
Android View, onClick, Activity, Modèle Vue Contrôleur

jean-michel Douin, douin au cnam point fr
version : 26 Septembre 2012

Notes de cours

Android_MVC

1

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

Android_MVC

2

Avertissement, pré-requis et 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/View et Listener/Contrôleur

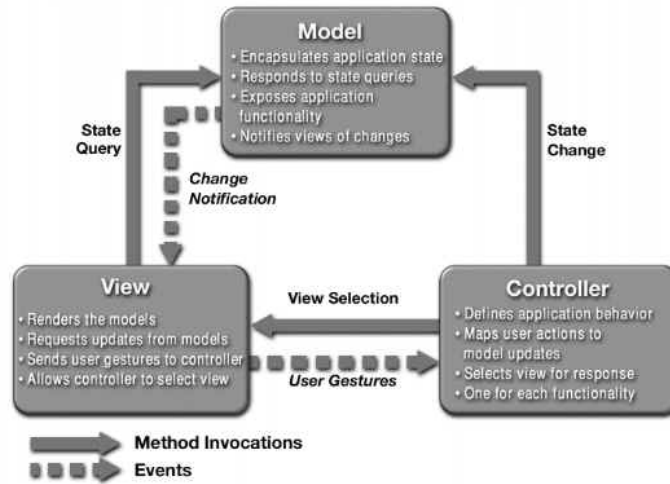
Pré requis, rappel

- **Pré requis**

- TP Calculette à pile déjà réalisé J2SE
 - Thème : Modèle Vue Contrôleur
 - http://jfod.cnam.fr/eicnam/tp_mvc/tp_mvc.html
 - Usage de l'applette à cette URL est conseillée
- >appletviewer http://jfod.cnam.fr/eicnam/tp_mvc/tp_mvc.html



Pré requis, MVC

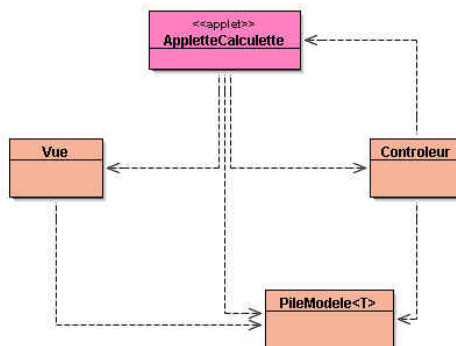


- http://java.sun.com/blueprints/guidelines/designing_enterprise_applications/introduction/summary/index.html

Android_MVC

5

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 événements issus des boutons +, -, *, /, [] (classe **Controleur**).
 - L'applette crée, assemble le modèle, la vue et le contrôle (classe **AppletteCalculette**).

Android_MVC

6

Cette architecture engendre 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**).



Android_MVC

7

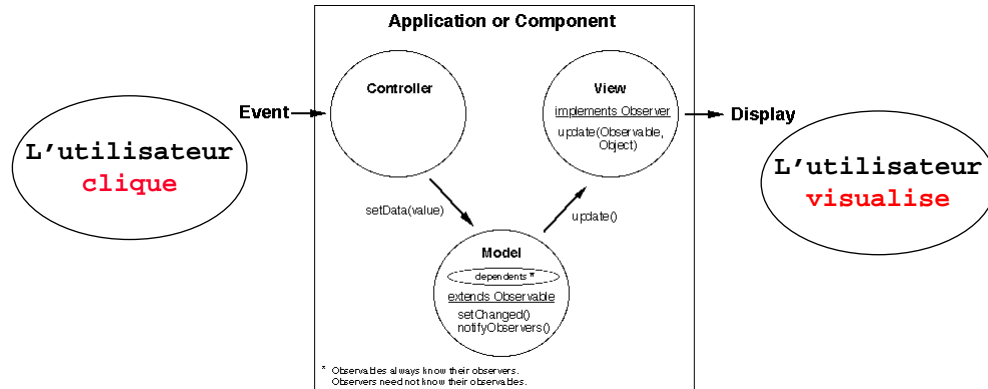
Discussions ... entre nous

- *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 paraît-il réaliste ?*
 - *Discussion, blabla, blabla, blabla*

Android_MVC

8

Architecture classique ... une valeur sûre

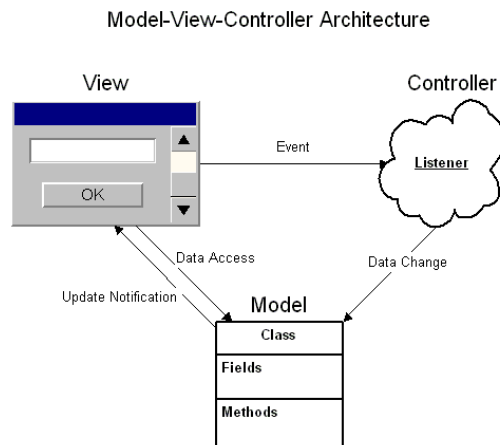


- Ici le Modèle hérite de `java.util.Observable`
- La Vue implémente `java.util.Observer`

Android_MVC

9

MVC encore



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

Android_MVC

10

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 événements issus des boutons +, -, *, /, [].

Architecture retenue

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

Architecture retenue

• Le Modèle

- La calculatrice munie de ses opérations (+,-,/,*,...)
 - Hérite de la classe `java.util.Observable`
 - Les sources du modèle sont ici
 - <http://douin.free.fr/tp4Calculatrice/>

• 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)

Android_MVC

13

L'IHM de la calculette

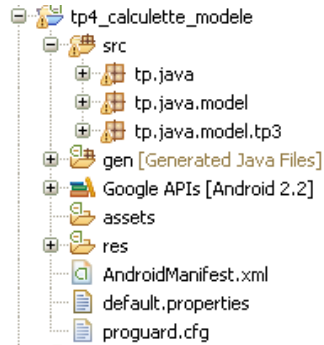


- Un fichier XML décrit complètement cette interface
- L'activité Calculette affiche, présente cette interface

Android_MVC

14

Android, la calculette



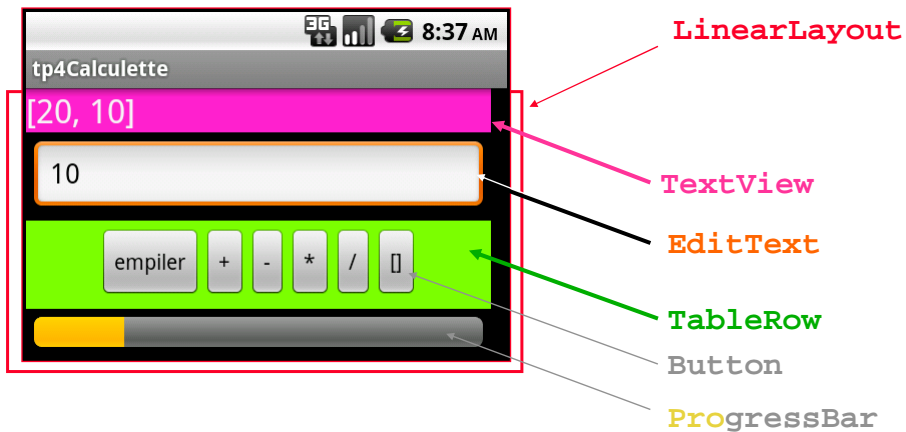
- L'activité affiche l'IHM
- Le modèle ne change pas
 - Cf. le TP

- Android : Démonstration ...

Android_MVC

15

IHM : Layout, View, Button...

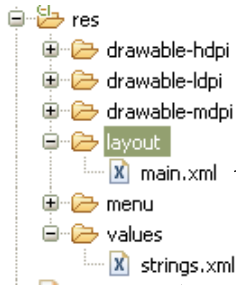


- Description de cette interface en XML
 - Fichier `res/layout/main.xml`

Android_MVC

16

Interface, IHM : Approche déclarative ./res/



```

20 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     android:orientation="vertical" android:layout_width="fill_parent"
4     android:layout_height="fill_parent" android:weightSum="1" android:layout_
50 <TextView android:textAppearance="?android:attr/textAppearanceLarge"
6     android:layout_width="match_parent" android:layout_height="wrap_conte
7     android:clickable="false" android:id="@+id/etatFile" android:text="@s
8     android:background="#ff23c1" android:textColor="#f0f0f0"></TextView>
9
10 <EditText android:layout_width="match_parent"
11     android:layout_height="wrap_content" android:enabled="true"
12     android:visibility="visible" android:clickable="false" android:id="@+
13     android:text="10" android:layout_margin="5dip">
14 </EditText>
15 <TableRow android:id="@+id/tableRow1" android:layout_width="match parent"
16     android:layout_height="wrap_content" android:background="#7cf000" and
17 <Button android:layout_height="wrap_content" android:id="@+id/push" and
18     android:layout_width="wrap_content" android:text="@string/push"
19     android:onClick="onClickPush"></Button>
20 <Button android:layout height="wrap content"
  
```

```

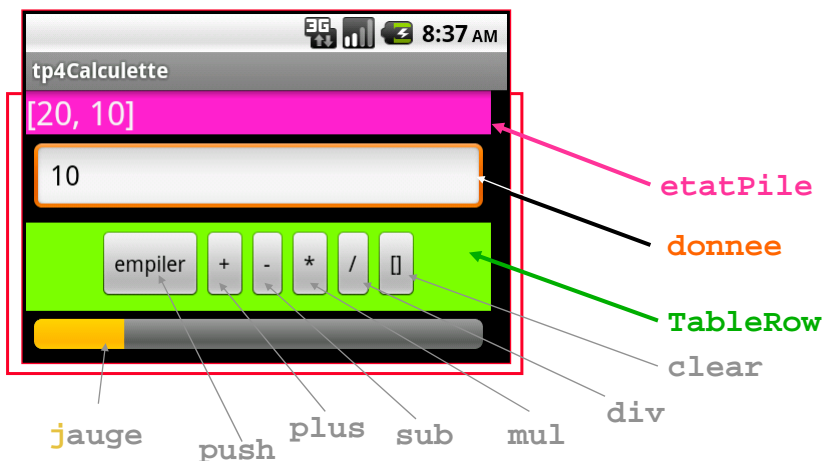
1 <?xml version="1.0" encoding="utf-8" ?>
2 <resources>
3     <string name="hello">Tp4CalculletteAct
4     <string name="app_name">tp4Calcullette
5
6     <string name="vide">[]</string>
7     <string name="push">empiler</string>
8     <string name="add"> + </string>
9     <string name="diy"> / </string>
10    <string name="sub"> - </string>
  
```

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

Android_MVC

17

Chaque composant a son id



- <Button android:id= "@+id/push"

Android_MVC

18

IHM, un outil de conception sous eclipse

- **Item comme classe**

- **Nom de la balise**

comme

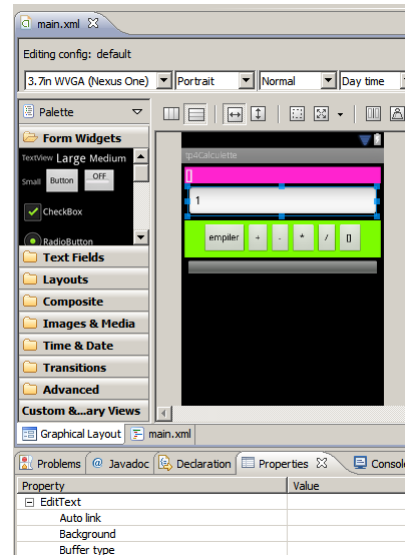
- **Nom de la classe**

- **Properties de chaque item**

- **Attribut XML, cf. Properties**

comme

- **Mais aussi comme Attribut de la classe,**



Android_MVC

19

Adéquation XML <-> java, le fichier R

- En XML
- **<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" />
```

...

```
<Button android:id="@+id/push"
```

En Java

```
EditText donnee = (EditText) findViewById(R.id.donnee);
Button empiler = (Button) findViewById(R.id.push);
```

Android_MVC

20

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/

Intégration de l'IHM, R.layout.main

- **Au sein d'une Activity**
 - 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
 - Le source java ne manquera pas de R, ni d'appels de R (facile ...)

Une Activity, « démarrage » par onCreate

```
public class TPCalculatrice extends Activity {  
  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
  
        // Affectation de l'IHM issue des fichiers XML  
        setContentView(R.layout.main);  
    }  
}
```

Android_MVC

23

Une Activity accède aux composants

```
public class TPCalculatrice 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);  
    }  
}
```

Android_MVC

24

Comportement : OnClickListener 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

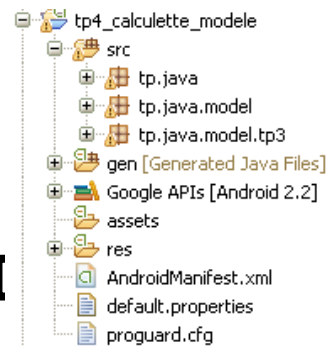
Une démonstration

- Avec au moins un bouton ... de bienvenue
 - À chaque clic la chaîne hello world est modifiée

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, tp4Calcullette

- **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=".Tp4CalculletteActivity"
      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>
```

Android_MVC

29

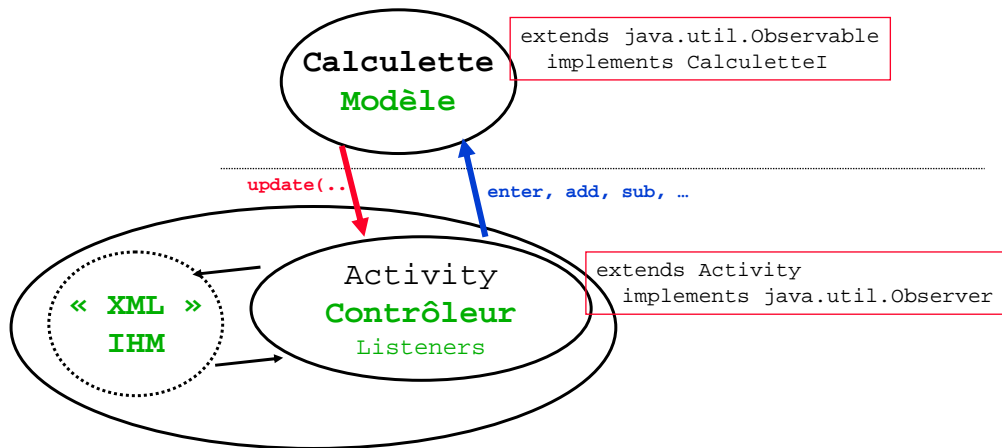
MVC et Android

- **Nous avons :**
 - Une IHM décrite en XML
 - Une activité qui implémente le comportement
- **Alors**
 - L'activité est le contrôleur de la vue, de l'IHM décrite en XML
 - L'activité met l'IHM au premier plan,
 - l'activité a accès aux composants graphiques
- **MVC ?**
 - Le modèle est une calcullette
 - La vue du modèle est l'activité

Android_MVC

30

MVC, Mise en Pratique : discussions

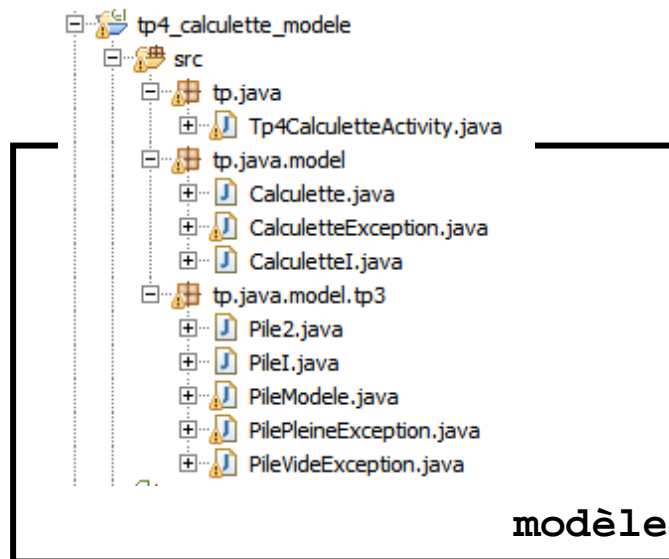


- L'activity Android est une vue du Modèle Calcullette (implements Observer)
- L'activity Android est le contrôleur de l'IHM décrite en XML (extends Activity)

Android_MVC

31

packages



modèle

Android_MVC

32

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
```

Android_MVC

33

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
    }
}
```

Android_MVC

34

TP4CalculletteActivity, le Contrôleur de l'IHM

```
public class Tp4CalculletteActivity extends Activity implements Observer{
    private Calcullette calcullette;

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

    ...

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

    public void onClickAdd(View v){
        ...

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

Android_MVC

35

TP4CalculletteActivity, est une vue du modèle

```
public class Tp4CalculletteActivity extends Activity implements Observer{
    private Calcullette calcullette;

    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(calcullette.toString());
        ProgressBar jauge = (ProgressBar) findViewById(R.id.jauge);
        jauge.setProgress(calcullette.size());
        actualiserInterface();
    }
}
```

Android_MVC

36

Architecture retenue

- **Application Calculette en résumé**

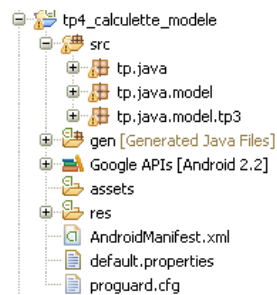
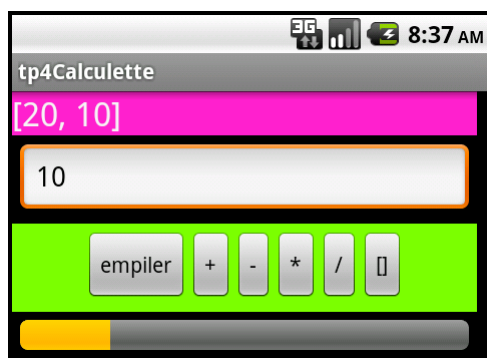
- **Observable** : Le modèle, la calculette
- **Observer** : l'Activity, mise à jour de l'interface

- **View** : le fichier XML (l'IHM)
- **Controller** : l'Activity

Android_MVC

37

Android, la calculette

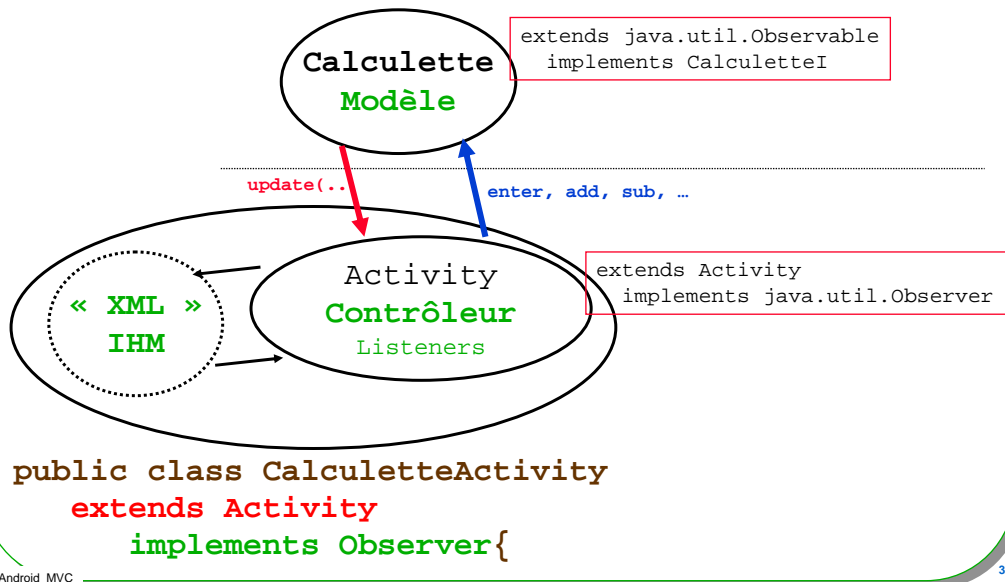


- **Discussion, réalisation ...**

Android_MVC

38

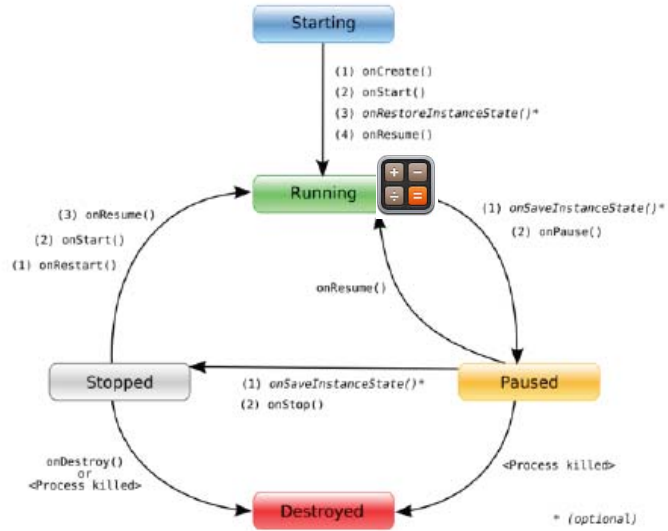
Android et MVC, discussion



Comportements attendus, cycle de vie

- **3 + 2 == 5 ?**
- **Appui sur la touche « retour »**
 - Fin de l'activité 
- **Appui sur la touche « HOME »**
 - L'activité est en Pause ... 
- **Pendant que je calcule 3 + 2, je reçois un urgent appel téléphonique**
 - telnet localhost 5554
 - gsm call 5554 
- **Une rotation de l'écran a lieu**
 - <http://developer.android.com/guide/developing/tools/emulator.html>
 - Ctrl-F11, Ctrl-F12

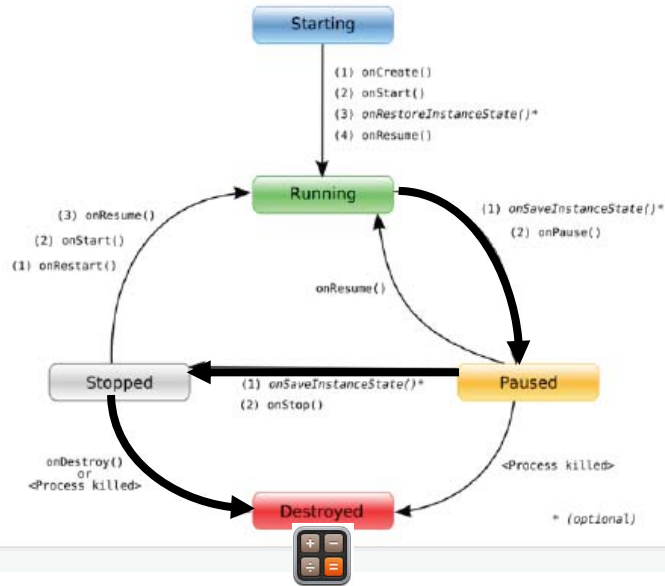
Cycle de vie d'une activity, je calcule



Android_MVC

41

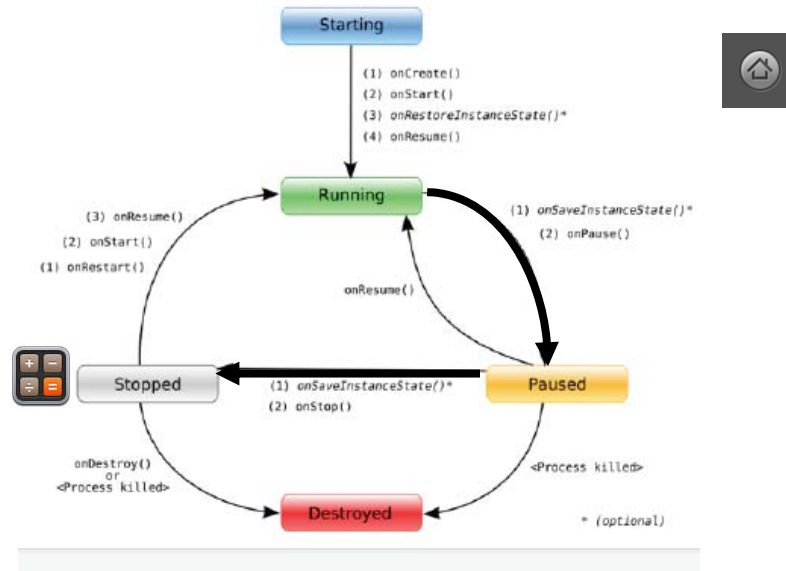
Cycle de vie d'une activity, j'ai fini



Android_MVC

42

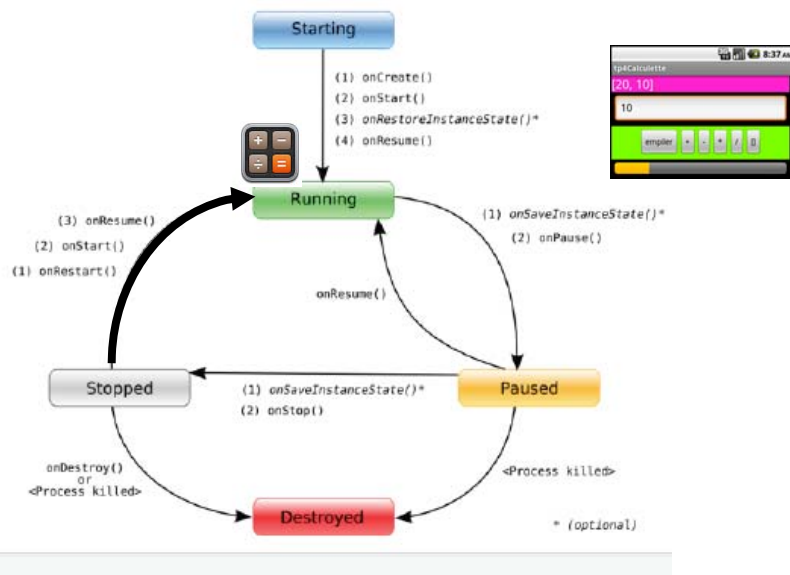
Cycle de vie d'une activity, je lance une autre activity



Android_MVC

43

Cycle de vie d'une activity, je calcule de nouveau

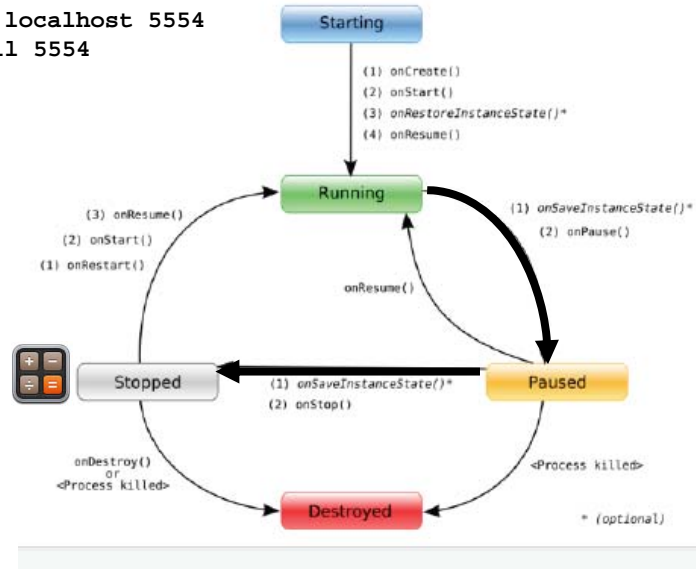


Android_MVC

44

Cycle de vie d'une activity, un appel urgent

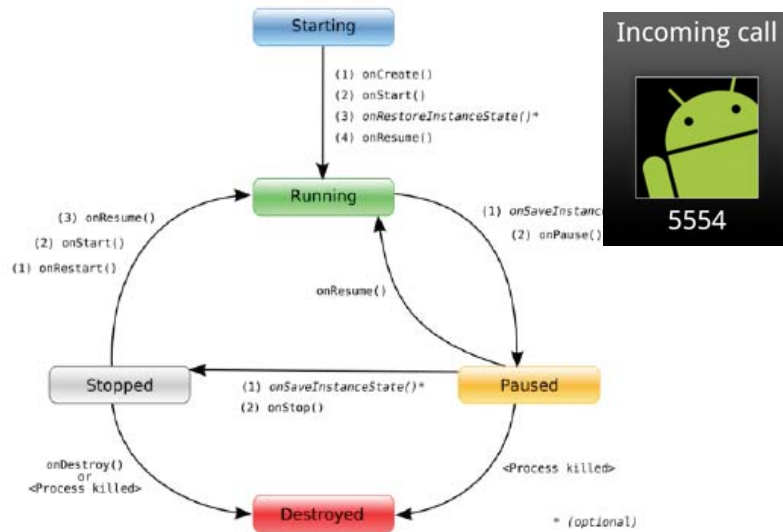
```
telnet localhost 5554
gsm call 5554
```



Android_MVC

45

Cycle de vie d'une activity, je répons



Android_MVC

46

Cycle de vie d'une activity, je raccroche

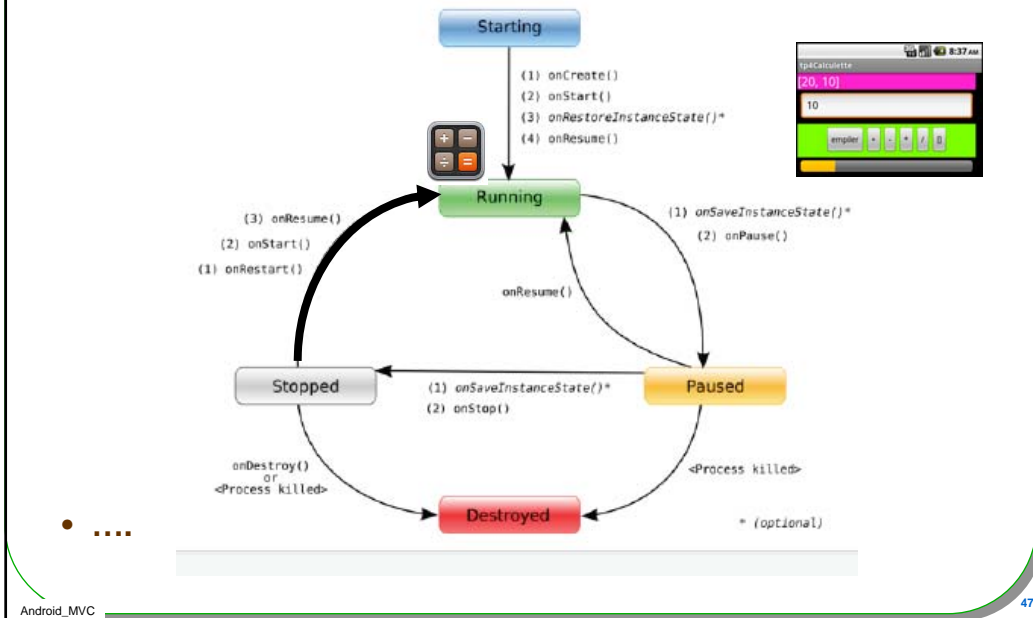
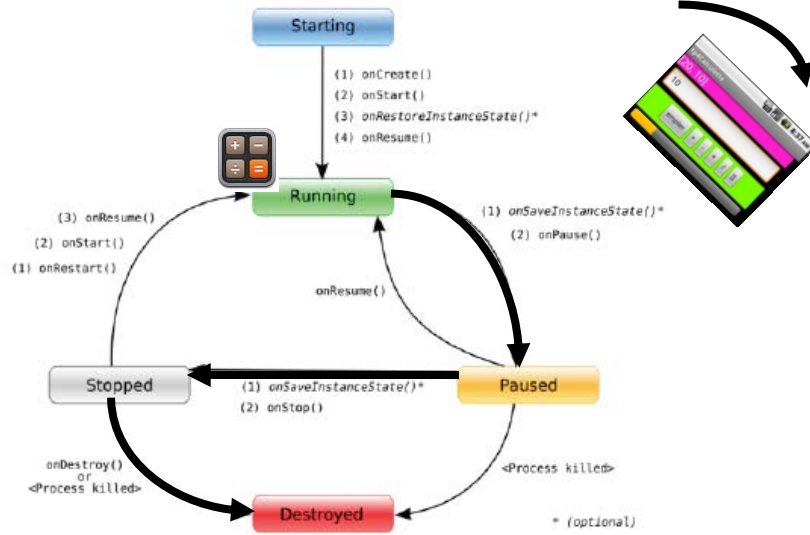


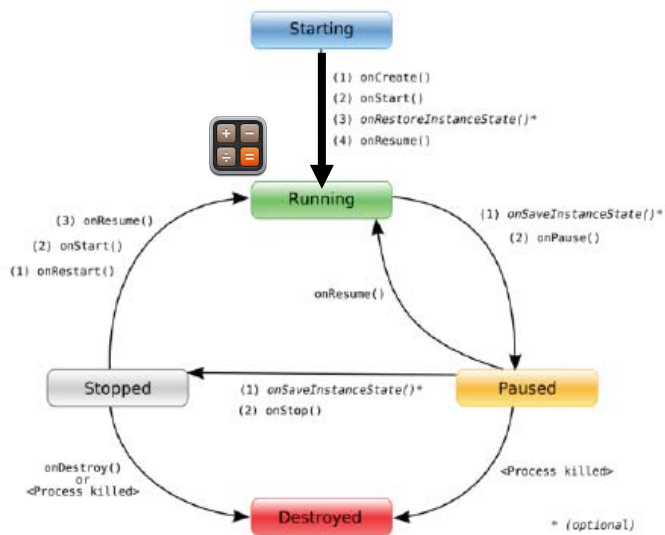
Illustration du cycle de vie

- **Illustration du cycle de vie**
 - Démonstration
- **Touche « Retour »**
 - onPause, onDestroyok
- **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

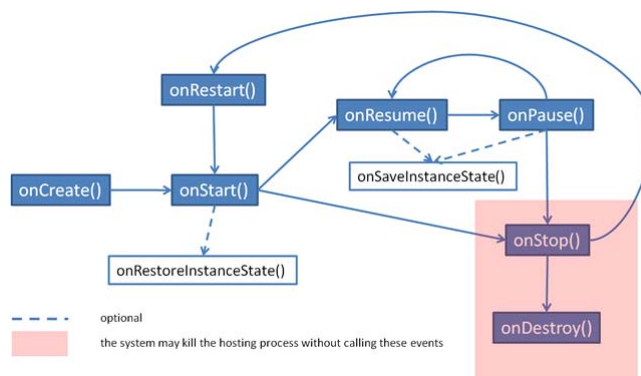
```
protected void onSaveInstanceState(Bundle out){
    out.putInt("taille",calculette.size());
    ...

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 ...

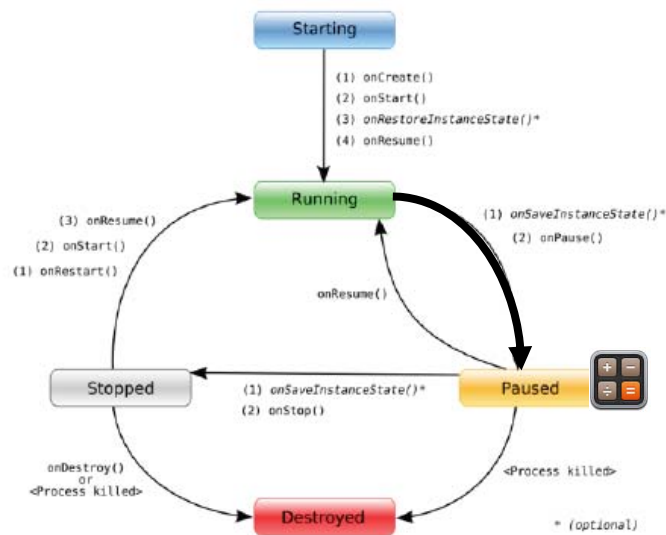


- http://www.itcsolutions.eu/wp-content/uploads/2011/08/Android_Activity_Events-Copy.png

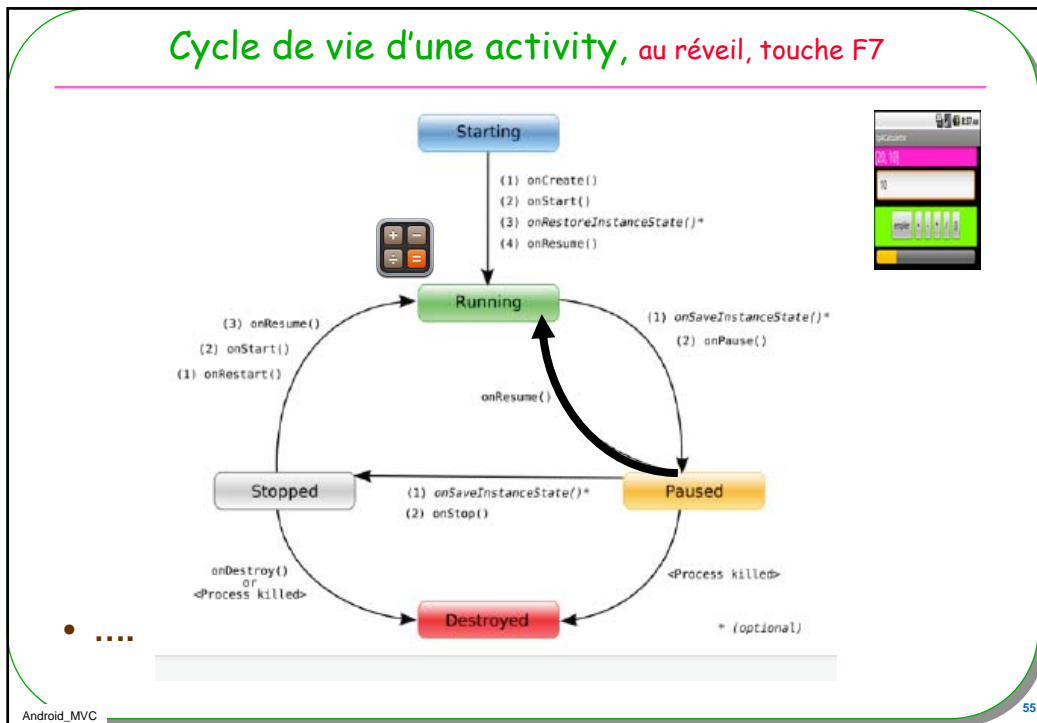
Illustration du cycle de vie

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

Cycle de vie d'une activity, écran de veille, touche F7



Cycle de vie d'une activity, au réveil, touche F7



Un extra pour la calculette

- **A chaque mauvais format du nombre, un sms est envoyé !**

- Une exception Over The Air ...

- 1) l'activité Calculette envoie le SMS

```
SmsManager sm = SmsManager.getDefault();
```

- Si j'ai la permission

- 2) L'activité standard d'envoi de SMS est appelée

- Intent sendIntent = new Intent(Intent.ACTION_VIEW);
- startActivity(sendIntent);

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" />`

Android_MVC

57

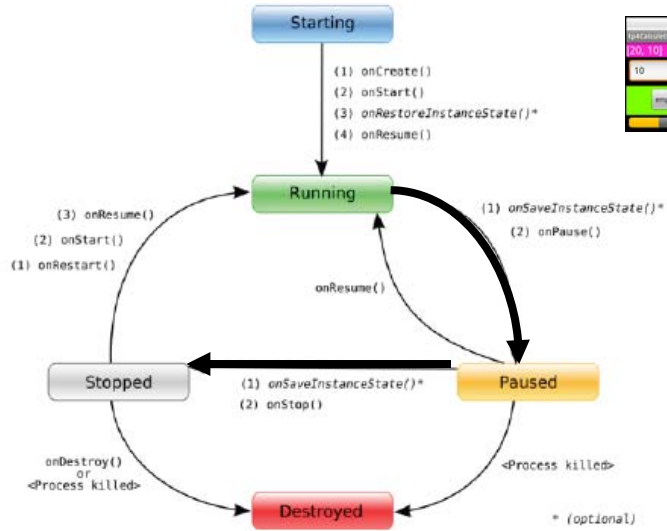
Une Activity en démarre une autre

- 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);**
- **Une pile d'activité en interne**
- **Cycle de vie**

Android_MVC

58

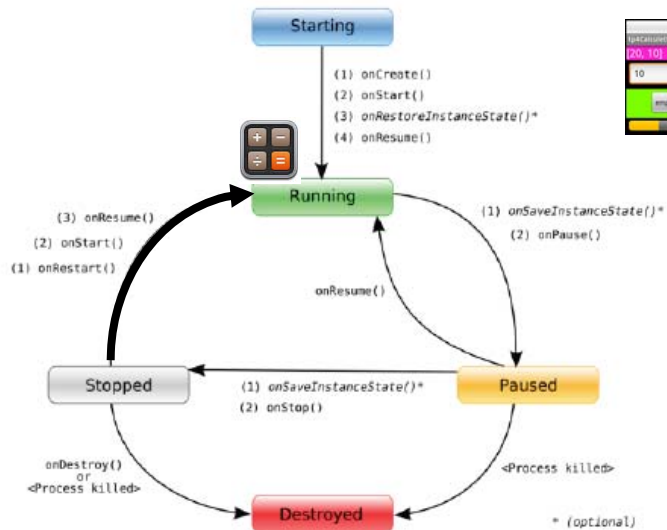
Cycle de vie d'une activity, **startActivity(sendIntent):**



Android_MVC

59

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



Android_MVC

60

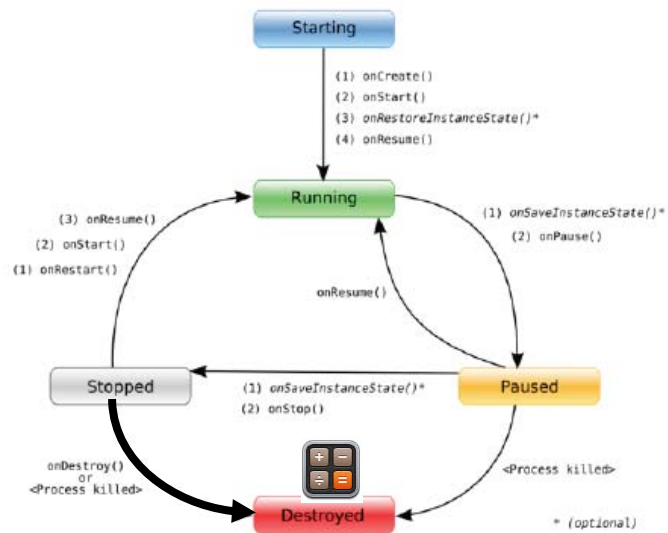
Quelques remarques

- Attention à la persistance
- Lorsque l'activité est en Pause ou Stoppée
 - Android peut décider de supprimer l'activité

Android_MVC

61

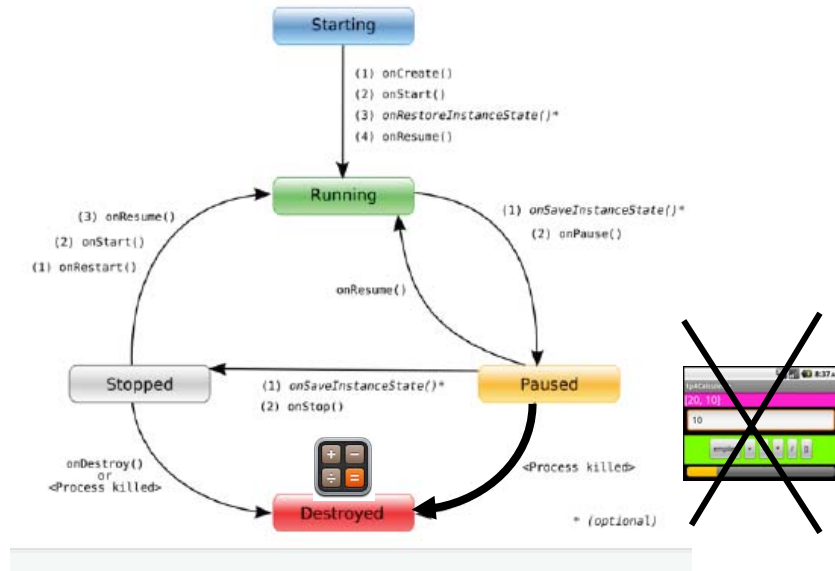
Cycle de vie d'une activité, android killer



Android_MVC

62

Cycle de vie d'une activity, android killer

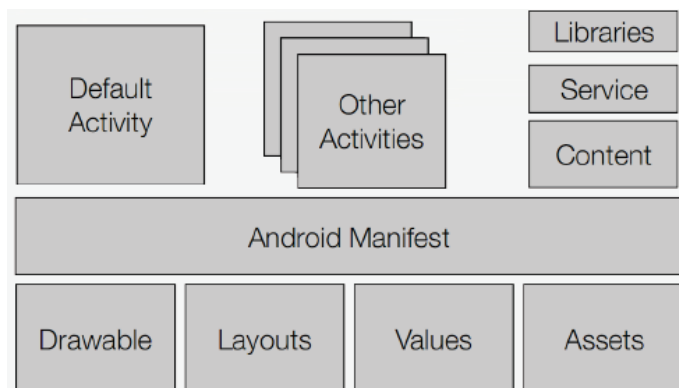


Android_MVC

63

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



Android_MVC

64

Extension possible : à chaque exception un sms !

- Trop de sms
- Un mauvais format de Nombre → Contrôle de la saisie

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

La pile des Activity



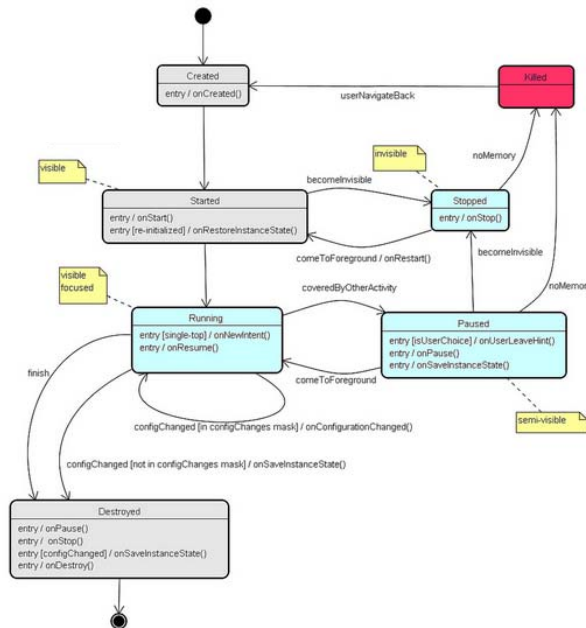
- <http://www.vineetgupta.com/2011/03/mobile-platforms-part-1-android/>
- <http://developer.android.com/guide/topics/fundamentals/tasks-and-back-stack.html>

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/>

Android_MVC

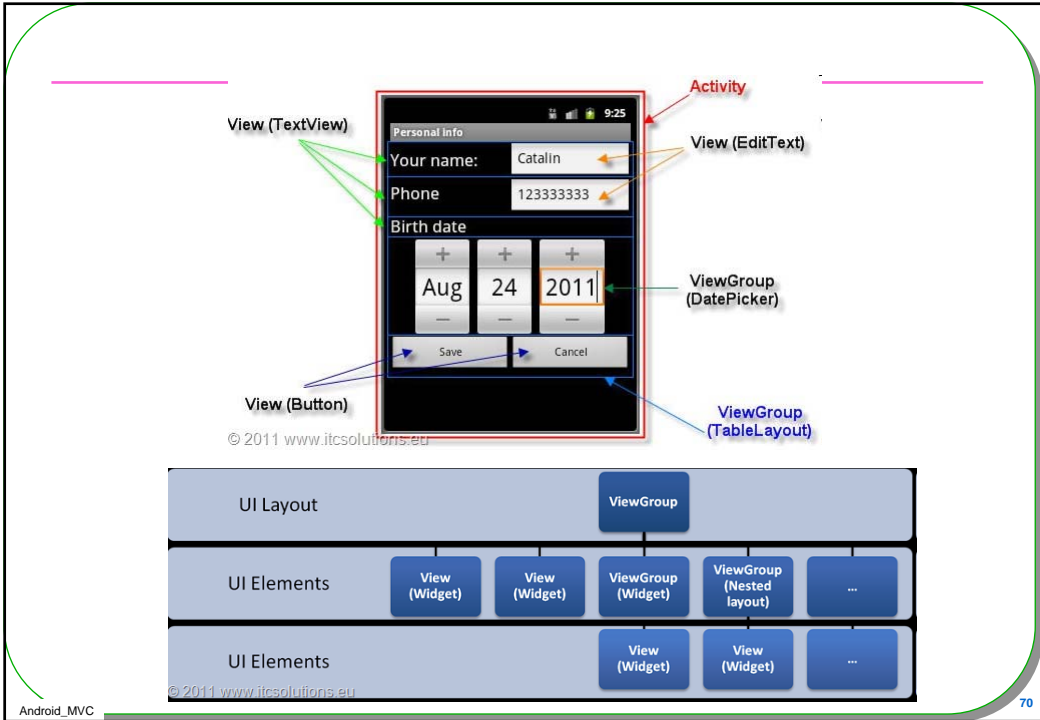
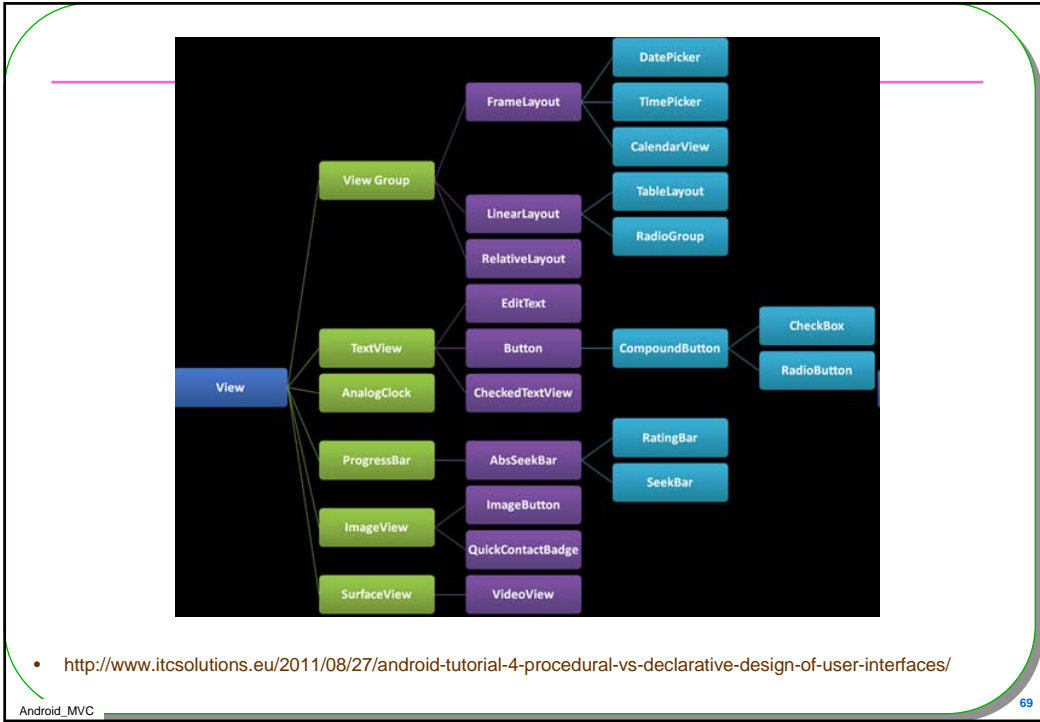
67



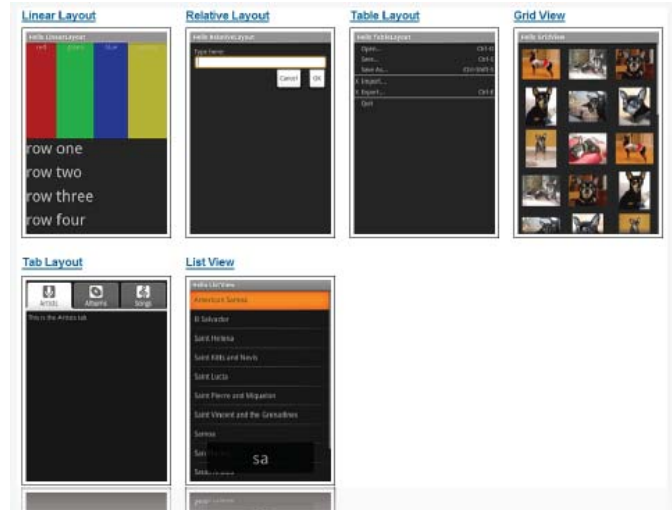
- <http://good-good-study.appspot.com/blog/posts/103001>

Android_MVC

68



Vocable



Vocable

