

Android Fragments and the ActionBar

Jim Wilson

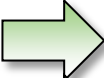


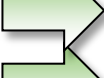

jimw@jwhh.com

@hedgehogjim



pluralsight
hardcore developer training

Outline

-  What is the ActionBar
-  Adding Fragment-specific menu options
-  ActionBar and Fragment navigation
-  List navigation
-  Tabbed navigation

Android ActionBar

➔ **Android ActionBar serves as the core of user interaction**

➔ Introduced in Android 3.0

- Available to pre-Android 3.0 via compatibility library

➔ Features

- Menu behavior, Application navigation, Much more

➔ ActionBar has close relationship with Fragments

- Fragments affect ActionBar behavior
- ActionBar utilizes Fragments for behavior

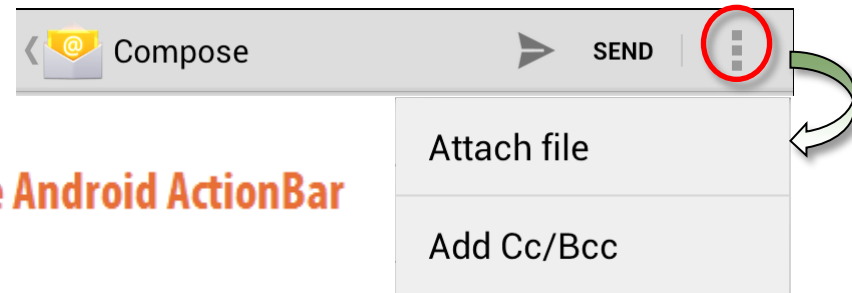
➔ Discussed in detail in Pluralsight ActionBar course

- "Improving User Interaction with the Android ActionBar"



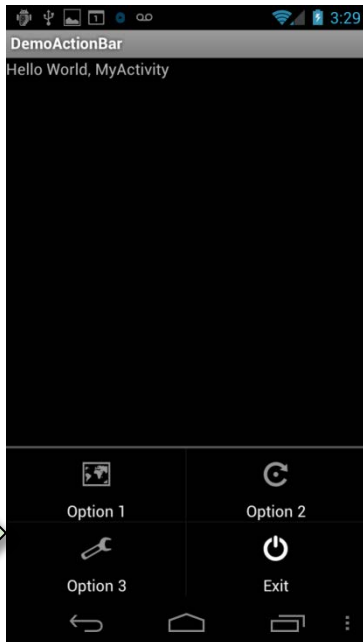
Improving User Interaction with the Android ActionBar

pluralsight
hardcore developer training



Adding the ActionBar to your app

- ➔ You must request the ActionBar
- ➔ By default, applications use the legacy menu behavior
- ➔ ActionBar available to Activities using a Theme.Holo derived theme
 - ❑ Automatic when minSdkVersion or targetSdkVersion 11 or higher in manifest



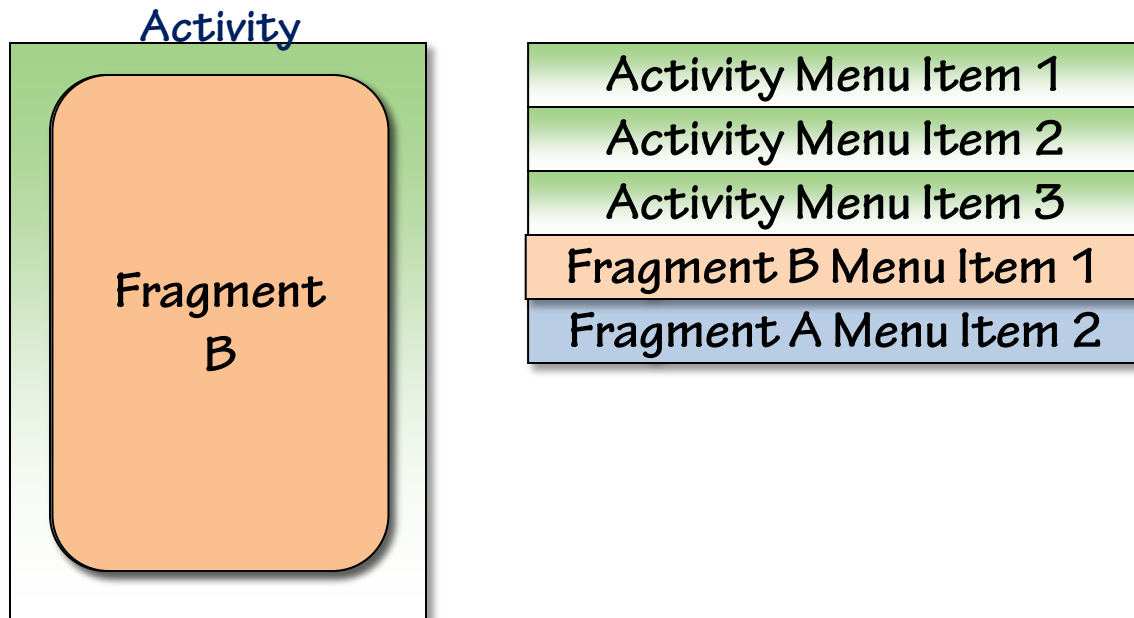
AndroidManifest.xml

```
<manifest . . . >
  <uses-sdk
    android:targetSdkVersion="14" />
  <application . . . >
    // . . .
  </application>
</manifest>
```



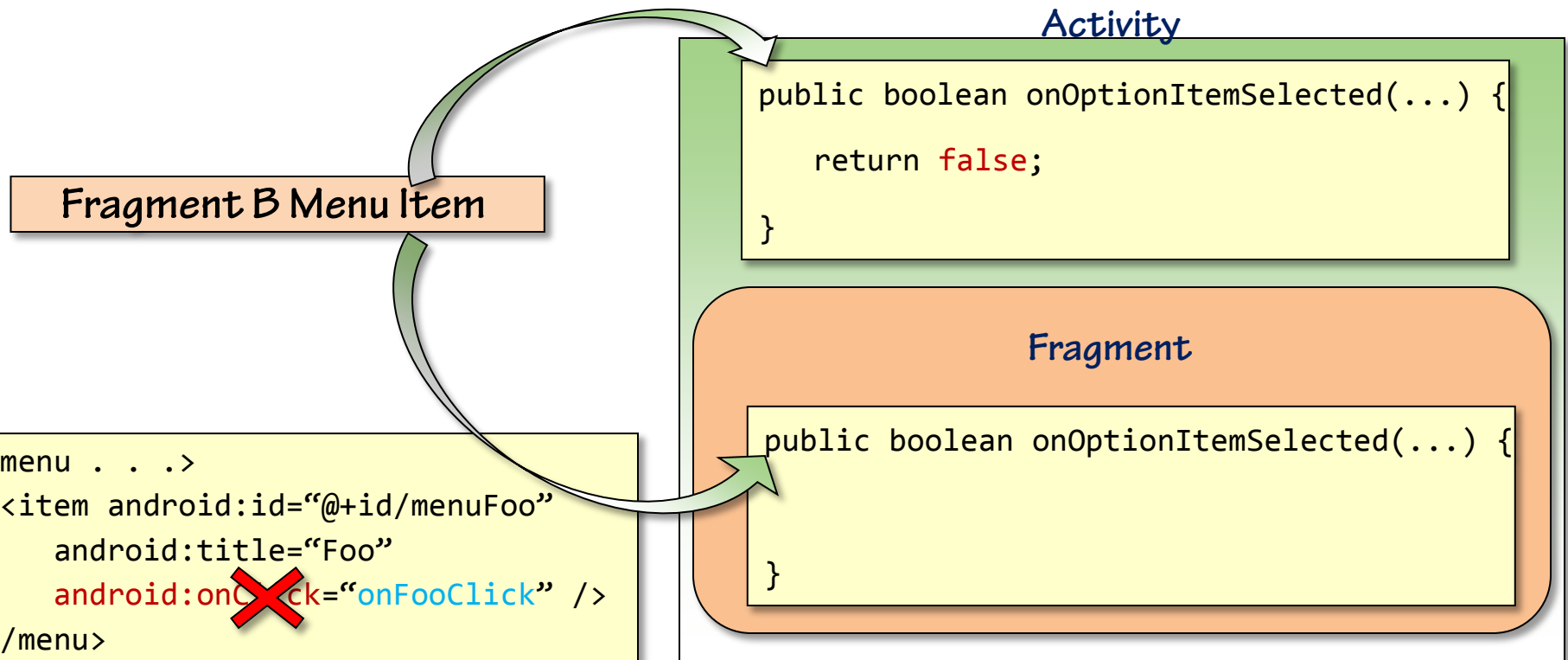
Adding Fragment-specific menu options

- ➔ **Fragments can add menu items to ActionBar**
- ➔ Allows per-Fragment specialization of the menu
 - Appear in menu after Activity-provided menu items
- ➔ Fragment must declare intention at creation time
 - Call `setHasOptionsMenu(true)` in `onCreate` callback
- ➔ Fragment creates menu options in `onCreateOptionsMenu` callback
 - In most cases simply inflate menu resource



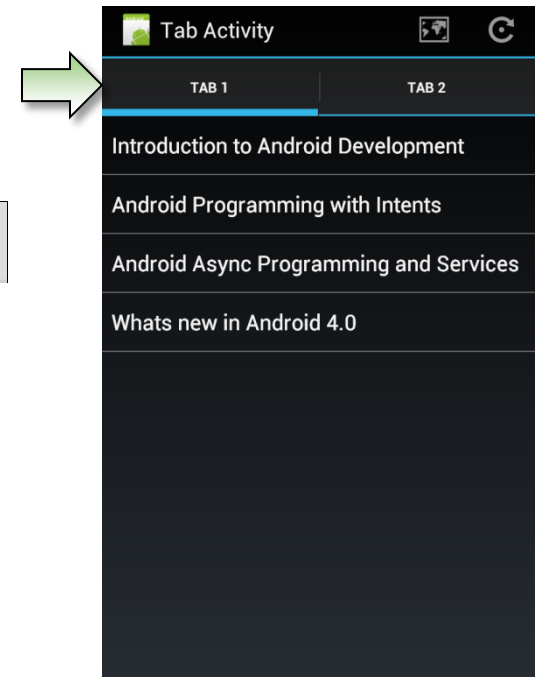
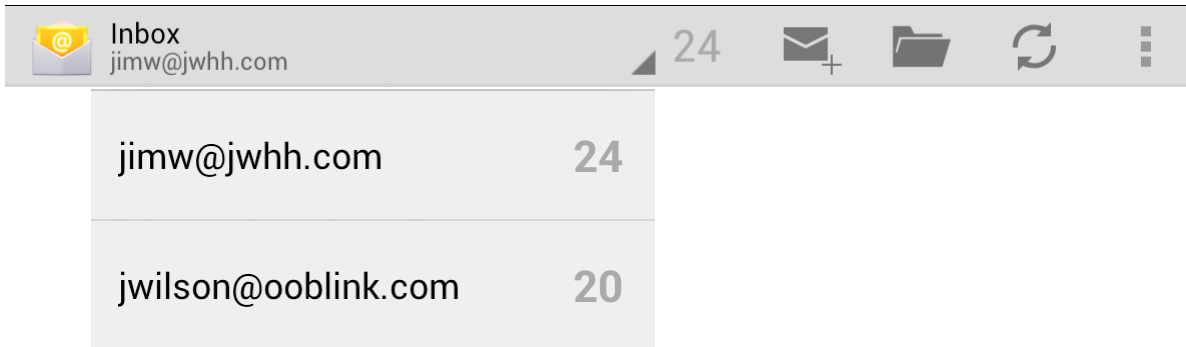
Handling Fragment-created menu options

- ➔ Fragment menu options can be handled by Activity or Fragment
- ➔ Activity's `onOptionsItemSelected` is called first
- ➔ Fragment's `onOptionsItemSelected` called only if not handled by Activity
- ➔ Avoid use of `onClick` in menu item resource definition
 - If used, the named method **must** be implemented in the Activity



Fragments at the heart of ActionBar Navigation

- ➔ **ActionBar provides rich navigation behavior**
- ➔ Provide easy ways to move between different Fragments
- ➔ List Navigation
 - ❑ Provides drop-down list of available Fragments
- ➔ Tabbed Navigation
 - ❑ Provides separate tab for each Fragment



List navigation

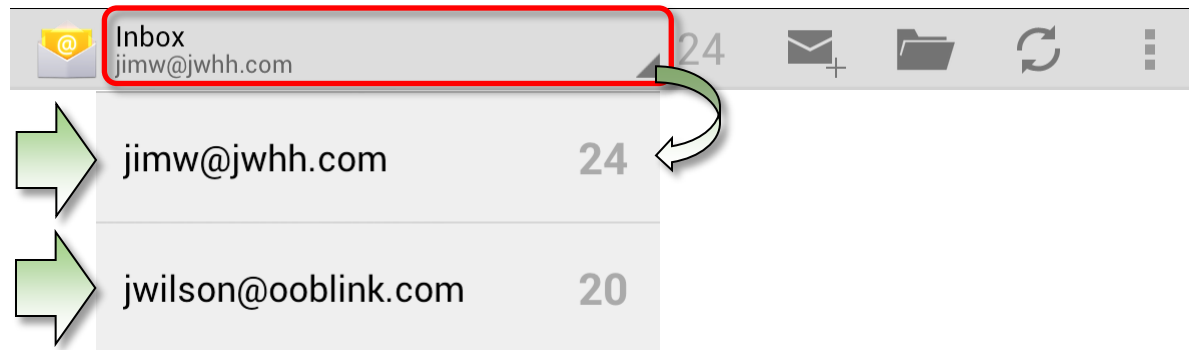
➡ **List navigation provides the user with a drop-down list of screens**

➡ List is created as a SpinnerAdapter

- ❑ Places a drop-down list on the action bar
- ❑ Allows you to populate the list from a variety of sources

➡ Each list item represents a screen

➡ Each screen implemented as a Fragment



Implementing list navigation

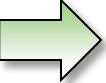
 **ActionBar.OnNavigationListener** handles changing Fragments

 A single instance handles all screens

 `onNavigationItemSelected` called each time user changes selection

- Passes index of selection

- Passes selection ID if applicable

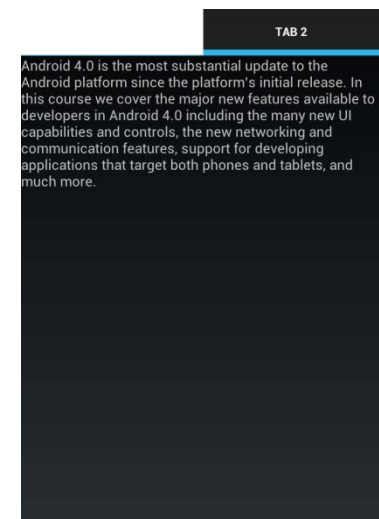
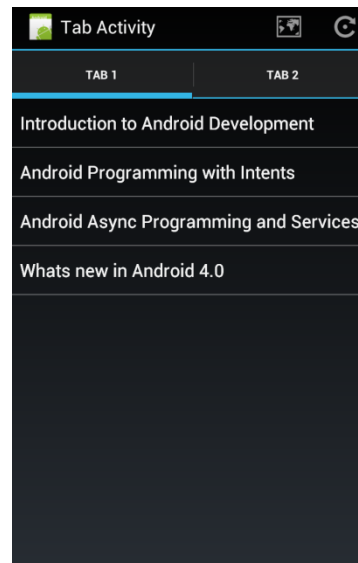
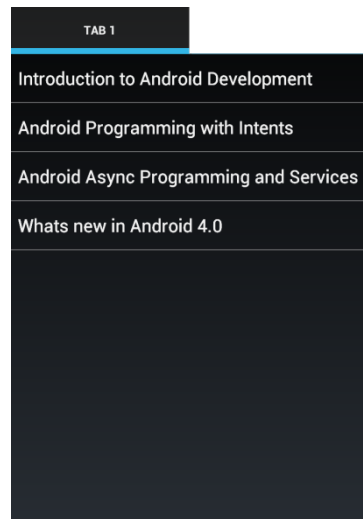
 Each selection handled by showing the appropriate Fragment

- Must create and commit `FragmentTransaction`

- Performs necessary actions within the transaction (usually a replace)

Creating tabbed navigation

- ➔ **Creating ActionBar-based tabbed navigation is a 3-phase process**
- ➔ Put the ActionBar into tabbed-navigation mode
 - ❑ Call `setNavigationMode` on the ActionBar with `NAVIGATION_MODE_TABS`
- ➔ Create a `ActionBar.Tab` instance for each tab
 - ❑ Create a Fragment for the tab window content
 - ❑ Provide `ActionBar.TabListener` interface implementations to manage tab
- ➔ Add each `ActionBar.Tab` to the ActionBar
 - ❑ Call `ActionBar.addTab` for each `ActionBar.Tab` instance



Handling tabbed navigation

- ➡ **ActionBar.TabListener** handles each tab's interaction
- ➡ Android framework manages the transactions
 - You don't need to call `beginTransaction` or `commit`
- ➡ **TabListener** implementation simply handles which **Fragment** is visible
 - **TabListener** implementation passed to each **Tab** as its created



```
FragmentManager fm = thisActivity.getFragmentManager();  
FragmentTransaction ft = fm.Manager.beginTransaction();
```

```
public void onTabXXX(ActionBar.Tab tab, FragmentTransaction ft)  
{  
    // Handle add/attach/detach Fragment for this tab  
}
```



```
ft.commit();
```

Implementing the ActionBar.TabListener

➡ The ActionBar.TabListener methods

➡ onTabSelected

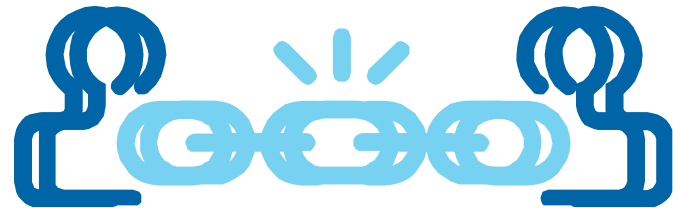
- ❑ Called when tab becomes the selected tab
- ❑ On first call must add the fragment to the transaction
- ❑ On successive calls, make the fragment visible by attaching to the transaction

➡ onTabUnselected

- ❑ Called when tab stops being the selected tab
- ❑ Must detach the fragment from the transaction

➡ onTabReselected

- ❑ Called when user taps the tab for the already selected tab
- ❑ Usually there's nothing we have to do



Summary

- ➔ **ActionBar is at the core of user interaction**
 - Provides menus, navigation, more
- ➔ **Fragments can add their own menu items to the menu**
 - Activity gets first opportunity to handle the menu item
- ➔ **List navigation provides a drop-down of Fragment selections**
 - Your code must handle all aspects of transaction and Fragment mgmt
- ➔ **Tab navigation provides tab-based Fragment selection**
 - Framework provides transaction management
 - Your code just handles Fragment details
- ➔ **See Pluralsight Course for more ActionBar details**



Improving User Interaction with the Android ActionBar

pluralsight
hardcore developer training