

Mobile Application Development

Lecture 16

Controllers of View Controllers

Lecture Summary

- Multiple MVCs
- UINavigationController
- Segues
- UITabBarController
- DEMO



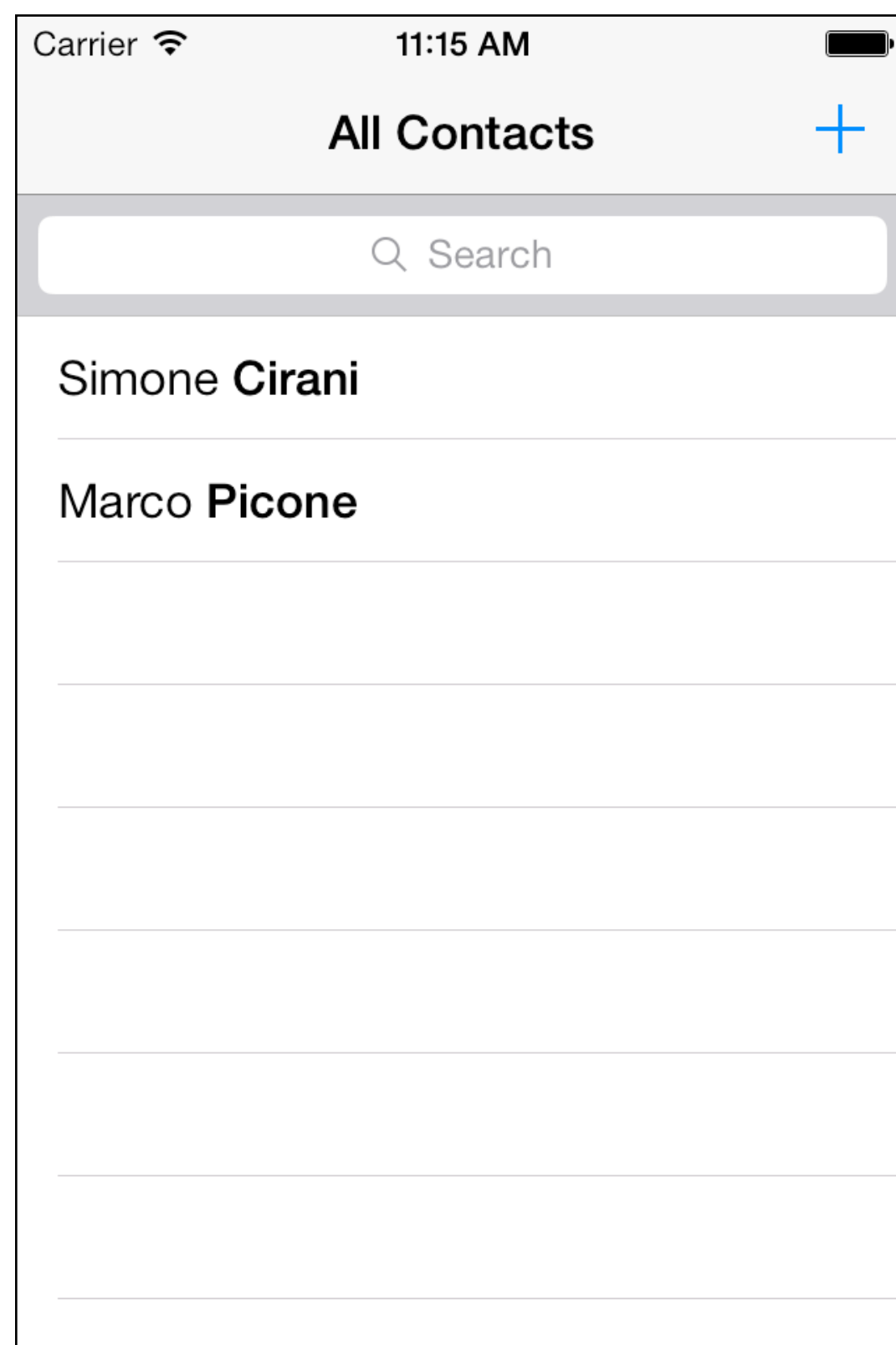
Multiple MVCs

- Complex iOS applications typically rely on multiple views to show content
- For instance, it might not be possible to display all the content inside a single view, so another view might be needed to display additional content
- Multiple views also make the flow of the application more natural and intuitive for the user
- Handling multiple views means that we also have multiple view controllers that must coordinate the flow of the application
- Each view is controlled by its own view controller, thus multiple MVCs come into play in an application

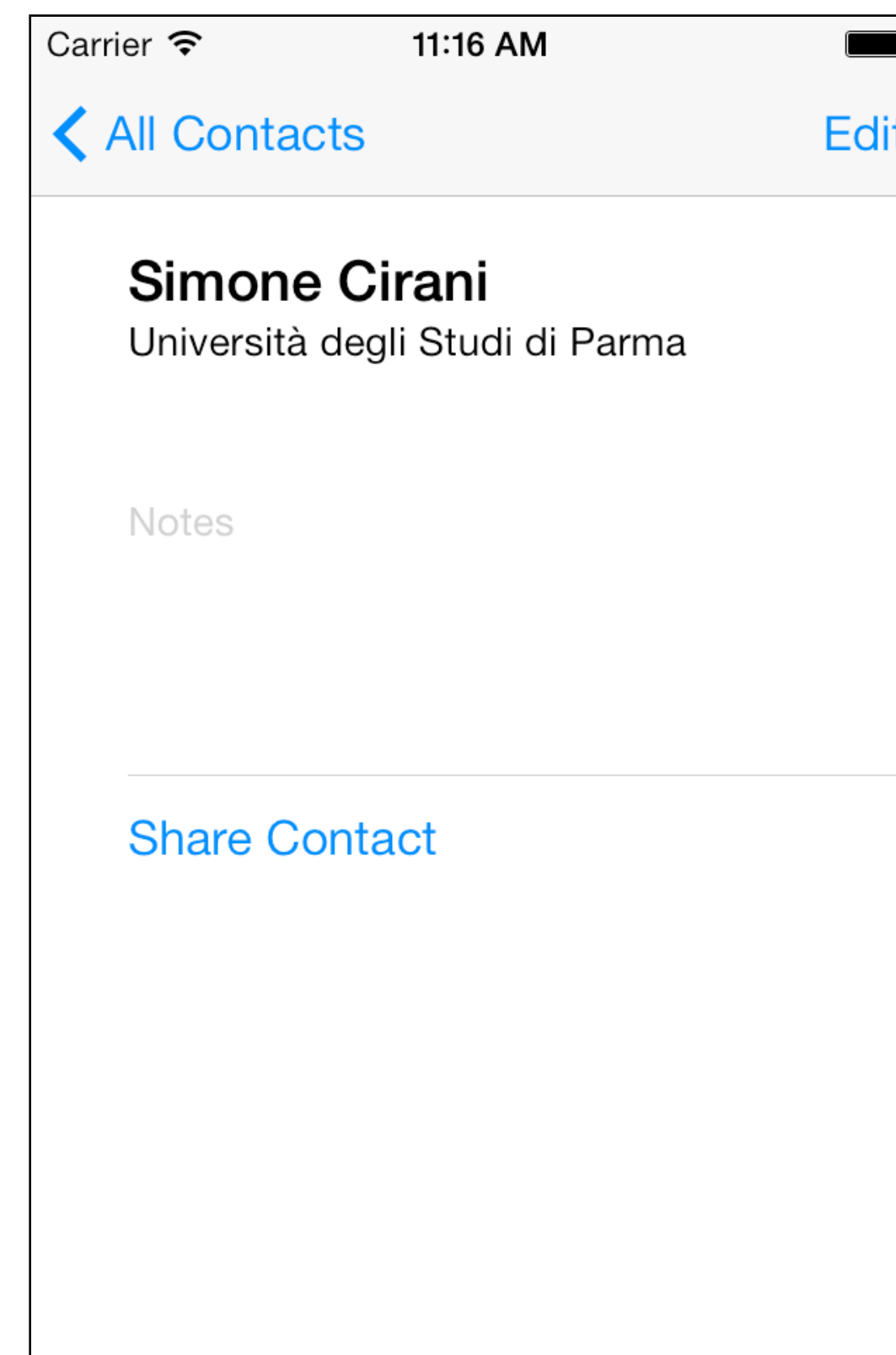
Multiple MVCs

- As in good OOP design, view controllers should be highly specialized and independent from each other
- Each view controller should be responsible to manage a view that display some specific content (e.g. a contact list, a contact details)
- Apple provides great examples of apps with multiple MVCs:
 - Contacts app
 - Calendar app
 - Music app

Multiple MVCs: Contact app



Contact list

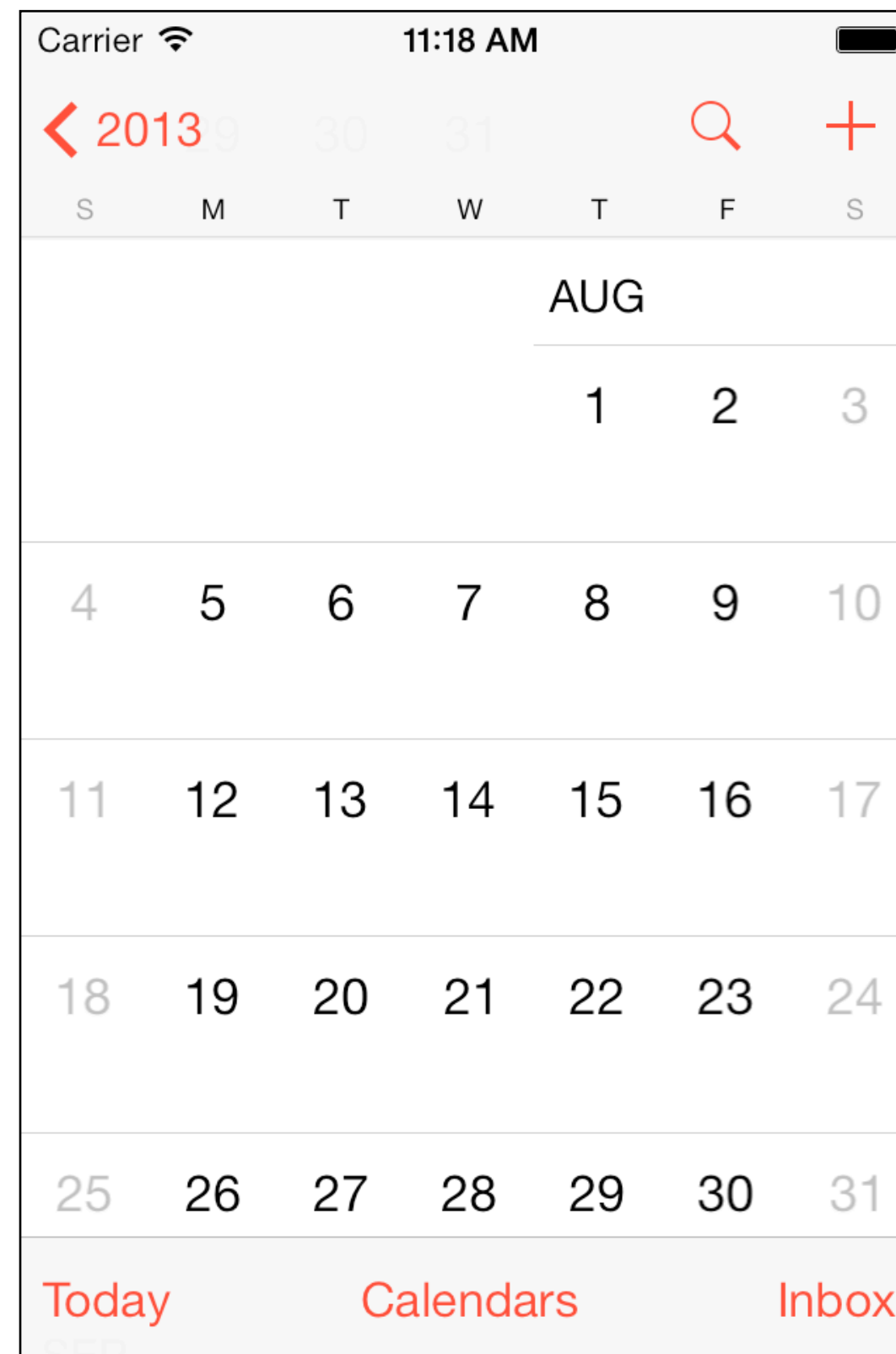


Contact details

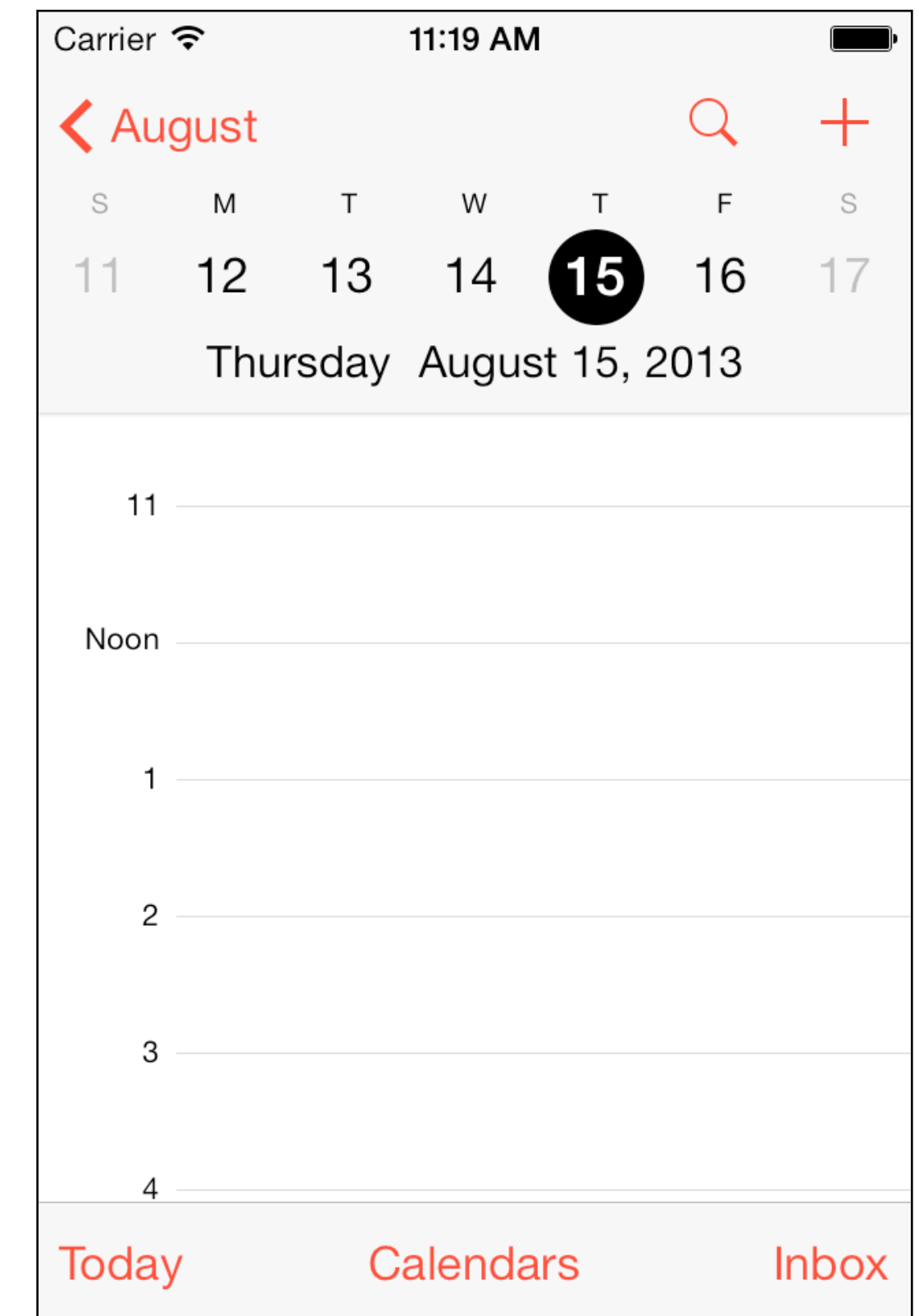
Multiple MVCs: Calendar app



Year view



Month view



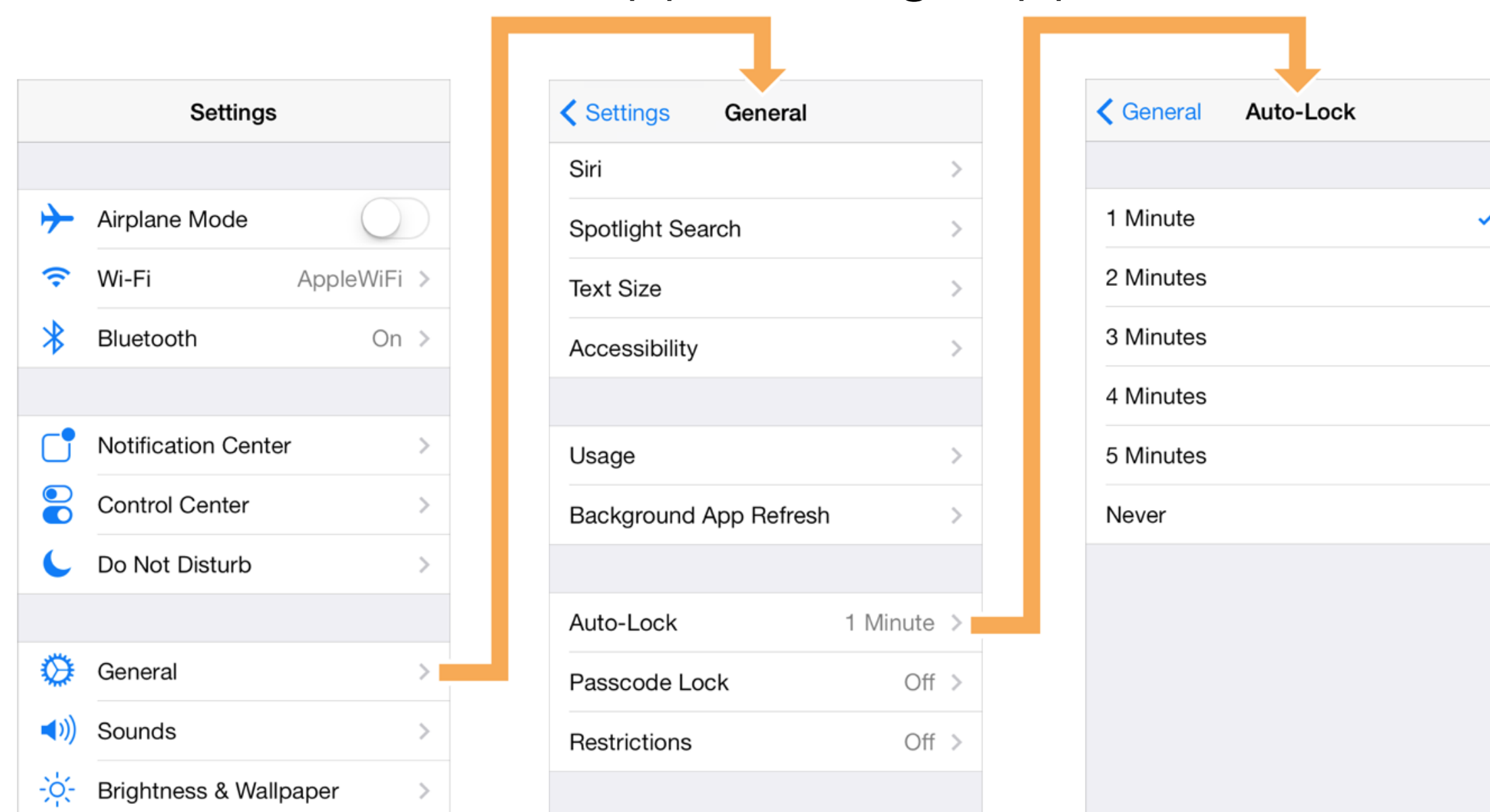
Day view

Multiple MVCs

- Multiple view controllers can be rendered in different ways depending on what content is going to be displayed throughout the app
- The management of the transition among different MVCs is provided by dedicated controllers
- Controllers of view controllers:
 - UINavigationController
 - UITabBarController
- Transitions among MVCs are called **segues**
- Segues can be triggered by controls in the view or by certain events that might occur in the app, depending on the application logic
- Other view controllers are added to the storyboard:
 1. drag a UIViewController from the object palette
 2. create a subclass of UIViewController using New File menu item
 3. in the Attributes inspector, set the class of the UIViewController to the newly created subclass

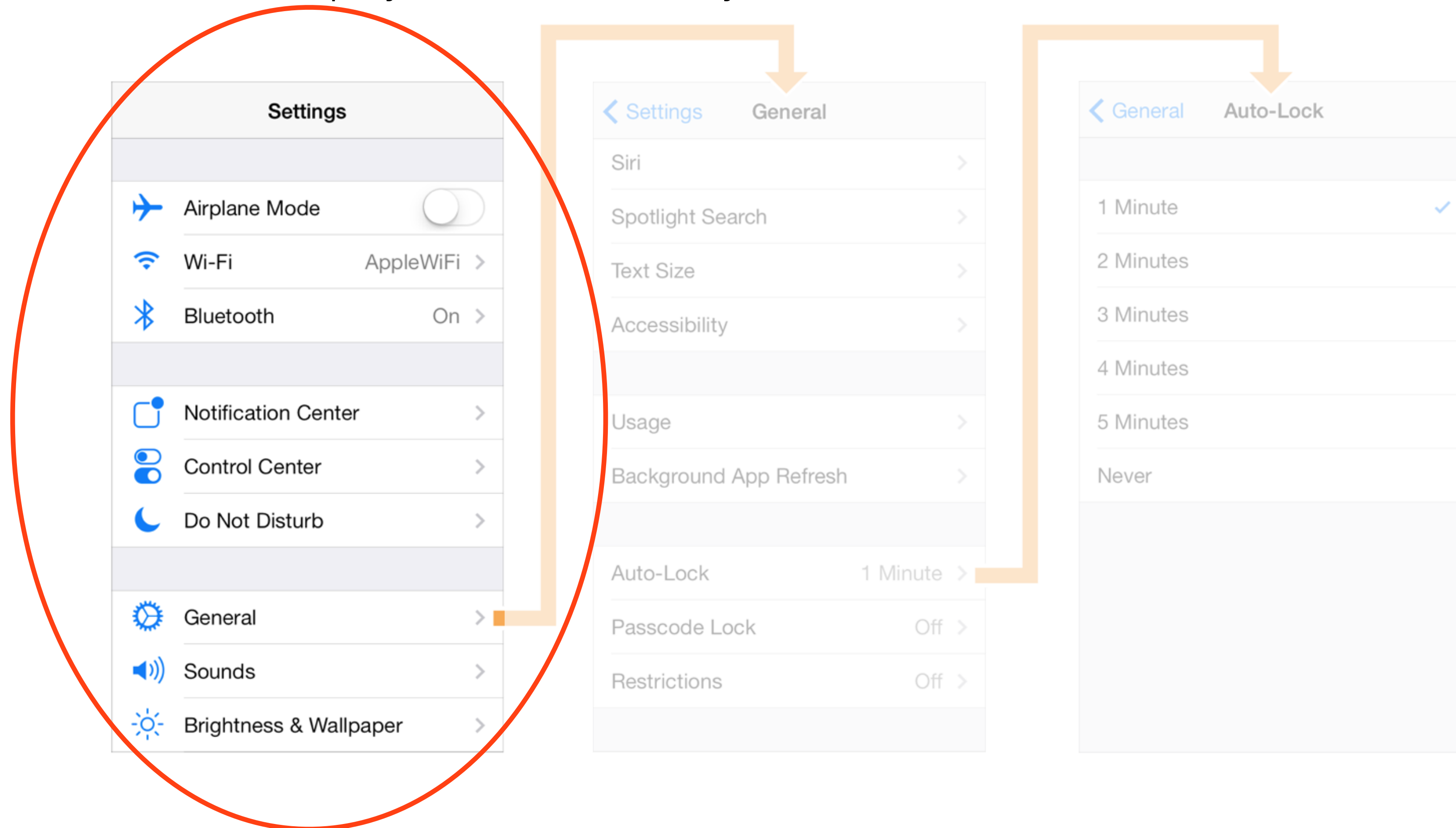
UINavigationController

- `UINavigationController` is a class that implements a specialized view controller that manages the navigation of hierarchical content
- Typical usage: **drill-down to detailed content** (e.g. Calendar app: year → month → day)
- The screens presented by a navigation interface typically mimic the hierarchical organization of data
- Example of a navigation controller-based app: Settings app



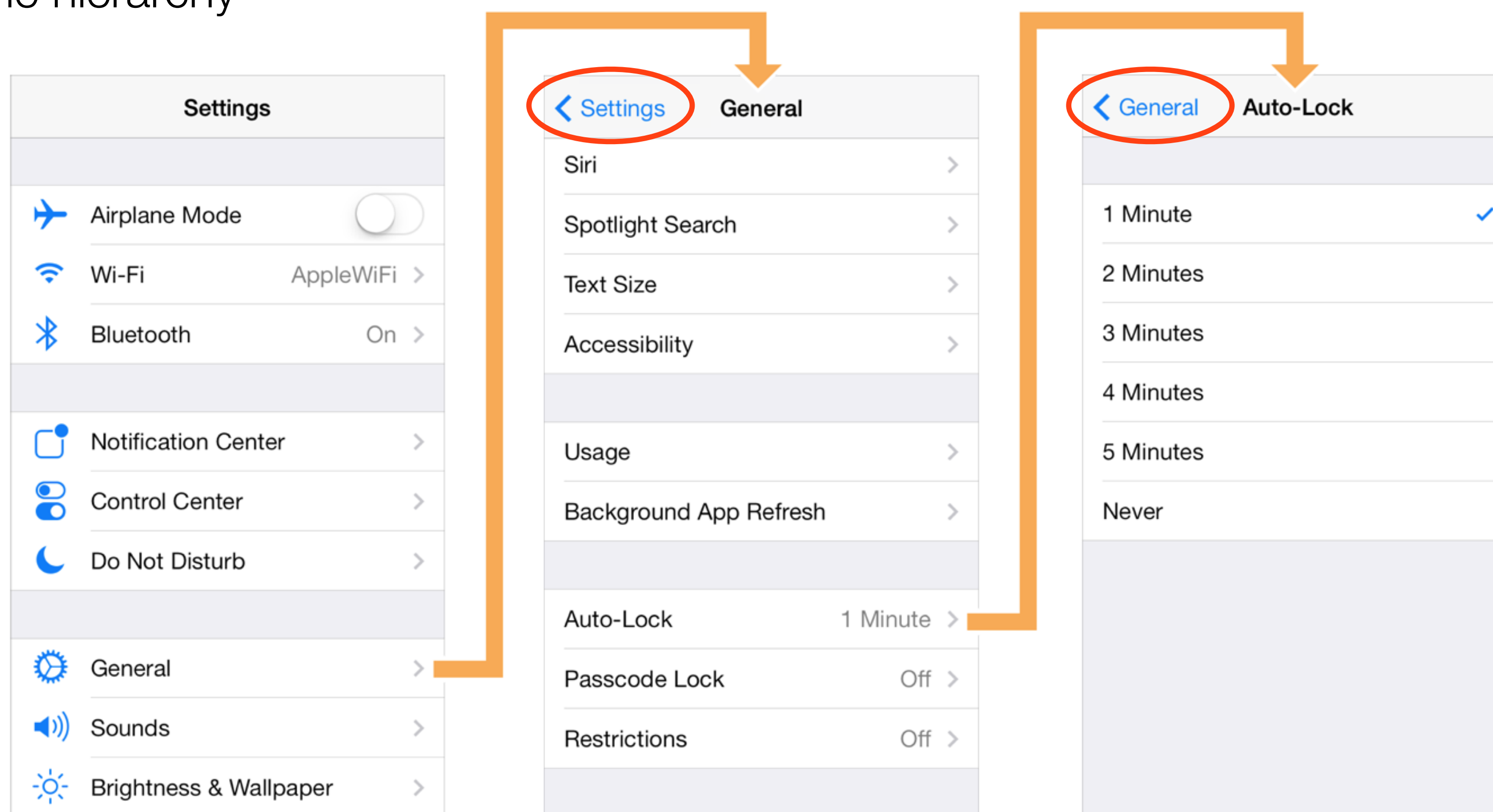
UINavigationController

- The first view to be displayed in the hierarchy is called **root view controller**



UINavigationController

- For all but the root view, the navigation controller provides a back button to allow the user to move back up the hierarchy

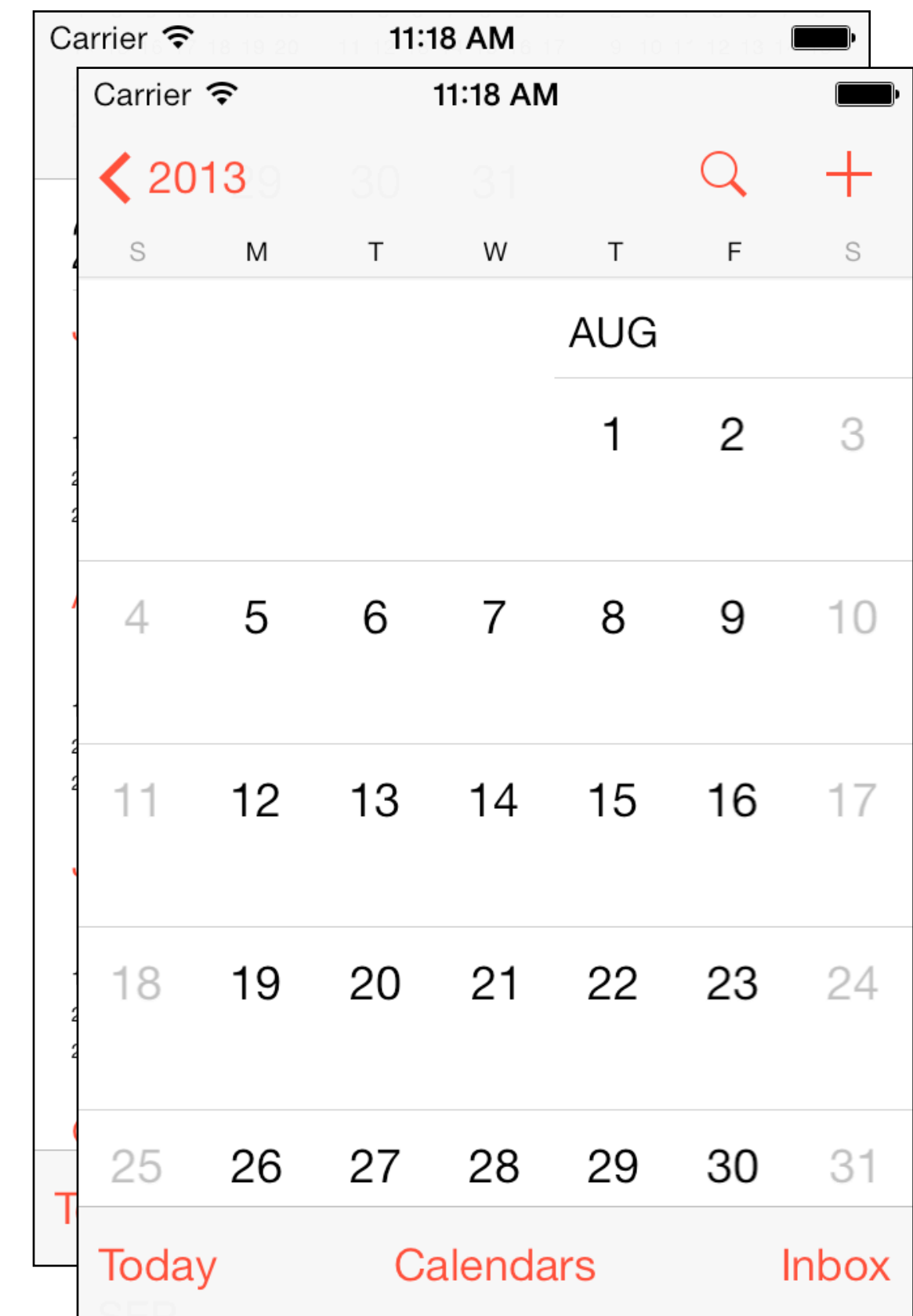
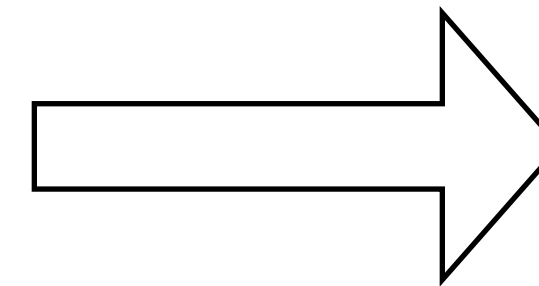
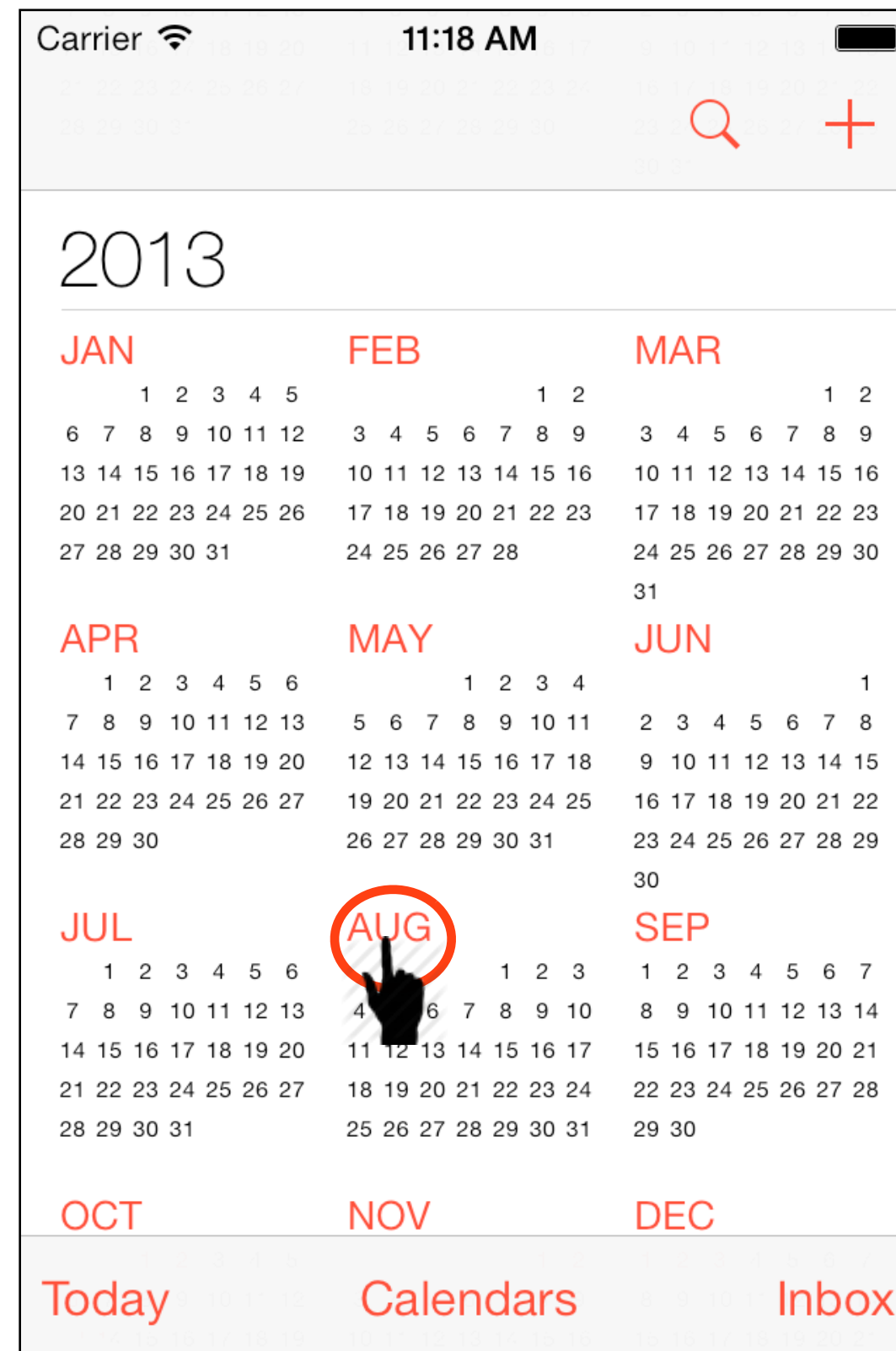


UINavigationController

- The navigation controller manages a **stack of view controllers**

The year view is the root view controller

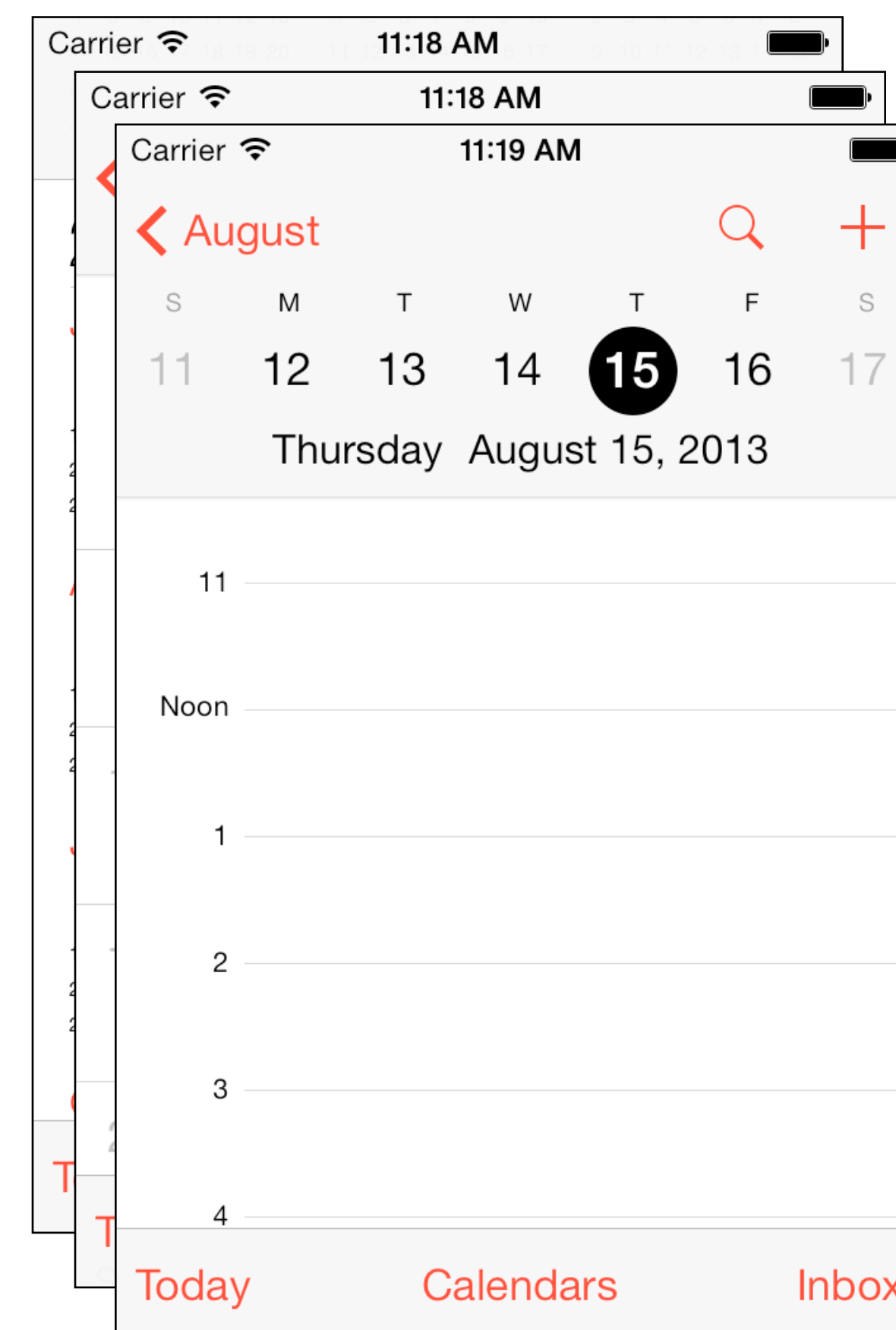
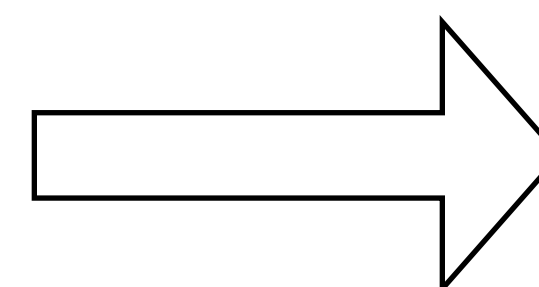
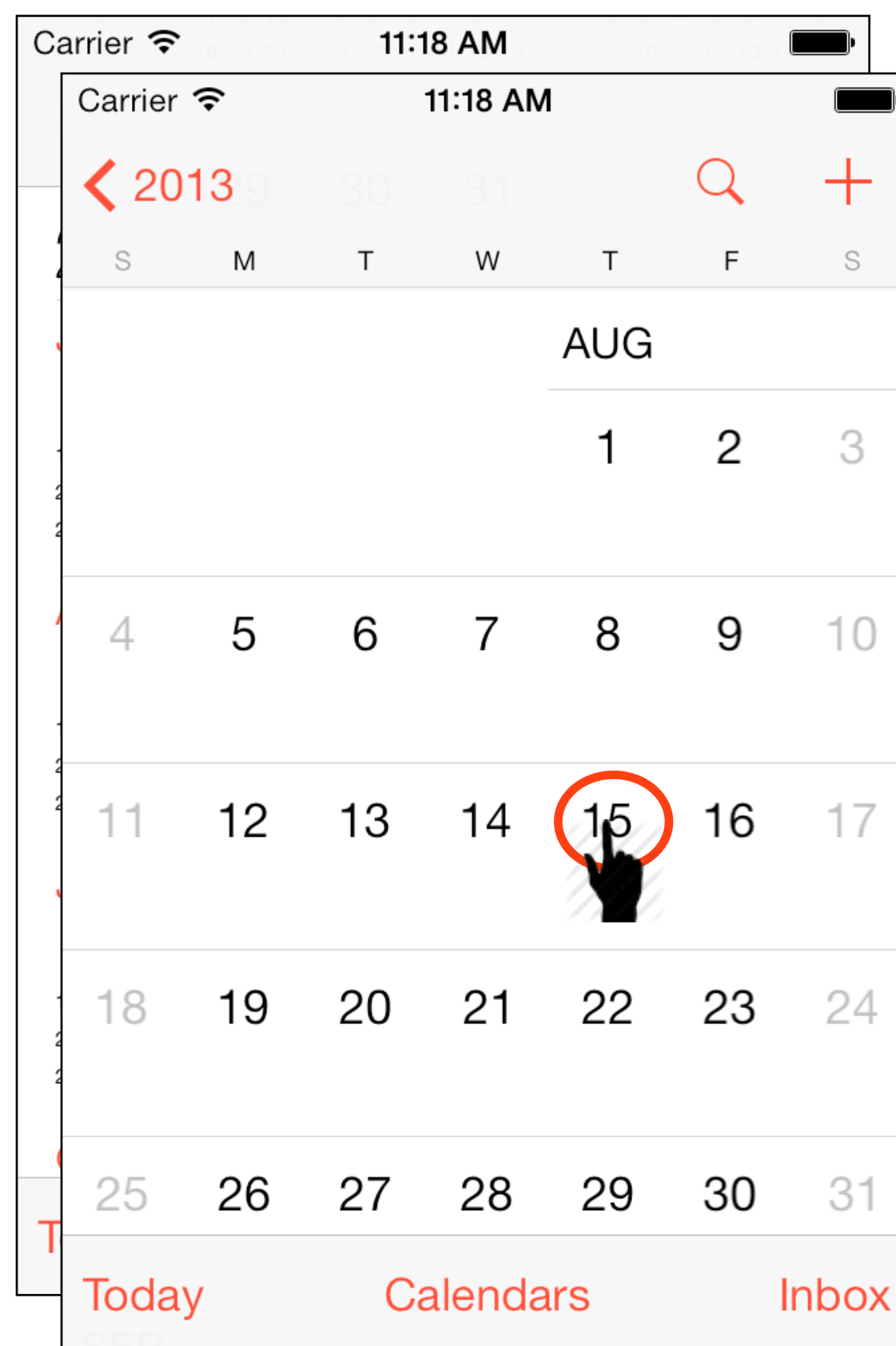
When a month is tapped, the month view is put at the top of the stack



UINavigationController

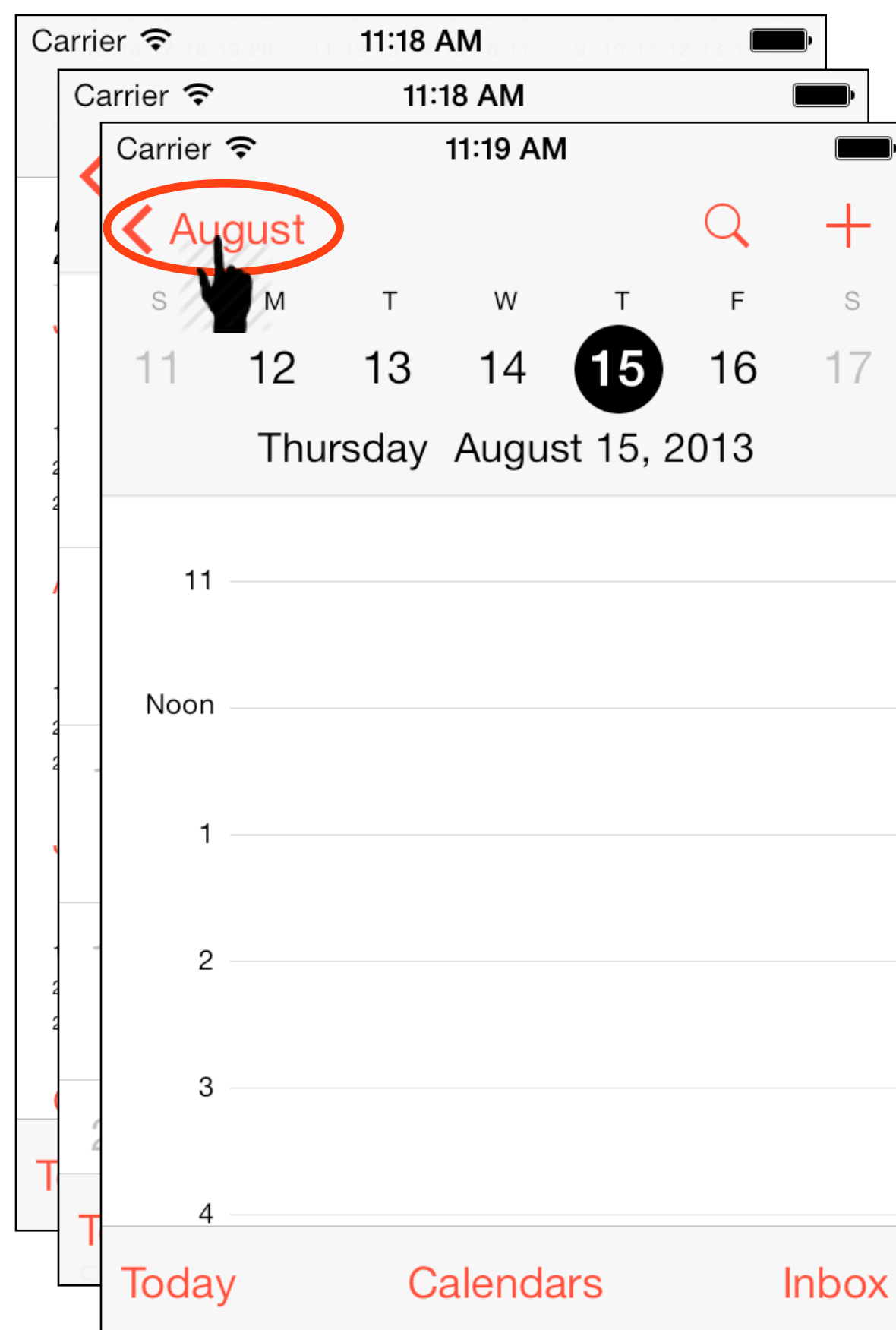
- The navigation controller manages a **stack of view controllers**

When a day is tapped, the day view is put at the top of the stack

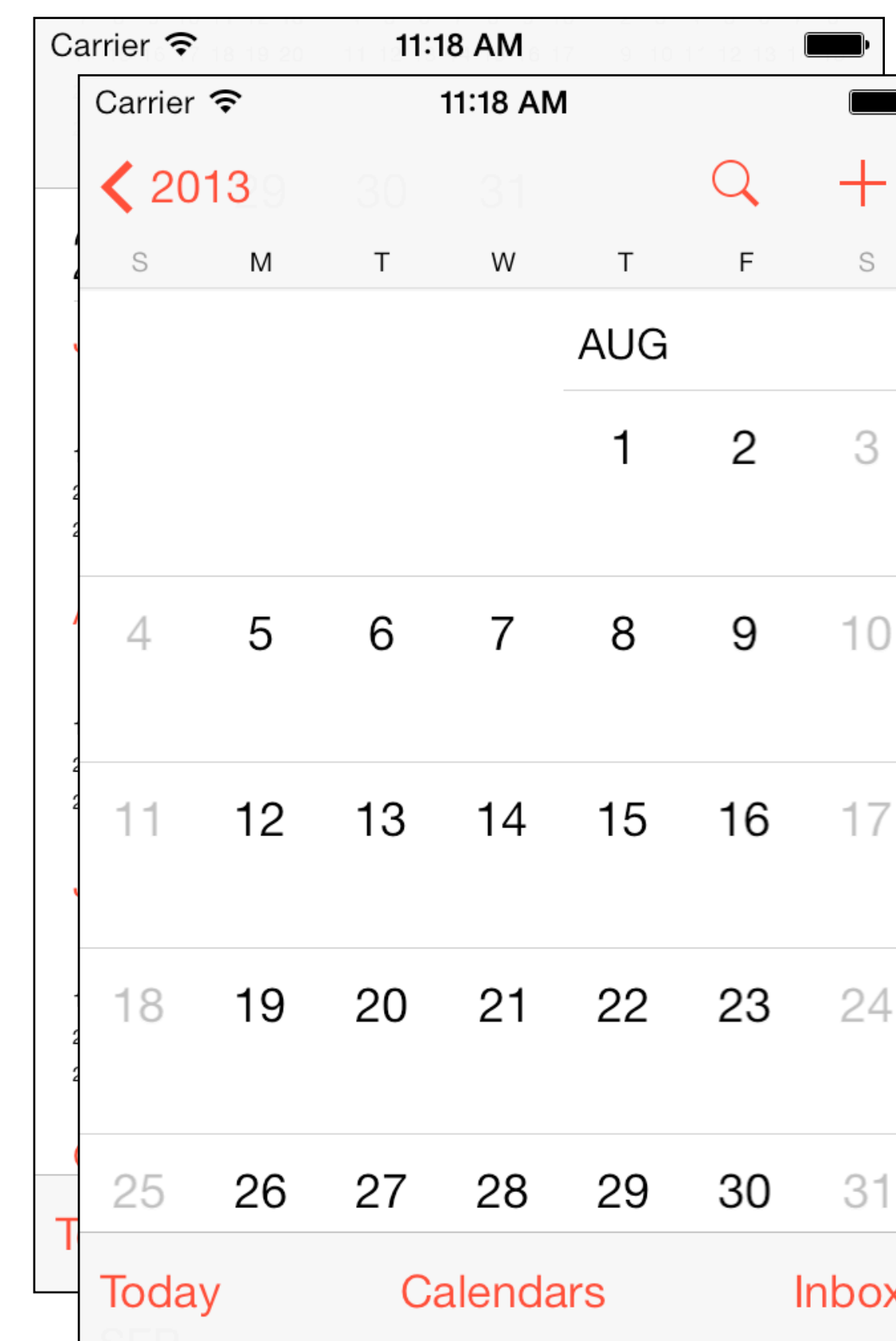
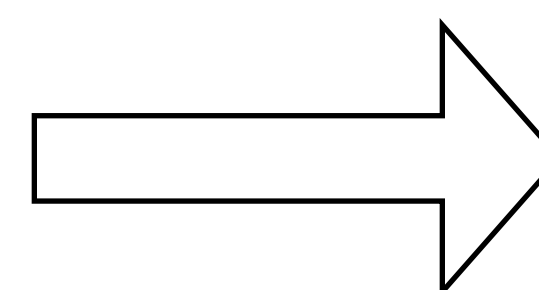


UINavigationController

- The navigation controller manages a **stack of view controllers**



Tapping the back button pops the view from the stack

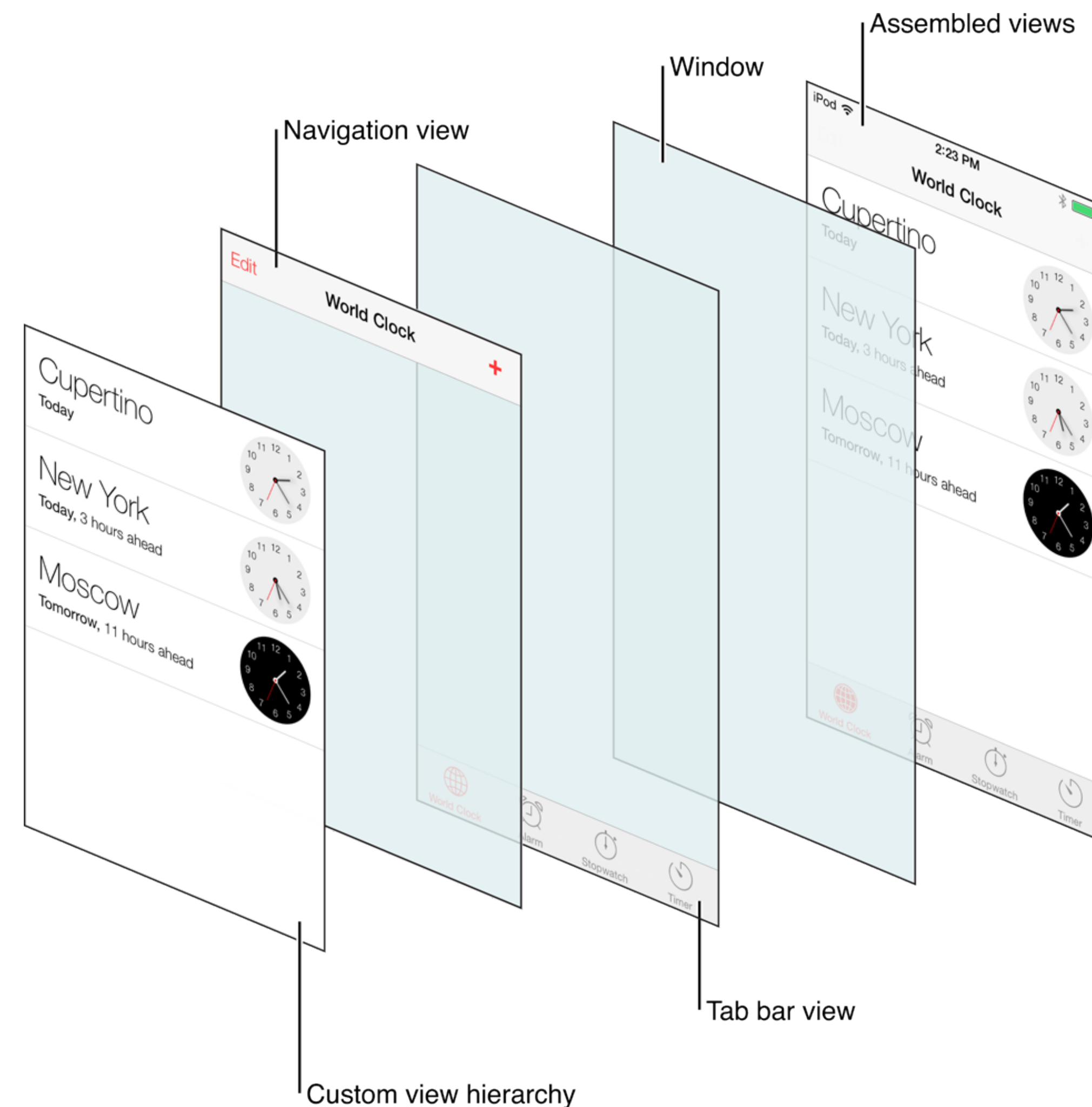


UINavigationController

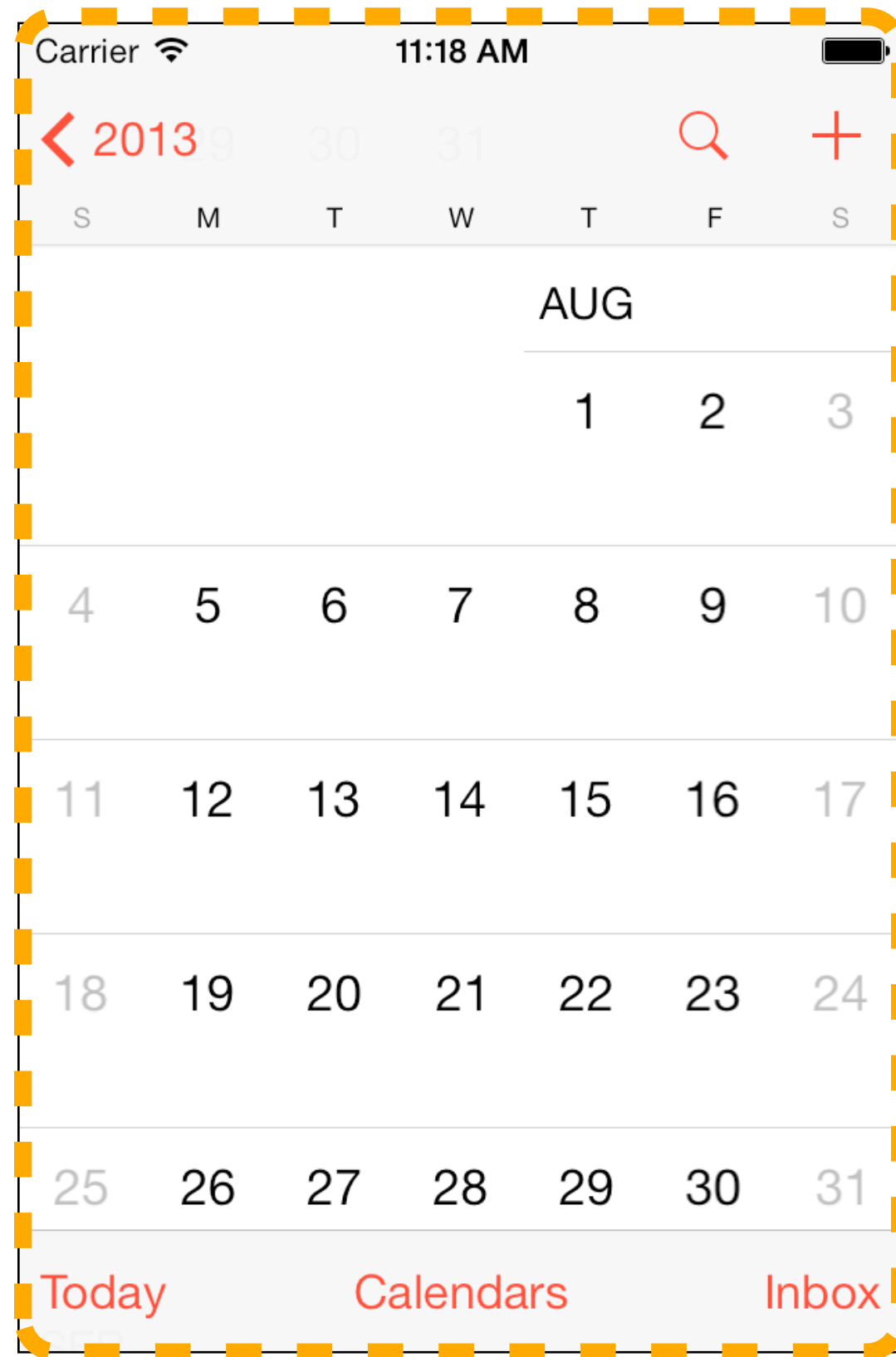
- The navigation controller manages a **stack of view controllers**
- The navigation controller object provides methods to modify the stack at runtime:
 - `(void)pushViewController:(UIViewController *)viewController animated:(BOOL)animated` is used to push a view controller at the top of the stack
 - `(UIViewController *)popViewControllerAnimated:(BOOL)animated` is used to pop a view controller from the top of the stack
 - `(NSArray *)popToViewController:(UIViewController *)viewController animated:(BOOL)animated` is used to pop all view controllers from the stack until the specified view controller is at the top of the stack
 - `(NSArray *)popToRootViewControllerAnimated:(BOOL)animated` is used to pop all the view controllers on the stack except the root view controller
- Every `UIViewController` has a `navigationController` property that can be used to access the `UINavigationController` it is embedded in

UINavigationController

- The view for a navigation controller is just a container for several other views:
 - a navigation bar
 - an optional toolbar
 - the view containing custom content
- When moving from a MVC to another, the content of the custom view changes, as well as of the navigation bar and toolbar
- Only the custom content view changes to reflect the view controller that is at the top of the navigation stack

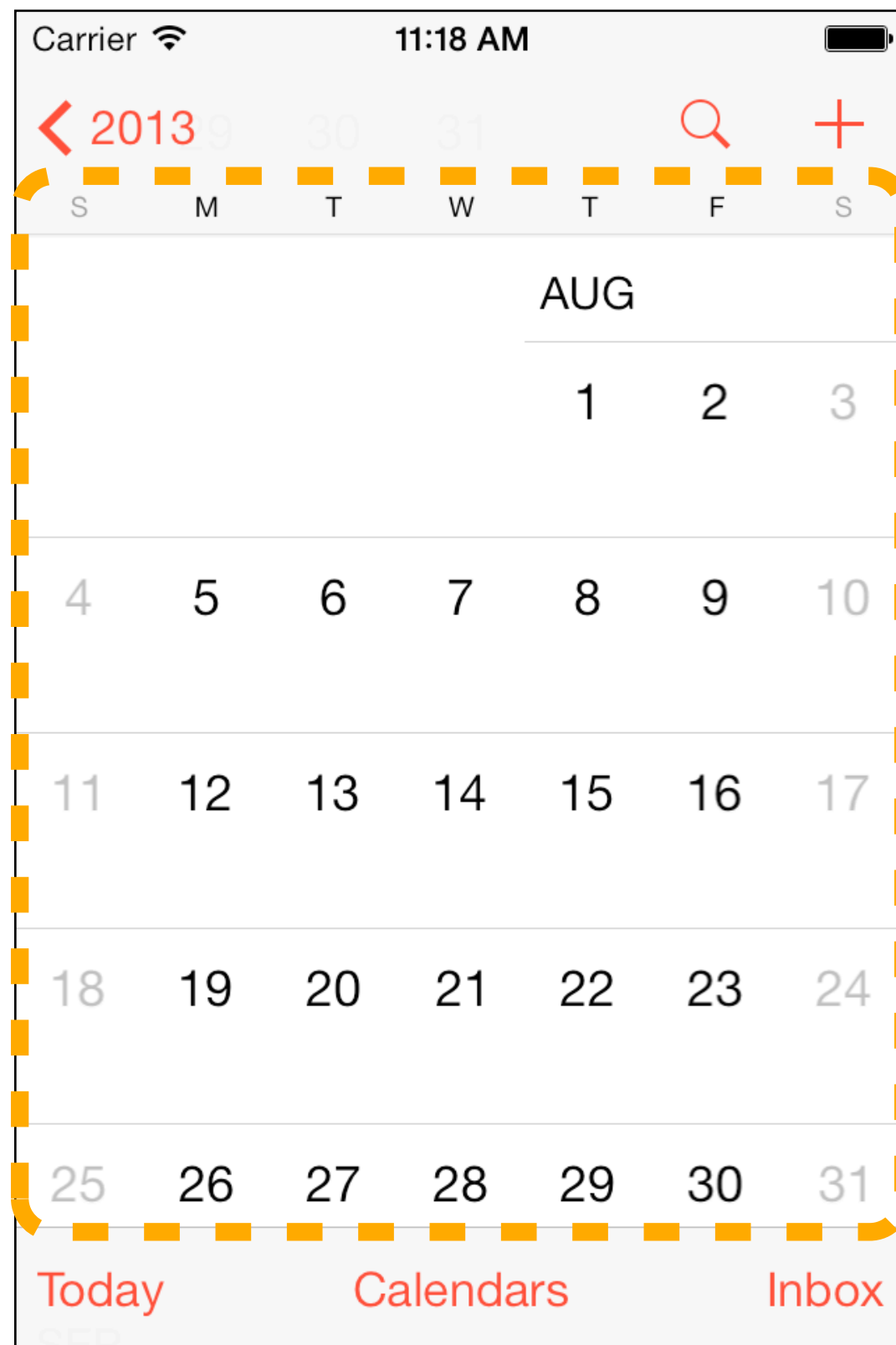


View of a UINavigationController



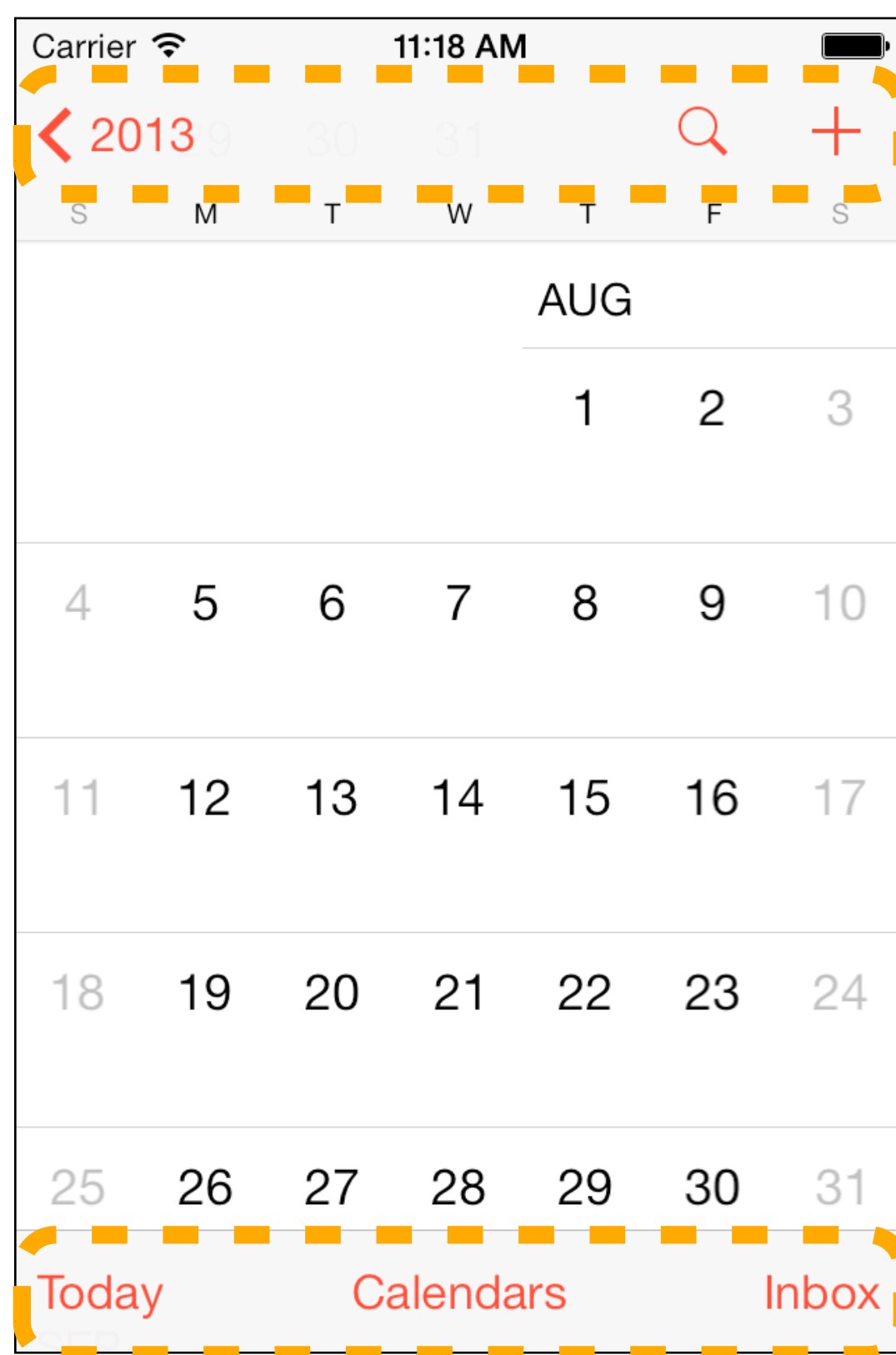
View of the navigation controller

View of a UINavigationController



Custom content of the navigation controller managed by the view controller at the top of the stack

View of a UINavigationController

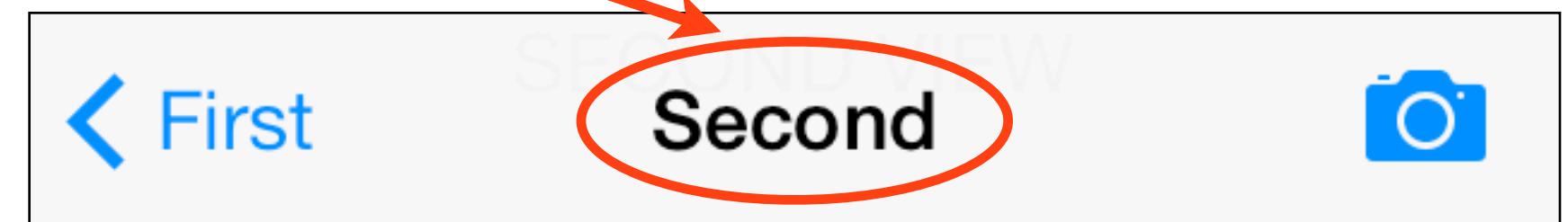


Navigation bar

Navigation toolbar (optional) -
`toolbarItems` property of the
embedded `UIViewController`

UINavigationController

- The `UINavigationController` displays:
 - a title for the currently displayed view controller; it can be set using the `title` property of the embedded `UIViewController`
 - a back button which displays the `title` of the previous `UIViewController` in the navigation stack
 - an array of `UIBarButtonItem` objects (NOT `UIButton`!) accessible with the `UIViewController` property `navigationItem.rightBarButtonItem`



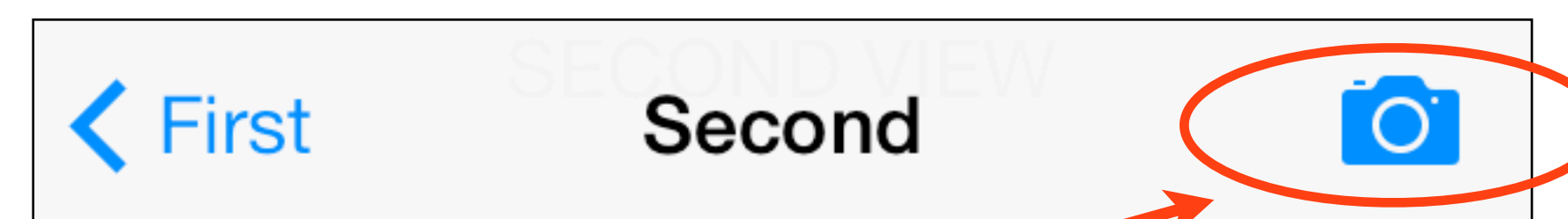
UINavigationController

- The `UINavigationController` displays:
 - a title for the currently displayed view controller; it can be set using the `title` property of the embedded `UIViewController`
 - a back button which displays the `title` of the previous `UIViewController` in the navigation stack
 - an array of `UIBarButtonItem` objects (NOT `UIButton`!) accessible with the `UIViewController` property `navigationItem.rightBarButtonItem`



UINavigationController

- The `UINavigationController` displays:
 - a title for the currently displayed view controller; it can be set using the `title` property of the embedded `UIViewController`
 - a back button which displays the `title` of the previous `UIViewController` in the navigation stack
 - an array of `UIBarButtonItem` objects (NOT `UIButton`!) accessible with the `UIViewController` property `navigationItem.rightBarButtonItem`



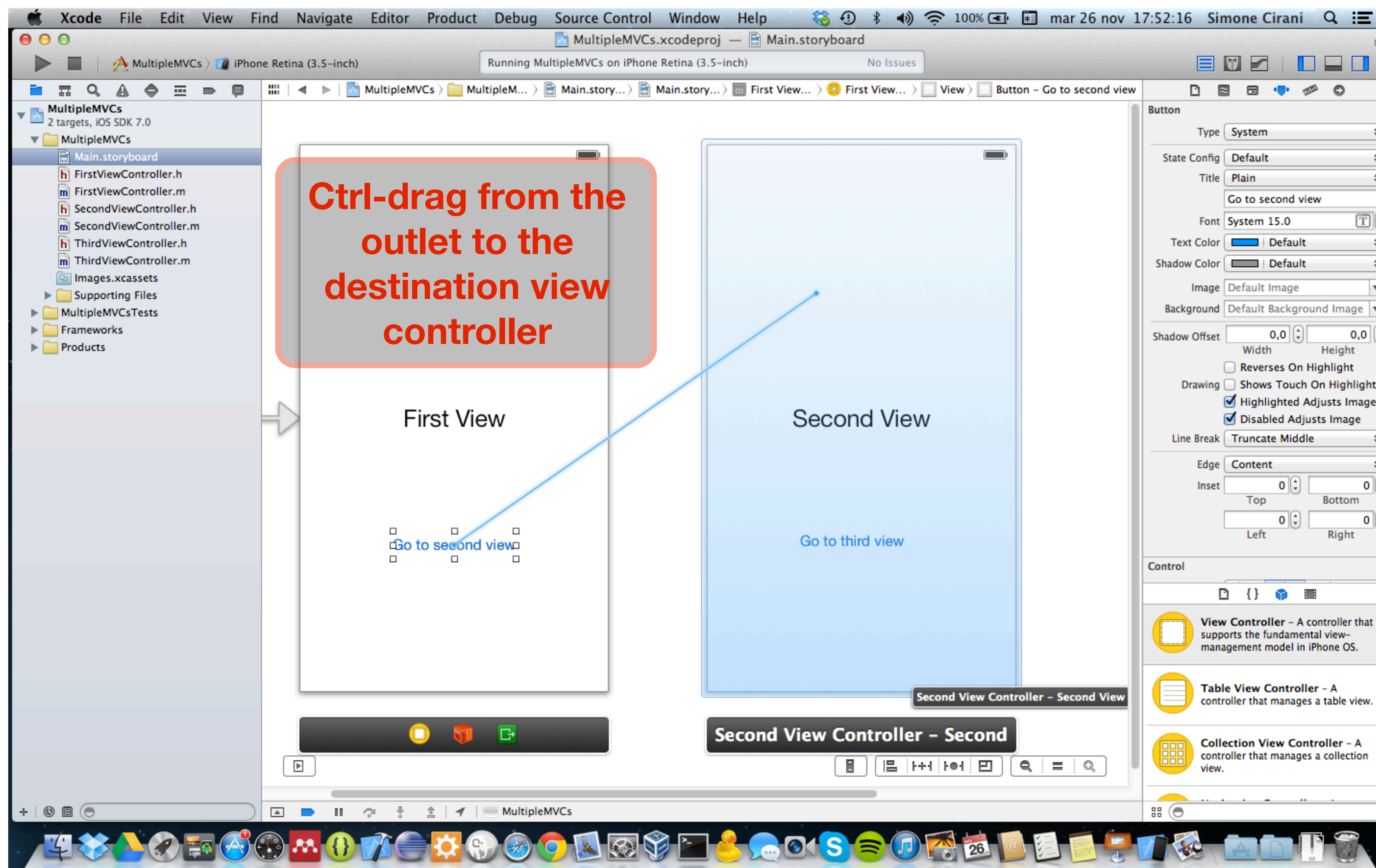
UINavigationController with multiple MVCs

- A `UINavigationController` starts displaying no content
- When the `rootViewController` property of the `UINavigationController` is set, the `UINavigationController` displays the view managed by the view controller
- Segues define the transitions from a view controller to the next one

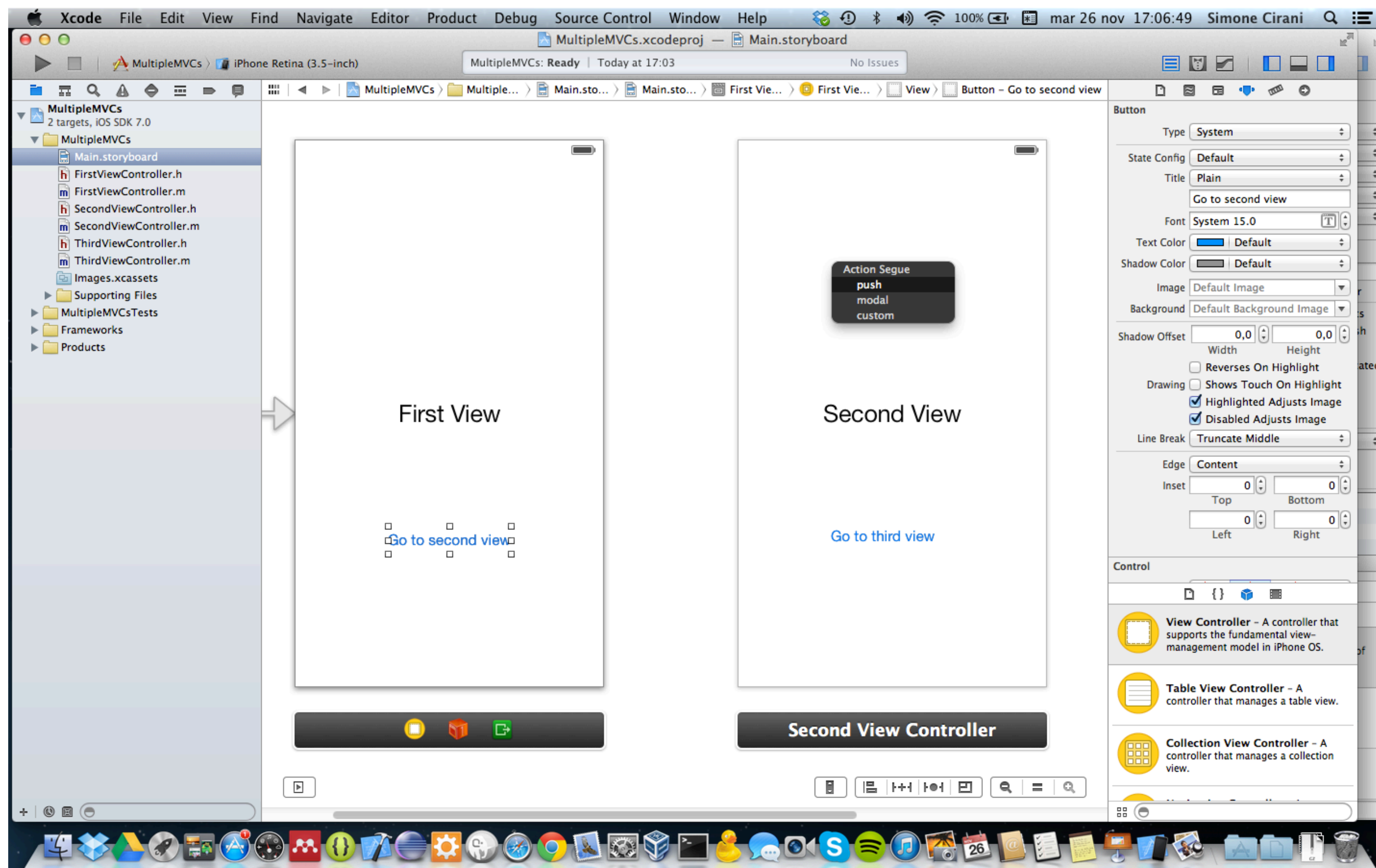
Segues

- Segues are responsible for performing the visual transition between two view controllers
- A transition can be triggered:
 - from an outlet in storyboard (segue)
 - programmatically
- Segues can be of different types:
 - push
 - modal
 - custom

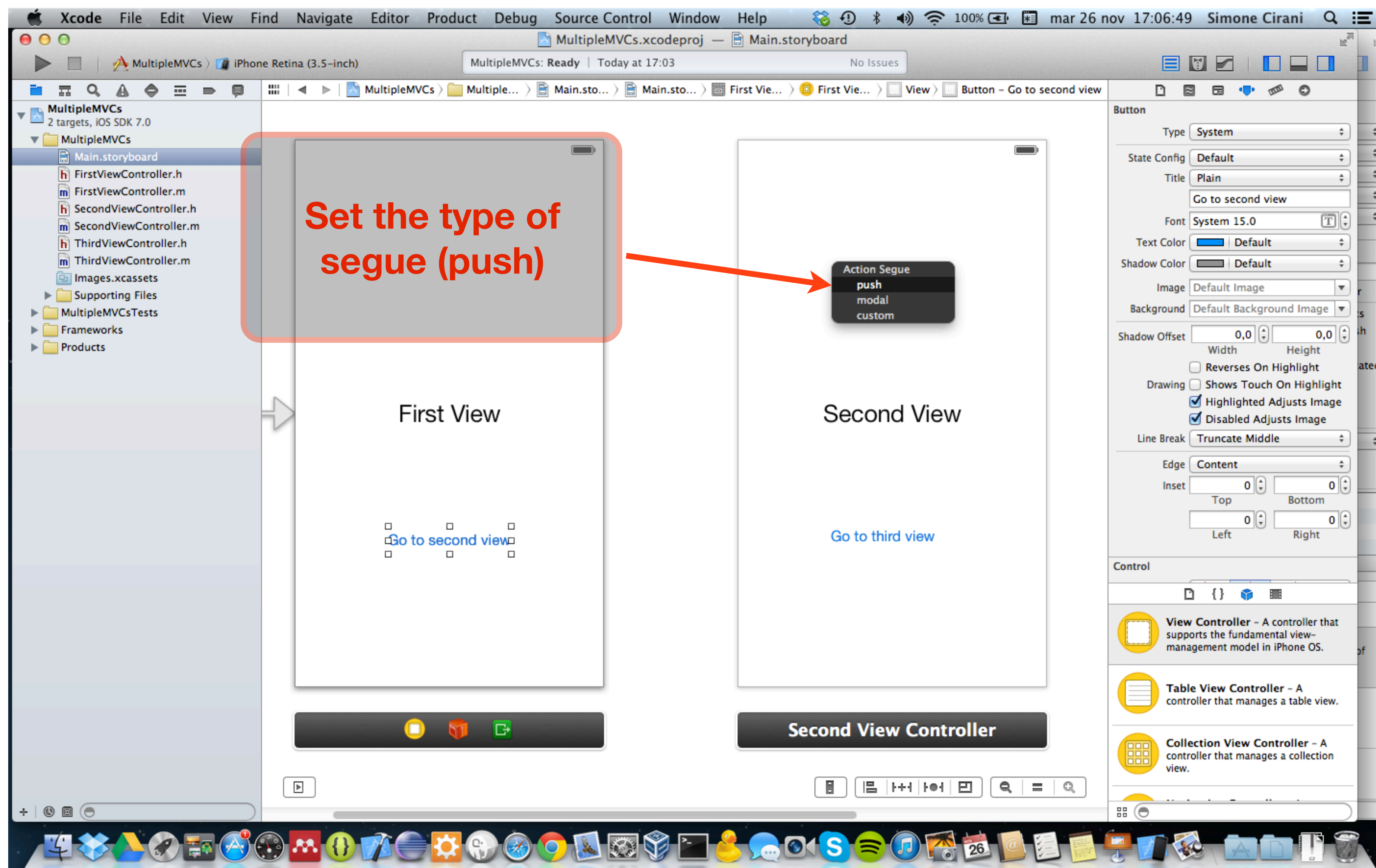
Segues in storyboard



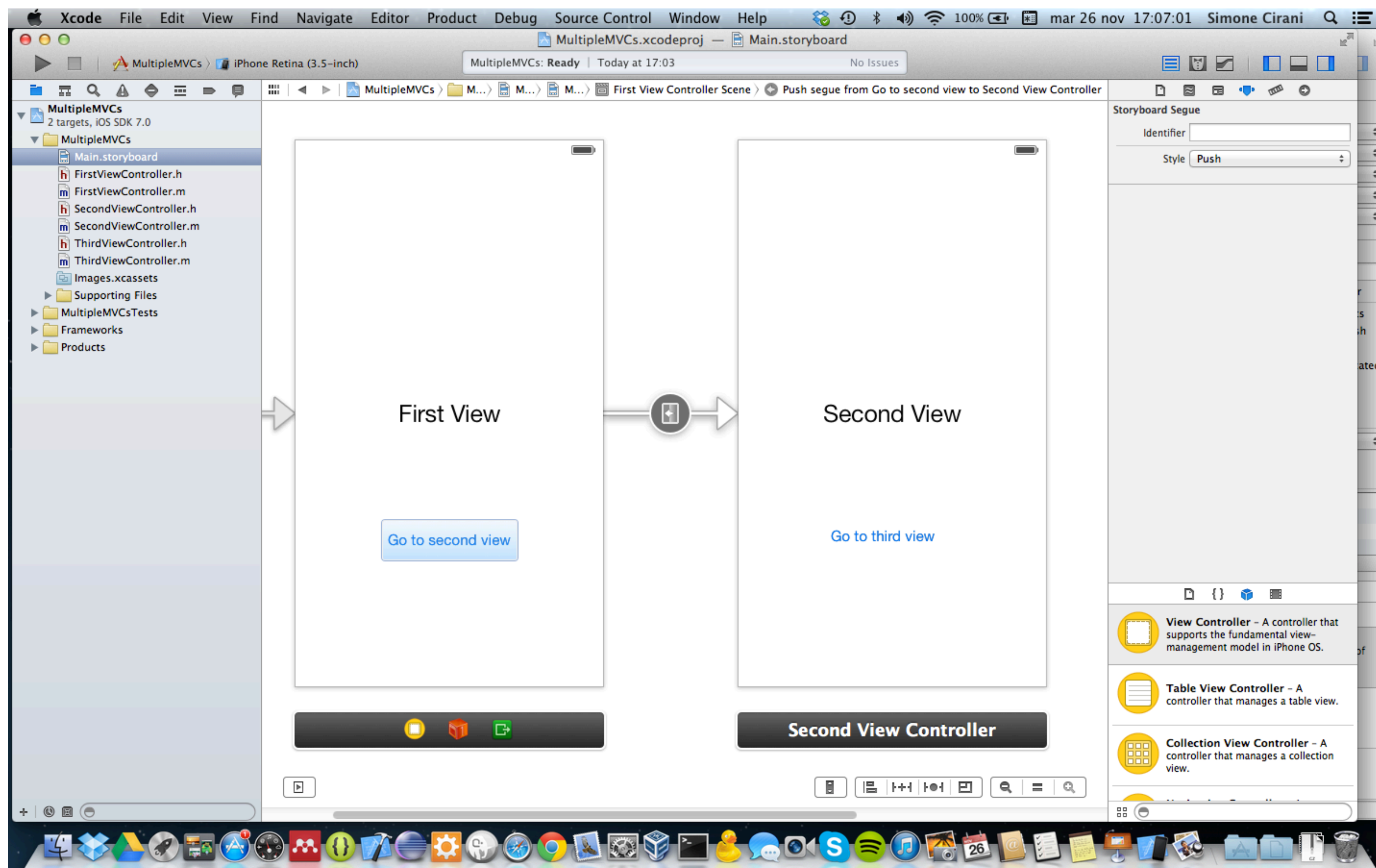
Segues in storyboard



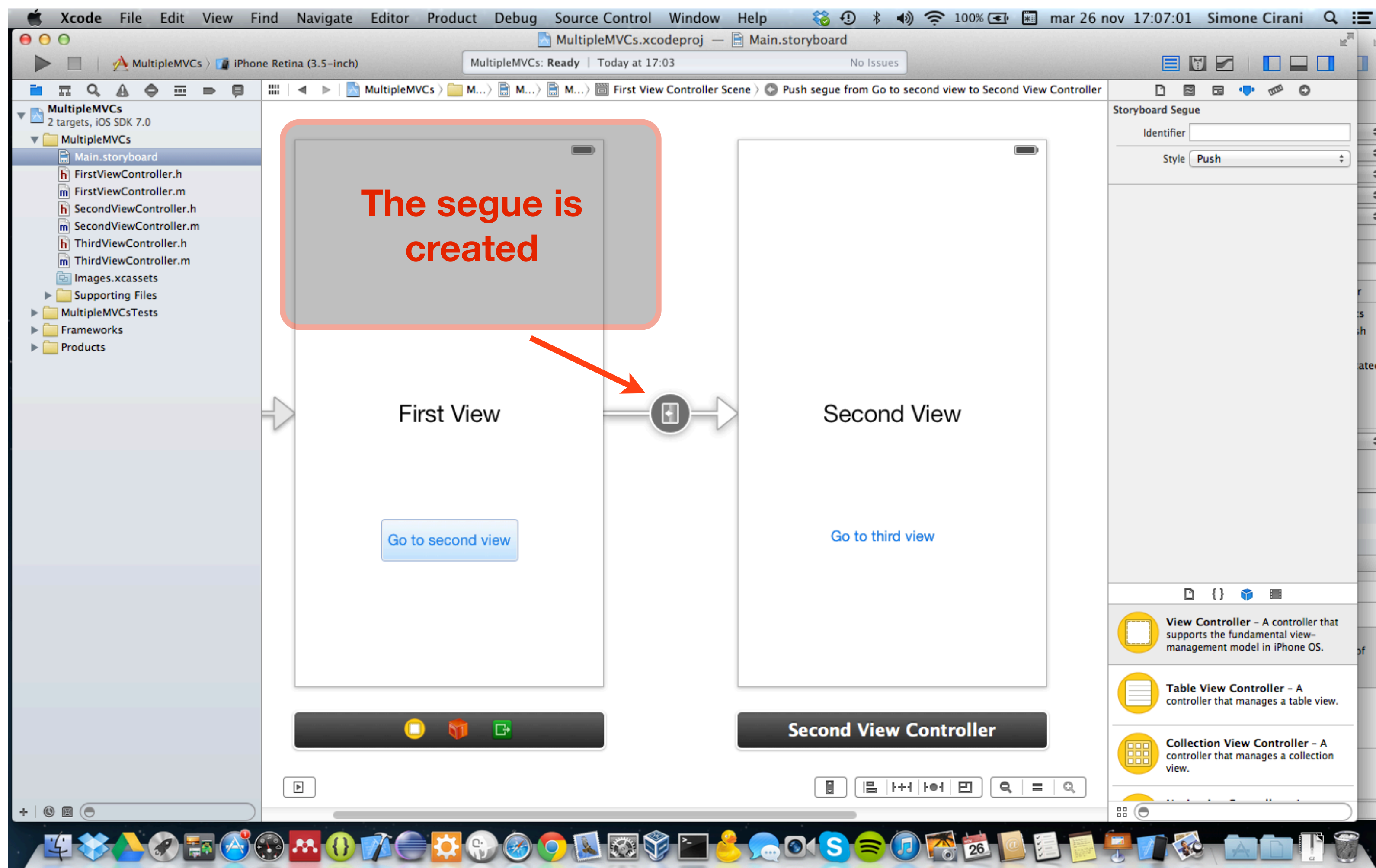
Segues in storyboard



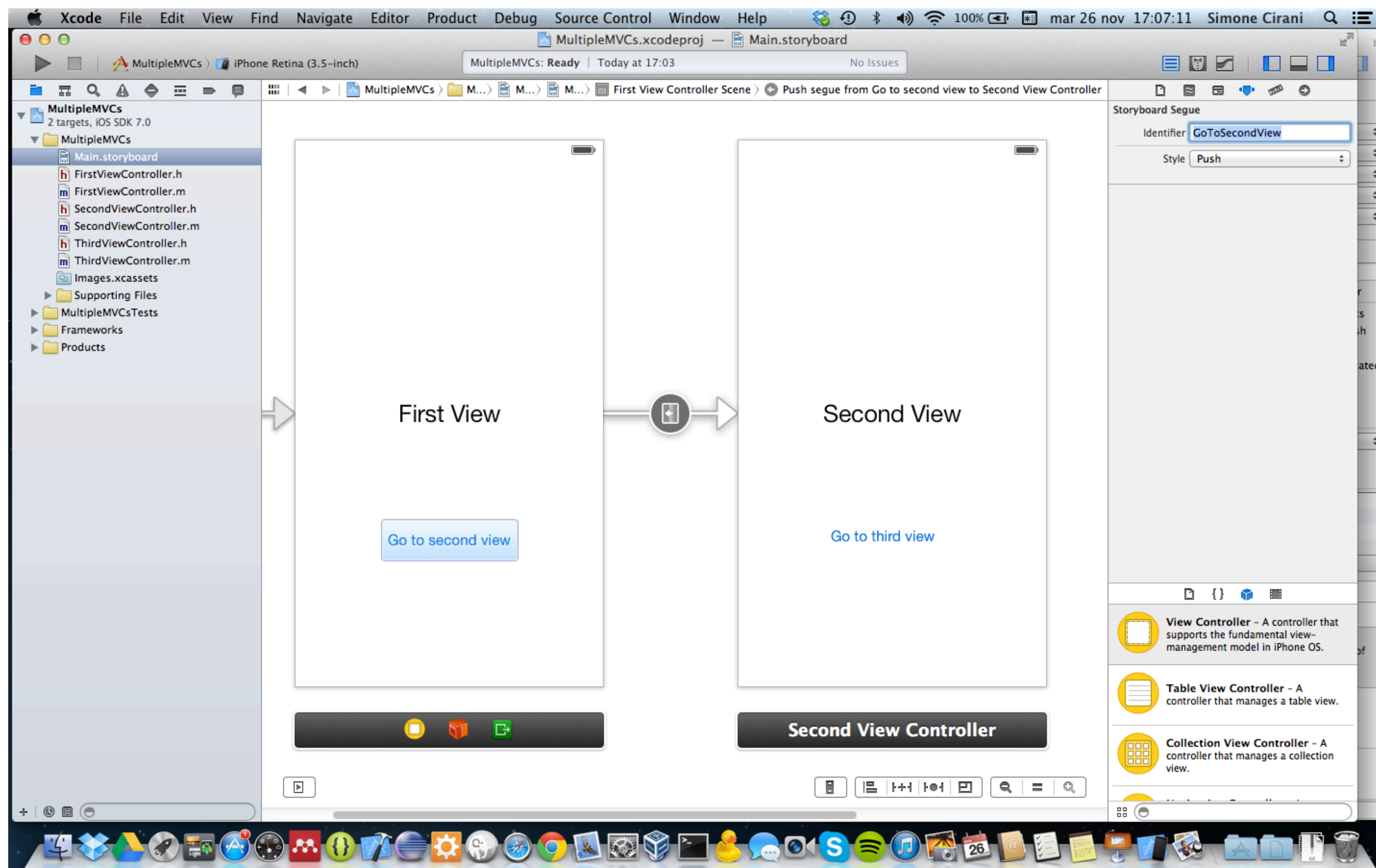
Segues in storyboard



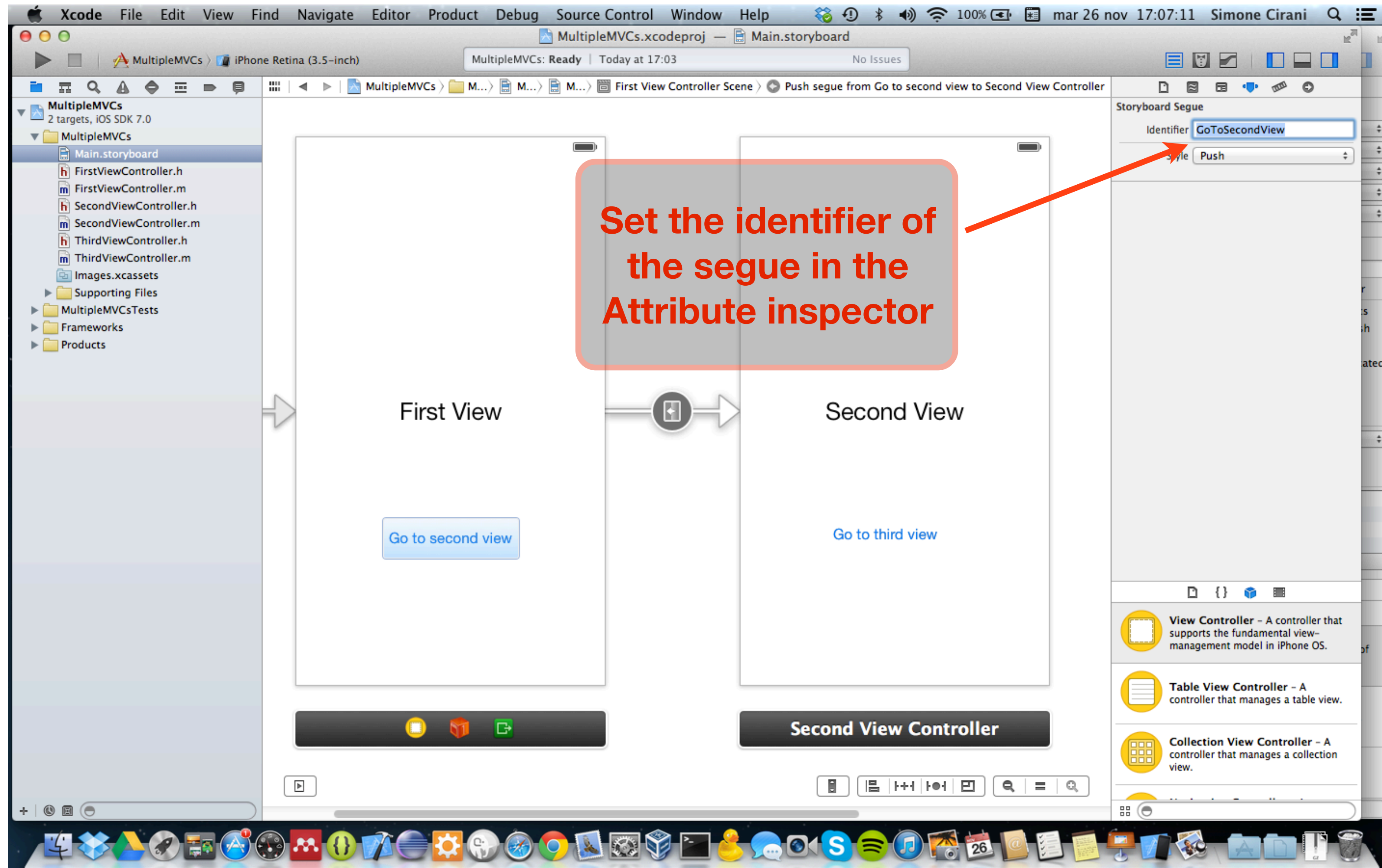
Segues in storyboard



Segues in storyboard



Segues in storyboard



Segues

- A view controller can perform additional setup before performing the segue (it should have the chance to be “prepared”)
- Preparing the view controller to perform the segue should allow to pass important data to the view controller before performing the segue
- Every UIViewController can override the following method to prepare for the segue:
 - `(void)prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender`
- Preparing the view controller to perform the segue should allow to pass important data to the view
- This method is called before the new view controller (`destinationViewController` of the segue) is presented
- The segue that will be executed is passed in as an argument; it can be identified through its `identifier` property (previously set in the Attribute inspector)

Preparing for a segue

```
- (void)prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender {
    if([segue.identifier isEqualToString:@"GoToSecondView"]) {
        if([segue.destinationViewController isKindOfClass:[SecondViewController class]]) {
            SecondViewController *sVC = (SecondViewController *)segue.destinationViewController;
            sVC.data = @"Some data";
        }
    }
}
```

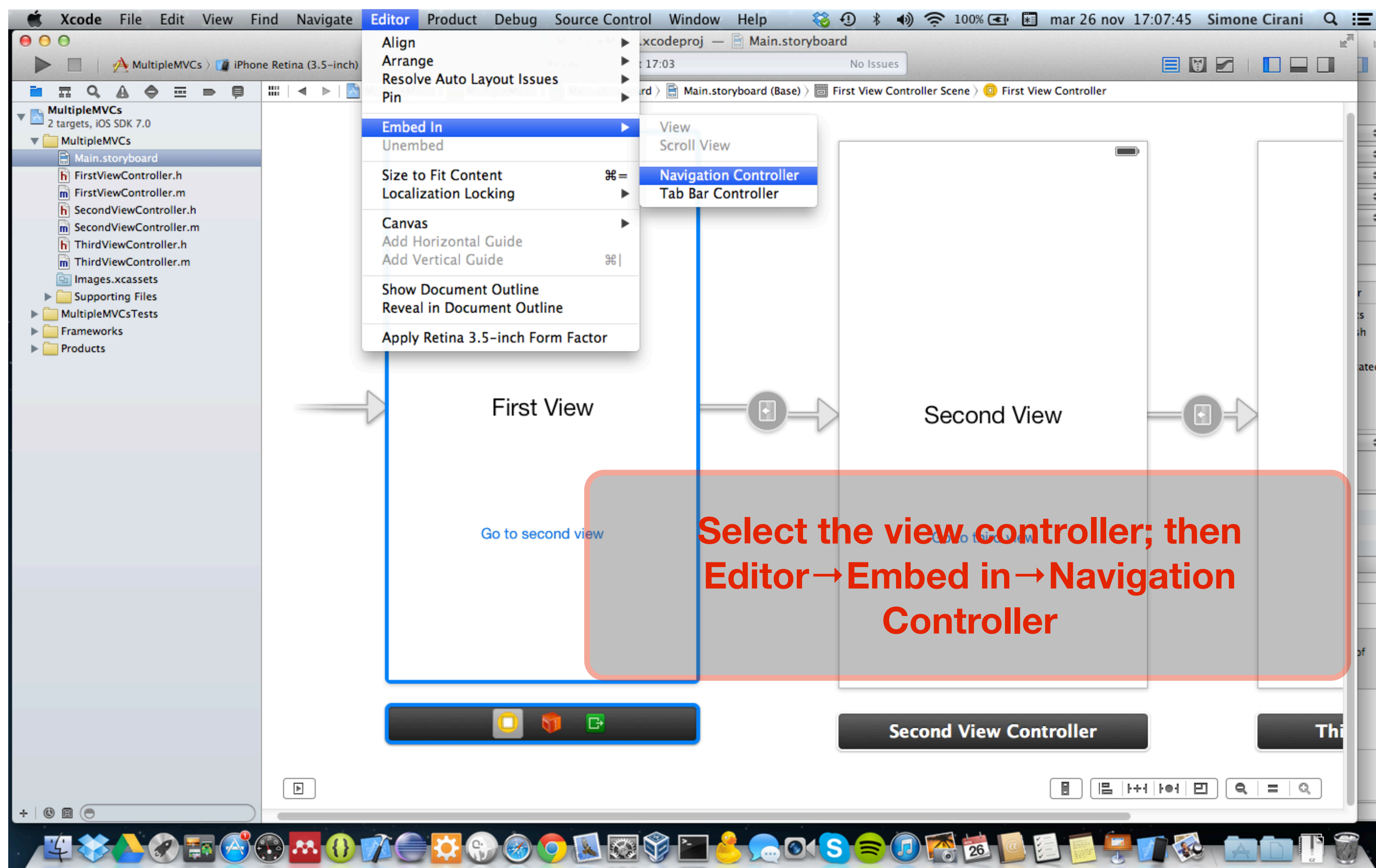
Instantiating view controllers programmatically

- A view controller can be instantiated “by-hand” (not from the storyboard)
- For example, when the type of destination view controller depends on some information that can be resolved only at runtime (e.g. an error screen or a content screen)

```
NSString *vc = @"SecondViewController";  
UIViewController *controller = [self.storyboard instantiateViewControllerWithIdentifier:vc];
```

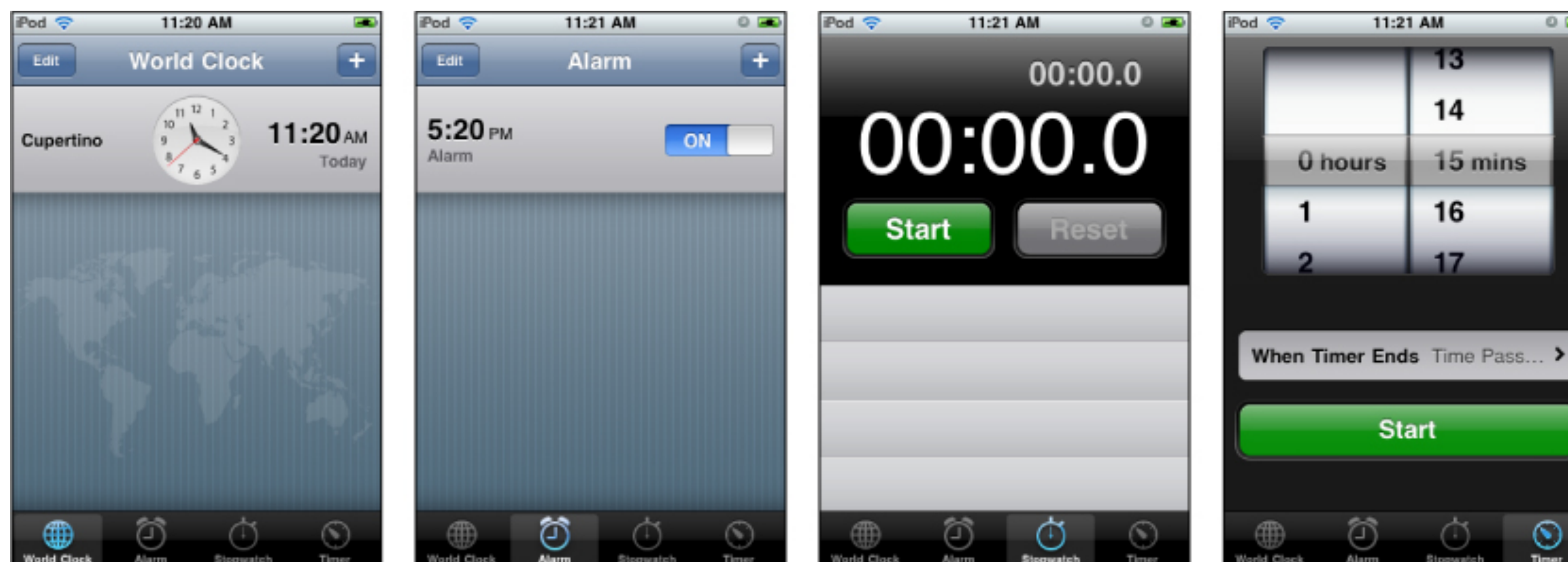
- The view controller is then pushed to the navigation stack with the `pushViewController:animated:` method

Embedding view controllers in a UINavigationController



UITabBarController

- `UITabBarController` is a class that implements a specialized view controller that manages a radio-style selection interface
- Typical usage: **multiple sections** (e.g. Clock app: Clock, Alarm, Stopwatch, Timer)
- Tabs are displayed at the bottom of the window (tab bar) for selecting between the different modes and for displaying the views for that mode
- Example of a navigation controller-based app: Clock app

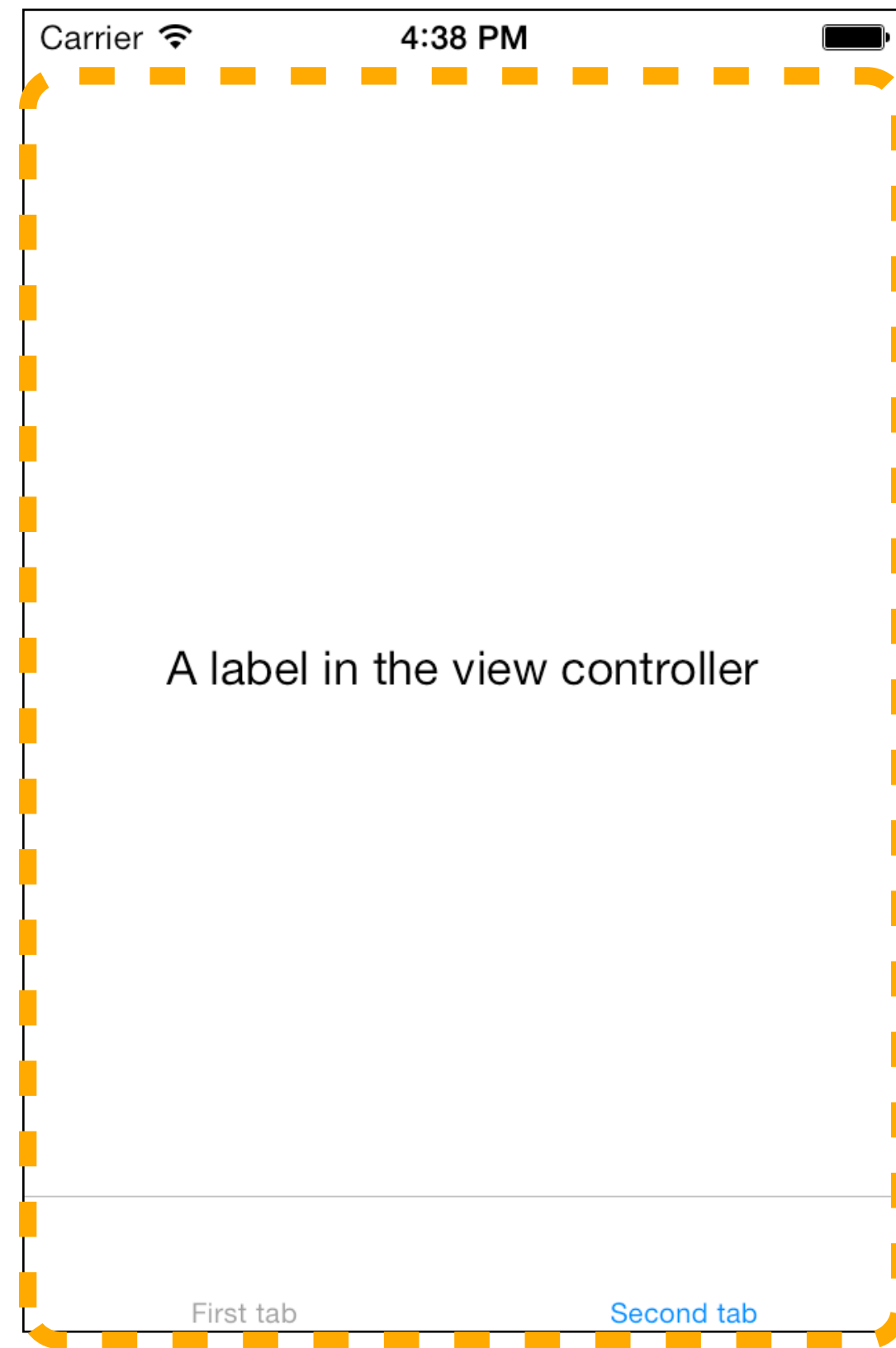


UITabBarController

- Each tab of a tab bar controller interface is associated with a custom view controller
- When the user selects a specific tab, the tab bar controller displays the root view of the corresponding view controller, replacing any previous views
- View controllers are assigned to the tab bar controller by setting the `viewController`s property of the `UITabBarController` (`NSArray` of `UIViewController`s); order is maintained

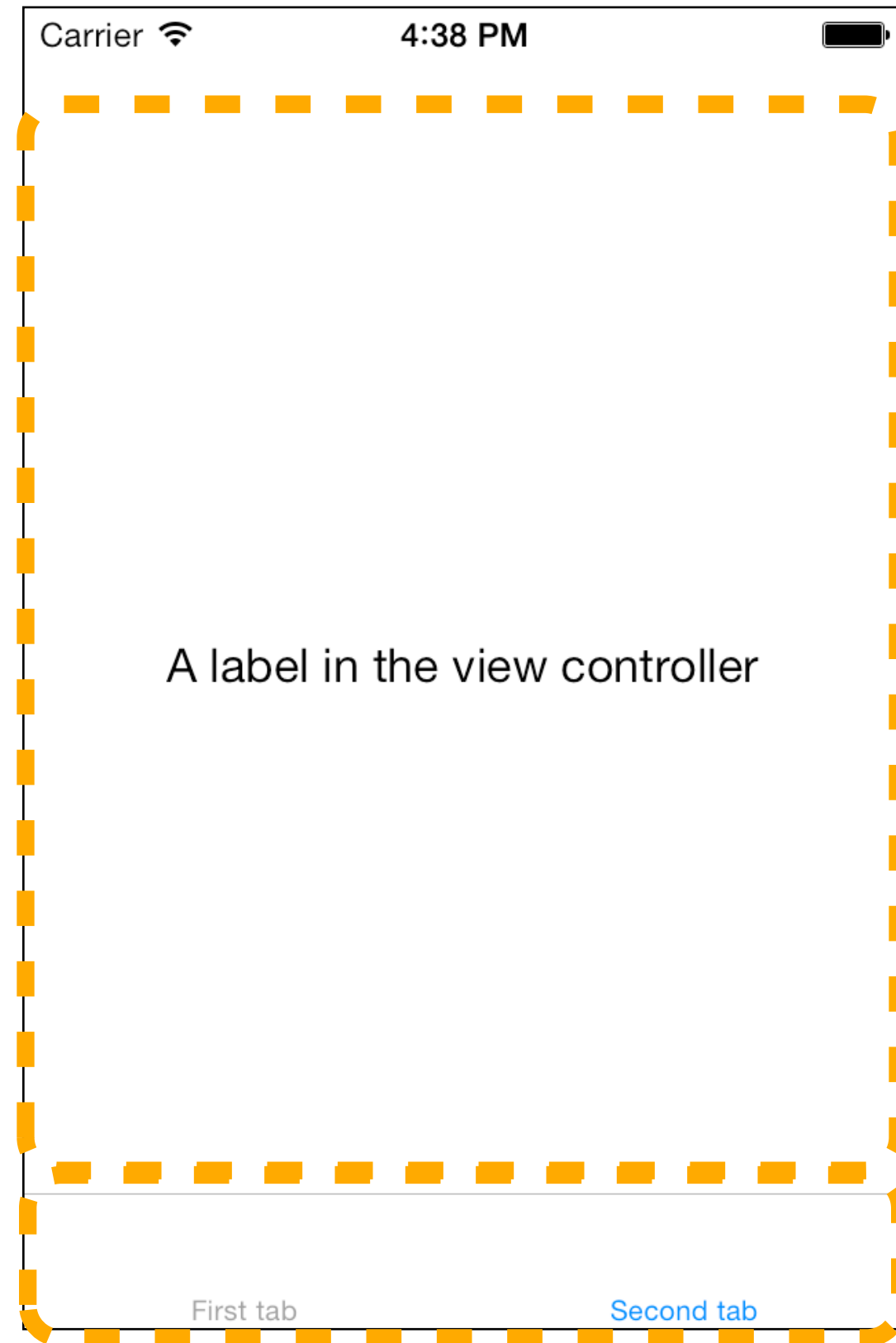


Views of a UITabBarController



View of the tab bar controller

Views of a UITabBarController

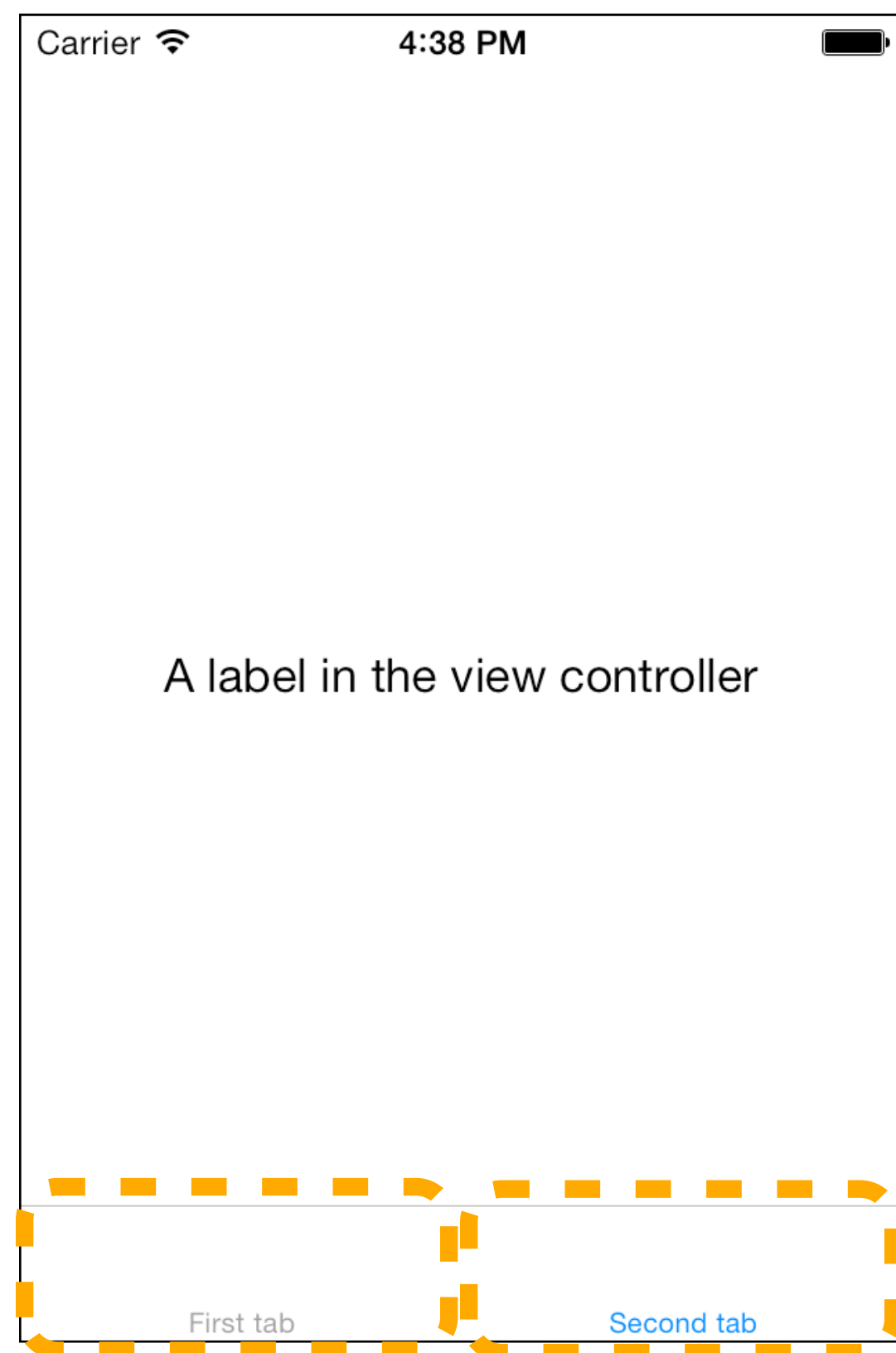


Custom content



Tab bar (UITabBar)

Views of a UITabBarController



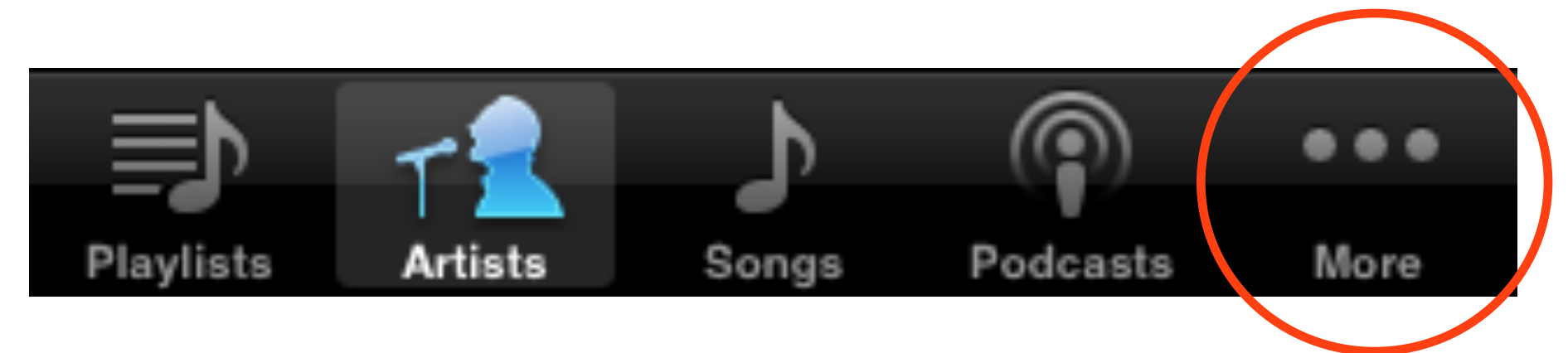
→ Tab bar items (`UITabBarItem`): icon + title

Views of a UITabBarController

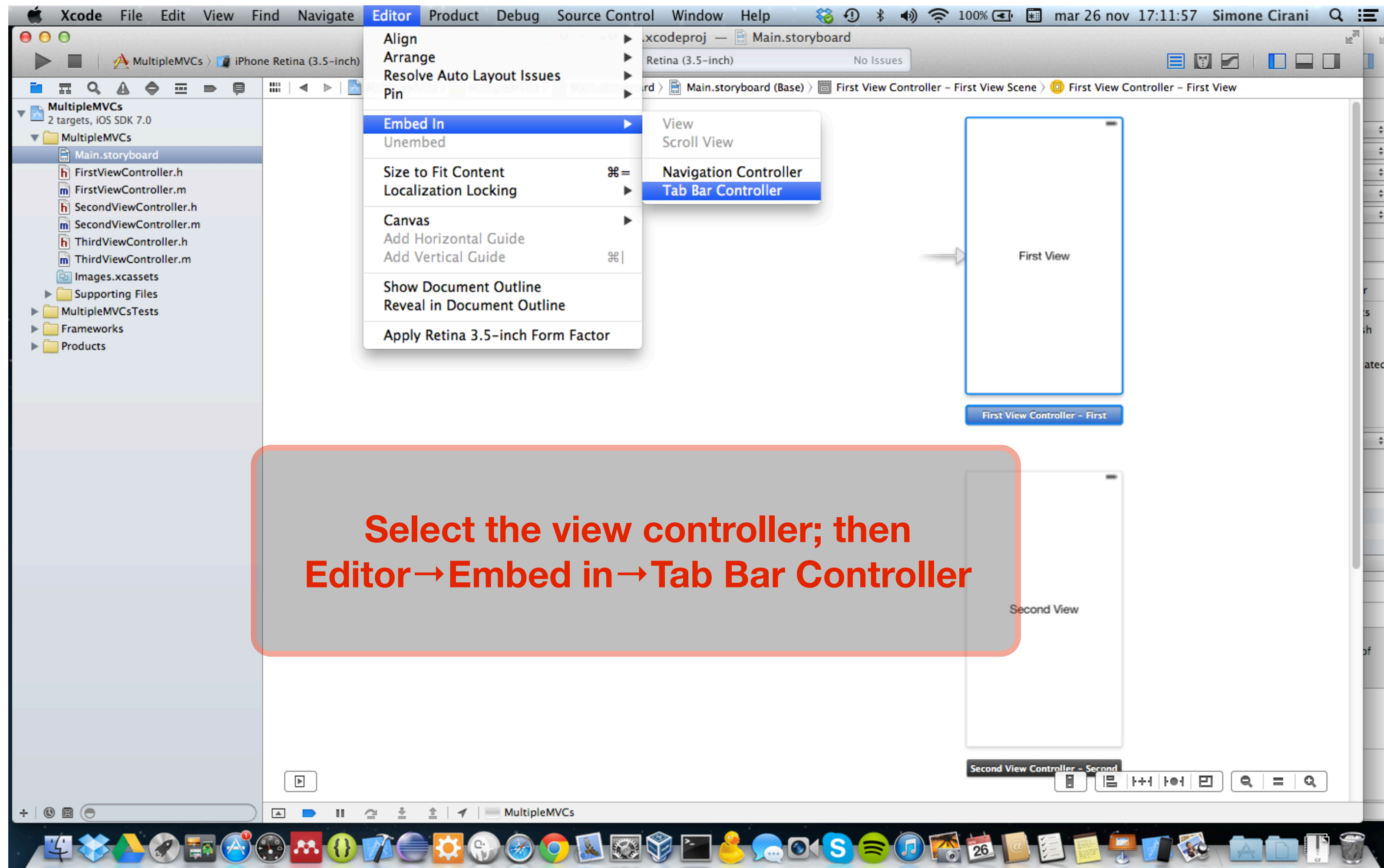
- The embedded view controller defines the appearance of its corresponding tab bar item
- The title of the tab bar item for the currently displayed view controller can be set using the `title` property of the embedded `UIViewController` (if there is no instance of `UITabBarItem` associated to the tab bar the title is displayed by default with no image)
- The appearance of the tab bar item (title and icon) can be configured in storyboard
- To associate a custom tab bar item programmatically:
 1. create an instance of `UITabBarItem` to configure its title and image
 2. set the `tabBarItem` property of the embedded view controller with the created item

More navigation controller

- The tab bar offers limited space to render tab items
- If 6 or more custom view controllers are added to a tab bar controller, the tab bar controller displays only the first four items plus the standard “More” item on the tab bar
- When the “More” tab is selected a view with all the remaining tabs is shown (automatically)



Embedding view controllers in a UITabBarController



Select the view controller; then
Editor -> Embed in -> Tab Bar Controller

Adding view controllers to a UITabBarController

The screenshot shows the Xcode interface for a project named 'MultipleMVCs'. The storyboard is titled 'Main.storyboard' and is currently displaying the 'Tab Bar Controller Scene'. The storyboard contains three view controllers: a 'Tab Bar Controller' (highlighted with a blue border), a 'First View Controller - First' (containing a 'First View' with a 'Go to second view' button), and a 'Second View Controller - Second' (containing a 'Second View' with a 'Go to third view' button). A blue arrow points from the 'Outlet' property on the 'Tab Bar Controller' to the 'First View Controller - First' view controller. A red box with white text is overlaid on the storyboard, containing the instruction: **Ctrl-drag from the outlet to the destination view controller**. The right-hand side of the Xcode window shows the 'View Controller' inspector, which is currently set to 'View Controller'. The 'Initial Scene' checkbox is checked, and the 'Transition Style' is set to 'Cover Vertical'. The bottom of the Xcode window shows the macOS dock with various application icons.

Adding view controllers to a UITabBarController

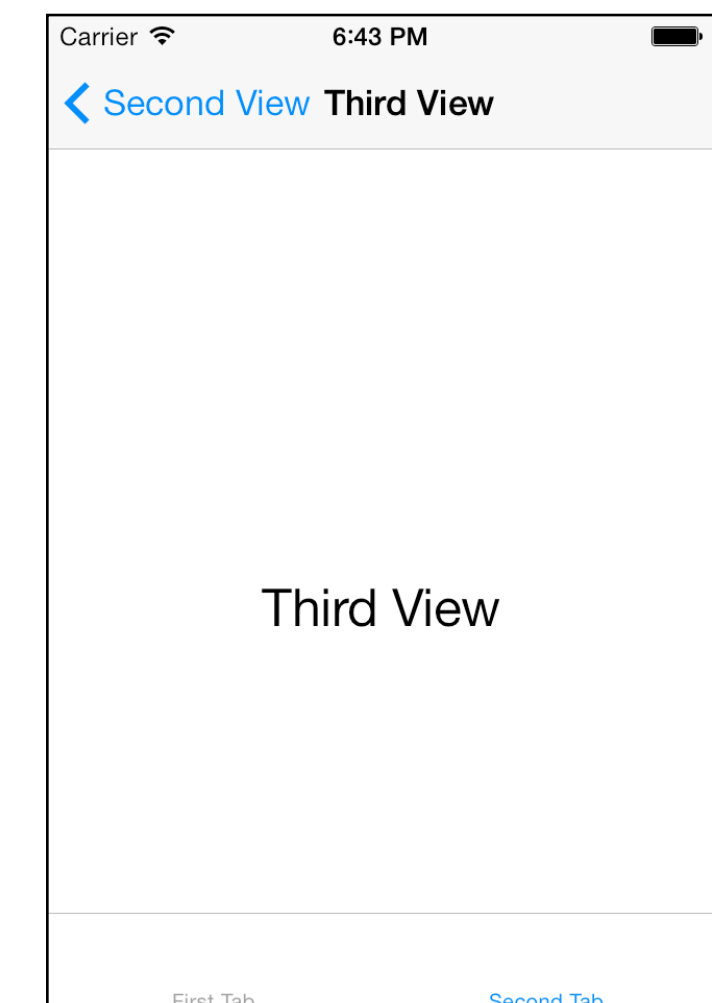
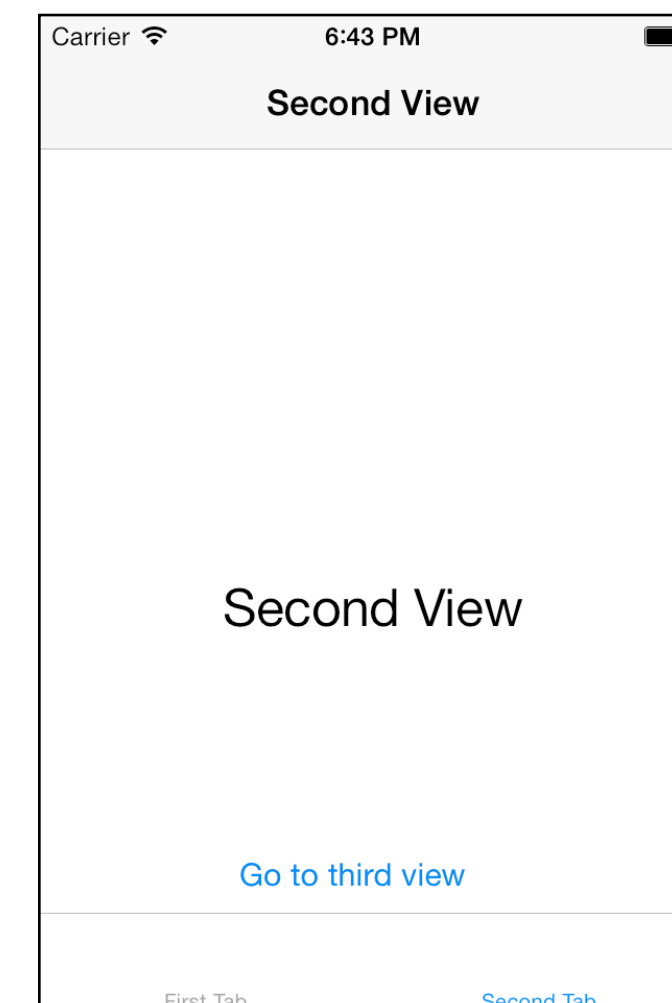
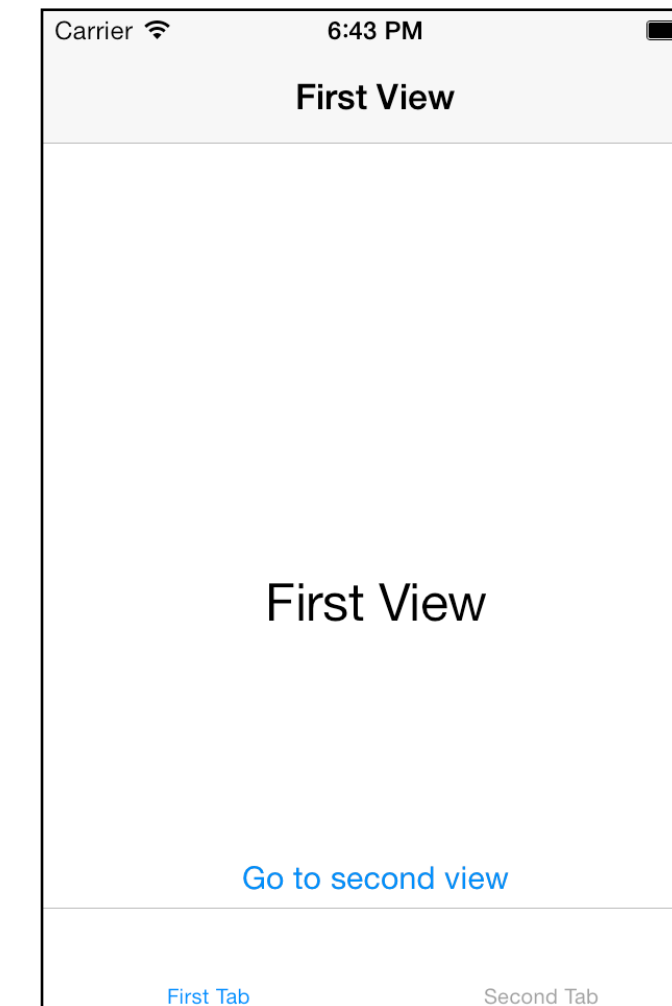
The screenshot shows the Xcode interface for editing a storyboard. On the left, the project navigator shows the file structure for 'MultipleMVCs'. The main canvas displays a storyboard with two view controllers: 'Tab Bar Controller' and 'First View Controller'. A segue arrow connects them. A red box highlights the segue type menu, which lists options: Manual Segue, push, modal, custom, Relationship Segue, and view controllers. A text box over the red box reads: 'Set the type of segue (Relationship: view controllers)'. The right-hand side of the interface shows the 'View Controller' inspector with various settings like 'Initial Scene', 'Layout', 'Extend Edges', and 'Transition Style'. The status bar at the top indicates the device is 'iPhone Retina (3.5-inch)' and the app is running.

Adding view controllers to a UITabBarController

The screenshot shows the Xcode interface for a project named 'MultipleMVCs'. The storyboard, 'Main.storyboard', is open and displays a 'Tab Bar Controller' scene. The storyboard contains three view controllers: 'Tab Bar Controller', 'First View', and 'Second View'. A segue is created between the 'Tab Bar Controller' and the 'First View', and another segue is created between the 'Tab Bar Controller' and the 'Second View'. A red box with the text 'The segue is created' is overlaid on the storyboard, pointing to the segue connection between the 'Tab Bar Controller' and the 'First View'. The right-hand pane shows the 'View Controller' settings for the selected view controller, including options for 'Initial Scene', 'Layout', 'Extend Edges', 'Transition Style', and 'Presentation'. The 'Initial Scene' checkbox is checked, and the 'Transition Style' is set to 'Cover Vertical'. The 'View Controller' settings pane also includes a description of the 'View Controller' class: 'A controller that supports the fundamental view-management model in iPhone OS.'

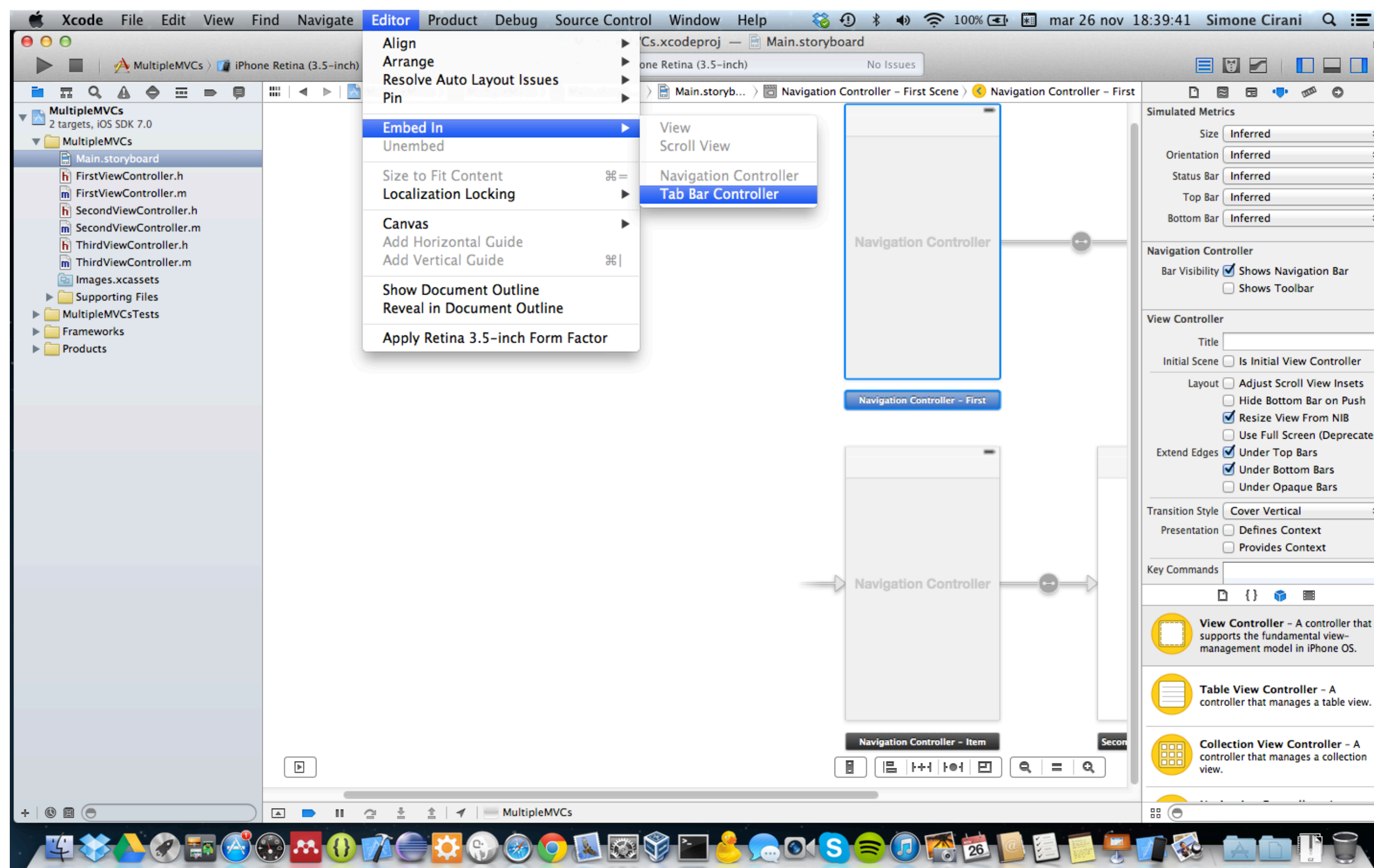
Combining tab bar controllers and navigation controllers

- It is possible to combine tab bar controllers and navigation controllers
- A navigation controller can be embedded inside a tab bar controller, if one or more sections of the app require a drill-down navigation
 - for example, the Music app has several sections (Artists, Albums, ...) and by getting into one of them we get into more detail
- Embedding a tab bar controller inside a navigation controller doesn't make much sense... probably redesigning the app would be better (indeed XCode does not allow you to do this)



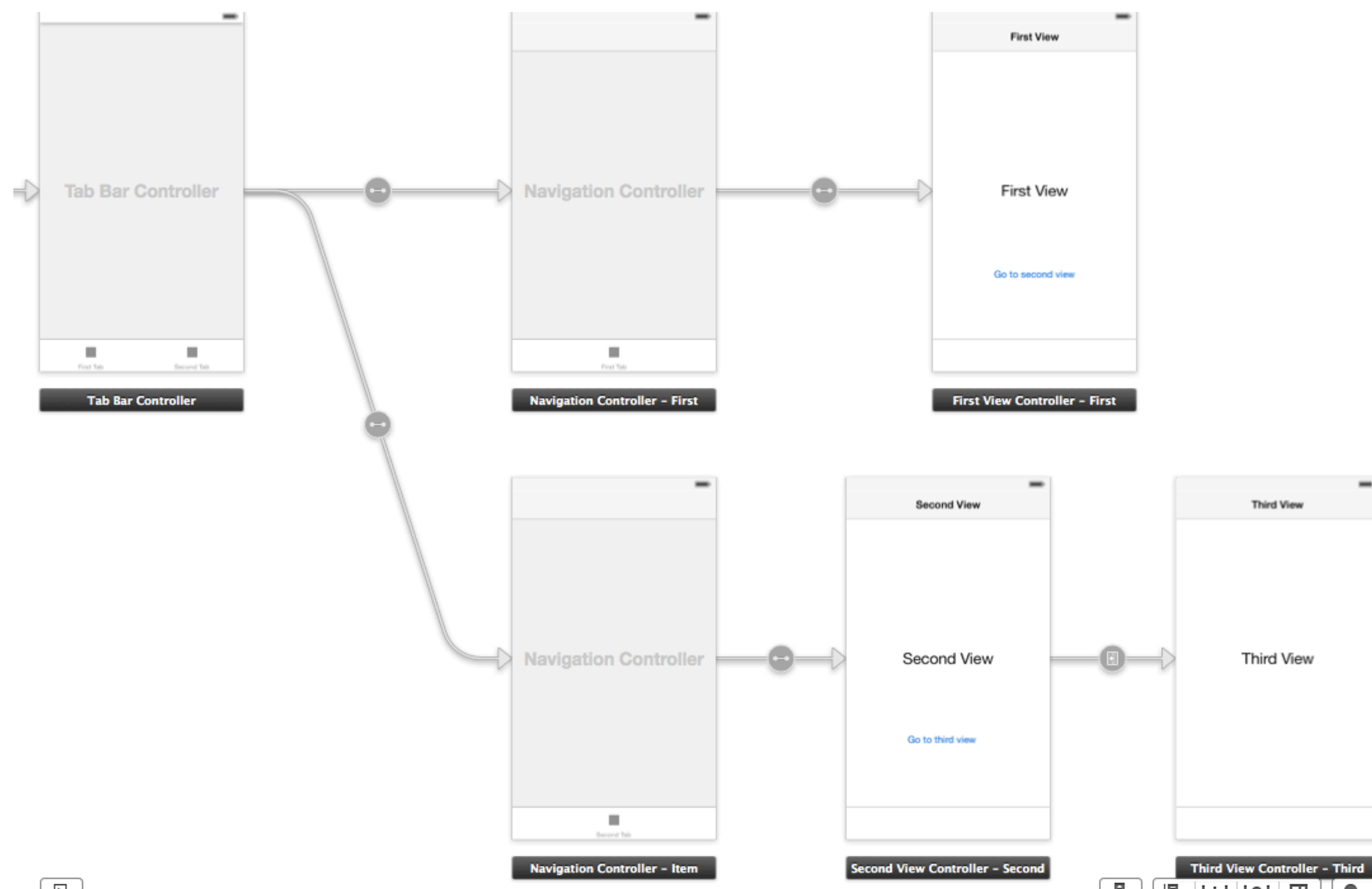
Combining tab bar controllers and navigation controllers

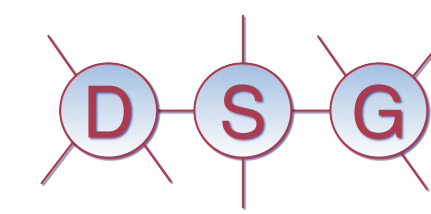
- To embed a UINavigationController inside a UITabBarController in storyboard, select the navigation controller, then go to Editor → Embed in → Tab Bar Controller



Combining tab bar controllers and navigation controllers

- It is possible to have complex storyboards for advanced user experience





Mobile Application Development

Lecture 16

Controllers of View Controllers