

iOS Foundations II

Day 4

- Homework Review
- Navigation Controller
- Segues
- 'Passing' objects through segues

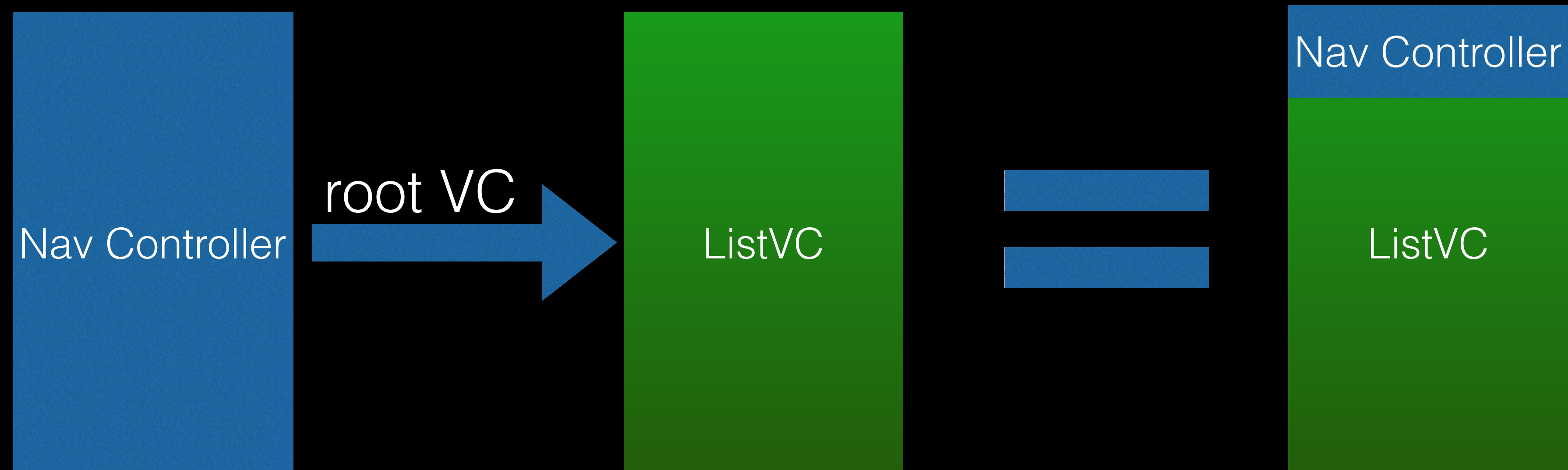
NavigationControllers



- “A navigation controller manages a stack of view controllers to provide a drill-down interface for hierarchical content.”

NavigationControllers

- “The navigation controller’s primary responsibility is to respond to user actions by pushing new content view controllers onto the stack or popping content view controllers off of the stack”
- The first ViewController you push onto the stack becomes the rootViewController and is never popped off because then no view would be on screen
- Nav Controllers have a property to the topViewController and an array property for all its viewControllers currently on the stack.
- The Nav bar up top can be customized or hidden.



A Navigation controller is always instantiated with a root view controller. This will be the first view controller you see. A view controller contained within a navigation controller has a navigation bar at the top

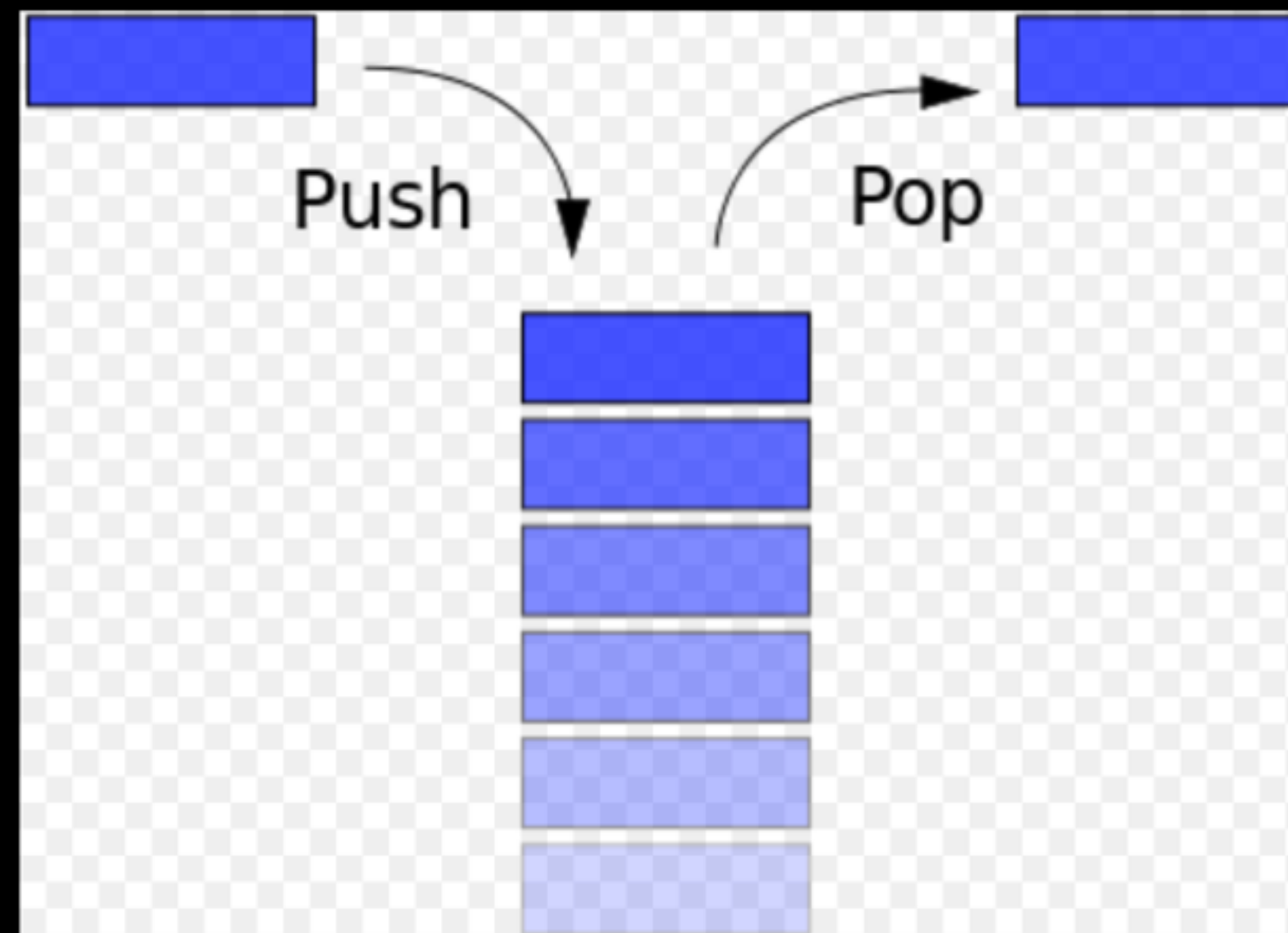
NavigationControllers

- 2 ways to get navigation controllers into your app:
 - Via storyboard, selected the view controller you want to set as the root view controller and Go up to the top menu, Editor> Embed In> Navigation Controller
 - In Code, you can instantiate an instance of UINavigationController and give it a root view controller

Demo

Stack Data Structure

- “A stack is particular kind of abstract data type or collection in which the only operations on the collection is adding (push) or removal (pop).” - Wikipedia
- LIFO : Last-In-First-Out. The last item added is the first to be removed.



NavigationControllers

- Storyboards make navigation controllers extremely easy to install into your app. Here's all the methods you need to do it in code without the storyboard:
- `init(rootViewController:)` UINavigationController is initialized with a rootViewController.
- `pushViewController(animated:)` To add or 'push' a view controller onto the stack.
- `popViewController(animated:)` To remove or 'pop' a view controller from the stack.

NavigationControllers and View Controllers

- **UIViewController provides a property called navigationController, that points to a navigation controller, if the view controller is actually inside a navigation controller.**
- **It will be nil if there is no navigation controller present.**

Demo

Segues

- Segues are **provided by the storyboard** to help you easily transition from one view controller to another.
- They are pronounced “SAYG-WAY”, not “SEEG”
- There are 2 primary segues that are used : Show and Present
- Show refers to pushing a view controller onto the navigation stack, it usually will slide in from the right to the left
- Present refers to modally presenting a view controller, it usually slides up from the bottom. Modal presentations are usually ‘one-off’ screens, like settings or quick uploads.
- You can create your own custom segue to customize the behavior to match your needs.

Creating Segues

- Segues are always created via storyboard.
- You cannot create a segue via code, but you can trigger a segue via code (more on this in a bit)
- If you do an app without storyboards, there will be no segues.
- That doesn't mean you are trapped in one view controller, transitions can be done in code without segues.
- Creating segues is just like creating outlets and actions, lots of dragging and hoping you dragged to the right thing

Demo

Segues in Code

- 2 primary methods for dealing with segues in code:
 1. `performSegueWithIdentifier:` triggers a segue manually in code
 2. `prepareForSegue:sender:` allows you to run some code before the segue actually fires. You don't call this method yourself, you just implement it in your view controller, it is called by the system.

performSegueWithIdentifier

- Call this method in your View Controller to trigger a segue in code.
- If you your segue isn't hooked up to be triggered by an action in your interface, you will need to call this method to trigger the segue.

prepareForSegue:sender:

- This method is called on the source view controller of the segue, right before the segue is actually performed.
- The first parameter is the segue itself, which is an instance of the UIStoryboardSegue class.
- The most important property of a UIStoryboardSegue instance is the destinationViewController property, which gives you a reference to the view controller you are about to segue to.
- This is a great spot to pass information to the next screen.

Demo

Passing data to View Controllers

- There are many patterns we can use to pass data around our app:
 - Delegation
 - Notification Center
 - Singletons
- But for now, we can do something as simple as passing a reference directly to the new view controller.

Passing data to View Controllers

ListViewController

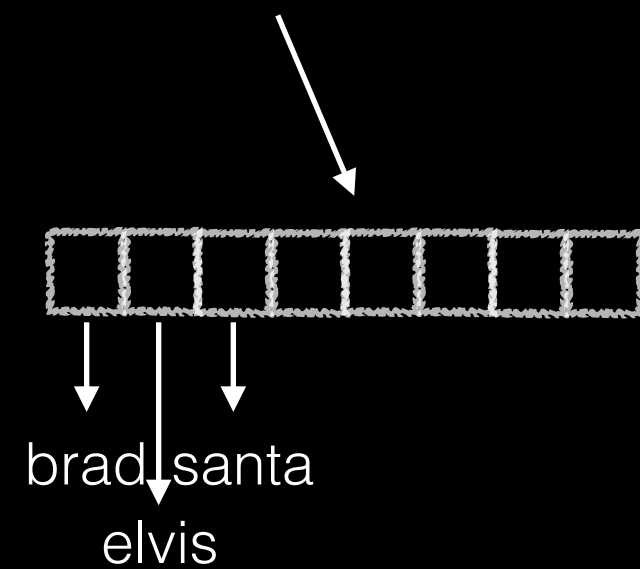


ListViewController creates an array of Person objects and sets its people property's value to that array

ListViewController



Self.people

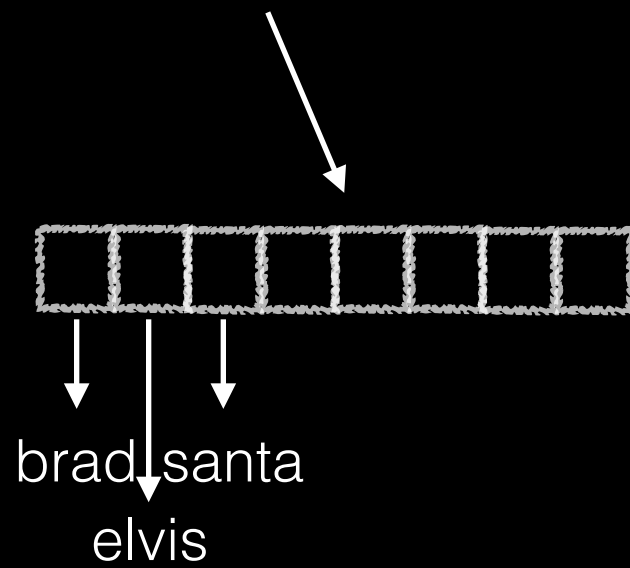


ListViewController, as the datasource of the tableview, uses the self.people array as the backing array for the tableview

ListViewController

brad
elvis
santa
Russell
Sherman
Clem

Self.people

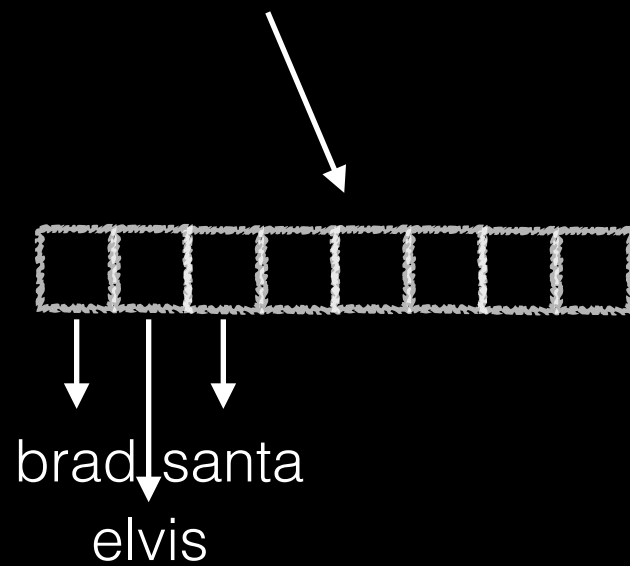


The User clicks on a cell, triggering our ShowPerson Segue. The PersonDetailViewController is initialized.

ListViewController

brad
elvis
santa
Russell
Sherman
Clem

Self.people



PersonDetailViewController



Self.selectedPerson

Show Person



In Prepare for Segue, the ListViewController intercepts the segue and grabs a reference to the destination view controller, which is the PersonDetailViewController.

```
override func prepareForSegue(segue: UIStoryboardSegue!, sender: AnyObject!) {  
    if segue.identifier == "ShowPerson" {  
        var personDetailViewController = segue.destinationViewController as  
        PersonDetailViewController  
  
        //now we can prepare this view controller to be displayed with what  
        ever data we need to pass to it.  
    }  
}
```

ListViewController sets PersonDetailViewController's selectedPerson property to reference the person that was clicked from the list.

ListViewController

PersonDetailViewController

brad
elvis
santa
Russell
Sherman
Clem

Show Person



Self.people

Self.selectedPerson



brad
santa
elvis

ListViewController sets PersonDetailViewController's selectedPerson property to reference the person that was clicked from the list.

```
if segue.identifier == "ShowPerson" {  
  
    var personDetailViewController = segue.destinationViewController as  
    PersonDetailViewController  
  
    //now we can prepare this view controller to be displayed with what  
    ever data we need to pass to it.  
  
    //grab the selected index path from our tableview  
    var selectedIndexPath = self.tableView.indexPathForSelectedRow()  
  
    //grab the selected person using the indexPath as the index in our  
    people array  
    var selectedPerson = self.people[selectedIndexPath.row]  
  
    //set destinationViewController's person property to reference the  
    selectedPerson  
    personDetailViewController.person = selectedPerson  
  
}
```


Now `PersonDetailViewController` can use its `selectedPerson` property to fill out his interface with that `Person`'s name properties. Any changes he makes to this `Person` object apply to the person object inside the original array!

`ListViewController`

brad
elvis
santa
Russell
Sherman
Clem

`Self.people`

`PersonDetailViewController`

elvis
presley

`Self.selectedPerson`

