Android
–
Activités - Événements

C. Raïevsky
Avec la courtoisie de S. Jean

UNIVERSITÉ Grenoble Alpes — IUT Valence

Département Informatique

2019

Descripteurs et ressources

- Retour sur l'exemple « *Hello World* »
  - `activity_main.xml` → layout de l'activité principale (cf. manifeste)
  - `hello_textview` → identifiant de la vue (de type `TextView`)
  - `hello_world` → alias texte défini dans `values(...)/strings.xml`

activity_main.xml | R.java | MainActivity.java

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      xmlns:tools="http://schemas.android.com/tools"
4      android:layout_width="match_parent"
5      android:layout_height="match_parent"
6      android:gravity="center_vertical|center_horizontal" >
7
8      <TextView
9          android:id="@+id/hello_textview"
10         android:layout_width="wrap_content"
11         android:layout_height="wrap_content"
12         android:text="@string/hello_world"
13         android:textSize="24sp" />
14
15 </RelativeLayout>
16
```

Graphical Layout | activity_main.xml

La classe `R`

- Classe **générée** contenant des **définitions d'attributs statiques**
  - `R.attr.(...)` → attributs de style
  - `R.drawable.(...)` → ressources graphiques
  - `R.layout.(...)` → layouts
  - `R.string.(...)` → alias textes

activity_main.xml | R.java

```
1  /* AUTO-GENERATED FILE.  DO NOT MODIFY.
7
8  package fr.iutvalence.info.android.caesar;
9
10 public final class R {
11     public static final class attr {
12     }
13     public static final class drawable {
14         public static final int ic_launcher=0x7f020000;
15     }
16     public static final class layout {
17         public static final int activity_main=0x7f030000;
18     }
19     public static final class string {
20         public static final int app_name=0x7f040000;
21         public static final int hello_world=0x7f040001;
22     }
23 }
24
```


Liens entre ressources et code

- Seconde version du *Hello World*
  - Plus de pré-remplissage du champs texte

activity_main.xml | R.java | MainActivity.java

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      xmlns:tools="http://schemas.android.com/tools"
4      android:layout_width="match_parent"
5      android:layout_height="match_parent"
6      android:gravity="center_vertical|center_horizontal" >
7
8      <TextView
9          android:id="@+id/hello_textview"
10         android:layout_width="wrap_content"
11         android:layout_height="wrap_content"
12         android:textSize="24sp" />
13 </RelativeLayout>
14
```

Graphical Layout | activity_main.xml

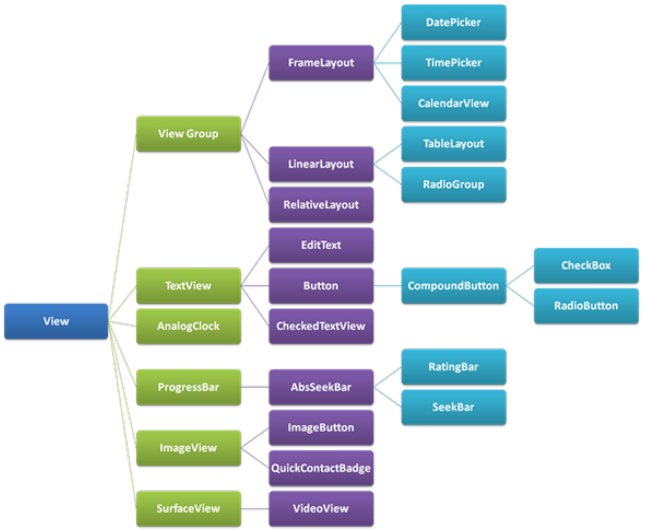
Code de l'activité

- `findViewByID`
  - Retourne la **référence d'un objet** permettant de manipuler une **vue désignée par son identifiant**
  - Remarque : à **transtyper** dans le type concret de la vue

activity_main.xml | R.java | MainActivity.java

```
1  package fr.iutvalence.info.android.caesar;
2
3  import android.app.Activity;
6
7  public class MainActivity extends Activity
8  {
9      @Override
10     protected void onCreate(Bundle savedInstanceState)
11     {
12         super.onCreate(savedInstanceState);
13         setContentView(R.layout.activity_main);
14
15         TextView textView = (TextView) findViewById(R.id.hello_textview);
16         textView.setText(R.string.hello_world);
17     }
18
19 }
20
```

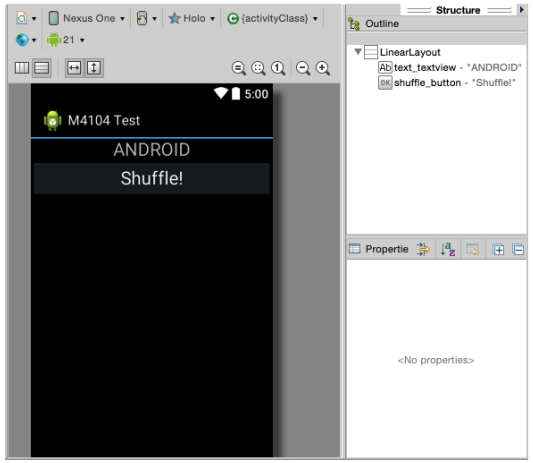

Hiérarchie des classes Android dérivant de `View`



- Source : `http://www.itcsolutions.eu`

Capture d'évènement sur un bouton : exemple

- Champs texte ré-rempli avec un mot
- Bouton permettant de mélanger les lettres du mot



Auditeur d'évènement : méthode classique

- Implémentation de l'interface `onClickListener` par l'activité
  - Redéfinition de la méthode `onClick(View v)`
  - Remarque :

    le paramètre `View` permet d'intercepter un évènement sur plusieurs composants, identifiables via un appel à `getID`
- **Enregistrement de l'activité** en tant qu'**auditeur d'évènement** auprès du bouton, dans la méthode `onCreate`
  - Remarque :

    l'objet correspondant au bouton n'est créé que lors de l'affichage, un appel à `findViewById()` placé avant l'installation du layout échoue

Auditeur d'évènement : méthode classique

activity_main.xml | R.java | MainActivity.java | strings.xml

```java
package fr.iutvalence.info.android.caesar;

import java.util.Random;

public class MainActivity extends Activity implements OnClickListener
{
    private String shuffle(String s)

    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Button button = (Button) findViewById(R.id.shuffle_button);
        button.setOnClickListener(this);
    }

    @Override
    public void onClick(View v)
    {
        TextView textView = (TextView) findViewById(R.id.text_textview);
        textView.setText(shuffle((String) textView.getText()));
    }

}
```


Auditeur d'évènement : raccourci

- Attribut `android:onClick`
  - Méthode (de l'activité) traitant l'évènement

activity_main.xml | R.java | MainActivity.java | strings.xml

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

    <TextView
        android:id="@+id/text_textview"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/text"
        android:textSize="24sp" />

    <Button
        android:id="@+id/shuffle_button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/shuffle"
        android:onClick="buttonOnClickListener"
        android:textSize="24sp" />

</LinearLayout>
```

Graphical Layout | activity_main.xml

Auditeur d'évènement : raccourci

- L'activité n'a plus besoin d'implémenter `onClickListener` (implicite)
- La méthode doit prendre en paramètre la référence d'un objet `View`

activity_main.xml | R.java | MainActivity.java | strings.xml

```java
package fr.iutvalence.info.android.caesar;

import java.util.Random;

public class MainActivity extends Activity
{
    private String shuffle(String s)

    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    public void buttonOnClickListener(View v)
    {
        TextView textView = (TextView) findViewById(R.id.text_textview);
        textView.setText(shuffle((String) textView.getText()));
    }

}
```


Intentions

- Les intentions (***intent***) sont des **objets/événements** destinés à
  - **déclencher le démarrage** d'une activité ou d'un service
  - **avertir** un récepteur de message

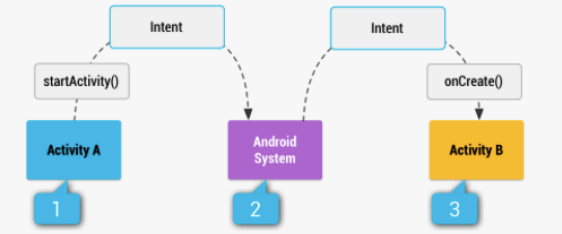


Figure 1. Illustration of how an implicit intent is delivered through the system to start another activity: **[1]** *Activity A* creates an `Intent` with an action description and passes it to `startActivity()`. **[2]** The Android System searches all apps for an intent filter that matches the intent. When a match is found, **[3]** the system starts the matching activity (*Activity B*) by invoking its `onCreate()` method and passing it the `Intent`.

Intentions

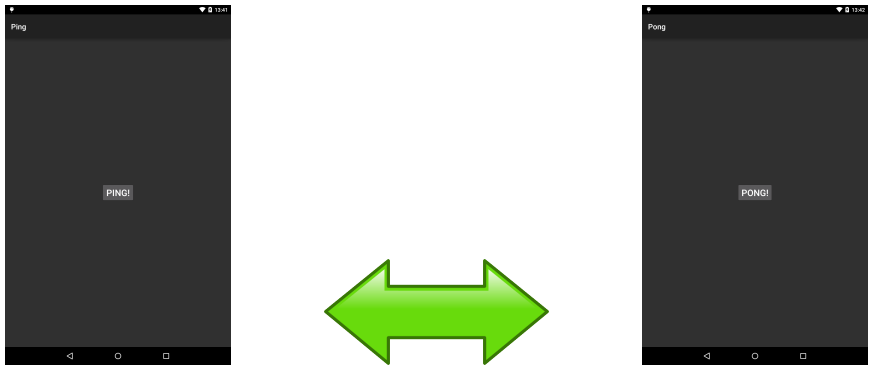
- Une intention est caractérisée par :
  - des **informations permettant au système de trouver la cible**
    - une **catégorie** (optionnelle)
    - une **action**
  - des **données** permettant de spécialiser l'intention
    - sous la forme ***nom/valeur***
- L'action peut être :
  - **Explicite**
    - **Classe de l'activité à démarrer**
  - **Implicite**
    - **Constante** (de type `String` prédéfinie ou définie par l'activité
    - Sémantique plus générale, activité non forcément existante ou unique
    - cf. `.../reference/android/content/Intent.html`

Orchestration des activités

- L'**exécution** d'une application peut être réalisée par l'**exécution séquentielle d'activités** (déclenchées par des intentions) appartenant à des **applications différentes**
  - L'activité n'a pas besoin d'embarquer du code lié à l'activité qu'elle déclenche via l'intention
  - L'activité n'a pas besoin de connaitre précisément la cible de l'intention (c'est le système qui la trouve)
- Exemple :
  - *Une activité qui appelle un numéro enfoui dans un QR-code pourrait le faire en démarrant les activités "scanner un QR-code" et "passer un appel téléphonique" via les intentions associées*

Démarrage d'une nouvelle activité

- Application exemple : ping/pong



Ping/Pong : manifeste

- Deux activités, une seule activité principale

PingActivity.java | ping_or_pong.xml | AndroidManifest.xml

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

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="21" />

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name">
        <activity
            android:name=".PingActivity"
            android:label="Ping" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".PongActivity"
            android:label="Pong" >
        </activity>
    </application>
</manifest>
```

AndroidManifest.xml

Ping/Pong : layout

- Layout commun (pour des raisons de simplicité)
- Remarque :

  Même nom de méthode pour le traitement d'évènements mais implémentation dans chaque activité

PingActivity.java | ping_or_pong.xml | AndroidManifest.xml

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

    <Button
        android:id="@+id/ping_pong_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="buttonOnClickListener"
        android:textSize="24sp" />
</LinearLayout>
```

Graphical Layout | ping_or_pong.xml

Ping/Pong : code de l'activité

- Remarque : code similaire pour l'autre activité

PingActivity.java | ping_or_pong.xml | AndroidManifest.xml

```java
package fr.iutvalence.info.android.caesar;

import android.app.Activity;

public class PingActivity extends Activity
{
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.ping_or_pong);
        Button button = (Button) findViewById(R.id.ping_pong_button);
        button.setText("Ping!");
    }

    public void buttonOnClickListener(View v)
    {
        Intent intent = new Intent(this, PongActivity.class);
        startActivity(intent);
    }
}
```


Ping/Pong : code de l'activité

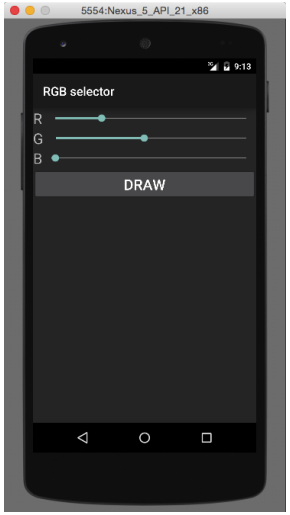
- Intent **explicite**, en paramètre à l'appel à `startActivity`

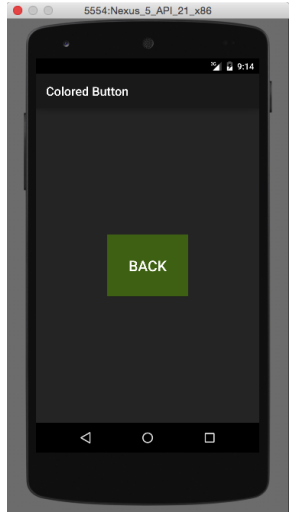
PingActivity.java | ping_or_pong.xml | AndroidManifest.xml

```java
package fr.iutvalence.info.android.caesar;

import android.app.Activity;

public class PingActivity extends Activity
{
    protected void onCreate(Bundle savedInstanceState)

    public void buttonOnClickListener(View v)
    {
        Intent intent = new Intent(this, PongActivity.class);
        startActivity(intent);
    }
}
```


Passage d'informations entre activités

- Application exemple : un bouton coloré sur demande





RGB Button : manifeste

- Deux activités

AndroidManifest | ColoredButtonAc | colored_button.

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

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="21" />

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name">
        <activity
            android:name=".RGBSelectorActivity"
            android:label="RGB selector" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".ColoredButtonActivity"
            android:label="Colored Button" >
        </activity>
    </application>
</manifest>
```

AndroidManifest.xml

RGB Button : layout de l'activité *RGB selector* (extrait)

RGBSelectorActi | AndroidManifest | rgb_selector.xm | Outline

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

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center_horizontal"
        android:orientation="horizontal" >

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_marginEnd="5dp"
            android:text="R"
            android:textSize="24sp" />

        <SeekBar
            android:id="@+id/seek_r"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:contentDescription="R"
            android:max="255" />
    </LinearLayout>
```

Outline : xml, LinearLayout, LinearLayout, TextView: R, SeekBar: @+id/seek_r, LinearLayout, TextView: G, SeekBar: @+id/seek_g, LinearLayout, TextView: B, SeekBar: @+id/seek_b, Button: @+id/buttonDraw

Graphical Layout | rgb_selector.xml

- Remarque :

  `android:max` fixe la borne supérieure de la barre de progression

RGB Button : layout de l'activité *Colored Button*

AndroidManifest | ColoredButtonAc | colored_button.

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

    <Button
        android:id="@+id/buttonBack"
        android:layout_width="130dp"
        android:layout_height="100dp"
        android:onClick="buttonOnClickListener"
        android:text="Back"
        android:textSize="24sp" />

</LinearLayout>
```

Graphical Layout | colored_button.xml

RGB Button : code de l'activité *RGB selector*

colored_button. | rgb_selector.xm | RGBSelectorActi

```java
package fr.iutvalence.info.android;

import android.app.Activity;

public class RGBSelectorActivity extends Activity
{
    protected void onCreate(Bundle savedInstanceState)

    public void buttonOnClickListener(View v)
    {
        SeekBar seekBarR = (SeekBar) findViewById(R.id.seek_r);
        SeekBar seekBarG = (SeekBar) findViewById(R.id.seek_g);
        SeekBar seekBarB = (SeekBar) findViewById(R.id.seek_b);

        Intent intent = new Intent(this, ColoredButtonActivity.class);
        intent.putExtra("r", seekBarR.getProgress());
        intent.putExtra("g", seekBarG.getProgress());
        intent.putExtra("b", seekBarB.getProgress());

        startActivity(intent);
    }
}
```

- `putExtra` : chargement de l'intention avec une nouvelle donnée
  - **paire clé/valeur**, clé de type `String`
  - valeur de type primitif ou `String`, ou tableau primitif/`String`

RGB Button : code de l'activité *Colored Button*

ColoredButtonAc | colored_button. | rgb_selector.xm

```
1  package fr.iutvalence.info.android;
2
3  import android.app.Activity;
9
10 public class ColoredButtonActivity extends Activity
11 {
12     @Override
13     protected void onCreate(Bundle savedInstanceState)
14     {
15         super.onCreate(savedInstanceState);
16         setContentView(R.layout.colored_button);
17
18         Intent intent = this.getIntent();
19
20         int r = intent.getIntExtra("r", 0);
21         int g = intent.getIntExtra("g", 0);
22         int b = intent.getIntExtra("b", 0);
23
24         Button button = (Button) findViewById(R.id.buttonBack);
25         button.setBackgroundColor(Color.rgb(r, g, b));
26     }
27
28     public void buttonOnClickListener(View v)
29     {
30         Intent intent = new Intent(this, RGBSelectorActivity.class);
31         startActivity(intent);
32     }
33 }
```

- `getIntent` :

  obtention de l'intention à l'origine du démarrage de l'activité

RGB Button : code de l'activité *Colored Button*

ColoredButtonAc | colored_button. | rgb_selector.xm

```
1  package fr.iutvalence.info.android;
2
3  import android.app.Activity;
9
10 public class ColoredButtonActivity extends Activity
11 {
12     @Override
13     protected void onCreate(Bundle savedInstanceState)
14     {
15         super.onCreate(savedInstanceState);
16         setContentView(R.layout.colored_button);
17
18         Intent intent = this.getIntent();
19
20         int r = intent.getIntExtra("r", 0);
21         int g = intent.getIntExtra("g", 0);
22         int b = intent.getIntExtra("b", 0);
23
24         Button button = (Button) findViewById(R.id.buttonBack);
25         button.setBackgroundColor(Color.rgb(r, g, b));
26     }
27
28     public void buttonOnClickListener(View v)
29     {
30         Intent intent = new Intent(this, RGBSelectorActivity.class);
31         startActivity(intent);
32     }
33 }
```

- `getXXXExtra` : **obtention de la valeur** typée associée à une clé
  - Remarque : valeur par défaut spécifiée si clé inexistante

Retour de résultat par une activité

- Pour le moment, le retour à l'activité *RGB selector* provoque une remise à zéro des seekbar

RGB Button : code de l'activité *RGB selector*

ColoredButtonAc | colored_button. | rgb_selector.xm | RGBSelectorActi

```
1  package fr.iutvalence.info.android;
2
3  import android.app.Activity;
8
9  public class RGBSelectorActivity extends Activity
10 {
12     protected void onCreate(Bundle savedInstanceState)
17
18     public void buttonOnClickListener(View v)
19     {
20         SeekBar seekBarR = (SeekBar) findViewById(R.id.seek_r);
21         SeekBar seekBarG = (SeekBar) findViewById(R.id.seek_g);
22         SeekBar seekBarB = (SeekBar) findViewById(R.id.seek_b);
23
24         Intent intent = new Intent(this, ColoredButtonActivity.class);
25         intent.putExtra("r", seekBarR.getProgress());
26         intent.putExtra("g", seekBarG.getProgress());
27         intent.putExtra("b", seekBarB.getProgress());
28
29         this.startActivityForResult(intent, 1);
30     }
31
33     protected void onActivityResult(int requestCode, int resultCode, Intent data)
46 }
```

- `startActivityForResult` : démarre une activité (intention passée en paramètre) produisant un **résultat attendu**
  - 2^{nd} paramètre $\rightarrow$ **code de requête** rappelé lors du résultat

RGB Button : code de l'activité *Colored Button*

ColoredButtonAc | colored_button.

```
1  package fr.iutvalence.info.android;
2
3  import android.app.Activity;
9
10 public class ColoredButtonActivity extends Activity
11 {
12     private int r, g, b;
13
15     protected void onCreate(Bundle savedInstanceState)
28
29     public void buttonOnClickListener(View v)
30     {
31         Intent intent = new Intent();
32         intent.putExtra("r", this.r);
33         intent.putExtra("g", this.g);
34         intent.putExtra("b", this.b);
35         this.setResult(RESULT_OK, intent);
36         this.finish();
37     }
38 }
```

- `setResult` : génération du résultat (code retour/ intention)
- `finish` : terminaison et retour du résultat

RGB Button : code de l'activité *RGB selector*

ColoredButtonAc | colored_button. | rgb_selector.xm | RGBSelectorActi

```
1  package fr.iutvalence.info.android;
2
3  import android.app.Activity;
8
9  public class RGBSelectorActivity extends Activity
10 {
12     protected void onCreate(Bundle savedInstanceState)
17
18     public void buttonOnClickListener(View v)
31
32     @Override
33     protected void onActivityResult(int requestCode, int resultCode, Intent data)
34     {
35         if (resultCode == RESULT_OK)
36         {
37             SeekBar seekBarR = (SeekBar) findViewById(R.id.seek_r);
38             SeekBar seekBarG = (SeekBar) findViewById(R.id.seek_g);
39             SeekBar seekBarB = (SeekBar) findViewById(R.id.seek_b);
40
41             seekBarR.setProgress(data.getIntExtra("r", 0));
42             seekBarG.setProgress(data.getIntExtra("g", 0));
43             seekBarB.setProgress(data.getIntExtra("b", 0));
44         }
45     }
46 }
```

- `onActivityResult` : capture du résultat
  - `resultCode` : code de retour (`RESULT_OK`, ...)
  - l'**intention** passée en paramètre contient les **données attendues**

Fin !