Android View, onClick,Activity, Modèle Vue Contrôleur

jean-michel Douin, douin au cnam point fr
version : 24 Octobre 2011

Notes de cours

Bibliographie utilisée

http://developer.android.com/resources/index.html

…

Android : Développer des applications mobiles pour les Google Phones, de Florent Garin, chez Dunod

Le cours de Victor Matos
http://grail.cba.csuohio.edu/~matos/notes/cis-493/Android-Syllabus.pdf

Android A Programmers Guide - McGraw Hill
Professional Android Application Development – Wrox

http://marakana.com/bookshelf/main_building_blocks_tutorial/table_of_contents.html

Pré-requis, Sommaire

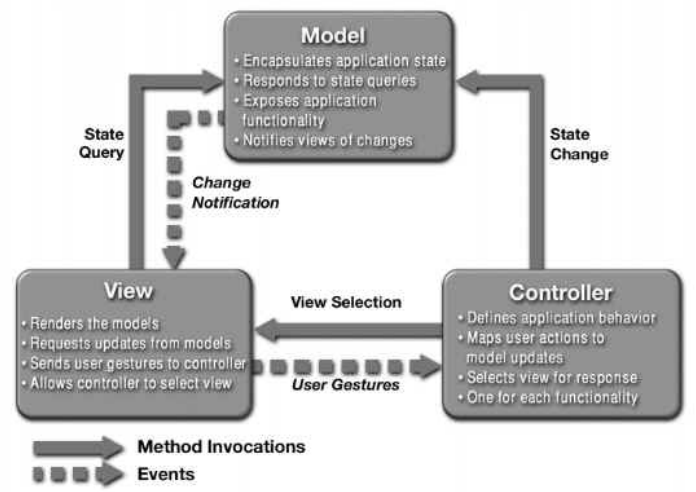
- **Pré requis indispensable**
  - Avoir réalisé le tp mvc,
  - Un tp utilisant une calculette à pile

- **Au sommaire**
  - Comment assurer un couplage faible des classes
  - Observable/Observateur
  - IHM/Vue et Listener/Contrôleur

Pré requis, rappel

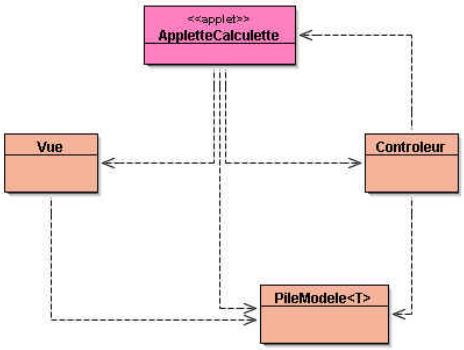
- **Pré requis**
  - TP Calculette à pile déjà réalisé J2SE
    - Thème : Modèle Vue Contrôleur
    - http://jfod.cnam.fr/eicnam/tp_mvc/tp_mvc.html

Pré requis, MVC



- http://java.sun.com/blueprints/guidelines/designing_enterprise_applications/introduction/summary/index.html

Pré requis, l'architecture retenue pour le TP

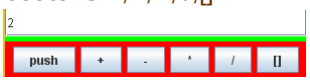


- Le Modèle est une pile (classe **PileModele<T>**).
- La Vue correspond à l'affichage de l'état de la pile (classe **Vue**).
- Le Contrôleur gère les évènements issus des boutons +, -, *, /,[] (classe **Controleur**).
  - L'applette crée, assemble le modèle, la vue et le contrôle (classe **AppletteCalculette**).

Ce découpage engendre bien des discussions

- Le Modèle est ici une pile (classe **PileModele<T>**).
- La Vue correspond à l'affichage de l'état de la pile (classe **Vue**).

- Le Contrôleur gère les évènements issus des boutons +, -, *, /,[]

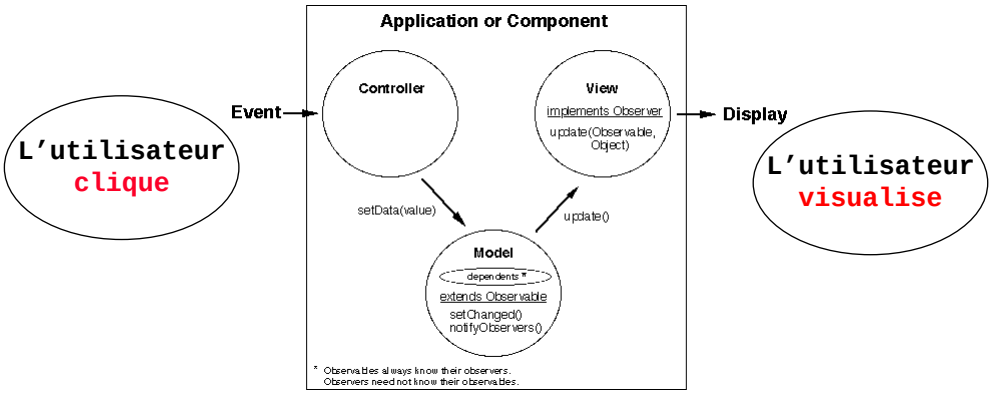


  - L'applette crée, assemble le modèle, la vue et le contrôle (classe **AppletteCalculette**).

Discussions ...

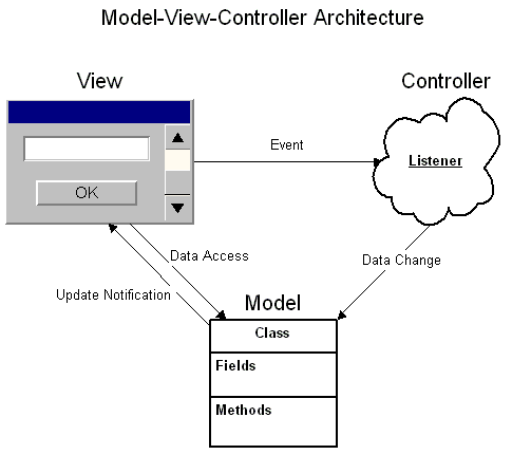
- *Le modèle pourrait être la calculette constituée pour ses calculs internes d'une pile,*
- *Pourquoi les "listeners" des boutons sont-ils locaux au contrôleur ?*
- *Pourquoi un JPanel pour le contrôleur ?*
- *Ce choix de découpage MVC vous parait-il réaliste ?*
  - ***Discussion, blabla, blabla, blabla***

Architecture classique … valeur sûre



- **Le Modèle hérite de Observable**
- **La Vue implemente Observer**

MVC encore



- **Model extends Observable**
- **View implements Observer**
- **Controller implements XXXXListener, YYYYListener**

Nouvelle architecture

En conséquence

Au tp

- Le Modèle est une pile (classe **PileModele<T>**).
- La Vue correspond à l'affichage de l'état de la pile (classe **Vue**).
- Le Contrôleur gère les évènements issus des boutons +, -, *, /,[].

Architecture retenue

- Le Modèle est une calculette
- La Vue correspond à l'IHM (au complet).
- Le Contrôleur gère les évènements issus des boutons +, -, *, /,[]

Architecture retenue

- **Le Modèle**
  - **La calculette munie de ses opérations (+,-,/,*,…)**
    - **Hérite de la classe java.util.Observable**
    - **Les sources du modèle sont ici**
      - **http://douin.free.fr/tp4Calculette/**
- **La Vue**
  - **L'IHM affichage, zone de saisie, boutons …**
    - **Implémente java.util.Observer**
- **Le Contrôleur**
  - **Réalisation, implémentation des listeners, (le comportement de l'IHM)**
    - **Implémente plusieurs ActionListener**

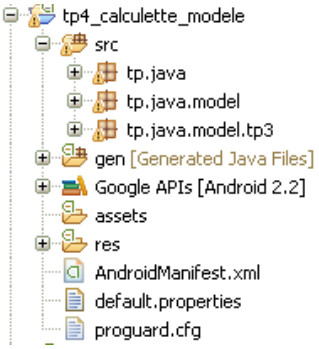
-> pour Android, quel découpage ?, quelles classes ?

Android, la classe Activity

- **Activité comme application élémentaire**
  - **À cette activité lui correspond une IHM, ce que l'on voit …**

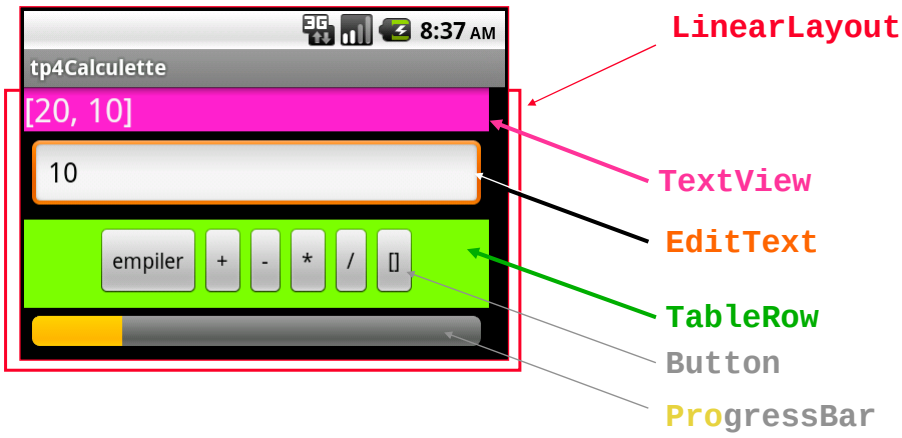
    public class Calculette **extends** Activity {
  - **Cette IHM est décrite par un fichier XML (la vue)**
  - **L'activité réagit aux sollicitations de l'utilisateur (le contrôleur)**
  - **Le modèle ne change pas**

Android, la calculette



- **Android : Démonstration …**

IHM : Layout, View , Button…



- **Description de cette interface en XML**
  - **Fichier res/layout/main.xml**

Interface, IHM : Approche déclarative ./res/

- **Chaque composant possède un id (android:id= "@+id/push")**

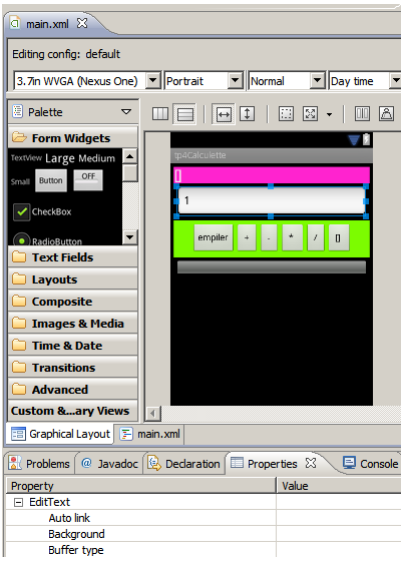
IHM, un outil de conception sous eclipse

- **Item comme classe**
  - Nom de la balise
  - Nom de la classe
- **Properties de chaque item**
  - Comme Attribut XML, cf. Properties
  - Mais aussi comme Attribut de la classe,
- **<EditText**

```
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:enabled="true"
android:visibility="visible"
android:clickable="false"
android:layout_margin="5dip"
android:numeric= "decimal"
android:inputType= "number"

android:id="@+id/donnee"
android:text="1" />
```

En Java

```
EditText donnee = (EditText) findViewById(R.id.donnee);
String txt = donnee.getText().toString();
```



Intégration de l'IHM, R.layout.main

- **Au sein d'une Activity**
  - **XML : accès en java via R**
    - **Les ressources XML sont accessibles via le fichier R**
    - **R.java est généré par Android**
    - **Convention : /layout/main -> R.layout.main, R.id. R.string. …**

      » R cf. dossier /gen/

  - **Alors dans une Activity,**
    - **Avec au préalable l'affectation de l'interface par l'appel de**
      - **setContentView(R.layout.main);**
  - **Les composants de l'IHM deviennent accessibles**
    - **Button empiler = (Button) findViewById(R.id.push);**
      - **findViewById est une méthode héritée de la classe Activity**

Une Activity, démarrage par onCreate

```
public class TPCalculette extends Activity {


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


      // Affectation de l'IHM issue des fichiers XML
       setContentView(R.layout.main);
```


Une Activity accès aux composants

```
public class TPCalculette extends Activity {

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

// accès aux composants de l'IHM
    Button empiler = (Button) findViewById(R.id.push);
    ProgressBar jauge = (ProgressBar) findViewById(R.id.jauge);

// accès aux chaînes de caractères ( plusieurs langues)
   String str = getString(R.string.app_name);
```

Comportement : listener et plus

- **Un seul « Listener » par composant**

- **Button empiler = (Button) findViewById(*R.id.push*);**

```
empiler.setOnClickListener(new View.OnClickListener(){
        public void onClick(View v){
            // traitement associé
        }
}
```


Ou bien usage de l'Attribut onClick

- **Extrait de layout/main.xml**
- <Button android:layout_height="wrap_content" android:id="@+id/push"
- android:layout_width="wrap_content" android:text="@string/push"
- **android:onClick="onClickEmpiler">**
- </Button>

- **Dans la classe de l'activity**

```
public void onClickEmpiler(View v){
        // traitement associé
}
```

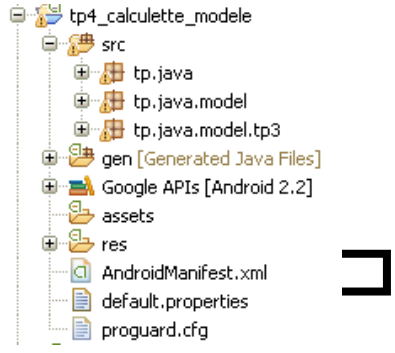
Démonstration

Une démonstration

- Avec au moins un bouton … de bienvenue
  - À chaque clic un Toast est affiché

Android : la description de l'application

- **Description de l'application**
  - Quelles activités ?
    - Plusieurs activités pour une application
  - Quelles permissions ?
    - SMS, Réseau, GPS, …
  - Quelles librairies ?



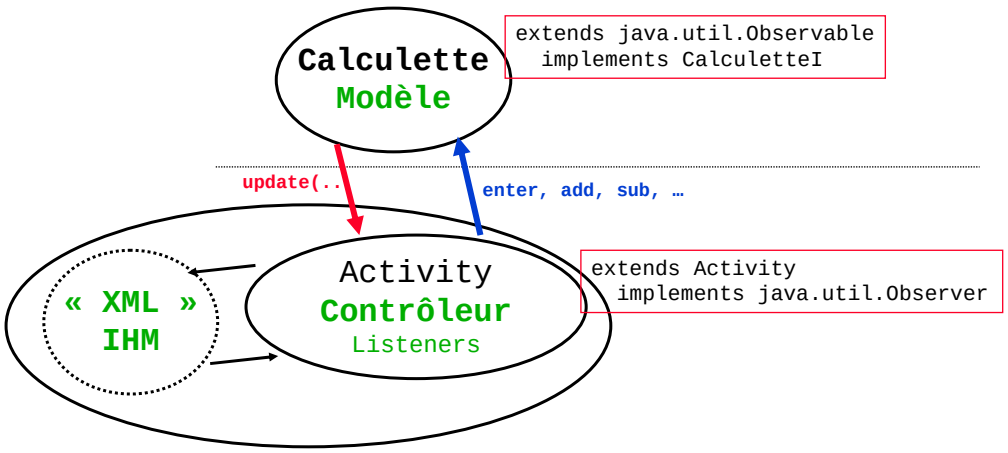
- **Dans un fichier XML**
  - AndroidManifest.xml

AndroidManifest.xml, tp4Calculette

- **La description de l'application,**
  - **destinée à l'hébergeur**

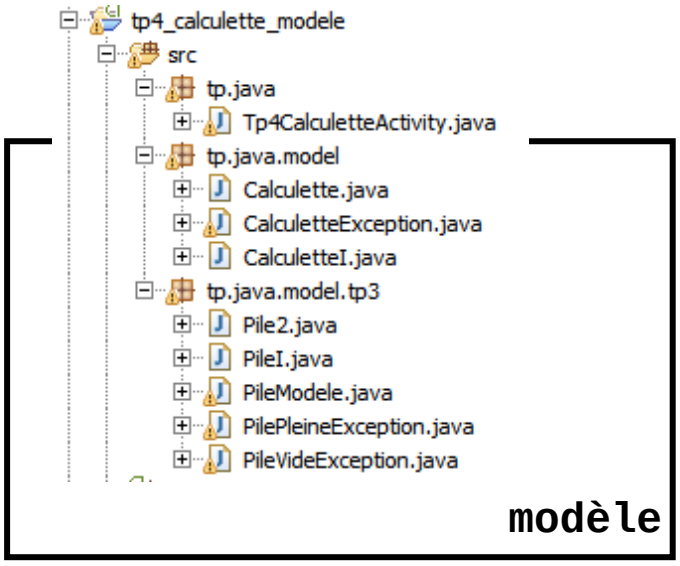
```
<?xml version="1.0" encoding="utf-8" ?>
- <manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="tp.java" android:versionCode="1" android:versionName="1.0">
    <uses-sdk android:minSdkVersion="8" />

  - <application android:icon="@drawable/icon_calculator"
      android:label="@string/app_name">
      <uses-library android:name="android.test.runner" />
    - <activity android:name=".Tp4CalculetteActivity"
        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>
```


MVC, Mise en Pratique : discussions



- **L'activity Android est une vue du Modèle Calculette** `(implements Observer)`
- **L'activity Android est le contrôleur de l'IHM décrite en XML (**`extends Activity)`

packages



Le Modèle : la Calculette

```
public interface CalculetteI {
 // operations
 void enter(int i) throws CalculetteException;

 void add() throws CalculetteException;
 void sub() throws CalculetteException;
 void div() throws CalculetteException;
 void mul() throws CalculetteException;

 void clear();
 int pop()  throws CalculetteException;

 // interrogations
 int result() throws CalculetteException;
 boolean isEmpty();
 boolean isFull();
 int size();
 int capacity();
}
```

```
public class Calculette
   extends java.util.Observable
      implements CalculetteI
```

TP4CalculetteActivity,
la vue du modèle + le Contrôleur de l'IHM

```
public class Tp4CalculetteActivity extends Activity implements Observer{
   private Calculette calculette;

   public void onCreate(Bundle savedInstanceState) {   // appelée par Android
      super.onCreate(savedInstanceState);
      this.calculette = new Calculette();                    // une calculette est créée
      this.calculette.addObserver(this);                 // c'est une vue du modèle calculette
      setContentView(R.layout.main);                      // l'IHM est associée à cette activité
      ….
    }

…
public void onClickEmpiler(View v){                          // attribut onClick balise <Button
}

…
public void update(Observable arg0, Object arg1) {                  // à chaque notification
}
```


TP4CalculetteActivity,
le Contrôleur de l'IHM

```
public class Tp4CalculetteActivity extends Activity implements Observer{
   private Calculette calculette;

   public void onCreate(Bundle savedInstanceState) {                  // appelée par Android
      …
      ….
    }

…
 public void onClickEmpiler(View v){              // attribut onClick balise <Button
   try{
      int operande = …
      this.calculette.empiler(operande);              // opération empiler sur le modèle
     }catch(CalculetteException nfe){}
   }

  public void onClickAdd(View v){
…
public void update(Observable arg0, Object arg1) {                  // à chaque notification
}
```

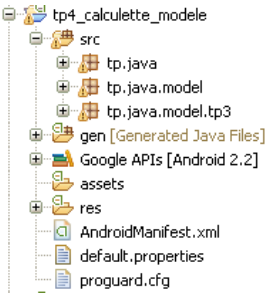
TP4CalculetteActivity, la vue du modèle

```
public class Tp4CalculetteActivity extends Activity implements Observer{
   private Calculette calculette;

  public void onCreate(Bundle savedInstanceState) {                    // appelée par Android
....
   }

...
 public void onClickEmpiler(View v){                                          // attribut onClick
}

...

public void update(Observable arg0, Object arg1) {      // à chaque notification du
   TextView etat = (TextView) findViewById(R.id.etatPile);                  // modèle
   etat.setText(calculette.toString());
   ProgressBar jauge = (ProgressBar) findViewById(R.id.jauge);
   jauge.setProgress(calculette.size());
   actualiserInterface();
}
```


Architecture retenue

- **Application Calculette en résumé**
  - **Observable : Le modèle, la calculette**
  - **Observer : l'Activity, mise à jour de l'interface**
  - **View : le fichier XML (l'IHM)**
  - **Controller : l'Activity**

Android, la calculette



- **Réalisation …**

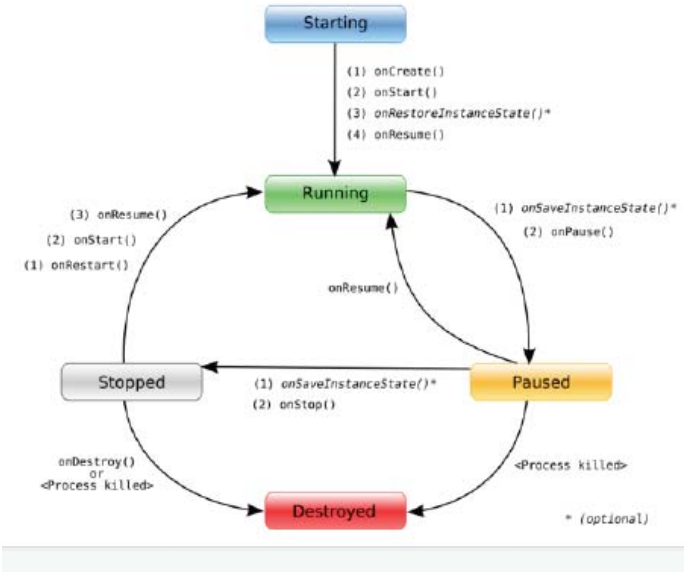
Comportement attendu, à vérifier

- **3 + 2 == 5 ?**
- **Appui sur la touche « retour »**
  - **Fin de l'activité**

- **Appui sur la touche « HOME »**
  - **L'activité est en Pause …**

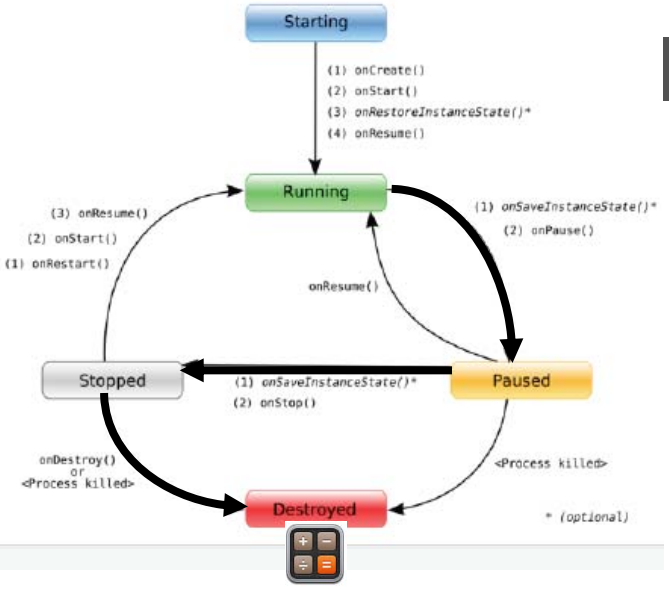
- **Un appel téléphonique**
  - **telnet localhost 5554**
  - **gsm call 5554**
- **Rotation de l'écran**
  - http://developer.android.com/guide/developing/tools/emulator.html
    - Ctrl-F11, Ctrl-F12
  - Que se passe-t-il ?

Cycle de vie d'une activity, première approche



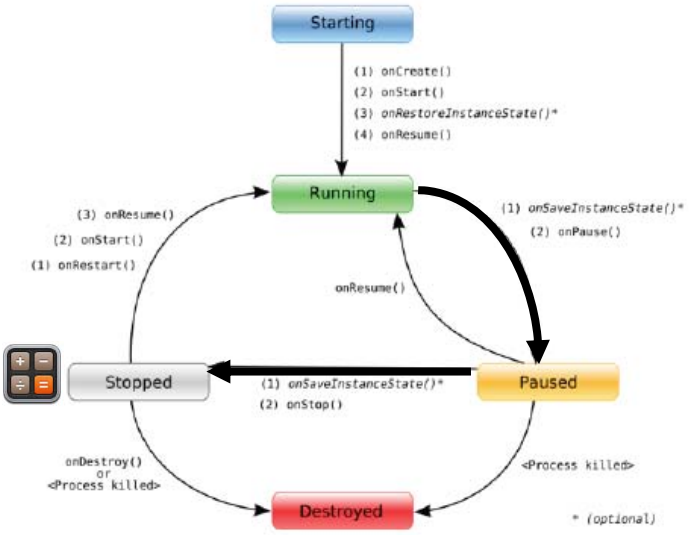
- ....

Cycle de vie d'une activity



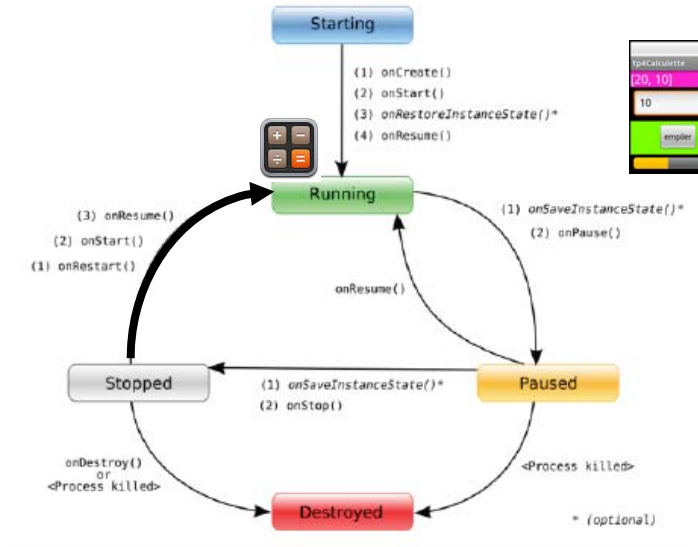
- ....

Cycle de vie d'une activity



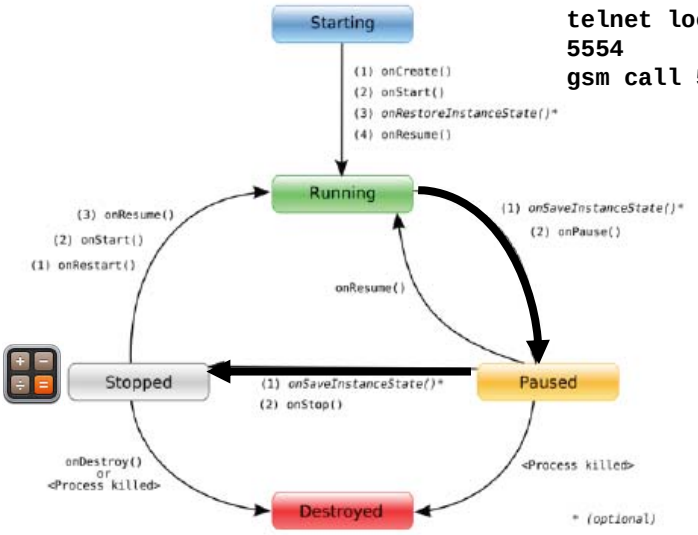
- ....

Cycle de vie d'une activity, re-sélection



- ....

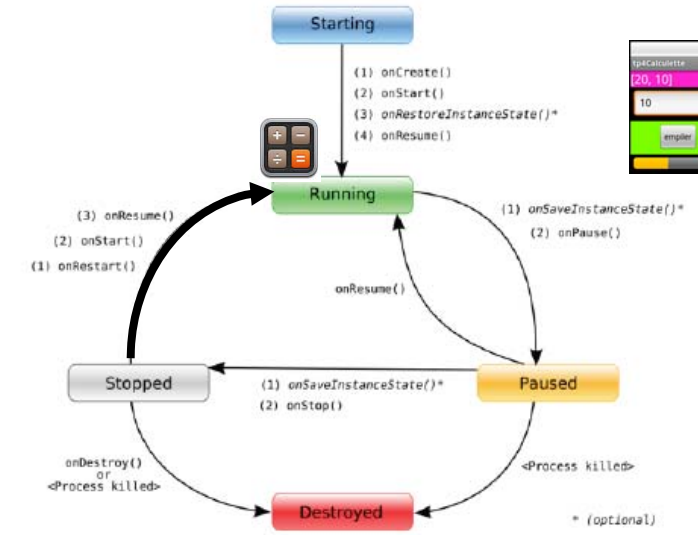
Cycle de vie d'une activity, un appel




```
telnet localhost
5554
gsm call 5554
```

- ....

Cycle de vie d'une activity, fin d'appel

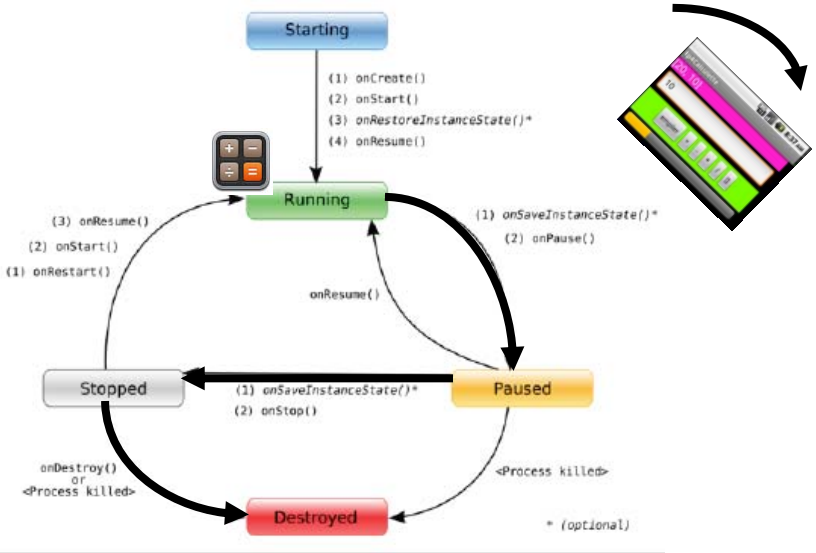


- ....

Illustration du cycle de vie

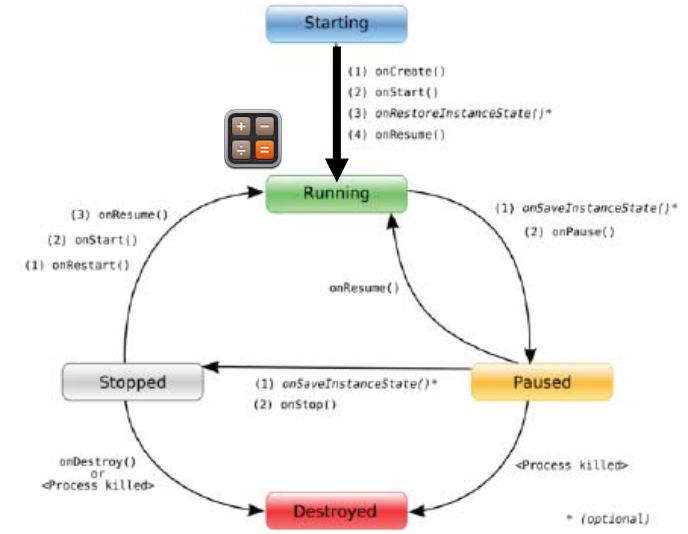
- **Illustration du cycle de vie**
  - **Démonstration**
- **Touche « Retour »**
  - **onPause, onDestroy ….ok**
- **Touche Menu**
  - **onSaveInstanceState, onPause, onStop ...ok**
  - **Sauvegarde par défaut …**
- **Un appel idem**
- **Rotation de l'écran** Ctrl-F11, Ctrl-F12
  - **… !!!**
    - **Android détruit votre vue en cas de rotation pour construire la nouvelle vue**

Cycle de vie d'une activity, onSaveInstanceState



- ….

Cycle de vie d'une activity, onRestoreInstanceState



- ....

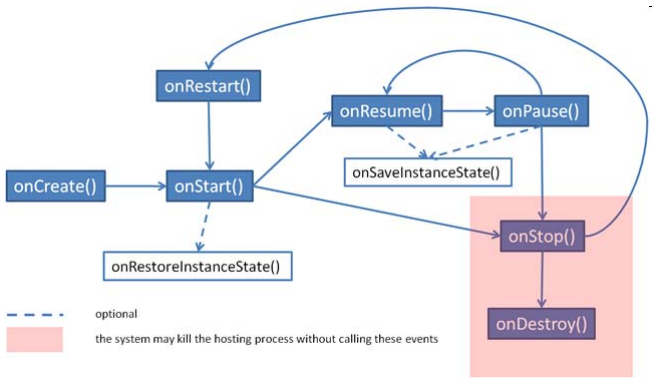
Mise en Pratique, suite

– **Suite du TP :**
  - **Sauvegarde de l'état de la calculette**
  - **Cf. cycle de vie**

```
protected void onSaveInstanceState(Bundle out){
    out.putInt("taille",calculette.size());
    …

protected void onRestoreInstanceState(Bundle in){
    int taille = in.getInt("taille");
    …
```

– **Sauvegarde et restitution via le bundle**
  - **http://developer.android.com/reference/android/os/Bundle.html**

Cycle de vie d'une activity …

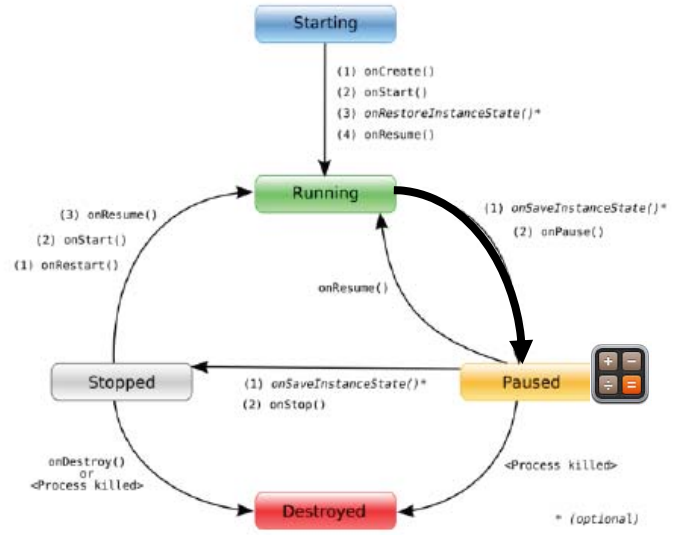


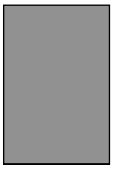
- http://www.itcsolutions.eu/wp-content/uploads/2011/08/Android_Activity_Events-Copy.png

Illustration du cycle de vie

- **Ecran de veille alors que la calculette était au premier plan**
  - **onPause**
  - **Noté semi-visible…**

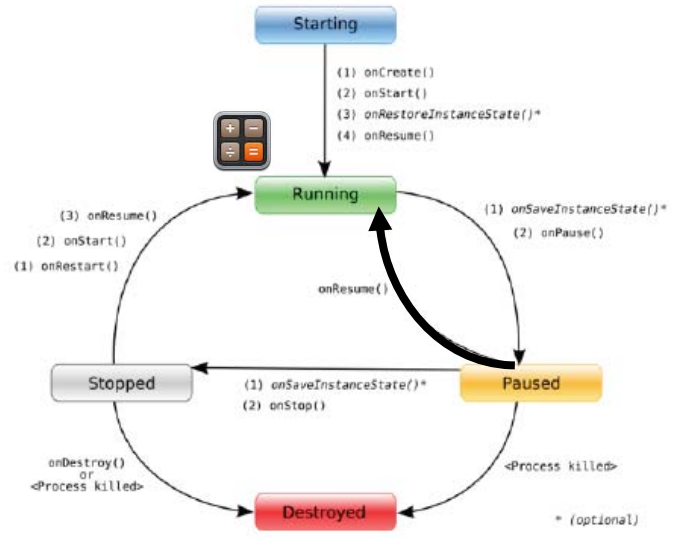
Cycle de vie d'une activity, écran de veille





- ....

Cycle de vie d'une activity, au réveil



- ....

Extension possible : à chaque exception un sms !

- **Un mauvais format de Nombre → envoi d'un SMS !**
  - **Propagation de l'exception jusqu'à votre mobile …**
  - **1) au sein de votre application**
  - **2) appel de l'activity prédéfinie**

Au sein de votre application

```
private void sendSMS(String msg){
     try{
       SmsManager sm = SmsManager.getDefault();
       String body = getString(R.string.app_name) + " : " + msg + "\n";

       sm.sendTextMessage(getString(R.string.numero_tel), null, body,
                          null, null);

// ou bien un Toast
Toast.makeText(getBaseContext(),
 " envoi d'un sms " + msg, Toast.LENGTH_LONG).show();
     }catch(Exception e){
       Toast.makeText(getBaseContext(), getString(R.string.erreur),
  Toast.LENGTH_LONG).show();
     }
   }
```

- Mais avez-vous la permission ?, -> AndroidManifest

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

Une Activity en démarre une autre

- Intent

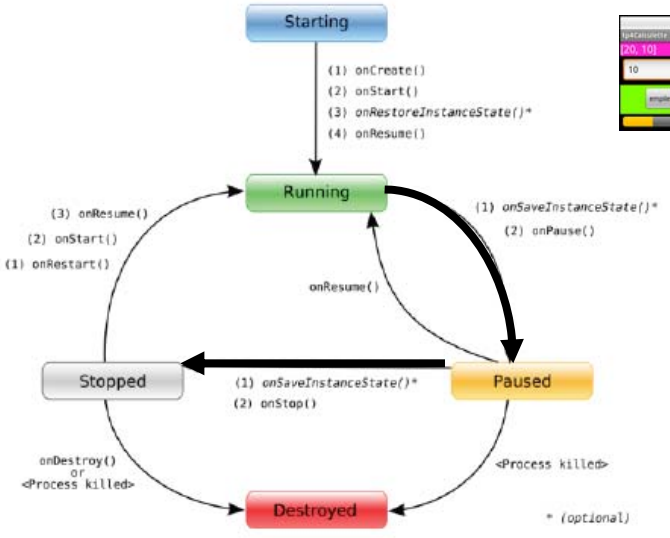
ou comment transmettre des paramètres à une activité

- Intent sendIntent = new Intent(Intent.ACTION_VIEW);
- sendIntent.putExtra("sms_body", "The SMS text");
- sendIntent.setType("vnd.android-dir/mms-sms");

- startActivity(sendIntent);

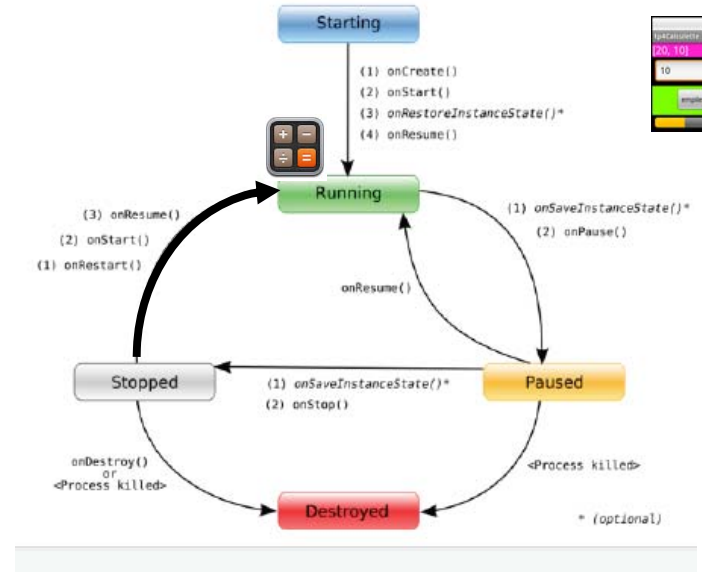
- **Cycle de vie**

Cycle de vie d'une activity, **startActivity(sendIntent);**



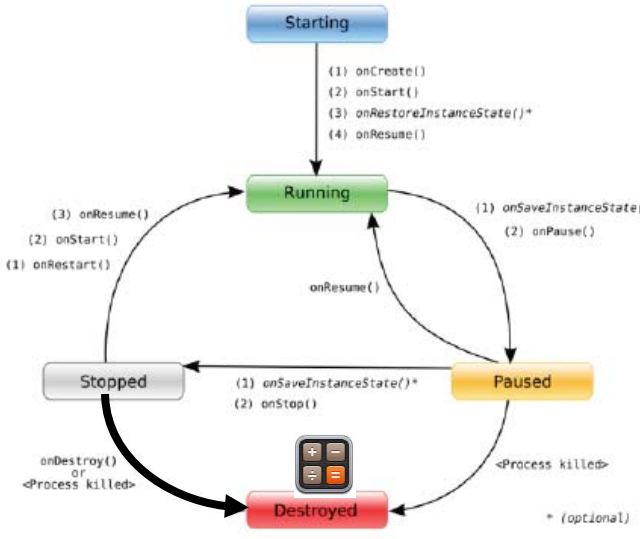
- ....

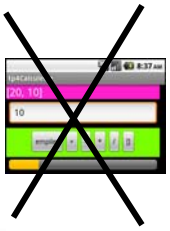
Cycle de vie d'une activity, **sms envoyé +**



- ....

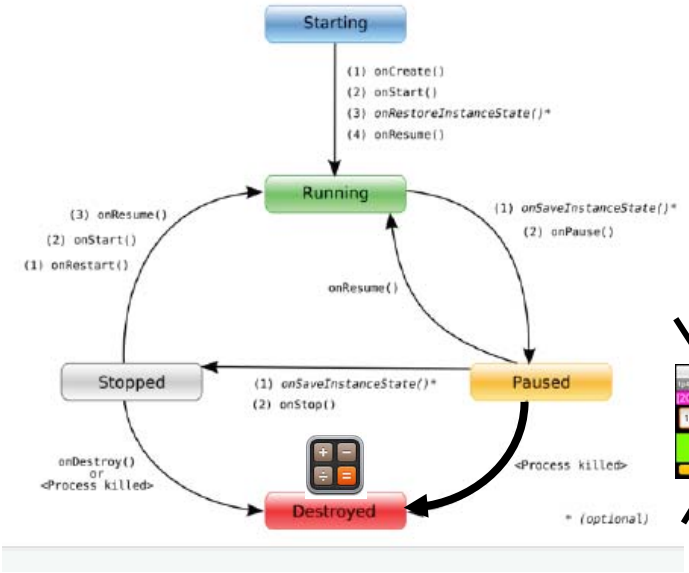
Cycle de vie d'une activity, **android killer**

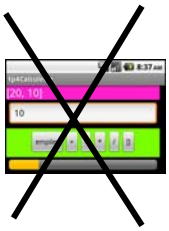


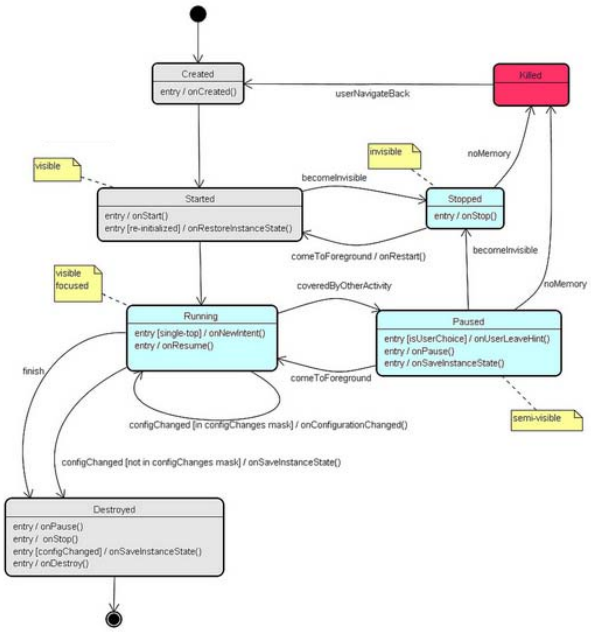


- ....

Cycle de vie d'une activity, **android killer**



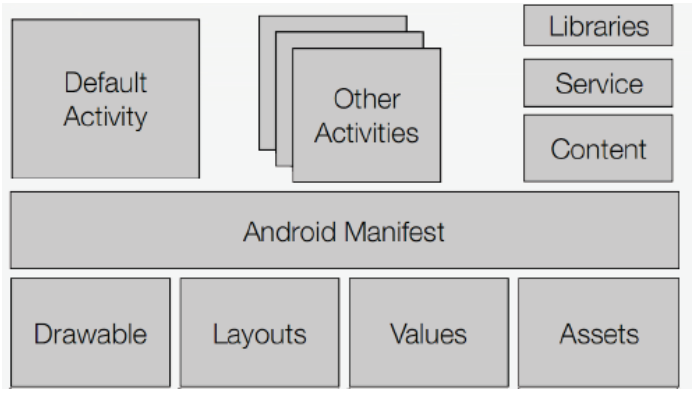




- http://good-good-study.appspot.com/blog/posts/103001

L'architecture se précise

- **Une Activity peut en déclencher une autre**
  - **A chaque activity son écran (son fichier XML)**
  - **Nécessaire gestion de ces activités, android utilise une pile**

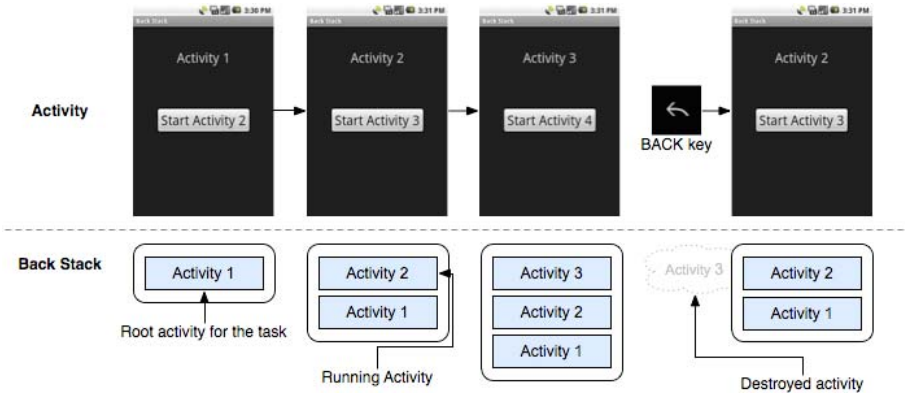


Extension possible : à chaque exception un sms !

- **Un mauvais format de Nombre → Contrôle de la saisie**

```
android:numeric="decimal"
android:inputType="number"
```

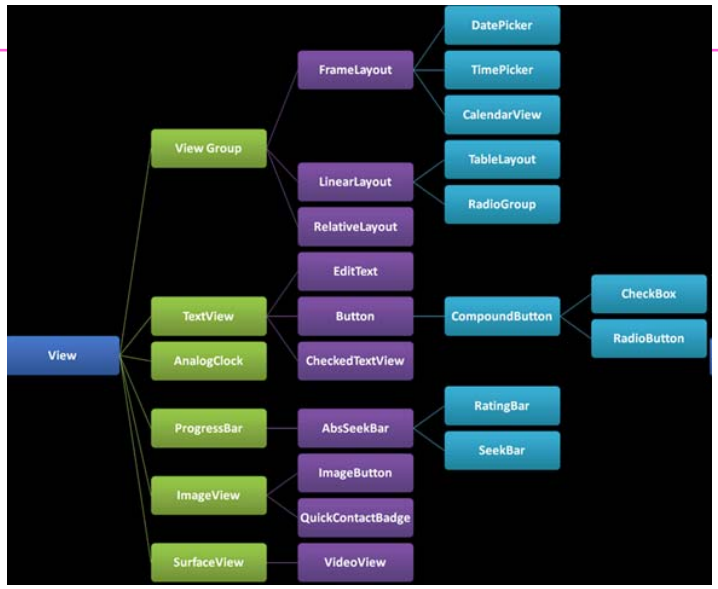
La pile des Activity



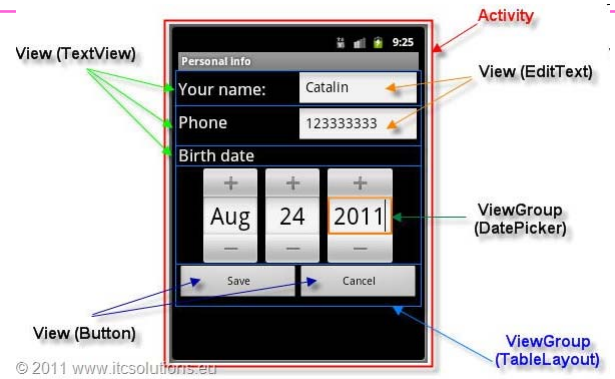
- http://www.vineetgupta.com/2011/03/mobile-platforms-part-1-android/
- http://developer.android.com/guide/topics/fundamentals/tasks-and-back-stack.html

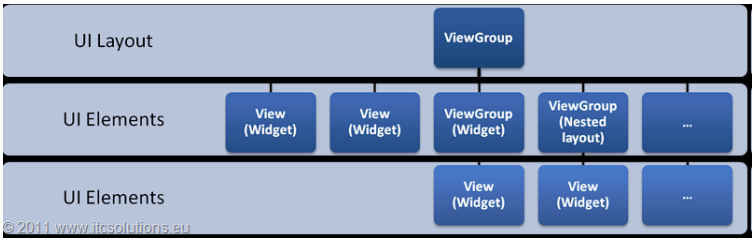
Annexe : un tutorial

- **http://www.itcsolutions.eu/2011/08/26/android-tutorial-overview-and-contents/**
- **http://www.itcsolutions.eu/2011/08/27/android-tutorial-4-procedural-vs-declarative-design-of-user-interfaces/**

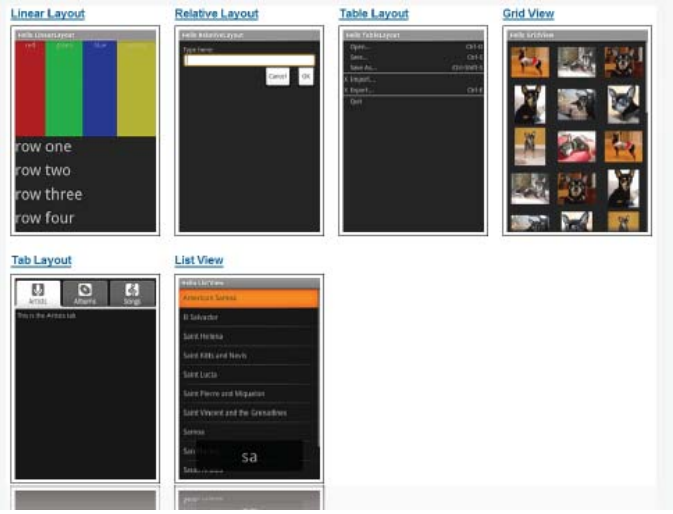


- http://www.itcsolutions.eu/2011/08/27/android-tutorial-4-procedural-vs-declarative-design-of-user-interfaces/





Vocable



Vocable