Android Introduction

Notes de cours

Sommaire

- **Un peu d'historique**
- **Andoid OS comme middleware**
- **Du Java sans JVM**
- **Principes de base**
- **Ces notes représentent une introduction …**

Bibliographie utilisée

Le livre de Mark Murphy chez Pearson

…

Le livre écrit par Florent Garin

Android

Le cours de Victor Matos

http://grail.cba.csuohio.edu/~matos/notes/cis-493/Android-Syllabus.pdf

Plusieurs livres

Android A Programmers Guide - McGraw Hill

Professional Android Application Development - Wrox

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. "

Qu'est-ce que 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)**
    - **Un canevas (framework) pour différents types d'applications**
  - **Nombreuses librairies**
    - **IHM,**
    - **Téléphonie,**
    - **Multimédia,**
    - **Capteurs,**
    - **Internet, cartographie**
    - **…**

Pourquoi Android ?

- **Indépendant d'une architecture matérielle**
- **Dédié aux systèmes embarqués et pas seulement**
- **Ambitions de Google/Apple**
  - **Marketing**
  - **Retard : au 25 Juin 2010**
    - **67000 applications sur AndroidMarket**
    - **270000 sur AppStore**

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/

Android les grandes lignes

- **Composants Android**
- **Outils de Développement**
- **Architecture Logicielle**

- **Développement**
  - **en java avec quelques directives en syntaxe XML**

- **Deux exemples**
  - **Démonstration**
    - **Un navigateur en deux appels de méthodes**
    - **Un MVC façon Android**

Composants Android

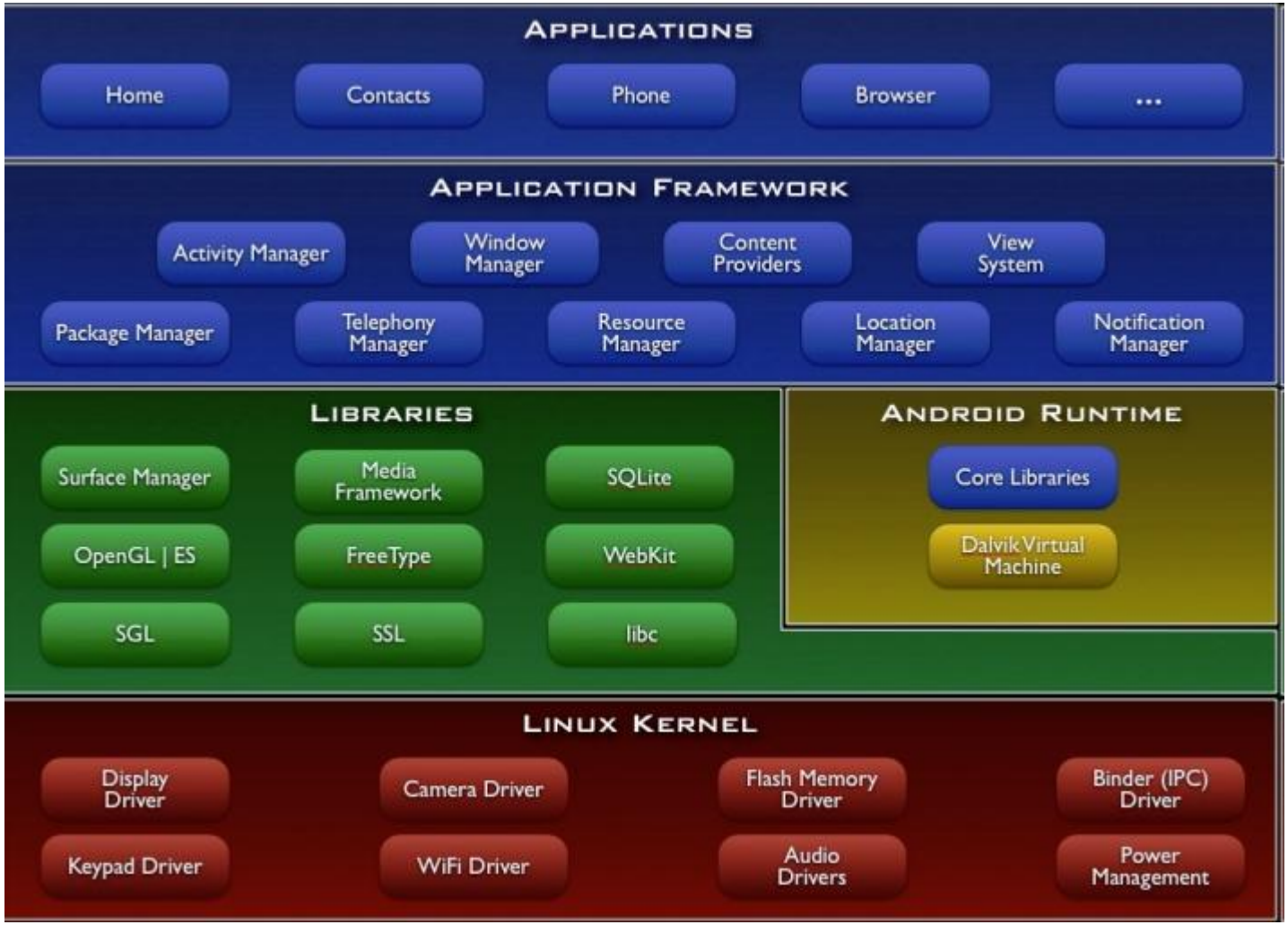
- **Framework de déploiement d'applications**
- **Dalvik comme machine virtuelle** (à registres != JVM de Sun à pile)
- **Navigateur intégré, WebKit** (webkit utilisé par safari, Google Chrome…)
- **SQLite**
- **Multimédia support PNG, JPG, GIF, MPEG4, MP3, H.263**

- **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**

Composants Android



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

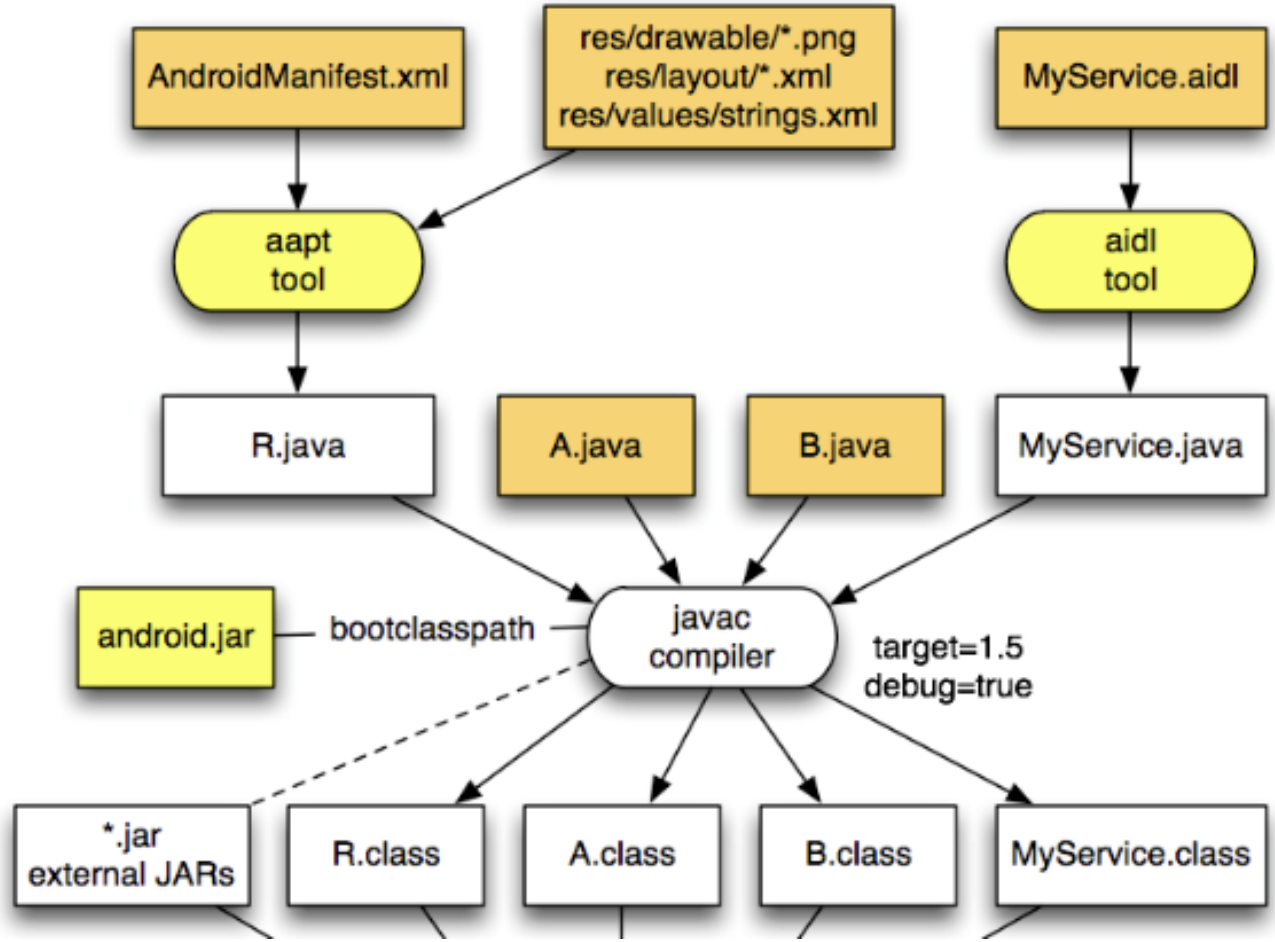
Développement 1/2

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

Développement 2/2

- **Du .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

En détail, développement 1/2
Compilation : obtention des .class



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

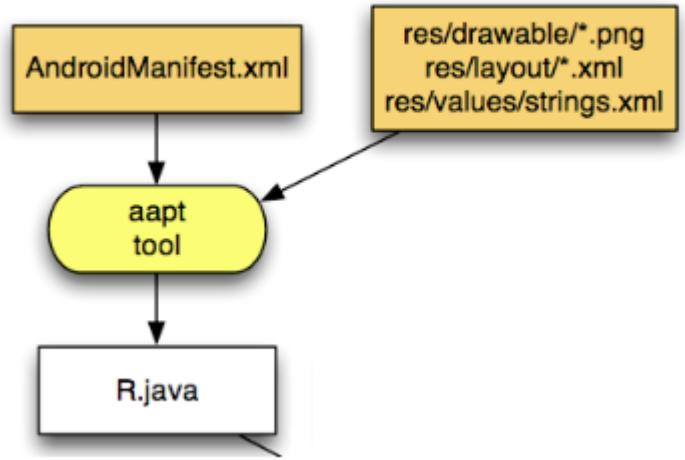
Développement 1/2
Fichier de configuration, AndroitManifest.xml

```
<?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=".BrowserDemo"
                  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>
```

Le fichier de Ressources, R.java

- **res/layout/main.xml**

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



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

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

Un Source java, juste pour la syntaxe…

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

public class BrowserDemo extends Activity {
    ….
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        ….
}
}
```

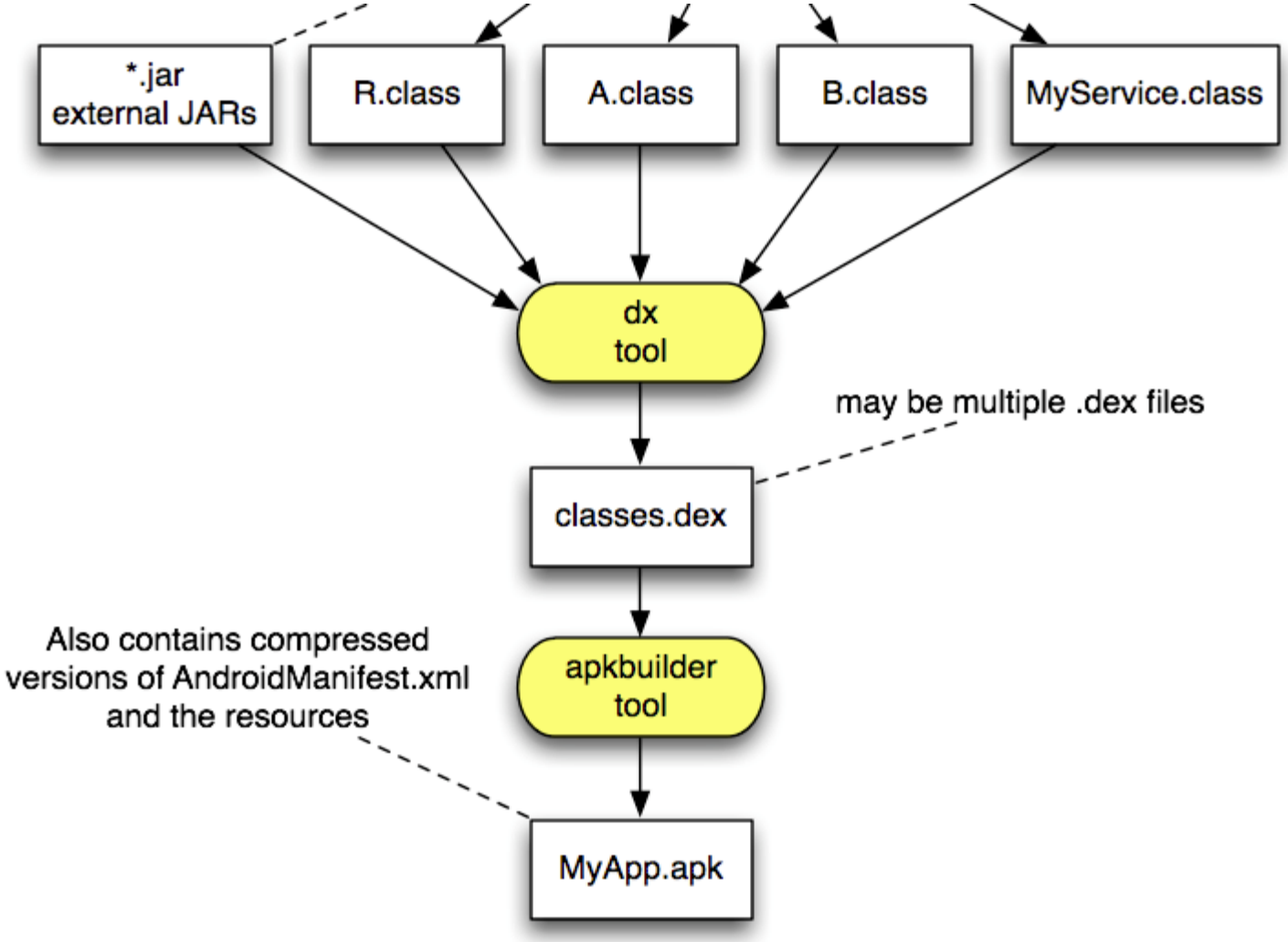
Développement 2/2

- **Du .class en .dex**
  - Assemblage de tous les .class vers un .dex
    - Un décompilateur : **http://dedexer.sourceforge.net/**
  - Une machine par application, un processus Linux
  - Les applications communiquent via l'intergiciel

- **Génération de l'application .apk**
  - **Assemblage de**
  - **Une archive signée**

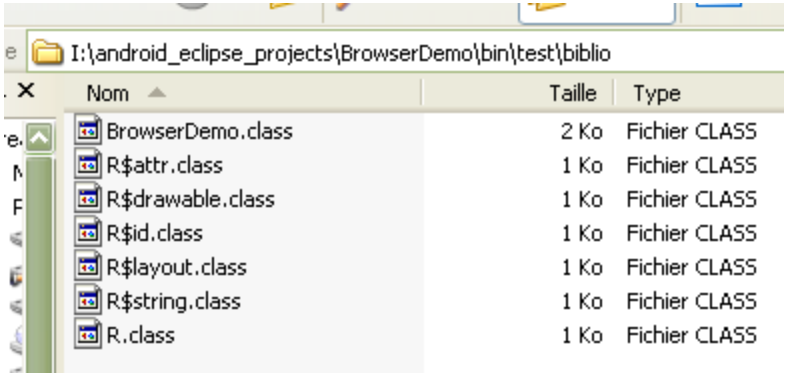
  - Téléchargement : émulateur ou mobile

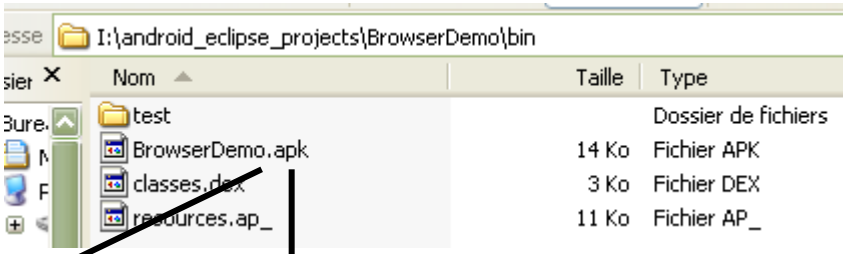
Développement 2/2 Génération de l'application

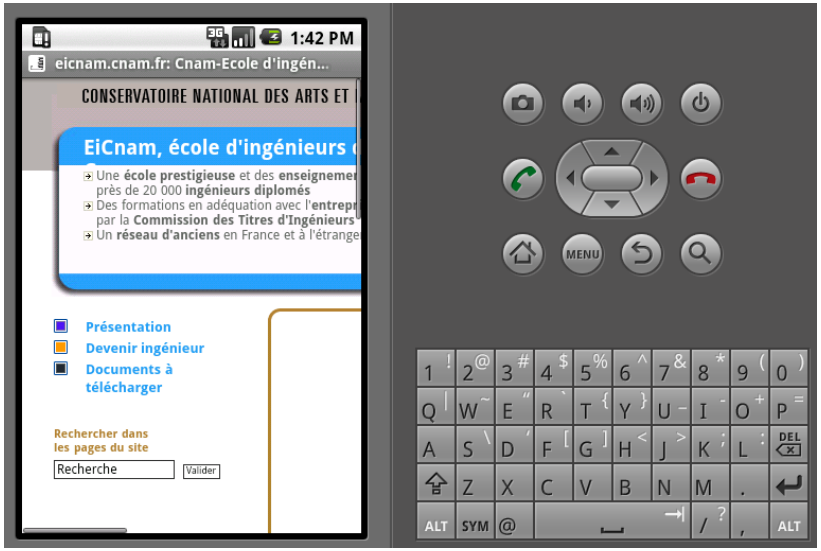


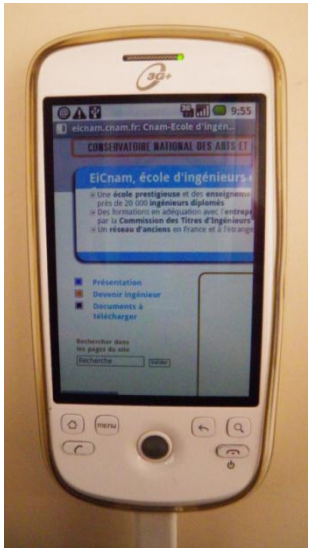
Source : http://stuffthathappens.com/blog/wp-content/uploads/2008/11/android_flow.png

Exécution émulateur et/ou mobile .apk









Android OS

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

Middleware Android OS, un extrait

- **View** System listes, boutons,… navigateur (WebView)
- **Resource Manager,** accès aux String, aux descriptifs de l'ihm
  - R.java …
- **Activity Manager** gestion du cycle de vie d'une application
  - Une application android
- **Content Providers** Accès aux données d'autres applications, partage de données
  - **Notification Manager** autorise des alertes dans la barre de statut
  - TelephonyManager
  - ….
  - http://developer.android.com/guide/basics/what-is-android.html

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**

Introduction aux Applications Android

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

- **Une Application hérite le la classe Activity**
  - Surcharge de certaines méthodes,
    - Du déjà vu : MIDlet, Applet, Servlet,…
  - Le cycle de vie est imposé par le framework
    - Déjà vu : pour une Applette init() puis …

Avertissement

- **C'est une introduction**
  - Le vocabulaire
  - Les grandes lignes
  - Quelques analogies seront faites avec du java « traditionnel »

- **Les Essentiels**
  - Activity
  - BroadcastReceiver
  - Service
  - ContentProvider

Introduction … Classes

- **Activity**
  - Une interface utilisateur
  - 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 SQLLite

Deux exemples, deux Activity

1. **Usage de WebKit**
   - Installation d'un navigateur en 2 lignes (WebView)

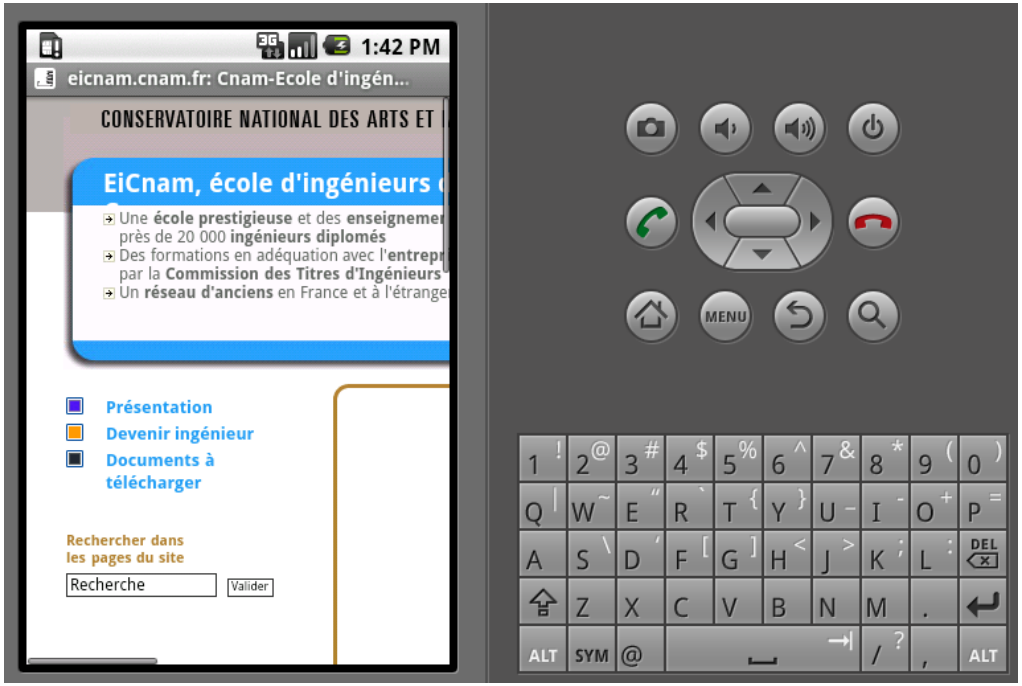
2. **Une IHM**
   - Un bouton, un écouteur, un clic et l'heure est affichée !

   - À télécharger ici
     - http://jfod.cnam.fr/seja/android/exemples/

Activity Usage du WebKit, 2 lignes

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

} 2 lignes



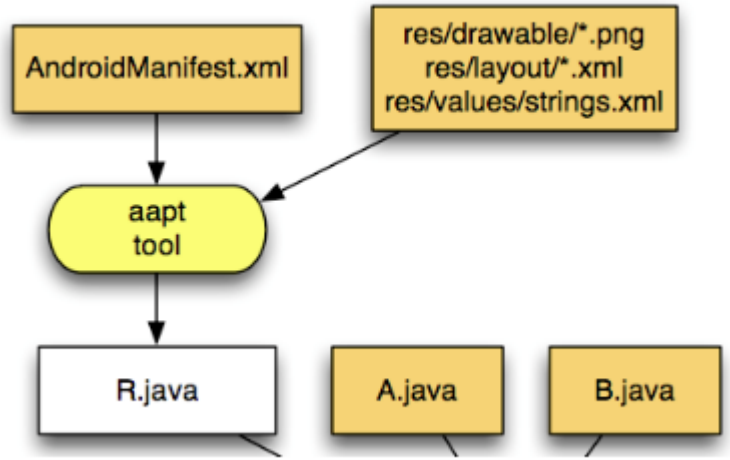
OnCreate est déclenché par le framework Android

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

        browser=(WebView)findViewById(R.id.webkit);
        browser.loadUrl("http://eicnam.cnam.fr/");
    }
}
```

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

Une configuration XML

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



Activity Un Click et l'heure est actualisée

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


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

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

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

Notes : Vue apparentée swing, sans descrpiption XML de l'IHM
Ici un MVC à lui tout seul …

Un clic et l'heure… AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="cnam.intro"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".Now"
                  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>
```

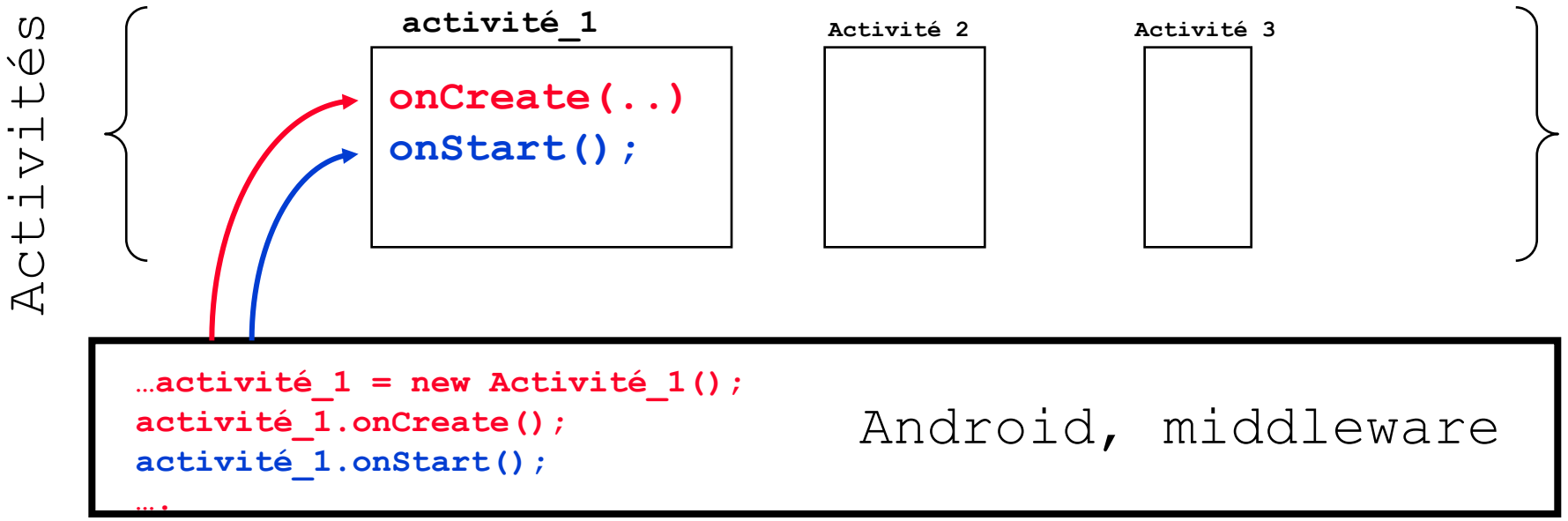
Activity

- **Une Vue**
- **En inversion de contrôle,**
  - **Méthodes onStart()…. Déclenchées par le middleware**
- **Diagramme d'états**

public class 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();
    }
```

Inversion de Contrôle… Rappel

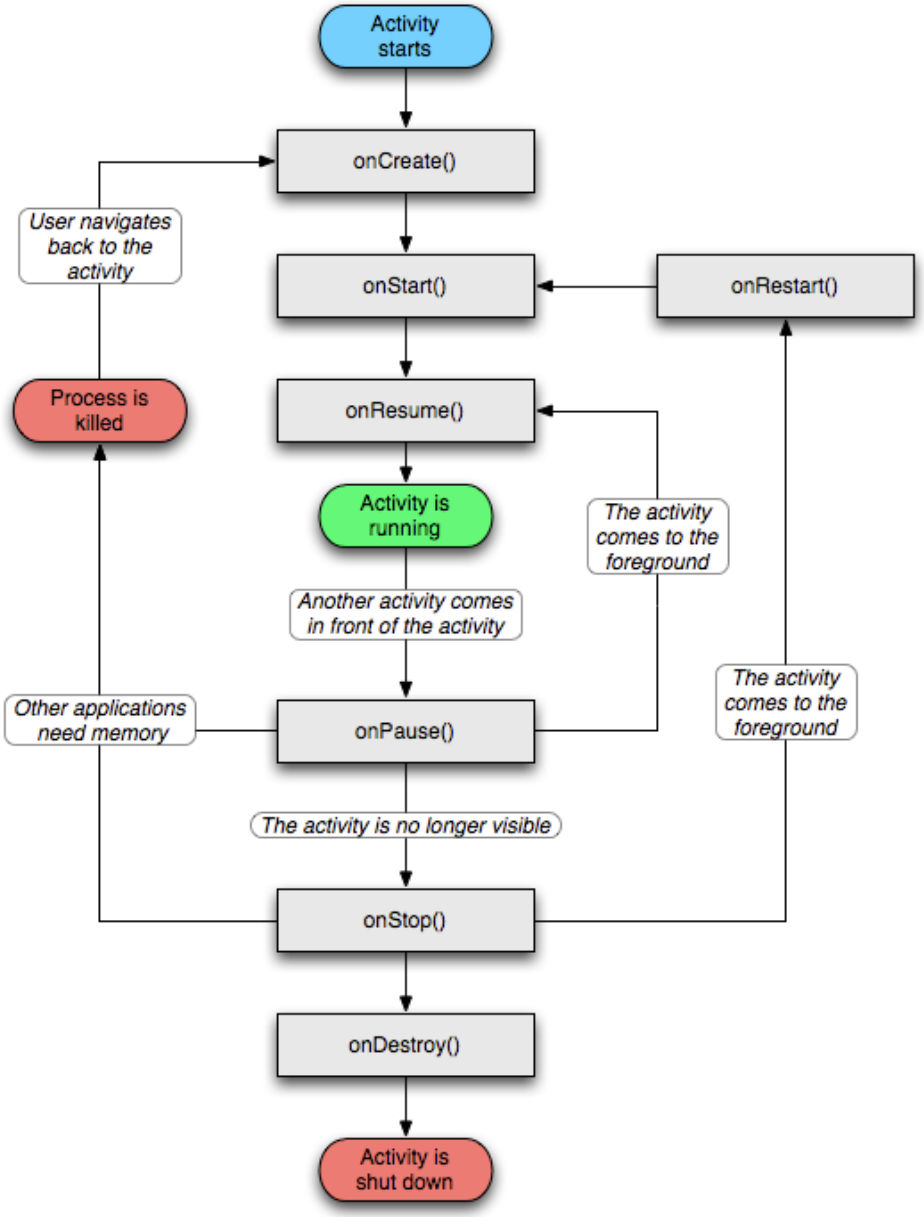



```
public class Activity extends ApplicationContext {
      protected void onCreate(Bundle savedInstanceState);
      protected void onStart();
       …..}
```

1

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

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



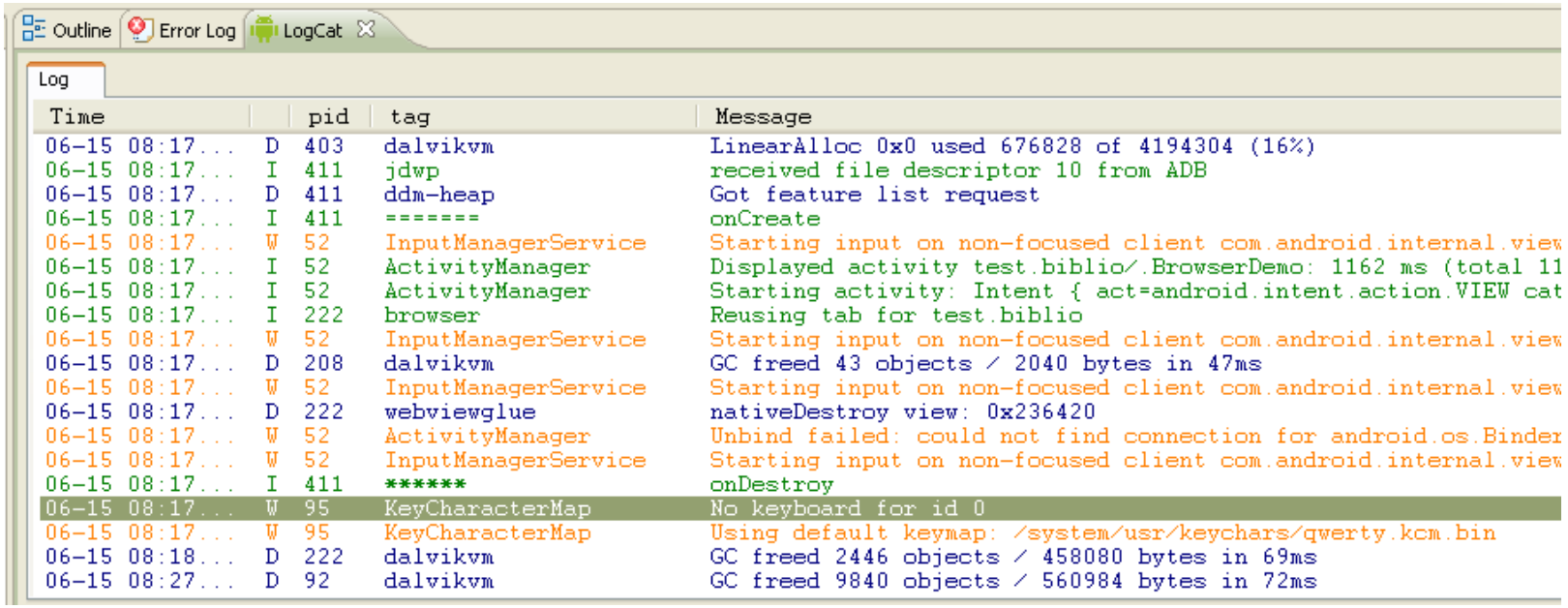
C'est le framework
qui contrôle tout

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

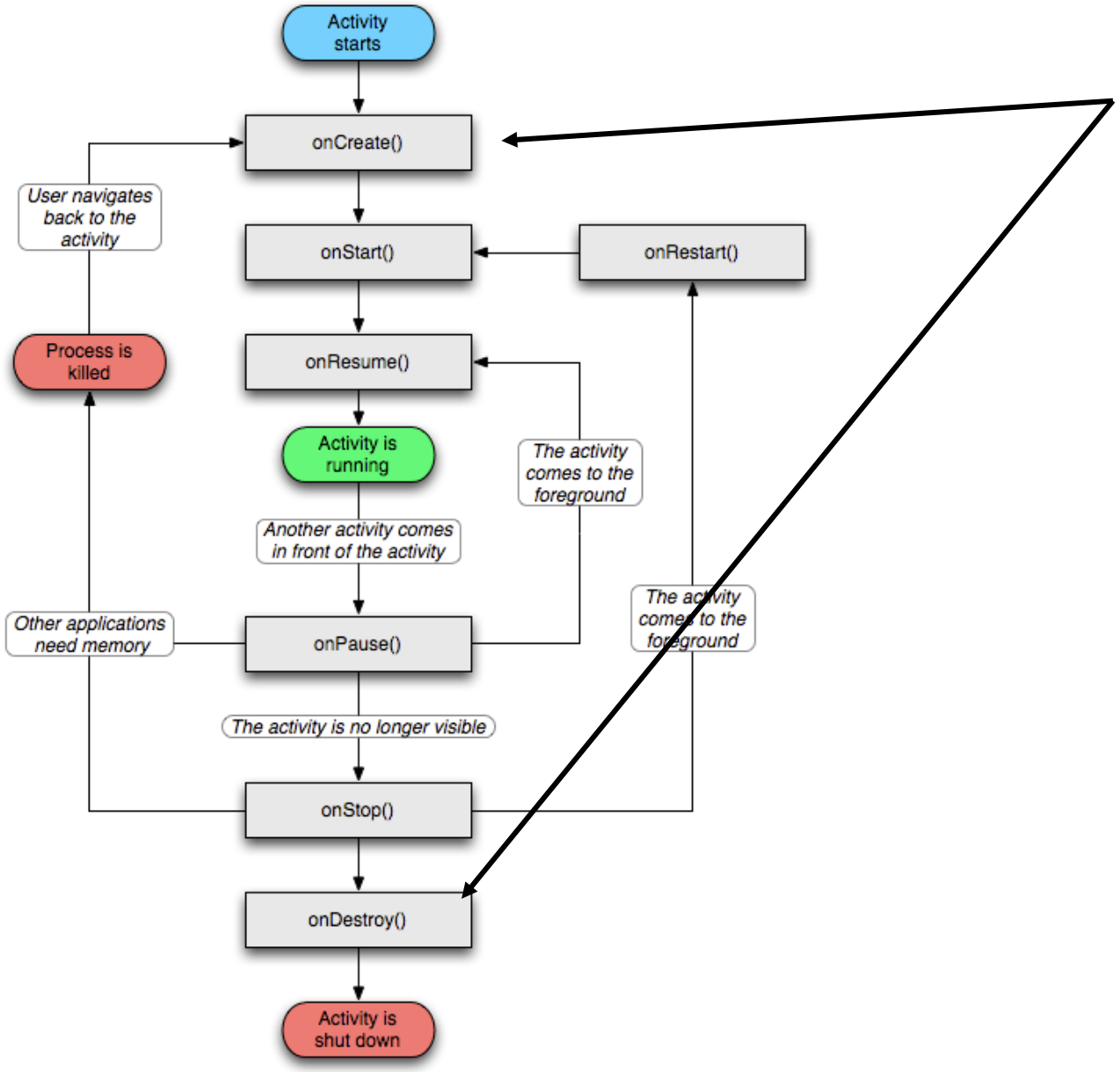
onResume
activité au 1er plan = Dépiler()

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

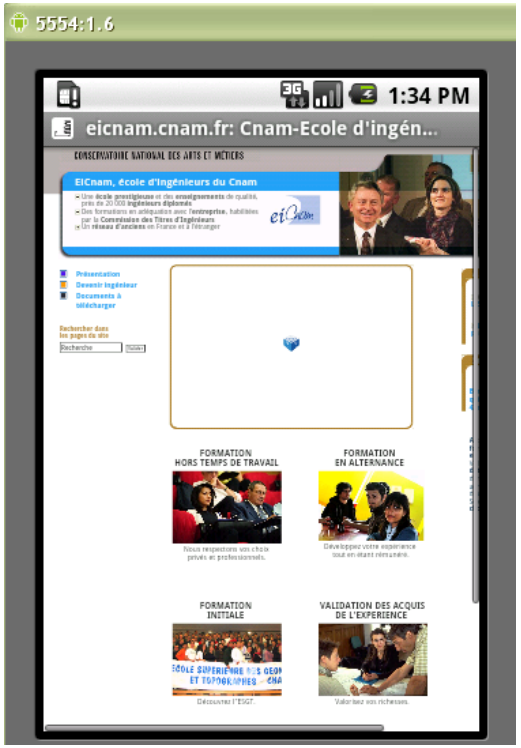


Démonstration, Activity dans tous ses états

```
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");
    }
}
```

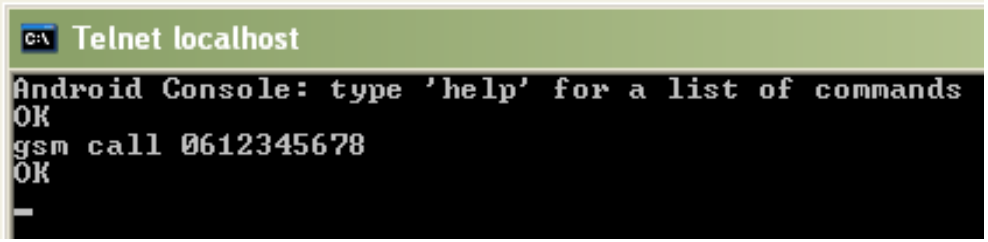
OnPause -> onResume

1)

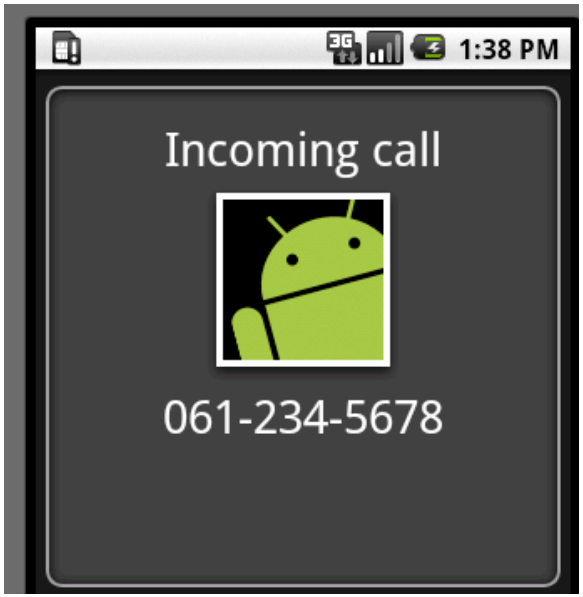


OnPause

2-1) `telnet localhost 5554`



2-2)

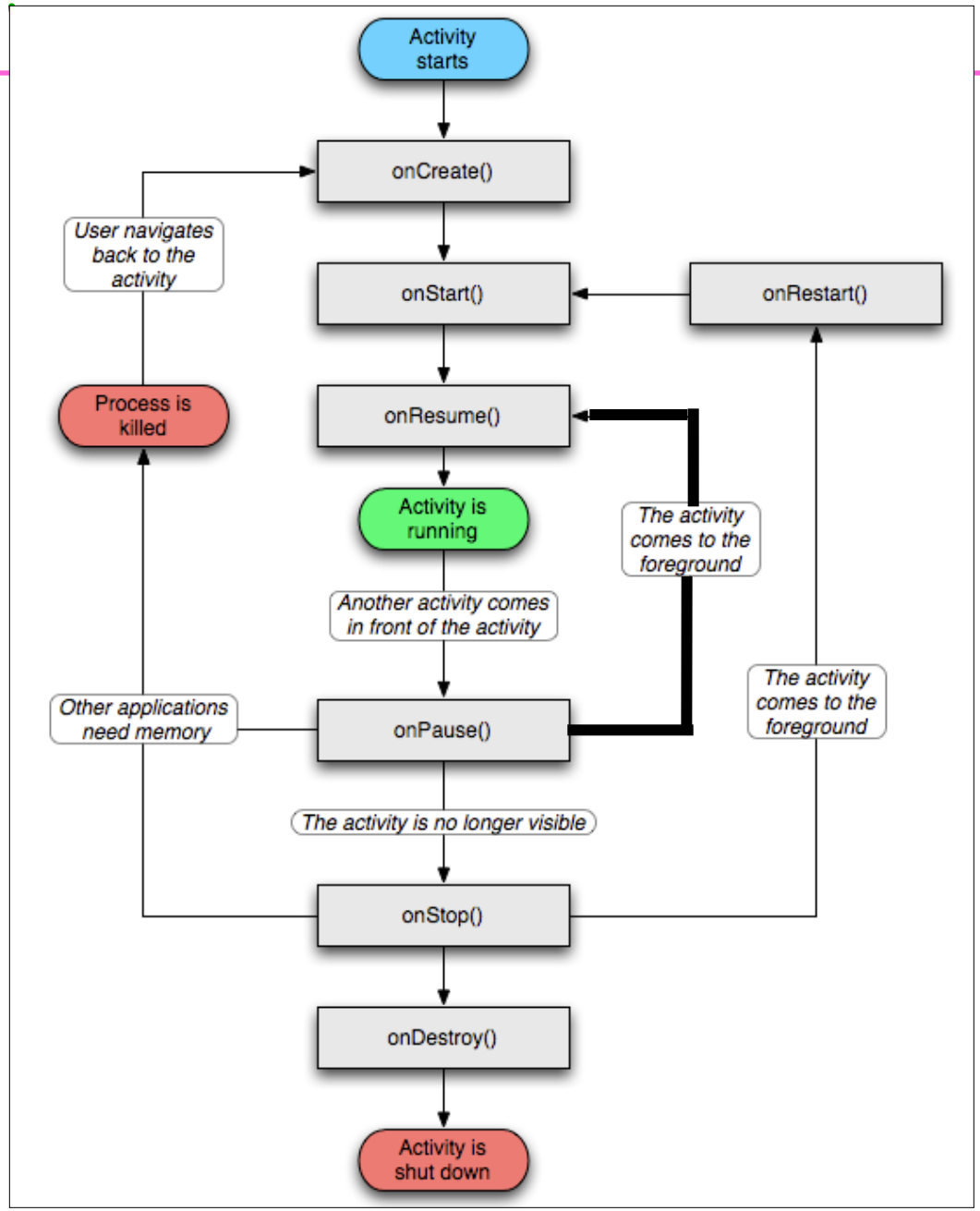


2)

OnResume

3)

En résumé :


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

Sauvegarde, restitution, Bundle

- **Un Bundle est une map,**
  - une table dont
    - les clés sont des String,
    - les valeurs des objets (Parcelable vu ensuite <T extends Parcelable>)

- **Sauvegarde**

```
@Override
public void onSaveInstanceState( Bundle out){
  out.putString("CurrentDate", new Date().toString());
  super. onSaveInstanceState(out);
}
```

- **Restitution**

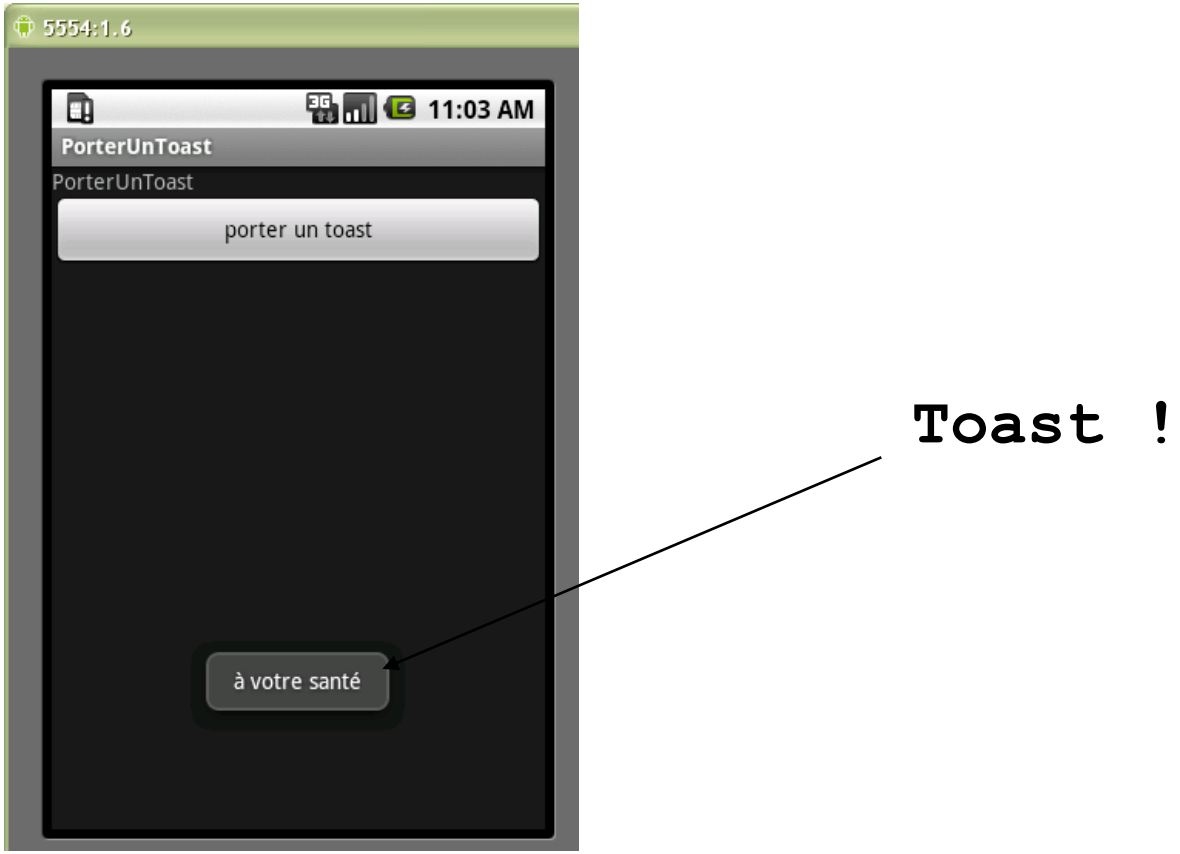
```
@Override
public void onRestoreInstanceState( Bundle save){
  out.putString("CurrentDate", new Date().toString());
}
```

Attention au cycle de vie, onRestoreInstanceState à vérifier

Communication inter-activités

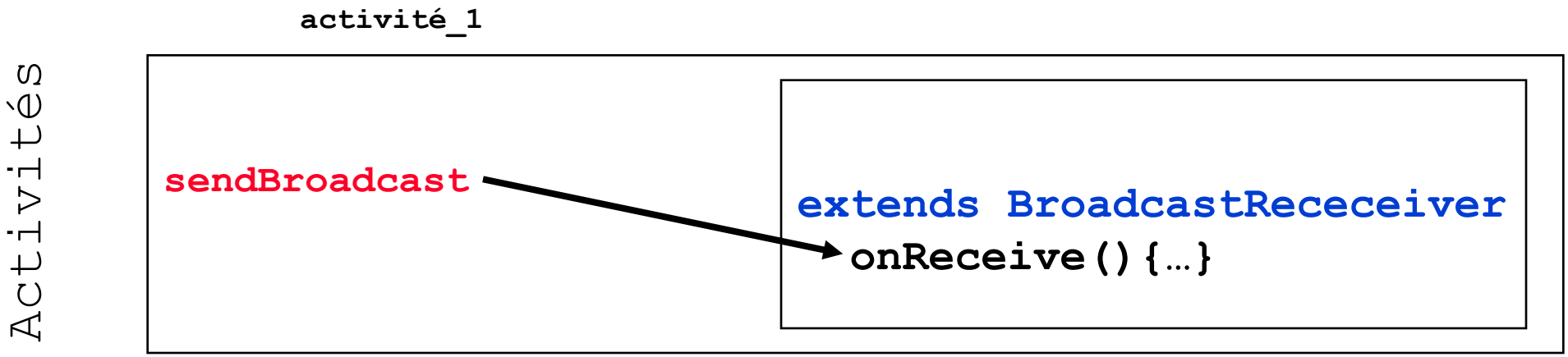
- **Communication comme intention**
  - **Entre deux applications, entre deux dalvik**
  - **Abonnements**

Intention de … porter un toast



- **Intent**

Android : Intents, comme Intention



Il faut :

- **Un receveur**

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

- **Enregistrement auprès du middleware**

- **Une intention d'émettre**

Intention de

```
Un receveur, qui porte un toast
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 :
 Receiver receiver = new Receiver();
 registerReceiver(receiver, new IntentFilter("PORTER_UN_TOAST"));
 registerReceiver(receiver,new IntentFilter(Intent.ACTION_AIRPLANE_MODE_CHANGED));
```

```
Émission de l'intention :
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*

À votre santé !


```
06-16 11:06...  I  52   ActivityManager          Displayed activity cnam.eicnam/.PorterUnToast:
06-16 11:06...  I  52   NotificationService      enqueueToast pkg=cnam.eicnam callback=android.
```

Intention de… 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

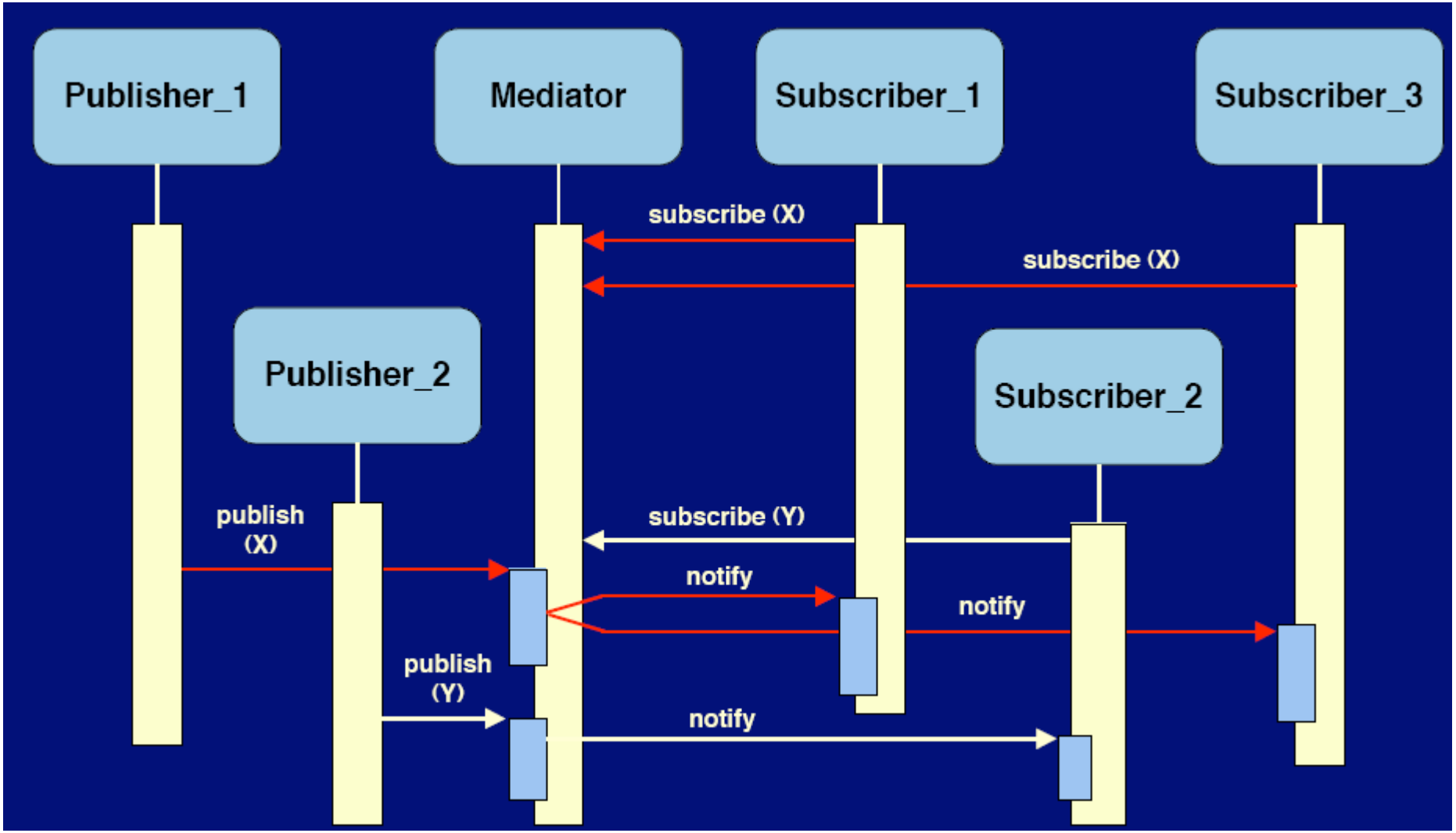
```
private static final String SMS_RECEIVED = "android.provider.Telephony.SMS_RECEIVED";
```

Et

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

Je suis maintenant abonné aux réception de SMS

Publish-Subscribe/Intent & Context



- Extrait de http://www2.lifl.fr/icar/Chapters/Intro/intro.html

Sous Android

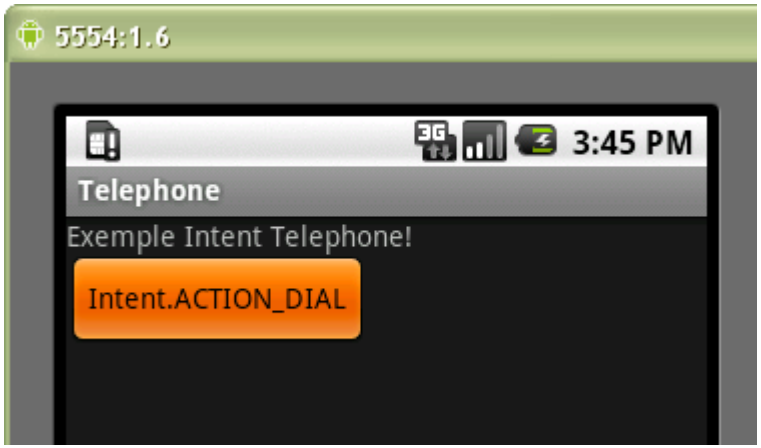
- **Mediator, classe Context**
  - http://developer.android.com/reference/android/content/Context.html
- **Subscriber, classe BroadcastReceiver**
  - http://developer.android.com/reference/android/content/BroadcastReceiver.html
- **X,Y les thèmes, classe Intent**
  - http://developer.android.com/reference/android/content/Intent.html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

Intention de téléphoner, à la pause …

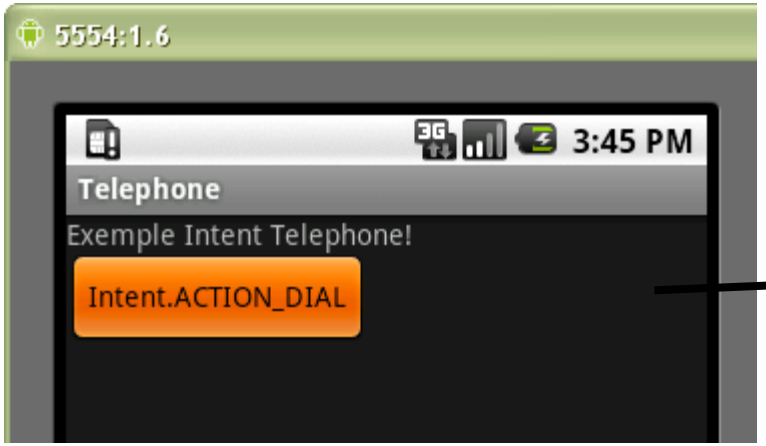
- **Intent**
- **Comme Communication**
- **Démarrer une activité prédéfinie : téléphoner**

Intention de téléphoner

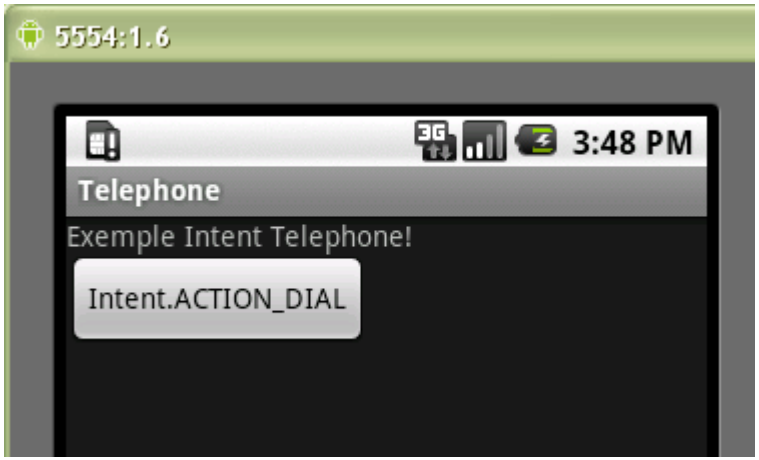
```
public class Telephone extends Activity implements  OnClickListener {

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***

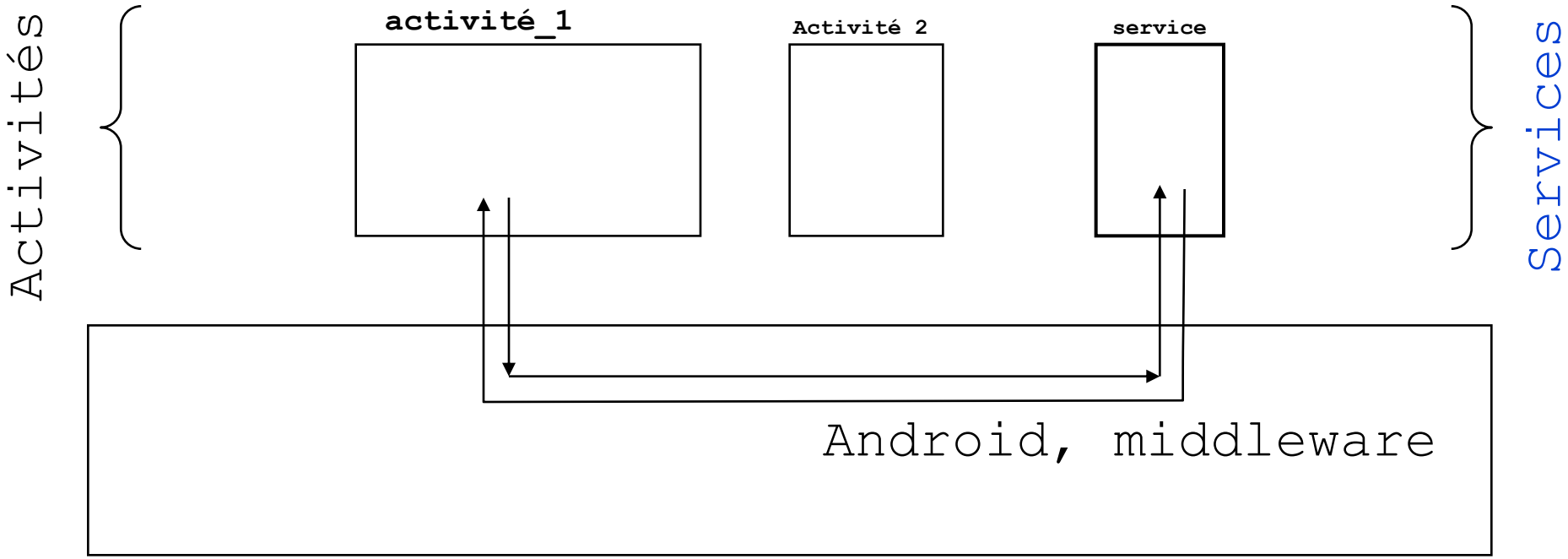
Activités un résumé

- **Activity**
  - Cycle de vie
  - Gérée par le framework
  - Une application est constituée de plusieurs activity (et services)
- **startActivity**
  - Une activité peut en démarrer une autre
- **Communication**
  - Intent

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



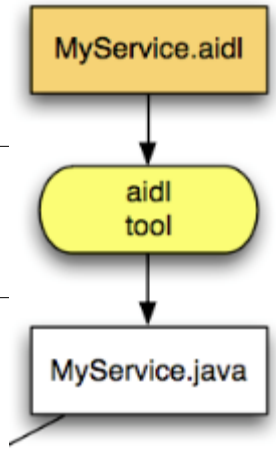
- **Service distant,**
  - **Découverte du service par une intention …**

SMSService : un exemple

- **Un compteur de SMS entrants**
  - A chaque sms un compteur est incrémenté
  - 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()");

    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++;
}
}
```

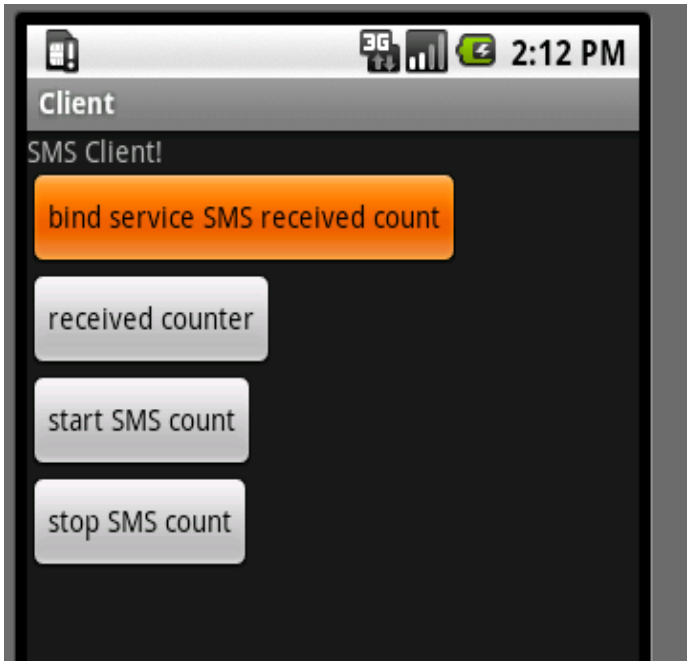
Le service 3/3, La souche le stub fournie à la demande

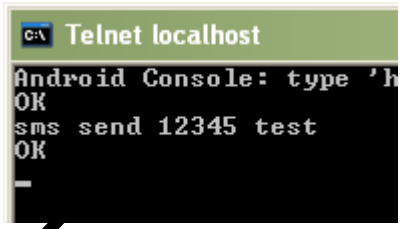
```
public IBinder onBind(Intent arg0) {
  return new SMSServiceStubImpl();
}


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;
  }
}
```

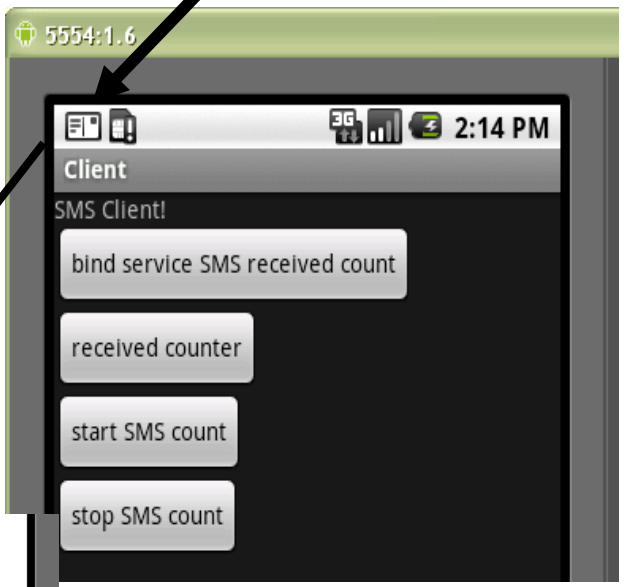
Le Client, une activity

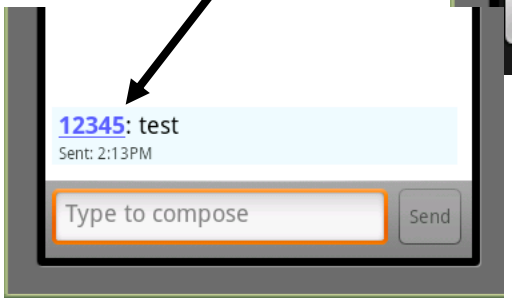
1. bind service
2. start SMS count

telnet localhost 5554
3. sms send 12345 test







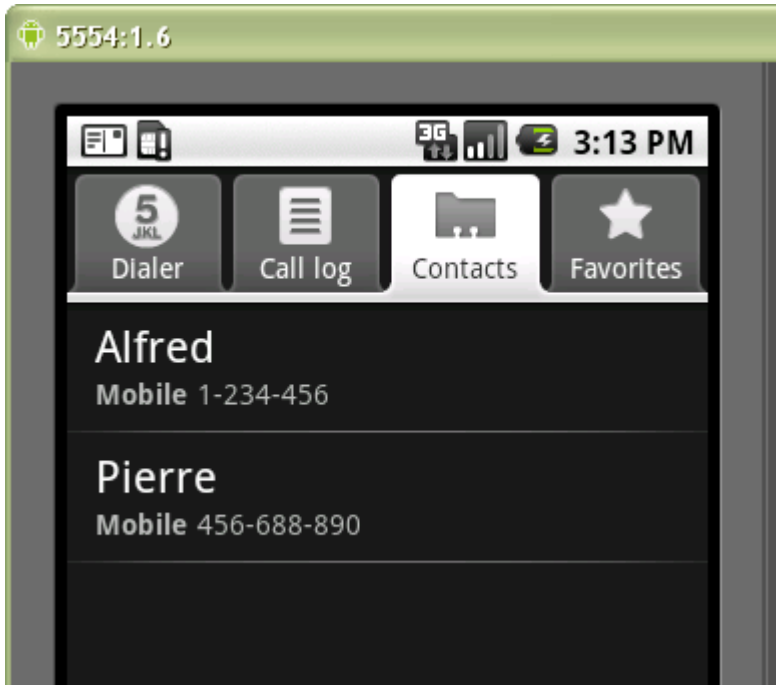


Démonstration…

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());
```

```
06-29 15:15...  I  211   name                 Alfred
06-29 15:15...  I  211   phone                1-234-456
06-29 15:15...  I  211   name                 Pierre
06-29 15:15...  I  211   phone                456-688-890
```

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" />
```

ContentProvider est une 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;
}
```

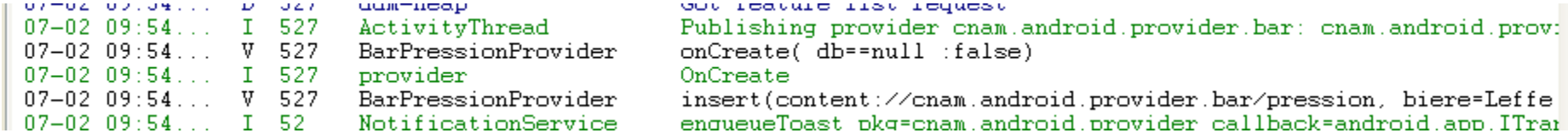
BarPressionSQLite est une classe interne

- **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);
```



Publication du Provider par :

Fichier AndroidManifest.xml, dans <application

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

BarPressionProvider, query, update

- **En exercice !**

Sqlite3, un outil d'interrogation de la bdd

```
I:\android-sdk-windows\tools>adb -s emulator-5554 shell
# sqlite3 data/data/cnam.android.provider/databases/bars_pression.db
sqlite3 data/data/cnam.android.provider/databases/bars_pression.db
SQLite version 3.5.9
Enter ".help" for instructions
```

```
sqlite> .dump BARS_PRESSION
.dump BARS_PRESSION
BEGIN TRANSACTION;
CREATE TABLE bars_pression ( id integer primary key autoincrement, bar TEXT, biere TEXT, latitude FLOAT, longitude FLOAT);
INSERT INTO "bars_pression" VALUES(1,'Le Bastille','Leffe',48.853668,2.370319);
COMMIT;
sqlite>
```