The Evolution of View Controllers on iOS

Session 236

Matt Gamble, Bruce D. Nilo
UIKit Engineers

The Evolution of View Controllers

Roadmap

- Looking forward to today
- View controllers today and new directions

Looking Forward to Today

The “why” and “how” of view controllers

Matt Gamble
UIKit Engineer

Why UIViewController?

Make common tasks simpler

Why UIViewController?

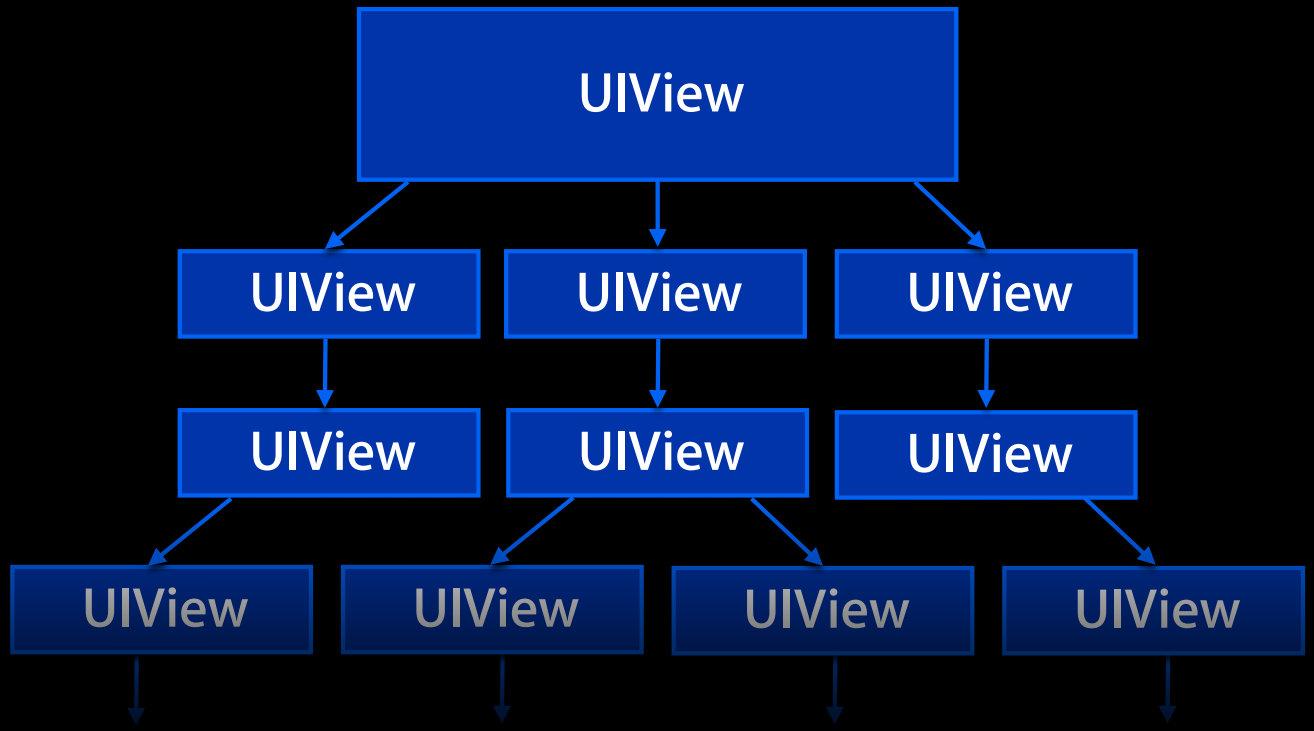
Manage a view hierarchy

Why UIViewController?

Manage a view hierarchy

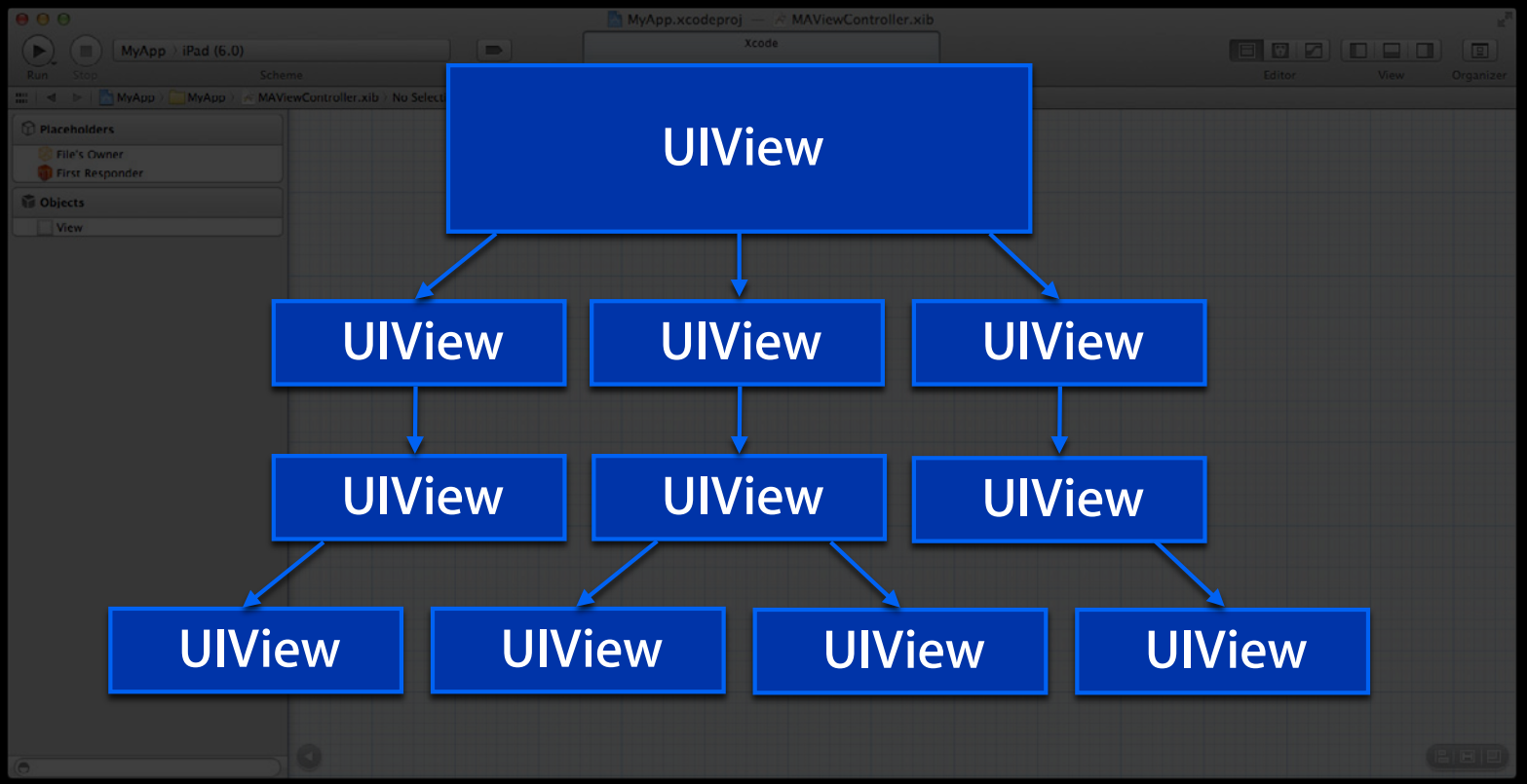
Why UIViewController?

Manage a view hierarchy



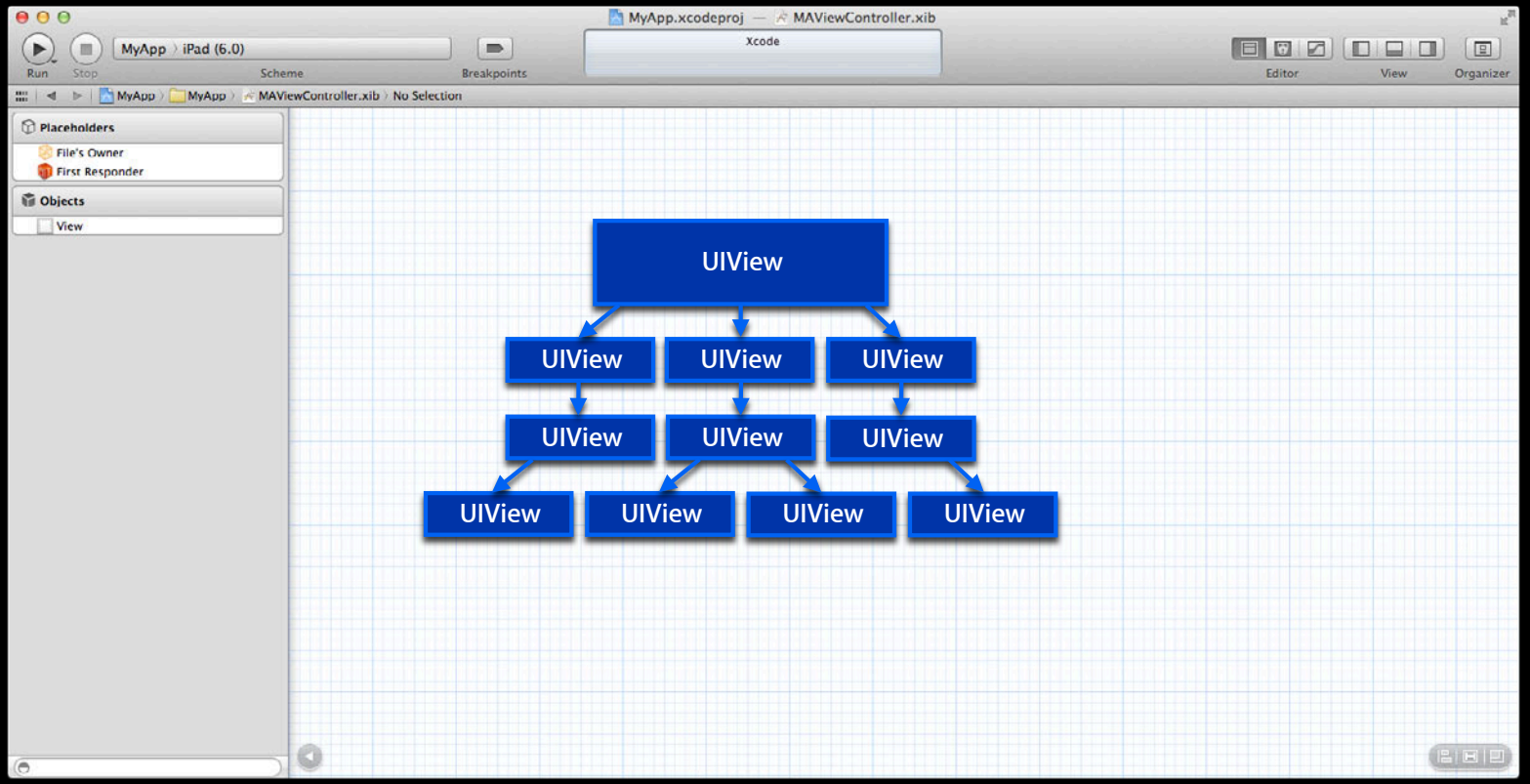
Why UIViewController?

Manage a view hierarchy



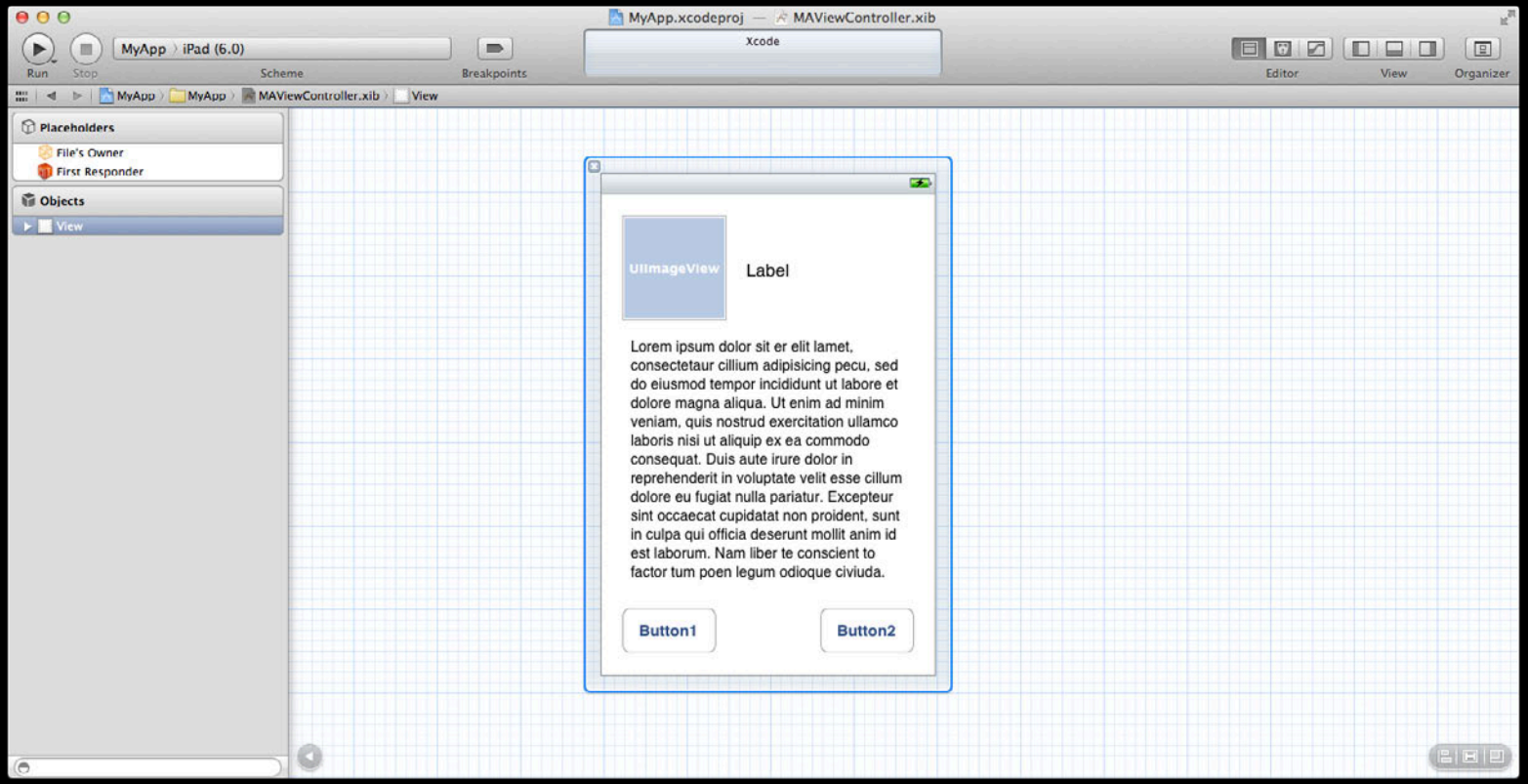
Why UIViewController?

Manage a view hierarchy



Why UIViewController?

Optionally load view from a nib



Why UIViewController?

Make common tasks simpler

- Manage a view
  - More specifically, a view hierarchy
  - Optionally load view from a nib

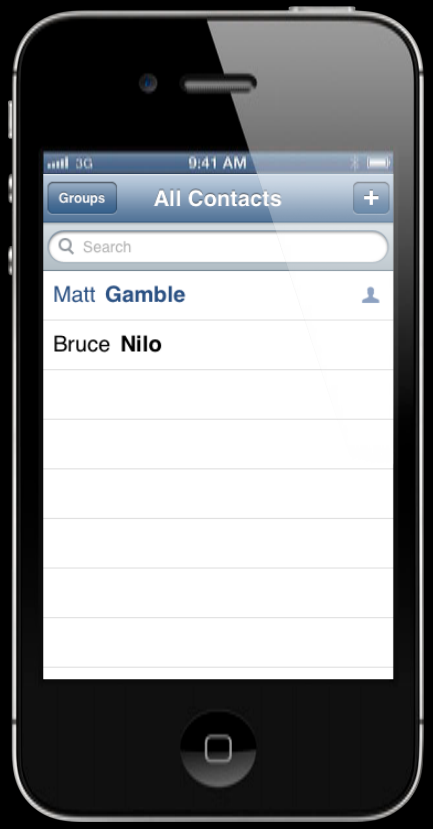
Why UIViewController?

Make common tasks simpler

- Manage a view
  - More specifically, a view hierarchy
  - Optionally load view from a nib
  - Convenient appearance calls

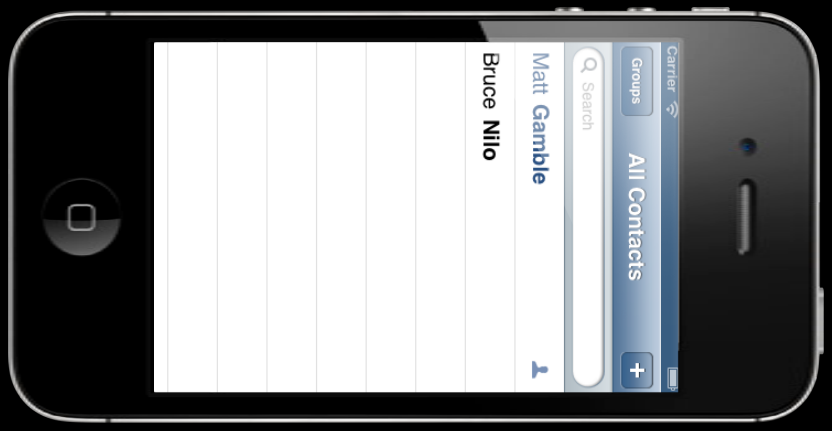
Why UIViewController?

Autorotation



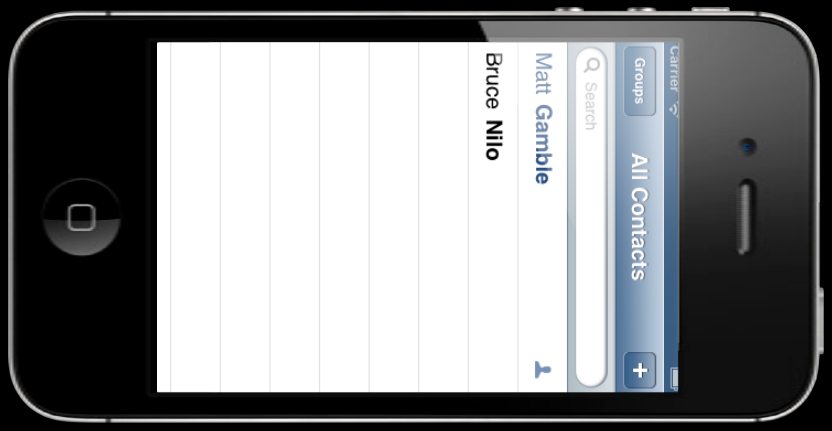
Why UIViewController?

Autorotation



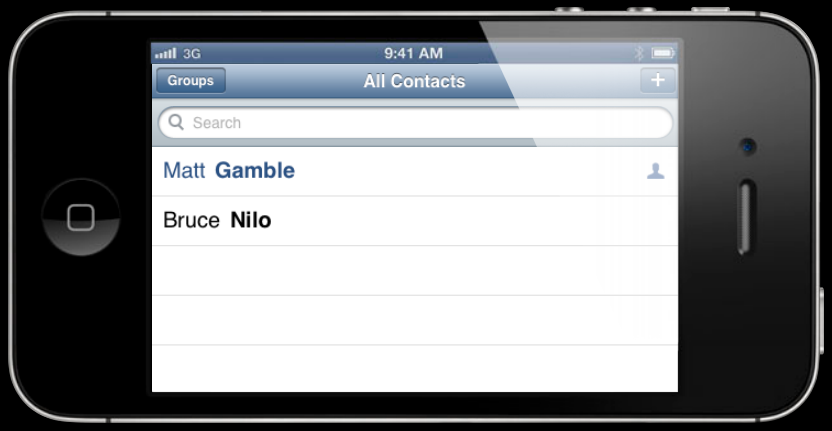
Why UIViewController?

Autorotation



Why UIViewController?

Autorotation



Why UIViewController?

Autorotation

Why UIViewController?

Autorotation

```
- (void)willRotateToInterfaceOrientation:(UIInterfaceOrientation)toInterfaceOrientation
                                duration:(NSTimeInterval)duration;
```

Why UIViewController?

Autorotation

```
- (void)willRotateToInterfaceOrientation:(UIInterfaceOrientation)toInterfaceOrientation
                                duration:(NSTimeInterval)duration;

- (void)willAnimateRotationToInterfaceOrientation:(UIInterfaceOrientation)toInterfaceOrientation
                                         duration:(NSTimeInterval)duration
```

Why UIViewController?

Autorotation

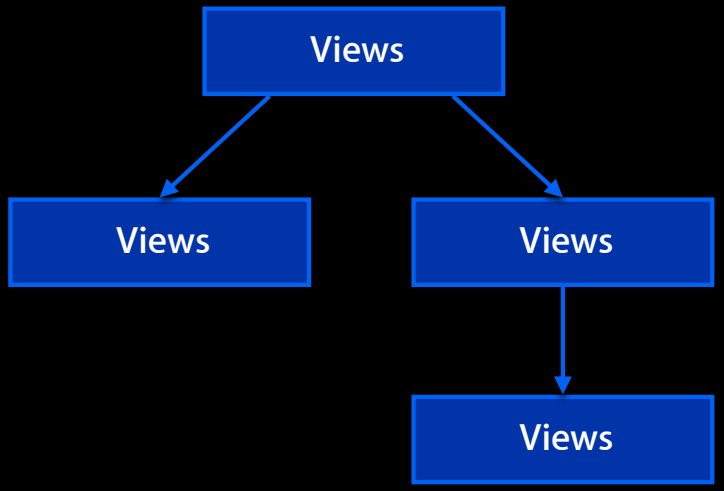
```
- (void)willRotateToInterfaceOrientation:(UIInterfaceOrientation)toInterfaceOrientation
                                duration:(NSTimeInterval)duration;

- (void)willAnimateRotationToInterfaceOrientation:(UIInterfaceOrientation)toInterfaceOrientation
                                         duration:(NSTimeInterval)duration

- (void)didRotateFromInterfaceOrientation:(UIInterfaceOrientation)fromInterfaceOrientation
```

Why UIViewController?

Centralize responsibility

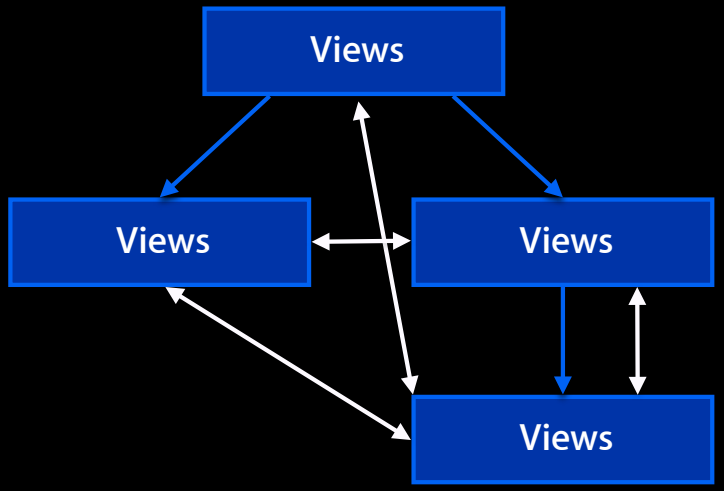
Why UIViewController?

Centralize responsibility



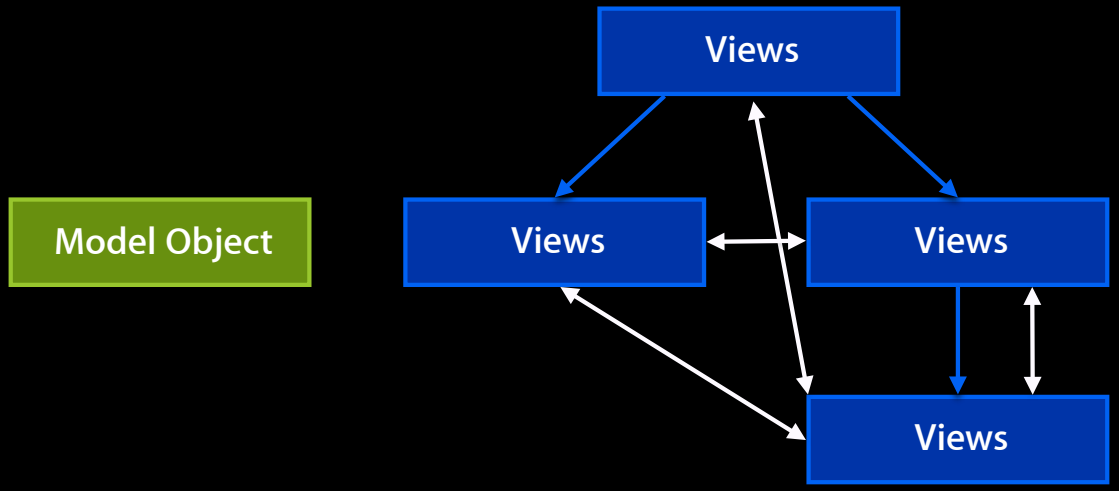
Why UIViewController?

Centralize responsibility



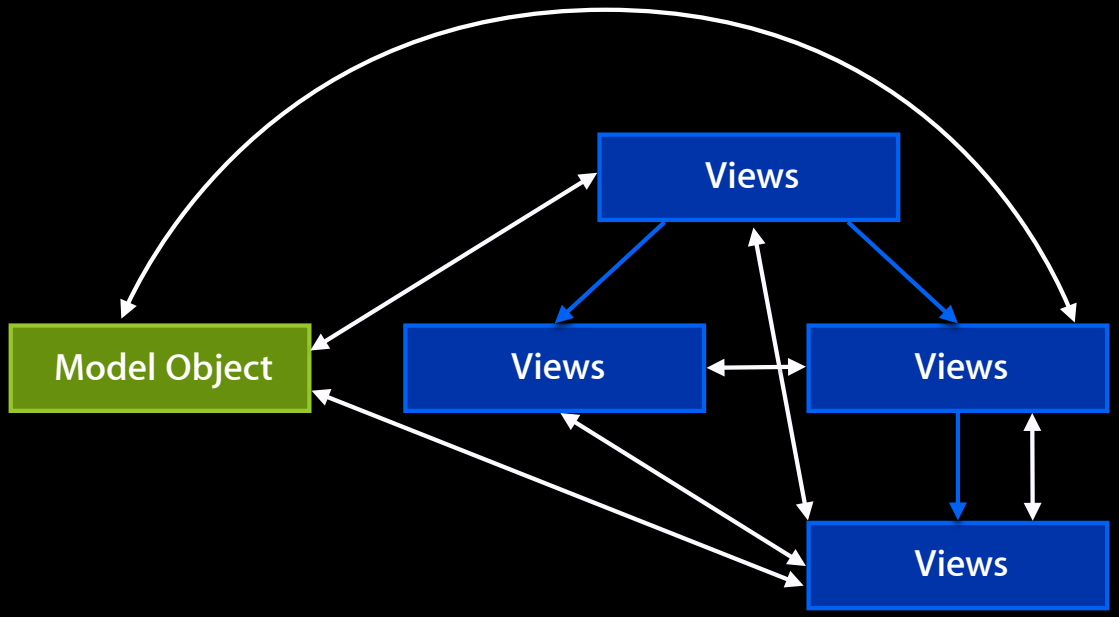
Why UIViewController?

Centralize responsibility



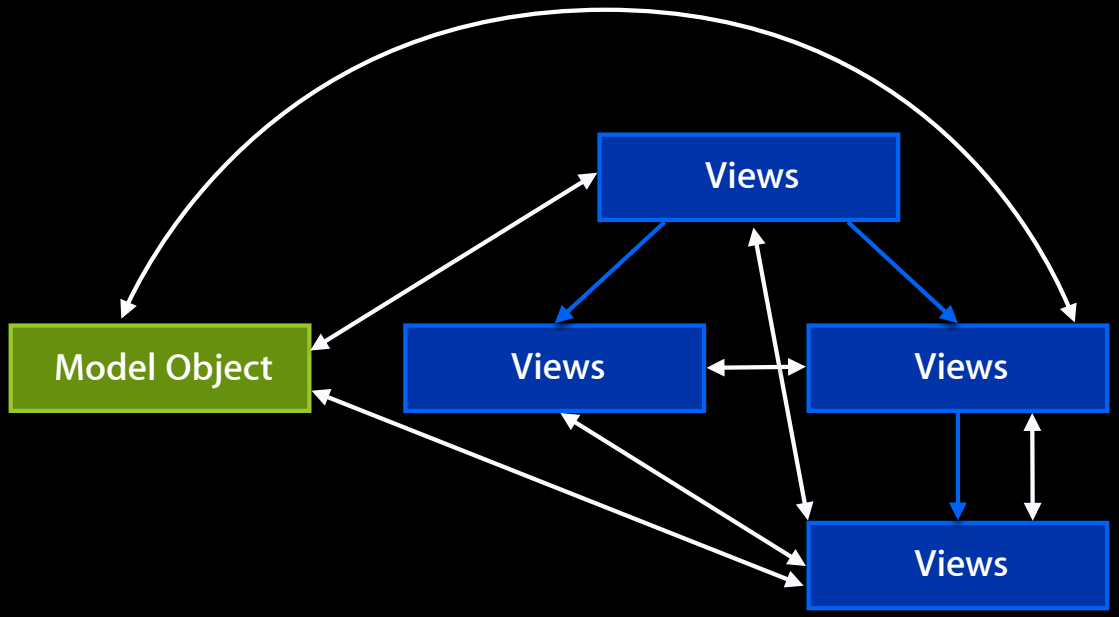
Why UIViewController?

Centralize responsibility



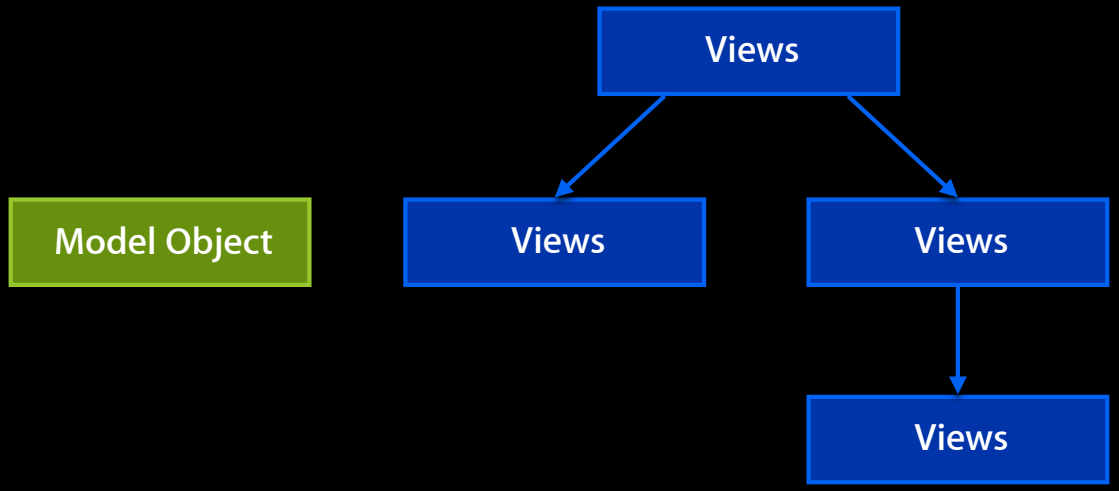
Why UIViewController?

Centralize responsibility



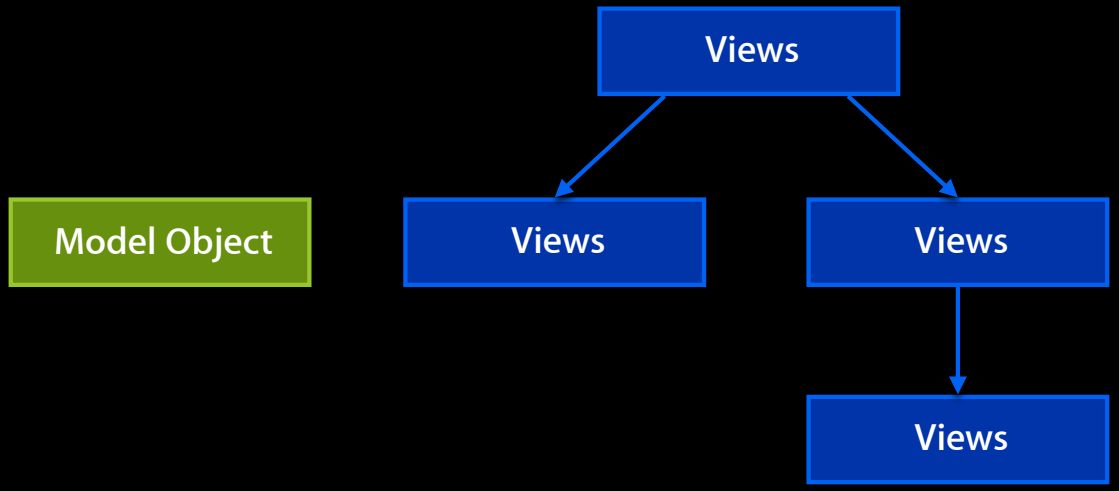
Why UIViewController?

Centralize responsibility



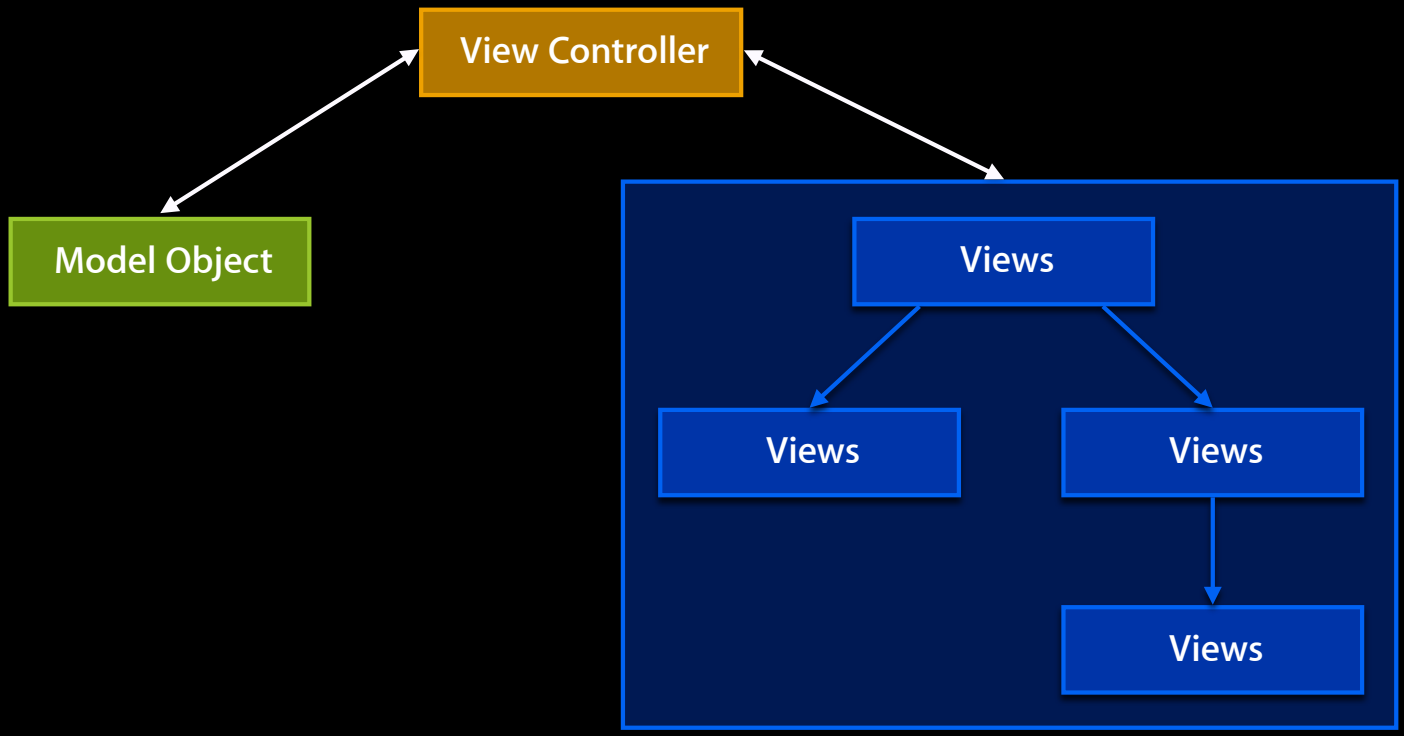
Why UIViewController?

Centralize responsibility



Why UIViewController?

Centralize responsibility



Why UIViewController?

Centralize responsibility (MVC pattern)

Why UIViewController?

Make common tasks simpler

Why UIViewController?

Make common tasks simpler

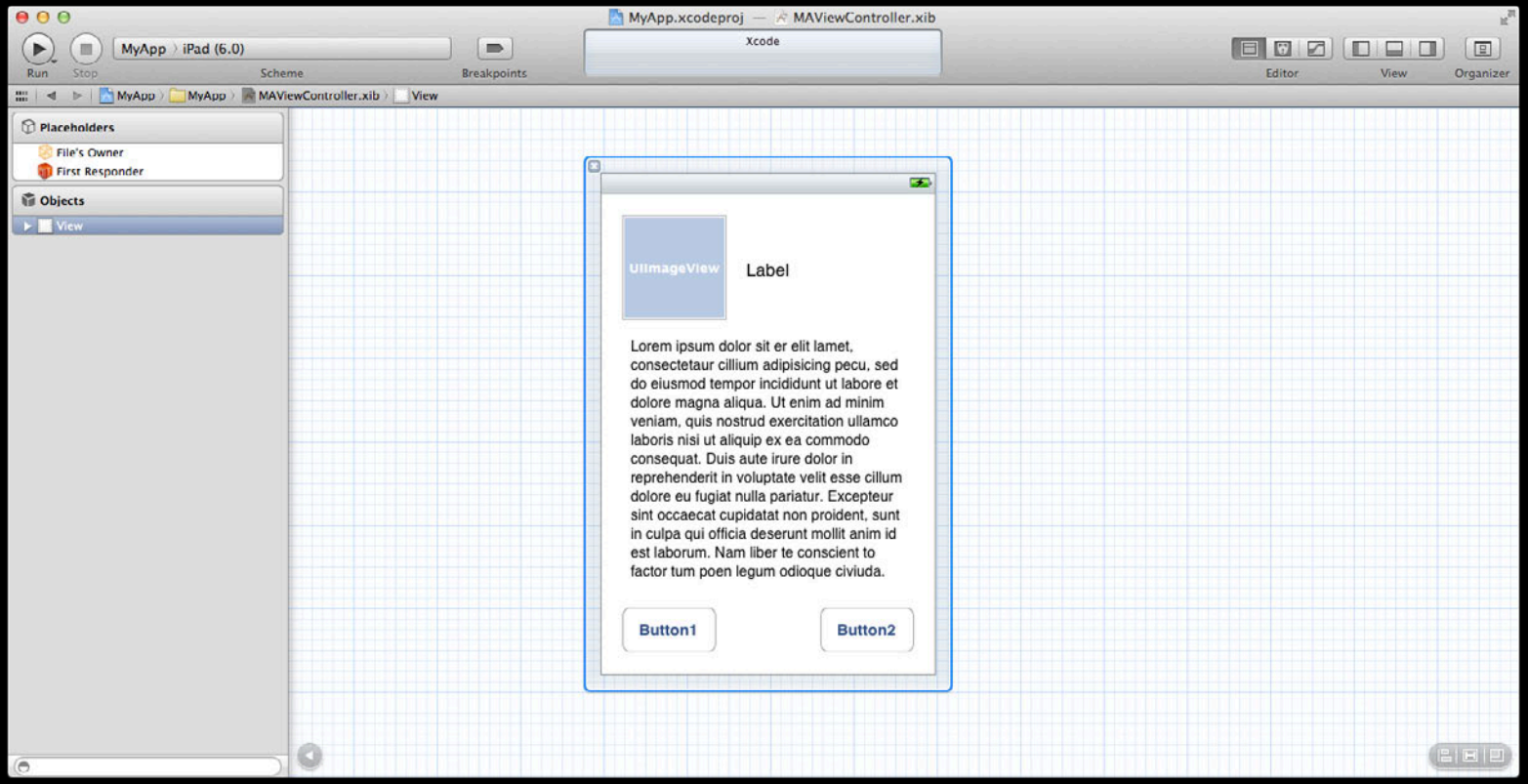
- Manage a view hierarchy
- Centralize responsibility

Why UIViewController?

Reusability

Why UIViewController?

Reusability



Why UIViewController?

Reusability

Why UIViewController?

Reusability

Why UIViewController?

Reusability

Why UIViewController?

Reusability

Why UIViewController?

Reusability

Why UIViewController?

Reusability

Why UIViewController?

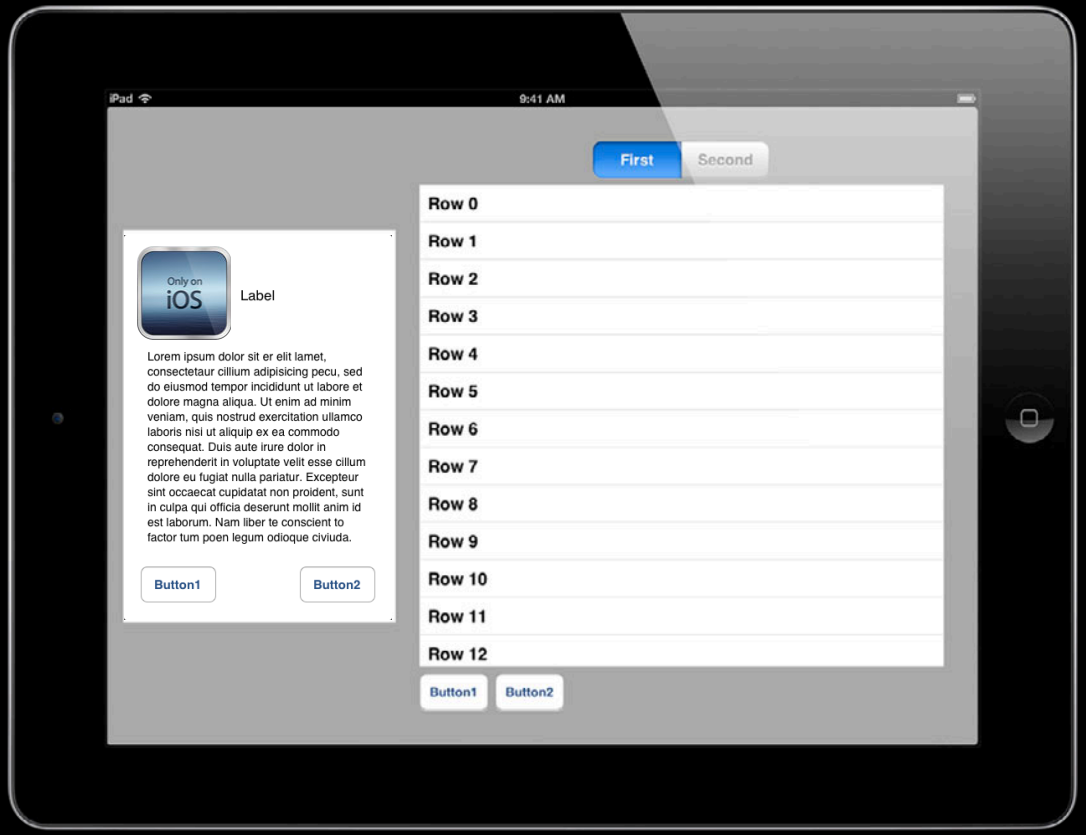
Reusability

Why UIViewController?

Reusability

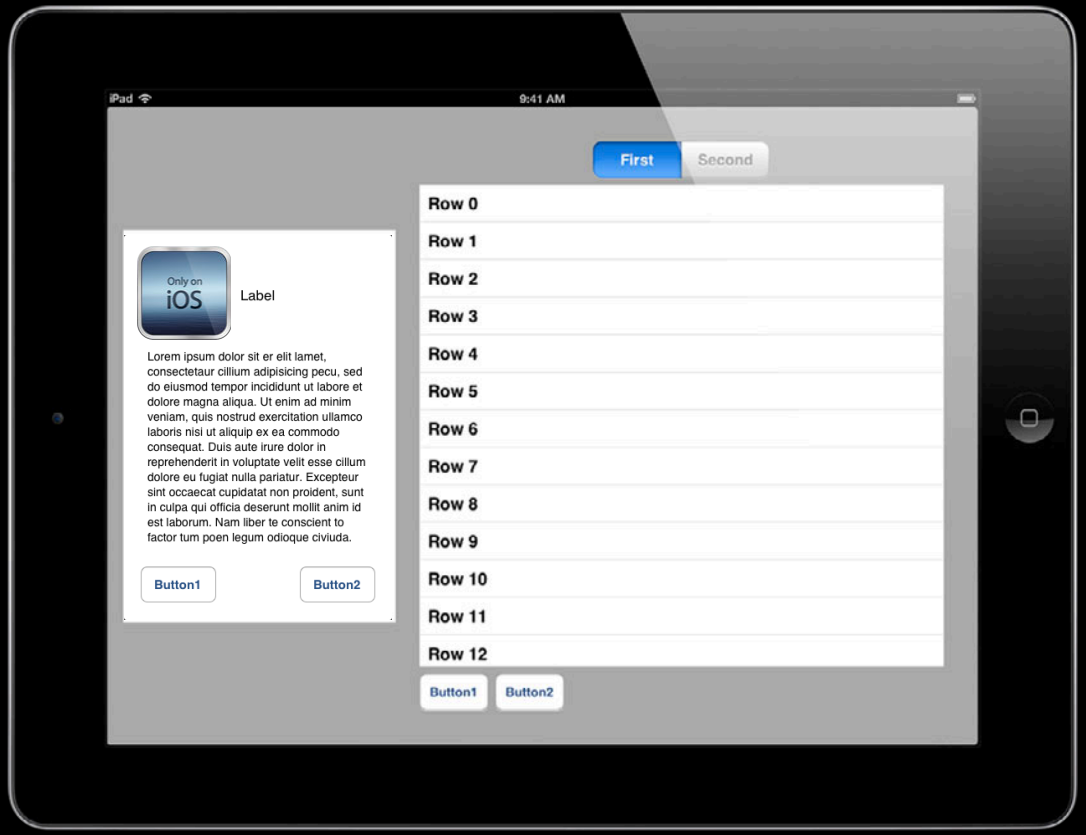
Why UIViewController?

Reusability



Why UIViewController?

Reusability



Why UIViewController?

Reusability

Why UIViewController?

Reusability—Larger logical unit

Why UIViewController?

Reusability—Larger logical unit

Why UIViewController?

Reusability—Larger logical unit

Why UIViewController?

Reusability—Larger logical unit

Why UIViewController?

Make common tasks simpler

- Manage a view hierarchy

Why UIViewController?

Make common tasks simpler

- Manage a view hierarchy
- Centralize responsibility
- Reusability—Larger logical unit

Using View Controllers Effectively

Using View Controllers Effectively

One window, one root view controller

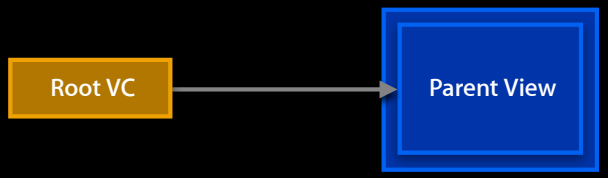
Using View Controllers Effectively

One window, one root view controller

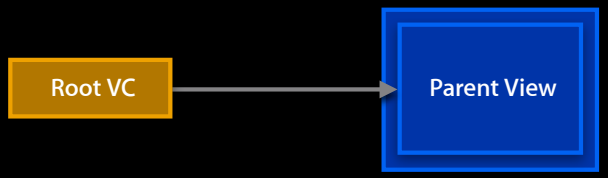
Using View Controllers Effectively

One window, one root view controller

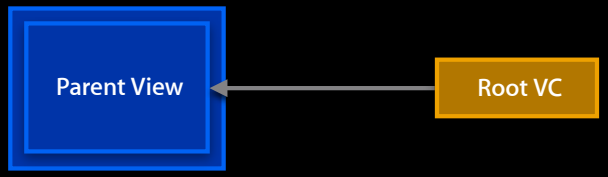
```
[window setRootViewController:rootViewController]
```



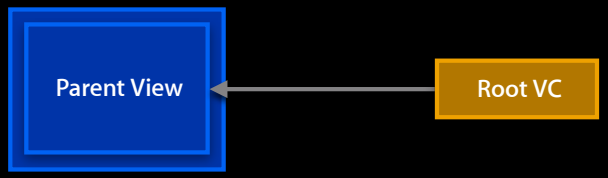
Using View Controllers Effectively



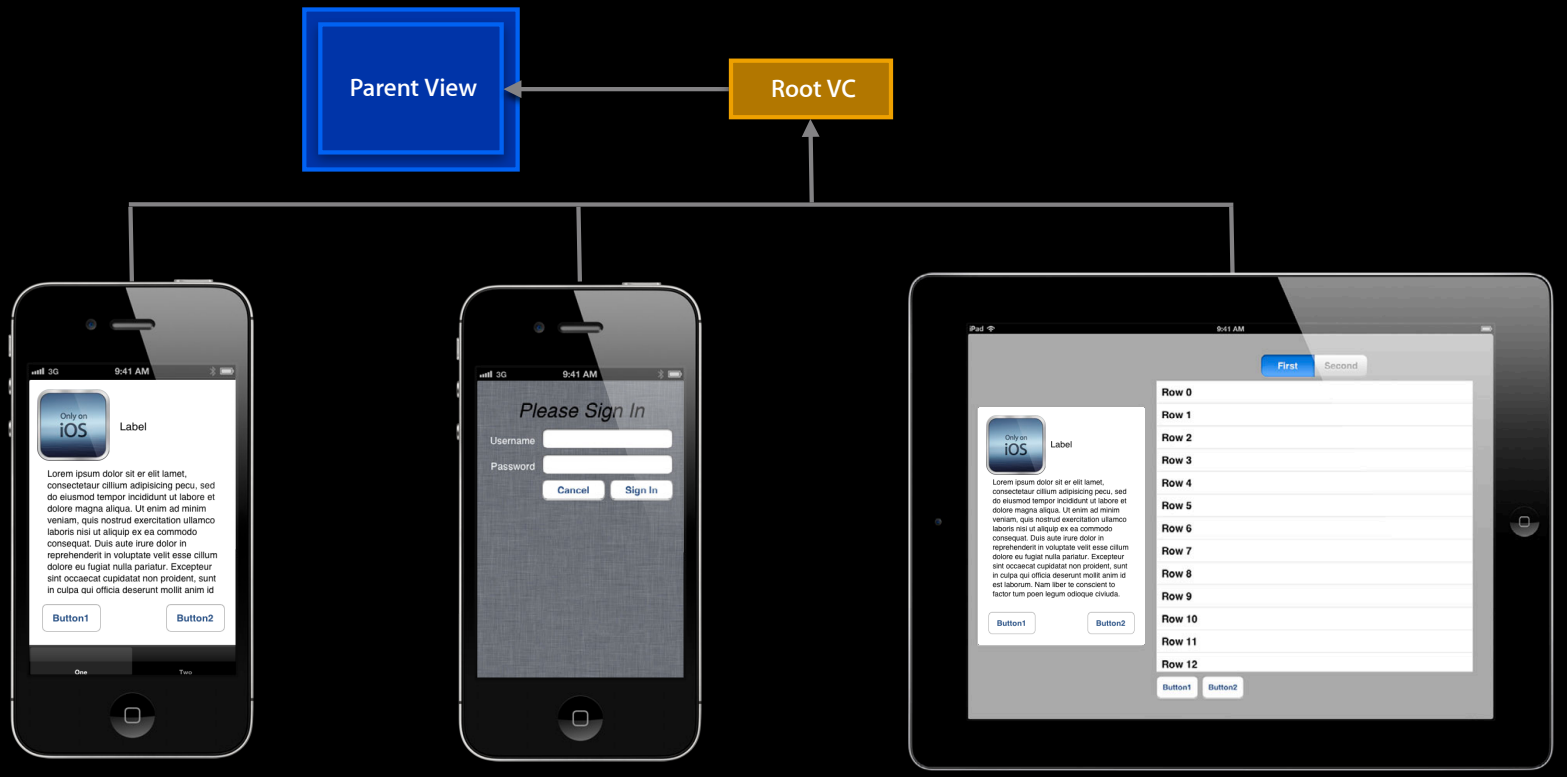
Using View Controllers Effectively



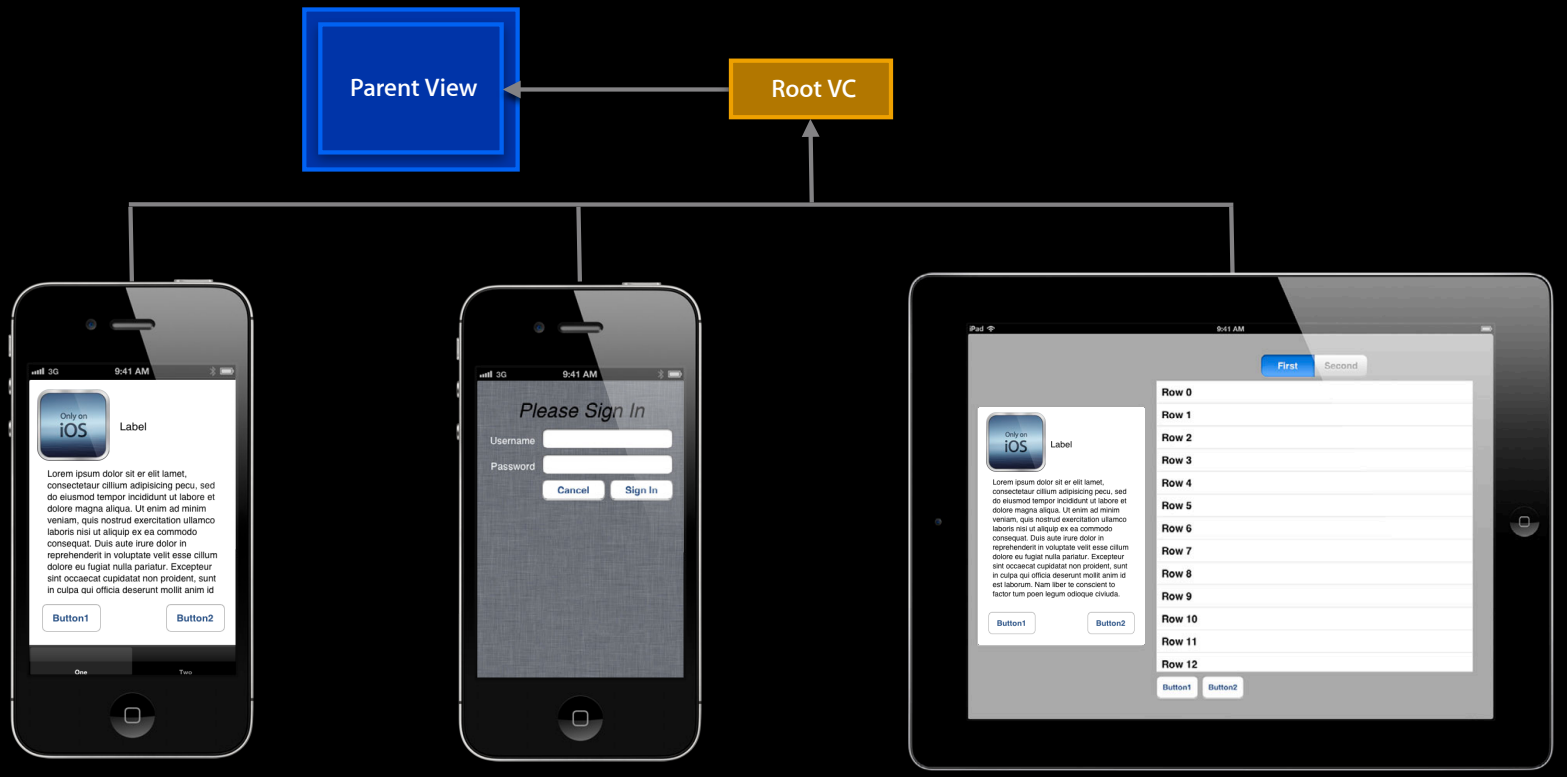
Using View Controllers Effectively



Using View Controllers Effectively



Using View Controllers Effectively



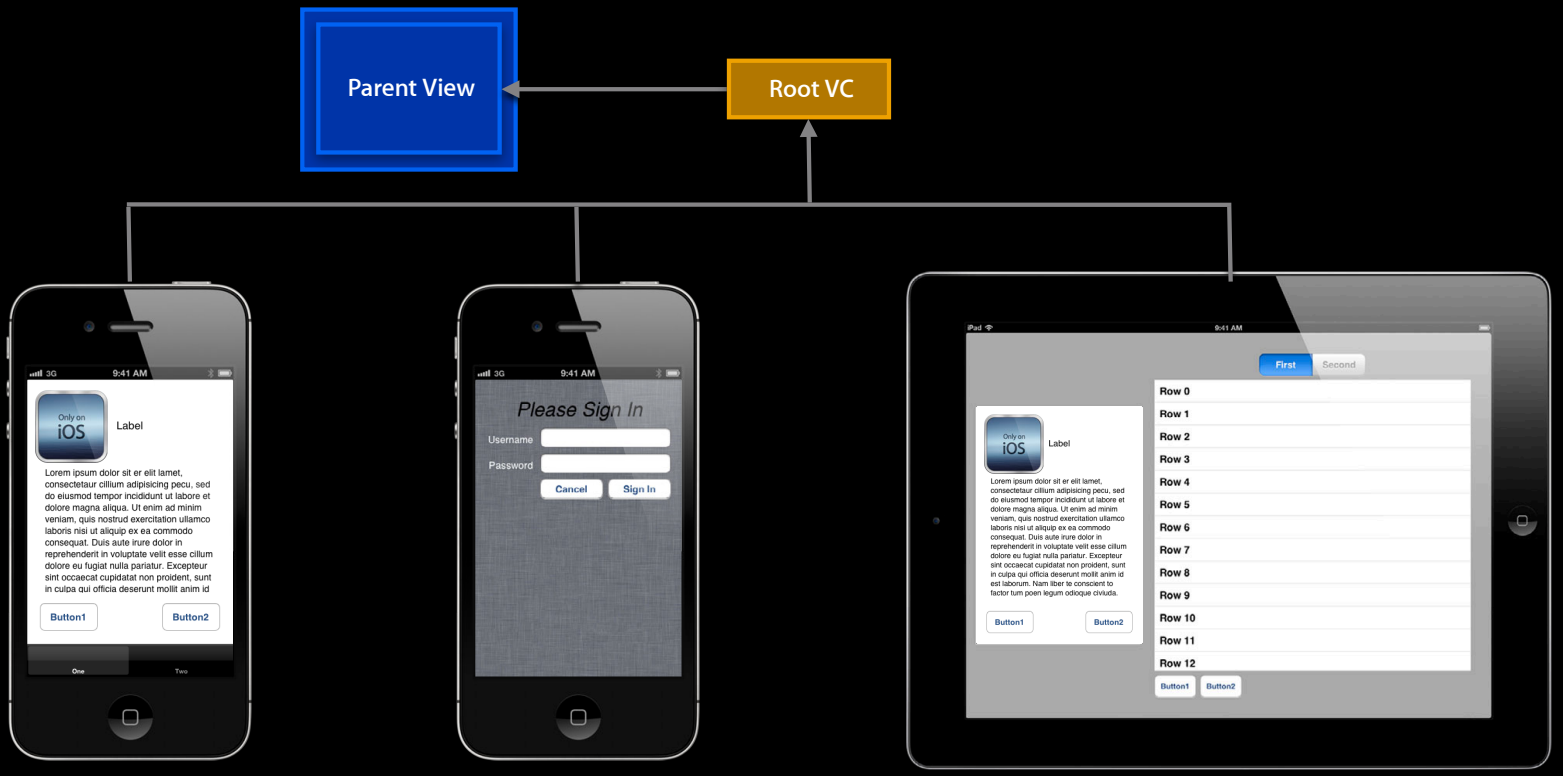
Using View Controllers Effectively

Using View Controllers Effectively

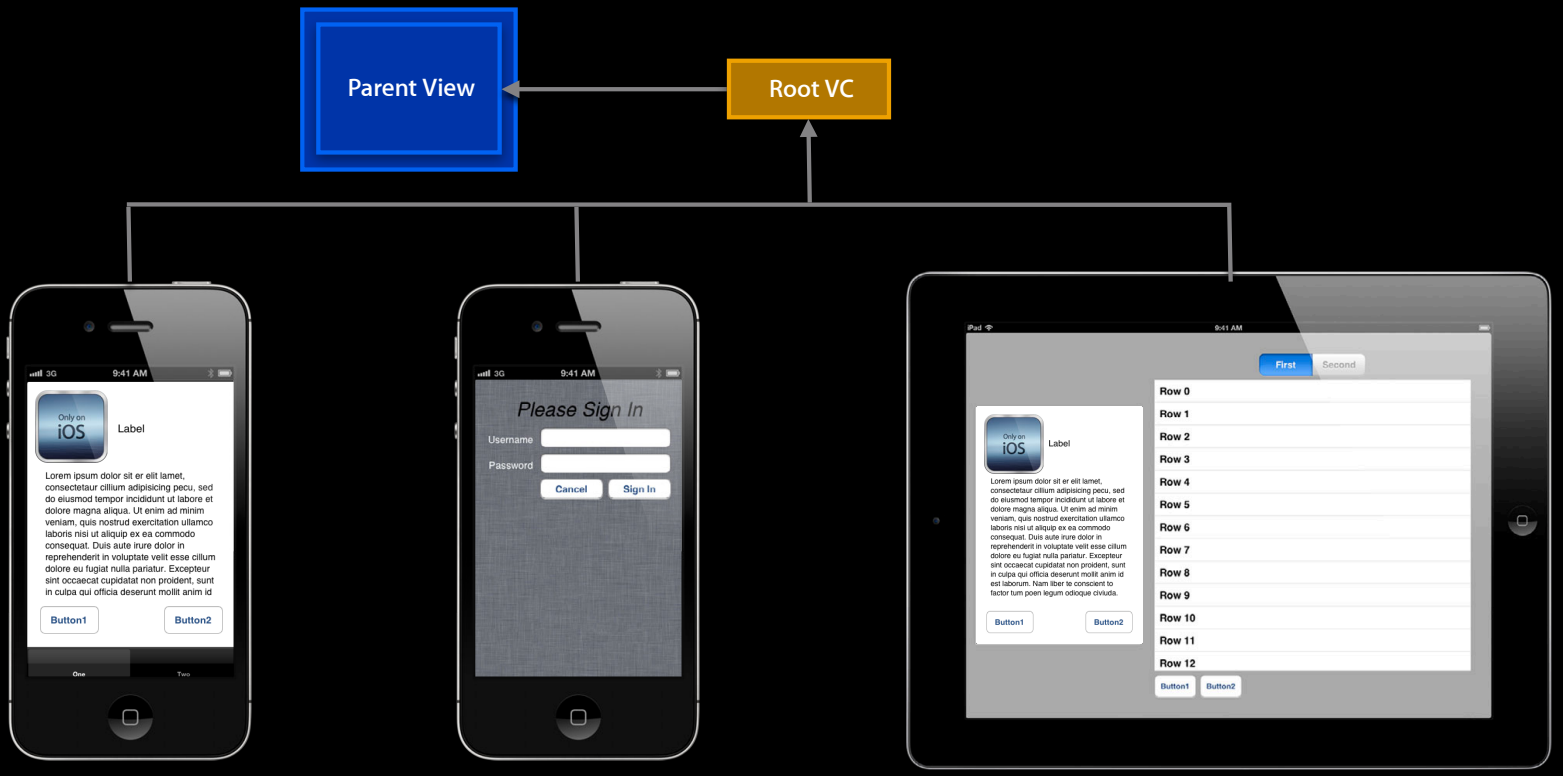
Using View Controllers Effectively

Using View Controllers Effectively

Using View Controllers Effectively



Using View Controllers Effectively



Using View Controllers Effectively

Using View Controllers Effectively

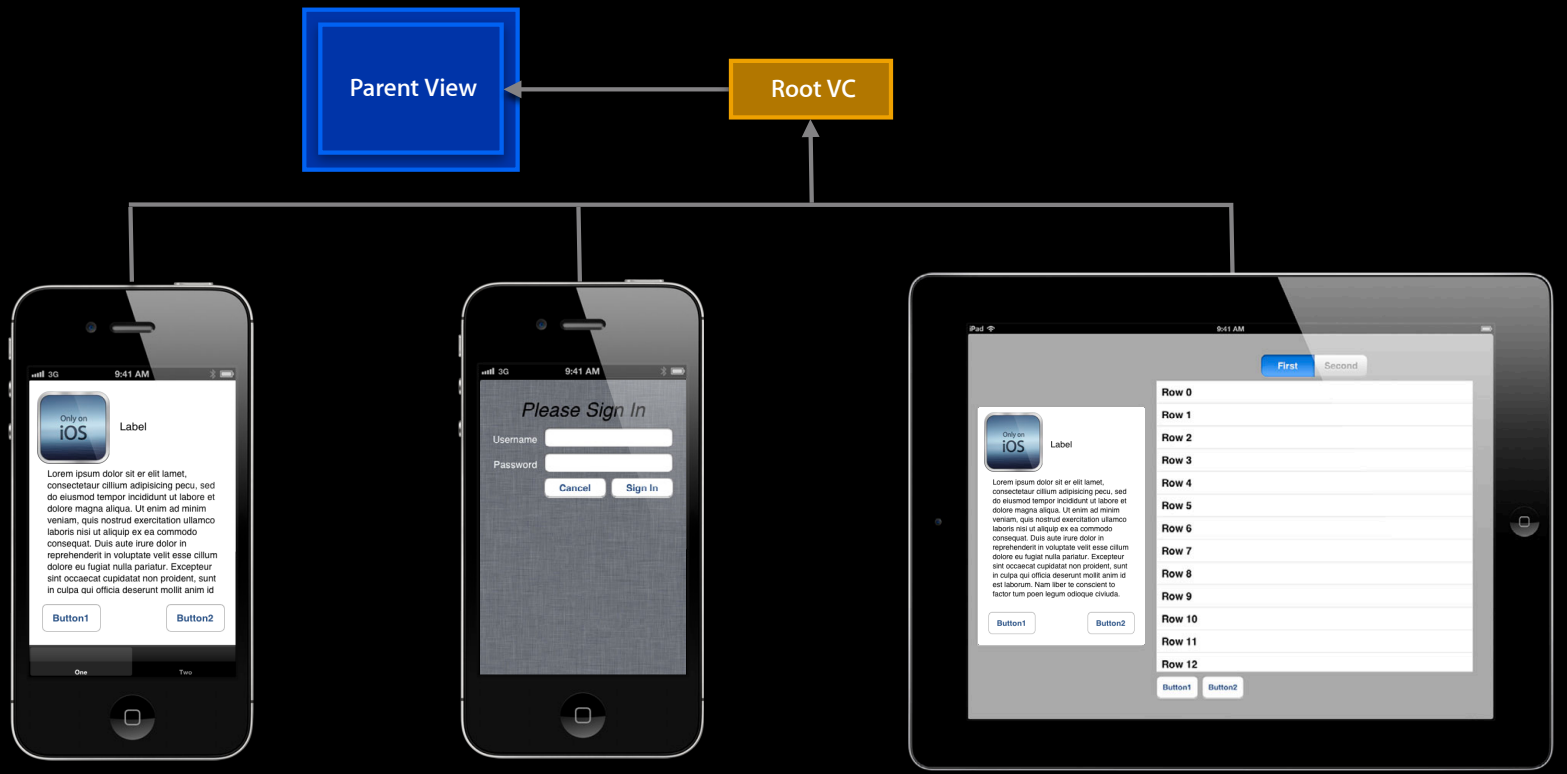
Presentation

Using View Controllers Effectively

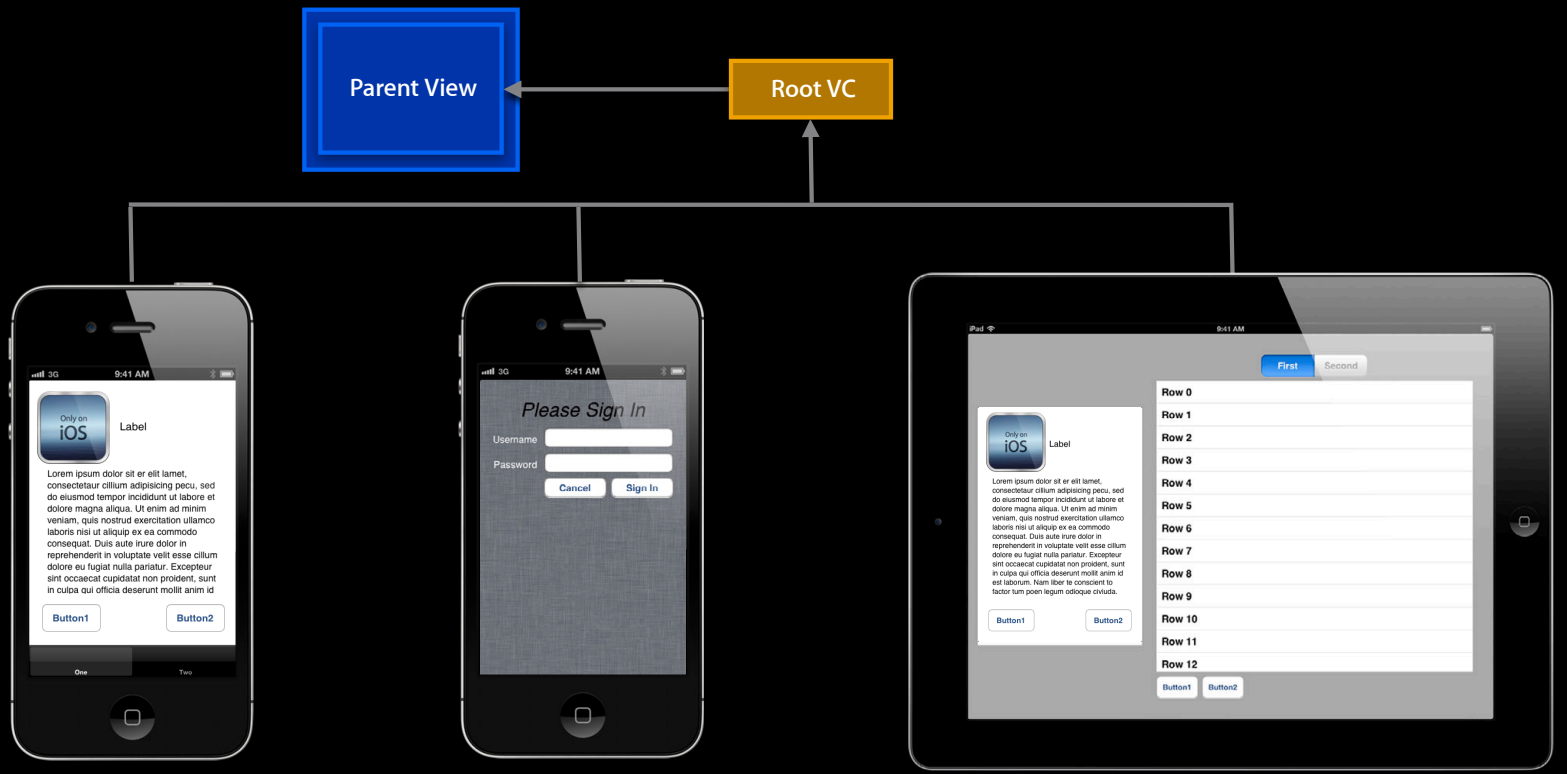
Presentation

Using View Controllers Effectively

Using View Controllers Effectively



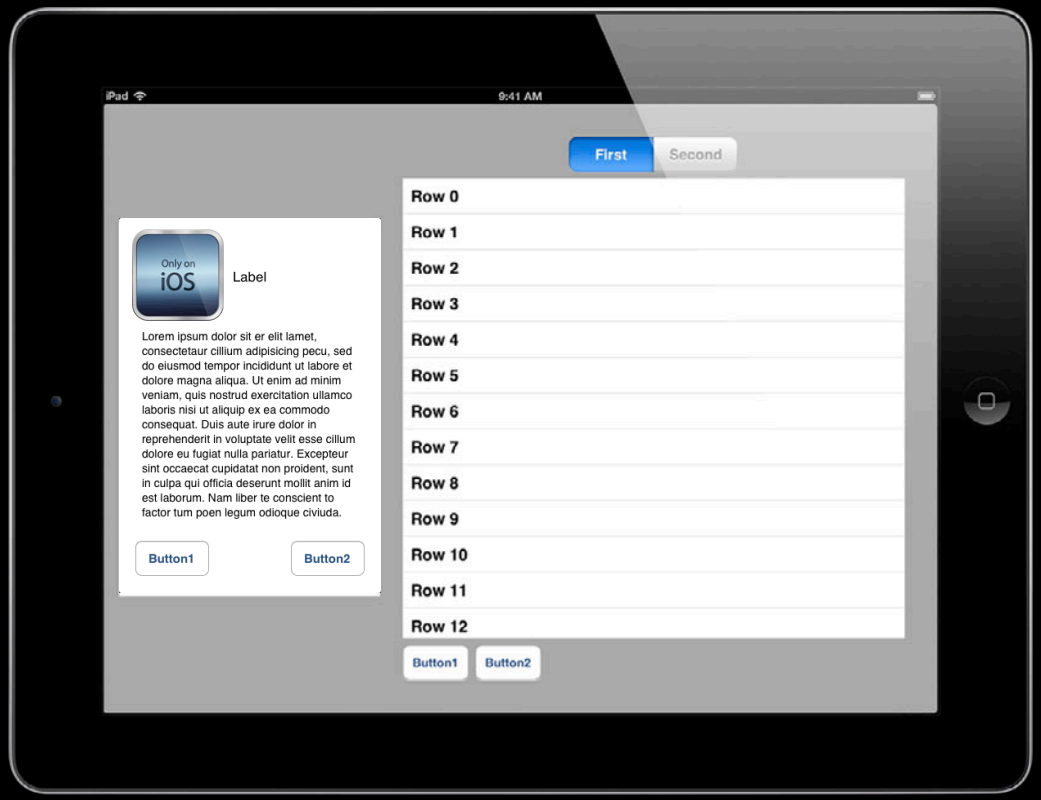
Using View Controllers Effectively



Using View Controllers Effectively

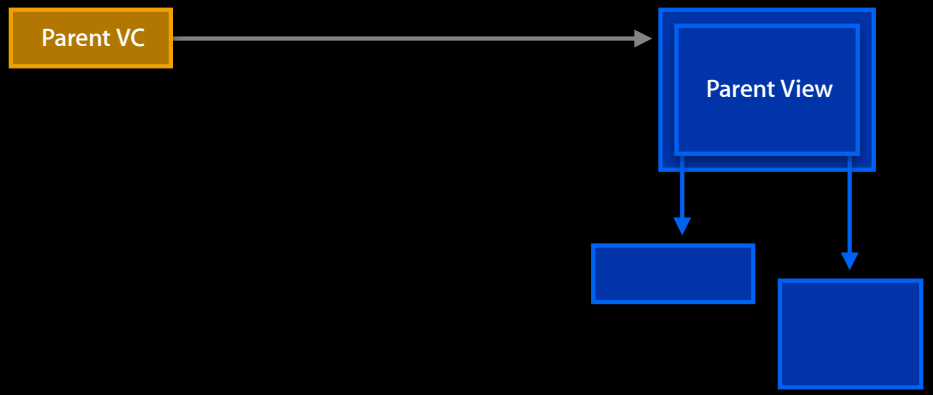
Using View Controllers Effectively

Custom container



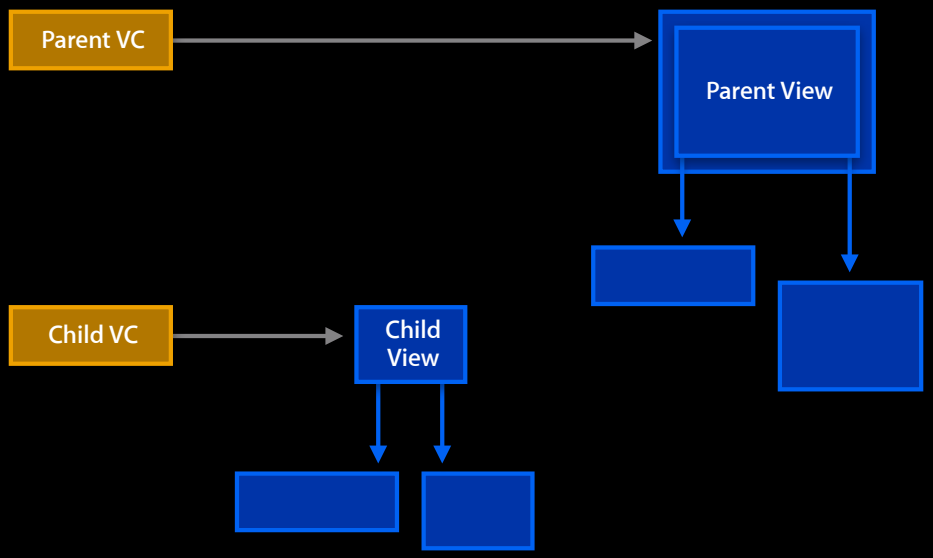
Using View Controllers Effectively

Custom container



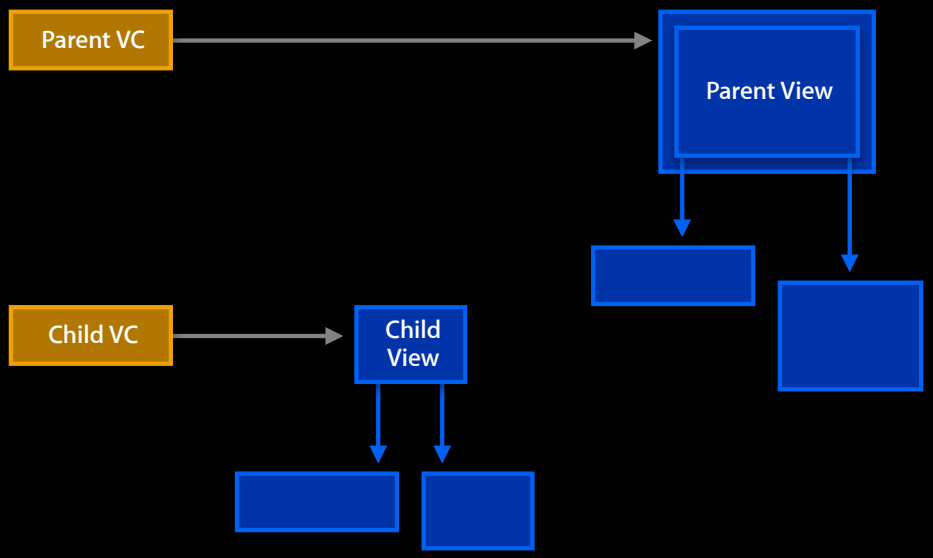
Using View Controllers Effectively

Custom container



Using View Controllers Effectively

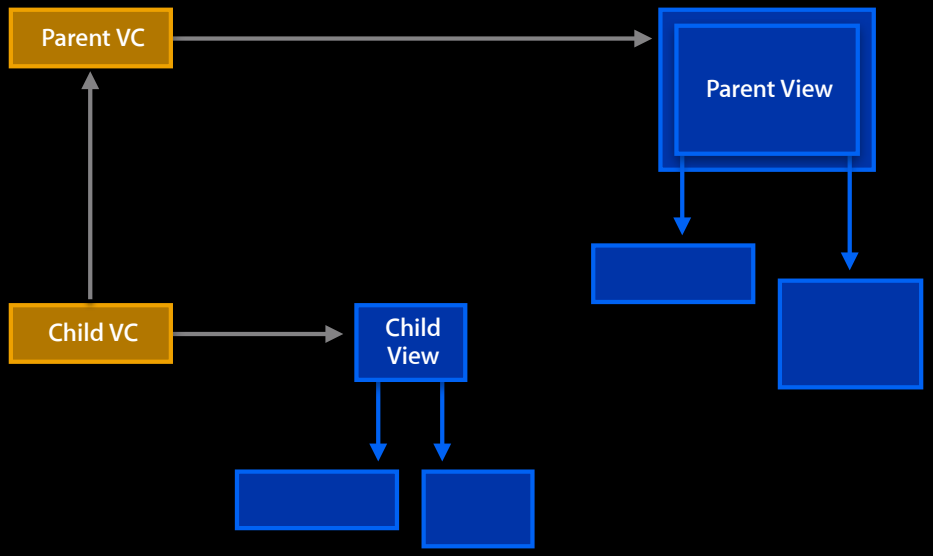
Custom container



Using View Controllers Effectively

Custom container

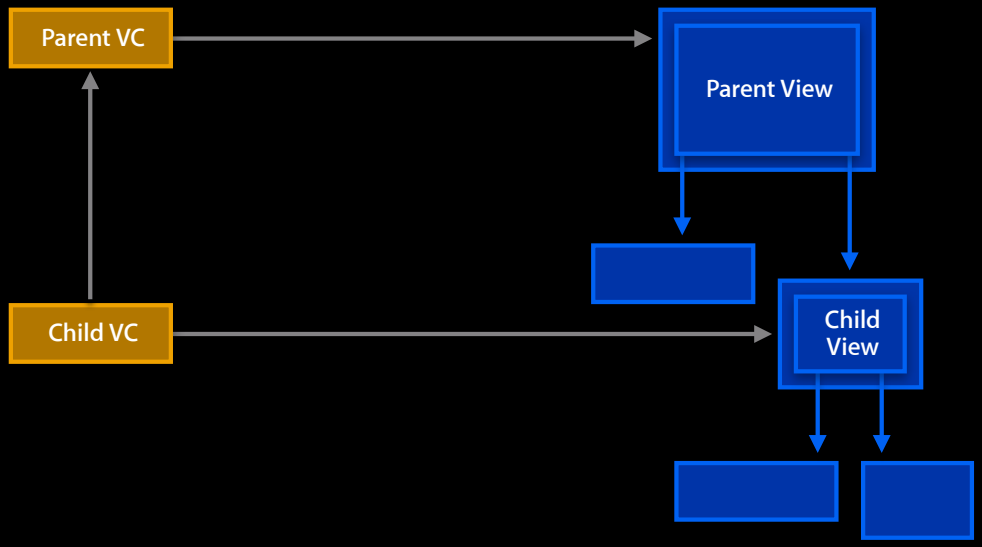
```
[parentViewController addChildViewController:childViewController]
```



Using View Controllers Effectively

Custom container

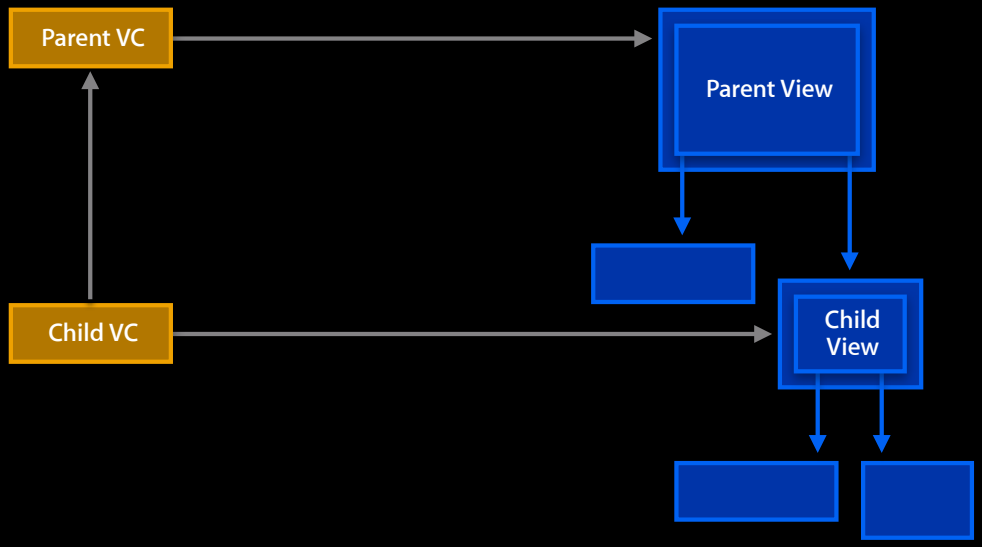
```
[parentViewController addChildViewController:childViewController]

[[parentViewController view] addSubview:[childViewController view]]
```



Using View Controllers Effectively

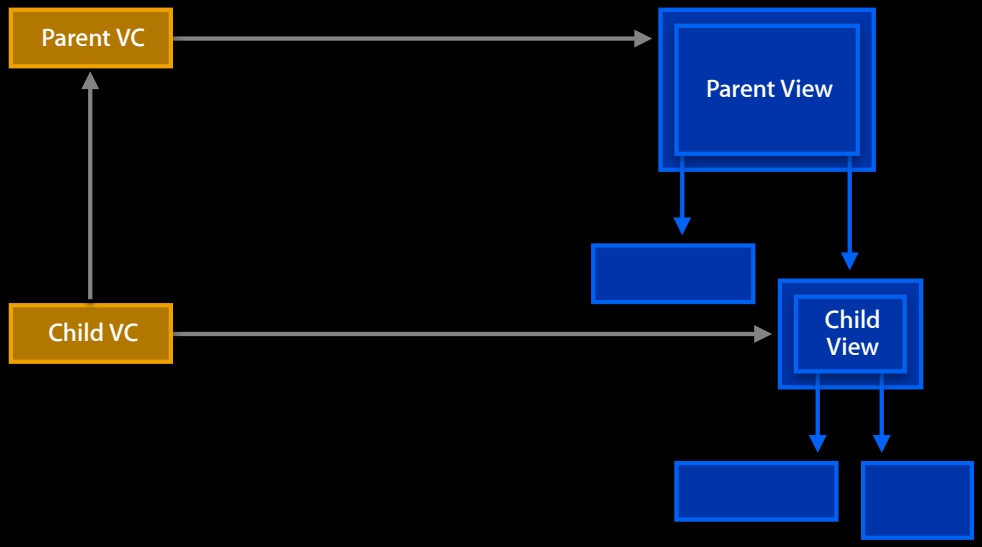
Custom container

```
[parentViewController addChildViewController:childViewController]

[[parentViewController view] addSubview:[childViewController view]]

[childViewController didMoveToParentViewController:parentViewController]
```



Using View Controllers Effectively

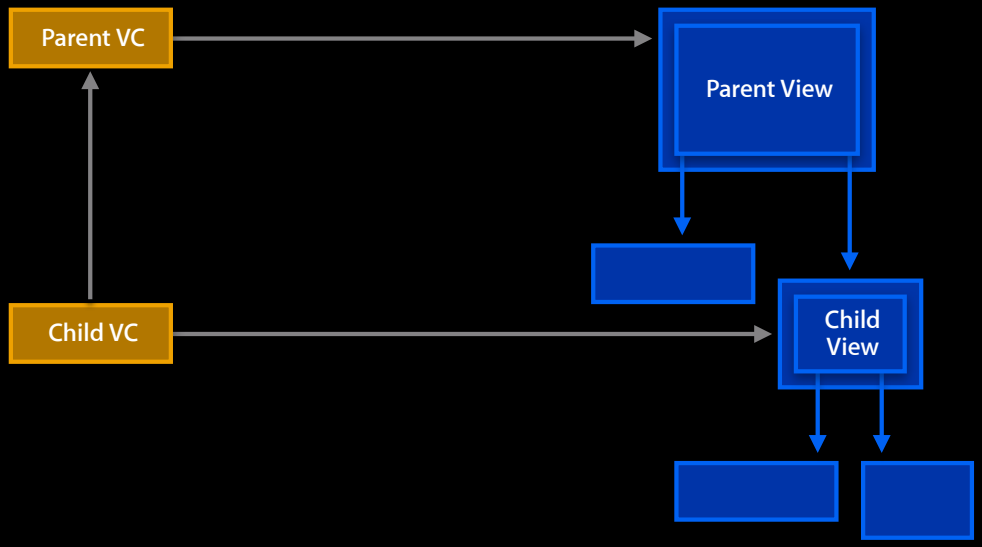
Custom container



Using View Controllers Effectively

Custom container

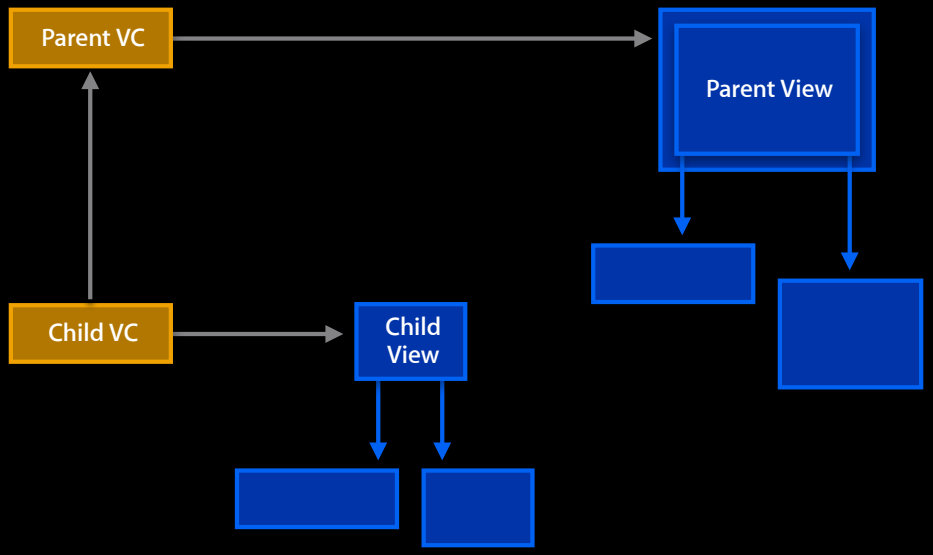
```
[childViewController willMoveToParentViewController:nil]
```



Using View Controllers Effectively

Custom container

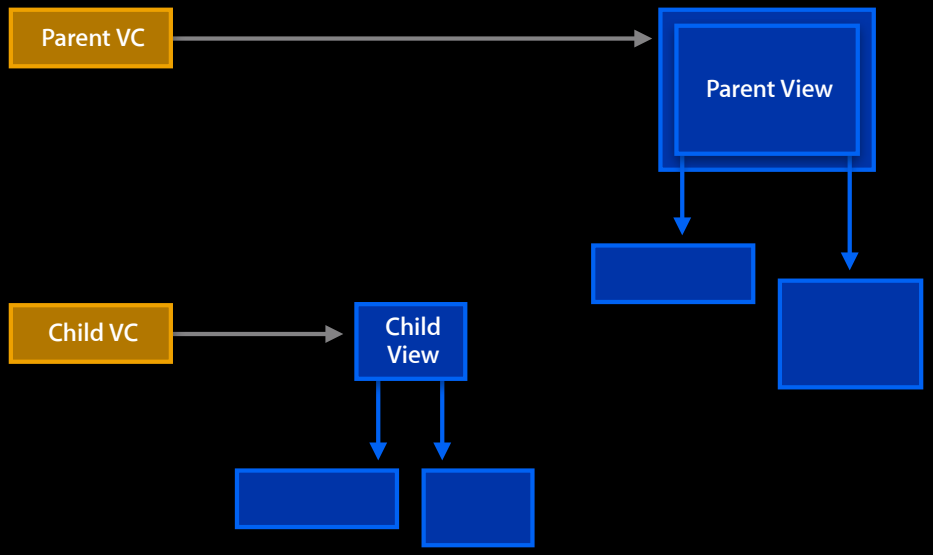
```
[childViewController willMoveToParentViewController:nil]

[[childViewController view] removeFromSuperview]
```



Using View Controllers Effectively

Custom container

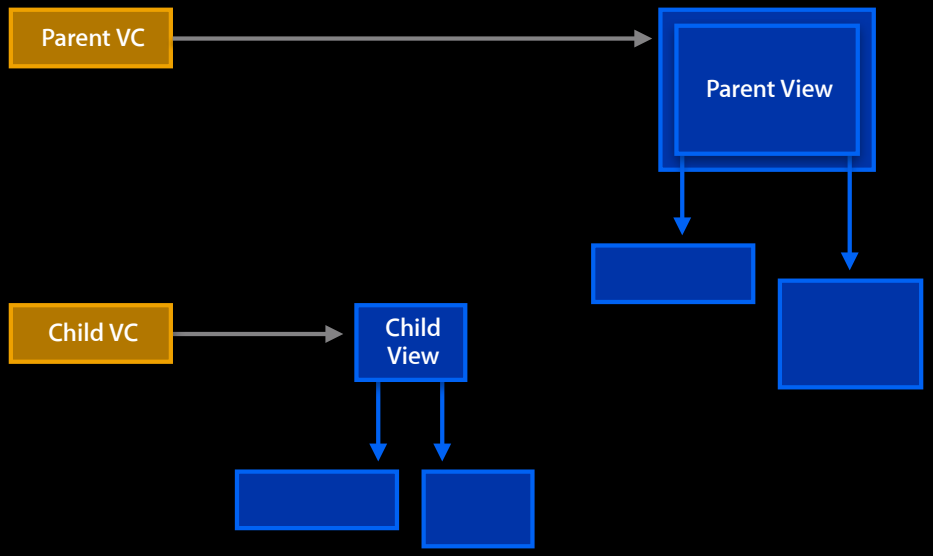
```
[childViewController willMoveToParentViewController:nil]

[[childViewController view] removeFromSuperview]

[childViewController removeFromParentViewController]
```



Using View Controllers Effectively

Custom container

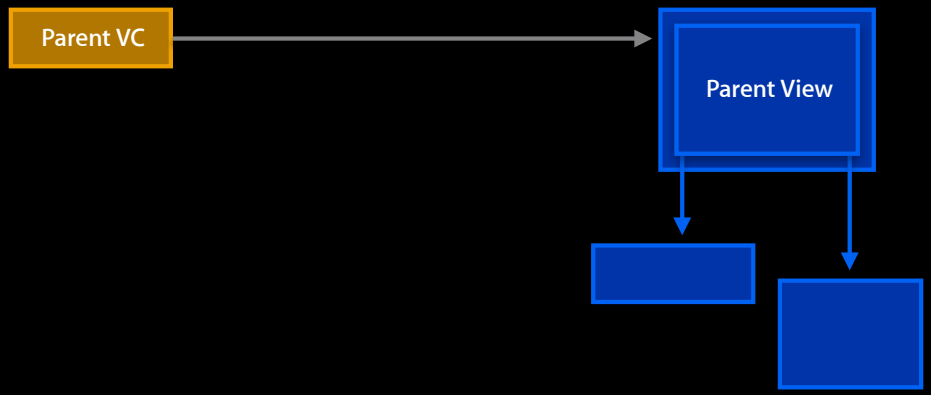
```
[childViewController willMoveToParentViewController:nil]

[[childViewController view] removeFromSuperview]

[childViewController removeFromParentViewController]
```



Using View Controllers Effectively

Custom container

```
[childViewController willMoveToParentViewController:nil]

[[childViewController view] removeFromSuperview]

[childViewController removeFromParentViewController]
```



Using View Controllers Effectively

Custom container

- Parents make the rules, children follow them

Using View Controllers Effectively

Custom container

- Parents make the rules, children follow them
  - Parents add children—not the other way!

Using View Controllers Effectively

Custom container

- Parents make the rules, children follow them
  - Parents add children—not the other way!
  - Parents manage their children's views

Summary

UIViewController FTW

- Why UIViewController?
  - Manage a view hierarchy
  - Centralize responsibility
  - Reusability—larger logical unit
- Using view controllers effectively
  - One window, one root view controller
  - Build consistent view controller hierarchies

View Controllers Today

New directions

Bruce D. Nilo
View Controller Mechanic

View Controllers Today

Roadmap

View Controllers Today

Roadmap

- Discuss new and deprecated API and behaviors

View Controllers Today

Roadmap

- Discuss new and deprecated API and behaviors
  - View controller containment

View Controllers Today

Roadmap

- Discuss new and deprecated API and behaviors
  - View controller containment
  - Autorotation

View Controllers Today

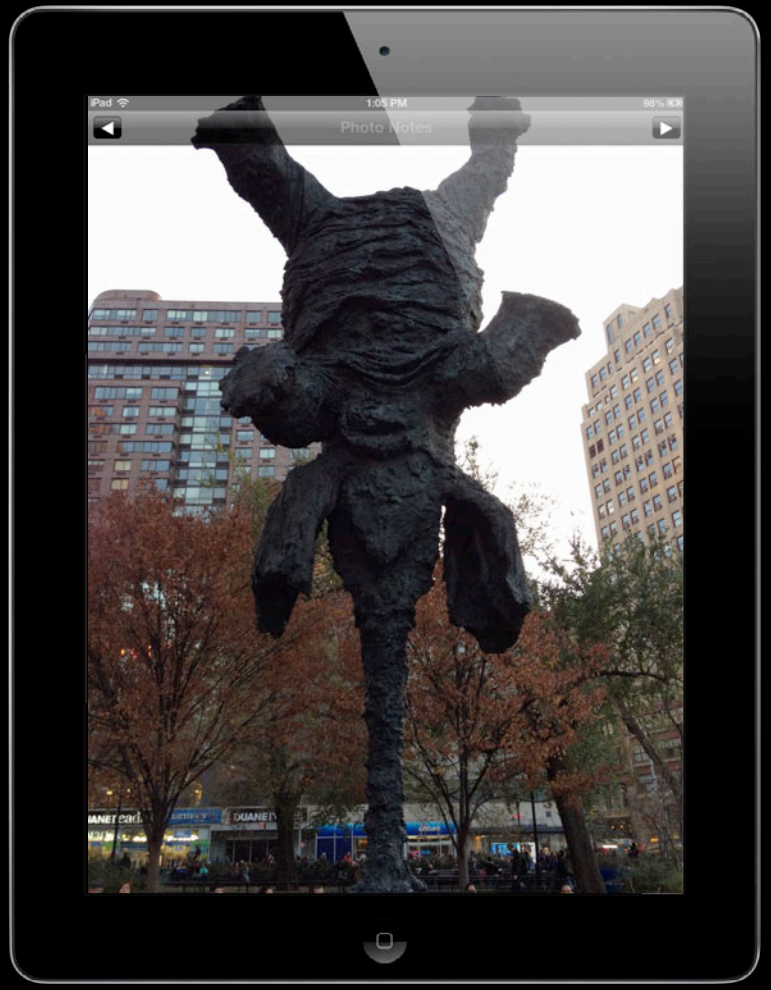
Roadmap

- Discuss new and deprecated API and behaviors
  - View controller containment
  - Autorotation
  - Other stuff

View Controllers Today

Roadmap

- PhotoNotes



View Controllers Today

Roadmap

- PhotoNotes
  - An app with a custom application flow

View Controllers Today

Roadmap

- PhotoNotes
- An app with a custom application flow
  - Best containment practices

View Controllers Today

Roadmap

- PhotoNotes
- An app with a custom application flow
  - Best containment practices
  - How to adopt the new autorotation behavior

View Controllers Today

Roadmap

- PhotoNotes
- An app with a custom application flow
  - Best containment practices
  - How to adopt the new autorotation behavior
  - How to ensure layout is independent of interface orientation

View Controllers Today

Roadmap

- PhotoNotes
- An app with a custom application flow
  - Best containment practices
  - How to adopt the new autorotation behavior
  - How to ensure layout is independent of interface orientation
  - Keyboard avoidance and more

View Controllers Today

Evolution

View Controller Evolution

A primary objective

View Controller Evolution

A primary objective

- View controllers should compose consistently with each other

View Controller Evolution

A primary objective

- View controllers should compose consistently with each other
  - New device types

View Controller Evolution

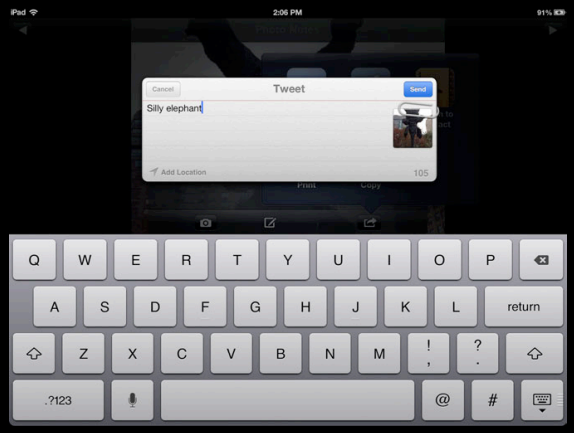
A primary objective

- View controllers should compose consistently with each other
  - New device types
  - Many view controllers on the screen at once

View Controller Evolution

A primary objective

- View controllers should compose consistently with each other
  - New device types
  - Many view controllers on the screen at once
  - Many new system view controllers are available



View Controllers Today

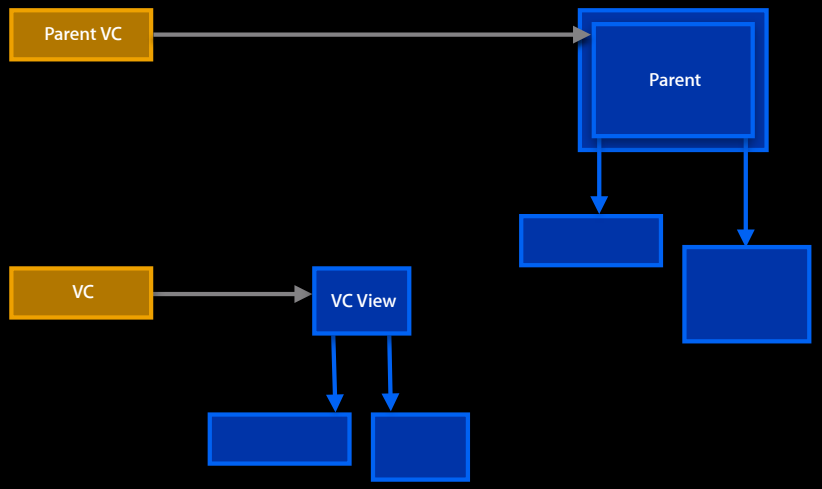
Containment API and behavioral changes

View Controller Evolution

Containment (embedded view controllers)

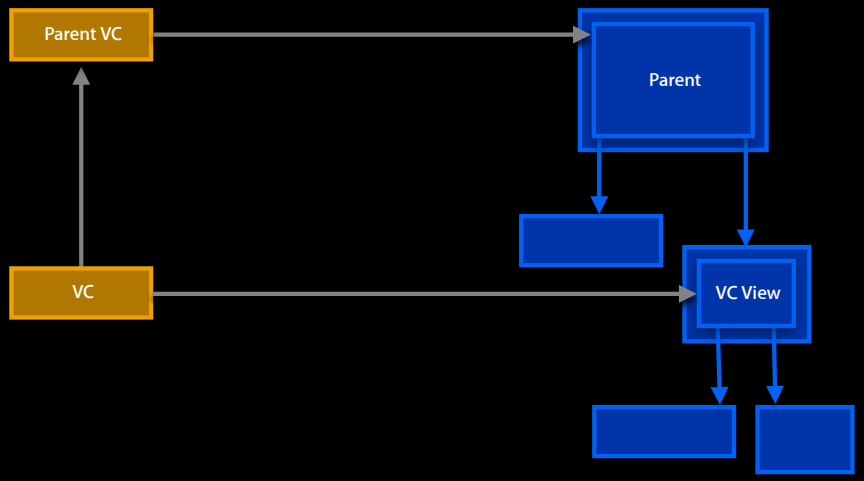
View Controller Evolution

Containment (embedded view controllers)



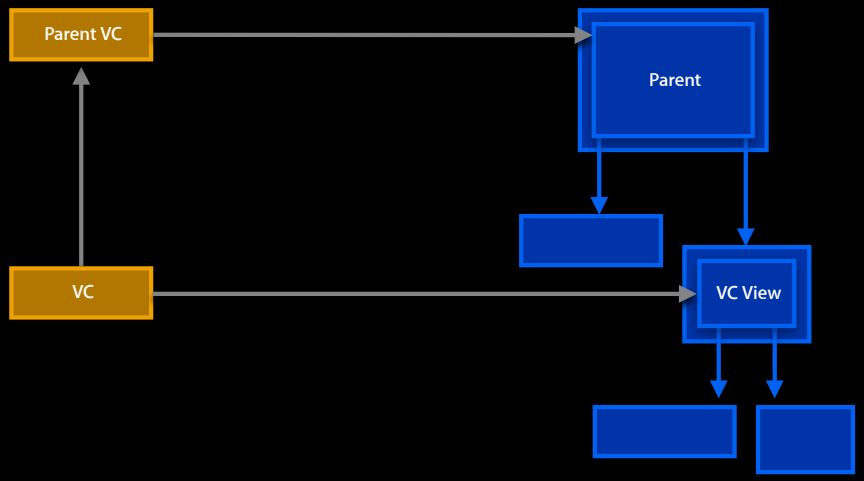
View Controller Evolution

Containment (embedded view controllers)



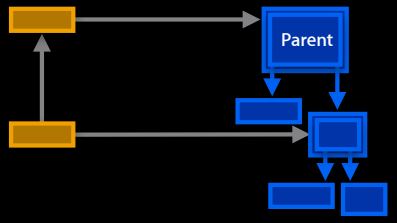
View Controller Evolution

Containment (embedded view controllers)



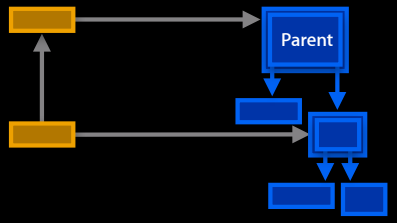
View Controller Evolution

Containment (embedded view controllers)



View Controller Evolution

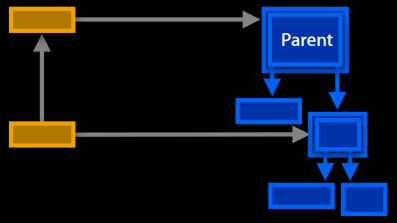
Containment (embedded view controllers)




```
- (BOOL)shouldAutomaticallyForwardAppearanceMethods
{
   return NO; // Override default which is YES
}
```

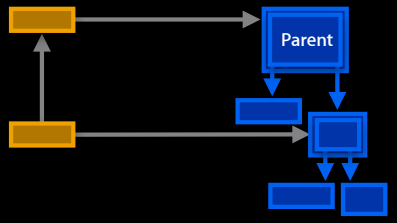
View Controller Evolution

Containment (embedded view controllers)



View Controller Evolution

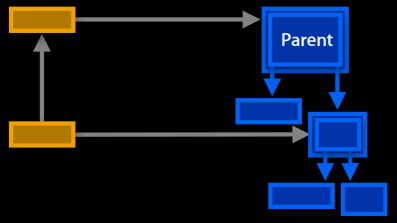
Containment (embedded view controllers)




```
- (void)revealChild:(UIViewController *)child
{
  [self addChildViewController:self.child];
  [child beginAppearanceTransition: YES
                          animated: YES];
  // [self.view addSubview:child.view]

  [UIView animateWithDuration:.5
    animations: ^{[self adjustFrameForChild:child]; }
    completion:^(BOOL finished) {
        [child endAppearanceTransition:
        [child didMoveToParentViewController:self];
     }];
}
```

View Controller Evolution

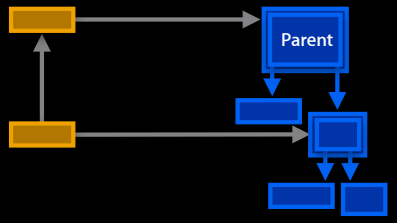
Containment (embedded view controllers)




```
- (void)revealChild:(UIViewController *)child
{
  [self addChildViewController:self.child];
  [child beginAppearanceTransition: YES
                 animated: YES];
  // [self.view addSubview:child.view]

  [UIView animateWithDuration:.5
    animations: ^{[self adjustFrameForChild:child]; }
    completion:^(BOOL finished) {
         [child endAppearanceTransition:
         [child didMoveToParentViewController:self];
     }];
}
```

View Controller Evolution

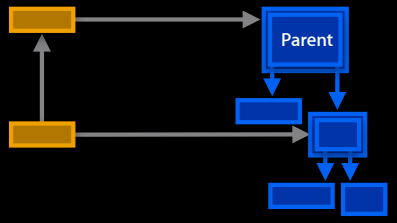
Containment (embedded view controllers)




```
- (void)revealChild:(UIViewController *)child
{
  [self addChildViewController:self.child];
  [child beginAppearanceTransition: YES
         animated: YES];
  // [self.view addSubview:child.view]

  [UIView animateWithDuration:.5
    animations: ^{[self adjustFrameForChild:child]; }
    completion:^(BOOL finished) {
         [child endAppearanceTransition:
         [child didMoveToParentViewController:self];
     }];
}
```

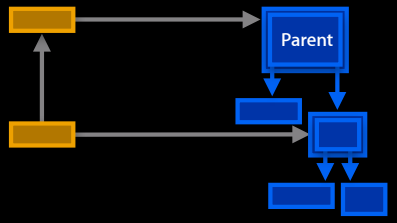
View Controller Evolution

Containment (embedded view controllers)



View Controller Evolution

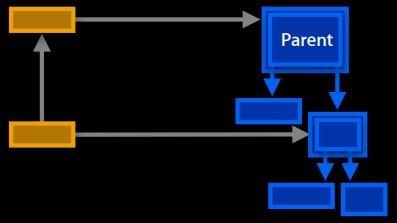
Containment (embedded view controllers)




```
- (void)viewWillAppear:(BOOL)animated {
  [self.child beginAppearanceTransition: YES
                 animated: animated];
}

- (void)viewDidAppear:(BOOL)animated {
  [self.child endAppearanceTransition];
}
```

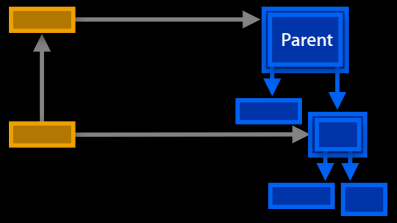
View Controller Evolution

Containment—Best practices



View Controller Evolution

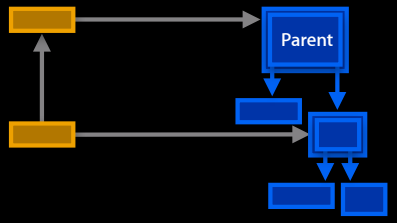
Containment—Best practices



- The container should expose methods that use the containment API
  - `addChildViewController:`
  - `removeFromParentViewController`

View Controller Evolution

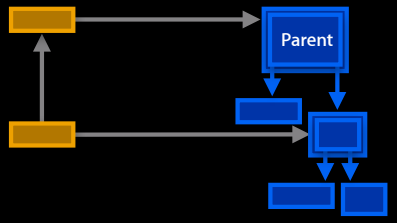
Containment—Best practices




```
- (void)revealChild:(UIViewController *)child
{
  [self addChildViewController:self.child];
  [child beginAppearanceTransition: YES
        animated: YES];
  [UIView animateWithDuration:.5
    animations: ^{[self adjustFrameForChild:child]; }
    completion:^(BOOL finished) {
         [child endAppearanceTransition:
         [child didMoveToParentViewController:self];
     }];
}
```

View Controller Evolution

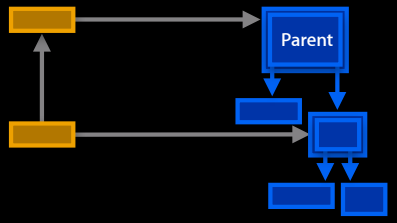
Containment—Best practices




```
- (void)revealChild:(UIViewController *)child
{
   [self addChildViewController:self.child];
   [child beginAppearanceTransition: YES
                           animated: YES];
   [UIView animateWithDuration:.5
     animations: ^{[self adjustFrameForChild:child]; }
     completion:^(BOOL finished) {
          [child endAppearanceTransition:
          [child didMoveToParentViewController:self];
      }];
}
```

View Controller Evolution

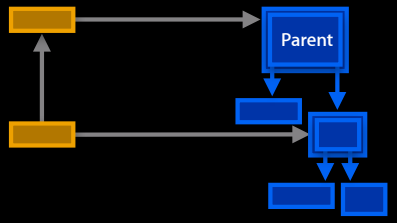
Containment—Best practices




```
- (void)revealChild:(UIViewController *)child
{
   [self addChildViewController:self.child];
   [child beginAppearanceTransition: YES
                  animated: YES];
   [UIView animateWithDuration:.5
     animations: ^{[self adjustFrameForChild:child]; }
     completion:^(BOOL finished) {
          [child endAppearanceTransition:
          [child didMoveToParentViewController:self];
      }];
}
```

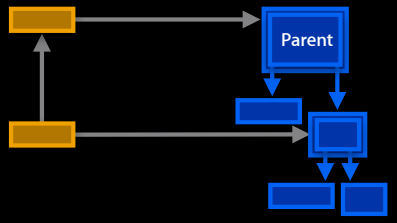
View Controller Evolution

Containment—Best practices



View Controller Evolution

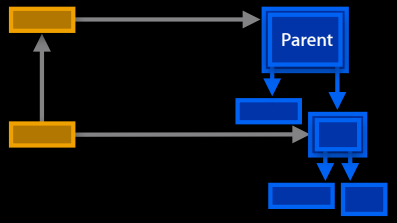
Containment—Best practices



- The parent is responsible for the frames of its children

View Controller Evolution

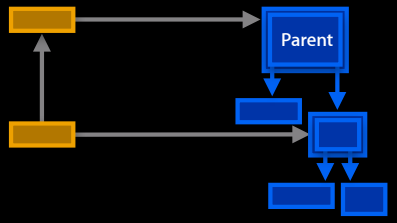
Containment—Best practices



- The parent is responsible for the frames of its children
- The child accesses its bounds in
  - `viewWillLayoutSubviews`
  - `updateViewConstraints`

View Controller Evolution

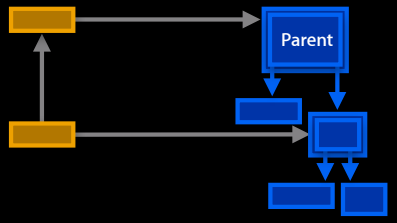
Containment (embedded view controllers)




```
- (void)revealChild:(UIViewController *)child
{
  [self addChildViewController:self.child];
  [child beginAppearanceTransition: YES
        animated: YES];
  [UIView animateWithDuration:.5
    animations: ^{[self adjustFrameForChild:child]; }
    completion:^(BOOL finished) {
         [child endAppearanceTransition:
         [child didMoveToParentViewController:self];
     }];
}
```

View Controller Evolution

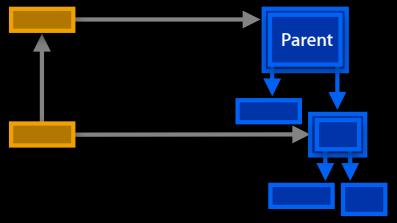
Containment (embedded view controllers)




```
- (void)revealChild:(UIViewController *)child
{
  [self addChildViewController:self.child];
  [child beginAppearanceTransition: YES
        animated: YES];
  [UIView animateWithDuration:.5
    animations: ^{[self adjustFrameForChild:child]; }
    completion:^(BOOL finished) {
         [child endAppearanceTransition:
         [child didMoveToParentViewController:self];
     }];
}
```

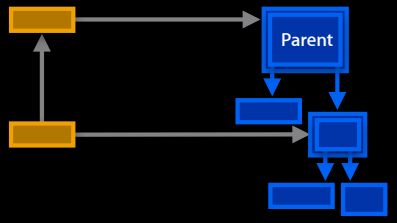
View Controller Evolution

Containment—Worst practice



View Controller Evolution

Containment—Worst practice



- Calling these methods on a class you did not implement

View Controller Evolution

Containment (embedded view controllers)

View Controller Evolution

Containment (embedded view controllers)

There should be a circle for that

View Controller Evolution

Summary—Containment API changes

View Controller Evolution

Summary—Containment API changes


```
// Deprecated in iOS 6.0
-(BOOL)automaticallyForwardAppearanceAndRotationMethodsToChildViewControllers
```

View Controller Evolution

Summary—Containment API changes


```
// Deprecated in iOS 6.0
-(BOOL)automaticallyForwardAppearanceAndRotationMethodsToChildViewControllers

// Introduced as API in iOS 6.0
- (BOOL)shouldAutomaticallyForwardRotationMethods;
- (BOOL)shouldAutomaticallyForwardAppearanceMethods;
```

View Controller Evolution

Summary—Containment API changes


```
// Deprecated in iOS 6.0
-(BOOL)automaticallyForwardAppearanceAndRotationMethodsToChildViewControllers

// Introduced as API in iOS 6.0
- (BOOL)shouldAutomaticallyForwardRotationMethods;
- (BOOL)shouldAutomaticallyForwardAppearanceMethods;

// Available in iOS 5.0 and iOS 6.0. Introduced as API in iOS 6.0
- (void)beginAppearanceTransition:(BOOL)isAppearing animated:(BOOL)animated ;
- (void)endAppearanceTransition;
```

View Controllers Today

Autorotation API and behavioral changes

View Controller Evolution

Autorotation—iOS 5 and earlier

View Controller Evolution

Autorotation—iOS 5 and earlier

- UIViewController's would override

```
-shouldAutoRotateToInterfaceOrientation:
```

View Controller Evolution

Autorotation—iOS 5 and earlier

- UIViewController's would override

  `-shouldAutoRotateToInterfaceOrientation:`

  - Called before rotation

View Controller Evolution

Autorotation—iOS 5 and earlier

- UIViewController's would override

  `-shouldAutoRotateToInterfaceOrientation:`

  - Called before rotation
  - Called before presentation

View Controller Evolution

Autorotation—iOS 5 and earlier

- UIViewController's would override

  `-shouldAutoRotateToInterfaceOrientation:`

  - Called before rotation
  - Called before presentation
- Containers often deferred to their children

View Controller Evolution

Autorotation—Prepare to think differently

View Controller Evolution

Autorotation—Prepare to think differently

- Problems with shouldAutorotateToInterfaceOrientation:

View Controller Evolution

Autorotation—Prepare to think differently

- Problems with shouldAutorotateToInterfaceOrientation:
  - Conflates supported interface orientations with rotation

View Controller Evolution

Autorotation—Prepare to think differently

- Problems with shouldAutorotateToInterfaceOrientation:
  - Conflates supported interface orientations with rotation
  - Allows children to veto the supported orientations of their parents

View Controller Evolution

Autorotation—Prepare to think differently

- Problems with shouldAutorotateToInterfaceOrientation:
  - Conflates supported interface orientations with rotation
  - Allows children to veto the supported orientations of their parents
  - Encourages the use of interface orientation as a way to control layout

View Controller Evolution

Autorotation—Prepare to think differently

- Problems with shouldAutorotateToInterfaceOrientation:
  - Conflates supported interface orientations with rotation
  - Allows children to veto the supported orientations of their parents
  - Encourages the use of interface orientation as a way to control layout
- Other problems
  - Interface orientation for many view controllers is meaningless

View Controller Evolution

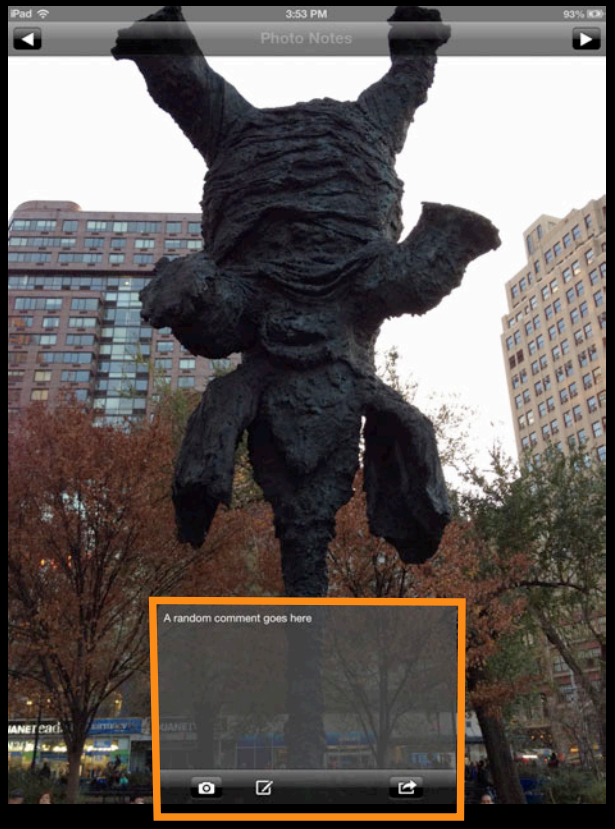
Autorotation—Prepare to think differently

View Controller Evolution

Autorotation—Prepare to think differently

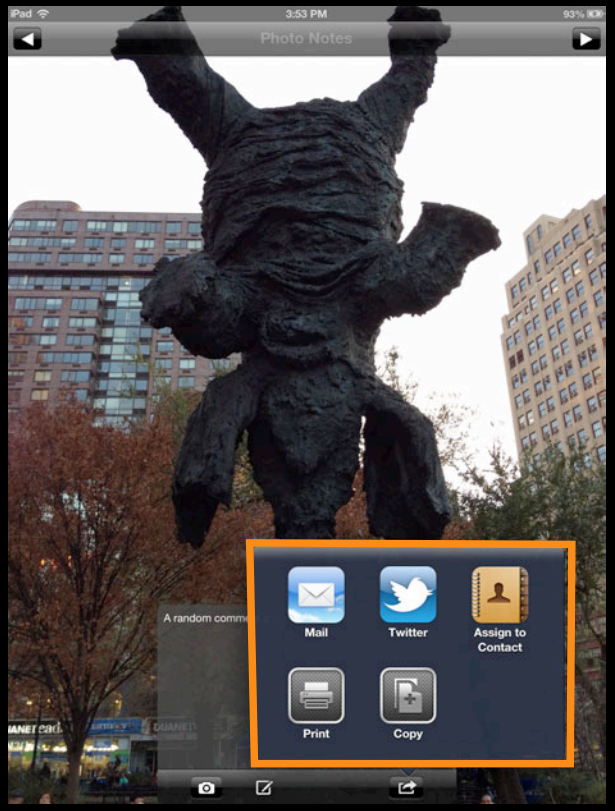
View Controller Evolution

Autorotation—Prepare to think differently



View Controller Evolution

Autorotation—Prepare to think differently



View Controller Evolution

Autorotation—Prepare to think differently

View Controller Evolution

Autorotation—Prepare to think differently

- Problems with shouldAutorotateToInterfaceOrientation:
  - Conflates supported interface orientations with rotation
  - Allows children to veto the supported orientations of their parents
  - Encourages the use of interface orientation as a way to control layout
- Other problems
  - Interface orientation for many view controllers is meaningless

View Controller Evolution

Autorotation—Prepare to think differently

- Problems with shouldAutorotateToInterfaceOrientation:
  - Conflates supported interface orientations with rotation
  - Allows children to veto the supported orientations of their parents
  - Encourages the use of interface orientation as a way to control layout
- Other problems
  - Interface orientation for many view controllers is meaningless
  - As of iOS 5, rotation cannot reliably be used for layout

View Controller Evolution

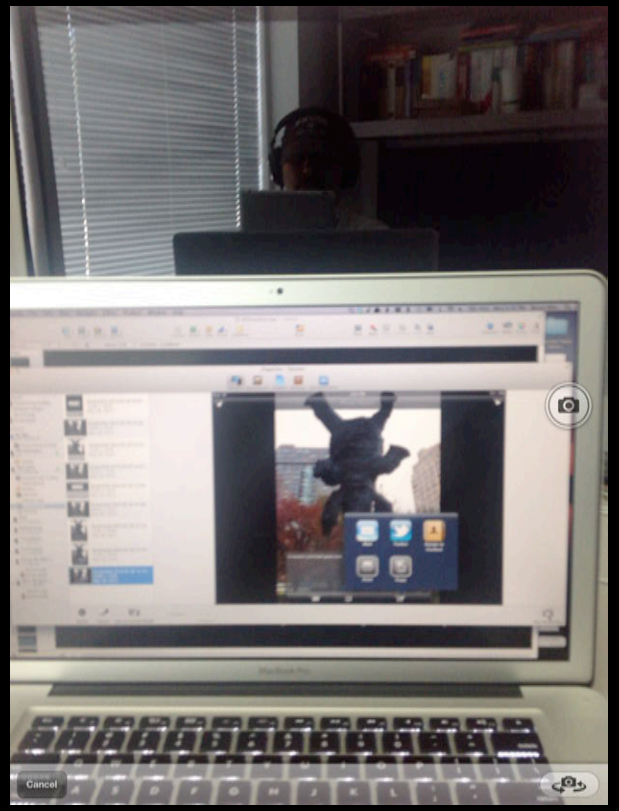
Autorotation—Prepare to think differently

View Controller Evolution

Autorotation—Prepare to think differently

View Controller Evolution

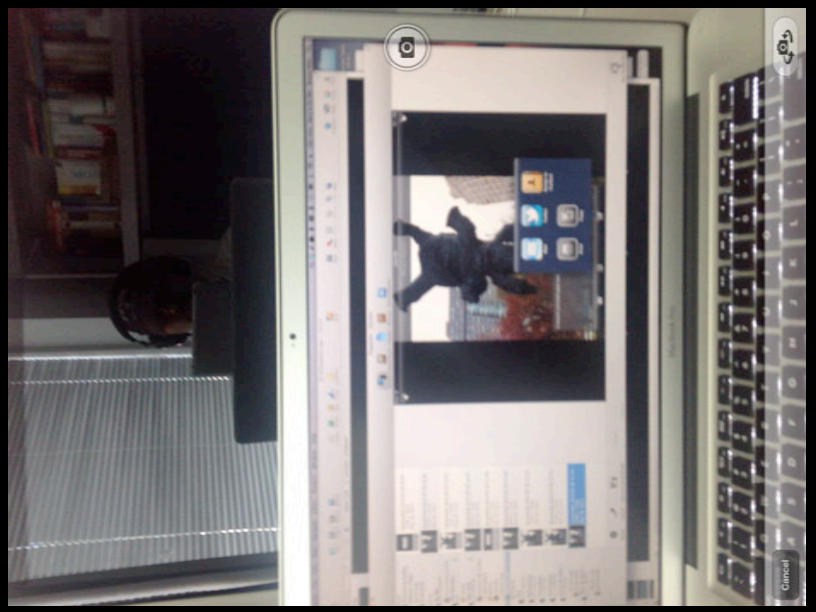
Autorotation—Prepare to think differently



View Controller Evolution

Autorotation—Prepare to think differently

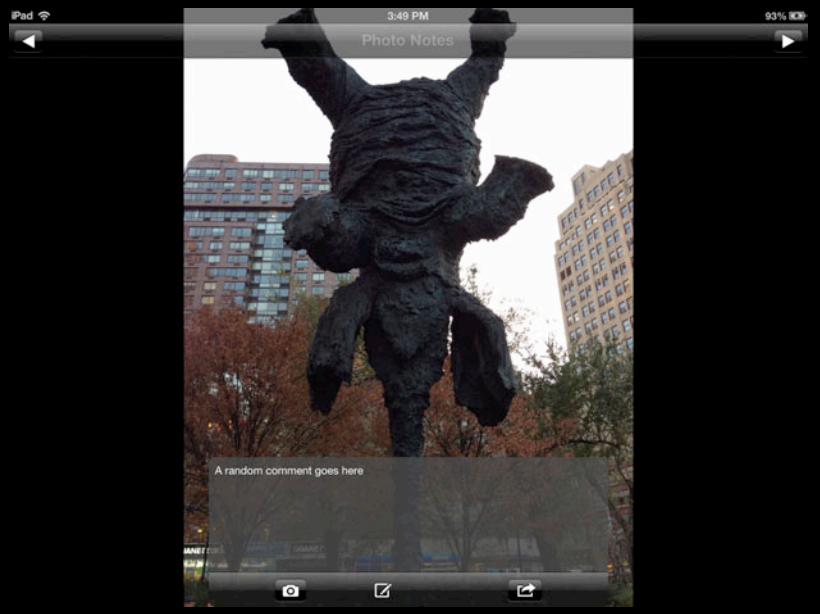
```
willRotateToInterfaceOrientation:duration:
willAnimateRotationToInterfaceOrientation:duration
didRotateFromInterfaceOrientation:
```



View Controller Evolution

Autorotation—Prepare to think differently

No rotation callbacks in iOS 5 and later



View Controller Evolution

Autorotation—Targeting earlier iOS releases

View Controller Evolution

Autorotation—Targeting earlier iOS releases

- On iOS 5 and later, rotation callbacks cannot reliably be used for layout

View Controller Evolution

Autorotation—Targeting earlier iOS releases

- On iOS 5 and later, rotation callbacks cannot reliably be used for layout
  - Pre iOS 5, behavior can be determined by

```
NO == [UIViewController
instancesRespondToSelector:@selector(viewWillLayoutSubviews)]
```

View Controller Evolution

Autorotation—Targeting earlier iOS releases

- On iOS 5 and later, rotation callbacks cannot reliably be used for layout
  - Pre iOS 5, behavior can be determined by

```
NO == [UIViewController
instancesRespondToSelector:@selector(viewWillLayoutSubviews)]
```

  - Refactor layout code to be used at multiple call sites
    - On iOS 6, use updateViewConstraints
    - On iOS 5, use viewWillLayoutSubviews
    - Pre iOS 5, a selector check is required

View Controller Evolution

Autorotation—Think differently

View Controller Evolution

Autorotation—Think differently

- View controllers should make a best effort to support ALL orientations

View Controller Evolution

Autorotation—Think differently

- View controllers should make a best effort to support ALL orientations
- A child view controller should be able to layout in any frame its parent specifies

View Controller Evolution

Autorotation—Think differently

- View controllers should make a best effort to support ALL orientations
- A child view controller should be able to layout in any frame its parent specifies
- View controllers can only support an orientation different from the status bar orientation when presented

View Controller Evolution

Autorotation—Think differently

- View controllers should make a best effort to support ALL orientations
- A child view controller should be able to layout in any frame its parent specifies
- View controllers can only support an orientation different from the status bar orientation when presented

```
presentViewController:animated:completion:
preferredInterfaceForPresentation
```

View Controller Evolution

Autorotation—Think differently

- View controllers should make a best effort to support ALL orientations
- A child view controller should be able to layout in any frame its parent specifies
- View controllers can only support an orientation different from the status bar orientation when presented

```
presentViewController:animated:completion:
preferredInterfaceForPresentation
```

- Only the root or topmost full screen controller is consulted

View Controller Evolution

Autorotation—Think differently

- View controllers should make a best effort to support ALL orientations
- A child view controller should be able to layout in any frame its parent specifies
- View controllers can only support an orientation different from the status bar orientation when presented

```
presentViewController:animated:completion:
preferredInterfaceForPresentation
```

- Only the root or topmost full screen controller is consulted
- An application should be able to indicate its supported orientations

View Controller Evolution

Autorotation—Think differently

- View controllers should make a best effort to support ALL orientations
- A child view controller should be able to layout in any frame its parent specifies
- View controllers can only support an orientation different from the status bar orientation when presented

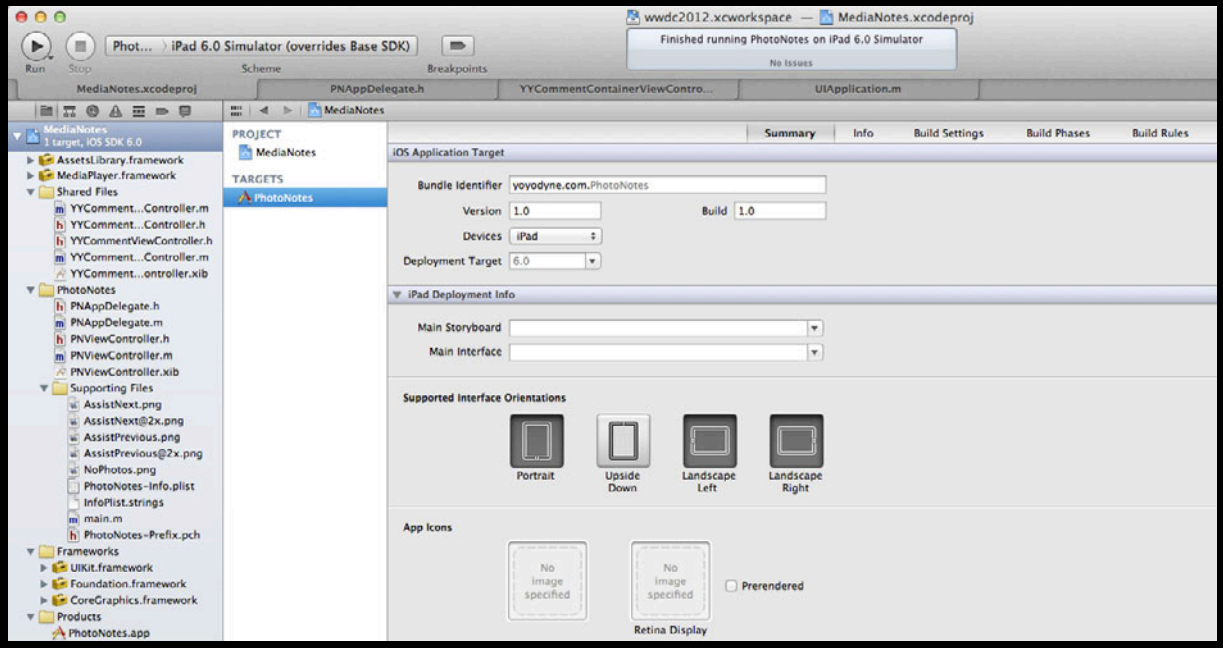
```
presentViewController:animated:completion:
preferredInterfaceForPresentation
```

- Only the root or topmost full screen controller is consulted
- An application should be able to indicate its supported orientations

```
Info.plist
```

View Controllers Today

Autorotation—Think differently



View Controller Evolution

Autorotation—Think differently

- View controllers should make a best effort to support ALL orientations
- A child view controller should be able to layout in any frame its parent specifies
- View controllers can only support an orientation different from the status bar orientation when presented

```
presentViewController:animated:completion:
preferredInterfaceForPresentation
```

- Only the root or topmost full screen controller is consulted
- An application should be able to indicate its supported orientations

```
Info.plist
```

View Controller Evolution

Autorotation—Think differently

- View controllers should make a best effort to support ALL orientations
- A child view controller should be able to layout in any frame its parent specifies
- View controllers can only support an orientation different from the status bar orientation when presented

```
presentViewController:animated:completion:
preferredInterfaceForPresentation
```

- Only the root or topmost full screen controller is consulted
- An application should be able to indicate its supported orientations

```
Info.plist
application:supportedInterfaceOrientationsForWindow:
```

View Controllers Today

Summary—Autorotation API changes

View Controllers Today

Summary—Autorotation API changes

- UIViewController.h

```
// Deprecated in iOS 6.0.
- (BOOL)shouldAutorotateToInterfaceOrientation(UIInterfaceOrientation)toOrientation;
```

View Controllers Today

Summary—Autorotation API changes

- UIViewController.h

```
// Deprecated in iOS 6.0.
- (BOOL)shouldAutorotateToInterfaceOrientation(UIInterfaceOrientation)toOrientation;

// Introduced as API in iOS 6.0
- (NSUInteger)supportedInterfaceOrientations
- (UIInterfaceOrientation)preferredInterfaceOrientationForPresentation;
```

View Controllers Today

Summary—Autorotation API changes

- UIViewController.h

```
// Deprecated in iOS 6.0.
- (BOOL)shouldAutorotateToInterfaceOrientation(UIInterfaceOrientation)toOrientation;

// Introduced as API in iOS 6.0
- (NSUInteger)supportedInterfaceOrientations
- (UIInterfaceOrientation)preferredInterfaceOrientationForPresentation;
```

- UIApplication.h

```
// Introduced as API in iOS 6.0
UIKIT_EXTERN NSString *const UIApplicationInvalidInterfaceOrientationException;
```

View Controller Evolution

Autorotation—Adapting to iOS 6

View Controller Evolution

Autorotation—Adapting to iOS 6

- Pre-iOS 6 autorotation behavior can be determined by

```
Class UIVC = [UIViewController class];
NO == [UIVC
instancesRespondToSelector:@selector(supportedInterfaceOrientations)];
```

View Controller Evolution

Autorotation—Adapting to iOS 6

- Pre-iOS 6 autorotation behavior can be determined by

```
Class UIVC = [UIViewController class];
NO == [UIVC
instancesRespondToSelector:@selector(supportedInterfaceOrientations)];
```

- Provide implementations for supportedInterfaceOrientations as necessary

View Controller Evolution

Autorotation—Adapting to iOS 6

- Pre-iOS 6 autorotation behavior can be determined by

```
Class UIVC = [UIViewController class];
NO == [UIVC
instancesRespondToSelector:@selector(supportedInterfaceOrientations)];
```

- Provide implementations for supportedInterfaceOrientations as necessary
- Container view controllers may need to be subclassed to override supportedInterfaceOrientations

View Controller Evolution

Autorotation—Adapting to iOS 6

- Pre-iOS 6 autorotation behavior can be determined by

```
Class UIVC = [UIViewController class];
NO == [UIVC
instancesRespondToSelector:@selector(supportedInterfaceOrientations)];
```

- Provide implementations for supportedInterfaceOrientations as necessary
- Container view controllers may need to be subclassed to override supportedInterfaceOrientations
- Apps that use setStatusBarOrientation: will need to convert to presentations

View Controllers Today

Summary—Autorotation API changes

View Controllers Today

Summary—Autorotation API changes

- UIApplication.h

```
// Delegate method introduced as API in iOS 6.0
- (NSUInteger)application:(UIApplication *)application
     supportedInterfaceOrientationsForWindow:(UIWindow *)window;
```

View Controllers Today

Summary—Autorotation API changes

- UIApplication.h

```
// Delegate method introduced as API in iOS 6.0
- (NSUInteger)application:(UIApplication *)application
     supportedInterfaceOrientationsForWindow:(UIWindow *)window;

// Deprecated in iOS 6.0
- (void)setStatusBarOrientation;
- (void)setStatusBarOrientation:animated:;
```

View Controllers Today

Autorotation—Think differently

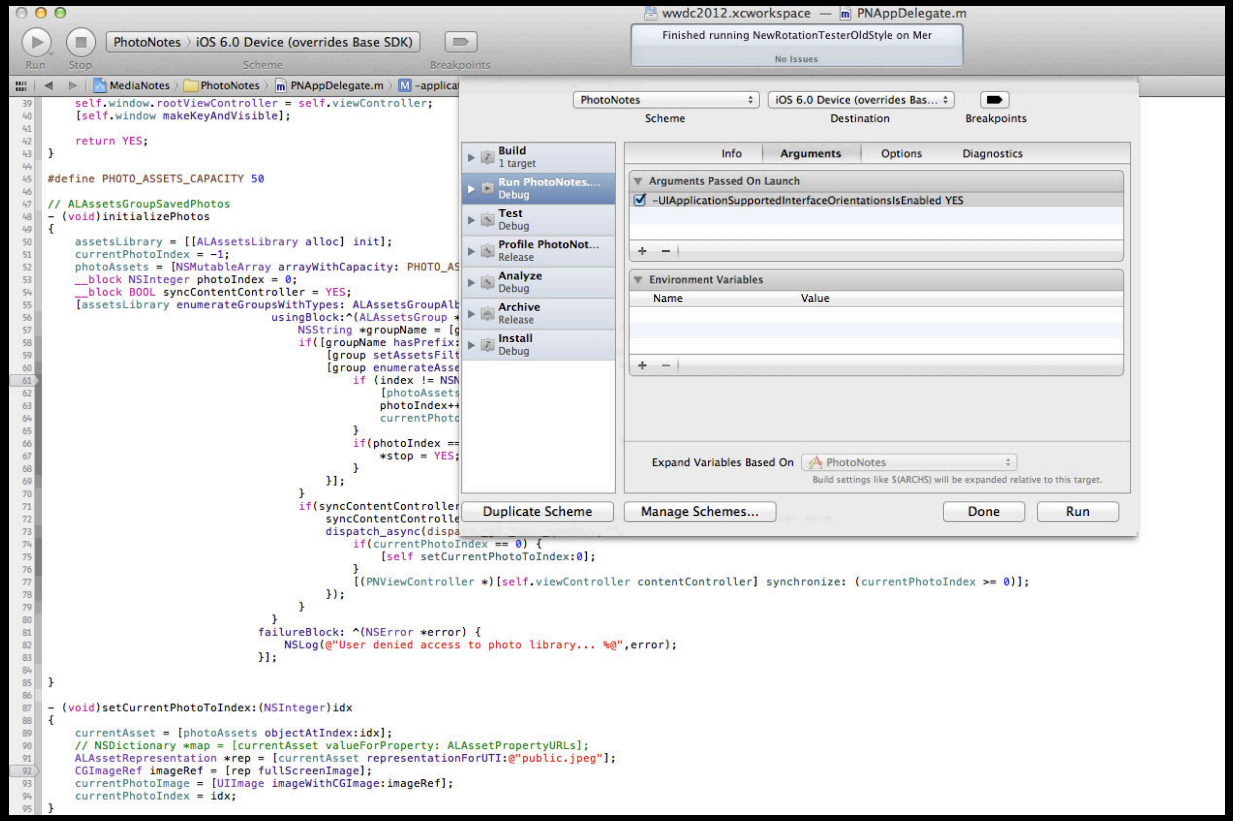
View Controllers Today

Autorotation—Think differently

- Is still evolving for iOS 6
  - A few minor additions still in the works
  - Stay tuned for seed updates and release notes

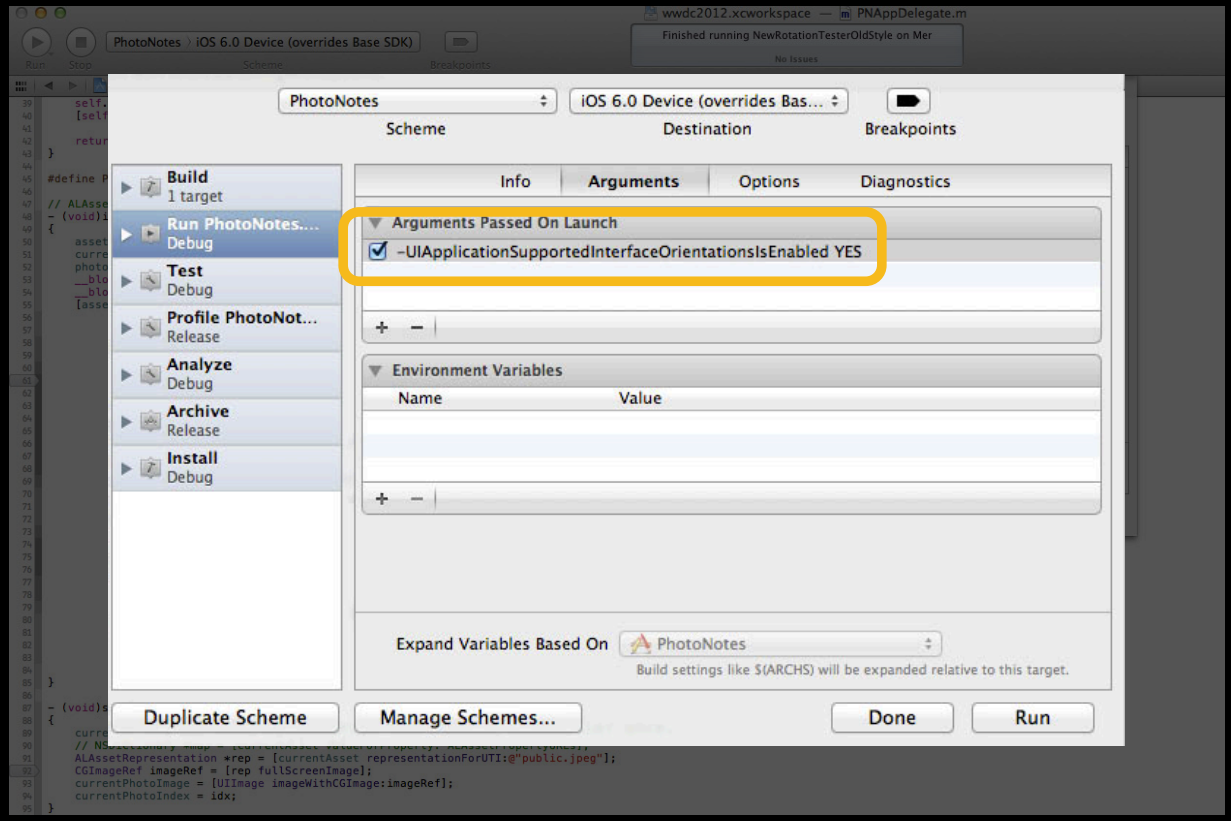
View Controllers Today

Autorotation—Think differently



View Controllers Today

Autorotation—Think differently



View Controllers Today

Other API changes

View Controllers Today

Deprecations

View Controllers Today

Deprecations

- UIViewController.h
  - `- (void)viewWillUnload;`
  - `- (void)viewDidUnload;`

View Controllers Today

Deprecations

- UIViewController.h

```
- (void)viewWillUnload;
- (void)viewDidUnload;
```

```
- (void)didReceiveMemoryWarning {
    if([self.view window] == nil) {
      [photoMap removeAllObjects];
      self.view = nil;
      self.photoImageView = nil;
    }
}
```

View Controllers Today

Deprecations

View Controllers Today

Deprecations

- UIViewController.h

```
@property(nonatomic,readonly) UIViewController *modalViewController;
- (void)presentModalViewController:(UIViewController *)modalViewController
                          animated:(BOOL)animated;
- (void)dismissModalViewControllerAnimated:(BOOL)animated;
```

View Controllers Today

Other new API

View Controllers Today

Other new API

- Constraint-Based Layout

```
// Introduced as API in iOS 6.0
- (void)updateViewConstraints;
```

View Controllers Today

Other new API

- Constraint-Based Layout

```
// Introduced as API in iOS 6.0
- (void)updateViewConstraints;
```

- Storyboard Support
  - Segue Unwinding

View Controllers Today

Other new API

- Constraint-Based Layout

```
// Introduced as API in iOS 6.0
- (void)updateViewConstraints;
```

- Storyboard Support
  - Segue Unwinding
- State Restoration
  - View Controllers are used to indicate what state is saved

Photo Notes—A Social App with a Custom View Controller Container

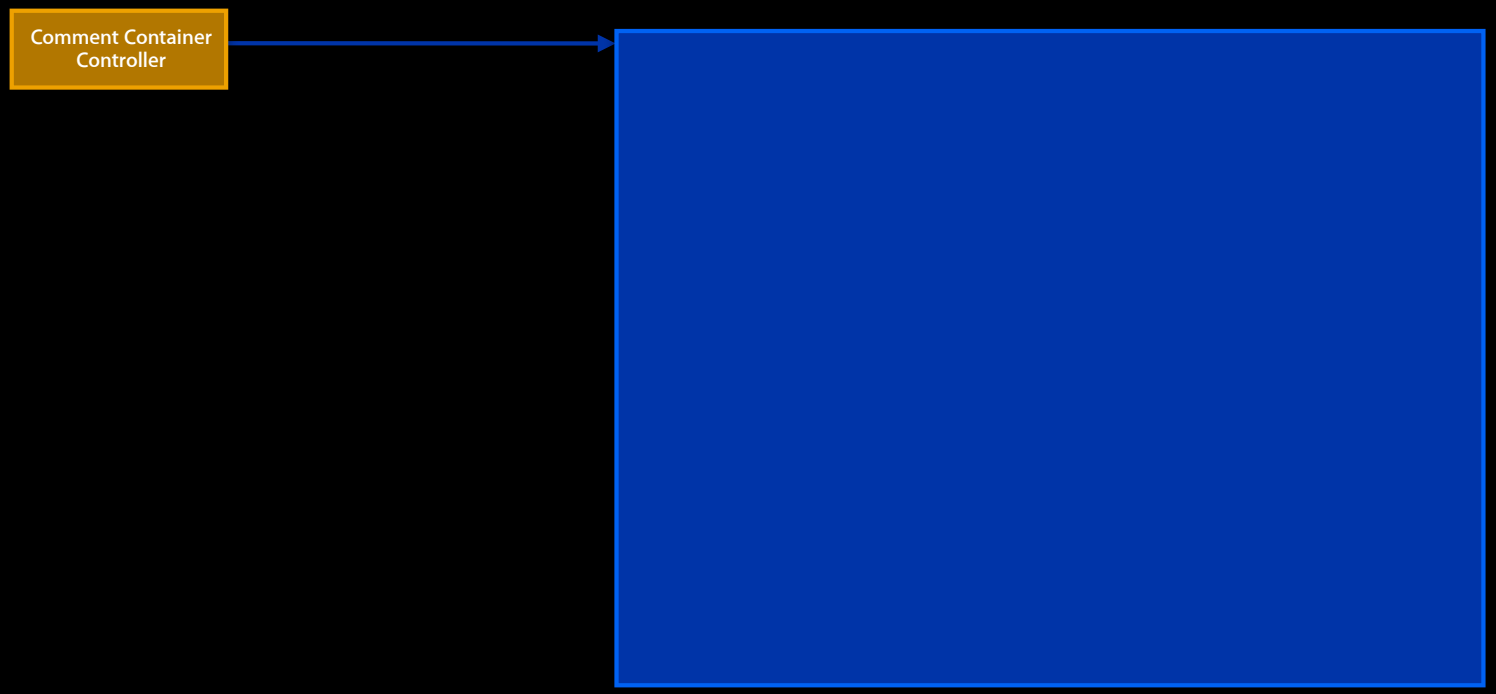
The parent is responsible where their children play

PhotoNotes

Basic design

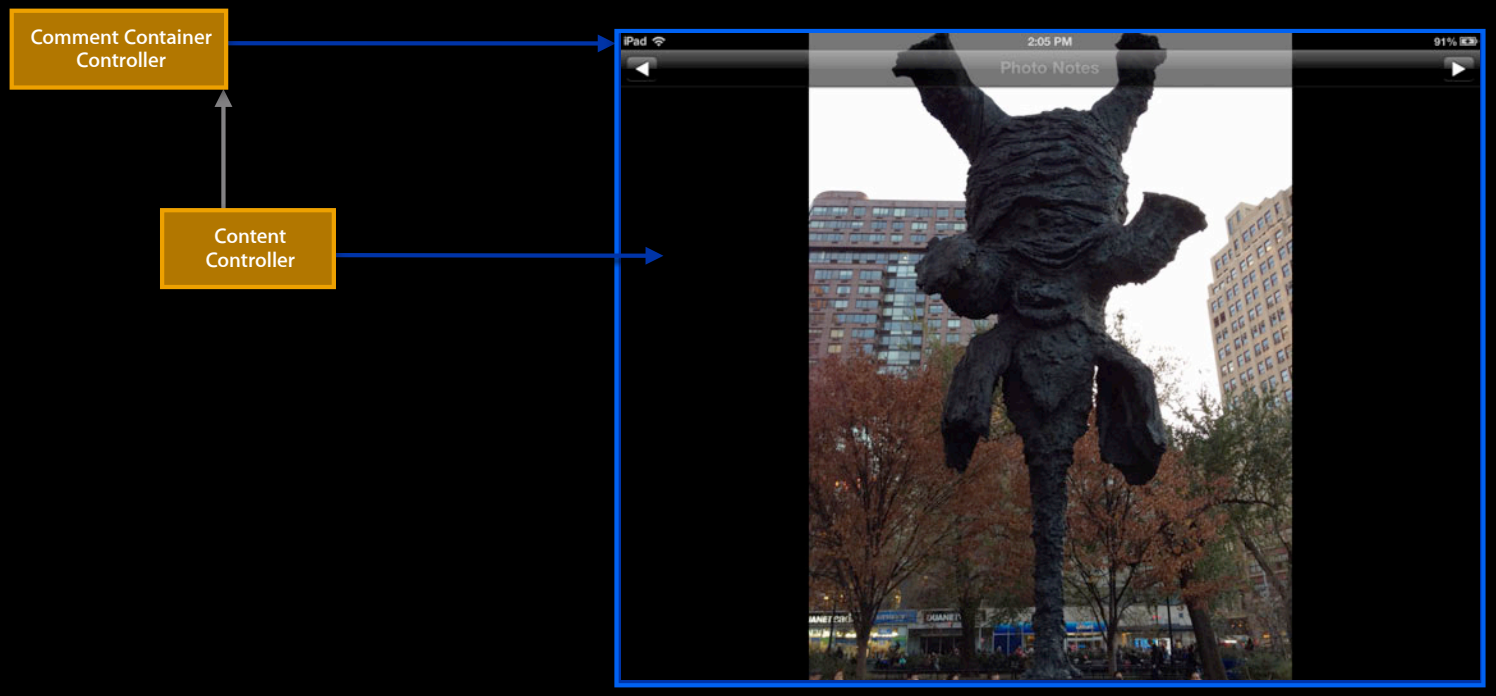
PhotoNotes

Basic design



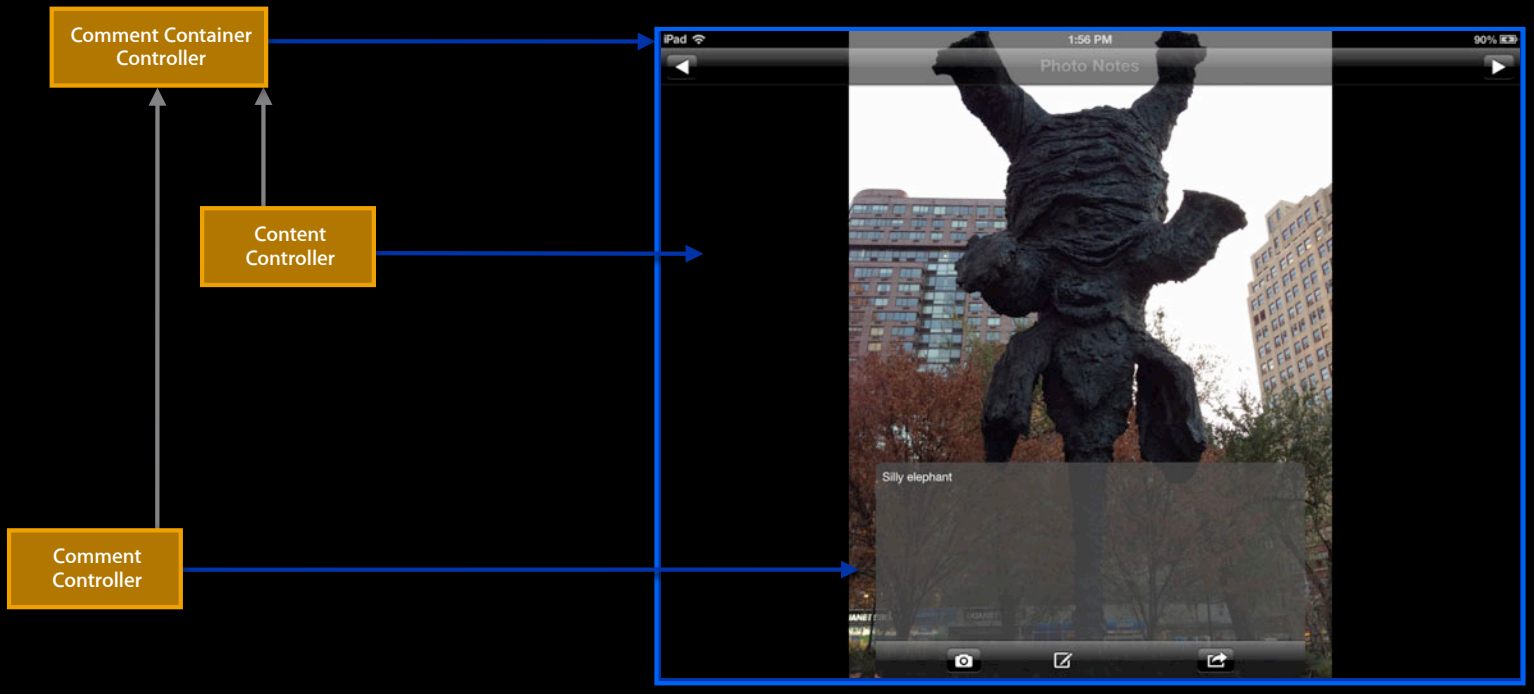
PhotoNotes

Basic design



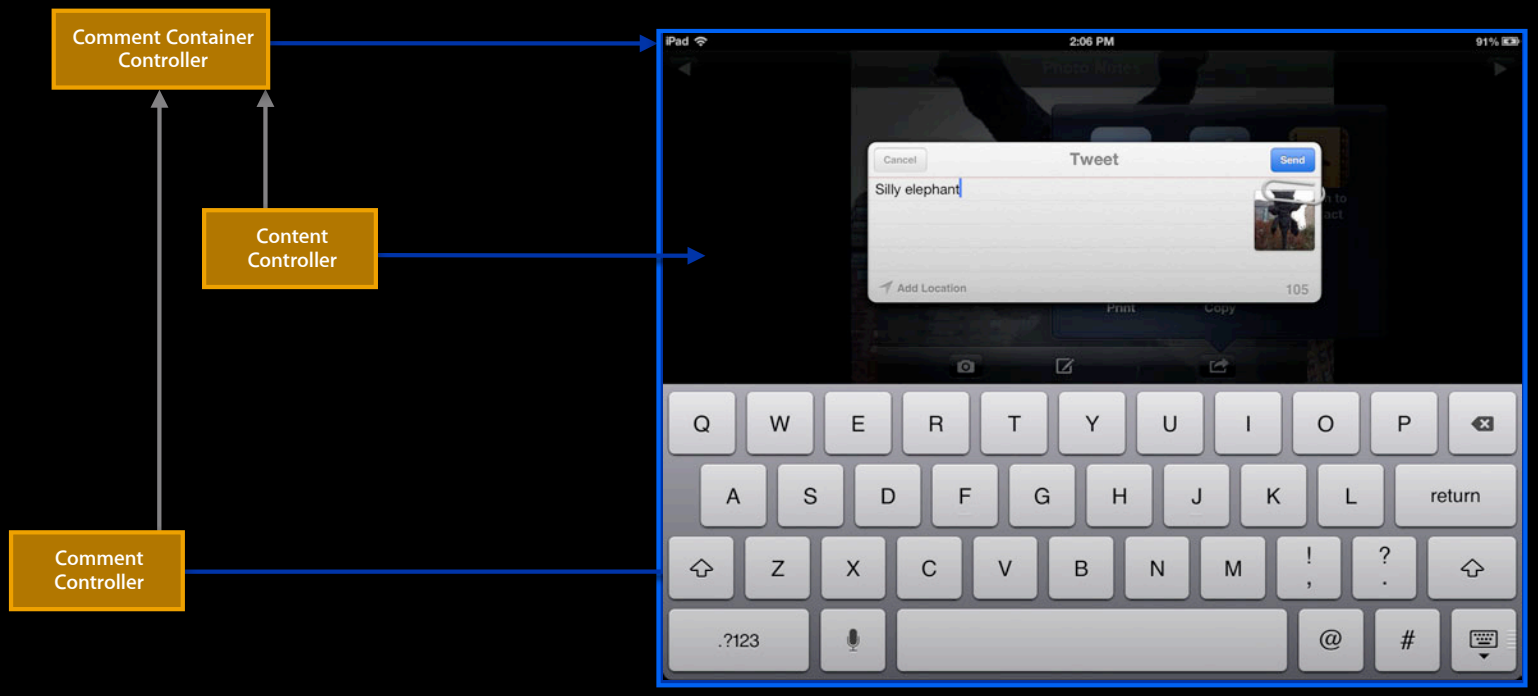
PhotoNotes

Basic design



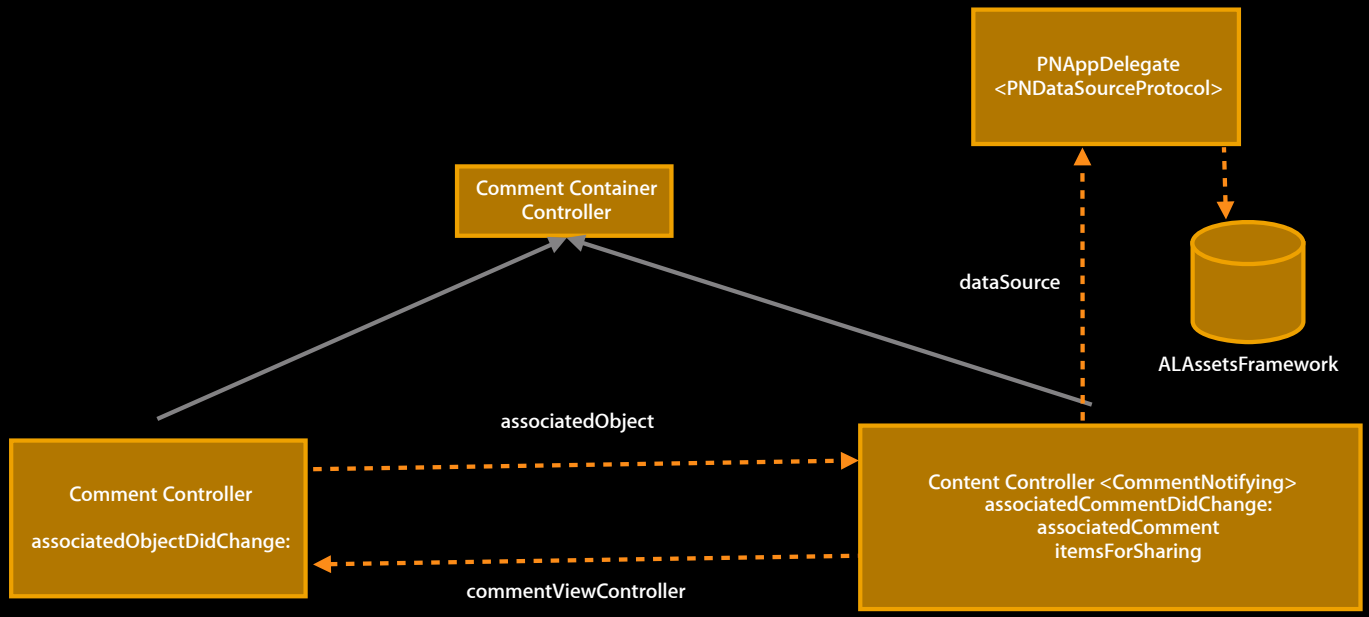
PhotoNotes

Basic design



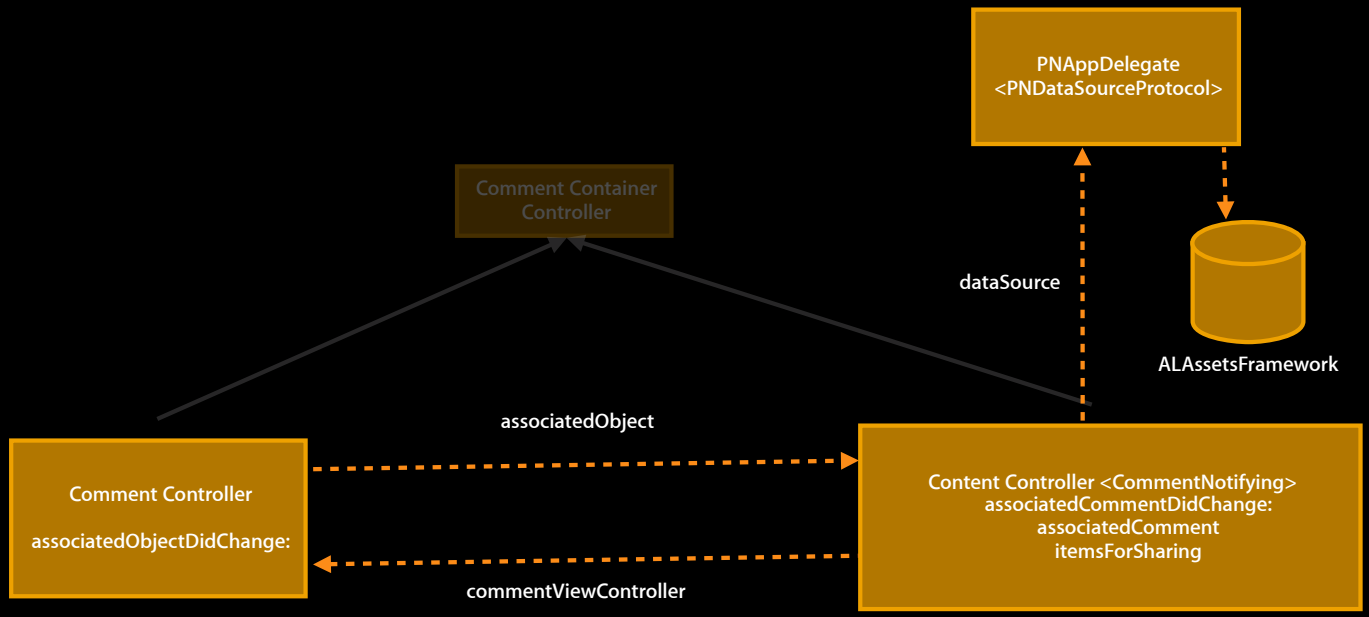
PhotoNotes

Basic design—Model business



PhotoNotes

Basic design—Model business



PhotoNotes

Why a custom container controller?

PhotoNotes

Why a custom container controller?

- It defines a custom application flow
  - It is reusable in different situations
  - It interoperates with the rest of UIKit

Demo

PhotoNotes

Takeaway Thoughts

View Controllers Today

Takeaway thoughts

- Custom container controllers are for new application flows
  - Otherwise use system containers

View Controllers Today

Takeaway thoughts

View Controllers Today

Takeaway thoughts

- Don’t rely on interface orientation for layout

View Controllers Today

Takeaway thoughts

- Don’t rely on interface orientation for layout
- A parent sets its child’s frame

View Controllers Today

Takeaway thoughts

- Don’t rely on interface orientation for layout
- A parent sets its child’s frame
  - A view controller should never set its own frame

View Controllers Today

Takeaway thoughts

View Controllers Today

Takeaway thoughts

- Autorotation is evolving

View Controllers Today

Takeaway thoughts

- Autorotation is evolving
  - Support all orientations

View Controllers Today

Takeaway thoughts

- Autorotation is evolving
  - Support all orientations
    - (Except upside down on the phone)

View Controllers Today

Takeaway thoughts

- Autorotation is evolving
  - Support all orientations
    - (Except upside down on the phone)
  - Apps can easily indicate the orientations they support

View Controllers Today

Takeaway thoughts

- Autorotation is evolving
  - Support all orientations
    - (Except upside down on the phone)
  - Apps can easily indicate the orientations they support
  - Rotation callbacks are for rotation

View Controllers Today

Takeaway thoughts

View Controllers Today

Takeaway thoughts

- View controllers are the cornerstones of most iOS apps
  - More features will continue to be added
  - More system API will be vended

View Controllers Today

Takeaway thoughts

- View controllers are the cornerstones of most iOS apps
  - More features will continue to be added
  - More system API will be vended
- Design your view controllers with an eye toward reuse
  - Think of how they compose

View Controllers Today

Takeaway thoughts

- View controllers are the cornerstones of most iOS apps
  - More features will continue to be added
  - More system API will be vended
- Design your view controllers with an eye toward reuse
  - Think of how they compose
- Future-proof your apps
  - Adopt new API and avoid deprecated API

Related Sessions

Saving and Restoring Application State on iOS	Russian Hill Thursday 3:15PM
Introduction to Auto Layout for iOS and OS X	Mission Tuesday 10:15AM
Adopting Storyboards in Your App	Marina Wednesday 2:00PM

More Information

Jake Behrens
Cocoa Touch/UIKit Evangelist
behrens@apple.com

Documentation
iOS Development Center
http://developer.apple.com/ios

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


WWDC2012