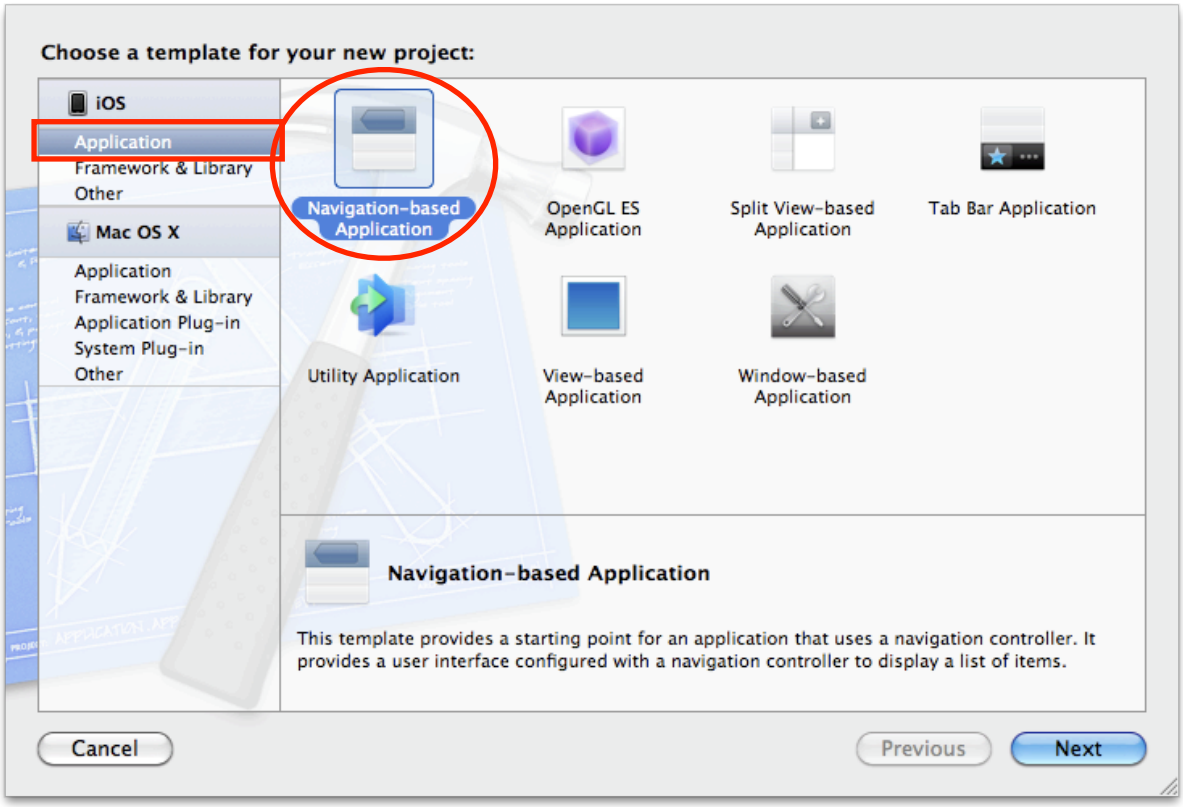
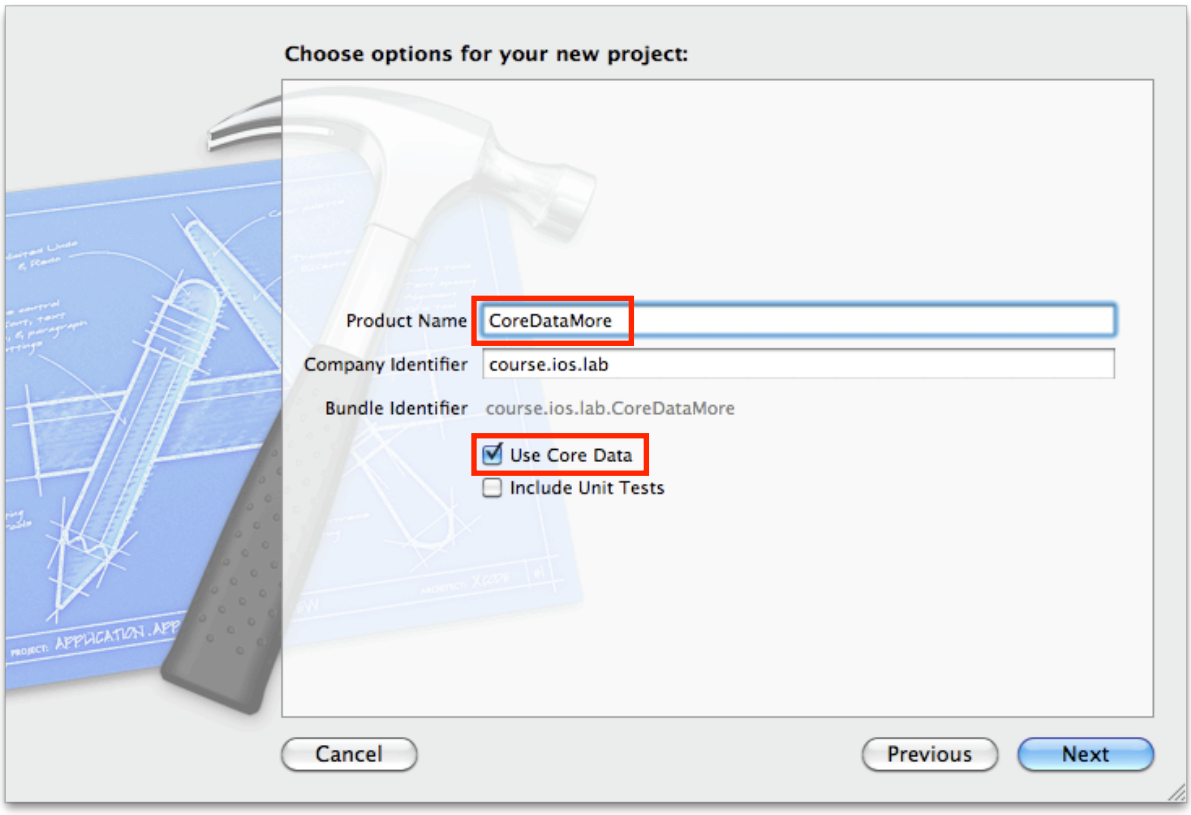
Lab CoreDataMore

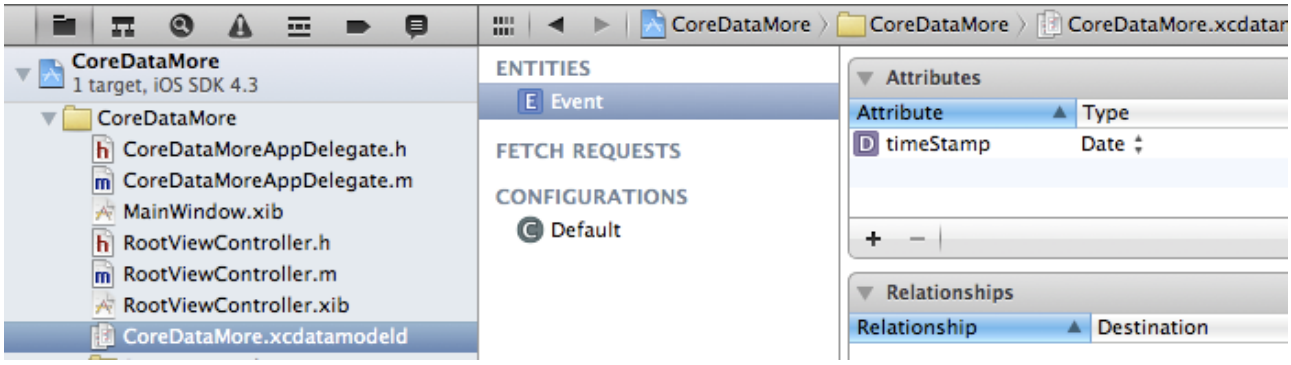
Step 1. 在 File>New>New File 開啓一個新的project, 選擇 Navigation-based application



將project命名為 **CoreDataMore** , 記得勾選 **Use Core Data**

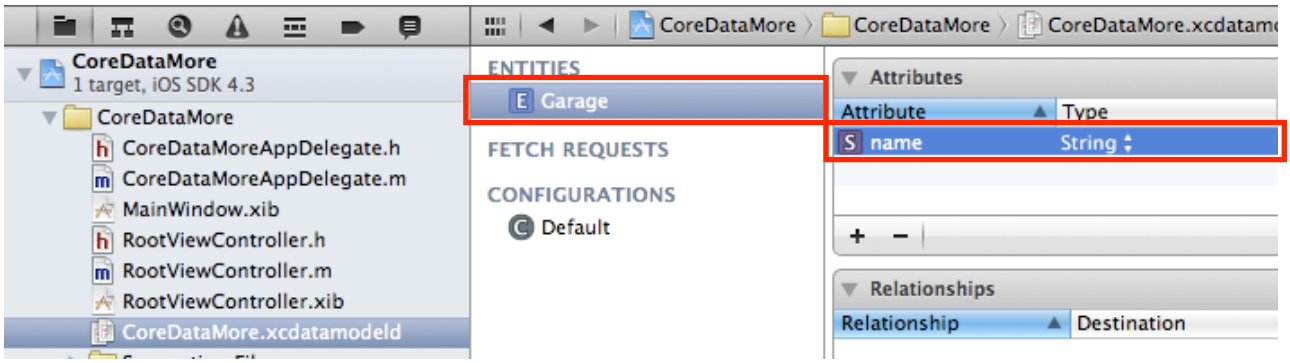


Step 2. 在 Project Navigator 裡開啓 CoreDataMore.xcdata.modeld

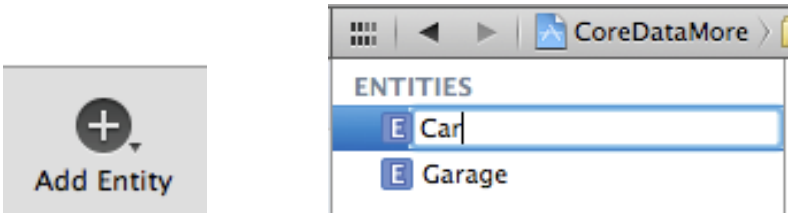


在此我們要做兩個Entity,主要是模擬我們可以在database裡增加多個Garage的Entity,然後每個Garage裡面又有多個Car,每個Car都有相對的vendor和 price

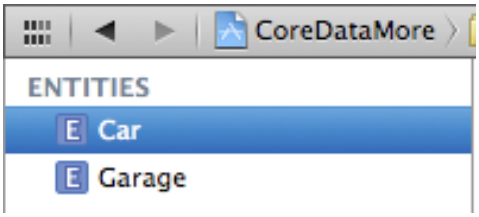
首先點選原有的Entity將名稱更改為 **Garage**, 然後將Attribute名稱改為 **name**, Type改為 **String**



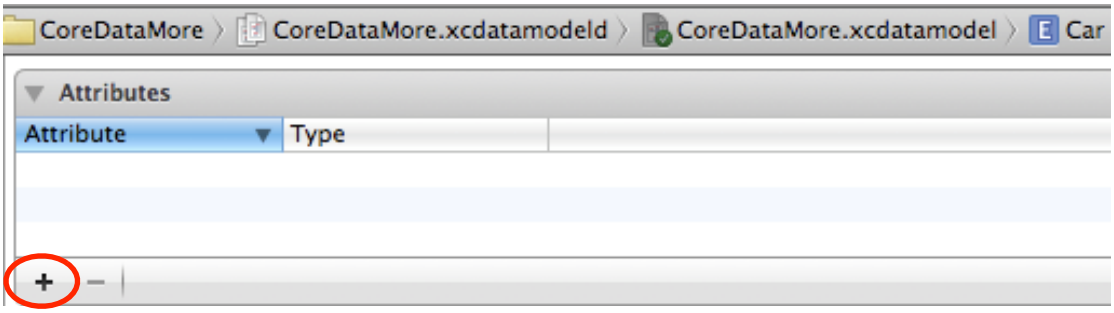
之後在增加另外一個Entity; 在中間視窗最下面點選 Add Entity, 並將新的 Entity 命名為 **Car**



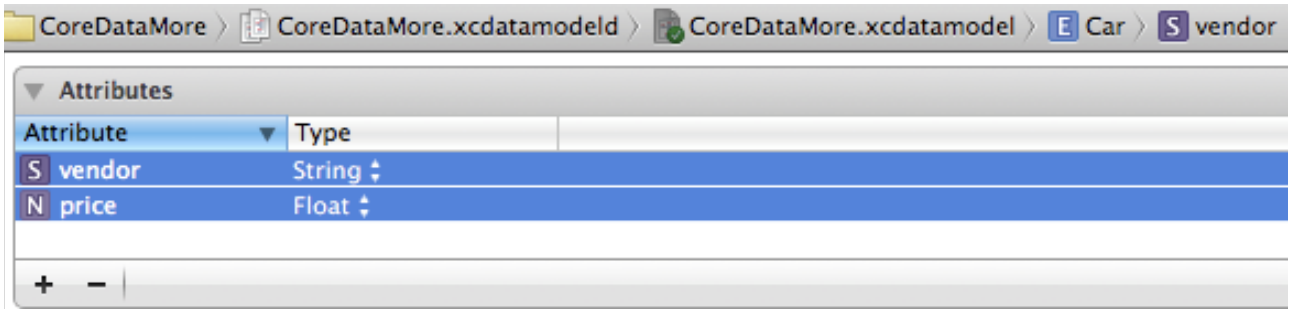
Step 3. 點選剛剛新增的Car這個Entity



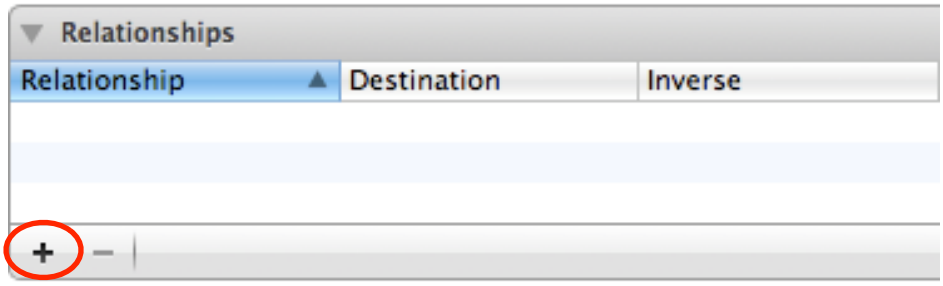
在Attribute 這欄點下面的加號來增加2個 attribute



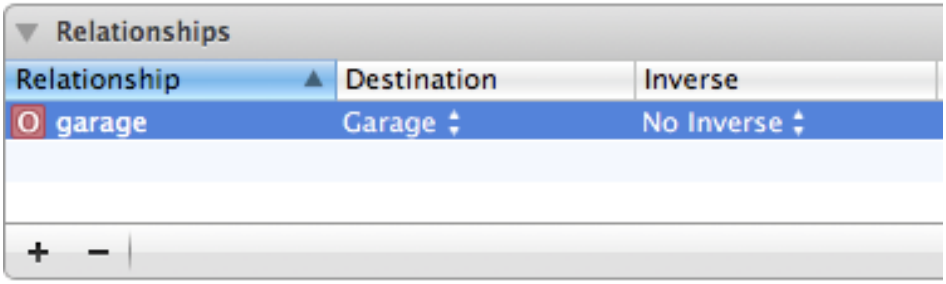
分別命名為 **price**, **vendor**, 並將 price的 type選為 **float**, 而將 vendor 的 type選為 **String**



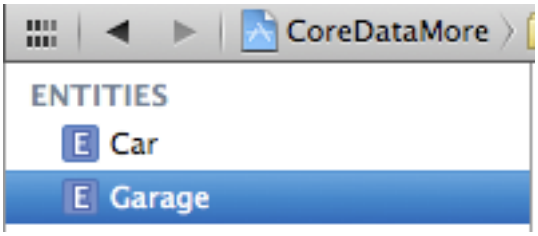
同樣點選 Car 這個Entitiy的Relationships下面的+號來增加Relationship



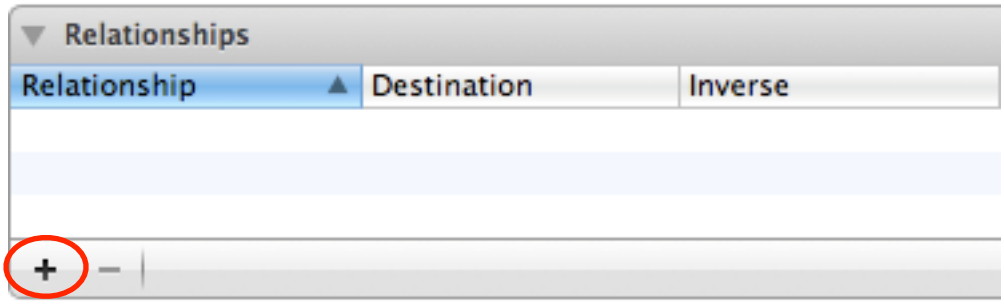
增加一個Relationship叫做 **garage** , Destination選擇另一個Entity **Garage**, Inverse等到我們下一步回到Garage這個Entity同樣設定完和Car這個Entity的Relationship之後再設定



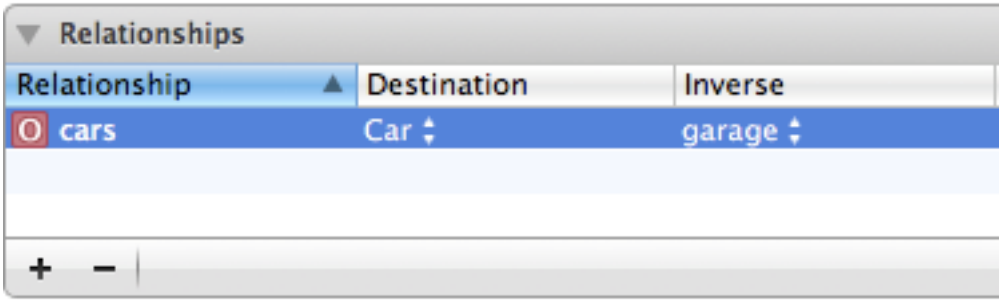
Step 4. 點選 Garage 這個Entity



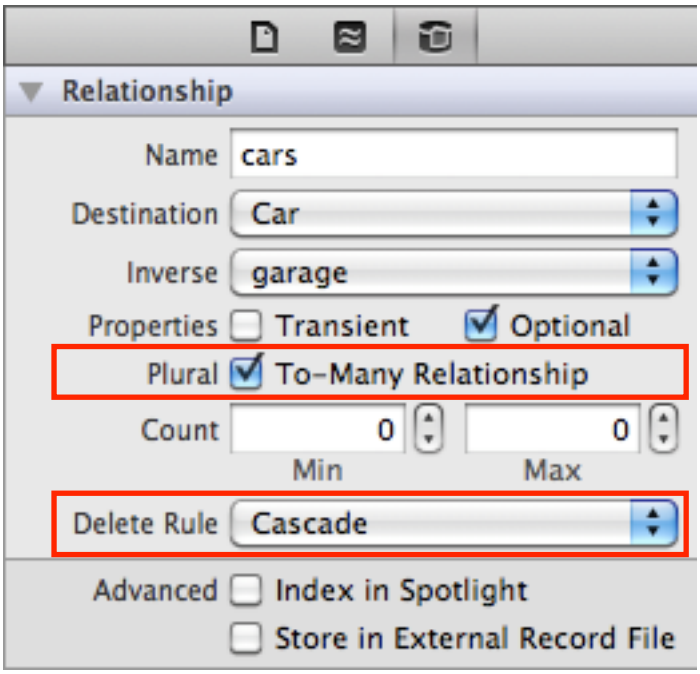
點選Relationships下面的+號來增加Relationship



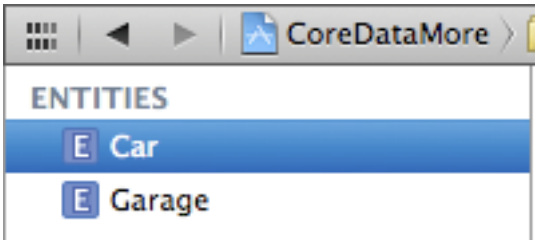
增加一個Relationship叫做 **cars** , Destination選擇另一個Entity **Car**, Inverse設定為Car裡面已經設定的relationship **garage**



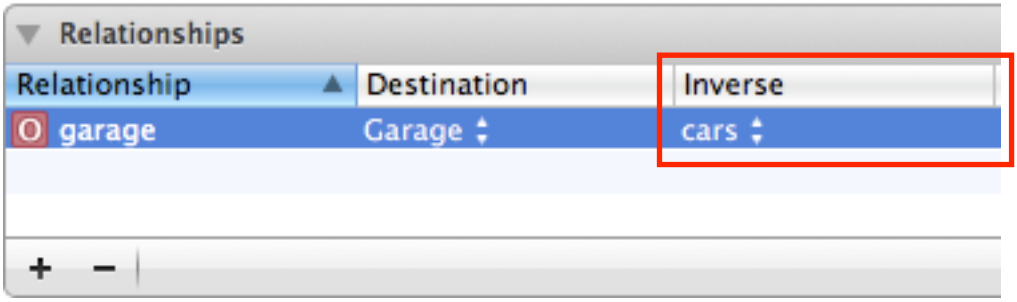
並開啓右邊視窗來configure **cars** 這個 Relationship



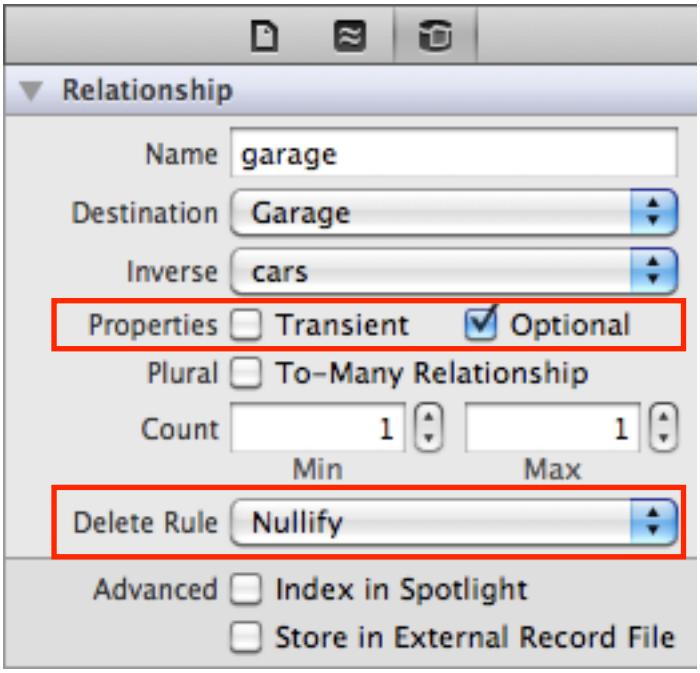
Step 5. 點選 Car 去回到 Car 這個Entity



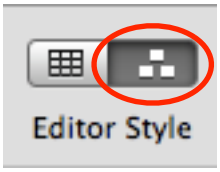
在Relationships這欄裡面將剛剛新增的garage這個Relationship的Inverse設為 **cars**



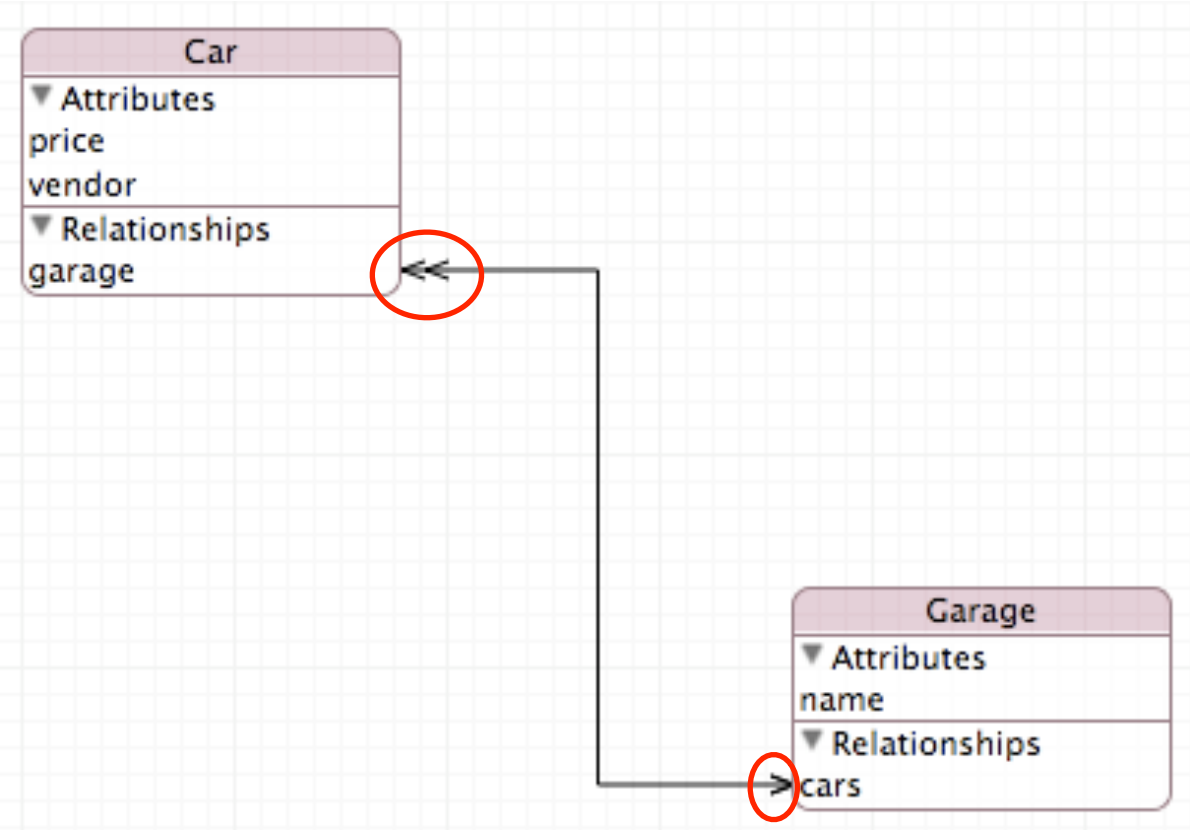
並也開啓右邊視窗來configure **garage** 這個 Relationship



Step 6. 同樣在 CoreDataMore.xcdata.model 視窗最右下角點選Editor Style右邊的Button

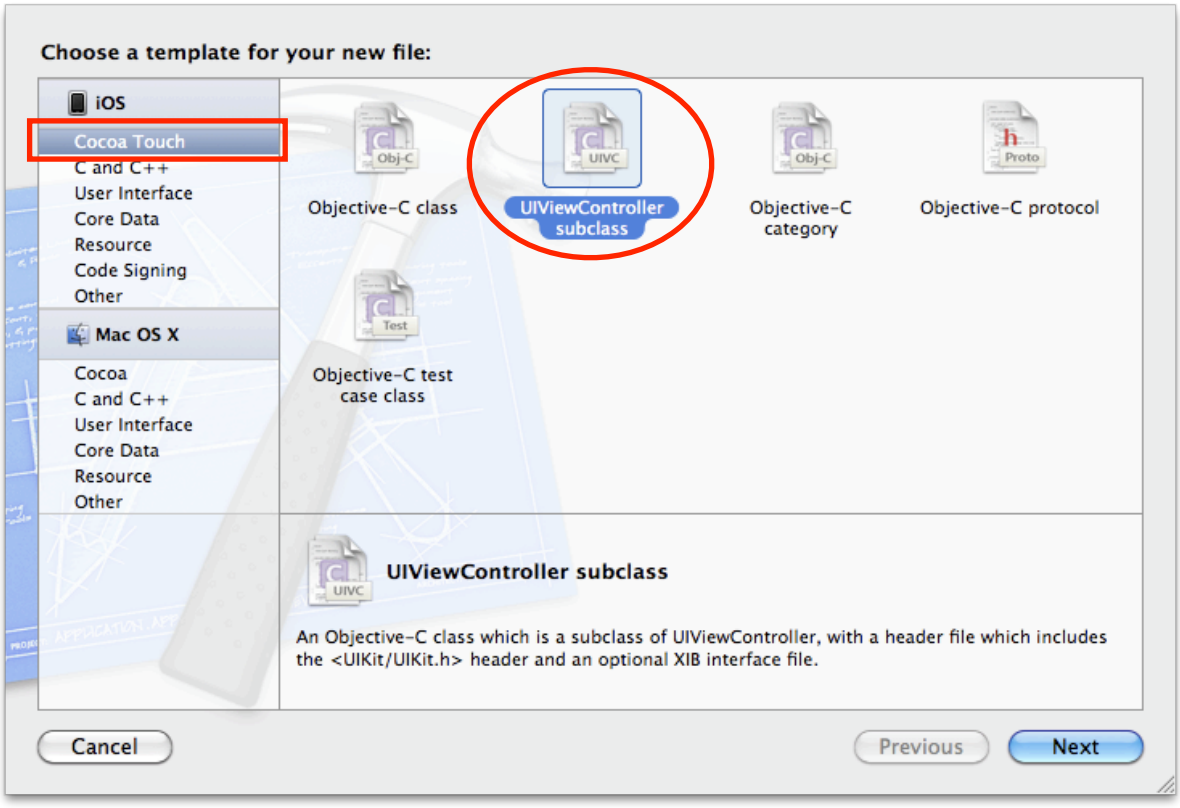


則可以看到現在兩個Entity的Attributes和Relationships的關聯性應該為下圖,注意兩邊Relationships的Inverse都有設定才會**同條線且箭頭才會為雙向且對Car方為多箭頭**

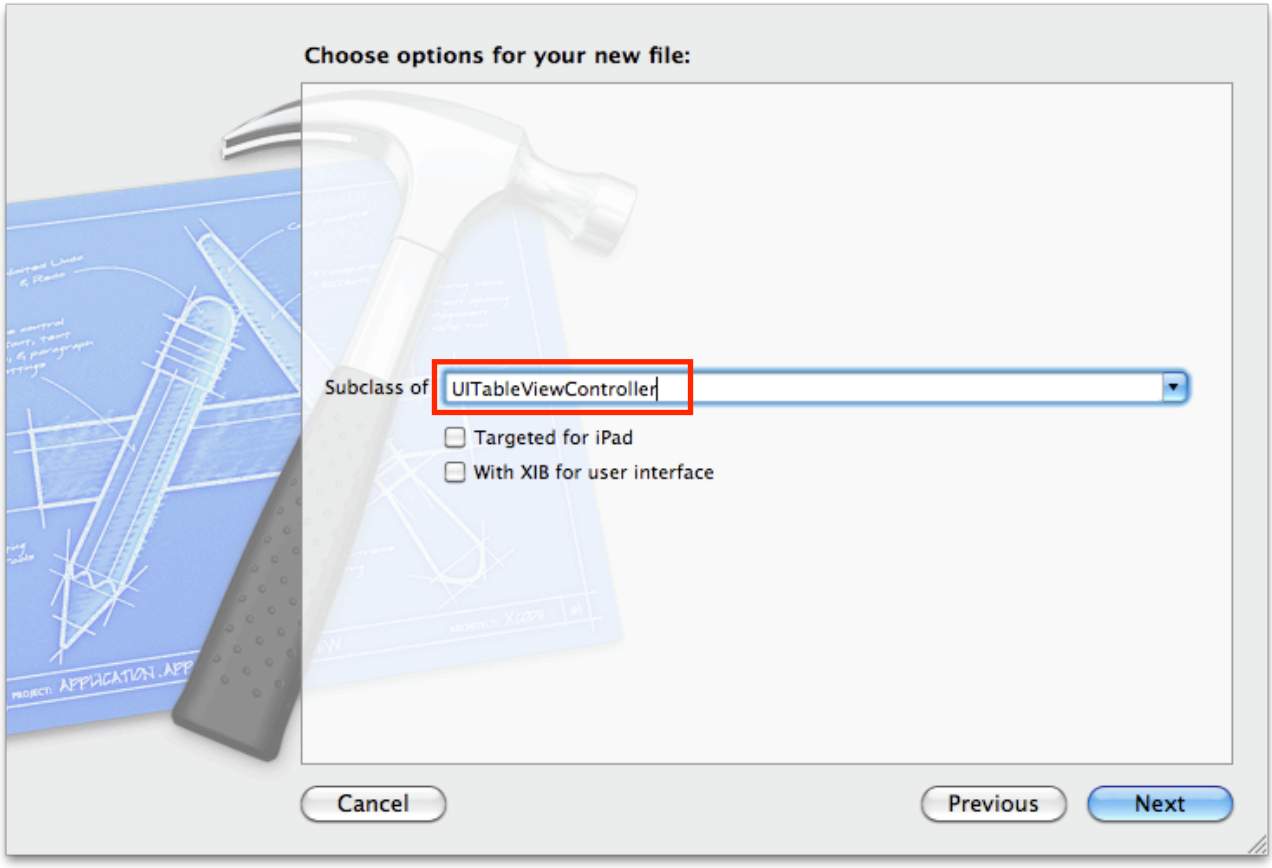


Step 7. 在左邊Project Navigator視窗裡在CoreDataMore資料夾上點右鍵,選擇New File,

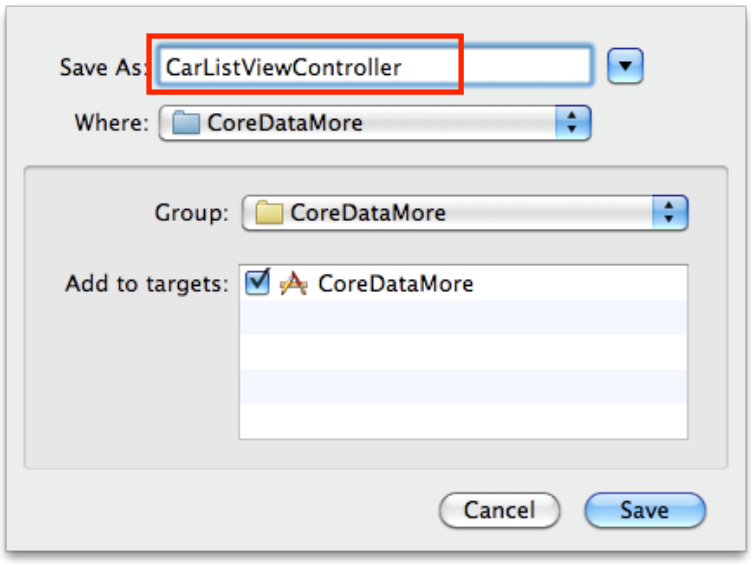
並點選iOS裡的Cocoa Touch目錄裡的 UIViewController subclass



注意!選擇 Subclass of **UITableViewController**



命名為 CarListViewController ,存檔



Step 8. 開啓 CarListViewController.h, 先#import "RootViewController.h",再設定兩個property包括一個RootViewController的object和一個NSManagedObject的object **garage**, 以及兩個method

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

@interface CarListViewController : UITableViewController {

}
@property (retain) RootViewController * rootViewController;
@property (nonatomic, retain) NSManagedObject *garage;
- (void) addCar;
- (NSArray *)sortCars;
@end
```

Step 9. 開啓 RootViewController.h, 加入一個method將context儲存SQLite的File.

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

#import <CoreData/CoreData.h>

@interface RootViewController : UITableViewController
<NSFetchedResultsControllerDelegate> {

}

@property (nonatomic, retain) NSFetchedResultsController
*fetchedResultsController;
@property (nonatomic, retain) NSManagedObjectContext
*managedObjectContext;

- (void)saveContext;

@end
```

Step 10. 開啓 RootViewController.m, 先#import "CarListViewController.h" , 找到

- (void)viewDidLoad{} 去設定此頁面的title

```
- (void)viewDidLoad
{
    [super viewDidLoad];
    self.title = @"Garages";
    // Set up the edit and add buttons.
    self.navigationItem.leftBarButtonItem = self.editButtonItem;

    UIBarButtonItem *addButton = [[UIBarButtonItem alloc]
initWithBarButtonSystemItem:UIBarButtonSystemItemAdd target:self
action:@selector(insertNewObject)];
    self.navigationItem.rightBarButtonItem = addButton;
    [addButton release];
}
```

Step 10. 然後找到 - (NSFetchedResultsController *)fetchedResultsController {} 這個method,修改下面紅框內的程式,主要是將原先產生的Entity從Event改成**Garage**, Attribute從timeStamp改成**name**

```
- (NSFetchedResultsController *)fetchedResultsController
{
    if (__fetchedResultsController != nil)
    {
        return __fetchedResultsController;
    }

    /*
     Set up the fetched results controller.
    */
    // Create the fetch request for the entity.
    NSFetchRequest *fetchRequest = [[NSFetchRequest alloc] init];
    // Edit the entity name as appropriate.
    NSEntityDescription *entity = [NSEntityDescription entityForName:@"Garage"
inManagedObjectContext:self.managedObjectContext];
    [fetchRequest setEntity:entity];

    // Set the batch size to a suitable number.
    [fetchRequest setFetchBatchSize:20];

    // Edit the sort key as appropriate.
    NSSortDescriptor *sortDescriptor = [[NSSortDescriptor alloc]
initWithKey:@"name" ascending:NO];
    NSArray *sortDescriptors = [[NSArray alloc] initWithObjects:sortDescriptor,
nil];

.......
```

Step 11. 找到 - (void)configureCell:(UITableViewCell *)cell atIndexPath: (NSIndexPath *)indexPath{} 將原先產生的Attribute從timeStamp改成**name**

```
- (void)configureCell:(UITableViewCell *)cell atIndexPath:(NSIndexPath *)
indexPath
{
    NSManagedObject *managedObject = [self.fetchedResultsController
objectAtIndexPath:indexPath];
    cell.textLabel.text = [[managedObject valueForKey:@"name"] description];
}
```

Step 12. 找到 - (void)insertNewObject{}將更改下列紅框內程式,主要是將原先產生的Attribute從timeStamp改成**name**,並將name的value改成每一次新增一個Garage的名稱

```
- (void)insertNewObject
{
    // Create a new instance of the entity managed by the fetched results
controller.
    NSManagedObjectContext *context = [self.fetchedResultsController
managedObjectContext];
    NSEntityDescription *entity = [[self.fetchedResultsController fetchRequest]
entity];
    NSManagedObject *newManagedObject = [NSEntityDescription
insertNewObjectForEntityForName:[entity name] inManagedObjectContext:context];

    // If appropriate, configure the new managed object.
    // Normally you should use accessor methods, but using KVC here avoids the
need to add a custom class to the template.
    id <NSFetchedResultsSectionInfo> sectionInfo =
[[self.fetchedResultsController sections] objectAtIndex:0];
    NSString * tempName = [NSString stringWithFormat:@"Garage %d",
[sectionInfo numberOfObjects]];
    [newManagedObject setValue:tempName forKey:@"name"];

.......
```

Step 13. 找到 (void)configureCell:(UITableViewCell *)cell atIndexPath:(NSIndexPath *)indexPath{} 加入一行讓每個row的cell都有一個DetailDisclosureButton

```
- (void)configureCell:(UITableViewCell *)cell atIndexPath:(NSIndexPath
  *)indexPath
{
    NSManagedObject *managedObject = [self.fetchedResultsController
objectAtIndexPath:indexPath];
    cell.textLabel.text = [[managedObject valueForKey:@"name"]
description];
    cell.accessoryType = UITableViewCellAccessoryDetailDisclosureButton;
}
```

Step 14. 同樣在 RootViewController.m 在 @implementation RootViewController 和 @end 程式之間加入一個 Override Method - (void)tableView:(UITableView *)tableView accessoryButtonTappedForRowWithIndexPath:(NSIndexPath *)indexPath{}

```
- (void)tableView:(UITableView *)tableView
accessoryButtonTappedForRowWithIndexPath:(NSIndexPath *)indexPath{
    NSLog(@"tapped %d", indexPath.row);
    CarListViewController * carList = [[CarListViewController alloc]
initWithStyle:UITableViewStylePlain];
    carList.rootViewController = self;
    carList.garage = [self.fetchedResultsController
objectAtIndexPath:indexPath];
    [self.navigationController pushViewController:carList animated:YES];
    [carList release];
}
```

Step 15. 同樣在 RootViewController.m 加入我們新增的將context儲存SQLite的File的Method – (void)saveContext {}

```
- (void)saveContext {
    NSManagedObjectContext *context = [self.fetchedResultsController
managedObjectContext];
    NSError *error = nil;
    if (![context save:&error]) {
        NSLog(@"Unresolved error %@, %@", error, [error userInfo]);
        abort();
    }
}
```

Step 16. 開啓 CarListViewController.m , 先將兩個已作@property的變數做@synthesize

```
#import "CarListViewController.h"

@implementation CarListViewController
@synthesize rootViewController, garage;
```

Step 17. 找到 – (void)viewDidLoad{} 將內部改成下列程式,去設定此頁面的title,以及新增一個addButton以及其對應的method **addCar**

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

    self.title = @"Cars";
    UIBarButtonItem *addButton = [[UIBarButtonItem alloc]
initWithBarButtonSystemItem:UIBarButtonSystemItemAdd target:self
action:@selector(addCar)];
    self.navigationItem.rightBarButtonItem = addButton;
    [addButton release];
}
```

Step 18. 同樣在 CarListViewController.m 實作 addCar這個Method

```
- (void) addCar {
    NSLog(@"added car");
    NSArray * carVendors = [[NSArray alloc] initWithObjects:@"Toyota",
@"Honda", @"Nissan", @"BMW", @"VW", nil];
    NSManagedObjectContext *context =
[rootViewController.fetchedResultsController managedObjectContext];
    NSManagedObject *car = [NSEntityDescription
insertNewObjectForEntityForName:@"Car" inManagedObjectContext:context];
    NSString * carVendor = [NSString stringWithFormat:@"%@", [carVendors
objectAtIndex:(arc4random()%5)]];
    [car setValue:carVendor forKey:@"vendor"];
    [car setValue:[NSNumber numberWithFloat:((arc4random()%10)*1000)]
forKey:@"price"];
    [car setValue:garage forKey:@"garage"];
```

```
    // Save the context.
    [rootViewController saveContext];
    [self.tableView reloadData];
}
```

Step 19. 找到 - (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView{} 和 - (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section{} 兩個method, 將兩行會造成Warning的程式Mark掉(在前面加入兩斜線), 並重寫return值

```
#pragma mark - Table view data source

- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView
{
//#warning Potentially incomplete method implementation.
    // Return the number of sections.
    return 1;
}

- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:
(NSInteger)section
{
//#warning Incomplete method implementation.
    // Return the number of rows in the section.
    return [(NSSet *)[garage valueForKey:@"cars"] count];
}
```

Step 20. 找到 - (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath{} 加入下面紅框內程式

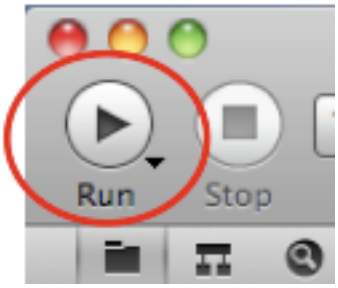
```
- (UITableViewCell *)tableView:(UITableView *)tableView
cellForRowAtIndexPath:(NSIndexPath *)indexPath
{
    static NSString *CellIdentifier = @"Cell";

    UITableViewCell *cell = [tableView
dequeueReusableCellWithIdentifier:CellIdentifier];
    if (cell == nil) {
        cell = [[[UITableViewCell alloc]
initWithStyle:UITableViewCellStyleDefault
reuseIdentifier:CellIdentifier] autorelease];
    }

    // Configure the cell...
    NSManagedObject *car = [[self sortCars]
objectAtIndex:indexPath.row];
    cell.textLabel.text = [NSString stringWithFormat:@"vendor: %@
price: %@", [[car valueForKey:@"vendor"] description], [[car
valueForKey:@"price"] description]];
    return cell;
}
```

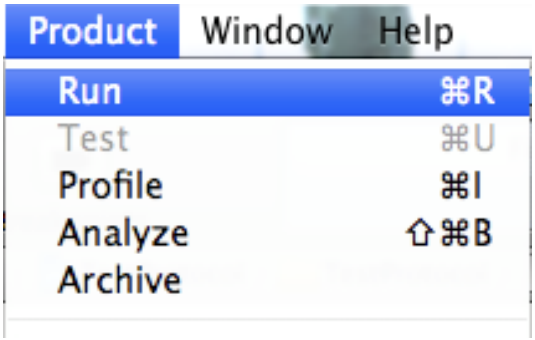
Step 21. 同樣在 CarListViewController.m 實作 sortCars 這個Method去依照price的大小來排列Car的順序(也可改為以vendor字母排序)

```
- (NSArray *)sortCars {
    NSSortDescriptor *sortLastNameDescriptor = [[[NSSortDescriptor
alloc] initWithKey:@"price" ascending:YES] autorelease];
    NSArray *sortDescriptors = [NSArray
arrayWithObjects:sortLastNameDescriptor, nil];
    return [[(NSSet *)[garage valueForKey:@"cars"] allObjects]
sortedArrayUsingDescriptors:sortDescriptors];
}
```

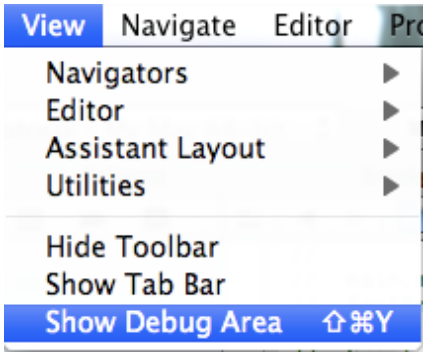
Step 22. Run (⌘+R)
在Xcode主頁左上角按下Run, 或是在Product > Run, 即開始Build code並執行

(注意若有改掉Entity或是attribute名稱或type時, 重新執行只會蓋掉binary執行檔,不會蓋掉以存成SQLite file的table, 所以若發生問題或當掉的話請在Simulator螢幕上Application的Icon長按,等待出現"X"字樣時將其Delete,再重新Run)





之後會自動開啓console, 沒開啓的話在右上角View點擊中間的Button, 或是選擇View > Show Debug Area 來開啓



執行後出現Garages的頁面,點右邊的+號加入新的Garage,或透過Edit刪除已存在的Garage

點下任意一個row的Garage就可進入Cars的頁面去新增Car, 新增是使用arc4random的方式去產生vendor和price,然後以price的大小來做遞增排列