

CS121 Tutorial 1

This is the first ios dev tutorial.

Purple bubbles give you information you'll need to know.

Yellow Bubbles tell you what to do.

Orange bubbles tell you what you're not expected to understand yet. 😊

Launch Xcode then click
here to start a new
project.

Welcome to Xcode

Version 4.2 (4C199)



Create a new Xcode project

Start building a new Mac, iPhone or iPad application from one of the included templates



Connect to a repository

Use Xcode's integrated source control features to work with your existing projects



Learn about using Xcode

Explore the Xcode development environment with the Xcode 4 User Guide



Go to Apple's developer portal

Visit the Mac and iOS Dev Center websites at developer.apple.com

Recents

- lab0
~/Desktop/iosProjects
- Keypad
~/Desktop/iosProjects
- tempConverter
~/Desktop/iosProjects
- ZSudoku
~/Desktop/iosProjects
- Blocker
~/Desktop/iosProjects
- Stars
~/Desktop/iosProjects
- zBlock
~/Desktop/iosProjects
- BlockerGame
~/Desktop/iosProjects
- cocos2d-ios
~/Desktop/cocos2d-iphone-2.0
- ScrGame
~/Desktop/iosProjects

Last opened Today 11:06 AM

Open Other...

Show this window when Xcode launches

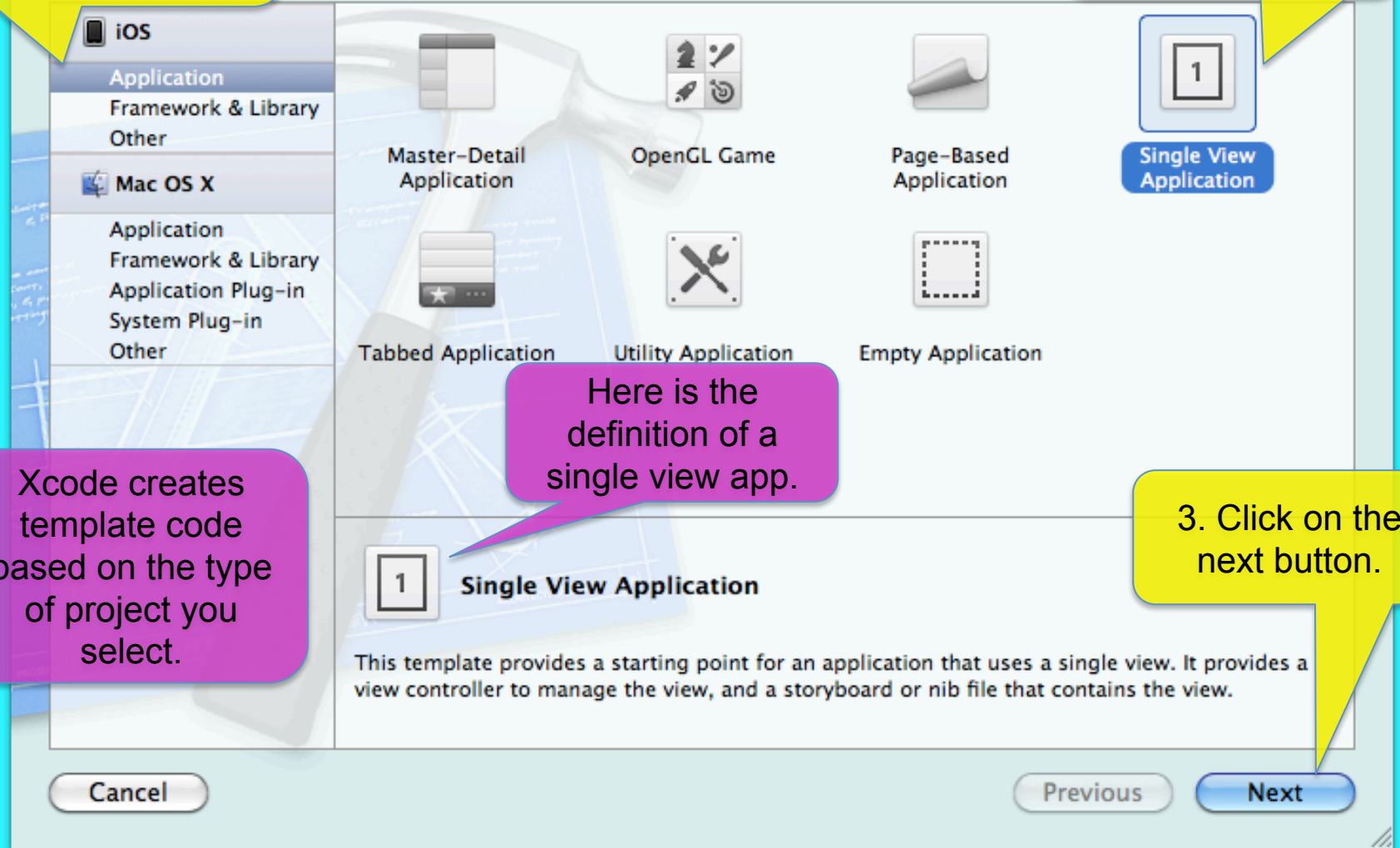
Cancel

Open

1. We are going to build an iOS application so click here.

2. We'll build a single view application, so click here.

Click a template for your new project:



Xcode creates template code based on the type of project you select.

Here is the definition of a single view app.

3. Click on the next button.

1. Name your product HW 1.

2. Use com. followed by your first initial and last name.

3. Use HW1 as your "class prefix".

Reversing the company's DNS lookup string is a common convention for producing this ID. It won't cause any problems if you don't own your own domain name.

Choose options for your new project:

Product Name

Company Identifier

Bundle Identifier

Class Prefix

Device Family

Use Storyboard

Use Automatic Reference Counting

Include Unit Tests

Cancel

Previous

Next

Choose options for your new project:

We can create apps for iPhone, iPad or both. We won't need much screen space for this app so iPhone is good enough.

1. Choose iPhone as the device.

Product Name
Company Identifier
Bundle Identifier
Class Prefix
Device Family

2. Make sure that Automatic Reference Counting (ARC) is the only block checked.

- Use Storyboard
- Use Automatic Reference Counting
- Include Unit Tests

iOS uses ARC in lieu of garbage collection. We'll talk more about this later.

Cancel

Previous

Next

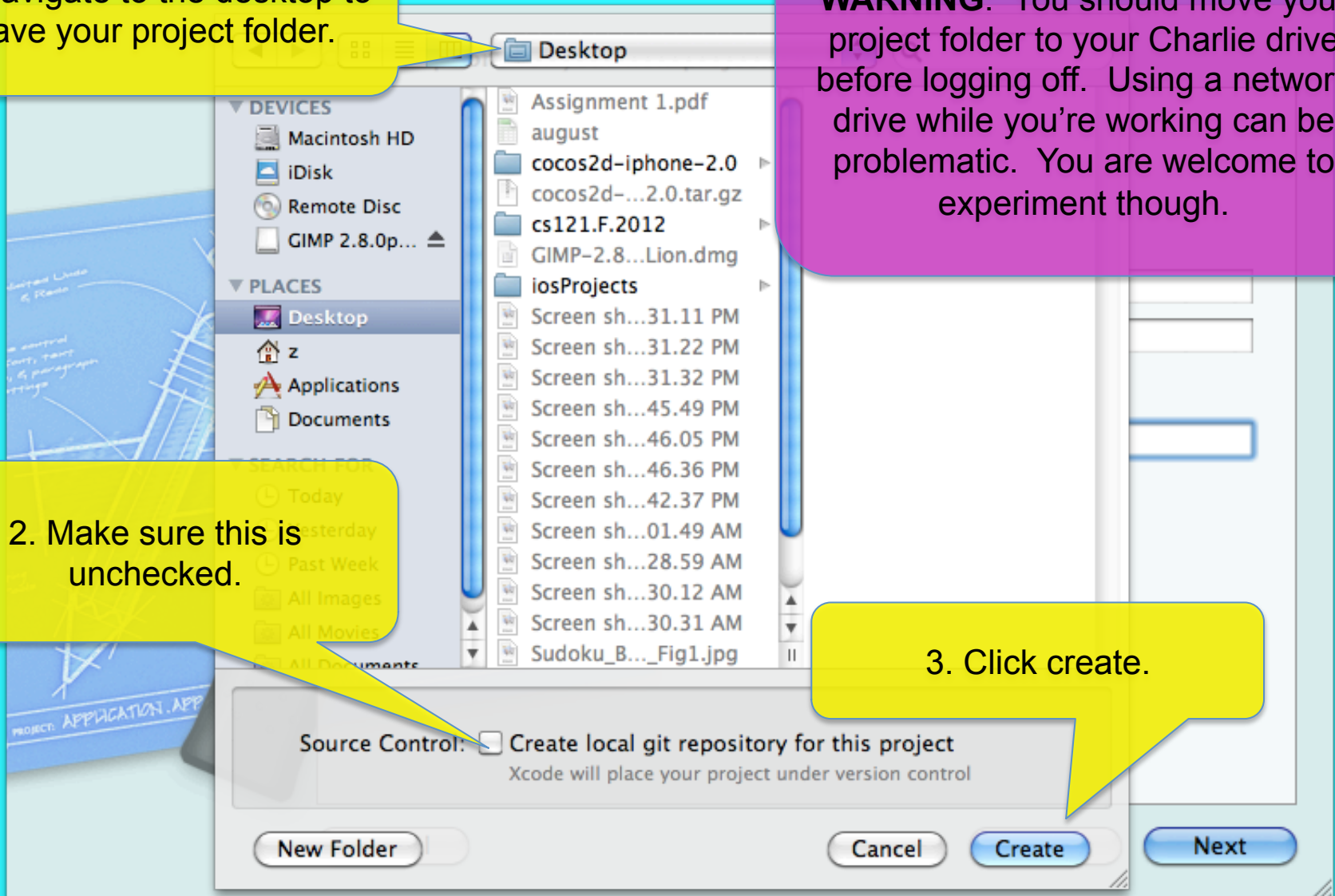
3. Click next.

1. Navigate to the desktop to save your project folder.

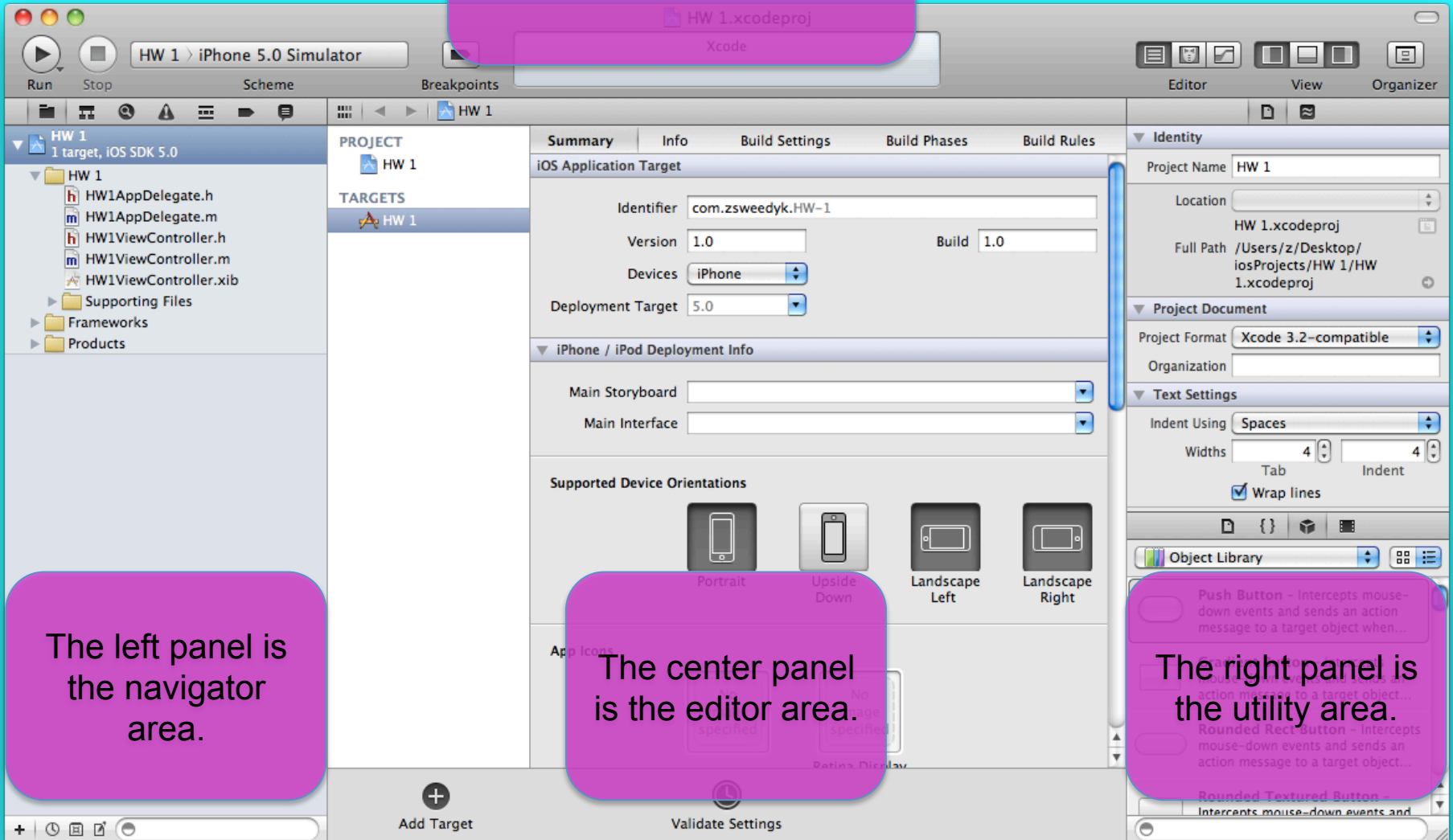
2. Make sure this is unchecked.

3. Click create.

WARNING: You should move your project folder to your Charlie drive before logging off. Using a network drive while you're working can be problematic. You are welcome to experiment though.



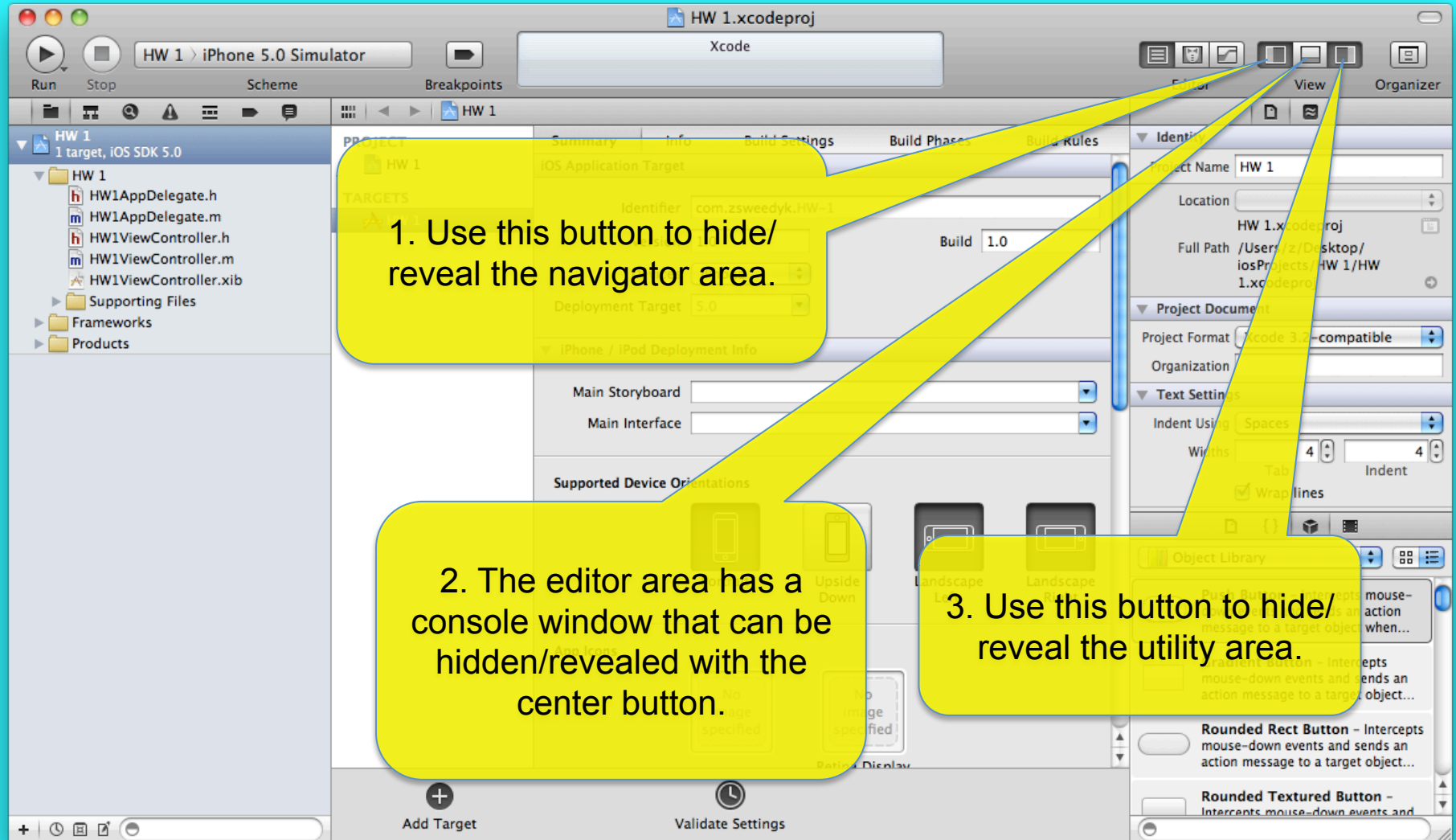
Woohoo! You've created your first iOS app.



The left panel is the navigator area.

The center panel is the editor area.

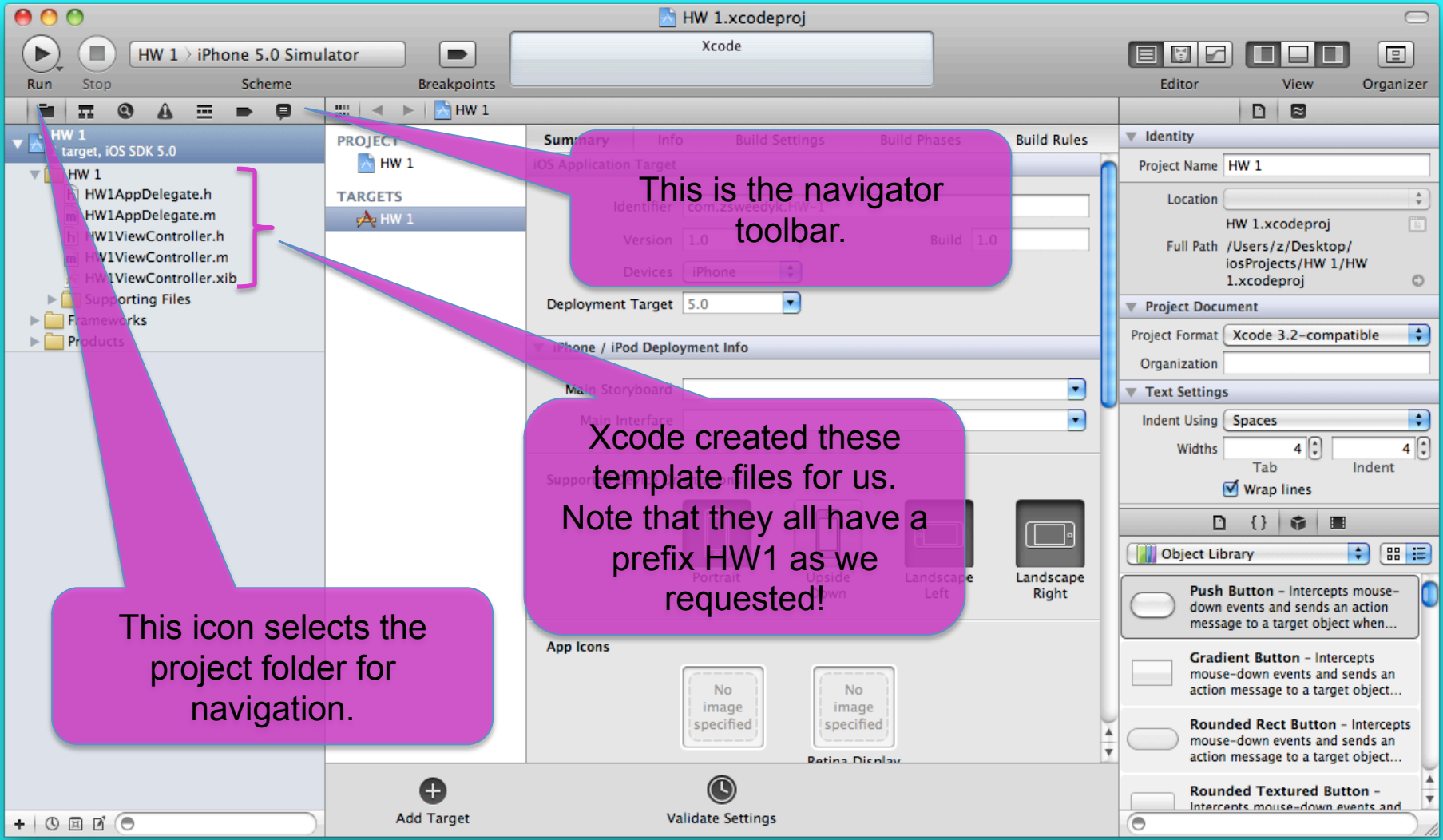
The right panel is the utility area.



1. Use this button to hide/reveal the navigator area.

2. The editor area has a console window that can be hidden/revealed with the center button.

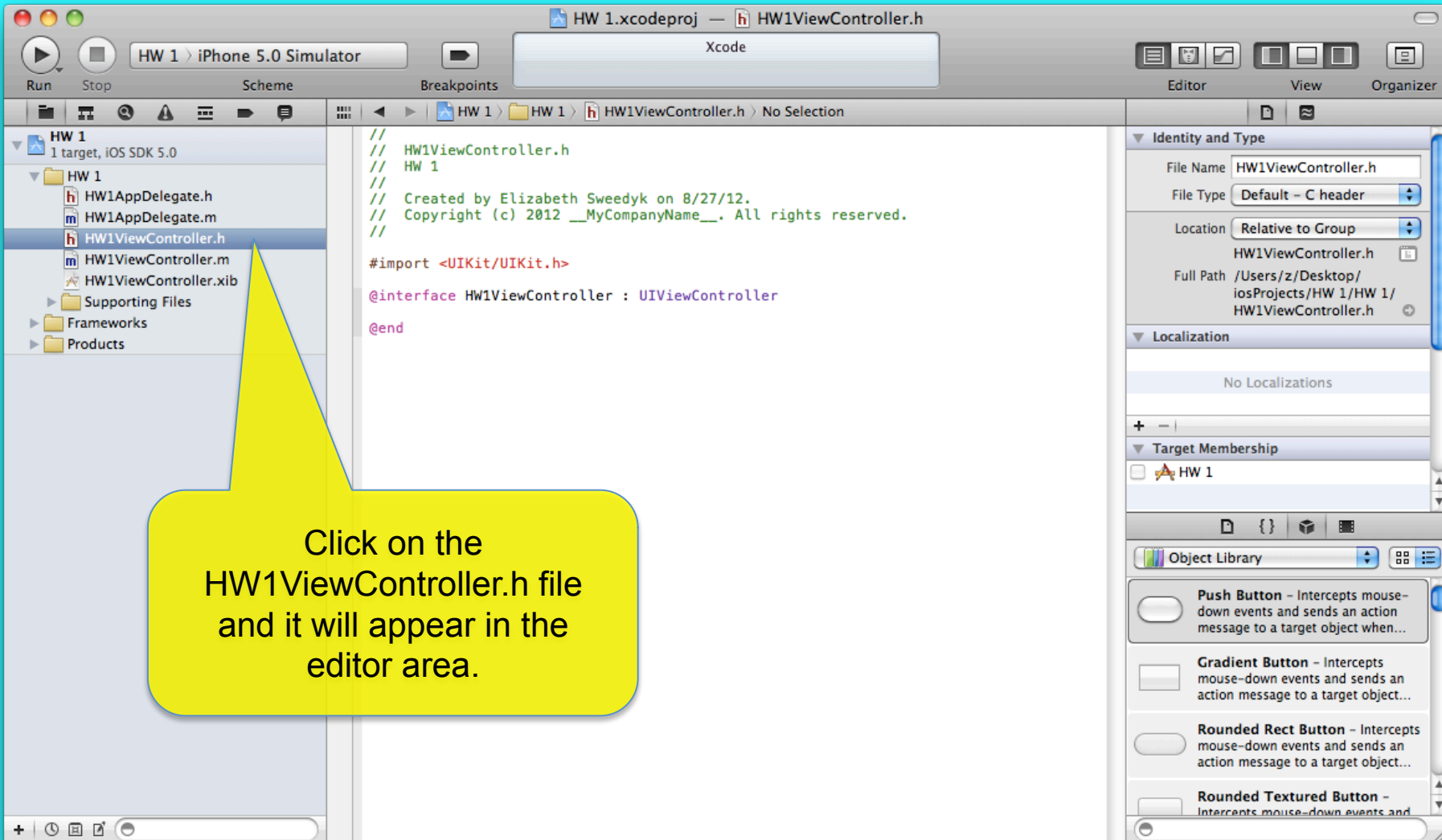
3. Use this button to hide/reveal the utility area.

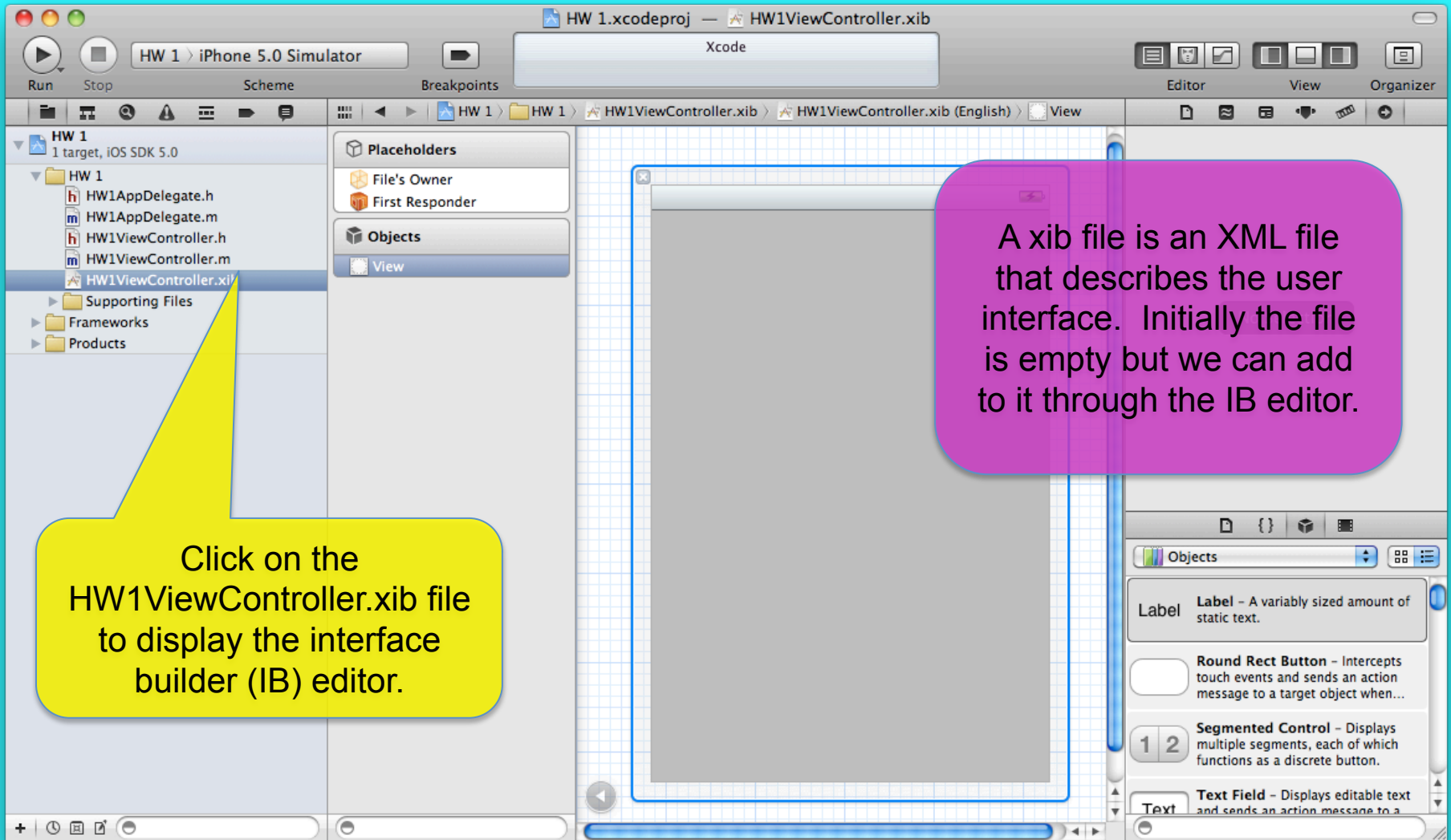


This is the navigator toolbar.

Xcode created these template files for us. Note that they all have a prefix HW1 as we requested!

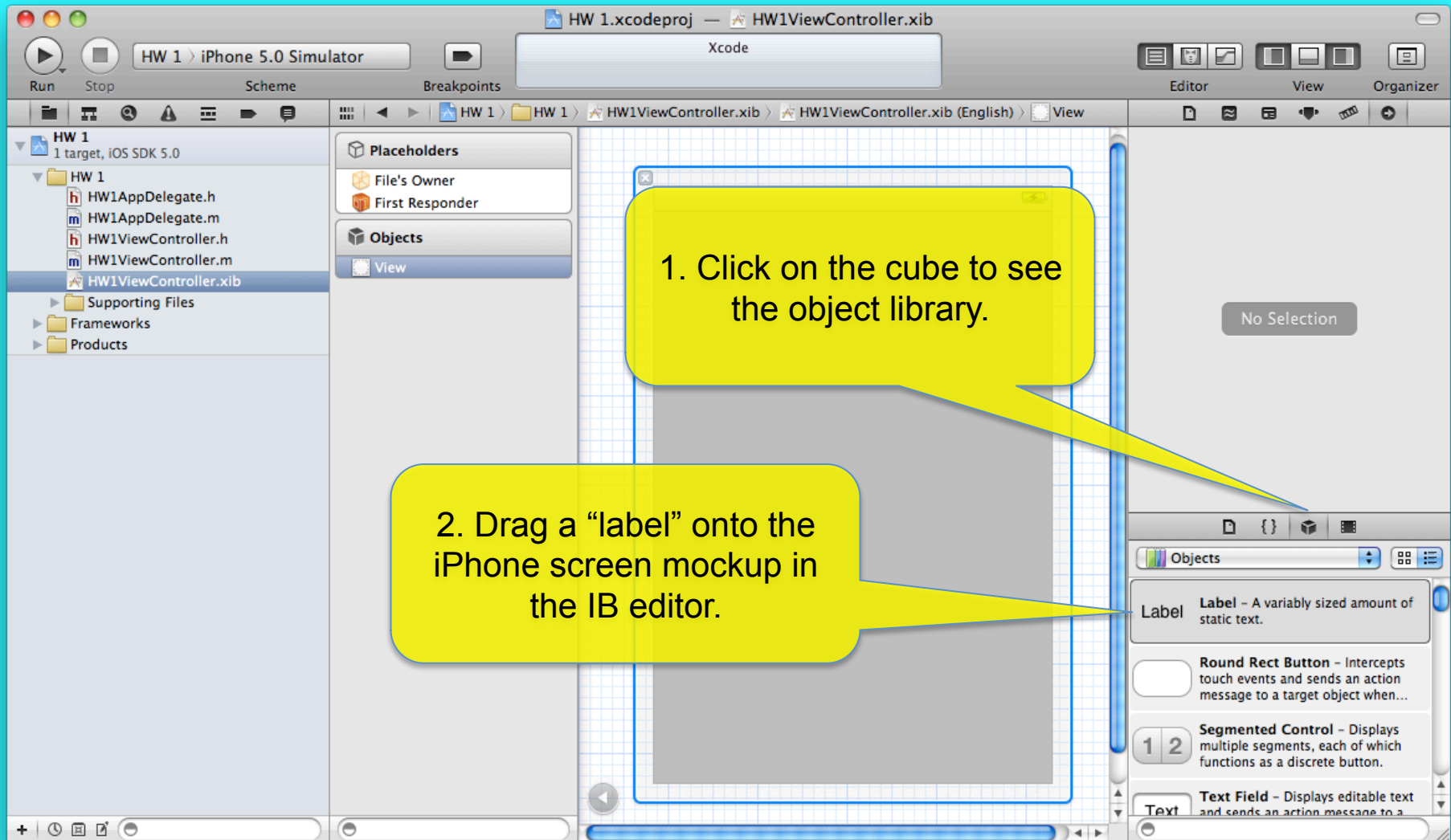
This icon selects the project folder for navigation.

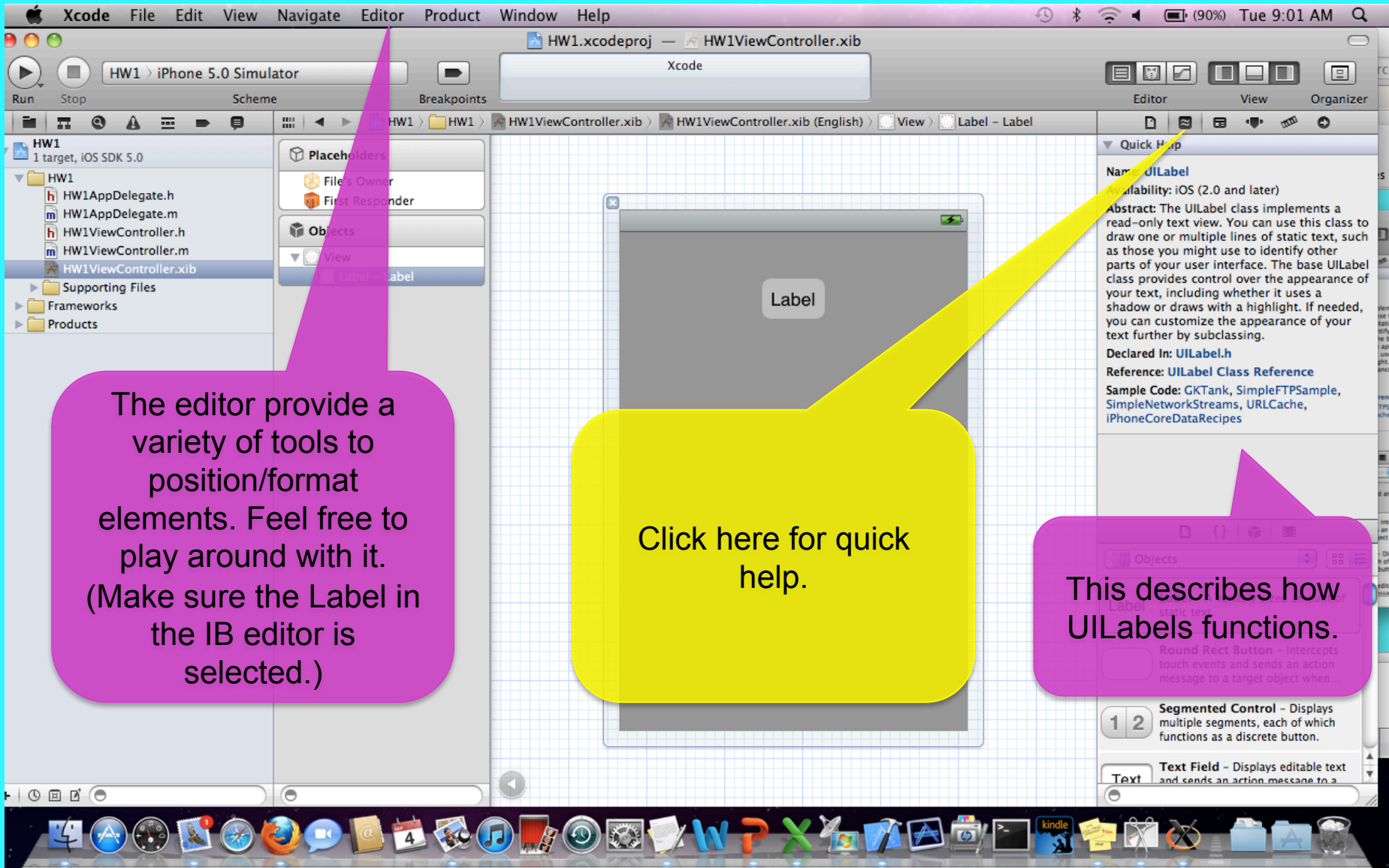




A xib file is an XML file that describes the user interface. Initially the file is empty but we can add to it through the IB editor.

Click on the HW1ViewController.xib file to display the interface builder (IB) editor.





The editor provide a variety of tools to position/format elements. Feel free to play around with it. (Make sure the Label in the IB editor is selected.)

Click here for quick help.

This describes how UILabels functions.

Quick Help

Name: UILabel

Availability: iOS (2.0 and later)

Abstract: The UILabel class implements a read-only text view. You can use this class to draw one or multiple lines of static text, such as those you might use to identify other parts of your user interface. The base UILabel class provides control over the appearance of your text, including whether it uses a shadow or draws with a highlight. If needed, you can customize the appearance of your text further by subclassing.

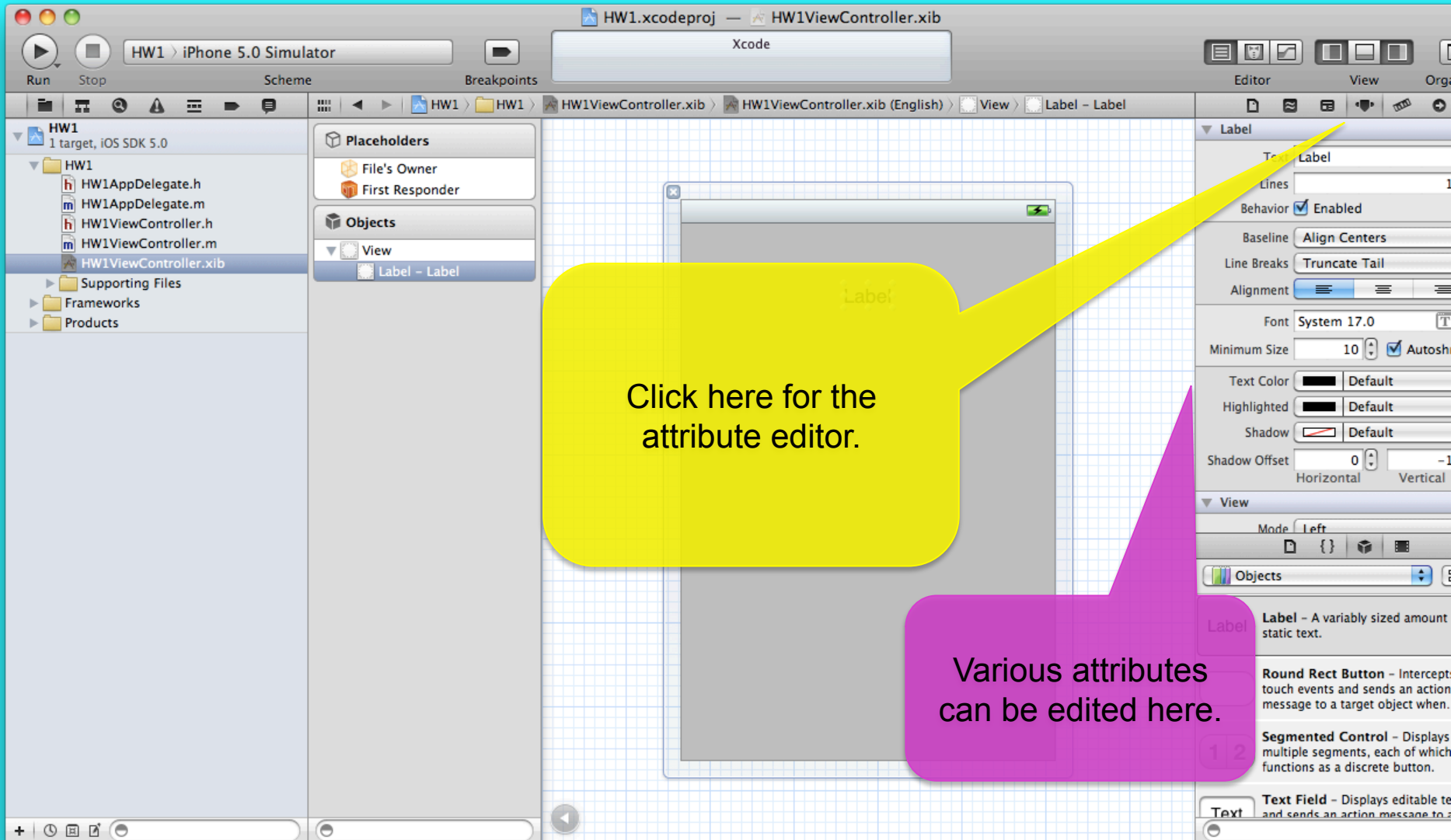
Declared In: UILabel.h

Reference: UILabel Class Reference

Sample Code: GKTank, SimpleFTPSample, SimpleNetworkStreams, URLCache, iPhoneCoreDataRecipes

Objects

- Label - static text
- Round Rect Button - Intercepts touch events and sends an action message to a target object when...
- 1 2 Segmented Control - Displays multiple segments, each of which functions as a discrete button.
- Text Text Field - Displays editable text and sends an action message to a...



Click here for the attribute editor.

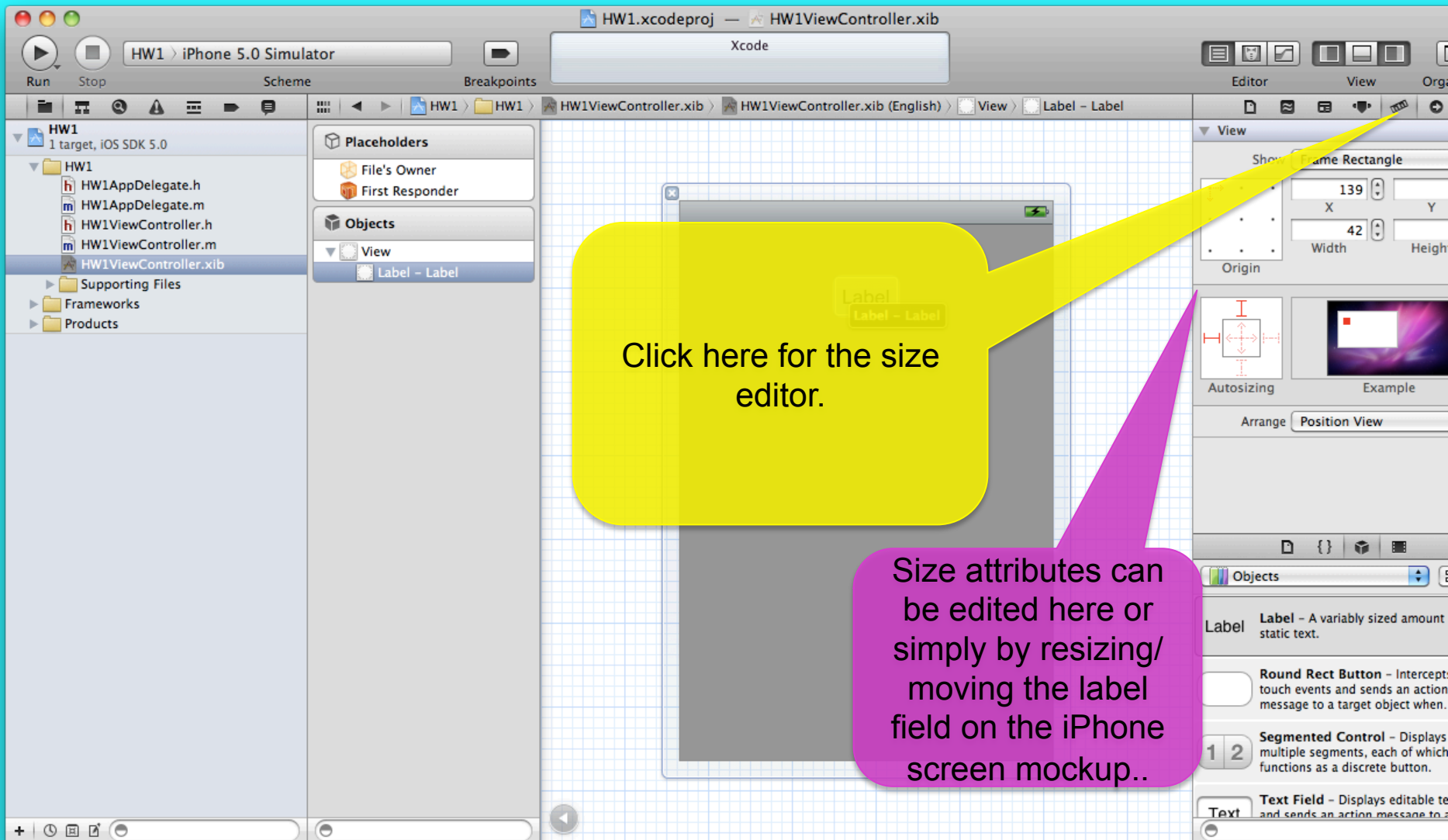
Various attributes can be edited here.

Label - A variably sized amount of static text.

Round Rect Button - Intercepts touch events and sends an action message to a target object when.

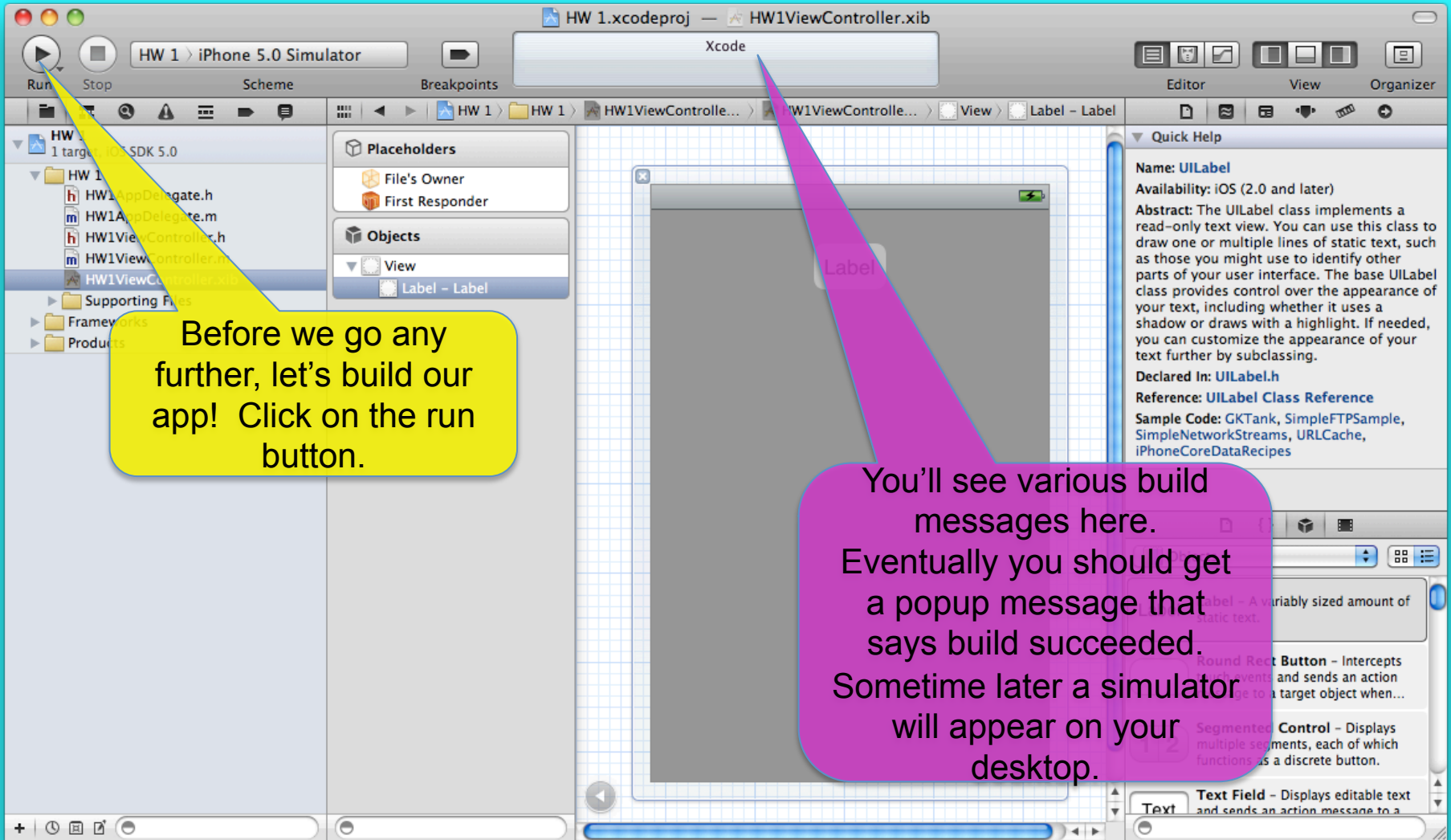
Segmented Control - Displays multiple segments, each of which functions as a discrete button.

Text Field - Displays editable text and sends an action message to a target object when.



Click here for the size editor.

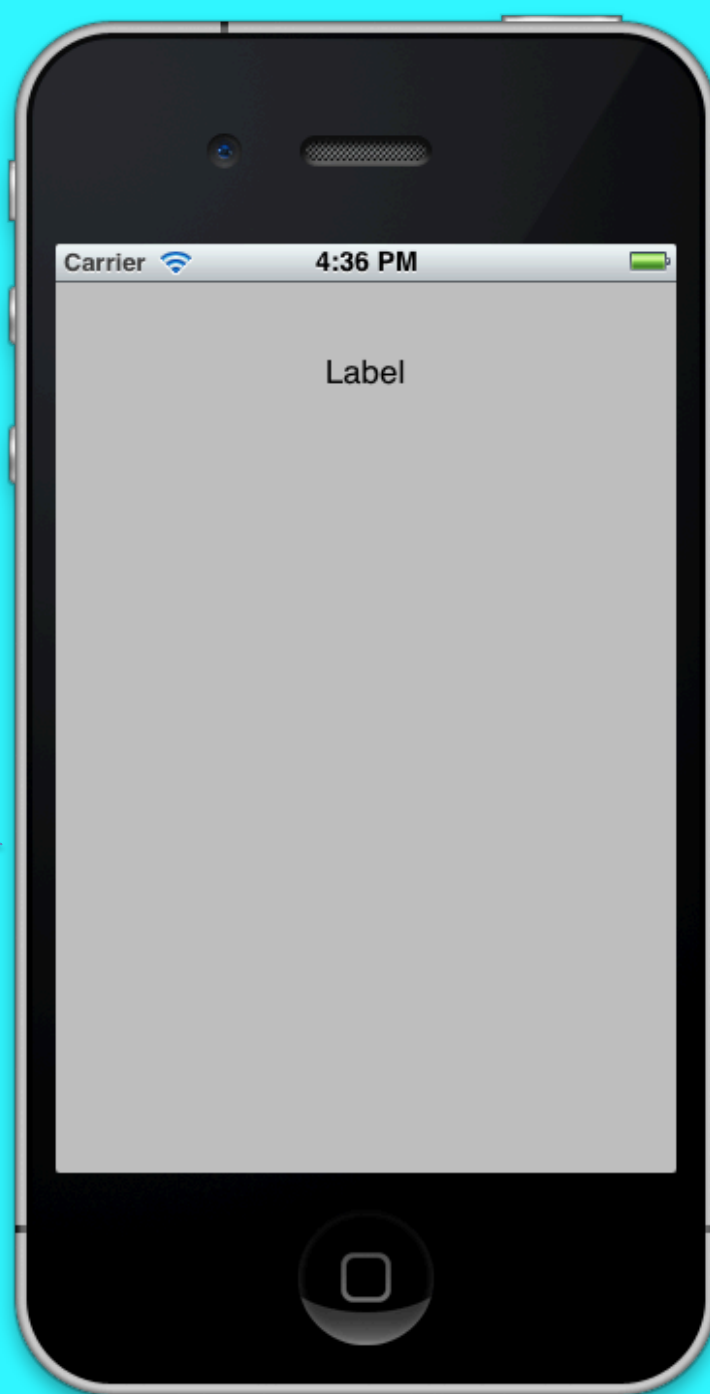
Size attributes can be edited here or simply by resizing/moving the label field on the iPhone screen mockup..

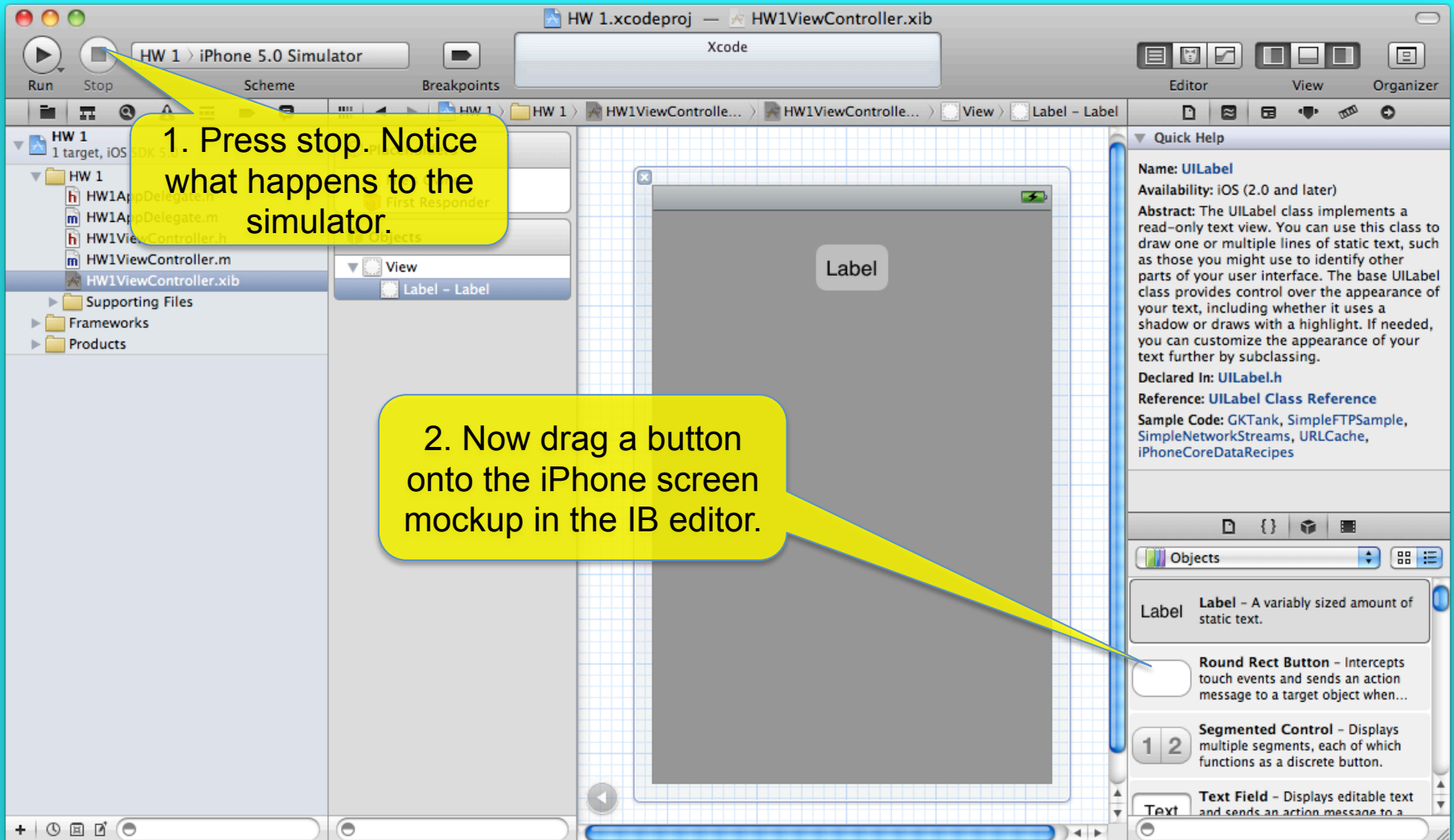


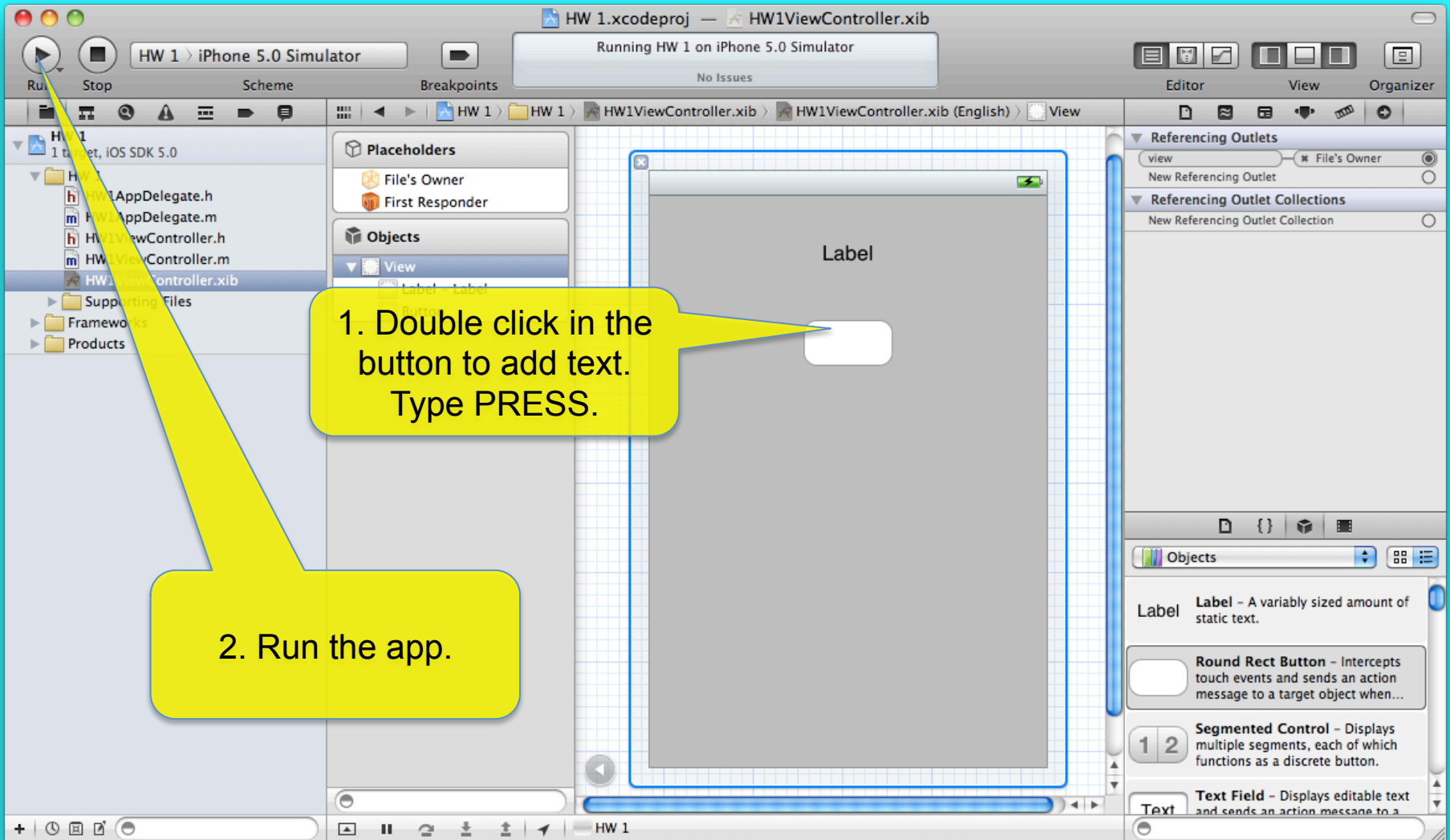
Before we go any further, let's build our app! Click on the run button.

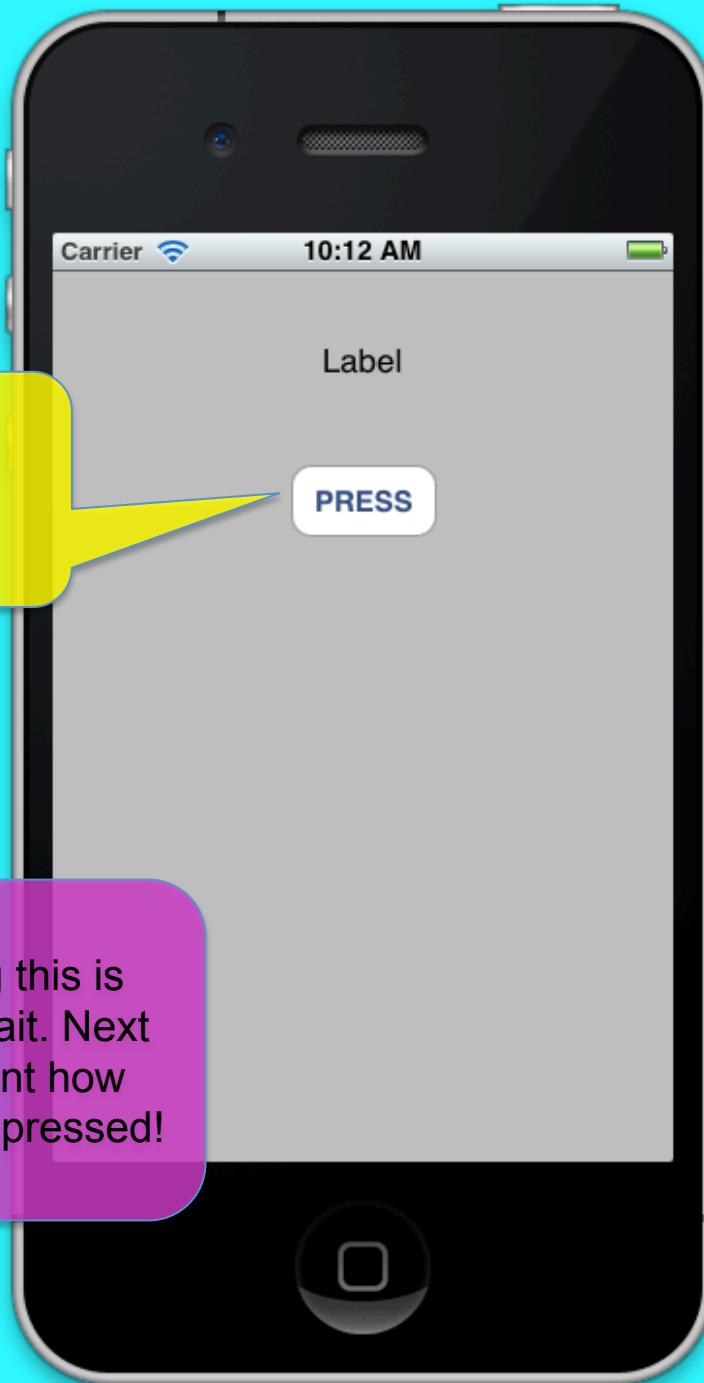
You'll see various build messages here. Eventually you should get a popup message that says build succeeded. Sometime later a simulator will appear on your desktop.

The app is now running in the iPhone simulator and appears on your desktop.









Press the button!

I know you are thinking this is awesome. 😊 But just wait. Next we'll make the app count how many times the button is pressed!

1. Stop the simulation.

2. Click on HW1ViewController.h to display the header file.

```
// HW1ViewController.h
// HW 1
// Created by Elizabeth Sweedyk on 8/27/12.
// Copyright (c) 2012 __MyCompanyName__. All rights reserved.

#import <UIKit/UIKit.h>

@interface HW1ViewController : UIViewController
@end
```

Class declarations have the following format:
@interface newClassName: parentClassName

```
{
    // ClassMembers
}
// ClassMethods
@end
```

The parentheses are optional when there are no class members.

iOS uses a Model-View-Controller (MVC) architectural pattern. We'll talk more about it in the next lecture. For now, all you need to know is that the HW1ViewController class is going to do the work of counting the button presses.

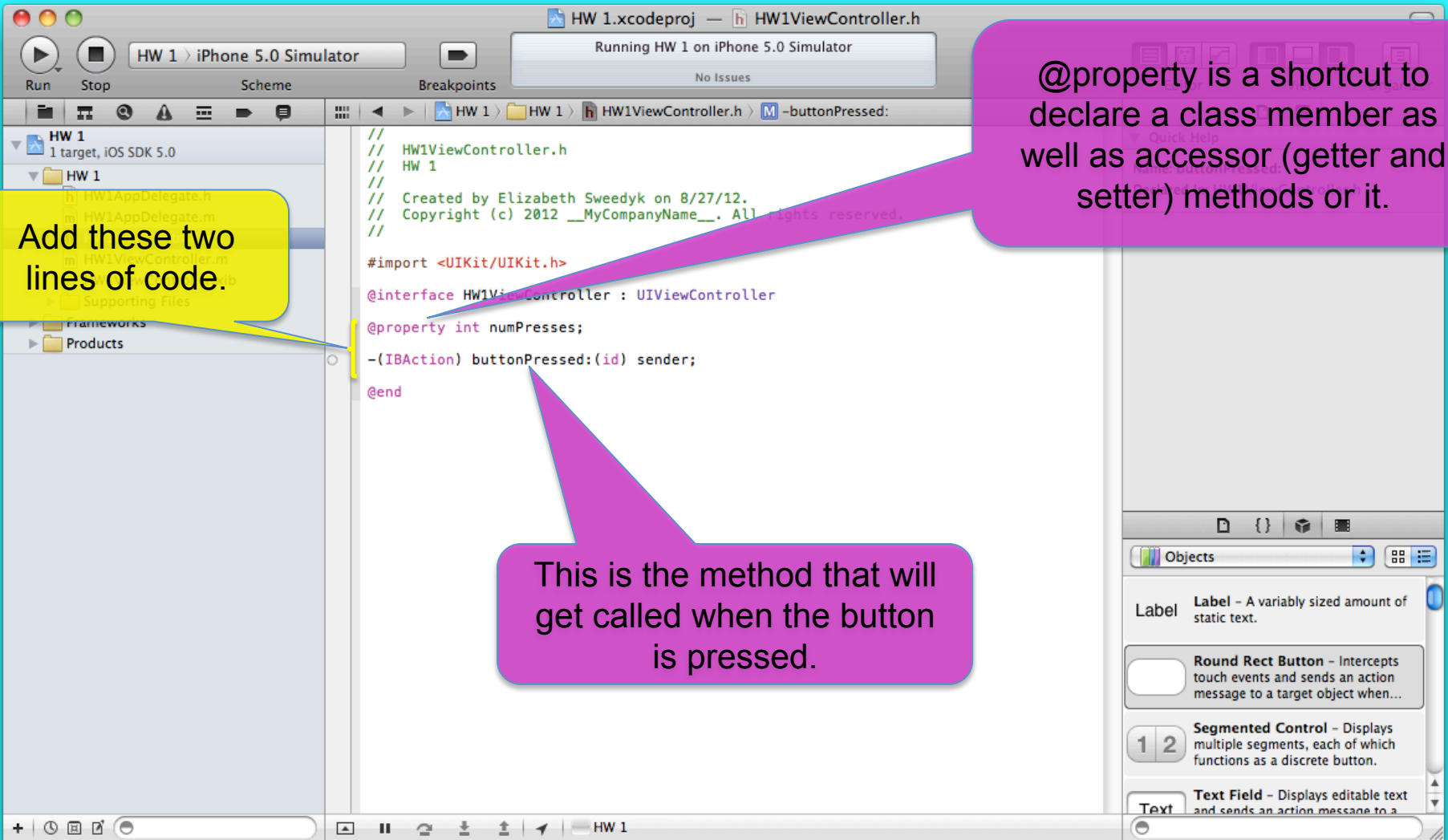
Abstract: The UIView class defines a rectangular area on the screen and the interfaces for managing the content in that area. At runtime, a view object handles the rendering of any content in its area and also handles any interactions with that content. The UIView class itself provides basic behavior for filling its rectangular area with a background color.

Label - A variably sized amount of static text.

Round Rect Button - Intercepts touch events and sends an action message to a target.

Segmented Control - Displays multiple segments, each of which functions as a discrete button.

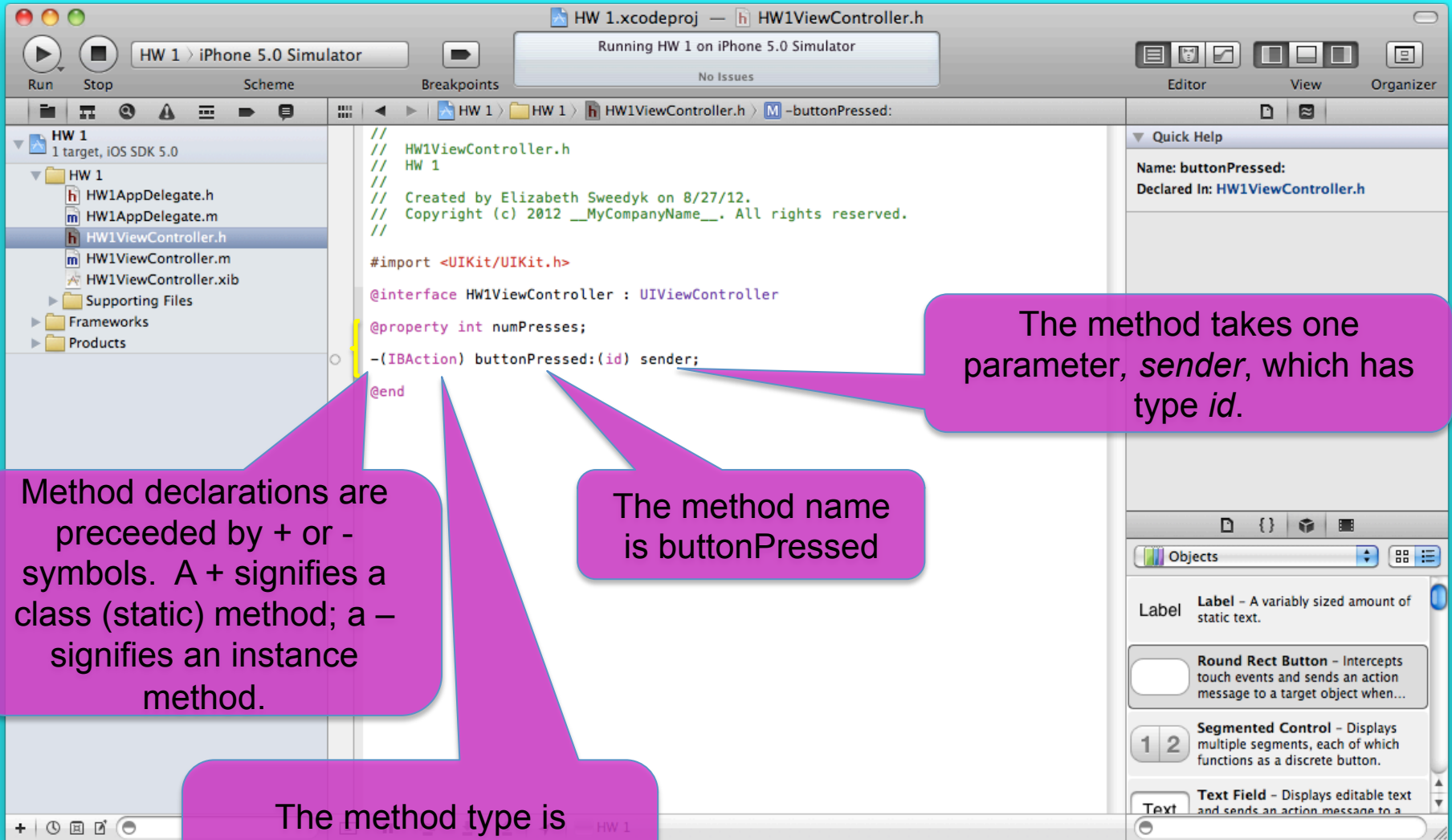
Text Field - Displays editable text and sends an action message to a target.



Add these two lines of code.

@property is a shortcut to declare a class member as well as accessor (getter and setter) methods or it.

This is the method that will get called when the button is pressed.



Method declarations are preceded by + or - symbols. A + signifies a class (static) method; a - signifies an instance method.

The method name is buttonPressed

The method type is IBAction. The IB prefix stands for Interface Builder.

The method takes one parameter, *sender*, which has type *id*.

Quick Help
Name: buttonPressed:
Declared In: HW1ViewController.h

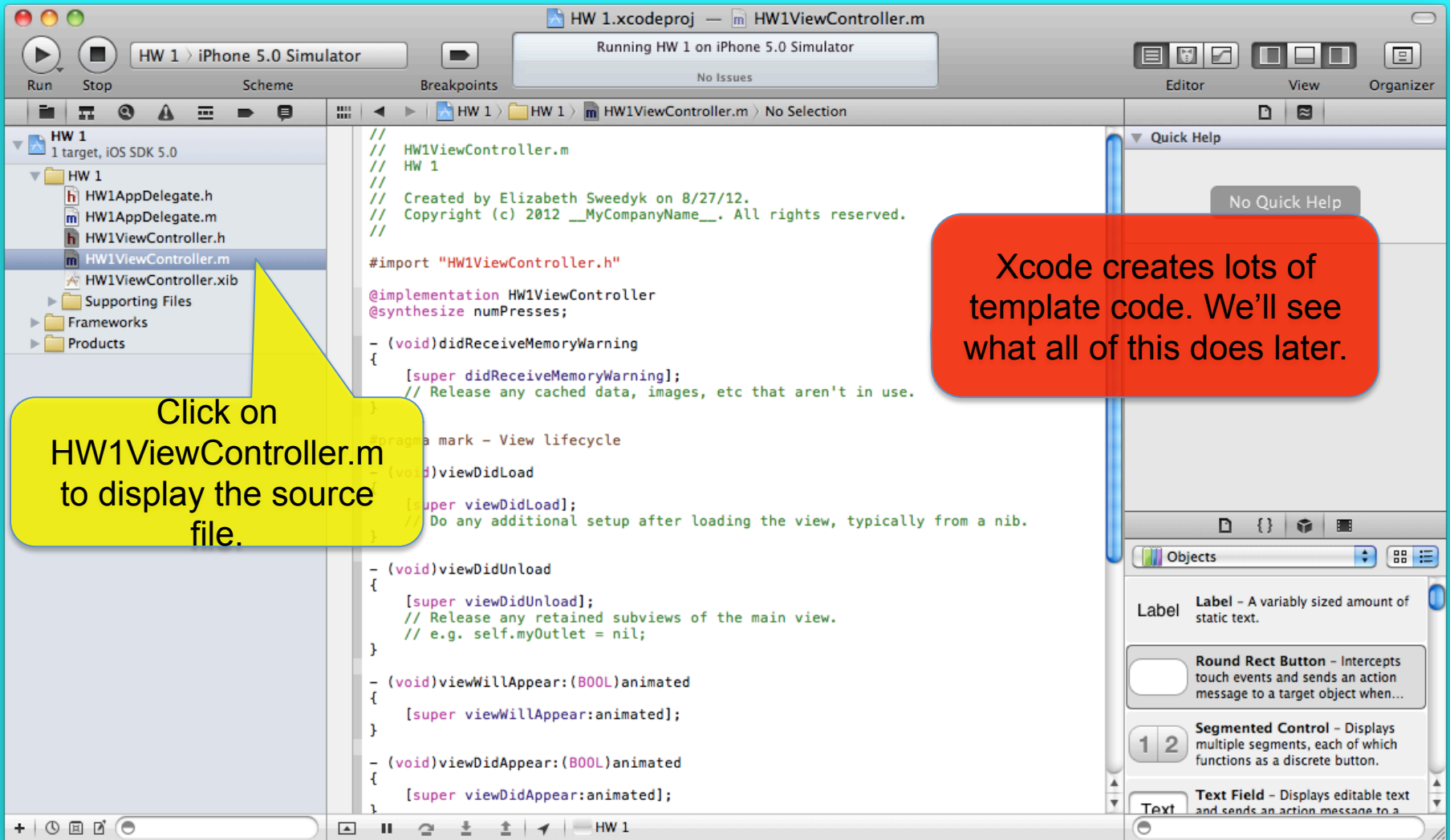
Objects

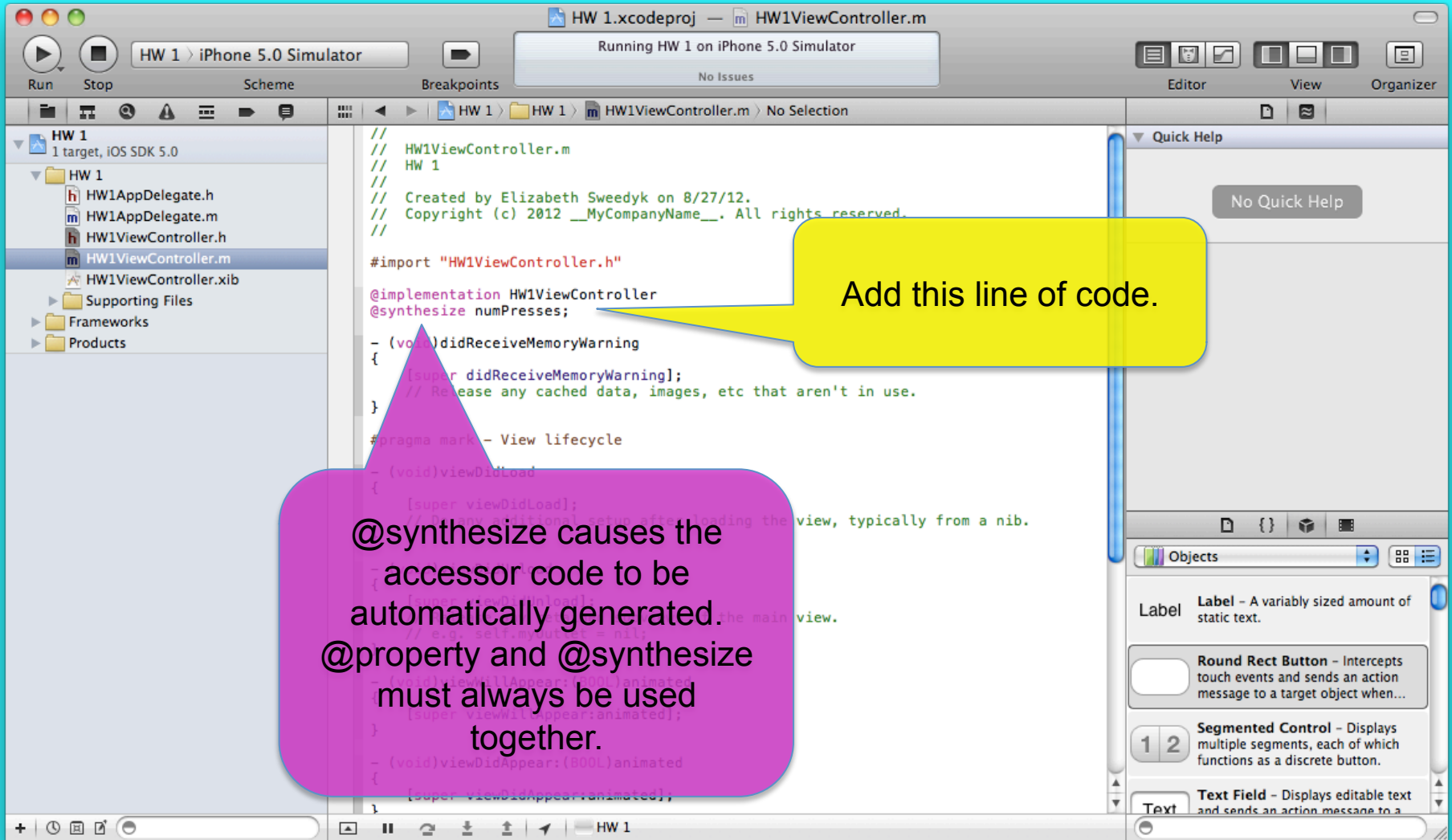
Label Label - A variably sized amount of static text.

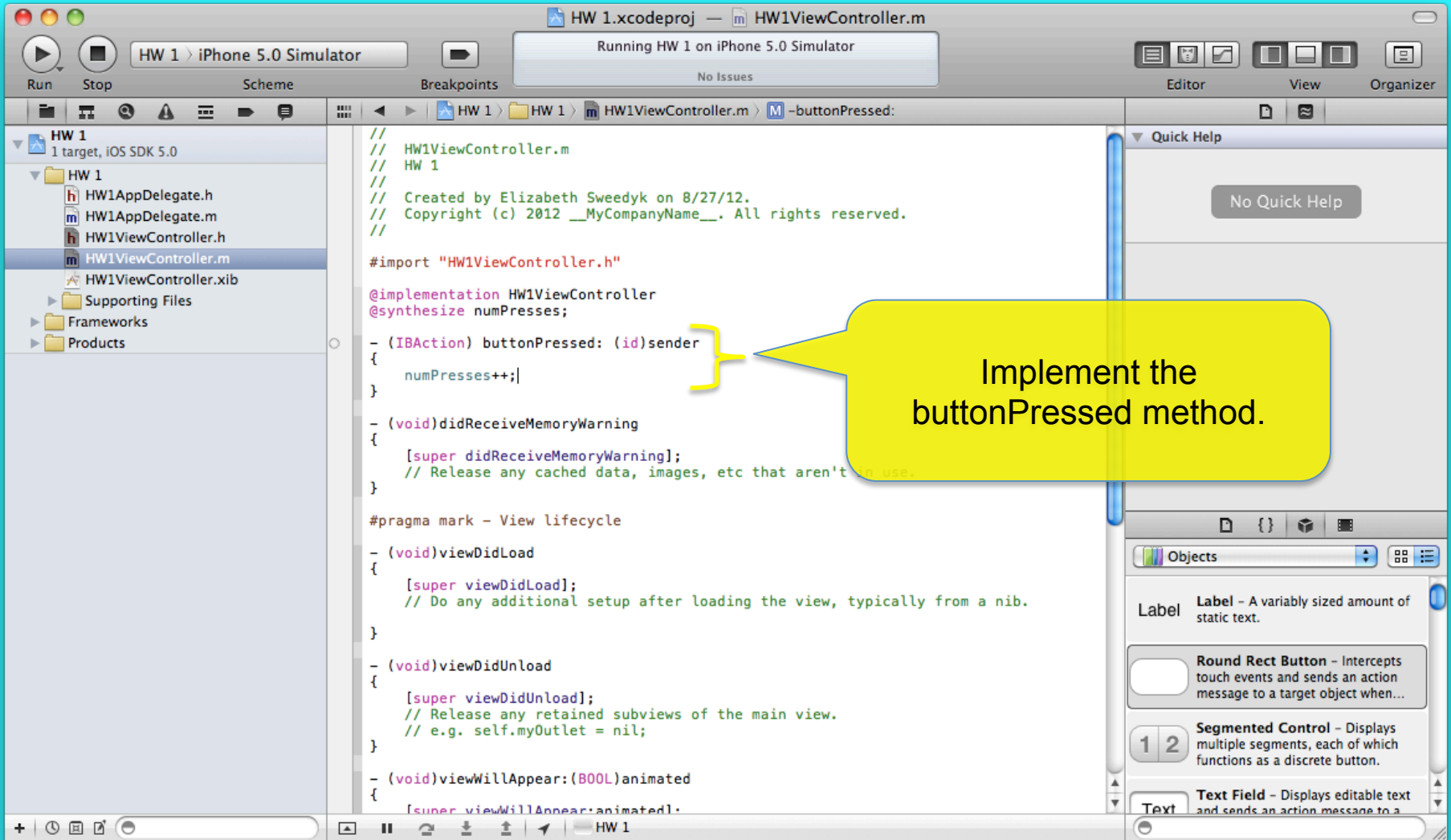
Round Rect Button - Intercepts touch events and sends an action message to a target object when...

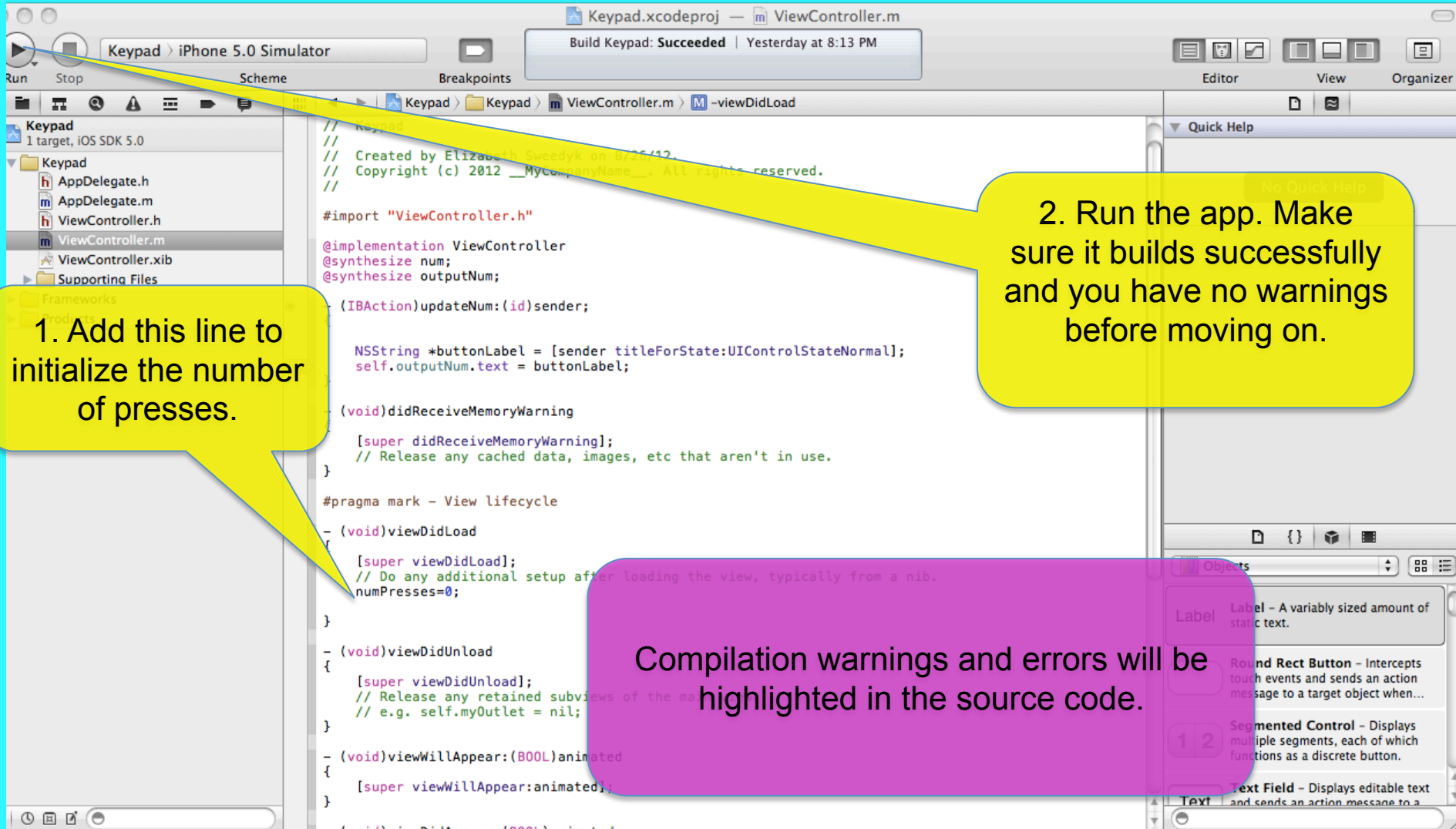
1 2 Segmented Control - Displays multiple segments, each of which functions as a discrete button.

Text Text Field - Displays editable text and sends an action message to a...







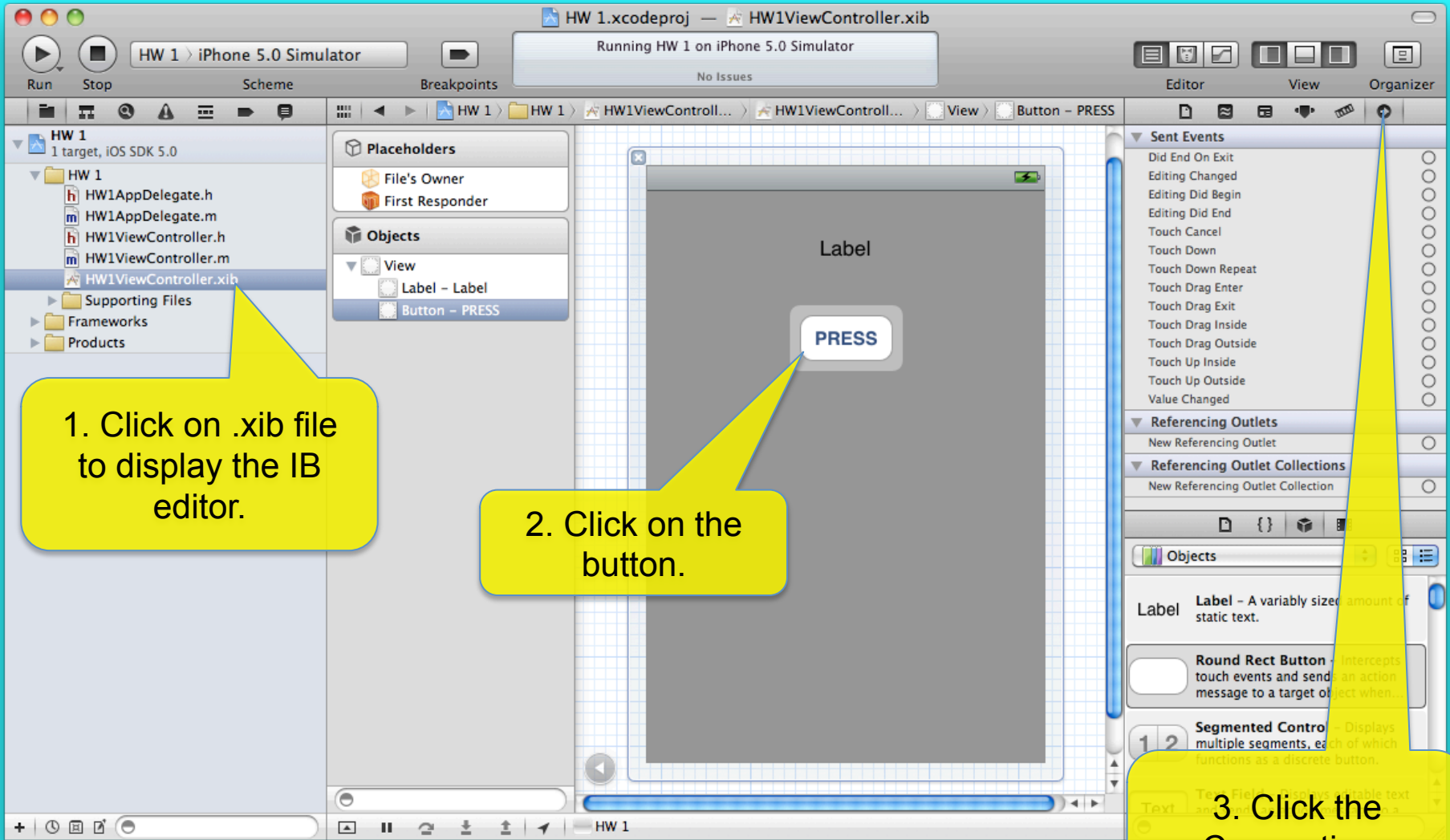


1. Add this line to initialize the number of presses.

2. Run the app. Make sure it builds successfully and you have no warnings before moving on.

Compilation warnings and errors will be highlighted in the source code.

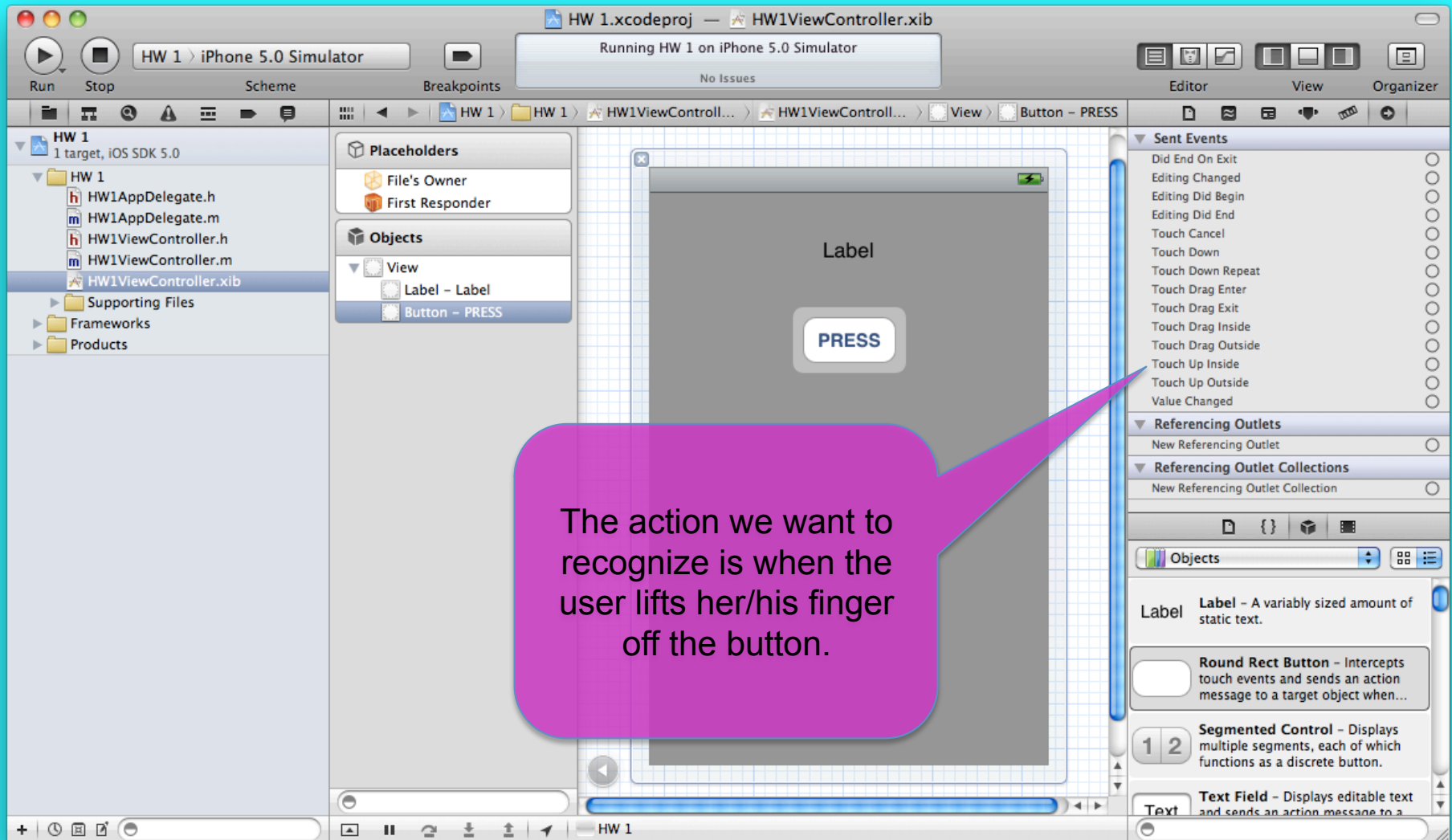
Next we need to “connect” our PRESS button in the UI to our buttonPressed method.

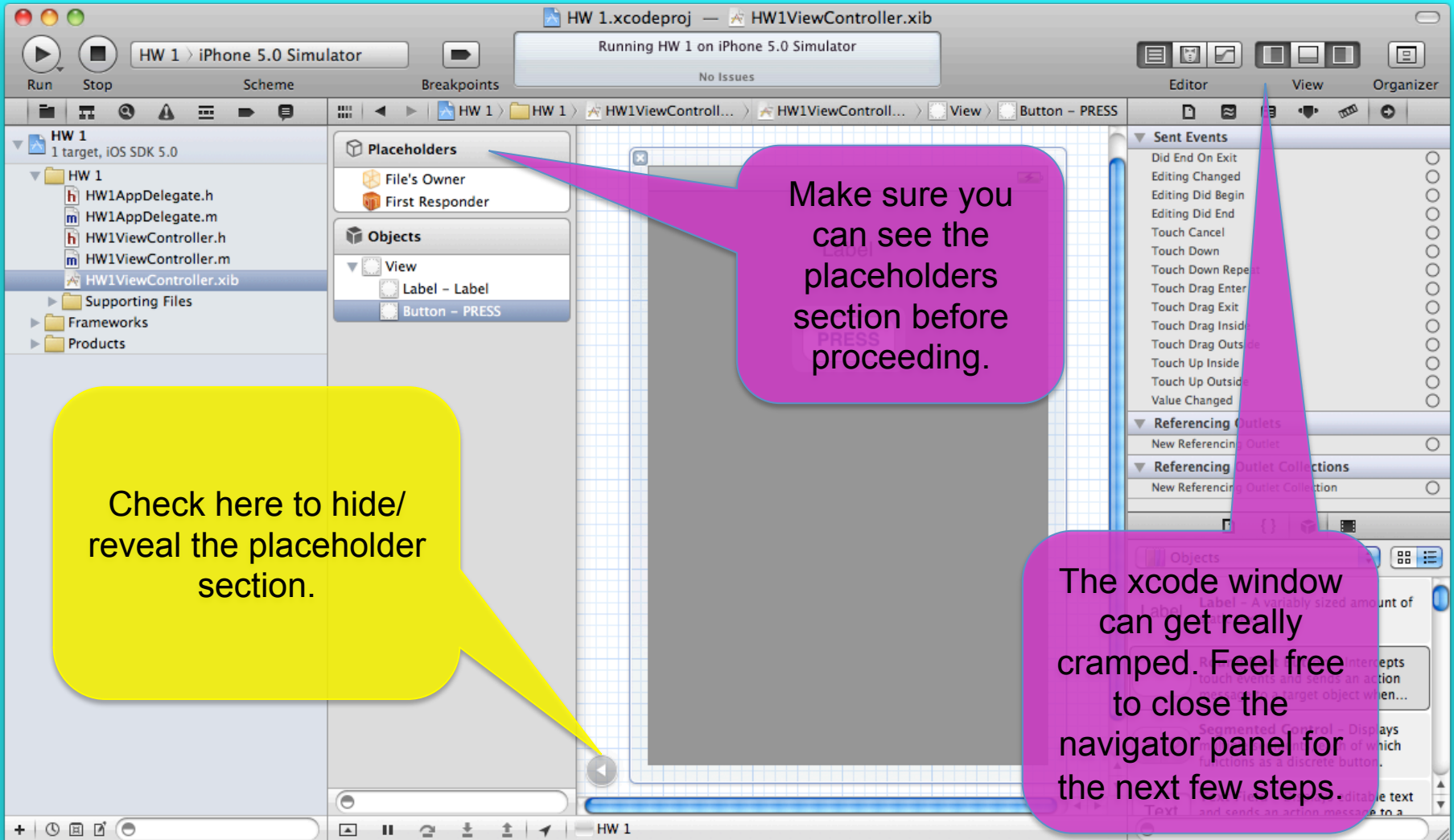


1. Click on .xib file to display the IB editor.

2. Click on the button.

3. Click the Connections Inspector icon.

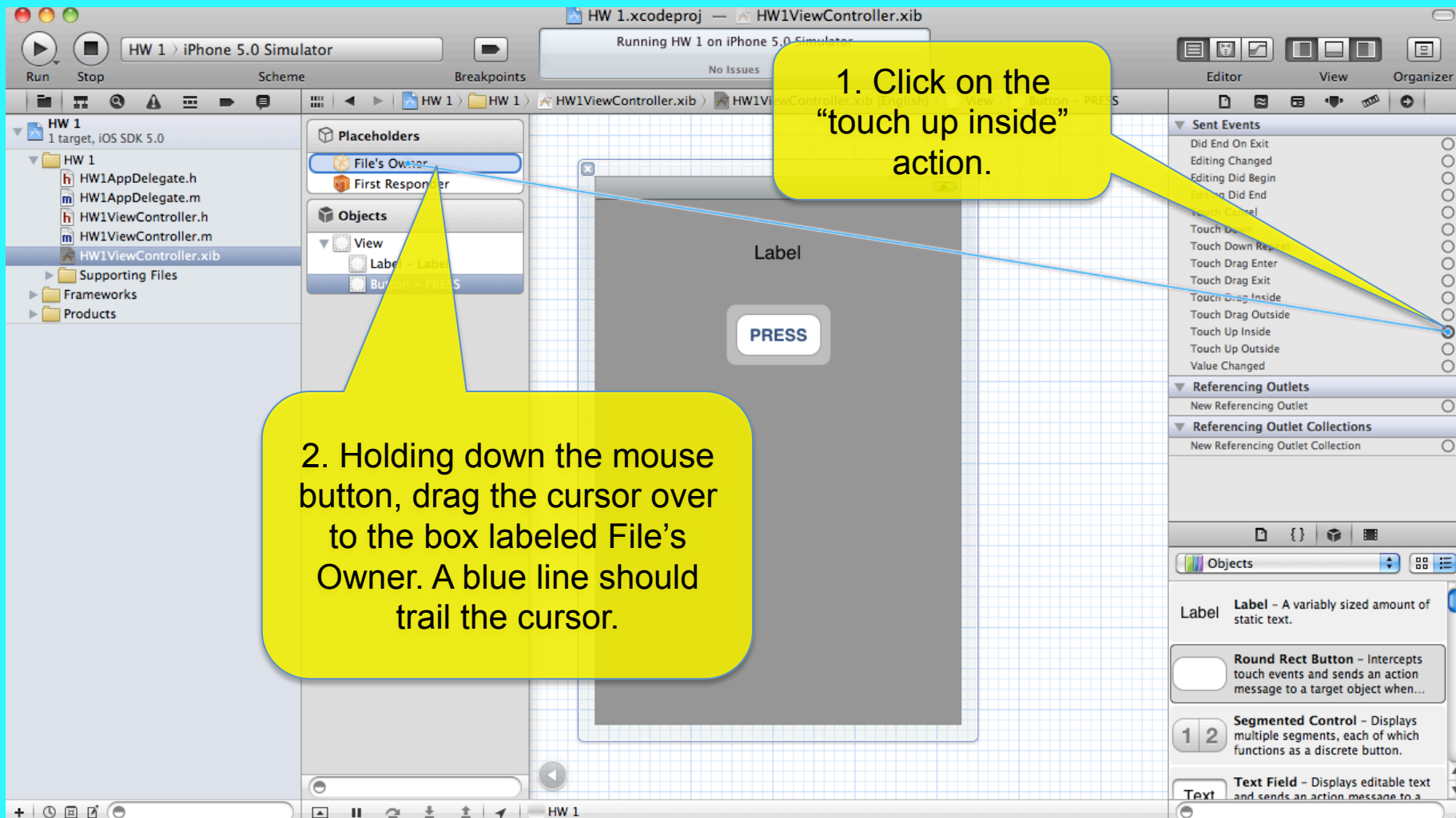


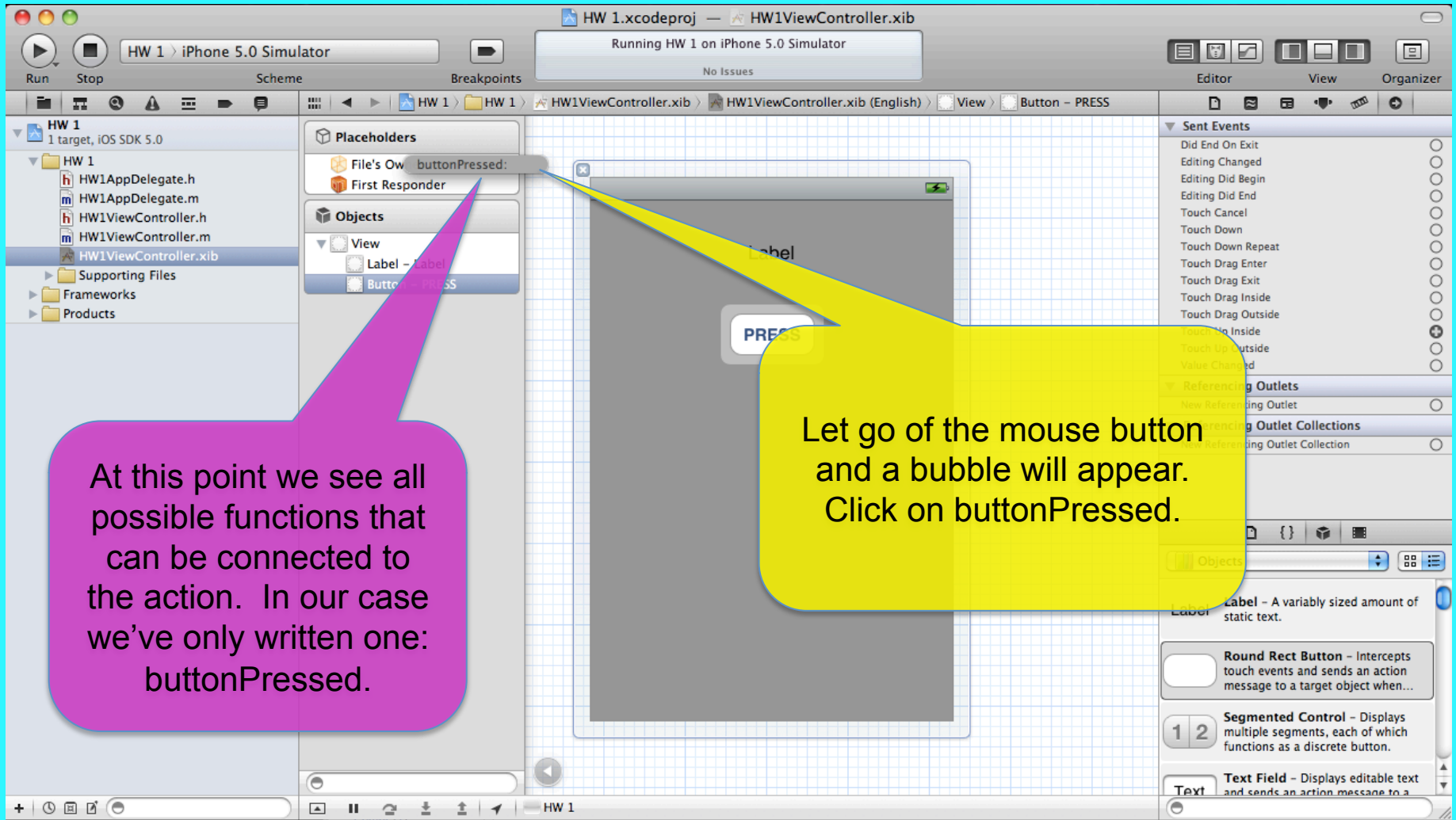


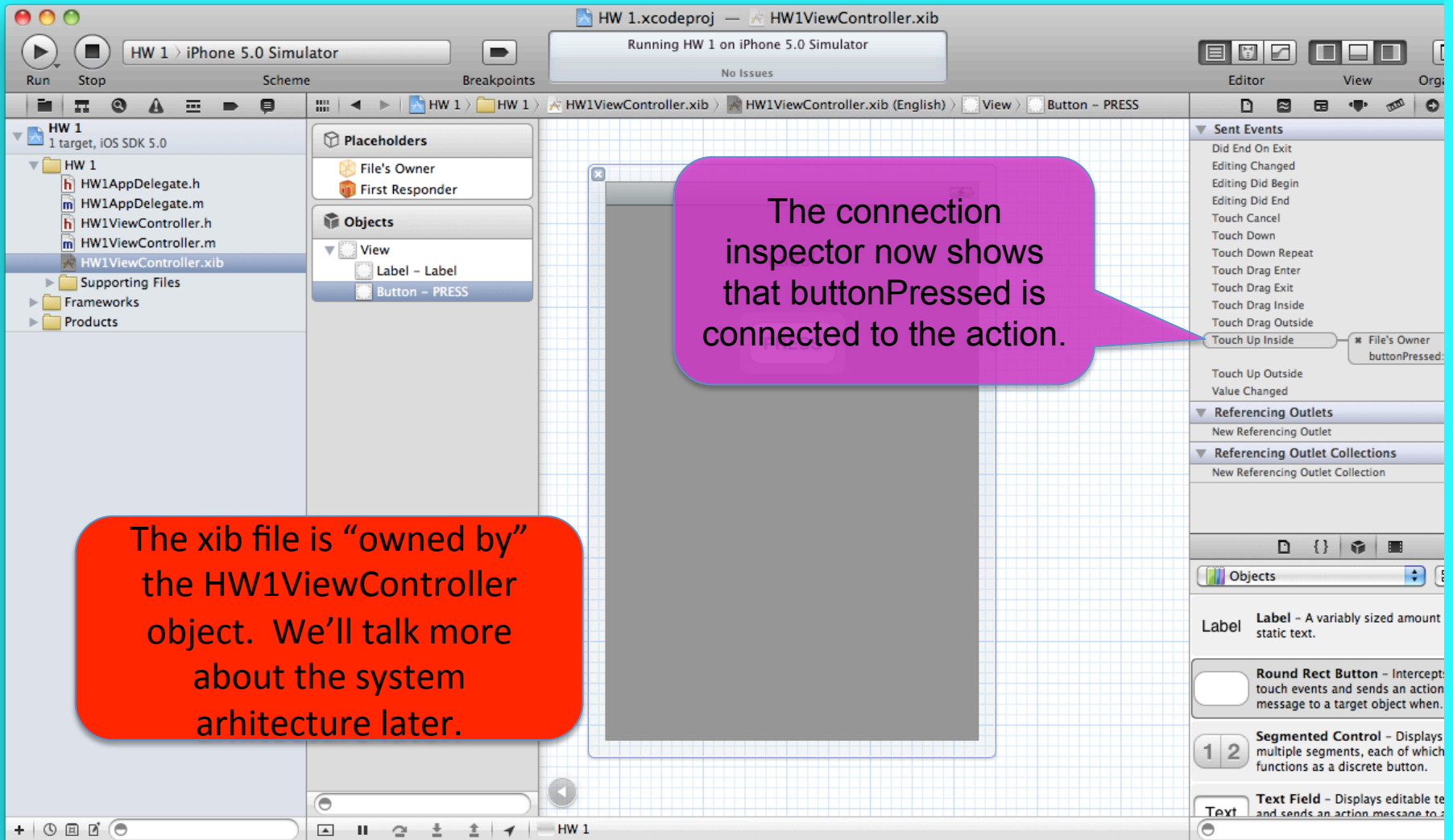
Check here to hide/
reveal the placeholder
section.

Make sure you
can see the
placeholders
section before
proceeding.

The xcode window
can get really
cramped. Feel free
to close the
navigator panel
for the next few steps.







The connection inspector now shows that buttonPressed is connected to the action.

The xib file is "owned by" the HW1ViewController object. We'll talk more about the system architecture later.

At this point the easiest way to check that we are counting button presses is to add a “print statement.”

1. Open the HW1ViewController.m file.

2. Add this line of code.

```
// HW1ViewController.m
// HW1
// Created by Elizabeth Sweedyk on 9/4/12.
// Copyright (c) 2012 __MyCompanyName__. All rights reserved.

#import "HW1ViewController.h"

@implementation HW1ViewController

@synthesize numPresses;

-(IBAction) buttonPressed:(id) sender
{
    numPresses++;
    NSLog(@"%d", numPresses);
}

-(void) didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Release any cached data, images, etc that aren't in use.
}

#pragma mark - View lifecycle

-(void) viewDidLoad
{
    [super viewDidLoad];
    // Do any additional setup after loading the view, typically from a nib.
    numPresses=0;
}

-(void) viewWillAppear:(BOOL)animated
{
    [super viewWillAppear:animated];
}
```

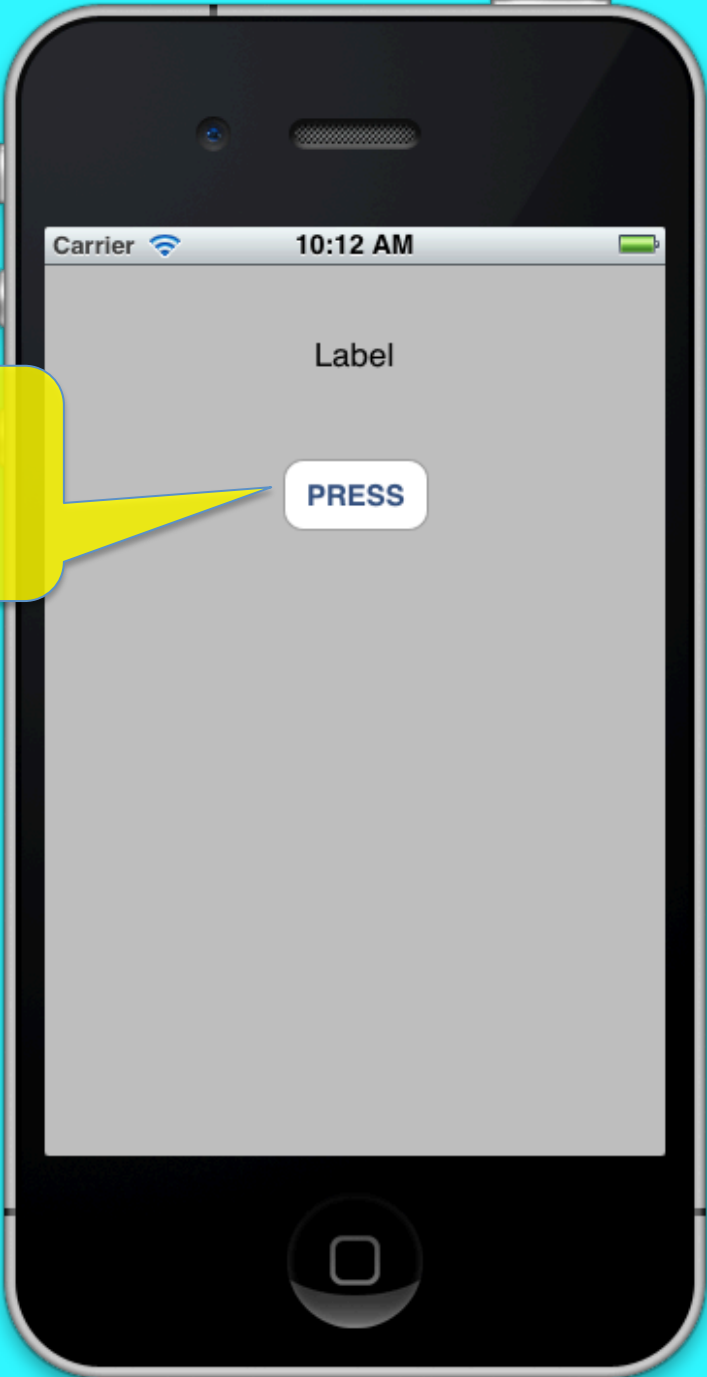
We will talk more about NSStrings next time.

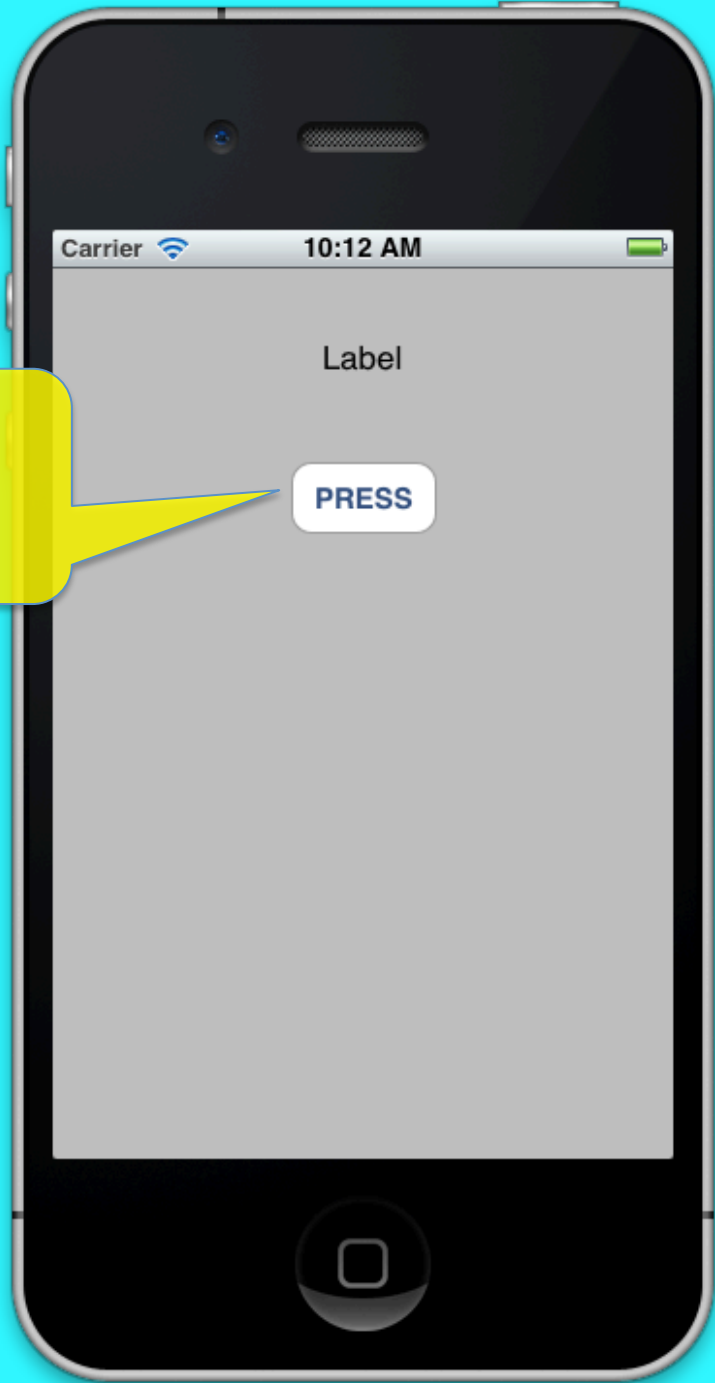
In C we'd use printf("%d", numPresses);

Objective-C strings (NSStrings to be precise) are preceded by an @ symbol.



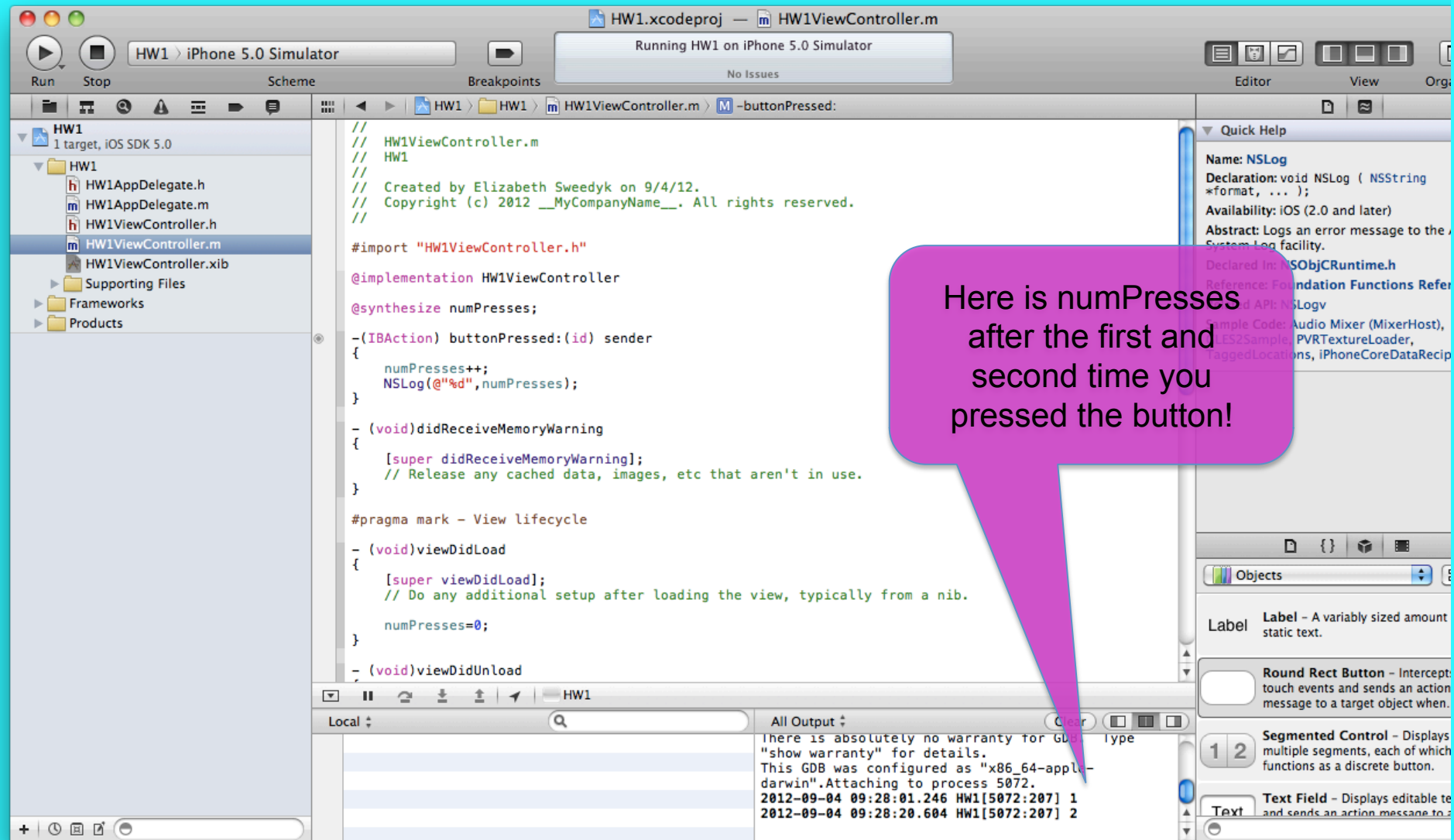
Press the button!



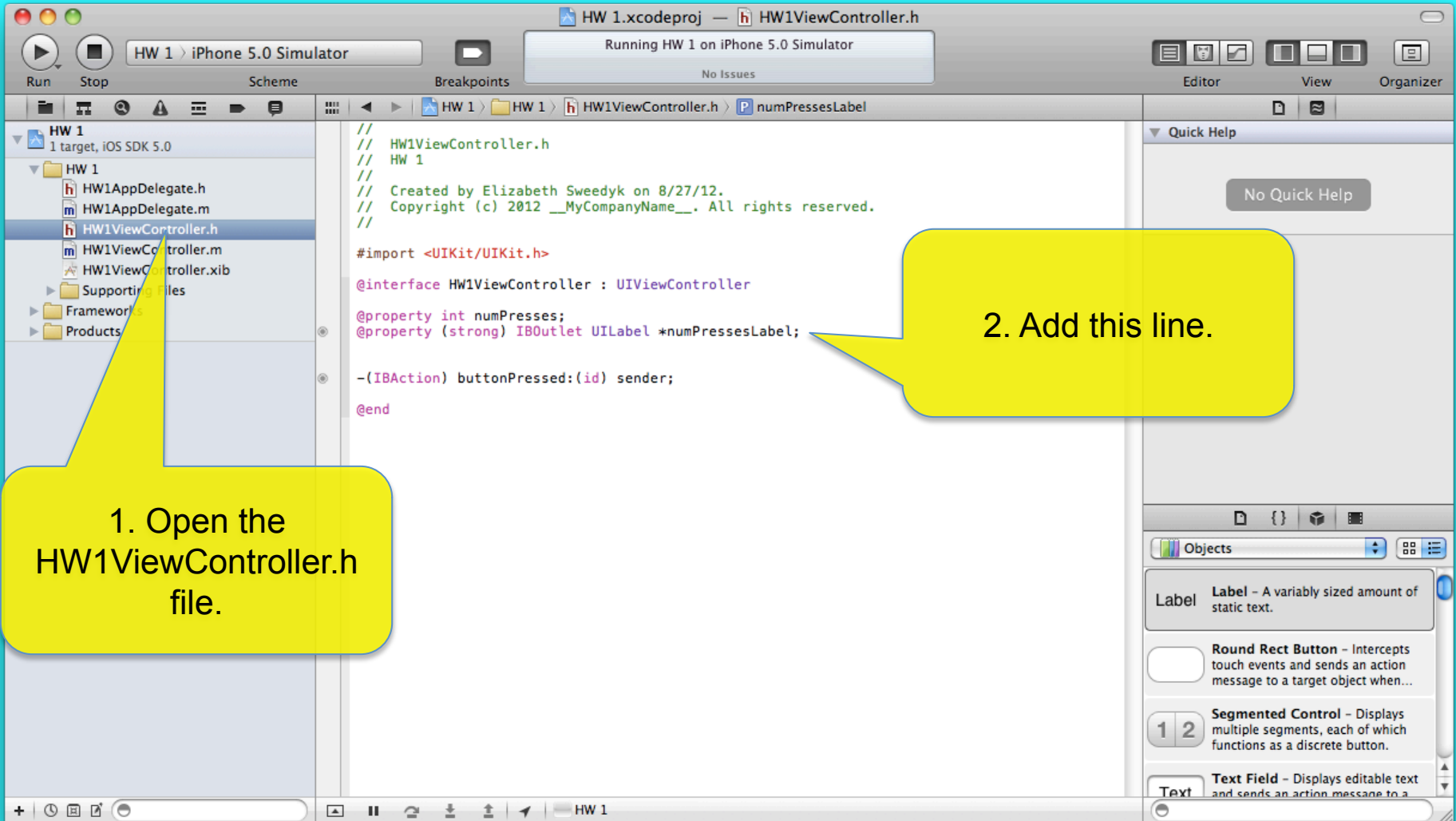


Press the button
again.

PRESS



Now we are going to display the number of button presses in the label field.



1. Open the HW1ViewController.h file.

2. Add this line.

We can connect interface elements to our code through *actions* and *outlets*.

This is related to ARC. We'll talk more about it later.

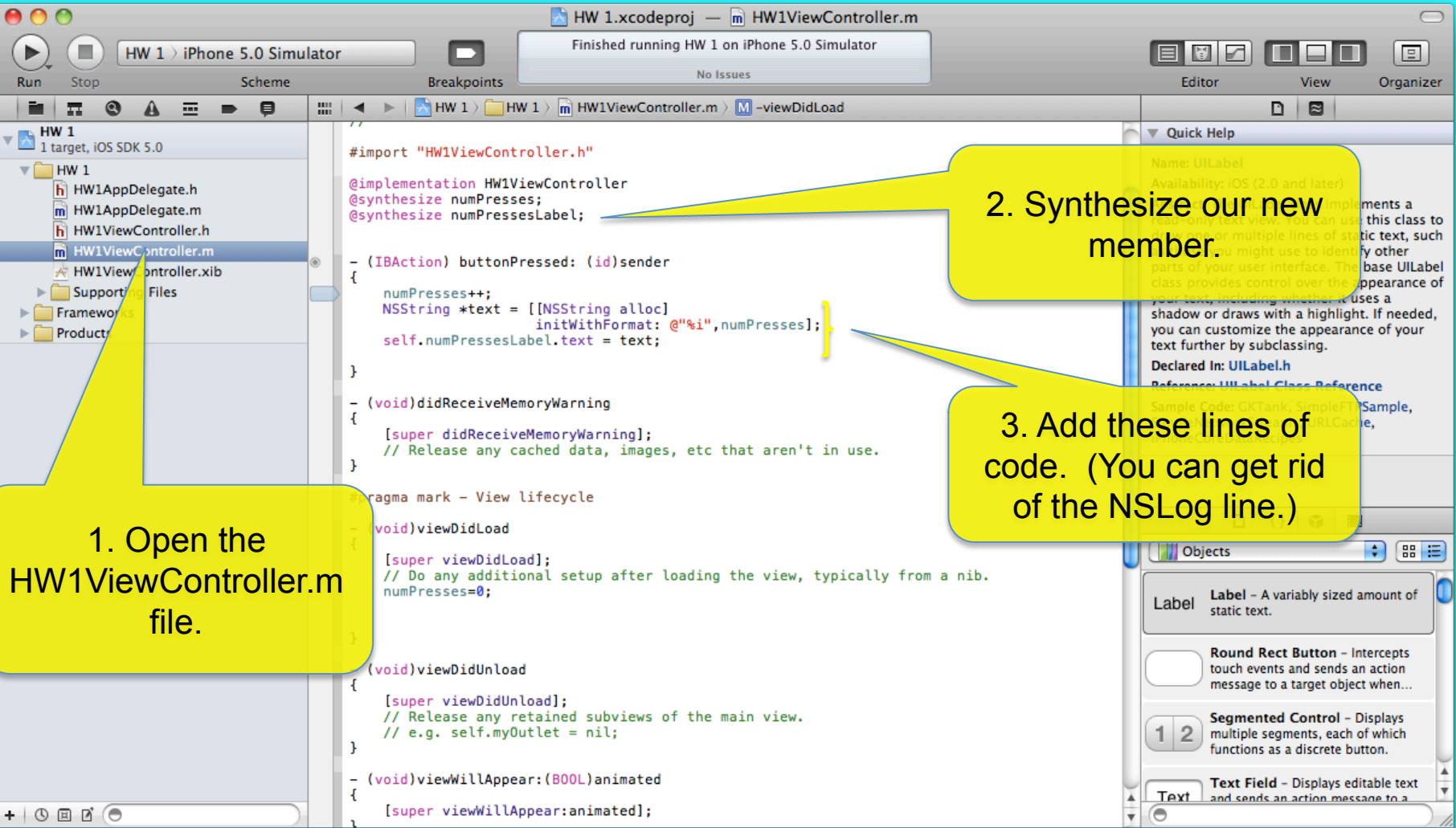
The IBOutlet keyword is ignored by the compiler. It is used to make the code more readable by indicating the class member numPressesLabel is connected to a UI element.

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

@interface HW1ViewController : UIViewController
@property int numPresses;
@property (strong) IBOutlet UILabel *numPressesLabel;

-(IBAction) buttonPressed:(id) sender;

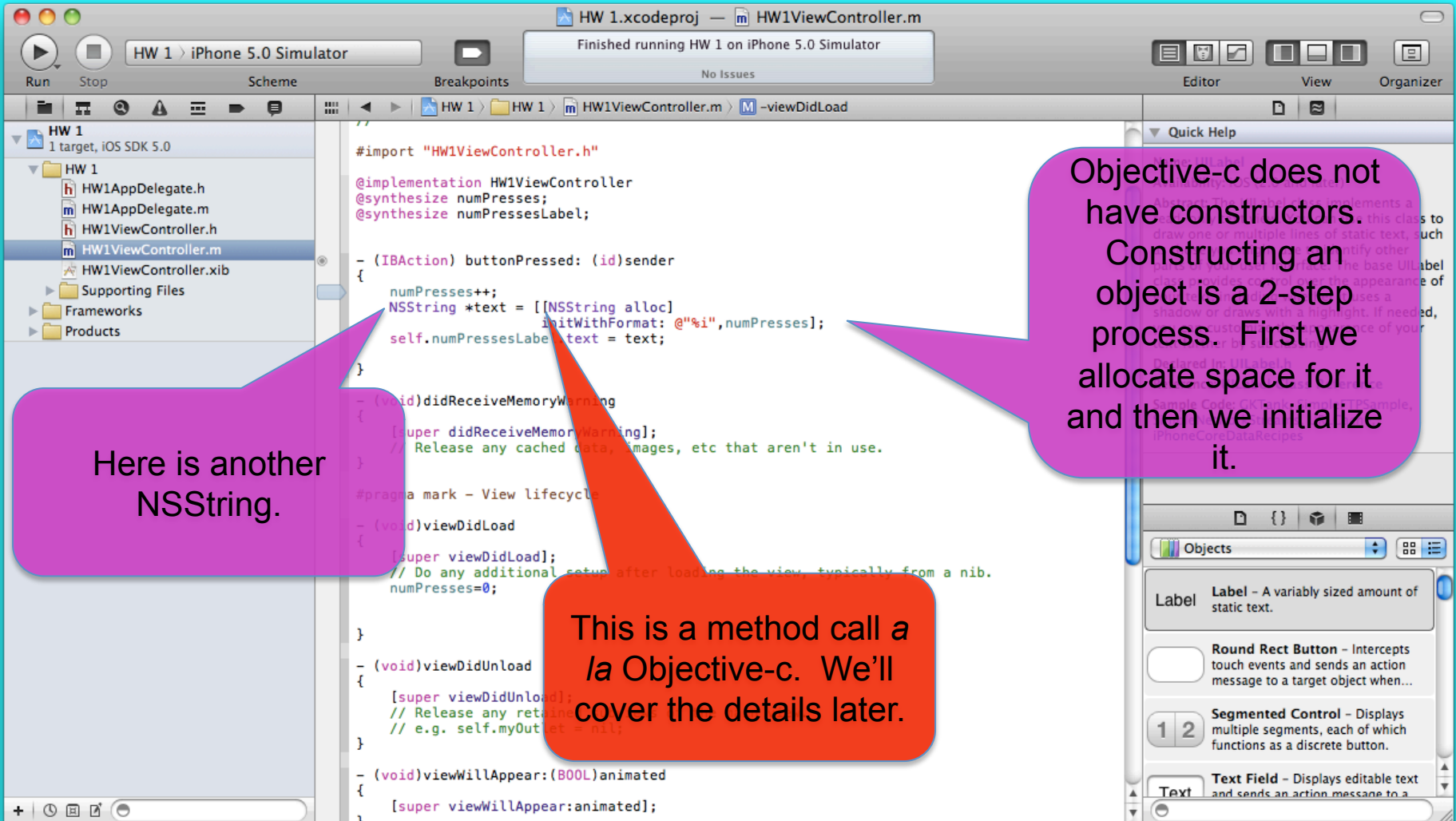
@end
```



1. Open the HW1ViewController.m file.

2. Synthesize our new member.

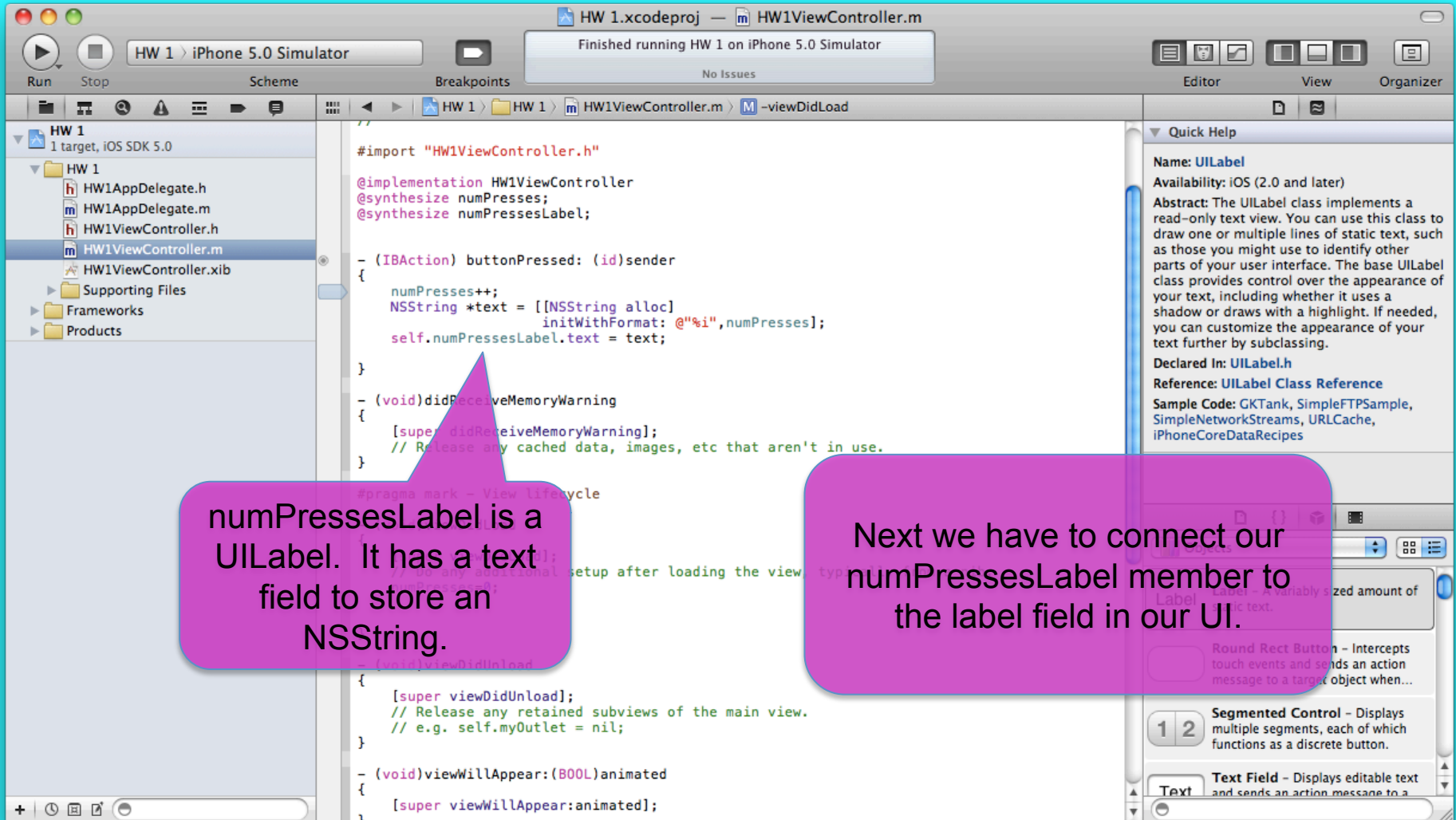
3. Add these lines of code. (You can get rid of the NSLog line.)

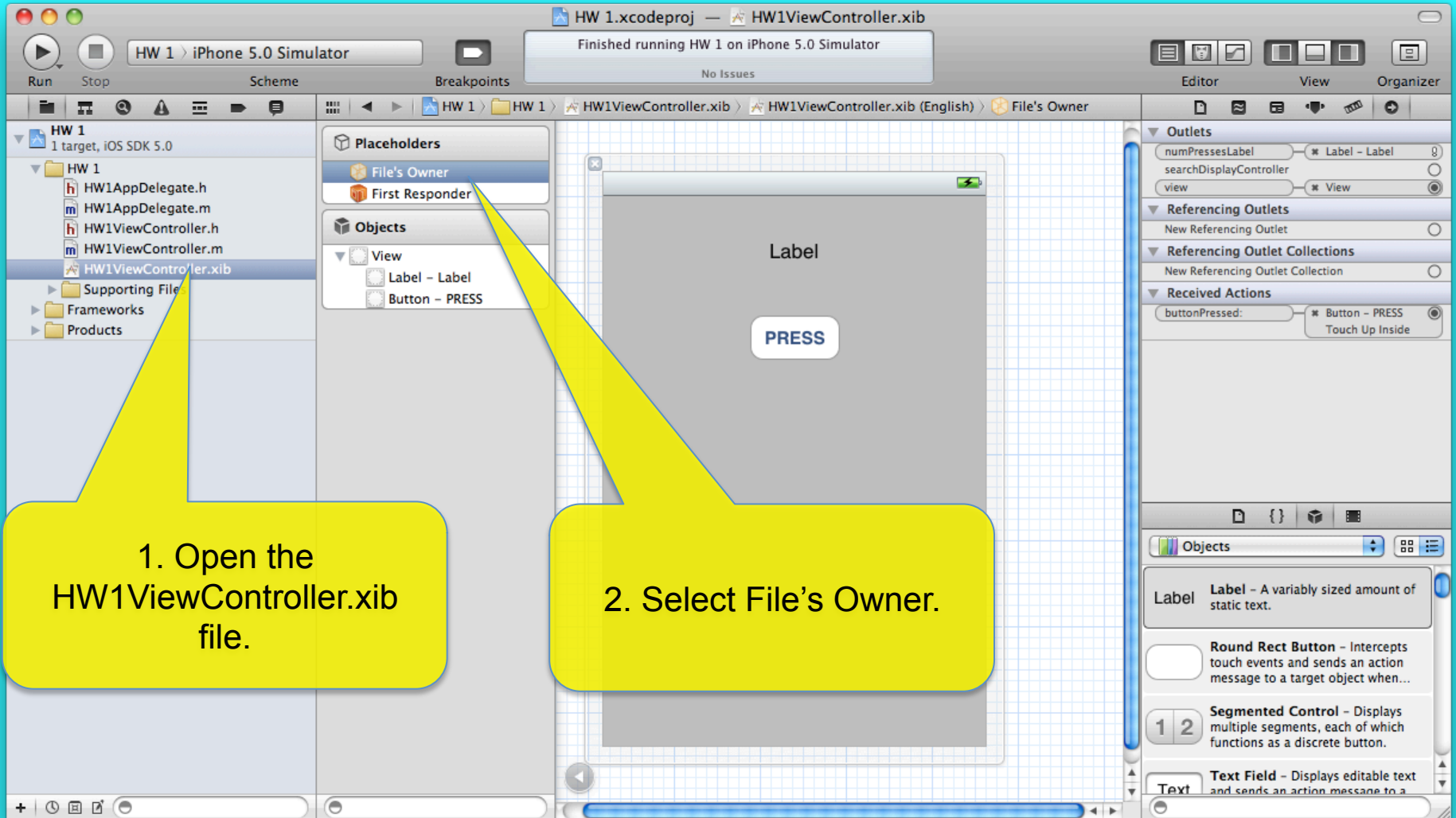


Here is another NSString.

This is a method call a *la* Objective-c. We'll cover the details later.

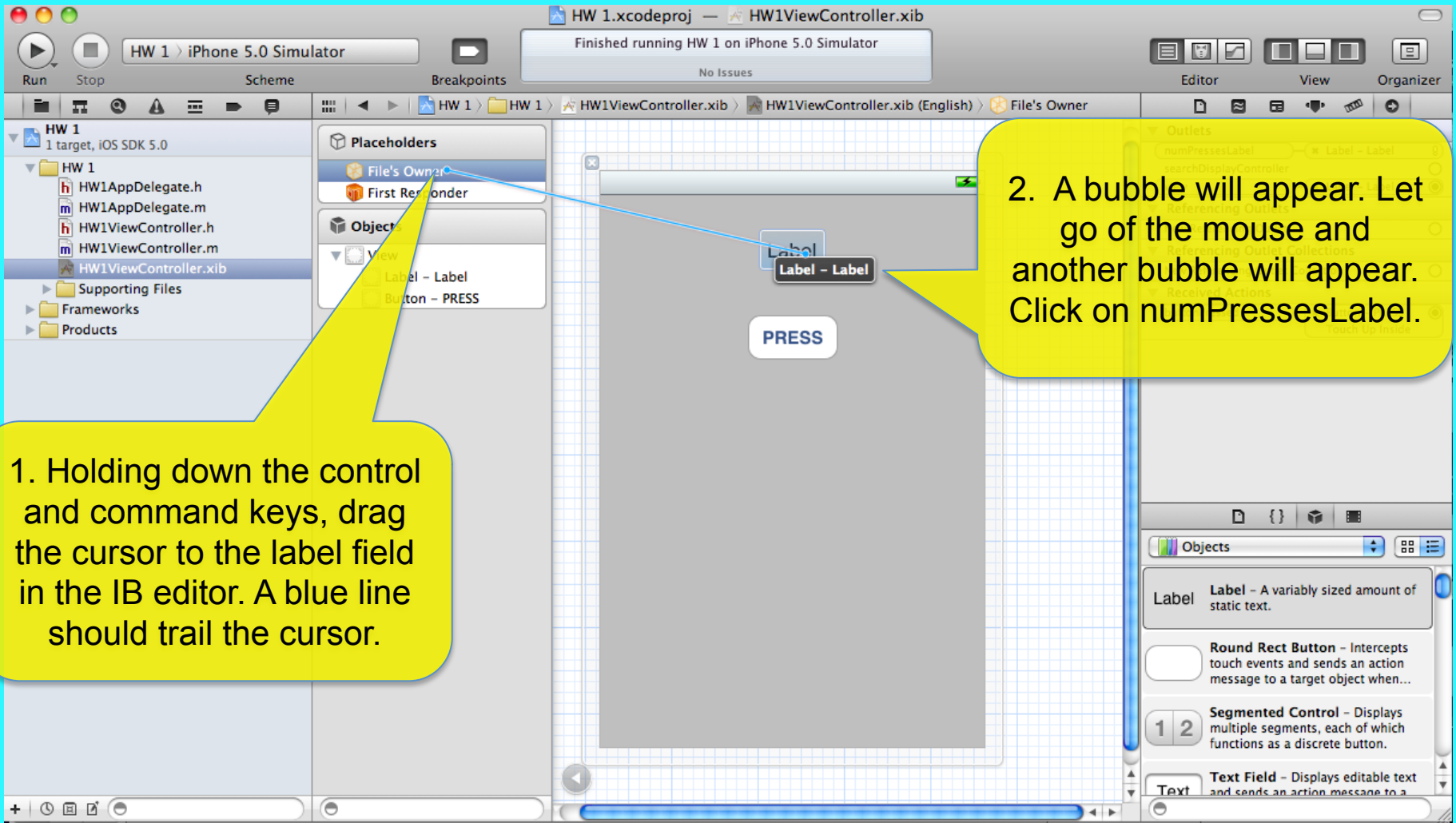
Objective-c does not have constructors. Constructing an object is a 2-step process. First we allocate space for it and then we initialize it.

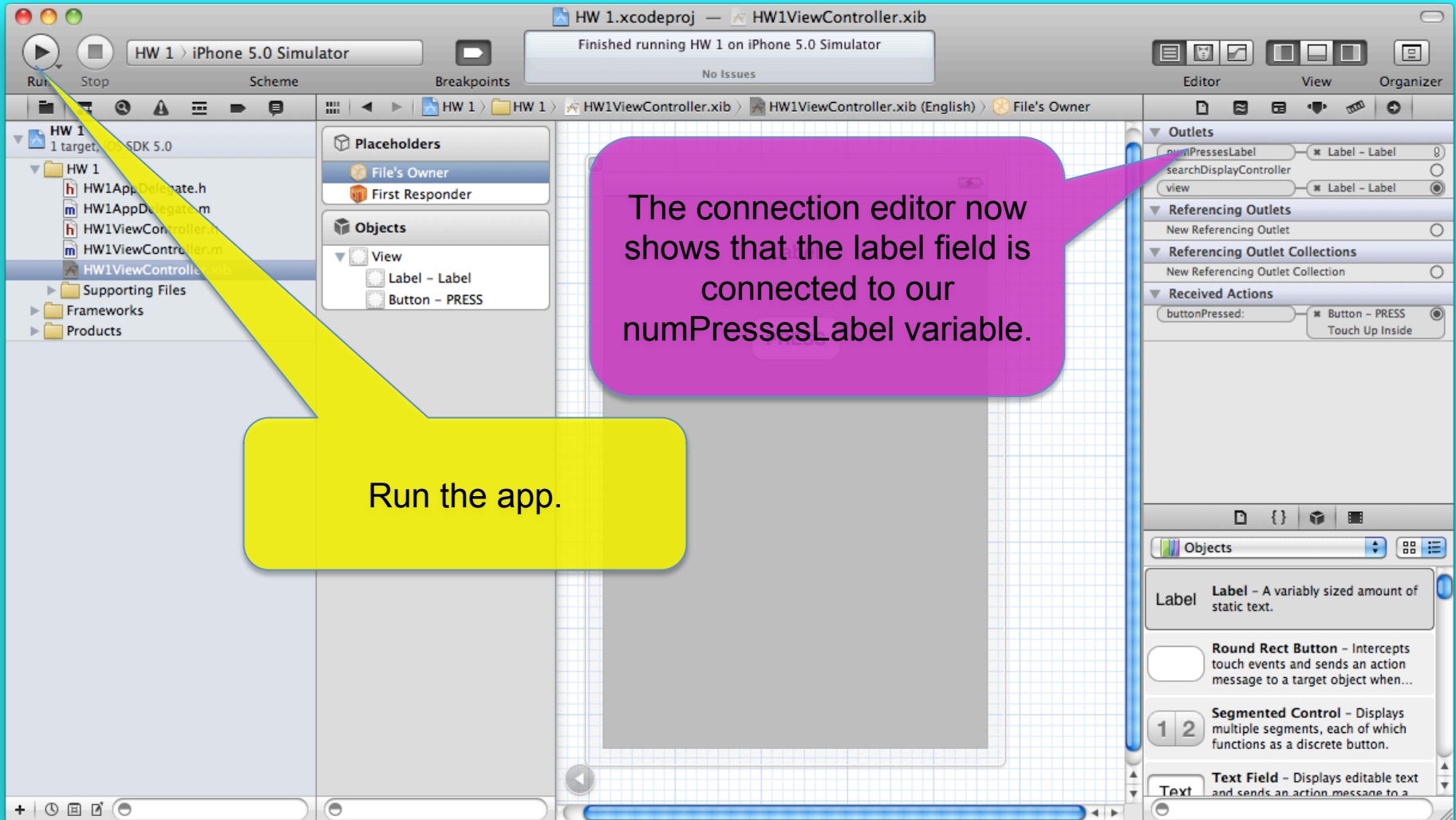


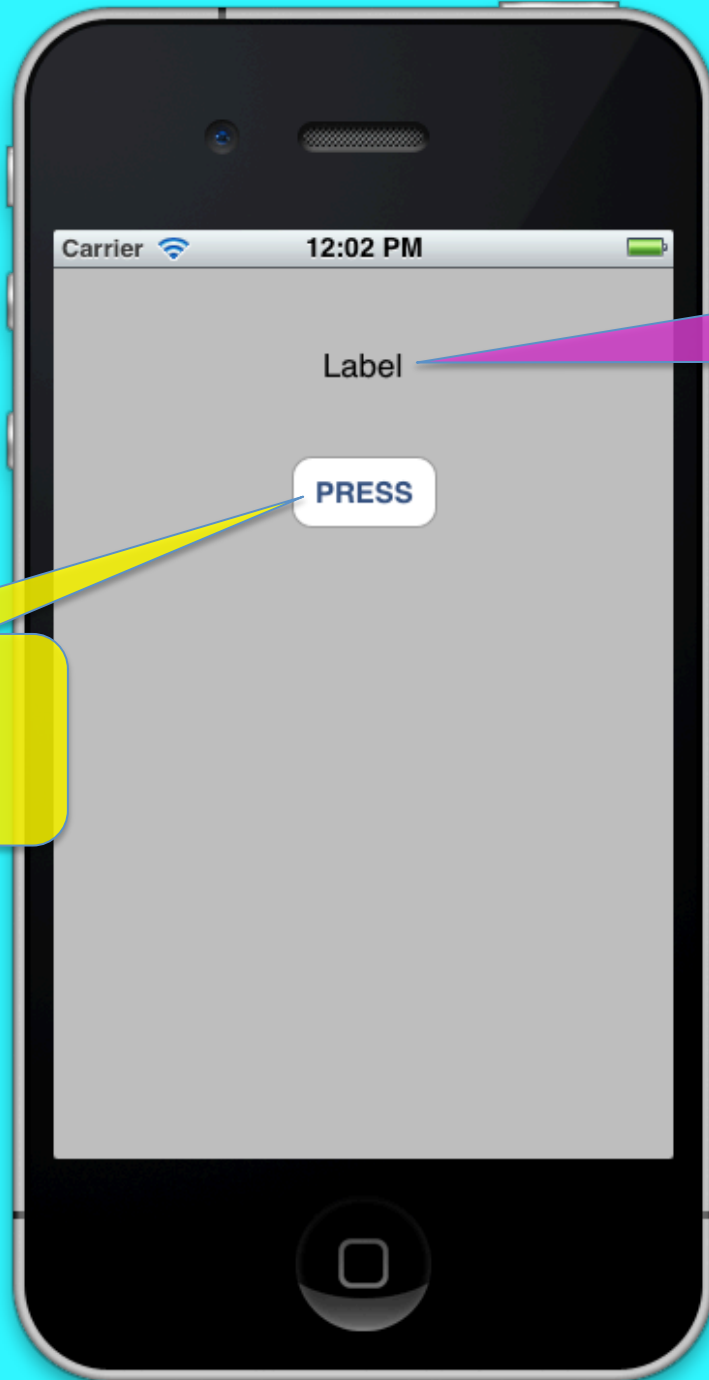


1. Open the HW1ViewController.xib file.

2. Select File's Owner.



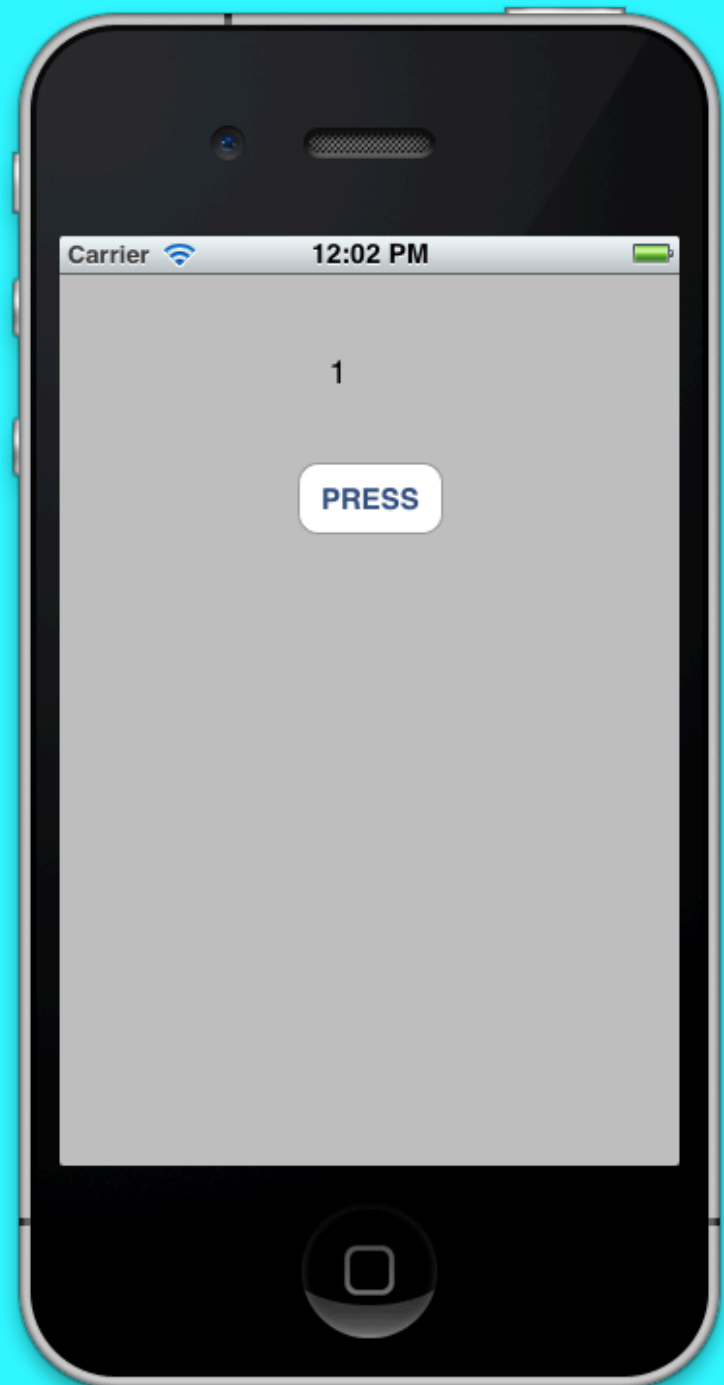




This isn't good! We'll fix it in a minute.

Click on PRESS

Now change your app so it starts with a 0 rather than the word Label. (There are several ways to do this. Take your pick.)



Zip up your project folder and
move it to your wiki page for
cs121.