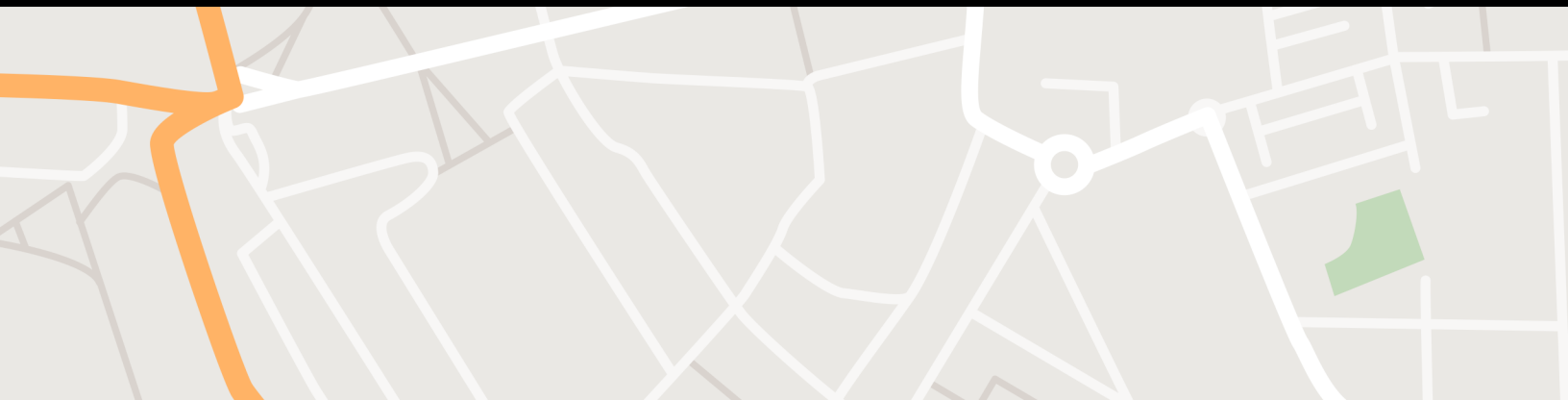
Building Mapping Apps for iOS With Swift

Jeff Linwood



Building Mapping Apps for iOS With Swift

Jeff Linwood

This book is for sale at http://leanpub.com/buildingmappingappsforioswithswift

This version was published on 2017-09-09

This is a Leanpub book. Leanpub empowers authors and publishers with the Lean Publishing process. Lean Publishing is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

Contents

Introduction

The purpose of this book is to introduce you to mapping and location technology in iOS. In the first chapter, we will build a very simple mapping app using Apple's MapKit framework, which comes with iOS. The next chapters will introduce more complicated aspects of the location and map frameworks built into iOS, CoreLocation and MapKit. We'll finish up by creating projects with the Google Maps for iOS SDK and the MapBox SDK.

Prerequisites

This book won't be an introduction to iOS app development, but it is written for beginning and intermediate iOS programmers. You should be able to understand the Swift programming language, and also know the basic concepts of iOS development, such as View Controllers and Storyboards. You don't have to know anything about mapping or location yet, and we'll introduce more advanced concepts as we use them.

You will need a relatively recent Mac to do your development for the projects in the book. This book is written with Apple's XCode 8 integrated development environment (IDE). XCode is free for download from the Mac App Store, and runs on recent versions of OS X. You do not need an iOS device for the projects in this book, but it is a lot more interesting if you do have one, especially for the projects that can use your current location. You can sign up for an Apple developer account on Apple's web site if you want to run your projects on your own iOS devices - at the time of writing, this doesn't require paying Apple any money:

https://developer.apple.com/

If you want to publish your apps on the App Store, you will need to have a paid Apple Developer account. You will not need one to follow these development exercises, however.

Creating your first MapKit app

Our first iOS app will be very simple - only one screen that displays a map, and then has one pin on it, with the location of my city, Austin, Texas. Feel free to use your own town for this example, of course!

Our finished application

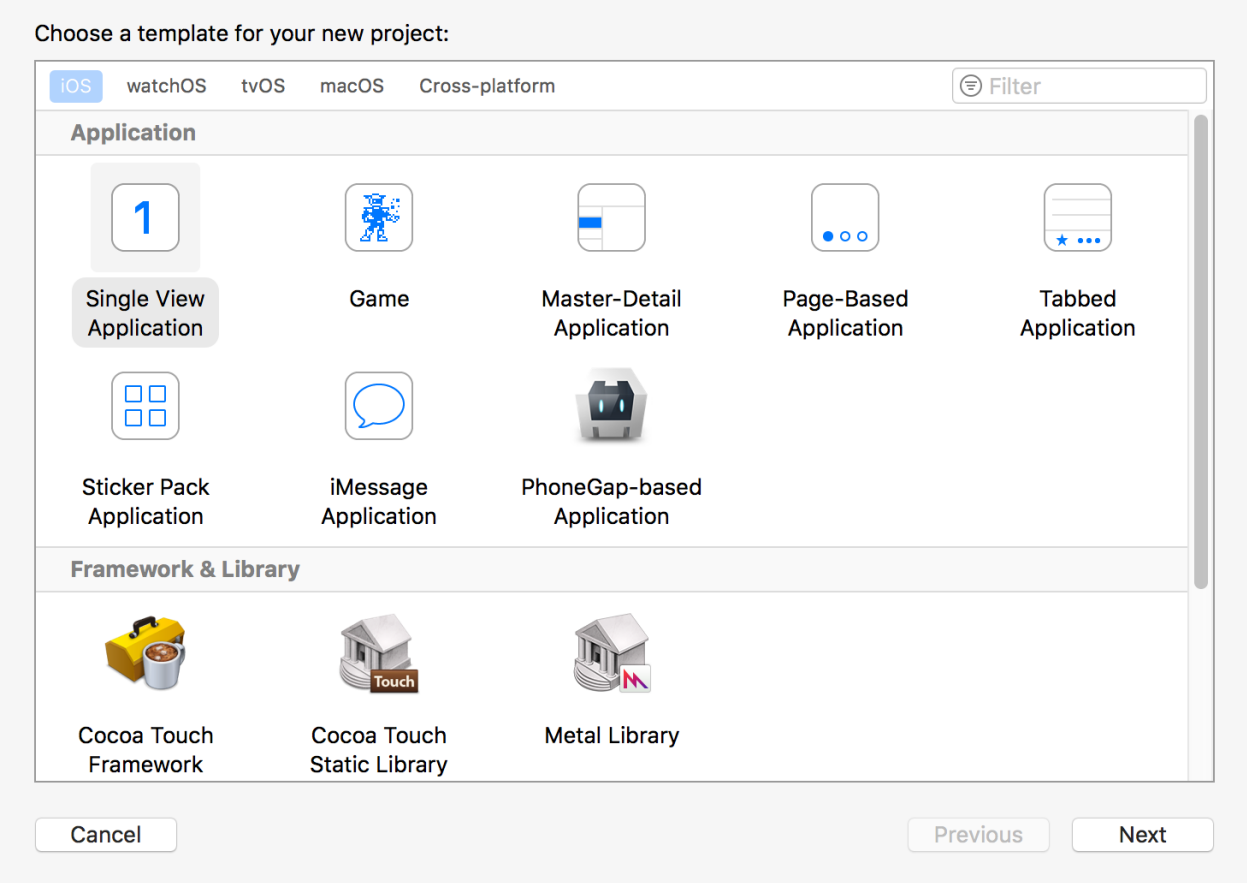
Getting Started

The first step is to make sure that you have a recent version of XCode (at the time of writing, XCode 8.2) installed on your Mac. If you're using earlier versions of XCode, this code may not compile, and you may not be able to follow directions.

We'll also be using the Swift programming language, instead of Apple's older programming language for iOS, Objective-C. Almost of all of this book would be directly applicable to an Objective-C application. The underlying application programming interfaces (APIs) used in iOS are the same.

The Swift language has been evolving since its first release. This book uses Swift 3, which is supported in XCode 8 and above.

Go ahead and open up XCode, and create a new application. We'll be creating a new Single View Application.



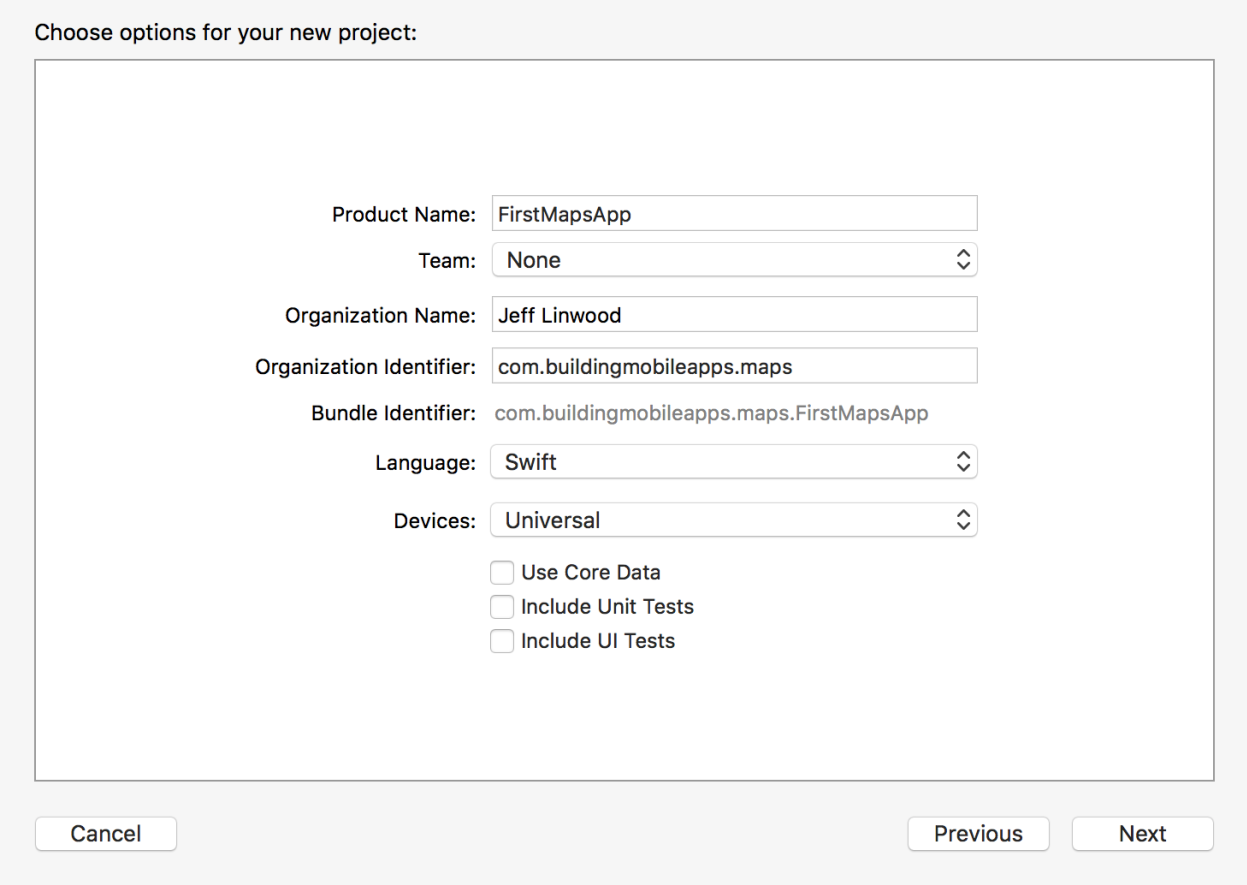
New project window

Click next, and then name your new project. I'm going to call it FirstMapsApp, and give it an organization identifier of com.buildingmobileapps.maps and use my name for the Organization Name.

Be sure to choose Swift as the Language, and make the app Universal - nothing we are doing in this app will restrict the app from running on an iPhone or iPad, as we need.

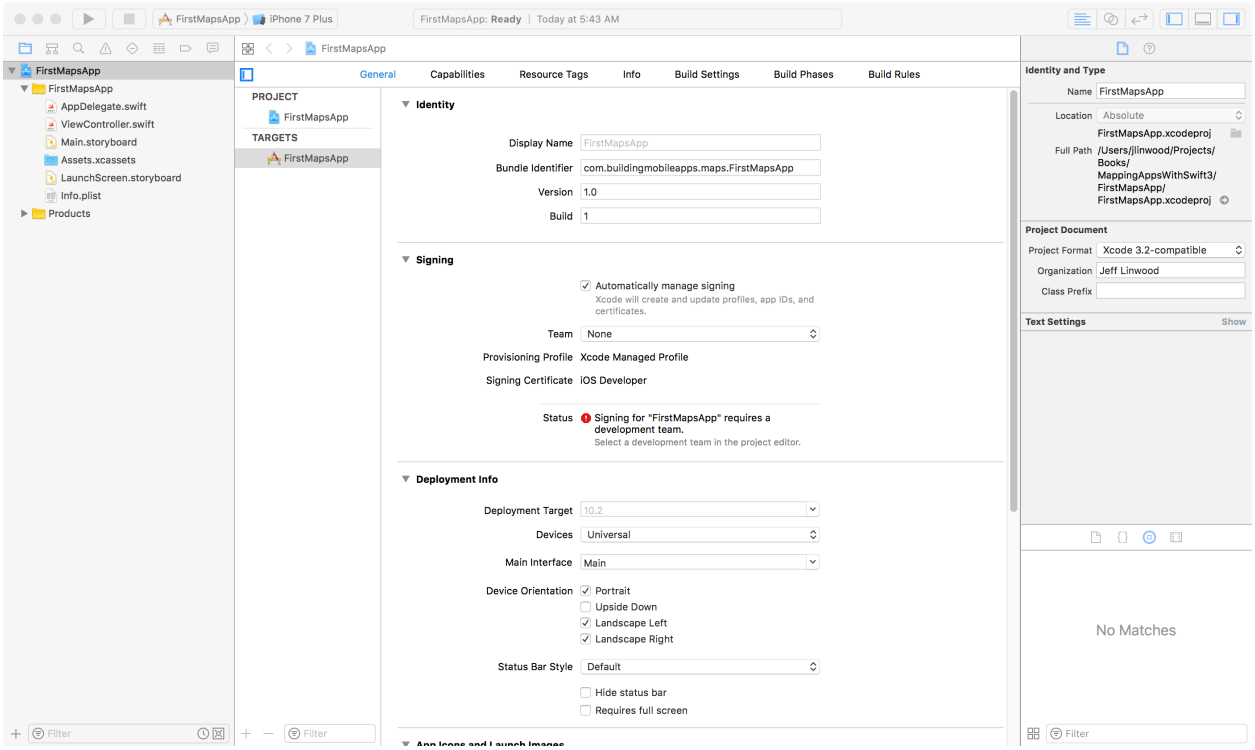
We do not need to include Core Data in our project - Core Data is an Apple technology used for storing data locally on iOS, and we won't need it for this example. We won't be using Core Data in this book.

You can also uncheck Include Unit Tests and Include UI Tests, as we won't be setting up any tests for this project.



New project options

Click Next, and save the project onto your hard drive. You can create a Git repository for your code if you want, but we won't be directly addressing source control in this book. It's always a good idea to keep up with Git commits as your project goes along, so that you can easily roll back to a working copy.



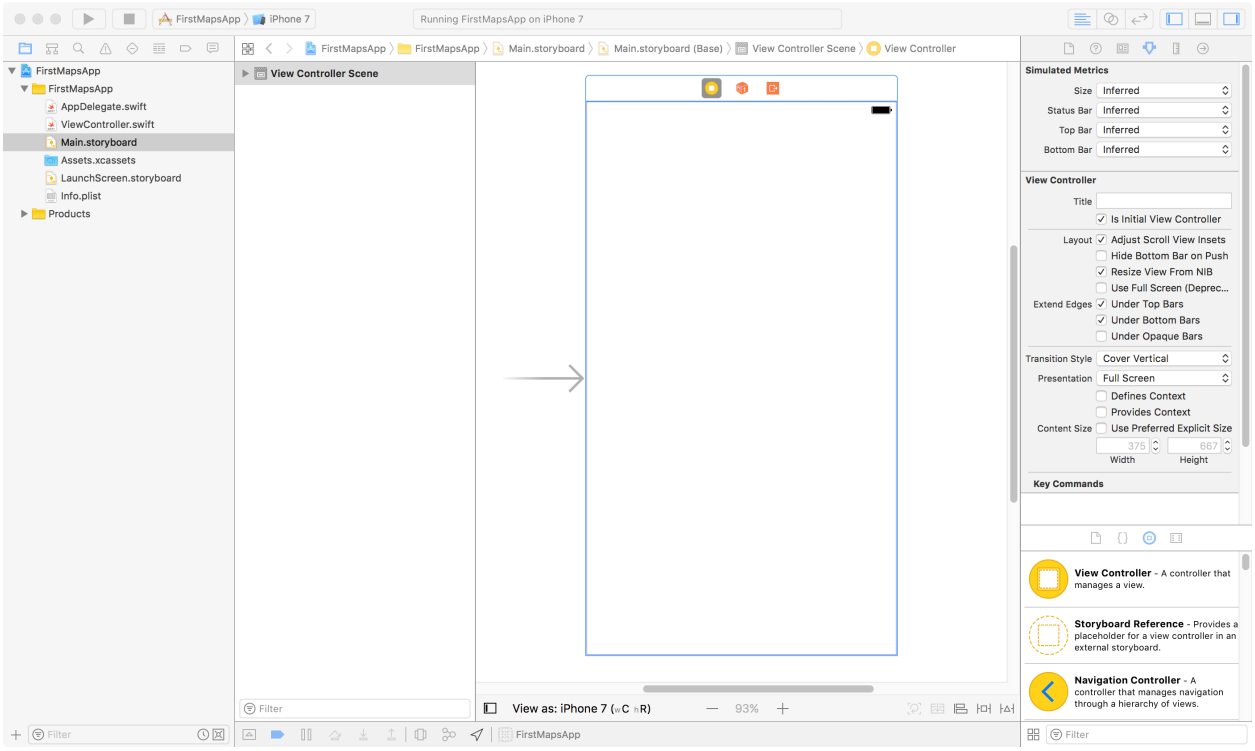
Project overview

You should now have a working XCode project - go ahead and run it in one of the iOS Simulators - for instance the iPhone 7:

Blank app

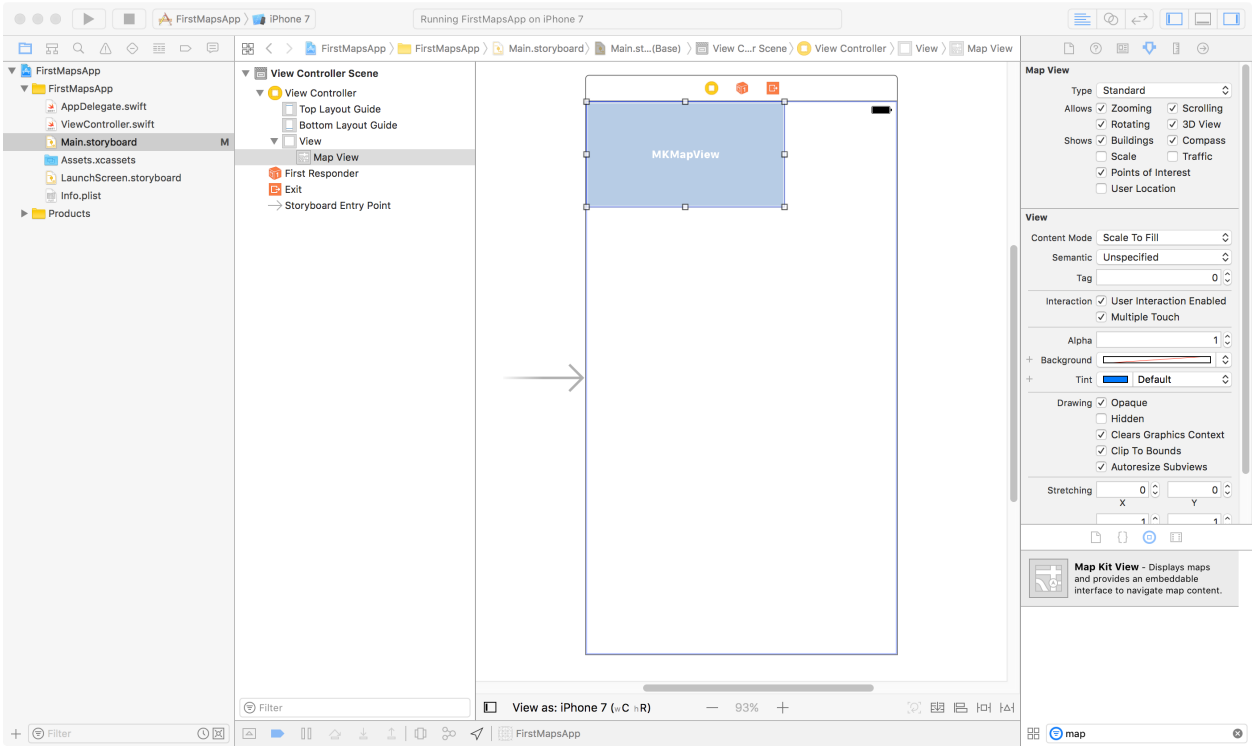
Adding a Map

Now it's time to add a map to our view controller. Select the storyboard (Main.storyboard), and then select the View Controller scene.



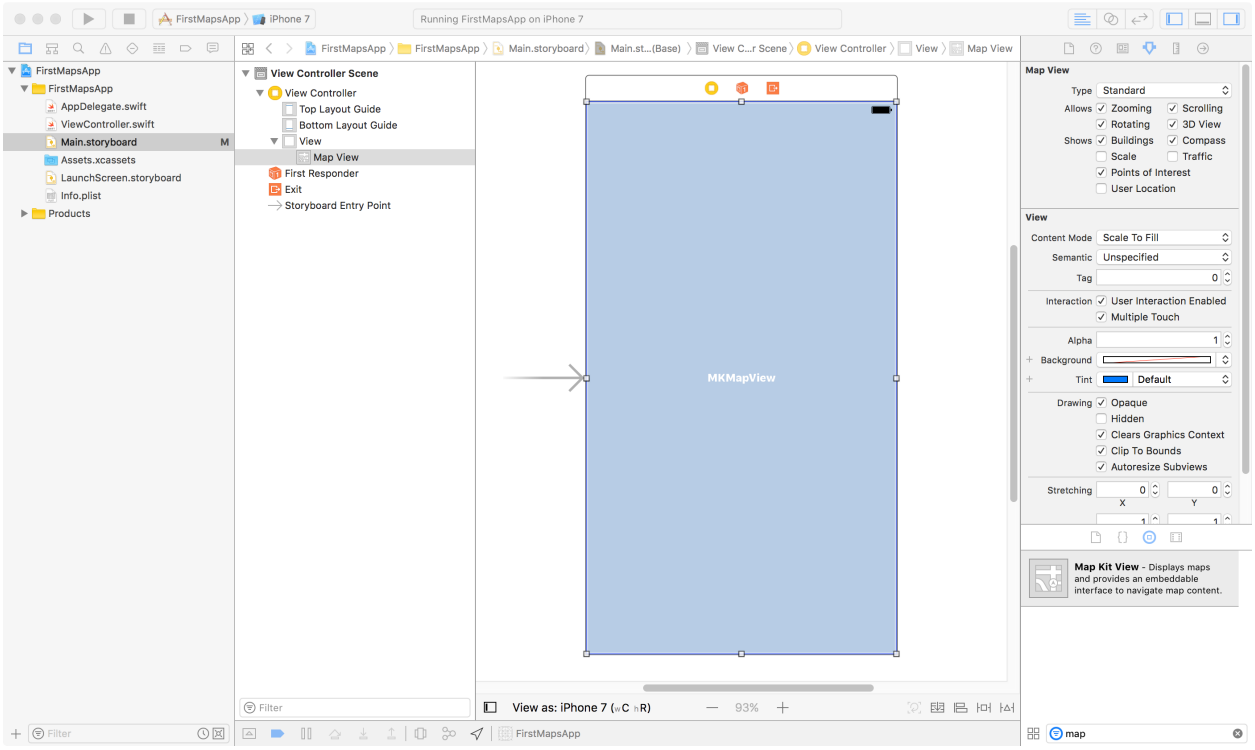
Empty storyboard

In the lower right hand corner of your XCode window, choose the third icon (the Object library), which is selected in the above screenshot. Either type Map into the search box underneath the list, or scroll down until you find the Map Kit View. Once you have found the Map Kit View, drag it onto your view controller.



Map View on storyboard

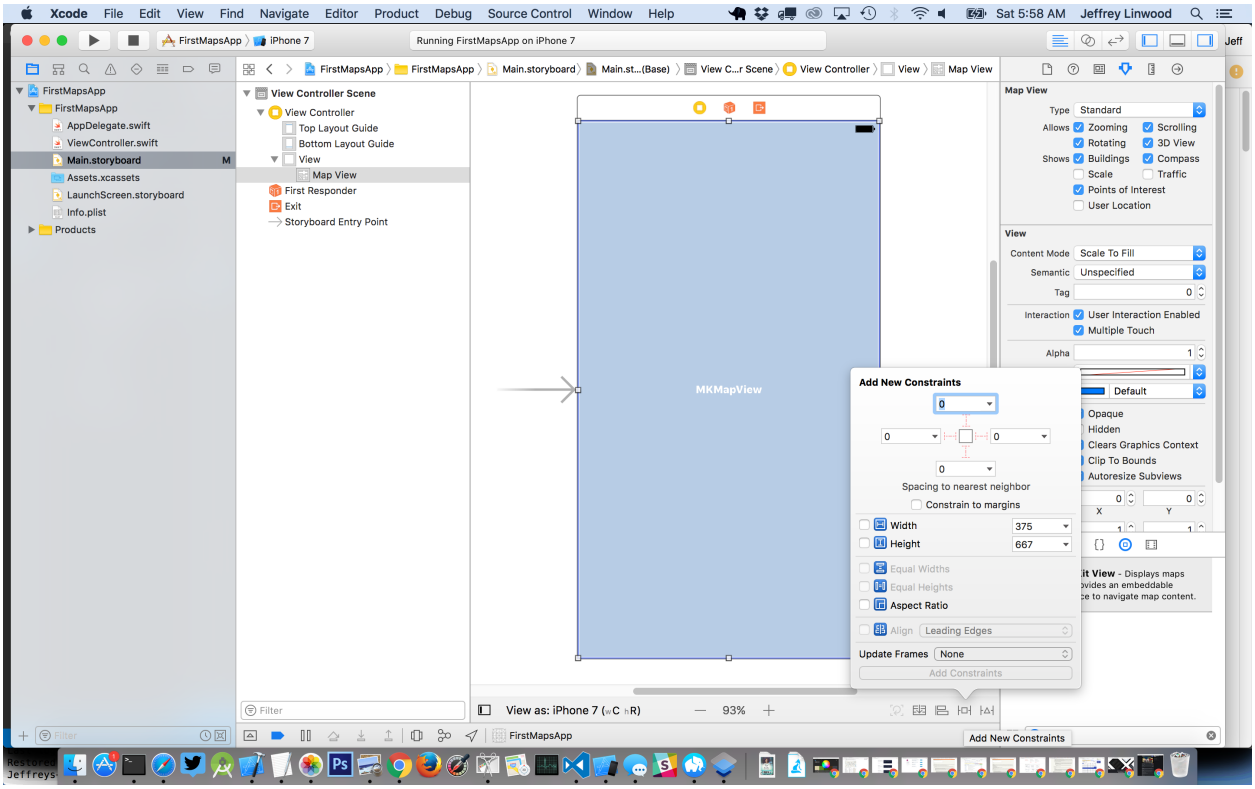
The map view won't automatically expand to fill the whole screen, so you will need to do that yourself by dragging the edges of the map view to fill the extent of the view.



Map View fills view

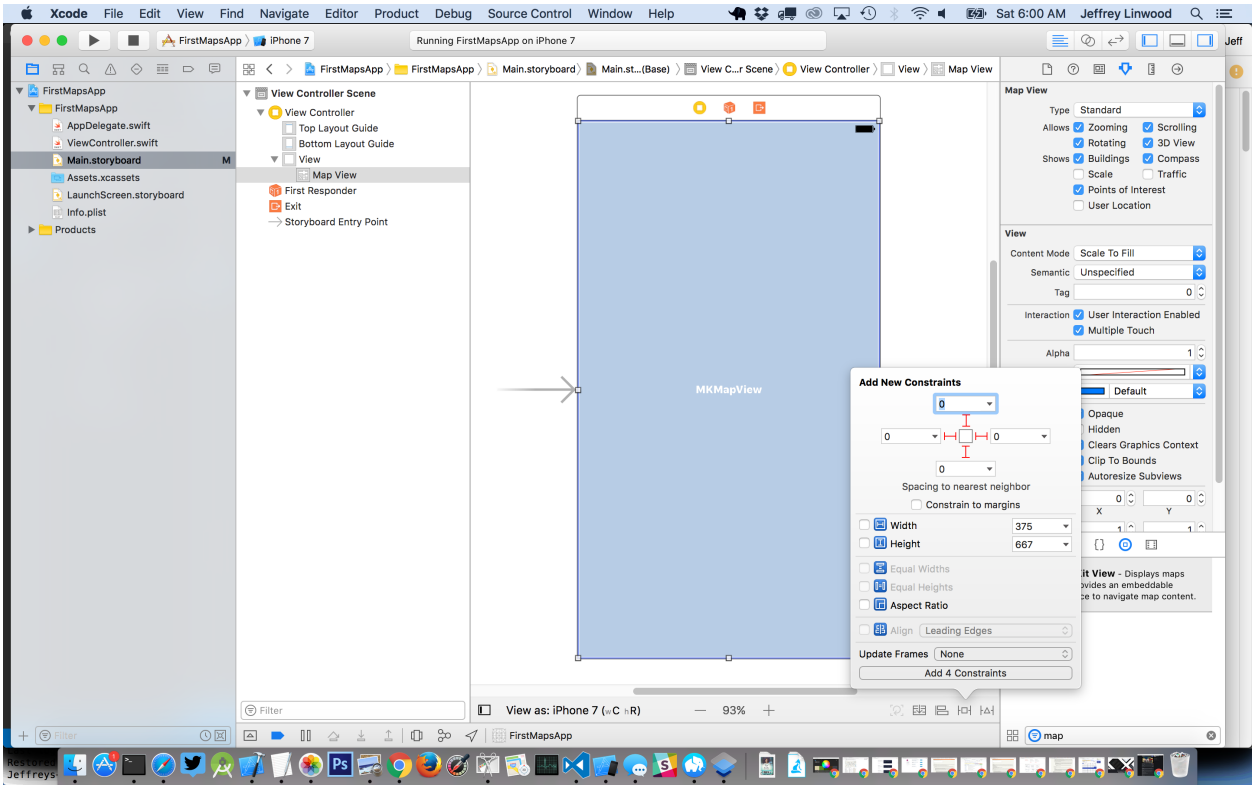
Even though we dragged the edges of the map view out to the edges of the view controller's view, that doesn't mean that the map view will use the entire screen on all sizes of the iPhone and iPad. To make the map view fill the view controller's view (also known as its parent view), we will need to add constraints to the map view.

On the right hand side of the toolbar underneath your view controller, you will see five icons. The fourth icon (Add New Constraints) opens up the Add New Constraints dialog box, which we can use for our layouts.



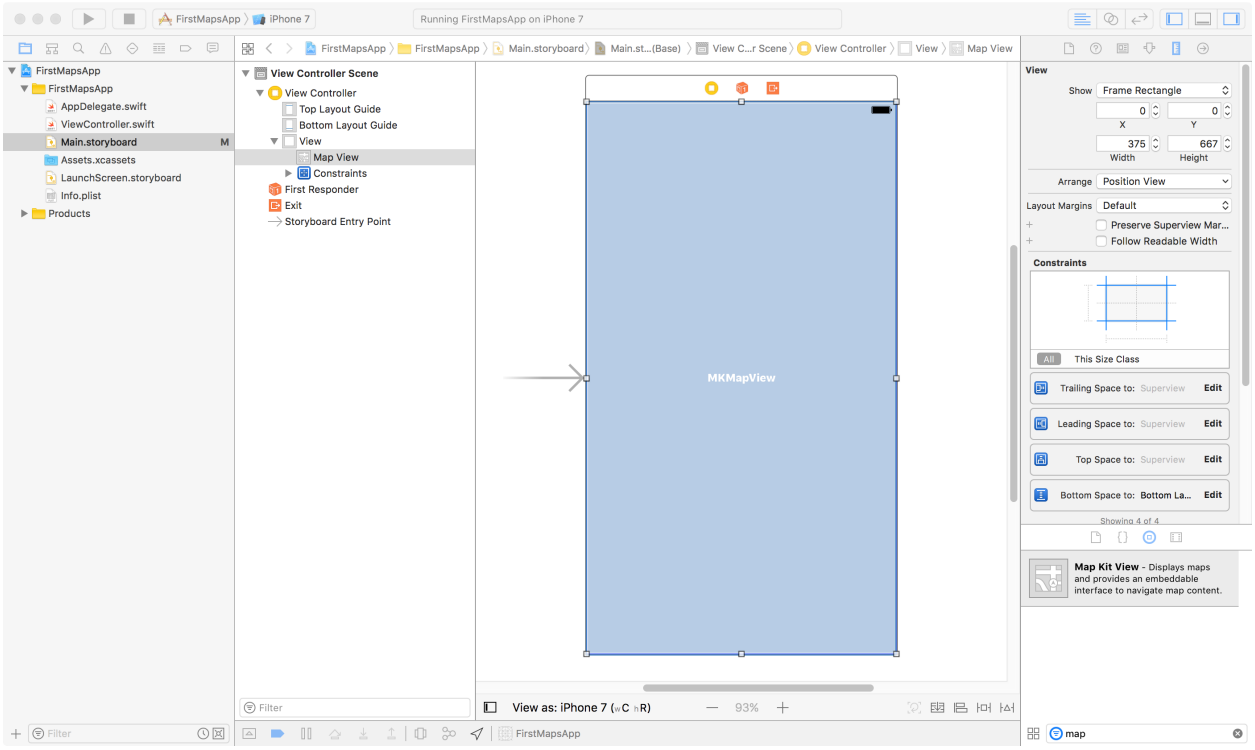
Adding constraints to Map View

Uncheck the “Constrain to margins” check box, as we are going to fill the entire view with the map, not leaving any margins. Go ahead and select the faint dashed red line for all four constraints (top, bottom, left and right). After selecting them, make sure that all of the values are 0, and press the “Add 4 Constraints” button.



Map View with constraints

Your map view will now properly fill up the entire screen on an iPhone or iPad. If you would like to double check this, select the Map View on the Storyboard. Next, choose the fifth icon on the right hand side, the Size Inspector, and you will see that you have constraints for all four sides of your Map View.

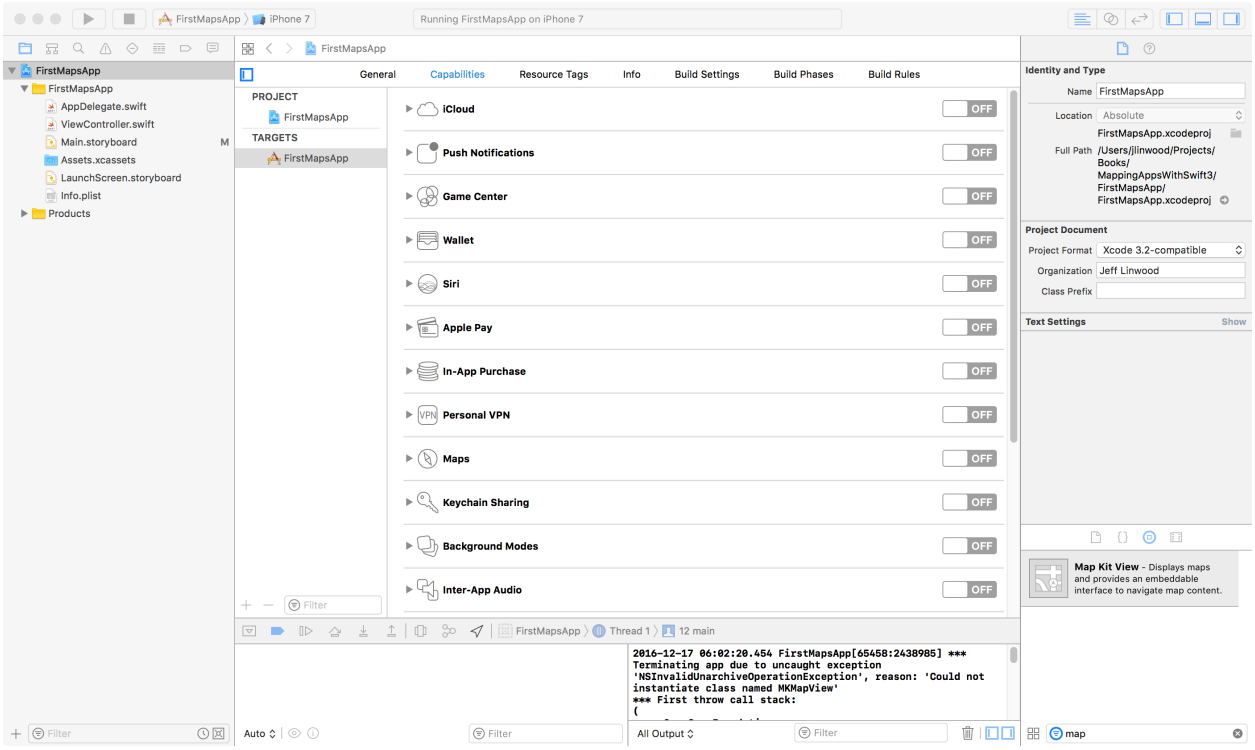


Size Inspector

You can go ahead and try to run your iOS app, but it will immediately crash, with a cryptic error:

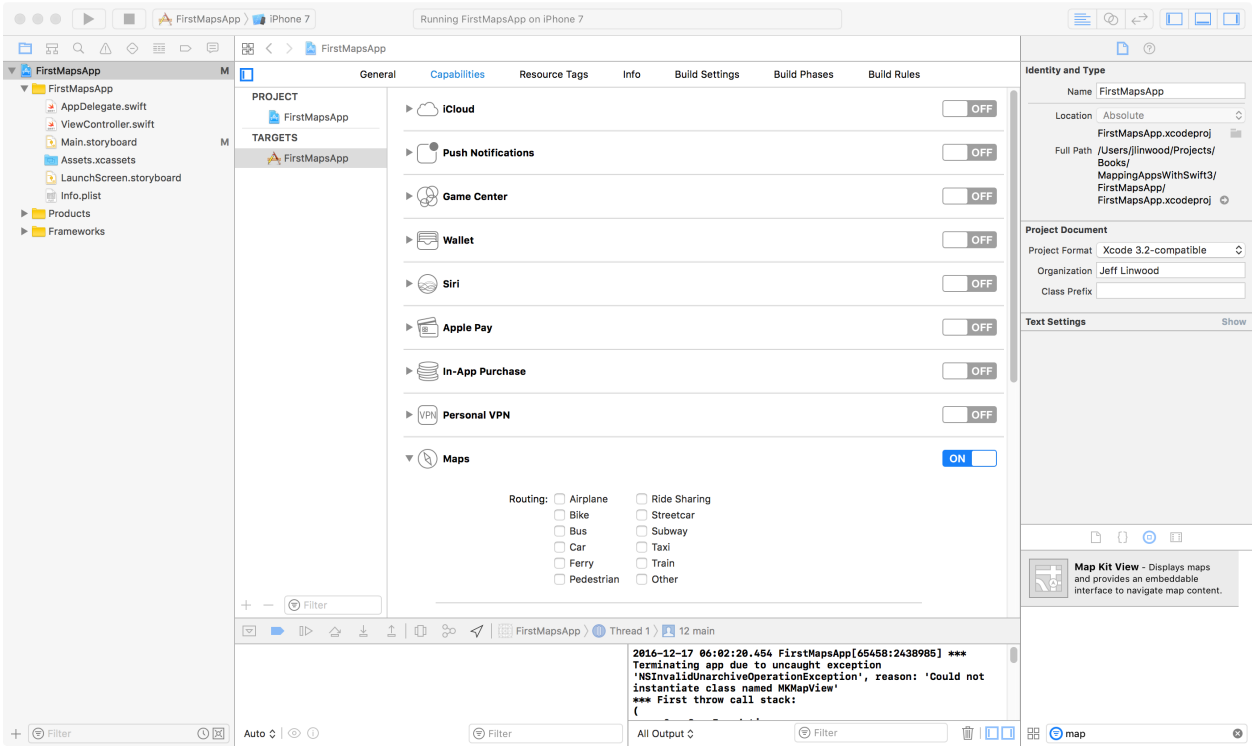
```
Terminating app due to uncaught exception 'NSInvalidUnarchiveOperationException'\
, reason: 'Could not instantiate class named MKMapView'
```

Your iOS app needs to include Apple's MapKit framework for the map view or any related functionality to work. In recent versions of XCode, this is now easily set up on the Capabilities tab of your project. Select your project in the file outline on the left hand size of XCode, and it will show the General tab with information about your project. The second tab is Capabilities, which you can select to get a list of different enhancements and features you can implement within your iOS app, such as background modes or push notifications. By default all of these capabilities are off. We will need to turn the Maps capability on.



Capabilities Screen

Just turn the Maps switch to on, and Xcode will link the MapKit.framework with your app.



Maps enabled

Now try running your iOS app, and you will see that you have a nice, large map on your app!

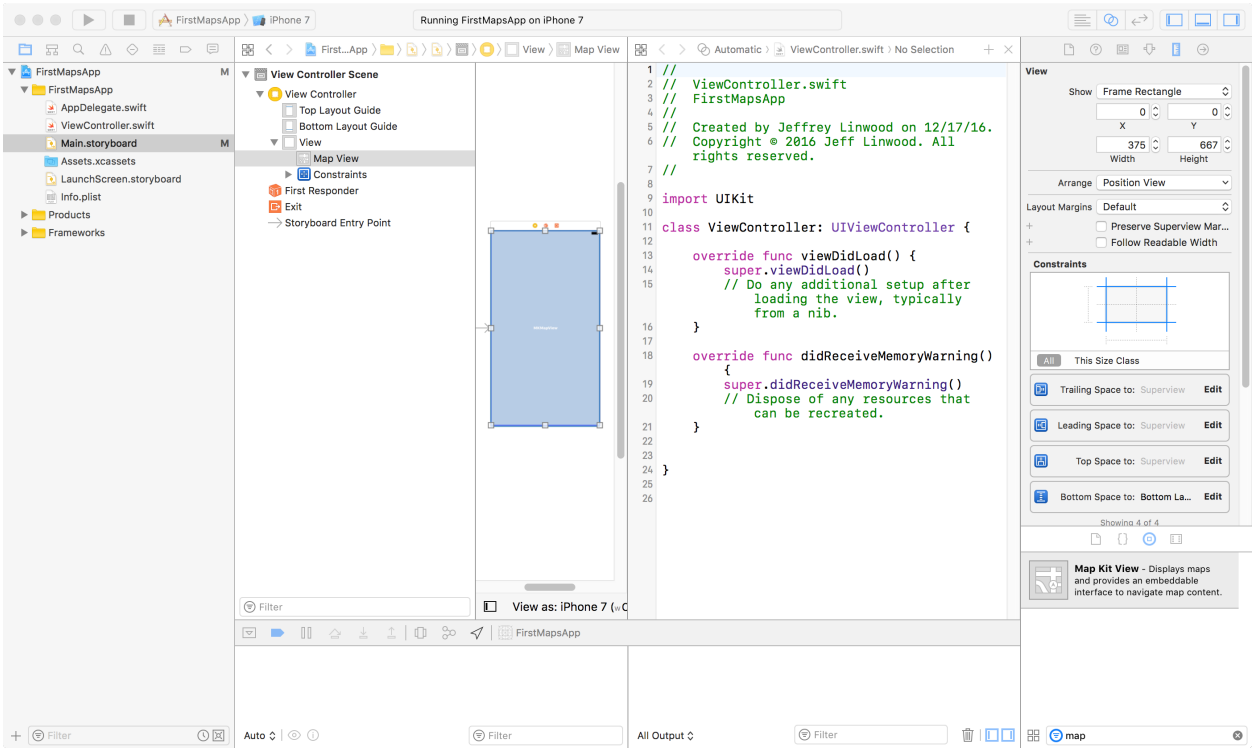

iPhone 7 – iOS 10.2 (14C89)
Carrier
4:25 PM
CANADA
Montréal
Toronto
Chicago
UNITED STATES
New York
Washington
MEXICO
Mexico City
CUBA
DOMINICAN REPUBLIC
HONDURAS
NICARAGUA
PANAMA
COLOMBIA
ECUADOR
PERU
Lima
Legal

App with Map

Adding a Pin to your Map

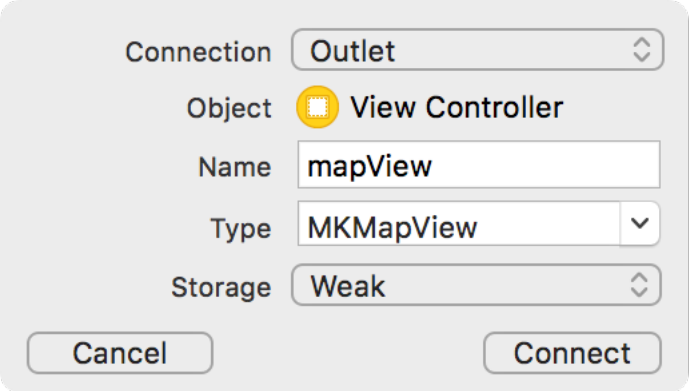
Now that we have our map, it's time to add a pin that shows our home city!

Before we add the pin to the map, we will need to create an outlet for the map, named mapView, using XCode's Assistant. Open the Assistant View (the interlocking circles on the right hand side of XCode), and you'll see the ViewController class open up next to your view controller in the storyboard.



XCode Assistant

Select the map view on the storyboard or on the outline view, hold down the Control key, and then drag an outlet into the ViewController class. Name it mapView.



MapView Outlet

You'll notice that the ViewController class will no longer compile - that is because our map is an MKMapView, part of the MapKit framework that we earlier linked with our project on the Capabilities tab. We need to import this framework into our ViewController class so that we can use classes from the MapKit framework.

Add this line right below the import UIKit statement

```
1  import MapKit
```

Your ViewController.swift file should look similar to this:

```
1  //
2  //  ViewController.swift
3  //  FirstMapsApp
4  //
5  //  Created by Jeffrey Linwood on 12/17/16.
6  //  Copyright © 2016 Jeff Linwood. All rights reserved.
7  //
8
9  import UIKit
10 import MapKit
11
12 class ViewController: UIViewController {
13
14     @IBOutlet weak var mapView: MKMapView!
15
16     override func viewDidLoad() {
17         super.viewDidLoad()
18         // Do any additional setup after loading the view, typically from a nib.
19     }
```

```
20
21      override func didReceiveMemoryWarning() {
22          super.didReceiveMemoryWarning()
23          // Dispose of any resources that can be recreated.
24      }
25
26
27  }
```

On iOS, we represent locations on the map as annotations. Annotations implement the MKAnnotation protocol, which consists of a latitude and longitude coordinate pair, and an optional title and subtitle. MapKit comes with a basic implementation of MKAnnotation, the MKPointAnnotation class, but for most apps, you will want to create your own implementation of MKAnnotation. In this chapter we will use MKPointAnnotation, but the later chapters of this book will use our own implementation, so you can see how it works both ways.

Once you have an annotation (or many annotations), you can just add it to the map using the addAnnotation() or addAnnotations() methods on the MKMapView class.

Annotations are not the actual pin that the map displays - those are annotation views, which are subclasses of the MKAnnotationView class. By default, you will get an MKPinAnnotationView, which is the standard red pin that you see in many mapping apps. You can customize the pin color a little, but for most apps, you will want to put in your custom images. We'll use our own custom images in the next chapters of this book.

To create an annotation as an MKPointAnnotation, we do need to be able to create a coordinate, which we can do with CLLocationCoordinate2DMake(). For our purposes in this chapter, we are going to add all of the code to the viewDidLoad() function in the ViewController class.

Pass in the latitude and the longitude (as double values) to create the coordinate. The MKPointAnnotation will need this coordinate set, such as in the following code:

```
1  let austin = MKPointAnnotation()
2  austin.coordinate = CLLocationCoordinate2DMake(30.25, -97.75)
```

The longitude for Austin is going to be negative, because Austin, Texas is in the Western Hemisphere. The latitude is positive because the city is in the Northern Hemisphere.

To give the annotation a title, we can simply set the title property

```
1  austin.title = "Austin"
```

And then after setting the title and coordinate properties, we can call one method on the map view to add the annotation:

```
mapView.addAnnotation(austin)
```

Run this class (the complete version is below), and you will see the iPhone app displaying the Austin pin. Go ahead and change the pin to your city, or to any other location you want. Add more pins for different locations!

```
//
//  ViewController.swift
//  FirstMapsApp
//
//  Created by Jeffrey Linwood on 12/17/16.
//  Copyright © 2016 Jeff Linwood. All rights reserved.
//

import UIKit

import MapKit

class ViewController: UIViewController {

    @IBOutlet weak var mapView: MKMapView!

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view, typically from a nib.
        let austin = MKPointAnnotation()
        austin.coordinate = CLLocationCoordinate2DMake(30.25, -97.75)
        austin.title = "Austin"
        mapView.addAnnotation(austin)
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be recreated.
    }
}
```

The iPhone app showing the Austin pin:

Our finished application