Programmation objets, web et mobiles (JAVA)

Programmation Androïd

Licence 3 Professionnelle - Multimédia

Philippe Esling (esling@ircam.fr)

Maître de conférences – UPMC

Equipe représentations musicales (IRCAM, Paris)

UPMC SORBONNE UNIVERSITÉS

ircam Centre Pompidou

UPMC SORBONNE UNIVERSITÉS

Sommaire

- Un peu d'historique
- Andoid OS comme *middleware*
  - Applications et évènements gérés par le middleware
  - Une approche déclarative des IHM en XML
  - Une configuration en XML
  - Linux et Java
    - sans la JVM mais avec une DVM
  - …
- Principes de base
- *Ce n'est qu'une introduction …*

Android : les objectifs

- http://www.android.com

- http://www.OpenHandsetAlliance.com

  "… Open Handset Alliance™, a group of 47 technology and mobile companies have come together to accelerate innovation in mobile and offer consumers a richer, less expensive, and better mobile experience.

  • Together we have developed Android™, the first complete, open, and free mobile platform.

  • We are committed to commercially deploy handsets and services using the Android Platform. "

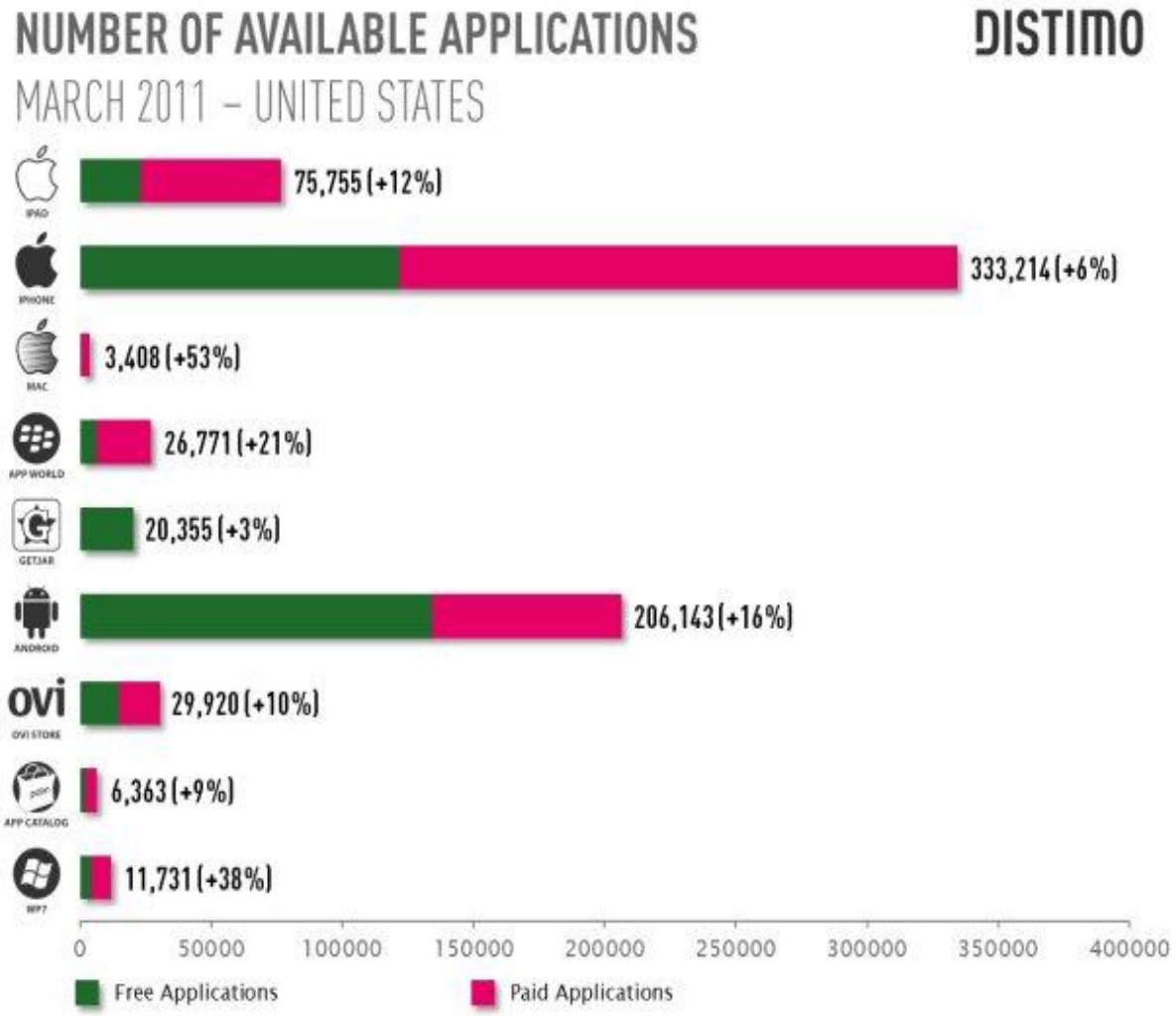
Android ?

- Une plate forme ouverte
- + un ensemble de librairies, de composants logiciels pour les systèmes embarqués et mobiles
  - Un système d'exploitation
    - **Linux**
  - Un intergiciel (middleware)
  - Nombreuses librairies
    - IHM,
    - Téléphonie,
    - Multimédia,
    - Capteurs,
    - Internet, cartographie
    - …

Pourquoi Android ?

- Indépendant d'une architecture matérielle
  - Avec des contraintes matérielles à respecter par les intégrateurs
- Dédié aux systèmes embarqués et pas seulement
- Ambitions de Google/Apple
  - Marketing
  - Les applications
    - Nombreuses et gratuites sur AndroidMarket
    - Nombreuses et payantes sur AppStore

Applications gratuites (2011)



Les autres

- Apple
- Microsoft
- Nokia
- Palm
- Research in Motion (BlackBerry)
- Symbian
- Quid de JavaFX et javaME ? ….
  - javaFX/Flash prometteur mais :
    - Lancé en 2009, qui l'utilise ?
      - http://java.sun.com/javafx/1/tutorials/core/
  - javaME obsolète ?
    - Smartphone ? Pour les pays riches et émergents ?

Projections

Table 1
Worldwide Mobile Communications Device Open OS Sales to End Users by OS (Thousands of Units)

OS	2010	2011	2012	2015
Symbian	111,577	89,930	32,666	661
Market Share (%)	37.6	19.2	5.2	0.1
Android	67,225	179,873	310,088	539,318
Market Share (%)	22.7	38.5	49.2	48.8
Research In Motion	47,452	62,600	79,335	122,864
Market Share (%)	16.0	13.4	12.6	11.1
iOS	46,598	90,560	118,848	189,924
Market Share (%)	15.7	19.4	18.9	17.2
Microsoft	12,378	26,346	68,156	215,998
Market Share (%)	4.2	5.6	10.8	19.5
Other Operating Systems	11,417.4	18,392.3	21,383.7	36,133.9
Market Share (%)	3.8	3.9	3.4	3.3
Total Market	**296,647**	**467,701**	**630,476**	**1,104,898**

Source: Gartner (April 2011)

Android les grandes lignes

- Composants Android
- Outils de Développement
- Architecture Logicielle
- Développement
  - en java (?)
  - quelques directives
  - configurations en syntaxe XML
- Deux exemples
  - Démonstration
    - Deux exemples en quelques lignes de java

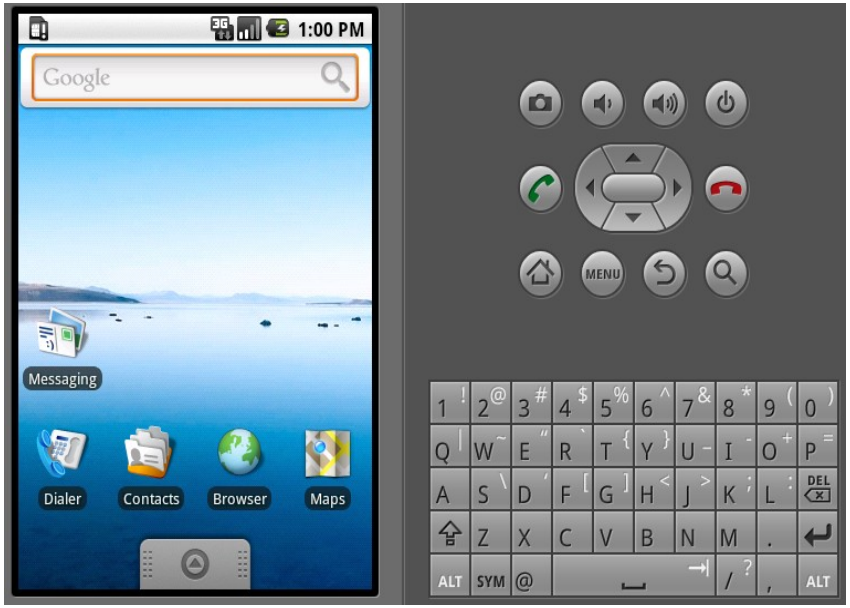
Composants Android

- Framework de déploiement d'applications
- Dalvik comme machine virtuelle (à registres != JVM à pile)
- Navigateur intégré, WebKit (webkit utilisé par safari, Google Chrome…)
- SQLite
- Multimédia support PNG, JPG, GIF, MPEG4, MP3

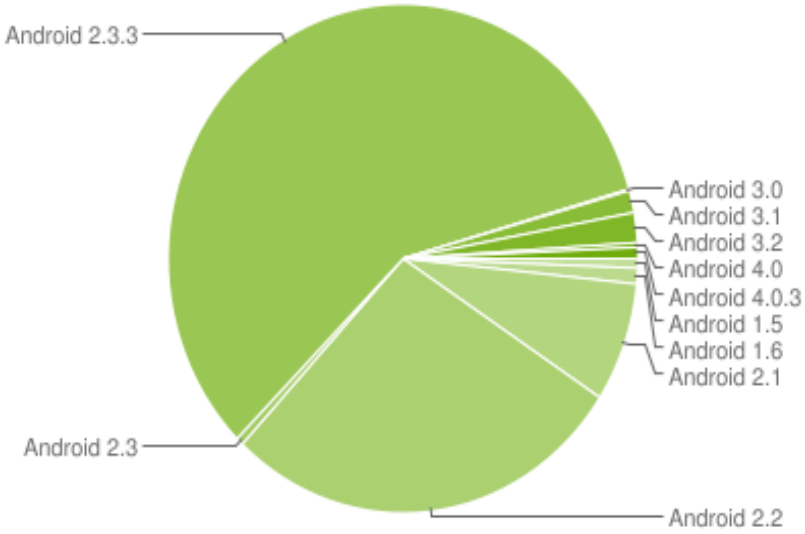
- Dépendant du matériel
  - GSM
  - Bluetooth, EDGE, 3G, WiFi
  - Caméra, GPS, boussole et accéléromètre
  - Température,
  - …

Outils de développement

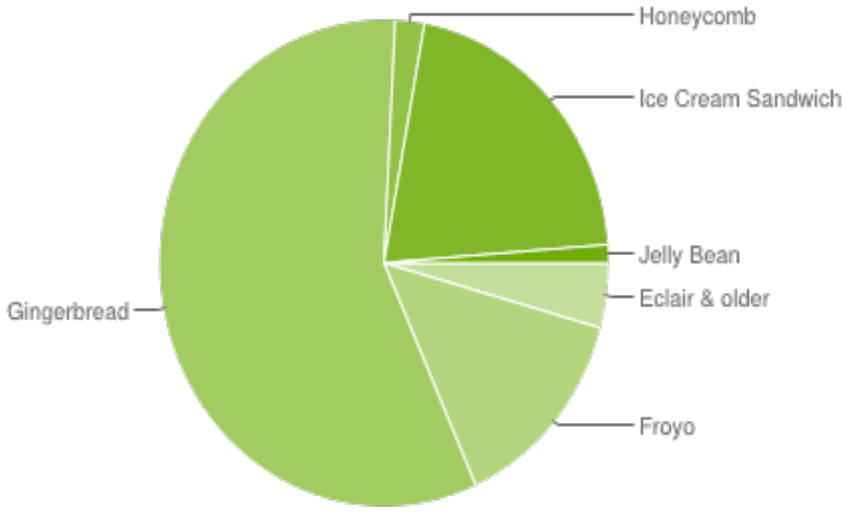
- SDK Android
  - En ligne de commandes
  - Plug-in sous eclipse
  - Émulateur
  - Débogueur
  - Traces fines d'exécution
  - Tests unitaires
  - Outils de mise au point
    - Mesure de mémoire et performance



Quelle version ?



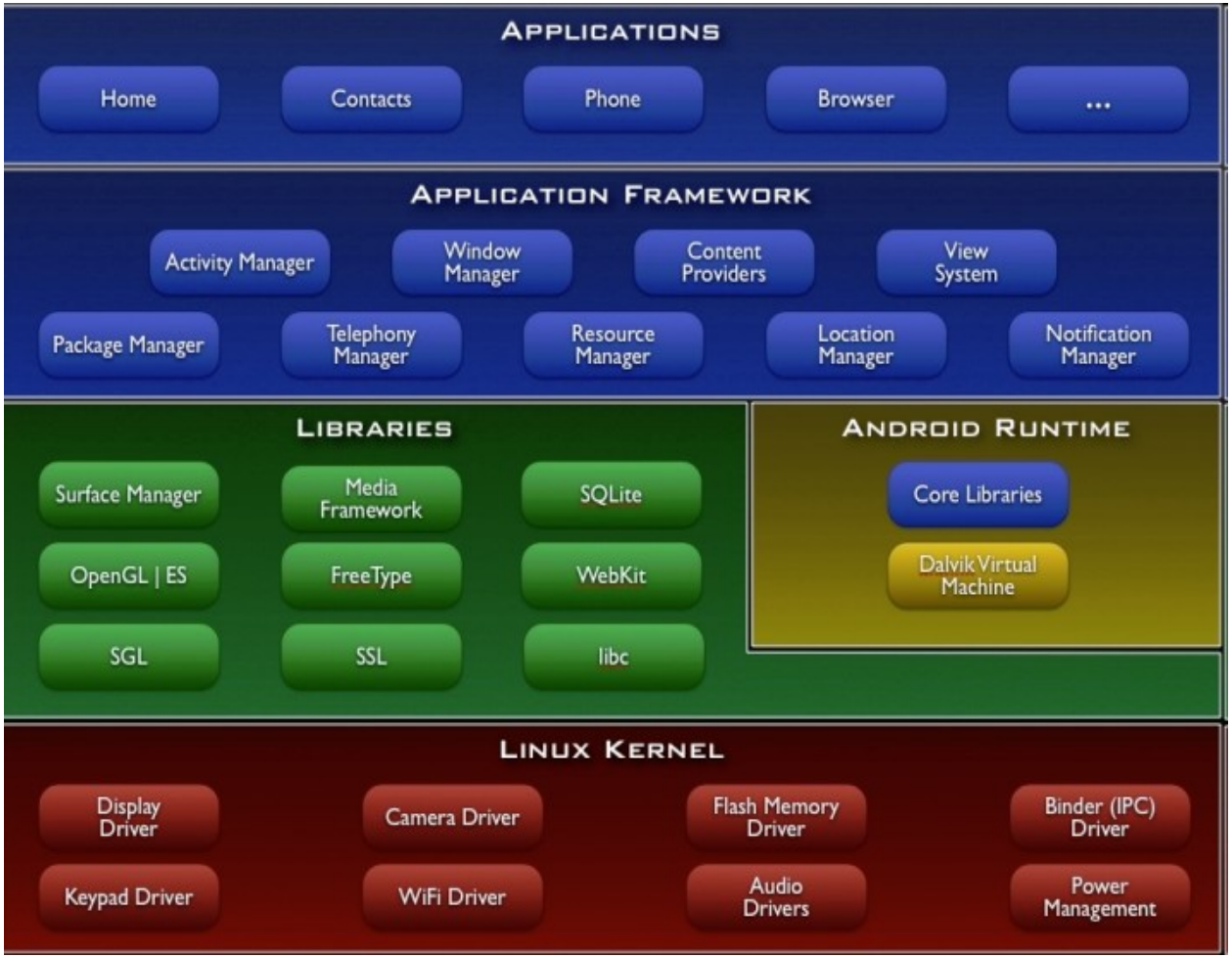
Mai 2012



Septembre 2012

- http://developer.android.com/resources/dashboard/platform-versions.html
  - Froyo = Android 2.2
  - Gingerbread 2.3.X

Composants Android



http://developer.android.com/guide/basics/what-is-android.html

Android OS

- **Un ensemble d'API**
  - http://developer.android.com/guide/basics/what-is-android.html

Middleware Android OS (extrait)

- View System listes, boutons,… navigateur (WebView)
- Resource Manager, accès aux String, aux descriptifs de lihm
  - R.java …
- Activity Manager gestion du cycle de vie d'une application
  - Une application android peut être composée de plusieurs activités
- Content Providers Accès aux données d'autres applications
  - partage, persistance de données
- Notification Manager autorise des alertes dans la barre de statut
- TelephonyManager

Librairies

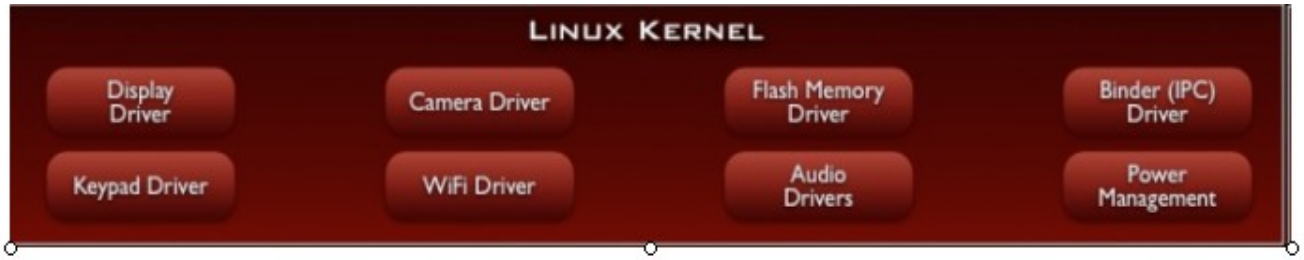
C/C++ …

- SGL comme moteur pour le 2D
- FreeType comme fontes de caractères

Dalvik VM, au lieu de JVM

- Machines à registres

- Chaque application à sa propre DVM
  - Communication inter-applications assurée par le middleware
  - Multi thread assuré par Linux
  - Accès aux capteurs par le noyau Linux



Développement d'une application

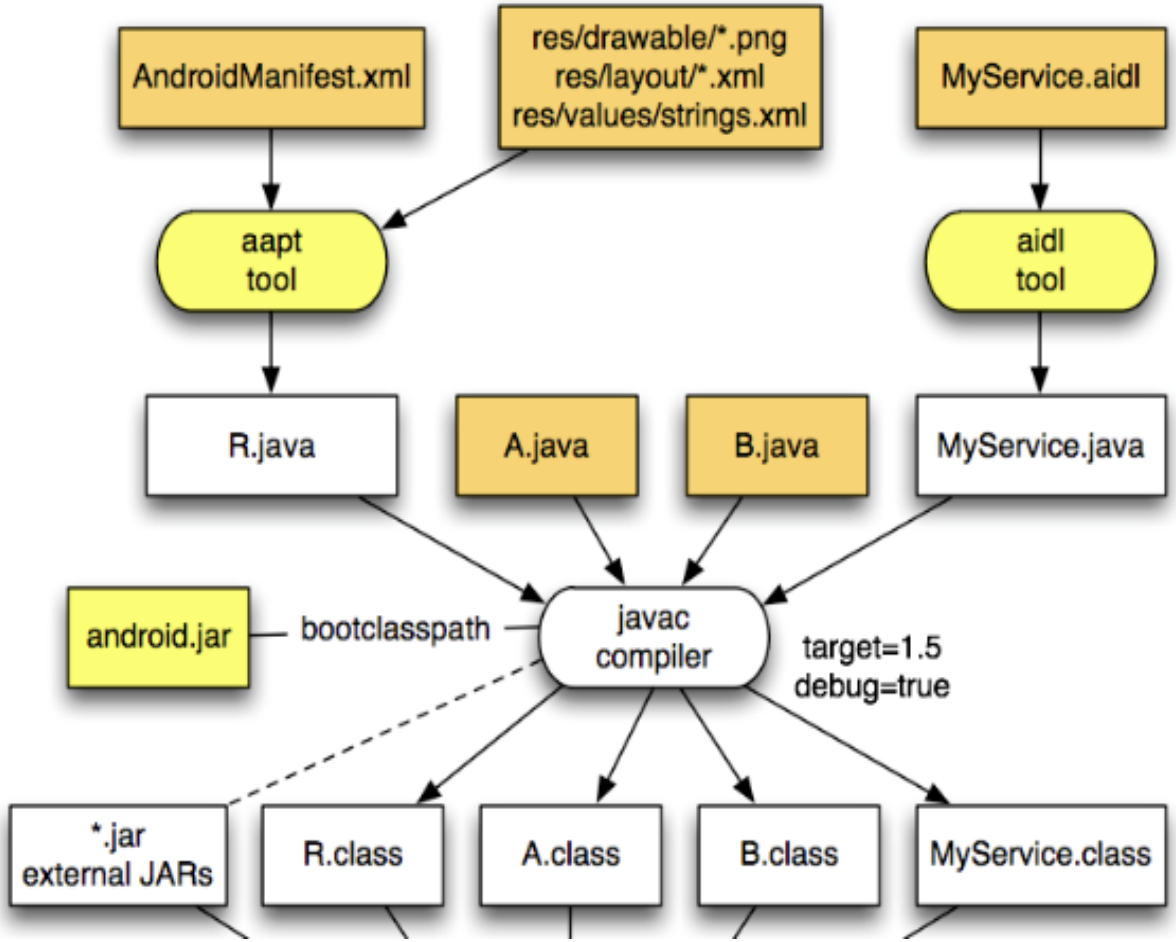
1. Obtention des .class
2. Génération de l'APK, (Android Package file)

1. Obtention des .class

- Fichier de configuration XML
  - Un source Java est généré, le fichier de ressources R.java
    - Configuration de l'application, IHM, String, …
      - Approche déclarative de l'IHM
- java
  - Paquetage java.lang,
    - Attention ce ne sont pas les librairies du JavaSE *(android.jar n'est pas rt.jar)*

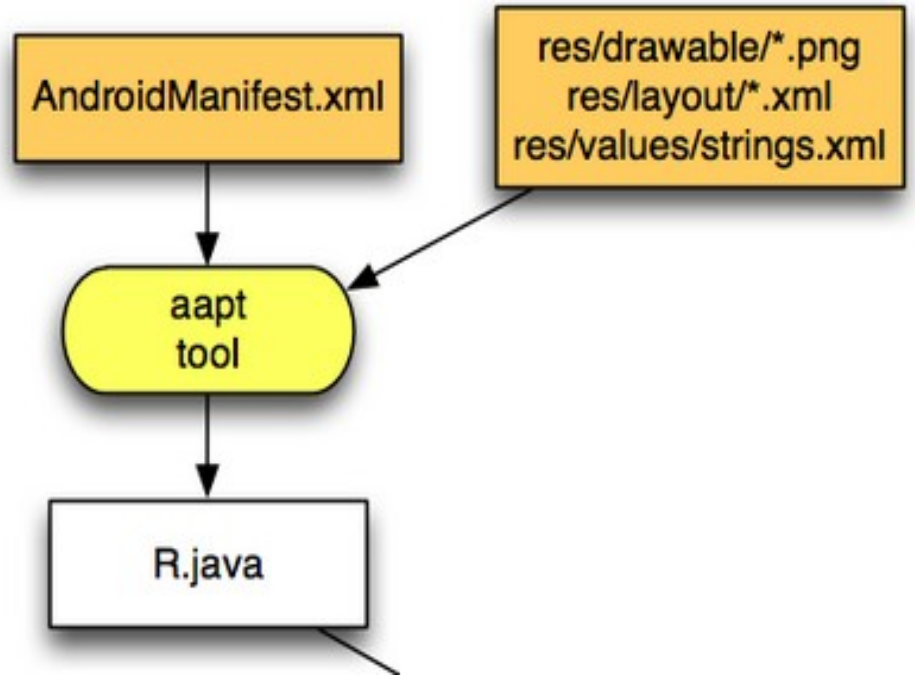
- Compilateur javac de Sun/Oracle
  - javac –bootclasspath android.jar android/introduction/*.java

Compilation : obtention des .class



Source : http://stuffthathappens.com/blog/wp-content/uploads/2008/11/android_flow.png

Aspect déclaratif et configuration



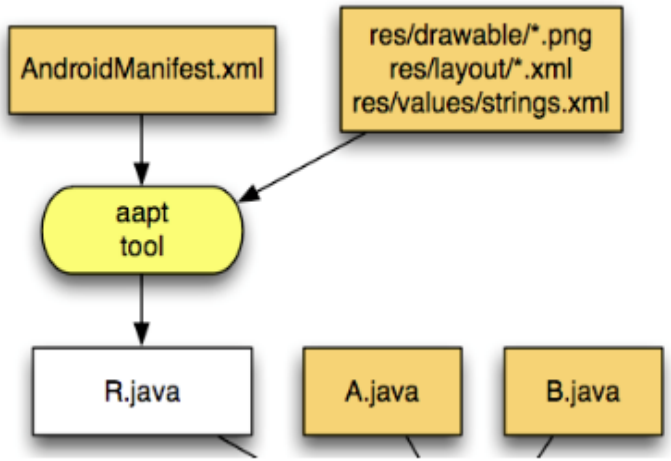
- Syntaxe XML
  - R.java généré par l'outil aapt ou eclipse

Développement 1/2

Fichier de configuration, AndroitManifest.xml

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

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

  <application android:icon="@drawable/icon" android:label="@string/app_name">
    <activity android:name=".Demo"
          android:label="@string/app_name">

      <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>
  </application>

</manifest>
```

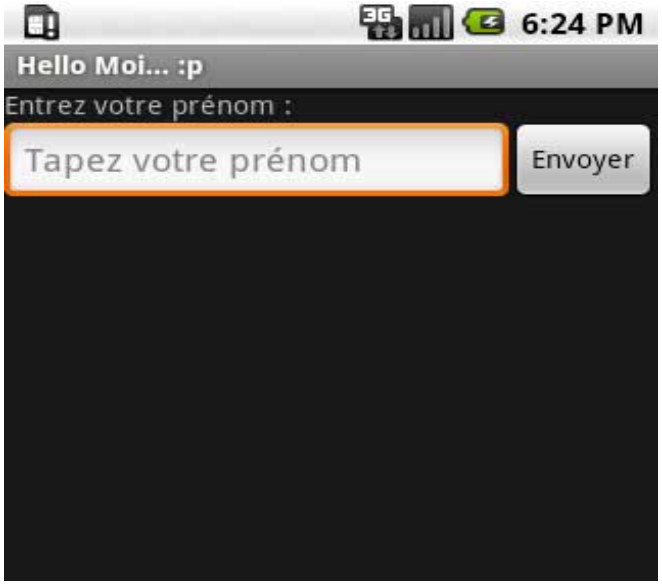
Fichier de Ressources XML associé




```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  android:orientation="vertical"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent">
  …
</LinearLayout>
```

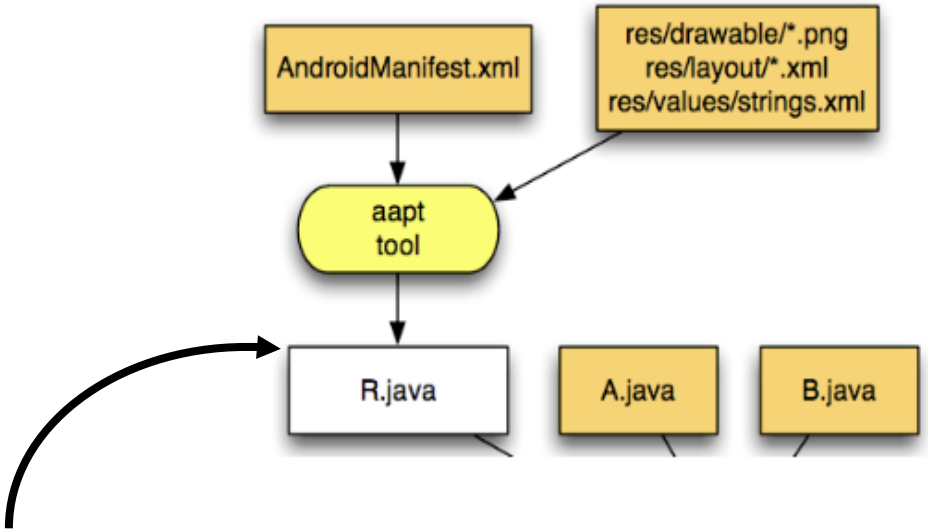
- Répertoire res/
  - Projet eclipse
- res/layout/main.xml

Fichier de Ressources XML associé




```
<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:id="@+id/TextViewPrenom' android:layout_width="fill_parent »
android:layout_height="wrap_content »
android:text="@string/prenom »/>

        <LinearLayout  android:orientation="horizontal"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content » >

          <EditText android:id="@+id/EditTextPrenom"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                 android:layout_weight="1"
                 android:layout_gravity="bottom"
                android:hint="@string/prenomHint »/>

          <Button android:id="@+id/ButtonEnvoyer"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/bouton »/>
        </LinearLayout>

        <TextView android:id="@+id/TextViewHello"
          android:layout_width="wrap_content"
          android:layout_height="fill_parent"
          android:layout_gravity="center_horizontal"
          android:textSize="@dimen/dimMessage"
          android:textColor="@color/couleurMessage »/>

</LinearLayout>
```

Le fichier R.java

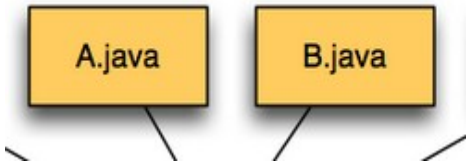



```
package test.biblio;
public final class R {
   ...
   public static final class layout {
      public static final int main=0x7f030000;
   }}
```

- /test/biblio/R.java AUTO-GENERATED FILE. DO NOT MODIFY.

Un premier source java, juste pour la syntaxe…

```
package test.biblio;
import android.app.Activity;
import android.os.Bundle;
```



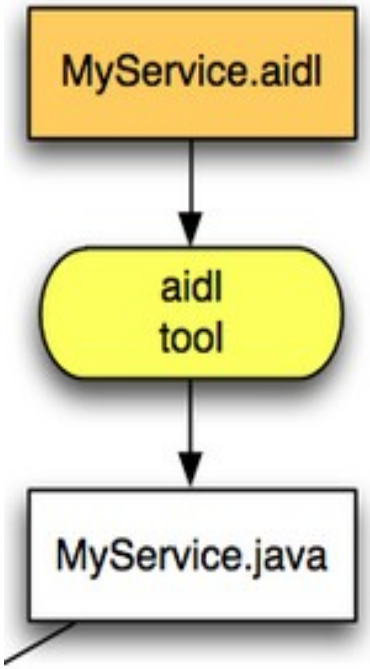

```
public class Demo extends Activity {
    ....
   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);

      setContentView(R.layout.main); // association IHM <-> Activity

         ....
}
}
```

```
package test.biblio;
public final class R {
   ...
   public static final class layout {
      public static final int main=0x7f030000;
   }}
```

Un premier source java, juste pour la syntaxe…

```
public class Demo extends Activity {
   private EditText editText;
   private Button button;

public void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);

       setContentView(R.layout.main); // association //récupération de l'EditText grâce à son ID
      editText = (EditText) findViewById(R.id.EditTextPrenom);
      //récupération du bouton grâce à son ID
      button = (Button) findViewById(R.id.ButtonEnvoyer);
      //on applique un écouteur d'évenement au clique sur le bouton
      button.setOnClickListener(
       new OnClickListener() {
   @Override
   public void onClick(View v) {
   ?? // On récupère le texte écrit dans l'EditText
   ?? // On crée une pop-up Toast
   ?? // On affecte le texte dans l'objet TextView
   }
   }
      );}
}
```

Services, même principe (tâche de fond)



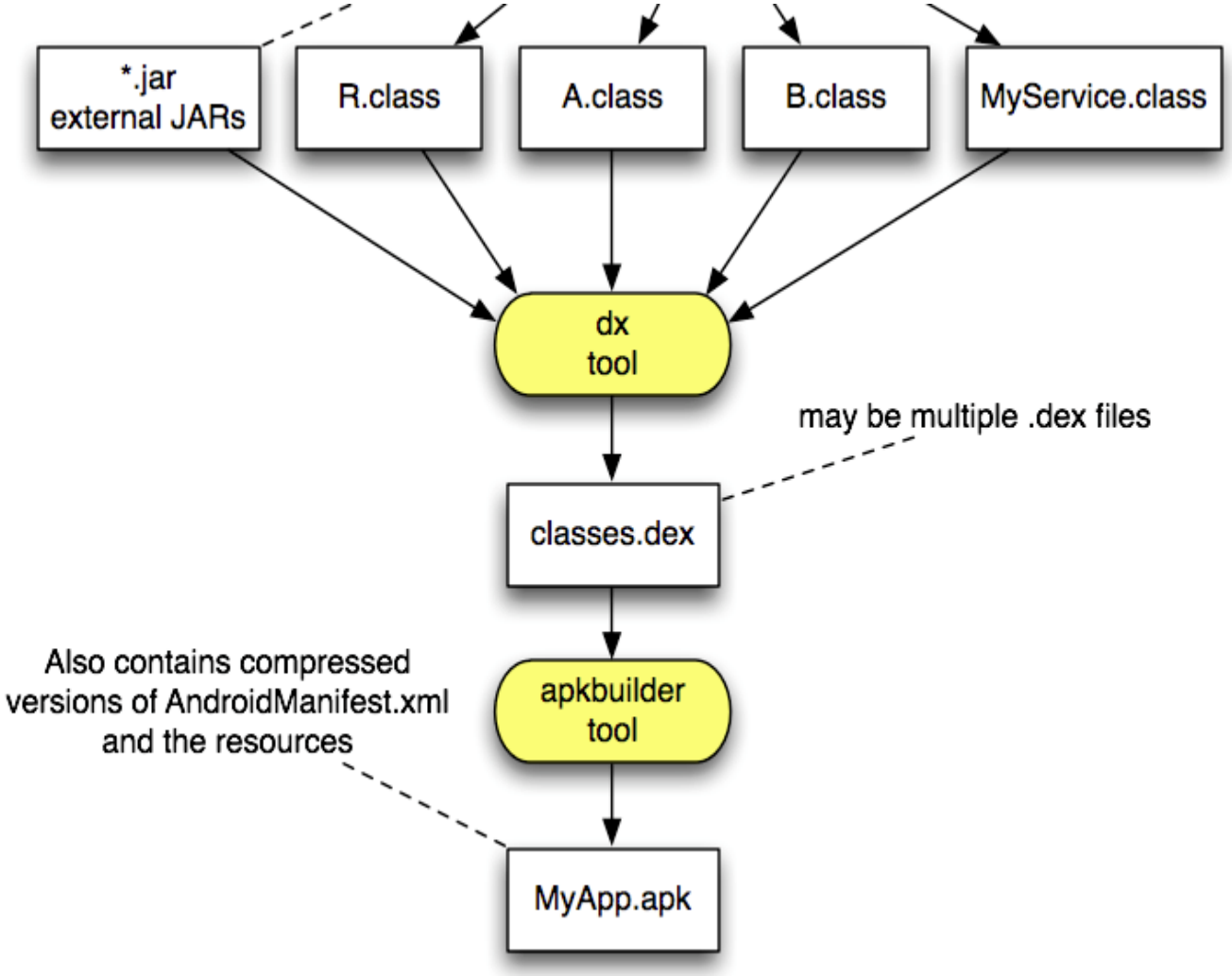
- Une application sans IHM,
  - un couple<processus,DVM> peut lui être dédié

Obtention de l'application

- De tous les .class en .dex
  - De la JVM à la machine Dalvik
  - D'une machine à pile en machine à registres

- Génération de l'application .apk
  - Une archive signée
  - Téléchargement : émulateur ou mobile

Génération de l'application



Source : http://stuffthathappens.com/blog/wp-content/uploads/2008/11/android_flow.png

Développement 2/2, suite

- Du .class en .dex
  - Assemblage de tous les .class vers un .dex
  - Une machine par application, un processus Linux
  - Les applications communiquent via l'intergiciel
    - Une application peut être composée de plusieurs activités
    - Les activités communiquent via des variables globales, de la mémoire persistante,…
- Génération de l'application .apk
  - Assemblage, édition des liens
  - Une archive signée
  - Téléchargement : émulateur ou mobile

Introduction aux Applications

- Une présentation, un vocabulaire

- Mots-clés
  - Applications,
  - Communication, évènements, intentions,
  - Services en tâche de fond,
  - Persistance.

- Une Application est composée d'une ou de plusieurs *Activity*
  - *Une activity*
  - Surcharge de certaines méthodes,
    - Du déjà vu : Applet, MIDlet, Servlet,…
  - Le cycle de vie est imposé par le framework
    - Déjà vu : pour une Applette init() puis start() …

Vocabulaire

- Les Essentiels
  - Activity
  - BroadcastReceiver
  - Service
  - ContentProvider

Introduction … Classes

- Activity
  - Une interface utilisateur
    - Démarre d'autres activités, émet des évènements(intentions, intent)
  - Une configuration de type XML, permissions, librairies,

- BroadcastReceiver
  - Bus de messages
  - Émission et réception d'intentions

- Service
  - Pas d'interface, un service à rendre, en tache de fond
  - Intention de servir

- ContentProvider
  - Données rendues persistantes (pour d'autres applications)
  - Un fichier, base SQLite

Deux exemples, deux Activity

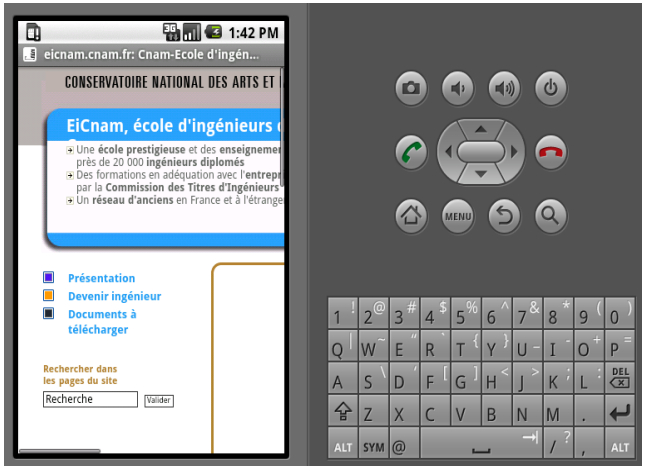
1. Installation d'un navigateur en 2 lignes (WebView)

2. Une toute petite IHM
   - Un écran constituée dun bouton, d'un écouteur,
   - A chaque clic, l'heure est affichée !

Activity Usage du WebKit

```
import android.app.Activity;
import android.os.Bundle;
import android.webkit.WebView;

public class BrowserDemo extends Activity {
   private WebView browser;
   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);
      browser=(WebView)findViewById(R.id.webWiew);
      browser.loadUrl("http://upmc.fr/");
   }
}
```

2 lignes



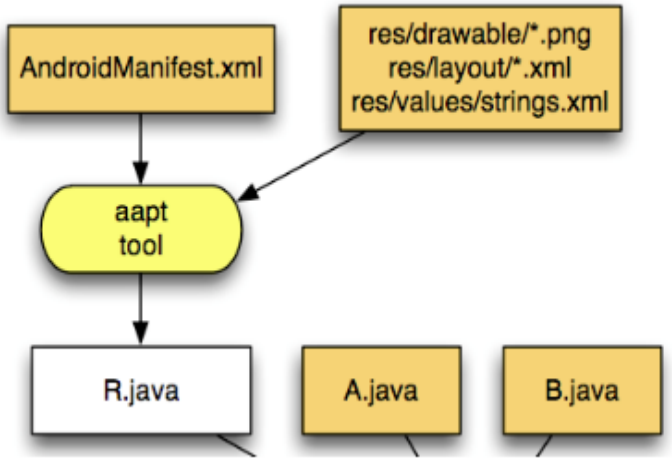
OnCreate est déclenché par le framework Android

```
public class BrowserDemo extends Activity {
    private WebView browser;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

// installation de l'IHM
        setContentView(R.layout.main);
// accès au composant graphique
        browser=(WebView)findViewById(R.id.webView);

        browser.loadUrl("http://eicnam.cnam.fr/");
    }
}
```

R.layout.*main*, R.id.*webView* ?

Retour sur la configuration XML

- R.id.*webView* ? *R.layout.main* ?
  - *En Entrée*
    - *Fichiers de configuration XML*
  - *En Sortie*
    - *Source Java, R.java*



Une IHM, deuxième exemple

Une IHM

- Un bouton, un écouteur, un clic et l'heure est affichée !
- En approche *traditionnelle*
  - Tout est codé en Java IHM comprise
- En approche *déclarative*
  - Usage d'XML pour la configuration, de java pour l'utilisation

Click pour actualiser l'heure

```
import android.app.Activity;
import android.os.Bundle;
import static android.view.View.OnClickListener ;
import android.widget.Button;
import java.util.Date;


public class Now extends Activity implements OnClickListener {
  private Button btn;

  @Override
  public void onCreate(Bundle bundle) {
    super.onCreate(bundle);
    btn = new Button(this);       // <- un bouton
    btn.setOnClickListener(this);// <- un écouteur auprès de cette vue

    setContentView(btn);          // <- le bouton occupe l'écran
  }

  public void onClick(View view) {  // <- à chaque click
    btn.setText(new Date().toString());
  }
}
```

Discussion : Vue apparentée swing

Ici un MVC à lui tout seul …

Activity, méthodes à redéfinir

- MonActivity extends `android.app.Activity;`

```
@Override
  protected void onCreate(Bundle
   savedInstanceState){
```

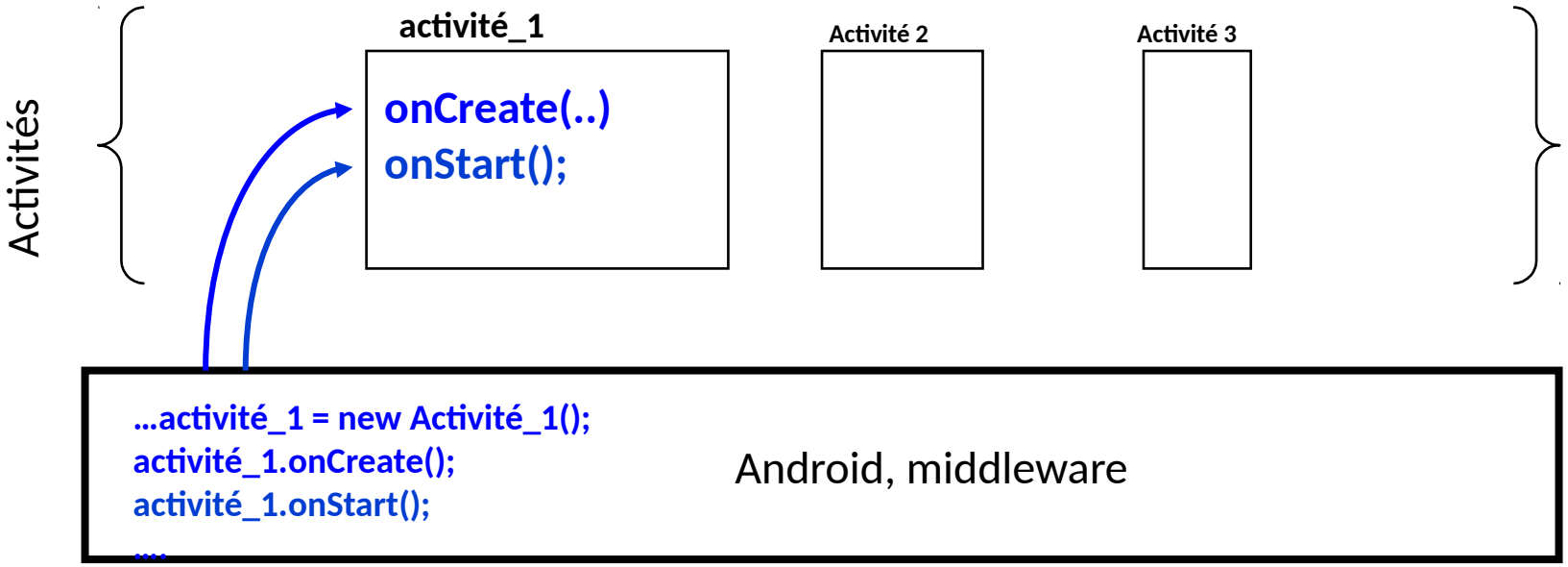
android.app.Activity

package android.app;

```
public class Activity extends ApplicationContext {

        protected void onCreate(Bundle savedInstanceState){

          protected void onStart();

          protected void onRestart();

          protected void onResume();

          protected void onPause();

          protected void onStop();

          protected void onDestroy();

        … etc …
}
```

induit un cycle de vie imposé par le « framework »

Inversion de Contrôle… Rappel




```
public class Activity extends ApplicationContext {  1
      protected void onCreate(Bundle savedInstanceState);
      protected void onStart();
       …..}
```

http://developer.android.com/reference/android/app/Activity.html

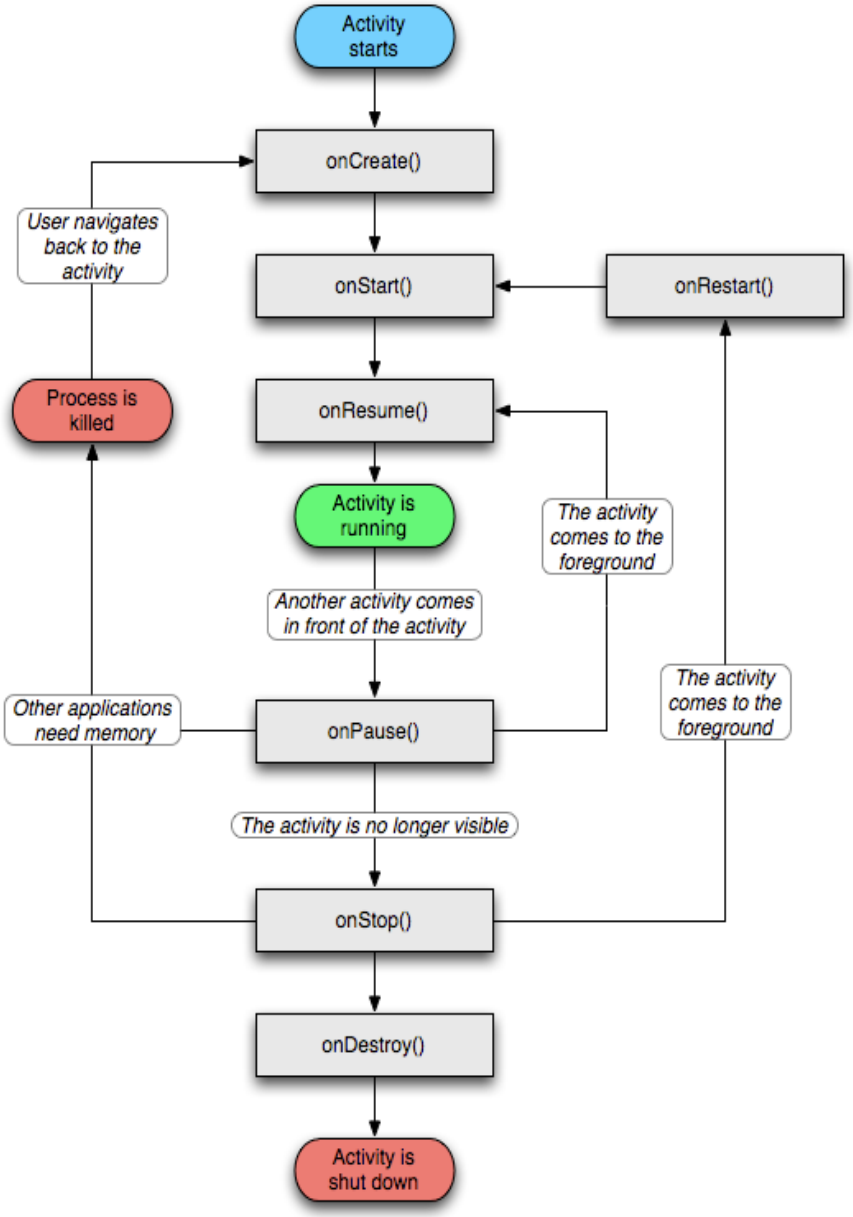
Petites précisions, rappels …

- Une application s'exécute dans un processus Linux

- Depuis ce processus
  - les méthodes d'une activité sont appelées selon un certain ordre
    - onCreate …onPause…. onResume… onPause …. onResume… onDestroy

  - L'appel de onDestroy n'engendre pas l'arrêt du processus initiateur

Activity : les états, http://developer.android.com/reference/android/app/Activity.html

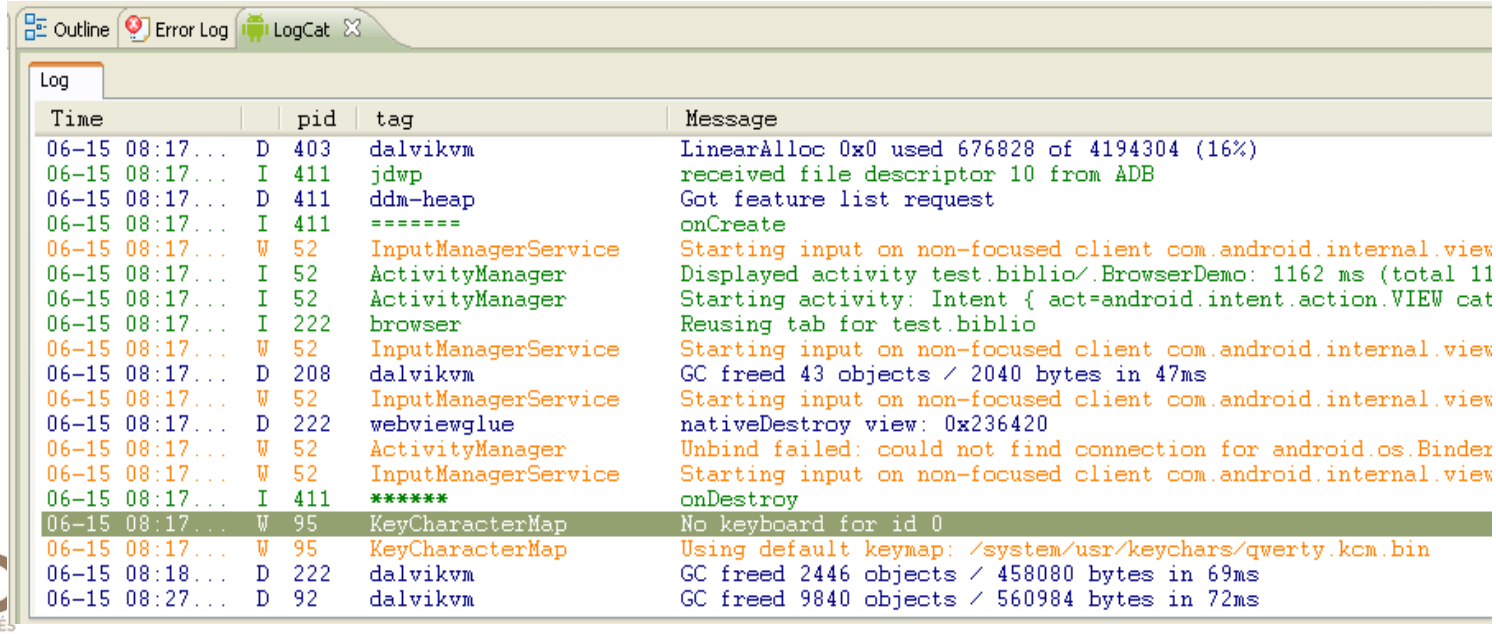


En tache de fond :
Empiler(l'activité);

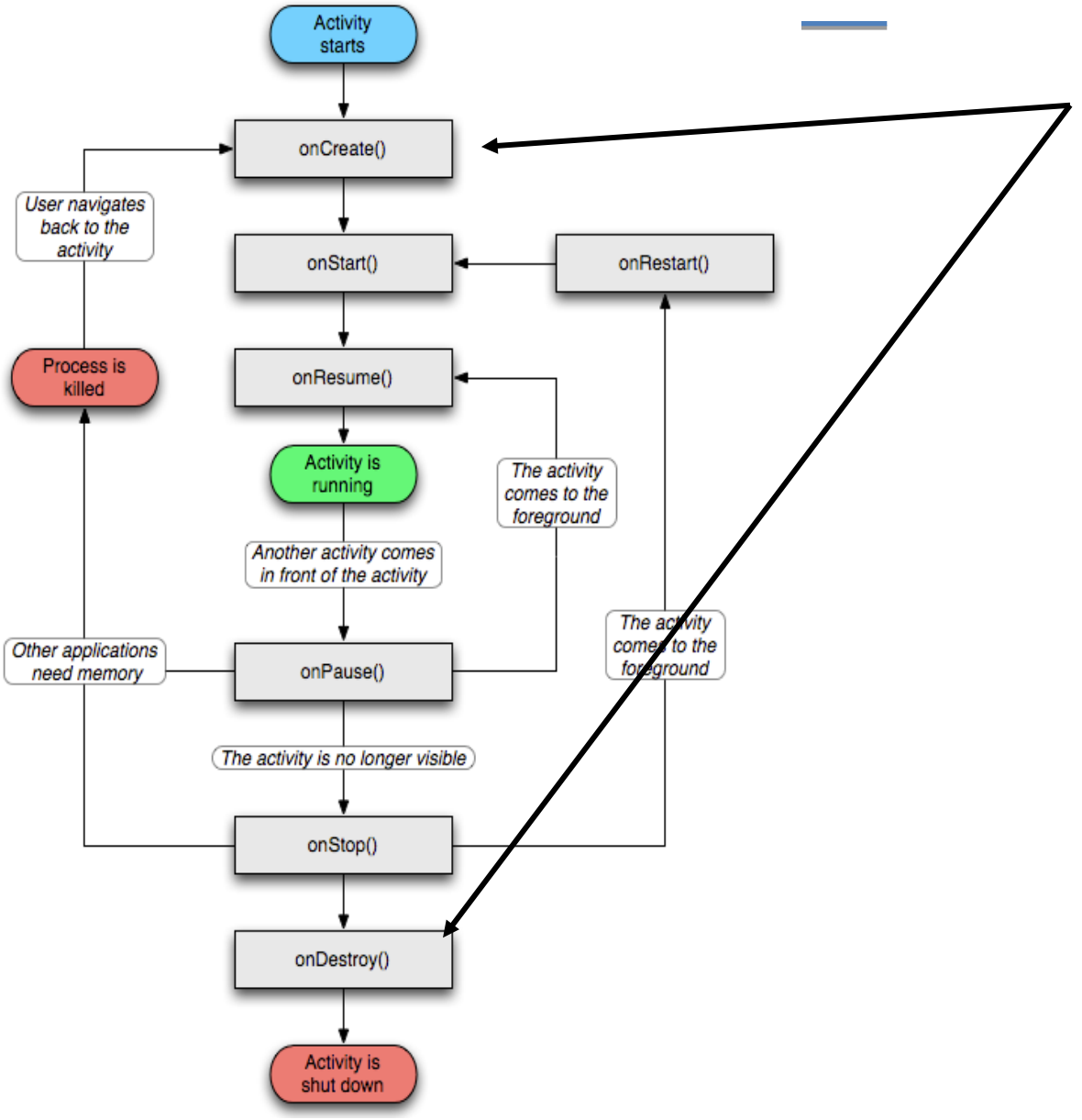
onResume
activité au 1er plan = Dépiler()

Démonstration, Activity dans tous ses états

```
public class BrowserDemo extends Activity {
  private WebView browser;
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Log.i("=======","onCreate");
        // cf. page précédente
    }
    public void onDestroy(){
      super.onDestroy();
      Log.i("******", "onDestroy");
    }
}
```



En résumé : émulateur + LogCat , traces bien utiles

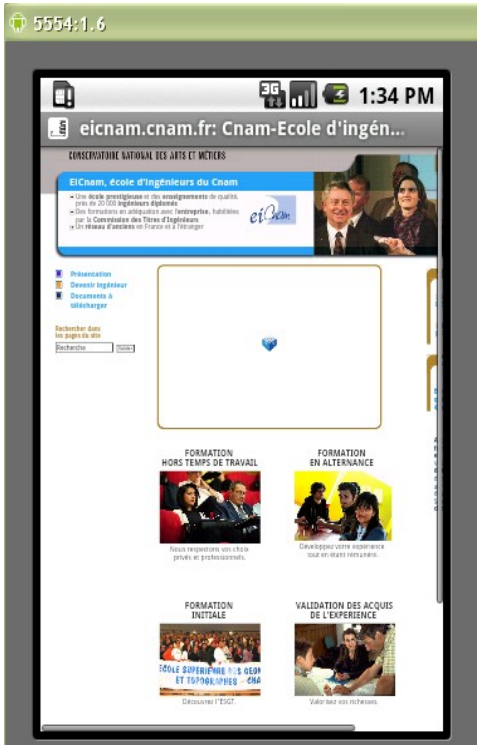


Activity dans tous ses états

```
public class BrowserDemo extends Activity {

  public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);Log.i("=======","onCreate");}

    public void onStart()
    {super.onStart();Log.i("=======","onStart");}

    public void onResume(){
      super.onResume();
      Log.i("=======","onResume");
    }

    public void onPause(){
      super.onPause();
      Log.i("=======","onPause");
    }

    public void onStop(){
      super.onStop();
      Log.i("******","onStop");
    }

     public void onDestroy(){
       super.onDestroy();
       Log.i("******", "onDestroy");
     }
}
```

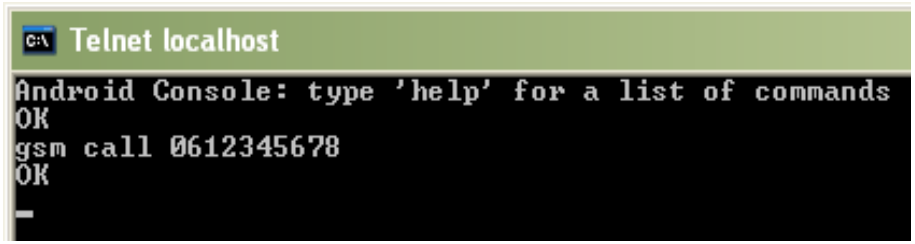
OnPause -> onResume

1)

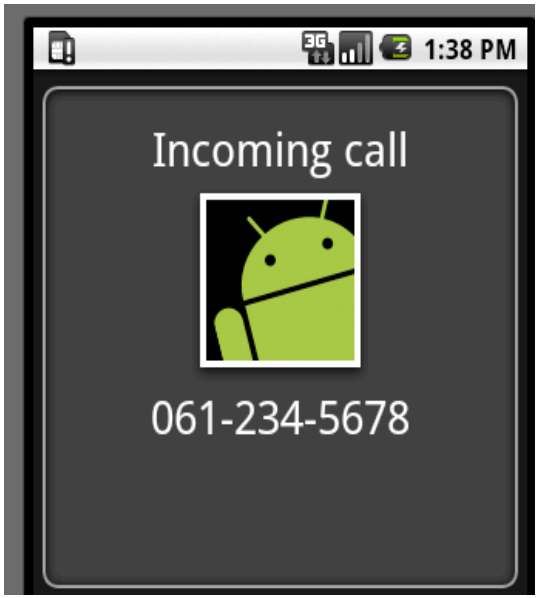


OnPause

2-1) telnet localhost 5554



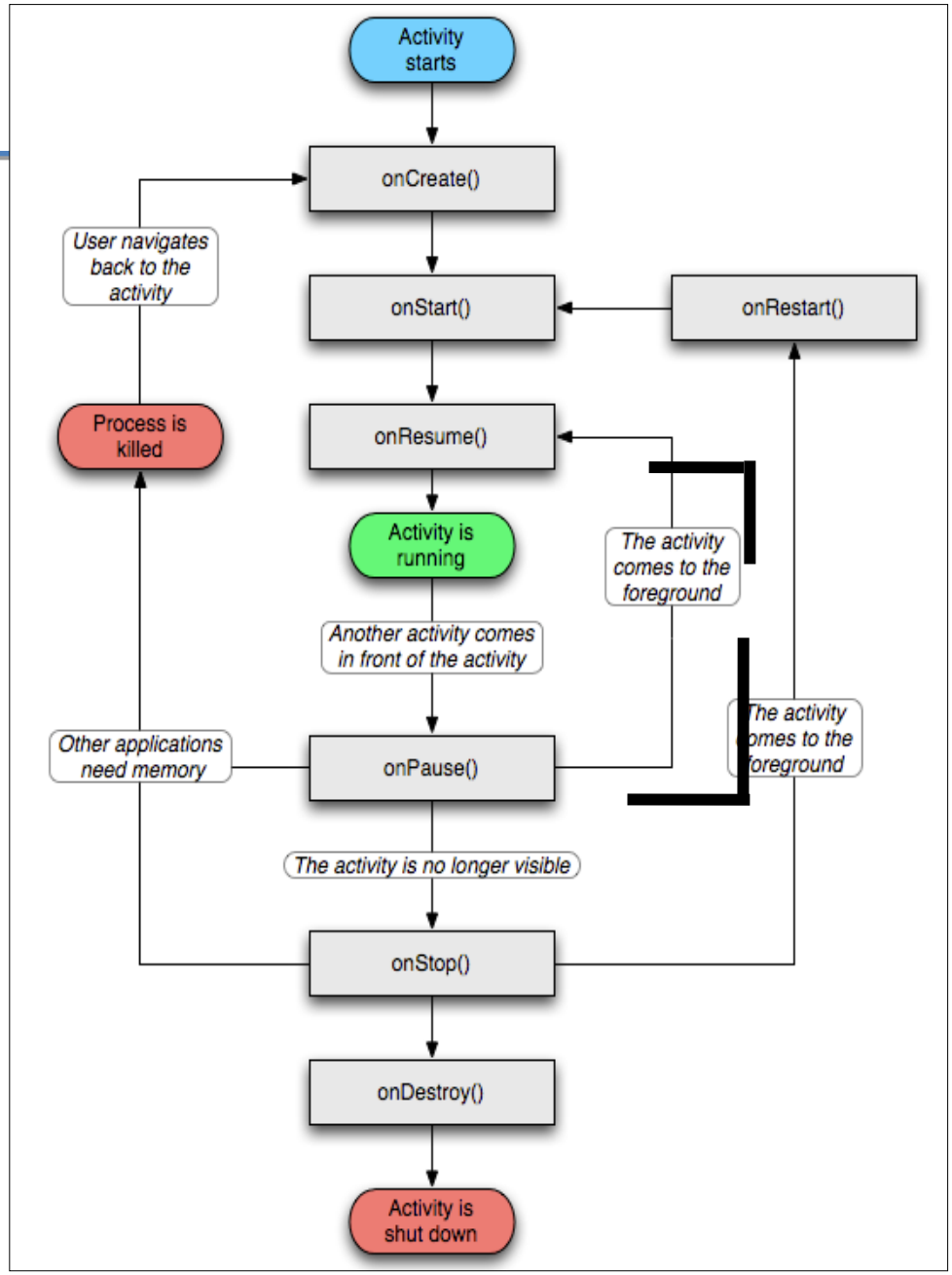
2-2)



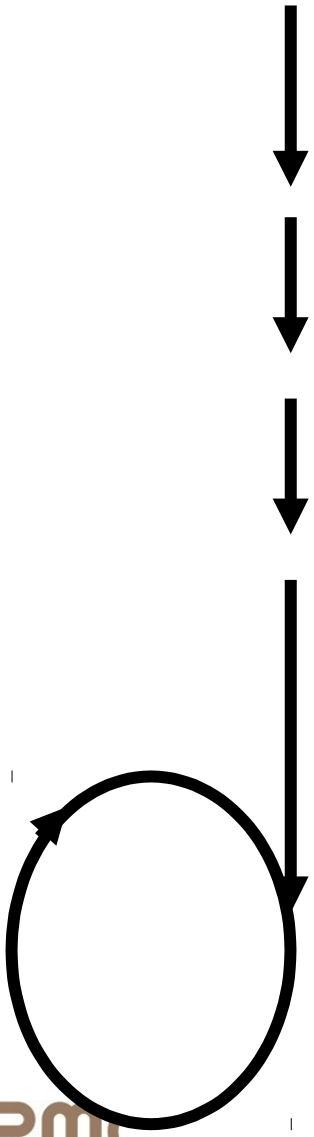
2)

OnResume

3)


Activity starts
onCreate()
User navigates back to the activity
onStart()
onRestart()
Process is killed
onResume()
Activity is running
The activity comes to the foreground
Another activity comes in front of the activity
Other applications need memory
onPause()
The activity comes to the foreground
The activity is no longer visible
onStop()
onDestroy()
Activity is shut down

Le fil d'exécution, une activity



1. Démarrage d'une A, un processus, une DVM
   - Création de l'instance par Android, un thread (main) lui est associé
2. Appel de onCreate()

- Appel de onStart()
- Appel de onResume()
- A est dans une boucle d'attente des évènements
  - Évènements de l'utilisateur
  - Intention du système
  - Un appel téléphonique

Application, Activity …

– Un processus linux contient une application,

– Une application, peut contenir une ou plusieurs activités,

– Une activité se trouve dans un certain état, cf. cycle de vie

– Les threads locaux au processus, sont indépendants d'une activité

– Une activité peut être dans un processus Linux

Communication inter-activités

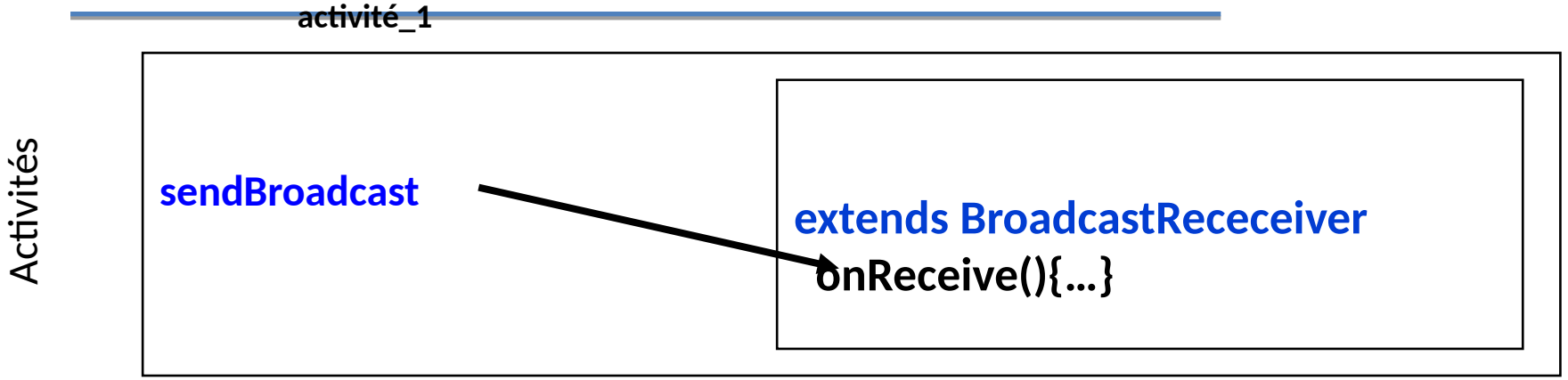
- Communication comme intention
  - Entre deux applications, entre deux dalvik
  - Abonnements

Intention de … porter un toast

- Intent

```
Toast.makeText(context,intent.getStringExtra("message"),3000).show();
```

Android : Intents, comme Intention



Il faut :

- Un receveur

```
public class Receiver extends BroadcastReceiver {
  public void onReceive(Context context, Intent intent) {
    ….
```

- Enregistrement auprès du middleware
- Une intention d'émettre
- Ajuster la configuration

```
<receiver android:name=".TestBroadcastReceiver"
android:label= "Test Broadcast receiver" />
```

Intention de

Un receveur, qui porte un toast : abonné

```
public static class Receiver extends BroadcastReceiver {
  public void onReceive(Context context, Intent intent) {
    Toast.makeText(context,intent.getStringExtra("message"),3000).show();
  }
}
```

Enregistrement auprès du middleware : abonnement

```
Receiver receiver = new Receiver();
registerReceiver(receiver, new IntentFilter("PORTER_UN_TOAST"));
registerReceiver(receiver,new IntentFilter(Intent.ACTION_AIRPLANE_MODE_CHANGED));
```

Émission de l'intention : notification

```
intent = new Intent("PORTER_UN_TOAST");
intent.putExtra("message", getResources().getString(R.string.message));
sendBroadcast(intent);
```

- Intent.*ACTION_AIRPLANE_MODE_CHANGED, un abonné à cet Intent*

Intention… auprès du middleware

Enregistrement auprès du middleware :

```
Receiver receiver = new Receiver();
registerReceiver(receiver, new IntentFilter("PORTER_UN_TOAST"));

registerReceiver(receiver,new IntentFilter(Intent.ACTION_AIRPLANE_MODE_CHANGED));
```

- Intent.*ACTION_AIRPLANE_MODE_CHANGED*, *je suis maintenant abonné à cet Intent*

Avec

```
private static final String SMS_RECEIVED = "android.provider.Telephony.SMS_RECEIVED";
```

et

```
registerReceiver(receiver,new IntentFilter(SMS_RECEIVED));
```

Je suis maintenant abonné aux réception de SMS
(si la permission est installée dans le AndroidManifest.xml)

Sous Android

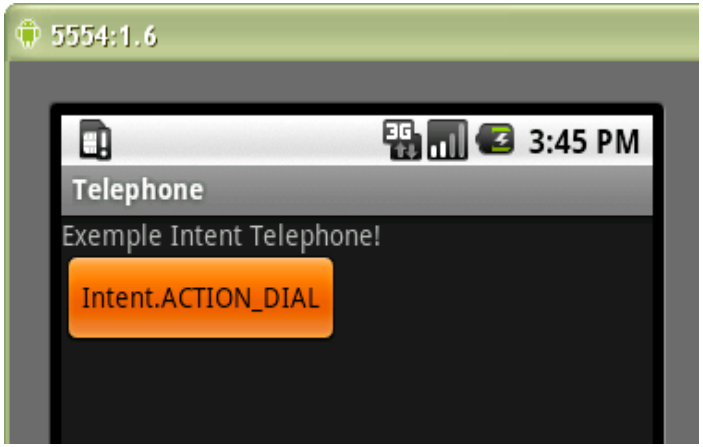
- Mediator, classe Context
  - http://developer.android.com/reference/android/content/Context.html
- Subscriber, classe BroadcastReceiver
  - http://developer.android.com/reference/android/content/BroadcastReceiver.html
- X,Y les thèmes, classe Intent
  - http://developer.android.com/reference/android/content/Intent.html
- IntentFilter
  - http://developer.android.com/reference/android/content/IntentFilter.html

Intention de téléphoner ?

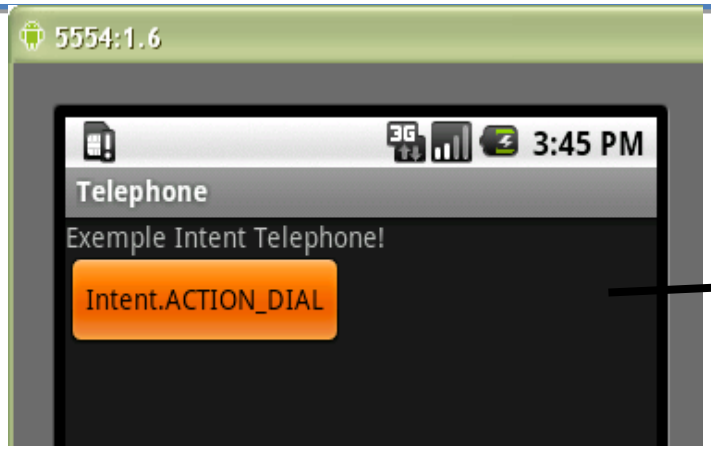
- Intent
- Comme Communication
- Démarrer une activité prédéfinie : téléphoner

Intention de téléphoner

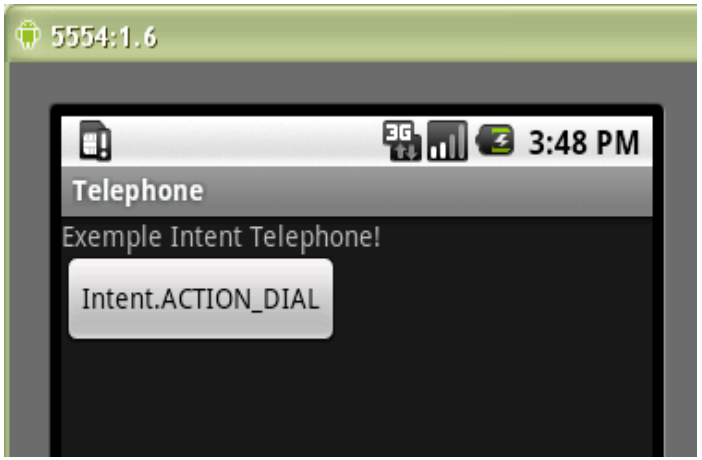
```
public class Telephone extends Activity implements  OnClickListener {

public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);
      ((Button)this.findViewById(R.id.Button01)).setOnClickListener(this);
   }

   public void onClick(View v){
      Intent intent = new Intent(Intent.ACTION_DIAL);
      startActivity(intent);
   }
}
```



En Images, activité de téléphoner



Click
empiler



Click
dépiler

Activités un résumé

- Activity
  - Cycle de vie
  - Géré par le framework
- startActivity
  - Une activité peut en démarrer une autre
  - Transmission des paramètres par les intentions
- Communication entre deux activités
  - Discussion
- Une Application peut contenir plusieurs activités
  - Communication entre activités d'applications différentes
    - Exemple des contacts

Service

- En tache de fond

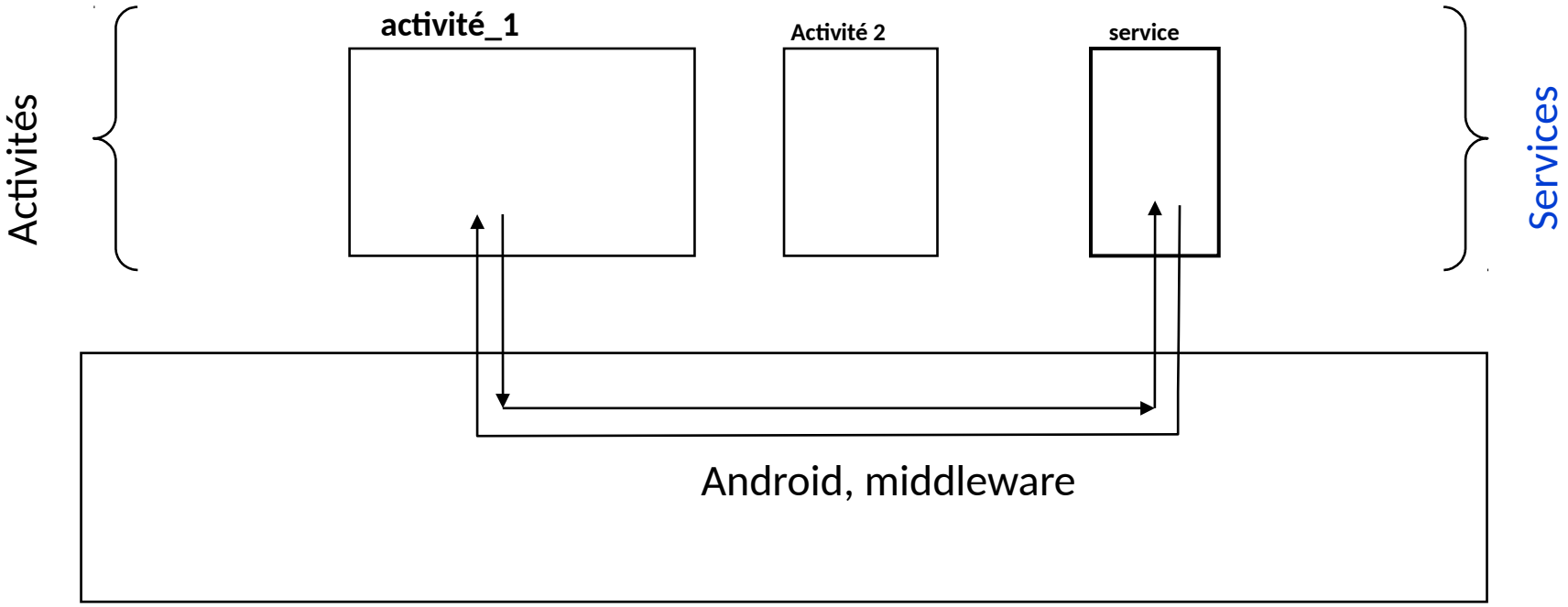
- Toujours disponible

  - Locaux
    - Service personnel, inaccessible pour les autres applications

  - Distants
    - Rmi en plus simple … même machine

    - J’écoute un fichier audio mp3, pendant que je navigue sur le web

Un Service



- Service distant,
  - Découverte du service par une intention …

SMSService : un exemple

- Un compteur de SMS reçus
  - A chaque sms reçu un compteur est incrémenté
  - Service accessible par toute application

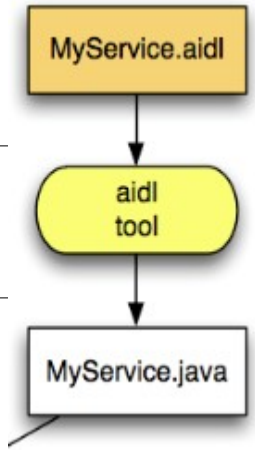
- Le service a l'intention de recevoir des SMS
  - IntentFilter filter = new IntentFilter(*SMS_RECEIVED*);
  - registerReceiver(new SMSReceiver(), filter);

- Un client recherchera ce service via le middleware

- Le service : SMSService

Service, aidl (android interface description language)

```
package cnam.android;

interface SMSService{
  void start();
  void stop();
  long received();
}
```




```
/*
This file is auto-generated.  DO NOT MODIFY.
*/
package cnam.android;
import ...;
public interface SMSService extends android.os.IInterface{

/** Local-side IPC implementation stub class. */
public static abstract class Stub extends android.os.Binder implements
cnam.android.SMSService{ ...
```

Le Client recherche le service

private SMSService statsSMS;

```
bindService(new Intent(SMSService.class.getName()),
            connexion,
            Context.BIND_AUTO_CREATE);
}


// Traitement asynchrone de la réponse venant de l'intergiciel

private ServiceConnection connexion = new ServiceConnection() {
  public void onServiceConnected(ComponentName name, IBinder service) {
    Log.v("service","onServiceConnected()");

    // réception de la souche
    Client.this.statsSMS = SMSService.Stub.asInterface(service);
  }

  public void onServiceDisconnected(ComponentName name) {
    Client.this.statsSMS = null;
  }
};
```

Le service 1/3, réveillé à chaque SMS

```
public class SMSServiceImpl extends Service {

private static final String TAG = "SMSServiceImpl";
private static final String SMS_RECEIVED =
  "android.provider.Telephony.SMS_RECEIVED";

    private boolean active;
    private int     countSMSReceived;

  public void onCreate() {
    super.onCreate();
    IntentFilter filter = new
  IntentFilter(SMS_RECEIVED);
    registerReceiver(new SMSReceiver(), filter);
}
```

Le service 2/3, réveillé à chaque SMS

```
private class SMSReceiver extends BroadcastReceiver{
  public SMSReceiver(){
    Log.v("SMSReceiver","SMSReceiver()");
  }

@Override
public void onReceive(Context context, Intent
  intent) {
  Log.v("SMSReceiver","onReceive()");
  if(active) SMSServiceImpl.this.countSMSReceived++;
}
}
```

Le service 3/3, La souche le stub fournie à la demande

```
public IBinder onBind(Intent arg0) {
  return new SMSServiceStubImpl();
}


public class SMSServiceStubImpl extends SMSService.Stub{

  public long received() throws android.os.RemoteException{
    return SMSServiceImpl.this.countSMSReceived;
  }

  public void start(){
    active = true;
  }
  public void stop() throws RemoteException {
    active = false;
  }
}
```

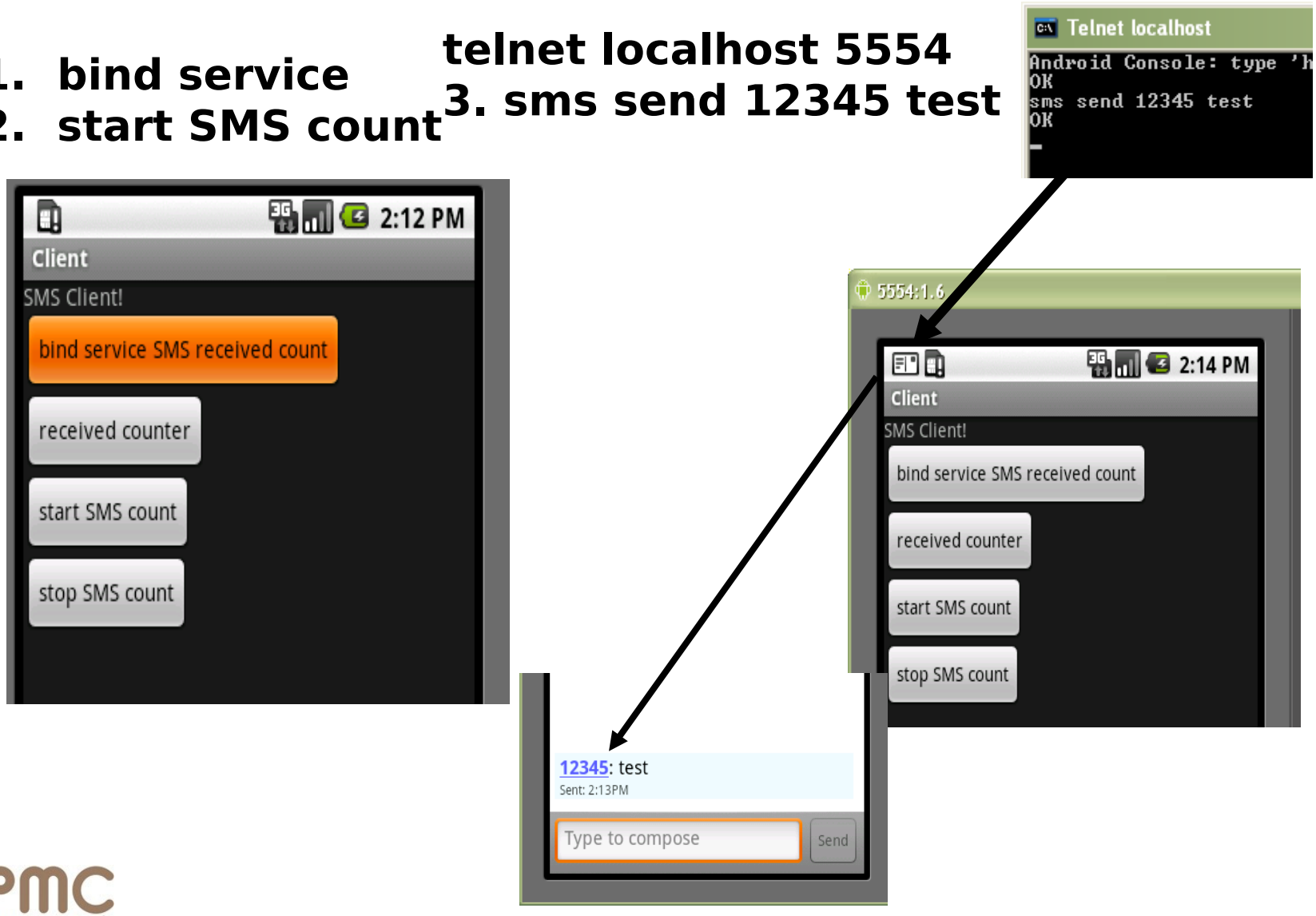
```
<service android:name=".SMSService"
        android:label= "mon service de SMS" />
```

UPMC SORBONNE UNIVERSITÉS

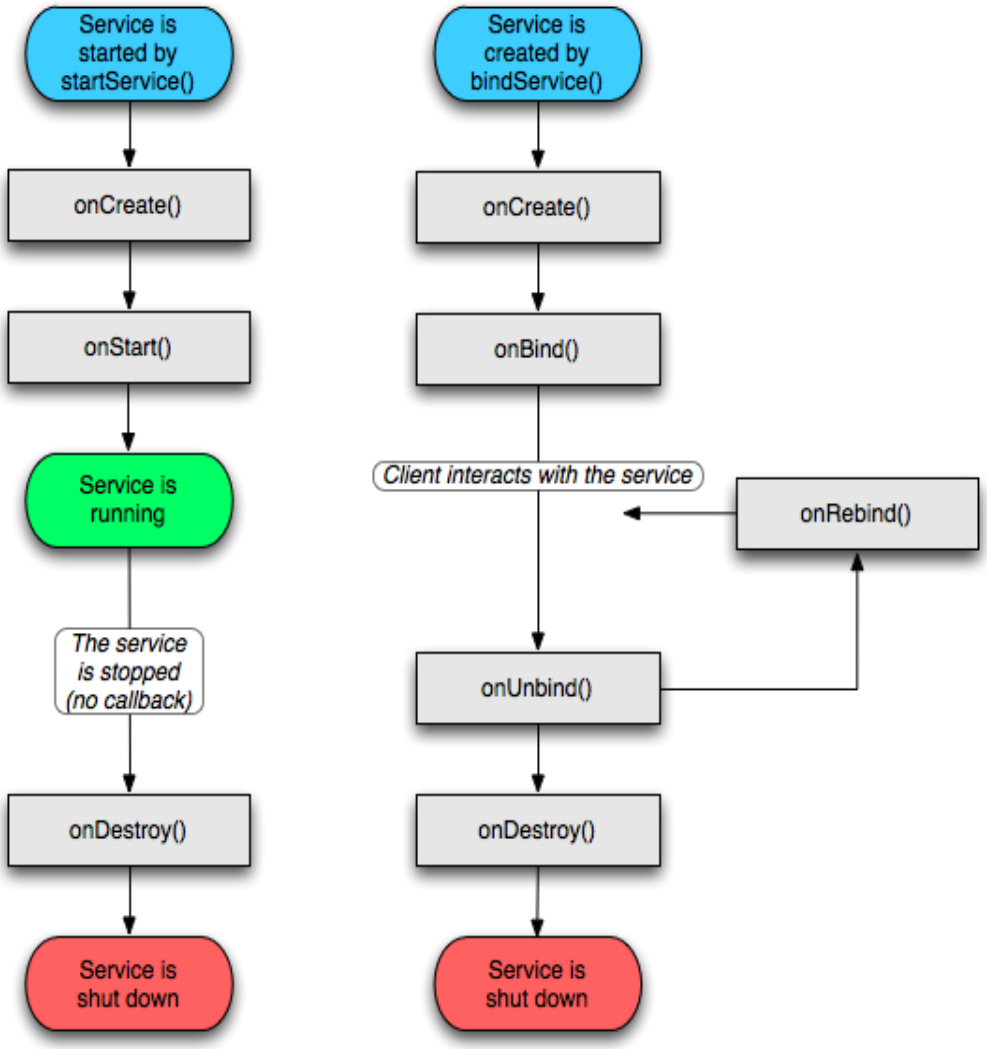
Le Client, une activity

1. **bind service**
2. **start SMS count**

telnet localhost 5554
3. sms send 12345 test



Services cycle de vie

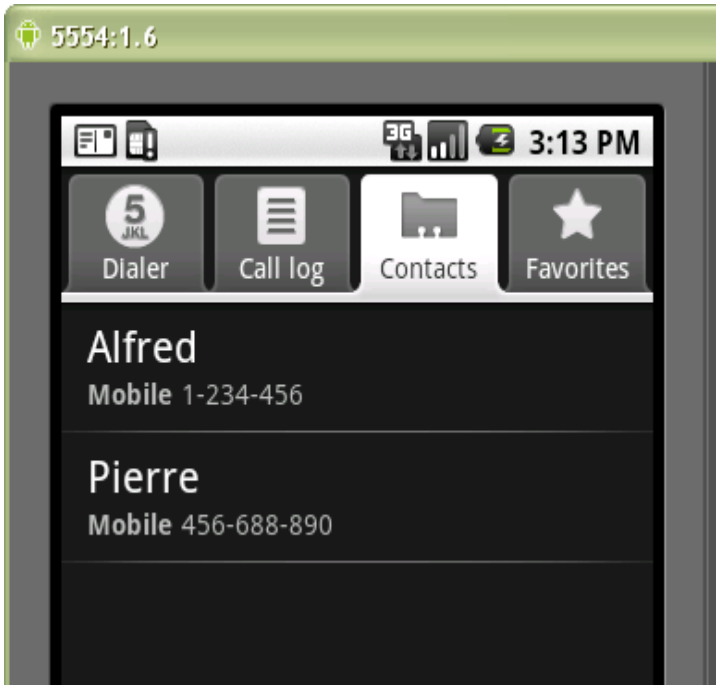


ContentProvider

- Sauvegarde et restitution de données
  - Données accessibles à toutes applications

- ContentProvider
  - Persistance
  - DAO/CRUD
    - Data Access Object
    - Create Retrieve Update et Delete

Mes Contacts, un client
le *ContentProvider* prédéfini : phone



ContentProvider

- ContentProvider, *comme les contacts*, *accès*

  – ContentResolver resolver = getContentResolver();

  Identification du bon « Contentprovider » grâce à son URI

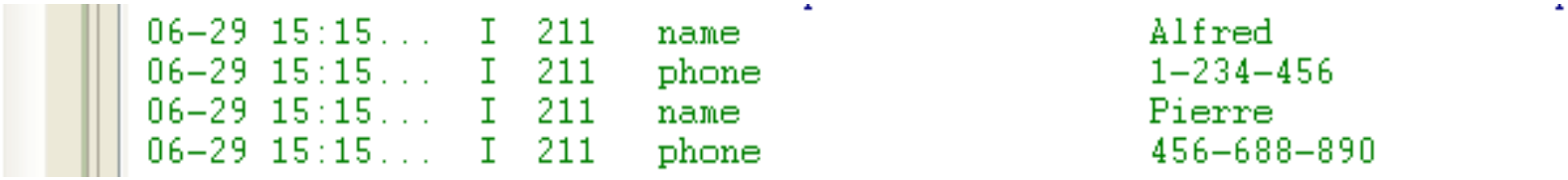
  – Uri uri = android.provider.Contacts.Phones.CONTENT_URI;
  – String s = uri.toString();
  – // assert s.equals("content://contacts/phone");

  – Appel de resolver.query
    - Pour une interrogation

  – Parcours à l'aide d'un « Cursor »

Cursor comme ResultSet

```
ContentResolver resolver = getContentResolver();
Uri uri = android.provider.Contacts.People.CONTENT_URI;
Cursor cr = resolver.query(uri,null,null,null,null);
cr.moveToFirst();
int columnName = cr.getColumnIndex(Contacts.People.NAME);
int columnPhone = cr.getColumnIndex(Contacts.People.NUMBER);
do{
  Log.i("name",cr.getString(columnName));
  Log.i("phone",cr.getString(columnPhone));
}while(cr.moveToNext());
```



Son propre ContentProvider

- public class MonPropreProvider extends ContentProvider

- Avec
  - Une Uri, l'accès à mon propre « *content provider* »
    - content://cnam.android.agenda/liste
    - content://….
  - Les méthodes
    - insert, update, delete et query
  - Et dans le fichier de configuration

```
<provider android:name="MonPropreProvider"
          android:authorities="cnam.android.agenda" />
```

Exercice final

- Construire une calculatrice de base
- Ensemble de boutons (0 => 9)
- Ensemble d'opérations (*,+,-,/)
- La calculatrice … calcule (#wow)
- Au lieu d'afficher le résultat …
- Le nombre est transformé en chaîne de caractères tous les 2 chiffres (val + 33) => ASCII
- Une page web s'ouvre avec la recherche google correspondante au résultat du calcul … transformé en chaînes de caractères

ContentProvider classe abstraite

- Réalisation,
  - implémentation des méthodes (c.f. DAO/CRUD)
    - insert, update, delete et query
  - Persistance Fichier ou base de données ?
    - Android propose SQLite
  - Un exemple de fournisseur
    - Coordonnées d'un bar servant une bière pression particulière en latitude, longitude
      - Où puis je trouver une Leffe pression, en fonction de ma position ? Itinéraire …
    - Persistance à l'aide ds SQLite
      - CREATE TABLE bars_pression (
        id integer primary key autoincrement,
        bar TEXT,
        biere TEXT,
        latitude FLOAT,
        longitude FLOAT);

BarPressionProvider, Constantes

```
public class BarPressionProvider extends ContentProvider {

    public static final
        Uri CONTENT_URI =
   Uri.parse("content://cnam.android.provider.bar/pression");

    public static final String DATABASE_NAME = "bars_pression.db";
    public static final int DATABASE_VERSION = 1;
    public static final String TABLE_NAME    = "bars_pression";

    // les champs
    public static final String ID        = "id";
    public static final String BAR       = "bar";
    public static final String BIERE     = "biere";
    public static final String LATITUDE  = "latitude";
    public static final String LONGITUDE = "longitude";

    // les indices des colonnes
    public static final int COL_ID        = 1;
    public static final int COL_BAR       = 2;
    public static final int COL_BIERE     = 3;
    public static final int COL_LATITUDE  = 4;
    public static final int COL_LONGITUDE = 5;

    private SQLiteDatabase db;
```

BarPressionProvider, create/insert

```
@Override
public boolean onCreate() {
  BarPressionSQLite dbHelper;
  dbHelper = new
   BarPressionSQLite(getContext(),DATABASE_NAME,null,DATABASE_VERSION);

  this.db = dbHelper.getWritableDatabase();
  return (db==null?false:true);
}
```

```
@Override
public Uri insert(Uri arg0, ContentValues arg1) {
  long rowID = db.insert(TABLE_NAME, "biere", arg1);
  return arg0;
}
```

BarPressionSQLite classe interne

- Help !

```
private static class BarPressionSQLite extends SQLiteOpenHelper {

        private static final String CREATE_TABLE =
          "create table " + TABLE_NAME + " ( " +
                    ID + " integer primary key autoincrement, " +
                   BAR + " TEXT, "  +
                 BIERE + " TEXT, "  +
             LATITUDE  + " FLOAT, " +
             LONGITUDE + " FLOAT);";

public BarPressionSQLite(Context context, String name,
                         CursorFactory cursor, int version){
       super(context, name, cursor, version);
    }


@Override
public void onCreate(SQLiteDatabase db){
  db.execSQL(CREATE_TABLE);
}
```

Le Client + LogCat

```
ContentResolver resolver = getContentResolver();
Uri uri = cnam.android.provider.BarPressionProvider.CONTENT_URI;

ContentValues cv = new ContentValues();
cv.put(BarPressionProvider.BAR, "Le Bastille");
cv.put(BarPressionProvider.BIERE, "Leffe");
cv.put(BarPressionProvider.LATITUDE, 48.853668);
cv.put(BarPressionProvider.LONGITUDE, 2.370319);
Uri uri2 = resolver.insert(uri, cv);
```

```
07-02 09:54...  I  527  ActivityThread          Publishing provider cnam.android.provider.bar: cnam.android.prov
07-02 09:54...  V  527  BarPressionProvider     onCreate( db==null :false)
07-02 09:54...  I  527  provider                OnCreate
07-02 09:54...  V  527  BarPressionProvider     insert(content://cnam.android.provider.bar/pression, biere=Leffe
07-02 09:54...  I  52   NotificationService     enqueueToast pkg=cnam.android.provider callback=android.app.ITra
```

Publication du Provider par :
Fichier AndroidManifest.xml, dans <application

```
<provider android:name=".BarPressionProvider"
        android:authorities="cnam.android.provider.bar" />
```

UPMC SORBONNE UNIVERSITÉS