THE UNIVERSITY OF CHICAGO

ADVANCED iOS APPLICATION DEVELOPMENT

MPCS 51032 • SPRING 2020 • SESSION 1A

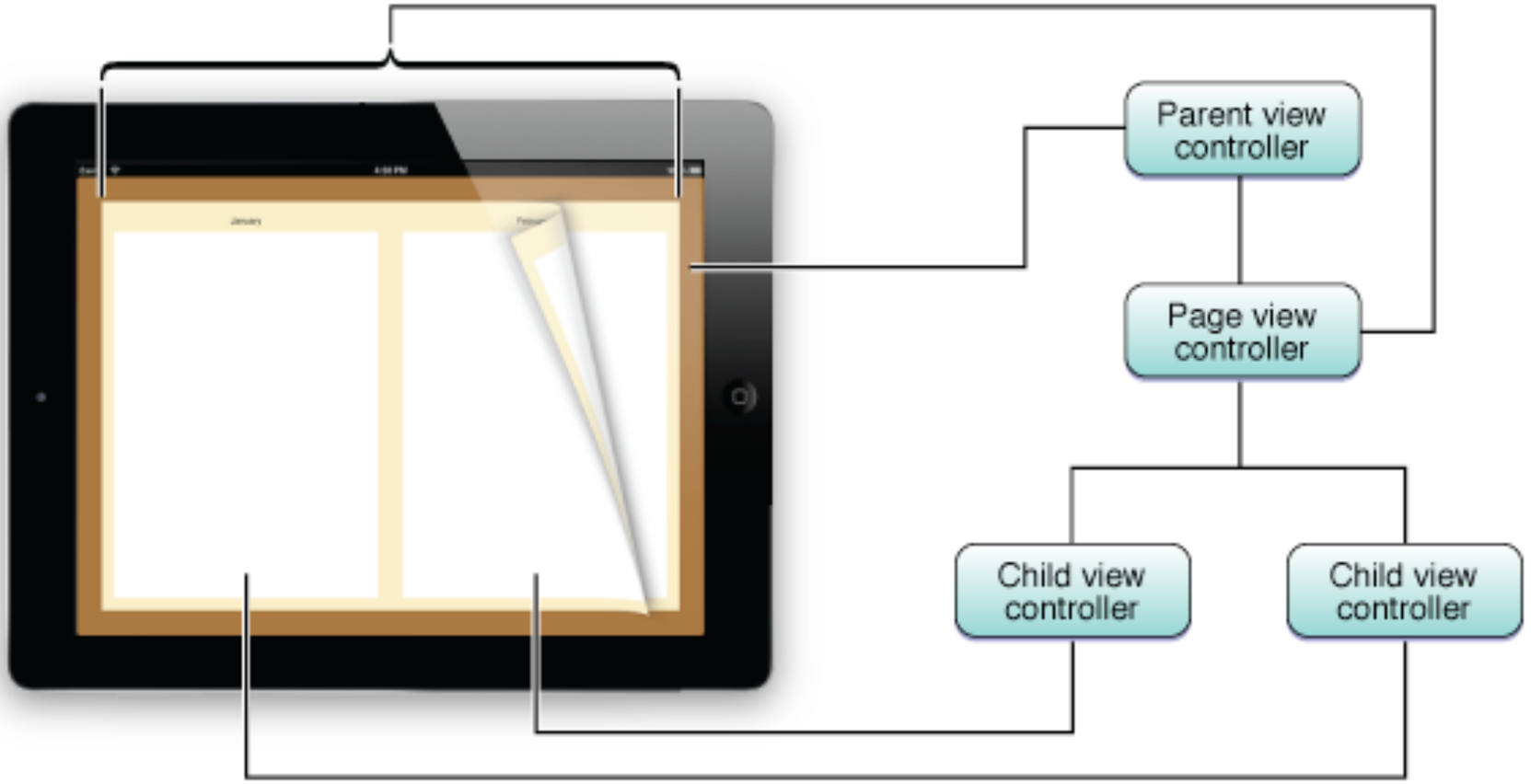
PAGE VIEW CONTROLLER

PAGE VIEW CONTROLLER

- Container view controller released in iOS5
- Present content in a page-by-page manner
  - Navigate among views with a page curl transition
  - Manages a self-contained view hierarchy
  - Manages child view controllers that present the content

PAGE VIEW CONTROLLER

- Parent view of this hierarchy is managed by the page view controller
- Child views are managed by the content view controllers that you provide



PAGE VIEW CONTROLLER

- Navigation can be controlled
  - Programmatically by your app
  - Directly by the user using gestures
- When navigating from page to page uses the transition that you specify to animate the change

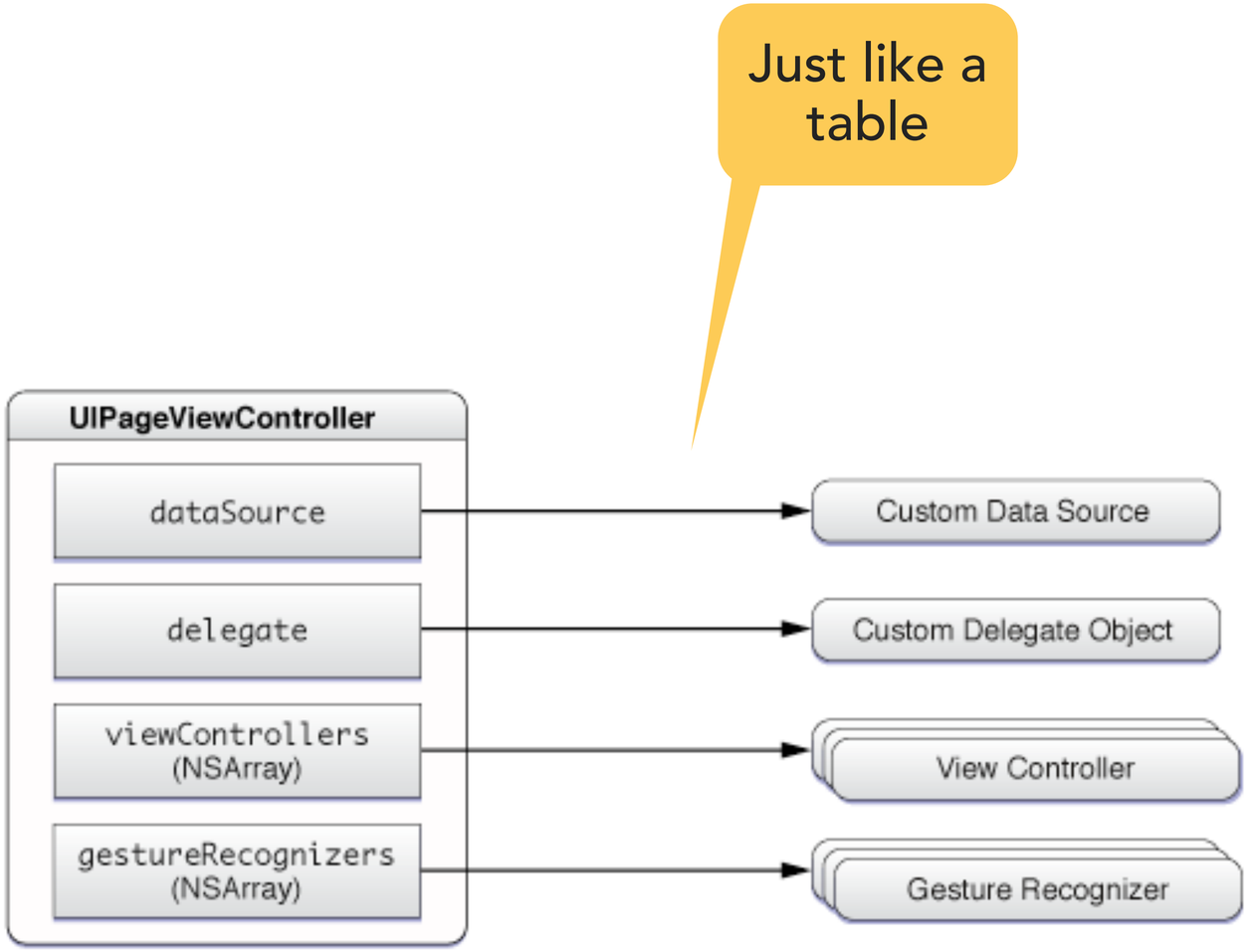
PAGE VIEW CONTROLLER

- Provide the content view controllers
  - One at a time
  - Two at a time
  - As-needed using a data source
- When providing content view controllers one at a time use the setViewControllers:direction:animated:completion: method to set the current content view controllers.
- To support gesture-based navigation, you must provide your view controllers using a data source object

PAGE VIEW CONTROLLER

UIPAGEVIEWCONTROLLER API

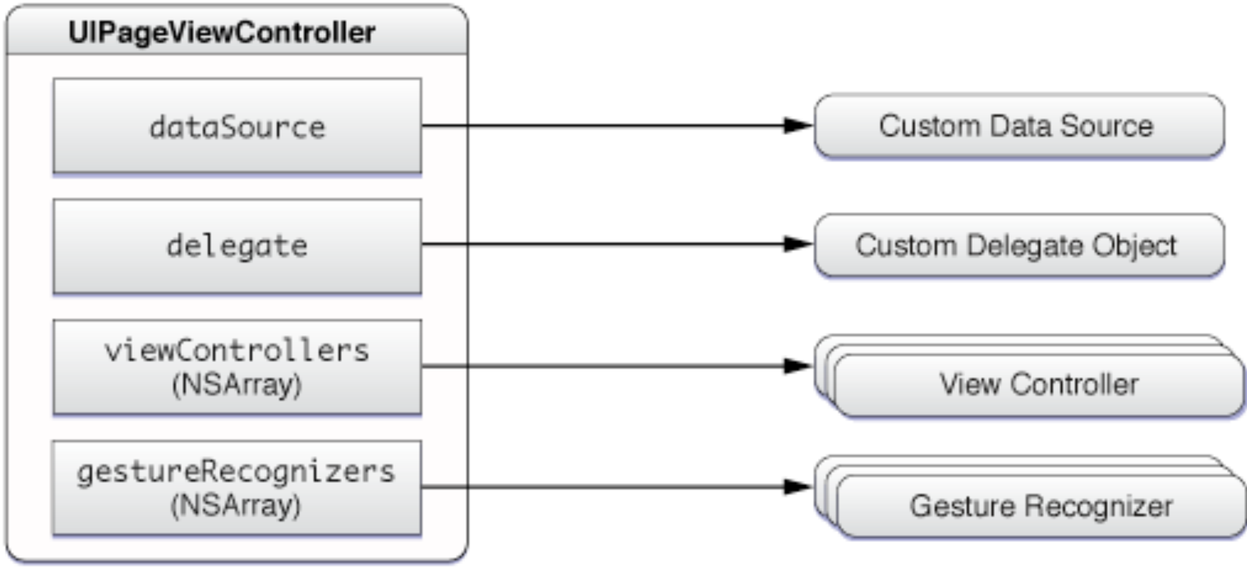
- A page view interface consists of the following objects
  - An optional delegate
  - An optional data source
  - An array of the current view controllers
  - An array of gesture recognizers



PAGE VIEW CONTROLLER

UIPAGEVIEWCONTROLLER API

- The data source is responsible for providing the content view controllers on demand
  - Must conform to the `UIPageViewControllerDataSource` protocol

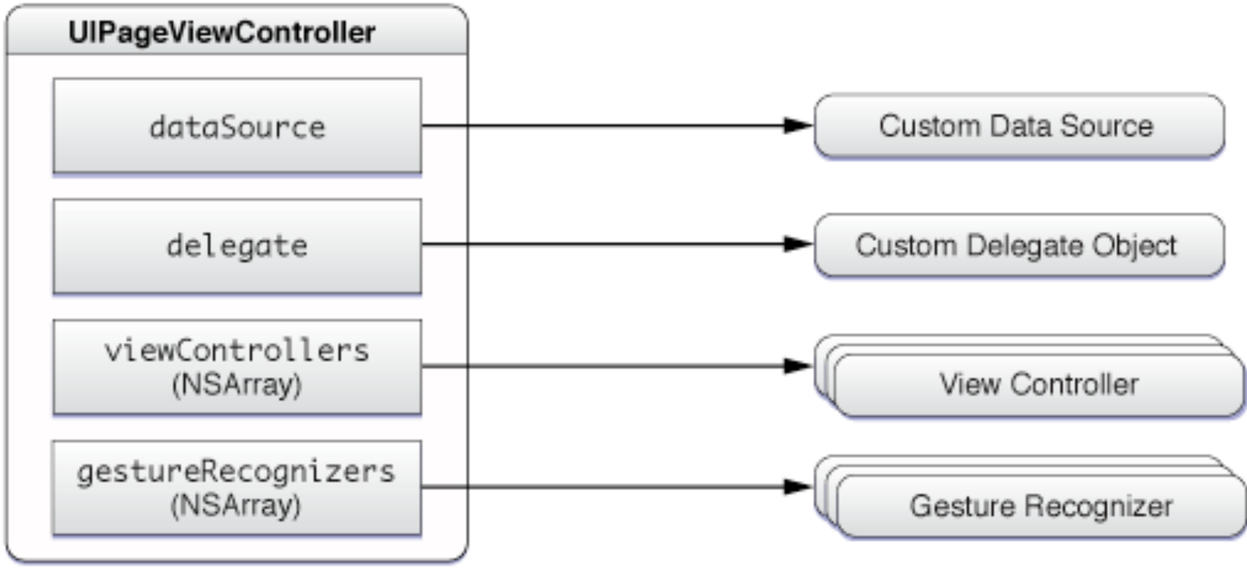


Reusing view controllers similar to table cells

PAGE VIEW CONTROLLER

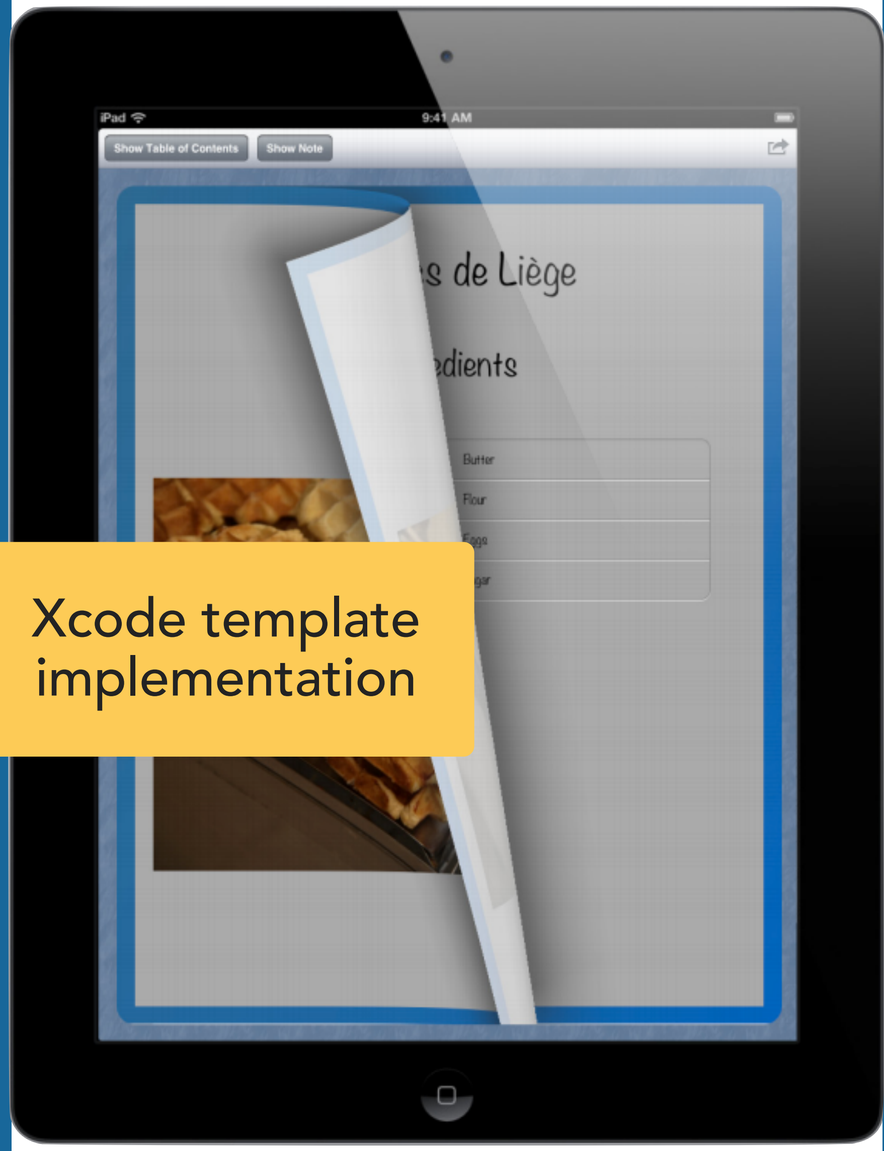
UIPAGEVIEWCONTROLLER API

- The delegate object
  - Provides some appearance-related information
  - Provides methods called in response to orientation changes
  - Receives notifications about gesture-initiated transitions



PAGE VIEW CONTROLLER

- Page view controller can be used in a wide variety of settings
- A page view controller's view can be resized and embedded in a view hierarchy
  - Unlike a navigation or tab bar controller
- Can be a window's root view controller



UIPAGEVIEWCONTROLLER

UIPAGEVIEWCONTROLLER

INITIALIZATION

transition style

```
// Configure the page view controller and add it as a child view controller.
self.pageViewController = UIPageViewController(transitionStyle: .PageCurl, navigationOrientation: .Horizontal, options: nil)
self.pageViewController!.delegate = self
```

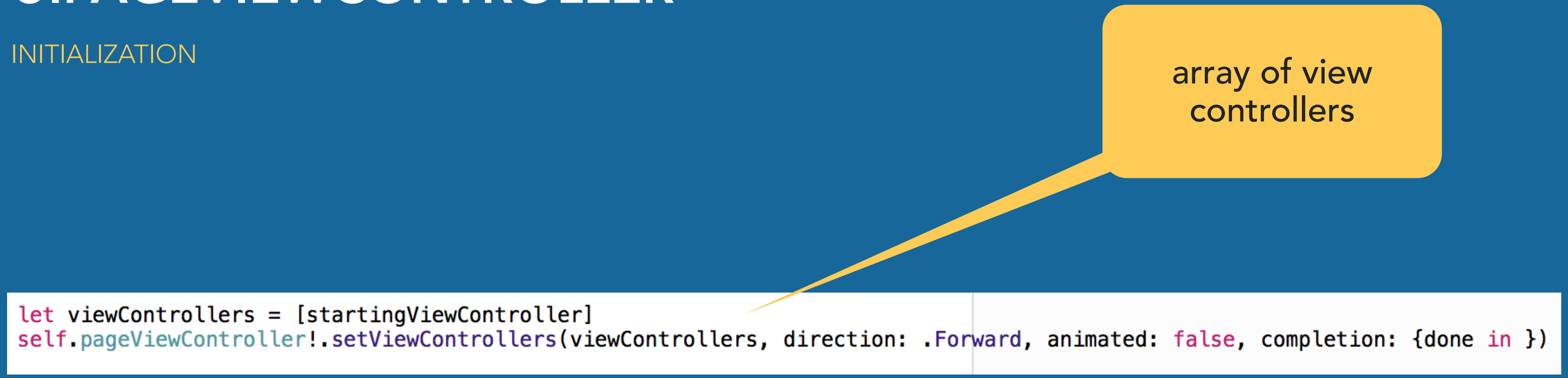
page flip direction

page spine

- Initialization

UIPAGEVIEWCONTROLLER

INITIALIZATION



- Initial view controllers
  - Use 2 view controllers for mid-spine look

UIPAGEVIEWCONTROLLER

INITIALIZATION

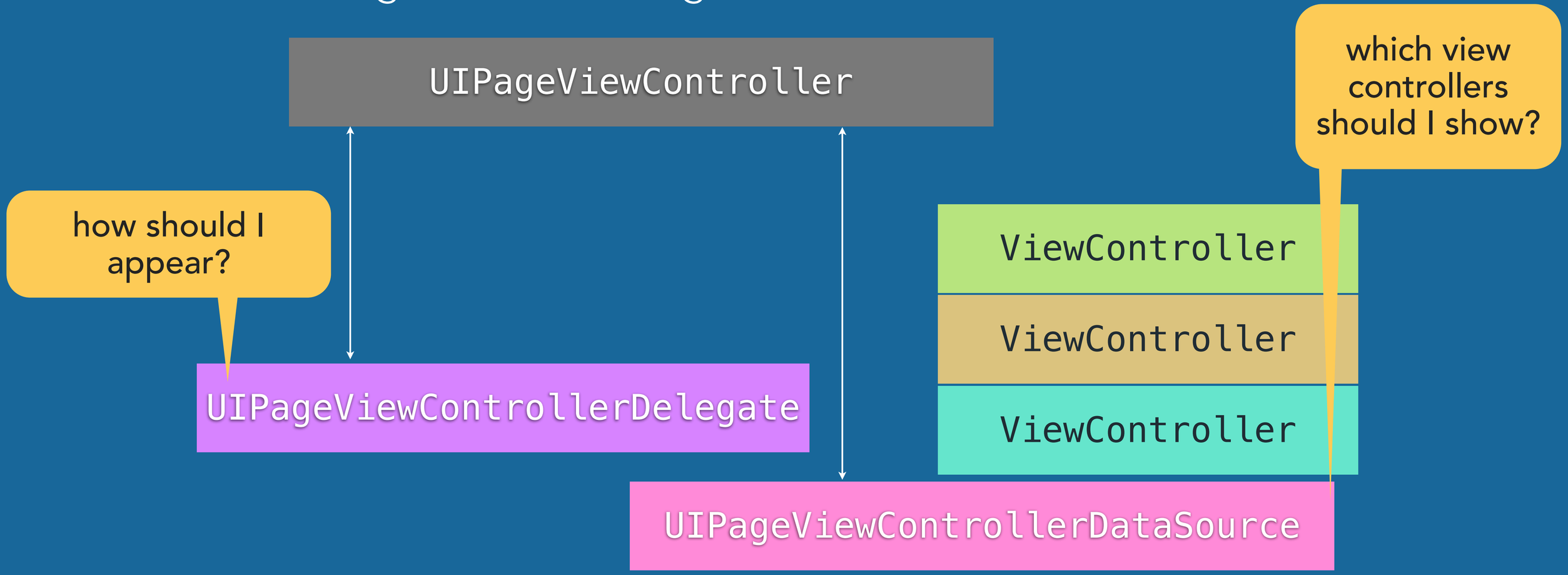
```
let viewControllers = [currentViewController]
self.pageViewController!.setViewControllers(viewControllers, direction: .Forward, animated: true, completion: {done in })
```

- Can also call `setViewControllers` programmatic navigation
  - Go directly to a different ViewController out of order

UIPAGEVIEWCONTROLLER

INITIALIZATION

- User-driven navigation uses delegate and data source

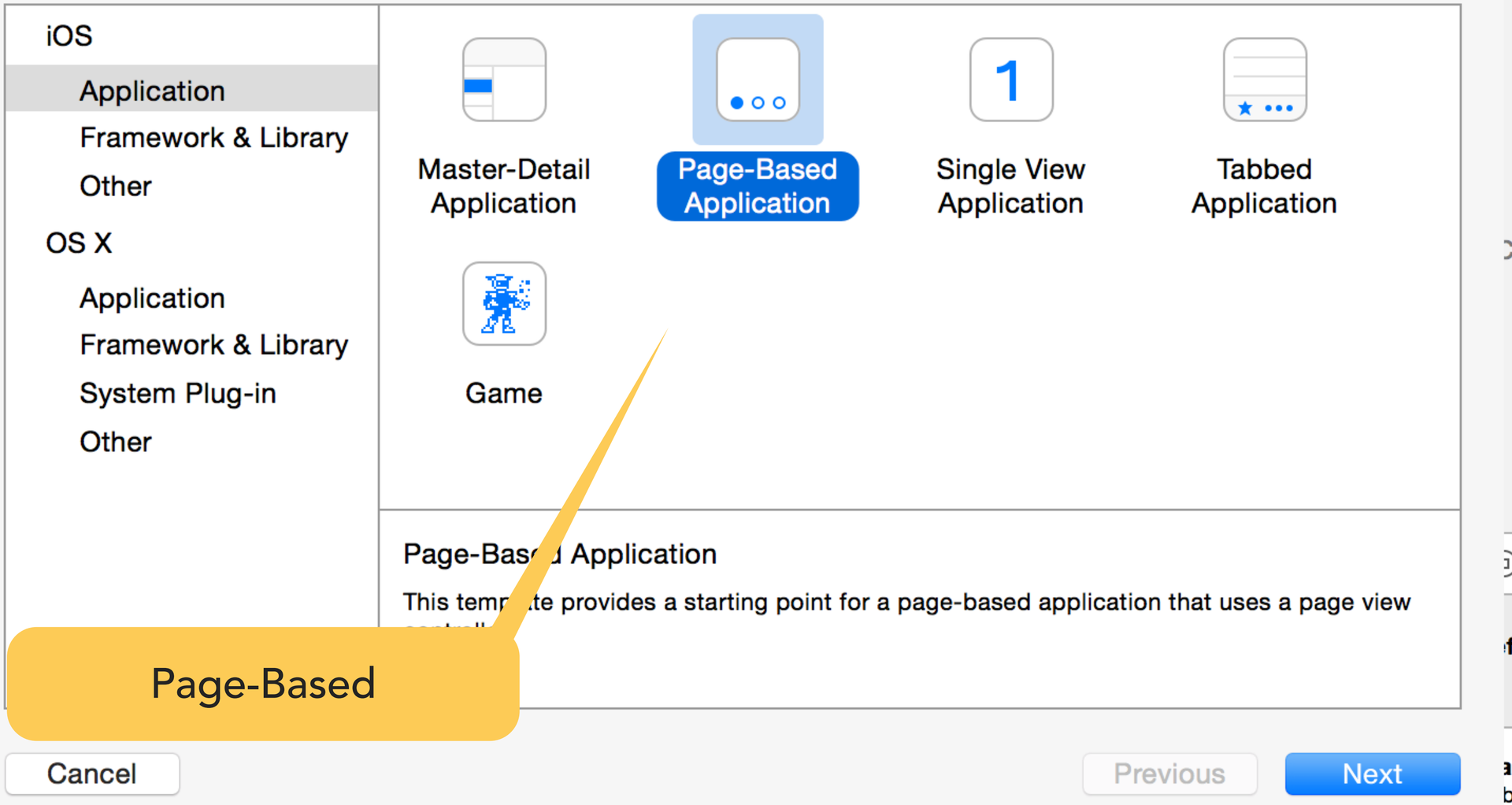


A SIMPLE PAGE VIEW CONTROLLER EXAMPLE

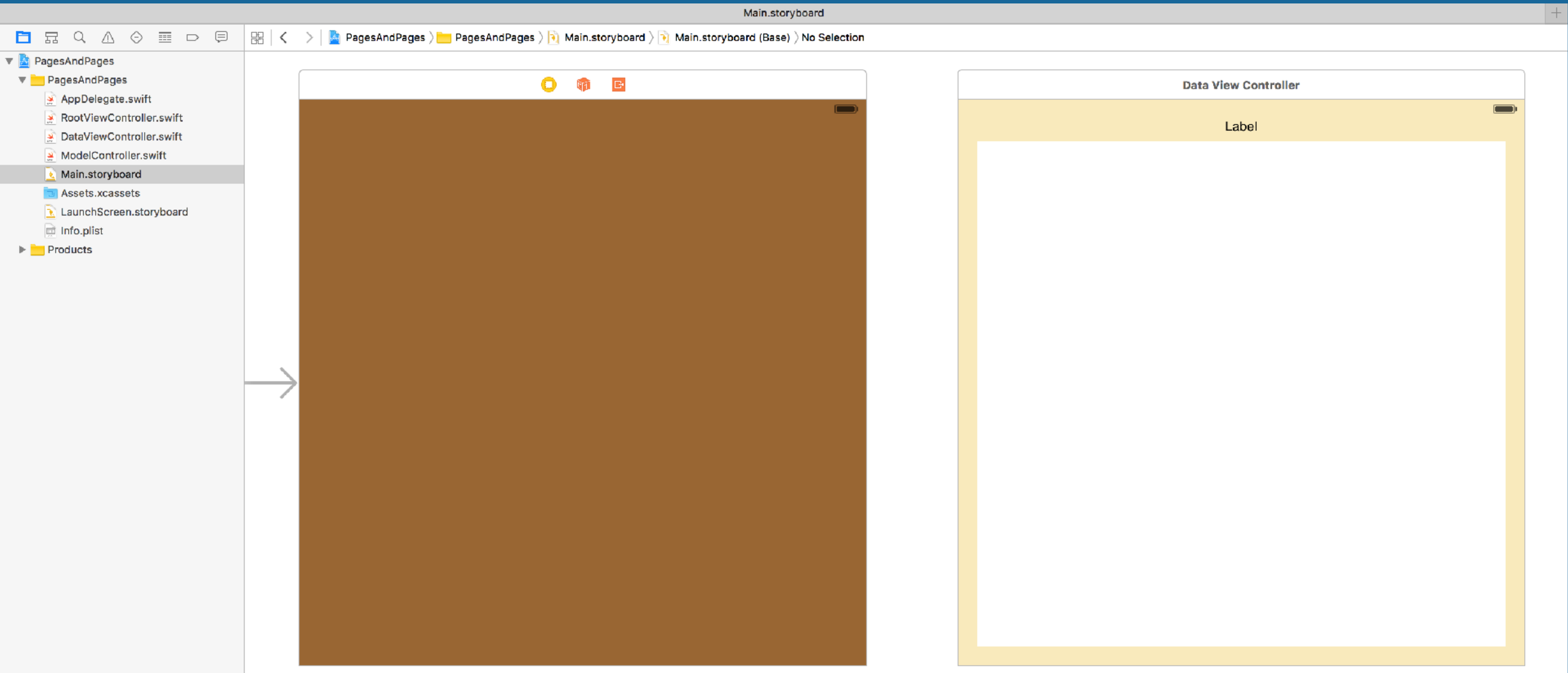
ONCE UPON A TIME...

PAGE VIEW CONTROLLER TEMPLATE

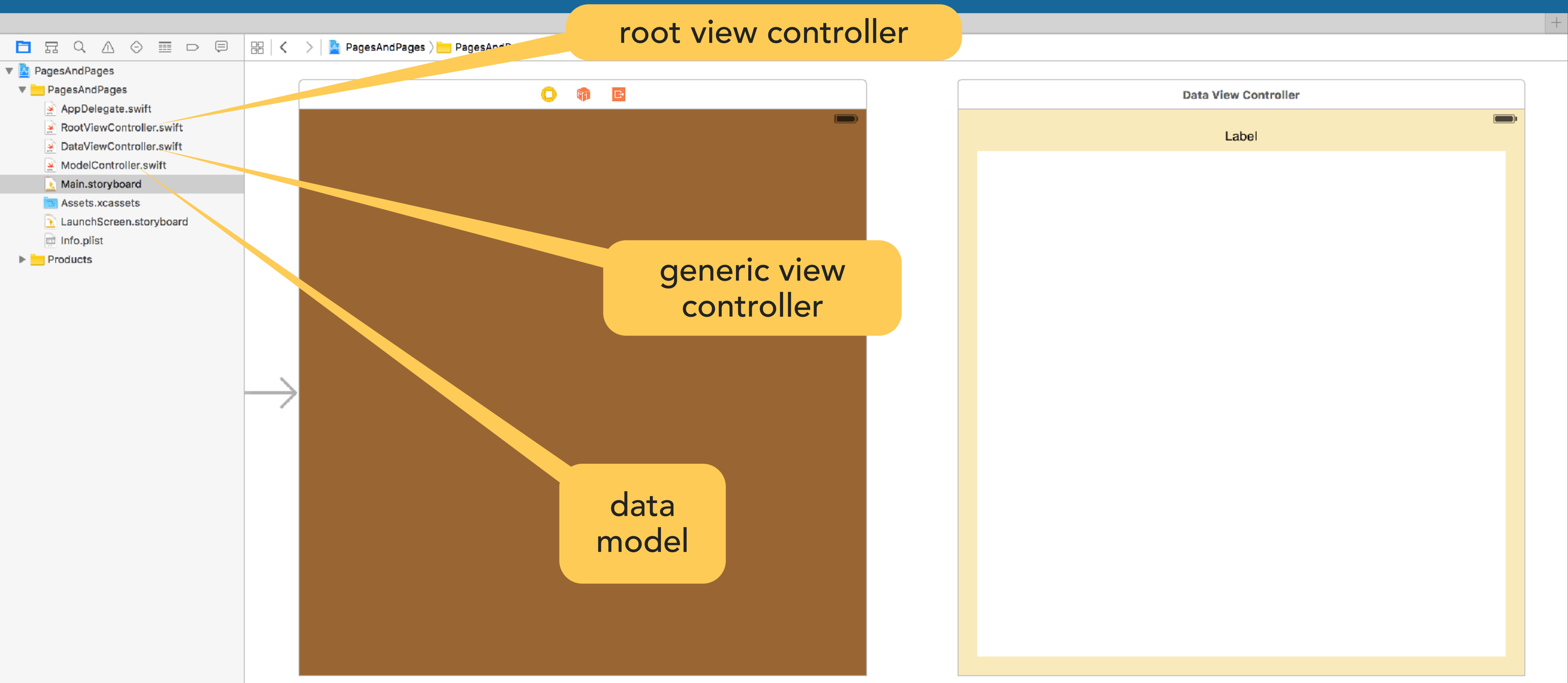
PAGE VIEW CONTROLLER TEMPLATE



PAGE VIEW CONTROLLER TEMPLATE

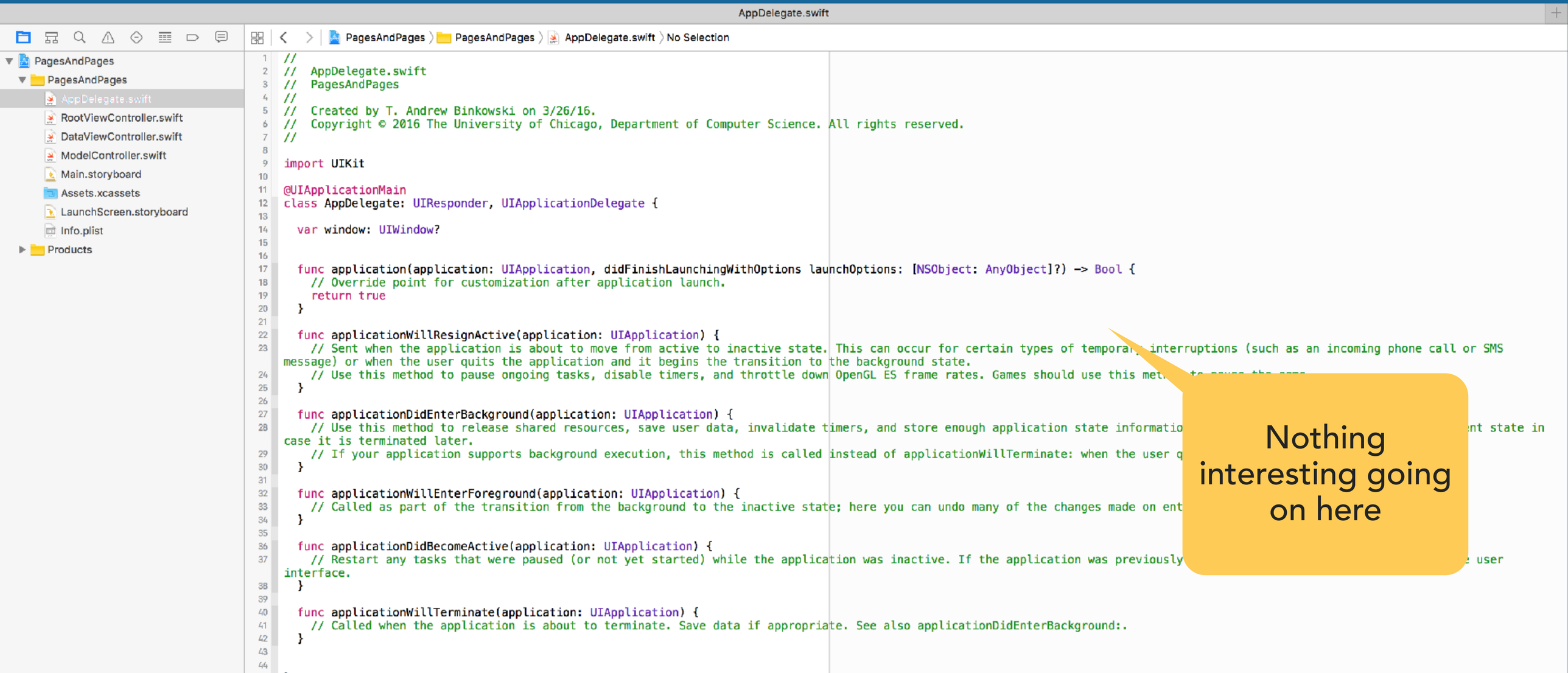


PAGE VIEW CONTROLLER TEMPLATE



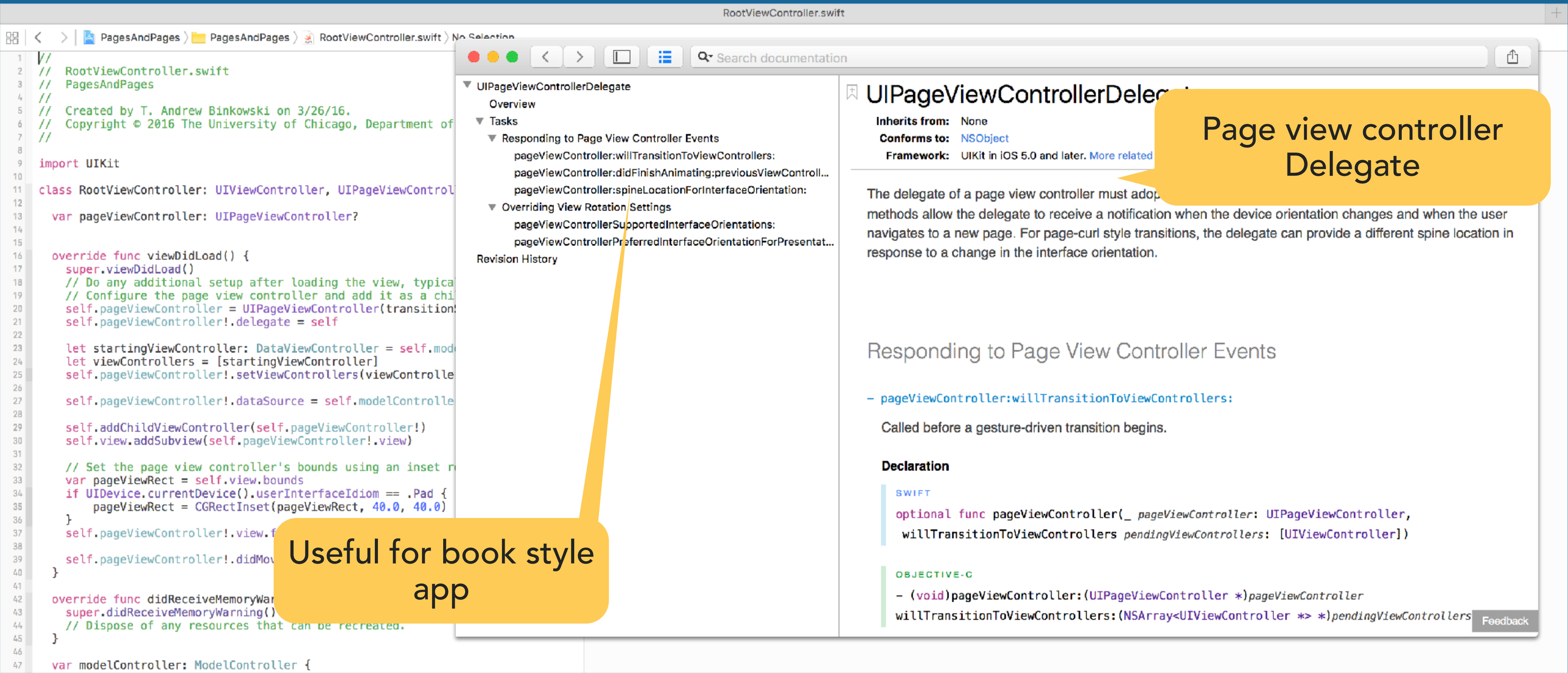
PAGE VIEW CONTROLLER TEMPLATE

ROOT VIEW CONTROLLER



PAGE VIEW CONTROLLER TEMPLATE

ROOT VIEW CONTROLLER



PAGE VIEW CONTROLLER TEMPLATE

ROOT VIEW CONTROLLER

Initialize the page view controller

```
override func viewDidLoad() {
  super.viewDidLoad()
  // Do any additional setup after loading the view, typically from a nib.
  // Configure the page view controller and add it as a child view controller.
  self.pageViewController = UIPageViewController(transitionStyle: .pageCurl, navigationOrientation: .horizontal, options: nil)
  self.pageViewController!.delegate = self

  let startingViewController: DataViewController = self.modelController.viewControllerAtIndex(0, storyboard: self.storyboard!)!
  let viewControllers = [startingViewController]
  self.pageViewController!.setViewControllers(viewControllers, direction: .forward, animated: false, completion: {done in })

  self.pageViewController!.dataSource = self.modelController

  self.addChildViewController(self.pageViewController!)
  self.view.addSubview(self.pageViewController!.view)

  // Set the page view controller's bounds using an inset rect so that self's view is visible around the edges of the pages.
  var pageViewRect = self.view.bounds
  if UIDevice.current.userInterfaceIdiom == .pad {
      pageViewRect = pageViewRect.insetBy(dx: 40.0, dy: 40.0)
  }
  self.pageViewController!.view.frame = pageViewRect

  self.pageViewController!.didMove(toParentViewController: self)
}
```

PAGE VIEW CONTROLLER TEMPLATE

ROOT VIEW CONTROLLER

```
override func viewDidLoad() {
  super.viewDidLoad()
  // Do any additional setup after loading the view, typically from a nib.
  // Configure the page view controller and add it as a child view controller.
  self.pageViewController = UIPageViewController(transitionStyle: .pageCurl, navigationOrientation: .horizontal, options: nil)
  self.pageViewController!.delegate = self

  let startingViewController: DataViewController = self.modelController.viewControllerAtIndex(0, storyboard: self.storyboard!)!
  let viewControllers = [startingViewController]
  self.pageViewController!.setViewControllers(viewControllers, direction: .forward, animated: false, completion: {done in })

  self.pageViewController!.dataSource = self.modelController

  self.addChildViewController(self.pageViewController!)
  self.view.addSubview(self.pageViewController!.view)

  // Set the page view controller's bounds using an inset rect so that self's view is visible around the edges of the pages.
  var pageViewRect = self.view.bounds
  if UIDevice.current.userInterfaceIdiom == .pad {
      pageViewRect = pageViewRect.insetBy(dx: 40.0, dy: 40.0)
  }
  self.pageViewController!.view.frame = pageViewRect

  self.pageViewController!.didMove(toParentViewController: self)
}
```

set the initial view controller

PAGE VIEW CONTROLLER TEMPLATE

ROOT VIEW CONTROLLER

```
override func viewDidLoad() {
  super.viewDidLoad()
  // Do any additional setup after loading the view, typically from a nib.
  // Configure the page view controller and add it as a child view controller.
  self.pageViewController = UIPageViewController(transitionStyle: .pageCurl, navigationOrientation: .horizontal, options: nil)
  self.pageViewController!.delegate = self

  let startingViewController: DataViewController = self.modelController.viewControllerAtIndex(0, storyboard: self.storyboard!)!
  let viewControllers = [startingViewController]
  self.pageViewController!.setViewControllers(viewControllers, direction: .forward, animated: false, completion: {done in })

  self.pageViewController!.dataSource = self.modelController

  self.addChildViewController(self.pageViewController!)
  self.view.addSubview(self.pageViewController!.view)

  // Set the page view controller's bounds using an inset rect so that self's view is visible around the edges of the pages.
  var pageViewRect = self.view.bounds
  if UIDevice.current.userInterfaceIdiom == .pad {
      pageViewRect = pageViewRect.insetBy(dx: 40.0, dy: 40.0)
  }
  self.pageViewController!.view.frame = pageViewRect

  self.pageViewController!.didMove(toParentViewController: self)
}
```

data source in modelController

PAGE VIEW CONTROLLER TEMPLATE

ROOT VIEW CONTROLLER

```
override func viewDidLoad() {
  super.viewDidLoad()
  // Do any additional setup after loading the view, typically from a nib.
  // Configure the page view controller and add it as a child view controller.
  self.pageViewController = UIPageViewController(transitionStyle: .pageCurl, navigationOrientation: .horizontal, options: nil)
  self.pageViewController!.delegate = self

  let startingViewController: DataViewController = self.modelController.viewControllerAtIndex(0, storyboard: self.storyboard!)!
  let viewControllers = [startingViewController]
  self.pageViewController!.setViewControllers(viewControllers, direction: .forward, animated: false, completion: {done in })

  self.pageViewController!.dataSource = self.modelController

  self.addChildViewController(self.pageViewController!)
  self.view.addSubview(self.pageViewController!.view)

  // Set the page view controller's bounds using an inset rect so that self's view is visible around the
  var pageViewRect = self.view.bounds
  if UIDevice.current.userInterfaceIdiom == .pad {
      pageViewRect = pageViewRect.insetBy(dx: 40.0, dy: 40.0)
  }
  self.pageViewController!.view.frame = pageViewRect

  self.pageViewController!.didMove(toParentViewController: self)
}
```

view controller containment

PAGE VIEW CONTROLLER TEMPLATE

ROOT VIEW CONTROLLER

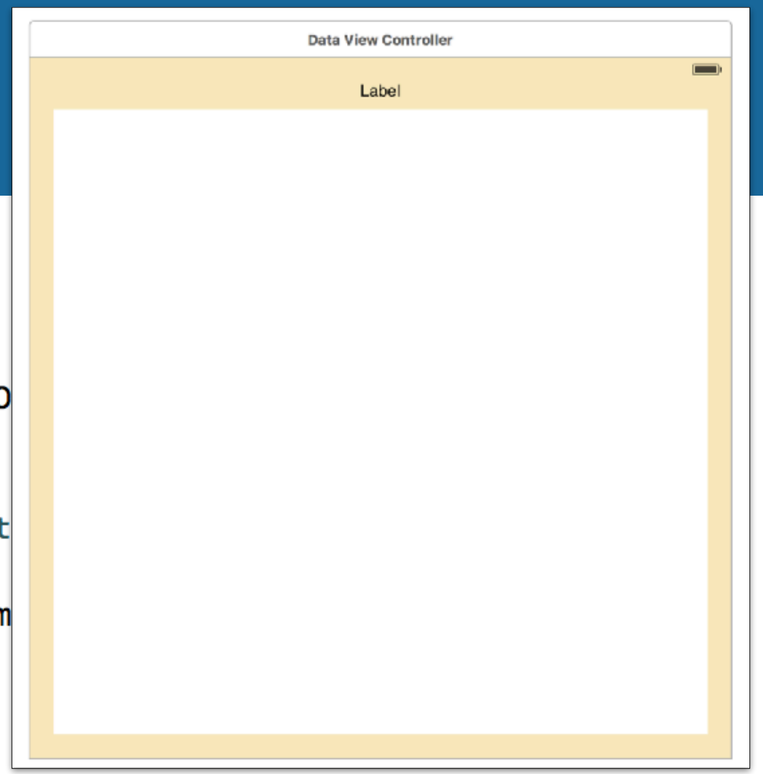
```
override func viewDidLoad() {
    super.viewDidLoad()
    // Do any additional setup after loading the view, typically from a nib.
    // Configure the page view controller and add it as a child view controlle
    self.pageViewController = UIPageViewController(transitionStyle: .pageCurl,                    : nil)
    self.pageViewController!.delegate = self

    let startingViewController: DataViewController = self.modelController.viewControllerAtIndex(0, storyboard: self.storyboard!)!
    let viewControllers = [startingViewController]
    self.pageViewController!.setViewControllers(viewControllers, direction: .forward, animated: false, completion: {done in })

    self.pageViewController!.dataSource = self.modelController

    self.addChildViewController(self.pageViewController!)
    self.view.addSubview(self.pageViewController!.view)

    // Set the page view controller's bounds using an inset rect so that self's view is visible around the
    var pageViewRect = self.view.bounds
    if UIDevice.current.userInterfaceIdiom == .pad {
        pageViewRect = pageViewRect.insetBy(dx: 40.0, dy: 40.0)
    }
    self.pageViewController!.view.frame = pageViewRect

    self.pageViewController!.didMove(toParentViewController: self)
}
```

This is just one way of implementing a page view controller. Provides "chrome" around the pages.

view controller containment

PAGE VIEW CONTROLLER TEMPLATE

ROOT VIEW CONTROLLER




```
override func viewDidLoad() {
  super.viewDidLoad()
  // Do any additional setup after loading the view, typically from a nib.
  // Configure the page view controller and add it as a child view controller.
  self.pageViewController = UIPageViewController(transitionStyle: .pageCurl, navigationO
  self.pageViewController!.delegate = self

  let startingViewController: DataViewControll                    rollerAt
  let viewControllers = [startingViewControlle
  self.pageViewController!.setViewControllers(                    rd, anim

  self.pageViewController!.dataSource = self.m

  self.addChildViewController(self.pageViewCo troller!)
  self.view.addSubview(self.pageViewControll r!.view)

  // Set the page view controller's bounds using an inset rect so that self's view is visible around the edges of the pages.
  var pageViewRect = self.view.bounds
  if UIDevice.current.userInterfaceIdiom == .pad {
      pageViewRect = pageViewRect.insetBy(dx: 40.0, dy: 40.0)
  }
  self.pageViewController!.view.frame = pageViewRect

  self.pageViewController!.didMove(toParentViewController: self)
}
```

create colored inset
(eye candy)

PAGE VIEW CONTROLLER TEMPLATE

ROOT VIEW CONTROLLER

```
func pageViewController(pageViewController: UIPageViewController, spineLocationForInterfaceOrientation orientation: UIInterfaceOrientation) -> UIPageViewControlle

if (orientation == .Portrait) || (orientation == .PortraitUpsideDown) || (UIDevice.currentDevice().userInterfaceIdiom == .Phone) {
    // In portrait orientation or on iPhone: Set the spine position to "min"
    // and the page view controller's view controllers array to contain just
    // one view controller. Setting the spine position to
    // 'UIPageViewControllerSpineLocationMid' in landscape orientation sets
    //the doubleSided property to true, so set it to false here.
    let currentViewController = self.pageViewController!.viewControllers![0]
    let viewControllers = [currentViewController]
    self.pageViewController!.setViewControllers(viewControllers, direction: .Forward, animated: true, completion: {done in })

    self.pageViewController!.doubleSided = false
    return .Min
}

// In landscape orientation: Set set the spine location to "mid" and the
// page view controller's view controllers array to contain two view
// controllers. If the current page is even, set it to contain the current
// and next view controllers; if it is odd, set the array to contain the
//  previous and current view controllers.
let currentViewController = self.pageViewController!.viewControllers![0] as! DataViewController
var viewControllers: [UIViewController]

let indexOfCurrentViewController = self.modelController.indexOfViewController(currentViewController)
if (indexOfCurrentViewController == 0) || (indexOfCurrentViewController % 2 == 0) {
    let nextViewController = self.modelController.pageViewController(self.pageViewController!, viewControllerAfterViewController: currentViewController)
    viewControllers = [currentViewController, nextViewController!]
} else {
    let previousViewController = self.modelController.pageViewController(self.pageViewController!, viewControllerBeforeViewController:
    viewControllers = [previousViewController!, currentViewController]
}
self.pageViewController!.setViewControllers(viewControllers, direction: .Forward, animated: true, completion: {done in })

return .Mid
}
```

show only one page (portrait)

custom methods to get view controller array

show two pages

PAGE VIEW CONTROLLER TEMPLATE

MODEL CONTROLLER

```
import UIKit

/*
 A controller object that manages a simple model — a collection of month names.

 The controller serves as the data source for the page view controller; it therefore implements pageViewController:viewControllerBeforeViewController: and pageViewCo
 It also implements a custom method, viewControllerAtIndex: which is useful in the implementation of the d                    he initial configuration of the

 There is no need to actually create view controllers for each page in advance -- indeed doing so incurs u                    he data model, these methods cr
demand.
 */


class ModelController: NSObject, UIPageViewControllerDataSource {

  var pageData: [String] = []


  override init() {...}

  func viewControllerAtIndex(index: Int, storyboard: UIStoryboard) -> DataViewController? {...}

  func indexOfViewController(viewController: DataViewController) -> Int {...}

  // MARK: - Page View Controller Data Source

  func pageViewController(pageViewController: UIPageViewController, viewControllerBeforeViewController viewController: UIViewController) -> UIViewController? {...}

  func pageViewController(pageViewController: UIPageViewController, viewControllerAfterViewController viewController: UIViewController) -> UIViewController? {...}

}
```

View controller for a data model at index

determine the index of a view controller

Data source methods

PAGE VIEW CONTROLLER TEMPLATE

MODEL CONTROLLER

```
class ModelController: NSObject, UIPageViewControllerDataSource {

  var pageData: [String] = []


  override init() {
      super.init()
    // Create the data model.
    let dateFormatter = NSDateFormatter()
    pageData = dateFormatter.monthSymbols
  }

  func viewControllerAtIndex(index: Int, storyboard: UIStoryboard) -> DataViewController? {
    // Return the data view controller for the given index.
    if (self.pageData.count == 0) || (index >= self.pageData.count) {
        return nil
    }

    // Create a new view controller and pass suitable data.
    let dataViewController = storyboard.instantiateViewControllerWithIdentifier("DataViewController") as! DataViewController
    dataViewController.dataObject = self.pageData[index]
    return dataViewController
  }

  func indexOfViewController(viewController: DataViewController) -> Int {
    // Return the index of the given data view controller.
    // For simplicity, this implementation uses a static array of model objects and the view controller stores the model object; you                                                    obje
    return pageData.indexOf(viewController.dataObject) ?? NSNotFound
  }

  // MARK: - Page View Controller Data Source

  func pageViewController(pageViewController: UIPageViewController, viewControllerBeforeViewController viewController: UIViewControlle                                                    {...}

  func pageViewController(pageViewController: UIPageViewController, viewControllerAfterViewController viewController: UIViewController) -> UIViewController? {...}
```

View controller for a data model at index

determine the index of a view controller

PAGE VIEW CONTROLLER TEMPLATE

MODEL CONTROLLER

View controller for a data model at index

```
// MARK: - Page View Controller Data Source

func pageViewController(_ pageViewController: UIPageViewController, viewControllerBefore viewController: UIViewController) ->
    var index = self.indexOfViewController(viewController as! DataViewController)
    if (index == 0) || (index == NSNotFound) {
        return nil
    }

    index -= 1
    return self.viewControllerAtIndex(index, storyboard: viewController.storyboard!)
}

func pageViewController(_ pageViewController: UIPageViewController, viewControllerAfter viewController: UIViewController) -> U
    var index = self.indexOfViewController(viewController as! DataViewController)
    if index == NSNotFound {
        return nil
    }

    index += 1
    if index == self.pageData.count {
        return nil
    }
    return self.viewControllerAtIndex(index, storyboard: viewController.storyboard!)
}

}
```

determine the index of a view controller

PAGE VIEW CONTROLLER TEMPLATE

MODEL CONTROLLER

```
demand.
 */


class ModelController: NSObject, UIPageViewControllerDataSource {

  var pageData: [String] = []


  override init() {...}

  func viewControllerAtIndex(index: Int, storyboard:                    troller? {

  func indexOfViewController(viewController: DataView

  // MARK: - Page View Controller Data Source

  func pageViewController(pageViewController: UIPageViewController, viewControllerBeforeViewController viewController: UIViewController) -> UIViewController? {
      var index = self.indexOfViewController(viewController as! DataViewController)
      if (index == 0) || (index == NSNotFound) {
          return nil
      }

      index--
      return self.viewControllerAtIndex(index, storyboard: viewController.storyboard!)
  }

  func pageViewController(pageViewController: UIPageViewController, viewControllerAfterViewController viewController: UIViewController) -> UIViewController? {
      var index = self.indexOfViewController(viewController as! DataViewController)
      if index == NSNotFound {
          return nil
      }

      index++
      if index == self.pageData.count {
          return nil
      }
```

EXTERNAL NAME

INTERNAL NAME

PAGE VIEW CONTROLLER TEMPLATE

DATA VIEW CONTROLLER

```
import UIKit

class DataViewController: UIViewController {

  @IBOutlet weak var dataLabel: UILabel!
  var dataObject: String = ""


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

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

  override func viewWillAppear(animated: Bool) {
    super.viewWillAppear(animated)
    self.dataLabel!.text = dataObject
  }


}
```

When is VC loaded?

Inset

LET'S BE LAZY

LAZY

```
var modelController: ModelController {
    // Return the model controller object, creating it if necessary.
    // In more complex implementations, the model controller may be
    // passed to the view controller.
    if _modelController == nil {
        _modelController = ModelController()
    }
    return _modelController!
}

var _modelController: ModelController? = nil
```

LAZY

- Lazy initialization is a technique for delaying the creation of an object until needed
- Swift `lazy` attribute
- Examples of best practices
  - Initial value for a property is not known until after the object is initialized
  - Initial value for a property is computationally intensive

```
var modelController: ModelController {
  // Return the model controller object, creating it if necessary.
  // In more complex implementations, the model controller may be
  // passed to the view controller.
  if _modelController == nil {
      _modelController = ModelController()
  }
  return _modelController!
}

var _modelController: ModelController? = nil
```

LAZY

Objective-C style pattern of lazy initialization

```
var modelController: ModelController {
    // Return the model controller object, creating it if necessary.
    // In more complex implementations, the model controller may be
    // passed to the view controller.
    if _modelController == nil {
        _modelController = ModelController()
    }
    return _modelController!
}

var _modelController: ModelController? = nil
```

LAZY

```
import UIKit

class RootViewController: UIViewController, UIPageViewControllerDelegate {

  var pageViewController: UIPageViewController?


  override func viewDidLoad() {•••}

  override func didReceiveMemoryWarning() {•••}

  lazy var modelController: ModelController = ModelController()


  //
  // MARK: - UIPageViewController delegate methods
  //

  func pageViewController(pageViewController: UIPageViewController, spineLocationForInterfaceOrientation


}
```

LAZY

Objective-C style pattern of lazy initialization

PagesAndPages 〉 Pa AndPages 〉 RootViewController.swift 〉 RootViewController

```swift
//
//  RootViewController.swift
//  PagesAndPages
//
//  Created by T. Andrew Binkowski on 3/26/16.
//  Copyright © 2016 The University of Chicago, Department of Computer Science. All rights reserve
//

import UIKit

class RootViewController: UIViewController, UIPageViewControllerDelegate {

  var pageViewController: UIPageViewController?


  override func viewDidLoad() {...}

  override func didReceiveMemoryWarning() {...}

  var modelController: ModelController {
    // Return the model controller object, creating it if necessary.
    // In more complex implementations, the model controller may be passed to the view controller.
    if _modelController == nil {
        _modelController = ModelController()
    }
    return _modelController!
  }

  var _modelController: ModelController? = nil


  //
  // MARK: - UIPageViewController delegate methods
  //

  func pageViewController(pageViewController: UIPageViewController, spineLocationForInterfaceOrien


}
```

```swift
//
//  Created by T. Andrew Binkowski on 3/26/16.
//  Copyright © 2016 The University of Chicago, Department of Com
//

import UIKit

class RootViewController: UIViewController, UIPageViewControllerD

  var pageViewController: UIPageViewController?


  override func viewDidLoad() {...}

  override func didReceiveMemoryWarning() {...}

  lazy var modelController: ModelController = ModelController()


  //
  // MARK: - UIPageViewController delegate methods
  //

  func pageViewController(pageViewController: UIPageViewControlle


}
```

LAZY

Main.storyboard

PagesAndPages 〉 PagesAndPages 〉 RootViewController.swift 〉 RootViewController

```
//
//  RootViewController.swift
//  PagesAndPages
//
//  Created by T. Andrew Binkowski on 3/26/16
//  Copyright © 2016 T                                    All rights reserve
//

import UIKit

class RootViewControll

  var pageViewControll


  override func viewDid

  override func didReceiveMemoryWarning() {...}

  var modelController: ModelController {
    // Return the model controller object, creating it if necessary.
    // In more complex implementations, the model controller may be passed to the view controller.
    if _modelController == nil {
        _modelController = ModelController()
    }
    return _modelController!
  }

  var _modelController: ModelController? = nil


  //
  // MARK: - UIPageViewController delegate methods
  //

  func pageViewController(pageViewController: UIPageViewController, spineLocationForInterfaceOrien

}
```

COMPUTED PROPERTY
– ALWAYS CALLED

```
//
//  Created by T. Andrew Binkowski on 3/26/16.
//  Copyright © 2016 The University of Chicago, Department of Com
//

import UIKit

class RootViewController: UIViewController, UIPageViewControllerD

  var pageViewController: UIPageViewController?


  override func viewDidLoad() {...}

  override func didReceiveMemoryWarning() {...}

  lazy var modelController: ModelController = ModelController()


  //
  // MARK: - UIPageViewController delegate methods
  //

  func pageViewCo                                          PageViewControlle

}
```

LAZY IS ONLY EVALUATED ONCE

THE UNIVERSITY OF CHICAGO

ADVANCED iOS APPLICATION DEVELOPMENT

MPCS 51032 • SPRING 2020 • SESSION 1A