Introduction to Apple's iOS Mobile Development

Adam Nagy – Autodesk

SD6000 Learn how quick and easy it is to start programming on iOS for Apple® iPhone®, iPad®, and iPod®. I create from scratch a simple iOS application with traditional UI elements like buttons, tabs, and lists. I then deploy this app on an iPad and show it running. Finally, I show how you can debug and test an iOS application. During these demonstrations, you learn where you need to go and what you need to do to start programming in iOS. I also show you how you can use GLKit to draw 3D graphics and how to access REST web services from your iOS device.

Learning Objectives

At the end of this class, you will be able to:

- Explain where to start to develop applications for iOS devices
- Create simple apps with a simple UI for iOS
- Show how to access REST web services from iOS
- See how to use GLKit and OpenGL ES

About the Speaker

Adam Nagy joined Autodesk back in 2005 and has been providing programming support, consulting, training and evangelism to external developers. He started his career in Budapest working for a Civil Engineering CAD software company, then worked for Autodesk in Prague for 3 years and now lives in South England, UK.

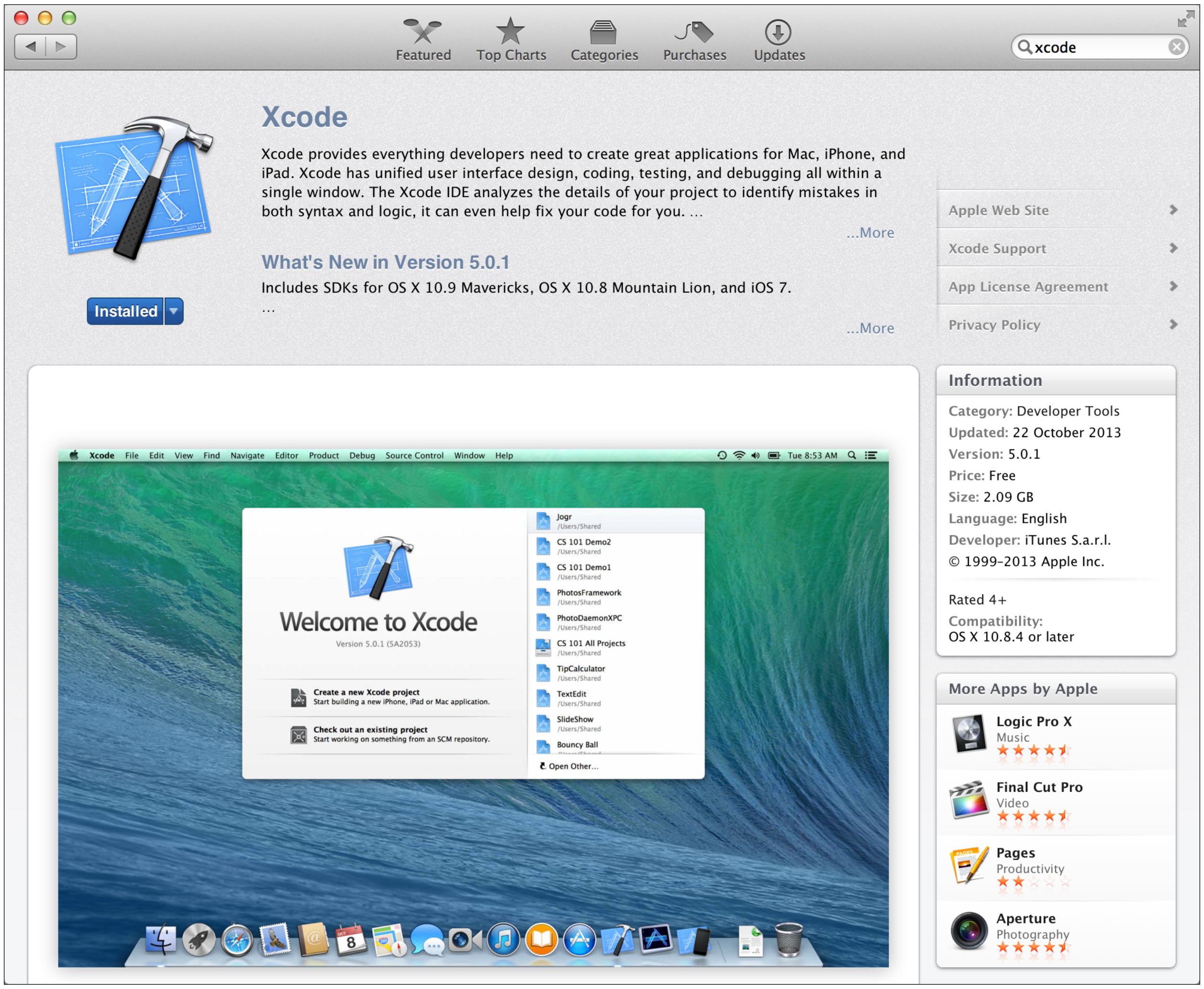
At the moment focusing on the MFG products, plus cloud and mobile related technologies.

Adam has a degree in Software Engineering and has been working in that area since even before leaving college.

adam.nagy@autodesk.com

Start to develop applications for iOS devices

First of all you'll need a Mac and Xcode. The latest version of Xcode that is being used during this presentation too is now accessible through the **App Store** application on the Mac. So it's quite easy to install it.



Xcode

Xcode provides everything developers need to create great applications for Mac, iPhone, and iPad. Xcode has unified user interface design, coding, testing, and debugging all within a single window. The Xcode IDE analyzes the details of your project to identify mistakes in both syntax and logic, it can even help fix your code for you. …

...More

What's New in Version 5.0.1

Includes SDKs for OS X 10.9 Mavericks, OS X 10.8 Mountain Lion, and iOS 7.
…

...More

Information

Category: Developer Tools
Updated: 22 October 2013
Version: 5.0.1
Price: Free
Size: 2.09 GB
Language: English
Developer: iTunes S.a.r.l.
© 1999-2013 Apple Inc.

Rated 4+
Compatibility:
OS X 10.8.4 or later

More Apps by Apple

Logic Pro X
Music

Final Cut Pro
Video

Pages
Productivity

Aperture
Photography

Xcode will also install **iOS Simulator** so you'll be able to test most of the functionality of your projects even without having an Apple Developer account or an iPad/iPhone.

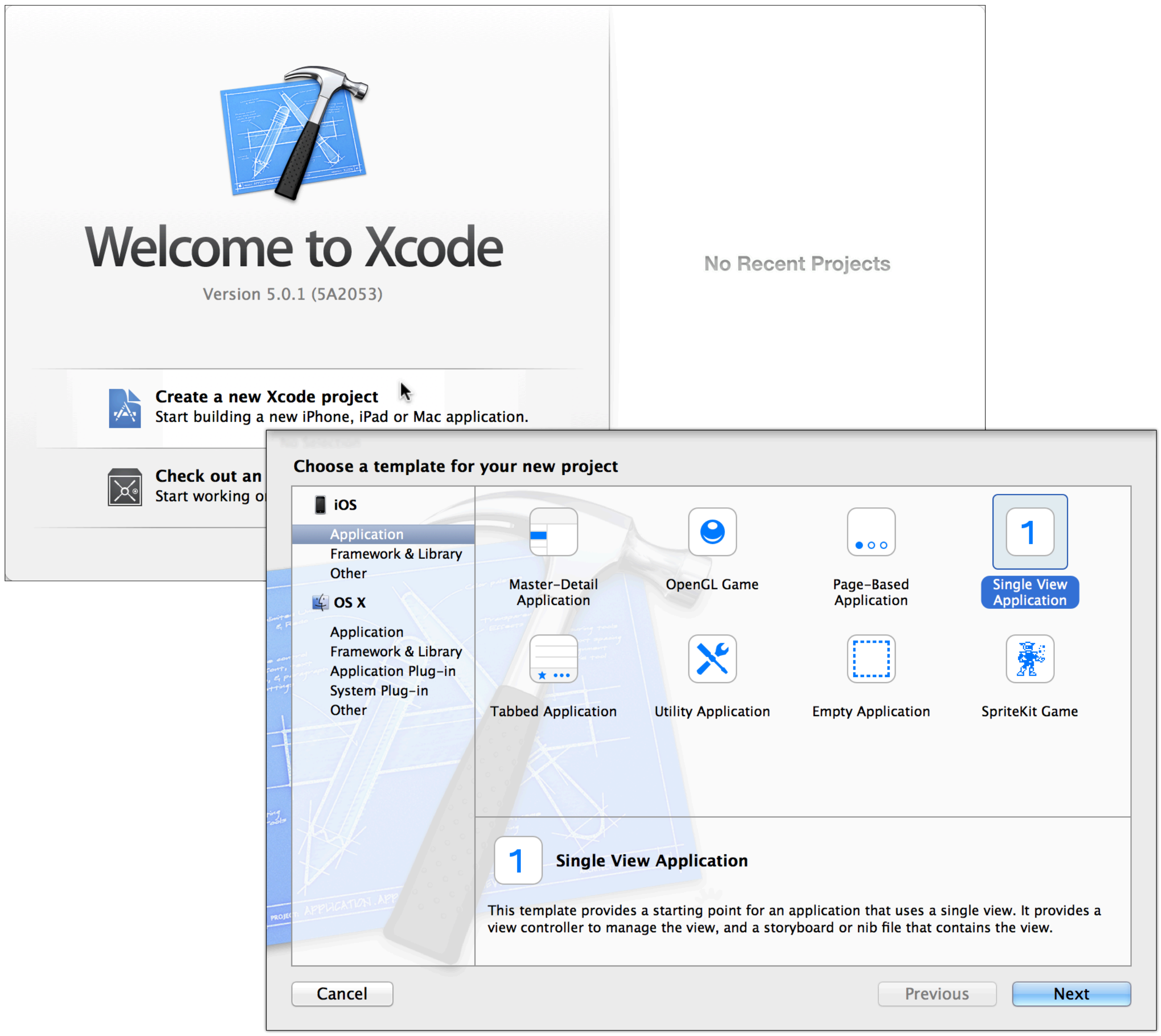
If you want to test how things work on a physical device, then you'll need to get an Apple Developer account: https://developer.apple.com

Once you have that then you need to register your device on the **iOS Provisioning Portal**. Then the simplest thing is to create an **App ID** with a wild-card character (*) so that it can be used for testing multiple iOS applications.

The following site provides detailed information on this including how to distribute your application later on to testers:

https://developer.apple.com/library/mac/documentation/IDEs/Conceptual/AppDistributionGuide/Introduction/Introduction.html

When creating a new project they tend to contain not only a skeleton but also sample functionality. This means that after creating a new project, you can run it straight away and play around with it. This way you'll have a better understanding of what's going on when starting to look inside the code.



Single View Application

This template provides a starting point for an application that uses a single view. It provides a view controller to manage the view, and a storyboard or nib file that contains the view.

Though you can use e.g. C and C++ as well inside your project the best option is probably to stick with the native Objective-C language, which is used by the project templates too. It might take a bit of getting used to, which the following should help with:

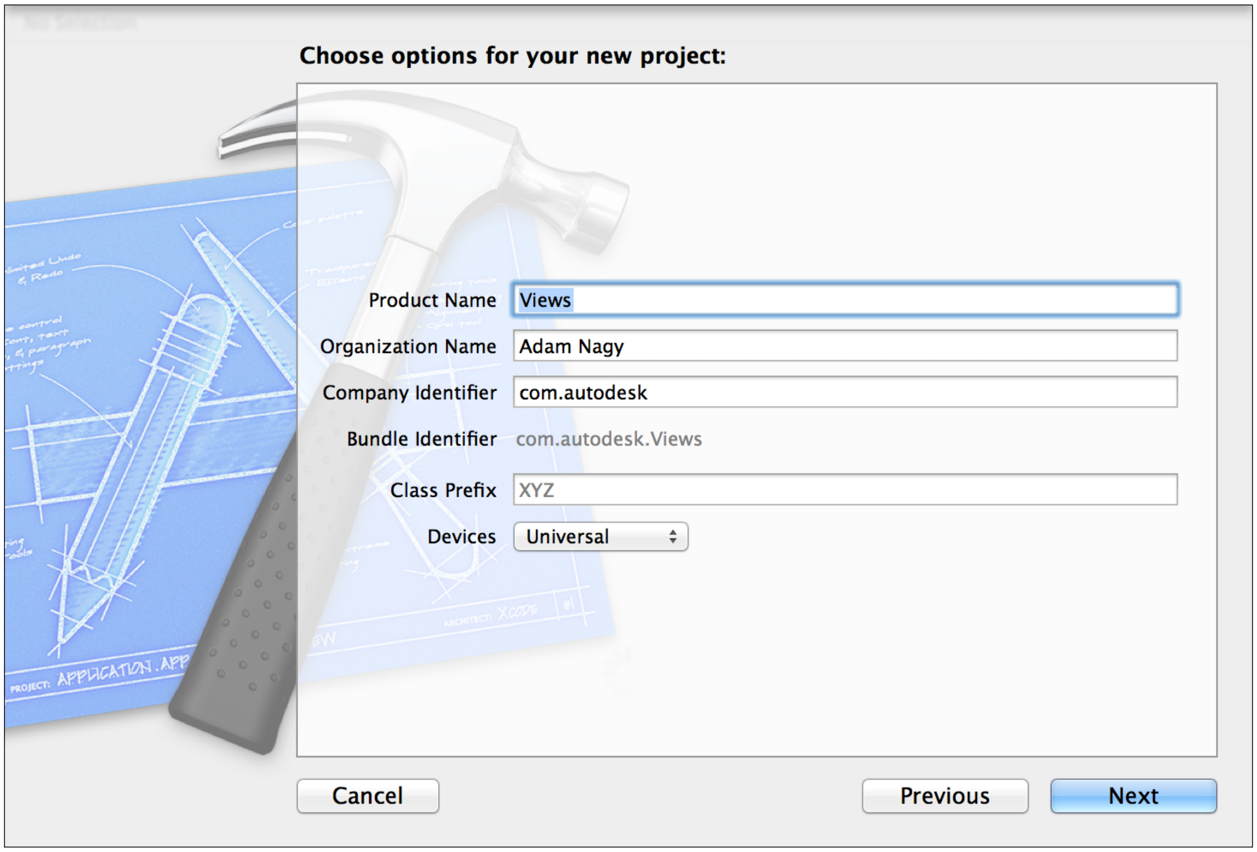
https://developer.apple.com/library/mac/#documentation/Cocoa/Conceptual/ObjectiveC/Introduction/introObjectiveC.html

There is also a nice overview of all aspects of iOS programming here:

http://developer.apple.com/library/ios/#referencelibrary/GettingStarted/RoadMapiOS/chapters/Introduction.html

Create simple apps with a simple UI for iOS

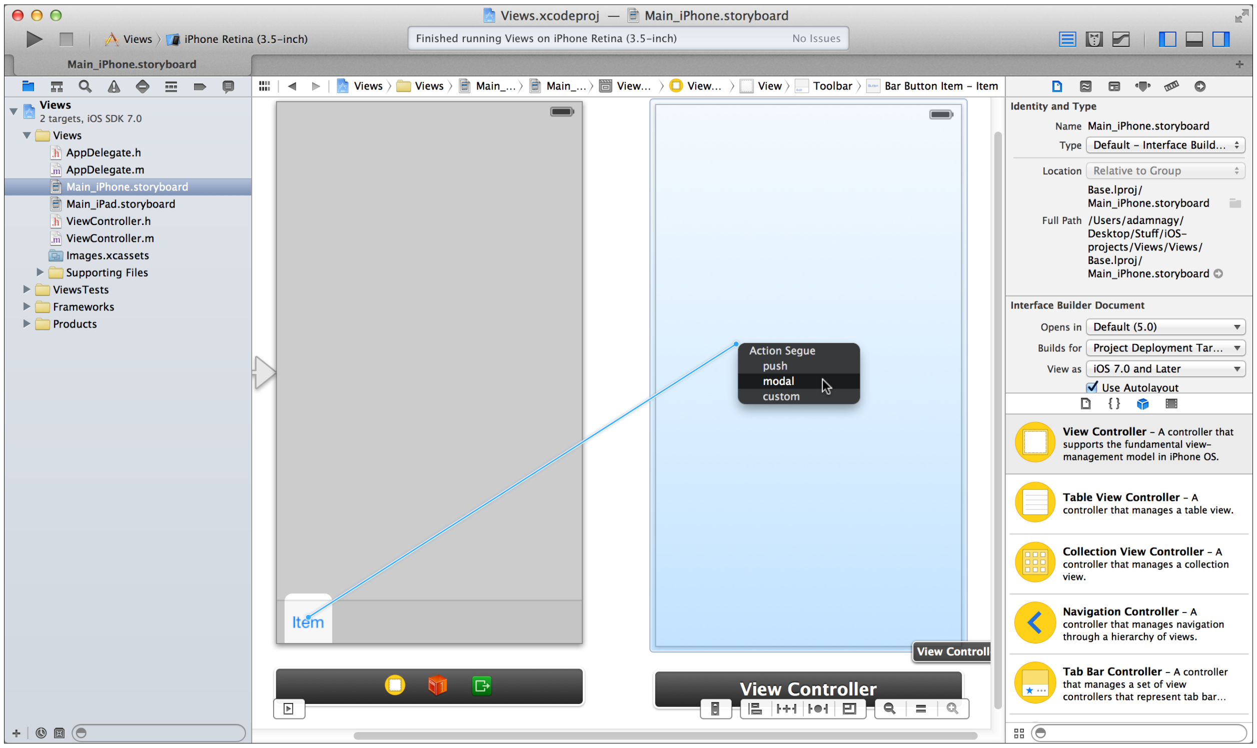
To get started the easiest is to just create projects using the various templates and try to understand what is going on inside them. Make sure **Use Storyboards** is selected when creating your project – in case of the latest Xcode all new projects use them.


Choose options for your new project:

Product Name	Views
Organization Name	Adam Nagy
Company Identifier	com.autodesk
Bundle Identifier	com.autodesk.Views
Class Prefix	XYZ
Devices	Universal

Storyboards make it much easier to visualize and understand how the various views of your application will change from one to the other. They enable you to specify certain transitions from one view to the other directly in the visual resource editor without having to write a single code.

You can do that using simple Ctrl+Drag from the control that should trigger the change to the view that should be shown:



Finished running Views on iPhone Retina (3.5-inch) No Issues

Identity and Type
Name Main_iPhone.storyboard
Type Default – Interface Build...
Location Relative to Group
Base.lproj/Main_iPhone.storyboard
Full Path /Users/adamnagy/Desktop/Stuff/iOS-projects/Views/Views/Base.lproj/Main_iPhone.storyboard

Interface Builder Document
Opens in Default (5.0)
Builds for Project Deployment Tar...
View as iOS 7.0 and Later
✓ Use Autolayout

View Controller – A controller that supports the fundamental view-management model in iPhone OS.

Table View Controller – A controller that manages a table view.

Collection View Controller – A controller that manages a collection view.

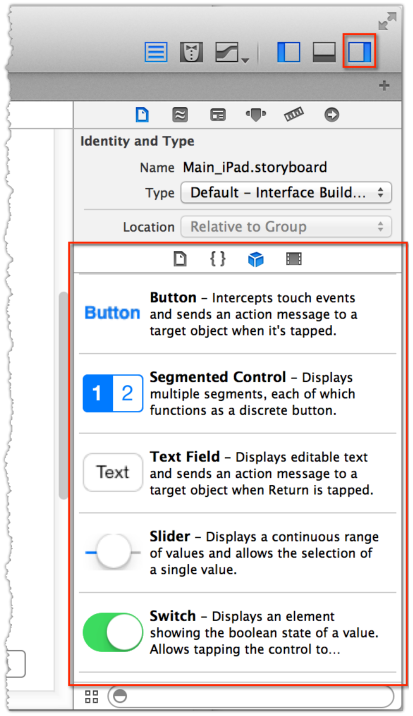
Navigation Controller – A controller that manages navigation through a hierarchy of views.

Tab Bar Controller – A controller that manages a set of view controllers that represent tab bar...

More information on Storyboards:

http://developer.apple.com/library/mac/#documentation/General/Conceptual/Devpedia-CocoaApp/Storyboard.html

Once you created an iOS application, e.g. a Single View application, you can click on the *.storyboard file of the project and start adding controls to your views via drag&drop


Identity and Type
Name Main_iPad.storyboard
Type Default – Interface Build...
Location Relative to Group

Button – Intercepts touch events and sends an action message to a target object when it's tapped.

Segmented Control – Displays multiple segments, each of which functions as a discrete button.

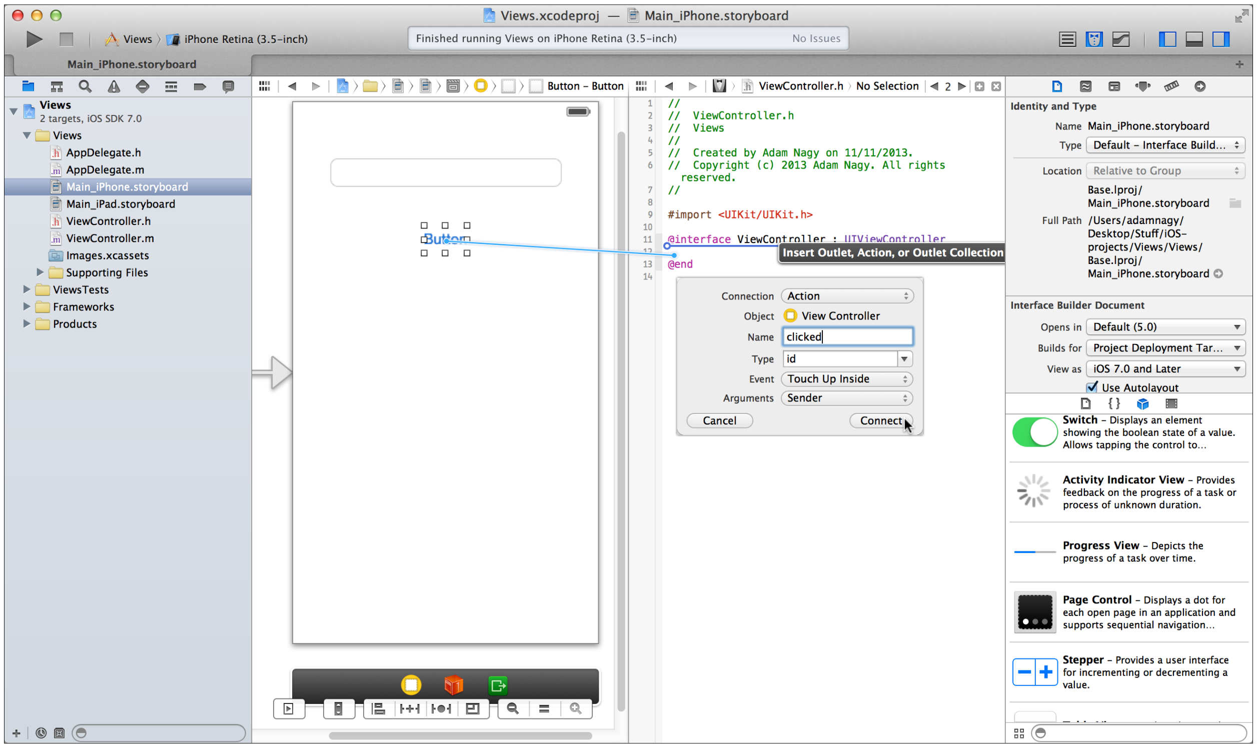
Text Field – Displays editable text and sends an action message to a target object when Return is tapped.

Slider – Displays a continuous range of values and allows the selection of a single value.

Switch – Displays an element showing the boolean state of a value. Allows tapping the control to...

Once you have your controls on the views then you also need to hook them up with the code. You either want a variable in your code that represents the given control or want to implement the function (action) that should be called when e.g. a button is clicked.

Here as well you can use Ctrl+Drag to speed things up, but for that you need to switch your Xcode environment so that it shows two resources (the storyboard and the code file) at the same time.



```
//
//  ViewController.h
//  Views
//
//  Created by Adam Nagy on 11/11/2013.
//  Copyright (c) 2013 Adam Nagy. All rights
 reserved.
//

#import <UIKit/UIKit.h>

@interface ViewController : UIViewController

@end
```

Connection	Action
Object	View Controller
Name	clicked
Type	id
Event	Touch Up Inside
Arguments	Sender

Cancel   Connect

Identity and Type

Name	Main_iPhone.storyboard
Type	Default – Interface Build…
Location	Relative to Group
	Base.lproj/Main_iPhone.storyboard
Full Path	/Users/adamnagy/Desktop/Stuff/iOS-projects/Views/Views/Base.lproj/Main_iPhone.storyboard

Interface Builder Document

Opens in	Default (5.0)
Builds for	Project Deployment Tar…
View as	iOS 7.0 and Later

[✓] Use Autolayout

Switch – Displays an element showing the boolean state of a value. Allows tapping the control to…

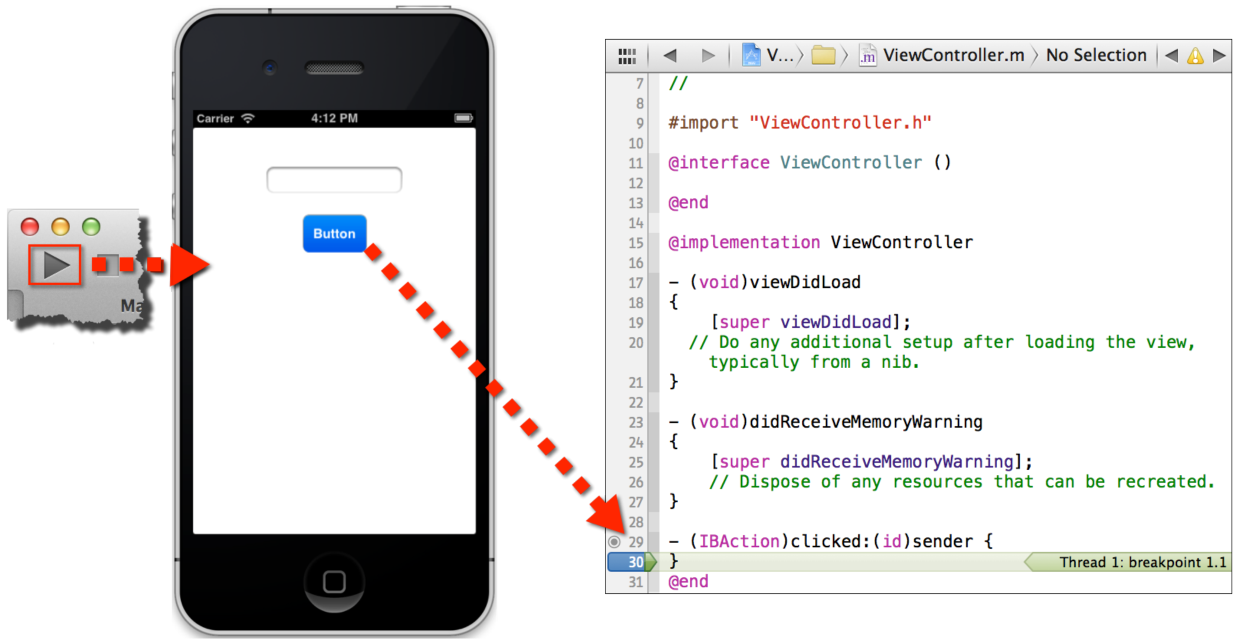
Activity Indicator View – Provides feedback on the progress of a task or process of unknown duration.

Progress View – Depicts the progress of a task over time.

Page Control – Displays a dot for each open page in an application and supports sequential navigation…

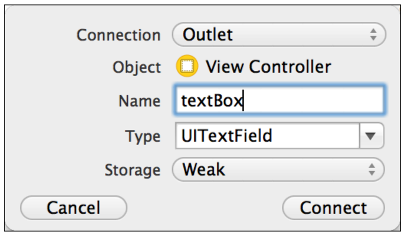
Stepper – Provides a user interface for incrementing or decrementing a value.

This inserts both the function declaration in the *.h file plus the implementation part in the *.m file. If you place there a breakpoint, click debug and then click the button then that part will be hit:



```
//

#import "ViewController.h"

@interface ViewController ()

@end

@implementation ViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
  // Do any additional setup after loading the view,
    typically from a nib.
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

- (IBAction)clicked:(id)sender {
}
@end
```

You can also hook up the Text Field in a similar fashion, but in that case choose the Outlet option. In this case I name it **textBox**.

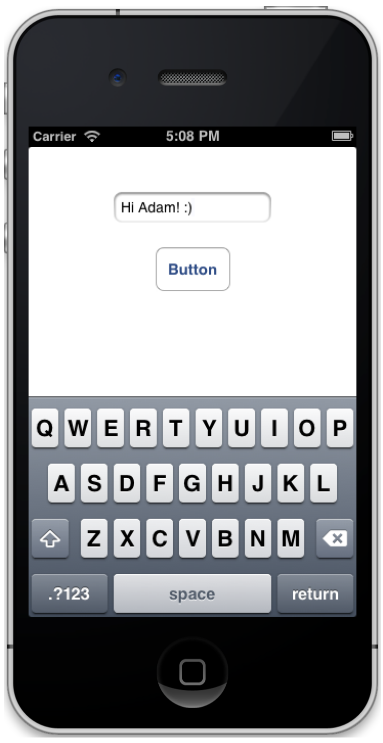


Connection: Outlet
Object: View Controller
Name: textBox
Type: UITextField
Storage: Weak

Now inside the button's **clicked** action I can add some code that will fill the text field with some string:

```
- (IBAction)clicked:(id)sender {
  // Combine the current text in the Text Field
  // with "Hi!"
  NSString * txt =
    [NSString stringWithFormat:@"Hi %@! :)",
      self.textBox.text];

  self.textBox.text = txt;
}
```

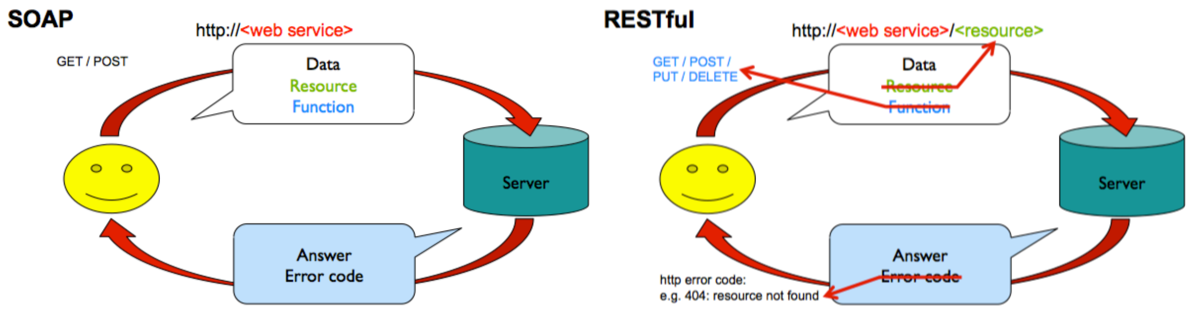
Now if we run the sample again, type in our name in the text field and then press the button, then this will happen:



Access REST web services from iOS

REST stands for REpresentational State Transfer. It is a style of software architecture used for distributed system that consists of clients and servers. A REST based web service is using HTTP for communication and so takes advantage of the HTTP verbs as well: GET, POST, PUT and DELETE.

REST is a lightweight alternative to other types of web services like SOAP. Instead of having all the information inside the SOAP envelope we only need to provide the URL of the resource then use HTTP verbs to define what we want to do with the resource. It is much more loosely coupled and is easier to test as well than its alternatives.



HTTP stands for Hyper Text Transfer Protocol that the Internet is based on, and so you take advantage of it day by day when e.g. surfing the net.

The web is full of resources and we use URL's (Unified Resource Locator) to identify them. When you type in a web address, you are basically specifying the id of a given resource. The resource could be a text file, a zip file, an image file, etc, but most of the time you'll run into html pages, which are basically like text documents that also include instructions for how they should be presented.

In case of a web browser, when you are opening a webpage, the browser send an HTTP GET request with the given URL or resource identifier. When trying to access a resource through a REST API, then you'll do the same: create an HTTP request, specify the URL of the resouce and the HTTP verb you want to use:

```
NSError * err = nil;

NSURL * url = [NSURL URLWithString: @"http://<resource id>"];
NSURLRequest * req = [NSURLRequest requestWithURL:url];

NSData * data = [NSURLConnection sendSynchronousRequest:req
  returningResponse:nil error:&err];
```

Depending on what data and in what format is received we can handle it in different ways. Most REST API's send back (or at least provide that option) in Json format. JSON stands for JavaScript Object Notation. This is a very popular format for data exchange between programs across the internet and it's a much more compact format than xml. More information on it at http://www.json.org/xml.html

If you need to iterate Json data then you can use `NSJSONSerialization`, which converts it into a tree of `NSArray` and `NSDictionary` objects:

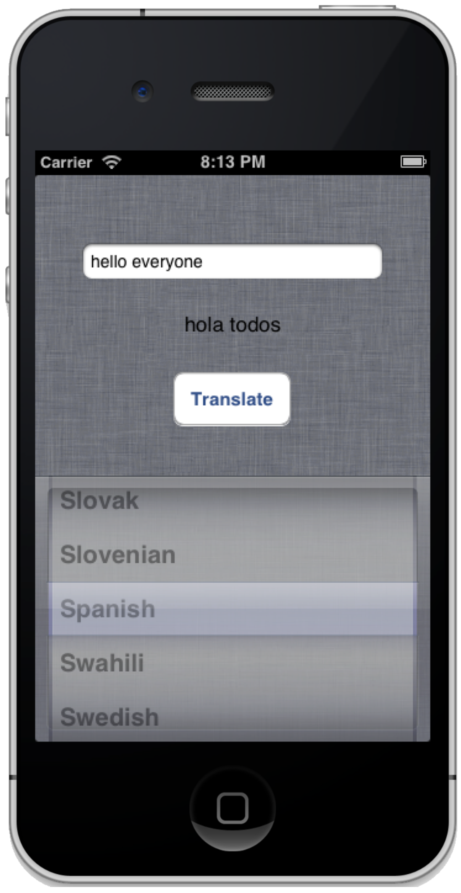
```
NSDictionary * json = [NSJSONSerialization
  JSONObjectWithData:data
  options:NSJSONReadingMutableContainers
  error:&err];
```

If you need to parse an xml string for information then you can create you own xml parser class and use it in combination with `NSXMLParser`:

```
// Create and init NSXMLParser object
NSXMLParser * nsXmlParser = [[NSXMLParser alloc] initWithData:data];

// Create and init our delegate
XmlParser * parser = [XmlParser alloc];

// Set delegate
[nsXmlParser setDelegate:parser];

// Parsing...
BOOL success = [nsXmlParser parse];
```

The Translator sample shows both, so you can have a look at that.

Source code:
http://adndevblog.typepad.com/cloud_and_mobile/2012/12/using-rest-translator-service-from-ios.html



hello everyone

hola todos

See how to use GLKit and OpenGL ES

OpenGL ES is a subset of OpenGL and enables you to draw 3d objects on the screen with all sorts of sophisticated mapping and lighting settings. It's not the easiest thing to use and so Apple created a helper kit, GLKit, that takes away some of the complexity of setting things up for 3d rendering on your iOS device.

This kit contains GLKView that you can use inside your project just like any other views and implements drawInRect() where you can draw your graphics. It also includes GLKViewController which implements an OpenGL ES rendering loop that keeps calling update() where you can update your graphics. It's useful if e.g. you want to show animations.

The easiest to get started is to create a project based on the "OpenGL Game" project template. This will contain lots of functionality, so it might be too much to understand the basics. You could have a look at articles like this one, which starts with en empty project and then takes you through the steps of adding the graphics: http://www.raywenderlich.com/5223/beginning-opengl-es-2-0-with-glkit-part-1

If you start with an empty project or with a "Single View Application" then you will need GLKit.Framework and OpenGLES.Framework added to the project then add a GLKView to your storyboard.

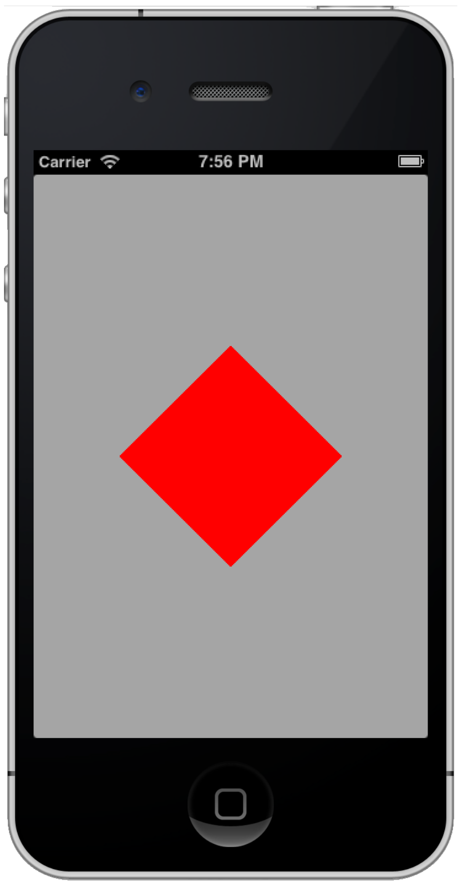
You can set things up when the view gets loaded like so:

```
- (void)viewDidLoad
{
  [super viewDidLoad];

  // Initialize the system

  GLKView * view = (GLKView *)self.view;
  view.context = [[EAGLContext alloc]
initWithAPI:kEAGLRenderingAPIOpenGLES2];

  // You can also set this in the storyboard
  view.delegate = self;

  [EAGLContext setCurrentContext:view.context];

  self.effect = [[GLKBaseEffect alloc] init];

  float aspect = fabsf(self.view.bounds.size.width /
self.view.bounds.size.height);
  GLKMatrix4 projectionMatrix =
GLKMatrix4MakePerspective(GLKMathDegreesToRadians(65.0f), aspect,
0.1f, 10.0f);
  self.effect.transform.projectionMatrix = projectionMatrix;
  self.effect.transform.modelviewMatrix = GLKMatrix4MakeLookAt(
    0, 0, 2,  // eye
    0, 0, 0,  // center
    0, 1, 0); // up vector

  // Or set "Enable setNeedsDisplay" e.g. in storyboard
  [view display];
}
```

When your ViewController class is hooked up to the GLKView, then your `drawInRect` implementation will be called where you can draw the graphics. The following code will draw a rectangle turned 45 degrees.

```
- (void)glkView:(GLKView *)view drawInRect:(CGRect)rect
{
  // Set background color

  glClearColor(0.65f, 0.65f, 0.65f, 1.0f);

  glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

  // Render the object with GLKit

  glEnable(GL_DEPTH_TEST);
  glEnable(GL_CULL_FACE);
  glDepthFunc(GL_LEQUAL);

  GLfloat vertices[] = {
    0.0, 0.5, 0.0,
   -0.5, 0.0, 0.0,
    0.5, 0.0, 0.0,
    0.0, -0.5, 0.0,
    0.5, 0.0, 0.0,
   -0.5, 0.0, 0.0};

  self.effect.constantColor = GLKVector4Make(1, 0, 0, 1);

  [self.effect prepareToDraw];

  // 1: number of float per vertex
  // 2: type
  // 3: distance between vertices
  // 4: pointer to vertices
  glVertexAttribPointer(GLKVertexAttribPosition, 3, GL_FLOAT, 0, 0,
vertices);

  // Enable its use
  glEnableVertexAttribArray(GLKVertexAttribPosition);

  // 1: type
  // 2: start index
  // 3: number of vertices
  glDrawArrays(GL_TRIANGLES, 0, 6);
}
```

Good overview of how you should organize your data to be used by OpenGL:
http://developer.apple.com/library/ios/#documentation/3DDrawing/Conceptual/OpenGLES_ProgrammingGuide/TechniquesforWorkingwithVertexData/TechniquesforWorkingwithVertexData.html



New programming language: Swift

Apple came out with a new programming language back in June, but it only reached release state at the end of October this year (2014) along with the release of the new OS X version Yosemite. This language looks much closer to other popular languages like C#, C++ or Java. This should make it much easier for developers to get started with iOS programming.

From my point of view the most important thing is actually not the language itself, but the libraries that are available for development. That will not change here. So if you write a new application for iOS or Max OS the same functionalities will be available both in Objective-C and Swift.

One cool new feature that is coming with this language is playgrounds, which make it much easier to get familiar with the language and study how it behaves:
https://developer.apple.com/swift/

This new language is only available in Xcode 6 1:
https://itunes.apple.com/us/app/xcode/id497799835?ls=1&mt=12