ATLS 4120/5120: Mobile Application Development
Week 9: Multiple View Controllers

Although we could build this in one view, in this app we're going to learn how to handle multiple views as well as store model data used in both views.

Favorites
Create a new project using the single view template called favorites.
Add a label on top that says Favorites.

Put a toolbar at the bottom. Add a flexible space bar to move the button over to the right.
Give the button the name Info.
We want to keep the toolbar at the bottom of the view and going all the way across the bottom no matter what size the view has so create pin constraints for leading(-16), trailing(-16), and bottom(8).

Now let's add a new scene to the storyboard. Drag a view controller object from the Object Library panel onto the canvas to the right of your first scene. It automatically gets selected.
Go into the Identity Inspector and see that its class is listed as UIViewController. But if we want to write any code for this view, we will need to create our own class to control it.
File | New | File
iOS | Source | Cocoa Touch class named Scene2ViewContoller.
Make sure the Subclass of menu is set to UIViewController.
Don't select Also create XIB file
Make sure the language is Swift.
Save it in Favorites.
This creates the Scene2ViewContoller.swift
You can drag it into favorites if you want to remain organized.
Go back into the MainStoryboard and select scene2's view controller button. In the identity inspector change the class to be Scene2ViewContoller.
This is a very important step as it determines that the Scene2ViewContoller will control the second view.

Now add a label to the top of this scene that says My Favorites.
Add a toolbar, a flexible space bar to move the button to the right, and a button labeled Done.
Add the pin constraint for leading, trailing, and bottom, same as before.

We can have more scenes but for today we're just going to have two.
As more scenes are added to a storyboard, it becomes increasingly difficult to see more than a few scenes at one time on the canvas. You can zoom by double clicking, right click for a menu, or pinching on the trackpad. Note that when zoomed out, it will not be possible to drag and drop items from the Object Library onto the scenes.

Now let's configure the segues between these two scenes.

Select the first scene. To setup a segue, hold down the Ctrl key, click on the toolbar button (make sure you get the bar button item and not the toolbar) and drag the resulting line to the second scene. When you let go of the mouse button a menu will appear. Select the present modally menu option to establish the segue. Click on the segue to see that it's present modally. Look in the attributes inspector to see that Animates is checked and there are 4 other transitions other than default. Try these later.

Go into the attributes inspector to give it the identifier favInfo.

Save and run and you'll see that the Info button takes you to the second scene, but you can't go back yet. Let's set that up now.

Although you could just set up another segue from the done button back to scene1, it's better to set up an unwind action.
First you need to add a method to scene1 (destination view controller).
In ViewController.swift add the method

```
    @IBAction func unwindSegue (segue:UIStoryboardSegue){
    }
```

It's ok that it's empty for now. We'd put any code in here we want to run when we return from scene 2 to 1.

This method returns an IBAction and takes in a UIStoryboardSegue. Using this method format will tell Xcode that this is a method that can be *unwound* to.

The next step is to establish the unwind segue. Go back into MainStoryboard and select scene 2. ctrl-click and drag from the Done button (make sure you get the bar button item and not the toolbar) to the "exit" icon (the orange button with the white square and arrow) at the top of the view. Release the line and select the unwindSegue method from the resulting menu.
Click on that segue and go into the attributes inspector to give it the identifier doneFavs.

Run your app and see that now the Done button takes you back to scene 1.

Now let's get some data and pass it between scenes.

In scene 1 add two labels, make them blank, and create outlets named bookLabel and authorLabel respectively. Make sure you make these connections to the right class – ViewController.

In scene 2 add a label for favorite book and a text field next to it.
Do the same for author.

Connect the textfields as outlets names userBook and userAuthor respectively. Make sure you make these connections to the right class – Scene2ViewController.
We also want to make sure the keyboard is dismissed for both textfields when the user hits return.
In Scene2ViewContoller.swift adopt the `UITextFieldDelegate` protocol

```
class Scene2ViewController: UIViewController, UITextFieldDelegate
```

Set the textfield's delegates

```
    override func viewDidLoad() {
        userBook.delegate=self
        userAuthor.delegate=self
        super.viewDidLoad()
    }
```

Implement `textFieldShouldReturn`

```
    func textFieldShouldReturn(textField: UITextField) -> Bool {
        textField.resignFirstResponder()
        return true
    }
```

Model data

Now we’re going to create the Favorite model class to keep track of the data.
File | New File
iOS Source Swift file
Name it Favorite
Add it to the favorites folder and make sure the target is checked as well.
Create
Let’s define the Favorite class to store the user’s favorite book and author
Edit Favorite.swift

```
class Favorite {
    var favBook : String?
    var favAuthor : String?
}
```

Swift will create an initializer method for us and both Strings will have the value nil or we can create our own initializer method if we want it to have an initial value. (in the class braces)

```
    init(){
        favBook=”Your favorite book”
        favAuthor=”Your favorite author”
    }
```

I’m not going to do this since I want the user to enter their favorites.

Now let’s create an instance for the Favorite class to ViewController.swift called user

```
    var user=Favorite()
```

In Scene2ViewController.swift we need to save the data the user enters back to our model. The prepareForSegue method is called automatically when a segue is about to appear.

```
override func prepareForSegue(segue: UIStoryboardSegue, sender: AnyObject?)
{
    if segue.identifier == "doneFavs"{
           var scene1ViewController:ViewController =
segue.destinationViewController as! ViewController
           //check to see that text was entered in the textfields
           if userBook.text.isEmpty == false{
                scene1ViewController.user.favBook=userBook.text
           }
           if userAuthor.text.isEmpty == false{
                scene1ViewController.user.favAuthor=userAuthor.text
           }
    }
}
```

In ViewController.swift we need to update our labels when we return from scene 2.

```
    @IBAction func unwindSegue (segue:UIStoryboardSegue){
        bookLabel.text=user.favBook
        authorLabel.text=user.favAuthor
    }
```

Snapshot: final3

Constraints
Scene 1:
Both labels align horizontal center
Top label pin to Favorites label
Bottom label pin to above label

Scene 2:
Pin the book label for the top and leading
Pin the author label bottom to leading and top to book label
Pin equal width for labels
Pin the textfields for their trailing space
Pint the textfields leading space to their label =25
Align baselines of text field and its label
Pin equal width for textfields