



Core Data



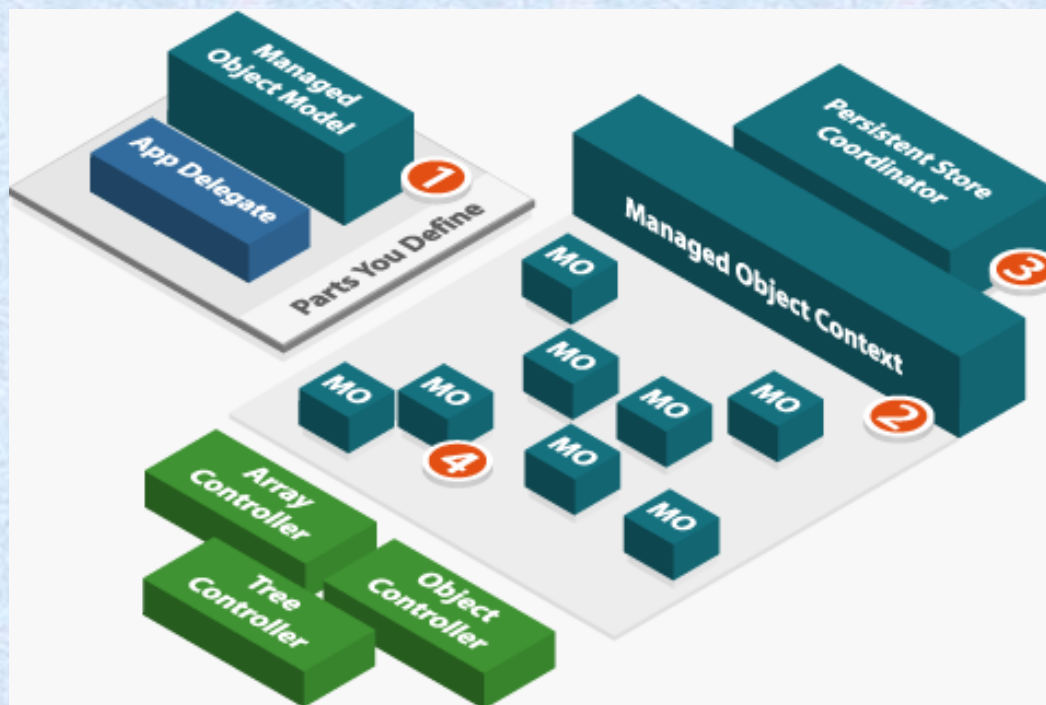
一、Core Data 介绍

- Core Data是什么?
- 为什么需要Core Data?

对象数据持久化的需要。



二、Core Data 框架组成





1、ManagedObjectModel

数据模型，这个模型包含实体（Entity），属性（Property），数据请求（Fetch Request）等。

在Core Data框架中对应的类是： `NSManagedObjectModel`

An `NSManagedObjectModel` object describes a **schema**—a collection of **entities** (data models) that you use in your application.

The model contains one or more **`NSEntityDescription`** objects representing the entities in the schema. Each `NSEntityDescription` object has property description objects (instances of subclasses of **`NSPropertyDescription`**) that represent the properties (or fields) of the entity in the schema. The Core Data framework uses this description in several ways:

A managed object model maintains a mapping between each of its entity objects and a corresponding managed object class for use with the persistent storage mechanisms in the Core Data Framework. You can determine the entity for a particular managed object with the `entity` method.

You typically create managed object models using the data modeling tool in Xcode, but it is possible to build a model programmatically if needed.



2、Managed Object Context

对数据对象进行监测和管理（增、删、查、改）。

在Core Data框架中对应的类是： `NSManagedObjectContext`

An instance of `NSManagedObjectContext` represents a single “object space” or scratch pad in an application. Its primary responsibility is to manage a collection of managed objects. These objects form a group of related model objects that represent an internally consistent view of one or more persistent stores. A single managed object instance exists in one and only one context, but multiple copies of an object can exist in different contexts. Thus object uniquing is scoped to a particular context.

说明可以对应一个实体可以有多个对象的拷贝，被不同的`NSManagedObjectContext`管理



与增、删、查、改相关的几个方法

- save:

Attempts to commit unsaved changes to registered objects to the receiver's parent store.

Declaration

OBJECTIVE-C

```
- (BOOL)save:(NSError **)error
```

Parameters

error

A pointer to an `NSError` object. You do not need to create an `NSError` object. The save operation aborts after the first failure if you pass `NULL`.

Return Value

YES if the save succeeds, otherwise NO.

保存变化了的数据对象至持久层。



- `executeFetchRequest:error:`

Returns an array of objects that meet the criteria specified by a given fetch request.

Declaration

OBJECTIVE-C

```
- (NSArray *)executeFetchRequest:(NSFetchRequest *)request  
                        error:(NSError **)error
```

Parameters

<i>request</i>	A fetch request that specifies the search criteria for the fetch.
<i>error</i>	If there is a problem executing the fetch, upon return contains an instance of <code>NSError</code> that describes the problem.

Return Value

An array of objects that meet the criteria specified by *request* fetched from the receiver and from the persistent stores associated with the receiver's persistent store coordinator. If an error occurs, returns `nil`. If no objects match the criteria specified by *request*, returns an empty array.

向持久层请求数据。



- deleteObject:

Specifies an object that should be removed from its persistent store when changes are committed.

Declaration

OBJECTIVE-C

```
- (void)deleteObject:(NSManagedObject *)object
```

Parameters

<i>object</i>	A managed object.
---------------	-------------------

Discussion

When changes are committed, *object* will be removed from the uniquing tables. If *object* has not yet been saved to a persistent store, it is simply removed from the receiver.

删除一个数据对象，注意要和Save方法配合使用



- insertObject:

Registers an object to be inserted in the receiver's persistent store the next time changes are saved.

Declaration

OBJECTIVE-C

```
- (void)insertObject:(NSManagedObject *)object
```

Parameters

<i>object</i>	A managed object.
---------------	-------------------

Discussion

The managed object (*object*) is registered in the receiver with a temporary global ID. It is assigned a permanent global ID when changes are committed. If the current transaction is rolled back (for example, if the receiver is sent a `rollback` message) before a save operation, the object is unregistered from the receiver.

插入一个数据对象。注意：建议使用NSEntityDescription对象中的insertNewObjectForEntityForName方法插入一个新的数据对象。



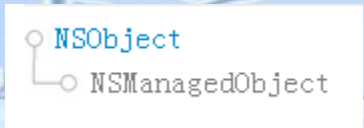
3、Persistent Store Coordinator

即数据对象的持久层协调器。（其负责和底层数据文件的读写，但这一切都被封装屏蔽，开发者无需关注具体实现细节。）

在Core Data框架中对应的类是： **NSPersistentStoreCoordinator**

Instances of **NSPersistentStoreCoordinator** associate persistent stores (by type) with a model (or more accurately, a configuration of a model) and serve to mediate between the persistent store or stores and the managed object context or contexts. Instances of **NSManagedObjectContext** use a coordinator to save object graphs to persistent storage and to retrieve model information. A context without a coordinator is not fully functional as it cannot access a model except through a coordinator. The coordinator is designed to present a façade to the managed object contexts such that a group of persistent stores appears as an aggregate store. A managed object context can then create an object graph based on the union of all the data stores the coordinator covers.

Coordinators do the opposite of providing for concurrency—they serialize operations. If you want to use **multiple threads for different write operations** you use **multiple coordinators**. Note that if multiple threads work directly with a coordinator, they need to **lock and unlock it** explicitly.



4、Managed Object

由与 Managed Object Context管理的数据对象。可以把其看成是保存在持久层中。

在Core Data框架中对应的类是： **NSManagedObject**

NSManagedObject is a generic class that implements all the basic behavior required of a Core Data model object. **It is not possible to use instances of direct subclasses of NSObject** (or any other class not inheriting from NSManagedObject) with a managed object context. You may create custom subclasses of NSManagedObject, although this is not always required. If no custom logic is needed, a complete object graph can be formed with NSManagedObject instances.

A managed object is associated with an entity description (an instance of NSEntityDescription) that provides metadata about the object (including the name of the entity that the object represents and the names of its attributes and relationships) and with a managed object context that tracks changes to the object graph. It is important that a managed object is properly configured for use with Core Data.

If you instantiate a managed object directly, you must call the designated initializer (**initWithEntity:insertIntoManagedObjectContext:**).



几个经常用到的属性和方法。

entity

The entity description of the receiver. (read-only)

Declaration

OBJECTIVE-C

```
@property(n nonatomic, readonly, strong) NSEntityDescription *entity
```

managedObjectContext

hasChanges

inserted

updated

deleted

—————→ 返回值是BOOL型



- `willSave`

Invoked automatically by the Core Data framework when the receiver's managed object context is saved.

- `didSave`

Invoked automatically by the Core Data framework after the receiver's managed object context completes a save operation.

- `prepareForDeletion`

Invoked automatically by the Core Data framework when the receiver is about to be deleted.



通过键值操作获取对象的属性（通过实现NSKeyValueCoding协议）

```
- valueForKey:
```

Returns the value for the property specified by *key*.

Declaration

OBJECTIVE-C

```
- (id)valueForKey:(NSString *)key
```

Parameters

<i>key</i>	The name of one of the receiver's properties.
------------	---

Return Value

The value of the property specified by *key*.

通过键获取值



- setValue:forKey:

Sets the specified property of the receiver to the specified value.

Declaration

OBJECTIVE-C

```
- (void)setValue:(id) value  
    forKey:(NSString *)key
```

Parameters

<i>value</i>	The new value for the property specified by <i>key</i> .
<i>key</i>	The name of one of the receiver's properties.

通过键设置属性值



在大部分情况，不建议直接利用上面的键值方法来获取和设置属性。更多情况下式是通过对NSManagedObject进行继承获得子类，在子类对象上直接进行属性操作。

具体操作见视频演示。



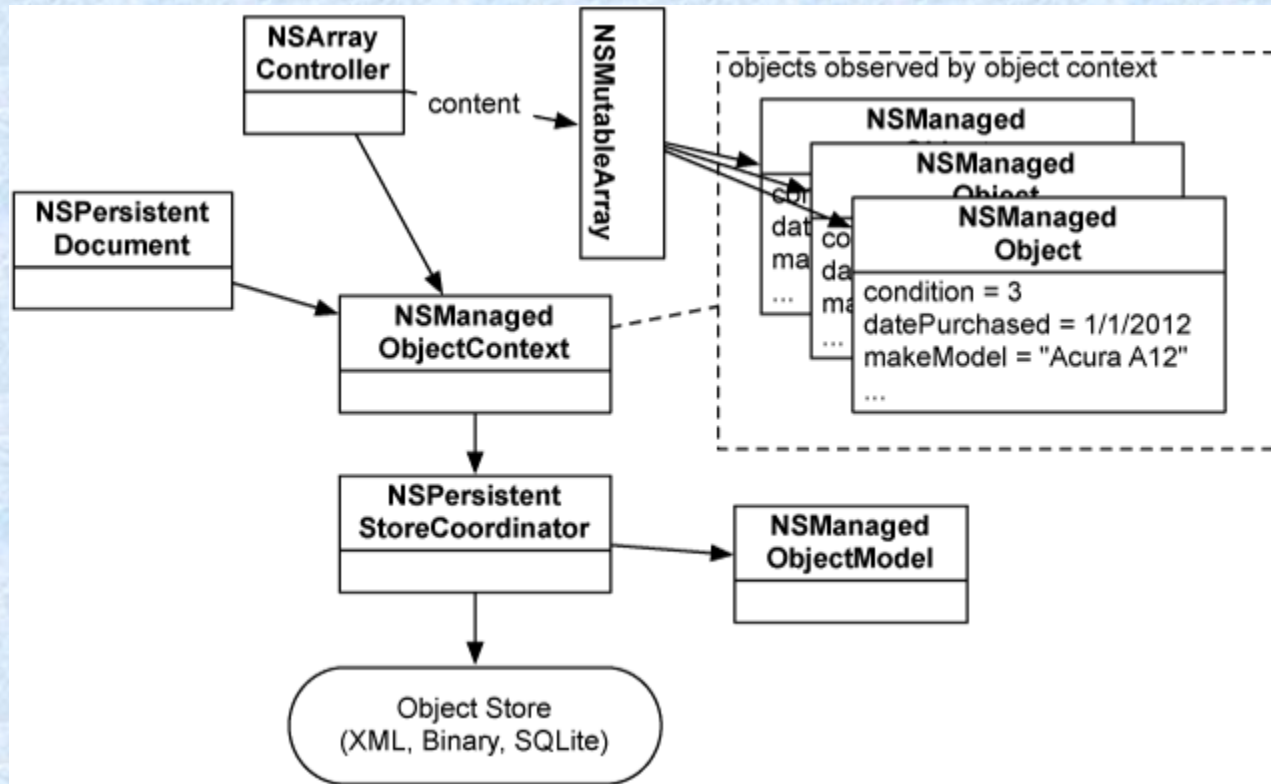


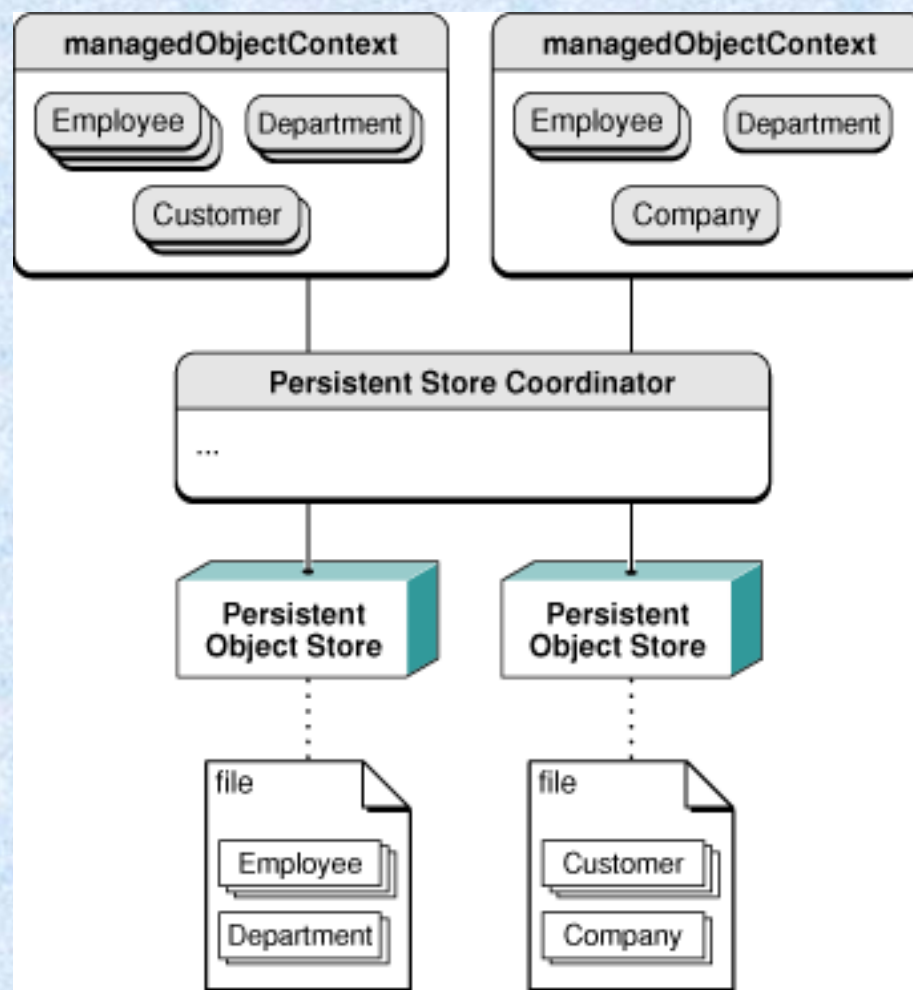
5、Controller

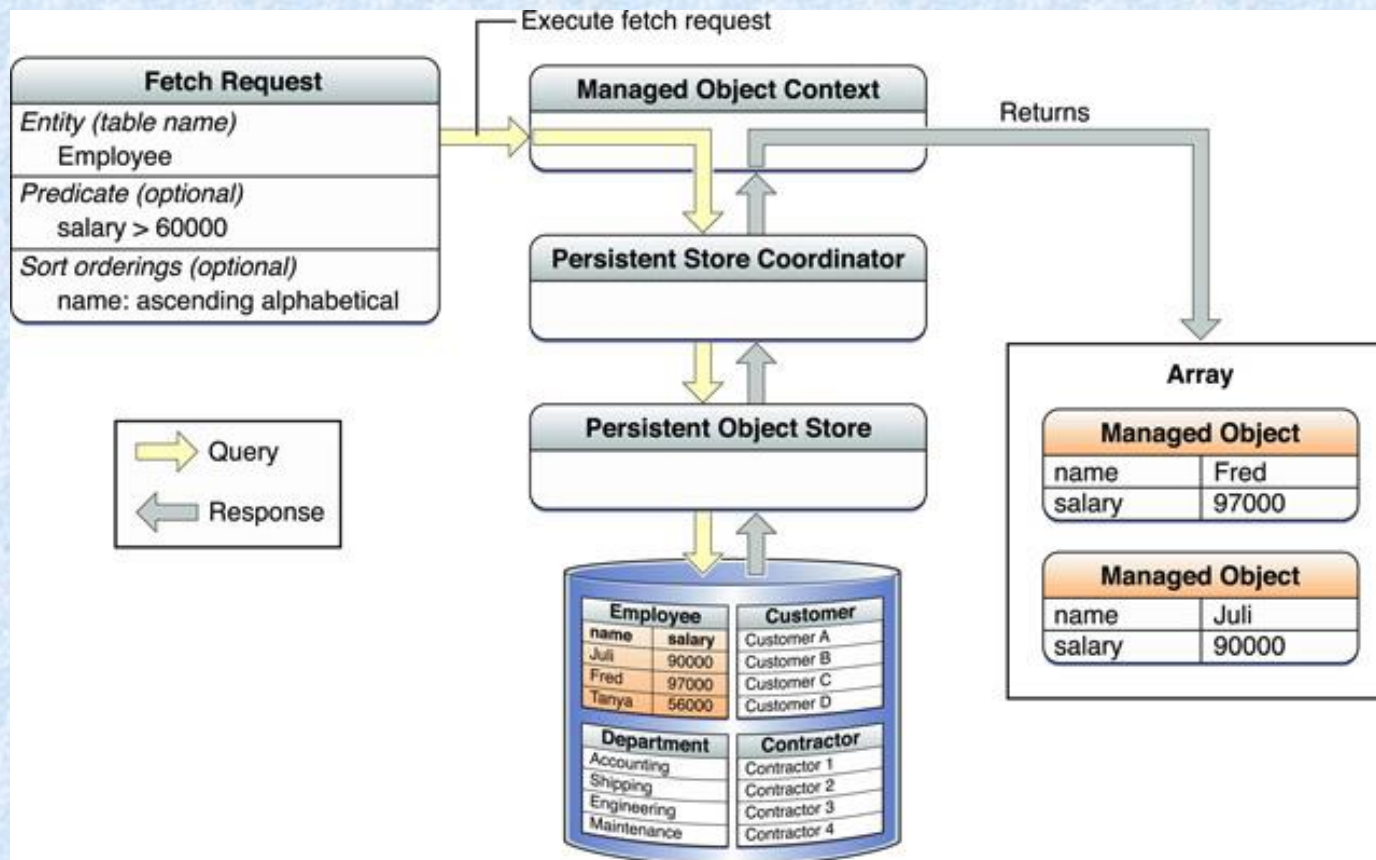
NSManagedObject对象的操作者， NSManagedObject对象可以像其它对象那样被NSArray等读取和修改，但在保存到持久层时候需要利用NSManagedObjectContext。



Core Data框架运作示例









6、NSFetchRequest

获取数据的请求，通过NSManagedObjectContext向持久层发出请求，请求获得需要的对象集合。可以通过对NSFetchRequest进行设置来发出不同需求的请求。

在Core Data框架中对应的类是： NSFetchRequest

An instance of NSFetchRequest describes search criteria used to retrieve data from a persistent store.



对数据对象进行筛选

predicate

The predicate of the receiver.

Declaration

OBJECTIVE-C

```
@property(nonatomic, strong) NSPredicate *predicate
```

```
NSString *str=@"Zhang";  
NSFetchRequest *request=[[NSFetchRequest alloc] init];  
request.predicate=[NSPredicate predicateWithFormat:@"name contains %@",str];
```




●比较运算符 >、<、==、>=、<=、!=

例：@ "number >= 20"

●范围运算符：IN、BETWEEN

例：@ "number BETWEEN {1,5}"

●字符串相关：BEGINSWITH、ENDSWITH、CONTAINS

例：@ "name CONTAINS 'Zh'" //包含某个字符串

@ "name BEGINSWITH 'sh'" //以某个字符串开头

@ "name ENDSWITH 'ang'" //以某个字符串结束

●通配符：LIKE

例：@ "name LIKE[cd] '*er*'" // *和?代表通配符

@ "name LIKE[cd] '???er*'"



对获得的数据对象进行排序

`sortDescriptors`

The sort descriptors of the receiver.

Declaration

OBJECTIVE-C

```
@property(nonatomic, strong) NSArray *sortDescriptors
```

Discussion

The sort descriptors specify how the objects returned when the fetch request is issued should be ordered—for example by last name then by first name. The sort descriptors are applied in the order in which they appear in the `sortDescriptors` array (serially in lowest-array-index-first order).

注意该属性是一个数组。用以指出多种排序方法。

```
request.sortDescriptors=@[[NSSortDescriptor sortDescriptorWithKey:@"score" ascending:NO],  
[NSSortDescriptor sortDescriptorWithKey:@"name" ascending:YES]];
```




对获得的数据对象数量进行限制

fetchLimit

The fetch limit of the receiver.

Declaration

OBJECTIVE-C

```
@property(nonatomic) NSUInteger fetchLimit
```

Discussion

The fetch limit specifies the maximum number of objects that a request should return when executed.

request.fetchLimit=100



对获得的数据对象的缓存进行设置

`fetchBatchSize`

The batch size of the receiver.

Declaration

OBJECTIVE-C

```
@property(nonatomic) NSUInteger fetchBatchSize
```

Discussion

The default value is 0. A batch size of 0 is treated as infinite, which disables the batch faulting behavior.

If you set a non-zero batch size, the collection of objects returned when the fetch is executed is broken into batches. When the fetch is executed, the entire request is evaluated and the identities of all matching objects recorded, but no more than *batchSize* objects' data will be fetched from the persistent store at a time. The array returned from executing the request will be a proxy object that transparently faults batches on demand. (In database terms, this is an in-memory cursor.) → 游标

You can use this feature to restrict the working set of data in your application. In combination with `fetchLimit`, you can create a subrange of an arbitrary result set.

`request.fetchBatchSize=20`



对获得的数据对象的进行偏移量设置

fetchOffset

The fetch offset of the receiver.

Declaration

OBJECTIVE-C

```
@property(nonatomic) NSUInteger fetchOffset
```

Discussion

The default value is 0.

This setting allows you to specify an offset at which rows will begin being returned. Effectively, the request will skip over the specified number of matching entries. For example, given a fetch which would normally return a, b, c, d, specifying an offset of 1 will return b, c, d, and an offset of 4 will return an empty array. Offsets are ignored in nested requests such as subqueries.

可利用该属性对表视图进行分页操作



指明(绑定)实体Entity

entity

The entity specified for the receiver.

Declaration

OBJECTIVE-C

```
@property(nonatomic, strong) NSEntityDescription *entity
```

指出想要获取的实体对象。



7、EntityDescription

代表数据模型(ManagedObjectModel)中的实体(Entity)

在Core Data框架中对应的类是： **NSEntityDescription**

An NSEntityDescription object describes an entity in Core Data. Entities are to managed objects what Class is to id, or—to use a database analogy—what tables are to rows. An instance specifies an entity's name, its properties (its attributes and relationships, expressed by instances of NSAttributeDescription and NSRelationshipDescription) and the class by which it is represented.



从数据模型中获得实体

+ entityForName:inManagedObjectContext:

Returns the entity with the specified name from the managed object model associated with the specified managed object context's persistent store coordinator.

Declaration

OBJECTIVE-C

```
+ (NSEntityDescription *)entityForName:(NSString *)entityName  
    inManagedObjectContext:(NSManagedObjectContext *)context
```

Parameters

<i>entityName</i>	The name of an entity.
<i>context</i>	The managed object context to use. Must not be nil.

Return Value

The entity with the specified name from the managed object model associated with *context*'s persistent store coordinator.

```
NSEntityDescription *entity=[NSEntityDescription entityForName: entityname  
                                inManagedObjectContext: self.context];
```



give

ed &

Q'S



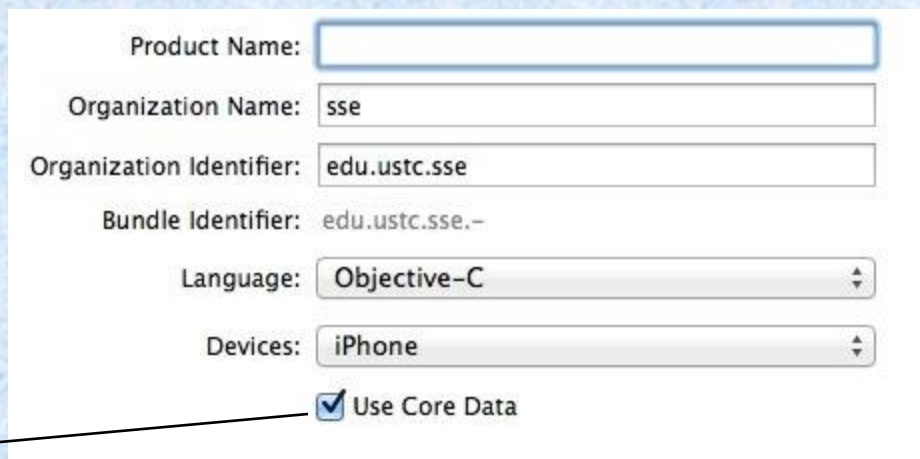
二、利用Core Data框架进行增、删、查和改

1、设计并创建数据对象模型ManagedObjectModel

可利用Xcode提供的数据对象模型编辑器快捷方便的创建。

添加模型的两种方法。

方法1、在创建工程时选中



The image shows the 'Create a new Xcode project' dialog in Xcode. The 'Product Name' field is empty and highlighted with a blue border. The 'Organization Name' is 'sse', 'Organization Identifier' is 'edu.ustc.sse', and 'Bundle Identifier' is 'edu.ustc.sse.-'. The 'Language' is set to 'Objective-C' and 'Devices' is set to 'iPhone'. At the bottom, the checkbox 'Use Core Data' is checked.

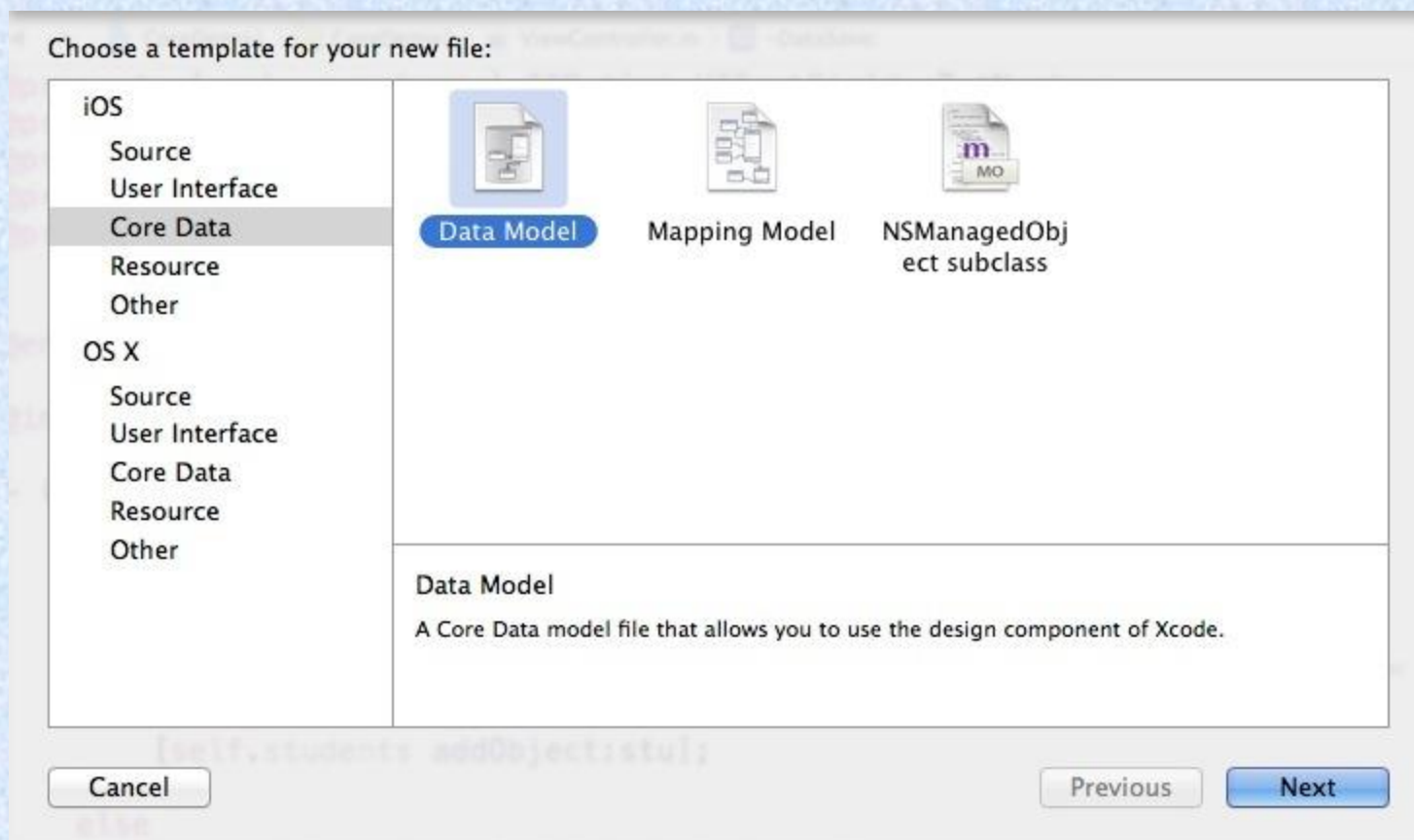
Product Name:	
Organization Name:	sse
Organization Identifier:	edu.ustc.sse
Bundle Identifier:	edu.ustc.sse.-
Language:	Objective-C
Devices:	iPhone
	<input checked="" type="checkbox"/> Use Core Data

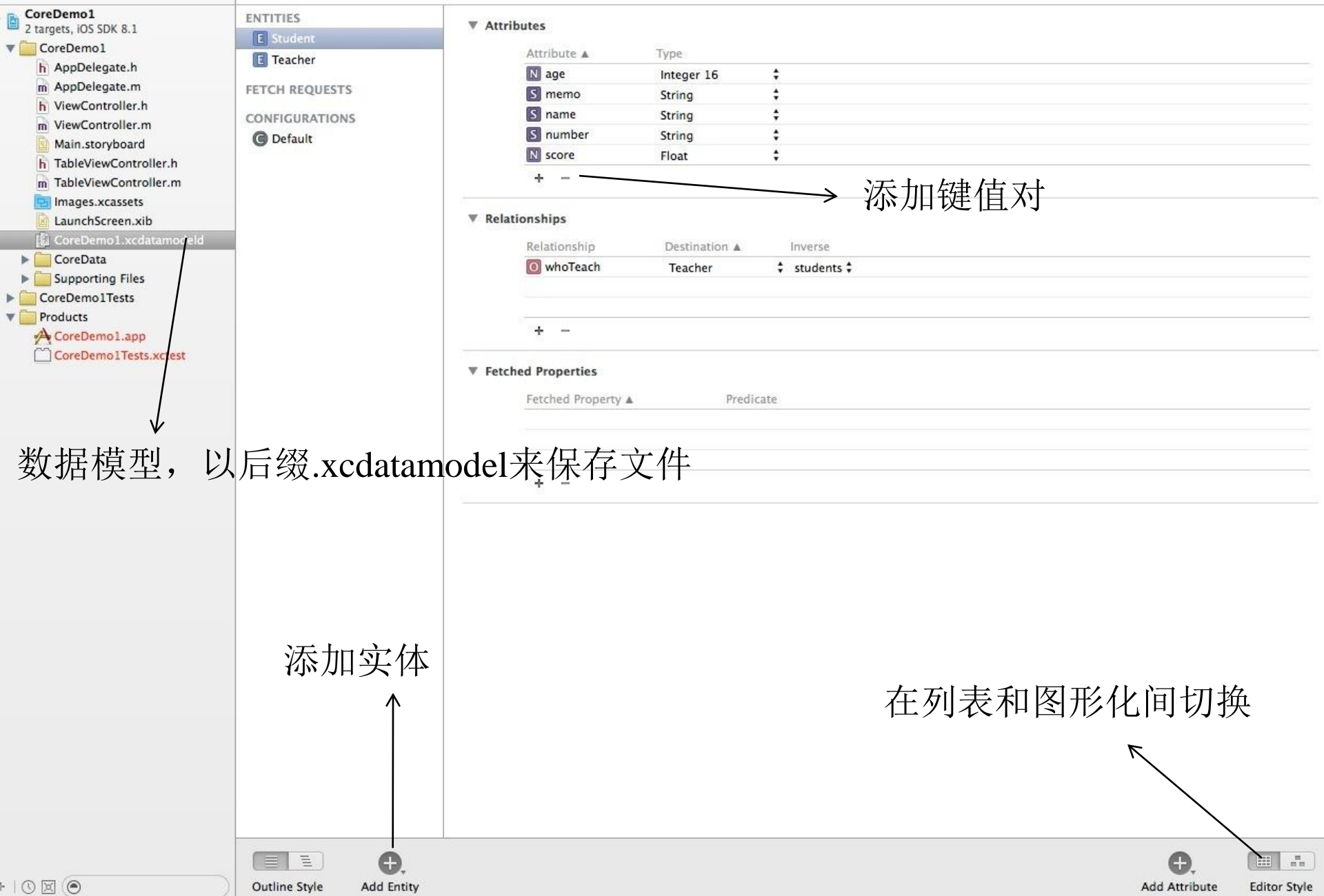
选中



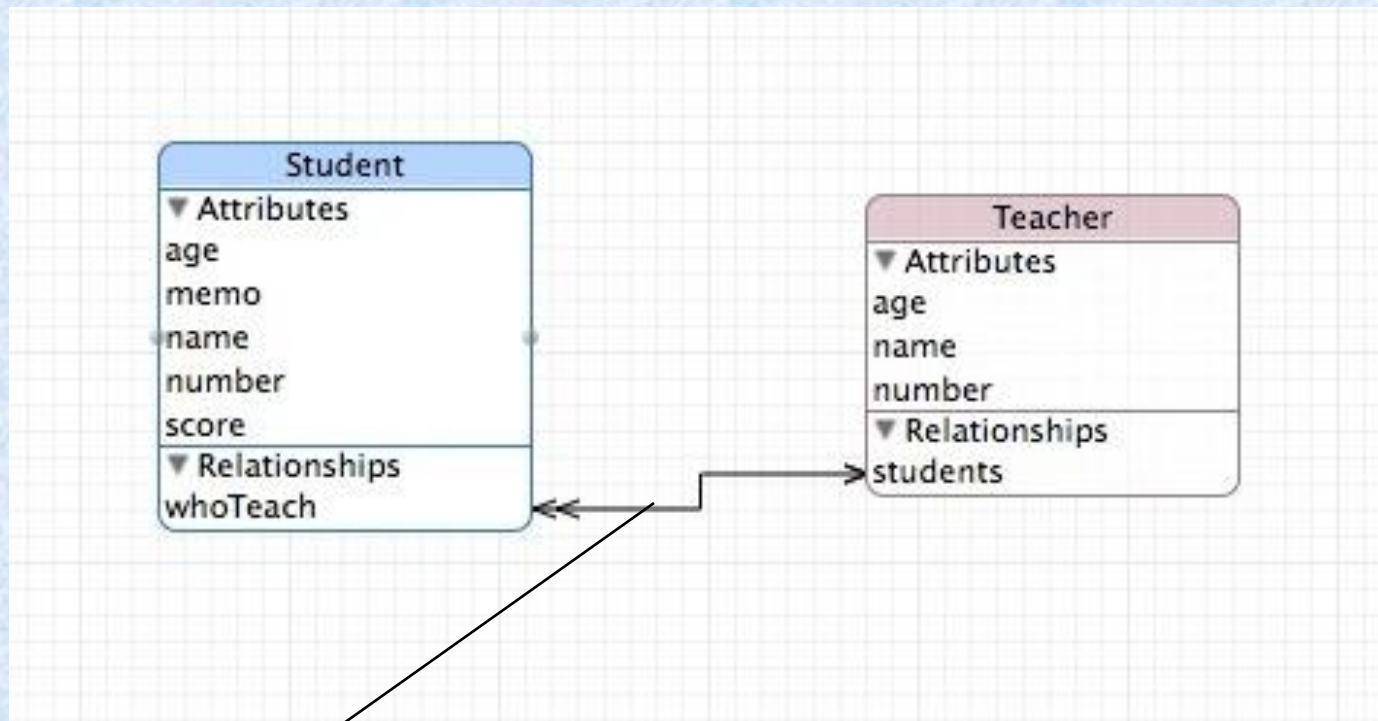


方法2、在已有工程中添加Data Model





.xcdatamodel文件编译后为.momd或.mom文件



添加关系，可以是一对多，多对一
直接Ctrl+drag拖动

Relationship

Name: students

Properties: ☐ Transient ☒ Optional

Destination: Student

Inverse: whoTeach

Delete Rule: Nullify

Type: ☒ To Many ☐ To One

Arrangement: ☐ Unbounded ☐ Minimum ☐ Maximum



databases

Filter by name

- CoreDemo1 (SQL)
 - Tables (4)
 - ZSTUD..
 - ZTEAC..
 - C...
 - I...
 - T...
 - Z_MET...
 - Z_PRI...
 - Views

ZSTUDENT (CoreDemo1)

Structure Data Constraints Indexes Triggers DDL

Grid view

Form view

	Z PK	Z ENT	Z OPT	Z AGE	WHOTEACI	Z SCORE	Z MEMO	'NAME	NUMBEI
1	7	1	1	23	2	91.0	IOS选课信息。	Jack	SA1002
2	8	1	1	22	2	90.0	IOS选课信息。	Mary	SA1003
3	9	1	1	21	2	85.0	IOS选课信息。	John	SA1004
4	10	1	1	22	2	99.0	IOS选课信息。	Ben	SA1005
5	11	1	2	23	2	89.0	IOS选课信息。	Suny	SA1006
6	12	1	10	22	3	90.0	IOS选课信息。	Tom	SA1001
7	16	1	1	23	2	90.0	IOS选课信息。	James	SA1007
8	17	1	1	21	2	88.0	IOS选课信息。	David	SA1010
9	18	1	1	19	2	75.0	IOS选课信息。	Lily	SA1008
10	19	1	2	20	2	81.0	IOS选课信息。	Emily	SA1009
11	20	1	1	22	2	95.0	IOS选课信息。	Jason	SA1011
12	21	1	1	19	2	85.0	IOS选课信息。	Alice	SA1012

ZTEACHER (CoreDemo1)

Structure Data Constraints Indexes Triggers DDL

Grid view

Form view

	Z PK	Z ENT	Z OPT	Z AGE	ZNAME	NUMBEI
1	1	2	11	50	Zhang San	ST00001
2	2	2	20	99	Bai Tian	ST00002
3	3	2	4	99	Bai Tian	ST00002
4	4	2	1	99	Bai Tian	ST00002



三、利用Core Data框架进行增、删、查和改

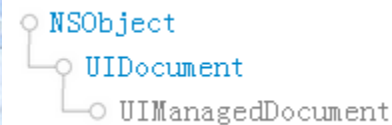
2、获得NSManagedObjectContext

有两种方法可以获得NSManagedObjectContext对象实例。

方法1、在创建工程时选中 ，则可以通过AppDelegate的managedObjectContext属性获得。

```
@interface AppDelegate : UIResponder <UIApplicationDelegate>
@property (strong, nonatomic) UIWindow *window;
@property (readonly, strong, nonatomic) NSManagedObjectContext *managedObjectContext;
@property (readonly, strong, nonatomic) NSManagedObjectModel *managedObjectModel;
@property (readonly, strong, nonatomic) NSPersistentStoreCoordinator *persistentStoreCoordinator;
- (void)saveContext;
- (NSURL *)applicationDocumentsDirectory;
@end
```

```
AppDelegate *coreDataManager=[[AppDelegate alloc] init];
self.context=[coreDataManager managedObjectContext];
```



方法2、使用UIManagedDocument

UIManagedDocument is a concrete subclass of UIDocument that integrates with Core Data. When you initialize a managed document, you specify the URL for the document location. The document object then creates a Core Data stack to use to access the document's persistent store using a managed object model from the application's main bundle.

实例化:

- initWithFileURL:

Returns a document object initialized with its file-system location.

Declaration

OBJECTIVE-C

```
- (instancetype)initWithFileURL:(NSURL *)url
```

Parameters

<i>url</i>	A file URL identifying the location in the application sandbox where document data is to be written. Passing in <i>nil</i> or an empty URL results in the throwing of an <code>NSInvalidArgumentException</code> .
------------	--

Return Value

A `UIDocument` object or *nil* if the object could not be created.



打开一个已有文档

- `openWithCompletionHandler:`

Opens a document asynchronously.

Declaration

OBJECTIVE-C

- (void)openWithCompletionHandler:(void (^)(BOOL success))completionHandler

Parameters

completionHandler

A block with code to execute after the open operation concludes. The block returns no value and has one parameter:

success

YES if the open operation succeeds, otherwise NO.

The block is invoked on the main queue.



创建一个文档

- `saveToURL:forSaveOperation:completionHandler:`

Saves document data to the specified location in the application sandbox.

Declaration

OBJECTIVE-C

```
- (void)saveToURL:(NSURL *)url  
    forSaveOperation:(UIDocumentSaveOperation)saveOperation  
    completionHandler:(void (^)(BOOL success))completionHandler
```

Parameters

<code>url</code>	The file URL identifying the location in the application sandbox to write the document data to. Typically, this is the URL obtained from the <code>fileURL</code> property.		
<code>saveOperation</code>	A constant that indicates whether the document file is being written the first time or whether it is being overwritten. See Document Save Operation for details.		
<code>completionHandler</code>	A block with code that is executed when the save operation concludes. The block returns no value and has one parameter: <table border="1" data-bbox="540 1199 1700 1270"><tr><td><code>success</code></td><td>YES if the save operation succeeds, otherwise NO.</td></tr></table> This block is invoked on the calling queue.	<code>success</code>	YES if the save operation succeeds, otherwise NO.
<code>success</code>	YES if the save operation succeeds, otherwise NO.		

```
enum {  
    UIDocumentSaveForCreating ,  
    UIDocumentSaveForOverwriting  
};  
typedef NSInteger UIDocumentSaveOperation;
```

注意线程



使用UIManagedDocument

```
self.document = [[UIManagedDocument alloc] initWithFileURL:(URL *)url];
if ([[NSFileManager defaultManager] fileExistsAtPath:[url path]]) {
    [document openWithCompletionHandler:^(BOOL success) {
        if (success) [self documentIsReady];
        if (!success) NSLog(@"couldn't open document at %@", url);
    }];
} else {
    [document saveToURL:url forSaveOperation:UIDocumentSaveForCreating
    completionHandler:^(BOOL success) {
        if (success) [self documentIsReady];
        if (!success) NSLog(@"couldn't create document at %@", url);
    }];
}
```

```
- (void)documentIsReady
{
    if (self.document.documentState == UIDocumentStateNormal) {
        NSManagedObjectContext *context = self.document.managedObjectContext;
        // start doing Core Data stuff with context
    }
}
```

```
enum {
    UIDocumentStateNormal = 0,
    UIDocumentStateClosed = 1 << 0,
    UIDocumentStateInConflict = 1 << 1,
    UIDocumentStateSavingError = 1 << 2,
    UIDocumentStateEditingDisabled = 1 << 3 }; typedef NSInteger UIDocumentState;
```




保存已有文档

UIManagedDocuments可以自动地异步保存数据对象至持久层，如果需要立即保存，可以使用SaveToUrl方法。

```
[document saveToURL:document.fileURL  
forSaveOperation:UIDocumentSaveForOverwriting  
completionHandler:^(BOOL success) { /* block to execute when save is done */ }];
```




关闭打开的文档

在ARC下，当没有强指针引用UIManagedDocument时，文档自动关闭。也可以通过下面直接方法关闭。

`- closeWithCompletionHandler:`

Asynchronously closes the document after saving any changes.

Declaration

OBJECTIVE-C

`- (void)closeWithCompletionHandler:(void (^)(BOOL success))completionHandler`

Parameters

completionHandler

A block with code to execute after the save-and-close operation concludes. The block returns no value and has one parameter:

success

YES if any save operation succeeds, otherwise NO.

The block is invoked on the main queue.

```
[self.document closeWithCompletionHandler:^(BOOL success) {  
    if (!success) NSLog(@"failed to close document %@", self.document.localizedName);  
}];
```



注意：可以使用多个UIManagedDocument的实例对象访问一个文档。但在保存的时候要注意数据可能会被覆盖。同时，当一个文档实例对象修改了底层文档后，其它文档并不会知道。需要重新加载文档的内容后才能更新。

可以通过广播实现数据的同步。



NSNotificationCenterDidSaveNotification广播用来通知数据已经改变。

```
- (void)viewDidAppear:(BOOL)animated
{
    [super viewDidAppear:animated];
    [center addObserver:self
                selector:@selector(contextChanged:)
                name:NSManagedObjectContextDidSaveNotification
                object:document.managedObjectContext]; // don't pass nil here!
}
- (void)viewWillDisappear:(BOOL)animated
{
    [center removeObserver:self
                name:NSManagedObjectContextDidSaveNotification
                object:document.managedObjectContext];
    [super viewWillDisappear:animated];
}
```

```
- (void)contextChanged:(NSNotification *)notification
{
    // The notification.userInfo object is an NSDictionary with the following keys:
    NSInsertedObjectsKey // an array of objects which were inserted
    NSUpdatedObjectsKey  // an array of objects whose attributes changed
    NSDeletedObjectsKey  // an array of objects which were deleted
}
```




三、利用Core Data框架进行增、删、查和改

3、查询

```
NSFetchRequest *request=[[NSFetchRequest alloc] init];
request.fetchLimit=100;
request.fetchBatchSize=20;
request.sortDescriptors=@[[NSSortDescriptor sortDescriptorWithKey:sortDesc ascending:asc]];
request.predicate=[NSPredicate predicateWithFormat:@"name contains %@",ps];
NSEntityDescription *entity=[NSEntityDescription entityForName:entityname inManagedObjectContext:
    self.context];
request.entity=entity;
NSError *error;
NSArray *arrs=[self.context executeFetchRequest:request error:&error];
if(error)
    NSLog(@"无法获取数据, %@",error);
return arrs;
```




三、利用Core Data框架进行增、删、查和改

4、新增

```
Student *stu;
stu=[NSEntityDescription insertNewObjectForEntityForName:@"Student" inManagedObjectContext:self.
    context];
stu.name=self.TxtName.text;
stu.number=self.TxtNumber.text;
stu.age=[NSNumber numberWithInt:[self.TxtAge.text floatValue]];
stu.score= [NSNumber numberWithInt:[self.TxtScore.text floatValue]];
stu.memo=self.TxtMemo.text;
stu.whoTeach=self.teacher;
NSError *errorstudent;
if (![self.context save:&errorstudent]) {
    NSLog(@"保存时出错: %@",errorstudent);
}
```



三、利用Core Data框架进行增、删、查和改

5、删除

```
if (editingStyle == UITableViewCellEditingStyleDelete) {  
    [self.context deleteObject:self.students[indexPath.row]];  
    [self.students removeObjectAtIndex:indexPath.row];  
    [tableView deleteRowsAtIndexPaths:@[indexPath] withRowAnimation:UITableViewRowAnimationFade];  
    NSError *err;  
    [self.context save:&err];  
}
```




三、利用Core Data框架进行增、删、查和改

5、修改

```
stu= self.students[self.indexPath.row];
stu.name=self.TxtName.text;
stu.number=self.TxtNumber.text;
stu.age=[NSNumber numberWithInt:[self.TxtAge.text floatValue]];
stu.score= [NSNumber numberWithInt:[self.TxtScore.text floatValue]];
stu.memo=self.TxtMemo.text;
stu.whoTeach=self.teacher;
NSError *errorstudent;
if (![self.context save:&errorstudent]) {
    NSLog(@"保存时出错: %@",errorstudent);
}
```



三、利用Core Data框架进行增、删、查和改

5、修改

```
stu= self.students[self.indexPath.row];  
stu.name=self.TxtName.text;  
stu.number=self.TxtNumber.text;  
stu.age=[NSNumber numberWithInt:[self.TxtAge.text floatValue]];  
stu.score= [NSNumber numberWithInt:[self.TxtScore.text floatValue]];  
stu.memo=self.TxtMemo.text;  
stu.whoTeach=self.teacher;  
NSError *errorstudent;  
if (![self.context save:&errorstudent]) {  
    NSLog(@"保存时出错: %@",errorstudent);  
}
```