TP03 - Fragments

Création d'une application pour visualiser des articles d'actualités.

Objectif:

- Utiliser des fragments pour créer une interface utilisateur flexible en dépendance de l'appareil de l'utilisateur.

Exercices

1- Créer une classe Ipsum qui contiendra deux tableaux statique avec le titre des articles et les articles en question:

```
static String[] Headlines = {
    "Article One",
    "Article Two"
};

static String[] Articles = {
    "Article One\n\nExcepteur pour-over occaecat …",
    "Article Two\n\nVinyl williamsburg non velit…"
};
```

2 - Créer une classe HeadLinesFragment qui sera une sous classe de ListFragment (ListFragment est un type de Fragment ou on peut organiser les éléments grace a une liste).

3 - Faire un *Override* de la méthode *onCreate(Bundle savedInstance)* de la classe HeadLinesFragment pour lui donné une liste a afficher:

- On utilise un layout de liste par défaut de android:

```
int layout = android.R.layout.simple_list_item_activated_1;
```

- On appelle la méthode *setListAdapter(ListAdapter adapter)* de la classe ListFragment pour fournir du contenu a notre liste;

```
setListAdapter(new ArrayAdapter<String>(getActivity(), layout, Ipsum.Headlines));
```

4 - Créer une interface pour que au clique sur un des titres de notre liste, on puisse communiquer avec l'autre Fragment:

```
public interface OnHeadlineSelectedListener {
    /** Ceci sera appeller quand on clique sur un élément de notre liste */
    public void onArticleSelected(int position);
}
```

-Ne pas oublié de déclarer un objet OnHeadlineSelectedListener dans notre classe:

```
OnHeadlineSelectedListener mCallback;
```

5 - La méthode onListItemClick(ListView l, View v, int position, long id) doit être implementer dans notre classe HeadLinesFragment, vu que c'est une interface de la classe ListFragment. Sur cette méthode, on va récupérer la position de l'élément cliquer et on va appeler la méthode *onArticleSelected(int position)* de l'objet *mCallback.* Aussi, on va appelé la méthode setItemChecked pour que notre élément soit souligné :

```
getListView().setItemChecked(position, true);
```

6 - Créer un layout *article_view.xml* ou on va afficher notre article. Un simple LinearLayout avec comme enfant un TextView est suffisant. Ne pas oublié de donner un id au TextView.

7 - Créer une classe ArticleFragment, sous classe de Fragment, qui aura comme variable un int mCurrentPosition ou on gardera quelle est article afficher et une variable View ou on va garder une reference de la View du Fragment.

8 - Faire un *Override* de la méthode *onCreateView()* pour injecter notre layout créer précédemment.

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,  Bundle
savedInstanceState) {
        //Si l'activité est recréer (par exemple, un changement d'orientation), on récupère l'article
        //sélectionner précédemment qui a été sauver dans la méthode onSaveInstanceState().
        //Ceci est nécessaire si on montre les deux fragment en même temps (sous tablette)
        if (savedInstanceState != null) {
                mCurrentPosition = savedInstanceState.getInt(ARG_POSITION);
        }

        //  et on inject le layout a notre objet View déclarer comme variable de la classe
        rootView = inflater.inflate(R.layout.article_view, container, false);

        //On renvoi notre View avec le layout injecter
        return rootView;
}
```

9 - Faire un Override de la méthode *onSaveInstanceState(Bundle outState)* pour sauver la position dans le cas ou le Fragment est détruit (un changement de rotation par exemple). Ne pas oublié d'appeler super!

```
@Override
public void onSaveInstanceState(Bundle outState) {
    super.onSaveInstanceState(outState);

    // Save the current article selection in case we need to recreate the fragment
    outState.putInt(ARG_POSITION, mCurrentPosition);
}
```

NOTE : Déclarer une clé statique "ARG_POSITION" pour être sur de ne pas ce tromper.

10 - Implementer une méthode *updateArticleView(int position)* en public pour être appeler par notre activité pour changer le contenu du *TextView*. On récupère l'objet *TextView* en utilisant la méthode *findViewById()* de la *View* déclarer précédemment et on lui change son texte avec le tableau *Articles* déclarer dans la classe *Ipsum*. On sauve la position à la fin de la méthode.

11 - Faire un *Override* de la méthode *onStart()* pour qu'on puisse donné a notre fragment le texte a afficher. On peut récupérer les arguments donné au *Fragment* grace a la méthode *getArguments()*. Si ce dernier est different de null (ça veut dire qu'on lui donnera un bundle dans notre activité) on peut appeler la méthode *updateArticleView(int position)* avec la position récupérer dans notre Bundle. On peut aussi avoir la position garder dans notre variable (si le fragment est recréer), alors on peut appeler la méthode updateArticleView avec la position garder.

12 - Maintenant qu'on a nos deux Fragment, on va créer les layout news_articles.xml. Deux layout different, un dans le dossier /res/layout (celui pour les téléphone) et l'autre dans le dossier /res/layout-large (pour les tablettes). Pour le layout destiné au téléphone, on aura simplement un *FrameLayout* avec l'id "fragment_container". Pour le layout destiné aux tablettes, on aura un LinearLayout avec une orientation horizontal qui aura comme enfants deux éléments *fragment* ou sur le tag *android:name* on donnera la correspondance au Fragment qu'on a créer. Il est nécessaire de donné le chemin complet de votre classe. Sur ces deux fragment, on mettra le layout_width à 0dp pour donner le poids avec le tag *android:layout_weight.* Celui de la liste avec un poids de 1, et l'autre de 2.

13 - Il est temps de créer notre activité principal (ne pas oublié de la déclarer dans le manifest avec l'intent-filter qui permet de pouvoir être lancer au démarrage). Elle sera une sous classe de la classe *FragmentActivity* et implémentera l'interface déclarer dans la classe HeadlinesFragment:

```
public class MainActivity extends FragmentActivity  implements
HeadlinesFragment.OnHeadlineSelectedListener {
    …
}
```

Ceci nous obligera a implementer la méthode *onArticleSelected(int position)*.

14 - Faire un *Override* à la méthode *onCreate()* pour associer notre layout *news_articles* à l'activité avec la méthode *setContentView()* (ne pas oublié d'appeler super). Ici, on va faire un test pour savoir si l'utilisateur utilise un smartphone, en cherchant le *FrameLayout* déclarer seulement dans le layout pour smartphone grace à un *findViewById().* Si ce dernier n'est pas null, il faut l'injecter le Fragment initial, le HeadLinesFragment (si le bundle n'est pas null, ceci veut dir que l'activité est recrée et que le fragment est deja injecter, il faut alors passer cette étape avec un *return*)

```
// Créer une instance du HealinesFragment
HeadlinesFragment firstFragment = new HeadlinesFragment();

// Addition le fragment au FrameLayout 'fragment_container'
getSupportFragmentManager().beginTransaction().add(R.id.fragment_container,
firstFragment).commit();
```

15 - Maintenant, dans la méthode *onArticleSelected(int position)* , on doit implementer le changement du contenu. Premier, il faut savoir si l'utilisateur est sur tablette ou smartphone, pour ça on essaye de récupérer le *ArticleFragment* déclarer dans le layout *news_articles* pour tablette.

```
ArticleFragment articleFrag = (ArticleFragment)
getSupportFragmentManager().findFragmentById(R.id.article_fragment);
```

Si ce dernier n'est pas null, on est bien sur tablette est il suffit de appelé la méthode *updateArticleView(position)* du fragment.

Si il est null, on est sur smartphone et on doit effectuer une transition de fragment:

- On déclare une nouvelle instance de *ArticleFragment*.
- On crée un *Bundle* pour lui donné la position de l'article cliquer
- On appelle la méthode *setArguments()* du fragment avec notre bundle en argument.
- On déclarer une instance d'un objet *FragmentTransaction* avec la méthode getSupportFragmentManager().beginTransaction() de notre activité.
- On remplace le fragment: transaction.replace(R.id.fragment_container, newFragment);
- Ne pas oublié de rajouté l'action a la pile avec transaction.addToBackStack(null);
- On finalise avec un commit().

16 - Dernière chose, dans notre classe *HeadLinesFragment*, on n'a pas initialiser notre callback, vu que l'activité n'était pas écrite et l'implementation de l'interface OnHeadlineSelectedListener n'existais pas. Faire un *Override* à la méthode *onAttach(Activity activity)* (ne pas oublié d'appeler super). Ici, on va simplement initialiser notre variable mCallback pour que au clique sur un élément de la liste, la méthode *onArticleSelected(position)* de notre activité soit appelé.

```
// On fait un try/catch pour être sur que l'interface a été implementé dans notre activité
// Sinon, on envoi une exception
try {
        mCallback = (OnHeadlineSelectedListener) activity;
} catch (ClassCastException e) {
        throw new ClassCastException(activity.toString()  + " doit implementer
OnHeadlineSelectedListener");
}
```