NickMikhalevich карма 3,0 рейтинг 0,0
3 голоса

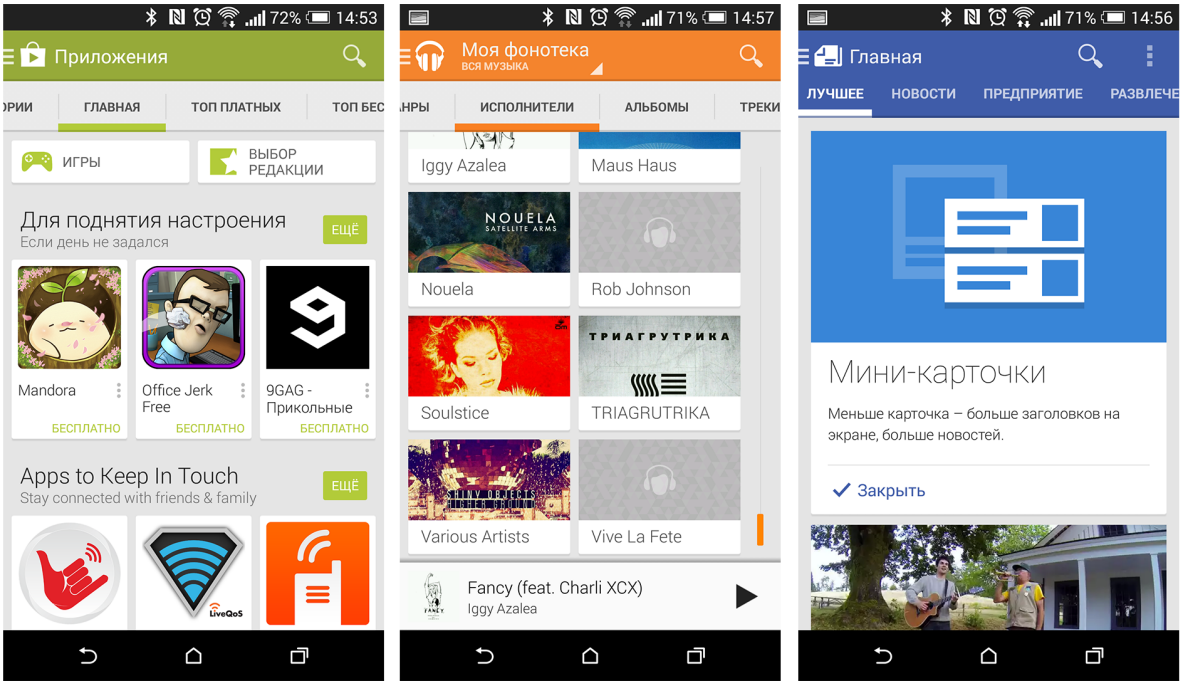
12 сентября 2014 в 11:46

Как легко сделать Navigation Drawer и вкладки, используемые в популярных приложениях от Google

из песочницы

Разработка под Android*, Mobile Development*

При использовании популярного приложения Play Маркет многие обратили внимание на вкладки для переключения контента. Такое применение вкладок можно найти и в других приложениях от Google, таких как Play Музыка, Play Пресса.



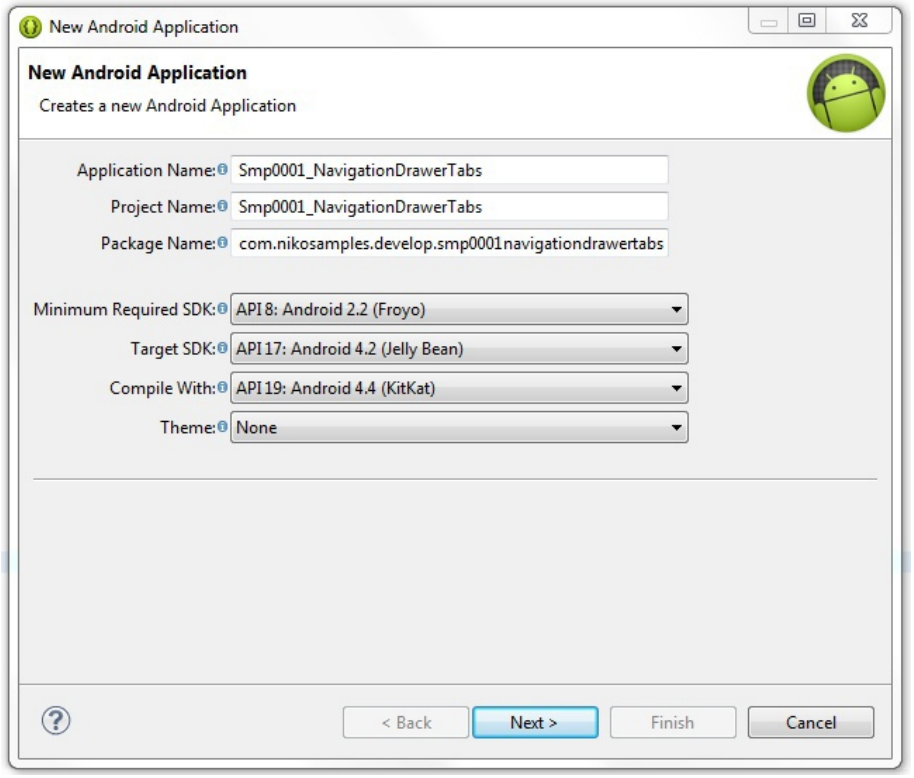
На этой почве возникает интерес, а иногда и необходимость (если заказчик просит) реализовать увиденное. Я не стал исключением и при проектировании нового приложения, дизайн которого был набросан на черновике, присутствовал очень схожий дизайн, хотя и имел всего несколько вкладок. Казалось бы, что сложного? Сейчас откроем официальную документацию, просмотрим необходимые разделы и приступим к делу. Но, изучив документацию, не смог обнаружить соответствующих примеров — и тут же возник новый вопрос. Почему Android разработчики из компании Google по умолчанию не предоставляют примеров с необходимой функциональностью, чтобы сделать это довольно просто, ведь это реализовано в каждом их приложении? Также, погуглив, нашлись аналогичные вопросы на Stack Overflow. Исходя из этого, оказалось, что существует проблема или, по крайней мере, нераскрытый вопрос, в котором следует разобраться.

Ниже хочу рассказать о том, как всё же можно реализовать паттерн Navigation Drawer вместе с вкладками, как в популярных приложениях от Google.

В примере будет использоваться интегрированная среда разработки Eclipse, но все действия можно будет воспроизвести, используя и другие среды, к примеру, недавно вышедшую и набирающую популярность Android Studio от компании Google, основанную на IntelliJ IDEA.

Создание проекта

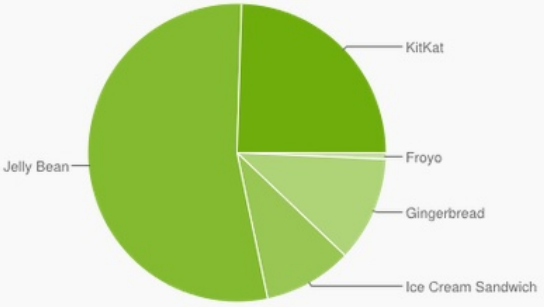
Создадим новый проект. Для этого перейдем в File > New > Android Application Project. Заполним поля, такие как имя приложения, пакета и версии SDK. Далее проследуем по экранам, нажимая клавишу Next и оставляя всё по умолчанию.



Среда разработки для нас создаст новый проект со стандартной структурой.

Пример будет работать, начиная с API версии 8. Это обосновано тем, что пользователи ещё пользуются девайсами со старой версией Android. Ниже приведены данные о количестве устройств, работающих под управлением различных версий платформы на состояние 12.08.2014.

Version	Codename	API	Distribution
2.2	Froyo	8	0.7%
2.3.3 - 2.3.7	Gingerbread	10	11.4%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	9.6%
4.1.x	Jelly Bean	16	25.1%
4.2.x		17	20.7%
4.3		18	8.0%
4.4	KitKat	19	24.5%



Action Bar для API 8

Но ActionBar, сочетающий в себе заголовок и меню, появился начиная с Android 3.0 (API 11). Для того, чтобы его использовать, необходимо подключить к проекту библиотеку Android-Support-v7-Appcompat, любезно предоставленную компанией Google. Детальную инструкцию по добавлению библиотеки к проекту можно найти по адресу: developer.android.com/tools/support-library/setup.html

Есть две возможности добавить библиотеку к проекту — без использования ресурсов и с использованием. В реализации этого проекта будет использоваться библиотека с использованием ресурсов. После того, как библиотека будет добавлена в дерево проектов, необходимо перейти в Properties, нажав по проекту правой клавишей мыши и выбрать в категориях Android, затем нажать клавишу Add. В появившемся списке выбрать android-support-v7-appcompat и нажать OK > Apply > OK. Библиотека добавлена в проект. Но если попытаться запустить приложение, то ActionBar будет ещё не виден. Необходимо в res/values/styles.xml изменить строчку:

```
<style name="AppBaseTheme" parent="android:Theme.Light">
```

в res/values-v11/styles.xml изменить строчку:

```
<style name="AppBaseTheme" parent="android:Theme.Holo.Light ">
```

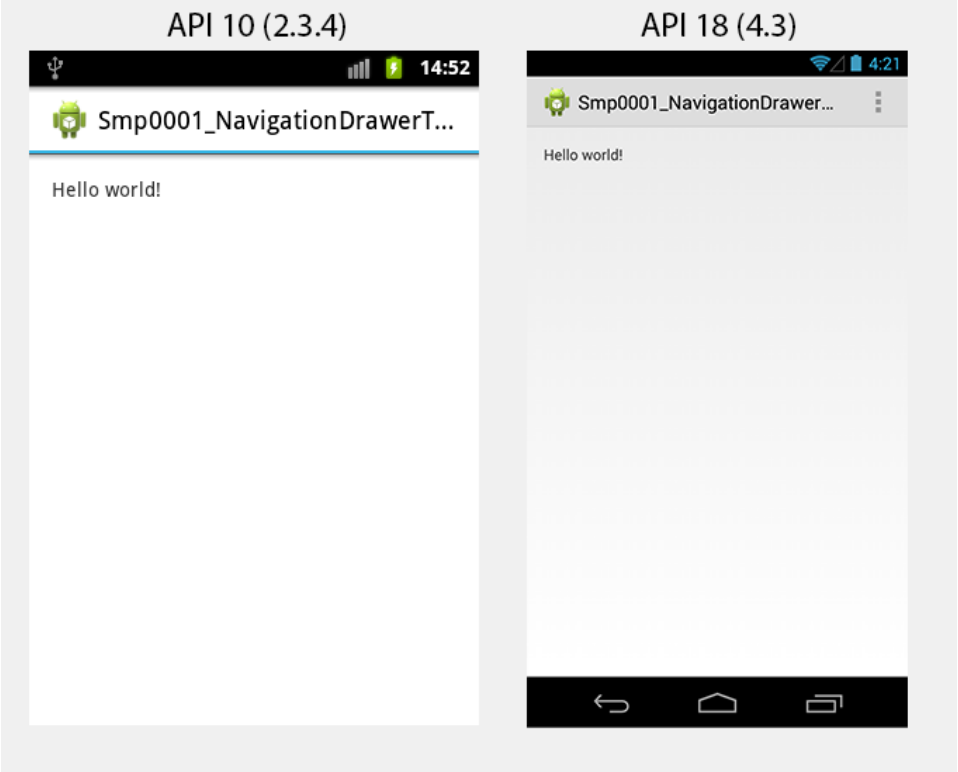
в res/values-v14/styles.xml изменить строчку:

```
<style name="AppBaseTheme" parent="android:Theme.Holo.Light.DarkActionBar">
```

на

```
<style name="AppBaseTheme" parent="@style/Theme.AppCompat.Light">
```

Также в главной активности необходимо наследоваться не от Activity, а от ActionBarActivity (android.support.v7.app.ActionBarActivity). После проделанных действий и запуска приложения можно увидеть ActionBar, включая и на ранних версиях API.



Зайдем в папку Menu и отредактируем файл main.xml, чтобы выглядел следующим образом:

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:sabd="http://schemas.android.com/apk/res-auto" >

    <item
        android:id="@+id/action_search"
        android:icon="@android:drawable/ic_menu_search"
        android:orderInCategory="100"
        android:title="@string/action_search"
        sabd:showAsAction="ifRoom"/>

</menu>
```

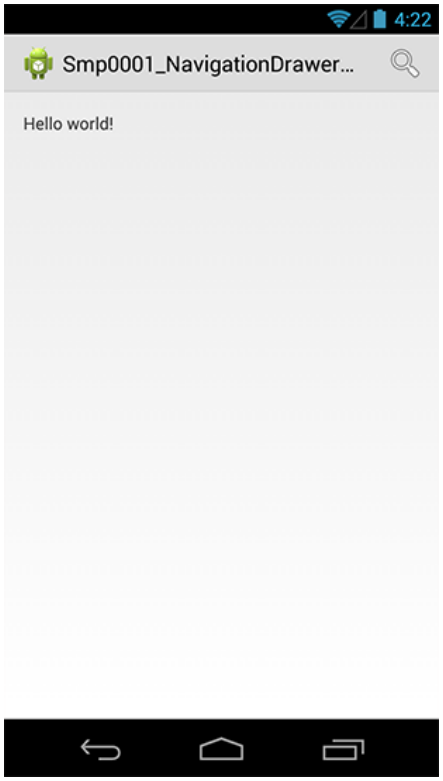
Необходимо обратить внимание на следующую строчку: xmlns:sabd=http://schemas.android.com/apk/res-auto. Теперь необходимо атрибут showAsAction задавать следующим образом: sabd:showAsAction=" ". Также следует зайти в strings.xml и изменить строчку:

```
<string name="action_settings">Settings</string>
```

на

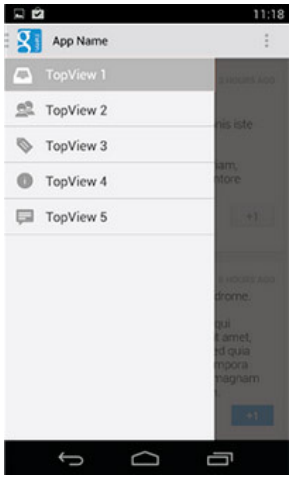
```
<string name="action_search">Search</string>
```

Теперь меню будет иметь привычный вид.



Внедрение бокового меню

Следующим нашим шагом будет внедрение бокового меню (Navigation Drawer). Это панель, которая отображает основные параметры навигации приложения в левом краю экрана. Раскрывается жестом от левого края экрана или по нажатию на значок приложения в панели действий.



Изменим основной ресурс activity_main.xml:

```
<android.support.v4.widget.DrawerLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
         android:id="@+id/drawer_layout"
         android:layout_width="match_parent"
         android:layout_height="match_parent">
          <!-- The main content view -->
         <FrameLayout
                 android:id="@+id/content_frame"
                 android:layout_width="match_parent"
                  android:layout_height="match_parent" />
          <!-- The navigation drawer -->
        <ListView android:id="@+id/left_drawer"
                 android:layout_width="240dp"
                        android:layout_height="match_parent"
                        android:layout_gravity="start"
                 android:choiceMode="singleChoice"
```

```
                    android:divider="@android:color/transparent"
                    android:dividerHeight="0dp"
                    android:background="#111"/>
</android.support.v4.widget.DrawerLayout>
```

Боковое меню будет заполняться в ListView, для этого добавим в string.xml строковый массив названий:

```
<string-array name="screen_array">
        <item>Screen 1</item>
        <item>Screen 2</item>
        <item>Screen 3</item>
    </string-array>
```

Необходимо определить, как будет выглядеть позиция в ListView. Для этого создадим в папке layout новый ресурс с названием drawer_list_item.xml:

```
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@android:id/text1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="@drawable/activated_background"
    android:gravity="center_vertical"
    android:minHeight="?attr/listPreferredItemHeightSmall"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:textAppearance="?android:attr/textAppearanceMedium"
    android:textColor="#fff" />
```

Для функционирования работы созданного ресурса далее дополнительно создадим в папке res новую папку drawable и в ней создадим селектор activated_background.xml, поместив в него:

```
<selector xmlns:android="http://schemas.android.com/apk/res/android">

    <item android:drawable="@color/blue" android:state_activated="true"/>
    <item android:drawable="@color/blue" android:state_selected="true"/>
    <item android:drawable="@color/blue" android:state_pressed="true"/>
    <item android:drawable="@color/blue" android:state_checked="true"/>
    <item android:drawable="@android:color/transparent"/>

</selector>
```

В папке values создадим ресурс для цветов color.xml и поместим туда цвет, который будет отвечать за выделение пункта списка в боковом меню:

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

    <item name="blue" type="color">#FF33B5E5</item>

</resources>
```

Следующим шагом будет добавление иконок в приложение, а именно значка бокового меню. Скачать архив иконок можно по прямой ссылке с официального сайта Google по адресу: developer.android.com/downloads/design/Android_Design_Icons_20130926.zip. В архиве будет многочисленное число иконок для разных событий, но нужны иконки из папки Navigation_Drawer_Indicator. Следует каждый графический объект с названием ic_drawer.png поместить в проект с правильной плотностью вида drawable-???..

Для оповещения о том, что меню открыто или закрыто, добавим в string.xml ещё записи:

```
<string name="drawer_open">Open navigation drawer</string>
<string name="drawer_close">Close navigation drawer</string>
```

Заодно удалим из ресурсов следующую строчку, так как она нам уже не понадобится:

```
<string name="hello_world">Hello world!</string>
```

Главное Activity

Теперь необходимо переписать класс MainActivity. Для поддержки старых устройств мы используем библиотеку поддержки Android Support Library (v4), при создании проекта она автоматически добавляется в папку libs. В листинге присутствуют комментарии, которые смогут дать возможность понять, как работает код. Для дополнительной информации можно воспользоваться официальной документацией: developer.android.com/training/implementing-navigation/nav-drawer.html. Ниже листинг.

▸ Листинг

Добавим фрагменты

Добавим новый пакет fragments, который будет содержать фрагменты. И поместим в него три класса-фрагмента. Для примера я выложу код одного фрагмента, остальные необходимо сделать по аналогии, изменив название класса, конструктора и разметки.

```
package com.nikosamples.develop.smp0001navigationdrawertabs.fragments;

import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import com.nikosamples.develop.smp0001navigationdrawertabs.R;

public class ScreenOne extends Fragment {

    public ScreenOne() {
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
            Bundle savedInstanceState) {

        View rootView = inflater.inflate(R.layout.screen_first, container,
                false);

        return rootView;
    }

}
```

Также добавим в ресурсы разметку для этих классов. Для примера, выложу разметку одного фрагмента screen_one.xml. В остальных необходимо изменить только текст атрибута android:text:

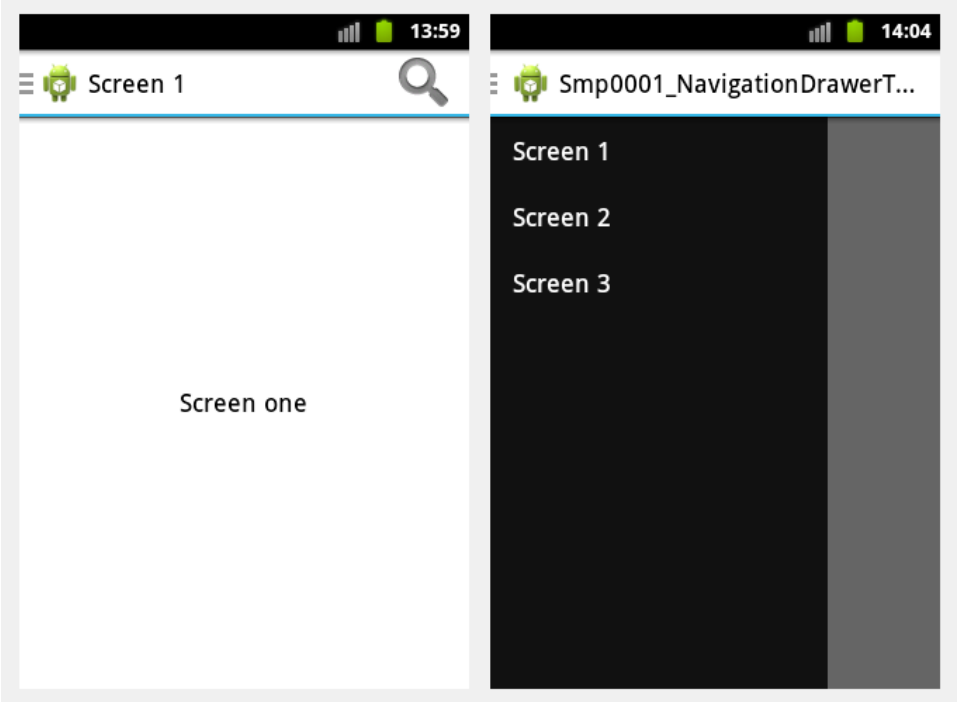
```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
```

```
    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:text="@string/screen_one"
        android:textAppearance="?android:attr/textAppearanceMedium" />

</RelativeLayout>
```

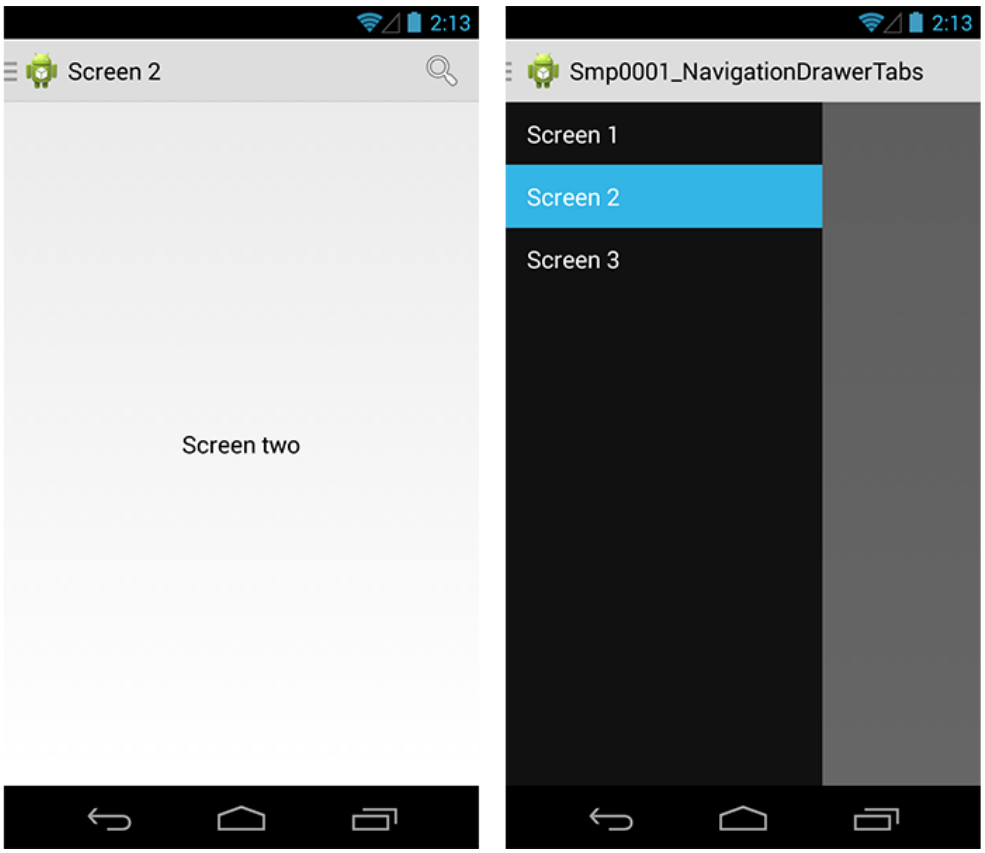
Добавим в string.xml ещё несколько строчек, которые будут информировать о том, какой экран открыт из меню:

```
<string name="screen_one">Screen one</string>
<string name="screen_two">Screen two</string>
<string name="screen_three">Screen three</string>
```

На этом этапе реализация Navigation Drawer завершена. Если запустить приложение, мы увидим работу бокового меню. Как на старых устройствах:



Так и на новых:



Внедрение вкладок

Теперь приступаем к добавлению вкладок. Первым делом, чтобы нам правильно добавить вкладки, необходимо понять принцип расположения элементов навигации. Существует строгая иерархия, в которой должны располагаться вкладки:

0 — Navigation Drawer занимает самый верхний уровень навигации.
1 — Action Bar второй уровень.
2 — Вкладки нижний уровень.

Для реализации необходимо использовать два дополнительных класса, которые Google снова любезно предоставляет. Это классы SlidingTabLayout и SlidingTabStrip. Добавим их в проект. Для этого создадим новый пакет view, там создадим новые классы с соответствующим названием и переместим в них код. При возникновении ошибок в методе createDefaultTabView(Context context) класса SlidingTabLayout следует подавить предупреждение, дописав над методом @SuppressLint(«NewApi»)

Внесем все новые изменения для фрагмента ScreenOne. Первым делом изменим разметку screen_one.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:orientation="vertical">

    <com.nikosamples.develop.smp0001navigationdrawertabs.view.SlidingTabLayout
          android:id="@+id/sliding_tabs"
          android:layout_width="match_parent"
          android:layout_height="wrap_content" />

    <android.support.v4.view.ViewPager
          android:id="@+id/viewpager"
          android:layout_width="match_parent"
          android:layout_height="0px"
          android:layout_weight="1"
          android:background="@android:color/white"/>

</LinearLayout>
```

Важно использовать полное имя пакета для SlidingTabLayout, так как он включен в наш проект. Далее создадим

новую разметку в папке layout для вкладок pager_item.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    android:orientation="vertical" >

    <TextView
        android:id="@+id/item_subtitle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/page"/>
        android:textAppearance="?android:attr/textAppearanceLarge" />

    <TextView
        android:id="@+id/item_title"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="80sp" />

</LinearLayout>
```

Войдем в string.xml и изменим строчку:

```
<string name="screen_one">Screen one</string>
```

на

```
<string name="page">Page:</string>
```

Так как нам уже не понадобится строковый ресурс, вместо него мы сразу отобразим ViewPager с номером вкладки. Далее изменим класс ScreenOne соответствующим образом:

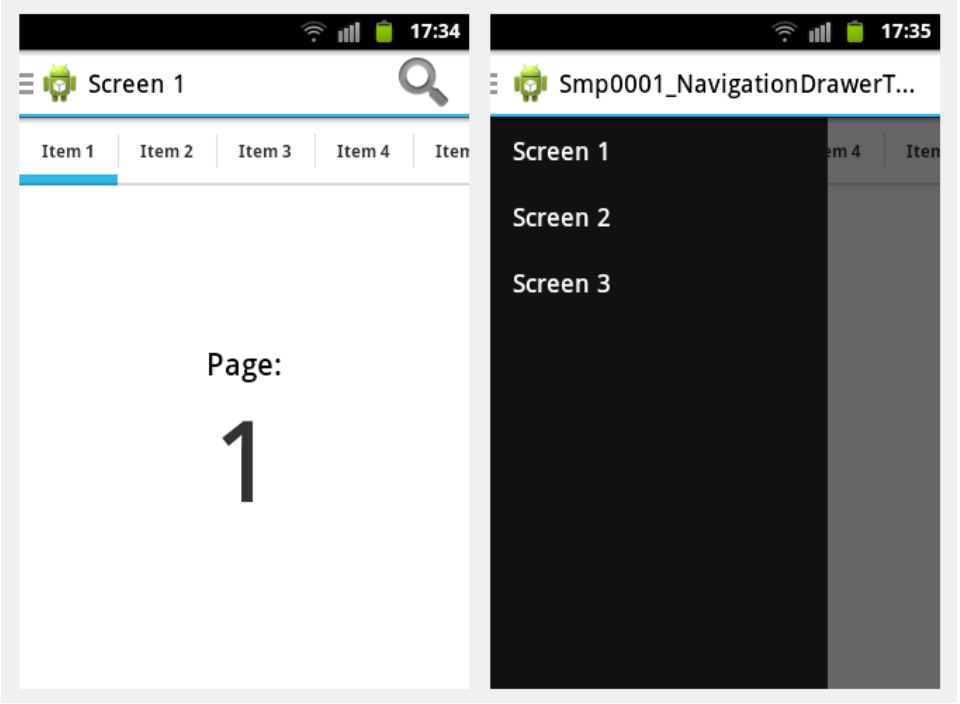
▼ Код класса ScreenOne

```
package com.nikosamples.develop.smp0001navigationdrawertabs.fragments;

import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.support.v4.view.PagerAdapter;
import android.support.v4.view.ViewPager;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import com.nikosamples.develop.smp0001navigationdrawertabs.R;
import com.nikosamples.develop.smp0001navigationdrawertabs.view.SlidingTabLayout;

public class ScreenOne extends Fragment {

    private SlidingTabLayout mSlidingTabLayout;
    private ViewPager mViewPager;

    public ScreenOne() {
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
```

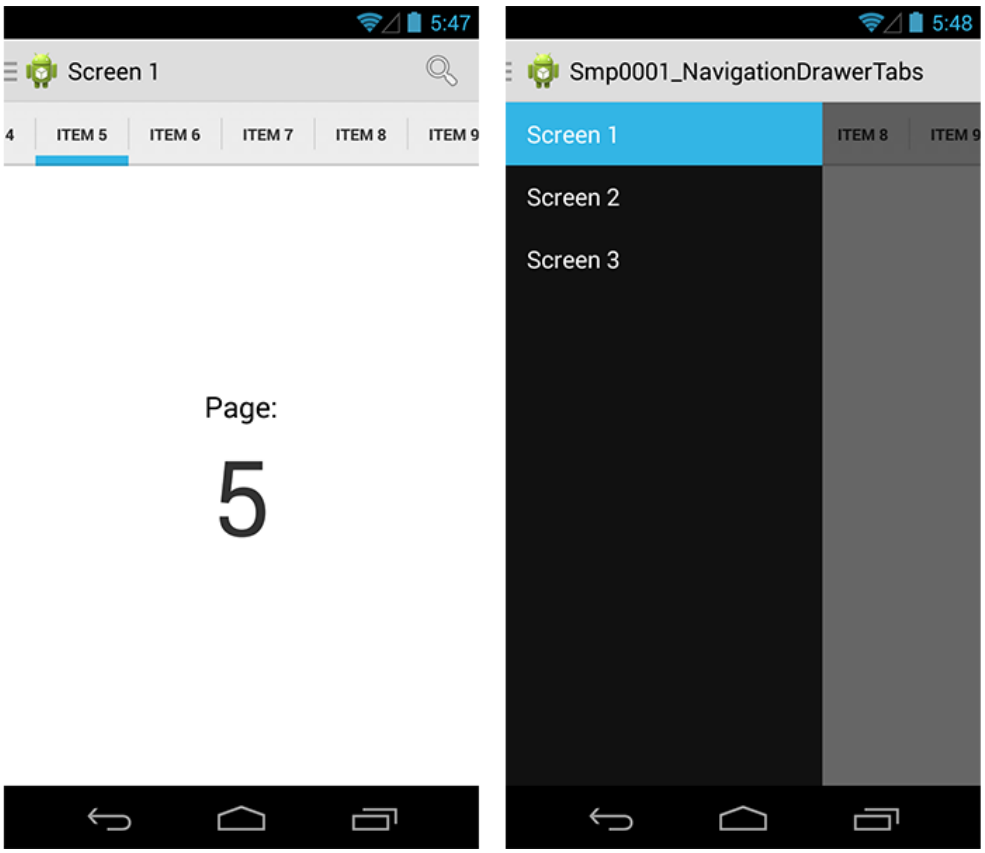
```
        Bundle savedInstanceState) {

    View rootView = inflater.inflate(R.layout.screen_one, container, false);

    return rootView;
}

@Override
public void onViewCreated(View view, Bundle savedInstanceState) {
    // Get the ViewPager and set it's PagerAdapter so that it can display items
    mViewPager = (ViewPager) view.findViewById(R.id.viewpager);
    mViewPager.setAdapter(new SamplePagerAdapter());

    // Give the SlidingTabLayout the ViewPager, this must be
    // done AFTER the ViewPager has had it's PagerAdapter set.
    mSlidingTabLayout = (SlidingTabLayout) view.findViewById(R.id.sliding_tabs);
    mSlidingTabLayout.setViewPager(mViewPager);
}

// Adapter
class SamplePagerAdapter extends PagerAdapter {

    /**
     * Return the number of pages to display
     */
    @Override
    public int getCount() {
        return 10;
    }

    /**
     * Return true if the value returned from is the same object as the View
     * added to the ViewPager.
     */
    @Override
    public boolean isViewFromObject(View view, Object o) {
        return o == view;
    }

    /**
     * Return the title of the item at position. This is important as what
     * this method returns is what is displayed in the SlidingTabLayout.
     */
    @Override
    public CharSequence getPageTitle(int position) {
        return "Item " + (position + 1);
    }

    /**
     * Instantiate the View which should be displayed at position. Here we
     * inflate a layout from the apps resources and then change the text
     * view to signify the position.
     */
    @Override
    public Object instantiateItem(ViewGroup container, int position) {
        // Inflate a new layout from our resources
        View view = getActivity().getLayoutInflater().inflate(R.layout.pager_item,
                container, false);
        // Add the newly created View to the ViewPager
        container.addView(view);

        // Retrieve a TextView from the inflated View, and update it's text
        TextView title = (TextView) view.findViewById(R.id.item_title);
        title.setText(String.valueOf(position + 1));

        // Return the View
        return view;
```

```
        }

        /**
         * Destroy the item from the ViewPager. In our case this is simply
         * removing the View.
         */
        @Override
        public void destroyItem(ViewGroup container, int position, Object object) {
            container.removeView((View) object);
        }
    }
}
```

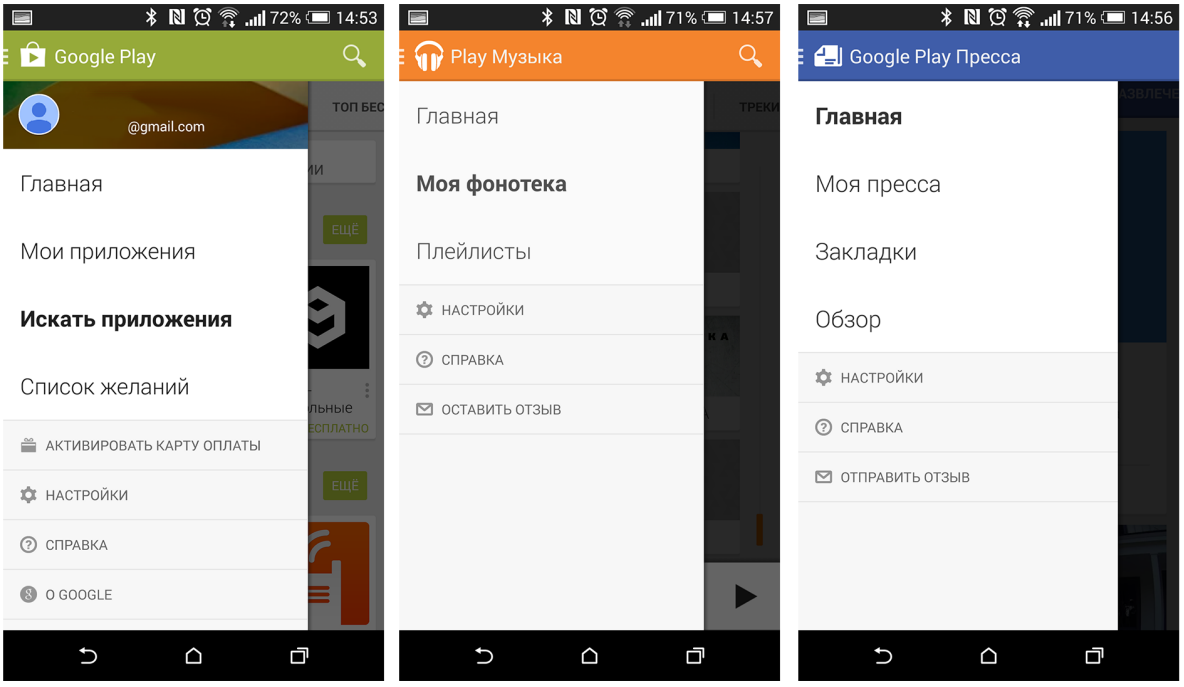
Теперь можно запустить приложение и увидеть, как работают вкладки как на старом устройстве:



Так и на новом:



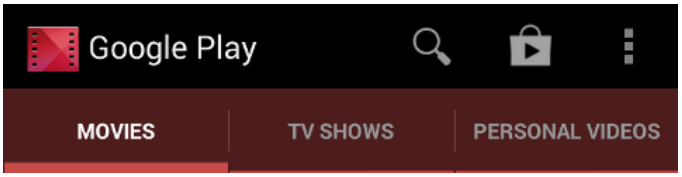
Можно заметить, что боковое меню покрывает вкладки, как в правилах навигации и приложениях от Google:



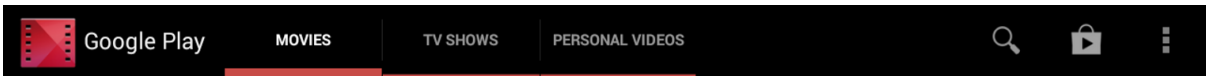
На этом пример полностью завершён и его можно использовать.

Вкладки Action Bar

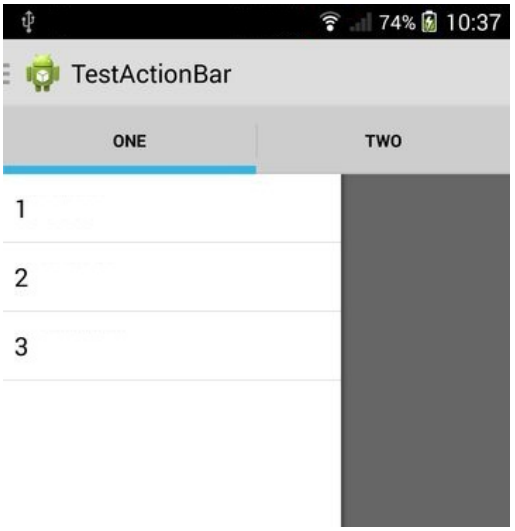
Хочется отметить ещё один последний, важный момент. Многие «путают» реализованные вкладки с вкладками ActionBar, они выглядят похоже:



Но у них реализация другая, поведение и в горизонтальной ориентации переносятся в ActionBar:



Если добавить вкладки через ActionBar, то боковое меню Navigation Drawer не перекроет вкладки, а выедет под ними:



На этом всё. Спасибо за внимание и приятного вам кодинга.

Среда разработки – Eclipse
Минимальная версия Android – >= 8

Ссылки

Dashboards
developer.android.com/about/dashboards/index.html?utm_source=ausdroid.net
Support Library
developer.android.com/tools/support-library/index.html
Creating a Navigation Drawer
developer.android.com/training/implementing-navigation/nav-drawer.html
Android Design in Action: Navigation Anti-Patterns
plus.google.com/photos/+NickButcher/albums/5981768132040708401
SlidingTabsBasic
developer.android.com/samples/SlidingTabsBasic/index.html
Action Bar
developer.android.com/guide/topics/ui/actionbar.html

Готовый пример можно скачать с GitHub.

android, google, tabs, tutorial, actionbar, navigation drawer, support library

+37 41478 453 NickMikhalevich 0,0

Комментарии (25)

hazanko 12 сентября 2014 в 14:21 # +2

Все хорошо и автор молодец, но античные версии андроида уже давно пора перестать использовать.

maseal 12 сентября 2014 в 14:34 # ↑ +2

При создании приложения не всегда можно с легкостью отказаться от 1/8 потенциальной аудитории (тех, у которых android версии < 4.0).

dark4eg 12 сентября 2014 в 14:57 # ↑ 0

пусть выкинут свой калькулятор)

Black_Shadow 13 сентября 2014 в 20:52 # ↑ 0

Они скорее приложение выкинут. С чего вдруг они должны выкидывать рабочий и вполне устраивающий их телефон?

bask 17 марта 2015 в 17:45 # ↑ 0

Ради этих 10% людей никто не будет заниматься геморроем. Пусть выбрасывают приложение. Этих людей мало и их не жалко потерять.

hazanko 12 сентября 2014 в 15:01 # ↑ +2

На мой взгляд отказ от версий < 4.0 дает куда больше профита, чем расчет на мнимую 1/8-ю аудитории.

Mikhail_dev 12 сентября 2014 в 15:11 # ↑ +4

Очень даже можно, если приложение пишется с нуля. Если приложение уже адаптировано под 2.3, то выбрасывать конечно нет смысла.

zorgzerg 12 сентября 2014 в 16:03 # ↑ 0

Есть подозрение, что человек юзающий 2.2 Android до сих пор не очень-то увлекается новинками, и скорее всего использует свой смартфон As Is, и даже может быть не подключен к маркету :) Сам знаком с несколькими такими — часто это люди пожилого и предпожилого возраста, которым пофиг на модные фишки и все такое. Считаю что сейчас достаточно поддерживать устройства начиная с Ice Cream, а может и с ~~Billie Jean~~ Jelly Bean

grishkaa 14 сентября 2014 в 02:51 # ↑ +1

Как показывает практика, пользователей 2.x хоть и 15%, но они намного менее активны в плане пользования приложениями и вообще интернетом с телефона. Все самые активные уже давно перешли на более нормальные версии. И, да, отсутствие нативных фрагментов, аниматоров и аппаратного ускорения — боль.

Newbilius 12 сентября 2014 в 15:55 # +1

Я ещё год назад считал, что 2.3 ещё можно поддерживать.
Сейчас уже дико сомневаюсь — аудитория не такая уж большая, а проблем с поддержкой выше крыши…

khim 12 сентября 2014 в 15:59 # ↑ +3

Главное в другом: 2.3 сейчас ставится только на супербюджетные андроиды. То есть их покупают люди либо жадные, либо бедные, либо те, кому нужен телефон, а не смартфон. В любом случае заработать на них не удастся ну никак. Тогда зачем тратить время и силы?

Newbilius 12 сентября 2014 в 17:21 # ↑ 0

Тут чуууть чуть поспорю: приложения порой делаются и бесплатные, для продвижения бренда и т.п. И человек, экономящий на телефоне, не факт что не сможет стать пользователем вашего основного продукта-товара. Другое дело что да, год-два назад android 2.3 можно было встретить и на средних моделях, а не только бюджетных. Сегодня же да, на него наткнуться куда сложнее.

sapl 13 сентября 2014 в 09:58 (комментарий был изменён) # ↑ +1

К сожалению, сейчас говнофонов за 3000р и с 4+ Android стало предостаточно.
И да, гемороя эти пользователи доставляют 90% и колы только в путь ставят в рейтинг.
Я бы многое отдал за фильтр не по версии, в по цене в Google Play.

Tishka17 12 сентября 2014 в 17:08 # +2

В статье упущен момент, что боковое меню должно быть доступно из всех активностей приложения и при нажатиях в нем, должно очищать стэк активностей. Что можете посоветовать для удобной реализации такого требования?

По поводу вкладок в ActionBar: в последней версии Hangouts боковое меню как раз вылезает ниже вкладок. С другой стороны, на скриншотах из документации по Material Design боковое меню навигации вылезает поверх ActionBar.

livotov 12 сентября 2014 в 17:26 # ↑ 0

> Что можете посоветовать для удобной реализации такого требования?

Запускайте новую активити из меню с соотв. флагом для очистки стека.
Если пользуетесь одной активити и фрагментами — чистите стек сами через FragmentManager

lukaville 12 сентября 2014 в 17:39 # ↑ 0

По поводу бокового меню: судя по всему Material Design в основном расчитан на Android L. Это можно проследить в их приложении для конференции Google I/O. Если запустить это приложение на устройствах с Android L, то Action Bar вылезает поверх ActionBar, а если запустить на Android <L (или отключив тему Material), то navigation drawer выглядит как обычно, снизу Action Bar.

jaleel 12 сентября 2014 в 23:53 # ↑ 0

Обещали поддержку в support library вплоть до 7 версии.

jaleel 12 сентября 2014 в 23:55 # ↑ 0

AUF – Always Use Fragments

sapl 13 сентября 2014 в 10:02 # ↑ 0

> меню должно быть доступно из всех активностей приложения

Не из всех.
Только там, где это логично.
Посмотрите тот же экран настроек в Google Play.

kozhevnikovv 12 сентября 2014 в 22:18 # –1

Хотелось бы добавить, что при листании «тяжелых» фрагментов с помощью NavigationDrawer, анимация закрытия последнего может немного подлагивать, что портит впечатление от приложения. Для себя я эту проблему решил следующим образом:

```
private void selectItem(final int position) {
    mDrawer.closeDrawer();
    new Handler().postDelayed(new Runnable() {
        @Override
        public void run() {
            Fragment fragment = new myFragment();
            Bundle args = new Bundle();
            fragment.setArguments(args);

            FragmentManager fragmentManager = getSupportFragmentManager();
            fragmentManager.beginTransaction().replace(R.id.content).commit();

        }
    }, 250);
}
```

То есть сначала закрываем навигацию, а потом меняем фрагмент. Как по мне это выглядит намного лучше.

sapl 13 сентября 2014 в 10:08 (комментарий был изменён) # ↑ +1

Не лучший совет.
Поддрагивает анимация потому, что отрисовка нового экрана/элемента меню происходит не мгновенно.
(от 50 до 300 мс). В вашем случае вы опытным путем нашли для себя 250мс.
В первую очередь надо оптимизировать это.

Если никак не выходит, то использовать события открытия/закрытия меню,
чтобы анимация не накладывалась на пересчеты лейаутов.

```
public void onDrawerClosed(View view) {
}
public void onDrawerOpened(View view) {
}
```

jaleel 13 сентября 2014 в 00:02 # +3

Немного непонятно зачем статья. Начинающему легче поставить Android Studio и сгенерить уже готовый Activity с Navigation drawer, а как потом добавить SlidingTabLayout и ViewPager можно и в интернетах почитать. И кстати, вместо ListView я бы лучше сделал FrameLayout, в который бы пихал NavigationDrawerFragment, так как можно в этот фрагмент отдельную логику вынести.

tumbler 15 сентября 2014 в 10:27 # ↑ 0

Начинающий начнет гуглить боковое меню и вкладки, и вообще не узнает про SlidingTabLayout. Даешь больше «правильных» связок подходов и компонентов на хабре!

0

NickMikhalevich 15 сентября 2014 в 12:47 # ↑

Новый шаблон Navigation Drawer Activity в Android Studio появился только в версии 0.5.6. Очень радует, что студия активно развивается, становится стабильней, получает новые шаблоны и функционал. Но не каждый начинающий разработчик, сможет увидеть у себя в студии этот шаблон, если он конечно не обновил студию если не до последней версии, то до 0.5.6. И ведь Eclipse ещё пользуется популярностью имея богатый функционал и множество подключаемых модулей.

jaleel 15 сентября 2014 в 13:38 # ↑ 0

На сайте сейчас Android Studio 0.8.6 и думаю есть очень большая вероятность, что именно со студии сейчас многие начинают. Ну а то, что не увидит — там тяжело не увидеть, при создании проекта или активити тебе все предлагается.

Только зарегистрированные пользователи могут оставлять комментарии. Войдите, пожалуйста.