Lab 01: Stack Navigation

Prerequisites

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.

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. These are included with this lab and can be found at:

Content/Exercises/NavigationiOS/Lab Resources

Lab Goals

The goal of this lab will be to get an understanding of how iOS handles navigation stacks. By completing this lab, you will learn about:

- The iOS stack history – how items are added to the back stack.
- How to create a stack, or hierarchy, navigation system on iOS.

Steps

Open the Starting Solution

1. Launch Xamarin Studio and open **MobileNavigationPatterns** solution file included in your lab resources.

iOS Stack Navigation

Creating the stack UI

1. Within the **iOSStack** project, open **AppDelegate.cs**.
2. To present a stack navigation system on iOS, you can use the provided **UINavigationController** class. We create one of those for a class-level field in `FinishedLaunching`.

```
UINavigationController evolveNavigationController;

public override bool FinishedLaunching (UIApplication app, NSDictionar
y options)
{
```

```
    // ...
    evolveNavigationController = new UINavigationController ();
    // ...
}
```

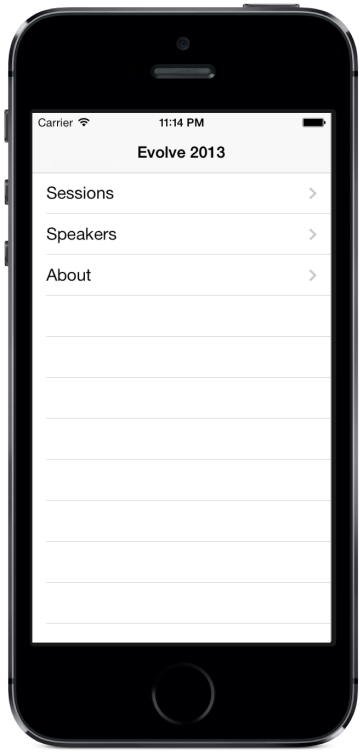
3. Navigation controllers need a root view controller to start the stack. In this case we will push on a **MenuTableViewController** with `PushViewController`. Alternatively, you could pass it in to a **UINavigationController** constructor overload.

```
evolveNavigationController.PushViewController (new MenuTableViewContro
ller (), false);
```

4. Then, as normal, we set our controller to be the RootViewController for the window.

```
window.RootViewController = evolveNavigationController;
```

5. Run the application in the simulator to see the styling that is automatically provided by the use of **UINavigationController**, then return to Xamarin Studio. Feel free to leave the app running in the simulator.



Adding to the back stack

The speaker and session lists are basic **UITableViewControllers** with simple backing **UITableViewSources**.

6. To see where we actually add to the back stack, open **SpeakersTableSource.cs**.

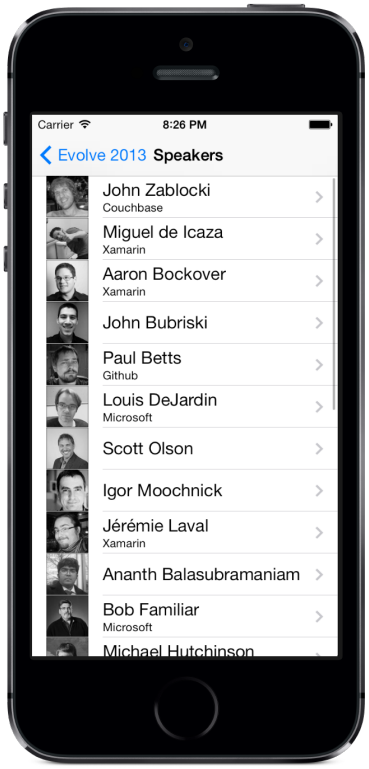
7. Inside `RowSelected`, we grab the data for the selected speaker from the data set.

```
var speaker = data [indexPath.Row];
```

8. Then we create a **SpeakerViewController** to display the details and push it onto the navigation stack. We access the current navigation controller in use, if any, from a controller's `NavigationController` property.

```
controller.NavigationController.PushViewController (new SpeakerViewCon
troller (speaker), true);
```

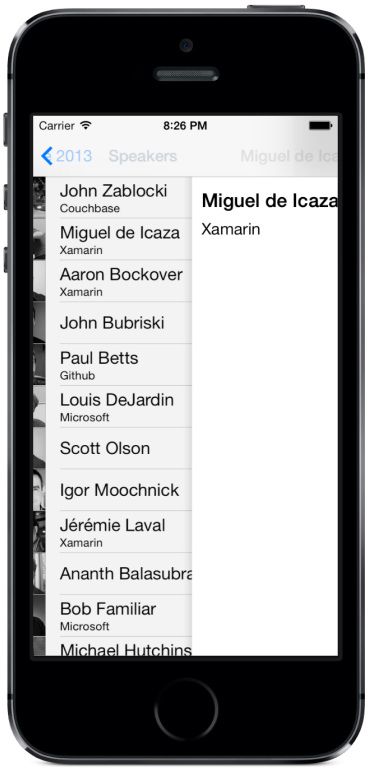
9. Return to the application in the simulator and click the Speakers item. This will animate in a new controller with the speaker list. The navigation bar will now have a titled back button that can return you to the root list. Return to Xamarin Studio. Feel free to leave the app running in the simulator.



Popping off the back stack

10. To pop back off the stack, **UINavigationController** provides a back button in the navigation bar.
11. In iOS 7 and 8 users can also use a swipe from the left-edge to trigger a back navigation as well.
12. Return to the application in the simulator and click the back button to return to the root menu.

13. [iOS 7+ only] Click through to another item. This time swipe from the left edge of the screen to see the use of the back gesture popping you back to the root menu.



Summary

In this lab, we saw how iOS offers stack navigation solutions for applications. We reviewed how items are added and popped off the stack.