TP : Menu glissant

1. Introduction

Android n'a cessé d'évoluer au cours des ses différentes versions, notamment en ajoutant de nouveaux composant d'interactions. À partir de la version Honeycomb, c'est à dire Android 3, sont apparues les ActionBar et avec elles, une multitude de possibilités dont, les menus glissant.

2. C'est quoi un menu glissant ?

Un menu glissant est un panel regroupant l'ensemble des options de navigation d'une application. Dans la plupart des cas, le menu est caché et on le fait apparaître en faisant « glisser » le doigt depuis le bord gauche de l'écran vers la droite. Le menu s'affiche au-dessus de tous les composants de l'application. On peut aussi déclencher le menu en appuyant sur l'icône de l'application se trouvant en haut à gauche de l'ActionBar.

3. Pourquoi as-t-on besoin d'un menu glissant ?

Le menu glissant nous permet une meilleure gestion de l'espace, l'ensemble des fonctionnalités de l'application y sont regroupées et de ce fait, on laisse plus de place au contenu. Une facilité d'accès aux options de l'application, en faisant un simple mouvement du doigt, de gauche vers la droite.

4. C'est parti !

La création d'un menu glissant passe par les étapes suivantes :

- Mise en place du layout DrawerLayout.
- Création de la liste des options.
- Création du menu glissant en JAVA.
- Gestion des événements sur les options.

Les liens permettant de télécharger les archives du projet se trouvent sur la dernière page. Le début du projet n'est en fait que le projet final avec des sections commentées. Vous pourrez soit, partir d'un projet vide et suivre les étapes décrites ci-dessous, où récupérer directement l'archive et suivre les étapes afin de comprendre qui fait quoi.

4.1. Mise en place du DrawerLayout

Pour ajouter un menu glissant à votre application, vous devez utiliser un layout de type Drawer comme racine de votre fichier XML. Dans le DrawerLayout, vous rajouterez une première vue qui va afficher le contenu principal de votre activité quand le menu glissant est fermé. Et une seconde vue, qui elle, va contenir les éléments, options...de votre menu glissant.

Dans l'exemple ci-dessous, le DrawerLayout possède deux enfants :

- Un LinearLayout qui contient une TextView, c'est le contenu principal de l'activité.
- Une ListeView qui est l'ensemble des éléments du DrawerLayout.

```
<!-- Le layout global qui contiendra notre menu glissant et le layout de l'activité -->
 <android.support.v4.widget.DrawerLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:id="@+id/slider_layout"
  android:layout_height="match_parent"
  android:layout_width="match_parent">

  <!-- Contenu principal -->
   <LinearLayout
    android:layout_height="match_parent"
    android:layout_width="match_parent" >
    <TextView
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:text="Contenu principal"
      android:padding="22dp"/>
  </LinearLayout>

  <!-- La liste de tous les éléments du menu glissant -->
   <ListView android:id="@+id/list_elements"
    android:layout_width="240dp"
    android:layout_height="match_parent"
    android:layout_gravity="start"
    android:choiceMode="singleChoice"
    android:divider="@android:drawable/divider_horizontal_textfield"
    android:dividerHeight="2dp"
    android:background="#9d9d9d" />

</android.support.v4.widget.DrawerLayout>
```

Plusieurs points sont à remarquer :

- Le premier enfant du DrawerLayout doit être votre vue principale.
- On doit spécifier un identifiant pour la ListView, dans mon cas c'était list_elements.
- Votre ListView doit spécifier son ***layout_gravity*** notamment pour la gestion des écritures droite vers la gauche.
- La ListView doit spécifier aussi sa largeur, dans notre cas c'était 240dp.

Évidemment si vous en avez l'envie, vous pourrez même utiliser une RecyclerView à la place de la ListView. Cela permettra des layout mieux optimisés si vous avez une tonne d'options :)

4.2. Mise en place de la structure des éléments du menu glissant

Une fois qu'on a définit la structure du DrawerLayout, il nous faudrait nous atteler à définir la structure générale des éléments de la ListView. Même si dans notre cas, ce n'est pas très complexe, rien ne vous empêche de créer des layouts qui le sont.

Dans votre dossier layout, créer un fichier XML, nommez le par exemple : *sliding_menu_item*. Et c'est au sein de ce fichier que nous allons décrire la structure général des éléments de la ListView du menu glissant. Dans mon cas, j'ai fais quelque chose d'assez simple, je vous laisse y jeter un coup d'œil :

```
<!-- Structuration de l'élément qui sera utilisé dans la listview du menu glissant -->
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="#dddddd"
    android:textSize="18sp"
    android:padding="20dp">

</TextView>
```

J'ai utilisé une TextView à la quelle j'ai mis un fond gris clair, une taille de police de 18sp et un padding de 20dp, rien de bien folichon:)

4.3. Implémentation du menu glissant dans la classe MainActivity

Dans cette partie nous allons voir comment initialiser les différents éléments du menu glissant et comment y ajouter la gestion d'événements afin de lancer une activité à chaque fois qu'on clique sur une option. Les éléments seront les différents Masters informatique, dont on affichera le nom dans la nouvelle activité.

4.3.1 Méthode onCreate()

Dans la méthode onCreate nous allons juste nous charger d'initialiser notre ListView, et attacher un event listener à chaque élément de la liste. Comme je l'ai dis dans l'introduction l'ActionBar a été introduite à partie de l'API level 11, et de ce faite là, et pour assurer une compatibilité avec les version antérieurs, j'ai décidé de ne pas utiliser l'ActionBar, et de juste implémenter le sliding menu.
Dans ma méthode onCreate(), je fais appel à une fonction populateListView() dans la quelle on fera ce qui a été énoncé ci-dessus.

4.3.2 Fonction populateListView()

Dans un premier temps nous allons définir la liste des éléments qui composent notre ListView, pour cela on va utiliser un tableau de String.

```
final String[] elements = {"E-Services", "IAGL", "MOCAD", "TIIR", "MIAGE", "IVI"};
```

Ensuite nous allons utiliser une instance de l'ArrayAdapter afin de lier nos éléments au layout qu'on a créé dans la section précédente, i.e : *sliding_menu_item*.

Il faut savoir que l'ArrayAdapter s'attend à recevoir un layout qui ne contient qu'une simple TextView, d'où la petit tour de magie qu'il fait pour lier les éléments au layout. Néanmoins, si vous souhaitez utiliser des structures plus complexes, il vous faudra ***@Override*** la fonction ***getView(int, View, ViewGroup)*** afin qu'elle retourne le type de vue que vous souhaitez.

Pour plus de détails sur l'ArrayAdapter n'hésitez pas à aller sur la documentation officielle Android.

Hop ! Retour au code ! Voici donc la partie de mon code permettant d'utiliser l'ArrayAdapter.

```
ArrayAdapter<String> elementsAdapter = new ArrayAdapter<>(
                this,                       // Le contexte ( this = MainActivity )
                R.layout.sliding_menu_item, // Le layout d'un élément de la ListView
                elements);                  // La liste des éléments de la ListView
```

Une fois qu'on a fait tout ça, il ne nous reste plus qu'a configurer la ListView pour qu'elle utilise notre Adapter, pour ce faire il faut procéder ainsi :

```
ListView listView = (ListView) findViewById(R.id.list_elements);
listView.setAdapter(elementsAdapter);
```

Dans la dernière phase nous allons ajouter des écouteurs d'événements aux éléments de la ListView, ces derniers nous permettront de lancer une activité à chaque fois qu'on cliquera sur un élément.

```
listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {

    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {

      // On cree un intent explicite pour lancer MastersActivity
      Intent intent = new Intent(MainActivity.this, MastersActivity.class);

      // On ajoute un extra pour transferer des données entre les acitivités
      intent.putExtra(ACTIVITY_MESSAGE, elements[position]);

      // On lance l'intent
      startActivity(intent);
    }
  });
```

On fera appel à la méthode *setOnItemClickListener()* de l'objet ListView, qui prend en paramètre une instance de *AdapterView.OnItemClickListener.* À chaque clique sur un item un intent explicite est lancé. L'intent contient le nom d'un Master informatique qu'on a ajouté grâce à la fonction *putExtra()*, qu'on affichera dans l'activité nouvellement lancée.

Vous remarquerait que dans les paramètres de la fonction *onItemClick()*, on à un entier nommé position, ce dernier nous permet de savoir sur quel élément on a cliqué. A partir de ce paramètre je récupère le nom du Master à partir du tableau éléments créé précédemment.

4.4. Création de la nouvelle activité MastersActivity

Dans le package contenant vos différentes activités créer une classe nommée **MastersActivity**, faites en sorte qu'elle étend la classe **AppCompatActivity**, et **@Override** la méthode *onCreate()*.

Vous devriez obtenir quelque chose qui ressemble à ça :

```
public class MastersActivity extends AppCompatActivity {

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

  }
}
```

4.4.1. Création d'un layout pour MastersActivity

Nous allons créer un structure assez simple pour l'activité Masters, ça sera un simple *LinearLayout*, avec une *TextView* dedans.

Si vous avez utilisé la création d'activité automatique d'Android Studio, vous aurez juste à éditer votre fichier XML, sinon, dans le dossier Layout, créer un fichier XML que vous nommerez **masters_activity**, et vous y insérerez le code suivant :

```
<?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:id="@+id/activity_es"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:padding="16dp"
  tools:context="com.wassim.slidingmenu.MastersActivity">

  <TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/master_name" />

</LinearLayout>
```

Il ne faut pas oublier de donner un identifiant à votre TextView, sans cela vous ne pourrez récupérer une instance de cette dernière dans la classe MastersActivity.

4.4.2. Réception de l'intent et affichage du texte

Retour sur la classe MastersActivity, où nous allons récupérer l'intent lancé par la MainActivity, récupérer le texte qu'il contient et l'afficher.

À votre méthode *onCreate()*, rajoutez la portion de code suivante :

```
// On récupère l'intent
Intent intent = getIntent();

// On recupère le textview de l'activity
TextView masterChoosed = (TextView) findViewById(R.id.master_name);

//On set le text du textview
masterChoosed.setText(String.format("Vous avez choisi le master : %s",
intent.getStringExtra(MainActivity.ACTIVITY_MESSAGE)));
```

On récupère une instance de l'intent avec la fonction *getIntent()*. On récupère une instance de la TextView grâce à la fonction *findViewById()*, et enfin on attribue le texte récupéré grâce à la fonction *getStringExtra()* à la TextView.

Vous remarquerez l'utilisation d'une variable static ACTIVITY_MESSAGE dans la fonction *getStringExtra()*, c'est une variable que j'ai défini dans la MainActivity pour me permettre de récupérer la clé du message de façon plus facile.

Voilà on est arrivé au terme du TP, maintenant vous savez créer un menu glissant compatible avec toutes les versions d'Android ou presque !

5. Conclusion

L'utilité d'un drawer n'est plus à démontrer, il est quasi indispensable de nos jour, et toutes les application ou presque en possède un. Il permet de regrouper plusieurs fonctionnalités d'une application, et de laisser place au contenu.

Liens utiles

- La documentation Google sur les DrawerLayout.
- La documentation Google sur les ActionBar.
- Un tutoriel du quel je me suis inspiré.
- Début du projet – Fin du projet.