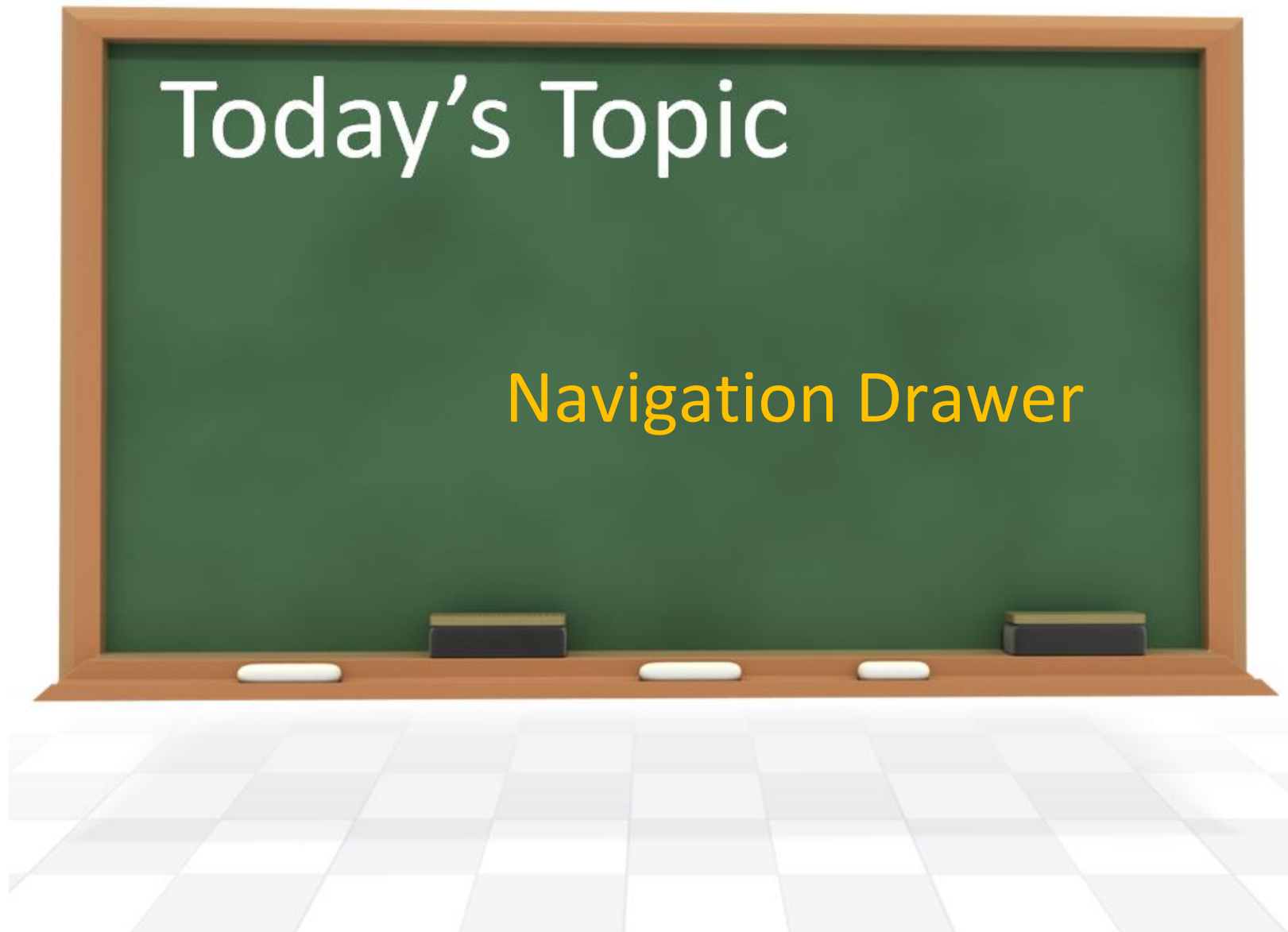


6/20/2014 (Part I)



## Discussion

## The Layout of the Activity and Fragment

### 1. Layout for the Main Activity

```
<!-- A DrawerLayout is intended to be used as the top-level content view using  
match_parent for both width and height to consume the full space available. -->
```

```
<android.support.v4.widget.DrawerLayout  
    xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:tools="http://schemas.android.com/tools"  
    android:id="@+id/drawer_layout"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    tools:context="com.example.teachweek6.app.MainActivity">
```

```
<!-- As the main content view, the view below consumes the entire  
space available using match_parent in both dimensions. -->
```

```
<FrameLayout  
    android:id="@+id/container"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent" />
```

```
<!-- android:layout_gravity="start" tells DrawerLayout to treat  
this as a sliding drawer on the left side for left-to-right  
languages and on the right side for right-to-left languages.  
If you're not building against API 17 or higher, use  
android:layout_gravity="left" instead. -->
```

```

        if you're not building against API 17 or higher, use
        android:layout_gravity="left" instead. -->
<!-- The drawer is given a fixed width in dp and extends the full height of
the container. -->
<fragment android:id="@+id/navigation_drawer"
    android:layout_width="240dp"
    android:layout_height="match_parent"
    android:layout_gravity="start"
    android:name="com.example.teachweek6.app.NavigationDrawerFragment" />

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

```

## 2. Layout for the Navigation Drawer Fragment

```

<ListView 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:choiceMode="singleChoice"
    android:divider="@android:color/transparent"
    android:dividerHeight="0dp"
    android:background="#cccc"
    tools:context="com.example.teachweek6.app.NavigationDrawerFragment" />

```

## Navigation Drawer Fragment

### **1. Set up the view inside the Fragment**

```

@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
    Bundle savedInstanceState) {
    mDrawerListView = (ListView) inflater.inflate(
        R.layout.fragment_navigation_drawer, container, false);

    mDrawerListView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
            selectItem(position);
        }
    });

    mDrawerListView.setAdapter(new ArrayAdapter<String>(
        getActionBar().getThemedContext(),
        android.R.layout.simple_list_item_activated_1,
        android.R.id.text1,
        new String[]{
            getString(R.string.title_section1),
            getString(R.string.title_section2),
            getString(R.string.title_section3),
        }
    ));

    mDrawerListView.setItemChecked(mCurrentSelectedPosition, true);
    return mDrawerListView;
}

```

## 2. Actions when an item is selected on the navigation drawer

```
private void selectItem(int position) {  
    mCurrentSelectedPosition = position;  
    if (mDrawerListView != null) {  
        // Highlight the selected entry  
        mDrawerListView.setItemChecked(position, true);  
    }  
    if (mDrawerLayout != null) {  
        // Close the drawer  
        mDrawerLayout.closeDrawer(mFragmentContainerView);  
    }  
    if (mCallbacks != null) {  
        // Tell the Activity about the selected item, and let  
        //     Activity decides what to do with it  
        mCallbacks.onNavigationDrawerItemSelected(position);  
    }  
}
```

## Handling Click Events

### 1. Declare an interface in Fragment

```
/**
 * Callbacks interface that all activities using this fragment must implement.
 */
public static interface NavigationDrawerCallbacks {
    /**
     * Called when an item in the navigation drawer is selected.
     */
    void onNavigationDrawerItemSelected(int position);
}
```

### 2 Implement the Internface in Activity



```
@Override
public void onNavigationDrawerItemSelected(int position) {
    FragmentManager fragmentManager = getFragmentManager();

    switch (position) {
        case 0:
            fragmentManager.beginTransaction()
                .replace(R.id.container, MovieListFragment.newInstance(position))
                .commit();
            break;
        case 1:
            fragmentManager.beginTransaction()
                .replace(R.id.container, MovieGridFragment.newInstance(position))
                .commit();
            break;
    }
}
```



```

/**
 * Users of this fragment must call this method to set up the navigation drawer interactions.
 *
 * @param fragmentId The android:id of this fragment in its activity's layout.
 * @param drawerLayout The DrawerLayout containing this fragment's UI.
 */
public void setUp(int fragmentId, DrawerLayout drawerLayout) {
    mFragmentContainerView = getActivity().findViewById(fragmentId);
    mDrawerLayout = drawerLayout;

    // set a custom shadow that overlays the main content when the drawer opens
    mDrawerLayout.setDrawerShadow(R.drawable.drawer_shadow, GravityCompat.START);
    // set up the drawer's list view with items and click listener

    ActionBar actionBar = getActionBar();
    actionBar.setDisplayHomeAsUpEnabled(true);
    actionBar.setHomeButtonEnabled(true);

    // ActionBarDrawerToggle ties together the the proper interactions
    // between the navigation drawer and the action bar app icon.
    mDrawerToggle = new ActionBarDrawerToggle(
        getActivity(),                    /* host Activity */
        mDrawerLayout,                   /* DrawerLayout object */
        R.drawable.ic_drawer,            /* nav drawer image to replace 'Up' caret */
        "Open navigation drawer",        /* "open drawer" description for accessibility */
        "Close navigation drawer"       /* "close drawer" description for accessibility */
    ) {
        @Override
        public void onDrawerClosed(View drawerView) {
            super.onDrawerClosed(drawerView);
            if (!isAdded()) { return; }

```

```

        super.onDrawerClosed(drawerView);
        if (!isAdded()) { return; }

        getActivity().invalidateOptionsMenu();
    }

    @Override
    public void onDrawerOpened(View drawerView) {
        super.onDrawerOpened(drawerView);
        if (!isAdded()) { return; }
        if (!mUserLearnedDrawer) {
            // The user manually opened the drawer; store this flag to prevent auto-showing
            // the navigation drawer automatically in the future.
            mUserLearnedDrawer = true;
            SharedPreferences sp = PreferenceManager
                .getDefaultSharedPreferences(getActivity());
            sp.edit().putBoolean(PREF_USER_LEARNED_DRAWER, true).apply();
        }

        getActivity().invalidateOptionsMenu();
    }

};

// If the user hasn't 'learned' about the drawer, open it to introduce them to the drawer,
// per the navigation drawer design guidelines.
if (!mUserLearnedDrawer && !mFromSavedInstanceState) {
    mDrawerLayout.openDrawer(mFragmentContainerView);
}

// Defers code dependent on restoration of previous instance state

```

```

}

// Defer code dependent on restoration of previous instance state.
mDrawerLayout.post(new Runnable() {
    @Override
    public void run() {
        mDrawerToggle.syncState();
    }
});

mDrawerLayout.setDrawerListener(mDrawerToggle);
}

```

**Indicate that the fragment would like to influence action bar.**

```

@Override
public void onActivityCreated (Bundle savedInstanceState) {
    super.onActivityCreated(savedInstanceState);
    // Indicate that this fragment would like to influence the set of actions in the action bar.
    setHasOptionsMenu(true);
}

```

## Listen for Open and Close Events

### Method 1:

To listen for drawer open and close events, call `setDrawerListener()` on your `DrawerLayout` and pass it an implementation of `DrawerLayout.DrawerListener`. This interface provides callbacks for drawer events such as `onDrawerOpened()` and `onDrawerClosed()`.

### Method 2: If your activity includes an action bar.

However, rather than implementing the `DrawerLayout.DrawerListener`, if your activity includes the `action bar`, you can instead extend the `ActionBarDrawerToggle` class. The `ActionBarDrawerToggle` implements `DrawerLayout.DrawerListener` so you can still override those callbacks, but it also facilitates the proper interaction behavior between the action bar icon and the navigation drawer (discussed further in the next section).

As discussed in the `Navigation Drawer` design guide, you should modify the contents of the action bar when the drawer is visible, such as to change the title and remove action items that are contextual to the main content. The following code shows how you can do so by overriding `DrawerLayout.DrawerListener` callback methods with an instance of the `ActionBarDrawerToggle` class:

```

// ActionBarDrawerToggle ties together the the proper interactions
// between the navigation drawer and the action bar app icon.
mDrawerToggle = new ActionBarDrawerToggle(
    getActivity(),                    /* host Activity */
    mDrawerLayout,                    /* DrawerLayout object */
    R.drawable.ic_drawer,             /* nav drawer image to replace 'Up' caret */
    "Open navigation drawer", /* "open drawer" description for accessibility */
    "Close navigation drawer" /* "close drawer" description for accessibility */
) {
    @Override
    public void onDrawerClosed(View drawerView) {
        super.onDrawerClosed(drawerView);
        if (!isAdded()) { return; }

        getActivity().invalidateOptionsMenu();
    }

    @Override
    public void onDrawerOpened(View drawerView) {
        super.onDrawerOpened(drawerView);
        if (!isAdded()) { return; }

        getActivity().invalidateOptionsMenu();
    }
};

```

```
// Defer code dependent on restoration of previous instance state.  
mDrawerLayout.post(new Runnable() {  
    @Override  
    public void run() {  
        mDrawerToggle.syncState();  
    }  
});
```

```
mDrawerLayout.setDrawerListener(mDrawerToggle);
```

```
public void invalidateOptionsMenu ()
```

Added in [API level 11](#)

Declare that the options menu has changed, so should be recreated. The `onCreateOptionsMenu(Menu)` method will be called the next time it needs to be displayed.

It causes action bars to be redrawn via the following order: Activity -> Navigation Drawer -> MovieListFragment

The Navigation Drawer's action bar menu overwrites Activity's one

The MovieListFragment's action bar menu adds to the action bar.

Basically, without the MovieListFragment's menu, the action bar either shows the Activity's or the Navigation Drawer's menu.

### Inside Activity's `onCreateOptionsMenu()`



```

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    if (!mNavigationDrawerFragment.isDrawerOpen()) {
        // Only show items in the action bar relevant to this screen
        // if the drawer is not showing. Otherwise, let the drawer
        // decide what to show in the action bar.
        getMenuInflater().inflate(R.menu.main, menu);
        restoreActionBar();
        return true;
    }
    return super.onCreateOptionsMenu(menu);
}

public void restoreActionBar() {
    ActionBar actionBar = getActionBar();
    actionBar.setNavigationMode(ActionBar.NAVIGATION_MODE_STANDARD);
    actionBar.setDisplayShowTitleEnabled(true);
    actionBar.setTitle(mTitle);
}

```

## Inside NavigationDrawerFragment's onCreateOptionsMenu()

```

@Override
public void onCreateOptionsMenu(Menu menu, MenuInflater inflater) {
    // If the drawer is open, show the global app actions in the action bar. See also
    // showGlobalContextActionBar, which controls the top-left area of the action bar.
    if (mDrawerLayout != null && isDrawerOpen()) {
        inflater.inflate(R.menu.global, menu);
        showGlobalContextActionBar();
    }
    super.onCreateOptionsMenu(menu, inflater);
}

```

```

/**
 * Per the navigation drawer design guidelines, updates the action bar to show the global app
 * 'context', rather than just what's in the current screen.
 */
private void showGlobalContextActionBar() {
    ActionBar actionBar = getActionBar();
    actionBar.setDisplayHomeAsUpEnabled(true);
    actionBar.setNavigationMode(ActionBar.NAVIGATION_MODE_STANDARD);
    actionBar.setTitle("Navigation Drawer");
}

```

## Inside MovieListFragment's onCreateOptionsMenu()

```

@Override
public void onCreateOptionsMenu(Menu menu, MenuInflater inflater) {
    Log.d("MovieListFragment", "onCreateOptionsMenu");
    inflater.inflate(R.menu.fragment_list_menu, menu);
    getActivity().getActionBar().setTitle("Movie ListView");
    super.onCreateOptionsMenu(menu, inflater);
}

```

## Calling Sequence of onCreateOptionsMenu()

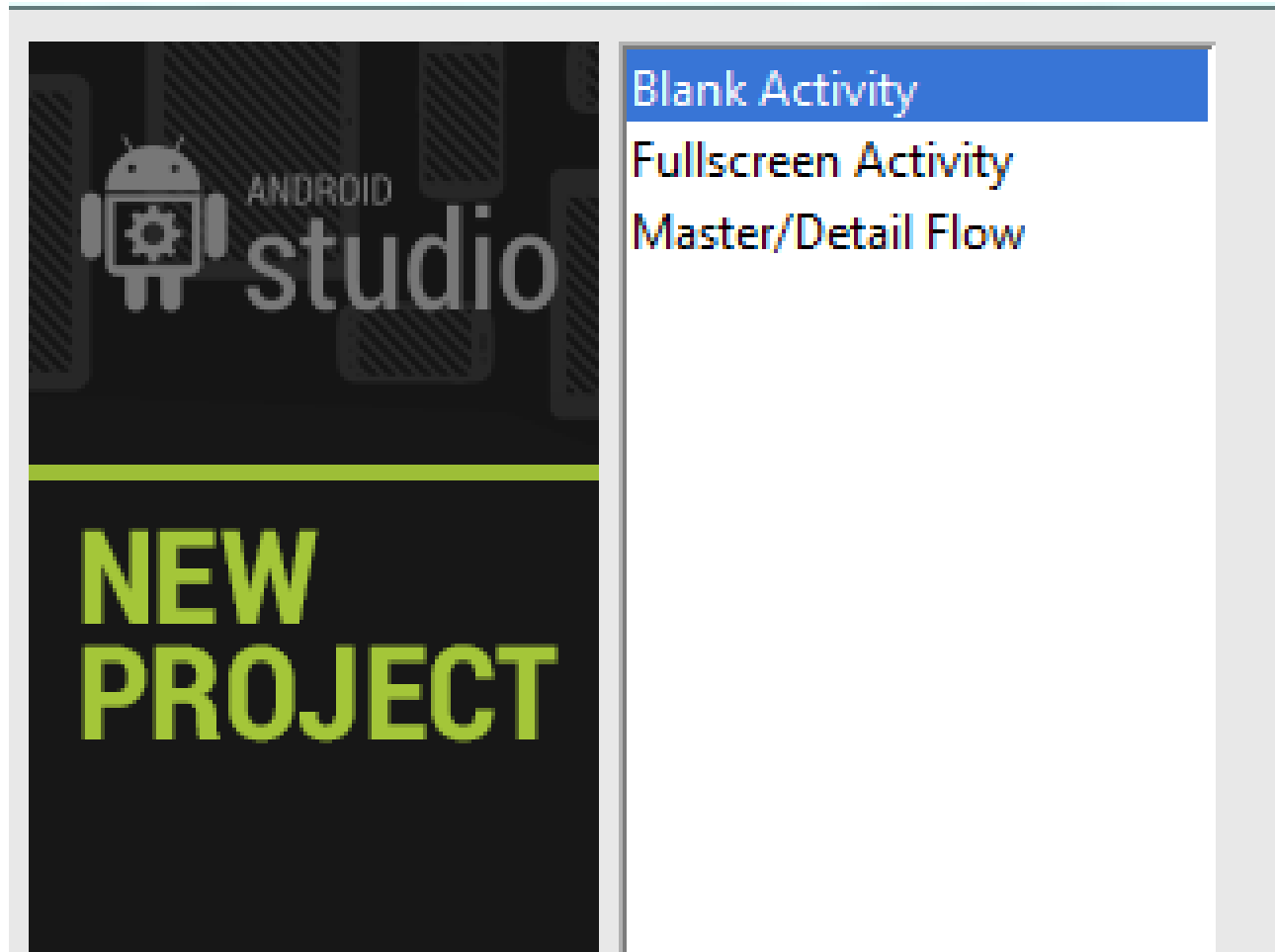
Activity → Navigation Drawer Fragment → Fragment

## Reference:

<http://developer.android.com/training/implementing-navigation/nav-drawer.html>


## Android Studio: Create a skeleton app with Navigation Drawer

### 1. When creating a new project, select "Blank Activity"



### 2. Choose "Navigation Drawer" in the "Additional Features" option

New Project

 ANDROID studio

**NEW PROJECT**

Activity Name	MainActivity
Layout Name	activity_main
Fragment Layout Name	fragment_main
Additional Features	Navigation Drawer ▼