



Terminaux mobiles et web services : Une nouvelle méthode d'accès aux données

Jeudi 5 Septembre 2013 – JDev2013

Romain Guidoux, Libo Ren,
Jonathan Fontanel, Philippe Lacomme



Terminaux mobiles et web services : Une nouvelle méthode d'accès aux données

Document sous Licence Creative Commons

Copyright (c) 2013. **Fontanel-Guidoux-Lacomme-Ren**



« Cette licence permet aux autres de remixer, arranger, et adapter votre œuvre à des fins non commerciales tant qu'on vous crédite en citant votre nom et que les nouvelles œuvres sont diffusées selon les mêmes conditions. »



Qui sommes-nous ?

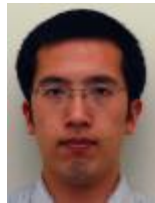
<http://www.isima.fr/~lacomme/pagewebservice/websevice/>



Jonathan Fontanel (Qualiac)



Philippe Lacomme (UBP, LIMOS)



Libo Ren (UdA, LIMOS)

- Stand INRA au salon de l'agriculture
- Application smartphone couplée à un web service
- Expert des systèmes mobiles Android™



Romain Guidoux (INRA)



Pré-requis



**Quelques
smartphones
disponibles en prêt**



Android is a trademark of Google Inc.

The Android robot is reproduced or modified from work created and shared by Google and used according to terms described in the Creative Commons 3.0 Attribution License.



Plan de l'atelier

- **Partie I (env. 40 min)**

- Intérêt des applications de type client-serveur
- Présentation de REST et SOAP
- Présentation de l'environnement Android

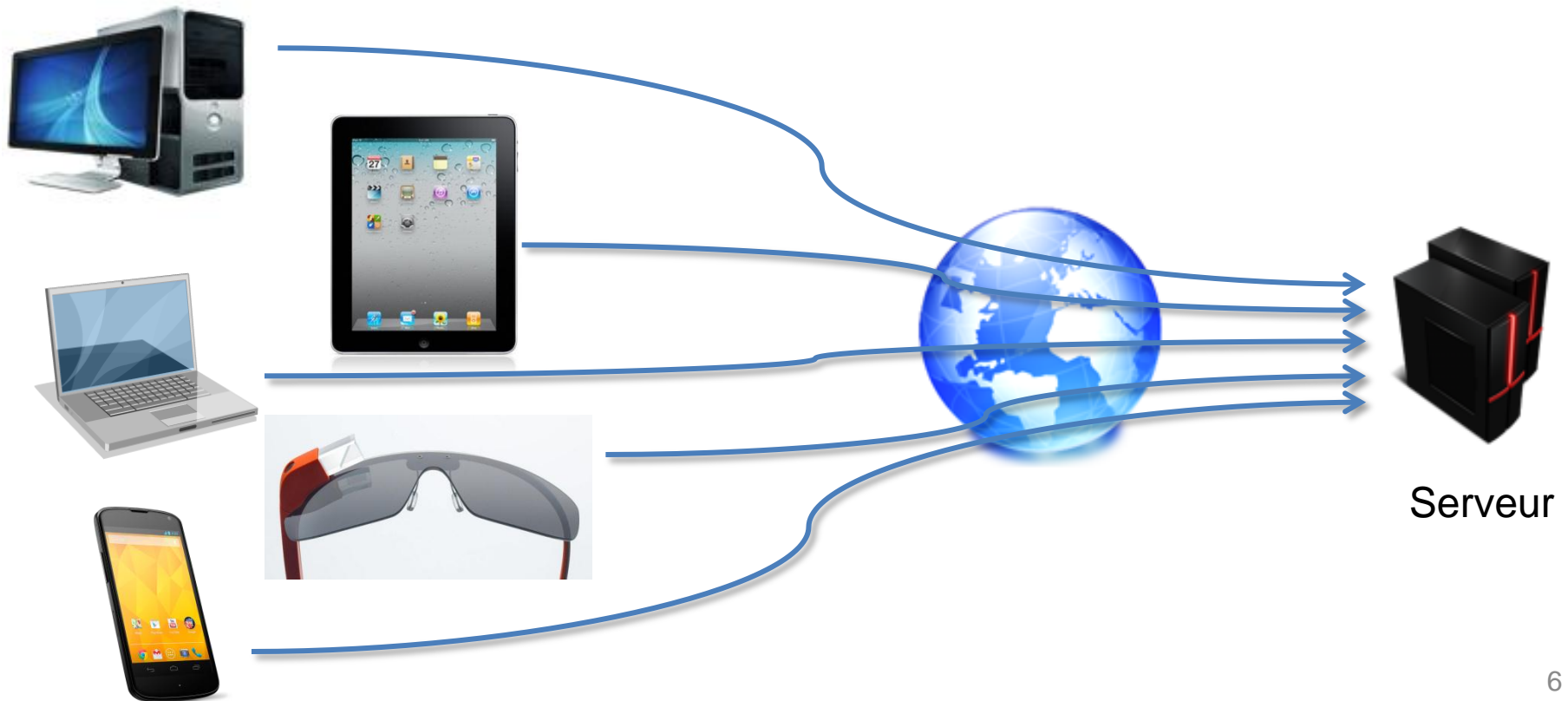
- **Partie II (env. 2h et 20 min)**

- Création d'un WS de géocodage d'adresse IP (REST)
- Création d'un WS de conversion de devises (SOAP)



Architecture client-serveur

- Les clients peuvent être divers et variés
- Les procédures sont stockées sur un serveur d'applications





Architecture client-serveur

- **Dans l'univers grand public :**

- Application météo
- Programme TV
- GPS Waze
- ...



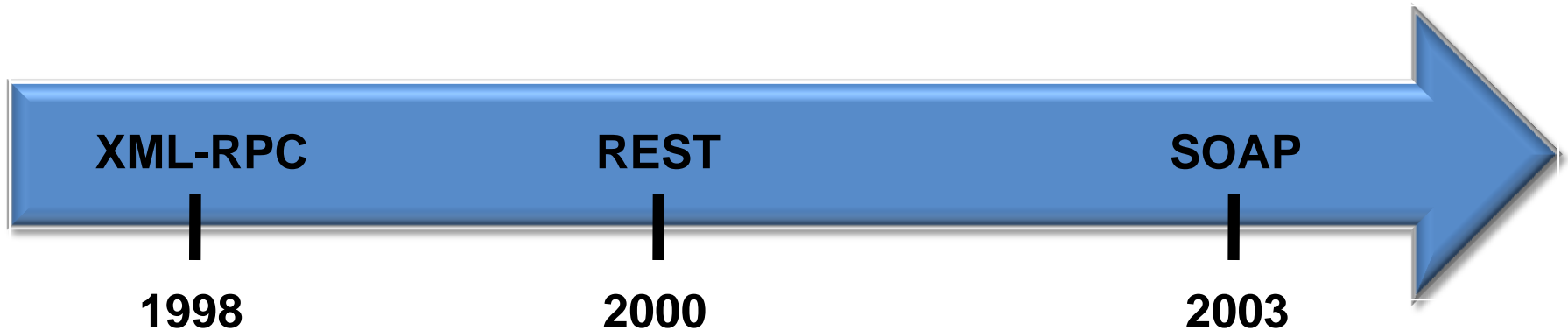
- **Dans l'univers de la recherche :**

- Archive HAL
- Développement d'une application d'estimation de la dépense énergétique à l'INRA
- Mise à disposition de métaheuristiques par le LIMOS
- ...





Un peu d'histoire



→ REST : R.T. Fielding and R.N. Taylor,
Principled design of the modern web architecture,
ACM Trans. on Internet Technol. (TOIT), 2(2), pp. 115–150. 2002

→ Issam RABHI
Testabilité des services Web.
LIMOS, soutenu en janvier 2012



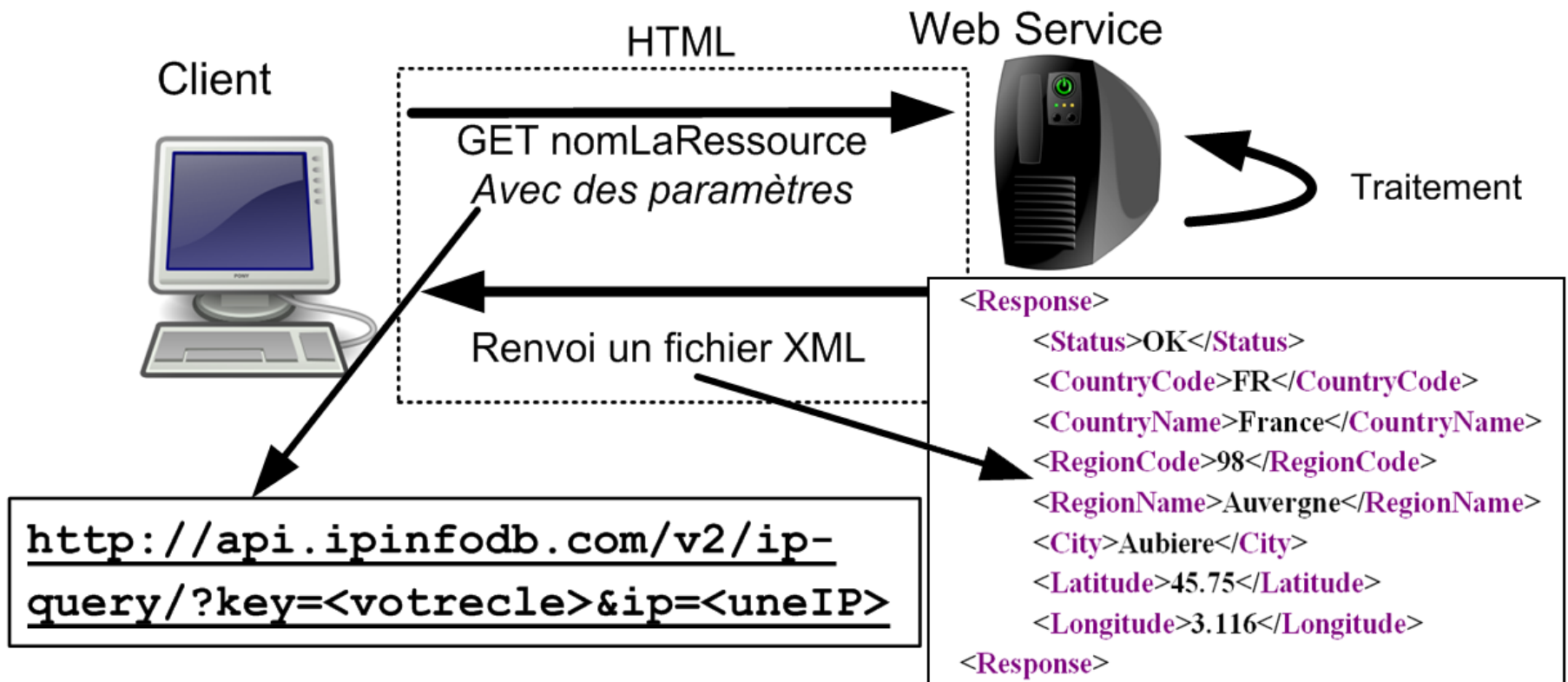
L'architecture REST

-
- Architecture => pas de spécification W3C
 - Le web service ne conserve pas l'état courant (stateless)
 - Communication via les méthodes du protocole HTTP :
 - **GET** : lecture
 - **POST** : écriture
 - **PUT** : modification
 - **DELETE** : suppression
 - Les ressources sont identifiées par une URI (ex : URL)
 - Aucune restriction sur le format des réponses (XML, JSON, etc)



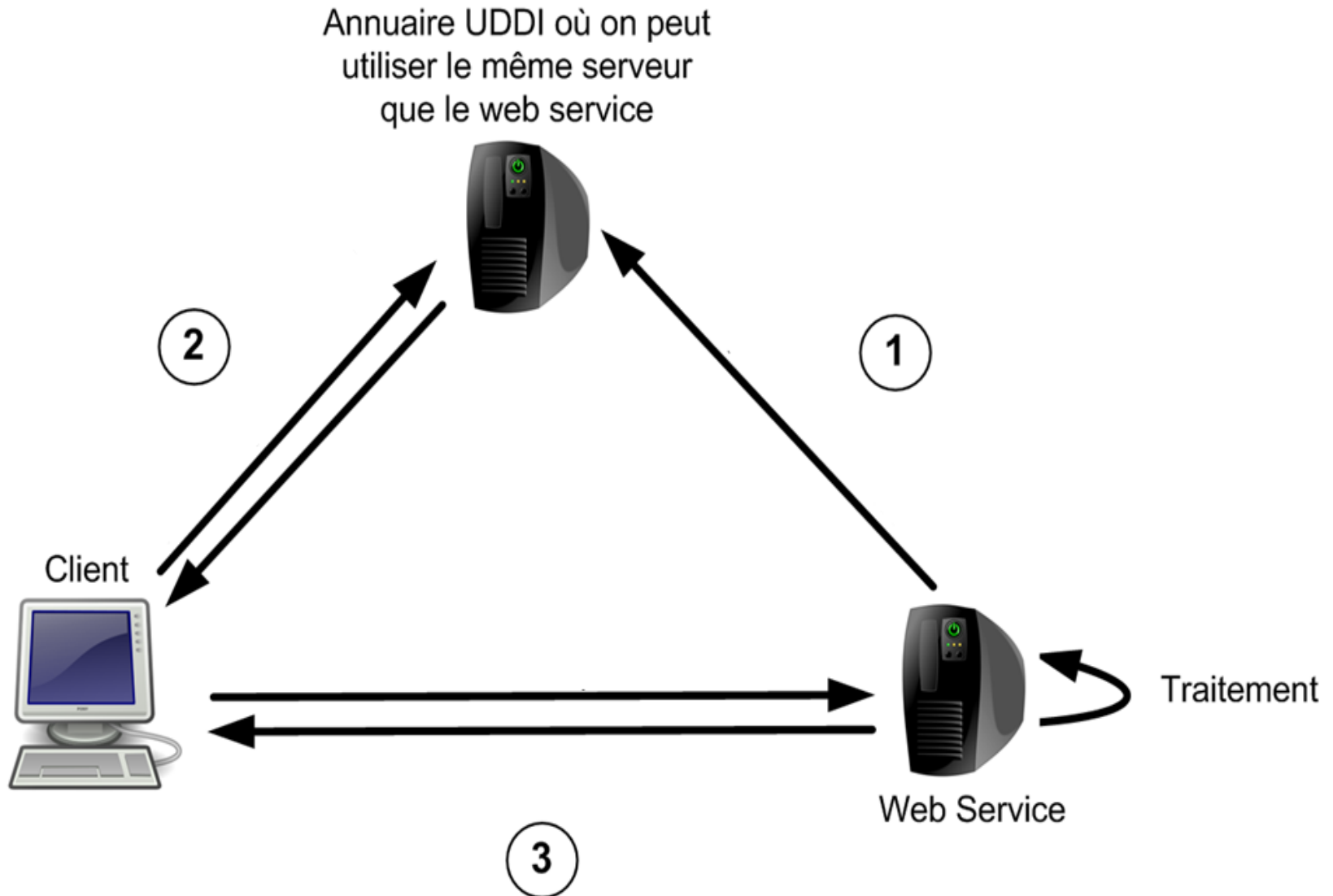
L'architecture REST

Architecture de type REST sur un exemple :





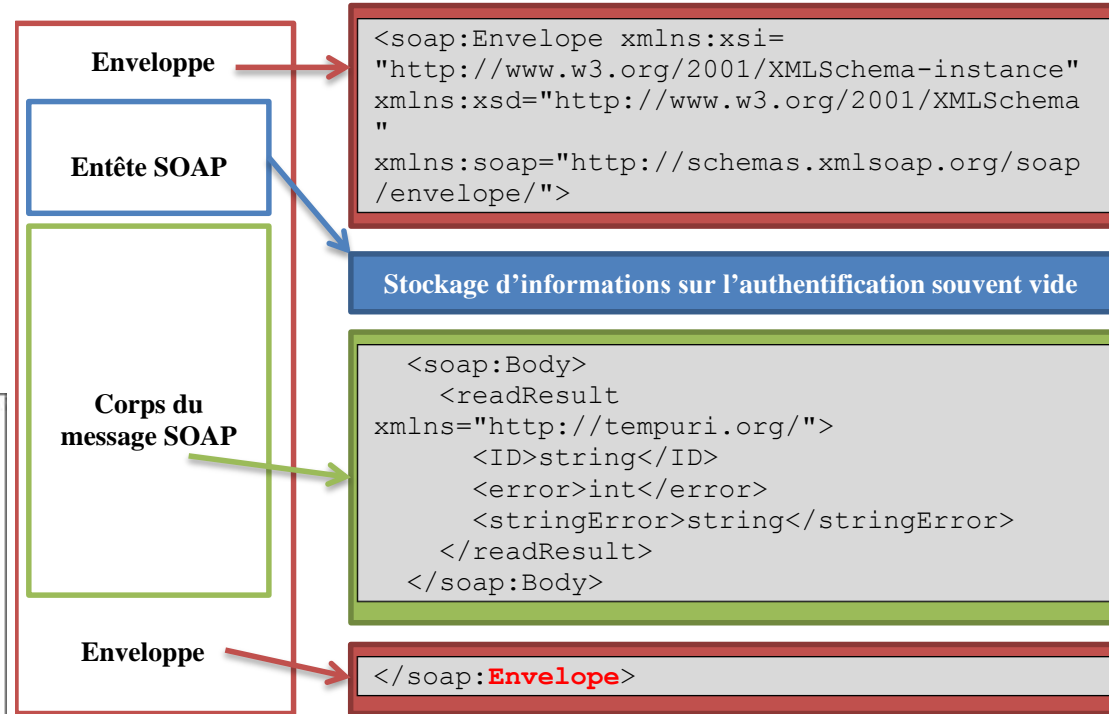
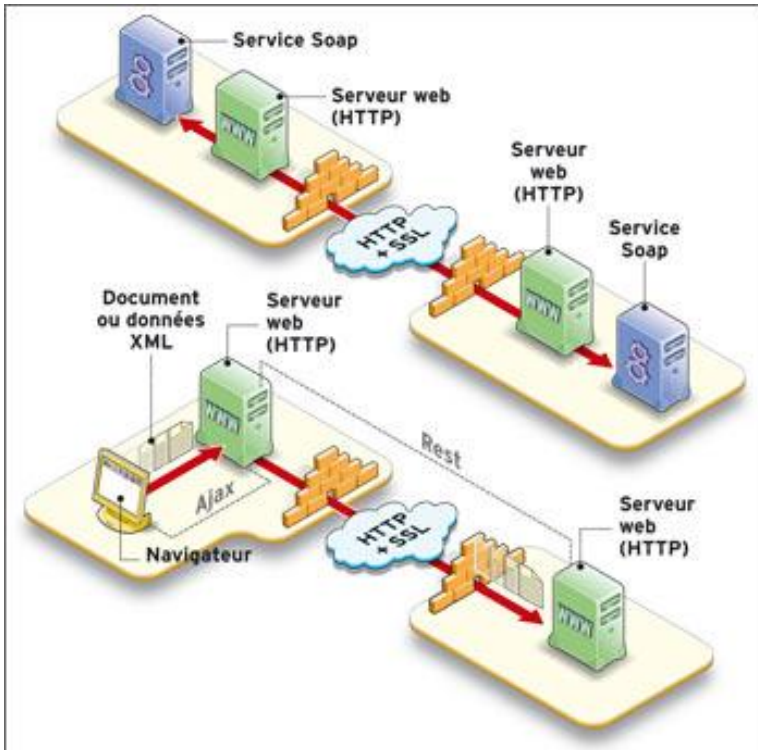
Le protocole SOAP





Le protocole SOAP

- Message SOAP :





SOAP versus REST

	SOAP	REST
Standard	-	+
Sécurité	+	-
Agile	-	+
Création client	+/-	-
Outils de dév	+	+/-
Performance	-	+
Support	+	-



Exemple d'un WS de géocodage d'adresses IP (REST)



Description du web service

- But : localiser une machine à partir de son adresse IP

http://ipinfodb.com/ip_location_api.php

IP Address Geolocation XML API

The API returns the location of an IP address (country, region, city, zipcode, latitude, longitude) and the associated timezone in XML format. You can find below code samples with PHP, Javascript, Ruby, Python and ASP.

- Réponse en XML, JSON ou Raw (par défaut)



Usage du web service

- Précision au niveau de la ville :

http://api.ipinfodb.com/v3/ip-city/?key=<api_key>&ip=<ip>

- Précision au niveau du pays :

http://api.ipinfodb.com/v3/ip-country/?key=<api_key>&ip=<ip>

API parameters

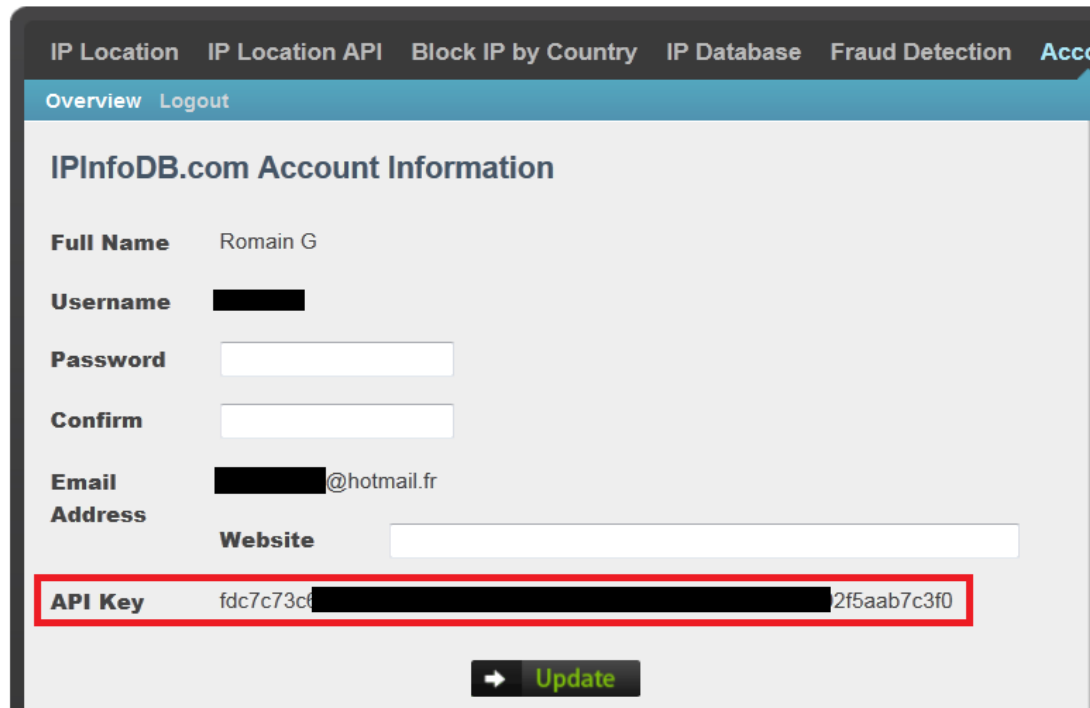
Parameter	Required	Default	Value
key	Yes	<empty>	API key provided with your free account.
ip	No	Client IP	IP address
format	No	raw	raw, xml, json
callback	No	<empty>	Required when using json callback.



Maintenant, trouvons une clé...

- But de la clé :
 - Identifier les utilisateurs du WS
 - Contrôler le nombre de requêtes

1. Création d'un compte : <http://ipinfodb.com/register.php>
2. La clé est disponible dans votre compte



The screenshot shows the 'IPInfoDB.com Account Information' page. The page has a dark header with navigation links: 'IP Location', 'IP Location API', 'Block IP by Country', 'IP Database', 'Fraud Detection', and 'Acco'. Below the header is a blue bar with 'Overview' and 'Logout' links. The main content area is titled 'IPInfoDB.com Account Information' and contains several fields: 'Full Name' (Romain G), 'Username' (redacted), 'Password' (input field), 'Confirm' (input field), 'Email Address' (redacted@hotmail.fr), and 'Website' (input field). At the bottom, the 'API Key' is displayed as 'fdc7c73ct[redacted]2f5aab7c3f0', highlighted with a red border. Below the API key is an 'Update' button with a right arrow icon.



Tester le WS manuellement

- Trouvez l'adresse IP de votre université / établissement
- Entrez l'adresse dans votre navigateur :

<http://api.ipinfodb.com/v3/ip-city/?key=<cle>&ip=178.237.110.205>

```
C:\Windows\system32\cmd.exe
Microsoft Windows [version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Tous droits réservés.

C:\Users\Romain>ping www.inra.fr

Envoi d'une requête 'ping' sur www.inra.fr [178.237.110.205] avec 32 octets de données :
Réponse de 178.237.110.205 : octets=32 temps=52 ms TTL=49
Réponse de 178.237.110.205 : octets=32 temps=51 ms TTL=49
Réponse de 178.237.110.205 : octets=32 temps=52 ms TTL=49
Réponse de 178.237.110.205 : octets=32 temps=54 ms TTL=49

Statistiques Ping pour 178.237.110.205:
    Paquets : envoyés = 4, reçus = 4, perdus = 0 (perte 0%),
    Durée approximative des boucles en millisecondes :
        Minimum = 51ms, Maximum = 54ms, Moyenne = 52ms

C:\Users\Romain>_
```

```
{
  "statusCode" : "OK",
  "statusMessage" : "",
  "ipAddress" : "178.237.110.205",
  "countryCode" : "FR",
  "countryName" : "FRANCE",
  "regionName" : "RHONE-ALPES",
  "cityName" : "GRENOBLE",
  "zipCode" : "-",
  "latitude" : "45.1667",
  "longitude" : "5.71667",
  "timeZone" : "+01:00"
}
```



Systeme d'exploitation Android



Qu'est-ce qu'Android ?

- Un système d'exploitation mobile
- Maintenu par Google depuis 2007
- Open source, basé sur un noyau Linux
- Langage de développement : Java

- Pour :
 - Smartphones
 - Tablettes
 - Lunettes (Glass)
 - Media players (Nexus Q)
 - TV
 - Auto-radios...





Android... sur quels téléphones ?

- Multitude de smartphones
- Chaque fabricant peut créer un smartphone Android
 - Android Compatibility Definition Document (CDD)

htc

SONY®

acer

lenovo



MOTOROLA



nexus





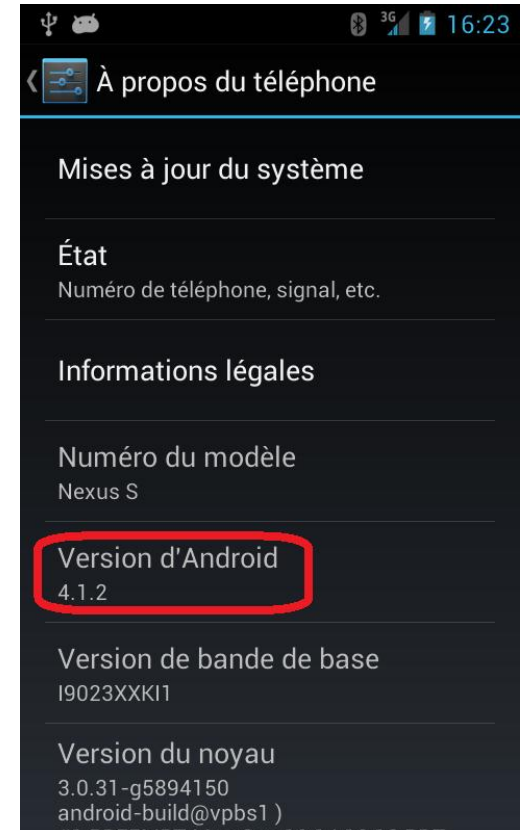
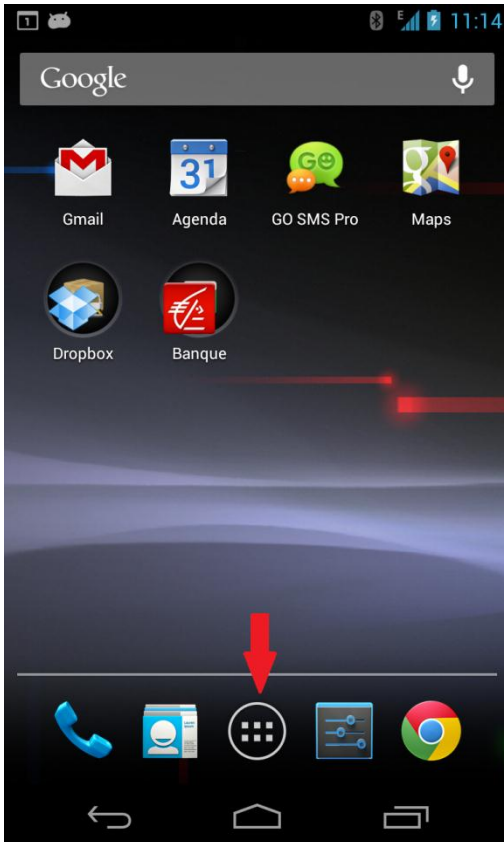
Dominateur du marché mondial

OS	Part marché Q2 12	Part marché Q2 13	Progression nb unités vendues
Android	69.1%	79.3%	73.5%
iOS	16.6%	13.2%	20.0%
Windows Phone	3.1%	3.7%	77.6%
BlackBerry	4.9%	2.9%	-11.7%
Linux	1.8%	0.8%	-35.7%
Symbian	4.2%	0.2%	-92.3%
Autres	0.2%	0.0%	-100.0%
Total	100.0%	100.0%	51.3%

Source : IDC Worldwide Mobile Phone Tracker, August 7, 2013
<http://tinyurl.com/lzglj9w>



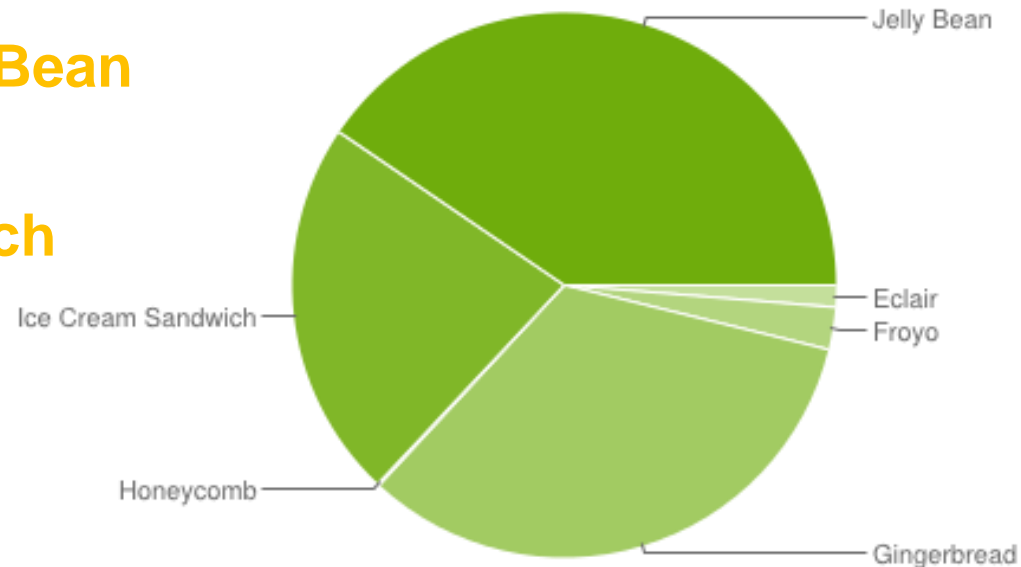
Quelle version avez-vous ?





Versions en circulation

- 2012-3 : **4.1, 4.2, 4.3 Jelly Bean**
- 2011 :
 - **4.0 Ice Cream Sandwich**
 - 3.0 Honeycomb
- 2010 :
 - **2.3 Gingerbread**
 - 2.2 Froyo
- 2009 :
 - 2.0 Eclair
 - 1.6 Donut
 - 1.5 Cupcake
- 2008 : 1.0



Au 1^{er} août 2013

Source : <http://developer.android.com/about/dashboards>



Quel IDE utiliser ?

- Eclipse : Plugin Android Developer Tools (ADT)
- IntelliJ IDEA : Android Studio (Google I/O 2013)
 - Early version (0.1)
- Software Development Kit
- Bundle proposé par Google :
 - Eclipse version Google
 - SDK intégré



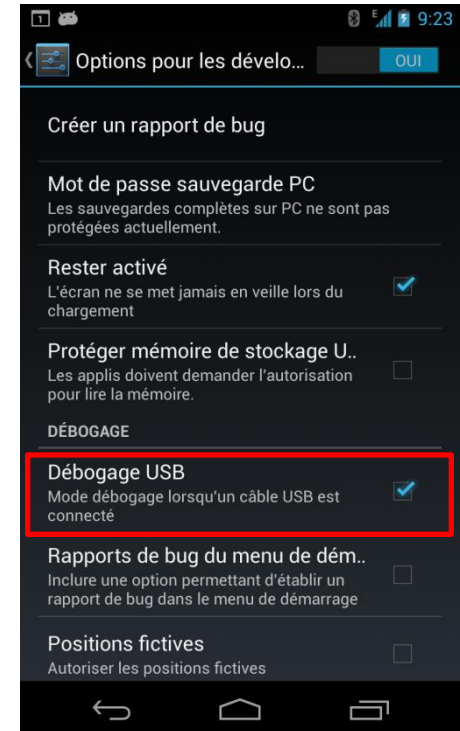
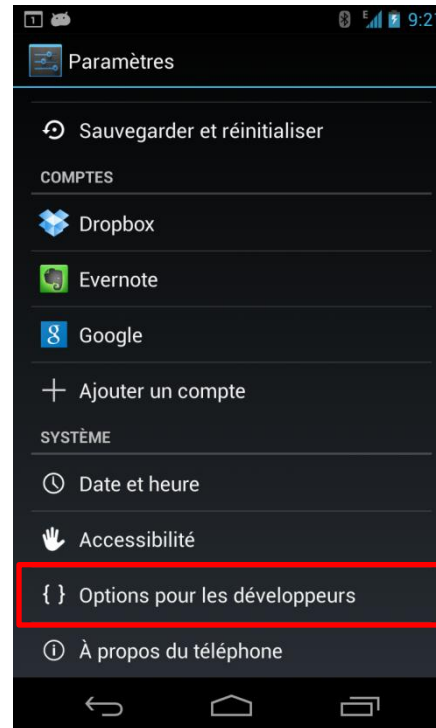
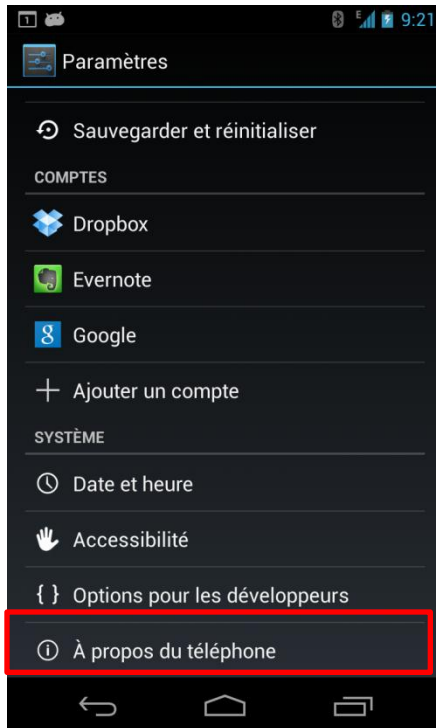


Configuration de l'environnement de développement



Configuration de l'environnement

1. Installer le JDK (Java Development Kit)
2. Dézipper le bundle
3. Activer le mode développeur





Configuration de l'environnement

4. Connecter votre téléphone en USB, installer le driver et vérifier avec la commande « adb devices »

```
C:\WINDOWS\system32\cmd.exe

D:\Android_SDK\platform-tools>adb devices
List of devices attached
4df183a85ef48f85      device

D:\Android_SDK\platform-tools>_
```

5. Lancer Eclipse (eclipse/eclipse.exe)
=> Choisir un workspace (répertoire de travail)



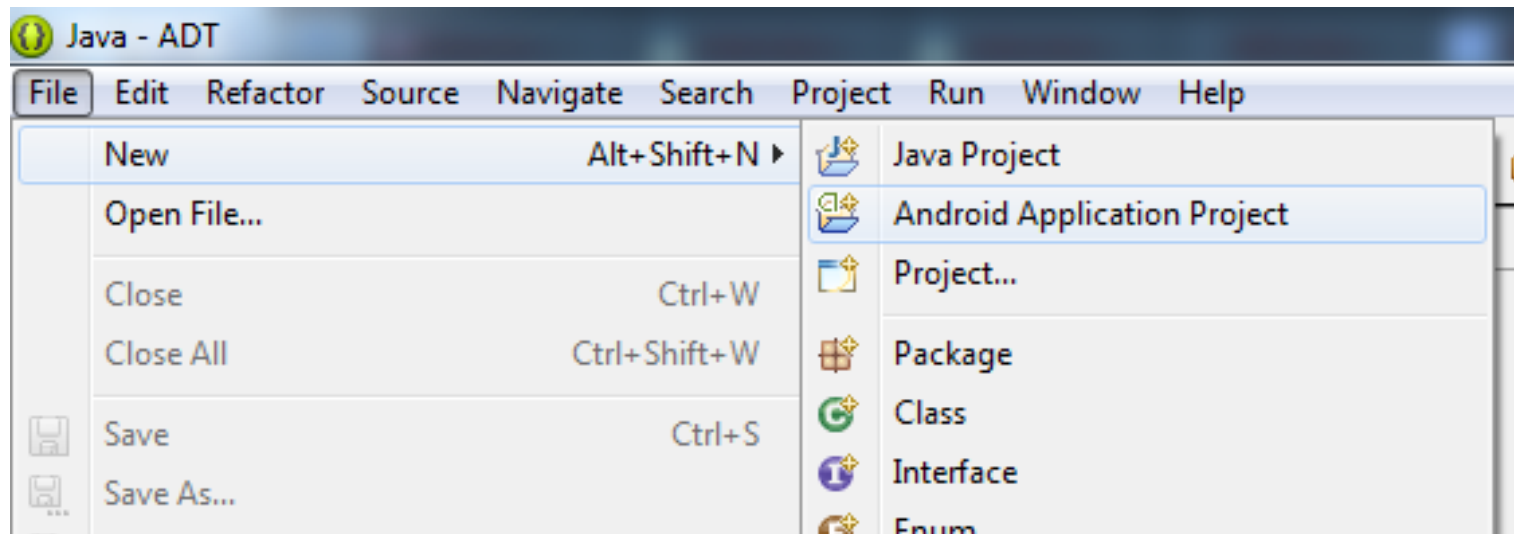
<http://developer.android.com/sdk/index.html>



Création d'un projet Android



Création d'un projet





Création du projet

New Android Application

New Android Application

The prefix 'com.example.' is meant as a placeholder and should not be used

Application Name:

Project Name:

Package Name:

Minimum Required SDK:

Target SDK:

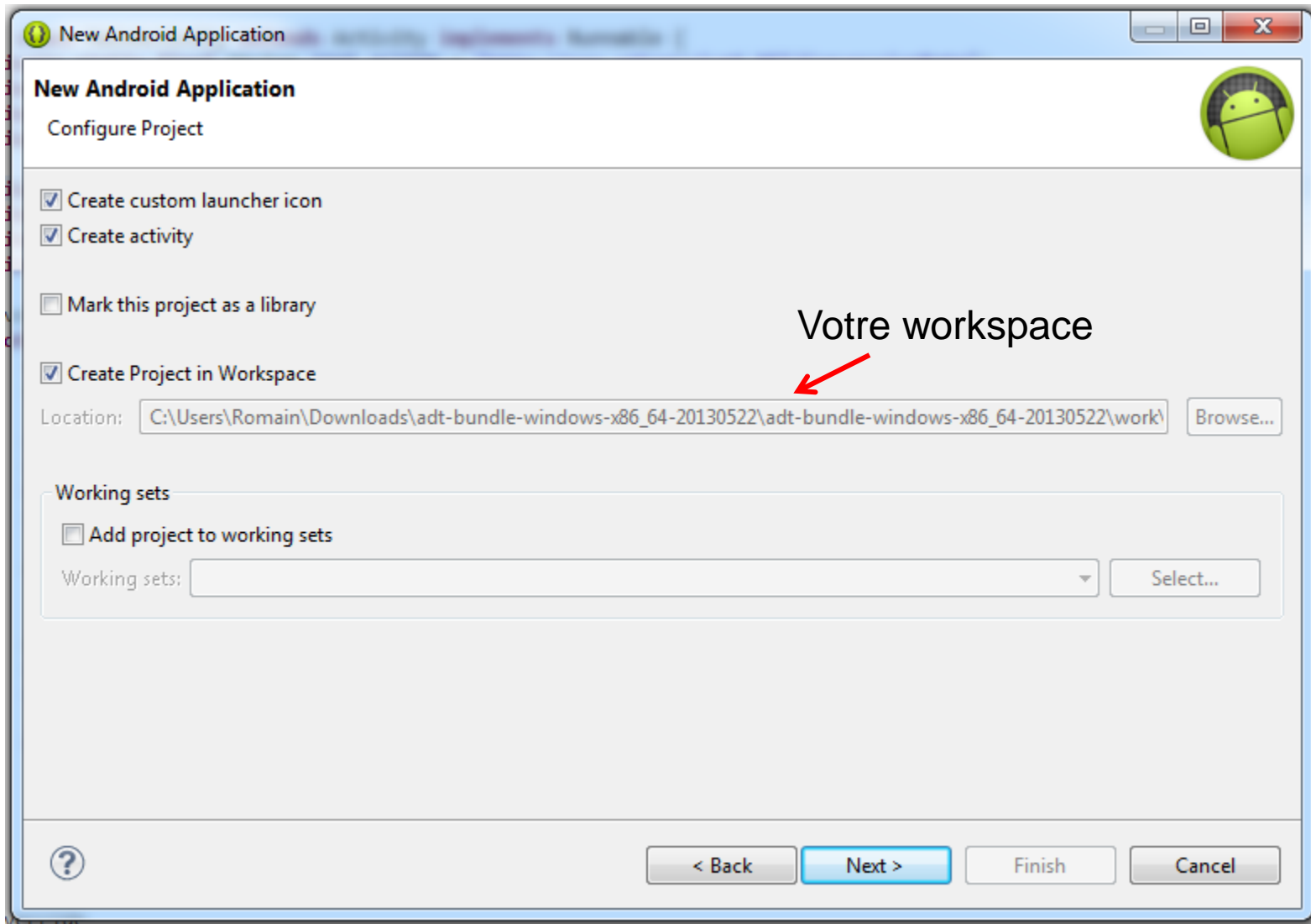
Compile With:

Theme:

Choose the base theme to use for the application

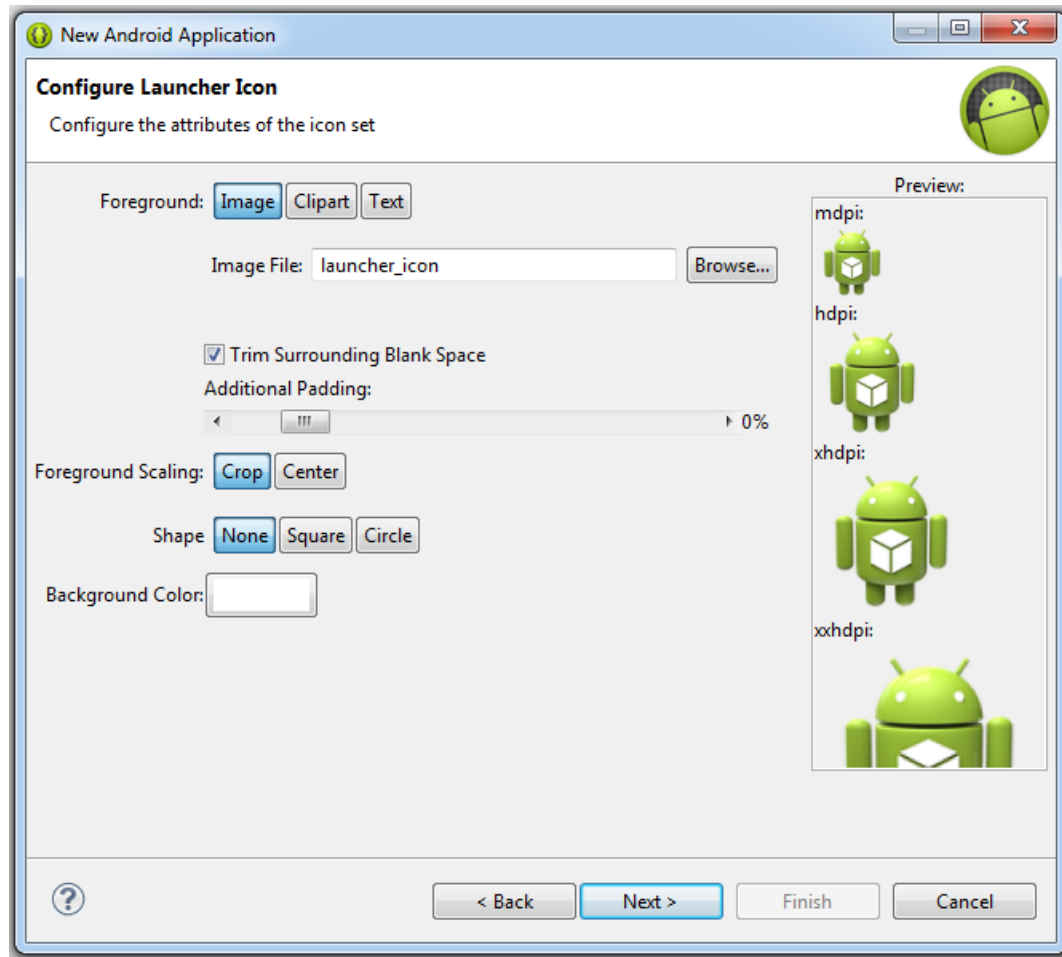


Création du projet



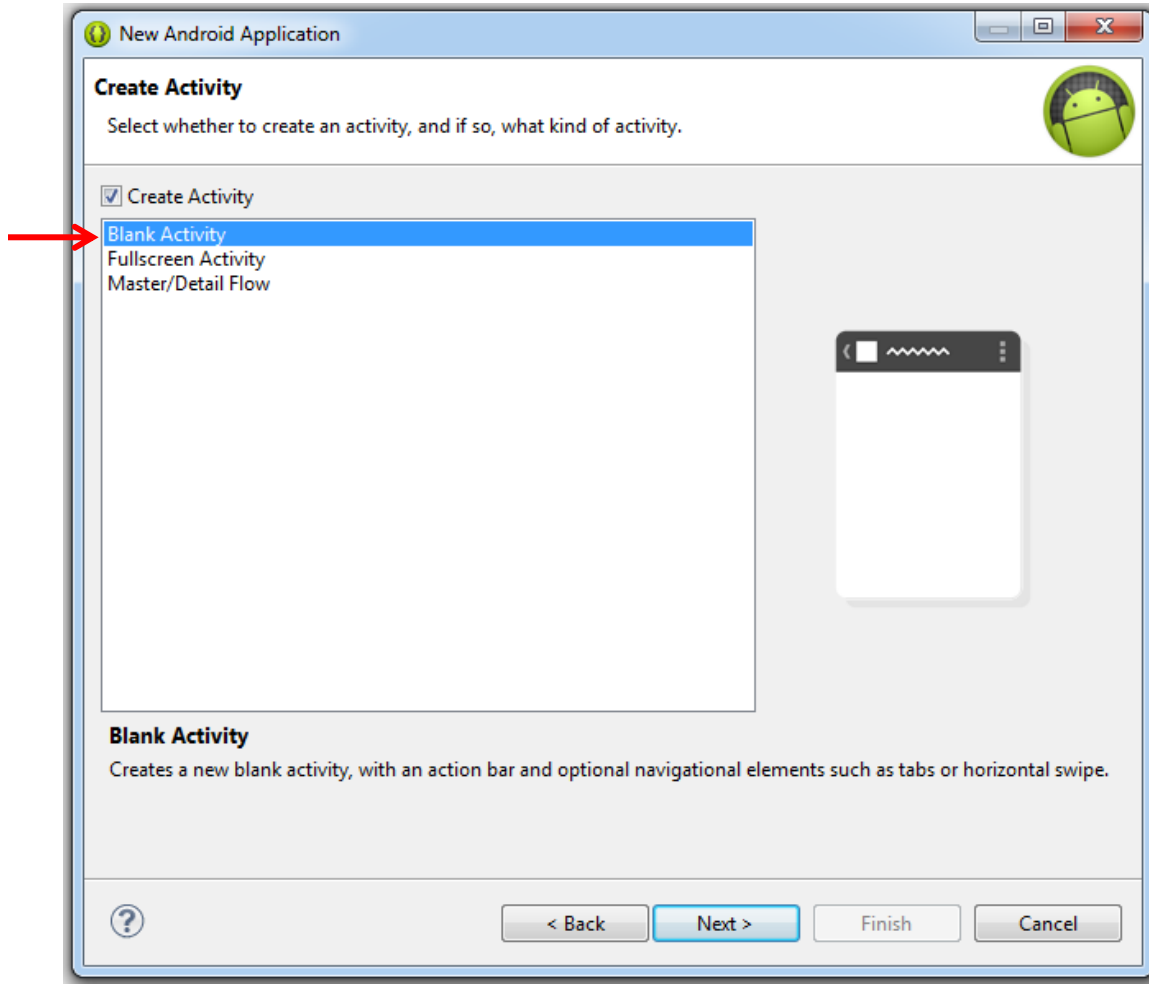


Création du projet



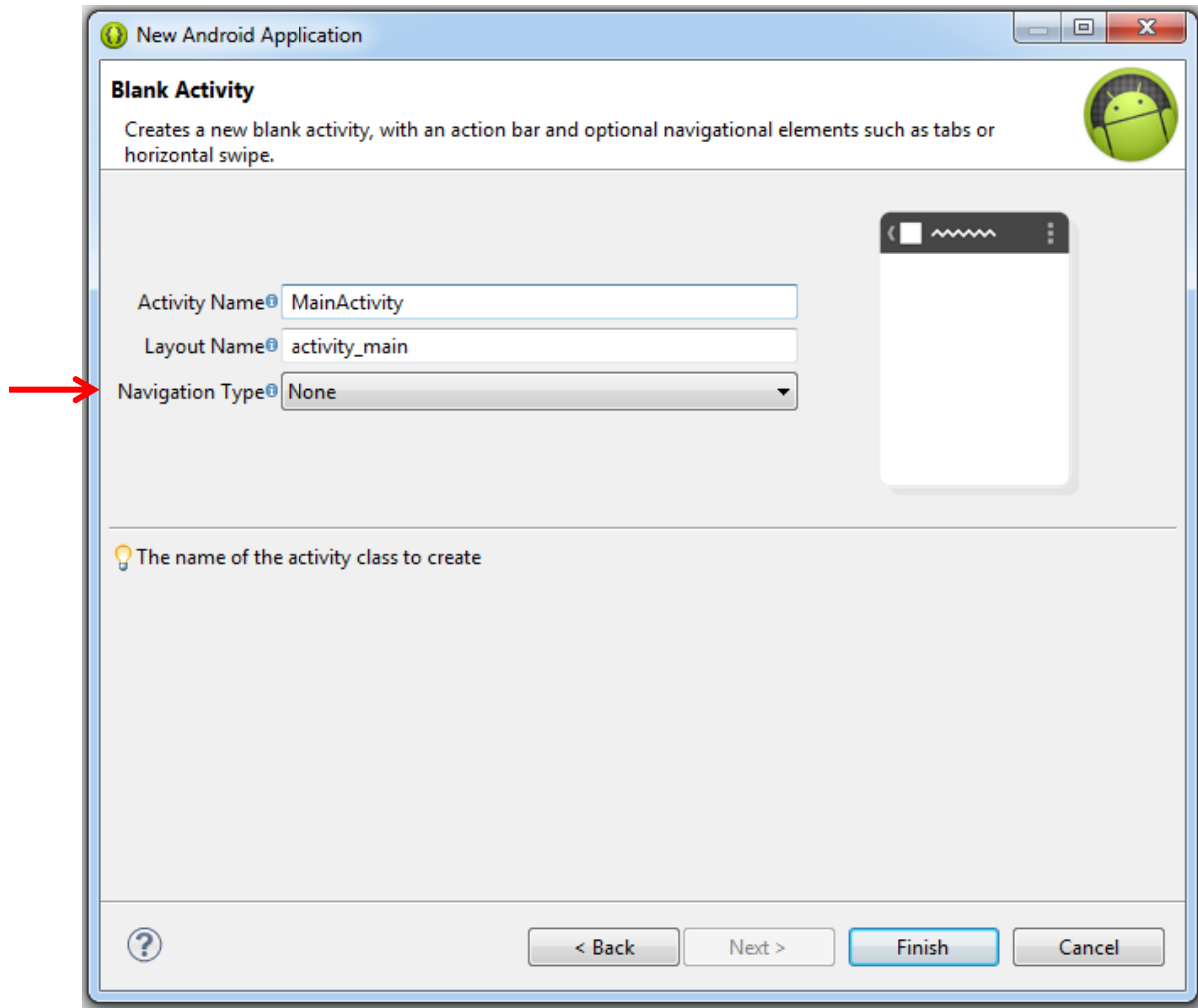


Création du projet





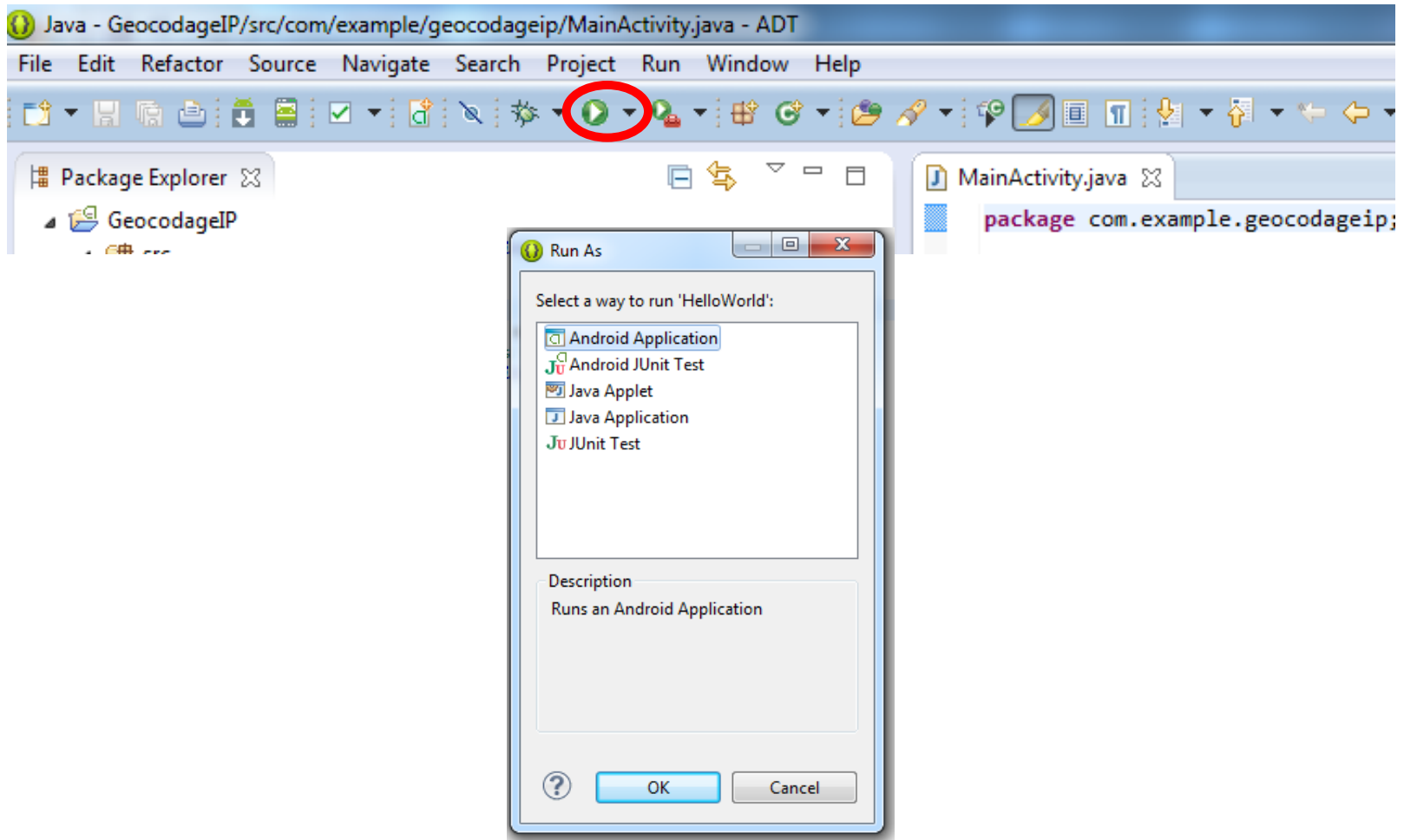
Création du projet





Compilation & lancement du projet

- Tentez de lancer le projet pour vérifier que tout fonctionne



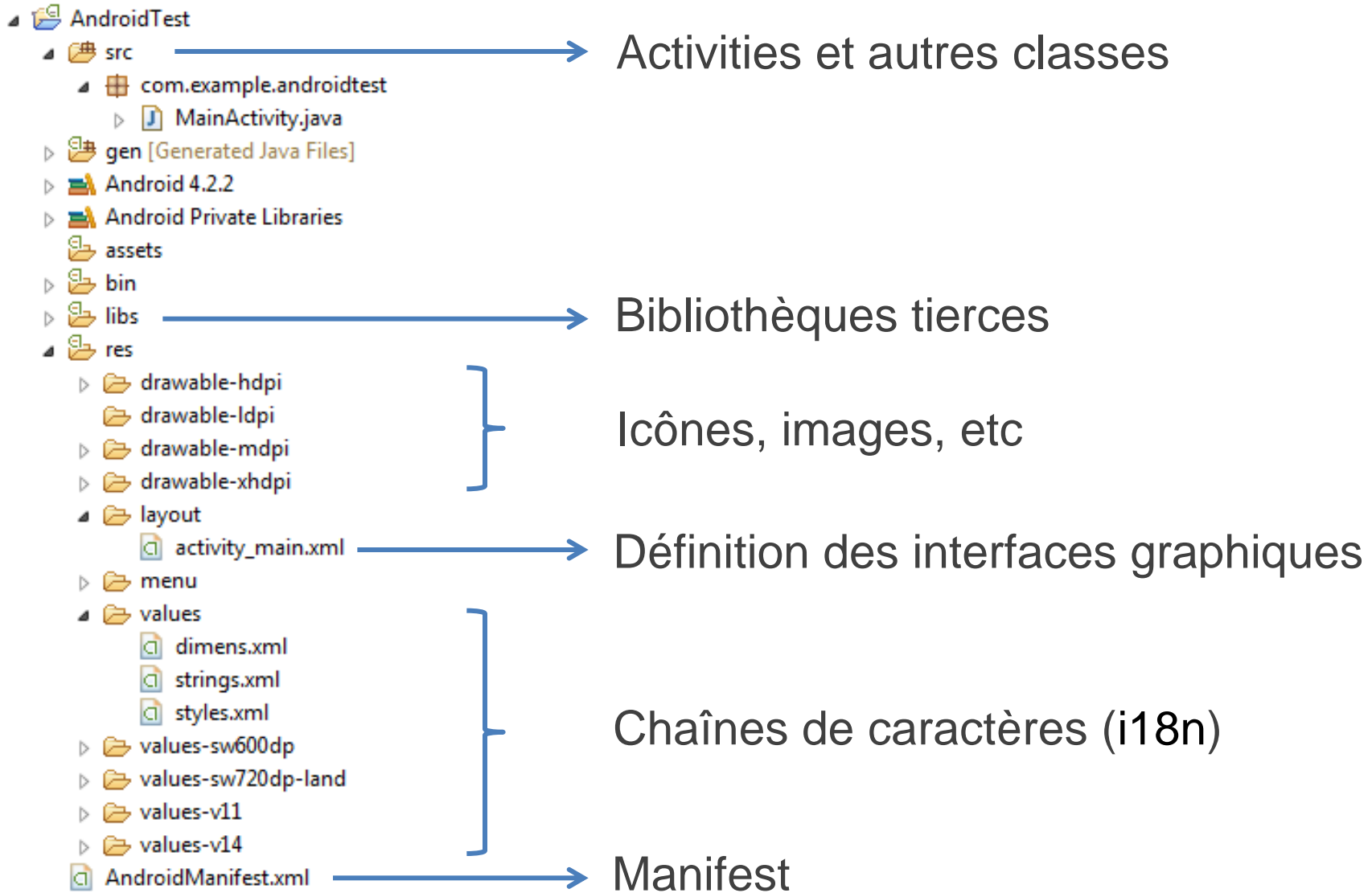


Un peu de vocabulaire...

- **Activity** : Une activity = une fenêtre graphique. C'est là que sont gérés les composants et les événements.
- **Manifest** :
 - Déclaration des Activities
 - Demande de permissions (Internet, appels, SMS...)
 - Version minimale d'Android
 - ...



Structure d'un projet





Les interfaces graphiques

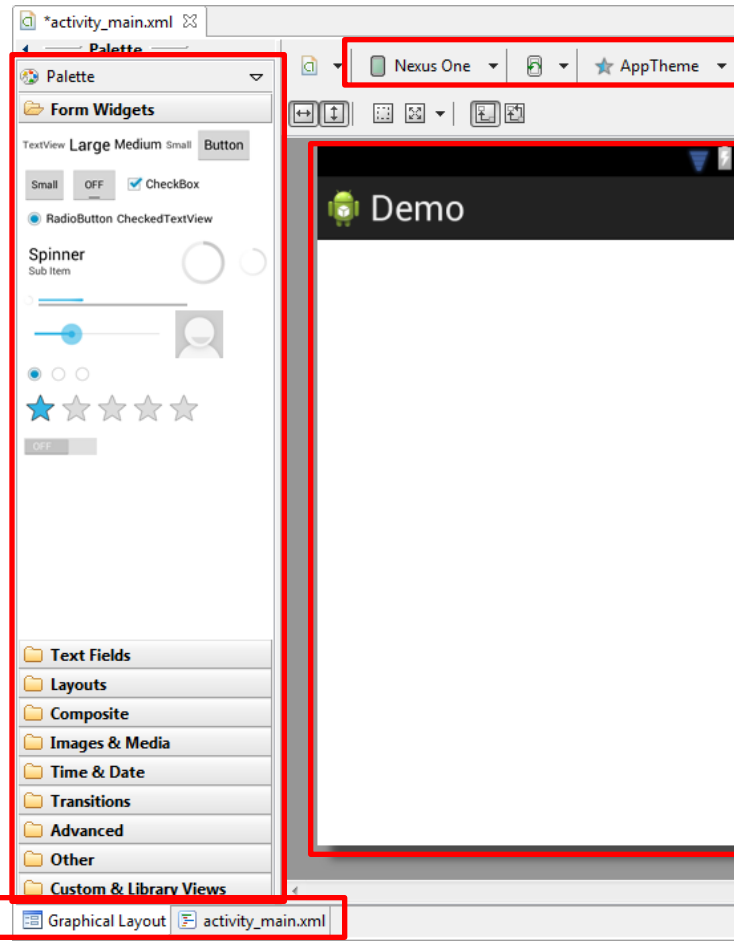
- Gestion par fichiers XML
- Editeur graphique disponible dans Eclipse

Composants

Aides à la conception

Vue graphique

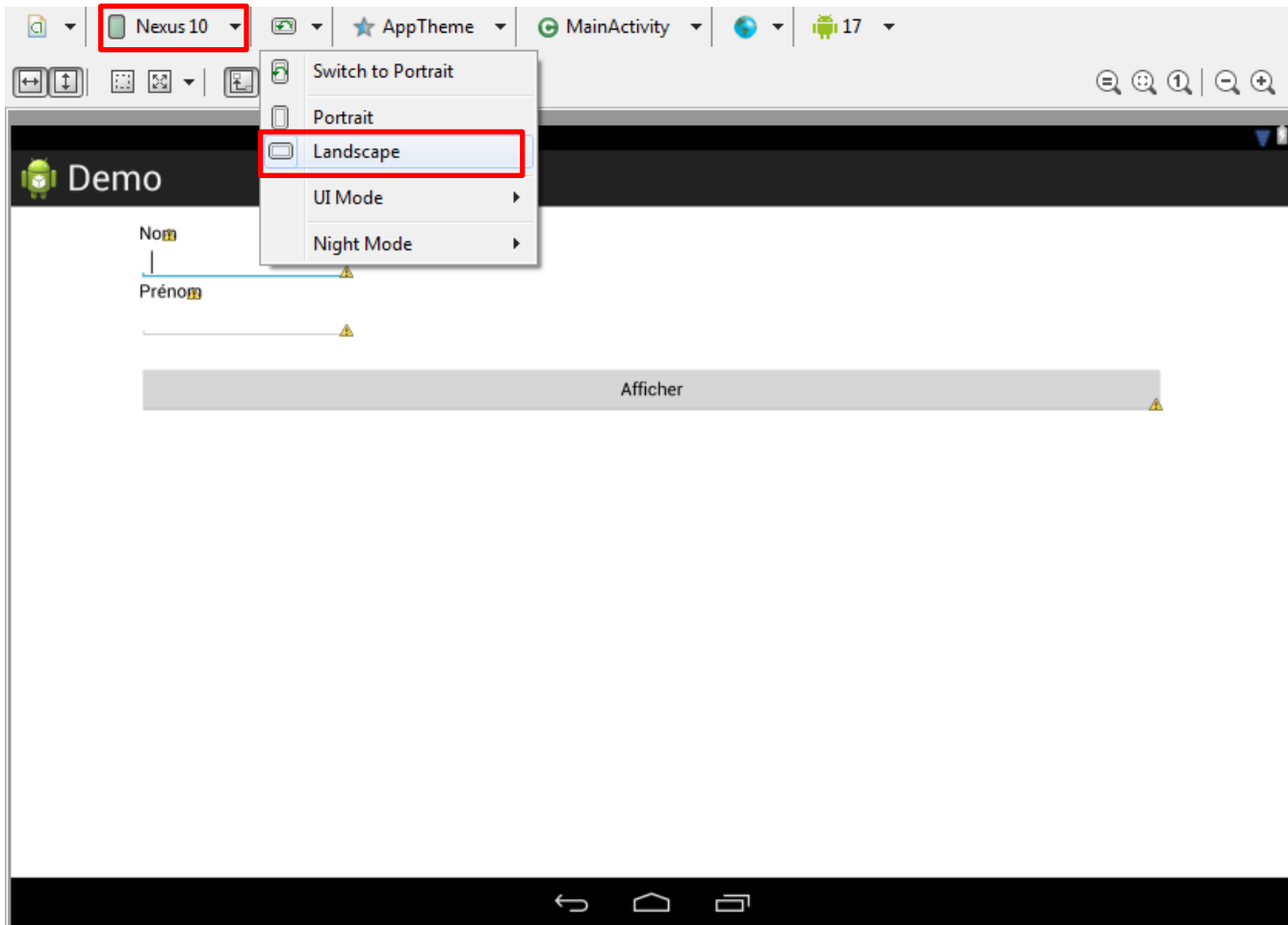
Editeur / XML





Les interfaces graphiques

- Aides à la conception très pratiques

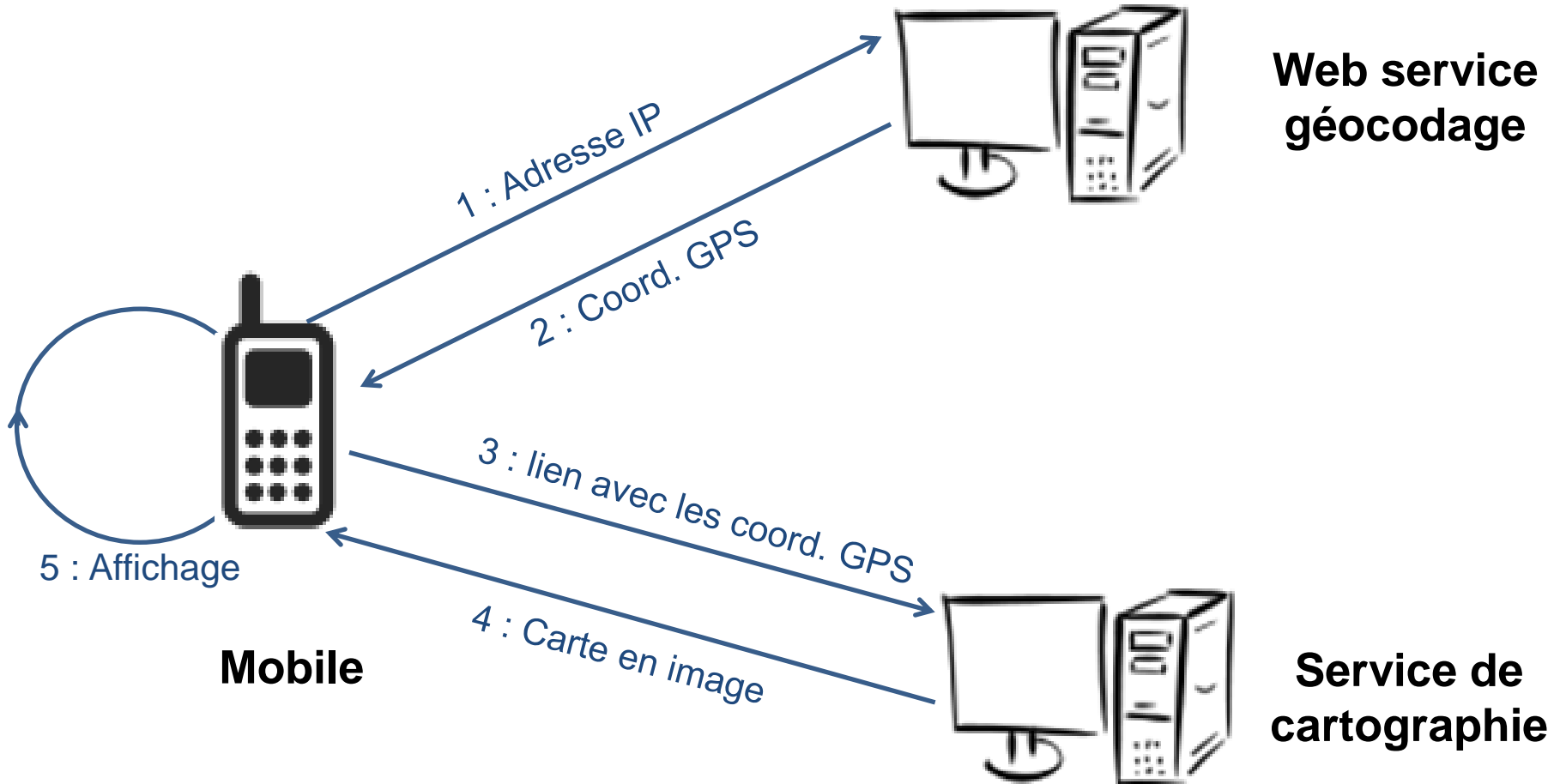




Utilisation d'un web service de géocodage d'adresses IP (REST)



Conception





Web service de géocodage : IPInfoDB

- Précision au niveau de la ville :

http://api.ipinfodb.com/v3/ip-city/?key=<api_key>&ip=<ip>

- Précision au niveau du pays :

http://api.ipinfodb.com/v3/ip-country/?key=<api_key>&ip=<ip>

API parameters

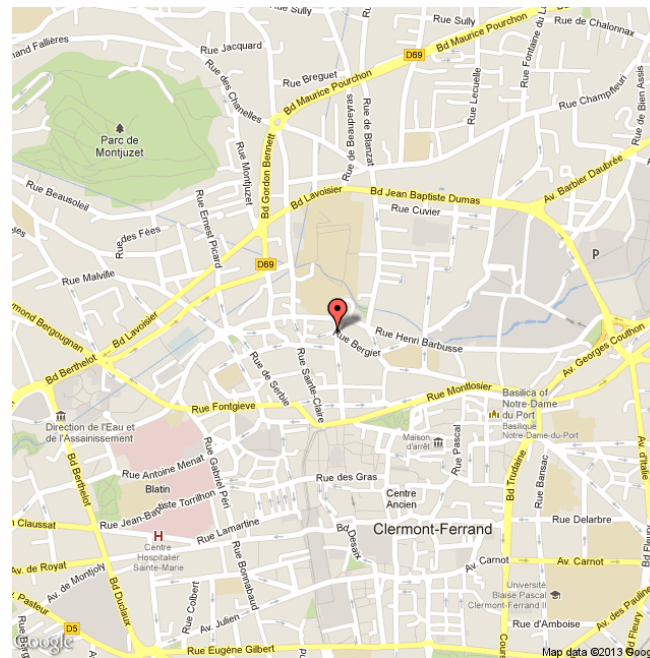
Parameter	Required	Default	Value
key	Yes	<empty>	API key provided with your free account.
ip	No	Client IP	IP address
format	No	raw	raw, xml, json
callback	No	<empty>	Required when using json callback.



Service de cartographie : Google Static Maps

- Donnée en entrée : position GPS (exemple : 45.783, 3.083)
- Réponse : carte sous forme d'image

<http://maps.googleapis.com/maps/api/staticmap?center=45.783,3.083&zoom=15&size=400x400&sensor=false&markers=color:red|45.783,3.083>





Création de l'interface

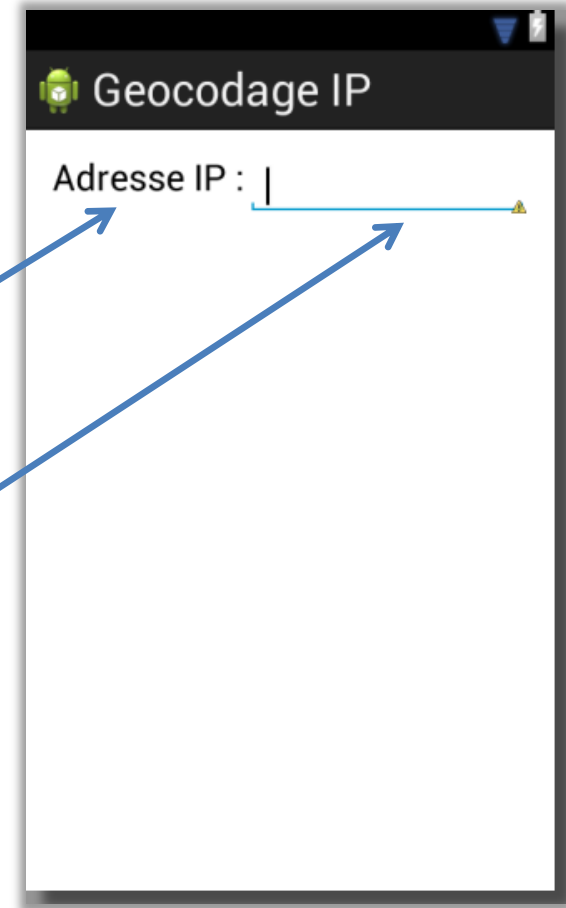
- Ajout des 2 premiers éléments (res/layout/activity_main.xml)

```
→ <RelativeLayout  
    xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:tools="http://schemas.android.com/tools"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:paddingBottom="@dimen/activity_vertical_margin"  
    android:paddingLeft="@dimen/activity_horizontal_margin"  
    android:paddingRight="@dimen/activity_horizontal_margin"  
    android:paddingTop="@dimen/activity_vertical_margin">
```

```
→ <TextView  
    android:id="@+id/textView1"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_alignParentLeft="true"  
    android:layout_alignParentTop="true"  
    android:text="@string/iplabel"  
    android:textAppearance="?android:attr/textAppearanceLarge" />
```

```
→ <EditText  
    android:id="@+id/editText1"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_alignParentRight="true"  
    android:layout_alignTop="@+id/textView1"  
    android:ems="8" >  
  
    <requestFocus />  
</EditText>
```

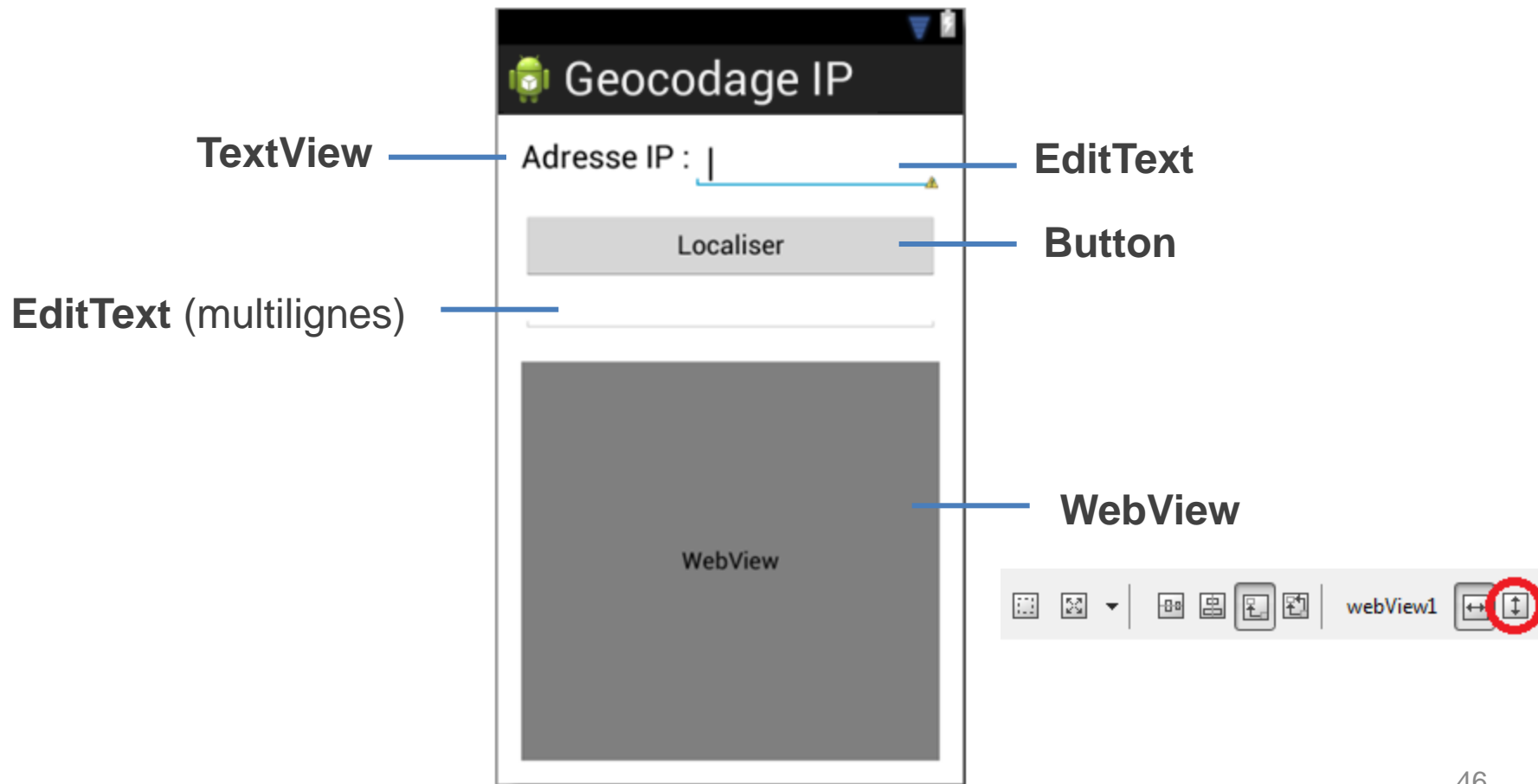
```
</RelativeLayout>
```





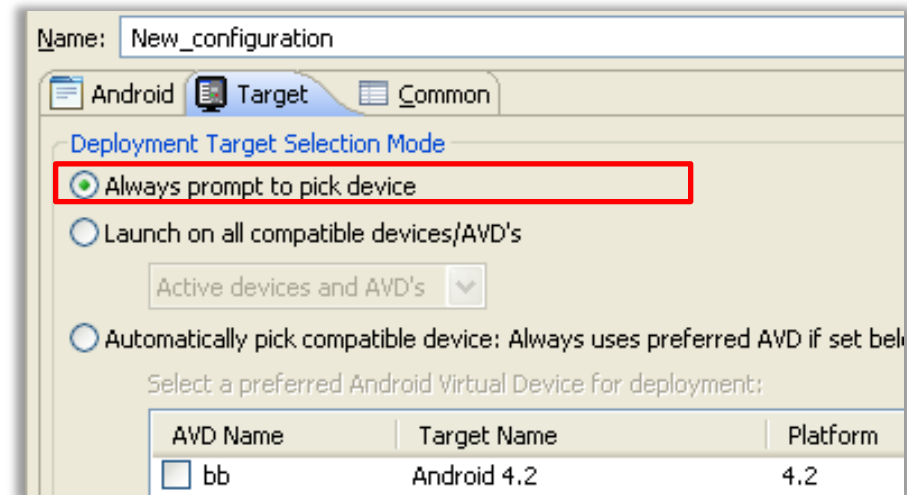
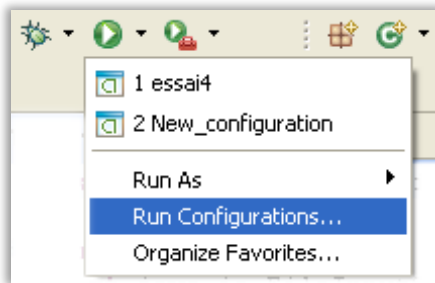
Création de l'interface

- Interface de l'appli qui utilisera le WS de géocodage d'adresse IP





- Configurer le mode de déploiement



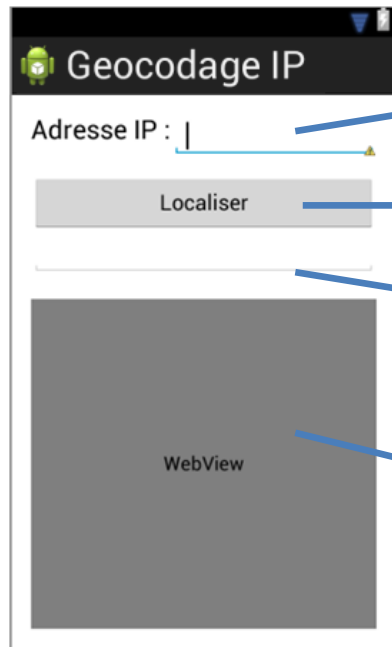
- Compiler et lancer l'application sur votre mobile

L'application avec l'interface graphique (non fonctionnelle pour l'instant)

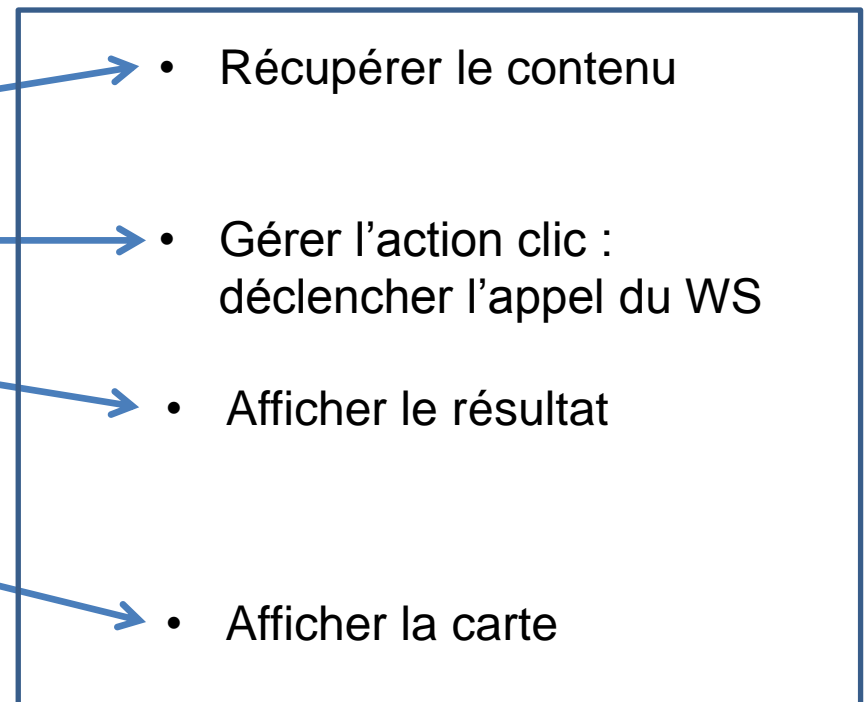


Code pour articuler l'interface

- Étape 1 : connecter les composants graphiques au code
- Étape 2 : réaliser les fonctionnalités



Activity_main.xml



MainActivity.java



Connexion des composants

- Créer un attribut pour chaque élément graphique (sauf le TextView)

```
public class MainActivity extends Activity {  
  
    private EditText    ipAddress;  
    private EditText    serverResponse;  
    private Button      btn;  
    private WebView     map;  
  
}
```

- Récupérer les composants lors de la construction de l'Activity :

```
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
  
    ipAddress      = (EditText)findViewById(R.id.editText1);  
    serverResponse = (EditText)findViewById(R.id.editText2);  
    btn            = (Button)findViewById(R.id.button1);  
    map            = (WebView)findViewById(R.id.webView1);  
}
```



Gérer l'action clic

- Créer un listener de clics sur le bouton (toujours dans onCreate())

```
btn.setOnClickListener(new View.OnClickListener() {  
    public void onClick(View v) {  
        // code ...  
    }  
});
```

- **Les communications réseau ne doivent pas se faire dans le thread principal !**

- Astuce simple : créer un thread basique

```
btn.setOnClickListener(new View.OnClickListener() {  
    public void onClick(View v) {  
        new Thread(MainActivity.this).start();  
    }  
});
```

```
...  
@Override  
public void run() {  
    // Appel du web service  
}
```

C'est ici que se fera
l'appel au WS via HTTP



Requête GET

1. Construction de l'URL

```
@Override
public void run() {
    String key = "<votre_cle>";
    String url = "http://api.ipinfodb.com/v3/ip-city/?format=json&key=" + key + "&ip=" + ipAddress.getText();
}
```

2. Envoi de la requête et réception du message

```
URLConnection connection = null;
BufferedReader response;
String line;
StringBuffer json = new StringBuffer();

try {
    connection = (URLConnection)new URL(url).openConnection(); → Envoi
    response = new BufferedReader(new InputStreamReader(connection.getInputStream())); → Réception

    while ((line = response.readLine()) != null) → Lecture et stockage
        json.append(line);

    response.close();

    handleServerResult(json.toString()); → Traitement
}
catch (Exception e) {
    Log.e(MainActivity.class.getName(), "An error occurred", e);
}
finally {
    connection.disconnect();
}
```



Traitement de la réponse

- Format demandé : JSON. Utilisation de `JSONObject`

```
        {
            "statusCode" : "OK"
private void handleServerResult(String json) {
    try {
        Log.i(MainActivity.class.getName(), "Reponse : " + json);

        JSONObject jsonObject = new JSONObject(json);

        String statusCode = jsonObject.getString("statusCode");
        → if (statusCode.equals("OK"))
            {
                // Traiter la réponse
            }
        → else
            Log.e(MainActivity.class.getName(), "Une erreur est survenue : " + jsonObject.getString("statusMessage"));
    } catch (JSONException e) {
        Log.e(MainActivity.class.getName(), "Une erreur est survenue : " + e);
    }
}
        }
    }
```

Lien vers le code : <http://www.isima.fr/~lacomme/ateliers/t8a2/?idx=2>



Traitement de la réponse

```
if (statusCode.equals("OK"))
{
    double latitude    = jsonObject.getDouble("latitude");
    double longitude   = jsonObject.getDouble("longitude");

    final StringBuilder display = new StringBuilder();
    display.append("IP address: ").append(jsonObject.getString("ipAddress")).append("\n");
    display.append("countryName: ").append(jsonObject.getString("countryName")).append("\n");
    display.append("regionName: ").append(jsonObject.getString("regionName")).append("\n");
    display.append("cityName: ").append(jsonObject.getString("cityName")).append("\n");
    display.append("latitude: ").append(latitude).append("\n");
    display.append("longitude: ").append(longitude).append("\n");

    displayMap(latitude, longitude);
    serverResponse.post(new Runnable() {
        @Override
        public void run() {
            serverResponse.setText(display.toString());
        }
    });
}
```

→ Affichage de la position dans la web view

→ Affichage des infos Renvoyées par le WS

Lien vers le code : <http://www.isima.fr/~lacomme/ateliers/t8a2/?idx=3>

- **La manipulation des composants ne peut se faire que dans le thread principal**



Affichage dans la WebView

- Utilisation du web service proposé par Google
- Données en entrée : les coordonnées GPS (latitude, longitude)

```
private void displayMap(double latitude, double longitude) {  
    String coord = latitude + "," + longitude;  
    final StringBuilder url = new StringBuilder("http://maps.googleapis.com/maps/api/staticmap?center=");  
    url.append(coord).append("&zoom=15&size=300x300&sensor=false&markers=color:red|").append(coord);  
  
    map.post(new Runnable() {  
        @Override  
        public void run() {  
            map.loadUrl(url.toString());  
  
            InputMethodManager imm = (InputMethodManager) getSystemService(Context.INPUT_METHOD_SERVICE);  
            imm.hideSoftInputFromWindow(serverResponse.getWindowToken(), 0);  
        }  
    });  
}
```

→ Formation de l'URL

→ Chargement

→ Cacher le clavier après saisie

Lien vers le code : <http://www.isima.fr/~lacomme/ateliers/t8a2/?idx=4>



Permission d'accès à Internet

- L'application aura besoin d'accéder à Internet...
- ... il faut demander la permission

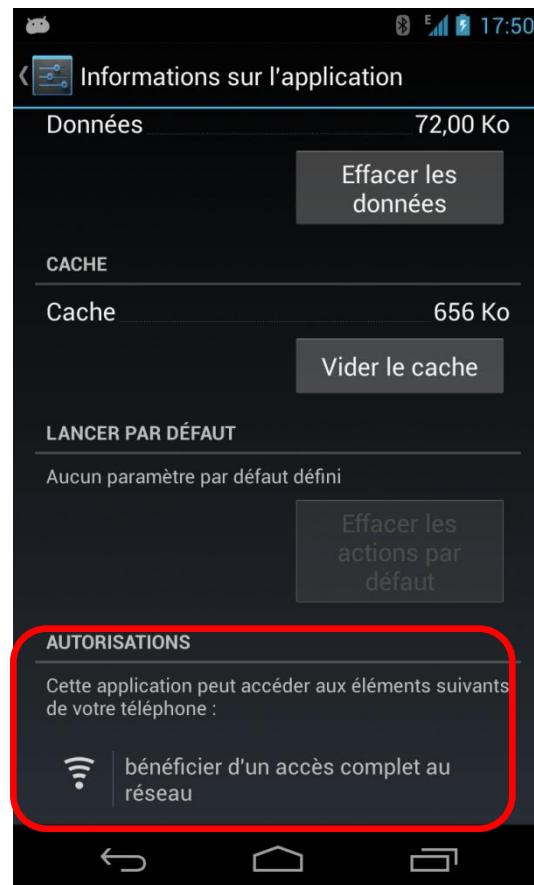
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.helloworld"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="8" android:targetSdkVersion="17" />

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

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
```

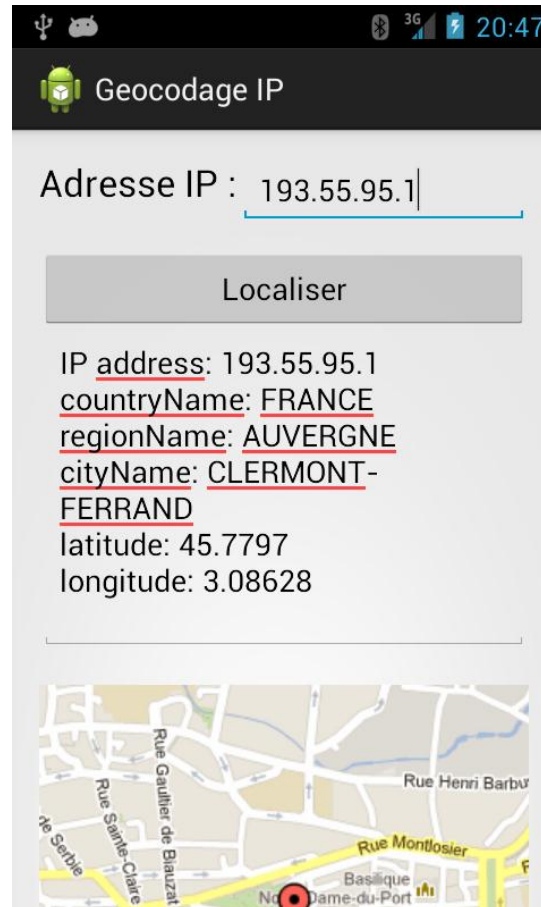
AndroidManifest.xml





Exécution sur le téléphone

- Si la taille de l'écran ne permet pas de visualiser la carte complète, modifier l'interface graphique pour ajouter une ScrollView





Utilisation d'un web service de conversion de devises (SOAP)



Description du web service

- But : calculer le ratio entre deux devises

<http://www.webservicex.net/CurrencyConvertor.asmx?op=ConversionRate>

SOAP 1.2

The following is a sample SOAP 1.2 request and response. The **placeholders** shown need to be replaced with actual values.

```
POST /CurrencyConvertor.asmx HTTP/1.1
Host: www.webservicex.net
Content-Type: application/soap+xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap12="http://www.w3.org/2003/11/soap-envelope">
  <soap12:Body>
    <ConversionRate xmlns="http://www.webserviceX.NET/">
      <FromCurrency>AFA or ALL or DZD or ARS or AWG or AUD or BSD or BHD or BDT or BBD or BZD or BMD or BTN or BOB or BWP or BR
      <ToCurrency>AFA or ALL or DZD or ARS or AWG or AUD or BSD or BHD or BDT or BBD or BZD or BMD or BTN or BOB or BWP or BR
    </ConversionRate>
  </soap12:Body>
</soap12:Envelope>
```

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap12="http://www.w3.org/2003/11/soap-envelope">
  <soap12:Body>
    <ConversionRateResponse xmlns="http://www.webserviceX.NET/">
      <ConversionRateResult>double</ConversionRateResult>
    </ConversionRateResponse>
  </soap12:Body>
</soap12:Envelope>
```



Description du web service

- Remarques :
 - Pas besoin de clé d'API
 - Ce web service est aussi disponible en REST

HTTP GET

The following is a sample HTTP GET request and response. The **placeholders** shown need to be replaced with actual values.

```
GET /CurrencyConvertor.asmx/ConversionRate?FromCurrency=string&ToCurrency=string HTTP/1.1 ←  
Host: www.webserviceX.net
```

```
HTTP/1.1 200 OK  
Content-Type: text/xml; charset=utf-8  
Content-Length: length
```

```
<?xml version="1.0" encoding="utf-8"?>  
<double xmlns="http://www.webserviceX.NET/">double</double>
```



Tester le WS manuellement

- L'interface de test utilise un WS REST

Test

To test the operation using the HTTP POST protocol, click the 'Invoke' button.

Parameter	Value
FromCurrency:	<input type="text" value="EUR"/>
ToCurrency:	<input type="text" value="GBP"/>

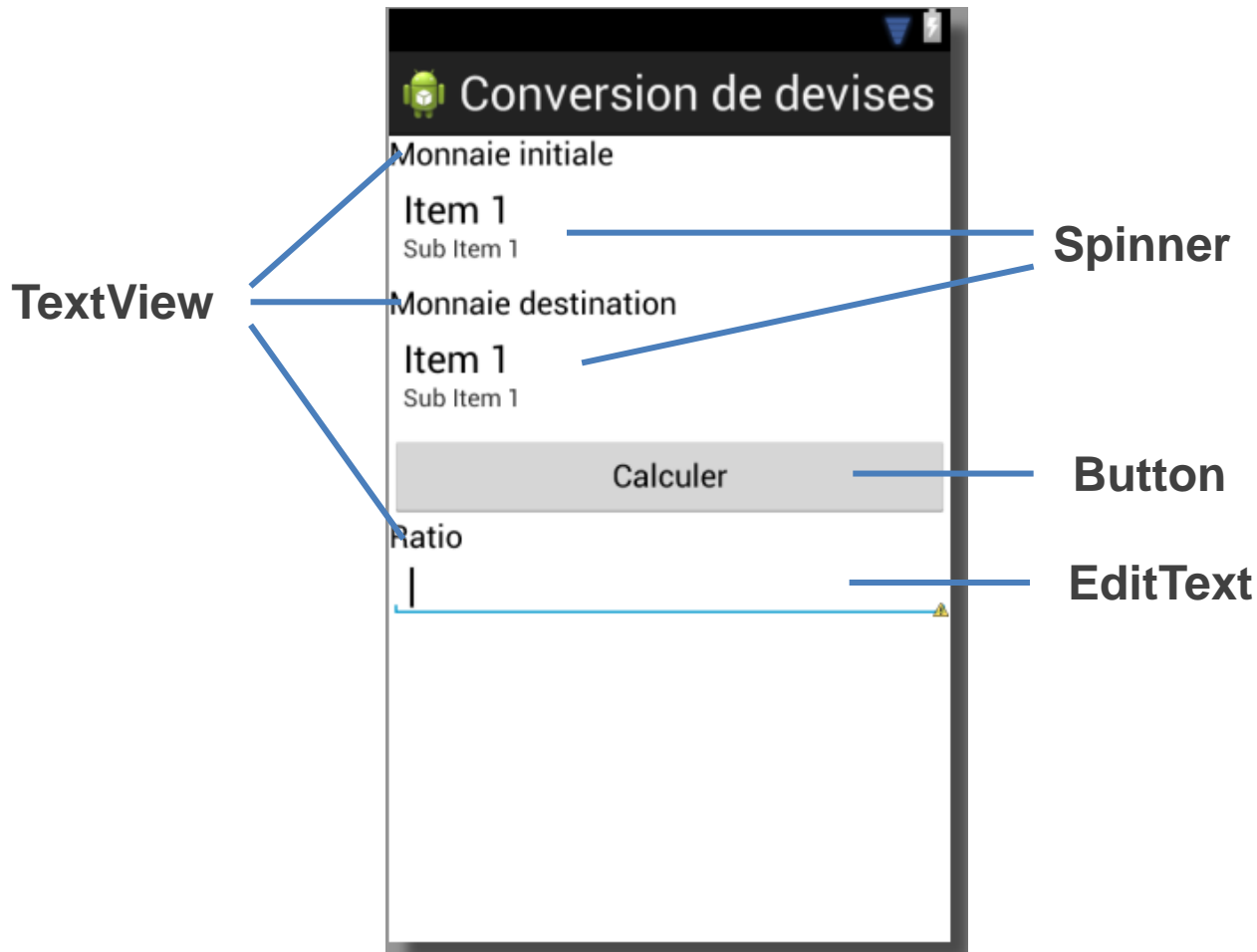
- Résultat au format XML

`<double>0.8445</double>`



Création de l'interface

- Interface proposée (res/layout/activity_main.xml)





Conception de l'Activity

- Créer un attribut pour chaque élément graphique (sauf les TextView)

```
private Spinner    currencyFrom;  
private Spinner    currencyTo;  
private Button     btn;  
private EditText   ratio;
```

- Récupérer les composants lors de la construction de l'Activity :

```
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
  
    currencyFrom    = (Spinner)findViewById(R.id.spinner1);  
    currencyTo      = (Spinner)findViewById(R.id.spinner2);  
    btn             = (Button)findViewById(R.id.button1);  
    ratio           = (EditText)findViewById(R.id.editText1);  
}
```



Remplir les spinners

- Valeurs du spinner = valeurs du fichier currencies.xml
- Déposer le fichier currencies.xml dans res/values

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string-array name="currencies">
    <item>AFA-Afghanistan Afghani</item>
    <item>ALL-Albanian Lek</item>
    <item>DZD-Algerian Dinar</item>
    ...
    <item>ZWD-Zimbabwe Dollar</item>
  </string-array>
</resources>
```

```
ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResource(this,
    R.array.currencies,
    android.R.layout.simple_spinner_item);
adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
currencyFrom.setAdapter(adapter);
currencyTo.setAdapter(adapter);
```



Gérer le bouton

- Là encore, threader
- Créer un listener de clics sur le bouton

```
public class MainActivity extends Activity implements Runnable {  
    ...  
    btn.setOnClickListener(new View.OnClickListener() {  
        public void onClick(View v) {  
            new Thread(new Runnable() {  
                public void run() {  
                    //Appel du web service..  
                }).start();  
            }  
        });  
    }  
    ...  
}
```

C'est ici que se fera l'appel au WS via HTTP

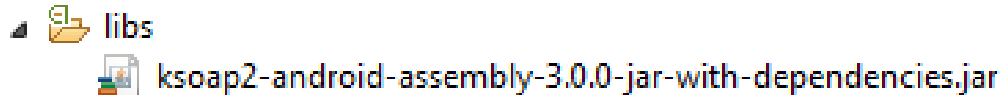


Préparation à l'utilisation du WS

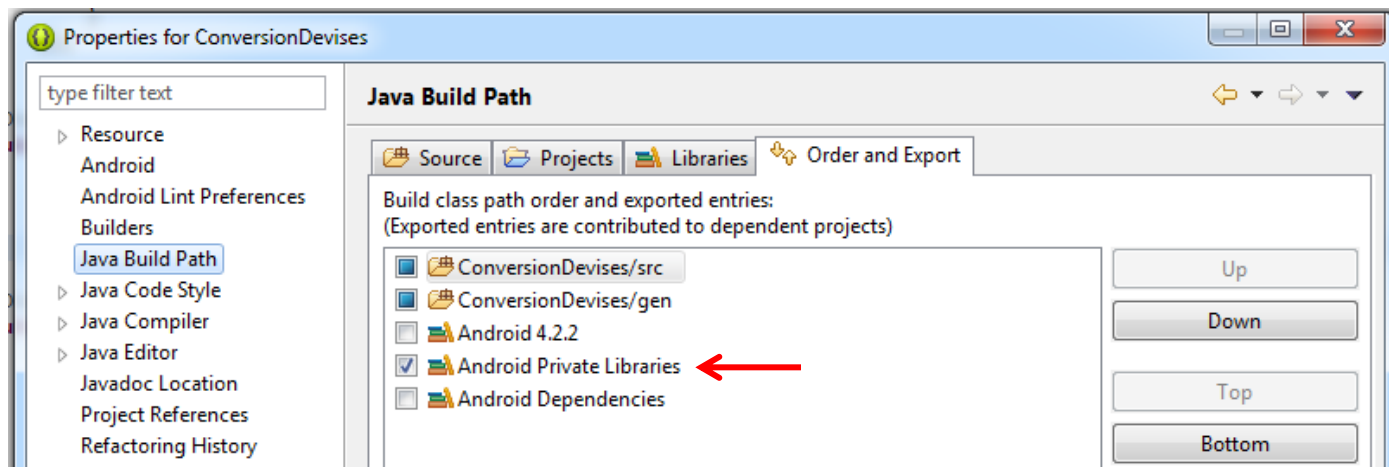
- Définition de constantes : infos à propos du WS

```
private static final String SOAP_ACTION = "http://www.webserviceX.NET/ConversionRate";  
private static final String METHOD_NAME = "ConversionRate";  
private static final String NAMESPACE = "http://www.webserviceX.NET/";  
private static final String URL = "http://www.websvcicex.net/CurrencyConvertor.asmx?WSDL";
```

- Ajout de la bibliothèque ksoap2 au projet



- Vérifier les préférences du projet (Properties > Java Build Path)





Envoi d'une requête au web service

- Création d'un objet SOAP
- Empaquetage des paramètres : les 2 devises
- Création de l'enveloppe (version 1.2)

SOAP 1.2

The following is a sample SOAP 1.2 request and response.

```
POST /CurrencyConvertor.asmx HTTP/1.1
Host: www.webservice.net
Content-Type: application/soap+xml; charset=utf-8
```

```
public void run() {
    SoapObject request = new SoapObject(NAMESPACE, METHOD_NAME);

    request.addProperty("FromCurrency", currencyFrom.getSelectedItem().toString().subSequence(0, 3));
    request.addProperty("ToCurrency", currencyTo.getSelectedItem().toString().subSequence(0, 3));

    SoapSerializationEnvelope envelope = new SoapSerializationEnvelope(SoapEnvelope.VER12);
    envelope.dotNet = true;
    envelope.setOutputSoapObject(request);

    try {
        HttpTransportSE http = new HttpTransportSE(URL);
        http.call(SOAP_ACTION, envelope);

        // TODO Traitement de la réponse
    }
    catch (Exception e) {
        Log.e(MainActivity.class.getName(), "An error occurred", e);
    }
}
```

Lien vers le code : <http://www.isima.fr/~lacomme/ateliers/t8a2/?idx=11>



Traitement de la réponse

- Récupération du résultat
- Automatisée grâce à la description du web service par le fichier WSDL
- Affichage dans l'élément « ratio »

```
final SoapPrimitive result = (SoapPrimitive)envelope.getResponse();
Log.i(MainActivity.class.getName(), "Résultat : " + result);
ratio.post(new Runnable() {
    @Override
    public void run() {
        ratio.setText(result.toString());
    }
});
```

Traitement de la réponse



Encore des permissions

- Là encore, l'application aura besoin d'accéder à Internet...
- Inscrire la permission dans le manifest

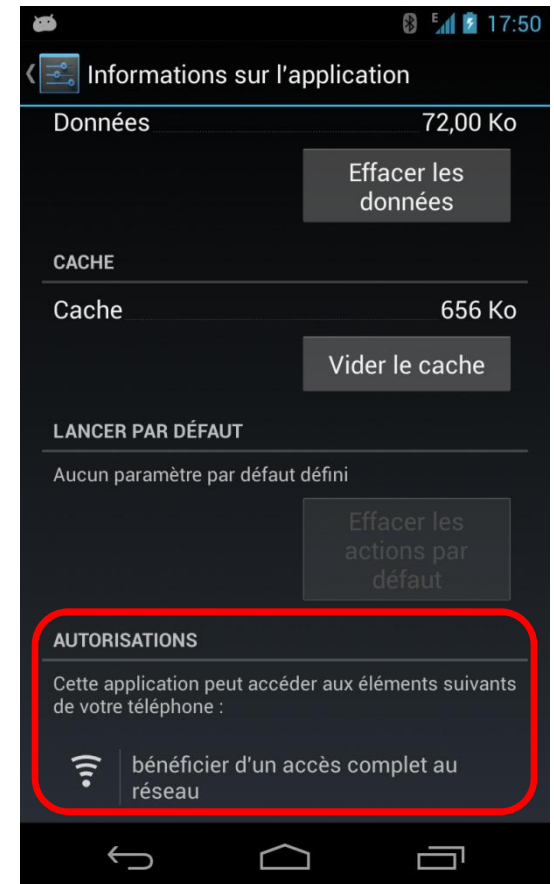
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.helloworld"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="8" android:targetSdkVersion="17" />

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

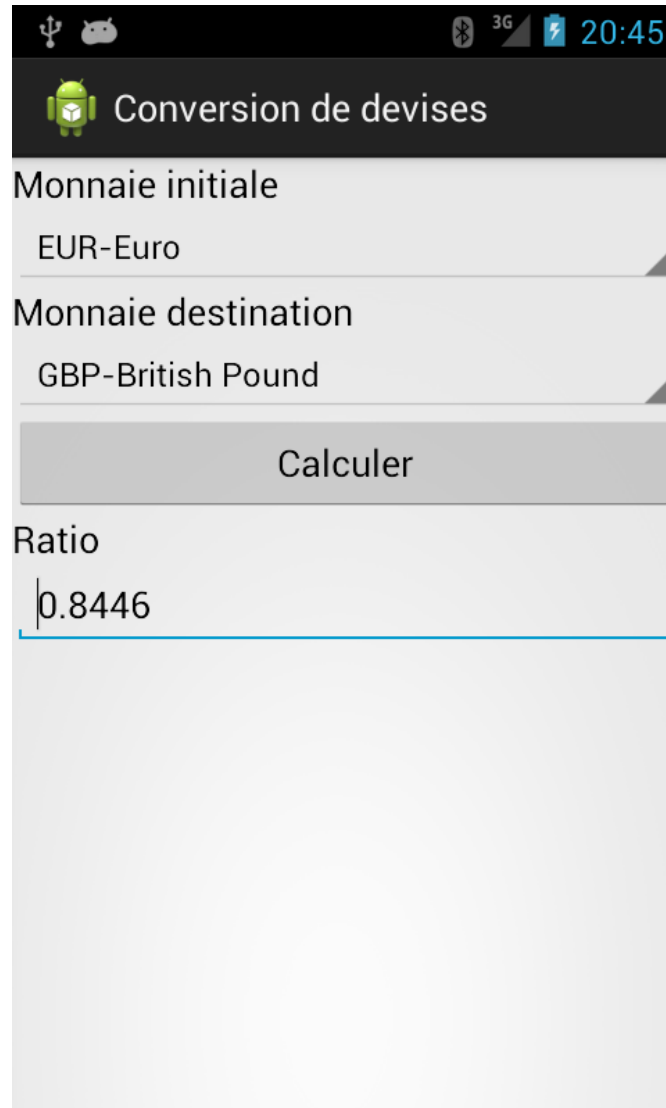
    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
```

AndroidManifest.xml





Exécution sur le téléphone

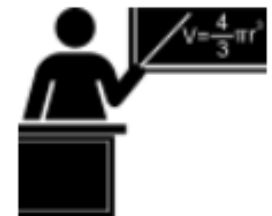


Conclusion



Un développement facilité

- **Les applications mobiles sont en forte progression**
 - Besoins croissants en web services
- **Développement facilité sous Android**
 - REST : communications HTTP standard
 - SOAP : bibliothèque ksoap2
- **Besoins en formation, consultance...**
 - Contactez nous !
 - placomme@isima.fr, ren@isima.fr,
 - romain.guidoux@clermont.inra.fr,
 - j.fontanel@qualiac.com



- **Ellipses**
 - 15 exemplaires gratuits
- **Nos organismes/employeurs**
 - Ressources
 - Disponibilité
 - ...

