Module Architecture des systèmes mobiles : Android

première partie - 2015 – Didier FERMENT - UPJV

Plan – 1ére partie :

Introduction
Première Activité :
- layout
- classe Activity
- manifeste

développement avec un IDE :
- projet
- AVD

Intention :
- appel explicite d'activité
- appel avec retour résultat
- intention implicite

Cycle de vie d'une activité
Composants Graphiques *(vite)*
- ListView

Filtre d'intention :
- action, catégorie, Uri/type
- PackageManager,
- intention différée

Permission
Toast, Notification
Content Provider et SQLite

Ressources :

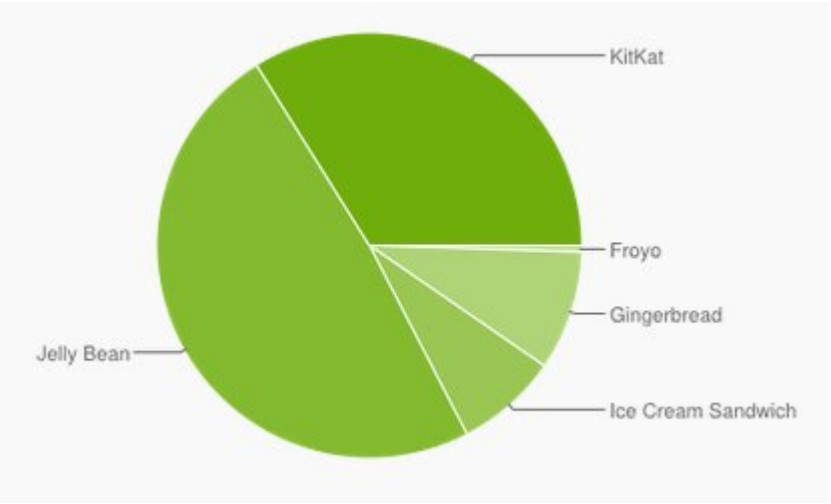
cours, TD/TP, code http://www.u-picardie.fr/~ferment/android
http://developer.android.com/ tout, tout, tout et download
http://www.vogella.com/android.html tutoriel
http://saigeethamn.blogspot.fr/ tutoriel
http://www.franck-simon.com cours très complet
l'Art du développement, M. Murphy, Pearson livre simple
Pro Android 4, S Komatineni D MacLean, Apress livre complet

Seconde partie avec M Christophe LOGE :
Les handlers, les asynctasks, les services, les broadcastReceivers, …
… de la programmation concurrente

Introduction (1/2) : Histoire … courte

2005 : Google rachète le développement d'une startup
2007 : création d'un consortium entre Google et des entreprises du mobile.
2008 : Le source du SDK 1.1 (Api level 2) disponible sous licence Apache, premier smartphone Android aux USA, lancement de l'Android Market
2009 : SDK 1.5 (Api level 3) ajouts importants : enregistrement vidéo, App Widgets, détection de rotation, de l'accéléromètre, … puis les différentes résolutions
2010 : les versions 2.X pour les smartphones (Api level 7 ...) Froyo et Gingerbread : HTML5, Wifi, stockage externe, NFC, VoIP, SIP, … ; Mai 2011 : 2.3.4 (Api level 10)
2011 : les versions 3.X pour les tablettes (Api level 11 à 13) Honeycomb
Octobre 2011 : fusion des 2 branches version 4.0 (Api level 14) Ice Cream Sandwich : gestion des écrans par fragment
Novembre 2012 : Jelly Bean 4.2 (Api level 17)
Novembre 2013 : KitKat 4.4 (Api level 19)
Octobre 2014 : Lollipop 5.0 (Api level 21) abandon de la Dalvik machine pour l'ART runtime

état au décembre 2014 : 85% des ventes
source http://developer.android.com/about/dashboards/index.html



<u>Les concurrents :</u>
Apple IOS 13% : propriétaire, payant, Objective-C, moins de PB de compatibilité !, Api mieux documenté
Tizen : début 2013, open OS, développé par Samsung, pour des web applications HTML5, ….

Introduction (2/2) : Architecture Android

Noyau Linux

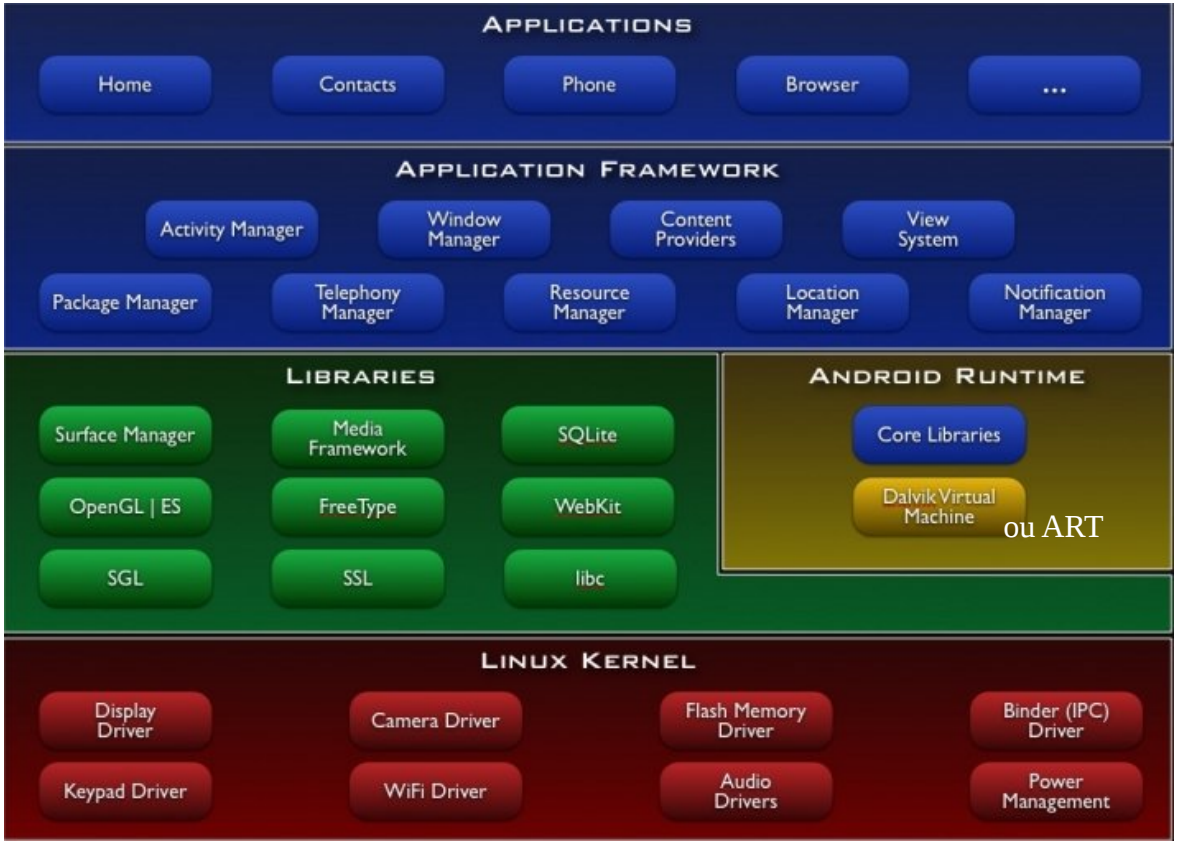
branche dérivé du noyau 2.6 avec la gestion des processus, des threads, de la mémoire, des drivers, de la sécurité, du réseau ...

Android Runtime
= ART jvm + Core librairies JAVA

Chaque application Android fonctionne dans son propre processus avec son instance de Java Virtuelle Machine (ART) qui exécute des ".dex" issues de la compilation du langage JAVA. Elle est écrite pour "fork-er" économiquement.

C/C++ Libraries

System C library : une BSD glic(libc)
Media Libraries : basé sur OpenCORE
Surface Manager gère l'écran, la 2D et la 3D
WebKit browser
OpenGL
FreeType pour le rendu des fontes
SQLite le SGBD



Source : http://developer.android.com/

Framework Android

La programmation d'application se fait dans le cadre d'un Framework apportant ses contraintes mais aussi des managers, des providers, …..
Le principe est la ré-utilisation des composants et leur coopération.

Une première Activité (1/11)

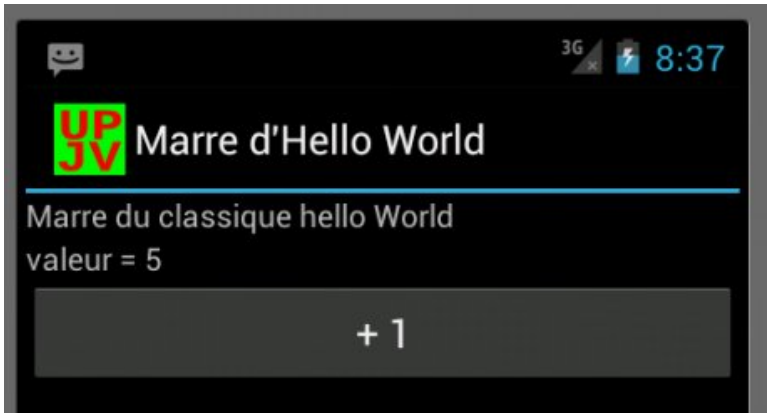
MarreDHelloWorldActivity

Pour importer puis exécuter les exemples, il suffit de les copier dans votre répertoire AndroidStudioProject !

Fichier layout de l'UI

fichier source res/layout/activity_marre_dhello_world.xml

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

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/text_invit" />

    <TextView
        android:id="@+id/val"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="" />

    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="+ 1"
        android:onClick="actionPlus1" />

</LinearLayout>
```



fichier de positionnement XML qui décrit l'interface utilisateur

le conteneur est une boite verticale comprenant :
une zone de texte
une autre zone de texte nommée "val"
un bouton affichant "+ 1" et déclenchant la méthode `actionPlus1`

Une première Activité (++ 2/11)

Fichier layout de l'UI

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/text_invit" />
    <TextView
        android:id="@+id/val"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="" />
    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="+ 1"
        android:onClick="actionPlus1" />
</LinearLayout>
```

fichier de positionnement XML :
même technique que XUL (XML user Interface) de Mozilla, Flex d'Adobe ou XAML ou GWT

le namespace est obligatoire à la racine
xmlns:android="
http://schemas.android.com/apk/res/android

Dans cet exemple, le gestionnaire de positionnement est une boite conteneur verticale.
Les attributs précisent les comportements

Il est possible de définir l'interface graphique dynamiquement dans le code JAVA

L'inflation : opération de création d'une UI à partir d'un fichier XML

Une première Activité (++ 3/11)

Fichier layout de l'UI

```
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/text_invit" />
<TextView
    android:id="@+id/val"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="" />
<Button
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="+ 1"
    android:onClick="actionPlus1" />
```

le premier widget "zone de texte non éditable" prend sa valeur dans le fichier res/values/string.xml sous le nom text_invit
@string/text_invit est une référence à une ressource string de nom "ext_invit" : cad fichier res/string.xml élément d'attribut name "text_invit"

Le second widget zone de texte est identifié par "val"
La notation @+id/val signifie une référence à l'identifiant val; + : s'il n'existe pas, elle est crée
Cela permet de récupérer une référence sur l'objet widget ainsi :
`(TextView)findViewById(R.id.val);`

le widget bouton possède un attribut onClick précisant le nom de la méthode qui traitera les événements "click"

fichier res/values/string.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">Marre d\'Hello World</string>
    <string name="text_invit">Marre du classique hello World</string>
</resources>
```

Une première Activité (++ 4/11)

Classe R des ressources

```
package df.cours22;

public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int upjv_launcher=0x7f020000;
    }
    public static final class id {
        public static final int bouton=0x7f050001;
        public static final int val=0x7f050000;
    }
    public static final class layout {
        public static final int main=0x7f030000;
    }
    public static final class string {
        public static final int app_name=0x7f040000;
        public static final int text_invit=0x7f040001;
...
}
```

L'IDE Android Studio génère automatique une classe static R de toutes les ressources : fichier app/df.cours22/test/R.java
Ce sont les références effectives sur les ressources
surtout ne pas modifier à la main ce fichier !

Les ressources sont nommées en JAVA :
[package.]R.type.nom exemple R.string.app_name
en XML :
@[package:]type/nom exemple @string/app_name

Une première Activité (5/11)

src/df.cours22/MarreDHelloWorldActivity.java

Le fichier JAVA de l'activité

```
package df.cours22;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;
public class MarreDHelloWorldActivity  extends Activity {
  private int val;
  private TextView text;
  @Override
  public void onCreate(Bundle bundle) {
    super.onCreate(bundle);
    setContentView(R.layout.activity_marre_dhello_world);
    val = 0;
    text = (TextView) findViewById(R.id.val);
    text.setText("valeur = "+val);
  }
  public void actionPlus1(View view) {
    text.setText("valeur = "+ ++val);
  }
}
```

➔1 activité =(simpliste) 1 page écran

la classe hérite d'Activity : elle débute par l'appel à sa méthode onCreate()

La méthode de création est redéfinie : le contenu graphique de l'application est obtenue à partir de res/layout/activity_marre_dhello_world.xml

La référence du widget text est obtenue à partir de son identifiant

La méthode `actionPlus1` incrémente val et l'affiche dans le widget text

Une première Activité (++ 6/11)

MarreDHelloWorldActivity

```
public class MarreDHelloWorldActivity  extends Activity {
  private int val;
  private TextView text;
  public void onCreate(Bundle bundle) {
    super.onCreate(bundle);
    setContentView(R.layout.activity_marre_dhello_world);
    val = 0;
    text = (TextView) findViewById(R.id.val);
    text.setText("valeur = "+val);
  }
  public void actionPlus1(View view) {
    text.setText("valeur = "+ ++val);
  }
}
```

➔la méthode onCreate est appelée à la création de l'objet :
elle doit obligatoirement comporter un appel à sa "super" sinon un exception est levée (contrainte assez fréquente du Framework!)
la méthode setContentView affecte le contenu graphique de l'activité en lui passant la référence de la vue `R.layout.activity_marre_dhello_world`

La méthode findViewById permet d'obtenir la référence d'un objet en fournissant son attribut XML id : R.id.val

Il est possible de créer "dynamiquement" le contenu visuel en instanciant les widgets comme dans SWING : indispensable aux cas d'interface dépendant des données en entrée

la méthode `actionPlus1` prévue pour traiter les événements click : elle reçoit la référence du widget cliqué en paramètre d'appel,
cela évite la programmation de l'écouteur anonyme

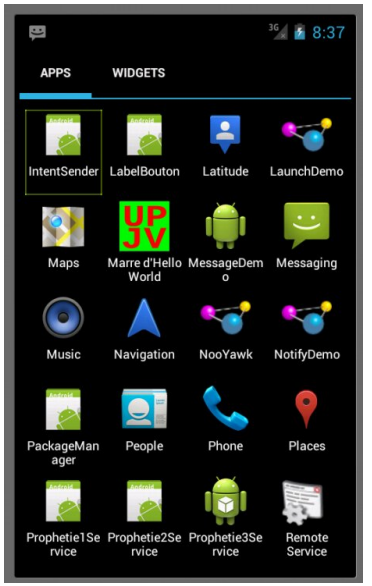
Une première Activité (7/11)

Le manifeste de l'application

le fichier AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="df.cours22"
    android:versionCode="1"
    android:versionName="1.0" >

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

    <application
        android:allowBackup="true"
        android:icon="@drawable/upjv_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="df.cours22.MarreDHelloWorldActivity"
            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>
```

➔spécifie l'application en précisant son nom et son icône
l'activité "main" à lancer : sa classe, son package
Dans le filtre d'intention : c'est la classe principale (MAIN) et elle apparaît dans le "Launcher"



Une première Activité (++ 8/11)

Le manifeste de l'application

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="df.cours22"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk android:minSdkVersion="8" android:minSdkVersion="17" />
    <application
        android:icon="@drawable/upjv_launcher"
        android:label="@string/app_name" >
        <activity
            android:name=".MarreDHelloWorldActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        ….
```

le manifeste contient les infos nécessaires au fonctionnement de l'application :
le package
les versions du SDK Android "uses-sdk" pour que l'application fonctionne
"versionCode" : le numéro de version de l'application
le nom et l'icône de l'application
L'application peut contenir plusieurs activités (et autres …) :
l'activité et le nom de sa classe
l'icône est une référence à la ressource res/drawable/upjv_launcher (qui n'existe pas mais ….)
l'intent-filter précise "à quoi sert l'activité".

Une première Activité (++ 9/11)

Ressources alternatives

➔l’icône est une référence à la ressource res/drawable/upjv_launcher qui n'existe pas mais res/drawable-hdpi/upjv_launcher.png est une image 72*72,
drawable-xdpi contient la version 96*96,
drawable-mdpi la 48*48
drawable-ldpi la 36*36

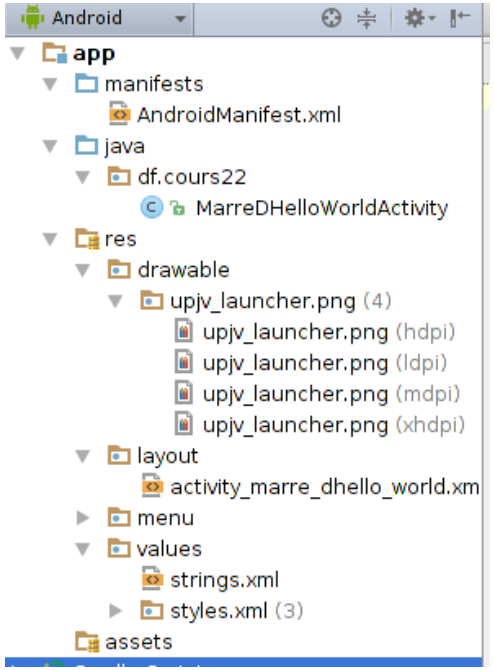
La ressource effectivement utilisée sera l'alternative la plus adaptée aux caractéristiques du terminal utilisé.

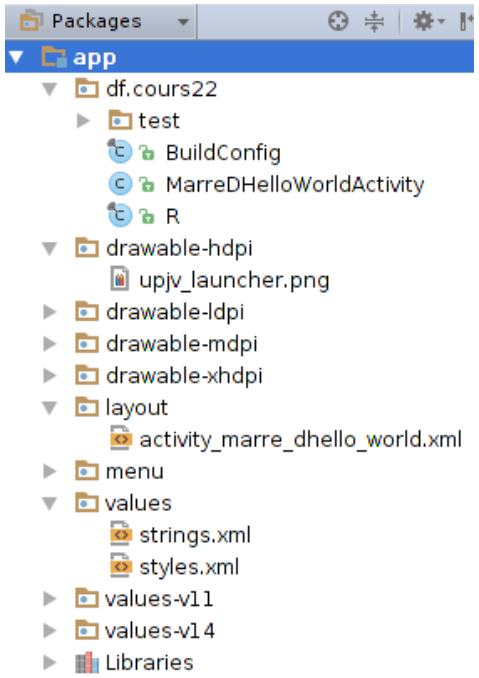
```
res/
├── values/
│   └── strings.xml
├── values-fr/
│   └── strings.xml
├── values-fr-rCA/
│   └── strings.xml
├── drawable-en/
├── drawable-en-port/
└── drawable-en-notouch-12key/
```

➔Mobile Country Code et Mobile Network Code
mcc208-mnc00 France Orange
Langage et région
fr-rFR la France aux français
Taille d'écran
normal medium-density HVGA >= 320x470 dp
Orientation d'écran
port portrait
Densité de pixel
mdpi Medium-density 160dpi
Type d'écran tactile
notouch, finger
Clavier
nokeys, qwerty, 12key
Type de navigation
nonav, dpad, trackball
….

Une première Activité (10/11)

L'arborescence de l'application





➔Dans Android Studio IDE, votre projet comporte :

en vue android :

un onglet manifest avec le fichier AndroidManifest.xml fichier de configuration de l'application

un onglet java avec vos packages et leurs classes

un onglet res contenant les ressources

- un onglet drawable des images
- un onglet layout avec le layout principal activity_marre_dhello_world.xml
- un onglet menu des layout menu
- un onglet values contenant le fichier string.xml
- un onglet assets des données brutes : exemple MP3

en vue Package :

un sous-onglet test (ne pas toucher !) des ressources générées, dont le fameux R.java

A la compilation :

Javac produit des .class à partir des .java

l'outil dx produit des .dex (Dalvik Executable) à partir des .class

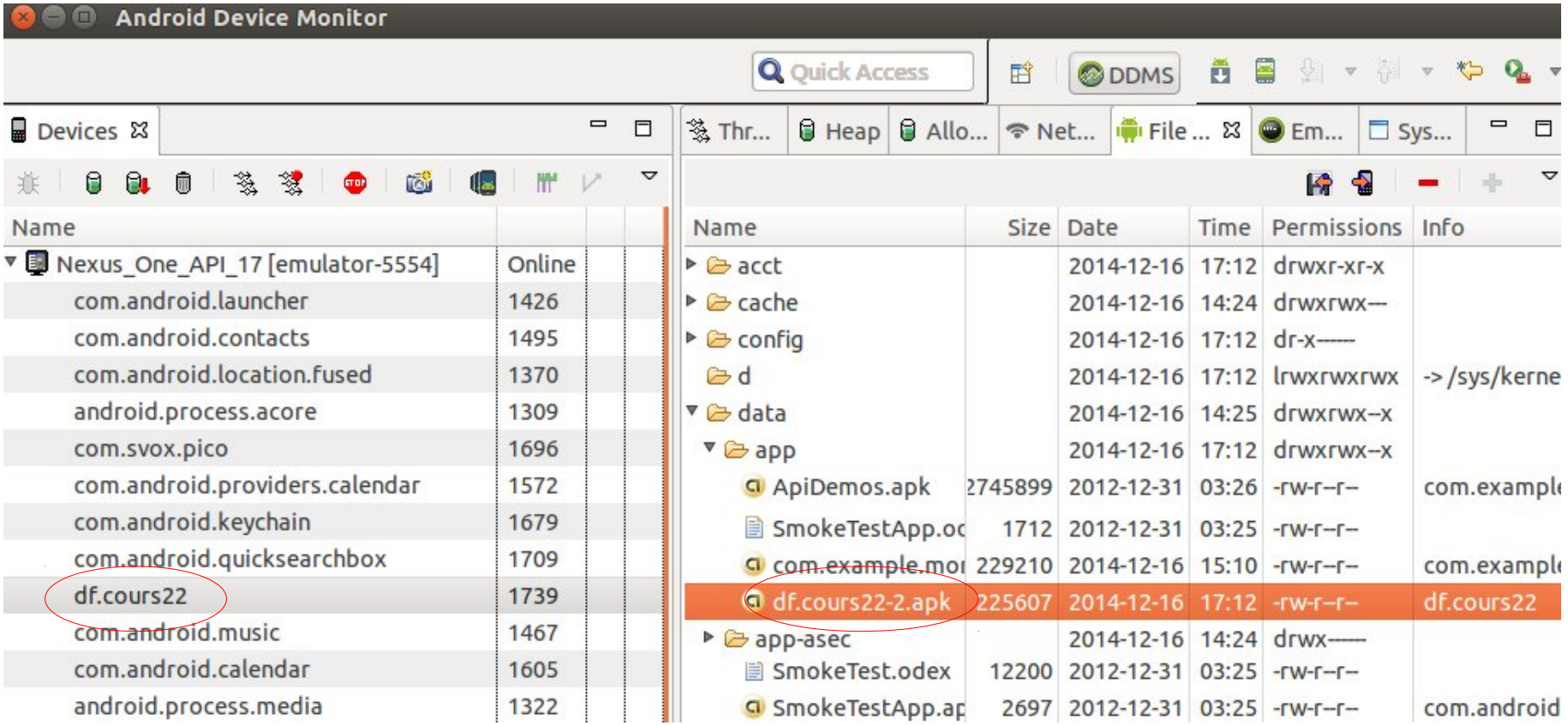
Puis création de l'android application package .apk qui contient les .dex, resources, assets et manifeste.

Ces fichiers ne sont pas directement accessibles !

Une première Activité (++ 11/11)

Le Device Monitor

→ tools → Android → android device monitor



➔DDMS (Dalvik Debug Monitor Service) possède plusieurs vues :
à droite : File Explorer
/data/app/df.cours22.apk 12267 -rw-r--r--
c'est une application zippé comme les Jar Java contenant un répertoire META-INF, un répertoire des ressources res, le fichier AndroidManifest.xml, l'exécutable classes.dex

à gauche : la vue Device affiche les processus en cours

Les outils de développement Android: (1/5)
Android Studio + SDK tools + émulateur AVD

http://developer.android.com/sdk/index.html
-> Download -> All Android Studio Packages

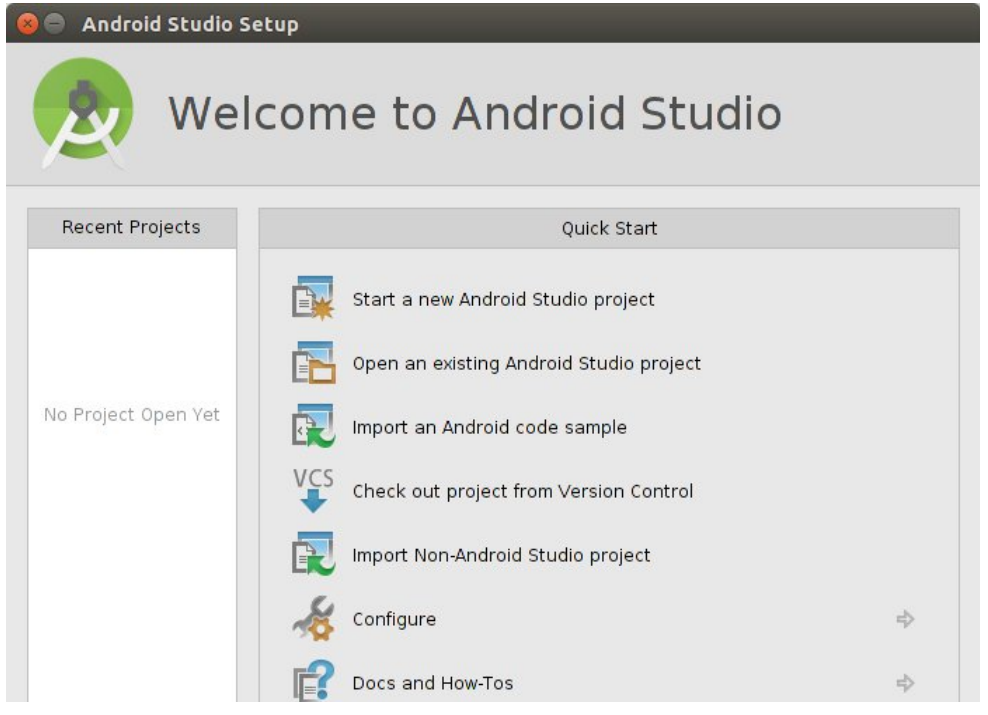
configuration :
Java Development Kit (JDK) 7 ou 6
+ prise en charge des applications 32 bits
soit en Linux :

```
apt-get install sun-java7-jdk
sudo dpkg --add-architecture i386
sudo apt-get update
sudo apt-get install libncurses5:i386 libstdc++6:i386 zlib1g:i386
```

dézipper et lancer l'IDE :

```
cd android-studio/bin
. studio.sh &
```

setup … et 3 Go supplémentaires à télécharger !



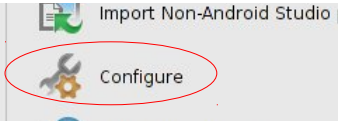
Pour tester vos applis sur votre mobile Android :
Enabling On-device Developer Options
Enable USB debugging

Connectez-le et repérez son identifiant :

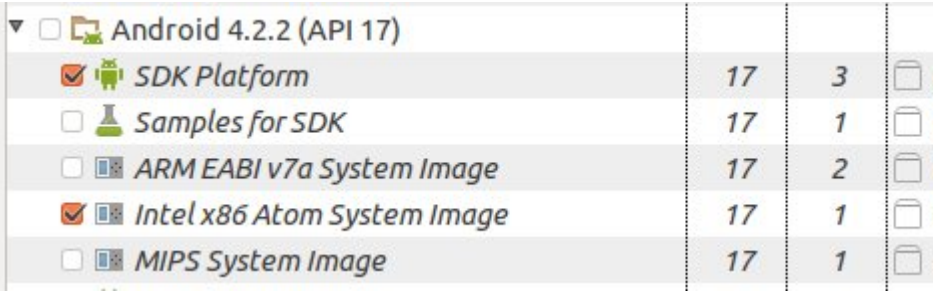
```
$ lsusb
Bus 007 Device 001: ID 1d6b:0001 Linux  1.1 root hub
.....
Bus 009 Device 001: ID 1d6b:0003 Linux  3.0 root hub
Bus 008 Device 005: ID 04e8:6860 Samsung Electronics
   Co.,  GT-I9300 Phone [Galaxy S III], ...
Bus 008 Device 001: ID 1d6b:0002 Linux 2.0 root hub
```

Ajouter l'identifiant au gestionnaire de devices

```
$ gedit /etc/udev/rules.d/51-android.rules
SUBSYSTEM=="usb", ATTR{idVendor}=="04e8", MODE="0666",
GROUP="plugdev"
$ chmod a+r /etc/udev/rules.d/51-android.rules
```



Pour tester sur un émulateur :
-> configure -> SDK manager
charger le SDK et l'émulateur de l'Api 17
(pas trop volumineuse !)



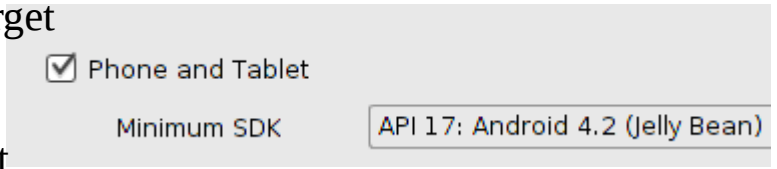
Les outils de développement Android (2/5)
premier projet

start a new android studio project

nom

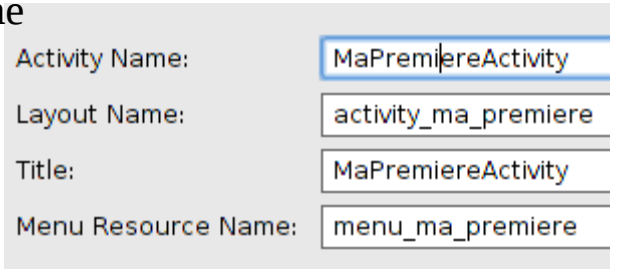
→ next

min target

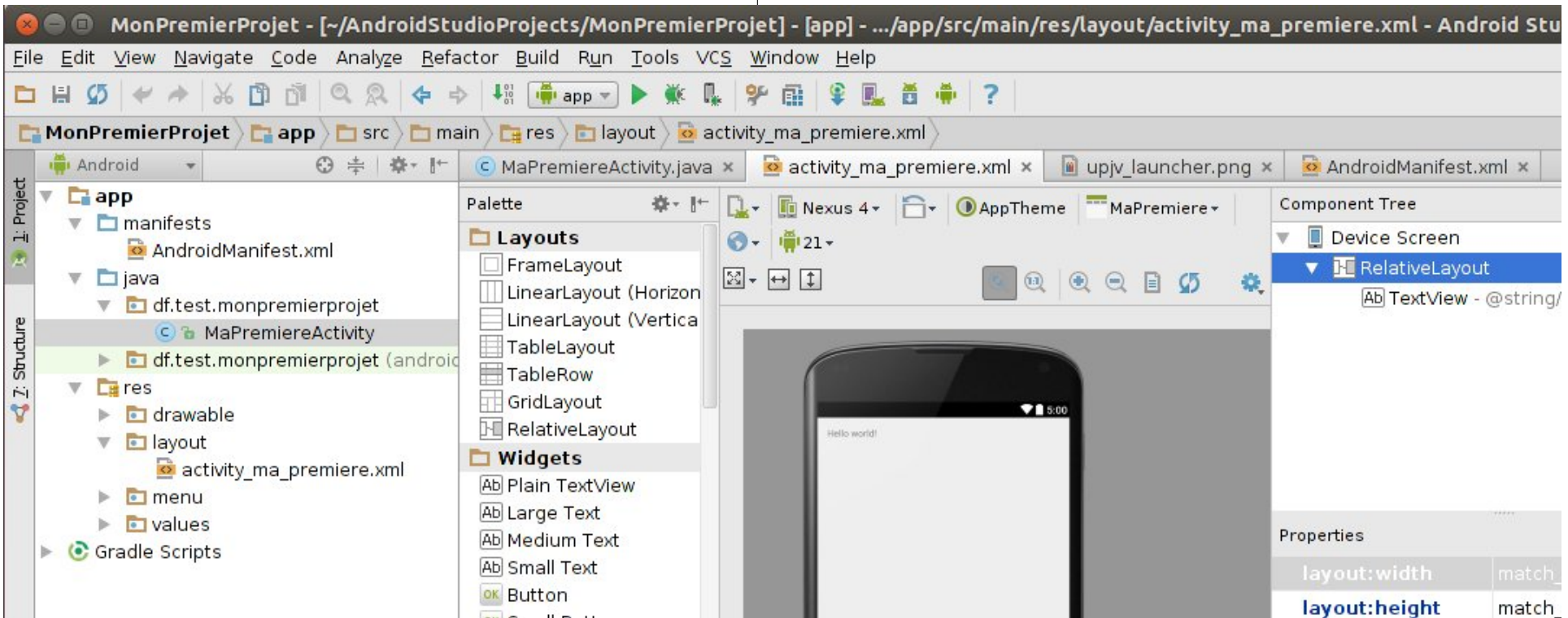


→ next

add an activity
-> blank activity
actity name



→ finish

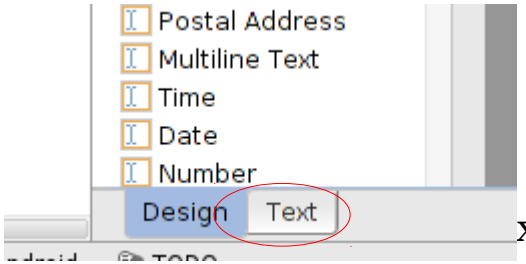


Les outils de développement Android (3/5)
premier projet

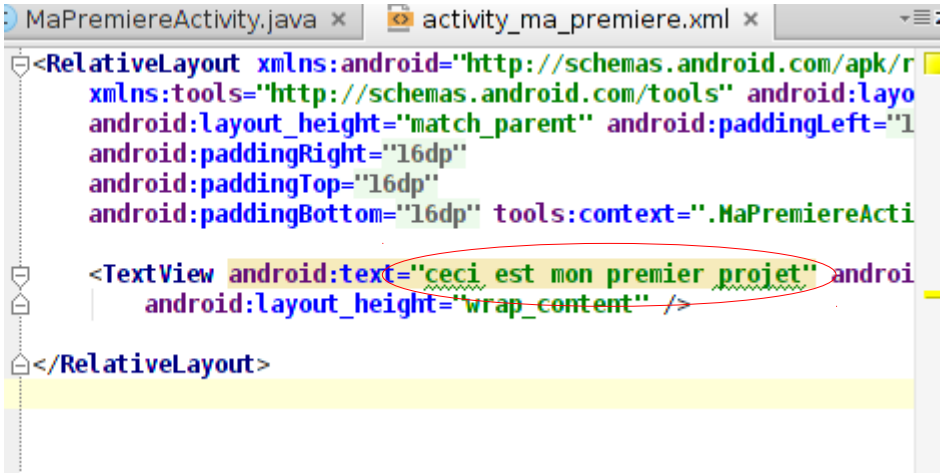
changer le texte

directement

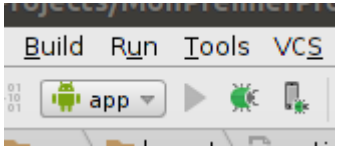
Sinon dans le fichier source

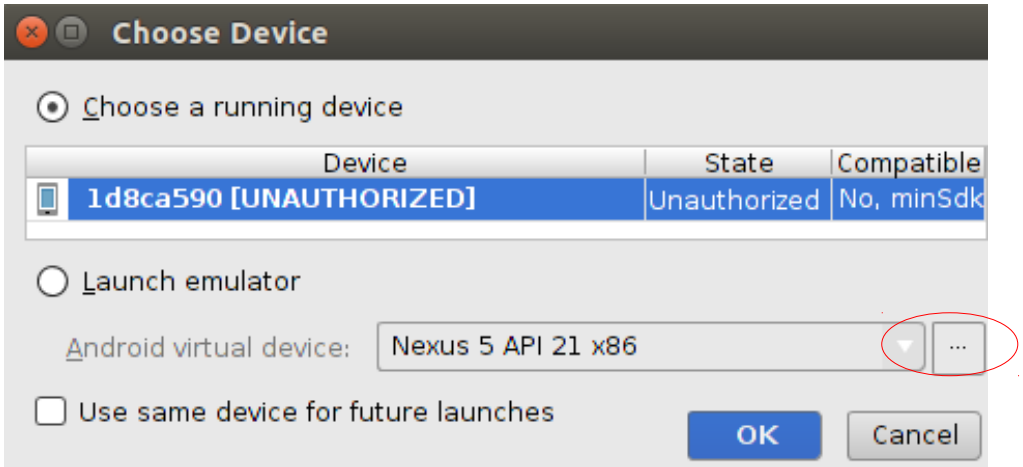


xml :



Run

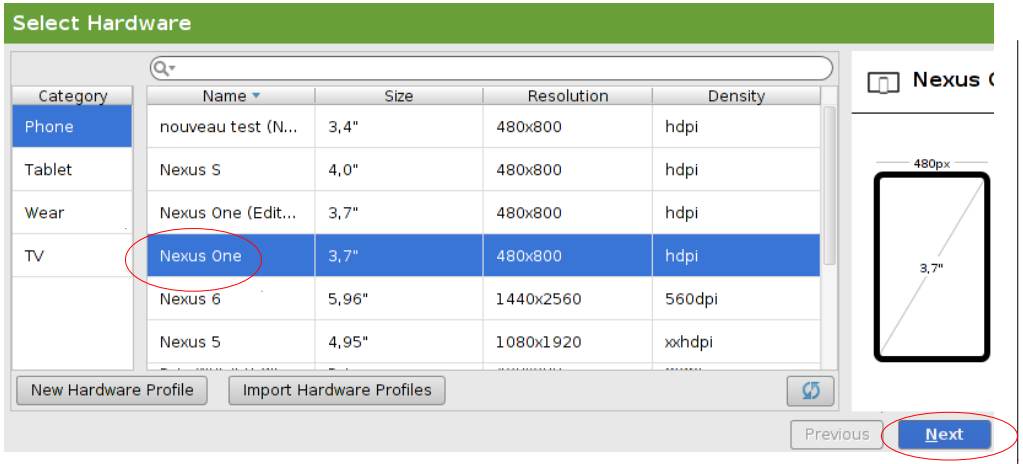


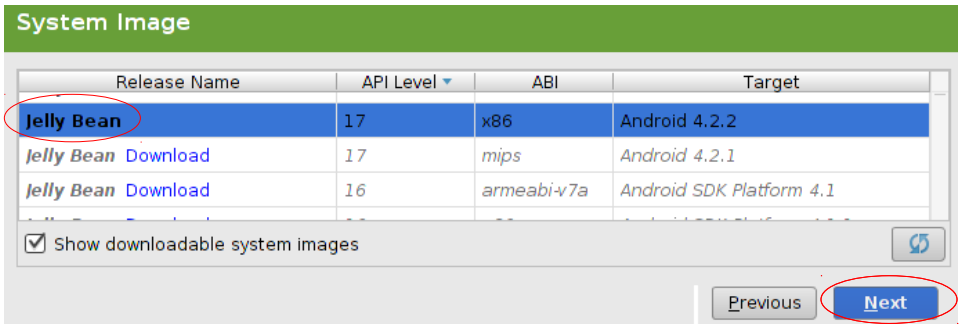


Soit vous choisissez votre mobile connecté
…. et ça devrait marcher … ? !

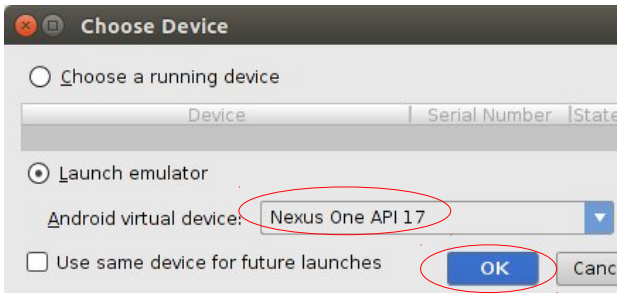
Soit un émulateur
→ …

Les outils de développement Android (4/5)
premier projet … run





lancer l'AVD que vous avez choisi



Après un temps fort long :

Les touches de contrôle de l'émulateur sont :

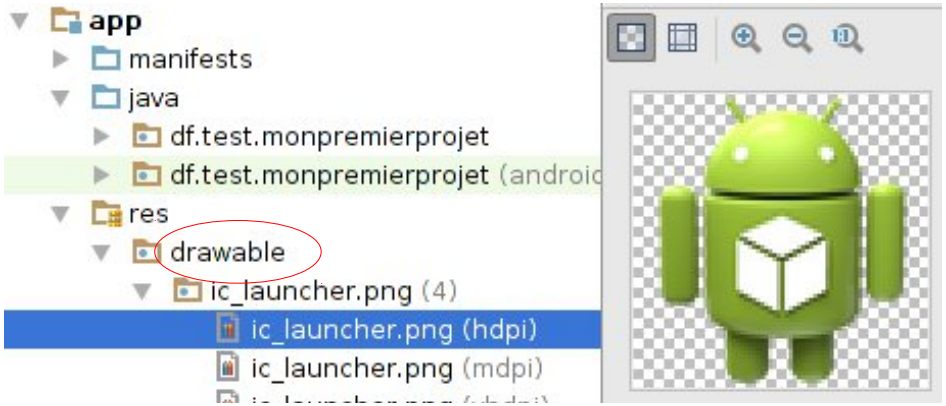
Home	home
Menu	F2 ou Page-up
Back	Esc
Search	F5
Star	shift-F2 ou Page-down
Bascule portrait-paysage	F11 et F12

…

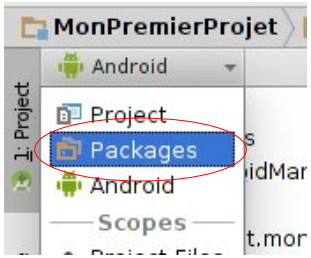
http://developer.android.com/tools/help/emulator.html

Les outils de développement Android (5/5)
premier projet … personnalisation

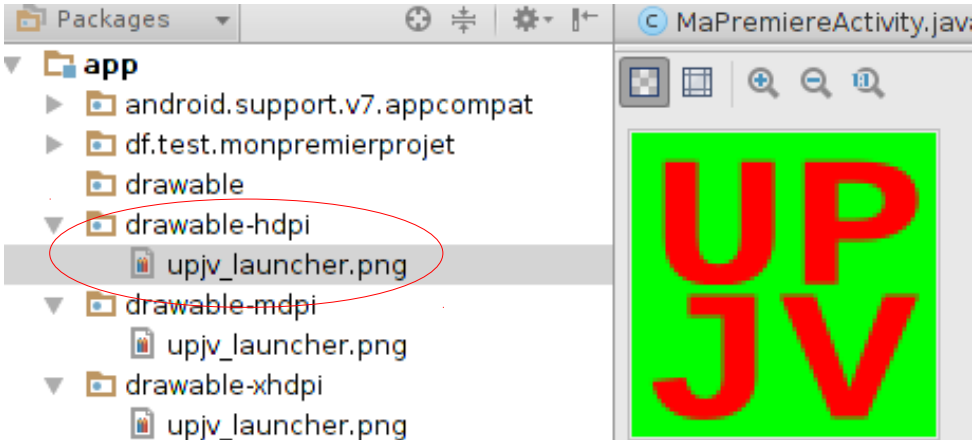
Dans le répertoire drawable se trouvent les vignettes



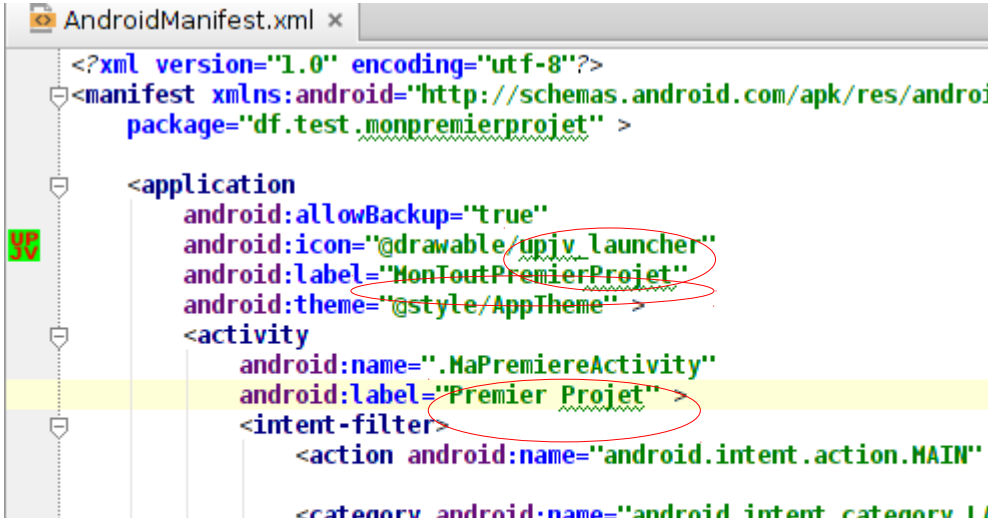
Passons en vue "Package"



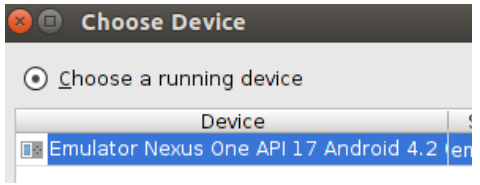
Et y mettre les vignettes "upjv" fournies sur le site du cours :

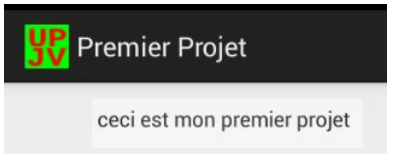


Et modifiez le manifest :

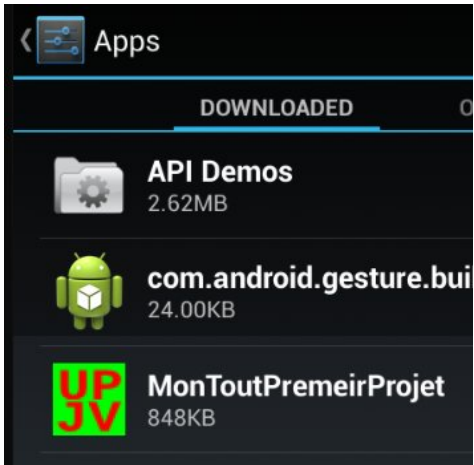


Re-exécutez sur le mobile ou l'émulateur qui fonctionne :





Puis Home → Settings → Apps

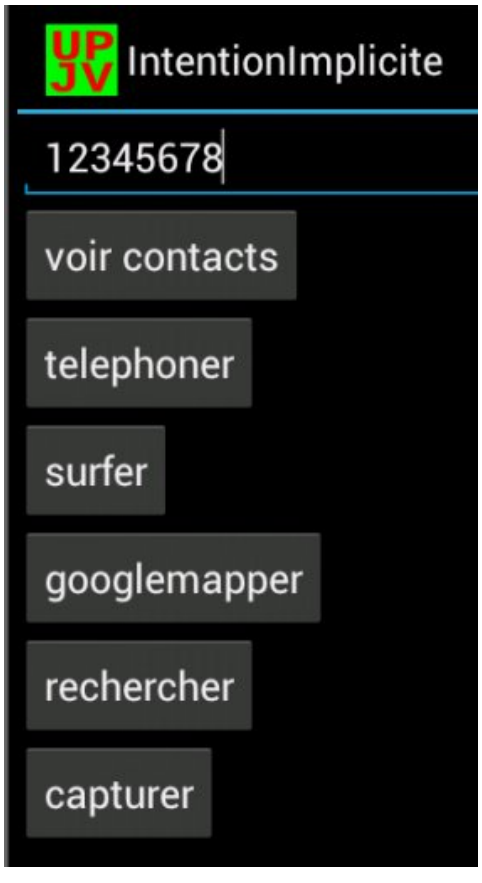


Activité appelant une activité : Intention (1/10)

➔Voici une activité qui en appelle d'autres : heureusement car sinon j'aurais été obligé de programmer un navigateur, un googlemap, la gestion du téléphone et de la caméra, ….. !

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

    <EditText
        android:id="@+id/parametre"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:inputType="text"
        android:text="parametres ..." />

    <Button
        android:id="@+id/voirContacts"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="onClickVoirContacts"
        android:text="voir contacts" />

    <Button
        android:id="@+id/telephoner"
        android:layout_width="wrap_content"
        ...
```

➔tapez www.u-picardie.fr pour surfer
"amiens" pour rechercher
ou googlemapper



Intention (2/10)

appel implicite d'activité

➔Appel d'activité sans les connaître :
sans connaître leur nom
ou un quelconque identifiant
simplement en indiquant ce que l'on souhaite comme action,
donc notre intention

```
package df.cours1;
...
public class IntentionImpliciteActivity
                          extends Activity {
...
  public void onClickVoirContacts(View view) {
    try {
      Intent intention = new Intent(Intent.ACTION_VIEW,
                                    Uri.parse("content://contacts/people/"));
      startActivity(intention);
    } catch (ActivityNotFoundException anfe) {
      Log.e("IntentionImpliciteActivity", "Voir les Contacts", anfe);
    }
  }
```

➔L'intention implicite indique l'action souhaitée voire une URI (Uniform Ressource Identifier) ou des données;
A charge pour le système de trouver la ou les activités qui peuvent convenir

la méthode startActivity() lance une sous-activité sans récupérer de résultat : l'appelante est mis en pause dans la pile d'activité

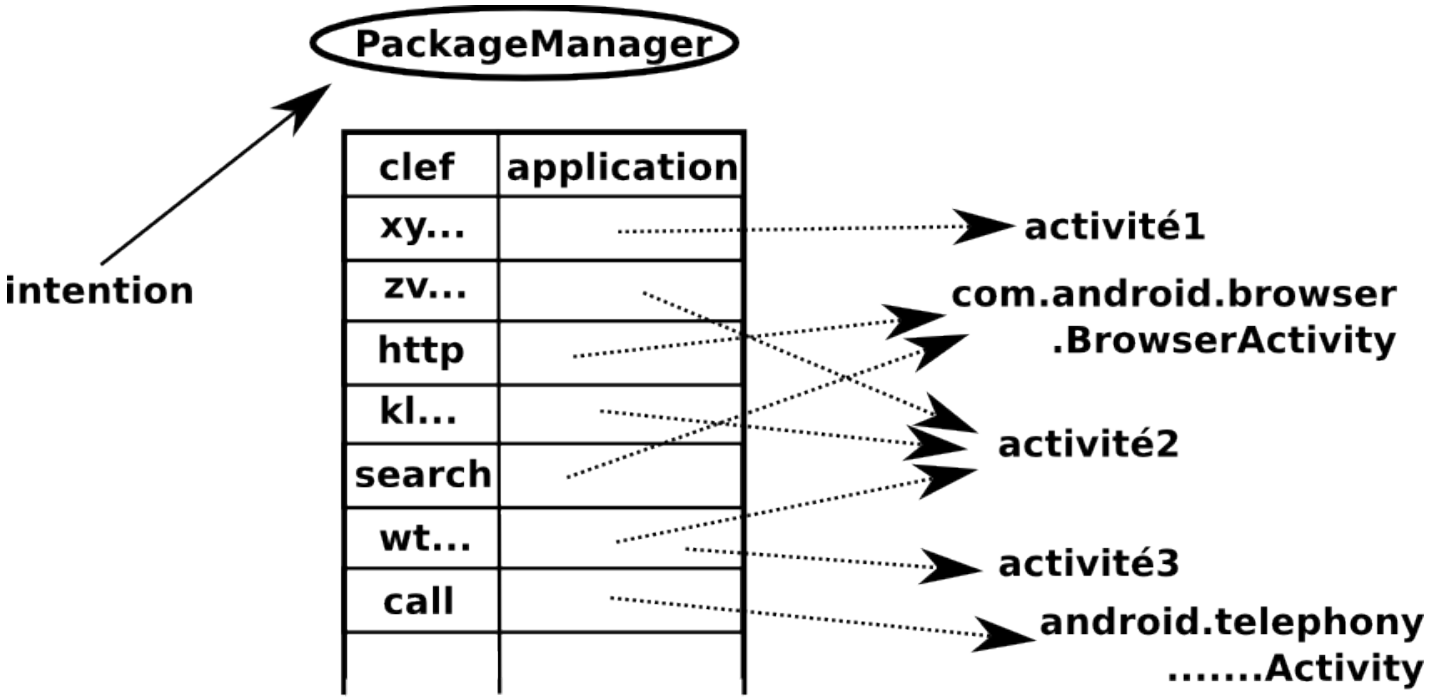
Intention (3/10)

appel implicite d'activité

```
...
onClickTelephoner :
      Intent intention = new Intent();
      intention.setAction(android.content.Intent.ACTION_DIAL);
      intention.setData(Uri.parse("tel:"+ parametre.getText()));
      startActivity(intention);

onClickSurfer :
      Intent intention = new Intent(Intent.ACTION_VIEW,
                        Uri.parse("http://"+parametre.getText().toString()));

onClickGoogleMapper :
      Intent intention = new Intent(Intent.ACTION_VIEW,
                        Uri.parse("geo:0,0?q="+parametre.getText().toString()));

onClickRechercher :
      Intent intention = new Intent(Intent.ACTION_WEB_SEARCH);
      intention.putExtra(SearchManager.QUERY, parametre.getText().toString());

onClickCapturer :
      Intent intention = new Intent("android.media.action.IMAGE_CAPTURE");
```

➔ Ici, la valeur tapée par l'utilisateur et passée dans paramètre est ajoutée dans l'intention

Intention (4/10) : résolution de l'intention



➔Coté appelant :
l'intention contient des données et informations pour l'appel.
Cas particulier de l'intention explicite: l'intention contient le nom de la classe appelée.

Coté appelée :
chaque application indique dans son manifeste le type d'intention qui peut l'appeler.
C'est un filtre d'intention … slide à venir

Le package manager gère un registre des types d'intention et des applications associées :
Une intention peut ne pas avoir d'application traitante.
Une intention peut avoir plusieurs applications traitantes et peut-être une par défaut.
Une application peut traiter différents types d'intentions.

Intention (5/10)

ActiviteAppelante
ActiviteAppelee

Appel explicite d'une activité

➔1 activité =simpliste= 1 page écran

Conséquence :
1 application = plusieurs activités
les activités doivent pouvoir s'appeler en fournissant des paramètres données et récupérant des résultats

Appel direct (explicite) d'une activité à une autre en spécifiant sa classe

Intention (6/10)

ActiviteAppelante
ActiviteAppelee

appel startActivityForResult

```
package df.cours0;
...
public class ActiviteAppelante extends Activity {
  private static final int REQUEST_CODE = 13;
  private EditText param1, param2  ;
...
  public void onClick(View view) {
    int entier1, entier2;
    try {
      entier1 = Integer.parseInt(param1.getText().toString());
      entier2 = Integer.parseInt(param2.getText().toString());
    } catch (Exception e) {
      return;
    }
    Intent intention = new Intent(this, ActiviteAppelee.class);
    intention.putExtra("param1", entier1);
    intention.putExtra("param2", entier2);
    startActivityForResult(intention, REQUEST_CODE);
  }
...
```

➔ L'activité demande au système de lancer une autre activité correspondant à l'intention donnée : l'intention (Intent) est ici composée des références explicites à une classe et 2 données entières

startActivityForResult() correspond à un appel de fonction/procédure : l'appelante sera stoppée jusqu'à ce que l'appelée termine sa tache et lui fournisse un résultat

le code de requête est un numéro qui sera retourné (si >=0) : cela permet de vérifier que le retour correspond bien à l'appel

Intention (7/10)

Manifeste d'activité appelante et appelée

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="df.cours0"
...
    <application
        android:allowBackup="true"
        android:icon="@drawable/upjv_launcher"
        android:label="@string/app_name" >
        <activity
            android:name=".ActiviteAppelante"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".ActiviteAppelee"
            android:label="ActiviteAppelee" >
        </activity>
    </application>
</manifest>
```

→L'activité appelée est dans le même package
mais les activités appelantes et appelées peuvent être dans des packages différents
Sa classe est indiquée comme activité

Intention (8/10)

Activité appelée

```
package df.cours0;
...
public class ActiviteAppelee extends Activity {
  private int entier1, entier2;
  public void onCreate(Bundle bundle) {
    super.onCreate(bundle);
    setContentView(R.layout.second);
    TextView params = (TextView)findViewById(R.id.params);
    Bundle extras = getIntent().getExtras();
    if (extras == null)
      return;
    entier1 = extras.getInt("param1", 0);
    entier2 = extras.getInt("param2", 0);
    params.setText("param1="+entier1+" param2="+entier2);
  }
  public void onClick(View view) {
    Intent intentionResult = new Intent();
    intentionResult.putExtra("somme", entier1 + entier2);
    setResult(RESULT_OK, intentionResult);
    this.finish();
  }
}
```

➔La méthode getIntent() permet de récupérer l'intention d'appel :
en particulier, les données de type Bundle obtenues par getExtras()

L'activité se termine par un appel à sa méthode finish()

Au préalable, le résultat est préparé son forme d'intention ayant une donnée
et un code de réponse RESULT_OK (sinon RESULT_CANCELED)

Intention (9/10)

ActiviteAppelante
ActiviteAppelee

Retour d'appel d'activité

```
...
public class ActiviteAppelante extends Activity {
....
  protected void onActivityResult(int requestCode, int resultCode, Intent
data) {
    if (resultCode == RESULT_OK && requestCode == REQUEST_CODE
        && data.hasExtra("somme")) {
      int recu = data.getExtras().getInt("somme",0);
      param1.setText("somme = ");
      param2.setText("        "+recu);
    }
  }
}
...
```

➔L'appelante est réactivée et sa méthode de "callback" onActivityResult() reçoit le résultat en paramètres :
le code de requête fournit à l'appel
le code réponse RESULT_OK ou RESULT_CANCELED
et une intention contenant les résultats

Donc une solution simple pour une application de plusieurs pages (vues) :
associer une activité par vue (page)
appeler les activités par intention explicite pour "enchaîner" les pages

Intention (10/10) : Classe Intent

➔ "intention" = message asynchrone de requête entre composants Android :
- une première activité Activity envoie un Intent au système pour en démarrer une seconde
- permet d'associer plusieurs composants afin d'effectuer une tache
- ou de signaler des événements
- peut contenir des données
- Intention explicite :

```
Intent intentionSomme = new Intent(this,mon.package.ActivitySommmeDe2Entiers.class);
```

- Intention implicite : si on peut définir la tache à effectuer voire une uri : exemple visualiser une page web

```
Intent intentionWeb = new Intent(Intent.ACTION_VIEW,
                                 Uri.parse("http://www.site.com/page.html"));
```

Le système cherchera un composant enregistré pour la tache "visualiser", Intent.ACTION_VIEW et le type associé, url : si plusieurs sont possibles, une fenêtre de dialogue demandera à l'utilisateur son choix.
exemple d'action :

ACTION_EDIT éditer la donnée dont l'uri est fournie
ACTION_CALL appeler un numéro de téléphone
ACTION_VIEW visualiser la donnée dont l'uri est fournie
ACTION_BATTERY_LOW signaler que le niveau de batterie est bas aux broadcasts receiver
df.cours.CALCULS mon action à moi

uri et type :

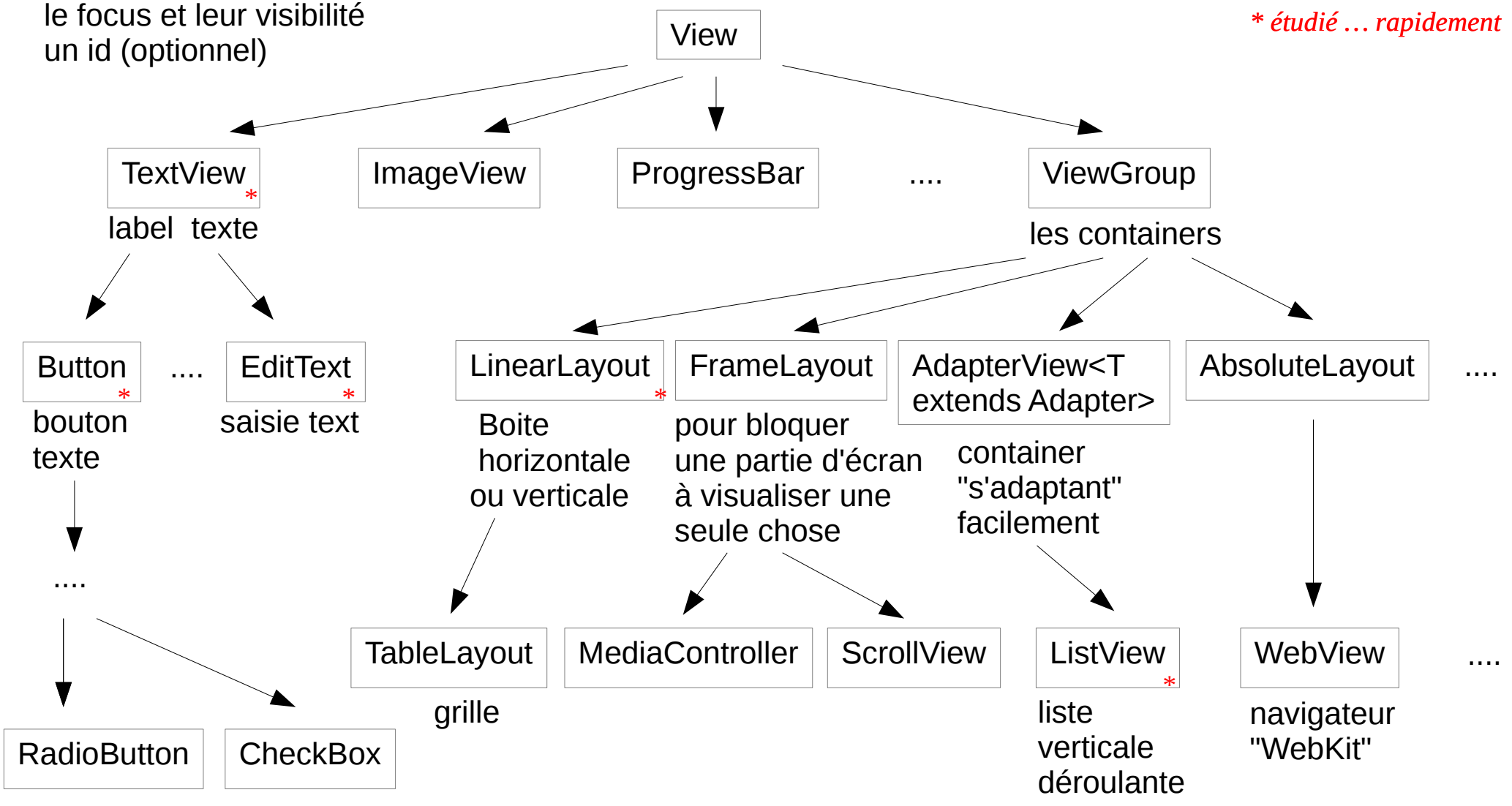
http://www.site.com/page.html
tel:678954322 numero de tel
geo:49.895243,2.29846

Des données supplémentaires optionnelles peuvent être fournie en donnée ou en résultat. Elles sont stockées en "array associatif" : paire clé-valeur avec String clé
pour en ajouter intentionSomme.putExtra("param1", 12);
pour les extraire : Bundle extras = getIntent().getExtras();
Ces données peuvent être des valeurs int, double, String voire plus complexes sérialisables (Parcelable)

Les Composants graphiques (1/9)

class View
la brique de base de l'interface utilisateur
= zone rectangulaire "dessinable" et source d'événement
dans une fenêtre d'application, les views sont organisées en arborescence. Elles sont définies par programmation et/ou par fichier XML :
leurs propriétés (x, y, text, width, backgroundColor, ...)
leurs éventuels listeners d'événements
le focus et leur visibilité
un id (optionnel)

** étudié … rapidement*



ListView : Quelques composants graphiques (2/9)

Elément graphique en version XML et en version JAVA :

```
<TextView
     android:id="@+id/text1"
     android:layout_width="fill_parent"
     android:layout_height="wrap_content"
     android:text="@string/text_invit" />
```

android.widget.TextView affiche un texte :
CharSequence getText() retourne le texte
void setText(CharSequence text) le change

```
<EditText
     android:id="@+id/text2"
     ...
     android:textSize="10"
     android:inputType="text"
     android:text="affiché" />
```

android.widget.EditText hérite de TextView pour n'offrir qu'un texte éditable

```
<Button
     android:id="@+id/bouton"
     ...
     android:onClick="ajout"
     android:text="ajouter" />
```

android.widget.Button hérite de TextView pour n'offrir qu'un texte « cliquable »
l'attribut XML onClick ajoute l'implémentation de l'interface android.view.View.OnClickListener ainsi :

```
button.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) { button.ajouter(v) ; }}) ;
```

```
<LinearLayout
     android:orientation="vertical"
     android:layout_width="fill_parent"
     android:layout_height="fill_parent" >
```

android.widget.LinearLayout est un layout container d'une colonne ou horizontal

ListView : Composant graphique (3/9)

ListView : un des widgets les plus importants et les plus utilisés
De base, une liste de texte au choix.

Le widget sélecteur est "rempli" à partir de données (tableau, résultat de requêtes SQLite, …) via un technique d'Adapter

Ci-dessous le layout principal :

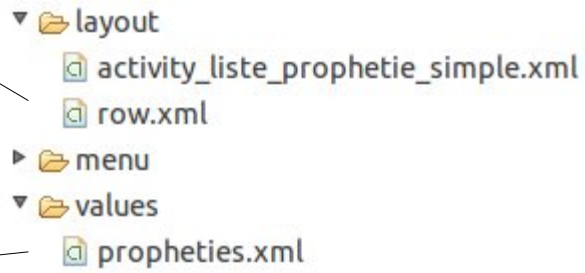
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >
    <TextView
        android:id="@+id/selection"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"/>
    <ListView
        android:id="@android:id/list"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:drawSelectorOnTop="false" />
</LinearLayout>
```

La référence list de la ListView est dans le package android : cela permet d'utiliser une ListActivity plus spécifique que Activity.

ListView : Composant graphique (4/9)


```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal" >
    <TextView
        android:id="@+id/texte_prophetique"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
</LinearLayout>
```

Le Row layout définit comment chaque item est visualisé au sein de la ListView




```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="baratin">
        <item >Peace and love !</item>
        <item >Travailler plus pour gagner moins</item>
        <item > Les temps sont durs ! Vive le MOU !</item>
...
    </string-array>
</resources>
```

Ressource tableau de prophéties

L'ArrayAdapter<String> va «associer » l'affichage global ListView, l'affichage d'un item row et la liste String des propheties pour offrir l'interface graphique ad-hoc.

ListView : Composant graphique (5/9)

```
public class ListeProphetieSimpleActivity extends ListActivity {
    private TextView selection;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_liste_prophetie_simple);
        selection = (TextView) findViewById(R.id.selection);
        Resources res = getResources();
        String[] propheties = res.getStringArray(R.array.baratin);
```

La classe Resource permet de récupérer les données du répertoire res

```
        setListAdapter(new ArrayAdapter<String>(this,
                            R.layout.row, R.id.texte_prophetique, propheties));
    }
```

La méthode setListAdapter() fournit un adapter à la ListView d'identifiant list
L'ArrayAdapter est un des plus simple : il faut lui fournir le contexte, le row layout, l'id recevant le string, un tableau de string

```
    protected void onListItemClick(ListView l, View v, int position, long id) {
         selection.setText(""+(position+1)+"-iéme prophétie !");
      }
}
```

La méthode `onListItemClick`() est appelée à chaque sélection :
le 1er paramètre est la ListView, le second est la view "clickée", puis la position "clickée", enfin l'id de la vue "clickée".

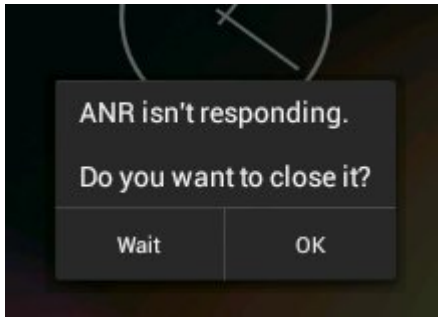
Programmation graphique (6/9)

ANR : Application Not Responding

```
<Button
    android:id="@+id/button1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Bloquer plus de 5 secondes"
    android:onClick="cliquage"/>
```

```
public class ANRActivity extends Activity {
 ...
    public void cliquage(View view)  {
        while (true)
          { }
    }
...
```

➔Appuyer sur le bouton
puis tenter de faire autre chose

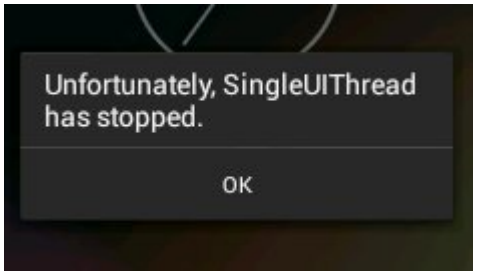


Le thread main est l'UI Thread.
Le système déclenche un ANR (application not responding) quand l'application ne répond plus aux actions de l'utilisateur .... environ 5 secondes. L'UI Thread de l'application en foreground ne doit pas geler l'interface utilisateur plus de 5 secondes.
En conséquence, les opérations longues sont à effectuer sur un thread autre que le thread main (ou UI Thread).
Solution à venir … dans la 2nde partie du cours

Programmation graphique (7/9) : Le « Single UI thread model »

➔ 2nde règle :
Toutes les actions sur l 'UI (interface utilisateur) se font sur le seul UI thread.

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_single_uithread);
    Thread tache = new Thread(new Runnable() {
        public void run() {
            try {
                Thread.sleep(2000);
            } catch (InterruptedException ie) {}
            TextView text = (TextView) findViewById(R.id.text);
            text.setText("accès !");
        }
    });
    tache.start();
}
...
```

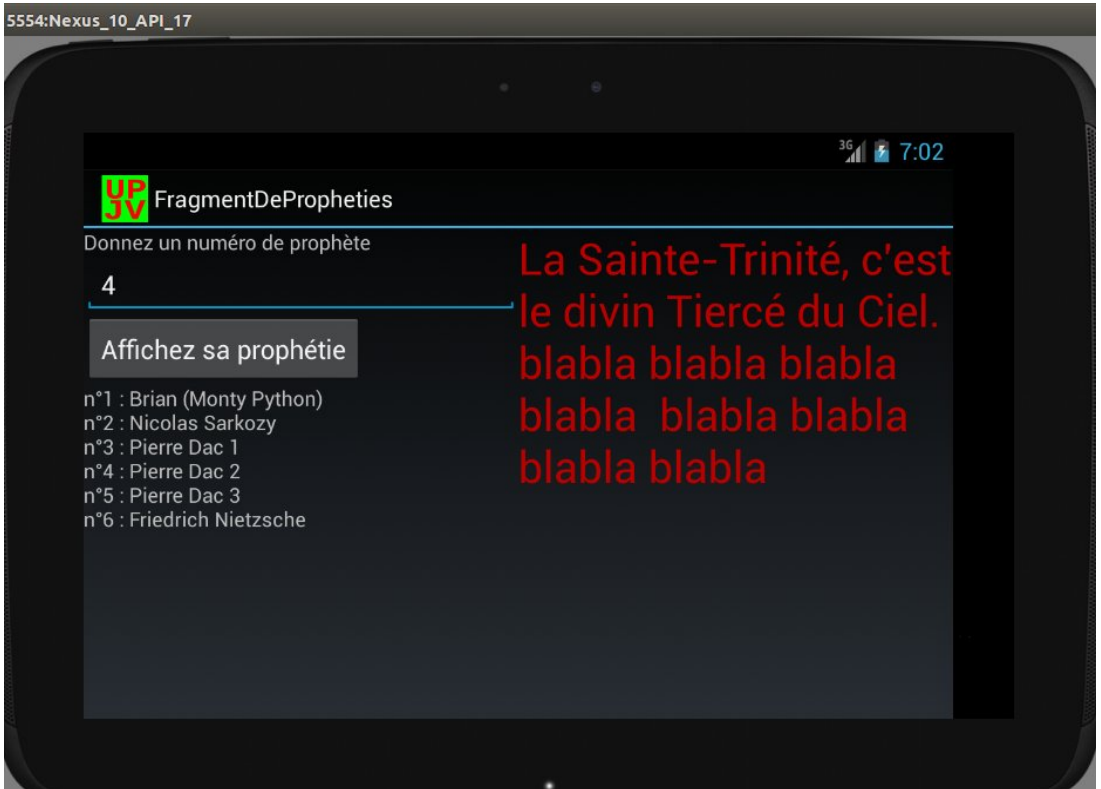



```
AndroidRunti FATAL EXCEPTION: Thread-129
AndroidRunti android.view.ViewRootImpl$CalledFromWrongThreadException: Only t ⏎
             he original thread that created a view hierarchy can touch its v ⏎
             iews.
```

➔Pour éviter ces problèmes, il existe des méthodes qui renvoient des actions sur l'UI-thread :
Activity.runOnUiThread(Runnable)
View.post(Runnable)
View.postDelayed(Runnable, long)
Solution à venir … dans la 2nde partie du cours

Fragment (8/9)
une seule application pour téléphone et tablette

Trop "tordu" pour être étudié içi !



Fragment (9/9)

Contraintes :
les vues sont les fragments
le contrôleur de choix est le fragment de la liste de choix
le contrôleur d'affichage est l'activité pilote : elle "sent" si l'écran est en lanscape ou non

Principe :
Donc le choix opéré dans le fragment de la liste de choix sera
transmit à l'activité de la liste de choix, activité pilote
l'activité pilote, selon le mode d'affichage :
soit transmettra au fragment de la vue choisie
soit transmettra/démarrera l'activité de la vue choisie
qui transmettra au fragment de la vue choisie

Cycle de vie de l'activité (1/10)


```
package df.cours22;
...
public class CycleDeVieActivity extends Activity {
  public void onCreate(Bundle savedInstanceState) {
    Log.v("CycleDeVieActivity", "onCreate");
    super.onCreate(savedInstanceState);
    ...
  }
  protected void onDestroy() {
    Log.v("CycleDeVieActivity", "onDestroy");
    super.onDestroy();
  }
  protected void onPause() {
    super.onPause();
    Log.v("CycleDeVieActivity", "onPause");
  }
  protected void onRestart() {
    Log.v("CycleDeVieActivity", "onRestart");
    super.onRestart();
  }
  protected void onResume() {
    Log.v("CycleDeVieActivity", "onResume");
    super.onResume();
  }
  protected void onStart() {
    Log.v("CycleDeVieActivity", "onStart");
    super.onStart();
  }
  protected void onStop() {
    Log.v("CycleDeVieActivity", "onStop");
    super.onStop();
  }
```

Les méthodes "callback" d'Activity

➔ Une activité a 4 états possible :

resumed : cad en exécution, en foreground : elle est visible à l'écran et interagit avec l'utilisateur

paused : partiellement visible, n'interagit pas, ne s'exécute pas

stopped : en background, n'est plus visible, mais ses données sont figées et maintenues en mémoire.

Hors mémoire : soit pas encore chargée pour exécution, soit détruite après exécution

Les méthodes de ses transitions d'états de vie peuvent être redéfinies mais elles doivent obligatoirement appeler leur "super" sinon une exception est levée

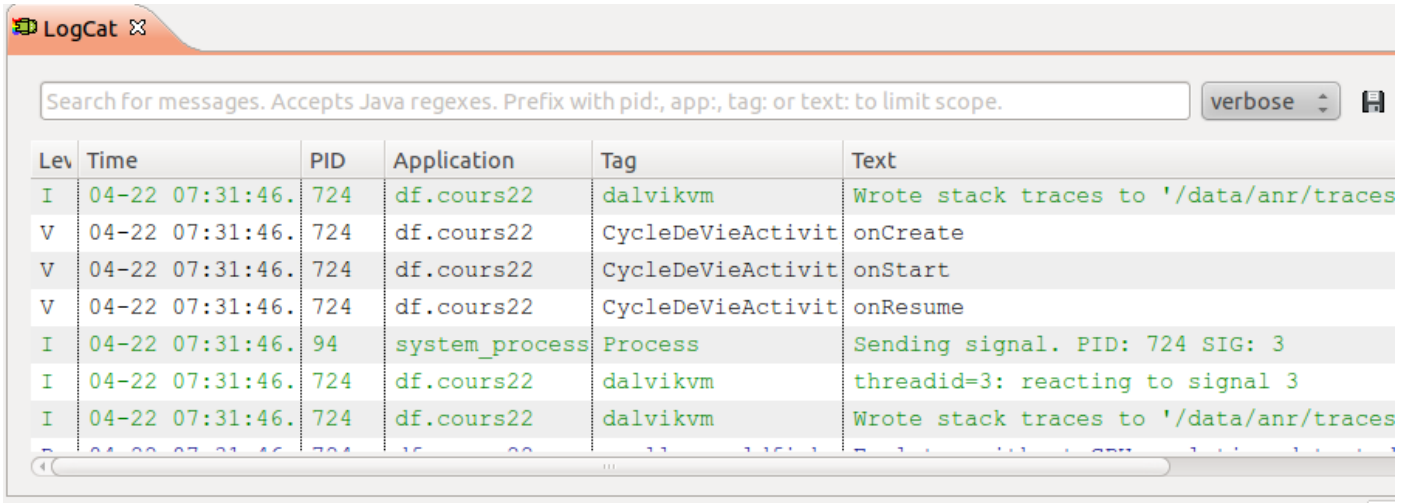
Cycle de vie de l'activité (2/10)

```
...
Log.v("CycleDeVieActivity", "onDestroy");
...
```

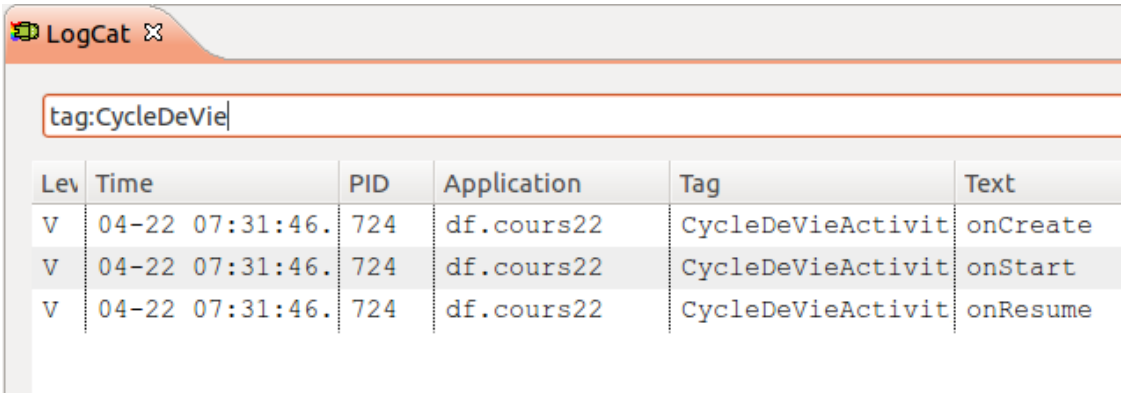
Les logs

➔Log la classe pour les logs :
méthodes : Log.v() Log.d() Log.i() Log.w() and Log.e()
pour respectivement ERROR, WARN, INFO en production et DEBUG, VERBOSE pour le développement

-> window -> show view -> other -> android -> logcat

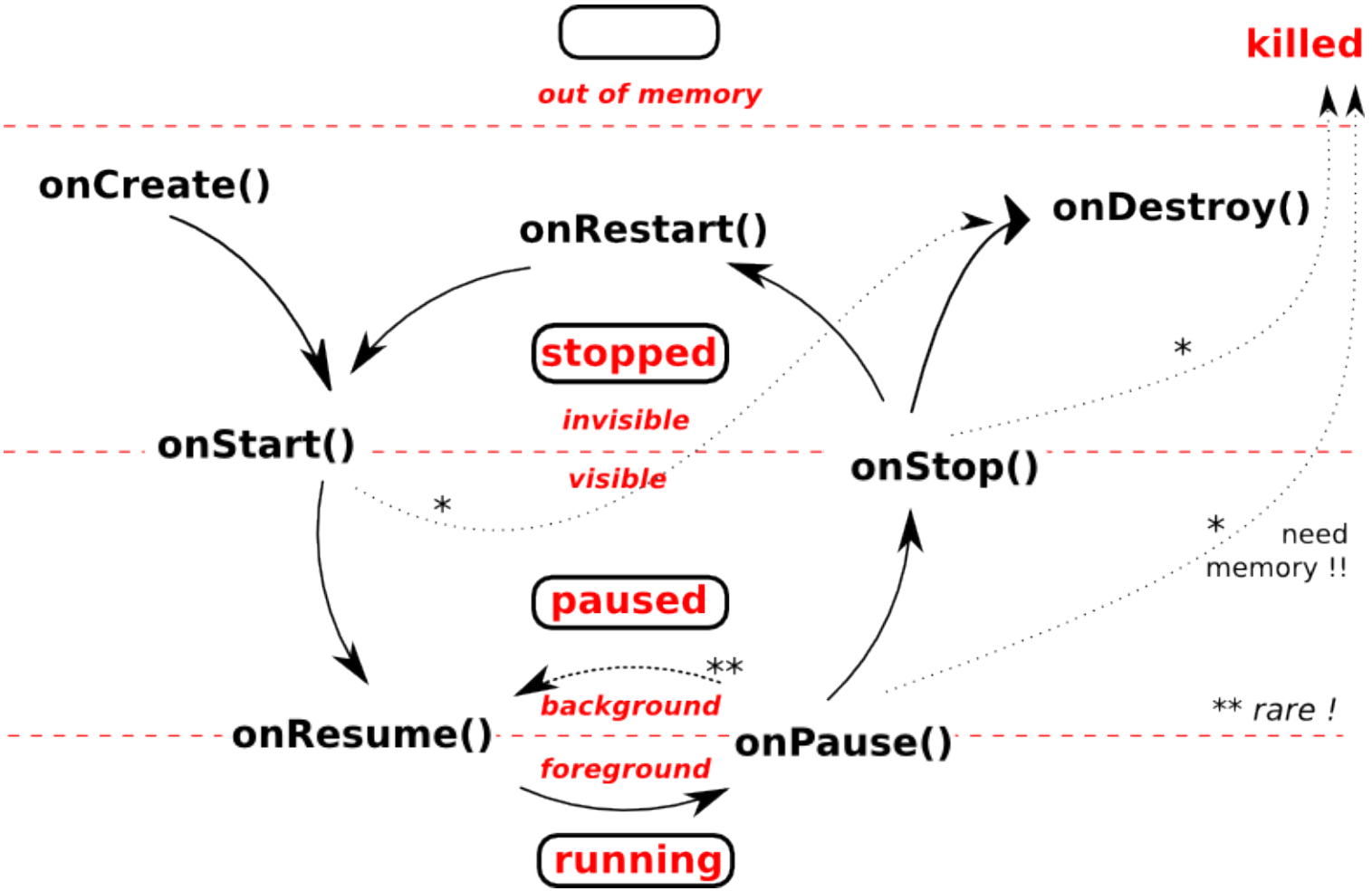


➔ filtrage sur tag:CycleDeVieActivity



Cycle de vie de l'activité (3/10)

Le Cycle de vie de l'activité et les fonctions « callbacks » appelées



Cycle de vie de l'activité (4/10)

Cycle et Pile d'activités

➔L'ActivityManager gère la pile d'activités :
la touche back "remonte" dans la pile
la touche home vide la pile
L'appel d'une activité par une autre l'empile et son retour la dépile.

➔démarrage :
- onCreate
- onStart
- onResume

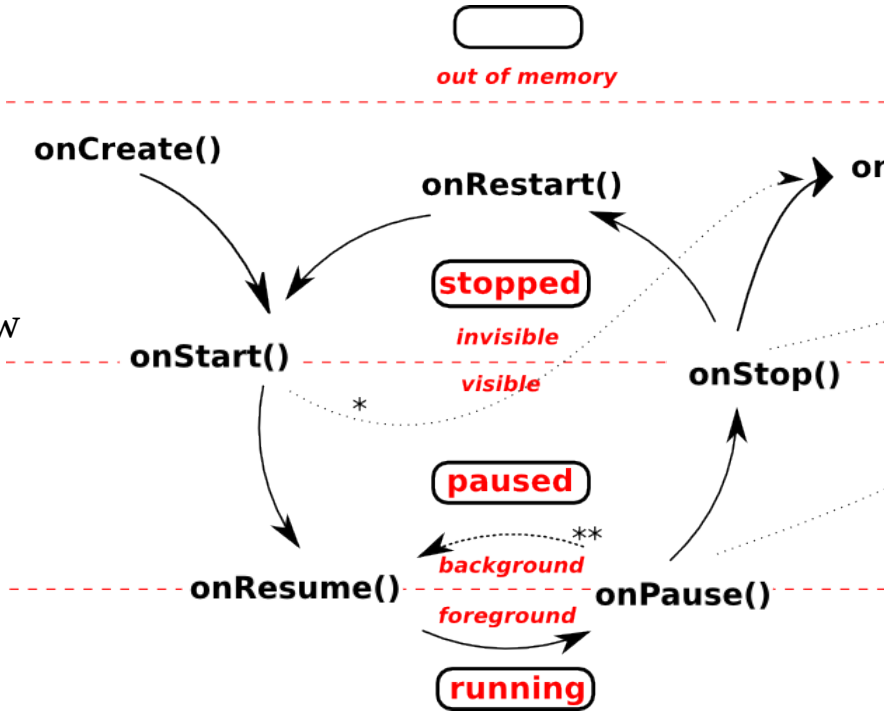
L'activité est en état running : visible et foreground. L'utilisateur interagit avec elle. L'activité passe au sommet de la pile d'activités.

appel de l'autre activité :
- onPause
- onStop

L'activité appelante est stoppée. Elle ne s'exécute pas et le "window manager" ne gère plus les ressources de l'activité.

retour de l'autre activité : par exemple, touche back
- onRestart
- onStart
- onResume

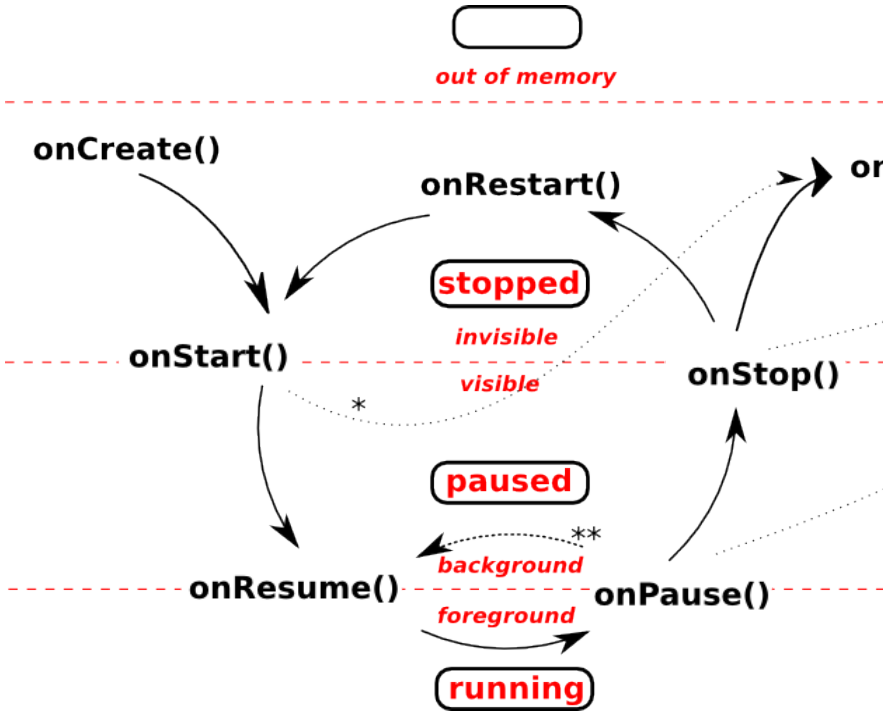
Elle est de nouveau au sommet de pile et redémarre par la méthode onRestart().



Cycle de vie de l'activité (5/10)

Cycle et Pile d'activités

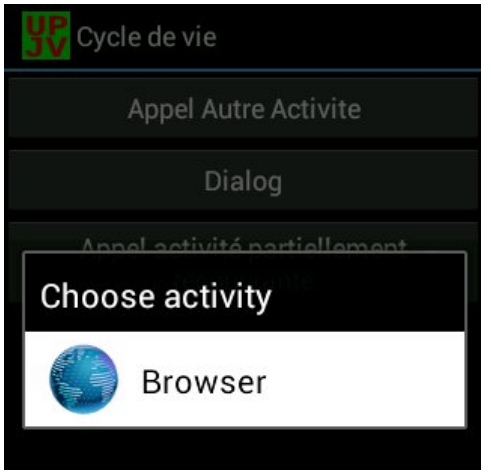
➔Bouton dialogue puis ne pas terminer l'activité :
Le dialogue fait partie de l'activité donc elle est toujours en état running : visible et foreground. L'utilisateur interagit avec elle.



Appel d'activité partiellement recouvrante puis touche back :

onPause
onResume

L'activité appelante est stoppée. Elle est partiellement recouverte, donc est état paused.



Cycle de vie de l'activité (6/10)

Cycle et Pile d'activités

➔fin de l'activité en cours : touche back ou home ou appel à la méthode finish() via le dialogue

- onPause
- onStop
- onDestroy

L'activité est supprimée de la pile d'activité et supprimée de la mémoire.

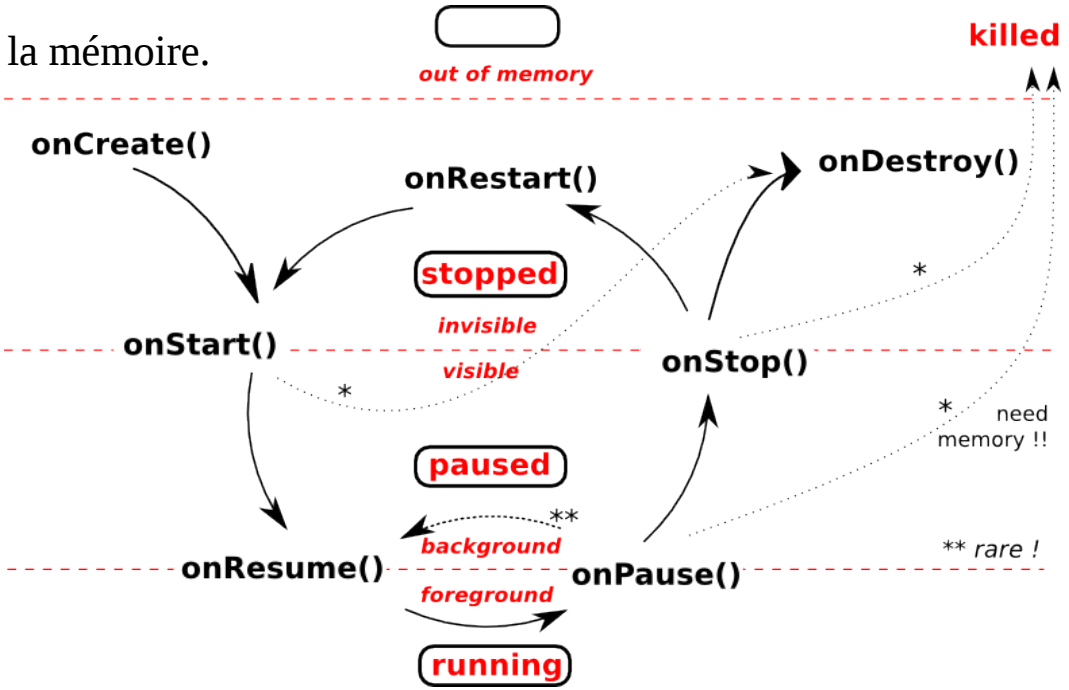
re-démarrage :

- onCreate
- onStart
- onResume

basculement d'écran :

- onPause
- onStop
- onDestroy
- onCreate
- onStart
- onResume

La rotation de l'écran (AVD : ctrl_F11 ou F12) entraîne la destruction de l'application car il faut exécuter la méthode onCreate() pour générer une nouvelle interface graphique.



Cycle de vie de l'activité (7/10)

Dialog et finish()

suite code :

```
...
  public void onClickDialog(View view) {
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setMessage("Terminer l'activité ?").setCancelable(false)
        .setPositiveButton("Oui", new DialogInterface.OnClickListener() {
          public void onClick(DialogInterface dialog, int id) {
            Log.v("CycleDeVieActivity", "finish activity");
            CycleDeVieActivity.this.finish();
          }
        }).setNegativeButton("Non", new DialogInterface.OnClickListener() {
          public void onClick(DialogInterface dialog, int id) {
            Log.v("CycleDeVieActivity", "dialog cancel");
            dialog.cancel();
          }
        });
    AlertDialog alert = builder.create();
    Log.v("CycleDeVieActivity", "Dialog");
    alert.show();
  }
```

→le 2éme bouton affiche un Dialog :
utilise la création dynamique de widget
utilise un Listener pour le bouton

Cycle de vie de l'activité (8/10)

Appel d'une activité partiellement recouvrante :

suite code :

```
...
public void onClickNonFullSizedActivity(View view) {
    Log.v("CycleDeVieActivity", "onClickNonFullSizedActivity");
    Intent intentionSouhaitee = new Intent(Intent.ACTION_VIEW,
            Uri.parse("http://www.u-picardie.fr"));
    Intent intention = new Intent(Intent.ACTION_PICK_ACTIVITY);
    intention.putExtra(Intent.EXTRA_INTENT, intentionSouhaitee);
    startActivityForResult(intention, 0);
  }
```

➔Demande de choisir une activité parmi celle qui peuvent afficher http://www.u-picardie.fr

Appel d'autre activité

```
public void onClickAutreActivite(View view) {
      Intent intent = new Intent(CycleDeVieActivity.this, AutreActivity.class);
      Log.v("CycleDeVieActivity", "onClickAutreActivite");
      startActivity(intent);
  }
```

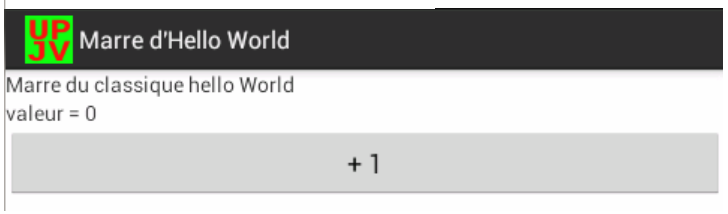
2ème classe minimale pour l'activité appelée:

```
package df.cours22;
public class AutreActivity extends Activity {
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.autre);
  }
}
```

Cycle de vie de l'activité (9/10)

MarreDHelloWorldActivity
SauverLesDonneesActivity

basculement écran ctrl-F11 ou F12



L' application est détruite donc les données en mémoire sont perdues, puisque non sauvegardées.

Sauver les données

➔En utilisant un Bundle = "array associatif" (une map) associant une clé string et un objet Parcelable (équivalent à Serializable)

putInt(clé, entier) sauve un entier
getInt(clé, défaut) restaure sinon retourne la valeurParDéfaut
containsKey(clé) …

```
package df.cours21;
...
public class SauverLesDonneesActivity  extends Activity
{
  private int val;
  private TextView text;
  public void onCreate(Bundle savedInstanceState) {
    ...
    if (savedInstanceState != null)
      val = savedInstanceState.getInt("val", -999);
    else
      val = 0;
    text.setText("valeur = "+val);
  }
  protected void onSaveInstanceState(Bundle outState) {
    super.onSaveInstanceState(outState);
    outState.putInt("val", val);
  }
...
```

Cycle de vie de l'activité (10/10)

SauverLesDonneesActivity

Sauver les données

Le mécanisme de sauvegarde et restauration ne coïncide pas complément avec le cycle de vie !

```
public void onCreate(Bundle savedInstanceState) {
  ...
  if (savedInstanceState != null)
    val = savedInstanceState.getInt("val", -999);
  else
    val = 0;
  text.setText("valeur = "+val);
}
protected void onSaveInstanceState(Bundle outState) {
  super.onSaveInstanceState(outState);
  outState.putInt("val", val);
}
```

➔onSaveInstanceState() est appelée quand l'application quitte le foreground et est susceptible d'être de nouveau en foreground. Elle est appelée avant onStop().
Attention ! Elle n'est pas toujours appelée : exemple, arrêt normal (touche back sur l'appli).
Appel obligatoire à super() qui sauve l'état des widgets.

Pour récupérer les valeurs :
soit dans le paramètre Bundle de la méthode onCreate
soit en « overridant » onRestoreInstanceState qui est appelé, si le bundle n'est pas vide, après onStart(). Appel obligatoire à super() qui restaure l'état des widgets.

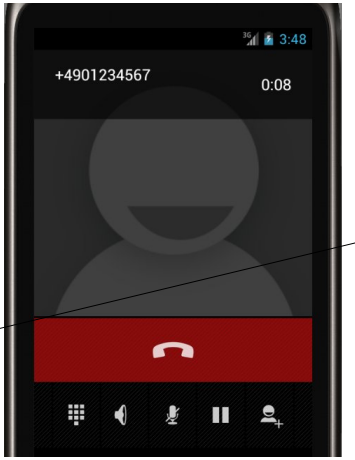
<u>Autre persistance des données :</u>
préférences attachées à l'application
lecture/écriture directe sur le système de fichiers sur stockage interne ou carte SD,
par base de données SQLite
voire, de façon partagée, par content provider

Permission (1/5)

Certaines activités appelées ci-dessous requiert des autorisations/permissions au moment de leur appel.
Comment notre activité appelante a-t'elle le droit de les appeler ?

```
Intent intention = new Intent(Intent.ACTION_CALL,
                  Uri.parse("tel:(+49)01234567"));
startActivity(intention);
```

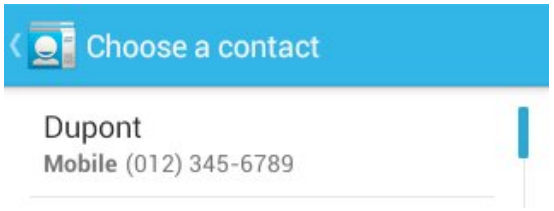
Lance l'appel téléphonique



Requiert une Permission

Ne requiert rien

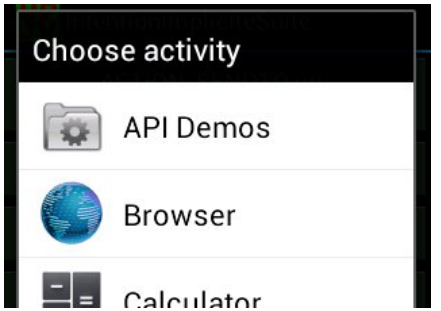
```
Intent intention = new Intent(Intent.ACTION_GET_CONTENT);
intention.setType("vnd.android.cursor.item/phone");
startActivity(intention);
```



Affiche la liste des contacts téléphoniques

```
Intent intentionSouhaitee = new Intent(Intent.ACTION_MAIN, null);
intentionSouhaitee.addCategory(Intent.CATEGORY_LAUNCHER);
Intent intention = new Intent(Intent.ACTION_PICK_ACTIVITY);
intention.putExtra(Intent.EXTRA_INTENT, intentionSouhaitee);
startActivityForResult(intention, 0);
```

Choisit et retourne une activité parmi celles souhaitées : celles du launcher



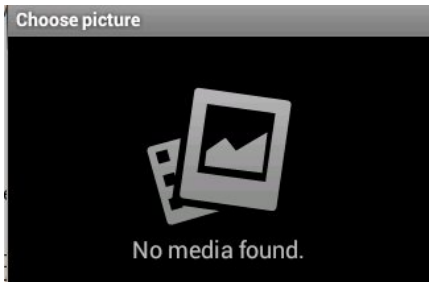
→ par les permissions dans le manifest

Filtre d'intention (1/10)

Comment les activités appelées ci-dessous font savoir qu'elles sont capables de répondre à l'intention indiquée ?

```
Uri uri = Uri.parse("smsto:0123456789");
Intent intention = new Intent(Intent.ACTION_SENDTO, uri);
intention.putExtra("sms_body", "Le texte du SMS ... a suivre");
startActivity(intention);
```

Envoyer un SMS à tel numéro
et avec un début de texte

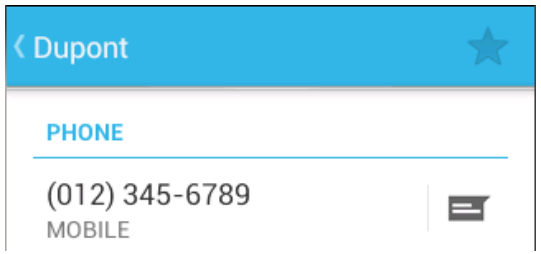



```
Intent intention = new Intent(Intent.ACTION_PICK);
intention.setType("image/*");
startActivityForResult(intention, 0);
```

Choisit et retourne une image

```
Intent intention = new Intent(Intent.ACTION_MAIN);
intention.addCategory(Intent.CATEGORY_HOME);
startActivity(intention);
```

Ecran d'accueil




```
Intent intention = new Intent(Intent.ACTION_VIEW,
        Uri.parse("content://contacts/people/1"));
startActivity(intention);
```

Afficher les informations de la personne numéro 1 dans les contacts

→ par les filtres d'intention … *à suivre*

Permission (2/5) ….

Permissions requises

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

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

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

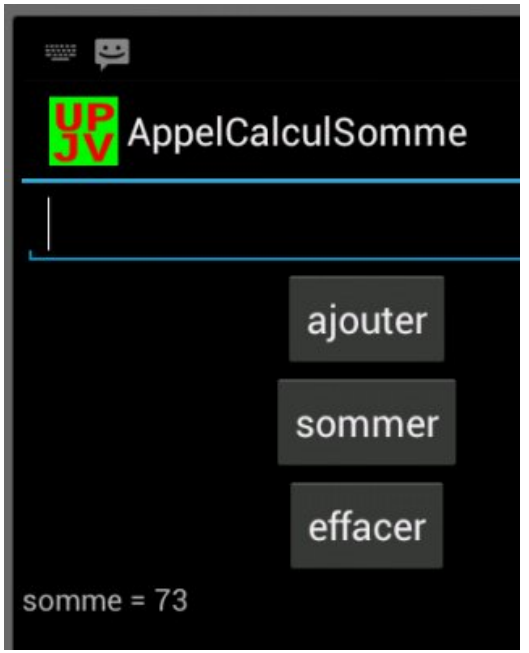
➔Le manifeste indique la/les permissions requises par les diverses sous-activités appelées

Normalement, cad hors développement, l'installation de notre activité demanderait à l'utilisateur s'il est d'accord pour autoriser ces permissions.

AppelCalculSommeActivity
CalculatriceActivity

Filtre d'intention (2/10)

```
package df.cours5;
...
public class AppelCalculSommeActivity extends Activity {
  private static final int REQUEST_CODE = 999;
  private ArrayList<Integer> entiers;
...
  public void onClickSommer(View view) {
    Intent intention = new Intent("df.cours.CALCULS");
    intention.putIntegerArrayListExtra("liste", entiers);
    intention.putExtra("operation", "+");
    startActivityForResult(intention, REQUEST_CODE);
  }
...
  protected void onActivityResult(int requestCode,
                                  int resultCode, Intent data) {
    if (resultCode == RESULT_OK && requestCode == REQUEST_CODE
        && data.hasExtra("result")) {
      int recu = data.getExtras().getInt("result",0);
      parametre.setText("");
      resultat.setText("somme = "+recu);
    } else ...
```



➔je définis ma propre action d'intention : df.cours.CALCULS

Filtre d'intention (3/10)

Où est le filtre d'intention ?

```
package df.cours6;
...
public class CalculatriceActivity extends Activity {
  private static final int PLUS=1, MOINS=2, MULT=3, DIV=4;

  public void onCreate(Bundle bundle) {
    super.onCreate(bundle);
    if (getIntent().hasExtra("liste")
       && getIntent().hasExtra("operation")) {
        ...
        for (int i = 1; i < nbre; ++i)
          result = (op==PLUS?result+entiers.get(i):
            (op==MOINS?result-entiers.get(i):
              (op==MULT?result*entiers.get(i):
                 result/entiers.get(i))));
        Intent intentionResult = new Intent();
        intentionResult.putExtra("result", result);
        setResult(RESULT_OK, intentionResult);
        this.finish();
      }
    }
    setResult(RESULT_CANCELED, null);
    this.finish();
  }
}
```

→L'activité CalculatriceActivity susceptible de convenir est dans un autre package

Rien n'indique dans son code qu'elle convient pour mon action df.cours.CALCULS

A signaler l'utilisation du code retour RESULT_CANCELED
et la méthode hasExtra()

Filtre d'intention (4/10)

Le filtre d'intention

AndroidManifest.xml de CalculatriceActivity :

```
<application
    android:icon="@drawable/upjv_launcher"
    android:label="@string/app_name" >
    <activity
        android:name=".CalculatriceActivity"
        android:label="@string/app_name" >
        <intent-filter>
            <action android:name="df.cours.CALCULS" />
            <category android:name="android.intent.category.DEFAULT" />
        </intent-filter>
    </activity>
</application>
```

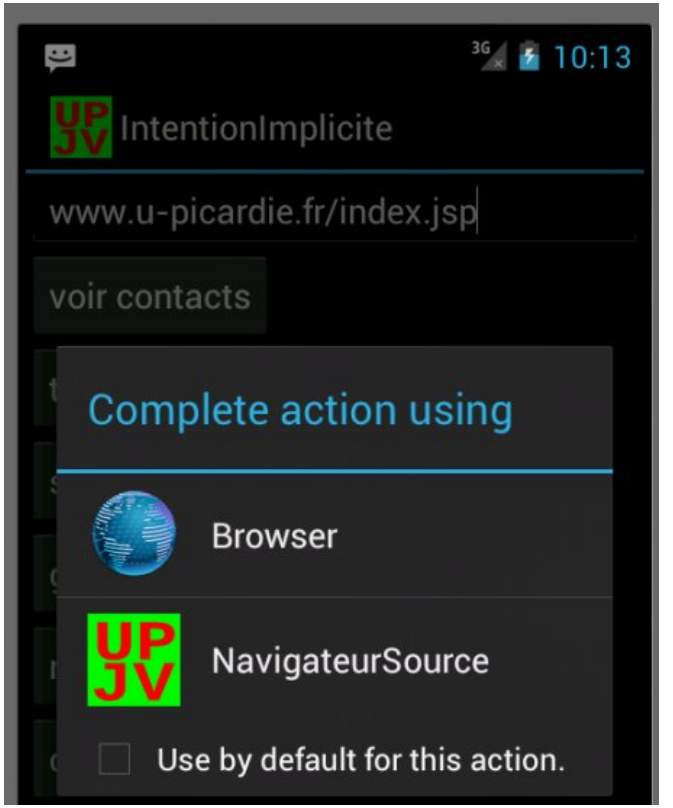
➔dans l'intent-filter de l'activité, figure mon action df.cours.CALCULS

une category est indispensable donc par défaut category DEFAULT
sinon on obtient une erreur `E/AndroidRuntime(840): Caused by: android.content.ActivityNotFoundException: No Activity found to handle Intent { act=df.cours.CALCULS (has extras) }`

Intent Filter :
Les filtres d'intention spécifient le type d'intention qu'un composant (activité, service, broadcast receiver) souhaite traiter. Cela permet au système Android qui reçoit une intention de déterminer quels composants enregistrés peuvent la traiter.

Filtre d'intention (5/10)

➔Exécuter IntentionImplicite avec le paramètre www.u-picardie.fr/index.jsp
un dialogue apparaît avec un choix d'activités possibles et la possibilité d'en définir une comme action par défaut



Même filtre d'intention

AndroidManisfeste.xml de NavigateurSourceActivity :

```
<uses-permission android:name="android.permission.INTERNET

<application
    android:icon="@drawable/upjv_launcher"
    android:label="@string/app_name" >
    <activity
       android:name=".NavigateurSourceActivity"
       android:label="@string/app_name" >
       <intent-filter>
          <action android:name="android.intent.action.VIEW" />
          <category android:name="android.intent.category.DEFAULT" />
          <data android:scheme="http" />
       </intent-filter>
    </activity>
</application>
```

Son intent-filter traite les intentions dont
- l'action est VIEW
- la catégorie DEFAULT
- le type d'URI utilise le http

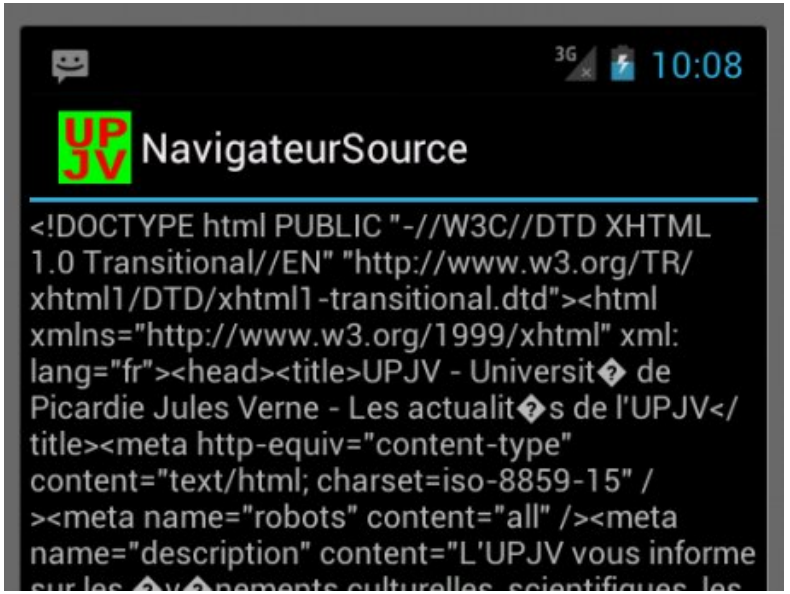
il est donc en concurrence avec le browser installé sur Android
A signaler que pour voir la page web il faut la permission INTERNET

Filtre d'intention (++ 6/10)

```
package df.cours2;
public class NavigateurSourceActivity extends Activity {
  ...
  private Handler handler = new Handler() {
    public void handleMessage(Message msg) {
      text.setText(source.toString());
    }
  };
  public void onCreate(Bundle savedInstanceState) {
    ...
    final Uri data = getIntent().getData();
    Thread telecharger = new Thread(new Runnable() {
      public void run() {
          URL url = new URL(data.getScheme(), data.getHost(), data.getPath());
          BufferedReader flot = new BufferedReader(new InputStreamReader(
              url.openStream()));
          String line = "";
          while ((line = flot.readLine()) != null) {
            source.append(line);
          }
          flot.close();
          handler.sendEmptyMessage(5);
      }
    });
    telecharger.start();
  }
}
```

Pour curiosité, le code source de NavigateurSourceActivity :
il télécharge et affiche le corps de la réponse HTTP

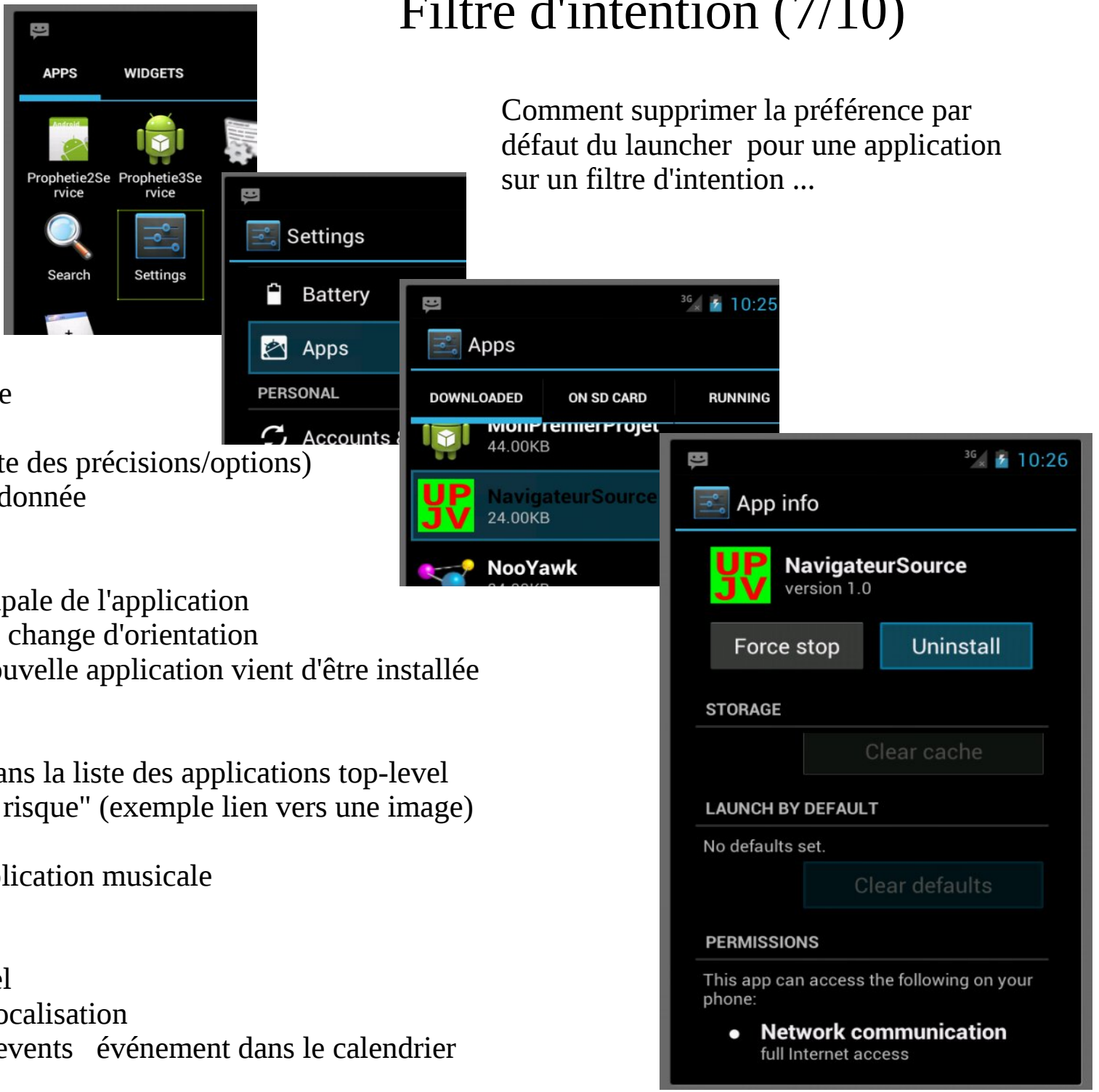
Thread et Handler seront étudiés ultérieurement … dans la 2nde partie du cours



Filtre d'intention (7/10)

Comment supprimer la préférence par défaut du launcher pour une application sur un filtre d'intention ...

Action catégorie URI/type



Rappel :
Une intention implicite comporte
- 0 ou 1 action
- 0 ou plusieurs catégories (ajoute des précisions/options)
- 0 ou plusieurs URI et type de donnée

exemple d'action :
ACTION_MAIN activité principale de l'application
ACTION_SCREEN_ON l'écran change d'orientation
ACTION_SCREEN_ON une nouvelle application vient d'être installée

exemple de catégorie :
LAUNCHER l'activité figure dans la liste des applications top-level
BROWSABLE affichable "sans risque" (exemple lien vers une image)
DEFAULT
CATEGORY_APP_MUSIC application musicale

uri et type :
tel:678954322 numéro de tel
Geo:49.895243,2.29846 géolocalisation
content://com.android.calendar/events événement dans le calendrier

Filtre d'intention (8/10) <intent-filter>

un filtre pour une activité qui débute une application et figure dans le "launcher" du terminal :

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

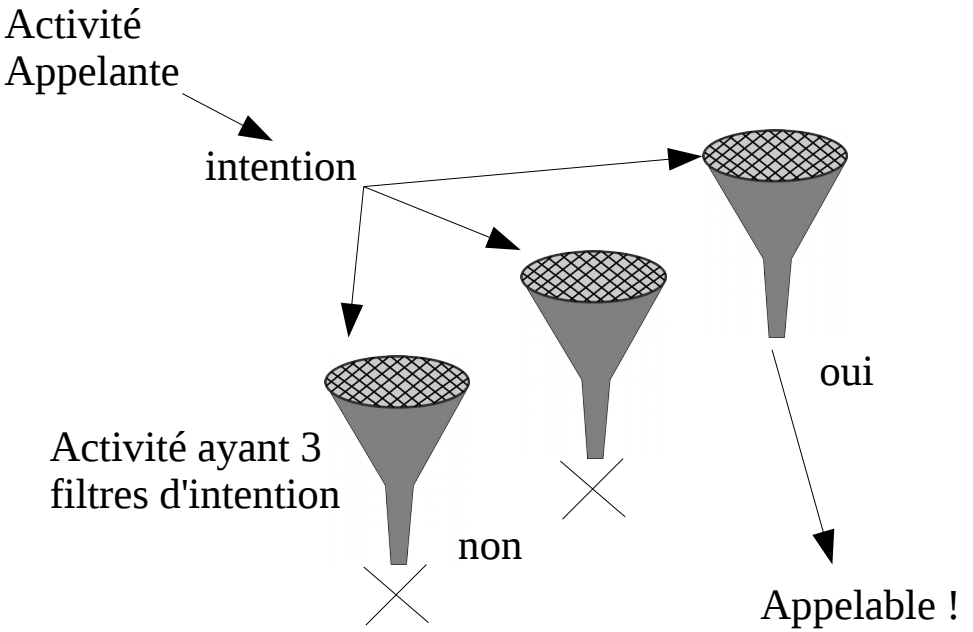
un filtre pour un mini éditeur :

```
 <intent-filter>
    <action android:name="android.intent.action.VIEW" />
    <action android:name="android.intent.action.EDIT" />
    <action android:name="android.intent.action.PICK" />
    <category android:name="android.intent.category.DEFAULT" />
    <data android:mimeType="vnd.android.cursor.dir/vnd.google.note" />
</intent-filter>
```

Ces filtres sont dans le fichier manifeste sous la forme d'élément intent-filter (Exception pour les BroadcastReceiver qui peuvent l'avoir dynamiquement par programmation)

Le manifeste peut comporter, pour chaque activité ou service décrit, plusieurs filtres d'intention voire aucun (dans ce cas, l'activité ne peut être lancée que par intention explicite).
Il suffit qu'un filtre soit positif pour lancer l'activité ou service.

Un filtre comporte des filtres d'actions, des filtres de catégories et des filtres de données/types de données
Pour "réussir" le filtre, il faut "réussir" la partie action, la partie catégorie et la partie data.



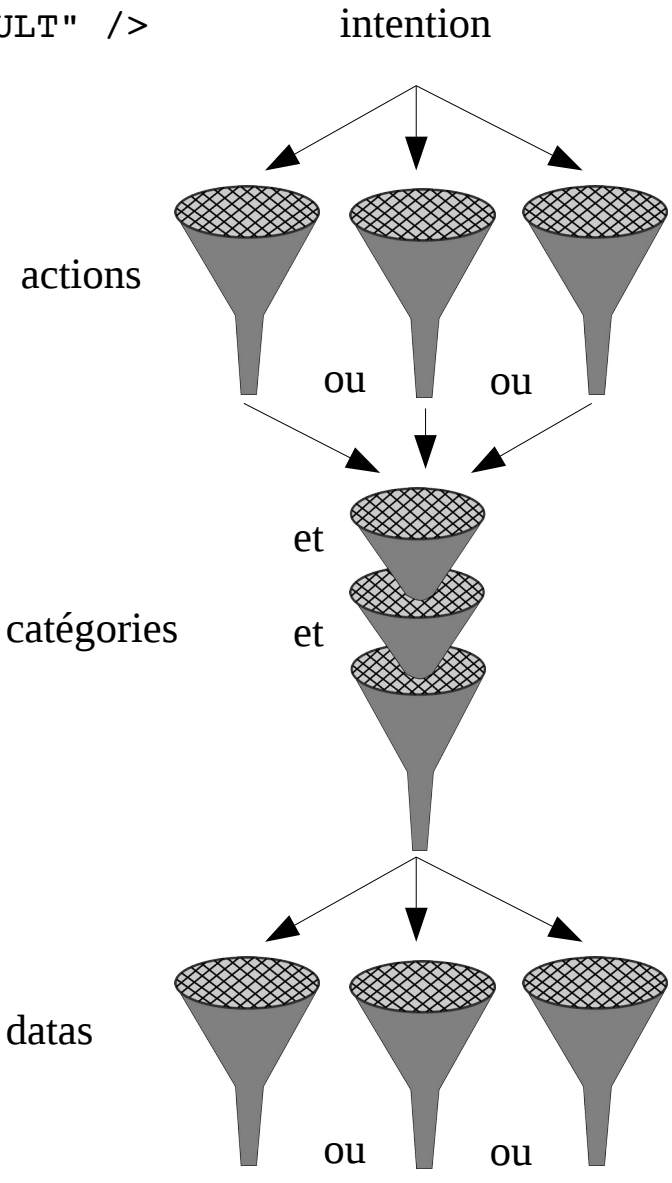
Filtre d'intention (9/10) <intent-filter>

```
<activity
   android:name=".NavigateurSourceActivity"
   android:label="@string/app_name" >
   <intent-filter>
      <action android:name="android.intent.action.VIEW" />
      <category android:name="android.intent.category.DEFAULT" />
      <data android:scheme="http" />
   </intent-filter>
</activity>
```

La partie **action** comporte au moins un filtre action :
il suffit qu'un filtre action réussisse : cad qu'une des actions de l'intention corresponde à un des filtres action.
Quand l'intention n'a pas d'action, cette partie est réussie.

la partie **catégorie** :
pour réussir, il faut que toutes les catégories de l'intention correspondent avec une catégorie du filtre.
Si l'intention ne contient pas de catégorie, elle réussit.
Sauf, lors de l'appel startActivity d'une intention implicite sans catégorie, la catégorie CATEGORY_DEFAULT est ajoutée !

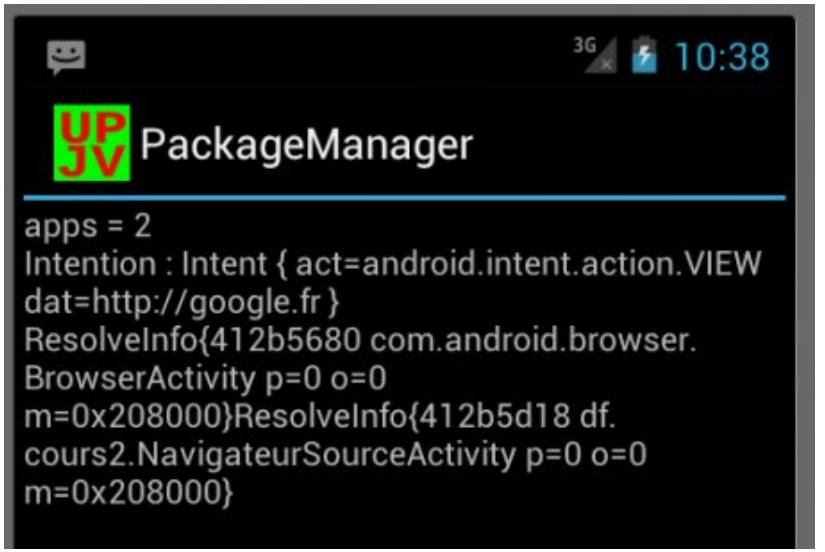
la partie **data** :
il peut y avoir 0 ou plusieurs éléments <data />
qui comporte des attributs uri et un attribut type
l'uri est composé par les attributs scheme, host, port et path
ainsi : "scheme://host:port/path"
pour réussir, il suffit qu'un filtre data corresponde.



Filtre d'intention (10/10)

La résolution d'une intention relativement aux filtres d'intention des activités (et services et…) installées est réalisé par le PackageManager.

Un certain nombre de managers constituent le Framework qui fait fonctionner les applications entre-elles.



PackageManager

```
package df.cours4;
...
public class PackageManagerActivity extends Activity {
  public void onCreate(Bundle savedInstanceState) {
    ....
    PackageManager packageManager = this.getPackageManager();
    Intent intention = new Intent(Intent.ACTION_VIEW,
                                  Uri.parse("http://google.fr"));
    List<ResolveInfo> listResolveInfo =
        packageManager.queryIntentActivities(intention,0);
    int nbre = listResolveInfo.size();
    if ( nbre == 0)
      text.setText("pas d'apps pour l'intention : "+intention.toString());
    else {
      text.setText("apps = "+nbre+"\nIntention : "+intention.toString()+"\n");
      for (ResolveInfo resolveInfo : listResolveInfo)
        text.append(resolveInfo.toString()+"\n");
    }
  }
```

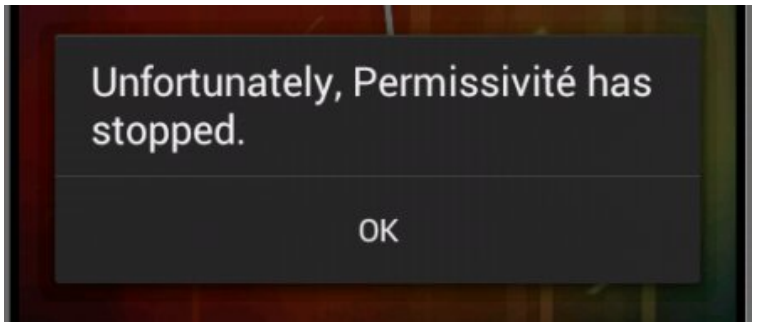
Permissions (3/5)


```
package df.cours14;
public class PermissiviteActivity extends Activity {
  public void onClickTelephonerAvecFilet(View view) {
    try {
      Intent intention = new Intent();
      intention.setAction(android.content.Intent.ACTION_CALL);
      intention.setData(Uri.parse("tel:12345678"));
      startActivity(intention);
    } catch (ActivityNotFoundException anfe) {
      Log.e("PermissiviteActivity", "telephoner", anfe);
    } catch (SecurityException se) {
      Log.e("PermissiviteActivity","telephoner sans filet",se);
    }
  }
...
```

N'ayant pas la permission android.permission.CALL_PHONE

```
04-24 08:29:56.795: E/PermissiviteActivity(1014):
java.lang.SecurityException: Permission Denial: starting Intent
{ act=android.intent.action.CALL dat=tel:xxxxxxxx
cmp=com.android.phone/.OutgoingCallBroadcaster } from
ProcessRecord{415268e8 1014:df.cours14/10075} (pid=1014, uid=10075)
requires android.permission.CALL_PHONE
```

Ci-contre le clic "sans filet",
cad sans capture de SecurityException



Permissions (4/5)

<permission> et attribut d'activity

```
<permission android:name="df.cours.VULGARITE"
    android:label="@string/permission_label"
    android:description="@string/permission_description"
 />
<application
    android:icon="@drawable/upjv_launcher"
    android:label="@string/app_name" >
    <activity
       android:permission="df.cours.permission.VULGARITE"
       android:name=".PetageDePlombActivity"
       android:label="@string/app_name" >
       <intent-filter>
          <action android:name="df.cours13.PETAGE" />
          <category android:name="android.intent.category.DEFAULT" />
       </intent-filter>
    </activity>
```

Le manifeste de PetageDePlombActivity indique une nouvelle permission df.cours.permission.VULGARITE et la requiert pour lancer l'activité.

Dans ressources :

```
<string name="permission_label">vulgarité vraiment vulgaire</string>
 <string name="permission_description">interdit au moins de 18 ans.
                réservé aux personnes de douteuse moralité</string>
```

L'utilisateur doit confirmer ces permissions à chaque installation d'une application.
Sauf en développement
Toute activité appelante devra la présenter !

Il n'y a pas de mécanisme pour accorder dynamiquement des permissions
donc uniquement et statiquement dans le manifeste.

Permissions (5/5)

<uses-permission>

le manifeste de PermissiviteActivity
indique qu'il autorise la permission utilisateur : df.cours.VULGARITE
Donc PermissiviteActivity peut appeler une activité qui la réclame !

```
<uses-permission android:name="df.cours.permission.VULGARITE" />
<application
   android:icon="@drawable/upjv_launcher"
   android:label="@string/app_name" >
   <activity
      android:name="df.cours14.PermissiviteActivity"
      ...
```

Les permissions systèmes commencent par android.permission.
Exemples : INTERNET pour ouvrir une connexion
WRITE_EXTERNAL_STORAGE pour écrire sur la carte DS ou tout autre support de stockage externe
ACCESS_COARSE_LOCATION et ACCESS_FINE_LOCATION si l'application a besoin de connaître l'emplacement du terminal

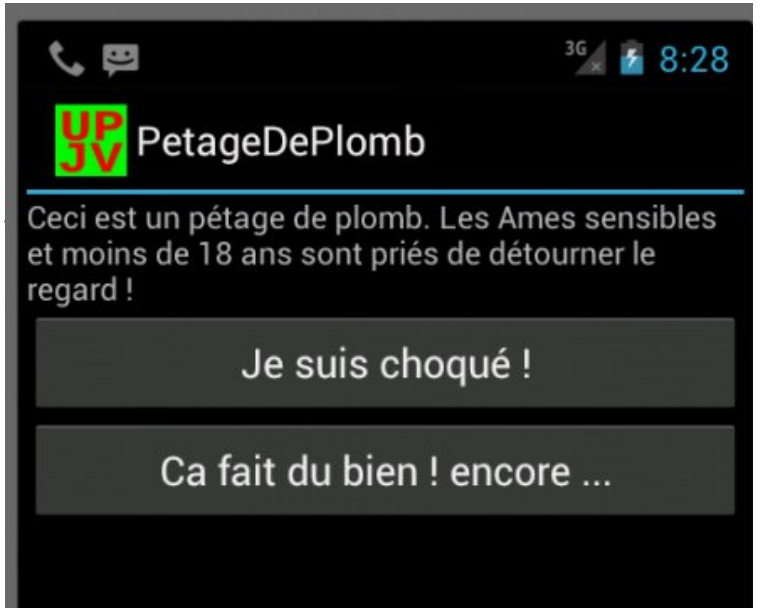
```
public class PermissiviteActivity ….
  public void onClickSurfer(View view) {
    Intent intention = new Intent(Intent.ACTION_VIEW,
                                  Uri.parse("http://www.u-picardie.fr"));
    startActivity(intention);
```

Pas besoin de <uses-permission … INTERNET/>
car le Browser requiert la permission INTERNET auprès de l'utilisateur lors de son installation
puis n'en réclame pas quand il est appelé par d'autres activités

Notifier l'utilisateur (1/2) : Toast

PetageDePlombActivity

```
package df.cours13;
public class PetageDePlombActivity extends Activity {
  private static final String[] grossieretes = { ... }
  public void onClickChoque (View view) {
    finish();
  }
  public void onClickEncore(View view) {
    seLacher();
  }
  private void seLacher() {
    Context context = getApplicationContext();
    pos = (pos+1) % grossieretes.length;
    int duree = Toast.LENGTH_SHORT;
    Toast toast = Toast.makeText(context, grossieretes[pos], duree);
    toast.show();
  }
```

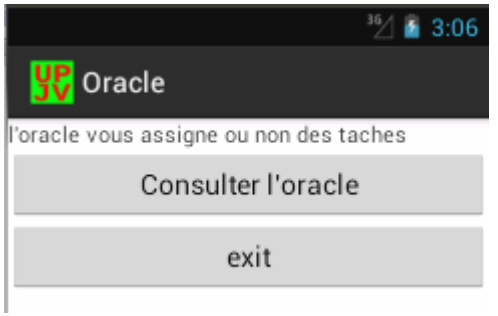


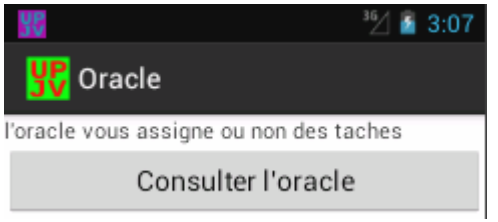
Toast :une forme de notification à l'utilisateur
affichage d'une info en premier-plan selon une durée déterminée LENGTH_SHORT ou LENGTH_LONG

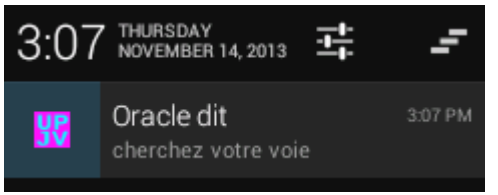
3 formes de Notification sont offertes par le framework :
les widgets Dialog qui sont impératifs : obligent à une action immédiate
les Toasts qui sont évanescents
et les Notifications qui signalent une information qui peut être traitée plus tard : exemple l'arrivée d'un SMS

Notifier l'utilisateur (2/2) : Notification

```
public void oracler(View view) {
  if (Math.random()>0.5) {
    String titreNotif = "Oracle dit";
    String textNotif;
    Intent intentionNotif;
    if (Math.random()>0.5) {
      textNotif = "cherchez votre voie";
      intentionNotif = new Intent(Intent.ACTION_WEB_SEARCH);
      intentionNotif.putExtra(SearchManager.QUERY, "Boudda");
    } else {
      textNotif = "allez vous faire ....";
      intentionNotif = new Intent("df.cours13.PETAGE");
    }
    Notification.Builder monBuilder = new Notification.Builder(this);
    monBuilder.setSmallIcon(R.drawable.upjv_notif)
            .setContentTitle(titreNotif)
            .setContentText(textNotif)
            .setAutoCancel(true);
    PendingIntent aFaire = PendingIntent.getActivity( this, 0,
            IntentionNotif, PendingIntent.FLAG_UPDATE_CURRENT);
    monBuilder.setContentIntent(aFaire);
    int maNotif = 001;
    NotificationManager notifyMgr =
            (NotificationManager) getSystemService(NOTIFICATION_SERVICE);
    notifyMgr.notify(maNotif, monBuilder.build());
  }
}
```





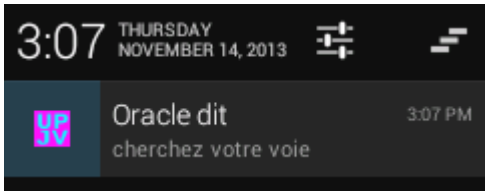


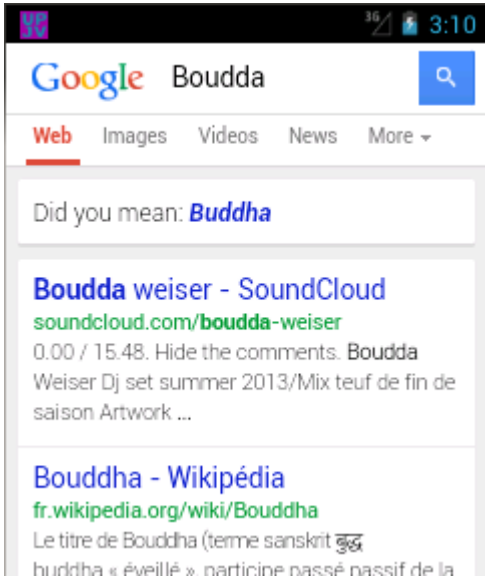
La Notification est une information signalée à l'utilisateur dans la barre en haut du mobile. Elle se déroule hors de l'UI thread de l'application et peut être traitée plus tard, car elle est gérée par le Notification Manager.

Intention (11/... 10) :

Intention différée dans une notification PendingIntent

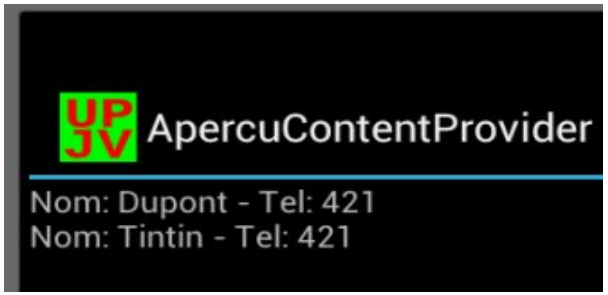
```
...
      intentionNotif = new Intent(Intent.ACTION_WEB_SEARCH);
      intentionNotif.putExtra(SearchManager.QUERY, "Boudda");
...
    Notification.Builder monBuilder = new Notification.Builder(this);
    PendingIntent aFaire = PendingIntent.getActivity( this, 0,
            IntentionNotif, PendingIntent.FLAG_UPDATE_CURRENT);
    monBuilder.setContentIntent(aFaire);
...
    notifyMgr.notify(maNotif, monBuilder.build());
  }
}
```



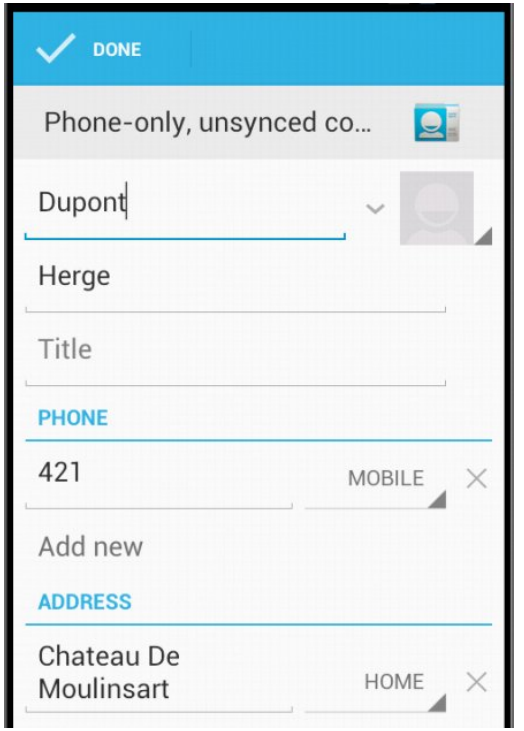


La Notification peut comporter une intention à exécuter quand et si l'utilisateur la déclenche.
Cette intention, qui n'est pas immédiatement exécutée sous forme d'activité, est une pending intent : une intention différée.

Content Provider (1/16) : Contacts



Avant de lancer l'application,
Ajoutez quelques contacts
dans votre AVD.




```
package df.cours17;
import android.database.Cursor;
import android.provider.ContactsContract;
import android.provider.ContactsContract.CommonDataKinds.Phone;
...
public class ApercuContentProviderActivity extends Activity {
  public void onCreate(Bundle savedInstanceState) {
    ...
    Uri uri = ContactsContract.CommonDataKinds.Phone.CONTENT_URI;
    String[] projection = new String[] { Phone._ID,Phone.DISPLAY_NAME, Phone.NUMBER };
    String selection = ContactsContract.Contacts.IN_VISIBLE_GROUP + " = '1'";
    String[] selectionArgs = null;
    String sortOrder = Phone.DISPLAY_NAME + " ASC";
    Cursor cursor = getContentResolver().query(uri, projection, selection,
                                               selectionArgs, sortOrder);
    while (cursor.moveToNext())
      contactsView.append("Nom: "
                +cursor.getString(cursor.getColumnIndex(Phone.DISPLAY_NAME))
                +" - Tel: "+cursor.getString(cursor.getColumnIndex(Phone.NUMBER))+"\n");
    cursor.close();
  }
}
```

Content Provider (2/16)

= Fournisseur de données accessibles par plusieurs applications
avec une architecture REST-like (Representational State Transfer)
Car elle utilise l'URI comme identifiant des ressources
et les opérations possibles sont CRUD.

Content URI : pour identifier les contents providers
exemples : content://contacts/people/23 content://media/internal/images
content://com.android.calendar/events content://com.android.calendar/time/3456209
forme = content://authority-name/path-segment1/path-segment2/....
Ainsi dans content://df.cours.bibliotheque/auteurs
df.cours.bibliotheque est l'autority, auteurs est le 1er path segment,
content://df.cours.bibliotheque/auteur/23/livres auteur 23 est 1er path segment, livres le 2nd
content://df.cours.bibliotheque/auteur/23/livre/4 livre et 4 est le second path segment

Méthodes : insert()
update()
delete()
query() avec une technique de Cursor pour lire le ou les "rows" résultat.

Permissions : android.permission.READ_CALENDAR
android.permission.WRITE_CALENDAR

Content Provider (3/16)

Type MIME :

standards text/htm audio/x-aiff application/pdf

"vendor-specific" vnd.android.cursor.dir/com.android.calendar pour plusieurs rows,
vnd.android.cursor.item/com.android.calendar pour un seul

Une requête d'URI content://df.cours.bibliotheque/auteurs a probablement plusieurs lignes résultats et donc pour type mime vnd.android.cursor.dir/df.cours.bibliotheque

Une requête d'URI content://df.cours.bibliotheque/auteur/23 a au mieux 1 ligne résultat et donc pour type mime vnd.android.cursor.item/df.cours.bibliotheque

Content Provider : de base

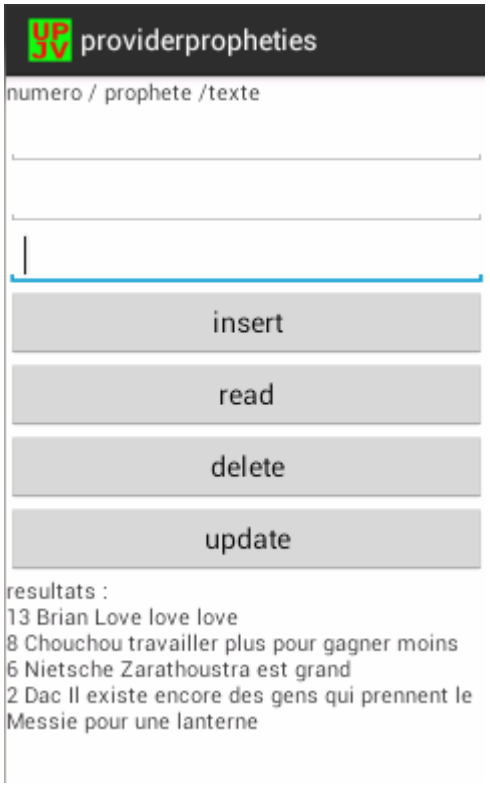
Calendar provider des événements
Contacts provider des informations sur les contacts
Media provider des audios et vidéos, artistes, playlists, ...
Settings provider des préférences systèmes et user
Telephony provider sur les SMS, ...
UserDictionary provider pour les mots proposés en rédaction assisté

Implémentation :

Table SQLite : Android dispose du moteur de base de données relationnelle SQLite3 très économe.
Fichiers
Service accessible via le réseau

Donc l'accès aux données via le content provider est unifiée/standardisée quelque soit son implémentation.

Content Provider (4/16) : insert()



Voici un provider de contenu ... prophétique :
Chaque prophétie a un numéro d'identification entier, un prophète et le texte de la prophétie.

Les 4 opérations de bases sont :
- afficher toutes les prophéties, ci-contre à gauche
- en créer, ci-contre à droite
- en modifier une en précisant son numéro d'identification
- en supprimer une en précisant son numéro.


```
import PropheticsContract.Propheties;
public class AccesProviderProphetieActivity extends Activity {
...
public void actionInsert(View v) {
  ContentValues valeur = new ContentValues();
  valeur.put(Propheties.COLUMN_NAME_PROPHETIE_ID, saisieNumero);
  valeur.put(Propheties.COLUMN_NAME_PROPHETE, saisieProphete);
  valeur.put(Propheties.COLUMN_NAME_TEXTE, saisieTexte);
  Uri uri = getContentResolver().insert(PropheticsContract.CONTENT_URI, valeur);
  textAffichage.setText("insertion "+uri.toString());
}
```

Les accès aux données de content provider se font au travers de l'objet ContentResolver

Content Provider (5/16) : Contract class

La « Contract » class d'un provider regroupe toutes les caractéristiques constantes : content_uri, mime type, noms des colonnes, … Ce n'est pas obligatoire mais pratique !

```
public final class ProphetiesContract {
    public static final String AUTHORITY = "df.cours";
    public static final String BASE_PATH = "propheties";
    public static final Uri CONTENT_URI =
                    Uri.parse("content://" + AUTHORITY + "/" + BASE_PATH);
    public static final String CONTENT_TYPE =
                    ContentResolver.CURSOR_DIR_BASE_TYPE + "/" + BASE_PATH;
    public static final String CONTENT_ITEM_TYPE =
                    ContentResolver.CURSOR_ITEM_BASE_TYPE + "/prophetie";
    public static final  class Propheties implements BaseColumns {
        public static final String TABLE_NAME = "prophetie";
        public static final String COLUMN_NAME_PROPHETIE_ID = "prophetieid";
        public static final String COLUMN_NAME_PROPHETE = "prophete";
        public static final String COLUMN_NAME_TEXTE = "texte";
    }
}
```

delete() a comme paramètre l'uri de la ligne à supprimer

suite AccesProviderProphetieActivity :

```
...
public void actionDelete(View v) {
    int numero = Integer.parseInt(saisieNumero);
    Uri uri = ContentUris.withAppendedId(ProphetiesContract.CONTENT_URI, numero);
    int combien = getContentResolver().delete(uri,  null, null);
    textAffichage.setText(combien + " supprimé(s)");
}
```

La méthode delete(Uri, String where, String[] selectionArgs) peut préciser une clause WHERE

La méthode withAppendedId() aide à composer l'uri

Content Provider (6/16) : query()

La lecture des données se fait en fournissant, outre l'Uri, les noms des colonnes des informations souhaitées.

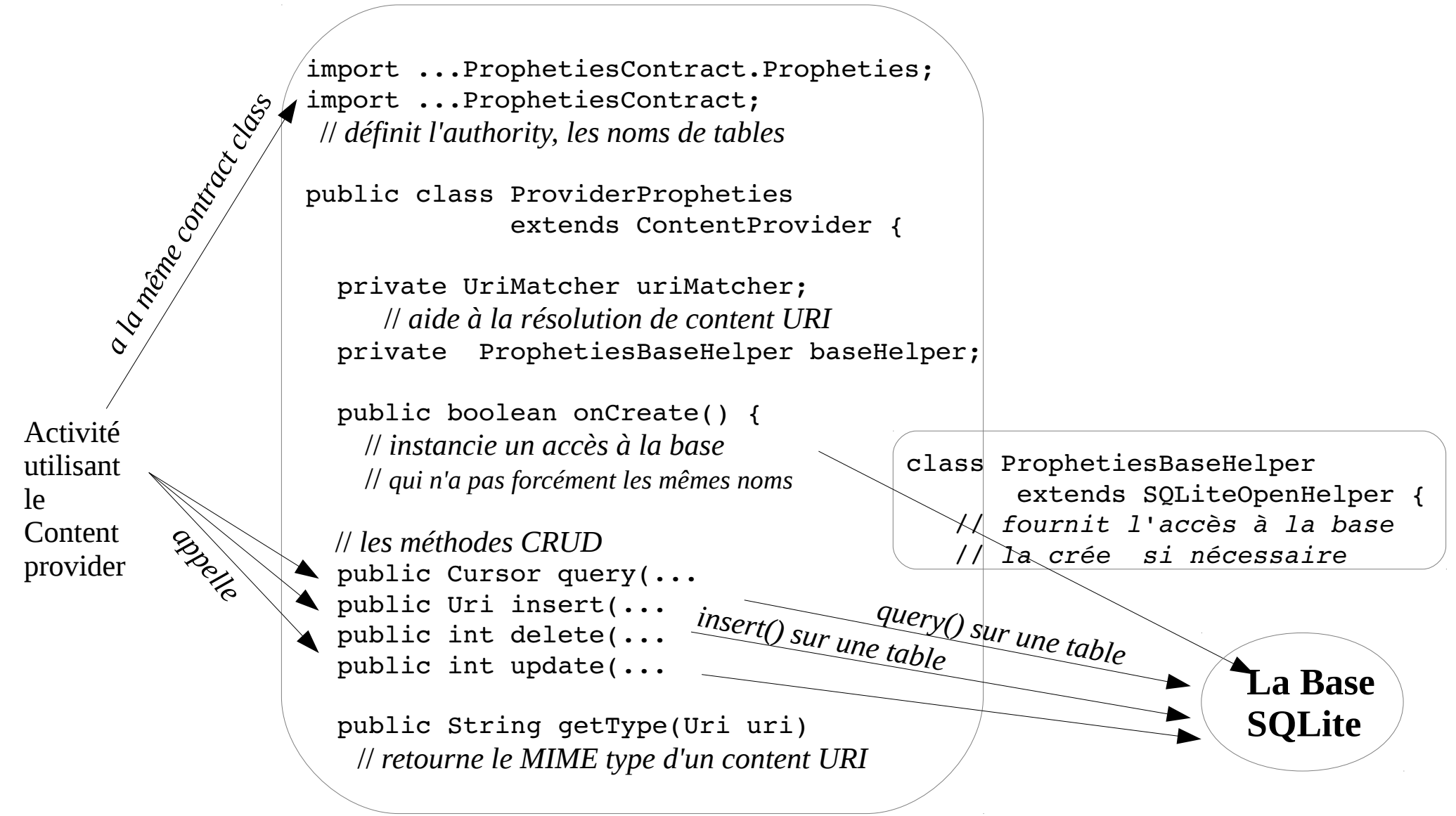
suite AccesProviderProphetieActivity :

```
…
public void actionRead(View v) {
  String[] projection = { Propheties._ID, Propheties.COLUMN_NAME_PROPHETIE_ID,
                          Propheties.COLUMN_NAME_PROPHETE,
                          Propheties.COLUMN_NAME_TEXTE };
  Cursor c = getContentResolver().query(ProphetiesContract.CONTENT_URI,
                                        projection, null, null, null);
  StringBuffer result = new StringBuffer("resultats :\n");
  if (c != null) {
    while (c.moveToNext())
      result.append(c.getString(c.getColumnIndex(Propheties.COLUMN_NAME_PROPHETIE_ID))
                    +" "+c.getString(c.getColumnIndex(Propheties.COLUMN_NAME_PROPHETE))
                    +" "+c.getString(c.getColumnIndex(Propheties.COLUMN_NAME_TEXTE))
                    +"\n");
     c.close();
  }
  textAffichage.setText(result.toString());
}
```

La méthode query(uri, projection, String selection, String[] selectionArgs, String sortOrder)
peut préciser la sélection clause WHERE avec ses valeurs et la clause ORDER-BY.

Le Cursor est une collection de lignes
Les méthodes d'extraction des données des colonnes sont basées sur le type et le numéro de colonne (la 1ère est numérotée 1!)

Content Provider (7/16) : implémentation avec une base SQLite



Content Provider (8/16) : implémentation avec une base SQLite

Classe d'assistance pour créer, ouvrir et gérer une base de données.

```
public class ProphetiesBaseHelper extends SQLiteOpenHelper {

  private static final String SQL_CREATE_ENTRIES =
        "CREATE TABLE " + Propheties.TABLE_NAME + " (" +
        Propheties._ID + " INTEGER PRIMARY KEY, " +
        Propheties.COLUMN_NAME_PROPHETIE_ID + " TEXTE, " +
        Propheties.COLUMN_NAME_PROPHETE + " TEXTE, " +
        Propheties.COLUMN_NAME_TEXTE + " TEXTE )";

  private static final String SQL_DELETE_ENTRIES =
        "DROP TABLE IF EXISTS " + Propheties.TABLE_NAME;

  public static final int DATABASE_VERSION = 1;
  public static final String DATABASE_NAME = "GenialesPropheties";

  public ProphetiesBaseHelper(Context context) {
    super(context, DATABASE_NAME, null, DATABASE_VERSION);
  }

  public void onCreate(SQLiteDatabase maBase) {
    maBase.execSQL(SQL_CREATE_ENTRIES);
  }

  public void onUpgrade(SQLiteDatabase maBase, int oldVersion, int newVersion) {
    maBase.execSQL(SQL_DELETE_ENTRIES);
    onCreate(maBase);
  }
```

Le nom de la base et des tables n'est pas forcément le même que ceux du content provider.

Les types de données d'une base SQLite sont limités à TEXT, INTEGER, REAL, NONE mais un mécanisme de typage dynamique permet de stocker tous les types de données.
Il n'y a pas de gestion des droits d'accès sur les tables.

Les méthodes abstraites onCreate et onUpgrade sont à définir.

Content Provider (9/16) : explorer une base SQLite

L'outil adb (Android Debug Bridge) permet de gérer le terminal android en console :
Lancez un shell
puis la commande sqlite3

```
$ cd Android/Sdk/platform-tools
$ ./adb devices
List of devices attached
emulator-5554 device
$ ./adb -s emulator-5554 shell
root@generic:/ # ls /data/data/df.cours34/databases
GenialesPropheties
GenialesPropheties-journal
```

La base est stockée relativement à l'application qui l'a crée : l'intégralité d'une base SQLite est stockée dans un seul fichier.

```
root@generic:/ # sqlite3 /data/data/df.cours34/databases/GenialesPropheties
SQLite version 3.7.11 2012-03-20 11:35:50
sqlite> .databases
seq  name             file
---  ---------------  ----------------------------------------------------
0    main             /data/data/df.cours34/databases/GenialesPropheties
sqlite> .tables
android_metadata  prophetie
sqlite> .dump prophetie
PRAGMA foreign_keys=OFF;
BEGIN TRANSACTION;
CREATE TABLE prophetie (_id INTEGER PRIMARY KEY, prophetieid TEXTE, prophete TEXTE, texte
TEXTE );
INSERT INTO prophetie VALUES(1,'4','c','ccc');
COMMIT;
sqlite> .exit
```

Content Provider (10/16) : implémentation avec une base SQLite

```
public class ProviderPropheties extends ContentProvider {

    private static final int PROPHETIES = 1;
    private static final int PROPHETIE_ID = 2;
    private static final UriMatcher uriMatcher;
    static {
        uriMatcher = new UriMatcher(UriMatcher.NO_MATCH);
        uriMatcher.addURI(ProphetiesContract.AUTHORITY,
                          ProphetiesContract.BASE_PATH, PROPHETIES);
        uriMatcher.addURI(ProphetiesContract.AUTHORITY,
                          ProphetiesContract.BASE_PATH+"/#", PROPHETIE_ID);
    }
```

L'UriMatcher identifie le type URI.
Le # représente un identifiant entier, tandis que * représente différents paths.

```
private  ProphetiesBaseHelper baseHelper = null;

    public boolean onCreate() {
        baseHelper = new ProphetiesBaseHelper(getContext());
        return true;
    }
```

onCreate() appelle l'assistant de manipulation de la base de données associée à ce provider.

Content Provider (11/16) : implémentation d'insert() et getType()

```
public Uri insert(Uri uri, ContentValues valeur) {
    SQLiteDatabase db = baseHelper.getWritableDatabase();
    int uriType = uriMatcher.match(uri);
    long id = 0;
    switch (uriType) {
    case PROPHETIES:
        id = db.insert(Propheties.TABLE_NAME, null, valeur);
        break;
    default:
        throw new IllegalArgumentException("Unknown URI: " + uri);
    }
    getContext().getContentResolver().notifyChange(uri, null);
    return Uri.parse(ProphetiesContract.BASE_PATH + "/" + id);
}
```

L'UriMatcher est indispensable !
getWritableDatabase() ouvre la base en lecture et écriture.
L'accès simultané sur une base SQLite n'est pas « sécure » s'il y a écriture de données. Donc le partage simultané d'un content provider par plusieurs activités doit être géré par exclusion mutuelle sur les méthodes CRUD.
NotifyChange() permet d'avertir des observers, s'ils sont enregistrés, d'un changement du content provider.
La méthode insert() retourne l'uri des données insérées.

```
public String getType(Uri uri) {
  switch(uriMatcher.match(uri)){
    case PROPHETIES : return "vnd.android.cursor.dir/"+ProphetiesContract.AUTHORITY;
    case PROPHETIE_ID : return "vnd.android.cursor.item/"+ProphetiesContract.AUTHORITY;
    default : throw new IllegalArgumentException("Unsupported URI : "+uri);
  }
}
```

Content Provider (12/16) : implémentation de query()

```
public Cursor query(Uri uri, String[] projection, String selection,
        String[] selectionArgs, String sortOrder) {
    SQLiteQueryBuilder queryBuilder = new SQLiteQueryBuilder();
    queryBuilder.setTables(Propheties.TABLE_NAME);
    int uriType = uriMatcher.match(uri);
    switch (uriType) {
    case PROPHETIES:
        break;
    case PROPHETIE_ID:
        queryBuilder.appendWhere(Propheties.COLUMN_NAME_PROPHETIE_ID + "="
                + uri.getLastPathSegment());
        break;
    default:
        throw new IllegalArgumentException("Unknown URI: " + uri);
    }
    SQLiteDatabase db = baseHelper.getReadableDatabase();
    Cursor cursor = queryBuilder.query(db, projection, selection,
            selectionArgs, null, null, sortOrder);
    if (cursor != null)
       cursor.setNotificationUri(getContext().getContentResolver(), uri);
    return cursor;
}
```

getReadableDatabase() ouvre la base en lecture seulement.
La méthode retourne un objet Cursor.

Content Provider (13/16) : implémentation de delete()

```
public int delete(Uri uri, String selection, String[] selectionArgs) {
    SQLiteDatabase db = baseHelper.getWritableDatabase();
    int count = -1;
    switch (uriMatcher.match(uri)) {
    case PROPHETIES:
        count = db.delete(Propheties.TABLE_NAME, selection, selectionArgs);
        break;
    case PROPHETIE_ID:
        String dbSelection = Propheties.COLUMN_NAME_PROPHETIE_ID + " = "
                           + uri.getPathSegments().get(1);
        if (selection != null)
           dbSelection += " AND " + selection;
        count = db.delete(Propheties.TABLE_NAME, dbSelection, selectionArgs);
        break;
    default:
        throw new IllegalArgumentException("Unknown URI: " + uri);
    }
    getContext().getContentResolver().notifyChange(uri, null);
    return count;
}
```

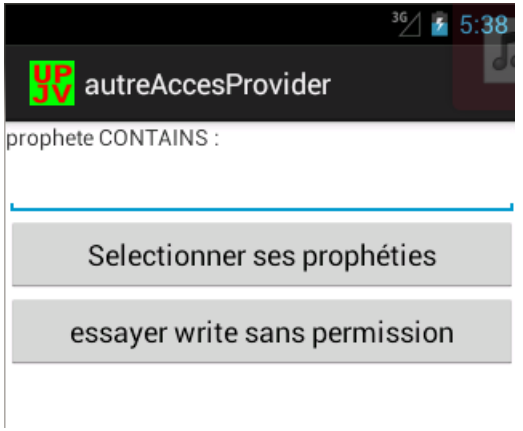
La méthode retourne le nombre de lignes supprimées.

Content Provider (14/16) : le manifeste

```
<permission android:name="df.cours.prophetie.provider.READ_PROPHETIE" />
<permission android:name="df.cours.prophetie.provider.WRITE_PROPHETIE" />

<application>
    <activity
        android:name="df.cours34.AccesProviderProphetieActivity" >
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
    <provider
        android:exported="true"
        android:name="df.cours34.ProviderPropheties"
        android:authorities="df.cours"
        android:readPermission="df.cours.prophetie.provider.READ_PROPHETIE"
        android:writePermission="df.cours.prophetie.provider.WRITE_PROPHETIE"
        >
    </provider>
</application>
```

Le manifeste déclare le content provider :
- son ou ses authorities
- name désigne l'activité qui l'implémente
- exported true autorise d'autres activités à accéder à ce content provider (sous réserve des droits s'il y a)
- `ReadPermission` permission requise pour lire des données du provider et réciproquement pour écrire, le nom de la permission doit respecter le « java package style »

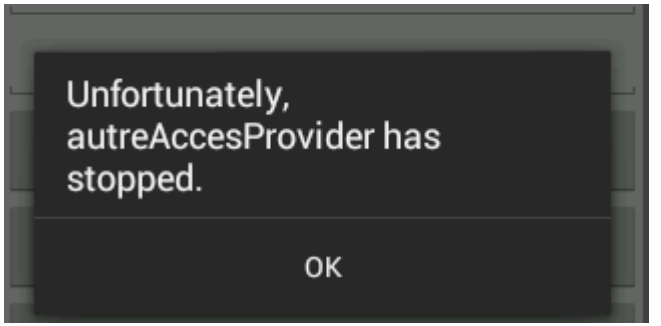
autre accès au Content Provider (15/16) :




```
package df.cours35;
...
public class AutreAccesProviderActivity extends Activity {
    public static final class ProphetiesContract {
        public static final String AUTHORITY = "df.cours";
        ...
    }
    ...
    public static Uri CONTENT_URI = Uri.parse("content://df.cours/propheties");
    ...
    public void selectionner(View v) {
        String[] projection = {  ProphetiesContract.PROPHETIE_ID,
                                 ProphetiesContract.PROPHETE,
                                 ProphetiesContract.TEXTE };
        String [] selectionArgs = { "%" + contains + "%" };
            c = getContentResolver().query(ProphetiesContract.CONTENT_URI , projection,
                    ProphetiesContract.PROPHETE + " LIKE ?" , selectionArgs, null);
        if (c != null) {
            while (c.moveToNext())
            result.append(c.getString(c.getColumnIndex(ProphetiesContract.PROPHETIE_ID))
                        +" "+....
```

Accès possible au Content provider car il est « exported ».
Utilisation d'une Contract Class pour simplifier la programmation.
La lecture utilise içi une clause « where » et ses arguments.

autre accès au Content Provider (16/16) :




```
...
    public void write(View view) {
        Uri uri = ContentUris.withAppendedId(ProphetiesContract.CONTENT_URI, 1);
        getContentResolver().delete(uri,  null, null);
    }
```

Le manifeste posséde la permission read mais pas la permission write :

```
manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="df.cours35"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="18" />
    <uses-permission android:name="df.cours.prophetie.provider.READ_PROPHETIE"/>

    <application >
        <activity
            android:name="df.cours35.AutreAccesProviderActivity"
            android:label="@string/app_name" >
...
```