



Créer des applications Android

Auteurs : Philippe Lacomme, Raksmei Phan

Plan (ceci n'est pas un cours)

Les outils nécessaires sont :

- Android SDK
- Eclipse
- Le plugin ADT de Eclipse

Outils: Préparation de l'environnement

1. Installation plugin ADT pour Android dans Eclipse
2. Installation d'un téléphone virtuel Android

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



Ce document s'inspire, reprend... des supports de cours disponibles sur Internet.

Ces supports ont servi de base, ils ont été enrichi (on l'espère), modifiés et complétés. Certaines parties du code ont été obtenues en suivant les recommandations de ces supports.

On peut citer :

<http://mickael-lt.developpez.com/tutoriels/android/personnaliser-listview/>

<http://michel-dirix.developpez.com/tutoriels/android/integration-google-maps-android/>

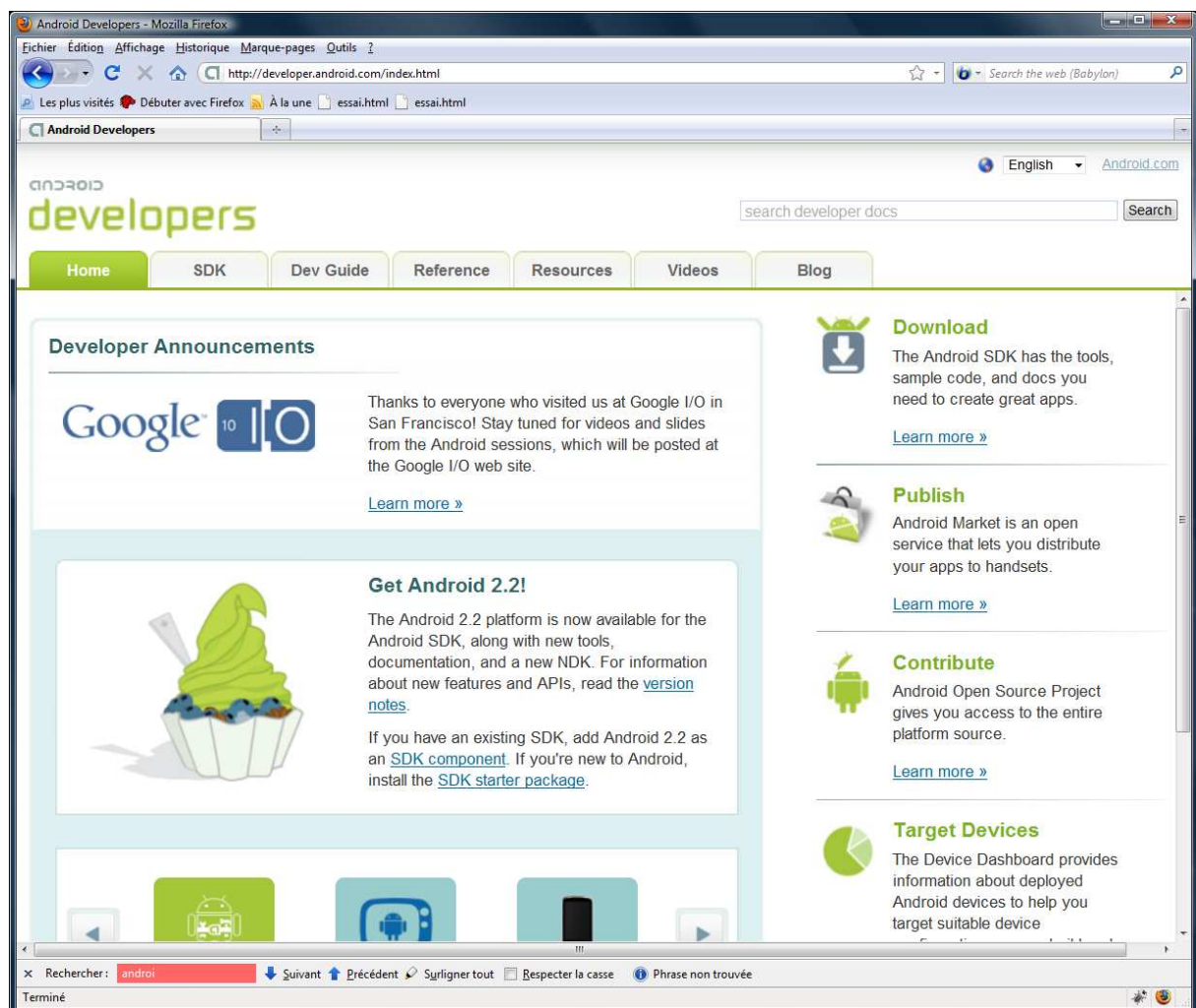
1. Choix d'une version d'Eclipse

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

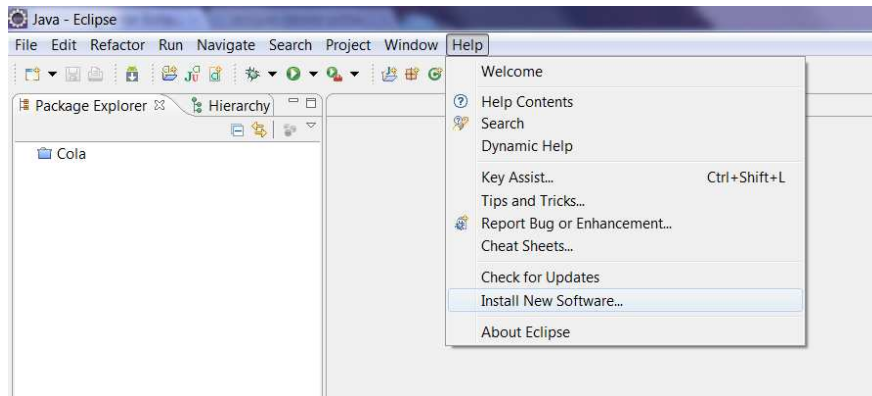
Voici le lien pour télécharger la version 32 bits que nous avons utilisé : <http://fc.isima.fr/~phan/tuto/ApplicationAndroid/eclipse-java-helios-win32.zip>

2. Installation du plugin ADT pour Eclipse

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



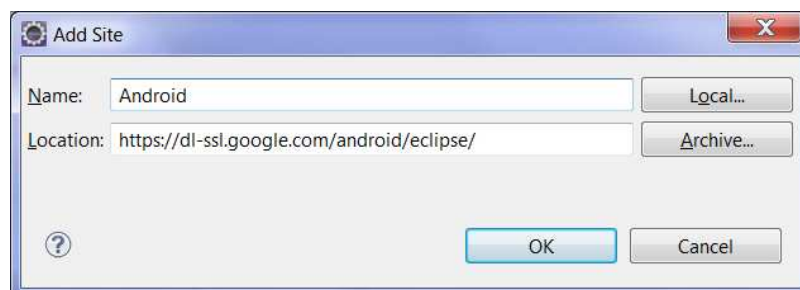
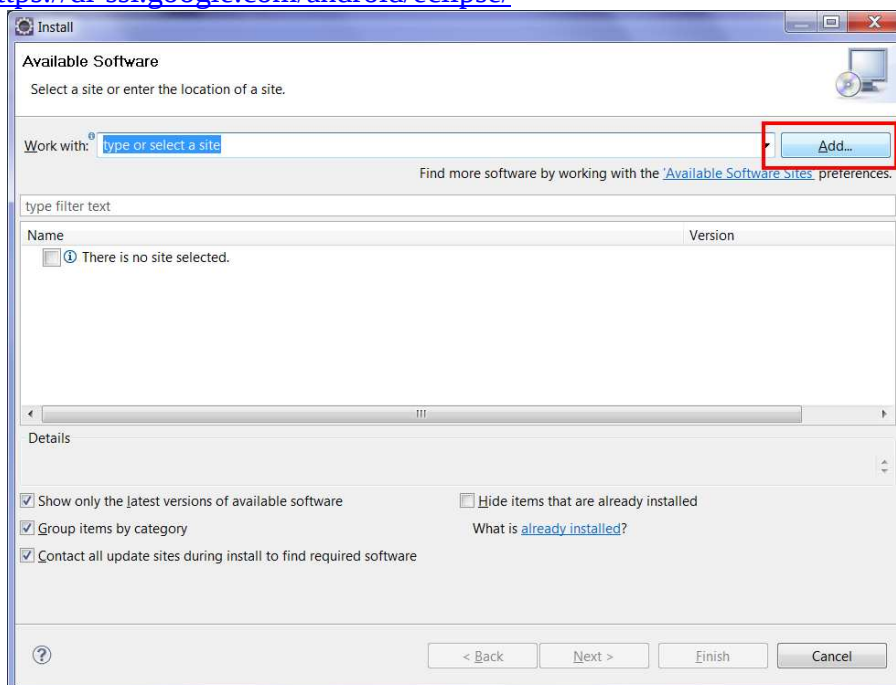
Installer un nouveau « Software ».



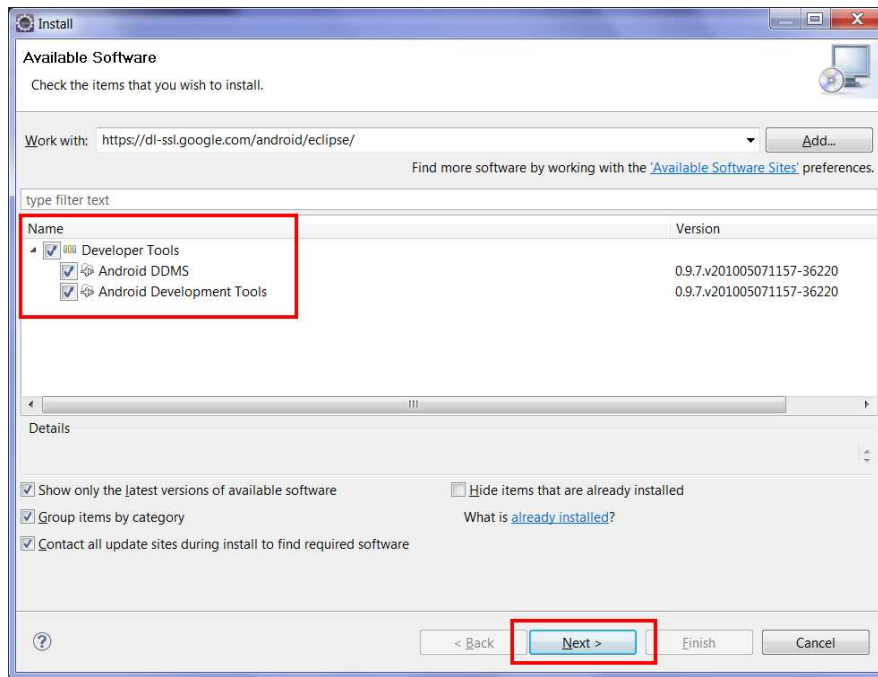
Ajouter un nouveau site.

Nom : Android

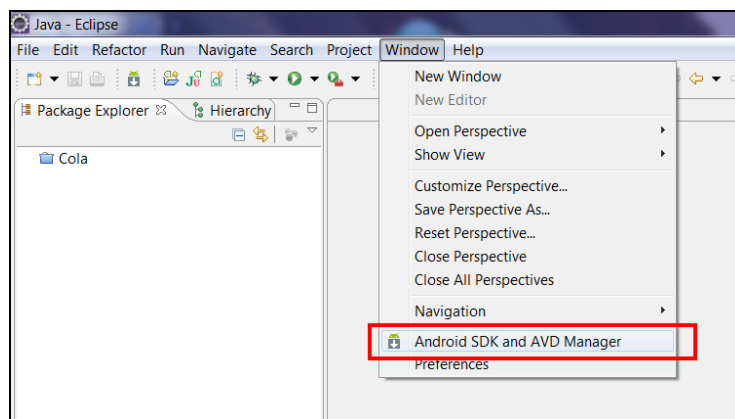
Adresse : <https://dl-ssl.google.com/android/eclipse/>



Sélectionner tout le package à télécharger. Et installer les.



Puis redémarrer Eclipse. Android devrait être installé et visible.



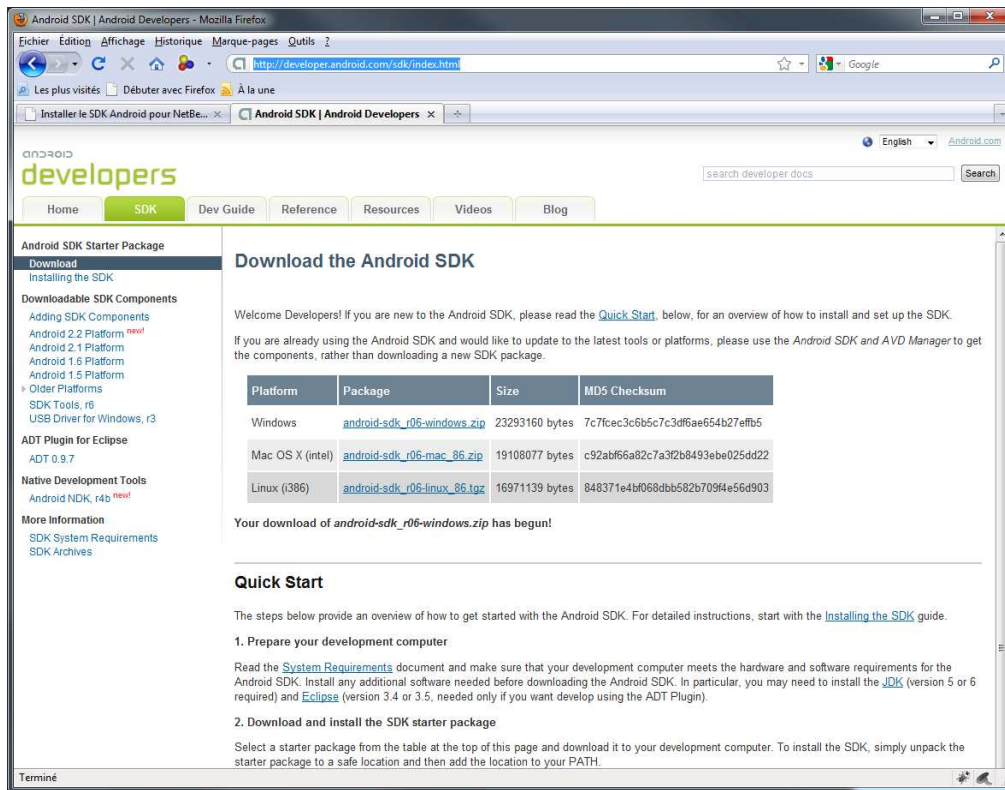
3. Installation d'un téléphone virtuel Android

Remarque :

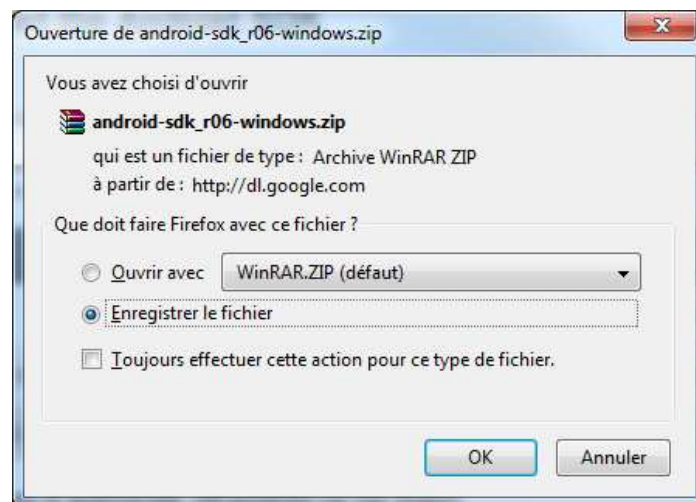
Android SDK est téléchargeable ici : <http://developer.android.com/sdk/index.html>

Ou bien ici : fc.isima.fr/~phan/tuto/ApplicationAndroid/android-sdk_r06-windows.zip

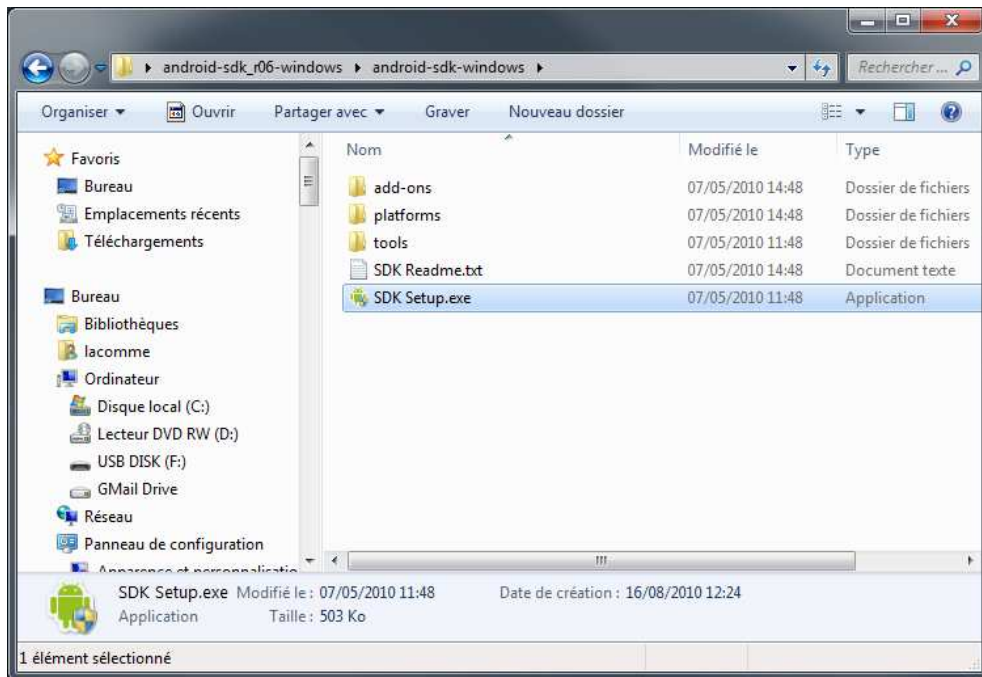
Son installation et configuration mérite quelques éclaircissements.



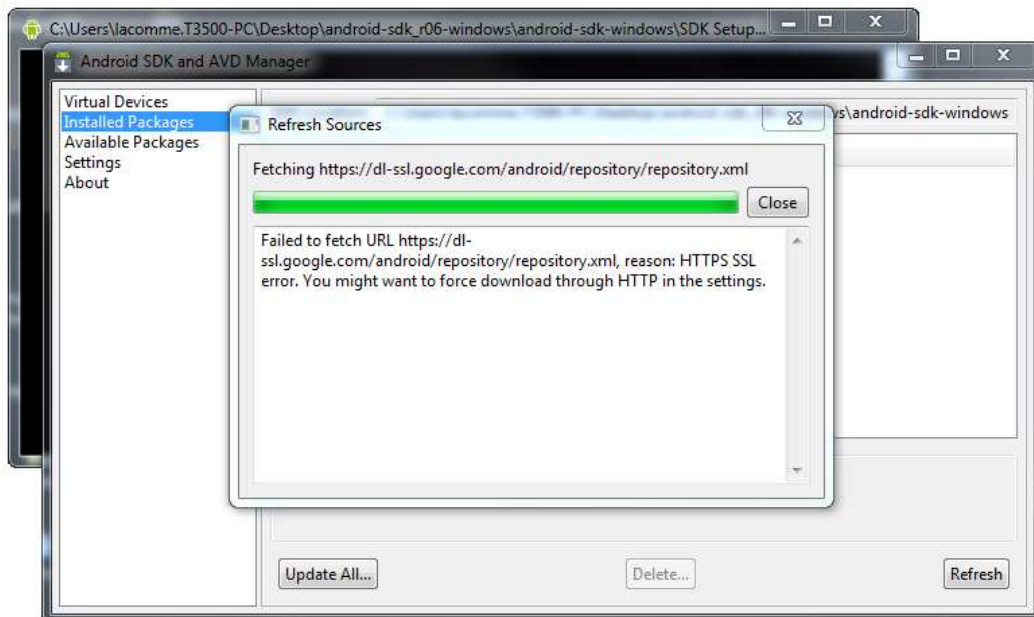
Il suffit de choisir la version correspondante à votre système, ici Windows :



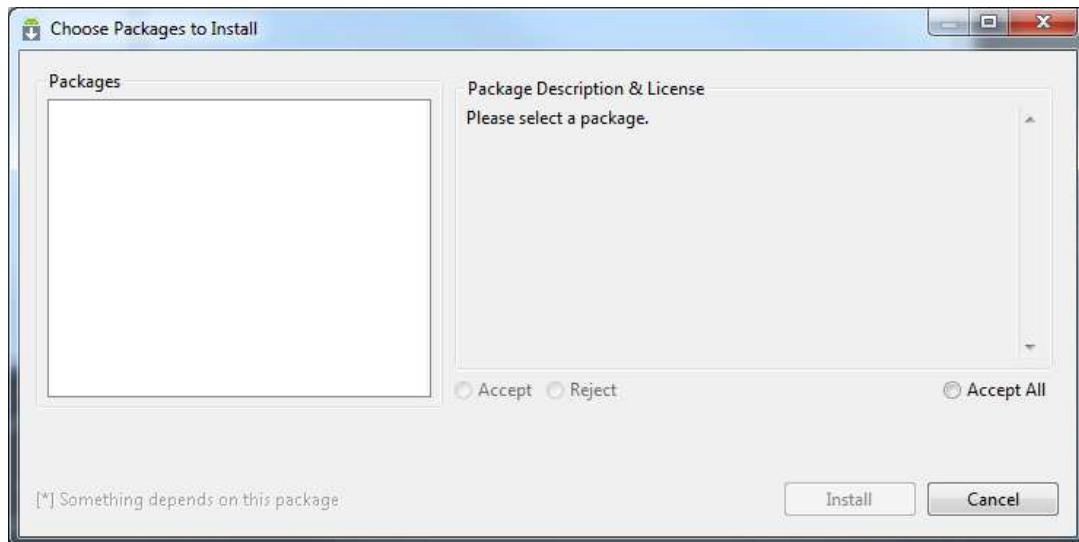
Ceci donne sur le disque dur un répertoire **android-sdk-windows** contenant un exécutable :



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

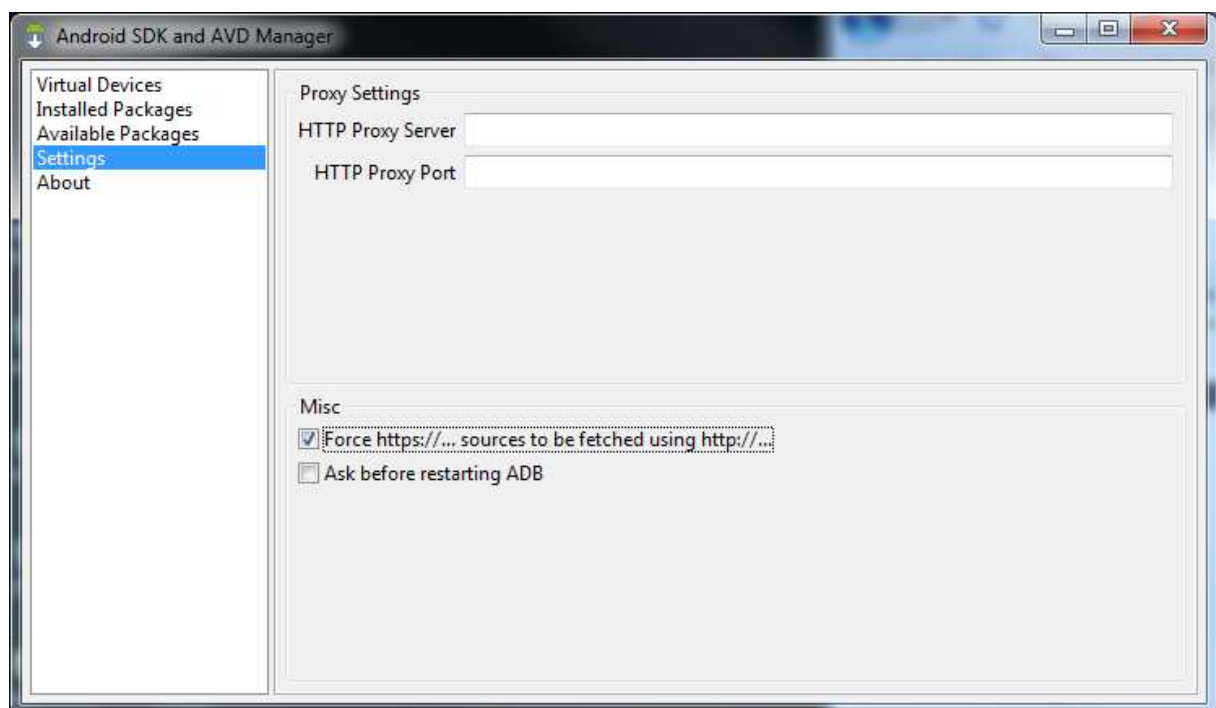


Fermer la fenêtre courante en utilisant le bouton **close**.

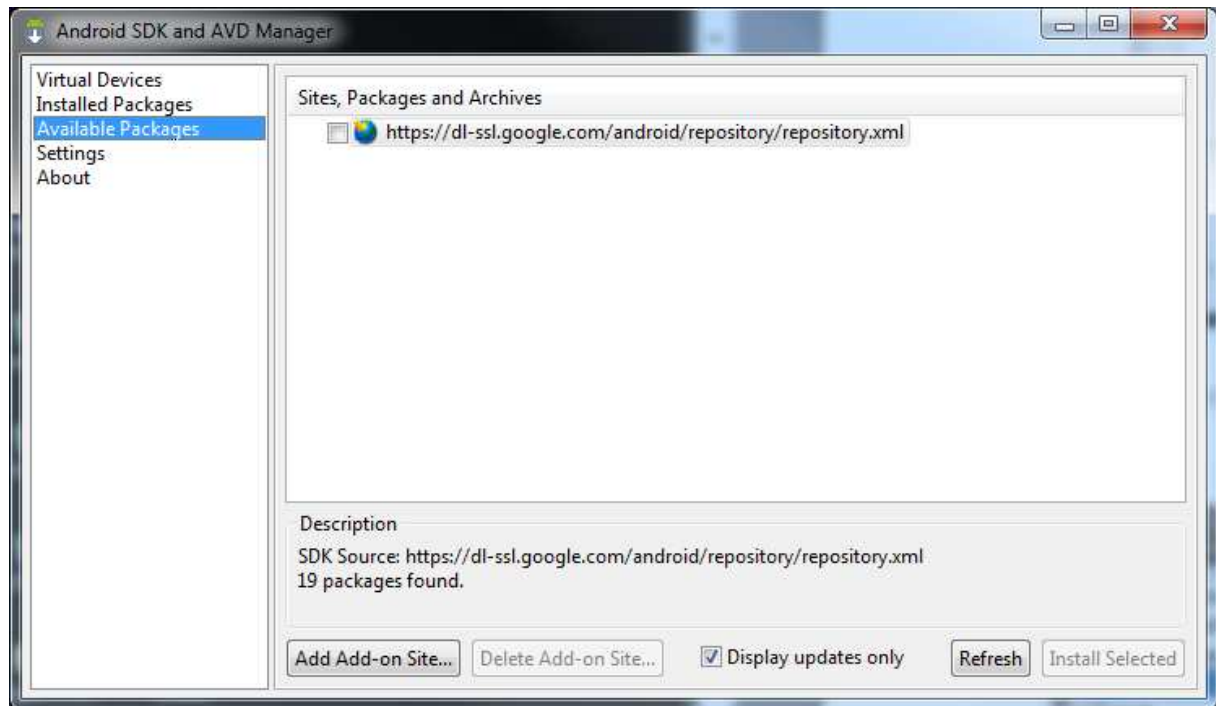


Fermer la deuxième fenêtre avec le bouton **Cancel**.

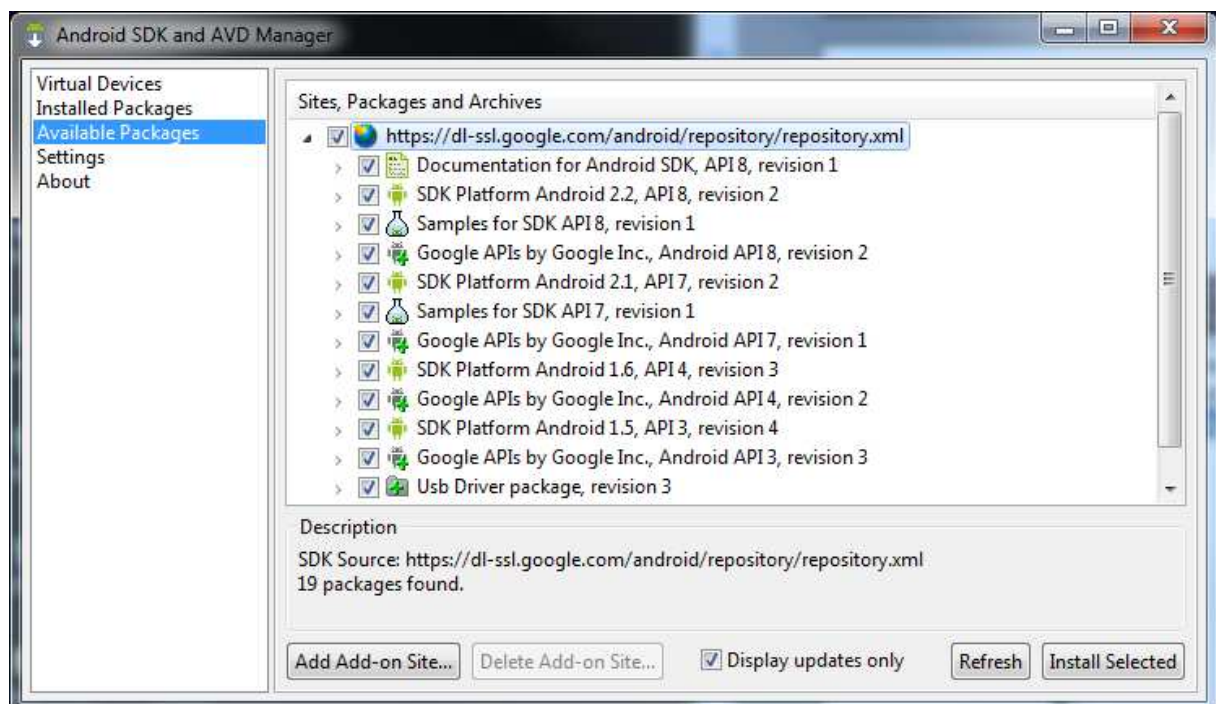
Allez dans le sous-menu **Settings** et cocher **Force https**



Choisir ensuite **Available Packages**.

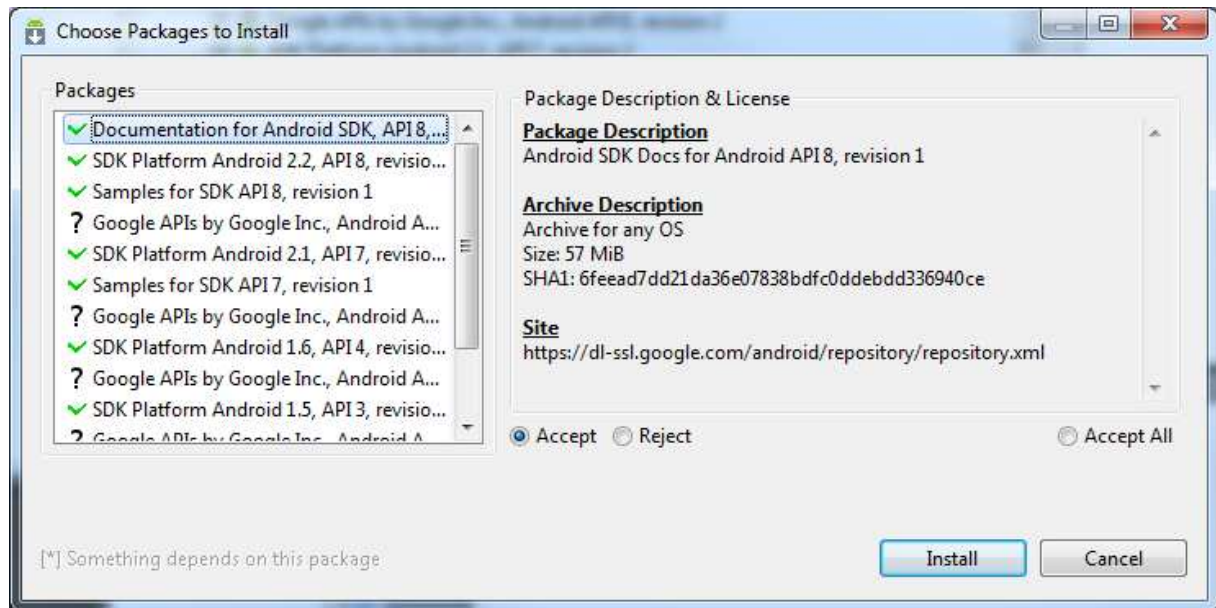


Cocher **https**.

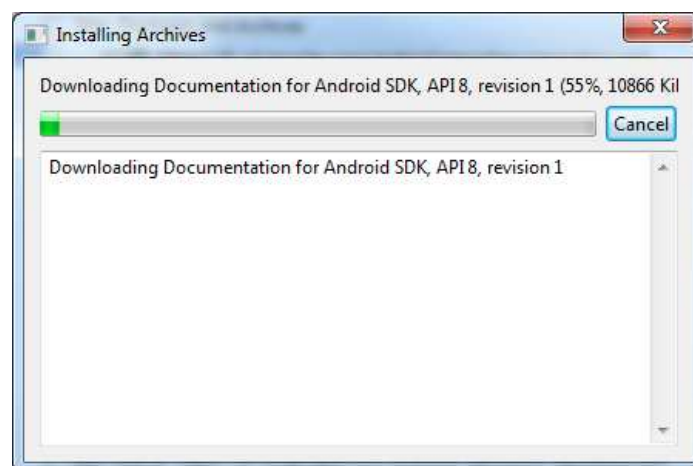


Choisir ensuite

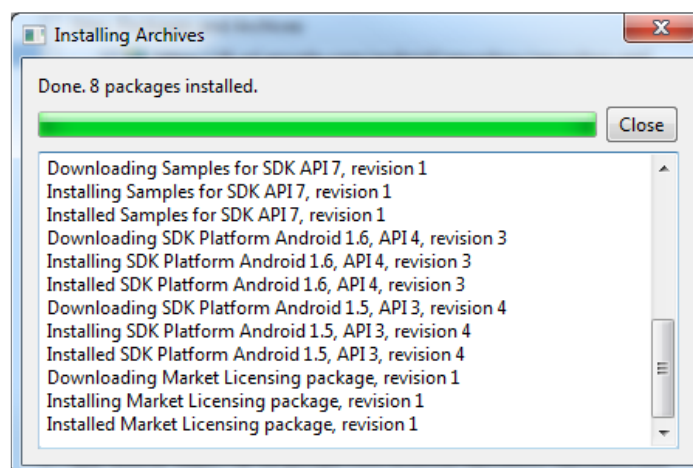
Install Selected



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

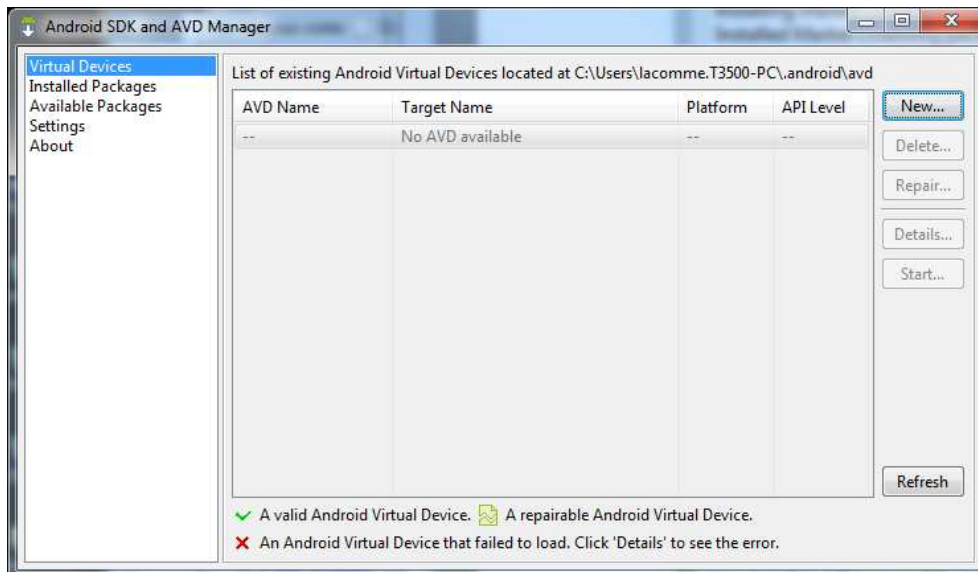


Cliquer sur **Close** à la fin des téléchargements.

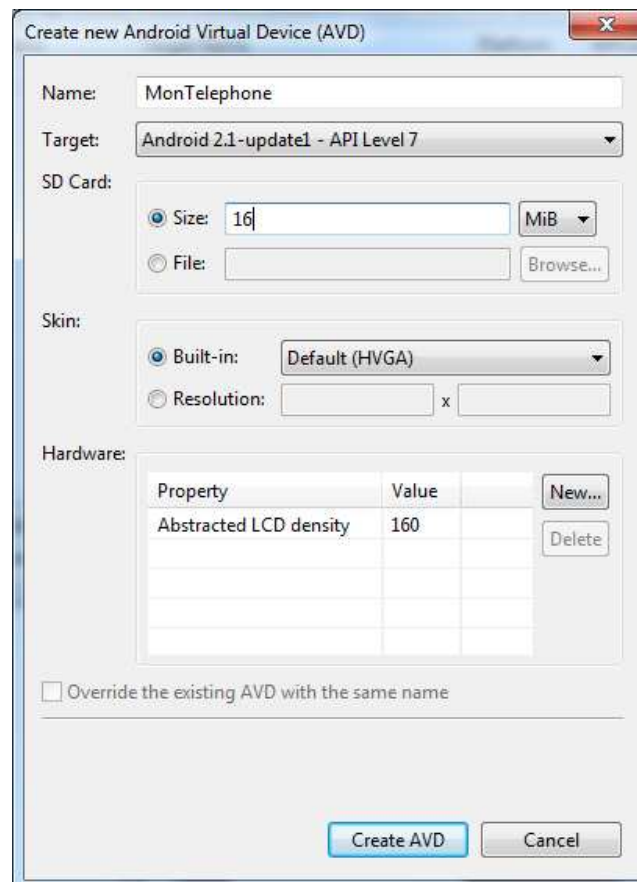


Il ne reste plus qu'à créer un Virtual Devices qui comme son nom l'indique sera un téléphone Android Virtuel.

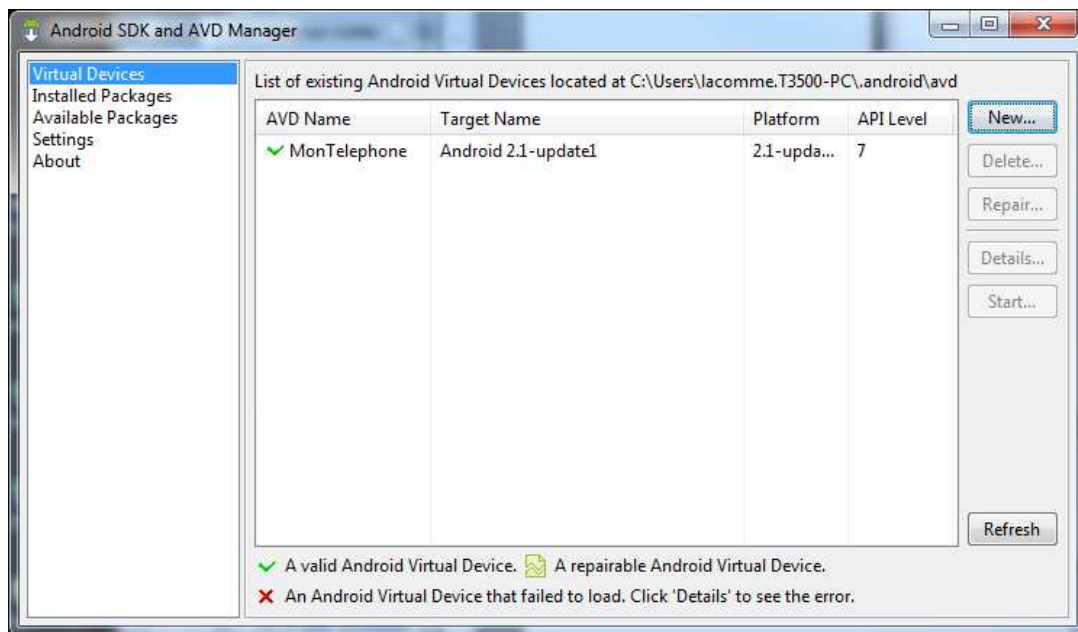
Choisir le menu **Virtual Devices**.



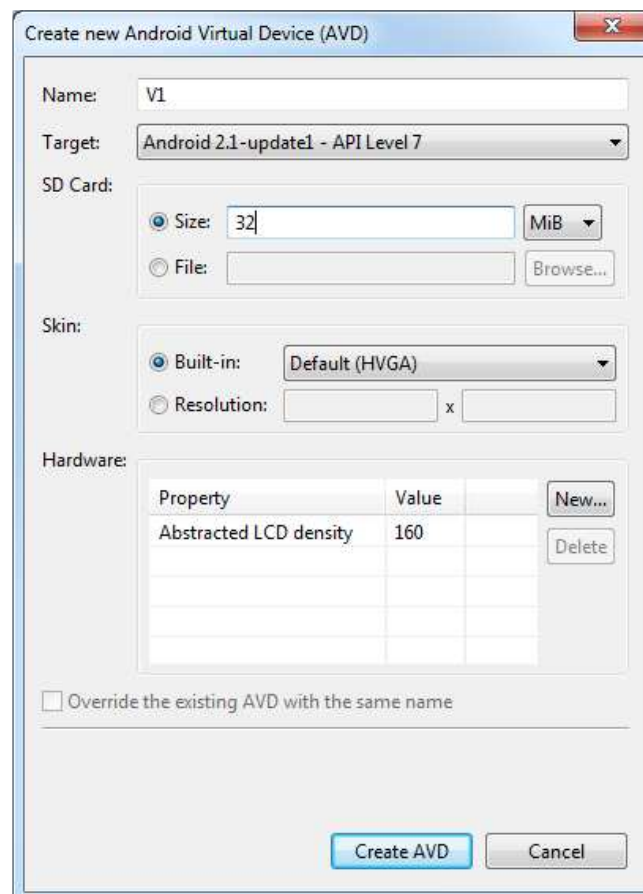
On peut créer sa propre configuration comme sur l'exemple ci-dessous :



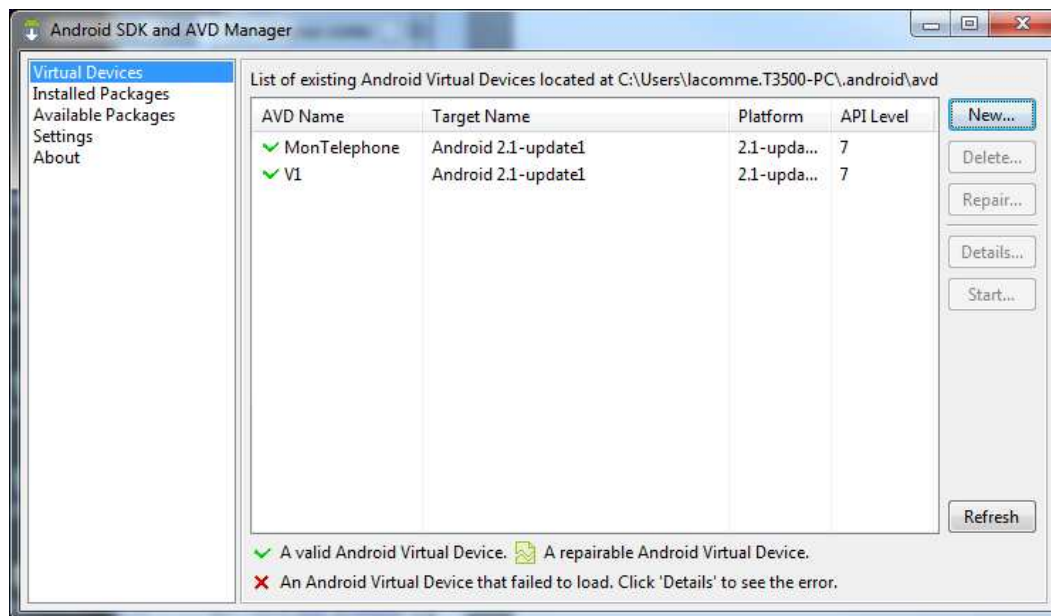
Le téléphone virtuel ainsi créé doit apparaître dans la liste des Virtual Devices.



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 :

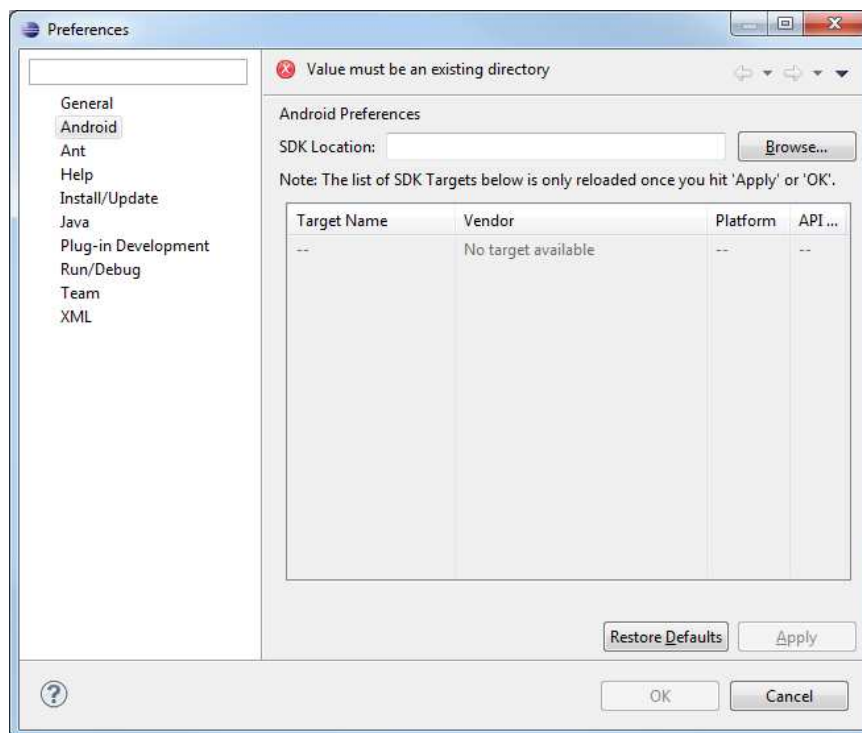
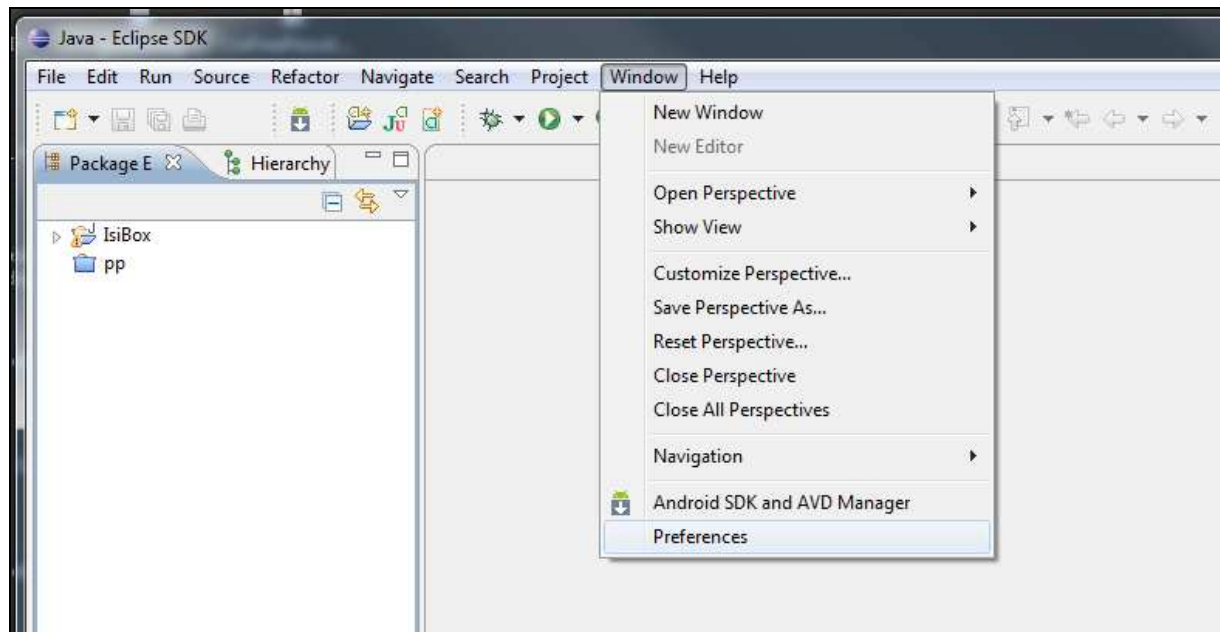


1. Création d'un projet ECLIPSE

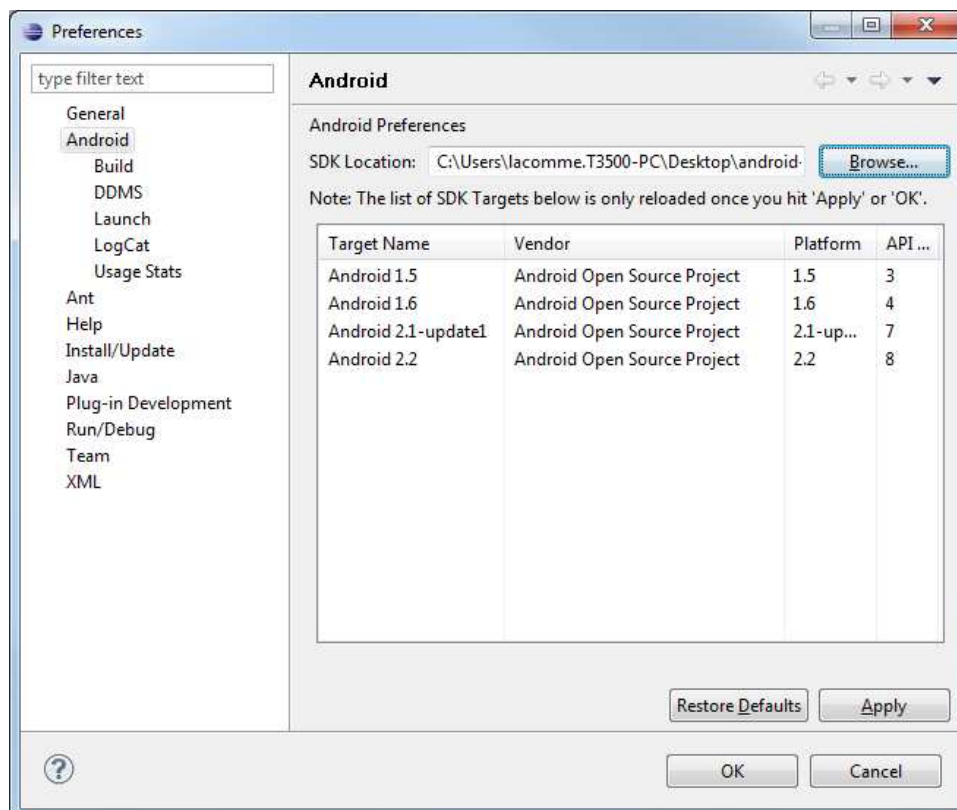
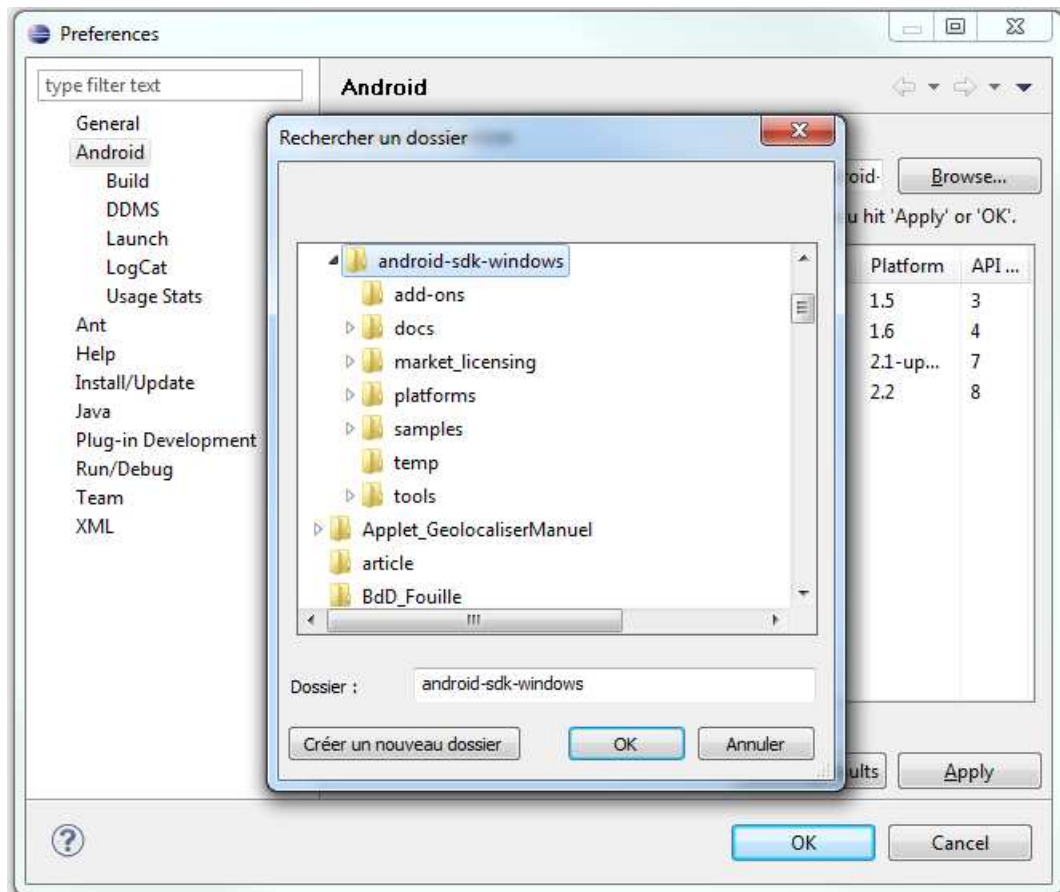
Démarrer Eclipse.



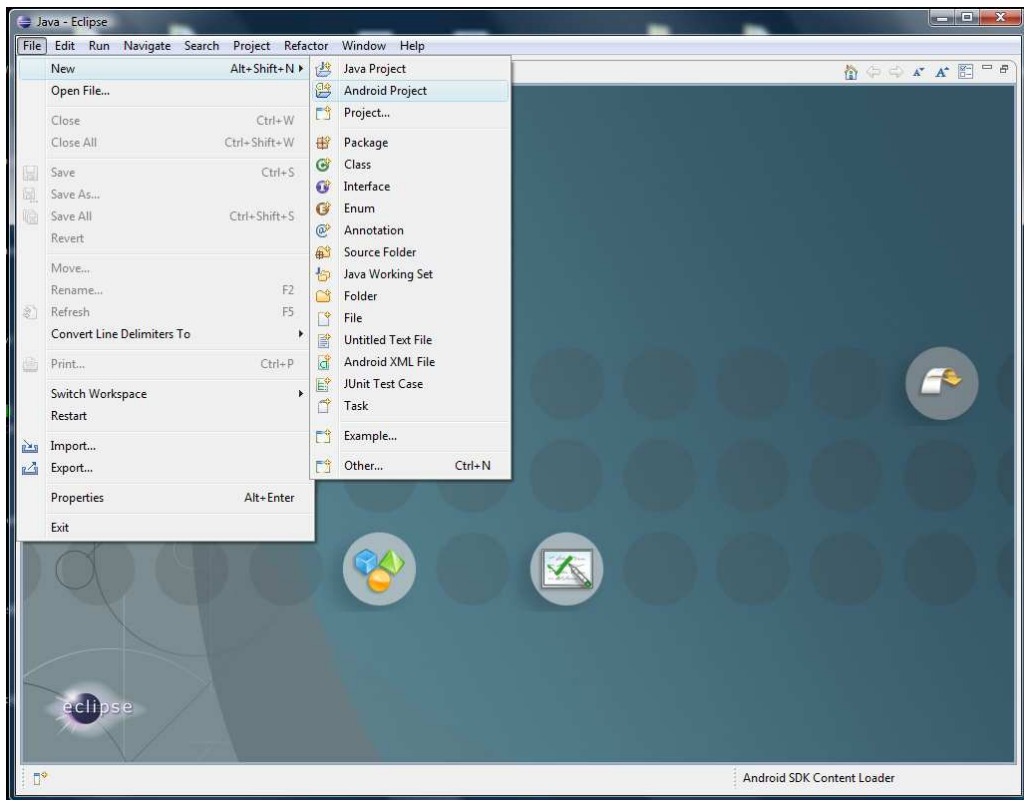
Allez dans **Windows / Preferences**.



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

New Android Project

Creates a new Android Project resource.

Project name:

Contents

☒ Create new project in workspace

☐ Create project from existing source

☐ Use default location

Location:

☐ Create project from existing sample

Samples:

Build Target

Target Name	Vendor	Platform	API ...
☐ Android 1.1	Android Open Source Project	1.1	2
☐ Android 1.5	Android Open Source Project	1.5	3
☐ Android 1.6	Android Open Source Project	1.6	4
☒ Android 2.0	Android Open Source Project	2.0	5
☐ Android 2.0.1	Android Open Source Project	2.0.1	6
☐ Android 2.1-upda...	Android Open Source Project	2.1-upd...	7
☐ Android 2.2	Android Open Source Project	2.2	8
☐ Google APIs	Google Inc.	2.2	8

Standard Android platform 2.0

Properties

Application name:

Package name:

☒ Create Activity:

Min SDK Version:

New Android Project

New Android Test Project
Creates a new Android Test Project resource.

☐ Create a Test Project

Test Project Name:

Content

☒ Use default location

Location:

Test Target

Test Target Package:

Build Target

Target Name	Vendor	Platform	API ...
☐ Android 1.1	Android Open Source Project	1.1	2
☐ Android 1.5	Android Open Source Project	1.5	3
☐ Android 1.6	Android Open Source Project	1.6	4
☐ Android 2.0	Android Open Source Project	2.0	5
☐ Android 2.0.1	Android Open Source Project	2.0.1	6
☐ Android 2.1-upda...	Android Open Source Project	2.1-upd...	7
☐ Android 2.2	Android Open Source Project	2.2	8
☐ Google APIs	Google Inc.	2.2	8

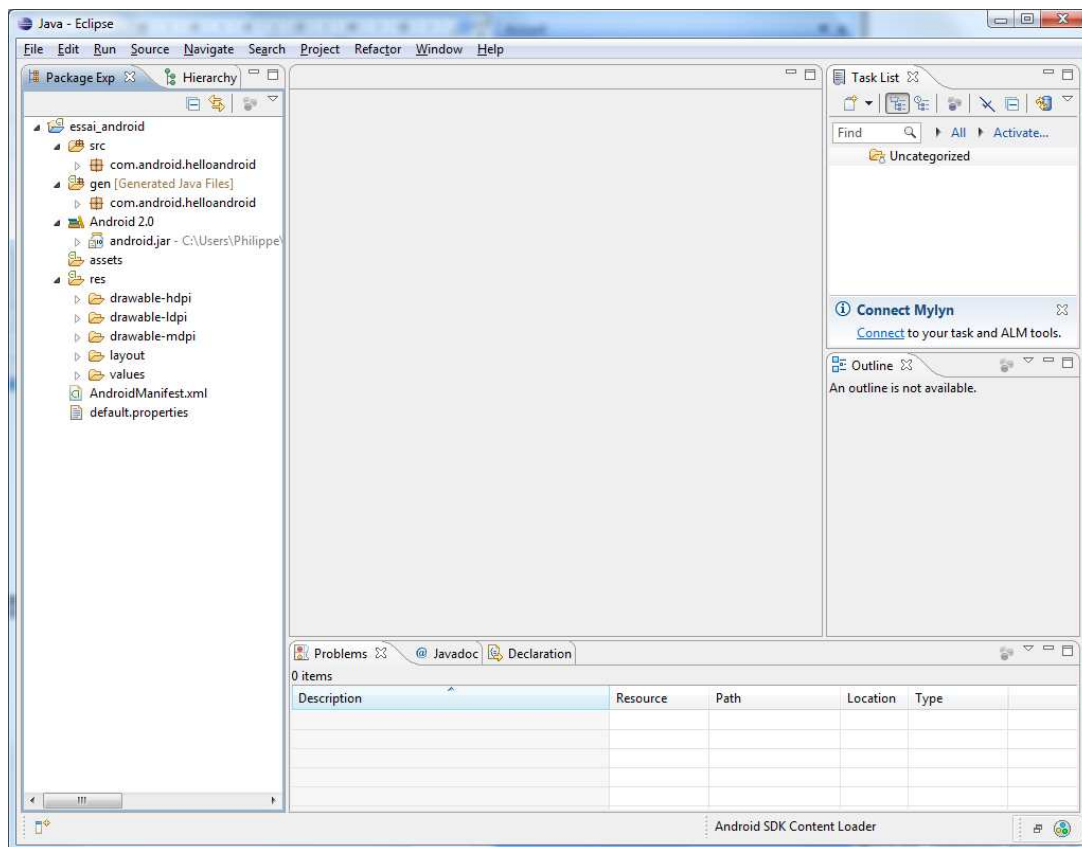
Properties

Application name:

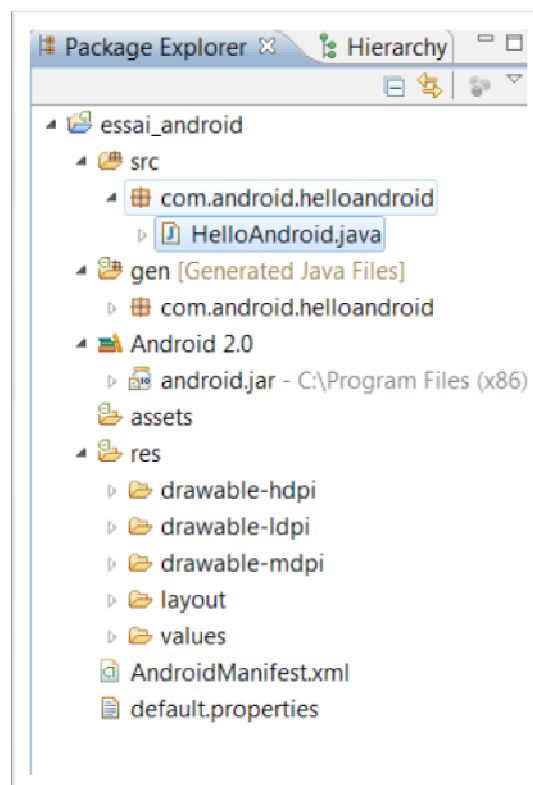
Package name:

Min. SDK Version:

Ceci devrait donner :



Le projet se présente comme ceci :



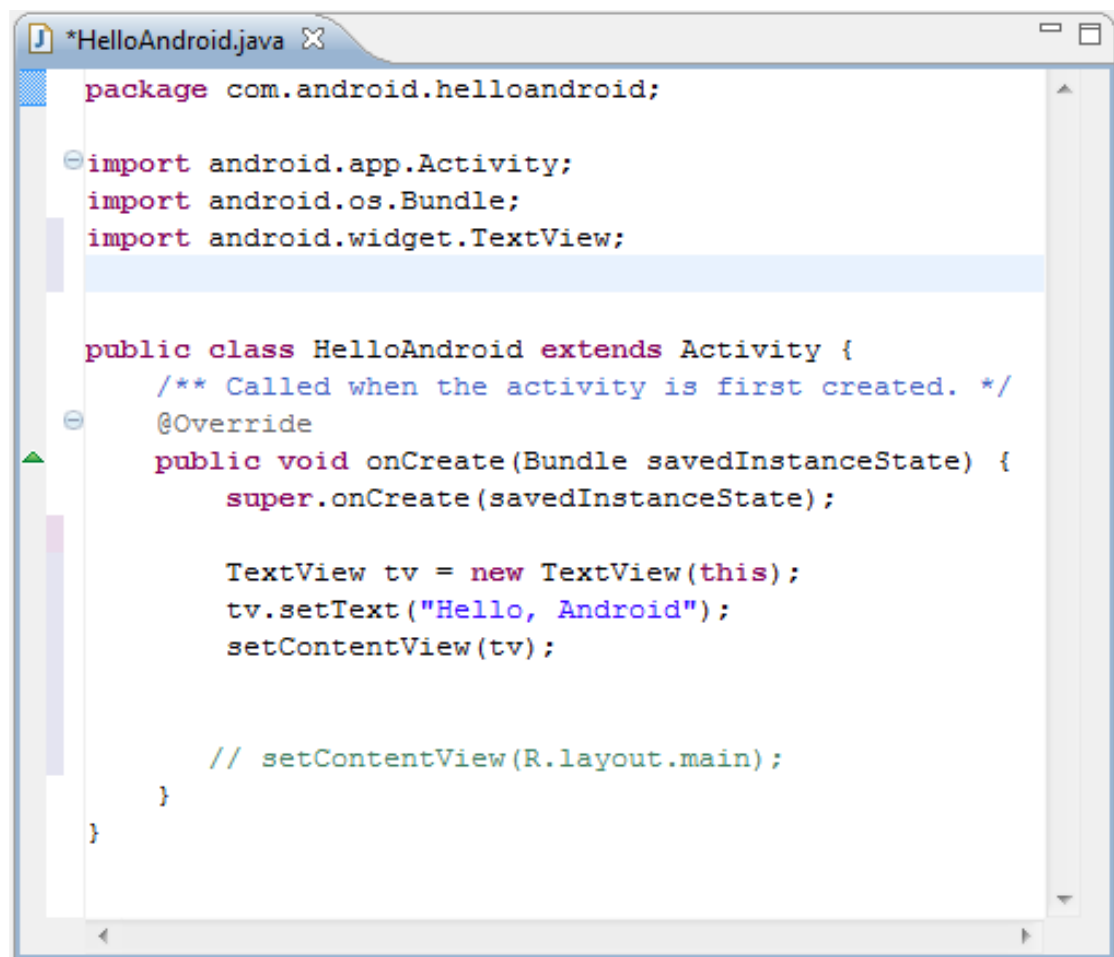
Modifier le code du fichier HelloAndroid.java.

```
package com.android.helloandroid;

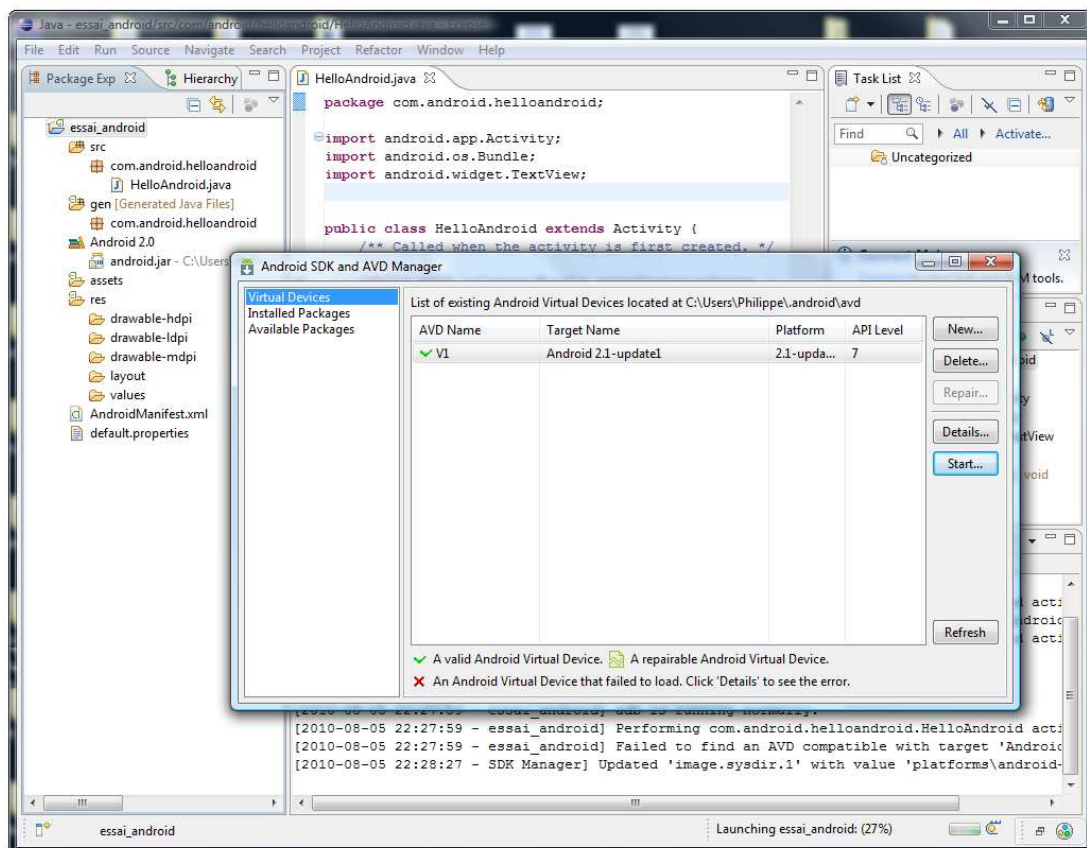
import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;

public class HelloAndroid extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

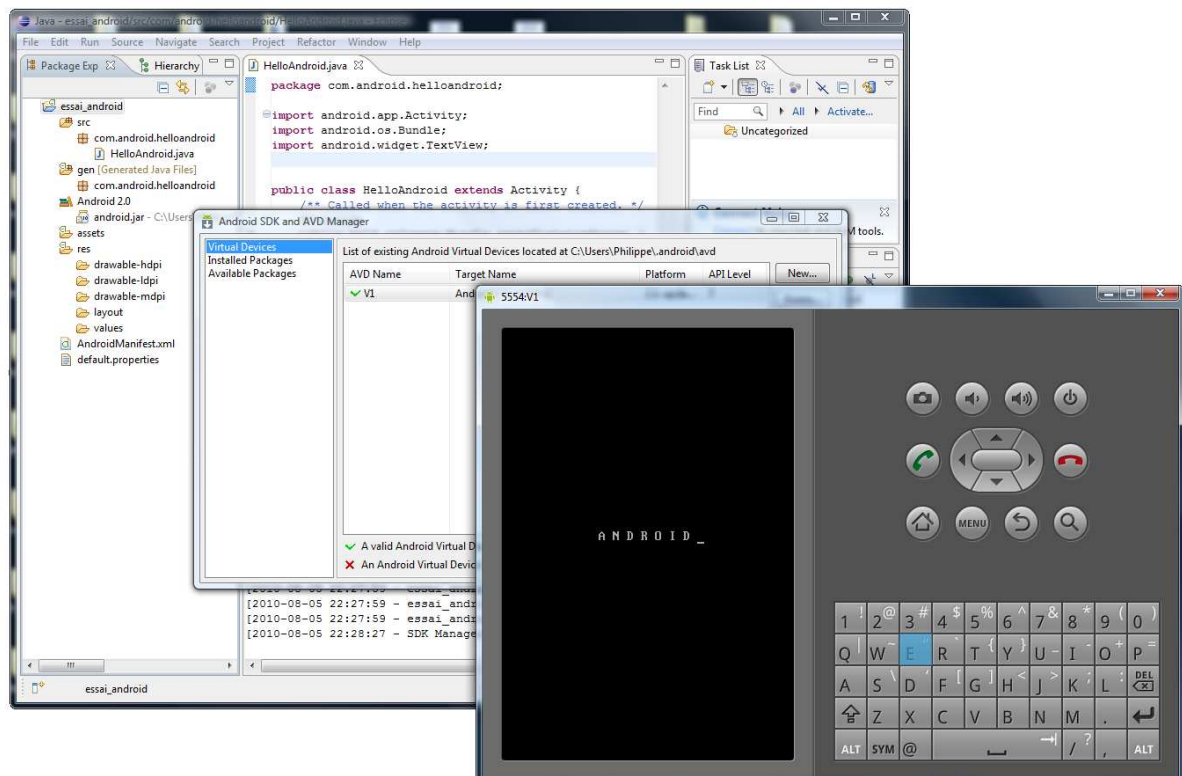
        TextView tv = new TextView(this);
        tv.setText("Hello, Android");
        setContentView(tv);
    }
}
```



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 ».

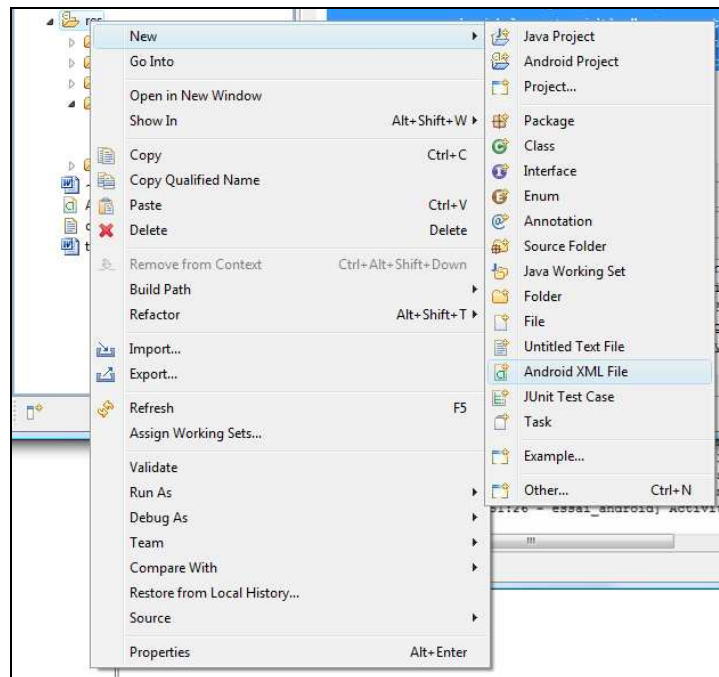


Ce qui donne :

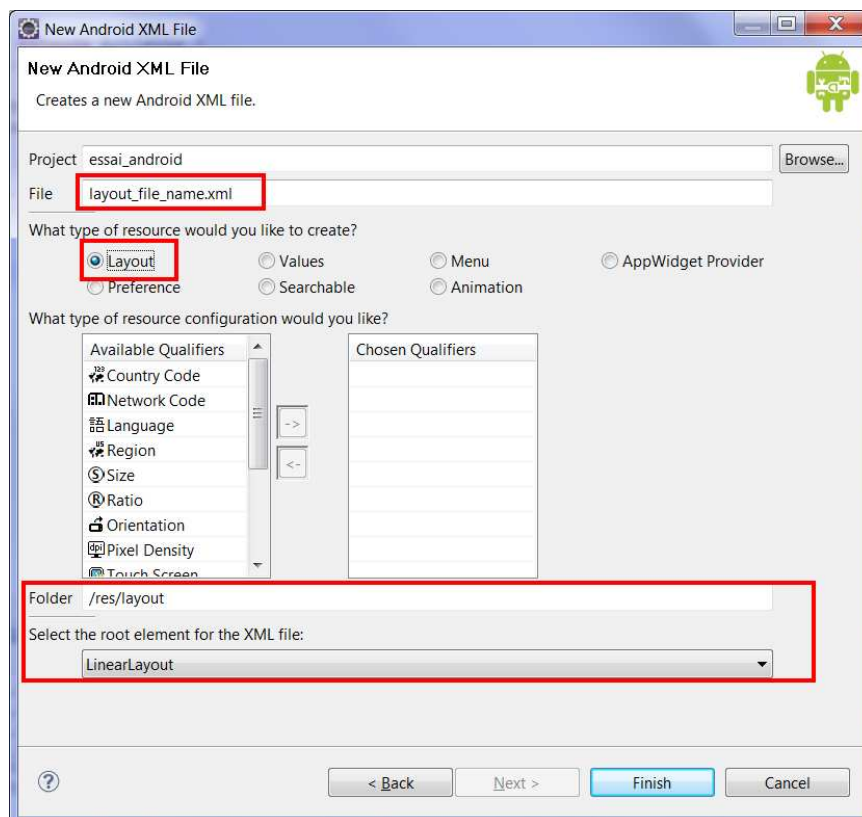


3. Gestion d'une l'interface

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



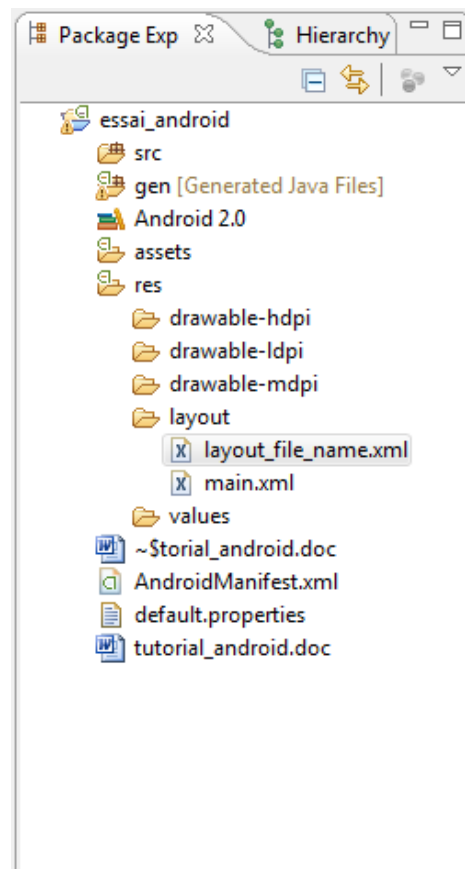
Choisir comme nom : **layout_file_name.xml**



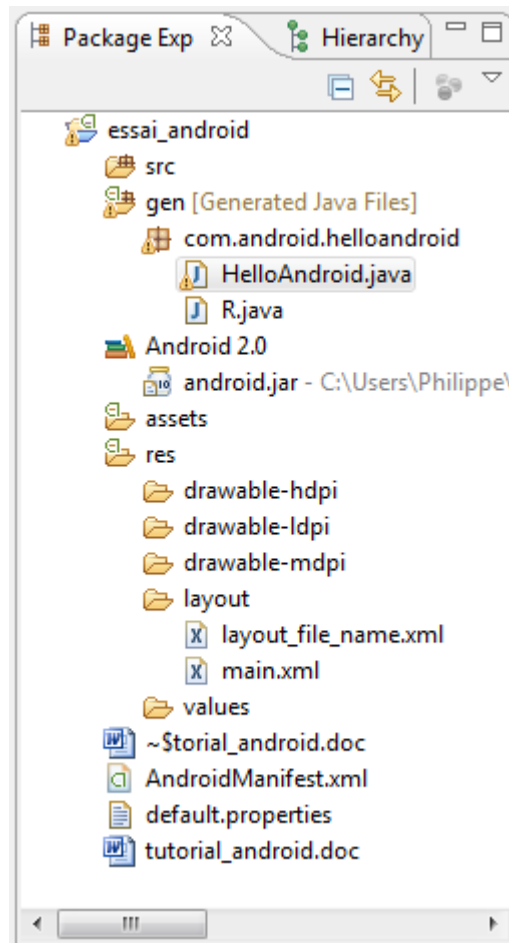
Saisir le texte suivant dans le fichier **layout_file_name.xml**

```
<?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.**



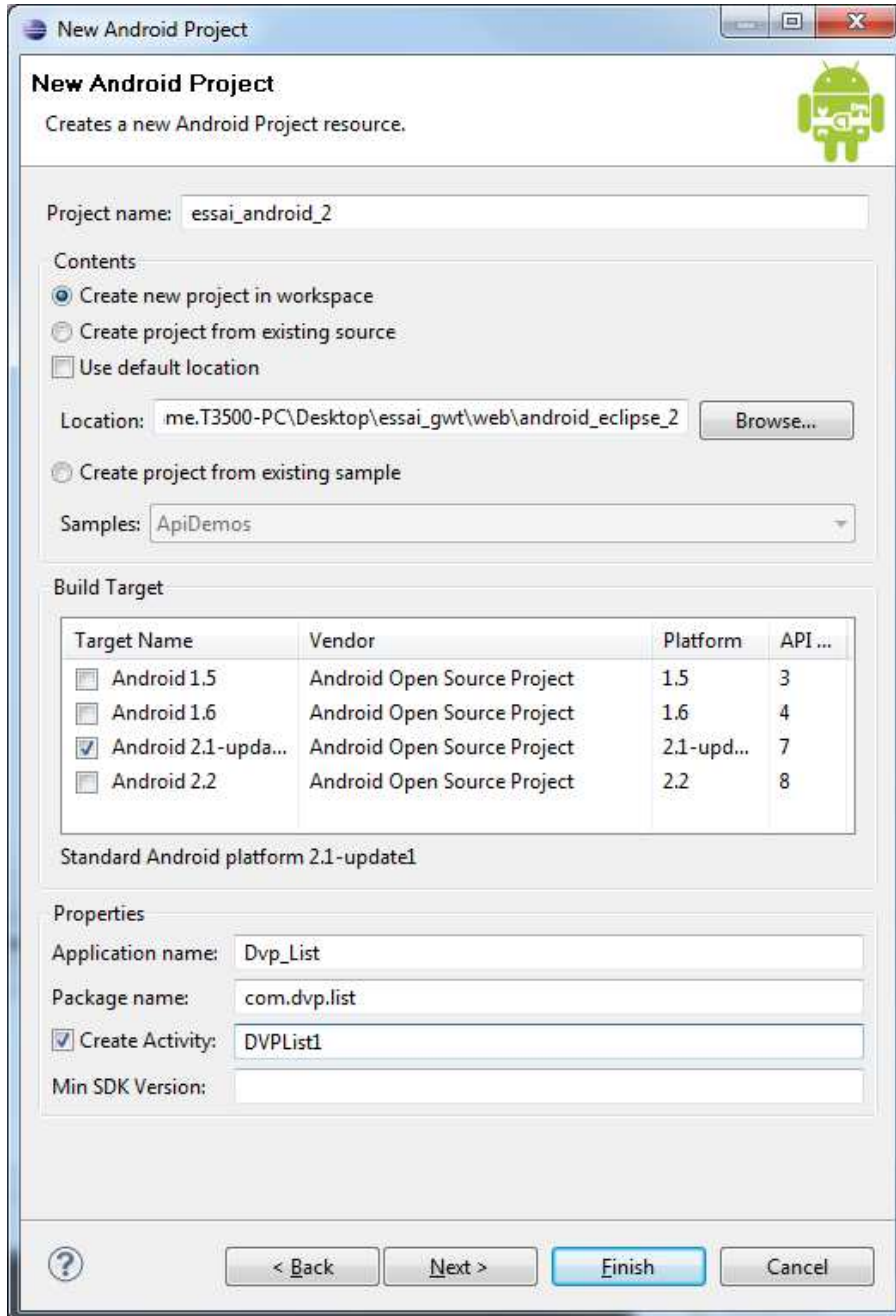
4. Gestion des listes déroulantes

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`



New Android Project
Creates a new Android Project resource.

Project name:

Contents

- ☒ Create new project in workspace
- ☐ Create project from existing source
- ☐ Use default location

Location:

☐ Create project from existing sample

Samples:

Build Target

Target Name	Vendor	Platform	API ...
☐ Android 1.5	Android Open Source Project	1.5	3
☐ Android 1.6	Android Open Source Project	1.6	4
☒ Android 2.1-upda...	Android Open Source Project	2.1-upd...	7
☐ Android 2.2	Android Open Source Project	2.2	8

Standard Android platform 2.1-update1

Properties

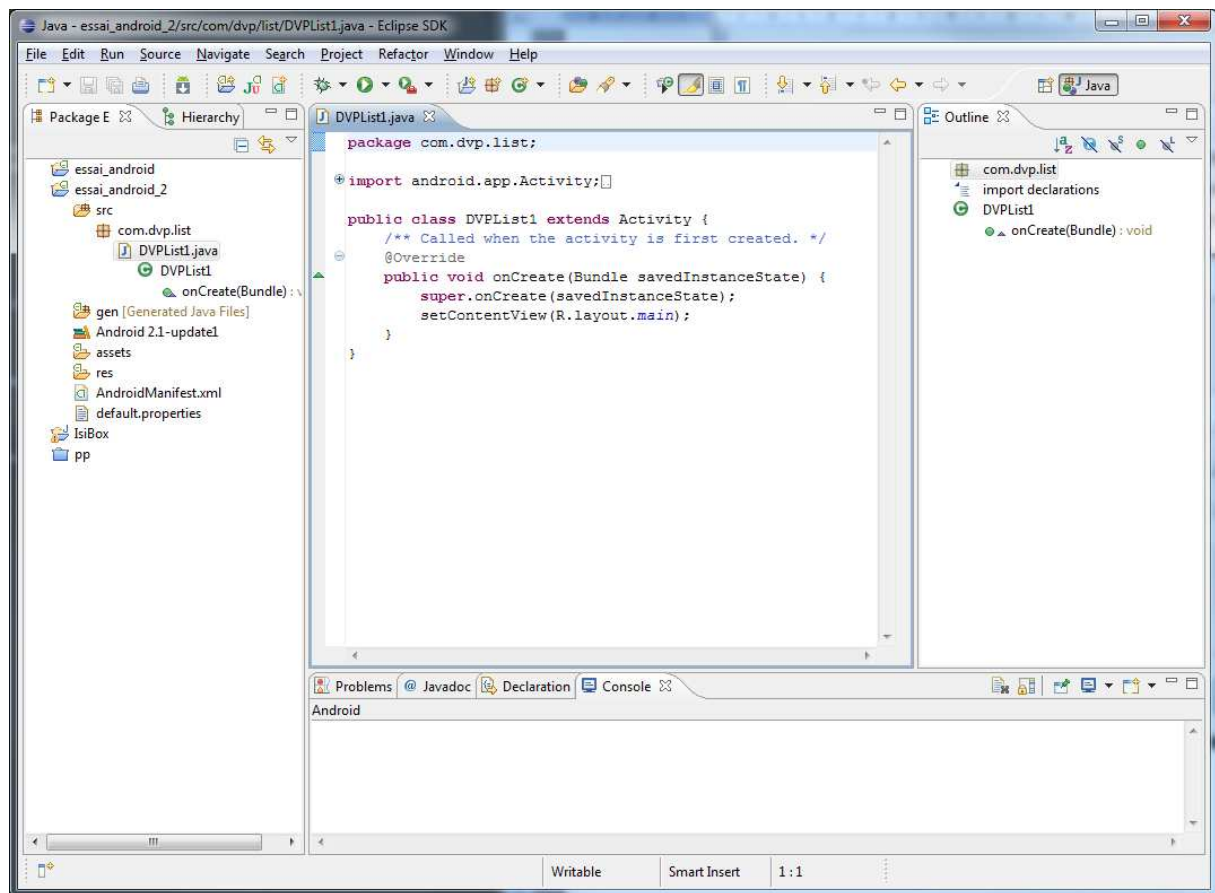
Application name:

Package name:

☒ Create Activity:

Min SDK Version:

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;

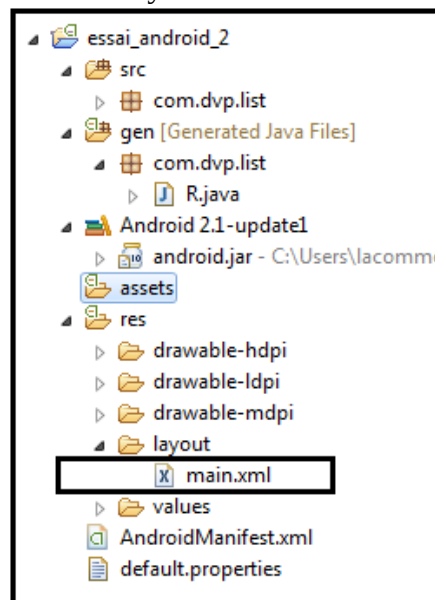
import android.app.Activity;

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 = {
        "AAAAAAA", "BBBBBBB", "CCCCCCC", "DDDDDDD", "EEEEEEE",
        "FFFFFFF", "GGGGGGG", "HHHHHHH", "IIIIIII", "JJJJJJJ",
        "KKKKKKK", "LLLLLLL", "MMMMMMM", "NNNNNNN", "OOOOOOO",
        "PPPPPPP", "QQQQQQQ", "RRRRRRR", "SSSSSSS", "TTTTTTT",
        "UUUUUUU", "VVVVVVV", "WWWWWWW", "XXXXXXXXX", "YYYYYYY",
        "ZZZZZZZ"
    };

    @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).

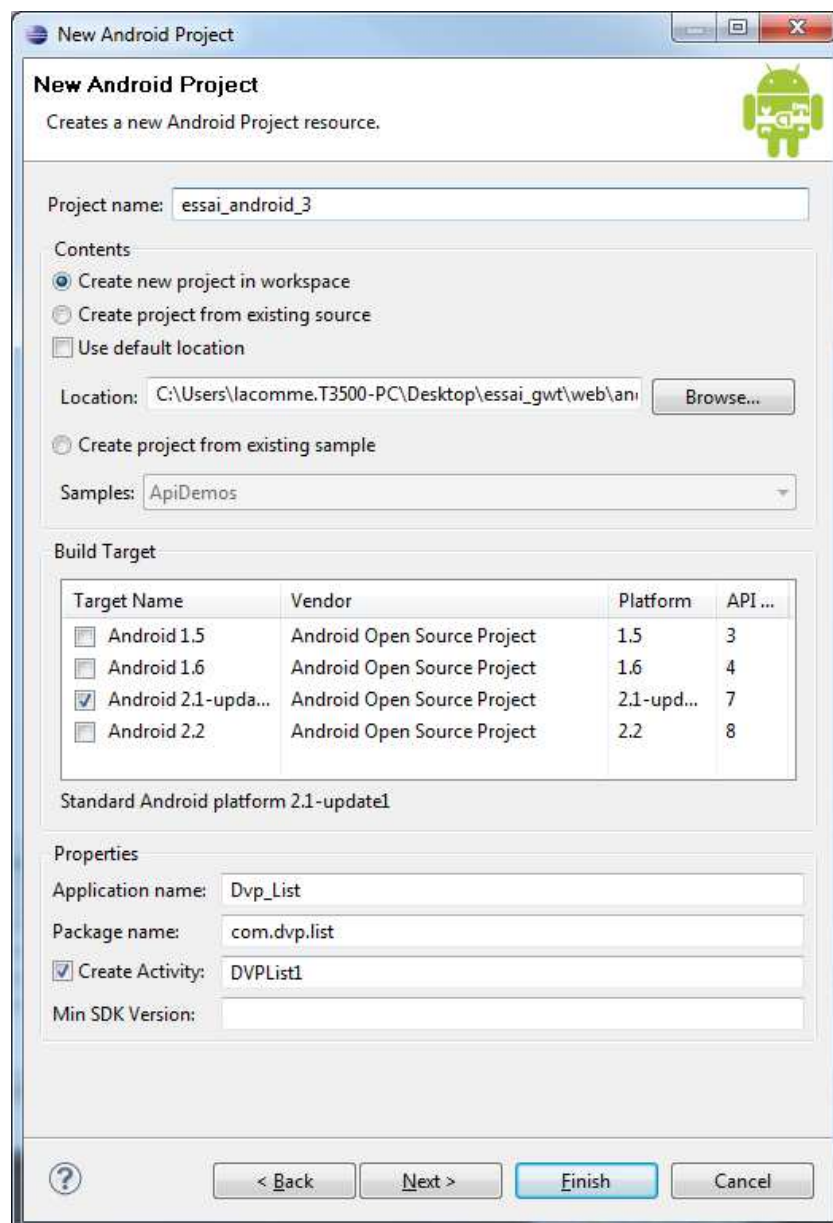
Nous allons reprendre un exemple similaire au précédent.

Etape 1. Créer un nouveau projet nommé **essai_android_3**.

Application name : Dvp_List

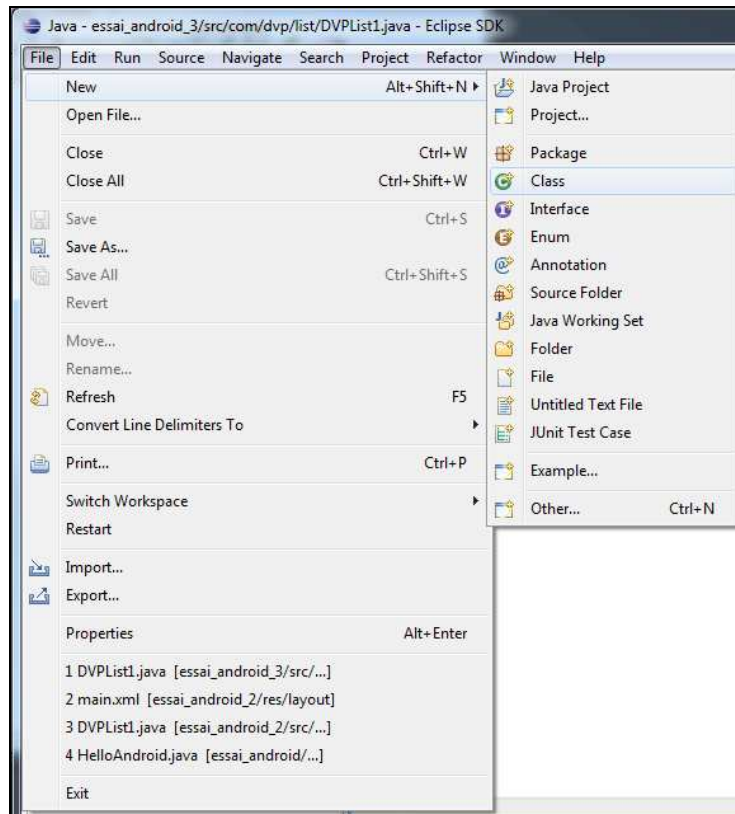
Package name : com.dvp.list

Create Activity : DVPList1

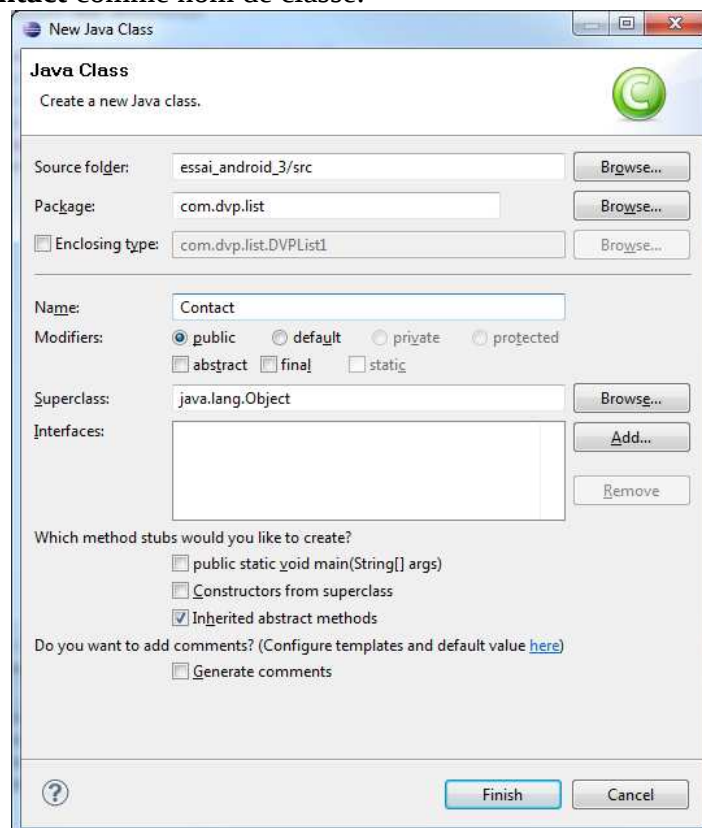


Etape 2. Créer une classe nommée **Contact.java**

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 aficionados 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", "Pecher", "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>
```

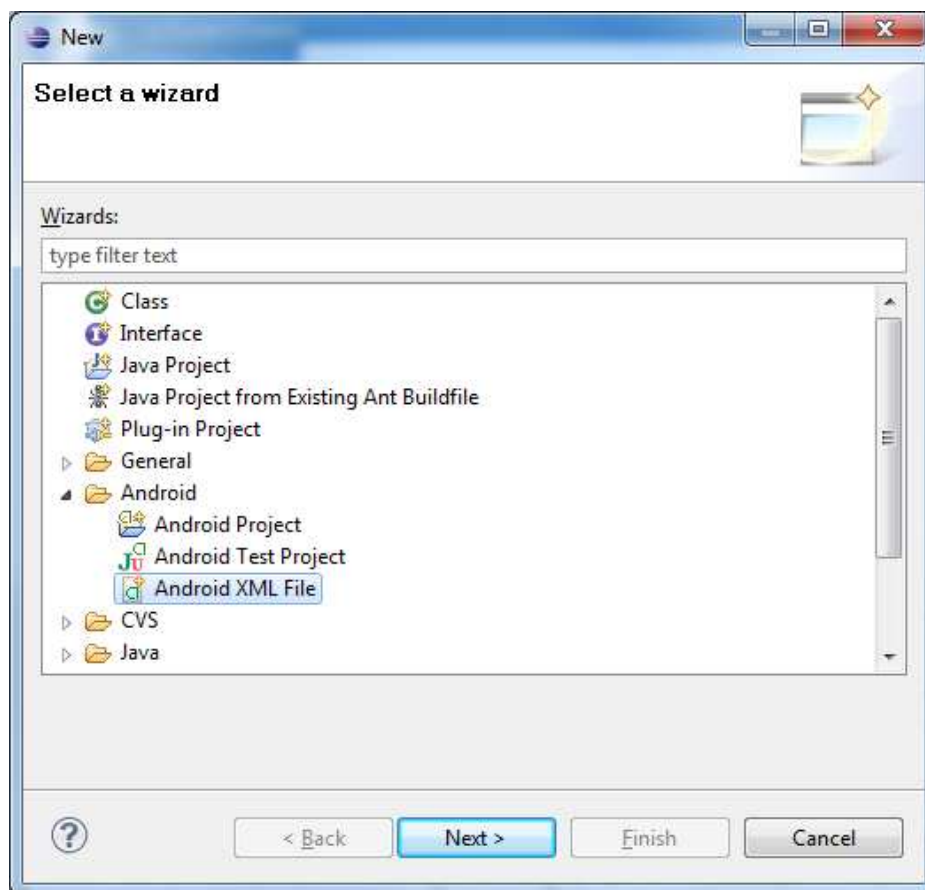
Il doit être modifié comme suit :

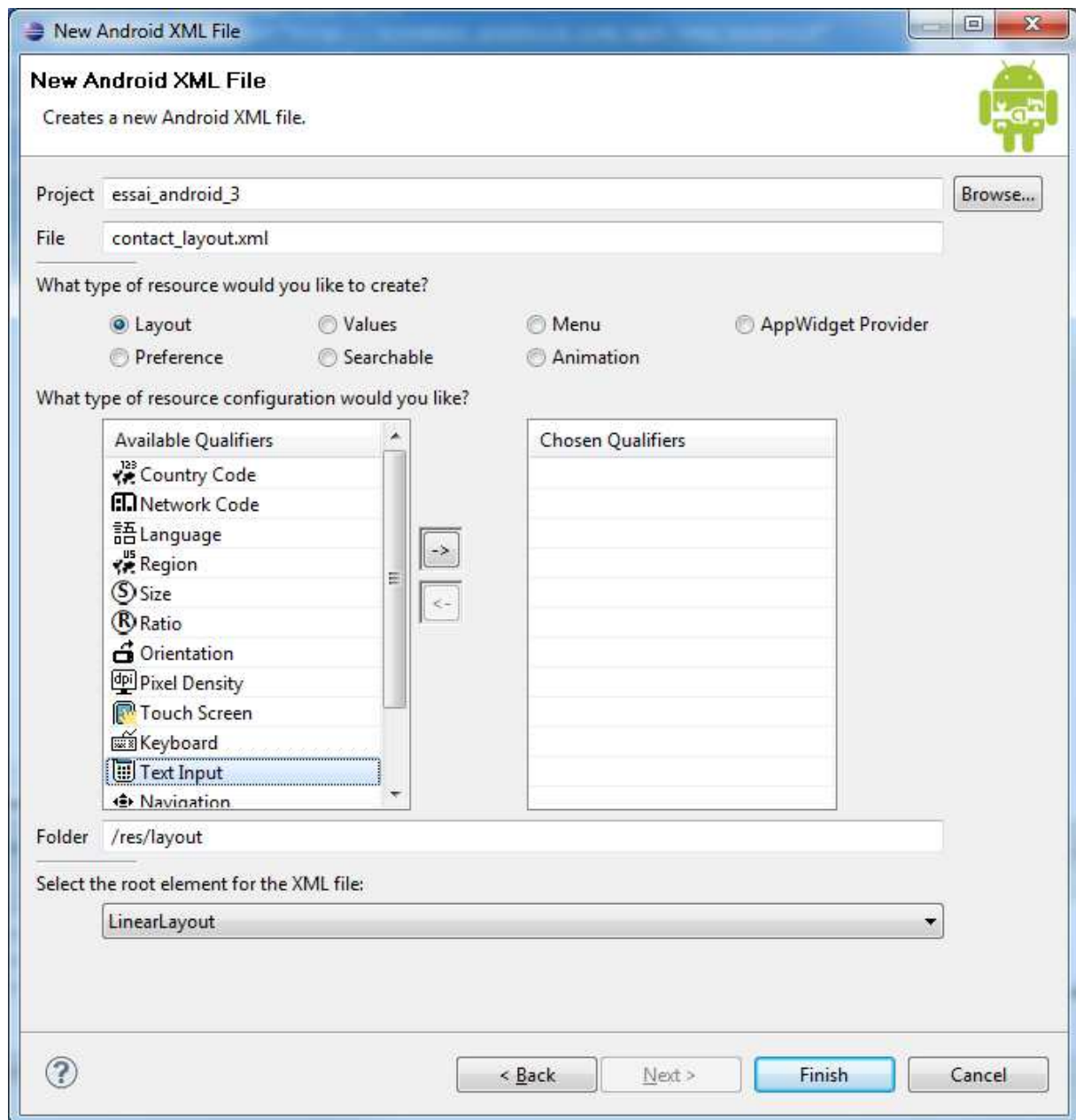
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">

    <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>

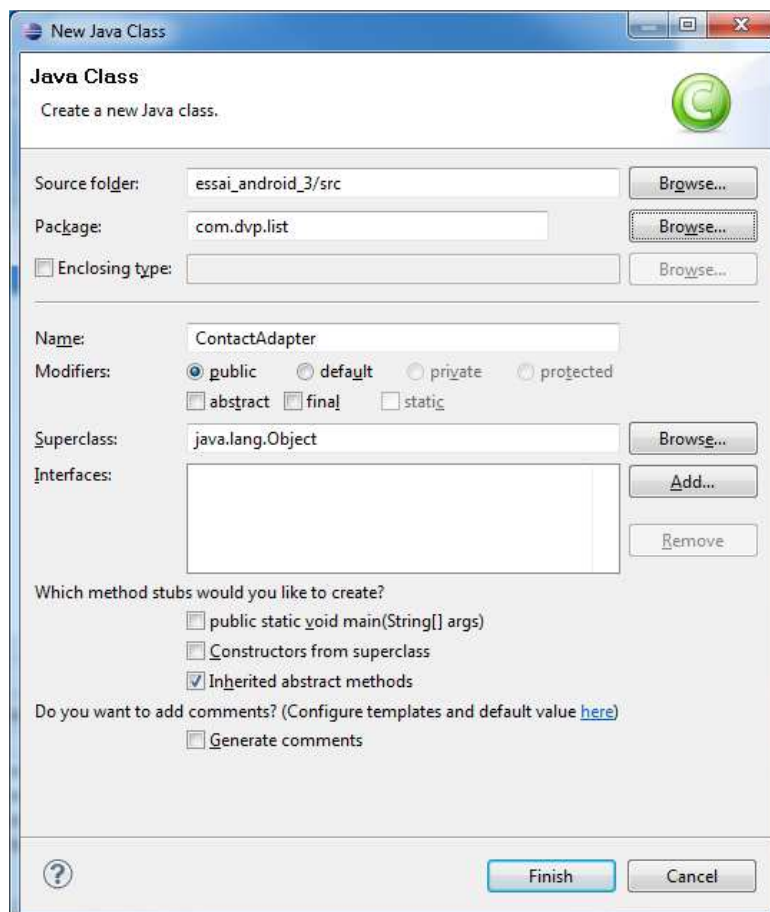
    <TextView android:text="Prénom" android:id="@+id/TV_Prenom"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">
    </TextView>

    <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 le 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);
    }
}
```

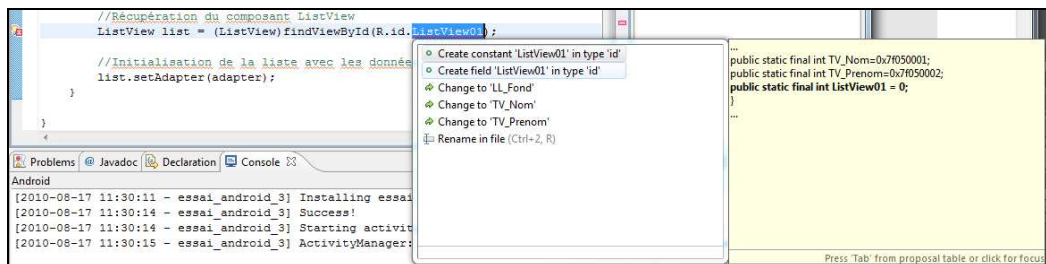


La ligne :

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

Est responsable 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
 * 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;
    }
}
```

Dernière mise à
jour du fichier

Etape 7. Tester le programme.

L'exécution doit donner ceci :



Etape 8. Gérer les événements 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 le transcoder
        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 événements 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".
        sendListner(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
    tv_Nom.setOnClickListener(ClickListener); ← modifications

    //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) {
        Builder builder = new AlertDialog.Builder(this);
        builder.setTitle("Contact Information");
        builder.setMessage("Le telephone est : " + item.telephone);
        builder.setPositiveButton("OK", null);
        builder.show();
    }
}

```

ajout

```

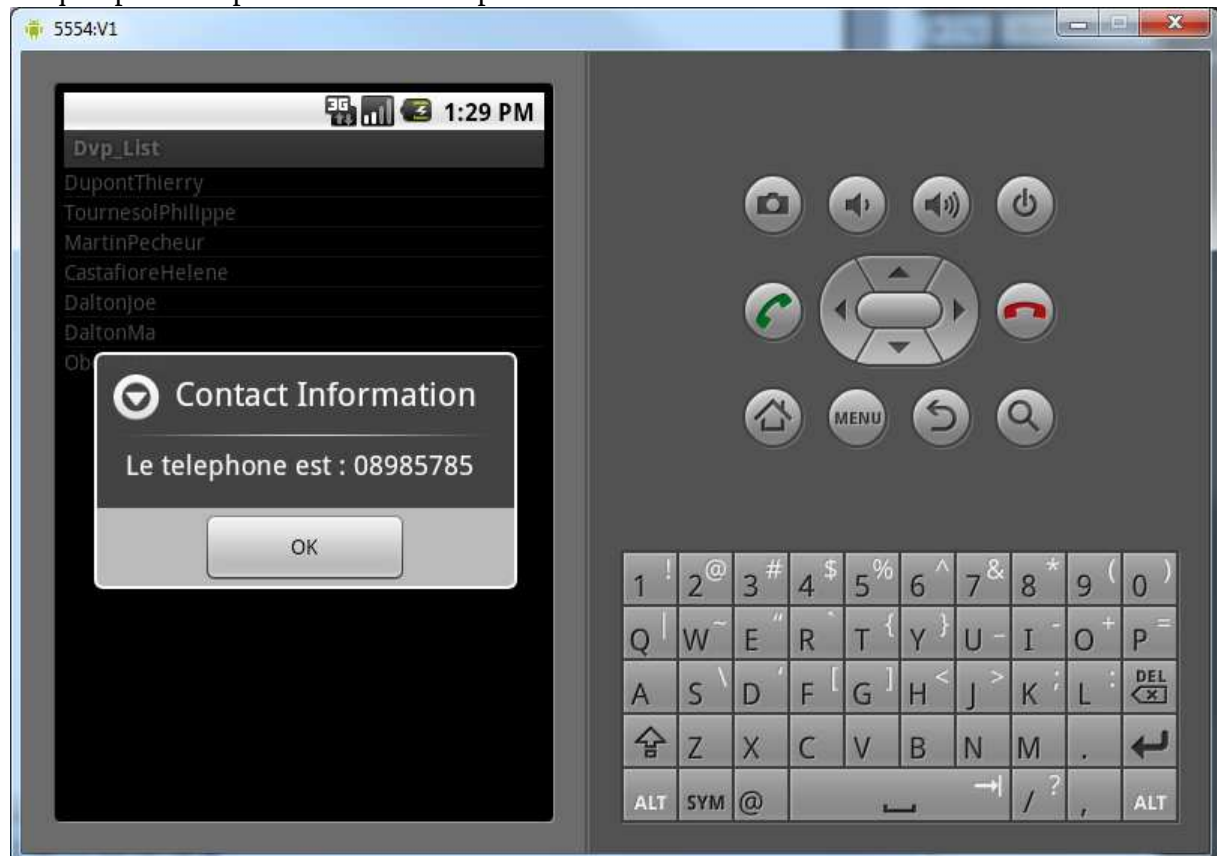
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

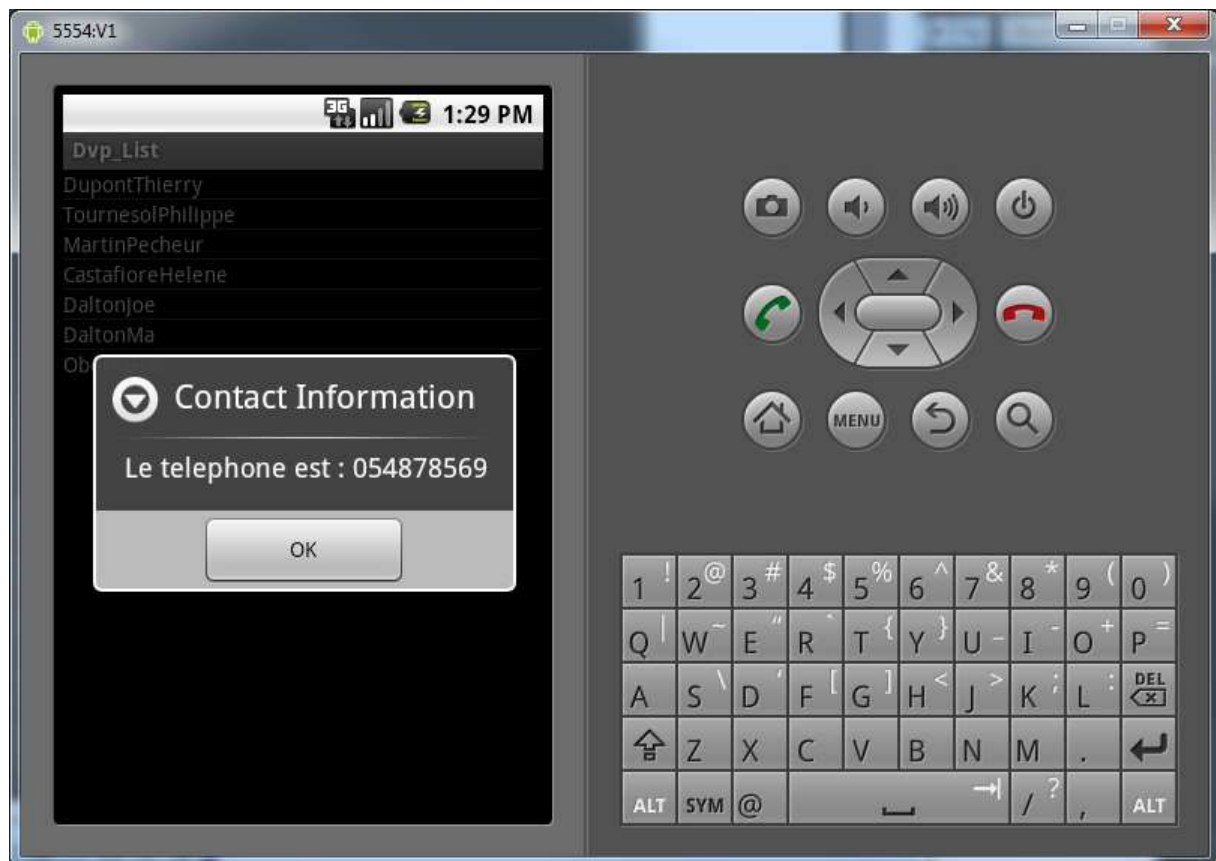
    //Récupération de la liste des personnes
    ArrayList<Contact> listP = Contact.Initialiser();
    //Création et initialisation de l'Adapter pour les contact
    ContactAdapter adapter = new ContactAdapter(this, listP);
    adapter.addListener(this); ← modifications
    //Récupération du composant ListView
    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 ».

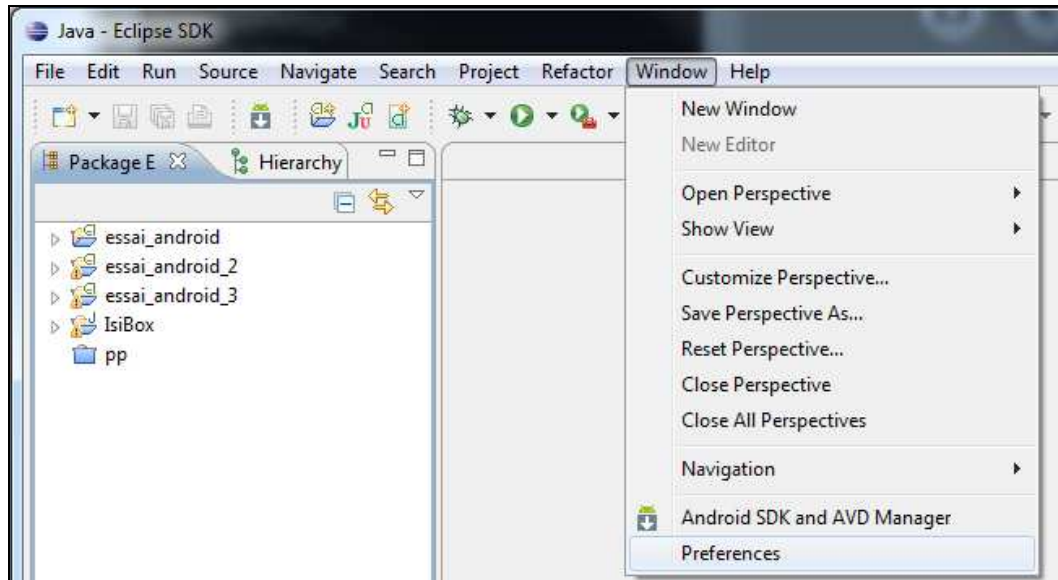




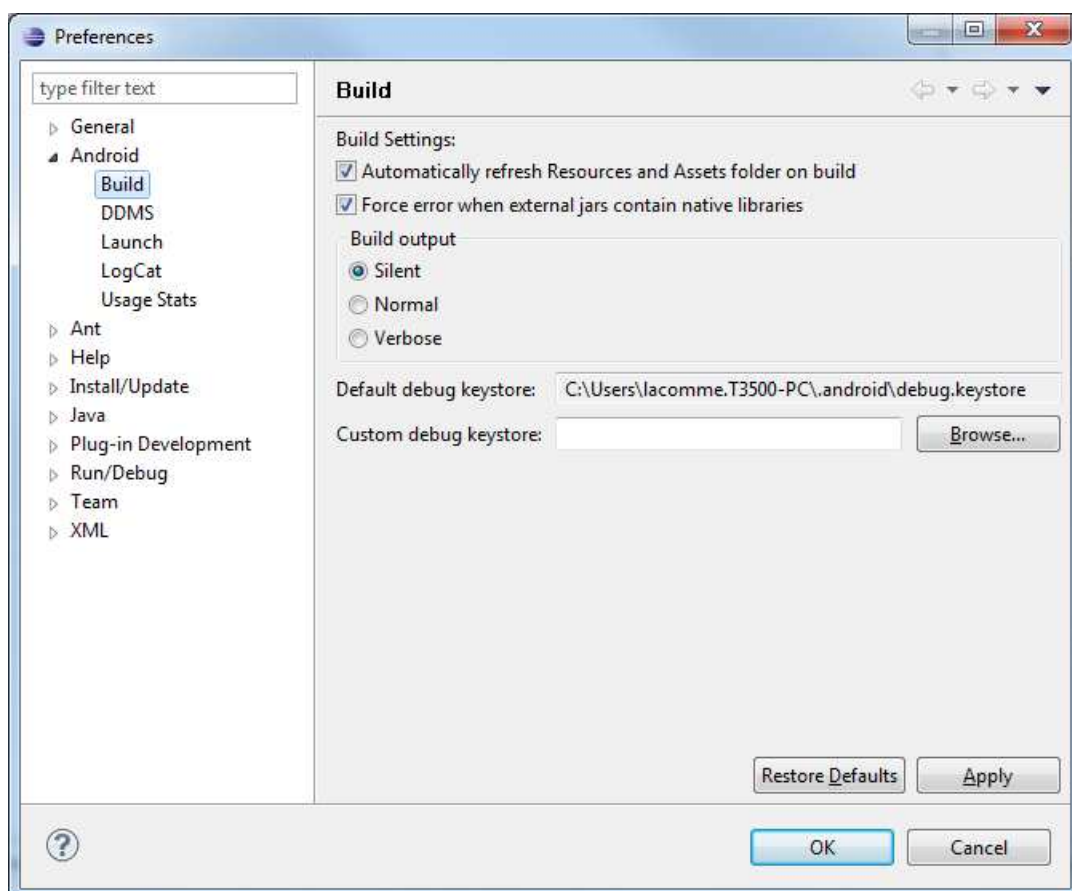
6. Intégrer Google Map dans Android

Etape 1. Obtenir le md5 checksum.

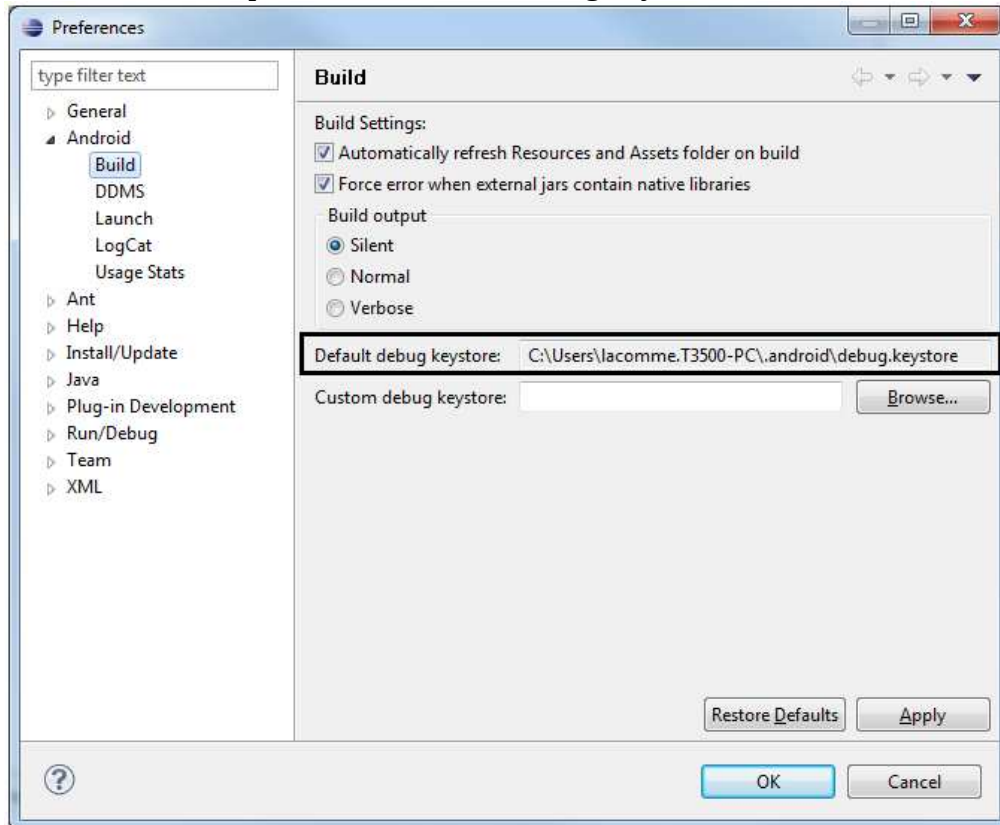
Aller dans **Window / Preferences.**



Choisir ensuite **Android / Build.**

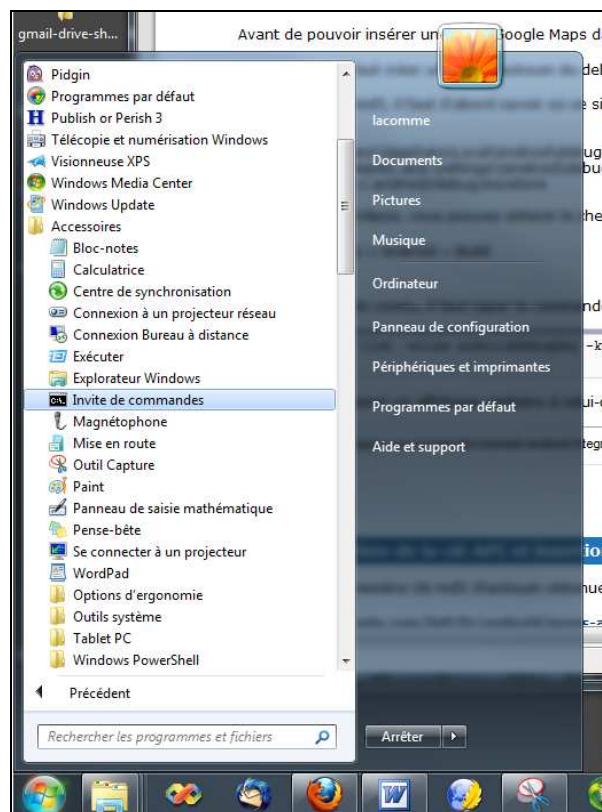


Relever le chemin d'accès par défaut à **Default debug keystore**.

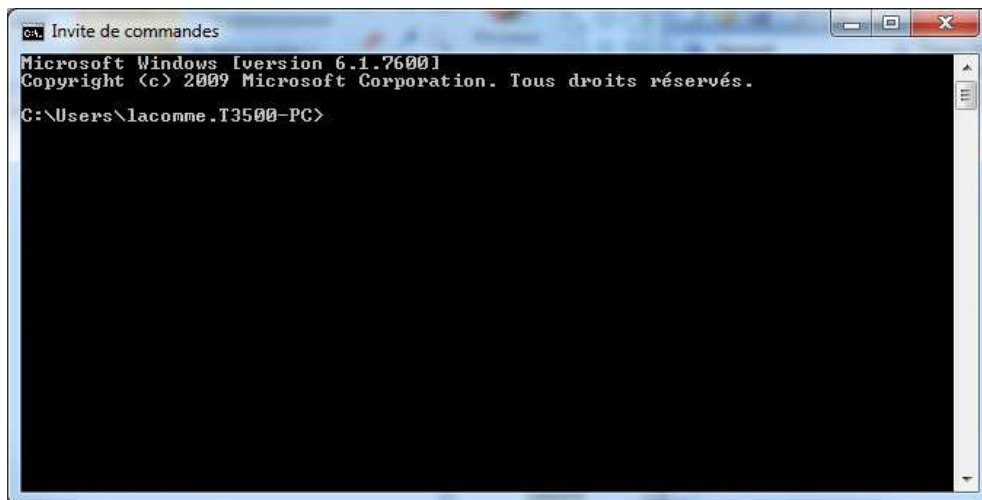


Dans mon cas particulier le chemin est : **C:\Users\Iacomme.T3500-PC\android**

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



Ce qui donne :



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

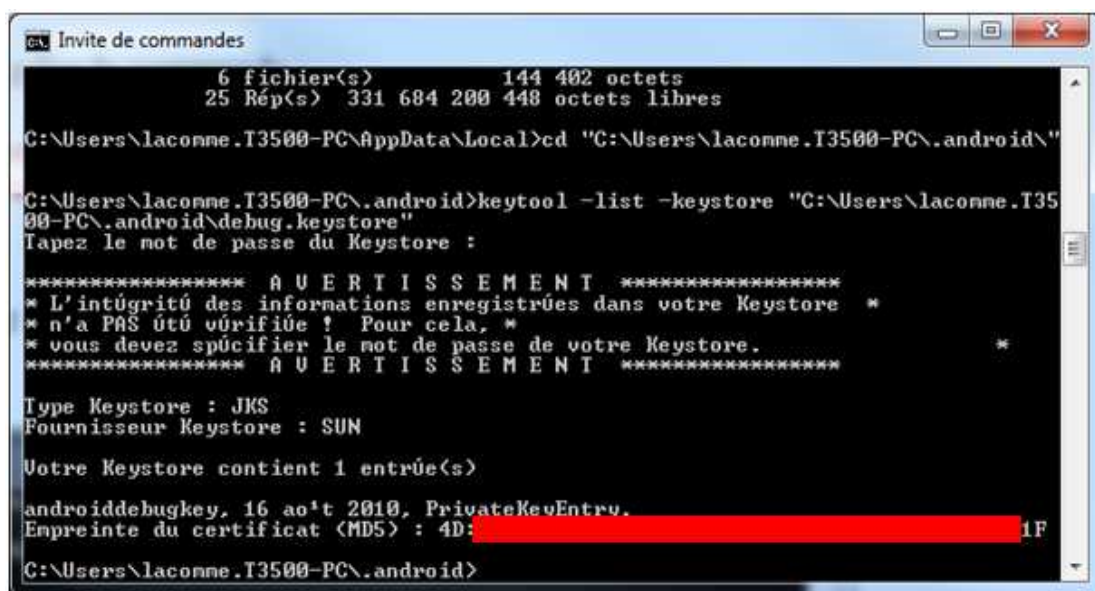
keytool -list -keystore <chemin_vers_le_fichier_debug>.keystore

La commande

keytool -list - keystore

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

donne sur ma machine :





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.

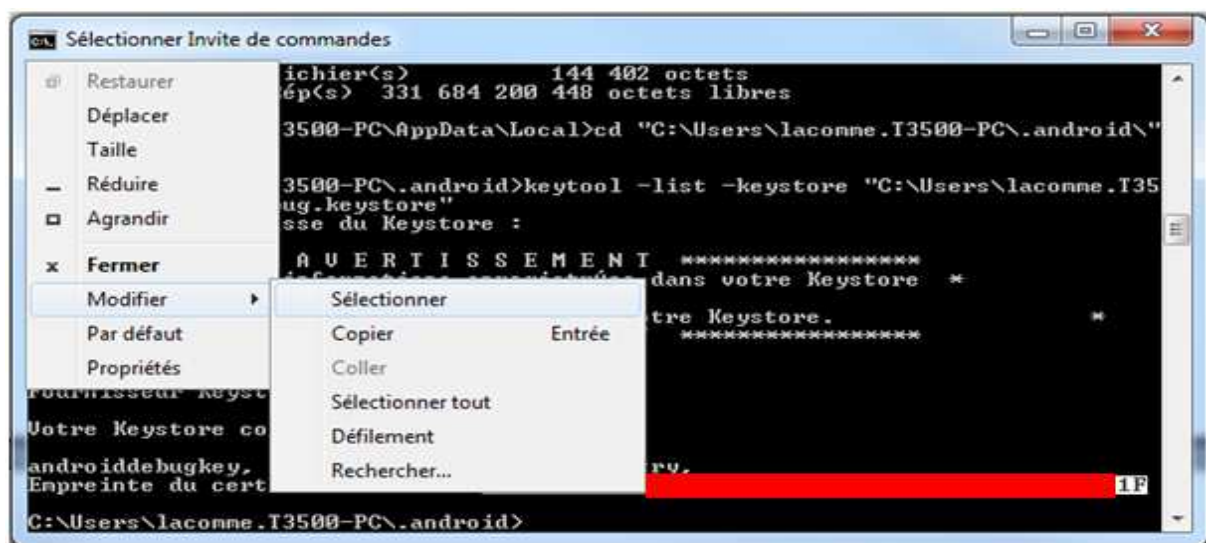
Si vous avez réalisé une installation standard des différents éléments (SDK etc...), il est probable que le JDK soit dans le répertoire Sun sous la racine.

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 :

```
6 fichier(s)          144 402 octets
25 Rép(s)  331 684 200 448 octets libres

C:\Users\lacomme.T3500-PC\AppData\Local>cd "C:\Users\lacomme.T3500-PC\.android\"

C:\Users\lacomme.T3500-PC\.android>keytool -list -keystore "C:\Users\lacomme.T3500-PC\.android\debug.keystore"
Tapez le mot de passe du Keystore :

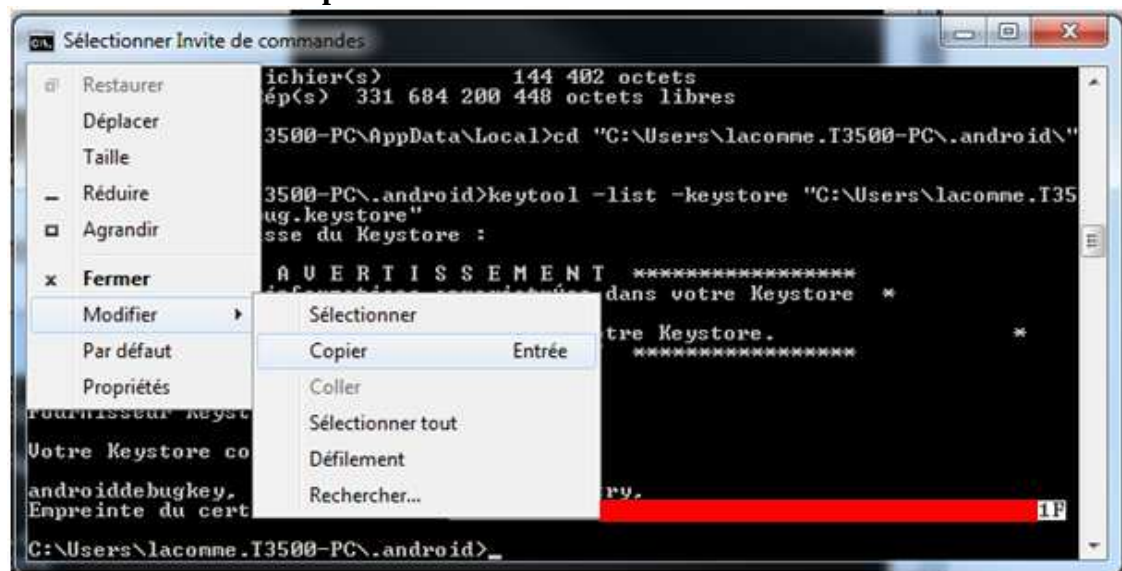
*****  A U E R T I S S E M E N T  *****
* L'intégrité des informations enregistrées dans votre Keystore *
* n'a PAS été vérifiée ! Pour cela, *
* vous devez spécifier le mot de passe de votre Keystore.      *
*****  A U E R T I S S E M E N T  *****

Type Keystore : JKS
Fournisseur Keystore : SUN

Votre Keystore contient 1 entrée(s)

androiddebugkey, 16 août 2010, PrivateKeyEntry,
Empreinte du certificat (MD5) : 4D: [redacted] 1F
C:\Users\lacomme.T3500-PC\.android>
```

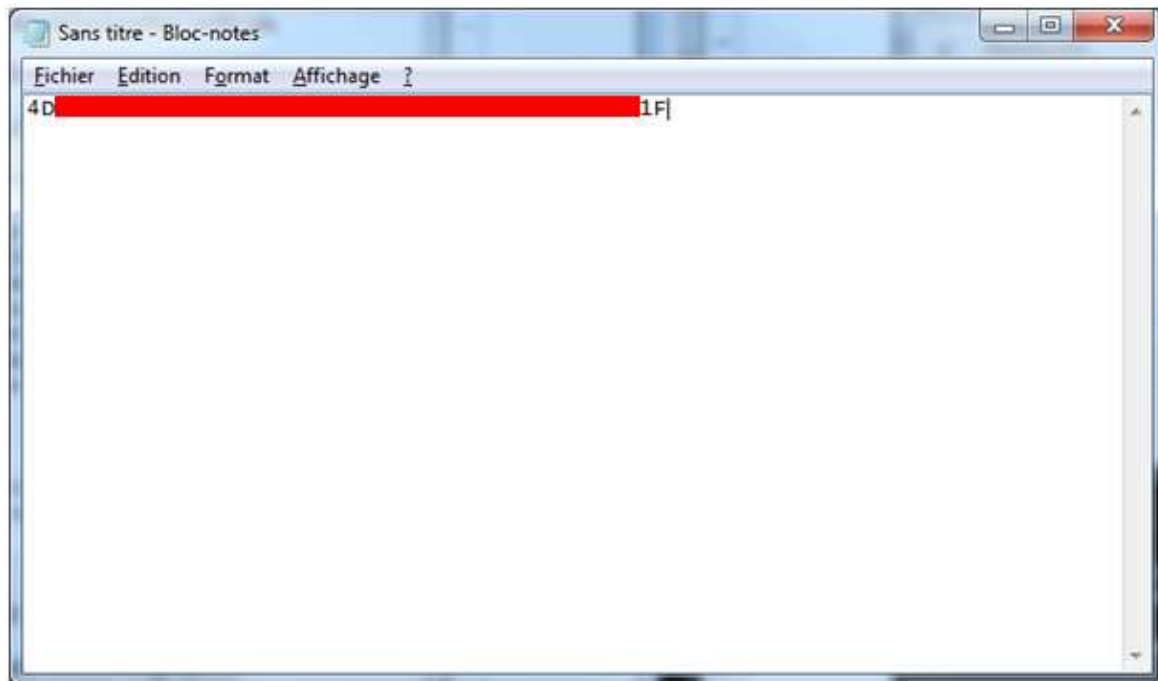
Faire ensuite **Modifier / Copier**.



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

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

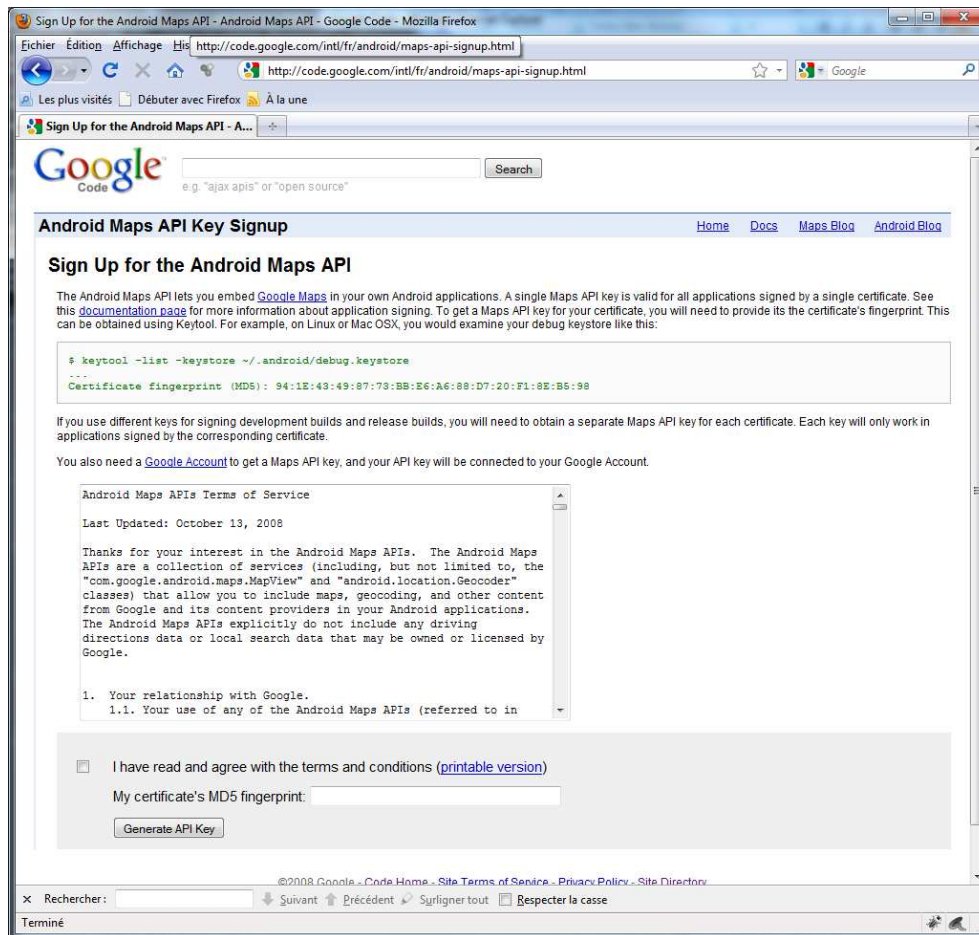
Vous devriez obtenir un document comme celui-ci :



Etape 2. Obtenir une clé Google.

Connectez vous à l'adresse suivante :

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

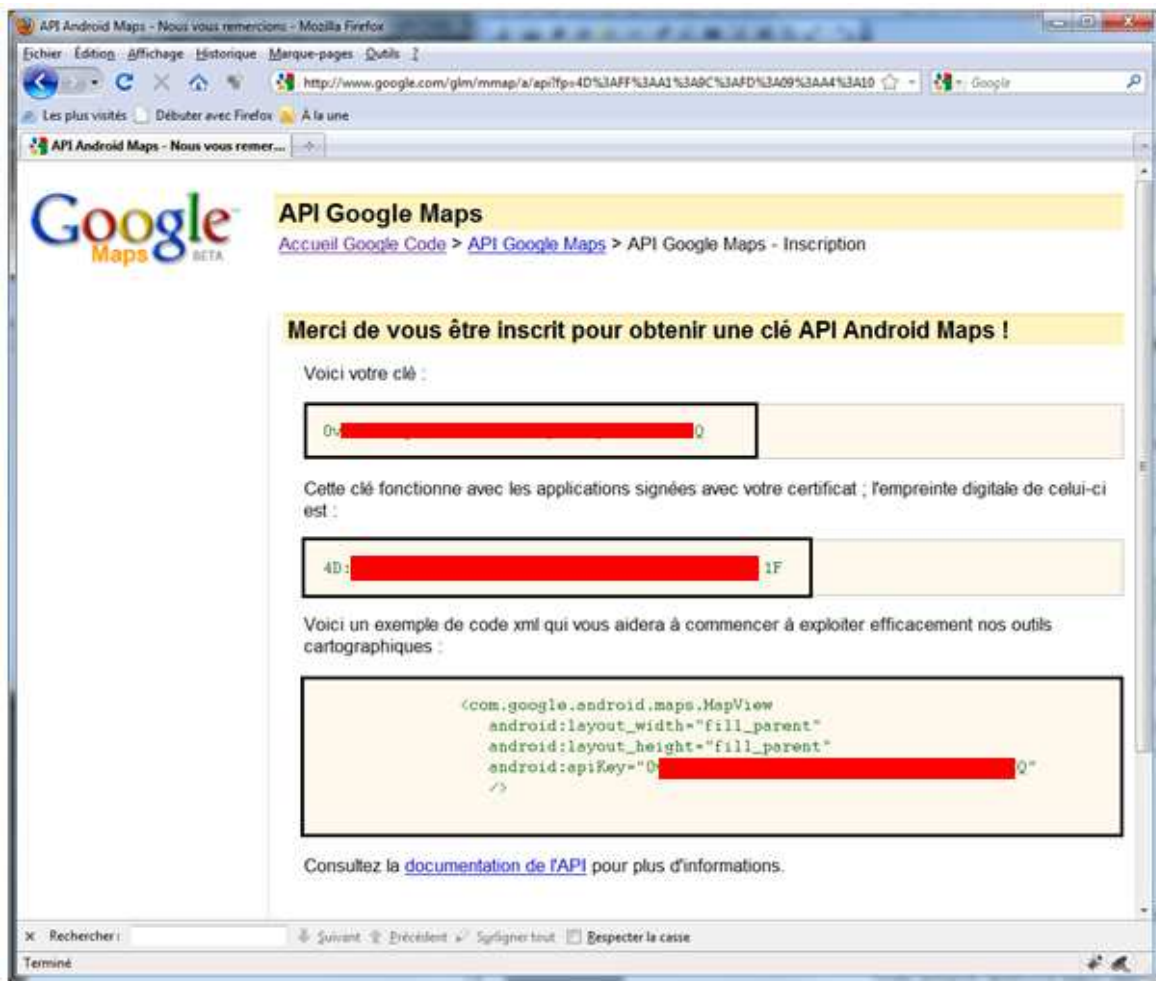


Faites CTRL-V dans le champ My certificate's MD5 fingerprint :

☒ I have read and agree with the terms and conditions ([printable version](#))

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 :

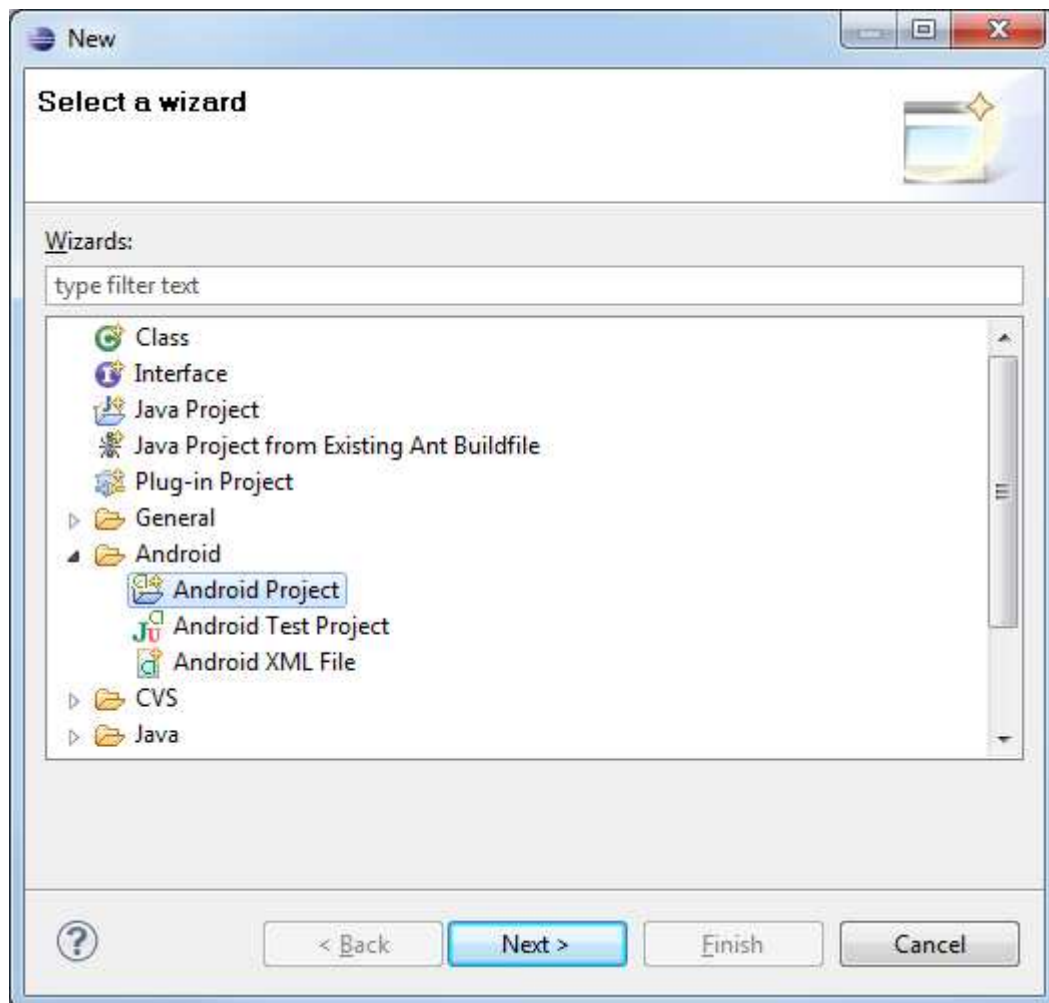
Clé : 0*****Q
MD5 : 4D*****1F

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 Android.

Faire New / Project et choisir Android Project.



New Android Project

New Android Project

Creates a new Android Project resource.



Project name:

Contents

☒ Create new project in workspace

☐ Create project from existing source

☒ Use default location

Location:

☐ Create project from existing sample

Samples:

Build Target

Target Name	Vendor	Platform	API ...
☐ Android 1.5	Android Open Source Project	1.5	3
☐ Android 1.6	Android Open Source Project	1.6	4
☒ Android 2.1-upda...	Android Open Source Project	2.1-upd...	7
☐ Android 2.2	Android Open Source Project	2.2	8

Standard Android platform 2.1-update1

Properties

Application name:

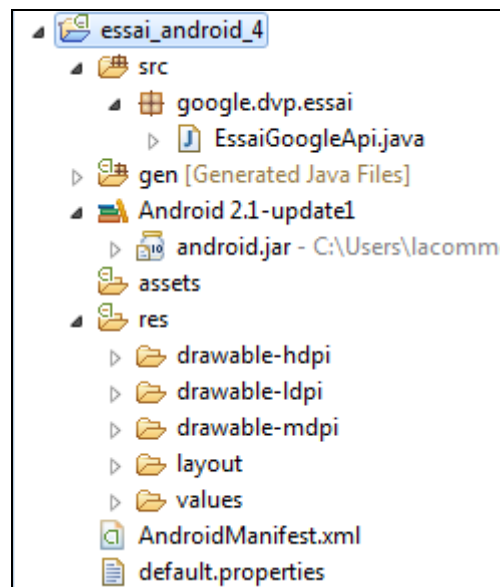
Package name:

☒ Create Activity:

Min SDK Version:

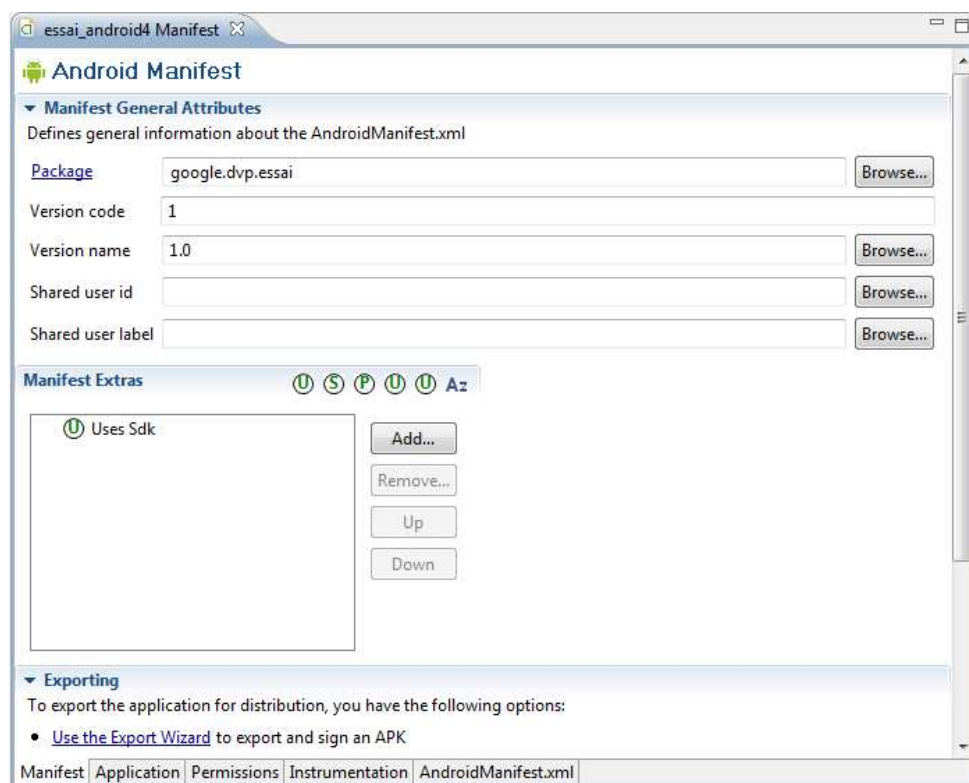


Le projet ainsi crée doit se présenter comme suit :

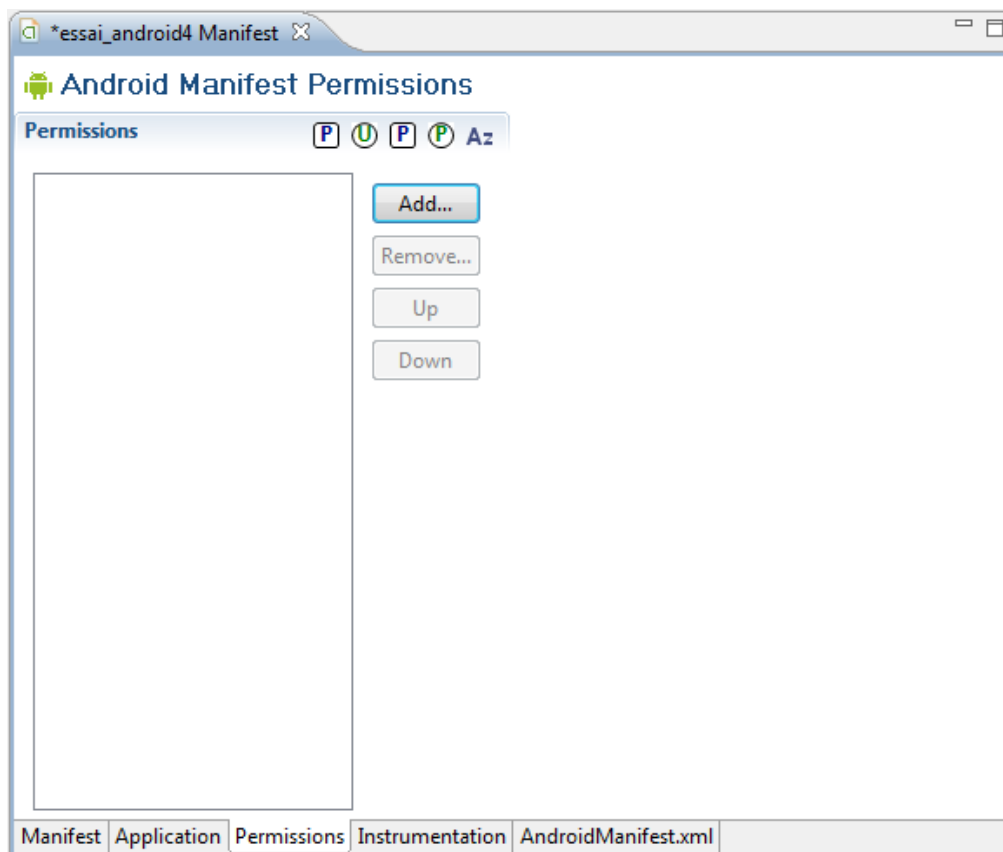


Etape 4. Modifier l'application Android pour utiliser les services google

Ouvrir le fichier **AndroidManifest.xml**

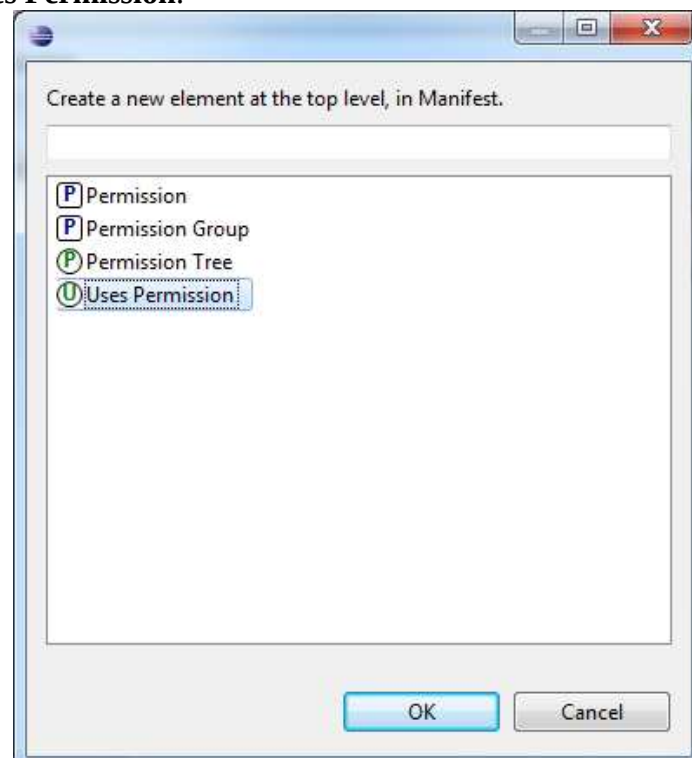


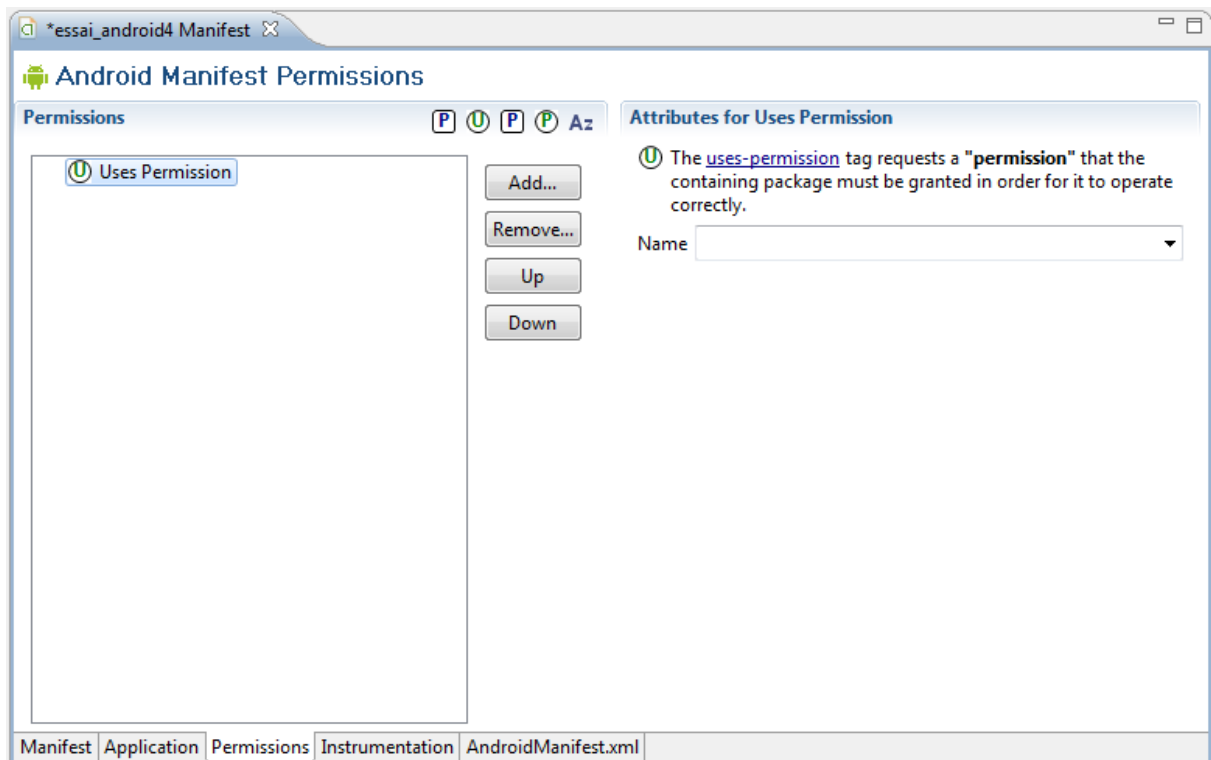
Aller dans l'onglet **Permissions**.



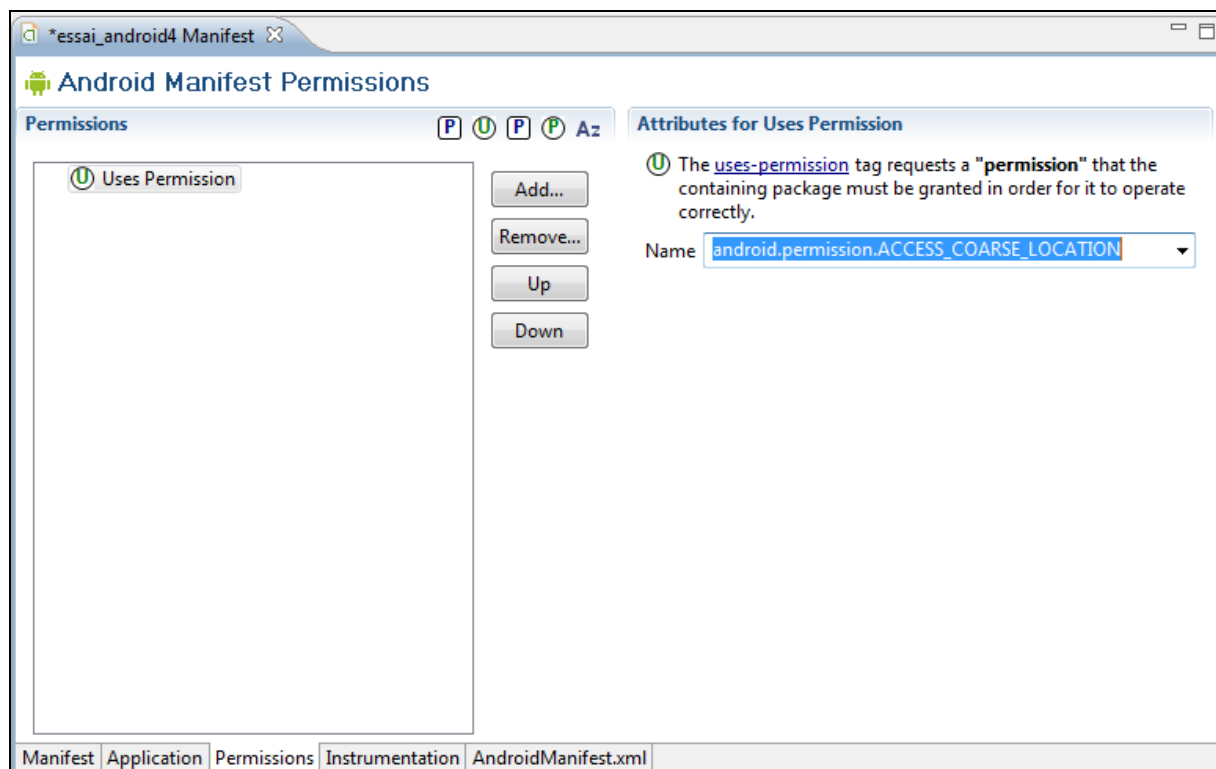
Faire ensuite **Add**.

Choisir ensuite **Uses Permission**.





Dans le champ Name choisir :
android.permission.ACCESS_COARSE_LOCATION



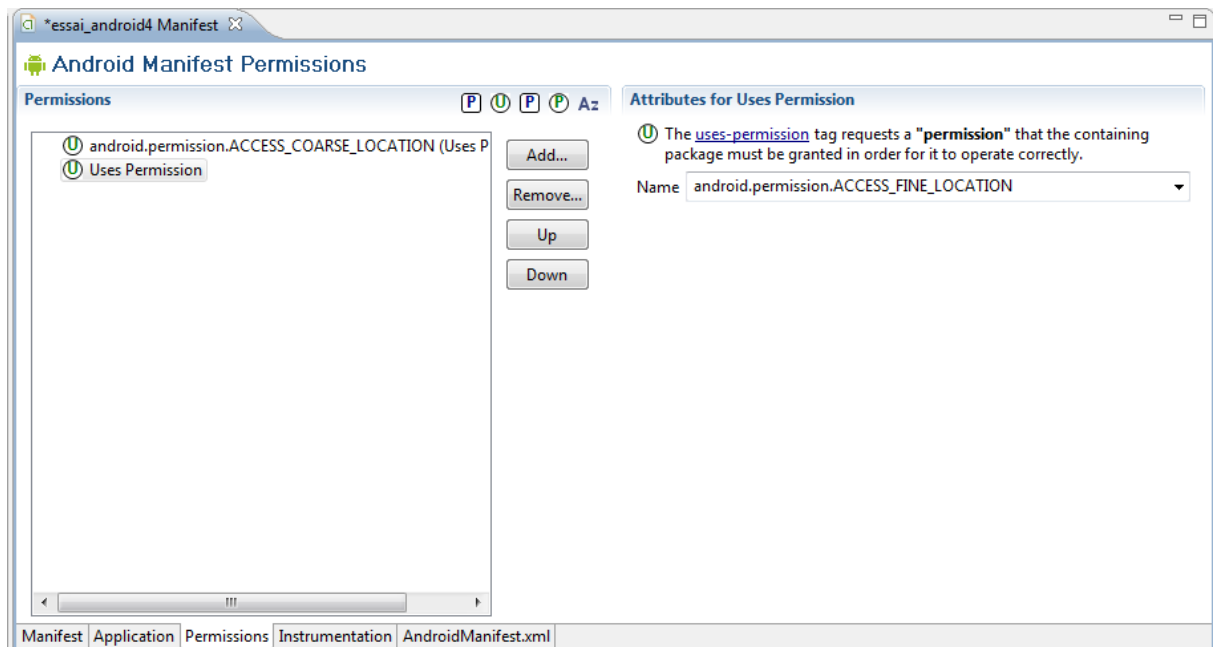
Passez ensuite sur l'onglet AndroidManifest.xml et vérifiez que vous obtenez un fichier similaire à celui-ci :



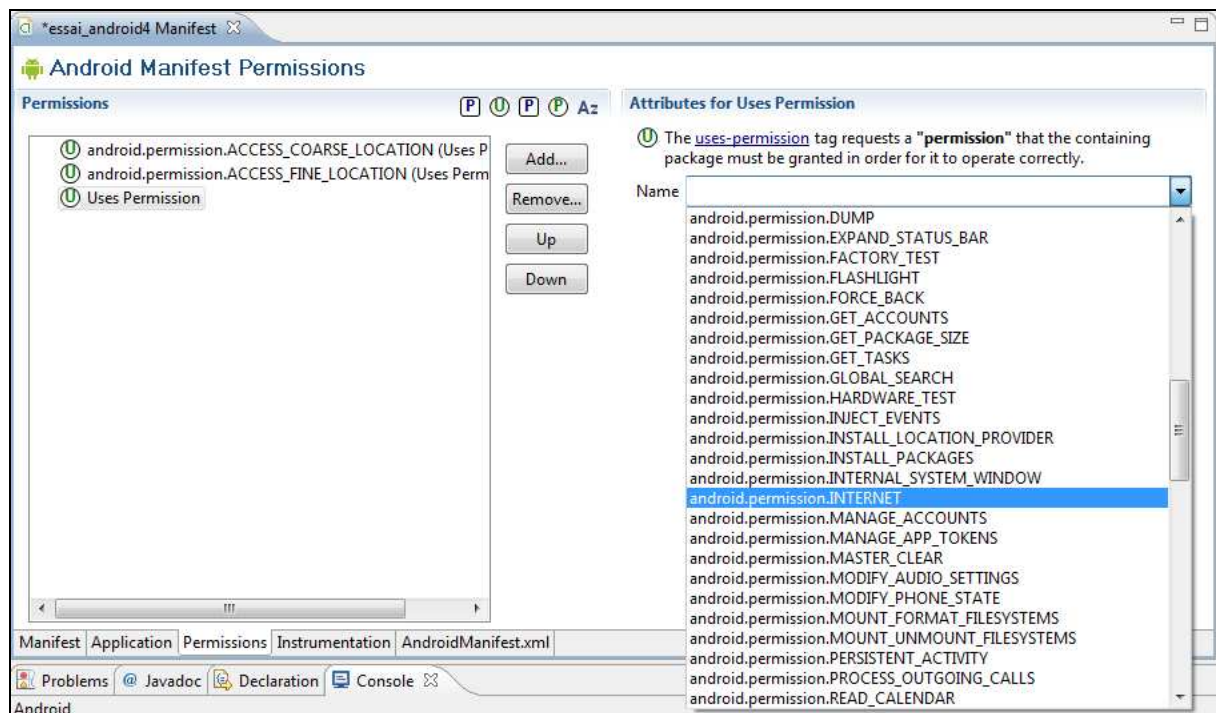
```
1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3     package="google.dvp.essai"
4     android:versionCode="1"
5     android:versionName="1.0">
6     <application android:icon="@drawable/icon" android:label="@string/app_name">
7         <activity android:name=".EssaiGoogleApi"
8             android:label="@string/app_name">
9             <intent-filter>
10                 <action android:name="android.intent.action.MAIN" />
11                 <category android:name="android.intent.category.LAUNCHER" />
12             </intent-filter>
13         </activity>
14     </application>
15
16
17
18
19
20 <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"></uses-permission>
21 </manifest>
```

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>

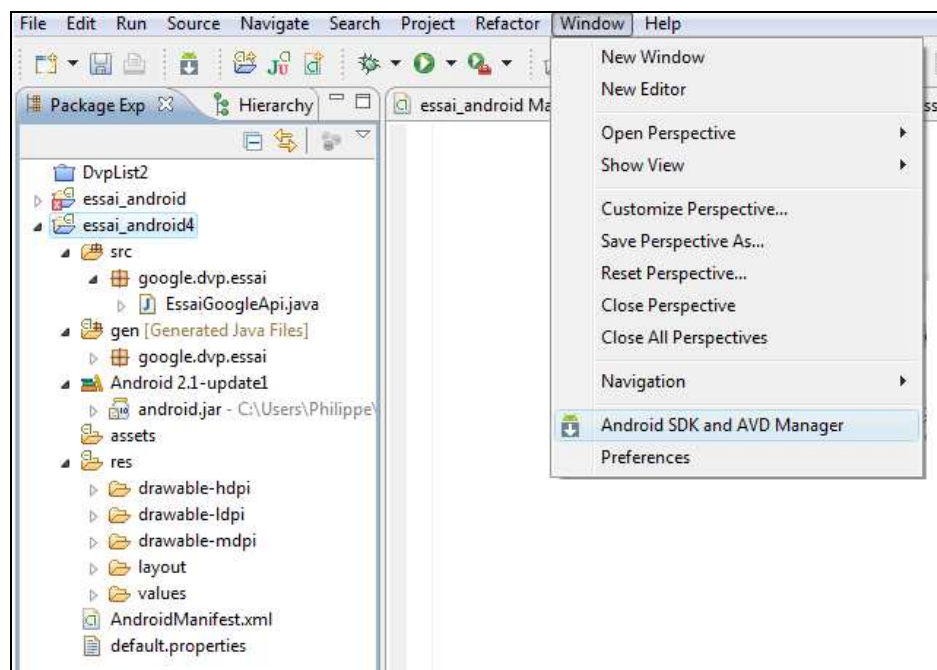
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"></uses-permission>
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"></uses-permission>
    <uses-permission android:name="android.permission.INTERNET"></uses-permission>
</manifest>
```

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

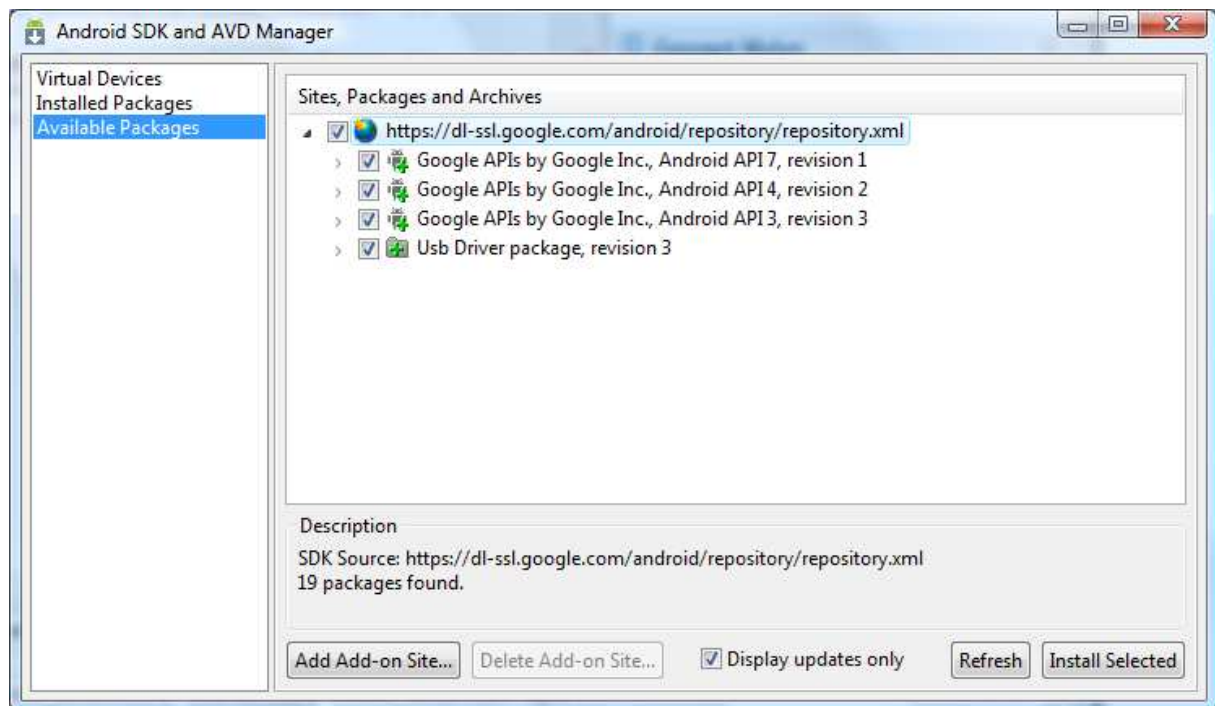


Etape 5. Ajouter l'API google

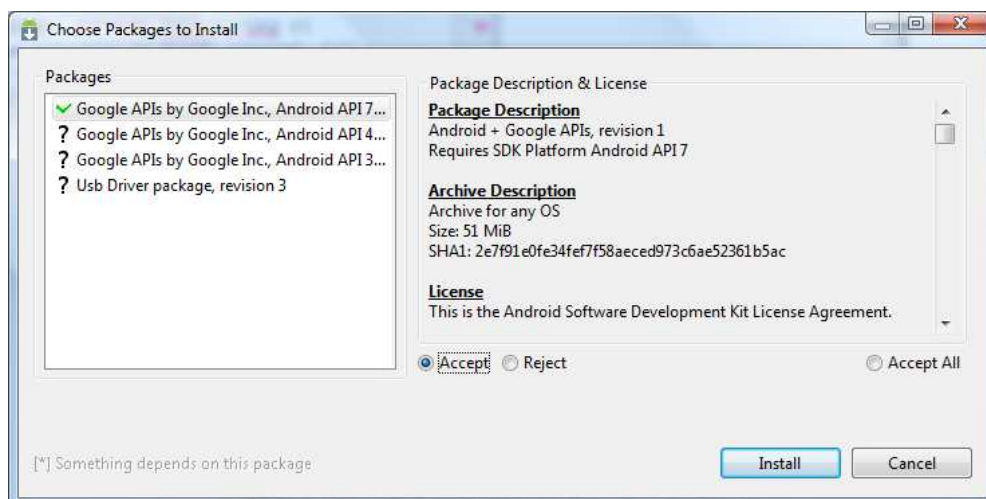
Aller dans **Window / Android SDK**



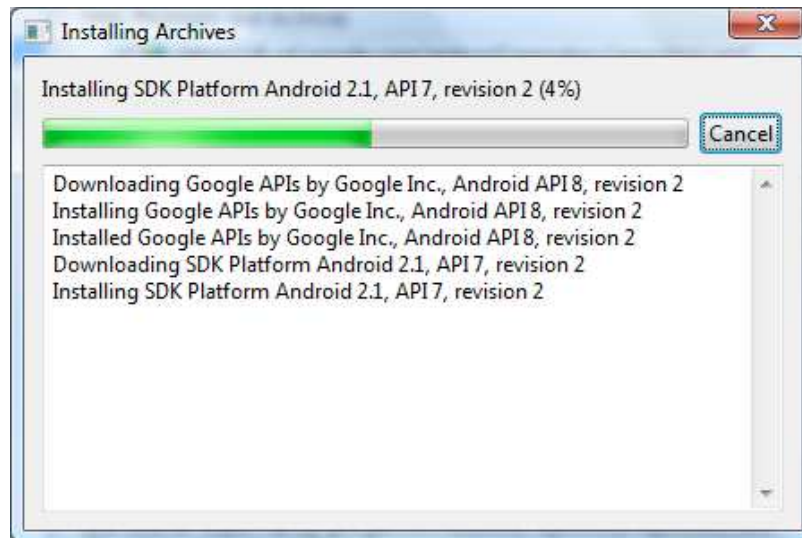
Aller dans **Available Packages**.



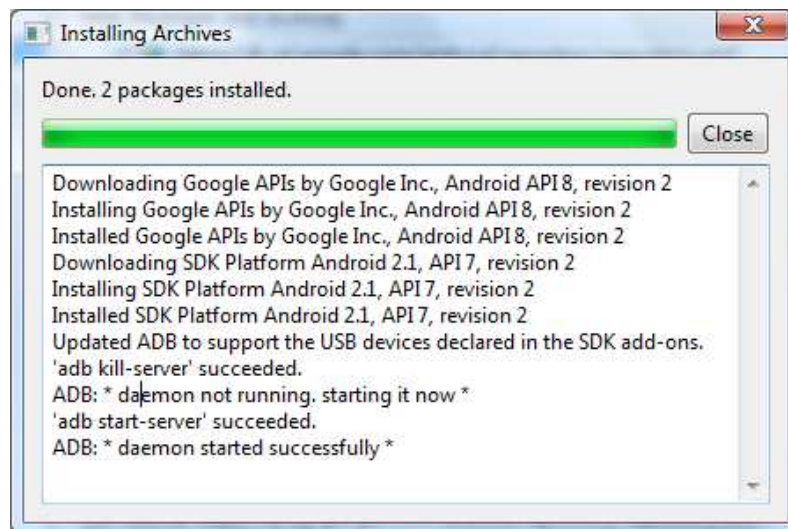
Choisir **Install Selected**.



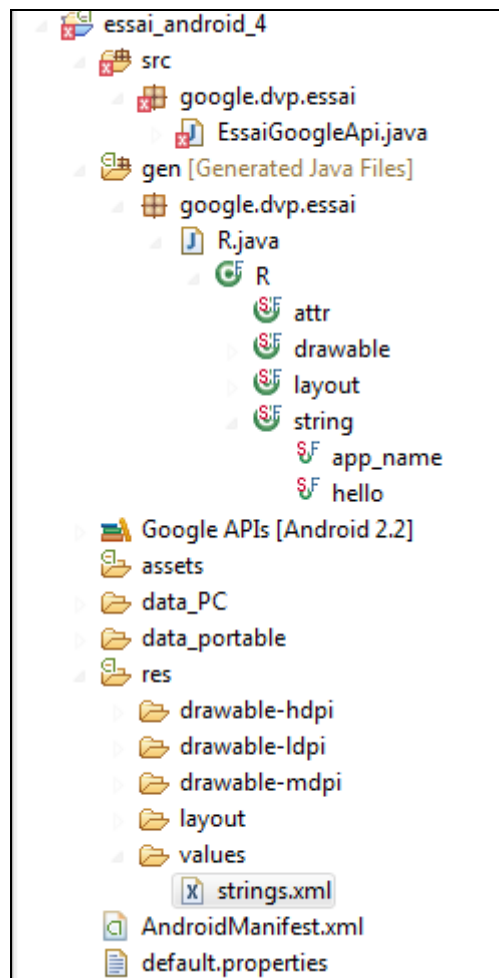
Accepter les conditions de la licence et faire **Install**.



Patientez jusqu'à la fin du processus.



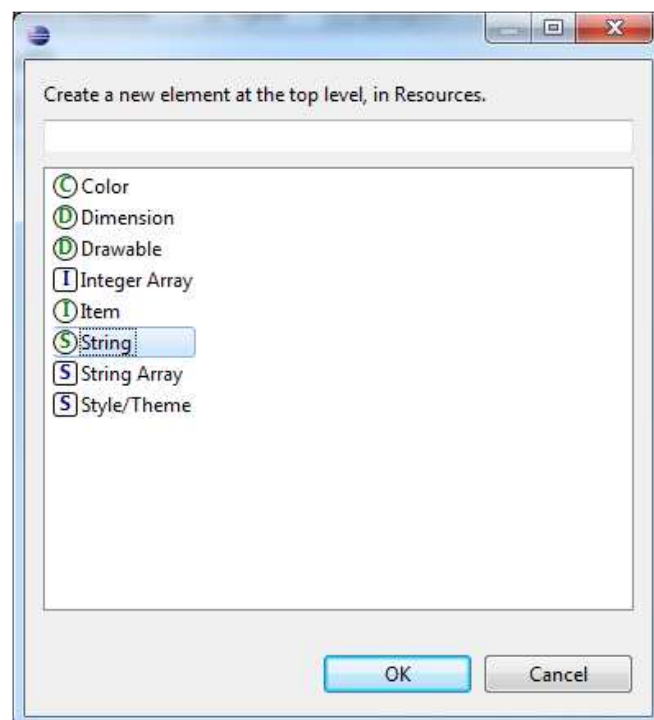
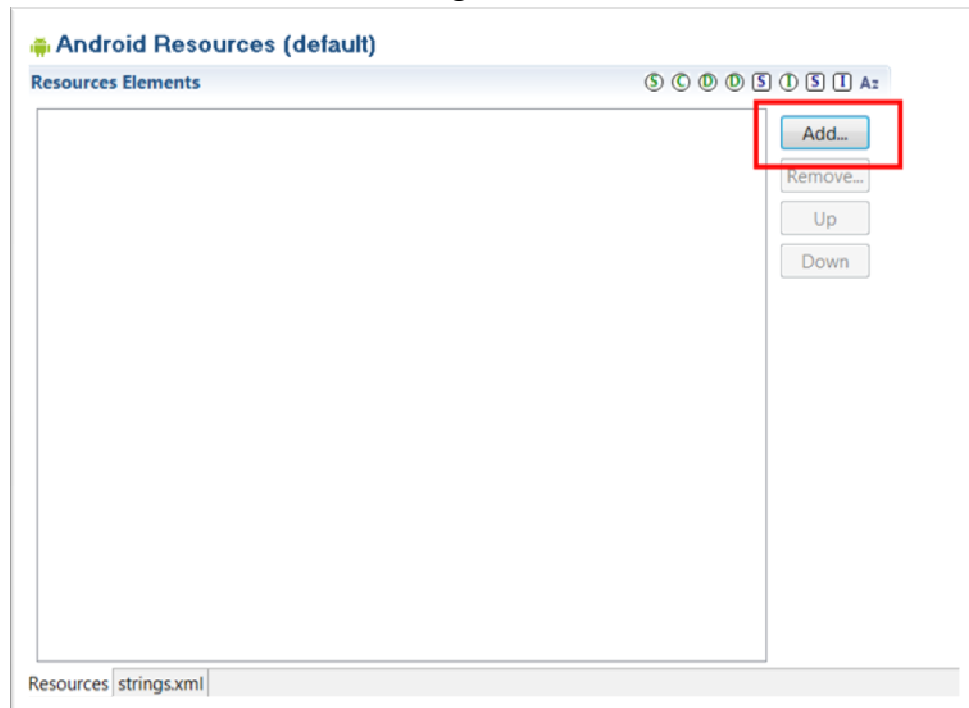
Etape 6. Ouvrir le fichier nommé **strings.xml**

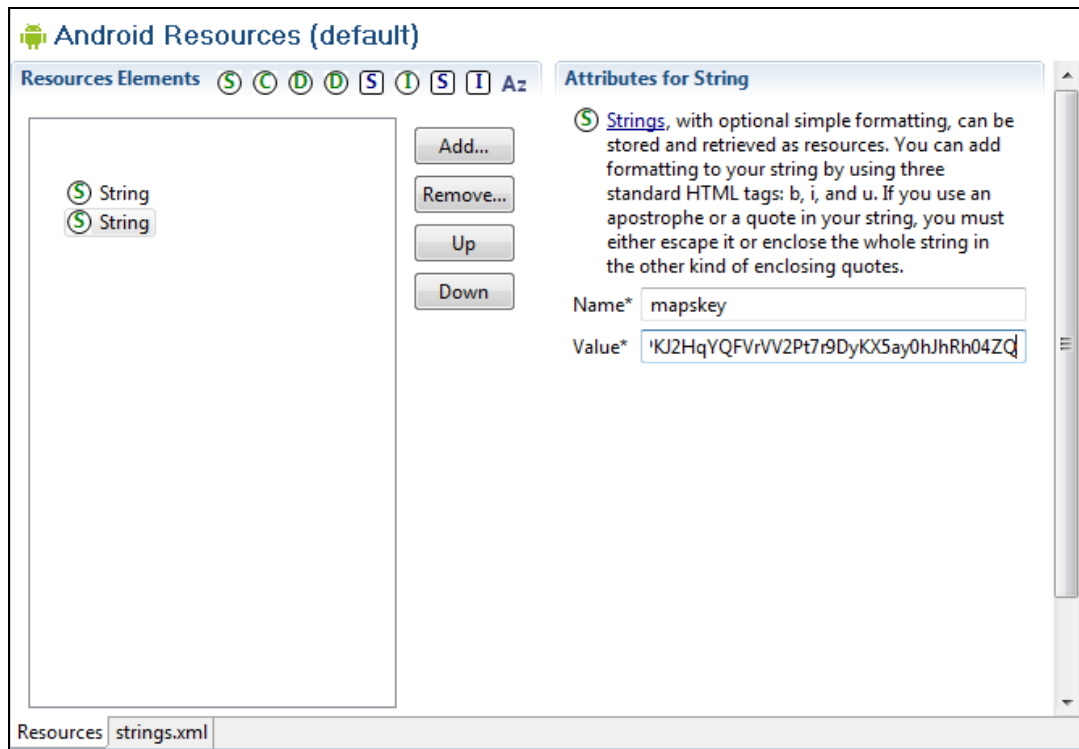


Le fichier initialement se présente comme suit.

```
<?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 **Ressources** 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"
    >

    <com.google.android.maps.MapView
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:apiKey="0vfPKJ2HqYQFVrVV2Pt7r9DyKX5ay0hJhRh04ZQ"
    />

</LinearLayout>
```

Correspond à
l'exemple de code
proposé par
Google

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">

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

        <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>

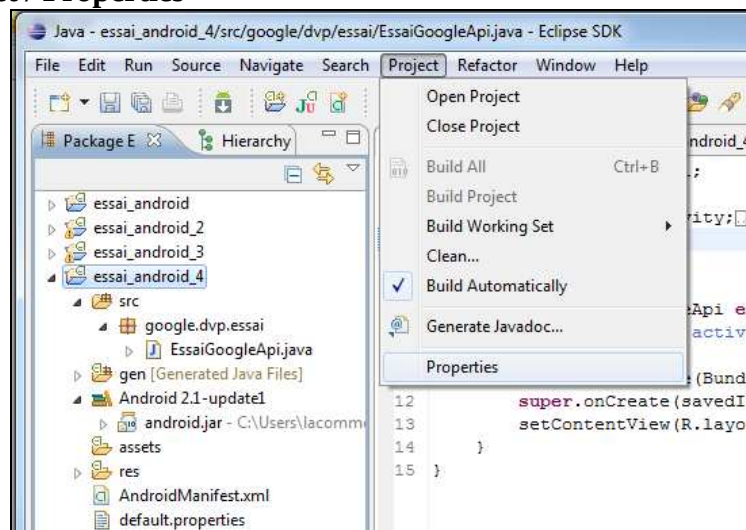
    </application>

    <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>

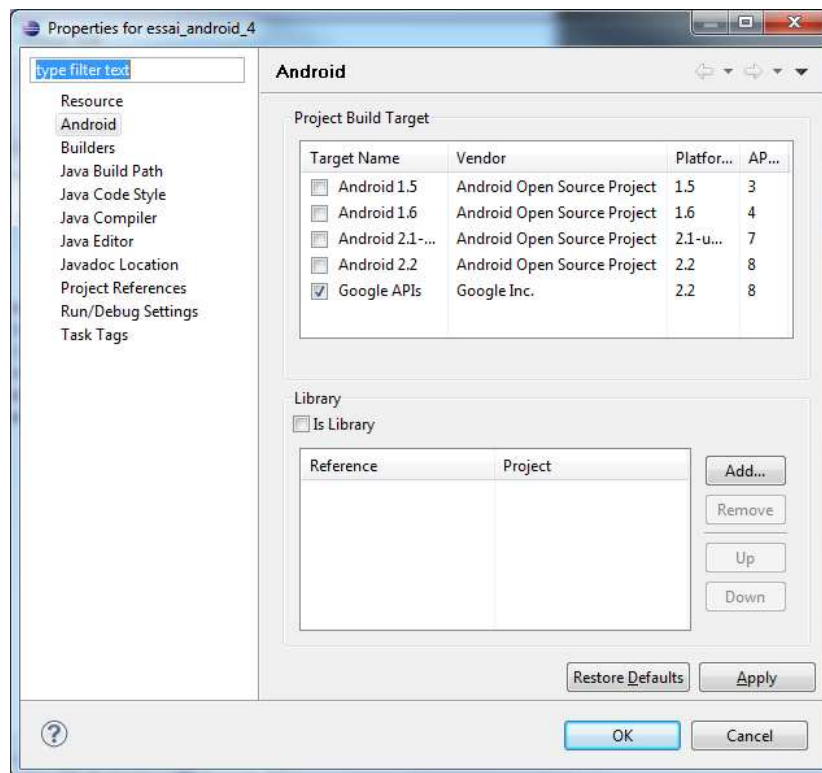
</manifest>
```

Etape 9. Modifier l'application principale

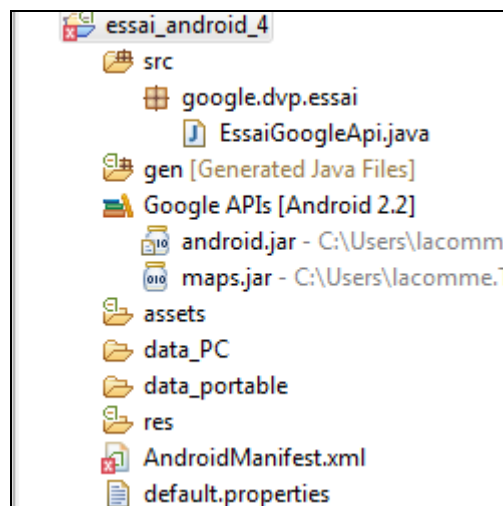
Aller dans **Project / Properties**



Aller dans la partie Android 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 {

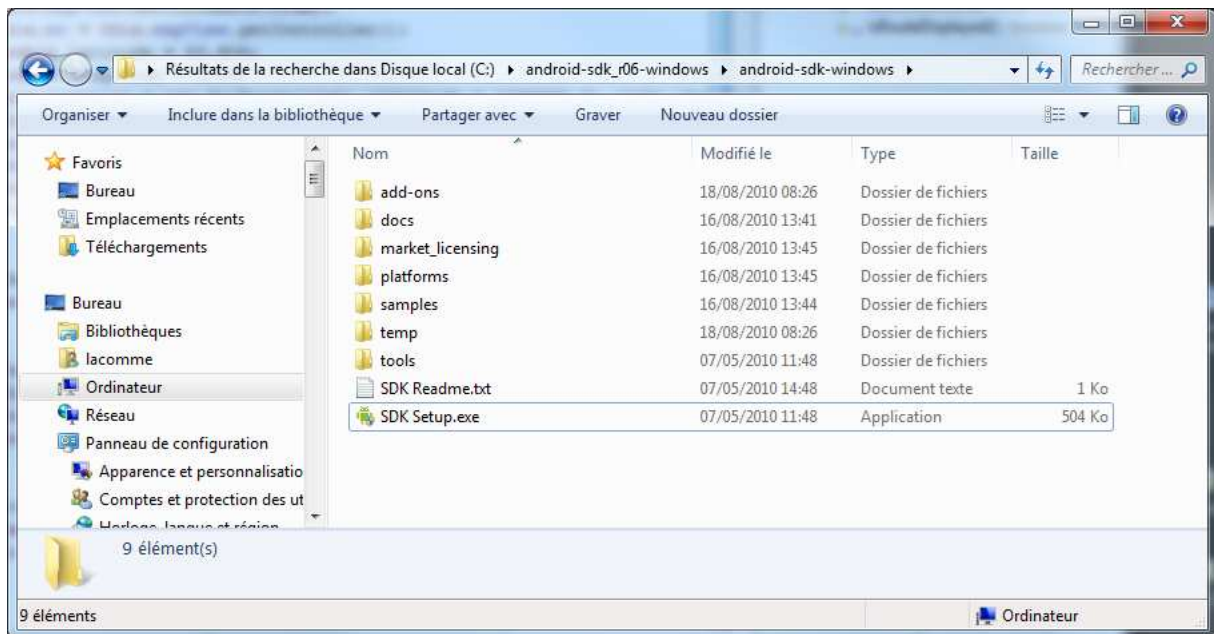
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }

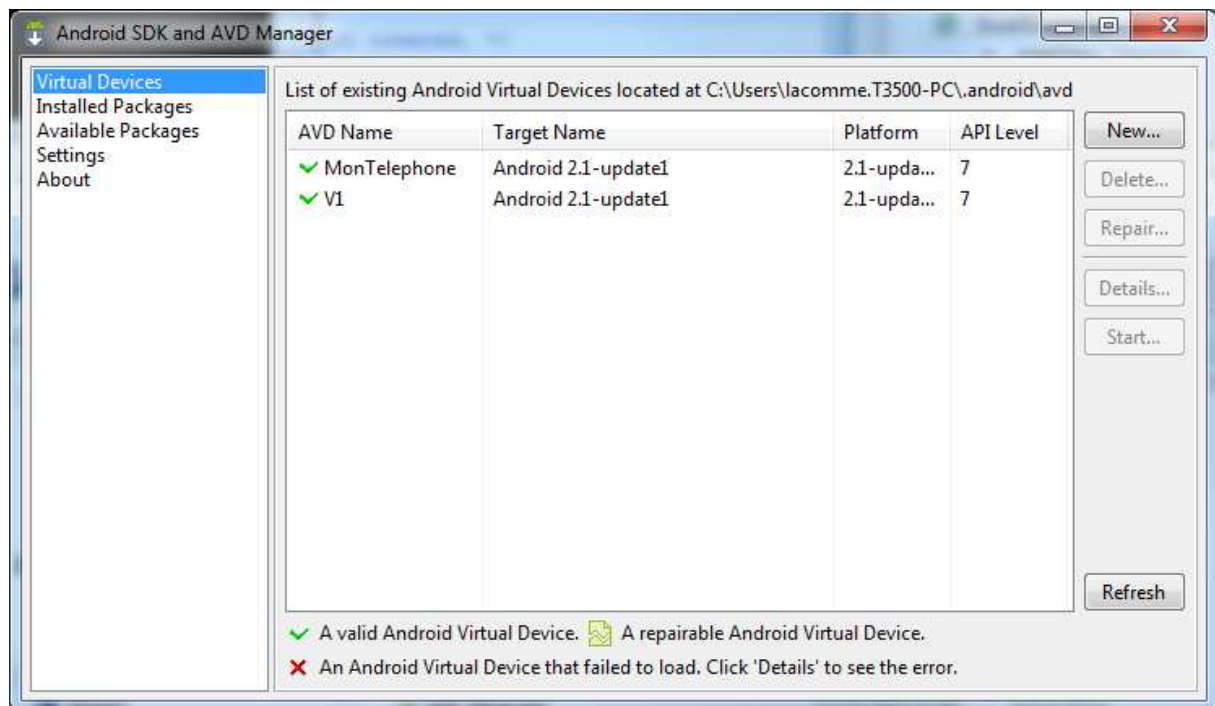
    @Override
    protected boolean isRouteDisplayed() {
        return false;
    }
}

```

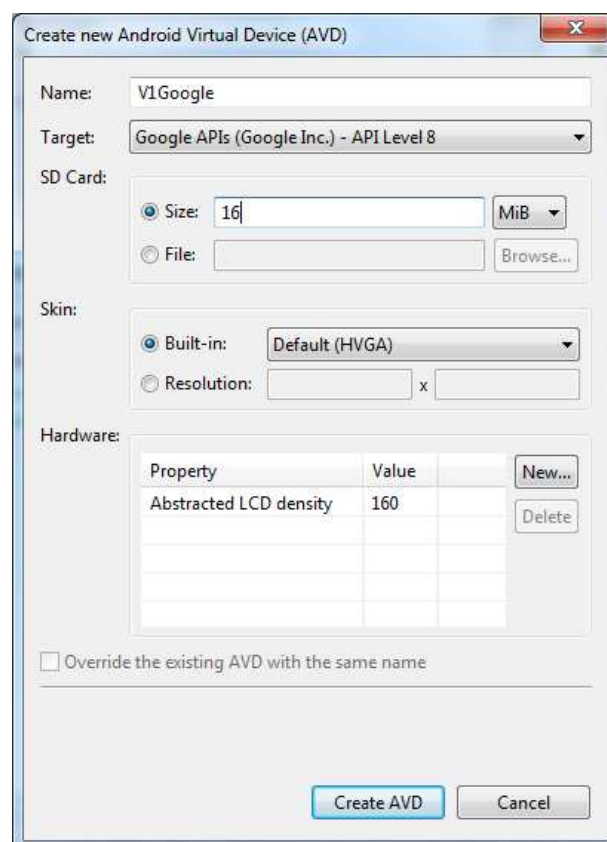
Etape 10. Créer un AVD compatible

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

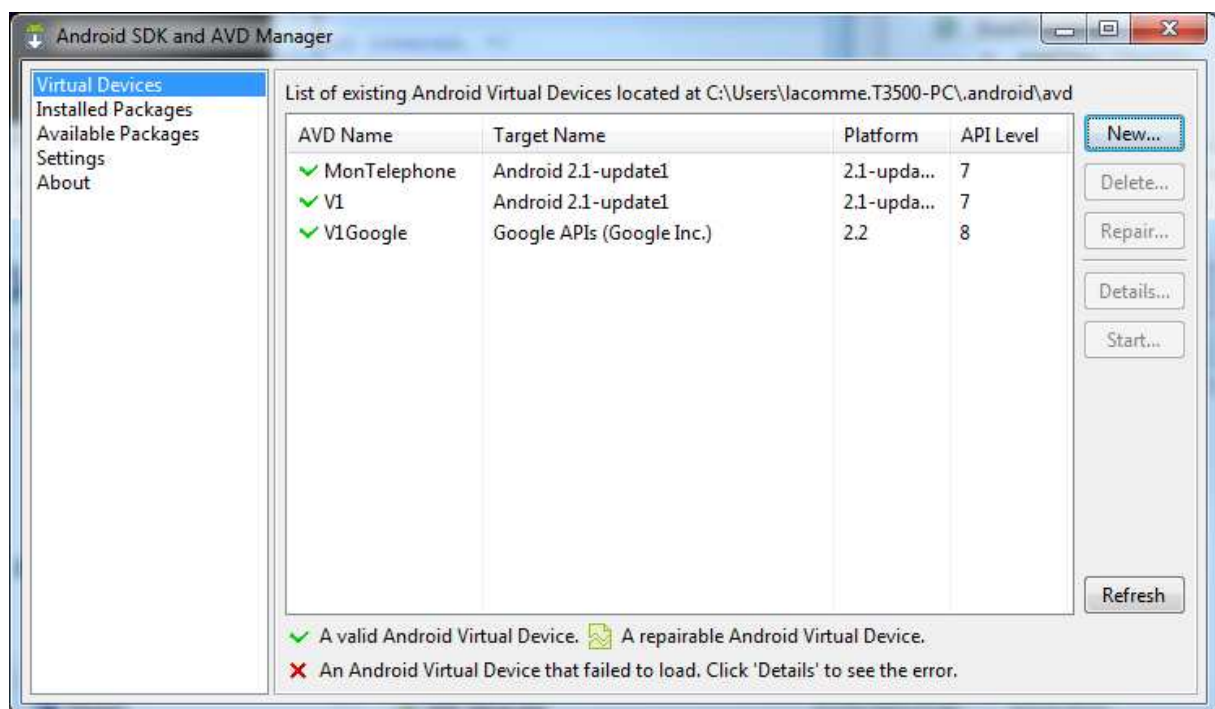




Faire ensuite **New**.



Ce qui donne à la fin :



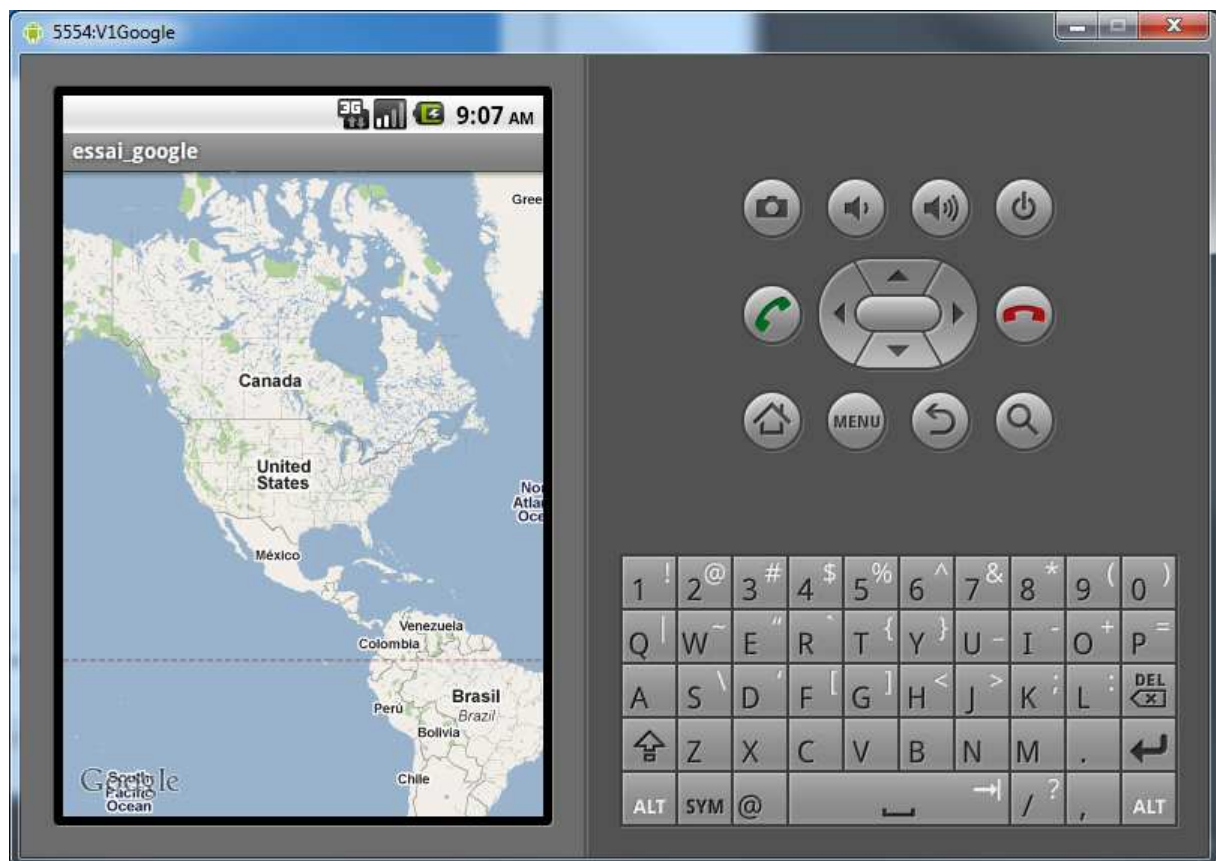
Etape 8. Exécuter l'application.

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);
    }

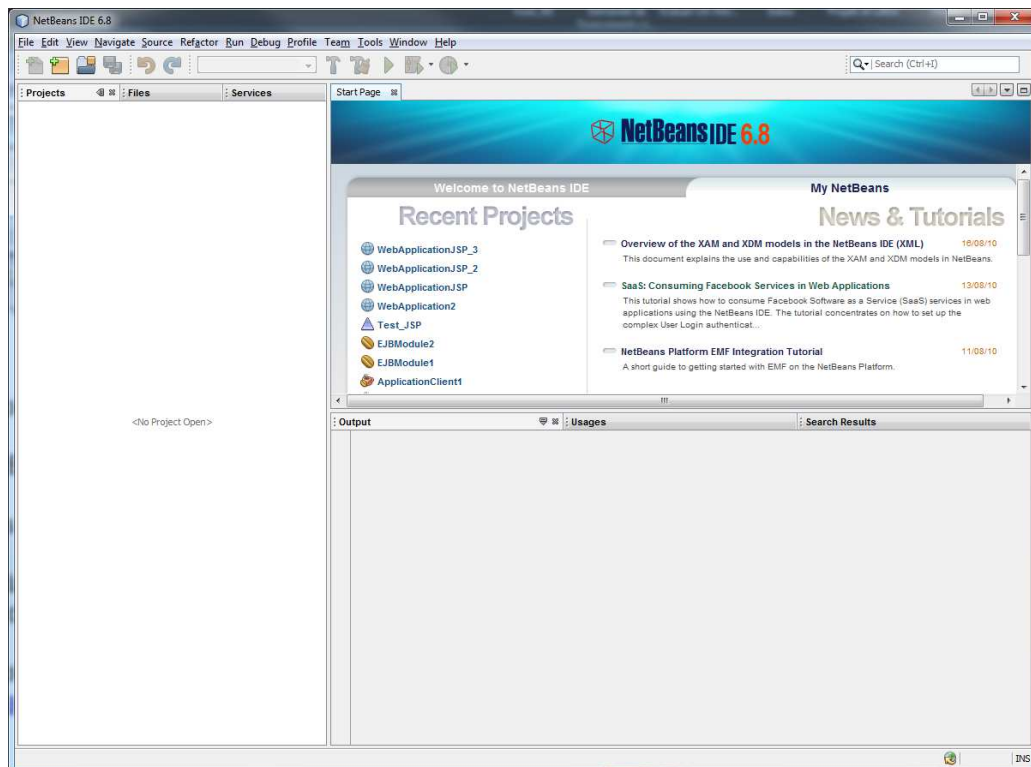
    @Override
    protected boolean isRouteDisplayed() {
        return false;
    }
}
```

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

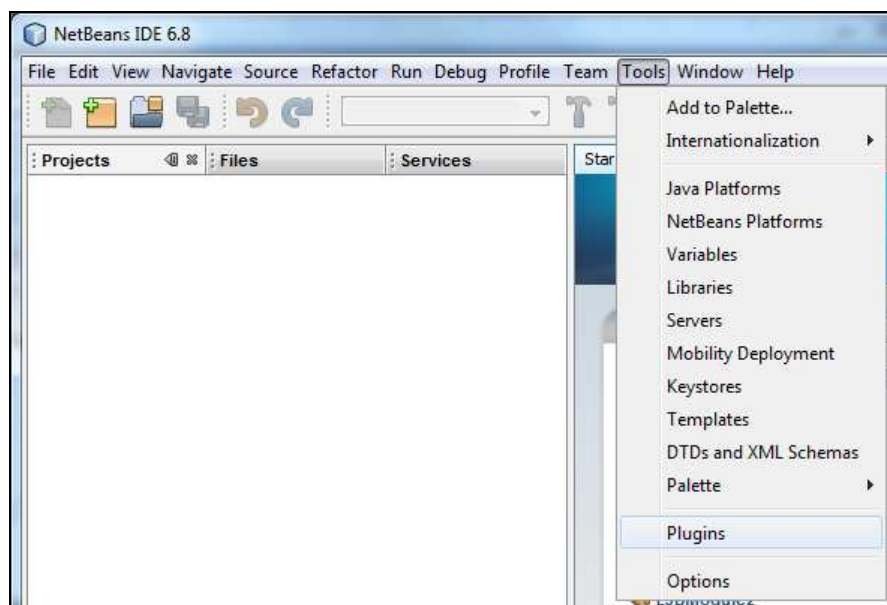


1. Configurer NetBeans

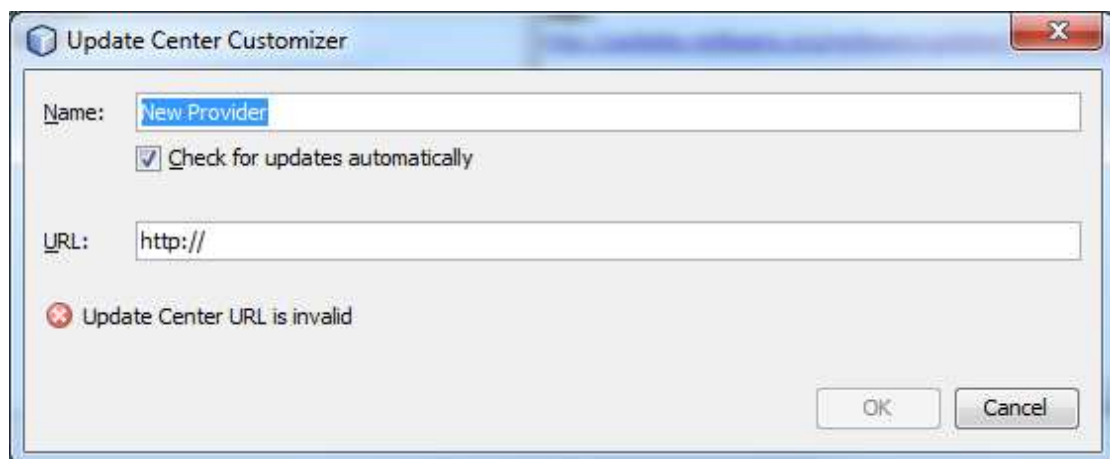
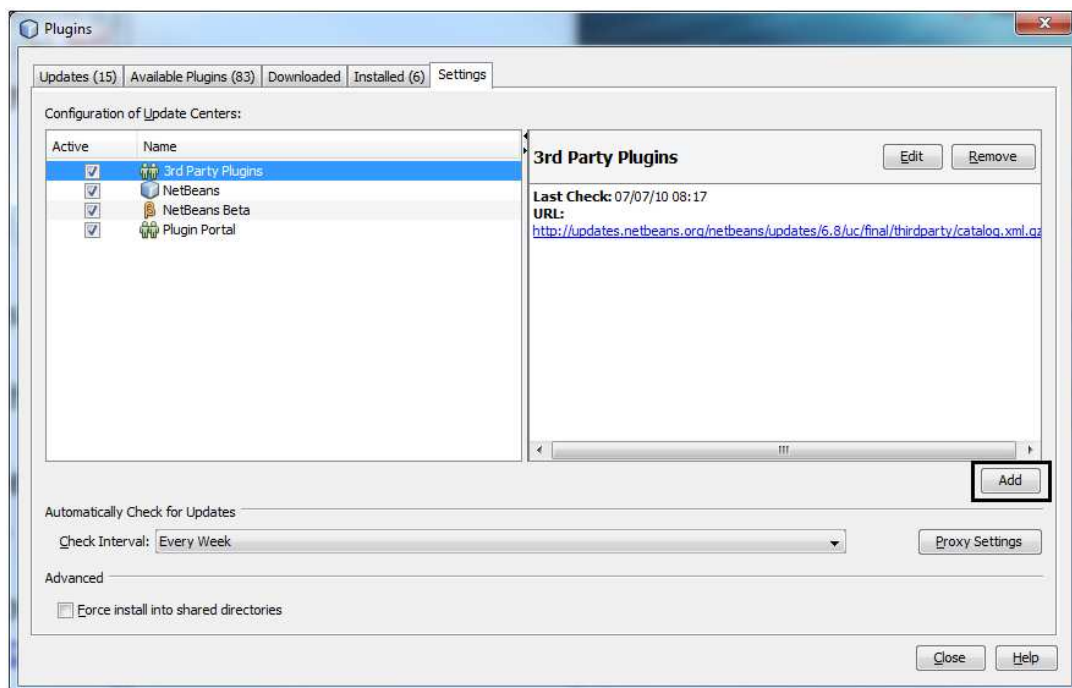
Lancer NetBeans.



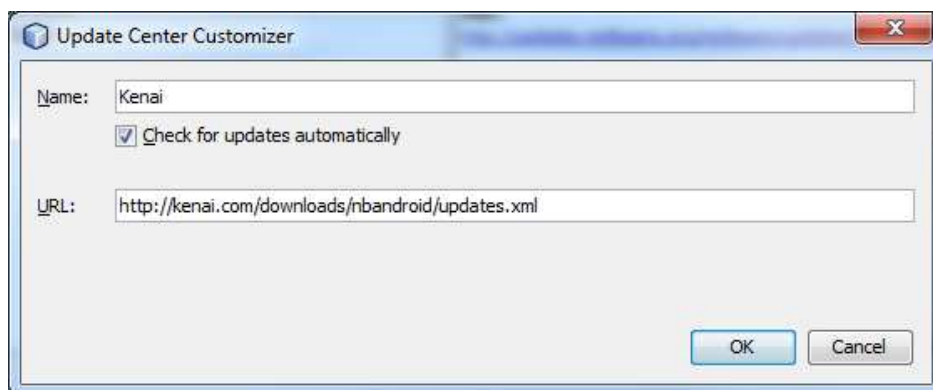
Aller dans le menu **Tools/Plugins**



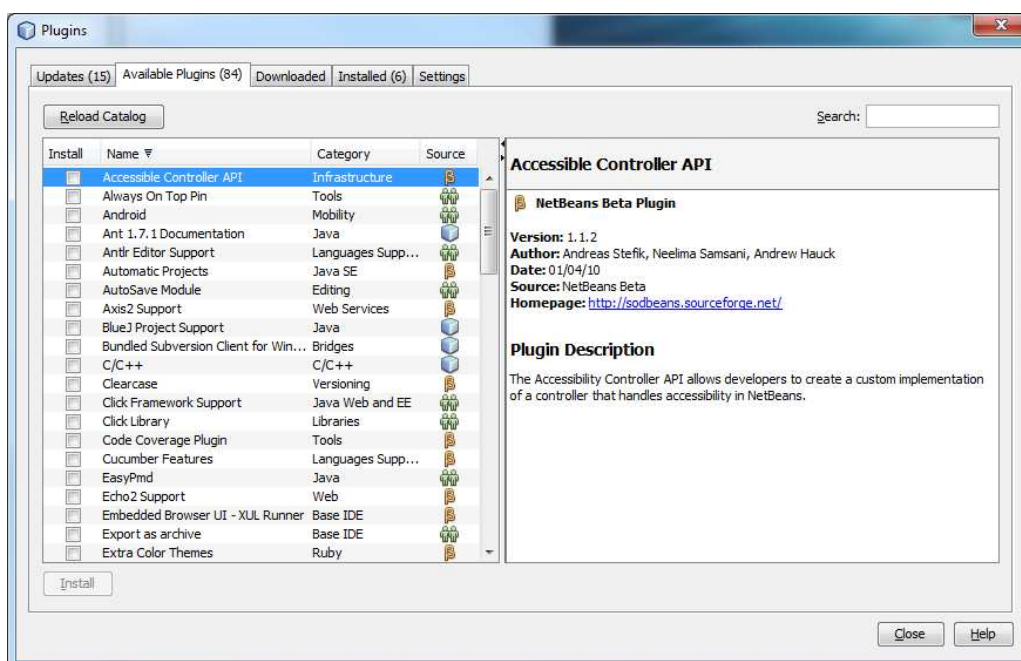
Aller dans **Settings** et faire ensuite **Add**.



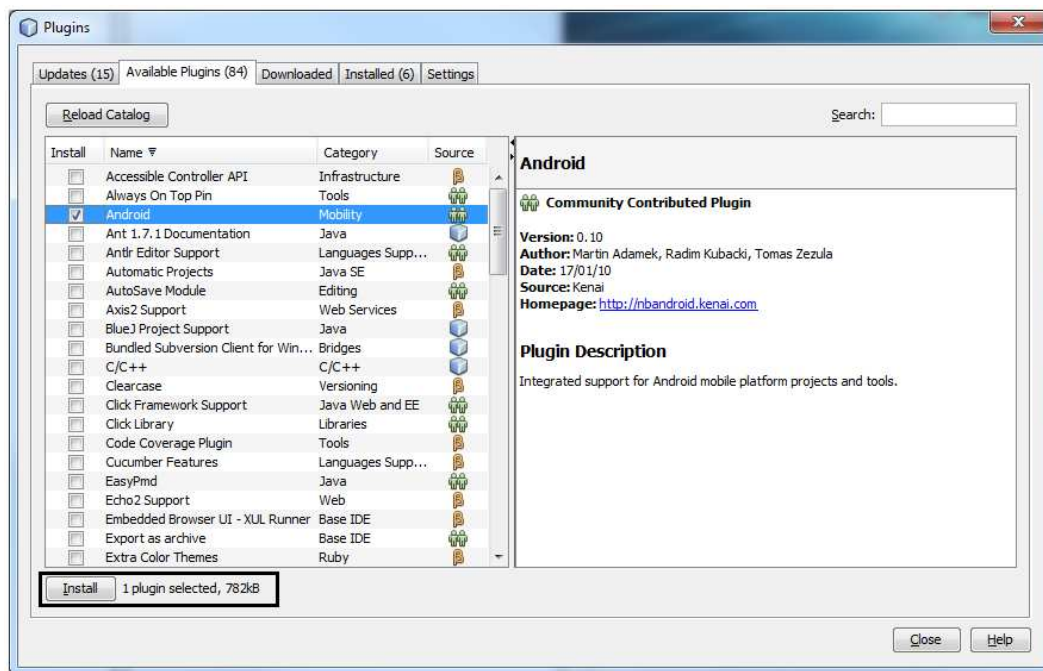
Dans le champ Name, on peut choisir Kenai par exemple et dans le champ URL l'adresse suivante : <http://kenai.com/downloads/nbandroid/updates.xml>



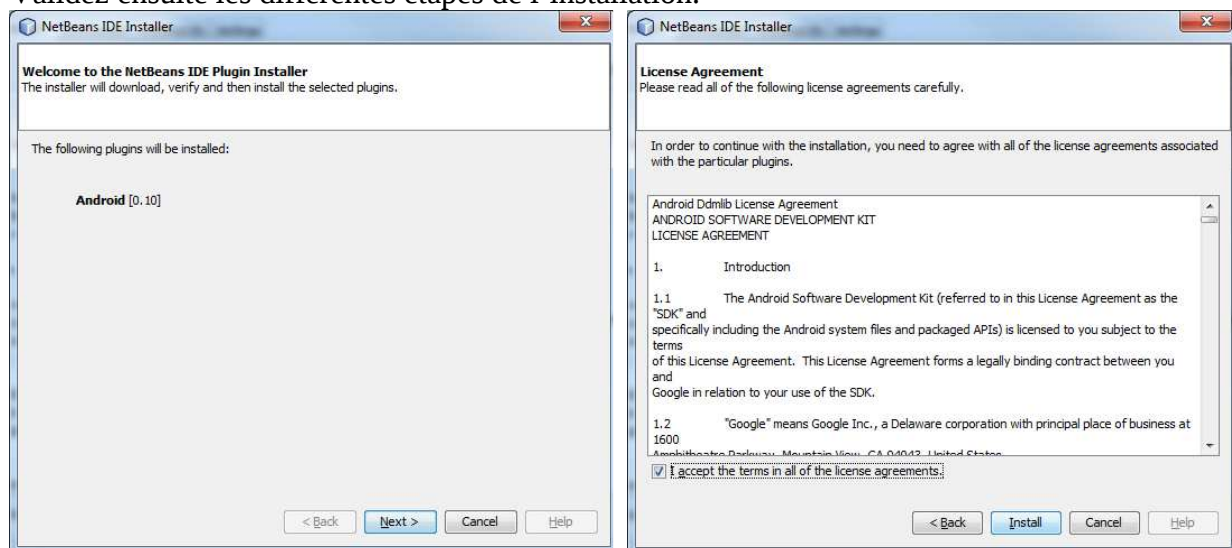
Consulter ensuite la partie **Available Plugins**.



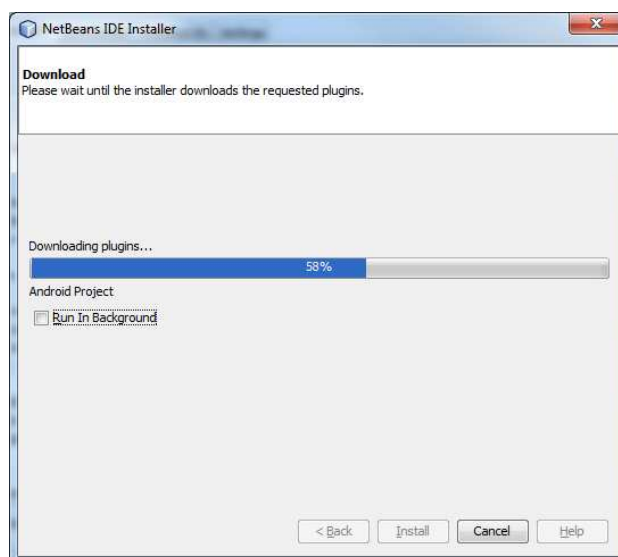
Choisir ensuite Android et lancez l'installation (bouton **Install**).



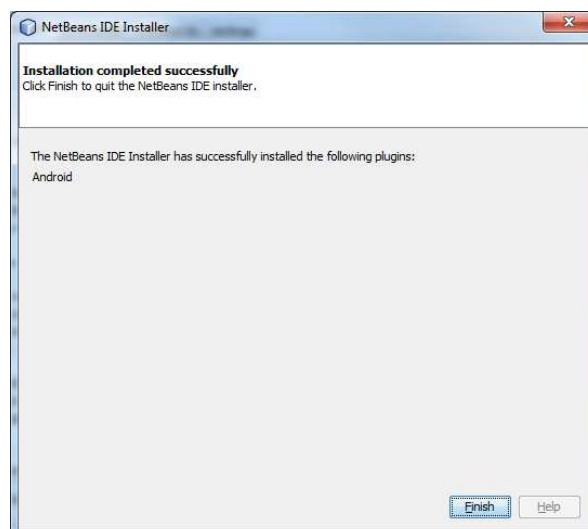
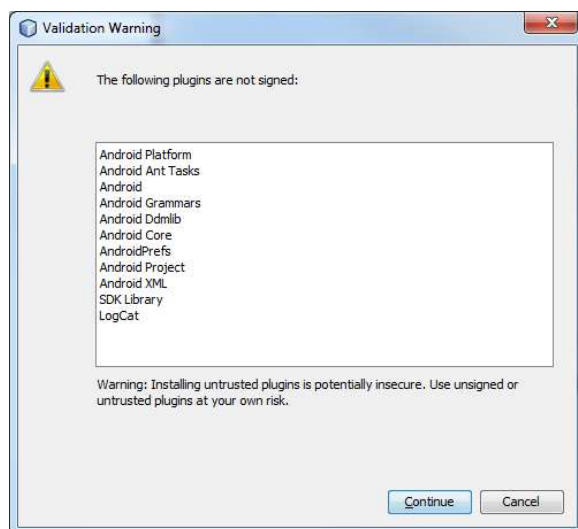
Validez ensuite les différentes étapes de l'installation.



Patientez quelques secondes pendant l'installation...

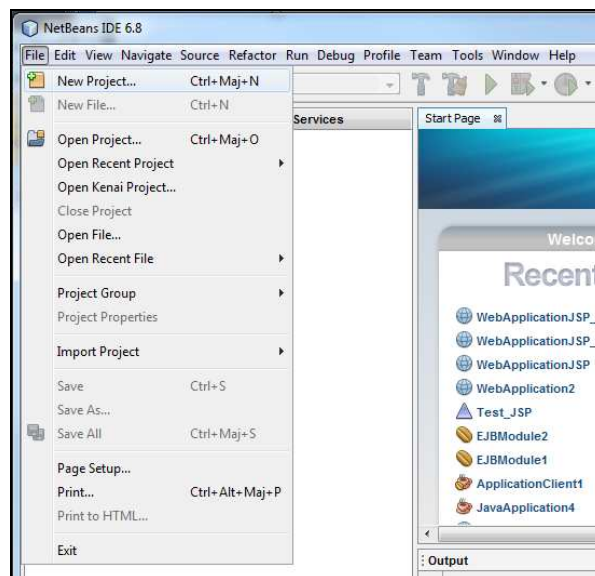


Certains plugins ne sont pas signés. Une autorisation est nécessaire pour réaliser l'installation.

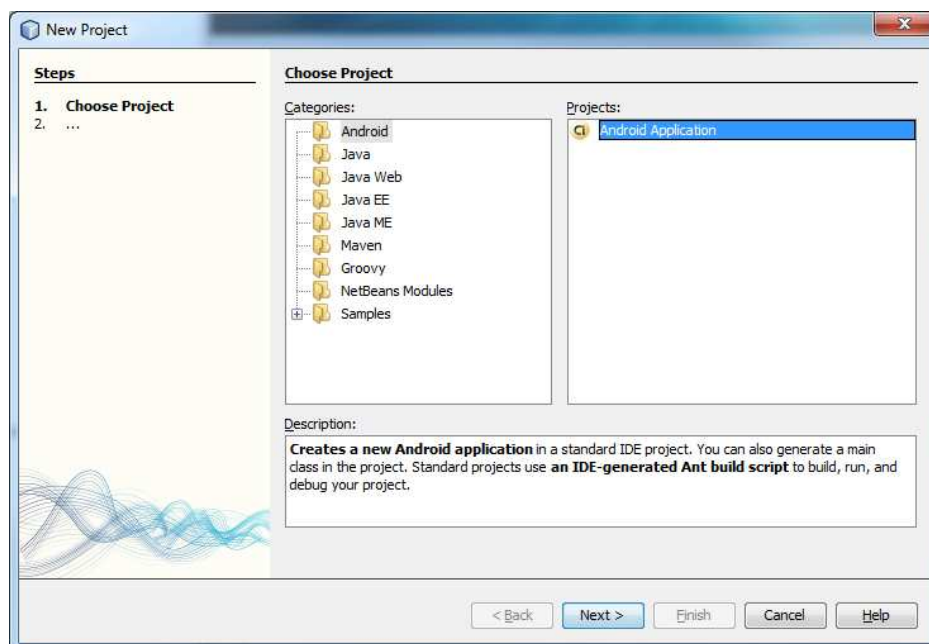


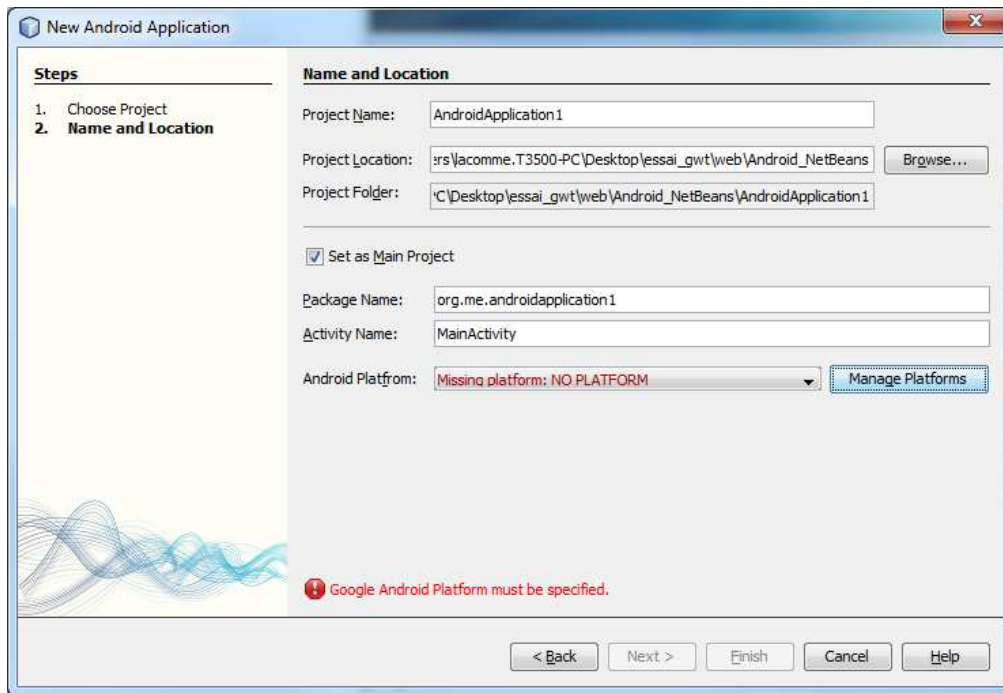
2. Créer un projet Android

Faire **File / New Project**

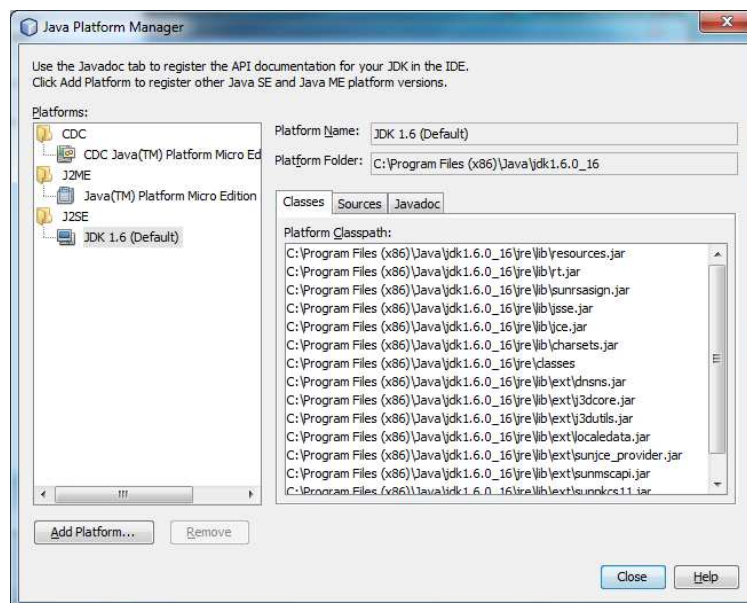


Choisir **Android** et **Android Application**.

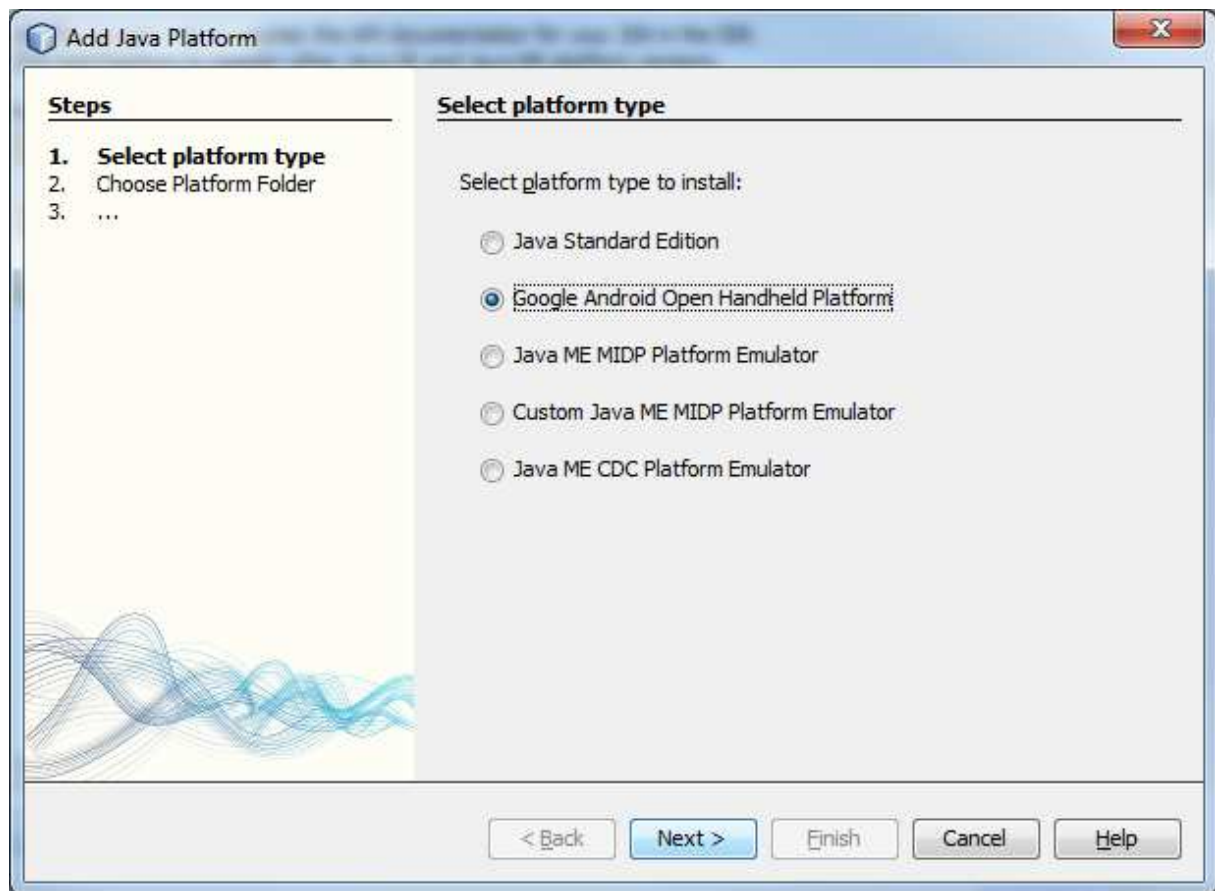




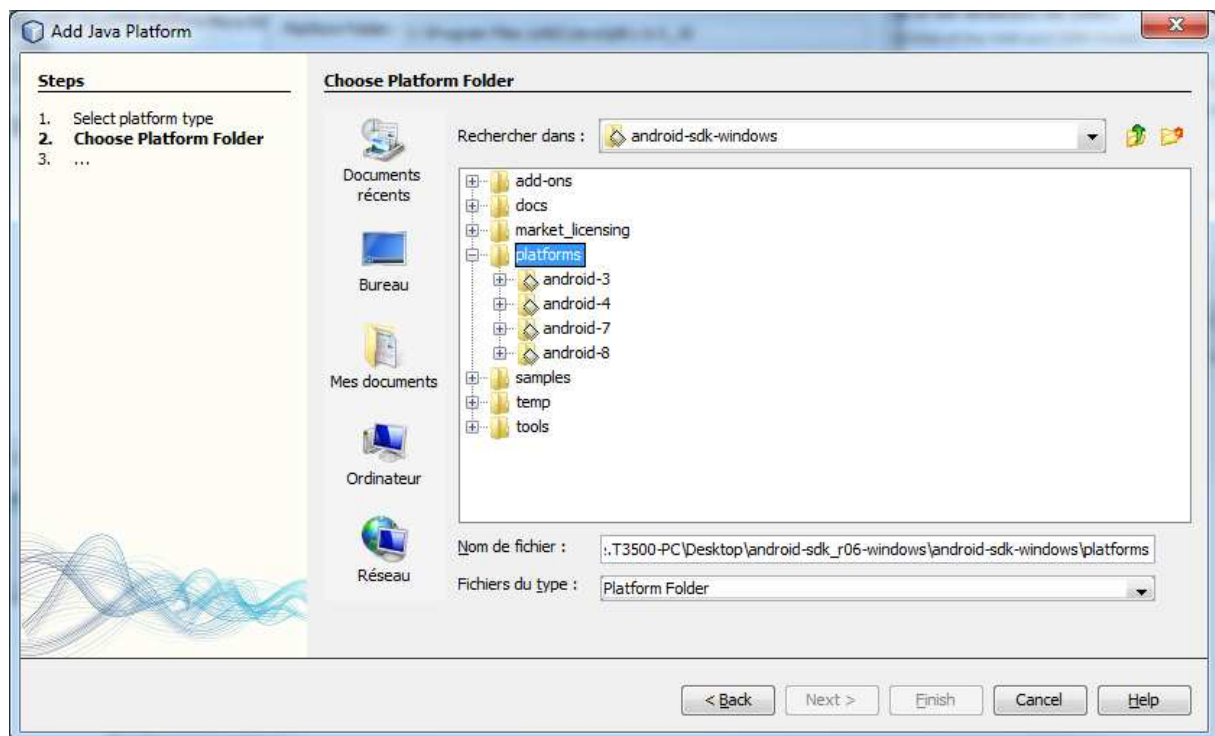
Choisir ensuite **Manage Platforms**.

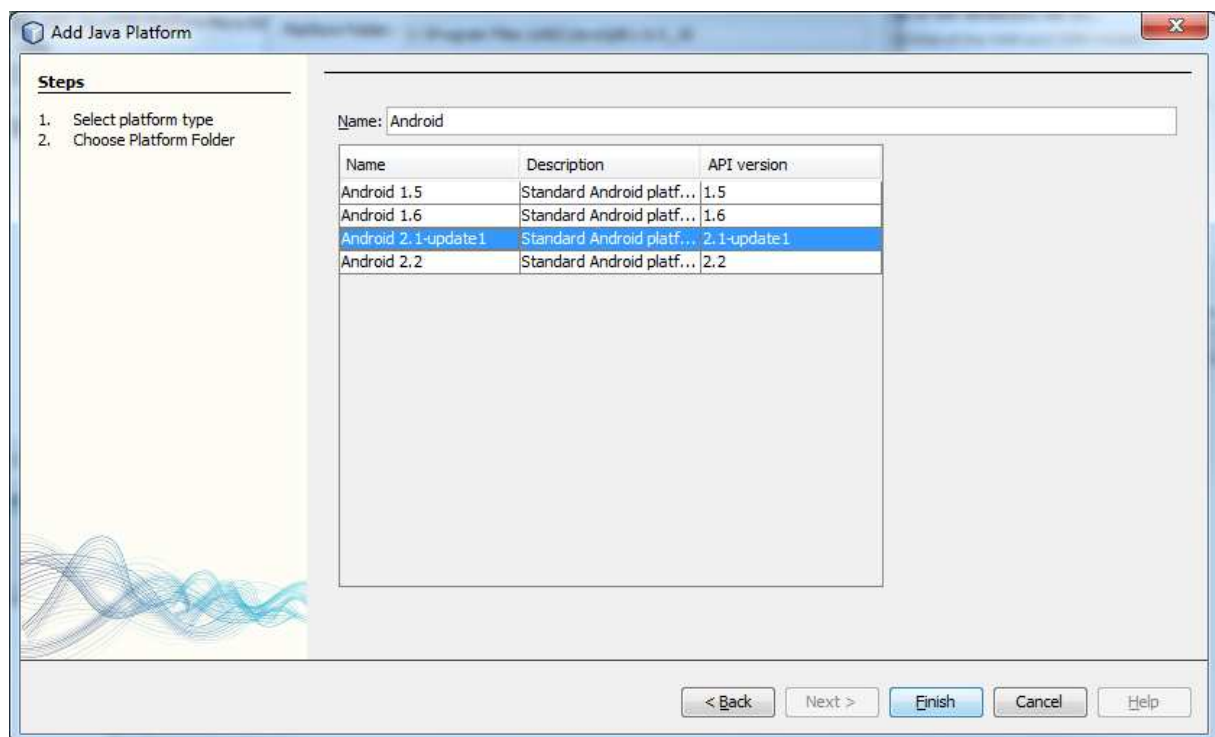
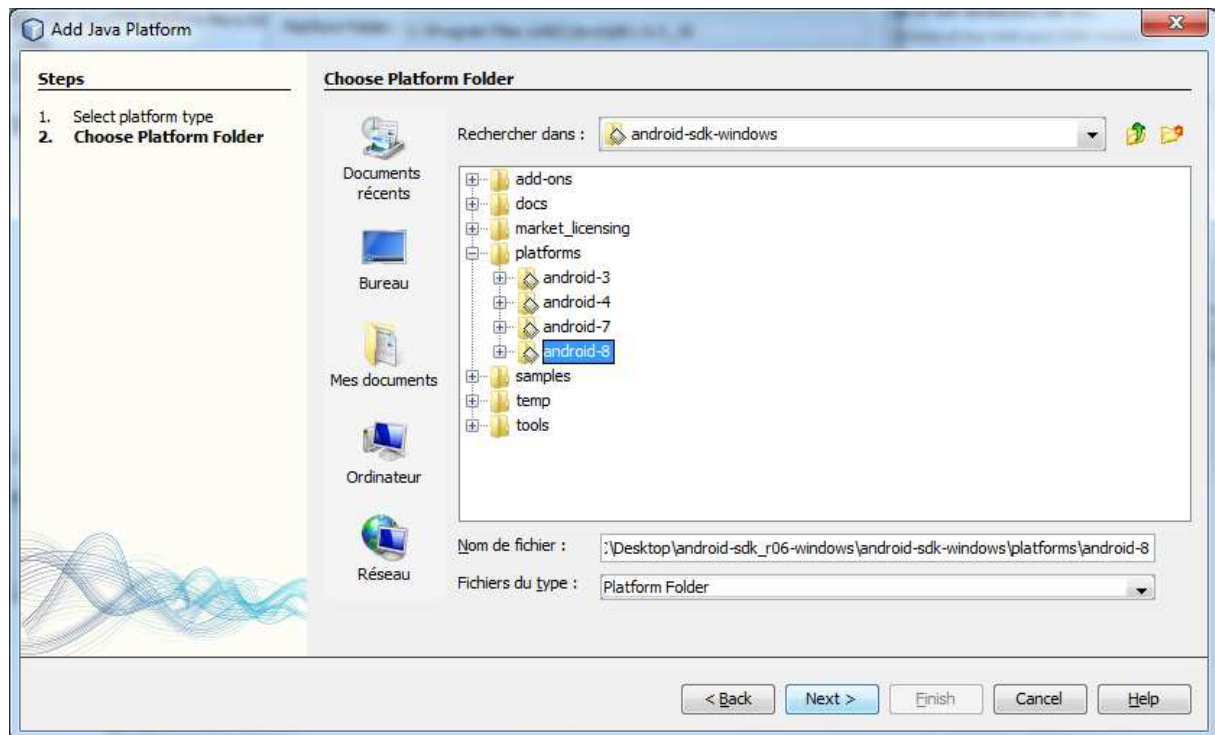


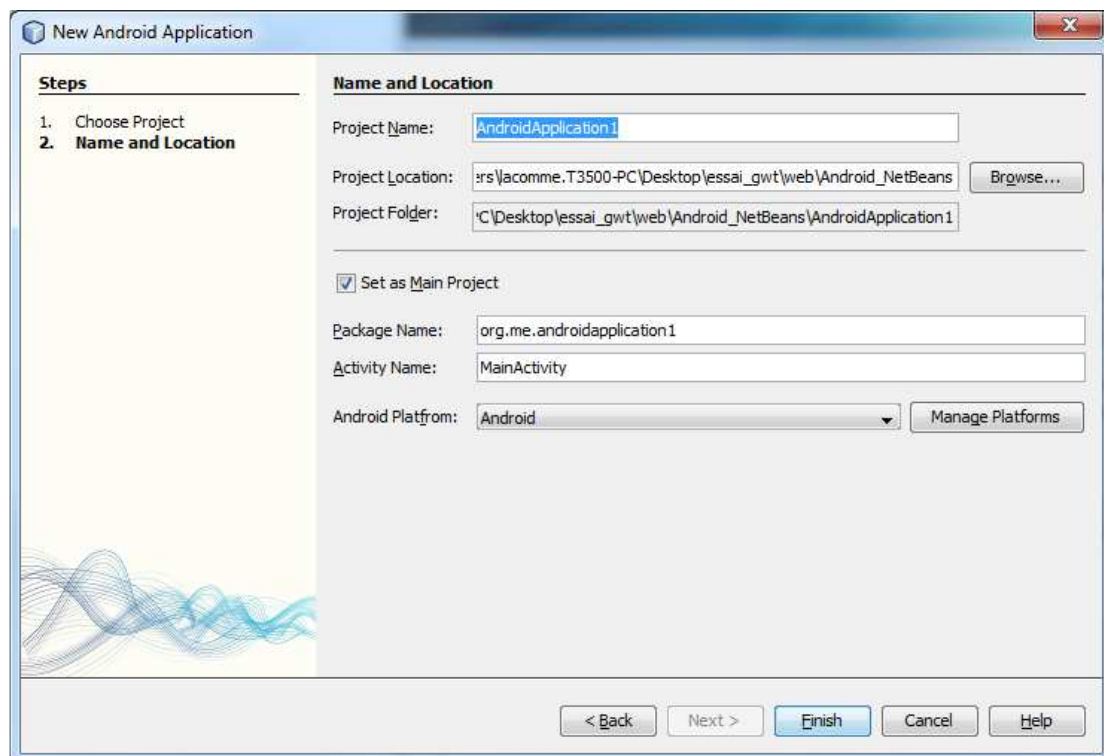
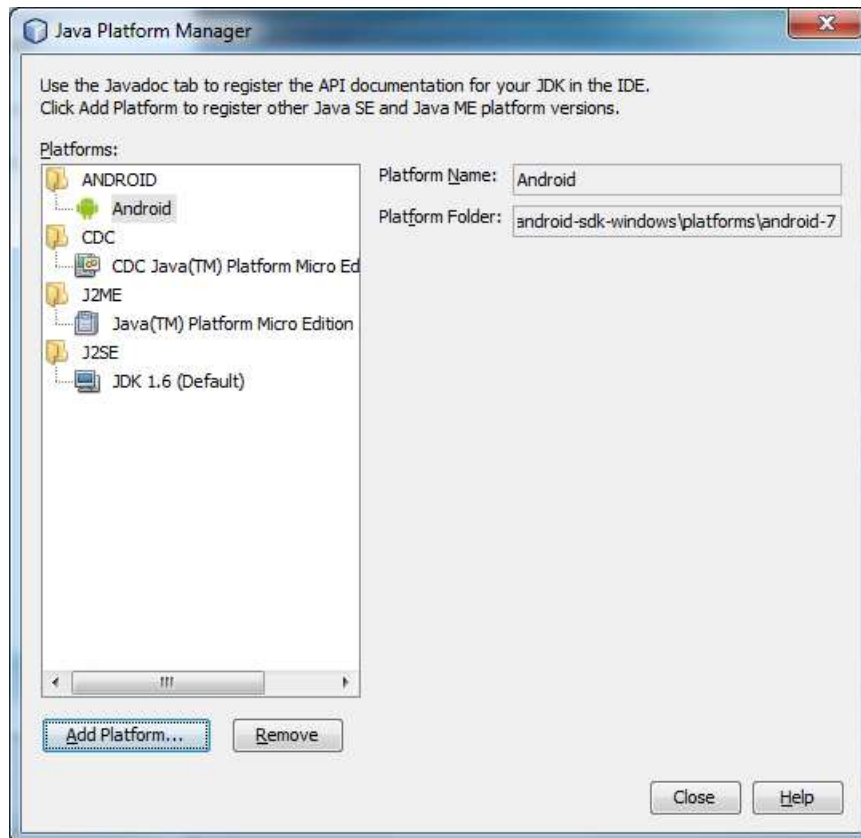
Choisir **Add Platform**

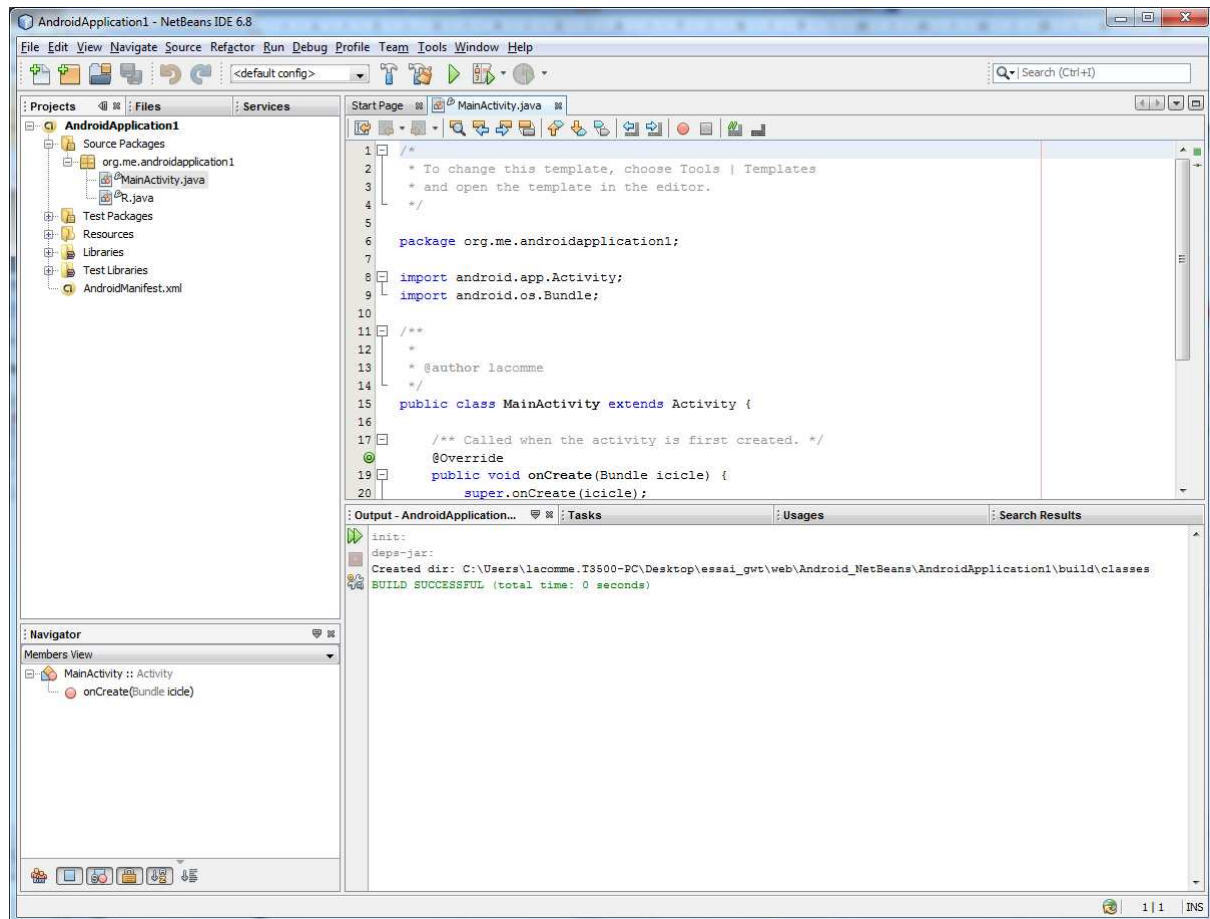


Allez dans le répertoire **android-sdk-windows / platforms**.









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 icle) {
        super.onCreate(icle);
        // 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);
    }
}
```

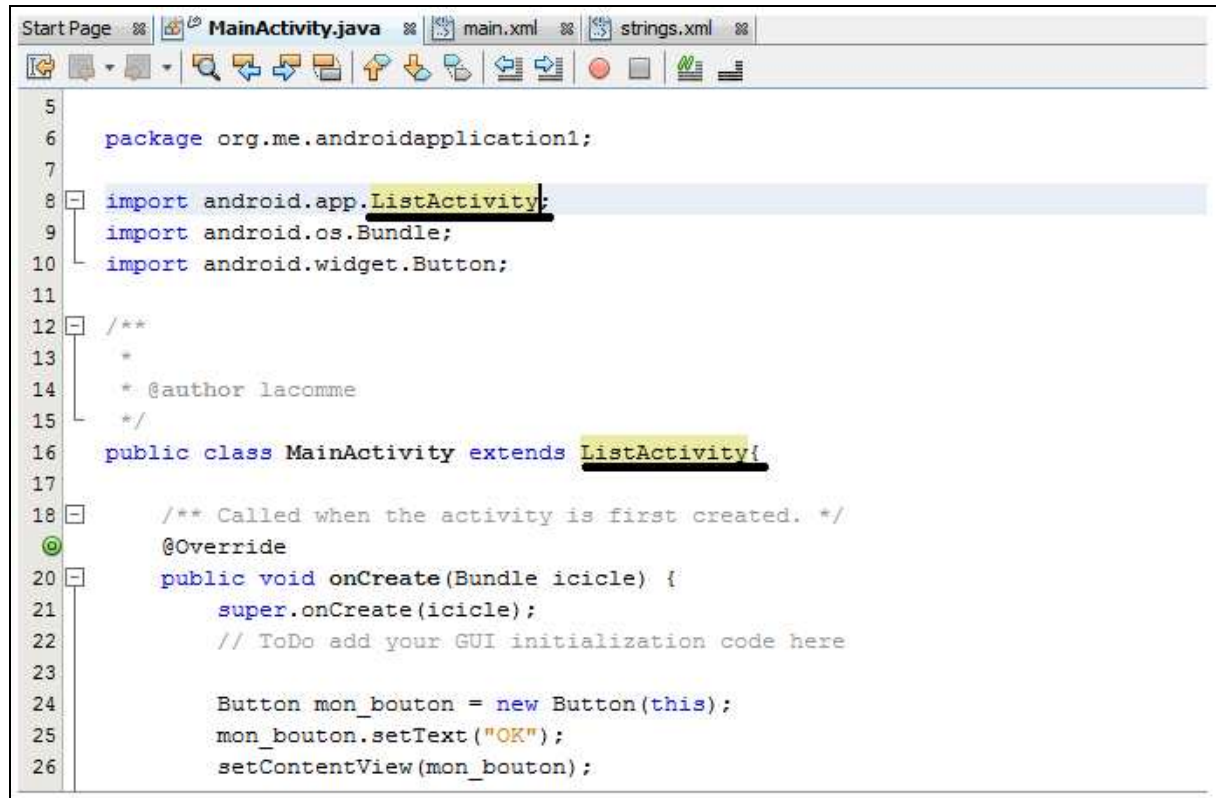
Le résultat d'exécution doit se présenter comme suit :



4. Gestion des listes

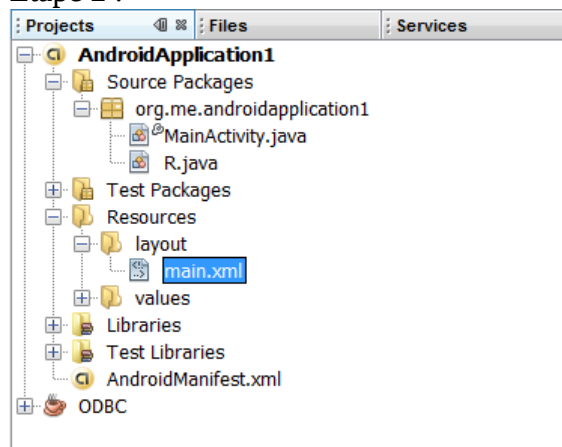
Création d'une liste.

Etape 1 : il faut modifier le type d'application créée. Remplacer Activity par ListActivity.



```
5
6 package org.me.androidapplication1;
7
8 import android.app.ListActivity;
9 import android.os.Bundle;
10 import android.widget.Button;
11
12 /**
13  *
14  * @author lacomme
15  */
16 public class MainActivity extends ListActivity{
17
18     /** Called when the activity is first created. */
19     @Override
20     public void onCreate(Bundle icle) {
21         super.onCreate(icle);
22         // ToDo add your GUI initialization code here
23
24         Button mon_bouton = new Button(this);
25         mon_bouton.setText("OK");
26         setContentView(mon_bouton);
27     }
28 }
```

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 = {
        "AAAAAAA", "BBBBBBBB", "CCCCCCCC", "DDDDDDDD", "EEEEEEEE",
        "FFFFFFF", "GGGGGGGG", "HHHHHHHH", "IIIIIIII", "JJJJJJJJ",
        "KKKKKKKK", "LLLLLLLL", "MMMMMMMM", "NNNNNNNN", "OOOOOOOO",
        "PPPPPPPP", "QQQQQQQQ", "RRRRRRRR", "SSSSSSSS", "TTTTTTTT",
        "UUUUUUUU", "VVVVVVVV", "WWWWWWW", "XXXXXXXX", "YYYYYYYY",
        "ZZZZZZZZ"
    };

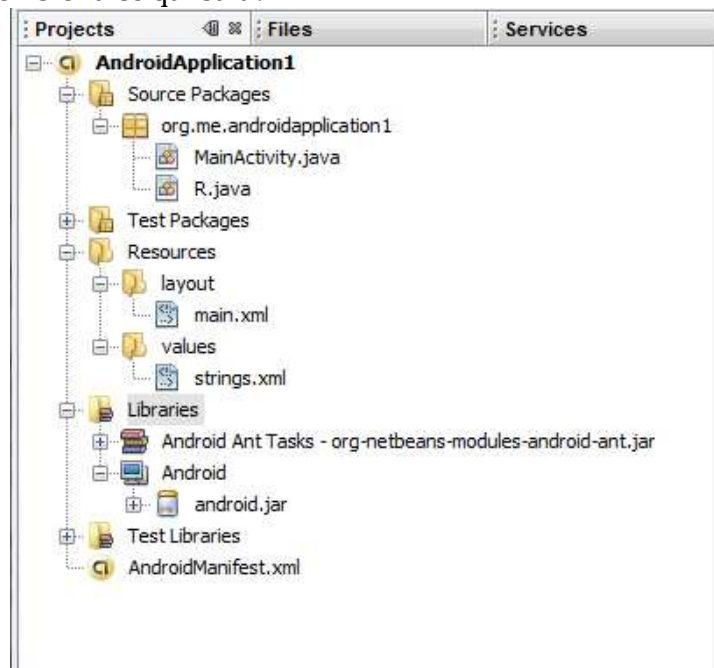
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        // ToDo add your GUI initialization code here
        setContentView(R.layout.main);
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
android.R.layout.simple_list_item_1, mStrings);

        setListAdapter(adapter);

    }
}
```

Etape 4 : Inclure la librairie Android Ant Task.

Le projet doit ressembler à ce qui suit :



Etape 5 : Tester la liste.



----- FIN -----