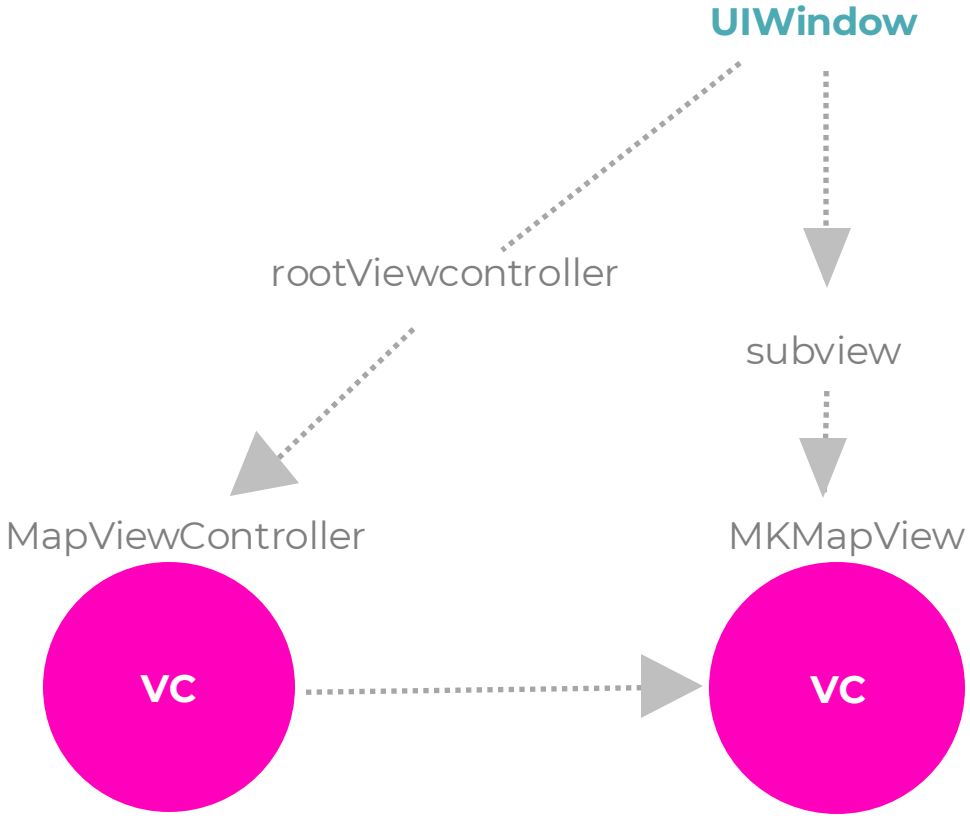
View Controllers

MK Map View

View Controller

Manages view hierarchy
Creates view objects
Handles events associated with view objects

View Controllers

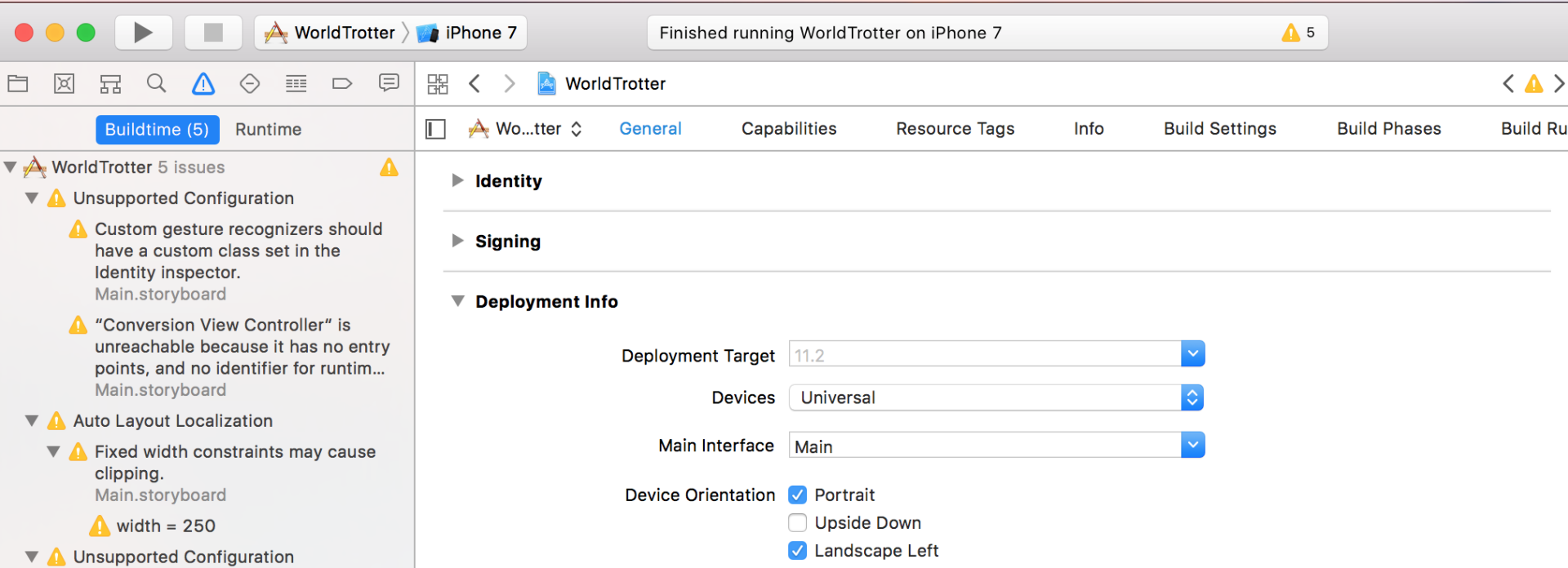


2 ways to create ViewController:
1. Using storyboard in IB
2. Programmatically overriding method loadView()

Each storyboard has only one Initial view controller.
It is an entry point to the storyboard.
This view controller shows first.

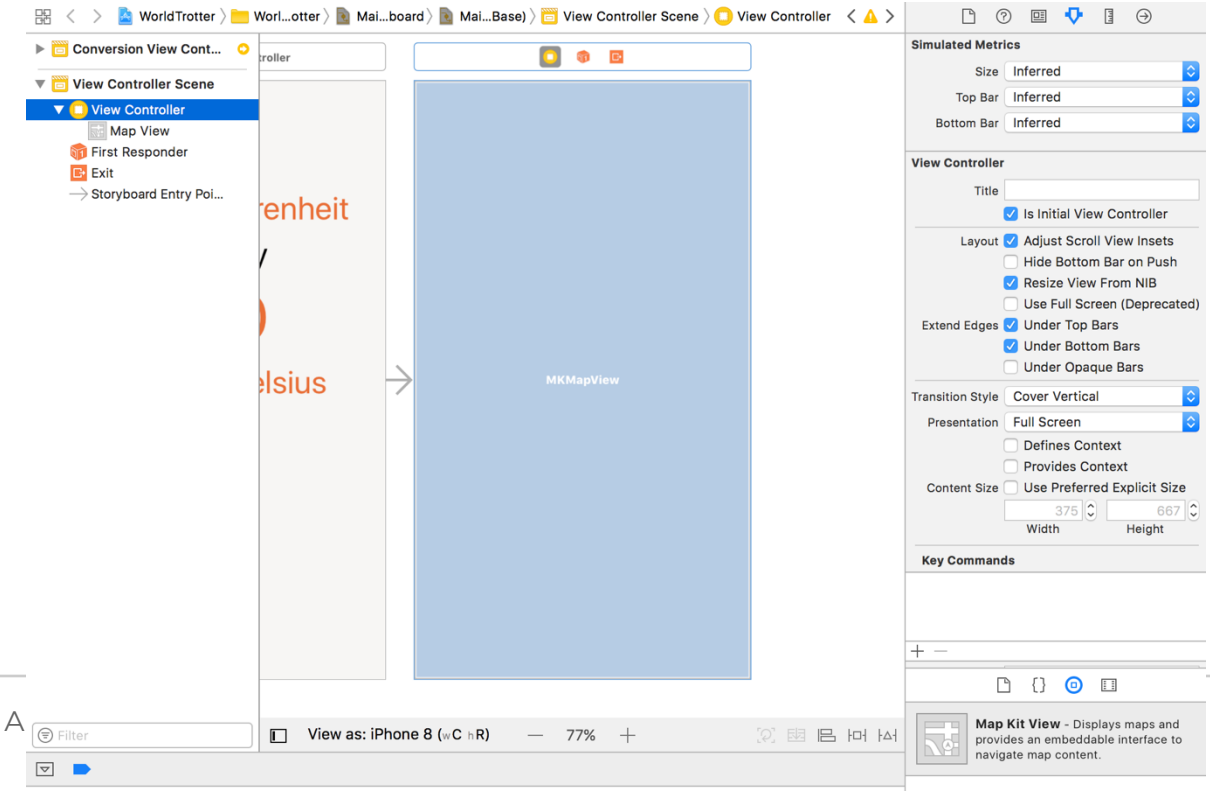
Add the MapKit framework

Each app has one main interface (main.storyboard)
When the app launches, the initial view controller gets set as the rootViewController of the window.
The main interface is set in the project settings >General > Deployment >Main Interface
Main corresponds to Main. Storyboard.



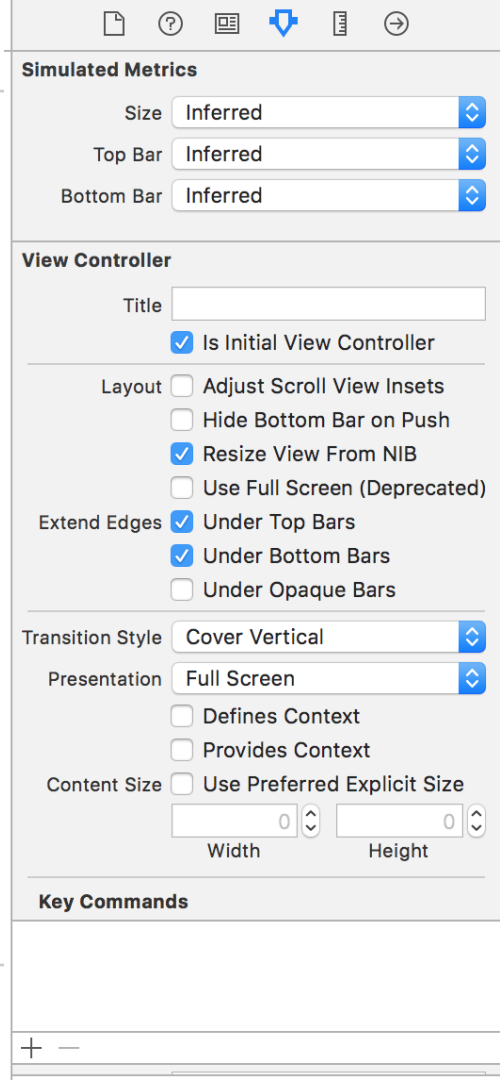
Setting Initial View Controller

Open Main.storyboard.
Drag View Controller onto canvas. Select the View of the ViewController – Delete.
Drag Map Kit View from object library (search) onto controller to set map view.



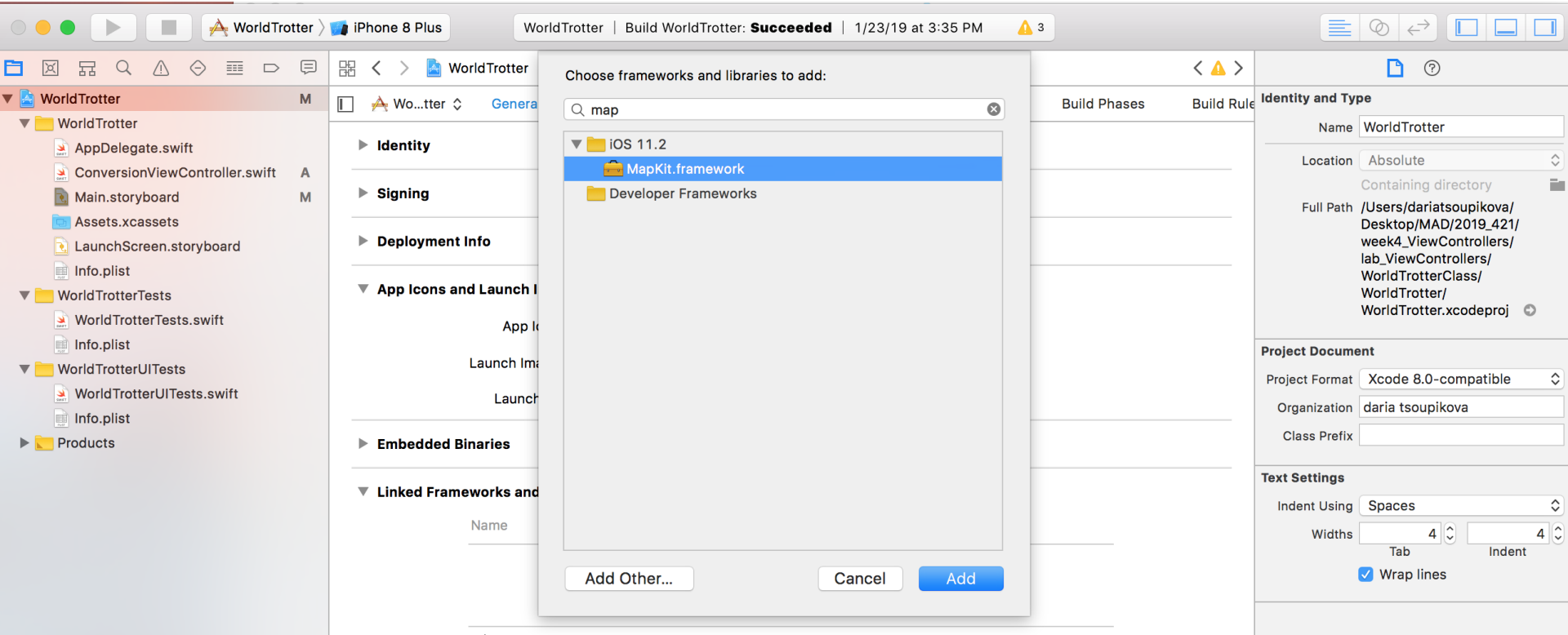
Setting Initial View Controller

Select View Controller and open Attributes inspector.
Check Is Initial View Controller
Notice the change of direction of the gray arrow
Now it points to View Controller now, not Conversion View Controller.
Test and run the app.



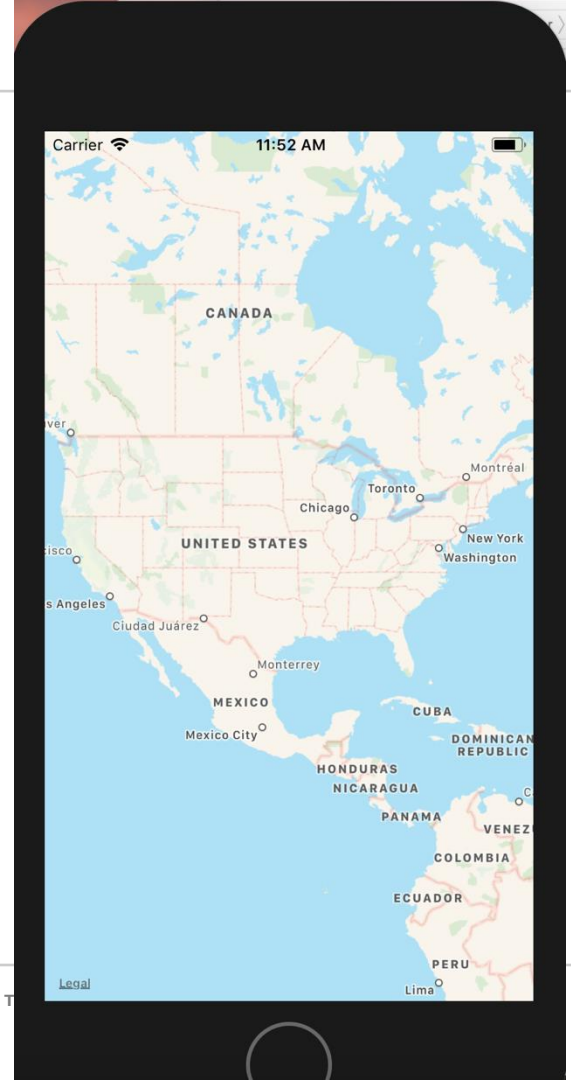
Add the MapKit framework

Import MapKit framework and link it to the view to load map view.
WorldTrotter project > Project Settings > General > Linked Frameworks and Libraries
Check + sign and search for MapKit framework. Click Add. Build and run to test map view.



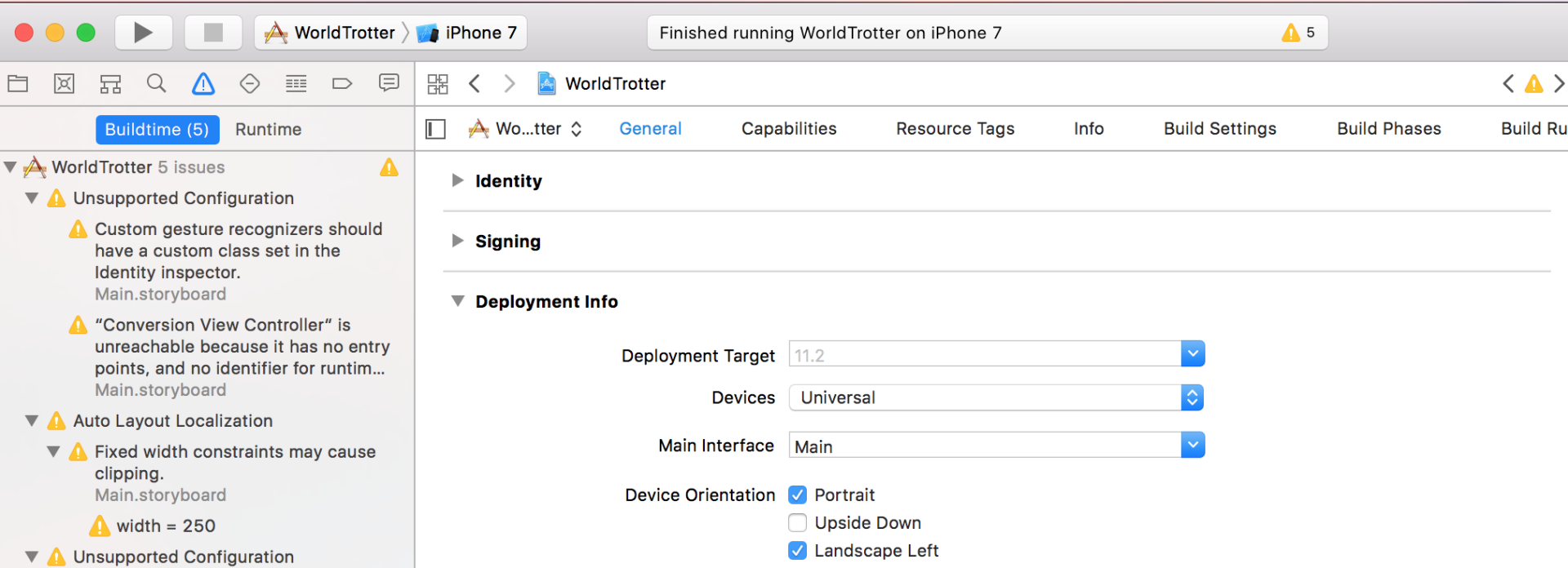
Add the MapKit framework

Build and run to test map view.



Add the MapKit framework

Each app has one main interface (main.storyboard)
When the app launches, the initial view controller gets set as the rootViewController of the window.
The main interface is set in the project settings >General > Deployment >Main Interface
Main corresponds to Main. Storyboard.

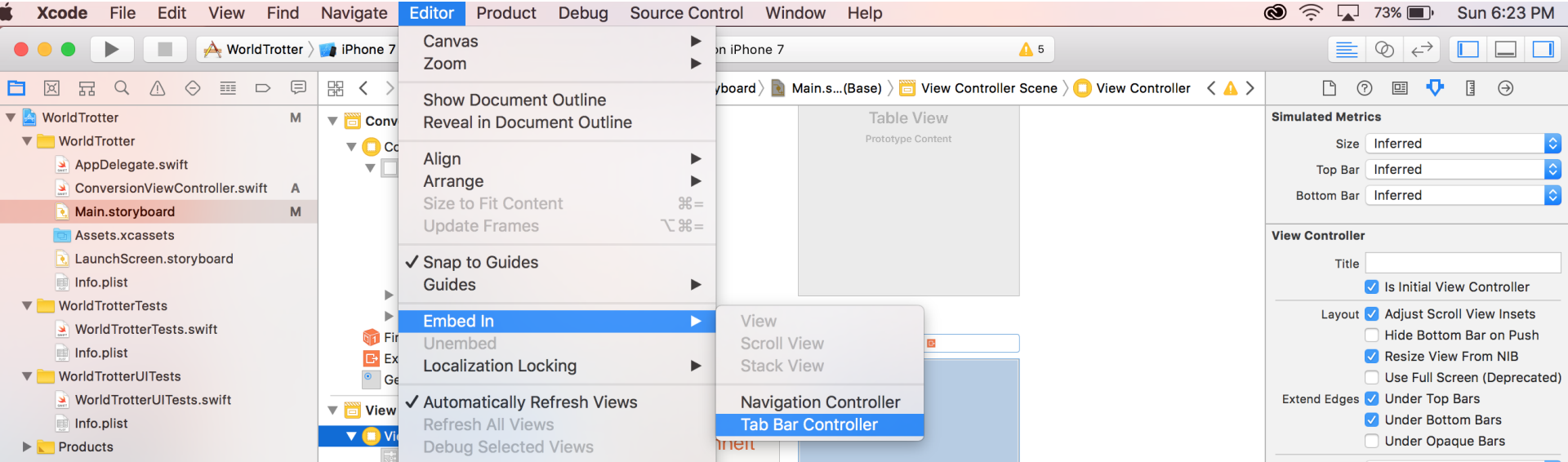


Add Transition

UITabBarController allows swap between the View Controllers.
Keeps array of view controllers
Maintains a menu to select view controllers

Open Main.storyboard> select View Controller

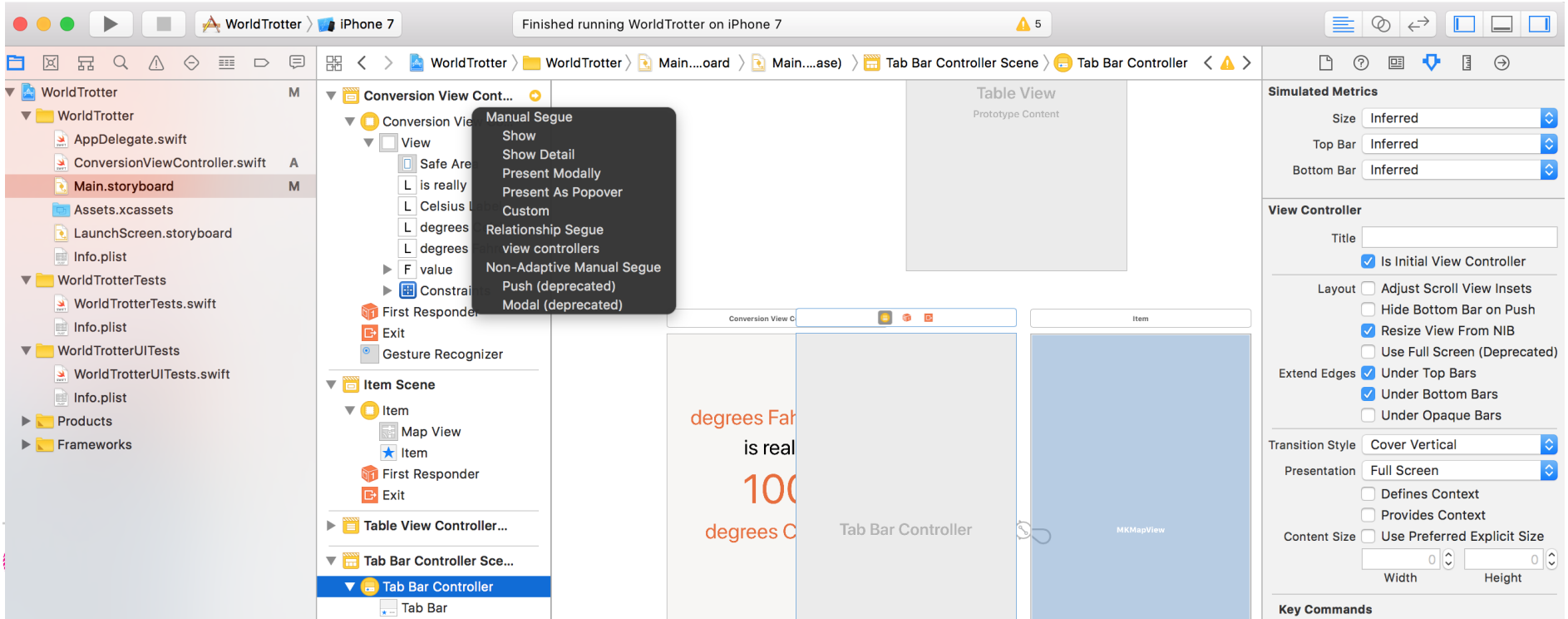
Editor > Embed In > Tab Bar Controller . Relationship arrow will be added to pointing from TabBar Controller to View Controller



Add Transition

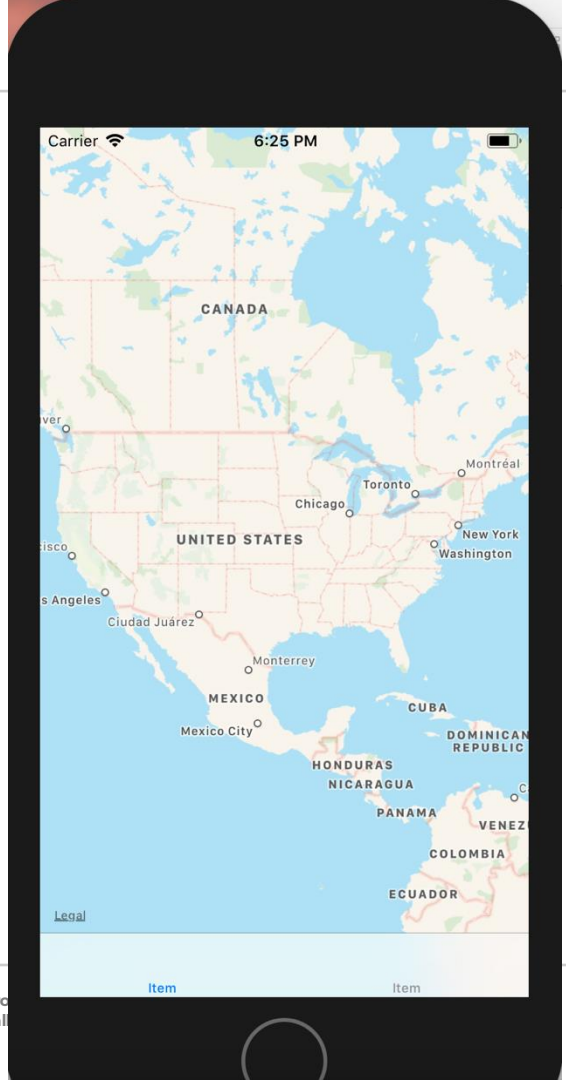
Add Conversion ViewController to TabBarController to switch between the screens:

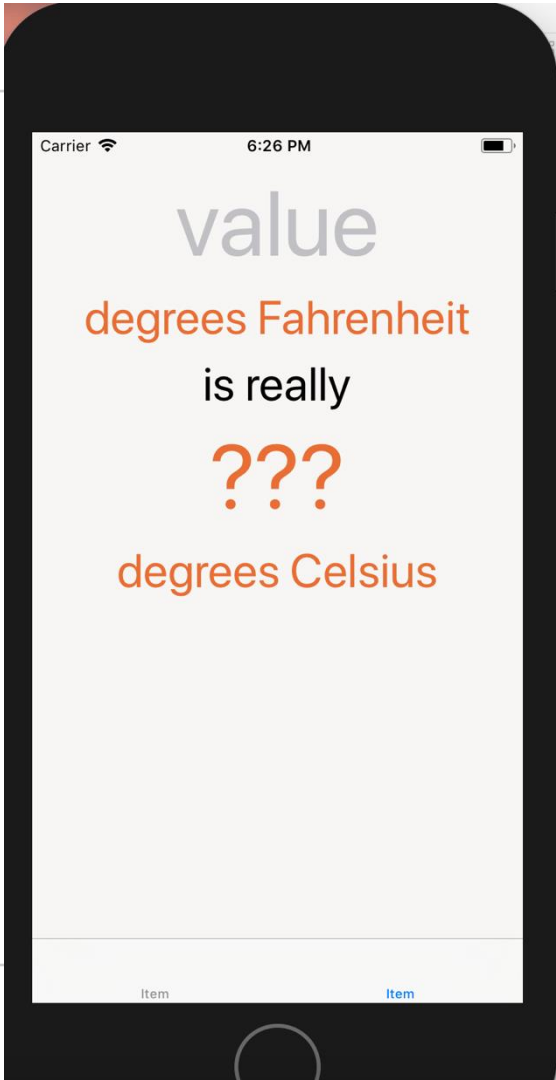
Control-drag from the TabBarController to the ConversionViewController
Choose: Relationship Segue > view controllers. Build, run and test switching views.



Add Transition

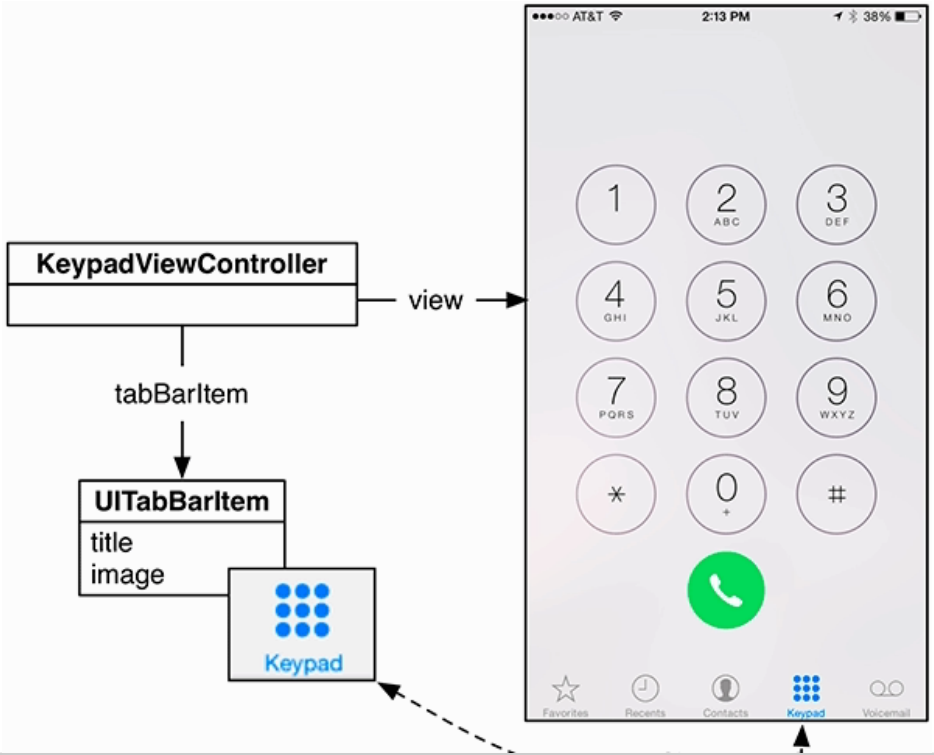
Tab menu bars to switch between the screens:





Modify the menu

Each tab bar in the menu can display a title and an image. (currently untitled “Item”)



Modify the menu

Add image by adding it to assets
Assets are files to be selected at runtime

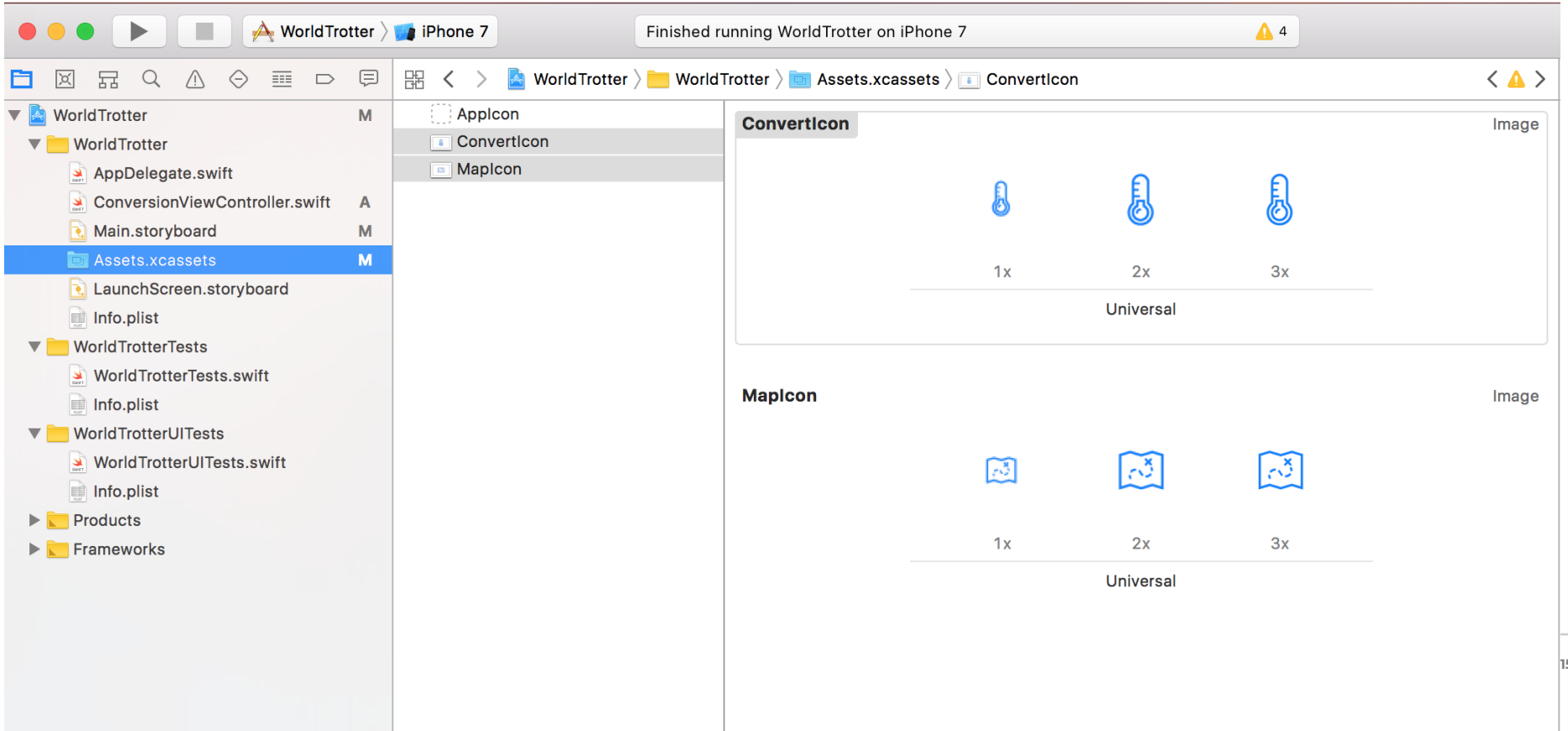
Open Assets.xcassets

download from images folder:

Under Ch. 5 Xcode Assets
ConvertIcon.png
ConvertIcon@2x.png
ConvertIcon@3x.png
MapIcon.png
MapIcon@2x.png
MapIcon@3x.png

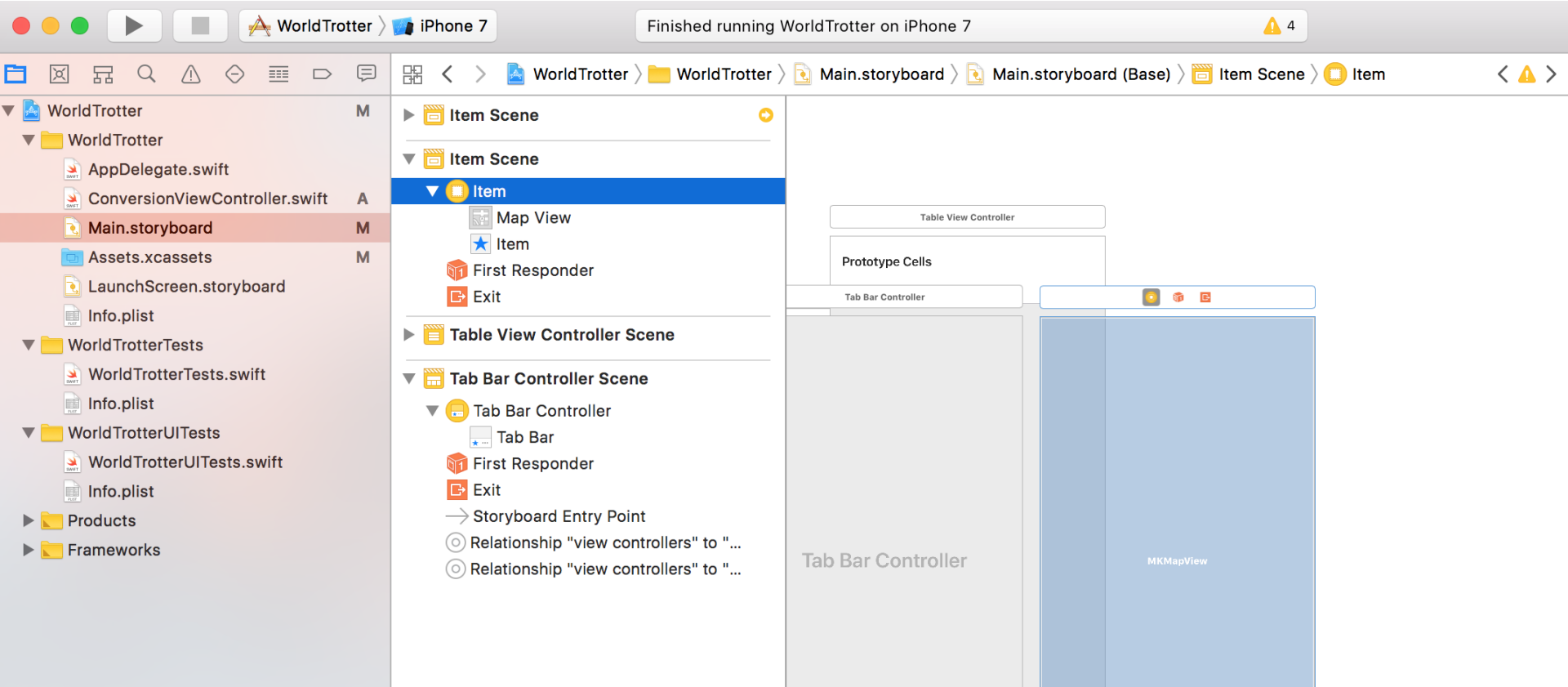
Modify the menu

Drag these images to images set list of the Assets Catalog.



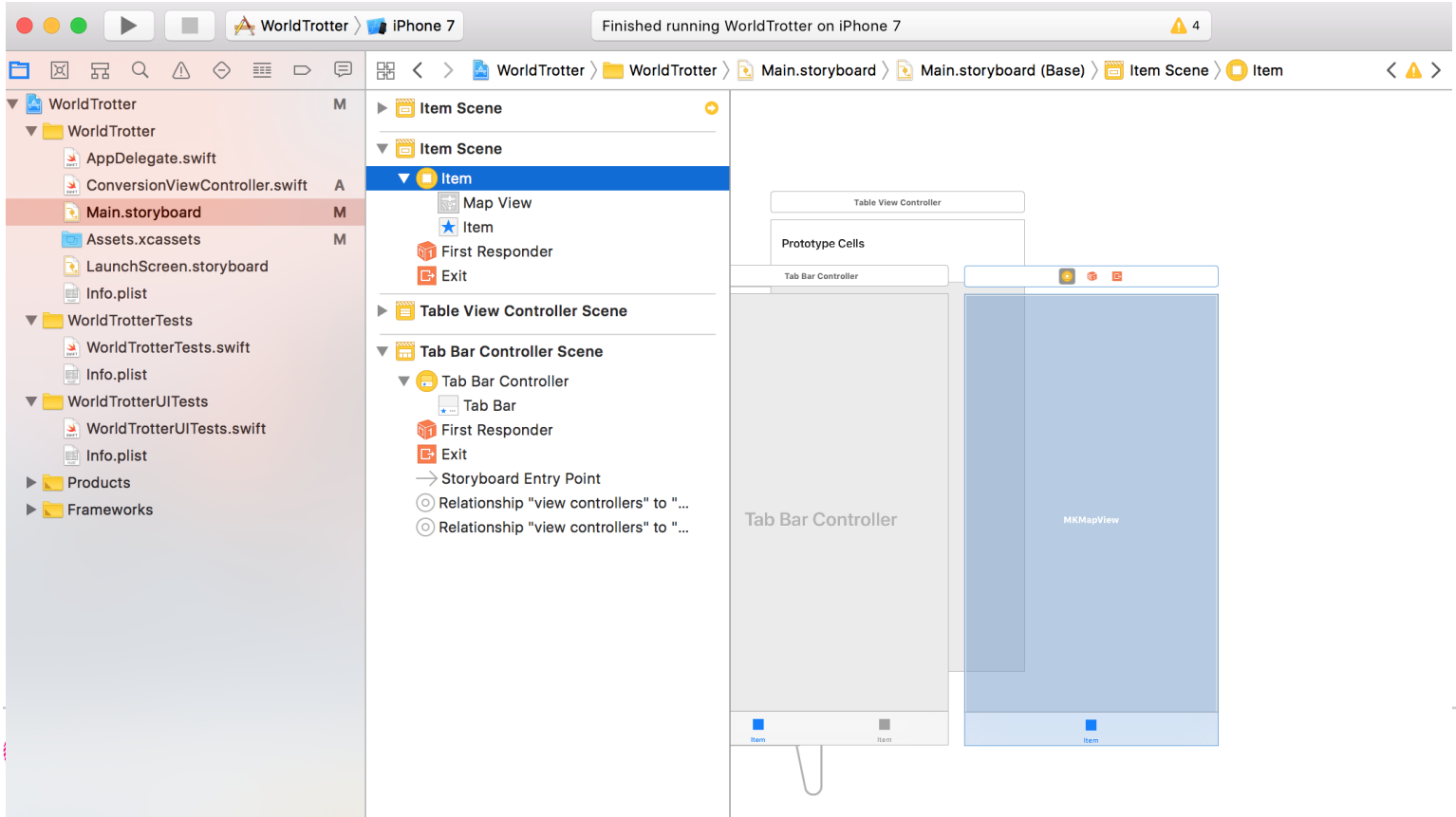
Modify the menu

In Main.storyboard select ViewController – it is named Item now



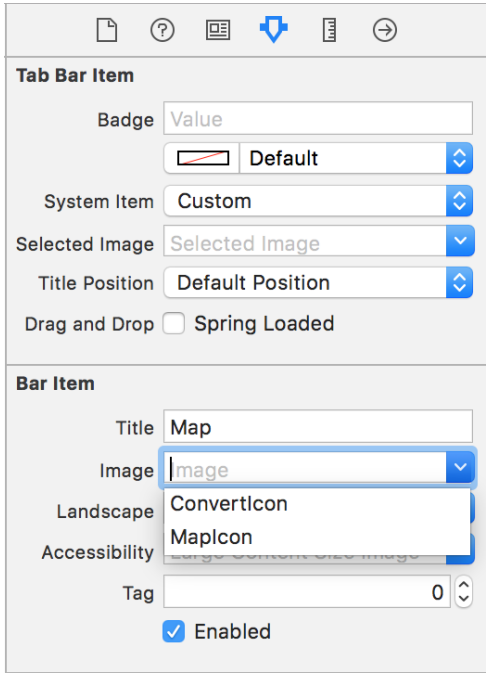
Modify the menu

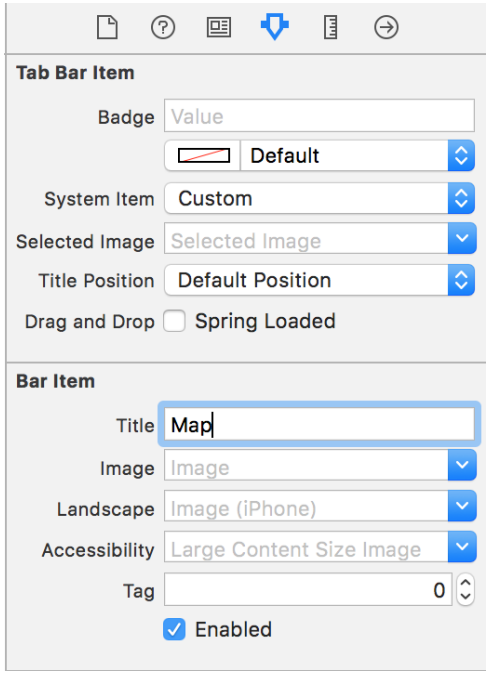
Notice the tab bar is added to the bottom of the Interface



Modify the menu

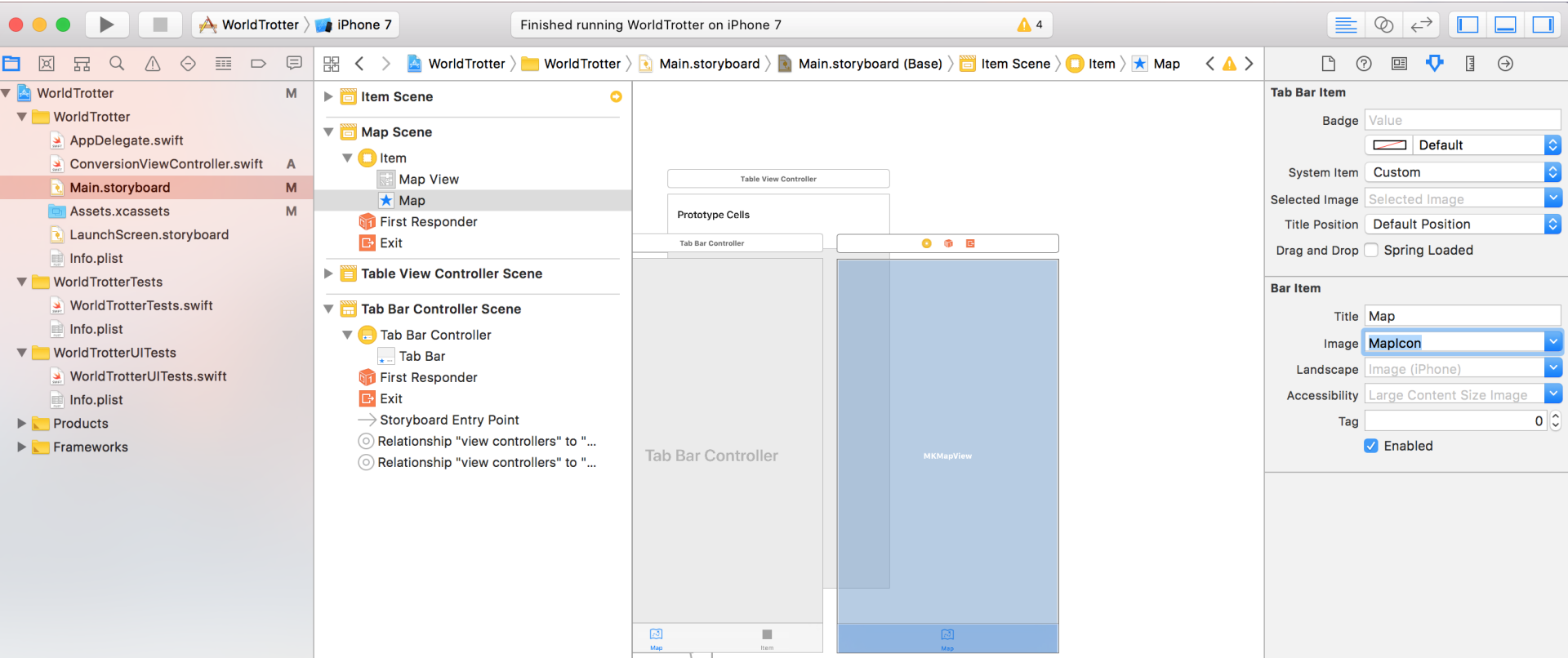
Select this tab bar and open attributes
Change name to Map
Choose MapIcon as Image





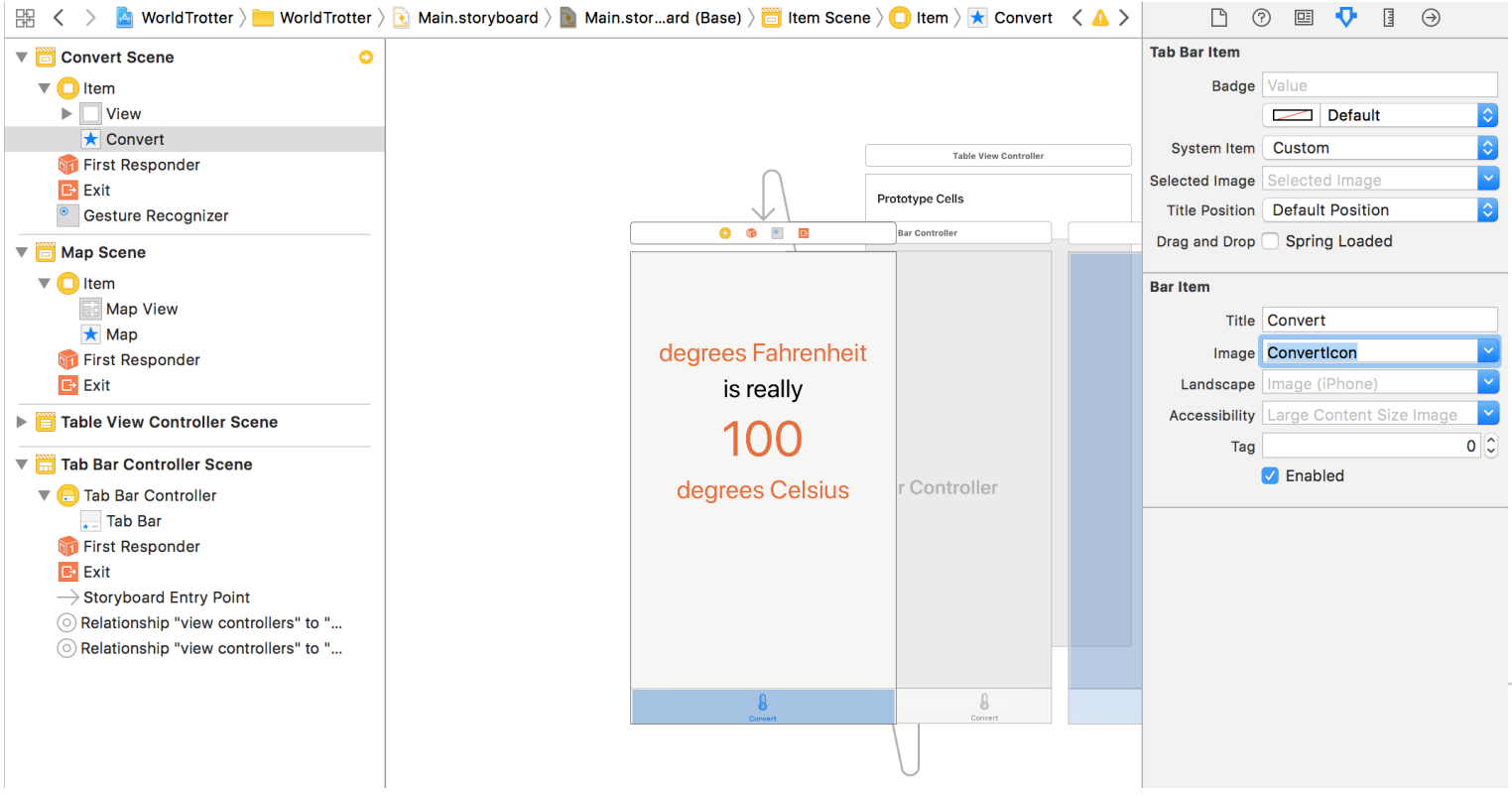
Modify the menu

Select this tab bar and open attributes
Change name to Map
Choose MapIcon as Image



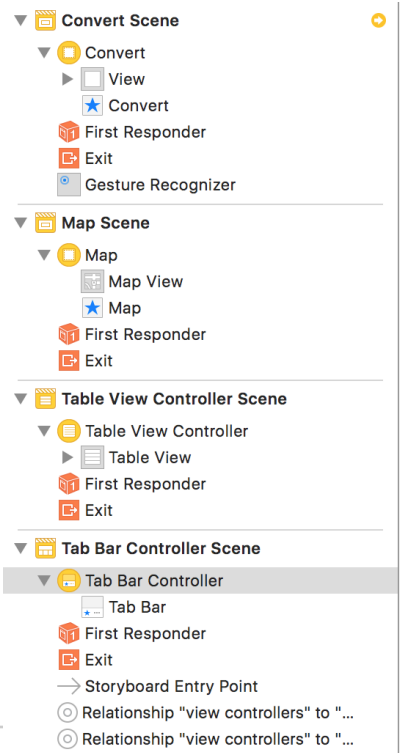
Modify the menu

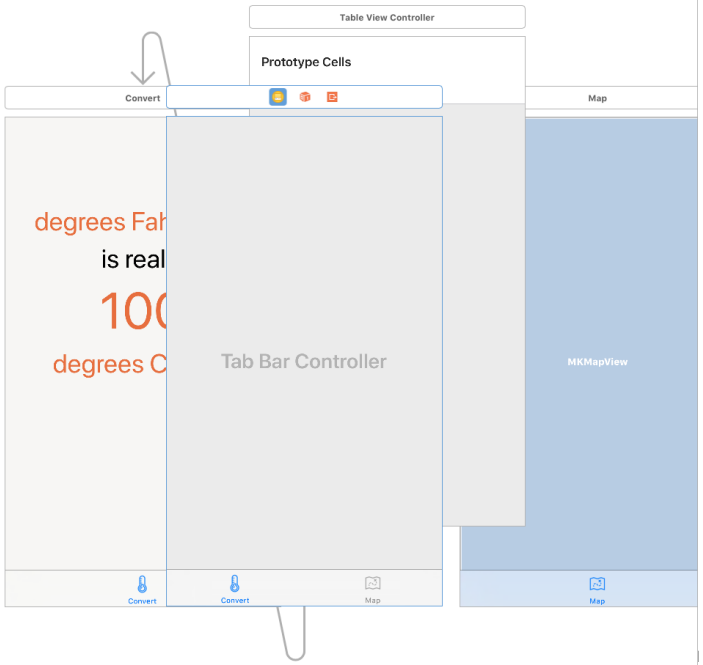
Select tab bar in ConversionViewController
Change name to “Convert”
Choose ConvertIcon as Image



Modify the menu

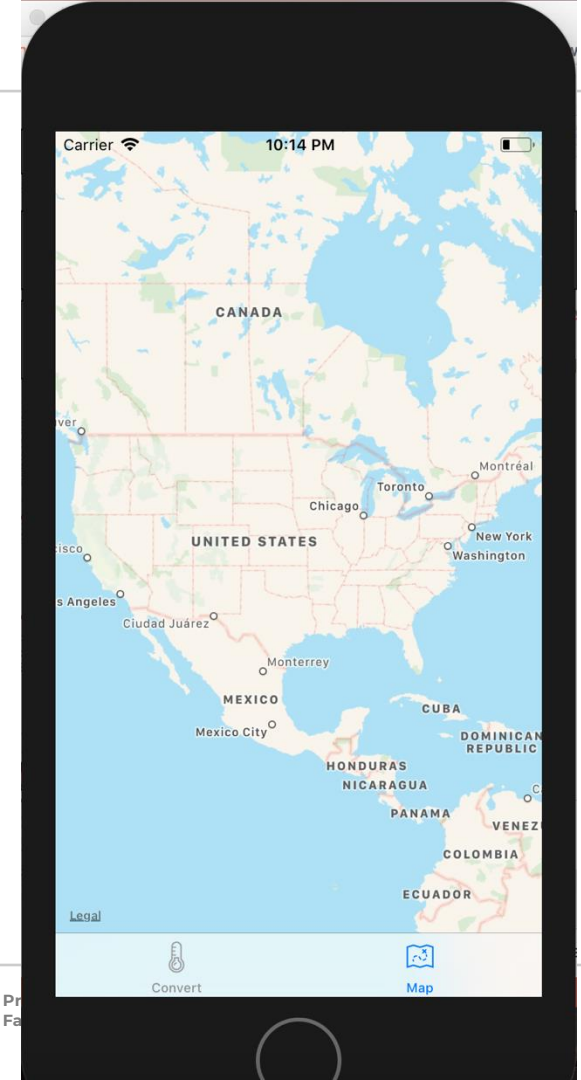
Select Tab Bar Controller on canvas (or in storyboard)
Drag the Convert tab to be on the left

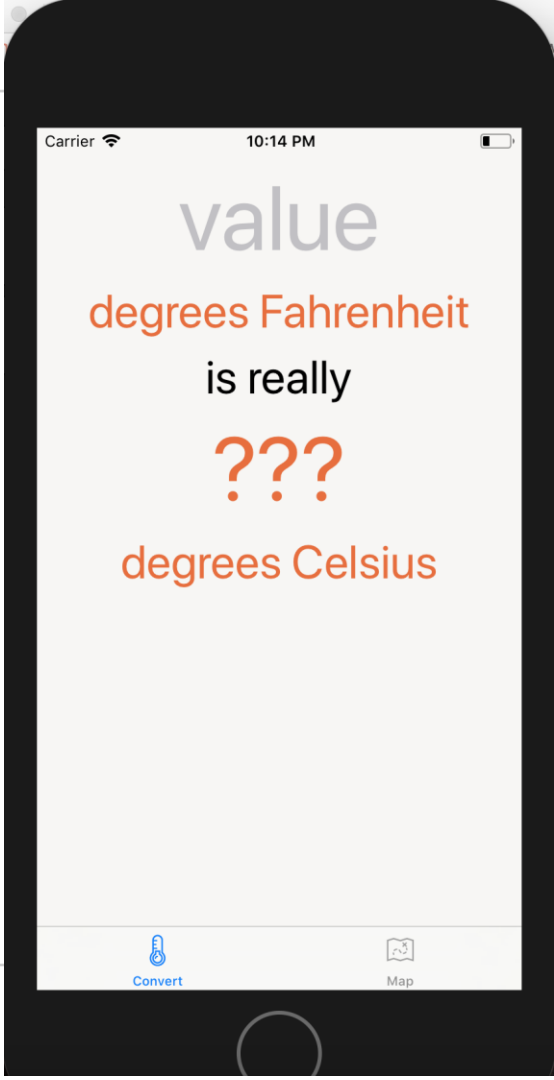




Modify the menu

Build, run and test

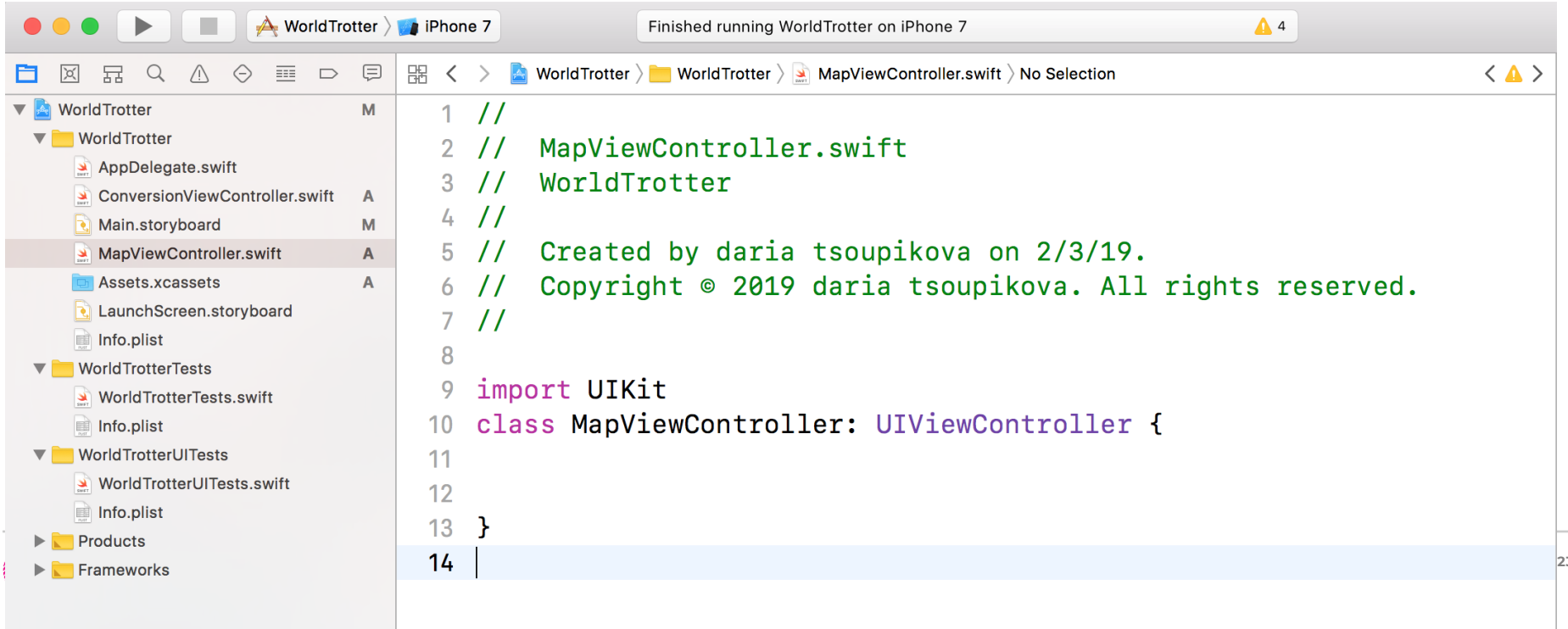




Optimize loading programmatically

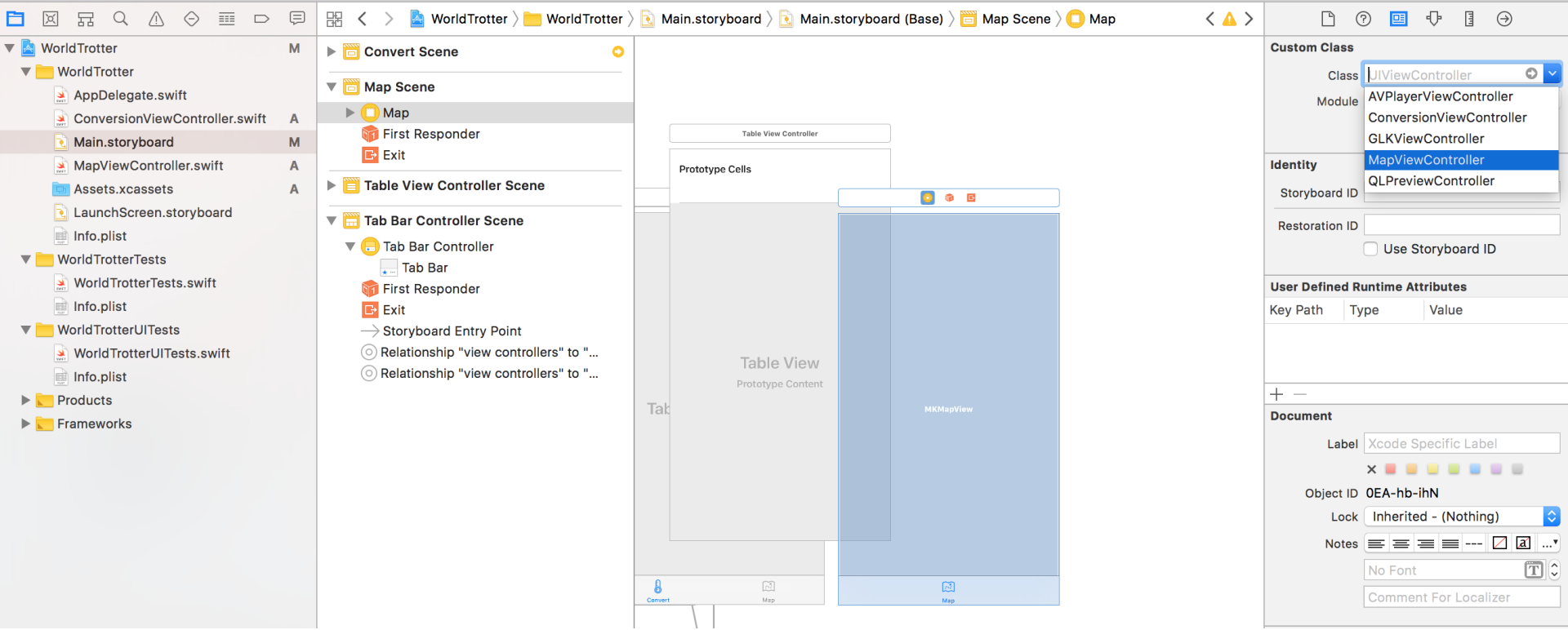
MapViewController is not needed until user taps on the menu item

Create new Swift file ”MapViewController”
Define a UIViewController subclass MapViewController



Optimize loading programmatically

Open Main.Storyboard and select map's view controller
Open identity inspector and change the class to MapViewController class.



Lazy loading

Memory is limited. Test and see how long it takes to load the map…

Lazy loading make app run faster and makes memory more efficient.
Defers allocation until user needs it.

- Optimization
- The controller’s view is not created until it needs to be presented on the screen
- Saves memory
- Improves performance

Optimize loading programmatically

In ConversionViewController.swift

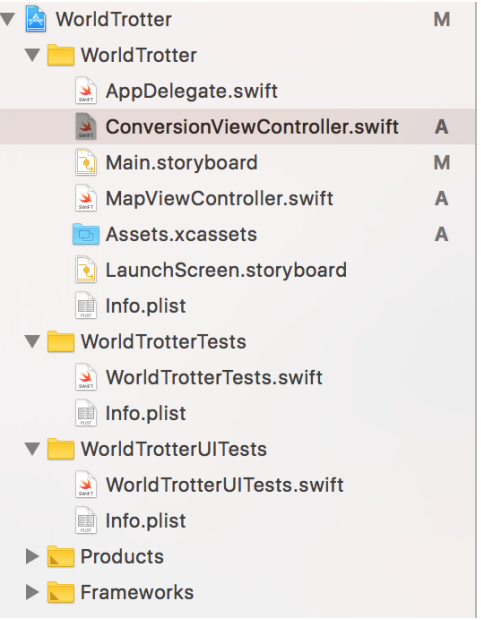
Update viewDidLoad()

```
override func viewDidLoad() {
super.viewDidLoad()

print("ConversionViewController loaded its view")


}
```

Optimize loading programmatically

In ConversionViewController.swift

Update viewDidLoad()




```
69              fahrenheitValue = nil
70          }
71      }
        @IBAction func dismissKeyboard(_ sender:
            UITapGestureRecognizer) {
73          textField.resignFirstResponder()
74      }
75
76
77      override func viewDidLoad() {
78          super.viewDidLoad()
79
80          print("ConversionViewController loaded its view")
81
```

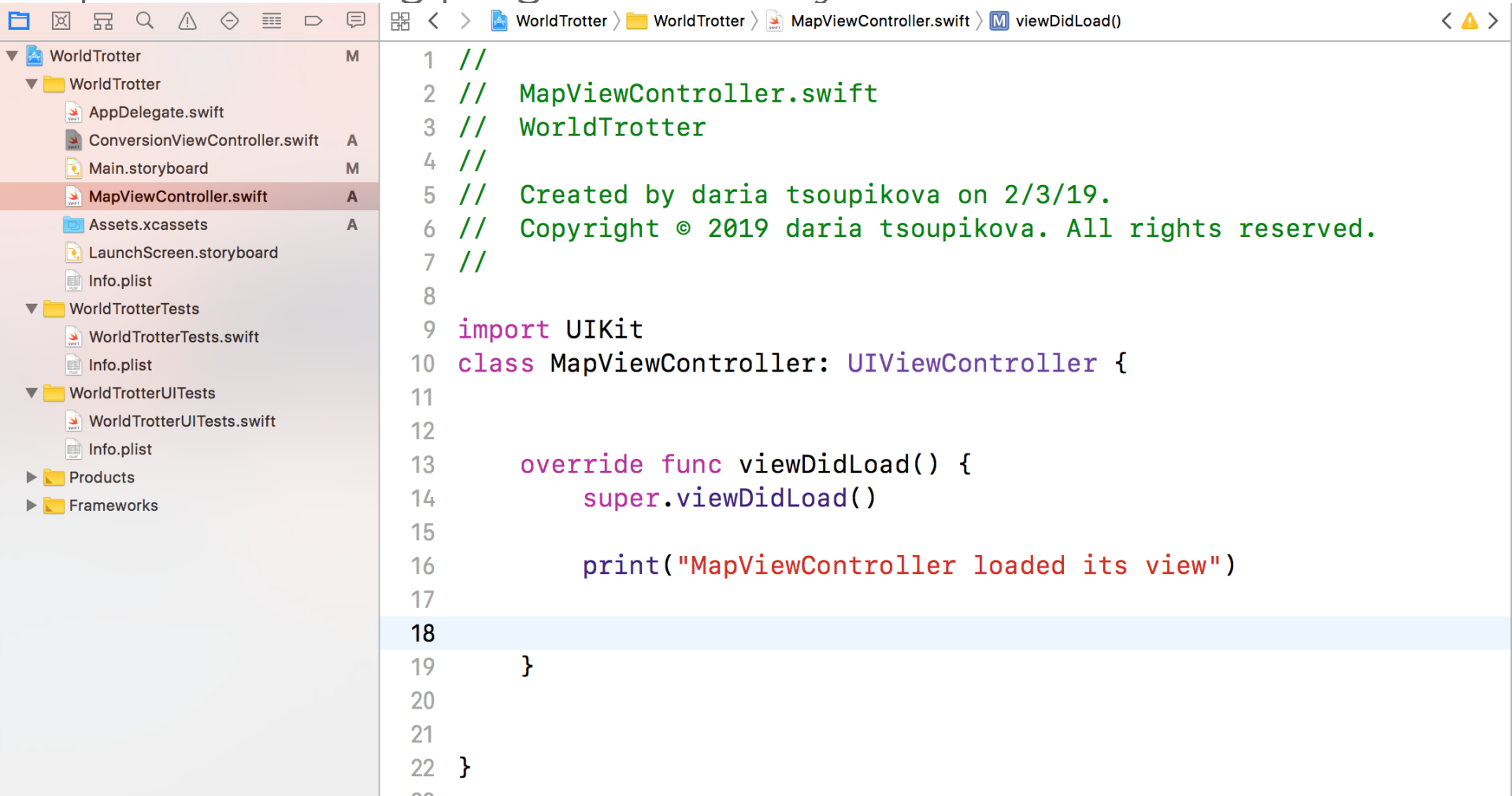
Optimize loading programmatically

In MapViewController.swift

Update viewDidLoad()

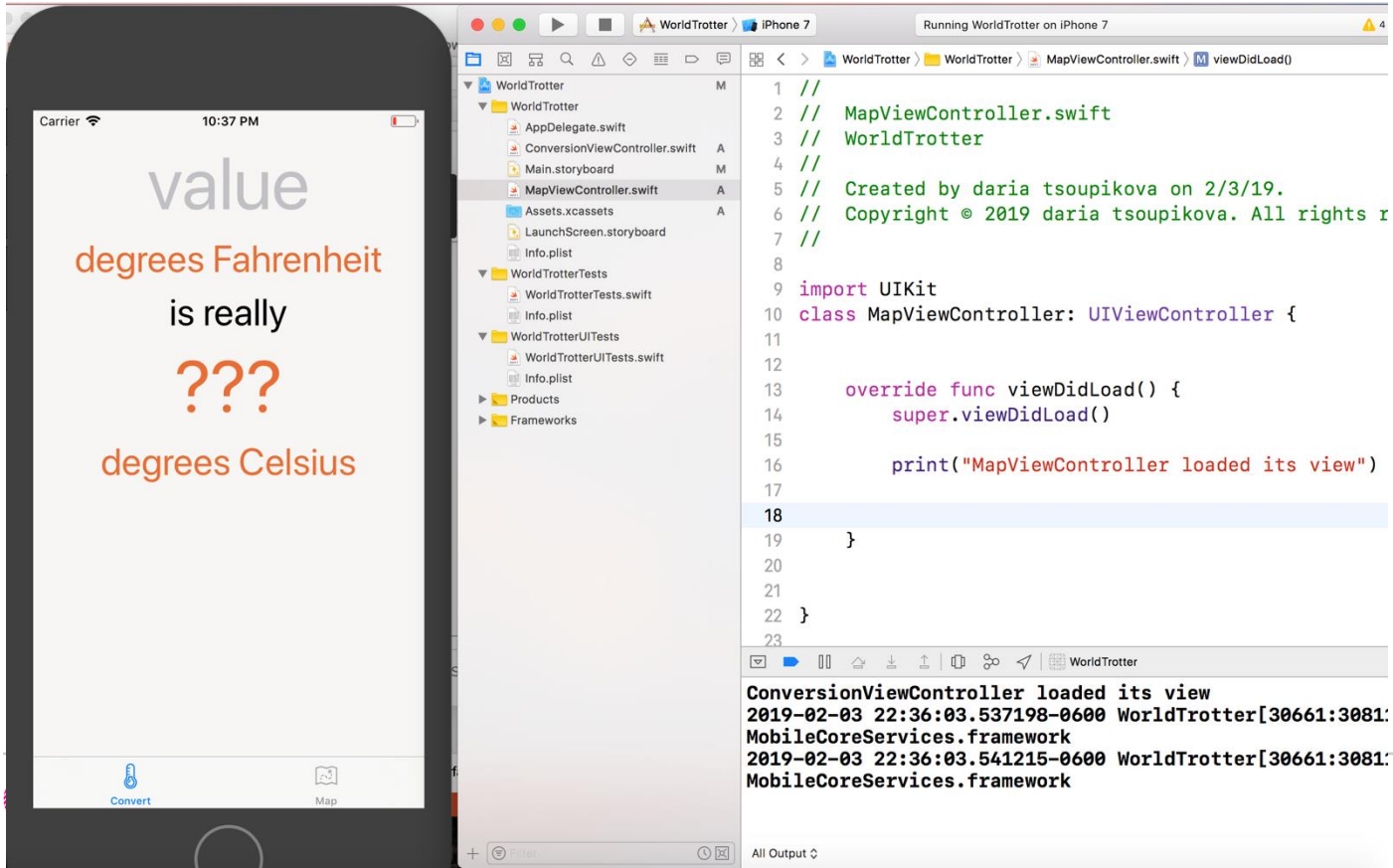
```
override func viewDidLoad() {
    super.viewDidLoad()
    // Do any additional setup after loading the view.

print("ConversionViewController loaded its view")
  }


}
```

Optimize loading programmatically



Optimize loading programmatically

Build, run and test.



Optimize loading programmatically

Build, run and test.