RxJava (http://developer.alexanderklimov.ru/android/rx/)

Советы (http://developer.alexanderklimov.ru/android/tips-android.php)

Статьи (http://developer.alexanderklimov.ru/android/articles-android.php)

Книги (http://developer.alexanderklimov.ru/android/books.php)

Java (http://developer.alexanderklimov.ru/android/java/java.php)

Kotlin (http://developer.alexanderklimov.ru/android/kotlin/)

Дизайн (http://developer.alexanderklimov.ru/android/design/)

Отладка (http://developer.alexanderklimov.ru/android/debug/)

Open Source (http://developer.alexanderklimov.ru/android/opensource.php)

Полезные ресурсы (http://developer.alexanderklimov.ru/android/links.php)

# Fragment (Фрагменты). Часть четвёртая

# Динамическое управление фрагментами

Мы использовали теги **fragment** в разметке для размещения фрагментов. Но существует ещё альтернативный вариант, когда фрагмент вставляется в какой-нибудь контейнер динамически. В качестве контейнера обычно используют **FrameLayout**, но можно и другие элементы из **ViewGroup**. Давайте попробуем изменить свою программу под новый способ.

В разметке **activity_main.xml** удалим тег **fragment**, а корневому компоненту присвоим идентификатор (у нас уже был идентификатор **@+id/linearLayout**, но я решил его заменить на более говорящий).

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/container"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center_horizontal"
    android:orientation="vertical"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity" >

    <!-- <fragment -->
    <!-- android:id="@+id/fragment1" -->
    <!-- android:name="ru.alexanderklimov.fragmentdemo.Fragment1" -->
    <!-- android:layout_width="match_parent" -->
    <!-- android:layout_height="0dp" -->
    <!-- android:layout_weight="1" -->
    <!-- tools:layout="@layout/fragment1" /> -->

</LinearLayout>
```

Ещё раз обращаю внимание, что заменить статический фрагмент программно нельзя. Если вы используете в разметке тег **fragment**, то это уже навсегда.

А у нас сложилась странная ситуация, мы удалили фрагмент из разметки и на что-то рассчитываем? Наивные.

На самом деле, всё не так безнадёжно. Сам класс фрагмента **Fragment1** у нас остался и его по-прежнему можно использовать.

Добавим в класс **MainActivity** новую переменную:

```
private boolean mIsDynamic;
```

Далее в методе **onCreate()** устроим небольшую проверку для второго фрагмента. Если он не существует (**null**) или не является частью разметки (**isInLayout**), то переменная **mIsDynamic** будет иметь значение **true**.

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

    FragmentManager fragmentManager = getSupportFragmentManager();
    Fragment2 fragment2 = (Fragment2) fragmentManager
            .findFragmentById(R.id.fragment2);
    mIsDynamic = fragment2 == null || !fragment2.isInLayout();
    Toast.makeText(getApplicationContext(), mIsDynamic + "", Toast.LENGTH_SHORT).show();
}
```

Я специально добавил в код вызов всплывающего сообщения, чтобы вы увидели, как меняется значение переменной **mIsDynamic**. При запуске в портретном режиме вы увидите пустой экран, так как удалили из разметки фрагмент, а переменная будет равна **true**. При повороте в альбомный режим загрузится альтернативная разметка с двумя фрагментами, которую мы не трогали. А булева переменная примет значение **false**.

Теперь, когда мы знаем, что при старте второй фрагмент не используется, загрузим программно первый фрагмент в том же методе **onCreate()**.

```
// Зная, что второго фрагмента нет, загружаем первый
if (mIsDynamic) {
        // начинаем транзакцию
        FragmentTransaction ft = fragmentManager.beginTransaction();
        // Создаем и добавляем первый фрагмент
        Fragment1 fragment1 = new Fragment1();
        ft.add(R.id.container, fragment1, "fragment1");
        // Подтверждаем операцию
        ft.commit();
}
```

Активность может иметь несколько фрагментов, которые должны выполнить какую-то операцию. Транзакция позволяет выполнить все операции скопом. Вы сообщаете менеджеру про все операции, запускаете их в методе **beginTransaction()** и подтверждаете своё намерение через метод **commit()**.

В методе **add()** мы указываем идентификатор контейнера **R.id.container**, в который нужно загрузить наш фрагмент. Запустите пример, чтобы увидеть, что приложение работает как и раньше.

Для несложного примера этого вполне достаточно, но иногда требуется более сложное взаимодействие между фрагментами. Поэтому продолжим эксперименты.

Перепишем у первой активности метод **onButtonSelected()** с использованием переменной **mIsDynamic**:

```
@Override
public void onButtonSelected(int buttonIndex) {
        // подключаем FragmentManager
        FragmentManager fragmentManager = getSupportFragmentManager();
        Fragment2 fragment2;

        // Если фрагмент недоступен
        if (mIsDynamic) {
                // Динамическое переключение на другой фрагмент (позже)
        } else {
                // Если фрагмент доступен
                fragment2 = (Fragment2) fragmentManager
                                .findFragmentById(R.id.fragment2);
                fragment2.setDescription(buttonIndex);
        }
}
```

Для альбомного режима всё осталось без изменений. Если второй фрагмент доступен, то выводим данные в соответствии с нажатой кнопкой.

С портретной ориентацией ситуация интереснее. Мы можем не запускать новую активность, которая содержит второй фрагмент, а динамически заменить первый фрагмент на второй. Удобно! Нам не нужна лишняя активность. Напишем код для условия **if** из предыдущего примера.

```
if (mIsDynamic) {
        // Динамическое переключение на другой фрагмент
        FragmentTransaction ft = fragmentManager.beginTransaction();
        fragment2 = new Fragment2();
        ft.replace(R.id.container, fragment2, "fragment2");
        ft.addToBackStack(null);
        ft.setCustomAnimations(
        android.R.animator.fade_in, android.R.animator.fade_out);
        ft.commit();
}
```

Запускаем проект и проверяем. Действительно, вместо первого фрагмента появляется второй. Правда при этом никак не учитывается нажатая кнопка. Непорядок.

А происходит это потому, что транзакция вызывается чуть раньше, чем методы фрагмента **onCreate()** и **onCreateView()**. Мы пойдём другим путём. У фрагментов есть методы **getArguments/setArguments()**, способные передавать и принимать данные.

Объявим переменные в классе **Fragment2**:

```
// Имя для аргумента
public static final String BUTTON_INDEX = "button_index";
// Значение по умолчанию
private static final int BUTTON_INDEX_DEFAULT = -1;
```

Переделаем в этом же классе метод **onCreateView()** с применением аргументов.

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
            Bundle savedInstanceState) {
    View rootView =
            inflater.inflate(R.layout.fragment2, container, false);

    mInfoTextView = (TextView) rootView.findViewById(R.id.textView1);
    mCatImageView = (ImageView) rootView.findViewById(R.id.imageView1);

    // загружаем массив из ресурсов
    mCatDescriptionArray = getResources().getStringArray(R.array.cats);

    // Получим индекс, если имеется
    Bundle args = getArguments();
    int buttonIndex = args != null ? args.getInt(BUTTON_INDEX,
            BUTTON_INDEX_DEFAULT) : BUTTON_INDEX_DEFAULT;
    // Если индекс обнаружен, то используем его
    if (buttonIndex != BUTTON_INDEX_DEFAULT)
        setDescription(buttonIndex);

    return rootView;
}
```

Если в фрагмент поступят данные, то обрабатываем их и выводим нужную информацию. Осталось только подготовить нужные данные. Сделаем это в методе первой активности **onButtonSelected()**

```
@Override
public void onButtonSelected(int buttonIndex) {
        // подключаем FragmentManager
        FragmentManager fragmentManager = getSupportFragmentManager();
        Fragment2 fragment2;

        // Если фрагмент недоступен
        if (mIsDynamic) {
                // Динамическое переключение на другой фрагмент
                FragmentTransaction ft = fragmentManager.beginTransaction();
                fragment2 = new Fragment2();

                // Подготавливаем аргументы
                Bundle args = new Bundle();
                args.putInt(Fragment2.BUTTON_INDEX, buttonIndex);
                fragment2.setArguments(args);

                ft.replace(R.id.container, fragment2, "fragment2");
        ft.addToBackStack(null);
        ft.setCustomAnimations(
                android.R.animator.fade_in, android.R.animator.fade_out);
        ft.commit();
        } else {
                // Если фрагмент доступен
                fragment2 = (Fragment2) fragmentManager
                                .findFragmentById(R.id.fragment2);
                fragment2.setDescription(buttonIndex);
        }
}
```

Это был заключительный аккорд. Мы переделали приложение, полностью отказавшись от второй активности, так как все фрагменты загружаются в один и тот же контейнер.

Дополнительные материалы по фрагментам есть в отдельном цикле статей (http://developer.alexanderklimov.ru/android/theory/fragments.php).

Важно понять, в каких случаях удобнее использовать статичные фрагменты, а в каких - динамические. В статье мы многое делали вручную. Но в студии есть готовые шаблоны, и поэтому часть работы будут автоматизирована. Кроме того есть ещё различные нюансы, о которых вы узнаете с практикой.

Часть пятая. Сохранение данных (fragment5.php)

# Дополнительное чтение

Обсуждение статьи (http://forum.alexanderklimov.ru/viewtopic.php?id=31) на форуме.

## Реклама