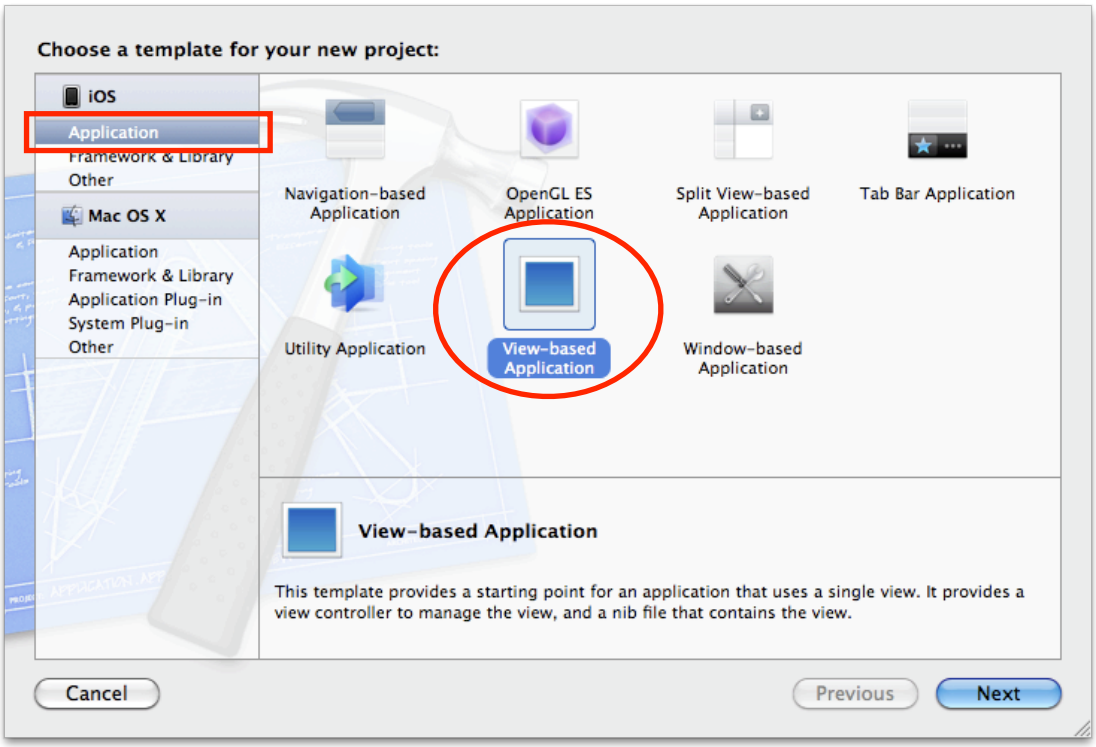
Lab SweetMemo

Step 1. 在File>New>New Project開啓一個新的專案, 在iOS的Application目錄裡面選擇 view based application

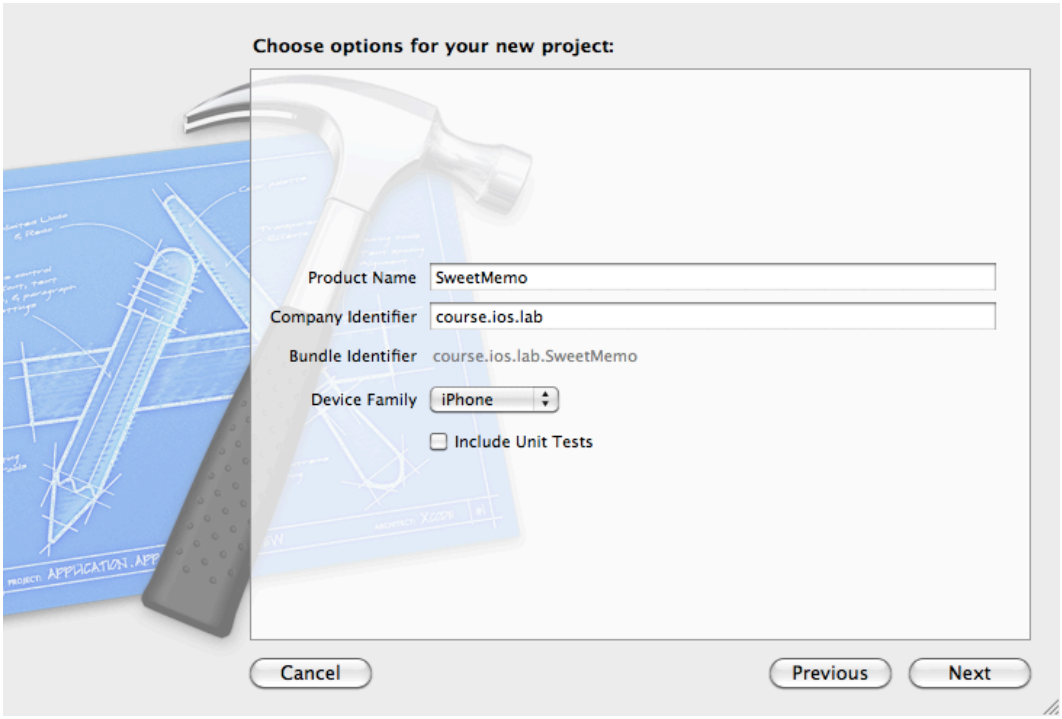


Step 2. 並將此專案命名為 **SweetMemo**

Company Identifier是填入Bundle的名稱,在此統一填入**course.ios.lab** (也可自行填入)

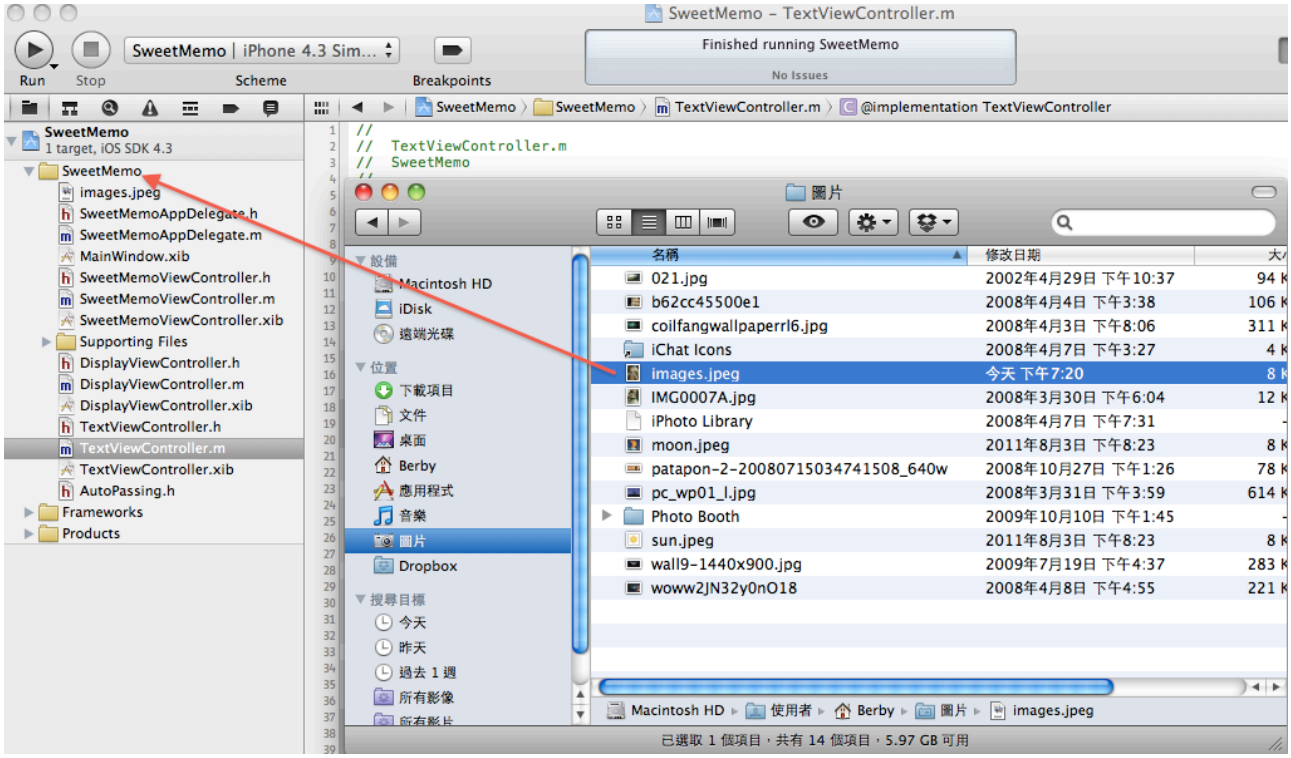
Device Family選擇**iPhone**

Include Unit Tests是做語意邏輯測試用,可勾選也可不勾選,在此我們統一**不勾選**, 存檔

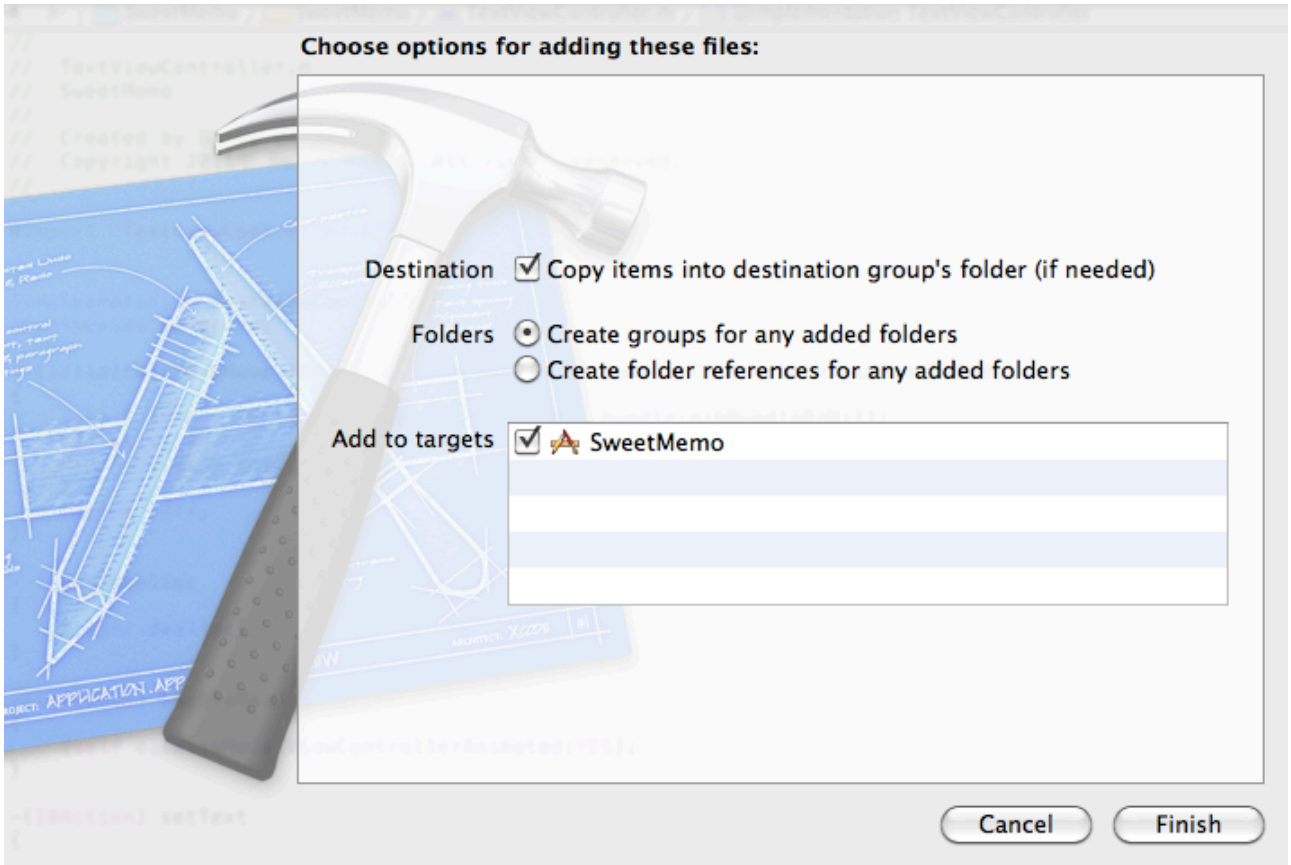


Step 3. 將準備好的圖檔 images, (練習用的圖檔可以在網路上取得)

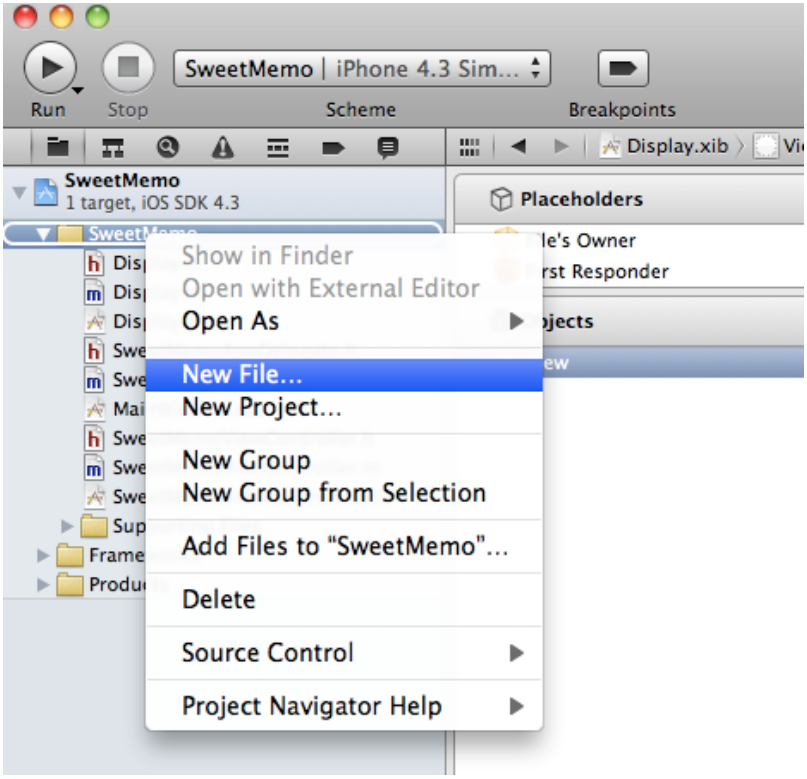
選擇 拖拉到Project內,



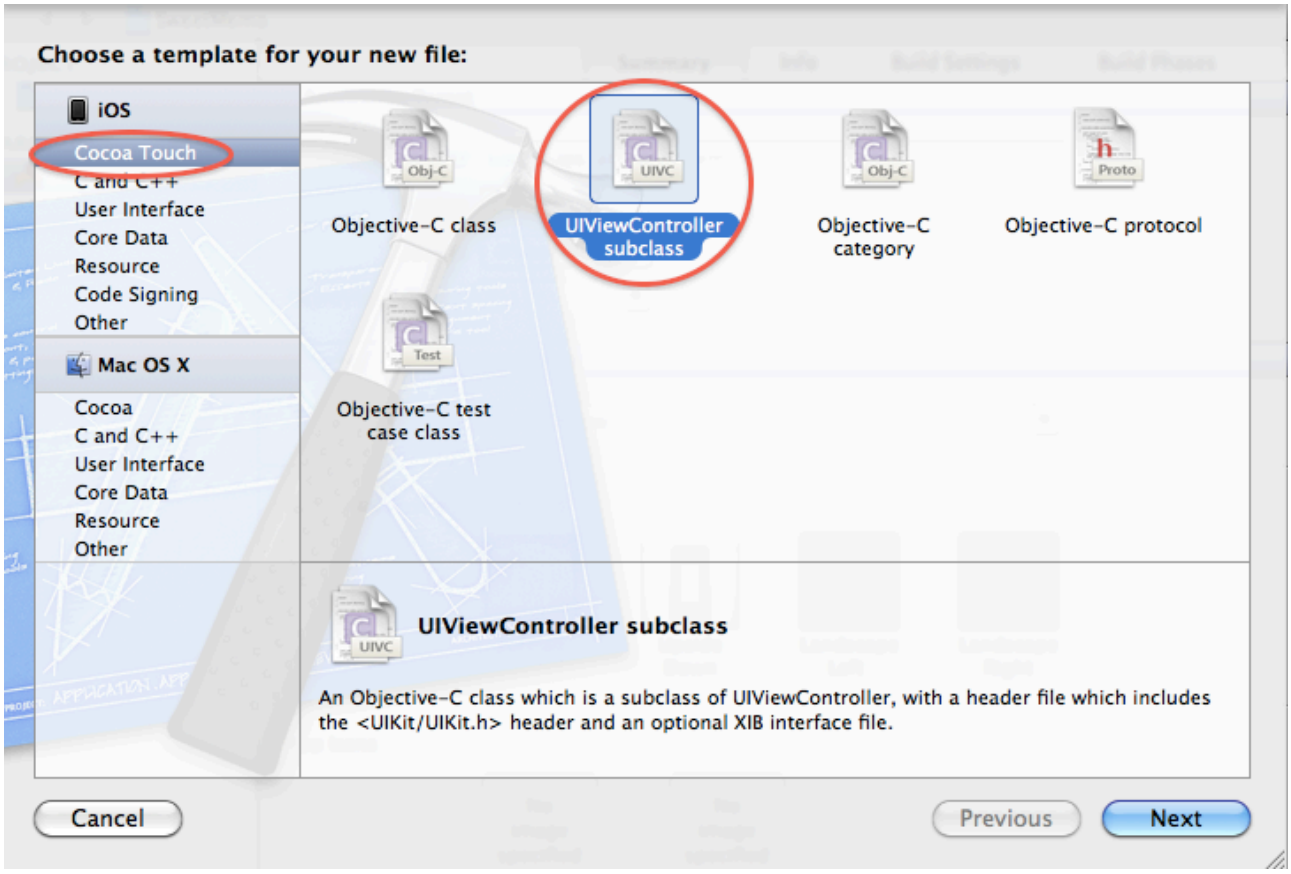
勾選 **Copy items into destination group's folder (if needed)**



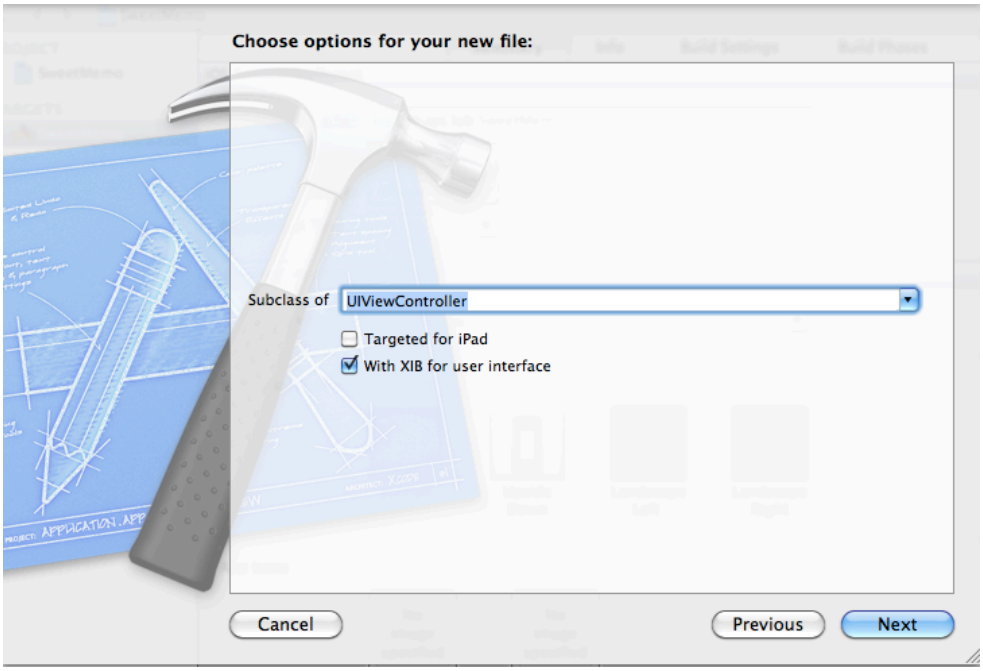
Step 4. 接著我們在SweetMemo project上點選右鍵(control + click), 選擇 add -> new file.



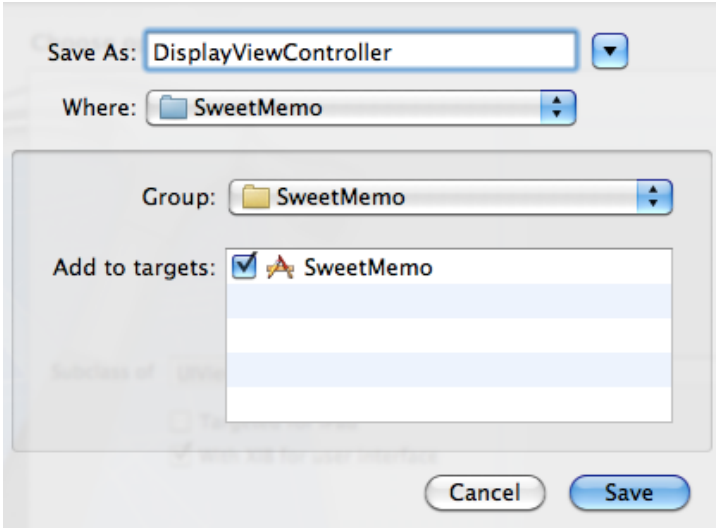
Step 5. 接著選取 UIViewController.



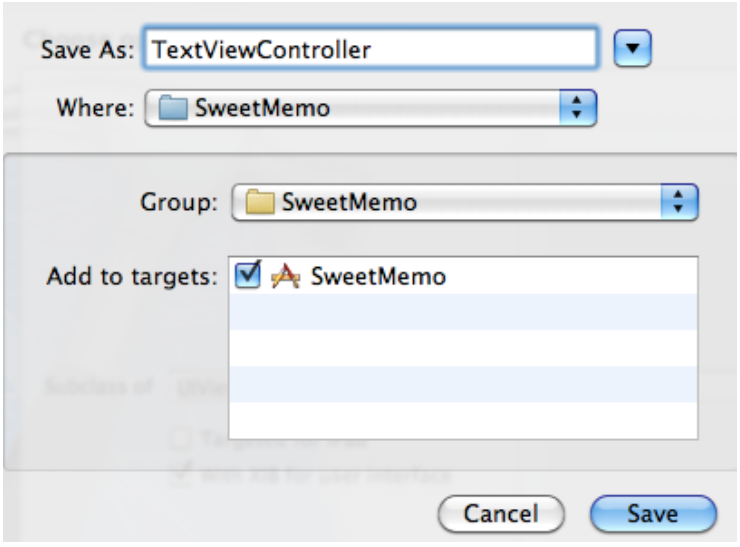
Step 6. 按下Next 之後, 選擇 Subclass of “UIViewController”



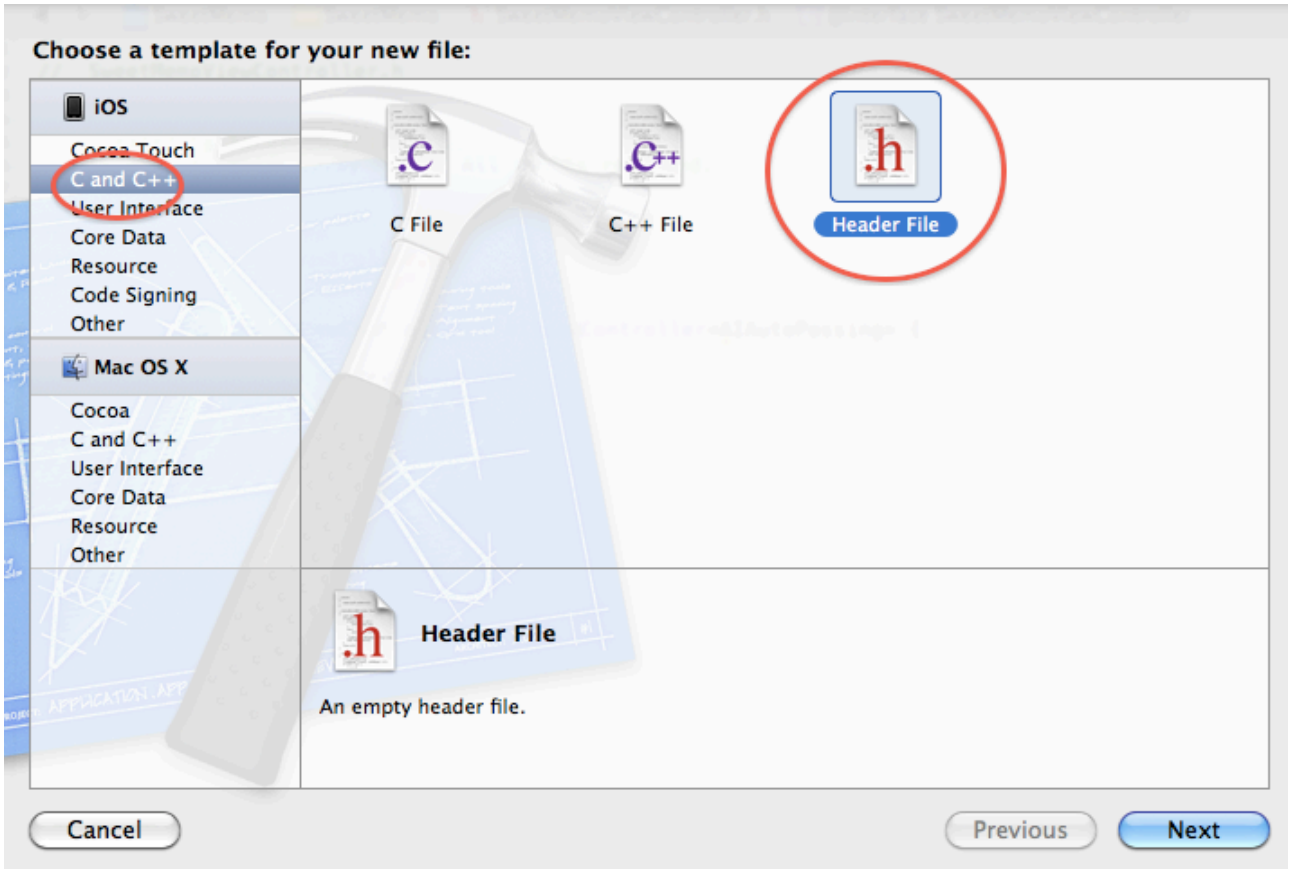
Step 7. 命名為 DisplayView



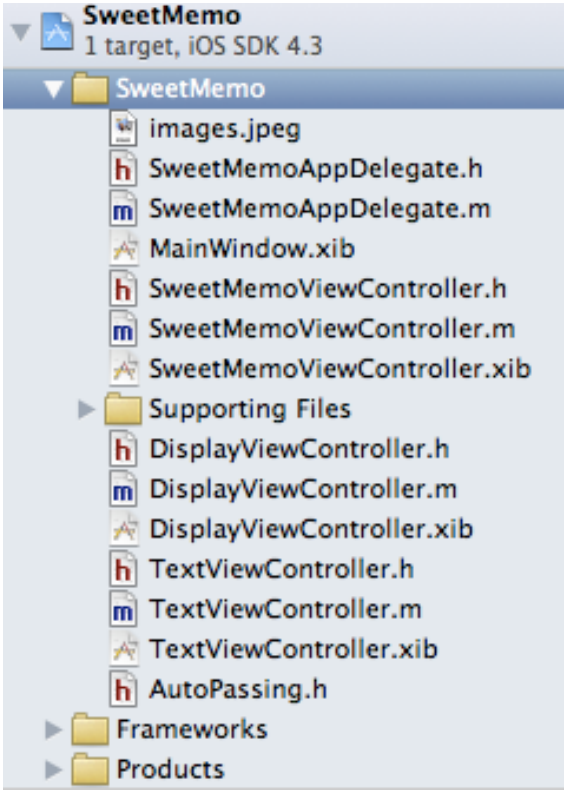
Step 8. 同樣的方法, 再新增另一個 UIViewController 命名為 TextView.



Step 9. 在這個範例中, 我們將使用 protocol, 來作 class 之間的溝通. 首先在SweetMemo project 上點選右鍵, 選擇 new file. 接著新增一個 header file, 取名為 AudoPassing.

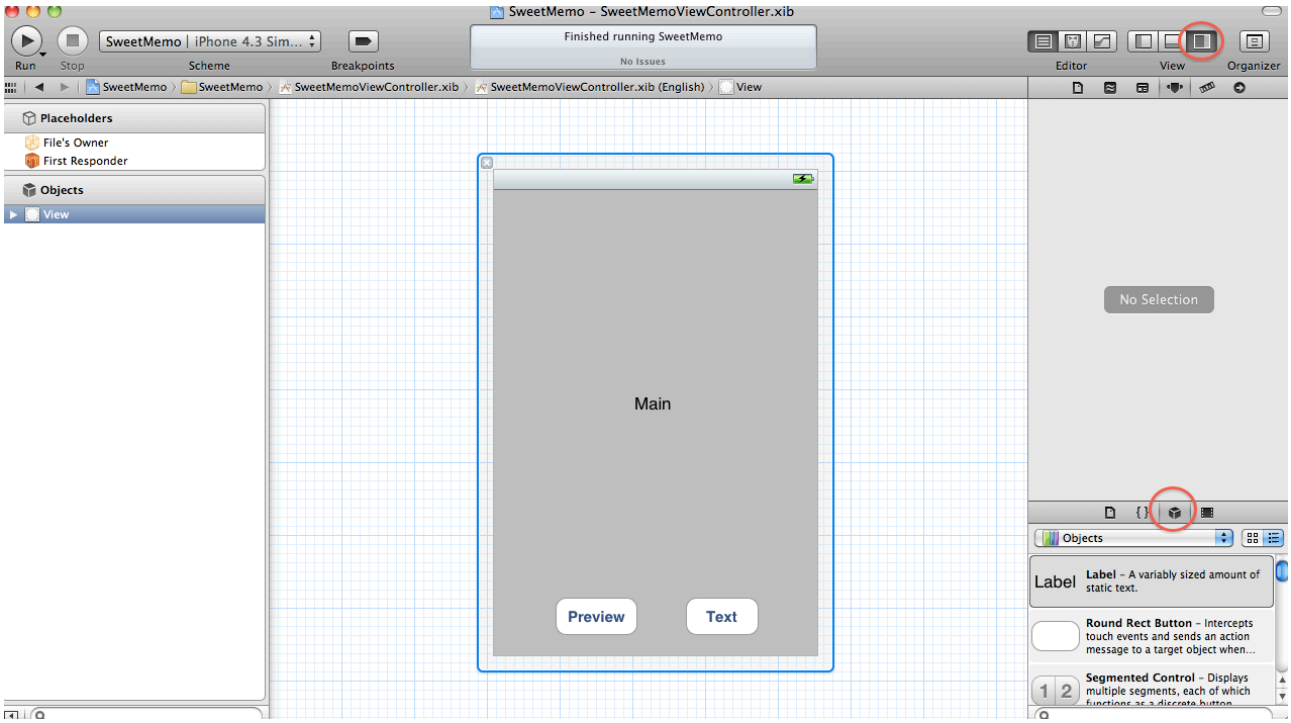


Step 10. 檢查 project navigator 是不是看起來如下圖



Step 11. 接著我們開始設計界面. 點選 SweetMemoViewController.xib.

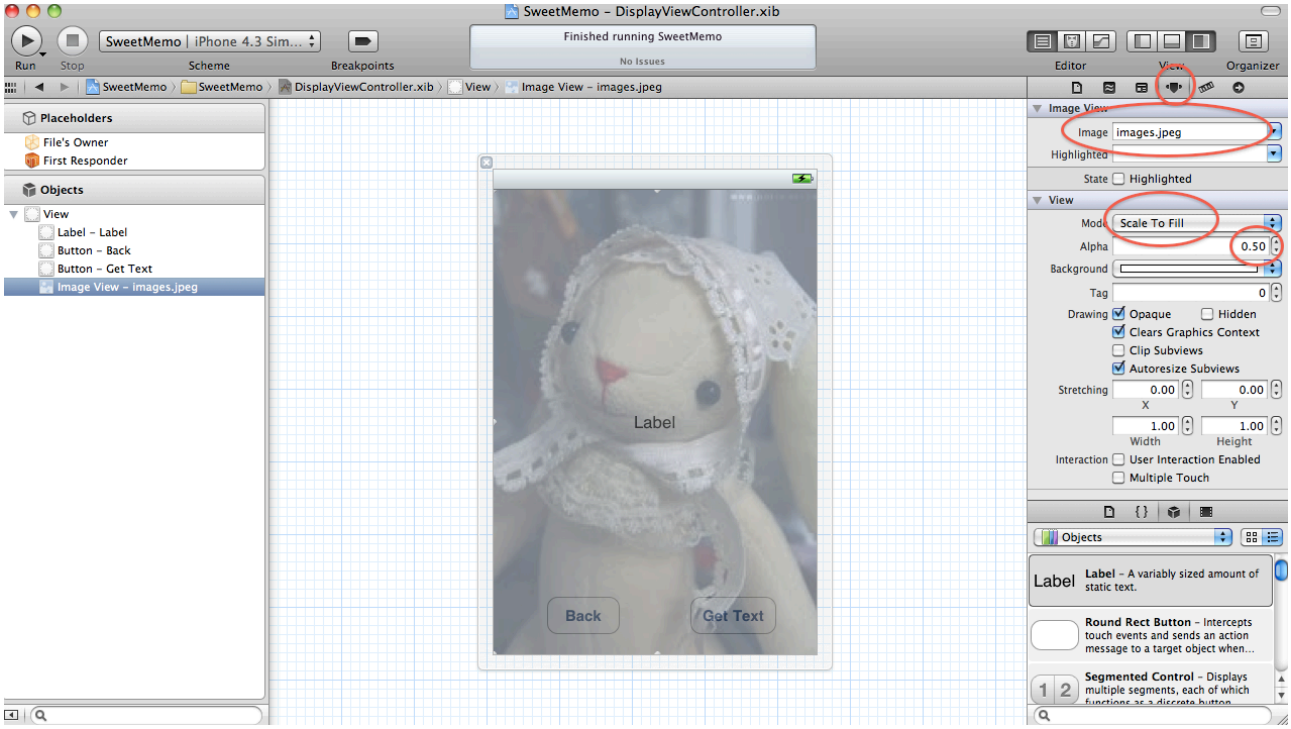
首先拉選一個Label, 改text為 Main. 拉選兩個Button, 改 Text 為 Preview 以及 Text.



Step 12. 點選 DisplayViewController, 首先在底面拉選一個Image view

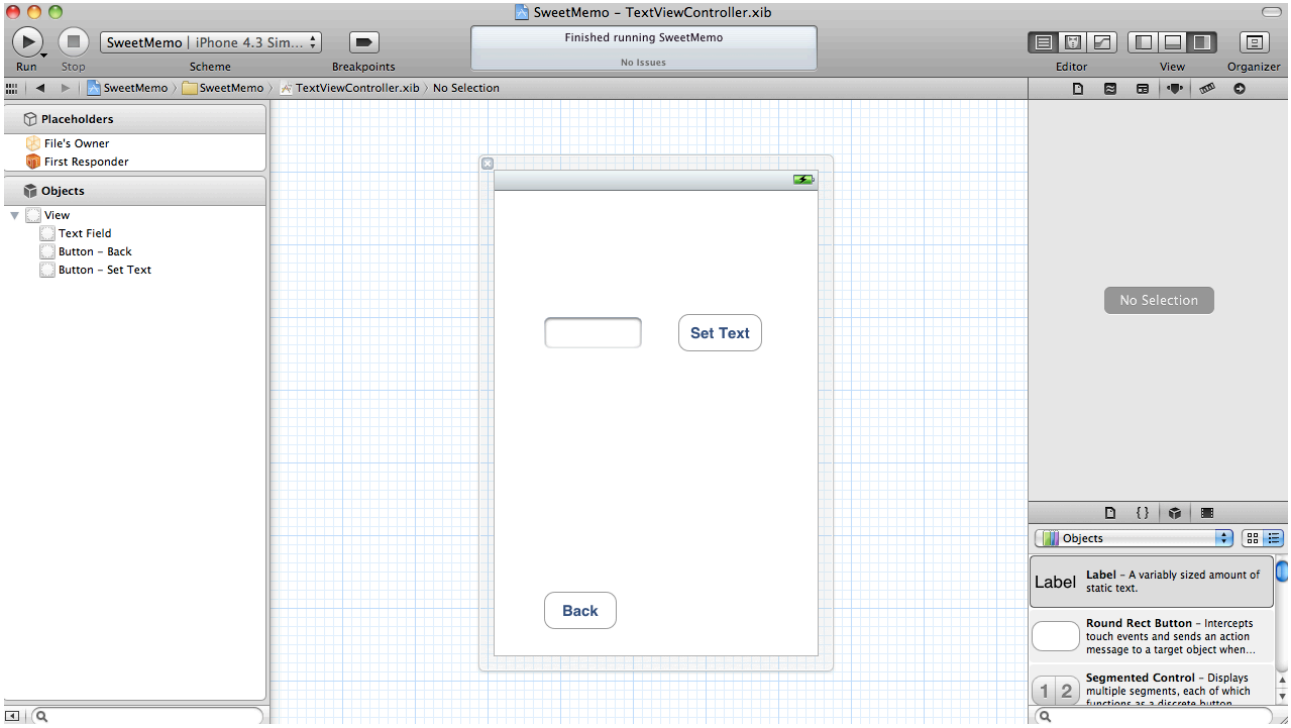
在Attribute inspector裡, 將 image 由下拉選單裡選擇為我們準備的images.jpeg. Mode 設為 Scale to fill. Alpha 設定為 0.5.

接著在中間拉選一個Label, 在下方拉選兩個 Button, 分別改 Text為 Back 以及 Get Text.



Step 13. 點選 TextViewController.xib,

我們在中間拉選一個 TextField. 兩個 Button, 分別命名為 Set Text 以及 Back.



Step 13. 我們的ViewController會用到 protocol, 所以在定義各個ViewController之前, 我們先點選 AutoString.h, 加入以下的 code 定義 protocol 裡的 action.

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


@protocol AIAutoPassing <NSObject>


-(void) passingView:(id)passingView target:(NSString *)targetID
passingData:(NSString *)passingData;


@end
```

Step 14. 接著我們將要在header檔案裡設定物件來與xib的界面連結.

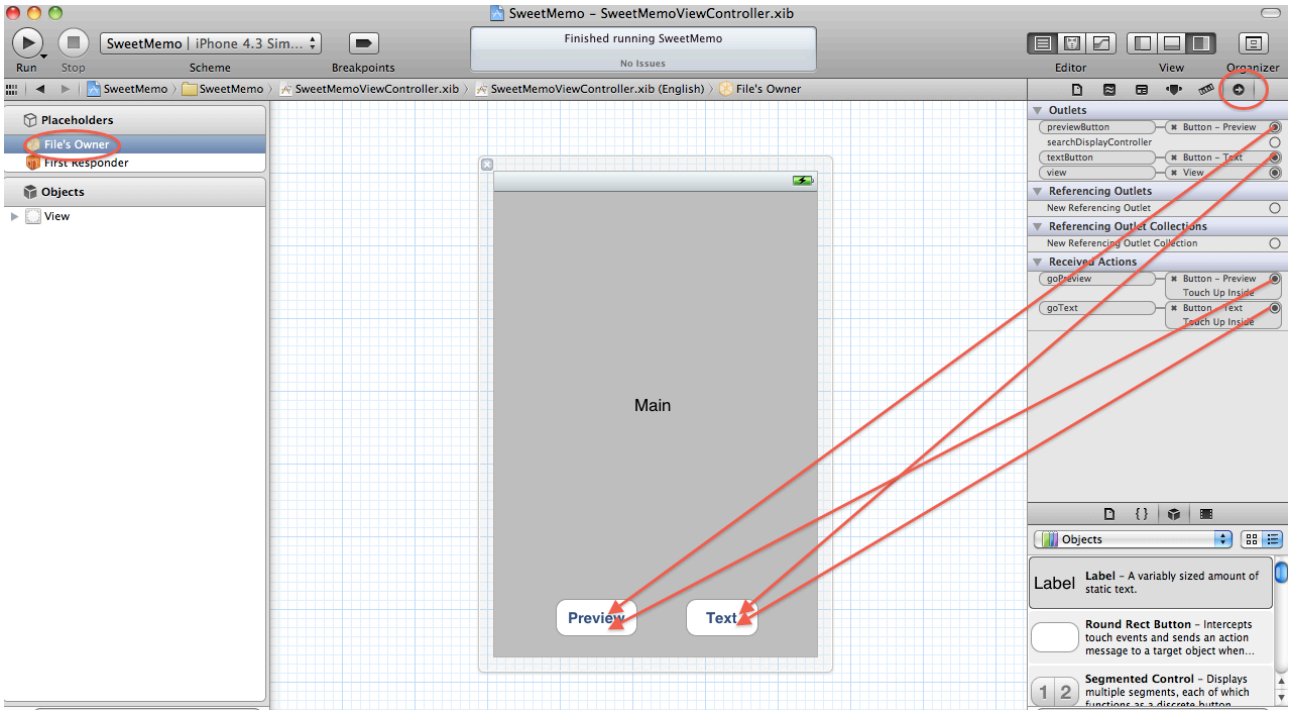
首先點選 SweetMemoViewController.h, 我們建立兩個 Button, 以及這兩個button的action.

另外定義兩個NSString 字串, 稍候用來輔助控制ViewController 的切換, 以及紀錄由 TextView 傳送而來的字串. 在這裡要記得在定義SweetMemoViewController的interface的時候, 要使用AIAutoPasing protocol.

```
#import <UIKit/UIKit.h>
#import "DisplayViewController.h"
#import "TextViewController.h"

@interface SweetMemoViewController : UIViewController<AIAutoPassing> {

    IBOutlet UIButton * previewButton;
    IBOutlet UIButton * textButton;
    NSString * targetView;

}

@property(retain) NSString * myData;

-(IBAction) goPreview;
-(IBAction) goText;

@end
```

Step 15. 接著點選 SweetMemoViewController.xib, 將物件與介面做連結.

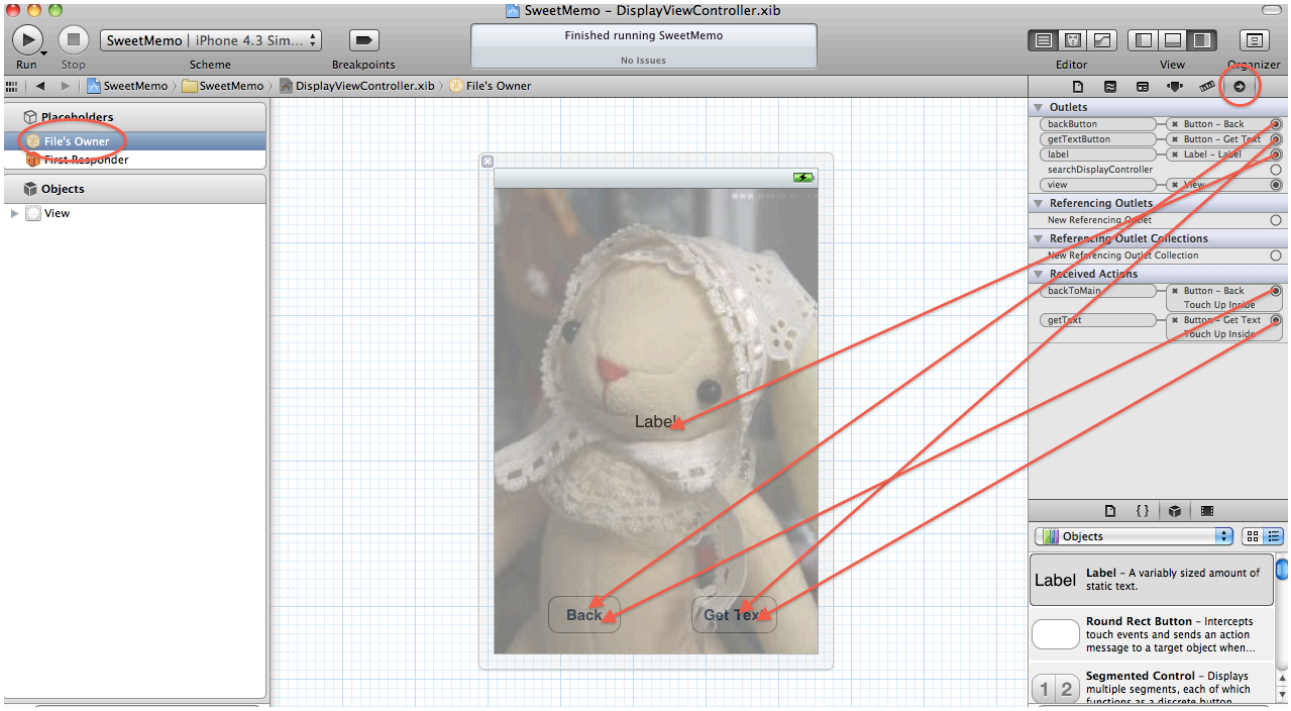
我們將 action 定義到 Button 的 Touch up inside.



Step 15. 點選DisplayViewController.h 加入以下的定義.

```
@interface DisplayViewController : UIViewController<AIAutoPassing> {

    id<AIAutoPassing> deletage;
    IBOutlet UILabel * label;
    IBOutlet UIButton * backButton;
    IBOutlet UIButton * getTextButton;

}

@property(nonatomic, retain)id<AIAutoPassing>delegate;
@property(retain) NSString * memo;
-(IBAction) backToMain;
-(IBAction) getText;


@end
```

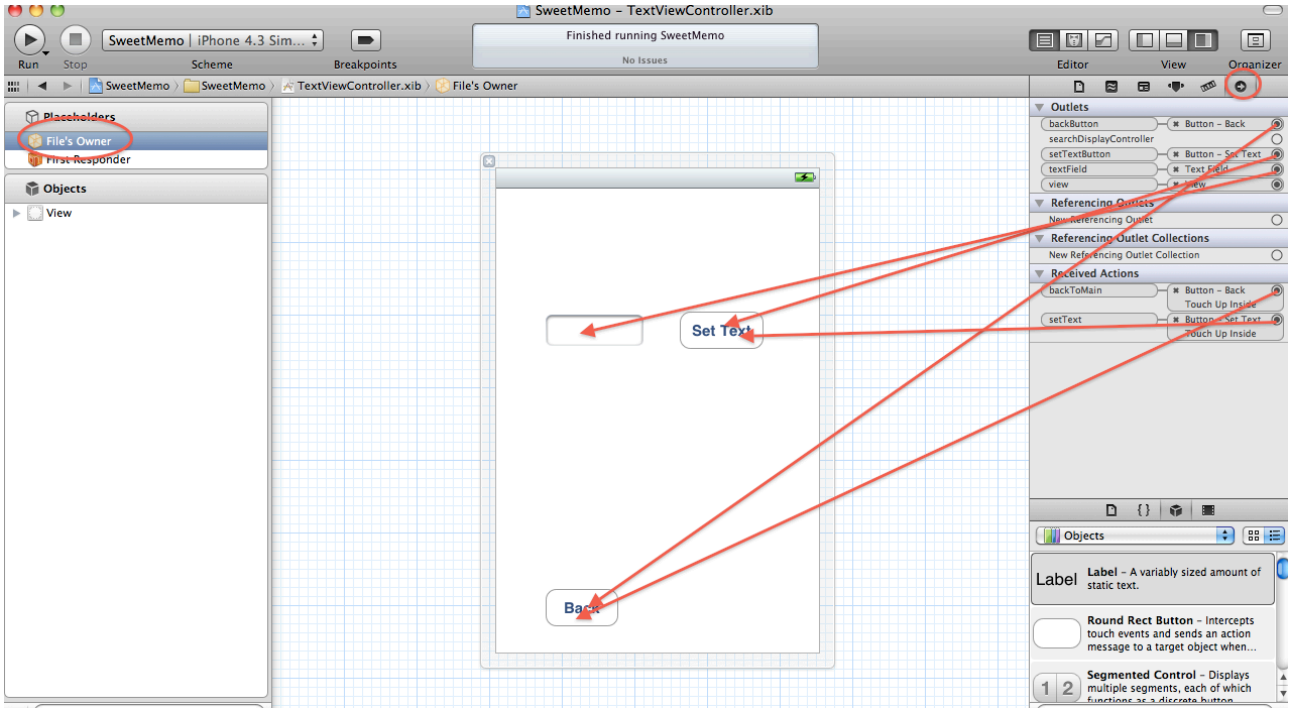
Step 16. 接著點選 DisplayViewController.xib, 將物件與介面做連結.



Step 17. 接著點選 TextViewController.h, 加入以下定義.

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


@interface TextViewController : UIViewController<AIAutoPassing>{

    id<AIAutoPassing>deletage;
    IBOutlet UIButton * backButton;
    IBOutlet UIButton * setTextButton;
    IBOutlet UITextField * textField;

}
@property(nonatomic, retain)id<AIAutoPassing>delegate;

-(IBAction) backToMain;
-(IBAction) setText;

@end
```

Step 18. 然後點選 TextViewController.xib 將物件與介面做連結.



Step 19. 定義完以後, 我們接著實作我們的module.

首先點選 DisplayViewController.m.

因為我們在header檔中有定義@property, 所以在 @implementation 後面, 我們必須加入 @synthesize (@property與@synthesize成對出現)

```
@implementation DisplayViewController
@synthesize delegate;
@synthesize memo;
```

接著實作 AIAutoPassing protocol 中的 action, passingView.

在這裡我們不做任何事, 留白.

```
-(void) passingView:(id)passingView target:(NSString *)targetID
passingData:(NSString *)passingData
{

}
```

接著實作 backToMain 以及 getText 這兩個IBAction.

在backToMain 裡, 我們直接 將目前的 modal view 給 dismiss.

在 getText 中, 我們利用passingView, 通知 SweetMemoViewController 來掛上 TextViewController.

```
-(IBAction) backToMain
{
    [self dismissModalViewControllerAnimated:YES];
}
-(IBAction) getText
{
    if([self.delegate respondsToSelector:@selector
(passingView:target:passingData:)]){
        [self.delegate passingView:self target:@"TextView"
passingData:nil];
    }
    [self dismissModalViewControllerAnimated:YES];
}
```

我們利用在 header檔中定義的 NSString memo 來設定 label, 所以當 memo 的值被設定而 DisplayView 被喚起時, 我們在 viewDidLoad中, 設定label 為 memo 所紀錄的 文字.

```
- (void)viewDidLoad
{
    [super viewDidLoad];
    [label setText:memo];

}
```

Step 20. 接著實作 TextViewController.m,

同樣的, 首先加入 @synthesize

```
@implementation TextViewController
@synthesize delegate;
```

接著實作 AIAutoPassing protocol 中的 action, passingView.

在這裡我們不做任何事, 留白.

```
-(void) passingView:(id)passingView target:(NSString *)targetID
passingData:(NSString *)passingData
{

}
```

然後實作 backToMain 以及 setText 這兩個 action.

注意在setText中, 我們將 textField的text使用 passingView傳送給 SweetMemoViewController.

```
-(IBAction) backToMain
{
    [self dismissModalViewControllerAnimated:YES];
}

-(IBAction) setText
{

    if([self.delegate respondsToSelector:@selector
(passingView:target:passingData:)]){

        [self.delegate passingView:self target:@"DisplayView"
passingData:textField.text];

    }

    [self dismissModalViewControllerAnimated:YES];
}
```

Step 21. 接著實作 SweetMemoViewController.m.

首先加入 synthesize.

```
@implementation SweetMemoViewController
@synthesize myData;
```

然後實作 passingView 這個 protocol 裡的 action.

```
-(void) passingView:(id)passingView target:(NSString *)targetID
passingData:(NSString *)passingData
{
    targetView = targetID;
    self.myData = passingData;
}
```

在這裡我們可以看到, 我們將由passingView傳來的直當作切換modalview的依據, 存在 targetView. 而將passingView傳送而來的 data, 放在 self.myData.

接著實作 goPreview 以及 goText這兩個 IBAction. 我們可以注意到, 當我們利用 DisplayViewController 實作instance之後,

```
viewController.delegate = self;
```

將instance中的delegate設置成self, 也就是將delegate設置成SweetMemoViewController.

```
-(IBAction) goPreview
{
    DisplayViewController * viewController = [[DisplayViewController
alloc] initWithNibName:@"DisplayViewController" bundle:nil];
    viewController.delegate = self;
    viewController.modalTransitionStyle =
UIModalTransitionStyleCoverVertical;
    [self presentModalViewController:viewController animated:YES];
    [viewController release];
}

-(IBAction) goText
{
    TextViewController * viewController = [[TextViewController alloc]
initWithNibName:@"TextViewController" bundle:nil];
    viewController.delegate = self;
    viewController.modalTransitionStyle =
UIModalTransitionStyleCrossDissolve;
    [self presentModalViewController:viewController animated:YES];
    [viewController release];

}
```

最後我們實作 viewDidAppear.

```
-(void) viewDidAppear:(BOOL)animated
{
    if([targetView isEqualToString:@"TextView"])
    {
        TextViewController * viewController = [[TextViewController
alloc] initWithNibName:@"TextViewController" bundle:nil];
        viewController.delegate = self;
        viewController.modalTransitionStyle =
UIModalTransitionStyleCrossDissolve;
        [self presentModalViewController:viewController animated:YES];
        [viewController release];
        targetView = nil;
        self.myData = nil;

    }
```

(viewDidAppear的實作接下頁 ...)

```
if([targetView isEqualToString:@"DisplayView"])
    {
        DisplayViewController * viewController = [[DisplayViewController
alloc] initWithNibName:@"DisplayViewController" bundle:nil];
        viewController.delegate = self;
        viewController.memo = myData;
        viewController.modalTransitionStyle =
UIModalTransitionStyleCoverVertical;
        [self presentModalViewController:viewController animated:YES];
        [viewController release];
        targetView = nil;
        self.myData = nil;

    }
}
```

我們將頁面的判斷以及present modalview 的工作放在 viewDidAppear 的目的是因為, 在我們設計的邏輯是: 首先由 main view 喚出 TextView, 設置完text後按下set text button(又或者是back button)之後, TextView會d被dismiss, 接著controller會回到main view. 所以我們在viewDidAppear去做判斷是十分合適的.

最後記得在viewDidUnload將以經不用的物件給release.

```
- (void)viewDidUnload
{
    [targetView release];
    [myData release];
    [super viewDidUnload];

    // Release any retained subviews of the main view.
}
```

Step 22. 最後 Run (⌘+R), 我們可以藉著 TextView來設置DisplayView的 Label.