Les fragments statiques

Ongenae Dorian (FI)
24/10/2017

Résumé

Les fragments sont un outil de plus en plus utilisé dans les applications Android. Ce sont des composants réutilisables dans diverses activités possédant leur propre comportement et permettant de créer des interfaces plus complexes s'adaptant aux différentes tailles d'écran en évitant la duplication d'activité.

Au cours de ce TP, nous comprendrons dans un premier temps le fonctionnement des fragments statiques et la syntaxe nécessaire pour les mettre en place pour ensuite, dans un second temps étudier un exemple simple d'utilisation de ces fragments pour créer une interface adaptative.

Pré-requis

- Savoir programmer une application Android avec plusieurs activités.
- Savoir utiliser les layouts avec des éléments simples (TextView, EditText, …).

Code source

Code source **initial** disponible à TODO

Code source **final** disponible à TODO

Explications du TP

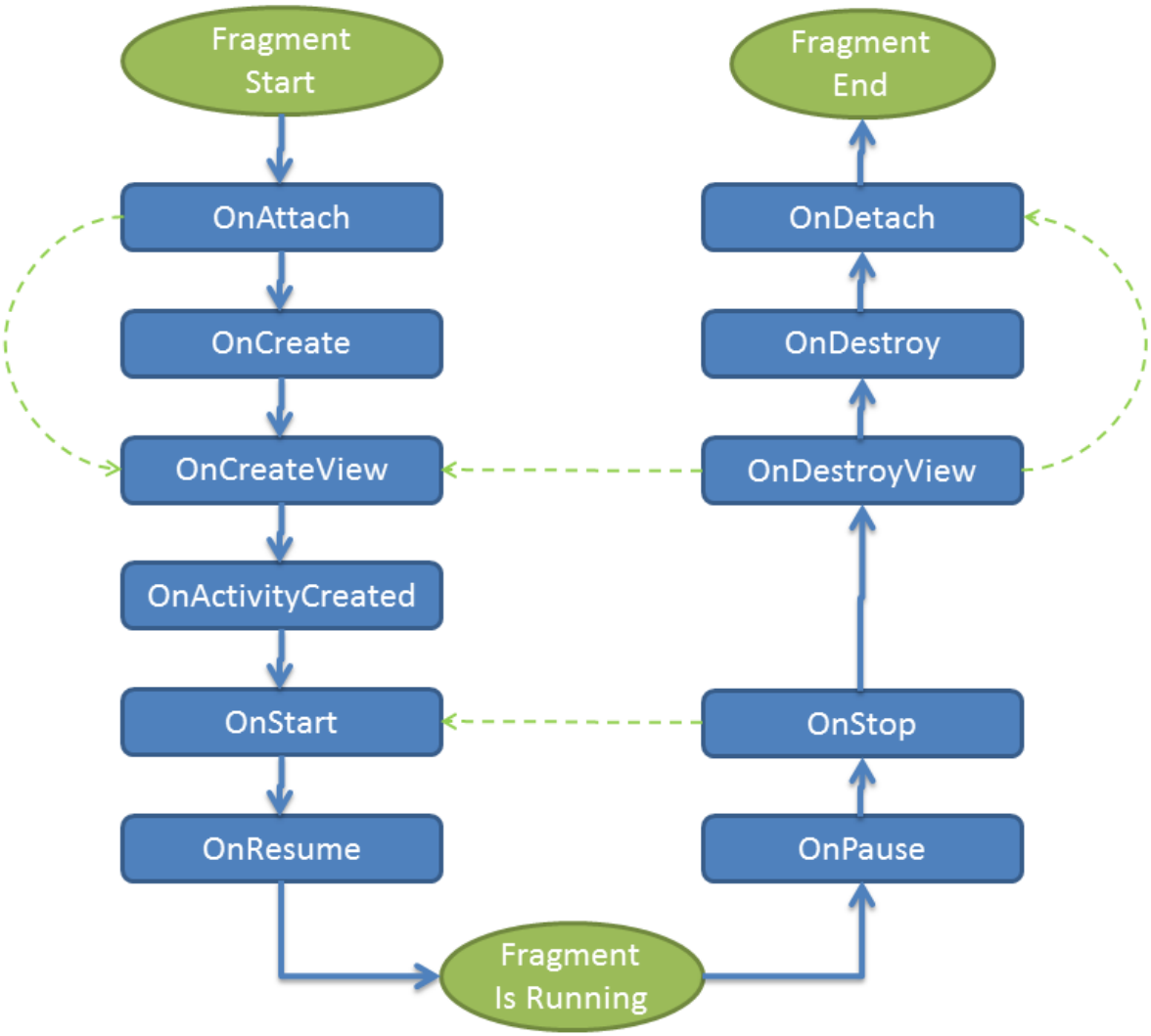
Un fragment est une portion d'interface possédant son propre comportement et utilisable dans plusieurs activités selon les besoins de l'application.

Les fragments dits « statiques » peuvent être décrits comme une section modulable de la vue d'une activité possédant son propre cycle de vie. Cela permettant de limiter la duplication de code entre les activités et la définition d'interface utilisateurs adaptatives. Ils doivent être intégrés dans le layout d'une activité avec cette balise :

```
<fragment
    android:id="@+id/mon_fragment"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:name="com.example.ongenae.myfragmentapp.MonFragment ">

</fragment>
```

Un fragment possède son propre cycle de vie différent de celui d'une activité, ce cycle se lançant quand l'application charge son layout contenant la balise fragment.



Pour plus de détails sur ces méthodes et les appels qui sont effectués, vous pouvez consulter la documentation officielle :

https://developer.android.com/guide/components/fragments.html

Pour comprendre l'implémentation d'un fragment statique on va s'intéresser à un exemple le plus simple possible.

Ici on a une application contenant un écran d'accueil et deux activités, se sont deux exercices de maths très simples mais le professeur dans son infinie mansuétude décide de laisser des notes de cours à coté de chacun d'eux.

Etape 1: Le problème de la duplication de code

Je vous invite dans un premier temps à consulter les layouts de ces activités.

activity_derivation.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal" android:layout_width="match_parent"
    android:layout_height="match_parent">
 <!-- Contenu de l'exercice -->
    <TextView
        android:id="@+id/excercice_derivation"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="@string/excercice_derivation"
         />

 <!-- Notes de cours -->
    <TextView
        android:id="@+id/notes_cours_deriv"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:textColor="@color/colorGreen"/>

</LinearLayout>
```

Activity_geometry.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent">

<!-- Contenu de l'exercice -->
    <ImageView
        android:id="@+id/img_geometrie"
        android:layout_weight="1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/thales"
        />
    <TextView
        android:id="@+id/texte_excercice_thales"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="@string/excercice_thales"/>

<!-- Notes de cours -->
    <TextView
        android:id="@+id/notes_cours_geo"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:textColor="@color/colorGreen"/>
</LinearLayout>
```

Jusqu'ici à part la partie note de cours qui est pratiquement la même dans les deux layouts il n'y a rien de choquant, mais si elle contenait des éléments plus complexes que juste un TextView on pourrait être un peu agacé..

Intéressons nous maintenant au code java.

math_activities/DerivationsActivity .java

```
public class DerivationsActivity extends AppCompatActivity {

    private TextView mTextView;

    /* Attributs contenant les notes de cours, entrées en dur pour plus de
simplicité */
    private String[] mTips = {"Si A, O, B, C, D sont cinq points tels
que:\n-> (AD) et (BC) sont parallèles.\n-> (AC) et (DB) se coupent en
O.\nAlors OB/OD = OC/OA = BC/AD.",
            "(a*u)' = a*u'", "(u + v)' = u' + v'", "(u*v)' = u'*v + u*v'",
"(u/v)' = (u'*v - u*v')/v²", "(u^a)' = a*u'*u^a-1"};

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

        mTextView = (TextView) findViewById(R.id.notes_cours_deriv);

        String notes = "notes de cours :\n";
        for (int i = 0; i < mTips.length; i++) {
            notes += "-" + mTips[i] + "\n\n";
        }
        mTextView.setText(notes);
    }
```

maths_activities/GeometryActivity .java

```
public class GeometryActivity extends AppCompatActivity {

    private TextView mTextView;

    private String[] mTips = {"Si A, O, B, C, D sont cinq points tels
que:\n-> (AD) et (BC) sont parallèles.\n-> (AC) et (DB) se coupent en
O.\nAlors OB/OD = OC/OA = BC/AD.",
            "(a*u)' = a*u'", "(u + v)' = u' + v'", "(u*v)' = u'*v + u*v'",
"(u/v)' = (u'*v - u*v')/v²", "(u^a)' = a*u'*u^a-1"};

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

        mTextView = (TextView) findViewById(R.id.notes_cours_geo);

        String notes = "notes de cours :\n";
        for (int i = 0; i < mTips.length; i++) {
            notes += "-" + mTips[i] + "\n\n";
        }
        mTextView.setText(notes);
    }
```

A partir de là on comprend que si les notes de cours nécessitaient un traitement de 30 lignes avec 4 éléments de layout le code deviendrait terriblement redondant.

Etape 2 : Implémenter un fragment statique

Je pense que vous avez compris où on arrive, maintenant on va créer un composant externe contenant ces fameuses notes de cours pour éviter la duplication de code.

Créons donc dans un premier temps la vue de ce fragment, rendez vous dans le layout correspondant.

Fragment_maths.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:id="@+id/text_fragment"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="@color/colorGreen"/>

</LinearLayout>
```

Voilà c'est simple, on a juste pris la partie commune des deux layouts vus précédemment.

Il faut maintenant définir la classe qui va gérer le fragment, celle-ci doit étendre la classe Fragment.

Fragments_maths/MathsTipsFragment.java

```
    // inflate la vue à utiliser
    @Nullable
    @Override
    public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup
container, @Nullable Bundle savedInstanceState) {
        return inflater.inflate(R.layout.fragment_maths, container, false);
    }

    // permet de modifier les élements contenus dans la vue
    @Override
    public void onViewCreated(View view, @Nullable Bundle
savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);
        mText = (TextView) view.findViewById(R.id.text_fragment);

        String notes = "Notes de cours :\n";
        for (int i = 0; i < mTips.length; i++) {
            notes += "- " + mTips[i] + "\n\n";
        }
        mText.setText(notes);
    }
```

Voilà les deux fonctions permettant de créer notre fragment.

Le OnCreateView va définir quel layout le Fragment doit « gonfler » pour générer la vue tandis que onViewCreated est appelée directement après et va nous permettre de faire le traitement nécessaire sur les éléments du layout.

A noter que l'on aurait pu mettre le contenu de onViewCreated() dans le onCreateView() mais il est recommandé de séparer initialisation et modification des éléments de la vue.

Etape 3 : Intégrer le fragment dans nos deux activités

Notre fragment est défini il reste juste à l'intégrer.

Dans un premier temps on va modifier les 2 layouts vus précedemment, **commentez** le TextView correspondant aux notes de cours et **décommentez** la balise fragment.

Astuce : Vous pouvez utiliser ctrl + / dans AndroidStudio pour commenter ou décommenter rapidement des sections.

```
<!-- Notes de cours -->
  <!--<TextView-->
      <!--android:id="@+id/notes_cours_deriv"-->
      <!--android:layout_width="0dp"-->
      <!--android:layout_height="wrap_content"-->
      <!--android:layout_weight="1"-->
      <!--android:textColor="@color/colorGreen"/>-->

  <fragment
  android:id="@+id/fragment1"
  android:layout_width="0dp"
  android:layout_height="wrap_content"
  android:layout_weight="1"

android:name="com.example.ongenae.myfragmentapp.fragments_maths.MathsTipsFragment">
  </fragment>
```

On n'oublie pas non plus de commenter la partie qui remplissait le TextView dans nos classes **DerivationActivity.java** et **GeometryActivity.java**.

```
@Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_derivation);
//        mTextView = (TextView) findViewById(R.id.notes_cours_deriv);
//
//        String notes = "notes de cours :\n";
//        for (int i = 0; i < mTips.length; i++) {
//            notes += "-" + mTips[i] + "\n\n";
//        }
//        mTextView.setText(notes);
```

Et.. c'est tout ! Nos activités vont charger leur layout, la classe qui définit le fragment dans **android:name** appellera les méthodes nécessaires à la création et on a notre partie notes de cours dans chacune de nos activités sans duplication de code.

Etape 4 : Faire interagir l'activité avec le fragment

Notre fragment est là, sauf que notre professeur se dit maintenant qu'avoir les notes de cours sur la dérivation dans un exercice demandant le théorème de Thalès et inversement manque peut-être de pertinence. Ce qu'on voudrait maintenant serait d'afficher uniquement les notes de cours utiles pour l'exercice en question.

Pour cela dans le fragment on va définir un tableau contenant les notes propres au premier exercice et un autre pour le second ainsi qu'une fonction permettant de remplir le TextView selon un paramètre souhaité (ce paramètre est défini en variable publique statique au début de la classe pour simplifier les choses dans les activités).

Commentez et **décommentez** comme ci-dessous :

MathsTipsFragment.java

```
    public static int NONE = 0;
    public static int DERIVATION = 1;
    public static int GEOMETRIE = 2;

    private TextView mText;

//    private String[] mTips = {"Si A, O, B, C, D sont cinq points tels
que:\n-> (AD) et (BC) sont parallèles.\n-> (AC) et (DB) se coupent en
O.\nAlors OB/OD = OC/OA = BC/AD.",
//            "(a*u)' = a*u'", "(u + v)' = u' + v'", "(u*v)' = u'*v +
u*v'", "(u/v)' = (u'*v - u*v')/v²", "(u^a)' = a*u'*u^a-1"};

    private String[] mDerivationTips = {"(a*u)' = a*u'", "(u + v)' = u' +
v'", "(u*v)' = u'*v + u*v'", "(u/v)' = (u'*v - u*v')/v²", "(u^a)' =
a*u'*u^a-1"};
    private String[] mGeometrieTips = {"Si A, O, B, C, D sont cinq points
tels que:\n-> (AD) et (BC) sont parallèles.\n-> (AC) et (DB) se coupent en
O.\nAlors OB/OD = OC/OA = BC/AD."};

    // inflate la vue à utiliser
    @Nullable
    @Override
    public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup
container, @Nullable Bundle savedInstanceState) {
        return inflater.inflate(R.layout.fragment_maths, container, false);
    }

    // permet de modifier les élements contenus dans la vue
    @Override
    public void onViewCreated(View view, @Nullable Bundle
savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);
        mText = (TextView) view.findViewById(R.id.text_fragment);

//        String notes = "Notes de cours :\n";
//        for (int i = 0; i < mTips.length; i++) {
//            notes += "- " + mTips[i] + "\n\n";
//        }
//        mText.setText(notes);
    }
```

```
    /**
     * permet de définir quelles règles sont à afficher
     * @param rule, une des variables statiques du début de la classe
     */
    public void chooseTips(int rule) {
        String tips = "Notes de cours :\n";
        if (rule == this.NONE) {
            tips += "Pas de notes de cours pour cet excercice, bonne chance
!";
        } else if (rule == this.DERIVATION) {
            for (int i = 0; i < mDerivationTips.length; i++) {
                tips += "- " + mDerivationTips[i] + "\n\n";
            }
        } else if (rule == this.GEOMETRIE) {
            for (int i = 0; i < mGeometrieTips.length; i++) {
                tips += "- " + mGeometrieTips[i] + "\n\n";
            }

        }
        mText.setText(tips);
    }
```

Cette dernière fonction est un « setter » que les activités pourront appeler pour interagir avec le fragment.

Il faut maintenant modifier le code des activités pour qu'elles appellent cette méthode.

A noter qu'on ne peut pas récupérer le fragment du layout avec un simple findViewById() mais il faut passer par un FragmentManager comme ci-dessous, **décommentez** et **commentez** les attributs ainsi que les deux lignes dans la méthode onCreate() de ces deux activités.

DerivationsActivity.java

```
private MathsTipsFragment mFrag;

  /* Attributs contenant les notes de cours, entré en dur pour plus de
simplicité */
//    private String[] mTips = {"Si A, O, B, C, D sont cinq points tels
que:\n-> (AD) et (BC) sont parallèles.\n-> (AC) et (DB) se coupent en
O.\nAlors OB/OD = OC/OA = BC/AD.",
//            "(a*u)' = a*u'", "(u + v)' = u' + v'", "(u*v)' = u'*v +
u*v'", "(u/v)' = (u'*v - u*v')/v²", "(u^a)' = a*u'*u^a-1"};

@Override
protected void onCreate(@Nullable Bundle savedInstanceState) {
(...)

        mFrag = (MathsTipsFragment)
getFragmentManager().findFragmentById(R.id.fragment1);
    mFrag.chooseTips(MathsTipsFragment.DERIVATION);

}
```

GeometryActivity .java

```
private MathsTipsFragment mFrag;

//    /**
//     * liste des notes de cours à afficher pour l'excercice (normalement
à remplir avec un setter mais ici en brut pour plus de simplicité)
//     */
//    private String[] mTips = {"Si A, O, B, C, D sont cinq points tels
que:\n-> (AD) et (BC) sont parallèles.\n-> (AC) et (DB) se coupent en
O.\nAlors OB/OD = OC/OA = BC/AD.",
//            "(a*u)' = a*u'", "(u + v)' = u' + v'", "(u*v)' = u'*v +
u*v'", "(u/v)' = (u'*v - u*v')/v²", "(u^a)' = a*u'*u^a-1"};


@Override
protected void onCreate(@Nullable Bundle savedInstanceState) {
(...)


        mFrag = (MathsTipsFragment)
      getFragmentManager().findFragmentById(R.id.fragment2);
      mFrag.chooseTips(MathsTipsFragment.GEOMETRIE);
}
```

Voilà, vous avez implémenté un fragment statique, puis vous l'avez intégré à vos activités pour finalement définir un comportement spécifique via un setter.

Etape 5 : Exemple d'utilisation des fragments pour définir une interface adaptative

Maintenant que l'on a compris le fonctionnement des fragments nous allons voir comment les fragments peuvent être utilisés de manière judicieuse pour créer des interfaces s'adaptant au format de l'écran.

Ici il s'agit de faire une application avec une partie permettant de remplir un relevé et une autre section contenant les instructions pour remplir ce formulaire.

En mode portrait sur un téléphone, au vu de la taille de ces éléments il est impossible d'afficher les deux proprement en même temps, on va donc séparer ces deux parties. Dans un autre cas de figure, sur une tablette par exemple, ou même en mode paysage sur un téléphone on pourrait imaginer qu'afficher les deux sur le même écran dans la même activité est plus pratique.

Voici l'objectif :

Soit les deux fragments sur la même activité soit chacun dans une activité

Le fragment A étant notre formulaire et le fragment B nos instructions (enfin presque, on va plutôt allouer la moitié de l'espace et non pas un tiers avec notre exemple).

En connaissant la syntaxe cela devient très simple et je vous invite à consulter les fichiers au fur et à mesure de votre lecture :

- On définit nos fragments donc notre formulaire dans la vue **fragment_form_releve.xml** qu'on implémente avec la classe **FormReleveFragment.java**, on fait de même pour les instructions avec **fragment_instruction_releve.xml** et **InstructionsReleveFragment.java**.

- On crée notre première activité correspondant au formulaire, ici on va avoir besoin d'un layout **activity_releve.xml (port)** pour le mode portrait, il contient le fragment du formulaire et le bouton pour accéder aux instructions avec sa classe java ** releveActivity.java**.

- On crée la seconde activité qui contiendra uniquement les instructions, donc le layout **activity_instructions.xml** avec le fragment instructions et la classe java associée **InstructionsActivity.java**.

- On arrive maintenant à la partie intéressante, on a définit un second layout **activity_releve.xml (land)** qui se chargera quand le téléphone est en mode paysage. Dans celui là on met les deux fragments pour obtenir l'écran vu plus haut, cela ne demande pas de code java supplémentaire et est totalement fonctionnel.

Pour aller plus loin si vous souhaitez définir un comportement différent pour l'activité selon le cas dans lequel on est il faut simplement vérifier si le fragment instructions est dans la vue et ajouter du comportement.

ReleveActivity.java

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_releve);

    mFragInstr = (InstructionsReleveFragment)
getFragmentManager().findFragmentById(R.id.fragment_instruction2);

    if (mFragInstr == null || !mFragInstr.isInLayout()) {
        // Cela veut dire qu'on est en mode portrait, on peut définir un
comportement différent ou changer le contenu d'un textview par exemple
    } else {
        // On est en mode paysage, de même si l'on souhaite ajouter du code
spécifique
    }
}
```

Conclusion

Les fragments sont un outil puissant et beaucoup utilisé dans les applications d'aujourd'hui, j'espère que vous avez pu comprendre les concepts de base dans l'utilisation de ces fragments statiques et tout l'intérêt qu'ils possèdent pour des applications plus légères en code mais aussi en mémoire, un fragment est beaucoup moins couteux en ressource qu'une activité.

Toutefois les fragments statiques sont très limités par rapport à leurs homonymes dynamiques qui permettent de créer des applications très complexes en limitant au maximum le nombre d'activités. Je vous invite donc à vous y intéresser par vous-même ou via les liens ci-dessous.

Si vous souhaitez en savoir plus vous pouvez consulter :

- https://developer.android.com/guide/components/fragments.html *: le guide complet d'android concernant les fragments qui aborde également les fragments dynamiques*
- http://mathias-seguy.developpez.com/tutoriels/android/comprendre-fragments/ *: un tutoriel plus poussé qui permet de réaliser une application combinant toutes ces connaissances avec un ListView et l'utilisation de listener dans les fragments selon le design pattern « MasterDetails ». En plus de notions sur les fragments dynamiques.*
- http://vogella.developpez.com/tutoriels/android/utiliser-fragments/ *: un tutoriel similaire mais détaillant un peu plus le code.*