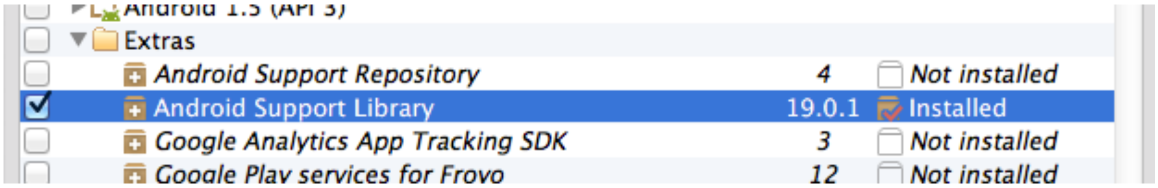
Lab 02: Tab Navigation

Prerequisites

This lab can be done with iOS, Android, or both. For Android, you will need a development environment for Android setup; either a Mac or PC with Xamarin Studio or a Windows PC with Visual Studio and the Xamarin tools installed. For iOS, remember that if you are using Windows, you will still need an accompanying Mac on your network with XCode and the Xamarin tools to build and run the code.

Many of the Android projects require Android Support Libraries (v4 and v7 AppCompat). If you do not already have them, they can be installed from the Android SDK Manager (`Tools->Open Android SDK Manager...` from the Xamarin Studio menu). Scroll down to the `Extras` and choose to install the `Android Support Library` item.



☐	Android 1.5 (API 3)		
☐	Extras		
☐	*Android Support Repository*	*4*	☐ *Not installed*
✓	Android Support Library	19.0.1	Installed
☐	*Google Analytics App Tracking SDK*	*3*	☐ *Not installed*
☐	*Google Play services for Froyo*	*12*	☐ *Not installed*

See the `Xamarin.Android` setup documentation if you need help getting your Android environment setup:

http://docs.xamarin.com/guides/android/getting_started/installation/

See the `Xamarin.iOS` setup documentation if you need help getting your iOS environment setup:

http://docs.xamarin.com/guides/ios/getting_started/installation/

Downloads

There are several projects in a single solution that we will use to explore the various mobile navigation patterns on iOS and Android. You can download the projects from the Xamarin University website:

http://university.xamarin.com/

Lab Goals

The goal of this lab will be to get an understanding of how iOS and Android handle the tab navigation pattern. By completing this lab, you will learn about:

- The iOS and Android variants of tab navigation – how they are constructed.
- How tab selection is handled on iOS or Android (or both).

Steps

Open the Starting Solution

1. Launch Xamarin Studio and open **MobileNavigationPatterns** solution file included in your lab resources.
2. Depending on the platform you want to use, make sure either **AndroidActionBarTabs** or **iOSTabs** is the startup project.
3. Proceed to the next step for iOS or skip to the next section for Android.

iOS Tab Navigation

Creating the Tab UI

1. Within the **iOSTabs** project, open **AppDelegate.cs**.
2. To present a tab navigation system on iOS, you can use the provided **UITabBarController** class. We create one of those for a class-level field in `FinishedLaunching`. and set it to be the window's `RootViewController`.

```
EvolveTabBarController evolveTabController;

public override bool FinishedLaunching (UIApplication app, NSDictionar
y options)
{
    // ...
    evolveTabController = new EvolveTabBarController ();

    // ...

    window.RootViewController = evolveTabController;

}
```

3. Tab bar controllers need to be provided a list of view controllers to present as tabs. In this case, the code to do so is isolated in a subclass of `UITabBarController`. Open **Navigation/EvolveTabBarController.cs**.
4. Inside `ViewDidLoad`, we create the controllers that will make up the tab choices. For each controller, we set up its `TabBarItem` to customize the tab's appearance. In this case, we set the title and give the tab an icon to use.

```
var vc1 = new UINavigationController (new SessionsViewController ());
vc1.TabBarItem = new UITabBarItem ("Sessions", UIImage.FromBundle ("im
ages/tabsession"), 0);
```

5. With all of the controllers created, we hand them over the tab bar controller's ViewControllers property.

```
ViewControllers = new UIViewController[] { vc1, vc2, vc3 };
```

6. Optionally, you can set the selected tab. It will default to the first tab.

```
SelectedIndex = 0;
```

7. Run the application in the simulator and switch between tabs. In iOS, this switching is handled for us by the `UITabBarController`.

Stack Within Separate Tabs

8. With the application running, click to the Sessions tab and then navigate to any session details.
9. Now click to the Speakers tab and navigate to any speaker details.
10. Flip back to the Sessions tab and notice iOS has maintained the user back stack for each tab independently for you, if this is what your app needs.

Android Tab Navigation

Creating the Tab UI

This project requires Android Support Libraries (v4 and v7 AppCompat). If you haven't already installed them from the SDK Manager, please see the **Prerequisites** section above.

1. Within the **AndroidActionBarTabs** project, open **MainActivity.cs**.
2. To create a tab navigation system in Android, you can use the ActionBar Tabs system introduced in Android v3.0 (and back-ported to earlier versions with the support libraries). First, our activity must inherit from the support `ActionBarActivity`. For simplicity, our activity will also handle the tab switching calls by implementing `ActionBar.ITabListener`.

```
public class MainActivity : ActionBarActivity, ActionBar.ITabListener
```

3. Next, in `OnCreate`, we make sure to tell the activity we will be navigating with tabs.

```
SupportActionBar.NavigationMode = ActionBar.NavigationModeTabs;
```

4. Then, we create the fragments that will be used by our tabs.

```
_fragments = new Fragment[] {
    new SessionListFragment (),
    new SpeakerListFragment (),
    new AboutFragment ()
};
```

5. Finally, we add tabs to the **ActionBar** system with the desired names and icons.

```
AddTabToActionBar (Resource.String.sessions_tab_label, Resource.Drawab
le.ic_action_sessions);
AddTabToActionBar (Resource.String.speakers_tab_label, Resource.Drawab
le.ic_action_speakers);
AddTabToActionBar (Resource.String.about_tab_label, Resource.Drawable.
ic_action_whats_on);
```

6. `AddTabToActionBar` is a helper method to do the **ActionBar** calls required to make a new tab appear.

```
ActionBar.Tab tab = SupportActionBar.NewTab ()
                                    .SetText (labelResourceId)
                                    .SetIcon (iconResourceId)
                                    .SetTabListener (this);
SupportActionBar.AddTab (tab);
```

7. If this was all the code we had, running the app in the emulator would simply show tabs that do not do anything when tapped. This is where the `ITabListener` interface comes in. This interface gives us methods that are called when a tab is selected, unselected, and reselected (clicked while already active). In this lab, we are only doing something interesting in `OnTabSelected`, the important part being swapping which fragment is being shown to the user.

```
public void OnTabSelected (ActionBar.Tab tab, FragmentTransaction ft)
{
    // ... line omitted for later explaination

    Fragment frag = _fragments [tab.Position];
    ft.Replace (Resource.Id.content_frame, frag);
}
```

8. Run the application in the emulator and click around the tabs. The fragment swap is changing out which content is shown without adding anything to the back stack.

Stack Within Separate Tabs

9. While still running the app, click through to a details fragment from the **Speakers** tab.

10. Now, press the back button once. Notice that going to the detail fragment was part of the back stack.

11. Open **SpeakerListFragment.cs** and find the `ShowDetails` method. To add the fragment replacement to the back stack, we simply added a call to `AddToBackStack` on the `FragmentTransaction`.

```
FragmentManager.BeginTransaction ()
    .Replace (Resource.Id.content_frame, details)
    .AddToBackStack (null)
    .Commit ();
```

12. Return to **MainActivity.cs**. Since we are now in control of what happens when a tab is selected, we have a couple choices about handling this back-stack navigation within a tab. We could maintain the stack manually and recreate it when a tab is selected after leaving. Alternatively, as we do in this lab, we simply clear out the back stack so that it doesn't create any issues when using the back button after switching tabs.

```
SupportFragmentManager.PopBackStack (null, FragmentManager.PopBackStac
kInclusive);
```

Summary

In this lab, we saw how iOS and Android offer tab navigation solutions for applications. We reviewed how to create tab systems on both platforms and saw how to handle tab switches by the user. As well, we saw how back stack is maintained (or not) within tabs.