Java

Kotlin

Дизайн

Отладка

Open Source

Полезные ресурсы

Fragment (Фрагменты)

Фрагменты немного пугают новичков. Постараюсь объяснить как можно проще, чтобы отдельные фрагменты пазла сложились в единую картину.

Зачем?

Создатели операционной системы Android оказались недальновидными разработчиками. Не посоветовавшись с котами, они разработали систему под маленькие экраны телефонов. Но котам было неудобно пользоваться такими экранами, и тогда придумали планшеты.

Старые программы прекрасно на них запускались, но обнаружилось несколько недостатков. На больших экранах интерфейс выглядел не слишком элегантно, появились большие пустые пространства. И тогда возникла идея объединить два отдельных экрана из смартфона в один экран на планшете. Это самый классический пример применения фрагмента. По сути, это костыль. Возможно, если бы сразу подумали головой, то придумали бы более элегантное решение. Но теперь поздно пить Боржоми, будем использовать предложенную концепцию.

Фрагменты были представлены в API 11 (Android 3.0), но в целях совместимости была написана специальная библиотека **Android Support library** для старых устройств. Таким образом сейчас существуют два класса **Fragment**: для новых устройств и для старых устройств. Названия методов и классов очень похожи, но не стоит смешивать в одном проекте два разных класса.

При желании можно было продолжить писать приложения в старом стиле, отслеживая размеры экрана. Но такой код получится слишком сложным. Пришлось бы писать один код для переключения от одной активности к другой при использовании смартфона и другой код, когда взаимодействие между объектами происходит на одном экране в планшете. Чтобы устранить это противоречие, были придуманы фрагменты. Хотя там тоже придётся писать много кода.

Несколько слов о том, как проще воспринимать фрагмент. Считайте, что фрагмент - это тот же компонент как **Button**, **TextView** или **LinearLayout** с дополнительными возможностями. Фрагмент, как и кнопку, нужно поместить на экран активности. Но фрагмент является модульным компонентом и один и тот же фрагмент можно встроить в две разные активности. С кнопкой такой номер не пройдёт. Для каждой активности вы должны создать свою отдельную кнопку, даже если их нельзя будет отличить друг от друга.

Но фрагменты - это не замена активности, они не существуют сами по себе, а только в составе активностей. Поэтому в манифесте прописывать их не нужно. Но в отличие от стандартной кнопки, для каждого фрагмента вам придётся создавать отдельный класс, как

В составе активности есть специальный менеджер фрагментов, который может контролировать все классы фрагментов и управлять ими. О нём позже.

Фрагменты являются строительным материалом для приложения. Вы можете в нужное время добавить новый фрагмент, удалить ненужный фрагмент или заменить один фрагмент на другой. Точно так же мы собираем пазл - подносим фрагмент кота в общую картину, иногда ошибаемся и тогда заменяем кусочек пазла на другой и т.д.

Фрагмент может иметь свою разметку, а может обойтись без неё. Также у фрагмента есть свой жизненный цикл, во многом совпадающий с жизненным циклом активности. Пожалуй, это единственное сходство с активностью.

Имеются специальные виды фрагментов, заточенные под определённые задачи - **ListFragment**, **DialogFragment** и другие, которые изучим в других уроках.

Есть два варианта использования фрагментов в приложении (при желании можно использовать сразу оба варианта). Первый вариант заключается в том, что вы в разметке сразу указываете фрагмент с помощью тега **fragment**, так же как и с другими компонентами.

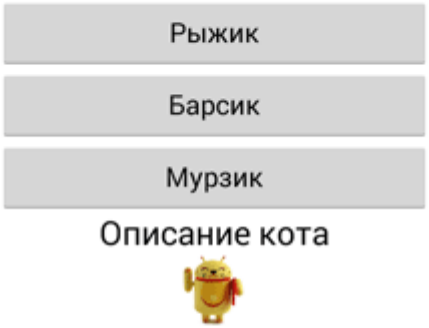
Второй вариант использует динамическое подключение фрагмента. Принцип следующий - в разметку помещается макет из группы **ViewGroup**, который становится контейнером для фрагмента. Обычно, для этой цели используют **FrameLayout**, но это не обязательное условие. И в нужный момент фрагмент замещает контейнер и становится частью разметки.

Поначалу фрагменты кажутся неудобными, так как количество кода увеличивается. Но если с ними работать постоянно, то станет понятнее их принцип.

Работаем по старинке

Чтобы было легче перестроиться на новую технологию, начнём издалека и создадим сначала стандартную программу. Набросаем на экран несколько кнопок и других компонентов.

```
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayout1"
    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" >

    <Button
        android:id="@+id/button1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Рыжик" />

    <Button
        android:id="@+id/button2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Барсик" />

    <Button
        android:id="@+id/button3"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Мурзик" />

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Описание кота"
        android:textAppearance="?android:attr/textAppearanceLarge" />

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:scaleType="fitCenter"
        android:src="@drawable/cat_yellow" />

</LinearLayout>
```



Думаю, вам уже не составит труда написать код для кнопок, чтобы в нижней части экрана менялась картинка и текстовое содержание про каждого кота. Но я этого делать пока не буду.

Заворачиваем в фрагменты

Фрагмент, как и активность, состоит из разметки и класса. Сначала займёмся разметкой.

Логически экран можно разделить на две части - верхняя неизменяемая часть с кнопками и нижняя часть с текстовым блоком и контейнером для картинки, которая изменяет свой вид в зависимости от нажатой кнопки.

Создадим две отдельные разметки и скопируем нужные части из общей разметки в разметки для фрагментов. Делаем щелчок правой кнопкой мыши на папке **res/layout** и выбираем **New | Layout resource file**.

Создаём новый файл **fragment1.xml** и размещаем верхнюю часть кода:

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

    <Button
        android:id="@+id/button1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Рыжик" />

    <Button
        android:id="@+id/button2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Барсик" />

    <Button
        android:id="@+id/button3"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Мурзик" />

</LinearLayout>
```

Также поступаем со вторым фрагментом - создаём новый файл **fragment2.xml** и в него копируем код из нижней части кода.

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

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Описание кота"
        android:textAppearance="?android:attr/textAppearanceLarge" />

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:scaleType="fitCenter"
        android:src="@drawable/cat_yellow" />

</LinearLayout>
```

В обоих случаях мы вставляли код в корневой контейнер **LinearLayout**. Но если у вас была бы более сложная разметка с использованием контейнерных элементов, то вы могли бы копировать сразу готовый кусок кода без необходимости оборачивать его корневым элементом, как в нашем случае.

Пока мы создали разметки для будущих фрагментов. Теперь нужно создать отдельные классы для двух фрагментов. Делаем щелчок правой кнопкой мыши на имени пакета и выбираем **New | Java Class**. Присваиваем имя **Fragment1**.

На данный момент студия создаёт проекты с использованием библиотеки совместимости, поэтому будем использовать фрагменты из этой библиотеки. В будущем переделать код под обычные фрагменты будет не сложно. Если ваш проект создан сразу под новые устройства (в главной активности строчка **MainActivity extends Activity**), то можете сразу использовать новые фрагменты.

Для начала укажем, что наш класс должен наследоваться от класса **Fragment**. Не копируйте, а пишите код самостоятельно.

```
public class Fragment1 extends Fragment {

}
```

Следите, чтобы импортировался класс **android.support.v4.app.Fragment**, а не **android.app.Fragment**, который предназначен для новых устройств.

Настало время подключить разметки к фрагментам. В активностях мы подключали разметку в методе **onCreate()** через метод **setContentView()**. В фрагментах метод **onCreate()** служит для других задач. А для подключения разметки используется отдельный метод **onCreateView()**.

Если у вас открыт код для **Fragment1**, то нажимаем комбинацию клавиш **Ctrl+O** (или меню **Code | Override Methods...**). Чтобы долго не искать нужный нам метод, просто вводите на клавиатуре первую и заглавные буквы метода - `ocv` . Такой комбинации соответствует только один метод, который нам и нужен. Нажимаем кнопку **OK** и в код фрагмента будет вставлен следующий шаблон:

```
@Nullable
@Override
public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container,
            @Nullable Bundle savedInstanceState) {
    return super.onCreateView(inflater, container, savedInstanceState);
}
```

У метода используются три параметра. В первом параметре используется объект класса **LayoutInflater**, который позволяет построить нужный макет, считывая информацию из указанного XML-файла. Удалим строчку, которая возвращает результат и напишем свой вариант.

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                Bundle savedInstanceState) {
        //return super.onCreateView(inflater, container, savedInstanceState);

    View rootView =
            inflater.inflate(R.layout.fragment1, container, false);
    return rootView;
}
```

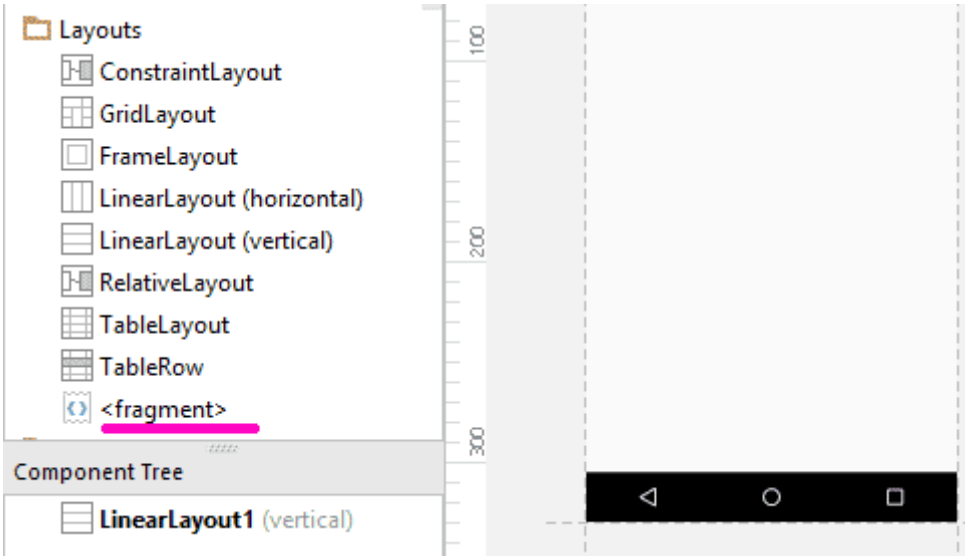
Для удобства код разбит на две части. Сначала мы получаем объект **View**, а затем уже его возвращаем в методе.

Скопируйте код метода **onCreateView()** и вставьте его в код класса **Fragment2**, не забыв указать разметку **R.layout.fragment2**.

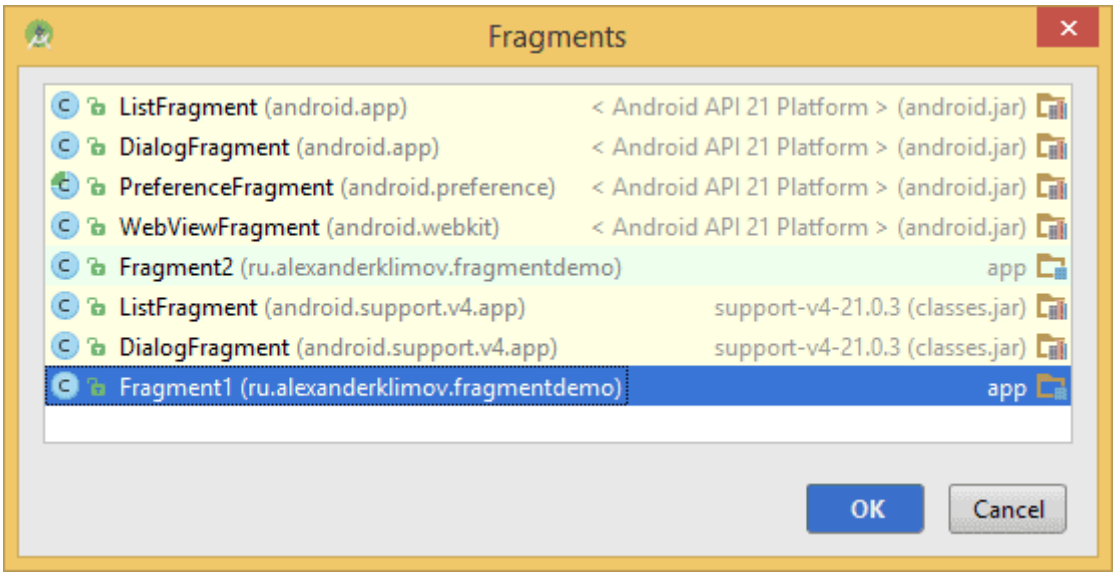
Остальные два параметра **container**, **false** используются в связке и указывают на возможнось подключения фрагментов в активность через контейнер. Мы обойдёмся без контейнеров, а создадим собственные блоки для фрагментов, поэтому у нас используется значение **false**.

которыи удалить не получится.

В панели **Palette** находим в разделе **Layout** элемент **<fragment>**.



Щёлкнув на нём, мы вызовем диалоговое окно, в котором нужно найти свой созданный фрагмент.



Разместите его на экране активности и повторите эти действия для размещения второго фрагмента.

В студии версии 2.2.3 пытался повторить эти шаги, но ничего не получилось. Возможно, это баг. Поэтому, если у вас тоже не получилось, то пишем код вручную (см. ниже).

Переключитесь в режим **Text** и посмотрите, как фрагменты описываются в XML-коде. Для фрагментов используется тег **fragment** с стандартными атрибутами. В атрибуте **name** указываются имена классов созданных фрагментов.

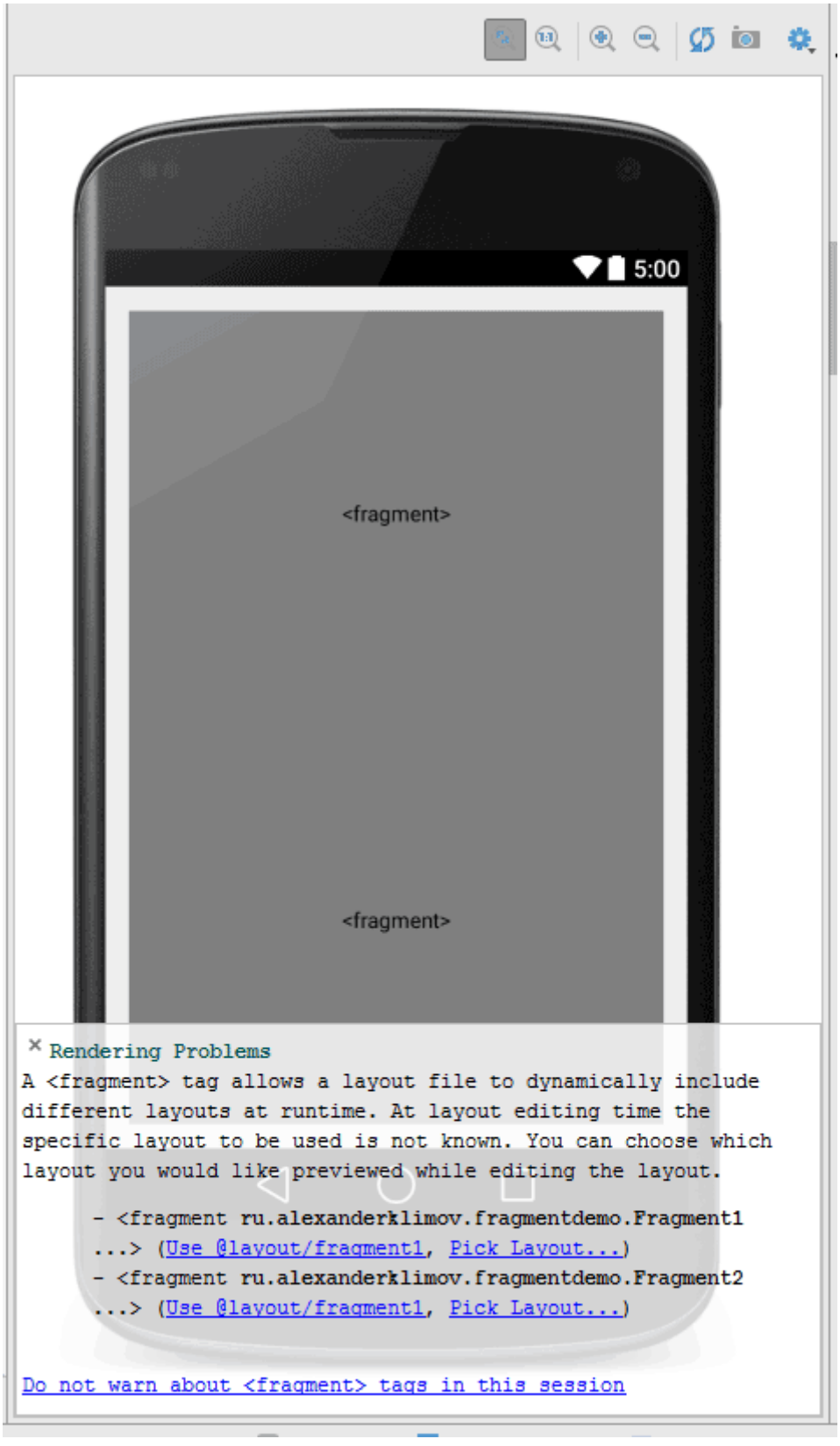
```
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center_horizontal"
    android:orientation="vertical"
    android:paddingBottom="@dimen/activity_vertical_margin"
    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/fragment"
        android:name="ru.alexanderklimov.fragmentdemo.Fragment1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal" />

    <fragment
        android:id="@+id/fragment2"
        android:name="ru.alexanderklimov.fragmentdemo.Fragment2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal" />
</LinearLayout>
```

Если вас не пугает XML-код, то можете создавать сразу в редакторе кода.

Стандартные настройки могут нас не устраивать. Укажем другие параметры, влияющие на вес элементов:

```
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayout"
    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" />

    <fragment
        android:id="@+id/fragment2"
        android:name="ru.alexanderklimov.fragmentdemo.Fragment2"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1" />

</LinearLayout>
```

Студия в окне **Preview** сообщает нам о проблемах. Она не может отобразить правильно разметку, так как немного туповата.



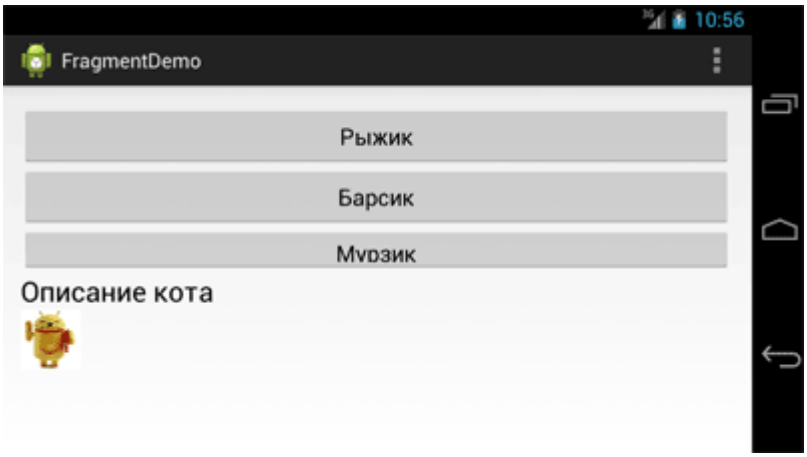
Вы можете помочь ей. Щёлкните по ссылке **Use @layout/fragment1**, и нужная разметка фрагмента подключится к разметке активности. Аналогично проделайте со вторым фрагментом. Ссылка **Pick Layout...** выводит диалоговое окно, если надо выбрать разметку из списка.

Внешне мы получим такой же результат, как и без фрагментов.

Если сейчас запустим приложение, то тоже никаких изменений не увидим. Зачем тогда потратили столько времени на создание фрагментов? Непонятно.

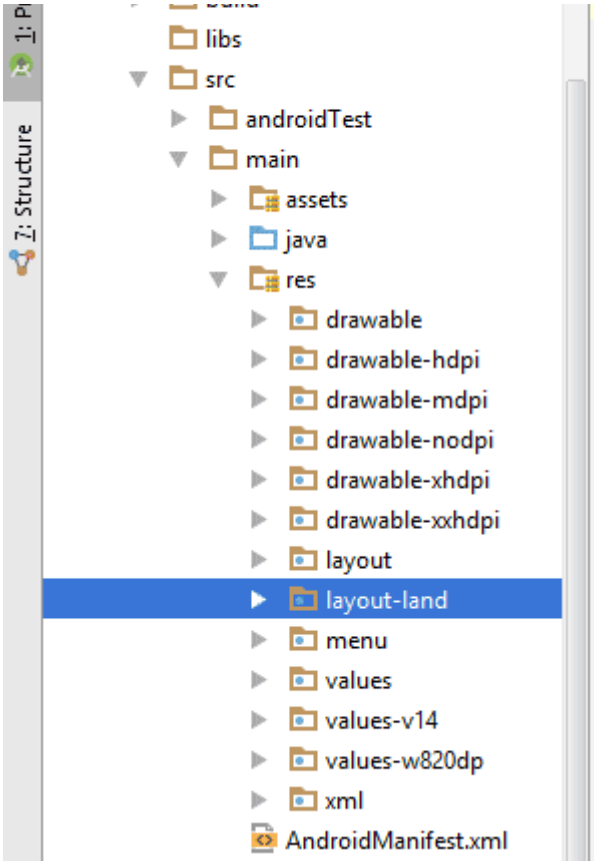
А, я понял. Можно теперь писать в резюме про свои умения: использую фрагменты.

Однако, продолжим. Если повернуть устройство в альбомную ориентацию, то программа будет выглядеть не слишком красиво.



Мы знаем, что можно создать отдельную папку **res/layout-land** (перечитайте урок Ориентация) и разместить там разметку для такого случая. Скопируем файл **activity_main.xml** и вставим его в новую папку.

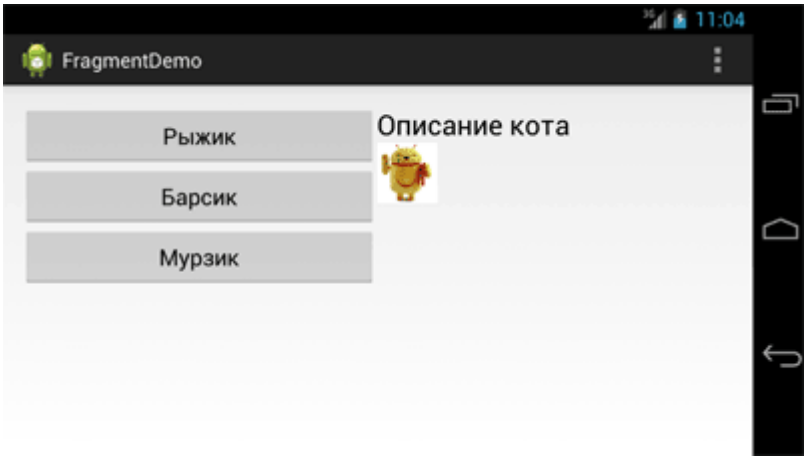
Скорее всего вы не видите созданную папку, но она есть! Переключитесь в режим **Project** и заново откройте структуру проекта, найдя нужную папку. В дальнейшем оставайтесь в этом режиме для данного урока.



Изменим разметку фрагмента для альбомной ориентации.

```
    android:id="@+id/LinearLayout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:baselineAligned="false"
    tools:context=".MainActivity" >

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

    <fragment
        android:id="@+id/fragment2"
        android:name="ru.alexanderklimov.fragmentdemo.Fragment2"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="1"
        tools:layout="@layout/fragment2" />

</LinearLayout>
```

Не забывайте в имени фрагментов использовать свои названия пакетов. Совсем другое дело. Теперь в альбомной ориентации приложение выглядит намного лучше.



Но это мы могли сделать и без фрагментов. Зачем же они нужны? Пока версия с лишней строчкой в резюме остаётся основной - чтобы работодатель уважал за прогрессивный стиль.

трогали.

Как уже говорилось, фрагменты были придуманы для того, чтобы обеспечить быстрое написание приложения под разные типы экранов - для смартфонов и планшетов. Часто бывает так, что на смартфоне на первом экране находится список, а когда пользователь нажимает на отдельный элемент списка, то запускается отдельная активность. А на планшете можно уместить список и дополнительные данные на одном экране, как можно увидеть на нашем последнем примере с альбомной ориентацией.

Давайте подключим поддержку планшетов. Создадим новую папку **layout-sw600dp** и скопируем в него файл из папки **layout-land**. Идентификатор **sw600** говорит о минимальной ширине 600dp, что соответствует 7-дюймовым планшетам в альбомной ориентации. Существуют и другие варианты для планшетов с большими размерами.

Тут возникает небольшая проблема - если нам понадобится что-то изменить в разметке для альбомной ориентации, то придётся редактировать файлы во всех папках. Но есть выход из этой ситуации - использовать псевдонимы.

Мы можем создать одну копию разметки и указать, чтобы её использовали все нужные размеры устройств.

Делаем следующее. В папке **layout-land** переименовываем файл **activity_main.xml** в **activity_main_wide.xml** (**Refactor | Rename**) и перемещаем файл в папку **layout**. Пустую папку **layout-land** можно удалить.

Теперь создайте новую папку **res/values-land**. В созданной папке создаём новый файл **refs.xml** (имя не имеет значения, но так принято).

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <item name="activity_main" type="layout">@layout/activity_main_wide
</item>

</resources>
```

Этот файл говорит, что в альбомной ориентации вместо ресурса **activity_main** следует подключать ресурс **layout/activity_main_wide**. Можете запустить приложение и убедиться, что ничего не изменилось.

Если у вас будет поддержка альбомных ориентаций для разных размеров планшетов, то просто копируйте файл **refs.xml** в папки типа **values-720dp_land** и др.

В первой части мы узнали, что для создания фрагмента необходимо создать разметку, затем новый класс и в методе **onCreateView()** указать разметку. Затем в разметке активности указать тег **fragment** и присвоить ему имя класса фрагмента.

Часть вторая. Продолжение

Дополнительное чтение

Обсуждение статьи на форуме.