Cours de programmation sous Android

Responsable de matière : Monsieur Houssem Lahiani.

Chapitre 3: Les vues sous android

Plan du chapitre

Chapitre 3 : Les vues sous Android

Dans cette partie, on va vous donner quelques exemples de composants importants et fréquemment utilisées.

Button (http://developer.android.com/reference/android/widget/Button.html) : Un bouton cliquable.
CheckBox (http://developer.android.com/reference/android/widget/CheckBox.html) : Une checkbox.
EditText (http://developer.android.com/reference/android/widget/EditText.html) : Un champ de texte éditable.
DatePicker (http://developer.android.com/reference/android/widget/DatePicker.html) : Sélection de dates.
RadioButton (http://developer.android.com/reference/android/widget/RadioButton.html) : Représente les boutons radios.
Toast (http://developer.android.com/reference/android/widget/Toast.html) : Un pop up qui s'affiche sur l'écran.
ImageButton (http://developer.android.com/reference/android/widget/ImageButton.html): Une image qui se comporte comme un bouton.

I. TextView

android:textColor : Couleur du texte. Toute les couleurs utilisées sont déclarées dans le fichier colors.xml et utilisées à l'aide de la syntaxe @color/nom_de_la_couleur.

android:paddingTop : Marge interne du haut.

android:textSize : Définie la taille du text.

II. EditText

android:inputType : Type du texte qui sera saisie dans la zone d'édition (adresse email pour la première zone d'édition et mot de passe pour la seconde).

III. Les toasts

- Le moyen le plus simple pour afficher un message à l'utilisateur
- Permet d'afficher un texte momentané qui pourra durer plusieurs seconde mais pas plus
- Ce code la affiche un texte momentané « msg msg » :

IV. Utilisation des vues, les boutons et les toasts par exemple

Comme dans le précédent tutoriel, avec Eclipse, créer un nouveau projet. Nous allons commencer par éditer le fichier "activity_main.xml" situé dans le dossier « res/layout ».

Effacer le contenu de ce fichier et mettez ce code la :

```
 <?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">
```

```
    <Button
            android:id="@+id/buttonToast"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Click here !" />
</LinearLayout>
```

Nous avons supprimé le code existant qui contient un TextView et nous avons mis un code qui contient un Button.

Le bouton a un id (identifiant) (`android:id="@+id/buttonToast"`), qui sera utilisé pour l'appeler dans votre code source java.

La largeur de notre bouton va occuper toute la largeur du conteneur (le conteneur est de type LinearLayout)(`android:layout_width="fill_parent"`) mais pour la hauteur il va occuper juste l'espace nécessaire pour contenir le texte qui s'affichera sur le bouton (`android:layout_height="wrap_content"`).

Le texte qui s'affichera sur le bouton est « Click here ! » (`android:text="Click here !"`).

Maintenant nous nous occuperons du code java :

Ouvrez le fichier "MainActivity.java" situé sous le package sous le dossier "src", voua allez trouver ce code :

```
import android.app.Activity;
import android.os.Bundle;
 public class MainActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
         super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);} }
```

Actuellement, si nous lançons notre application et vous cliquez sur le bouton, rien ne se passera. Pour résoudre ce problème, nous allons ajouter du code pour définir l'action et c'est là que les Listeners entrent en jeu.

Qu'est-ce qu'un Listener?

Un écouteur (Listener), comme son nom l'indique, écoute une source jusqu'à son utilisation.

Pour une action à effectuer, il est nécessaire qu'un émetteur envoie un signal à l'écouteur qui est déclenché.

Action: Cliquez sur un bouton.

Listener: Écoutez lorsque le bouton est cliqué.

Il n'y a pas plus simple que cela, nous verrons un plus tard dans les exemples pour mettre en œuvre un Listener.

Nous ne pouvons pas appeler directement notre bouton dans notre activité, nous devons récupérer son ID à partir du fichier XML.

Chaque activité (classe java qui hérite de la classe Activity) a une méthode findViewById qui sert à appeler un élément dans le layout (l'interface graphique), avec un paramètre l'identifiant de l'objet que vous souhaitez appeler. Il est stocké dans la ResourcesManager qui est appelé R (qui se trouve sous le dossier gen et qui contient toutes les ressources de votre projet, les ressource du projet sont situées sous le dossier Res).

Pour appeler le bouton situé dans le layout xml il faut mettre ce code :

```
Button myButton = (Button)findViewById(R.id.buttonToast);
```

Un bouton ne peut pas faire l'action suite à un clics par lui-même, il a besoin d'un Listener, nous devons donc utiliser la méthode setOnClickListener. Quand un bouton est cliqué, le listener réagit et exécute le code de la méthode onClick. Nous devons donc ajouter ces lignes de code :

```
myButton.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
    // TODO Auto-generated method stub
  }});
```

Nous allons ajouter un court message qui s'affiche lorsque vous cliquez sur le bouton, le message est affiché via une classe nommée Toast (un message pop up qui s'affiche sur l'écran pour une courte durée).

La classe Toast possède une méthode statique appelé makeText, qui prend comme paramètres le contexte (notre activité actuel), le message à afficher et enfin une période de temps.

La période de temps peut être Toast.LENGTH_SHORT ou Toast.LENGTH_LONG.

Donc à l'intérieur de la méthode onClick mettez ce code :

```
Toast.makeText(MainActivity.this, "Hello !", Toast.LENGTH_SHORT).show();
```