android

An Open Handset Alliance Project

Ressources et fichiers XML

O.Legrand
G.Seront

Ressources et fichiers XML

- Les ressources suivantes peuvent être extraites du code source:
  - Couleurs
  - Strings
  - Dimensions
  - Images
  - Layouts
  - Descriptions d'animations
  - Thèmes et styles
  - Menus et préférences

developer.android.com/guide/topics/resources/index.htmll

Ressources et fichiers XML

- Séparer les ressources du code source :
  - facilite la conception et les modifications de l'une indépendamment des autres;
    - on peut concevoir et modifier l'interface graphique sans toucher à la couche métier et inversement;
    - travaux réalisés par des personnes ayant des compétences spécifiques (infographistes vs programmeurs);
    - facilite le déploiement dans des environnements différents: langues différentes, devices différents,...;
  - facilite la réutilisation de composants graphiques et logiciels.

developer.android.com/guide/topics/resources/index.html

Les Strings

- Les Strings peuvent être :
  - définies dans un fichier XML;
  - être formatées en utilisant les tags HTML : <b>, <i>, <u>
- Les guillemets et apostrophes sont acceptés:

Exemples :

```
<string name="good_example">"This'll work"</string>
<string name="good_example_2">This\'ll also work</string>

<string name="bad_example">This won't work!</string>
```

developer.android.com/guide/topics/resources/index.html

Lien XML <=> Java

- Chaque ressource XML possédant un id ou un nom génère une entrée dans la classe R.
- Cette classe est générée (pas touche!)
- Une liste par type d'objets:
  - R.string.good_example
  - R.string.good_example_2
  - R.drawable.xxxx
  - R.layout
  - …
- R.string.good_example est un id numérique qui sera utilisé pour récupérer la ressource (voir plus loin)

developer.android.com/guide/topics/resources/index.html

Les Strings

- Format du fichier source : *fichier XML*
- Localisation du fichier:

  res/values/strings.xml
- Nom de la référence à la string :
  - en Java : *R.string.nom_string*
  - en XML : *@[package:]string/nom_string*
  - Syntaxe :

<string name=string_name>string_value</string>

Les Strings

- Exemple de déclaration XML:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="simple_message">Welcome!</string>
    <string name="styled_message">We are <b><i>so</i></b> glad to
    see you.</string>
</resources>
```

developer.android.com/guide/topics/resources/index.html

Les Strings

- Exemple d'utilisation de code en Java :

```
CharSequence message =
    getResources().getString( R.string.styled_message );
TextView tv = (TextView) findViewByID( R.id.text );
 tv.setText( message );
```

- en XML:

```
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:textAlign="center"

    android:text="@string/simple_message"/>
```

developer.android.com/guide/topics/resources/index.html

Les images

- Les formats supportés:
  - *png* (encouragé)
  - *jpg* (accepté)
  - *gif* (déconseillé)
- Localisation du fichier:

  res/drawable/mon_image.png
- Nom de la référence à l'image :
  - en Java : *R.drawable.mon_image*
  - en XML : *@[package:] drawable/mon_image*
- Remarque : la ressource est référencée par le nom du fichier sans son extension.

developer.android.com/guide/topics/resources/index.html

Images : exemple de code Java

```
// Charge un tableau d'images
 private Integer[ ] tableImages =
    { R.drawable.image0, R.drawable.image1, R.drawable.image2 };
 // Renvoie une vue contenant une image
 public View getView(int position) {
    ImageView i = new ImageView( mContext );
    i.setImageResource( tableImages[ position ] );
    i.setAdjustViewBounds( true );
    i.setLayoutParams(
       new Gallery.LayoutParams( LayoutParams.WRAP_CONTENT,
                   LayoutParams.WRAP_CONTENT));
    return i;
 }
```

developer.android.com/guide/topics/resources/index.html

Images : exemple de code XML

```
<ImageView id="@+id/icon"

    android:layout_width="wrap_content"
    android:layout_height="wrap_content "

    android:src="@drawable/mon_image"/>
```

developer.android.com/guide/topics/resources/index.html

Les layouts

- Les interfaces utilisateurs peuvent être décrites dans des fichiers XML:
  - ➢ comme une page web en HTML.
- Un tel fichier peut décrire :
  - la totalité d'un écran;
  - ou une partie de celui-ci.
- Le composant racine doit être de type *View.*
- Le fichier sera compilé et fournira une référence à la ressource de type *View.*

developer.android.com/guide/topics/resources/index.html

Layout : exemple de fichier XML

```
<?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:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:text="@string/body" />

    <EditText
     android:id="@+id/body "
     android:layout_width="fill_parent"
     android:layout_height="wrap_content"
     android:layout_weight="1" />
</LinearLayout>
```

developer.android.com/guide/topics/resources/index.html

Désérialisation des layouts

- Pour être utilisé à l'exécution, un layout doit être désérialisé ;
- 2 façons :

```
//  Dans une activité
this.setContentView(R.layout.main);
```

```
// Dans une classe de type View
LayoutInflater inflater = (LayoutInflater) context
.getSystemService(Context.LAYOUT_INFLATER_SERVICE);

inflater.inflate(R.layout.vue, vueParent);
```

developer.android.com/guide/topics/resources/index.html

Désérialisation des layouts

- La désérialisation est un processus coûteux :
  - Gardez les layouts aussi simples que possible ;
  - Evitez les imbrications inutiles ; *(testez-les avec layoutopt)*
  - Evitez d'utiliser trop de vues dans un layout (max. 80) ;
  - Evitez les imbrications trop profondes (max.10)

developer.android.com/guide/developing/tools/layoutopt.html

Les couleurs

- Les couleurs peuvent être définies dans un fichier XML;
- Le format RGB est utilisé. Le niveau de transparence peut être spécifié (facteur alpha);
- Les formats suivants peuvent être utilisés:
  - *#RGB*
  - *#ARGB*
  - *#RRGGBB*
  - *#AARRGGBB*

developer.android.com/guide/topics/resources/index.html

Les couleurs

- Format du fichier source : *fichier XML*
- Localisation du fichier :

  res/values/couleurs.xml
- Nom de la référence à la couleur :
  - en Java : *R.color.couleur1*
  - en XML : *@[package:]color/couleur1*
- Syntaxe :

```
<color name=color_name>#color_value</color>
```

developer.android.com/guide/topics/resources/index.html

Les couleurs

- Exemple de déclaration XML:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="opaque_red">#f00</color>
    <color name="translucent_red">#80ff0000</color>
</resources>
```

developer.android.com/guide/topics/resources/index.html

Les couleurs

- Exemple d'utilisation de code en Java :

```
// Pour obtenir la valeur entière de la couleur
    int color = getResources().getColor(R.color.opaque_red);
```

- en XML:

```
<TextView android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:textAlign="center"
        android:textColor="@color/translucent_red"
        android:text="Some Text"/>
```

developer.android.com/guide/topics/resources/index.html