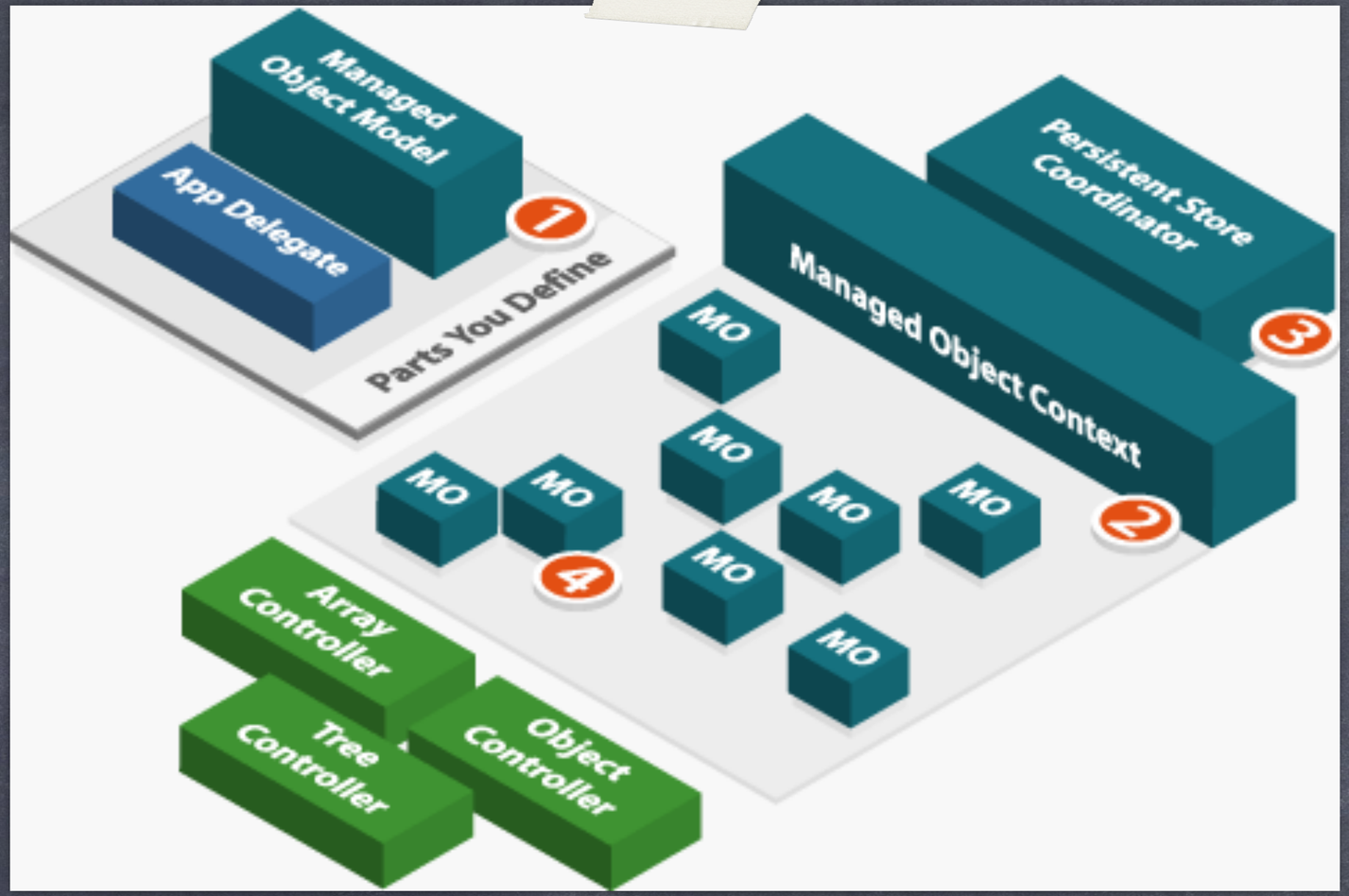
CoreData

什么是CoreData

- CoreData是Mac OS X中Cocoa API的一部分，首次出现在Mac OS 10.4 和iOS 3.0中。它提供了对象－关系映射（ORM）功能。可以将OC对象转化为数据，保存在SQLite数据库文件中，也能够将数据库中的文件还原成OC对象。在此期间，我们不需要编写任何SQL语句。
- 可以简单的理解为Cocoa对SQLite的一层封装。

为什么要使用CoreData

- 极大的减少Model层的代码量。
- 优化了使用SQLite时候的性能。
- 提供了可视化设计。


Managed Object Model
App Delegate
1
Parts You Define
Persistent Store Coordinator
Managed Object Context
3
2
MO
MO
MO
MO
MO
MO
MO
MO
4
Array Controller
Tree Controller
Object Controller

框架详解

- Managed Object Model是描述应用程序的数据模型，这个模型包含实体（Entity），特性（Property），读取请求（FetchRequest）等。
- Managed Object Context参与对数据对象进行操作的全过程，并检测数据对象的变化，以提供对undo / redo的支持及更新绑定到数据的UI。
- PersisterStore Coordinator相当于数据文件管理器，处理底层的对数据文件的读取与写入。一般我们不跟它打交道。
- Managed Object是数据对象，与Managed Object Context相关联。
- Controller是图中的绿色部分，一般是通过control + drag将Managed Object Context绑定到它们，这样我们就可以在xib中可视化的操作数据。Mac特有，iOS中没有该功能。

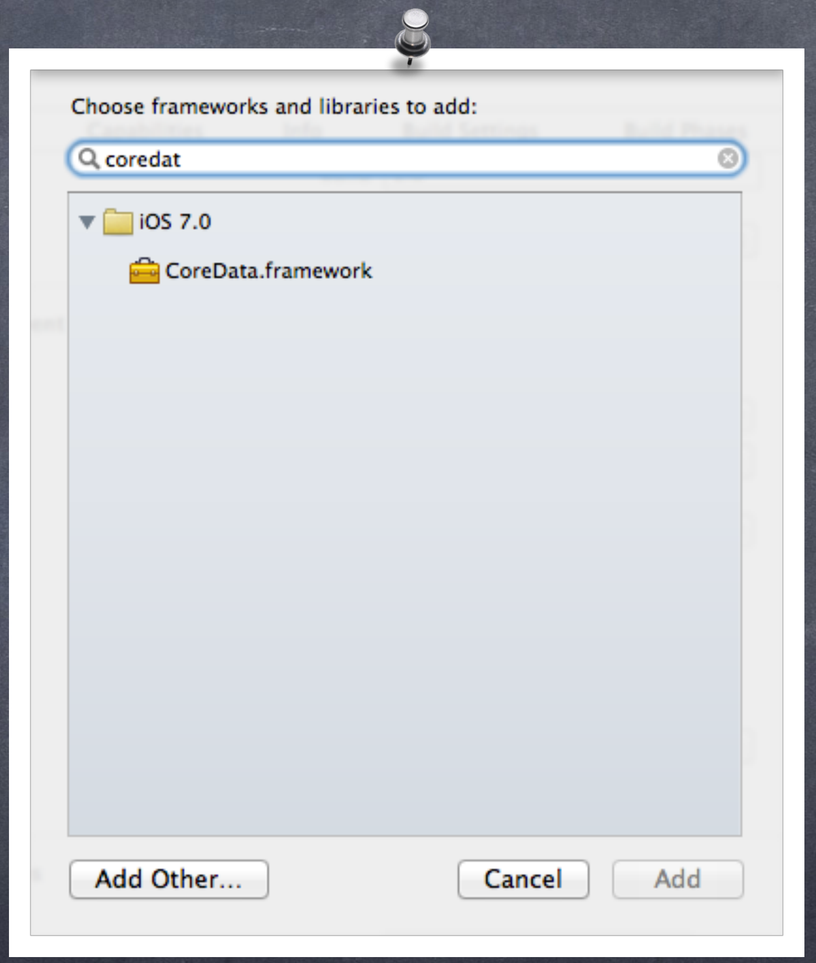
- 应用程序先创建或读取模型文件（后缀为xcdataModeld，编译后后缀为momd）生成NSManagedObjectModel对象。

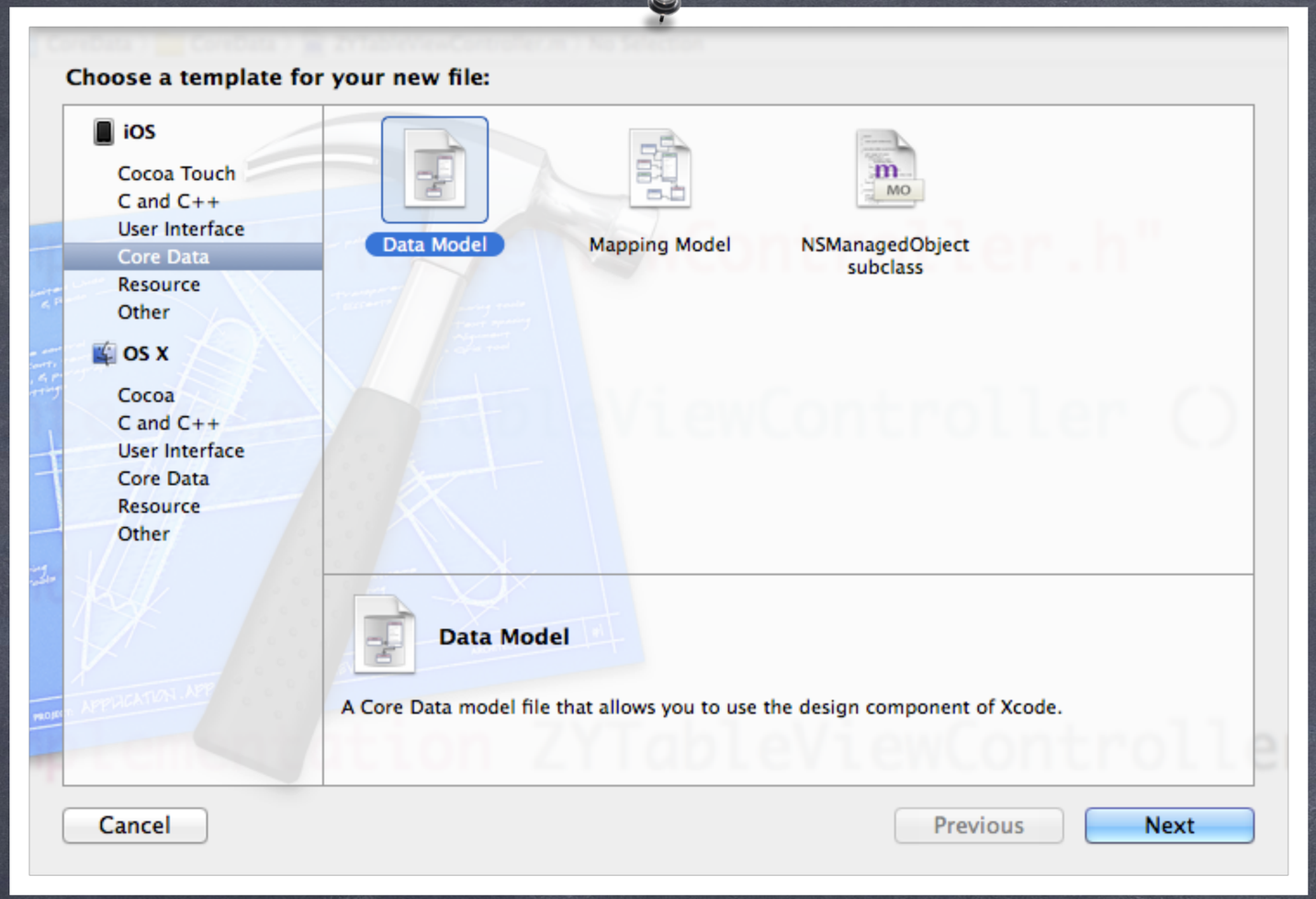
- 然后生成NSManagedObjectContext和NSPersisterStoreCoordinator对象。前者对用户透明的调用后者进行数据文件的读写。
- NSPersisterStoreCoordinator负责从数据文件（xml，sqlite，二进制文件等）中读取数据生成NSManagedObject，或保存NSManagedObject写入数据库。
- NSManagedObjectContext参与对数据进行各种操作的整个过程，持有NSManagedObject。整个类最常被用到。
- controller同上。

Demo1

准备工作

- 创建项目
- xcode4版本的导入CoreData库，并引入头文件。

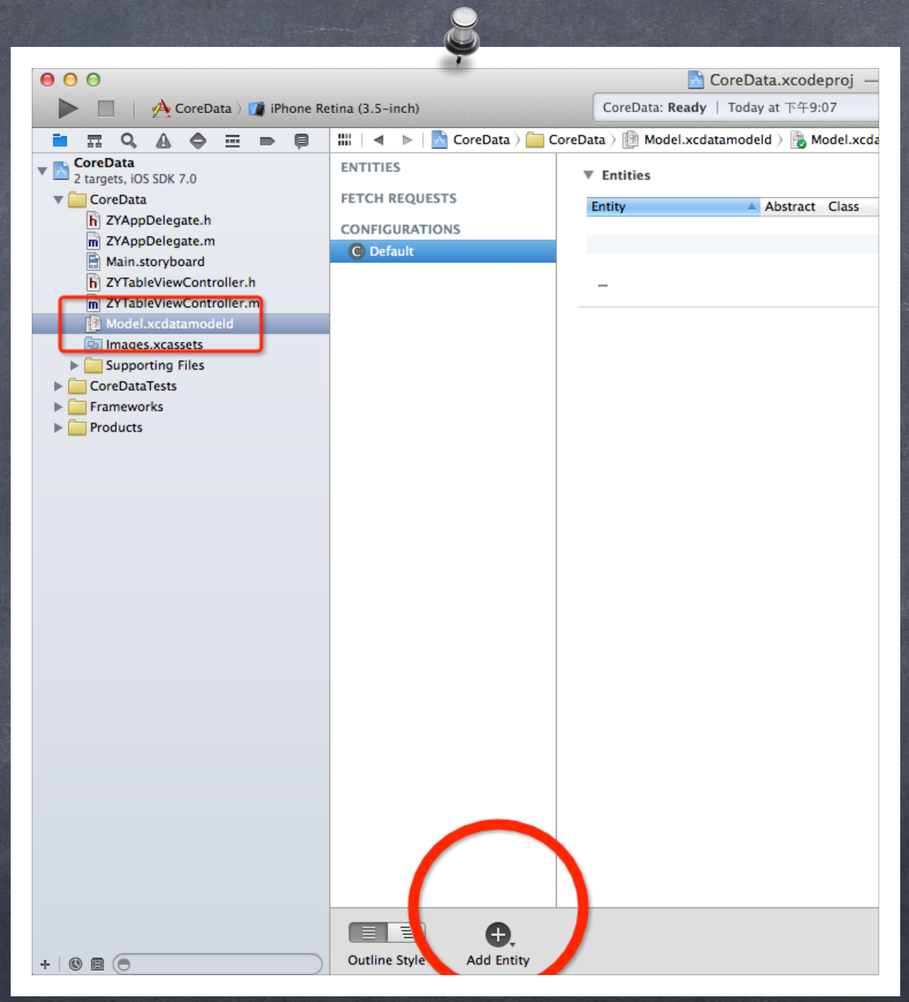


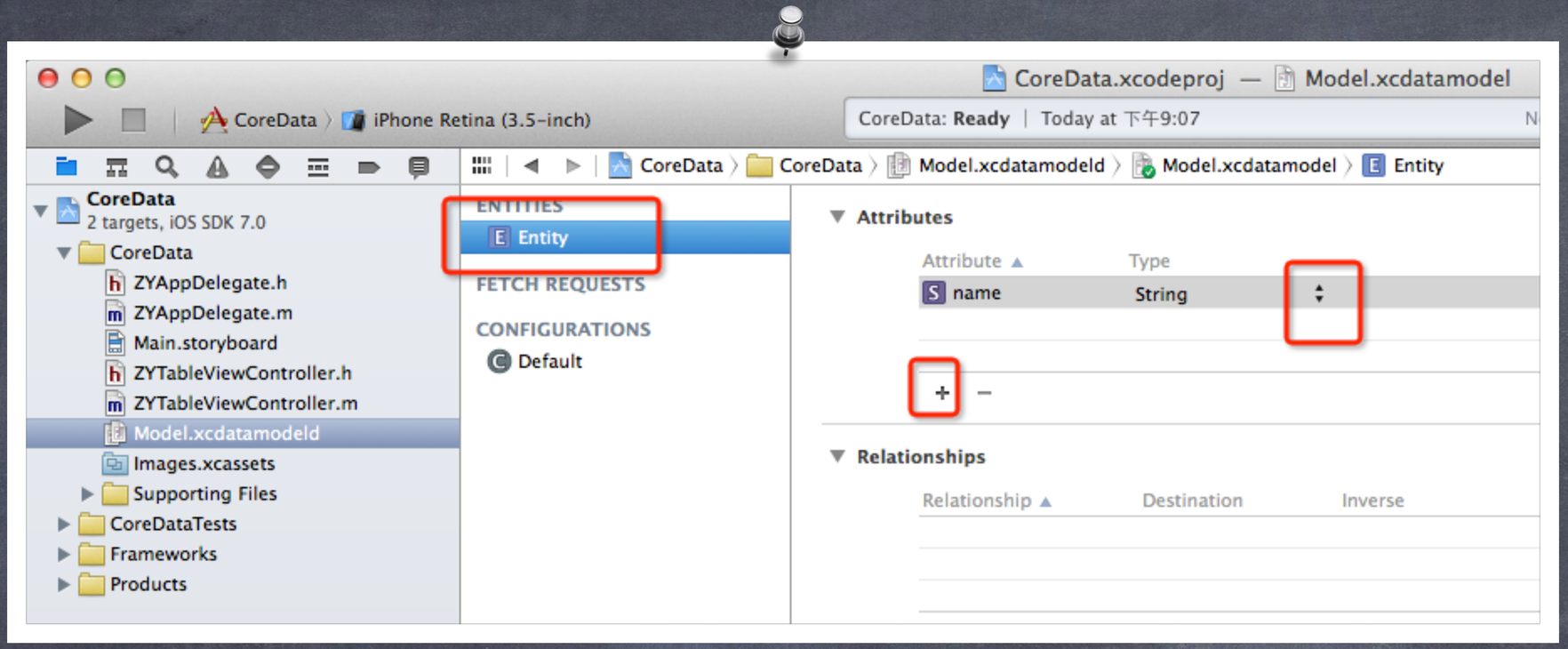


创建xcdataModeld对象

添加一个表（Entity）

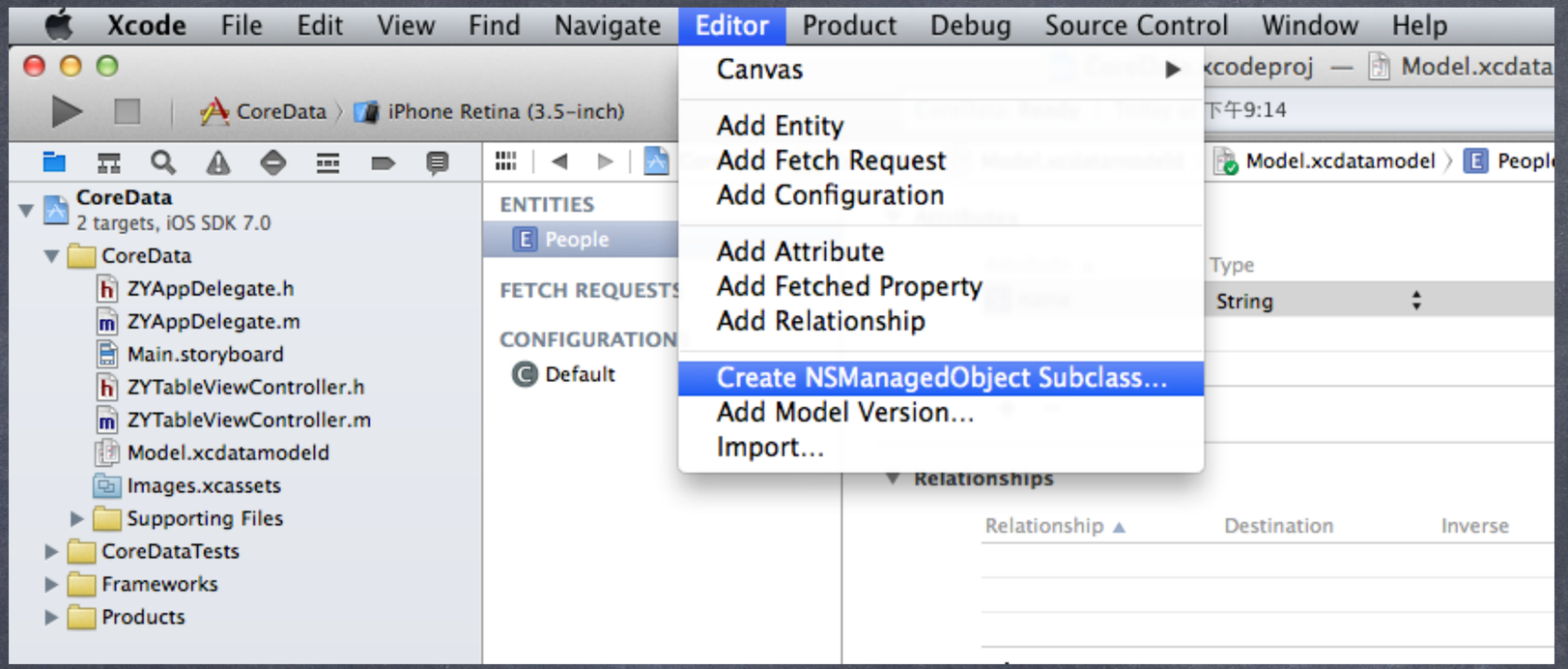
xcode5中点击底部的Add Entity按钮





为表添加属性

点击Entity修改表名称，点击加号，添加属性名称，点击箭头，修改属性的类型



创建NSManagedObject

点击如上图操作，然后一路next。

```objc
#import "ZYDAO.h"

@interface ZYDAO () {
    NSManagedObjectContext* _context;
    NSPersistentStoreCoordinator*
            _persistentStoreCoordinator;
    NSManagedObjectModel* _model;
}

@end
```

创建数据库管理类

创建ZYDAO单例类并声明三个对象。思考一下为什么要写为单例类?

```objc
//链接数据库
- (void)connectSqlite {
    //xcdataModeld文件编译后为momd文件,从资源文件加载NSManagedObjectModel对象
    NSURL* url = [[NSBundle mainBundle] URLForResource:@"Model"
            withExtension:@"momd"];
    _model = [[NSManagedObjectModel alloc] initWithContentsOfURL:url];
    //通过NSManagedObjectModel对象创建NSPersistentStoreCoordinator对象
    _persistentStoreCoordinator = [[NSPersistentStoreCoordinator alloc]
            initWithManagedObjectModel:_model];
    //获取Document文件夹下的sqlite文件
    NSString* path = [[NSSearchPathForDirectoriesInDomains
            (NSDocumentDirectory, NSUserDomainMask, YES) lastObject]
            stringByAppendingPathComponent:@"Model.sqlite"];
   //加载sqlite数据库文件
   if ([_persistentStoreCoordinator addPersistentStoreWithType:
           NSSQLiteStoreType configuration:nil URL:[NSURL fileURLWithPath:
           path] options:nil error:nil]) {
       //创建NSManagedObjectContext对象,
       _context = [[NSManagedObjectContext alloc] init];
       //为NSManagedObjectContext对象设置存储协调着
       [_context setPersistentStoreCoordinator:_persistentStoreCoordinator];
   }else {
       //加载出错就退出程序
       abort();
   }
```

```objc
//向表中插入一条新数据
- (void)insertPeopleWithName:(NSString* )name {
    //向People表中插入一条新数据，要注意表名完全一致，区分大小写。这里使用了宏定义。
    People* people = [NSEntityDescription insertNewObjectForEntityForName:
            PeopleTable inManagedObjectContext:_context];
    //为people的name属性赋值。
    people.name = name;
    //保存更改
    NSError* error = nil;
    if (![_context save:&error]) {
        NSLog(@"插入数据失败");
    }
}
```

向表中插入数据

注意导入People头文件，宏定义PeopleTable = People，一定要与表名称完全一致。

```objc
//检索所有的peple
- (NSArray* )searchAllPeople {
    //创建查询请求
    NSFetchRequest* request = [[NSFetchRequest alloc] init];
    //创建表描述对象
    NSEntityDescription* entityDescription = [NSEntityDescription
            entityForName:PeopleTable inManagedObjectContext:_context];
    //设置需要查询的表
    [request setEntity:entityDescription];
    //在NSManagedObjectContext中执行查询请求
    NSArray* result = [_context executeFetchRequest:request error:nil];

    return result;
}
```

检索people表的所有值

使用NSFetchRequest

```objc
//更新people对象
- (void)updatePeople:(People* )people {
    //保存更改
    NSError* error = nil;
    if (![_context save:&error]) {
        NSLog(@"更新数据失败");
    }
}
```

更新一个People对象的值

直接修改people对象的属性值，然后保存即可。

```objc
//删除一行people数据
- (void)deletePeople:(People* )people {
    //从NSManagedObjectContext中删除一个people对象
    [_context deleteObject:people];
    //保存更改
    NSError* error = nil;
    if (![_context save:&error]) {
        NSLog(@"更新数据失败");
    }
}
```

删除一行people数据

结束

-王礼星