Travaux Dirigés et Pratiques de Programmation Android n° 5

Interfaces dynamiques

Les objectifs de ce TD sont :

- La création d'interfaces dynamiques grâce aux fragments ;
- L'interaction entre l'activité et ses fragments ;
- L'utilisation des fragments pour la persistance des données.

Pour cela, nous allons revenir sur le jeu « Click the button! », mais avec cette fois plusieurs fragments au lieu de plusieurs activités.

Mise en place

Commencer une nouvelle application, que l'on pourra par exemple nommer `TD5Activity`, en choisissant `Empty Activity`, et *en cochant la case « Use a Fragment »*.

Observer le code de cette application :

- l'interface de `TD5Activity` contient le fragment ;
- l'interface de `TD5ActivityFragment` contient ce qu'on a habituellement directement dans l'activité: un `RelativeLayout` avec une zone de texte affichant `Hello world!` ;
- la classe `TD5ActivityFragment` contient un constructeur vide (que l'on peut compléter si besoin) et surcharge la méthode `onCreateView`, de manière à charger l'interface lorsque le fragment est attaché à une activité.

Par défaut, AndroidStudio utilise une ancienne bibliothèque de fragments, et non celle de la bibliothèque standard. Pour corriger cela, dans la liste d'importations de `TD5ActivityFragment`, remplacer `import android.support.v4.app.Fragment;` par `import android.app.Fragment;`

Attachement du fragment au niveau Java

Pour l'instant, le fragment est attaché à l'activité `TD5Activity` lors de la création de celle-ci, et ne sera détaché que lors de sa destruction. Cela offre l'un des usages des fragments : le partage d'interfaces entre plusieurs activités. En revanche, cela ne permet pas d'avoir des interfaces dynamiques : le même fragment sera en place pendant toute la durée de l'activité.

Pour éviter cela, il faut attacher le fragment au niveau du code Java, et non dans l'interface XML de l'activité.

- Remplacer l'interface XML actuelle de `TD5Activity` par une boîte `LinearLayout` contenant juste une feuille `FrameLayout`. Un `FrameLayout` est un emplacement vide, servant à réserver de la place pour des objets : c'est là que nous allons attacher notre fragment. Donner un identifiant au `FrameLayout`, comme par exemple `frameLayoutFragment`.
- Ajouter un champ à la classe `TD5Activity` contenant le fragment que l'on veut utiliser :
  ```
  private TD5ActivityFragment frag = new TD5ActivityFragment();
  ```
- Ajouter un autre champ qui servira à contenir le gestionnaire de fragments :
  ```
  private FragmentManager fragmentManager;
  ```
- Initialiser ce champ et attacher le fragment à la fin de la création de l'activité (méthode `onCreate`) :

```
fragmentManager = getFragmentManager();
FragmentTransaction transaction = fragmentManager.beginTransaction();
transaction.add(R.id.frameLayoutFragment, frag);
transaction.commit();
```

– Tester.

Attachement et détachement

Avec cette façon d'attacher le fragment, il est maintenant possible de le détacher et de l'attacher à nouveau, rendant ainsi l'interface plus dynamique.

– Ajouter un bouton à l'interface de `TD5Activity`, avec comme text « Attacher/Détacher ». Utiliser les poids pour que la place réservée au fragment soit toute la place restante.
– Associer au clic sur ce bouton une méthode permettant d'attacher le fragment si celui-ci est détaché, et de le détacher sinon. On pourra utiliser les méthodes `boolean isAdded ()` de la classe `Fragment` et `FragmentTransaction remove (Fragment fragment)` de la classe `FragmentTransaction`. Tester.
  Rappel: chaque nouvelle transaction doit commencer par `beginTransaction()` et finir par `commit()`.
– En utilisant la méthode `FragmentTransaction setTransition (int transit)` de la classe `FragmentTransaction`, ajouter des transitions de type `TRANSIT_FRAGMENT_OPEN` à l'attachement du fragment et de type `TRANSIT_FRAGMENT_CLOSE` à son détachement. Tester.

Jeu « Click the button! »

Nous allons maintenant implanter notre jeu au sein du fragment.

– Modifier l'interface du fragment pour qu'il contienne un bouton « Click the button! » et deux zones de texte « Nombre de clics : » et le nombre effectif, « 0 » au départ.
– Donner un identifiant au bouton (par exemple, `buttonClick`) et à la deuxième zone de texte (par exemple, `textViewNbClics`). Récupérer ces vues dans des champs de la classe `TD5ActivityFragment` lors de la création de la vue liée au fragment, en modifiant la méthode suivante :

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
Bundle savedInstanceState) {
    View v = inflater.inflate(R.layout.fragment_td5, container, false);
    buttonClick = (Button) v.findViewById(R.id.buttonClick);
    textViewNbClics = (TextView) v.findViewById(R.id.textViewNbClics);
    return v;
}
```

– Lors de la création de la vue associée au fragment, on génère cette vue, mais au lieu de la retourner immédiatement comme c'était le cas auparavant, on récupère d'abord les vues que l'on souhaite.
– Associer au bouton une méthode qui compte le nombre de clics et met la jour la zone de texte correspondante.
– Tester. Essayer de détacher et attacher le fragment, puis de cliquer. Que se passe-t-il ? Les clics déjà effectués sont bien préservés, mais l'affichage lorsqu'on attache le fragment de nouveau n'est pas correct. Y remédier.

Persistance des données

- Faire en sorte que le fragment conserve ses données à l'aide de la méthode `public void setRetainInstance (boolean retain)`.
- Essayer de faire tourner la tablette lorsque le fragment est attaché, et lorsqu'il est détaché. Que se passe-t-il ? Corriger à l'aide de la méthode `public abstract Fragment findFragmentById (int id)`.

Interaction entre le fragment et l'activité

Le jeu n'est pas encore complet : à chaque fois que l'on a fait cinq clics, on gagne un niveau. On souhaite que le niveau soit toujours visible indépendamment du fragment, et donc soit un élément de l'interface de l'activité et non de celle du fragment.

- Ajouter à l'interface de `TD5Activity` deux zones de texte "Niveau :" et le niveau effectif, "0" au départ. Attribuer un identifiant à la deuxième zone (par exemple `textViewNiveau`).

On va vouloir modifier cette zone de texte, non pas dans l'activité, mais dans le fragment qu'elle contient.

- Récupérer la vue correspondant à la zone de texte dans un champ de la classe `TD5ActivityFragment`. Pour cela, ajouter à la méthode `onCreateView` :

```
textViewNiveau = (TextView) getActivity().findViewById(R.id.textViewNiveau);
```

- Faire en sorte que l'affichage soit mis à jour lorsqu'on gagne un niveau.
- Tester.

On a ainsi mis à jour des objets de l'activité depuis le fragment. Le contraire n'est pas possible directement, mais il est possible d'appeler des méthodes du fragments (qui peuvent mettre à jour des objets de ce fragment) depuis l'activité.

- Ajouter un bouton « Click the button! » à l'interface de l'activité.
- Faire en sorte que des clics de l'utilisateur sur ce bouton augmente le nombre de clics (et donc, le niveau tous les cinq clics) en appelant sur l'objet `frag` une méthode de la classe `TD5ActivityFragment` (à définir également) qui va effectuer la mise à jour souhaitée. Attention à factoriser le code !
- Tester lorsque le fragment est attaché et détaché.

À vous de jouer

1. Ajouter à `TD5Activity` un deuxième fragment, qui sera également un objet de la classe `TD5ActivityFragment` (mais différent de `frag`). L'attacher au démarrage de l'activité.
2. Ajouter un deuxième bouton « Attacher/Détacher » pour ce fragment. Factoriser le code au maximum et tester.
3. Ajouter un deuxième bouton « Click the button! » au niveau de l'activité pour ce fragment. Factoriser le code au maximum et tester.
4. Comment réagit le niveau aux clics sur les différents boutons ? Faire en sorte qu'on gagne un niveau à chaque fois que la **somme** des deux nombres de clics est multiple de 5.
   Remarque : on rappelle l'existence du mot-clé `static`...