Développement Android (4.3)

Module 03 - Graphical User Interface (GUI)

WARNING

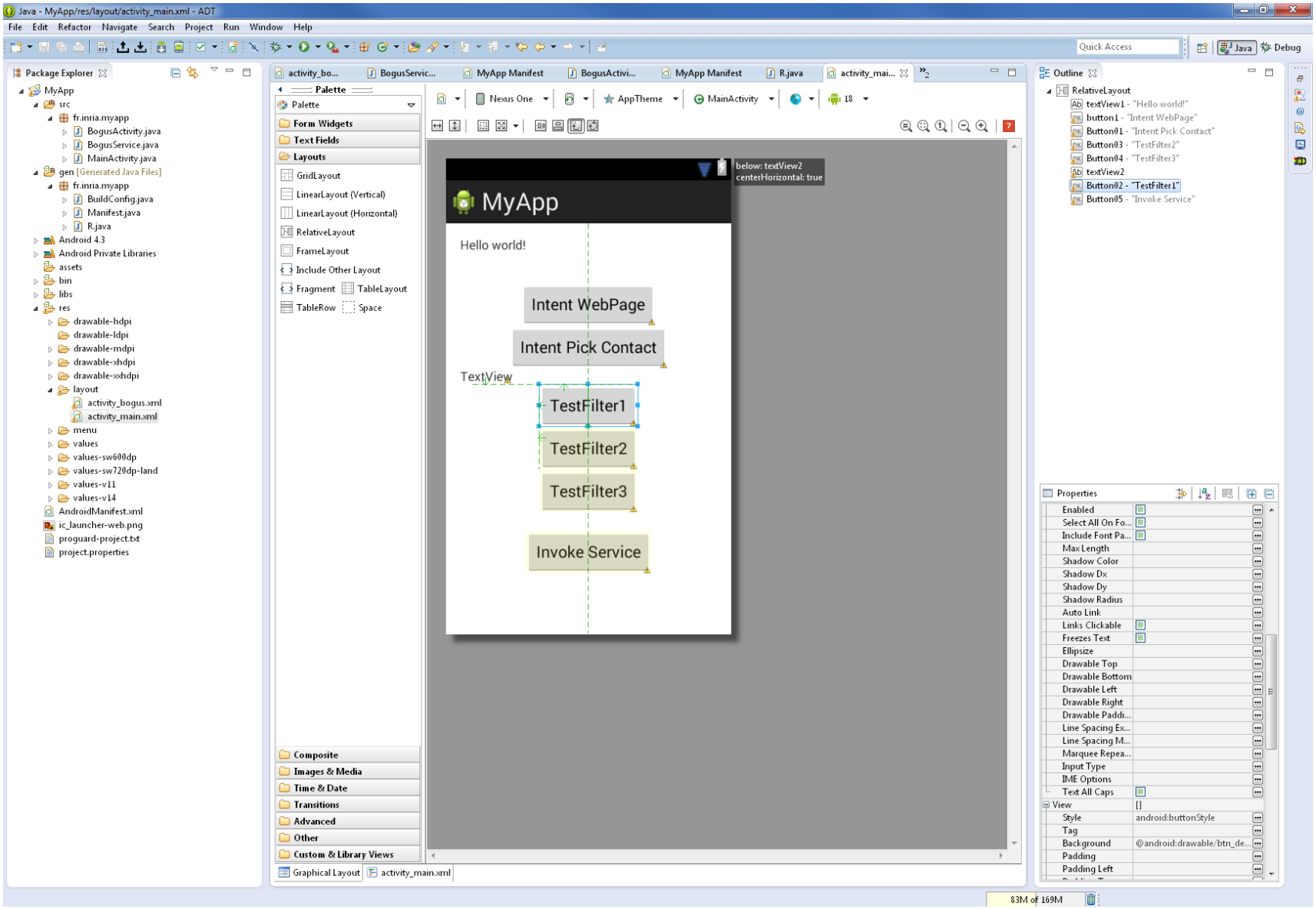
Le contenu de cette présentation est basé sur la documentation anglophone officielle d'Android, diffusée sous licence *Creative Commons Attribution 2.5* :

developer.android.com

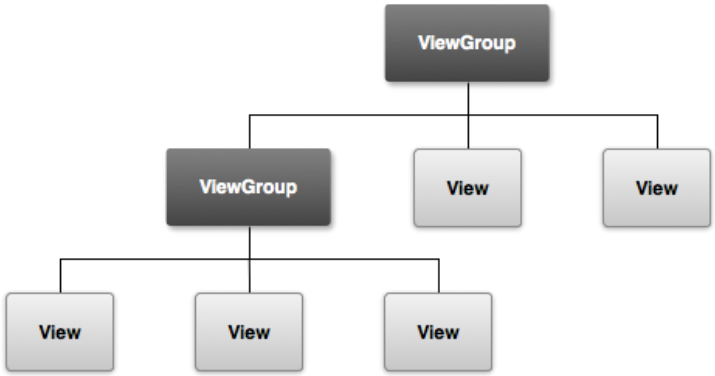
La plupart des schémas qui composent ce cours proviennent de cette documentation et sont, par conséquent, soumis à cette même licence.

http://creativecommons.org/licenses/by/2.5/

INTRODUCTION



INTRODUCTION




```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:id="@+id/text"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:text="I am a TextView" />
    <Button android:id="@+id/button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="I am a Button" />
</LinearLayout>
```

- Tous les composants graphiques héritent de View et ViewGroup (pattern Composite).
- L'interface graphique est décrite en XML.
- Le SDK transforme cette description en fichier R.java. La classe R permettra d'accéder aux instances concrètes de chaque composant grâce à leurs IDs.
- Spoiler : Android gère l'interface graphique dans le thread principal (ou UI Thread).

INTRODUCTION

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>

<string name="app_name">MyApp</string>
<string name="hello_world">Hello world!</string>

</resources>
```

strings.xml

- @+id/id_of_an_element : déclare un id.
- @id/id_of_an_element : référence un id déjà déclaré.
- @string/string_name : une chaîne dans strings.xml.
- @drawable/my_image : my_image.png dans /res/drawable
- @anim/my_animation : my_animation.xml (descripteur d'animation) dans /res/anim (descripteur d'animation).
- @layout/my_layout : my_layout.xml dans /res/layout (pour inclusion, par exemple).

Introduction

- View :
  - Identifiant

    ```
    findViewById(R.id.myElementID);
    ```
  - layout_width, layout_height
    - Valeur exacte (px, mm, in, pt, dp).
    - wrap_content : s'adapter au contenu.
    - match_parent : s'adapter à la taille du parent.
  - Quelques méthodes :
    - getLeft(), getTop(), getRight(), getBottom()
    - getPaddingLeft(), getPaddingTop(), …
- ViewGroup :
  - layout_marginLeft, layout_marginTop, …

INTRODUCTION

```java
protected void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main_layout); // res/layout/main_layout.xml

    Button myButton = (Button) findViewById(R.id.my_button);
}
```

```xml
<Button android:id="@+id/my_button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/my_button_text" />
```

Remarque : Attention à bien manipuler l'interface après le setContentView, sinon findViewById retourne null à tous les coups.

Introduction

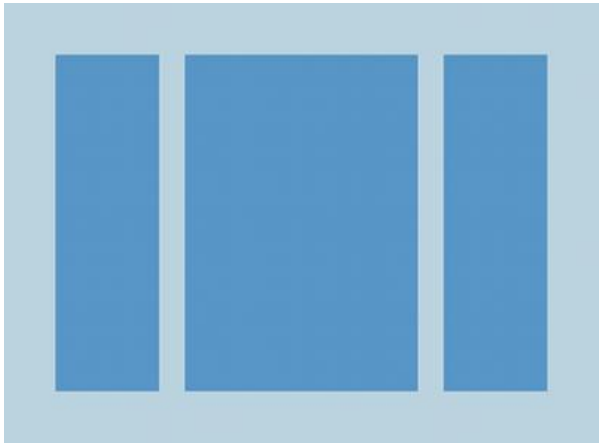
```
<Button android:layout_width="100dp" android:layout_height="wrap_content"
    android:text="@string/send"
    android:onClick="sendMessage"
    android:id="@+id/button_send"
/>

public class MyActivity extends Activity
{
    public void sendMessage(View v) // solution 1
    {
    }

    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.my_activity);
        Button button = (Button) findViewById(R.id.button_send);
        button.setOnClickListener(new View.OnClickListener() // solution 2
        {
            @Override
            public void onClick(View v)
            {
            }
        });
    }
}
```

Les layouts

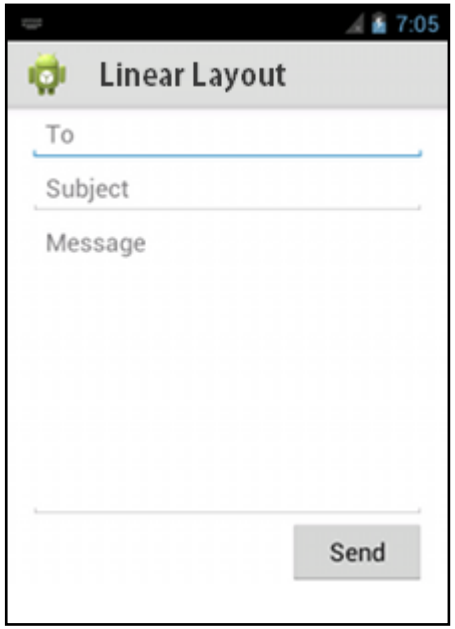
- Un layout définit une structure d'organisation pour tous les composants qu'il contient (ViewGroup).
- L'interface graphique d'une activité utilise un layout (la racine de la description XML).
- Les layouts peuvent contenir des layouts.
- Les layouts, comme la plupart des ressources, se trouvent dans le dossier "res", puis "layout".

```
public void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main_layout); // res/layout/main_layout.xml
}
```

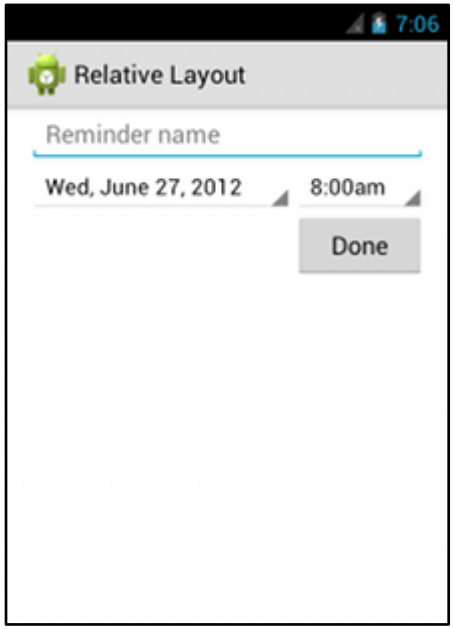
Les layouts

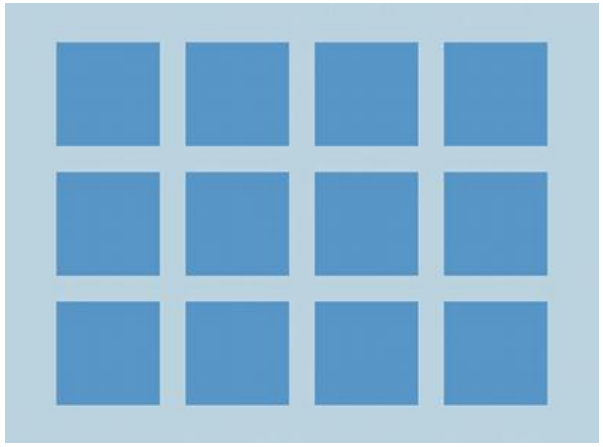


LinearLayout

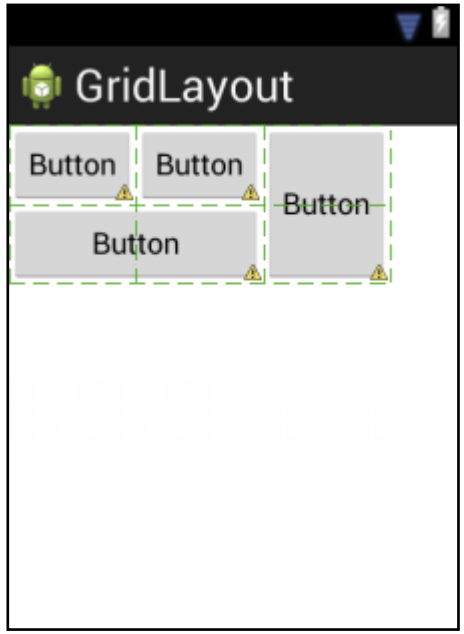


RelativeLayout





GridLayout



LinearLayout

```
<?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:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:orientation="vertical" >
    <EditText
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:hint="@string/to" />
    <EditText
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:hint="@string/subject" />
    <EditText
        android:layout_width="fill_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:gravity="top"
        android:hint="@string/message" />
    <Button
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:layout_gravity="right"
        android:text="@string/send" />
</LinearLayout>
```

- Le weight indique que le composant doit utiliser l'espace restant (d'où height = 0).
- Si plusieurs composants définissent un poid, alors l'espace est partagé relativement à chaque poids.

RelativeLayout

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp" >
    <EditText
        android:id="@+id/name"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:hint="@string/reminder" />
    <Spinner
        android:id="@+id/dates"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/name"
        android:layout_alignParentLeft="true"
        android:layout_toLeftOf="@+id/times" />
    <Spinner
        android:id="@id/times"
        android:layout_width="96dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/name"
        android:layout_alignParentRight="true" />
    <Button
        android:layout_width="96dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/times"
        android:layout_alignParentRight="true"
        android:text="@string/done" />
</RelativeLayout>
```

- Positionner un composant relativement à un autre :
  - toLeftOf, toRightOf
  - below, above
  - alignTop, alignLeft
  - …
- Positionner un composant relativement au parent :
  - alignParentLeft, alignParentTop
  - centerVertical, centerHorizontal
  - …

GridLayout

```
<?xml version="1.0" encoding="utf-8"?>
<GridLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/GridLayout1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:columnCount="3"
    android:rowCount="2"
    tools:context=".GridLayoutActivity" >

    <Button
        android:id="@+id/button3"
        android:layout_column="0"
        android:layout_gravity="left|top"
        android:layout_row="0" />
    <Button
        android:id="@+id/button1"
        android:layout_column="1"
        android:layout_gravity="left|top"
        android:layout_row="0" />
    <Button
        android:id="@+id/button2"
        android:layout_column="2"
        android:layout_gravity="fill_vertical"
        android:layout_row="0"
        android:layout_rowSpan="2" />
    <Button
        android:id="@+id/button4"
        android:layout_column="0"
        android:layout_columnSpan="2"
        android:layout_gravity="fill_horizontal"
        android:layout_row="1" />
</GridLayout>
```

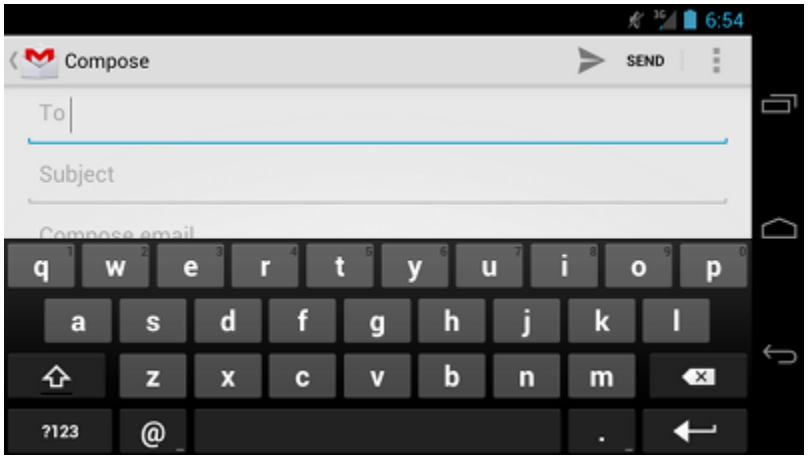
- column, row : positionner le composant dans la grille.
- columnSpan, rowSpan : fusionner des colonnes et des lignes.

Button


```
<Button
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="@string/button_text"
... />

<ImageButton
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:src="@drawable/button_icon"
... />

<Button
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="@string/button_text"
android:drawableLeft="@drawable/button_icon"
... />
```

EditText

```
<EditText
    android:id="@+id/email_address"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:hint="@string/email_hint"
    android:inputType="textEmailAddress" />
```

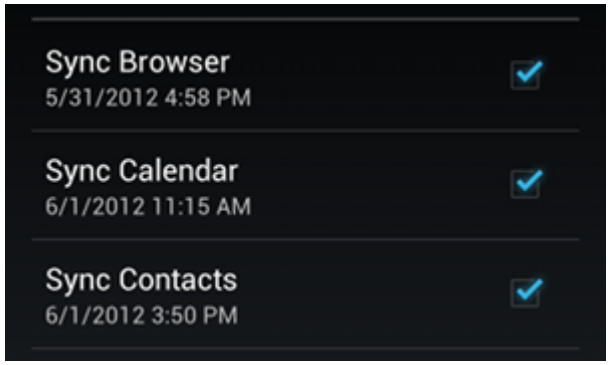


inputType = " textEmailAddress"

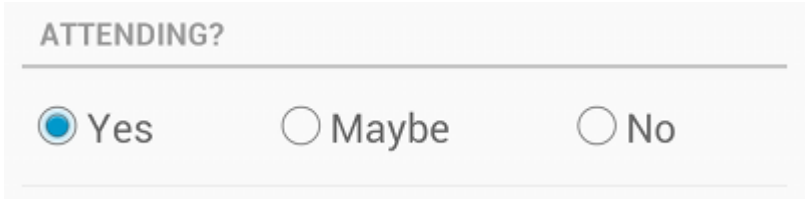
inputType = "phone"

CheckBox, RadioButton, ToggleButton

```
<CheckBox android:id="@+id/checkbox"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/cheese"
        android:onClick="onCheckboxClicked"/>
```




```
<RadioGroup
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">
    <RadioButton android:id="@+id/radio_yes"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/yes"
        android:onClick="onRadioButtonClicked"/>
    <RadioButton android:id="@+id/radio_no"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/no"
        android:onClick="onRadioButtonClicked"/>
</RadioGroup>
```




```
<ToggleButton
    android:id="@+id/togglebutton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textOn="Vibrate on"
    android:textOff="Vibrate off"
    android:onClick="onToggleClicked"/>
```

Switch extends ToggleButton

QUELQUES AUTRES

```
<SeekBar
        android:id="@+id/seekBar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:max="100"
        android:progress="75" />
```

```
<ProgressBar
        android:id="@+id/progressBar"
        style="?android:attr/progressBarStyleHorizontal"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:indeterminate="true"
        android:max="100"
        android:progress="45" />
```



```
 style="?android:attr/progressBarStyleLarge"
```

Un petit exemple avec RadioButton

```
public void onRadioButtonClicked(View view)
{
    boolean checked = ((RadioButton) view).isChecked();
    switch(view.getId())
    {
        case R.id.radio_yes:
            if (checked)
                // yes
            break;
        case R.id.radio_maybe:
            if (checked)
                // maybe
            break;
        case R.id.radio_no:
            if (checked)
                // no
            break;
    }
}
```

Un petit exemple avec RadioButton

```
RadioGroup radio = (RadioGroup)findViewById(R.id.gender);
switch(user.getGender())
{
     case Male:
          radio.check(R.id.gender_male);
          break;
     case Female:
          radio.check(R.id.gender_female);
          break;
     case Unspecified:
          radio.check(R.id.gender_unspecified);
          break;
}

RadioGroup radio = (RadioGroup)findViewById(R.id.gender);
switch(radio.getCheckedRadioButtonId())
{
     case R.id.gender_male:
          user.setGender(Gender.Male);
          break;
     case R.id.gender_female:
          user.setGender(Gender.Female);
          break;
     default:
          user.setGender(Gender.Unspecified);
}
```