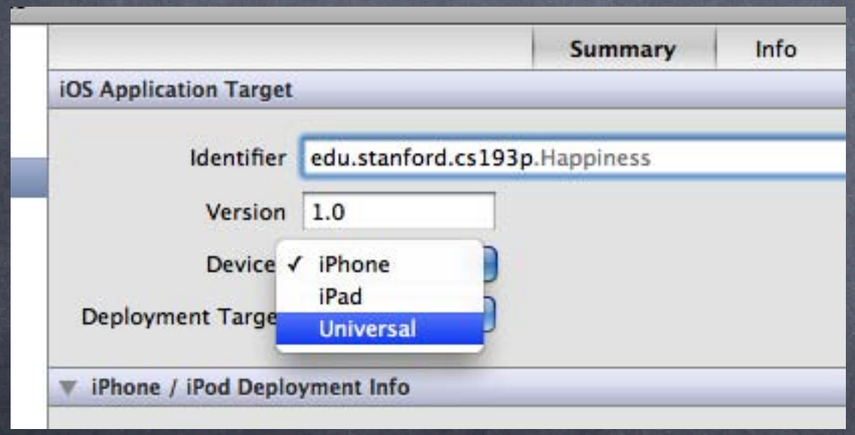
View Controller Lifecycle

Matthias Tretter, @myell0w

Aside: Universal Applications

- A "Universal" application that will run on iPhone or iPad
  Single binary image.
- How to create one
  In an existing iOS 5 iPhone project, select the project in the Navigator (on the left in Xcode)
  Change the Devices pull down under the Summary tab to be Universal.
  Then, lower down in that window, choose the storyboard for each platform.
  An iPad storyboard must be created as an iPad storyboard from the start (no conversion).



Aside: Universal Applications

- How do I figure out "am I on an iPad?"

```
BOOL iPad = (UI_USER_INTERFACE_IDIOM() == UIUserInterfaceIdiomPad);
```

This is not a first-resort. You might well want to use other conditionals before using this.
E.g., checking `self.splitViewController` in your Controller to see if you're in a split view.
We'll see some examples of "other things to check" on the next slide.

Aside: Universal Applications

- Code conditional on whether you're on screen

  Because there is room for more view controllers to be on screen at the same time on the iPad.
  E.g. left and right split view versus views that appear one at a time via navigation controller.
  A simple way to do that is to check a UIView's `window` property. If `nil`, then it's off screen.
  For example, this code snippet might appear in a `UIViewController` somewhere …

```
if (self.view.window) …
```

- How big is the current screen?

```
CGRect screenBounds = [[UIScreen mainScreen] bounds];  // in points
```

  Probably wouldn't want to check an exact size here, but maybe a threshold?

- What is the resolution of the view I'm drawing in?

  Use UIView's `@property (CGFloat) contentScaleFactor`.

View Controller Lifecycle

- View Controllers have a "Lifecycle"
  A sequence of messages is sent to them as they progress through it
- Why does this matter?
  You very commonly override these methods to do certain work
- We've talked about the first part of the lifecycle
  Creation
  This is done mostly either via a segue or storyboard's `instantiateViewControllerWithIdentifer:`.
  Because of this, we rarely (never?) override `UIViewController`'s designated initializer in iOS 5.
  `awakeFromNib` is an option, but we rarely do that either.
  There are better methods to initialize in ...

View Controller

- After instantiation and outlet-setting, `viewDidLoad` is called

  `- (void)viewDidLoad;`

  This is an exceptionally good place to put a lot of setup code.

  But be careful because the geometry of your view (its `bounds`) is not set yet!

  If you need to initialize something based on the geometry of the view, use the next method …

- Just before the view appears on screen, you get notified

  `- (void)viewWillAppear:(BOOL)animated;`

  When this is called, your `bounds` has been set (via your `frame` by your `superview` or some such).

  Your view will probably only get "loaded" once, but it might appear and disappear a lot.

  So don't put something in this method that really wants to be in `viewDidLoad`.

  Otherwise, you might be doing something over and over unnecessarily.

  Use this to optimize performance by waiting until this method (i.e. just before view appears)
  to kick off an expensive operation (might have to put up a spinning "loading" icon though).

  Summary: this method is for geometry-related initialization and lazy execution for performance.

View Controller

- And you get notified when you will disappear off screen too

  This is where you put "remember what's going on" and cleanup code.

```
- (void)viewWillDisappear:(BOOL)animated
{
    [super viewWillDisappear:animated];  // call super in all the viewWill/Did... methods
    // let's be nice to the user and remember the scroll position they were at ...
    [self rememberScrollPosition];  // we'll have to implement this, of course
    // do some other clean up now that we've been removed from the screen
    [self saveDataToPermanentStore]; // maybe do in did instead?
    // but be careful not to do anything time-consuming here, or app will be sluggish
    // maybe even kick off a thread to do what needs doing here
}
```

- There are "did" versions of both of these methods too

```
- (void)viewDidAppear:(BOOL)animated;
- (void)viewDidDisappear:(BOOL)animated;
```

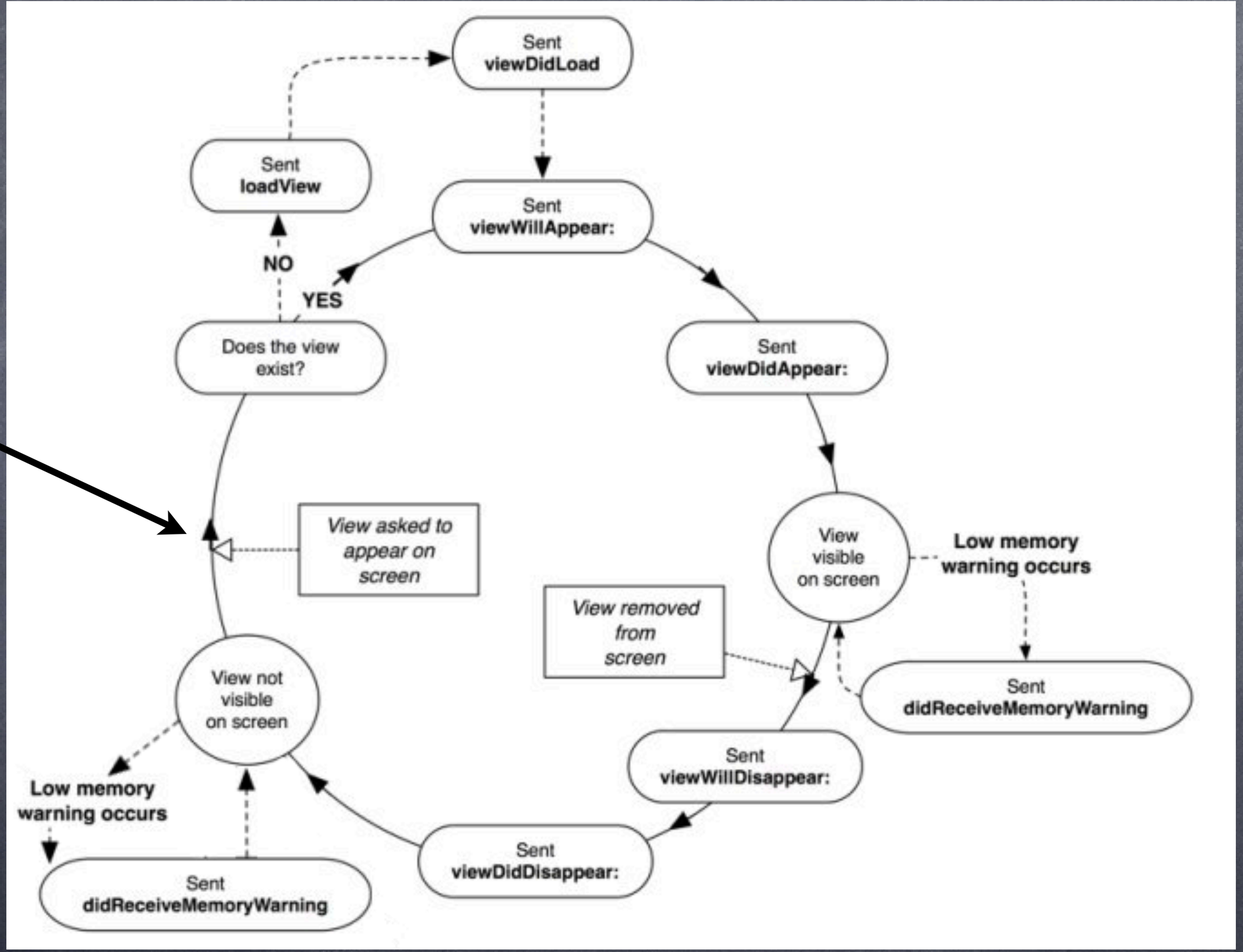
View Controller

- Frame changed? Here's a good place to layout subviews manually
  (if struts and springs are not enough)
  `- (void)view{Will,Did}LayoutSubviews;`
  Called any time a view's `frame` changed and its `subviews` were thus re-layed out.
  For example, autorotation.
  You can reset the `frames` of your `subviews` here (e.g. re-layout your Calculator!)
- Specific notification that rotation will/did happen

```
- (void)willRotateToInterfaceOrientation:(UIInterfaceOrientation)anOrientation
                                duration:(NSTimeInterval)seconds;
- (void)willAnimateRotationToInterfaceOrientation:(UIInterfaceOrientiion)orient
                                duration:(NSTimeInterval)seconds;
- (void)didRotateFromInterfaceOrientation:(UIInterfaceOrientation)anOrientation;
@property UIInterfaceOrientation interfaceOrientation;
```

The property will have the current orientation when each of the above is called.
Example use: doing anything expensive (e.g. an animation maybe?) in `will` and resume it in `did`.

Start


Sent viewDidLoad
Sent loadView
Sent viewWillAppear:
NO
YES
Does the view exist?
Sent viewDidAppear:
View asked to appear on screen
View visible on screen
Low memory warning occurs
View removed from screen
Sent didReceiveMemoryWarning
View not visible on screen
Sent viewWillDisappear:
Low memory warning occurs
Sent viewDidDisappear:
Sent didReceiveMemoryWarning

View Controller Initialization

- Creating a UIViewController from a .xib file
  - This is the old, iOS 4 way. Not covered in this class.
  - You create a .xib file with the same name as your UIViewController subclass.
  - Then use alloc/init to create one.
  - Designated initializer (only if you need to override it, use init otherwise):
  - `- (id)initWithNibName:(NSString *)nibName bundle:(NSBundle *)bundle;`
- Creating a UIViewController's UI in code (no .xib, no storyboard)
  - Override the method - (void)loadView and set self.view to something.
  - This is either/or with storyboards/.xibs.
  - Do NOT implement loadView if you use a storyboard/.xib to create the UIViewController.
  - Do NOT set self.view anywhere else besides in loadView.
  - Do NOT implement loadView without setting self.view (i.e. you must set self.view in loadView).

View Controller Initialization

- Avoid `awakeFromNib` if possible

  It is an acceptable place to initialize stuff for a `UIViewController` from a storyboard/`.xib`.

  But try to put stuff in `viewDidLoad`, `viewWillAppear:` or the segue preparation code instead.

UIView's frame

- Who's responsible for setting a UIView's frame?
  Answer: The object that puts the UIView in a view hierarchy.
- In Interface Builder, you set all view's frames graphically
  You do this by dragging on the little handles.
- What about the frame passed to initWithFrame:?
  If you're putting it into a view hierarchy right away, pick the appropriate frame.
  If you are not, then it doesn't really matter what frame you choose (but avoid CGRectZero).
  The code that eventually DOES put you in a view hierarchy will have to set the frame.
- Setting frames in viewDidLoad
  Recall that your final bounds are not set in viewDidLoad.
  If you create views in code in viewDidLoad, pick sensible frames based on the view's bounds then.
  But be sure to set struts/springs (UIView's autoresizingMask property).
  Think of adding something in viewDidLoad as the same as laying it out in Xcode.
  In both cases, you have to anticipate that the top-level view's bounds will be changed.

UIViewController Container

- used to group other ViewControllers on screen
- pre-defined:
  UITabBarController, UINavigationController (iPhone & iPad)
  UISplitViewController, UIPopoverController (iPad only)
- custom ones can be made („UIViewController containment“)

UITabBarController

„The UITabBarController class implements a specialized view controller that manages a radio-style selection interface. This tab bar interface displays tabs at the bottom of the window for selecting between the different modes and for displaying the views for that mode. This class is generally used as-is“

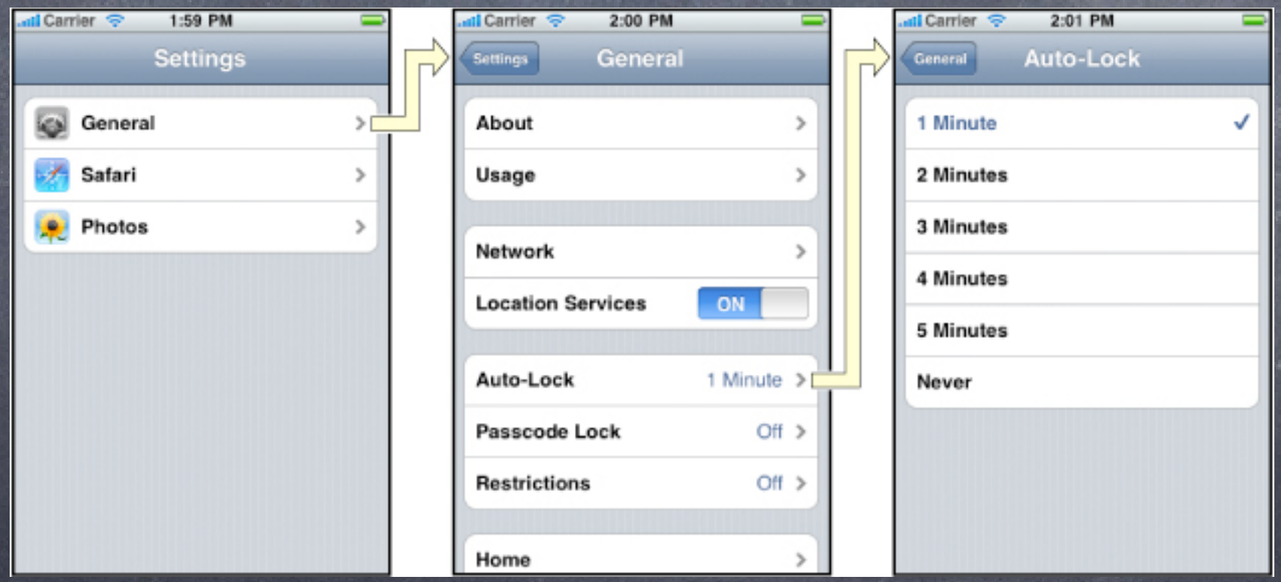
UITabBarController

```
- (void)setViewControllers:(NSArray *)viewControllers
                  animated:(BOOL)animated

@property (nonatomic, weak) UIViewController *selectedViewController
```

UINavigationController

„The UINavigationController class implements a specialized view controller that manages the navigation of hierarchical content. The screens presented by a navigation interface typically mimic the hierarchical organization of your data. At each level of the hierarchy, you provide an appropriate screen to display the content at that level."

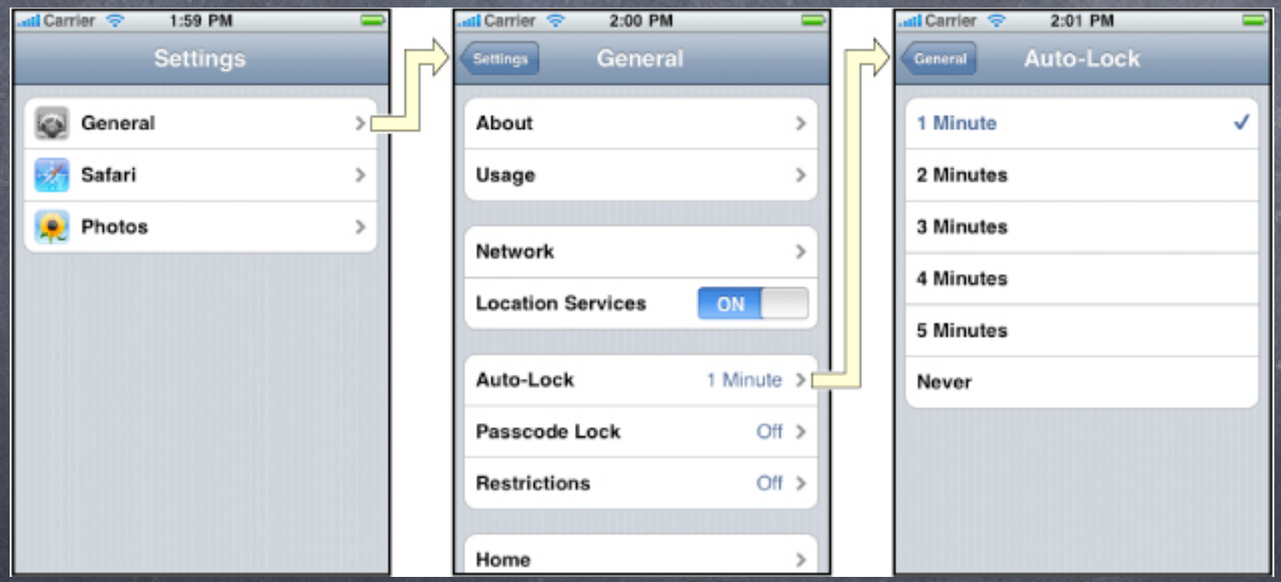


UINavigationController

```
- (void)pushViewController:(UIViewController *)viewController
                  animated:(BOOL)animated

- (UIViewController *)popViewControllerAnimated:(BOOL)animated
- (NSArray *)popToRootViewControllerAnimated:(BOOL)animated

@property(nonatomic, readonly) UIViewController *visibleViewController
```

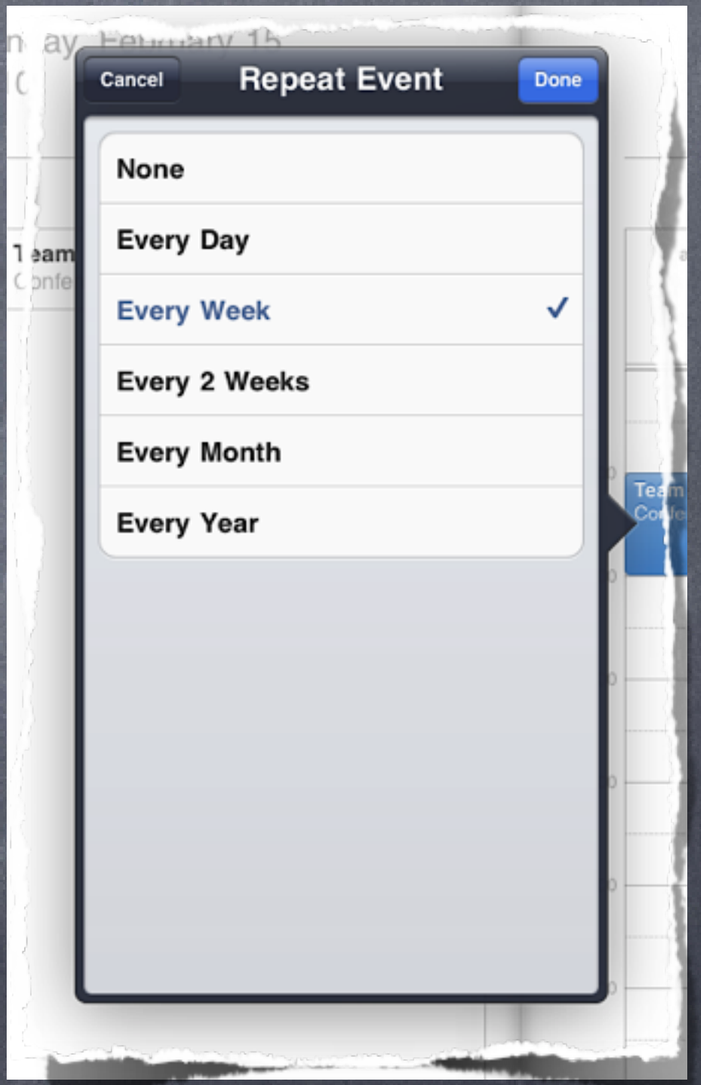


UISplitViewController

„The UISplitViewController class is a container view controller that manages the presentation of two side-by-side view controllers. You use this class to implement a master-detail interface, in which the left-side view controller presents a list of items and the right-side presents details of the selected

UIPopoverController

„The UIPopoverController class is used to manage the presentation of content in a popover. You use popovers to present information temporarily but in a way that does not take over the entire screen“



Further Resources

- UIViewController Class Reference, Apple
  http://developer.apple.com/library/ios/#documentation/uikit/reference/UIViewController_Class/Reference/Reference.html
- iOS Human Interface Guidelines, Apple
  http://developer.apple.com/library/ios/#documentation/UserExperience/Conceptual/MobileHIG/Introduction/Introduction.html