MSSE 652: iOS Enterprise Software Development

Topic 1: Xcode 5 Storyboards

Agenda

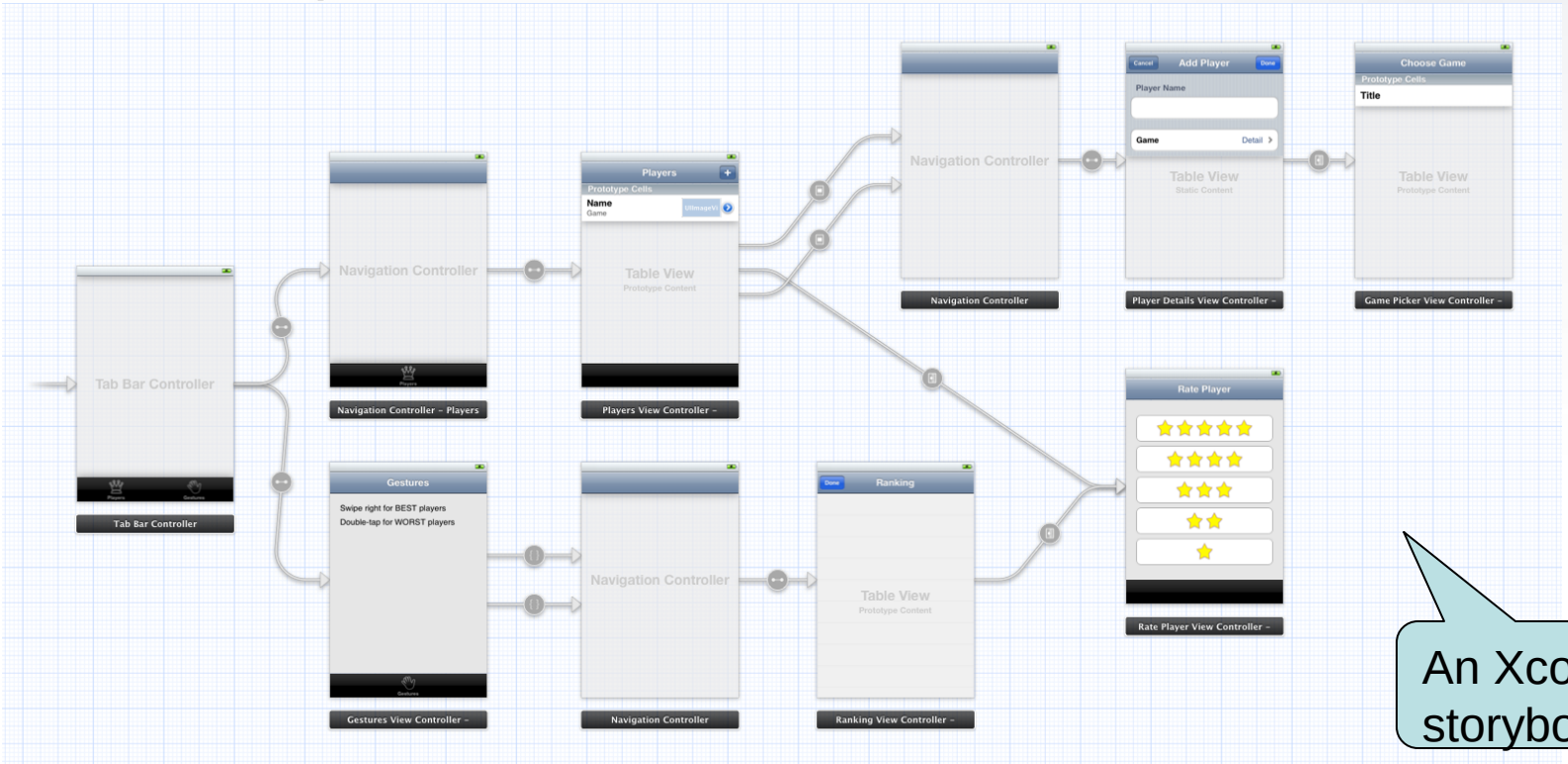
- Resources
- Introduction
- Xcode 5 Storyboards
  - Creating scenes
  - Creating view controllers
  - Creating segues
  - Transition modes / presentation modes
  - Transitions: advancing, unwinding
  - Tab bars
  - Navigation bars

Resources

- Online resources
  - https://developer.apple.com/library/ios/documentation/general/conceptual/Devpedia-CocoaApp/Storyboard.html
  - https://developer.apple.com/library/ios/featuredarticles/ViewControllerPGforiPhoneOS/UsingViewControllersinYourApplication/UsingViewControllersinYourApplication.html
  - https://developer.apple.com/library/ios/documentation/iPhone/Conceptual/SecondiOSAppTutorial/GettingStarted/GettingStarted.html
  - https://developer.apple.com/library/ios/DOCUMENTATION/UserExperience/Conceptual/TableView_iPhone/TableView_iPhone.pdf

Introduction

- iOS Storyboards – a graphical representation of an application's user interface (i.e., the screens)
  - for example



Introduction (cont'd)

- Benefits of iOS Storyboards; they …
  - provide a overarching graphical view of the entire UI of an app
    - by identifying all the screens & their navigational relationships
  - reduce the amount of time required to build the UI
  - reduce the amount of code required to integrate the various screens
- Storyboard terminology …

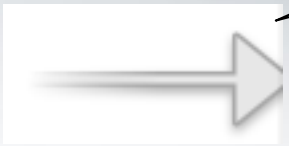
Introduction (cont'd)

- View
  - a UI screen, visible to the user
- View Controller
  - a class that handles the events associated with a view
    - e.g., multi-touch events
- Scene
  - a container that hosts a view and a view controller
- Segue
  - a relationship between two scenes that facilitates the transition from one scene to another

Introduction (cont'd)

- iOS Storyboard
  - a graphical representation of various …
    - scenes, consisting of …
      - views
      - view controllers
    - segues
      - that facilitate transitions between scenes
  - stored in a single file, in an XML format
    - with suffix *.storyboard
      - e.g., the default "main" storyboard is typically named …
        - » Main.storyboard

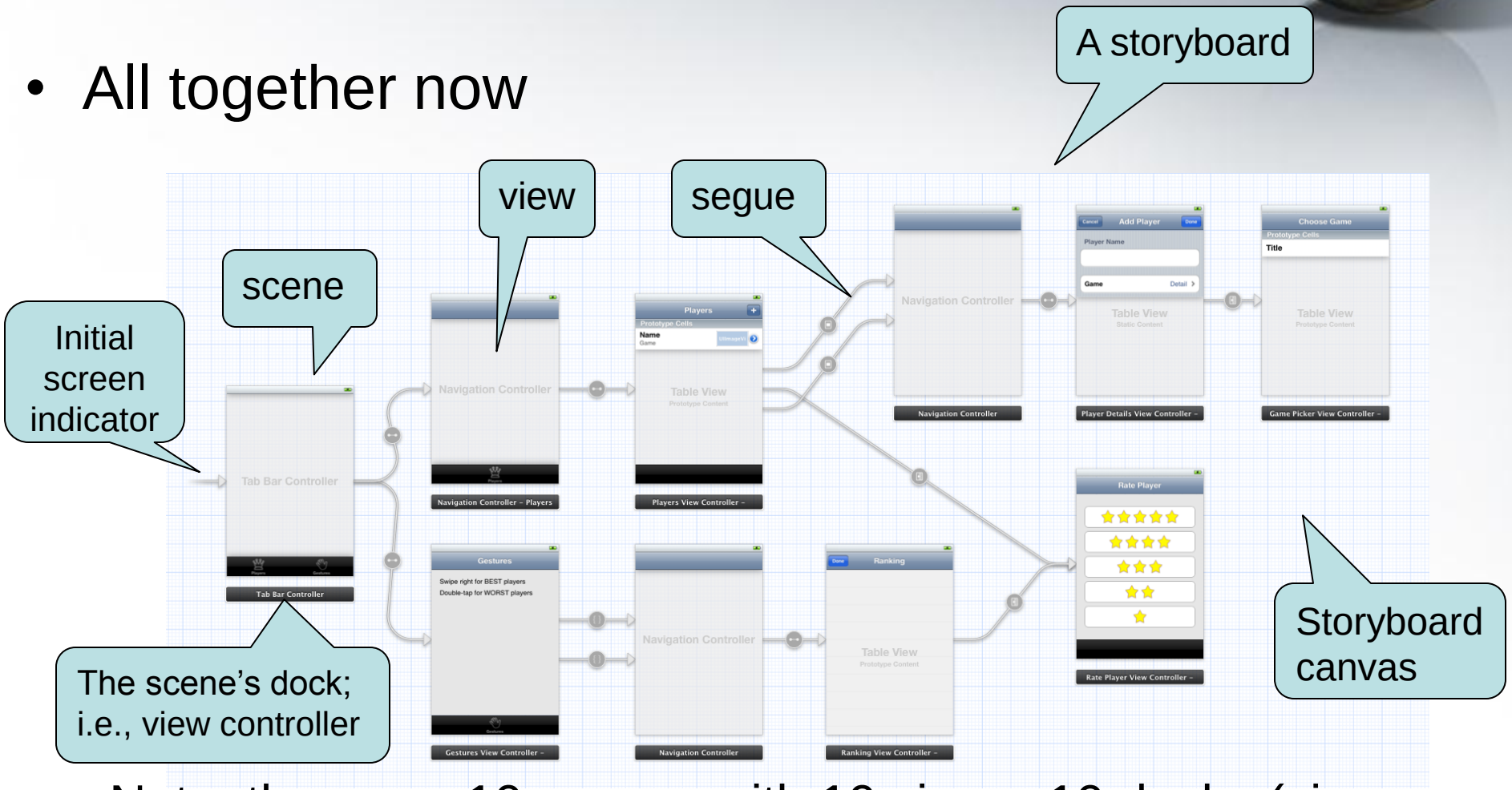
Introduction (cont'd)



- Initial Screen Indicator
  - an arrow that indicates the storyboard's initial scene
    - i.e., it identifies the initial screen displayed to the user
- Storyboard canvas
  - the visual background of the storyboard on which you create scenes and segues
- The Dock
  - a black bar at the bottom of each scene that represents the view controller

Introduction (cont'd)

- All together now



Note: there are 10 scenes with 10 views, 10 docks (view controllers) and 12 segues in the above storyboard

Introduction (cont'd)

- One more thing …
  - Xcode also provides a hierarchical view of the storyboard components in what's known as the Document Outline

- The Document Outline lists …
  - each scene
    - and directly under each scene is the scene's view controller
      - which contains the scene's view
        - » which contains all the UI components that make up the view

More on this later

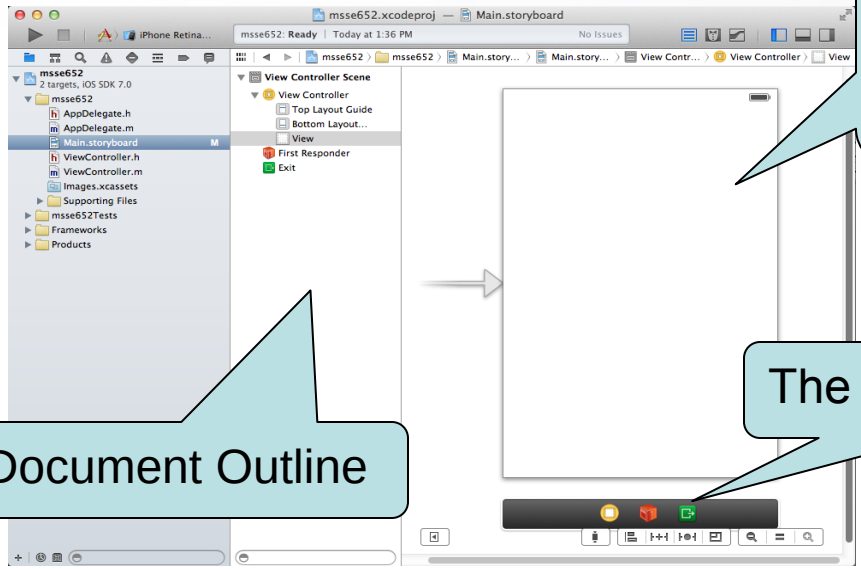
Xcode 5 Storyboard Techniques

- Creating scenes
- Creating view controllers
- Creating segues
- Transition modes / presentation modes
- Transitions: advancing, unwinding
- Tab bars
- Navigation bars

Let’s get started …

Creating an Xcode project

- The techniques identified on the previous page will be explained in the context of a new Xcode 5 project,
  - by using the Single View Application template
    - with project name: msse652



Project Navigator with Main.storyboard selected

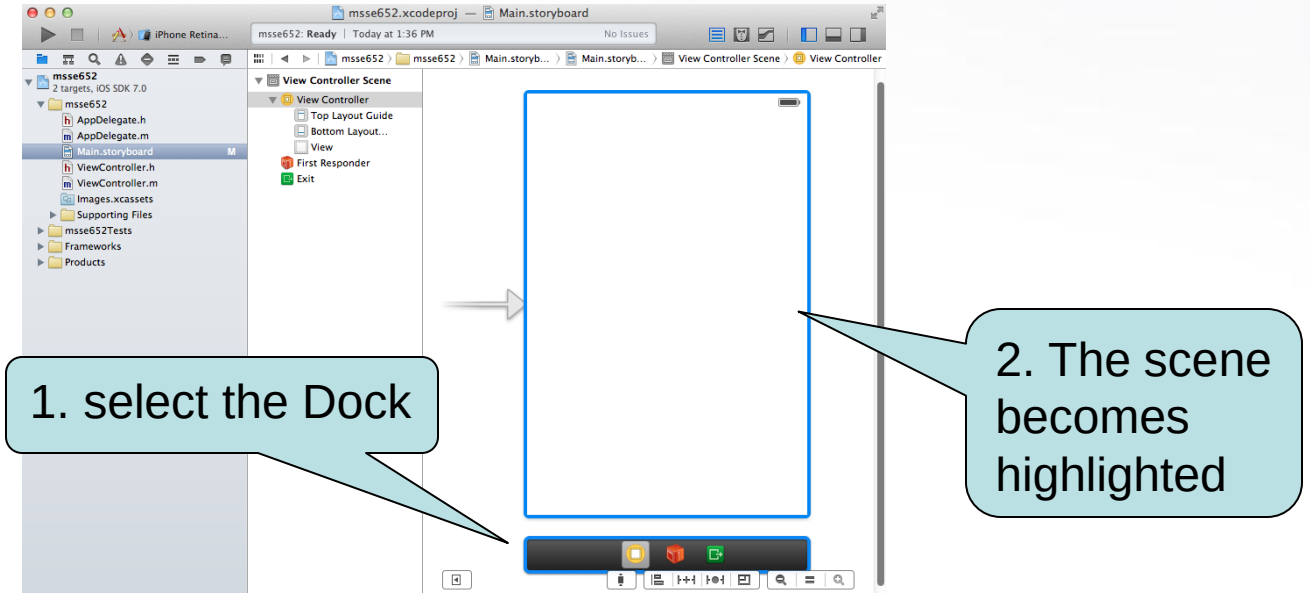
A storyboard containing a single scene

The Document Outline

The scene's Dock

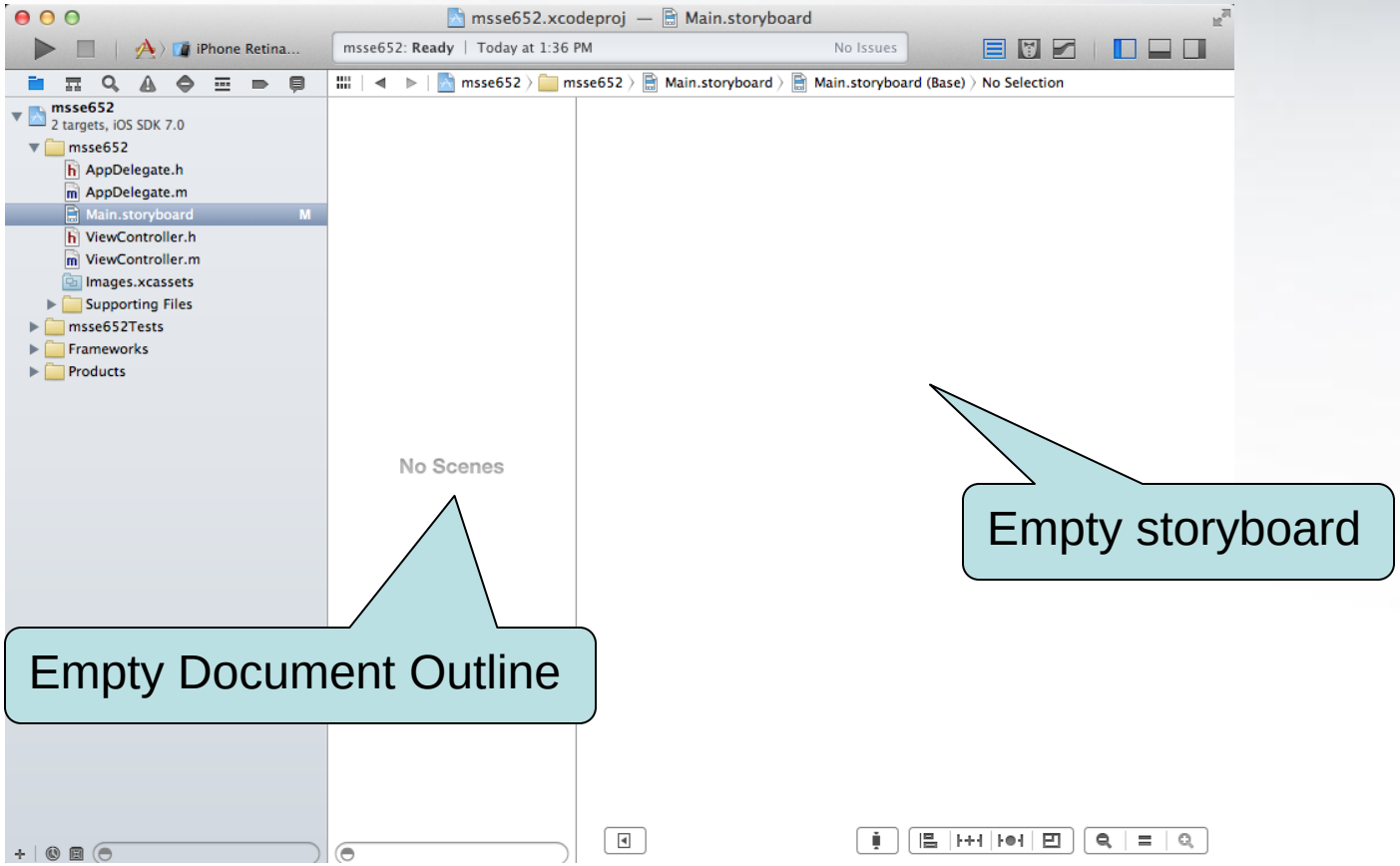
Starting from scratch

- The Single View Application template provides a single scene displayed in the storyboard
  - let's delete the scene to create a empty storyboard …
    - click the scene's dock to select it (it becomes highlighted)
    - then use the "delete" button



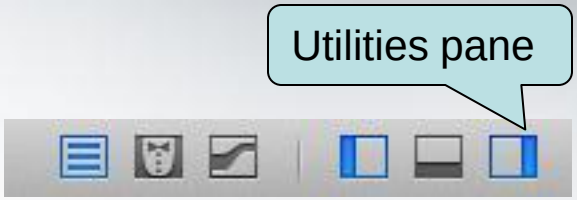
An empty storyboard

- Deleting the scene yields an empty storyboard

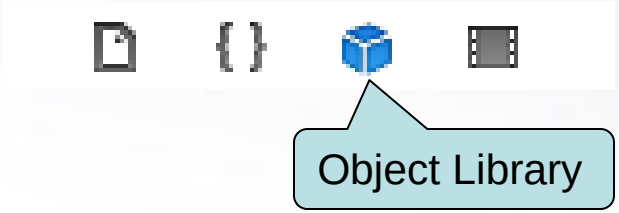


Creating scenes

- To create a new scene on the storyboard …
  - display the Utilities pane
    - select the far right icon on the toolbar

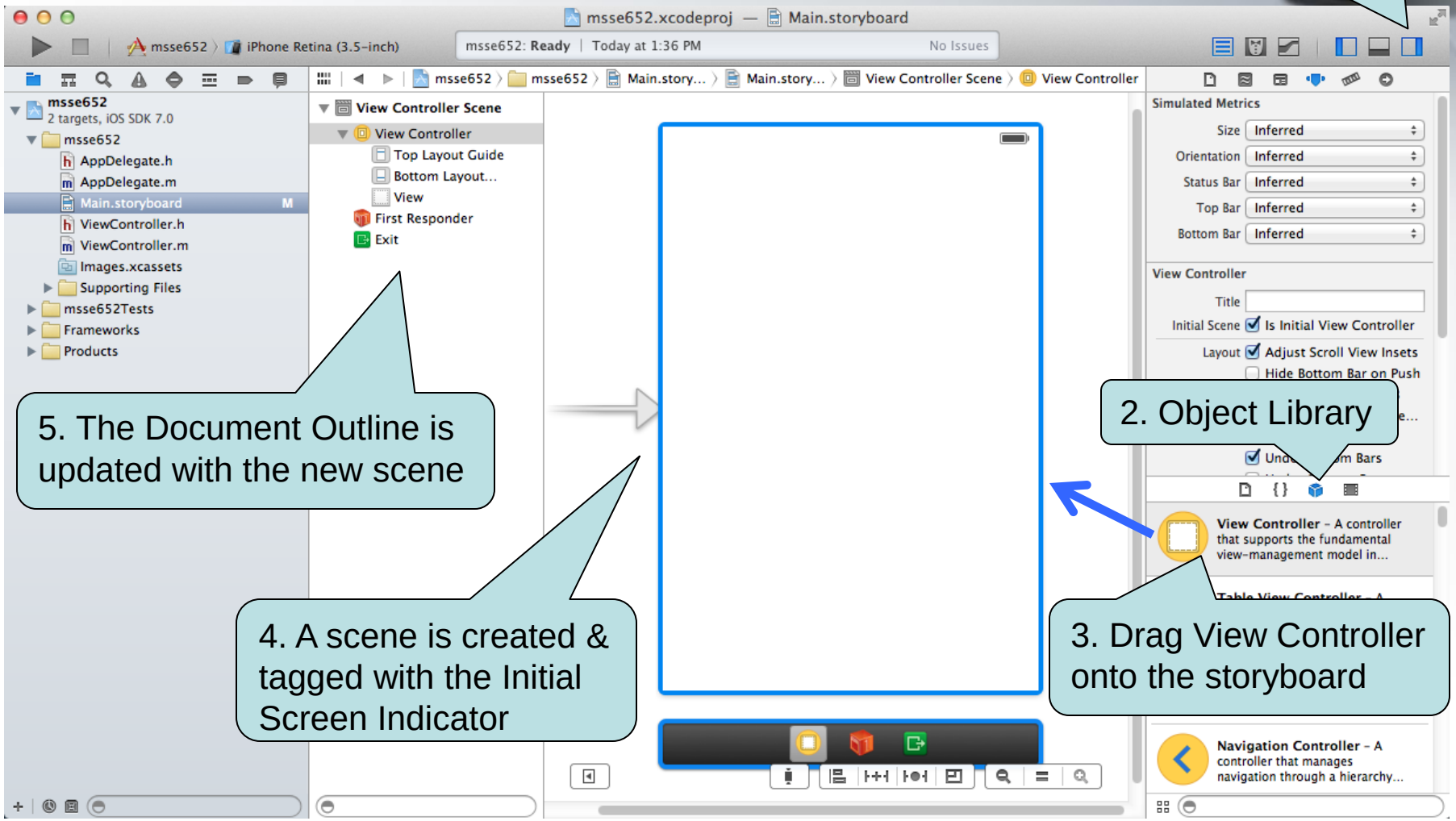


  - in the Utilities pane, display the Object Library
    - by selecting the “cube looking” icon



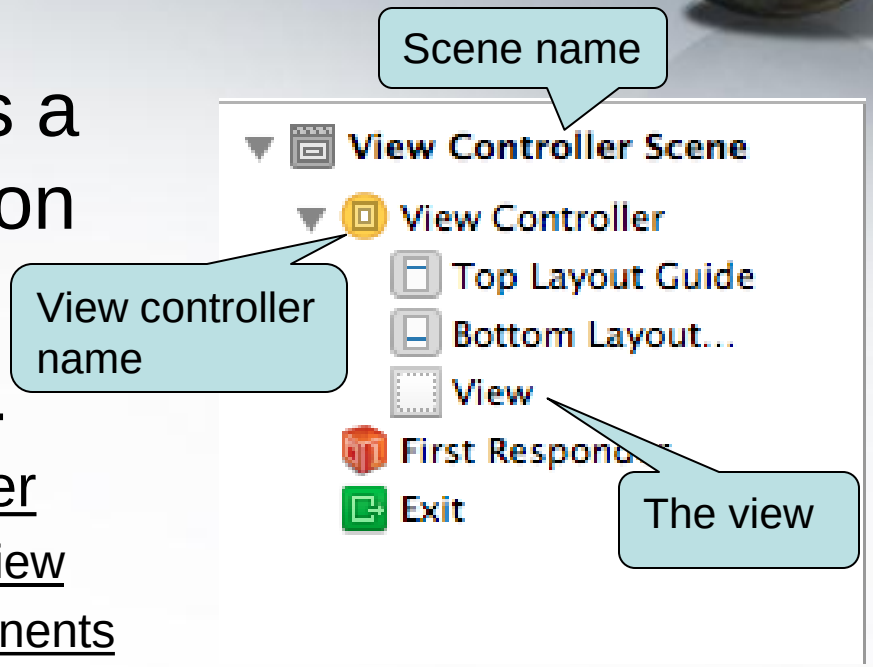
  - from the Object Library pane, …
    - drag a “View Controller” onto the storyboard

Creating scenes (cont'd)



The Document Outline

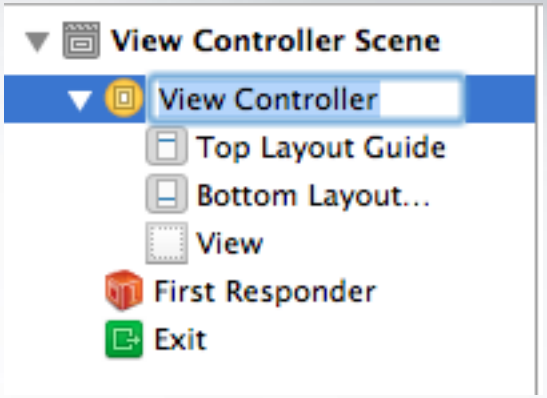
- The Document Outline is a hierarchical representation of the storyboard
  - listing each scene, and …
    - the scene's view controller
      - and the view controller's view
        - » and the view's components



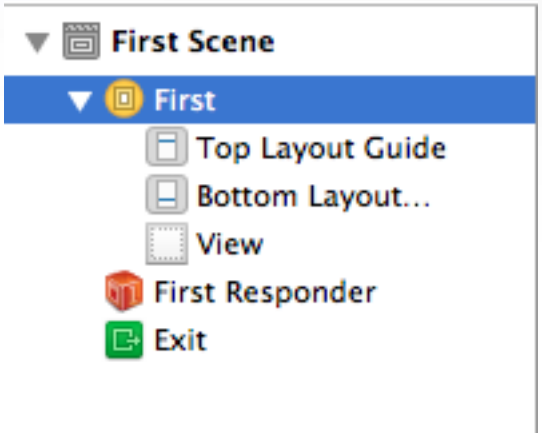
- Notice Xcode's naming convention …
  - the view controller's default name is "View Controller"
  - the scene's name is derived from the view controller
    - Xcode appends "Scene" to the view controller's name

Changing the name(s)

- You can change the name of the view controller
  - In the Document Outline,
    - select the View Controller
      - and then click it again

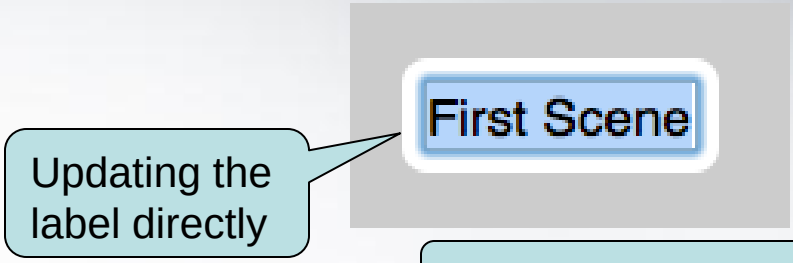


    - type any “meaningful” name
      - e.g., First
    - and the scene’s name is updated too
      - e.g., First Scene

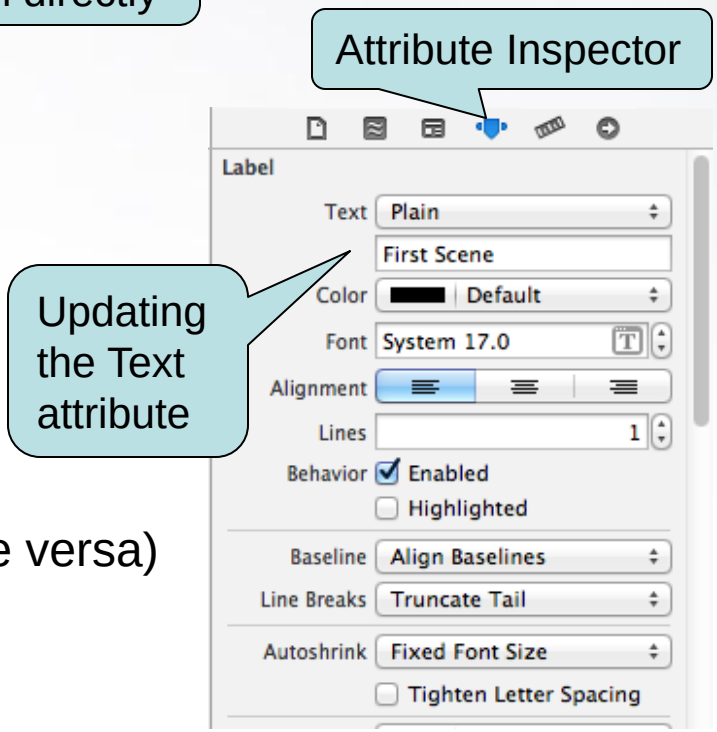


Adding a Label to a scene

- Drag and drop a Label from the Object library
  - and change its text …
    - double click on it and type



    - or use the Attributes Inspector
      - and update the “Text” field



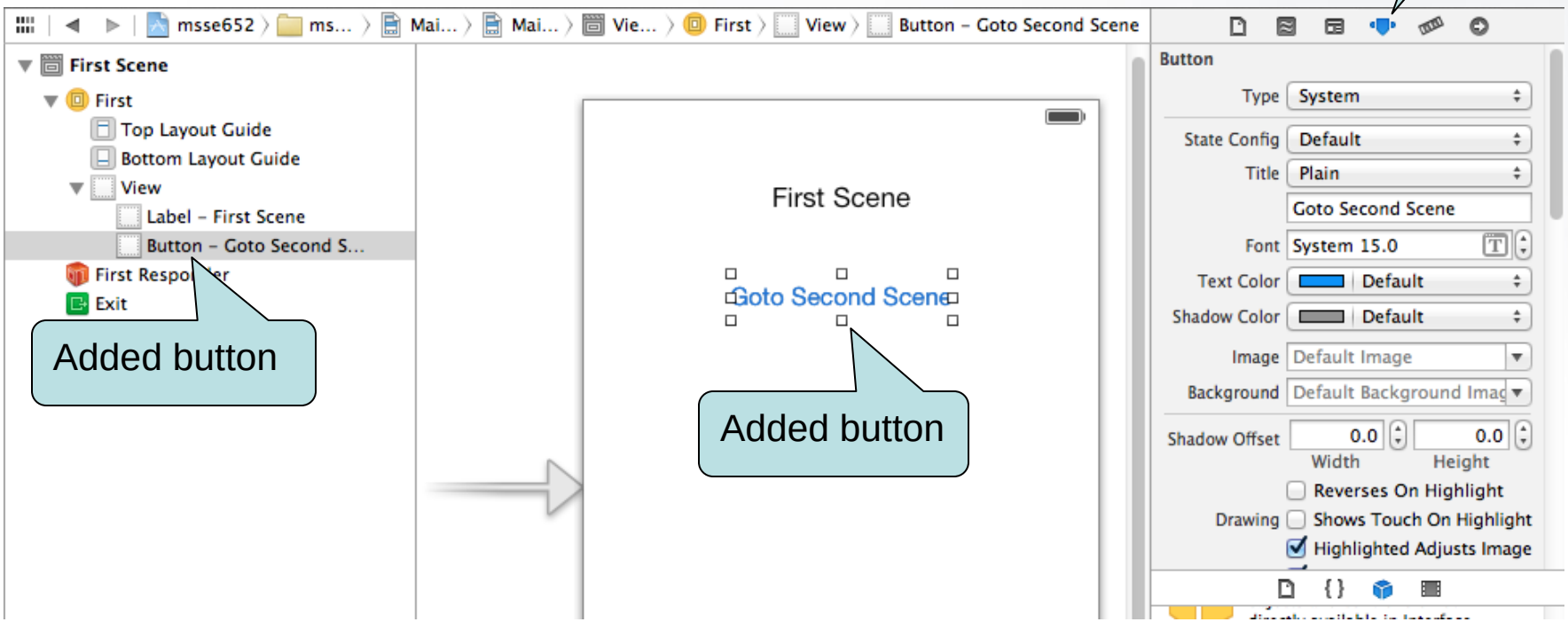
  - Note: either way works
    - if you update the label directly …
      - the attribute is also updated (and vice versa)

The Document Outline is updated

- The label is added to the Document Outline
  - under the corresponding View

Adding a button to the scene

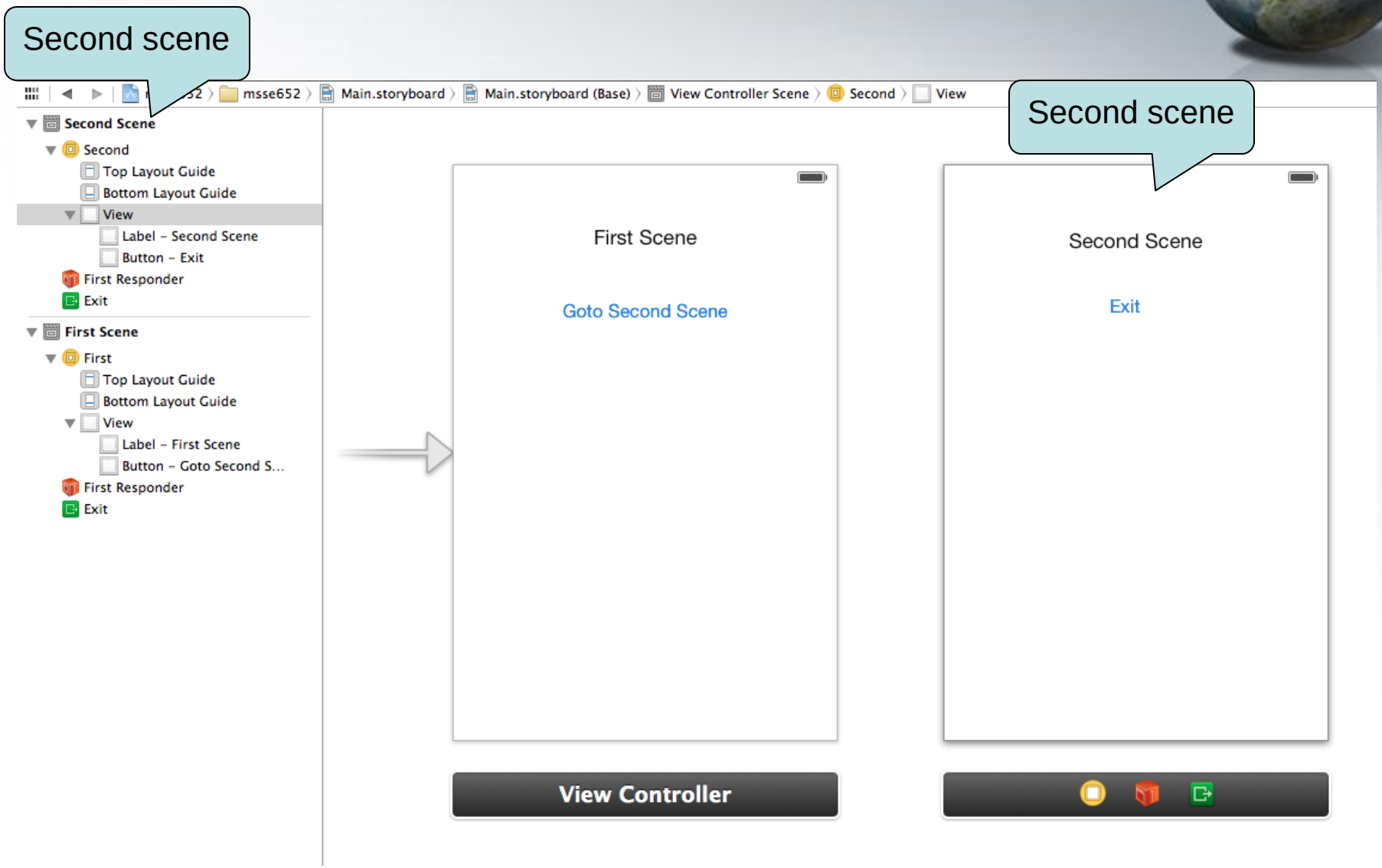
- Now add a button from the Object library
  - and change its text to “Goto Second Scene:
    - by either double clicking or using the Attribute Inspector



Adding a second scene

- Let’s repeat the process and add a second scene to the storyboard
  - Drag a View Controller from the Object library
  - In the Document Outline, rename the View Controller to “Second”
    - the scene’s name should be updated to Second Scene
  - Add a label to the scene with text “Second”
  - Add a button to the scene with text “Exit”
- For instance …

Adding a second scene (cont'd)

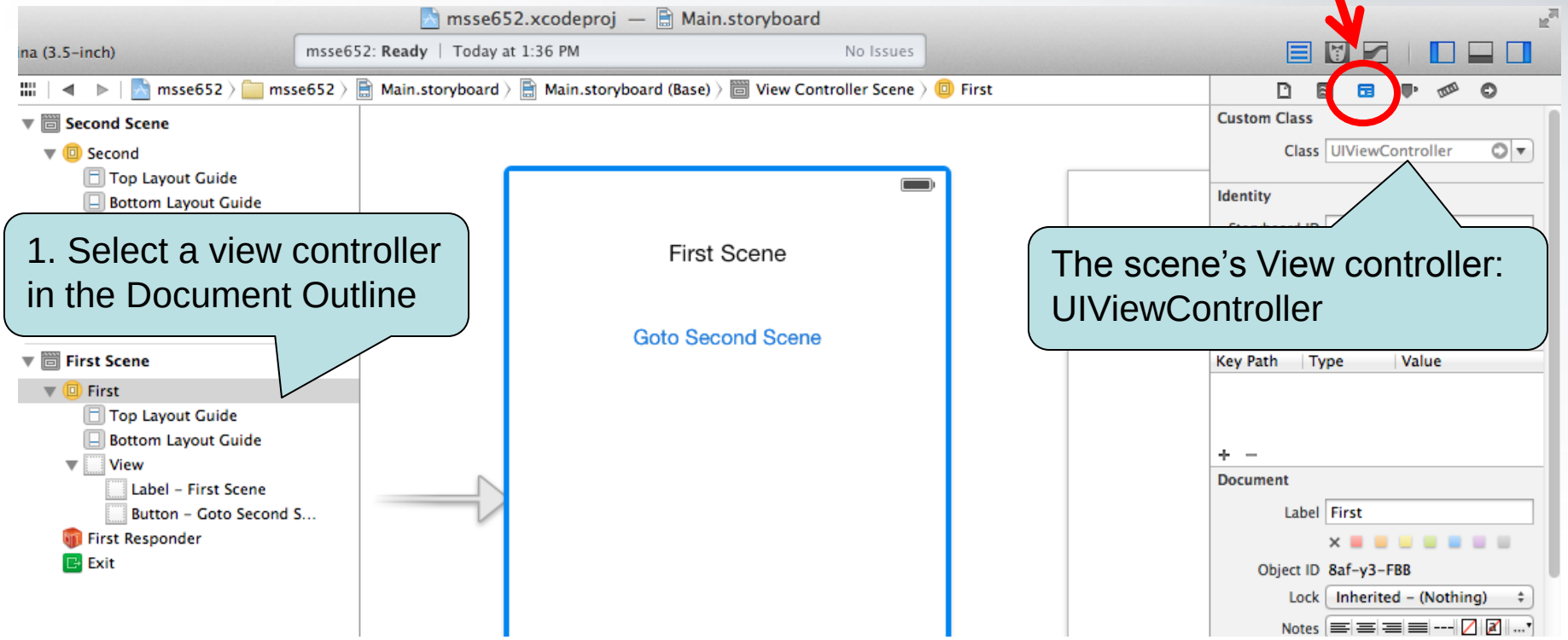


View controllers

- When a scene is added to the storyboard, its view controller is the class UIViewController, …
  - a class defined by the Cocoa Touch framework
- In order to provide custom behavior that responds to user events,
  - we need to create a custom view controller class
    - and associate it with the appropriate scene
- For instance …

The Identity Inspector

- Displays the scene's view controller



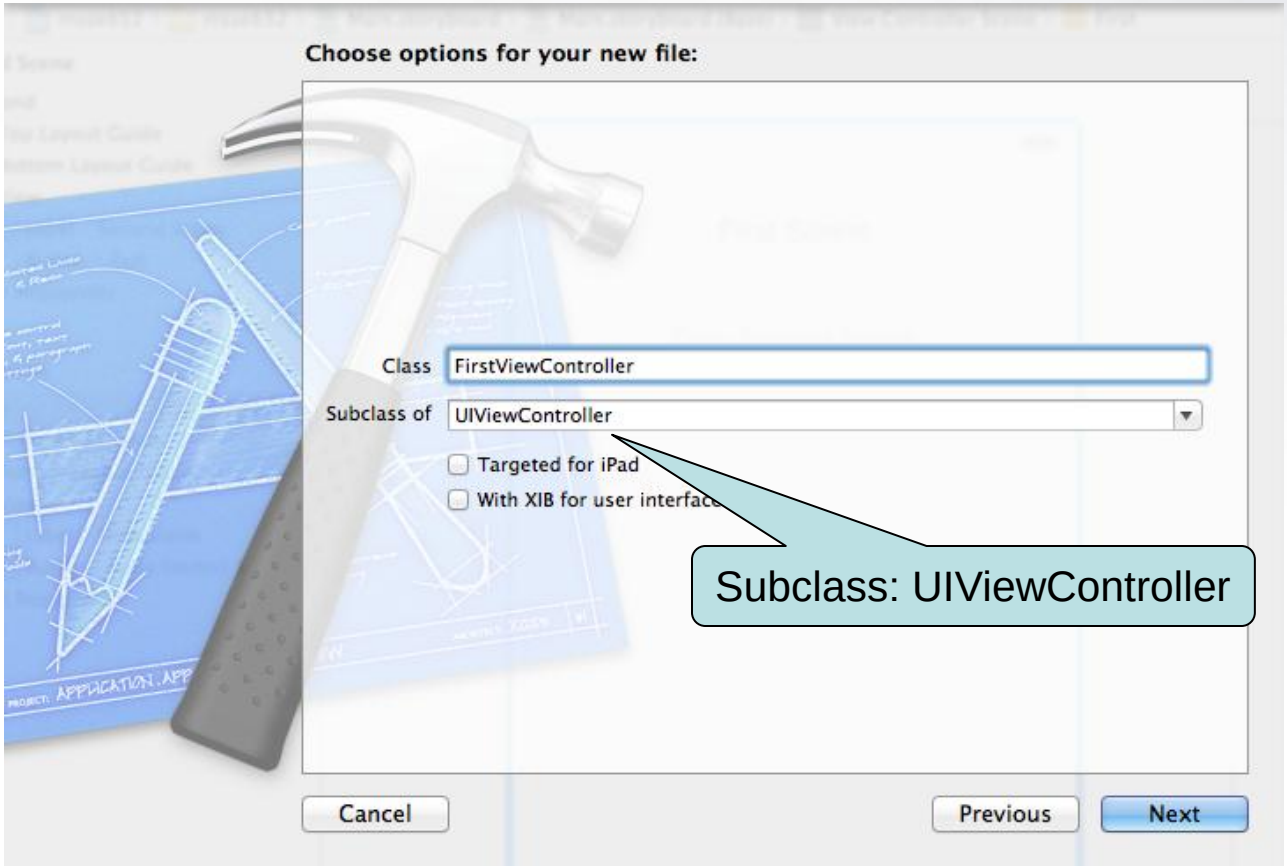
Creating Custom View Controllers

- A two step process:
  - first, create a class the extends UIViewController
    - using either File -> New -> File …
    - or the + sign at the bottom of the Project Navigator
  - then assign the above custom class to an appropriate scene using the Identity Inspector's dropdown listbox

- Let's do the above for both of our scenes
  - name one view controller: FirstViewController
  - name the other view controller: SecondViewController

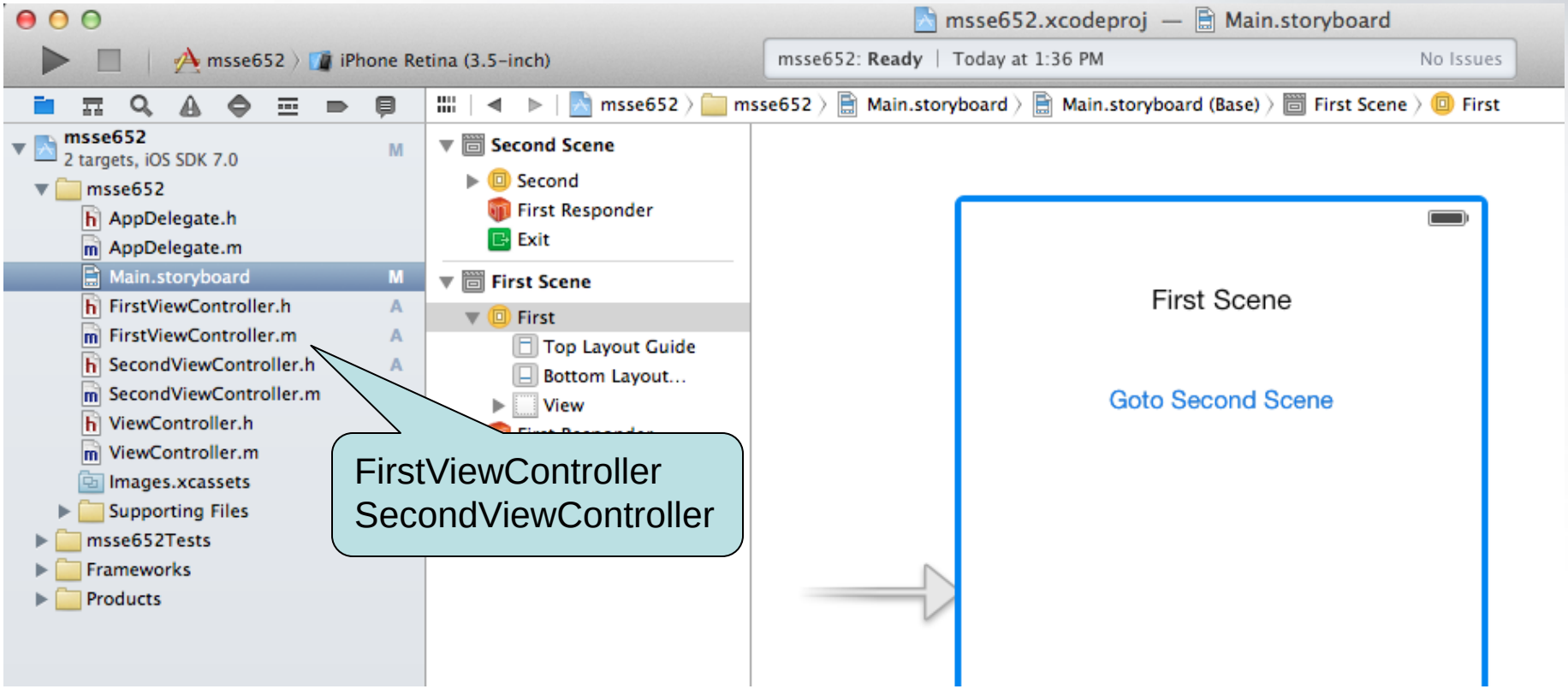
Creating View Controllers (cont'd)

- When you create the custom view controller class,
  - specify UIViewController as the subclass



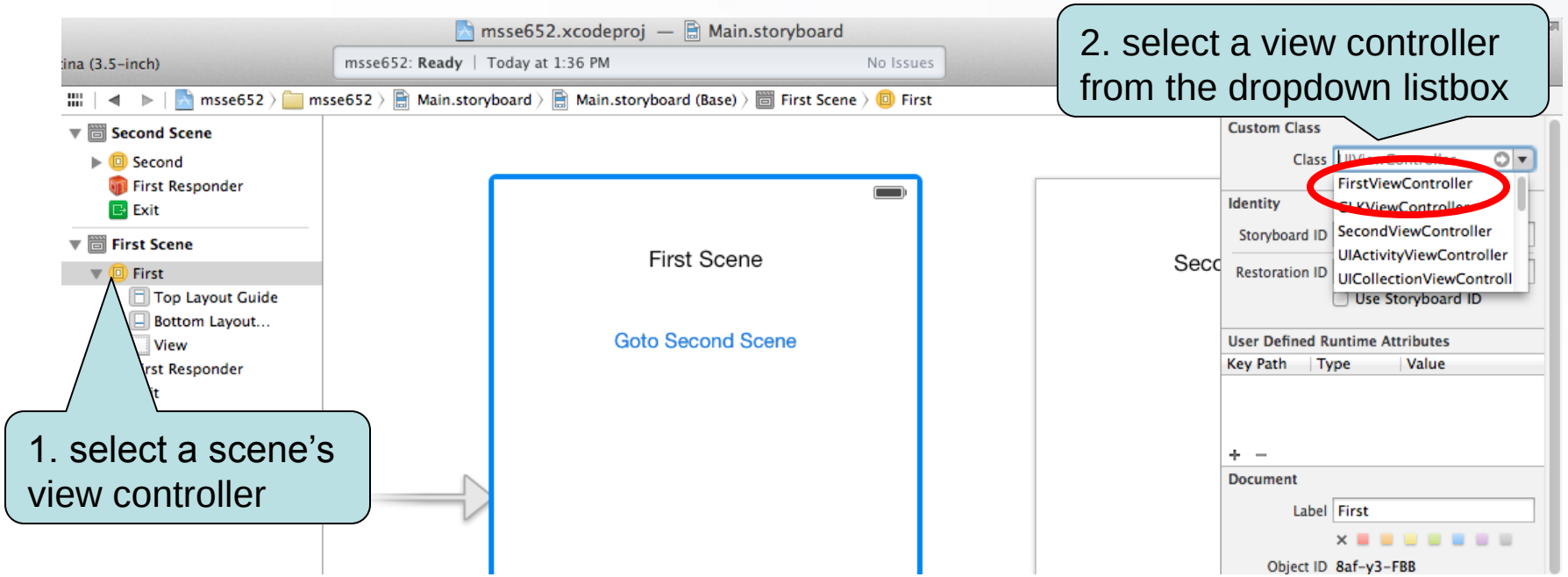
Creating View Controllers (cont'd)

- Updated Project Navigator



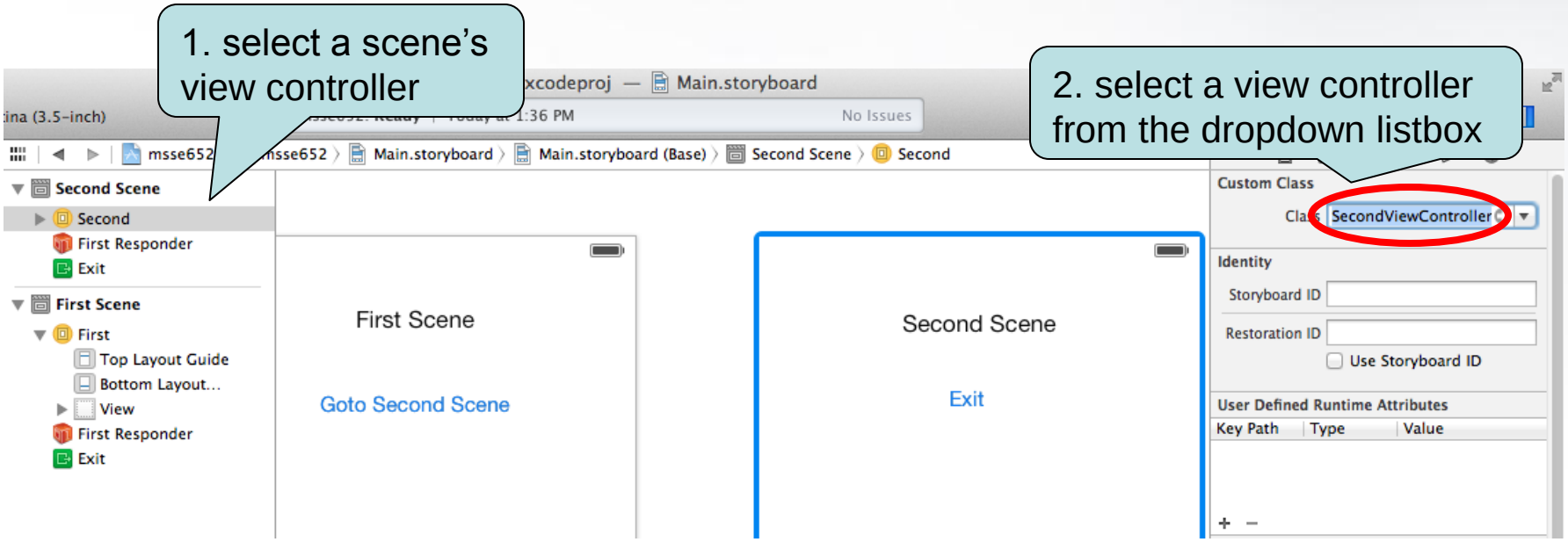
Assigning View Controllers

- To assign a custom view controller …
  - select a scene's view controller in the Document Outline
  - and then display the Identity Inspector and select the view controller from the dropdown list box



Assigning View Controllers (cont'd)

- Repeat the process for the second scene's view controller



Creating segues

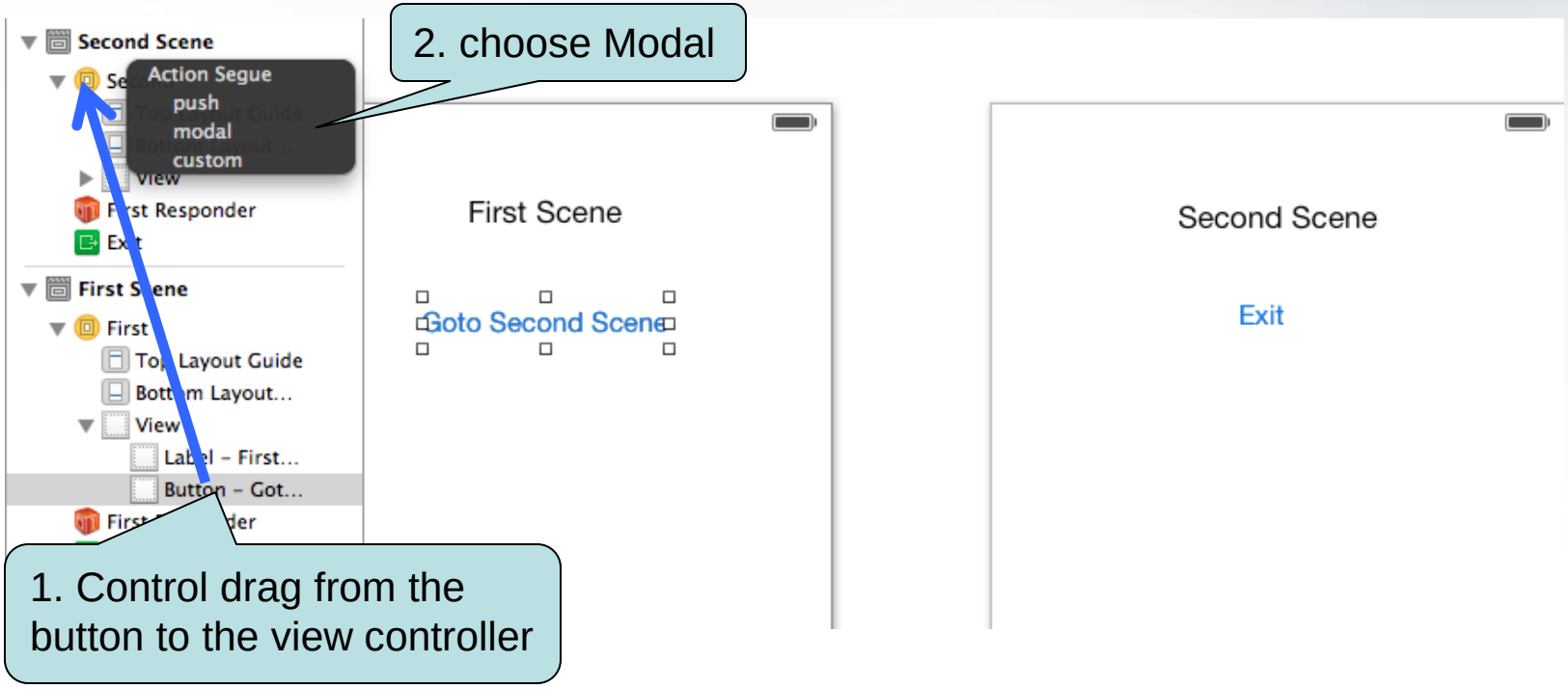
- Segues represent navigation paths between scenes; e.g.,
  - if one scene can launch another scene, it's denoted in the storyboard with a segue
    - note: the segue can be used to share info between scenes
- We want to create segues between our two scenes; that is, we want to …
  - transition from First Scene to Second Scene
  - and then "unwind" from the Second Scene to the First

Creating segues (cont'd)

- Let's first create a segue for the forward direction
  - i.e., from the First scene to the Second scene
- In Xcode, …
  - display the storyboard
  - position the scenes so they are both visible
  - then "control key, click & drag" …
    - from the First scene's button (Goto Second Scene)
    - to the Second scene's view controller
      - either in the Document Outline or in the storyboard

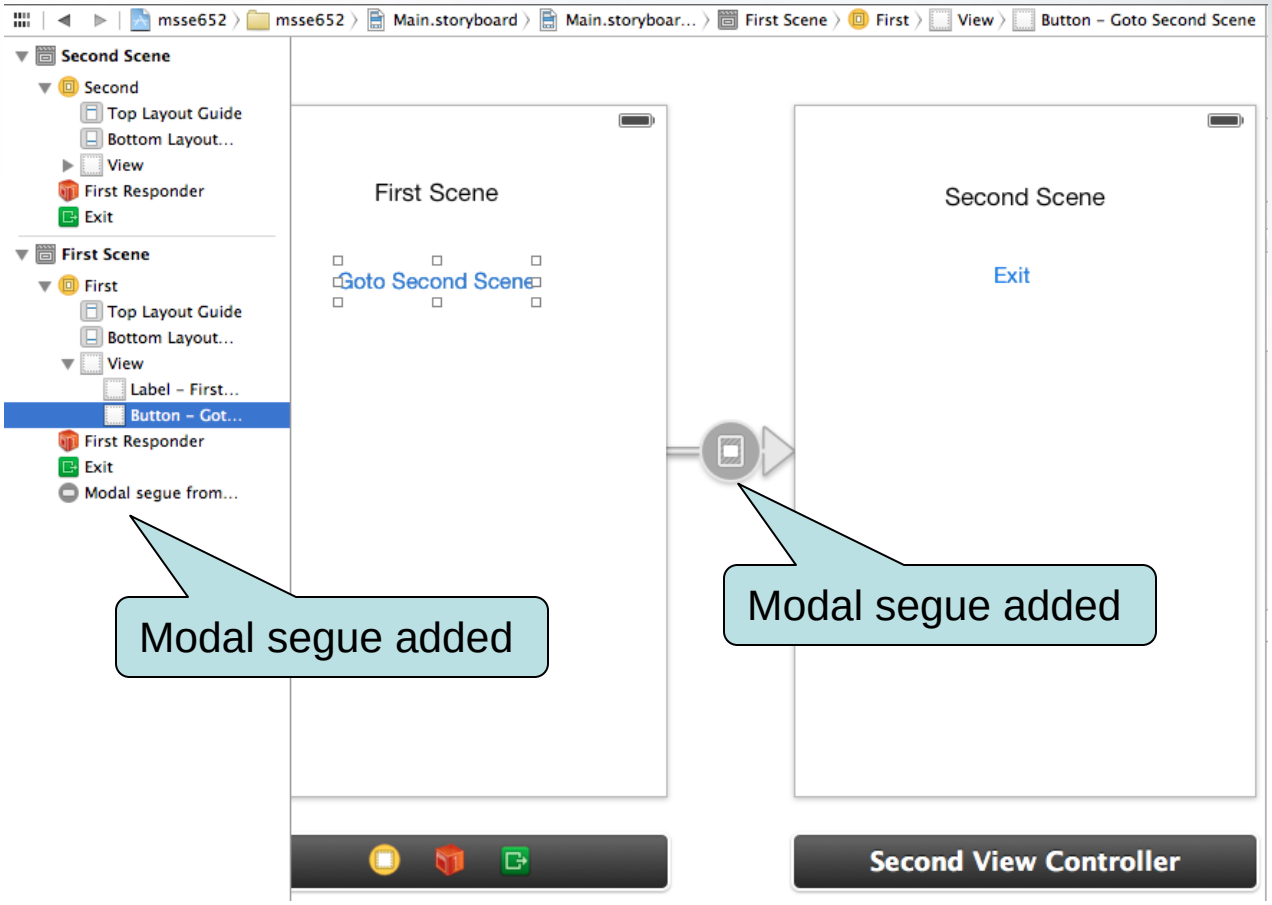
Creating segues (cont'd)

- A popup appears … choose "modal"



Creating segues (cont'd)

- A segue appears in the storyboard



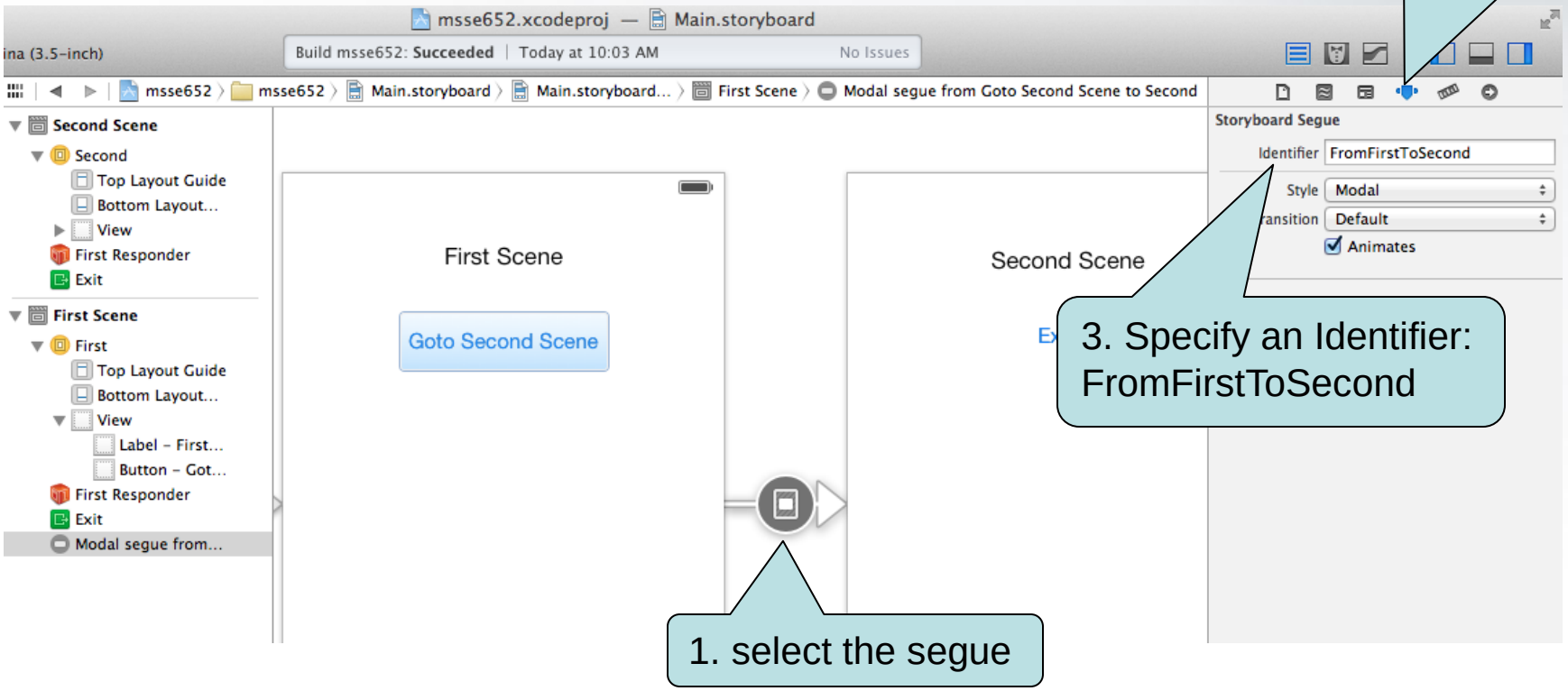
Specifying a segue identifier

- Segues should/must be assigned a unique identifier to allow them to be accessed in code

- To specify an identifier for a particular segue …
  - select the segue in the Document Outline
  - display the Attributes Inspector in the Utilities pane
  - in the Storyboard Segue pane,
    - specify a text value in the Identifier field
      - note: the value should describe the segue: FromFirstToSecond

Specifying a segue id (cont'd)

- Id: FromFirstToSecond



Run the app in the simulator

- If you run the app, here's what happens:
  - the First Scene is displayed
  - if you click "Goto Second Scene",
    - the Second Scene appears
  - but if you click on "Exit", nothing happens
    - that's because we haven't setup a segue to get back to the First Scene
- What shall we do?
  - Well, you could repeat the previous process …
    - and it would work, but don't do that; and why is that? …

Modal segues and Unwinding

- Modal segues are used to …
  - display a new scene
  - and then once the user is done with the scene, the UI should “unwind” back to a previous scene
    - “unwinding” removes scenes from an internal iOS stack
- In particular, the way to unwind is to navigate to the “Exit” icon of the destination scene
  - note: the Exit button is displayed in both …
    - the Document Outline
    - the Scene’s Dock

Unwinding a Modal Scene

- So, to unwind a Modal scene, here's what we need to do …
  - create a stubbed out method in the destination scene view controller (i.e., FirstViewController)
    - with the following method signature:

      ```
      - (IBAction) exitHere:(UIStoryboardSegue *)sender;
      ```

  - create a segue from the modal scene back to the Exit icon in the destination scene
    - and choose the above stubbed out method in the popup that gets displayed when the segue is created

Creating the “Exit” method

- In both the interface and the implementation

msse652.xcodeproj — FirstViewController.h

msse652 › iPhone Retina (3.5-inch) | Finished running msse652 on iPhone Retina (3.5-inch) | No Issues

msse652
2 targets, iOS SDK 7.0
- msse652
  - AppDelegate.h
  - AppDelegate.m
  - Main.storyboard M
  - FirstViewController.h A
  - FirstViewController.m A
  - SecondViewController.h A
  - SecondViewController.m A
  - ViewController.h
  - ViewController.m
  - Images.xcassets
  - Supporting Files
- msse652Tests
- Frameworks
- Products

FirstViewCo... › @interface FirstViewController

```
//
//  FirstViewController.h
//  msse652
//
//  Created by Robert Sjodin on 10/13/13.
//  Copyright (c) 2013 regis. All rights reserved.
//

#import <UIKit/UIKit.h>

@interface FirstViewController : UIViewController

- (IBAction) exitHere:(UIStoryboardSegue *)sender;

@end
```

method: exitHere

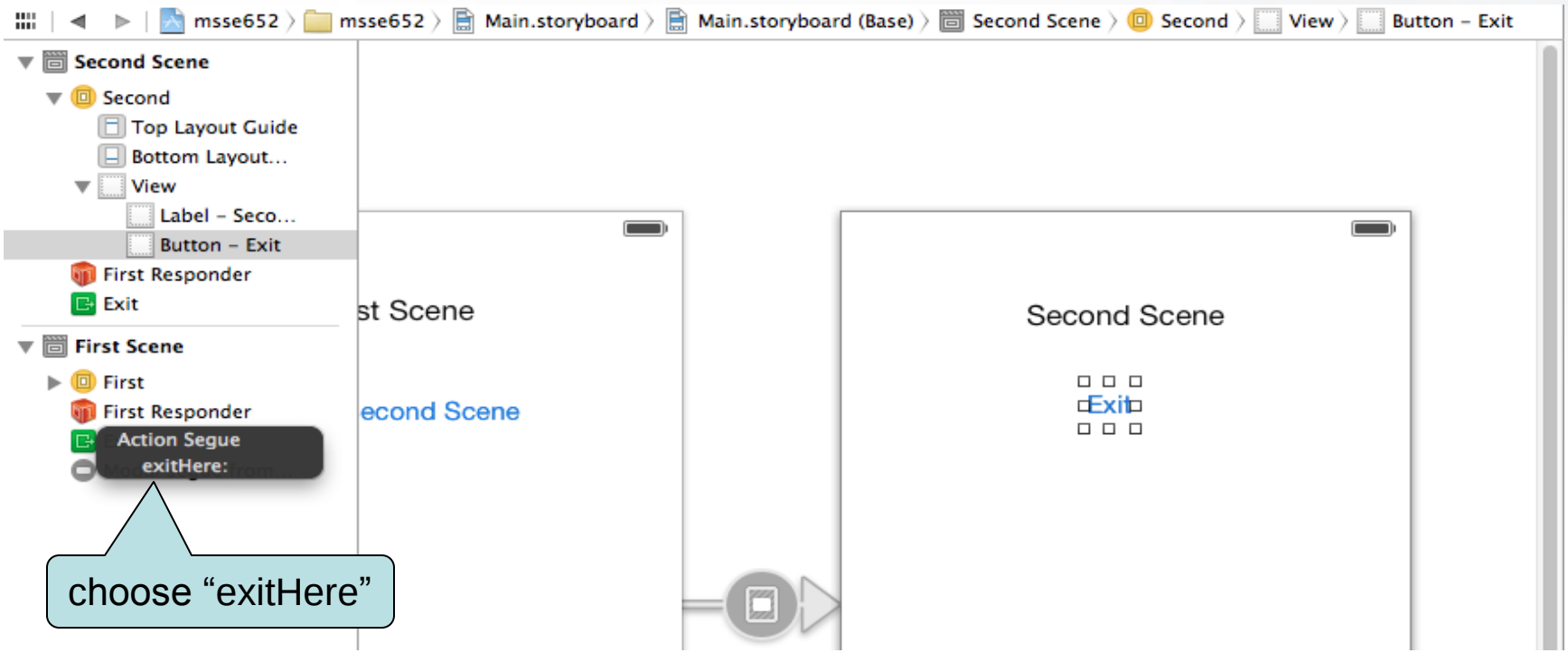
Fir... › @implementation FirstViewController

```
//
//  FirstViewController.m
//  msse652
//
//  Created by Robert Sjodin on 10/13/13.
//  Copyright (c) 2013 regis. All rights reserved.
//

#import "FirstViewController.h"

@interface FirstViewController ()

@end

@implementation FirstViewController

- (IBAction) exitHere:(UIStoryboardSegue *)sender {

}

- (id)init...(NSString *)nibNameOrNil bundle:

    if (self) {
        // Custom initialization
    }
    return self;
}
```

stubbed out exitHere impl

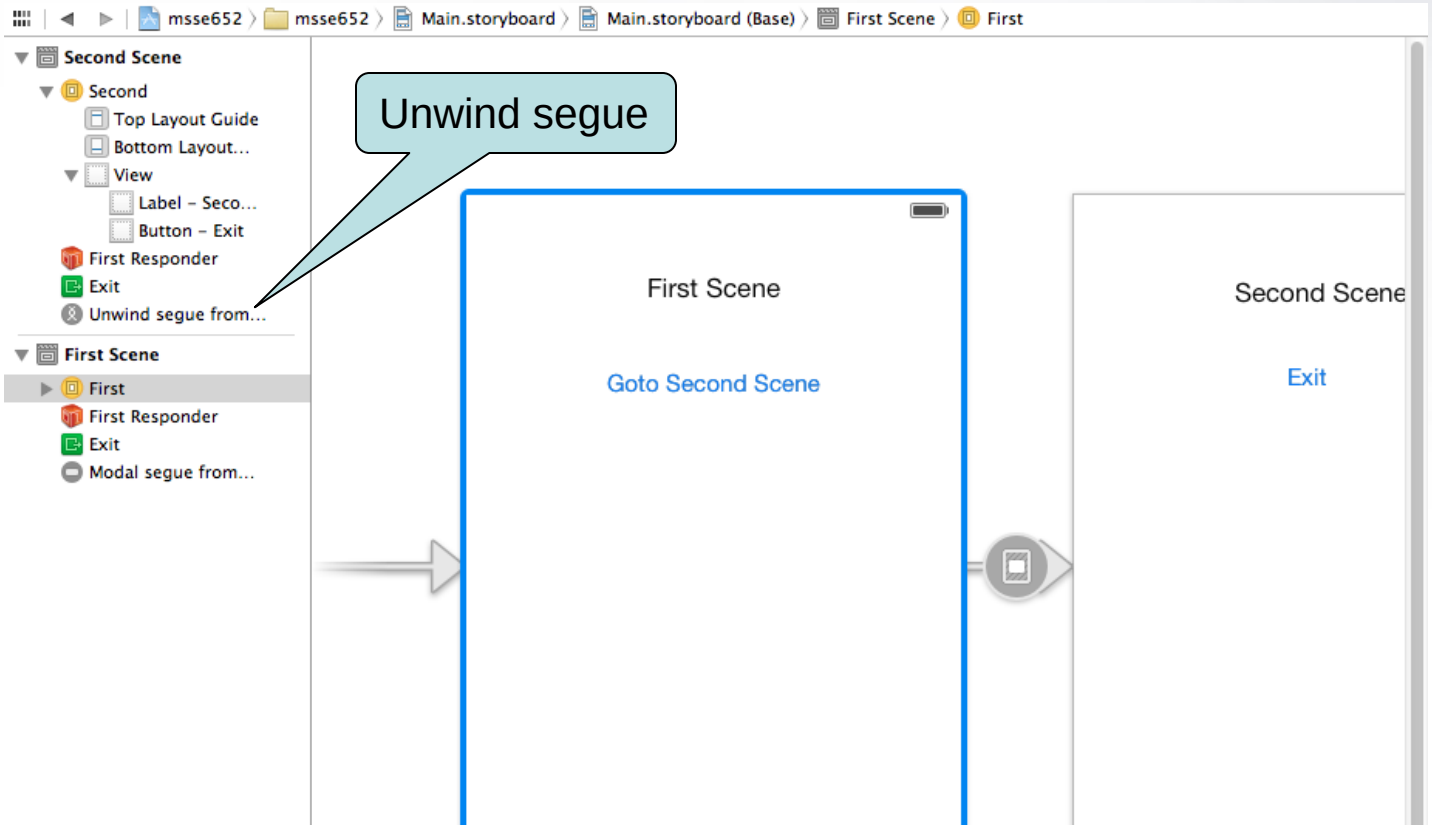
Create the “unwind” segue

- Control-drag from the Exit button to the First Scene’s view controller
  - in the popup, choose “exitHere”



Create the “unwind” segue (cont’d)

- The Unwind segue appears in the Document Outline (but not in the storyboard canvas)



The segue identifier

- As before, to specify an identifier for a particular segue …
  - select the segue in the Document Outline
  - display the Attributes Inspector in the Utilities pane
  - in the Storyboard Segue pane,
    - specify a text value in the Identifier field
      - note: the value should describe the segue: SecondToFirst

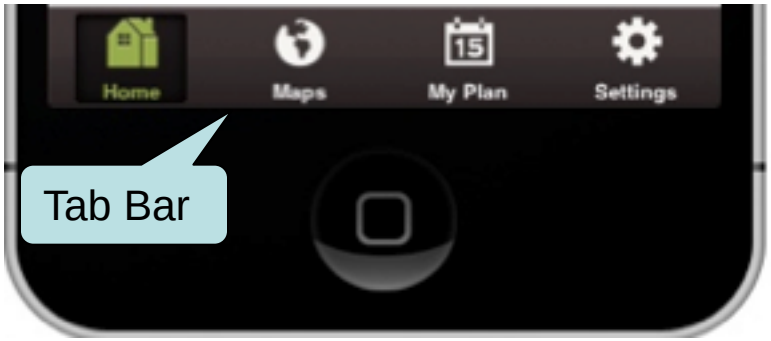
Run the app

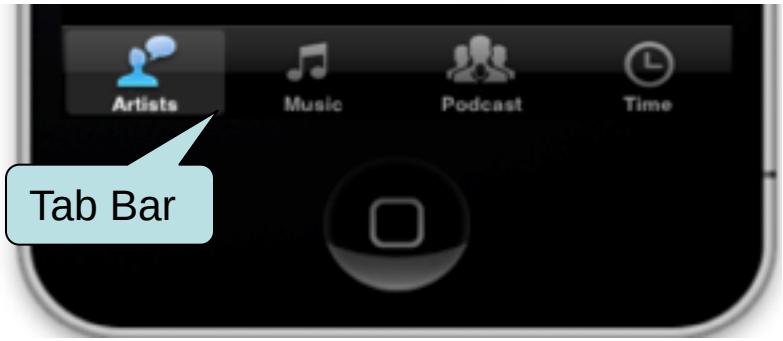
- If you run the app, here's what happens:
  - the First Scene is displayed
  - if you click "Goto Second Scene", …
    - the Second Scene appears
  - and if you click on "Exit", …
    - you should transition back (unwind) to the first scene

Tab Bars

- iOS Tab Bars contain tabs that allow the user to launch various scenes
  - where each scene represents a high-level capability of the app
    - e.g., a use case; a user story; a functional capability

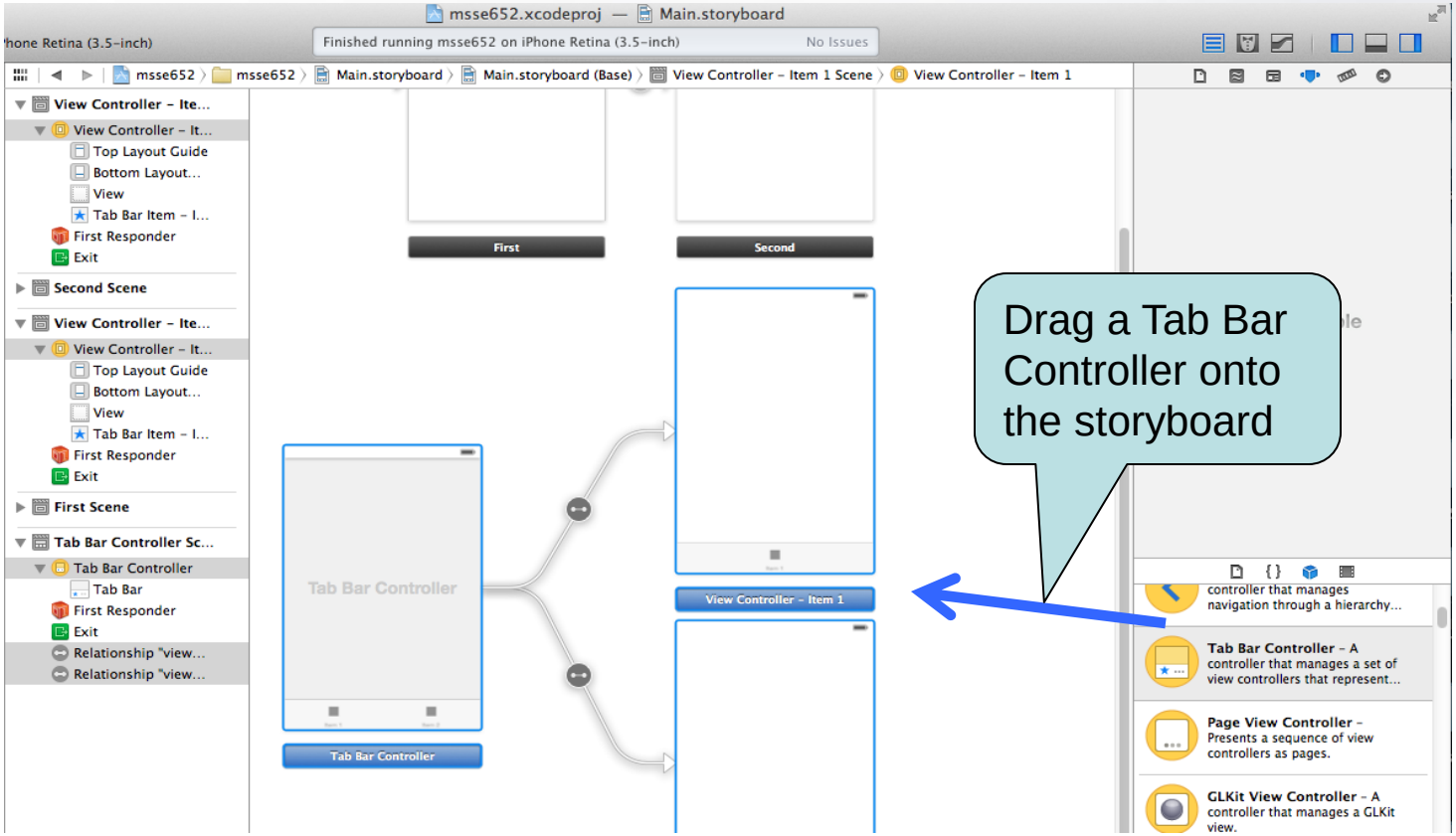
- The Tab Bar is located at the bottom of the screen





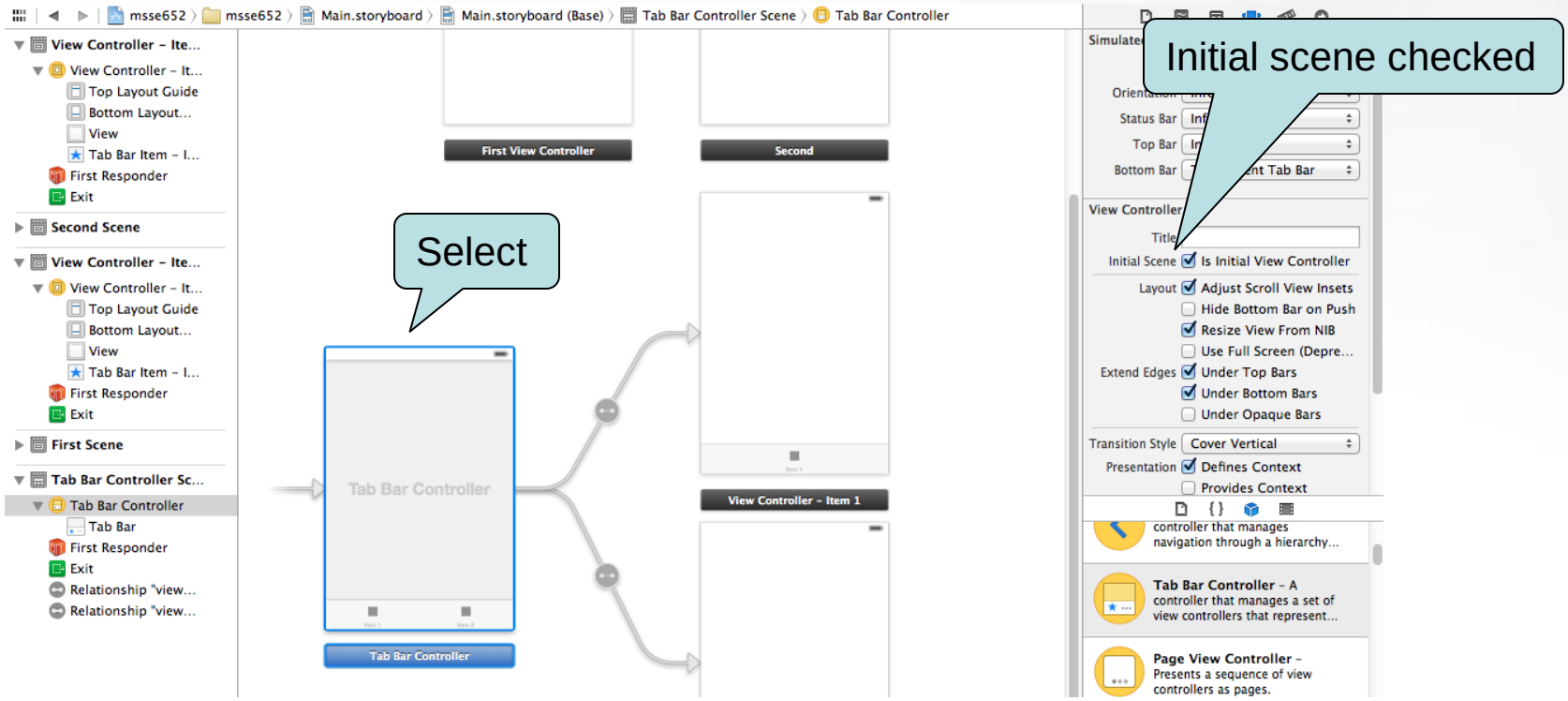
Creating a Tab Bar

- Technique: drag a Tab Bar Controller from the Object library onto the storyboard



Setting the Initial Screen Indicator

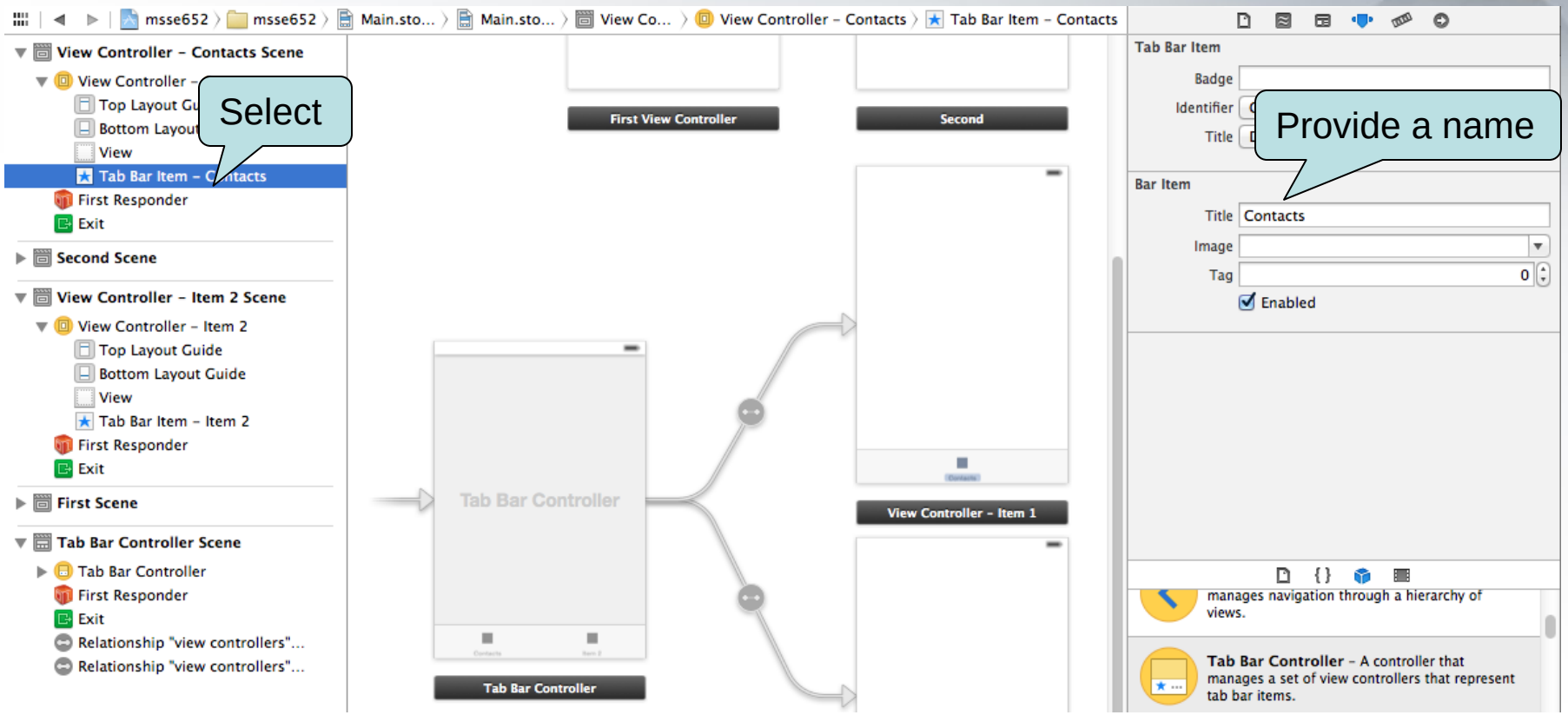
- With the Tab Bar selected, display the Attributes Inspector and set “Initial Screen Indicator”
  - so the Tab Bar is displayed when the app starts



Renaming the Tab Bar items

- The Tab Bar comes with two tabs, named …
  - Item 1
  - Item 2
- To rename the tabs, use the Document Outline and the Attributes Inspector
  - In the Document Outline,
    - expand the View Controller
    - Select the Tab Bar entry
  - In the Attributes Inspector, update the Bar Item Title
    - With a meaningful name: e.g., Contacts

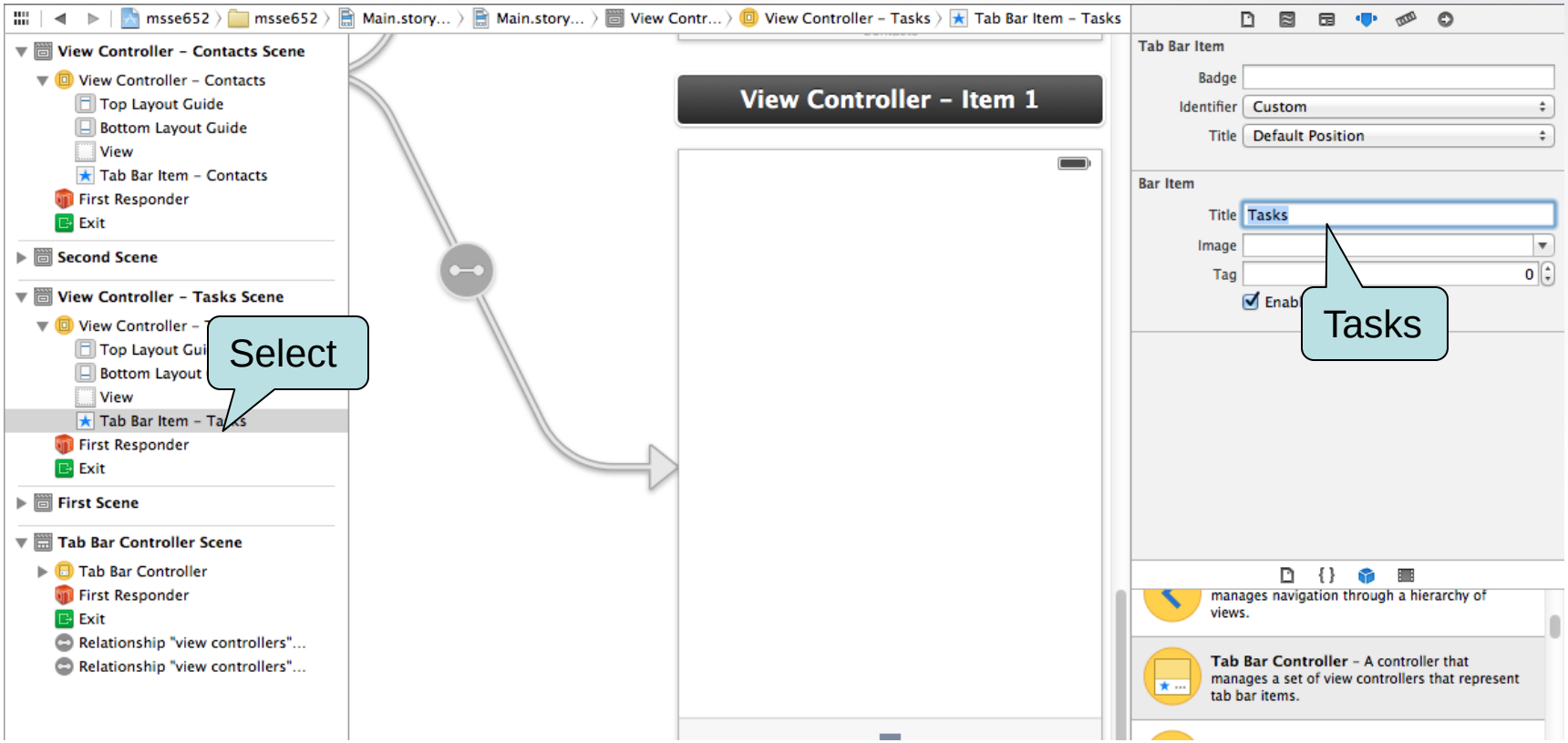
Renaming Tab Bar items (cont'd)



Note: the name you assign to the Tab Bar Item propagates up to both the View Controller and Scene

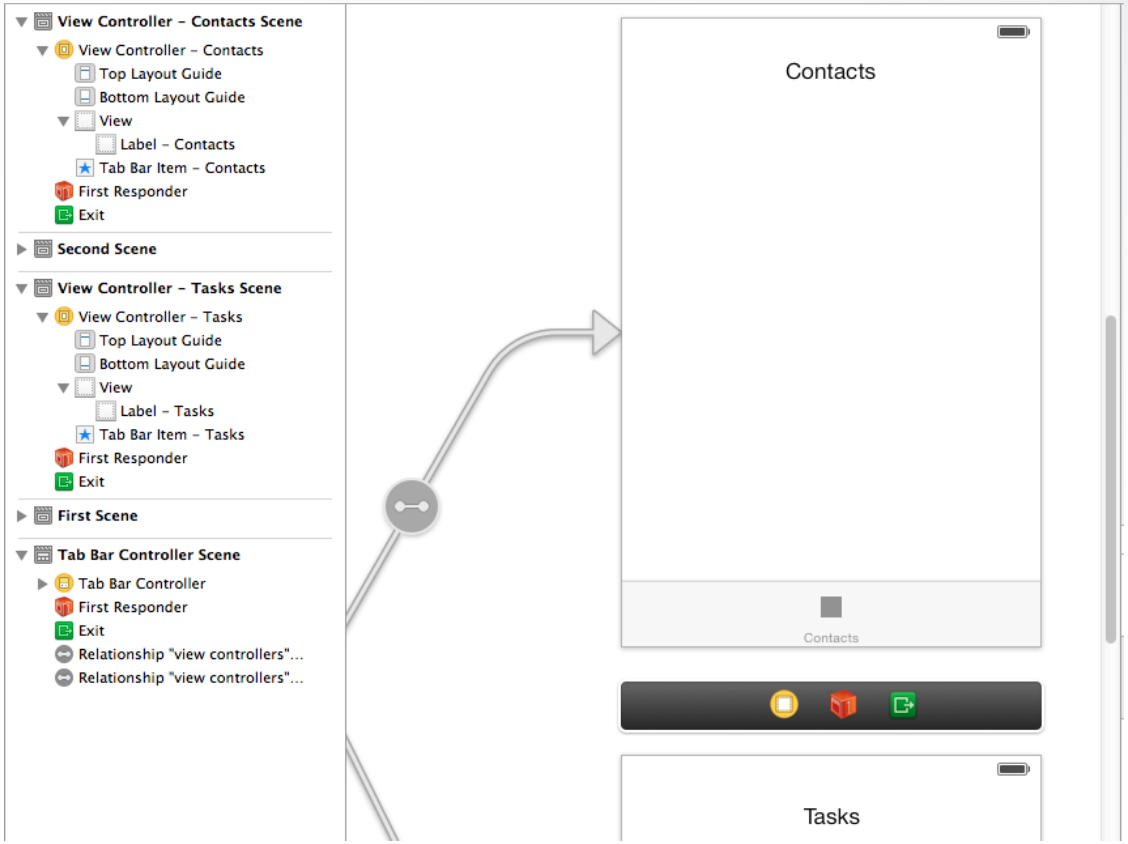
Renaming Tab Bar items (cont'd)

- Repeat the process for the second tab
  - giving it the name "Tasks"



Updating each scene

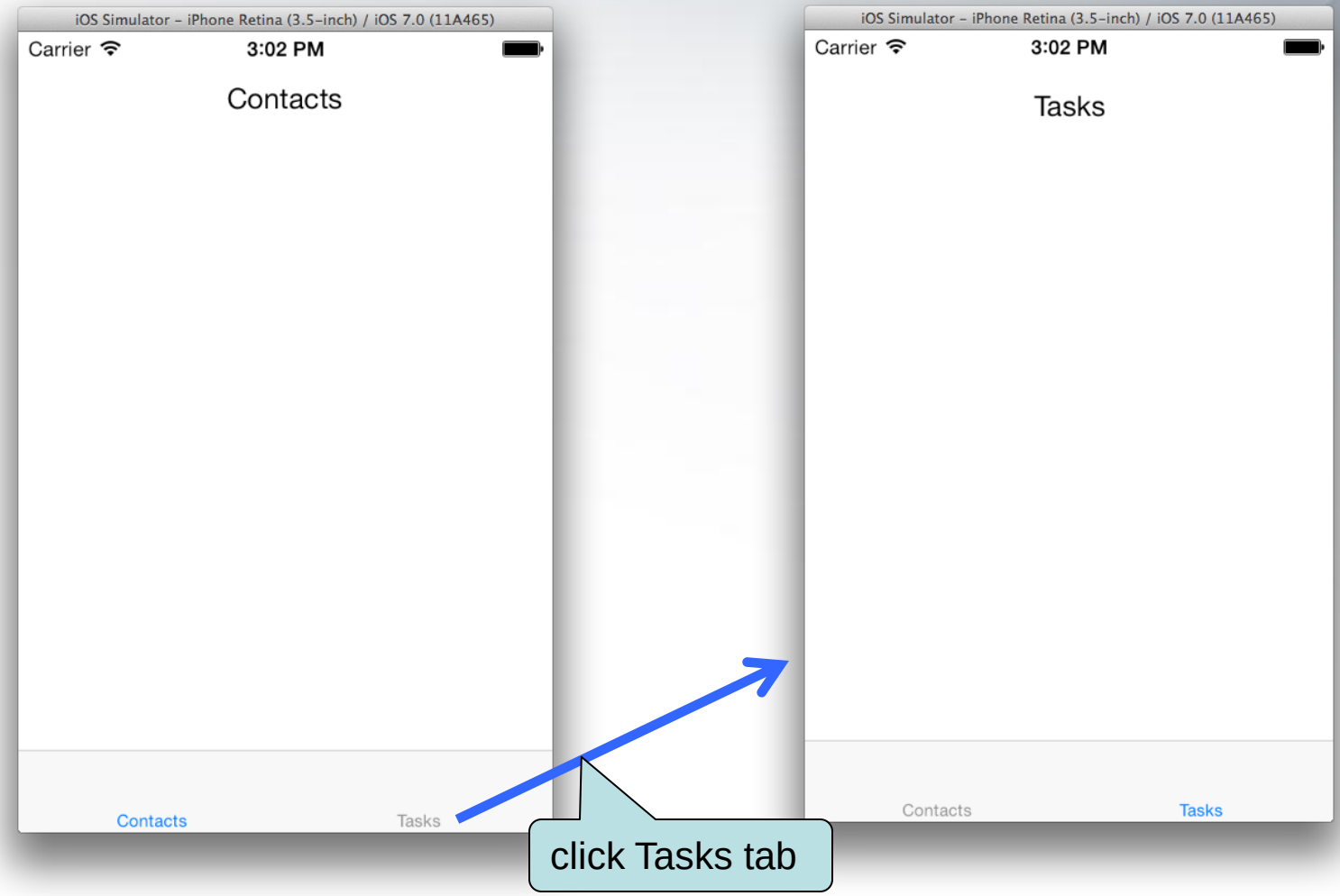
- Finally, place a distinct label on each scene
  - e.g., Contacts and Tasks



Run the app

- When you launch the app, the Tab Bar should be displayed at the bottom of the screen
  - initially with the Contacts tab activated
- If you choose the Tasks tab …
  - the Tasks scene will be displayed
- If you choose the Contacts tab …
  - the Contacts scene will be displayed

Running the App



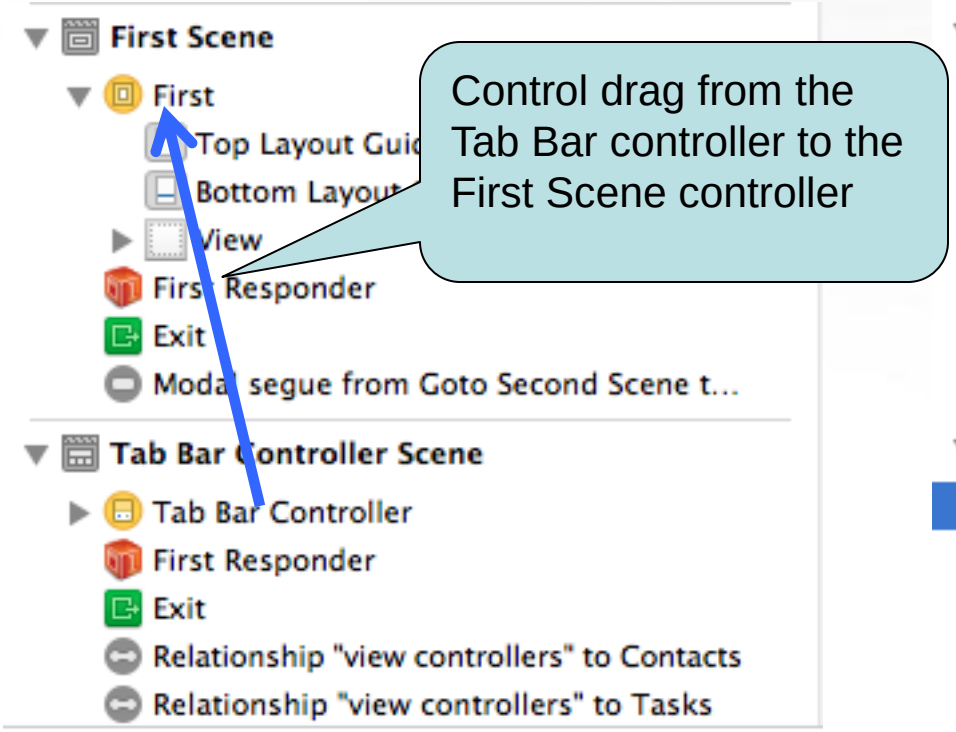
Adding additional tabs

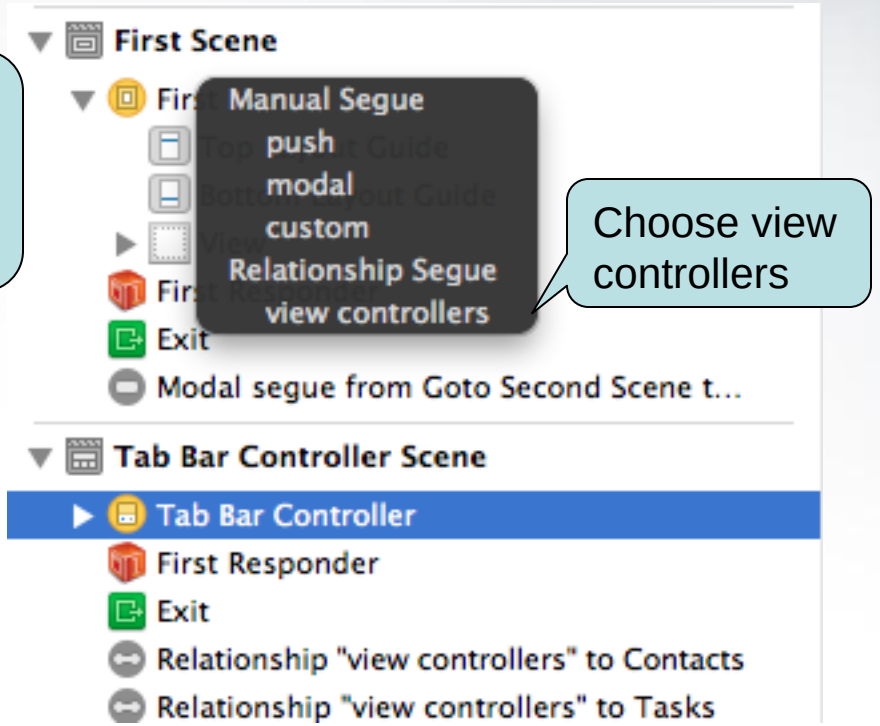
- To add an additional tab, you create a relational segue from the Tab Bar to an existing scene
  - that is, you control-drag …
    - from the Tab bar
    - to a scene on the storyboard
  - please note …
    - the new tab will be named “Item”
      - but it can be renamed using the technique described on the previous charts
        - » the name you give the tab will propagate up to the view controller and scene; they will acquire/inherit the tab name

For instance, let’s add our exiting First scene to the Tab Bar …

Adding the “First” scene

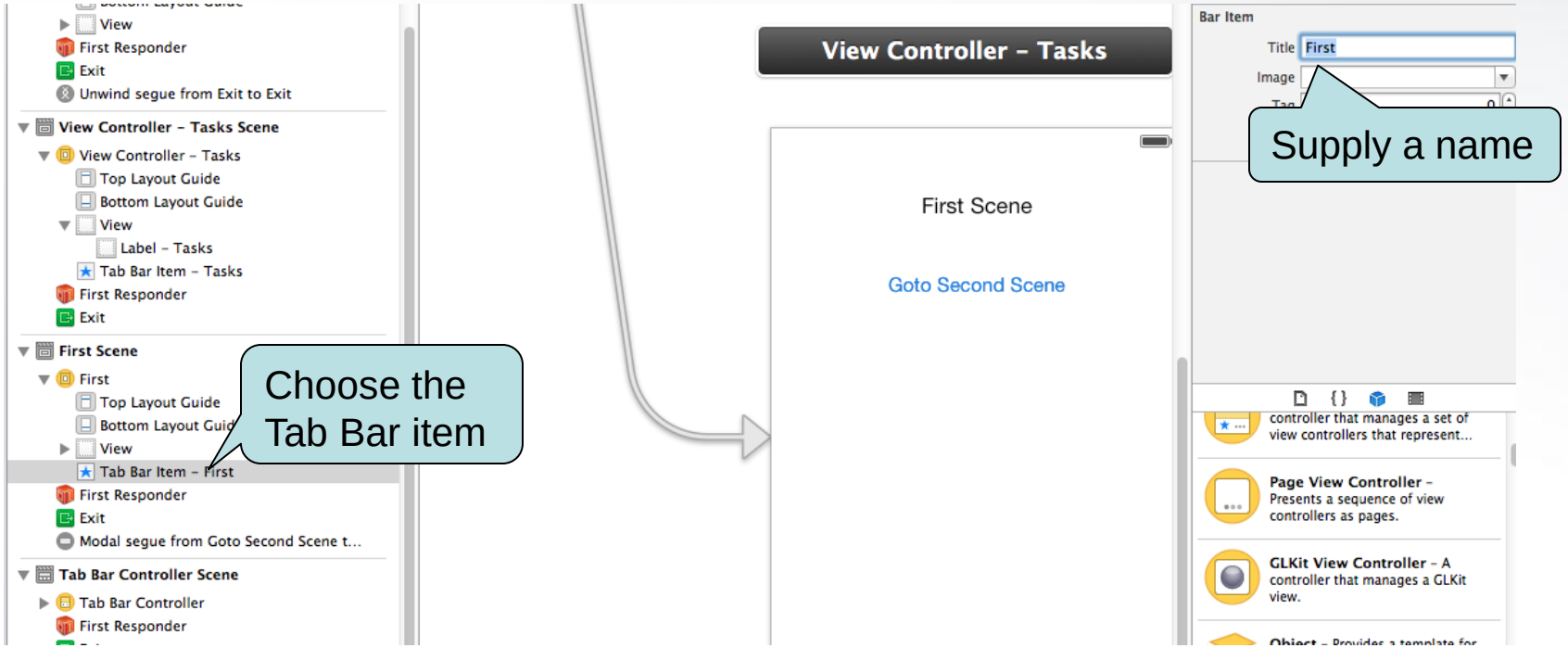
- Control-drag from the Tab Bar to the First Scene
  - in the popup, choose “view controllers”





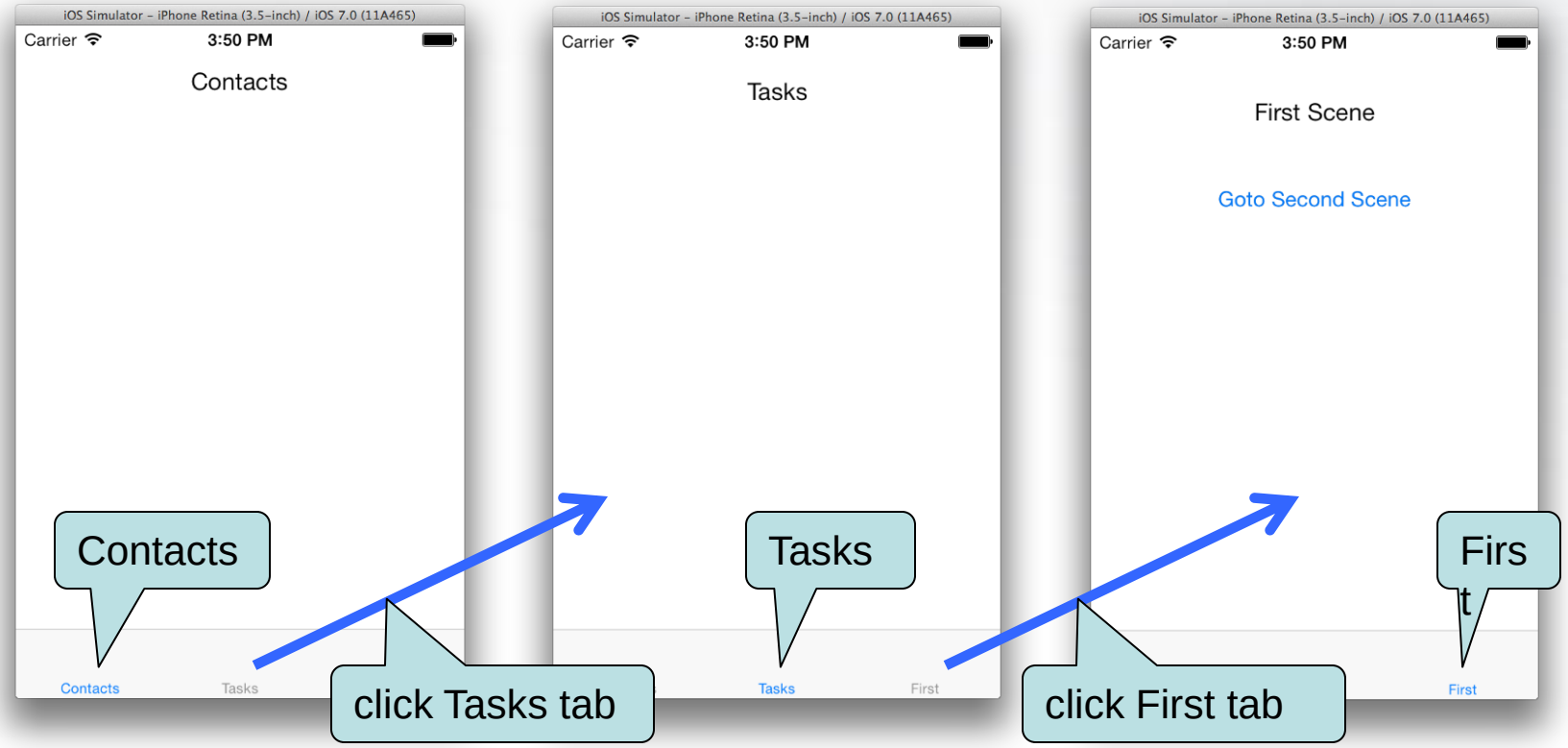
Adding the “First” scene (cont’d)

- Once the segue / tab is added, select it and change its name in the Attributes Inspector



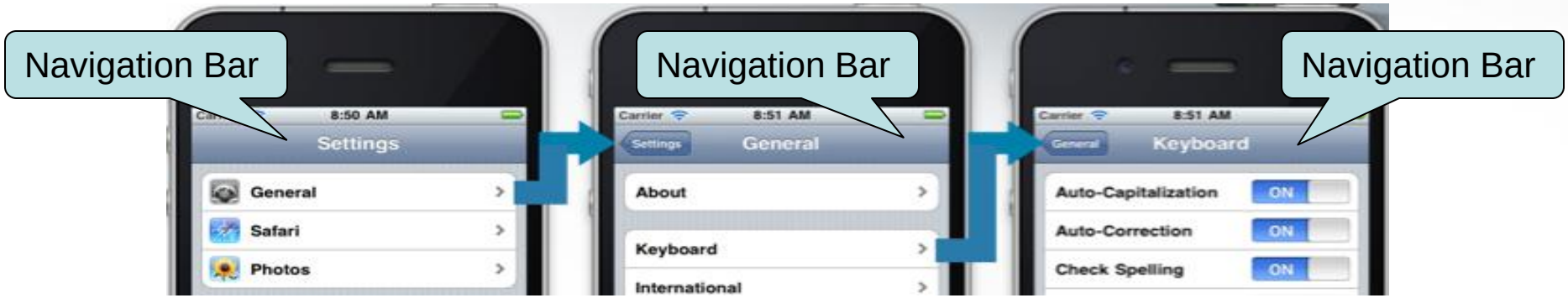
Run the app

- The Tab Bar now has three tabs
  - Contacts, Tasks, and First
    - and you can navigate from the First scene to the Second



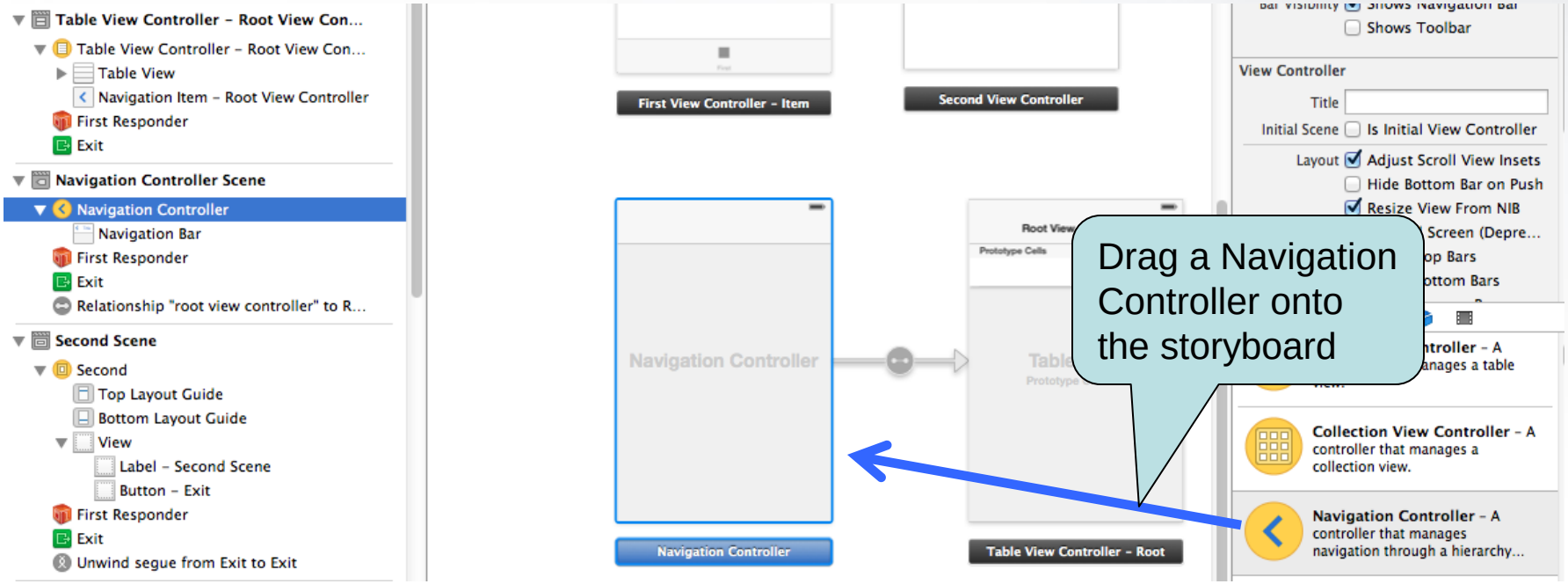
Navigation bars

- Navigation bars are used to facilitate the forward and backward navigation of scenes;
  - scenes that abstract …
    - high-level information
    - and more detailed information
- When present, Navigation bars are located at the top of the screen (e.g., consider the Settings app)



Creating a Navigation Bar

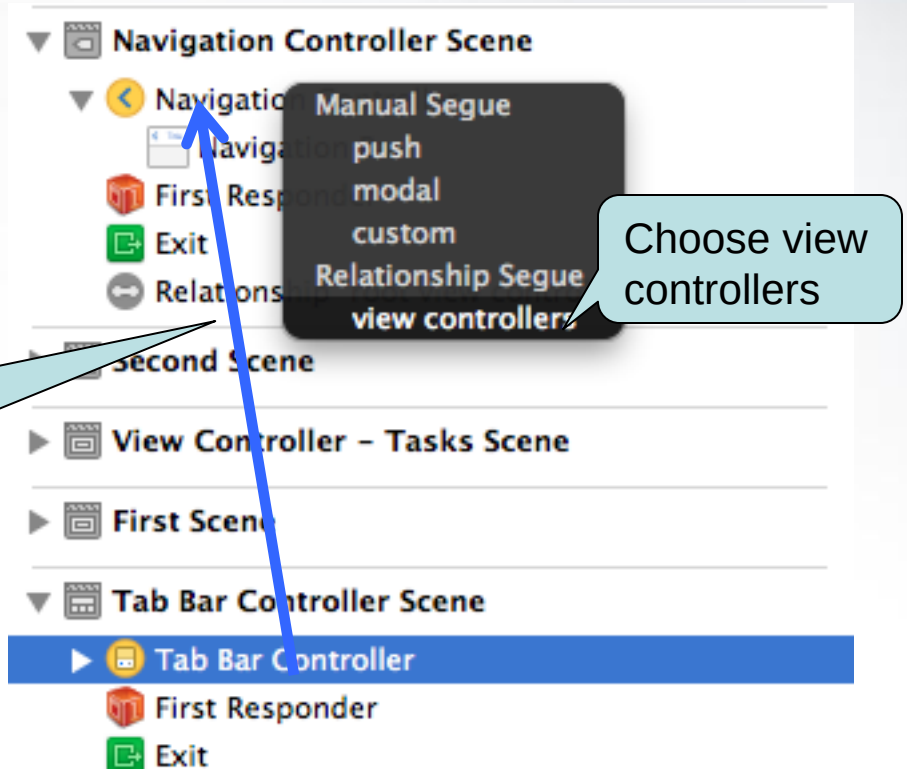
- Technique: drag a Navigation Controller from the Object library onto the storyboard



Note: the Navigation Controller comes with a Table View Controller

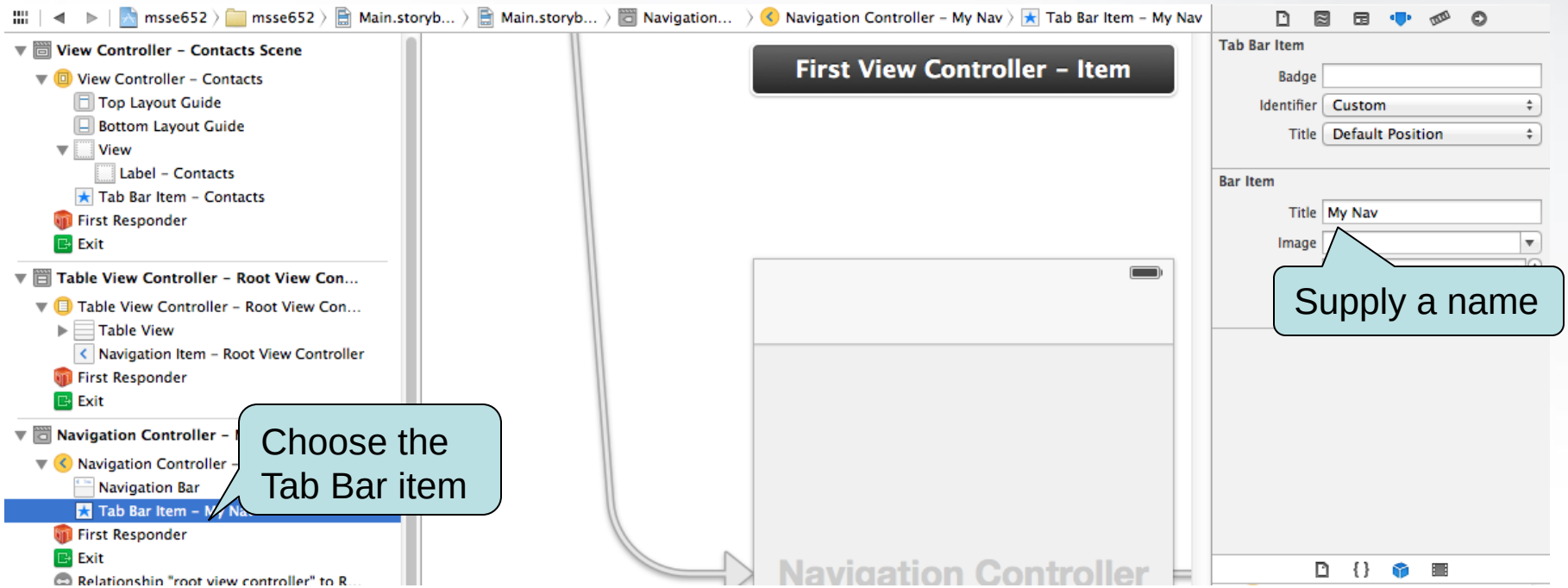
Linking the Nav to the Tab Bar

- In the Document Outline, control drag …
  - from the Tab Bar Controller to the Navigation Controller
- In the popup,
  - choose “view controllers”



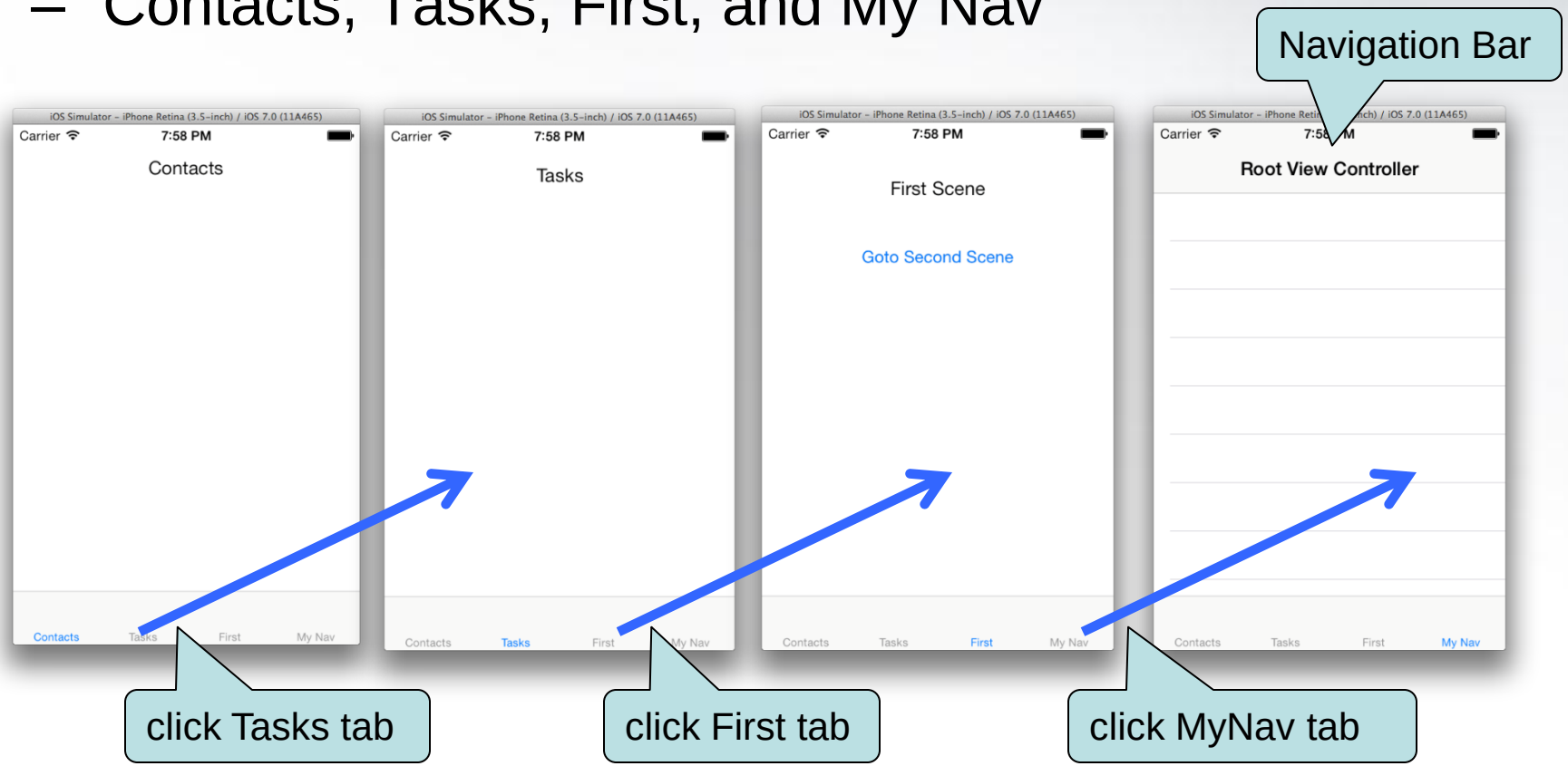
Updating the Tab's name

- Once the segue / tab is added, select it and change its name in the Attributes Inspector



Run the app

- The Tab Bar now has four tabs
  - Contacts, Tasks, First, and My Nav



Hierarchical Information

- Navigation Controllers and Table Views go hand in hand with “hierarchical information”
  - By hierarchical information we mean different levels of information, ranging from …
    - broad, high-level data
      - to more detailed data
        - » to even more detailed data, etc.
- For example, consider …
  - University academic programs (broad information)
    - Courses offered in each program (more detailed information)
      - Individual course information (very detailed information)

Nav Controllers and Table Views

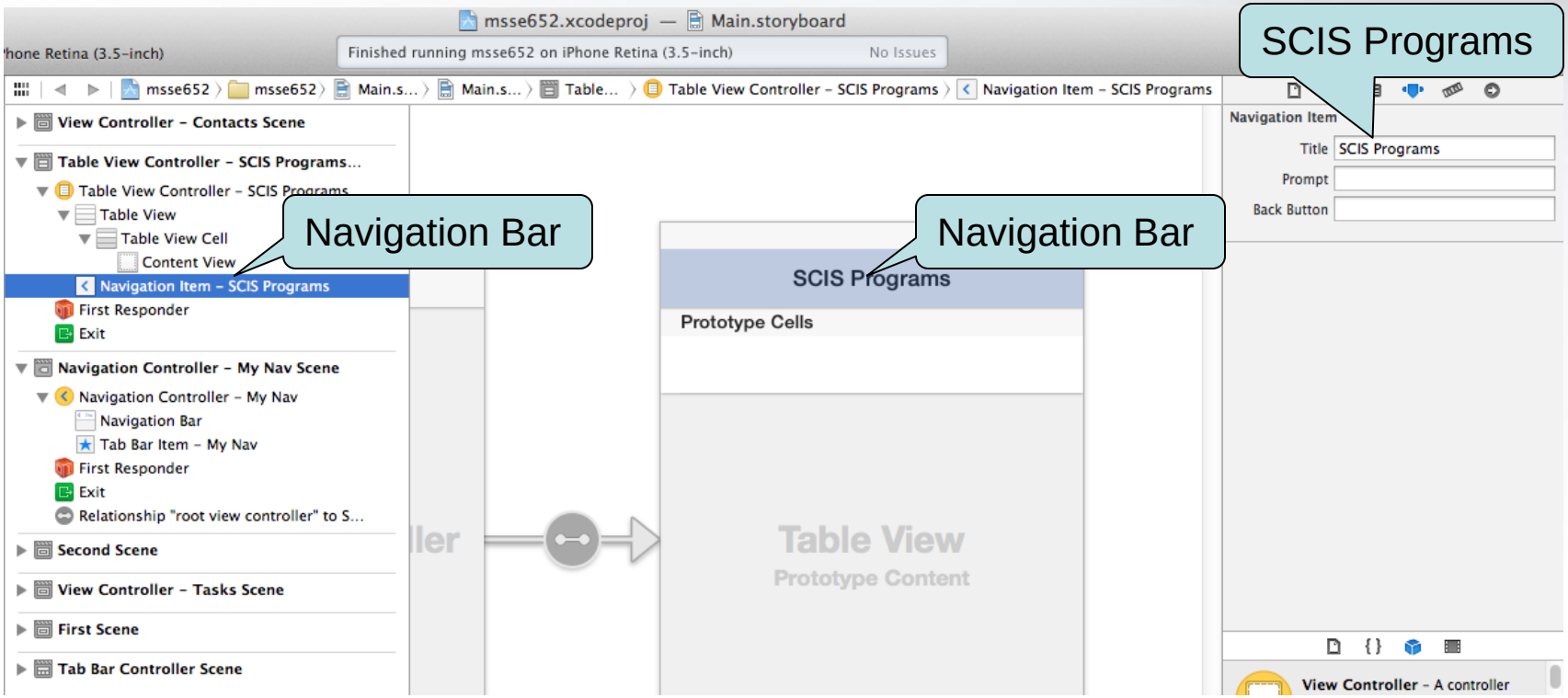
- So, how do Navigation Controllers and Table Views support hierarchical information?
  - Table View Controllers support the display of information at each level
  - Navigation Controllers support the traversal of Table View Controllers
    - i.e., moving between Table View Controllers
- Let’s set the scene’s Nav Bar’s display name …

The Table View Controller's name

- Select the Table View Controller's Navigation Bar, either in …
  - the Document Outline
  - or at the top of the scene in the Storyboard
- Then, in the Attributes Inspector, …
  - update the Title field
    - e.g., with "SCIS Programs" (w/o the quotes)
- For instance …

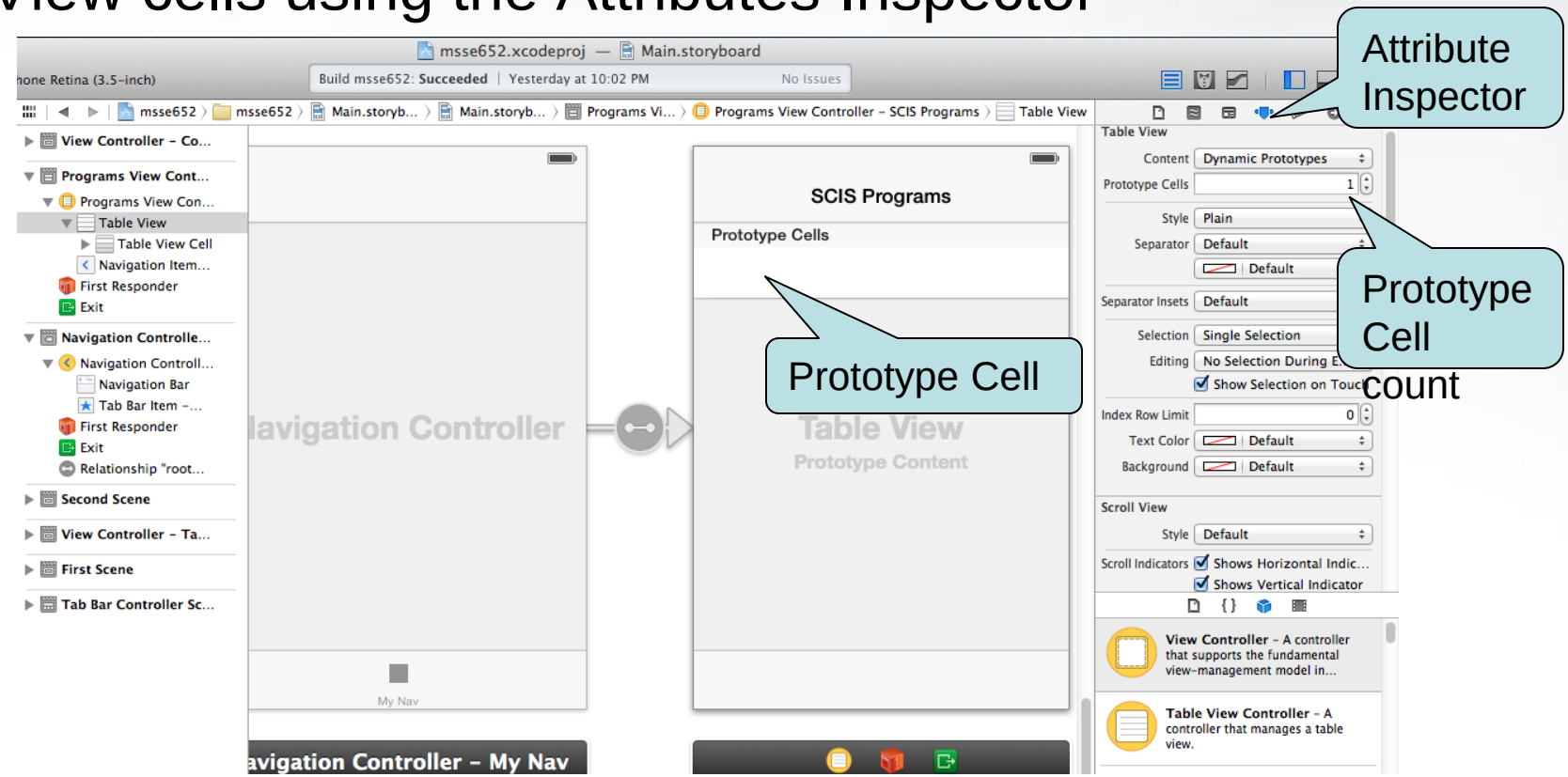
Updating Table View Controller

- “SCIS Programs”



Prototype Cell

- The Storyboard editor supports “Prototype Cell:
  - allows you to customize the appearance of the Table View cells using the Attributes Inspector



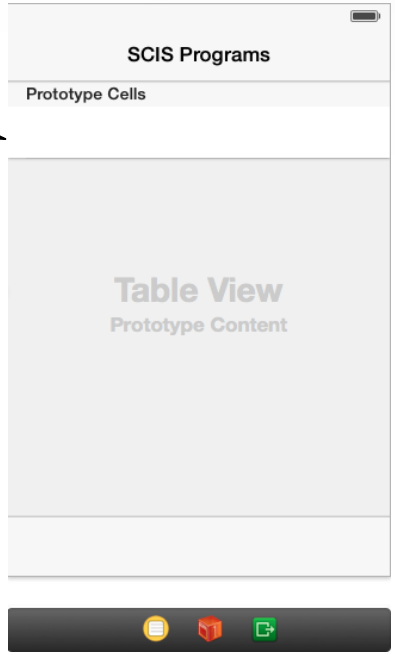
Prototype Cell (cont'd)

- The Prototype Cell count specifies the number of prototype cells
  - The Nav Bar controller's Table View has the prototype cell count set to 1
    - so there is 1 Prototype cell (duh)
  - As an experiment, set the count to 2, and then 0
    - and notice what happens to the table view
  - And then set the count back to 1

Prototype Cell (cont'd)

- Note, when the count is non zero,
  - the storyboard editor displays a Prototype Cell
    - so you can see it's "look and feel"
  - but when you run the app, it's not displayed

Prototype cell appears in editor



Does not appear when app runs

Modifying the Prototype Cell

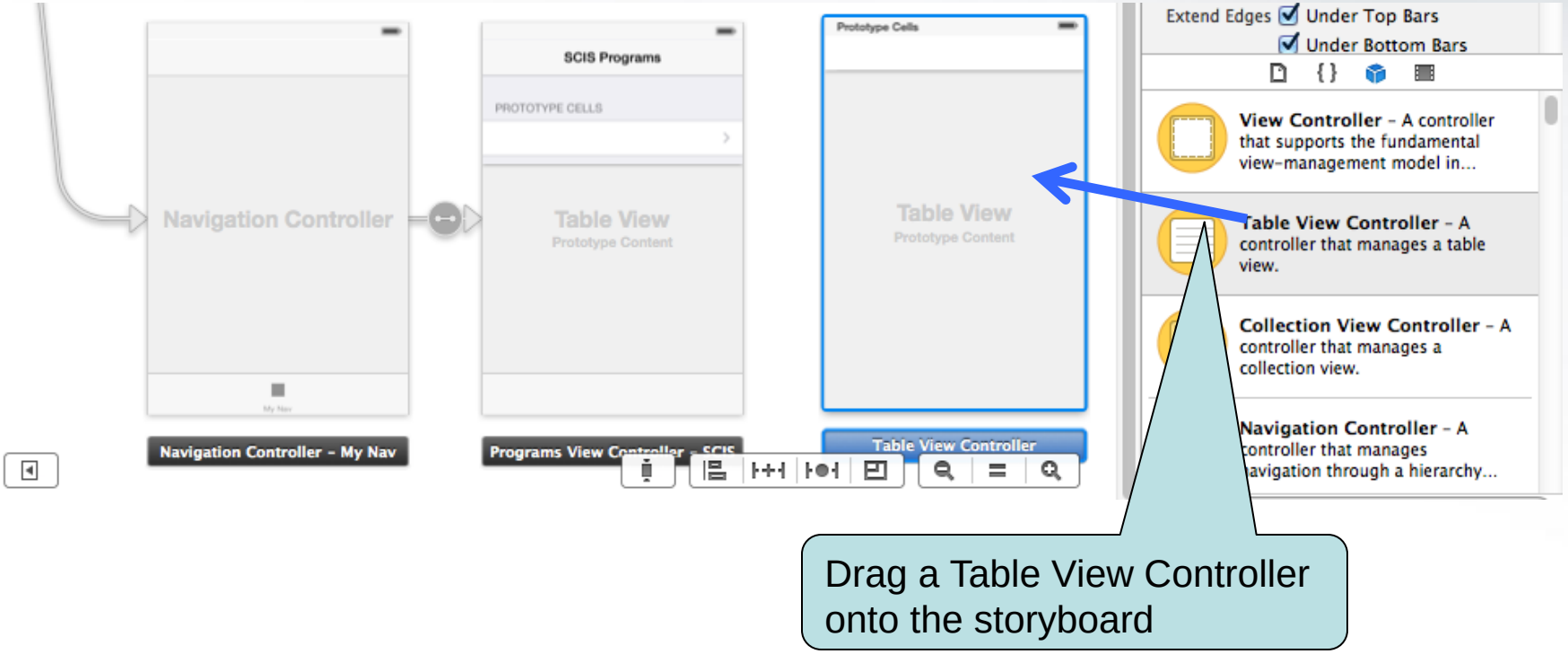
- With the count set to 1, let's make the following change
  - In the storyboard, select the prototype cell, and change the following item:
    - Accessory: Disclosure Indicator
  - The prototype cell is updated with an arrow ">" to indicate that tapping the cell reveals more information
    - i.e., more detailed information

Creating a “detail” screen/scene

- We will now add another scene that provides the details of the tapped item
  - i.e., the courses associated with the selected SCIS program
- From the Object library in the Utilities pane,
  - drag a Table View Controller onto the storyboard
  - for instance …

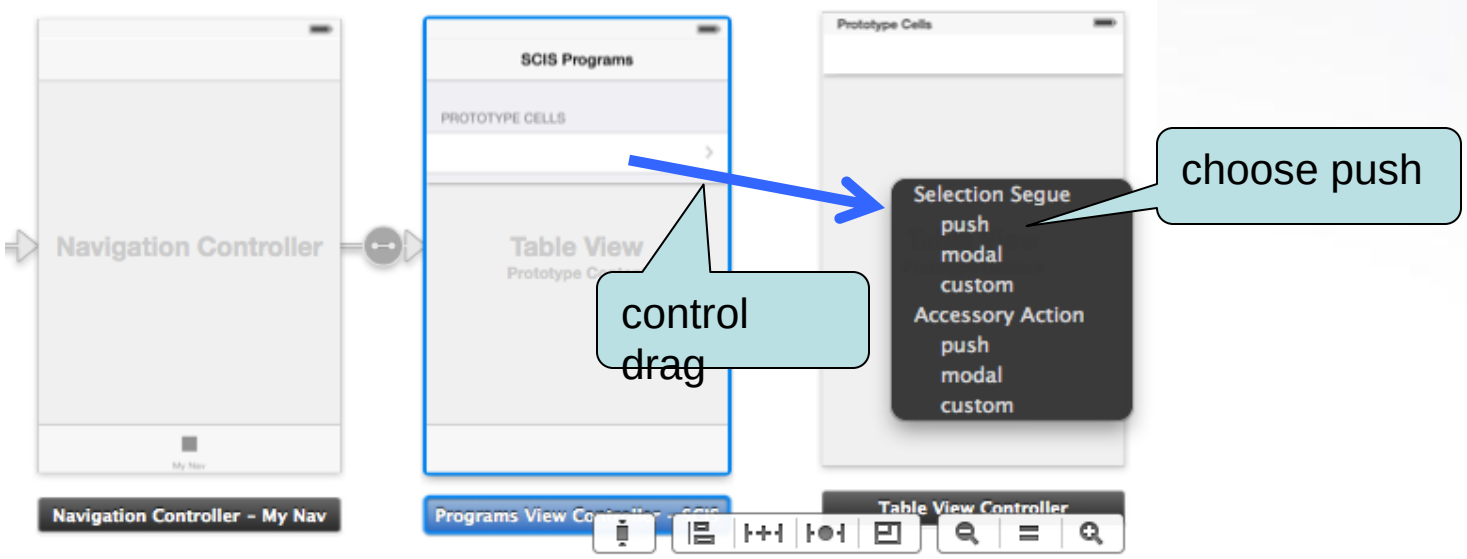
The detail Table View Controller

- Add a Table View Controller to the storyboard



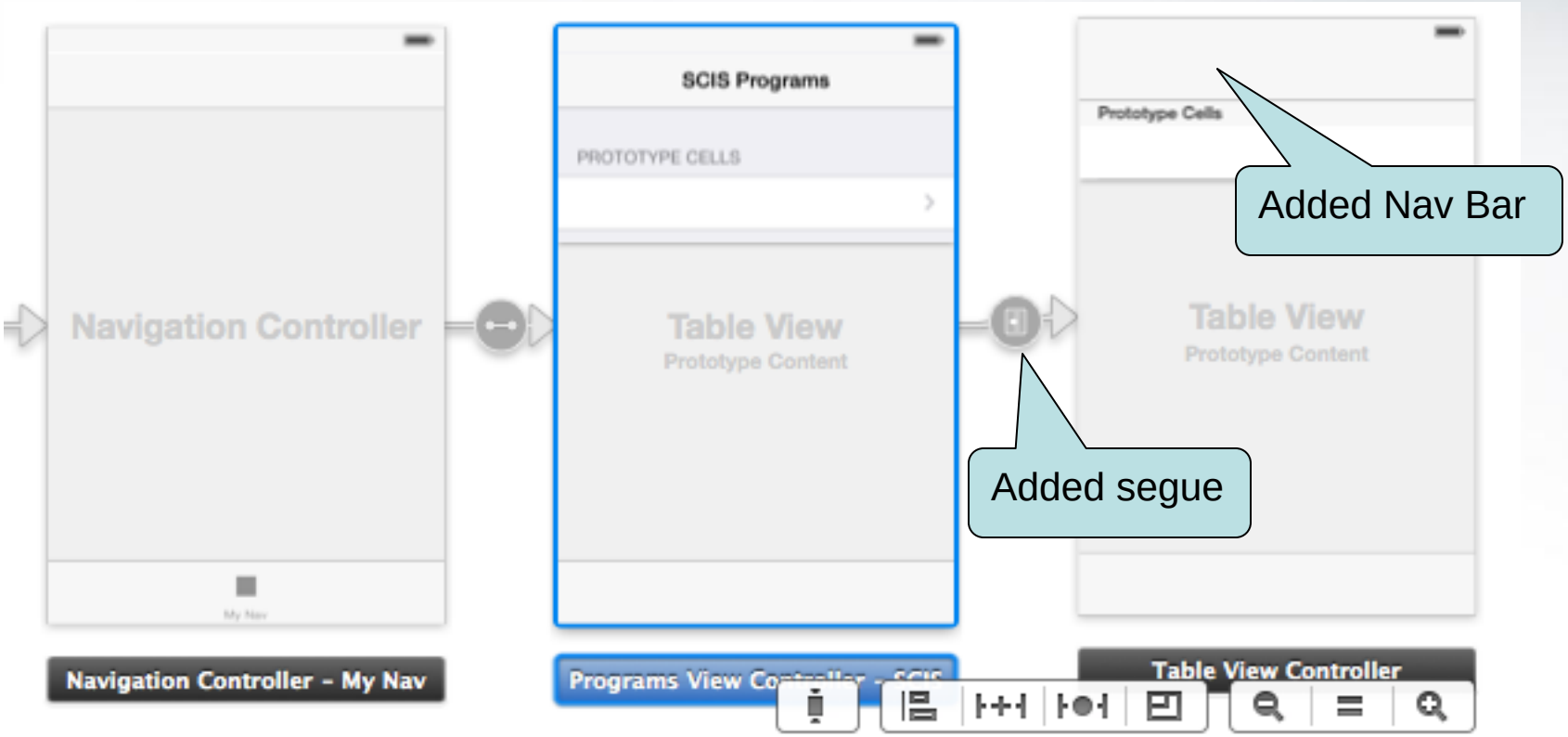
Adding a segue

- To facilitate navigation, we add a segue …
  - from the first table view to the second table view
- Technique:
  - control-drag …
    - from the first table's prototype cell to the second table view



Updated Storyboard

- A segue is added, and a Nav Bar is added too

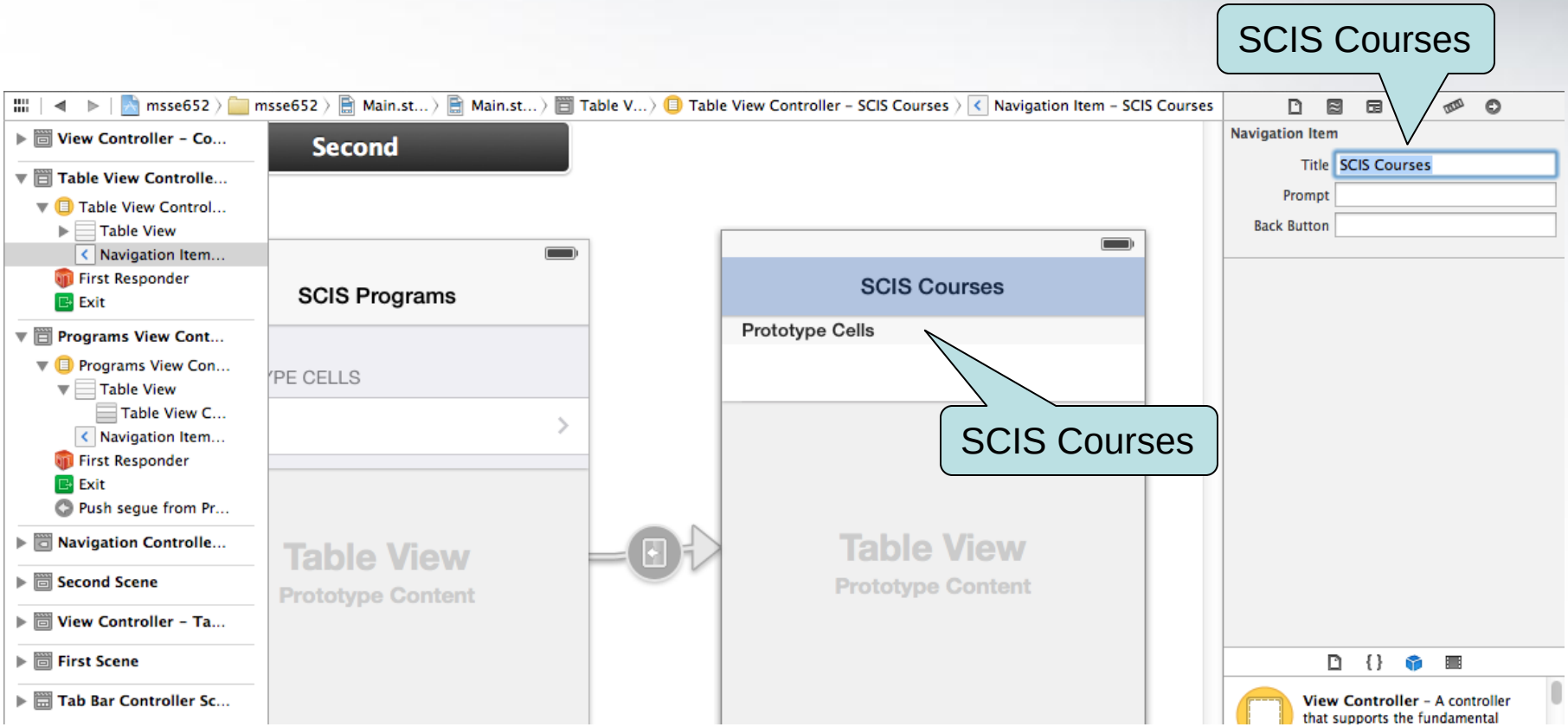


Setting the Nav Bar name

- Using the technique we used before …
  - Select the new Table View Controller's Navigation Bar, either in …
    - the Document Outline
    - or at the top of the scene in the Storyboard
  - Then, in the Attributes Inspector, …
    - update the Title field
      - e.g., with "SCIS Courses" (w/o the quotes)
- For instance …

Setting the Nav Bar name (cont'd)

- Nav Bar updated with "SCIS Courses"



Now run the app

- You can now traverse back and forth between the two views ☺