XIBFiles Tutorial

Getting Started: Start a new project in XCode.

-- use the “Empty Application” Template

-- call the project “XIBFiles”

-- enable Automatic Reference Counting

-- Do not use Storyboards

-- Set the Device for iPhone only

Part 1

Objective: Add a new UIViewController that uses a XIB file to the project, and display that as the rootViewController of the application.

1. Right-click on the *XIBFiles* group in the Project Explorer and select **New File**.
2. Select **iOS->Cocoa Touch-->Objective-C Class** and click **Next**.
3. Name the class *ExampleViewController.*
4. Select **UIViewController** from the **Subclass of** selection box.
5. Ensure **With XIB for user interface** is selected.
6. Click **Next**, and then **Create.** (Ensure you're in the correct folder.)

WHAT THIS DOES: The above steps added three files to the project, the .h, .m, and .xib files for the ExampleViewController.

7. Select *ExampleViewController.xib* to open the XIB file in the XIB editor.
8. Open the **Object Library.** (View-->Utilities-->Show Object Library)
9. Select the **Round Rect Button** and drag it onto the XIB.
10. Add the following code to the d*idFinishLaunchingWithOptions:*selector in **AppDelegate.m**

```
- (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions{
      self.window = [[UIWindow alloc] initWithFrame:[[UIScreen mainScreen]
bounds]];
      // Override point for customization after application launch.
      self.window.rootViewController = [[ExampleViewController alloc]
initWithNibName:@"ExampleViewController" bundle:nil];
      self.window.backgroundColor = [UIColor whiteColor];
      [self.window makeKeyAndVisible];
      return YES;
}
```

What this does: This will create a new instance of the ExampleViewController class, initializing it with a XIB file. Setting the rootViewController to this new viewController will automatically display it on the screen.

11. **ERROR:** There is now an error in the code. This is because the *ExampleViewController* class wasn't **imported** into the class that was using it.

12. Add the following code to the top of **AppDelegate.m**.

```
#import "ExampleViewController.h"
```

    What this does: This imports the class definition for *ExampleViewController* so the compiler knows what to do with it.

13. RUN THE APP. The screen should appear with the new button on the screen. Clicking on the button doesn't do anything!

NOTE: At this point, this application, that was created from an *Empty Application* template, is effectively the same as an application created from a *Single View Application* template.

14. Select the *ExampleViewController.xib* file and open the **Assistant Editor** (View-->Assistant Editor-->Show Assistant Editor). Double-check to ensure that the file opened in the Assistant Editor is *ExampleViewController.h*.

15. Right click on the button in the XIB file, and drag from **Touch Up Inside** over to the opened .h file. Drag to an empty line between the **@interface** and **@end** directives. Releasing the mouse will cause a pop-up to appear. Enter *myButtonClick* into the name and press **Connect**.

16. Close the Assitant Editor.

17. Open *ExampleViewController.m* and scroll down to the bottom to find the newly created *myButtonClick:* selector. Notice the grey circle in the margin beside this line. Selecting this will display the connection in the XIB, and clicking on that connection will open the XIB file.

    WHAT THIS DOES: The above steps created a link between the UIButton and the newly created selector in the class.

18. Add the following code to the newly created *myButtonClick* selector in *ExampleViewController.m*

```
-(void)myButtonClick:(id)sender{
        NSLog(@"The button was clicked:%@", sender);
}
```

19. **RUN THE APP.** This will display the message “The button was clicked” in the console whenever the button is clicked. You'll need to display the console within XCode (⇧⌘C)

Part 2

Objective: Add a UILabel to the screen.

20. With the *ExampleViewController.xib* selected, drag a UILabel from the Object Library onto the screen. (This is a similar process to adding the UIButton that was added in a previous step.)

21. Open the Assistant Editor (ensure the file it has opened is *ExampleViewController.h*). Right click on the label and select **New Referencing Outlet**. Drag that over to the code in the Assistant Editor below the **@interface** directive. Set the **name** of the outlet as *myLabel* and press Connect.

    What this does: This creates a public property called *myLabel*. Note that the same Referencing Outlet can be made directly in the *ExampleViewController.m* file, but that would be a private property.

22. Now let's talk to the button! Add the following code after the NSLog line in the *buttonClick:* selector.

```
NSLog(@"The button was clicked: %@", sender);
self.myLabel.text = @"Goodbye";
```

    What this does: This changes the text being displayed by the label when the button is clicked.

23. RUN THE APP. Now when you click on the button, the text in the label will change.