Programmation Android

GUI (widgets) et ressources

Jérémy VINET

IE-Concept 2017

GUI (widgets) et ressources

- Les widgets
- Les ressources
- La classe java R
- Ressources

Les widgets

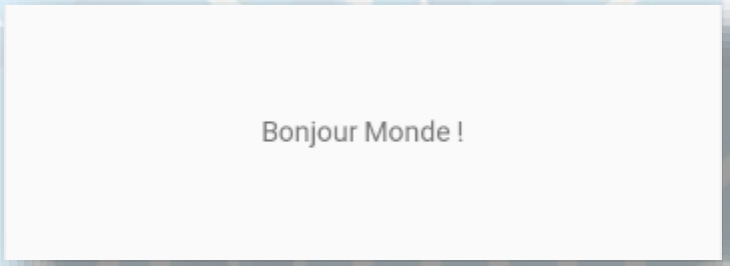
- Généralités sur les widgets
- Les widgets les plus classiques
- Implémentation des widgets en XML et Java
- Gestion des évènements

Généralités sur les widgets

- Un widget est un élément de base qui permet d'afficher du contenu à l'utilisateur ou lui permet d'interagir avec l'application.
- Il existe un grand nombre de widget dans l'API dans d'Android.
- Il est possible de créer soit même des widgets, pour cela il faut soit hériter de la classe View soit d'un widget existant.
- Il est important de donner à chaque widget un attribut id unique qui permettra de le retrouver pour l'utiliser en Java.

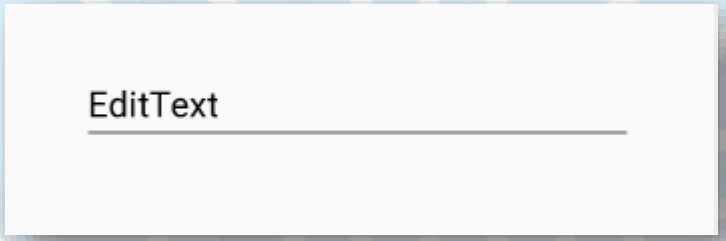
Les widgets les plus simples

- **TextView**
  - Permet d'afficher une chaîne de caractères que l'utilisateur ne peut modifier
  - Configuration de la police d'affichage



Les widgets les plus simples

- **EditText**
  - Permet à l'utilisateur d'écrire des textes.
  - Il s'agit en faite d'un TextView éditable.
  - Il hérite de TextView, ce qui signifie qu'il peut prendre les mêmes attributs que TextView en XML et qu'on peut utiliser les mêmes méthodes en Java.
  - Lorsqu'il a le focus il peut fait apparaitre un clavier pour l'utilisateur

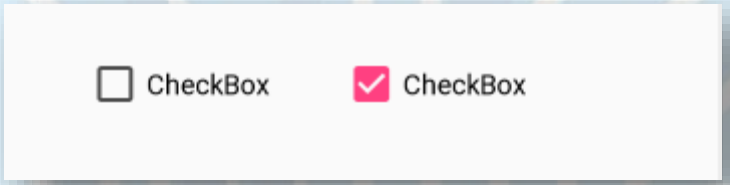


Les widgets les plus simples

- **Button**
  - Un simple bouton cliquable.
  - Il s'agit en faite d'un TextView cliquable.
  - Il hérite de TextView, ce qui signifie qu'il peut prendre les mêmes attributs que TextView en XML et qu'on peut utiliser les mêmes méthodes en Java.

Les widgets les plus simples

- **CheckBox**
  - Une case qui peut être dans deux états : cochée ou pas cochée.
  - Il hérite de Button, ce qui signifie qu'il peut prendre les mêmes attributs que Button en XML et qu'on peut utiliser les mêmes méthodes en Java.



Implémentation de widget XML et Java

- **TextView**
  - Il peut être ajouté directement en XML, dans le fichier layout, avec le code suivant.

```
<TextView
    android:id="@+id/textView"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/textHello"
    android:textSize="8sp"
    android:textColor="#112233" />
```

  - Ou en Java, dans le fichier d'activité, avec l'exemple ci-dessous

```
TextView textView = new TextView( context: this);
textView.setText(R.string.textHello);
textView.setTextSize(8);
textView.setTextColor(0x112233);
```

Implémentation de widget XML et Java

- **EditText**
  - Il peut être ajouté directement en XML, dans le fichier layout, avec le code suivant.

```
<EditText
    android:id="@+id/editText"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:hint="@string/editText"
    android:inputType="textMultiLine"
    android:lines="5" />
```

  - Ou en Java, dans le fichier d'activité, avec l'exemple ci-dessous

```
EditText editText = new EditText( context: this);
editText.setHint(R.string.editText);
editText.setInputType(InputType.TYPE_TEXT_FLAG_MULTI_LINE);
editText.setLines(5);
```

Implémentation de widget XML et Java

- **Button**
  - Il peut être ajouté directement en XML, dans le fichier layout, avec le code suivant.

```
<Button
    android:id="@+id/button"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/textBtn" />
```

  - Ou en Java, dans le fichier d'activité, avec l'exemple ci-dessous

```
Button button = new Button( context: this);
button.setText(R.string.textBtn);
```

Implémentation de widget XML et Java

- **CheckBox**
  - Il peut être ajouté directement en XML, dans le fichier layout, avec le code suivant.

```
<CheckBox
    android:id="@+id/checkbox"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/checkbox"
    android:checked="true" />
```

  - Ou en Java, dans le fichier d'activité, avec l'exemple ci-dessous

```
CheckBox checkBox = new CheckBox( context: this);
checkBox.setText(R.string.checkbox);
checkBox.setChecked(true);
if(checkBox.isChecked()) {
    // TODO
}
```

Gestion des évènements

- Il existe plusieurs façons d'interagir avec une interface graphique
  - Cliquer sur un bouton
  - Entrer un texte
  - Cocher une case, …
- Pour réagir à ces évènements, il faut utiliser un listener, un objet qui va détecter l'évènement.
- Un listener est une interface qui oblige de redéfinir des méthodes de callback qui seront appelées lorsqu'un évènement se produit

Gestion des évènements

- **De façon générale :**
  - Une interface est associée à chaque type d'évènements.
  - On observe les évènements en connectant des listeners.
  - Pour cela, on utilise les méthodes « setOn*Listener ».
  - Un évènement entraîne l'appel d'une méthode de l'interface sur l'objet que vous avez connecté à la vue.

- **Trois méthodes pour écouter les évènements**
  - Implémentation avec une classe classique.
  - Utilisation d'une classe interne.
  - Utilisation d'une classe anonyme.

Gestion des évènements *(exemple)*

- Une classe implémente View.OnClickListener et définit onClick.
- Utilisation de setOnClickListener pour lier le bouton.

```
public class MainActivity extends Activity implements View.OnClickListener {

    @Override
    public void onClick(View view) {
        // TODO: react at click
    }

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

        Button button = new Button( context: this);
        button.setText(R.string.textBtn);
        button.setOnClickListener(this);

    }
}
```

Gestion des évènements *(exemple)*

- Utilisation d'une classe interne qui implémente View.OnClickListener et définit onClick.
- Utilisation de setOnClickListener pour lier le bouton.

```
public class MainActivity extends Activity {

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

        Button button = new Button( context: this);
        button.setText(R.string.textBtn);
        button.setOnClickListener(new MyOnClickListener());

    }

    class MyOnClickListener implements View.OnClickListener {
        @Override
        public void onClick(View v) { // TODO: react at click
        }
    }
}
```

Gestion des évènements *(exemple)*

- Utilisation d'une classe anonyme qui implémente View.OnClickListener et définit onClick.
- Utilisation de setOnClickListener pour lier le bouton.

```
public class MainActivity extends Activity {

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

        Button button = new Button( context: this);
        button.setText(R.string.textBtn);
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) { // TODO: react at click
            }
        });
        setContentView(button);
    }
}
```

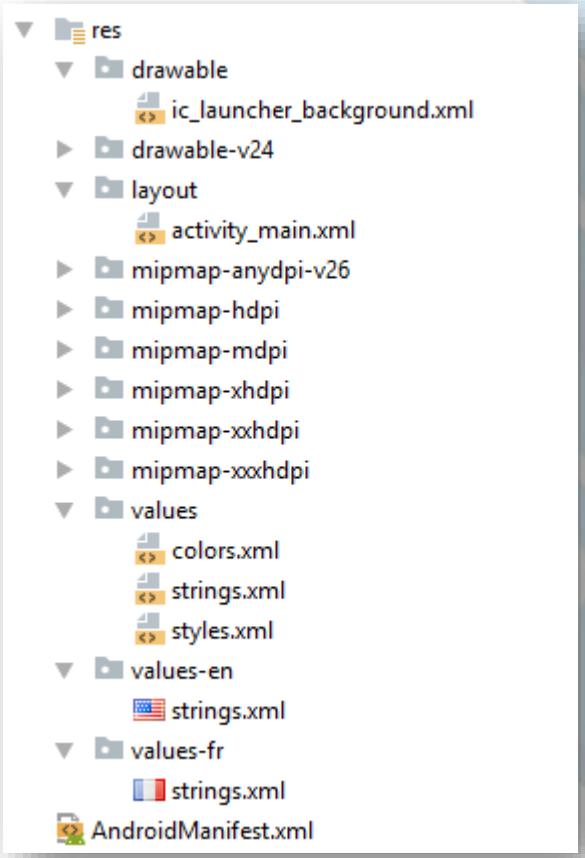
Les ressources

- Généralités
- Les références
- La classes Ressources
- Les types de ressources
- Internationalisation et dimensions

Généralités

- **Les ressources permettent d'externaliser :**
  - Les images.
  - Les chaînes de caractères.
  - Les modèles d'interfaces utilisateur.
  - Les animations …
- **De fournir des ressources différentes en fonction de la configuration :**
  - Taille de l'écran.
  - Orientation de l'écran.
  - Langue …
- **De maintenir ces éléments sans connaître Java.**

Généralités

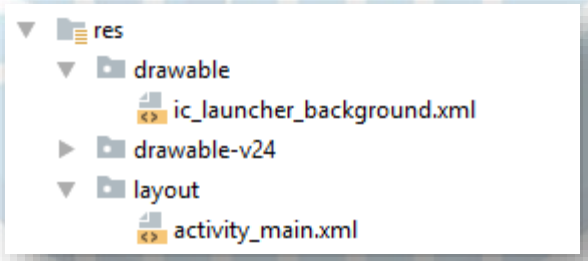


- Elles sont dans le répertoire res.
- Elles sont organisées dans des répertoires.
- Elles contiennent des images, fichiers XML, …
- L'arborescence en dossier permet une meilleure lisibilité et maintenabilité.

Généralités

- Le framework Android génère et maintient à jour automatiquement une classe R contenant les identifiants (valeur entière) des ressources.

- Ils permettent de désigner les ressources dans le code Java.
  - Par exemple, le fichier activity_main.xml a comme identifiant R.layout.activity_main



Les références

- Voici la syntaxe pour désigner une ressources dans un document XML
  - @[<package_name>:]<resource_type>/<resource_name>

    - <package_name> est le nom du paquet (peut ne pas être mentionné).
    - <ressource_type> est le type de la ressource.
    - <ressource_name> est le nom du fichier qui contient la ressource (sans l'extension) ou la valeur de l'attribut android:name d'un élément XML (pour les valeurs simples).

```
<ImageView
    android:id="@+id/imageView"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:contentDescription="@string/textHello"
    app:srcCompat="@android:drawable/btn_star_big_on" />
```

Les références

- Pour accéder à une ressources en Java on utilise la classe R
  - R.<resource_type>.<resource_name>
    - <ressource_type> est le type de la ressource (string, id, …).
    - <ressource_name> est le nom du fichier qui contient la ressource (sans l'extension) ou la valeur de l'attribut android:name d'un élément XML (pour les valeurs simples).

```
ImageView image = (ImageView)findViewById(R.id.image);
image.setImageResource(R.drawable.ic_launcher_background);
```

La classe Ressources

- **Permet l'accès aux ressources répertoriées dans R**
- **On obtient une instance de cette classe par getRessources() de l'activité**
- **Principales méthodes de la classe Resources** (le paramètre int est un identifiant défini dans R de la forme R.type.nom):
  - Boolean getBoolean(int)
  - Int getInteger(int)
  - String getString(int)
  - Int getColor(int)
  - Float getDimension(int)
  - Drawable getDrawable(int)

```
String titre = getResources().getString(R.string.app_name);
```

La classe Ressources

- **Accès aux ressources dans l'application**
  - Mise en place de l'interface principale

```
setContentView(R.layout.activity_main);
```

  - Accès direct à une valeur ou à une ressource

```
String titre = getResources().getString(R.string.app_name);
```

```
float dim = getResources().getDimension(R.dimen.layout_padding);
```

Les types de ressources

- **Les différents types de ressources :**
  - Layouts
  - Chaînes de caractères
  - Menus
  - Animations
  - Images
  - Couleurs
  - Styles
  - Préférences
  - Dimensions
  - …

Internationalisation

- **Il est possible de définir des valeurs pour plusieurs langues :**
  - values/strings.xml contient le texte en anglais des chaînes.
  - Values-fr/strings.xml contient le texte en français des chaînes.
  - values-es/strings.xml contient le texte en espagnol des chaînes.

- **Si title n'est défini qu'en français et en anglais et que l'on fait référence à R.string.title alors :**
  - Si le terminal n'est pas configuré en français, Android charge le texte qui se trouve dans le fichier values/strings.xml
  - Si le terminal est configuré en français, Android charge le texte qui se trouve dans le fichier values-fr/strings.xml
  - Si le terminal est configuré en italien, Android charge le texte qui se trouve dans le fichier values/strings.xml

Les dimensions

- Pour faciliter la prise en charge des différentes tailles d'écran, il est préférable d'utiliser des références vers les dimensions pour vos widgets et layout :

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center_vertical|center_horizontal"
    android:orientation="vertical"
    android:padding="@dimen/layout_padding">
```

Les dimensions

- Les dimensions sont définies dans le fichier dimens.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <dimen name="layout_padding">40dp</dimen>
</resources>
```

- Il est également possible d'accéder aux dimensions en Java :

```java
float dimension = getResources().getDimension(
        R.dimen.layout_padding);
```

Chargement des différentes ressources

- **Il est possible de définir des ressources en fonction de l'orientation du terminal, de la résolution, de la taille de l'écran, de la version de l'API, etc :**
  - values/dimens.xml contient les dimensions par défaut.
  - values-w820dp/dimens.xml contient les dimensions pour les écrans dont la largeur est supérieure à 820dp.
  - layout/activity.xml contient la version par défaut d'un layout.
  - layout-land/activity.xml contient la version paysage de ce même layout.

- **Remarques :**
  - Le système va aller chercher la bonne ressource automatiquement.
  - Cela est valable pour tous les types de ressources.
  - Quasiment tous les types de ressources peut avoir des versions différentes

Ressources

- https://developer.android.com/guide/topics/resources/index.html
- https://developer.android.com/guide/topics/manifest/manifest-intro.html
- https://developer.android.com/guide/topics/ui/index.html
- https://developer.android.com/