

iOS Apprentice Review 2

Part 3: Lab Instructions

Copyright © 2014 Razeware LLC.

All rights reserved. No part of this book or corresponding materials (such as text, images, or source code) may be reproduced or distributed by any means without prior written per- mission of the copyright owner.

This book and all corresponding materials (such as source code) are provided on an "as is" basis, without warranty of any kind, express or implied, including but not limited to the warranties of merchantability, fitness for a particular purpose, and noninfringement. In no event shall the authors or copyright holders be liable for any claim, damages or other liability, whether in action of contract, tort or otherwise, arising from, out of or in connec- tion with the software or the use or other dealings in the software.

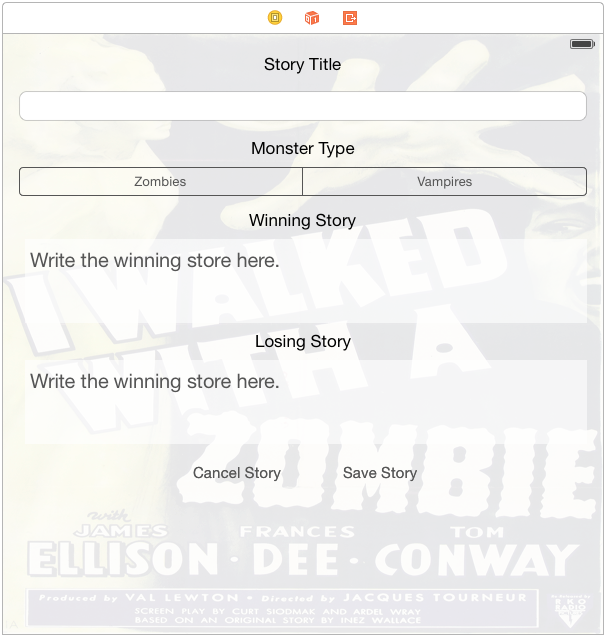
All trademarks and registered trademarks appearing in this book are the property of their respective owners.

You’ve seen how to take a Single View app and covert it into a multi screen app, passing data between each of the View Controllers. In this lab, you’re going to expand on this.

The app, in its current state, is basically a read-only. What happens if your users want to write their own story? In this Lab, you’ll create a new View Controller that will allow users to do just that.

# Part 1: Setup the Controller

A view controller has already been created for you. If you look in your storyboard, you should seen an view controller that looks like such:



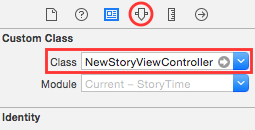
This view controller allows you to create a new story by providing a title, a winning story, and a losing story.

## Step 1: Create the Controller File

Select **File / New / File.** Select **Source,** then choose **Cocoa Touch Class.** For the class name, write **NewStoryViewController**. Make sure it is a subclass of a UIViewController and that the Language is **Swift**. Click **Next**, then **Create**.

## Step 2: Add the Outlets

Open **Main.storyboard**. Select your new View Controller, and in the Identity inspector, set the class to be **NewStoryViewController.**



Next, open the Assistant Editor, and create IBOutlets for the following controls: storyTitle (Text Field), monsterType (Segmented Control), winningStory (Text View), losingStory (Text View), backgroundImage (Image View).

**Note:** Forgot how set up an IBOutlet? First, open your storyboard, then switch to your Assistant Editor. Select the control and control drag down to your source code. Release the button, and provide a name for the outlet.

## Step 3: Add Some Properties

Switch back to **NewStoryViewController.swift**. Underneath your IBOutlets, add the following properites:

var newStory: Story?

var didCancel: Bool = false

## Step 4: Create Some IBActions

Typically, you create your IBActions in your storyboard and then fill out the bodies in code. In these examples, you’ll do the reverse. With **NewStoryViewController.swift** open, add the following IBAction that saves the changes made by the user:

@IBAction func save(sender: AnyObject) {

if !storyTitle.text.isEmpty && !winningStory.text.isEmpty &&

!losingStory.text.isEmpty {

var storyType: StoryType!

if monsterType.selectedSegmentIndex == 0 {

storyType = StoryType.zombies

} else {

storyType = StoryType.vampires

}

newStory = Story(title: storyTitle.text, winStory: winningStory.text, loseStory: losingStory.text, type: storyType)

performSegueWithIdentifier("DismissNewStory", sender: nil)

} else {

let alert = UIAlertController(title: "Validation Error", message: "Please fill out all of the fields.", preferredStyle: UIAlertControllerStyle.Alert)

let alertAction = UIAlertAction(title: "OK", style: .Default, handler:nil)

alert.addAction(alertAction)

presentViewController(alert, animated: false, completion:nil)

}

}

Notice that you are calling a new Segue? Don’t worry, you’ll add it in just a second. Now add the cancel action.

@IBAction func cancelStory(sender: AnyObject!) {

didCancel = true

performSegueWithIdentifier("DismissNewStory", sender: nil)

}

Now to change the background image when the user selects a different story type.

@IBAction func changeMonsterType(sender: UISegmentedControl!) {

if sender.selectedSegmentIndex == 0 {

backgroundImage.image = UIImage(named: "zombies")

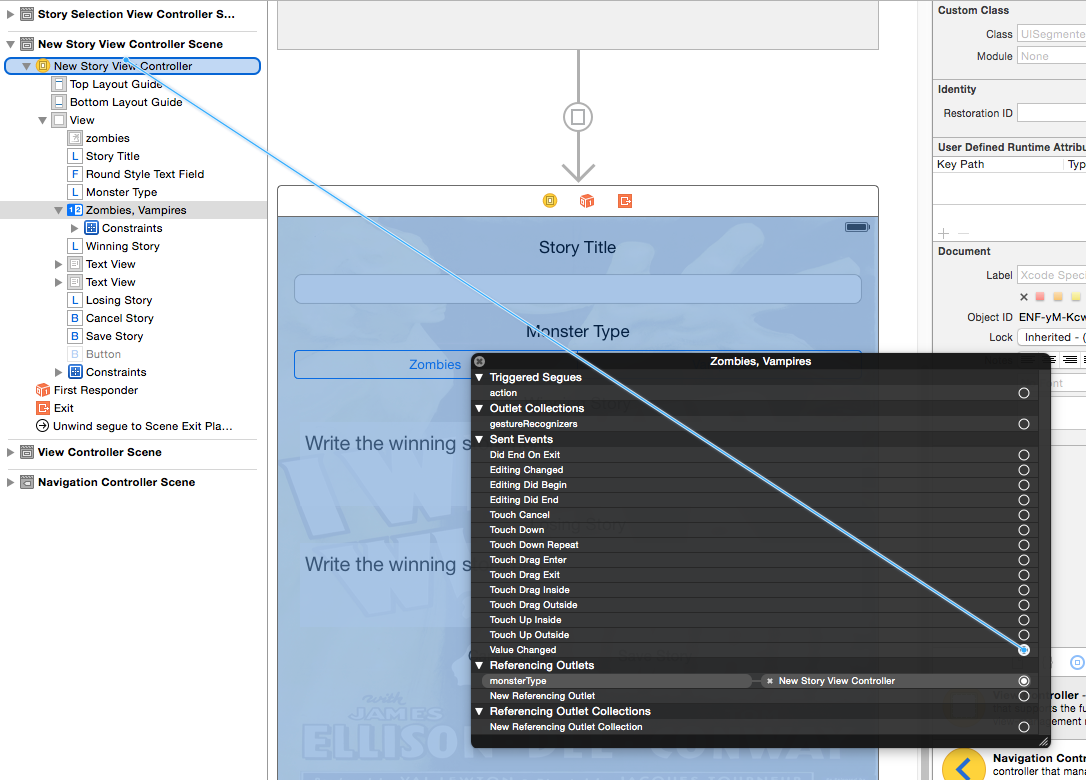
} else {

backgroundImage.image = UIImage(named: "vampires")

}

}

With all the IBActions created, switch back to **Main.storyboard**. Select the Segmented Control, and control click it. You should see a list of actions. At the end of the **Value Changed** action, drag the circle to the main New Story View Controller.



From the resulting dropdown, select the **changeMonsterType:** action. For the cancel button, control click and drag from **Touch Up Inside**. Select **cancelStory:**. Finally, for the save button, control drag from **Touch Up Inside** and select **save:**.

## Step 5: Dismissing the Keyboard

While this new view controller works, users will not be able to dimiss the keyboard. Switch back to **NewStoryViewController.swift**, and add the following code in viewDidLoad():

let tapGesutureRecognizer = UITapGestureRecognizer(target: self, action: "dismissKeyboard")

view.addGestureRecognizer(tapGesutureRecognizer)

Gesture recognizers make sure a great way handle to touch controls. In this case, a simple tap gesture will call dismissKeyboard(). Add the following method:

func dismissKeyboard() {

view.endEditing(true)

}

This method forces any open keyboards to resign their responder status.

Finally, you need to add code to dismiss the keyboard when the user taps the return key.

At the top of the file**,** change the class definition to look like the following:

class NewStoryViewController: UIViewController, UITextFieldDelegate, UITextViewDelegate {

Now, add the following mthods:

func textViewShouldEndEditing(textView: UITextView) -> Bool {

textView.resignFirstResponder()

return true

}

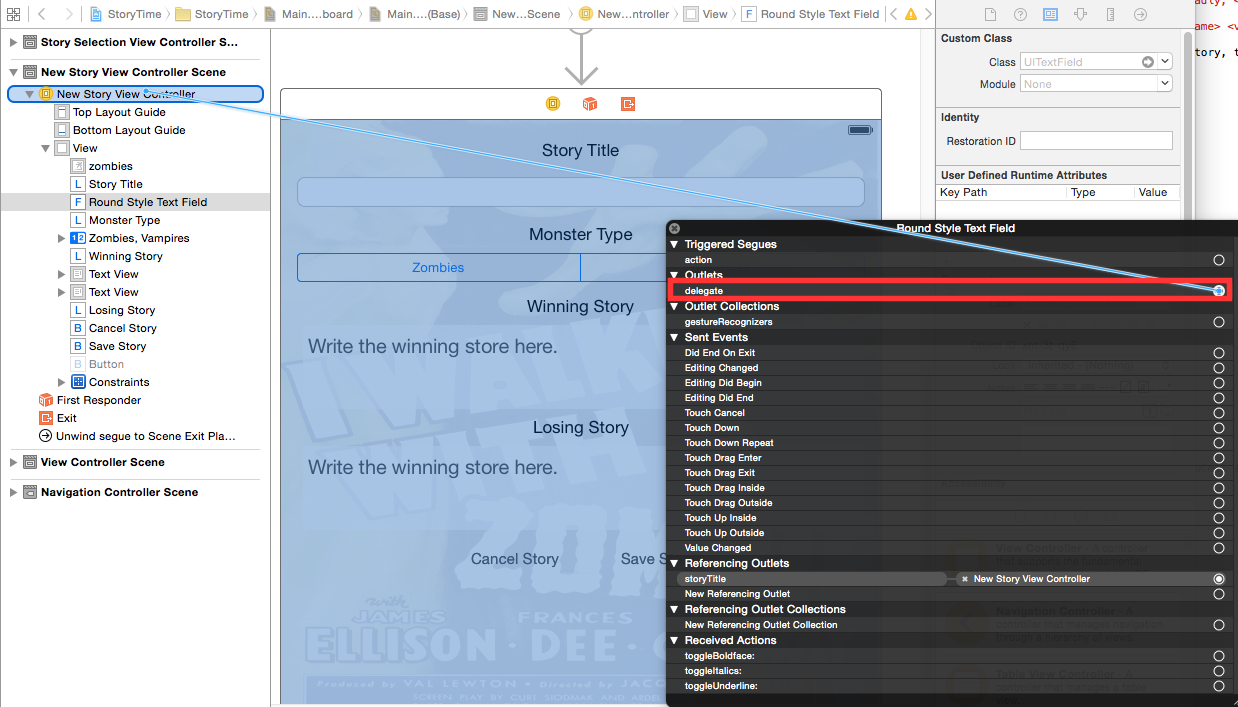
func textFieldShouldReturn(textField: UITextField) -> Bool {

textField.resignFirstResponder()

return true

}

Open **Main.storyboard** and control click on the first text field. Under the list of Outlets, look for the item labeled **delegate**. Drag from the circle all the way to the New Story View Controller.



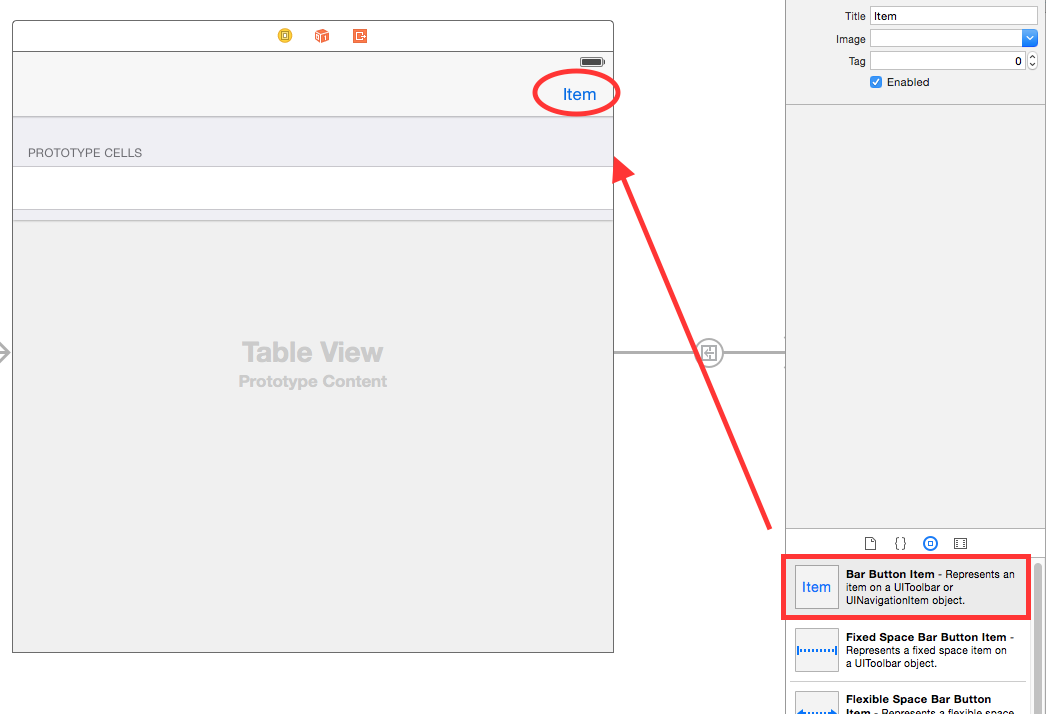
Do the same for both of the text views.

# Part 3: Tying it all Together

Now that your view controller is all set, it’s time to hook it into the storyboard.

## Step 1: Add a Bar Button Item

Open **Main.storyboard** and in the object library, look for a Bar Button Item. Drag it to the navigation bar of the Story Selection View Controller.



With the bar button selected, open the Attributes Inspector and change Identifier to **Add**.

Select the bar button item, and control click from it to the New Story View Controller. Select the **Present Modally** option. Next, select the segue and in the Attributes Inspector, give it the Identifier: **NewStory.**

## Step 2: Create the Unwind Segue

An unwind segue is used when you want to return to a previous view controller. In your case, you presented the New Story View Controller and once the user has created that story, you need a way to get it.

Open **StorySelectionViewController.swift** and add the following code:

@IBAction func close(segue:UIStoryboardSegue) {

let newStoryViewController = segue.sourceViewController as

NewStoryViewController

if !newStoryViewController.didCancel {

if let story = newStoryViewController.newStory {

if story.type == .zombies {

zombieStories.append(story)

} else {

vampireStories.append(story)

}

tableView.reloadData()

}

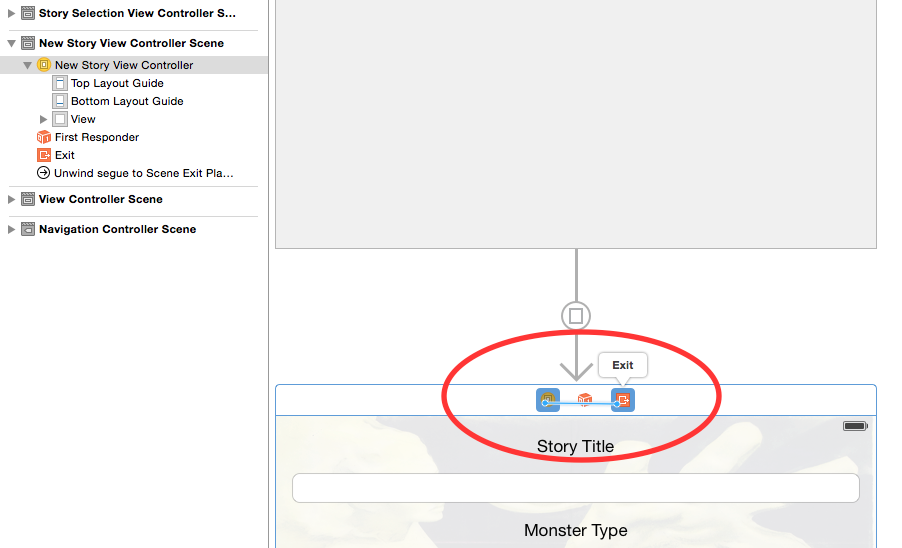
}

}

Once have the segue added, make sure to **build** your project before moving to the next step.

## Step 3: Hook Up the Unwind Segue

Open **Main.storyboard**. In the New Story View Controller, select the yellow View Controller icon and control drag over to the last red icon.



From the dropdown, choose the close: segue. Congratulations! You made it through the lab and have a working app!

## 