Menu glissant

**David Guillon (FA)  
29/12/2018**

# Résumé

Présentez ici en 10 lignes maxi de quoi parle le TP, et ce qu’on apprend à y faire. Par exemple :

*Un menu glissant permet de naviguer entre les différentes « pages » ou fragments de notre application. La plupart du temps, le menu glissant est caché sur l’activité courante et on peut l’activer en cliquant sur l’icône du menu, ou en faisant glisser son doigt de gauche à droite. (Pour un menu à gauche.)*

*Dans ce TP, nous verrons comment créer un menu glissant, ajouter des icônes et naviguer entre les divers fragments de notre application grâce à celui-ci.*

# Pré-requis

* *Savoir programmer une application Android avec plusieurs activités qui s’appellent et qui appellent d’autres applications*
* *Maîtriser les layouts*

# Code source

Code source **initial** : http://.... (URL…)

Code source **final** : http://... (URL…)

# Explications du TP

Nous allons voir étape par étape comment réaliser un menu glissant.

Au cours de ce TP, vous aurez la possibilité de récupérer le code correspondant à l’avancement réalisé entre chaque étape via git.

Pour ce faire ouvrez un terminal à la racine du projet et faites un git checkout sur les branches que je vous indiquerai pendant le TP.

**Étape 1 : Installer le projet.**

*Rendez-vous à l’adresse indiquer pour le code source initial et tapez :*

*git clone https://*

*Ouvrez le projet sur Android Studio.*

**Étape 2 : Créer les layout.**

Pour cette étape, vous allez commencer par faire :

git checkout create\_layout

depuis la racine de votre projet.

La création des fichiers xml qui composeront notre application étant assez fastidieuse et un peu répétitive, je vais me contenter de vous expliquer ce qu’on fait au cours de cette première étape.

a. L’activité principale (activity\_main.xml)

Pour faire notre menu, nous avons besoin d’un « navigation drawer », c’est un panneau d’interface utilisateur qui nous montre le menu de navigation. Quand il n’est pas utilisé, il est caché, mais il apparaît quand on swipe du côté gauche ou droit de l’écran, ou quand on appuie sur le bouton menu.

Pour ce faire nous avons besoin d’un layout pour créer un menu, le DrawerLayout.

Ce qui ce trouve dans notre RelativeLayout contiendra notre contenu.

Pour l’élément ListView, c’est une partie de notre menu.

On définit ici sa hauteur et sa longueur, on lui donne un @id pour nous permettre de l’utiliser plus tard.

On peut ici personnaliser la couleur d’arrière plan de notre menu.

On indique que notre menu est une liste à choix unique.

Enfin n’oubliez pas la propriété layout\_gravity. Cette propriété indique de quel côté notre menu glissant s’ouvrira :

*- start : du côté gauche.*

*- end : du côté droit.*

*b.* item\_sliding\_menu.xml

Cette xml et celui qui va modéliser les différents éléments de notre menu.

Comme c’est un menu, nous commençons par utiliser un linearLayout.

Chaque item du menu sera composé d’une image, dont nous pourrons modifier les proportions et d’un texte.

c. Les fragments.

Comme vous pouvez le voir, nous avons ici trois fragments ; chaque fragment est implémenté de la même façon.

C’est un layout qui contient un texte simple.

**Étape 3 : Créer un adapter pour le menu glissant.**

**a. Créer une classe pour gérer les item du menu**

Dans votre dossier java/lille.univ.slidingmenu, créer un nouveau package ***model***, puis dans ce package créez la classe public ***ItemSlidingMenu*** qui contiendra deux attributs :

- int imageId (l’id d’une image)

- String title (le titre de notre item)

Ensuite générer les getters/setters, puis le constructeur de la classe.

**b. Créer l’adaptateur.**

Créer un nouveau package adapter puis la classe SlidingMenuAdapter.

Avant de rajouter de commencer cette classe, voici les imports que vous pouvez rajouter :

import android.content.Context;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.ImageView;  
import android.widget.TextView;  
import lille.univ.slidingmenu.R;  
import lille.univ.slidingmenu.model.ItemSlideMenu;  
  
import java.util.List;

Votre classe doit étendre la classe BaseAdapter

Faites un clique droit sur BaseAdapter, puis « implements method ».

Dans la petite fenêtre, sélectionnez tout de android.widget.Adapter.

Avant de modifier ses méthodes, nous allons créer deux attributs :

- Context context

- List<ItemSlideMenu> listItem

Générez le constructeur.

Maintenant nous allons modifier les méthodes que nous avons importé avec la classe BaseAdapter.

Modifiez la méthode getCount() pour qu’elle retourne la taille de notre attribut listItem.

Modifiez la méthode getItem() pour qu’elle retourne la position de l’argument de cette méthode dans listItem.

Modifiez getItemId() qui retourne simplement son argument.

Insérez le code suivant dans à la place de la méthode getView() ;

public View getView(int position, View view, ViewGroup viewGroup) {  
 View v = View.*inflate*(context, R.layout.*item\_sliding\_menu*, null);  
 ImageView img = (ImageView) v.findViewById(R.id.*item\_img*);  
 TextView tv = (TextView) v.findViewById(R.id.*item\_title*);  
  
 ItemSlideMenu item = listItem.get(position);  
 img.setImageResource(item.getImgId());  
 tv.setText(item.getTitle());  
  
 return v;  
}

View.inflate()

permet de générer une vue à partir d'une ressource XML.

Cette méthode pratique englobe la classe LayoutInflater, qui offre une gamme complète d'options pour visualiser l'inflation.

On va chercher les éléments associés à l’image et au texte (ImageView et textView).

On récupère ensuite l’item qui nous intéresse, on affecte l’image et le titre et on retourne la vue.

Avant de passer à l’étape suivante, vous pouvez si vous le souhaitez faire un git checkout -f create\_adapter\_for\_sliding\_menu

**Étape 4 : Créer les classes pour les fragments.**

Pour commencer, nous allons créer un package fragment.

Ensuite, vous pouvez créer une classe Fragment1 qui étendra android.app.Fragment.

Dans cette classe, nous allons créer un constructeur vide et overrider la méthode onCreateView() comme ceci :

@Override  
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {  
 View rootView = inflater.inflate(R.layout.*fragment1*, container, false);  
 return rootView;  
}

Cette méthode va générer la vue des fragments à partir des fragments xml que nous avons créer plus tôt.

Créez la classe Fragment2 et Fragment3.

**Étape 5 : Créer la classe principale de l’activité.**

Pour cette étape je vous invite à git checkouter la branche create\_activity\_class.

Cette classe étant un peu plus lourde que les autres.

# Informations complémentaires

Mettez ici une webographie commentée par exemple :

* [*https://openclassrooms.com/courses/developpez-une-application-mobile-multi-plateforme-avec-ionic*](https://openclassrooms.com/courses/developpez-une-application-mobile-multi-plateforme-avec-ionic) *: permet à un débutant de comprendre MCV et de créer sa première application*
* [*https://kotlinlang.org/docs/tutorials/kotlin-android.html*](https://kotlinlang.org/docs/tutorials/kotlin-android.html) *: apprenez à développer en Kotlin. Il faut bien connaitre Android pour cela.*

Vous pouvez aussi indiquer d’autres infos, par exemple des bonnes pratiques, des trucs et astuces, des pièges à éviter, des livres à consulter, etc….