CS 326
UITextFields

Stephen Freund

UITextField

- Like UILabel but editable
- Current contents in **`text`** property.
- When tapped
  - text field becomes "first responder"
  - pops up keyboard
  - keyboard goes away when you tell text field to **`resignFirstResponder()`**
- Make sure your text field isn't hidden by keyboard!
  - scroll it into view
  - or just keep it near the top of the window...

UITextField

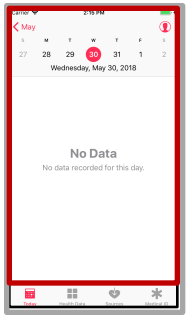
- UITextFieldDelegate
  - **`textFieldShouldReturn(sender:)->Bool`**
    - returns true if you want normal processing to happen.
    - good place to call **`resignFirstResponder()`**
    - eg: target/action hooked up in Storyboard
  - See docs for other methods
- Lots of properties (check out in Storyboard)
  - type of keyboard
  - name of return key
  - auto-correction, capitalization
  - ...

CS 326
Multiple MVCs

Stephen Freund

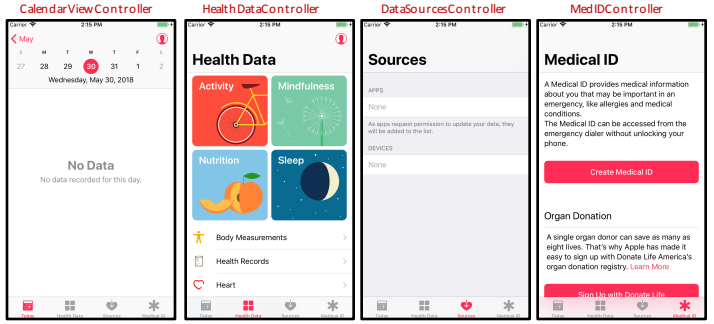
UITabBarController

CalendarViewController

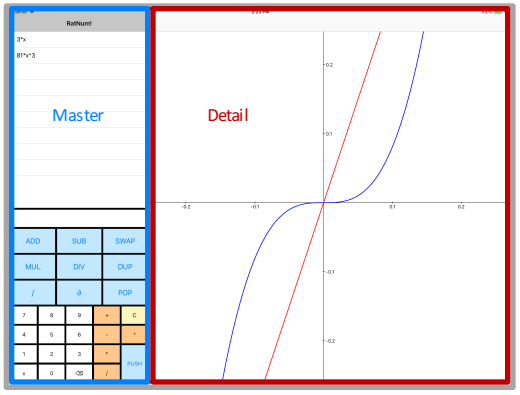


UITabBarController

UITabBarController

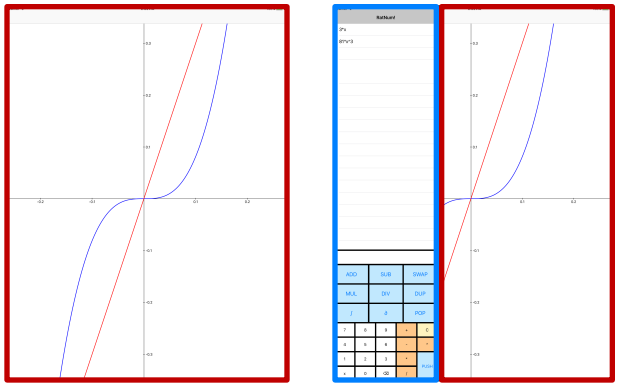


UISplitViewController



landscape

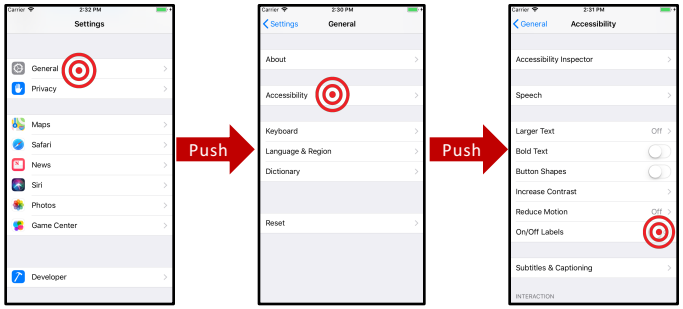
UISplitViewController



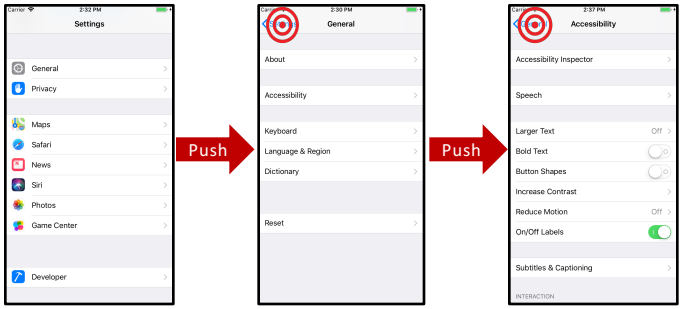
Swipe Left To Show Master

portrait

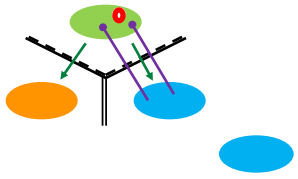
UINavigationController



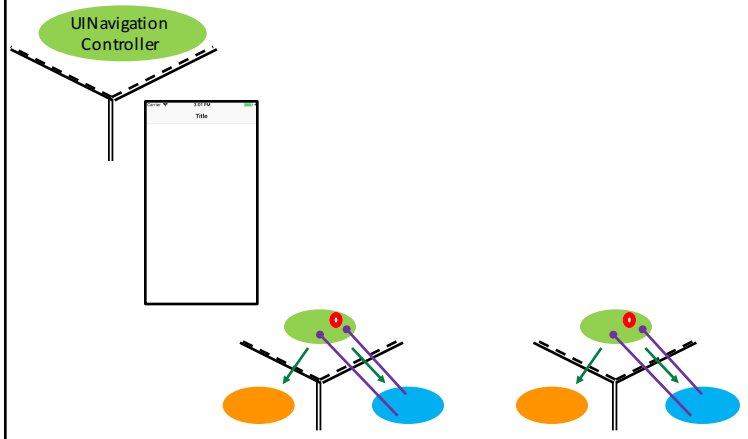
UINavigationController



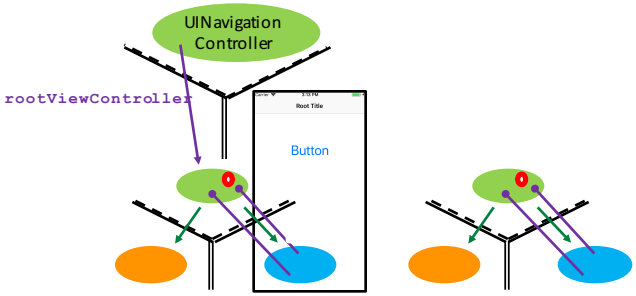
UINavigationController



UINavigationController

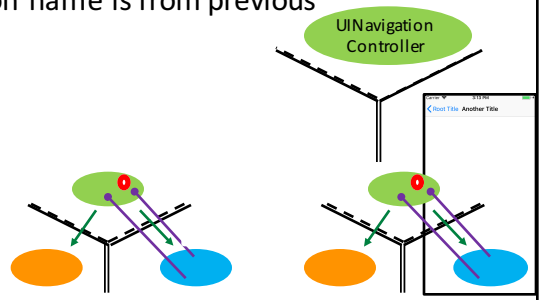


UINavigationController



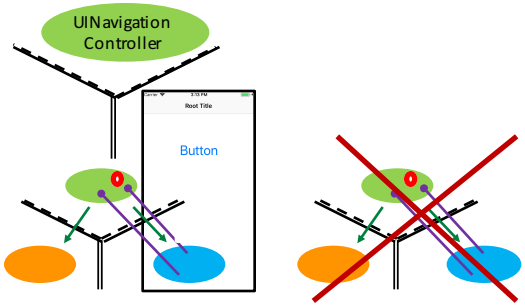
"Show" Segue to Another MVC

- **Creates a new MVC every time**
- Title is from new controller
- Back button name is from previous



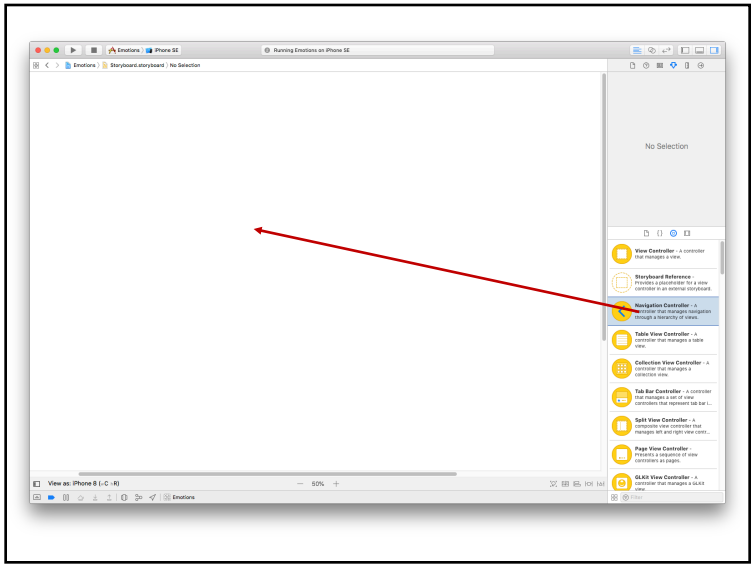
Back to Previous

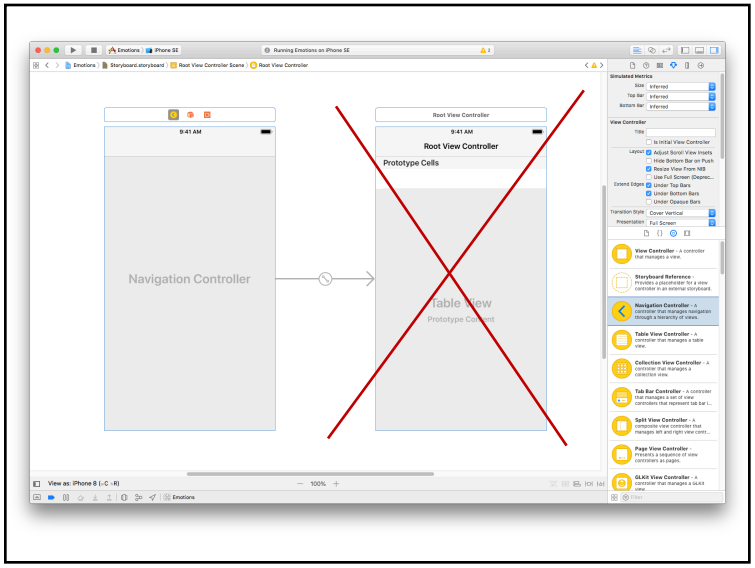
- MVCs come and go
- Store data in models to keep around!

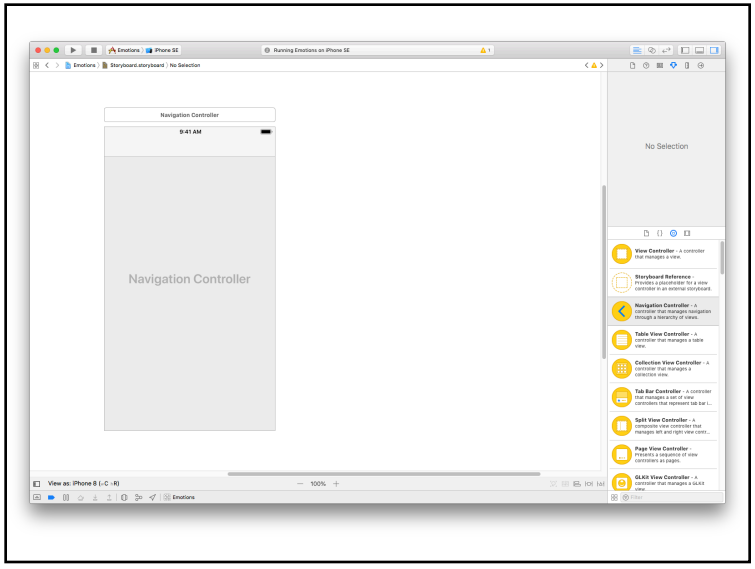


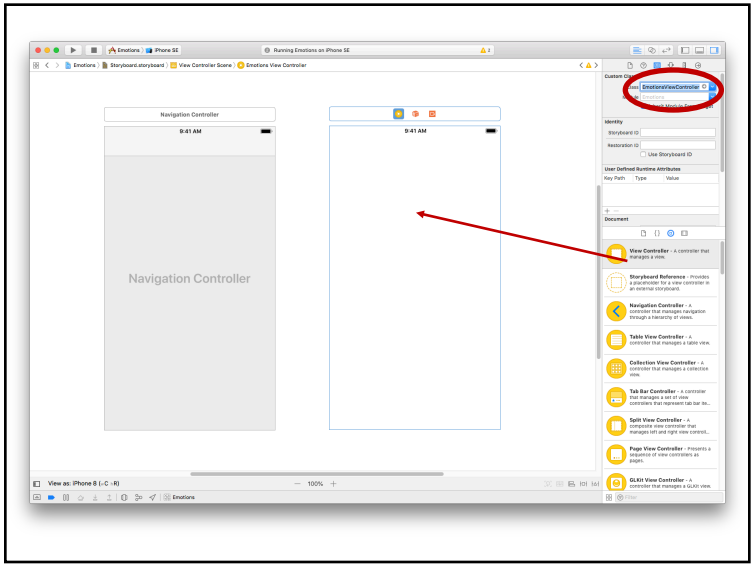
Accessing sub-MVCs and super-MVCs

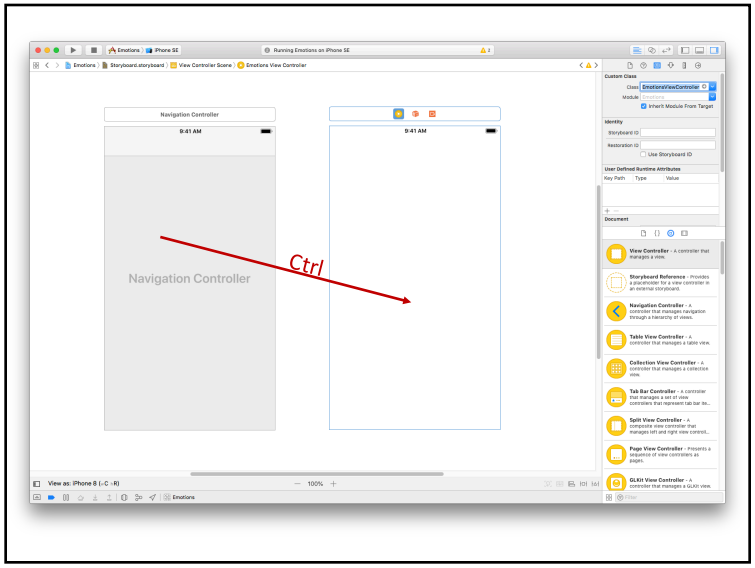
- UIViewController
  - var viewControllers : [UIViewController]?
    - tab bar: [0] is furthest left, [last] is furthest right
    - split view: [0] is master, [1] is detail
    - navigation: [0] is top of stack, [last] is bottom of stack
- Can always get enclosing controller
  - var tabBarController : UITabBarController?
  - var splitViewController : UISplitViewController?
  - var navigationController : UINavigationController?
  - if let detail = splitViewController?.viewControllers[1] {...}

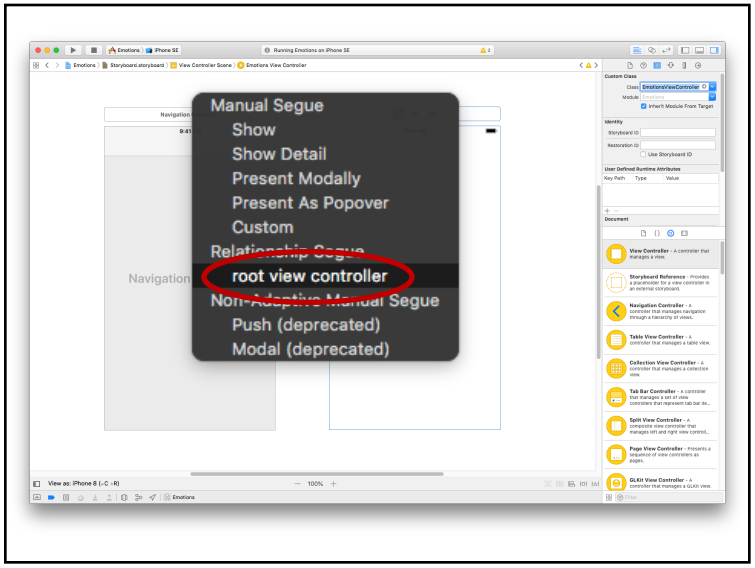

No Selection

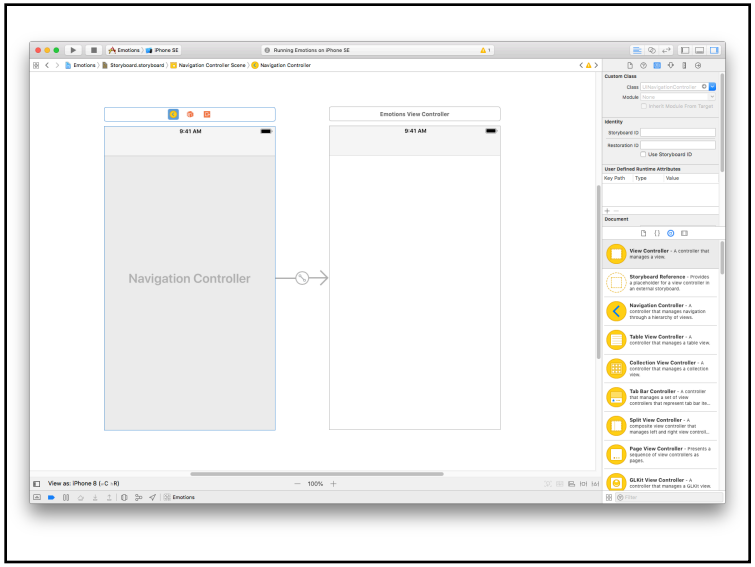

Root View Controller
Prototype Cells
Navigation Controller
Table View
Prototype Content

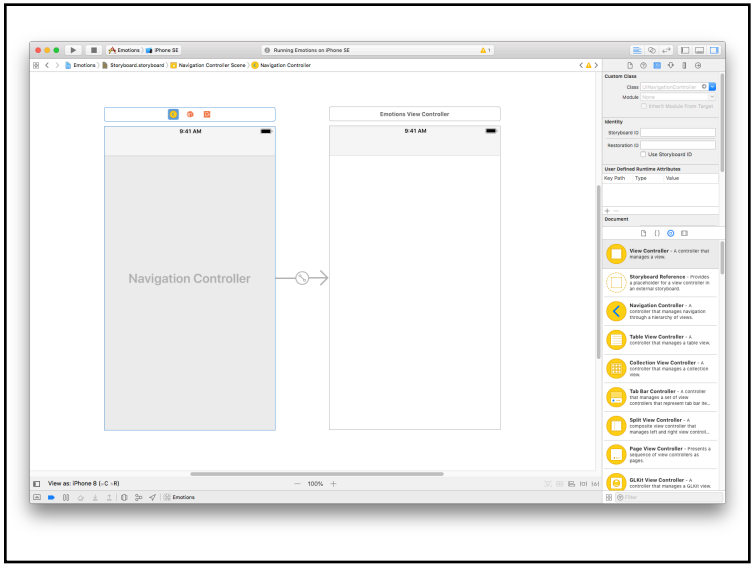

Navigation Controller
No Selection

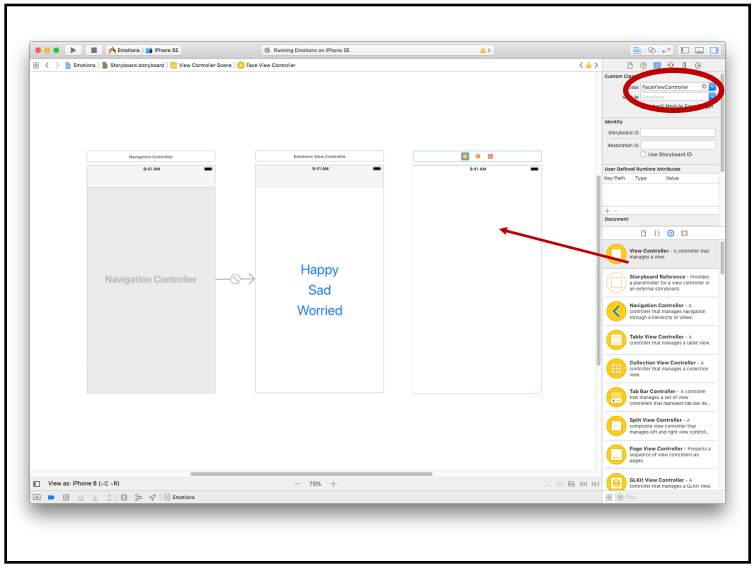

Navigation Controller

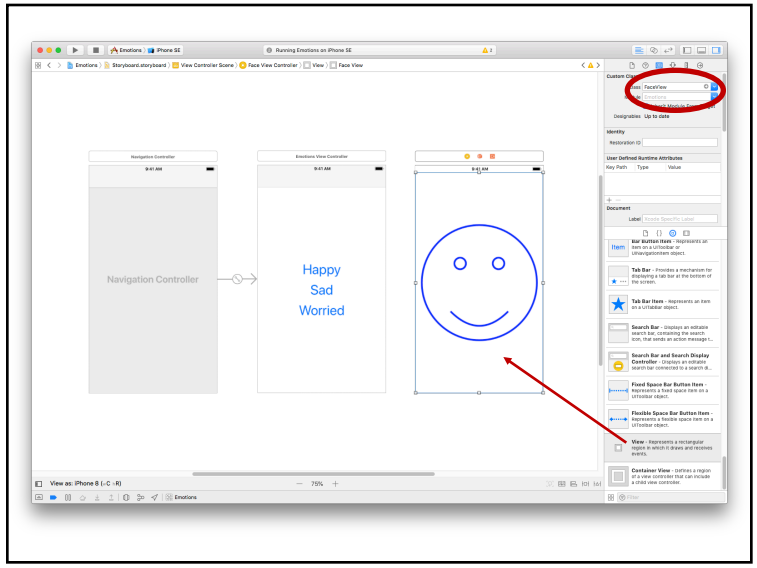

Navigation Controller
Ctrl

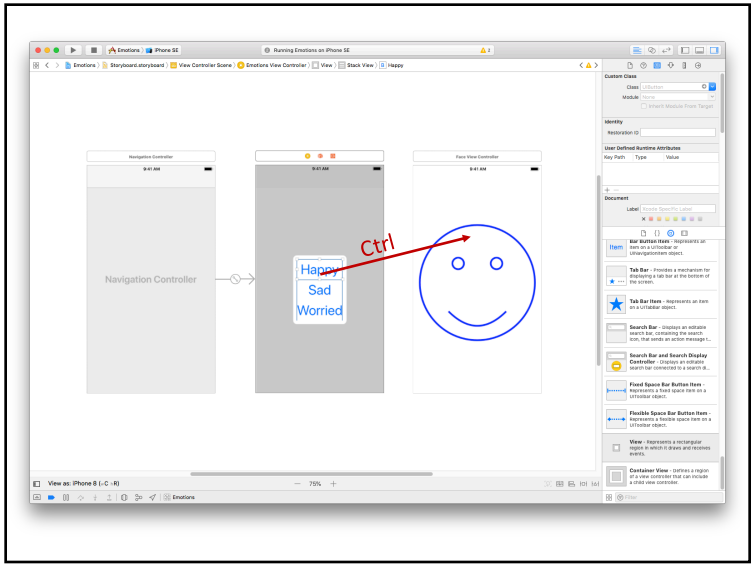

Manual Segue
Show
Show Detail
Present Modally
Present As Popover
Custom
root view controller
Push (deprecated)
Modal (deprecated)

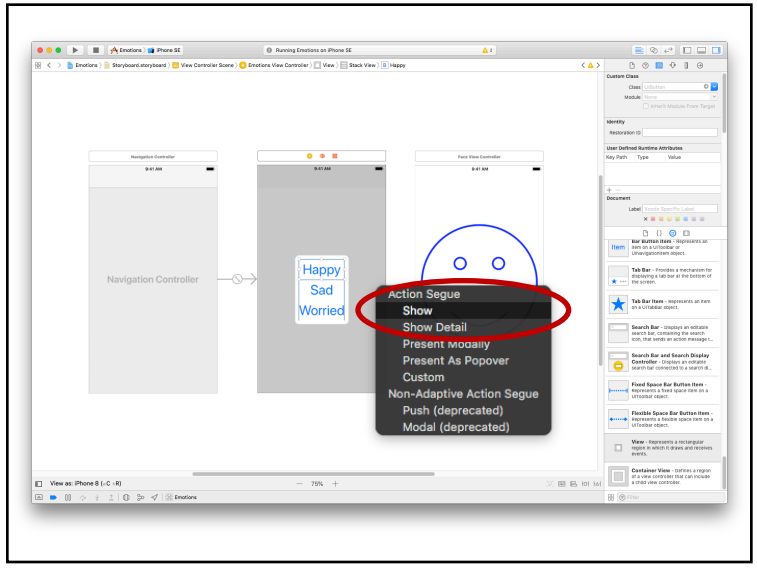

Navigation Controller
Emotions View Controller

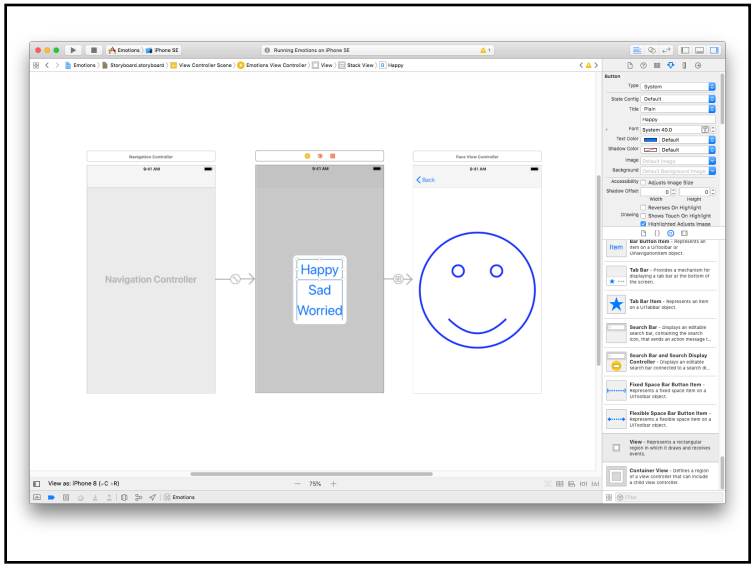

Navigation Controller
Emotions View Controller

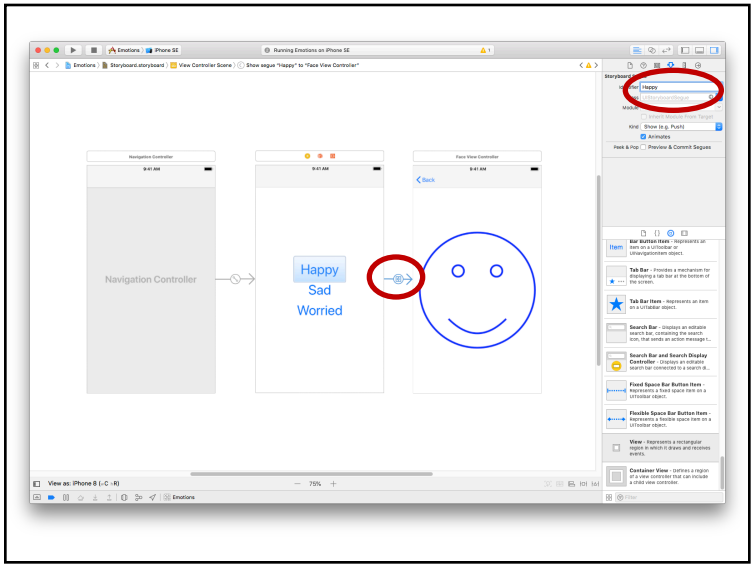

Navigation Controller
Happy
Sad
Worried

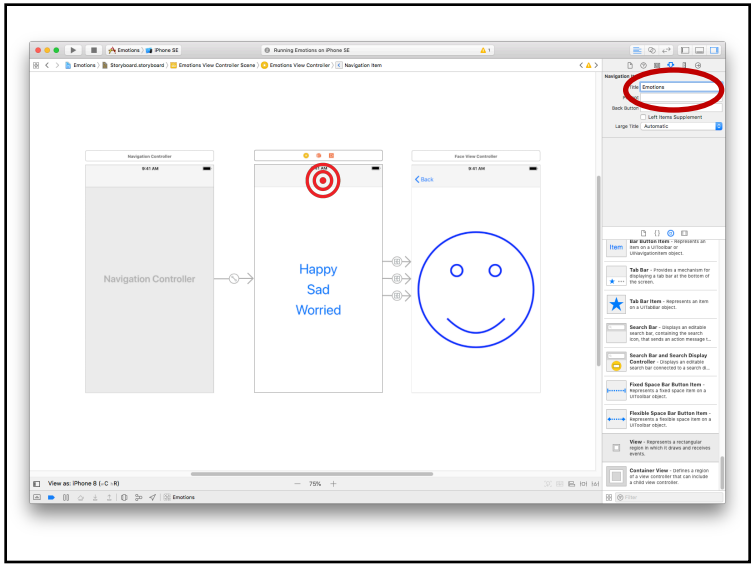

Navigation Controller
Happy
Sad
Worried

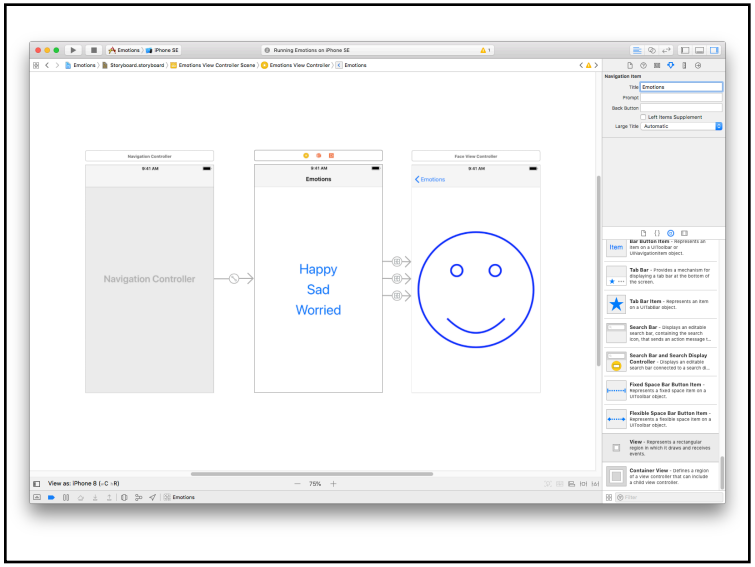

Navigation Controller
Happy
Sad
Worried
Ctrl


Navigation Controller
Happy
Sad
Worried
Action Segue
Show
Show Detail
Present Modally
Present As Popover
Custom
Non-Adaptive Action Segue
Push (deprecated)
Modal (deprecated)


Navigation Controller
Happy
Sad
Worried


Navigation Controller
Happy
Sad
Worried


Navigation Controller
Happy
Sad
Worried


Navigation Controller
Happy
Sad
Worried

Segues

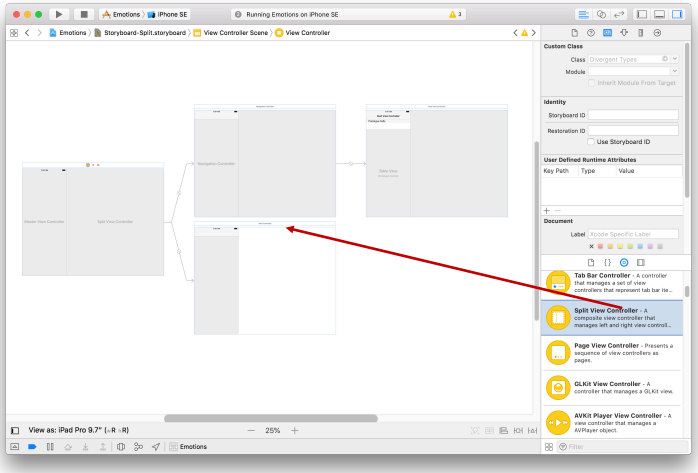
- Kinds
  - Show (push in a Navigation Controller, else Modal)
  - Show Detail (show in Detail of a Split View or push in a Navigation Controller)
  - Modal (take over the entire screen)
  - Popover (appears in a little popover window)
- Segues **always** create a new instance of an MVC
  - Even the Detail of a Split View will get replaced with a new instance of that MVC every time you switch to it.

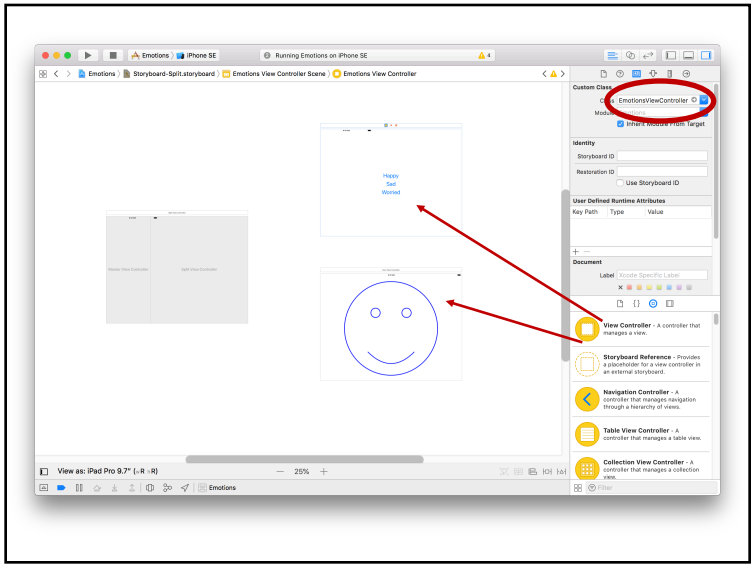
EmotionsViewController

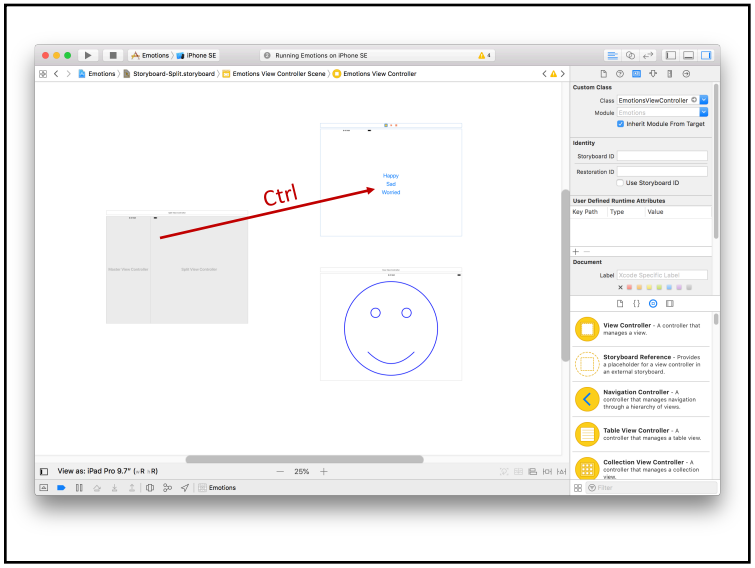
```
func prepare(for segue: UIStoryboardSegue, sender: Any?) {
  let dest = segue.destination
  if let faceViewController = dest as? FaceViewController,
     let identifier = segue.identifier,
     let expression = emotionalFaces[identifier] {
    faceViewController.facialExpression = expression
    faceViewController.navigationItem.title =
                    (sender as? UIButton)?.currentTitle
  }
}

private let emotionalFaces = [
  "Happy" : FacialExpression(eyes: .open, mouth: .smile), ...
]
```

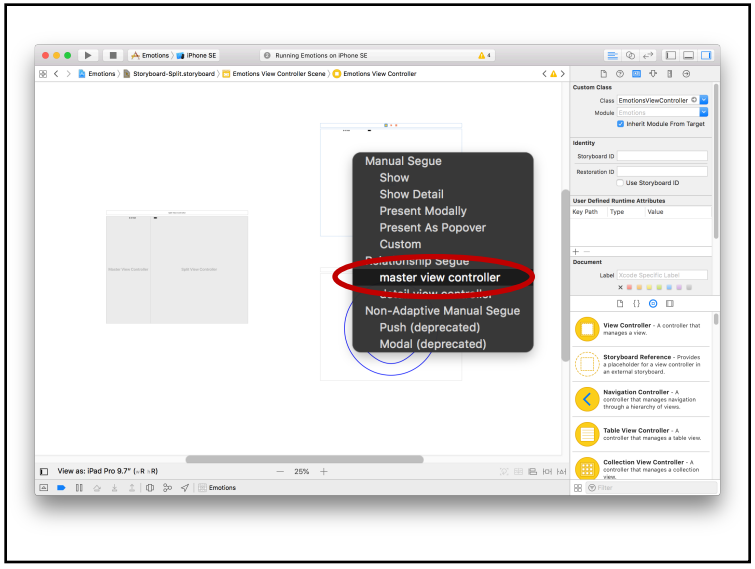
Segues

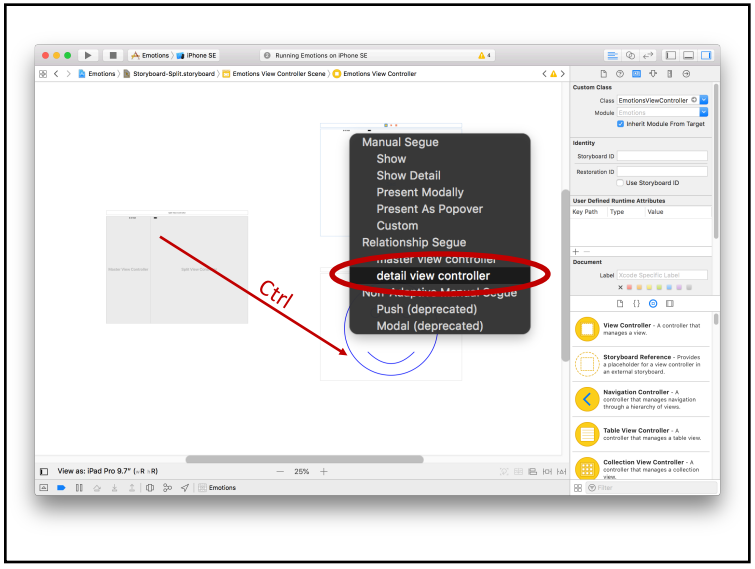
- It is crucial to understand that this preparation is happening BEFORE outlets of destination are set!
- It is a very common bug to prepare an MVC thinking its outlets are set

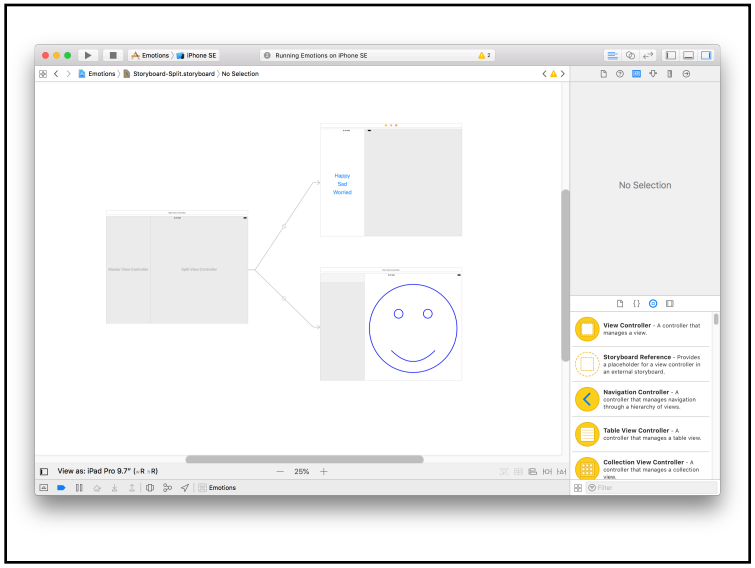


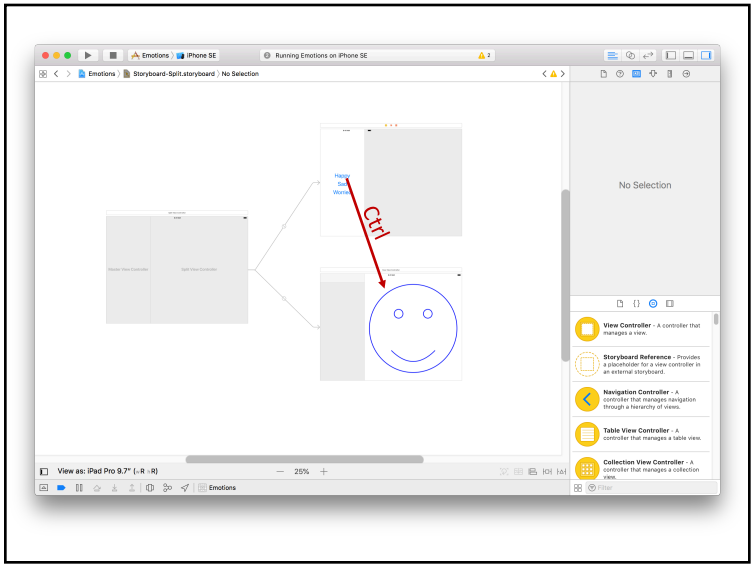


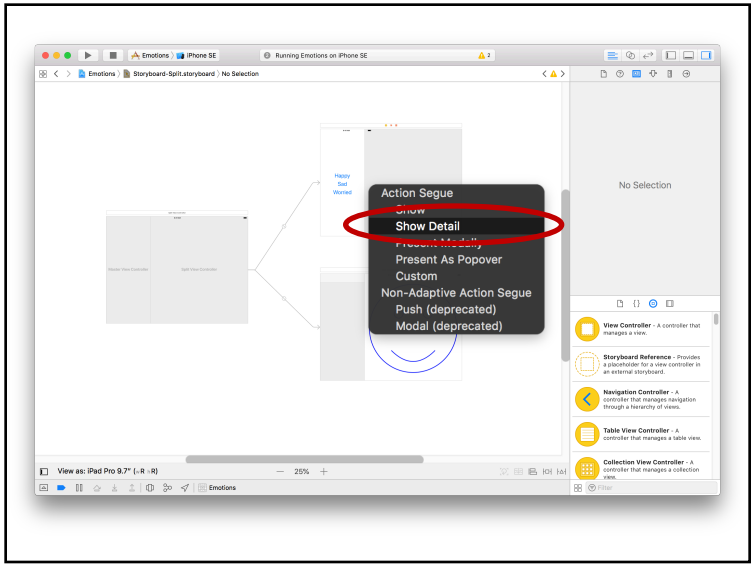

Ctrl

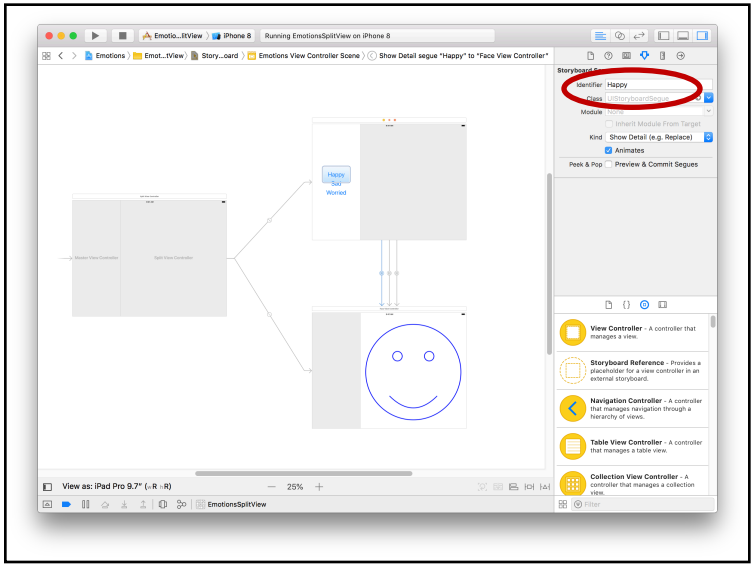

Manual Segue
Show
Show Detail
Present Modally
Present As Popover
Custom
master view controller
Non-Adaptive Manual Segue
Push (deprecated)
Modal (deprecated)

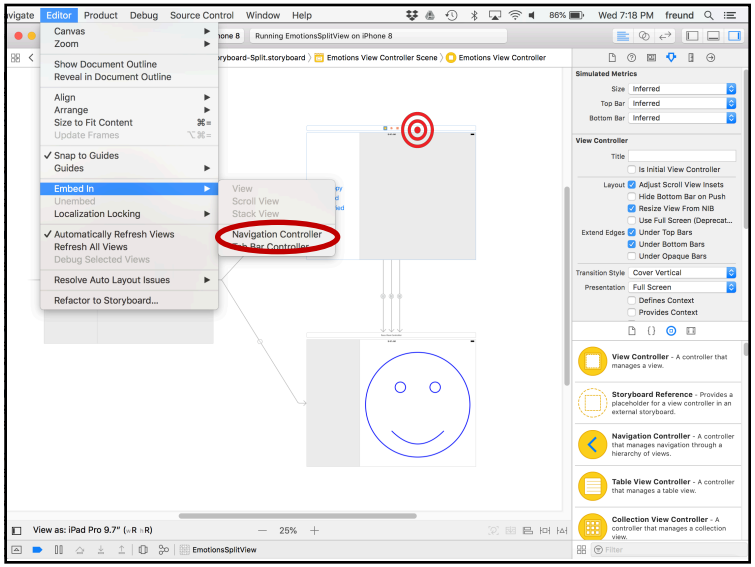

Manual Segue
Show
Show Detail
Present Modally
Present As Popover
Custom
Relationship Segue
detail view controller
Push (deprecated)
Modal (deprecated)
Ctrl

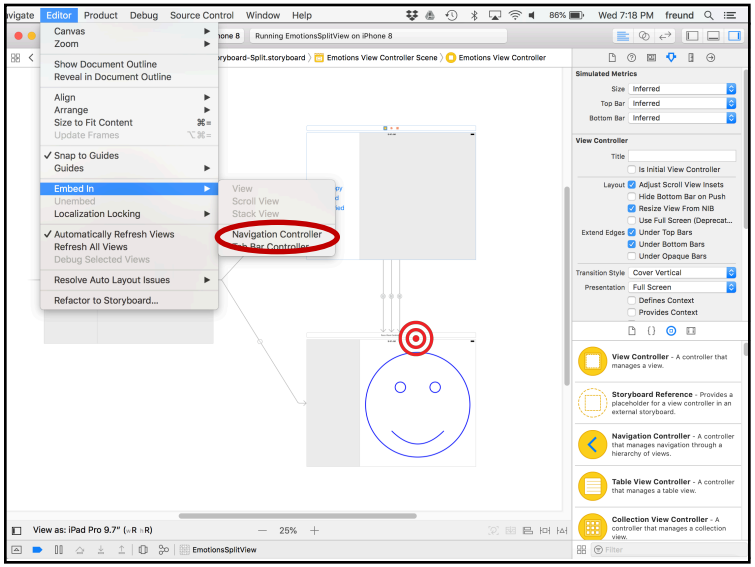

No Selection

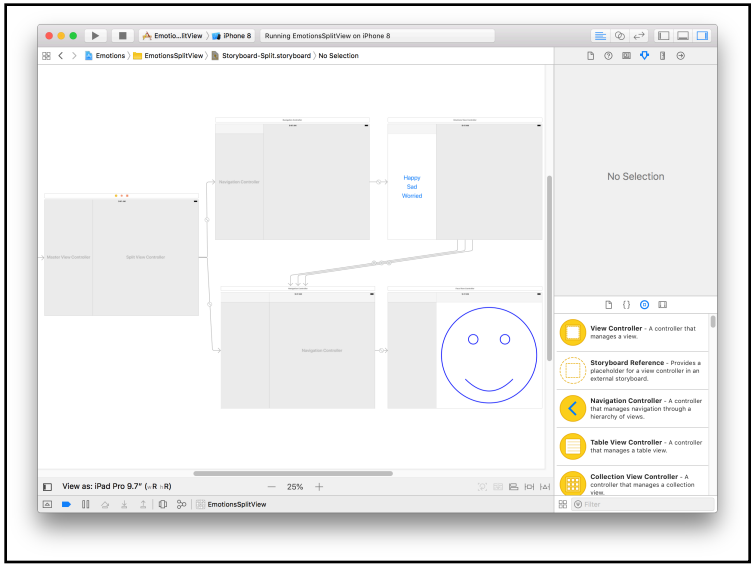

Ctrl
No Selection


Action Segue
Show Detail
Present As Popover
Custom
Non-Adaptive Action Segue
Push (deprecated)
Modal (deprecated)
No Selection


Identifier Happy
Kind Show Detail (e.g. Replace)
Animates
Peek & Pop Preview & Commit Segues







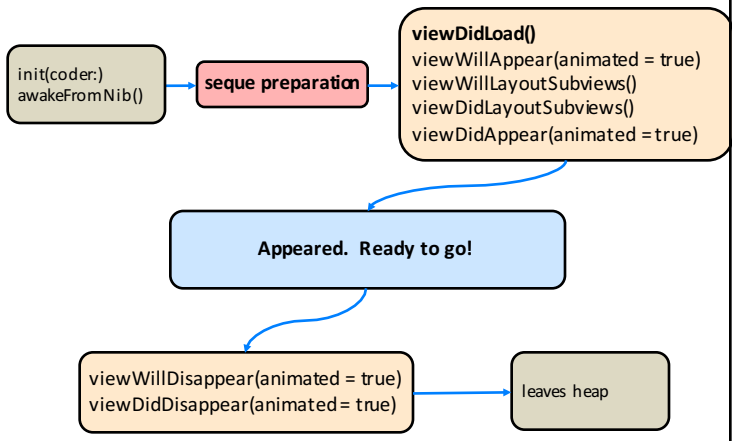
EmotionsViewController

```
func prepare(for segue: UIStoryboardSegue, sender: Any?) {
  var dest = segue.destination
  if let navigationController = dest as? UINavigationController {
    dest = navigationController.visibleViewController ?? dest
  }
  if let faceViewController = dest as? FaceViewController,
     let identifier = segue.identifier,
     let expression = emotionalFaces[identifier] {
    faceViewController.facialExpression = expression
    faceViewController.navigationItem.title =
                    (sender as? UIButton)?.currentTitle
  }
}

private let emotionalFaces = [
  "Happy" : FacialExpression(eyes: .open, mouth: .smile), ...
]
```

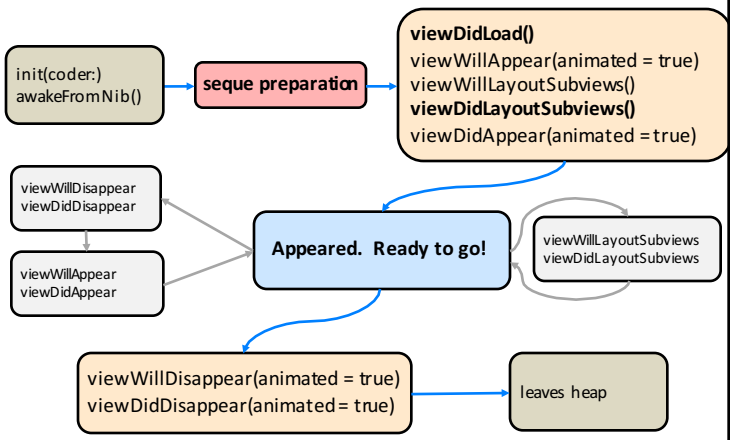
View Controller Lifecycle

- Creation and Initialization
  - instantiated from storyboard
  - segue preparation in triggering controller
  - outlets are set
  - ```
    func viewDidLoad() {
        super.viewDidLoad()
        // your init here...
    }
    ```
- And then layout, appearing, etc.

Typical Lifecycle



Typical Lifecycle



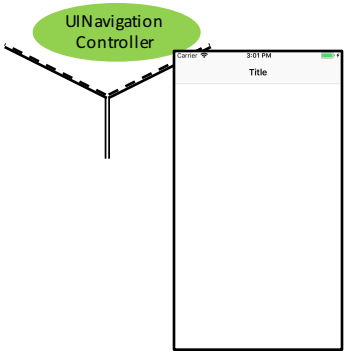
View Controller Lifecycle

- View Events
  - ```
    func viewWillAppear(_ animated: Bool) {
        super.viewWillAppear(animated)
        // your code here...
    }
    ```
  - `func viewDidAppear(_ animated: Bool)`
  - `func viewWillDisappear(_ animated: Bool)`
  - `func viewDidDisappear(_ animated: Bool)`

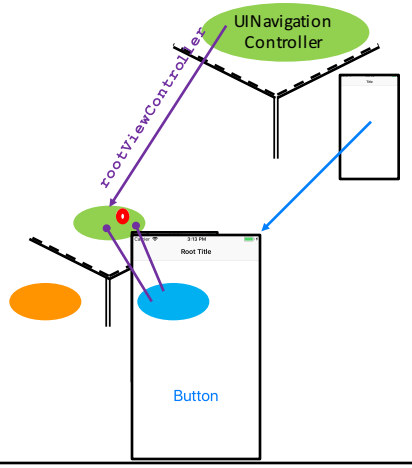
View Controller Lifecycle

- Other Events
  - `func viewWillLayoutSubviews()`
    (… then autolayout happens, then …)
    `func viewDidLayoutSubviews()`
  - If memory gets low, you might get …
    `func didReceiveMemoryWarning()`

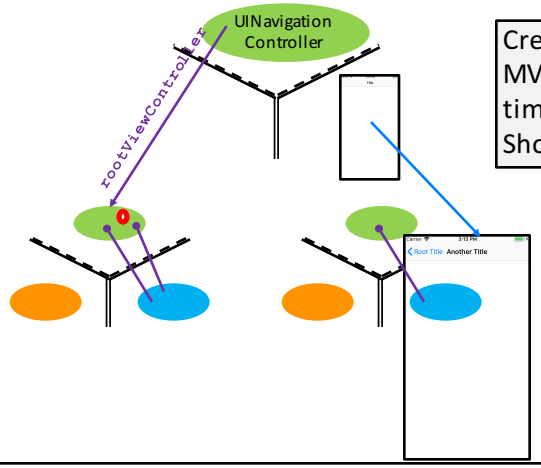
UINavigationController



`rootViewController` Outlet

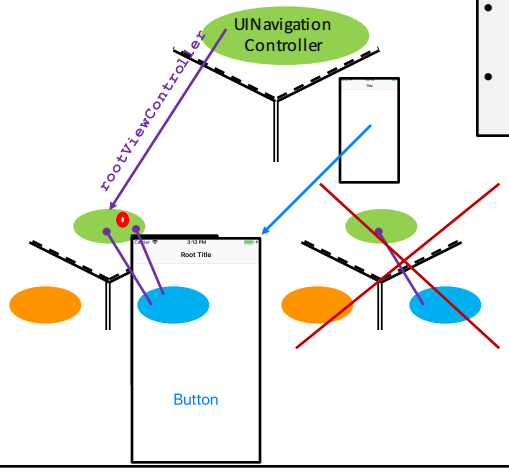


"Show" Segue to Another MVC



Creates a new MVC every time you do a Show Segue

Back to Previous



- MVCs come and go
- Models stick around!

Back To Previous

- Creates a new MVC every time
- Title is from new controller
- Back button is from previous