Implementing UIViewController Containment

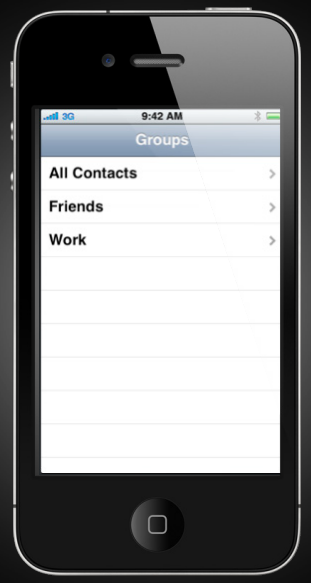
Advanced view controller topics

Session 102

Bruce D. Nilo and Matt Gamble
Software Engineers—iOS Applications and Frameworks

In the beginning…

There was a UIViewController



And we are truly are good.

Goals for This Talk

- Understand the difference between content and container view controllers
- Know how and when to implement custom view controller containers
- Introduce the UIPageViewController new on iOS 5
- Share tips and some new API

View Controllers

Basics

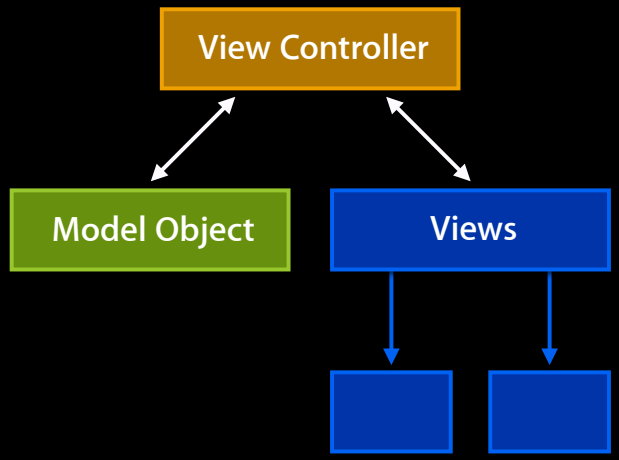
Why Should I Use View Controllers?

Two quick answers

- They make it is easy to make high quality apps
- They are reusable

View Controllers (Basics)

Design



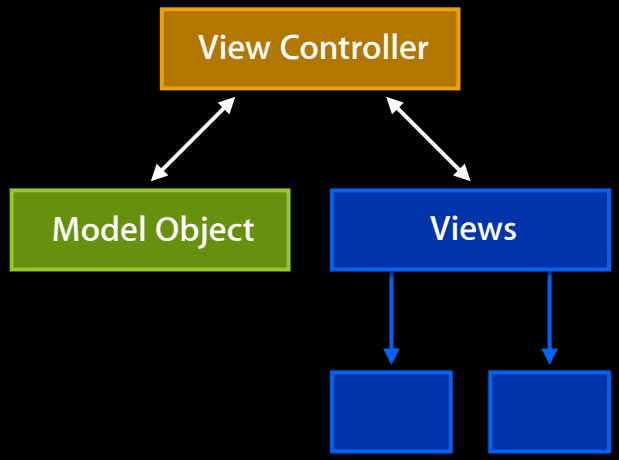
- They are the "C" in MVC
- They manage a "screenful of content"
- They are often "packaged" with their model object

Ready to Use System View Controllers

- TWTweetComposeViewController
- UIImagePickerController
- EKEventViewController
- MFMailComposeViewController
- MPMediaPickerController

View Controllers (Basics)

Design



- They are the "C" in MVC
- They manage a "screenful of content"
- They are often "packaged" with their model object
- Your application flows between view controllers

View Controllers (Basics)

Design

Table view controller
Left side of SVC

Detail controller
Right side of SVC

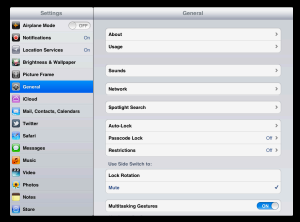
- They manage a "screenful of content" ?
- Their views have flexible bounds
- They manage a self-contained "unit of content"

View Controllers (Basics)

Design

- The root view controller does manage a “screenful of content”

Window

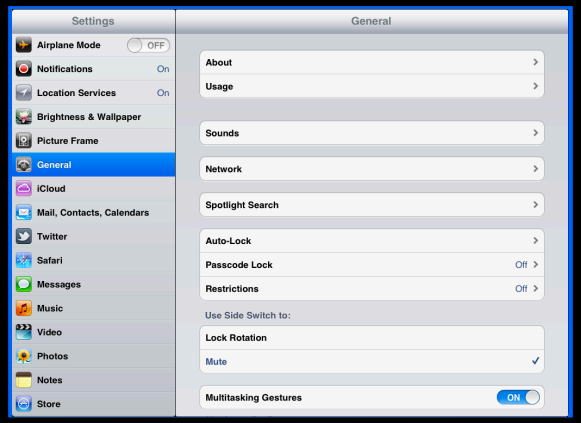


View Controllers (Basics)

Design

- The root view controller does manage a “screenful of content”

```
window.rootViewController = rootViewController;
```



View Controller (Basics)

- Custom view controllers
  - Subclass UIViewController
  - Associate the view controller with a view hierarchy
  - Override select API
  - Add your application logic
  - Do the app provisioning tango

View Controller (Basics)

Commonly used UIViewController callbacks

- Appearance callbacks
  - `- (void)viewWillAppear:`
  - `- (void)viewDidAppear:`
  - `- (void)viewWillDisappear:`
  - `- (void)viewDidDisappear:`
- Rotation callbacks
  - `- (void)viewWillRotateToInterfaceOrientation:duration:`
  - `- (void)viewWillAnimateRotationToInterfaceOrientation:`
  - `- (void)viewDidRotateFromInterfaceOrientation:`

View Controllers (Basics)

Summary

- View controllers are just controllers—the "C" in MVC
- View controllers manage a view hierarchy—not a single view!
- View controllers are typically self contained (reusable)
- View controllers connect and support common iOS application flows

Roadmap

Understanding view controller containers

- View controller and view hierarchies
- The three ways view controllers "connect" with each other
- Designing a custom container controller
- Discussion of new and changed API along the way

View Controller Containers

A tale of two hierarchies

A Tale of Two Hierarchies

Legend

Blue
View color

Blue Arrow
Subview relationship

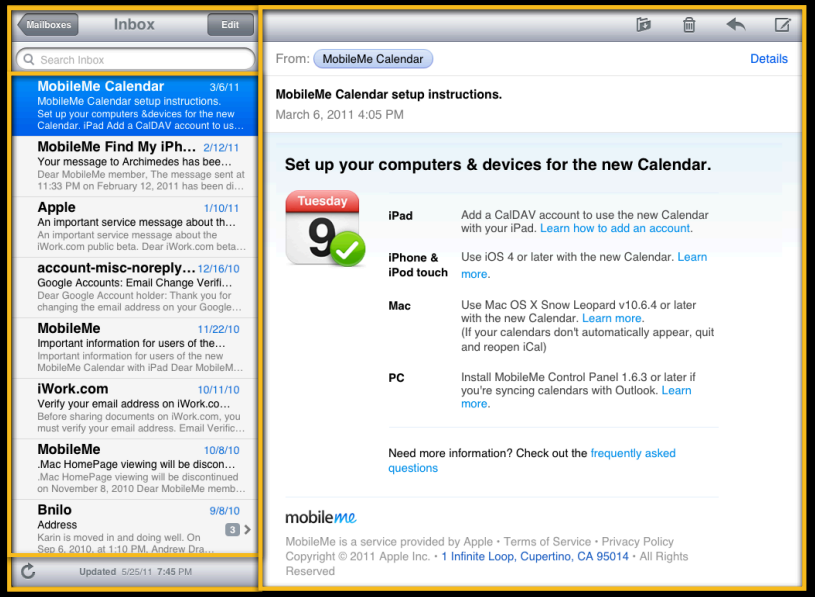
Gold
View controller color

Split View Controller

Gray Arrow
Parent view controller relationship

Containers Versus Content

Hierarchies



Split View Controller

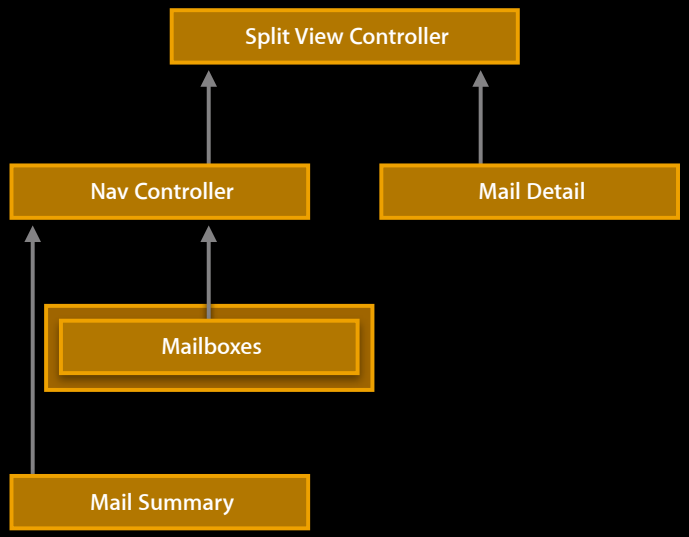
Nav Controller

Mail Detail

Mail Summary

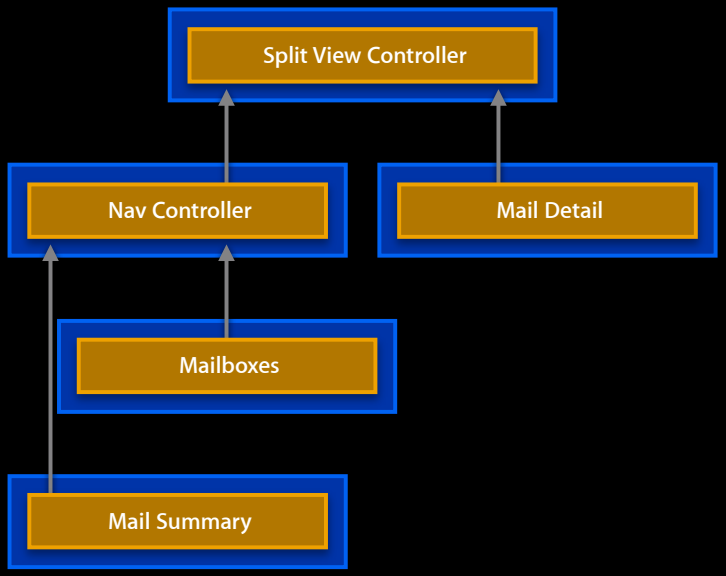
View Controller Containers

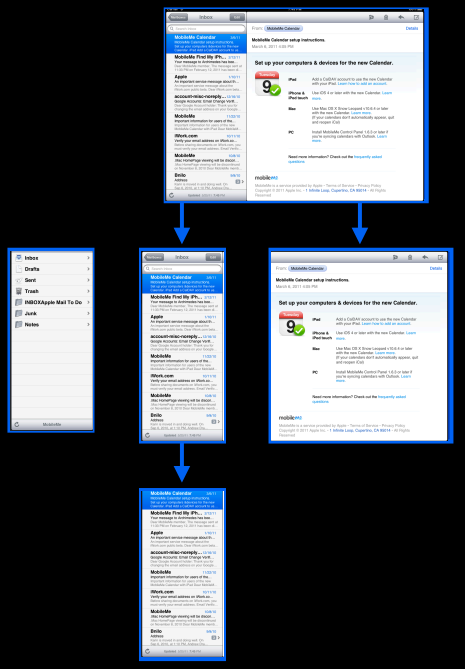
Hierarchies



View Controller Containers

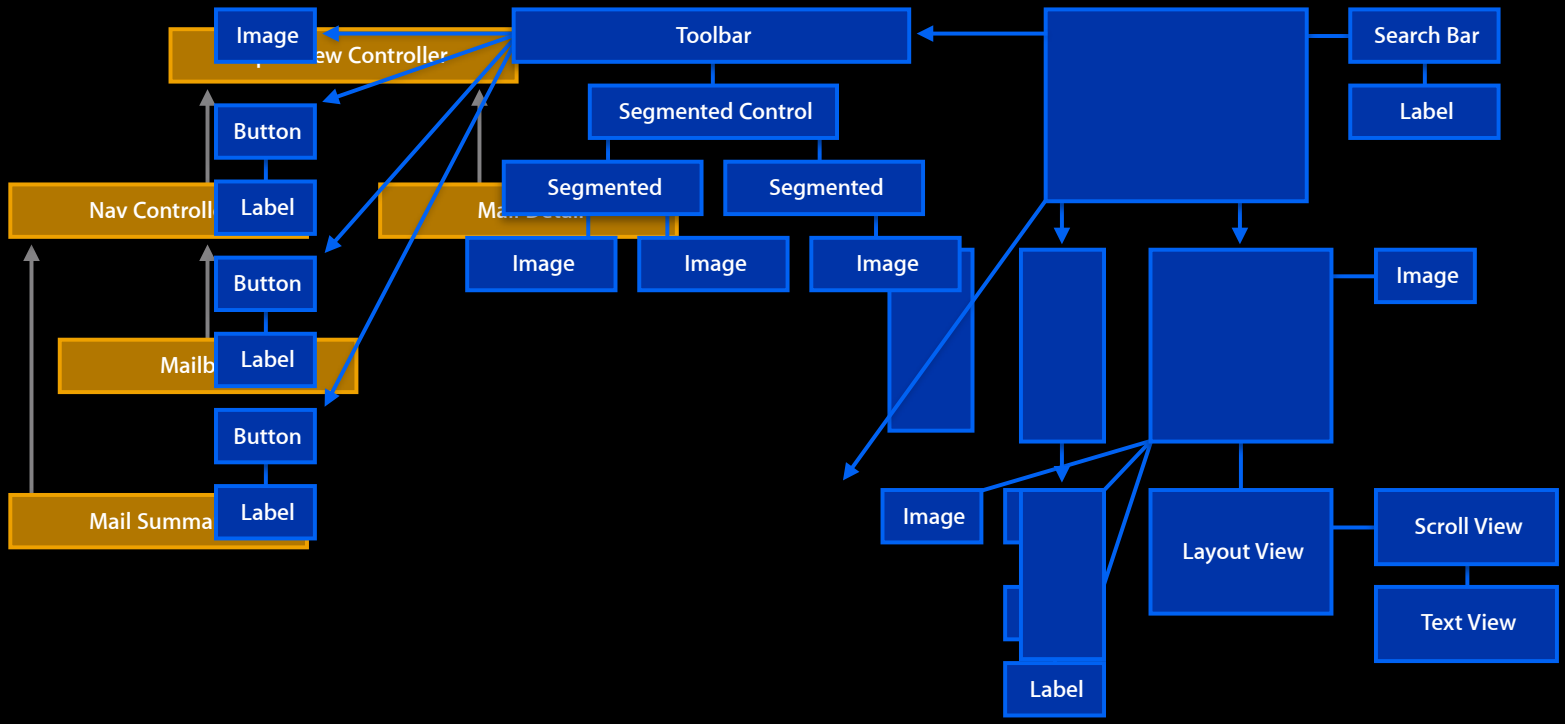
Hierarchies





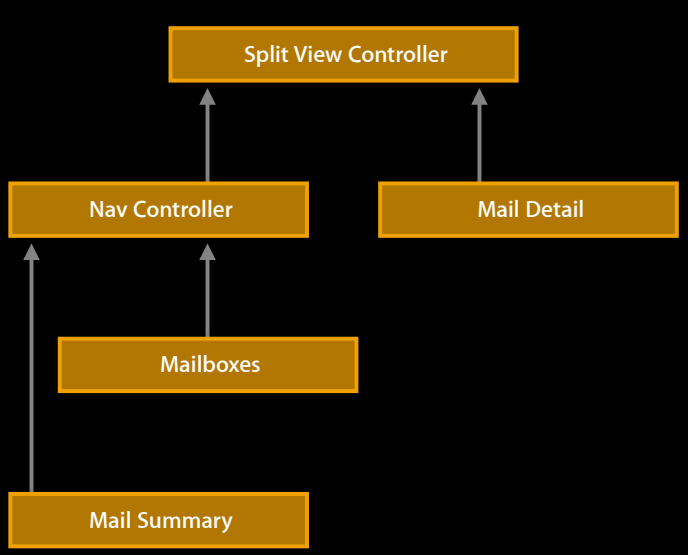
View Controller Containers

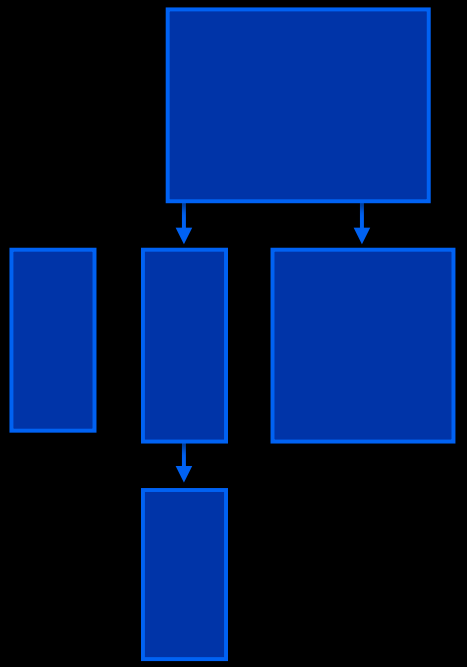
Hierarchies



View Controller Containers

Hierarchies





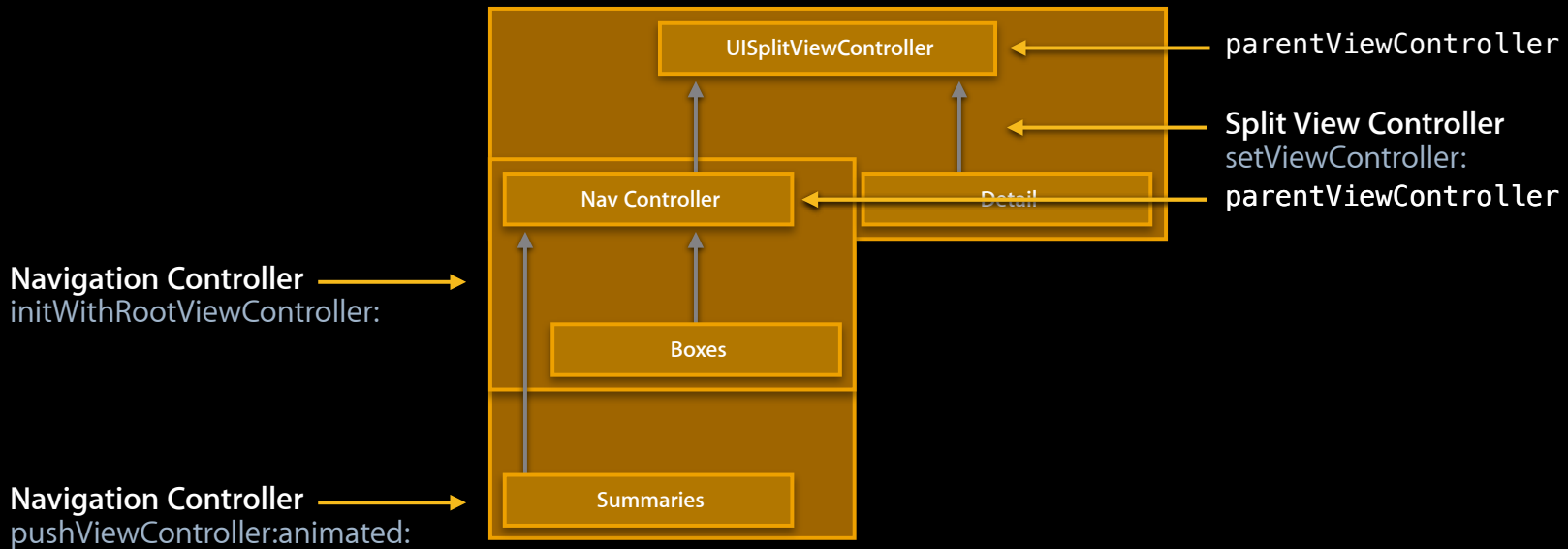
View Controller Containers

Hierarchies

- What you should know
  - Container controllers are responsible for child/parent relationships

View Controller Containers

API and the controller hierarchy



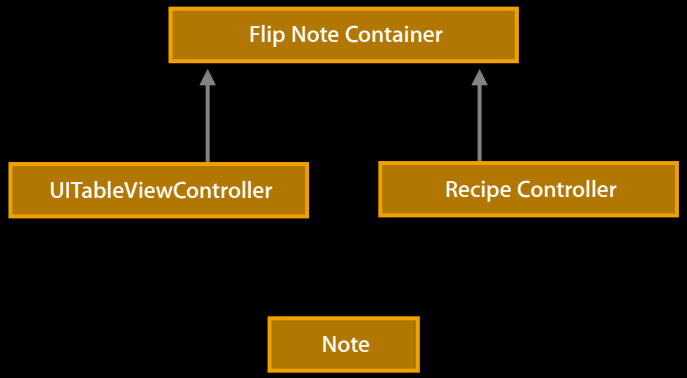
View Controller Containers

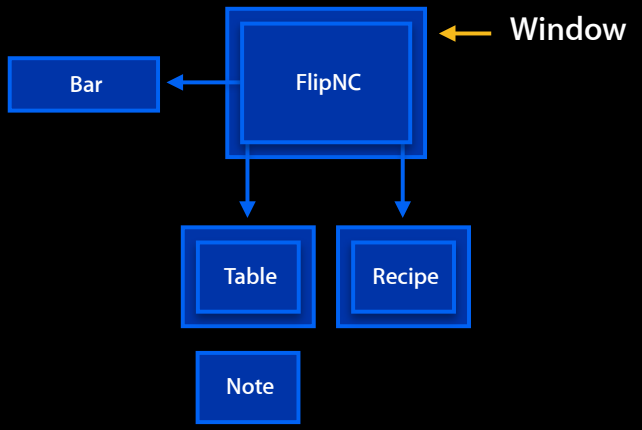
API and the controller hierarchy

Note

View Controller Containers

API and the controller hierarchy





View Controller Containers

API and the controller hierarchy

- Adding and removing child controllers

```
- (void)addChildViewController:(UIViewController *)childController;
- (void)removeFromParentViewController;
```

- Accessing child controllers

```
@property(nonatomic,readonly) NSArray *childViewControllers;
```

- Child callbacks

```
- (void)willMoveToParentViewController:(UIViewController *)parent;
- (void)didMoveToParentViewController:(UIViewController *)parent;
```

View Controller Containers

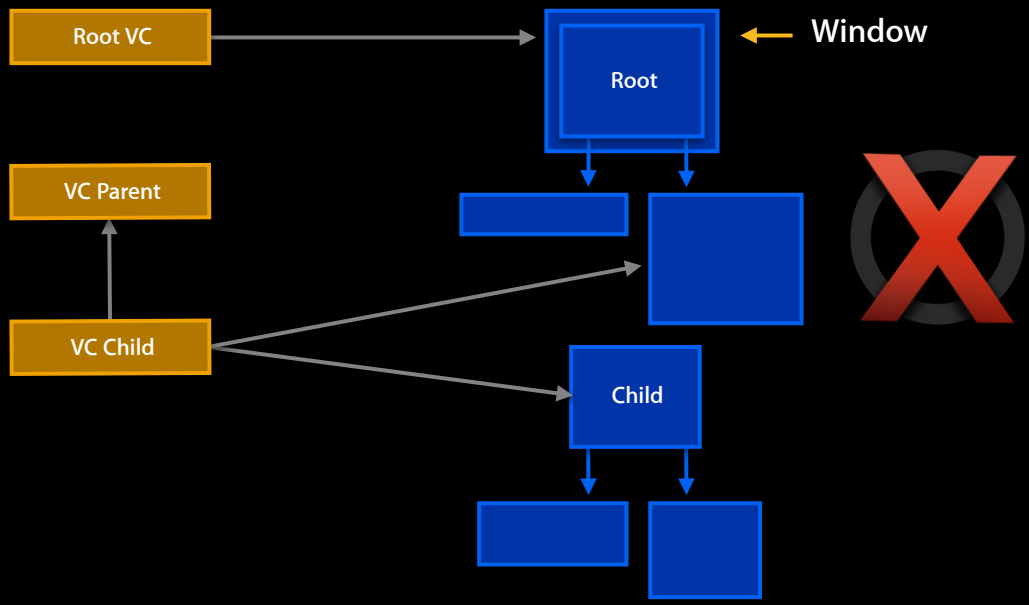
Hierarchies

- What you should know
  - Container controllers are responsible for child/parent relationships
  - There are consistent and inconsistent hierarchies

View Controller Containers

Inconsistent hierarchies

```
[root.view addSubview: child.view]
```



View Controller Containers

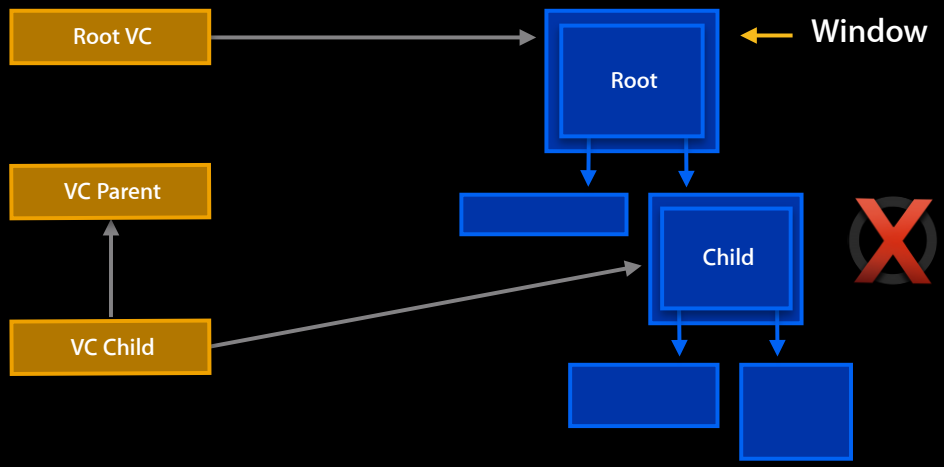
Inconsistent hierarchies


```
UIViewControllerHierarchyInconsistencyException
```

View Controller Containers

Inconsistent hierarchies

Why is this bad?



View Controller Containers

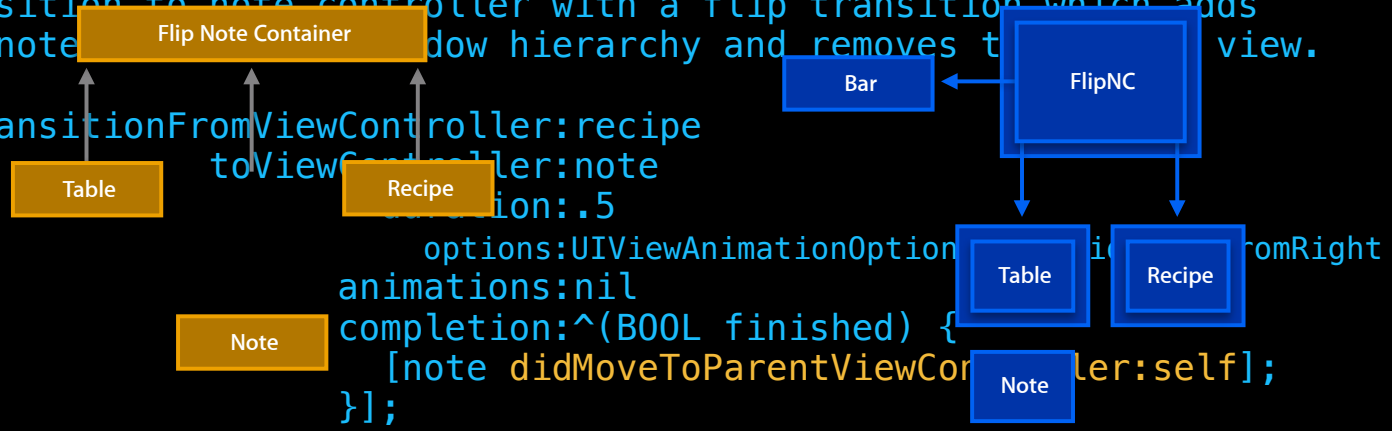
Hierarchies

- What you should know
  - Container controllers are responsible for child/parent relationships
  - There are consistent and inconsistent hierarchies
  - When are the appearance callbacks actually made

View Controller Containers

Appearance callbacks

```
[FNC flipToNote: note]

[self addChildViewController:note];

 // Transition to note controller with a flip transition which adds
 // tne note                    dow hierarchy and removes t          view.

[self transitionFromViewCont roller:recipe
                      toView  ller:note
                          ion:.5
                      options:UIViewAnimationOption      omRight
                   animations:nil
                   completion:^(BOOL finished) {
                     [note didMoveToParentViewCo      ler:self];
                   }];
```



View Controller Containers

Appearance callbacks

```
[self addChildViewController:note];

 // Transition to note controller with a flip transition which adds
 // tne note view to the window hierarchy and removes the recipe view.

[self transitionFromViewController:recipe
                  toViewController:note
                          duration:.5
                            options:UIViewAnimationOptionTransitionFlipFromRight
                        animations:nil
                        completion:^(BOOL finished) {
                          [note didMoveToParentViewController:self];
                        }];
```

View Controller Containers

Appearance callbacks

`viewWillAppear:`

- Called before the view is added to the windows' view hierarchy
- Called before `[vc.view layoutSubviews]` (if necessary)

`viewDidAppear:`

- Called after the view is added to the view hierarchy
- Called after `[vc.view layoutSubviews]` (if necessary)

`viewWillDisappear:`

- Called before the view is removed from the windows' view hierarchy

`viewDidDisappear:`

- Called after the view is removed from the windows' view hierarchy

View Controller Containers

Appearance callbacks

```
[self addChildViewController:note];

 // Transition to note controller with a flip transition which adds
 // tne note view to the window hierarchy and removes the recipe view.

[self transitionFromViewController:recipe
                  toViewController:note
                          duration:.5
                            options:UIViewAnimationOptionTransitionFlipFromRight
                        animations:nil
                        completion:^(BOOL finished) {
                          [note didMoveToParentViewController:self];
                        }];
```

View Controller Containers

API and the controller hierarchy

- Transitioning between children view controllers

```
- (void)transitionFromViewController:(UIViewController *) fromVC
                    toViewController:(UIViewController *)toVC
                            duration:(NSTimeInterval)duration
                             options:(UIViewAnimationOptions)options
                          animations:(void (^)(void))animations
                          completion:(void (^)(BOOL finished))completion;
```

- Laying out the view hierarchy

```
- (void)viewWillLayoutSubviews
- (void)viewDidLayoutSubviews
```

Connections

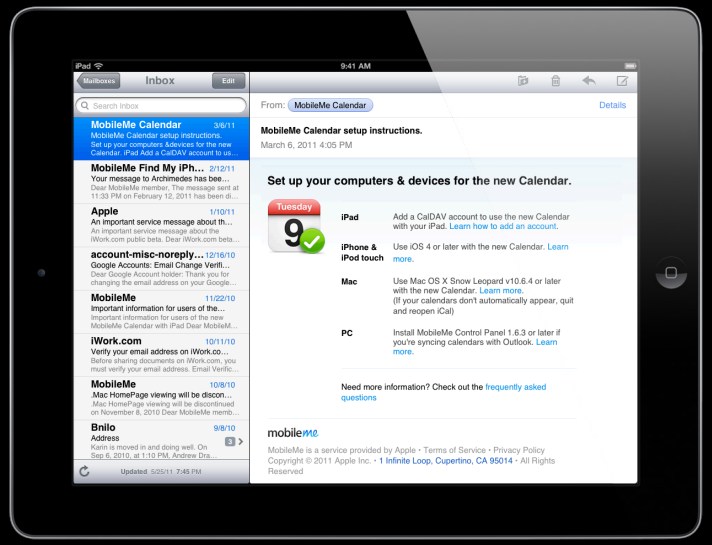
The flow—getting view controllers on and off the screen

Connecting View Controllers

- By container controllers
- By presentation and dismissal
- By direct view manipulation

Connecting View Controllers

Containers

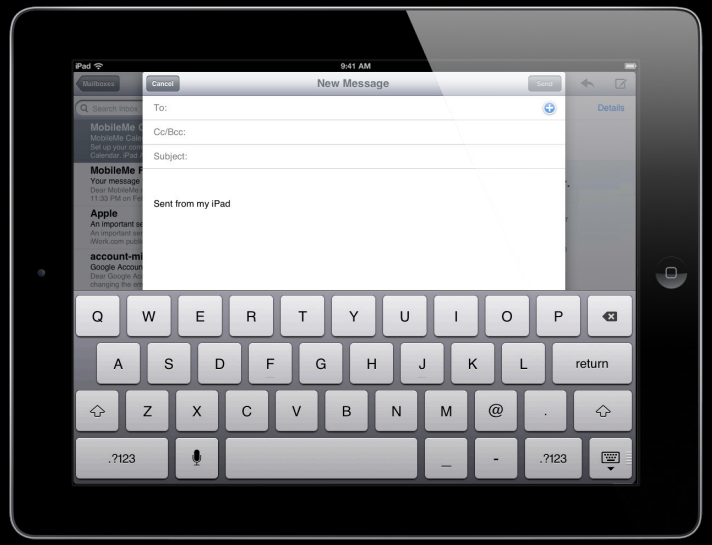



```
- (void)pushViewController:
                  animated:
- (void)popViewControllerAnimated:
```

Connecting View Controllers

Presentation and dismissal

Before 5




```
- (void) presentModalViewController:
                                animated:
- (void) dismissModalViewControllerAnimated:
```

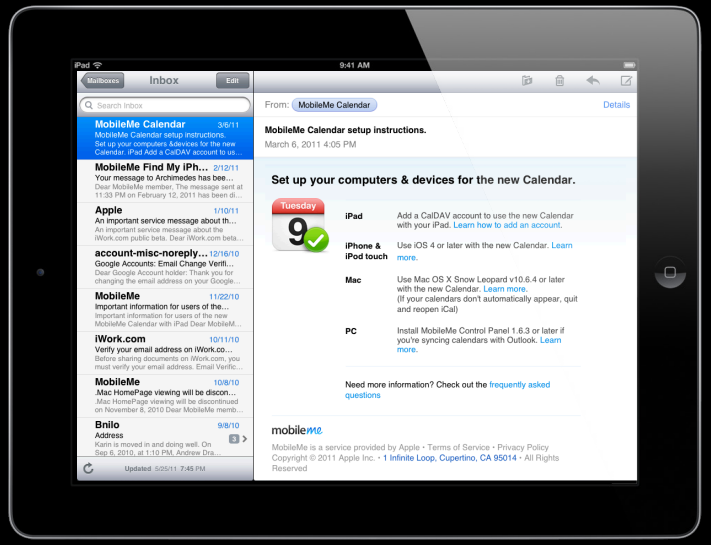
Connecting View Controllers

Presentation and dismissal


```
- (void)presentViewController: (UIViewController *)vc
                     animated: (BOOL)animated
                   completion: (void (^)(void))completion;

- (void)dismissViewControllerAnimated:(BOOL)animated
                           completion: (void (^)(void))completion;
```

Connecting View Controllers

Presentation and dismissal

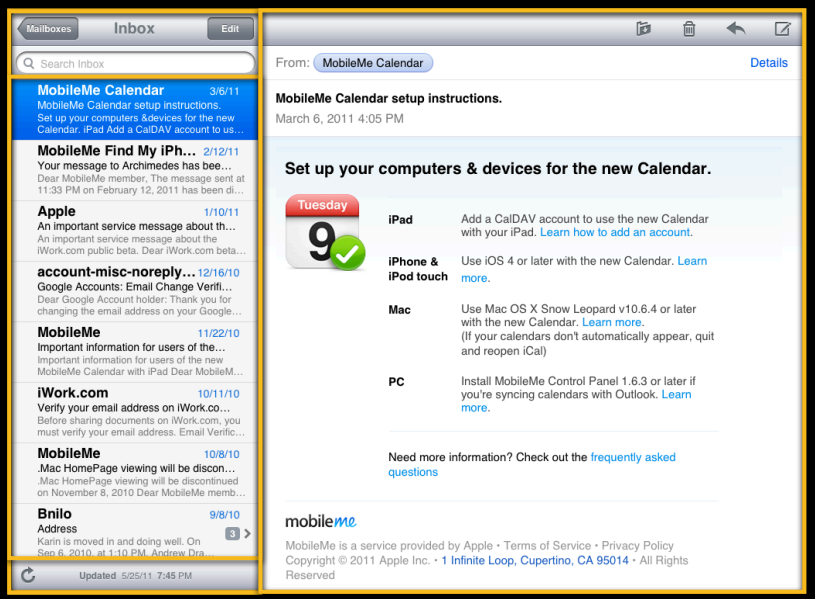


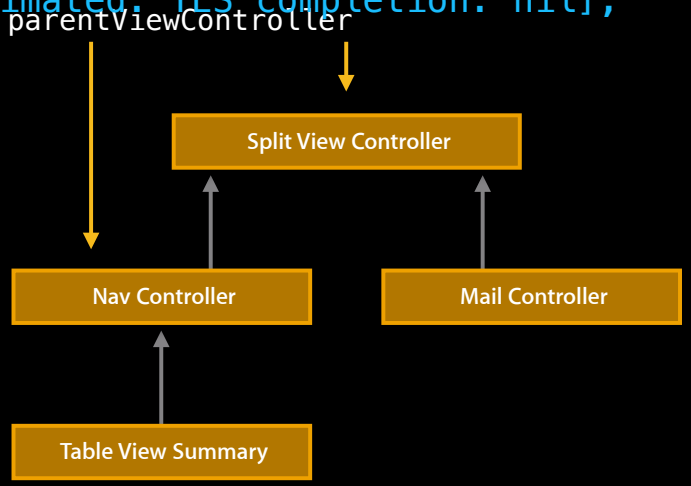

```
- (UIViewController *)parentViewController;

- (UIViewController *)presentingViewController;
```

Connecting View Controllers

Presentation and dismissal

```
composeSheet.modalPresentationStyle = UIModalPresentationFormSheet;
[tvc presentViewController: composeSheet animated: YES completion: nil];
```

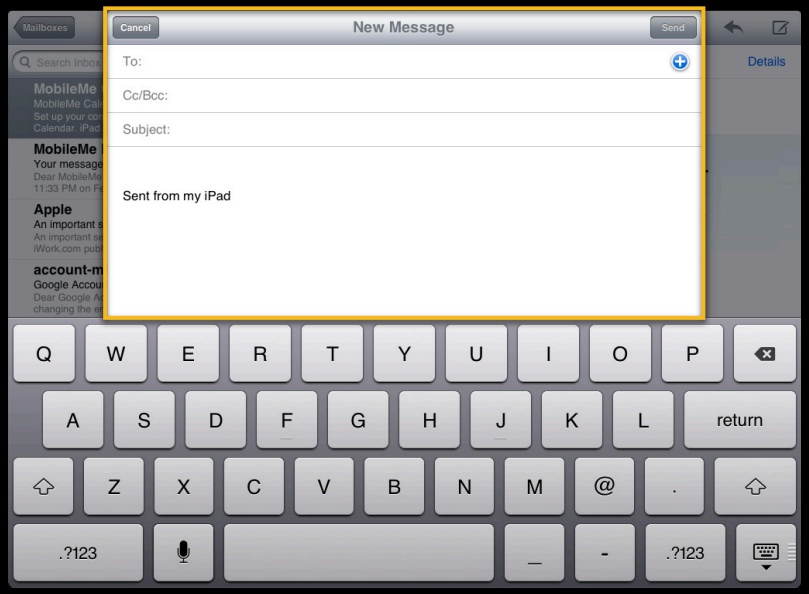


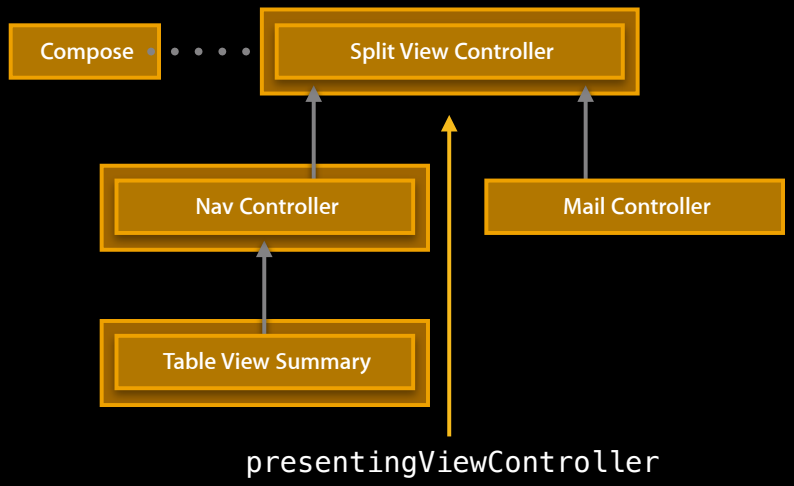


Connecting View Controllers

Presentation and dismissal

```
composeSheet.modalPresentationStyle = UIModalPresentationFormSheet;
[tvc presentViewController: composeSheet animated: YES completion: nil];
```





Connecting View Controllers

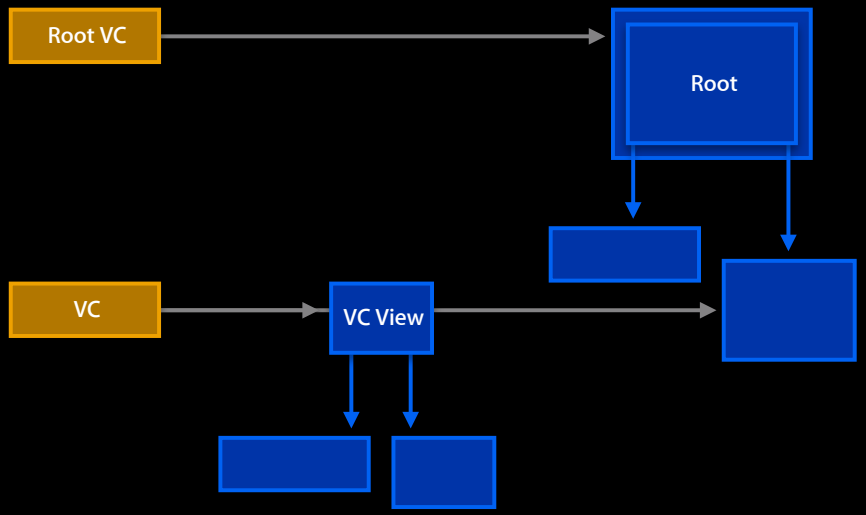
View manipulation

```
[window addSubview:rootViewController.view];
window.rootViewController = rootViewController;
```

Connecting View Controllers

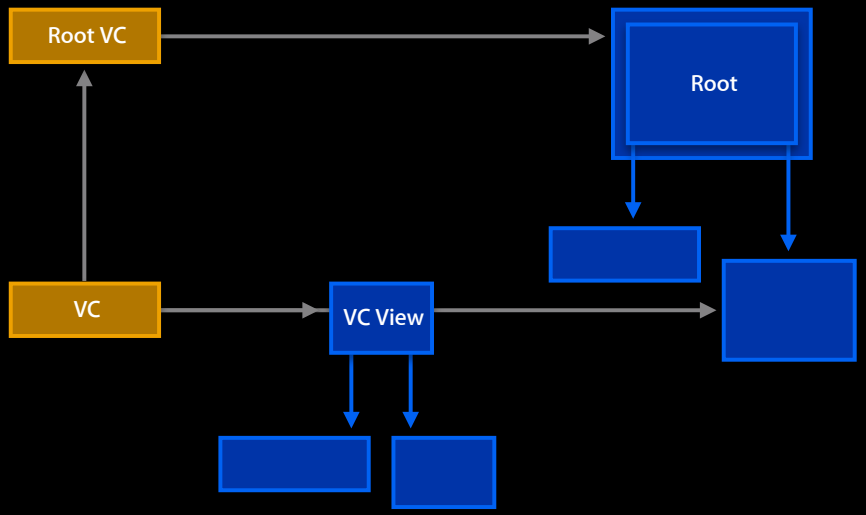
View manipulation

```
[root.someView addSubview: vc.view]
```



Connecting View Controllers

View manipulation




```
[rootVC addChildViewController: vc]
```

View Controller Containers

Inception

View Controller Containers

Inception

- When should you consider creating a custom view controller container?
  - Aesthetics
  - Custom application flows
  - Your application manipulates the view hierarchy directly

View Controller Containers

Inception

View Controller Containers

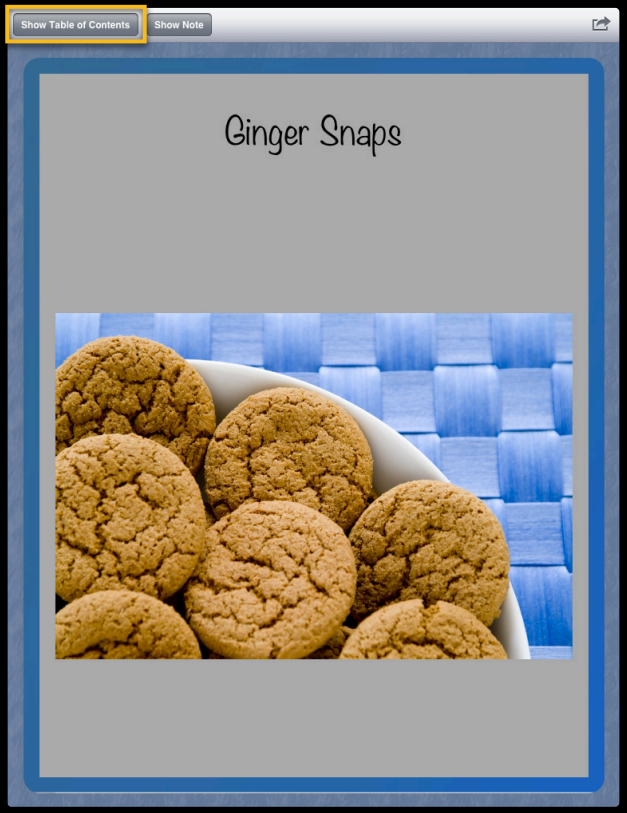
Inception

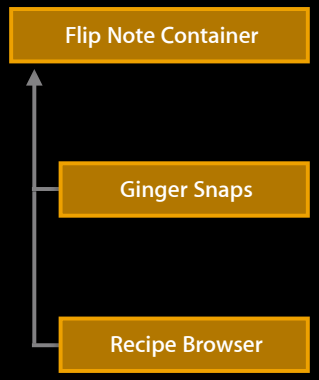
Only on iPad

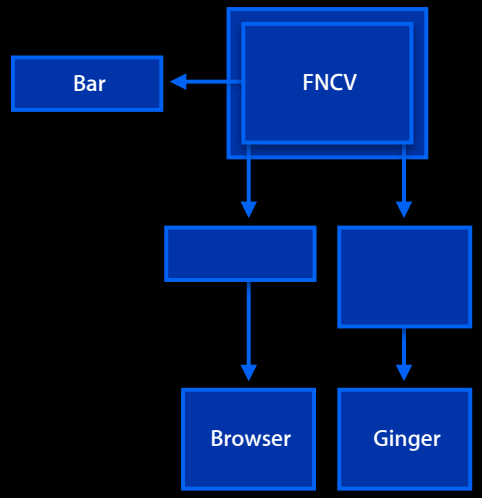

```
@protocol UISplitViewControllerDelegate
...
// Returns YES if a view controller should be hidden by
// the split view controller in a given orientation.
// (This method is only called on the leftmost view controller
// and only discriminates portrait from landscape.)
- (BOOL)splitViewController: (UISplitViewController*)svc
   shouldHideViewController:(UIViewController *)vc
              inOrientation:(UIInterfaceOrientation)orientation;
@end
```

Designing a New App Flow

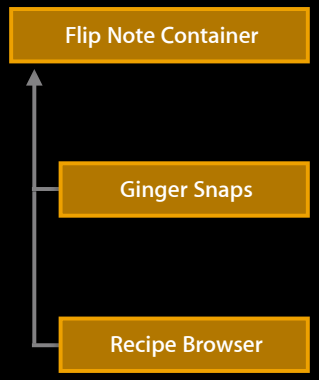
Creating the application flow for a revised recipe app

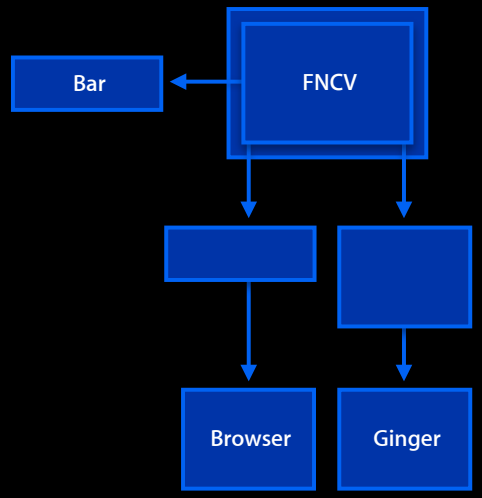

Show Table of Contents
Show Note
Ginger Snaps


Flip Note Container
Ginger Snaps
Recipe Browser

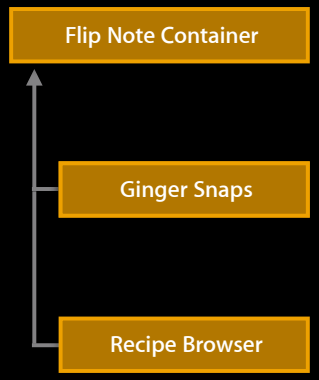

Bar
FNCV
Browser
Ginger

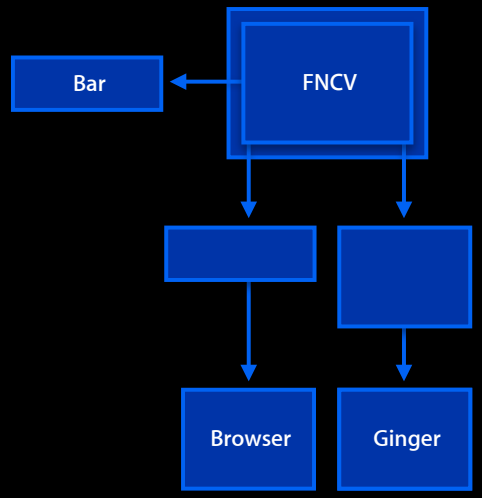

Hide Table of Contents
Show Note
Gaufres de Liège
Chocolate Cake
Ginger Snaps
Ginger Snaps


Flip Note Container
Ginger Snaps
Recipe Browser


Bar
FNCV
Browser
Ginger


Hide Table of Contents
Show Note
Gaufres de Liège
Chocolate Cake
Ginger Snaps
Ginger Snaps


Flip Note Container
Ginger Snaps
Recipe Browser


Bar
FNCV
Browser
Ginger

Container View Controller Demo

Creating the application flow for a revised recipe app

View Controller Containers

Demo highlights—moving in and out of containers

View Controller Containers

Moving in and out of containers

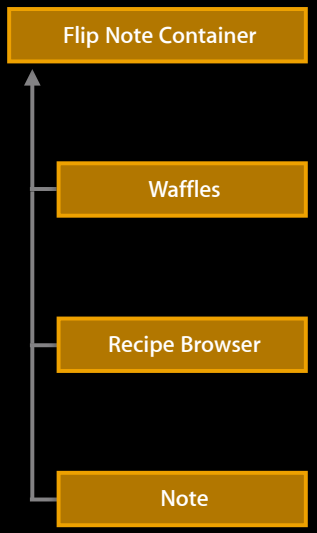
```
- (IBAction)flipToNote
{
  if(...) {
 ...
  [self addChildViewController:_noteController];
  [self transitionFromViewController:_contentController
                    toViewController:_noteController duration:.5
                             options:UIViewAnimationOptionTransitionFlipFromRight
                          animations:nil
                          completion:^(BOOL finished) {
                             _flipNoteButton.title = @"Hide Note";
                             _flipNoteButton.action = @selector(flipFromNote);
                            [_noteController didMoveToParentViewController:self];
                          }];
  }
}
```

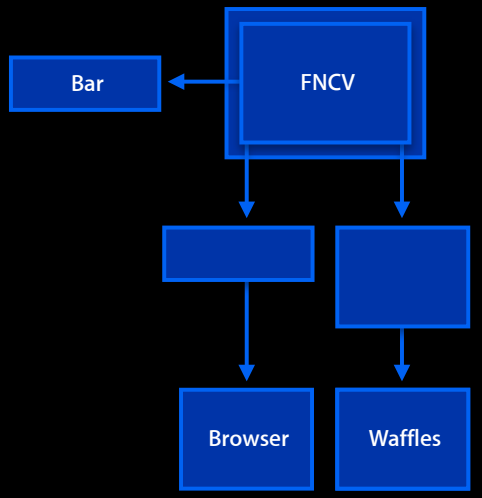
View Controller Containers

Moving in and out of containers

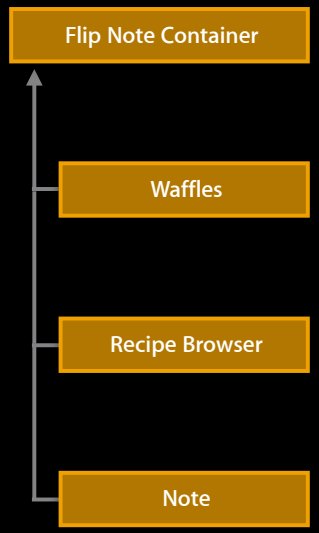
```
- (IBAction)flipFromNote
{
  if(_isNoteBeingShown) {
    [_noteController willMoveToParentViewController:nil];
    [self transitionFromViewController:_noteController
                      toViewController:_contentController duration:0.5
                               options:UIViewAnimationOptionTransitionFlipFromLeft
                            animations:nil
                            completion:^(BOOL finished) {
                                _flipNoteButton.title = @"Show Note";
                                _flipNoteButton.action = @selector(flipToNote);
                                [_noteController removeFromParentViewController];
                                _isNoteBeingShown = NO;
                             }];
  }
}
```

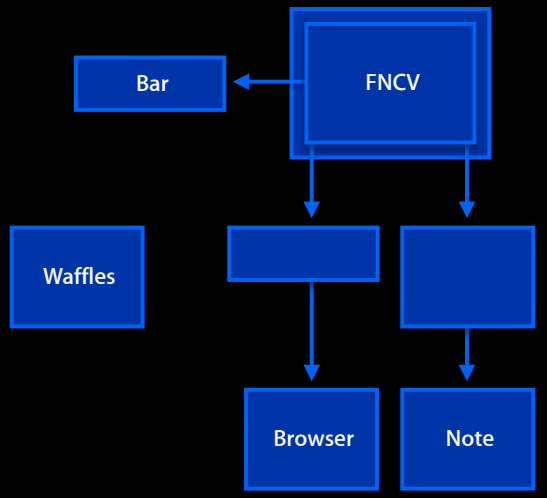

Hide Table of Contents
Show Note
Gaufres de Liège
Chocolate Cake
Ginger Snaps
Gaufres de Liège


Flip Note Container
Waffles
Recipe Browser
Note


Bar
FNCV
Browser
Waffles


Hide Table of Contents
Hide Note
Gaufres de Liège
Chocolate Cake
Ginger Snaps
These waffles are to die for!


Flip Note Container
Waffles
Recipe Browser
Note


Bar
FNCV
Waffles
Browser
Note

View Controller Containers

Moving in and out of containers

```
- (void)viewDidAppear:(BOOL)animated
{
  [super viewDidAppear:animated];
  if(![self isMovingToParentViewController]) {
   [[self parentViewController] updateSelectionForListOfContentIdentifiersIfNecessary];
  }
}
```

View Controller Containers

Inception


```
- (BOOL)isMovingToParentViewController;   // Used in appearance callbacks
- (BOOL)isMovingFromParentViewController; // Used in disappearance callbacks


- (BOOL)isBeingPresented;
- (BOOL)isBeingDismissed;
```

View Controller Containers

Moving in and out of containers


```
- (BOOL)automaticallyForwardAppearanceAndRotationMethodsToChildViewControllers;
```

View Controller Containers

Moving in and out of containers

```
- (IBAction)flipToNote
{
  if(...) {
       ...
    [self addChildViewController:_noteController];
    [_noteController viewWillAppear: YES];
    // Some fancy animation that culminates in the view swap
    // E.g [[_contentController.view superview] addSubview:_noteController.view];
       ...

     // Finally this is usually called in a completion handler
     // after the animation completes
    [_noteController viewDidAppear: YES];
    [_noteController didMoveToParentViewController:self];
  }
}
```

Container View Controller Demo

Creating the application flow for a revised recipe app

View Controller Containers

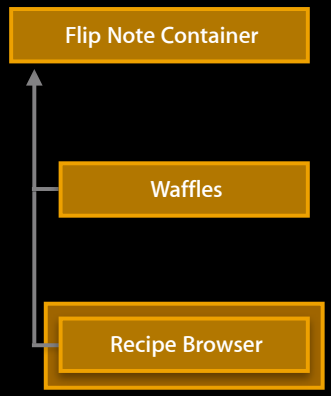
Demo highlights—defining presentation context

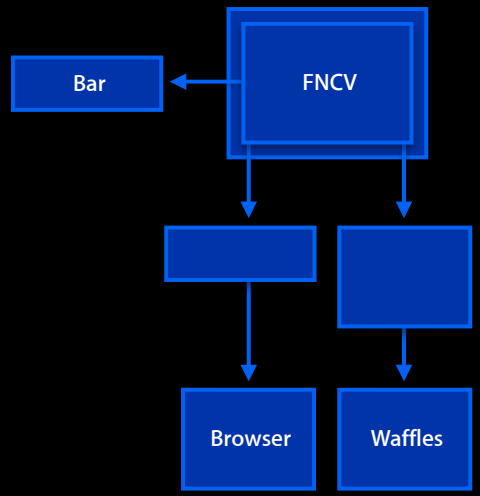
View Controller Containers

Defining presentation context

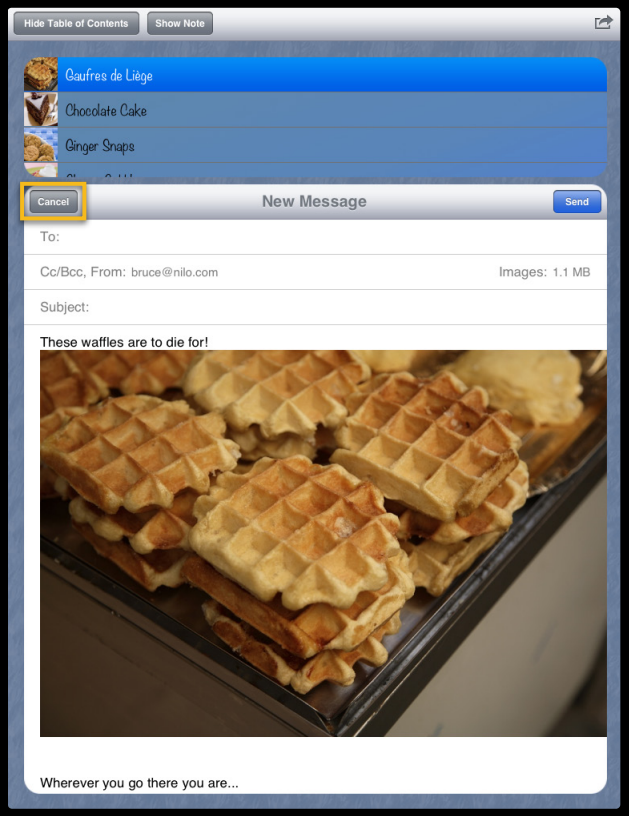
Only on iPad

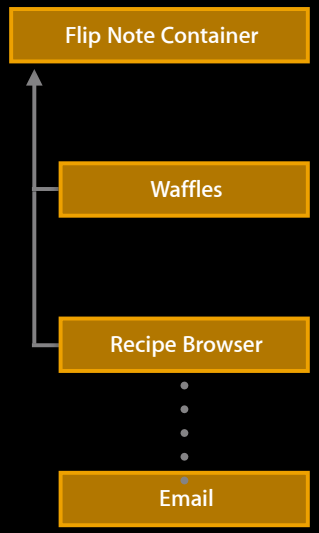
```
- (IBAction)emailContent
{
  UIViewController *presenter = _isNoteBeingShown ? _noteController :
                                                    _contentController;
  ...
  mailController.modalPresentationStyle = UIModalPresentationCurrentContext;
  if(_contentController && [MFMailComposeViewController canSendMail]) {
   ...
    data = [_contentProvider dataForContentIdentifier:self.contentControllerIdentifier
                                                      mimeType:&mimeType];
    note = [_contentProvider noteForContentIdentifier:self.contentControllerIdentifier];
   ...
  [presenter presentViewController:mailController
                          animated:YES
                        completion:^{[mailController release];}];
  }
}
```

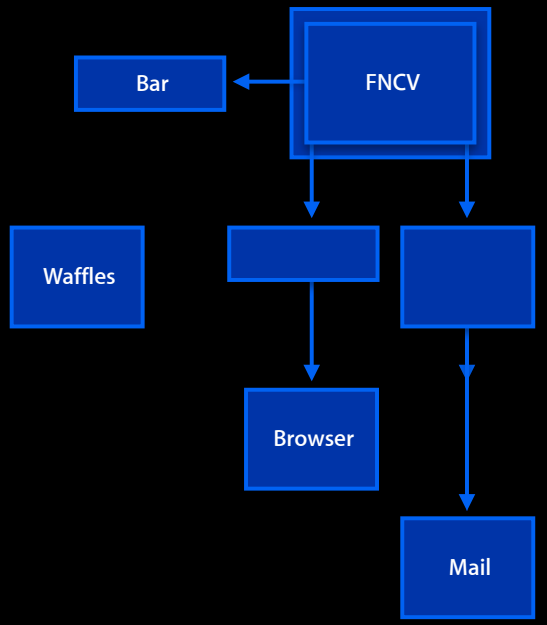


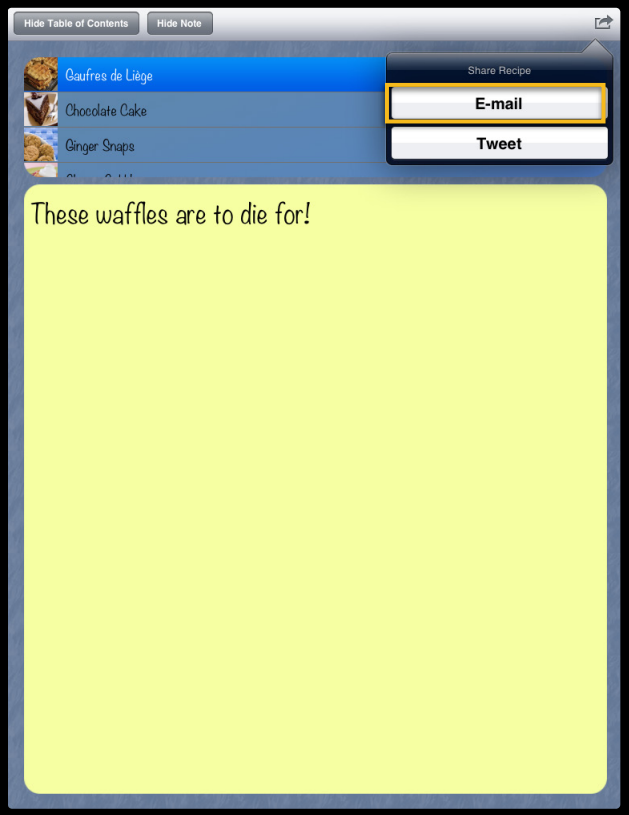


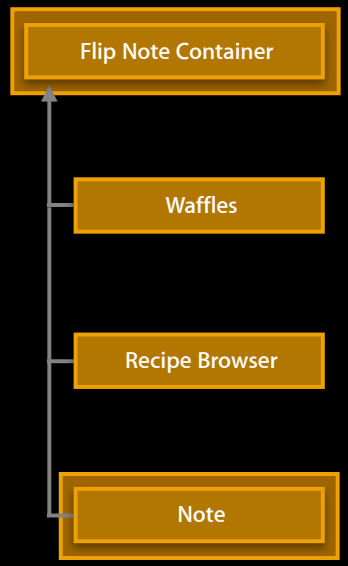

```
mc.modalPresentationStyle = UIModalPresentationCurrentContext;
[rb presentViewController:mailController
                 animated:YES
               completion:^{...}];
```

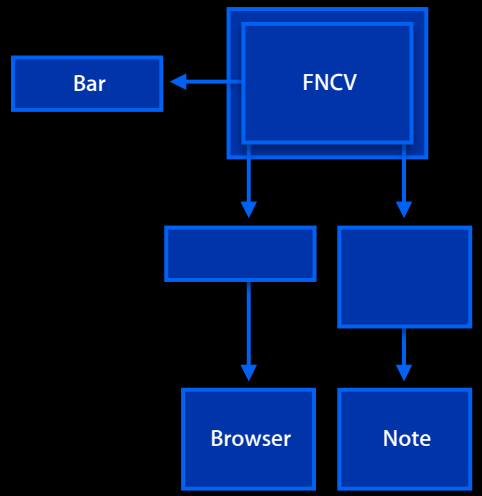

Hide Table of Contents
Show Note
Gaufres de Liège
Chocolate Cake
Ginger Snaps
Cancel
New Message
Send
To:
Cc/Bcc, From: bruce@nilo.com
Images: 1.1 MB
Subject:
These waffles are to die for!
Wherever you go there you are...


Flip Note Container
Waffles
Recipe Browser
Email

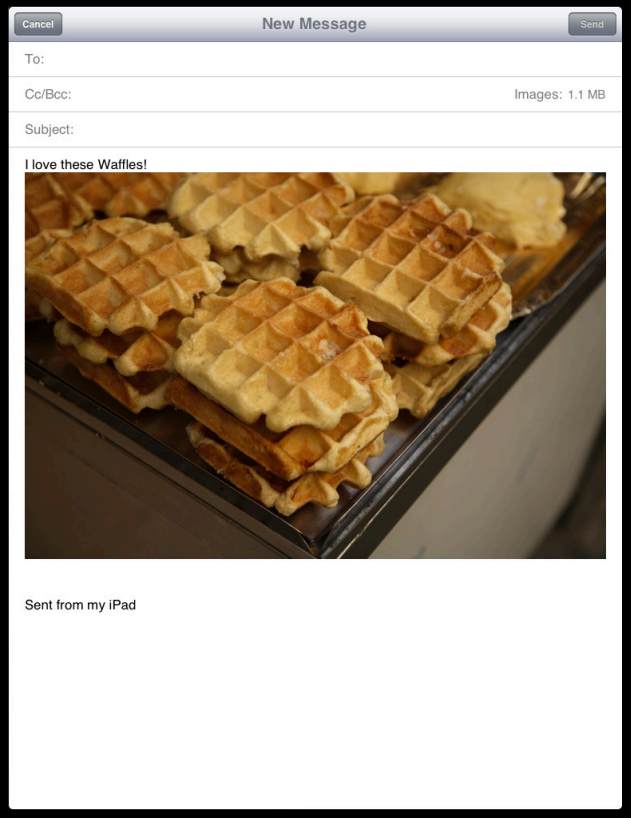

Bar
FNCV
Waffles
Browser
Mail

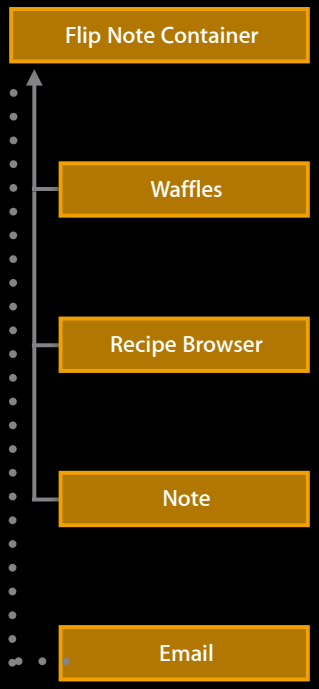


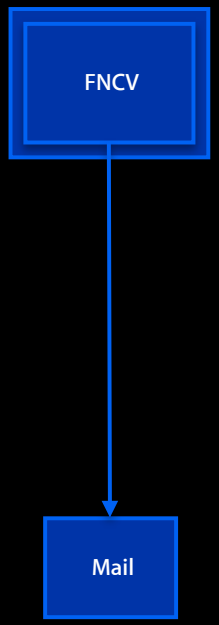





```
mc.modalPresentationStyle = UIModalPresentationCurrentContext;
[note presentViewController:mailController
                   animated:YES
                 completion:^{...}];
```


Cancel
New Message
Send
To:
Cc/Bcc:
Images: 1.1 MB
Subject:
I love these Waffles!
Sent from my iPad


Flip Note Container
Waffles
Recipe Browser
Note
Email


FNCV
Mail

View Controller Containers

Defining presentation context

```
- (void)viewDidLoad
{
  [super viewDidLoad];
  self.view.autoresizingMask = UIViewAutoresizingFlexibleWidth |
                               UIViewAutoresizingFlexibleHeight;
  self.definesPresentationContext = YES;
  ...
}
```

View Controller Containers

Defining presentation context


```
@property(nonatomic,assign) BOOL definesPresentationContext;

// A controller that defines the presentation context can also
// specify the modal transition style if this property is true.

@property(nonatomic,assign) BOOL providesPresentationContextTransitionStyle;
```

View Controller Containers

Summary

View Controller Containers

Summary

- Use existing containers if possible
  - A view controller can manage more than one view!
  - Not every view needs a view controller
- Create custom view controller containers when needed
  - To define new application flows or appearances
  - Instead of direct view manipulation
    - This will future-proof your apps
- The API is simple
  - But understand your hierarchies

UIPageViewController

Matt Gamble

UIPageViewController

Navigate among views with a page curl transition

UIPageViewController

A container view controller

- Manages child view controllers that present content
- Presents a prepared application flow

UIPageViewController

Initialization

```
- initWithTransitionStyle:
    navigationOrientation:
                  options:
```

UIPageViewController

Initialization

```
UIPageViewController *myPVC = [[UIPageViewController alloc]
    initWithTransitionStyle:UIPageViewControllerTransitionStylePageCurl
      navigationOrientation:UIPageViewControllerNavigationOrientationHorizontal
                    options:[NSDictionary dictionaryWithObjectsAndKeys:
    [NSNumber numberWithInteger:UIPageViewControllerSpineLocationMid],
    UIPageViewControllerOptionSpineLocationKey]];
```

UIPageViewController

Initial view controllers

```
- setViewControllers:
                direction:
                animated:
              completion:
```

UIPageViewController

UIViewController

UIPageViewController

Initial view controllers

```
[myPVC setViewControllers:[NSArray arrayWithObjects:firstVC, secondVC, nil]
                direction:UIPageViewControllerNavigationDirectionForward
                 animated:NO
               completion:nil];
```

UIPageViewController

Programmatic navigation

```
[myPVC setViewControllers:[NSArray arrayWithObjects:thirdVC, fourthVC, nil]
                direction:UIPageViewControllerNavigationDirectionForward
                 animated:YES
               completion:nil];
```

```
^(BOOL finished) {
    NSLog(@"Page curl completed.");
}];
```

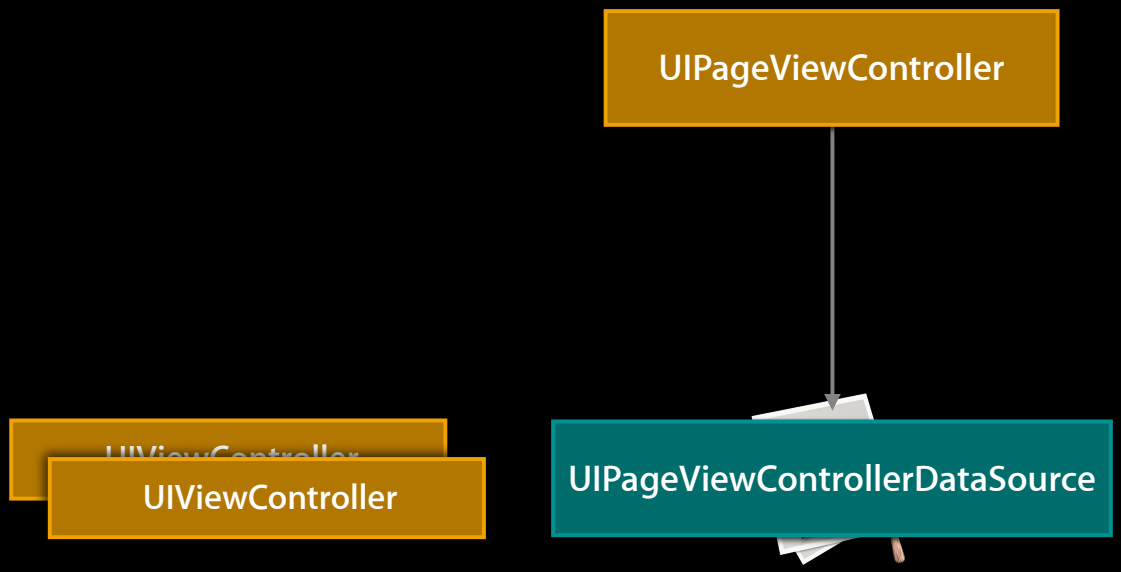
UIPageViewController

Programmatic navigation

```
[myPVC setViewControllers:[NSArray arrayWithObjects:thirdVC, fourthVC, nil]
                direction:UIPageViewControllerNavigationDirectionForward
                 animated:YES
               completion:^(BOOL finished) {
                   NSLog(@"Page curl completed.");
               }];
```

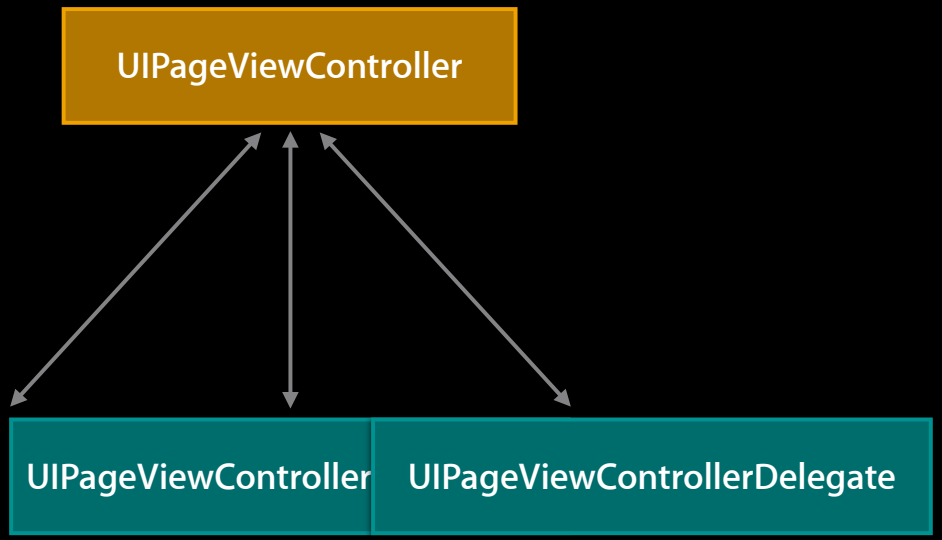
UIPageViewController

User-driven navigation



UIPageViewController

Structure



Demo

Demo Summary

- What we knew going in:
  - Initialize page view controller with transition style, navigation orientation, and any options (spine location)
  - Set initial view controllers (and drive programmatic navigation)
  - Allow user-driven navigation by setting a data source
- What we learned coming out:
  - Customize gesture area by placing gesture recognizers
  - Change spine location on rotation with delegate

Summary

- Understand the difference between content and container view controllers
- Use custom view controller containers…
  - …to define new application flows or looks
  - …in place of direct view manipulation
- Use existing containers if possible
  - UINavigationController, UITabBarController, UISplitViewController, etc.
  - New container view controller, UIPageViewController

More Information

Bill Dudney
iOS Apps & Frameworks Evangelist
dudney@apple.com

Documentation
Mac OS X Human Interface Guidelines
http://developer.apple.com/ue

Apple Developer Forums
http://devforums.apple.com

Related Sessions

What's New in Cocoa Touch	Pacific Heights Thursday 4:30–5:30PM
Introducing Interface Builder Storyboarding	Russian Hill Thursday 10:15–11:15AM
Twitter Integration	Presidio Thursday 2:00–3:00PM

Labs

Cocoa Touch Lab	Application Frameworks Lab D Wednesday 2:00PM–5:30PM
UIViewController Lab	Application Frameworks Lab B Tuesday 2:00PM–4:15PM


WWDC2011