Introduction à la programmation sous Android

Tutos-android

par Feanorin

Date de publication : 08 avril 2011

Dernière mise à jour :

Ce tutoriel a pour but de vous présenter succinctement Android, ainsi que les prémices de la programmation sous celui-ci.

I - Android, c'est quoi ?

Android est un OS mobile Open Source pour smartphone, PDA, MP3 et tablette. Conçu initialement par Android Inc , il a été racheté par Google en 2005.
Pour commencer la programmation Android, il faut d'abord installer le **SDK Android** et comprendre les bases de la programmation sous Android. Puis nous allons faire notre premier programme sous Android c'est-à-dire le bien connu « **Hello Word** » pour bien comprendre ces bases.

II - Composantes d'une application Android

Une application Android est composée d'éléments de base :

2.1 - Activities (Activités en français)

Une activité est la composante principale pour une application Android. Elle représente l'implémentation métier dans une application Android.
Prenant l'exemple d'une application qui liste tous vos fichiers mp3 présents dans votre téléphone, le projet pourrait se décomposer comme ci-dessous :

- une vue pour afficher la liste des mp3 ;
- une activité pour gérer le remplissage et l'affichage de la liste ;
- si l'on veut pouvoir rajouter, supprimer des mp3, on pourrait rajouter d'autres activités.

2.2 - Services

Un service, à la différence d'une activité, ne possède pas de vue mais permet l'exécution d'un algorithme sur un temps indéfini. Il ne s'arrêtera que lorsque la tâche est finie ou que son exécution est arrêtée.
Il peut être lancé à différents moments :

- au démarrage du téléphone ;
- au moment d'un événement (arrivée d'un appel, SMS, mail, etc.) ;
- lancement de votre application ;
- action particulière dans votre application.

2.3 - Broadcast and Intent Receivers

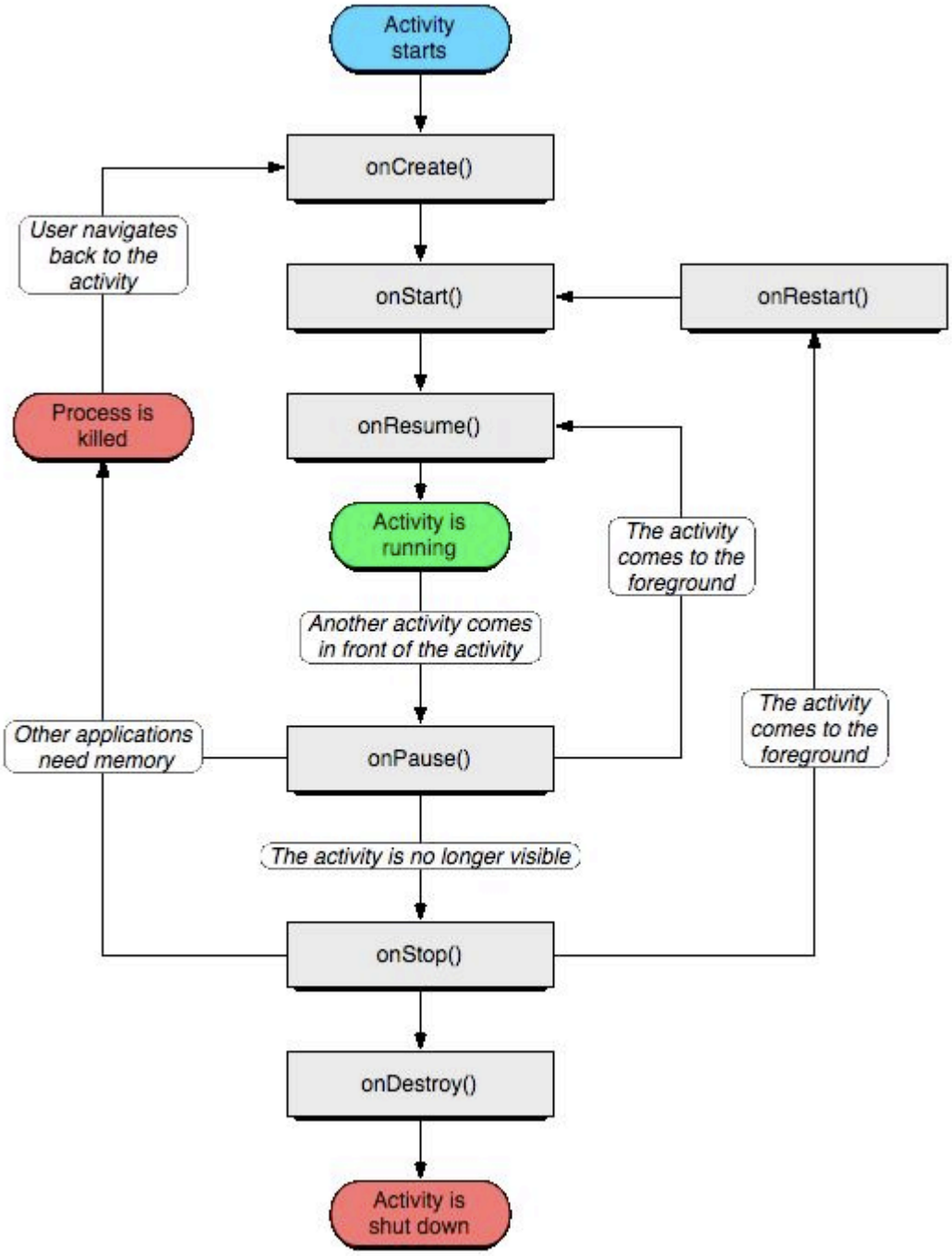
Un Broadcast Receiver comme son nom l'indique permet d'écouter ce qui se passe sur le système ou sur votre application et déclencher une action que vous aurez prédéfinie. C'est souvent par ce mécanisme que les services sont lancés.

2.4 - Content providers

Les « content providers » servent à accéder à des données depuis votre application. Vous pouvez accéder :

- aux contacts stockés dans le téléphone ;
- à l'agenda ;
- aux photos ;
- ainsi qu'à d'autres données depuis votre application grâce aux content providers.

III - Cycle de vie d'une application Android



3.1 - OnCreate

Cette méthode est appelée à la création de votre activité (Activity). Elle sert à initialiser votre activité ainsi que toutes les données nécessaires à cette dernière.
Quand la méthode **OnCreate** est appelée, on lui passe un Bundle en argument. Ce Bundle contient l'état de sauvegarde enregistré lors de la dernière exécution de votre activité.

3.2 - onStart

Cette méthode est appelée dans le cas où votre application est en arrière-plan et qu'elle repasse en avant-plan.
Si votre activité ne peut pas aller en avant-plan quelle que soit la raison, l'activité sera transférée à **OnStop**.

3.3 - onResume

Cette méthode est appelée après **OnStart** (au moment où votre application repasse en avant-plan).
OnResume est aussi appelée quand votre application passe en arrière-plan à cause d'une autre application.

3.4 - onPause

Appelée juste avant qu'une autre activité que la vôtre passe en **OnResume**. À ce stade, votre activité n'a plus accès à l'écran, vous devez arrêter de faire toute action en rapport avec l'interaction utilisateur. Vous pouvez par contre continuer à exécuter des algorithmes nécessaires mais qui ne consomment pas trop de CPU.

3.5 - onStop

Appelée quand votre activité n'est plus visible quelle que soit la raison.

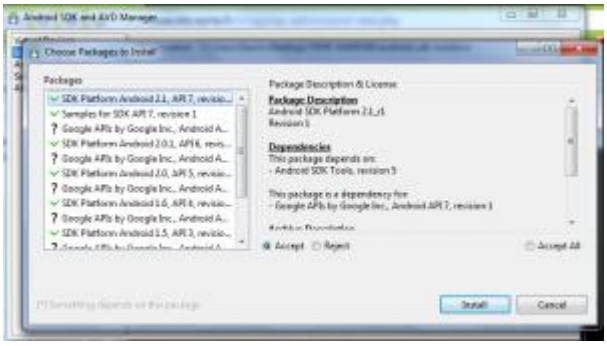
3.6 - onDestroy

Appelée quand votre application est totalement fermée (Processus terminé).

IV - Installer votre environnement de développement

4.1 - Installation du SDK Android

1. Pour commencer allez sur le lien suivant : **http://developer.android.com/sdk/index.html** et téléchargez la version du SDK qui convient à votre OS. Pour la suite on est sur un Windows 7.
2. Vous avez le choix entre la version Zip ou la version exe. Dans cet exemple on a pris la version Zip.
3. Une fois le SDK téléchargé, allez dans le dossier où se trouve le fichier Zip et l'extraire dans le dossier de votre choix.
4. Lancez l'exécutable « SDK Setup » qui se trouve à la racine du dossier.
5. La fenêtre suivante apparait (la dernière version actuelle du SDK est 2.3) :



6. Choisissez ce que vous voulez installer. Par exemple :

- **SDK Platform Android 2.x** : correspond tout simplement au SDK Android basique en version 2.x ;
- **Samples for SDK API 7** : correspond à quelques exemples ;

- **Android + Google APIs** : correspond au SDK Android (1re option) + Google Api qui inclut différentes fonctions comme GoogleMap, etc. ;
- **Galaxy Tab** : c'est le SDK pour la tablette Samsung Galaxy Tab.
- En cas de problème d'installation, allez dans « **Settings** » et cochez « **Force https:// sources to be fetched using http://** ».

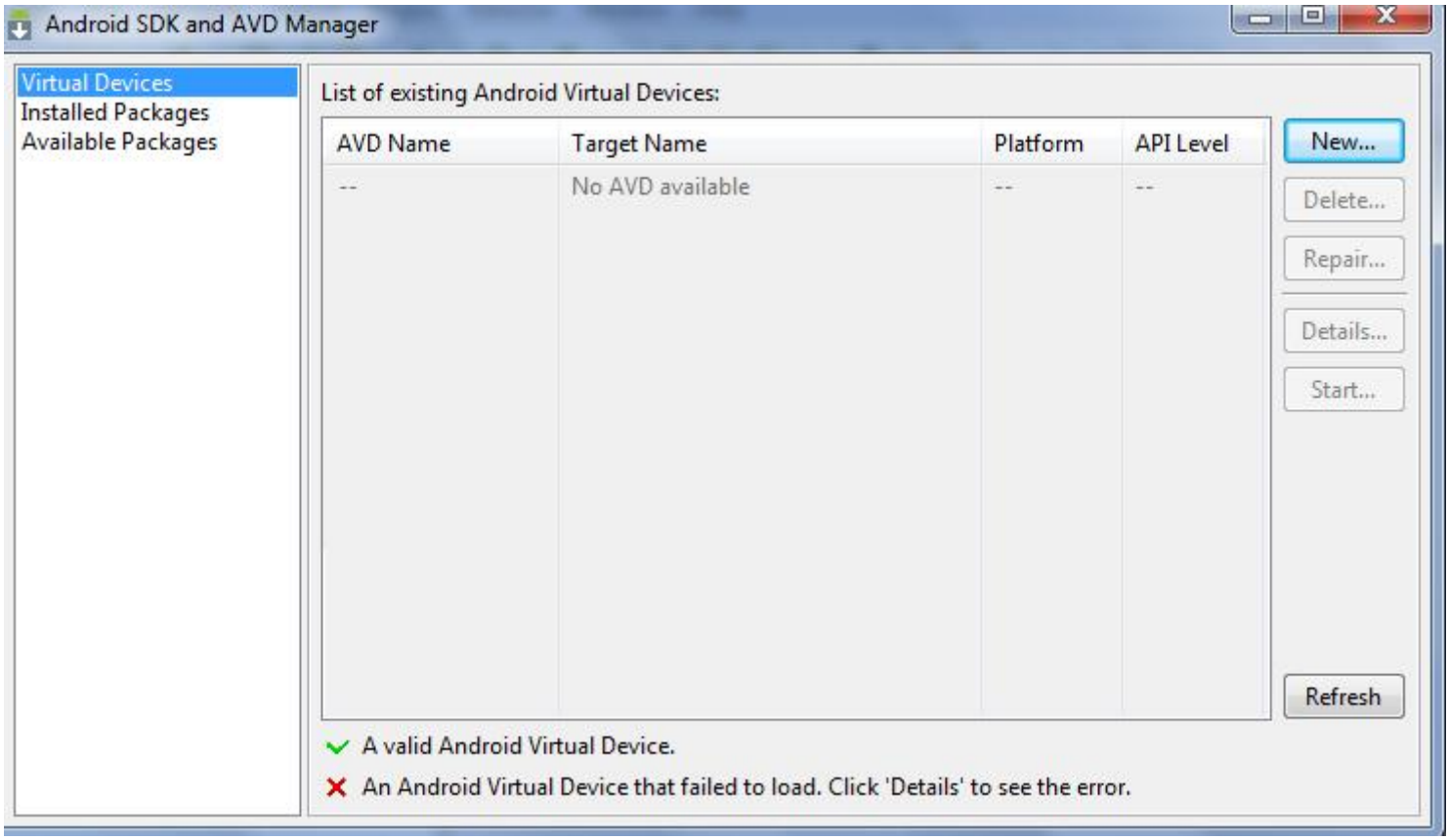
7. Cliquez sur « Install ».
8. Installez Eclipse : **http://www.eclipse.org/downloads/** (téléchargez une version comprenant Java, soit la version pour les développeurs Java ou la version pour les développeurs J2EE) : version installée pour ce tutoriel : 3.6.1 Helios.
9. Il faut aussi installer JDK (Java Development Kit) et JRE (Java Runtime Environement) si ce n'est pas déjà fait : **http://java.sun.com/javase/downloads/index.jsp**.
10. Lancez votre Eclipse, allez dans le menu « Help and Install New Software »
11. Dans la partie « Available Software », cliquez sur « Add ».
12. Rajoutez le nom du site (« ADT plugin » par exemple). Dans la location rajoutez : **https://dl-ssl.google.com/android/eclipse/**, puis cliquez sur OK.
13. Revenez dans « Available Software », vous devez voir « Developer Tools ». Sélectionnez-le et appuyez sur « Next ».
14. Cliquez sur « Next » pour lire et accepter la licence et puis cliquez sur « Finish ».
15. Pour finir l'installation relancez Eclipse.

Voilà votre environnement de développement est prêt à être utilisé.

4.2 - Configuration de votre environnement de développement

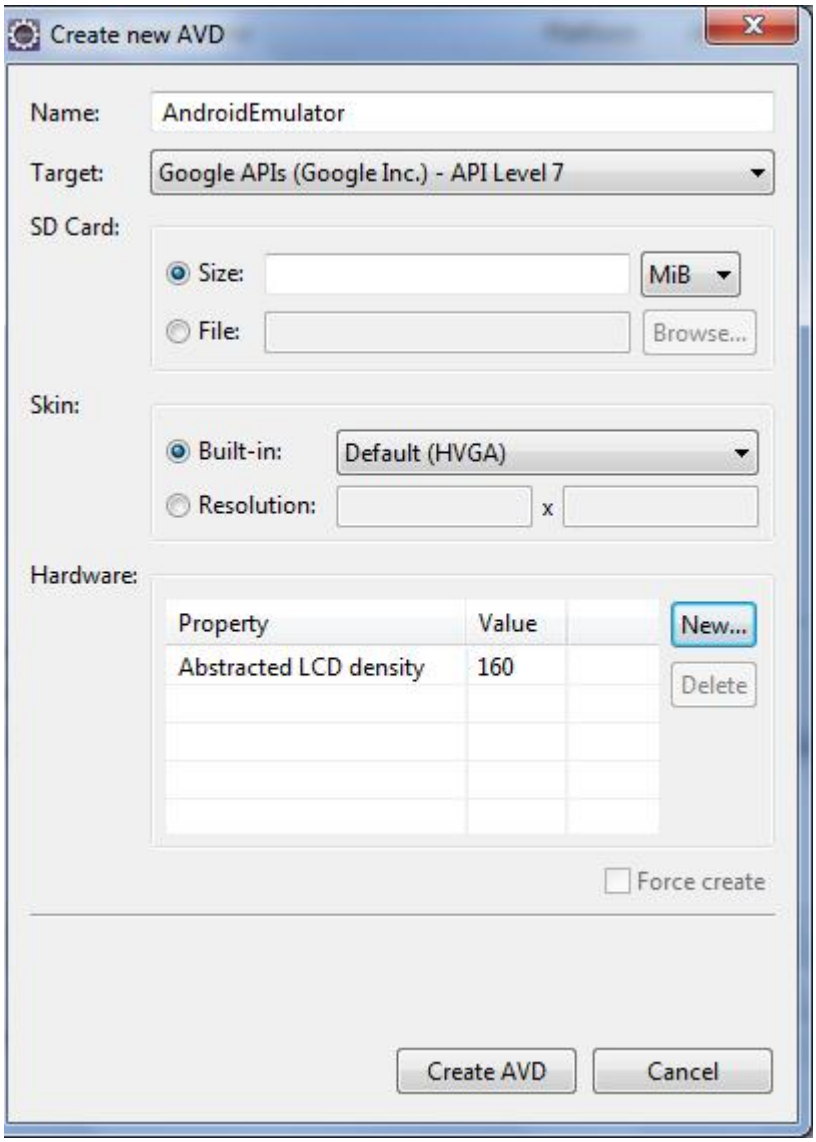
Vous avez dû remarquer qu'un nouvel élément est apparu dans votre Eclipse dans le menu du haut (un petit Android qui sort d'une boîte) :

Cliquez sur l'icône en question, une nouvelle fenêtre va apparaître.



Cet écran vous permettra de :

- installer de nouveaux paquets (Available Packages) ;
- mettre à jour vos paquets ;
- voir les paquets déjà installés (Installed Packages) ;
- créer votre émulateur Android et cela grâce à l'onglet « Virtual Devices ». Cliquez sur le bouton « New ». Une nouvelle fenêtre pour la création de votre émulateur apparaîtra.



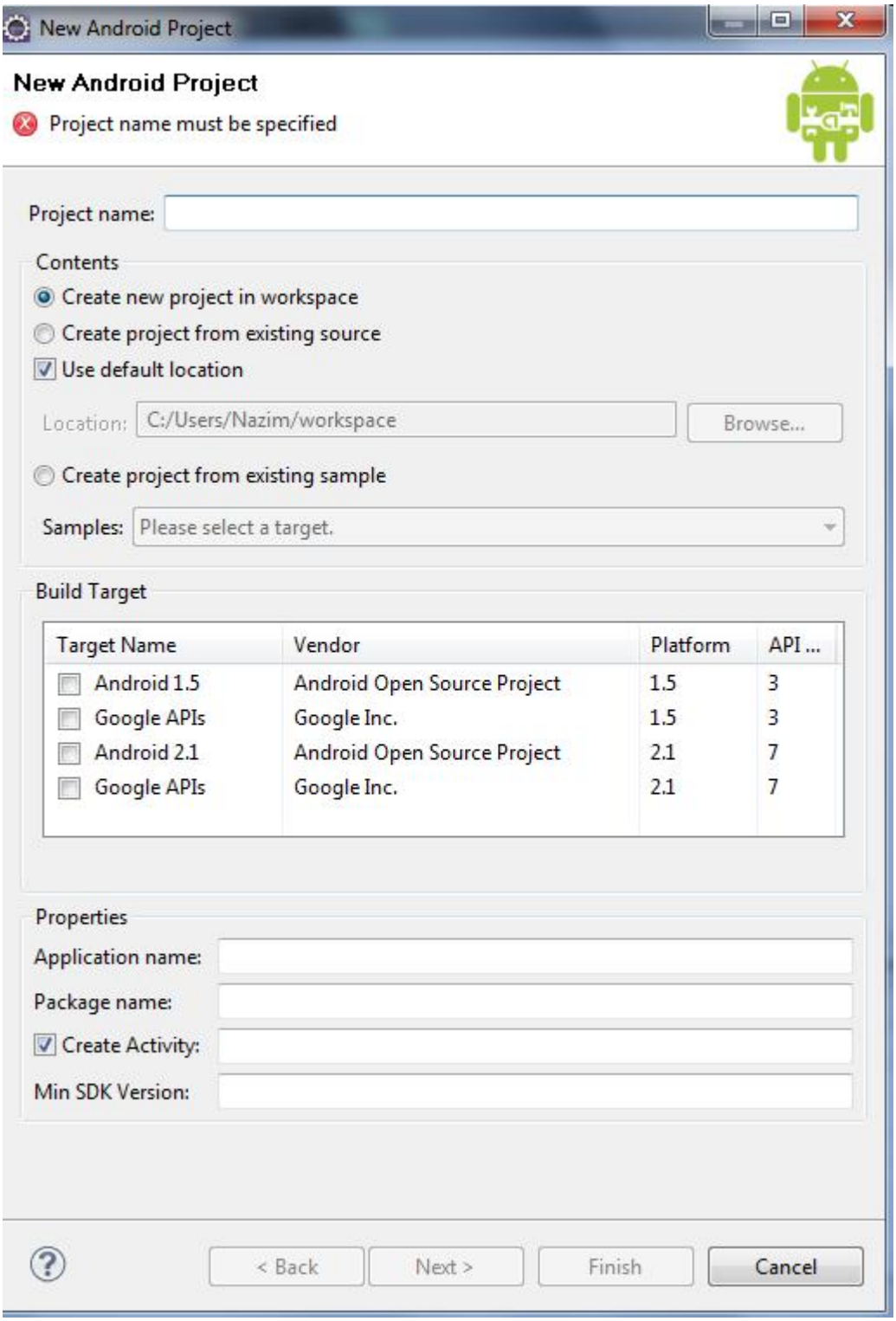
- **Name** : le nom de votre émulateur (sans espace).
- **Target** : version du SDK Android de l'émulateur.
- **SD Card (facultatif)** : configuration de la SD Card (Taille, etc.).
- **Skins** : choisissez la taille, résolution de votre émulateur. Des émulateurs préconfigurés se trouvent dans la partie Built-in.
- **Hardware** : cette partie permet de rajouter le matériel spécifique à votre émulateur. Par exemple vous pouvez rajouter un GPS, configurer le clavier, l'accéléromètre, etc.

L'émulateur apparaîtra maintenant dans la liste des émulateurs disponibles.
Maintenant on va passer à la partie la plus intéressante.

V - Ma première application sous Android

5.1 - Création du projet « Hello World »

Dans Eclipse, cliquez sur « File -> New -> Android Project ». La fenêtre ci-dessous s'affichera :



Remplissez les champs :

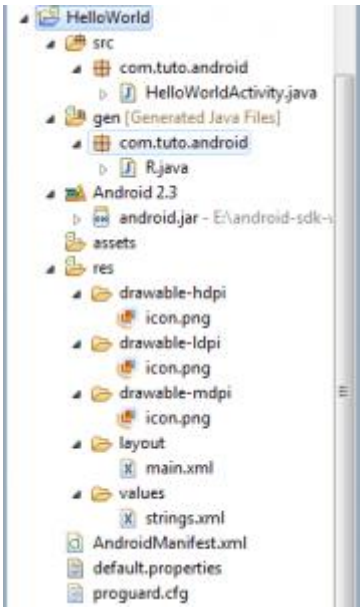
- **Project name** : le nom du projet. Pour notre exemple on choisira **Hello World**.
- **Build Target** : cochez la SDK que vous souhaitez. On prendra **Android 2.3**.
- **Properties** :
- -------**Application name** : le nom de l'application. On choisira Hello World ;
- -------**Package name** : le nom du package principal de l'application. Il faut que ce dernier comporte au moins deux identifiants séparés par des points. On prendra com.tuto.android ;
- -------**Create Activity** : si vous laissez coché, vous devez spécifier un nom pour l'activité de base de votre application. Nous choisirons HelloWorldActivity ;

- -------**Min SDK Version** : vous pouvez spécifier quelle version minimum du SDK est nécessaire pour le fonctionnement de votre application. Ce champ est facultatif.

Puis cliquez sur « Finish », le projet « **Hello World** » va apparaître dans l'arborescence d'Eclipse.

5.2 - Explication de l'arborescence du projet

Voici le résultat de la création de votre projet et l'arborescence de ce dernier



- **src** : ce dossier contient les sources de votre application (code JAVA) et les packages.
- **com.tuto.android** : un package de votre application. Bien sûr, vous pouvez avoir plusieurs packages dans votre application.
- **HelloWorldActivity.java** : notre principale activité. (je vous conseille d'avoir plusieurs activités pour les différentes parties de votre code).
- **gen** : dossier qui contiendra le fichier R.java (ce fichier est généré automatiquement à partir de vos vues et fichiers de ressource).
- **R.java** : ce fichier est automatiquement généré par le SDK Android à chaque précompilation.
- **assets** : contient des données qui pourront être utilisées dans votre application (images, vidéos, licence, etc.).
- **res** : c'est le dossier qui contiendra les ressources de votre application (images, vidéos, styles).
- **drawable-hpdi** : contient toutes les images, bitmaps dont vous avez besoin pour votre application en haute résolution.
- **drawable-ldpi** : contient toutes les images, bitmaps dont vous avez besoin pour votre application en basse résolution.
- **drawable-mdpi** : contient toutes les images, bitmaps dont vous avez besoin pour votre application en moyenne résolution.
- **Icon.png** : l'icône de votre application, cette icône sera affichée sur le bureau.
- **layout** : le SDK Android offre une technique de création d'interfaces graphiques à l'aide de fichiers XML. C'est dans ce dossier que vous inclurez l'ensemble des fichiers décrivant vos interfaces. Vous pouvez créer d'autres dossiers pour les menus par exemple.
- **Main.xml** : le fichier principal de votre interface.
- **values** : ce dossier contient un ensemble de fichiers décrivant les valeurs (pseudovariables) utilisées par votre application. On peut, par exemple, y mettre des chaînes de caractères (strings.xml), des tableaux (arrays.xml), des entiers, des couleurs, etc.
- **Strings.xml** : fichier qui contient vos déclarations de chaînes de caractères.
- **AndroidManifest.xml** : définit le comportement de votre application au système Android. Ce fichier définit par exemple le nom, l'icône, la version min du SDK, les activités, les services, etc.

5.3 - Hello, World!

Le projet exemple créé de base par Eclipse représente un « **Hello World!** ». Vous pouvez le lancer en tant qu'une application Android. Pour cela, il vous suffit de cliquer droit sur le projet, puis sélectionner l'option « Run As -> Android Application » et la l'émulateur devrait se lancer. L'émulateur prendra un peu de temps à se lancer la première fois (ne le fermez pas entre vos différentes modifications).
Voilà le deuxième écran qui devrait s'afficher si tout se passe bien (le premier est pareil mais avec juste écrit « Android »)

Notre Hello Word est bien fonctionnel mais reste encore à le comprendre. Allons voir le code pour comprendre ce qui se passe.

- **AndroidManifest.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
      xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.tuto.android"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon"
                  android:label="@string/app_name">
        <activity android:name=".HelloWorldActivity"
                  android:label="@string/app_name">
            <intent-filter>
              <action android:name="android.intent.action.MAIN" />
              <category
              android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

- La balise « **manifest** » à la ligne deux contient plusieurs arguments, le plus important est « package », qui donne le nom du package dans lequel se trouve votre activité principale.
- La balise « **application** » sert à la déclaration de différentes propriétés de votre application :
- -------**android:icon** : l'emplacement où se trouve l'icône de votre application ;
- -------**android:label** : le nom de votre application (il se trouve dans strings.xml).
- La balise « **activity** » permet de déclarer une activité, à chaque nouvelle activité il faut remettre cette balise.
- -------**android:name** : le nom de la classe Java qui représente l'activité. Le nom doit commencer par un . et on ne met pas le .java à la fin.
- -------**android:label** : le label de l'activité en question.
- -------**intent-filter** : c'est pour spécifier une action.
- ----------la sous-balise action est pour spécifier l'action à exécuter, dans notre cas c'est le main.
- ----------la sous-balise category est là pour spécifier la catégorie de l'action.

- ----------Voici un lien qui explique les différents types d'actions et de catégories : **http://developer.android.com/guide/topics/intents/intents-filters.html**.
- **strings.xml**

```
<xml version="1.0" encoding="utf-8">
<resources>
<string name="hello">
Hello World, HelloWorldActivity!
</string>
<string name="app_name">Hello World</string>
</resources>
```

- dans les balises **resources**, on met une balise string à chaque fois que l'on a besoin de déclarer une chaîne de caractères ;
- on déclare deux chaînes :
- -------la chaîne **hello** qui contiendra « Hello World, HelloAndroActivity! » qui est le message qui sera affiché dans l'application ;
- -------la chaîne **app_name** qui contient « Hello Andro » qui représente le nom de l'application.
- **main.xml**
- ------- On dispose de deux modes de visualisation.
- ----------**Onglet Layout** : mode visualisation et édition d'interface ;
- ----------**Onglet main.xml** : mode code source.
- -------On commence par une balise qui définit le layout : ici **LinearLayout**.
- -------Voici un lien pour la liste des différents layouts : **http://developer.android.com/reference/android/view/ViewGroup.html**.
- -------On déclare une composante **TextView** pour afficher du texte et on
- lui dit qu'elle doit afficher le contenu de **@string/hello**, donc de la
- variable hello qui est déclarée dans **strings.xml** c'est-à-dire « **Hello World, HelloWorldActivity!** ».
- **HelloAndroActivity.java**

```
package com.mti.android;

import android.app.Activity;
import android.os.Bundle;

public class HelloAndroActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

- Notre main activité. Elle doit hériter de la classe Activity ou d'une sous-classe de cette dernière.
- **http://developer.android.com/reference/android/app/Activity.html**.
- La méthode « **OnCreate** » est équivalente au main, elle est appelée à la création de votre vue.
- On appelle simplement le **OnCreate** de la classe mère puis on initialise la vue. Puis, on met dedans **R.layout.main**, c'est-à-dire la vue déclarée dans le fichier **main.xml**.
- À chaque fois que vous voyez « **R** » , c'est-à-dire que l'on utilise du code qui a été généré par les différents fichiers xml de ressources.
- « **R.layout** » : on va chercher la vue déclarée dans le dossier layout et qui s'appelle **main** donc notre **main.xml**.
- R.java

```
/* AUTO-GENERATED FILE.  DO NOT MODIFY.
 *
 * This class was automatically generated by the
 * aapt tool from the resource data it found.  It
```

```
 * should not be modified by hand.
 */

package com.mti.android;

public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int icon=0x7f020000;
    }
    public static final class layout {
        public static final int main=0x7f030000;
    }
    public static final class string {
        public static final int app_name=0x7f040001;
        public static final int hello=0x7f040000;
    }
}
```

Vous ne devez pas toucher à ce fichier, il est généré automatiquement à partir de vos fichiers qui se trouvent dans le dossier des ressources (res).

- Vous remarquez que toutes les variables déclarées dans strings.xml sont présentes, que l'interface déclarée dans main.xml aussi. Ce qui explique l'utilisation de la ligne R.layout.main dans le HelloWorldActivity.java ainsi que l'icône de l'application.

VI - Conclusion

Voilà on s'arrête ici pour ce premier tutoriel, d'autres tutoriels vont suivre très prochainement et aborderont des sujets plus approfondis.

VII - Remerciements

Je tiens à remercier tout particulièrement **Feanorin** qui a mis ce tutoriel au format Developpez.com.
Merci également à **ClaudeLELOUP** ainsi qu'à **jpcheck** d'avoir pris le temps de le relire et de le corriger.

VIII - Liens

Tutoriel origine sur Tutos-Android