

# Praktikum Entwicklung von Mediensystemen mit iOS

Sommersemester 2013

Fabius Steinberger, Dr. Alexander De Luca

# Today

---

- Organization
- Introduction to iOS programming
- Hello World
- Assignment 1

# Organization

---

- 6 ECTS
- Bachelor: Vertiefendes Thema
- Master: Gruppenpraktikum
- Thursday 14 - 16, Amalienstr. 17 A107
- Check your emails (cip / campus)
- <http://www.medien.ifl.mu.de/lehre/ss13/pem/>

# Roadmap

---

- *April, May*: weekly lectures and individual assignments
- *May, June, July*: app development in teams, 4 milestone presentations
- *July*: final presentation (probably 18.7.2013)

# iOS

---

- Mobile operating system by Apple for iPhone, iPad and iPod Touch
- Based on Unix, derived from OS X
- Latest release: iOS 6 (2012). iOS 7 maybe this summer.
- High market share, high user engagement, high willingness to pay for apps.
- Overall smartphone / tablet market is huge and still growing, and many PEM skills also apply to Android development.



# Layers of iOS

---

## Cocoa Touch

Multi-touch, Web View, Map Kit, Camera, Image Picker...

## Media

Core Audio, PDF, Core Animation, Quartz 2D, OpenGL...

## Core Services

Core Location, Preferences, Address Book, Preferences...

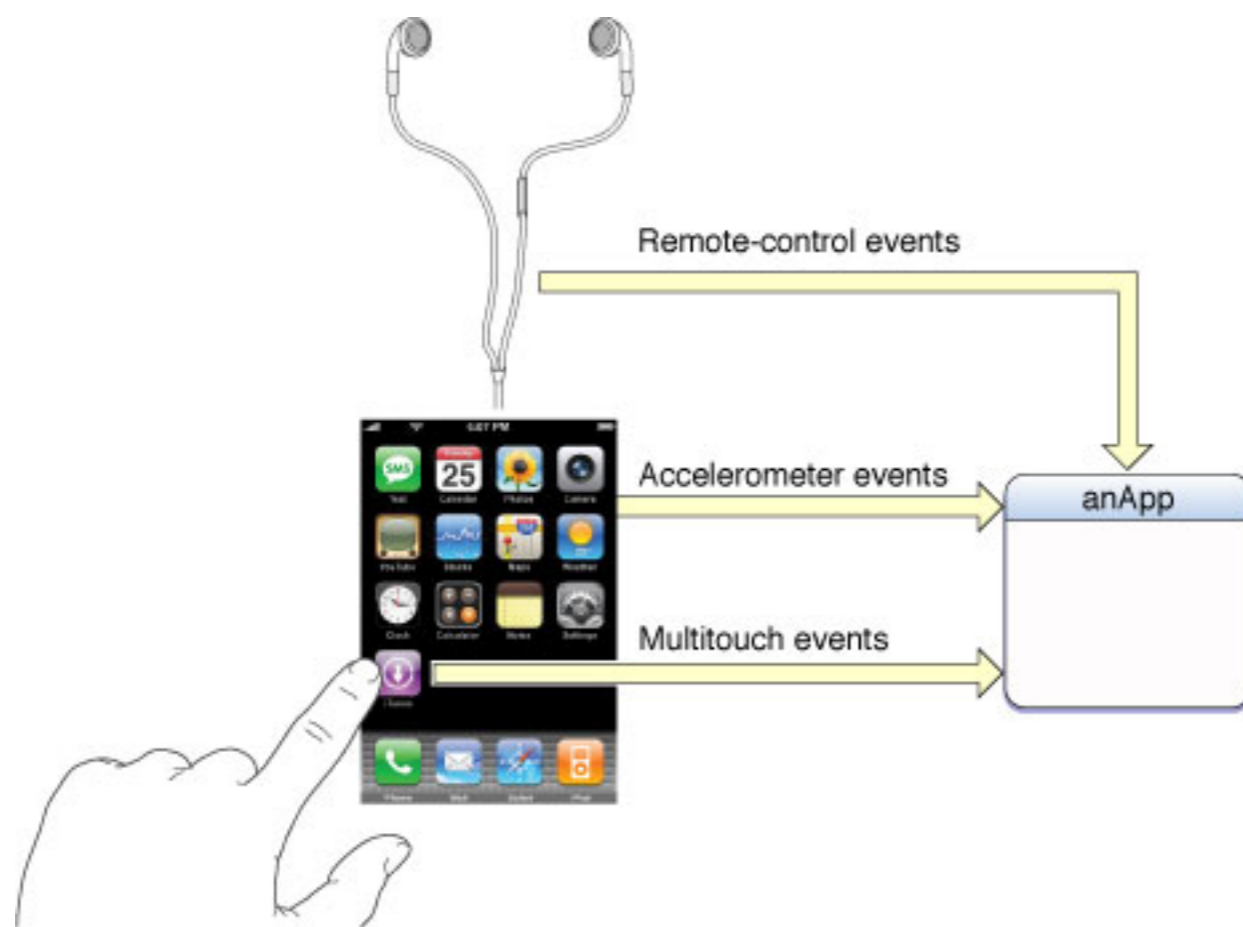
## Core OS





File System, Kernel, Power Management, Security...

# User input

---

- GUI controls: buttons, sliders, switches etc.
- Multi-touch gestures: tap, pinch, rotate, swipe, pan
- Accelerometer: shaking, rotating



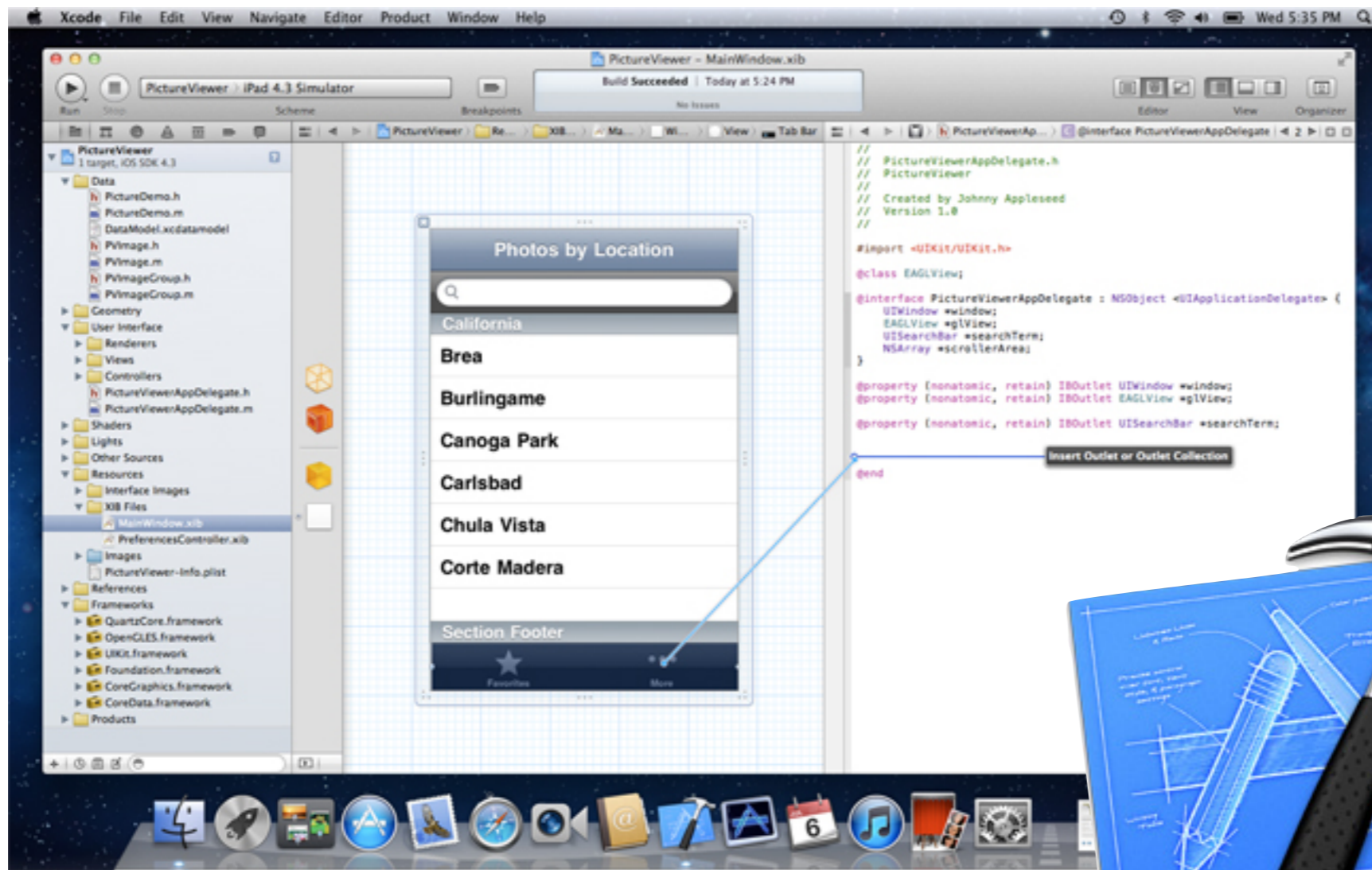
	<b>Tap Gesture Recognizer</b> – Provides a recognizer for tap gestures which land on the view.
	<b>Pinch Gesture Recognizer</b> – Provides a recognizer for pinch gestures which are invoked on the view.
	<b>Rotation Gesture Recognizer</b> – Provides a recognizer for rotation gestures which are invoked on the view.
	<b>Swipe Gesture Recognizer</b> – Provides a recognizer for swipe gestures which are invoked on the view.
	<b>Pan Gesture Recognizer</b> – Provides a recognizer for panning (dragging) gestures which are invoked on the view.
	<b>Long Press Gesture Recognizer</b> – Provides a recognizer for long press gestures which are invoked on the view.

# iOS Development





# Development Environment



XCode

# XCode

---



- **Source editor:** code completion, syntax highlighting, context-sensitive information



- **Interface builder:** UI elements library and inspector, split editor to connect UI with code, Storyboards



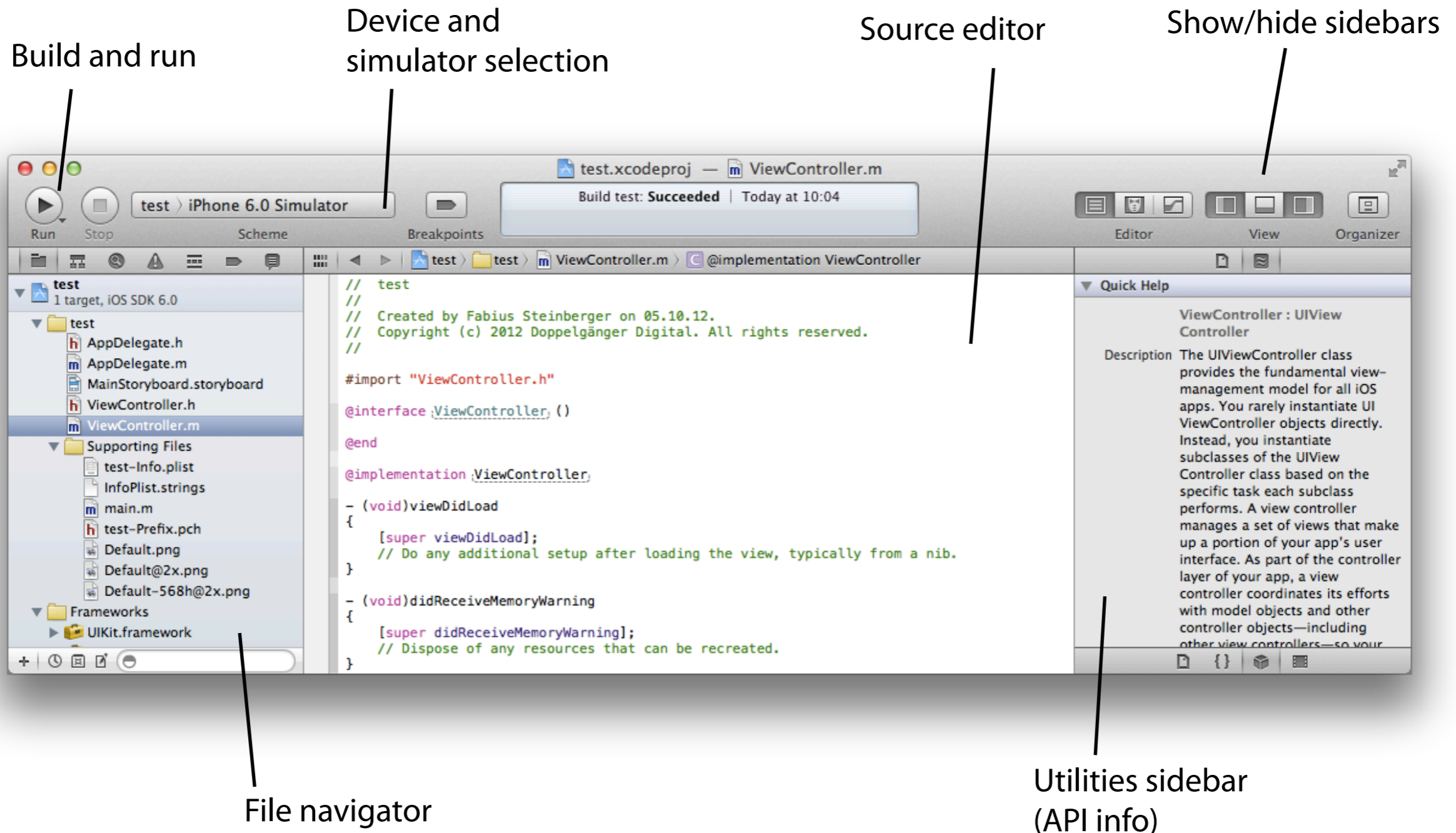
- **Compiler:** C, C++, Objective-C

- **iOS Simulator:** run and test apps on a Mac



- **More:** refactoring, version control, debugging, analysis  
(<https://developer.apple.com/technologies/tools/>)

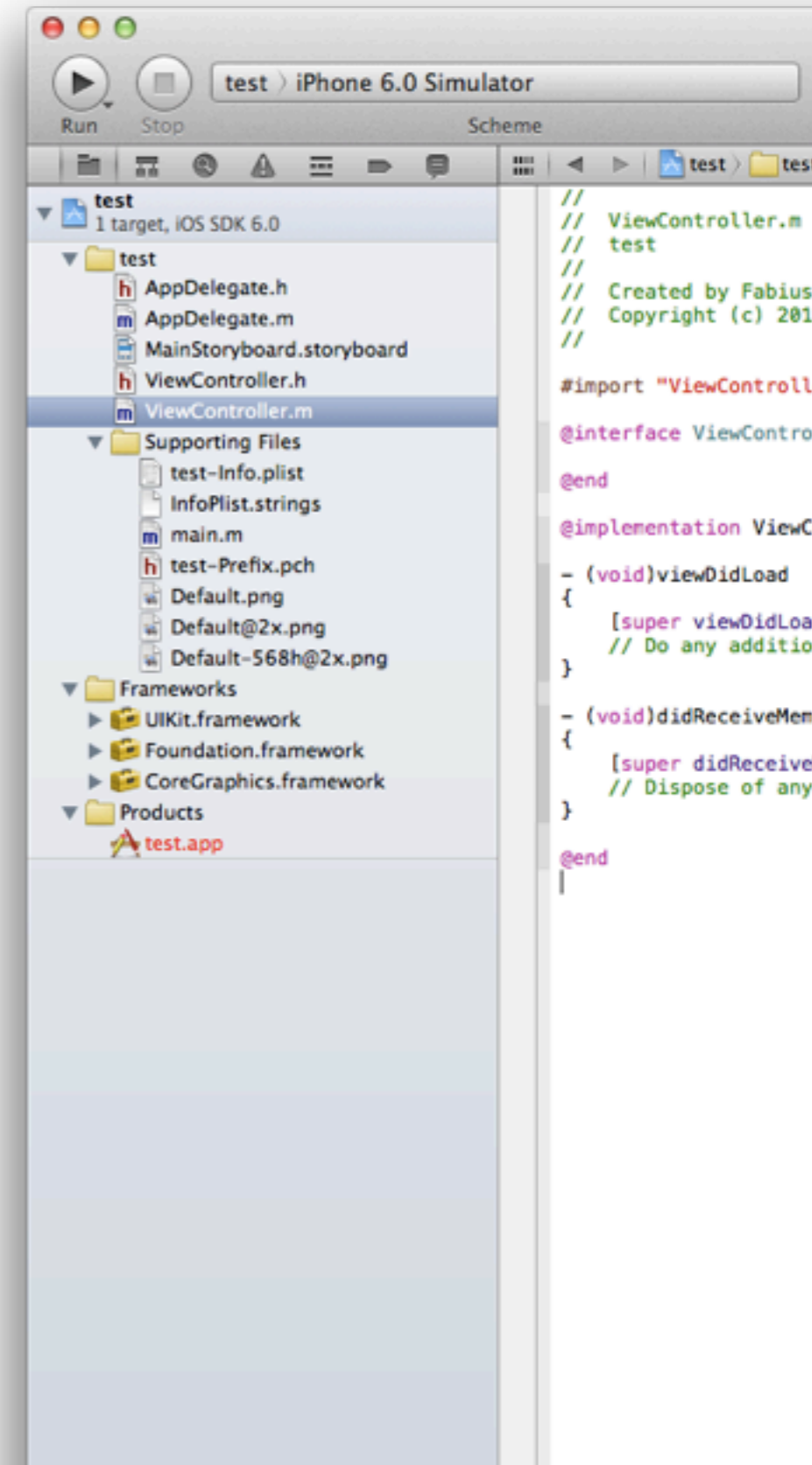
# XCode



# Contents of an XCode project

---

- Source code files (.h and .m)
- User interface files (.storyboard and .xib)
- Libraries (.framework)
- Resources, e.g. images (.png)
- App configuration file (Info.plist)



# Objective-C

---

- Language for programming iOS and Mac apps, also used by Apple to create much of OS X, iOS, APIs
- Superset of C
- Object-orientated

Short introduction: [https://developer.apple.com/library/mac/#referencelibrary/GettingStarted/Learning\\_Objective-C\\_A\\_Primer/index.html](https://developer.apple.com/library/mac/#referencelibrary/GettingStarted/Learning_Objective-C_A_Primer/index.html)

Detailed introduction: <https://developer.apple.com/library/mac/#documentation/Cocoa/Conceptual/ObjectiveC/Introduction/introObjectiveC.html>

# Elements of Objective-C

---

Java	Objective-C
MyClass.java	Header.h Implementation.m
Methods and method calls	Methods and messages *
Attributes, setters, getters	Properties, instance variables
Constructor	Initializer *
Interface	Protocol *
Garbage Collection	Automatic Reference Counting (ARC) *

\* Different terminology, but for us very similar to writing Java code

# Methods

---

- Definition (in .h):

```
- (void) doSomething;
```

```
- (void) doSomethingWithA: (NSString *) a  
andB: (NSString *) b;
```

---

- Implementation (in .m):

```
- (void) doSomething {  
    // do something  
}
```

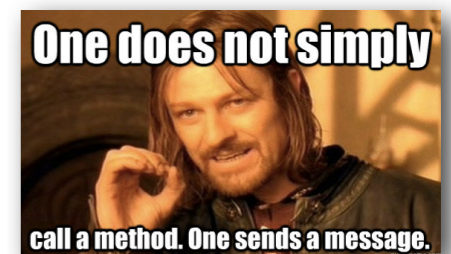
```
- (void) doSomethingWithA: (NSString *) a  
andB: (NSString *) b {  
    // do something with a and b  
}
```

---

- Method call ("message") (in .m):

```
[self doSomething];
```

```
NSString* a = @"a";  
NSString* b = @"b";  
[self doSomethingWithA:a andB:b];
```



# Instance Variables (“ivars”)

---

- Like private/protected attributes in Java
- Definition (in .h): `NSString* _name;`
- Use (in .m):  

```
_name = @"Max";  
labelText = _name;
```
- You don't have to use the underscore ( `_` ), but it's good practice. Otherwise you accidentally mix up ivars and properties (see next slide).



# Properties

---

- Auto-creation of an instance variable (protected) as well as a getter and setter (public)

- Definition (in .h):

```
@property(strong, nonatomic) NSString *name;
```

**strong/weak:** refers to ownership. Always use strong except for properties that point to a parent.

- Using getters (in .m):

```
NSString *labelText = self.name;  
labelText = [self name];
```

**nonatomic/atomic:** use nonatomic to avoid multi-threading issues.

- Using setters (in .m):

```
[self setName:@"Max"];  
self.name = @"Max";
```

**self.name:** this syntax does NOT access the variable itself. It's a getter/setter, just like the other syntax.

- Using the instance variable (in .m):

```
_name = @"Max";  
labelText = _name;
```

**\_name:** Use this instance variable in custom setters/getters and in init-methods only. In any other case, use the getter/setter.

# Object Initialization

---

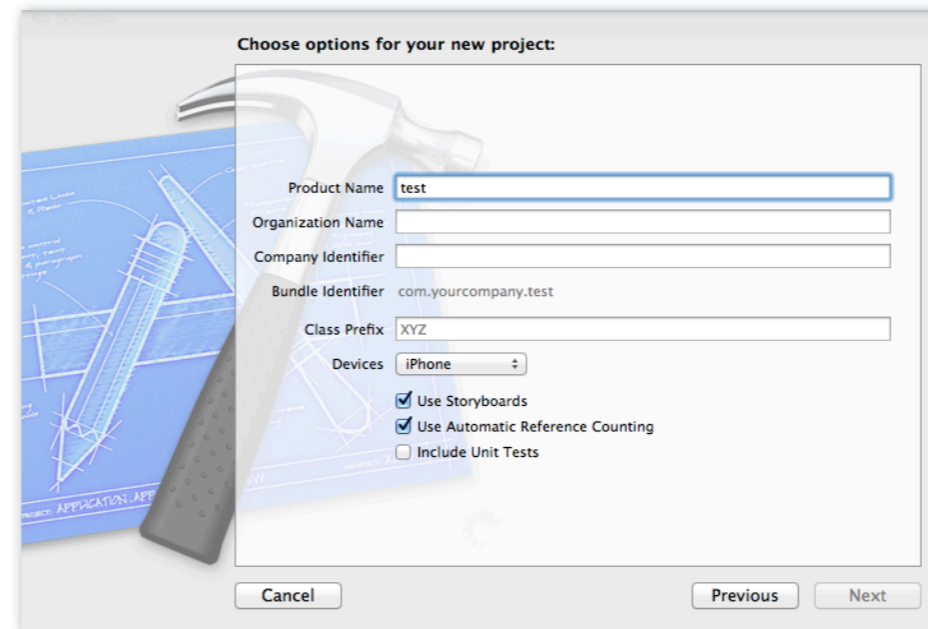
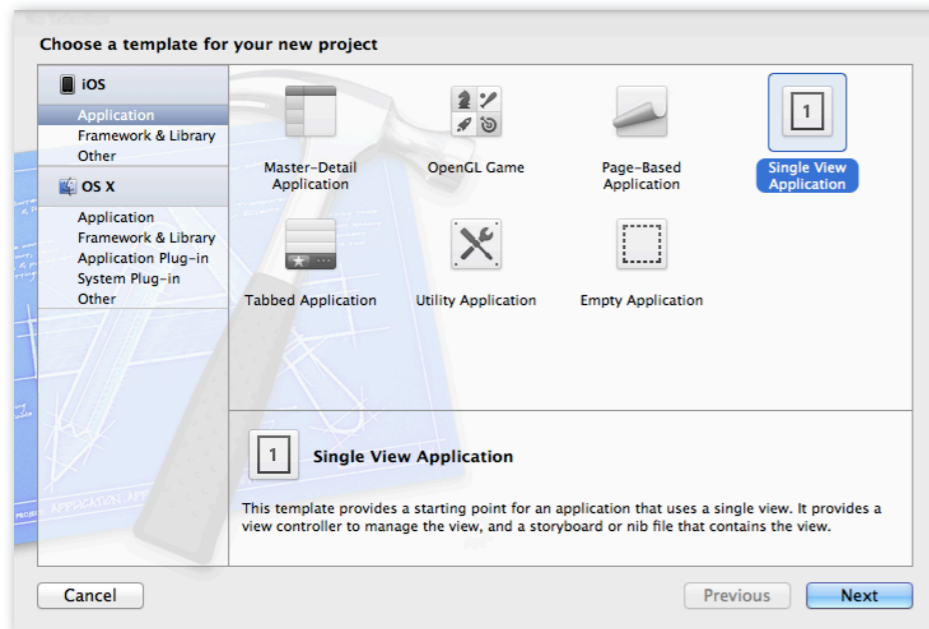
- Object: `MyClass *myObject = [[MyClass alloc] init];`
- Object with parameter: `MyClass *myObject = [[MyClass alloc] initWithParameter: parameter];`
- String: `NSString *hello = @"Hello";`  
`NSString *helloWorld = [NSString stringWithFormat:@"%@ World", hello];`
- Array: `NSArray *colors = @[@"Green", @"Red", @"Yellow"];`  
`NSMutableArray *mutableColors = @[@"Green", @"Red", @"Yellow"] mutableCopy];`



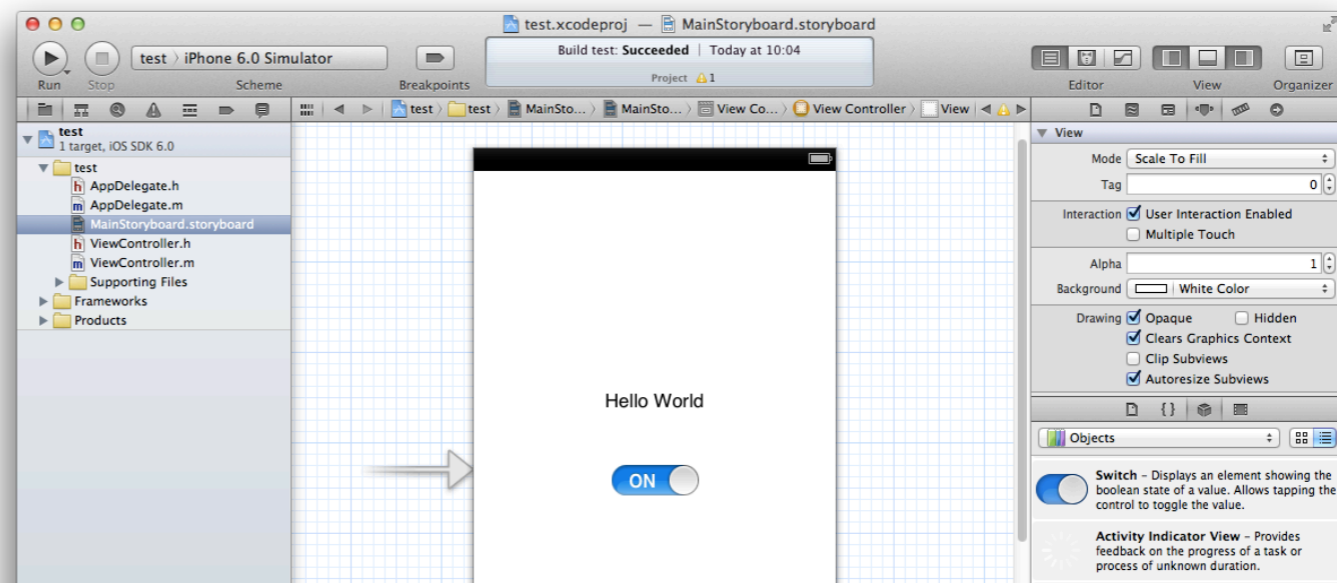
If your app doesn't work properly, make sure your objects aren't `nil`. THERE ARE NO NULL POINTER EXCEPTIONS - Less crashes, more confusion.

# Hello World

- New XCode Project: Single View Application




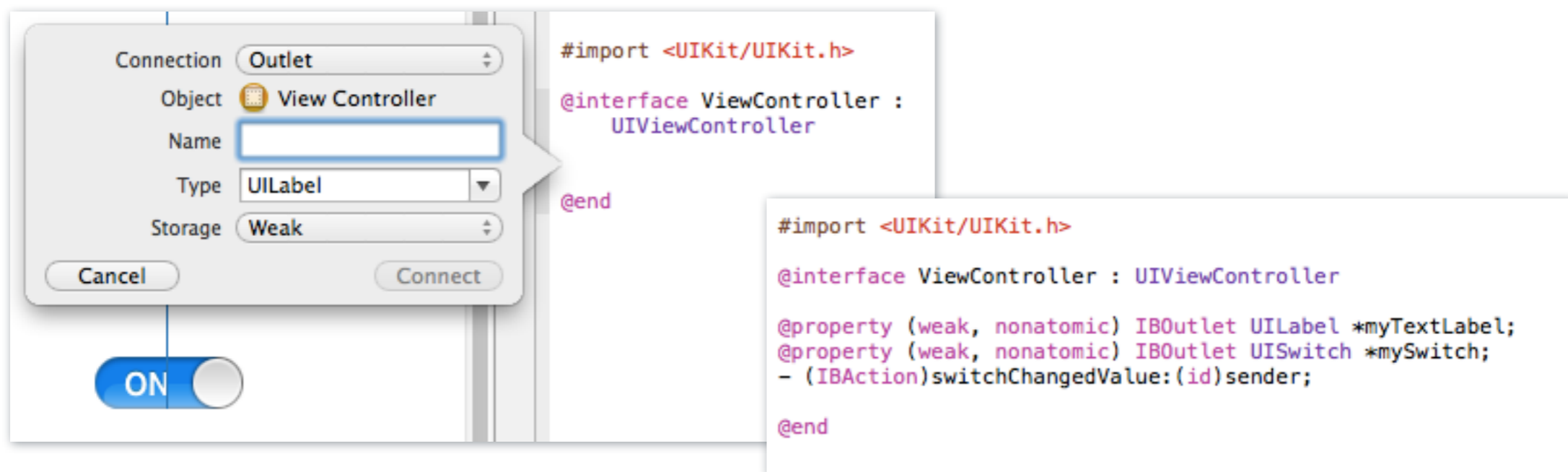
- In the storyboard, drag a text label and a switch onto the screen



# Hello World

---

- Open the assistant editor  and ctrl-drag the text label into ViewController.h. Enter a name and click Connect. *You now have access to the UI element in your code.* Do the same for the switch.
- Again, ctrl-drag the switch into the code. This time, select Action instead of Outlet. Enter a name and click Connect. *You now have a listener method that is called by the OS when the user changes the value of our switch.*



```
#import <UIKit/UIKit.h>

@interface ViewController : UIViewController

@end
```

```
#import <UIKit/UIKit.h>

@interface ViewController : UIViewController

@property (weak, nonatomic) IBOutlet UILabel *myTextLabel;
@property (weak, nonatomic) IBOutlet UISwitch *mySwitch;
- (IBAction)switchChangedValue:(id)sender;

@end
```

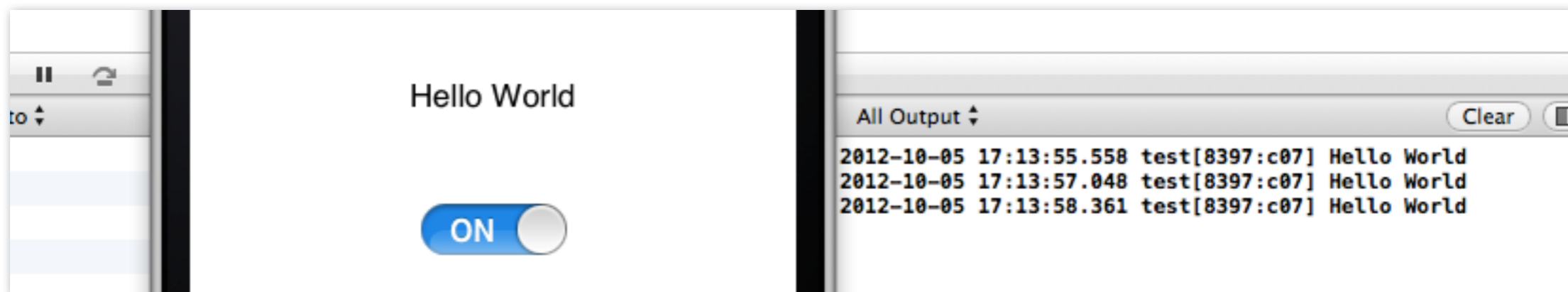
# Hello World

---

- Close the assistant editor and go to ViewController.m. Complete the IBAction method:

```
- (IBAction)switchChangedValue:(id)sender {  
    if (self.mySwitch.on) {  
        self.myTextLabel.text = @"Hello World";  
        NSLog(@"Hello World");  
    } else {  
        self.myTextLabel.text = @"";  
    }  
}
```

- Open the debug area and run the code.



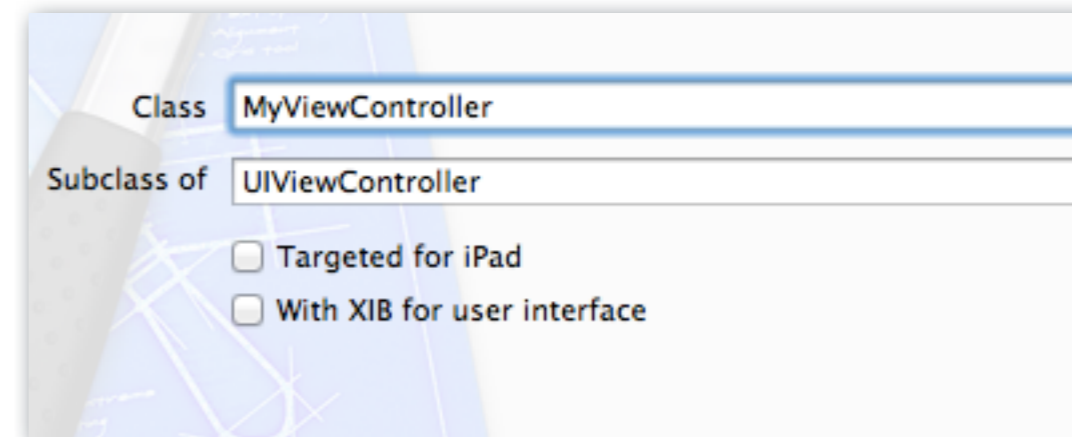
# UIViewController

---

- One of the most important classes in iOS programming
- You have to subclass UIViewController when creating a new screen
- Provides methods for managing the view hierarchy throughout its life cycle and for reacting to events (also great for debugging), e.g.

```
- viewDidLoad:  
- viewWillAppear:  
- viewDidAppear:  
- viewWillDisappear:  
- viewDidDisappear:  
- (void)willRotateToInterfaceOrientation:(UIInterfaceOrientation)toInterfaceOrientation  
duration:(NSTimeInterval)duration;
```

- For more see [http://developer.apple.com/library/ios/#documentation/uikit/reference/UIViewController\\_Class/Reference/Reference.html](http://developer.apple.com/library/ios/#documentation/uikit/reference/UIViewController_Class/Reference/Reference.html)



# App Delegate

---

- Every app must have an App Delegate.
- Provides methods for managing the app throughout its life cycle (also great for debugging), e.g.
  - `application:didFinishLaunchingWithOptions:`
  - `applicationDidBecomeActive:`
  - `applicationDidEnterBackground:`
  - `applicationWillEnterForeground:`
  - `applicationWillTerminate:`
- **For more see:** [http://developer.apple.com/library/ios/#documentation/uikit/reference/UIApplicationDelegate\\_Protocol/Reference/Reference.html](http://developer.apple.com/library/ios/#documentation/uikit/reference/UIApplicationDelegate_Protocol/Reference/Reference.html)
- There are lots of protocols (often named Delegate), e.g. for managing the keyboard, table views, date pickers.

# Top 5 Resources

---

According to last semester's students:

- **Official documentation:** <https://developer.apple.com/library/ios>
- **Tutorials:** <http://www.raywenderlich.com/tutorials>
- **Solutions to specific problems:** Google + Stackoverflow
- **Book:** "iOS Programming: The Big Nerd Ranch Guide" by Joe Conway and Aaron Hillegass
- **Developer videos:** <https://developer.apple.com/videos/>



# Assignment 1

---

- Individual assignment
- Get to know XCode and Objective-C
- Due next Thursday 12:00, upload to Uniworx
  
- Questions?