ActionBar на Android 2.1+ с помощью SupportLibrary tutorial

[Разработка под Android](https://habrahabr.ru/hub/android_dev/)\*

Здравствуй, Хабрахабр!  
  
Недавно я [писал](http://habrahabr.ru/post/187976) о том, что Google добавил поддержку ActionBar в свою Support Library. Думаю, стоит рассказать, как же им пользоваться. Под катом — инструкция по правильному импорту библиотеки в свой проект и основные моменты использования SupportActionBar.

Импорт библиотеки

Чтобы наш ActionBar был виден на старых устройствах, нужно наследовать тему приложения от **Theme.AppCompat**. Сама собой она ниоткуда не возьмётся, поэтому нужно создать проект библиотеки (оригинальная инструкция на английском для Eclipse и Android Studio [здесь](http://developer.android.com/intl/ru/tools/support-library/setup.html#libs-with-res)(смотреть пункт **Adding libraries with resources**)):  
1. Сначала нужно убедиться, что у нас закачана последняя версия Support Library. Для этого открываем SDK Manager и листаем в самый низ, до папки Extras. В ней есть пункт Android Support Library – он то нам и нужен. Обновляем его до последней версии (сейчас – rev. 18), если не сделали этого раньше.  
2. Обновили? Молодцы. Теперь нажимаем **File > New >** В папке Android выбираем **Android Project from existing code**.  
3. Нажимаем кнопку **Browse…** и ищем нужную нам папку. Путь до неё примерно такой: <ваш SDK>/extras/android/support/v7/appcompat/. Выделяем появившийся пункт в списке и нажимаем Finish.  
4. Теперь у вас в Project Explorer должен появиться проект **android-support-v7-appcompat**. Открываем его, в папке **libs/** на обеих .jar – файлах кликаем правой кнопкой и нажимаем **Build Path > Add to Build Path**.  
5. Щелчок правой кнопкой мыши по проекту, выбираем Build Path > Configure Build Path.  
6. На странице **Build Path** во вкладке **Order and Export** отмечаем два только что добавленных .jar – файла и снимаем отметку с**Android Dependencies**.  
7. Нажимаем ОК для сохранения изменений. Всё – библиотека готова к использованию!

Создание приложения

Теперь создадим проект своего приложения, которое мы и будем делать. Имя – SupportActionBarDemo, пакет – com.habrahabr.sabd, минимальный API level 7 (таковы требования библиотеки). Создаём Activity, имя – MainActivity, layout – main.  
Теперь кликаем правой кнопкой мыши по этому проекту и нажимаем **Properties**. На странице **Android** под заголовком **Library**нажимаем **Add** и выбираем в появившемся окне **android-support-v7-appcompat**, затем – **ОК** и ещё раз **ОК**, чтобы сохранить изменения. Теперь библиотека добавлена в проект!

Простой пример

Прежде всего идём в **res/values/styles.xml**, **res/values-v11/styles.xml**, **res/values-v14/styles.xml** (спасибо [DeusModus](https://habrahabr.ru/users/deusmodus/))и пишем

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

Теперь ActionBar будет виден на любых версиях андроида.  
  
Открываем **res/values/strings.xml** и добавляем строки:

<string name="action\_item\_1">Item 1</string>

<string name="action\_item\_2">Item 2</string>

<string name="action\_item\_3">Item 3</string>

Открываем **res/menus/main.xml** и пишем там:

<menu xmlns:android="http://schemas.android.com/apk/res/android"

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

<item

android:id="@+id/action\_settings"

android:title="@string/action\_settings"

android:icon="@android:drawable/ic\_menu\_add"

sabd:showAsAction="always" />

<item

android:id="@+id/action\_item\_1"

android:title="@string/action\_item\_1"

android:icon="@android:drawable/ic\_menu\_delete"

sabd:showAsAction="ifRoom" />

<item

android:id="@+id/action\_item\_2"

android:title="@string/action\_item\_2"

android:icon="@android:drawable/ic\_menu\_add"

sabd:showAsAction="ifRoom|withText" />

<item

android:id="@+id/action\_item\_3"

android:title="@string/action\_item\_3"

android:icon="@android:drawable/ic\_menu\_add"

sabd:showAsAction="never" />

</menu>

Обратите внимание на атрибуты **xmlns:sabd=«**[**schemas.android.com/apk/res-auto**](http://schemas.android.com/apk/res-auto)**»** и **sabd:showAsAction** – без них ActionBar будет неправильно работать. Иконки случайные, значения не имеют. Я всё-таки напишу, что означает атрибут **sabd:showAsAction**:  
  
always — элемент всегда будет виден, если места не хватает, заголовок будет показан не полностью  
ifRoom — элемент будет виден, только если для него есть место  
never — элемент никогда не будет виден, для его показа нужно нажать кнопку Меню на устройстве или кнопку Overflow на ActionBar при отсутствии первой  
  
withText -элемент будет показываться только с его заголовком  
collapseActionView — элемент может сворачиваться в кнопку или разворачиваться на всю ширину Actionbar по нажатию, далее я приведу его пример  
  
Теперь приступаем непосредственно к написанию кода. Первым делом в коде Activity нужно заменить **extends Activity** на **extends ActionBarActivity** и добавить её в импорт. В последних версиях ADT при создании Activity автоматически создаётся метод**onCreateOptionsMenu(Menu menu)**, в котором мы и создаём меню:

**import** android.os.Bundle;

**import** android.support.v7.app.ActionBarActivity;

**import** android.view.Menu;

**public** **class** **MainActivity** **extends** **ActionBarActivity** {

**@Override**

**public** **void** **onCreate**(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.main);

}

**@Override**

**public** **boolean** **onCreateOptionsMenu**(Menu menu) {

getMenuInflater().inflate(R.menu.main, menu);

**return** **true**;

}

**@Override**

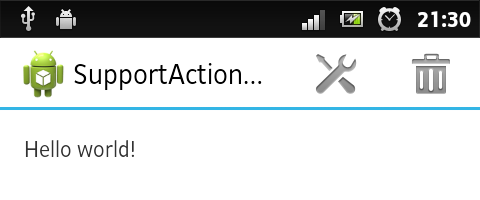
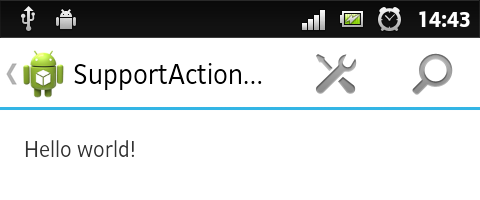
**public** **boolean** **onOptionsItemSelected**(MenuItem item) {

Log.d("MENU", "Cliced MenuItem is " + item.getTitle());

**return** **super**.onOptionsItemSelected(item);

}

}

Запускаем приложение и видим наш ActionBar:  
  
  
  
Item 2 не отобразился, так как для него не хватило места. Item 3 ни при каких условиях не будет виден, потому что мы выставили атрибут **sabd:showAsAction=«never»**. Добраться до них можно с помощью кнопки «Меню» на устройстве.  
Обрабатывать нажатия на элементы меню можно там же, где и раньше — в методе **onOptionsItemSelected(MenuItem item)**. Обрабатывать нажатия на иконку приложения можно в этом же методе, она имеет ID android.R.id.home. Чтобы добавить на ActionBar кнопку «Вверх» («Up Button»), нужно использовать метод [ActionBar.setDisplayHomeAsUpEnabled(boolean showHomeAsUp)](http://developer.android.com/reference/android/support/v7/app/ActionBar.html" \l "setDisplayHomeAsUpEnabled(boolean)):  
  


Поиск

Иногда нужно сделать поиск, например, как в Google Play. На помощь приходит ActionView. Открываем **res/menu/main.xml** и удаляем 3 последних элемента — они нам не нужны, а место занимать будут. Вместо них добавляем один новый:

<item android:id="@+id/action\_search"

android:title="@string/action\_search"

android:icon="@android:drawable/ic\_menu\_search"

sabd:showAsAction="always|collapseActionView"

sabd:actionViewClass="android.support.v7.widget.SearchView" />

атрибут **sabd:actionViewClass=«android.support.v7.widget.SearchView»** обозначает, какой View будет использован вместо обычного. Текст **collapseActionView** в атрибуте **sabd:showAsAction** говорит о том, что ActionView может быть сворачиваться в кнопку или разворачиваться на всю ширину по нажатию. Чтобы использовать его в Activity, изменим код **MainActivity**:

**import** android.os.Bundle;

**import** android.support.v4.view.MenuItemCompat;

**import** android.support.v7.app.ActionBarActivity;

**import** android.support.v7.widget.SearchView;

**import** android.support.v7.widget.SearchView.OnQueryTextListener;

**import** android.view.Menu;

**import** android.view.MenuItem;

**public** **class** **MainActivity** **extends** **ActionBarActivity** **implements** **OnQueryTextListener** {

**@Override**

**protected** **void** **onCreate**(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.main);

}

**@Override**

**public** **boolean** **onCreateOptionsMenu**(Menu menu) {

getMenuInflater().inflate(R.menu.main, menu);

MenuItem searchItem = menu.findItem(R.id.action\_search);

SearchView searchView = (SearchView) MenuItemCompat.getActionView(searchItem);

searchView.setQueryHint("Поиск");

searchView.setOnQueryTextListener(**this**);

**return** **true**;

}

**@Override**

**public** **boolean** **onOptionsItemSelected**(MenuItem item) {

Log.d("MENU", "Cliced MenuItem is " + item.getTitle());

**return** **super**.onOptionsItemSelected(item);

}

**public** **boolean** **onQueryTextChange**(String text\_new) {

Log.d("QUERY", "New text is " + text\_new);

**return** **true**;

}

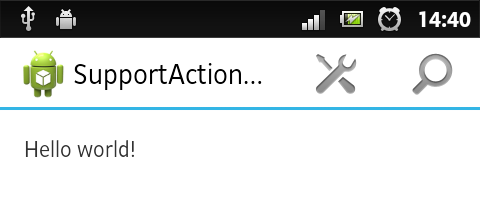
**public** **boolean** **onQueryTextSubmit**(String text) {

Log.d("QUERY", "Search text is " + text);

**return** **true**;

}

}

Запускаем приложение и видим:  
  
  
  
Нажимаем на кнопку поиска:  
  
  
  
Кстати, клавиатура появляется автоматически. Выйти из режима поиска можно, нажав на кнопку Вверх в левой части ActionBar или нажав на аппаратную кнопку назад.  
В методе **onQueryTextChange()** мы получаем текст из поля ввода, когда пользователь набирает очередную букву. В методе**onQueryTextSubmit(String text)** нам даётся текст, который пользователь ищет. У SearchView есть такие полезные методы:  
[setQuery(CharSequence query, boolean submit)](http://developer.android.com/intl/ru/reference/android/support/v7/widget/SearchView.html#setQuery(java.lang.CharSequence, boolean)) — изменяет текст в поле ввода на тот, который ему передают, опционально делает его конечным (начинает поиск)  
[getQuery()](http://developer.android.com/intl/ru/reference/android/support/v7/widget/SearchView.html#getQuery()) — возвращает текст, который сейчас есть в поле ввода  
[setQueryHint(CharSequence hint)](http://developer.android.com/intl/ru/reference/android/support/v7/widget/SearchView.html#setQueryHint(java.lang.CharSequence)) — изменяет подсказку на ту, которую ему передают  
[getQueryHint()](http://developer.android.com/intl/ru/reference/android/support/v7/widget/SearchView.html#getQueryHint()) — возвращает подсказку  
[setSuggestionsAdapter(CursorAdapter adapter)](http://developer.android.com/reference/android/support/v7/widget/SearchView.html#setSuggestionsAdapter(android.support.v4.widget.CursorAdapter)) — добавляет выпадающий список, как у AutoCompleteTextView  
[getSuggestionsAdapter()](http://developer.android.com/reference/android/support/v7/widget/SearchView.html#getSuggestionsAdapter()) — возвращает адаптер этого списка  
[setOnCloseListener(SearchView.OnCloseListener listener)](http://developer.android.com/reference/android/support/v7/widget/SearchView.html#setOnCloseListener(android.support.v7.widget.SearchView.OnCloseListener)) — ставит на него обработчик закрытия  
[setOnSuggestionListener(SearchView.OnSuggestionListener listener)](http://developer.android.com/intl/ru/reference/android/support/v7/widget/SearchView.html#setOnSuggestionListener(android.support.v7.widget.SearchView.OnSuggestionListener)) — ставит на него обработчик нажатия на элемент выпадающего списка

Заключение

А ребята из гугла молодцы, не забывают о поддержке старых версий своей ОС. Например, фрагменты, ViewPager и NavigationDrawer с API v4 доступны, Actionbar – с API v7. Хотя мне кажется, что всё это было сделано ради Android 2.3, а более ранние платформы – так, «за компанию».  
  
Буду рад, если эта статья кому-нибудь поможет :-)  
  
[Часть 2](http://habrahabr.ru/post/189678/) — Навигация с помощью вкладок и выпадающего списка  
[Часть 3](http://habrahabr.ru/post/189676/) — Дополнительные функции

Начало формы

**А какие версии Android поддерживаете вы?**

|  |  |
| --- | --- |
|  | Android 1.5+ |
|  | Android 1.6+ |
|  | Android 2.1+ |
|  | Android 2.2+ |
|  | Android 2.3+ |
|  | Android 3.0+ |
|  | Android 3.1+ |
|  | Android 3.2+ |
|  | Android 4.0+ |
|  | Android 4.1+ |
|  | Android 4.2+ |
|  | Android 4.3+ |

Конец формы

ActionBar на Android 2.1+ с помощью Support Library. Часть 2 — Навигация tutorial

[Разработка под Android](https://habrahabr.ru/hub/android_dev/)\*

Привет, Хабр!  
  
В [предыдущей статье](http://habrahabr.ru/post/189680/) я рассказал о добавлении Support Library в ваш проект и привёл простой пример SupportActionBar. Но очень часто ActionBar используется не только как замена меню, но и как способ навигации по приложению. Под катом написано, как её реализовать.

Способы навигации

У ActionBar есть 3 способа навигации:  
[NAVIGATION\_MODE\_STANDART](http://developer.android.com/reference/android/support/v7/app/ActionBar.html#NAVIGATION_MODE_STANDARD) – по сути вообще не навигация, просто ActionBar с элементами;  
[NAVIGATION\_MODE\_LIST](http://developer.android.com/reference/android/support/v7/app/ActionBar.html#NAVIGATION_MODE_LIST) – вместо заголовка выпадающий список;  
[NAVIGATION\_MODE\_TABS](http://developer.android.com/reference/android/support/v7/app/ActionBar.html#NAVIGATION_MODE_TABS) – вкладки под ActionBar.

Выпадающий список

Давайте не будем ничего создавать, а возьмём проект из [предыдущей статьи](http://habrahabr.ru/post/189680/). Создадим новый класс – **ScreenFragment**, он будет аналогом разных экранов приложения:

**import** android.os.Bundle;

**import** android.support.v4.app.Fragment;

**import** android.view.LayoutInflater;

**import** android.view.View;

**import** android.view.ViewGroup;

**import** android.widget.TextView;

**public** **class** **ScreenFragment** **extends** **Fragment** {

**@Override**

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

TextView tv = **new** TextView(getActivity());

tv.setText("Screen " + getArguments().getInt(MainActivity.key\_screen\_number));

tv.setTextSize(30);

**return** tv;

}

}

Я не стал создавать отдельный xml-файл разметки, он здесь не особо нужен. Мы берём из аргументов номер экрана и вставляем его в программно созданный TextView, который потом показываем.  
Изменим код метода **onCreate()** и добавим ещё один в **MainActivity**:

**public** **static** **final** String key\_screen\_number = "key\_screen\_number";

ActionBar ab;

FragmentTransaction ft;

ScreenFragment screen\_fragment;

**@Override**

**public** **void** **onCreate**(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

ab = getSupportActionBar();

ab.setNavigationMode(ActionBar.NAVIGATION\_MODE\_LIST);

String[] screens = **new** String[] {"Screen 1", "Screen 2", "Screen 3"};

ArrayAdapter<String> sp\_adapter = **new** ArrayAdapter<String>(**this**, android.R.layout.simple\_spinner\_item, screens);

sp\_adapter.setDropDownViewResource(android.R.layout.simple\_spinner\_dropdown\_item);

ab.setListNavigationCallbacks(sp\_adapter, **this**);

selected\_list\_item\_position = -1;

ab.setSelectedNavigationItem(0);

}

**public** **boolean** **onNavigationItemSelected**(**int** position, **long** id) {

ft = getSupportFragmentManager().beginTransaction();

screen\_fragment = **new** ScreenFragment();

Bundle args = **new** Bundle();

args.putInt(key\_screen\_number, position + 1);

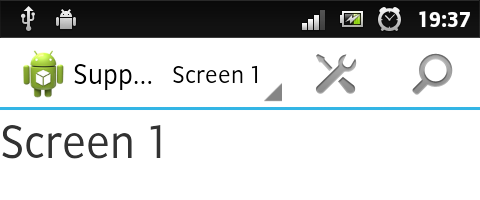
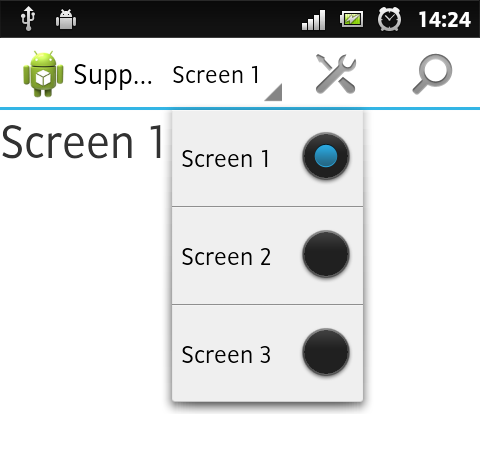
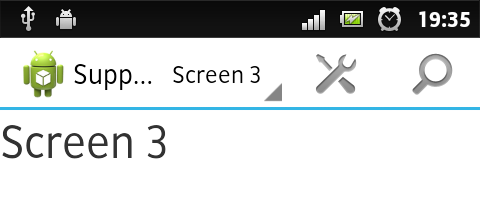
screen\_fragment.setArguments(args);

ft.replace(android.support.v7.appcompat.R.id.action\_bar\_activity\_content, screen\_fragment);

ft.commit();

**return** **true**;

}

В onCreate мы говорим ActionBar, что будем использовать метод навигации – список, и подготавливаем адаптер для него, а также присваиваем обработчик событий. У него всего один метод — **onNavigationItemSelected(int position, long id)**. Он вызывается, когда пользователь выбирает какой–нибудь элемент выпадающего списка. Здесь мы создаём новый **ScreenFragment** и даём ему номер экрана, чтобы он мог его показать. Затем начинаем **FragmentTransaction** и добавляем этот фрагмент в View с id=android.support.v7.appcompat.R.id.action\_bar\_activity\_content. Это FrameLayout, куда добавляется наш layout из setContentView(). Запускаем приложение и выбираем различные экраны:  
  
  
  
  
  
  
  
В качестве разметки для элементов выпадающего списка я использую системный layout, но он выглядит не очень красиво. Поэтому лучше использовать свой. За его добавление отвечает метод **Adapter.setDropDownViewResource().**

Вкладки

Чтобы изменить способ навигации на вкладки, подправим **MainActivity**:

**public** **void** **onCreate**(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

ab = getSupportActionBar();

ab.setNavigationMode(ActionBar.NAVIGATION\_MODE\_TABS);

Tab tab = ab.newTab();

tab.setText("Screen 1");

tab.setTabListener(**this**);

ab.addTab(tab, 0, **true**);

tab = ab.newTab();

tab.setText("Screen 2");

tab.setTabListener(**this**);

ab.addTab(tab, 1, **false**);

tab = ab.newTab();

tab.setText("Screen 3");

tab.setTabListener(**this**);

ab.addTab(tab, 2, **false**);

}

Также нужно сделать **MainActivity... implements... TabListener**. Это обработчик нажатий на вкладки. У него есть целых 3 метода:  
[onTabUnselected(Tab tab, FragmentTransaction ft)](http://developer.android.com/reference/android/support/v7/app/ActionBar.TabListener.html#onTabUnselected(android.support.v7.app.ActionBar.Tab, android.support.v4.app.FragmentTransaction)) — вызывается, когда текущая вкладка закрывается;  
[onTabSelected(Tab tab, FragmentTransaction ft)](http://developer.android.com/reference/android/support/v7/app/ActionBar.TabListener.html#onTabSelected(android.support.v7.app.ActionBar.Tab, android.support.v4.app.FragmentTransaction)) — вызывается, когда открывается новая вкладка (срабатывает сразу после предыдущего);  
[onTabReselected(Tab tab, FragmentTransaction ft)](http://developer.android.com/reference/android/support/v7/app/ActionBar.TabListener.html#onTabReselected(android.support.v7.app.ActionBar.Tab, android.support.v4.app.FragmentTransaction)) — когда пользователь нажимает на уже открытую вкладку:

**public** **void** **onTabUnselected**(Tab tab, FragmentTransaction ft) {

}

**public** **void** **onTabSelected**(Tab tab, FragmentTransaction ft) {

screen\_fragment = **new** ScreenFragment();

Bundle args = **new** Bundle();

args.putInt(key\_screen\_number, tab.getPosition() + 1);

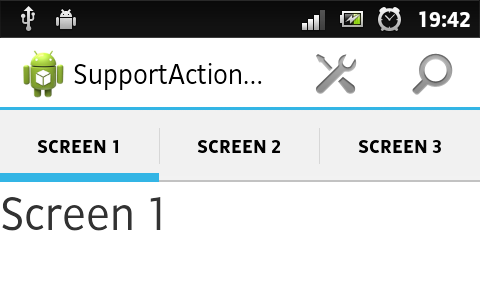
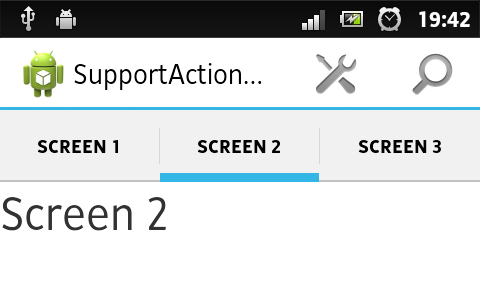
screen\_fragment.setArguments(args);

ft.replace(android.support.v7.appcompat.R.id.action\_bar\_activity\_content, screen\_fragment);

}

**public** **void** **onTabReselected**(Tab tab, FragmentTransaction ft) {

}

Здесь нам уже не нужно создавать FragmentTransaction, она даётся нам изначально (предполагается, что мы будем работать с фрагментами). Но для этой FragmentTransaction нельзя вызывать методы addToBackStack() и commit(). Также у нас есть нажатая вкладка, из которой мы можем вытащить всё, что нужно — текст, иконку, позицию и т.д.   
Вкладкам можно присваивать свой View, если системный вас не устраивает — [setCustomView(int layoutResId)](http://developer.android.com/reference/android/support/v7/app/ActionBar.Tab.html" \l "setCustomView(int))  
Запускаем приложение, щёлкаем по вкладкам:  
  
  
  
  
  
Кстати, если вкладок очень много, то их заголовки можно скроллить по горизонтали (как в Google Play), но ниже заголовков свайп не работает.

Дополнение к «Выпадающий список»

Скорее всего, при нажатии на уже выбранный элемент навигации, на экране ничего не нужно менять. Ну, со вкладками всё понятно — не трогать метод **onTabReselected()** и всё. А как же быть со списком? Всё очень просто: добавляем в **MainActivity** переменную

**private** **int** selected\_list\_item\_position;

И изменяем код **onNavigationItemSelected(int position, long id)**:

**public** **boolean** **onNavigationItemSelected**(**int** position, **long** id) {

**if** (position != selected\_list\_item\_position) {

ft = getSupportFragmentManager().beginTransaction();

screen\_fragment = **new** ScreenFragment();

Bundle args = **new** Bundle();

args.putInt(key\_screen\_number, position + 1);

screen\_fragment.setArguments(args);

ft.replace(android.support.v7.appcompat.R.id.action\_bar\_activity\_content, screen\_fragment);

ft.commit();

selected\_list\_item\_position = position;

**return** **true**;

}

**return** **false**;

}

Теперь новый экран будет открываться только при выборе не открытого элемента навигации.

Меню

На разных вкладках обычно размещается разный контент, и меню для него должно быть разным. Ребята из гугла сделали такую возможность. Далее я буду показывать всё на примере вкладок. Добавим в **ScreenFragment** следующий код:

**public** **static** **final** String key\_menu\_resource = "key\_menu\_resource";

**@Override**

**public** **void** **onCreateOptionsMenu**(Menu menu, MenuInflater inflater) {

**super**.onCreateOptionsMenu(menu, inflater);

inflater.inflate(getArguments().getInt(key\_menu\_resource), menu);

}

**@Override**

**public** **boolean** **onOptionsItemSelected**(MenuItem item) {

**super**.onOptionsItemSelected(item);

Log.d("MENU", "Cliced MenuItem is " + item.getTitle() + " (from ScreenFragment)");

**return** **true**;

}

Создадим в папке **res/menu/** три файла:  
  
**screen\_1.xml**:

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

<item

android:id="@+id/item1"

android:title="Item 1"

android:icon="@android:drawable/ic\_menu\_add"/>

<item

android:id="@+id/settings"

android:title="Settings"

android:icon="@android:drawable/ic\_menu\_edit"/>

</menu>

**screen\_2.xml**:

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

<item

android:id="@+id/item2"

android:title="Item 2"

android:icon="@android:drawable/ic\_menu\_camera"/>

<item

android:id="@+id/settings"

android:title="Settings"

android:icon="@android:drawable/ic\_menu\_edit"/>

</menu>

**screen\_3.xml**:

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

<item

android:id="@+id/item3"

android:title="Item 3"

android:icon="@android:drawable/ic\_menu\_call" />

</menu>

Изменим **onTabSelected()**:

**private** **int**[] menu\_resources = **new** **int**[] {R.menu.screen\_1, R.menu.screen\_2, R.menu.screen\_3};

**public** **void** **onTabSelected**(Tab tab, FragmentTransaction ft) {

screen\_fragment = **new** ScreenFragment();

Bundle args = **new** Bundle();

args.putInt(key\_screen\_number, tab.getPosition() + 1);

args.putInt(ScreenFragment.key\_menu\_resource, menu\_resources[tab.getPosition()]);

screen\_fragment.setArguments(args);

screen\_fragment.setHasOptionsMenu(**true**);

ft.replace(android.support.v7.appcompat.R.id.action\_bar\_activity\_content, screen\_fragment);

}

Теперь нужно удалить (ну или лучше закомментить) метод **onCreateOptionsMenu** — он нам сейчас будет только мешать. И**onOptionsItemSelected()** в MainActivity тоже подправим:

**@Override**

**public** **boolean** **onOptionsItemSelected**(MenuItem item) {

**if** (item.getItemId() != R.id.settings) {

**return** **false**;

} **else** {

Log.d("MENU", "Cliced MenuItem is " + item.getTitle() + " (from MainActivity)");

**return** **true**;

}

}

Сейчас поясню, что я здесь накодил. Дело в том, что во фрагменте тоже можно создавать меню. Чтобы оно было видно, нужно вызывать метод Fragment.setHasOptionsMenu(true). Если мы создаём меню не в Activity, а в фрагменте, то метод**onOptionsItemSelected()** вызывается сначала в **MainActivity**, а лишь затем в **ScreenFragment**, если в Activity возвращается *false*. Здесь вместо **if** должно быть **switch/case**, в конце каждого **case** — **return true;** Это значит, что мы уже обработали нажатие и не нужно вызывать onOptionsItemSelected во фрагменте. Например, на каждой вкладке есть пункт меню «Настройки». Чтобы не набирать код в каждом фрагменте, при нажатии на этот пункт возвращаем *true*. Тогда onOptionsItemSelected() вызывается только в Activity, где мы можем открыть новую SettingsActivity, например. Если запустить программу и на разных вкладках нажимать кнопку «Меню» на устройстве, то будут показаны разные элементы.  
При нажатии на пункты меню в логах будет не только их имя, но и в каком классе были обработаны нажатия. А можно вообще создать отдельный xml-файл в папке **res/menu/** с этим самым элементом Settings, а в **MainActivity** в методе**onCreateOptionsMenu()** создавать меню из этого файла. Тогда 2 меню как бы объединятся, и будут видны пункты обоих.

Сохранение состояния

Часто бывает, что при переключении между вкладками состояние контента на них должно сохраняться. Для этого у фрагментов есть специальный метод — [setRetainInstance(boolean retain)](http://developer.android.com/reference/android/support/v4/app/Fragment.html" \l "setRetainInstance(boolean)). Если ему передать *true* в параметре, то фрагмент не будет создаваться заново. Чтобы проверить это, перепишем метод **onTabSelected()** в **MainActivity**:

**private** **int**[] menu\_resources = **new** **int**[] {R.menu.screen\_1, R.menu.screen\_2, R.menu.screen\_3};

**private** ScreenFragment[] screens = **new** ScreenFragment[] {**new** ScreenFragment(), **new** ScreenFragment(), **new** ScreenFragment()};

**public** **void** **onTabSelected**(Tab tab, FragmentTransaction ft) {

screen\_fragment = screens[tab.getPosition()];

Bundle args = **new** Bundle();

args.putInt(key\_screen\_number, tab.getPosition() + 1);

args.putInt(ScreenFragment.key\_menu\_resource, menu\_resources[tab.getPosition()]);

screen\_fragment.setArguments(args);

screen\_fragment.setHasOptionsMenu(**true**);

screen\_fragment.setRetainInstance(**true**);

ft.replace(android.support.v7.appcompat.R.id.action\_bar\_activity\_content, screen\_fragment);

}

Послесловие

Ну вот в общем-то и всё, что я хотел сказать. Статья получилась большая, но, надеюсь, полезная)

# ActionBar на Android 2.1+ с помощью Support Library. Часть 3 — Полезные функции tutorial

[Разработка под Android](https://habrahabr.ru/hub/android_dev/)\*

Всем доброго времени суток!  
  
Мда, давно не писал я статей на Хабр. Что ж, буду это дело исправлять.  
Вообще, ActionBar в Android – довольно сложная штука. В [первой](http://habrahabr.ru/post/189680/) и [второй](http://habrahabr.ru/post/189678/) частях я рассмотрел только основные его функции – меню и навигация. Но есть ведь и множество дополнительных: Split ActionBar, кастомный View для элемента меню и ActionProvider. О них и пойдёт речь. Кроме того, бонус: многие знают, что кнопка Up служит для перехода на предыдущее Activity. Но если на экран можно попасть только с одной другой Activity, то можно не кодить переход вверх. Заинтригованы? Прошу под кат.

#### Split ActionBar

  
  
Split ActionBar – это полоска внизу экрана с элементами меню. Он может быть использован, если в основном ActionBar не хватает места (в середине), или он отсутствует (справа). Добавить его легко: достаточно в файле манифеста добавить для Activity или всего приложения следующие строки:

<manifest ...>

<activity …

uiOptions="splitActionBarWhenNarrow" ... > //Для API 14+

<meta-data android:name="android.support.UI\_OPTIONS" // Для Support Library API 7+

android:value="splitActionBarWhenNarrow" />

</activity>

</manifest>

Чтобы убрать верхний ActionBar (как на рисунке справа) нужно вызвать методы ActionBar [setDisplayShowHomeEnabled(false)](http://developer.android.com/reference/android/support/v7/app/ActionBar.html" \l "setDisplayShowHomeEnabled(boolean)) и[setDisplayShowTitleEnabled(false)](http://developer.android.com/reference/android/support/v7/app/ActionBar.html#setDisplayShowTitleEnabled(boolean)).

#### ActionView

По умолчанию элемент меню имеет вид картинки (иконки). Есть два способа изменить его:  
-Добавить атрибут actionLayout для тега . Он принимает на вход ссылку на layout-файл  
-Добавить атрибут actionViewClass для тега , куда записать имя класса, наследника View, для использования вместо стандартной вьюхи

<menu xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:yourapp="http://schemas.android.com/apk/res-auto" >

<item android:id="@+id/custom\_action "

android:title="@string/custom\_action "

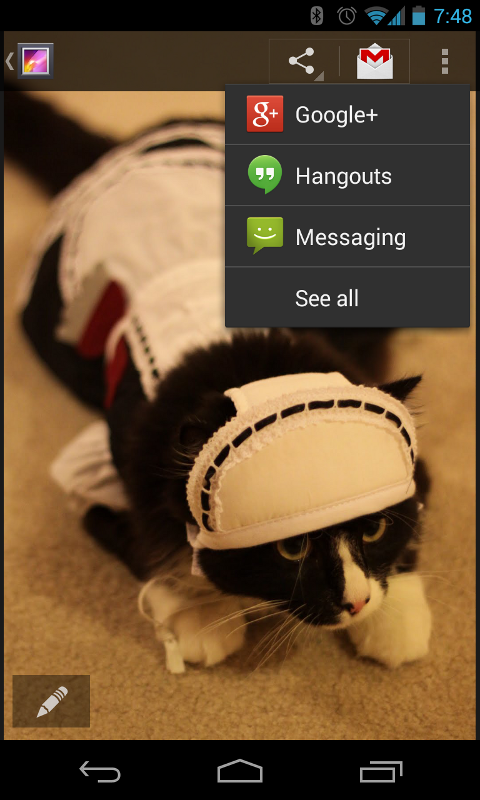
android:icon="@drawable/ic\_custom\_action "

yourapp:actionViewClass="com.you.app.CustomView" />

</menu>

В пространстве имён android в старых версиях нет этих атрибутов, поэтому нужно использовать пространство имён Support Library – для этого добавляем строчку **xmlns:yourapp=«**[**schemas.android.com/apk/res-auto**](http://schemas.android.com/apk/res-auto)**»** и используем атрибут**yourapp:actionViewClass**. Программно изменить разметку MenuItem’a можно с помощью методов MenuItemCompat[setActionView(MenuItem item, int resId)](http://developer.android.com/reference/android/support/v4/view/MenuItemCompat.html#setActionView(android.view.MenuItem, int)) и [setActionView(MenuItem item, View view)](http://developer.android.com/reference/android/support/v4/view/MenuItemCompat.html" \l "setActionView(android.view.MenuItem, android.view.View)), получить текущий View – [getActionView(MenuItem item)](http://developer.android.com/reference/android/support/v4/view/MenuItemCompat.html" \l "getActionView(android.view.MenuItem)).  
  
Один из примеров ActionView – SearchView, о нём подробнее написано [здесь](http://habrahabr.ru/post/189680/).

#### Action Provider

  
  
ActionProvider имеет много общего с ActionView. Он также заменяет кнопку на собственный layout, но может показывать выпадающий список и контролировать поведение элемента меню. Чтобы задать его для пункта меню, достаточно для тега добавить атрибут**actionProviderClass**, куда записать адрес вашего класса. Он должен наследоваться от [ActionProvider](http://developer.android.com/intl/ru/reference/android/support/v4/view/ActionProvider.html) и переопределять следующие методы:  
-[ActionProvider(Context context)](http://developer.android.com/reference/android/support/v4/view/ActionProvider.html" \l "ActionProvider(android.content.Context)) –конструктор, принимает на вход контекст, который нужно сохранить в поле класса для использования в других методах;  
-[onCreateActionView(MenuItem)](http://developer.android.com/reference/android/support/v4/view/ActionProvider.html" \l "onCreateActionView(android.view.MenuItem)) – здесь создаём вьюшку для элемента меню с помощью LayoutInflater, полученного из контекста, и возвращаем её;  
-[onPerformDefaultAction()](http://developer.android.com/reference/android/support/v4/view/ActionProvider.html" \l "onPerformDefaultAction()) – вызывается, когда пользователь нажимает на пункт в ActionBar.  
  
Если вы используете ActionProvider, то НЕ должны обрабатывать нажатия на него в **onOptionsItemSelected()**, ну или возвращать там false, потому что иначе **onPerformDefaultAction()** не будет вызван.  
  
Если ваш ActionProvider показывает подменю, то нужно переопределить его метод [hasSubMenu()](http://developer.android.com/reference/android/support/v4/view/ActionProvider.html" \l "hasSubMenu()) и возвращать true. Тогда вместо**onPerformDefaultAction()** будет вызван [onPrepareSubMenu (SubMenu subMenu)](http://developer.android.com/reference/android/support/v4/view/ActionProvider.html" \l "onPrepareSubMenu(android.view.SubMenu)). Там создаёте или изменяете подменю.

#### ShareActionProvider

В Android есть свой ActionProvider для отправки контента – [ShareActionProvider](http://developer.android.com/intl/ru/reference/android/support/v7/widget/ShareActionProvider.html). Когда создаём меню с ним, необходимо в методе onCreateOptionsMenu получить его экземпляр с помощью [MenuItemCompat.getActionProvider(MenuItem)](http://developer.android.com/reference/android/support/v4/view/MenuItemCompat.html" \l "getActionProvider(android.view.MenuItem)) и вызвать [setShareIntent()](http://developer.android.com/reference/android/support/v7/widget/ShareActionProvider.html" \l "setShareIntent(android.content.Intent)), куда передать Intent с ACTION\_SEND и прикреплённым контентом:

**private** ShareActionProvider mShareActionProvider;

**@Override**

**public** **boolean** **onCreateOptionsMenu**(Menu menu) {

getMenuInflater().inflate(R.menu.main\_activity\_actions, menu);

Intent share\_intent = **new** Intent(Intent.ACTION\_SEND);

intent.setType("image/\*");

mShareActionProvider = (ShareActionProvider) getActionProvider(menu.findItem(R.id.action\_share));

mShareActionProvider.setShareIntent(share\_intent);

**return** **super**.onCreateOptionsMenu(menu);

}

Так как контент на экране может изменяться (например, перелистывание картинок), то нужно вызывать **setShareIntent()** каждый раз, когда контент меняется.

#### Родительская Activity

Если на экран в приложении можно попасть только с ОДНОЙ другой Activity, то имеет смысл настроить автоматический переход на неё по нажатию кнопки Up в ActionBar. Сделать это очень просто: в файле манифеста для нужной Activity добавить строки:

<application ... >

<activity

android:name="com.you. app.MainActivity" ...> // Главная Activity

...

</activity>

...

<activity

android:name="com.you. app.DetailsActivity"

android:label="@string/title\_activity\_display\_message"

android:parentActivityName="com.example.myfirstapp.MainActivity" > //Для API 16+

<meta-data

android:name="android.support.PARENT\_ACTIVITY" // Для Support Library API 7+

android:value="com.you. app.MainActivity" />

</activity>

</application>

Напомню, чтобы кнопка Home работала как Up, вызовите методы SupportActionBar [setHomeButtonEnabled(true)](http://developer.android.com/reference/android/support/v7/app/ActionBar.html" \l "setHomeButtonEnabled(boolean)) и[setDisplayHomeAsUpEnabled(true)](http://developer.android.com/reference/android/support/v7/app/ActionBar.html#setDisplayHomeAsUpEnabled(boolean)).

Для API 11+

Вся информация взята [отсюда](http://developer.android.com/intl/ru/guide/topics/ui/actionbar.html) (официальный гайдлайн на английском).