iPhone App Development

CM420-09-2016-C

Lesson 2

Lecturer

Bryant Tang

bryant.tang14mo@gmail.com

CPTTMLAB_B
pw: cpttm1234

Git

https://github.com/bryanttang/iOS-Class-2016-9

Review

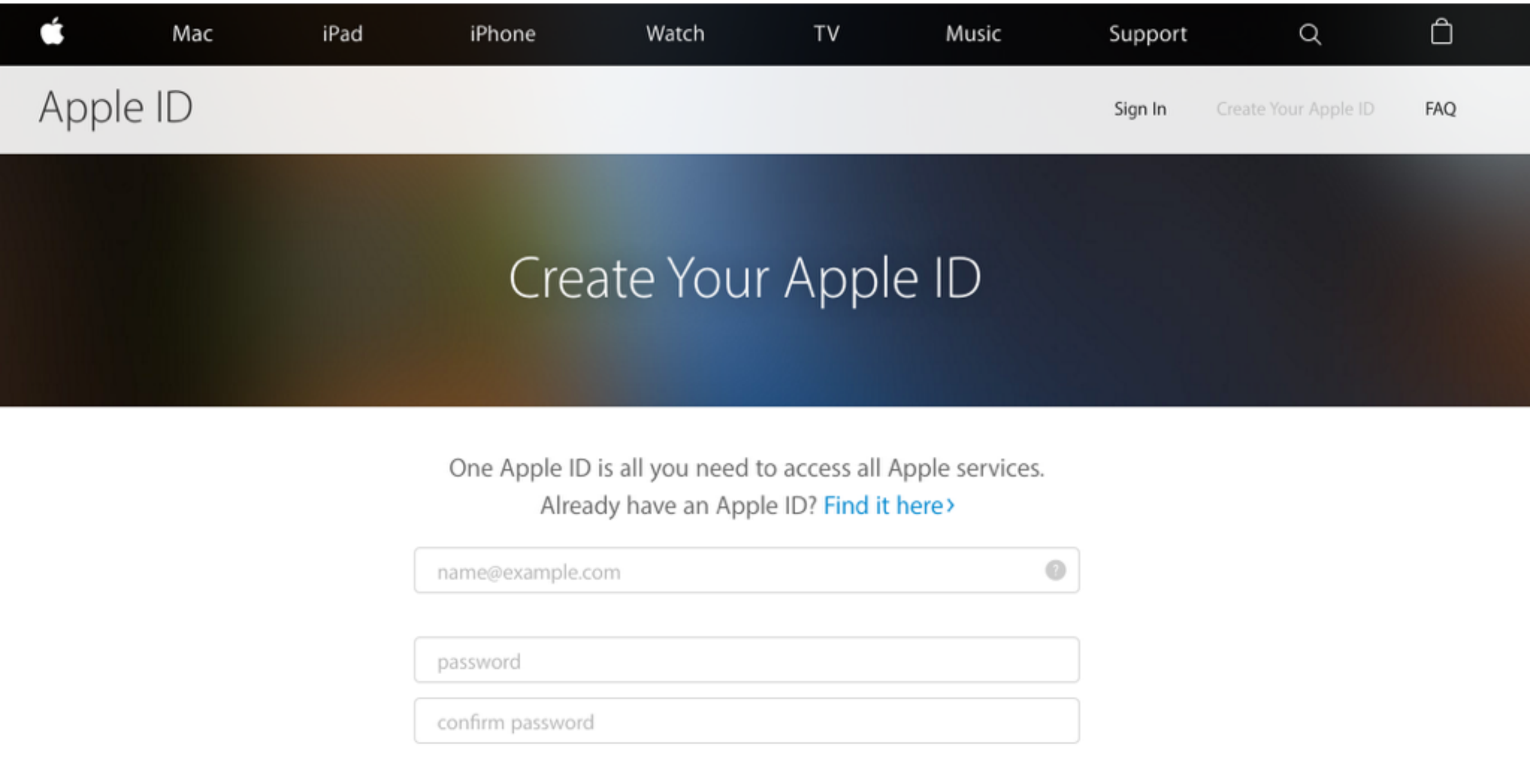
- iOS Architecture - Four layers
- Objective-C
- Create a new project
- Control a UI in a view

Run App on Device

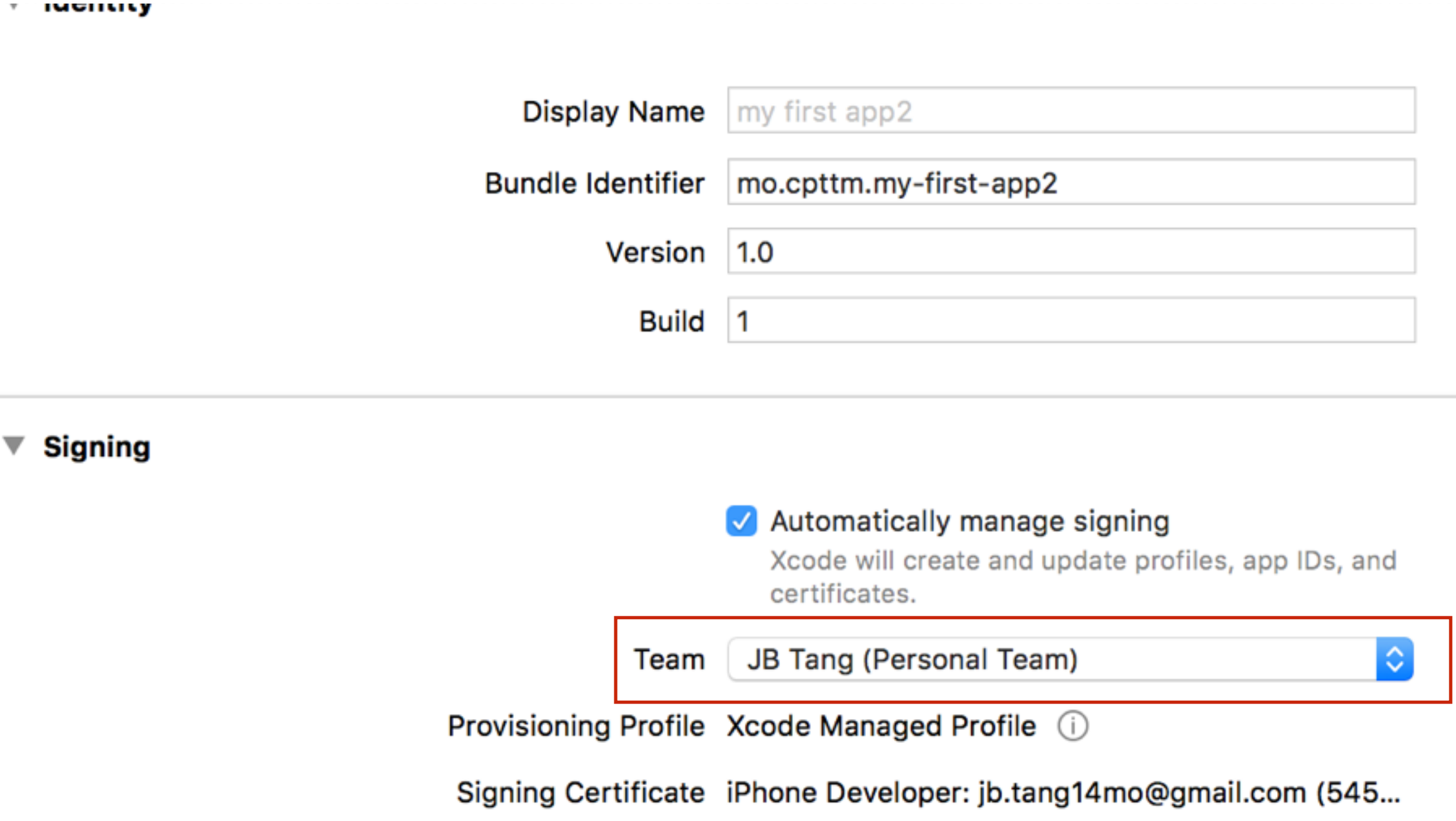
- Register Apple ID
- Add account in Team of Signing section
- Run your app in device
- "Trust" the profile in device Settings . Settings → General → Profile
- Your app is work!

- Register Apple ID

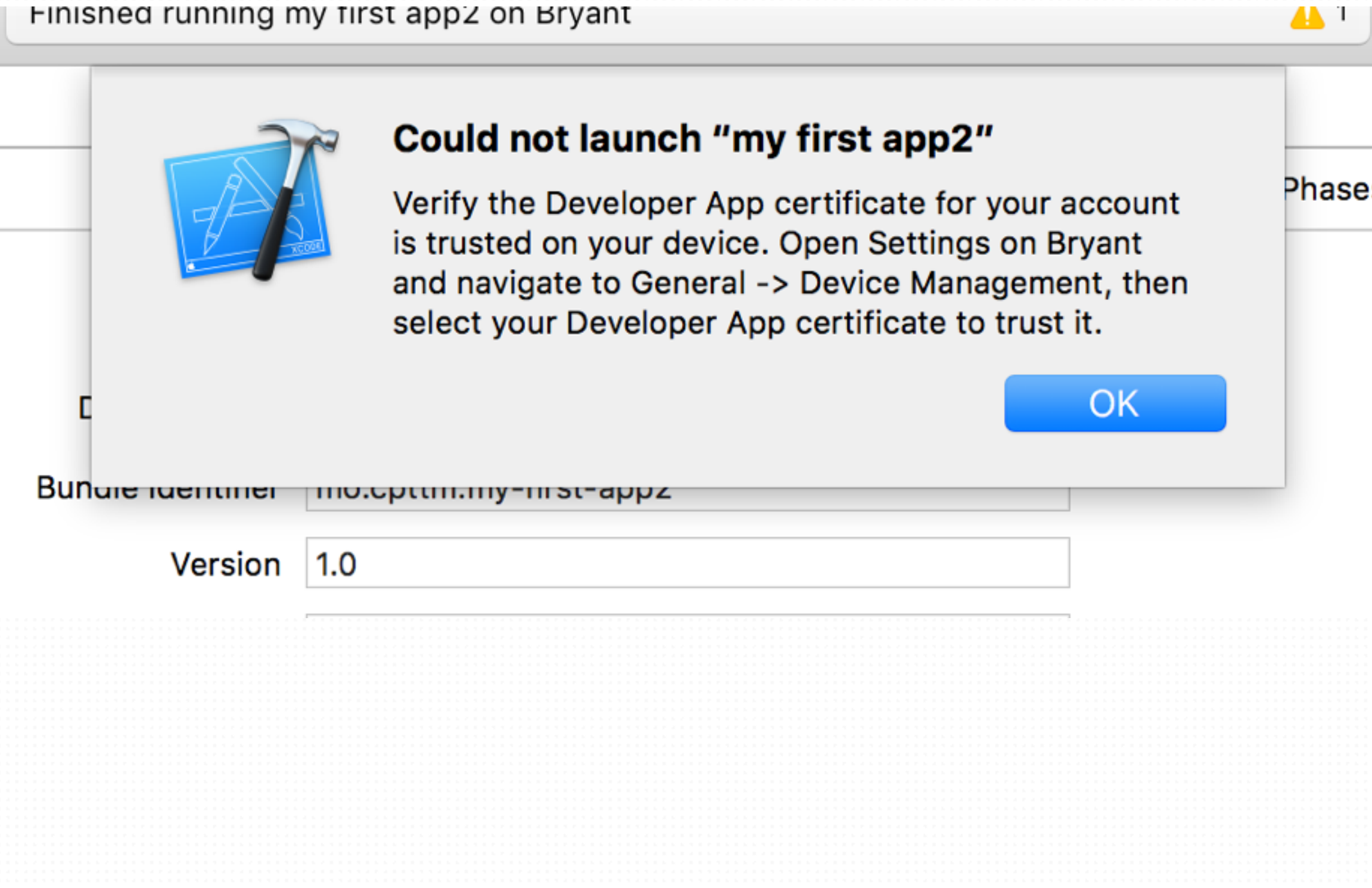
Create Account: https://appleid.apple.com/account#!&page=create



- Add account in Team of Signing section



- Run your app in device



- "Trust" the profile in device Settings

Source

GitHub

Converter

https://github.com/bryanttang/iOS-Class-2015-9

Today's topic

- Objective-C (Object)
- NSString and tracing log message
- Using UILabel
- Delegation
- Events & Using UITextField
- Actions & Using UIButton

Objective-C

Basic of Obj-C

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

Basic of Obj-C

Type

int float double bool char

Declare

```
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.
```

Basic of Obj-C

Loop

1

```
for (int i; i < 10; i++) {
   printf("abc");
}
```

2

```
int i =0;
while (i<10) {
     printf("number: %d \n",i);
     i++;
}
```

Condition

1

```
if (i == 1) {
     printf("i equal 1");
}
```

2

```
switch (i) {
         case 1:
              printf("i equal 1");
              break;
         case 2:
              printf("i equal 2");
              break;
         default:
              printf("i equal nothing");
              break;
    }
```

Object

How to produce a car?

Car Factory

Object

How to produce a car?

Use steel and screw with tool

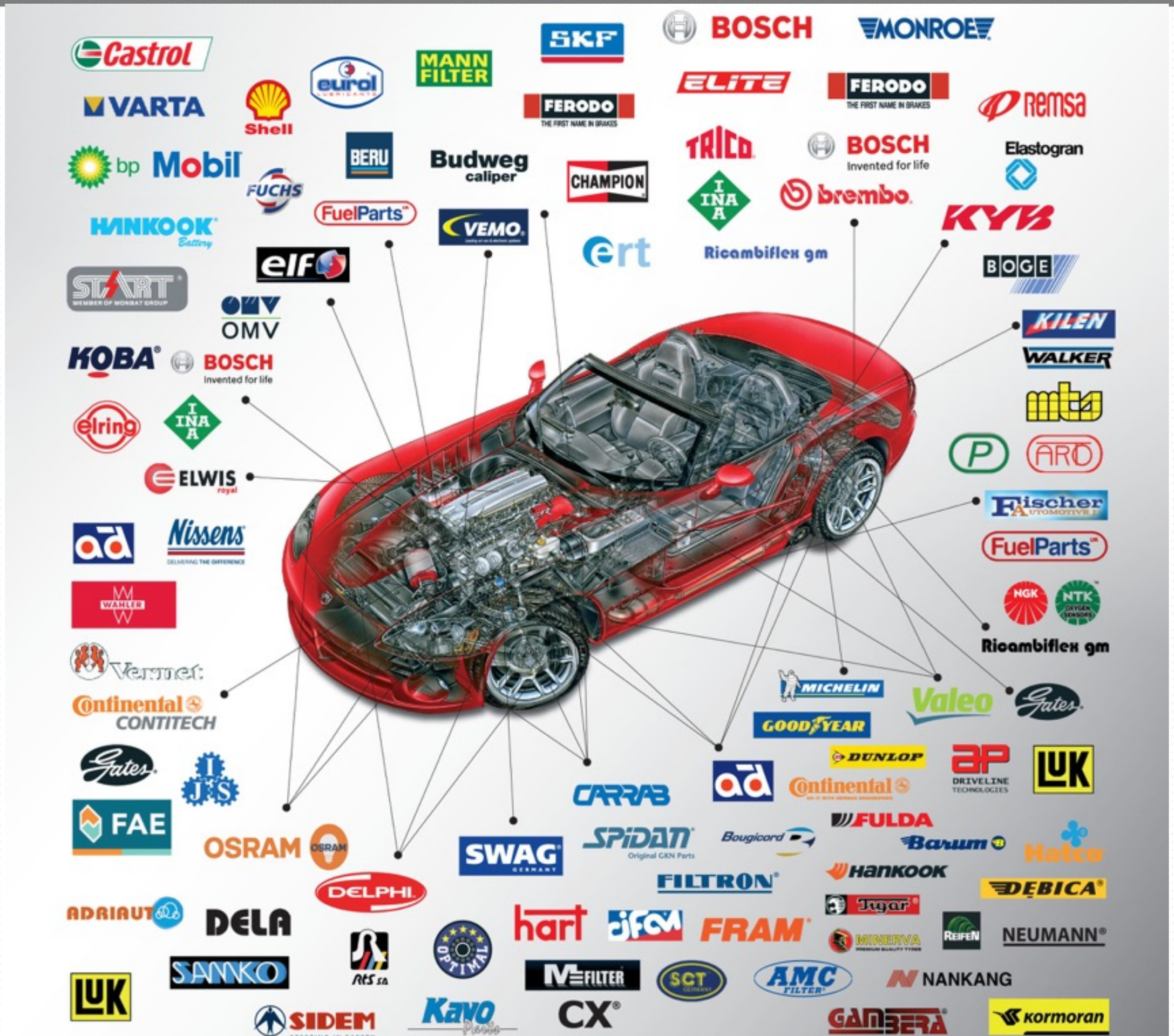
Object

How to produce a car?

Use steel and screw with tool — **Sure, but not smart !!**

Object

- Engine
- Braking System
- Power System
- HiFi
- Air conditioning System
- …


Castrol
VARTA
Shell
eurol
MANN FILTER
SKF
BOSCH
MONROE
ELITE
FERODO
THE FIRST NAME IN BRAKES
Remsa
bp
Mobil
FUCHS
BERU
Budweg caliper
TRICO
BOSCH
Invented for life
Elastogran
CHAMPION
INA
brembo
HANKOOK
Battery
FuelParts
VEMO
ert
Ricambiflex gm
KYB
elf
START
MEMBER OF MONBAT GROUP
BOGE
OMV
KILEN
KOBA
BOSCH
Invented for life
WALKER
elring
INA
ELWIS
royal
ARD
Fischer
AUTOMOTIVE
ad
Nissens
DELIVERING THE DIFFERENCE
FuelParts
WAHLER
NGK
NTK
OXYGEN SENSORS
Ricambiflex gm
Vernet
MICHELIN
Valeo
Continental
CONTITECH
GOODYEAR
Gates
DUNLOP
ap
DRIVELINE TECHNOLOGIES
LUK
Gates
IJS
ad
Continental
CARRAB
FAE
FULDA
OSRAM
SWAG
SPIDAN
Original GKN Parts
Bougicord
Barum
Hatco
HANKOOK
FILTRON
DELPHI
DEBICA
ADRIAUT
DELA
hart
FRAM
Tigar
MINERVA
REIFEN
NEUMANN
SAMKO
RTS SA
OPTIMAL
MFILTER
SCT
AMC FILTER
NANKANG
LUK
SIDEM
Kavo Parts
CX
GAMBERA
kormoran

Why Object?

- Reuse
- Encapsulation
- Productivity
- Maintainability

Object

- Engine

  - Speed
  - Temperature
  - Status

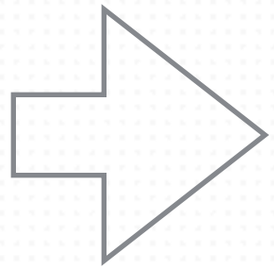
  Attributes/Property

  - - Start
  - - Shut down
  - - Accelerate
  - - Moderate

  Behaviour

Object

- Engine
  - Speed
  - Temperature
  - Status
  - - Start
  - - Shut down
  - - Accelerate
  - - Moderate



- Object
  - variable1
  - variable 2
  - variable 3
  - - method1
  - - method2
  - - method3
  - - method4

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 behaviour of object.
- Methods have strong relationship with data inside the object.
- We can think in higher level to tackle the problem.

Practice

- How do you design Calculator as an object?
- List the main feature and behaviour in a simple Calculator.

Practice

- Calculator
  - Result
  - Operand
  - Operator (Addition, Subtraction, multiplication, Division)
  - Calculate
  - GetResult
  - Clear

Syntax

- Property -

Syntax

Some.property

Example

calculator.result

Syntax

- Function -

Syntax [Some doSomething]

Example [calculator printResult]

Syntax

With input

Syntax [calculator doSomethingWithParam1: Param2:]

Example [calculator add:11 :12]

With Output

Syntax variable = [calculator doSomething]

Example int result = [calculator getStoreResult]

Class and Instance

- Class is prototype of a kind of Object.
- Class is like a factory of 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

Interface

- An example of Obj-C interface

```
#import <UIKit/UIKit.h>

@interface Engine: NSObject{
    NSString *Brand;
    NSString *Status;
    NSNumber *speed;
    NSNumber *temperature;

}

-(void)startUp;
-(void)stop;
-(void)Accelerate;
-(void)Moderate;
@end
```

Implementation

- An example of .m

```
#import "Engine"

@interface Engine

@end

@implementation

-(void)startUp{
    status = @“start”;
}

-(void)stop{
    status = @“stop”;
}

-(void)Accelerate{
    speed+=2;
}

-(void)Moderate{
    speed-=2;
}

@end
```

Instance

New an instance from Engine

```
Engine *engine1 = [[Engine alloc] init];
[engine1 startUp];
```

Build a Currency Convertor


電信業者
12:52 PM
USD
12
Convertor to MOP
MOP = 95.849520
Q W E R T Y U I O P
A S D F G H J K L
Z X C V B N M
123
space
return

NSString

NSString

NSString != String

String is **data type**

NSString is **Object**

NSString

Create NSString object

1

```
NSString * str = @“hello”;
```

2

```
NSString * str = [NSString stringWithFormat:@“%@”, @“hello”];
```

3

```
NSString * str = [@“hello ” stringByAppendingString:
@“everyone”];
```

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"
```

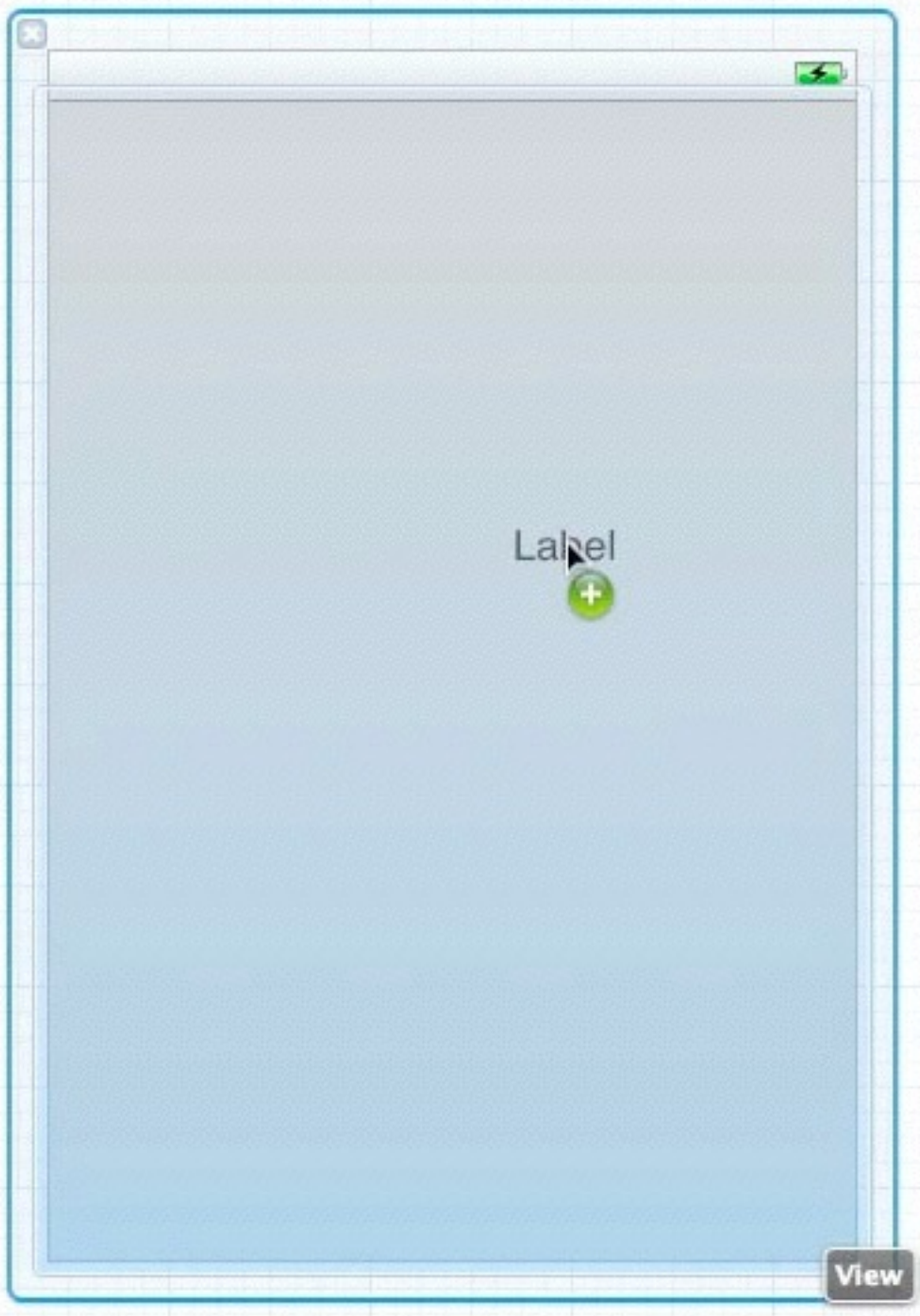
NSLog - Log Message

- **%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.

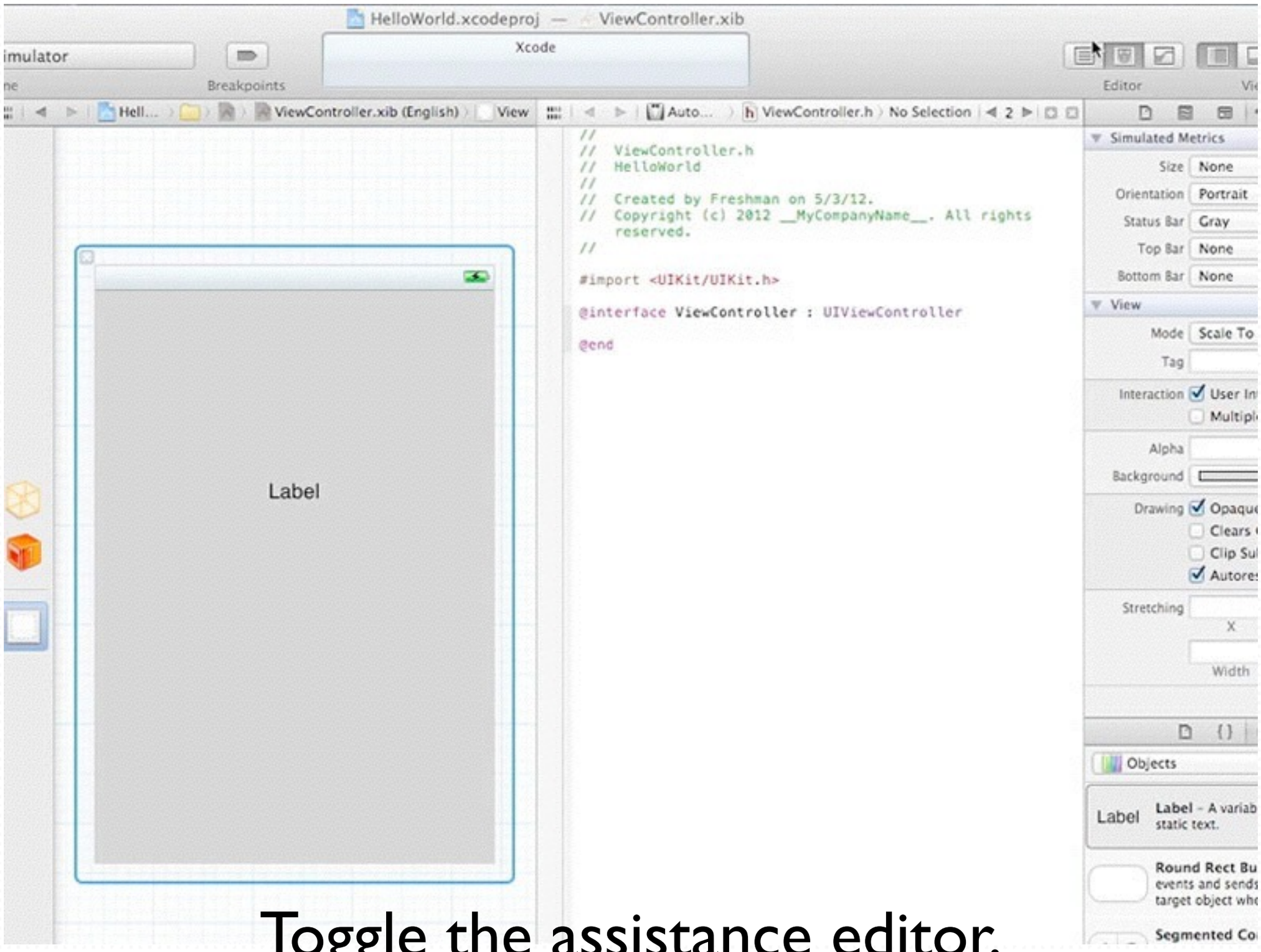
Using UILabel

Using UILabel



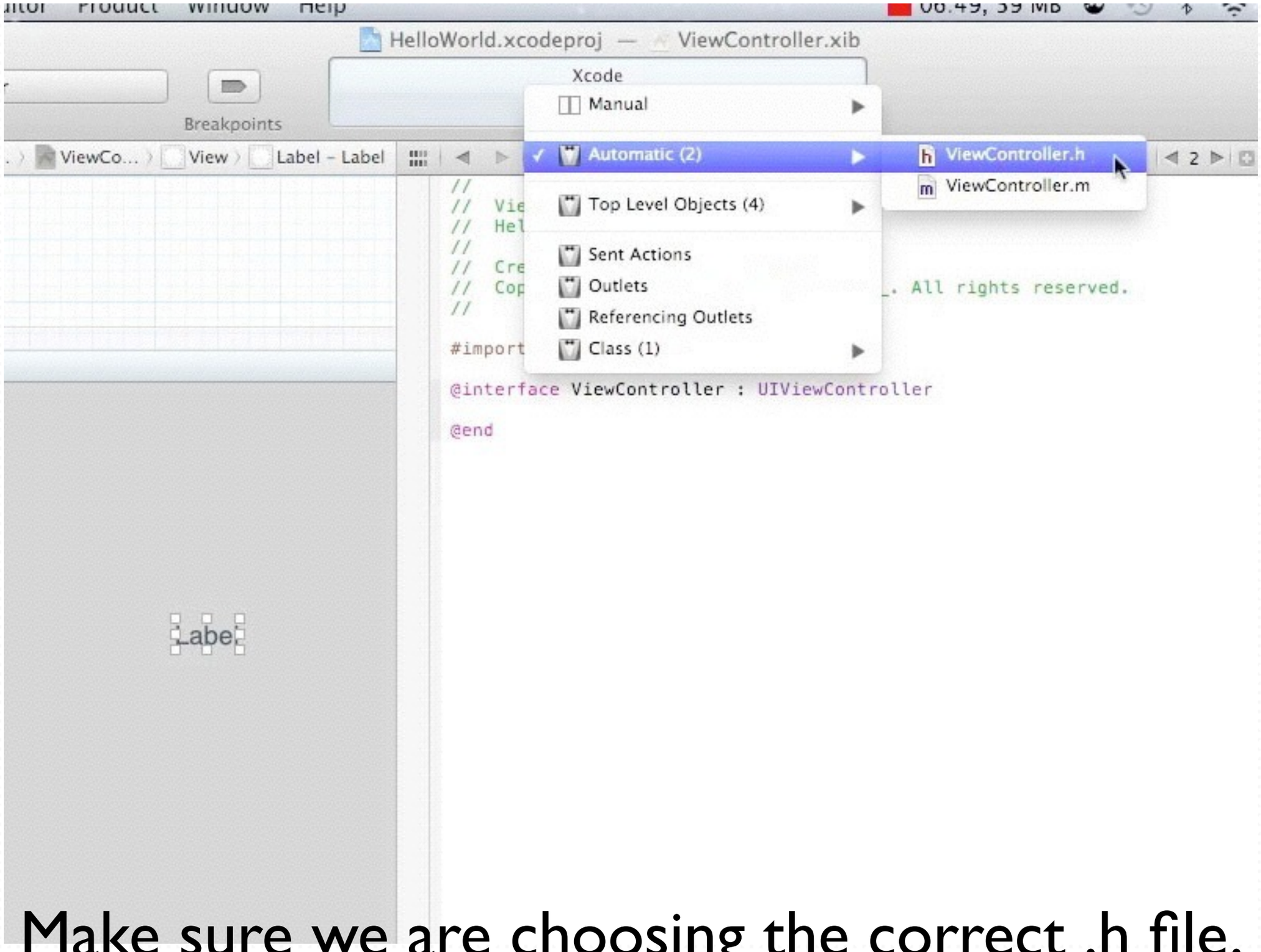
Dragging a Label into the view.

Using UILabel



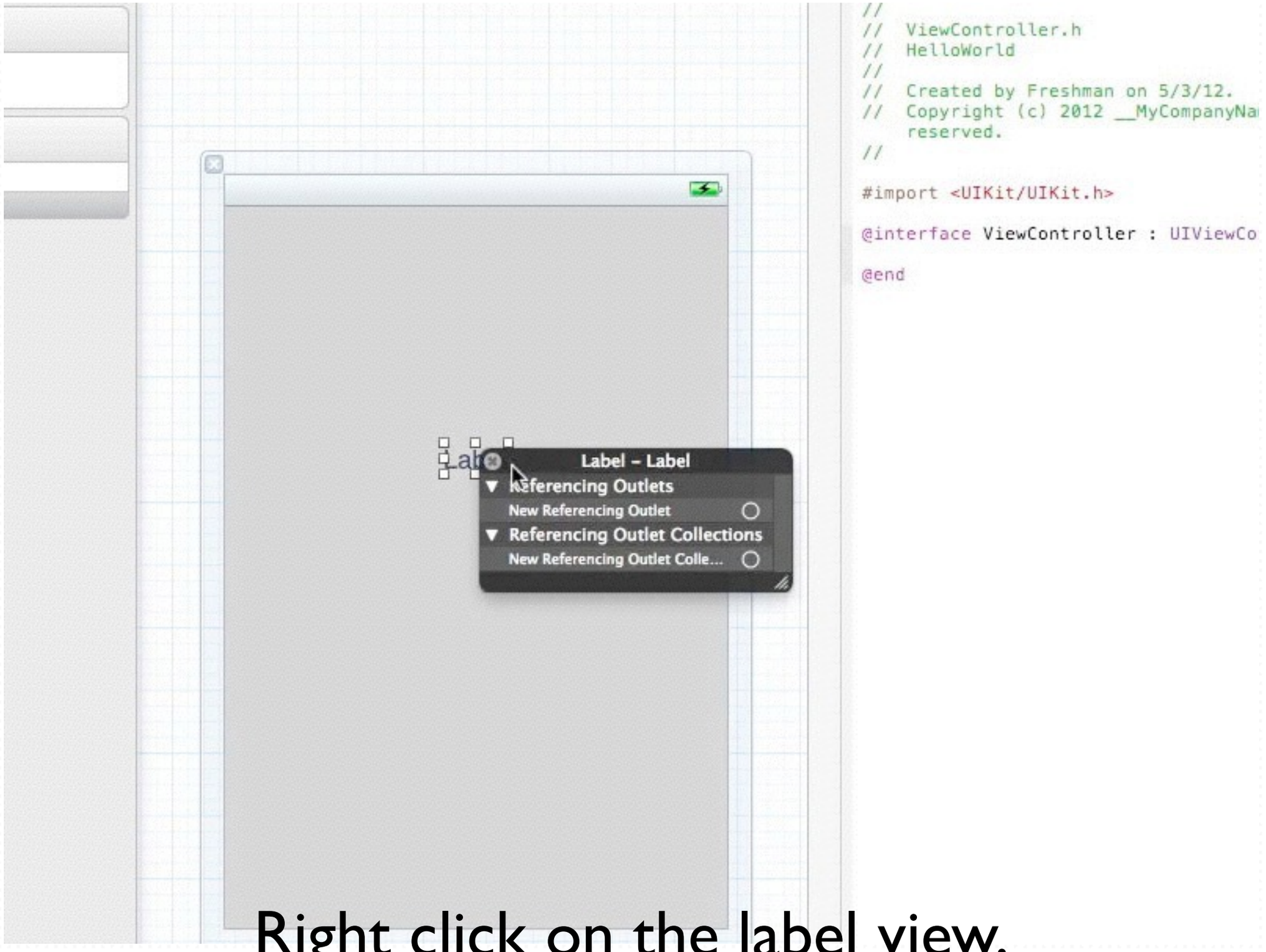
Toggle the assistance editor.

Using UILabel



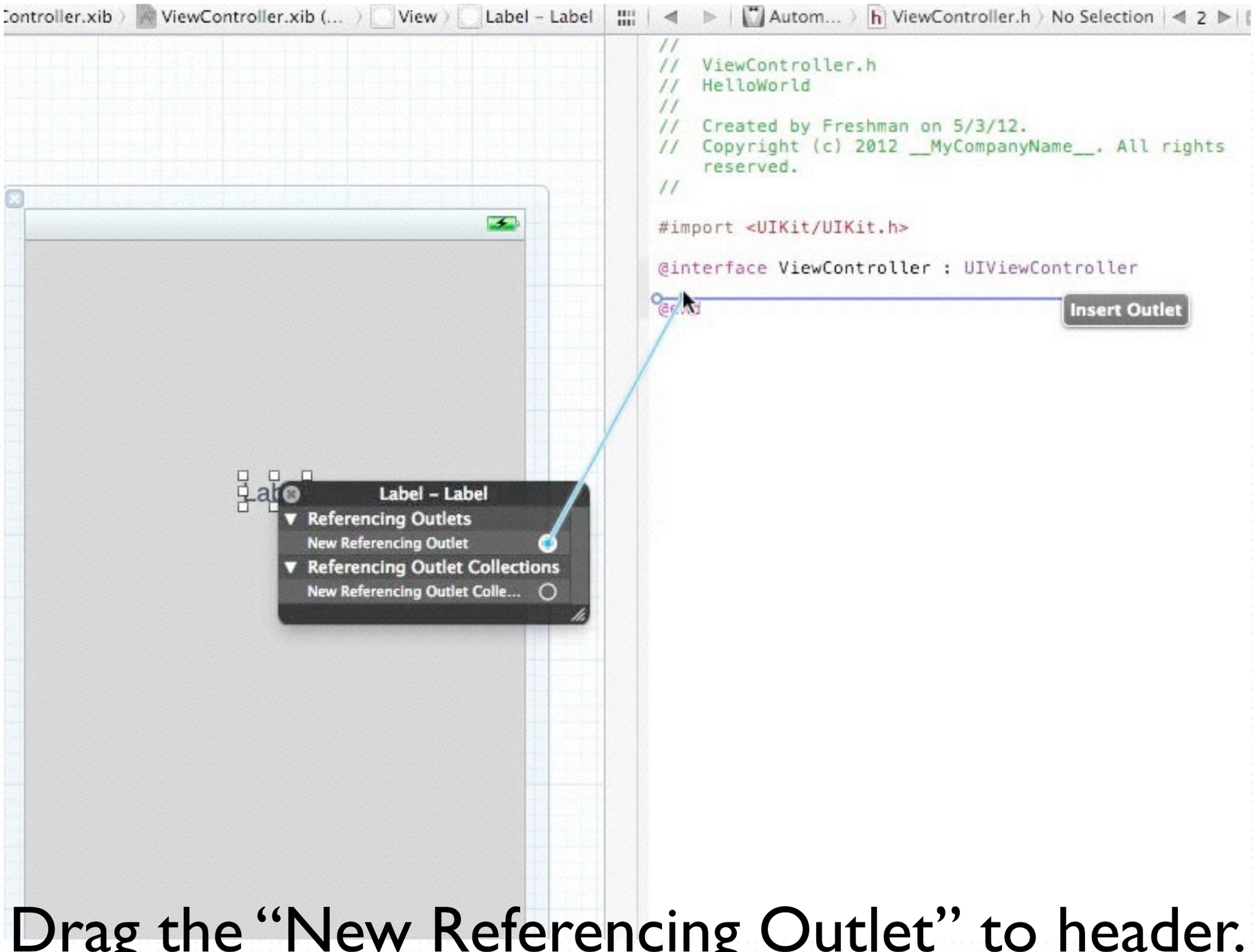
Make sure we are choosing the correct .h file.

Using UILabel



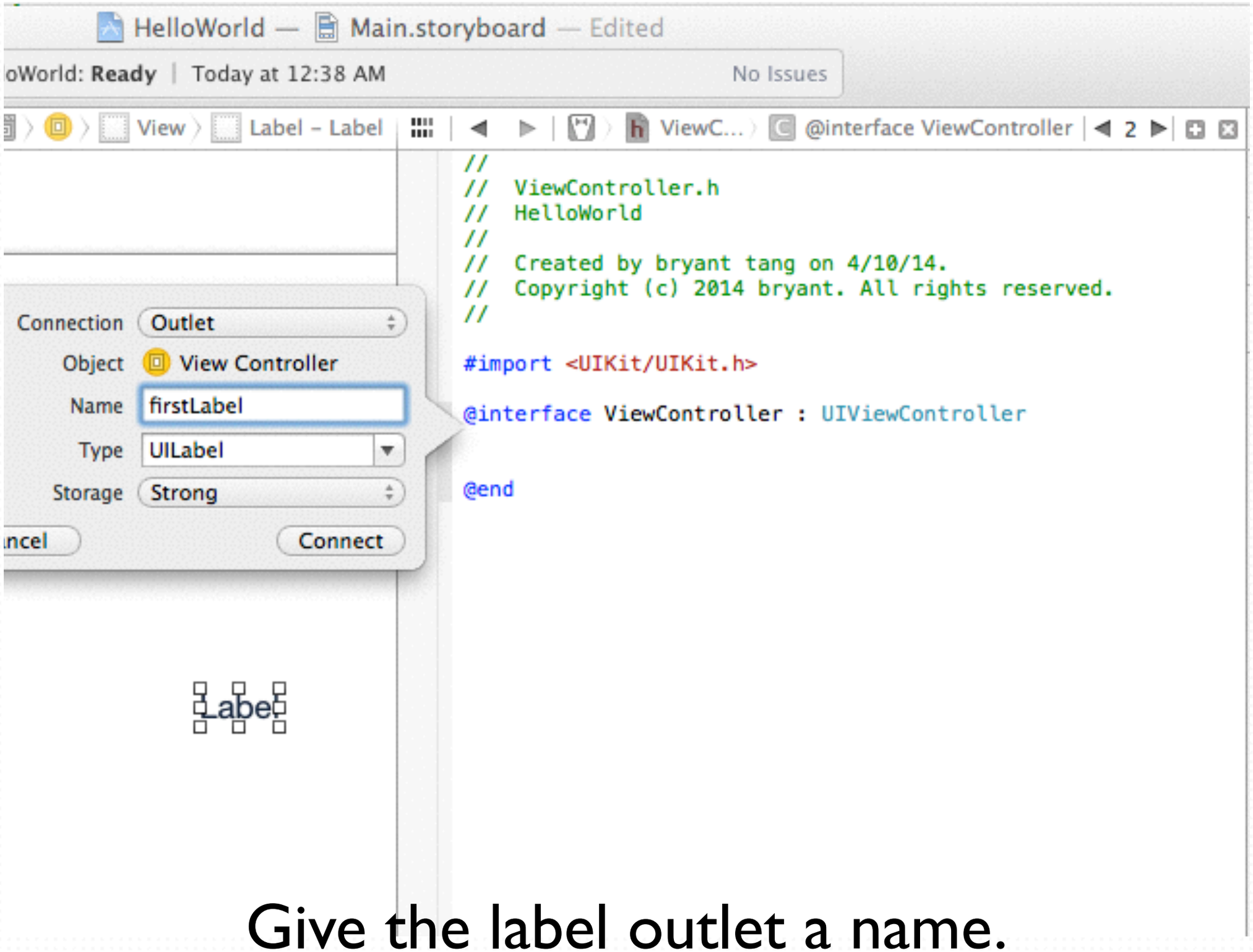
Right click on the label view.

Using UILabel



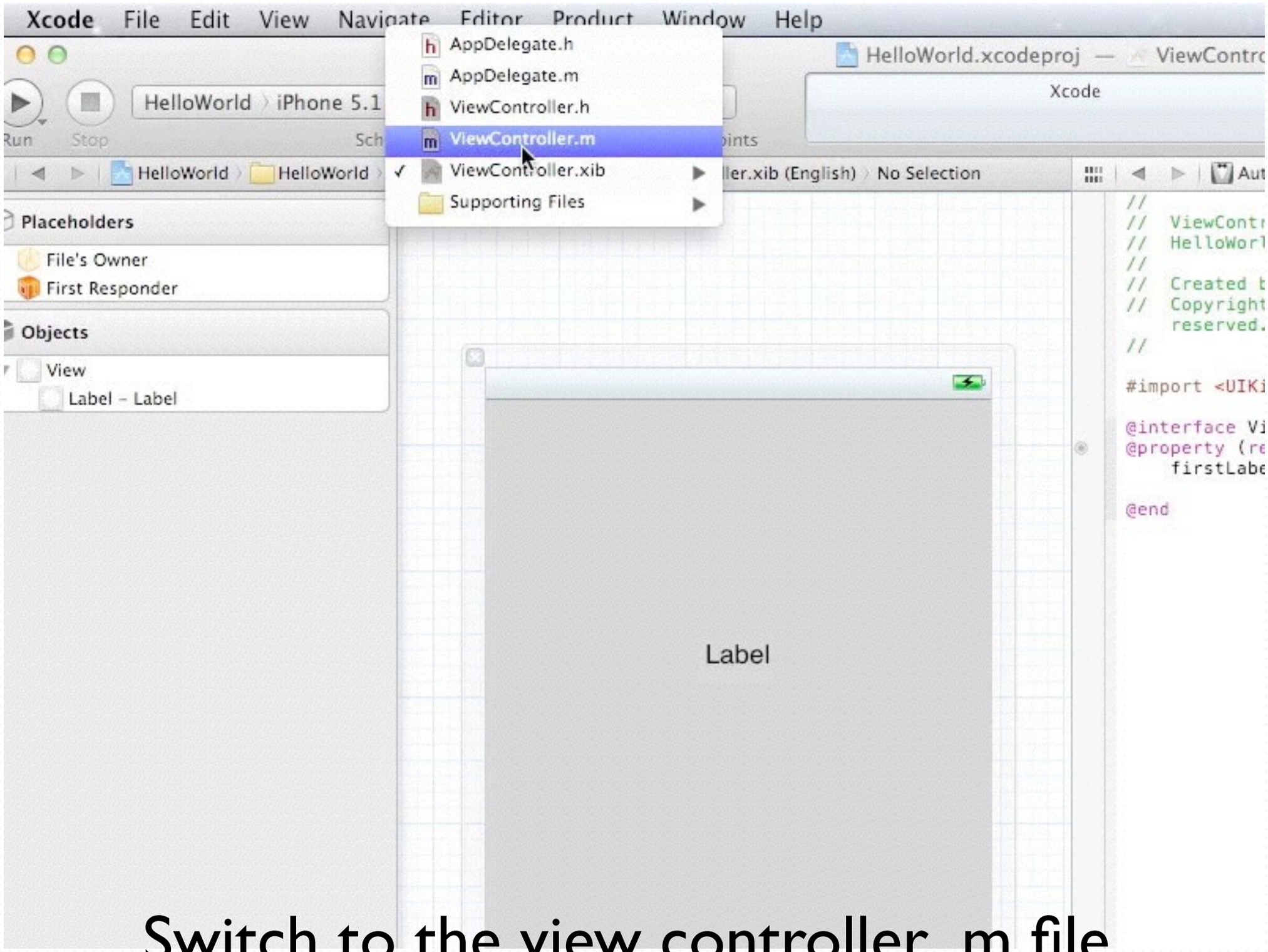
Drag the “New Referencing Outlet” to header.

Using UILabel



Give the label outlet a name.

Using UILabel



Switch to the view controller .m file.

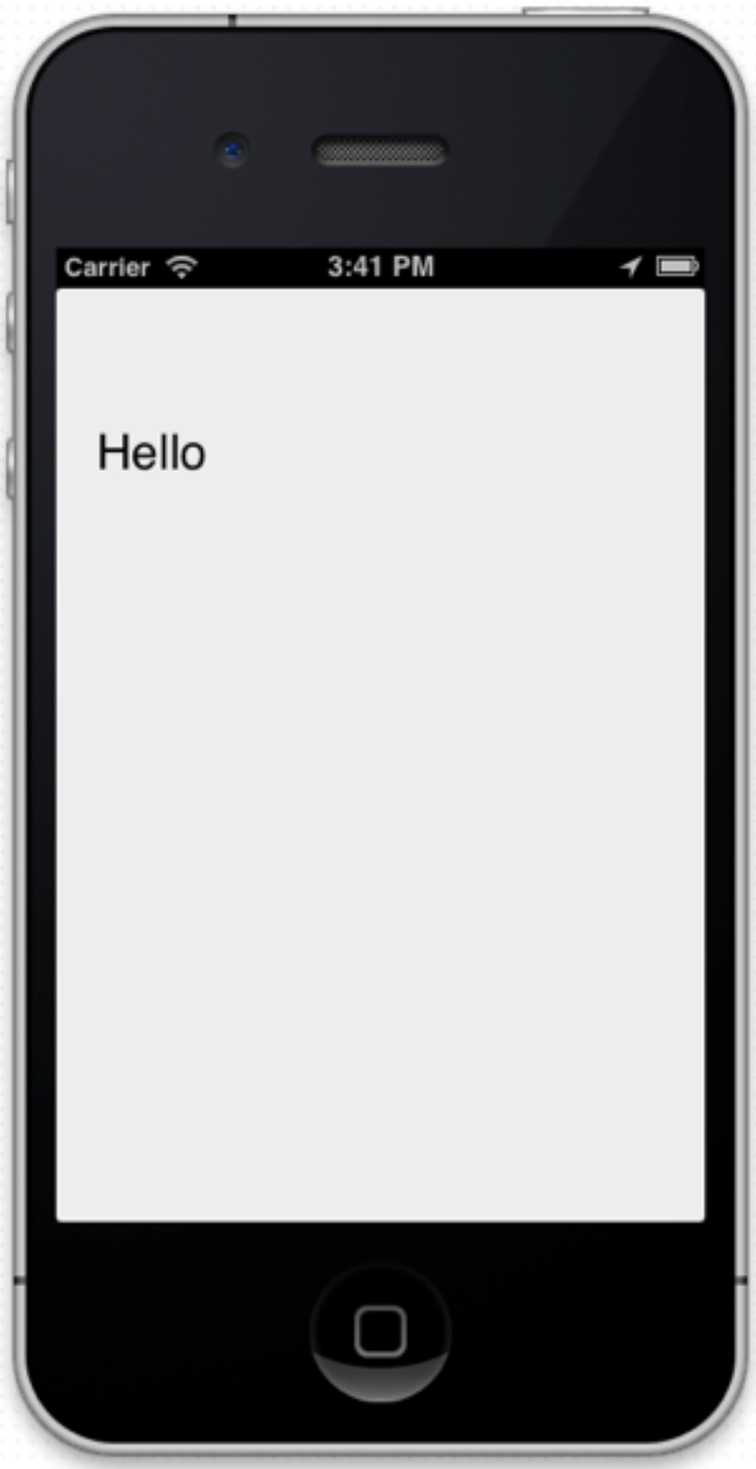
Using UILabel

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

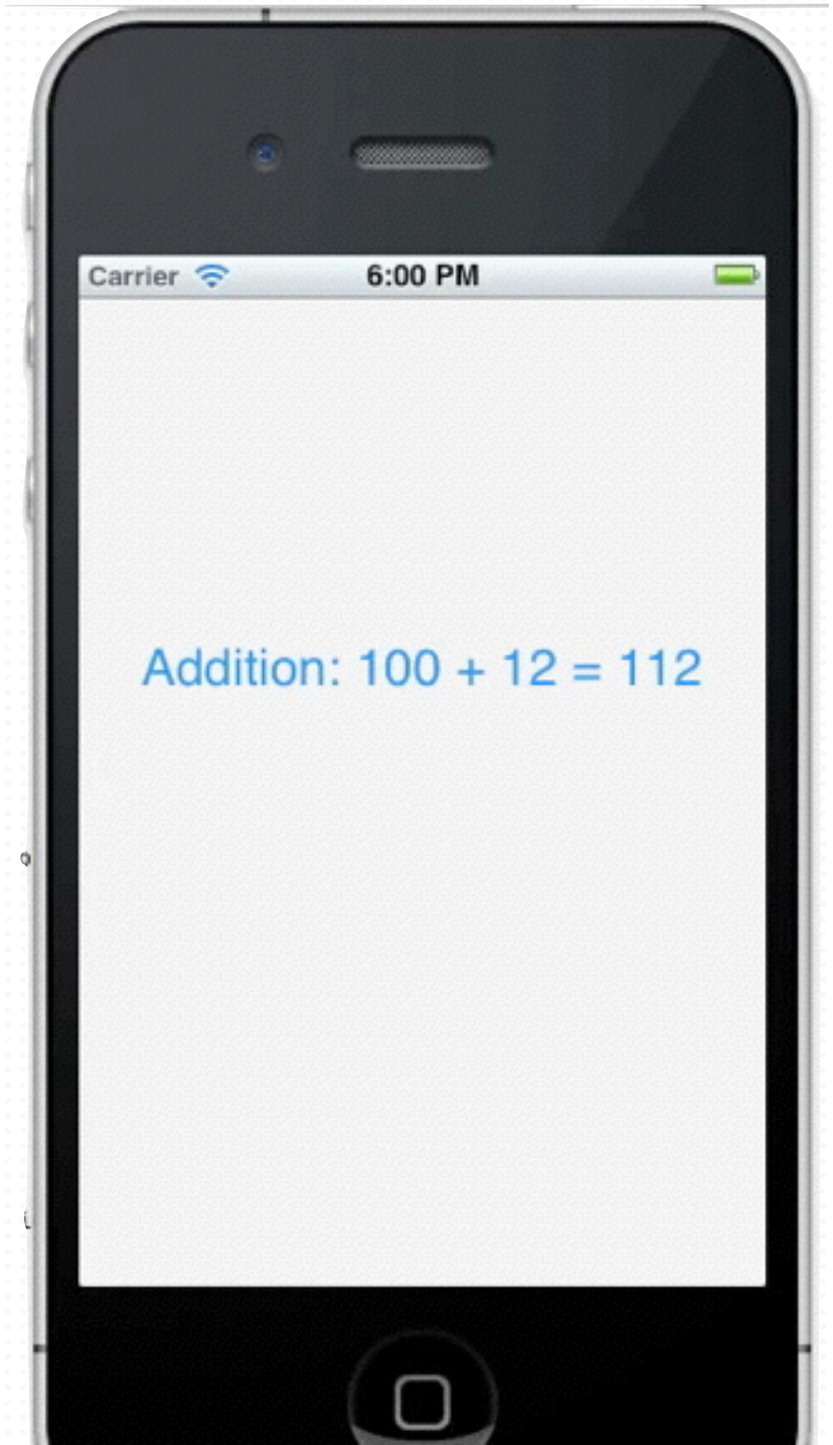


Using UILabel

- 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

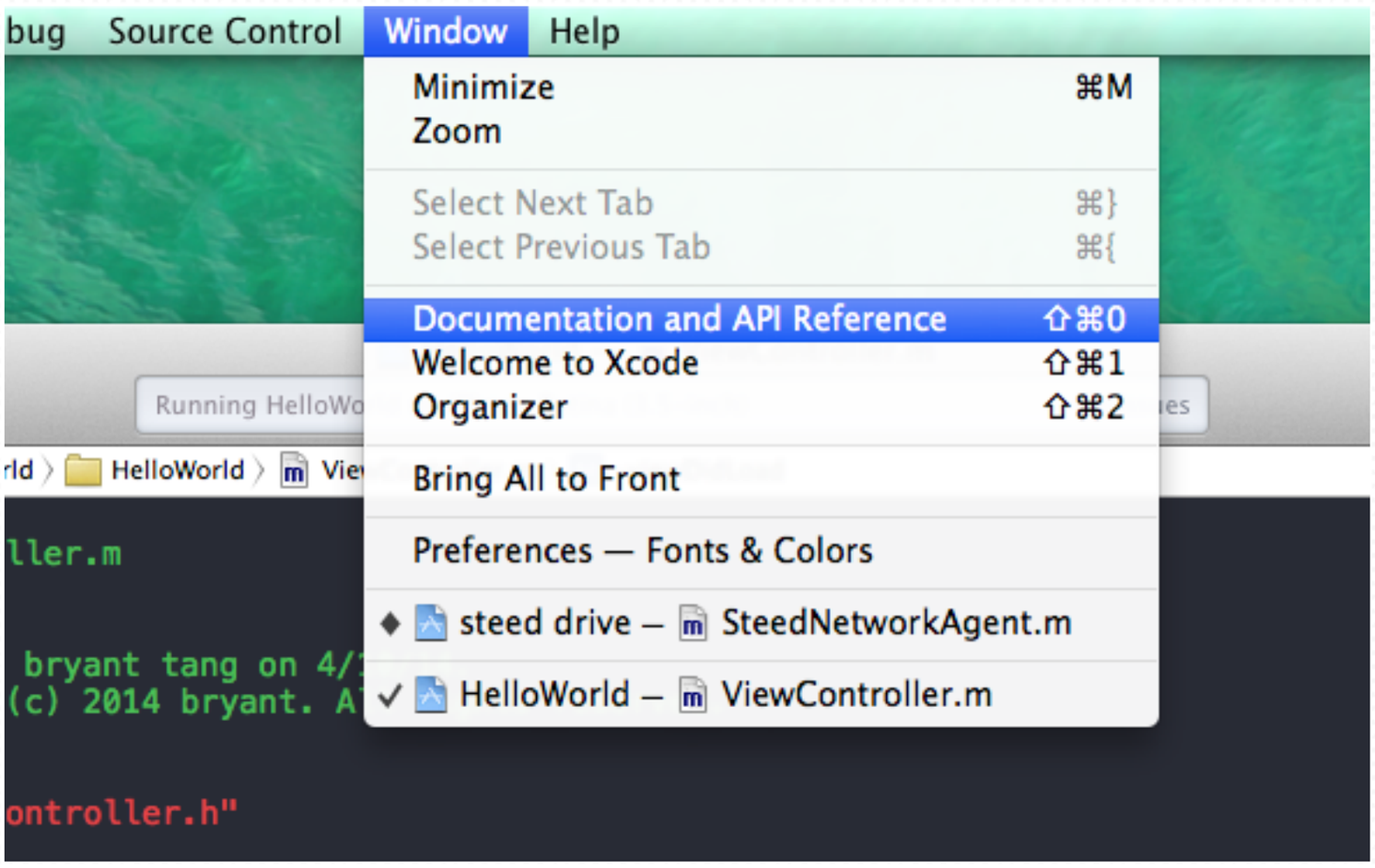
- Setting text color

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

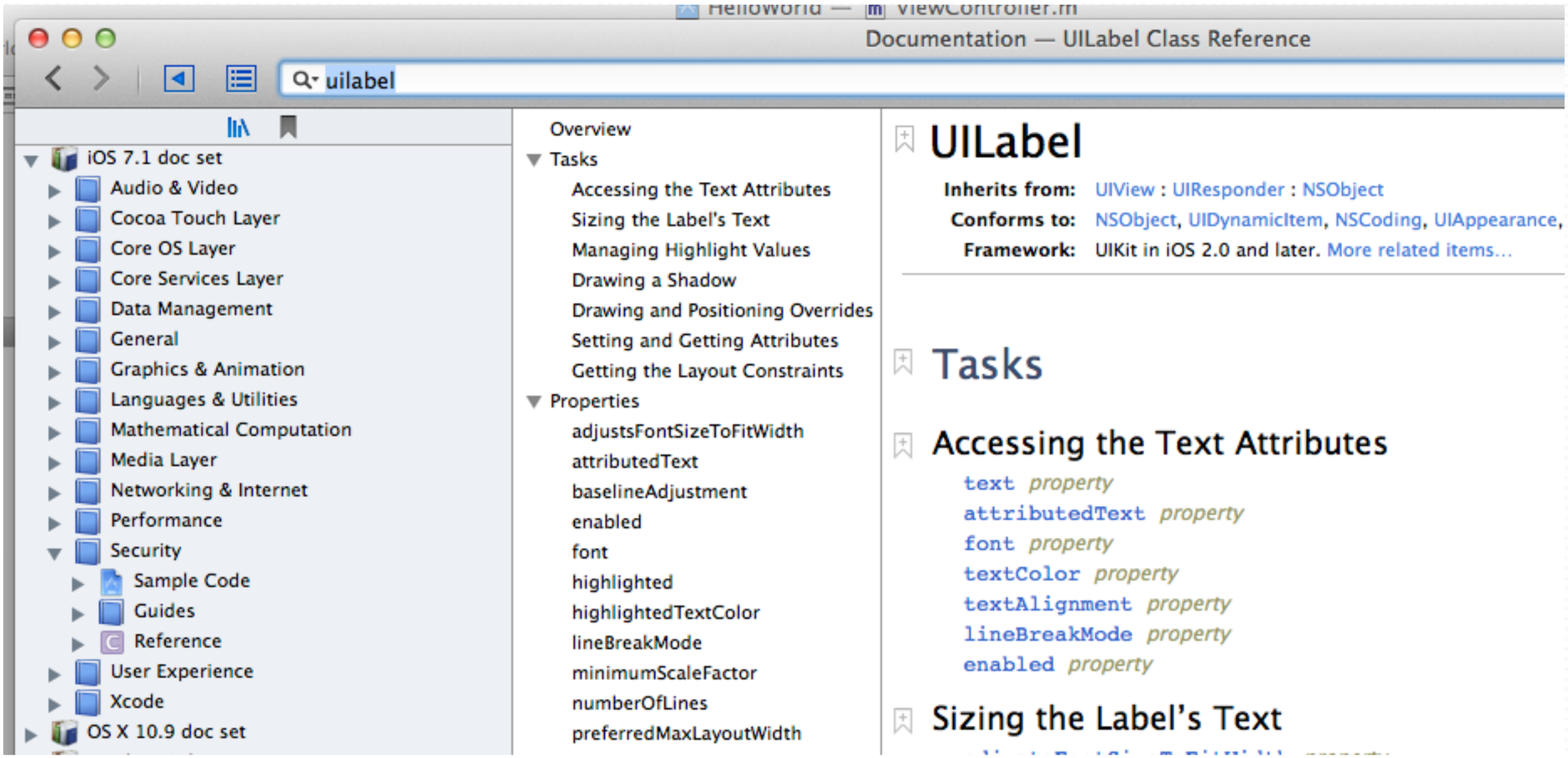
- Setting font size

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

Using UILabel - Look up Document



Using UILabel - Look up Document



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.

Delegates

Delegate Protocol

Boss

To meet a client

To write a report

Delegate

Assistant

To hire a manage

To fire a work

Delegate

Assistant

Delegate Protocol

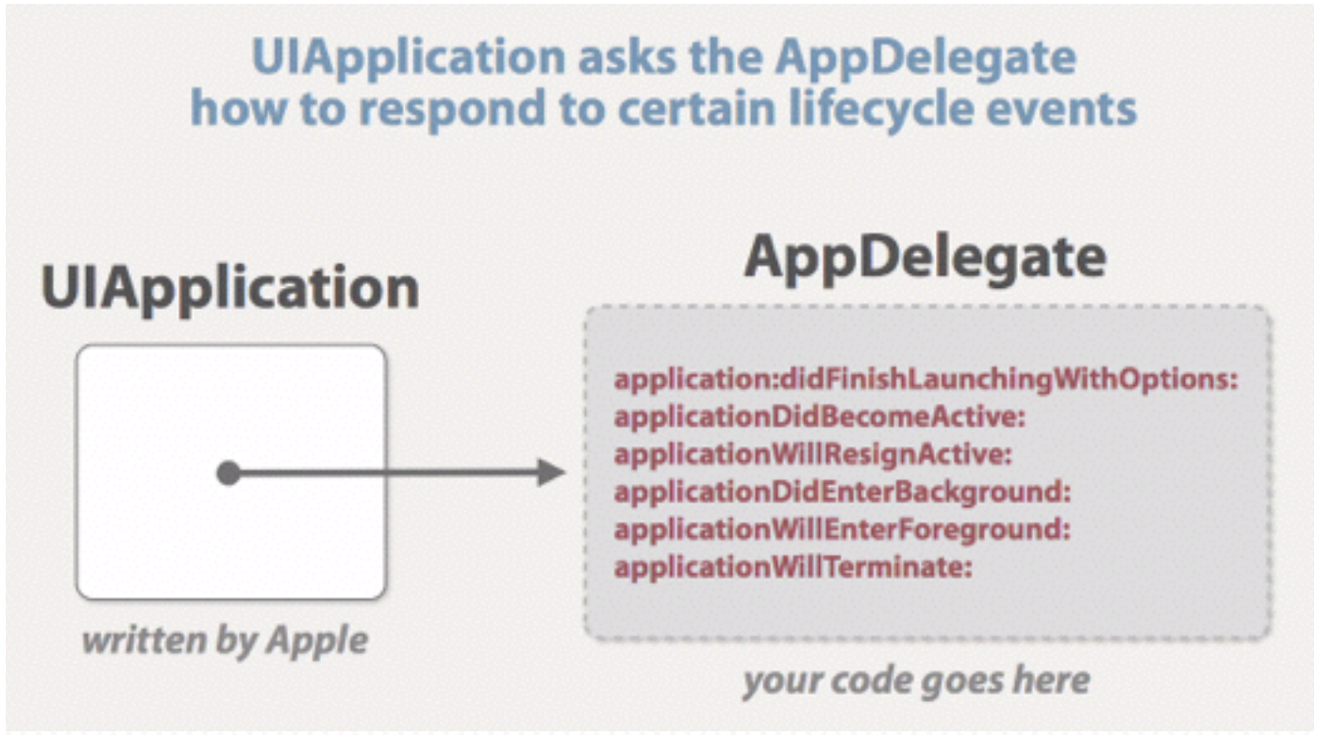
- Define a collection of methods.
- These methods act as communication bridge between two classes.
- The original class pass the task to the delegate to execute.
- We use delegate to customise behaviour.

Delegate Protocol

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

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

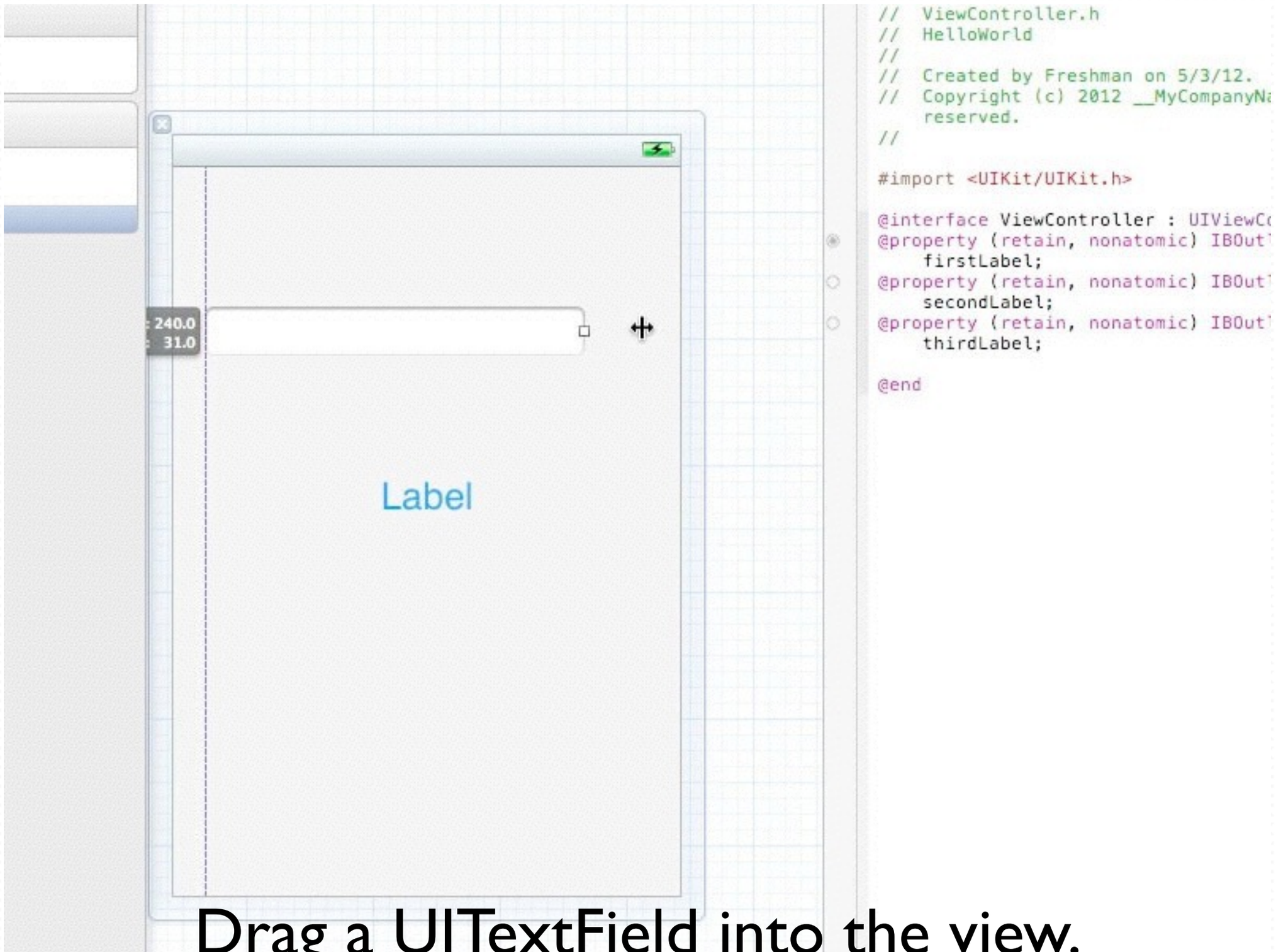
Delegate Protocol



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

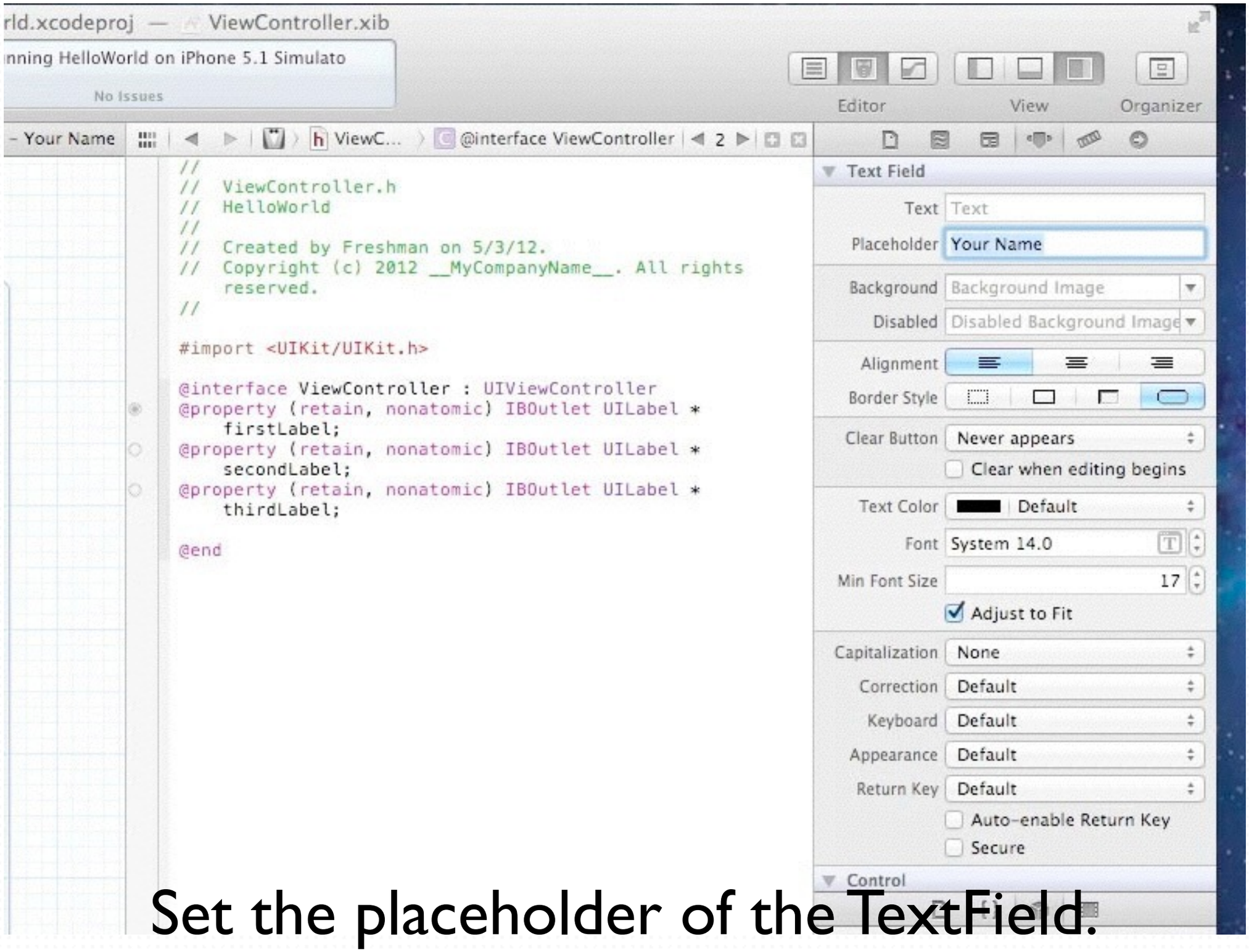
Using UITextField

Using UITextField



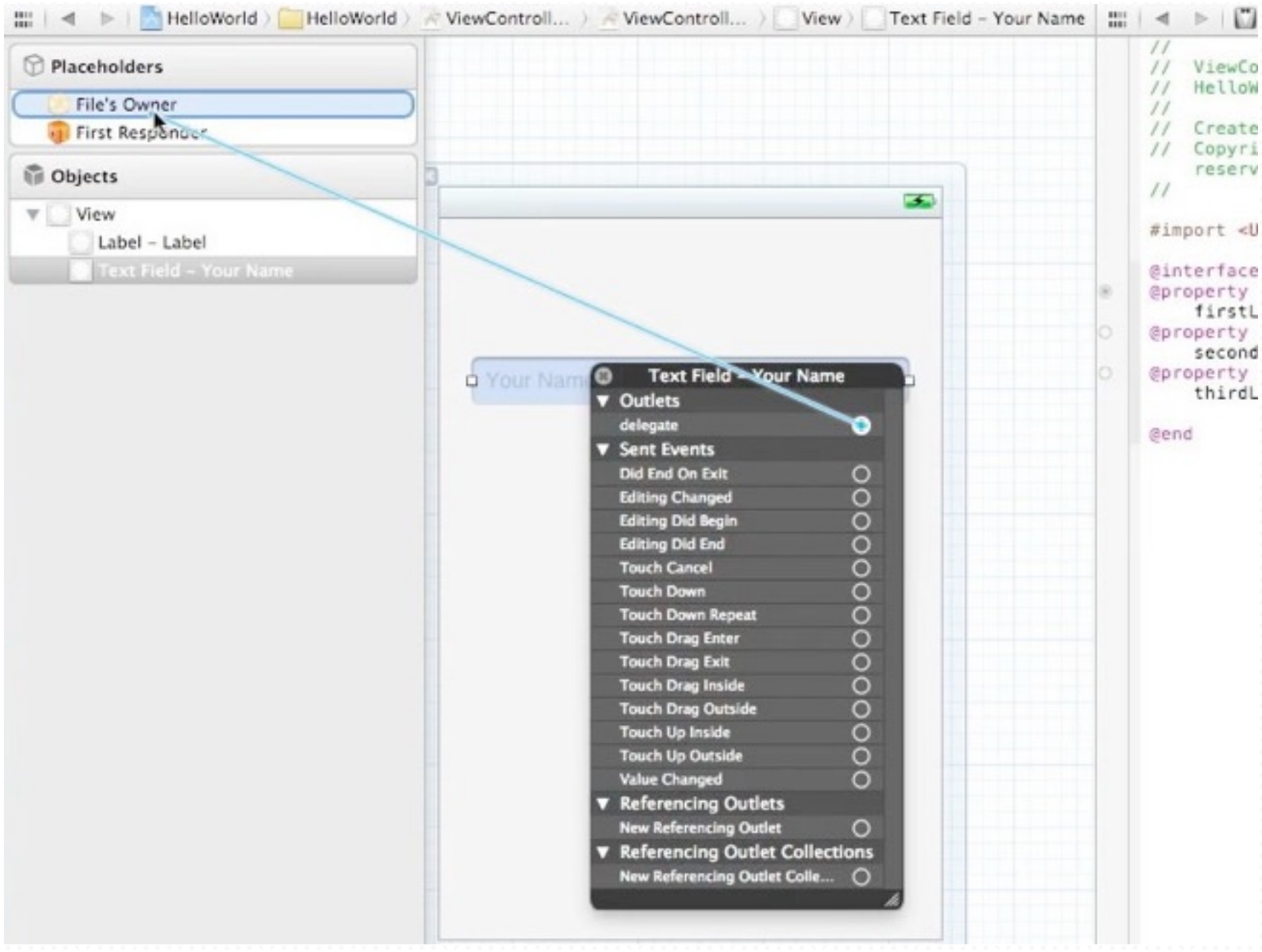
Drag a UITextField into the view.

Using UITextField



Set the placeholder of the TextField.

Using UITextField



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

UITextField & Events

Events

Press return->textFieldShouldReturn:

UITextField

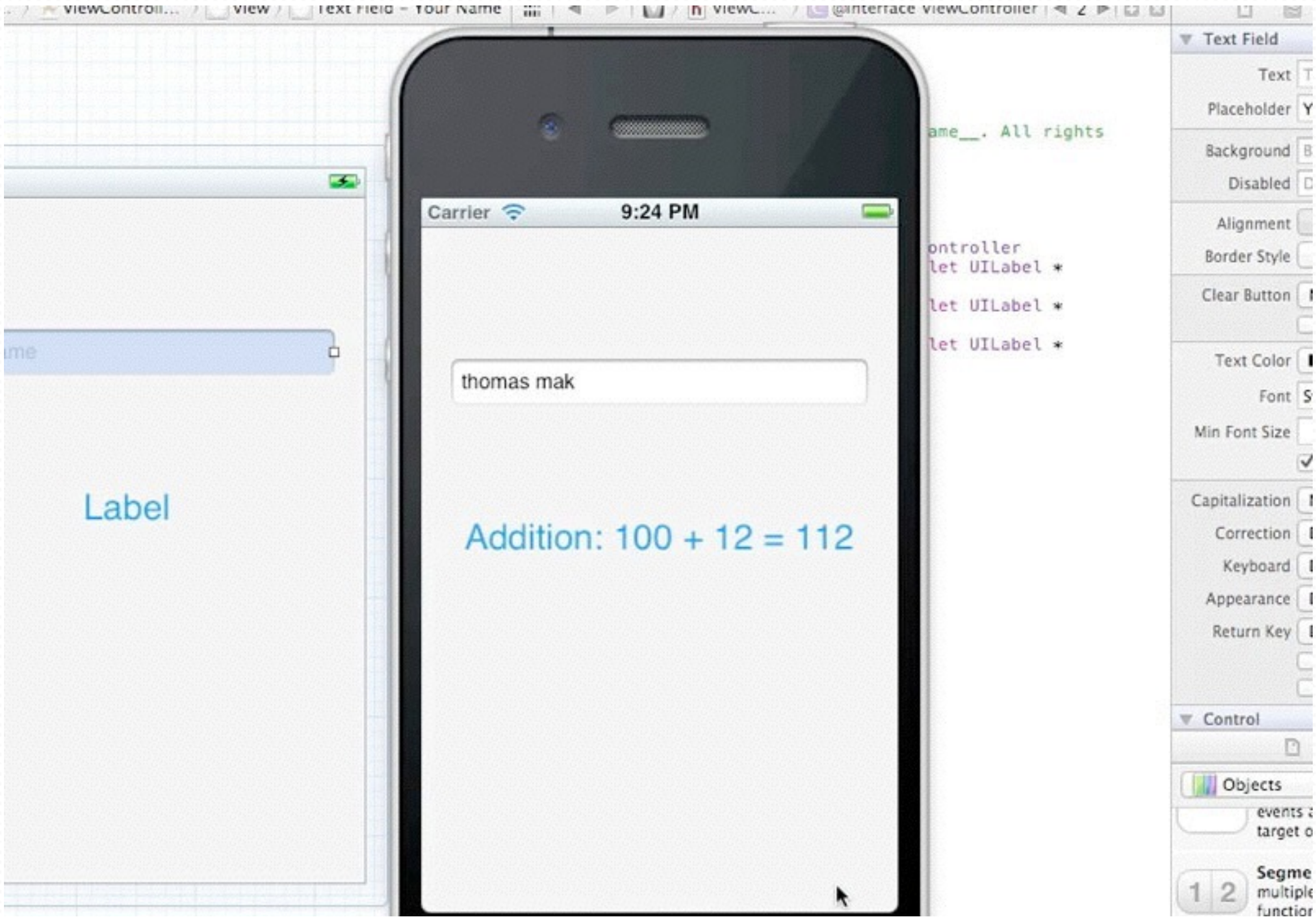
Delegate

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



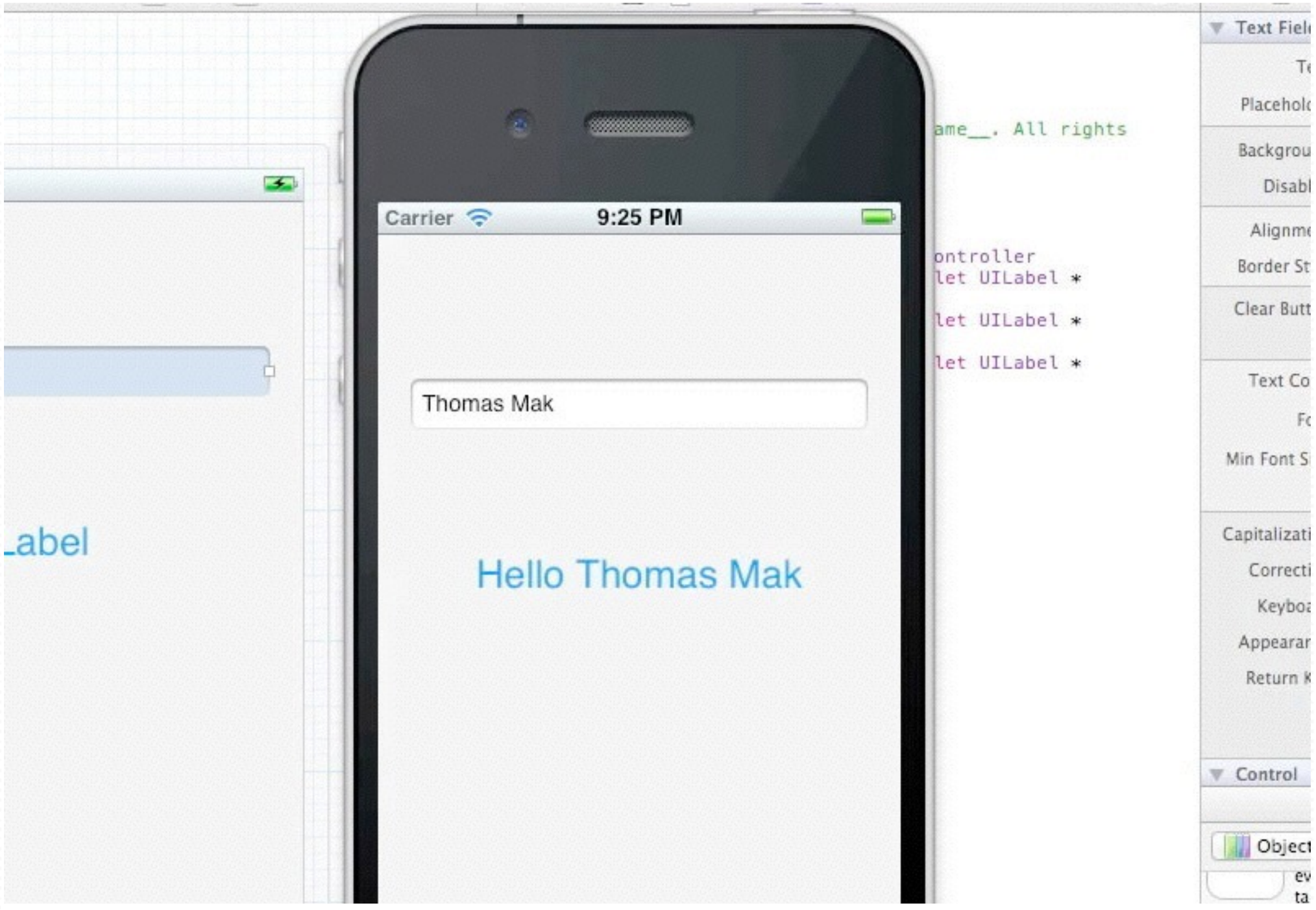
Let's test it in Simulator

Using UITextField

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

Add the highlighted code into the delegate method.

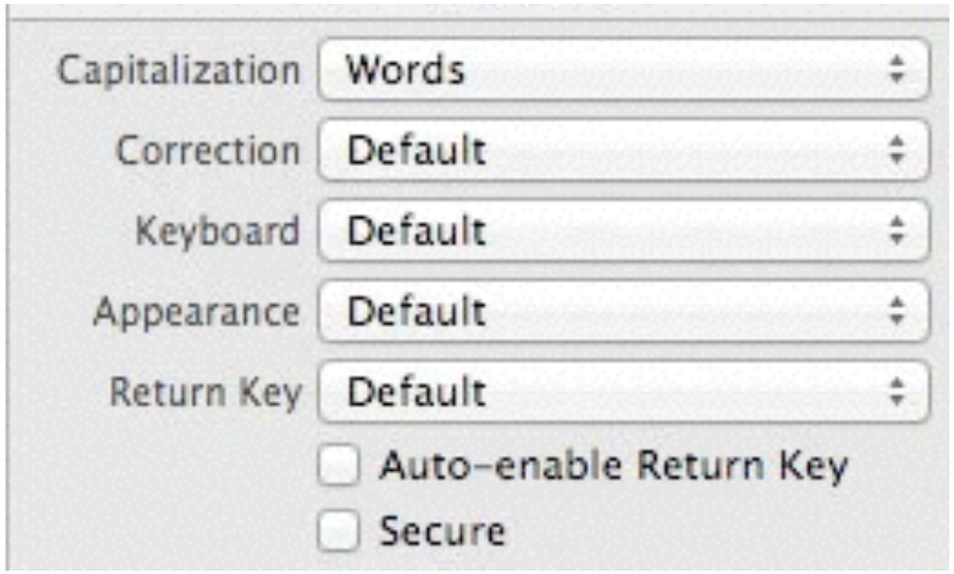
Using UITextField



Let’s test it in Simulator

Using UITextField

- Capitalization
- Correction
- Keyboard
- Appearance
- Return Key



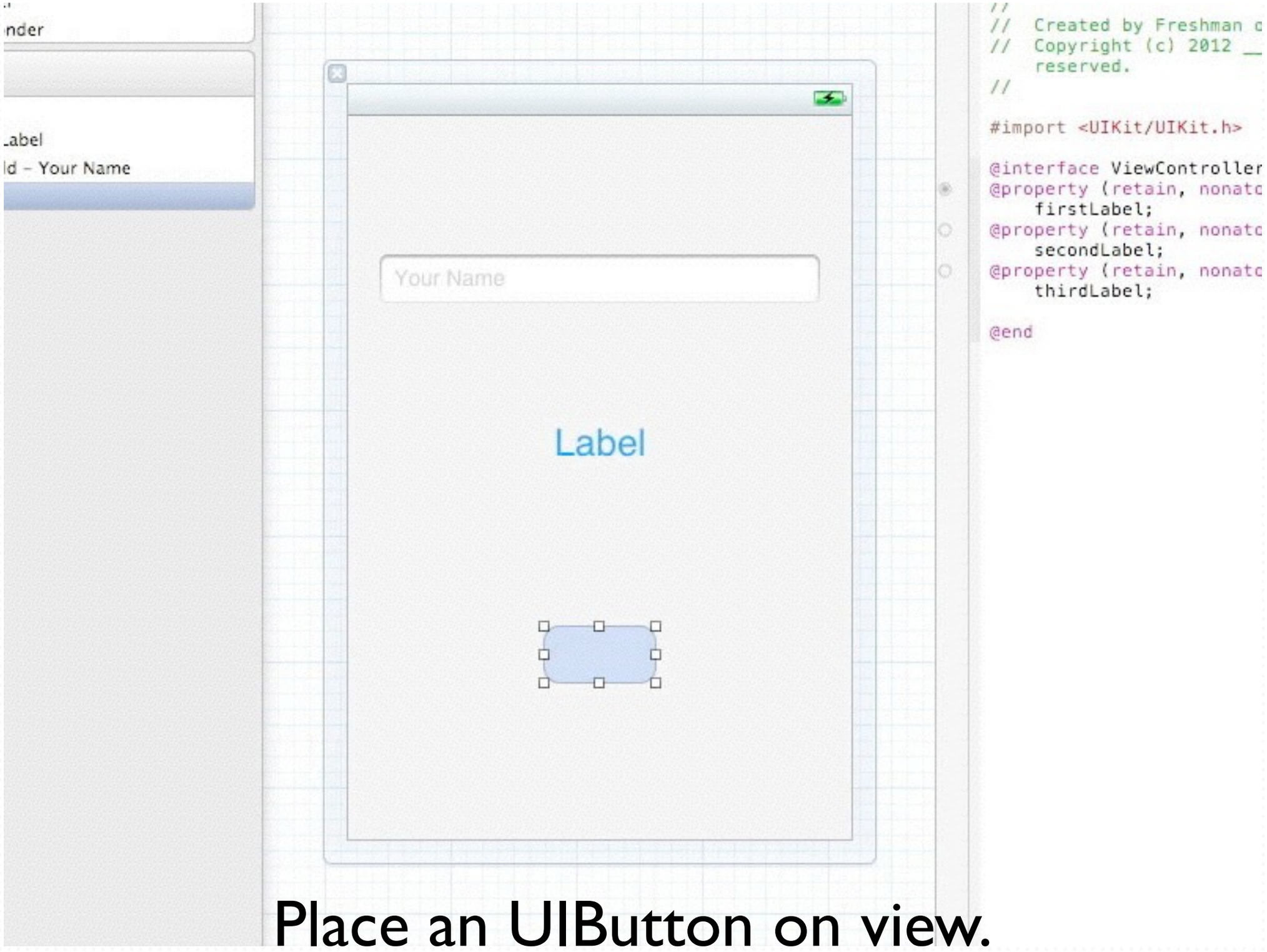
Using UITextField

- Return Key

Graph from TapWorthy book.

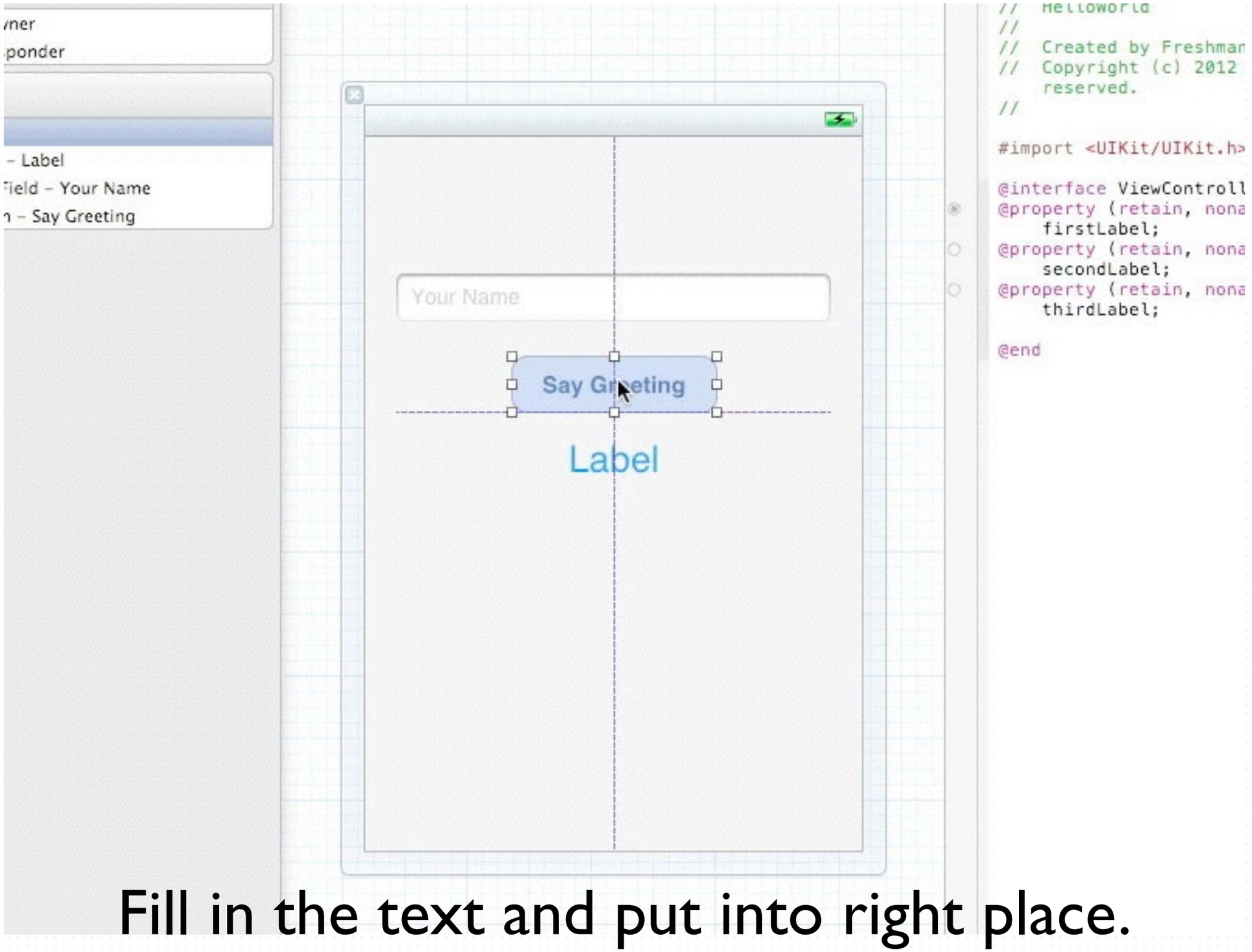
Using UIButton

Using UIButton



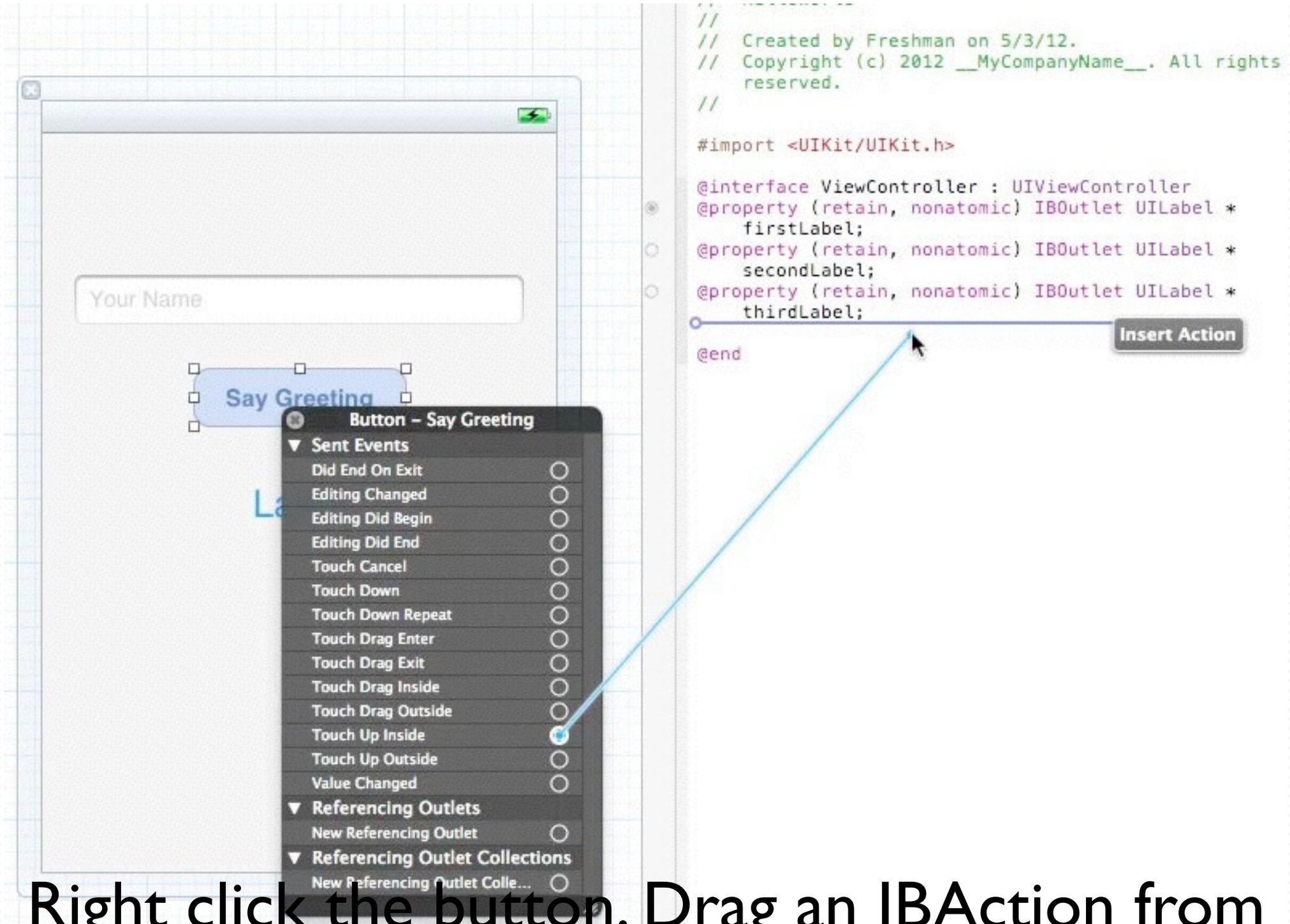
Place an UIButton on view.

Using UIButton



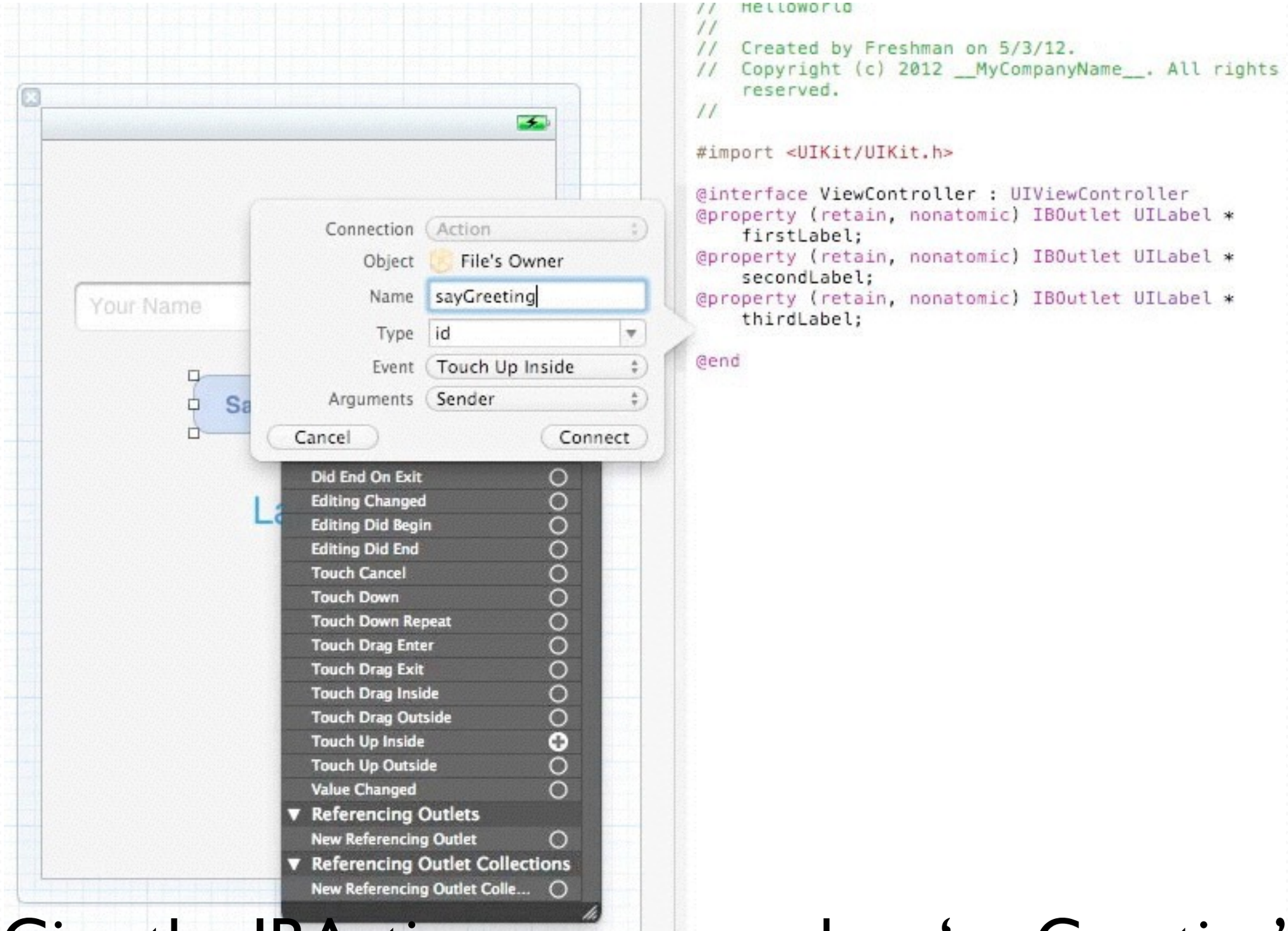
Fill in the text and put into right place.

Using UIButton



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

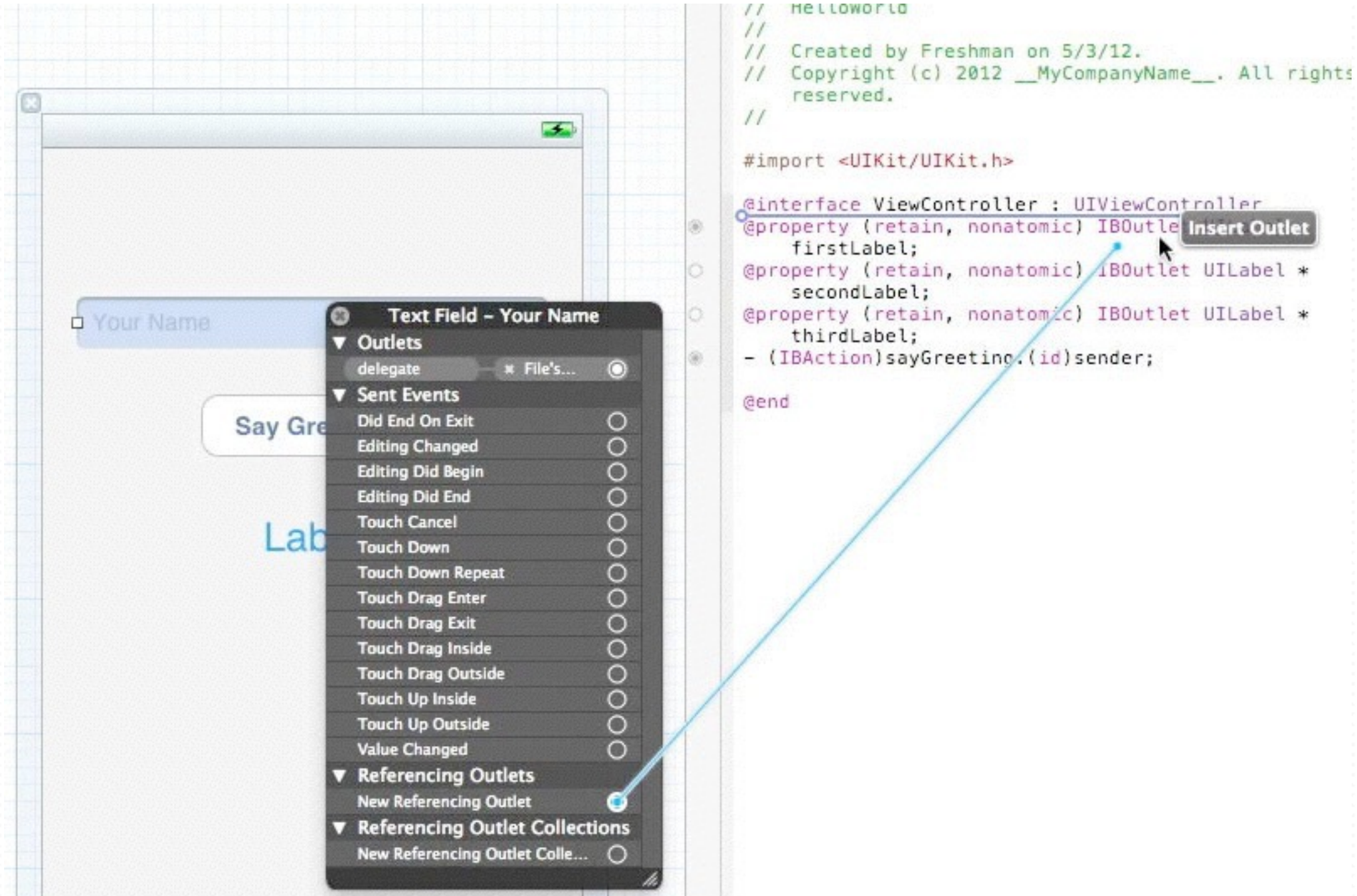
Using UIButton




```
//  Helloworld
//
//  Created by Freshman on 5/3/12.
//  Copyright (c) 2012 __MyCompanyName__. All rights
    reserved.
//

#import <UIKit/UIKit.h>

@interface ViewController : UIViewController
@property (retain, nonatomic) IBOutlet UILabel *
    firstLabel;
@property (retain, nonatomic) IBOutlet UILabel *
    secondLabel;
@property (retain, nonatomic) IBOutlet UILabel *
    thirdLabel;

@end
```

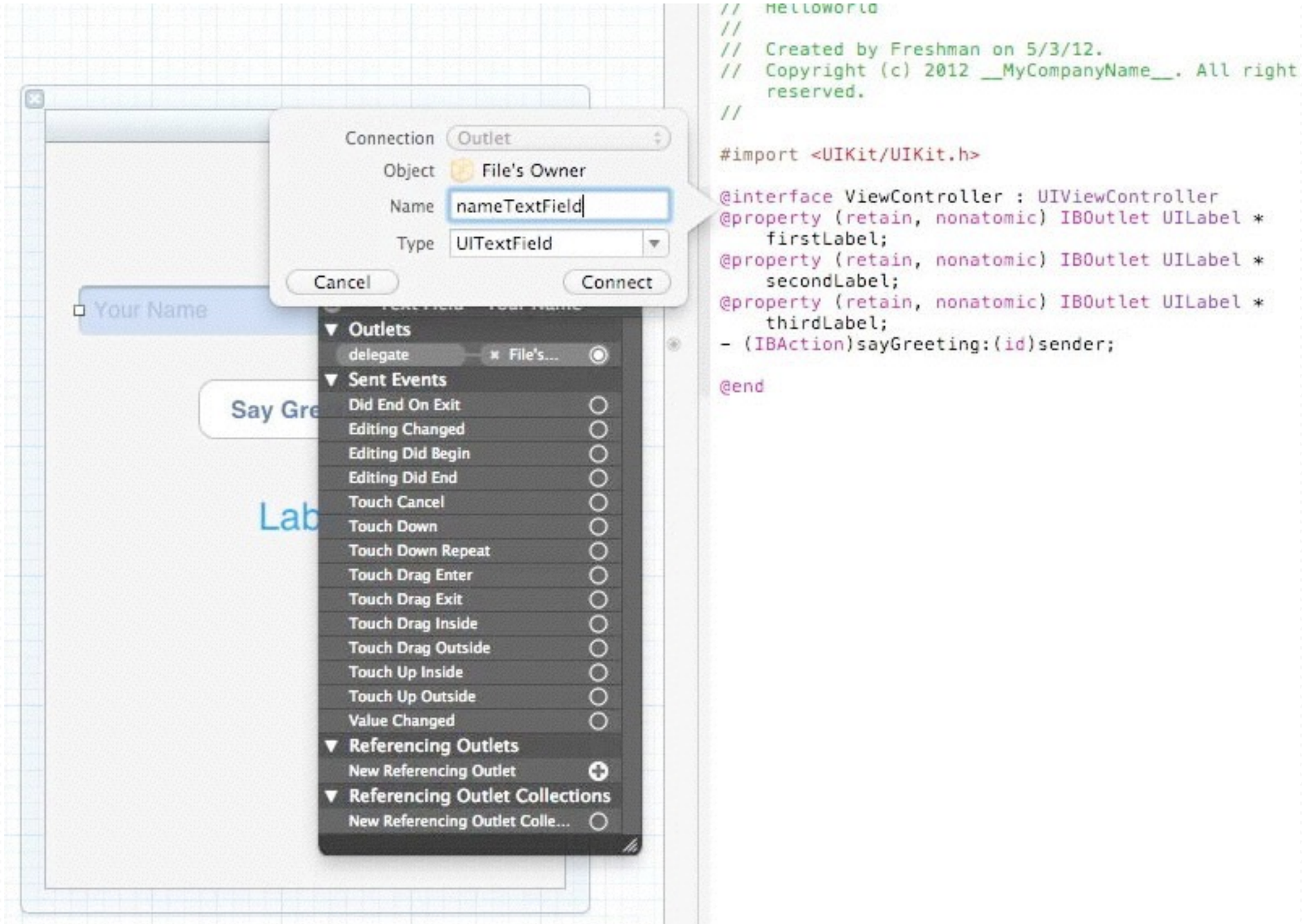
Give the IBAction a name, such as ‘sayGreeting’

Using UIButton



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

Using UIButton




```
//  HelloWorld
//
//  Created by Freshman on 5/3/12.
//  Copyright (c) 2012 __MyCompanyName__. All right
    reserved.
//

#import <UIKit/UIKit.h>

@interface ViewController : UIViewController
@property (retain, nonatomic) IBOutlet UILabel *
    firstLabel;
@property (retain, nonatomic) IBOutlet UILabel *
    secondLabel;
@property (retain, nonatomic) IBOutlet UILabel *
    thirdLabel;
- (IBAction)sayGreeting:(id)sender;

@end
```

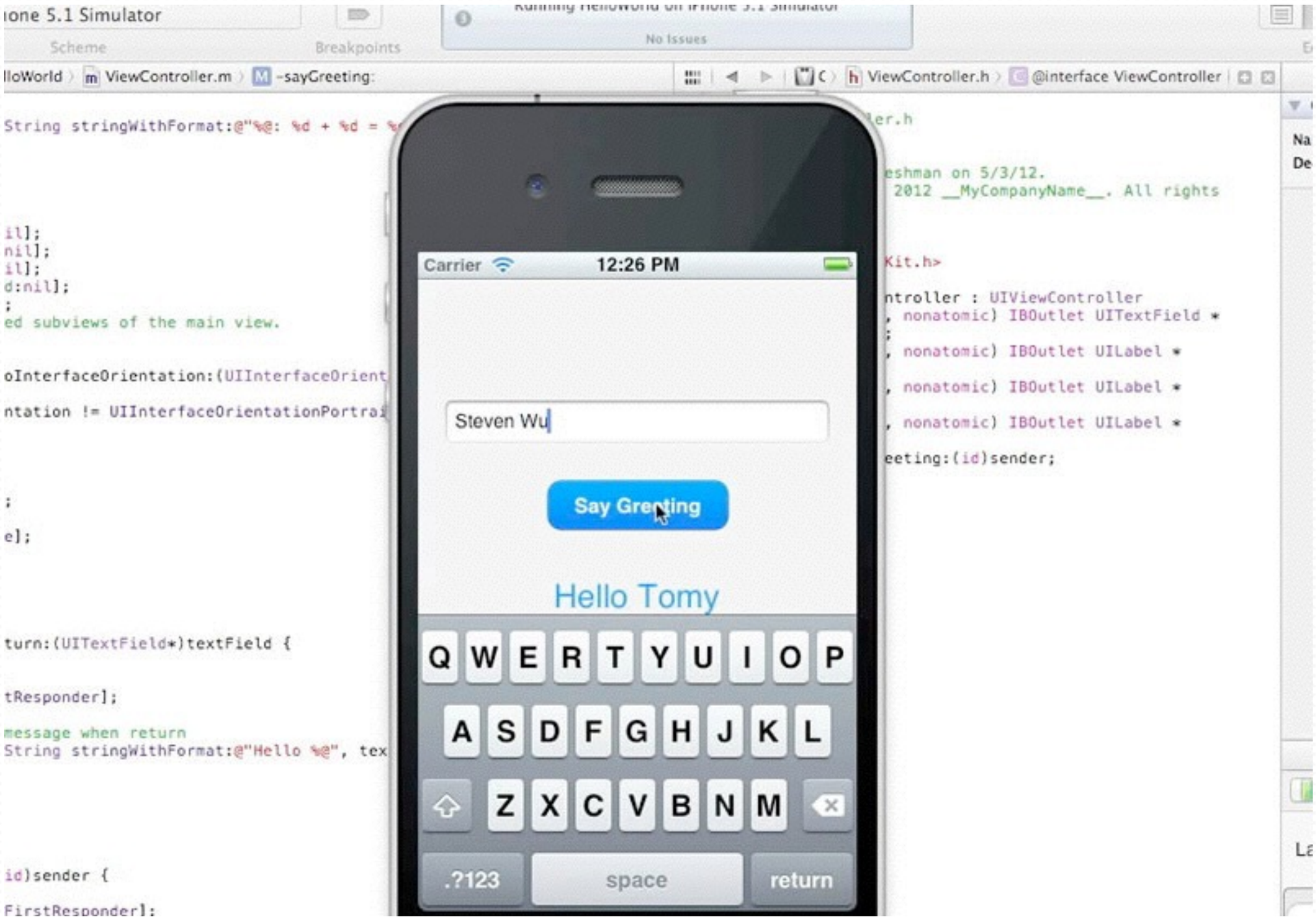
Name the IBOutlet ‘nameTextField’

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
```

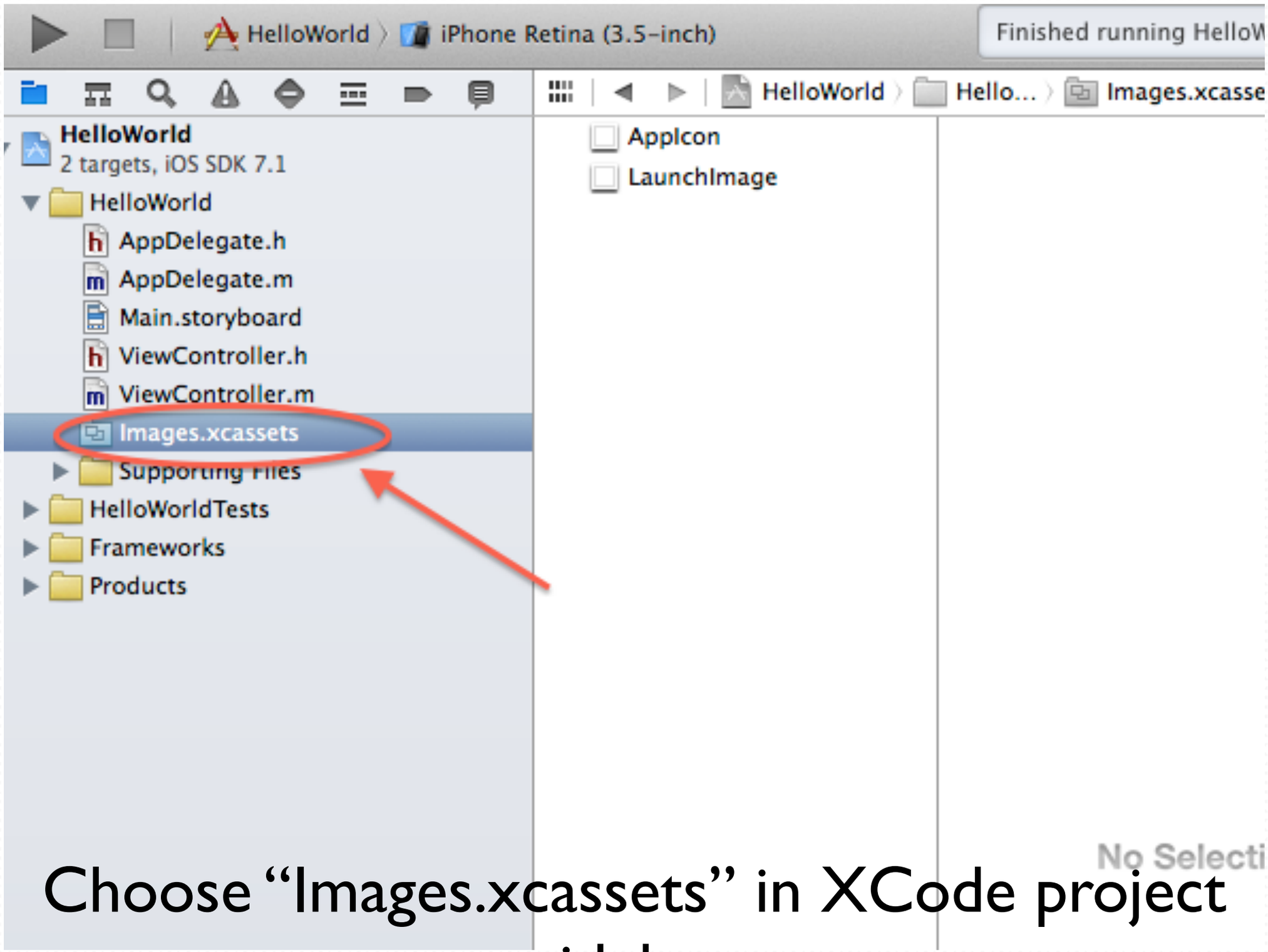
Using Image in iOS

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

Using Image in iOS

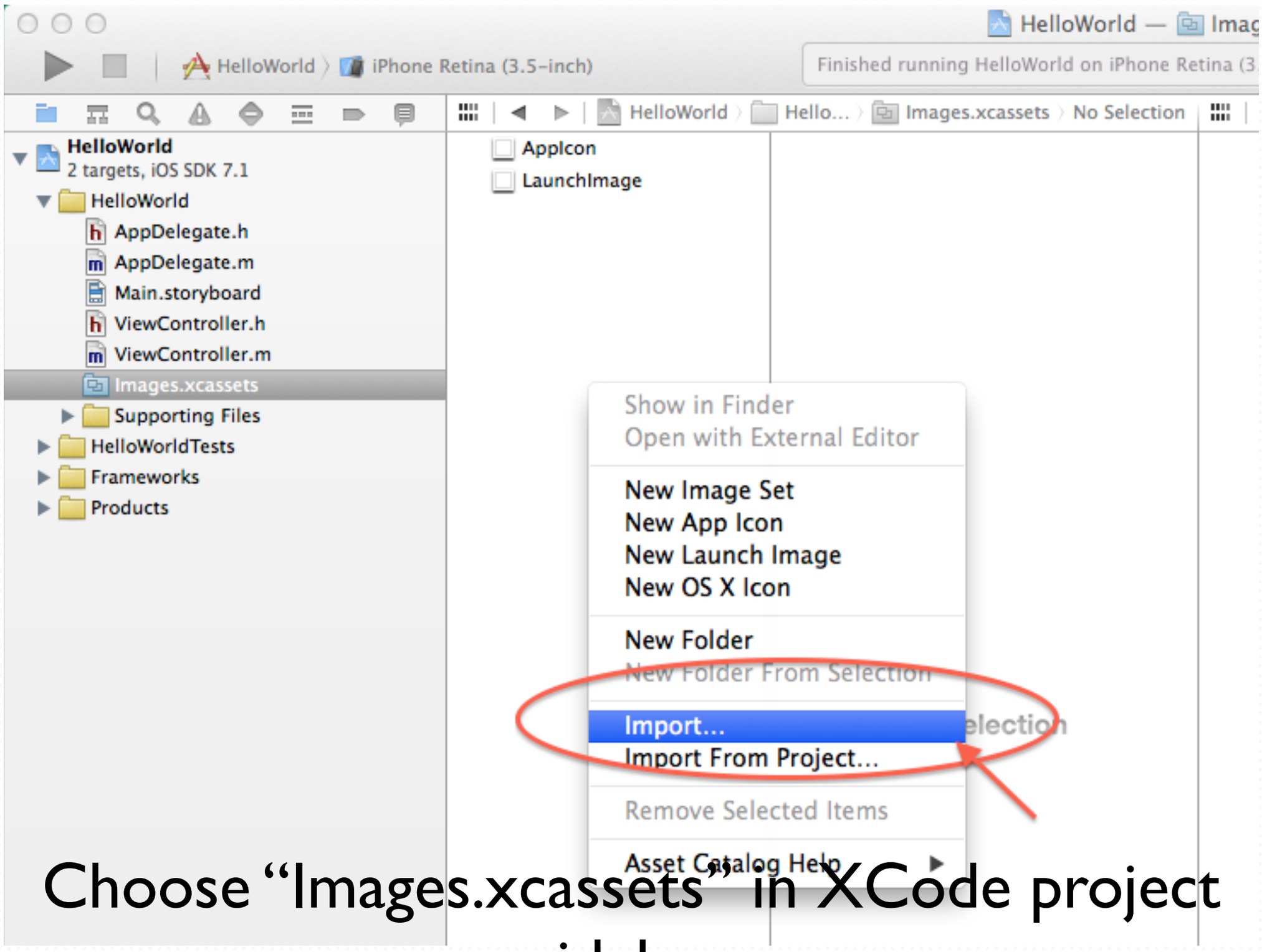
button@2x.png
474 x 80

Images in UIButton



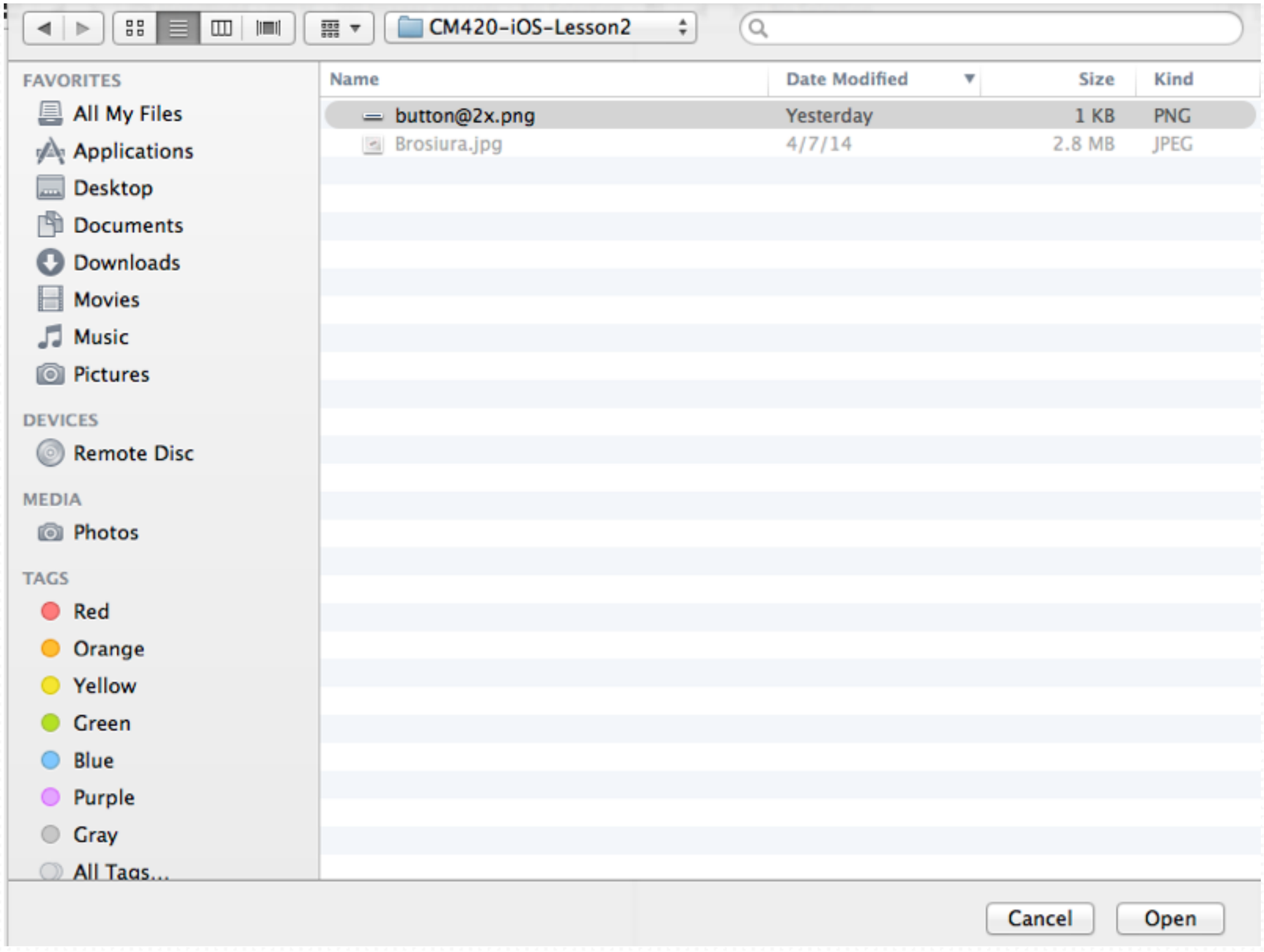
Choose “Images.xcassets” in XCode project sidebar.

Images in UIButton

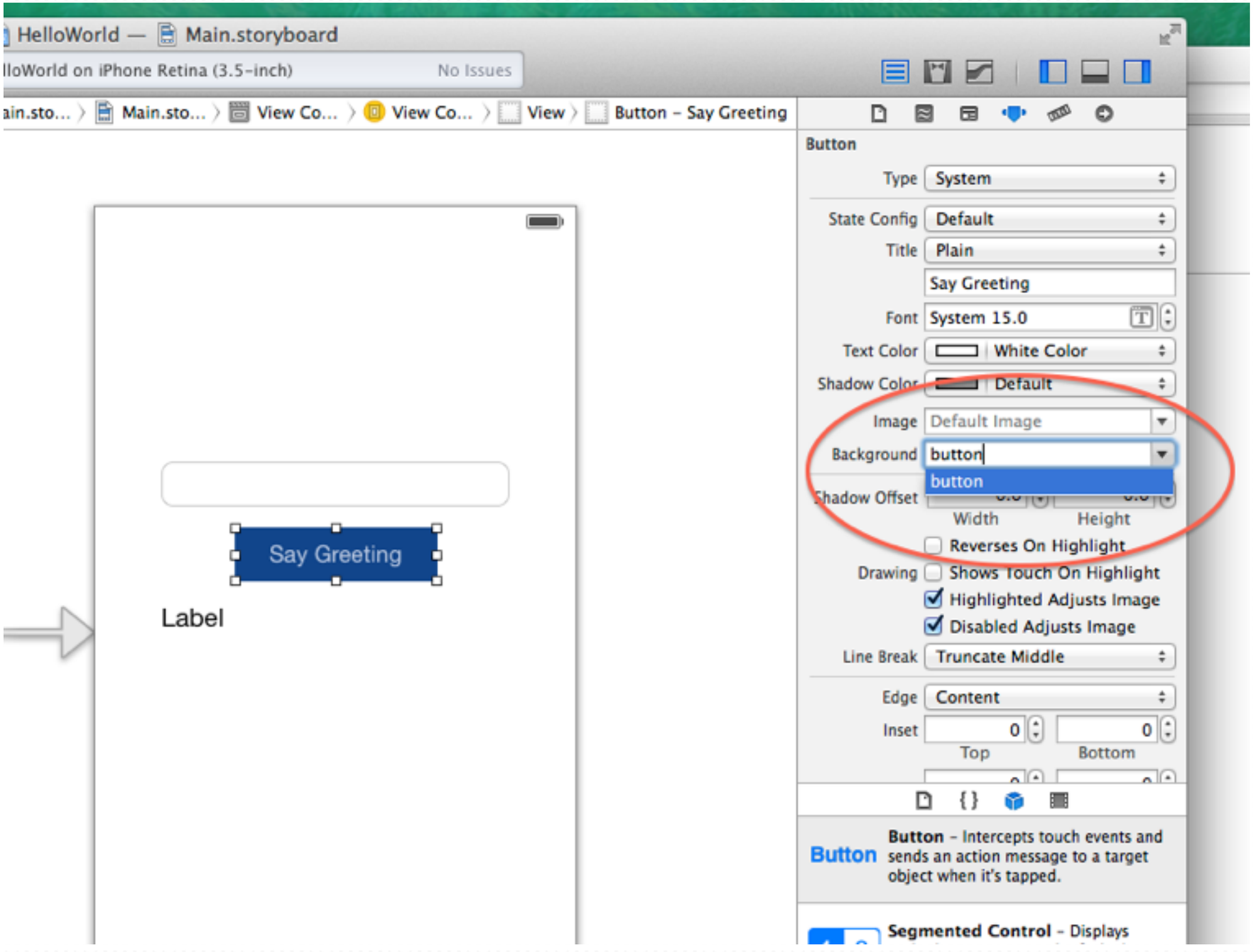


Choose “Images.xcassets” in XCode project sidebar.

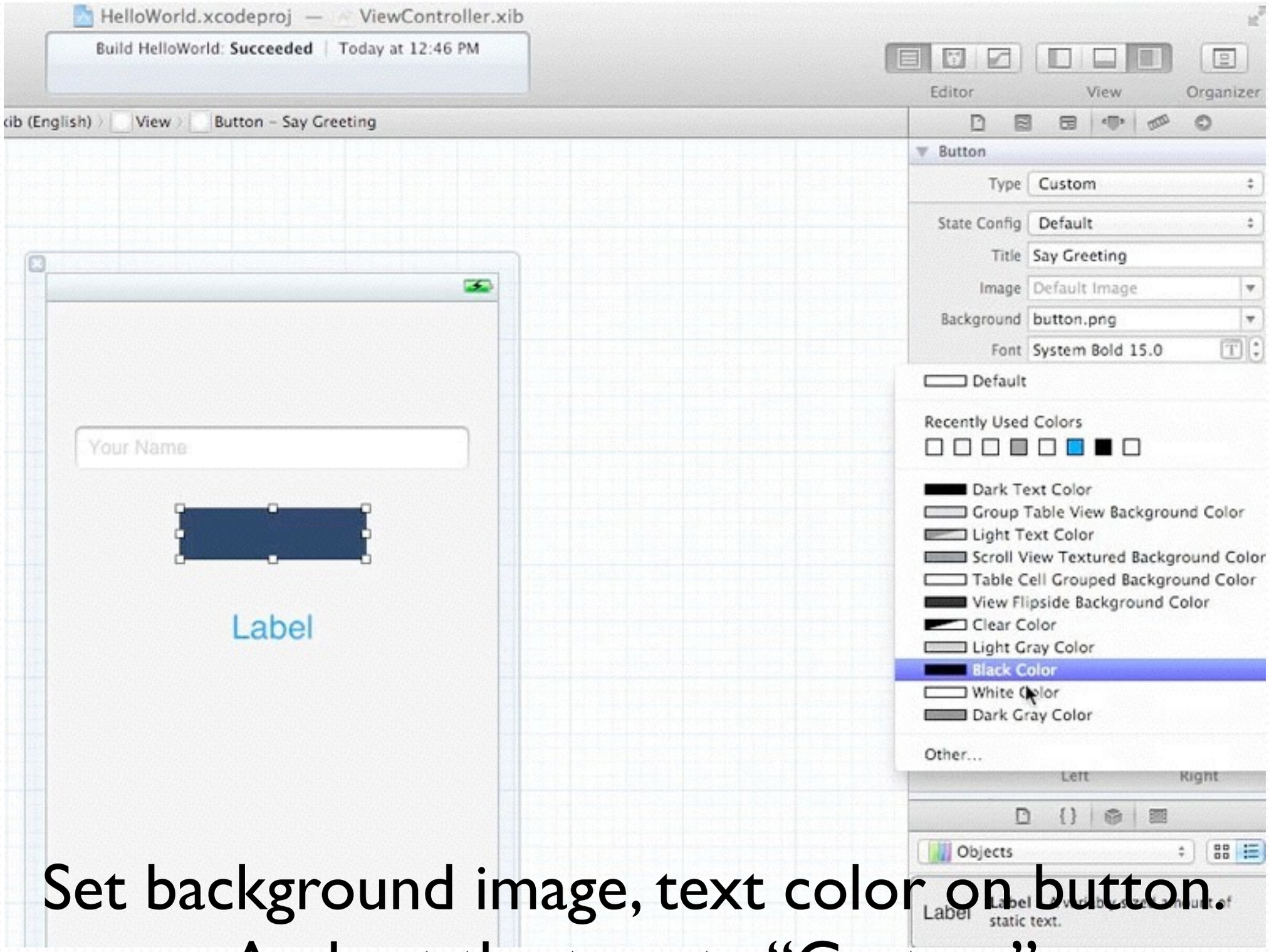
Images in UIButton



Images in UIButton

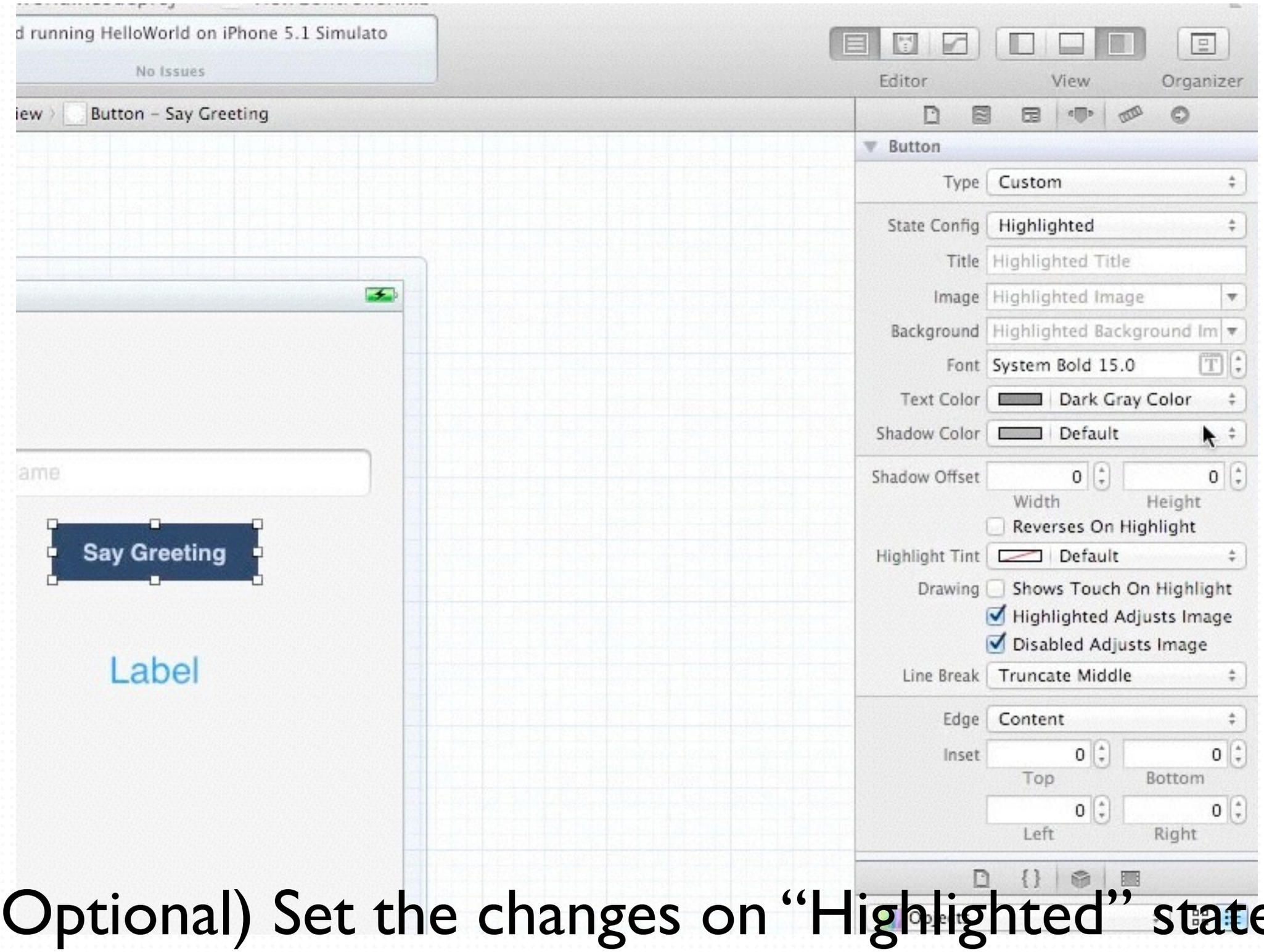


Images in UIButton



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

Images in UIButton



(Optional) Set the changes on “Highlighted” state.

Homework

- ✓ Design an Converter with more the three exchange.
  (MOP -> US, MOP -> HKD, MOP-> CNY)
- ✓ Present it to the class in next lesson.