Creating Tabbed Applications

Xamarin Android Level 1, Chapter 6

Xamarin Inc.

Overview

Tabs are a popular User Interface pattern in mobile applications because of their simplicity and usability. They allow for an easy way to divide up an application into navigable sections and provide a constant way to switch between them. This chapter discusses two ways to create tabbed layouts in Xamarin.Android and then proceeds with a walkthrough on the more robust of the two techniques.

Introduction to Tab Layout

In order to implement a tabbed interface, an application must contain a `TabHost`, which then contains a `TabWidget` for displaying the tabs and a `FrameLayout` for displaying the tab content.

The content of tabs can be implemented one of two ways; each tab can be a separate Activity that is launched when that tab is selected, or each tab can be a different view within a single Activity. Which method an application should use depends on the requirements for the application. In many cases each tab will be a distinct user task, so using an Activity per tab will result in a codebase that is modular and easier to maintain.

Creating a Tabbed Application Walkthrough

Lets create a tabbed application that uses a separate Activity for each tab.

1. Start a new project named `HelloTabWidget`.
2. Next, create three separate Activity classes in your project: `ArtistsActivity`, `AlbumsActivity`, and `SongsActivity`. These will each represent a separate tab. For now, make each one display a simple message using a `TextView`. For example:

```
[Activity]
public class ArtistsActivity : Activity
{
    protected override void OnCreate (Bundle savedInstanceState)
    {
        base.OnCreate (savedInstanceState);

        TextView textview = new TextView (this);
        textview.Text = "This is the Artists tab";
        SetContentView (textview);
    }
}
```

   Notice that this doesn't use a layout file. Just create a **TextView**, give it some text and set that as the content. Duplicate this for each of the three activities.

3. Each tab will require an icon. Download the file `06 - Creating Tabbed Applications - TabIcons.zip`, and extract the two icons in the zip file to the folder `/Resources/drawable`. Ensure that the **Build Action** is set to **AndroidResource**.

4. Next add an XML file to the folder `/Resources/drawable` named `ic_tab_artists.xml`. Ensure that the **Build Action** of this new file is set to **AndroidResource**, and then edit this file by inserting the following XML:

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <!-- When selected, use grey -->
    <item android:drawable="@drawable/ic_tab_artists_grey"
          android:state_selected="true"/>
    <!-- When not selected, use white-->
    <item android:drawable="@drawable/ic_tab_artists_white"/>
</selector>
```

This XML file defines a *State-List Drawable*. Recall from Chapter 4 – Displaying Data in Lists, state-list drawables are a special drawable resource that allow you to specify different that are specific to that item's state. In this example there is one image that is used when a tab is selected, and another that is used when the tab is not selected.

5. Next, we need to create the layout file that will host the tabs. Open the `Resources/Layout/Main.axml` file and insert the following:

```
<?xml version="1.0" encoding="utf-8"?>
<TabHost xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@android:id/tabhost"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <LinearLayout
        android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:padding="5dp">
        <TabWidget
            android:id="@android:id/tabs"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"/>
        <FrameLayout
            android:id="@android:id/tabcontent"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:padding="5dp"/>
    </LinearLayout>
</TabHost>
```

The `TabHost` must have two child views inside it: a `TabWidget` and a `FrameLayout`. To position the `TabWidget` and `FrameLayout` vertically, a `LinearLayout` is used. The `FrameLayout` is where the content for each tab goes, which is empty now because the `TabHost` will automatically embed each Activity within it.

Notice that the `TabWidget` and the `FrameLayout` elements have the IDs `tabs` and `tabcontent`, respectively. These exact names must be used so that the `TabHost` can retrieve references to each of them.

6. Now open `HelloTabWidget.cs` and make it subclass `TabActivity`:

```
[Activity (MainLauncher=true, Label="@string/app_name",
Theme="@android:style/Theme.NoTitleBar")] public class HelloTabWidget :
TabActivity { }
```

`TabHost` requires that any activity it manages be derived from *TabActivity*. Therefore, it is important to subclass `TabActivity` here – a regular Activity will not work.

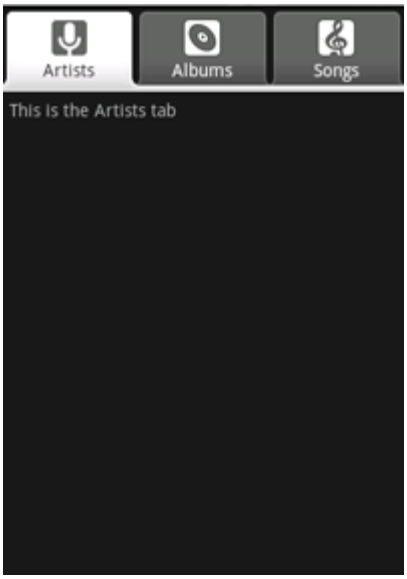
7. Next `OnCreate` must be modified to initialize the tabs. Use the following code for the `OnCreate()` method:

```
protected override void OnCreate (Bundle bundle)
{
    base.OnCreate (bundle);
    SetContentView (Resource.Layout.Main);

    TabHost.TabSpec spec;     // Resusable TabSpec for each tab
    Intent intent;            // Reusable Intent for each tab

    // Create an Intent to launch an Activity for the tab (to be reused)
    intent = new Intent (this, typeof (ArtistsActivity));
    intent.AddFlags (ActivityFlags.NewTask);

    // Initialize a TabSpec for each tab and add it to the TabHost
    spec = TabHost.NewTabSpec ("artists");
    spec.SetIndicator ("Artists", Resources.GetDrawable
(Resource.Drawable.ic_tab_artists));
    spec.SetContent (intent);
    TabHost.AddTab (spec);

    // Do the same for the other tabs
    intent = new Intent (this, typeof (AlbumsActivity));
    intent.AddFlags (ActivityFlags.NewTask);

    spec = TabHost.NewTabSpec ("albums");
    spec.SetIndicator ("Albums", Resources.GetDrawable
(Resource.Drawable.ic_tab_artists));
    spec.SetContent (intent);
    TabHost.AddTab (spec);

    intent = new Intent (this, typeof (SongsActivity));
    intent.AddFlags (ActivityFlags.NewTask);

    spec = TabHost.NewTabSpec ("songs");
    spec.SetIndicator ("Songs", Resources.GetDrawable
(Resource.Drawable.ic_tab_artists));
    spec.SetContent (intent);
    TabHost.AddTab (spec);

    TabHost.CurrentTab = 2;
}
```

The `TabHost.TabSpec` class is a specialized builder contains the data specific each tab: the text and icon, the content, and a unique tag to identify it. Each tab will have a `TabHost.TabSpec` instance created for it, and added to the `TabHost`. When

Android displays the `TabHost`, it will use `TabHost.TabSpec` the to render the tab in the `TabWidget` and launch the Activity for the tab's content.

8. Run the application. Your application should look something like the following:



That's it! As we can see, tabs are extremely simple to setup, yet they provide a sophisticated way to navigate to different sections of an application.

Summary

This chapter discussed tabbed layouts and guided the student through creating a tabbed application. The tabbed application demonstrated how to use a `TabActivity` to inflate a layout file that hosting a `TabHost` and a `TabWidget`. The `TabHost` was then populated with a collection of `TabHost.TabSpec` objects which would be used by the `TabHost` at runtime to instantiate the activities that would be used in each tab.