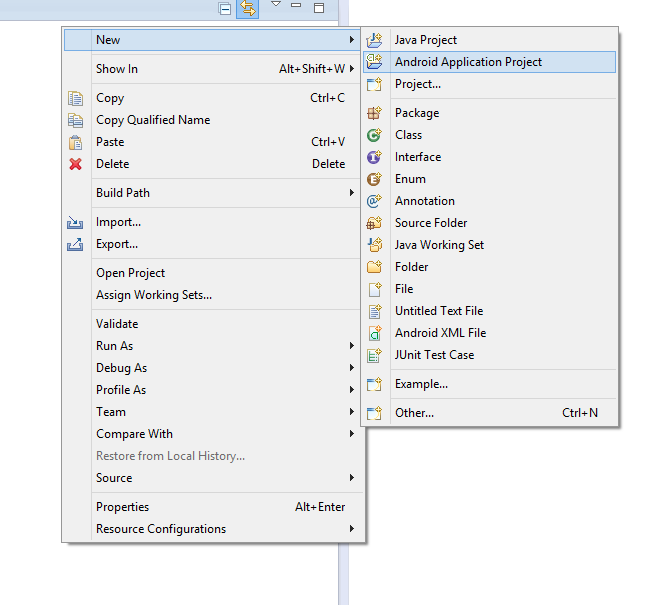
# Les premiers pas avec Android

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

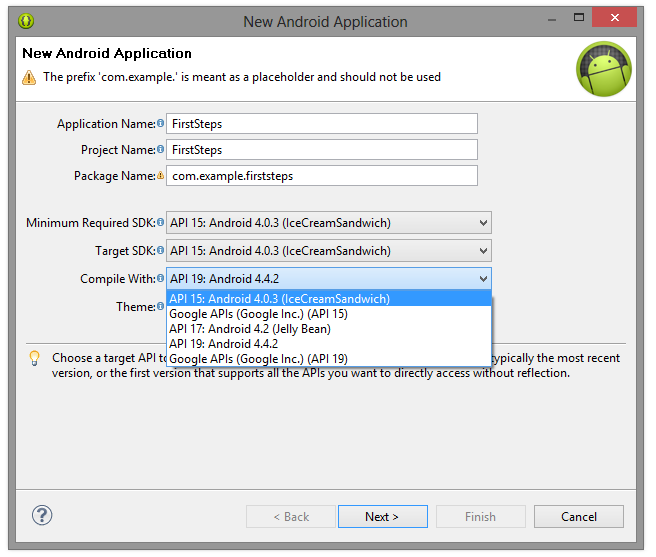
La version d’Eclipse fournie dans ADT est déjà complètement configurée afin de créer des applications pour Android.

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 :

#### C:\Users\Steven\Dropbox\Captures d'écran\Capture d'écran 2014-02-23 12.45.50.pngSquelette 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 accessible.

#### R.java

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

Exemple:

R.layout.activity\_main : Correspondant 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 : Correspondant à l’icône de lancement de l’application (l’utilisation de la bonne résolution est automatique. Si 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 connaitre 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 décrirons le principe de ces permissions.
  + Ainsi que la plage des versions sur lequel votre application compte fonctionner.
  + Et bien d’autres…

#### Strings.xml

Ce fichier doit (devrait) contenir toutes les chaines de caractères utilisées dans votre application. L’intérêt est de pouvoir, entre autres, faciliter la gestion des multi langues.



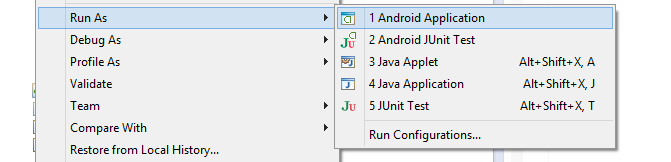
Si on souhaite utiliser la chaine de caractère « Hello world ! » contenue dans « strings.xml », il vous suffit d’utiliser : « @string/hello\_world » en XML, ou bien, « R.string.hello\_world » en Java. Attention il s’agit d’un identifiant de ressources (un entier) et non la chaine de caractères elle-même.



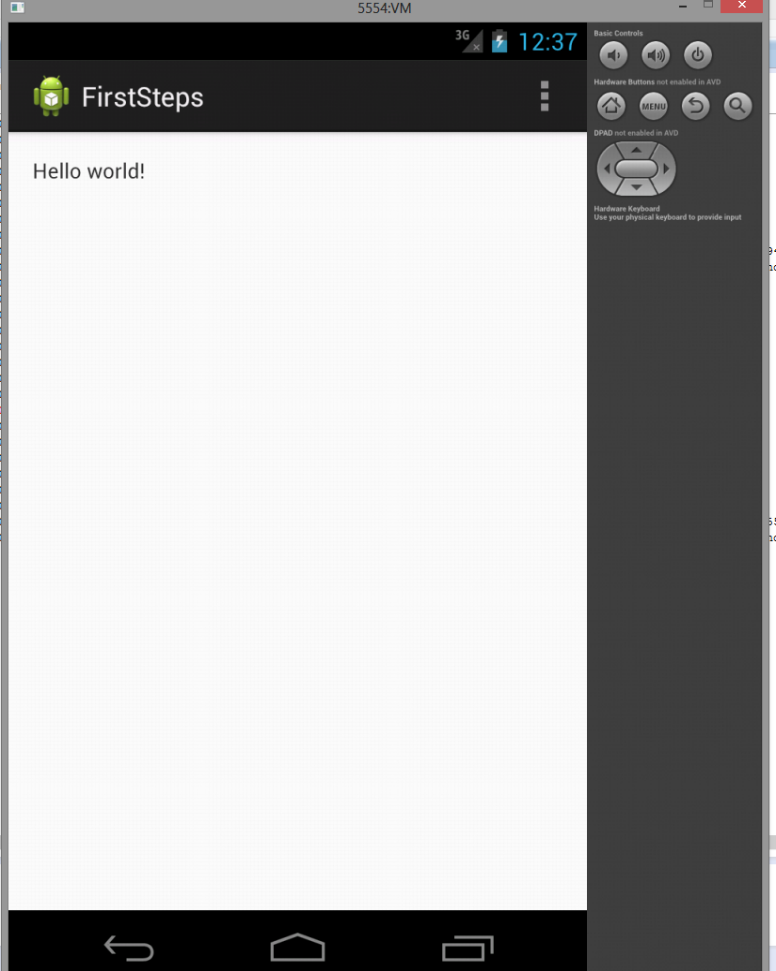
(« TextView » est un objet permettant d’afficher une chaine de caractère dans votre vue).

### 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 apparaitre 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 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 l’installation, le téléphone devrait être immédiatement et automatiquement reconnu par Eclipse.

En utilisant 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) :



### Linux

Sous linux, il est plus simple faire reconnaître un périphérique au 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. ».



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) :



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 :



*Ces étapes sont aussi disponibles sur le site d’android :* [*http://developer.android.com/tools/device.html*](http://developer.android.com/tools/device.html)