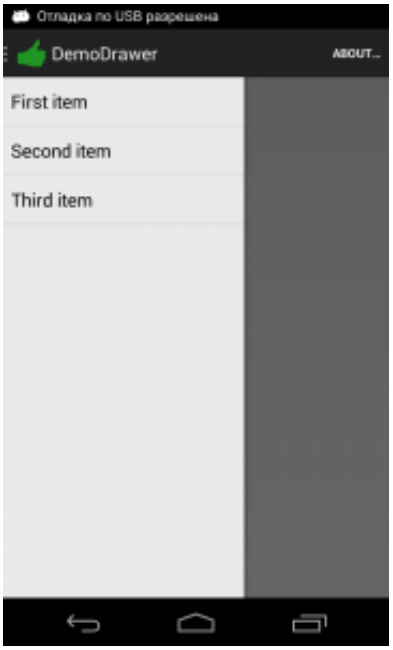
Октетология

ненаука о байтах

DrawerLayout: панелька из Гугла

Похоже, Google всё серьёзнее берётся за унификацию внешнего вида и поведения приложений под Android. Нет, серьёзных репрессивных мер пока что не ожидается, но правилами Google Play с недавнего времени жёстко запрещён вносящий путаницу софт (и, хвала богам, AirPush тоже наконец-то!), а раздел Design на Android Developers регулярно пополняется новыми подробными рекомендациями. И не просто теоретическими, а подкреплёнными набором инструментов для разработчиков!



DrawerLayout в действии, демо-приложение

Одним из рекомендуемых (и уже знакомых) подходов к построению интерфейса является Navigation Drawer — панелька, выезжающая сбоку по нажатию на action bar или по свайпу с края экрана. Для её реализации готовы все технические средства, и написан официальный мануал разработчика, — но, пожалуй, слишком многословный и при этом не затрагивающий ряда тонкостей. Потому здесь я попробую исправить этот недочёт.

Где брать?

Одним из подходов, активно применяемых сейчас Google, является включение всех новых контролов в Android Support Library, — и только туда, без реализации «родной» версии в новых ОС. Именно в `android.support.v4` появился когда-то сверхпопулярный ViewPager, встречающийся ныне в каждом втором приложении; там же, как нетрудно догадаться, лежит и требующийся нам DrawerLayout (не путать с устаревшим SlidingDrawer!). Это позволяет заранее избежать проблем совместимости, пусть и ценой «размазывания» кода внутрь .apk.

Дополнительных библиотек (кроме support library) подключать не потребуется. Однако не стоит забывать, что графика для Android не может быть нормально включена в jar-файлы, потому иконку для ActionBar нужно добавить в ресурсы проекта вручную. Готовые иконки для стилей Holo и Holo.Light можно скачать с официального сайта: Action Bar Icon Pack или, в нестандартных случаях, воспользоваться генератором из Android Asset Studio. В дальнейшем будем предполагать, что иконка названа так же, как в официальном паке: `ic_drawer.png`.

Графика drawer-а

Если drawer должен отбрасывать тень, можно воспользоваться её готовым изображением: `drawer_shadow.9.png`. Не забудьте скопировать этот (или какой-то свой) файл в ресурсы приложения.

Как нарисовать?

И вот мы плавно подошли к началу начал: правильному layout-файлу. Да, слово «правильный» здесь не случайно — именно некорректный layout служит причиной большинства проблем при знакомстве с этим контролом, а в официальной документации внимание на важных деталях не всегда заострено.

В общем и целом принцип прост: первый дочерний контрол у DrawerLayout должен служить областью контента (то, что пользователь видит при **закрытом** drawer-е), второй дочерний контрол — выезжающей панелькой. Именно так: первый — лежащий снизу; по тому же принципу, что и слои на FrameLayout. Но не спешите — просто так накидав контролы в дизайнере вы получите при запуске следующее замечательное сообщение об ошибке:

java.lang.IllegalArgumentException: No drawer view found with absolute gravity LEFT

Это связано с тем, что DrawerLayout ищет «свои» менюшки всё же не по очерёдности, а по layout gravity. Потому следует задать нужному дочернему контролу (т.е. выезжающей панельке) правильную «гравитацию» — Gravity.START.

```
android:layout_gravity="start"
```

Лучше задавать именно `start`, а не `left` — это универсальное решение, облегчающее в перспективе поддержку RTL-раскладок. Не пугайтесь того, что это значение появилось только с API 14 — всё будет работать и на более низких уровнях, поскольку ресурсы компилируются в binary XML, а для числовых констант версия платформы безразлична.

Тут возникнет вопрос — **но если важен только этот атрибут, то зачем же тогда менюшка должна быть именно второй?** Лучший ответ — практика: поменяйте дочерние контролы местами, посмотрите на результат. Скорее всего — drawer перестанет реагировать на жесты, открываясь и закрываясь только по прямым вызовам из кода. Если нужно именно такое поведение — пользуйтесь этим как готовым хаком, но вряд ли такой паттерн использования будет частым.

```
<android.support.v4.widget.DrawerLayout
     xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/drawer"
     >

    <!-- Область контента, текст в ней -->
    <FrameLayout
         android:id="@android:id/content"
        android:layout_width="match_parent"
        android:layout_height="match_parent" >

        <TextView
             android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/demo"
            android:layout_gravity="center"
            android:textAppearance="?android:attr/textAppearanceLarge" />

    </FrameLayout>

    <!-- Меню drawer-а -->
    <ListView
         android:id="@+id/list_menu"
        android:layout_width="240dp"
        android:layout_height="match_parent"
        android:entries="@array/menu_items"
        android:layout_gravity="start"
        android:background="@drawable/menu_background"
        android:textColor="@android:color/white"
     />

</android.support.v4.widget.DrawerLayout>
```

Другой важный нюанс — не задавайте padding в DrawerLayout, результат получается страшноватым и неожиданным. Кроме того, при возможности, укажите точное значение `layout_width` для выезжающего меню (например, `240dp`; документация советует не задавать более чем `320dp` без дополнительных проверок экрана). Иначе панелька расползётся и займёт слишком много места в альбомной ориентации.

Как стартовать?

Итак, мы благополучно применяем созданный layout — и... ничего, конечно же, ещё не работает как надо. Нет, мы уже можем открывать drawer свайпом по экрану, но он пока ещё слишком плоский и не реагирует на action bar. Для полноценной работы нужна небольшая инициализация и несколько оповещений о конфигурации.

Итак, первым делом создадим ActionBarDrawerToggle — это класс, отвечающий за те самые индикаторные «полосочки» в action bar-е, картинки для которых мы добавляли на подготовительном этапе. Затем взаимно свяжем созданный объект с нашим drawer-ом (они автоматически обменяются командами и событиями), а напоследок разрешим кнопку в ActionBar-е, если уже не сделали этого в используемой теме.

```
@SuppressLint("InlinedApi")
private void initDrawer() {
    mDrawer = (DrawerLayout) findViewById(R.id.drawer);

    // Если drawer не используется, то не паникуем
    if (mDrawer == null)
        return;

    // Создадим drawer toggle для управления индикатором сверху
    mDrawerToggle = new ActionBarDrawerToggle(this, mDrawer,
        R.drawable.ic_drawer, R.string.opened, R.string.closed);

    // Назначим его drawer-у как слушателя
    mDrawer.setDrawerListener(mDrawerToggle);

    // Для красоты добавим тень с той же гравитацией
    mDrawer.setDrawerShadow(R.drawable.drawer_shadow, Gravity.START);

    // Включим кнопки на action bar
    this.getActionBar().setDisplayHomeAsUpEnabled(true);
    this.getActionBar().setHomeButtonEnabled(true);
}

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

    // Вызовем инициализацию нашего drawer-а
    initDrawer();

    // Этот код обрабатывает нажатия на пункты списка в выезжающей панели.
    // По такому нажатию мы будем закрывать drawer.
    mDrawerMenu = (ListView) findViewById(R.id.list_menu);
    mDrawerMenu.setOnItemClickListener(new OnItemClickListener() {
        @Override
```

```
        public void onItemClick(AdapterView<?> listView, View item, int position, long
            if (mDrawer != null)
                mDrawer.closeDrawers();
        }
    });
}
```

Достаточно ли этого? Пока ещё нет: необходимо доверить созданному ActionBarDrawerToggle управление связанными с drawer-ом событиями. Реализуется это очень просто: из `onConfigurationChanged()` и `onOptionsItemSelected()` вызываем одноименные методы созданного экземпляра ActionBarDrawerToggle.

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Если событие обработано переключателем, то выходим
    if (mDrawerToggle != null && mDrawerToggle.onOptionsItemSelected(item))
        return true;

    // Иначе — всё как обычно
    return super.onOptionsItemSelected(item);
}

@Override
public void onConfigurationChanged(Configuration newConfig) {
    super.onConfigurationChanged(newConfig);

    // Просто вызов
    if (mDrawerToggle != null)
        mDrawerToggle.onConfigurationChanged(newConfig);
}
```

Кроме того, не забудем на всякий случай синхронизировать состояние индикатора и drawer-а после инициализации активити:

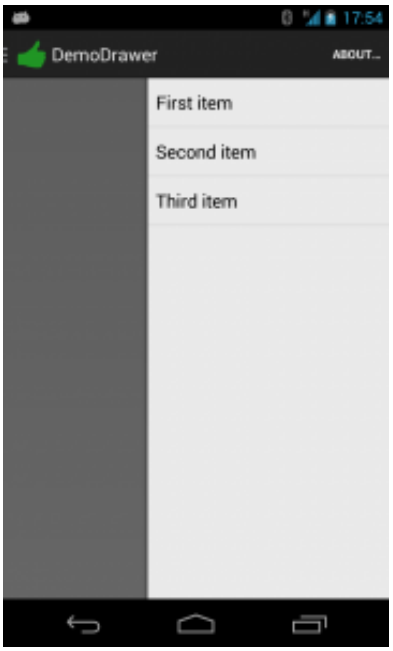
```
@Override
protected void onPostCreate(Bundle savedInstanceState) {
    super.onPostCreate(savedInstanceState);

    if (mDrawerToggle != null)
        mDrawerToggle.syncState();
}
```

И вот теперь-то в изысканиях можно ставить точку. Это всё. Правда — всё. Теперь оно работает: открывается по нажатию на action bar, или по свайпу через край экрана.

Или есть ещё что-то?

А несколько drawer-ов, слева и справа?

Такой паттерн демонстрирует, например, официальный клиент Google+. Выезжающая панелька слева содержит меню категорий, справа — список оповещений. Сложно ли такое сделать? Категорически **нет**!



Drawer справа в том же приложении

Именно здесь и кроется причина поиска drawer-а по его gravity, а не по его позиции. Чтоб получить вторую выезжающую панель, следует всего лишь добавить второй дочерний элемент в DrawerLayout и установить его `layout_gravity` как `end`. И, **в принципе**, дополнительной настройки для правого drawer-а уже не нужно — разве только установить для него подходящую тень, полученную при помощи поворота имеющейся картинки на 180° (не вручную, конечно, а средствами системы, подробнее здесь: Android: повернуть картинку через XML).

```
mDrawer.setDrawerShadow(R.drawable.shadow_mirrored, Gravity.END);
```

Впрочем, есть нюансы. Например, на экране можно увидеть обе панели одновременно, а в «эталонном» для нас приложении Google+ одна панель соглашается выезжать только когда полностью скрылась вторая. Кроме того, индикатор в приложениях Google работает только для основной панели, а у нас срабатывает на любую (это можно заметить на скриншоте выше: хорошо видно, что «полосочки» спрятались). В общем, сам по себе DrawerLayout работает, но вот с индикатором уже не слишком дружит.

Устранению этих недостатков посвящена отдельная статья.

Но как же планшеты?

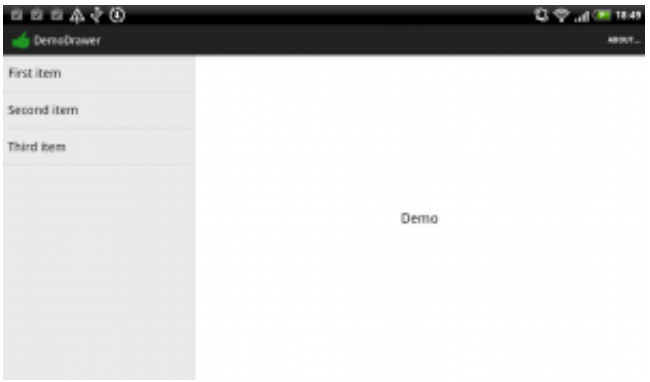
Действительно — на планшетах, особенно 10", может быть нецелесообразным скрытие бокового меню в drawer-е, ведь оно и так отлично поместится на экран.

Скажу без лишних вводных слов: код выше такой сценарий **уже** учитывает. Вам, думаю, бросились в глаза множественные проверки на предмет существования связанных с DrawerLayout объектов:

```
if (mDrawer != null)
```

```
if (mDrawerToggle != null)
```

Так вот: они предназначены не для каких-то мистических случаев, когда Android внезапно «забудет» существующий объект. Они нужны именно для того, чтоб мы могли использовать любой layout-файл, в том числе — не содержащий DrawerLayout-а в раскладке.



Старенький планшет HTC Flyer, Android 2.3, всё работает как надо

Да, именно так: DrawerLayout хорош тем, что в принципе не навязывается нам. По аналогии с Unobtrusive Javascript можно было бы сказать: «unobtrusive DrawerLayout» — ведь вся внутренняя кухня качественно скрыта от глаз, и замена одного layout-а на другой не создаёт каких-либо проблем сама по себе.

Просто помещаем в layout-large или layout-xlarge «плоскую» версию раскладки:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
     xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal"
    android:weightSum="10" android:baselineAligned="false">

    <ListView
         android:id="@+id/list_menu"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="3"
        android:background="@drawable/menu_background"
        android:entries="@array/menu_items"
        android:textColor="@android:color/white"
        tools:listitem="@android:layout/simple_list_item_1" >
    </ListView>

    <FrameLayout
         android:id="@android:id/content"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="7" >

        <TextView
              android:layout_width="wrap_content"
            android:layout_height="wrap_content"
```

```
            android:layout_gravity="center"
            android:text="@string/demo"
            android:textAppearance="?android:attr/textAppearanceLarge" />
    </FrameLayout>

</LinearLayout>
```

Далее система всё сделает сама — мы необходимые проверки уже заранее добавили. Просто запустите на планшете и убедитесь.

Где скачать демо-проект?

Демо-проект для этой статьи лежит тут: **DemoDrawer.zip**

Важно уточнить, что в архиве лежит код сразу для нескольких статей (этой, про DrawerLayout и ActionBarSherlock, и про две панельки). Потому не удивляйтесь, заметив несоответствие или требование подключить дополнительные библиотеки. Всё так и должно быть.

Для совместимости проект собирался в Eclipse: хотя Google и намекает, что будущее живёт в Android Studio, но всё же официально полноценный инструмент разработки до сих пор остаётся прежним. В любом случае, импорт проекта не вызовет проблем.

На этой оптимистичной ноте я и завершаю статью, и без того получившуюся длинноватой. Вопросы и ругань традиционно ожидаются в комментариях, а пока пожелаю всем успешной разработки современно выглядящих (то есть — использующих DrawerLayout) приложений.

Да, и ещё: после некоторых раздумий тег «КМБ» я убираю. Каюсь — для новичков такие темы сложноваты, не подумал. Но и под этот тег новые статьи тоже скоро будут.

Запись опубликована 10.11.2013 [http://www.dimasokol.ru/drawerlayout-panel-from-google/] в рубрике Разработка с метками android, UI.

Комментарии: ВКонтакте (1) Facebook (0) Обычные (8)

1 комментарий

Ваш комментарий...

Отправить

Прикрепить

Дмитрий Ильин

Отличная статья, премного благодарен.

12 июл в 0:07 | Комментировать

Мне нравится