The Evolution of View Controllers on iOS

Session 236

Matt Gamble, Bruce D. Nilo UIKit Engineers

These are confidential sessions—please refrain from streaming, blogging, or taking pictures

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

Manage a view hierarchy

Manage a view hierarchy

UIView

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









Autorotation

Autorotation

Autorotation

- (void)willRotateToInterfaceOrientation:(UIInterfaceOrientation)toInterfaceOrientation duration:(NSTimeInterval)duration;
- (void)willAnimateRotationToInterfaceOrientation:(UIInterfaceOrientation)toInterfaceOrientation duration:(NSTimeInterval)duration
- (void)didRotateFromInterfaceOrientation:(UIInterfaceOrientation)fromInterfaceOrientation







Centralize responsibility



Centralize responsibility







Centralize responsibility



Centralize responsibility (MVC pattern)



Make common tasks simpler

Make common tasks simpler

- Manage a view hierarchy
- Centralize responsibility














iPad 🗢	9:41 AM	
	First Second	
Row 0		iOS Label
Row 1		Lorem ipsum dolor sit er ellt lamet, consecteaur cillium adipising pecu, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrute exercitation ullamoo laboris nisi ut aliquip ex ea commodo consequat. Duis aute inure dolor in reprehenderit in voluptate veilt esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in cuipa qui officia deserunt moliit anim id est laborum. Nam liber te conscient to factor tum poen legum odioque civuda.
Row 2		
Row 3		
Row 4		
Row 5		
Row 6		
Row 7		Button1 Button2
Row 8		
Row 9		
Row 10		
Row 11		
Row 12		
Button1 Button2		

	9:41 AM	
		Popover
	First Second	Only on
Row 0		Label
Row 1		
Row 2		consectetaur cillium adipisicing pecu, sed do eiusmod tempor incididunt ut labore et
Row 3		veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo
Row 4		consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cilium dolore ou funct auto la paciativa. Executiva
Row 5		sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id
Row 6		est laborum. Nam liber te conscient to factor tum poen legum odioque civiuda.
Row 7		Button1 Button2
Row 8		
Row 9		
Row 10		
Row 11		
Row 12		
Button1 Button2		















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 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 Presentation



Using View Controllers Effectively Presentation










Image: Control of the control of th					
Image: Section 1 Row 0 Row 1 Row 2 Image: Section 2 Row 1 Image: Section 2 Row 10	Pad 주		9:41 AM		
Image: Constant of the second seco		Row 0			
Image: Display and provide the prov		Row 1			
Window Row 3 Bow 1 Row 4 Normality of the search atom with a draining the search atom with a draining the search atom with a marking the search atom w		Row 2			
Loren ipsum dolor sit er elit lamet, consectetaur cilium adipisicing pecu, sed do eusmod tempor incidutori tabore dolor emogra aliqua. Ut enim ad minim veniam, quis nostud exercitation ullamos dolor eu tripian dula parta termina dolor eu tripian tula parta termina dolor eu tripian termina termin		Row 3			
do eiusmot tempor incidiunt ¹ u labore et dolore mapa alqua. Ut enim ad minim ventam, quis nostrud exercitation ullamo laboris nisi ut aliquip ex se commodo consequat. Disi autie fuire dolor in merciendent ruin volupate veit esse ciliund ofore eu fugita dueserut nulla rainin di est laborum. Nam liber te conscient to factor tum poen legum odioque civiuda. Row 5 Button1 Button2 Row 10 Row 11 Row 12 Button1	Lorem ipsum dolor sit er elit lamet, consectetaur cillium adipisicing pecu, se	lamet, Row 4			
weniam, quis nostrud exercitation ullamocha planois nuis taliquipe exe a commodo consequat. Duis aute irure dolor in reprehendent irure dolor in reprehendent irus dolate asse ullimati in culpa qui officia deserunt moliti anin id est laborum. Nam liber te conscient to factor tum poen legum odique civida. Row 6 Button1 Button2 Button1 Button2 Button1 Button2	do elusmod tempor incididu dolore magna aliqua. Ut enir	nt ut labore et m ad minim Row 5			0
Button1 Button2 Button1 Button2 Button1 Button2 Button1 Button2	veniam, quis nostrud exercit laboris nisi ut aliquip ex ea o conseguat Duis aute irure d	tation ullamco commodo Row 6			
sint occaecat cupidatat non proident, sunt ucupa qui offica deserunt mollitanin id est laborum. Nam liber te conscient to factor tum poen legum odioque civiuda. Button1 Button2 Row 10 Row 11 Row 12 Button1 Button2	reprehenderit in voluptate ve dolore eu fugiat nulla pariati	elit esse cillum ur. Excepteur Row 7			
Button1 Button2 Button1 Button2 Button1 Button2	sint occaecat cupidatat non in culpa qui officia deserunt	proident, sunt mollit anim id Row 8			
Button1 Button2 Row 10 Row 11 Row 12 Button1 Button2 Button1 Button2	factor tum poen legum odioc	que civiuda. Row 9			
Row 11 Row 12 Button1 Button2	Button1	Button2 Row 10			
Row 12 Button1 Button2		Row 11			
Button1 Button2		Row 12	·		
		Button1	Button2		







Custom container

[parentViewController addChildViewController:childViewController]



Custom container

[parentViewController addChildViewController:childViewController]

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



Custom container

[parentViewController addChildViewController:childViewController]

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

[childViewController didMoveToParentViewController:parentViewController]





Custom container

[childViewController willMoveToParentViewController:nil]



Custom container

[childViewController willMoveToParentViewController:nil]

[[childViewController view] removeFromSuperview]



Custom container

[childViewController willMoveToParentViewController:nil]

[[childViewController view] removeFromSuperview]

[childViewController removeFromParentViewController]



Custom container

[childViewController willMoveToParentViewController:nil]

[[childViewController view] removeFromSuperview]

[childViewController removeFromParentViewController]



Custom container

[childViewController willMoveToParentViewController:nil]

[[childViewController view] removeFromSuperview]

[childViewController removeFromParentViewController]



• Parents make the rules, children follow them

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

- 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

• Discuss new and deprecated API and behaviors

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

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

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

PhotoNotes



- PhotoNotes
- An app with a custom application flow



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



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



- 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



- 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)





- (B00L) shouldAutomaticallyForwardAppearanceMethods
 - return NO; // Override default which is YES
- }





Containment (embedded view controllers)





- (void)revealChild:(UIViewController *)child

// [self.view addSubview:child.view]

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

Containment (embedded view controllers)





- (void)revealChild:(UIViewController *)child

// [self.view addSubview:child.view]

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

Containment (embedded view controllers)





- (void)revealChild:(UIViewController *)child

```
{
```

// [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)viewWillAppear:(B00L)animated {
 [self.child beginAppearanceTransition: YES
 animated: animated];
 }
- (void)viewDidAppear:(B00L)animated {
 [self.child endAppearanceTransition];
 }

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

```
Parent
Parent
```

```
- (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];
        }];
}
```

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

Containment—**Best practices**



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



```
- (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];
        }];
}
```



```
- (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];
        }];
}
```

Containment—Worst practice



View Controller Evolution Containment—Worst practice



• Calling these methods on a class you did not implement



Containment (embedded view controllers)

There should be a circle for that



View Controller Evolution Summary—Containment API changes



Summary—Containment API changes



// Deprecated in iOS 6.0

-(B00L)automaticallyForwardAppearanceAndRotationMethodsToChildViewControllers

Summary—Containment API changes

EDK

// Deprecated in iOS 6.0

 $-({\tt B00L}) automatically {\tt ForwardAppearanceAndRotation} Methods {\tt ToChildViewControllers}$

// Introduced as API in iOS 6.0

- (B00L)shouldAutomaticallyForwardRotationMethods;
- (B00L)shouldAutomaticallyForwardAppearanceMethods;

Summary—Containment API changes



// Deprecated in iOS 6.0

-(B00L)automaticallyForwardAppearanceAndRotationMethodsToChildViewControllers

- // Introduced as API in iOS 6.0
- (B00L)shouldAutomaticallyForwardRotationMethods;
- (B00L)shouldAutomaticallyForwardAppearanceMethods;
- // Available in iOS 5.0 and iOS 6.0. Introduced as API in iOS 6.0
- (void)beginAppearanceTransition:(B00L)isAppearing animated:(B00L)animated ;
- (void)endAppearanceTransition;

View Controllers Today Autorotation API and behavioral changes

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



Autorotation—Prepare to think differently

• Problems with shouldAutorotateToInterfaceOrientation:

Autorotation—Prepare to think differently

Problems with shouldAutorotateToInterfaceOrientation:
Conflates supported interface orientations with rotation

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

- 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

- 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









- 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

- 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





Autorotation—Prepare to think differently

willRotateToInterfaceOrientation:duration: willAnimateRotationToInterfaceOrientation:duration didRotateFromInterfaceOrientation:



Autorotation—Prepare to think differently

No rotation callbacks in iOS 5 and later



Autorotation—Targeting earlier iOS releases

Autorotation—Targeting earlier iOS releases

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

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

N0 == [UIViewController
instancesRespondToSelector:@selector(viewWillLayoutSubviews)]

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

N0 == [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

Autorotation—Think differently

Autorotation—Think differently

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

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

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

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

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

000				😤 wwdc2012.xcwor	kspace — 🚹	MediaNotes.xcodeproj		
Phot) iPad 6.0 Simulator (overrides Base SDK)				Finished running PhotoNotes on IPad 6.0 Simulator				
Run Stop Scheme Breakpoints				No Issues				
MediaNotes.xcodeproj	PNApp	Delegate.h	YYCommentContain	erViewContro	UIA	pplication.m		
IN OA I - P	🏥 🔺 🕨 🛅 MediaNe	otes						
Media/Nares Media/Payer.framework Media/Payer.framework Media/Payer.framework Media/Payer.framework Media/Payer.framework Mommentcontroller.m Y'Commentcontroller.m Y'Commentcontroller.m Y'Commentcontroller.m Y'Commentcontroller.m Y'Commentcontroller.m Y'Nommentcontroller.m Y'Nommentcontroller.m Y'Nommentcontroller.m Y'Nommentcontroller.m Y'NewController.m NYViewController.m NYViewController.m Supporting Files AssistYevious.png AssistYevious.png AssistYevious.png AssistYevious.png AssistYevious.png AssistYevious.png AssistYevious.png Supporting Files Methotos.png AssistYevious.png AssistYevious.png AssistYevious.png AssistYevious.png AssistYevious.png AssistYevious.png AssistYevious.png KistRemeworks FondoNotes.png KistRemeworks FondoNotes.png KistRemework FondoNotes.png	m	IOS Application Target Bundle Identifier Version Devices Deployment Target W IPad Deployment Ir Main Storyboard Main Interface Supported Interface	t vovodyne.com.PhotoN 1.0 iPad = 6.0 * fo Orientations Portrait Upsi Upsi Dow No specified	otes Build 1.1	Summary	Info Build Settings	Build Phases	Build Rules
A PhotoNotes.app				Retina Display				

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

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.

- (B00L)shouldAutorotateToInterfaceOrientation(UIInterfaceOrientation)toOrientation;


View Controllers Today

Summary—Autorotation API changes

• UIViewController.h

- // Deprecated in iOS 6.0.
- (B00L)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.
 - (B00L)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;

Autorotation—Adapting to iOS 6

Autorotation—Adapting to iOS 6

Pre-iOS 6 autorotation behavior can be determined by
 Class UIVC = [UIViewController class];
 N0 == [UIVC instancesRespondToSelector:@selector(supportedInterfaceOrientations)];

Autorotation—Adapting to iOS 6

• Pre-iOS 6 autorotation behavior can be determined by

Class UIVC = [UIViewController class]; N0 == [UIVC instancesRespondToSelector:@selector(supportedInterfaceOrientations)];

• Provide implementations for supportedInterfaceOrientations as necessary

Autorotation—Adapting to iOS 6

• Pre-iOS 6 autorotation behavior can be determined by

Class UIVC = [UIViewController class]; N0 == [UIVC instancesRespondToSelector:@selector(supportedInterfaceOrientations)];

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

Autorotation—Adapting to iOS 6

• Pre-iOS 6 autorotation behavior can be determined by

Class UIVC = [UIViewController class]; N0 == [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

00	0		💾 wwdc2012.xcworkspace — 📠 PNAppDelegate.m					
	PhotoNotes) iOS 6.0 Device (overrides Base SDK)		Finished running NewRotationTesterOldStyle on Mer					
Rur	Stop Scheme Break	points	No Issues					
	ৰ 🔺 🖹 MediaNotes 🤉 🧰 PhotoNotes 🤉 💼 PNAppDelegate.m > 🕅 -applica							
39	<pre>self.window.rootViewController = self.viewController;</pre>	PhotoM	lotes					
40	[self.window makeKeyAndVisible];		Scheme Destination Breakpoints					
42	return YES;	n. II.I						
43	}	► D l target	Info Arguments Options Diagnostics					
45	#define PHOTO_ASSETS_CAPACITY 50	. 👝 Run PhotoNotes	Transaction Arguments Passed On Launch					
46	// Al Assats Group Saved Photos		UlApplicationSupportedInterfaceOrientationsIsEnabled YES					
48	- (void)initializePhotos	Test						
49	{	Debug						
50	currentPhotoIndex = -1;	Release	+ -					
52	<pre>photoAssets = [NSMutableArray arrayWithCapacity: PHOTO_AS</pre>	Analyze	Environment Variabler					
54	block BOOL syncContentController = YES;	Debug	Name Value					
55	[assetsLibrary enumerateGroupsWithTypes: ALAssetsGroupAlb	Archive	Name Value					
57	NSString *groupName = [g	Release						
58	if([groupName hasPrefix:	Install Debug						
60	[group setAssetSrift		+ -					
61	if (index != NSN							
63	photoIndex++							
64	currentPhoto							
65	} if(photoIndex ==							
67	<pre>*stop = YES;</pre>		Expand Variables Based On A PhotoNotes					
68			Build settings like \$(ARCHS) will be expanded relative to this target.					
70	}							
71	1t(syncContentController syncContentControlle	Duplicate Scheme	Manage Schemes Done Run					
73	dispatch async (dispa							
74	if(currentPhotoIndex == 0) { Iself = cfurrentPhotoIndex:0]:							
76	}							
77	[(PNViewControll	er *)[self.viewControll	er contentController] synchronize: (currentPhotoIndex >= 0)];					
79	111							
80	fallursBlacks 2(NEEron sorran) (
82	naluureouck: "(NSError ≉error) ∢ NSLog(@"User denied access to photo library %@",error);							
83	}];							
85	}							
86	- (void) setCurrentPhotoToIndex: (NCInteger) idv							
87	<pre>{</pre>							
89	<pre>currentAsset = [photoAssets objectAtIndex:idx]; // NSDictionary man = [currentAsset valueSarDeportunal)</pre>	Asset Property IIPI s1.						
91	ALAssetRepresentation *rep = [currentAsset representation	ForUTI:@"public.jpeg"];						
92	CGImageRef imageRef = [rep fullScreenImage];							
93	<pre>currentPhotoIndex = idx;</pre>							
95								

View Controllers Today Autorotation—Think differently

f. Photo	Notes \$ iOS 6	.0 Device (overrides Bas	•			
lf	Scheme	Destination	Breakpoints			
P ▶ Ø Build 1 target	Info Arg	uments Options	Diagnostics			
et	Arguments Passed On Launch					
Test Debug	+ -					
Profile PhotoNot						
► 🔊 Analyze Debug	v Environment Variables					
►	Name	Value				
► 🕅 Install Debug						
	+ -					
		2				
	Expand Variables Based On	A PhotoNotes	\$			
		Build settings like \$(ARCHS) will be expanded relative to this target.				
Duplicate Scheme	Manage Schemes		Done	Run		

View Controllers Today Other API changes



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



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

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







UIViewController.h

@property(nonatomic,readonly) UIViewController *modalViewController;

- (void)presentModalViewController:(UIViewController *)modalViewController animated:(B00L)animated;
- (void)dismissModalViewControllerAnimated:(B00L)animated;



- Constraint-Based Layout
 - // Introduced as API in iOS 6.0
 - (void)updateViewConstraints;



- Constraint-Based Layout
 - // Introduced as API in iOS 6.0
- (void)updateViewConstraints;
- Storyboard Support
 - Segue Unwinding



- 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









Basic design—Model business



Basic design—Model business



Why a custom container controller?

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

Takeaway thoughts

• Don't rely on interface orientation for layout

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

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

Takeaway thoughts

- Takeaway thoughts
- Autorotation is evolving

- Autorotation is evolving
 - Support all orientations

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

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

- 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

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 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 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

ÉWWDC2012