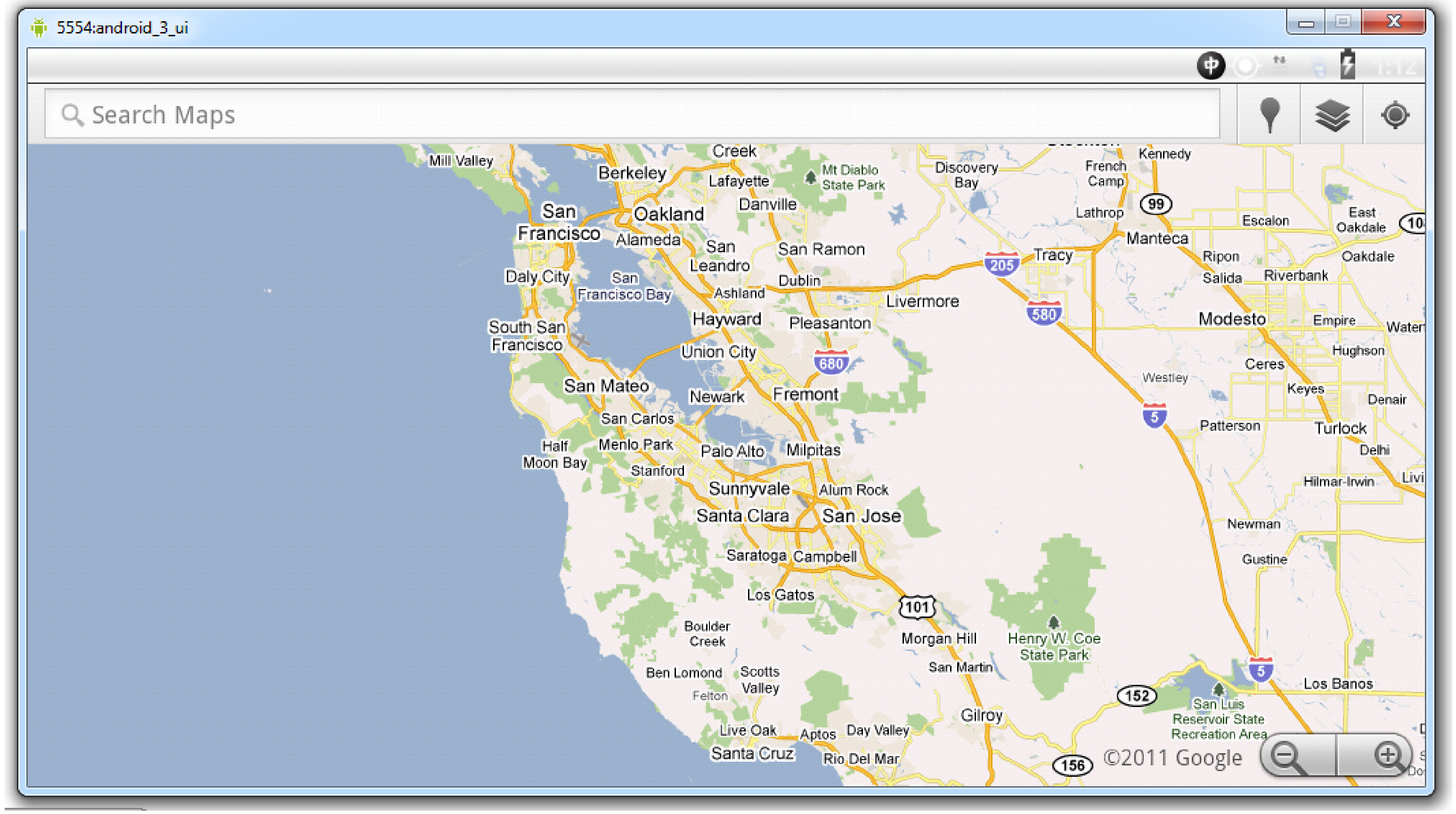
Action Bar

Introduction

- The Action Bar is a widget that is shown on top of the screen. It includes the application logo on its left side together with items available from the options menu on the right.
- When creating an action bar we choose which items of the options menu will be displayed. Items of the options menu that are not displayed will be accessible through a drop down list.
- The Action Bar can provide tabs for navigation between the fragments.

Introduction



Adding Action Bar

- There is no need to add the action bar. It is already included by default in all activities that target android 3.0 (at the minimum).
- The "holographic" theme, which is the default theme when targeting android 3, is the one that creates the action bar.
- An application is considered to "target" Android 3.0 when either `android:minSdkVersion` or `android:targetSdkVersion` attribute in the `<uses-sdk>` element was set to "11" or greater.

Removing Action Bar

- We can remove the action bar from a specific activity by setting its theme to `Theme.Holo.NoActionBar`.

```
<activity android:theme="@android:style/Theme.Holo.NoActionBar">
```

- We can programmatically show and hide the action bar by calling the `hide()` and `show()` methods accordingly.

Adding Action Items

- Each menu item in our options menu can be an action item in our action bar.

Adding Action Items

ActionBarDemoActivity.java

```
public class ActionBarDemoActivity extends Activity
{
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button showBt = (Button) findViewById(R.id.showBt);
        showBt.setOnClickListener(new OnClickListener()
        {
            public void onClick(View view)
            {
                ActionBar actionBar = getActionBar();
                actionBar.show();
            }
        });
```

Adding Action Items

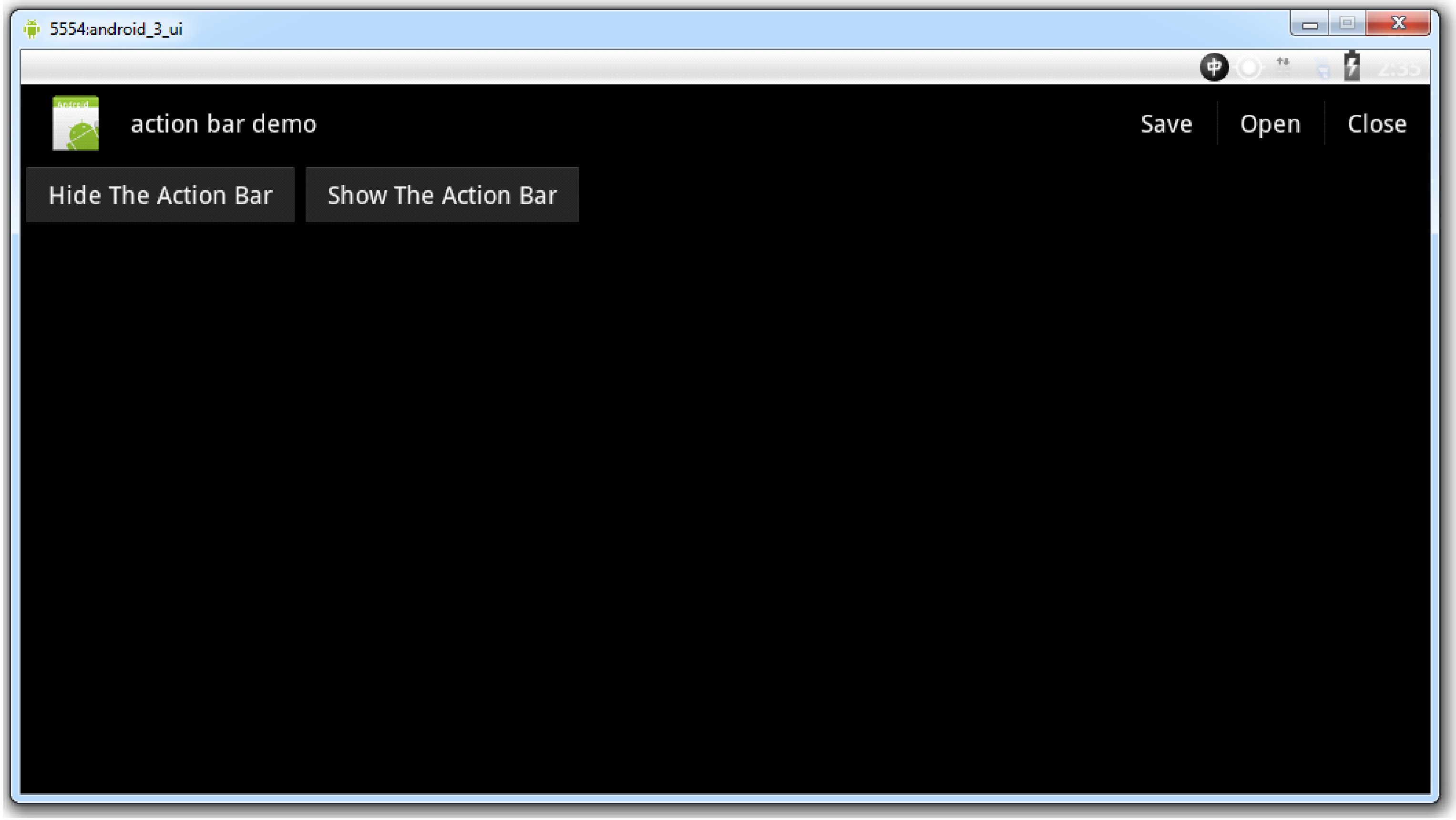
```
        Button hideBt = (Button) findViewById(R.id.hideBt);
        hideBt.setOnClickListener(new OnClickListener()
        {
            public void onClick(View view)
            {
                ActionBar actionBar = getActionBar();
                actionBar.hide();
            }
        });
    }

    public boolean onCreateOptionsMenu(Menu menu)
    {
        super.onCreateOptionsMenu(menu);
        MenuItem add = menu.add(0, 1, 0, "Save");
        MenuItem open = menu.add(0, 2, 1, "Open");
        MenuItem close = menu.add(0, 3, 2, "Close");
        add.setShowAsAction(MenuItem.SHOW_AS_ACTION_IF_ROOM);
        open.setShowAsAction(MenuItem.SHOW_AS_ACTION_IF_ROOM);
        close.setShowAsAction(MenuItem.SHOW_AS_ACTION_IF_ROOM);
        return true;
    }
}
```

Adding Action Items

main.xml

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

<Button android:layout_width="wrap_content"
android:layout_height="wrap_content" android:id="@+id/hideBt"
android:text="Hide The Action Bar"></Button>
<Button android:layout_width="wrap_content"
android:layout_height="wrap_content" android:id="@+id/showBt"
android:text="Show The Action Bar"></Button>

</LinearLayout>
```

Adding Action Items



Adding Action Items

- When creating the menu using XML we can mark those menu items we wan to be in our action bar.

Adding Action Items

ActionBarXMLActivity.java

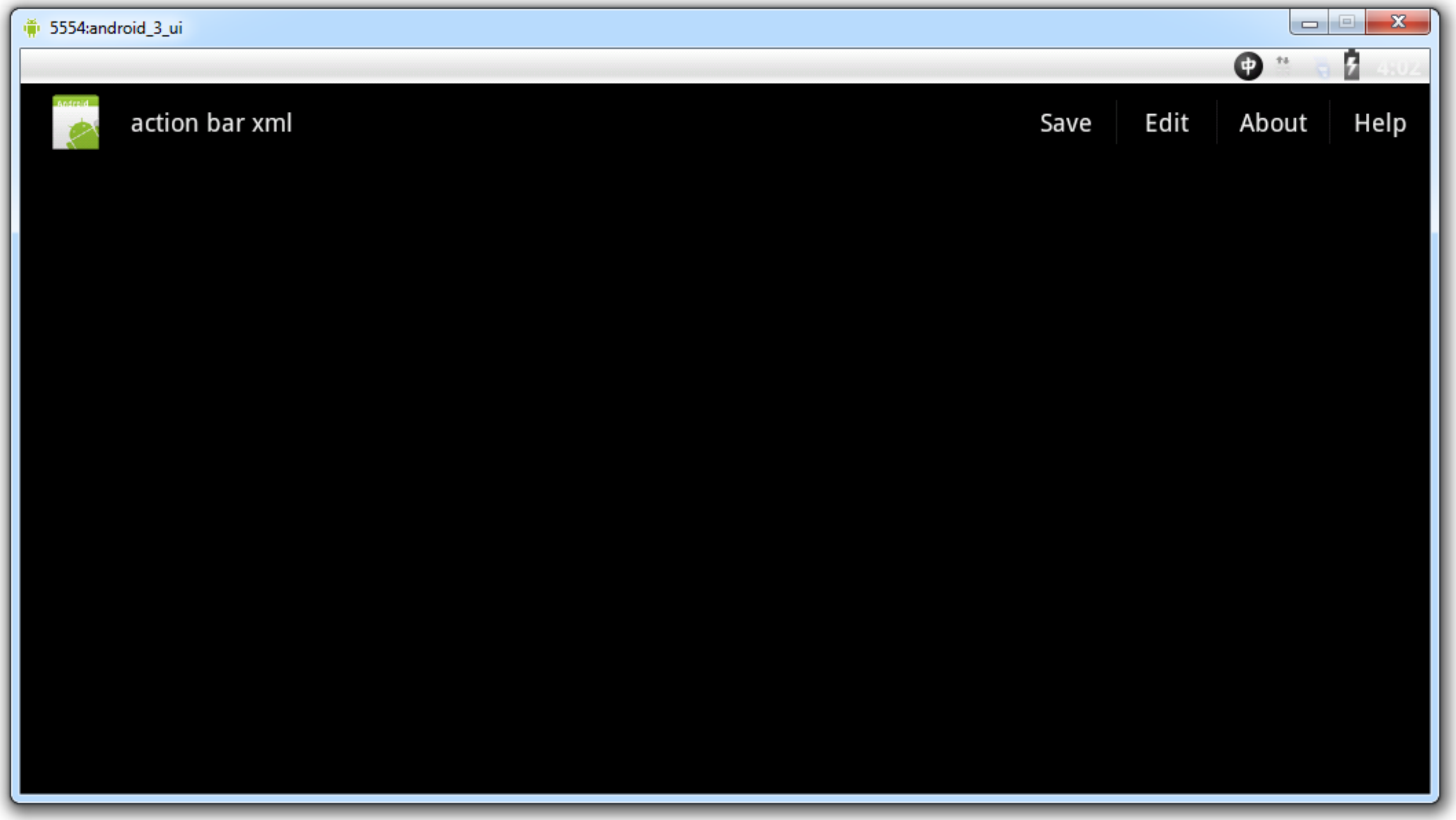
```
public class ActionBarXMLActivity extends Activity
{
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu)
    {
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.optionsmenu, menu);
        return true;
    }
}
```

Adding Action Items

optionsmenu.xml

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item   android:orderInCategory="1" android:id="@+id/item1"
            android:showAsAction="ifRoom|withText"
            android:title="@string/save" />
    <item   android:orderInCategory="2" android:id="@+id/item1"
            android:showAsAction="ifRoom|withText"
            android:title="@string/edit" />
    <item   android:orderInCategory="3" android:id="@+id/item1"
            android:showAsAction="ifRoom|withText"
            android:title="@string/about" />
    <item   android:orderInCategory="4" android:id="@+id/item1"
            android:showAsAction="ifRoom|withText"
            android:title="@string/help" />
</menu>
```

Adding Action Items



The Overflow Menu

- When creating an action bar with too many items the ones that won't have room will be displayed in a separated overflow menu accessible through the top right corner of the screen.

The Overflow Menu

```
public class ActionBarXMLActivity extends Activity
{
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu)
    {
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.optionsmenu, menu);
        return true;
    }
}
```

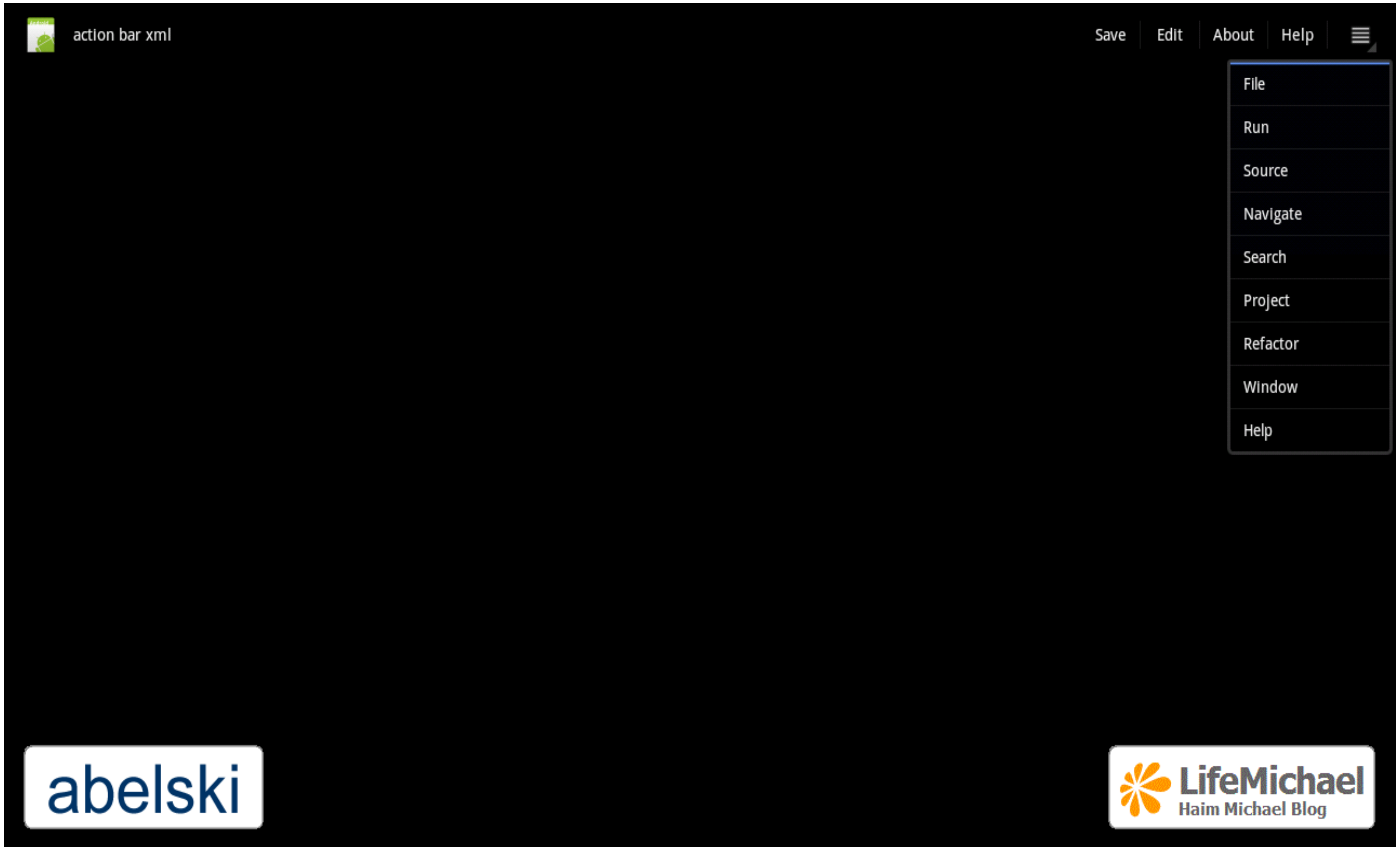
The Overflow Menu

```
<?xml version="1.0" encoding="utf-8"?>
<menu   xmlns:android="http://schemas.android.com/apk/res/android">
    <item   android:orderInCategory="1" android:id="@+id/item1"
            android:showAsAction="ifRoom|withText"
            android:title="@string/save" />
    <item   android:orderInCategory="2" android:id="@+id/item2"
            android:showAsAction="ifRoom|withText"
            android:title="@string/edit" />
    <item   android:orderInCategory="3" android:id="@+id/item3"
            android:showAsAction="ifRoom|withText"
            android:title="@string/about" />
    <item   android:orderInCategory="4" android:id="@+id/item4"
            android:showAsAction="ifRoom|withText"
            android:title="@string/help" />
    <item   android:orderInCategory="5" android:id="@+id/item5"
            android:showAsAction="ifRoom|withText"
            android:title="@string/file" />
    <item   android:orderInCategory="6" android:id="@+id/item6"
            android:showAsAction="ifRoom|withText"
            android:title="@string/run" />
```

The Overflow Menu

```
    <item   android:orderInCategory="7" android:id="@+id/item7"
            android:showAsAction="ifRoom|withText"
            android:title="@string/source" />
    <item   android:orderInCategory="8" android:id="@+id/item8"
            android:showAsAction="ifRoom|withText"
            android:title="@string/navigate" />
    <item   android:orderInCategory="9" android:id="@+id/item9"
            android:showAsAction="ifRoom|withText"
            android:title="@string/search" />
    <item   android:orderInCategory="10" android:id="@+id/item10"
            android:showAsAction="ifRoom|withText"
            android:title="@string/project" />
    <item   android:orderInCategory="11" android:id="@+id/item11"
            android:showAsAction="ifRoom|withText"
            android:title="@string/refactor" />
    <item   android:orderInCategory="12" android:id="@+id/item12"
            android:showAsAction="ifRoom|withText"
            android:title="@string/window" />
    <item   android:orderInCategory="13" android:id="@+id/item13"
            android:showAsAction="ifRoom|withText"
            android:title="@string/help" />
</menu>
```

The Overflow Menu



The `withText` Flag

- The default behavior when the menu item has both an icon and a text is to show the icon only.
- If we want to show the text then we should add the `withText` flag (when the menu is created using XML).
- When creating the menu in our code we should use the `SHOW_AS_ACTION_WITH_TEXT` flag calling the `setShowAsAction()` method.

Action Events

- The menu items placed in our action bar trigger the same callback methods as any other item the options menu includes.
- When the user selects a menu item from a fragment the `onOptionsItemSelected()` method is first called for the activity that uses the fragment. If it returns false then there will be a call to the `onOptionsItemSelected()` method defined in the fragment.

The Application Icon

- The application icon appears in the action bar on its left side. This is the default behavior.
- When the user taps the application bar it responds the same way action items do. The system calls your activity's `onOptionsItemSelected()` method with the `android.R.id.home` ID. We can override this method and add a condition in order perform the appropriate action, such as to start the main activity of our application instead of getting the user back to the home screen.

The Application Icon

- If we choose to return the user back to the main activity of our application then we better use the `FLAG_ACTIVITY_CLEAR_TOP` flag in the Intent object we create.
- Using this flag if the main activity of our application already exists in the activities task then all activities on top of it will be destroyed and the main activity of our application will be brought to the front and we wont end up with the creation of new instances of the main activity of our application.

The Application Icon

```
@Override
public boolean onOptionsItemSelected(MenuItem item)
{
    switch (item.getItemId())
{
        case android.R.id.home:
            Intent intent = new Intent(this, MainActivity.class);
            intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
            startActivity(intent);
            return true;
        default:
            return super.onOptionsItemSelected(item);
    }
}
```

The Application Icon

- When we implement code that when the user taps the action bar icon it takes him back to the previous activity we can indicate about this behavior by calling the `setDisplayHomeAsUpEnabled(true)` method on our action bar.

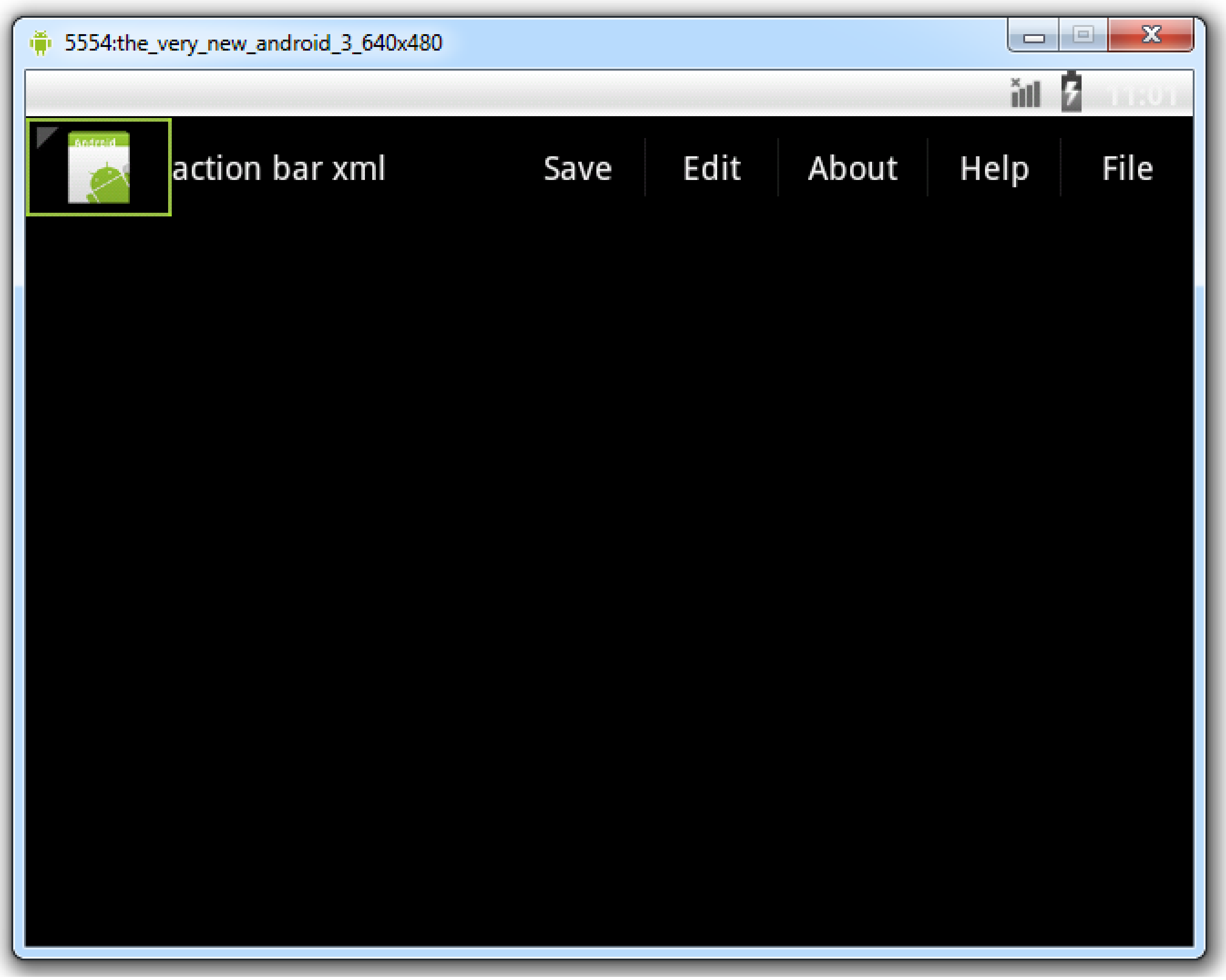
The Application Icon

```
public class HomyUpActivity extends Activity
{
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }

    @Override
    protected void onStart()
    {
        super.onStart();
        ActionBar actionBar = this.getActionBar();
        actionBar.setDisplayHomeAsUpEnabled(true);
    }
```

The Application Icon

```
    @Override
    public boolean onCreateOptionsMenu(Menu menu)
    {
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.optionsmenu, menu);
        return true;
    }
}
```

The Application Icon



Adding Action View

- We can add a view as an action item. We do so by adding into the item xml element the `android:actionLayout` attribute assigned with the id of the layout resource we want to display (e.g. `@layout/mysearchview`).
- We can alternatively add the `android:actionViewClass` attribute assigned with the full qualified name of the class that describes the view (e.g. `android.widget.SearchView`) we want to display.

Adding Action View

optionsmenu.xml

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

    <item android:orderInCategory="4" android:id="@+id/item4"
        android:showAsAction="ifRoom|withText"
        android:title="@string/help" />
    <item android:orderInCategory="5" android:id="@+id/item5"
        android:showAsAction="ifRoom|withText"
        android:title="@string/file" />
    <item android:id="@+id/search"
        android:title="Search"
        android:showAsAction="ifRoom"
        android:actionViewClass="android.widget.SearchView" />

</menu>
```

Adding Action View

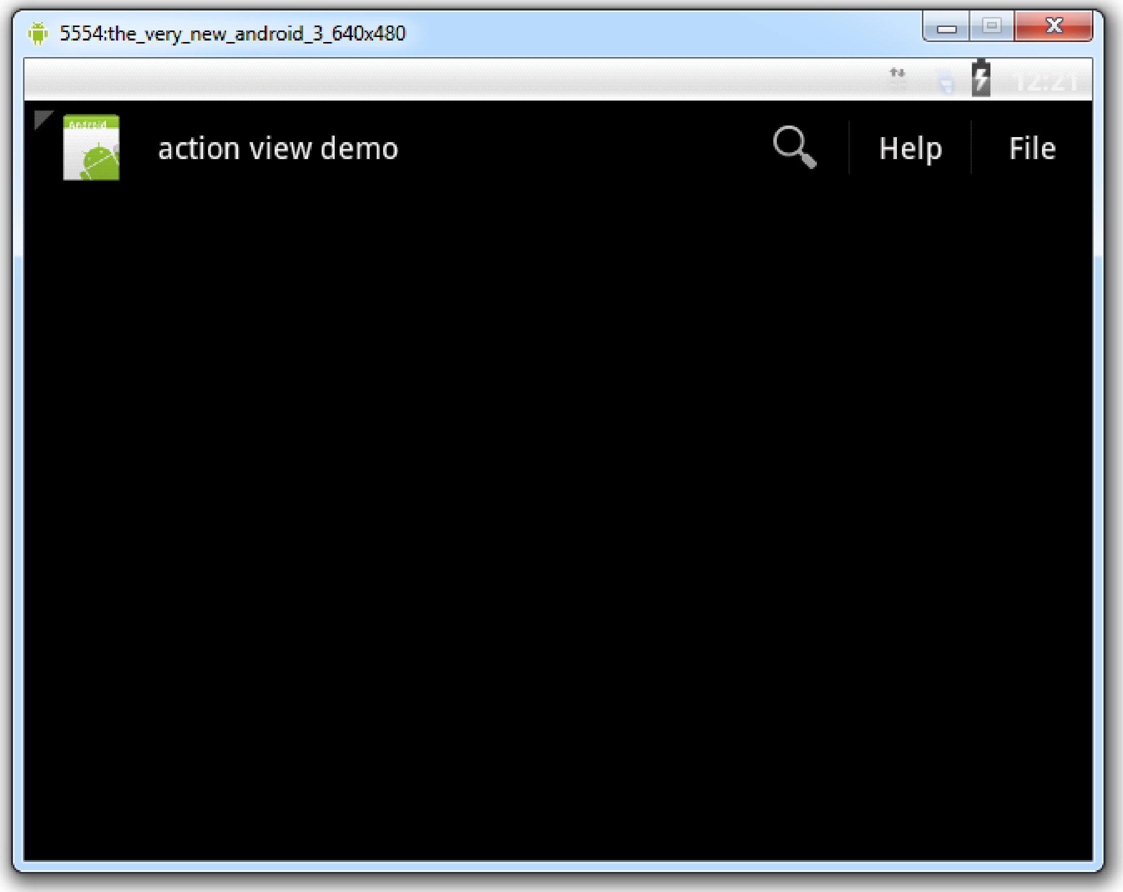
HomeUpActivity.java

```
public class HomyUpActivity extends Activity
{
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }

    @Override
    protected void onStart()
    {
        super.onStart();
        ActionBar actionBar = this.getActionBar();
        actionBar.setDisplayHomeAsUpEnabled(true);
    }
```

Adding Action View

```
    @Override
    public boolean onCreateOptionsMenu(Menu menu)
    {
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.optionsmenu, menu);
        return true;
    }
}
```

Adding Action View



Action Bar Tabs

- The Action Bar can display tabs that allow the user navigating between different fragments the activity uses.
- Each one of the tabs can be with an icon and/or a textual title.

Action Bar Tabs

TabsActivity.java

```
public class TabsActivity extends Activity
{
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        ActionBar bar = getActionBar();
        bar.setNavigationMode(ActionBar.NAVIGATION_MODE_TABS);
        ActionBar.Tab tabA = bar.newTab().setText("A Tab");
        ActionBar.Tab tabB = bar.newTab().setText("B Tab");
        ActionBar.Tab tabC = bar.newTab().setText("C Tab");
        Fragment fragmentA = new AFragmentTab();
        Fragment fragmentB = new BFragmentTab();
        Fragment fragmentC = new CFragmentTab();
```

Action Bar Tabs

```
bar.addTab(tabA);
bar.addTab(tabB);
bar.addTab(tabC);

protected class MyTabsListener implements ActionBar.TabListener
{
    private Fragment fragment;
    public MyTabsListener(Fragment fragment)
    {
        this.fragment = fragment;
    }
    @Override
    public void onTabSelected(Tab tab, FragmentTransaction ft)
    {
        ft.add(R.id.fragment_place, fragment, null);
    }
```

Action Bar Tabs

```
    @Override
    public void onTabUnselected(Tab tab, FragmentTransaction ft)
    {
        ft.remove(fragment);
    }

    @Override
    public void onTabReselected(Tab tab, FragmentTransaction ft)
    {
        //...
    }

}
```

Action Bar Tabs

AFragmentTab.java

```
public class AFragmentTab extends Fragment
{
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
        Bundle savedInstanceState)
    {
        return inflater.inflate(R.layout.fragment_a, container, false);
    }
}
```

Action Bar Tabs

BFragmentTab.java

```
public class BFragmentTab extends Fragment
{
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
        Bundle savedInstanceState)
    {
        return inflater.inflate(R.layout.fragment_b, container, false);
    }
}
```

Action Bar Tabs

CFragmentTab.java

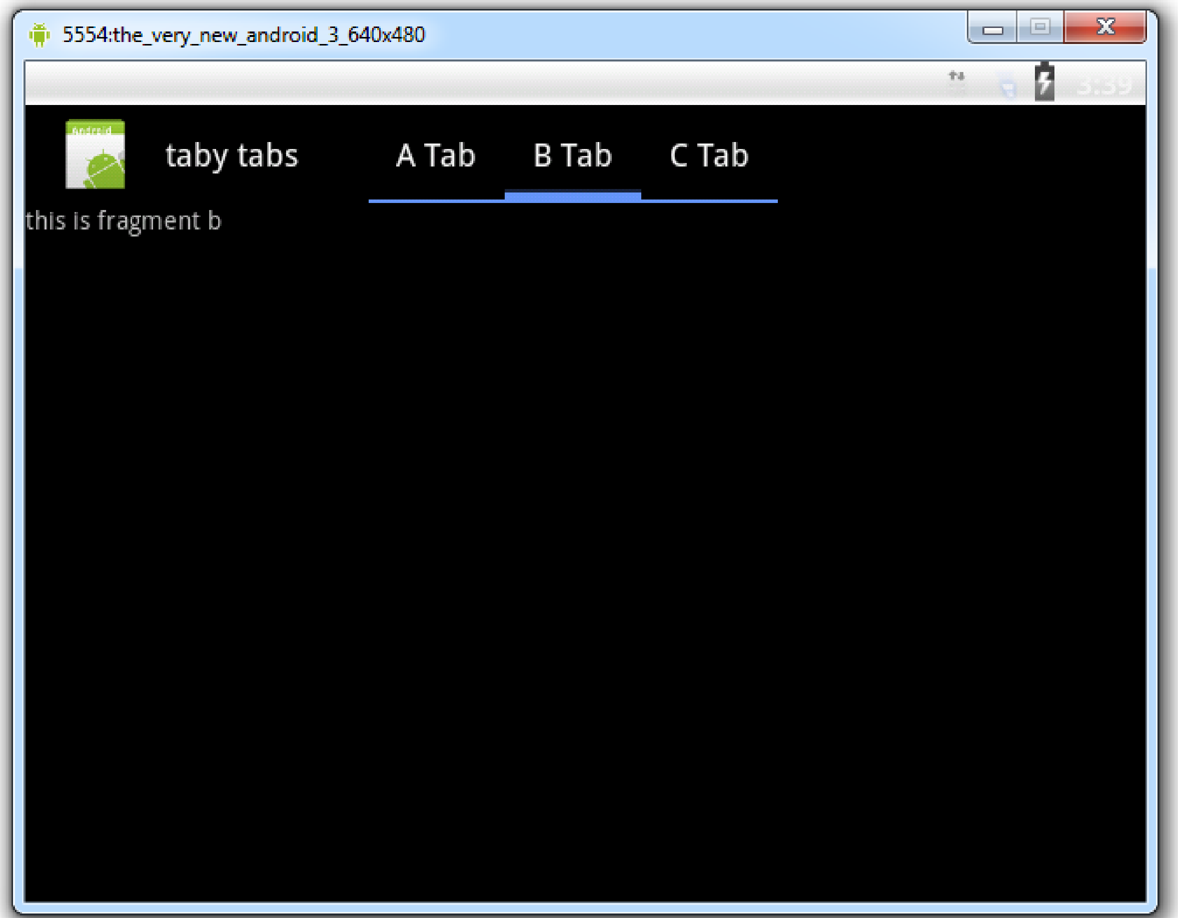
```
public class CFragmentTab extends Fragment
{
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
        Bundle savedInstanceState)
    {
        return inflater.inflate(R.layout.fragment_c, container, false);
    }
}
```

Action Bar Tabs

main.xml

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

<LinearLayout android:layout_height="wrap_content"
android:layout_width="match_parent"
android:id="@+id/fragment_place"></LinearLayout>

</LinearLayout>
```

Action Bar Tabs



this is fragment b