



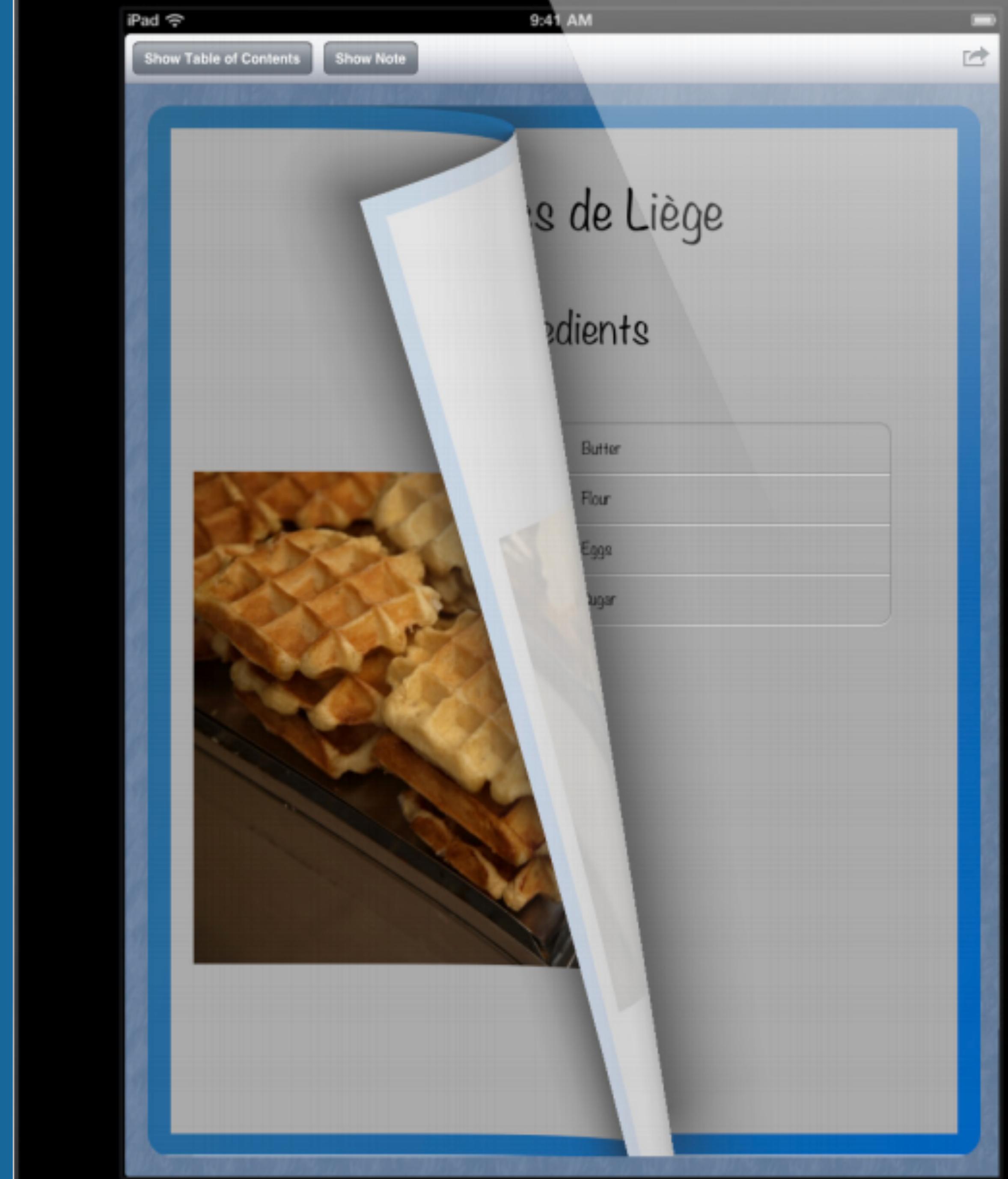
ADVANCED iOS APPLICATION DEVELOPMENT

MPCS 51032 • SPRING 2020 • SESSION 1A

PAGE VIEW CONTROLLER

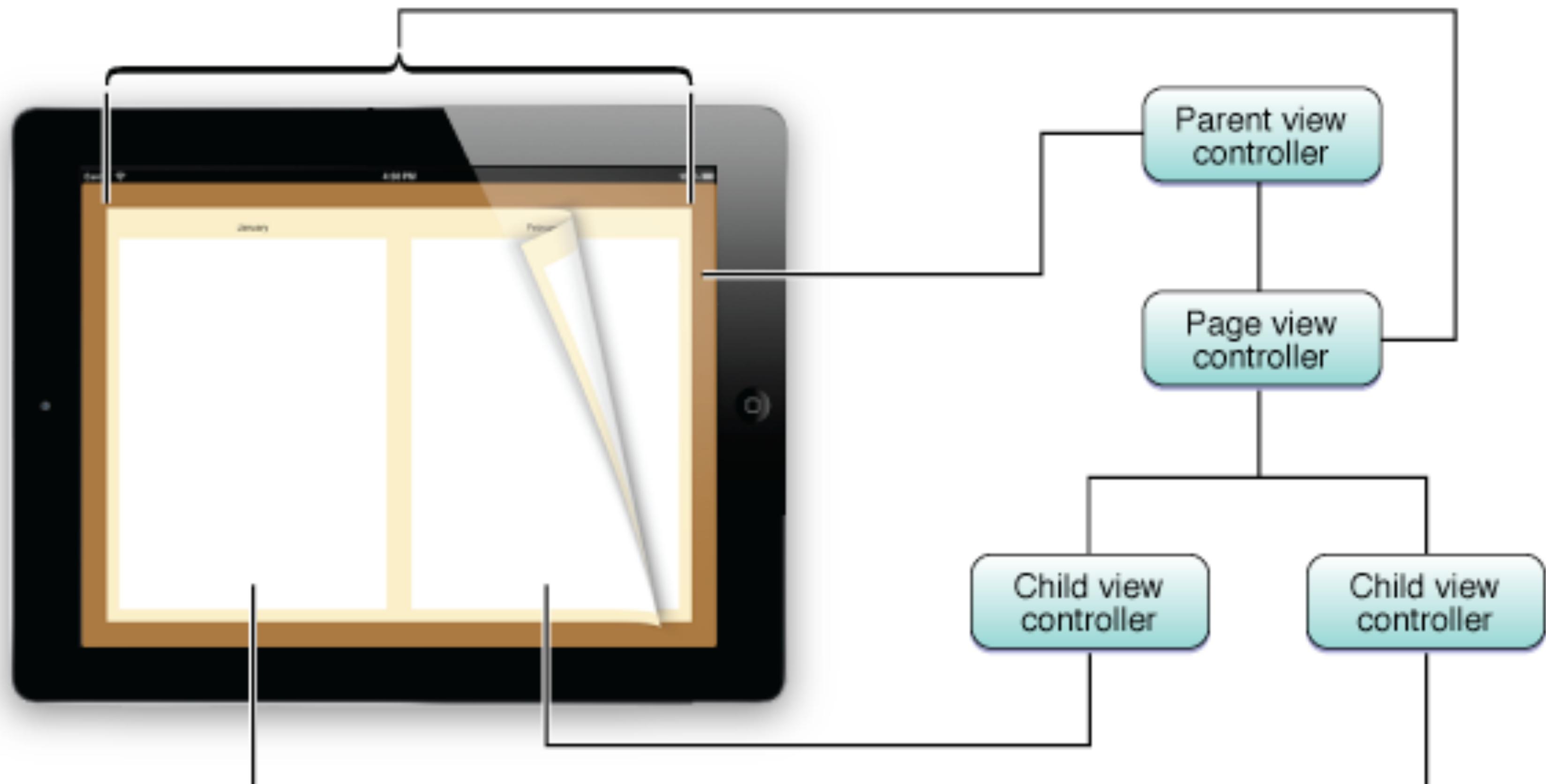
PAGE VIEW CONTROLLER

- 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



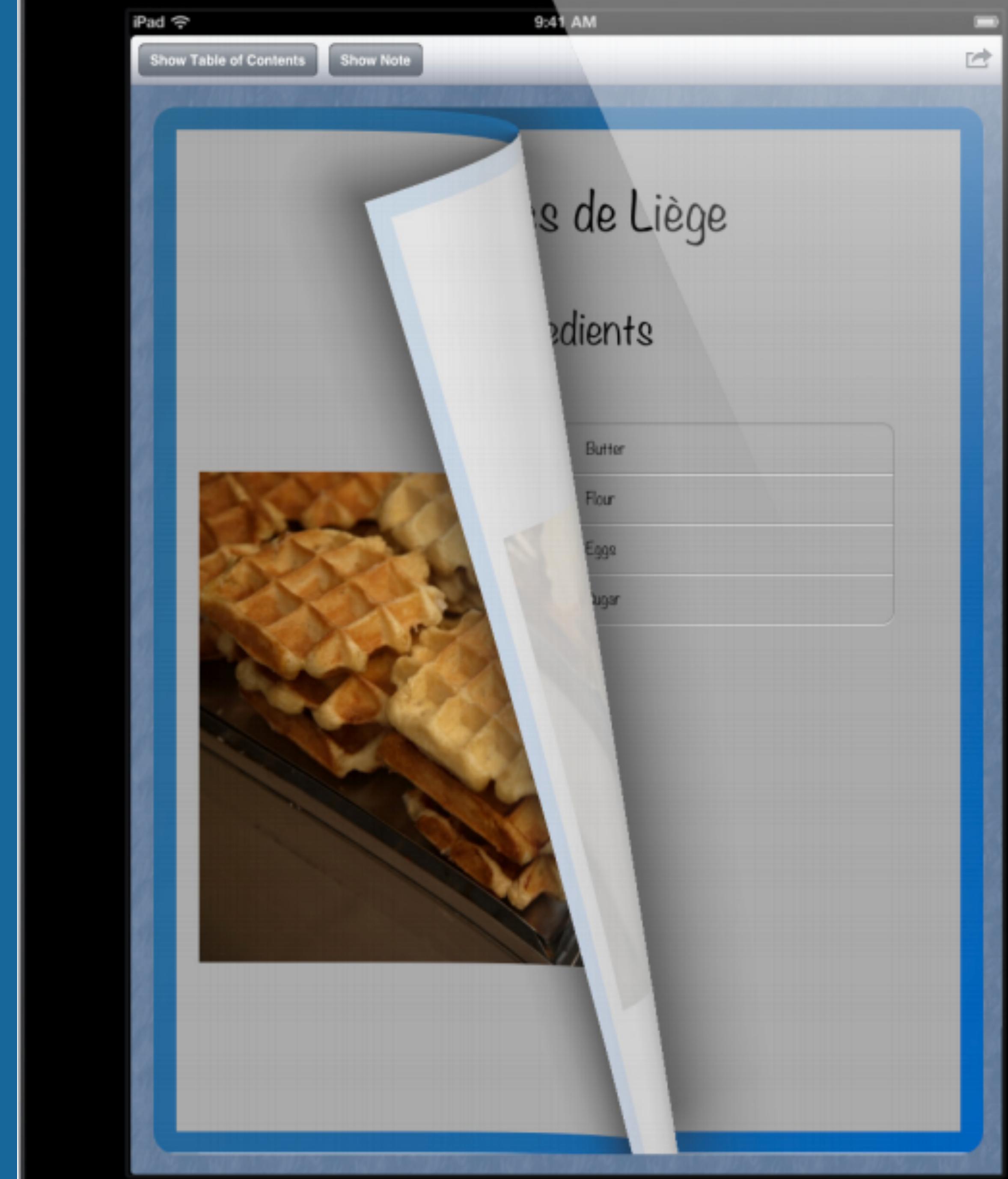
PAGE VIEW CONTROLLER

- 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

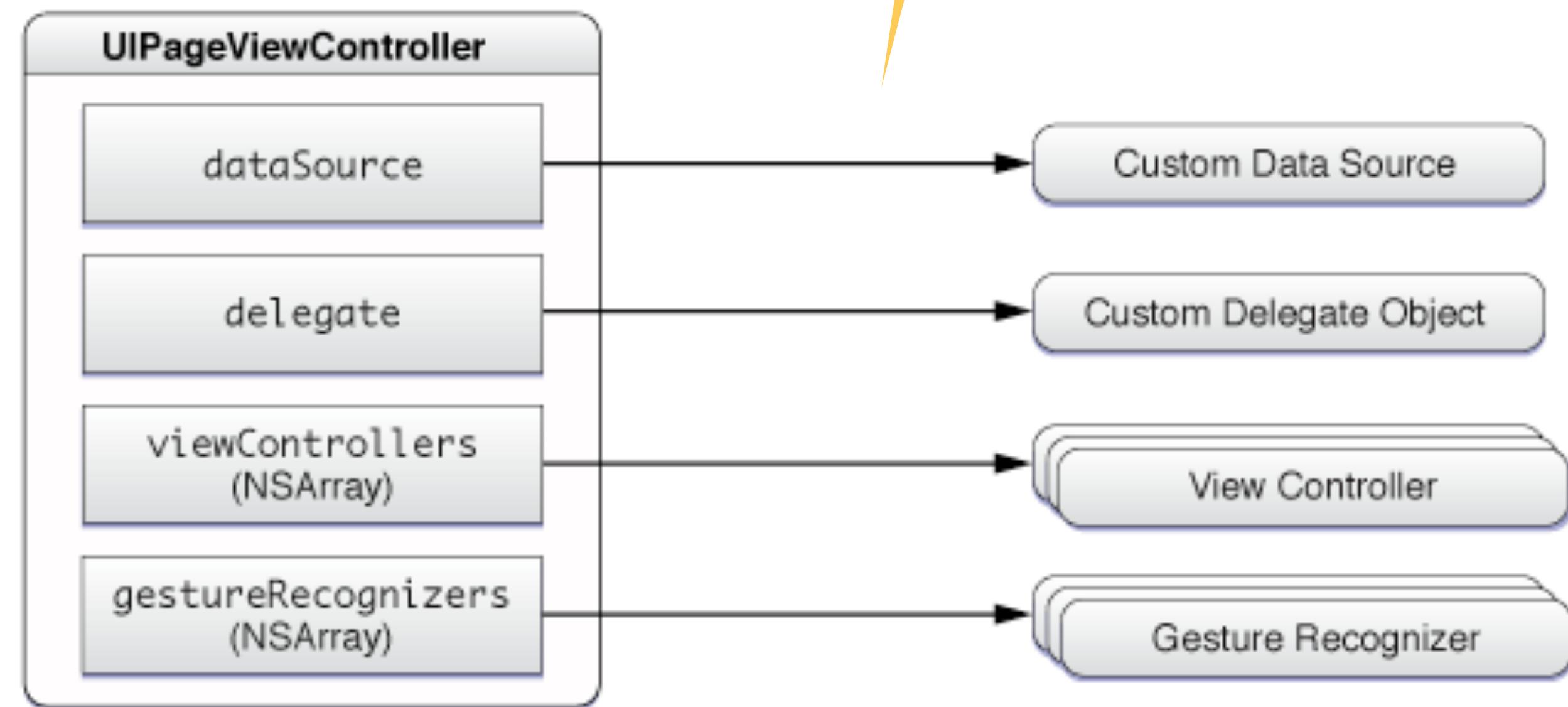
- Provide the content view controllers
 - One at a time
 - Two at a time
 - As-needed using a data source
- When providing content view controllers one at a time use the `setViewControllers:direction:animated:completion:` method to set the current content view controllers.
- To support gesture-based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

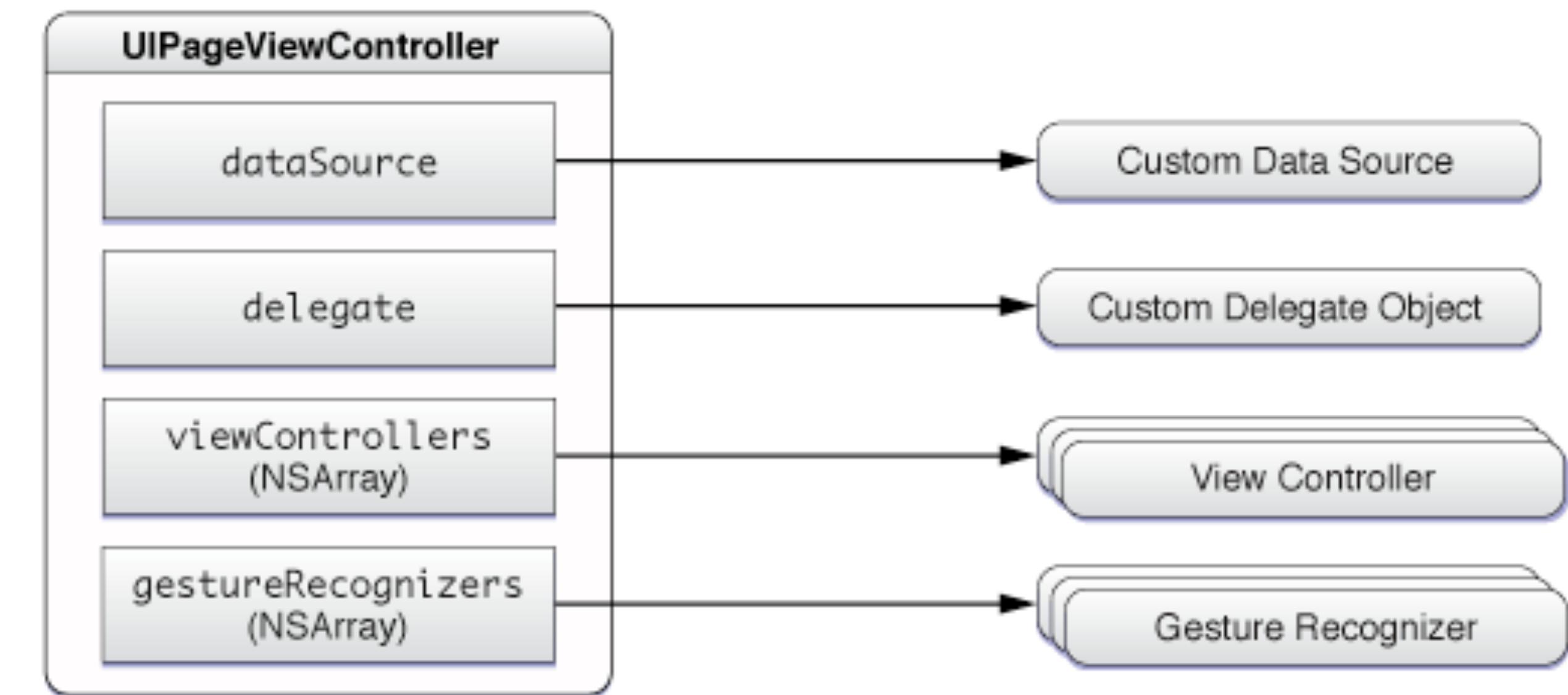


Just like a table

PAGE VIEW CONTROLLER

UIPAGEVIEWCONTROLLER API

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

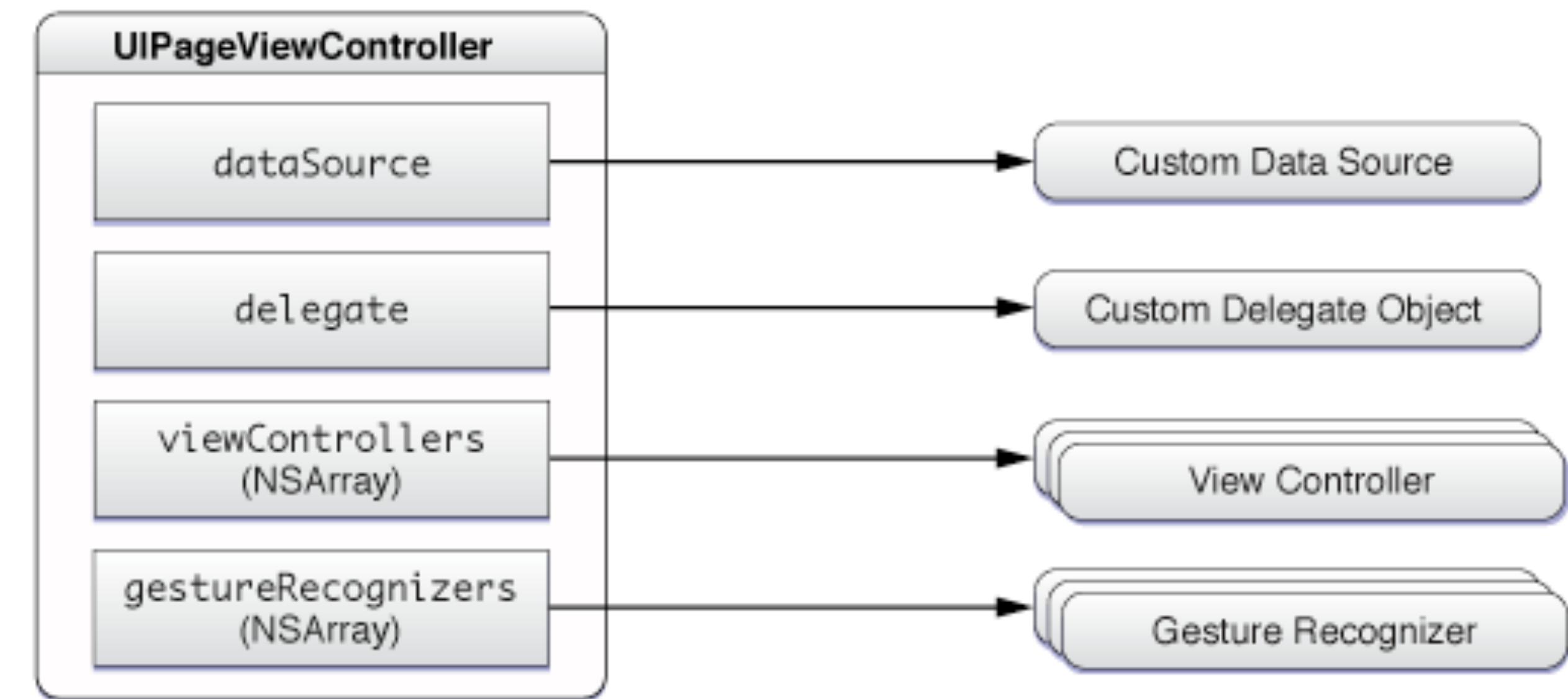


Reusing view controllers
similar to table cells

PAGE VIEW CONTROLLER

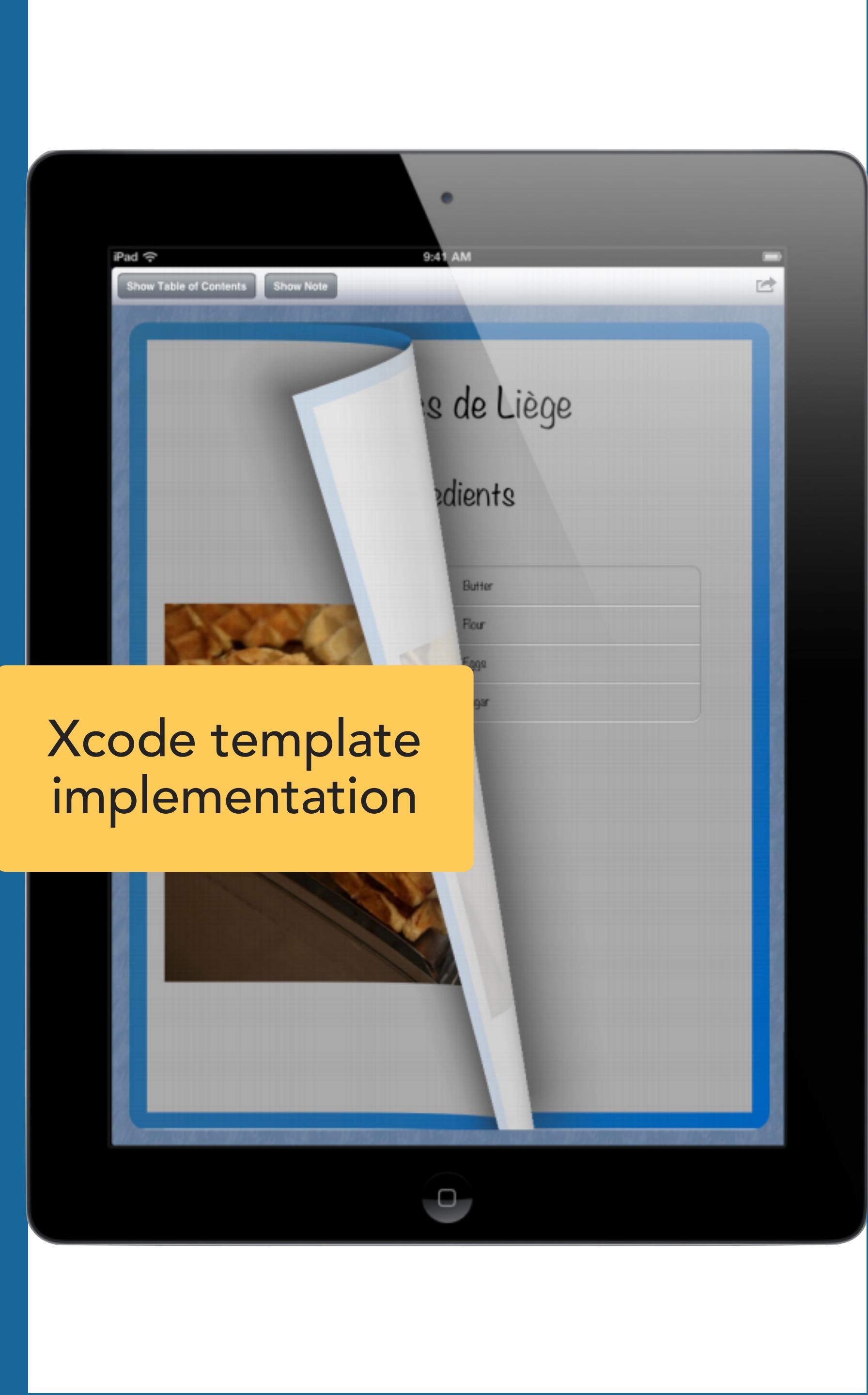
UIPAGEVIEWCONTROLLER API

- 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

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

array of view controllers

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

UIPAGEVIEWCONTROLLER

INITIALIZATION

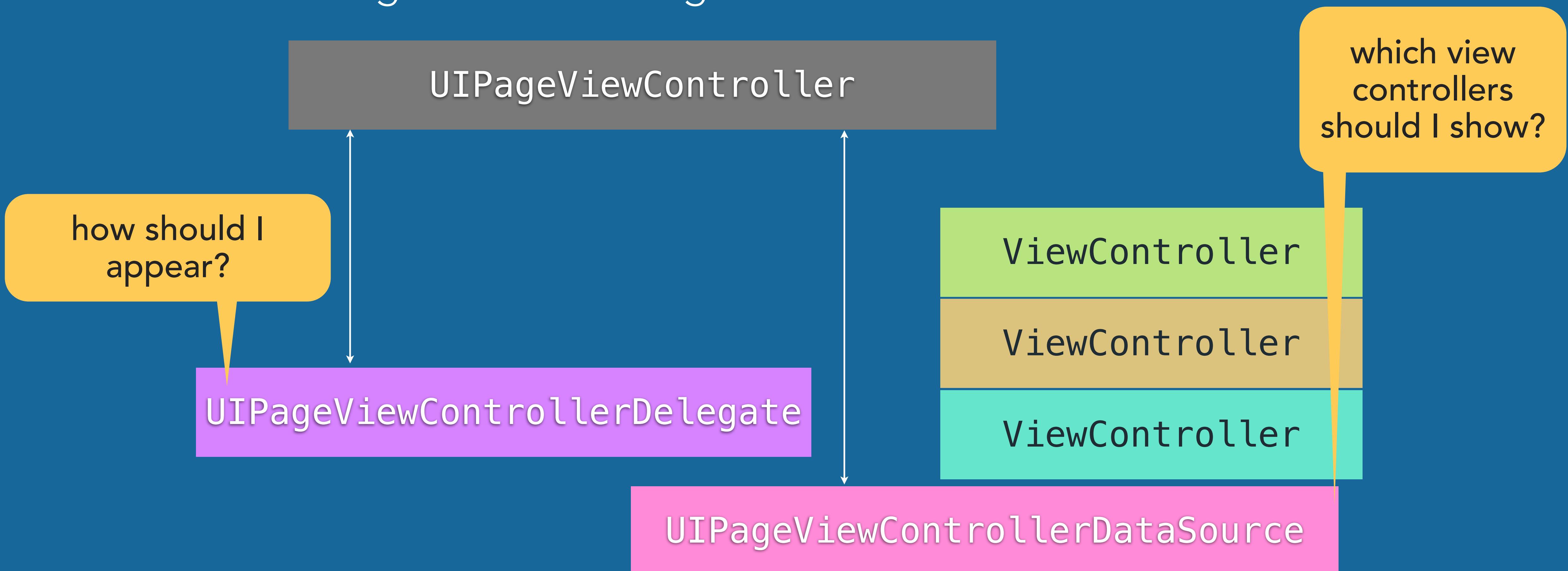
```
let viewControllers = [currentViewController]
self.pageViewController!.setViewControllers(viewControllers, direction: .Forward, animated: true, completion: {done in })
```

- 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

iOS

Application

Framework & Library

Other

OS X

Application

Framework & Library

System Plug-in

Other

 Master-Detail Application

 Page-Based Application

 1 Single View Application

 Tabbed Application

 Game

Page-Based Application

This template provides a starting point for a page-based application that uses a page view controller to manage multiple pages of content.

Page-Based

Cancel Previous Next

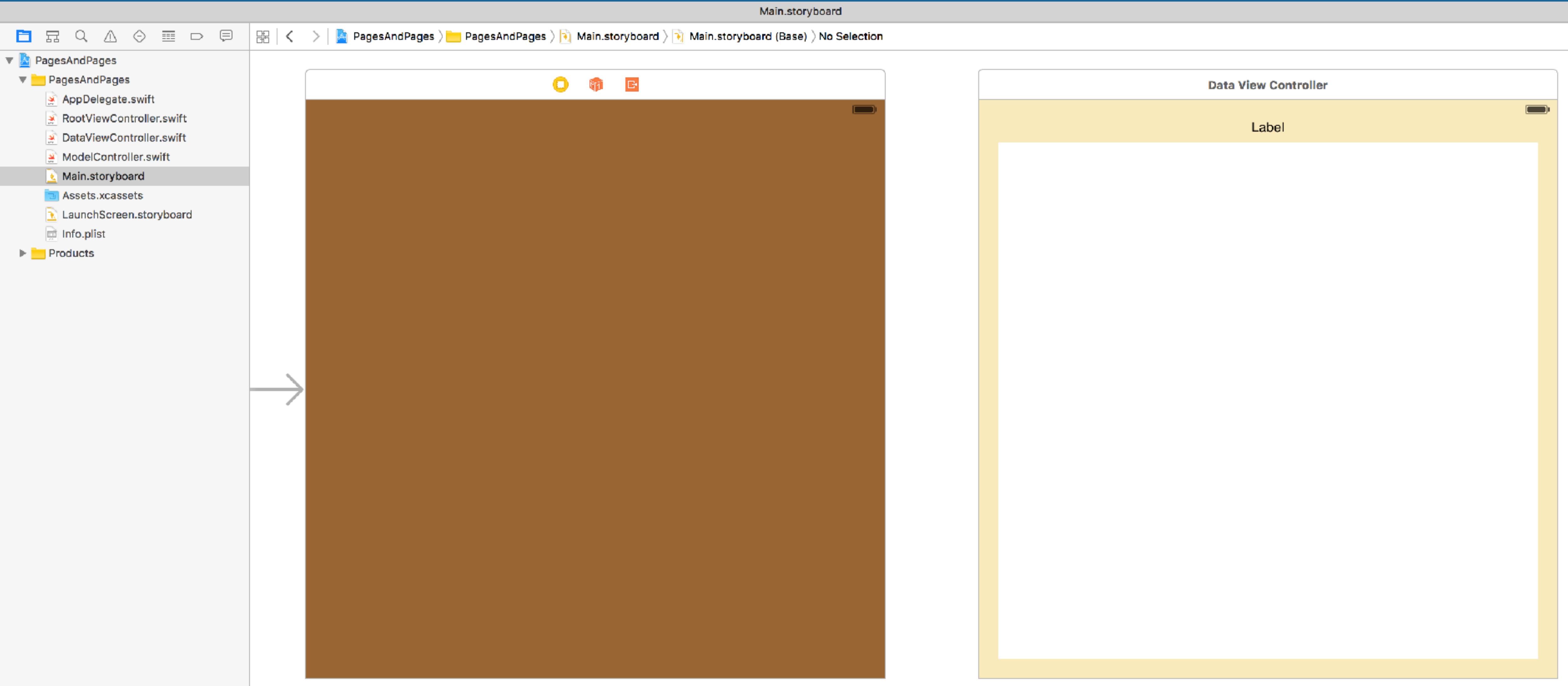
ction



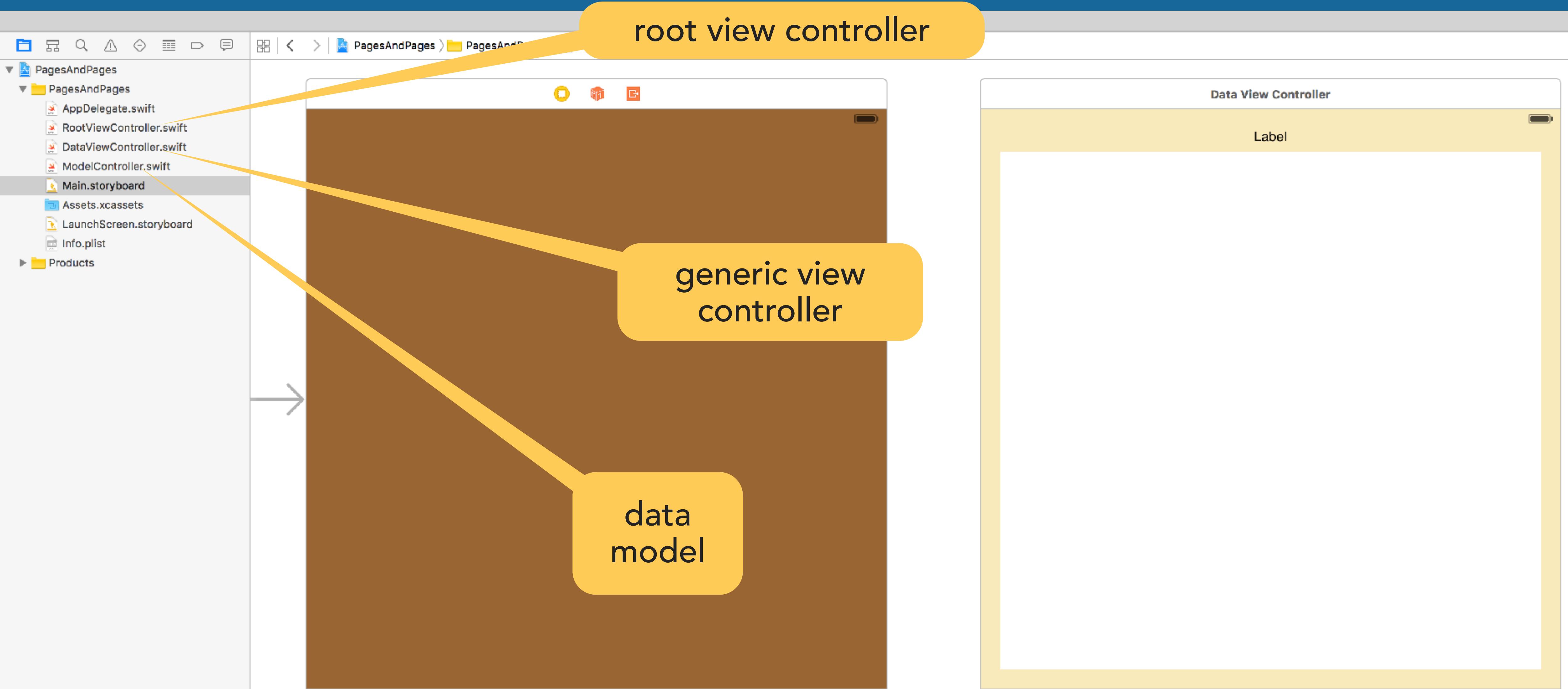
f - Define a block as

as Variable - Save
able to allow reuse or
argument.

PAGE VIEW CONTROLLER TEMPLATE

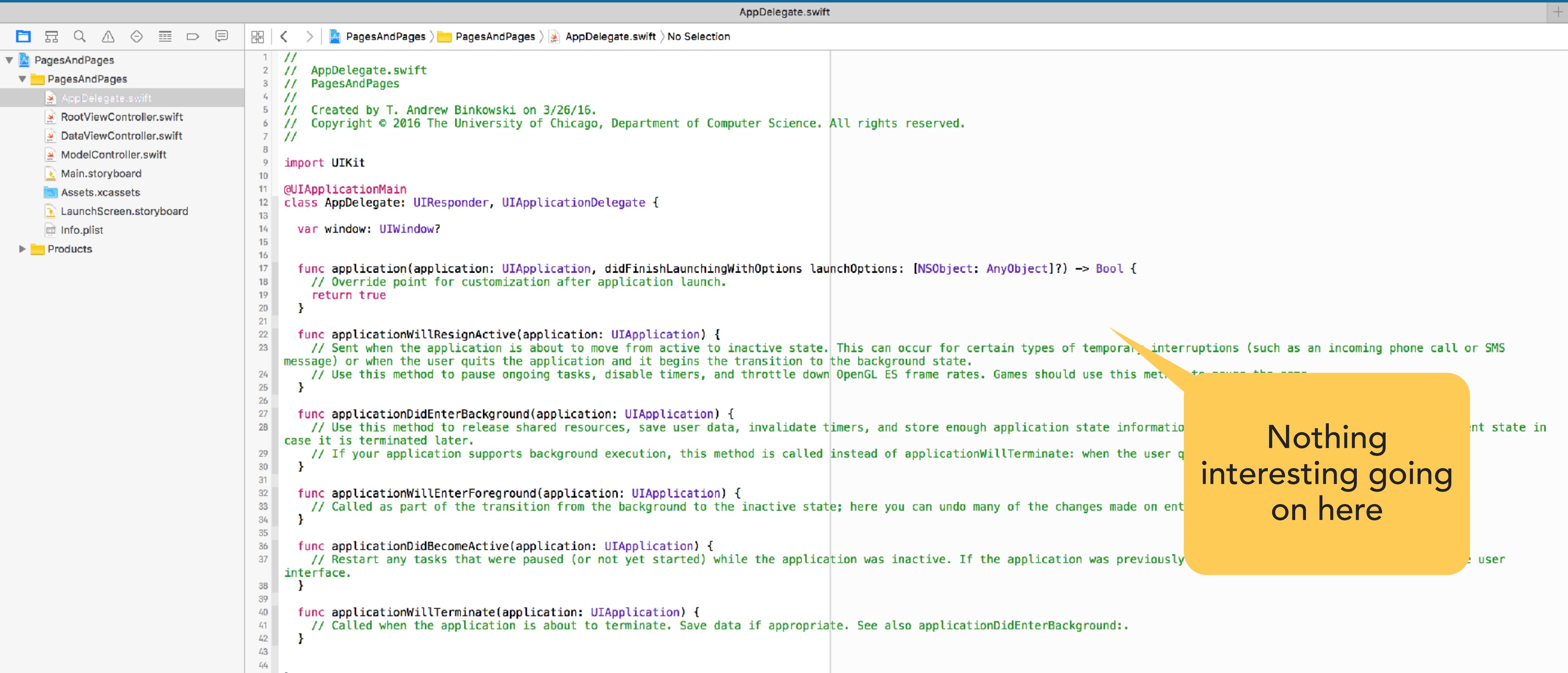


PAGE VIEW CONTROLLER TEMPLATE



PAGE VIEW CONTROLLER TEMPLATE

ROOT VIEW CONTROLLER



The screenshot shows the Xcode interface with the file `AppDelegate.swift` open. The left sidebar shows the project structure with files like `RootViewController.swift`, `DataViewController.swift`, and `ModelController.swift`. The main editor area contains the following Swift code:

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

import UIKit

@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate {

    var window: UIWindow?

    func application(application: UIApplication, didFinishLaunchingWithOptions launchOptions: [NSObject: AnyObject]?) -> Bool {
        // Override point for customization after application launch.
        return true
    }

    func applicationWillResignActive(application: UIApplication) {
        // Sent when the application is about to move from active to inactive state. This can occur for certain types of temporary interruptions (such as an incoming phone call or SMS message) or when the user quits the application and it begins the transition to the background state.
        // Use this method to pause ongoing tasks, disable timers, and throttle down OpenGL ES frame rates. Games should use this method to pause the game.
    }

    func applicationDidEnterBackground(application: UIApplication) {
        // Use this method to release shared resources, save user data, invalidate timers, and store enough application state information
        // case it is terminated later.
        // If your application supports background execution, this method is called instead of applicationWillTerminate: when the user quits.
    }

    func applicationWillEnterForeground(application: UIApplication) {
        // Called as part of the transition from the background to the inactive state; here you can undo many of the changes made on entering the background.
    }

    func applicationDidBecomeActive(application: UIApplication) {
        // Restart any tasks that were paused (or not yet started) while the application was inactive. If the application was previously
        // suspended, the system calls this method before the first line of the current run loop executes.
    }

    func applicationWillTerminate(application: UIApplication) {
        // Called when the application is about to terminate. Save data if appropriate. See also applicationDidEnterBackground:.
    }
}
```

A yellow callout bubble with the text "Nothing interesting going on here" points to the bottom of the code.

PAGE VIEW CONTROLLER TEMPLATE

ROOT VIEW CONTROLLER

The screenshot shows the Xcode interface with the file `RootViewController.swift` open. The code implements a `RootViewController` class that conforms to `UIViewController` and `UIPageViewControllerDelegate`. It initializes a `pageViewController` and sets its delegate to `self`. The `viewDidLoad` method configures the page view controller's data source and adds it as a child. A callout bubble points from the text "Useful for book style app" to the `UIPageViewControllerDelegate` documentation.

```
// RootViewController.swift
// PagesAndPages
// Created by T. Andrew Binkowski on 3/26/16.
// Copyright © 2016 The University of Chicago, Department of
// import UIKit
class RootViewController: UIViewController, UIPageViewControllerDelegate {
    var pageViewController: UIPageViewController?
    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.
        self.pageViewController = UIPageViewController(transitionStyle: .scroll)
        self.pageViewController!.delegate = self
        let startingViewController: DataViewController = self.modelController!.createViewController()
        let viewControllers = [startingViewController]
        self.pageViewController!.setViewControllers(viewControllers, direction: .forward, animated: true, completion: nil)
        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 rectangle.
        var pageViewRect = self.view.bounds
        if UIDevice.currentDevice().userInterfaceIdiom == .Pad {
            pageViewRect = CGRectInset(pageViewRect, 40.0, 40.0)
        }
        self.pageViewController!.view.frame = pageViewRect
        self.pageViewController!.didMove(toParentViewController: self)
    }
    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be recreated.
    }
}
```

UIPageViewControllerDelegate

Inherits from: None
Conforms to: NSObject
Framework: UIKit in iOS 5.0 and later. [More related](#)

The delegate of a page view controller must adopt the `UIPageViewControllerDelegate` protocol. This protocol defines methods that allow the delegate to receive a notification when the device orientation changes and when the user navigates to a new page. For page-curl style transitions, the delegate can provide a different spine location in response to a change in the interface orientation.

Responding to Page View Controller Events

- `pageViewController:willTransitionToViewControllers:`

Called before a gesture-driven transition begins.

Declaration

SWIFT

```
optional func pageViewController(_ pageViewController: UIPageViewController, willTransitionToViewControllers pendingViewControllers: [UIViewController])
```

OBJECTIVE-C

```
-(void)pageViewController:(UIPageViewController *)pageViewController willTransitionToViewControllers:(NSArray<UIViewController *> *)pendingViewControllers
```

[Feedback](#)

Page view controller Delegate

Useful for book style app

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)
}
```

Initialize the page 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)
}
```

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

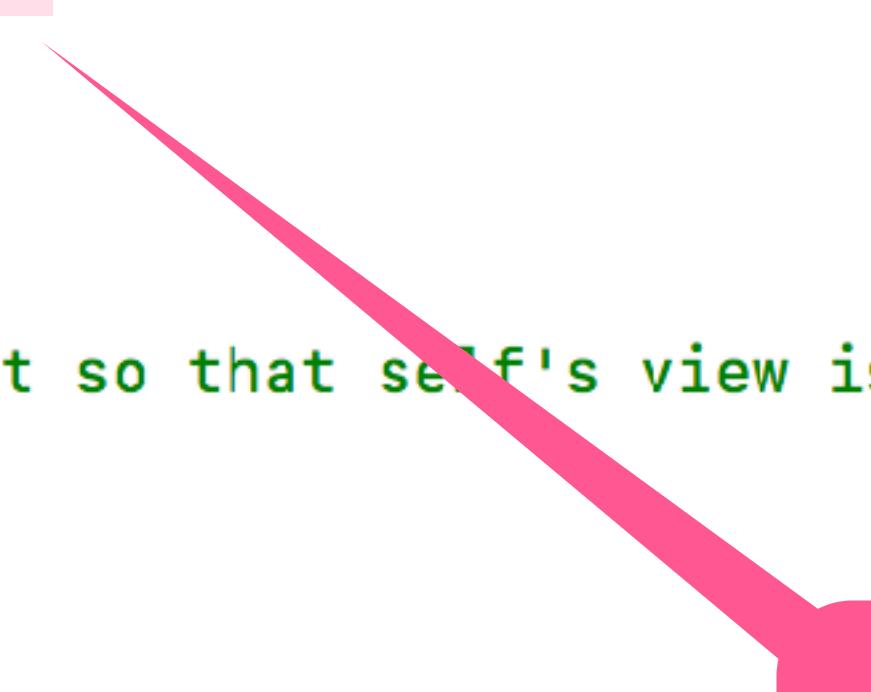
    let startingViewController: DataViewController = self.modelController.viewControllerAtIndex(0, storyboard: self.storyboard)!)
    let viewControllers = [startingViewController]
    self.pageViewController!.setViewControllers(viewControllers, direction: .forward, animated: false, completion: {done in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

    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 page view controller
    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 controller
    self.pageViewController = UIPageViewController(transitionStyle: .pageCurl,
                                                initialPage: 0,
                                                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 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)
}
```

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, navigationOrientation: .horizontal)
    self.pageViewController!.delegate = self

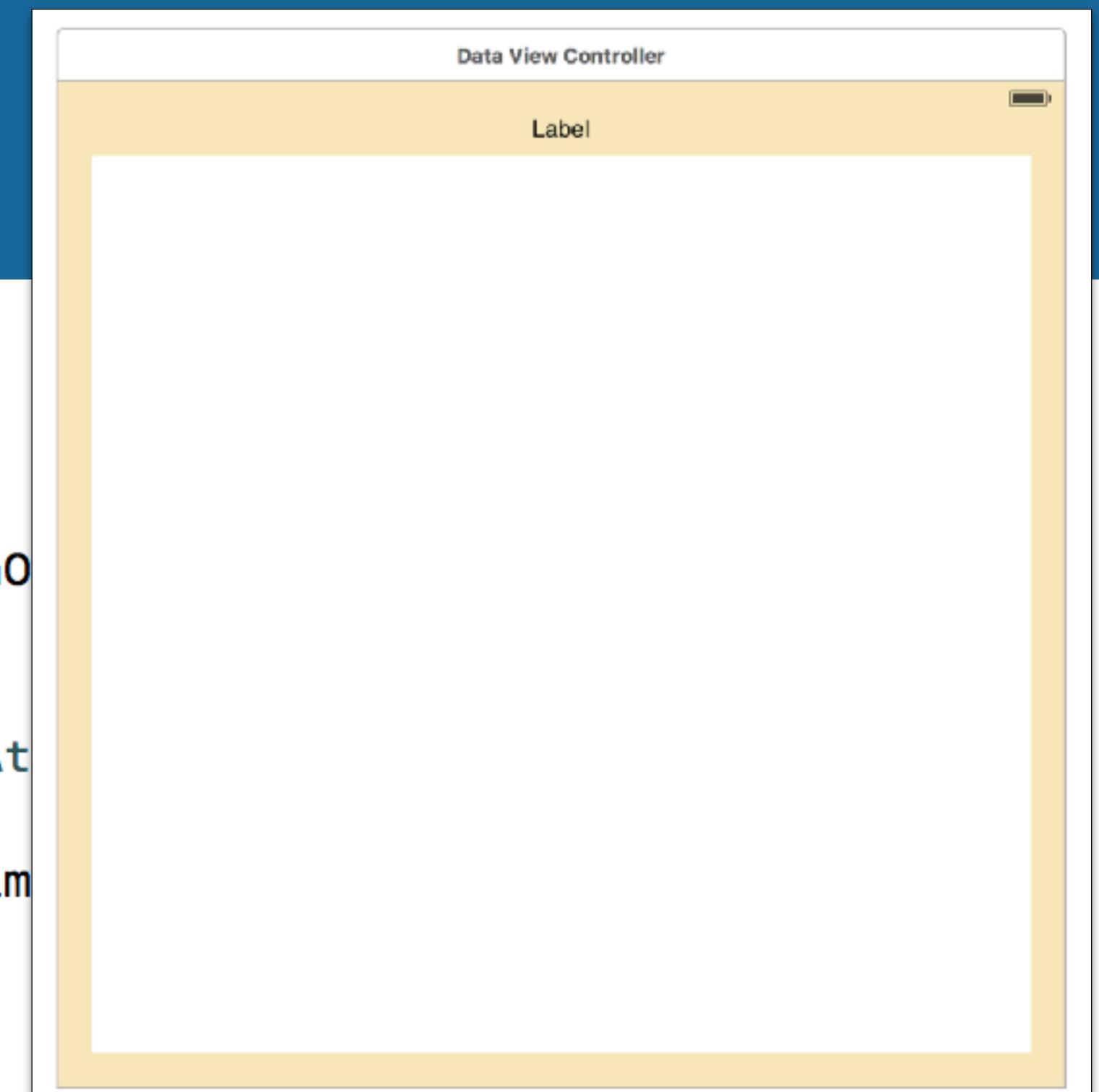
    let startingViewController: DataViewController = DataViewController()
    let viewControllers = [startingViewController]
    self.pageViewController!.setViewControllers(viewControllers, direction: .forward, animated: true, completion: nil)

    self.pageViewController!.dataSource = self

    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)
}
```



create colored inset
(eye candy)

PAGE VIEW CONTROLLER TEMPLATE

ROOT VIEW CONTROLLER

show only one page (portrait)

```
func pageViewController(pageViewController: UIPageViewController, spineLocationForInterfaceOrientation orientation: UIInterfaceOrientation) -> UIPageViewController {  
    if (orientation == .Portrait) || (orientation == .PortraitUpsideDown) || (UIDevice.currentDevice().userInterfaceOrientation == .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 viewController = self.pageViewController!.viewControllers![0]  
        let viewControllers = [viewController]  
        self.pageViewController!.setViewControllers(viewControllers, direction: .Forward, animated: true, completion: {done in })  
  
        self.pageViewController!.doubleSided = false  
        return .Min  
    }  
  
    // In landscape orientation: 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 viewController = 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
```

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 pageViewController:viewControllerAfterViewController:.

It also implements a custom method, viewControllerAtIndex:, which is useful in the implementation of the delegate methods.

There is no need to actually create view controllers for each page in advance -- indeed doing so incurs unnecessary overhead. Instead, the controller maintains a collection of view controllers in memory, and uses them on 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 = DateFormatter()  
        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  
        return pageData.indexOf(viewController.dataObject) ?? NSNotFound  
    }  
  
    // 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

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) -> 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) -> UIViewController? {
    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() {  
        super.init()  
    }  
  
    func viewControllerAtIndex(index: Int, storyboard: UIStoryboard) -> UIViewController? {  
        let viewController = storyboard.instantiateViewControllerWithIdentifier("DataViewController")  
        return viewController  
    }  
  
    func indexOfViewController(viewController: DataViewController) -> Int {  
        let index = self.pageData.indexOf(viewController.pageData)  
        if index == NSNotFound {  
            return -1  
        }  
        return index  
    }  
  
    // 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  
        }  
        return self.viewControllerAtIndex(index, storyboard: viewController.storyboard!)  
    }  
}
```



The diagram illustrates the mapping between external names and internal names. Two yellow callout boxes point to the storyboard parameter in the pageViewControllers methods. The left box is labeled "EXTERNAL NAME" and the right box is labeled "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.
    }

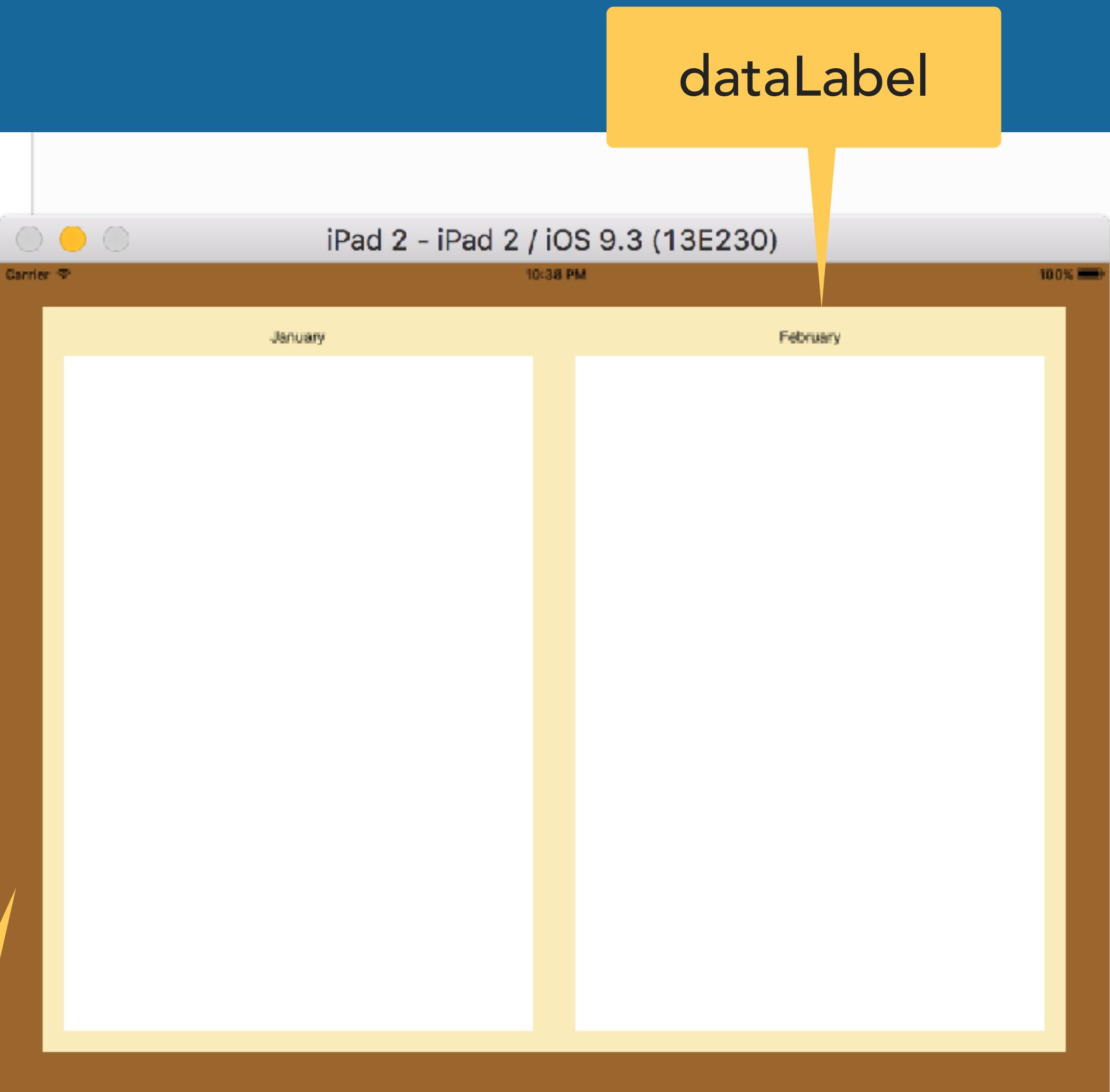
    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be recreated.
    }

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

When is VC loaded?

Inset

dataLabel



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

```
1 // RootViewController.swift
2 // PagesAndPages
3 // Created by T. Andrew Binkowski on 3/26/16.
4 // Copyright © 2016 The University of Chicago, Department of Computer Science. All rights reserved.
5
6 import UIKit
7
8 class RootViewController: UIViewController, UIPageViewControllerDelegate {
9     var pageViewController: UIPageViewController?
10
11     override func viewDidLoad() { ... }
12
13     override func didReceiveMemoryWarning() { ... }
14
15
16     var modelController: ModelController {
17         // Return the model controller object, creating it if necessary.
18         // In more complex implementations, the model controller may be passed to the view controller.
19         if _modelController == nil {
20             _modelController = ModelController()
21         }
22         return _modelController!
23     }
24
25     var _modelController: ModelController? = nil
26
27
28     // MARK: - UIPageViewController delegate methods
29
30
31     func pageViewController(pageViewController: UIPageViewController, spineLocationForInterfaceOrientation: UIInterfaceOrientation) -> CGFloat {
32         return 0.5
33     }
34
35
36     // MARK: - UIScrollViewDelegate methods
37
38
39     func scrollViewDidScroll(scrollView: UIScrollView) {
40         let offset = scrollView.contentOffset.y
41         let page = Int(offset / 100)
42         self.pageViewController?.setViewControllers([self.modelController.pageAtIndex(page)], direction: .Right, animated: true)
43     }
44
45
46     func scrollViewWillEndDragging(scrollView: UIScrollView, withVelocity velocity: CGPoint, targetContentOffset: UnsafePointer<CGPoint>) {
47         let offset = scrollView.contentOffset.y
48         let page = Int(offset / 100)
49         self.pageViewController?.setViewControllers([self.modelController.pageAtIndex(page)], direction: .Right, animated: true)
50     }
51
52
53     func scrollViewDidEndDecelerating(scrollView: UIScrollView) {
54         self.scrollViewWillEndDragging(scrollView, withVelocity: CGPointZero, targetContentOffset: UnsafePointer<CGPoint>(nil))
55     }
56
57
58
59     // MARK: - UIScrollViewDelegate methods
60
61
62     func scrollViewDidEndDecelerating(scrollView: UIScrollView) {
63         self.scrollViewWillEndDragging(scrollView, withVelocity: CGPointZero, targetContentOffset: UnsafePointer<CGPoint>(nil))
64     }
65
66
67
68     // MARK: - UIScrollViewDelegate methods
69
70
71     func scrollViewDidEndDecelerating(scrollView: UIScrollView) {
72         self.scrollViewWillEndDragging(scrollView, withVelocity: CGPointZero, targetContentOffset: UnsafePointer<CGPoint>(nil))
73     }
74
75
76
77     // MARK: - UIScrollViewDelegate methods
78
79
80     func scrollViewDidEndDecelerating(scrollView: UIScrollView) {
81         self.scrollViewWillEndDragging(scrollView, withVelocity: CGPointZero, targetContentOffset: UnsafePointer<CGPoint>(nil))
82     }
83
84
85
86     // MARK: - UIScrollViewDelegate methods
87
88
89     func scrollViewDidEndDecelerating(scrollView: UIScrollView) {
90         self.scrollViewWillEndDragging(scrollView, withVelocity: CGPointZero, targetContentOffset: UnsafePointer<CGPoint>(nil))
91     }
92
93
94
95     // MARK: - UIScrollViewDelegate methods
96
97
98     func scrollViewDidEndDecelerating(scrollView: UIScrollView) {
99         self.scrollViewWillEndDragging(scrollView, withVelocity: CGPointZero, targetContentOffset: UnsafePointer<CGPoint>(nil))
100    }
101
102
103
```

```
//
// Created by T. Andrew Binkowski on 3/26/16.
// Copyright © 2016 The University of Chicago, Department of Computer Science. All rights reserved.
//
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: UIInterfaceOrientation) -> CGFloat {
        return 0.5
    }
}
```

LAZY

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, UIPageViewControllerData...  
  
var pageViewController: UIPageViewController?  
  
override func viewDidLoad() { ... }  
  
override func didReceiveMemoryWarning() { ... }  
  
lazy var modelController: ModelController = ModelController()  
  
//  
// MARK: - UIPageViewController delegate methods  
  
func pageViewCo...  
}  
  
LAZY IS ONLY EVALUATED ONCE
```



ADVANCED iOS APPLICATION DEVELOPMENT

MPCS 51032 • SPRING 2020 • SESSION 1A