

iOS_初階

Become a XCoder

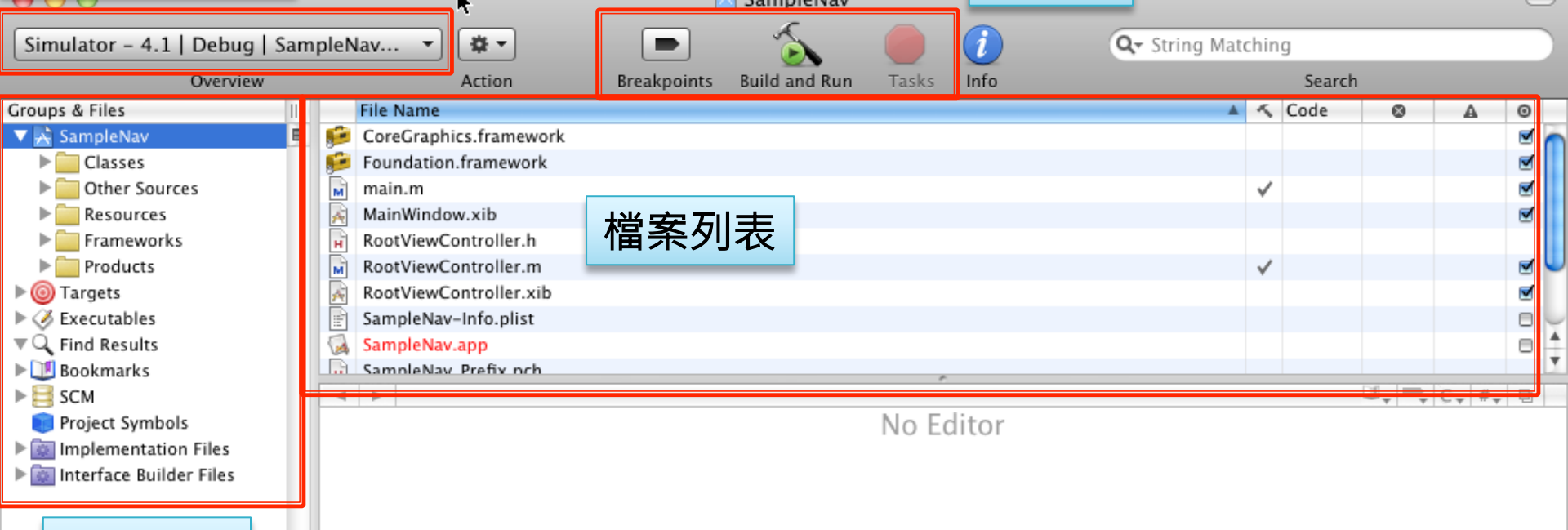
Outline

- ▶ Introduction to Xcode
- ▶ iOS 4.2 快速建立新專案
- ▶ iOS 4.2 由空專案建立專案
- ▶ Introduction to Object-C
- ▶ 元件練習UIButton+Label
- ▶ 元件練習UIImageView
- ▶ 元件練習TextField
- ▶ 元件練習TextView
- ▶ 元件練習TableView
- ▶ 切換頁面練習

Introduction to Xcode

編譯種類設定

快速鍵



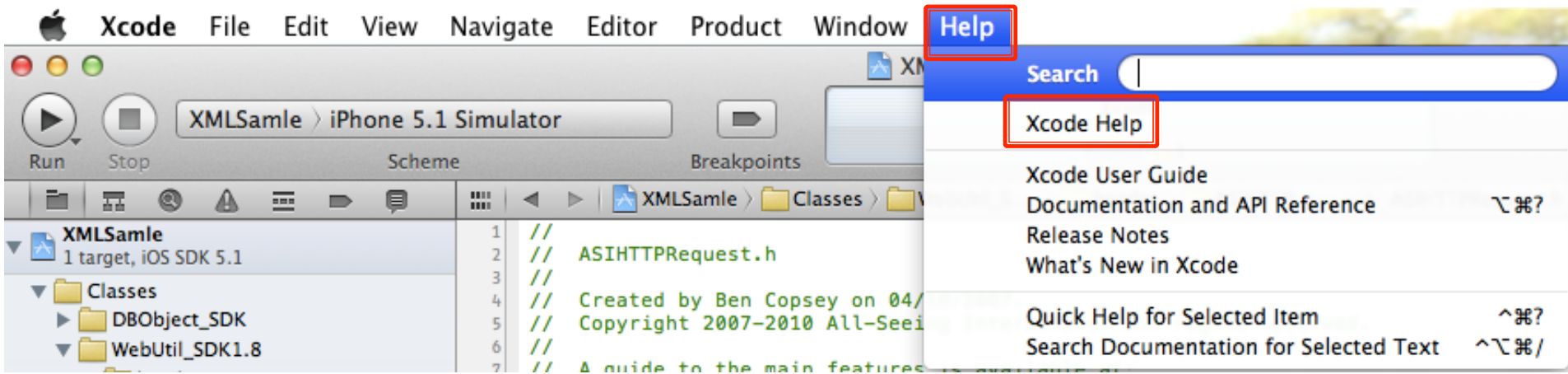
檔案列表

專案總管

快捷鍵

- ▶ 常用快捷
 - 複製(Command+C)
 - 貼上(Command+V)
 - 剪下(Command+X)
 - 返回(Command+Z)

HELP



HELP

The screenshot shows the Xcode documentation interface for the `UITableView` class. The left sidebar contains a search bar and a list of results. A red box highlights the search bar containing the text "輸入查詢字串" (Input search string). Another red box highlights the "Sample Code" section, which lists various code examples like "DateCell", "HeaderFooter", etc. A green box highlights the "UITableView" class in the search results, with a green arrow pointing to the main content area. The main content area is titled "UITableView Class Reference" and contains a table of properties and a section for "Overview". A blue box with the text "內容" (Content) is positioned in the top right corner of the main content area.

輸入查詢字串

Next

UITableView Class Reference

內容

Inherits from	UIScrollView : UIView : UIResponder : NSObject
Conforms to	NSCoding NSCoding (UIScrollView) NSCoding (UIView) UIAppearance (UIView) UIAppearanceContainer (UIView) NSObject (NSObject)
Framework	/System/Library/Frameworks/UIKit.framework
Availability	Available in iOS 2.0 and later.
Companion guide	Table View Programming Guide for iOS
Declared in	UITableView.h
Related sample code	DrillDownSave iPhoneCoreDataRecipes LocateMe TableViewSuite TheElements

Overview

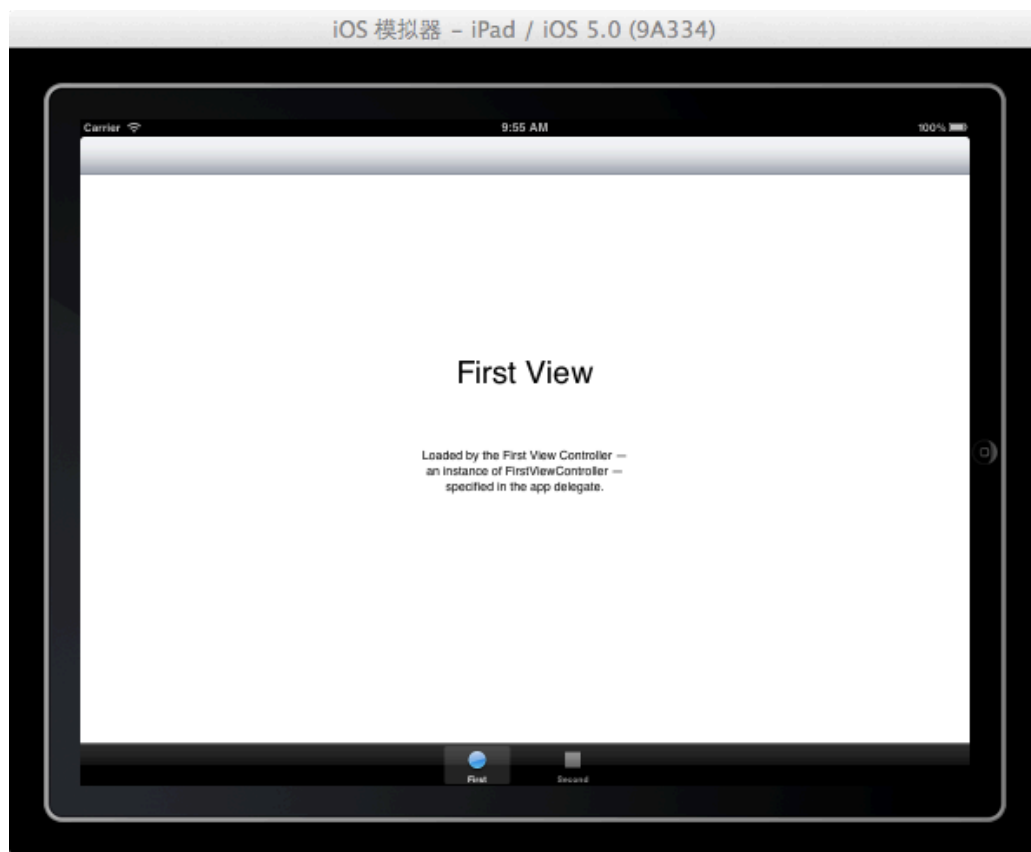
An instance of `UITableView` (or simply, a table view) is a means for displaying and editing hierarchical lists of information.

範例程式

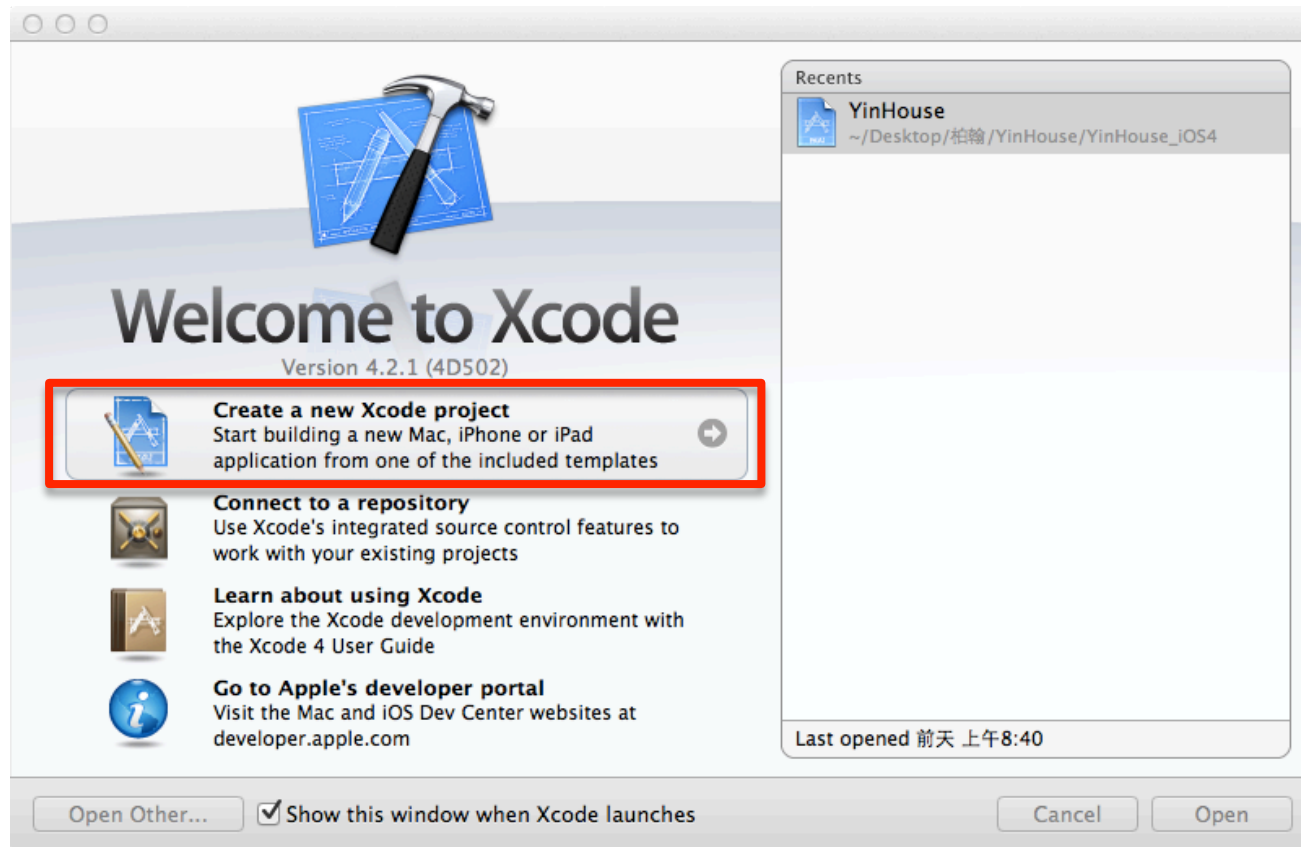
iOS 4.2 快速建立新專案

- ▶ 學習目的
 - 快速建立新專案，並進程式撰寫
 - 熟悉Xcode操作介面
 - 簡易修改程式內容

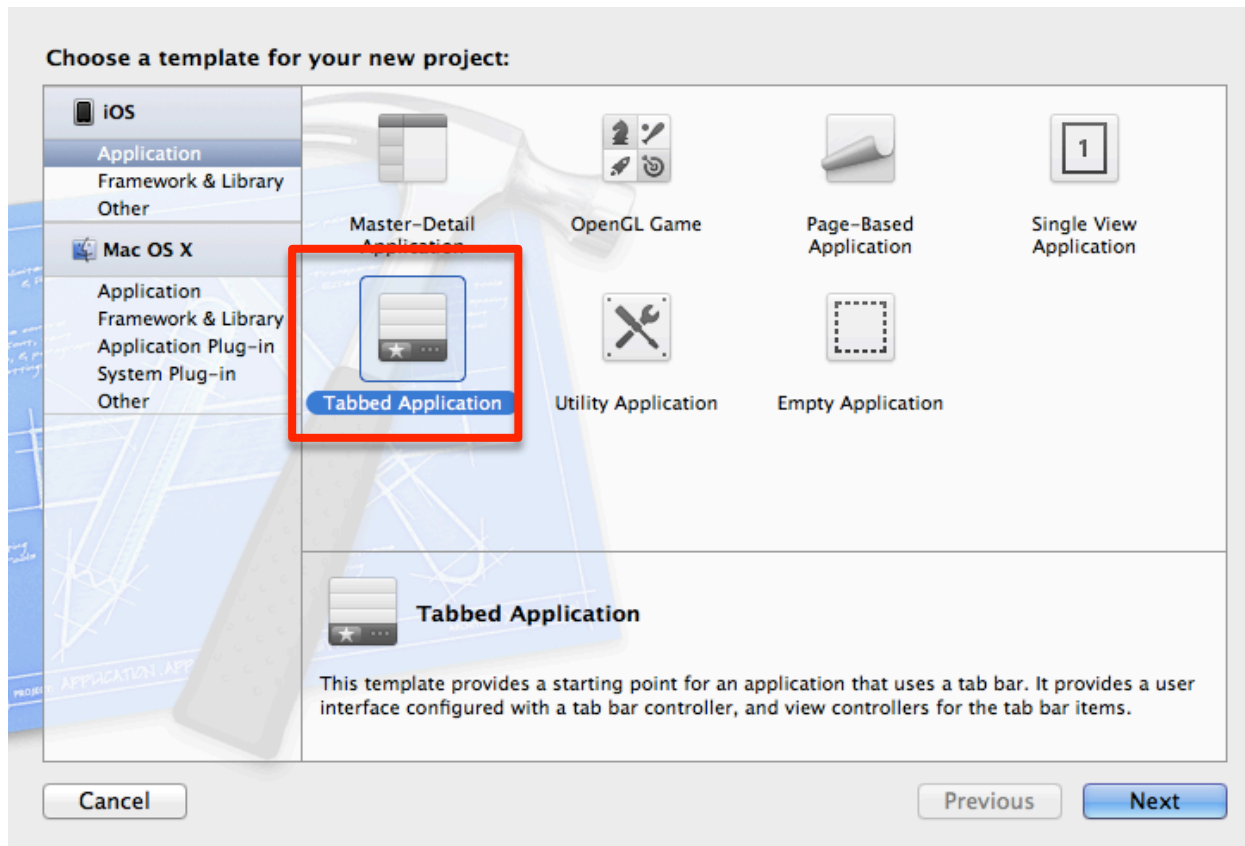
執行結果



Step 1 - 建立新專案



Step 2 – 選擇新專案類型



Step 3 - 設定新專案資料

Choose options for your new project:

Product Name

Company Identifier

Bundle Identifier

Class Prefix

Device Family

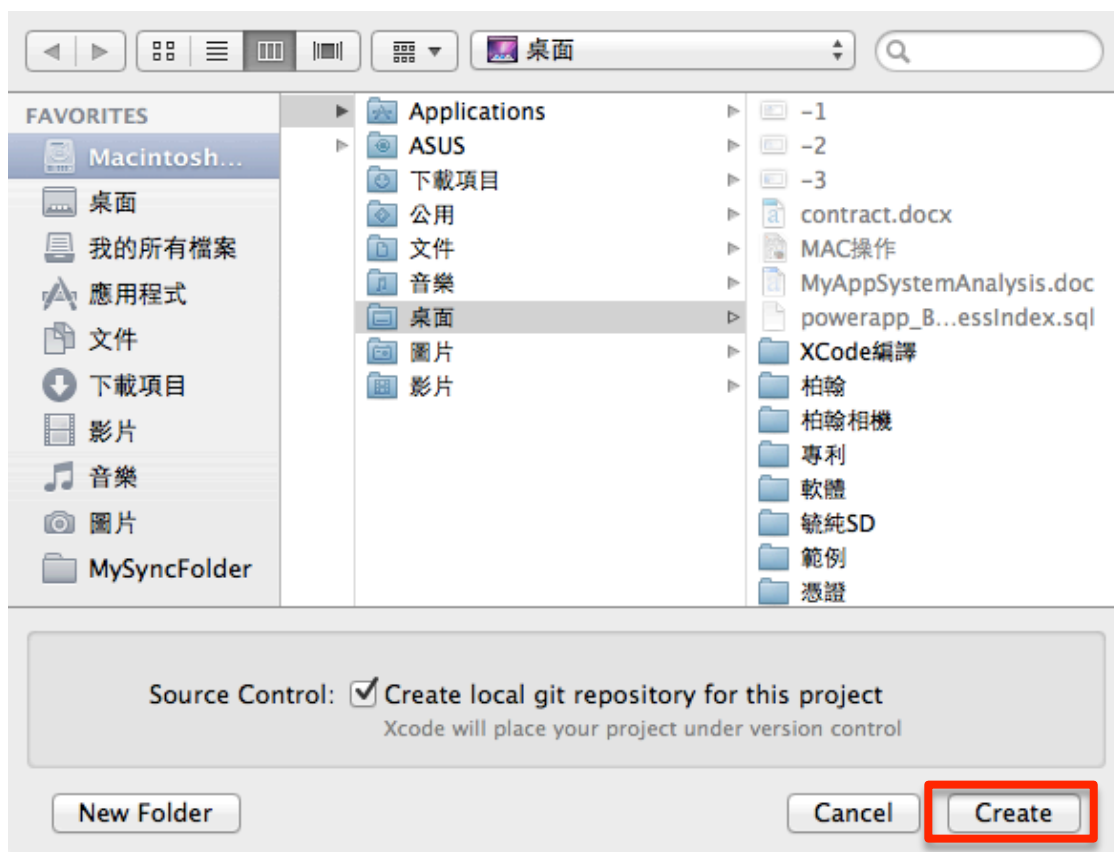
Use Storyboard

Use Automatic Reference Counting

Include Unit Tests

取消使用
Storyboard以及
Automatic Reference Counting(ARC)

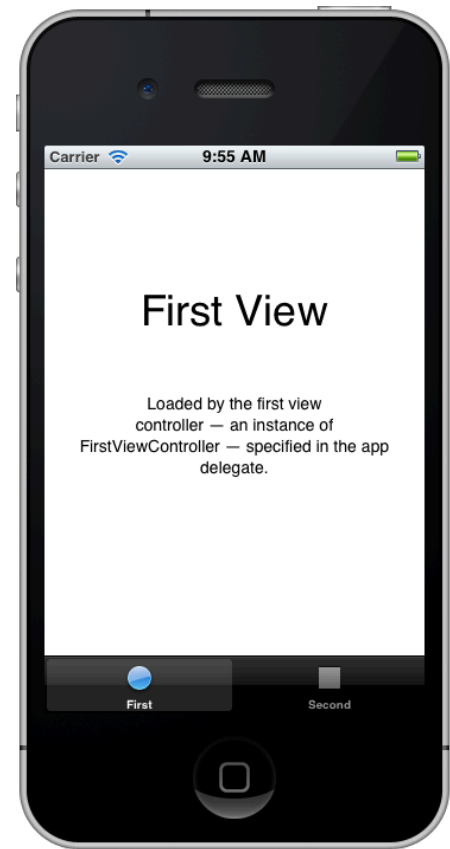
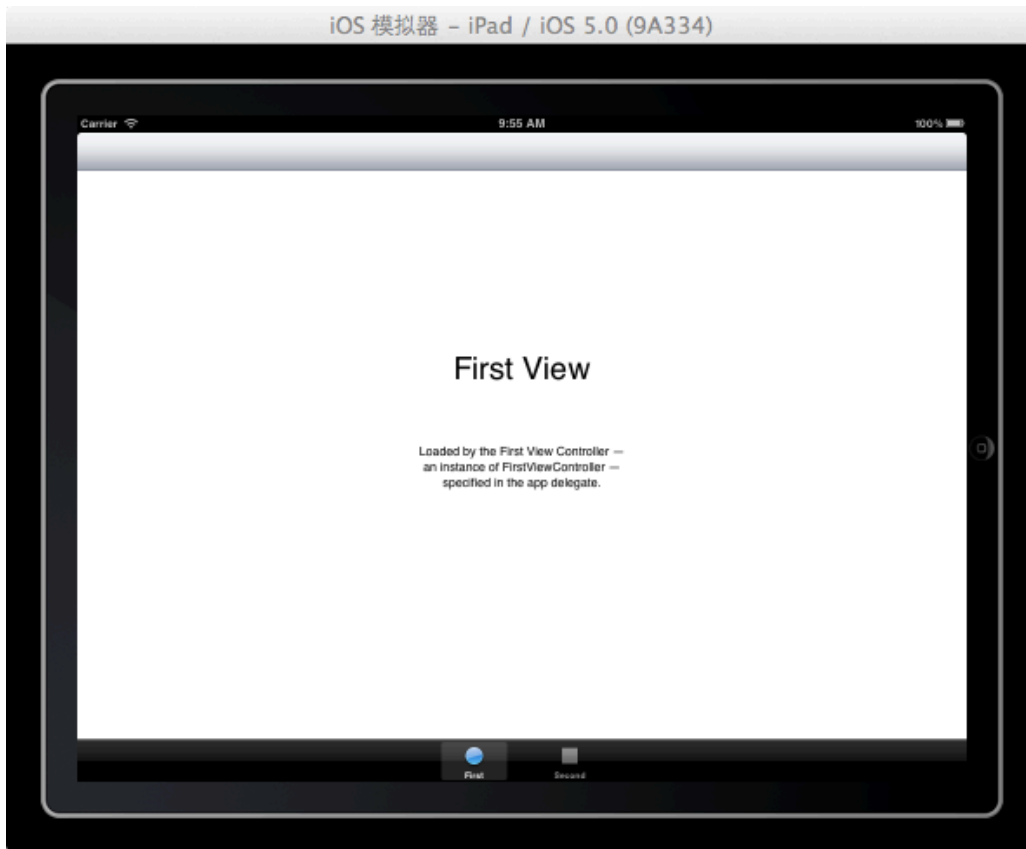
Step 4 – 選擇新專案放置目錄



Step 5 - 開啓新專案

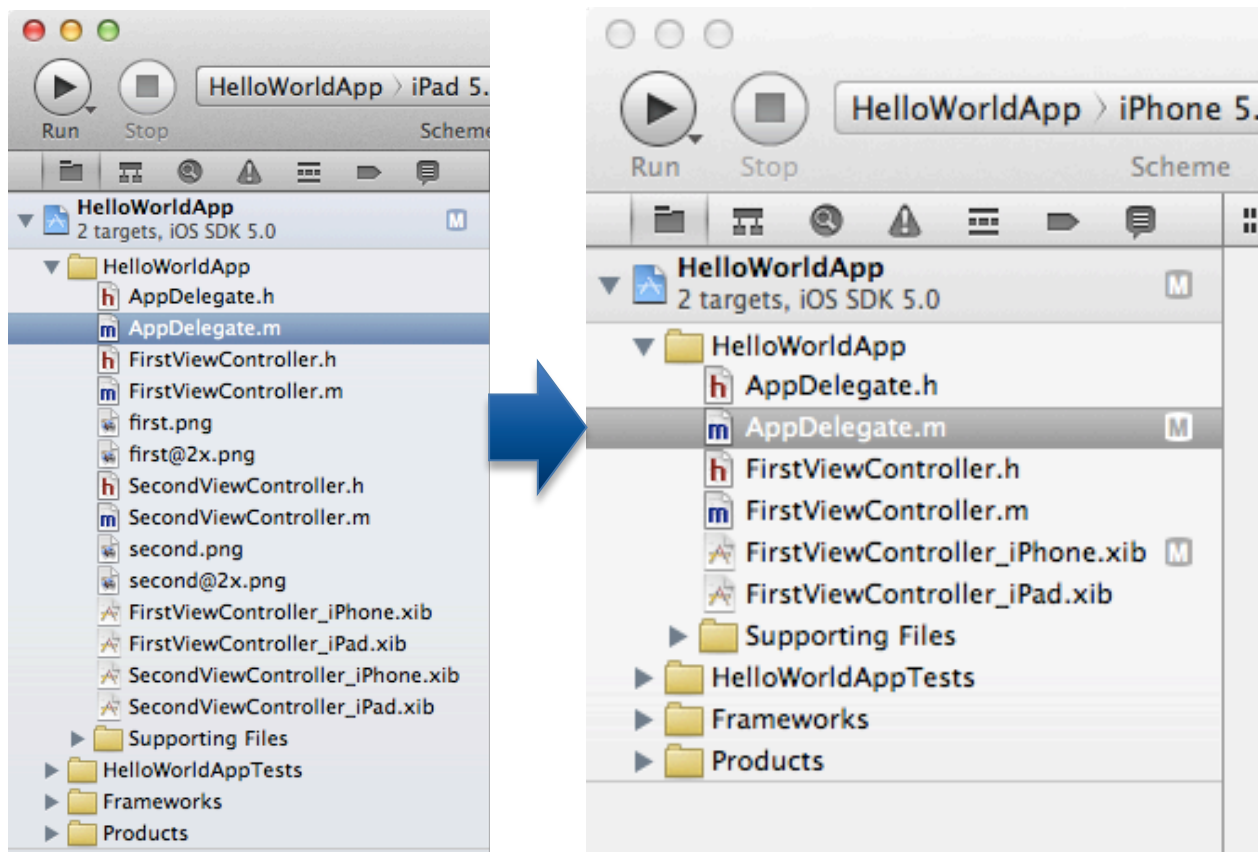


Step 6 - 執行結果



Step 7 - 進階修改

- ▶ 刪除不需要的檔案



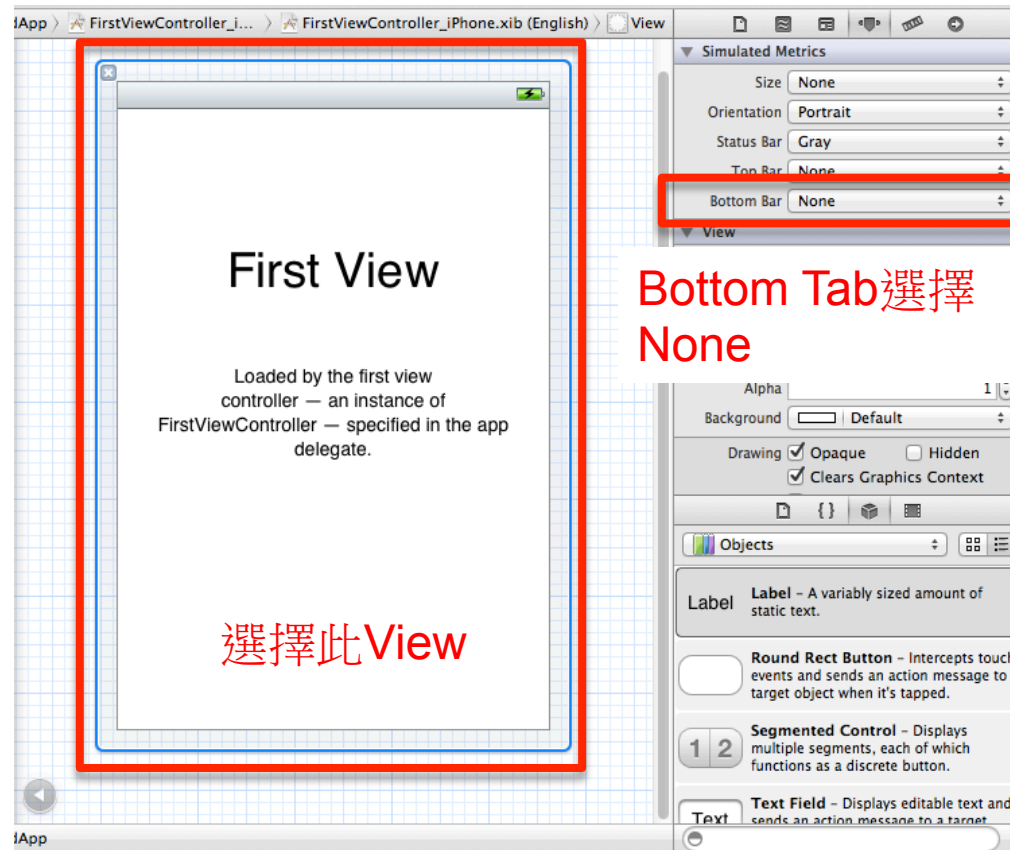
Step 7 - 進階修改

▶ 修改AppDelegate.m

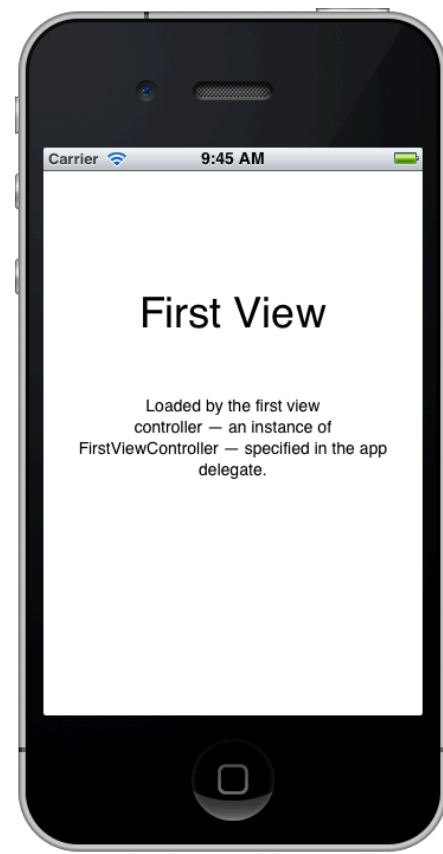
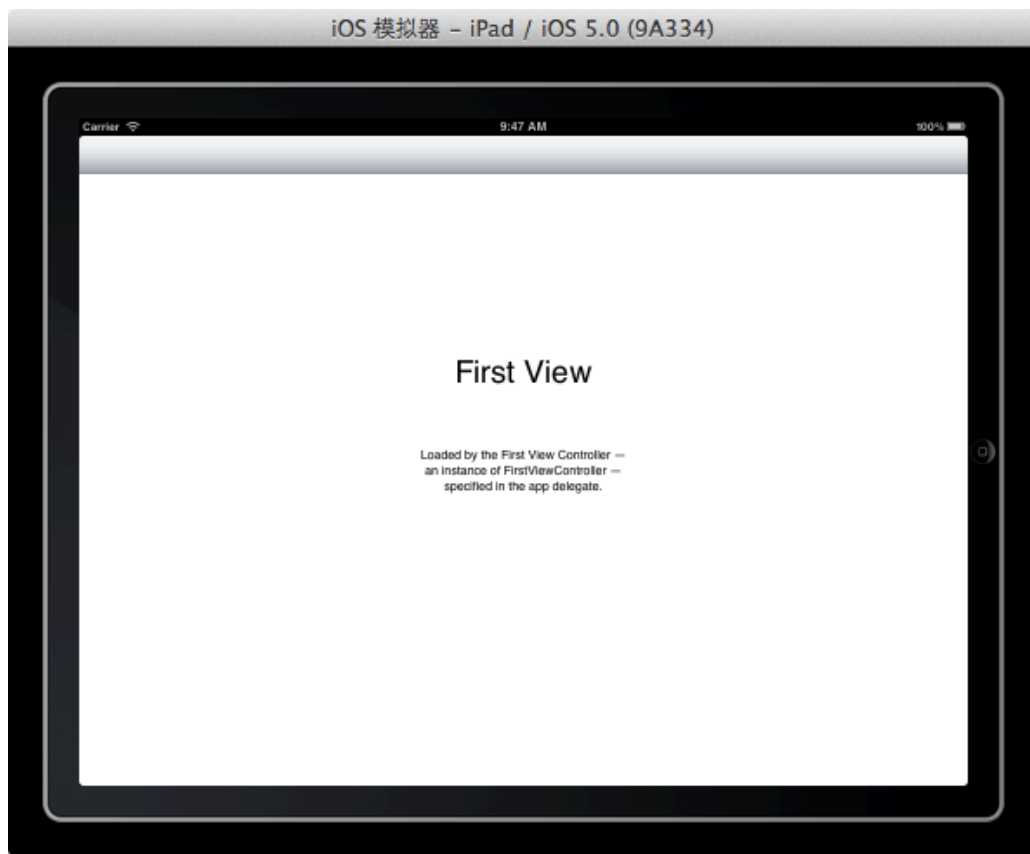
```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)
  launchOptions
{
    self.window = [[[UIWindow alloc] initWithFrame:[[UIScreen mainScreen] bounds]] autorelease];
    // Override point for customization after application launch.
    UIViewController *viewController1;
    if ([[UIDevice currentDevice] userInterfaceIdiom] == UIUserInterfaceIdiomPhone) {
        viewController1 = [[[FirstViewController alloc] initWithNibName:@"FirstViewController_iPhone"
            bundle:nil] autorelease];
    } else {
        viewController1 = [[[FirstViewController alloc] initWithNibName:@"FirstViewController_iPad"
            bundle:nil] autorelease];
    }
    self.window.rootViewController = viewController1;
    [self.window makeKeyAndVisible];
    return YES;
}
```

Step 7 – 進階修改

▶ 修改FirstViewController_iPhone.xib



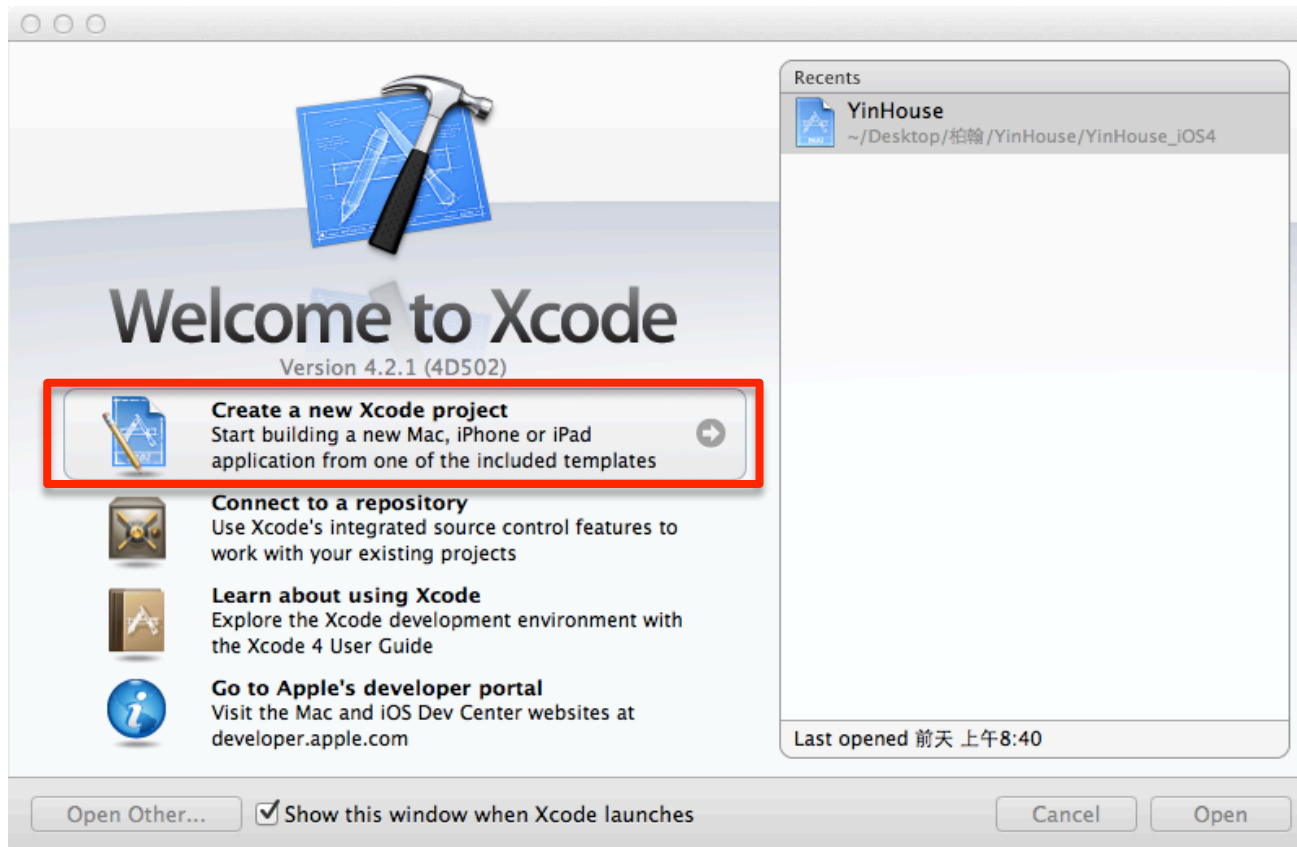
Step 7 - 進階修改 (完成)



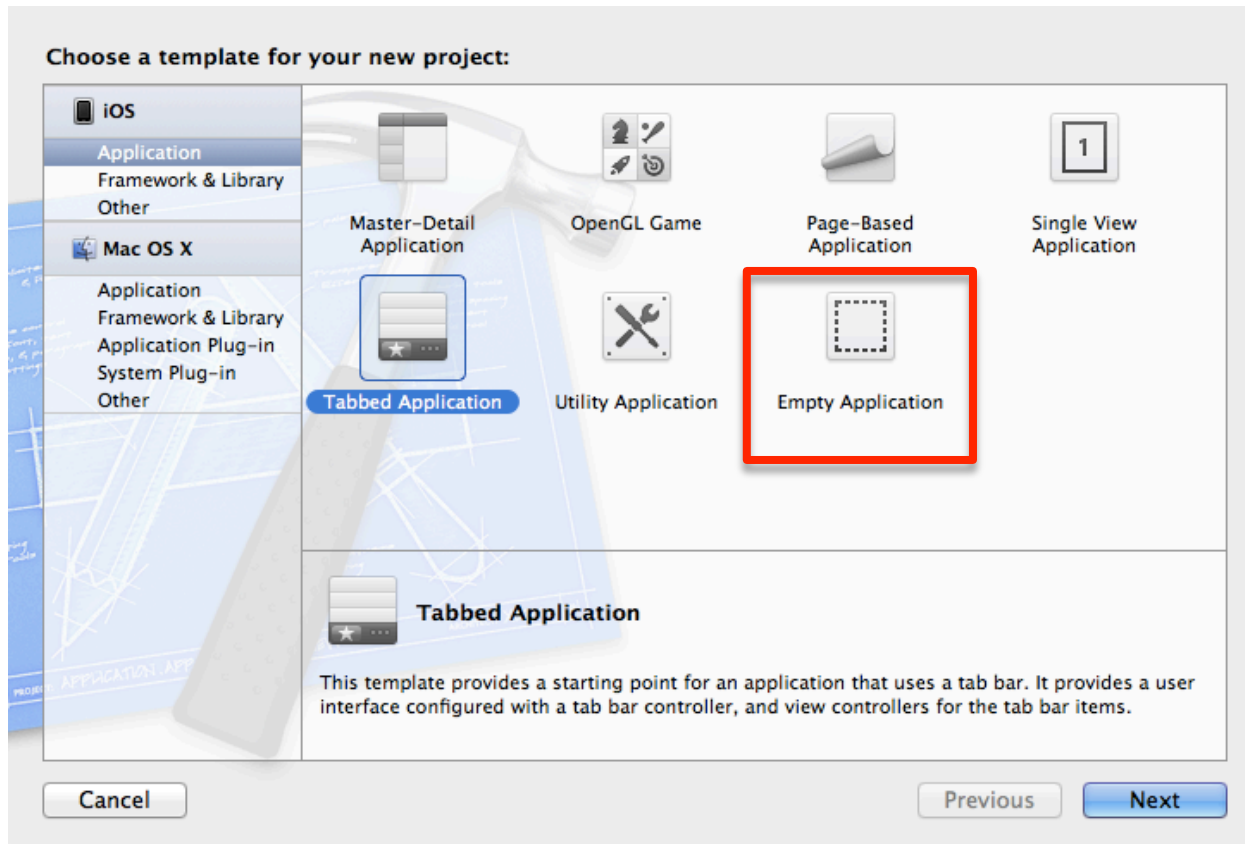
iOS 4.2 由空專案建立專案

- ▶ 學習目的
 - 建立空專案，並進行程式撰寫
 - 加入所需的畫面元件

Step 1 - 建立新專案



Step 2 – 選擇新專案類型



Step 3 - 設定新專案資料

Choose options for your new project:

Product Name

Company Identifier

Bundle Identifier

Class Prefix

Device Family

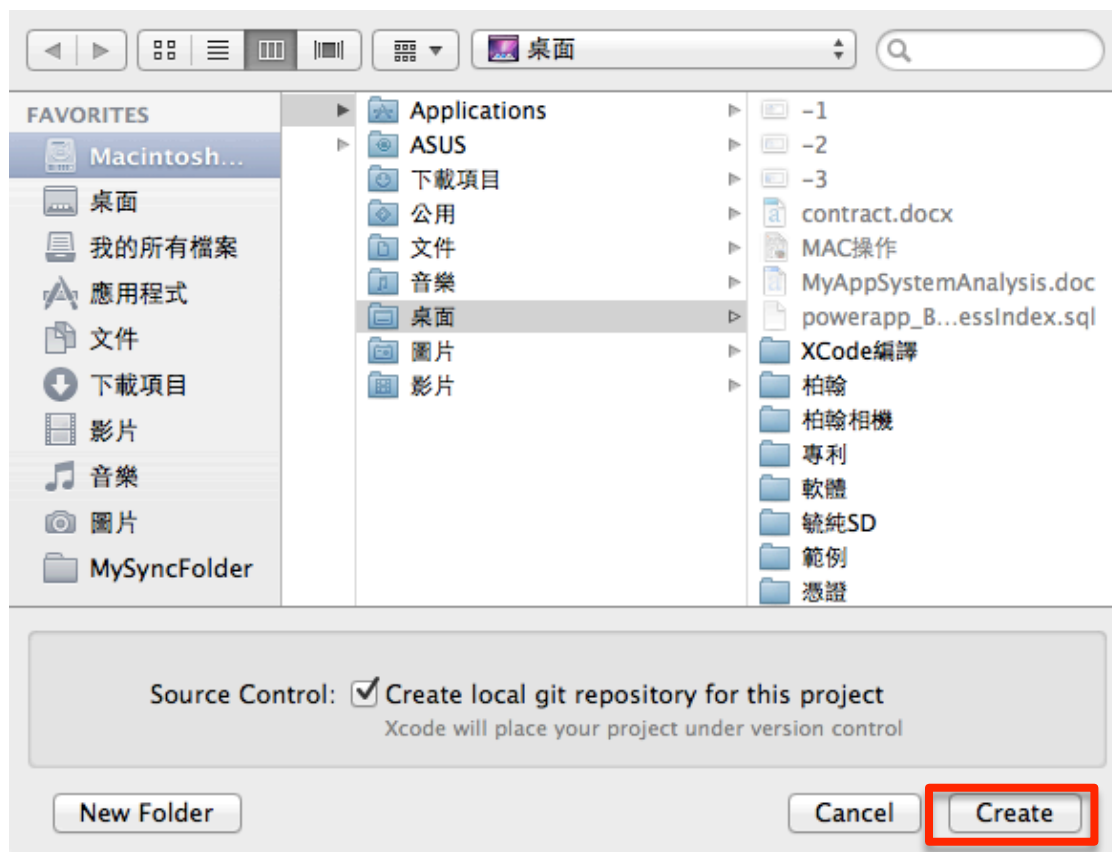
Use Core Data

Use Automatic Reference Counting

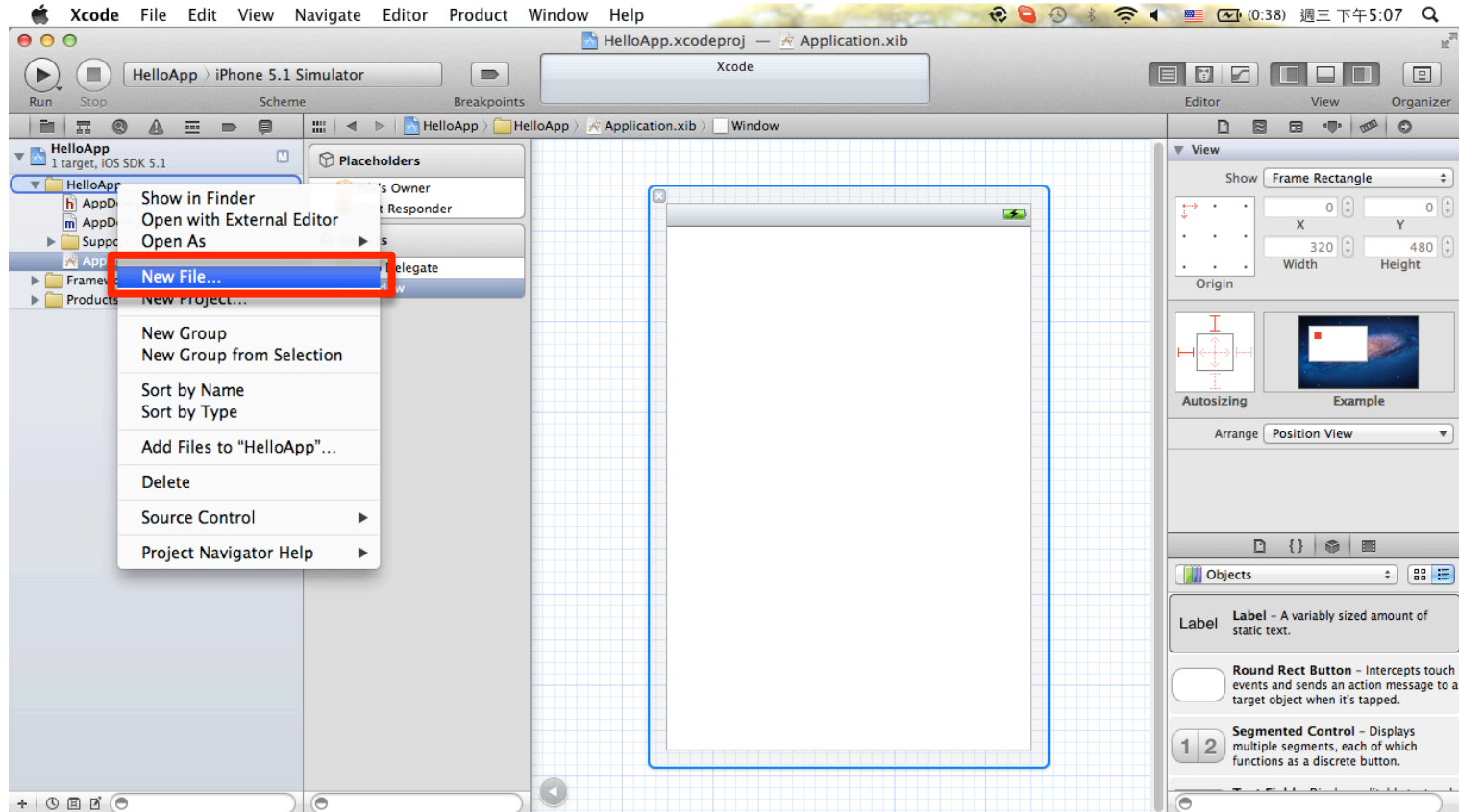
Include Unit Tests

取消使用
Storyboard 以及
Automatic Reference Counting(ARC)

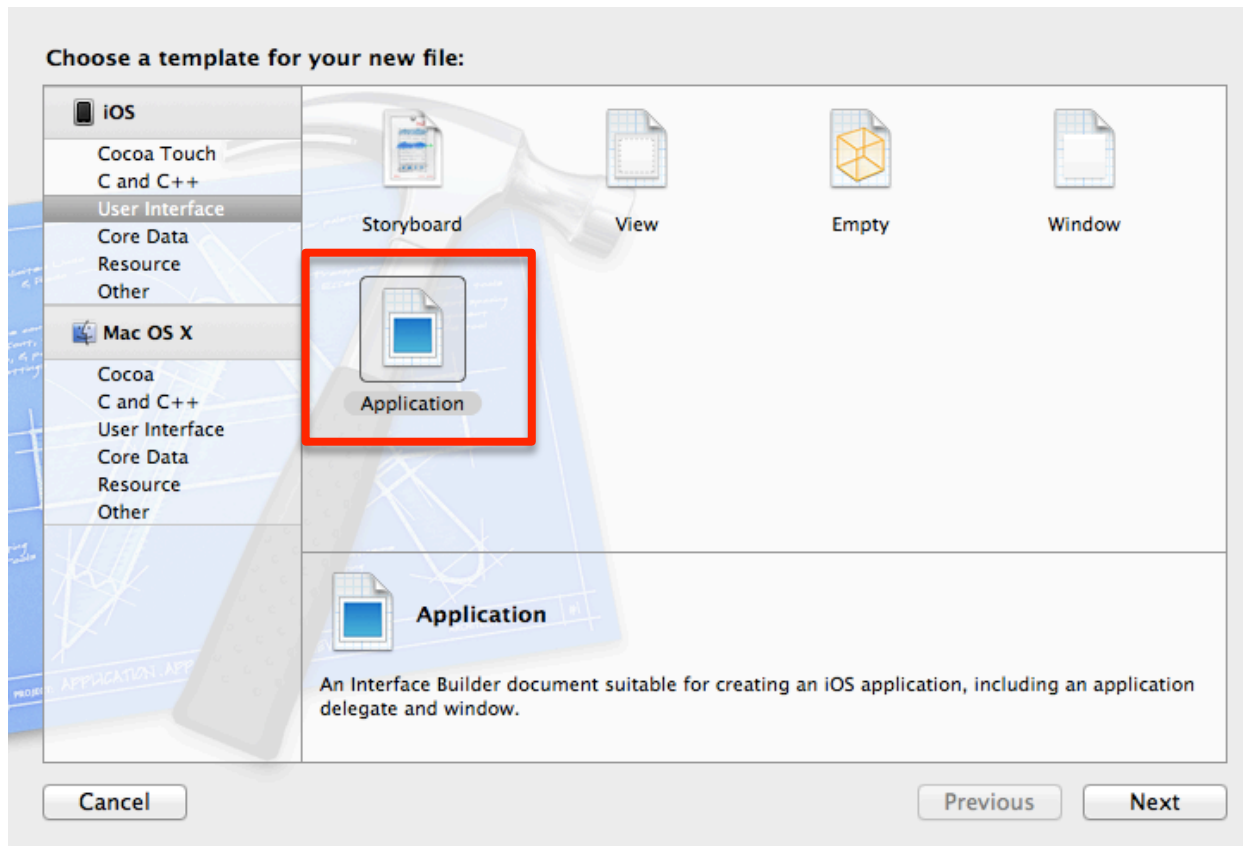
Step 4 – 選擇新專案放置目錄



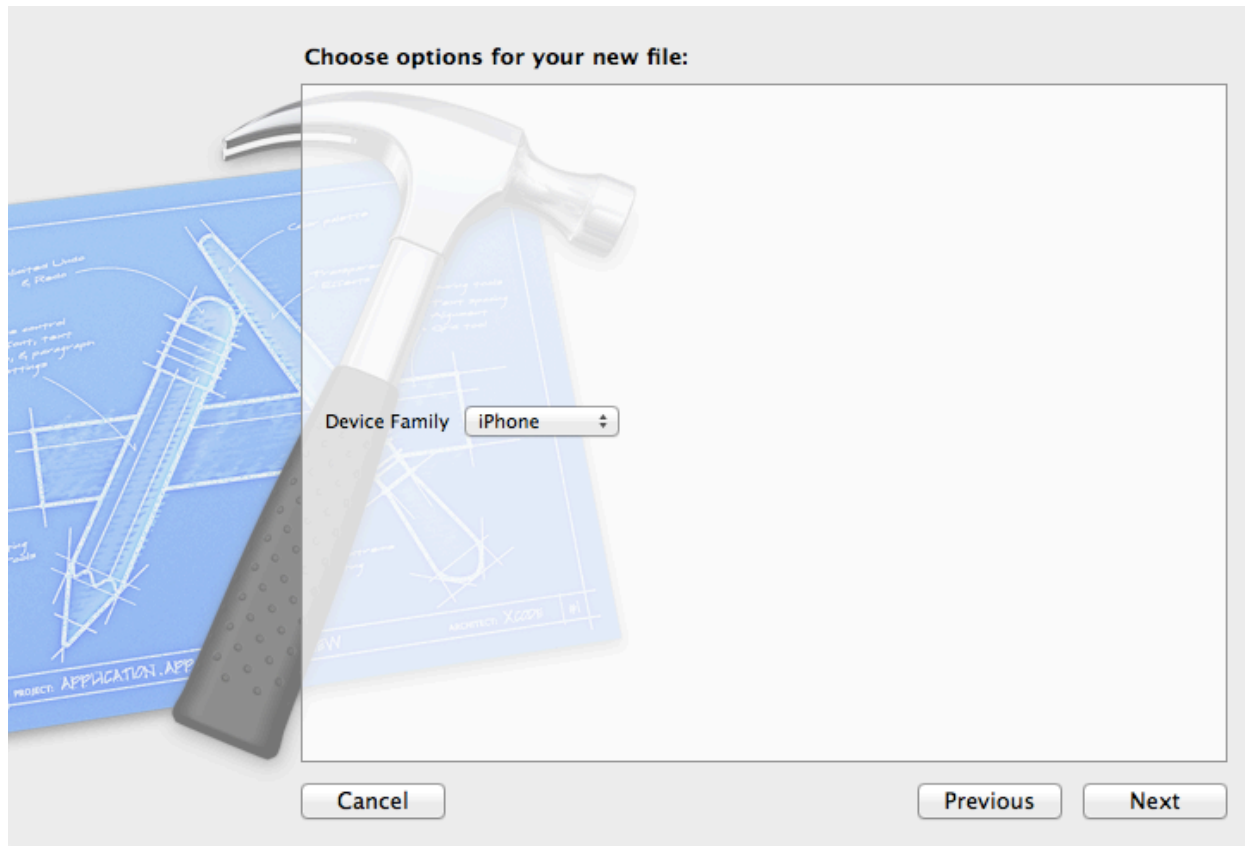
加入新檔案



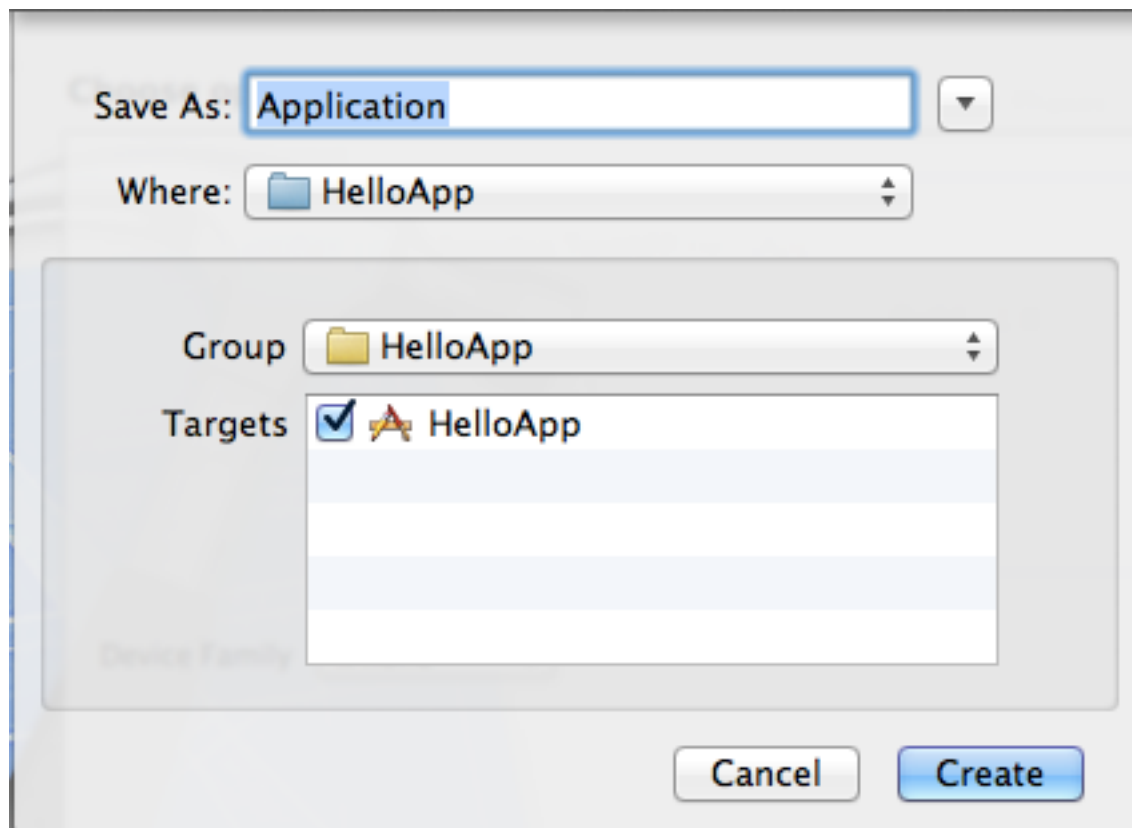
選擇Application



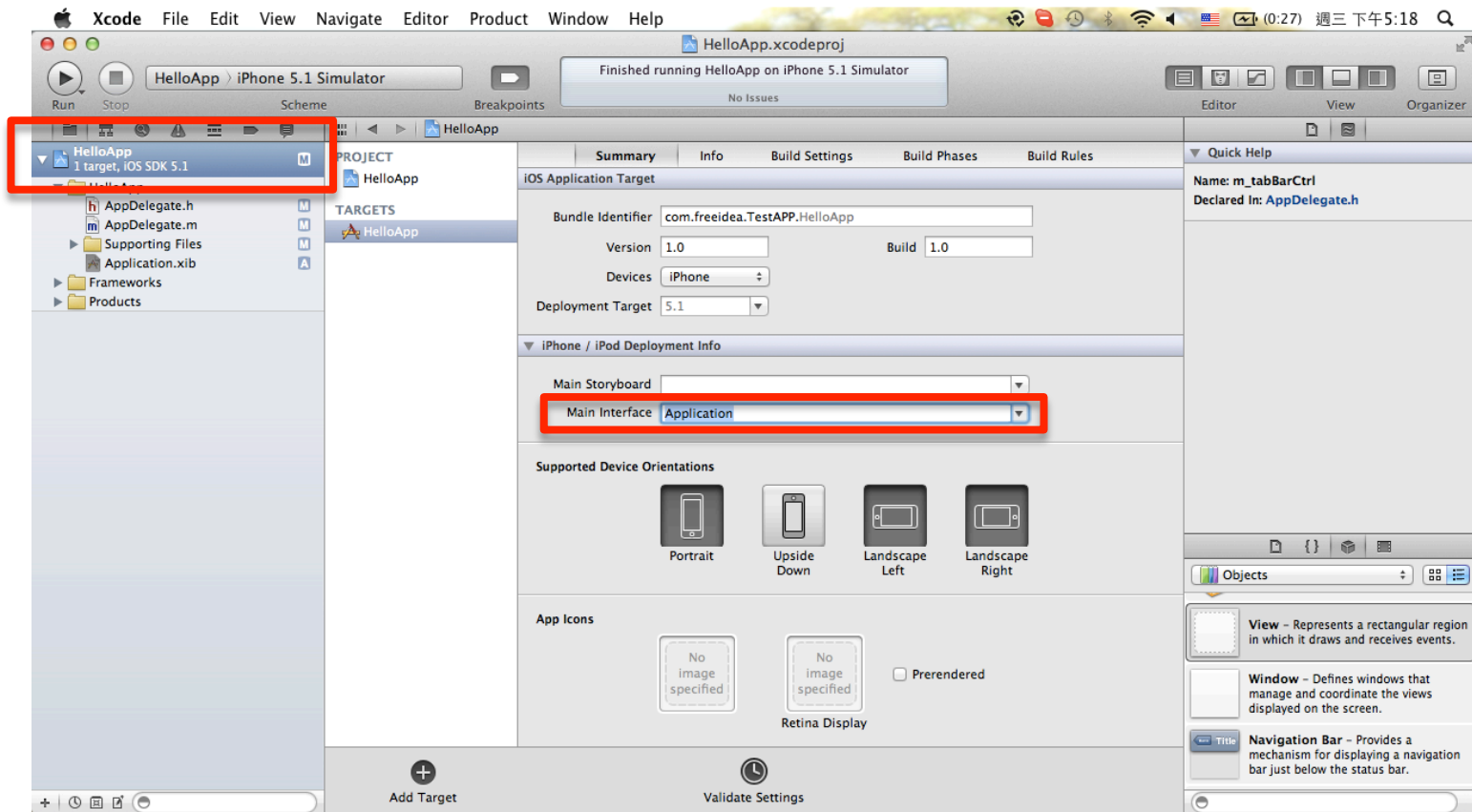
選擇Device



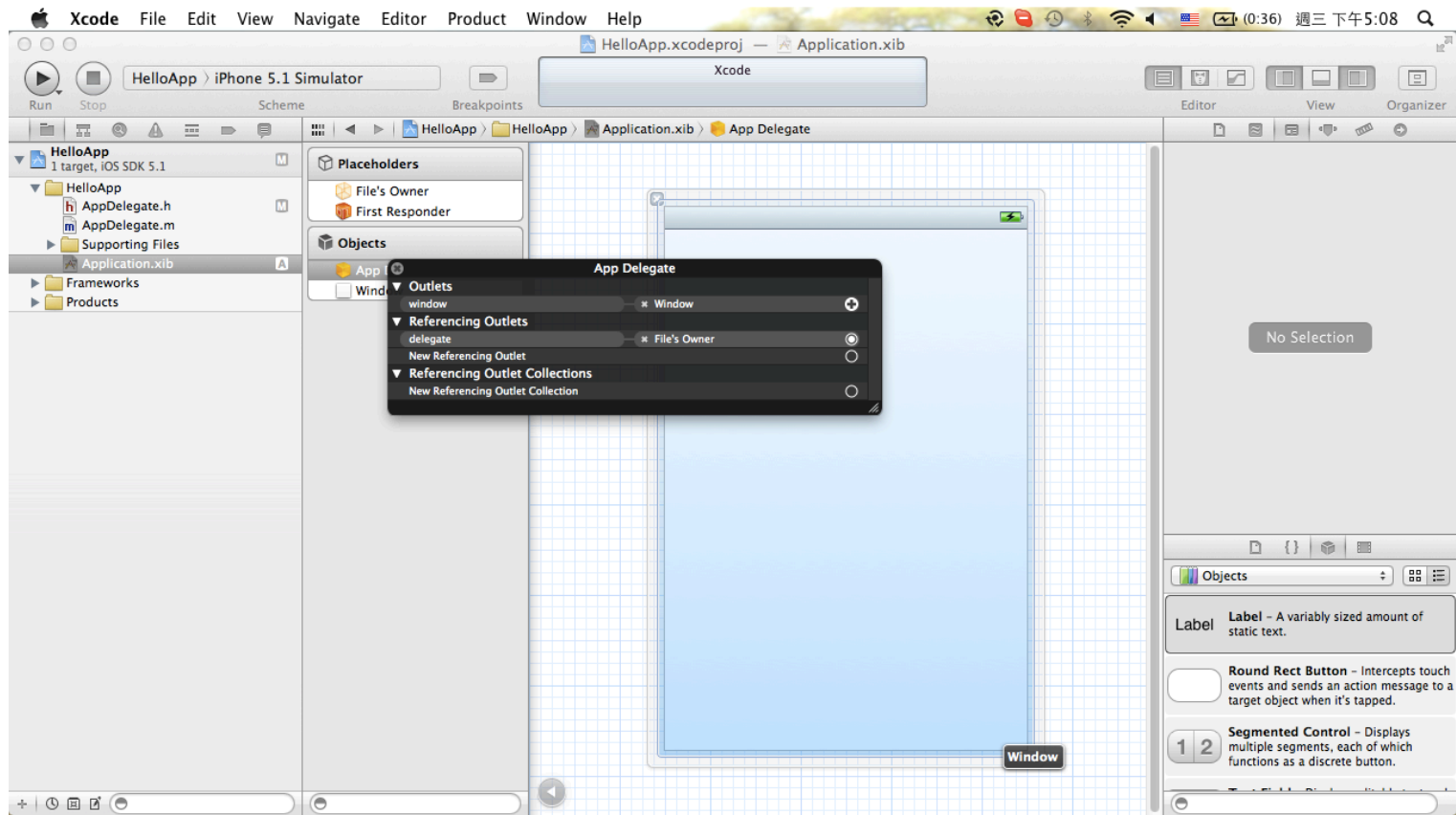
儲存



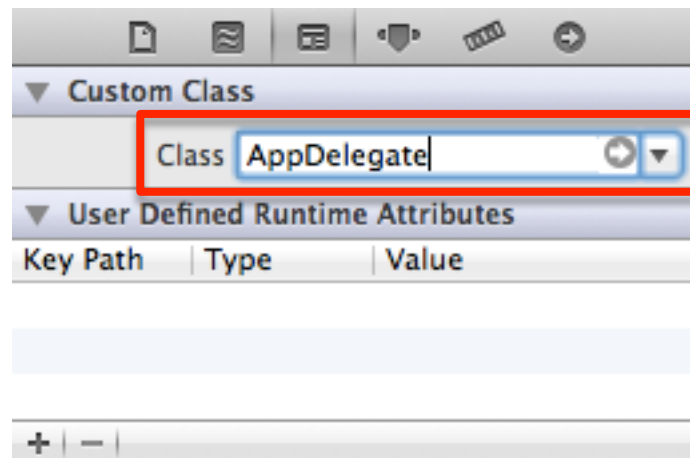
設定Main Interface



Application.xib



設定AppDelegate的Class



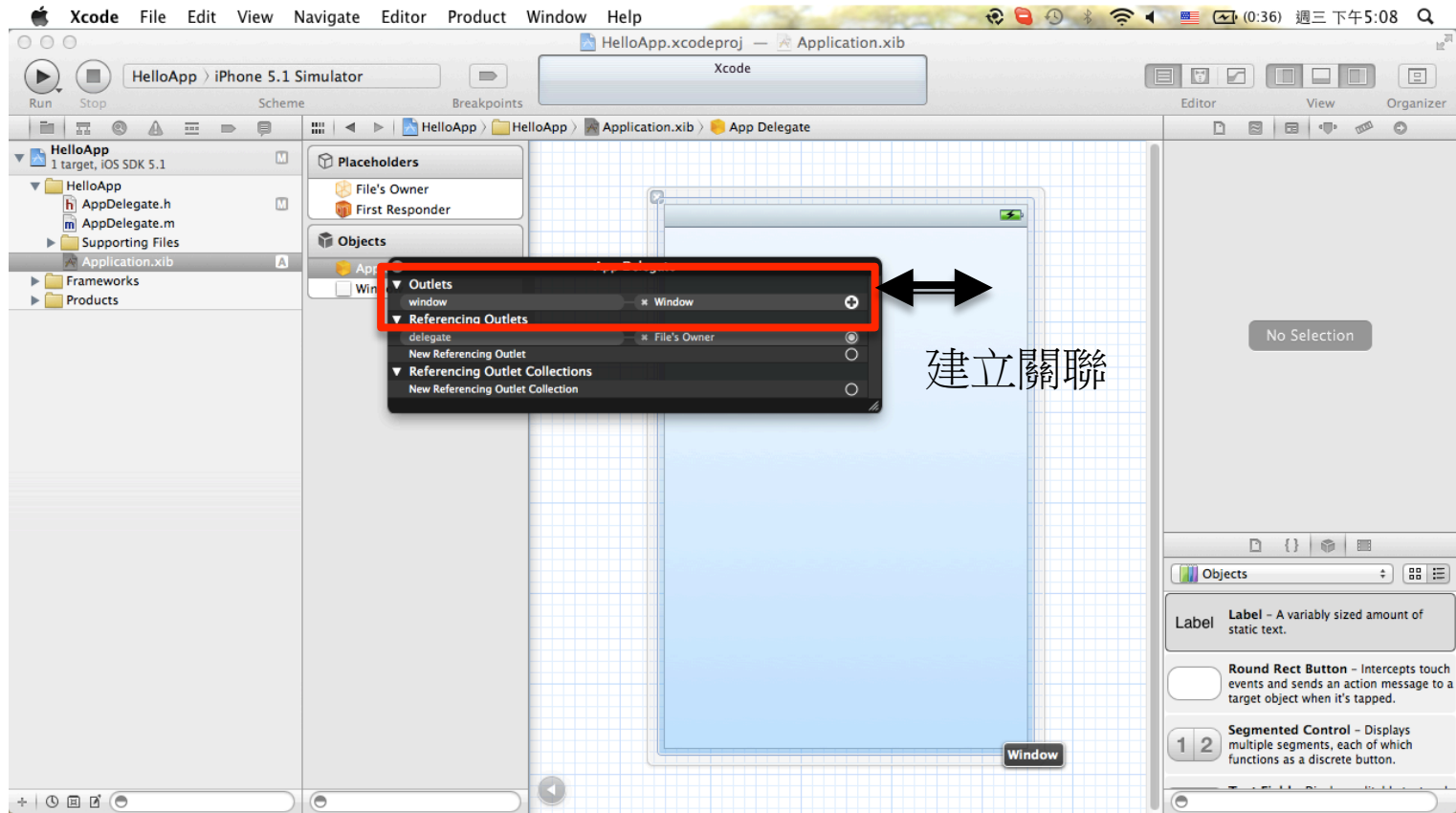
增加IBOutlet – window

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

@interface AppDelegate : UIResponder <UIApplicationDelegate>
{
    IBOutlet UIWindow *window;
}
@property (strong, nonatomic) UIWindow *window;

@end
```

建立window關聯



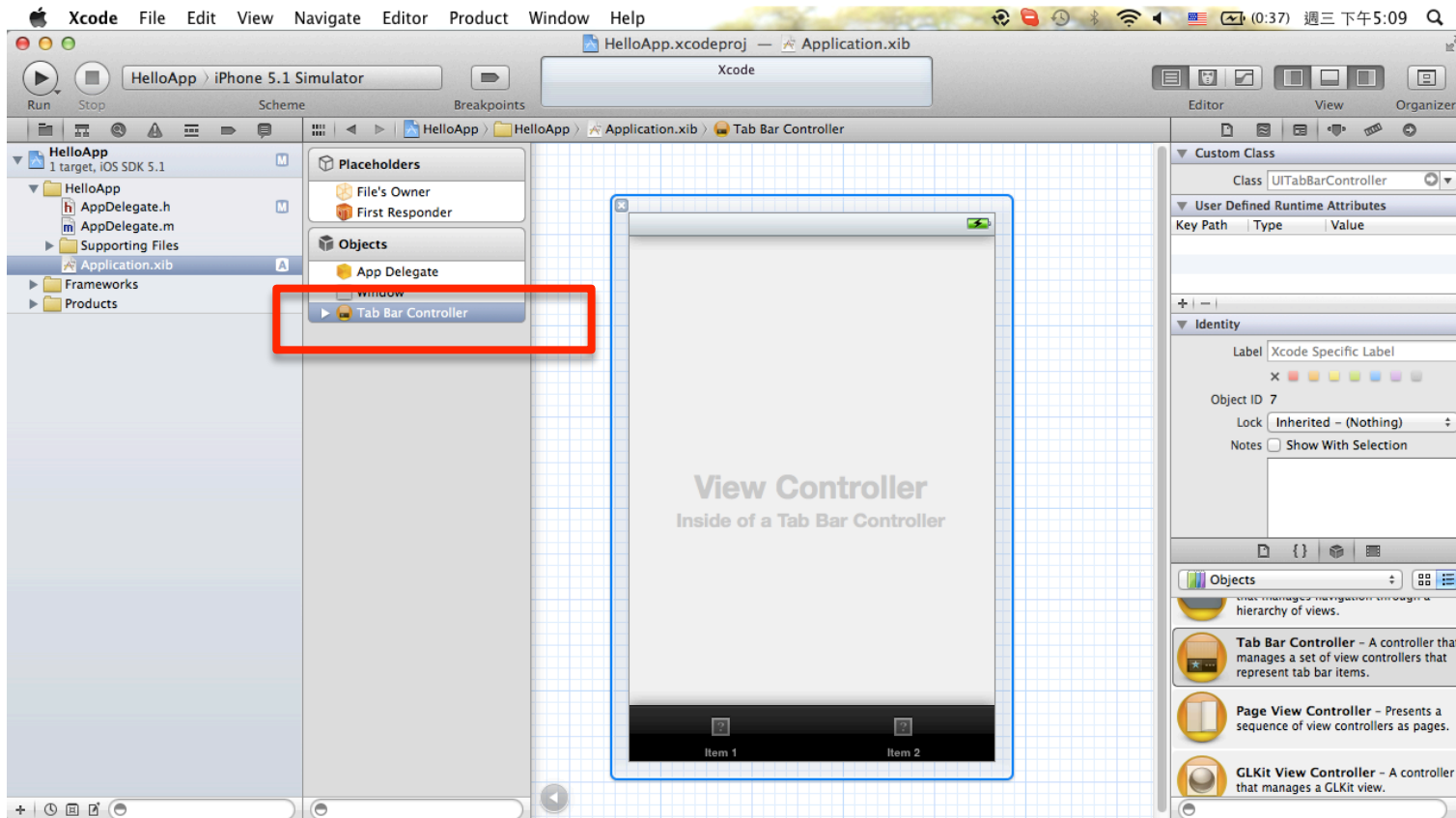
增加TabBarController

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

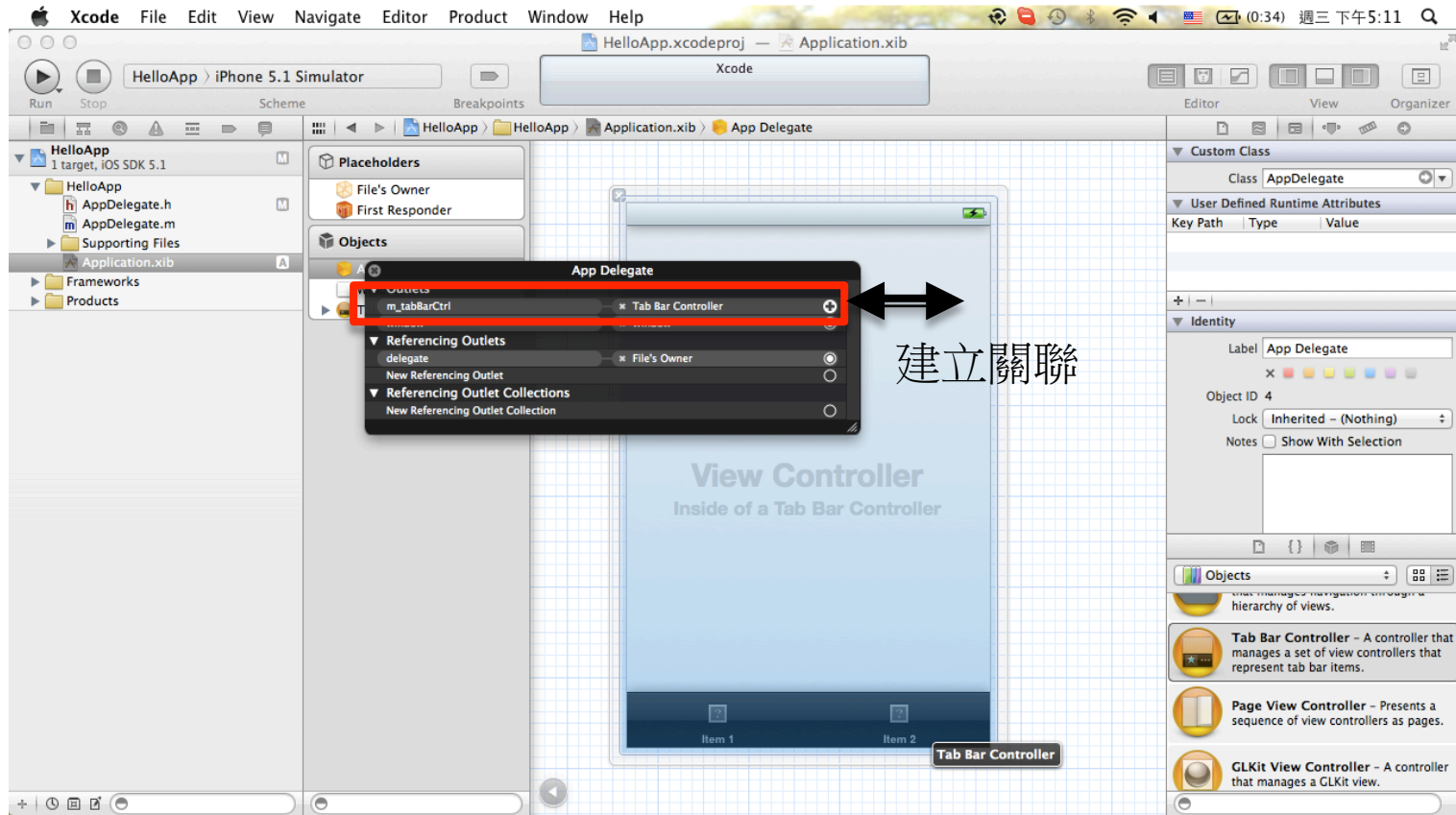
@interface AppDelegate : UIResponder <UIApplicationDelegate>
{
    IBOutlet UIWindow *window;
    IBOutlet UITabBarController *m_tabBarCtrl;
}
@property (strong, nonatomic) UIWindow *window;

@end
```


於Application.xib中增加



建立關聯



加入顯示語法

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)  
    launchOptions  
{  
    // Override point for customization after application launch.  
    [self.window addSubview:m_tabBarCtrl.view];  
    [self.window makeKeyAndVisible];  
    return YES;  
}
```

增加觸發語法

```
- (void)tabBarController:(UITabBarController *)tabBarController didSelectViewController:  
    (UIViewController *)viewController {  
    NSLog(@"%d", tabBarController.selectedIndex);  
}
```

增加觸發關聯

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

@interface AppDelegate : UIResponder <UIApplicationDelegate, UITabBarControllerDelegate>
{
    IBOutlet UIWindow *window;
    IBOutlet UITabBarController *m_tabBarCtrl;
}
@property (strong, nonatomic) UIWindow *window;

@end
```

Introduction to Object-C

- ▶ 函式呼叫
 - #include -> #import
 - object.fun() -> [object fun];
- ▶ 記憶體配置
 - Object *obj = (Object *)malloc(1*sizeof(Object));
 - Object *obj = [[Object alloc] init];
- ▶ Self
 - 自身記憶體參照
 - 呼叫變數時不一定使用，使用時為指定類別公用變數
 - 呼叫函式時必要，因此須設定
- ▶ NSLog(@"format", ...);
 - 於終端機印出資料
 - %@: 列印物件

Introduction to Object-C

► .h

引用函式庫

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

繼承

繼承類別屬性

```
@interface FirstViewController : UIViewController <UITableViewDelegate, UITableViewDataSource>
{
    IBOutlet id delegate;
    IBOutlet UILabel *label;
    NSInteger local_type;
    NSInteger _global_type;
}
```

類別私有變數宣告

```
@property (nonatomic, assign) id delegate;
@property (nonatomic, retain) UILabel *label;
@property (nonatomic, assign) NSInteger global_type;
```

類別公用變數宣告

```
- (void) can_callout;
- (NSInteger) getType;
@end
```

類別公用函式

```
@protocol FirstViewControllerDelegate
@required
- (void) callback_required:(NSInteger) i_type;
@optional
- (void) callback_optional:(NSInteger) i_type;
@end
```

類別類別屬性定義

Introduction to Object-C

▶ .m

```
#import "FirstViewController.h"
```

```
@interface FirstViewController ()  
{  
    NSInteger i_default;  
}  
  
- (NSInteger) addWithA:(NSInteger)inA B:(NSInteger)inB;  
  
@end
```

類別私有變數宣告

類別私有函式

```
@implementation FirstViewController
```

```
@synthesize delegate;  
@synthesize label;  
@synthesize global_type = _global_type;
```

類別公用變數連結

Introduction to Object-C

▶ .m

```
- (void)can_callout {
    //do
}

- (NSInteger)getType {
    return self.global_type;
}

- (NSInteger) addWithA:(NSInteger)inA B:(NSInteger)inB {
    return inA+inB+i_default;
}

- (IBAction)btnAction:(id)sender {
    //do action
}

- (IBAction)btnAction2:(UIButton *)sender {
    //do action
}
```

Introduction to Object-C

- ▶ (NSDictionary) NSMutableDictionary
 - 儲存池 (可變內容儲存池)
 - 可擺放任何類別的資料
 - 須設定取回資料的key
- ▶ NSArray (NSMutableArray)
 - 陣列 (可變大小陣列)
 - 可擺放任何類別的資料
 - 可交錯擺放
- ▶ NSString (NSMutableString)
 - 字串 (可變字串)

Introduction to Object-C

```
NSArray *arr = [NSArray arrayWithObjects:@"arr1", @"arr2", nil];

NSMutableArray *marr = [[NSMutableArray alloc] initWithCapacity:0];
[marr addObject:@"marr1"];
[marr addObject:@"marr2"];
[marr addObjectsFromArray:arr];

NSDictionary *dict = [NSDictionary dictionaryWithObjects:[NSArray arrayWithObjects:@"obj1", @"obj2", nil]
                    forKeys:[NSArray arrayWithObjects:@"key1", @"key2", nil]];
[marr addObject:dict];

NSMutableDictionary *mdict = [[NSMutableDictionary alloc] initWithCapacity:0];
[mdict setObject:@"obj1" forKey:@"key1"];
[mdict setObject:@"obj2" forKey:@"key2"];
[marr addObject:mdict];

NSString *str = [NSString stringWithFormat:@"%d", 5];
[marr addObject:str];

NSMutableString *mstr = [NSMutableString stringWithFormat:@"%d", 5];
[mstr appendFormat:@"%c", '6'];
[mstr insertString:@"test" atIndex:0];
NSLog(@"%@", mstr);
[marr addObject:mstr];
NSLog(@"%@", marr);
```

```
All Output ↓ Clear [ ] [ ] [ ]
2012-07-22 12:40:48.068 TableViewSample[10040:f803]
test56
2012-07-22 12:40:48.069 TableViewSample[10040:f803] (
  marr1,
  marr2,
  arr1,
  arr2,
  {
    key1 = obj1;
    key2 = obj2;
  },
  {
    key1 = obj1;
    key2 = obj2;
  },
  5,
  test56
)
```

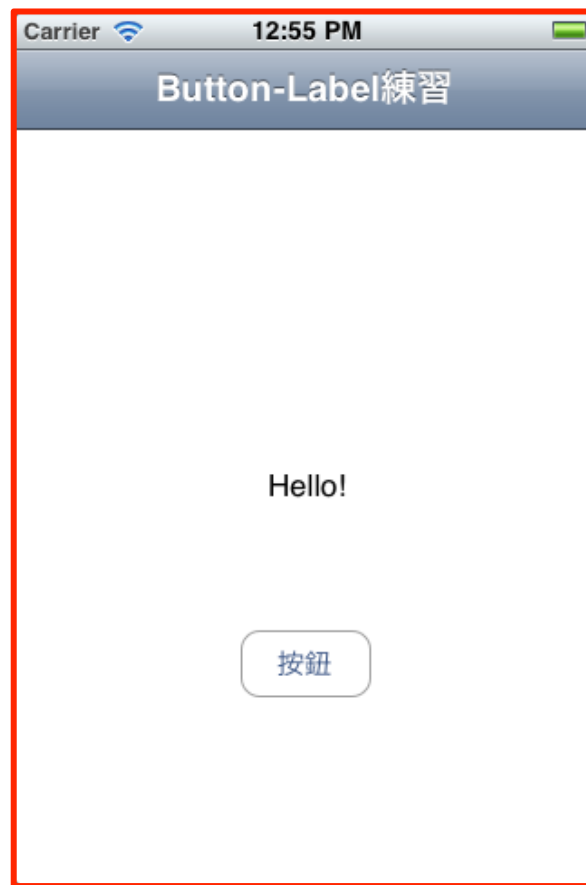
Introduction to Object-C

- ▶ 彈跳訊息框 (UIAlertView)
 - UIAlertView * errorAlert = [[UIAlertView alloc] initWithTitle:@"Title" message:@"Message" delegate:self cancelButtonTitle:@"OK" otherButtonTitles:nil];
 - [errorAlert show];
 - [errorAlert release];

元件練習UIButton+Label

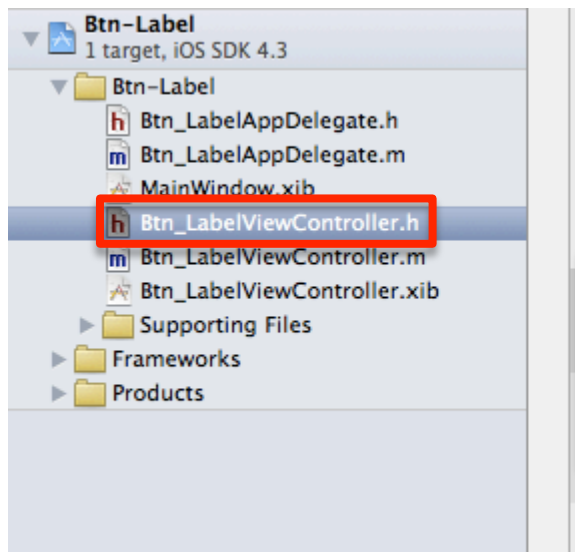
- ▶ 學習目的
 - 使用基本的元件: 按鈕及文字框
 - 單純的物件與變數連結
 - 使用元件與副函式連結

執行結果



定義介面參數(.h)

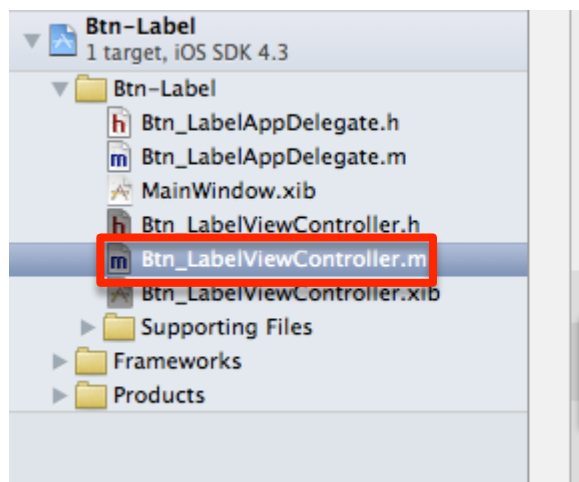
- ▶ 說明
 - IBOutlet: 可連結介面元件
 - IBAction: 可連結介面元件觸發
 - 定義UIButton類別的變數btn
 - 定義UILabel類別的變數lbl
 - 定義副函式Btn_Act()



```
//  
// Btn_LabelViewController.h  
// Btn-Label  
//  
// Created by cilab on 2011/8/5.  
// Copyright 2011年 __MyCompanyName__. All rights reserved.  
//  
  
#import <UIKit/UIKit.h>  
  
@interface Btn_LabelViewController : UIViewController {  
    IBOutlet UIButton *btn;  
    IBOutlet UILabel *lbl;  
}  
  
- (IBAction) Btn_Act;|  
  
@end
```

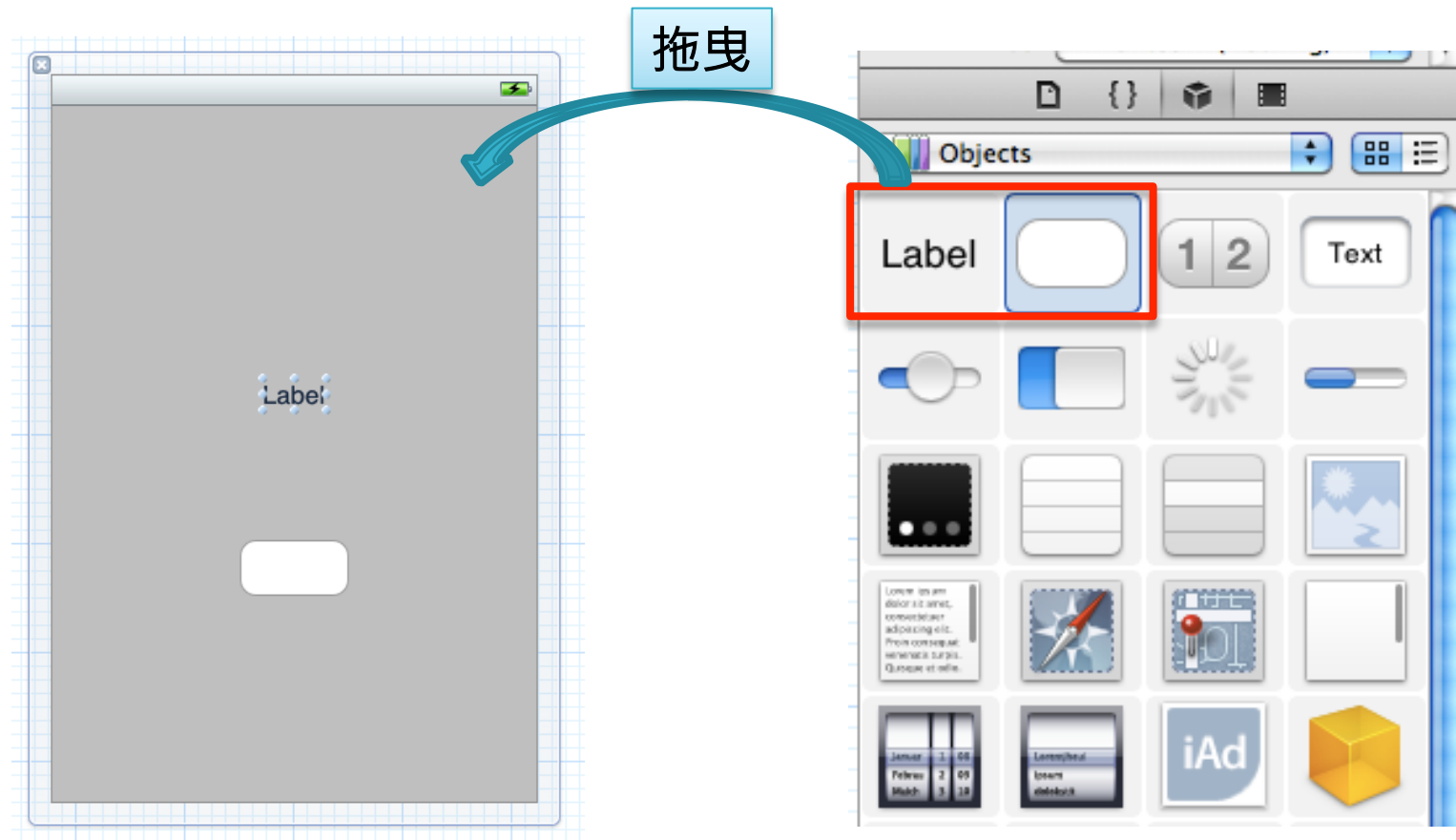
實作(.m)

▶ 實作Btn_Act函式



```
//  
// Btn_LabelViewController.m  
// Btn-Label  
//  
// Created by cilab on 2011/8/5.  
// Copyright 2011年 __MyCompanyName__. All rights reserved.  
//  
#import "Btn_LabelViewController.h"  
  
@implementation Btn_LabelViewController  
  
- (IBAction) Btn_Act {  
    lbl.text = @"Hello";  
}  
}
```

製作操作介面



連結程式定義與操作介面

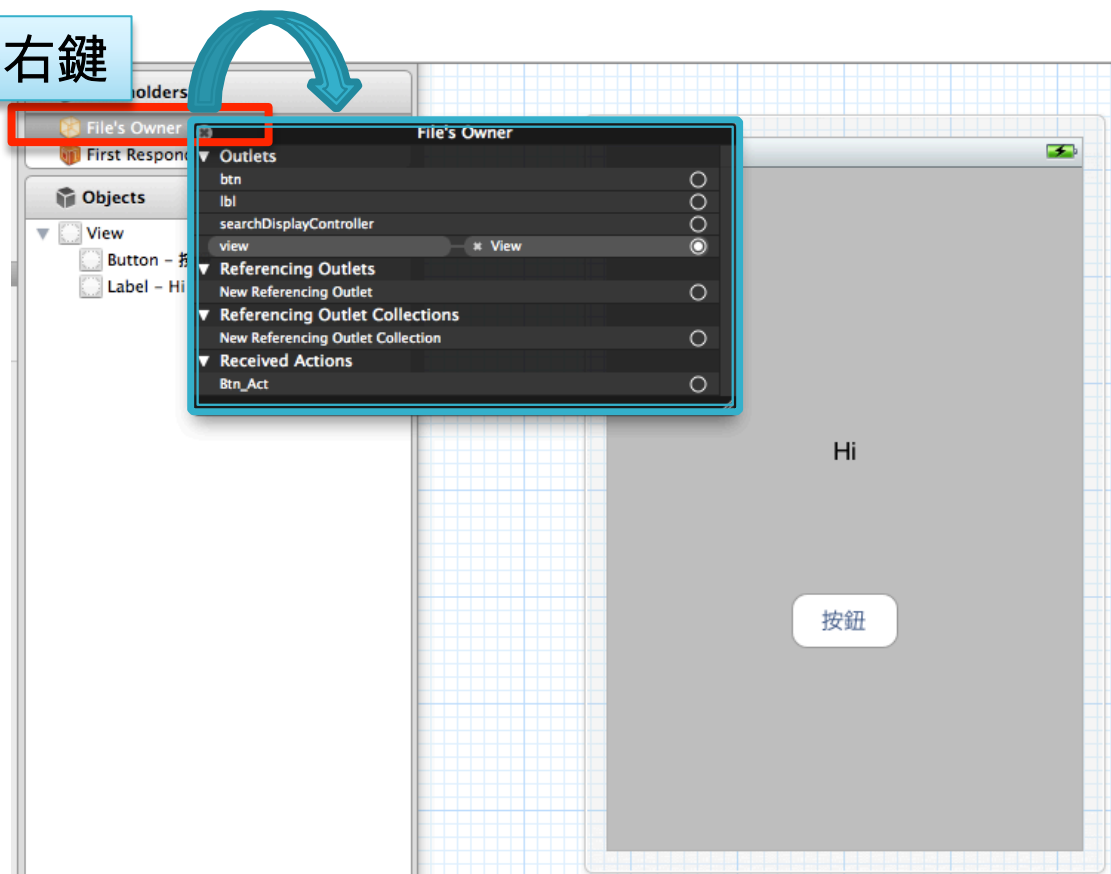
▶ IBOutlet

- 物件連結

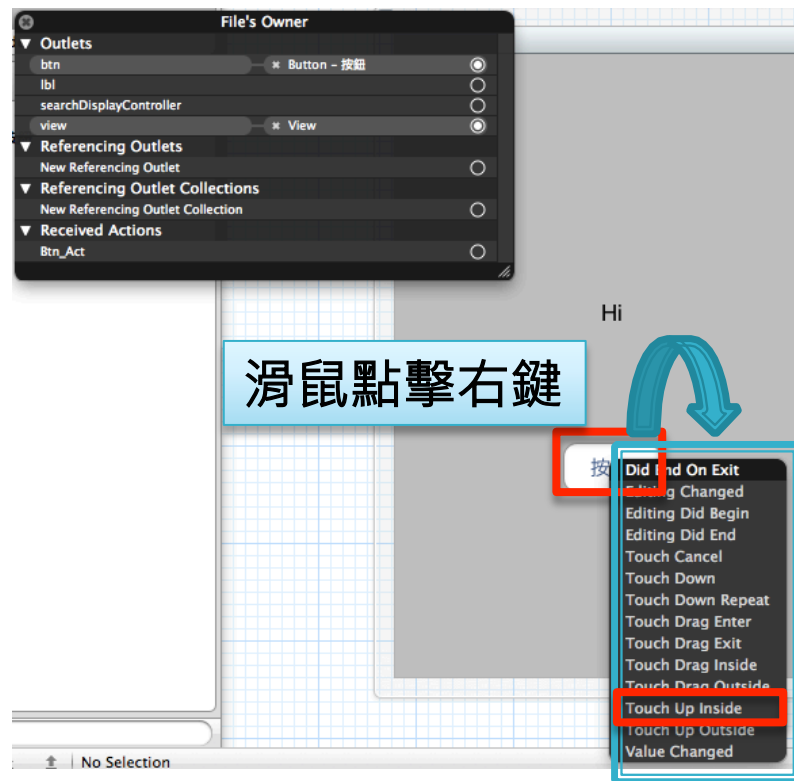
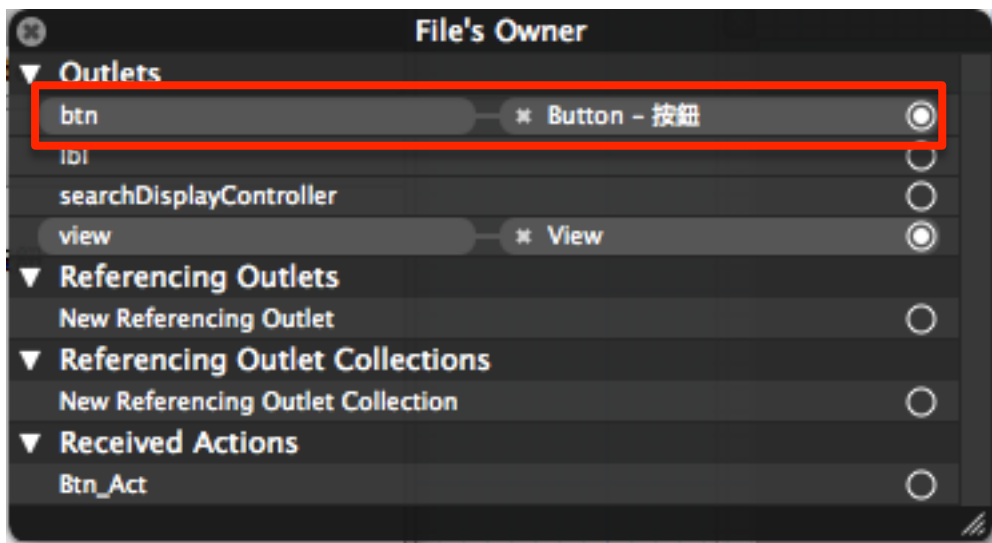
▶ 連結

- btn → Button元件
- lbl → Label元件

滑鼠點擊右鍵



連結程式定義與操作介面



▶ IBAction

- 觸發連結

▶ 連結

- Btn_Act → Button元件中的Touch Up Inside

備註

▶ UILabel常用的指令

- 設置 myLabel 的字體及並根據實際 iPhone / iPad 的屏幕設置字體大小

```
Label.font = [UIFont fontWithName:... size:...];
```

- 設置 myLabel 的文字內容

```
Label.text = ...;
```

- 設置 myLabel 的背景顏色為透明

```
Label.backgroundColor = [UIColor clearColor];
```

- 設置 myLabel 的文字顏色

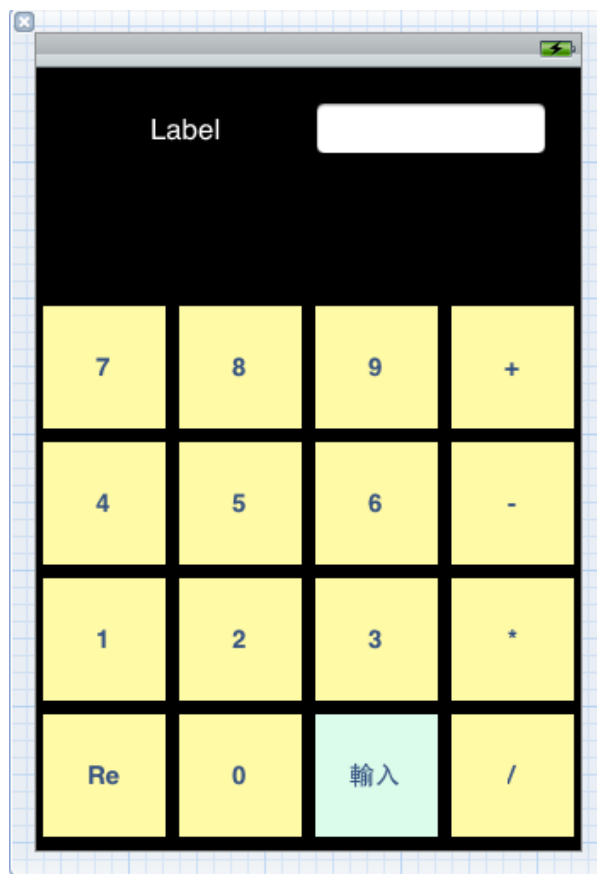
```
Label.textColor = ...;
```

- 設置 myLabel 的文本對齊方式為對齊中心

```
Label.textAlignment = NSTextAlignmentCenter;
```

實作練習

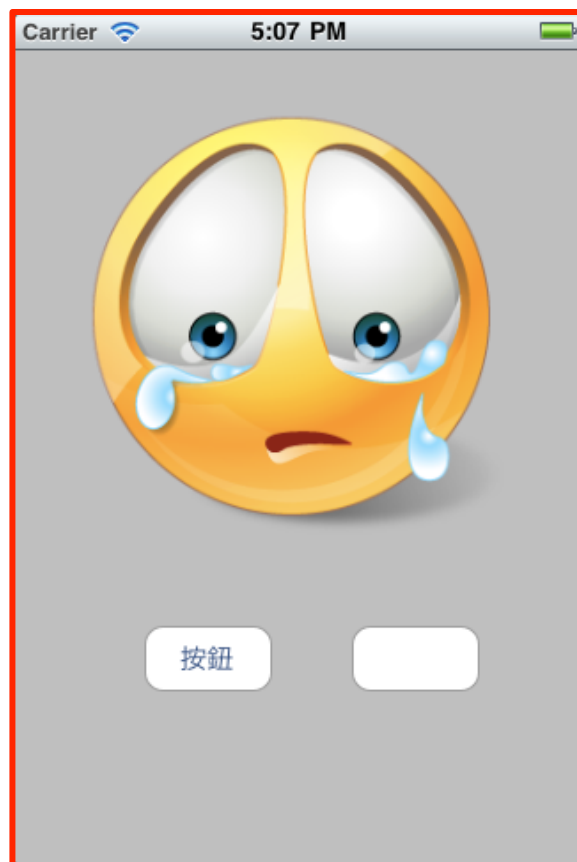
- ▶ 二元計算機(只有加減乘除)



元件練習UIImageView

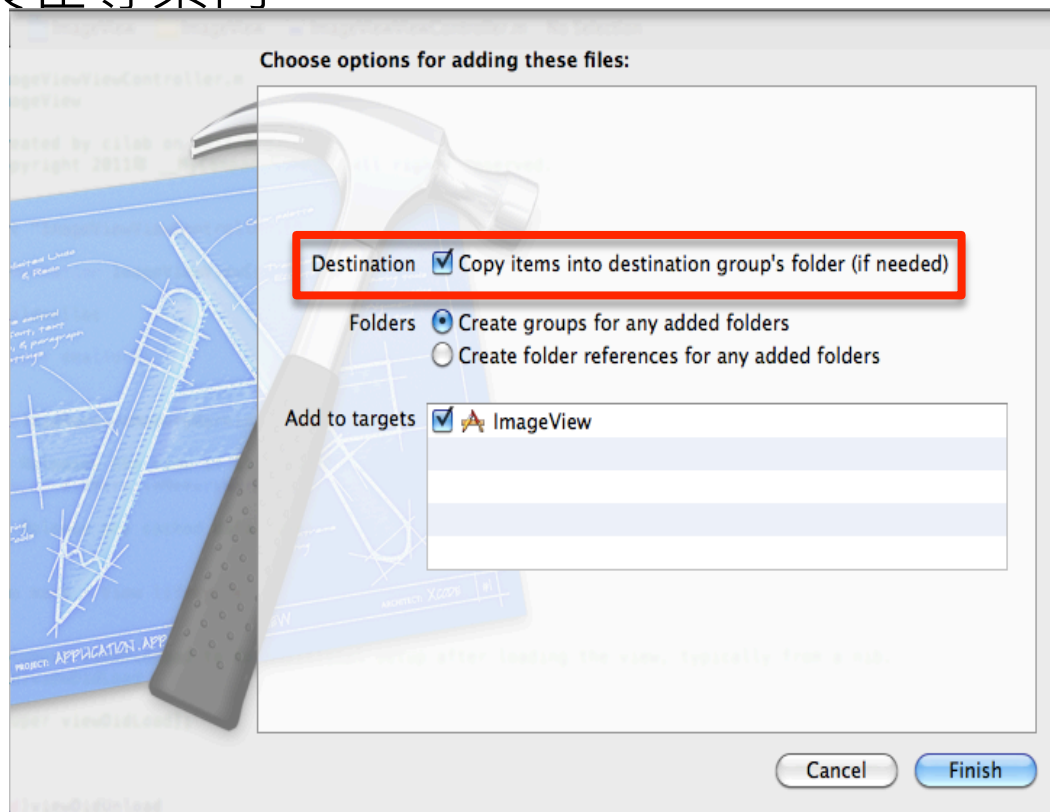
- ▶ 學習目的
 - 圖片如何放置
 - 加深使用UIButton元件的練習
 - 動畫運用

執行結果



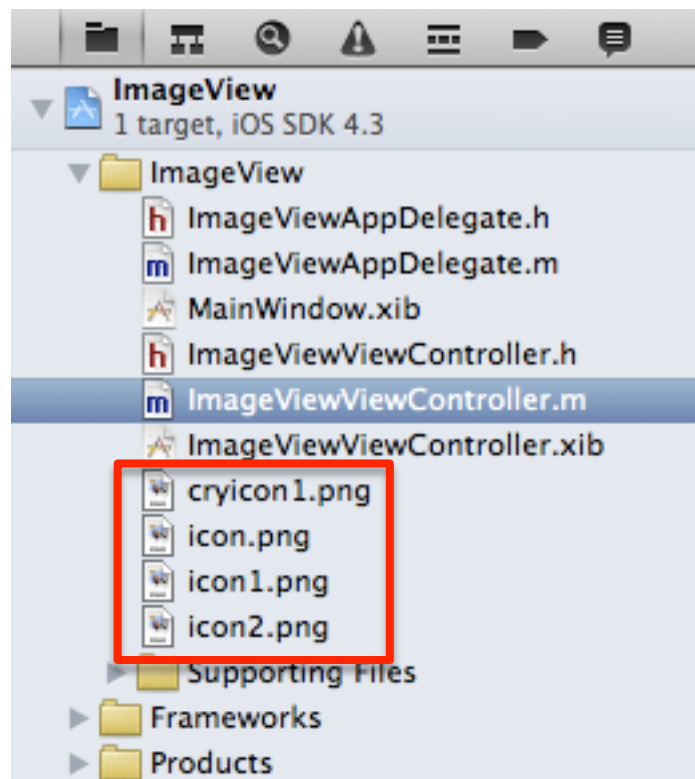
加入圖片

- ▶ 是否將檔案加入專案
 - 是否將原始檔一併搬進去專案內
 - 或者是將圖片直接複製在專案內



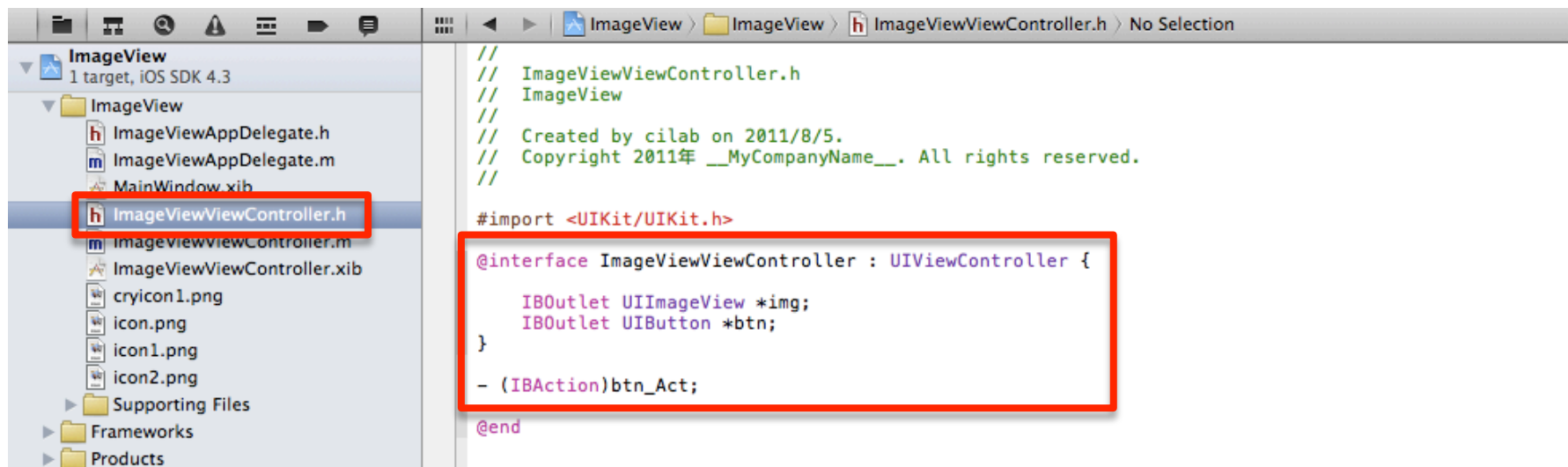
加入圖片

▶ 成功



定義介面參數(.h)

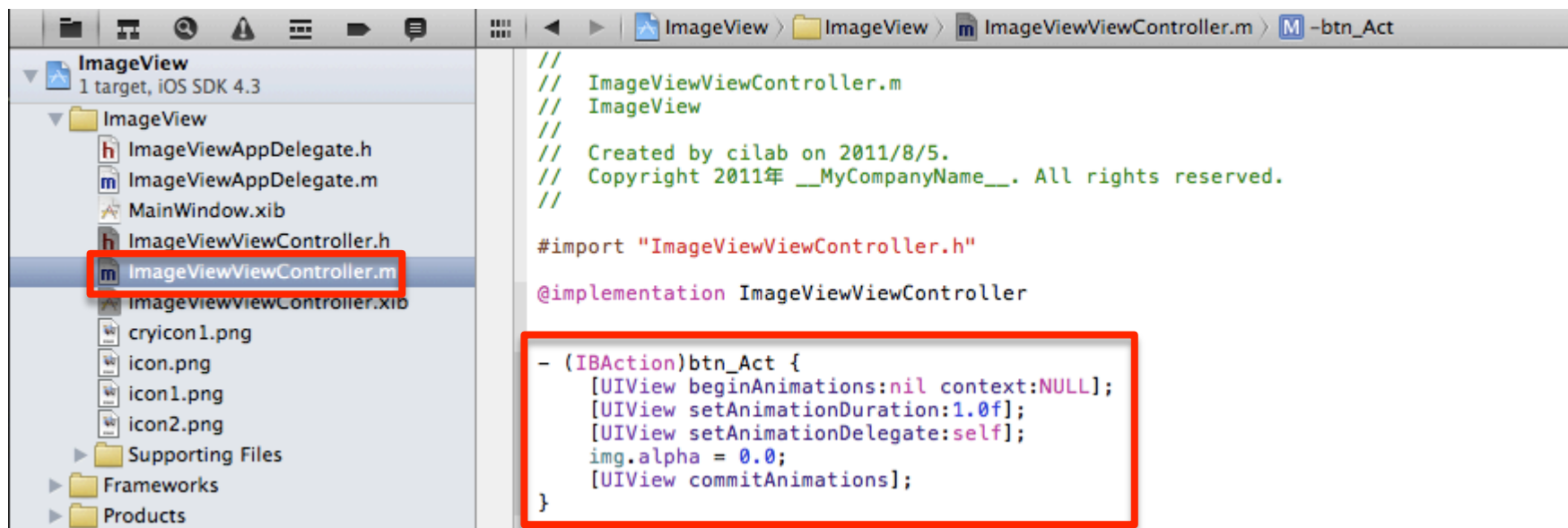
- ▶ 說明
 - 定義UIButton類別的變數btn
 - 定義UIImageView類別的變數img
 - 定義副函式btn_Act()



```
//  
// UIImageViewViewController.h  
// UIImageView  
//  
// Created by cilab on 2011/8/5.  
// Copyright 2011年 __MyCompanyName__. All rights reserved.  
//  
#import <UIKit/UIKit.h>  
  
@interface UIImageViewViewController : UIViewController {  
    IBOutlet UIImageView *img;  
    IBOutlet UIButton *btn;  
}  
  
- (IBAction)btn_Act;  
  
@end
```

實作(.m)

- ▶ 實作btn_Act函式
 - 加入動畫語法



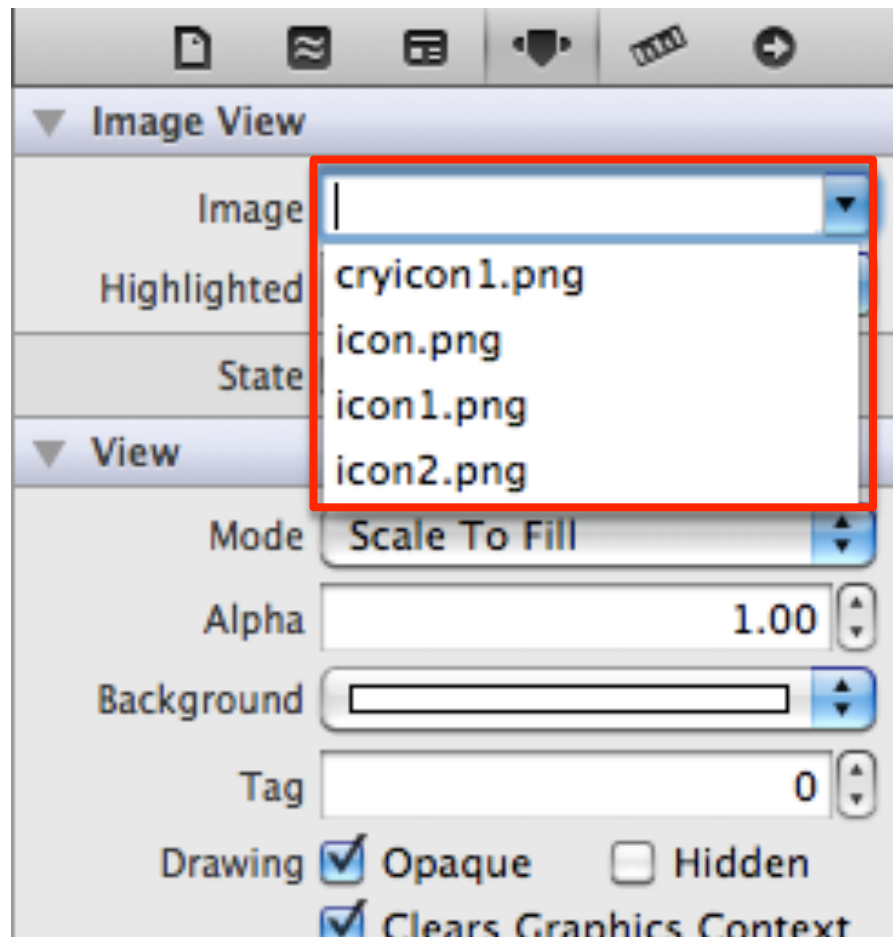
```
//  
//  ImageViewViewController.m  
//  ImageView  
//  
//  Created by cilab on 2011/8/5.  
//  Copyright 2011年 __MyCompanyName__. All rights reserved.  
//  
#import "ImageViewViewController.h"  
  
@implementation ImageViewViewController  
  
- (IBAction)btn_Act {  
    [UIView beginAnimations:nil context:NULL];  
    [UIView setAnimationDuration:1.0f];  
    [UIView setAnimationDelegate:self];  
    img.alpha = 0.0;  
    [UIView commitAnimations];  
}
```

製作操作介面

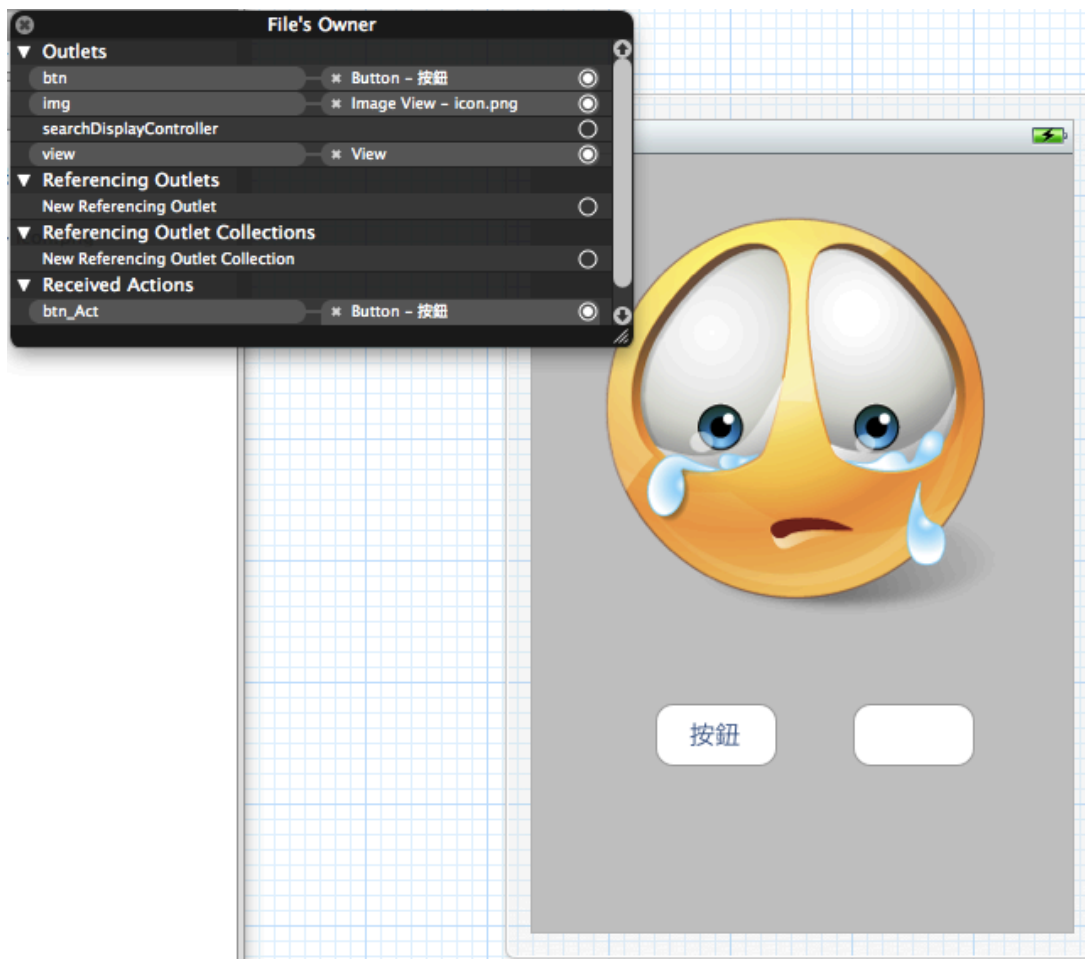


設定預設顯示圖片

- ▶ 右上方的IB屬性欄位
 - 設定ImageView預設圖片



連結程式定義與操作介面



備註

▶ 動畫相關語法

- 宣告開始 UIView 動畫效果

```
[UIView beginAnimations:nil context:nil];
```

- 設置動畫開始至結束的時間(秒)

```
[UIView setAnimationDuration:3.0f];
```

- 設置動畫延遲開始的時間(秒)

```
[UIView setAnimationDelay:1.0f];
```

- 設置動畫效果種類

```
[UIView setAnimationCurve:UIViewAnimationCurveEaseOut];
```

- 執行動畫效果

```
[UIView commitAnimations];
```

元件練習TextField

- ▶ 學習目的
 - 取得使用者輸入的資料
 - 取得使用者輸入狀態

TextField

The image shows a screenshot of the Xcode interface. At the top, the breadcrumb navigation shows the path: ViewController.xib > FirstViewController.xib (English) > View > Text Field. A blue box with the Chinese characters '屬性' (Properties) is positioned above the properties panel. The main storyboard area displays a window titled 'First View' containing a blue rounded rectangle representing a text field, which is highlighted with a red border. To the right, the 'Text Field' properties panel is visible, also outlined in red. It includes fields for Text, Placeholder, Background, Disabled, Alignment, Border Style, Clear Button, Text Color, Font, Min Font Size, Capitalization, Correction, Keyboard, Appearance, Return Key, and an 'Auto-enable Return Key' checkbox. Below the storyboard, a 'Text Field' info panel is open, providing a description of the widget. At the bottom right, the 'Objects' list shows 'Text Field' selected, with a red box highlighting its description: 'Text Field - Displays editable text and sends an action message to a target object when Return is tapped.' Other objects like 'Slider' and 'Switch' are also visible in the list.

屬性

Text Field

Text Text

Placeholder Placeholder Text

Background Background Image

Disabled Disabled Background Image

Alignment

Border Style

Clear Button Never appears

Clear when editing begins

Text Color Default

Font System 14.0

Min Font Size 17

Adjust to Fit

Capitalization None

Correction Default

Keyboard Default

Appearance Default

Return Key Default

Auto-enable Return Key

Text Text Field
UITextField

Displays a rounded rectangle that can contain editable text. When a user taps a text field, a keyboard appears; when a user taps Return in the keyboard, the keyboard disappears and the text field can handle the input in an application-specific way. UITextField supports overlay views to display additional information, such as a bookmarks icon. UITextField also provides a clear text control a user taps to erase the contents of the text field.

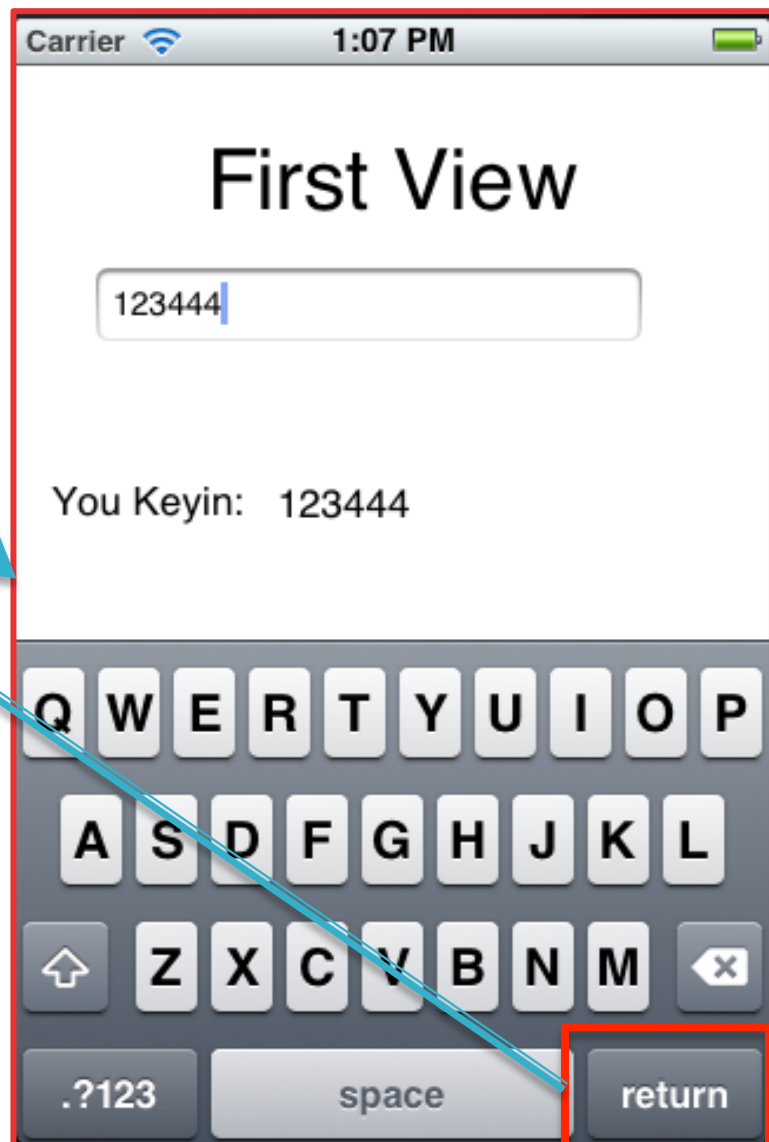
Done

Text 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 toggle the value.

執行結果



定義介面參數(.h)

▶ 說明

- 定義UITextField類別的變數textField
- 定義UILabel類別的變數labelKeyin
- 設定繼承UITextFieldDelegate屬性

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

@interface FirstViewController : UIViewController <UITextFieldDelegate>
{
    IBOutlet UITextField *m_TextField;
    IBOutlet UILabel *labelKeyin;
}

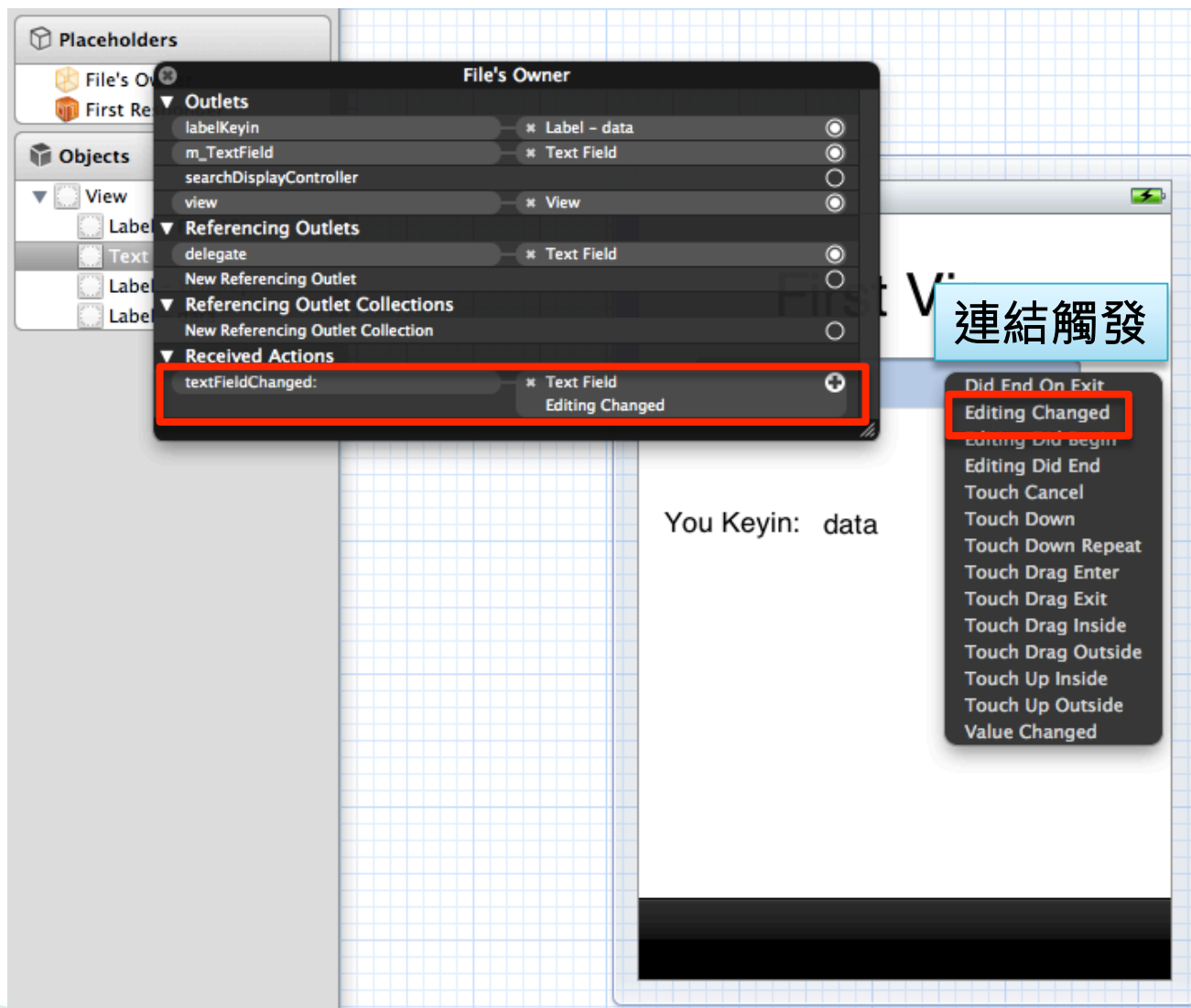
@end
```

實作(.m)

```
- (IBAction)textFieldChanged:(UITextField *)textField {
    labelKeyin.text = textField.text;
}

#pragma mark - UITextFieldDelegate
- (BOOL)textFieldShouldReturn:(UITextField *)textField {
    [textField resignFirstResponder];
    return YES;
}
```


製作操作介面



備註

▶ 常用函示

- 關閉輸入鍵盤: [textField resignFirstResponder];
- 開啓輸入鍵牌: [textField becomeFirstResponder];

▶ UITextFieldDelegate函式列表

- return NO to disallow editing
 - – (BOOL)textFieldShouldBeginEditing:(UITextField *)textField;
- became first responder
 - – (void)textFieldDidBeginEditing:(UITextField *)textField;
- return YES to allow editing to stop and to resign first responder status. NO to disallow the editing session to end
 - – (BOOL)textFieldShouldEndEditing:(UITextField *)textField;
- may be called if forced even if shouldEndEditing returns NO (e.g. view removed from window) or endEditing:YES called
 - – (void)textFieldDidEndEditing:(UITextField *)textField;
- – (BOOL)textFieldShouldClear:(UITextField *)textField;
- – (BOOL)textFieldShouldReturn:(UITextField *)textField;

元件練習TextView

- ▶ 學習目的
 - 取得使用者輸入的資料

TextView

The screenshot shows the Xcode interface with a storyboard on the left and the Properties Inspector on the right. The storyboard displays a window titled "Second View" containing a text field with the text "keyin", an "OK" button, and a label "data". A red box highlights the text field. A tooltip for "Text View" is visible, providing a description of UITextView. The Properties Inspector on the right shows the "Text View" section with various settings such as "Text" (keyin), "Behavior" (Editable), "Alignment", "Text Color", "Font" (System 14.0), and "Scroll View" options. A red box highlights the entire Properties Inspector area.

屬性

Second View

keyin

OK

data

Text View
UITextView

UITextView displays a region that can contain multiple lines of editable text. When a user taps a text view, a keyboard appears; when a user taps Return in the keyboard, the keyboard disappears and the text view can handle the input in an application-specific way. You can specify attributes, such as font, color, and alignment, that apply to all text in a text view.

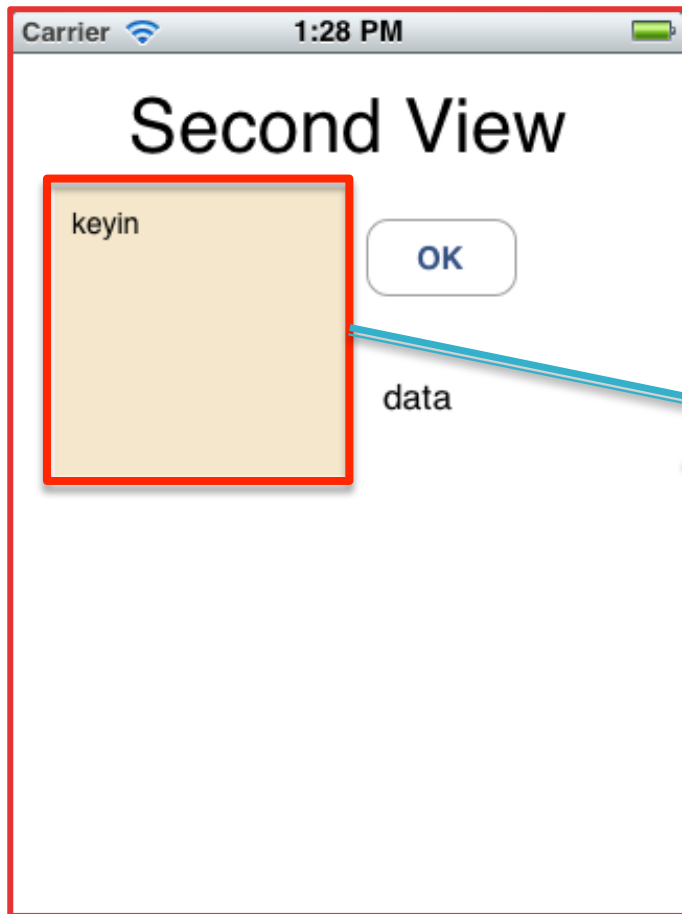
Done

Text View – Displays multiple lines of editable text and sends an action message to a target object when...

Web View – Displays embedded web content and enables content navigation.

Map View – Displays maps and provides an embeddable interface to navigate map content.

執行結果



定義介面參數(.h)

▶ 說明

- 定義UITextView類別的變數m_TextView
- 定義UILabel類別的變數labelKeyin
- 定義UIButton類別的變數btn_OK
- 設定繼承UITextViewDelegate屬性

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

@interface SecondViewController : UIViewController <UITextViewDelegate>
{
    IBOutlet UITextView *m_TextView;
    IBOutlet UILabel *labelKeyin;
    IBOutlet UIButton *btn_OK;
}

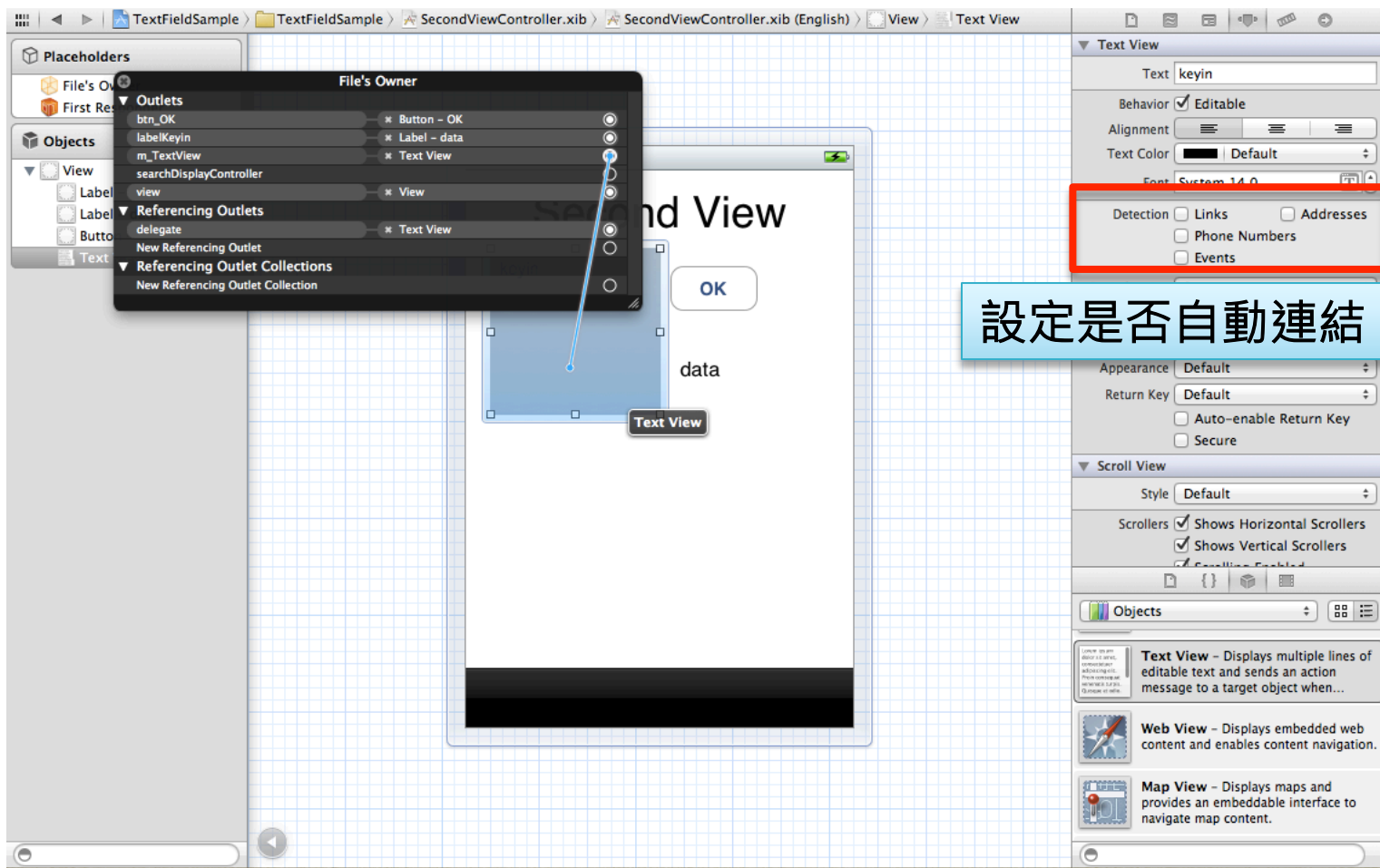
@end
```

實作(.m)

```
- (IBAction)btnOKAction:(id)sender {
    [m_TextView resignFirstResponder];
}

#pragma mark - UITextViewDelegate
- (void)textViewDidChange:(UITextView *)textView {
    labelKeyin.text = m_TextView.text;
}
```

製作操作介面



備註

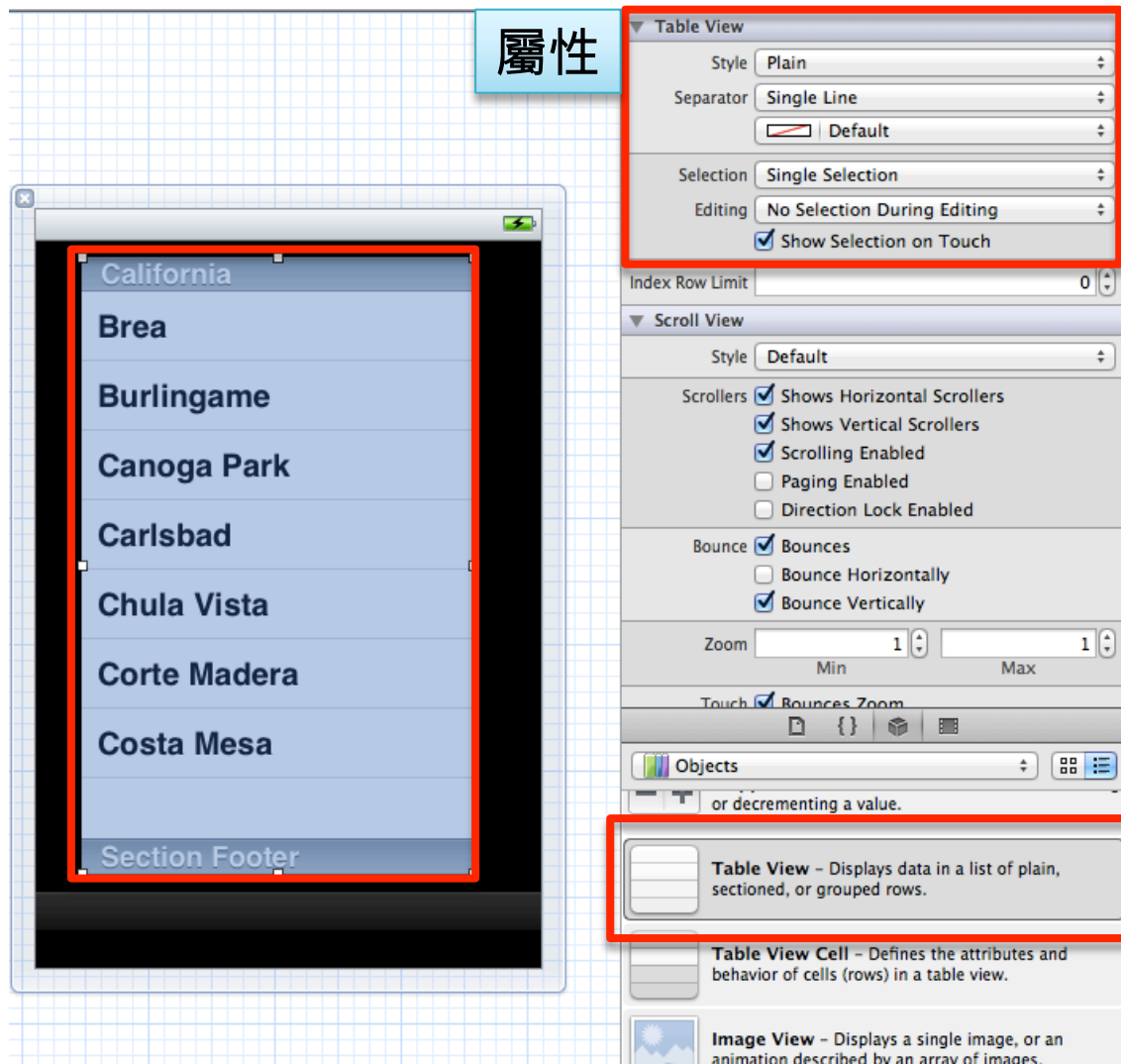
- ▶ UITextViewDelegate 函式列表
 - – (BOOL)textViewShouldBeginEditing:(UITextView *)textView;
 - – (BOOL)textViewShouldEndEditing:(UITextView *)textView;
 - – (void)textViewDidBeginEditing:(UITextView *)textView;
 - – (void)textViewDidEndEditing:(UITextView *)textView;
 - – (BOOL)textView:(UITextView *)textView shouldChangeTextInRange:(NSRange)range replacementText:(NSString *)text;
 - – (void)textViewDidChange:(UITextView *)textView;
 - – (void)textViewDidChangeSelection:(UITextView *)textView;

元件練習TableView

- ▶ 學習目的
 - 列表元件TableView顯示資料
 - TableView特性

TableView

屬性



The image shows a screenshot of the Xcode interface. On the left, a UITableView is displayed with a list of California cities: California, Brea, Burlingame, Canoga Park, Carlsbad, Chula Vista, Corte Madera, and Costa Mesa. A red box highlights the entire table view. On the right, the Attributes Inspector is open, showing the 'Table View' settings. A red box highlights the 'Table View' section, which includes the following settings: Style: Plain, Separator: Single Line, Selection: Single Selection, Editing: No Selection During Editing, and Show Selection on Touch: checked. Below the 'Table View' section, the 'Scroll View' settings are visible, including Style: Default, Scrollers: Shows Horizontal Scrollers, Shows Vertical Scrollers, Scrolling Enabled, Paging Enabled, and Direction Lock Enabled. The 'Zoom' settings are also visible, with Min and Max both set to 1. At the bottom of the Attributes Inspector, a red box highlights the 'Table View' description: 'Table View - Displays data in a list of plain, sectioned, or grouped rows.'

Table View

- Style: Plain
- Separator: Single Line
- Selection: Single Selection
- Editing: No Selection During Editing
- Show Selection on Touch:

Index Row Limit: 0

Scroll View

- Style: Default
- Scrollers: Shows Horizontal Scrollers, Shows Vertical Scrollers, Scrolling Enabled, Paging Enabled, Direction Lock Enabled
- Bounce: Bounces, Bounce Horizontally, Bounce Vertically
- Zoom: Min 1, Max 1
- Touch: Bounces Zoom

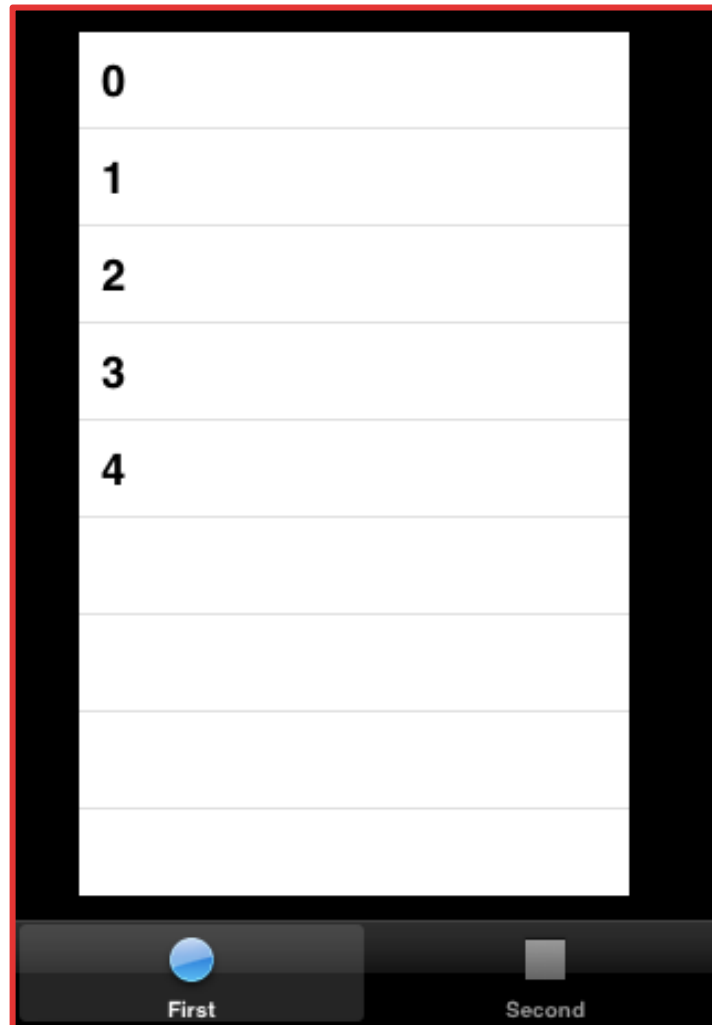
Objects

Table View - Displays data in a list of plain, sectioned, or grouped rows.

Table View Cell - Defines the attributes and behavior of cells (rows) in a table view.

Image View - Displays a single image, or an animation described by an array of images.

執行結果



定義介面參數(.h)

▶ 說明

- 定義UITableView類別的變數tableView
- 設定繼承UITableViewDelegate屬性
- 設定繼承UITableViewDataSource屬性

```
10
11 @interface FirstViewController : UIViewController <UITableViewDelegate, UITableViewDataSource>
12 {
13     IBOutlet UITableView *m_tableView;
14 }
15 @end
16
```

實作(.m)

```
44 #pragma mark - UITableViewDataSource Methods
45 - (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView {
46     return 1;
47 }
48
49 #if 0
50 - (CGFloat)tableView:(UITableView *)tableView heightForRowAtIndexPath:(NSIndexPath *)indexPath {
51     return 44.0f;
52 }
53 #endif
54
55 - (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section {
56     return 5;
57 }
58
59 - (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath {
60
61     NSString *CellIdentifier = @"UITableViewCell";
62     UITableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:CellIdentifier];
63     if (cell == nil) {
64         cell = [[UITableViewCell alloc] initWithStyle:UITableViewCellStyleDefault reuseIdentifier:
65             CellIdentifier];
66     }
67     cell.textLabel.text = [NSString stringWithFormat:@"%d", indexPath.row];
68     return cell;
69 }
```

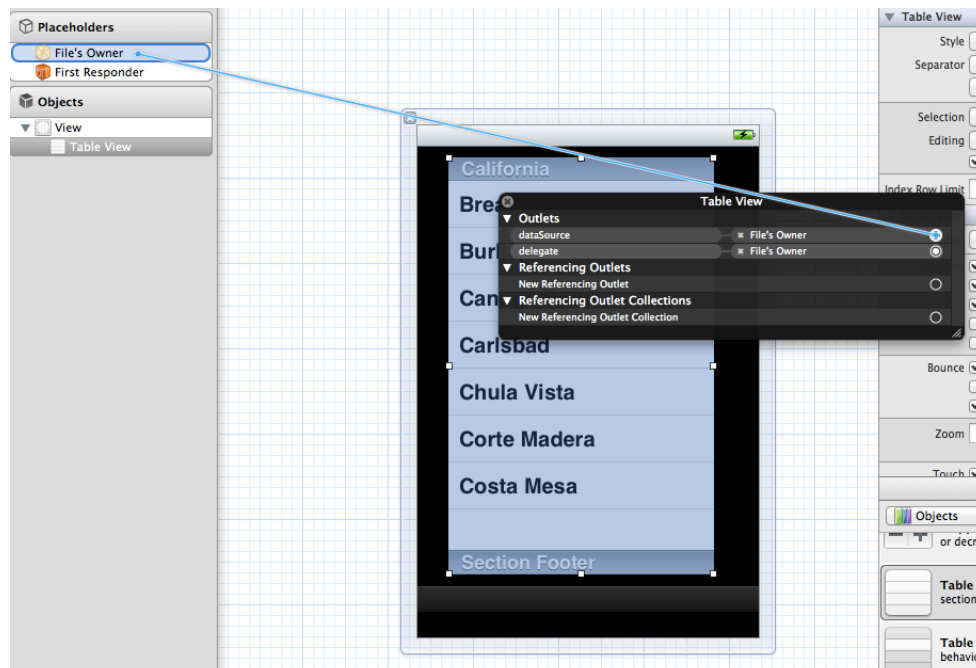
實作(.m)

- ▶ 點選項目後，消除選擇
 - [tableView deselectRowAtIndexPath:indexPath animated:YES];

```
71 #pragma mark - tableViewDelegate Methods
72 - (void)tableView:(UITableView *)tableView didSelectRowAtIndexPath:(NSIndexPath *)indexPath {
73     [tableView deselectRowAtIndexPath:indexPath animated:YES];
74 }
75 }
```

製作操作介面

▶ 連結Delegate及DataSource



備註

▶ UITableViewDataSource常用函式

◦ 必要

- – (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section;
- – (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath;

◦ 非必要

- – (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView;
- – (NSString *)tableView:(UITableView *)tableView titleForHeaderInSection:(NSInteger)section;
- – (BOOL)tableView:(UITableView *)tableView canEditRowAtIndexPath:(NSIndexPath *)indexPath;
- – (BOOL)tableView:(UITableView *)tableView canMoveRowAtIndexPath:(NSIndexPath *)indexPath;
- – (void)tableView:(UITableView *)tableView moveRowAtIndexPath:(NSIndexPath *)sourceIndexPath toIndexPath:(NSIndexPath *)destinationIndexPath;

備註

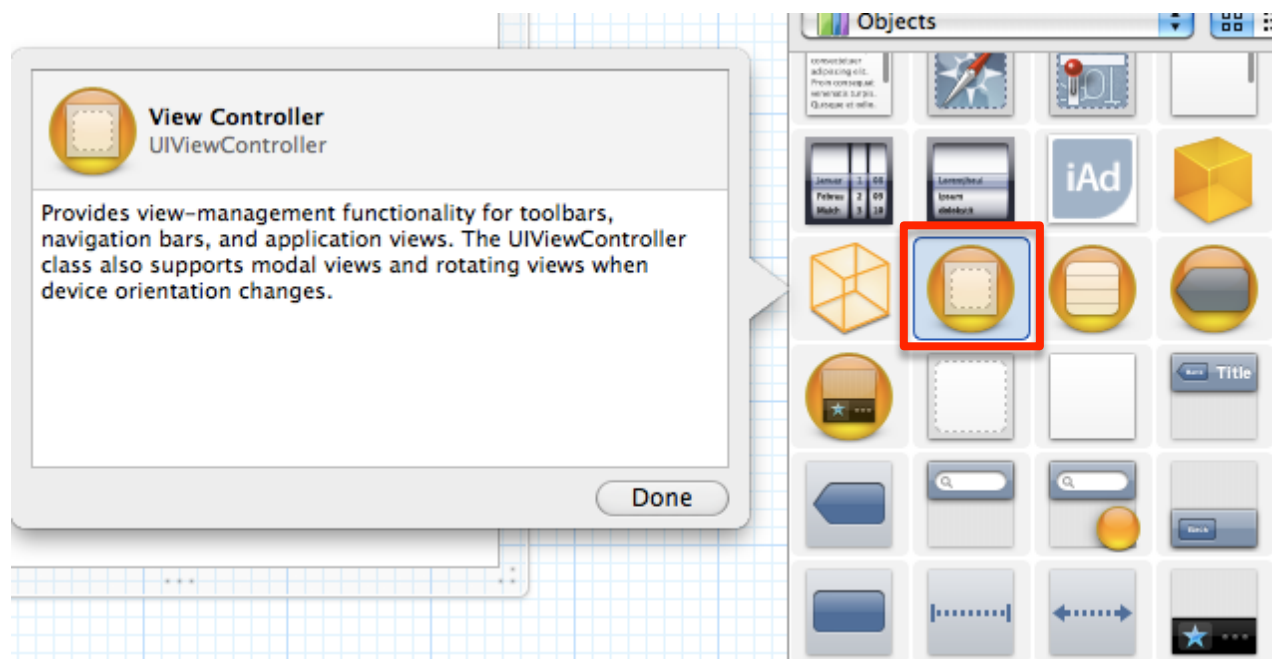
- ▶ 更新資料
 - – (void)reloadData;
- ▶ UITableViewDelegate常用函式
 - 非必要
 - – (CGFloat)tableView:(UITableView *)tableView heightForRowAtIndexPath:(NSIndexPath *)indexPath;
 - – (void)tableView:(UITableView *)tableView didSelectRowAtIndexPath:(NSIndexPath *)indexPath;

切換頁面練習

- ▶ 學習目的
 - 運用各種的換頁技巧
 - 使用控制項

UIViewController

- ▶ 控制項物件
 - 能與UIViewController類別的Class做連結



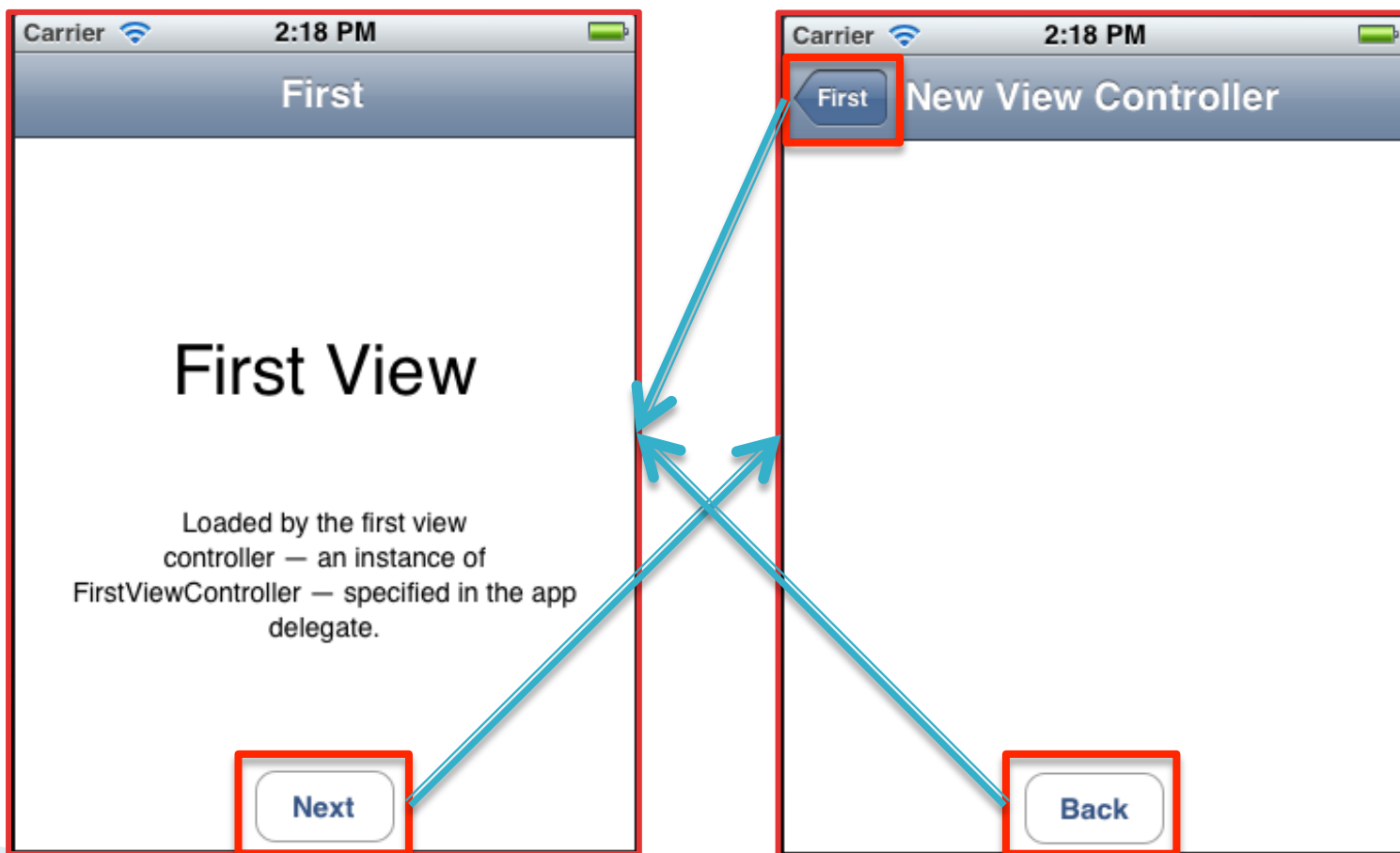
UIView

- ▶ 擺放元件的顯示器



切換頁面練習(導覽列)

▶ 執行結果



修改AppDelegate.m

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    self.window = [[[UIWindow alloc] initWithFrame:[[UIScreen mainScreen] bounds]] autorelease];
    // Override point for customization after application launch.
    UIViewController *viewController1 = [[[FirstViewController alloc] initWithNibName:@"FirstViewController" bundle:nil]
    autorelease];
    UINavigationController *navCtrl1 = [[[UINavigationController alloc] initWithRootViewController:viewController1]
    autorelease];
    UIViewController *viewController2 = [[[SecondViewController alloc] initWithNibName:@"SecondViewController" bundle:
    nil] autorelease];
    self.tabBarController = [[[UITabBarController alloc] init] autorelease];
    self.tabBarController.viewControllers = [NSArray arrayWithObjects:navCtrl1, viewController2, nil];
    self.window.rootViewController = self.tabBarController;
    [self.window makeKeyAndVisible];
    return YES;
}
```

定義介面參數(.h)

- ▶ 說明
 - 定義UIViewController的變數viewCtrl

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

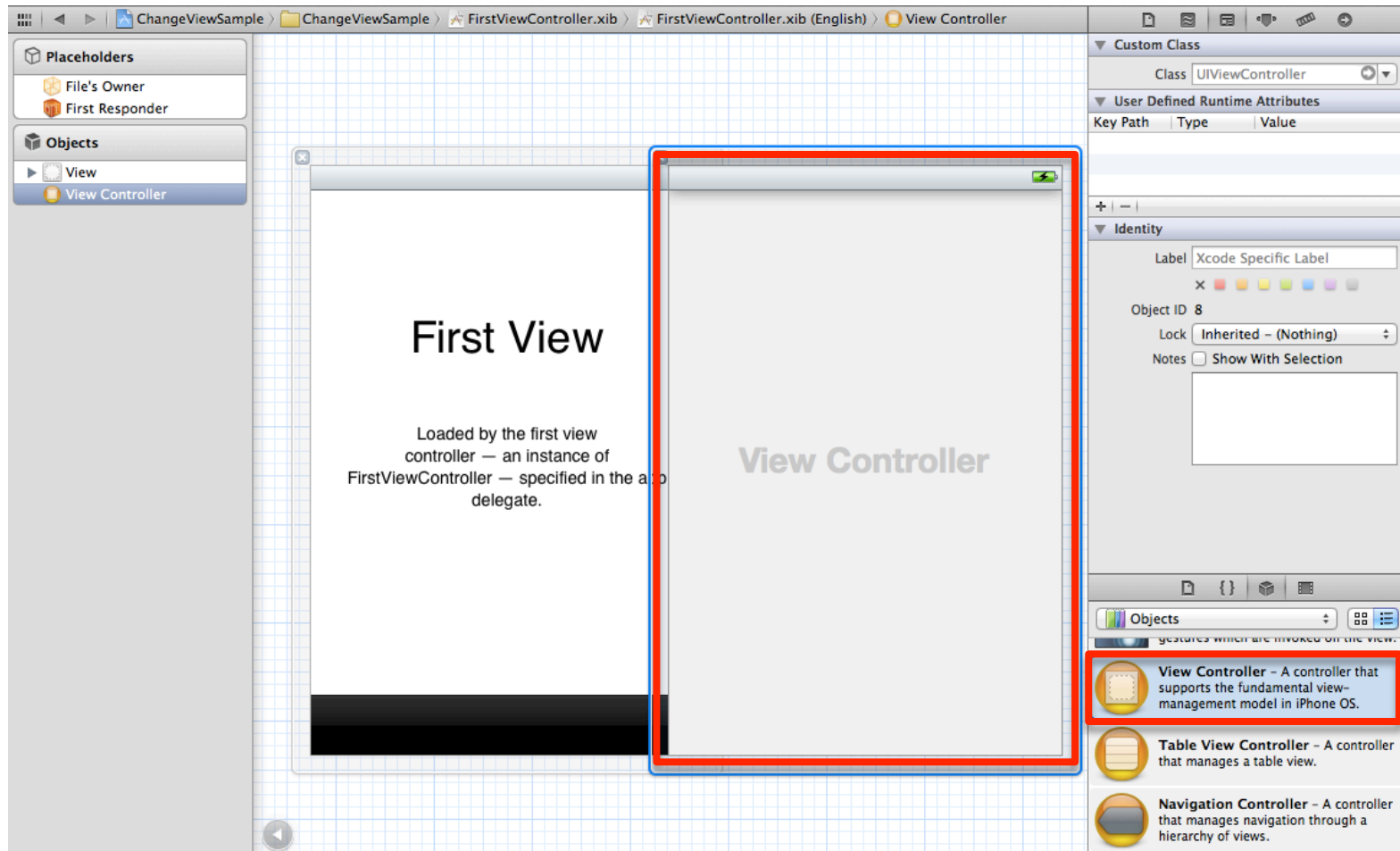
@interface FirstViewController : UIViewController
{
    IBOutlet UIViewController *viewCtrl;
}
@end
```

實作(.m)

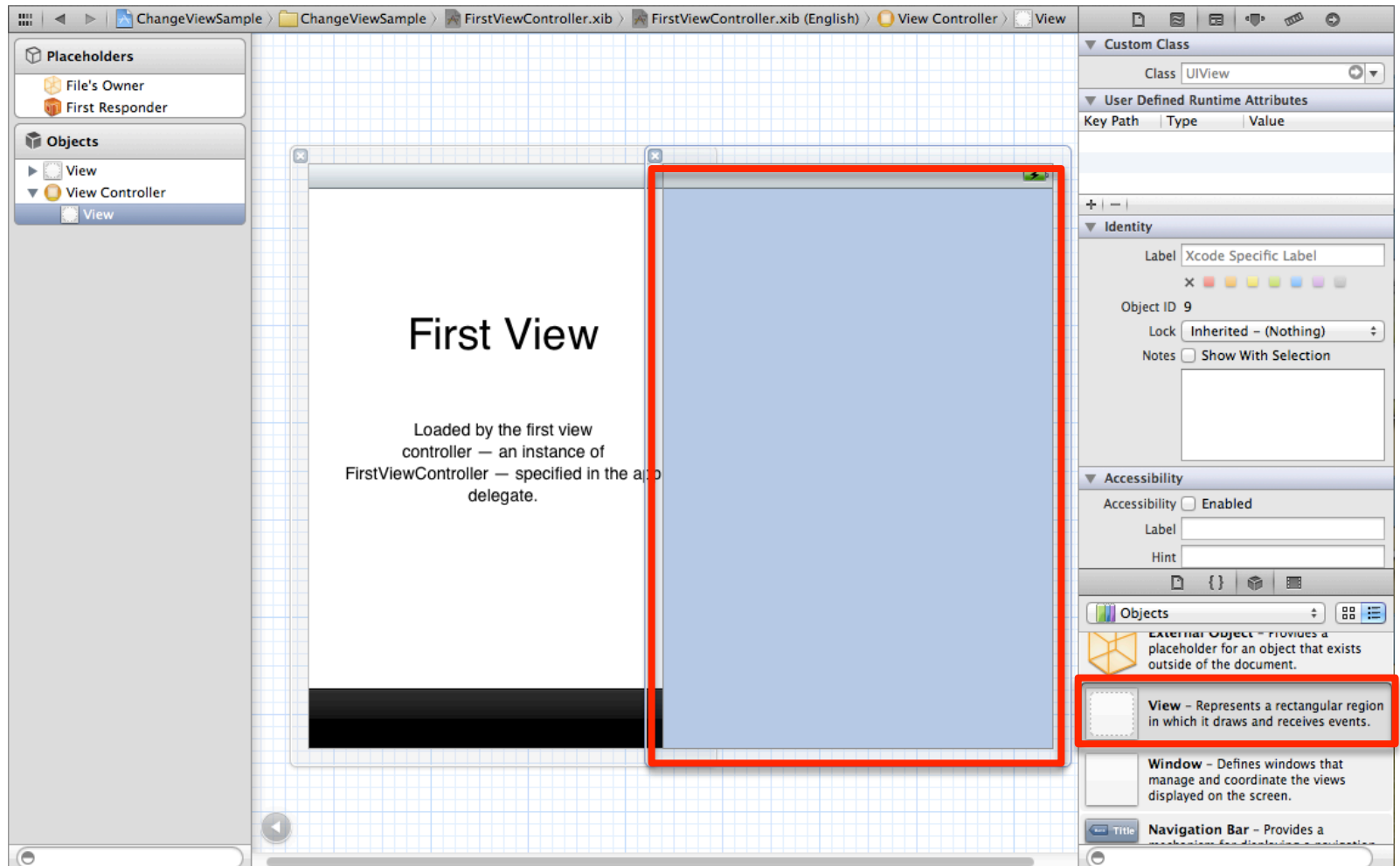
```
- (IBAction)btnNextAction:(id)sender {
    [self.navigationController pushViewController:viewCtrl animated:YES];
    viewCtrl.title = @"New View Controller";
}

- (IBAction)btnBackAction:(id)sender {
    [self.navigationController popViewControllerAnimated:YES];
}
```

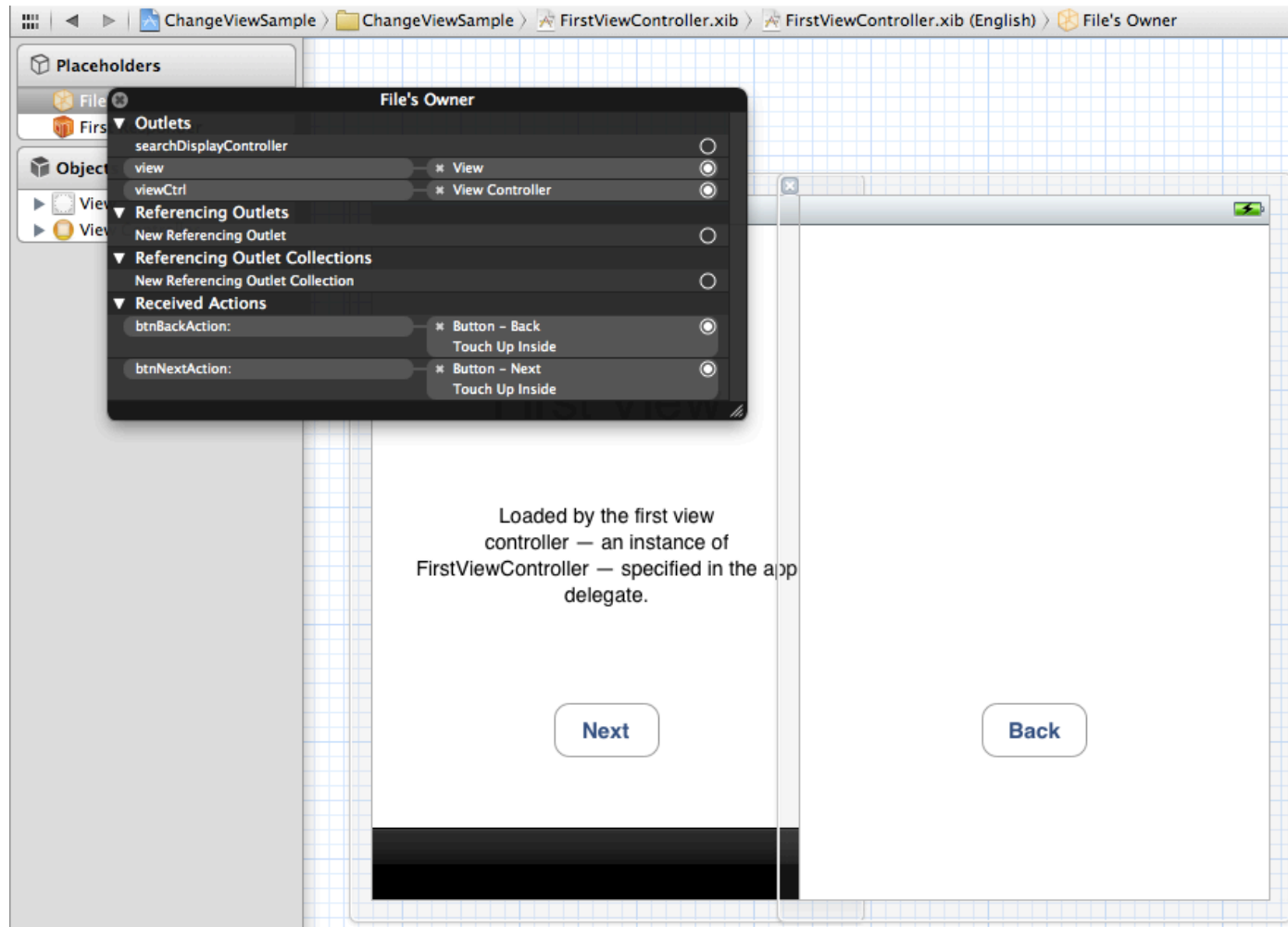
製作操作介面-加入UIViewController



製作操作介面-加入View



製作操作介面



備註

▶ UINavigationController常用的指令

- 修改Bar上的標題

```
self.title = @"內容";
```

- 把畫面推出

```
[self.navigationController  
  pushViewController:viewController animated:YES];
```

- 讓navigationBar隱藏

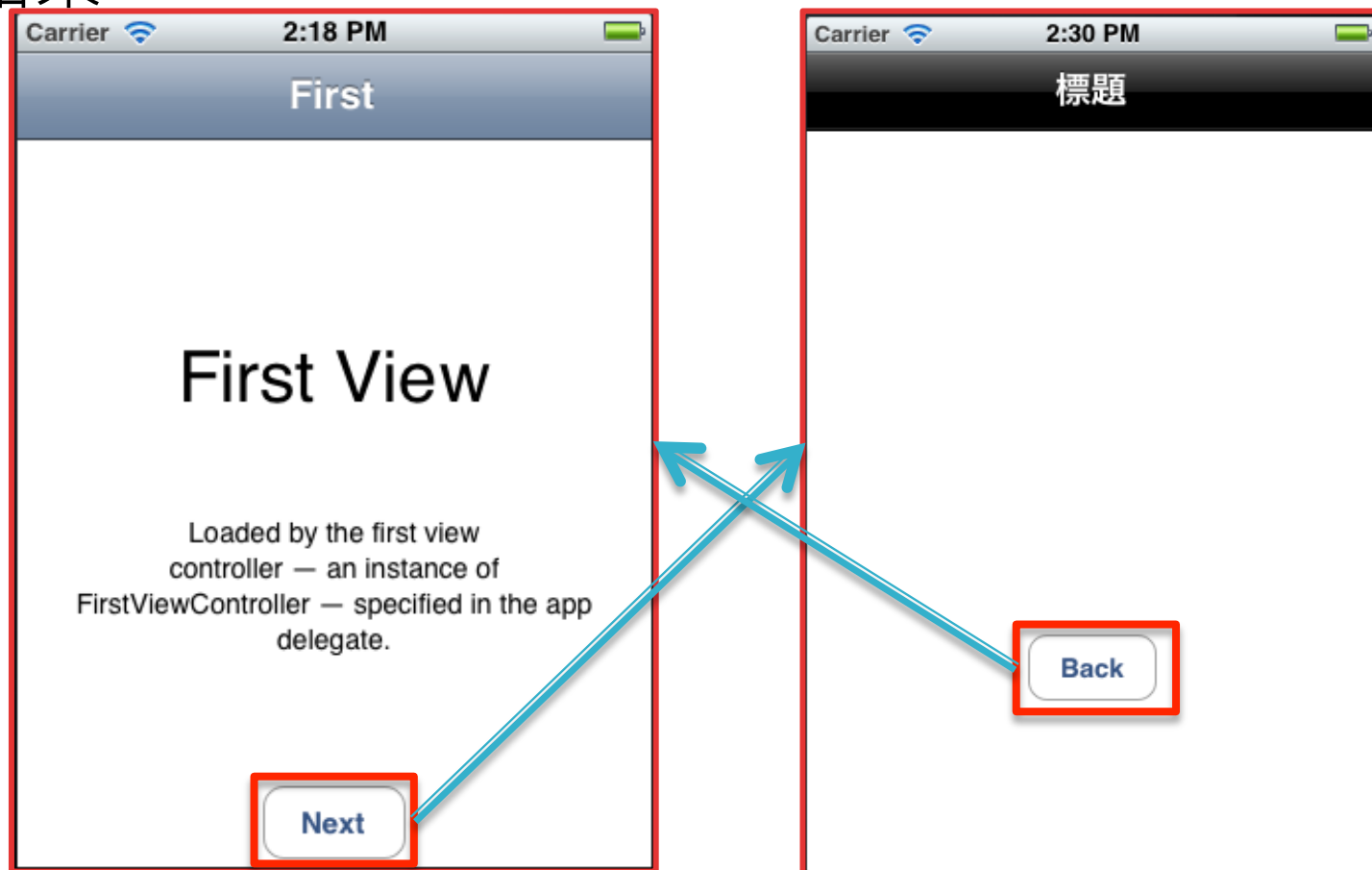
```
self.navigationController.navigationBarHidden = YES;
```

- 讓navigationBar顯示

```
self.navigationController.navigationBarHidden = NO;
```

切換頁面練習(由下推出)

▶ 執行結果



實作(.m)

```
- (IBAction)btnNextAction:(id)sender {  
    // [self.navigationController pushViewController:viewCtrl animated:YES];  
    // [viewCtrl.title = @"New View Controller"];  
    [self presentViewController:viewCtrl animated:YES];  
}  
  
- (IBAction)btnBackAction:(id)sender {  
    // [self.navigationController popViewControllerAnimated:YES];  
    [viewCtrl dismissViewControllerAnimated:YES];  
}
```

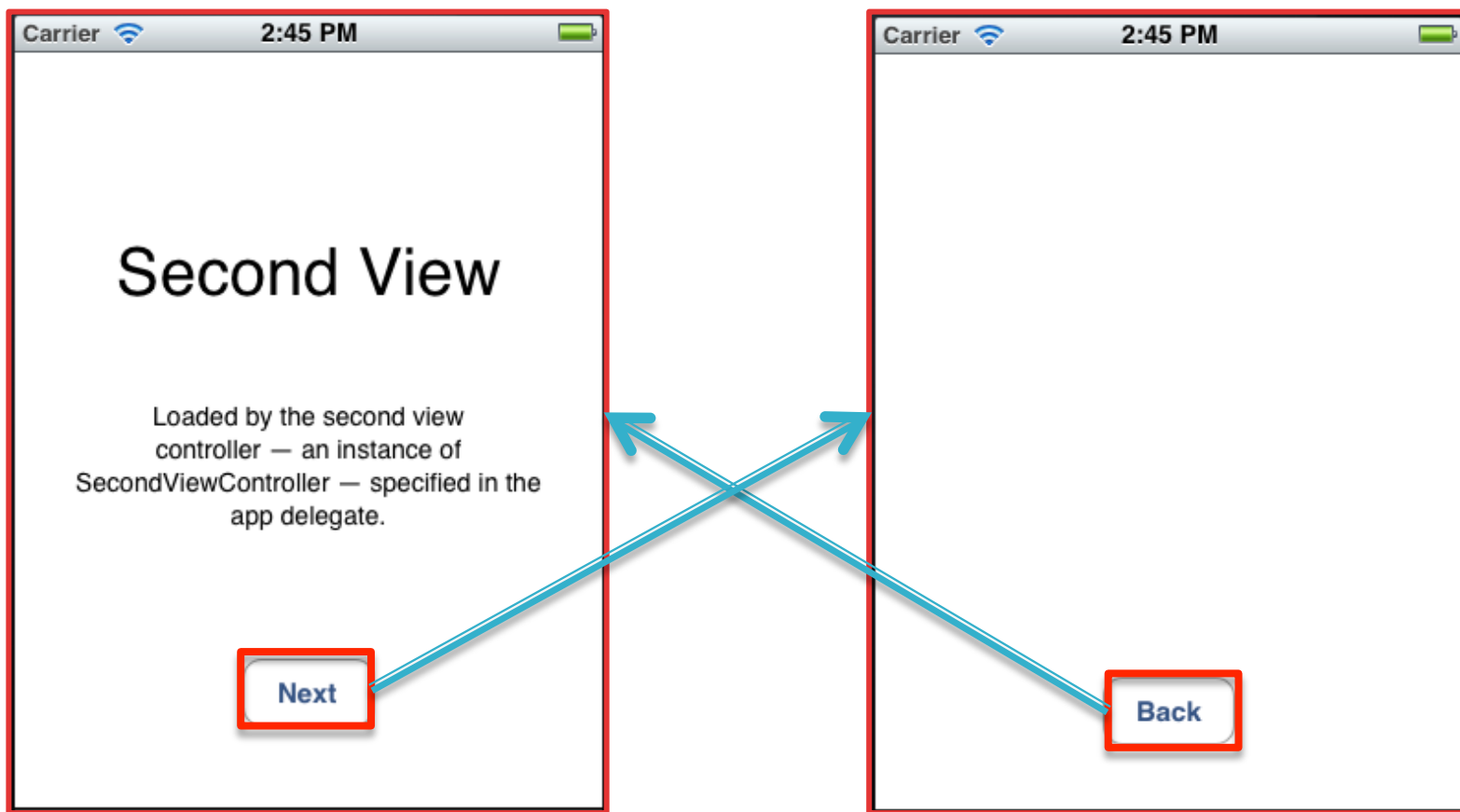
製作操作介面

▶ 加入標題

The screenshot shows the Xcode interface for editing a Navigation Bar. The main canvas displays a dark blue navigation bar with the title '標題' (Title) in white text. A red box highlights the navigation bar. To the right, the 'Attributes Inspector' (屬性) is open, showing the 'Navigation Bar' section with 'Style' set to 'Black Opaque' and 'TintColor' set to 'Default'. Below the canvas, a 'Library' panel shows the 'Navigation Bar' component selected, with a red box around its description: 'Provides a mechanism for displaying a navigation bar just below the status bar. To support navigation of hierarchical content, UINavigationController uses a stack to manage instances of UINavigationControllerItem, each of which represents a state of the navigation bar. By default, UINavigationController displays a back button on the left and a title in the center, but you can specify custom views for these, in addition to providing an optional button on the right of the navigation bar. Note that if you use a UINavigationController object to manage hierarchical navigation, you should not directly access the navigation bar'. The 'Done' button is visible at the bottom of the Library panel.

切換頁面練習(自訂動畫滑出)

▶ 執行結果



定義介面參數(.h)

- ▶ 說明
 - 定義UIView的變數view2

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

@interface SecondViewController : UIViewController
{
    IBOutlet UIView *view2;
}
@end
```

實作(.m)

```
- (IBAction)btnNextAction:(id)sender {
    [self.view addSubview:view2];
    view2.frame = CGRectMake(self.view.center.x, self.view.center.y, 0, 0);

    [UIView beginAnimations:nil context:nil];
    [UIView setAnimationDuration:1.0f];
    [UIView setAnimationBeginsFromCurrentState:YES];
    view2.frame = self.view.frame;
    [UIView commitAnimations];
}

- (IBAction)btnBackAction:(id)sender {
    [UIView animateWithDuration:1.0f animations:^(void) {
        view2.frame = CGRectMake(self.view.center.x, self.view.center.y, 0, 0);
    } completion:^(BOOL finished) {
        [view2 removeFromSuperview];
    }];
}
```

製作操作介面-加入View

The screenshot displays the Xcode interface for editing a storyboard. The main canvas shows two views: "Second View" on the left and "View2" on the right. "View2" is highlighted with a red border. The "Attributes Inspector" on the right is also highlighted with a red border and has a blue label "屬性" (Attributes) next to it. The "Objects" list on the left shows "View2" selected. The "Simulated Metrics" panel at the top right shows settings for Size, Orientation, Status Bar, etc. The "View" section of the Attributes Inspector shows settings for Mode, Tag, Interaction, Alpha, Background, Drawing, and Stretching. The "Objects" list at the bottom shows "View" and "Window" definitions.

Placeholders

- File's Owner
- First Responder

Objects

- View
- View2

Second View

Loaded by the second view controller — an instance of SecondViewController — specified in the app delegate.

Next

Back

Simulated Metrics

- Size: None
- Orientation: Portrait
- Status Bar: None

屬性

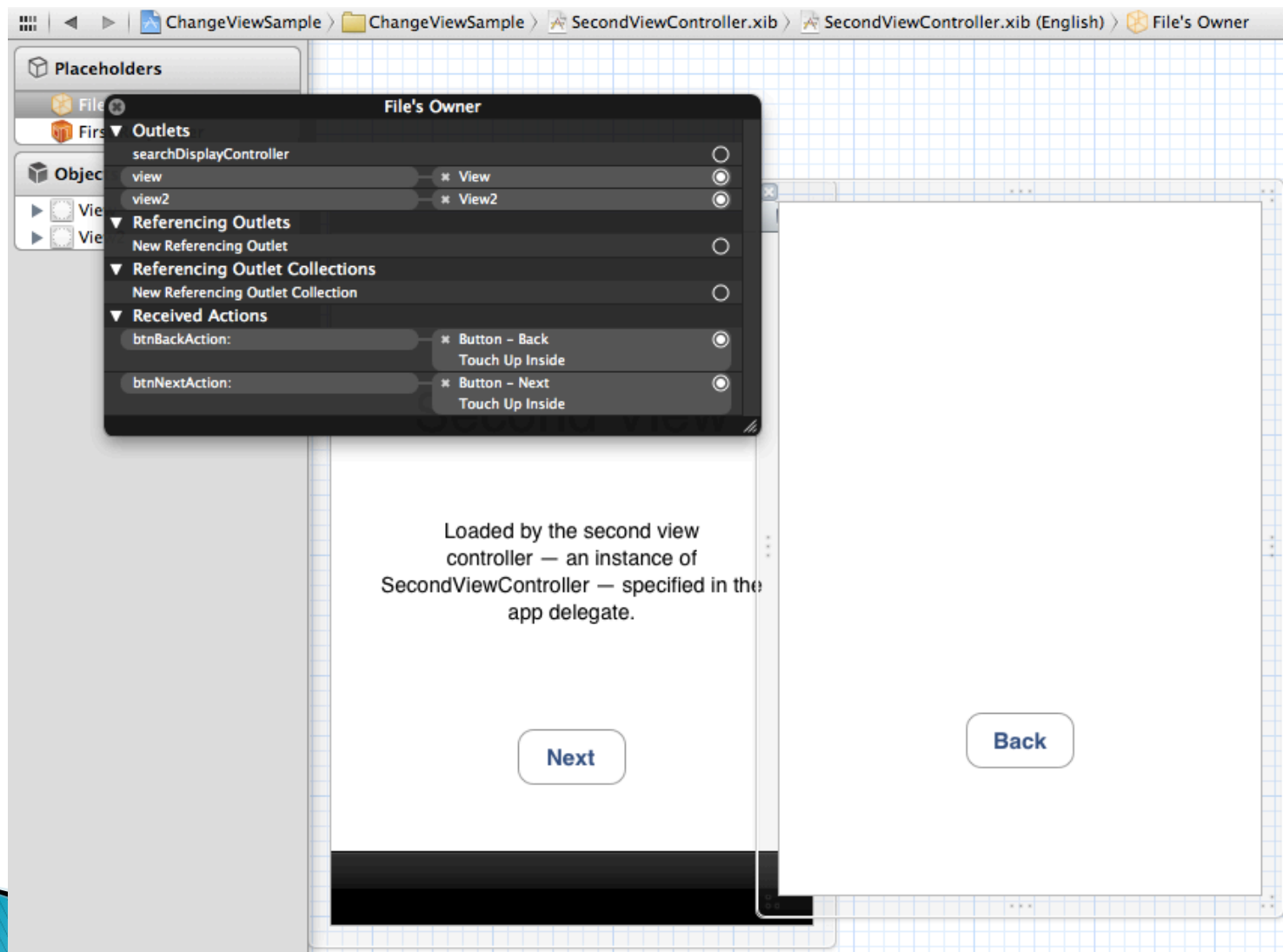
View

- Mode: Scale To Fill
- Tag: 0
- Interaction: User Interaction Enabled, Multiple Touch
- Alpha: 1
- Background: White Color
- Drawing: Opaque, Hidden, Clears Graphics Context, Clip Subviews, Autocomplete Subviews
- Stretching: X: 0, Y: 0, Width: 1, Height: 1

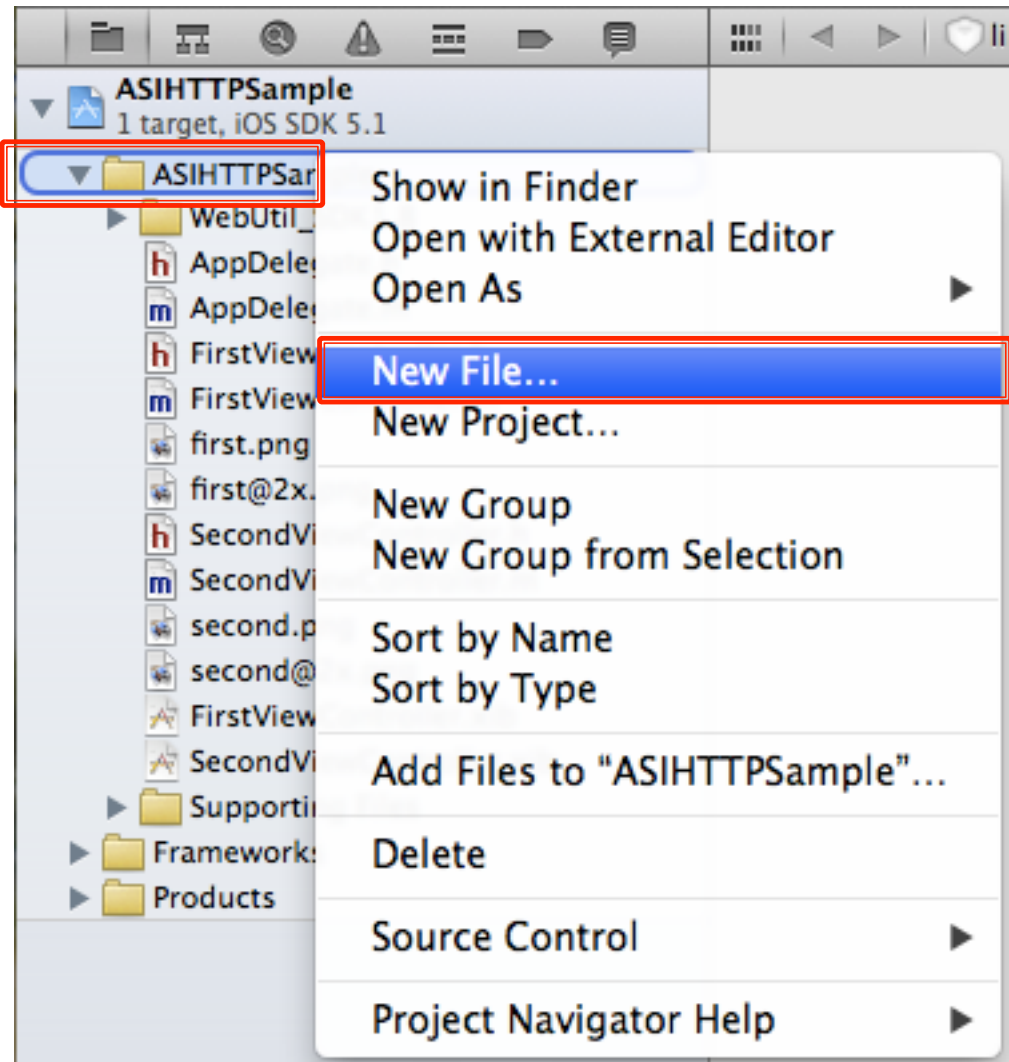
Objects

- outside of the document.
- View - Represents a rectangular region in which it draws and receives events.
- Window - Defines windows that manage and coordinate the views displayed on the screen.

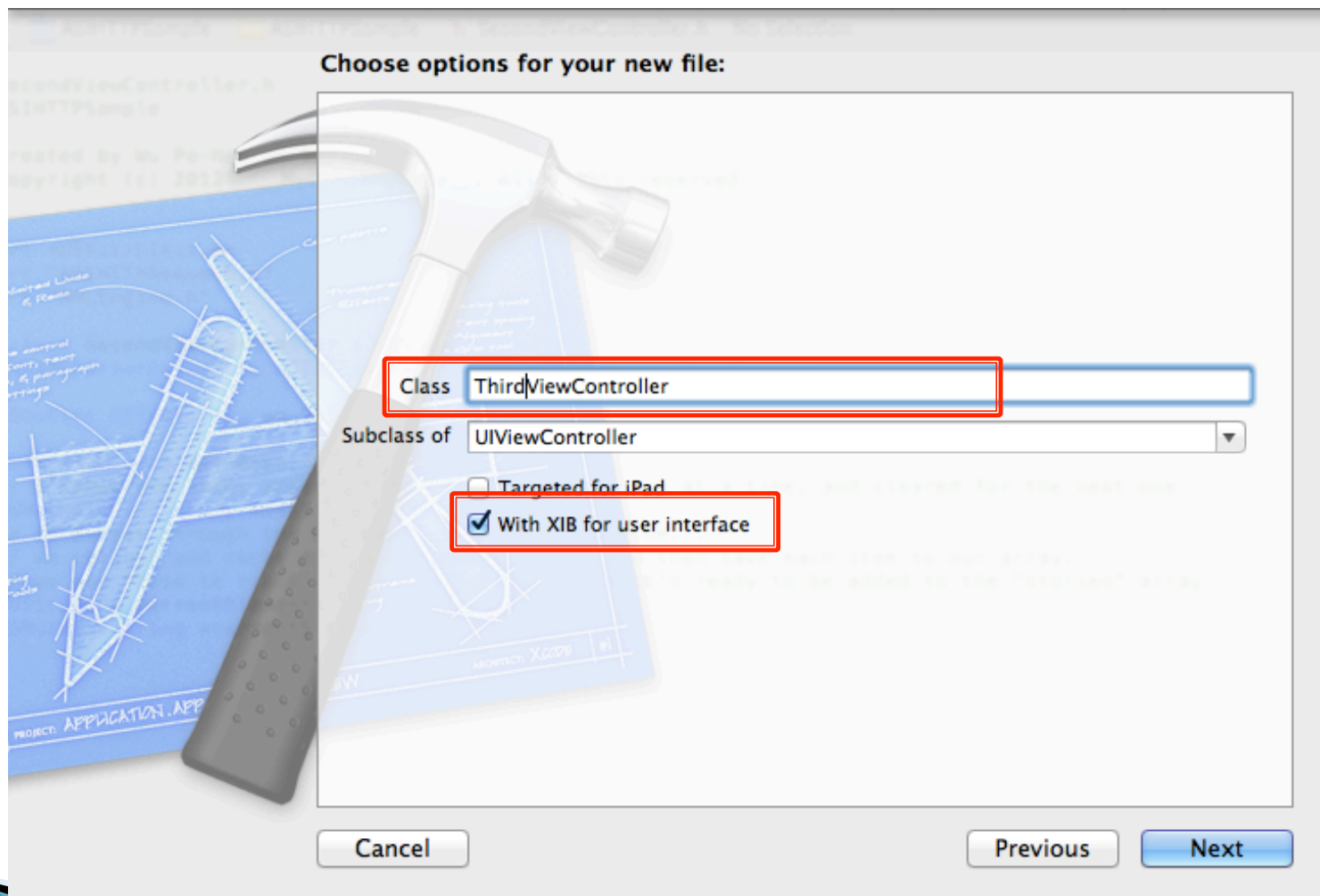
製作操作介面



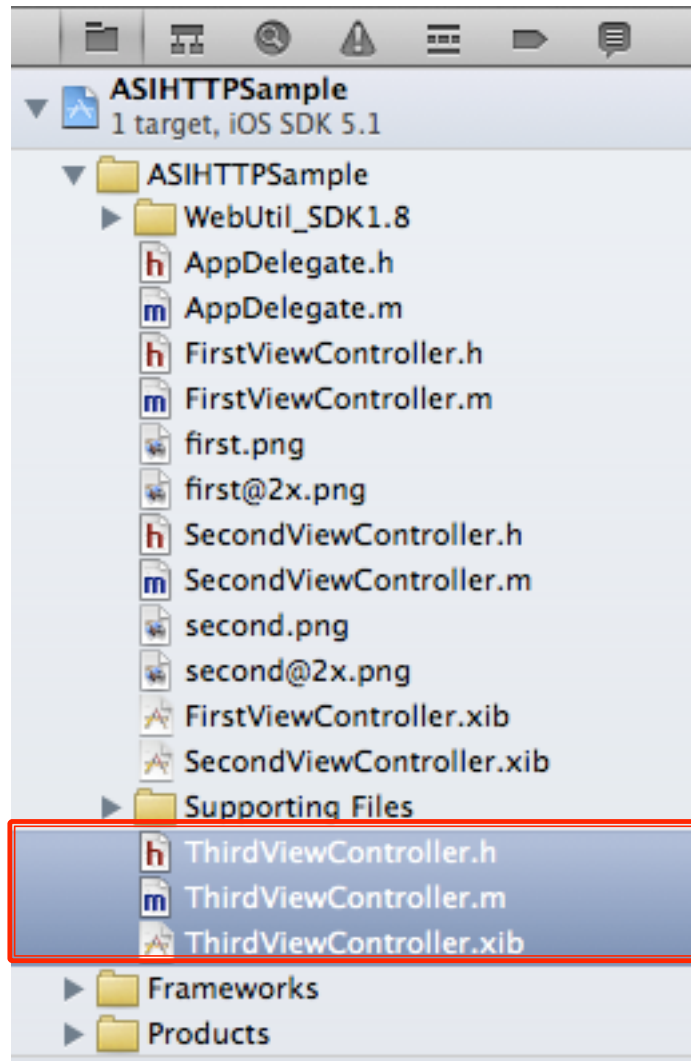
補充-加入第三個ViewController



加入第三個ViewController



加入第三個ViewController



修改AppDelegate.h

```
#import "AppDelegate.h"

#import "FirstViewController.h"
#import "SecondViewController.h"
#import "ThirdViewController.h"

@implementation AppDelegate

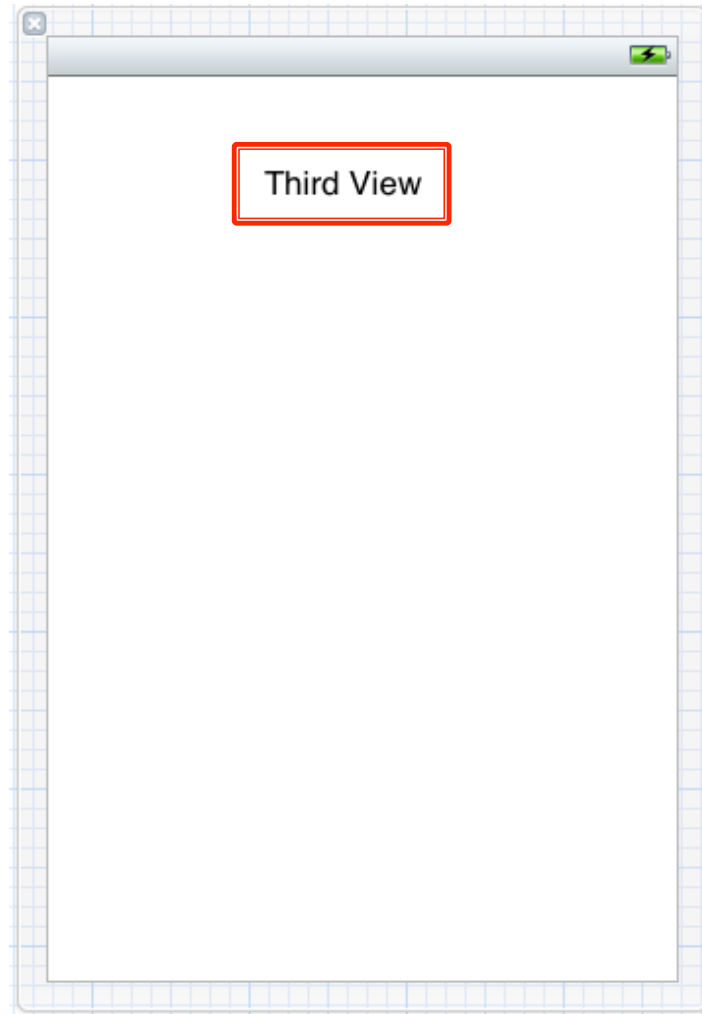
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    self.window = [[[UIWindow alloc] initWithFrame:[[UIScreen mainScreen] bounds]] autorelease];
    // Override point for customization after application launch.
    UIViewController *viewController1 = [[[FirstViewController alloc] initWithNibName:@"FirstViewController" bundle:nil
    ] autorelease];
    UIViewController *viewController2 = [[[SecondViewController alloc] initWithNibName:@"SecondViewController" bundle:
    nil] autorelease];
    UIViewController *viewController3 = [[[ThirdViewController alloc] initWithNibName:@"ThirdViewController" bundle:nil
    ] autorelease];
    self.tabBarController = [[[UITabBarController alloc] init] autorelease];
    self.tabBarController.viewControllers = [NSArray arrayWithObjects:viewController1, viewController2, viewController3
    , nil];
    self.window.rootViewController = self.tabBarController;
    [self.window makeKeyAndVisible];
    return YES;
}
```

實作(.m)

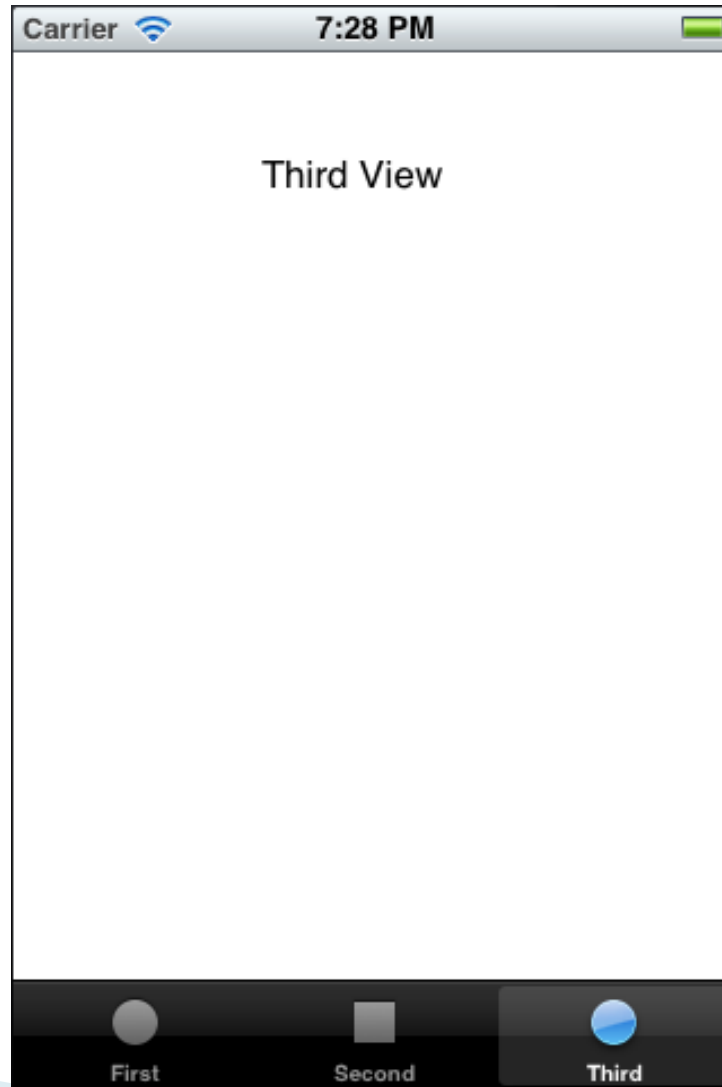
```
@implementation ThirdViewController

- (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil
{
    self = [super initWithNibName:nibNameOrNil bundle:nibBundleOrNil];
    if (self) {
        // Custom initialization
        self.title = NSLocalizedString(@"Third", @"Third");
        self.tabBarItem.image = [UIImage imageNamed:@"First"];
    }
    return self;
}
```

製作操作介面



執行結果



實作練習

▶ 簡易筆記本

