

COEN 168/268

Mobile Web Application Development

Introduction to iOS Development

Peter Bergström (pbergstrom@scu.edu)

Santa Clara University



Introduction to iOS Development

A Brief History

- iOS was released on the first iPhone in 2007
- Since then, there have been 7 versions of iOS
- iPhone introduced in 2007
- iPad introduced in 2010



The App Store

- Opened on July 10th, 2008 in conjunction with the iPhone 3G and iOS 2.0.1
- Allows developers to sell apps for a price or free
- Over 1.2 million apps
- Over 75 billion downloads
- For **only** \$99 a year, you can be a developer submitting apps

From Wikipedia

The iOS SDK

- Using XCode, developers can create iOS apps for iPhone and iPad devices
- Currently, the latest released version of iOS is 7.1.2
- However, iOS 8 is right around the corner and rumored to be released mid-September
- Every release comes out with many new features that make developers excited

Getting XCode

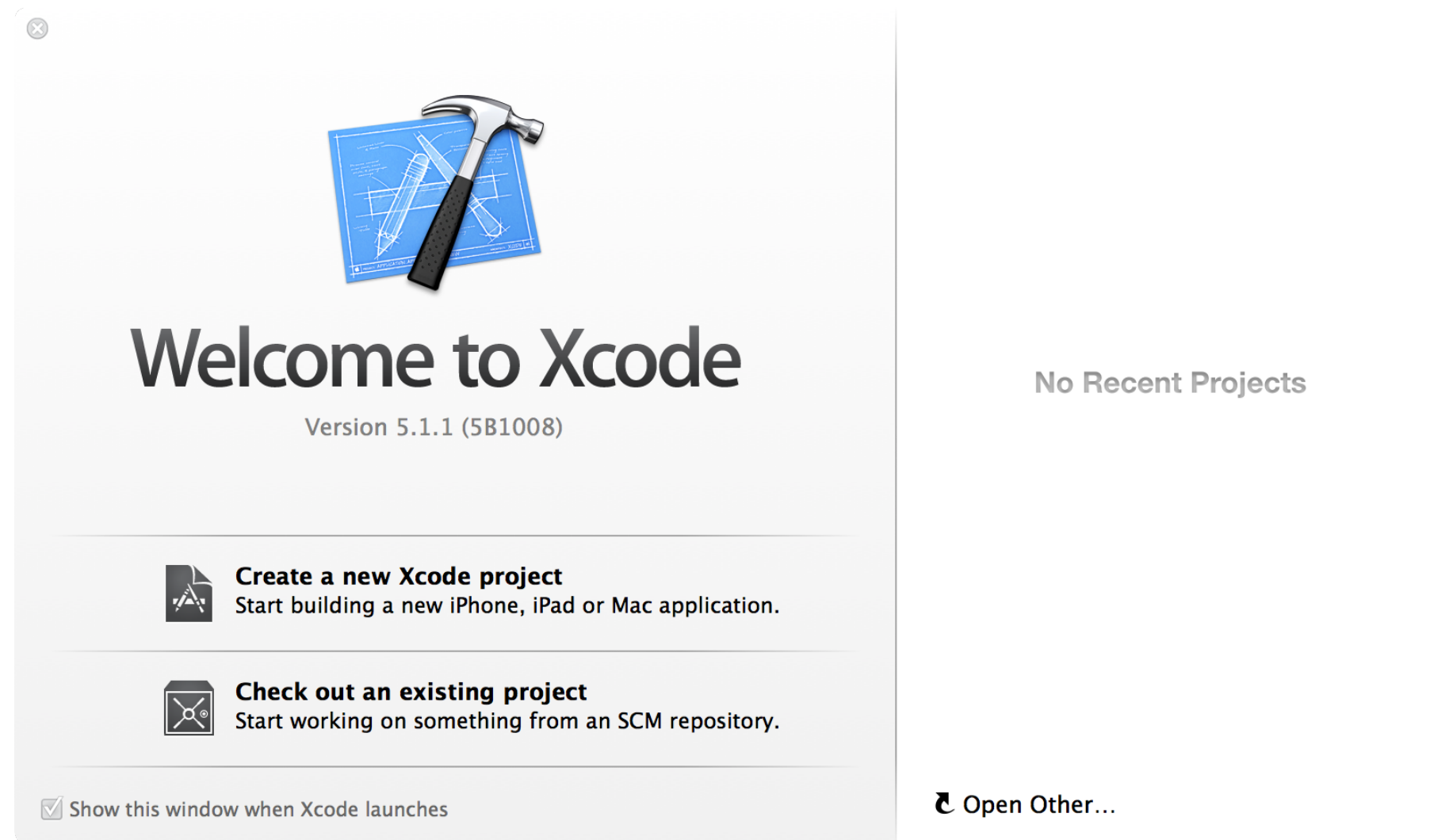


Getting XCode

- On your Mac, download Apple's XCode IDE from the App Store
- This is enough to get started developing and testing apps in the included iOS Simulator
- If you do not pay for the developer account, you cannot:
 - Build on device
 - Ship an app to the App Store

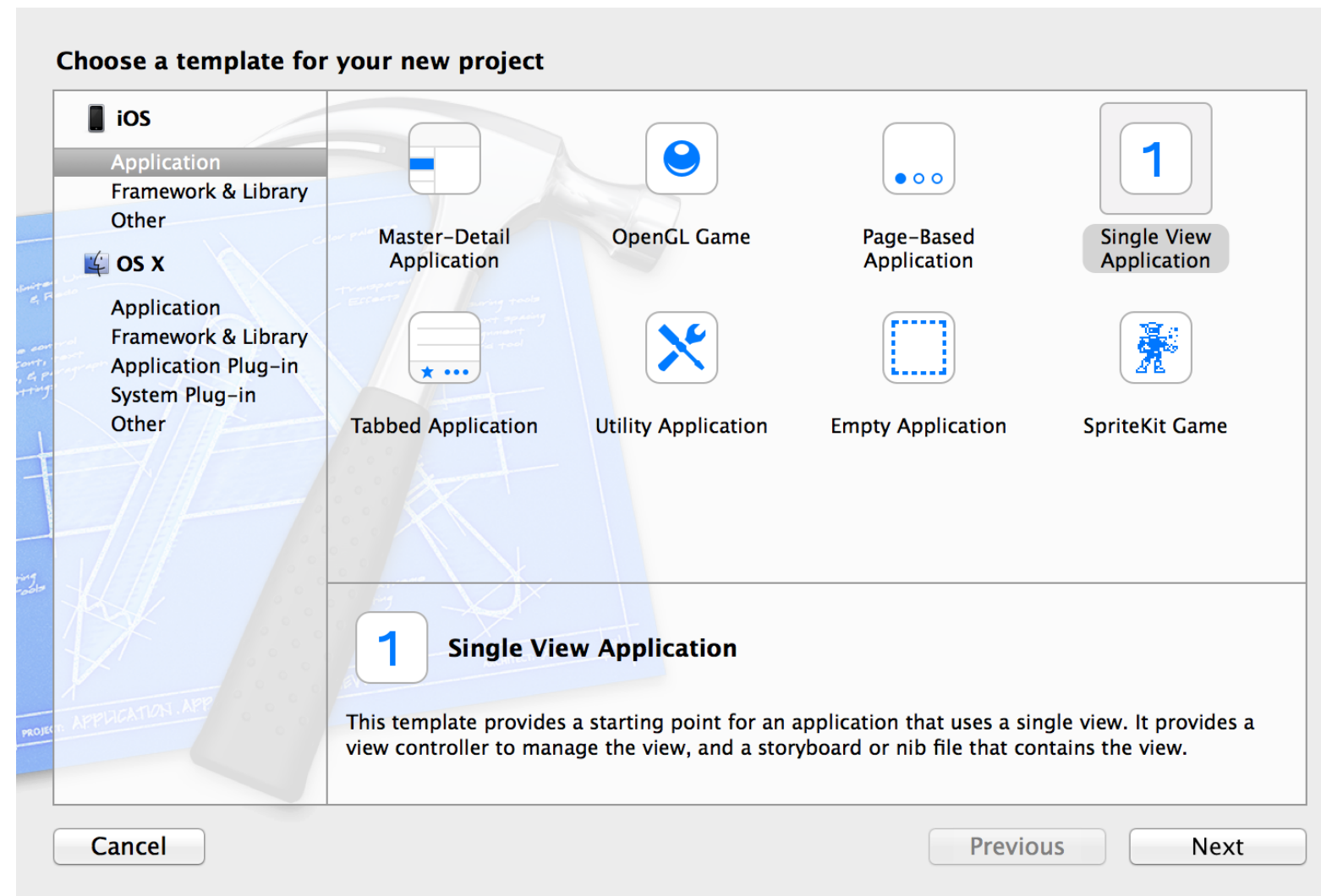
Let's Start XCode...

Create a new project



Or open an existing one.

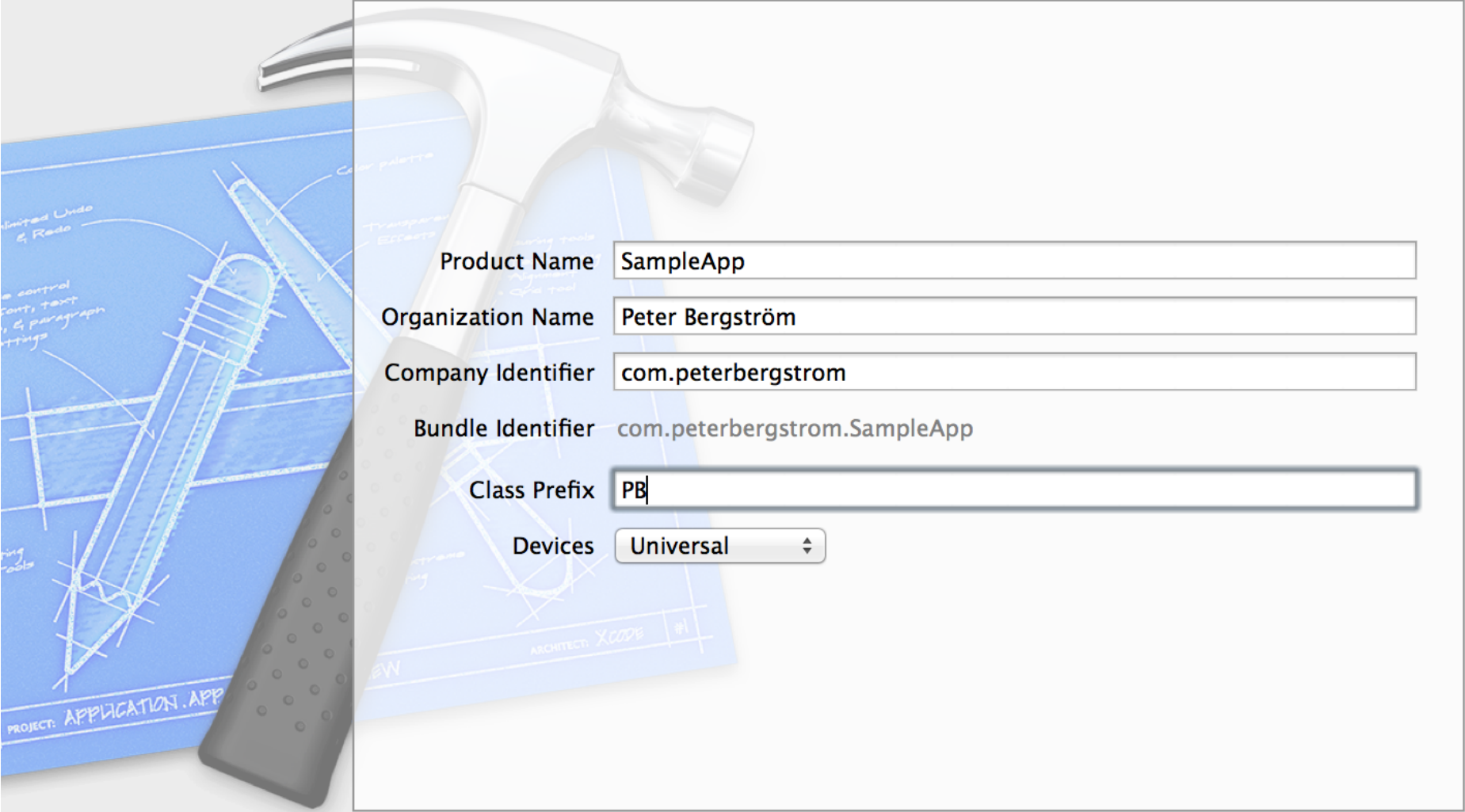
Choose a project template



We will choose "Single Page Application"

Configure project settings

Choose options for your new project:



Product Name

Organization Name

Company Identifier

Bundle Identifier

Class Prefix

Devices

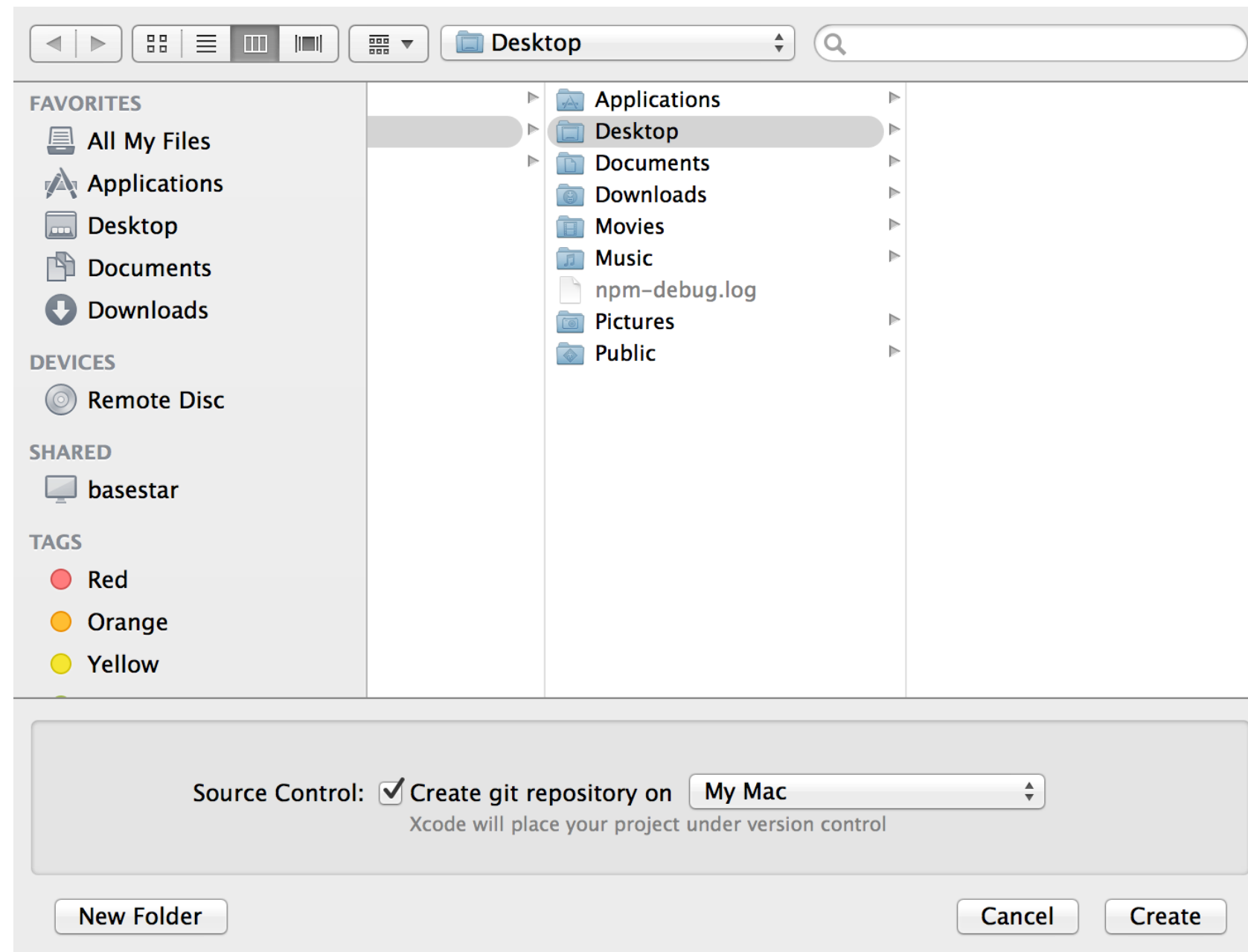
Configure project settings

- **Product Name:** The name of your app
- **Organization Name:** Can be your name or the name of your organization. Will be added to copyright statement in the source code.
- **Company Identifier:** A unique string that is used to create a bundle identifier. Apple recommends 'com.companyname'

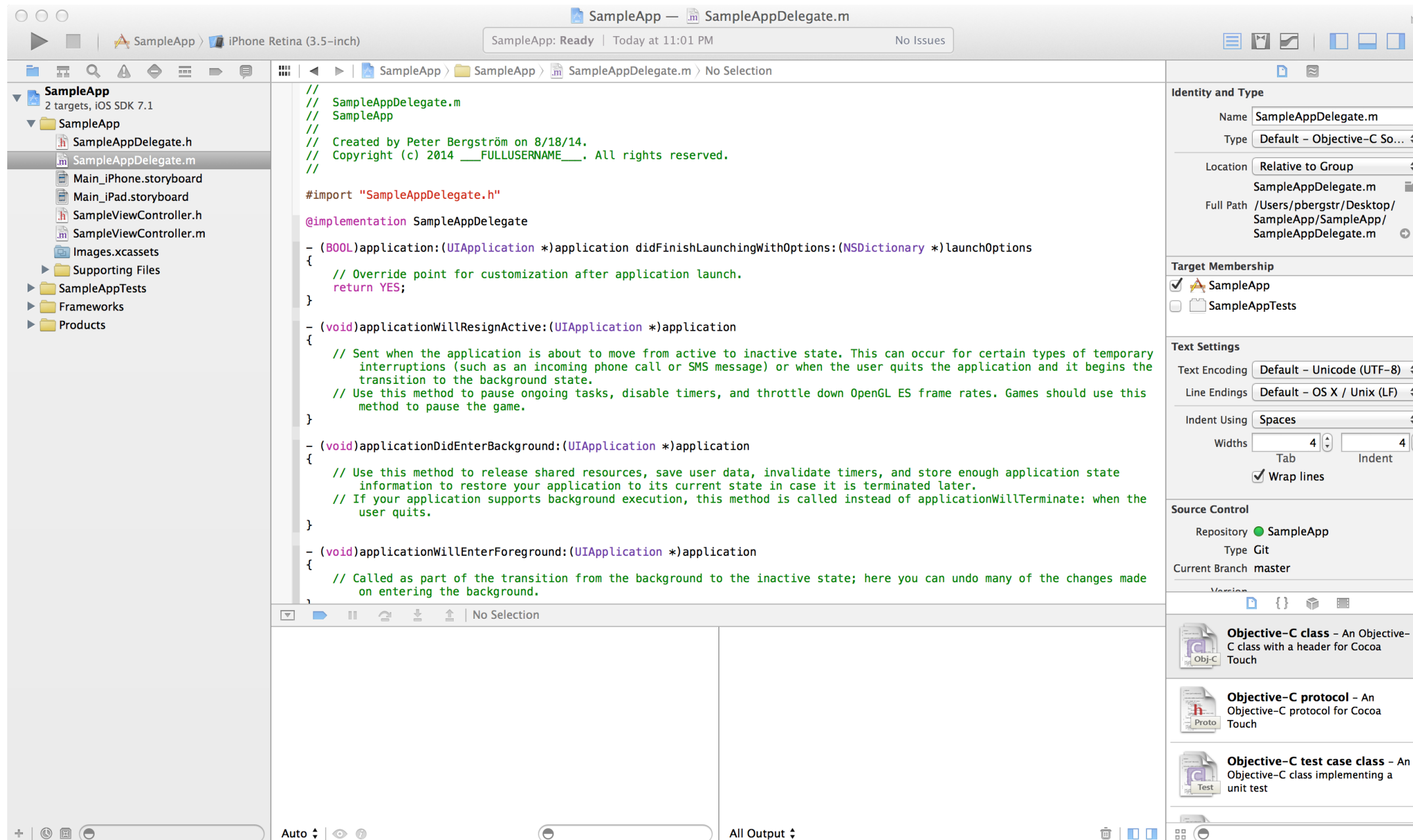
Configure project settings

- **Bundle Identifier:** The company identifier with the product name forms the Bundle Identifier.
- **Class Prefix:** The name given here will be prefixed with the name of the new classes that you create. This avoids potential name collisions between your code and other libraries.
- **Devices:** Specify whether you are targeting this application for iPhone, iPad or 'Universal' for both.

Pick a project destination

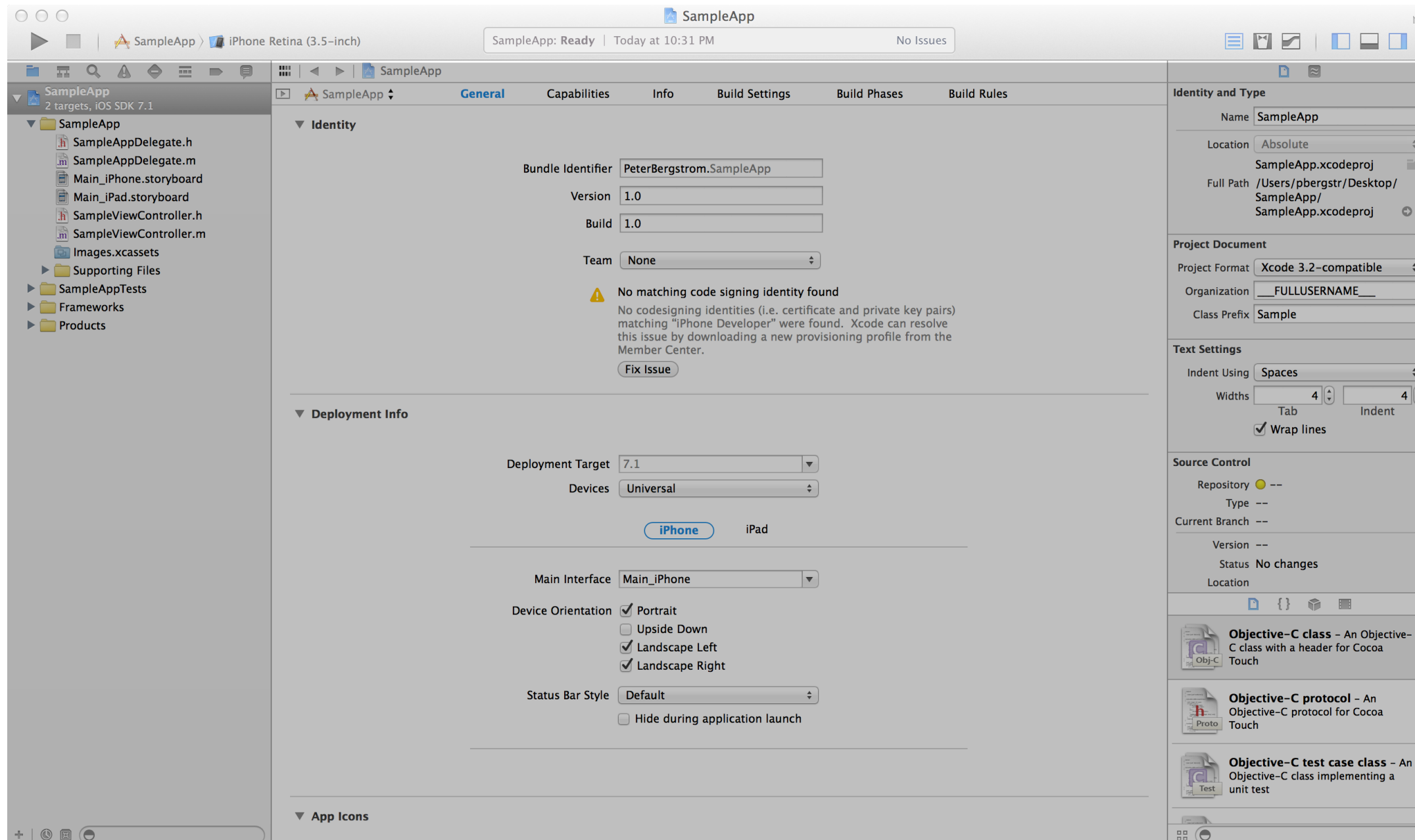


Voilà, you have a (blank) app



XCode UI Components

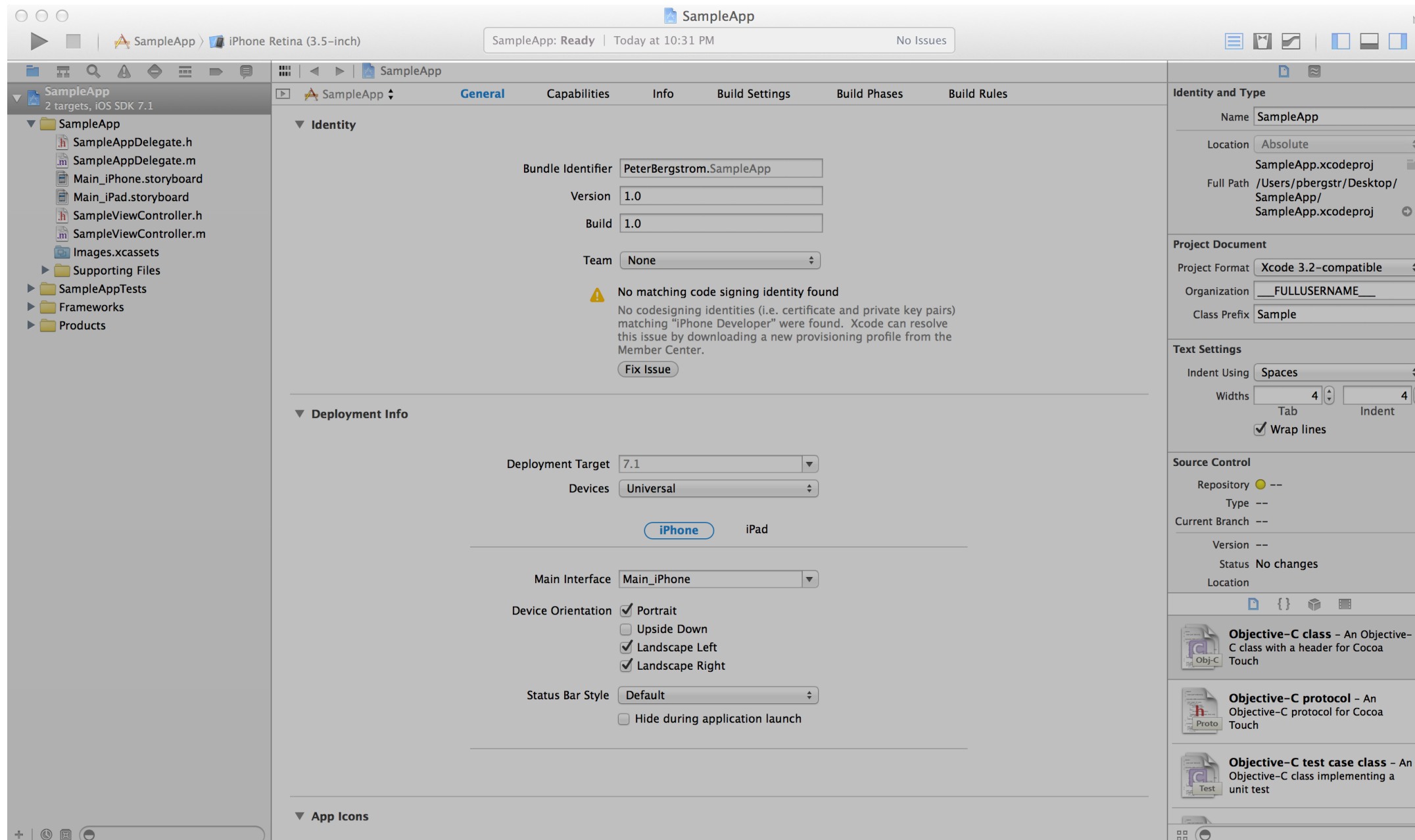
The Toolbar



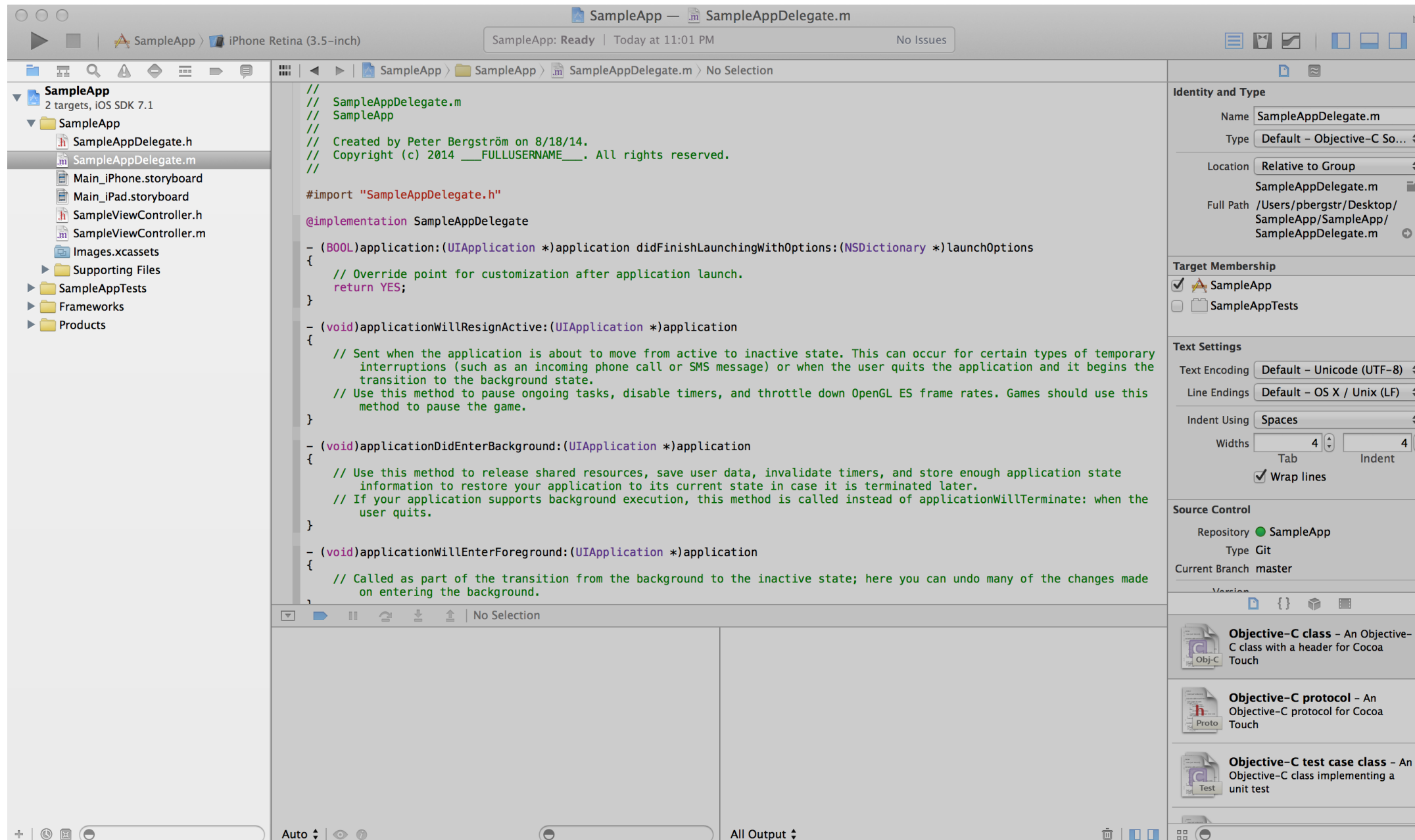
The Toolbar

- The icon that looks like a 'play' icon allows you to build the project
- The icon that looks like a 'stop' icon stops the build
- The drop down to the immediate right of those icons allows you to pick the build target and the device to build to
- The buttons on the top right toggle the different other visible areas that I will show next

The Toolbar



The Navigator



The Navigator

You can:

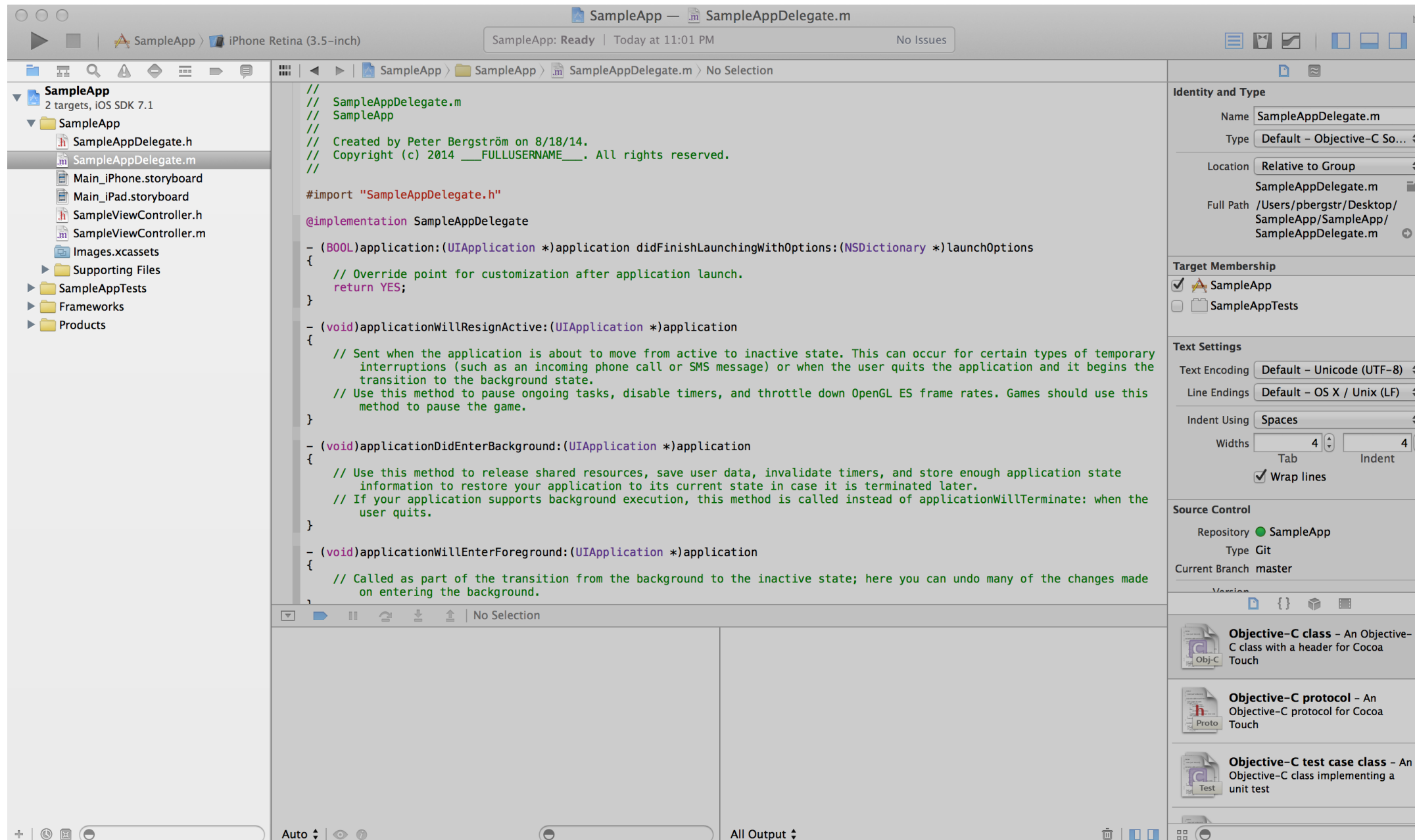
- Browse the source code
- Browse symbols
- Find and replace in the source code
- View build issues, warnings, and errors

The Navigator

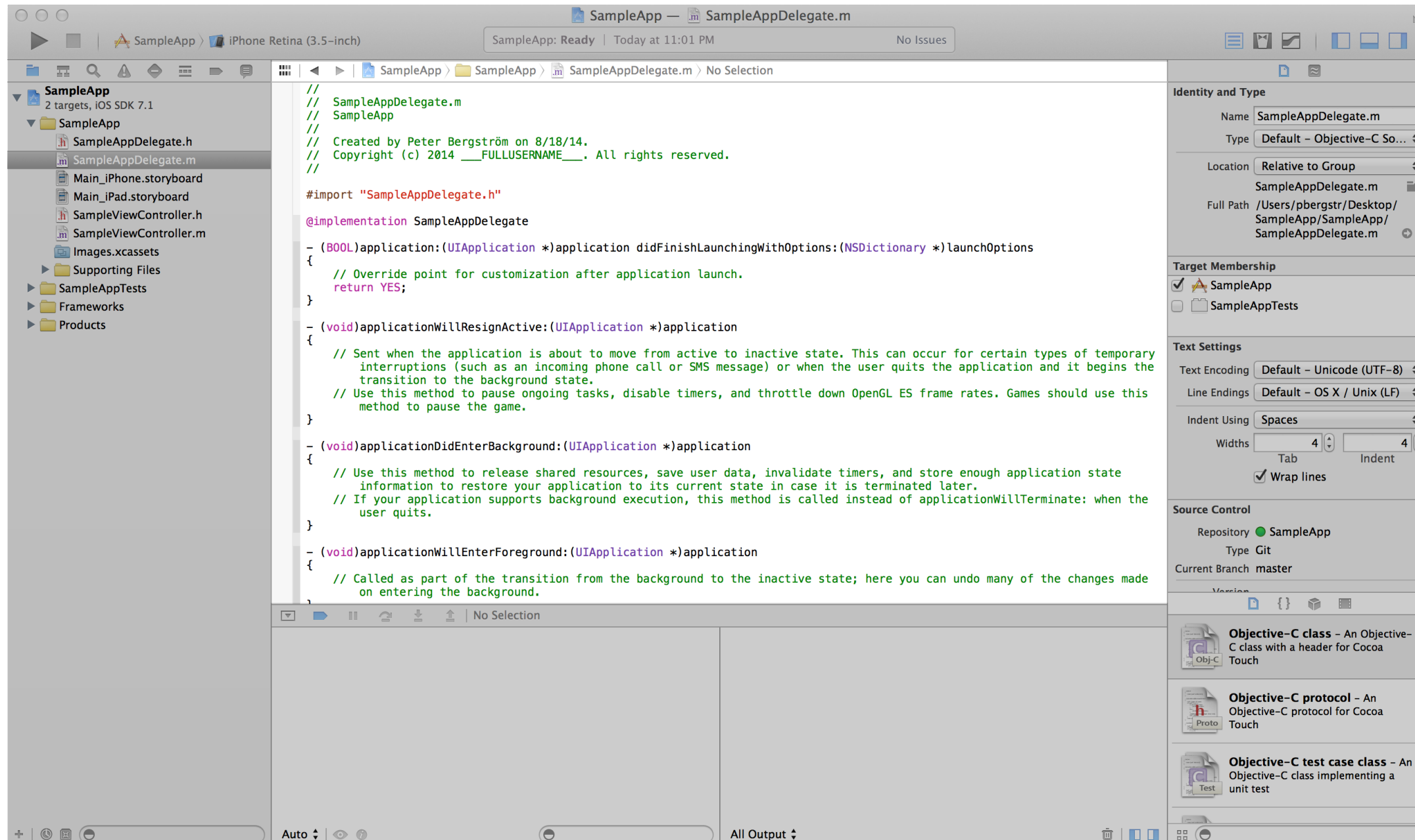
You can also:

- View tests
- Manage your debugging session
- Set and manage break points
- Show the log navigator

The Navigator



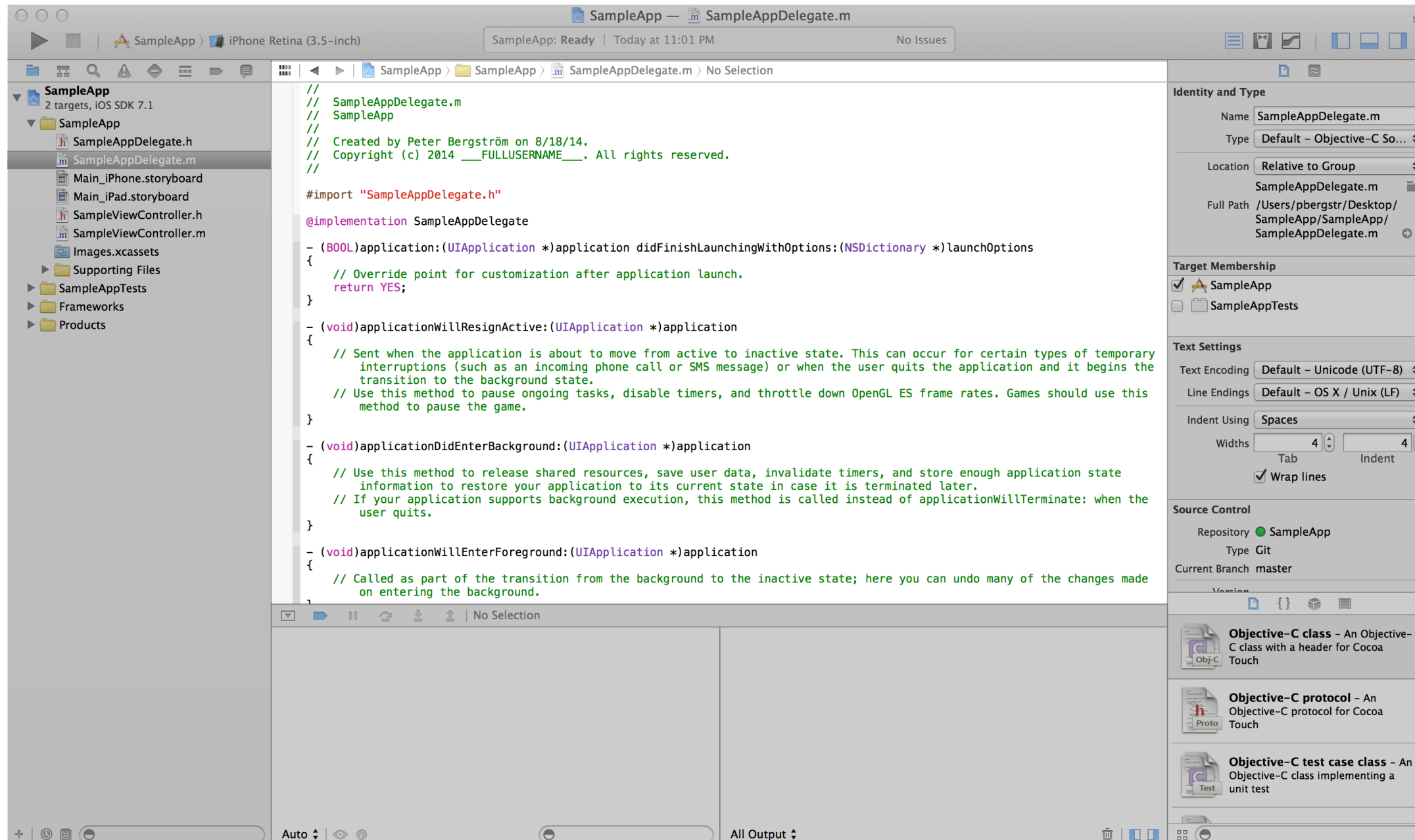
The Main View



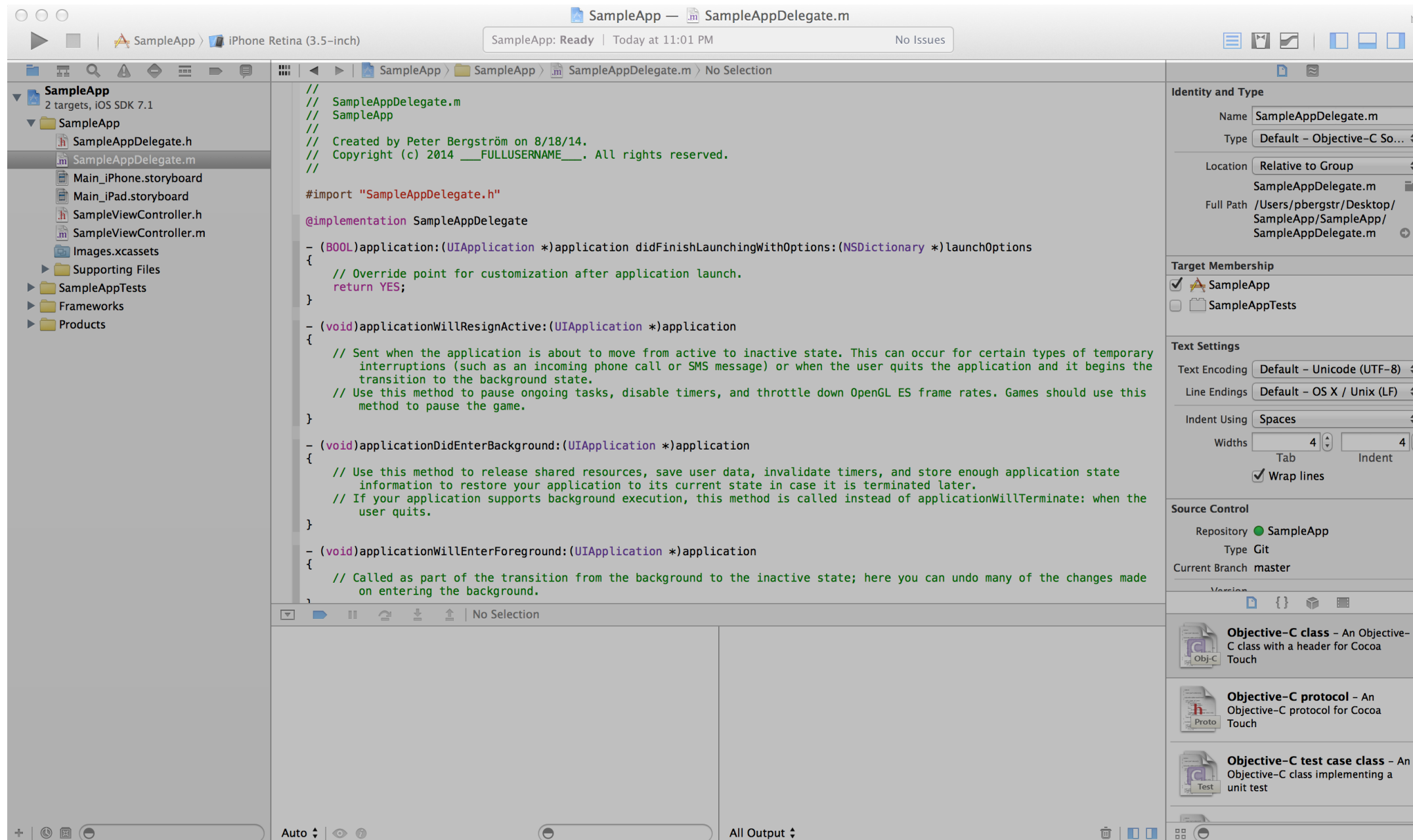
The Main View

- The main view is where you write your code
- Can be customized with different colors, etc
- At the top of the view, you can browse methods inside of the file that you are currently editing

The Main View



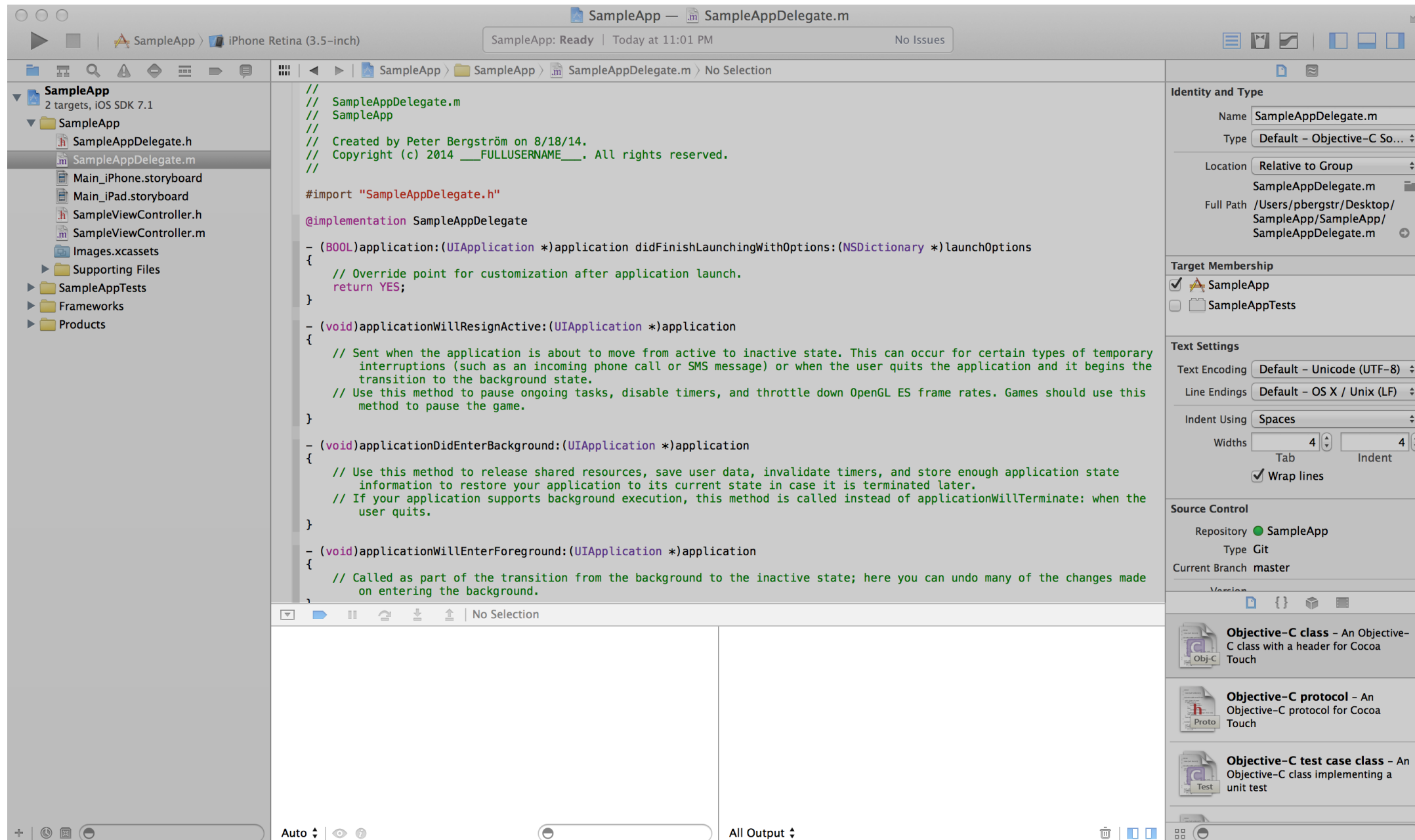
The Inspector



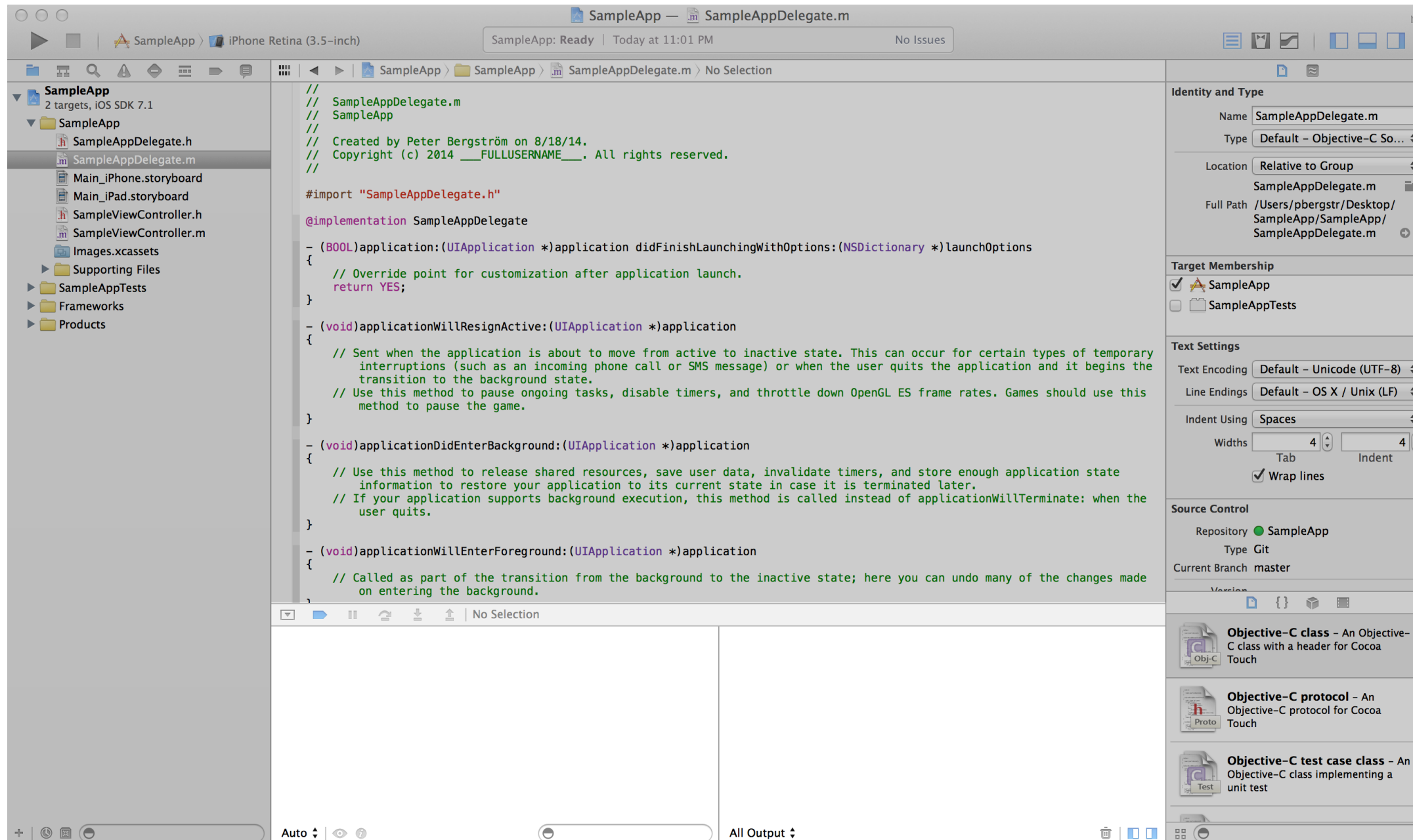
The Inspector

- Provides a set of contextual tabs where you can set properties for the file that you are currently viewing in the main view
- When you are writing code, it is not heavily used, but when you are updating interface files, it is where you do the majority of your work
- I recommend hiding it if you are not updating an Interface Builder file (.xib file)

The Debug Area



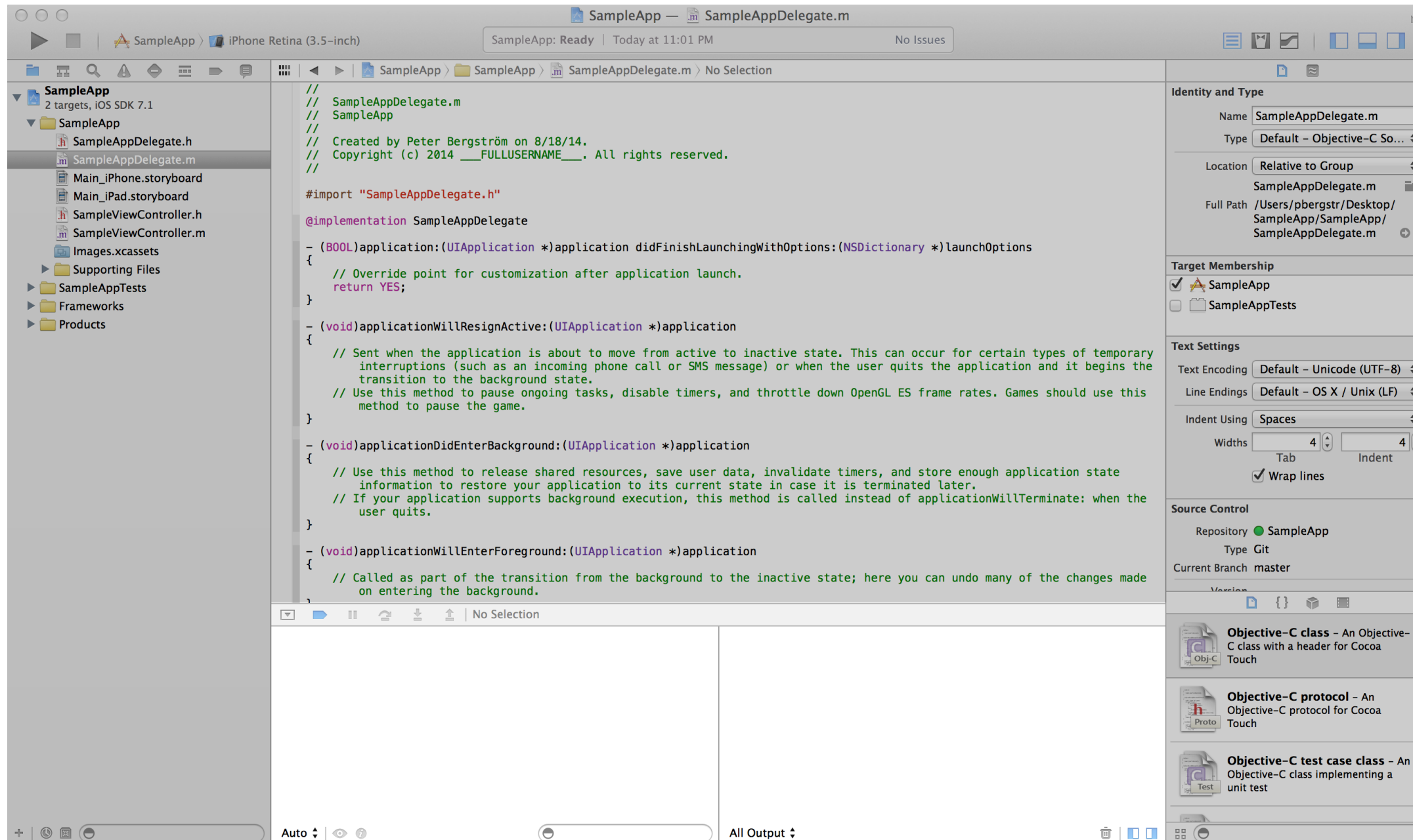
The Debug Area



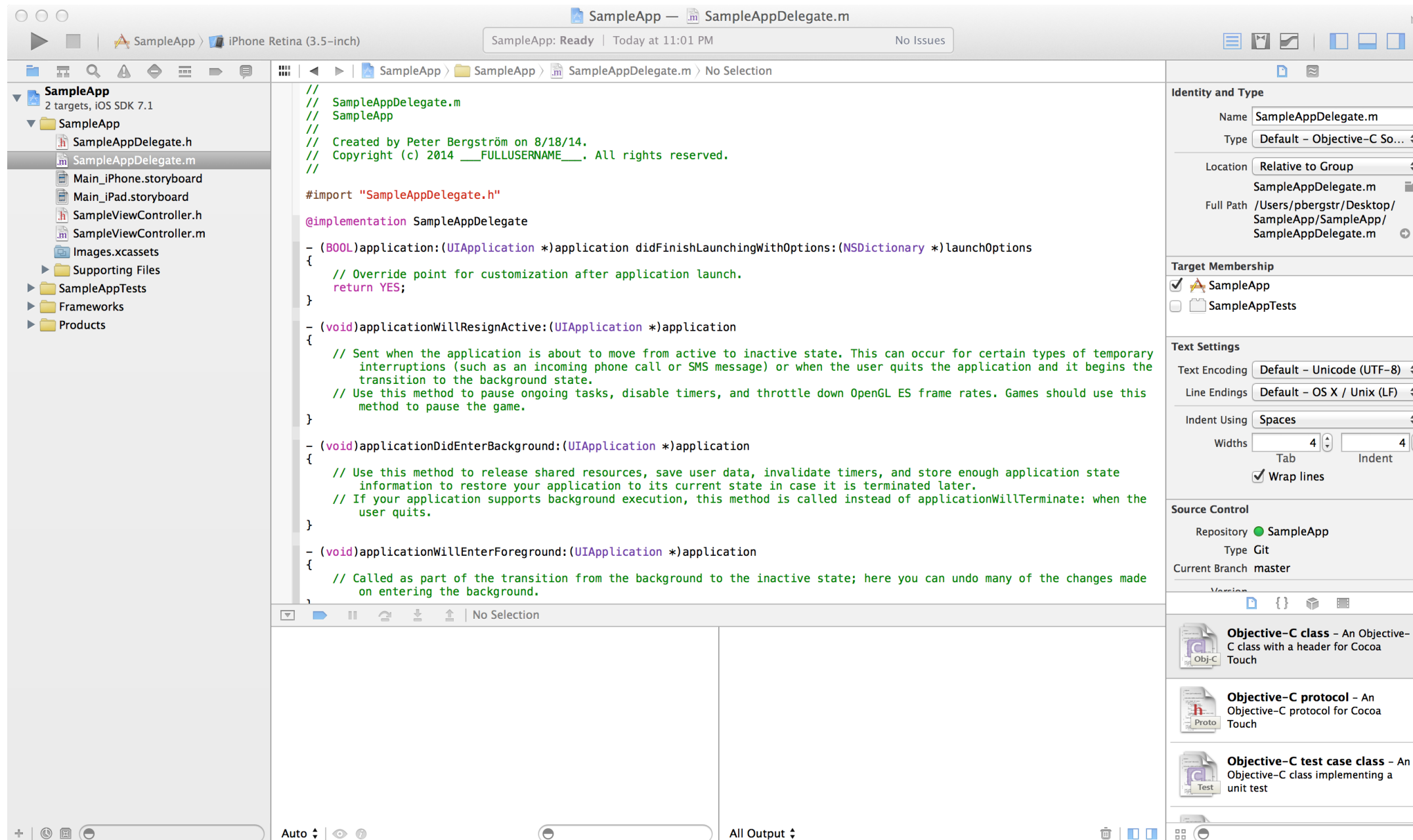
The Debug Area

- On the top, allows you to manage stepping through code when stopped at a break point
- Allows you to inspect objects and properties when paused in the debugger on the left
- Allows you to view logs on the right

The Debug Area



That's XCode in a nutshell



Building and running you app

- If there are errors building, this is when you'll find out
- Select a simulator or device to target
- Then click the 'play' icon to build the project
- The simulator will automatically open and the app will appear

Running the blank app



Pretty boring, so let's show something more interesting running

How to you write an iOS app?

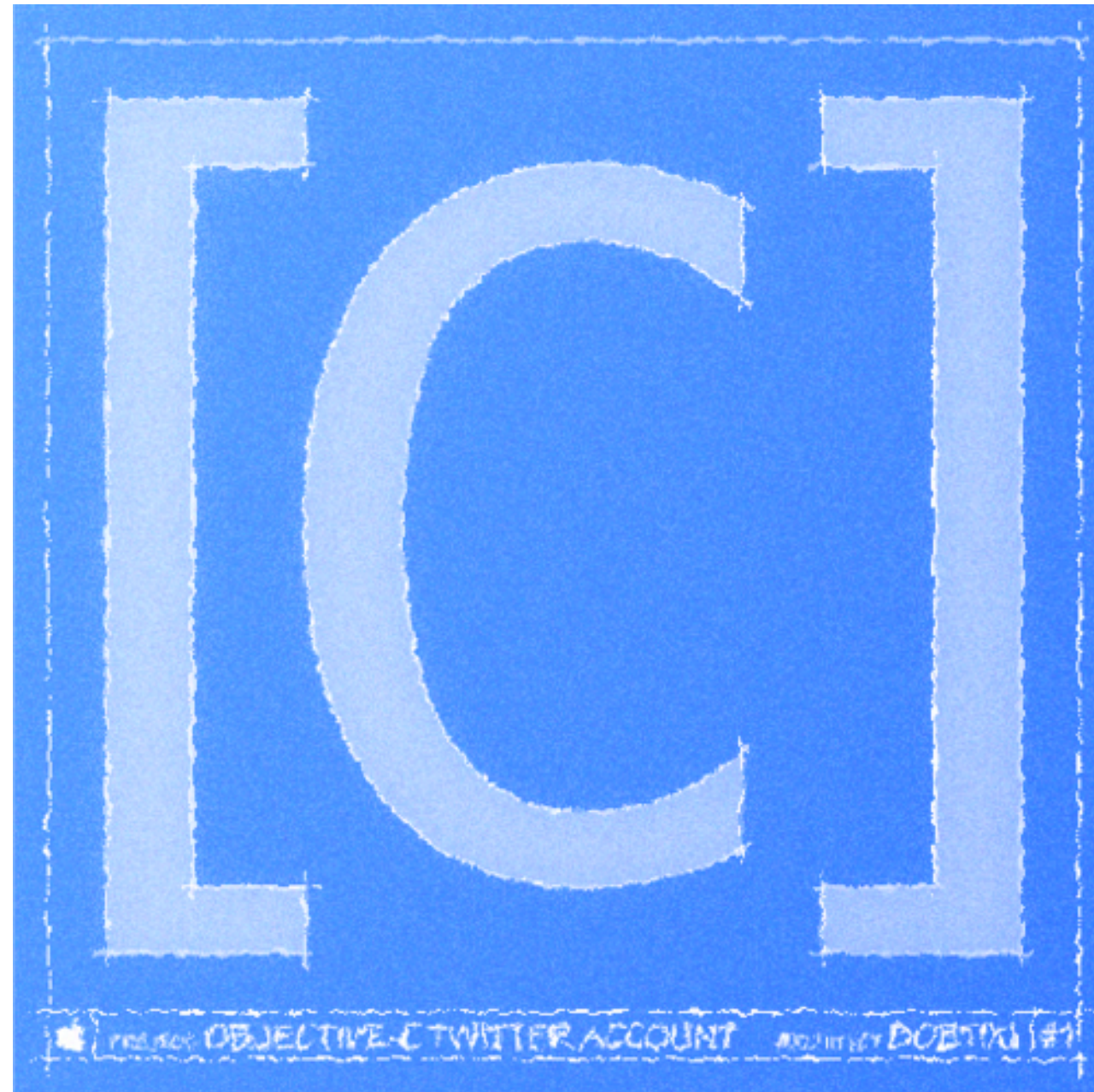
As of iOS 8, there are two languages

Objective-C

Swift

Both implement Cocoa and CocoaTouch, the UI libraries used for Mac and OS X development

Objective-C



Objective-C History

- Object oriented programming language
- Originated in the 1980s
- Popularized by NeXT, founded by Steve Jobs then acquired by Apple in 1996
- It is now the programming language used by Apple for Mac OS X and iOS development
- At this point, it is considered a dated programming language

Objective-C Features

- A thin layer over C and a superset of C so you can write C inside your iOS apps if you want
- Adds Smalltalk style messaging, interfaces and implementations, protocols, dynamic typing, blocks, forwarding, categories, posing, literals, subscripting, and more...
- More recently, Automatic Reference Count (ARC) has been added to ease memory management
- Each class has matching interface (.h) and implementation (.m) files

Objective-C Syntax

Remember how to create a `Car` class in
JavaScript?

Car Class in JavaScript

```
1 function Car(make, model, year) {  
2     this.make = make;  
3     this.model = model;  
4     this.year = year;  
5  
6     this.description = function() {  
7         return this.year + " " + this.make + " " + this.model;  
8     };  
9  
10    return this;  
11 }  
12  
13 var mustang = new Car("Ford", "Mustang", 2015);  
14 console.log(mustang.description());
```

Car Class in Objective-C

Interface: Car.h

```
1 #import <Foundation/Foundation.h>
2
3 @interface Car : NSObject
4
5 @property (nonatomic, copy) NSString *make;
6 @property (nonatomic, copy) NSString *model;
7 @property (nonatomic, assign) NSInteger year;
8
9 -(id)initWithMake:(NSString *)make model:(NSString *)model year:(NSInteger)year;
10 -(NSString *)description;
11
12 @end
```


Implementation: Car.m

```
1 #import "Car.h"
2
3 @implementation Car
4
5 -(id)initWithMake:(NSString *)make model:(NSString *)model year:(NSInteger)year {
6     self = [super init];
7     if (self) {
8         _make = make;
9         _model = model;
10        _year = year;
11    }
12    return self;
13 }
14
15 -(NSString *)description {
16     return [NSString stringWithFormat:@"%s %s %d", self.make, self.model, self.year];
17 }
18 }
19
20 @end
```

Creating a Car instance

```
#import "Car.h"
```

```
// somewhere down in the code..
```

```
Car *mustang = [[Car alloc] initWithMake:@"Ford" model:@"Mustang" year:2014];  
NSLog(@"%@", [mustang description]);
```

As you can see, Objective-C is pretty verbose,
but it grows on you



Introducing Swift

Swift

- Introduced Summer 2014 at WWDC
- Modern replacement for Objective-C that works side by side with Objective-C in your app
- Design for safety by adding safe patterns and eliminating unsafe classes
- Fast and powerful
- Has interactive playgrounds that allow you to write code and see your changes live

Implementation: Car.swift

```
1 class Car {
2     let make: String
3     let make: String
4     let model: Int
5
6     init:(make:String, model:String, year:Int) {
7         self.make = make
8         self.model = model
9         self.year = year
10    }
11
12    func description() {
13        return year + " " + make + " " + model
14    }
15 }
```

Creating a Car instance

```
// somewhere down in the code...
```

```
let mustang = Car(make: "Ford", model: "Mustang", year: 2014)  
println(mustang.description())
```

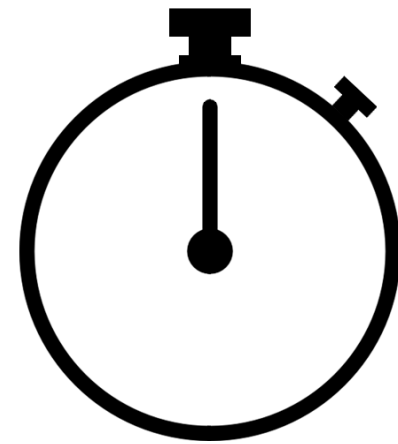
How Do You Construct UI?

- Manually using Objective-C or Swift
- Using Interface Builder and .xib files
- Storyboards

Demo

Calculator Demo in XCode

LaborTime



Demo

How do you integrate a mobile web
app into a native iOS app?

There are several ways

- Use a `UIWebView` to load your app from a web server
- Save your JavaScript source code locally and load a `UIWebView`
- Use PhoneGap to help you bundle your web app into a native iOS app
 - Allows you easy access to a lot of hardware integration without a lot of manual work

Using a UIWebView and loading a web server

- This is a lightweight way to do it and requires minimal code
- It is easy to update your app as the user would hit your URL all the time to access the app
- However, your users *have* to have an internet connection to access it which is some times bad

Using a UIWebView and loading a web server

```
1 // In a view controller in a single page app..
2
3 - (void)viewDidLoad {
4     [super viewDidLoad];
5
6     UIWebView *webView = [[UIWebView alloc] initWithFrame:self.view.frame];
7     NSURL *clockAppUrl = [NSURL URLWithString:@"http://coen268.peterbergstrom.com/clock"];
8
9     [webView loadRequest:[NSURLRequest requestWithURL:clockAppUrl]];
10    [self.view addSubview:webView];
11
12 }
```

Demo

Loading The Ember Clock App in a
UIWebView from a web server

Code is available on the lectures page

Demo

Loading The CalculatorDemo App in a
UIWebView from a web server

Code is available on the lectures page

Using a UIWebView & loading your app locally

- This is more complicated than loading from a server
- You can't update your JS code without submitting an new app to the store
- However, it will work offline, which is a huge benefit
- Also, it will load faster because you don't have to download source code and assets
- **If loading an Ember.js app, set `locationType` to "none"**

Demo

Loading The CalculatorDemo App in a
UIWebView locally

Code is available on the lectures page

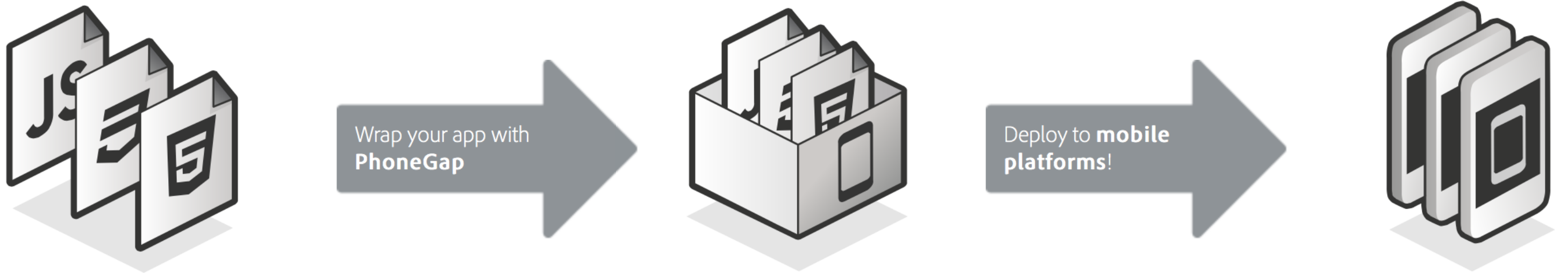
PhoneGap

- PhoneGap is an open source framework for quickly building cross-platform mobile apps using HTML5, Javascript and CSS.
- Makes it easy to deploy your app to multiple platforms, without writing a lot of code.
- Instead of writing code, you can automate the creation of these apps on various platforms.
- Makes it easy to also interact with the hardware of the devices

PhoneGap Hardware Features

	iPhone / iPhone 3G	iPhone 3GS and newer	Android	Blackberry OS 6.0+	Blackberry 10	Windows Phone 8	Ubuntu	Firefox OS
Accelerometer	✓	✓	✓	✓	✓	✓	✓	✓
Camera	✓	✓	✓	✓	✓	✓	✓	✓
Compass	X	✓	✓	X	✓	✓	✓	✓
Contacts	✓	✓	✓	✓	✓	✓	✓	✓
File	✓	✓	✓	✓	✓	✓	✓	X
Geolocation	✓	✓	✓	✓	✓	✓	✓	✓
Media	✓	✓	✓	X	✓	✓	✓	X
Network	✓	✓	✓	✓	✓	✓	✓	✓
Notification (Alert)	✓	✓	✓	✓	✓	✓	✓	✓
Notification (Sound)	✓	✓	✓	✓	✓	✓	✓	✓
Notification (Vibration)	✓	✓	✓	✓	✓	✓	✓	✓
Storage	✓	✓	✓	✓	✓	✓	✓	✓

PhoneGap Development Pipeline



Installing PhoneGap

```
$ sudo npm install -g phonegap
```

Basic Usage

```
$ phonegap create my-app
```

```
$ cd my-app
```

```
$ phonegap run ios
```

For more info, go to <http://phonegap.com>

Making your app work with PhoneGap

- After you create your app, go into the directory.
- cd into the www directory
- Put your HTML, CSS, and JavaScript files there replacing the standard files
- Make sure to add this line in your index.html file:

```
<script type="text/javascript" src="cordova.js"></script>
```

Demo

Loading The CalculatorDemo App in PhoneGap

Code is available on the lectures page

COEN 168/268

Mobile Web Application Development

Introduction to iOS Development

Peter Bergström (pbergstrom@scu.edu)

Santa Clara University