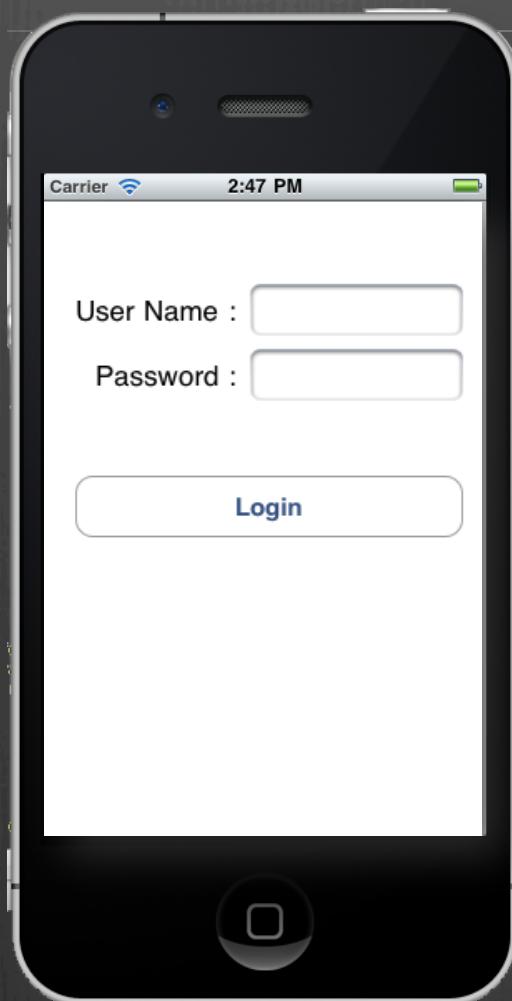


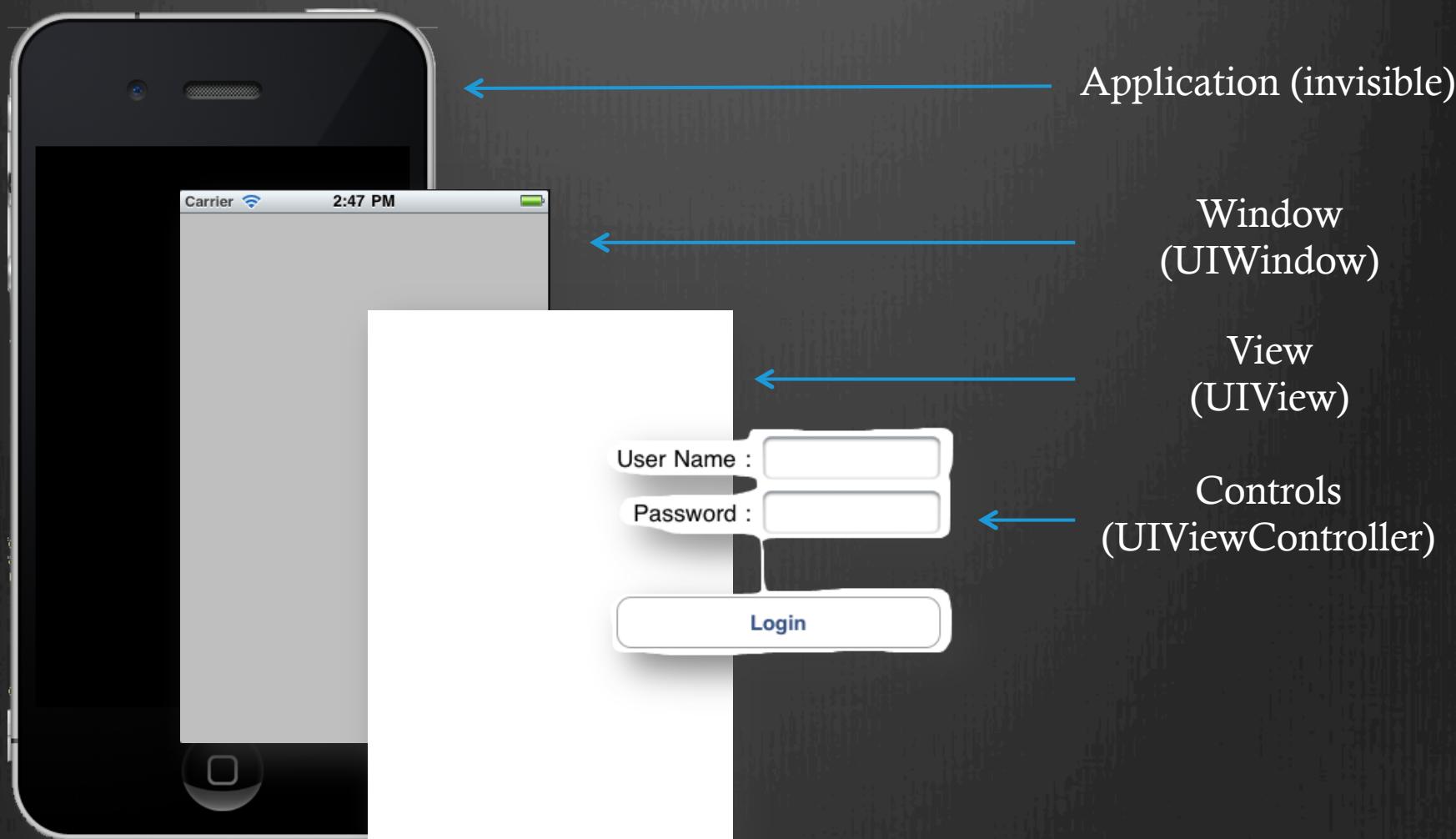
Chapter 5

Understand iOS Application & Views

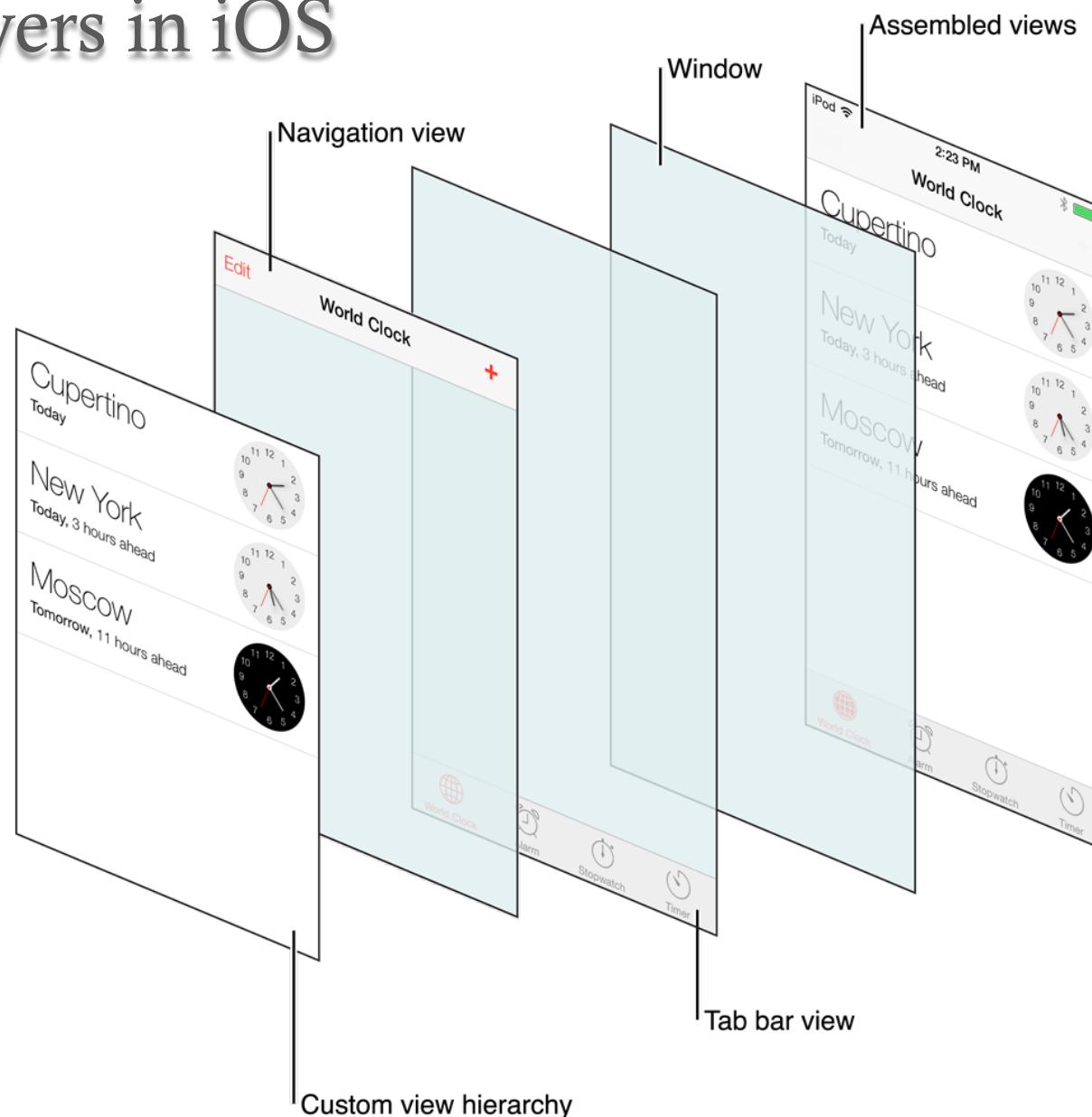
View Layers in iOS App



View Layers in iOS



View Layers in iOS



See iOS 7 Design Resources at <http://goo.gl/QdIT85>

iOS 6 to iOS 7 Design Transition

Carrier 6:05 AM

By Name

A	B
89 Ac	Actinium
13 Al	Aluminum
95 Am	Americium
51 Sb	Antimony
18 Ar	Argon
33 As	Arsenic
85 At	Astatine
B	
56	

Zinc Number Symbol State

iOS 6

Carrier 6:06 AM

By Name

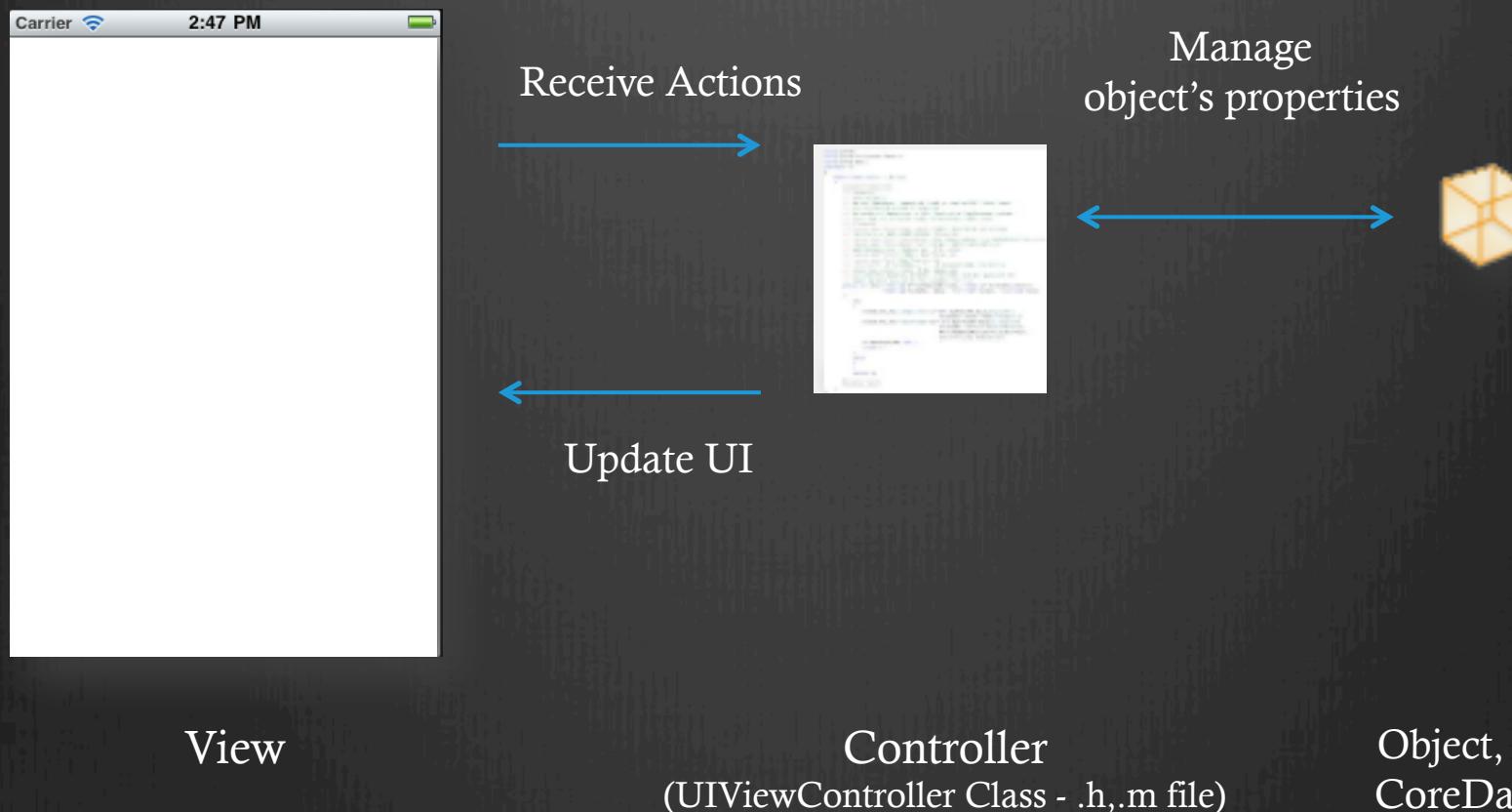
A	B
89 Ac	Actinium
13 Al	Aluminum
95 Am	Americium
51 Sb	Antimony
18 Ar	Argon
33 As	Arsenic
85 At	Astatine
B	
56	

Zinc Number Symbol State

iOS 7

<http://goo.gl/afJ4fa>

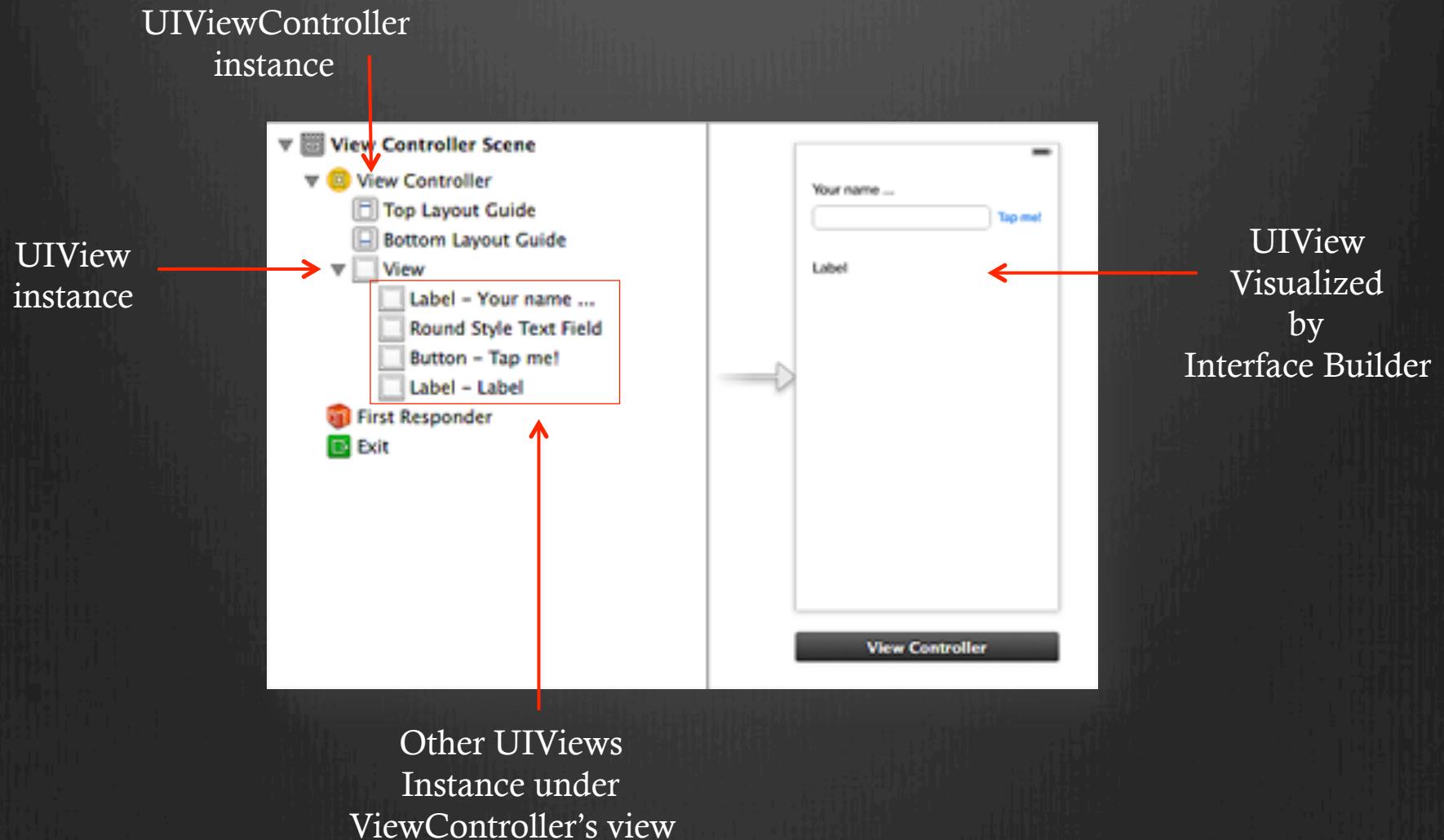
MVC Pattern



UIView & UIViewController

- ⊕ UIViewController เป็น class หลักที่ทำหน้าที่ควบคุมและจัดการ UI ของ iOS App เมื่อเราจะใช้ controller เราจะสร้าง sub class ของ UIViewController ขึ้นมาเป็น controller class ของเราเองก่อน
- ⊕ หน้าจอที่เราเห็น ใน interface builder (Main.storyboard) คือ instance ของ sub class ของ “UIView” โดย Interface Builder จะทำการ visualize ขึ้นมาเป็นภาพเพื่อให้เราสามารถออกแบบ UI ด้วยการ drag & drop ได้โดยง่าย
- ⊕ ใน class ของ UIViewController นั้นจะมี instance ของ UIView ประกาศอยู่ และเช่นเดียวกัน ใน UIView เองก็จะมี instance ของ UIViewController ประกาศไว้เช่นเดียวกัน
- ⊕ object ของ UIView และ UIViewController สามารถทำงานได้อย่างอิสระ แต่เมื่อไหร่ที่เราต้องการควบคุมการทำงานของ UIView เราต้องใช้ UIViewController ไปควบคุมการทำงานและแสดงผล
- ⊕ เมื่อเราระบุว่า class ไหนเป็น UIViewController ของ UIView, Interface Builder จะทำการ binding instance ของทั้ง 2 class ให้เราอัตโนมัติ

UIView & UIViewController



View + Controller



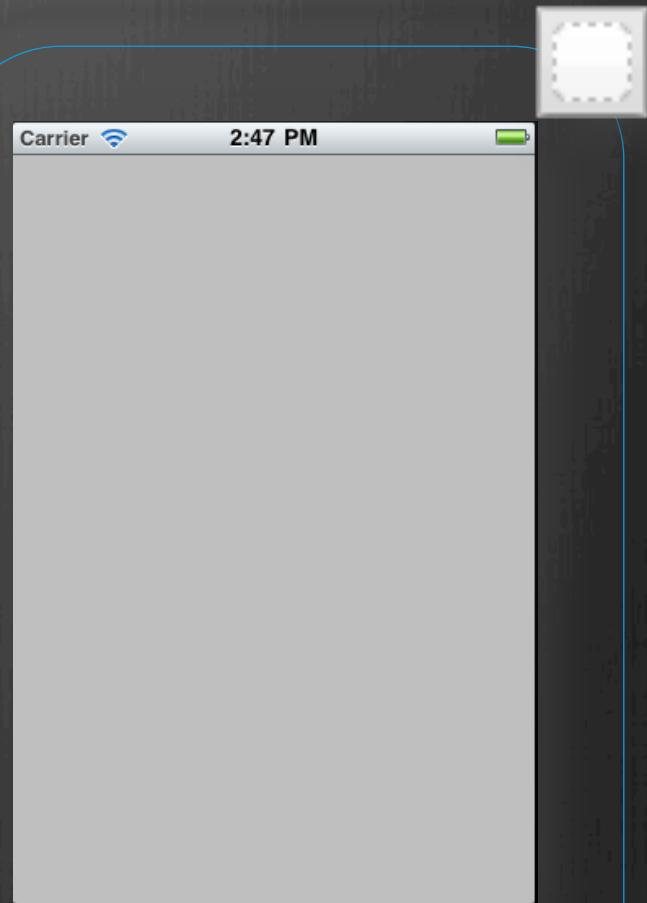
ViewController.h



ViewController.m



Controller



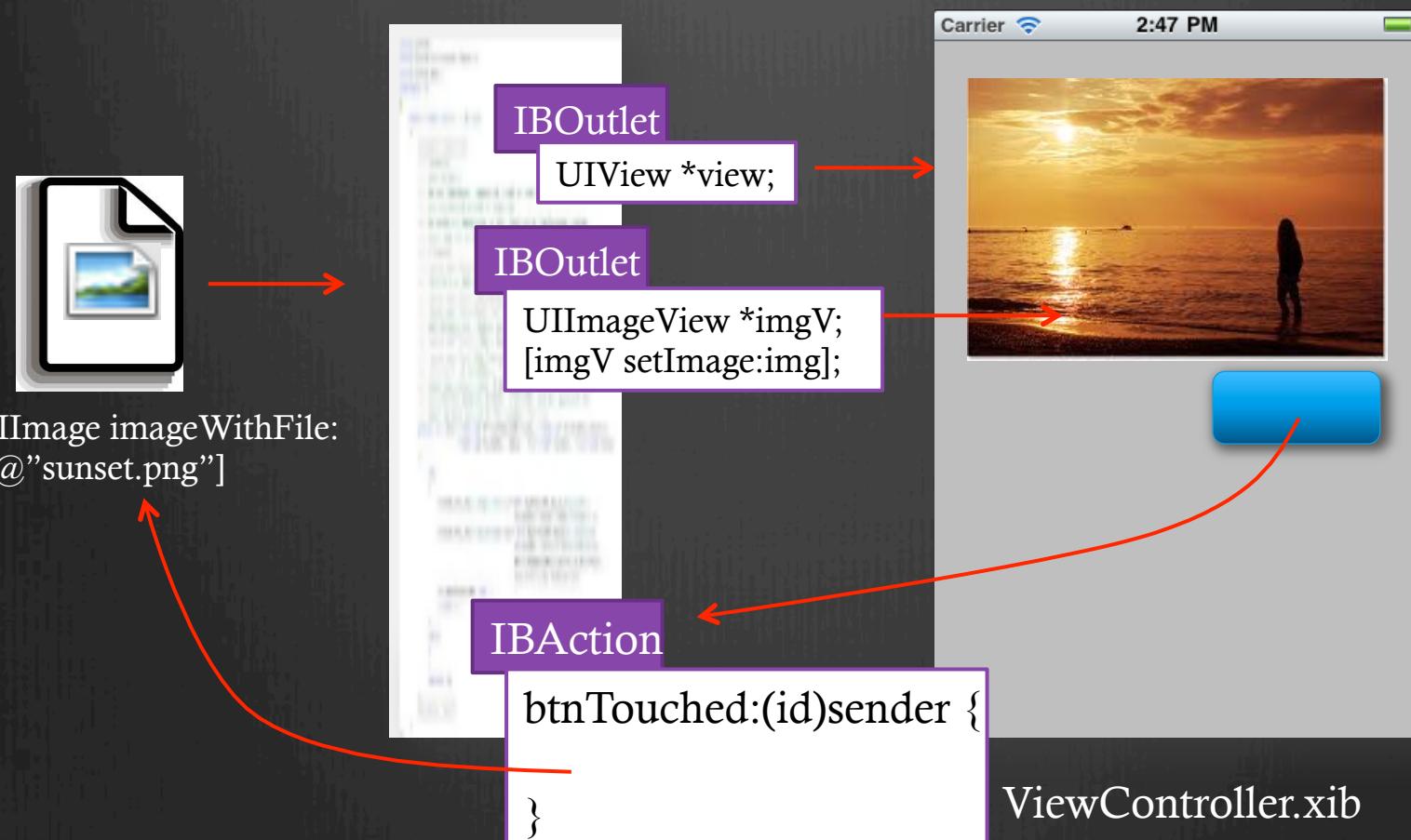
ViewController.xib

View

IBOutlet & IBAction

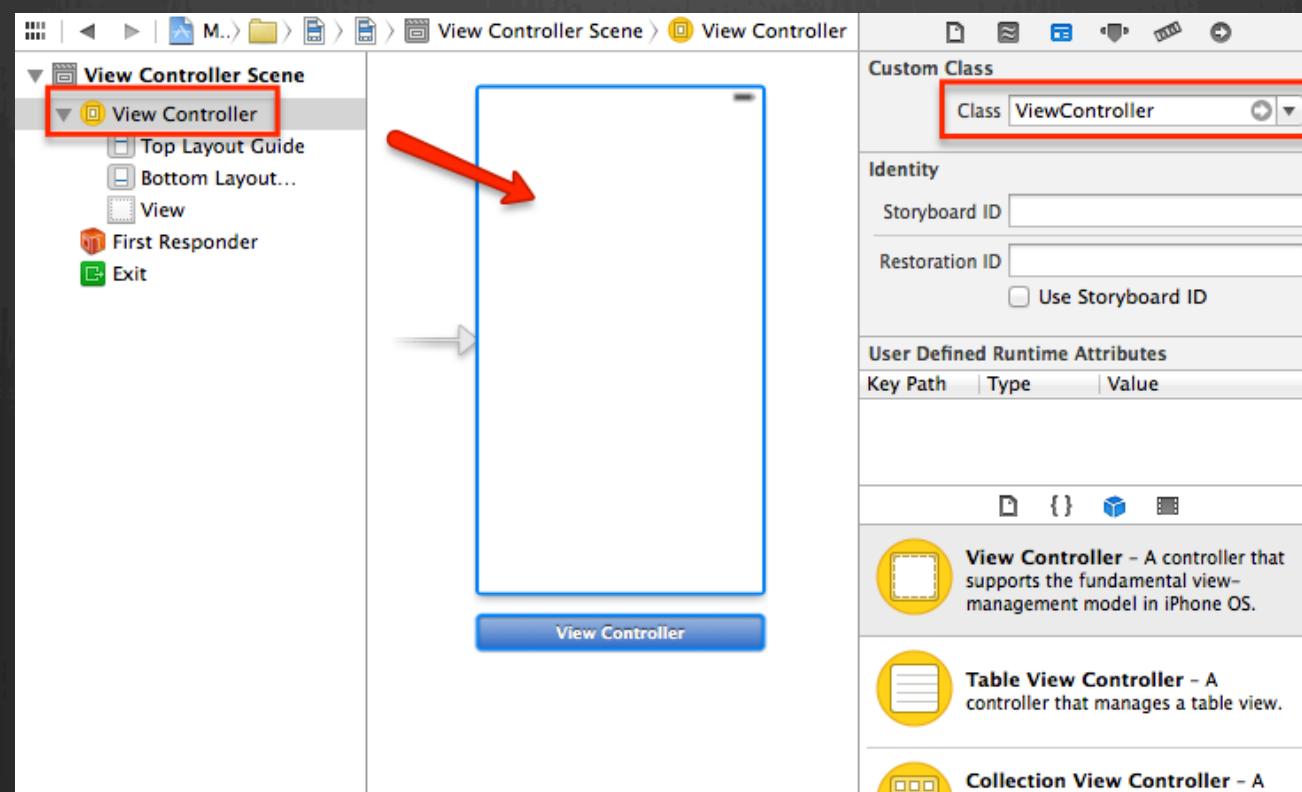
- ⦿ การที่ controller จะ control view ได้ หรือรับ event จาก view ได้นั้น จะต้องอาศัยการ binding ระหว่าง controller และ view
- ⦿ ใน iOS นั้นจะมีตัว binding ระหว่าง view กับ controller อยู่ 2 แบบ นั่นคือ IBOutlet และ IBAction
 - ⦿ IBOutlet เป็นช่องทางที่ใช้เพื่อให้ controller สามารถควบคุม control ต่างๆ บน view ได้ ซึ่ง Outlet จะอยู่ในรูปของ instance บน controller class
 - ⦿ IBAction เป็นช่องทางที่ใช้เพื่อรับ event ที่เกิดขึ้นจาก UI ซึ่ง Action จะอยู่ในรูปของ method ที่จะถูกเรียกเมื่อ User ปฏิสัมพันธ์กับ control นั้น เช่น Touch หรือ Swipe เป็นต้น

MVC In Action



Assign UIViewController class to UIView's viewController

- เมื่อสร้าง UIView และ UIViewController แล้ว เราต้องระบุว่า view นั้นใช้ class ไหนเป็น view controller โดยกำหนดใน Identity Inspector



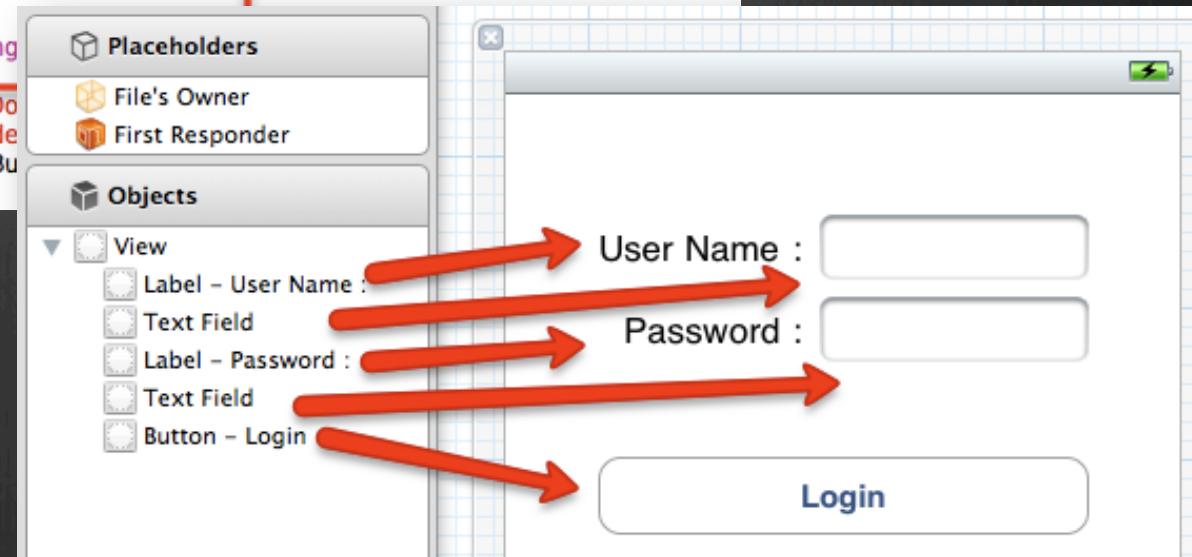


Xib File

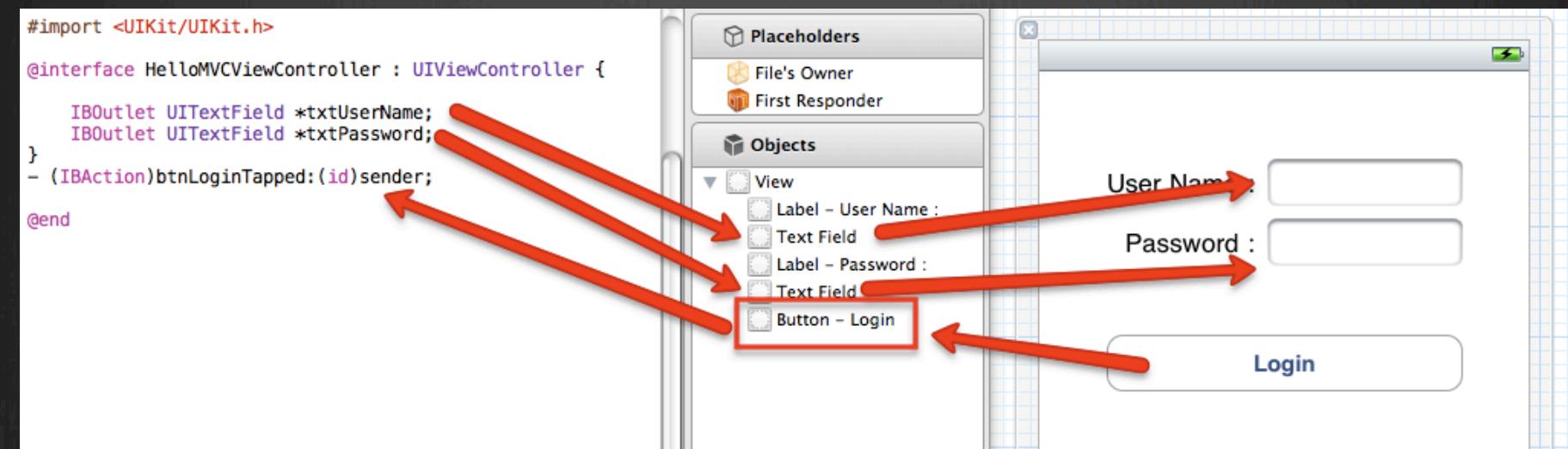
- ⊗ Xib เป็นไฟล์ XML ที่ใช้เพื่ออธิบาย โครงสร้างของ UI ของ application บน Mac OS X และ iOS
- ⊗ เดิมที่ xib นั้นเขียนว่า nib ซึ่งมาจากเทคโนโลยีของบริษัท NeXT ที่ Apple ซื้อกิจการมาหลังจาก Steve Jobs กลับมาบริหารงานที่ Apple
- ⊗ UI ของเครื่อง NeXT จะใช้ interface แบบ nib ไฟล์เป็นพื้นฐานของ Mac OSX ตลอดมาจนถึง version 9 เรียกว่า Carbon
- ⊗ ในปี 2000 Apple ทำการ upgrade Mac OS เป็น version 10 (OS X) และทำการเปลี่ยนสถาปัตยกรรมของ Mac ใหม่หมด และเปลี่ยน โครงสร้างของ nib file แบบเดิมมาใช้ XML เป็น โครงสร้าง จึงเปลี่ยนนามสกุลเป็น .xib และเรียกสถาปัตยกรรมใหม่นี้ว่า Cocoa แต่ก็ยังคงเรียก xib ว่า nib เมื่อเดิม
- ⊗ ในปี 2008 หลังจากที่ Apple เปิดตัว iPhone เครื่องแรก 1 ปี ก็มีไอเดียที่จะให้นักพัฒนาเขียน app และขายบน AppStore จึงพัฒนา Xcode ให้เขียน iOS App ได้ด้วย โดยยกสถาปัตยกรรมของ OS X ให้สามารถทำงานบน iOS ที่มีทรัพยากรน้อยได้ และปรับ Layer ของ application ให้ lightweight กว่า และเรียกสถาปัตยกรรมของ UI บน iOS ว่า Cocoa Touch

.xib file

```
<?xml version="1.0" encoding="UTF-8"?>
<archive type="com.apple.InterfaceBuilder3.CocoaTouch.XIB" version="7.10">
  <data>
    <int key="IBDocument.SystemTarget">1056</int>
    <string key="IBDocument.SystemVersion">10J869</string>
    <string key="IBDocument.InterfaceBuilderVersion">1306</string>
    <string key="IBDocument.AppKitVersion">1038.35</string>
    <string key="IBDocument.HIToolboxVersion">461.00</string>
    <object class="NSMutableDictionary" key="IBDocument.PluginVersions">
      <string key="NS.key.0">com.apple.InterfaceBuilder.IBCocoaTouchPlugin</string>
      <string key="NS.object.0">301</string>
    </object>
    <object class="NSArray" key="IBDocument.IntegratedClassDependencies">
      <bool key="EncodedWithXMLCoder">YES</bool>
      <string>IBUITextField</string>
      <string>IBUIButton</string>
      <string>IBUIView</string>
      <string>IBUILabel</string>
      <string>IBProxyObject</string>
    </object>
    <object class="NSArray" key="IBDocument.PluginDependencyGraph">
      <bool key="EncodedWithXMLCoder">NO</bool>
      <string>com.apple.InterfaceBu
    </object>
  </data>
</archive>
```



Binding between code & xib

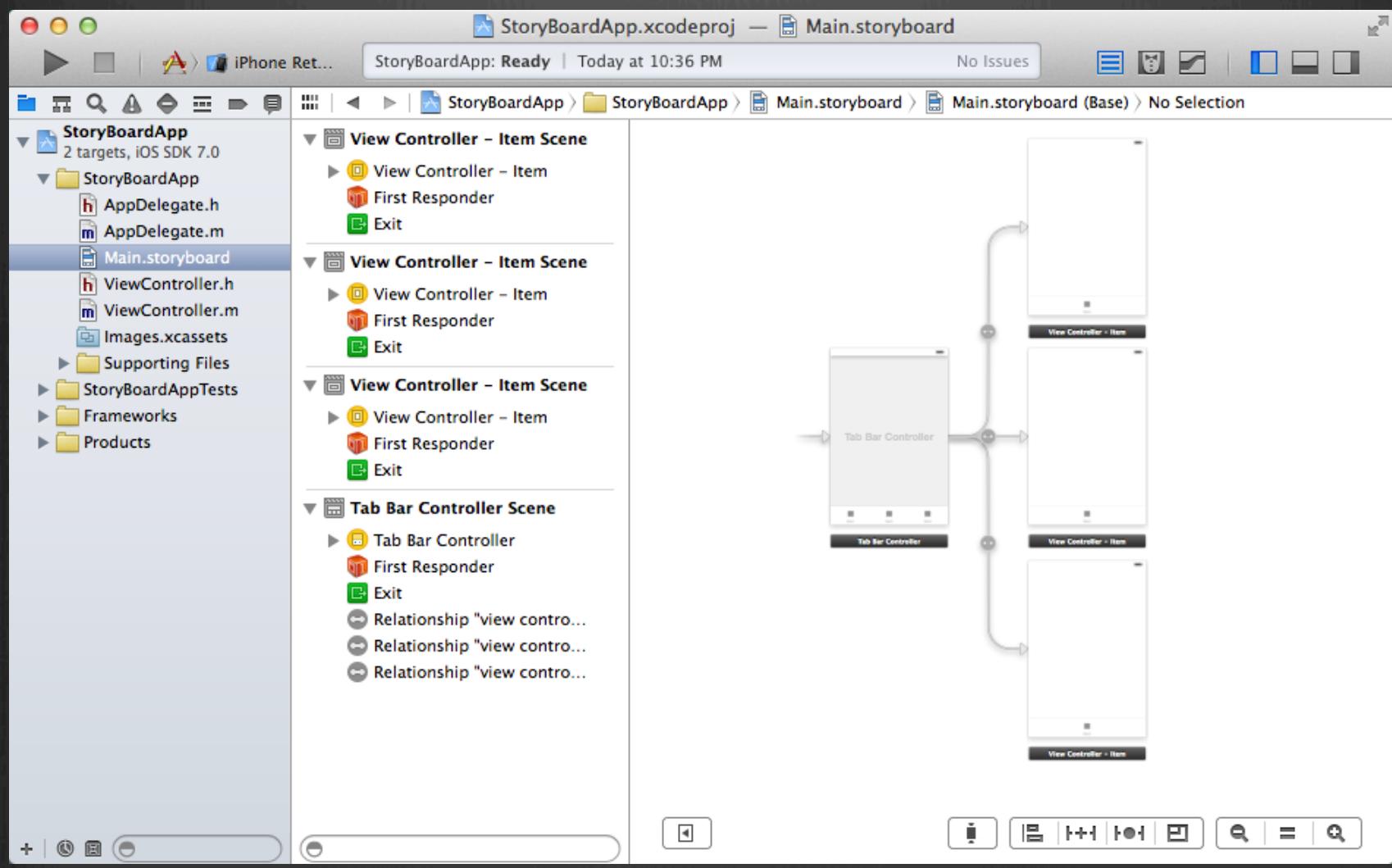


Storyboard

- ◉ Storyboard เป็น feature ที่ถูกเพิ่มเข้ามาใน iOS5 สามารถเลือกได้ว่าจะสร้าง app แบบ xib-based หรือใช้ storyboard แต่ใน Xcode 5 จะไม่มีให้เลือกแล้ว ต้องใช้ storyboard เป็นหลัก
- ◉ ช่วยให้นักพัฒนาสามารถพัฒนา app ได้เร็ว สามารถสร้าง prototype ของ app ได้ในเวลาอันสั้น
- ◉ Focus ที่การ transit ระหว่าง scene ต่างๆ ใน storyboard และค่อย implement code ทีหลัง
- ◉ Storyboard เป็น collection ของ xib file โดยที่ใน storyboard จะมี xib file ย่อยๆ เรียกว่า scene
- ◉ Storyboard ก็เป็น xml file เมื่อน xib file สามารถเปิดดูด้วย text editor เช่นเดียวกัน

Note : เราจะพูดถึง storyboard โดยละเอียดอีกครั้ง ในบทที่ 11

Storyboard



App Delegate (1)

- ✿ จาก layer ของ iOS App ตอนต้น จะเห็นว่า app จะมี object application ออยู่ด้วย (class UIApplication) คือเป็น object ที่เป็นตัว application เอง แต่เราไม่สามารถจัดการอะไรได้ เพราะ object application นั้นถูกสร้างโดย iOS
- ✿ App Delegate จึงเป็น class ที่ใช้เพื่อเขียน code สำหรับ handle กับ event ต่างๆ ของ app ได้โดยไม่ต้องแตะต้อง object application เลย เช่น ถ้าเราต้องการทำอะไรบางอย่างในขณะที่ app กำลังเริ่ม start เรา ก็เขียน code ใน method “application didFinishLaunching” หรือถ้าต้องการเขียน code เพื่อ save state ของเกมส์ไว้เมื่อ user กดปุ่ม home ก็สามารถเขียนได้ที่ method “applicationDidEnterBackground” เป็นต้น
- ✿ AppDelegate จะเป็น class แรกที่ทำงานและถูกเรียกจาก main method ของ app (อยู่ในไฟล์ >Supporting Files > main.m)

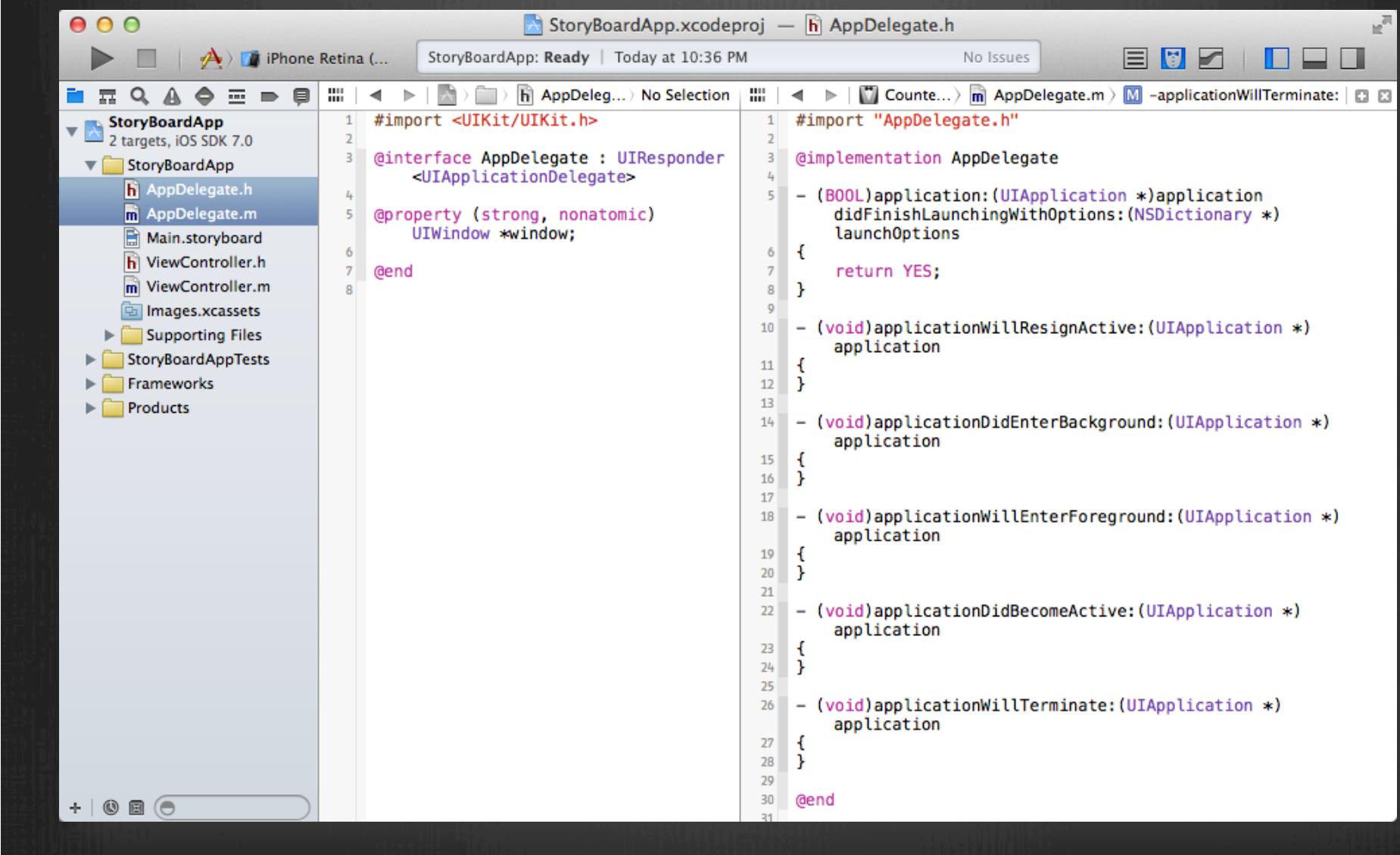
```

int main(int argc, char * argv[]) {
    @autoreleasepool {
        return UIApplicationMain(argc, argv, nil,
                               NSStringFromClass([AppDelegate class]));
    }
}

```

Note : หลักการของ AppDelegate นั้นจะพูดถึงอีกครั้ง ในเรื่อง delegate (บทที่ 7,8,9)

App Delegate (2)



The screenshot shows the Xcode interface with the following details:

- Project Navigator:** Shows the project structure for "StoryBoardApp". The "StoryboardApp" folder contains "AppDelegate.h", "AppDelegate.m", "Main.storyboard", "ViewController.h", "ViewController.m", "Images.xcassets", "Supporting Files", "StoryBoardAppTests", "Frameworks", and "Products".
- Editor:** The left panel displays the "AppDelegate.h" file, which includes the standard UIResponder and UIApplicationDelegate protocols and defines a window property.
- Editor:** The right panel displays the "AppDelegate.m" file, which contains implementations for various application lifecycle methods: `-applicationWillTerminate:`, `-applicationDidBecomeActive:`, `-applicationDidEnterBackground:`, `-applicationWillEnterForeground:`, `-applicationWillResignActive:`, and `-applicationDidFinishLaunchingWithOptions:`.
- Toolbar:** Standard Xcode toolbar with icons for play, stop, run, and others.
- StatusBar:** Shows "iPhone Retina (..)" and "StoryBoardApp: Ready | Today at 10:36 PM".
- Navigation Bar:** Shows "StoryBoardApp.xcodeproj — AppDelegate.h" and "No Issues".

```
#import <UIKit/UIKit.h>
@interface AppDelegate : UIResponder <UIApplicationDelegate>
@property (strong, nonatomic) UIWindow *window;
@end

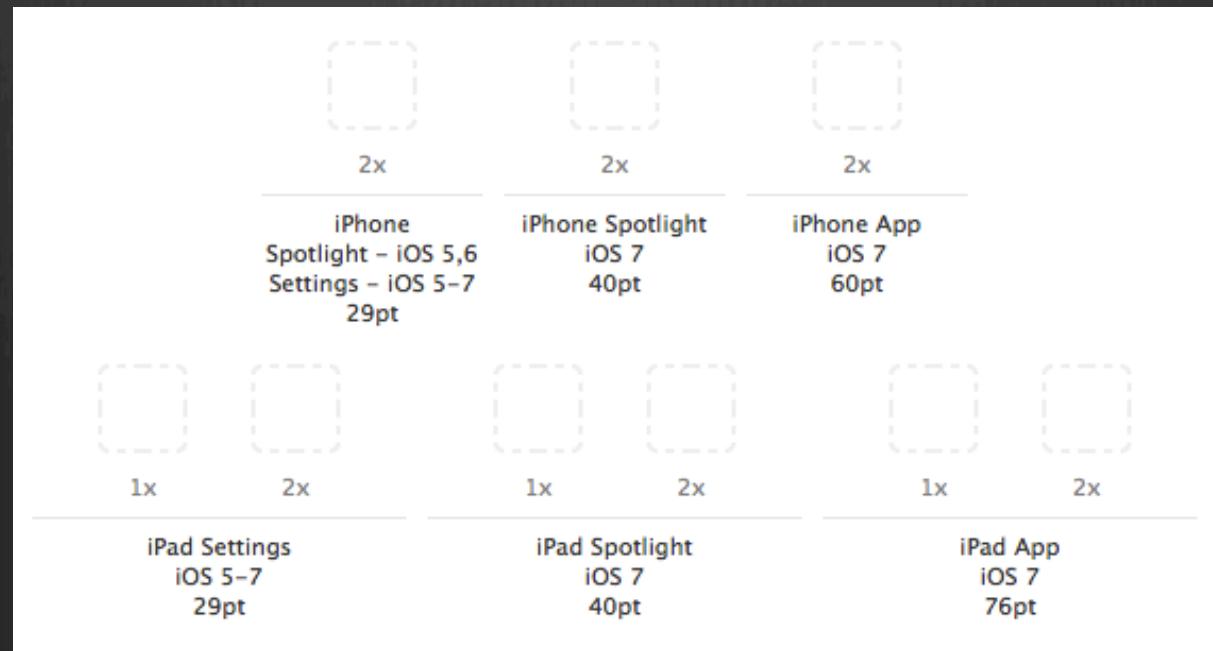
#import "AppDelegate.h"
@implementation AppDelegate
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    return YES;
}
- (void)applicationWillResignActive:(UIApplication *)application
{
}
- (void)applicationDidEnterBackground:(UIApplication *)application
{
}
- (void)applicationWillEnterForeground:(UIApplication *)application
{
}
- (void)applicationDidBecomeActive:(UIApplication *)application
{
}
- (void)applicationWillTerminate:(UIApplication *)application
{
}
@end
```

App Assets

- ◎ ใน Xcode 5 นั้นจะมี folder พิเศษใช้เก็บ resource ที่ถูกใช้ใน app ไว้ที่เดียว คือ Images.xcassets
- ◎ เมื่อมี asset ที่ถูกเพิ่มเข้าไป Xcode จะทำการสร้าง guideline ให้ว่า asset เหล่านั้นถูกต้องตาม UX Guideline ของ Apple หรือไม่ เช่น ขนาดรูปที่จะนำมาใช้เป็น icon ของ app มีขนาดถูกต้องหรือไม่
- ◎ เพื่อให้ภาพใน app มีความคมชัดเมื่อแสดงบนจอภาพปกติและ Retina Display รูปที่นำมาใช้ควรจะมีขนาดเท่ากับขนาดที่แสดงจริง
- ◎ เนื่องจาก iOS device มีหน้าจอ 2 แบบคือธรรมดากับ Retina display จึงต้องใช้รูป 2 รูป รูปแรกคือขนาดธรรมดากับรูปที่ 2 ที่มีขนาดขยาย 2 เท่าสำหรับ Retina Display ซึ่ง Xcode จะจัดการให้เราเองว่าเมื่อไหร่ต้องแสดงรูปไหนเมื่อ app ทำงานบนอุปกรณ์จริง
- ◎ เราจะต้องตั้งชื่อรูปโดยต่อท้าย @2x ในชื่อไฟล์เพื่อบอก Xcode ว่ารูปนี้คือขนาด 2 เท่า เช่น ถ้ารูปชื่อ Football.png รูปที่มีขนาด 2 เท่าจะต้องชื่อว่า Football@2x.png

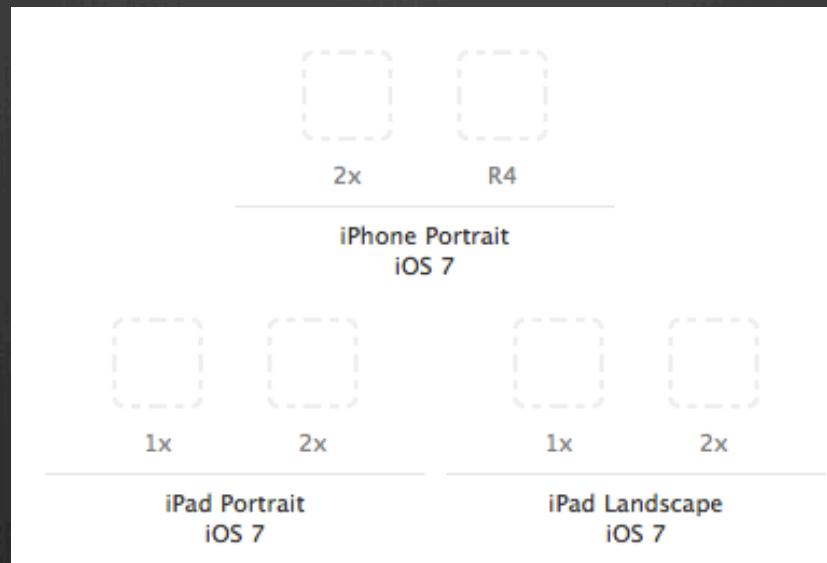
App Icons

- เนื่องจาก Xcode 5 และ iOS 7 นั้น support iPhone 4 หรือใหม่กว่าเท่านั้น (เป็น Retina display แล้ว) จึงตัด icon สำหรับ app ที่ทำงานบนหน้าจอแบบเก่าออกไป แต่ iPad ยังคงมีอุปกรณ์ที่ไม่ใช่ Retina อยู่ คือ iPad mini gen1 ดังนั้น icon จึงต้องมี 2 ขนาดคือ 1x และ 2x
- ขนาดของ icon จะเป็นขนาด 2 เท่า (x2) ของ pixel ที่แสดงใน asset



Launch Images

- ขนาดของ Launch image ของ iPhone จะรองรับขนาดหน้าจอ 2 ขนาด คือหน้าจอที่มีอัตราส่วน 3:4 (iPhone 4, 4s) และ 16:9 (5, 5s, 5c) นั่นคือ **640x960** และ **640x1136px**
- ส่วน Launch image ของ iPad จะเป็น 4:3 ทั้งหมด แต่ iPad มีทั้งหน้าจอธรรมดากลาง Retina display จึงต้องใช้ 2 ขนาด คือ **1024x768** และ **2048x1536px** และถ้า app บน iPad รองรับการแสดงผลทั้งแนวตั้งและแนวนอน launch image ก็จะต้องมีทั้ง 2 แนวเพื่อให้การแสดงผลคมชัดสมบูรณ์

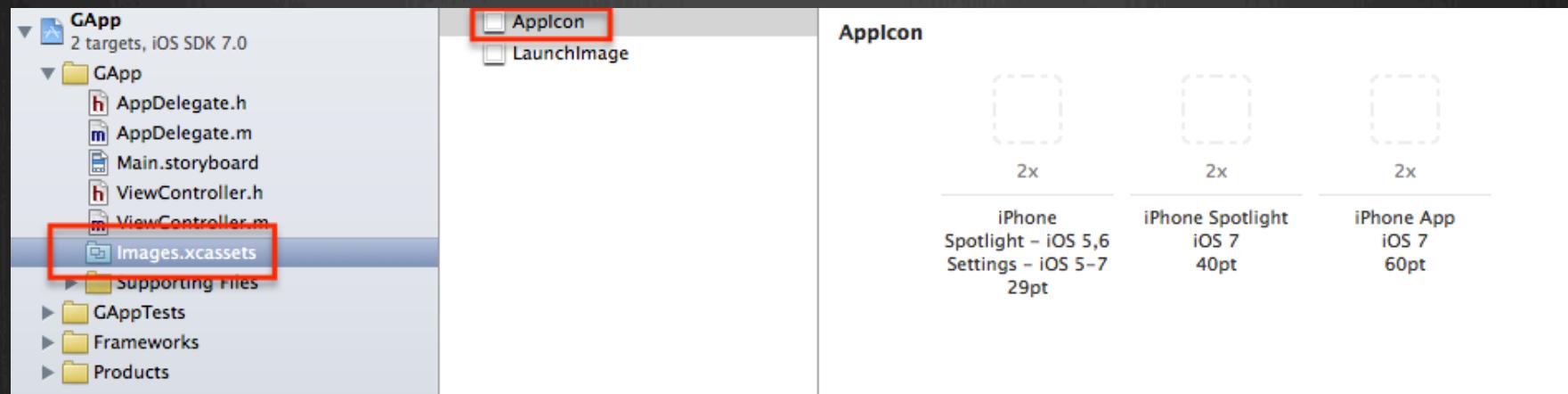


Lab : App with Login Screen (1/12)

- ⦿ วัตถุประสงค์
 - ⦿ เพื่อให้เข้าใจพื้นฐานการเขียนโปรแกรมเพื่อใช้ Storyboard ทำงานร่วมกับ xib file
 - ⦿ เข้าใจการทำงานระหว่าง view ต่างๆ บน application และการ display view แบบ “Modal”
 - ⦿ เข้าใจการ setup resource ต่างๆ ใน Xcode
 - ⦿ การใช้ static instance และ static (class) method
 - ⦿ ตัวอย่างหนึ่งของการทำหน้าจอ Login
- ⦿ ขั้นตอน
 - ⦿ เพิ่ม icon และรูปต่างๆ ที่จะใช้ใน app
 - ⦿ พัฒนา main view บน storyboard
 - ⦿ สร้าง login view แบบ xib file และเรียกใช้แบบไม่ผ่าน storyboard

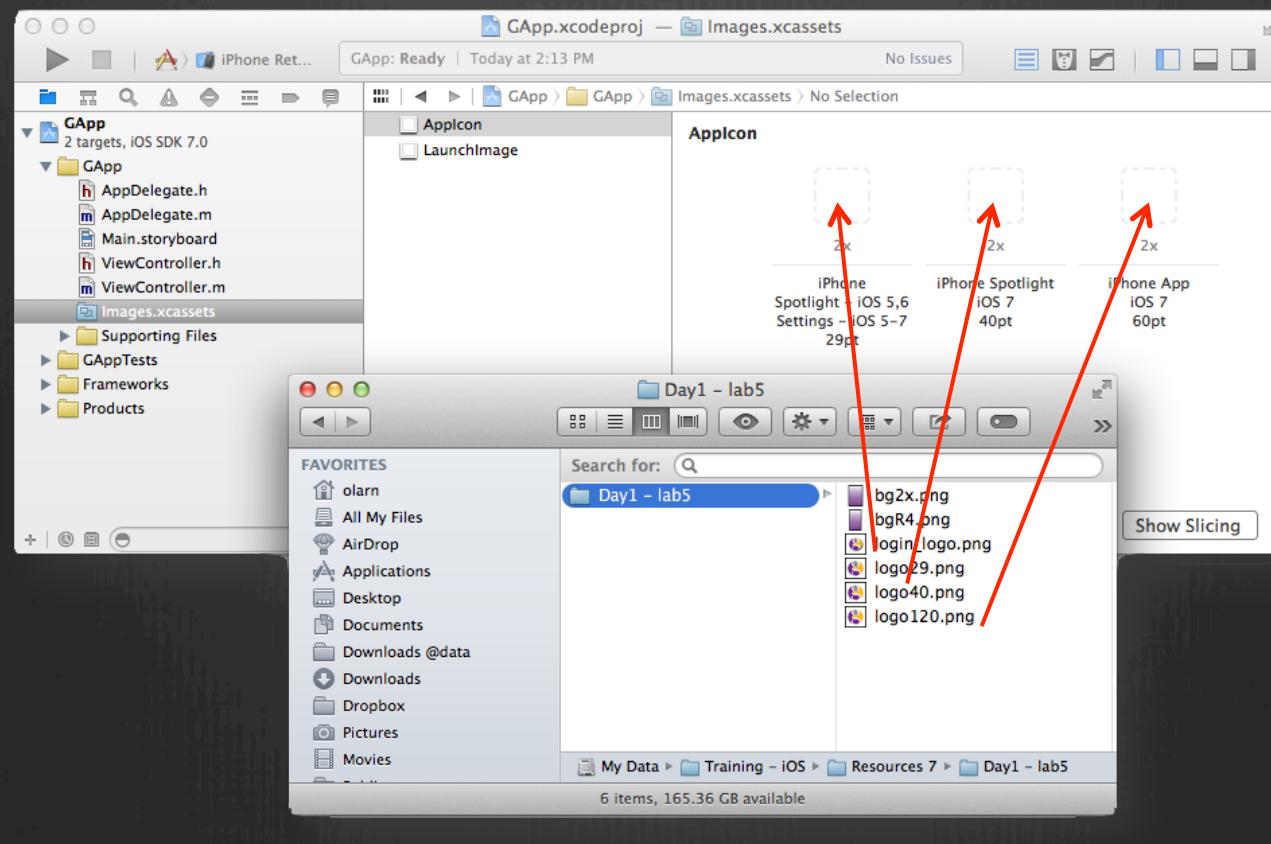
Task : เพิ่ม resource ใน project (2/12)

1. Create project ใหม่ (iOS – Single View Application) ตั้งชื่อว่า GApp และกำหนด device เป็น iPhone
2. จาก Navigation Pane ให้เลือก Images.xcassets > AppIcon



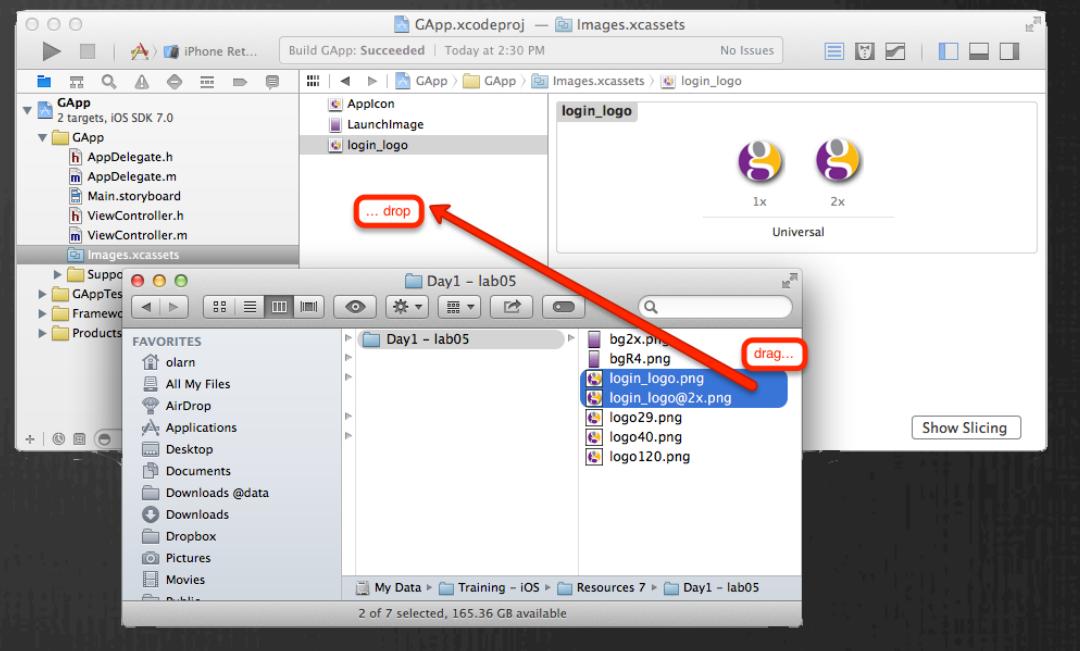
Task : เพิ่ม resource ใน project (3/12)

- เปิด Finder ไปยัง folder “Resources 7/Day1 – lab05/” และ drag ไฟล์ “logo29.png”, “logo40.png”, และ “logo120.png” ไปยัง AppIcon ขนาด 29pt, 40pt, และ 120pt ตามลำดับ



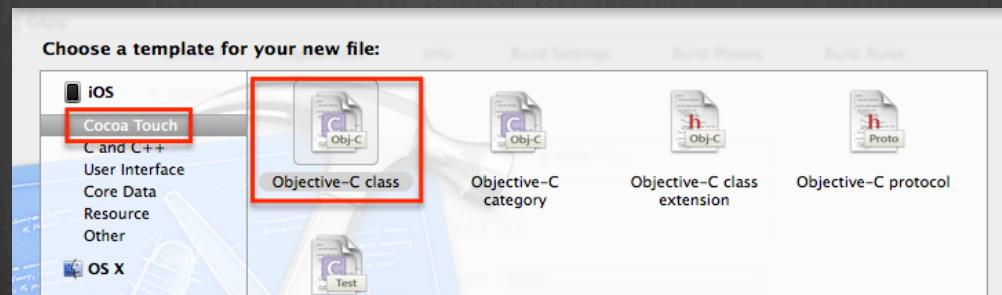
Task : เพิ่ม resource ใน project (4/12)

4. จากนั้น click ที่ LaunchImage และ drag ไฟล์ “launch2x.png” และ “launchR4.png” ไปวางไว้ใน icon 2x และ R4 ตามลำดับ
5. จาก Finder ให้กด Command ค้างไว้แล้ว click ที่ไฟล์ “login_logo.png” และ “login_logo@2x.png” เพื่อเลือก 2 ไฟล์พร้อมกัน จากนั้น drag ทั้ง 2 ไฟล์ไปยังพื้นที่ว่างๆ ใต้ AppIcon และ LaunchImage เพื่อสร้าง image 2 ขนาด ใน Asset
6. ทำซ้ำข้อ 5 ด้วยไฟล์ “bg.png” และ “bg@2x.png”

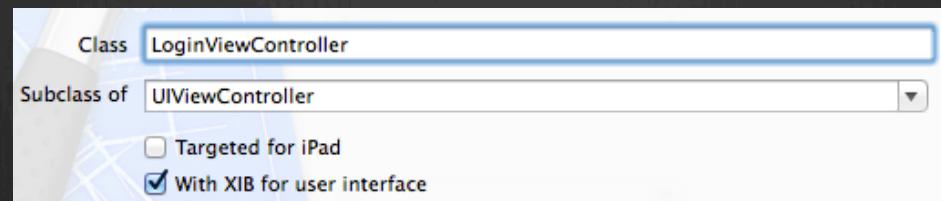


Task : Create Login Screen (5/12)

- สร้าง file ใหม่สำหรับใช้เป็นหน้าจอ login โดย click ขวาที่ folder “GApp” ใน Navigation Pane และเลือก “New File...” ใน Template dialog ให้เลือก iOS > Cocoa Touch > Objective-C class และ click ปุ่ม “Next”

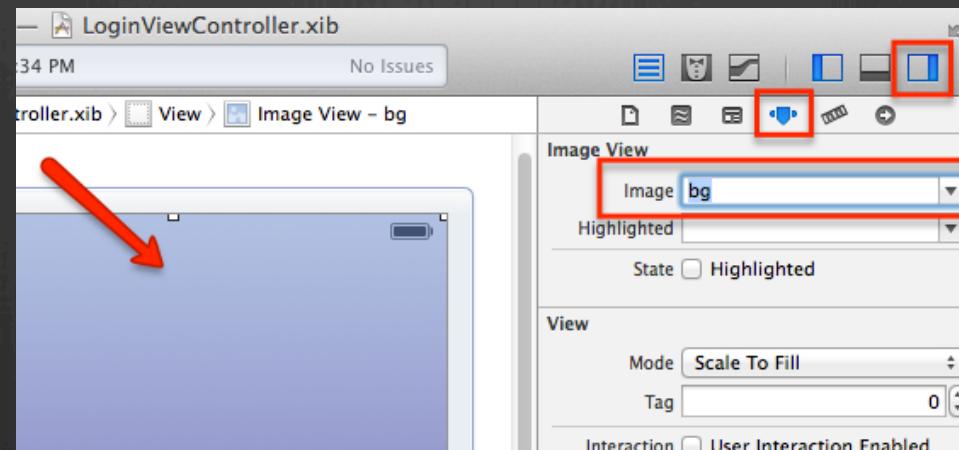


- ตั้งชื่อ class ว่า “LoginViewController” และเลือก subclass of เป็น “UIViewController” เอา check box Targeted for iPad ออก และเลือก check box “With XIB for user interface” และ click “Next” เลือก folder ที่จะ save file และ click “Create”



Task : Create Login Screen (6/12)

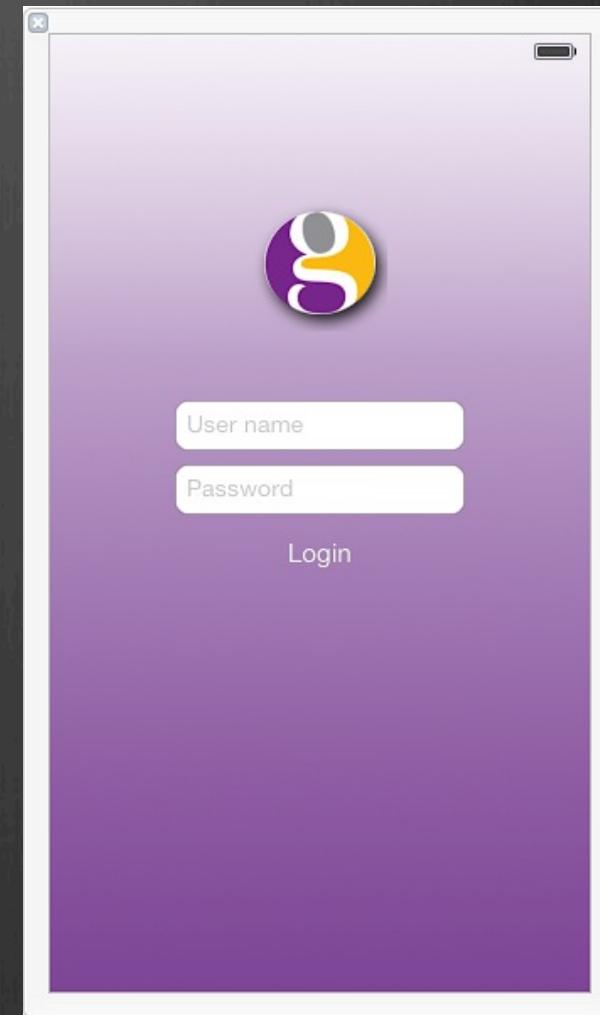
9. Click ที่ไฟล์ “LoginViewController.xib” ใน Navigation Pane และจาก control จาก Library Pane ชื่อ “Image View” มาวางบน View โดยจัดขนาดของ image view ให้เต็ม view พอดี
10. Click เลือก Image View และกำหนดรูปภาพให้กับ image view โดยเลือก Image ชื่อ “bg”



Task : Create Login Screen (7/12)

11. วาง control เพิ่ม ดังนี้

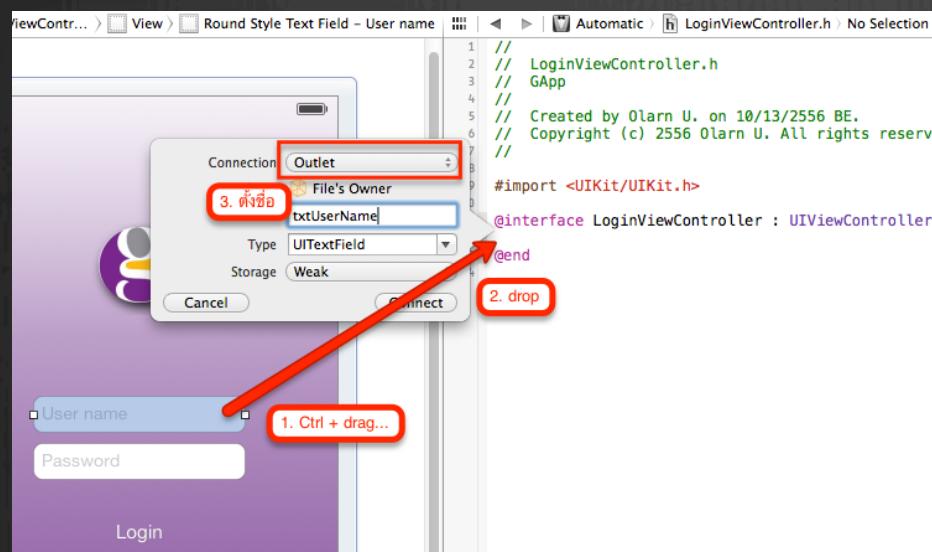
- เพิ่ม “Image View” มาอีก 1 control วางบน Image View เดิม จากนั้นกำหนด property “Image” เป็น “login_logo” และเลือก property “Mode” เป็น “Center”
- เพิ่ม control “Text Field” และป้อนค่าในช่อง Placeholder ว่า “User name”
- เพิ่ม control “Text Field” และป้อนค่าในช่อง Placeholder ว่า “Password” และ click property (check box) “Secure” เป็น “checked”
- เพิ่ม control “Button” ลงบน View และ double click บน button เปลี่ยน title เป็น “Login” และเปลี่ยน Text Color เป็น “White Color”



Task : Implement Login Screen (8/12)

12. เปลี่ยน editor mode เป็น Assistance editor (สังเกตว่า pane ด้านขวาจะต้องเป็น LoginViewController.h) ทำการผูก Outlet และ Action โดยที่

- Textbox แรกเป็น Outlet ตั้งชื่อว่า “txtUserName”
- Textbox ที่ 2 เป็น Outlet ตั้งชื่อว่า “txtPassword”
- Button เป็น **Action** ตั้งชื่อว่า “btnLoginTouched”



Code ที่ได้จะเป็นแบบนี้

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

@interface LoginViewController : UIViewController

@property (weak, nonatomic) IBOutlet UITextField *txtUserName;
@property (weak, nonatomic) IBOutlet UITextField *txtPassword;
- (IBAction)btnLoginTouched:(id)sender;
@end
```

Task : Implement Login Screen (9/12)

13. จาก LoginViewController.h เพิ่ม static method “getInstance”

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

@interface LoginViewController : UIViewController

@property (weak, nonatomic) IBOutlet UITextField *txtUserName;
@property (weak, nonatomic) IBOutlet UITextField *txtPassword;

- (IBAction)btnLoginTouched:(id)sender;
+ (LoginViewController *)getInstance; // เป็น static method

@end
```

14. เพิ่ม static instance และเขียน method “getInstance” ใน LoginViewController.m

```
#import "LoginViewController.h"

static LoginViewController * loginViewController;

@implementation LoginViewController

+ (LoginViewController *)getInstance
{
    if (!loginViewController) {
        loginViewController = [[LoginViewController alloc] init];
    }
    return loginViewController;
}
```

Task : Implement Login Screen (10/12)

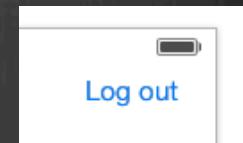
15. ເຂີຍນ code method “- (IBAction)btnLoginTouched:(id)sender” ໃນ LoginViewController.m

```
- (IBAction)btnLoginTouched:(id)sender
{
    if ([self.txtUserName.text isEqualToString:@"user"] &&
        [self.txtPassword.text isEqualToString:@"1234"])
    {
        self.txtUserName.text = @""; // login ผ่านแล้วให้ clear text ทิ้ง
        self.txtPassword.text = @"";
        [self dismissViewControllerAnimated:YES completion:nil];
    }
}
```

16. เปิดไฟล์ AppDelegate.m เพิ่ม code เพื่อแสดงหน้าจอ login ตอน app เริ่มทำงาน

Task : Implement Logout (11/12)

17. เปิด Main.storyboard และเพิ่ม control “Button” ที่มุมขวาบนของ view



18. Double click ที่ button และเปลี่ยน title เป็น Log out

19. เปลี่ยน editor mode เป็น “Assistance Editor” และผูก **Action** ของ Button กับ ViewController.h ตั้งชื่อ Action ว่า “btnLogoutTouched” (อย่าลืมเปลี่ยน Outlet เป็น Action)

20. เปิดไฟล์ “ViewController.m” เขียน code ใน method “– (IBAction)btnLogoutTouched:(id)sender” ดังนี้

```
#import "ViewController.h"
#import "LoginViewController.h"

@implementation ViewController

- (IBAction)btnLogoutTouched:(id)sender
{
    LoginViewController * loginViewController = [LoginViewController getInstance];

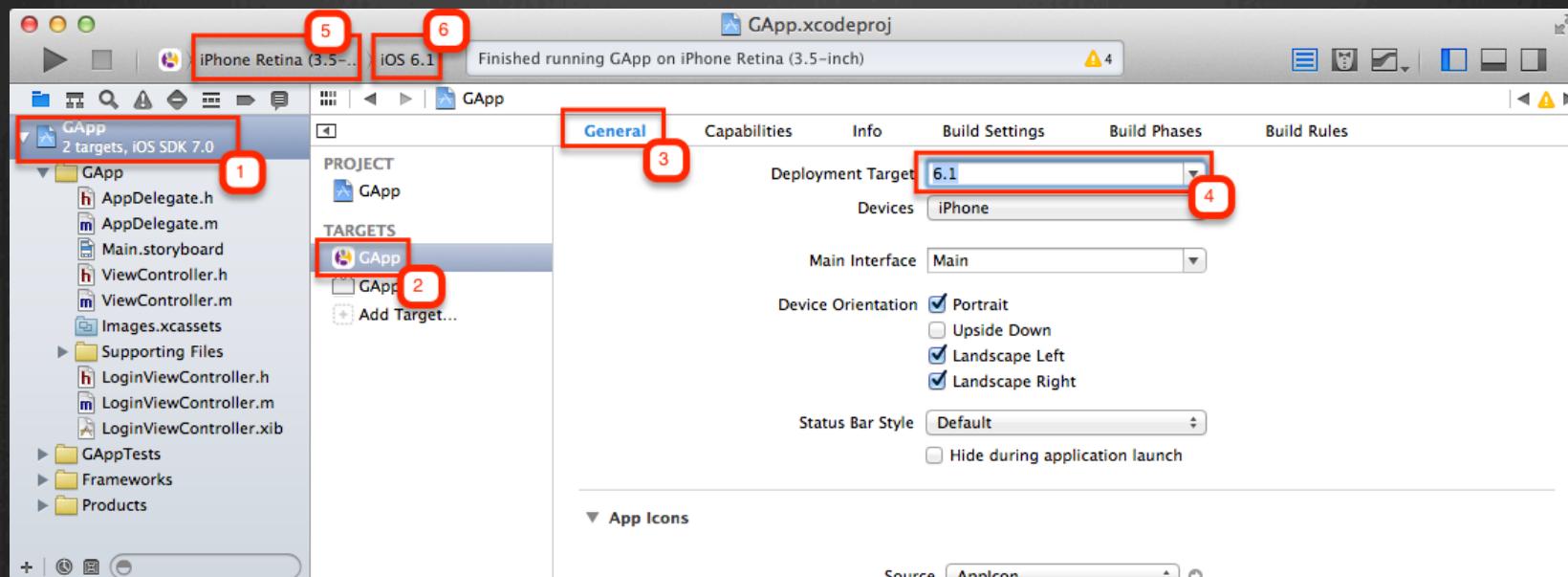
    UIViewController * rootViewController =
        [[[UIApplication sharedApplication] windows] objectAtIndex:0] rootViewController];

    [rootViewController presentViewController:loginViewController
                                animated:YES
                              completion:nil];
}
```

Task : Run to Test (12/12)

21. Run โปรแกรมเพื่อดูผลลัพธ์

- ทดลองเปลี่ยนขนาดหน้าจอเพื่อดูผลลัพธ์
- ทดลองเปลี่ยน Deployment Target เพื่อ run โปรแกรมบน iOS 6.0 – 6.1



- ทดลองใช้ Assistance Editor เปรียบเทียบ design ระหว่าง iOS 7 และ 6