
26         tools:layout_editor_absoluteX="156dp"
27         tools:layout_editor_absoluteY="421dp" />
28
29 </androidx.constraintlayout.widget.ConstraintLayout>
30
```

9. Reconstruction du projet

Chaque modification d'un source ou d'une ressource fait reconstruire le projet. C'est automatique.

Dans certains cas (travail avec un gestionnaire de sources comme Subversion ou Git), il peut être nécessaire de reconstruire manuellement.

Il suffit de sélectionner le sur la barre de menu principal *Build* > *Rebuild Project.*

En cas de confusion d'Android Studio (compilation directe en ligne de commande), ou de mauvaise mise à jour des sources (partages réseau), il faut parfois nettoyer le projet.

Sélectionner le menu principal *Build* > *Clean Project.*

Remarque

Ces actions lancent l'exécution de *Gradle*.

10. Gradle

Gradle est un outil de construction de projets comme *Make* (projets C++ sur Unix), *Ant* (projets Java dans Eclipse) et *Maven*.

De même que *make* se sert d'un fichier *Makefile*, *Gradle* se sert d'un fichier nommé *build.gradle* pour construire le projet.

C'est assez compliqué car AndroidStudio fait une distinction entre le projet global et l'application. Donc il y a deux *build.gradle* :

- un script *build.gradle* dans le dossier racine du projet. Il indique quelles sont les dépendances générales (noms des dépôts Maven contenant les librairies utilisées).
- un dossier *app* contenant l'application du projet.
- un script *build.gradle* dans le dossier app pour compiler l'application.

11. Structure d'un projet AndroidStudio

Un projet Android Studio est constitué ainsi :

```
1 .
2 +-- app/
```

```
| +-- build/                                                    FICHIERS
TEMPORAIRES
| +-- build.gradle
COMPILATION
| `-- src/
| +-- androidTest/                                              TESTS
ANDROID
| +-- main/
| | +-- AndroidManifest.xml                                     DESCR. DE
L'APPLICATION
| | +-- java/                                                   SOURCES
| | `-- res/                                                    RESSOURCES
| `-- test/                                                     TESTS
JUNIT
+-- build/                                                      FICHIERS
TEMPORAIRES
+-- build.gradle
DEPENDANCES
`-- gradle/                                                     FICHIERS
DE GRADLE
```

12. Mises à jour

Le *SDK* ainsi que *Gradle* sont régulièrement mis à jour, automatiquement avec Android Studio. Cependant, vous devrez parfois éditer les *build.gradle* à la main pour en tenir compte.

Par exemple, ce build.gradle de projet fait appel à Gradle 3.0.1 et Realm 4.3.2.

```
buildscript {
 repositories { ... }
 dependencies {
  classpath 'com.android.tools.build:gradle:3.0.1'
  classpath 'io.realm:realm-gradle-plugin:4.3.2'
 }
}
```

Il faudra changer les numéros de version manuellement en cas de mise à jour, puis reconstruire le projet (*sync now ou try_again*).

13. Mises à jour (suite)

C'est fréquent d'avoir à modifier le *build.gradle* situé dans le dossier *app*.

```
apply plugin: 'com.android.application'
android {
 compileSdkVersion 27
 buildToolsVersion "28.0.3" <<== PAS PAR DEFAUT
 ...
}
dependencies {
 compile 'com.android.support:appcompat-v7:27.1.1'
 ...
}
```

Cette application dépend du SDK API 27 et de l'outil de compilation 28.0.3. Ça sera à changer lors d'une mise à jour du SDK.

14. Utilisation de bibliothèques

Certains projets font appel à des bibliothèques externes. Cela fait généralement rajouter quelques lignes dans les deux *build.gradle*.

Par exemple, *Realm* (une base de données distribuée), voir *prerequisites* :

- dans le build.gradle du dossier app :
  apply plugin: 'realm-android'
- dans le build.gradle à la racine du projet :

```
dependencies {
 classpath "io.realm:realm-gradle-plugin:5.8.0"
}
```

La reconstruction du projet fait automatiquement télécharger la bibliothèque

Première exécution

IV

1. Exécution de l'application

L'application est prévue pour tourner sur un appareil (smartphone ou tablette) réel ou simulé (virtuel).

Le SDK Android permet de :

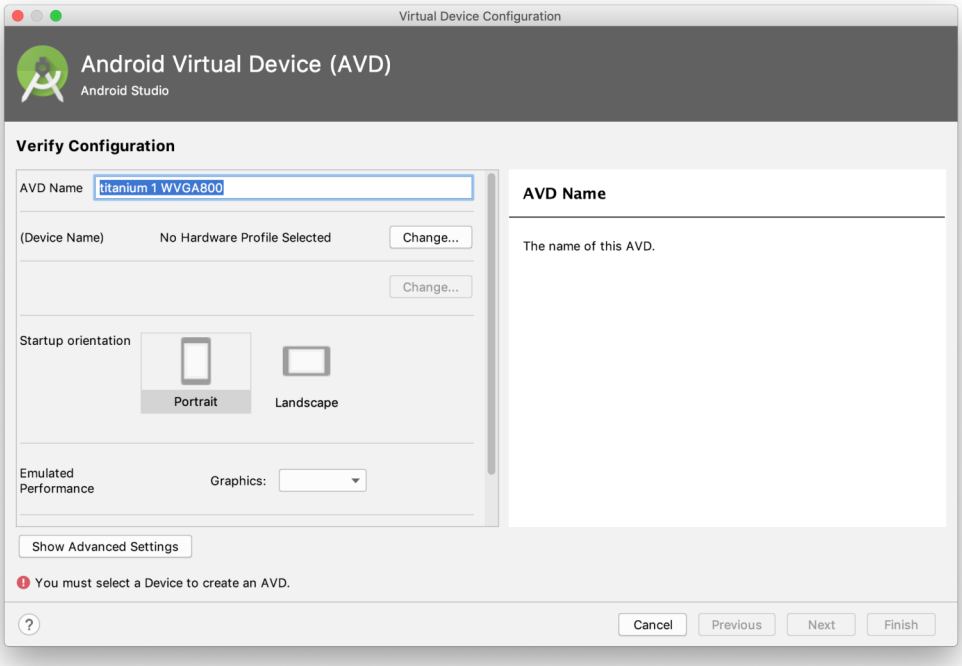
- Installer l'application sur une vraie tablette connectée par USB
- Simuler l'application sur une tablette virtuelle AVD

AVD = Android Virtual Device

C'est une machine virtuelle comme celles de VirtualBox et VMware, mais basée sur *QEMU*

QEMU est en licence GPL, il permet d'émuler toutes sortes de CPU dont des ARM7, ceux qui font tourner la plupart des tablettes Android.

2. Assistant de création d'une tablette virtuelle



Android Virtual Device

3. Caractéristiques d'un AVD

L'assistant de création de tablette demande :

- Modèle de tablette ou téléphone à simuler,
- Version du système Android,
- Orientation et densité de l'écran
- Options de simulation :
    - *Snapshot* : mémorise l'état de la machine d'un lancement à l'autre, mais exclut Use Host GPU,
    - *Use Host GPU* : accélère les dessins 2D et 3D à l'aide de la carte graphique du PC.
- Options avancées :
    - *RAM* : mémoire à allouer, mais est limitée par votre PC,
    - *Internal storage* : capacité de la flash interne,
    - *SD Card* : capacité de la carte SD simulée supplémentaire (optionnelle).

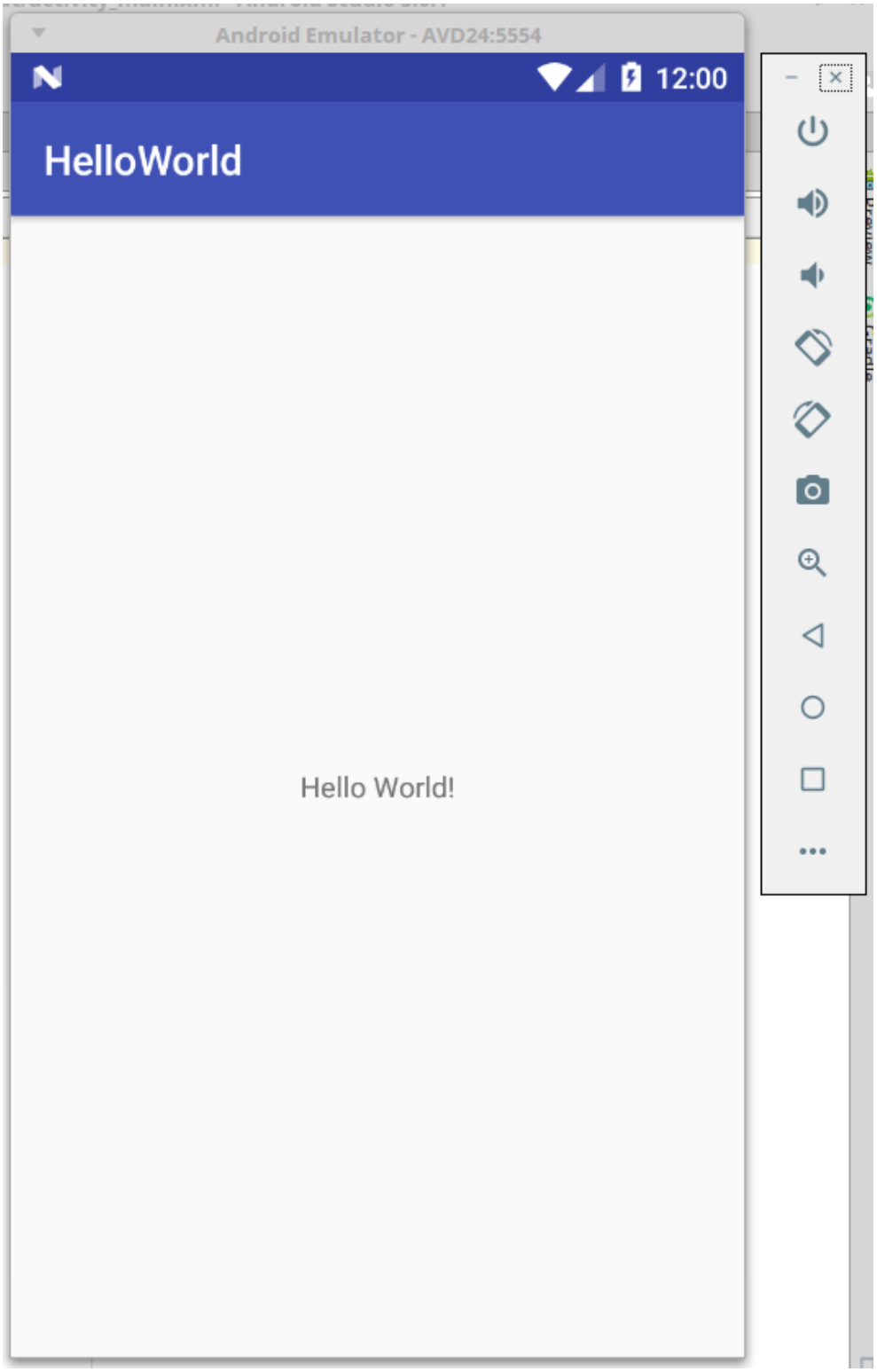
4. Lancement d'une application

Bouton vert pour exécuter, bleu pour déboguer :

⚠ *Attention*

Les icônes changent selon la version d'Android Studio.

5. Application sur l'AVD



L'apparence change d'une version à l'autre du SDK.