**Créer des applications Android**

**Les outils nécessaires sont :**

* **Android SDK**
* **Eclipse**
* **Le plugin ADT de Eclipse**

**Outils: Préparation de l’environnement**

1. **Installation pluging ADT pour Androïd dans Eclipse**
2. **Installation d’un téléphone virtuel Androïd**

**Outils: ECLIPSE and co.**

1. **Création d’un projet ECLIPSE**
2. **Gestion d’une interface**
3. **Gestion des listes déroulantes**
4. **Gestion des listes déroulantes et des événements**
5. **Intégrer Google Map dans Android**

**Outils : NETBEANS and co.**

**1. Configurer NetBeans**

**2. Créer un projet Android**

**3. Gestion d’une interface**

**4. Gestion des listes déroulantes**

|  |  |
| --- | --- |
|  |  |

# outils : Préparation de l’environnement

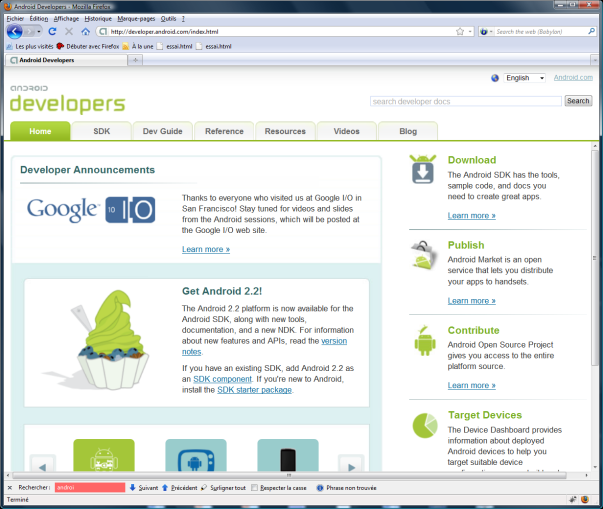
1. **Choix d’une version d’Eclipe**

Nous avons essayé plusieurs versions d’Eclipse et il apparait que la version Helios (celle utilisé dans ce tuto) est celle qui fonctionne le mieux pour développer un Projet Androïd.

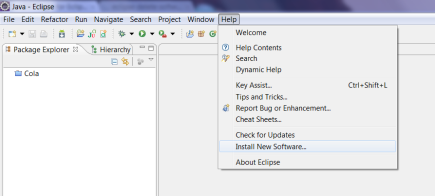
Voici le lien pour télécharger la version 32 bits que nous avons utilisé : http://fc.isima.fr/~phan/tuto/ApplicationAndroid/eclipse-java-helios-win32.zip

1. **Installation du plugin ADT pour Eclipse**

Cette partie est une reprise de l’explication de la page <http://developer.android.com/>



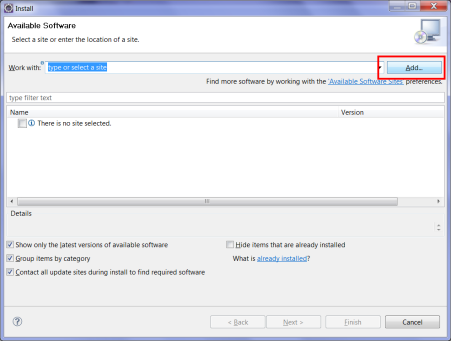
Installer un nouveau « Software ».

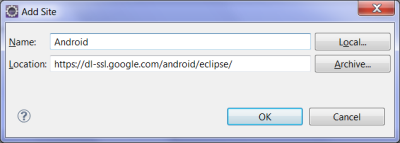


Ajouter un nouveau site.

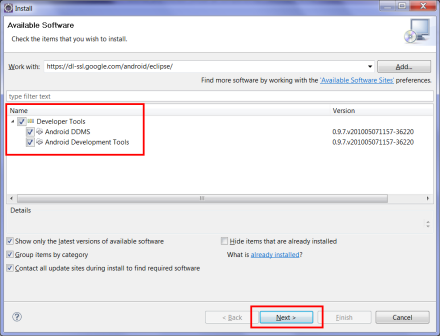
Nom : Android

Adresse : <https://dl-ssl.google.com/android/eclipse/>

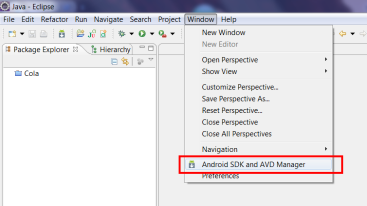




Sélectionner tout le package à télécharger. Et installer les.



Puis redémarrer Eclipse. Androïd devrait être installé et visible.



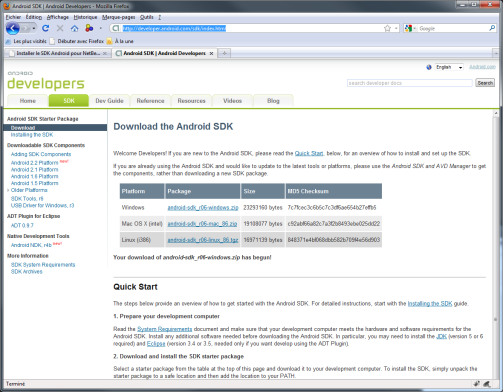
1. **Installation d’un téléphone virtuel Androïd**

**Remarque :**

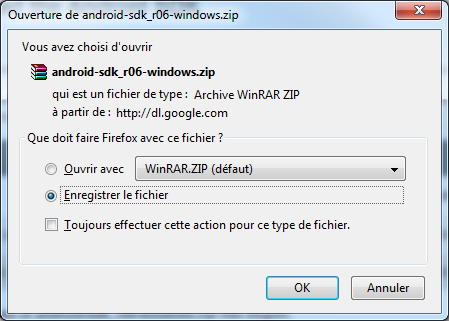
**Android SDK** est téléchargeable ici : <http://developer.android.com/sdk/index.html>

Ou bien ici : fc.isima.fr/~phan/tuto/ApplicationAndroid/android-sdk\_r06-windows.zip

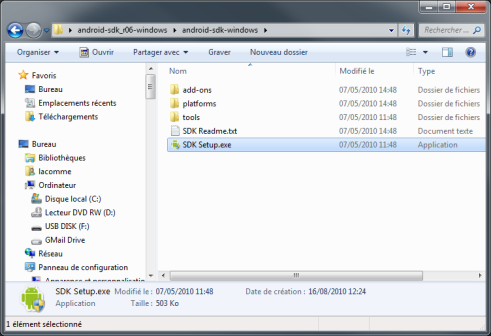
Son installation et configuration mérite quelques éclaircissements.



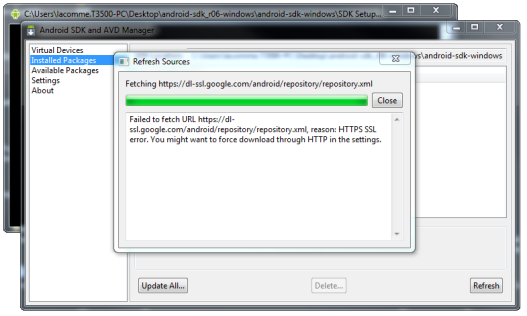
Il suffit de choisir la version correspondante à votre système, ici Windows :



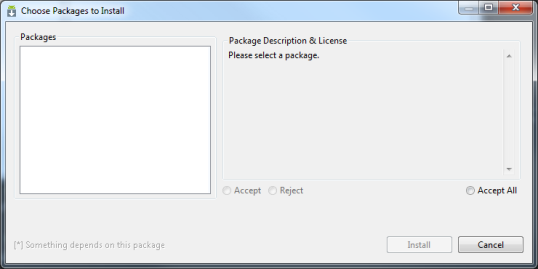
Ceci donne sur le disque dur un répertoire **android-sdk-windows** contenant un exécutable :



En lançant l’exécutable, il est probable que vous ayez une erreur de ce type :

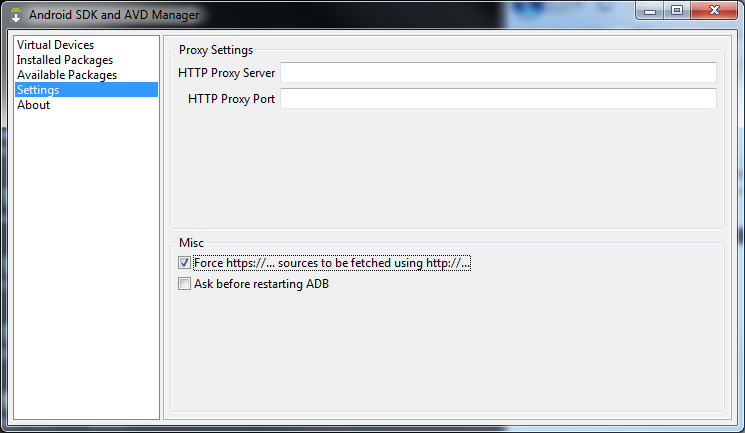


Fermer la fenêtre courante en utilisant le bouton **close**.

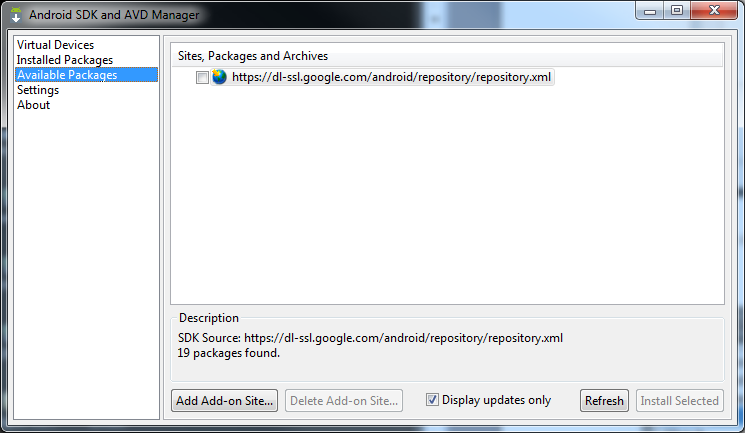


Fermer la deuxième fenêtre avec le bouton **Cancel**.

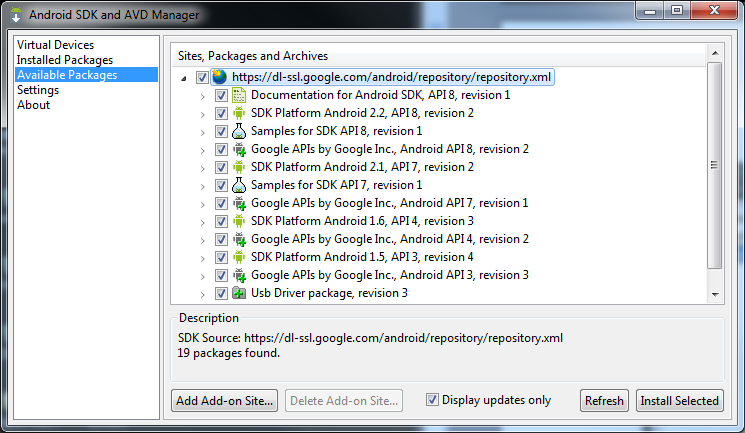
Allez dans le sous-menu **Settings** et cocher **Force https**



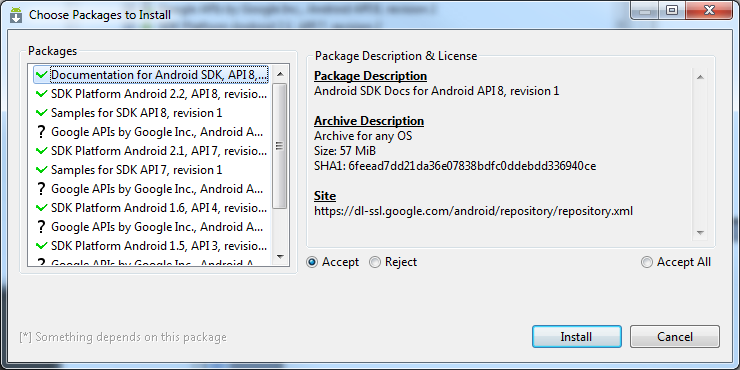
Choisir ensuite **Available Packages**.



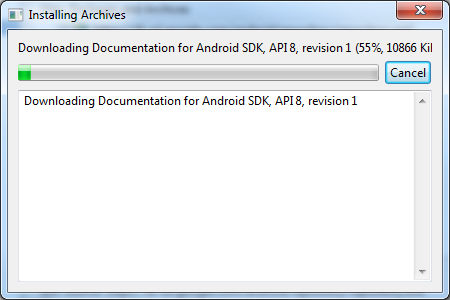
Cocher **https**.



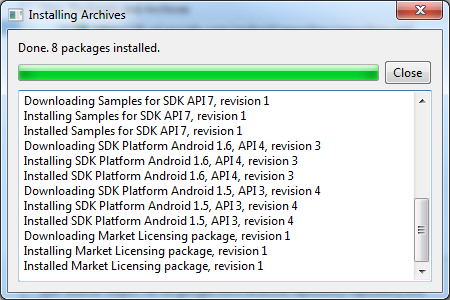
|  |  |
| --- | --- |
| Choisir ensuite |  |



Choisir **Install** et attendre la fin des téléchargements.

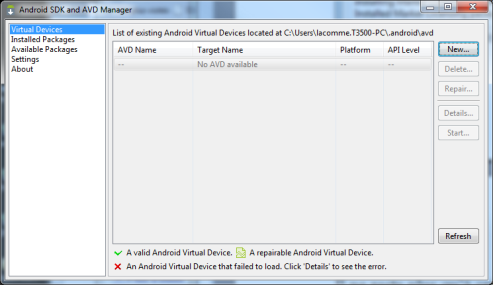


Cliquer sur **Close** à la fin des téléchargements.

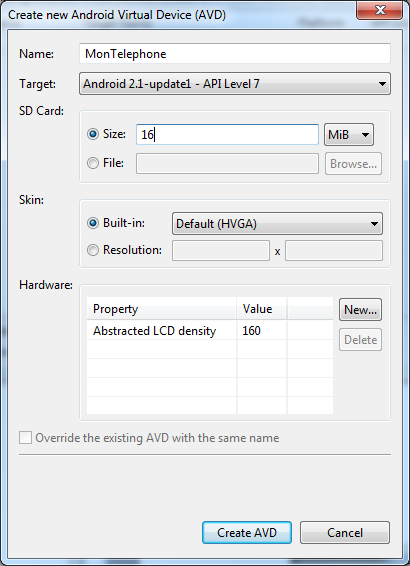


Il ne reste plus qu’à créer un Virtual Devices qui comme son nom l’indique sera un téléphone Android Virtuel.

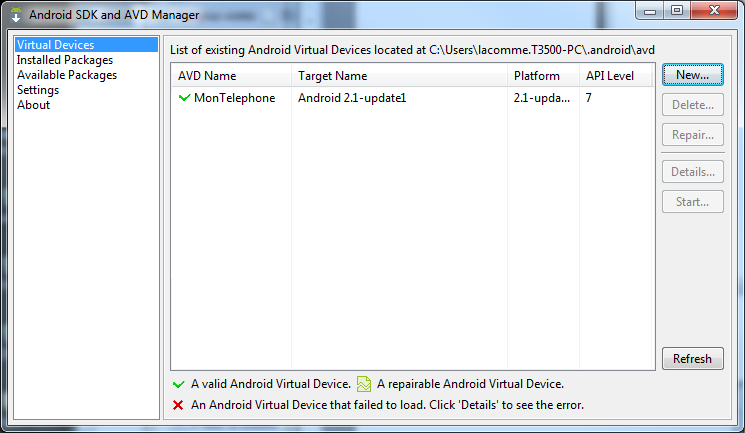
Choisir le menu **Virtual Devices**.



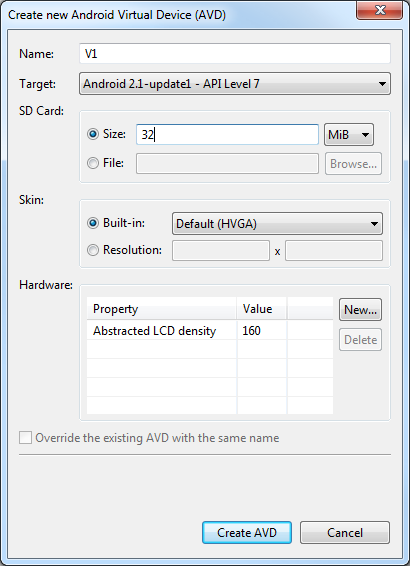
On peut créer sa propre configuration comme sur l’exemple ci-dessous :



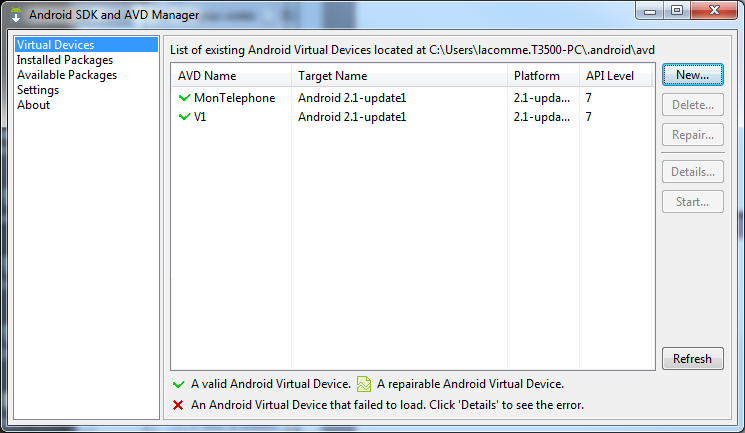
Le téléphone virtuel ainci crée doit apparaitre dans la liste des Virtual Devices.



En fonction de vos besoins, vous pouvez créer autant de téléphone virtuel que nécessaires. Ici par exemple, on crée un téléphone V1 en recommençant l’opération une deuxième fois.



Ce qui au final donnera :



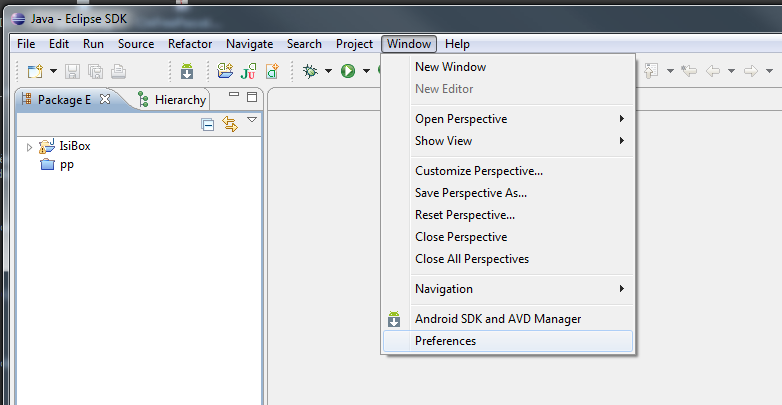
# outils : ECLIPSE and CO

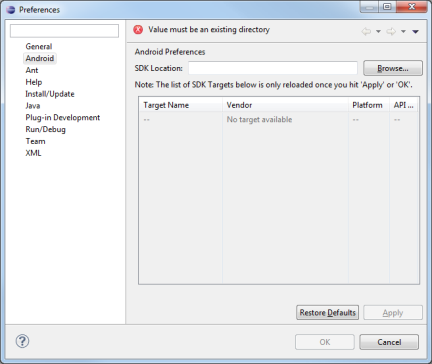
1. **Création d’un projet ECLIPSE**

Démarrer Eclipse.

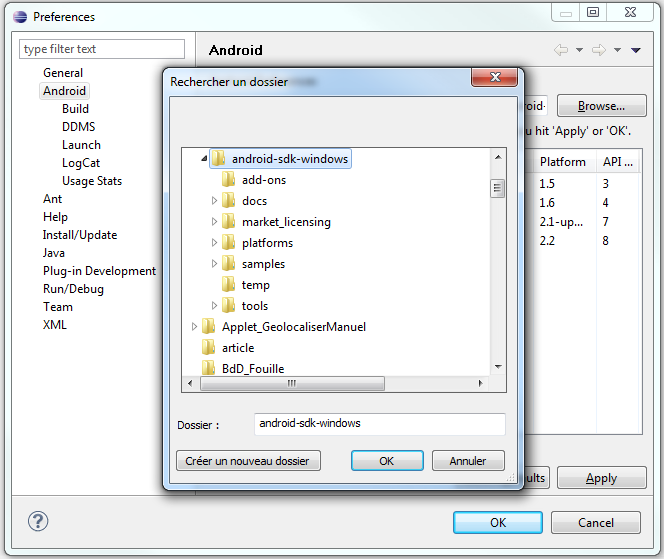


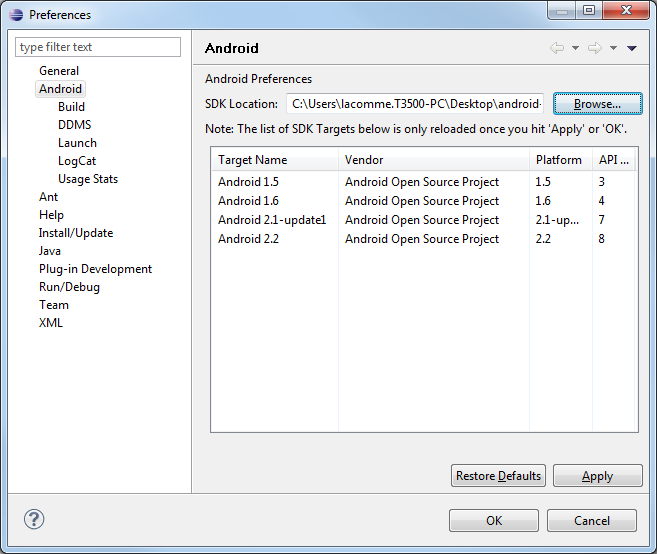
Allez dans **Windows / Preferences**.



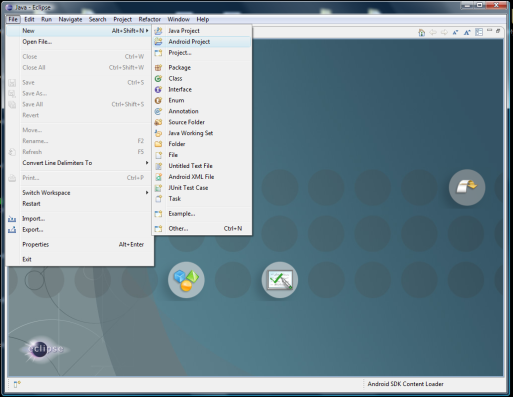


Choisir le répertoire **android-sdk-windows**





Créer un projet



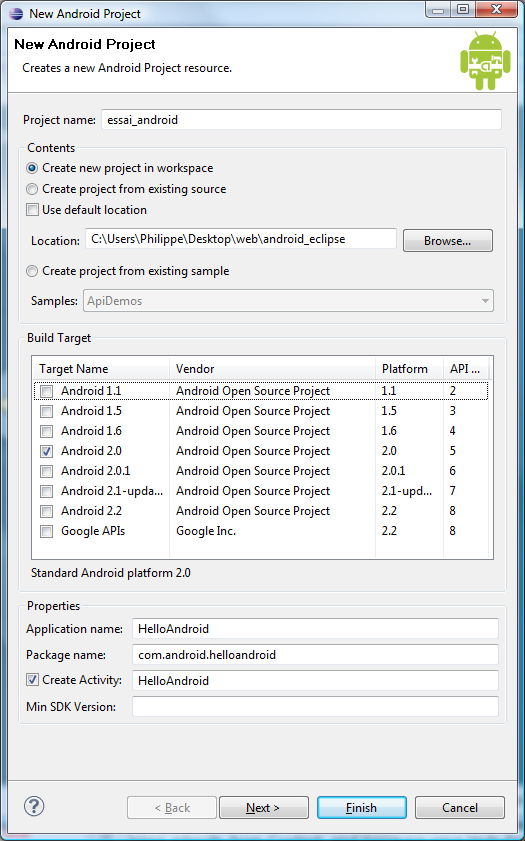
Choisir **essai\_android** comme nom du projet.

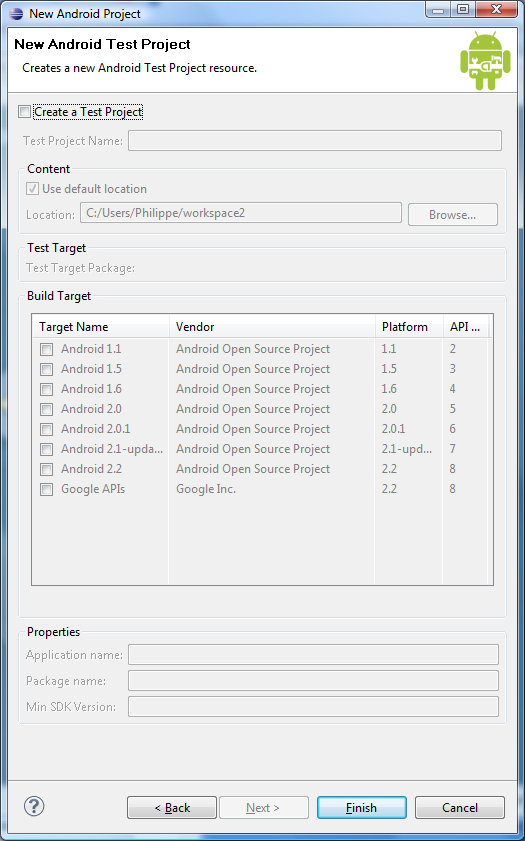
Et mettre les informations suivantes :

Application name : HelloAndroid

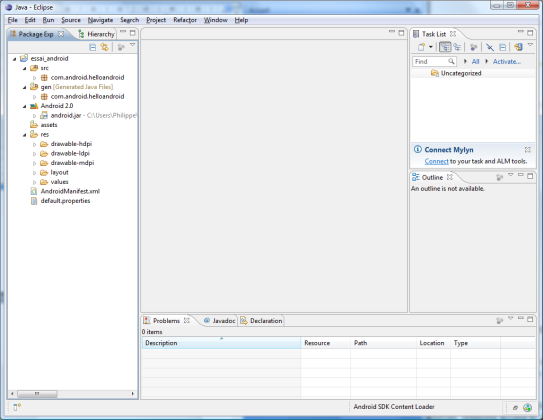
Package name : com.android.helloandroid

Create Activity : HelloAndroid

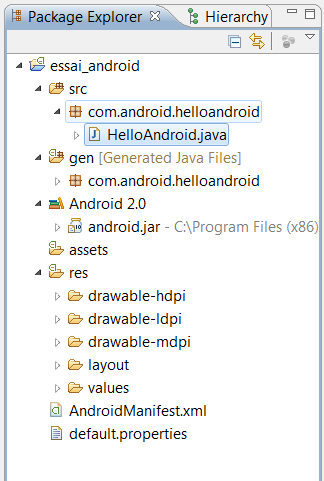




Ceci devrait donner :



Le projet se présente comme ceci :



Modifier le code du fichier HelloAndroid.java.

**package com.android.helloandroid;**

**import android.app.Activity;**

**import android.os.Bundle;**

**import android.widget.TextView;**

**public class HelloAndroid extends Activity {**

**/\*\* Called when the activity is first created. \*/**

**@Override**

**public void onCreate(Bundle savedInstanceState) {**

**super.onCreate(savedInstanceState);**

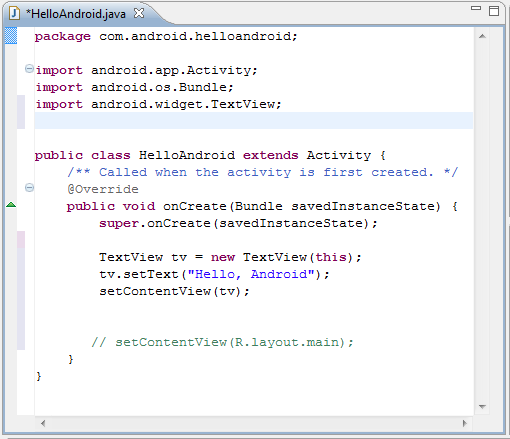
**TextView tv = new TextView(this);**

**tv.setText("Hello, Android");**

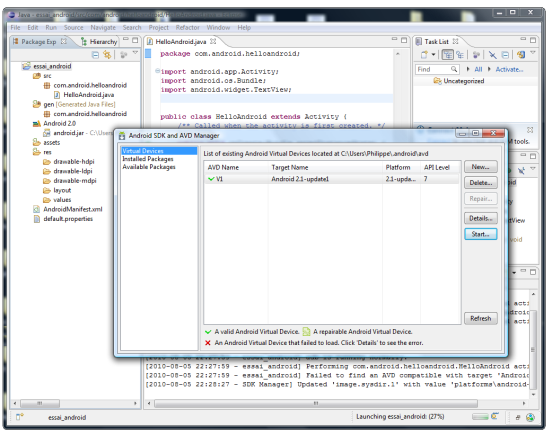
**setContentView(tv);**

**}**

**}**

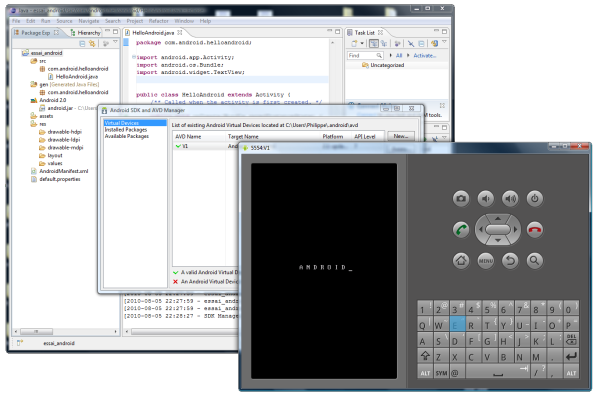


Faire **Run / Run**



Le gestionnaire d’AVD permet alors de choisir la machine virtuelle à utiliser.

Si tout se passe bien, l’écran de votre ordinateur se présente maintenant comme suit :



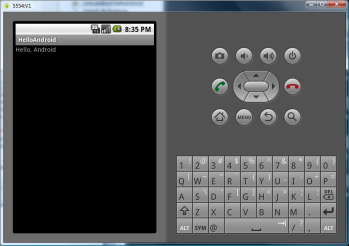
Attendre que le système Android démarre sur le téléphone.



Cliquer sur « MENU ».

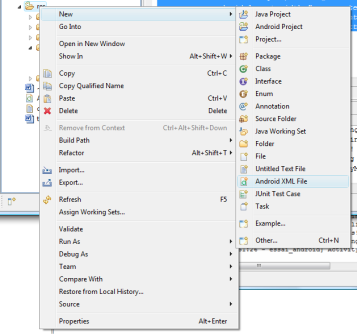


Ce qui donne :

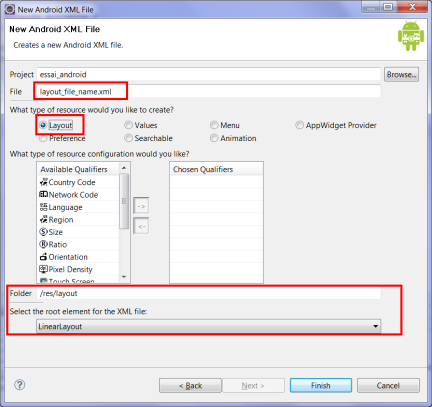


**3. Gestion d’une l’interface**

Faire un click droit sur **res**. Faire **New / Androïd XML File.**



Choisir comme nom : **layout\_file\_name.xml**



Saisir le texte suivant dans le fichier **layout\_file\_name.xml**

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

**<LinearLayout xmlns:android=*"http://schemas.android.com/apk/res/android"***

**android:layout\_width=*"fill\_parent"***

**android:layout\_height=*"fill\_parent"***

**android:orientation=*"vertical"* >**

**<TextView android:id=*"@+id/text"***

**android:layout\_width=*"wrap\_content"***

**android:layout\_height=*"wrap\_content"***

**android:text=*"Hello, I am a TextView"* />**

**<Button android:id=*"@+id/button"***

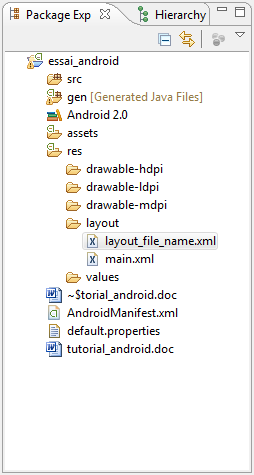
**android:layout\_width=*"wrap\_content"***

**android:layout\_height=*"wrap\_content"***

**android:text=*"Hello, I am a Button"* />**

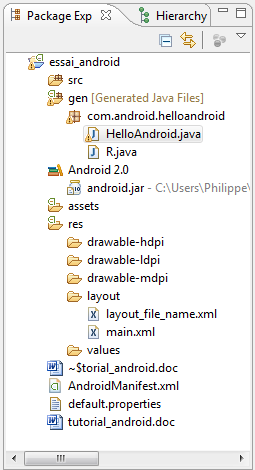
**</LinearLayout>**

Le projet doit se présenter comme suit :



Ouvrir ensuite le fichier nommé HelloAndroid.java

Ce fichier se trouve dans **gen/com.android.helloandroid**/



Remplacer le code par le code suivant :

**package** com.android.helloandroid;

**import** android.app.Activity;

**import** android.os.Bundle;

**import** android.widget.TextView;

**public** **class** HelloAndroid extends Activity {

/\*\* Called when the activity is first created. \*/

@Override

**public** void onCreate(Bundle savedInstanceState) {

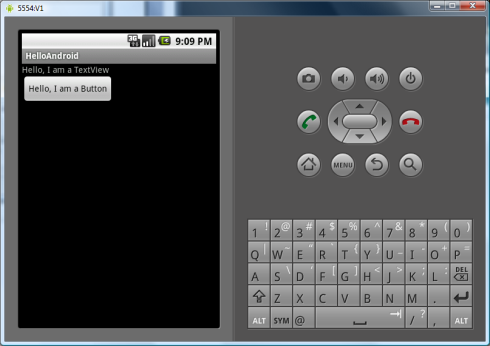
super.onCreate(savedInstanceState);

setContentView(R.layout.layout\_file\_name);

}

}

Faire **Run / Run**.



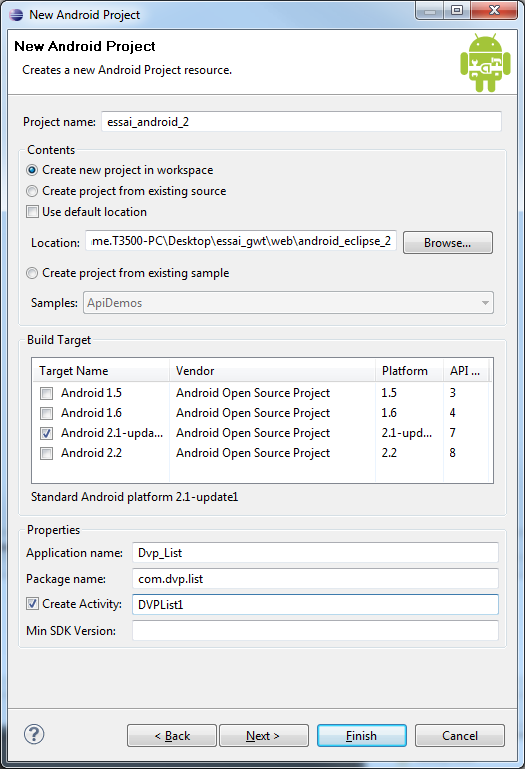
**4. Gestion des listes déroulantes**

Etape 1. Créer un nouveau projet nommé essai\_android\_2 avec les paramètres suivantes :

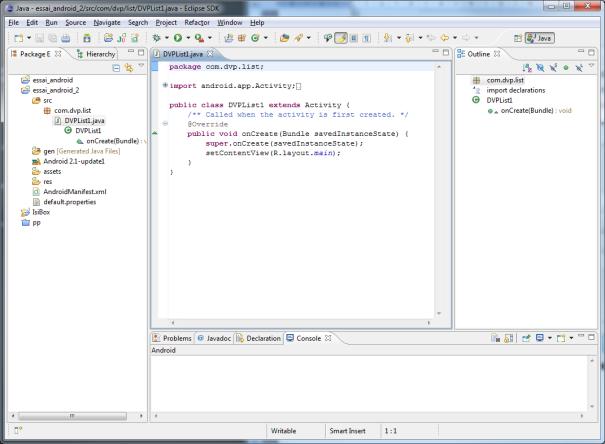
Application name : Dvp\_List

Package name : com.dvp.list

Create Activity : DVPList1



L’environnement Eclipse doit présenter le projet comme suit :



Etape 2. Modifier le type de **DVPList1**

Le code Java actuel du fichier DVPList1.java est le suivant :

**package** com.dvp.list;

**import** android.app.Activity;

**import** android.os.Bundle;

**public** class DVPList1 **extends** Activity {

/\*\* Called when the activity is first created. \*/

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.main);

}

}

Le code doit être modifié comme suit :

**package** com.dvp.list;

**import** android.app.Activity;

**import** android.app.ListActivity;

**import** android.os.Bundle;

**public** **class** DVPList1 **extends** ListActivity {

/\*\* Called when the activity is first created. \*/

@Override

**public** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

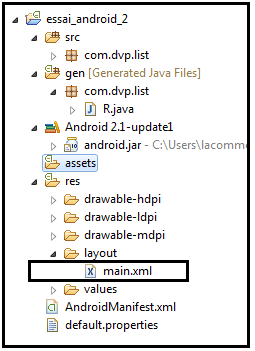
setContentView(R.layout.*main*);

}

}

Etape 3. Modifier le fichier **main.xml**

Le fichier main.xml se trouve dans res/layout.



Actuellement le fichier contient le code suivant :

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

<LinearLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*

android:orientation=*"vertical"*

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"fill\_parent"*

>

<TextView

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"wrap\_content"*

android:text=*"@string/hello"*

/>

</LinearLayout>

Il faut le modifier comme suite :

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

<LinearLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*

android:orientation=*"vertical"*

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"fill\_parent"*

>

<ListView android:id=*"@android:id/list"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*>

</ListView>

</LinearLayout>

Etape 4. Modifier le fichier **DVPList1.java**

Le code actuel est le suivant :

**package** com.dvp.list;

**import** android.app.Activity;

**import** android.app.ListActivity;

**import** android.os.Bundle;

**public** **class** DVPList1 **extends** ListActivity {

/\*\* Called when the activity is first created. \*/

@Override

**public** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*main*);

}

}

Le code doit être modifié comme suit :

**package** com.dvp.list;

**import** android.app.Activity;

**import** android.app.ListActivity;

**import** android.os.Bundle;

**import** android.widget.ArrayAdapter;

**public** **class** DVPList1 **extends** ListActivity {

/\*\* Called when the activity is first created. \*/

**private** String[] mStrings = {

"AAAAAAAA", "BBBBBBBB", "CCCCCCCC", "DDDDDDDD", "EEEEEEEE",

"FFFFFFFF", "GGGGGGGG", "HHHHHHHH", "IIIIIIII", "JJJJJJJJ",

"KKKKKKKK", "LLLLLLLL", "MMMMMMMM", "NNNNNNNN", "OOOOOOOO",

"PPPPPPPP", "QQQQQQQQ", "RRRRRRRR", "SSSSSSSS", "TTTTTTTT",

"UUUUUUUU", "VVVVVVVV", "WWWWWWWW", "XXXXXXXX", "YYYYYYYY",

"ZZZZZZZZ"

};

@Override

**public** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*main*);

ArrayAdapter<String> adapter = **new** ArrayAdapter<String>(**this**, android.R.layout.*simple\_list\_item\_1*, mStrings);

setListAdapter(adapter);

}

}

Etape 5. Tester le code.

Le résultat d’exécution donne :



**5. Gestion des listes déroulantes et des événements**

Par exemple, on peut considérer que chaque élément de la liste est un contact et se caractérise par :

* un nom
* un prénom
* un numéro (de téléphone).

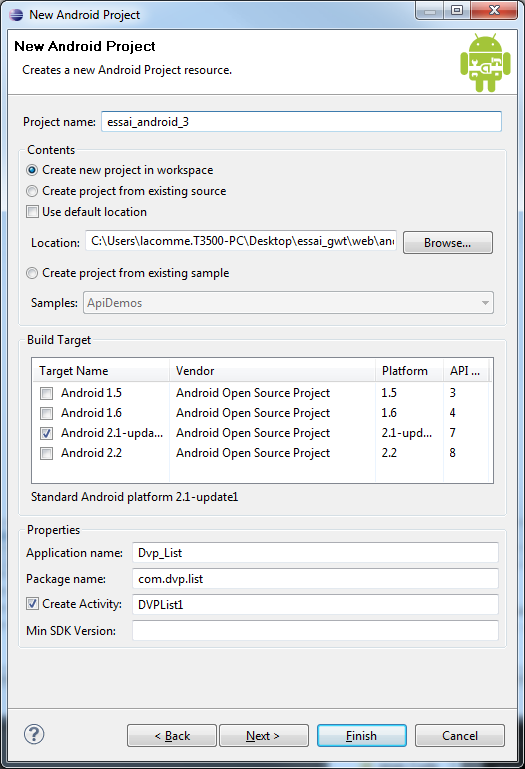
Nous allons reprendre un exemple similaire au précédent.

Etape 1. Créer un nouveau projet nommé **essai\_android\_3.**

Application name : Dvp\_List

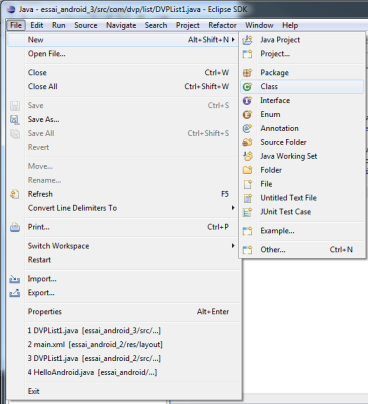
Package name : com.dvp.list

Create Activity : DVPList1

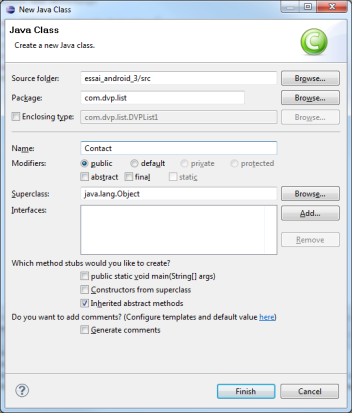


Etape 2. Créer une classe nommée **Contact.java**

Faire **New / Class**.



Choisir ensuite **Contact** comme nom de classe.



On peut définir de manière très simple la classe contact avec uniquement un constructeur et une méthode de classe permettant de créer une liste de contacts.

Tout ceci peut faire hurler les afficionados de l’approche objets mais cela reste simple et lisible.

**package** com.dvp.list;

**import** java.util.ArrayList;

**public** **class** Contact {

**public** String nom;

**public** String prenom;

**public** String telephone;

**public** Contact(String aNom, String aPrenom, String aTelephone) {

nom = aNom;

prenom = aPrenom;

telephone = aTelephone;

}

**public** **static** ArrayList<Contact> Initialiser ()

{

ArrayList<Contact> listContact = **new** ArrayList<Contact>();

Contact MonContact = **new** Contact("Dupont", "Thierry", "0124524521");

listContact.add(MonContact);

MonContact = **new** Contact("Tournesol", "Philippe", "054878569");

listContact.add(MonContact);

MonContact = **new** Contact("Martin", "Pecheur", "048578544");

listContact.add(MonContact);

MonContact = **new** Contact("Castafiore", "Helene", "08985785");

listContact.add(MonContact);

MonContact = **new** Contact("Dalton", "Joe", "0356898547");

listContact.add(MonContact);

MonContact = **new** Contact("Dalton", "Ma", "9874587444");

listContact.add(MonContact);

MonContact = **new** Contact("Obelix", "Gros", "025445836");

listContact.add(MonContact);

**return** listContact;

}

}

Etape 3. Modifier le fichier **main.xml**

Actuellement, le fichier main.xml ressemble à ceci :

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

<LinearLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*

android:orientation=*"vertical"*

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"fill\_parent"*

>

<TextView

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"wrap\_content"*

android:text=*"@string/hello"*

/>

</LinearLayout>

Il doit être modifié comme suit :

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

<LinearLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*

android:orientation=*"vertical"*

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"fill\_parent"*>

<ListView android:id=*"@+id/ListView01"*

android:layout\_width=*"wrap\_content"*

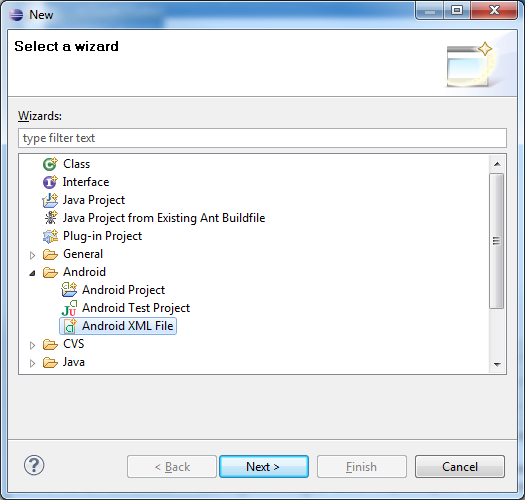
android:layout\_height=*"wrap\_content"*>

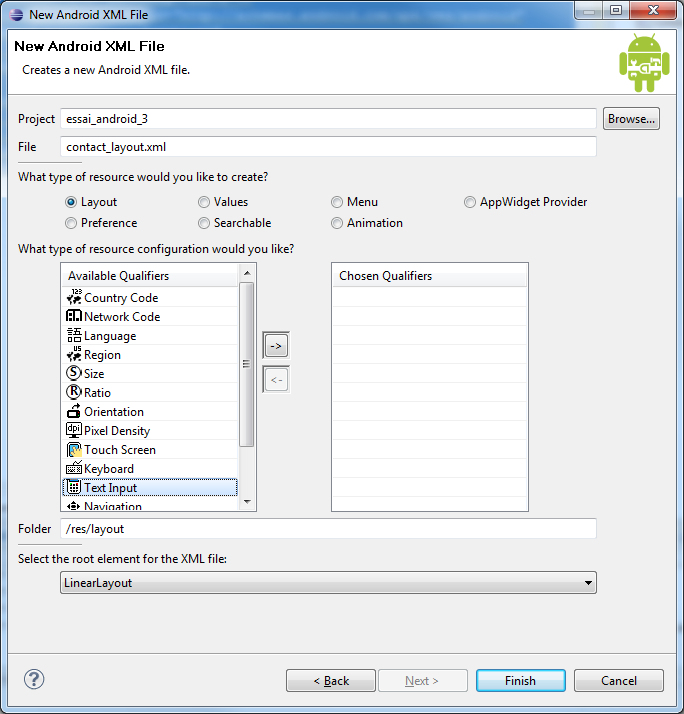
</ListView>

</LinearLayout>

Etape 4. Création d’un fichier **contact\_layout.xml**

Faire un clic droit sur **layout** et choisir **New / Others… / Android / Android XML**.





Ce fichier va définir la manière dont une instance de la classe Contact sera affichée.

Par défaut le fichier contient le code xml suivant :

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

<LinearLayout

xmlns:android=*"http://schemas.android.com/apk/res/android"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*>

</LinearLayout>

On peut par exemple définir ce fichier comme suit :

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

<LinearLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*

android:layout\_width=*"wrap\_content"* android:layout\_height=*"wrap\_content"*

android:id=*"@+id/LL\_Fond"*>

<TextView android:text=*"Nom"* android:id=*"@+id/TV\_Nom"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*>

</TextView>

<TextView android:text=*"Prénom"* android:id=*"@+id/TV\_Prenom"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*>

</TextView>

<ListView android:id=*"@+id/ListView01"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*>

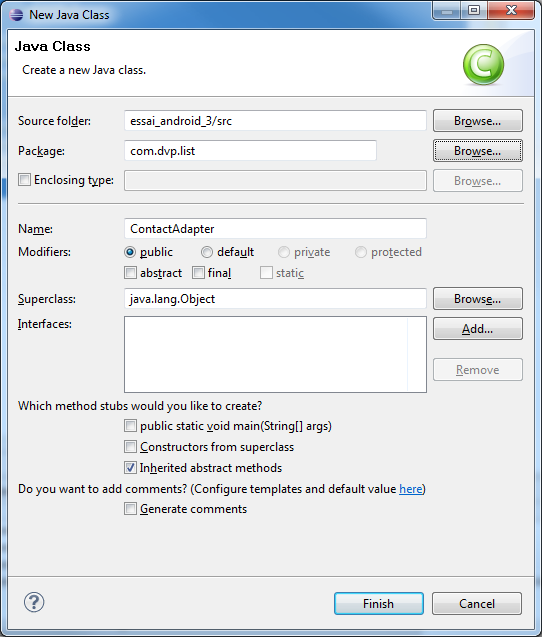
</ListView>

</LinearLayout>

Etape 5. Création d’une classe **ContactAdapter**

Nous allons créer un objet qui se chargera de gérer le mapping entre nos données et le layout des items. Ce composant sera basé sur un Adapter.

Faire **New / Class**



Le code généré par défaut est le suivant :

**package** com.dvp.list;

**public** **class** ContactAdapter {

}

Le code doit être modifié comme suit :

**package** com.dvp.list;

**import** java.util.List;

**import** android.content.Context;

**import** android.view.LayoutInflater;

**import** android.view.View;

**import** android.view.ViewGroup;

**import** android.widget.BaseAdapter;

**import** android.widget.LinearLayout;

**import** android.widget.TextView;

**public** **class** ContactAdapter **extends** BaseAdapter {

// Une liste de contact

**private** List<Contact> mListP;

//Le contexte dans lequel est présent notre adapter

**private** Context mContext;

//Un mécanisme pour gérer l'affichage graphique depuis un layout XML

**private** LayoutInflater mInflater;

// le constructeur

**public** ContactAdapter(Context context, List<Contact> aListP) {

mContext = context;

mListP = aListP;

//Le LayoutInflater permet de parser un layout XML et de te transcoder en IHM Android.

mInflater = LayoutInflater.*from*(mContext);

}

//Pour respecter l'interface BaseAdapter, il nous faut spécifier la méthode "count()".

**public** **int** getCount() {

**return** mListP.size();

}

**public** Object getItem(**int** position) {

**return** mListP.get(position);

}

**public** **long** getItemId(**int** position) {

**return** position;

}

// Maintenant il faut surcharger la méthode pour renvoyer une "View"

// en fonction d'une position donnée.

**public** View getView(**int** position, View convertView, ViewGroup parent) {

LinearLayout layoutItem;

//(1) : Réutilisation des layouts

**if** (convertView == **null**) {

//Initialisation de notre item à partir du layout XML "personne\_layout.xml"

layoutItem = (LinearLayout) mInflater.inflate(R.layout.*contact\_layout*, parent, **false**);

} **else** {

layoutItem = (LinearLayout) convertView;

}

//(2) : Récupération des TextView de notre layout

TextView tv\_Nom = (TextView)layoutItem.findViewById(R.id.*TV\_Nom*);

TextView tv\_Prenom = (TextView)layoutItem.findViewById(R.id.*TV\_Prenom*);

//(3) : Renseignement des valeurs

tv\_Nom.setText(mListP.get(position).nom);

tv\_Prenom.setText(mListP.get(position).prenom);

//On retourne l'item créé.

**return** layoutItem;

}

}

Etape 6. Modification du fichier DVPList1.java

Le code peut être

**package** com.dvp.list;

**import** java.util.ArrayList;

**import** android.app.Activity;

**import** android.os.Bundle;

**import** android.widget.ListView;

**import** com.dvp.list.ContactAdapter;

**public** **class** DVPList1 **extends** Activity {

/\*\* Called when the activity is first created. \*/

@Override

**public** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*main*);

//Récupération de la liste des personnes

ArrayList<Contact> listP = Contact.*Initialiser*();

//Création et initialisation de l'Adapter pour les contact

ContactAdapter adapter = **new** ContactAdapter(**this**, listP);

//Récupération du composant ListView

ListView list = (ListView)findViewById(R.id.ListView01);

//Initialisation de la liste avec les données

list.setAdapter(adapter);

}

}

|  |  |
| --- | --- |
| attenzione_architetto_fr_01 | La ligne :  ListView list = (ListView)findViewById(R.id.ListView01);  Est responsible d’une erreur.  Faire un clic gauche dans la marge et choisir **Create Field** |

Le fichier R.java ressemble alors à ce qui suit :

/\* AUTO-GENERATED FILE. DO NOT MODIFY.

\*

\* This class was automatically generated by the

\* aapt tool from the resource data it found. It

Dernière mise à jour du fichier

\* should not be modified by hand.

\*/

**package** com.dvp.list;

**public** **final** **class** R {

**public** **static** **final** **class** attr {

}

**public** **static** **final** **class** drawable {

**public** **static** **final** **int** *icon*=0x7f020000;

}

**public** **static** **final** **class** id {

**public** **static** **final** **int** *LL\_Fond*=0x7f050000;

**public** **static** **final** **int** *ListView01*=0x7f050003;

**public** **static** **final** **int** *TV\_Nom*=0x7f050001;

**public** **static** **final** **int** *TV\_Prenom*=0x7f050002;

}

**public** **static** **final** **class** layout {

**public** **static** **final** **int** *contact\_layout*=0x7f030000;

**public** **static** **final** **int** *main*=0x7f030001;

}

**public** **static** **final** **class** string {

**public** **static** **final** **int** *app\_name*=0x7f040001;

**public** **static** **final** **int** *hello*=0x7f040000;

}

}

Etape 7. Tester le programme.

L’exécution doit donner ceci :



Etape 8. Gérer les événéments de la liste

Sous-Etape 8.1. Ajouter une interface à la classe ContactAdapter et des listeners

Ouvrir le fichier ContactAdapter.java et remplacer par le code suivant :

**package** com.dvp.list;

**import** java.util.ArrayList;

**import** java.util.List;

**import** android.content.Context;

**import** android.view.LayoutInflater;

**import** android.view.View;

**import** android.view.ViewGroup;

**import** android.widget.BaseAdapter;

**import** android.widget.LinearLayout;

**import** android.widget.TextView;

**public** **class** ContactAdapter **extends** BaseAdapter {

// Une liste de contact

**private** List<Contact> mListP;

//Le contexte dans lequel est présent notre adapter

**private** Context mContext;

//Un mécanisme pour gérer l'affichage graphique depuis un layout XML

**private** LayoutInflater mInflater;

// le constructeur

**public** ContactAdapter(Context context, List<Contact> aListP) {

mContext = context;

mListP = aListP;

//Le LayoutInflater permet de parser un layout XML et de te transcoder en IHM Android.

mInflater = LayoutInflater.*from*(mContext);

}

//Pour respecter l'interface BaseAdapter, il nous faut spécifier la méthode "count()".

**public** **int** getCount() {

**return** mListP.size();

}

**public** Object getItem(**int** position) {

**return** mListP.get(position);

}

**public** **long** getItemId(**int** position) {

**return** position;

}

// Maintenant il faut surcharger la méthode pour renvoyer une "View"

// en fonction d'une position donnée.

**public** View getView(**int** position, View convertView, ViewGroup parent) {

LinearLayout layoutItem;

//(1) : Réutilisation des layouts

**if** (convertView == **null**) {

//Initialisation de notre item à partir du layout XML "personne\_layout.xml"

layoutItem = (LinearLayout) mInflater.inflate(R.layout.*contact\_layout*, parent, **false**);

} **else** {

layoutItem = (LinearLayout) convertView;

}

//(2) : Récupération des TextView de notre layout

TextView tv\_Nom = (TextView)layoutItem.findViewById(R.id.*TV\_Nom*);

TextView tv\_Prenom = (TextView)layoutItem.findViewById(R.id.*TV\_Prenom*);

//(3) : Renseignement des valeurs

tv\_Nom.setText(mListP.get(position).nom);

tv\_Prenom.setText(mListP.get(position).prenom);

//On retourne l'item créé.

**return** layoutItem;

}

// Interface pour écouter les évènements sur le nom d'un contact

**public** **interface** ContactAdapterListener {

**public** **void** onClickNom(Contact item, **int** position);

}

//Contient la liste des listeners

**private** ArrayList<ContactAdapterListener> mListListener = **new** ArrayList<ContactAdapterListener>();

// Pour ajouter un listener sur notre adapter

**public** **void** addListener(ContactAdapterListener aListener) {

mListListener.add(aListener);

}

// permet de prévenir tous les listeners

**private** **void** sendListener(Contact item, **int** position) {

**for**(**int** i = mListListener.size()-1; i >= 0; i--) {

mListListener.get(i).onClickNom(item, position);

}

}

}

Sous-Etape 8.2. Modification de la méthode **getView**.

La version actuelle de la méthode est la suivante :

**public** View getView(**int** position, View convertView, ViewGroup parent) {

LinearLayout layoutItem;

//(1) : Réutilisation des layouts

**if** (convertView == **null**) {

//Initialisation de notre item à partir du layout XML "personne\_layout.xml"

layoutItem = (LinearLayout) mInflater.inflate(R.layout.*contact\_layout*, parent, **false**);

} **else** {

layoutItem = (LinearLayout) convertView;

}

//(2) : Récupération des TextView de notre layout

TextView tv\_Nom = (TextView)layoutItem.findViewById(R.id.*TV\_Nom*);

TextView tv\_Prenom = (TextView)layoutItem.findViewById(R.id.*TV\_Prenom*);

//(3) : Renseignement des valeurs

tv\_Nom.setText(mListP.get(position).nom);

tv\_Prenom.setText(mListP.get(position).prenom);

//On retourne l'item créé.

**return** layoutItem;

}

Ajouter la définition d’une variable **Click Listener** comme suit (et importer le package nécessaire : android.view.View.OnClickListener) :

OnClickListener ClickListener = **new** OnClickListener() {

@Override

**public** **void** onClick(View v) {

//Lorsque l'on clique sur le nom, on récupère la position de la "Personne"

Integer position = (Integer)v.getTag();

//On prévient les listeners qu'il y a eu un clic sur le TextView "TV\_Nom".

sendListener(mListP.get(position), position);

}

};

Et copier le code suivant dans **getView**.

**public** View getView(**int** position, View convertView, ViewGroup parent) {

LinearLayout layoutItem;

//(1) : Réutilisation des layouts

**if** (convertView == **null**) {

//Initialisation de notre item à partir du layout XML "contact\_layout.xml"

layoutItem = (LinearLayout) mInflater.inflate(R.layout.*contact\_layout*, parent, **false**);

} **else** {

layoutItem = (LinearLayout) convertView;

}

//(2) : Récupération des TextView de notre layout

TextView tv\_Nom = (TextView)layoutItem.findViewById(R.id.*TV\_Nom*);

TextView tv\_Prenom = (TextView)layoutItem.findViewById(R.id.*TV\_Prenom*);

//(3) : Renseignement des valeurs

tv\_Nom.setText(mListP.get(position).nom);

tv\_Prenom.setText(mListP.get(position).prenom);

// (4) : On mémorise la position de la "Contact" dans le composant textview

tv\_Nom.setTag(position);

//On ajoute un listener

modifications

tv\_Nom.setOnClickListener(ClickListener);

//On retourne l'item créé.

**return** layoutItem;

}

Sous-Etape 9.3. Modification de la classe DvpList1

Ouvrir le fichier **DvpList1.java** et modifier le code en ajoutant :

… **implements** ContactAdapterListener …

Ceci nécessite de modifier les « **imports** »…

Au final la nouvelle classe DVPList1 doit ressembler à ceci :

**package** com.dvp.list;

**import** java.util.ArrayList;

**import** android.app.Activity;

**import** android.app.AlertDialog;

**import** android.app.AlertDialog.Builder;

**import** android.os.Bundle;

**import** android.widget.ListView;

**import** com.dvp.list.ContactAdapter;

**import** com.dvp.list.ContactAdapter.ContactAdapterListener;

**public** **class** DVPList1 **extends** Activity **implements** ContactAdapterListener {

/\*\* Called when the activity is first created. \*/

**public** **void** onClickNom(Contact item, **int** position) {

ajout

Builder builder = **new** AlertDialog.Builder(**this**);

builder.setTitle("Contact Information");

builder.setMessage("Le telephone est : " + item.telephone);

builder.setPositiveButton("OK", **null**);

builder.show();

}

@Override

**public** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*main*);

//Récupération de la liste des personnes

ArrayList<Contact> listP = Contact.*Initialiser*();

//Création et initialisation de l'Adapter pour les contact

ContactAdapter adapter = **new** ContactAdapter(**this**, listP);

modifications

adapter.addListener(**this**);

//Récupération du composant ListView

ListView list = (ListView)findViewById(R.id.*ListView01*);

//Initialisation de la liste avec les données

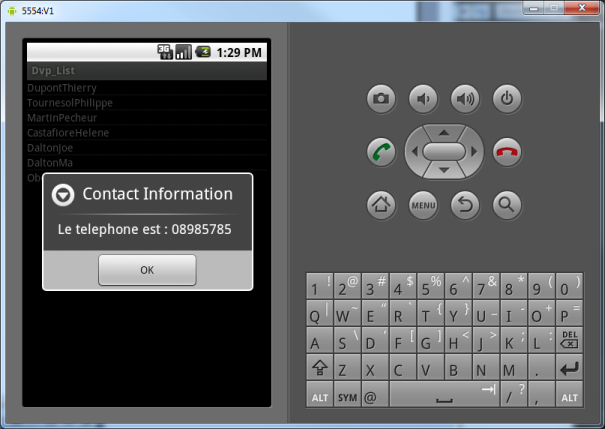
list.setAdapter(adapter);

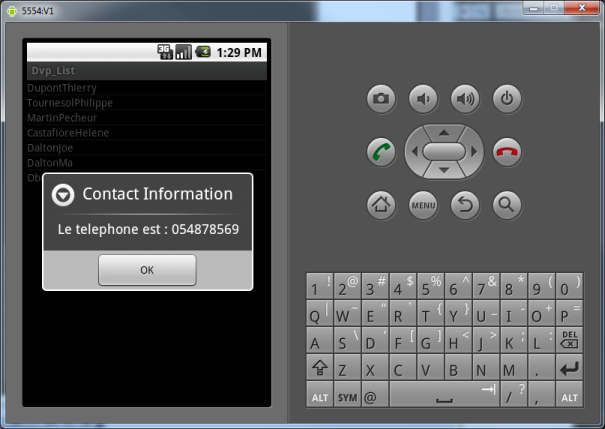
}

}

Sous-Etape 8.4. Tester le programme

Cliquer par exemple sur lenom « Dupont ».

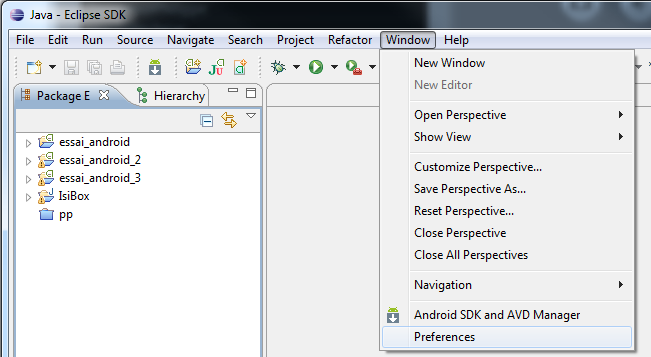




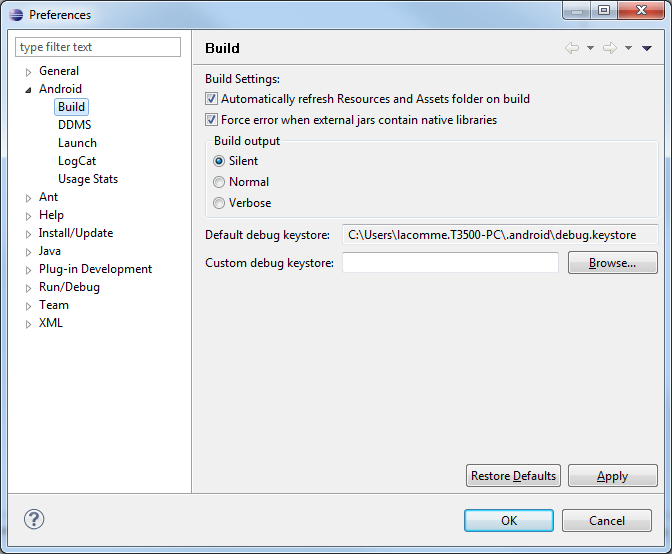
**6. Intégrer Google Map dans Android**

Etape 1. Obtenir le md5 checksum.

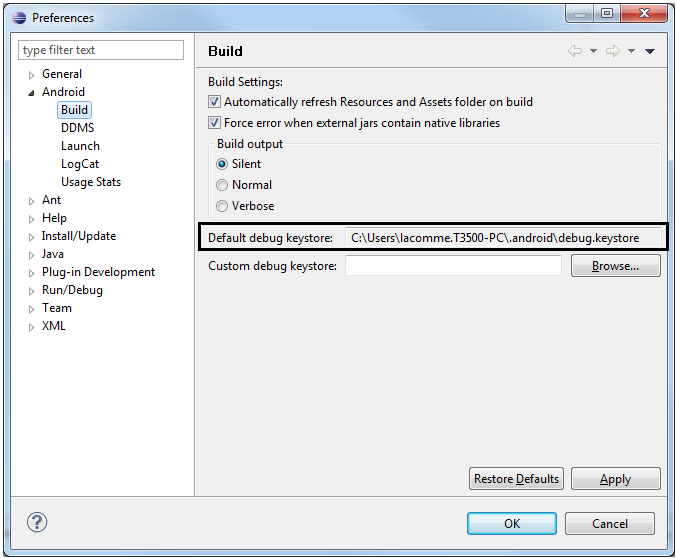
Aller dans **Window / Preferences**.



Choisir ensuite **Android / Build**.

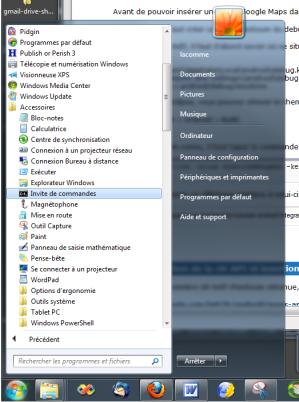


Relever le chemin d’accès par défaut à **Default debug keystore**.

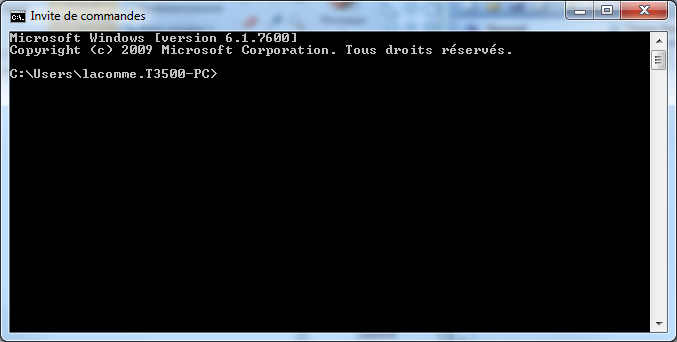


Dans mon cas particulier le chemin est : **C:\Users\lacomme.T3500-PC\.android\**

Ouvrir une console : **menu Demarrer / Accessoires / Invite de Commandes**.



Ce qui donne :



Dans le fenêtre Invite de commandes, il faut taper la commande suivante :

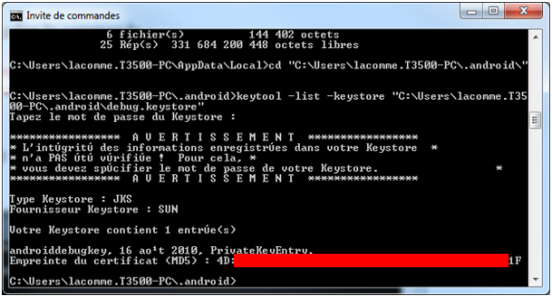
**keytool -list -keystore <chemin\_vers\_le\_fichier\_debug>.keystore**

La commande

**kytool -list – keystore**

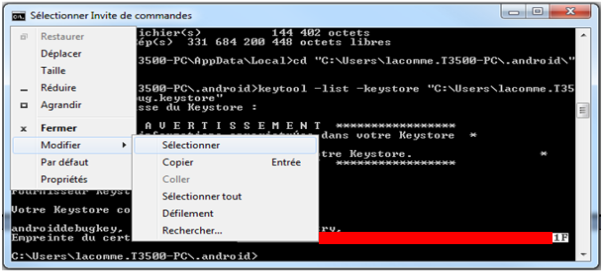
**« C:\Users\lacomme.T3500-PC\.android\debug.keystore »**

donne sur ma machine :

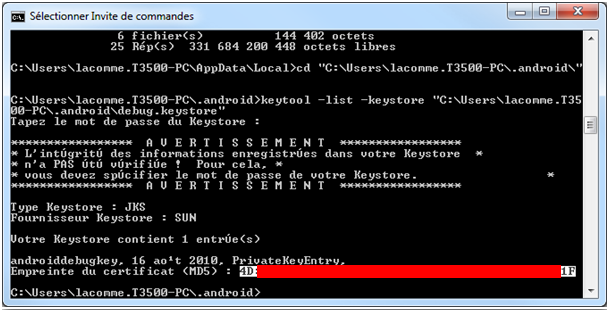


|  |  |
| --- | --- |
| attenzione_architetto_fr_01 | Il est possible que vous obteniez l’erreur suivante :  **Microsoft Windows [version 6.0.6001]**  **Copyright (c) 2006 Microsoft Corporation. Tous droits réservés.**  **C:\Users\Philippe>keytool -list -keystore …………….**  **'keytool' n'est pas reconnu en tant que commande interne**  **ou externe, un programme exécutable ou un fichier de commandes.**  Cela veut dire que le chemin d’accès dans votre path ne contient pas le chemin correct vers l’exécutable.  Si vous avez réaliser une installation standard des différents éléments (SDK etc…), il est probable que le JDK soit dans le répertoire Sun sous la racine.  Dans la fenêtre MSDOS, faire :  **cd c:\sun\SDK\jdk\bin**  **keytool -list – keystore …..**  et tout devrait fonctionner. |

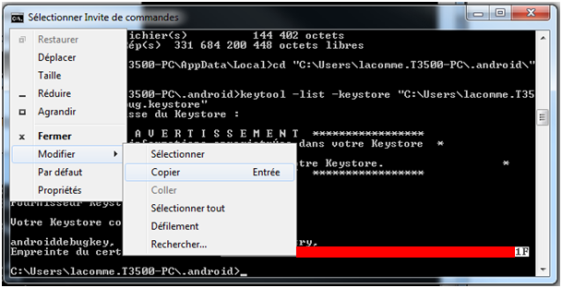
Faire ensuite : **Modifier / Sélectionner**.



Avec la souris, sélectionner votre MD5 :



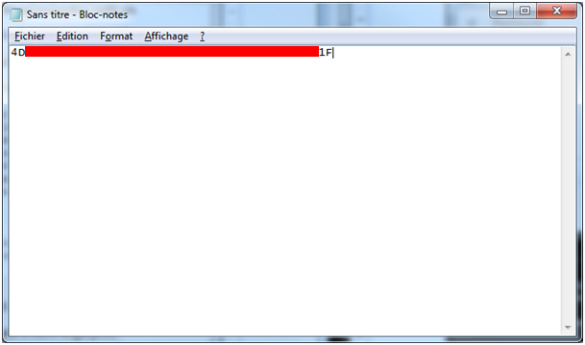
Faire ensuite **Modifier / Copier**.



Vérifier que la copie de votre MD5 s’est effectuée correctement.

Pour cela ouvrez le bloc notes et faites **Edition / Coller.**

Vous devriez obtenir un document comme celui-ci :



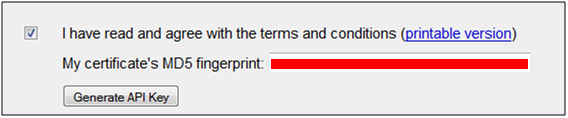
Etape 2. Obtenir une clé Google.

Connectez vous à l’adresse suivante :

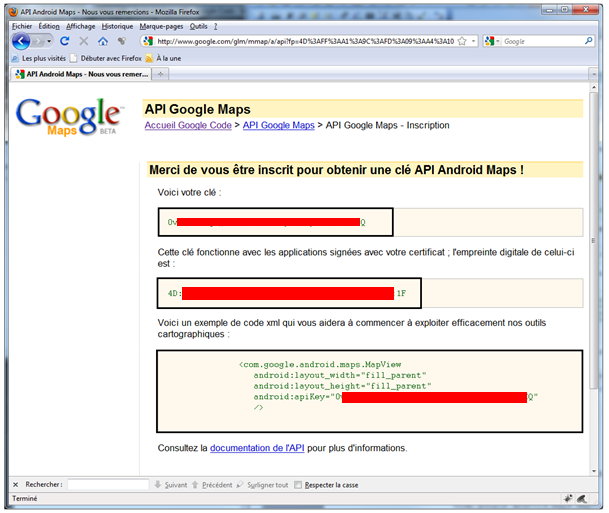
<http://code.google.com/intl/fr/android/maps-api-signup.html>



Faites CTRL-V dans le champ My certificate’s MD5 fingerprint :



Après quelques secondes, vous devriez obtenir une page web vous donnant votre clé :



Dans mon cas particulier, j’ai obtenu :

Clé : 0\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Q

MD5 : 4D\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*1F

Et un exemple de code :

<com.google.android.maps.MapView

android:layout\_width="fill\_parent"

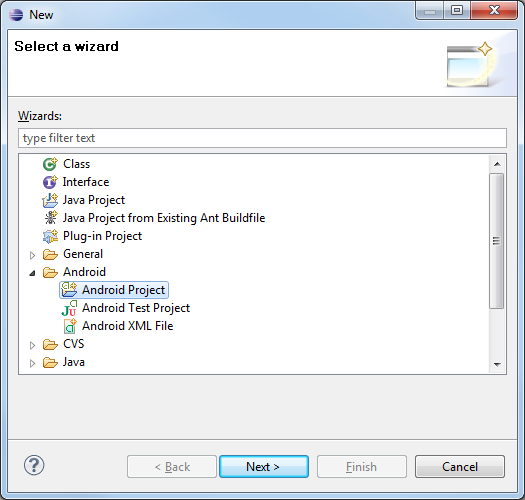
android:layout\_height="fill\_parent"

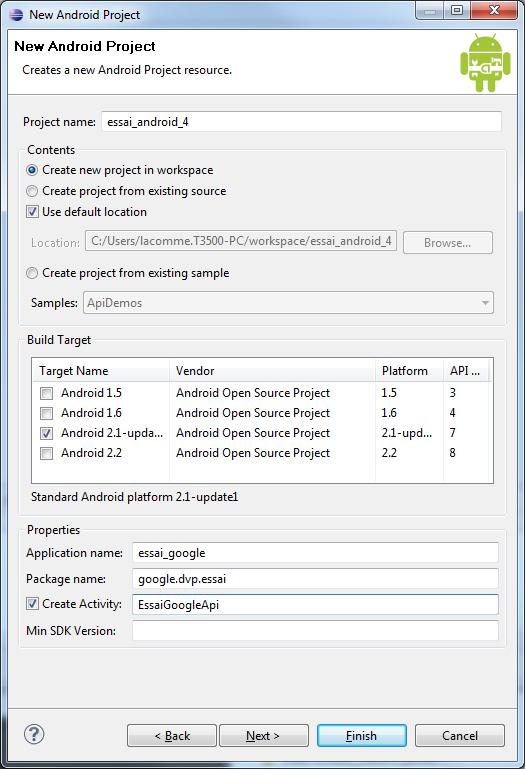
android:apiKey="0\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Q"

/>

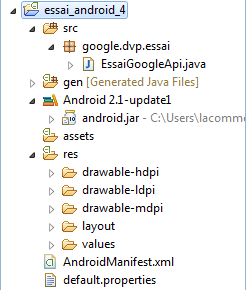
Etape 3. Créer une application Androïd.

Faire New / Project et choisir Android Project.



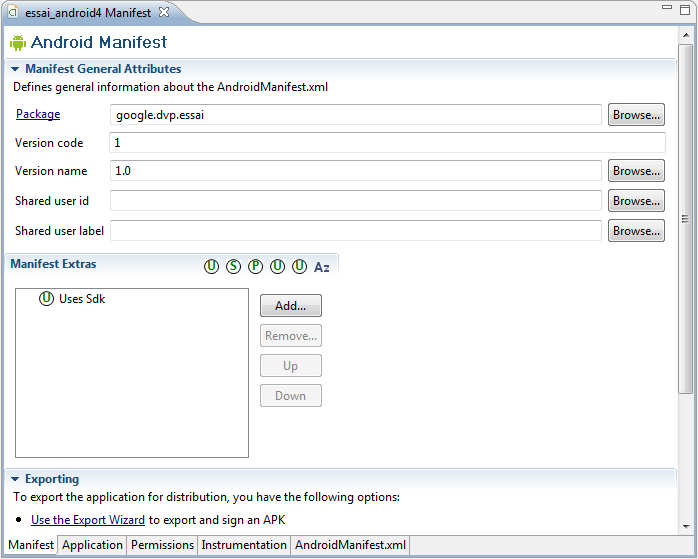


Le projet ainsi crée doit se présenter comme suit :

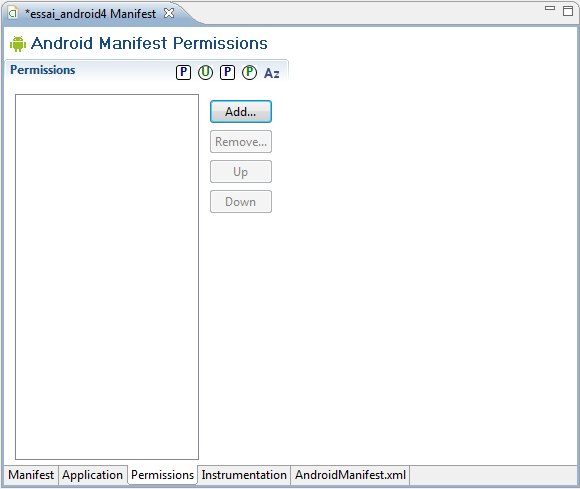


Etape 4. Modifier l’application Androïd pour utiliser les services google

Ouvrir le fichier **AndroidManifest.xml**

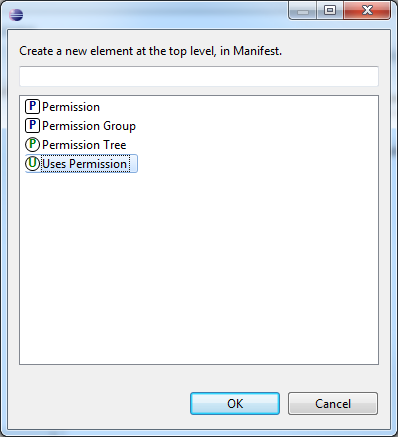


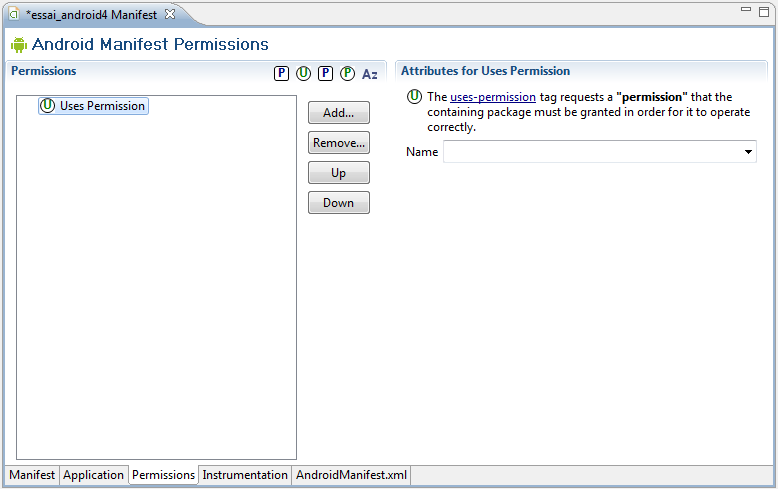
Aller dans l’onglet **Permissions**.



Faire ensuite **Add**.

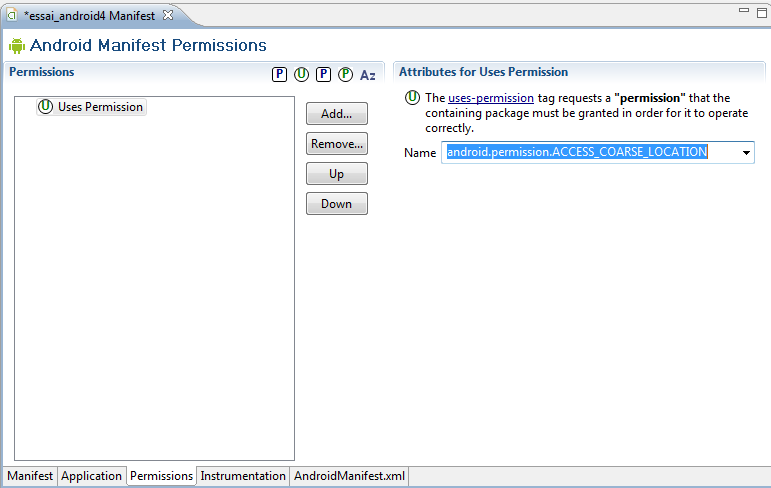
Choisir ensuite **Uses Permission**.





Dans le champ Name choisir :

**android.permission.ACCESS\_COARSE\_LOCATION**

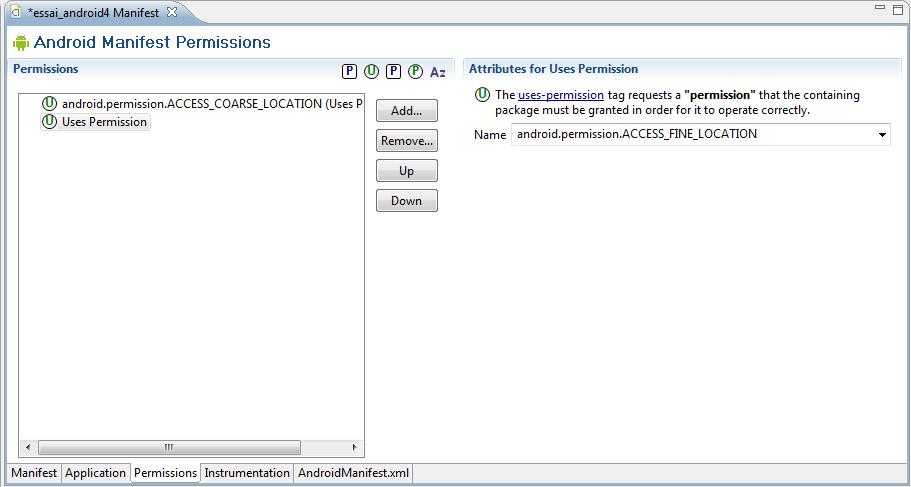


Passez ensuite sur l’onglet AndroidManifest.xml et vérifier que vous obtenez un fichier similaire à celui-ci :



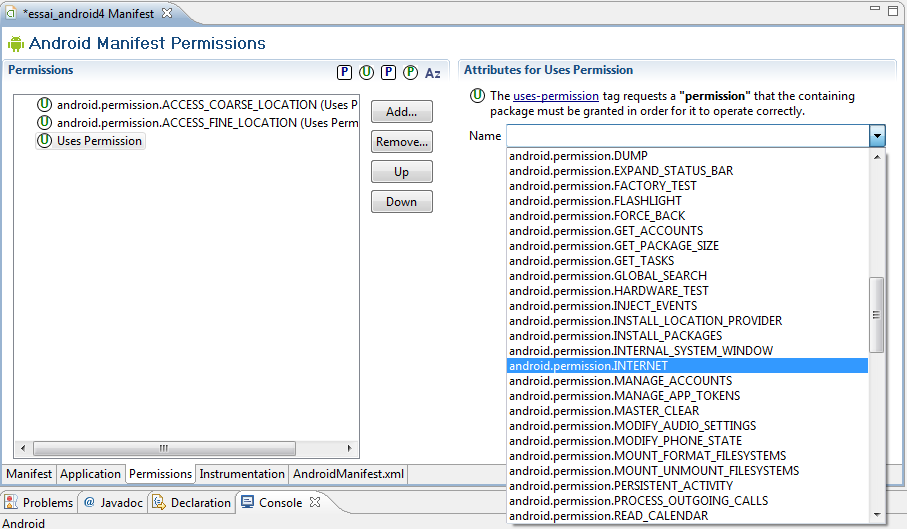
Recommencer l’opération en choisissant :

**android.permission.ACCESS\_FINE\_LOCATION**



Et finalement une troisième fois en choisissant :

**android.permission.INTERNET**



Le fichier AndroidManifest.xml doit ressembler à ceci :

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

<manifest xmlns:android=*"http://schemas.android.com/apk/res/android"*

package=*"google.dvp.essai"*

android:versionCode=*"1"*

android:versionName=*"1.0"*>

<application android:icon=*"@drawable/icon"* android:label=*"@string/app\_name"*>

<activity android:name=*".EssaiGoogleApi"*

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>

<uses-permission android:name=*"android.permission.ACCESS\_COARSE\_LOCATION"*></uses-permission>

<uses-permission android:name=*"android.permission.ACCESS\_FINE\_LOCATION"*></uses-permission>

<uses-permission android:name=*"android.permission.INTERNET"*></uses-permission>

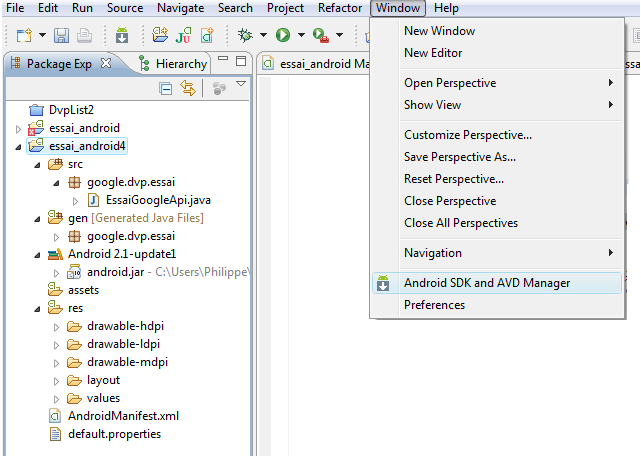
</manifest>

Prenez le temps de vérifier que le programme compile et s’exécute. Ceci devrait donner :

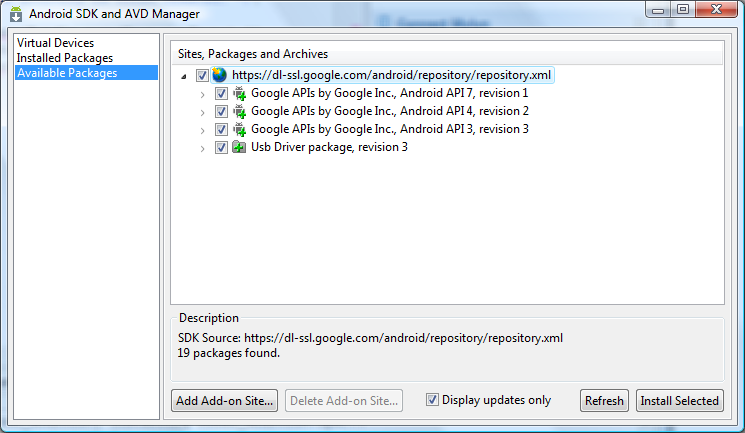


Etape 5. Ajouter l’API google

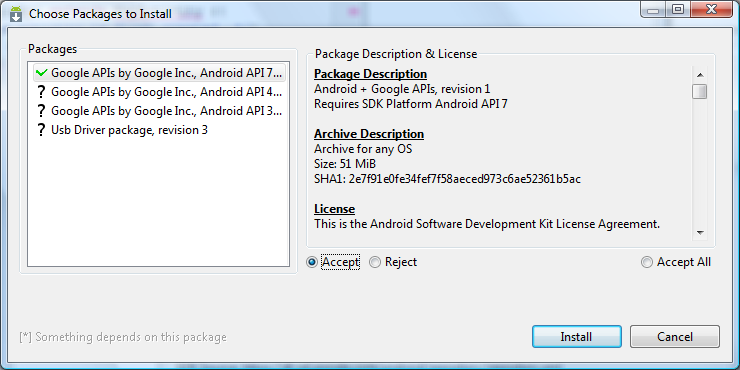
Aller dans **Window / Android SDK**



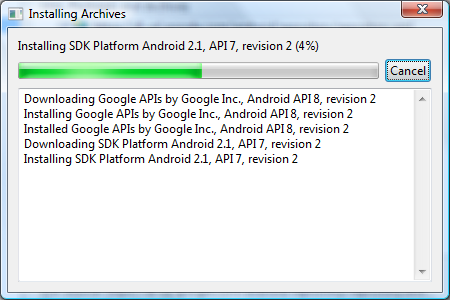
Aller dans **Available Packages**.



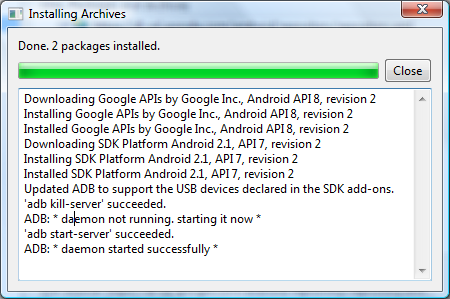
Choisir **Install Selected.**



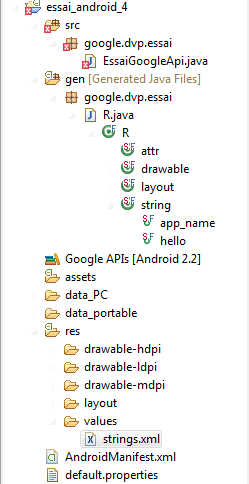
Accepter les conditions de la licence et faire **Install**.



Patientez jusqu’à la fin du processus.



Etape 6. Ouvrir lefichier nommé **strings.xml**



Le fichier initialement se présente comme suit.

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

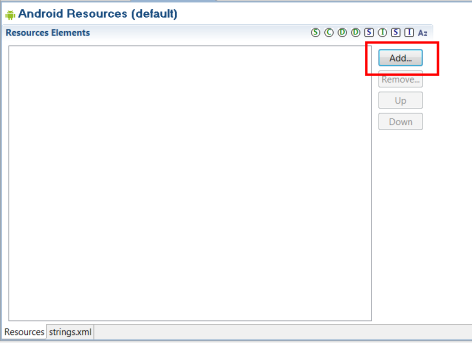
<resources>

<string name=*"hello"*>Hello World, EssaiGoogleApi!</string>

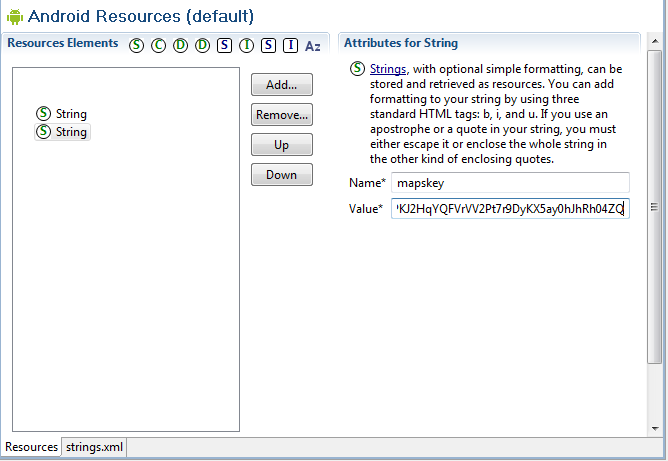
<string name=*"app\_name"*>essai\_google</string>

</resources>

Aller dans **Ressoures** et choisir **add->String**.







Le fichier doit se présenter comme suit :

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

<resources>

<string name=*"hello"*>Hello World, EssaiGoogleApi!</string>

<string name=*"app\_name"*>essai\_google</string>

<string name=*"mapskey"*>0vfPKJ2HqYQFVrVV2Pt7r9DyKX5ay0hJhRh04ZQ</string>

</resources>

Etape 7. Modifier le fichier main.xml

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

<LinearLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*

android:orientation=*"vertical"*

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"fill\_parent"*

>

Correspond à l’exemple de code proposé par Google

<com.google.android.maps.MapView

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"fill\_parent"*

android:apiKey=*"0vfPKJ2HqYQFVrVV2Pt7r9DyKX5ay0hJhRh04ZQ"*

/>

</LinearLayout>

Etape 8. Modifier le fichier AndroidManifest.xml

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

<manifest xmlns:android=*"http://schemas.android.com/apk/res/android"*

package=*"google.dvp.essai"*

android:versionCode=*"1"*

android:versionName=*"1.0"*>

<application android:icon=*"@drawable/icon"* android:label=*"@string/app\_name"* android:debuggable=*"true"*>

Modification

<uses-library android:name=*"com.google.android.maps"* />

<activity android:name=*".EssaiGoogleApi"*

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>

<uses-permission android:name=*"android.permission.ACCESS\_COARSE\_LOCATION"*></uses-permission>

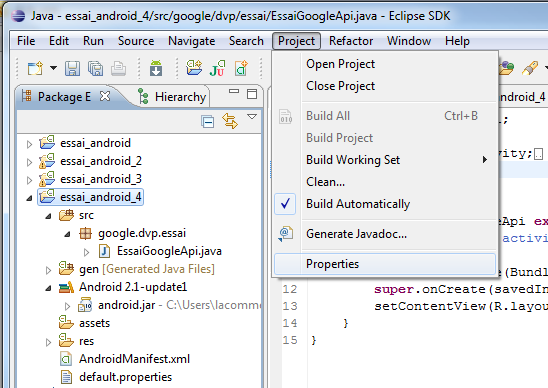
<uses-permission android:name=*"android.permission.ACCESS\_FINE\_LOCATION"*></uses-permission>

<uses-permission android:name=*"android.permission.INTERNET"*></uses-permission>

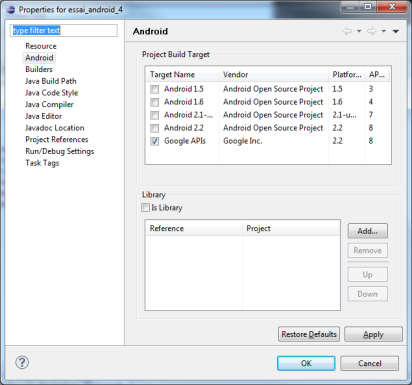
</manifest>

Etape 9. Modifier l’application principale

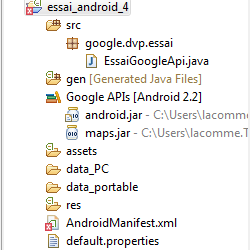
Aller dans **Project / Properties**



Aller dans la partie Androïd et modifiez la « target ».



Ouvrez le fichier **EssaiGoogleApi.java.**



**package** google.dvp.essai;

**import** android.app.Activity;

**import** android.os.Bundle;

**import** android.\*;

**import** com.google.android.maps.GeoPoint;

**import** com.google.android.maps.MapActivity;

**import** com.google.android.maps.MapController;

**import** com.google.android.maps.MapView;

**public** **class** EssaiGoogleApi **extends** MapActivity {

/\*\* Called when the activity is first created. \*/

@Override

**public** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*main*);

}

@Override

**protected** **boolean** isRouteDisplayed() {

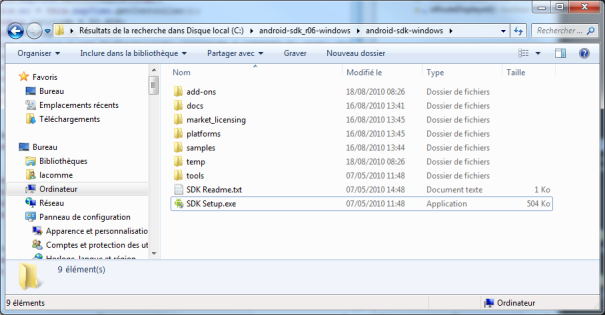
**return** **false**;

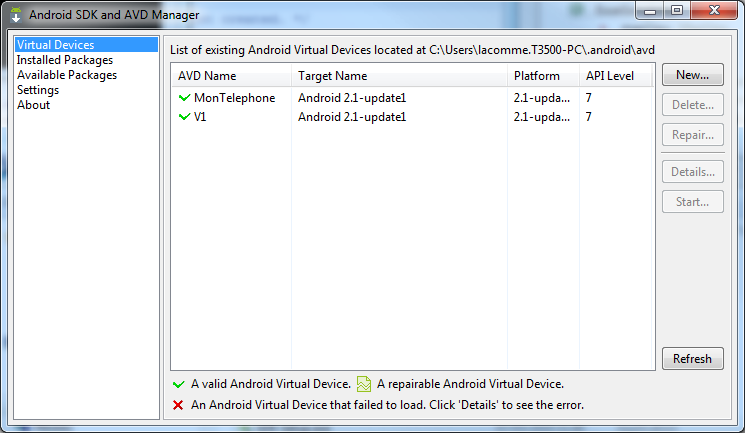
}

}

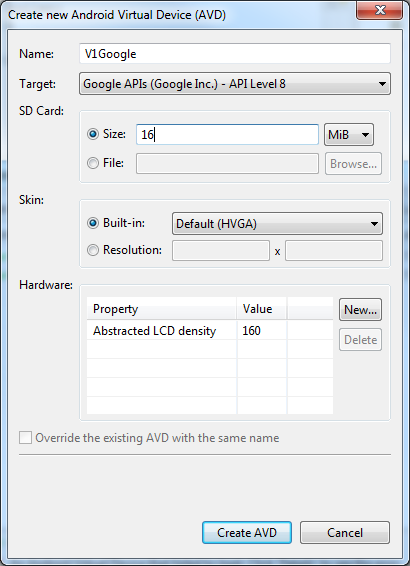
Etape 10. Créer un AVD compatible

Aller dans le répertoire d’installation du SDK et lancer l’exécutable.

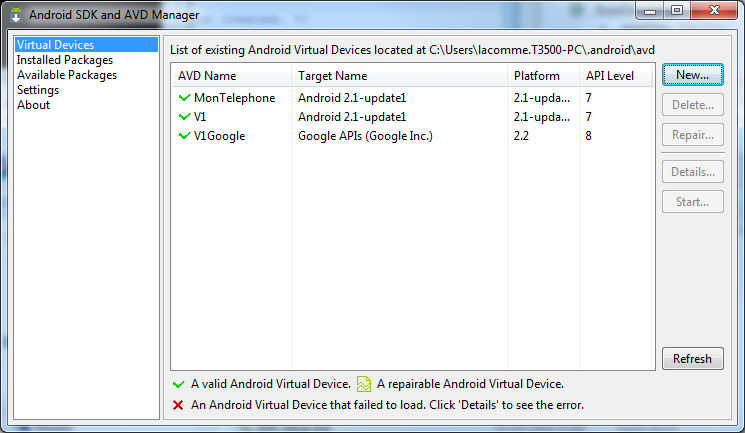




Faire ensuite **New**.

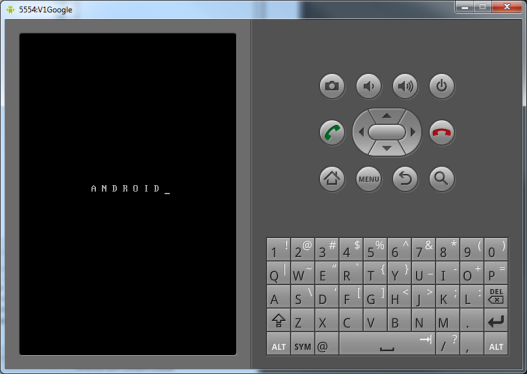


Ce qui donne à la fin :



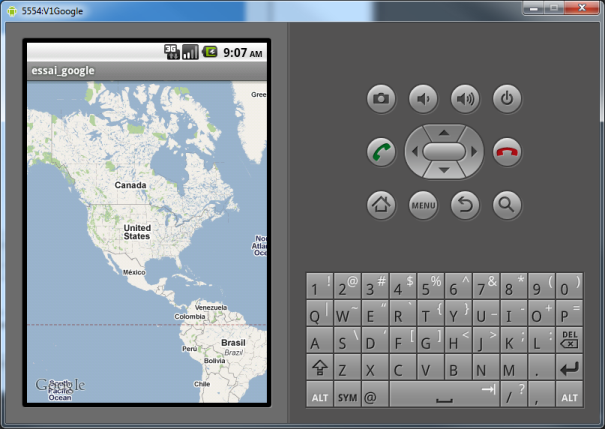
Etape 8. Exécuter l’application.

Comme notre machine V1Google est la seule compatible elle est immédiatement sélectionnée.





Ce qui donne :



Etape 11. Géolocaliser un point

Ouvrir le fichier EssaiGoogleApi.java et modifier le code comme suit :

**package** google.dvp.essai;

**import** android.app.Activity;

**import** android.os.Bundle;

**import** android.\*;

**import** com.google.android.maps.GeoPoint;

**import** com.google.android.maps.MapActivity;

**import** com.google.android.maps.MapController;

**import** com.google.android.maps.MapView;

**public** **class** EssaiGoogleApi **extends** MapActivity {

**private** MapView mapView;

**private** MapController mc;

**private** GeoPoint location;

/\*\* Called when the activity is first created. \*/

@Override

**public** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

**this**.mapView = **new** MapView(**this**,**this**.getResources().getString(R.string.*mapskey*));

**this**.mapView.setClickable(**true**);

**this**.mc = **this**.mapView.getController();

// on positionne la carte sur l'isima

**double** latitude = 45.758891;

**double** longitude = 3.111223;

**this**.location = **new** GeoPoint((**int**)(latitude\*1000000.0),(**int**)(longitude\*1000000.0));

**this**.mc.setCenter(**this**.location);

**this**.mc.setZoom(25);

**this**.mapView.setSatellite(**true**);

**this**.mapView.invalidate();

setContentView(**this**.mapView);

}

@Override

**protected** **boolean** isRouteDisplayed() {

**return** **false**;

}

}

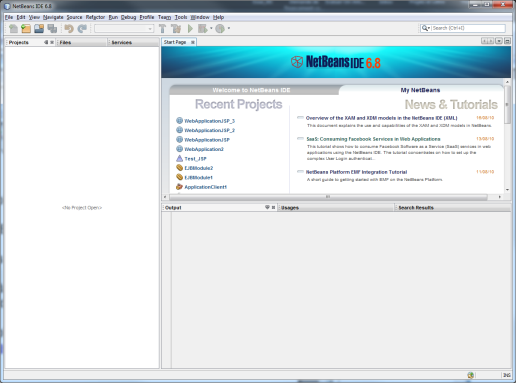
A l’exécution, on obtient une carte centrée sur l’ISIMA.



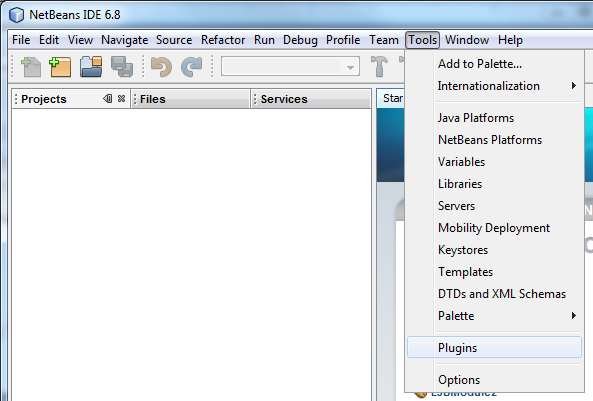
# outils : NETBEANS and CO

**1. Configurer NetBeans**

Lancer NetBeans.

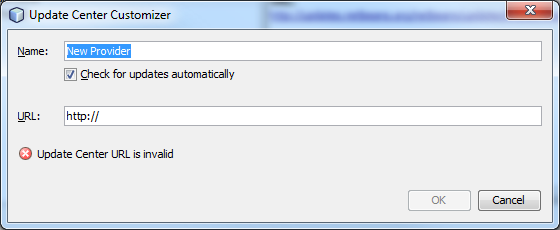


Aller dans le menu **Tools/Plugins**

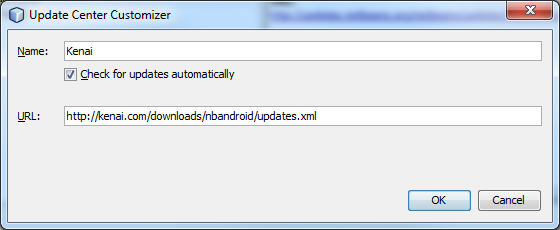


Aller dans **Settings** et faire ensuite **Add**.

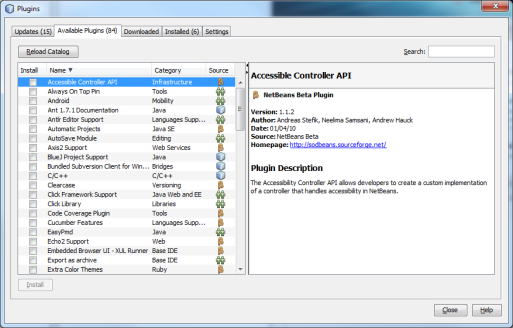




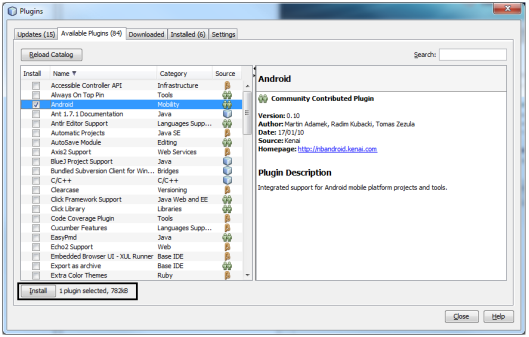
Dans le champ Name, on peut choisir Kenai par exemple et dans le champ URL l’adresse suivante : <http://kenai.com/downloads/nbandroid/updates.xml>



Consulter ensuite la partie **Available Plugins**.



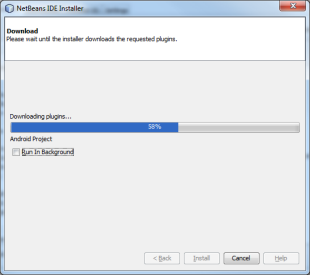
Choisir ensuite Androïd et lancez l’installation (bouton **Install**).



Validez ensuite les différentes étapes de l’installation.

|  |  |
| --- | --- |
|  |  |

Patientez quelques secondes pendant l’installation…

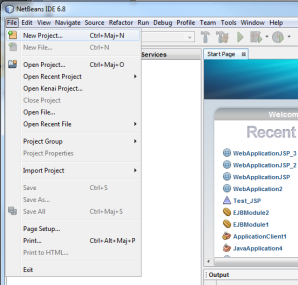


Certains plugins ne sont pas signés. Une autorisation est nécessaire pour réaliser l’installation.

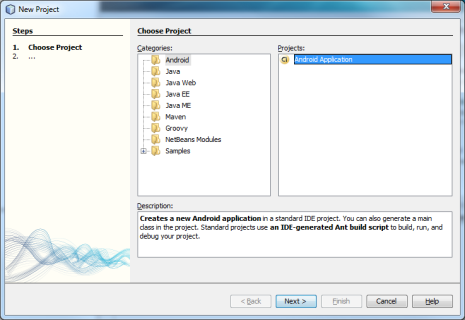
|  |  |
| --- | --- |
|  |  |

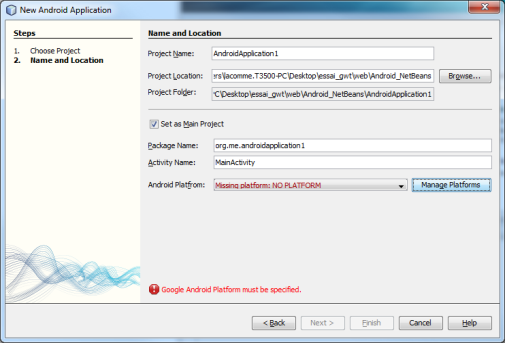
**2. Créer un projet Android**

Faire **File / New Project**

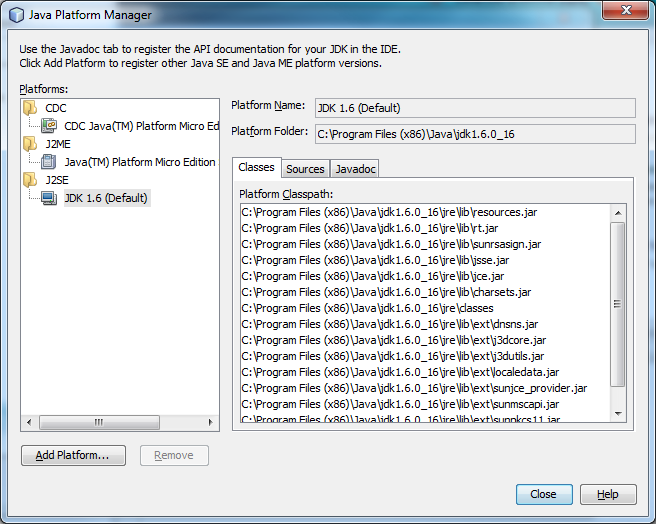


Choisir **Android** et **Android Application**.

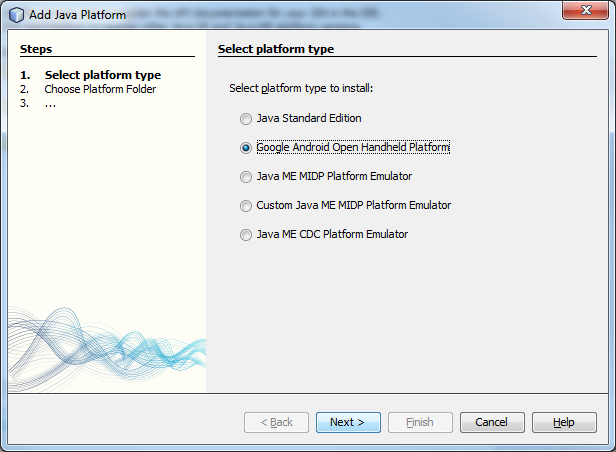




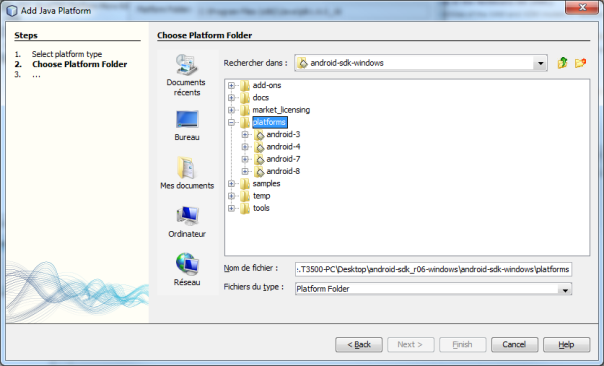
Choisir ensuite **Manage Plateforms**.

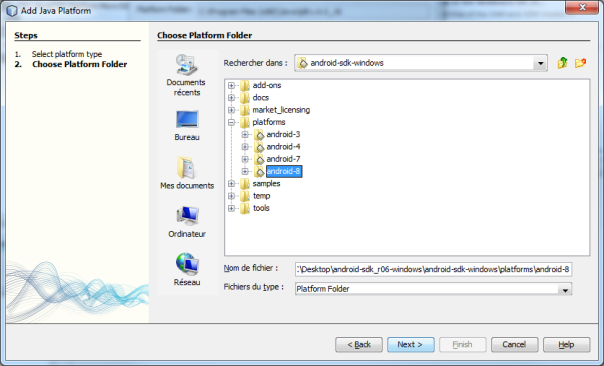


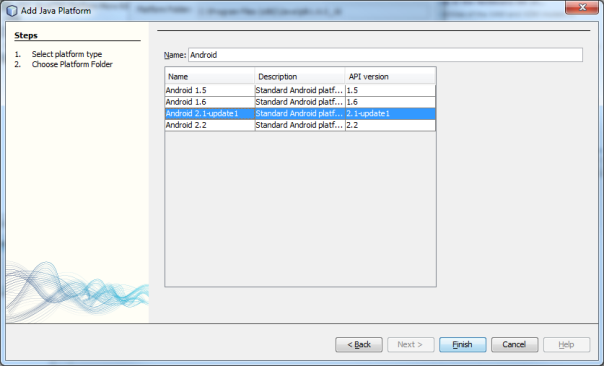
Choisir **Add Platform**

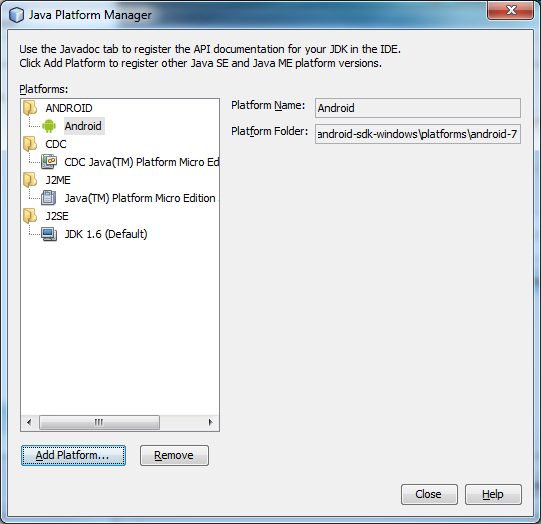


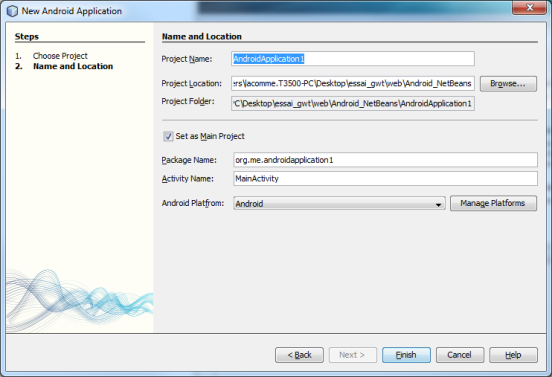
Allez dans le répertoire **android-sdk-windows / platforms.**

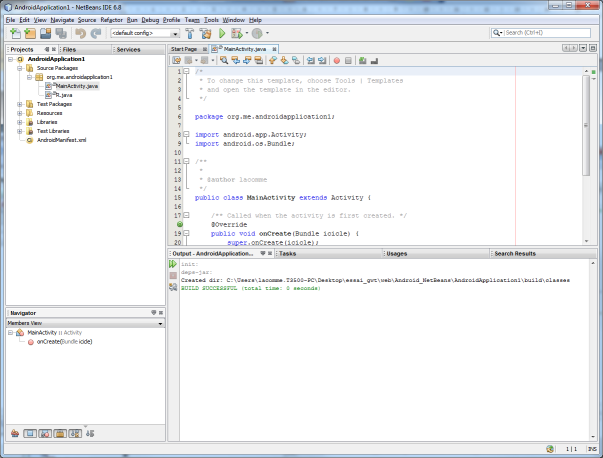












Modifier le programme principal comme suit :

package org.me.androidapplication1;

import android.app.Activity;

import android.os.Bundle;

import android.widget.TextView;

public class MainActivity extends Activity {

/\*\* Called when the activity is first created. \*/

@Override

public void onCreate(Bundle icicle) {

super.onCreate(icicle);

// ToDo add your GUI initialization code here

TextView tv = new TextView(this);

tv.setText("Hello, Android");

setContentView(tv);

}

}

Ceci devrait donner comme résultat d’exécution :





**3. Créer un bouton (un exemple simpliste !)**

Ouvrir le fichier nommé MainActivity.java et remplacer le code par le code suivant :

package org.me.androidapplication1;

import android.app.Activity;

import android.os.Bundle;

import android.widget.Button;

import android.widget.TextView;

public class MainActivity extends Activity {

/\*\* Called when the activity is first created. \*/

@Override

public void onCreate(Bundle icicle) {

super.onCreate(icicle);

// ToDo add your GUI initialization code here

Button mon\_bouton = new Button(this);

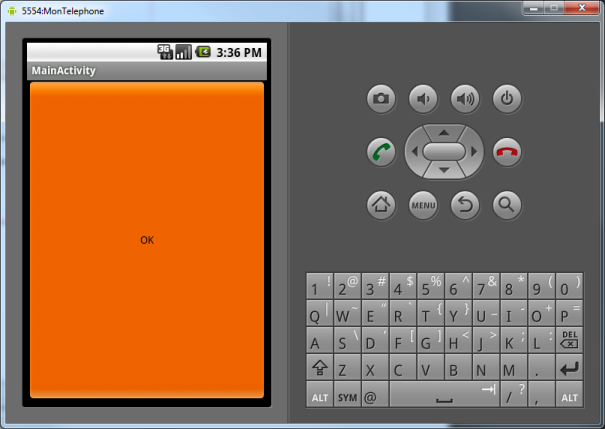
mon\_bouton.setText("OK");

setContentView(mon\_bouton);

}

}

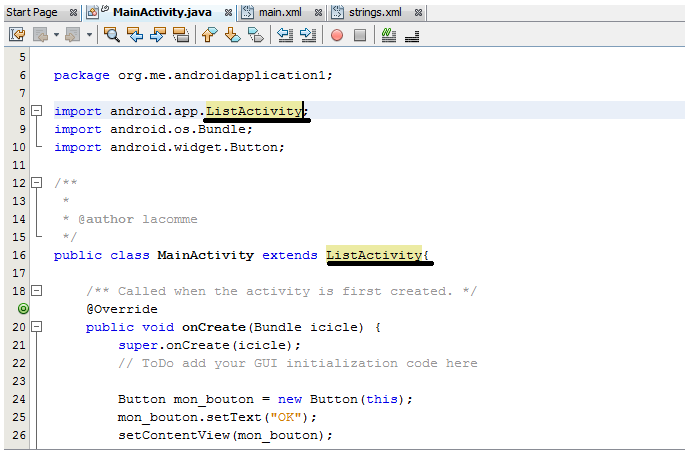
Le résultat d’exécution doit se présenter comme suit :



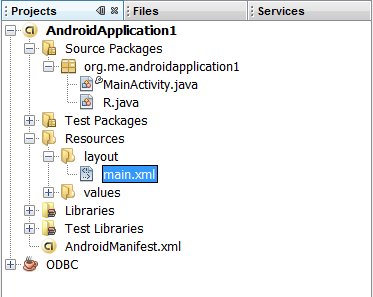
**4. Gestion des listes**

**Création d’une liste.**

Etape 1 : il faut modifier le type d’application crée. Remplacer Activity par ListActivity.



Etape 2 :



Modifier le fichier **main.xml** comme suit :

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

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:orientation="vertical"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent"

>

<ListView android:id="@android:id/list"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content">

</ListView>

</LinearLayout>

Etape 3. Modifier le code de MainActivity.java comme suit :

package org.me.androidapplication1;

import android.app.ListActivity;

import android.os.Bundle;

import android.widget.ArrayAdapter;

import android.widget.Button;

public class MainActivity extends ListActivity{

private String[] mStrings = {

"AAAAAAAA", "BBBBBBBB", "CCCCCCCC", "DDDDDDDD", "EEEEEEEE",

"FFFFFFFF", "GGGGGGGG", "HHHHHHHH", "IIIIIIII", "JJJJJJJJ",

"KKKKKKKK", "LLLLLLLL", "MMMMMMMM", "NNNNNNNN", "OOOOOOOO",

"PPPPPPPP", "QQQQQQQQ", "RRRRRRRR", "SSSSSSSS", "TTTTTTTT",

"UUUUUUUU", "VVVVVVVV", "WWWWWWWW", "XXXXXXXX", "YYYYYYYY",

"ZZZZZZZZ"

};

/\*\* Called when the activity is first created. \*/

@Override

public void onCreate(Bundle icicle) {

super.onCreate(icicle);

// ToDo add your GUI initialization code here

setContentView(R.layout.main);

ArrayAdapter<String> adapter = new ArrayAdapter<String>(this, android.R.layout.simple\_list\_item\_1, mStrings);

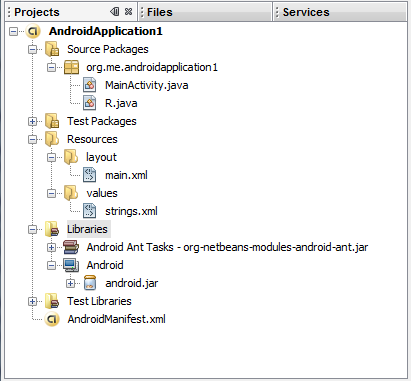
setListAdapter(adapter);

}

}

Etape 4 : Inclure la librairies Android Ant Task.

Le projet doit ressembler à ce qui suit :



Etape 5 : Tester la liste.

