

## **Tuto 1 : Android Studio**

### **1. Installation**

Vous aurez besoin d'Android Studio et du SDK Android.

1. Aller à : <https://developer.android.com/studio/index.html>
2. Télécharger et installer Android Studio.

Si vous choisissez une installation « custom », le programme vous demandera quels outils installer, en plus d'Android Studio (l'IDE). Vous aurez au minimum besoin de :

- SDK Android
  - SDK API level 21 (correspond à la version d'Android 5.0 « Lollipop »)
  - l'émulateur (image x86) pour mettre au point et tester votre application sans périphérique Android
- .....

### **2. Hello World**

Nous allons commencer par faire un classique Hello World sur Android.

Création du projet

Dans l'IDE Android Studio :

- Démarrer un nouveau projet Android, ou File -> New Project...
  - Application Name : Hello Android
  - Company Domain : helloandroid.m2dl.com ----- Next
- Choix des cibles du projet
  - Sélectionnez seulement Phone and Tablet
  - Dans le menu déroulant, choisir le niveau d'API 21 (Android 5.0 Lollipop)
  - Vérifiez que votre périphérique est bien *au moins* en version 5.0, sinon choisissez le niveau correspondant à votre version. ----- Next
- Choisir Empty Activity ----- Next
- Configuration de l'activité
  - Activity Name : HelloAndroid
  - Laissez les autres options par défaut ----- Next ---- Finish

Vous pouvez désormais compiler l'application via Build -> Make Project (ou Ctrl + F9).

### **3. Préparation d'un émulateur**

Si vous avez besoin d'un émulateur pour tester votre application, vous pouvez créer un nouveau *Android Virtual Device* (AVD) :

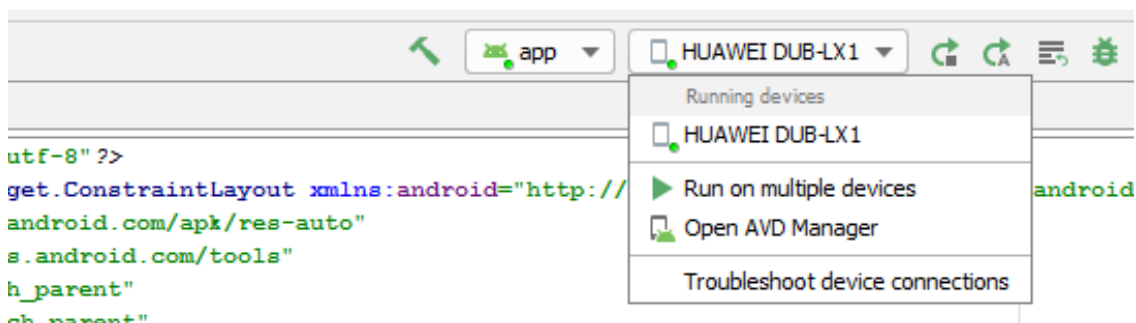
- Dans Tools -> Android -> AVD Manager ---- Create a virtual devices
  - Choisissez un périphérique, par exemple *Nexus 5X* -- Next
  - Choisissez une image ayant un niveau d'API  $\geq 21$  et une ABI x86
  - Téléchargez l'image si elle n'est pas déjà présente -- Next -- Next
  - Vous pouvez lancer l'émulateur (bouton vert en face de votre nouvel AVD).

Lancer l'application en utilisant l'émulateur ou votre propre appareil Android.

Connectez votre appareil par câble USB à l'ordinateur et installez le pilote si nécessaire. Activez l'option de débogage USB sur l'appareil en allant dans les paramètres, sous développement ou options pour les développeurs. Pour les versions supérieures à 4.2, cette option est cachée par défaut, pour la faire apparaître, allez dans Paramètres> A propos .. et touchez Numéro de build sept fois.

Retournez ensuite à l'écran Paramètres , vous verrez apparaître Options pour les développeurs, rentrez y et activez le débogage.

Lancez l'application depuis Android Studio comme précédemment. Si on vous demande de choisir l'appareil, sélectionnez Choose a running device, puis votre téléphone ou tablette. Android Studio installera l'application sur votre appareil et la lancera.



On obtient comme résultat « Hello Word » au centre de l'écran.

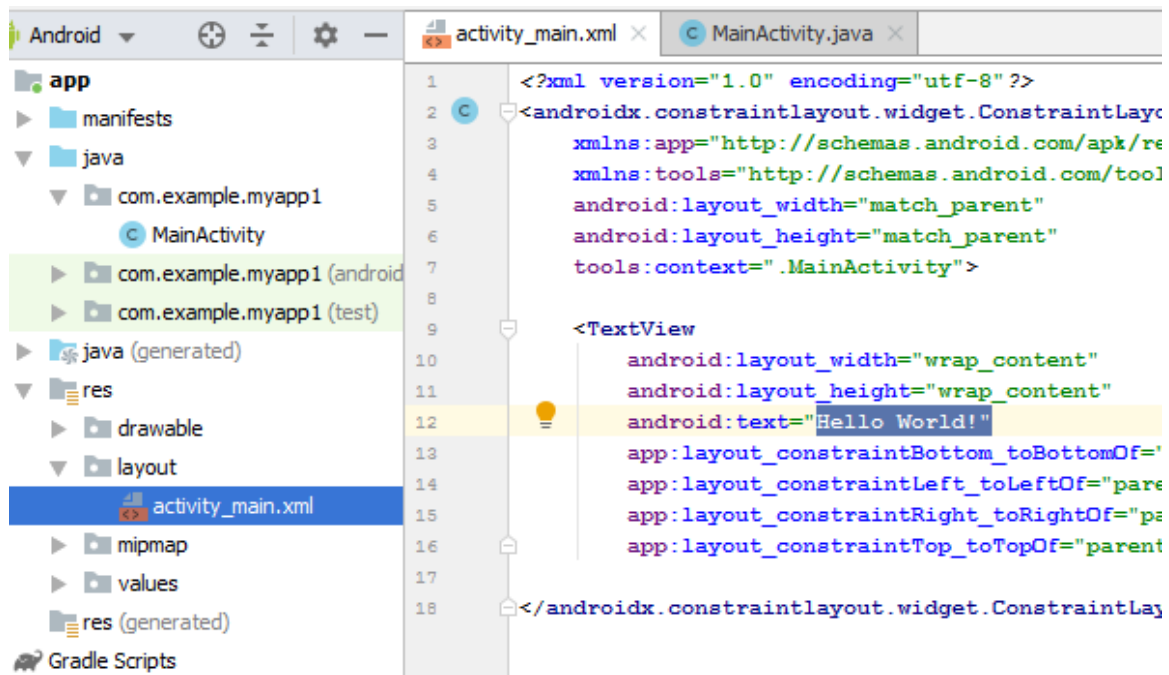
### **4. Structure du projet**

Tout projet Android doit respecter une hiérarchie bien précise qui permettra au compilateur de retrouver les différents éléments et ressources lors de la génération de l'application. Cette hiérarchie favorise la modularité des applications Android.

A la création du projet, Android Studio crée automatiquement des dossiers pour contenir les fichiers de code Java, les fichiers XML, et les fichiers multimédias.

L'explorateur de projet vous permettra de naviguer dans ces dossiers. Allez voir le dossier correspondant au projet, Z:\AndroidStudioProjects\VotreProjet :

- Le dossier du projet contient le premier build.gradle et le dossier app
- app contient le second build.gradle et un dossier src
- app/src/main contient le cœur du projet : manifeste, sources et ressources.
- app/src/main/java contient les dossiers correspondant au package que vous avez déclaré, avec les sources Java tout au bout du package,
- app/src/main/res/drawable contiendra des images vectorielles utilisées dans votre projet et tout élément qui peut être dessiné sur l'écran : image (en PNG de préférence), formes, animations, transitions, etc.. Cinq dossiers **drawable** permettent aux développeurs de proposer des éléments graphiques pour tout genre d'appareil pour offrir une interface qui s'adapte à chaque résolution d'écran. Ldpi (low-resolution dots per inch~120dpi) mdpi (~160dpi) hdpi (~240dpi) xhdpi (~320dpi) xxhdpi (~480dpi).
- app/src/main/res/layout contient les dispositions d'écran des activités de votre projet : regroupe les fichiers XML qui définissent la disposition des composants sur l'écran. Il contient déjà, dès la création du projet, le layout de l'activité principale que nous avons créée. une interface qui s'adapte à chaque résolution d'écran.
- app/src/main/res/menu contient les menus contextuels et d'application du projet
- app/src/main/res/mipmap\* contient les icônes bitmap dans diverses résolutions
- app/src/main/res/values contient les constantes du projet, dont les chaînes de caractères avec toutes leurs traductions.



Les dossiers que nous utiliserons le plus sont **java et res**. Le premier contient le code Java qui définit le comportement de l'application (situé dans le répertoire de votre projet sous app\src\main), et le second comporte des sous dossiers (dans app\src\main\res) où sont stockés les ressources qui définissent l'interface de l'application (l'apparence).

## 5. Modification de notre application

On va commencer par faire une application simple (une seule activité, qui ne fera qu'afficher «**Hello, Android**» cette fois).

Editez res/values/string.xml et remplacer *Hello world* par *Bonjour le monde* ;)

Ou bien

Allez dans le fichier app -> java -> com.m2dl.helloandroid.helloandroid -> HelloAndroid.

Il contient notre activité principale.

Ajouter l'import de TextView au début du fichier :

```
import android.widget.TextView
```

On va maintenant remplacer la méthode onCreate par le code suivant :

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
  
    // on crée un nouveau TextView, qui est un widget permettant  
    // d'afficher du texte  
    TextView tv = new TextView(this);  
  
    // configurer le texte à faire afficher par notre widget  
    tv.setText("Hello, Android");  
  
    // remplacer tout le contenu de notre activité par le TextView  
    setContentView(tv);  
}
```

On remarquera ici que la description de l'interface pourra autant se faire dans le code que dans le fichier **xm**l représentant le layout de l'activité.

La bonne pratique étant évidemment de séparer autant que possible le layout du code.

## 6. Vibration

Il est possible d'utiliser le vibreur du téléphone (si un vibreur est accessible). Pour le faire vibrer pendant 1 seconde (1000 millisecondes), on fait :

```
Vibrator vibrator = (Vibrator) getSystemService(Context.VIBRATOR_SERVICE);  
vibrator.vibrate(1000);
```

Rajoutez cela dans le code et testez sur l'émulateur et sur le téléphone.

Un message indiquant un manque de permission sera affiché. Dans le fichier *AndroidManifest.xml* (via *app* -> *manifests* -> *AndroidManifest.xml*), on va rajouter la permission `android.permission.VIBRATE` comme ceci :

```
<?xml version="1.0" encoding="utf-8"?>  
<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
    <uses-permission android:name="android.permission.VIBRATE" />  
    <application>
```

## 7. Création d'un paquet installable

Un paquet Android est un fichier .apk. C'est une archive signée (authentifiée) contenant les binaires, ressources compressées et autres fichiers de données. La création est relativement simple avec Studio :

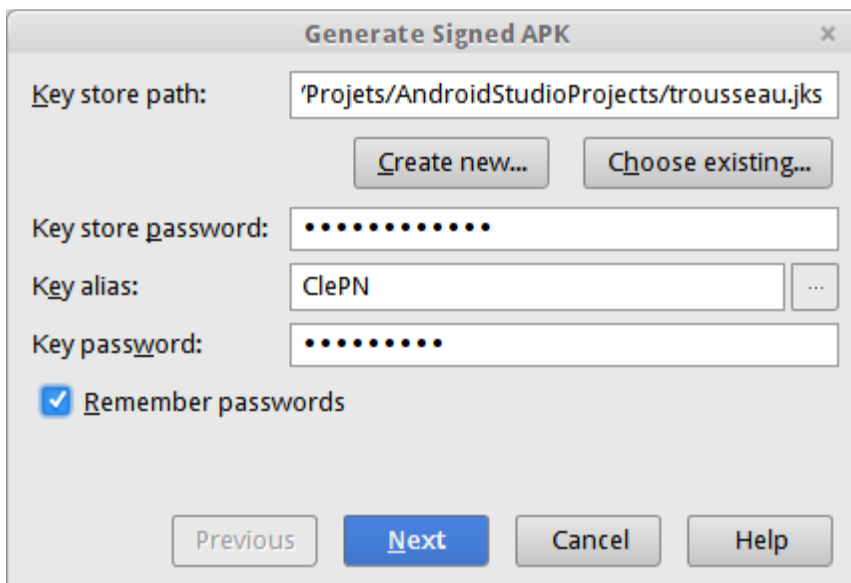
1. Menu contextuel du projet Build..., choisir Generate Signed APK,

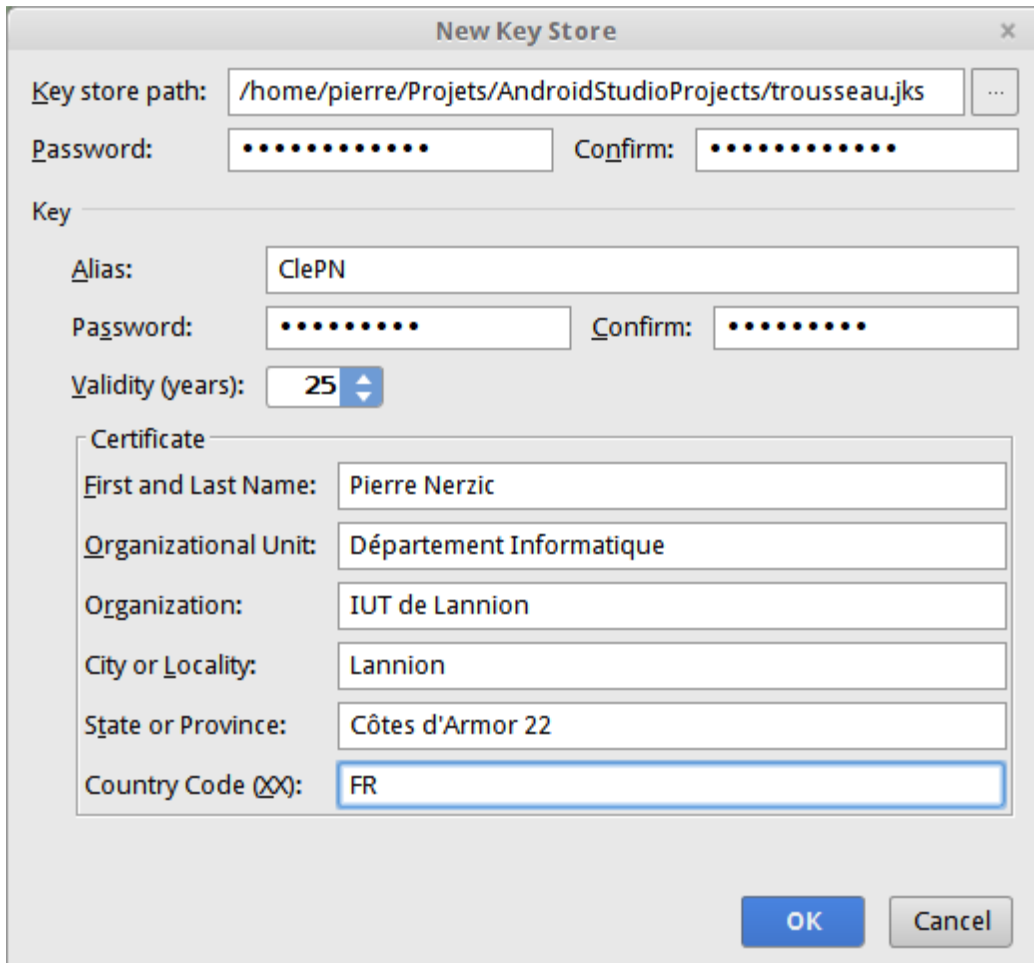
2. Signer le paquet à l'aide d'une clé privée,
  3. Définir l'emplacement du fichier .apk.
- Le résultat est un fichier .apk dans le dossier spécifié.

Lors de la mise au point, Studio génère une clé qui ne permet pas d'installer l'application ailleurs. Pour distribuer une application, il faut une clé privée. Les clés sont stockées dans un keystore = trousseau de clés. Il faut le créer la première fois. C'est un fichier crypté, protégé par un mot de passe, à ranger soigneusement.

Ensuite créer une clé privée :

- alias = nom de la clé, mot de passe de la clé
- informations personnelles complètes : prénom, nom, organisation, adresse, etc. Les mots de passe du trousseau et de la clé seront demandés à chaque création d'un .apk. **Ne les oubliez pas.**





**New Key Store**

Key store path: /home/pierre/Projets/AndroidStudioProjects/trousseau.jks

Password: ..... Confirm: .....

Key

Alias: ClePN

Password: ..... Confirm: .....

Validity (years): 25

Certificate

First and Last Name: Pierre Nerzic

Organizational Unit: Département Informatique

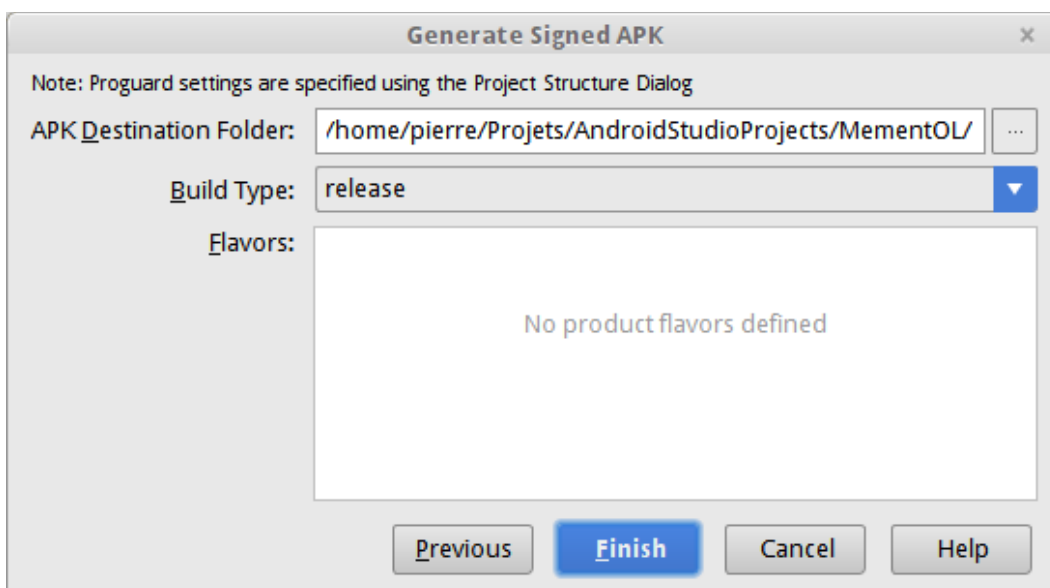
Organization: IUT de Lannion

City or Locality: Lannion

State or Province: Côtes d'Armor 22

Country Code (XX): FR

OK Cancel



**Generate Signed APK**

Note: Proguard settings are specified using the Project Structure Dialog

APK Destination Folder: /home/pierre/Projets/AndroidStudioProjects/MementOL/

Build Type: release

Flavors:

No product flavors defined

Previous Finish Cancel Help