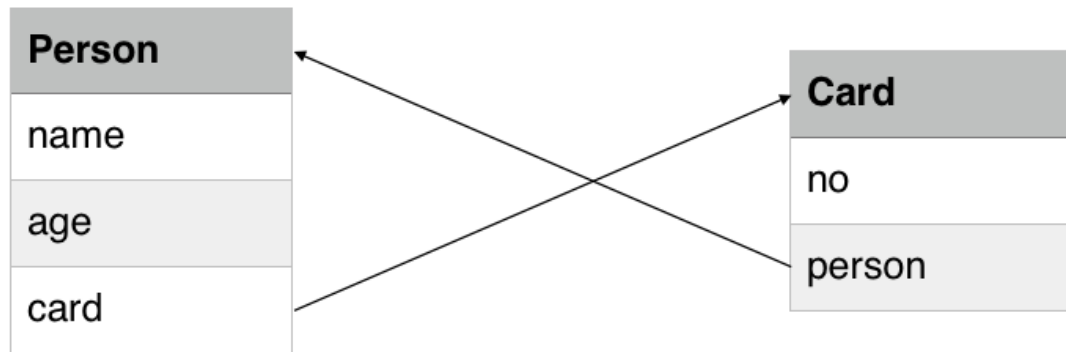


使用Core Data

在Core Data，需要进行映射的对象称为实体(entity)，而且需要使用Core Data的模型文件来描述app中的所有实体和实体属性，Core data的使用从创建模型开始。这里以Person(人)和Card(身份证)2个模型为例子，先看看模型之间的关联关系：



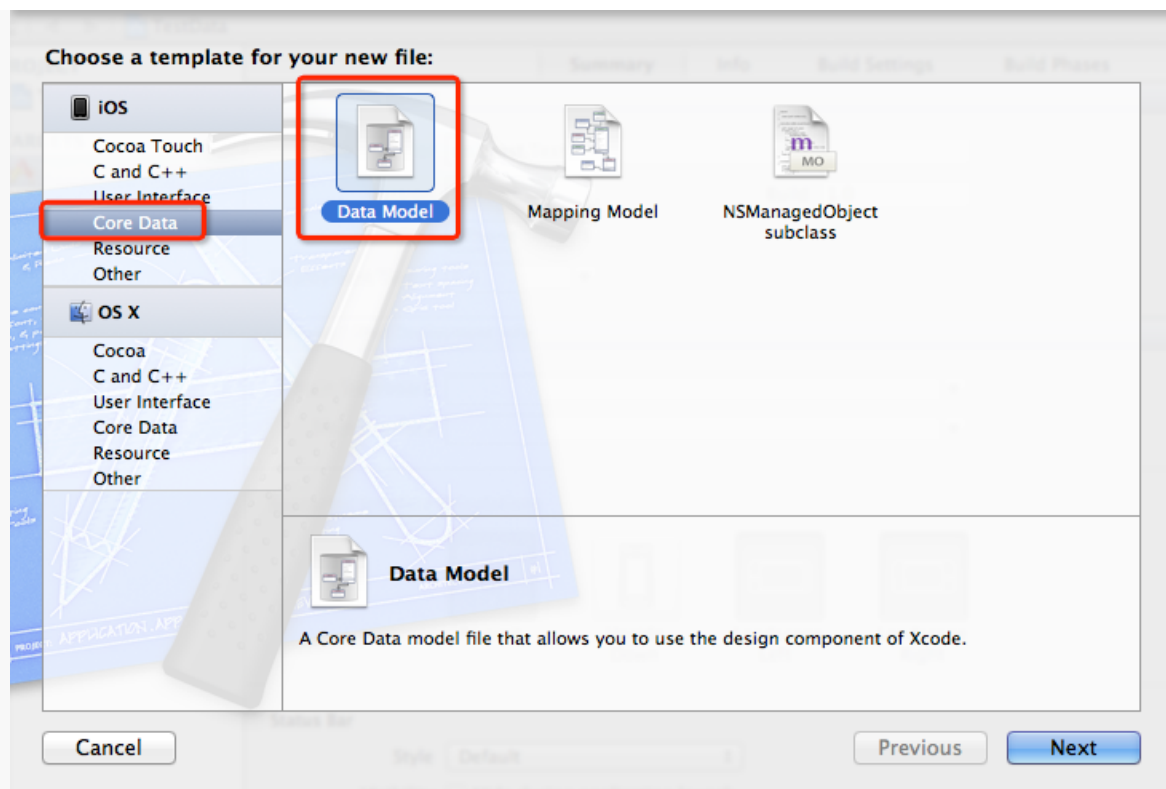
Person实体中有：name（姓名）、age（年龄）、card（身份证）三个属性 Card实体中有：no（号码）、person（人）两个属性

对应的数据库该怎么建立？

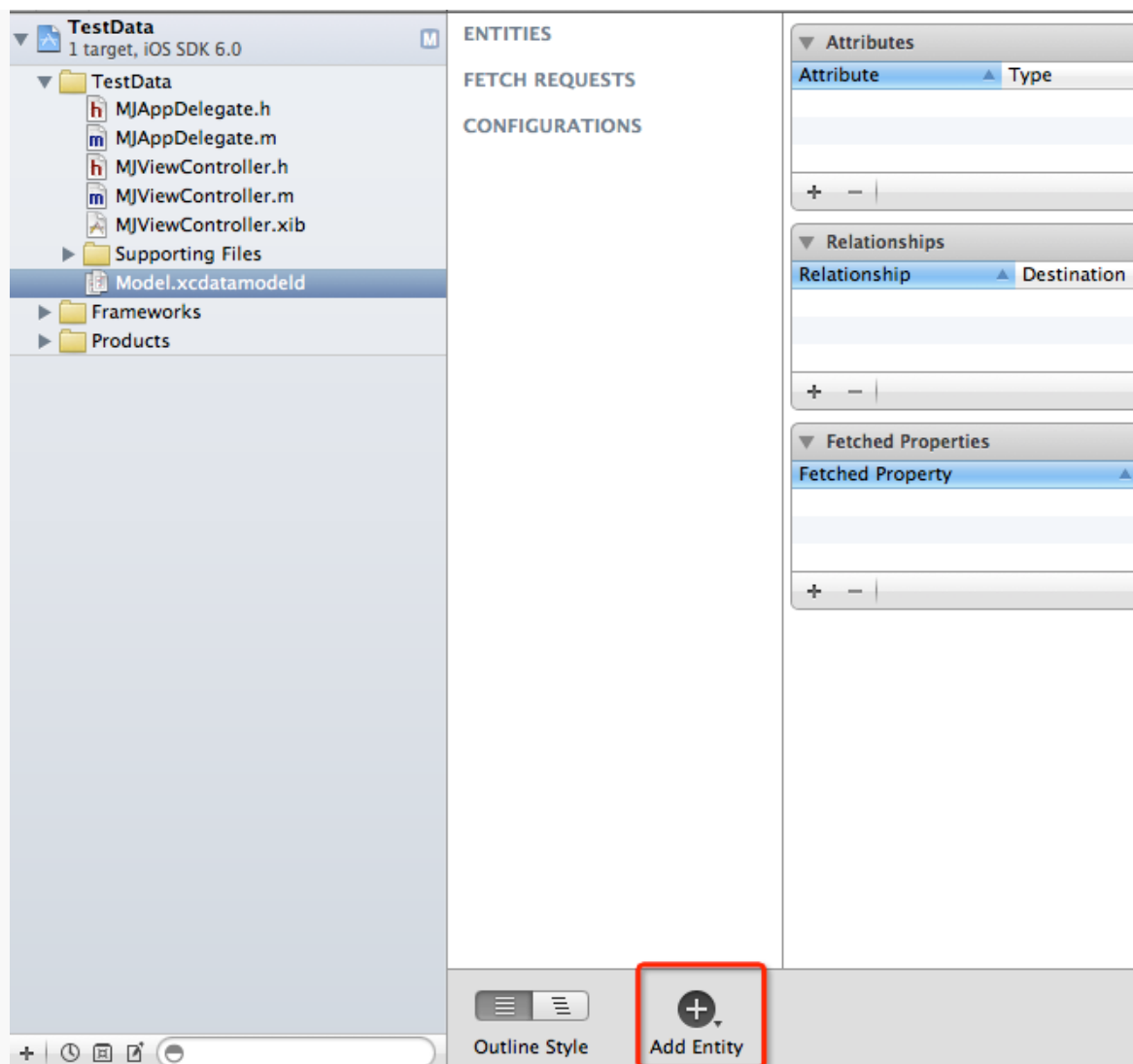
模型文件

使用Core Data 从建立模型开始，其它模块的工作根据建立的模型进行处理。接下来看看创建模型文件的过程：

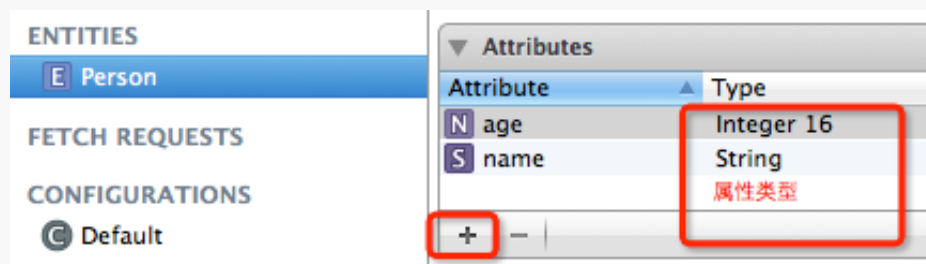
1.选择模板



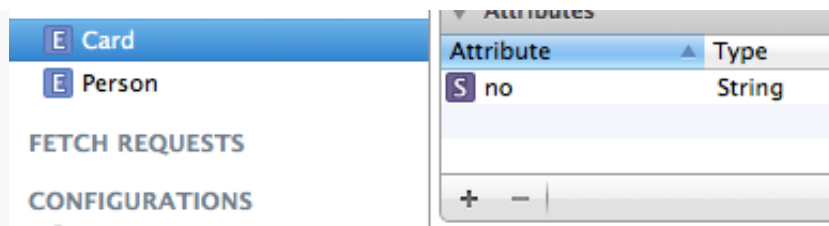
2. 添加实体



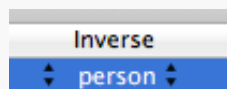
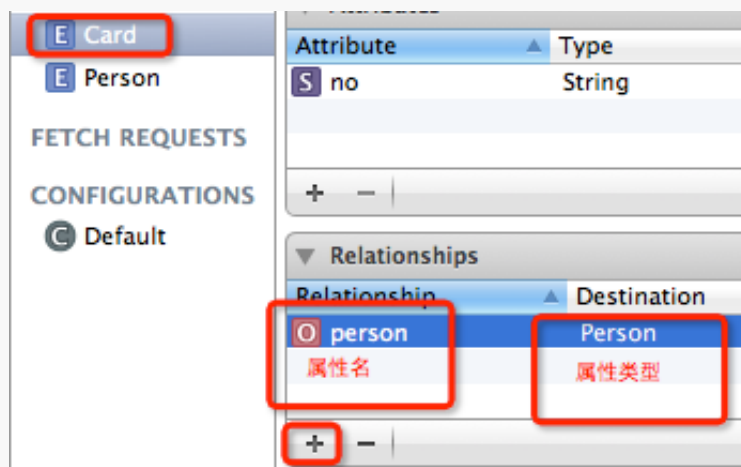
3. 添加Person的2个基本属性




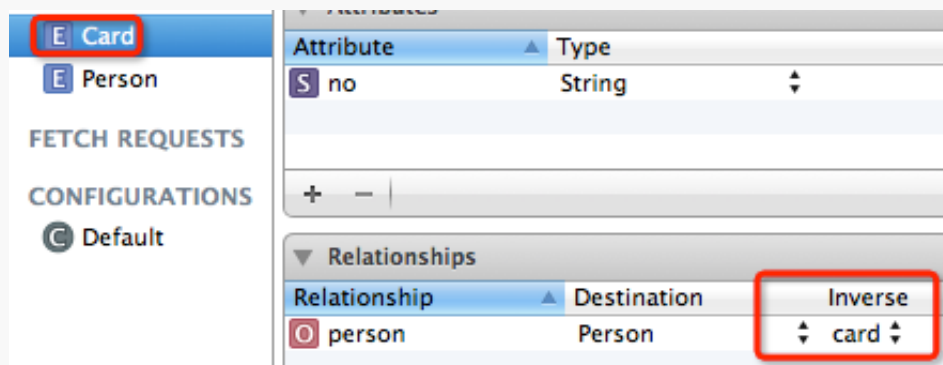
4. 添加Card的1个基本属性



5.建立Card和Person的关联关系

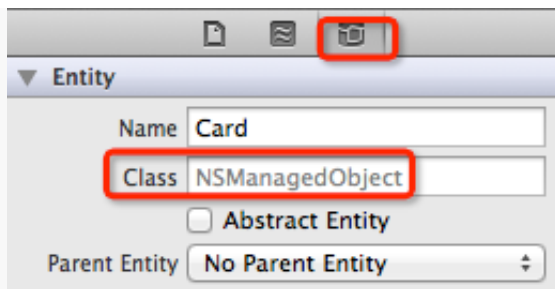


上图中的  表示Card中有个Person类型的人属性，目的就是建立Card跟Person之间的一对一关联关系(建议补上这一项)，在Person中加上Inverse属性后，你会发现Card中Inverse属性也自动补上了



了解NSManagedObject

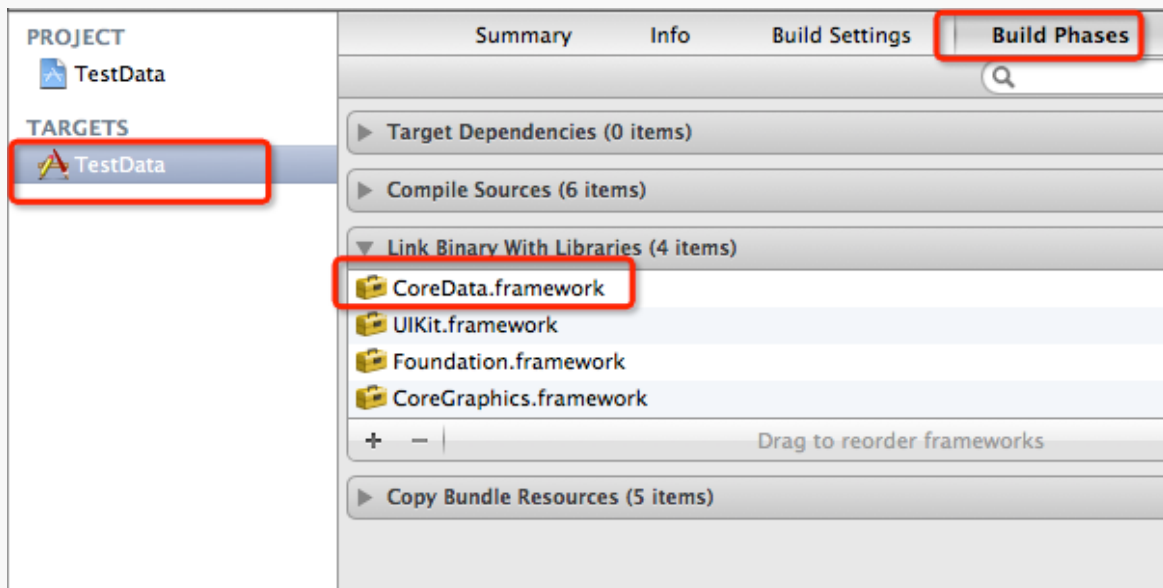
1.上面创建的模型，默认情况下都是NSManagedObject类型，对应的Core Data中的模型对象都是NSManagedObject类的对象



2.NSManagedObject的工作模式有点类似于NSDictionary对象，通过键-值对来存取所有的实体属性 1> setValue:forKey:存储属性值(属性名为key) 2> valueForKey:获取属性值(属性名为key)

代码实现

先添加CoreData.framework和导入主头文件<CoreData/CoreData.h>



1.搭建上下文环境

```

// 从应用程序包中加载模型文件
NSManagedObjectModel *model = [NSManagedObjectModel mergedModelFromBundles:nil];
// 传入模型对象, 初始化NSPersistentStoreCoordinator
NSPersistentStoreCoordinator *psc = [[NSPersistentStoreCoordinator alloc] initWithManagedObjectModel:model] autorelease];
// 构建SQLite数据库文件的路径
NSString *docs = [NSSearchPathForDirectoriesInDomains(NSDocumentDirectory, NSUserDomainMask, YES) lastObject];
NSURL *url = [NSURL fileURLWithPath:[docs stringByAppendingPathComponent:@"person.data"]];
// 添加持久化存储库, 这里使用SQLite作为存储库
NSError *error = nil;
NSPersistentStore *store = [psc addPersistentStoreWithType:NSSQLiteStoreType configuration:nil URL:url options:nil error:&error];
if (store == nil) {
    // 直接抛异常, 如果存储结构和模型不匹配
    [NSException raise:@"添加数据库错误" format:@"%@", [error localizedDescription]];
}
// 初始化上下文, 设置persistentStoreCoordinator属性
NSManagedObjectContext *context = [[NSManagedObjectContext alloc] init];
context.persistentStoreCoordinator = psc;

```

2. 添加数据到数据库

```

// 传入上下文，创建一个Person实体对象
NSManagedObject *person = [NSEntityDescription insertNewObjectForEntityForName:@"Person" inManagedObjectContext:context];
// 设置Person的简单属性
[person setValue:@"xhz" forKey:@"name"];
[person setValue:[NSNumber numberWithInt:27] forKey:@"age"];
// 传入上下文，创建一个Card实体对象
NSManagedObject *card = [NSEntityDescription insertNewObjectForEntityForName:@"Card" inManagedObjectContext:context];
[card setValue:@"4414241933432" forKey:@"no"];
// 设置Person和Card之间的关联关系
[person setValue:card forKey:@"card"];
// 利用上下文对象，将数据同步到持久化存储库
NSError *error = nil;
BOOL success = [context save:&error];
if (!success) {
    [NSException raise:@"访问数据库错误" format:@"%@", [error localizedDescription]];
}
// 如果是想做更新操作：只要在更改了实体对象的属性后调用[context save:&error]，就能将更改的数据同步到数据库

```

3.从数据库中查询数据

```

// 初始化一个查询请求
NSFetchRequest *request = [[[NSFetchRequest alloc] init] autorelease];
// 设置要查询的实体
request.entity = [NSEntityDescription entityWithName:@"Person" inManagedObjectContext:context];
// 设置排序（按照age降序）
NSSortDescriptor *sort = [NSSortDescriptor sortDescriptorWithKey:@"age" ascending:NO];
request.sortDescriptors = [NSArray arrayWithObject:sort];
// 设置条件过滤(搜索name中包含字符串"Itcast-1"的记录，注意：设置条件过滤时，数据库SQL语句中的%要用*来代替，所以%Itcast-1%应该写成*Itcast-1*)
NSPredicate *predicate = [NSPredicate predicateWithFormat:@"name like %@", @"*a*"];
request.predicate = predicate;
// 执行请求
NSError *error = nil;
NSArray *objs = [context executeFetchRequest:request error:&error];
if (error) {
    [NSException raise:@"查询错误" format:@"%@", [error localizedDescription]];
}
// 遍历数据
for (NSManagedObject *obj in objs) {
    NSLog(@"name=%@", [obj valueForKey:@"name"]);
}

```

注：Core Data不会根据实体中的关联关系立即获取相应的关联对象，比如通过Core Data取出Person实体时，并不会立即查询相关联的Card实体；当应用真的需要使用Card时，才会再次查询数据库，加载Card实体的信息。这个就是Core Data的延迟加载机制

4.删除数据库中的数据

```

// 传入需要删除的实体对象
[context deleteObject:managedObject];
// 将结果同步到数据库
NSError *error = nil;
[context save:&error];
if (error) {
    [NSException raise:@"删除错误" format:@"%@", [error localizedDescription]];
}

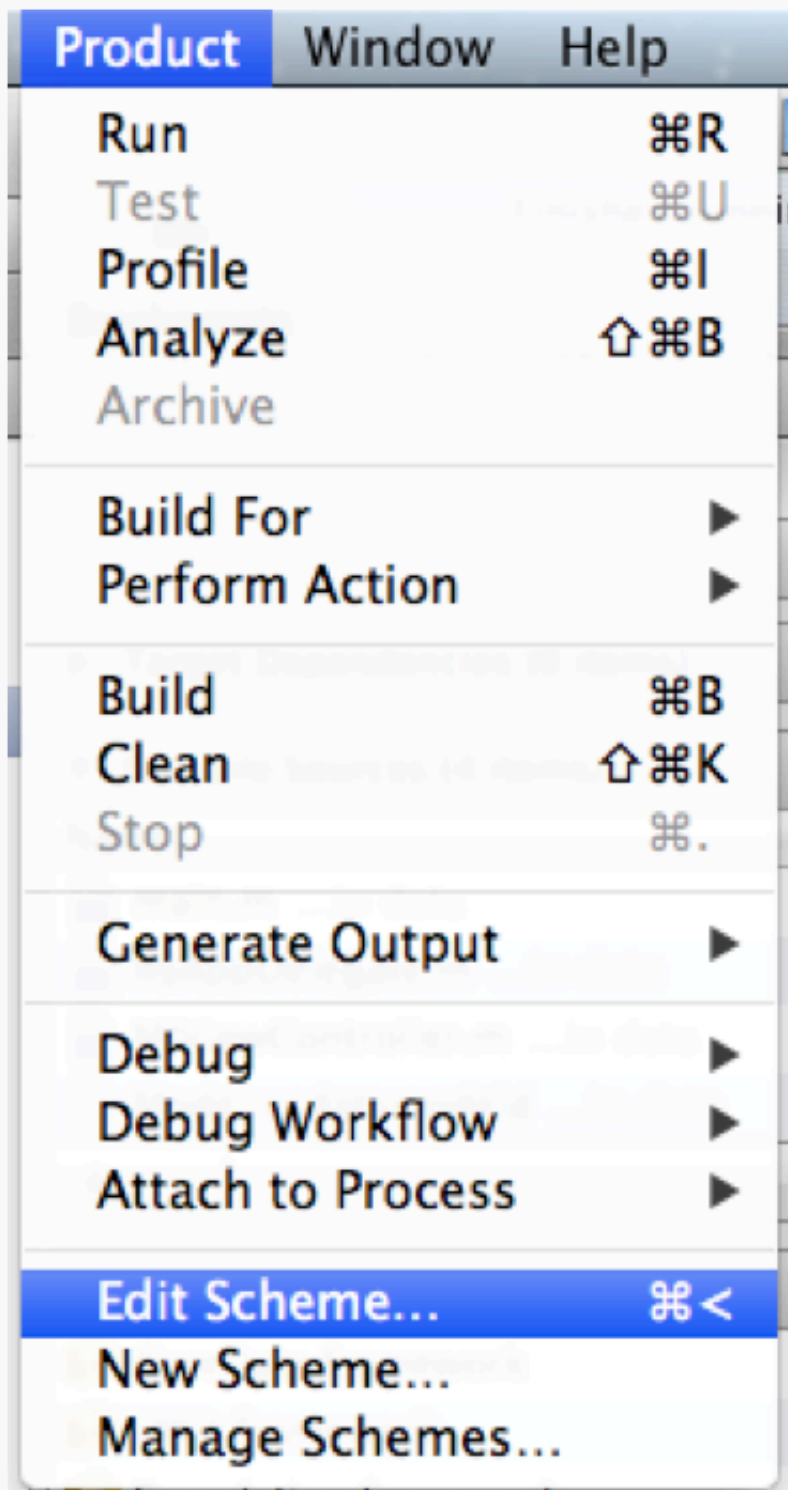
```

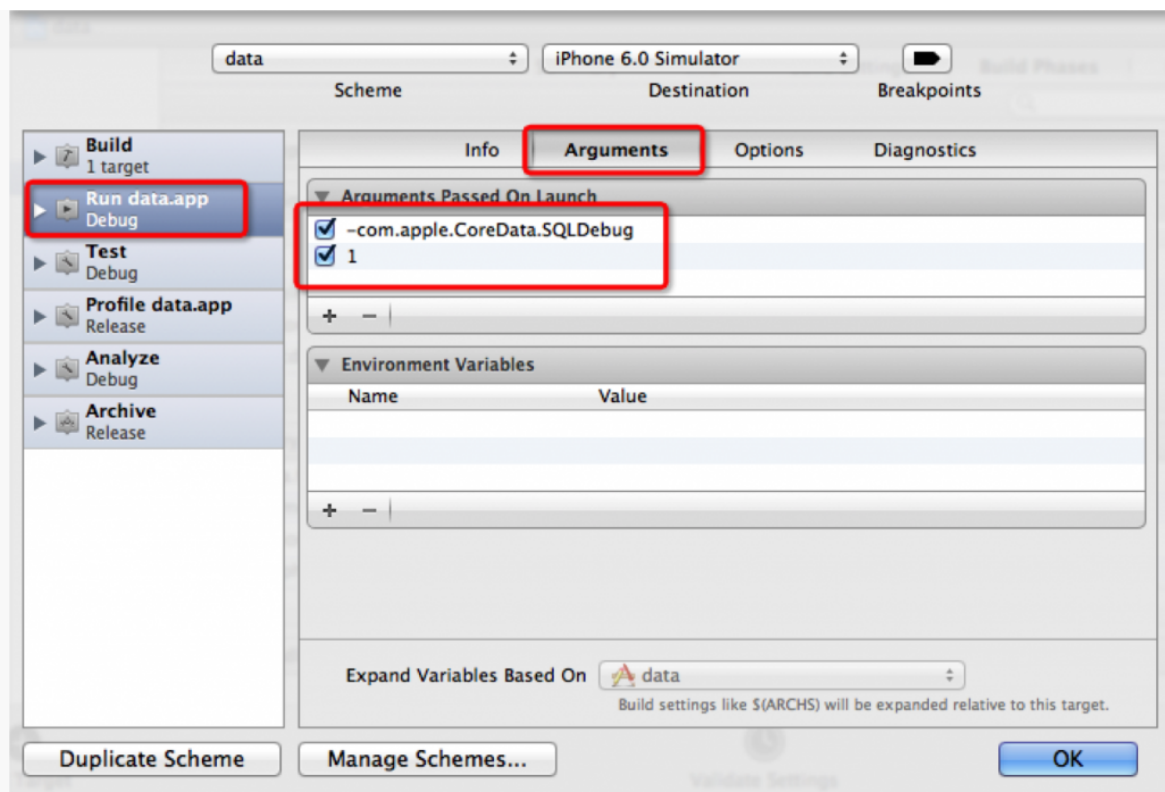
打开CoreData的SQL语句输出开关

1. 打开Product，点击EditScheme...

2. 点击Arguments, 在ArgumentsPassed On Launch中添加2项

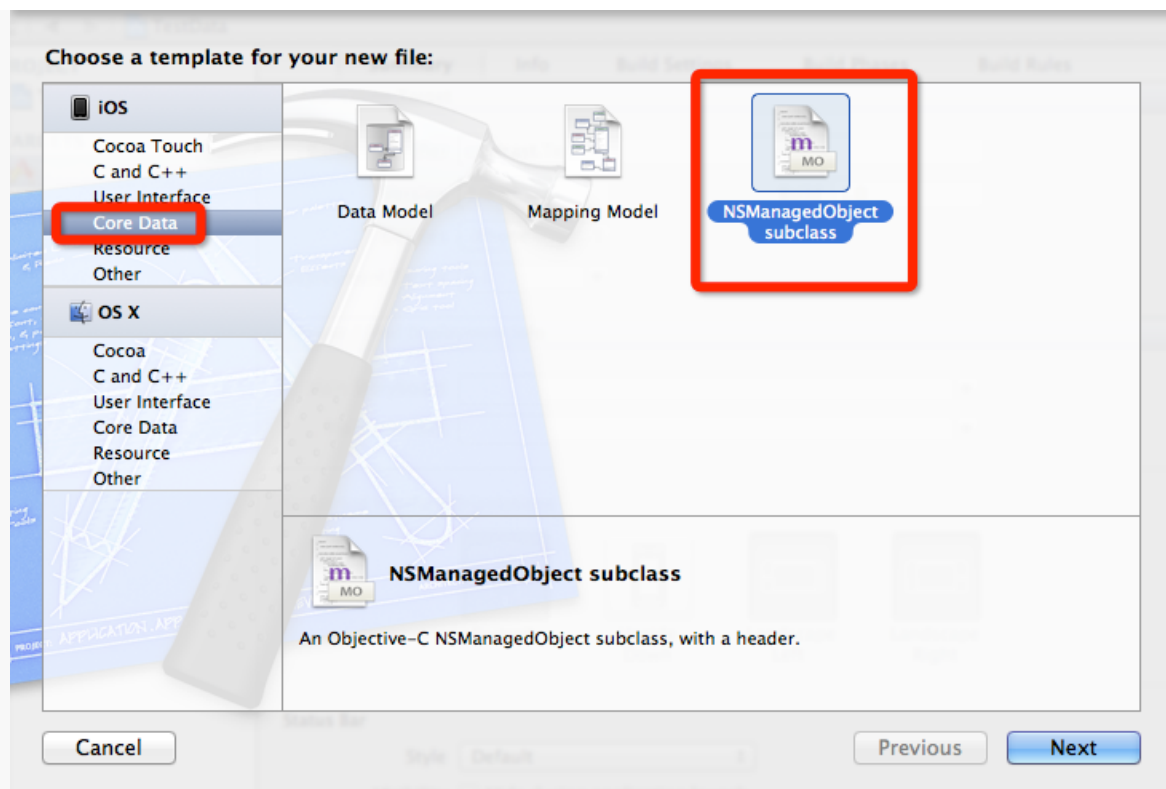
- -com.apple.CoreData.SQLDebug
- 1



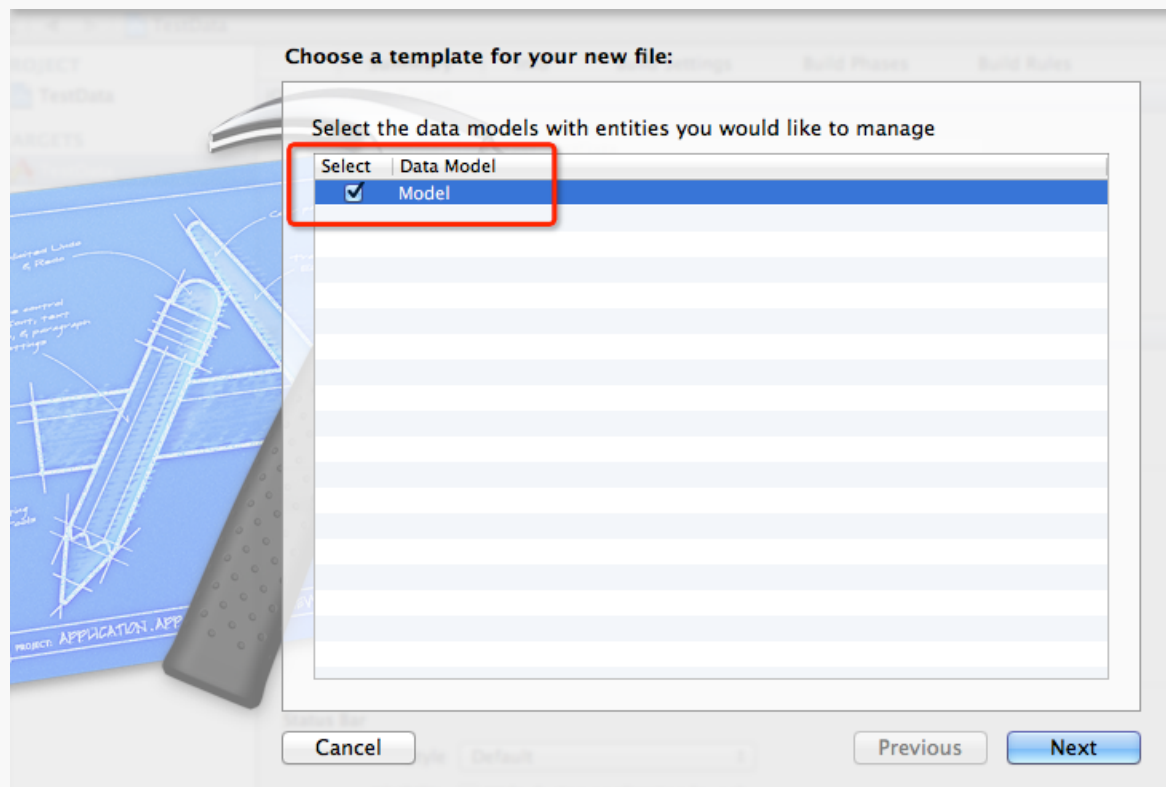


创建NSManagedObject的子类

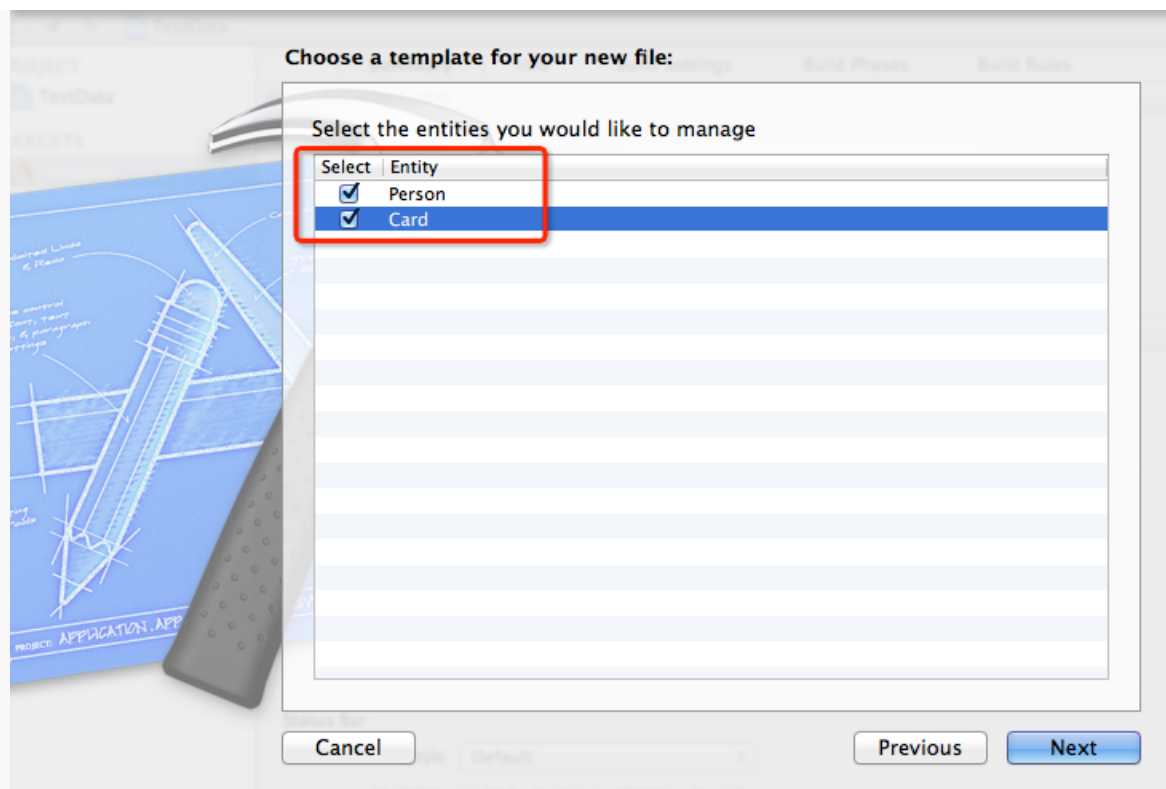
默认情况下，利用Core Data取出的实体都是NSManagedObject类型的，能够利用键-值对来存取数据。但是一般情况下，实体在存取数据的基础上，有时还需要添加一些业务方法来完成一些其他任务，那么就必须创建NSManagedObject的子类



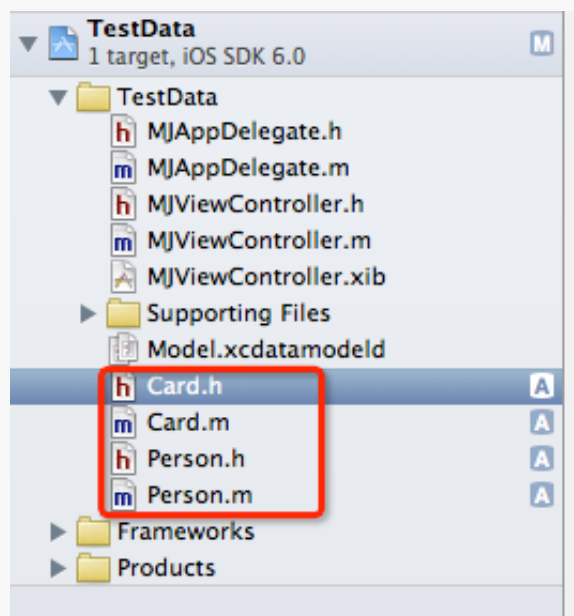
选择模型文件



选择需要创建子类的实体



创建完毕后，多了2个子类



文件内容展示：

Person.h

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

@class Card;

@interface Person : NSManagedObject

@property (nonatomic, retain) NSString * name;
@property (nonatomic, retain) NSNumber * age;
@property (nonatomic, retain) Card *card;

@end
```

Person.m

```
#import "Person.h"

@implementation Person

@dynamic name;
@dynamic age;
@dynamic card;

@end
```

Card.h

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

@class Person;

@interface Card : NSManagedObject

@property (nonatomic, retain) NSString * no;
@property (nonatomic, retain) Person *person;

@end
```

Card.m

```
#import "Card.h"
#import "Person.h"

@implementation Card

@dynamic no;
@dynamic person;

@end
```

使用NSManagedObject的子类

```
Person *person = [NSEntityDescription insertNewObjectForEntityForName:@"Person" inManagedObjectContext:context];
person.name = @"XHZ";
person.age = [NSNumber numberWithInt:27];

Card *card = [NSEntityDescription insertNewObjectForEntityForName:@"Card" inManagedObjectContext:context];
card.no = @"4414245465656";
person.card = card;
// 最后调用[context save&error];保存数据
```