ITM103 iOS Application Development

Topic 3: View Controllers

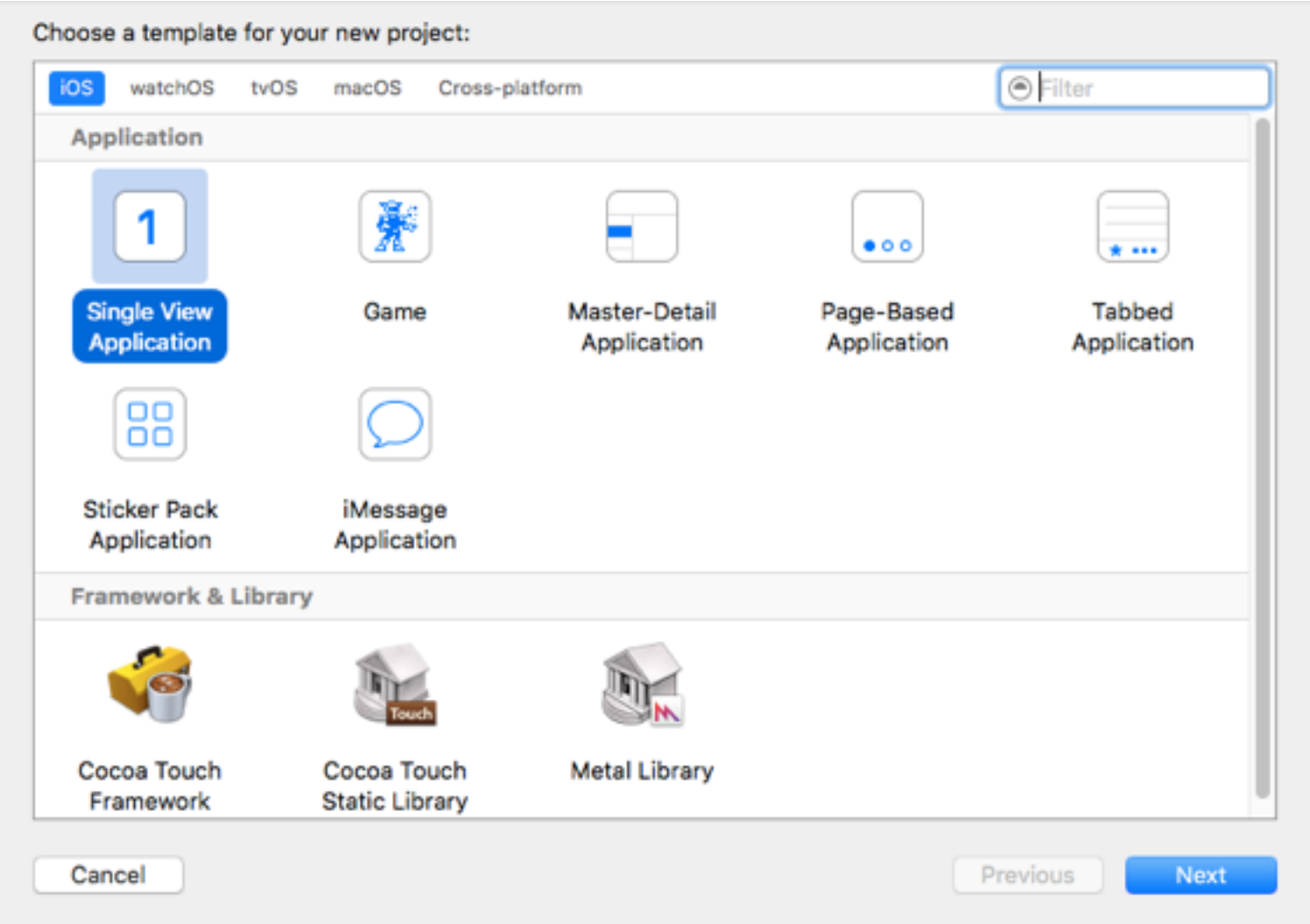
Objectives

- By the end of the lesson, you will be able to:
  - Be familiar with the iOS Project Templates in XCode
  - Be familiar with the Storyboard, Outlets and Actions
  - Be familiar with the Model-View-Controller pattern
  - understand the basics of View Controller
  - evaluate the different types of controllers
  - describe the methods that forms the life cycle of the view controller
  - understand what is storyboard
  - understand the delegation pattern

View Controllers

Quick Look at the Xcode UI

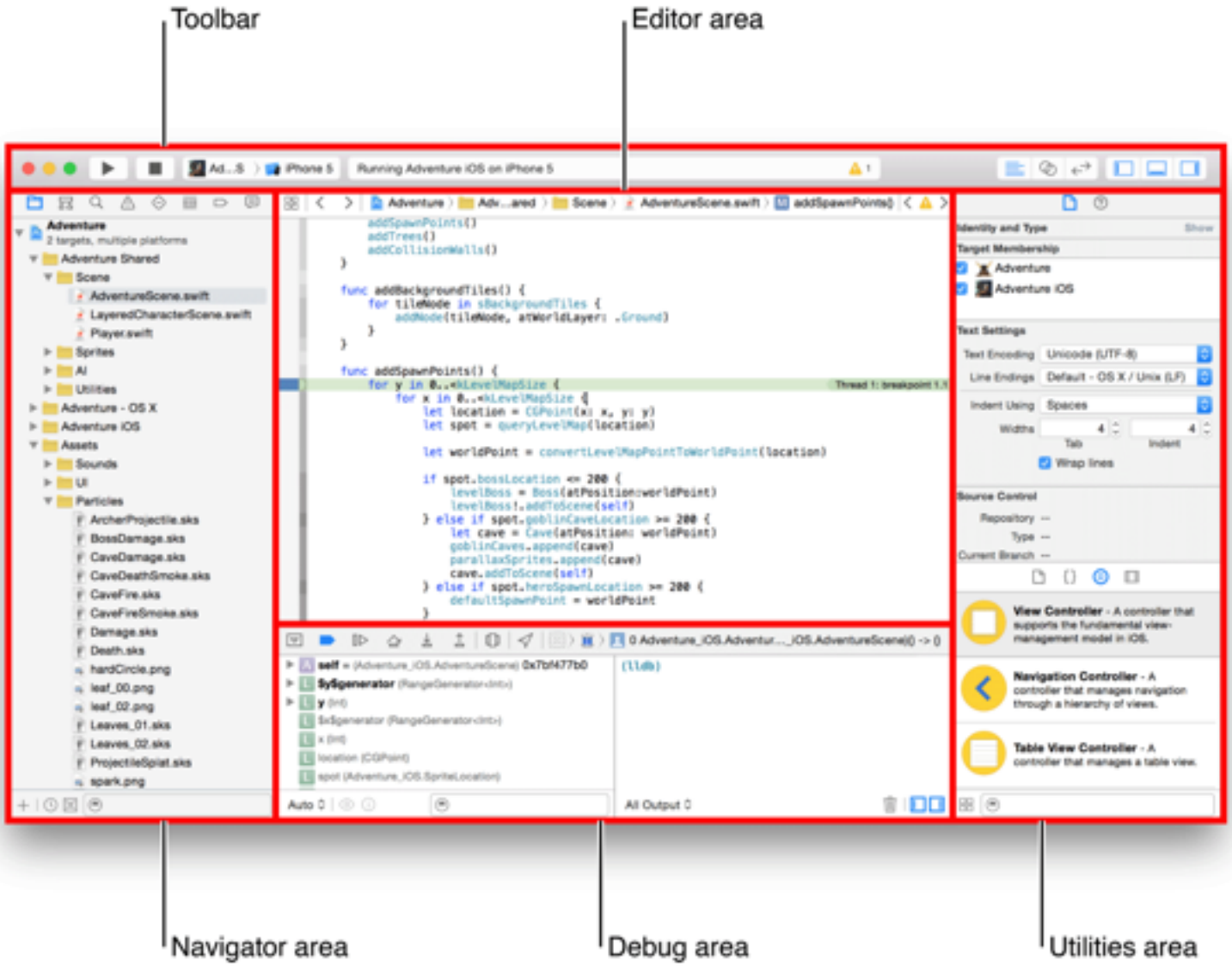
iOS Application Templates



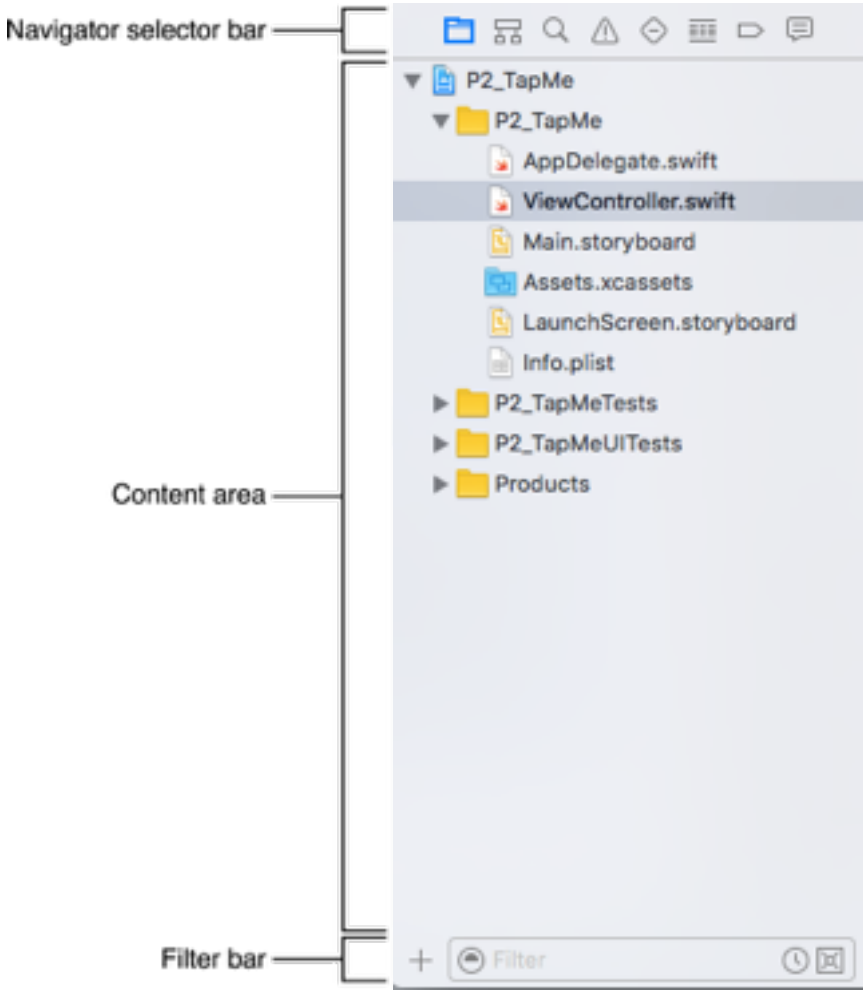
iOS Application Templates

Template	Description
Master-Detail Application	Master-Detail application template includes a navigation controller to display the master list of items.
Page-Based Application	Page-based application template includes a page view controller, which simplifies dealing with multiple pages of information in the application.
Single View Application	Single View application is used when you have only one single view.
Tabbed application	Tabbed application includes a tab bar and a view controller for the tab bar. Used it when you want to create an application with tab bar.
Game	Starting point for Game
Sticker Pack Application	Create a sticker pack application (for iMessage)
iMessage Application	Create an app integrated into iMessages for added functionality

Xcode workspace window



Xcode workspace window



Navigator

Project navigator. Add, delete, group, and otherwise manage files in your project, or choose a file to view or edit its contents in the editor area.

Symbol navigator. Browse the class hierarchy of the symbols in your project.

Find navigator. Use search options and filters to quickly find any string within your project.

Issue navigator. View issues such as diagnostics, warnings, and errors found when opening, analyzing, and building your project.

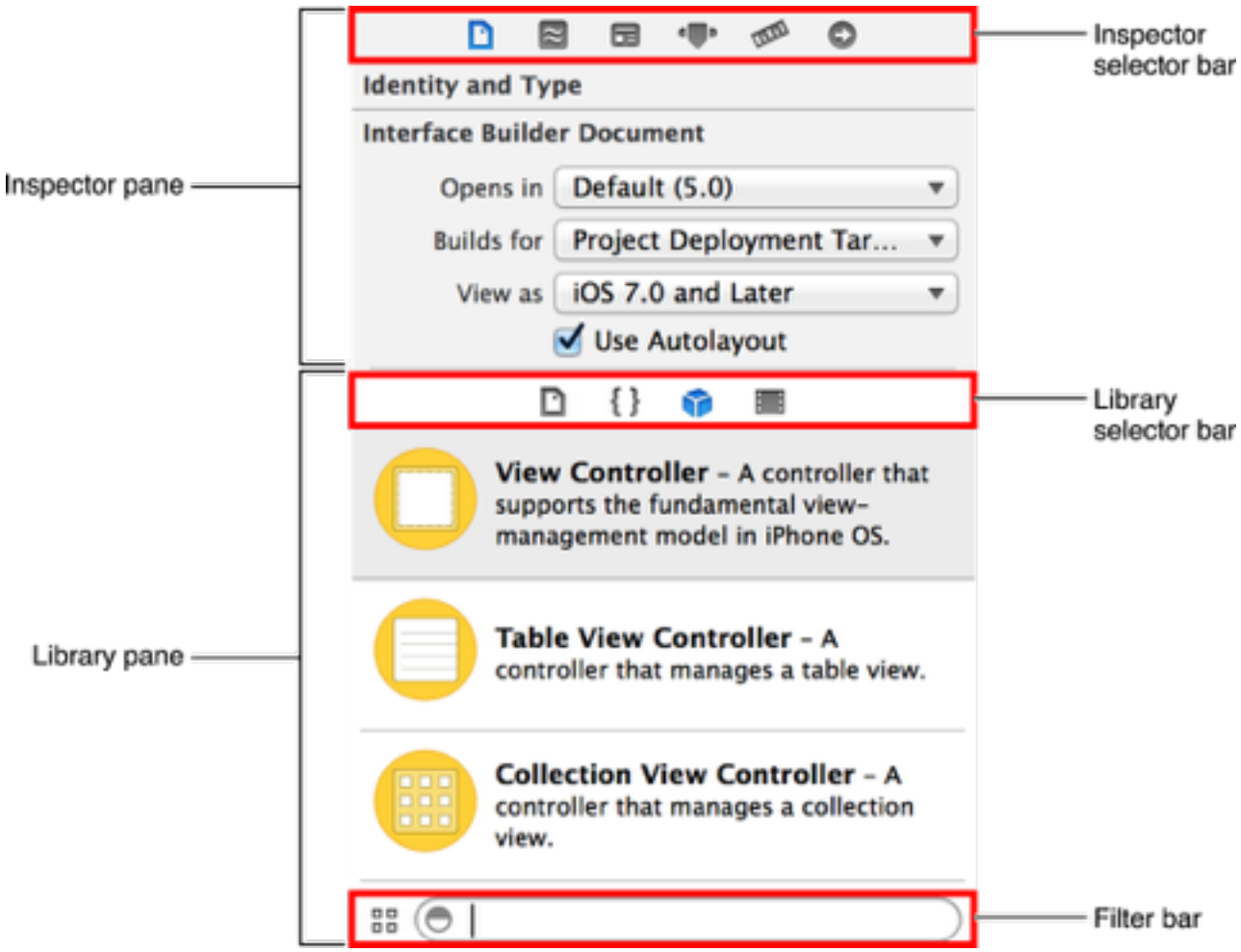
Test navigator. Create, manage, run, and review unit tests.

Debug navigator. Examine the running threads and associated stack information at a specified point or time during program execution.

Breakpoint navigator. Fine-tune breakpoints by specifying characteristics such as triggering conditions.

Log navigator. View the history of your build, run, debug, continuous integration, and source control tasks.

Xcode workspace window



Utility Area

The utility area gives you quick access to these resources: Inspectors, for viewing and modifying characteristics of the file open in an editor Libraries of ready-made resources for use in your project

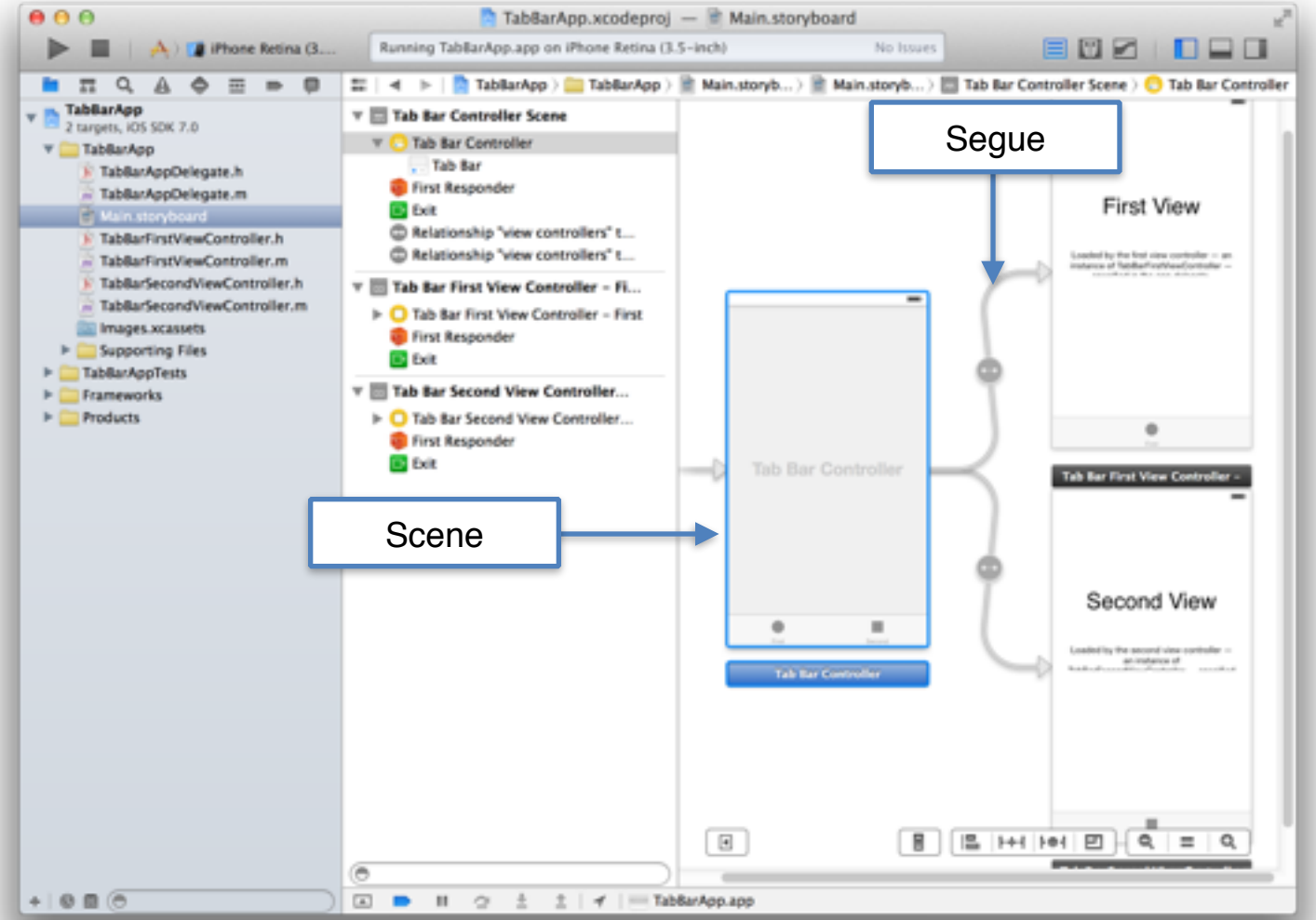
Storyboard

- A storyboard is the visual representation of all the screens in an application. It also tells you about the transitions between various screens.

- A storyboard is represented as scenes and segues.
  - A **scene** refers to a single view or view controller. Each scene has a dock to make outlet and action connections between the view and its view controller.

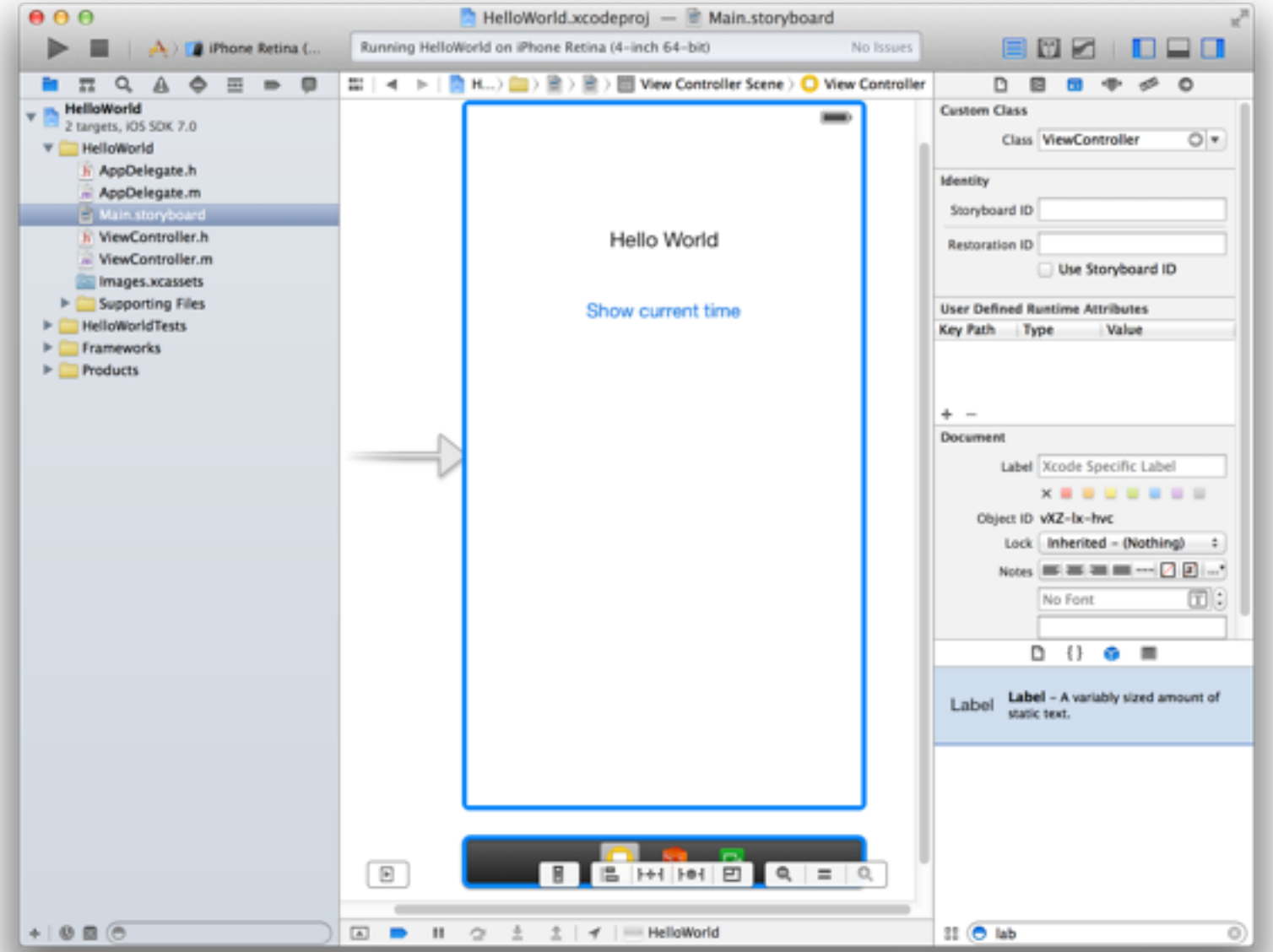
  - A **segue** manages the transition between two scenes. A segue is established by pressing ctrl key and dragging from one scene to the other.

Storyboard



A **storyboard** is a visual representation of the app's user interface, showing screens of content and the transitions between them

Designing your user interface



Outlet and Action

ViewController.swift

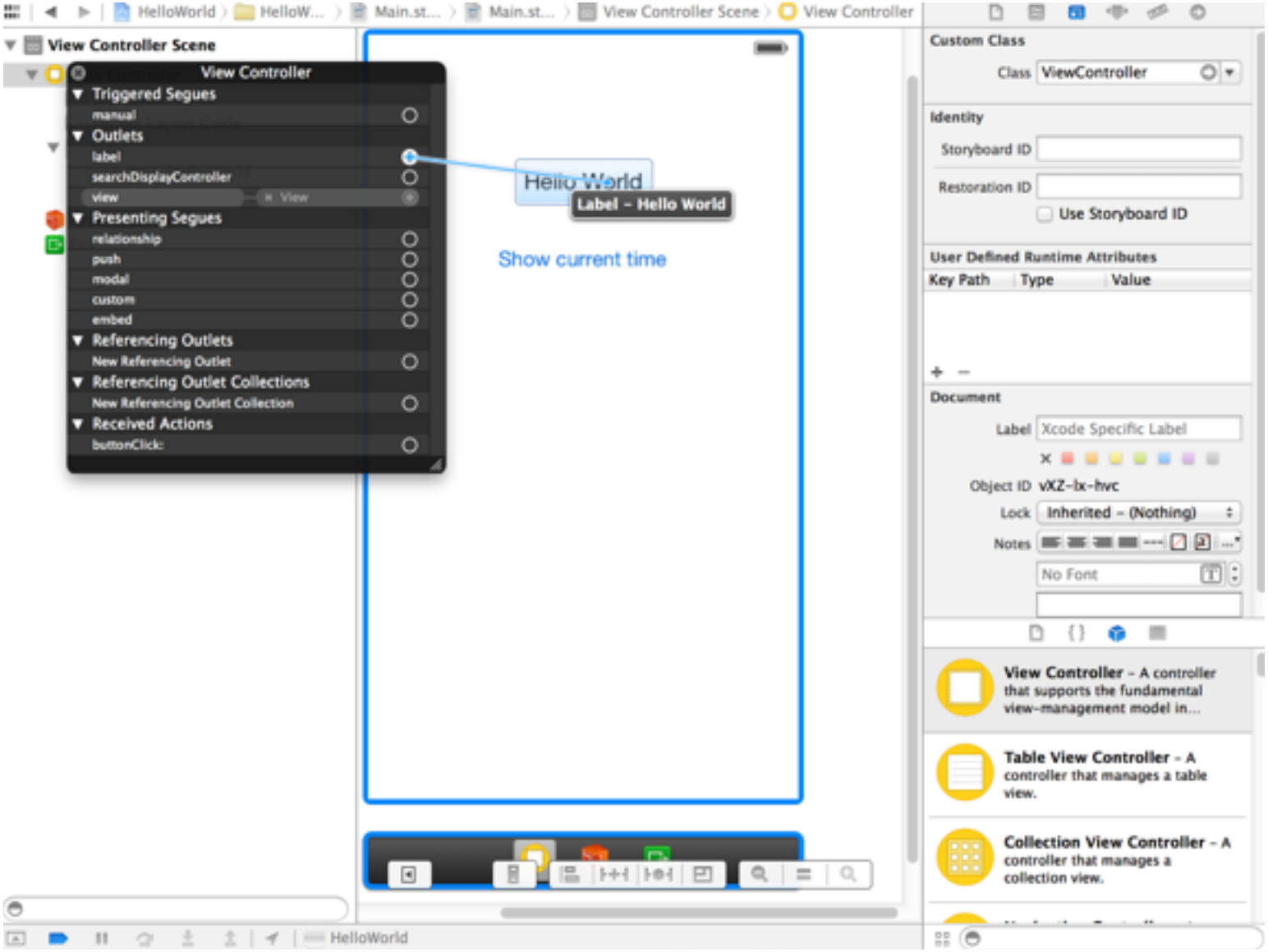
```
@IBOutlet weak var label : UILabel!

@IBAction func buttonPressed(sender: AnyObject) {
    let dateFormatter = DateFormatter()
    dateFormatter.dateFormat = "HH:mm:ss"
    label.text = dateFormatter.string(from: Date())
}
```

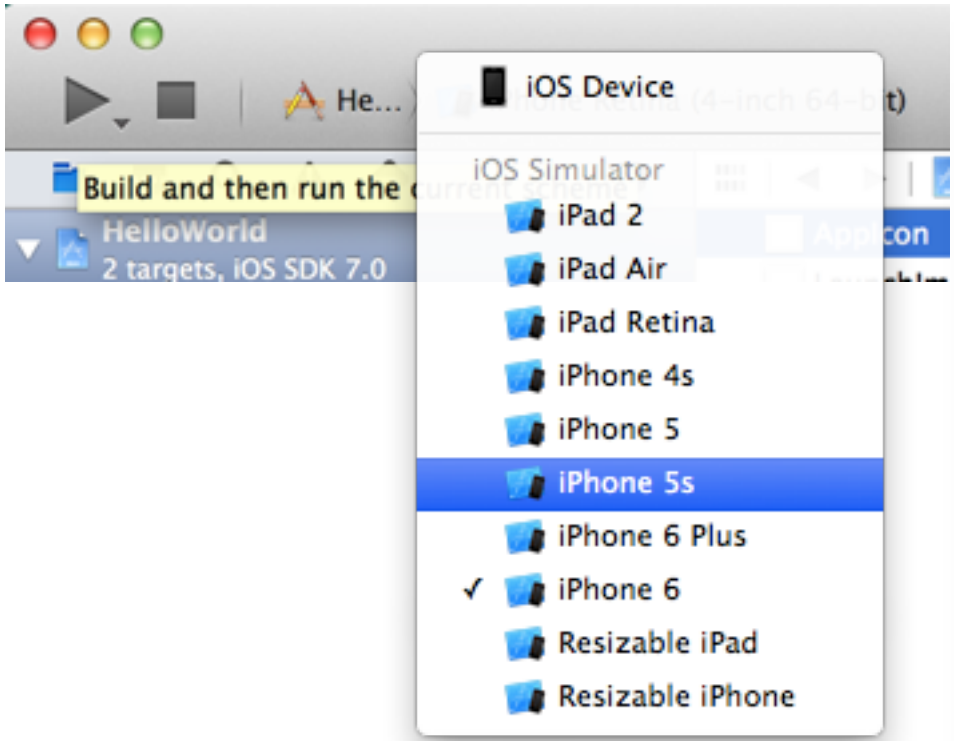
The keyword **IBOutlet** simply denotes that the property can be exposed to Interface Builder and connect with UIElements created from Interface Builder.

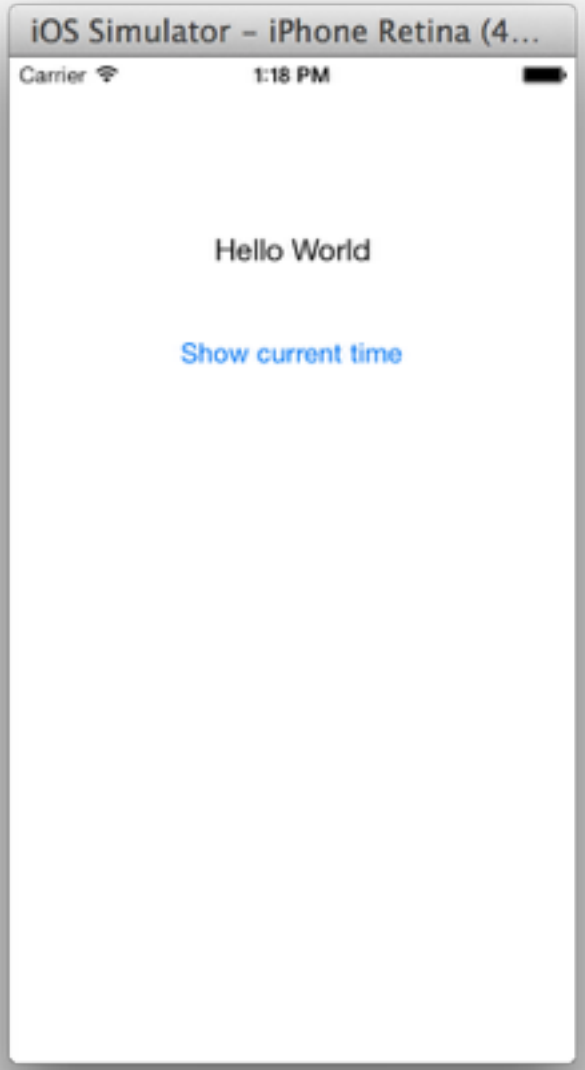
The **IBAction** is a type qualifier used by Interface Builder to synchronize actions. Use this type as the return type of any action methods defined in your project

Connecting Outlet and Action



Running your app using Simulator

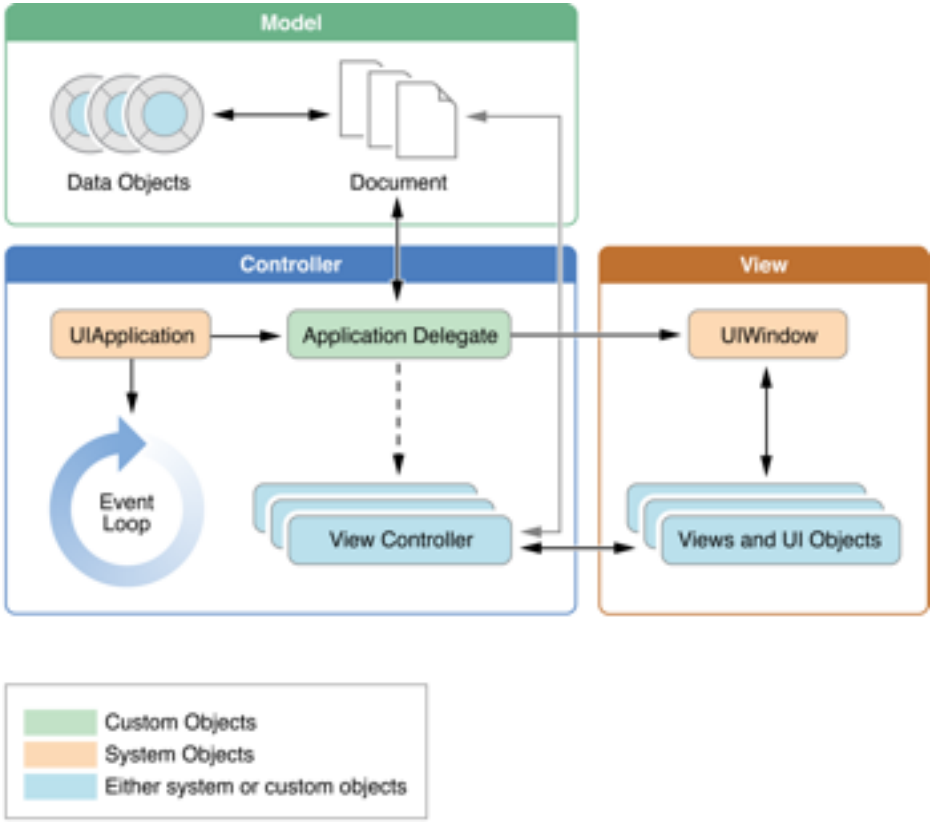




View Controllers

MVC - Model View Controller

Model-View-Controller



View

What the user sees and interacts with

Controller

The mediation between the model & the view

Model

Hold data (business logic) & know nothing about the user interface

Model-View-Controller

View

Model

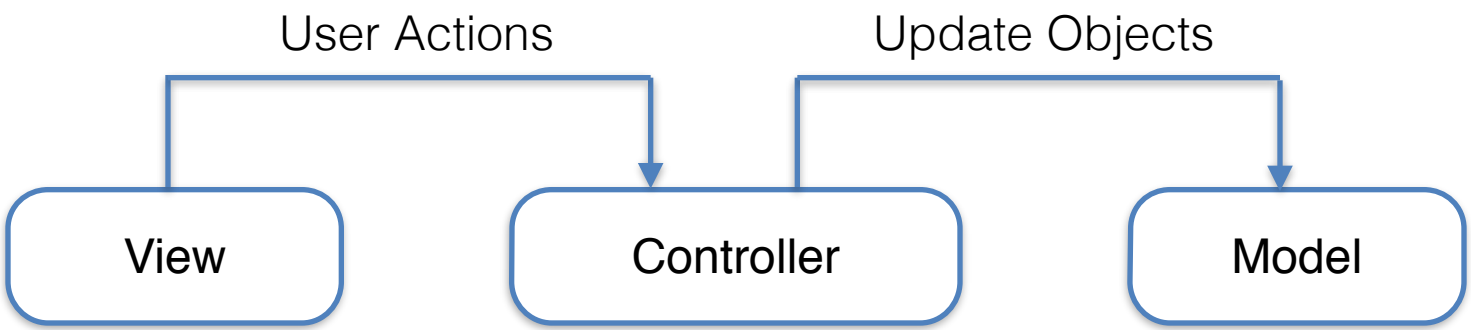
Model-View-Controller

View

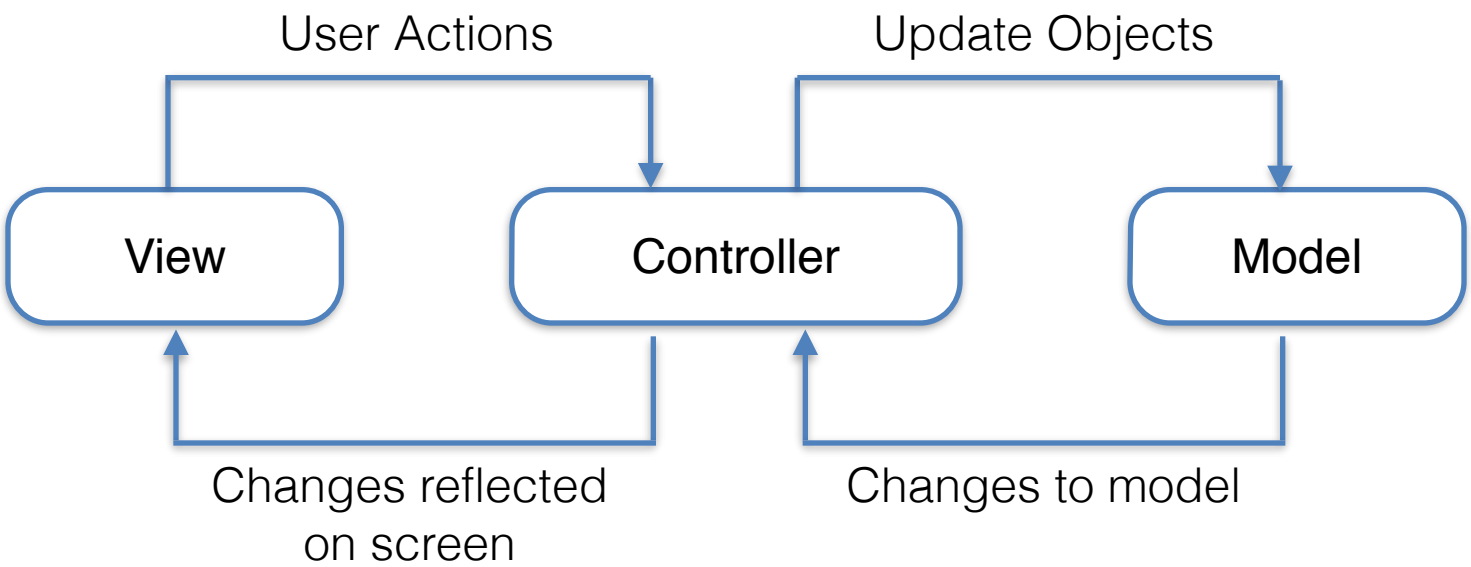
Controller

Model

Model-View-Controller



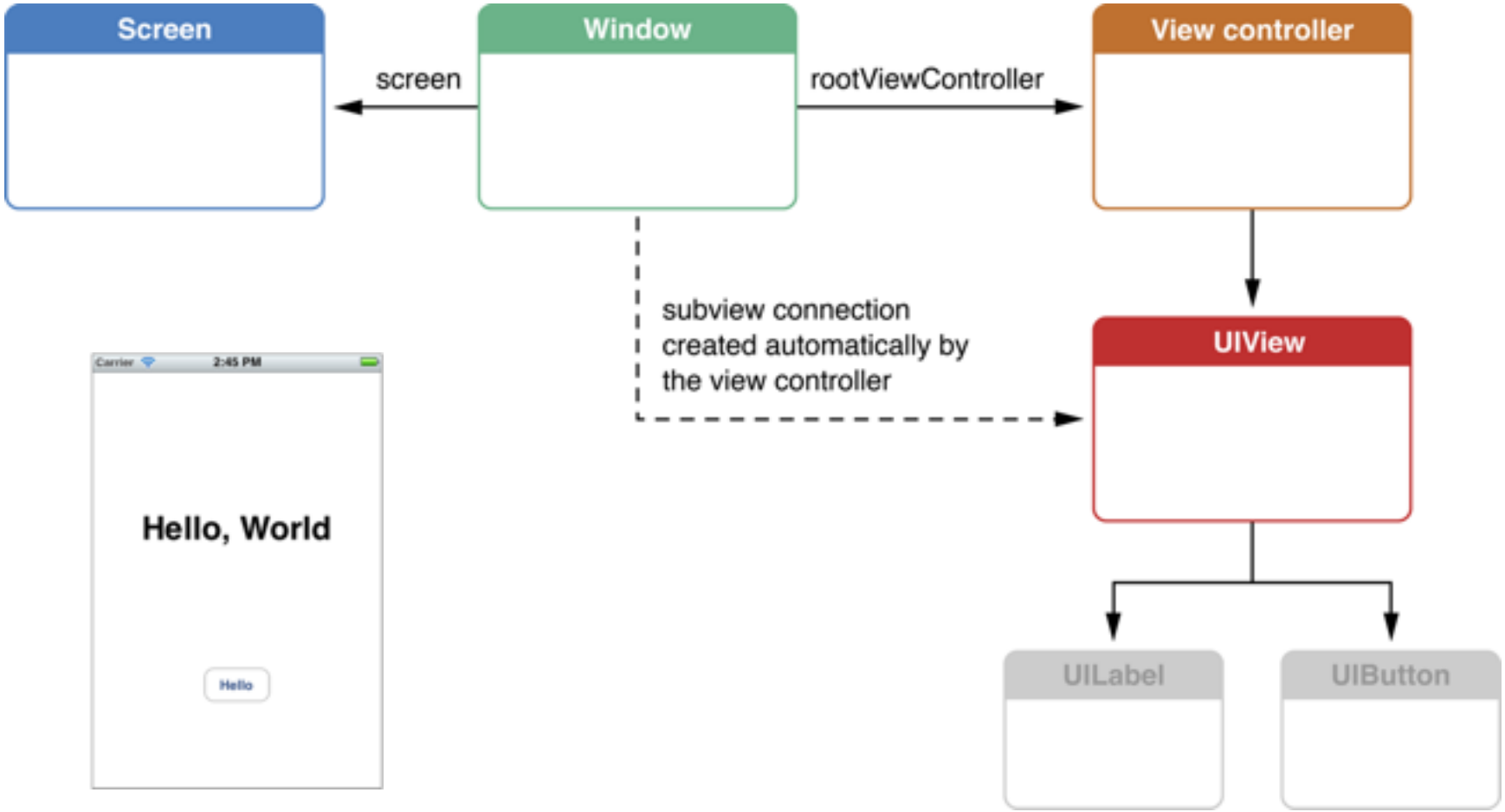
Model-View-Controller



View Controllers

- Organizes and controls a view, often the root view of a view hierarchy
- Are controller objects in the MVC pattern
- Has specific tasks defined in UIViewController
- Always have a reference to the top-level UIView.

View Controllers Manage Views

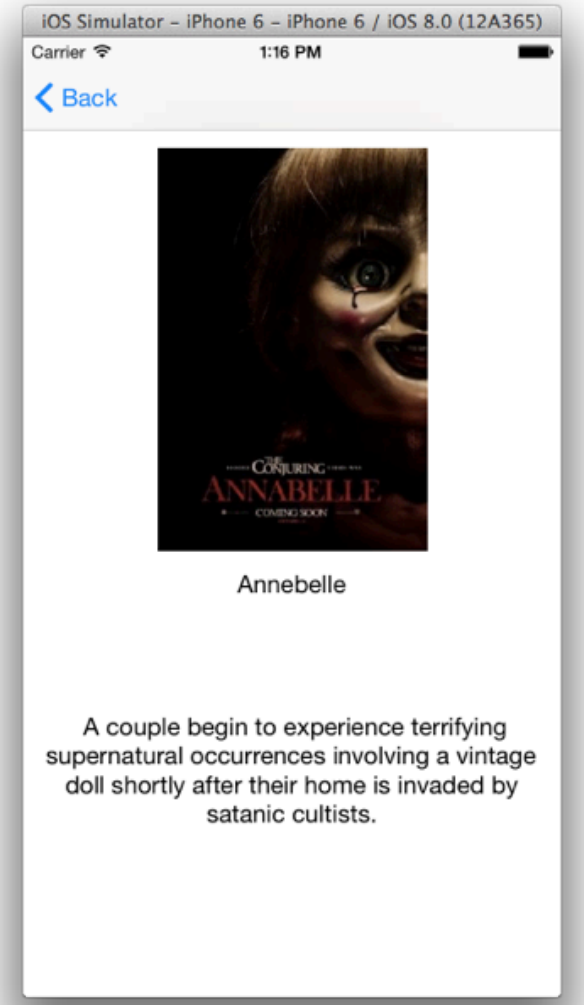


A view controller attached to a window automatically adds its view as a subview of the window

View Controllers

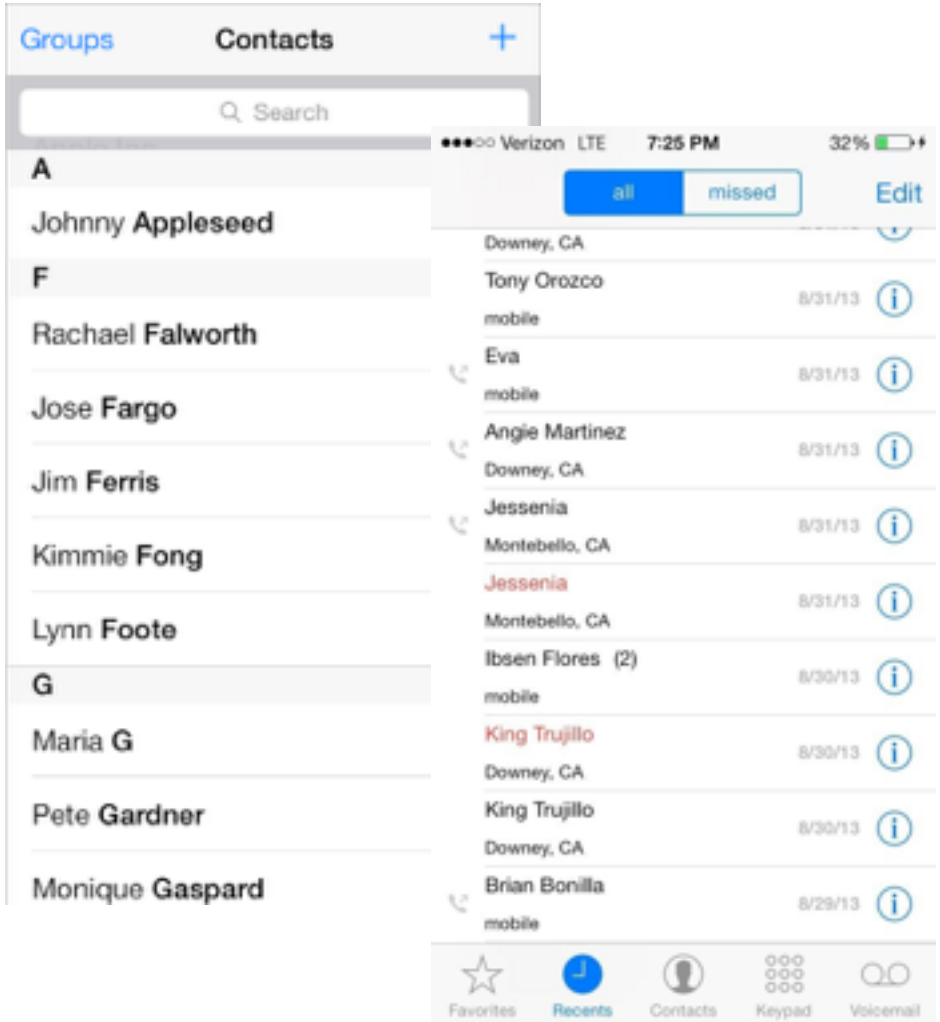
Types of View Controllers

UIViewController



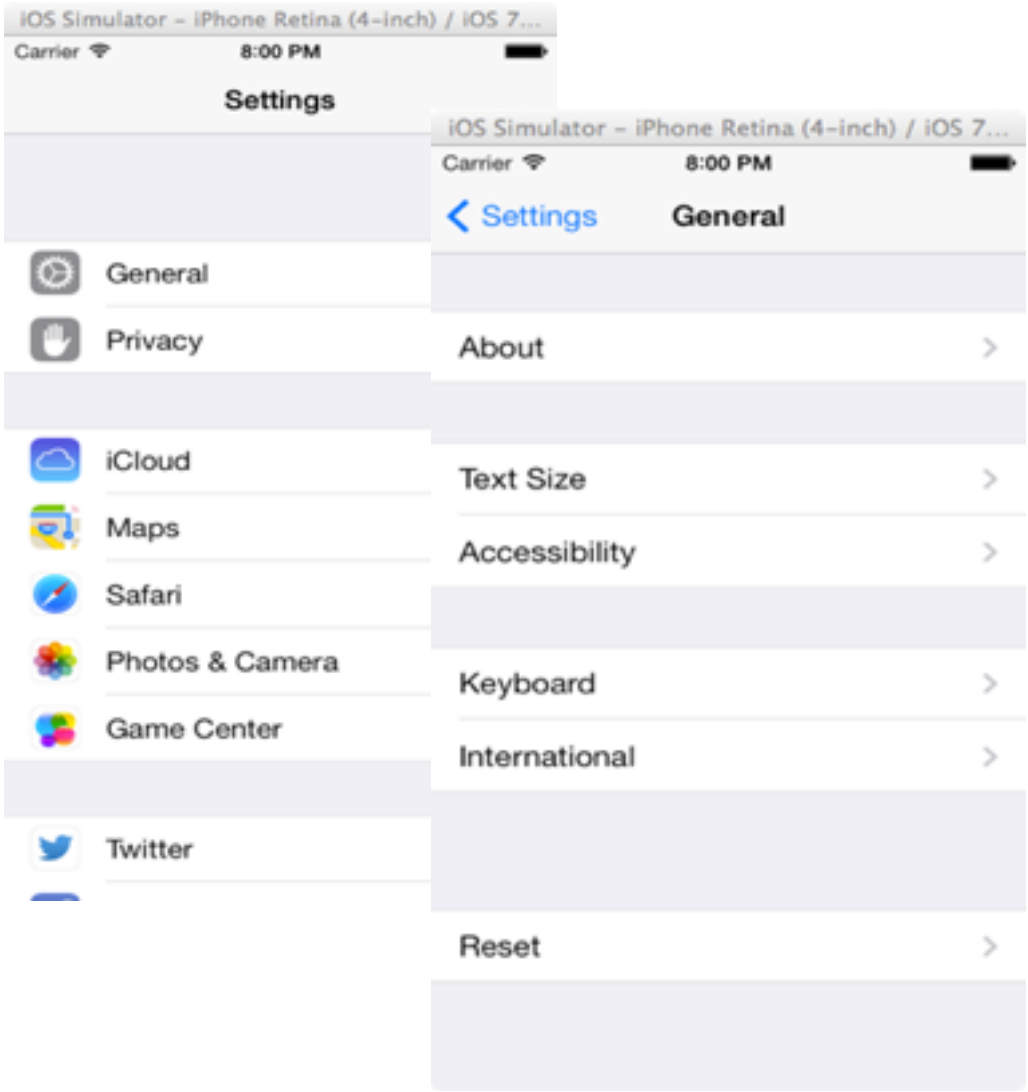
- Starts as an empty view controller.
- Presents any custom user interface designed by the developer.

UITableViewController



- Presents data or actions in a list or table form

UINavigationController

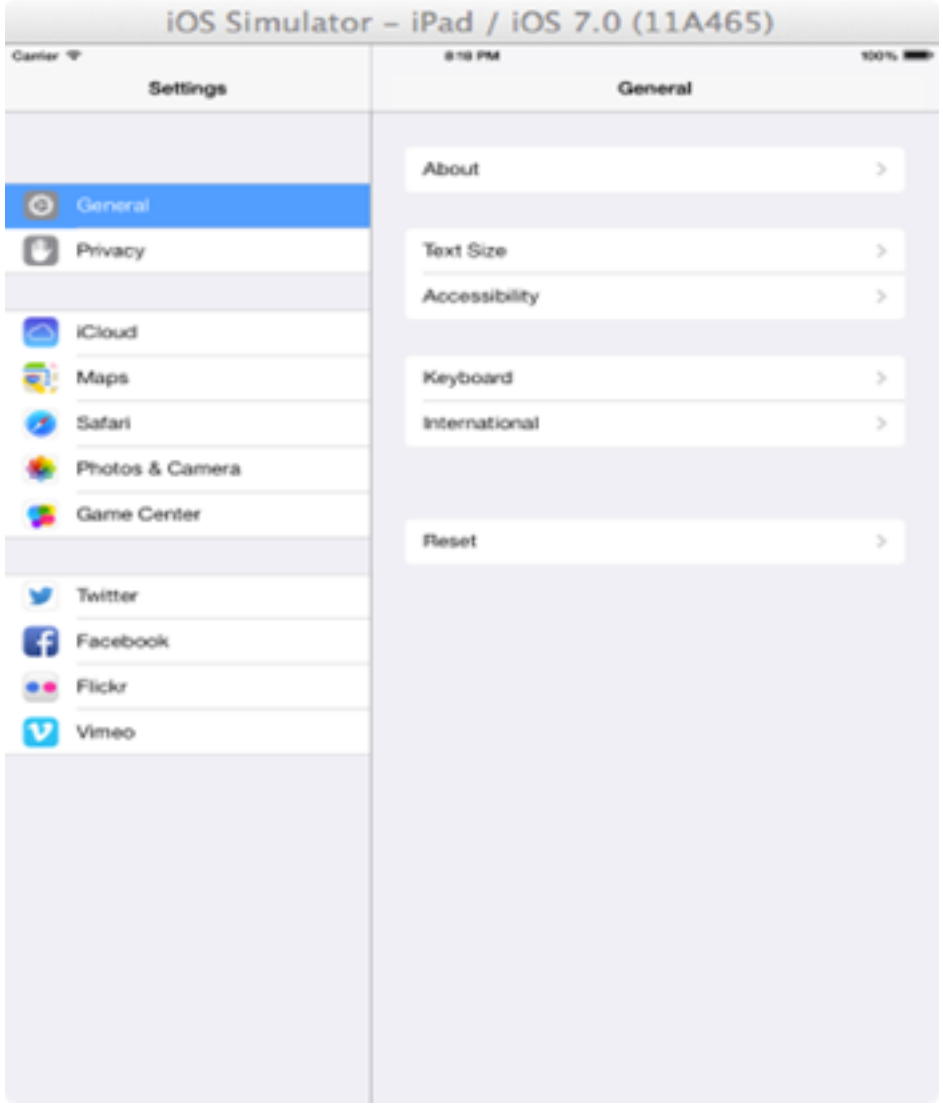


- Manages a navigational or hierarchical flow of MVCs.
- Controllers are arranged in the form of a stack

UITabBarController

- Manages a collection of MVCs.
- A container view controller that you use to divide your app into two or more distinct modes of operation
- The tab bar has multiple tabs, each represented by a child view controller

UISplitViewController

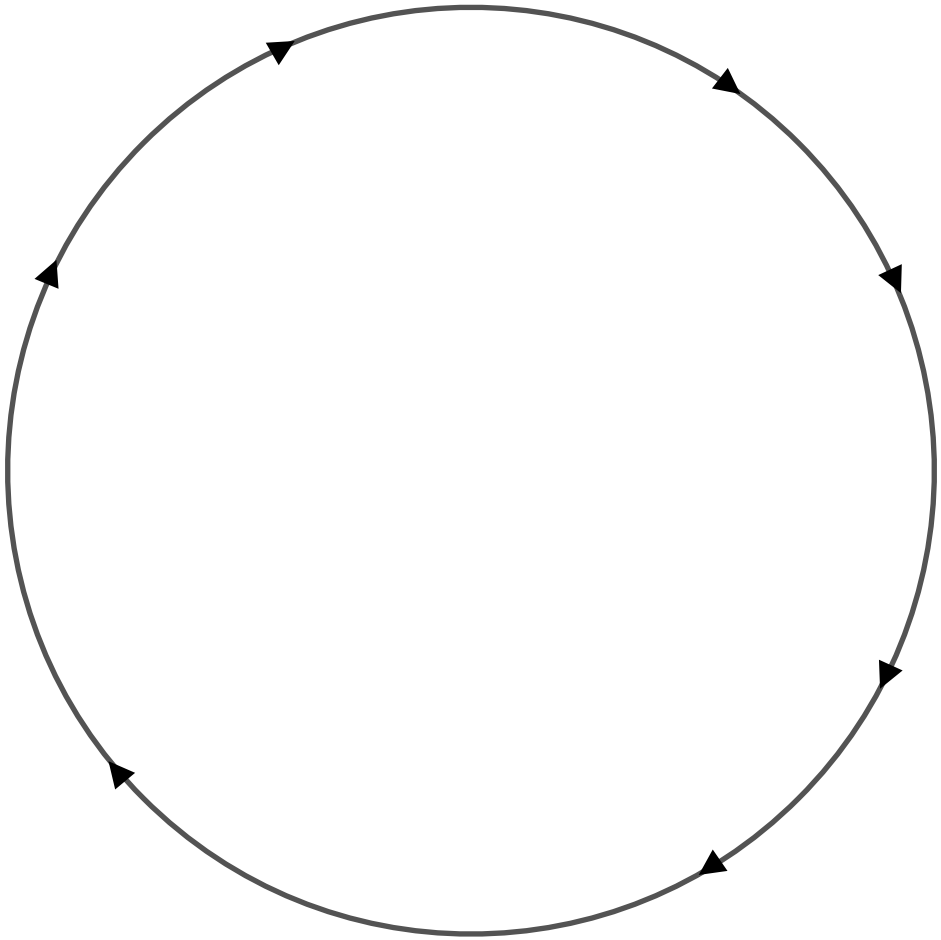


- Manages two MVCs at a time.
- It contains a master (left side) and detail view controller (right side)
- Split view controllers are supported on iPad and iPhone for iOS 8

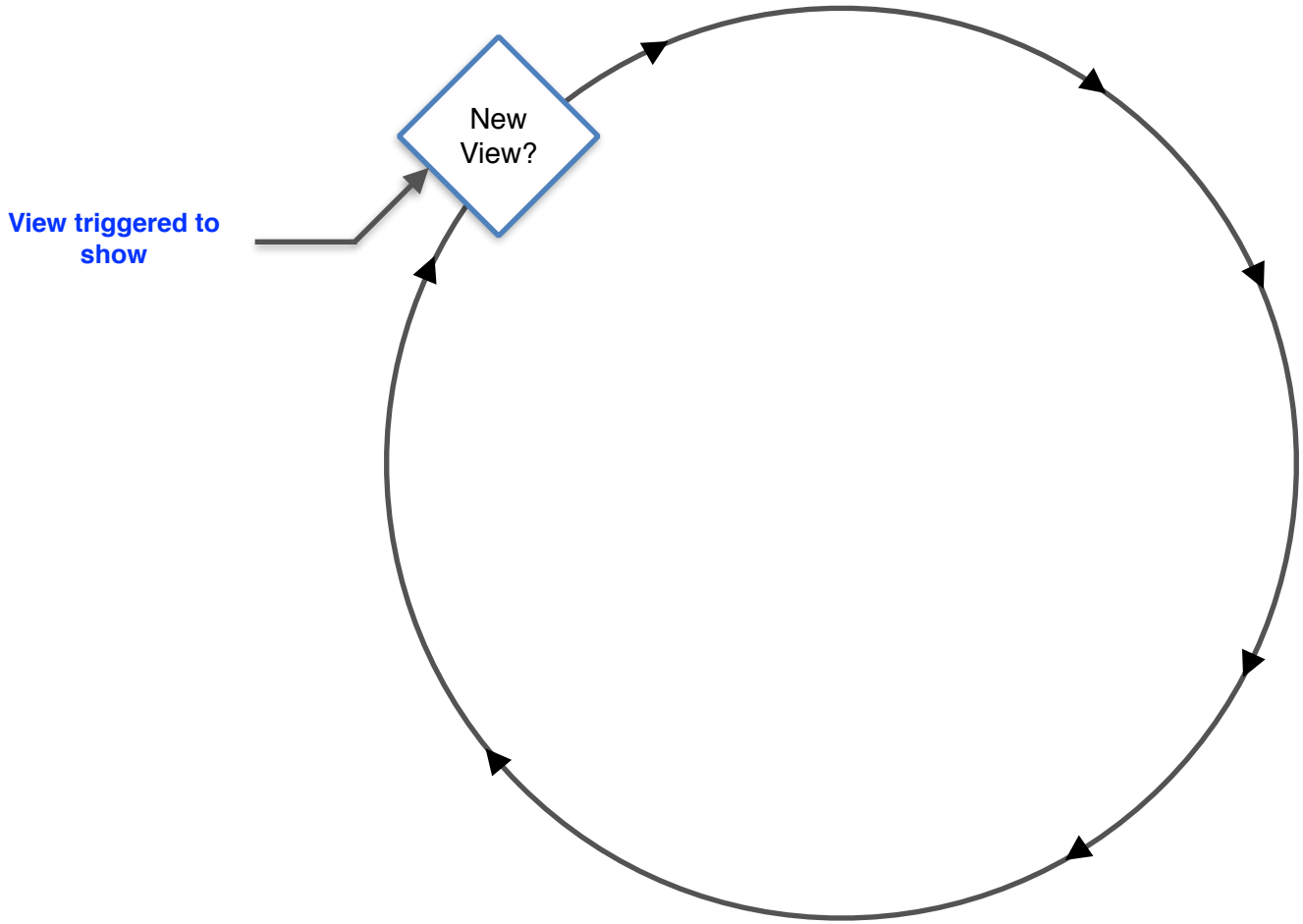
View Controllers

View Controller Lifecycle

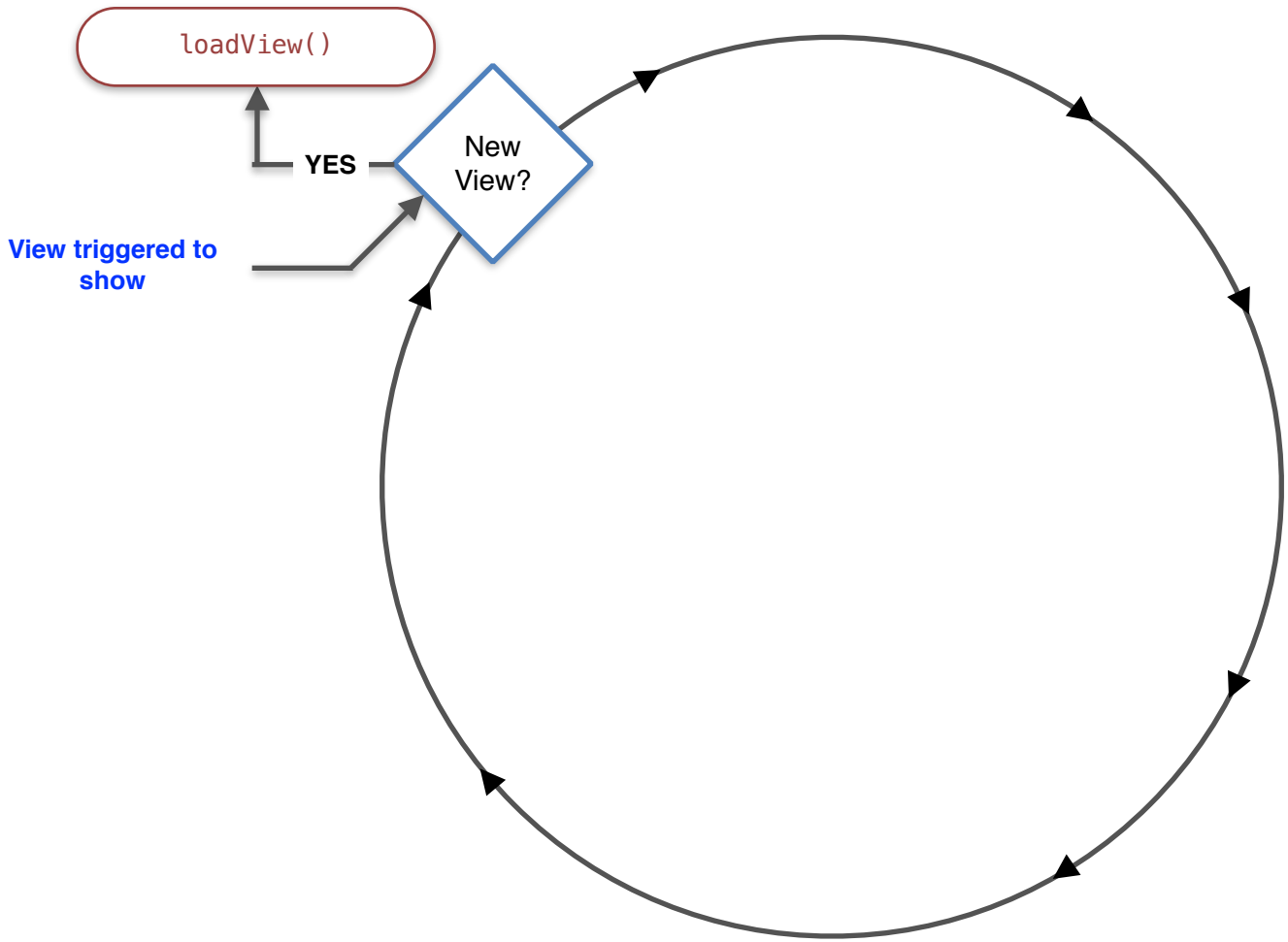
View Controller Life Cycle



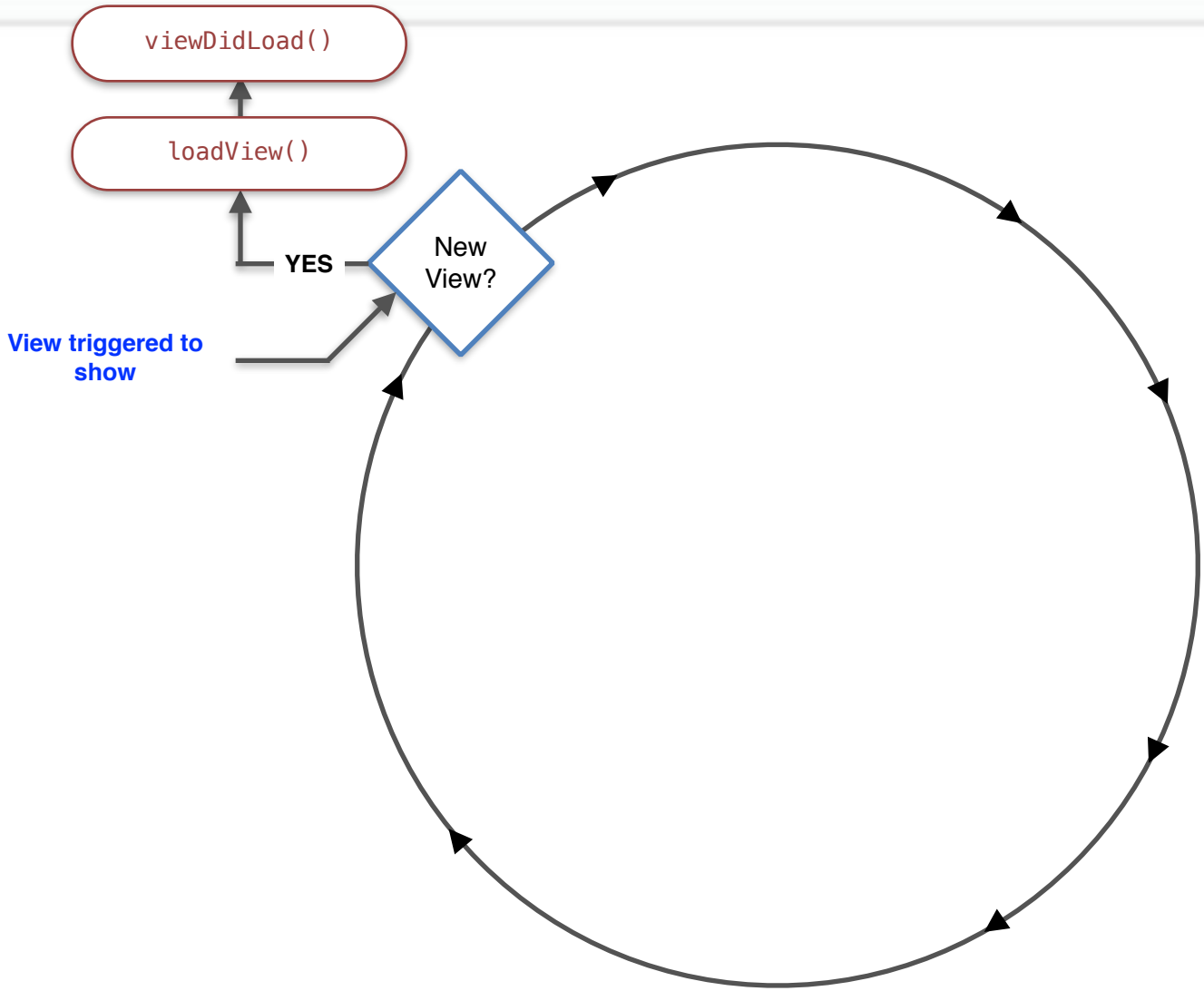
View Controller Life Cycle



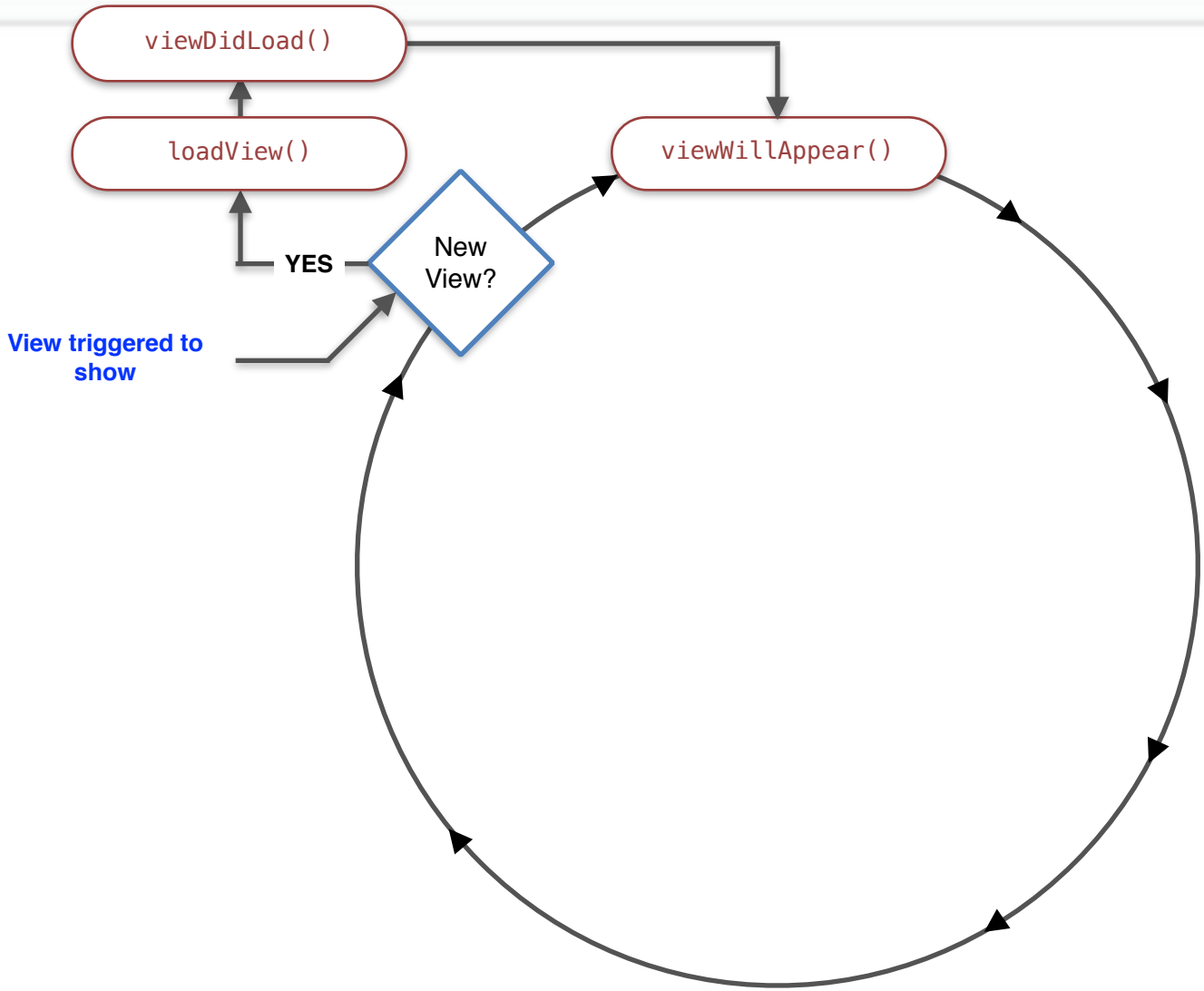
View Controller Life Cycle



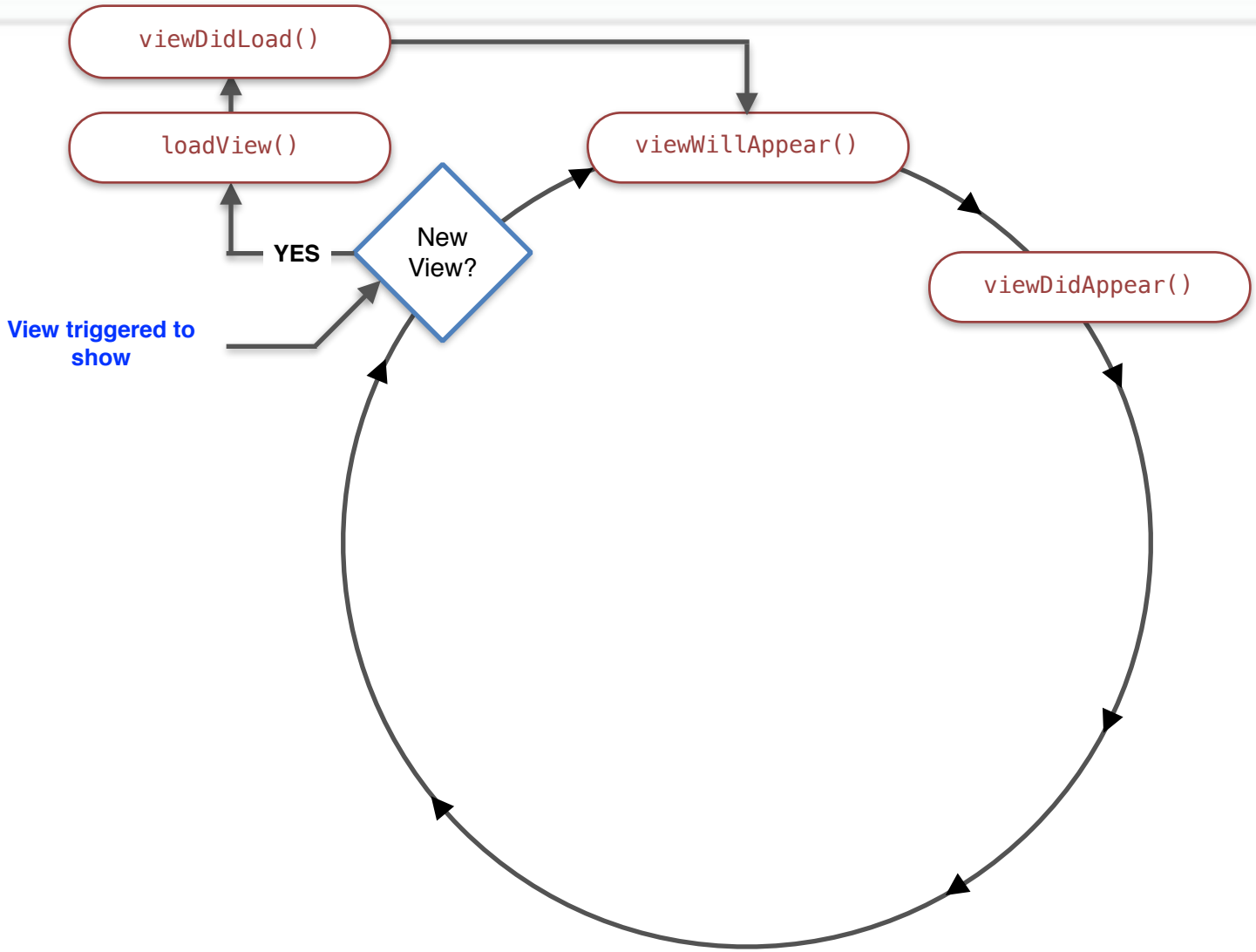
View Controller Life Cycle



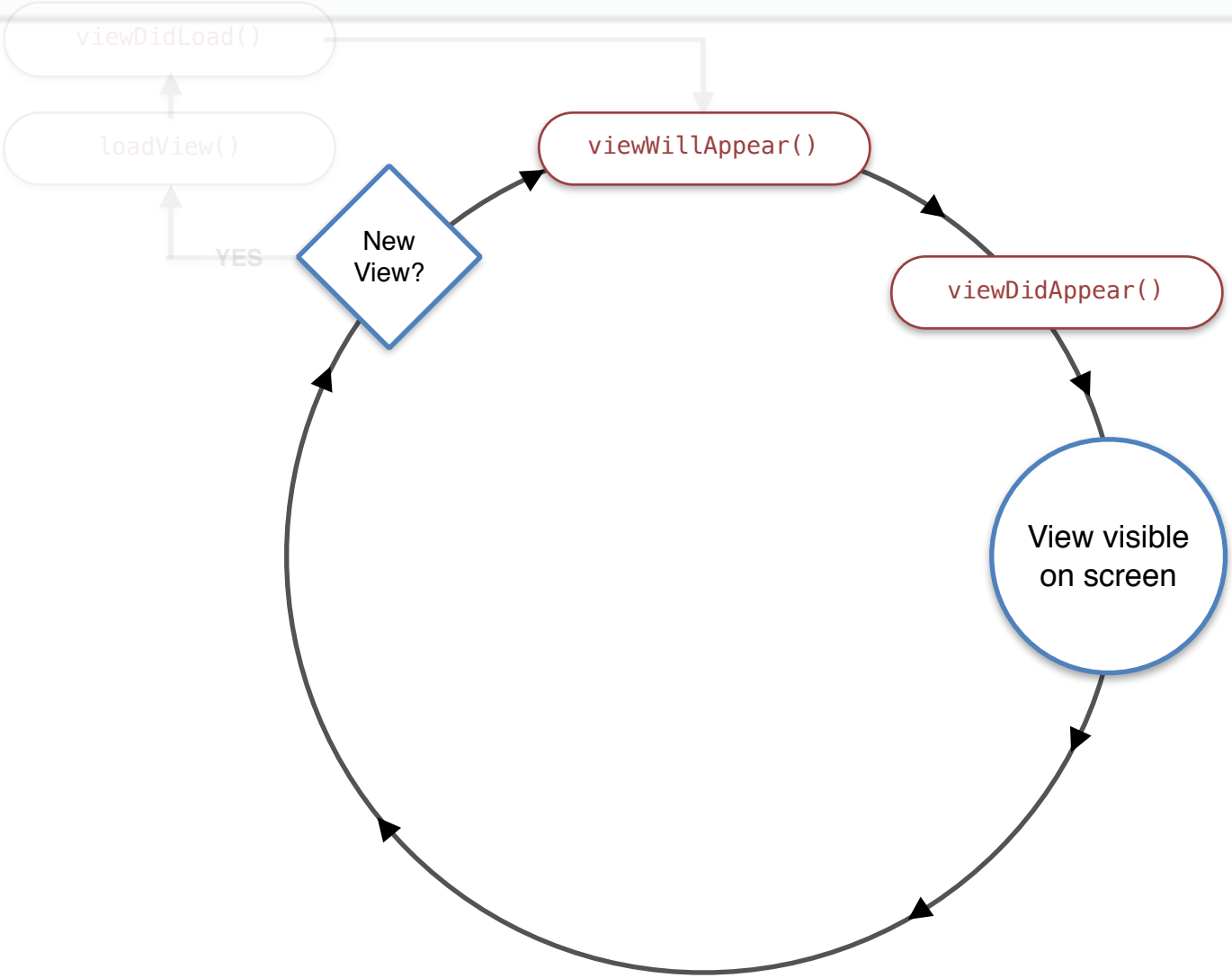
View Controller Life Cycle



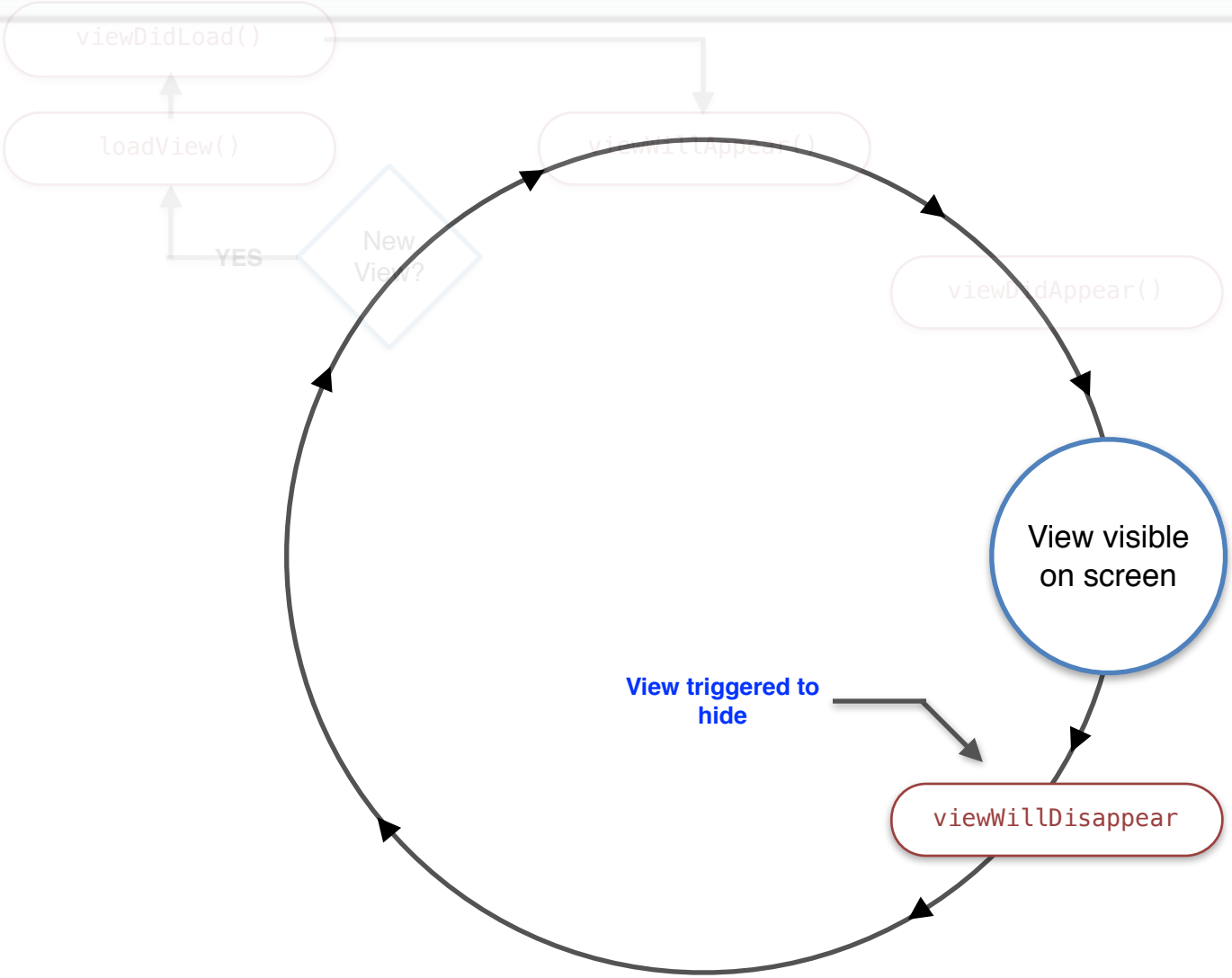
View Controller Life Cycle



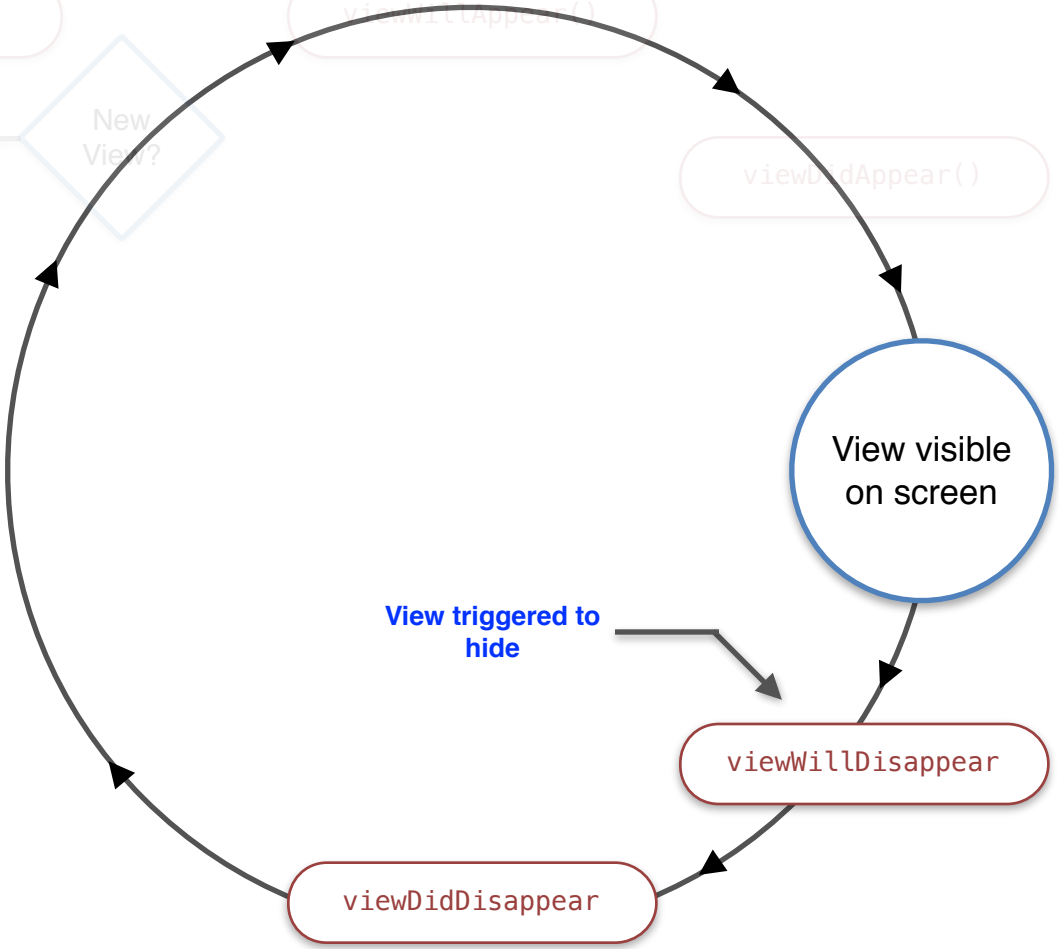
View Controller Life Cycle



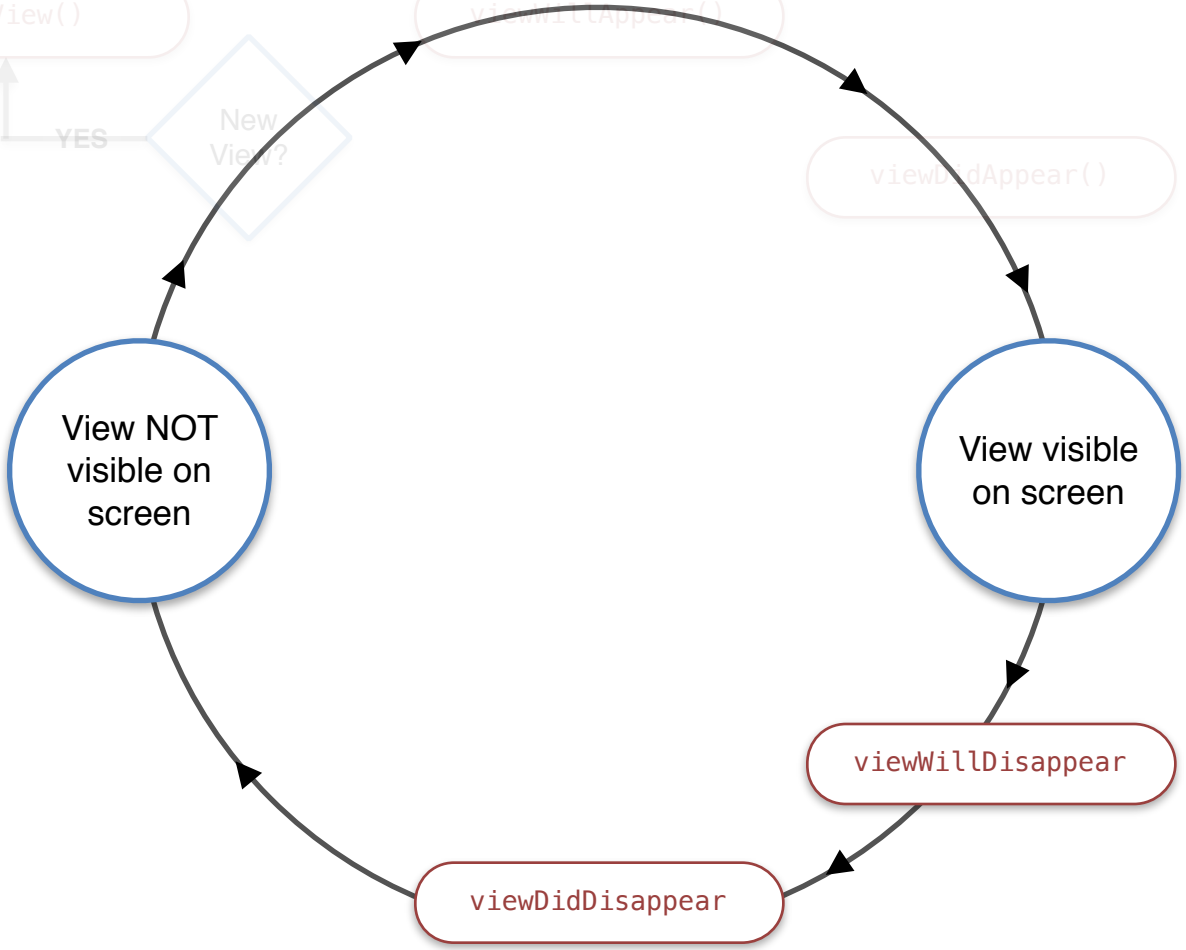
View Controller Life Cycle



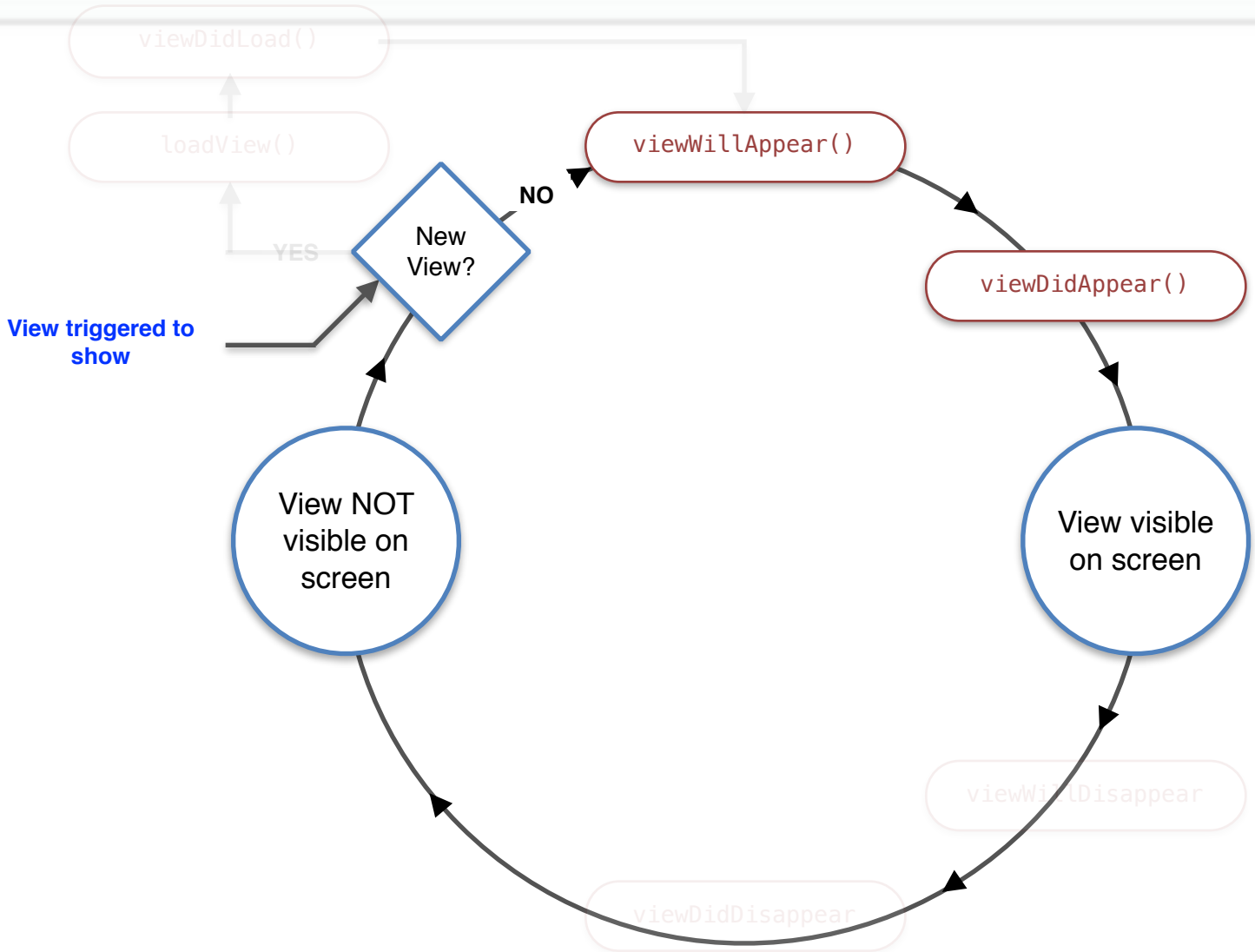
View Controller Life Cycle



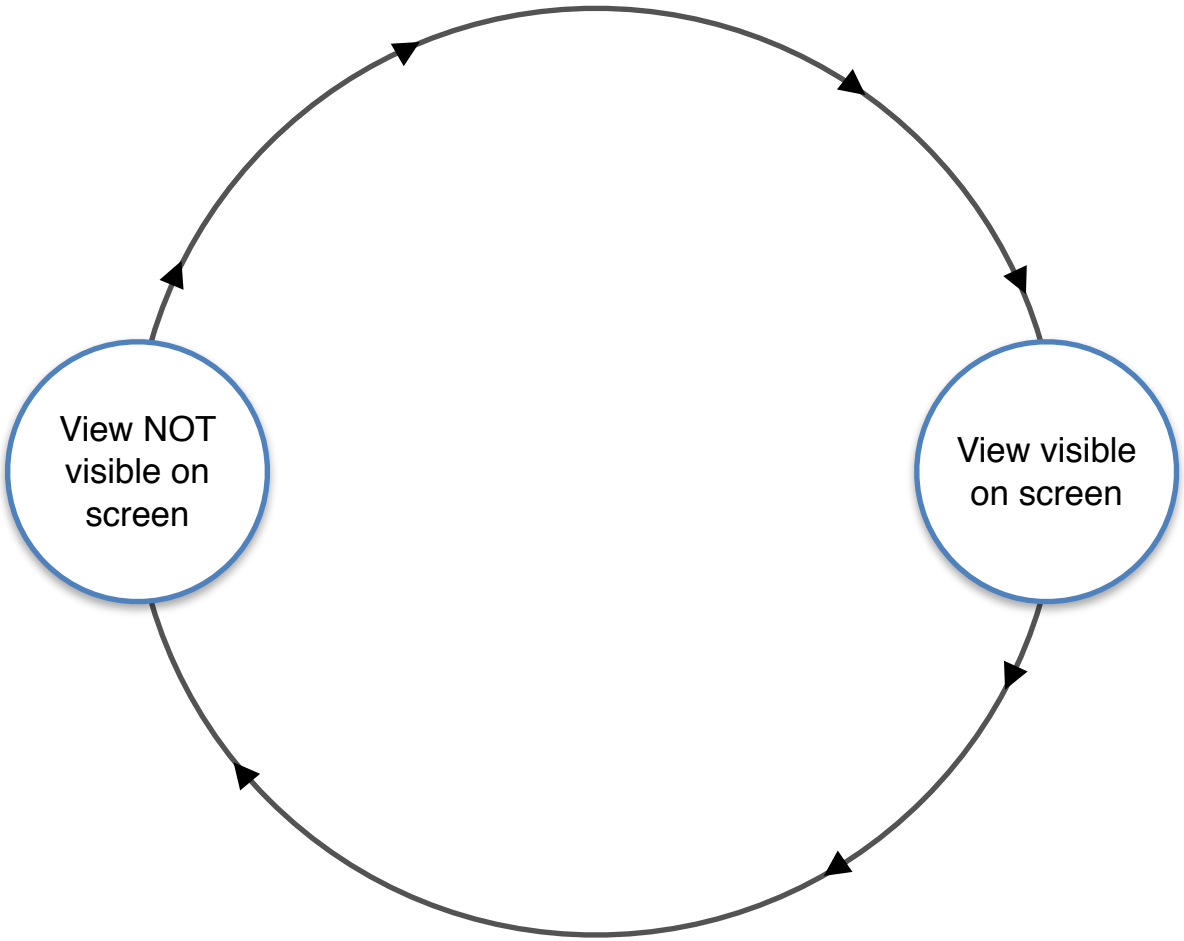
View Controller Life Cycle



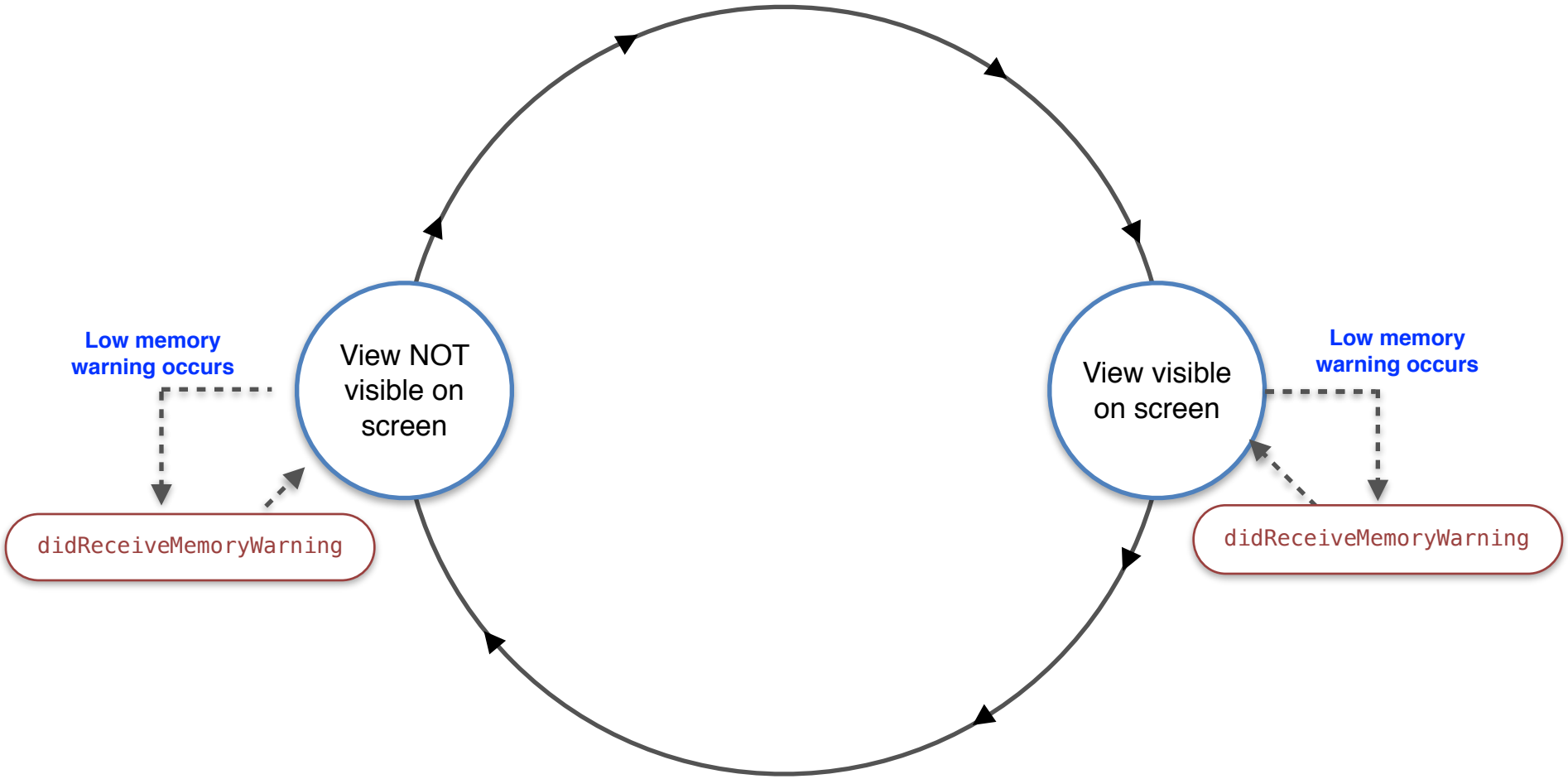
View Controller Life Cycle



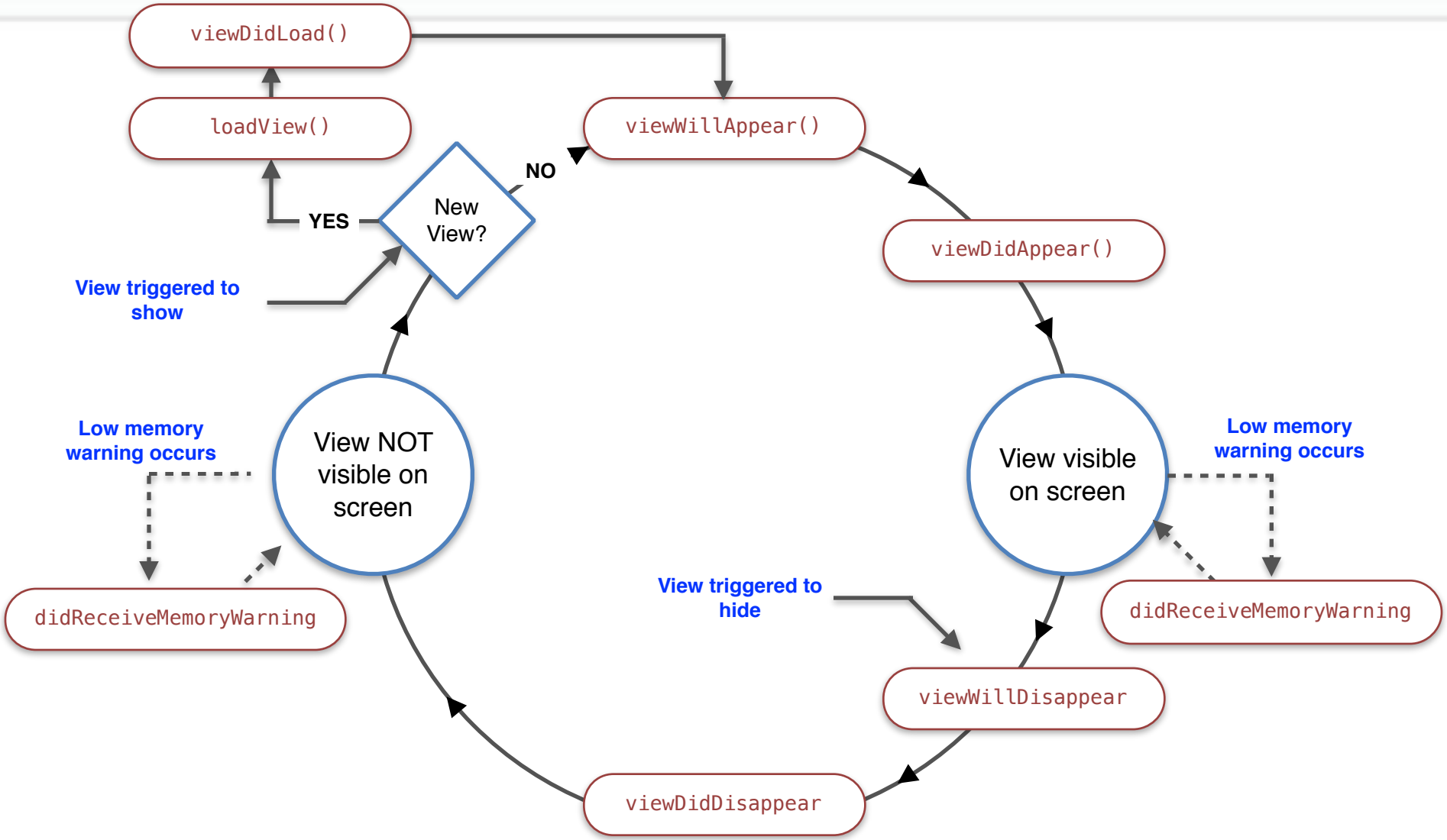
View Controller Life Cycle



View Controller Life Cycle



View Controller Life Cycle



View Controller Life Cycle

Method for View Life cycle event	When called
didReceiveMemoryWarning:	When a controller receive a memory warning
viewDidAppear:	Just before a controller's view appear
viewDidDisappear:	Just before a controller's view disappear
viewDidLoad:	After a controller's view loads into memory
viewWillAppear:	When a controller's view will appear
viewWillDisappear:	When a controller's view will disappear
viewWillTransitionToSize	After a view controller's view rotate.
willTransitToTraitCollection	When the size class for your app changes on-the-fly. This can happen when you rotate an iPad, which will trigger both viewWillTransitionToSize and willTransitToTraitCollection.

viewDidLoad()

- **viewDidLoad** called after:
  - instantiation and
  - the outlets are set.

- As a rule-of-thumb, we will give the super class a chance to do any additional set up in all the life cycle methods given below.

- Use this to:
  - Perform one-time initialization.

```
override func viewDidLoad() {
    super.viewDidLoad()
    // Do any additional setup after loading the view, typically from a
        nib.
}
```

viewWillAppear()

- **viewWillAppear** is called when:
  - the view is about to appear on the screen.
  - The animated parameter specifies whether you want the view to appear instantly or in an animated manner.
- Use this to:
  - initialise anything before the view appears.

```
override func viewWillAppear(animated: Bool) {
    super.viewWillAppear(animated)
}
```

viewWillDisappear()

- **viewWillDisappear** is called when:
  - the view is about to disappear from the screen.

- Use this to:
  - save the data in the view or
  - save the view state can be written here.
  - any other clean up.

```
override func viewWillDisappear(animated: Bool) {
    super.viewWillDisappear(animated)
}
```

viewDidAppear()

- **viewDidAppear** is called when:
  - the view is just before appearing on the screen.

- Use this to:
  - perform additional tasks associated with presenting the view.

```
override func viewDidAppear(animated: Bool) {
    super.viewDidAppear(animated)
}
```

viewDidDisappear()

- **viewDidDisappear** is called when:
  - the view is just before disappearing from the screen.

- Use this to:
  - perform additional tasks associated with dismissing / hiding the view.

```
override func viewDidDisappear(animated: Bool) {
    super.viewDidDisappear(animated)
}
```

didReceiveMemoryWarning()

- **didReceiveMemoryWarning** called when:
  - the application runs out of memory
- If your app loads too many images or media into memory, you may receive this call.
- Use this to:
  - Set references to large, unused media to nil
  - Release any large, unused resources

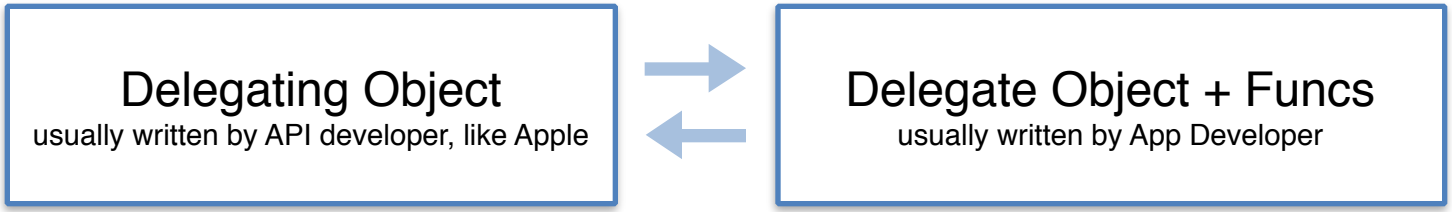
```
override func didReceiveMemoryWarning() {
    super.didReceiveMemoryWarning()
    // Dispose of any resources that can be recreated.
}
```

View Controllers

Delegates and Data Source

Delegates

- A simple and powerful pattern:

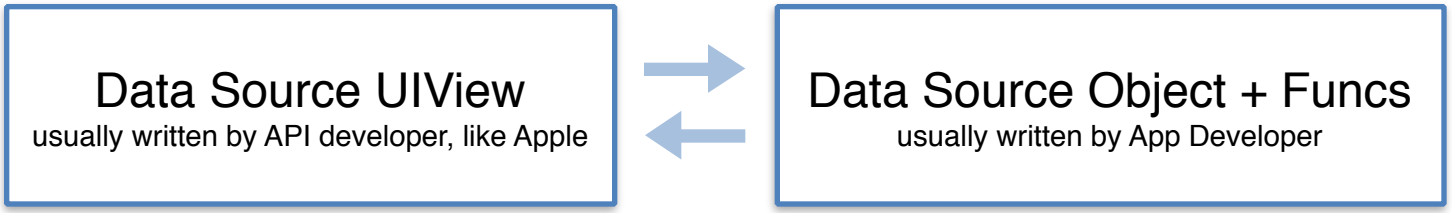


- The delegating object keeps a reference to the delegate and at the appropriate event, calls the method.

- The delegate may respond by:
  - Updating the UI, other objects, or
  - Advice if an event should happen.

Data Source

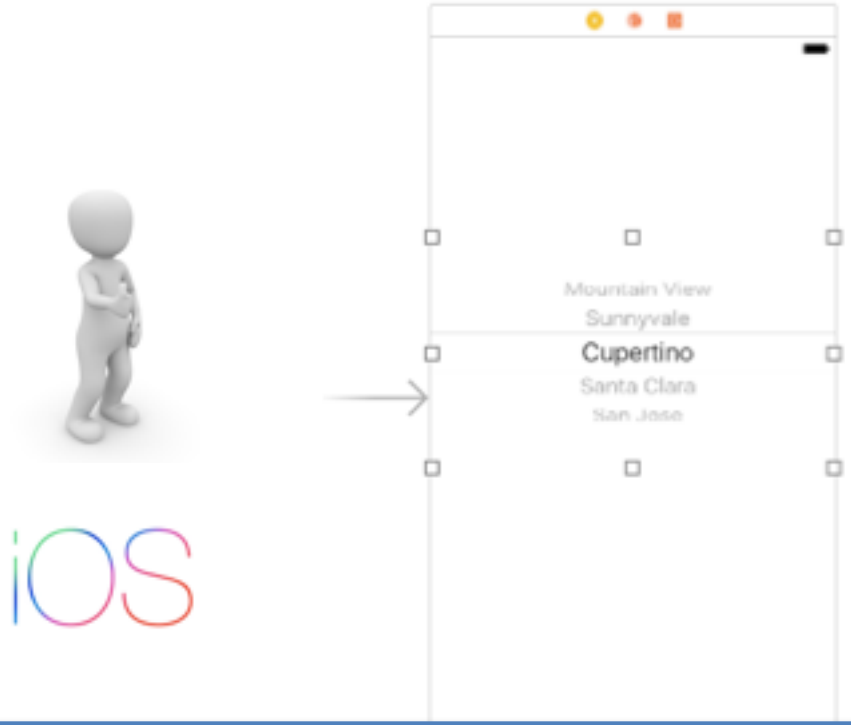
- Similar to delegates:



- The delegating object keeps a reference to the Data Source object and calls the method when it requires data from the app to show in the UI.

- The Data Source must respond by:
  - Providing the data required.

Delegates



This Storyboard has a UIPickerView.

Let's see what happens when I have to display this on the UI.

Delegates

So you, the dev, want a UIPickerView? I'll do it!

But first, tell me how many slots (components) there are?

There are 2 components

iOS

Delegates

iOS

Hmm, seems like the dev wants a picker view with two components (slots) like that:

Next, I have to know how many items are there in each component

Delegates

How many items in component number 1?

2 items

How many items in component number 2?

3 items

iOS

Delegates

Great, seems like the developer wants a Picker that looks like that:

But I still need more information.

Delegates

Ok, in component 1, what should I display as the text for item 1?

"Weekday"

Component 1, item 2?

iOS

"Weekend"

Delegates

iOS

Fantastic! I now know enough to display this:

Weekday	?????
Weekend	?????
	?????

Just a little more info…

Delegates

Now, in component 2, what should I display as the text for item 1?

"Morning"

Component 2, item 2?

"Afternoon"

iOS

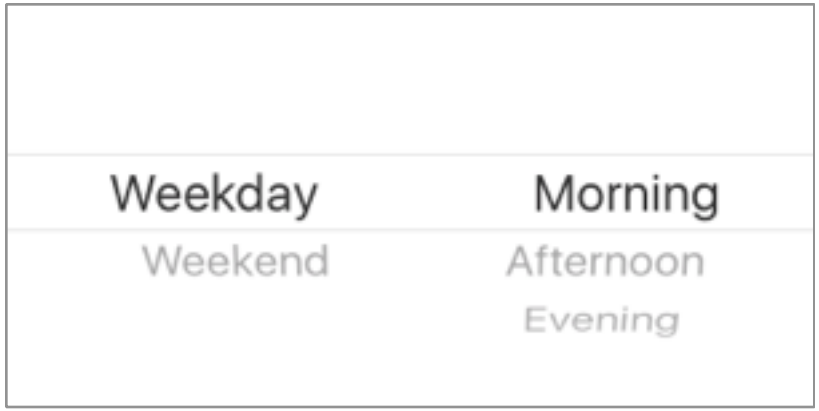
Component 2, item 3?

"Evening"

Delegates

iOS

Perfect! So this is what I am going to display!



Example of PickerView Delegation

PickerView

PickerViewDelegate

```
func pickerView(
    _ pickerView: UIPickerView,
    titleForRow row: Int,
    forComponent component: Int) -> String?
```

PickerViewDataSource

```
func numberOfComponents(
    in pickerView: UIPickerView) -> Int

func pickerView(
    _ pickerView: UIPickerView,
    numberOfRowsInComponent component: Int)
    -> Int
```

The app developer will need to implement these methods in his/her app to “answer” iOS questions about how the PickerView’s contents should look like.

Summary

- Quick Look at Xcode UI
- Model-View-Controller pattern
- Basics of View Controller
- Different types of controllers
- View Controller Life Cycle
- Storyboard
- Delegates