Les premiers pas avec Android

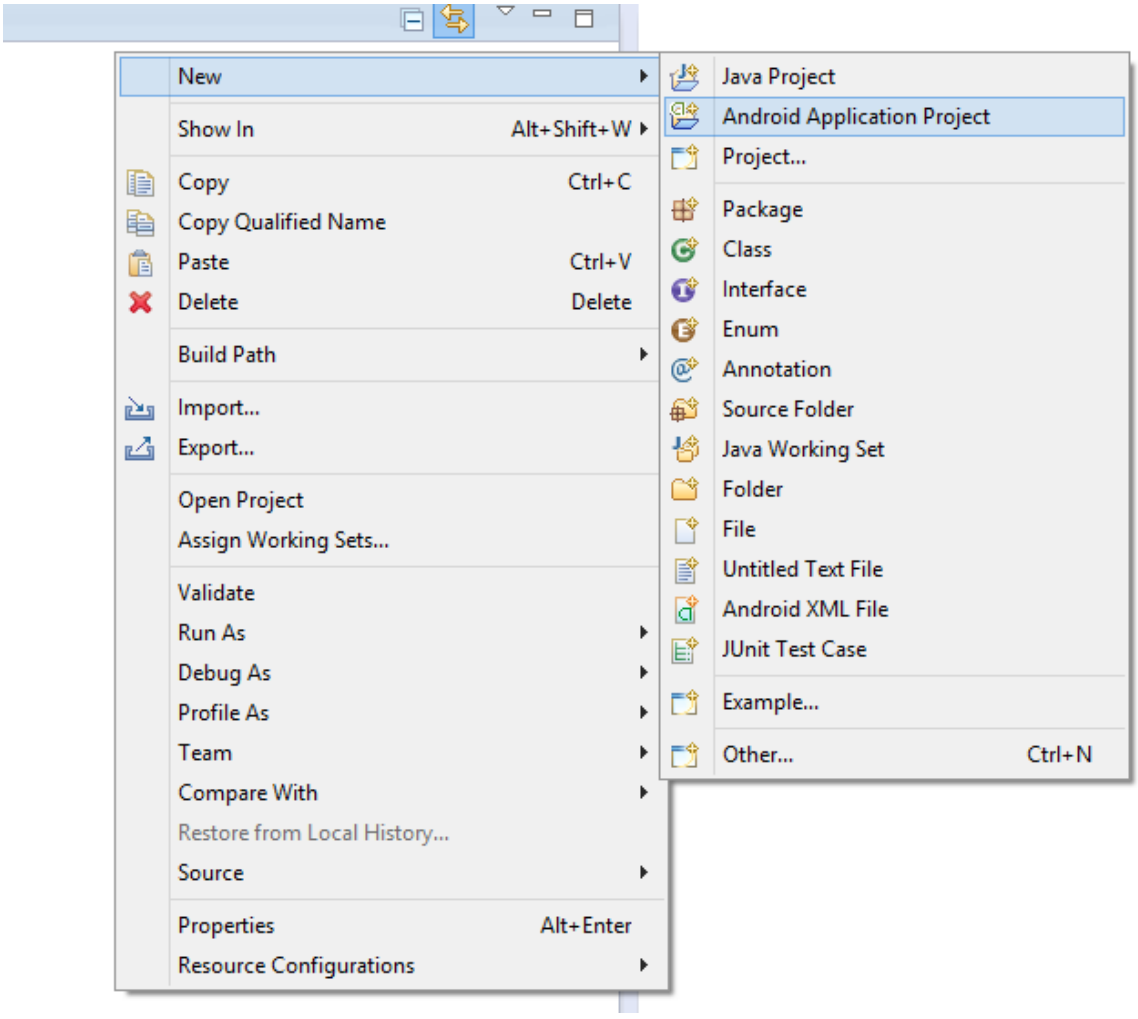
Le but de ce TP est d'expliquer les étapes nécessaires à la création et aux tests d'une application sous Android.

Dans un second temps, il vous permettra de configurer votre PC afin d'utiliser un périphérique physique comme cible en lieu et place d'une machine virtuelle Android.

Créer un projet Android

La version d'Eclipse fournie dans ADT est déjà complètement configurée afin de créer des applications pour Android.

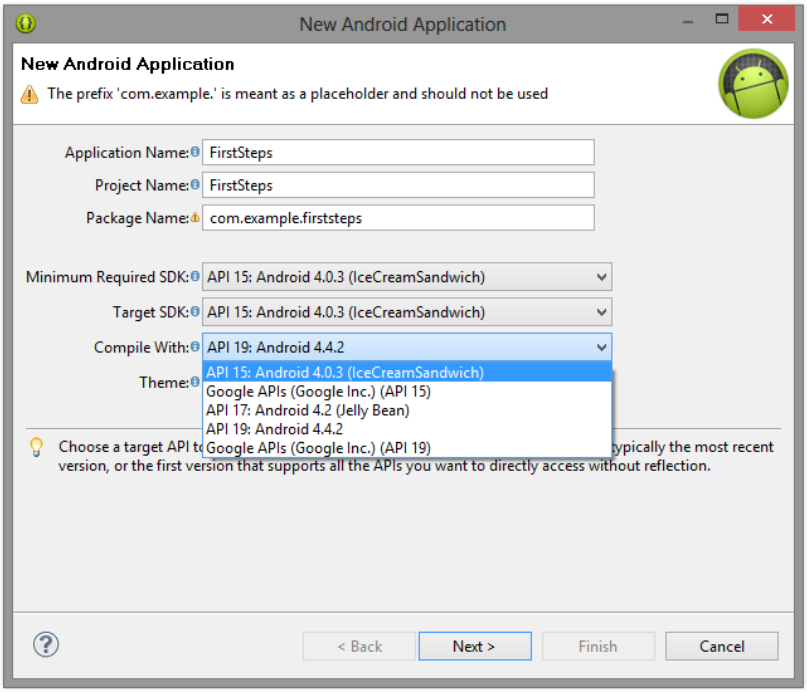
Ainsi, il suffit de lancer l'assistant de création à partir du menu contextuel : « New > Android Application Project ».



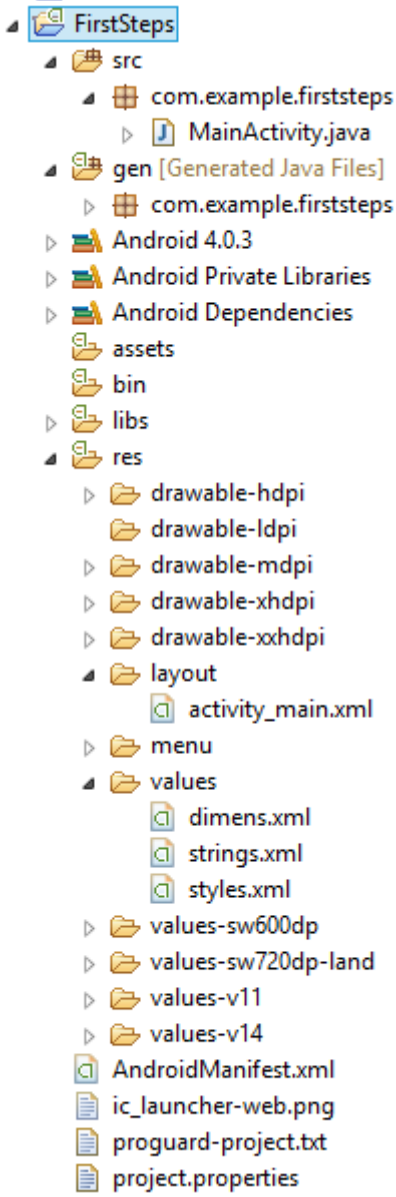
Assistant création

Lors de la création d'un projet Android, la première étape de l'assistant est la seule à comporter des valeurs obligatoire, le reste comportant des valeurs par défaut pouvant largement vous contenter dans un premier temps (en tout cas pour ces TP).

Après avoir saisi un nom d'application, veuillez choisir la bonne version du SDK à utiliser. Dans notre cas, nous allons travailler avec la version 4.0.3 (API 15).



Une fois le projet créé, vous devriez vous retrouver avec une arborescence de ce type :



Squelette d'un projet

« src »

Très classique dans un projet Java, ce dossier contient tous vos fichiers sources

« res »

Il contient toutes les ressources de votre application :

- Drawable: Les images (.png, .jpg, .gif) en fonction des résolutions
- Layout: Fichiers XML contenant la structure de vos interfaces
- Values: Fichiers XML qui contient diverses informations en lecture uniquement (Tableaux de String, ou d'Int, …)
- Menu: Description en XML de vos menus

« assets »

Ce dossier permet de stocker tout ce qui n'est pas possible d'ajouter dans le dossier « res ». Ainsi vos fichiers sqlite, txt devront se trouver dans ce dossier sous peine de ne pas être accessibles

R.java

Ce fichier contenu dans le dossier « gen » est très important dans le fonctionnement des applications Android, il est automatiquement généré et tient à jour une liste de vos ressources (layout, images, …) afin de pouvoir les utiliser dans votre code

Exemple:

R.layout.activity_main : Correspond au layout de l'application généré automatiquement
R.string.app_name : Le nom de votre application contenu dans le fichier « values/strings.xml »
R.drawable.ic_launcher : Correspond à l'icône de lancement de l'application (l'utilisation de la bonne résolution est automatique. Si vous décidez ne pas fournir vos ressources pour toutes les résolutions (hdpi, ldpi, mdpi …), le système s'occupera de prendre la version offrant la meilleure résolution en fonction des attentes du périphériques.

Il existe un « Android.R. » généré par le système permettant d'utiliser des ressources « standards » dans votre application de la même manière que les vôtres.

AndroidManifest.xml

Toutes les applications Android doivent avoir un fichier manifeste. Il contient les informations essentielles que le système doit connaître pour faire fonctionner l'application:

- Le nom du package java
- Les composants de l'application : activité, services, etc…
- Les permissions (Elles sont affichées lorsque vous achetez une application). Les prochains TP décriront le principe de ces permissions
- La plage des versions sur lequel votre application compte fonctionner
- Et bien d'autres…

Strings.xml

Ce fichier doit (devrait) contenir toutes les chaînes de caractères utilisées dans votre application. L'intérêt est de pouvoir, entre autres, faciliter la gestion multilingue.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">FirstSteps</string>
    <string name="action_settings">Settings</string>
    <string name="hello_world">Hello world!</string>

</resources>
```

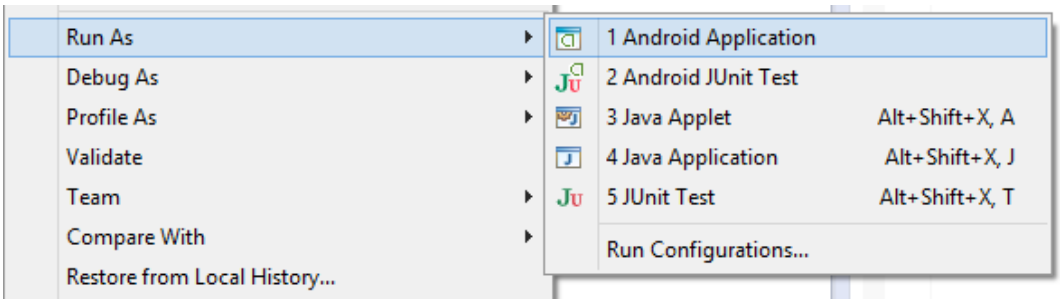
Si on souhaite utiliser la chaîne de caractère « Hello world ! » contenue dans « strings.xml », il vous suffit d'utiliser : « @string/hello_world » en XML, ou bien, « R.string.hello_world » en Java. Attention, il s'agit d'un identifiant de ressources (un entier) et non de la chaîne de caractères elle-même.

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

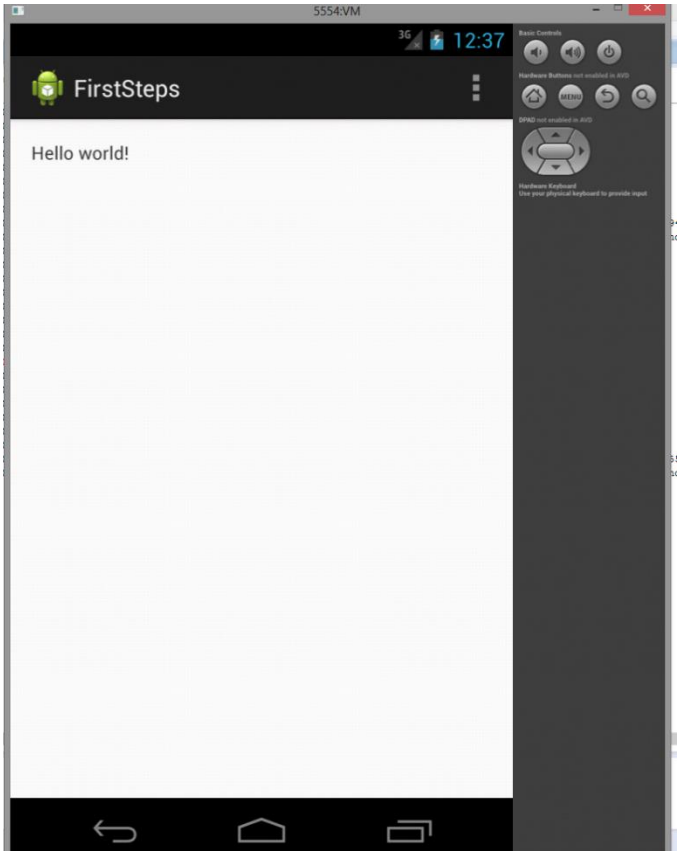
(« TextView » est un objet permettant d'afficher une chaine de caractère dans votre vue)

Compilation et Lancement

Afin de tester l'application, il vous suffit de la lancer grâce au menu contextuel « Run As > Android Application ». :



Après le lancement de la machine virtuelle, vous devriez voir apparaître votre application comme ci-dessous :



Utiliser un périphérique externe

Afin d'accélérer le développement de votre application, il est vivement conseillé d'utiliser un périphérique physique comme cible pour le lancement de l'application. Cette étape requiert très peu de manipulation.

Windows

Installation du pilote

Dans notre cas, nous utilisons des téléphones « Orange » gracieusement fournis par Intel. Les pilotes pour ce type de téléphone sont disponibles à cette adresse :

http://software.intel.com/en-us/android/articles/intel-android-device-usb-driver-end-user-license-agreement

Après installation, le téléphone devrait être immédiatement et automatiquement reconnu par Eclipse.

Notez que l'outil « adb » contenu dans le dossier « sdk\platform-tools », permet de lister les périphériques connectés (et ainsi permettre de vérifier s'il est reconnu) :

```
C:\adt-bundle-windows-x86_64-20131030\sdk\platform-tools>adb.exe devices
List of devices attached
MedfieldD94CF362        device
```

Linux

Sous linux, il est plus simple de faire reconnaître un périphérique par le système. Les étapes ci-dessous nécessitent les droits root.

Dans un premier temps, vous devez déterminer l'ID du téléphone utilisé en utilisant « lsusb ». Dans notre cas, il s'agit de la première ligne « […] 8087:09fc Intel Corp. ».

```
[root@vm-arch steven]# lsusb
Bus 002 Device 002: ID 8087:09fc Intel Corp.
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 003: ID 0e0f:0002 VMware, Inc. Virtual USB Hub
Bus 001 Device 002: ID 0e0f:0003 VMware, Inc. Virtual Mouse
Bus 001 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
```

Une fois l'id trouvée, vous devez créer une nouvelle règle pour qu'udev puisse reconnaître votre périphérique (où « idVendor » et « idProduct » correspondent à votre périphérique) :

```
[root@vm-arch steven]# echo 'SUBSYSTEM=="usb",
ATTRS{idVendor}=="8087",ATTRS{idProduct}=="09ef", MODE="0666"' >
/etc/udev/rules.d/51-android.rules

[root@vm-arch steven]# systemctl restart systemd-udevd
```

Après avoir redémarré udev et rebranché votre téléphone, il devrait être reconnu par Eclipse.

Exécutez « adb devices » contenu dans « sdk/platform-tools » pour s'en assurer :

```
[root@vm-arch platform-tools]# ./adb devices
* daemon not running. starting it now on port 5037 *
* daemon started successfully *
List of devices attached
MedfieldD94CF362     device
```

Ces étapes sont aussi disponibles sur le site d'android : http://developer.android.com/tools/device.html