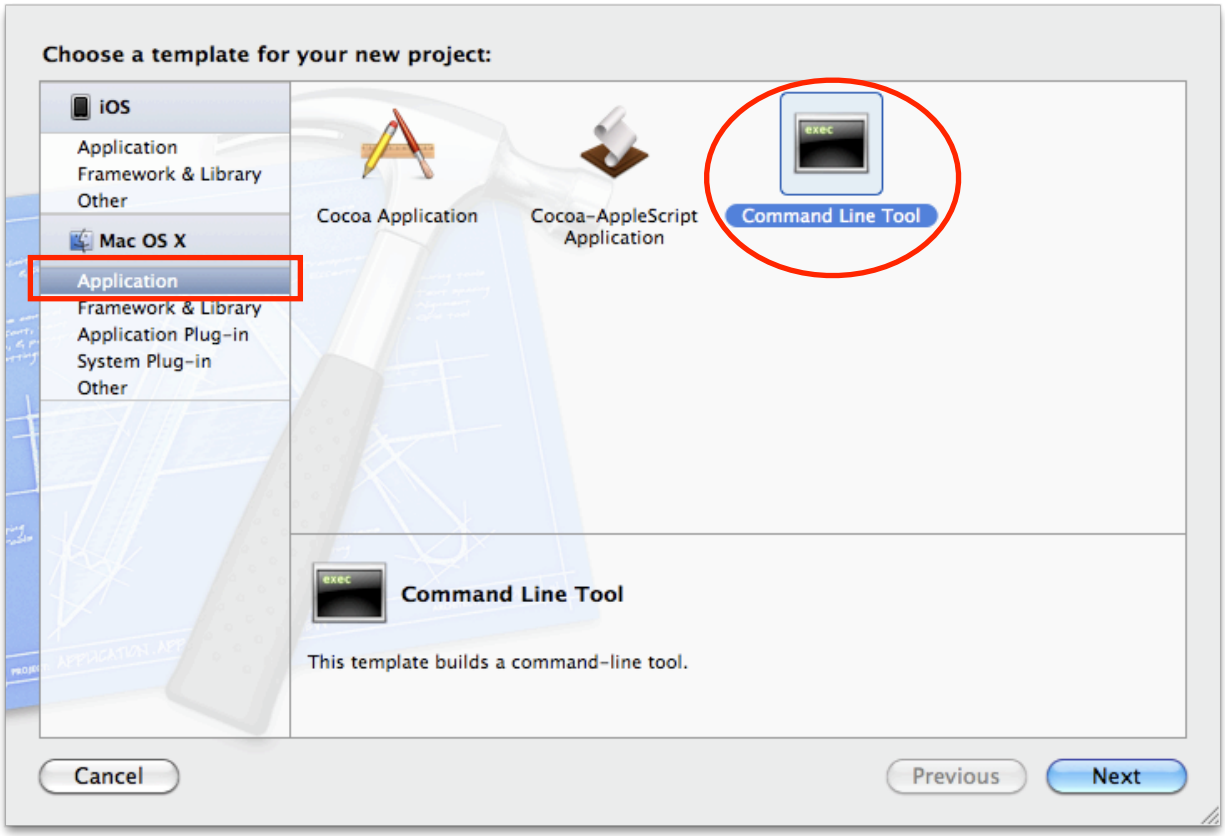
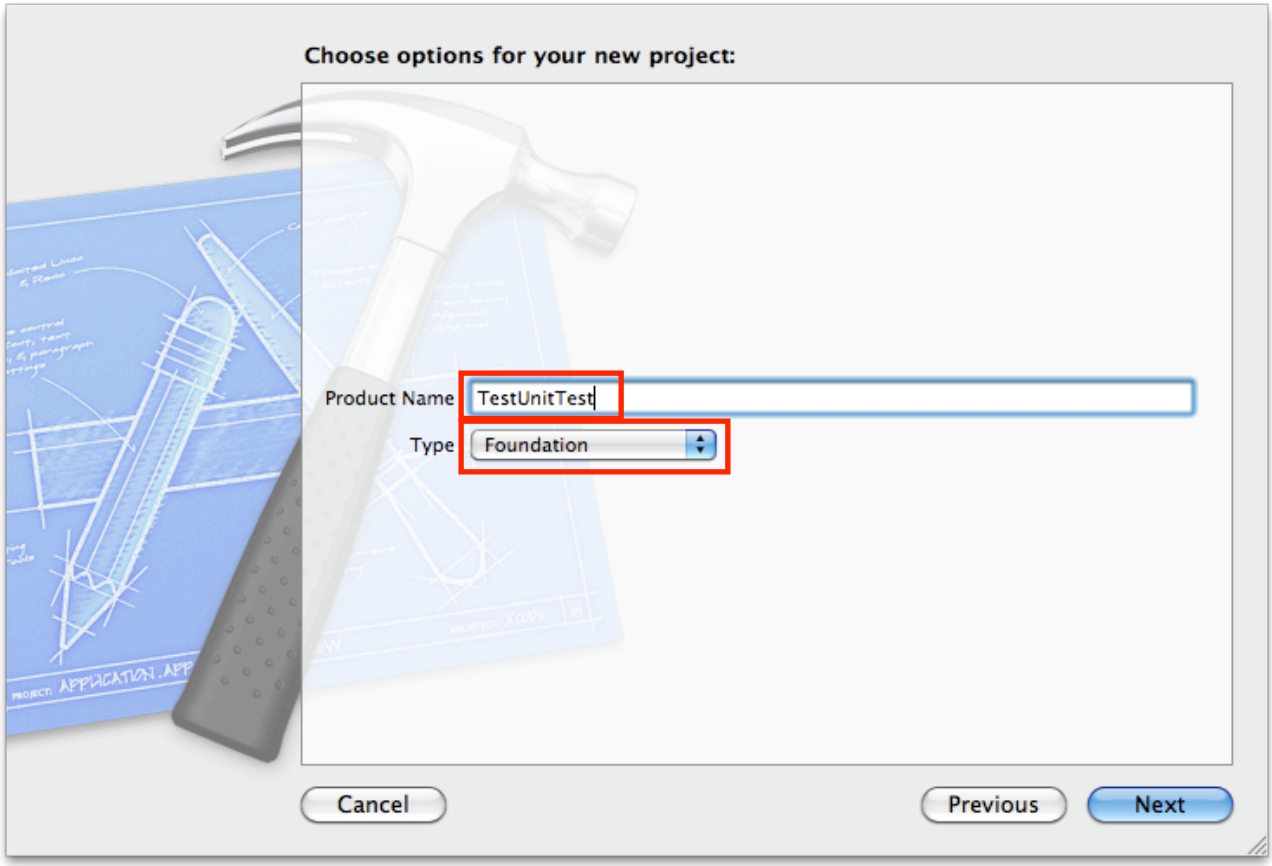
Lab TestUnitTest

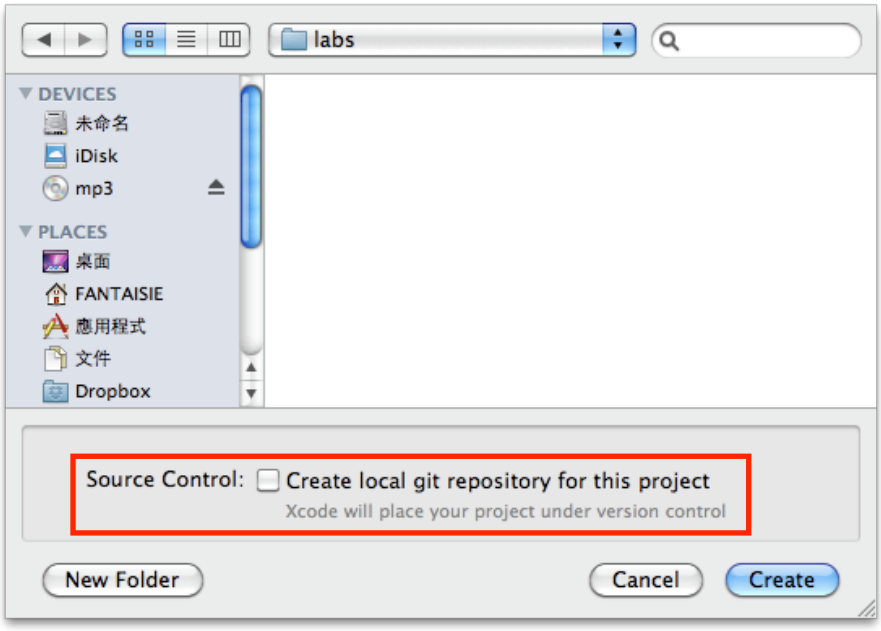
Step1. 在File > New > New Project開啓一個新的project, 選擇MAC OS X的Application目錄裡的 **Command Line Tool**



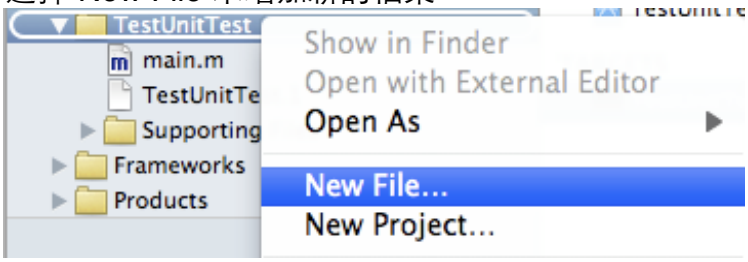
Type選擇 **Foundation**, 將project命名為 **TestUnitTest**



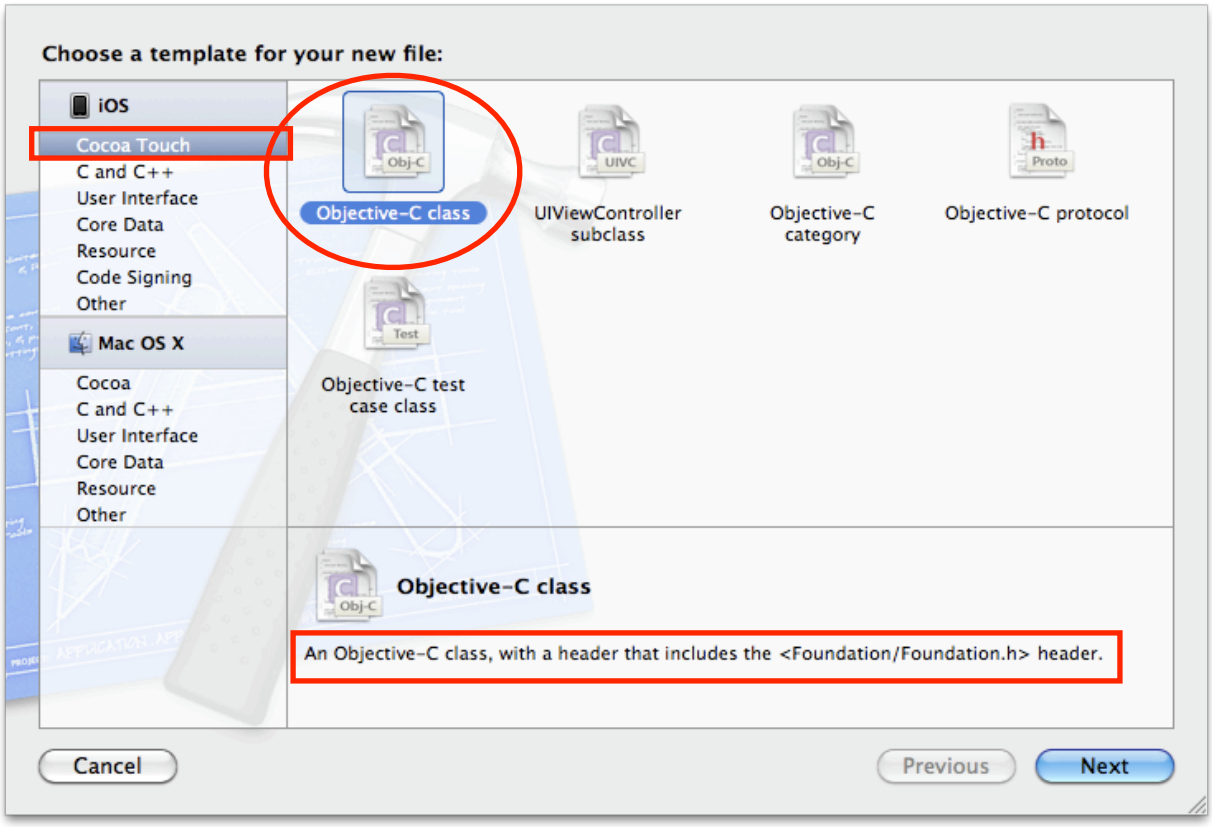
在此我們不做version control的動作,所以不勾選Create local git repository for this project



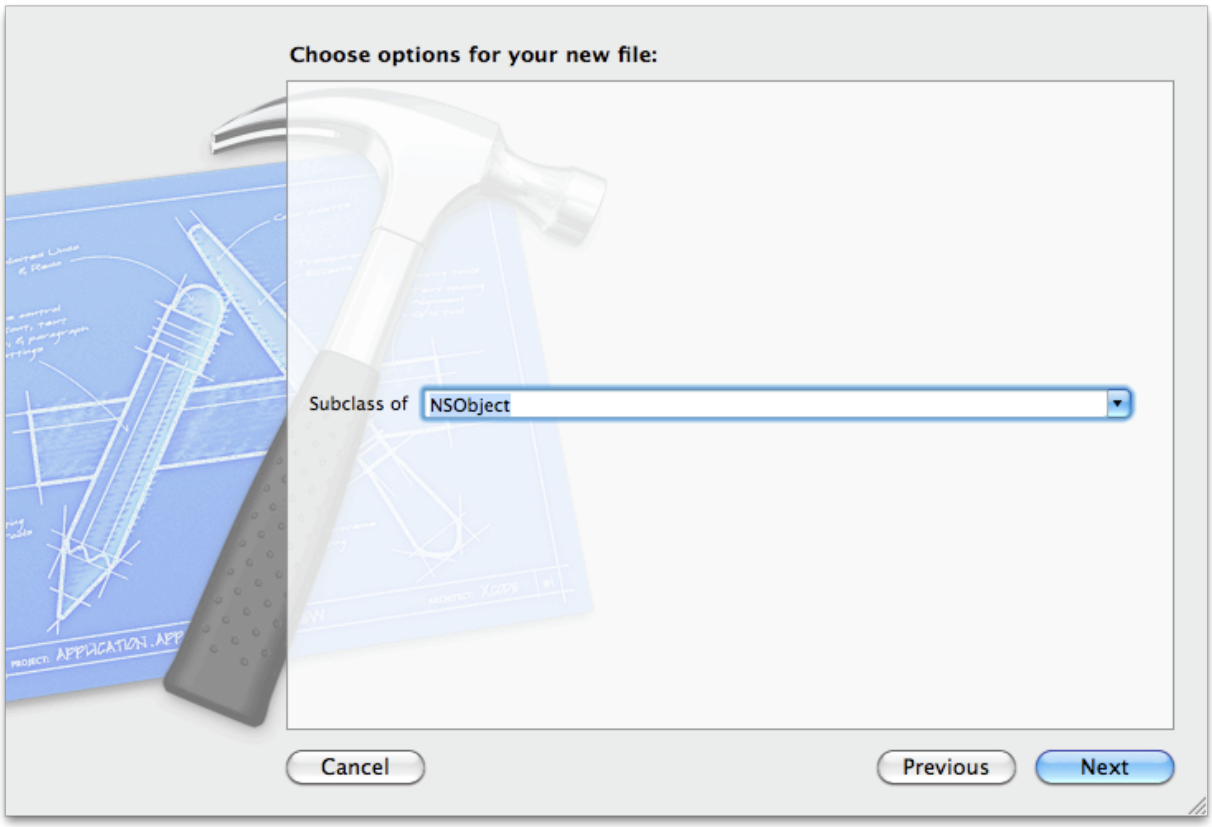
Step 2. 在Xcode左邊Project Navigator 視窗中, 在TestUnitTest上點右鍵(若無滑鼠ctrl+點擊)選擇 New File 來增加新的檔案



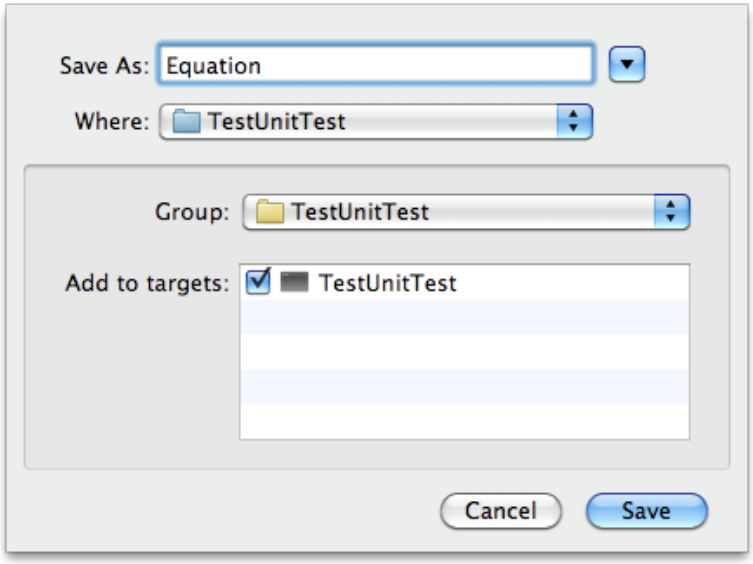
選擇iOS裡的Coca Touch 裡的Objective-C class, 下方有敘述這個class有includes <Foundation/Foundation.h> 這個標頭檔

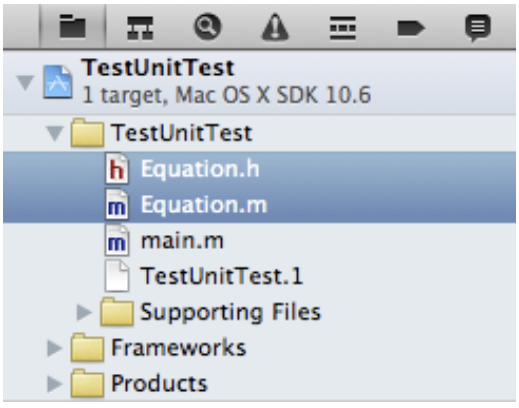


選擇Subclass of **NSObject**



Step 3. 我們要實作一個計算指數次方的程式, 將這個class命名為 **Equation**, 來產生 Equation.m和 Equation.h





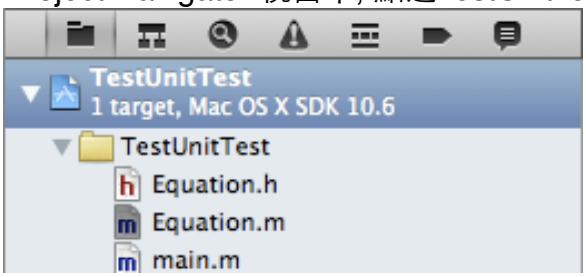
Step 4. 在Equation.h裡面宣告我們要計算的基礎數字,對他做@property來自動以及可輸入參數值次方的指數運算

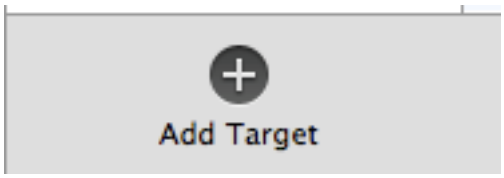
```
#import <Foundation/Foundation.h>


@interface Equation : NSObject {
    int number;
}
@property (assign) int number;

-(int) getResult:(int)exponent;

@end
```

Step 5. 在Equation.m裡實作 `getResult:(int)exponent` 取得做輸入參數值次方的指數運算結果

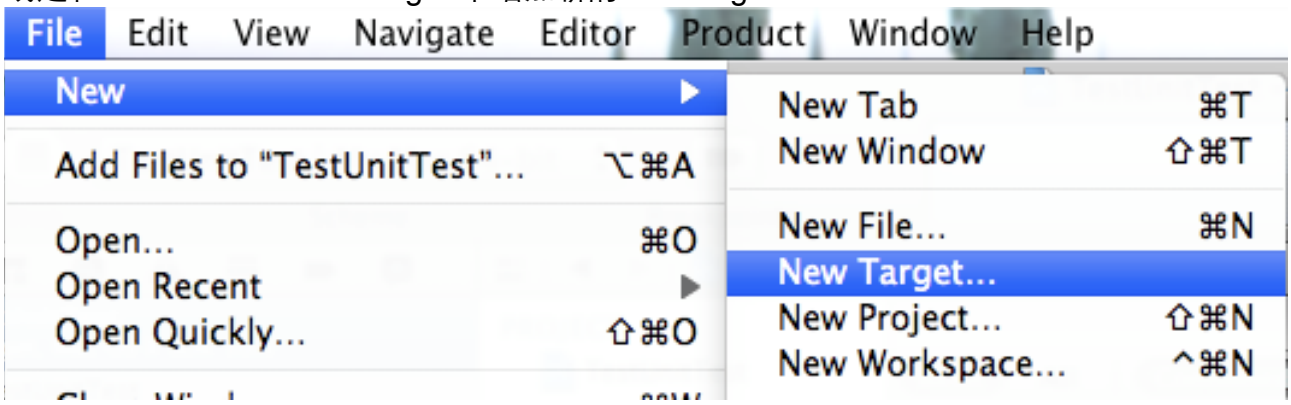
```
#import "Equation.h"

@implementation Equation
@synthesize number;

-(int) getResult:(int)exponent{
    return (number) ^ exponent;
}

@end
```

Step 6. 接下來是來測試我們計算指數運算的method是否符合我們需要的邏輯,在Xcode左邊Project Navigator 視窗中, 點選TestUnitTest Project檔, 在中間最下面在點選Add Target

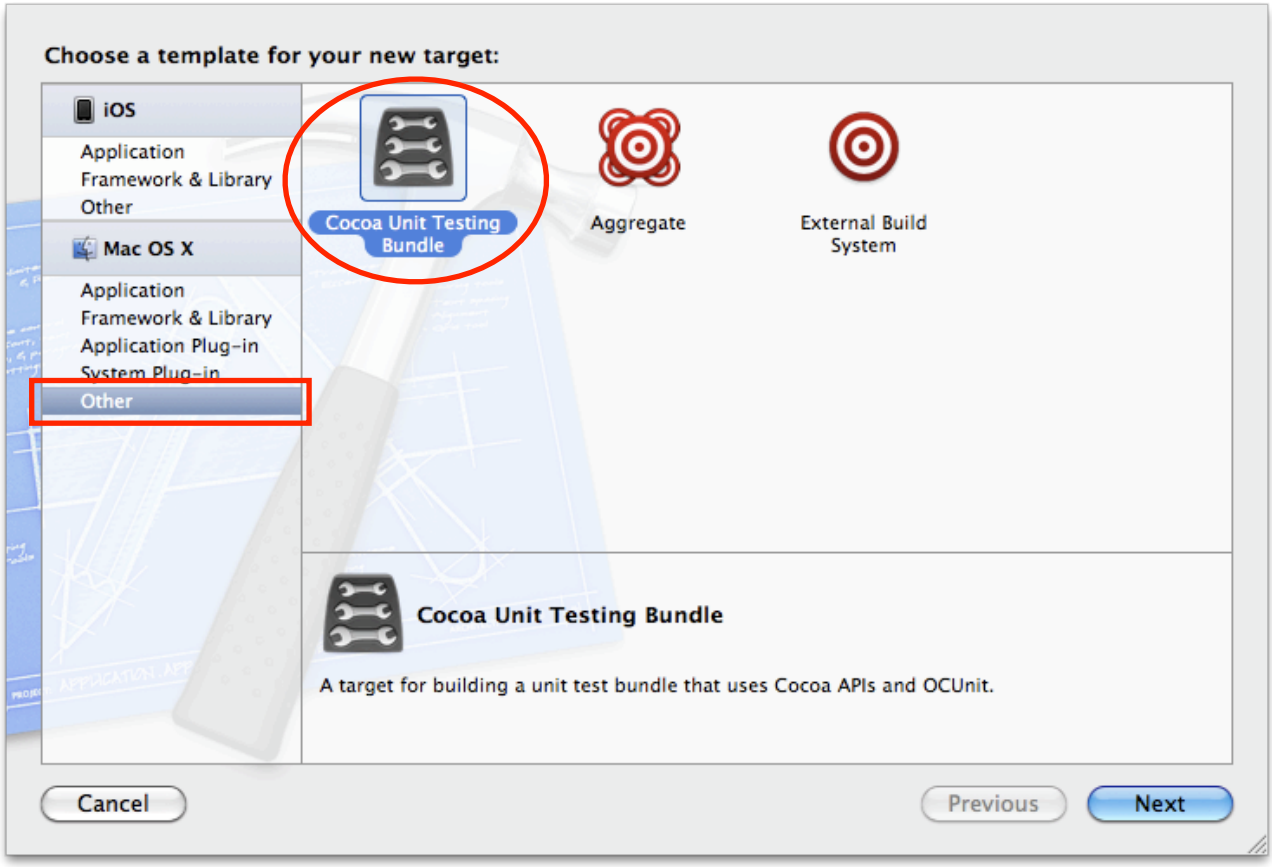




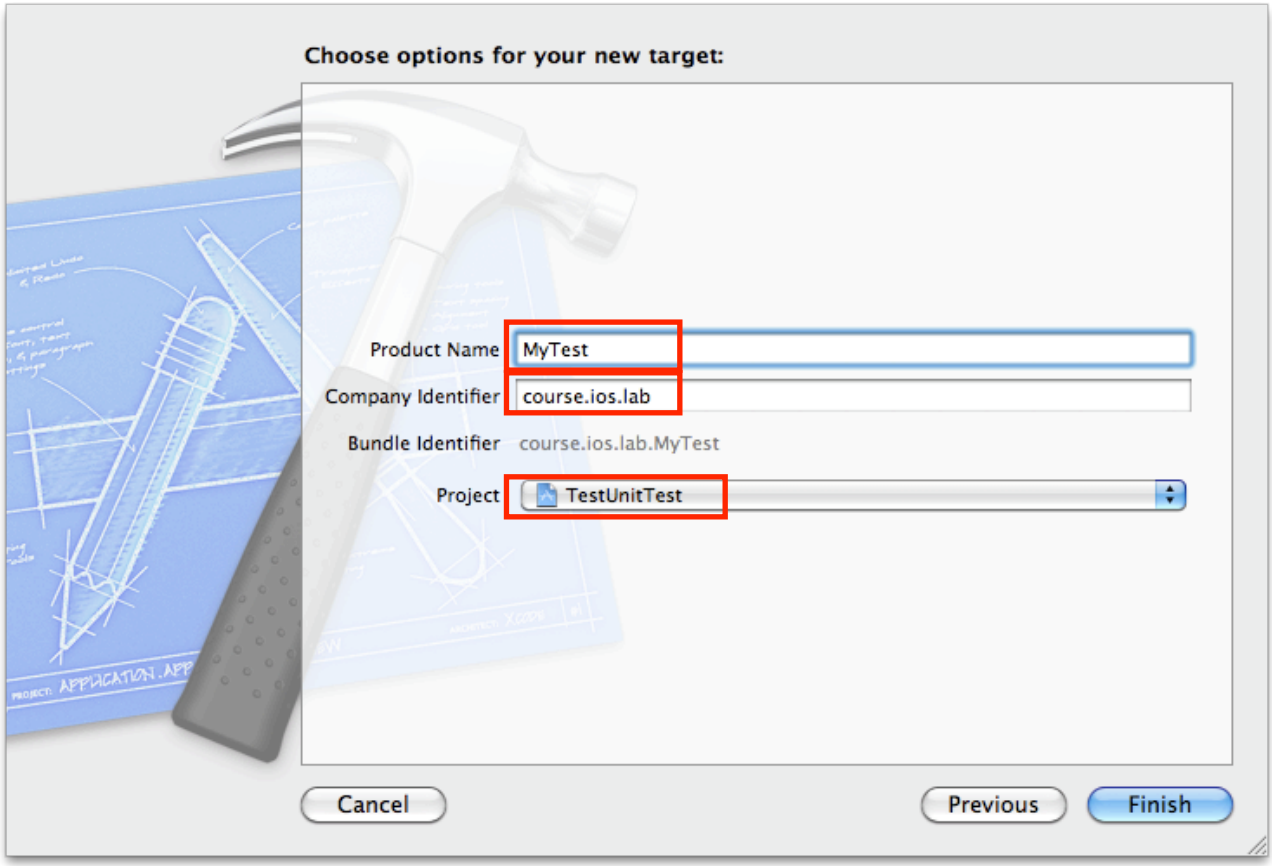
或是在File > New > New Target 來增加新的Test Target



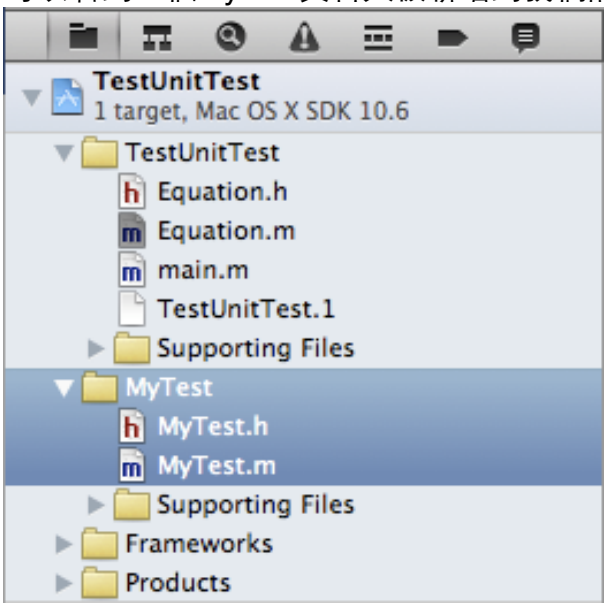
選擇Mac OS X的Other目錄, 選擇Cocoa Unit Testing Bundle



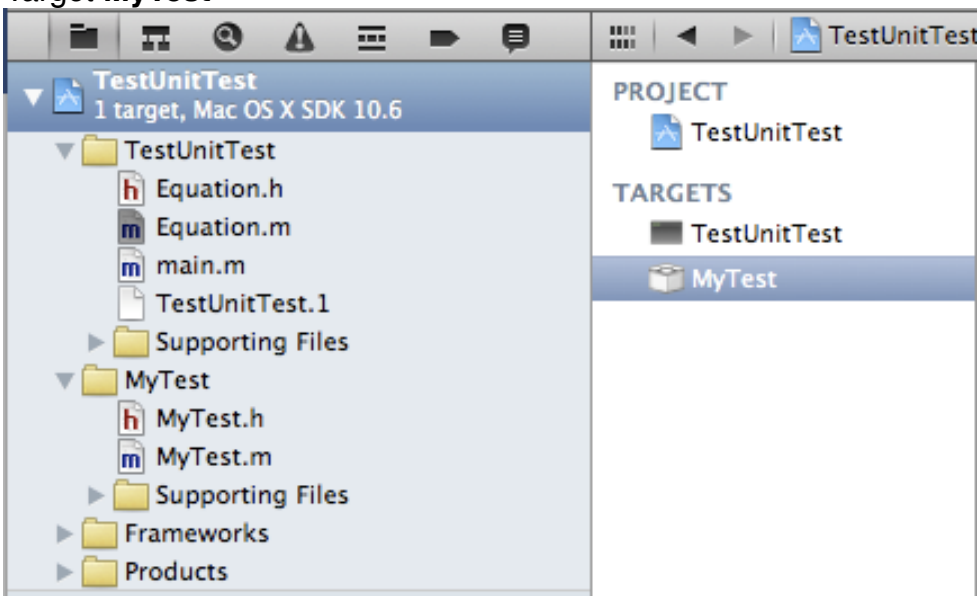
Step 7. 將新增的New Unit Test Bundle的Product Name命名為MyTest, Company Identifier仍命為 course.ios.lab



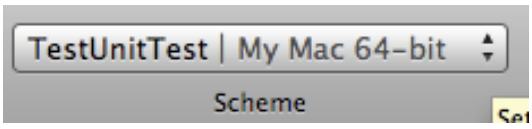
可以看到一個MyTest資料夾被新增到我們的Project裡

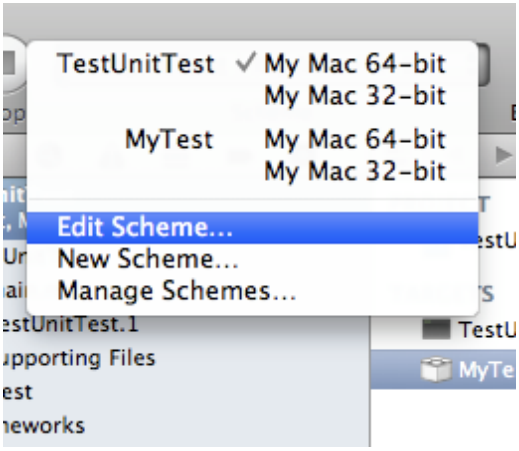


在Xcode左邊Project Navigator 視窗中, 點選TestUnitTest Project檔, 也可以看到我們新增的Target **MyTest**

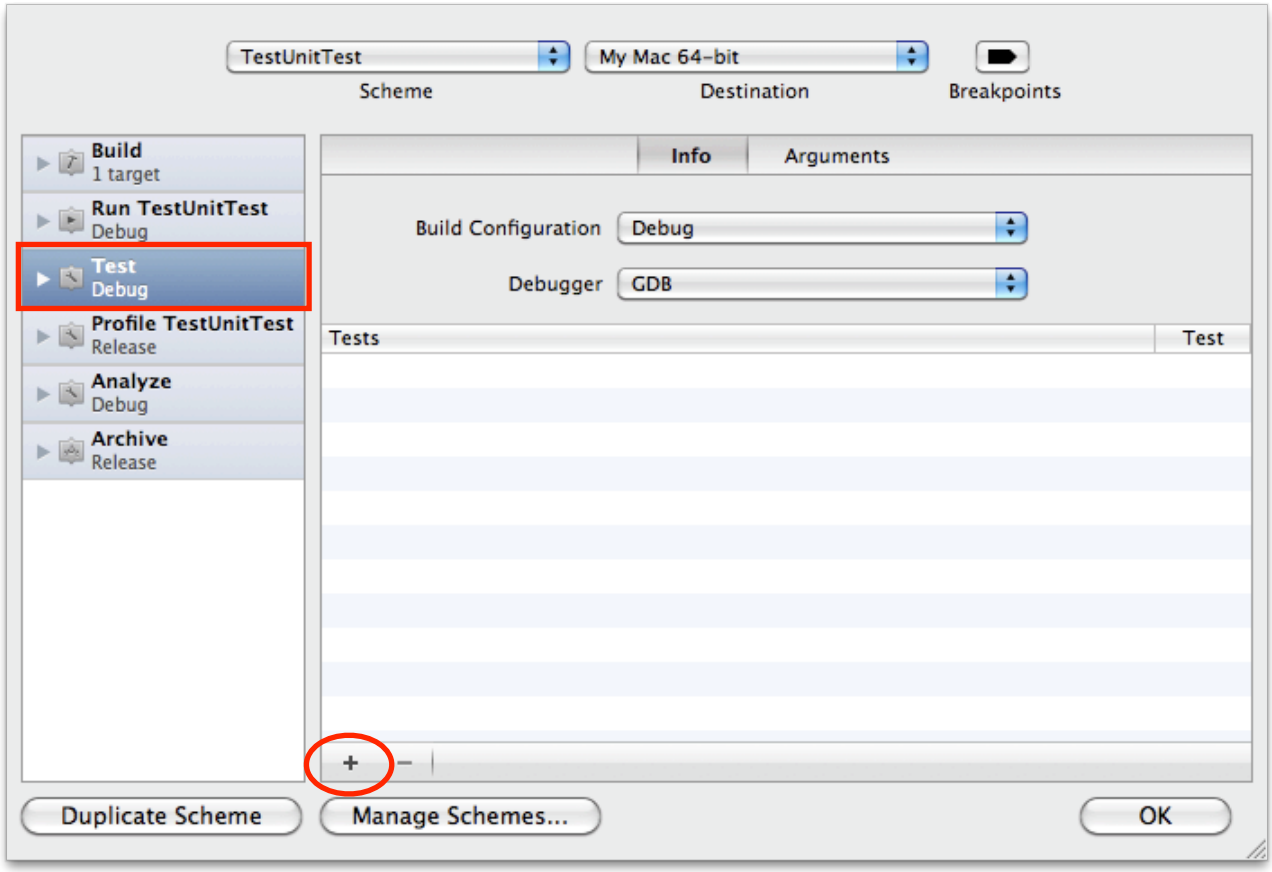


Step 8. 在上面的 Scheme 點一下, 選擇 Edit Scheme

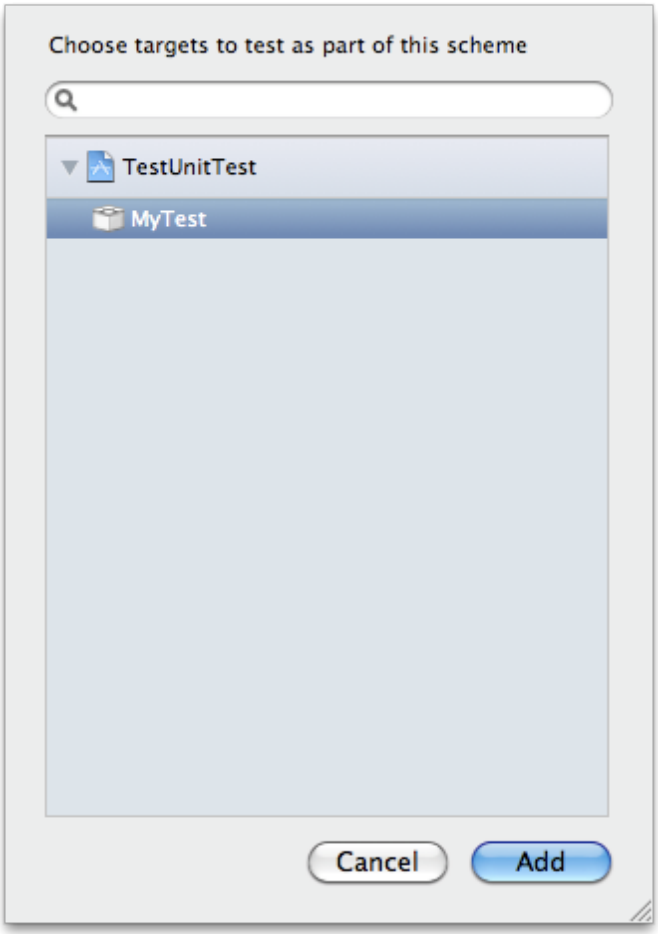




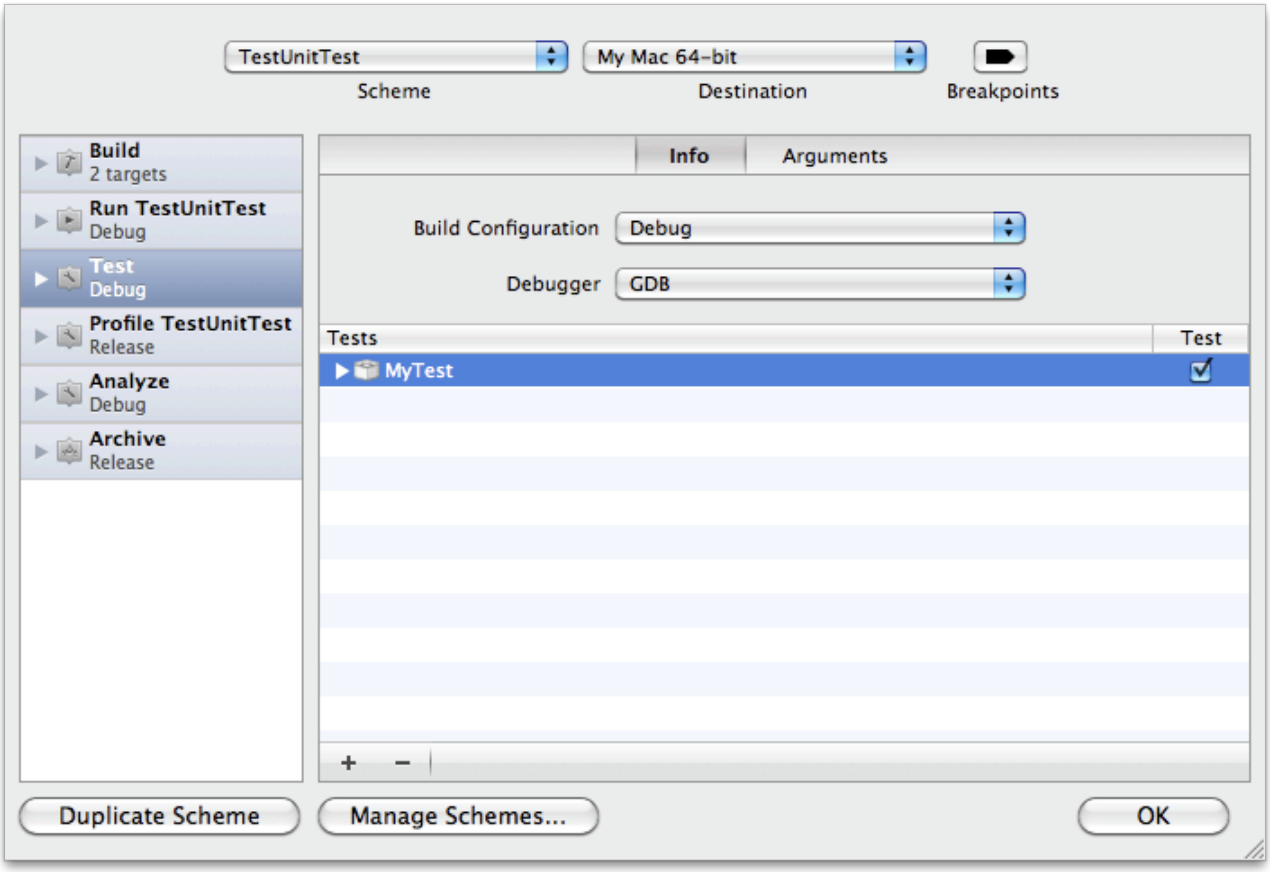
選擇 Test (Debug) 這個 Tab, 點選下面的加號



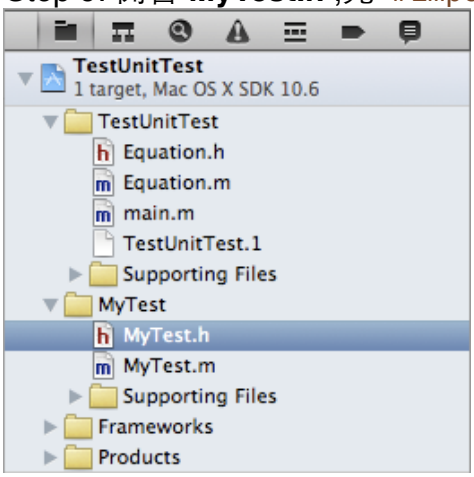
點選我們剛剛增加的 Target **MyTest**, 點 Add



MyTest就出現在 Tests 欄中了, OK



Step 9. 開啓 **MyTest.h** ,先 `#import "Equation.h"` ,再新增一個myEquation的物件




```
#import <SenTestingKit/SenTestingKit.h>
#import "Equation.h"

@interface MyTest : SenTestCase {
@private
    Equation *myEquation;
}

@end
```

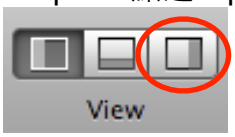
Step 10. 然後在 MyTest.m 裡,實作我們的測試程式
先做SetUp的動作,在裡面把myEquation定址並實體化(new), 然後設定number的初始值為10

在testExample裡將原來的 STFail mark掉, 並設定測試條件, 我們預想10的三次方應該為1000, 若測試fail則印出後面的String, 其中含有目前計算指數運算的回傳值

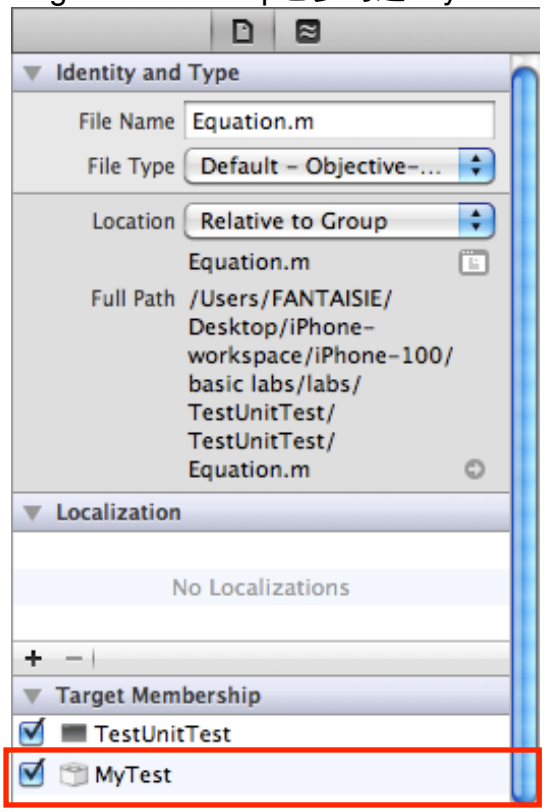
最後在tesrDown完成測試並將myEquation release,避免memory leak.

```
#import "MyTest.h"


@implementation MyTest

- (void)setUp
{
    [super setUp];

    // Set-up code here.
    myEquation = [Equation new];
    [myEquation setNumber:10];
    NSLog(@"start point");
}

- (void)tearDown
{
    // Tear-down code here.
    [myEquation release];
    NSLog(@"end point");

    [super tearDown];
}

- (void)testExample
{
    //STFail(@"Unit tests are not implemented yet in MyTest");
    STAssertEquals([myEquation getResult:3], 1000,
                   [@"Should be 1000. Now it is "
stringByAppendingString:
                    [NSString stringWithFormat:@"%d", [myEquation
getResult:3]]]);
}

@end
```

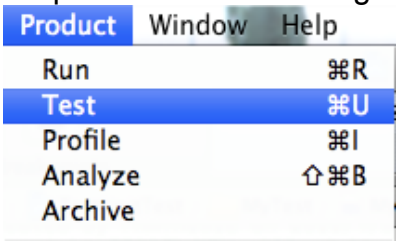
Step 11. 點選Equation.m, 並開啓右邊File Inspector的View



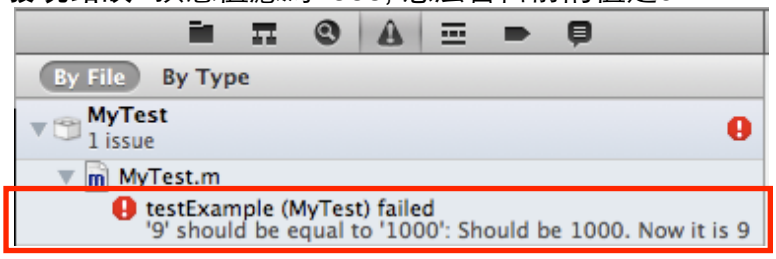
Target Membership也多勾選 MyTest



Step 12. Build For Testing and Test (⌘+U)



發現錯誤! 預想值應為**1000**, 怎麼會目前的值是**9**?




```
- (void)testExample
{
    //STFail(@"Unit tests are not implemented yet in MyTest");
    STAssertEquals([myEquation getResult:3], 1000,
                   [@"Should be 1000. Now it is " stringByAppendingString:
                    [NSString stringWithFormat:@"%d", [myEquation getResult:3
                        ]]]);
}

@end
```

'9' should be equal to '1000': Should be 1000. Now it is 9

Step 13. 原來是在Equation.m裡面,我們的 getResult:(int)exponent 這個method回傳(number) ^ exponent 其實是代表number和exponent兩個變數做XOR而不是number的exponent次方.

```
#import "Equation.h"

@implementation Equation
@synthesize number;

-(int) getResult:(int)exponent{
    return (number) ^ exponent;
}

@end
```

Step 14. 修改 Equation.m裡的 getResult:(int)exponent 這個method的回傳值為pow((number), exponent) 為C做指數運算的function, 語意即為我們想要的number的exponent次方.(註: 其實此回傳值為double,但在此不多加敘述研究)

```
#import "Equation.h"


@implementation Equation
@synthesize number;

-(int) getResult:(int)exponent{
    return pow((number), exponent);
}

@end
```

Step 15. 再次 Build For Testing and Test (⌘+U)
測試成功無Error, 意思就是計算指數運算的method符合我們想要的語意邏輯,接下來就可以繼續完成我們的application了!