CS275
Fall 2020

iOS: View Controllers

BNRG CHAPTER 5

Topics

- View Controllers
- Navigation

View Controller

A View Controller manages a view hierarchy

- it's responsible for creating the view objects that make up that hierarchy
- and for handling events associated with the view objects in its hierarchy

A View Controller is an instance of a subclass of `UIViewController`

View Controller

Every "screen" of an app has a View Controller

- so to have a second screen--for a second task in this case--we will need a second view controller
- in WorldTrotter, the existing `ConversionViewController` will manage the F to C conversion views
- the new view controller will display a map

View Controller

- As a subclass of `UIViewController`, all view controllers inherit an important property:

```
var view: UIView!
```

- This property points to the root of the view hierarchy

- When the view of a view controller is added as a subview of the window, the view controller's entire view hierarchy is added

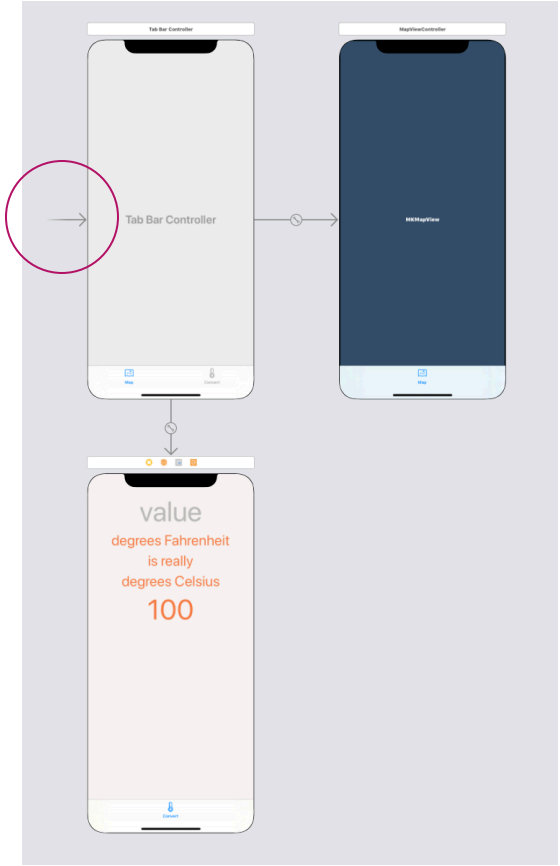
View Controller

There are two ways that a view controller can create its view hierarchy

1. in Interface Builder, by using an interface file such as a storyboard
2. programmatically, by overriding the `UIViewController` method `loadView()`

The Initial View Controller

- A storyboard for an app can have several view controllers
- But only one of them can be the initial view controller: the entry point into the storyboard
- In the storyboard, the initial view controller will have a big arrow pointing to it



Storyboard

- A storyboard is a visual representation of the user interface of an iOS application
- It shows screens of content and the connections between those screens
- A storyboard is composed of a sequence of scenes
- Each scene represents a view controller and its views
- Scenes are connected by segue objects, which represent a transition between two view controllers

Storyboards and Team Development

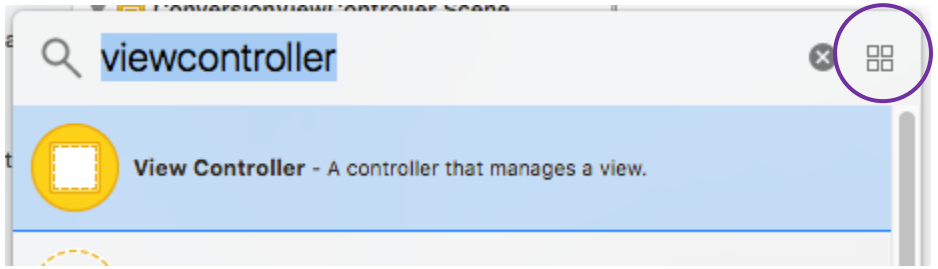
- Under the covers, a storyboard is encoded as an xml file
  - in fact, if you look at the storyboard as a file, you'll see that it is xml

- Important implication of this: for team development, it makes sense to split the UI into separate storyboards
  - if there is only a single storyboard (i.e. file) for the UI for an app, then it's awkward for more than one person to make changes to the UI for the app
  - if not done carefully, the result will be many ugly merge conflicts!
  - in a storyboard, it's possible to have a reference to a different storyboard
  - and this is the safest way to let a team work collaboratively on an interface

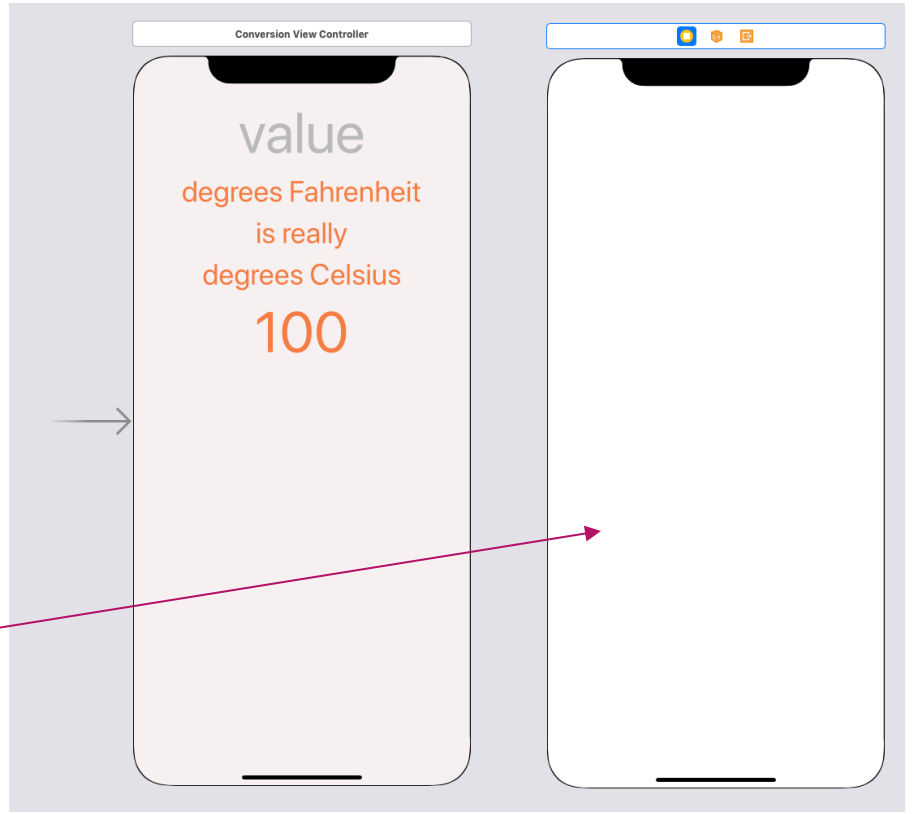
Creating a New View Controller

- By selecting a View Controller from the Object Library
- And dragging it onto the storyboard canvas

this is useful: switch to icon view

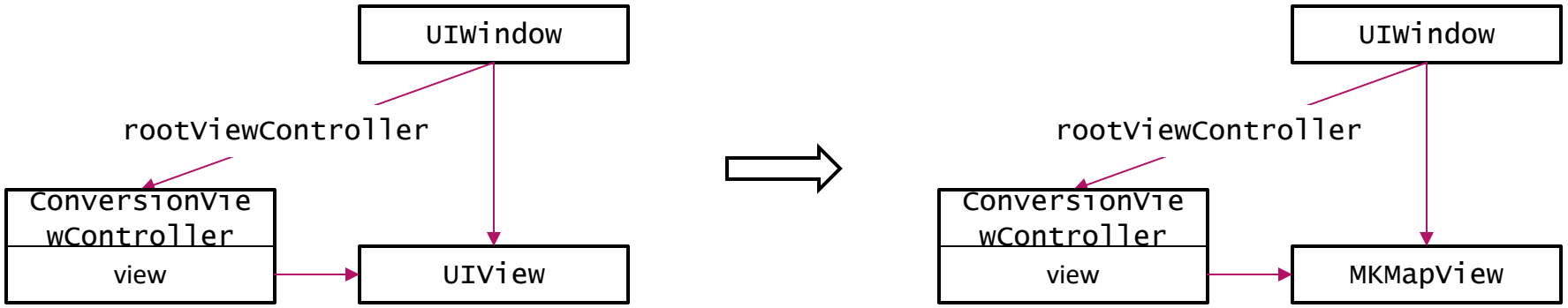


here's the new view controller



Creating a Map View

- `MKMapView` is the UI element that displays a map
- To make the map fill the entire window, delete the view controller's view and add an `MKMapView`



Initial View Controller

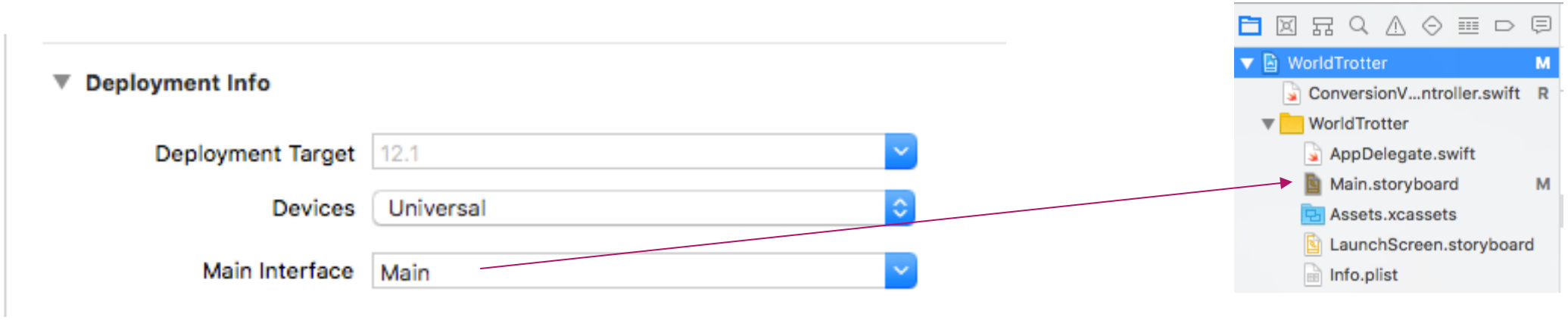
A storyboard can have many view controllers (= many separate screens)

But a storyboard has a single *initial view controller*

- this is the entry point into the storyboard
- it will correspond to the starting screen when the app first loads

Initial View Controller

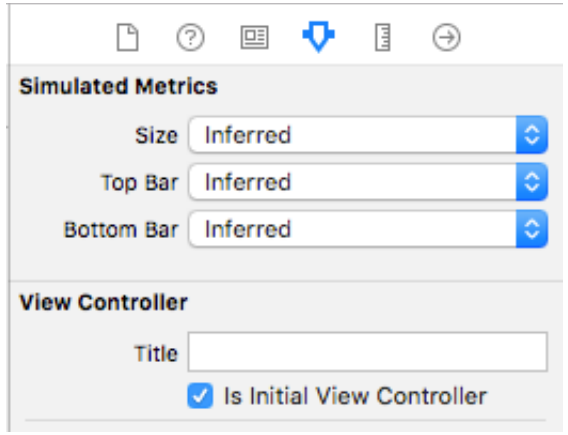
- An app has a `UIWindow` as the root of its view hierarchy
- `UIWindow` has a property `rootViewController`
- The app also has a one main user interface, which is a reference to a storyboard
- When the app launches, the initial view controller for the app gets set as the `rootViewController` for the window
- The main interface for an app is set in the project settings



Initial View Controller

To change the initial view controller:

- select the new View Controller in the hierarchy in the document outline, open the attributes inspector, and check the box next to Initial View Controller
- this will change the big arrow in the canvas so that it points to the new view controller

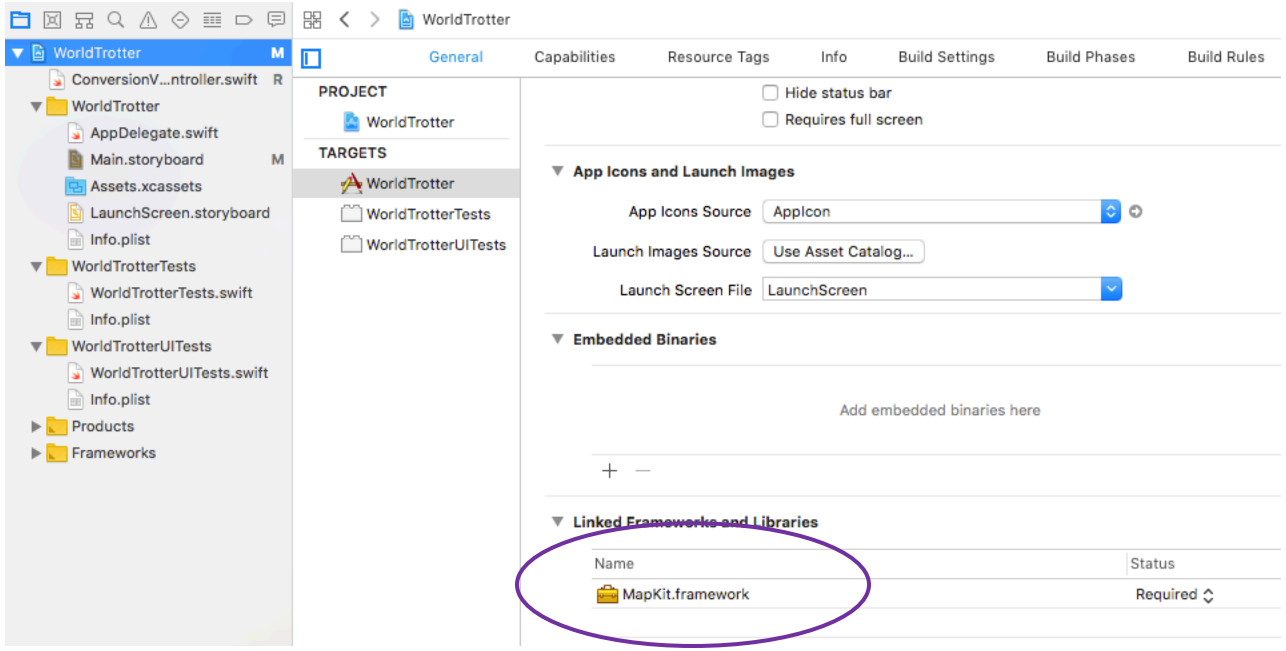


Frameworks

- A framework is a library of code that includes associated resources such as interfaces and images
- UIKit and Foundation are frameworks
- MKMapView requires a different framework: MapKit
- A quirk (feature, annoyance) of Xcode is that even if we put `import MapKit` in the source-code file, the compiler will optimize it out
  - since we haven't actually put in any code yet that explicitly uses MapKit
- Solution: must add MapKit as a Linked Framework

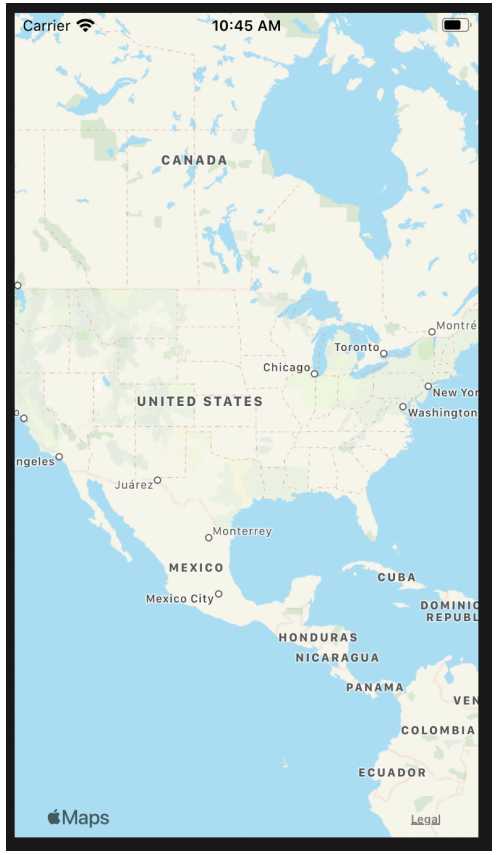
Adding MapKit to the Project

Adding a linked framework:



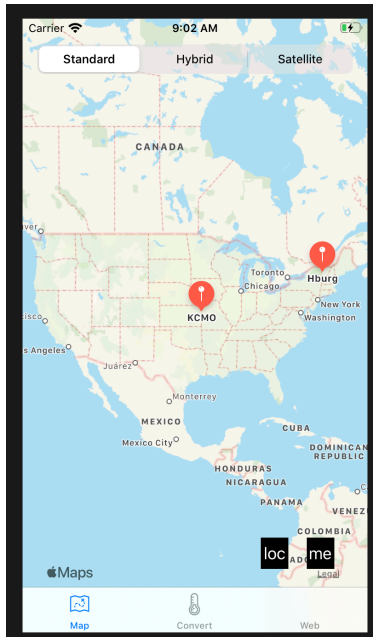
MKMapView in Action

Here's what it looks like:

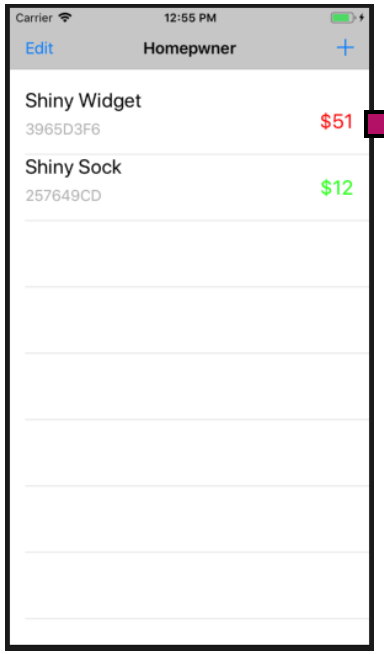


Navigation

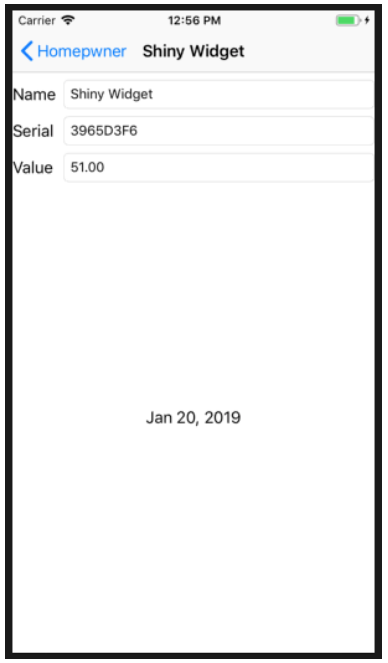
There are different mechanisms for moving from one screen to a different screen



Tab Bar Controller



View Segue

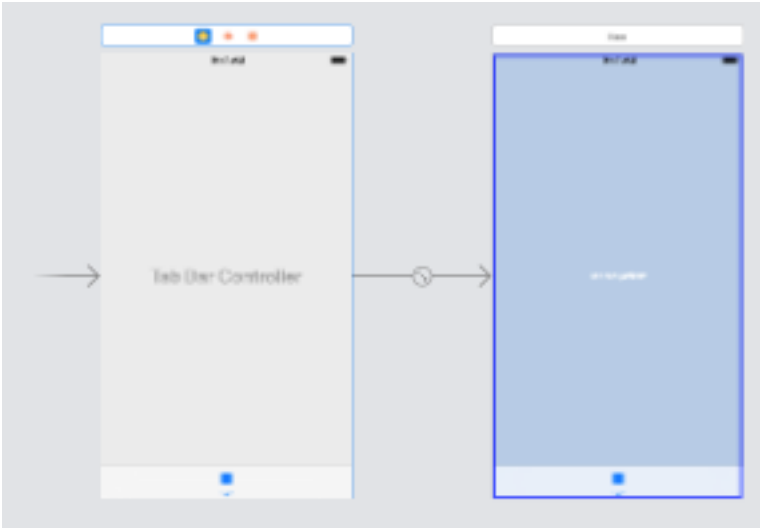


`UINavigation` Controller

UITabBarController

- `UITabBarController`: the view controller for a navigation bar
- Lets us switch between view controllers
- `UITabBarController` contains an array of view controllers
- It also puts a graphic navigation bar at the bottom of the screen

- To implement this:
  - Select the View Controller (the one for the map) and do Editor -> Embed In -> Tab Bar Controller
  - This also makes the Tab Bar Controller the initial view controller for the storyboard



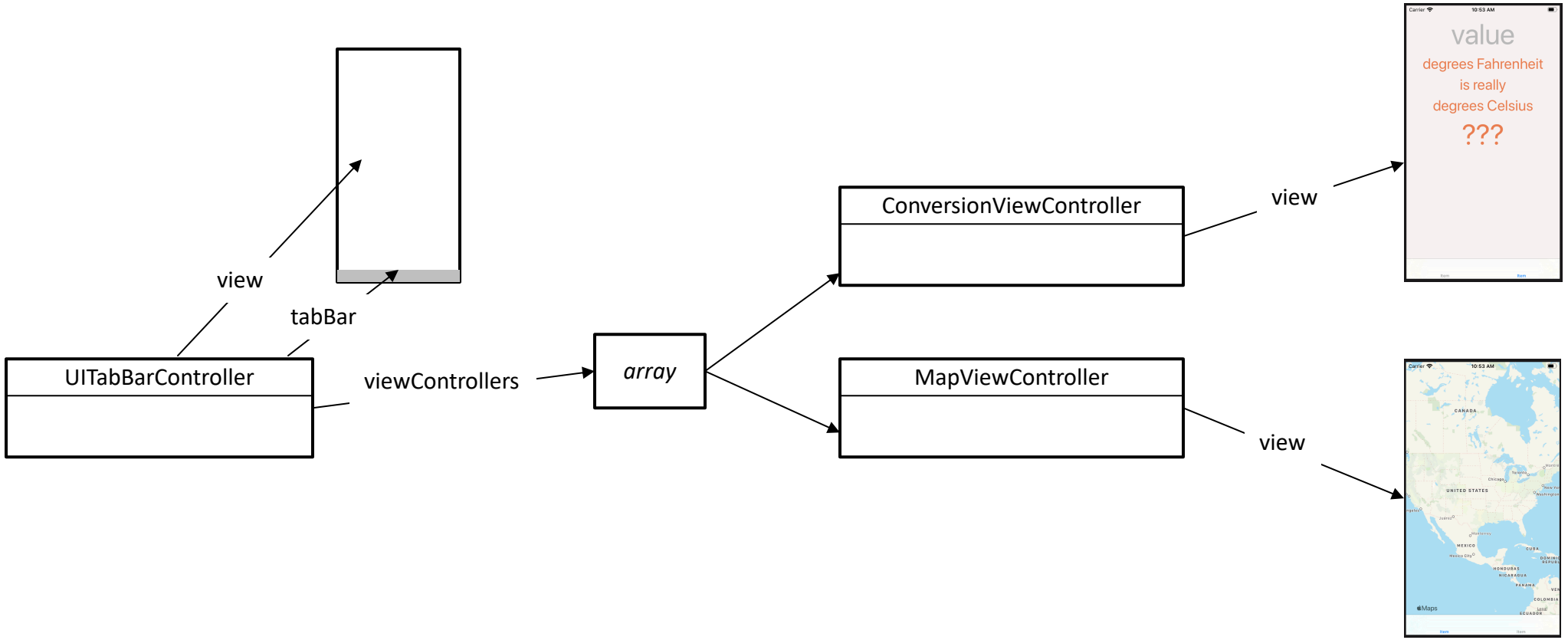
Tab Bar Controller in Action

Here's what it looks like:

Each entry in the tab bar has a title and and an icon (must set these during the design of the UI)

UITabBarController

Here are the relationships between the parts of the interface:



Tab Bar Items

- Each tab on the tab bar can display a title and an image
- Each view controller maintains a `UITabBarItem` property for this purpose

`UITabBarItem`
title image

Assets

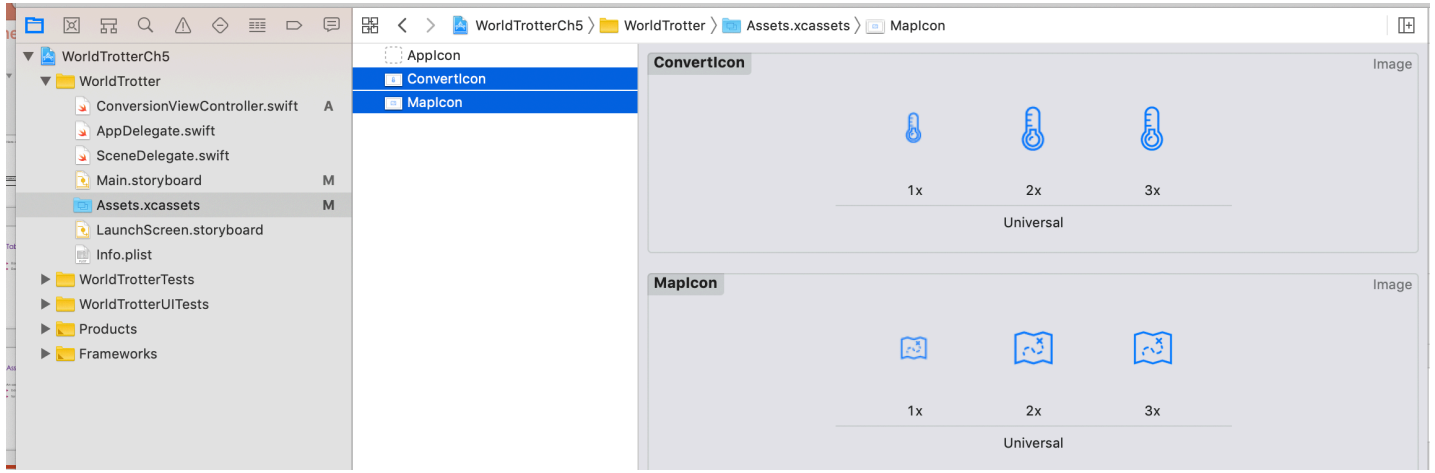
An asset is a set of files from which a single file will be selected at runtime

- based on the active configuration of the device
- for example, an image might have different sizes to match different pixel densities

Adding Assets

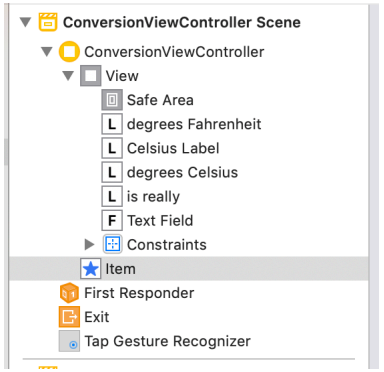
To add images to the Assets Catalog, for displaying in the Tab Bar

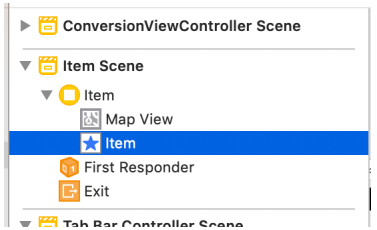
- get png images from the download described in the book
- drag the three files (*.png, *@2x.png, *@3x.png) to the Assets Catalog

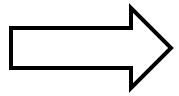


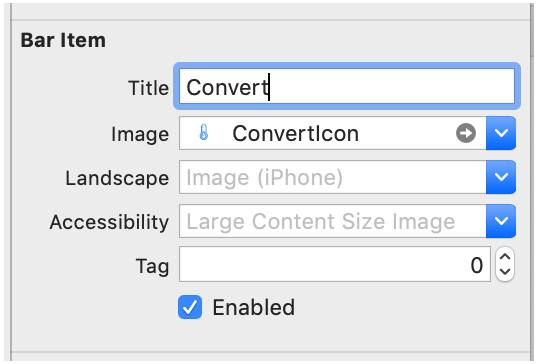
Setting Tab-Bar Images

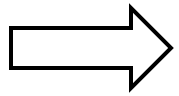
Set the title and image for the tab bar items

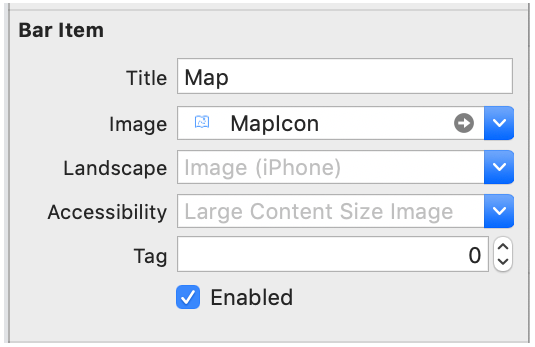












`viewDidLoad()` and Lazy Loading

- The `viewDidLoad()` method on a view controller is called by the UI framework when that view controller finishes loading its view
- This loading happens only one time during the lifetime of an app

```
class MapViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()
        print("MapViewController loaded its view.")
    }
}
```

Accessing Subviews

- The OS will call `viewWillAppear(_:)` each time before the view actually appears each on the screen—this gives you a chance to configure the view each each time it will be displayed on the screen

```
override func viewWillAppear(_ animated: Bool) {
    super.viewWillAppear(animated)
    print("MapViewController view will appear")
}
```

- This gives us a hook to modify views created through IB
- And there are additional view-lifecycle methods (p. 101)

Challenge

Do the Silver Challenge: Dark Mode