iPhone App Dev

Lesson 2

Practice

- Browse the App Store. Find some apps that impress you. Discuss why they are good.
- Do you have any problem that want to solve in a mobile phone? What app do you want to develop?

Summary

- Introducing Objective-C
- NSString and tracing log message
- Using UILabel
- Application Life Cycle
- Delegation
- Using UITextField
- Using UIButton
- Detecting Touches

Objective-C

Introducing Obj-C

- Obj-C is a super set of C
- We can use C stuff, such as primitive type

```
1 int a = 12;
2 int b = 100;
3 int c = a+b; // we have c = 112.
4 a=10;
5 int d = a+b; // we have d = 110.
```

Data is Dumb

- Data is not smart.
- Data does not have any ability.
- Data is manipulated and transformed and it is still dumb data.

For example, given a number variable 13. This data itself doesn't know the next integer.

Function is not smart

- Function take input data and calculate the output data.
- Function does not care about the data.
- Function has no relationship with data.
- Function and Data encourage Procedural Programming

Object is smart

- Object contains both data and function, which we call it method.
- We define behavior of object.
- Methods have strong relationship with data inside the object.
- We can think in higher level to tackle the problem.

Class and Instance

- Class is prototype of a kind of Object.
- Class is like a factory object. We build objects based on the factory setting.

Class and Instance

- Instance is the object built from the factory, Class.
- We cannot create object instance without a class.
- Each object instance has the same data and methods as defined in its Class.

Obj-C Class

- Objective-C Class contains both Interface and Implementation.
- Interface is usually named .h file
- Implementation is usually named .m file
- Obj-C++ implementation is named .mm

Interface

- An example of Obj-C interface

```
1  #import <UIKit/UIKit.h>
2
3
4
5  @interface MainController : NSObject
6
7  - (void)touchesBegan:(NSSet *)touches withEvent:(UIEvent *)event;
8
9  - (void)touchesMoved:(NSSet *)touches withEvent:(UIEvent *)event;
10
11 - (void)touchesEnded:(NSSet *)touches withEvent:(UIEvent *)event;
12
13 - (void)sayHello;
14
15 @end
```

Example

```
MainViewController *controller = [[MainViewController alloc] init];
[controller sayHello];
```

Message Passing

```
1 @interface Person : NSObject
2 - (void) sayMessage:(NSString*)msg to:(NSString*)target;
3 @end
4
5
6 Person *person = [[Person alloc] init];
7 [person sayMessage:@"hello" to:@"Thomas"];
```

NSString

NSLog - Log Message

- NSLog takes a NSString format argument.
  So we can log a text in console with

```
1 NSLog(@"hello world");
2 // output "hello world"
3
4 NSLog(@"this is a number: %d", 123);
5 // output "this is a number: 123"
6
7 NSLog(@"%d + %d = %d", 1, 2, 3);
8 // output "1 + 2 = 3"
```

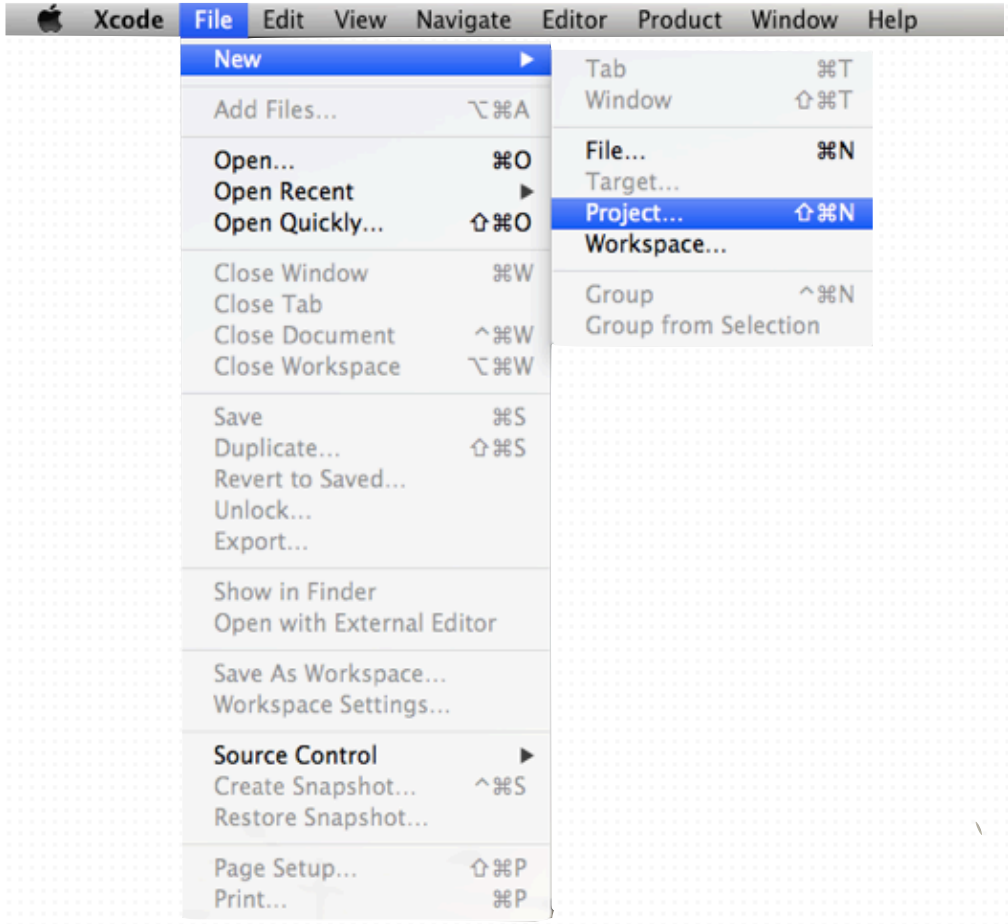
NSLog - Log Message

- `%d` is decimal number.
- `%f` is floating point number
- `%@` is a string description of an object

- `%02d` means decimal with at least 2 digits.
- `%.2f` means floating point with 2 decimal places.

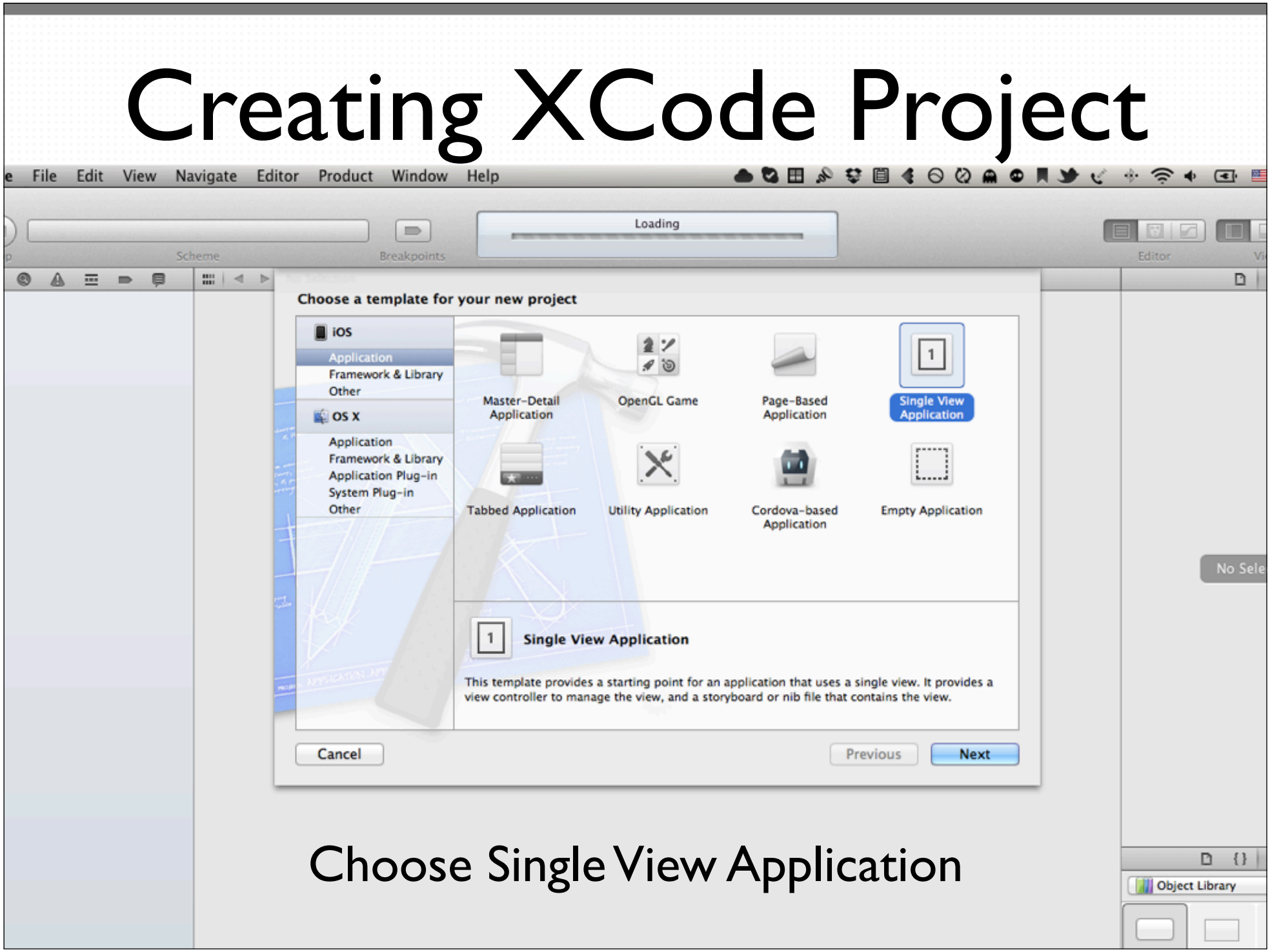
Creating XCode Project

Creating XCode Project



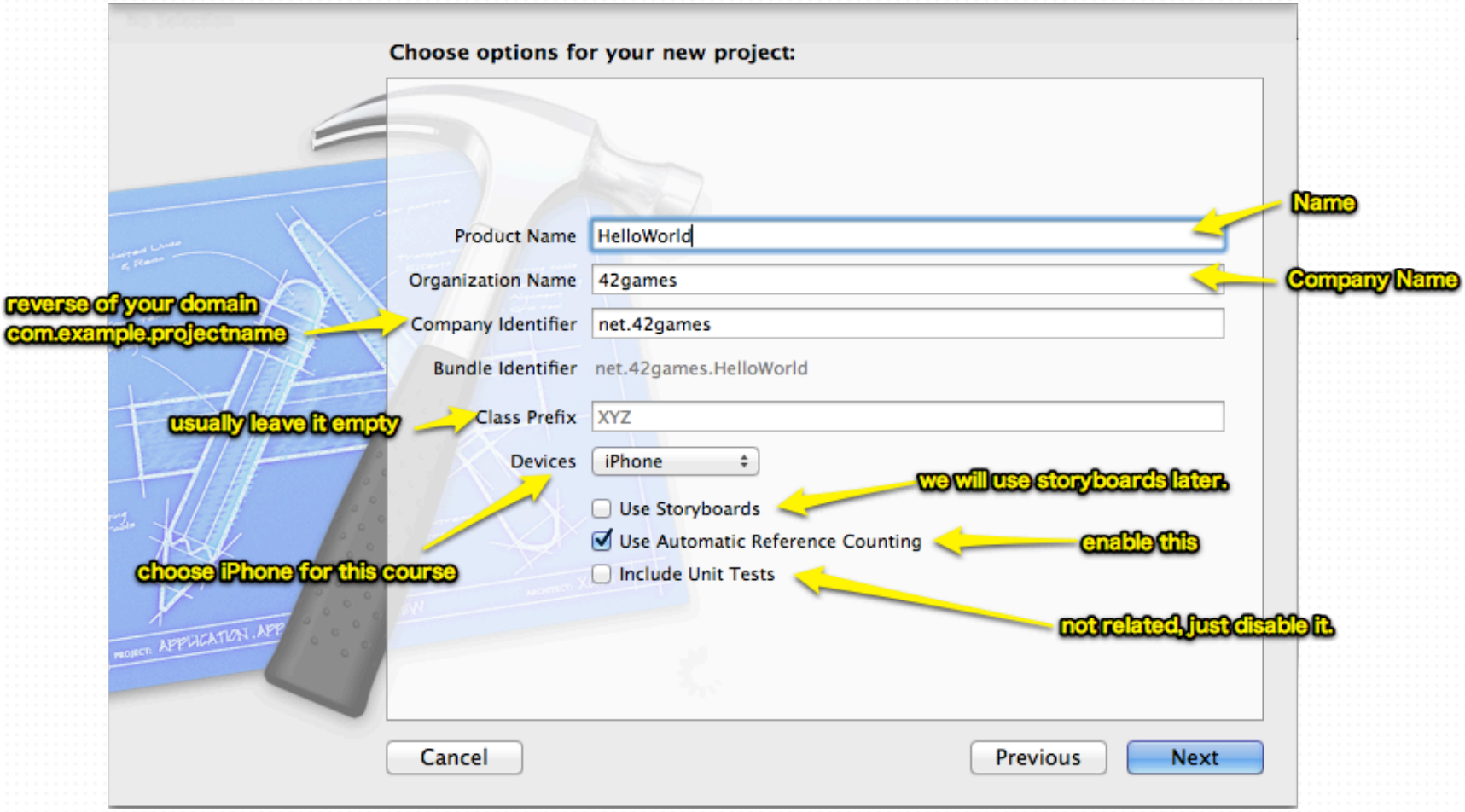
File | New | Project

Creating XCode Project



Choose Single View Application

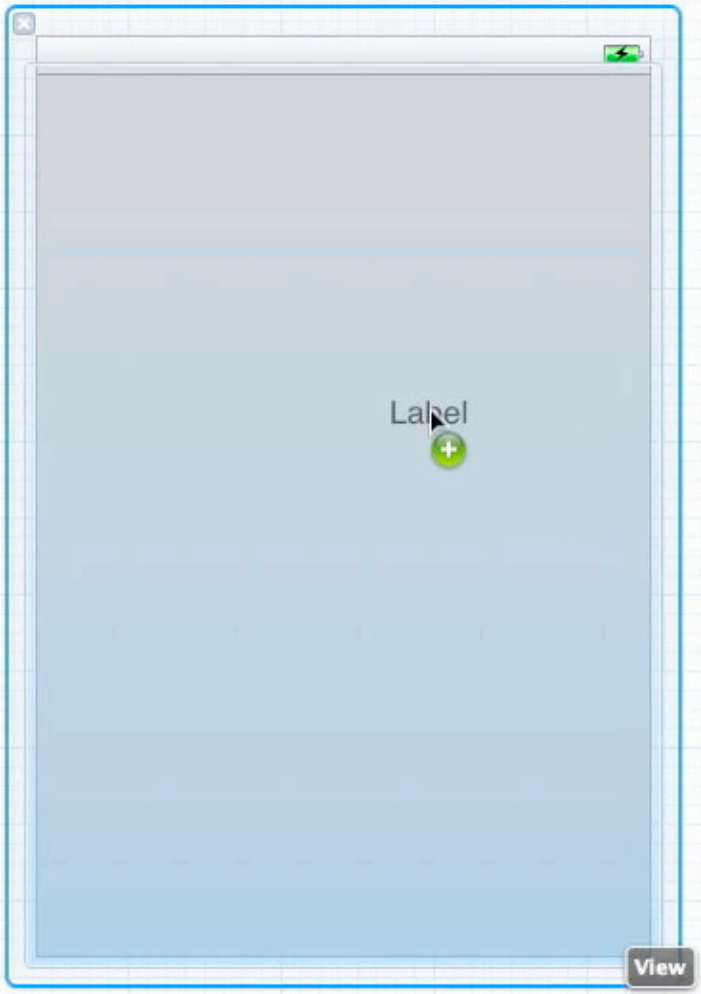
Creating XCode Project



Choose Single View Application

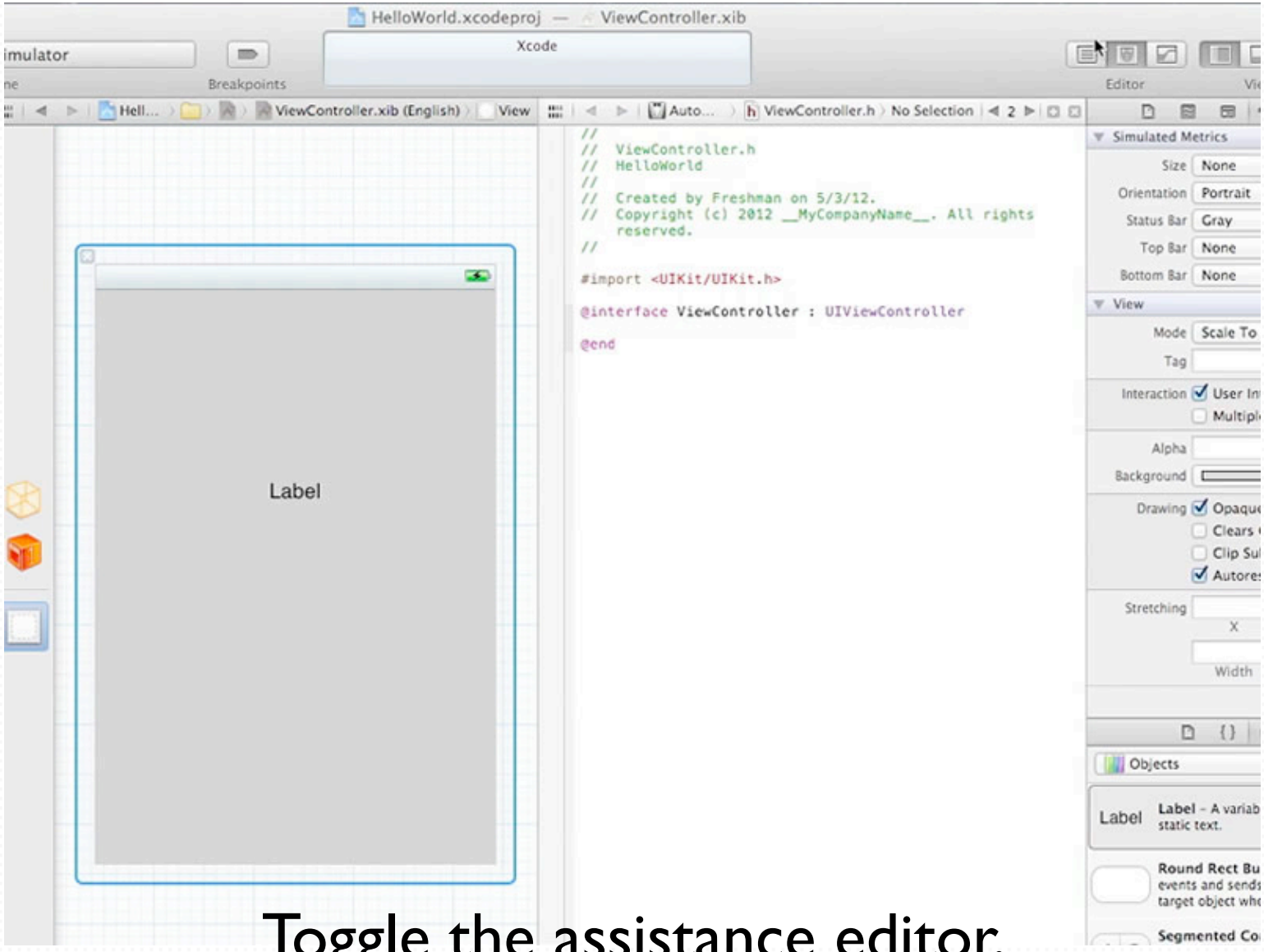
Using UILabel

Using UILabel



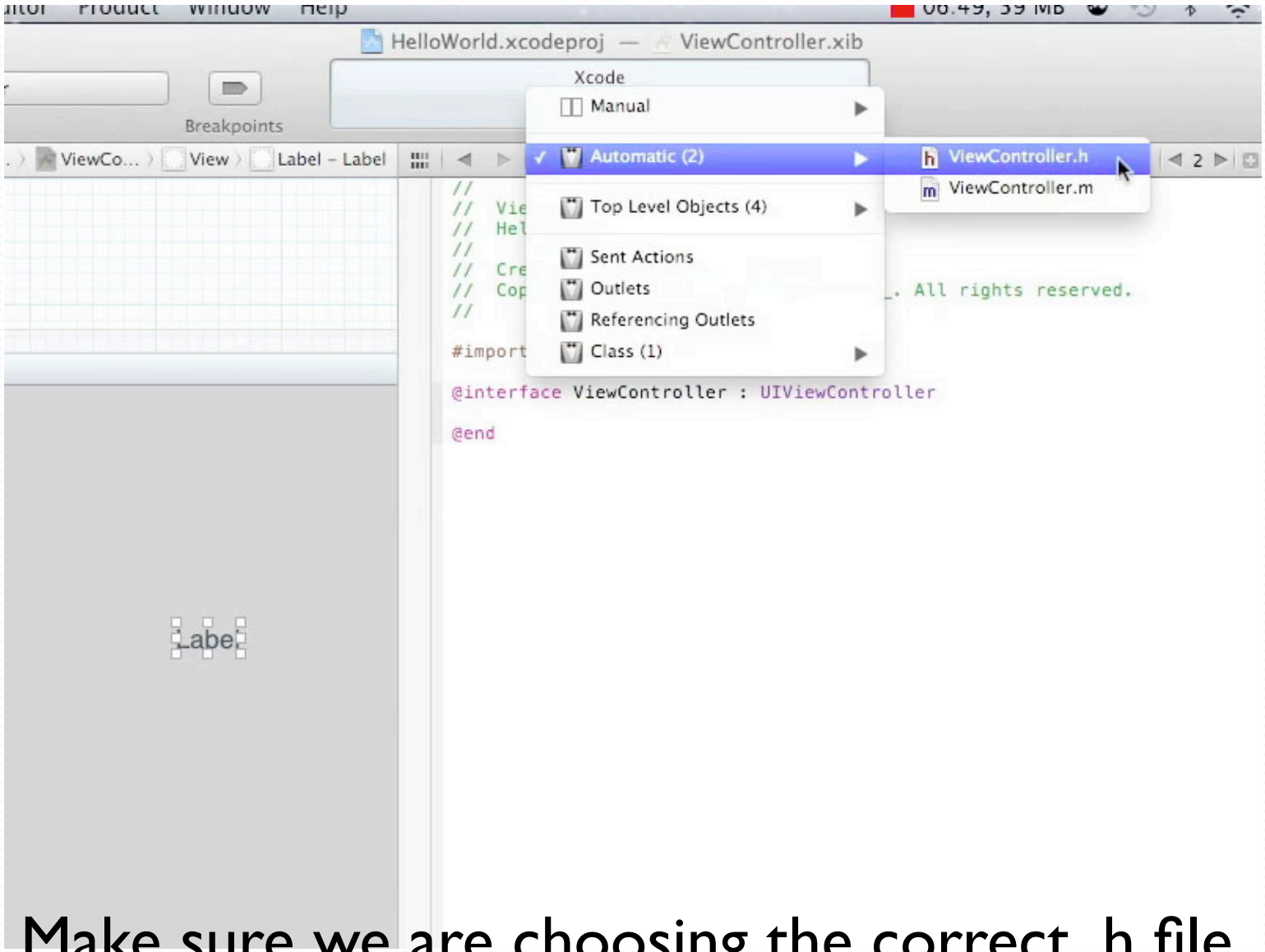
Dragging a Label into the view.

Using UILabel



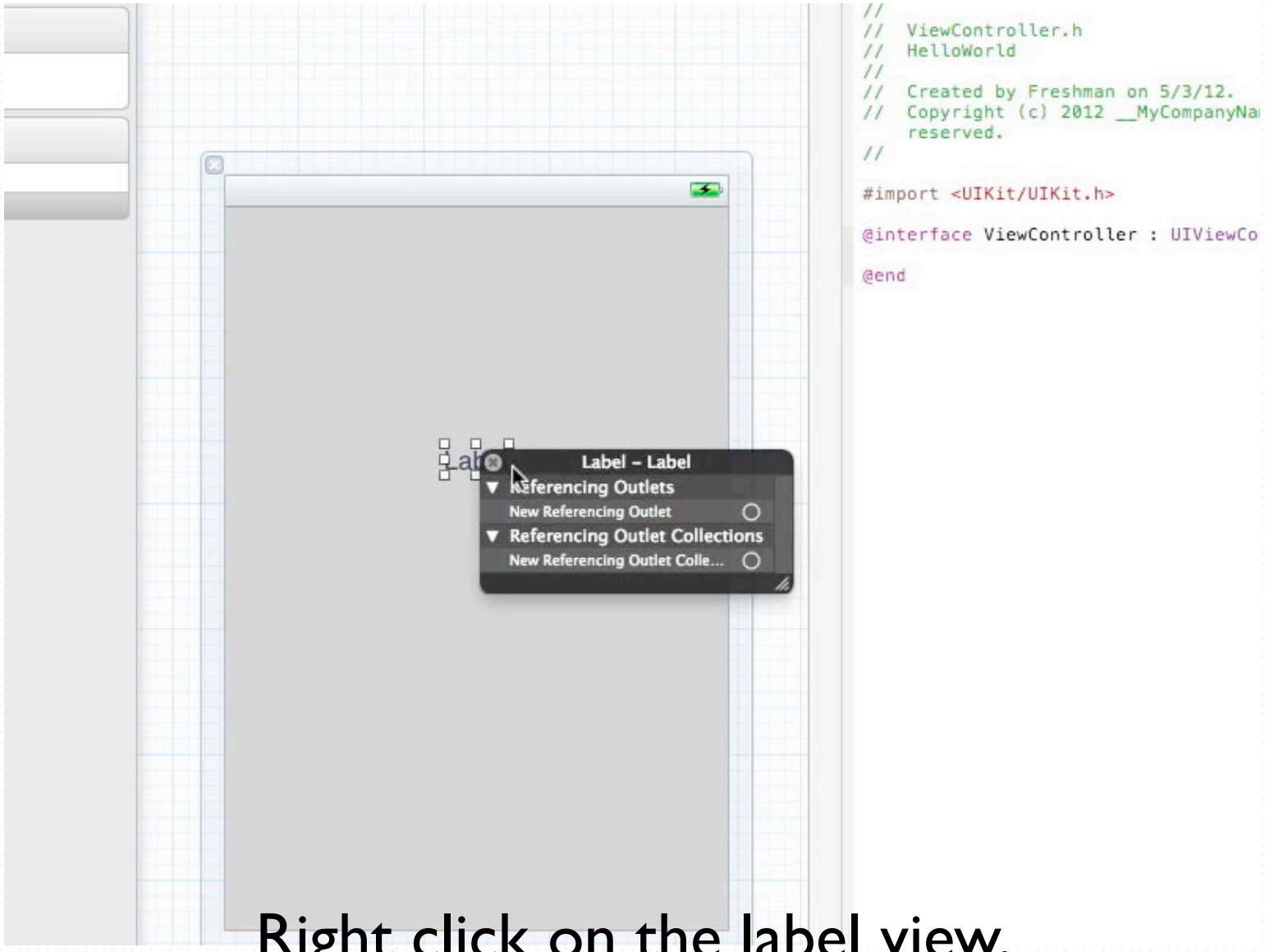
Toggle the assistance editor.

Using UILabel



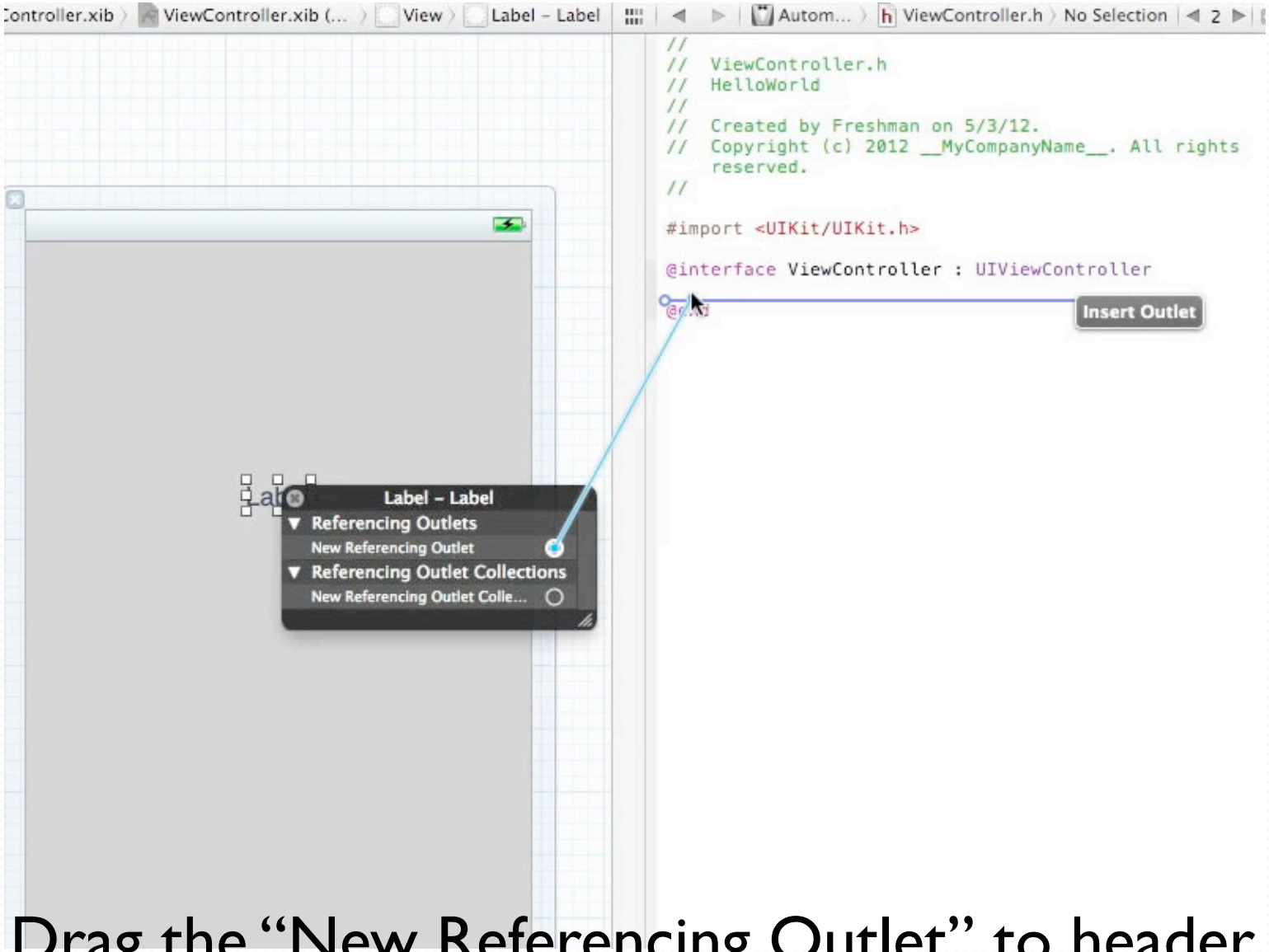
Make sure we are choosing the correct .h file.

Using UILabel



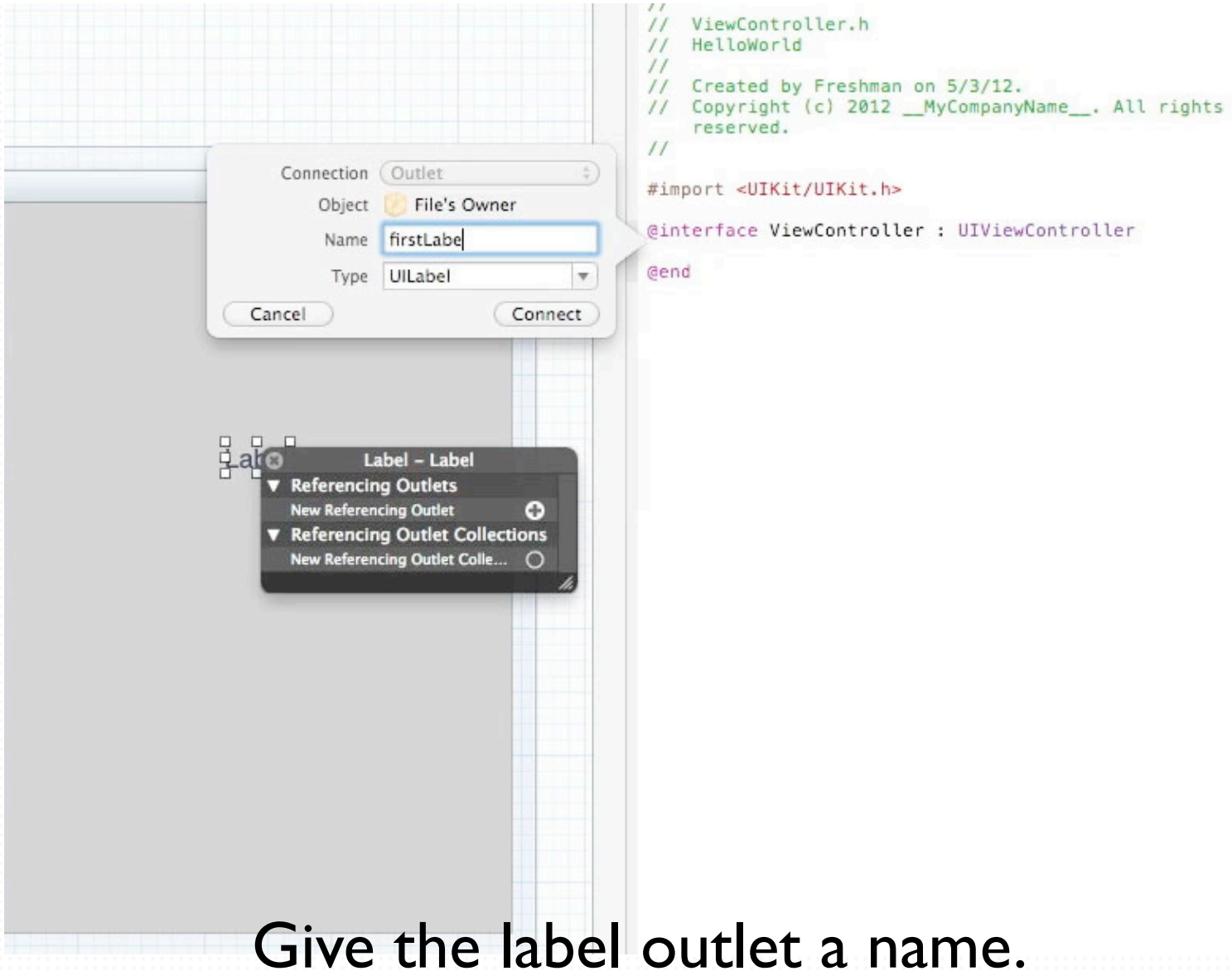
Right click on the label view.

Using UILabel



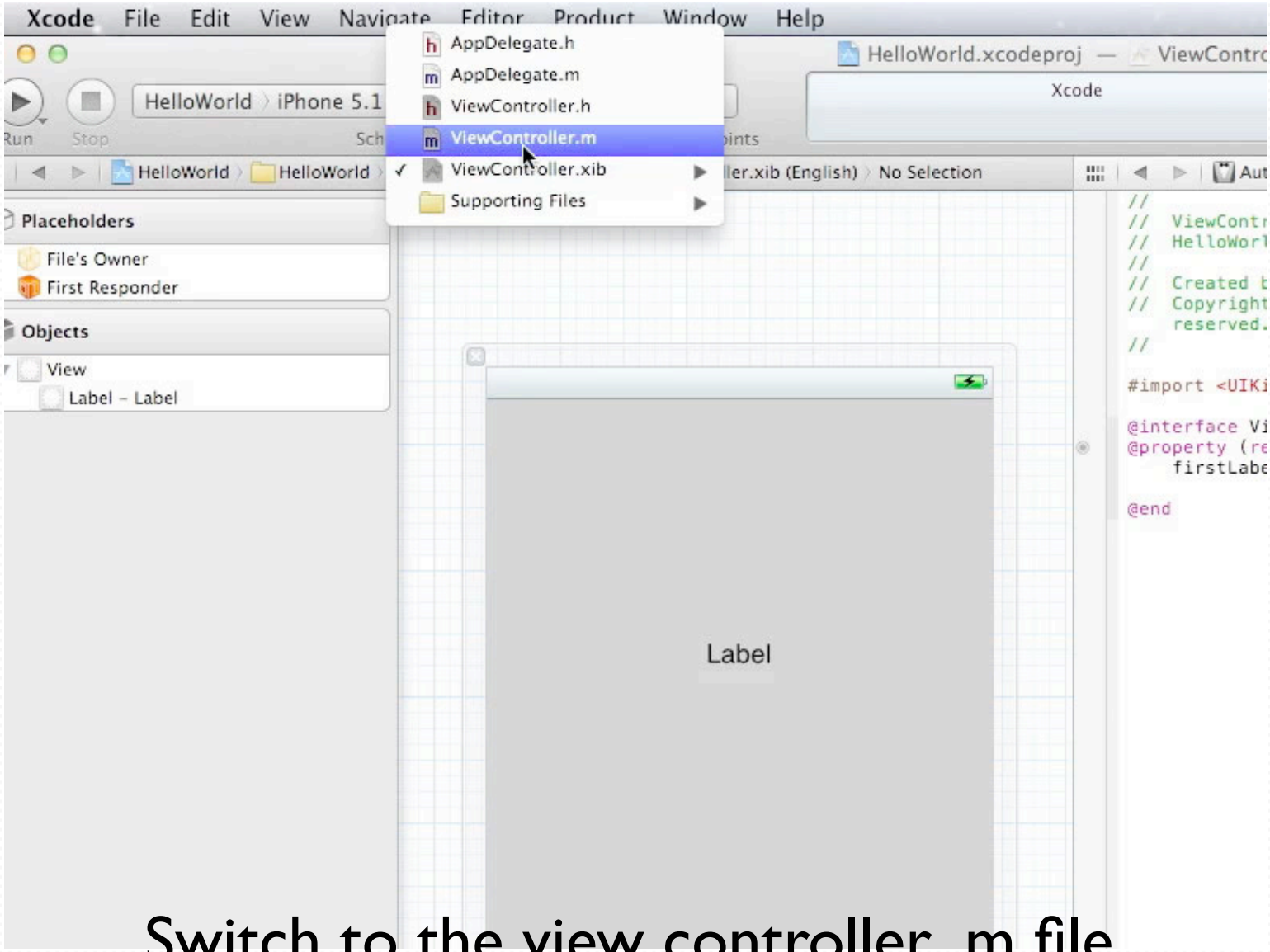
Drag the "New Referencing Outlet" to header.

Using UILabel



Give the label outlet a name.

Using UILabel



Switch to the view controller .m file.

Using UILabel

Setting text label

```
@implementation ViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
    // Do any additional setup after loading the view, typically from a nib.

    self.firstLabel.text = @"Hello";

}
```

Add the label text changing code in viewDidLoad delegate method

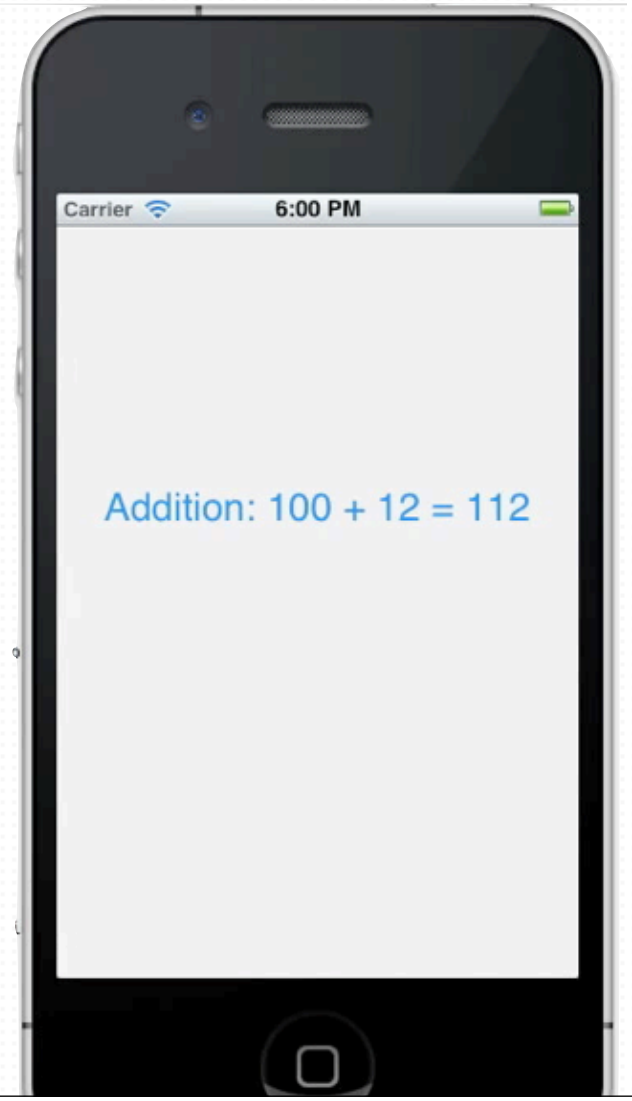
Using UILabel

Using UILabel

- Setting text label with string formatter

```
1 NSString *caption = @"Addition";
2
3 int a = 100;
4 int b = 12;
5 int c = a + b;
6
7 self.firstLabel.text = [NSString
stringWithFormat:@"%@: %d + %d = %d", caption, a, b, c];
```

Using UILabel



Using UILabel

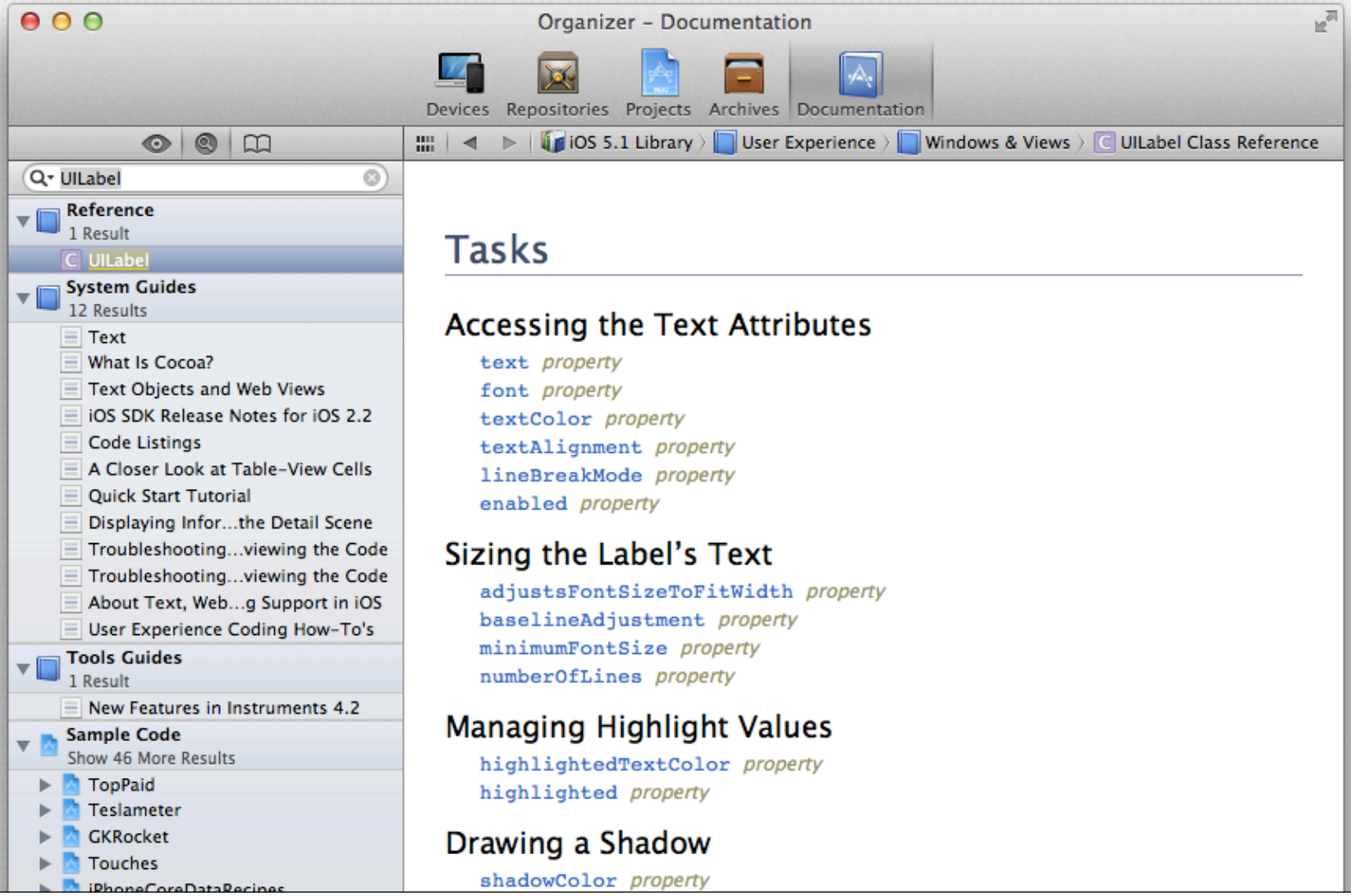
- Setting text color

```
1 self.firstLabel.textColor = [UIColor redColor];
```

- Setting font size

```
1 self.firstLabel.font = [UIFont systemFontOfSize:24.0f];
```

Using UILabel



How the label links with code?

- IBOutlet
- The xib remember which outlet variable the view is pointing to.
- The code change the variable and inform the xib to update the view.

Application Life Cycle

Application Life Cycle

- main.m > UIApplication > AppDelegate Class
- AppDelegate is our app's entry point.

<UIApplicationDelegate>

```
- (void)applicationDidFinishLaunching:(UIApplication *)application;
```

- Entry point of the application.
- Execute every time when the app launched.

<UIApplicationDelegate>

```
- (void)applicationDidBecomeActive:(UIApplication *)application;
```

- Execute every time when the app is open from background to foreground. (Switch from other app)

<UIApplicationDelegate>

- (void)applicationWillResignActive:(UIApplication *)application;

- Execute when the app is moved from active state into inactive state.
- Including when the multi-task bar appear on bottom.
- Good point to pause any running logic, such as game loop.

<UIApplicationDelegate>

- (void)applicationDidEnterBackground:(UIApplication *)application;

- Execute when the user switch to home screen or other application.
- The app entered background.
- Good point to save all user data into persistent storage.

<UIApplicationDelegate>

```
- (void)applicationWillTerminate:(UIApplication *)application;
```

- Execute when the application is going to be terminated.
- Usually not much we can do here. All data should be saved when the app entered background.

<UIApplicationDelegate>

- (BOOL)application:openURL:sourceApplication:annotation:

- Execute when the app is launched by calling the app URL scheme.
- App that is launched with

your-app-scheme://something

- This is usually done by executing an app from the other app.
- Facebook uses this approach for Single Sign On.

Delegates

Delegate Protocol

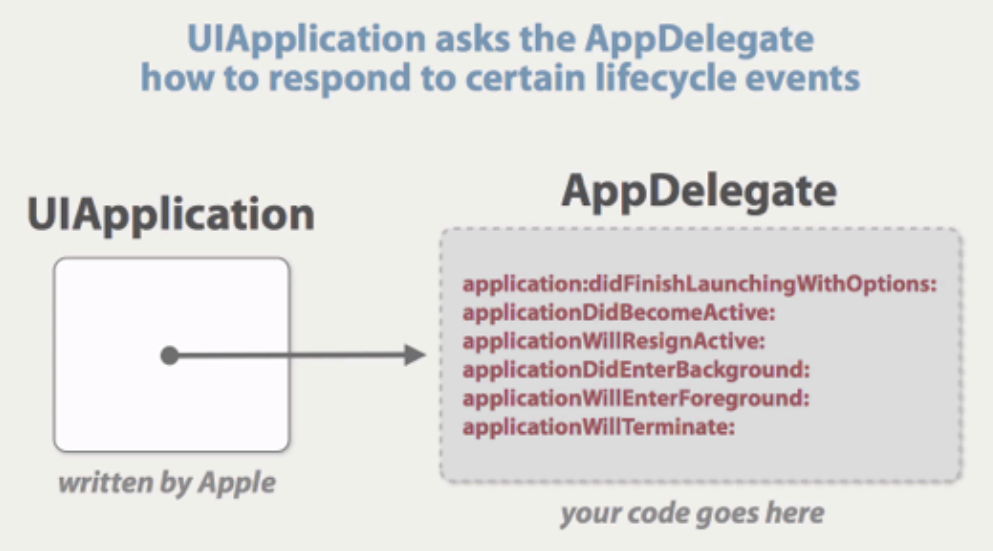
- Define a collection of methods.
- These methods act as communication bridge between two classes.
- The original class pass the task to the delegate to execute.
- We use delegate to customize behavior.

Delegate Protocol

- Application cycle are handled in UIApplication in OS.
- We provide delegate method to define behavior when UIApplication informs us.

```
1 @protocol UIApplicationDelegate<NSObject>
2
3 - (void)applicationDidFinishLaunching:(UIApplication *)application;
4
5 @end
```

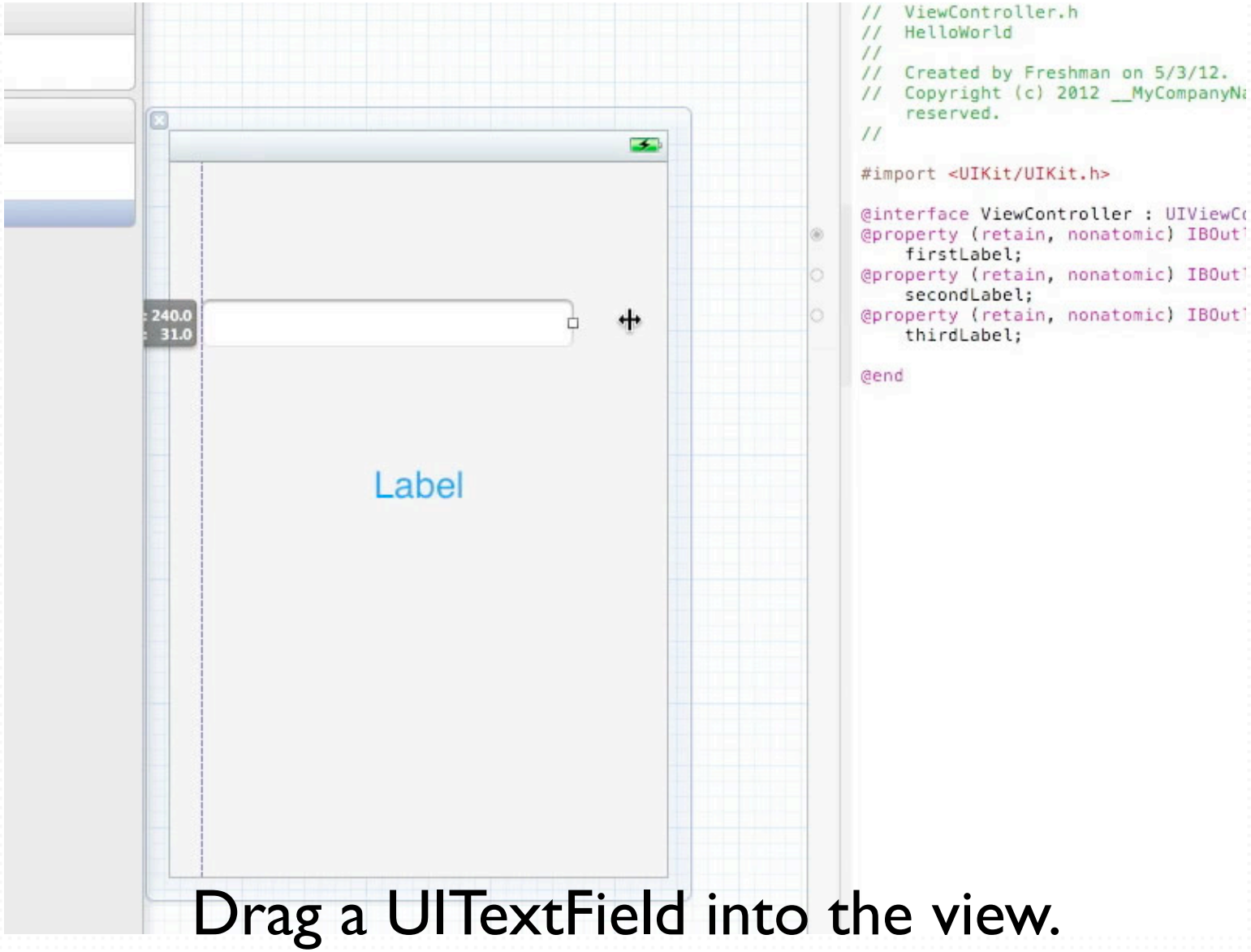
Delegate Protocol



Graph from http://pragmaticstudio.com/screencasts/rubymotion

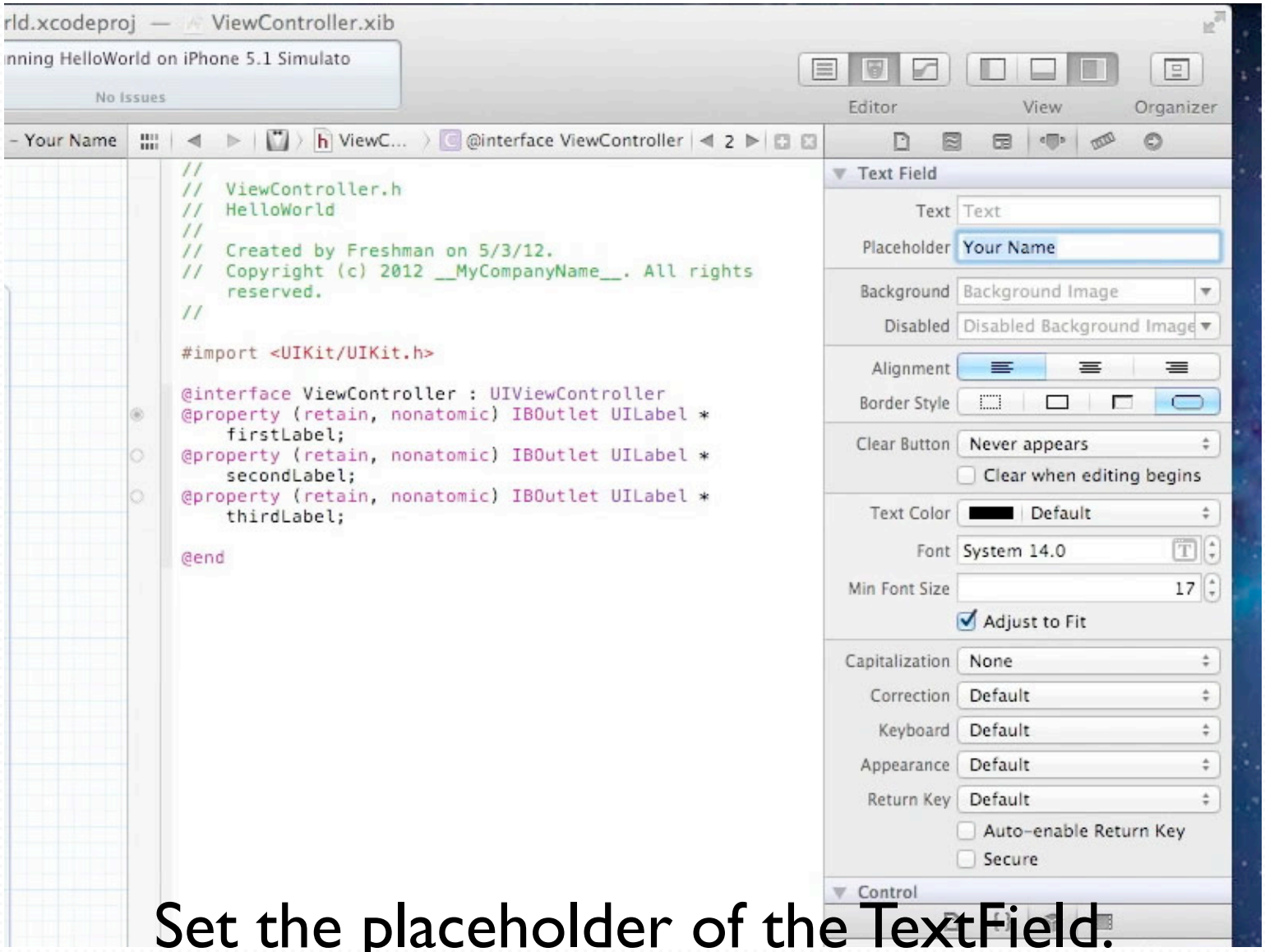
Using UITextField

Using UITextField



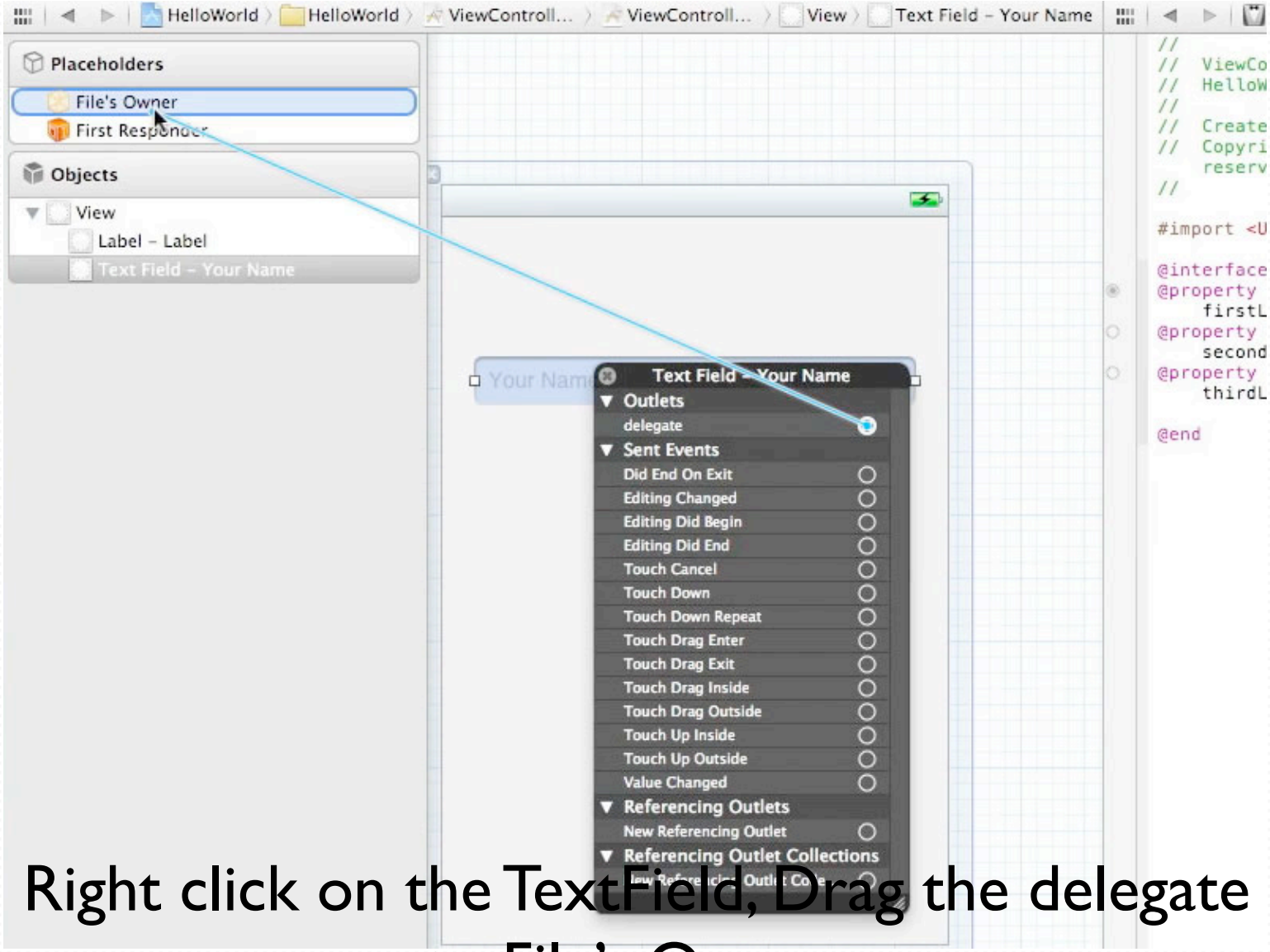
Drag a UITextField into the view.

Using UITextField



Set the placeholder of the TextField.

Using UITextField



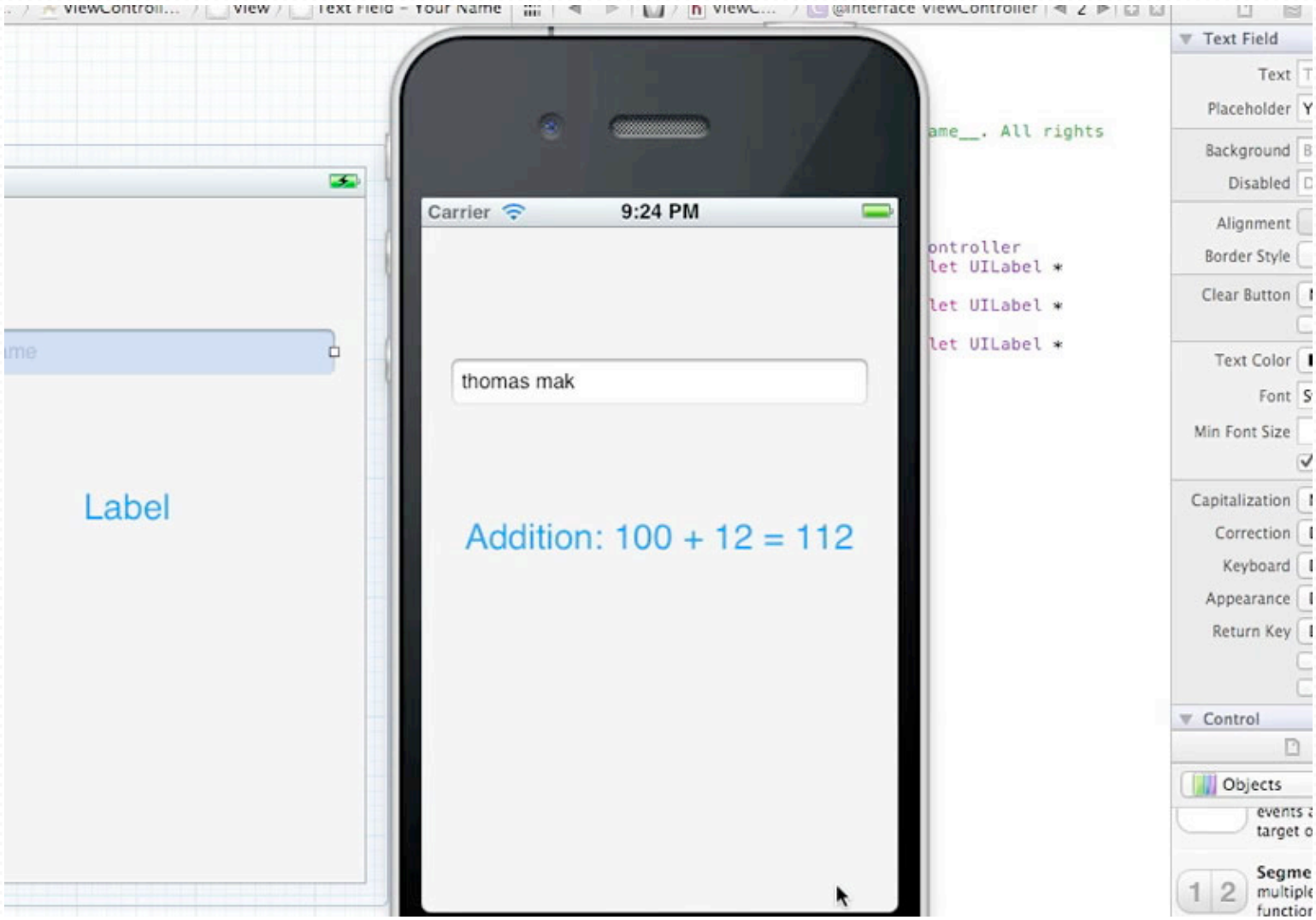
Right click on the TextField, Drag the delegate to File's Owner.

Using UITextField

```
1 - (BOOL)textFieldShouldReturn:(UITextField*)textField
2 {
3  [textField resignFirstResponder];
4  return NO;
5 }
```

Add the above delegate method to the view controller .m file

Using UITextField



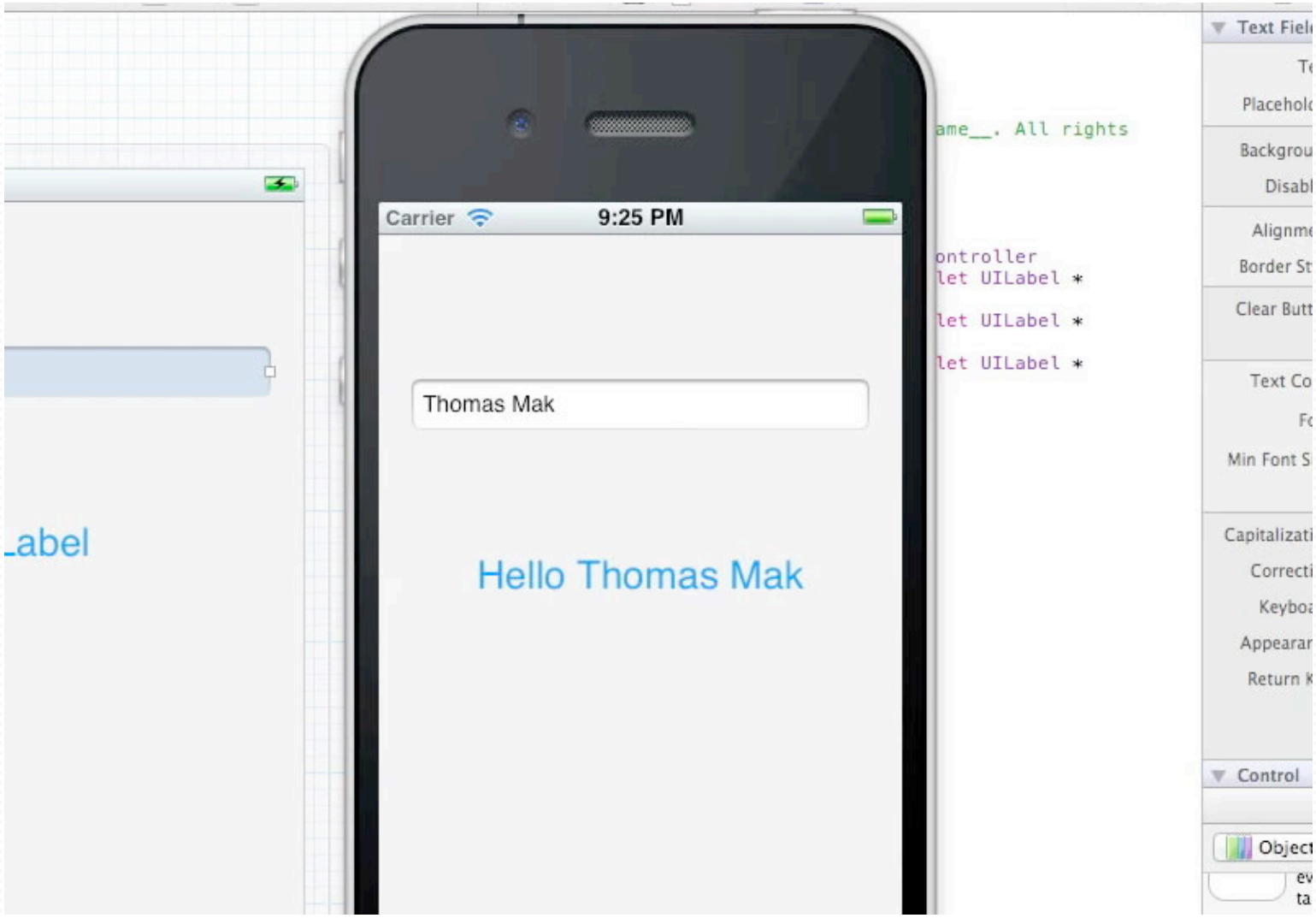
Let's test it in Simulator

Using UITextField

```
1 - (BOOL)textFieldShouldReturn:(UITextField*)textField
2 {
3  [textField resignFirstResponder];
4  self.firstlabel.text = [NSString
stringWithFormat:@"Hello %@", textField.text];
5  return NO;
6 }
```

Add the highlighted code into the delegate method.

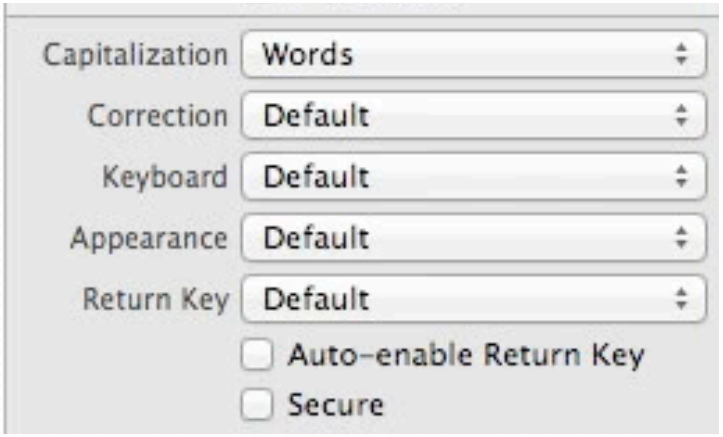
Using UITextField



Let's test it in Simulator

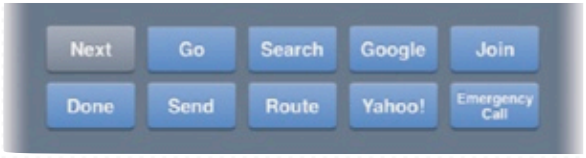
Using UITextField

- Capitalization
- Correction
- Keyboard
- Appearance
- Return Key



Using UITextField

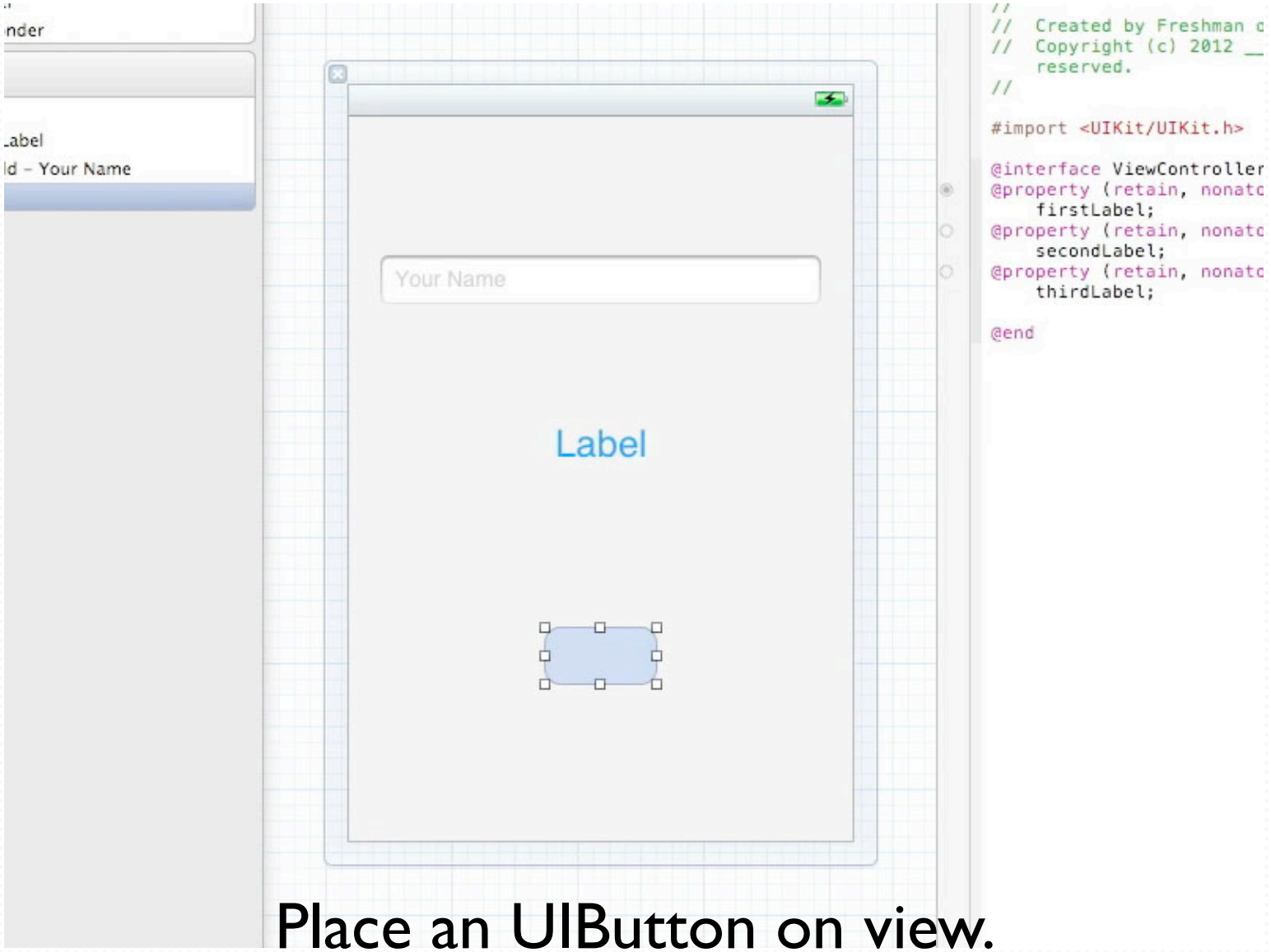
- Return Key



Graph from TapWorthy book.

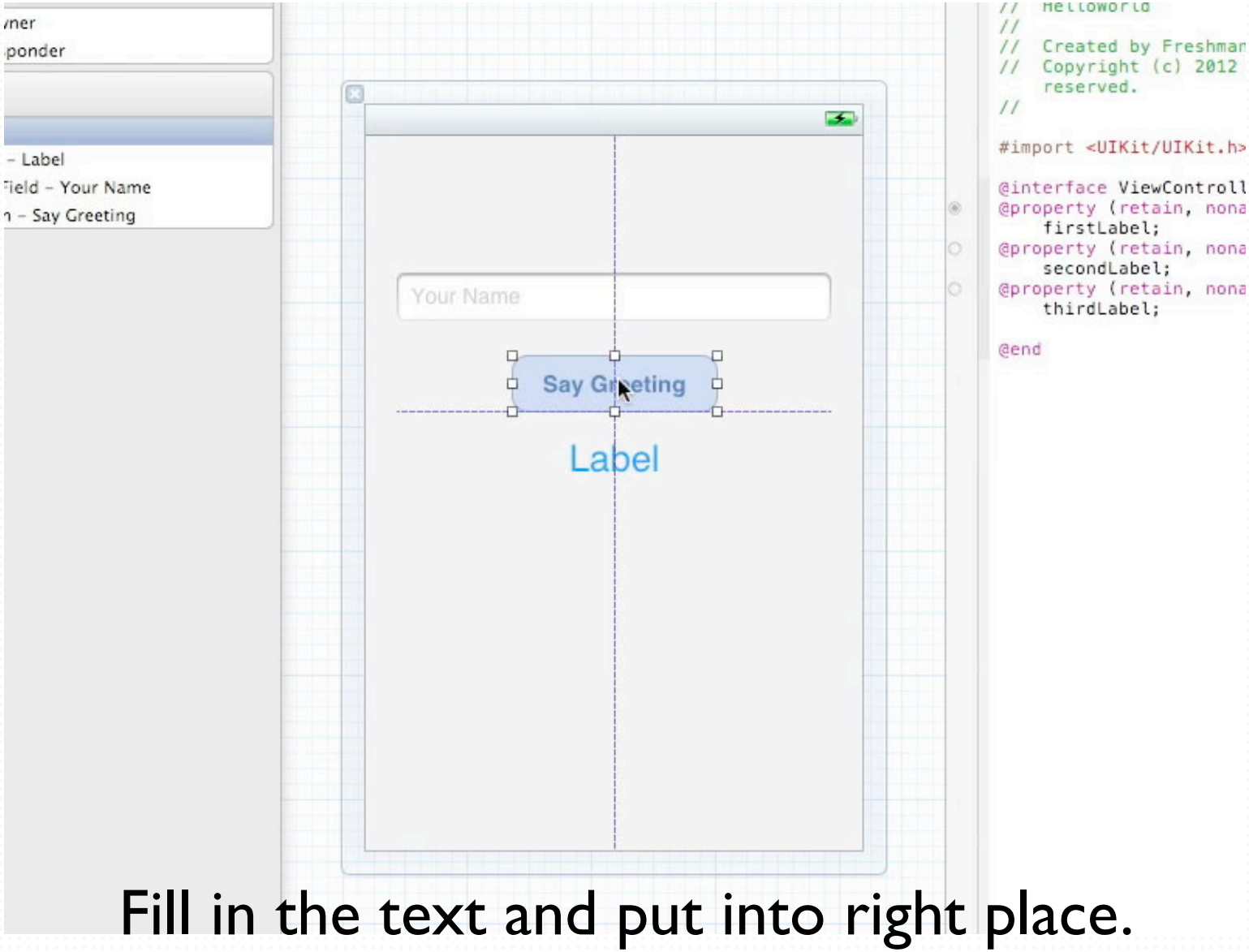
Using UIButton

Using UIButton



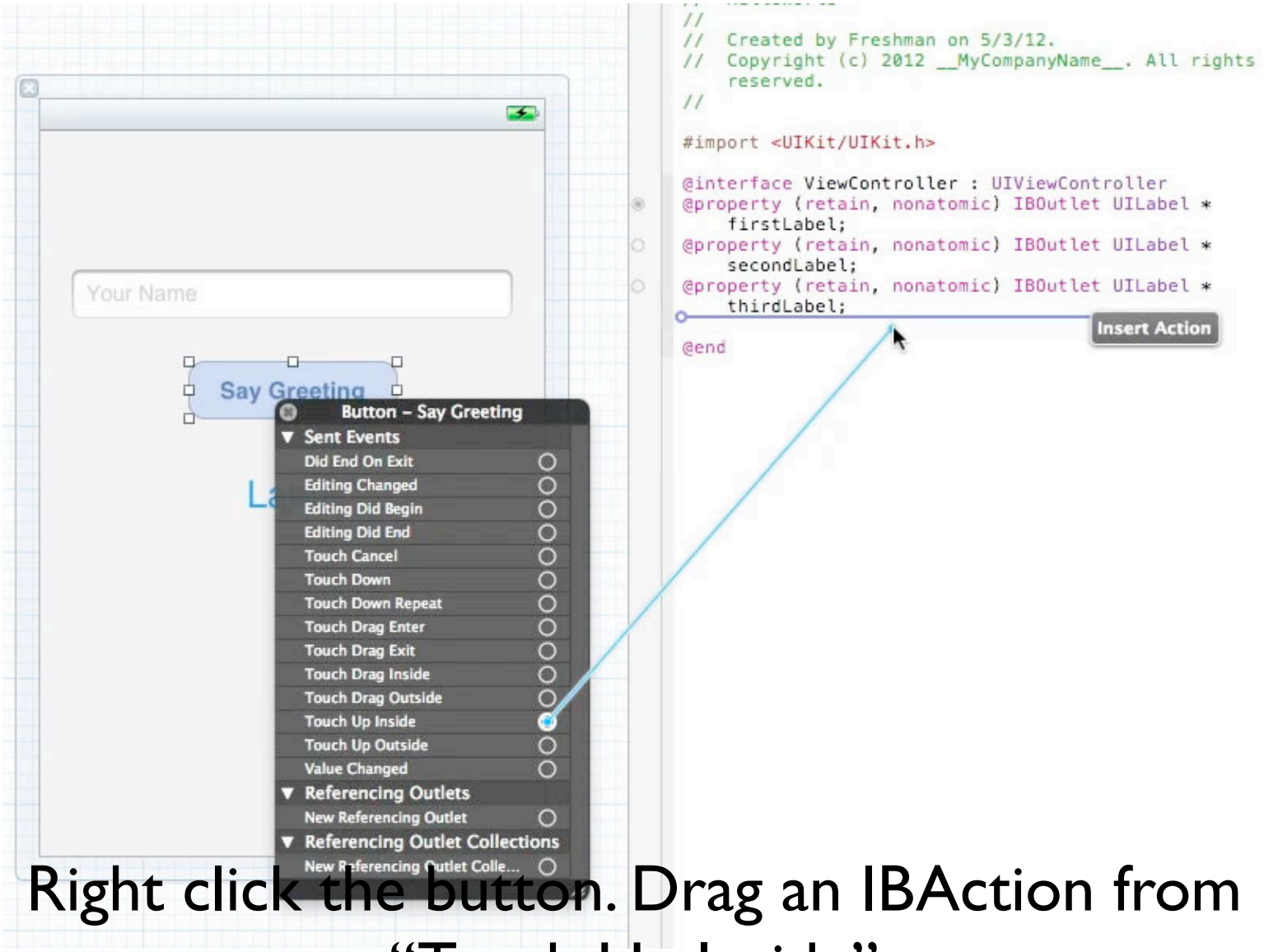
Place an UIButton on view.

Using UIButton



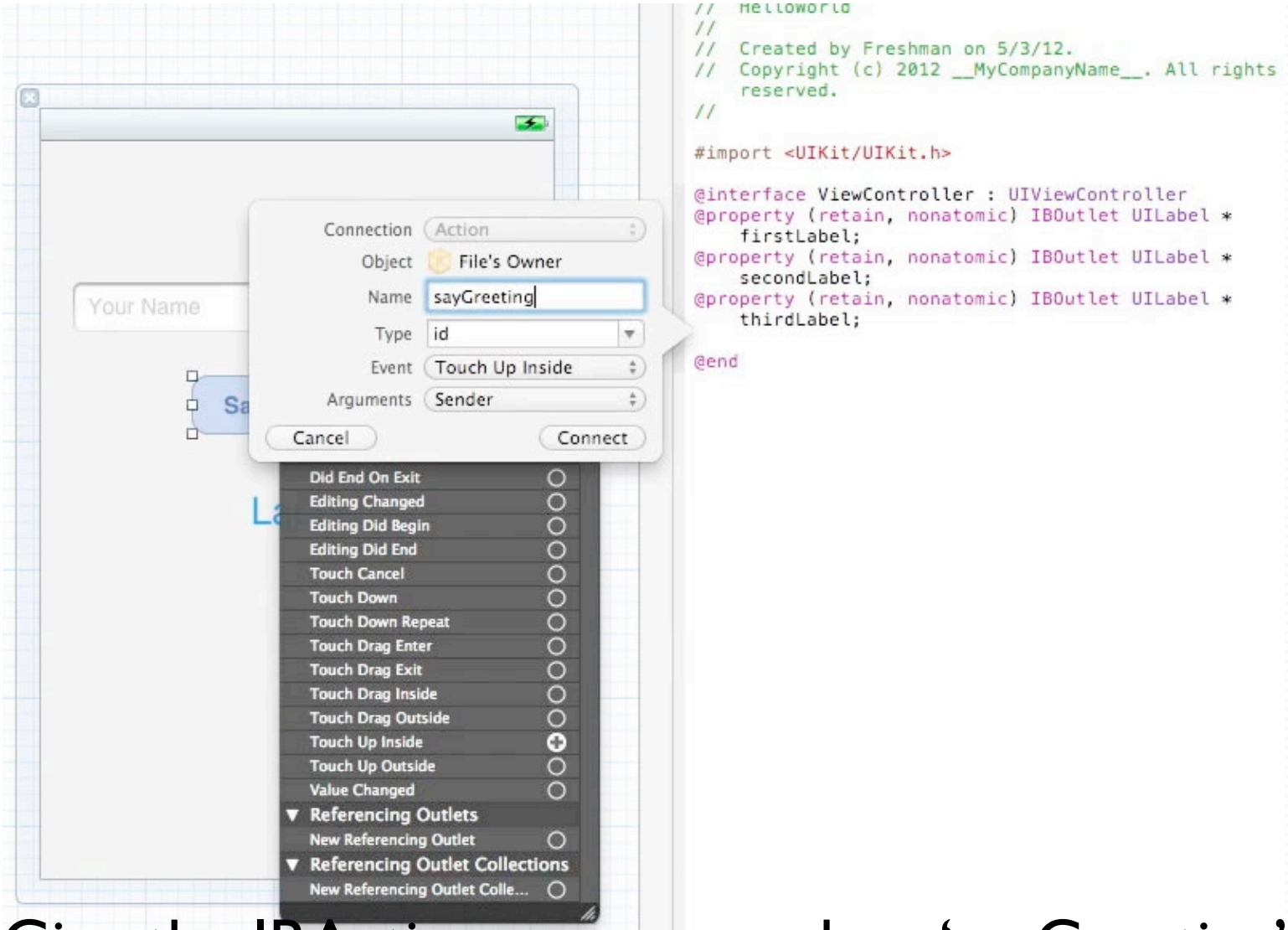
Fill in the text and put into right place.

Using UIButton



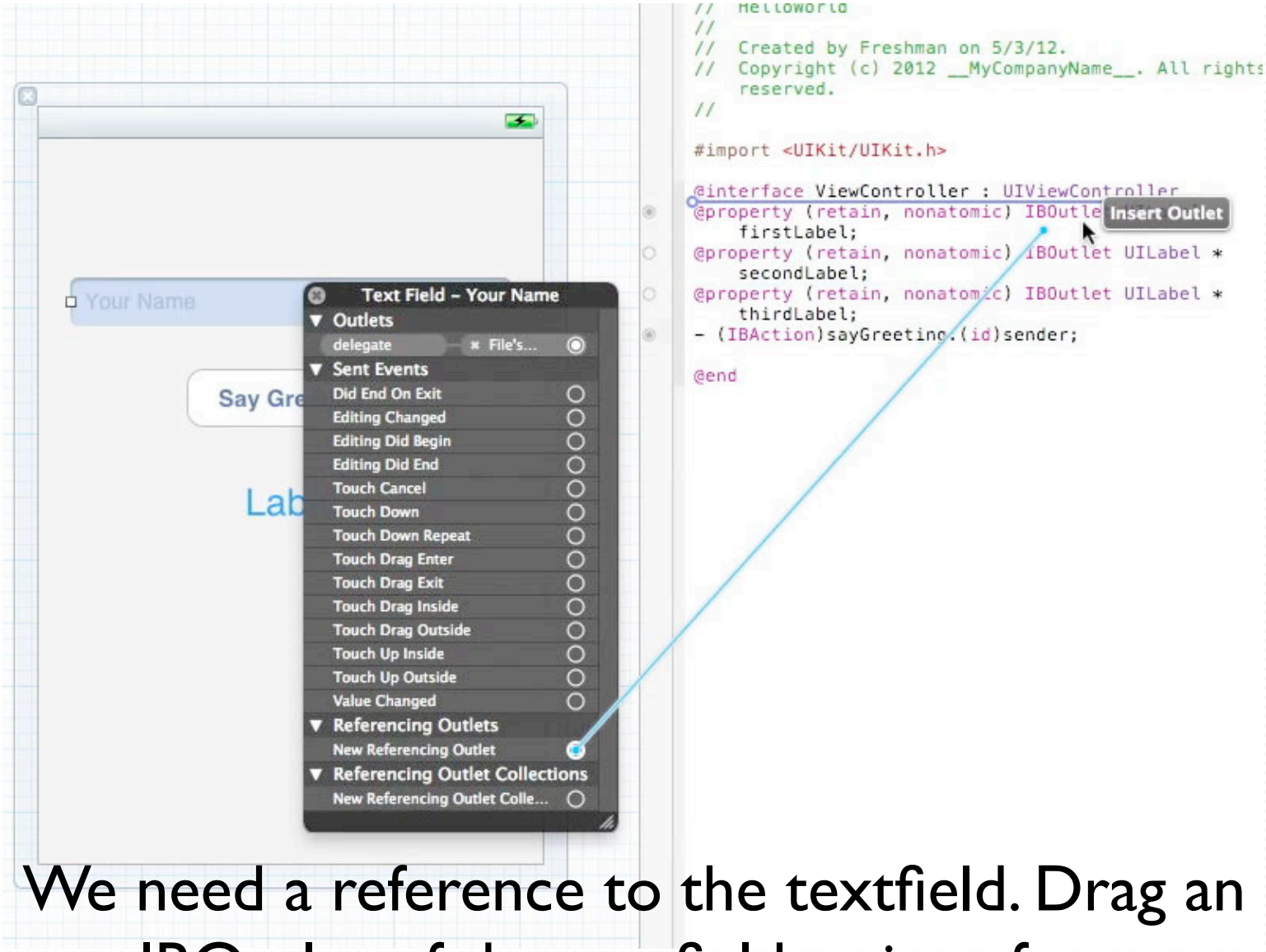
Right click the button. Drag an IBAction from "Touch Up Inside"

Using UIButton



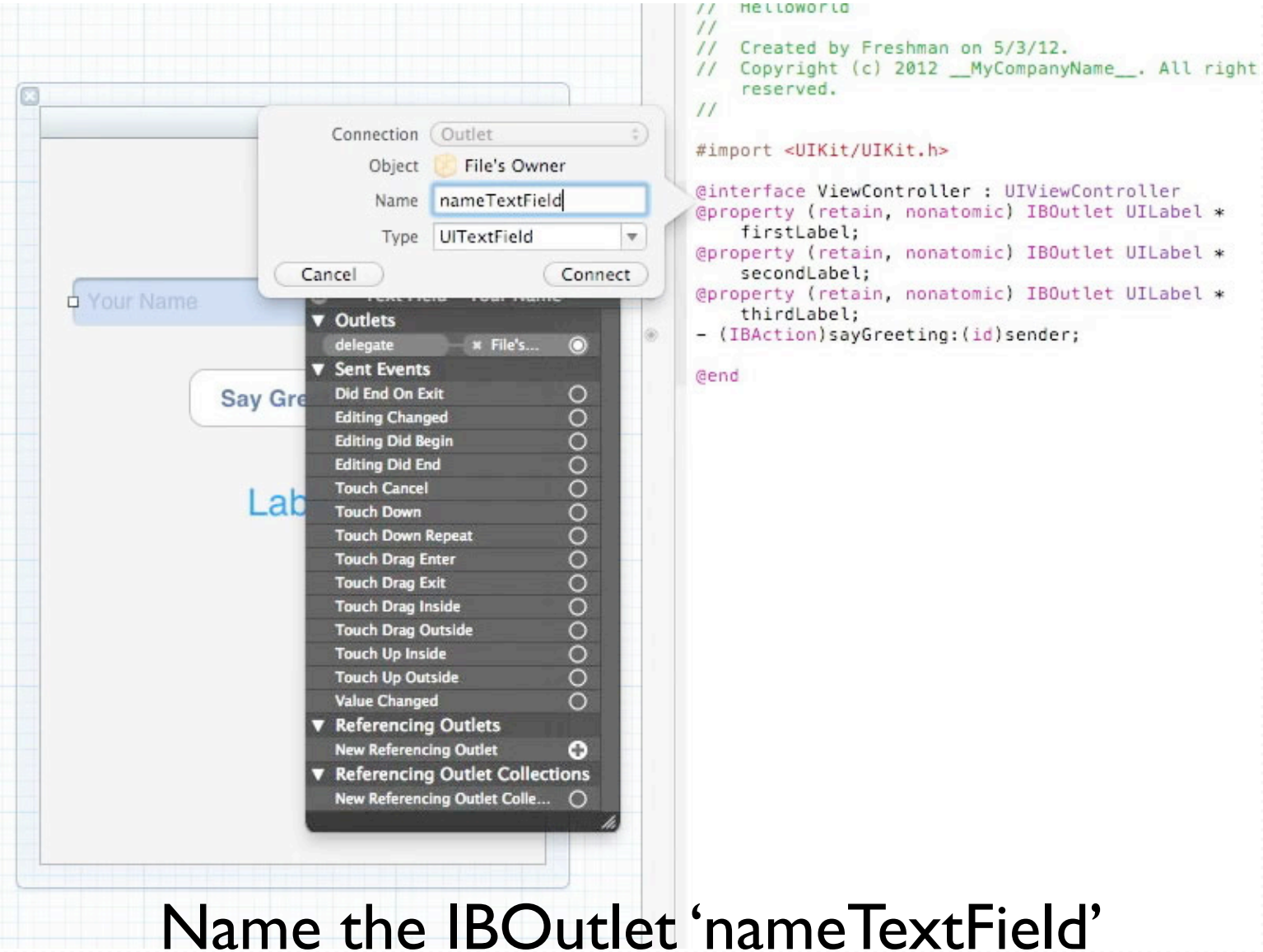
Give the IBAction a name, such as 'sayGreeting'

Using UIButton



We need a reference to the textfield. Drag an IBOutlet of the textfield to interface.

Using UIButton



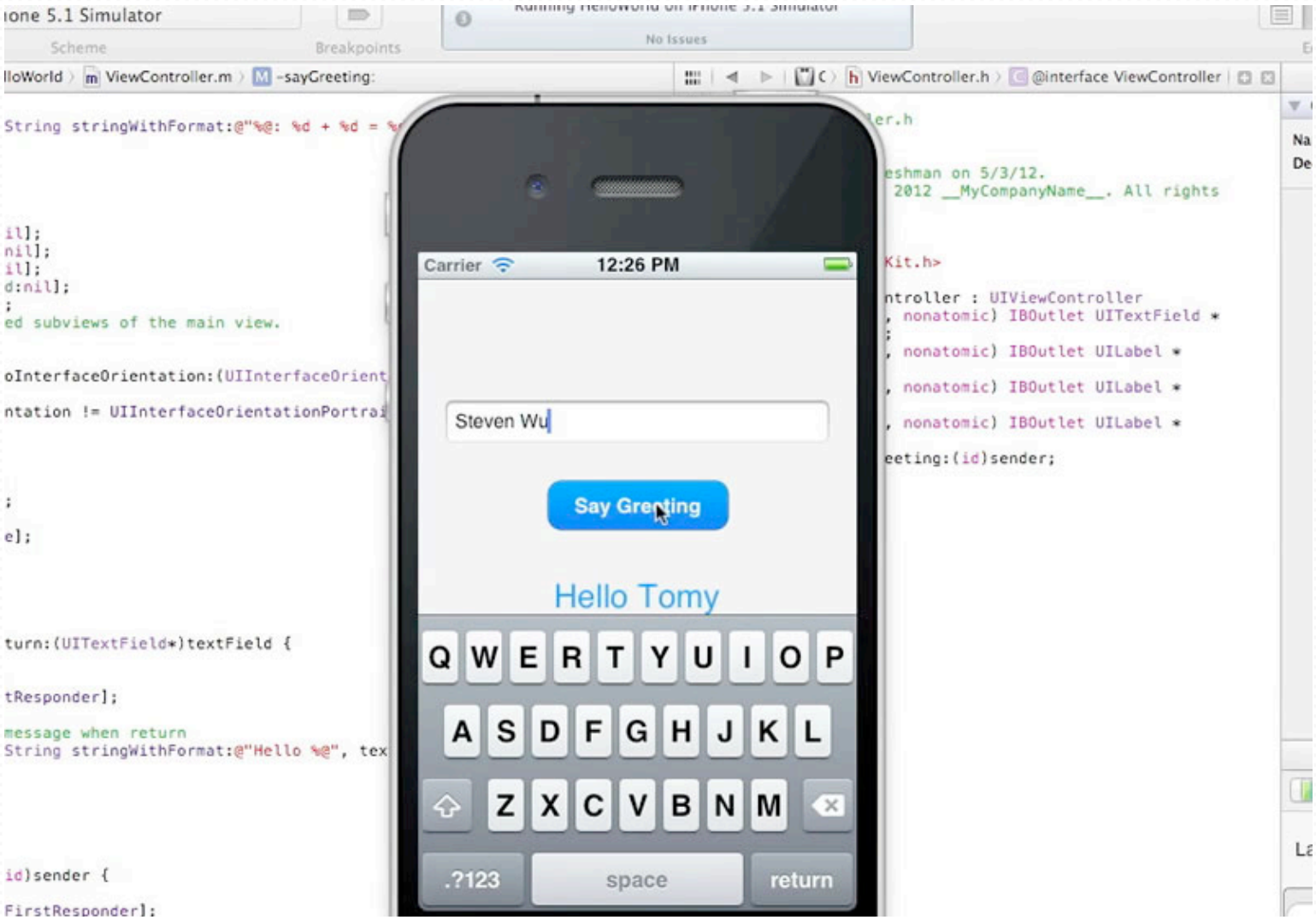
Name the IBOutlet ‘nameTextField’

Using UIButton

```
1 - (IBAction)sayGreeting:(id)sender {
2     // hide the keyboard.
3     [self.nameTextField resignFirstResponder];
4
5     // Show the greeting message when return
6     self.firstLabel.text = [NSString
stringWithFormat:@"Hello %@", self.nameTextField.text];
7
8 }
```

Implement the sayGreeting method.

Using UIButton



Test the app in simulator.

Extracting Method

- Now that we have similar behavior in both text field return delegate and button action method.
- We can extract them to have cleaner code.

Extracting Method

```
 1 - (void)showGreetingMessage {
 2     // hide the keyboard.
 3     [self.nameTextField resignFirstResponder];
 4
 5     // Show the greeting message when return
 6     self.firstLabel.text = [NSString
stringWithFormat:@"Hello %@", self.nameTextField.text];
 7 }
 8
 9
10 // TextField delegates
11 - (BOOL)textFieldShouldReturn:(UITextField*)textField {
12
13     [self showGreetingMessage];
14
15     return NO;
16 }
17
18 - (IBAction)sayGreeting:(id)sender {
19     [self showGreetingMessage];
20 }
```

Private and Public

- Methods are private when it is declared in .m
- The following interface code are declared in .m

```
1 @interface ViewController ()
2 - (void)showGreetingMessage;
3 @end
```

Private and Public

- Note that there is empty () after the ViewController interface.
- () is called Category in Obj-C.
- In simple words, this allows us to declare private methods and properties.

```
1 @interface ViewController ()
2 - (void)showGreetingMessage;
3 @end
```

Private and Public

- Methods that are declared in header are public.
- Public methods are visible by others.
- Others can invoke the public methods.

```
1 @interface ViewController
2 - (void)showGreetingMessage;
3 @end
```

Using Image in iOS

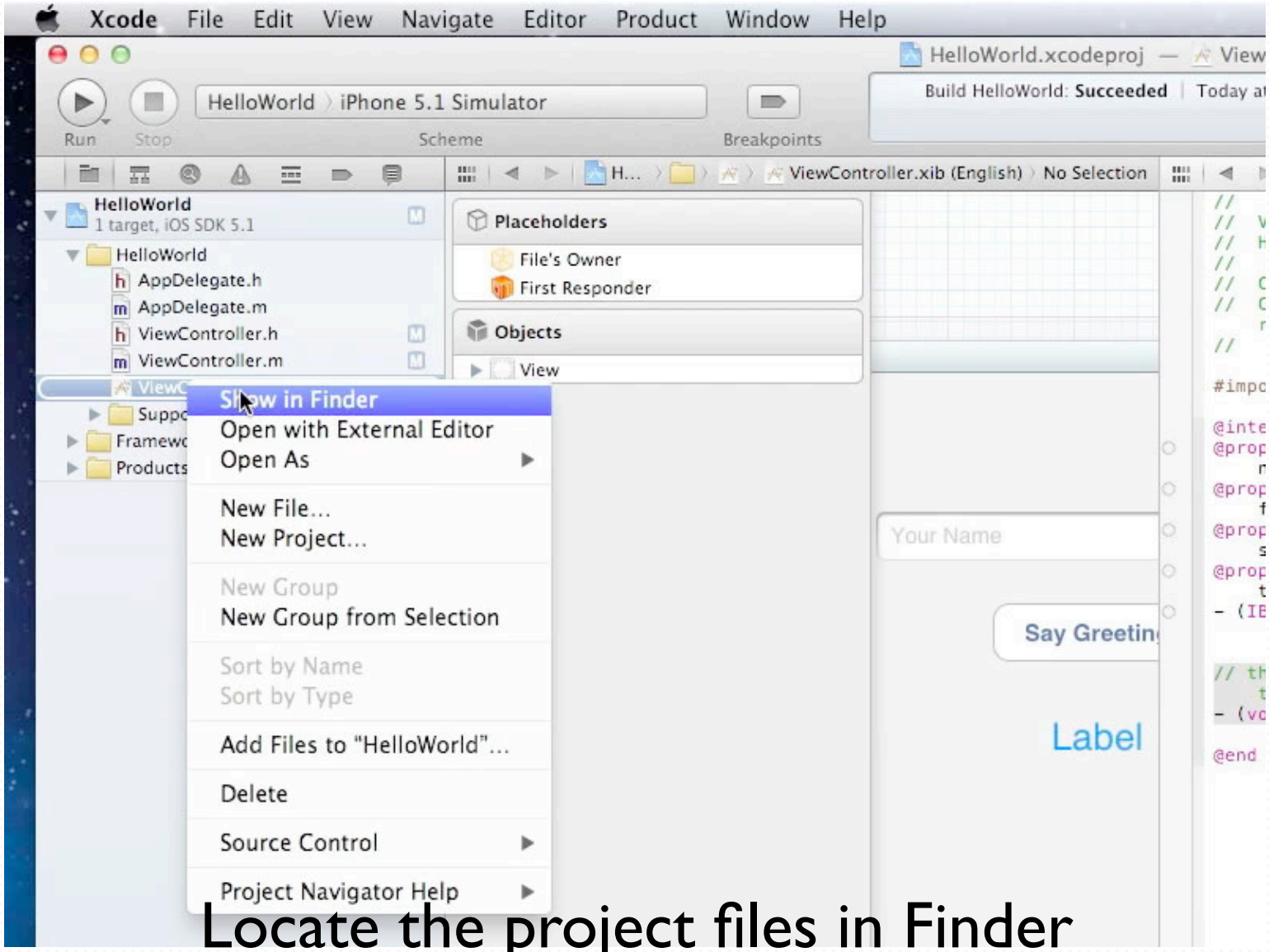
- filename.png
- filename@2x.png
- filename.jpg
- filename@2x.jpg

Using Image in iOS

button@2x.png
474 x 80

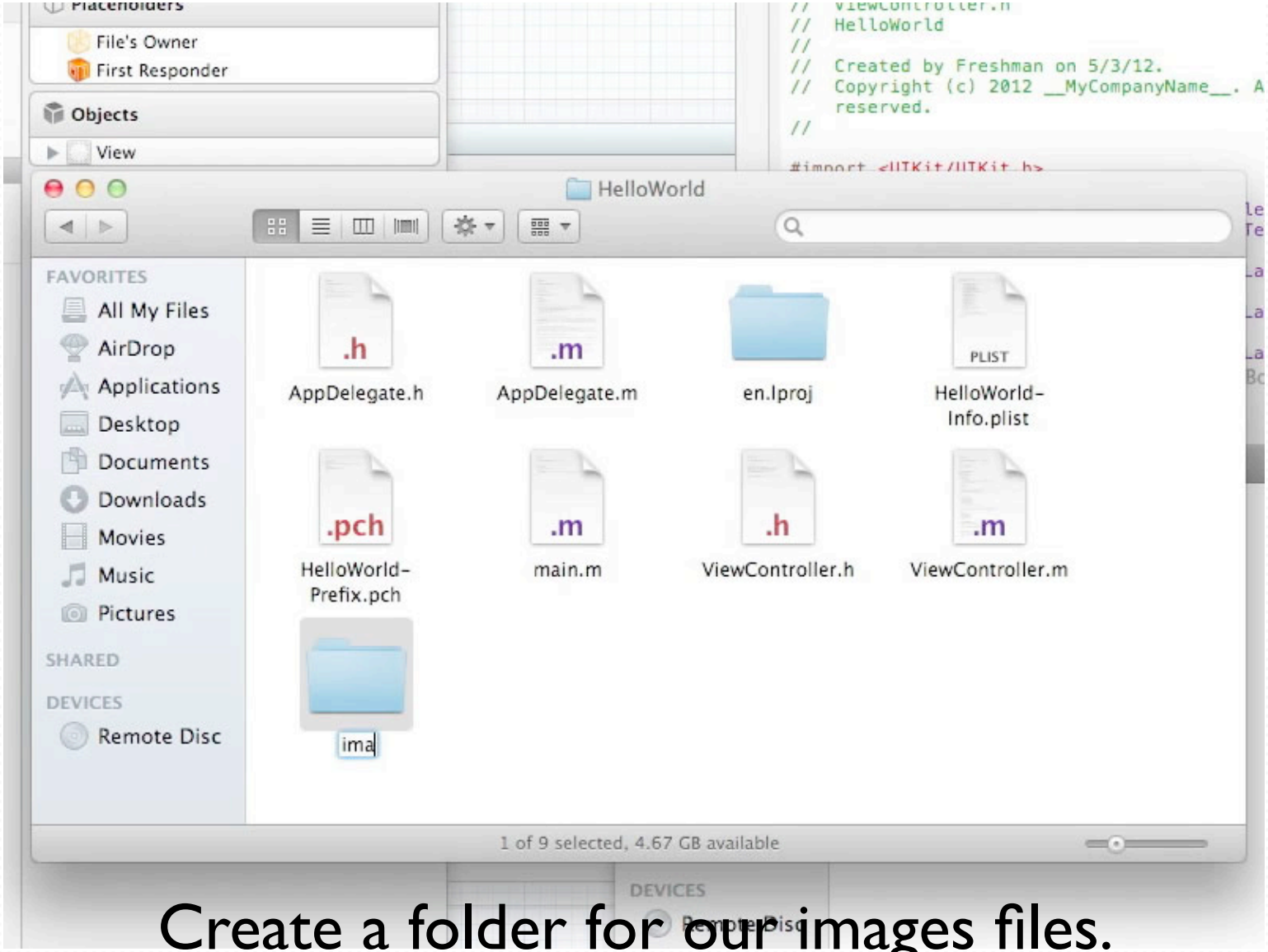
button.png
237 x 40

Images in UIButton



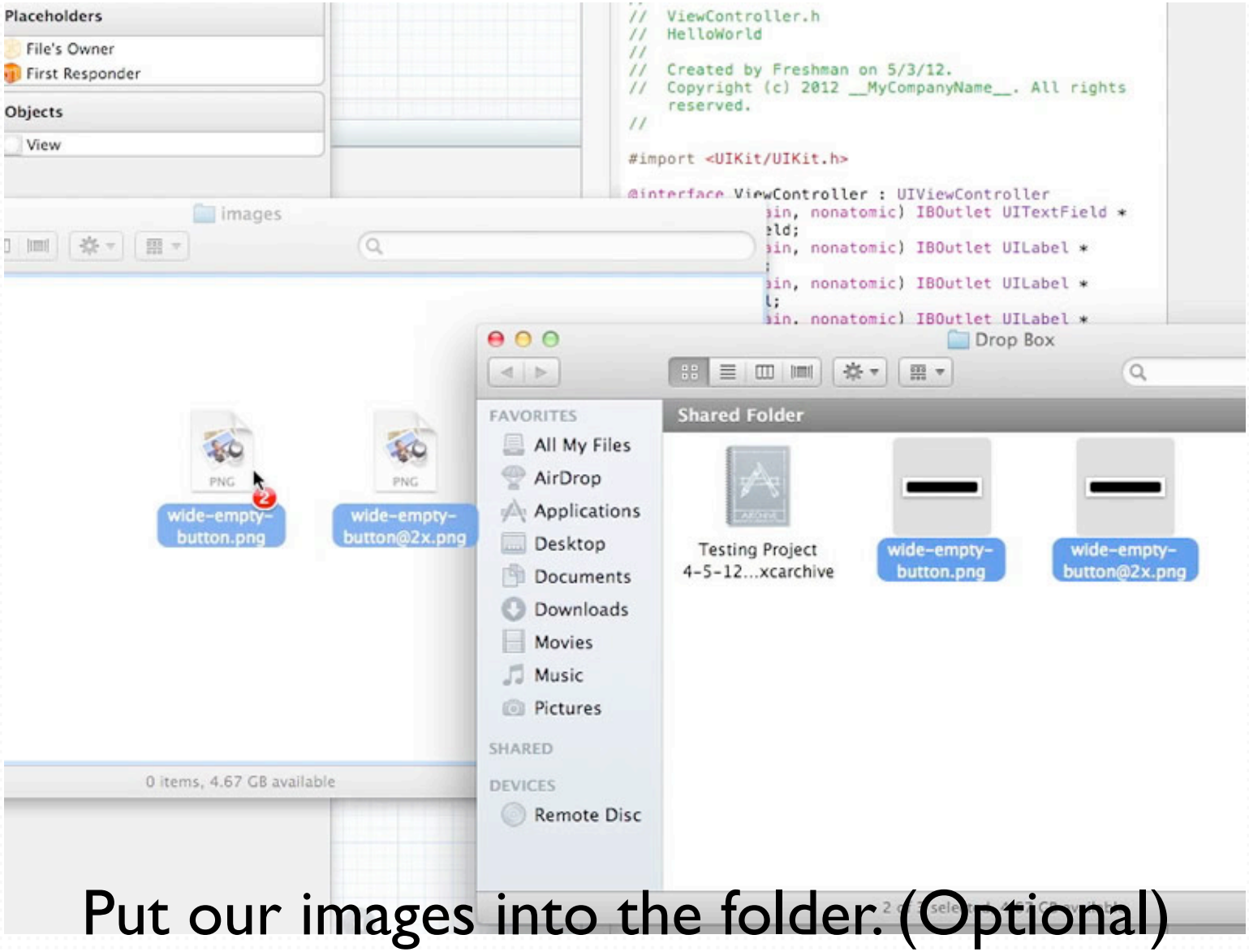
Locate the project files in Finder

Images in UIButton



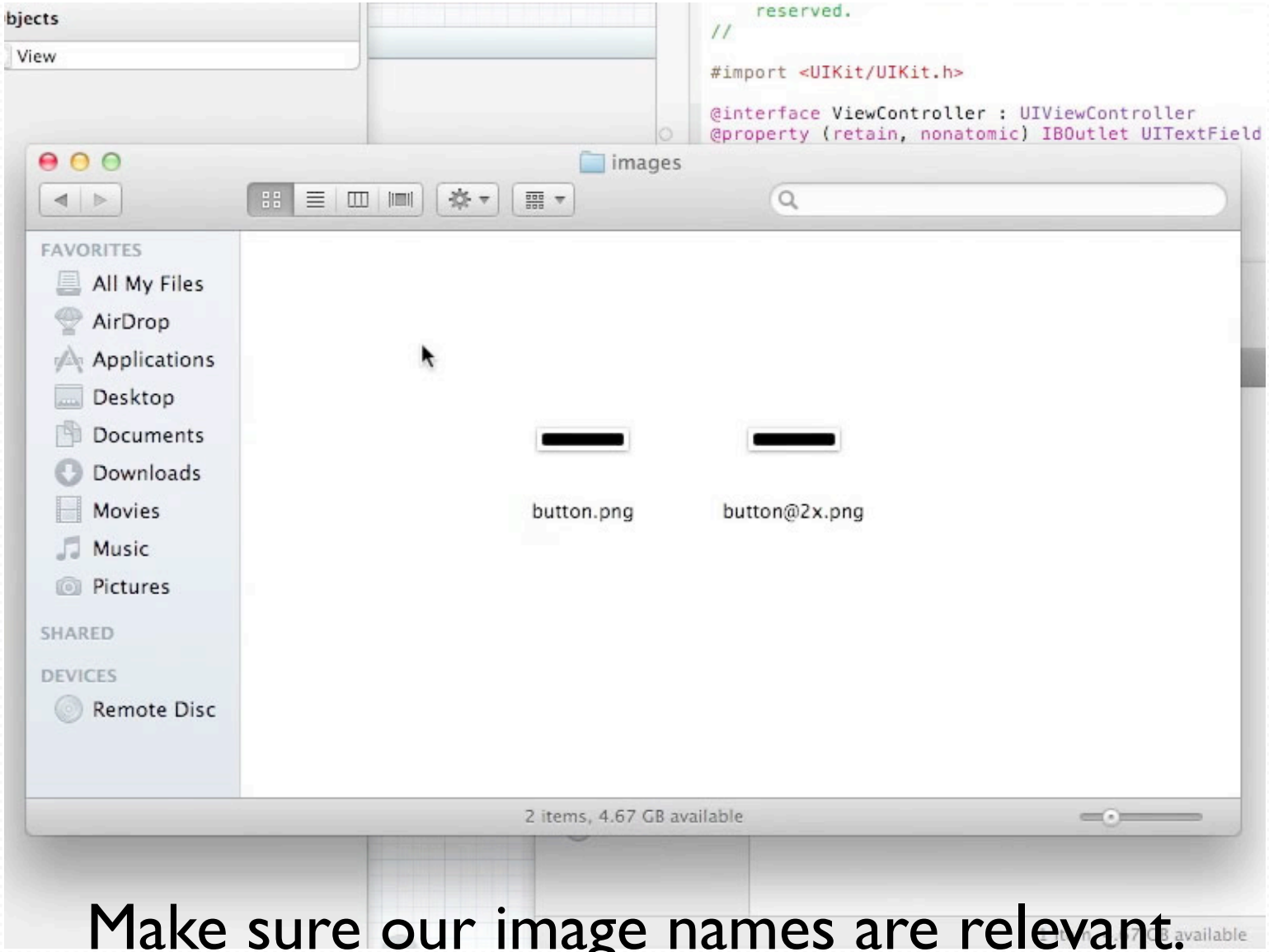
Create a folder for our images files.

Images in UIButton



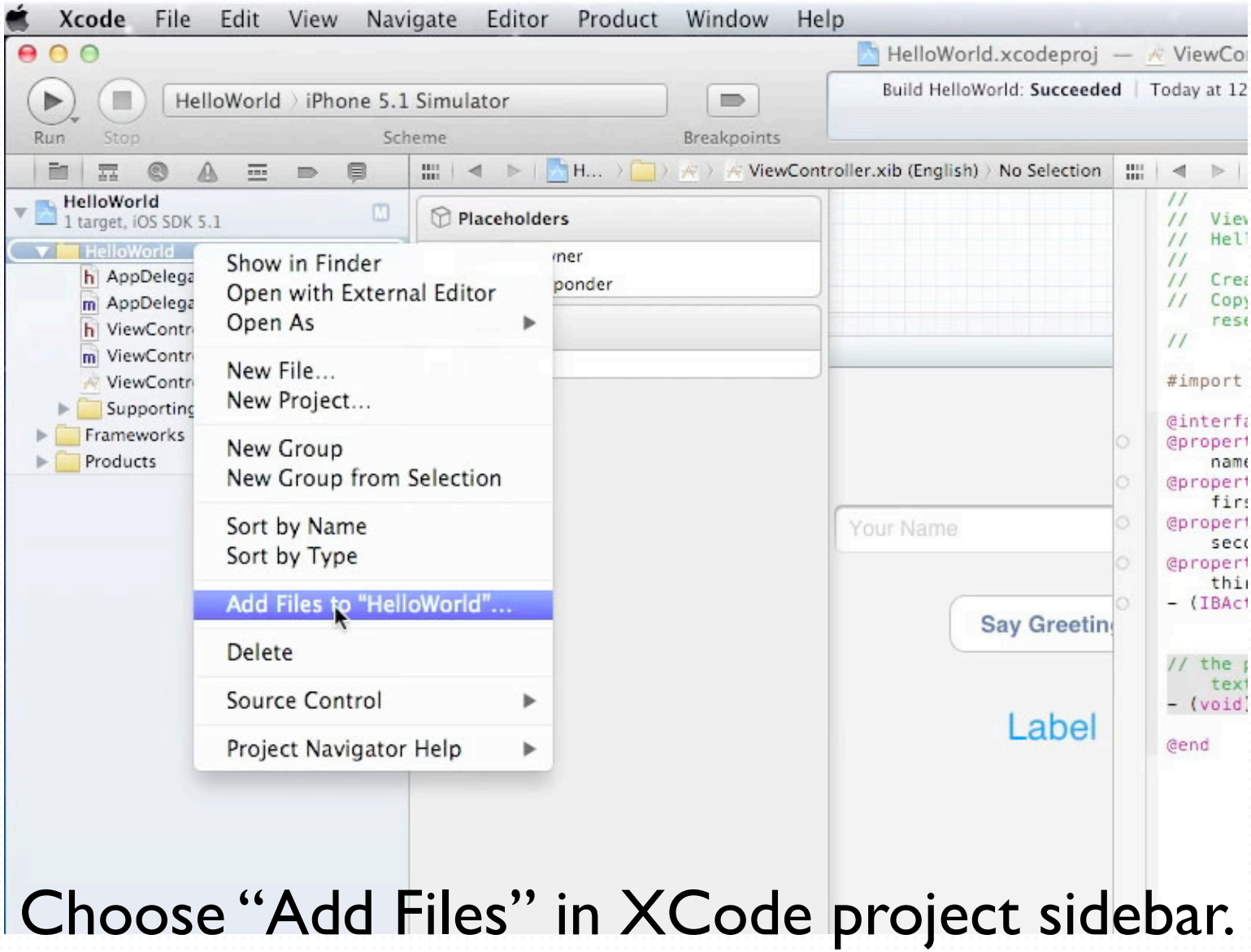
Put our images into the folder. (Optional)

Images in UIButton



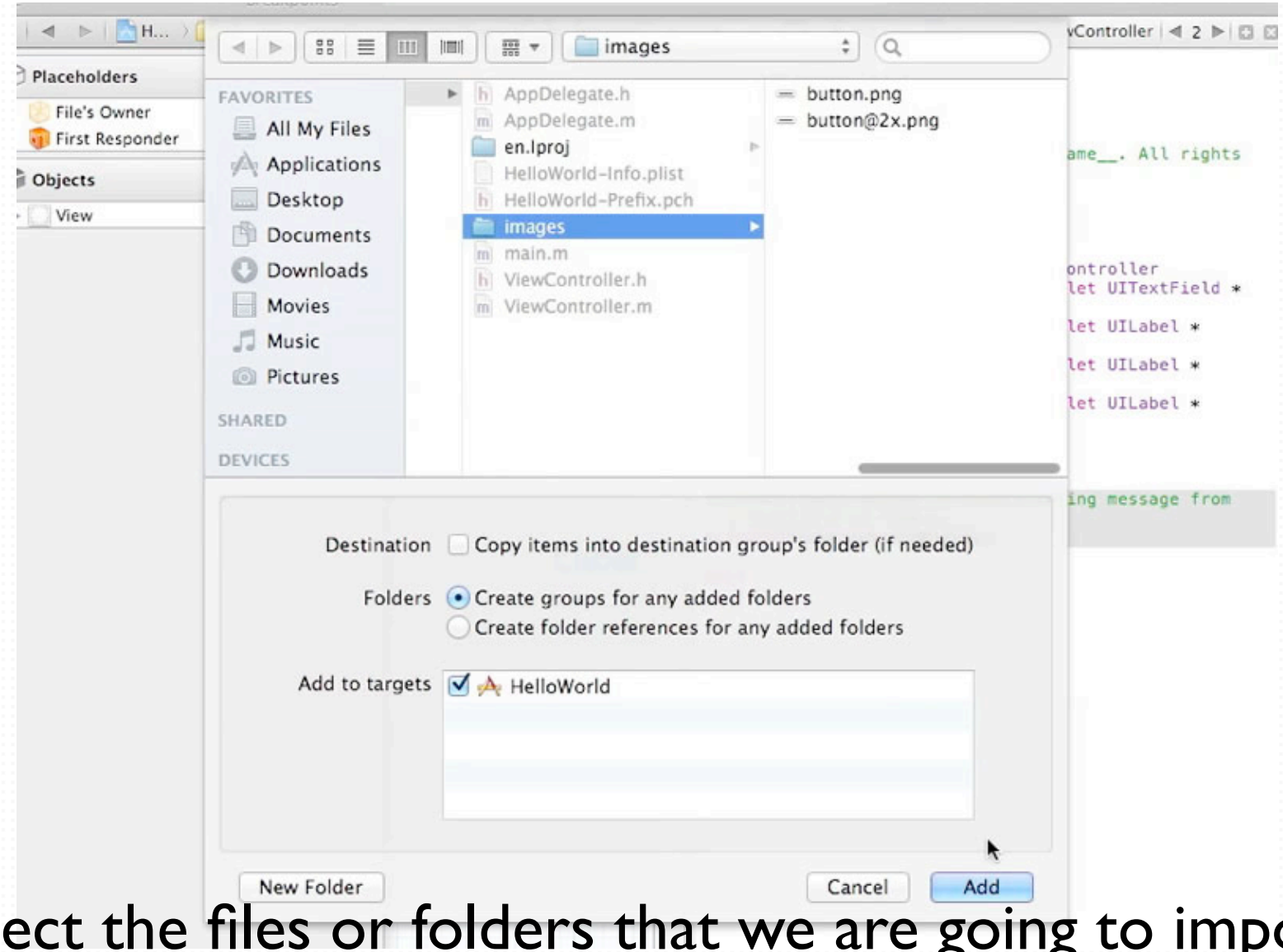
Make sure our image names are relevant.

Images in UIButton



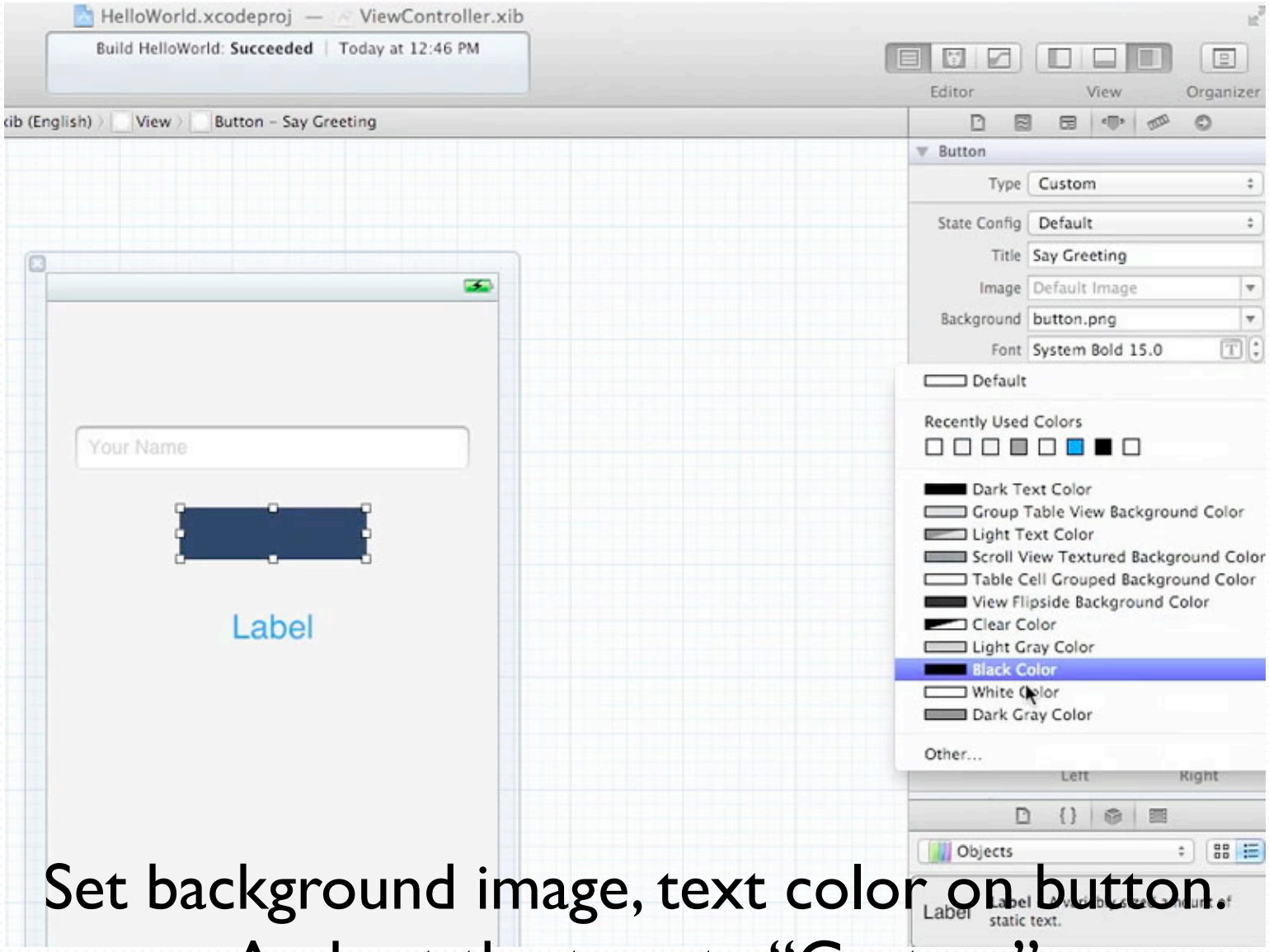
Choose “Add Files” in XCode project sidebar.

Images in UIButton



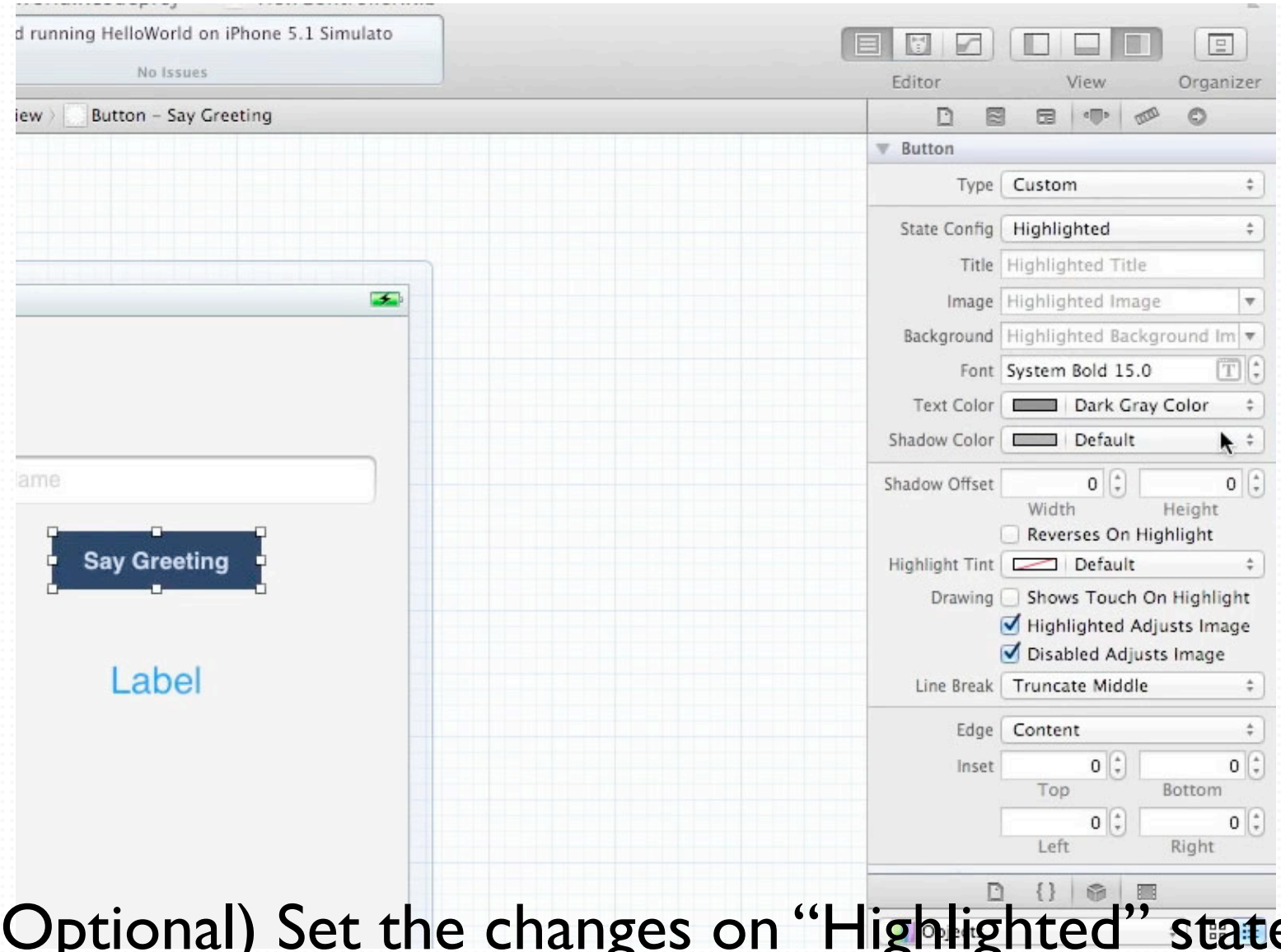
Select the files or folders that we are going to import.

Images in UIButton



Set background image, text color on button.
And set the type to "Custom"

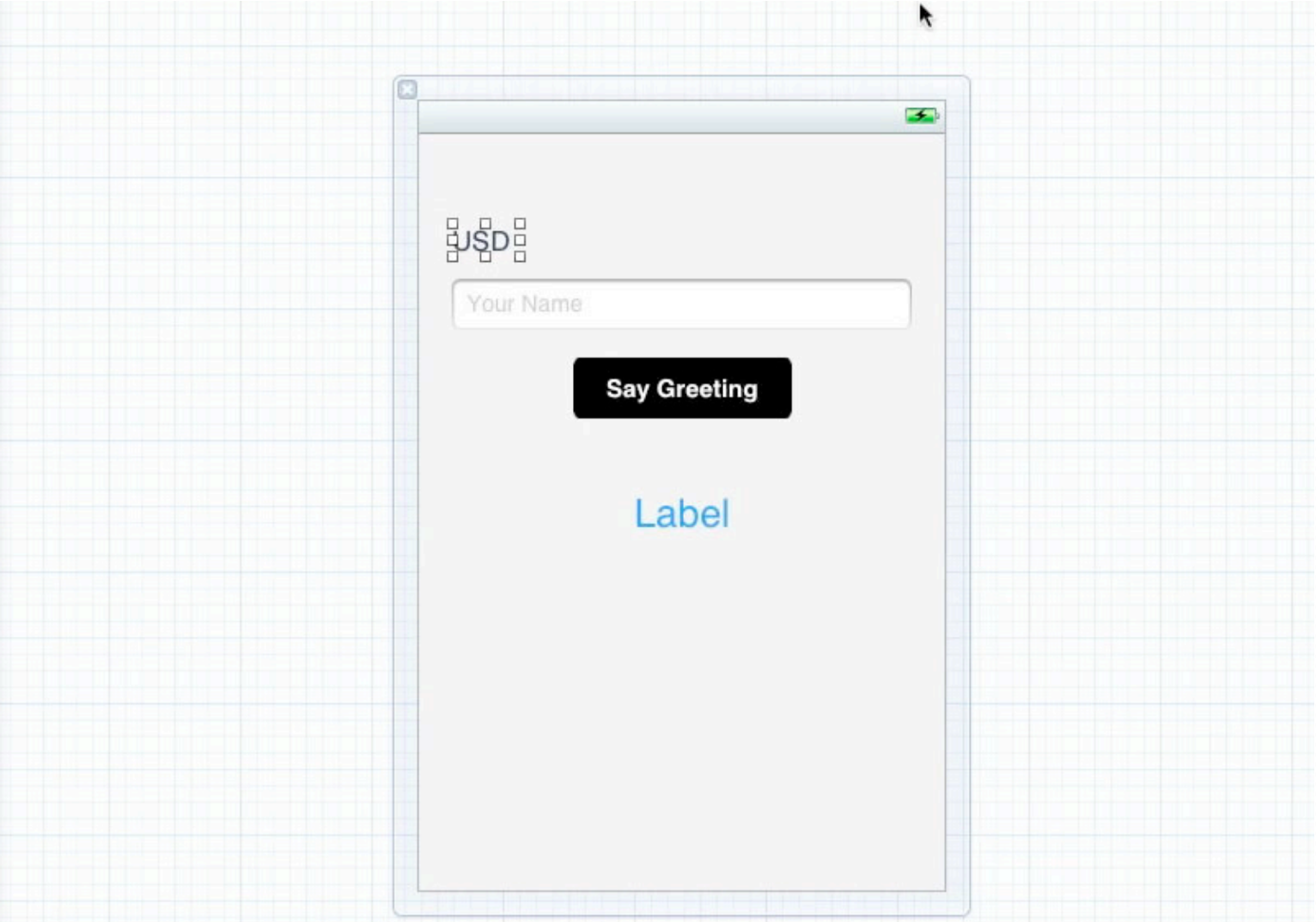
Images in UIButton



(Optional) Set the changes on "Highlighted" state.

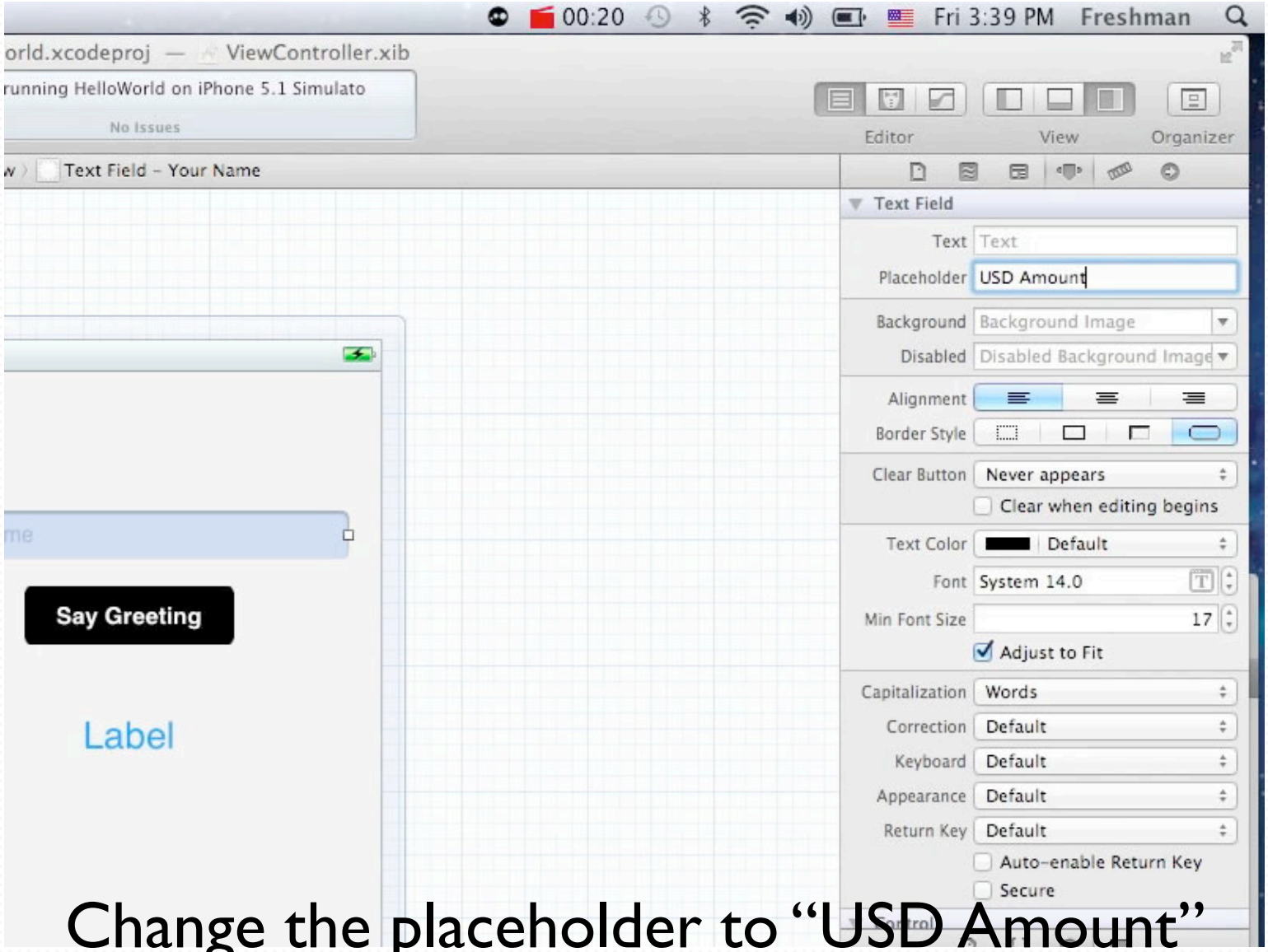
Build a Currency Convertor

Using UIButton



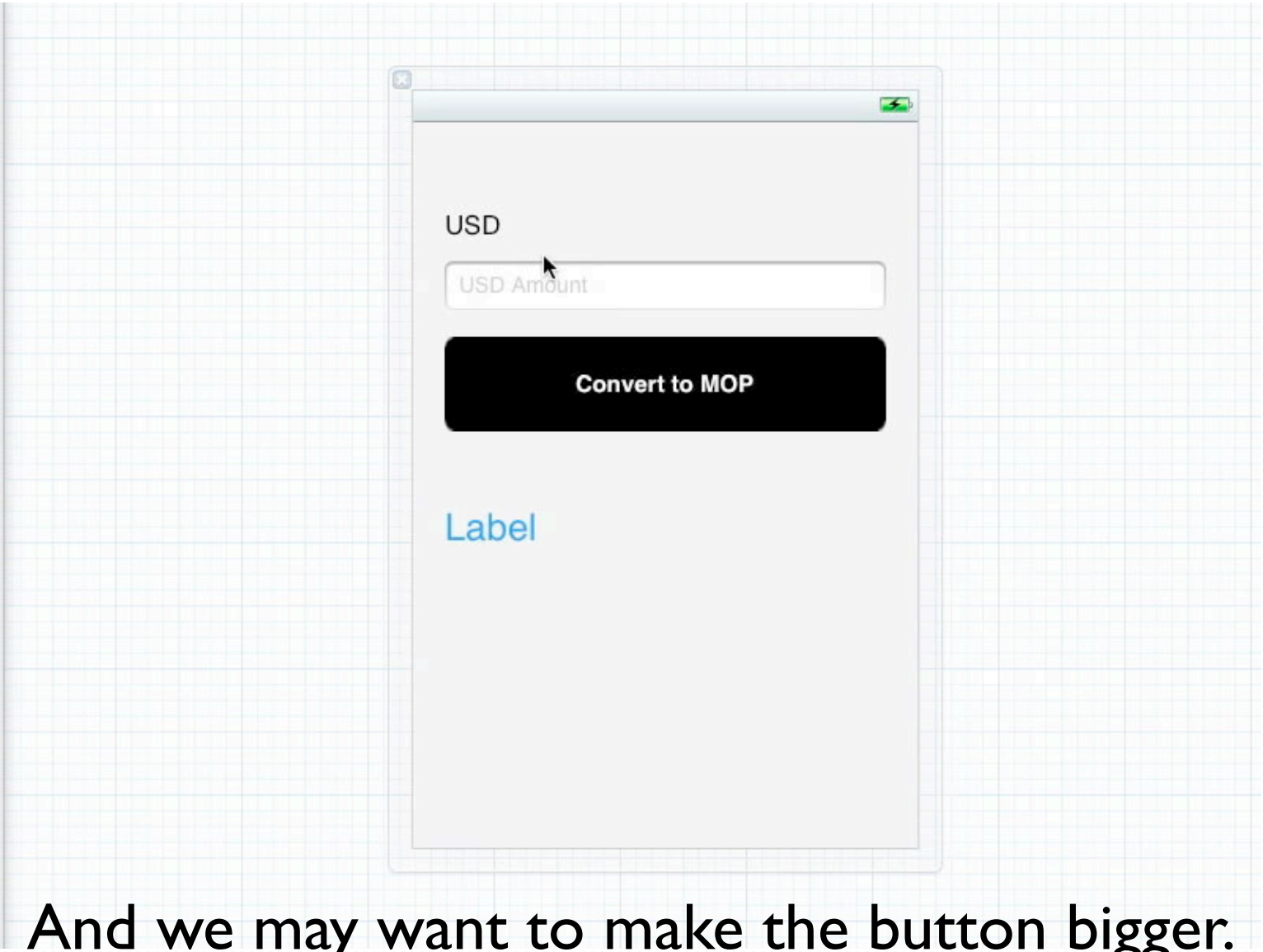
Let's modify the label views for the convertor.

Using UIButton



Change the placeholder to "USD Amount"

Using UIButton



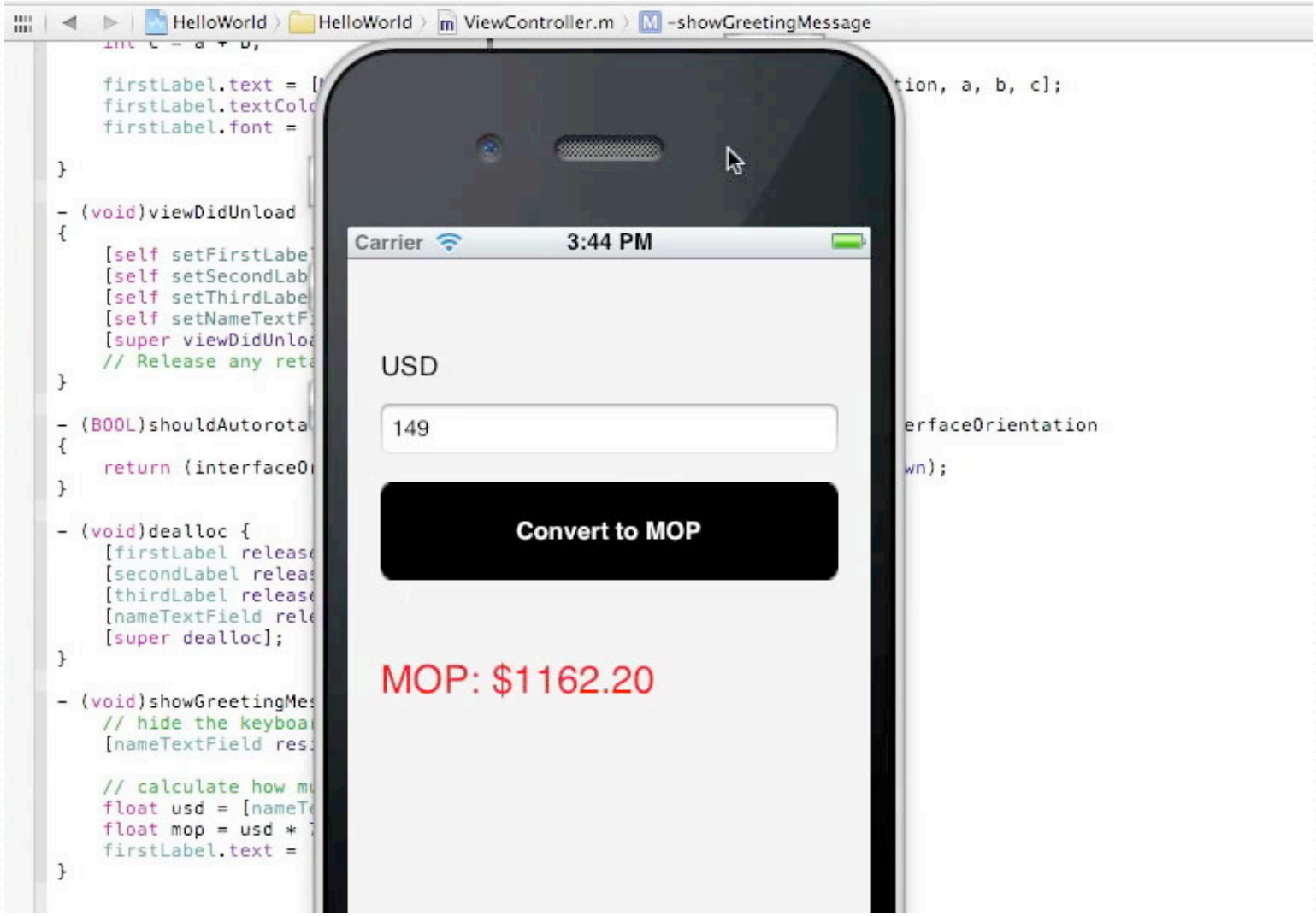
And we may want to make the button bigger.

Using UIButton

```
1 - (void)showGreetingMessage {
2     // hide the keyboard.
3     [self.nameTextField resignFirstResponder];
4
5     // calculate how much is the MOP
6     float usd = [self.nameTextField.text floatValue];
7     float mop = usd * 7.8;
8     self.firstLabel.text = [NSString stringWithFormat:@"MOP:
$%0.2f", mop];
9 }
```

We add some calculation logic when tap the button.

Using UIButton



Finally, let's test the app in simulator.

Exercise

- ✓ Design an utility app.
- ✓ Present it to the class in next lesson.