访问： 25211次
积分： 626
等级： BLOG 3
排名： 千里之外

原创： 34篇 转载： 10篇
译文： 2篇 评论： 4条

原 CoreData 从入门到精通 二 数据的增删改查

标签： CoreData iOS
2016-12-09 16:06 477人阅读 评论(0)

分类：
CoreData（5） iOS（15） ObjectiveC（9）

在上篇博客中，讲了数据模型和 CoreData 栈的创建，那下一步就是对数据的操作了。和数据库一样，CoreData 里的操作也无非是增删改查。下面我们将逐步讲解在 CoreData 中进行增删改查的方式。

基本的增删改查

插入条目

先来看一下插入条目的方式，在插入之前，我们需要先创建要插入的数据， 使用 `NSEntityDesctiption` 类的

```
+ (__kindof NSManagedObject *)insertNewObjectForEntityForName:(NSString *)entityN
```

方法来创建一个新的 `NSManagedObject` 对象，入参分别是 `entityName` 和 `managedObjectContext`， `entityName` 也就是实体类的名字，例如，我要插入一条新的 `Student` 字段， `entityName` 就是 @”Student”;

`context` 是 `NSManagedObjectContext` 对象，新增的实体类对象会添加到对应的 `context` 上下文对象中。这个方法返回的是一个 `NSManagedObject` 实例，可以根据具体情况转换成相应的子类:

```
1  Student *student = [NSEntityDescription insertNewObjectForEntityForName:
2
3  student.studentName = @"小明";
4  student.studentId = 1;
5  student.studentAge = 20;
6
7  NSError *error;
8  [self.context save:&error];
```

调用 `save` 方法时，可以传入一个 `NSError` 的指针，如果数据保存出错的话，错误信息会保存到 `error` 里，这也是 **`Objective-C`** 里通常处理错误的方式。

查询条目

从数据库中查询数据，会用到三个类：`NSFetchRequest`，`NSPredicate`，`NSSortDescriptor`，分别说一下这三个类的作用：

- NSFetchRequest — fetchRequest 代表了一条查询请求，相当于 SQL 中的 SELECT 语句
- NSPredicate — predicate 翻译过来是谓词的意思，它可以指定一些查询条件，相当于 SQL 中的 WHERE 子句，有关 NSPredicate 的用法，可以看我之前写过的一篇文章：使用 NSPredicate 进行数据库查询
- NSSortDescriptor — sortDescriptor 是用来指定排序规则的，相当于 SQL 中的 ORDER BY 子句

在 `NSFetchRequest` 中有两个属性 `predicate`、`sortDescriptors`，就是用来指定查询的限制条件的。其中 `sortDescriptors` 是一个 `NSSortDescriptor` 的数组，也就是可以给一个查询指定多个排序规则，这些排序规则的优先级就是它们在数组中的位置，数组前面的优先级会比后面的高。除此之外，`NSFetchRequest` 还有下面这些属性

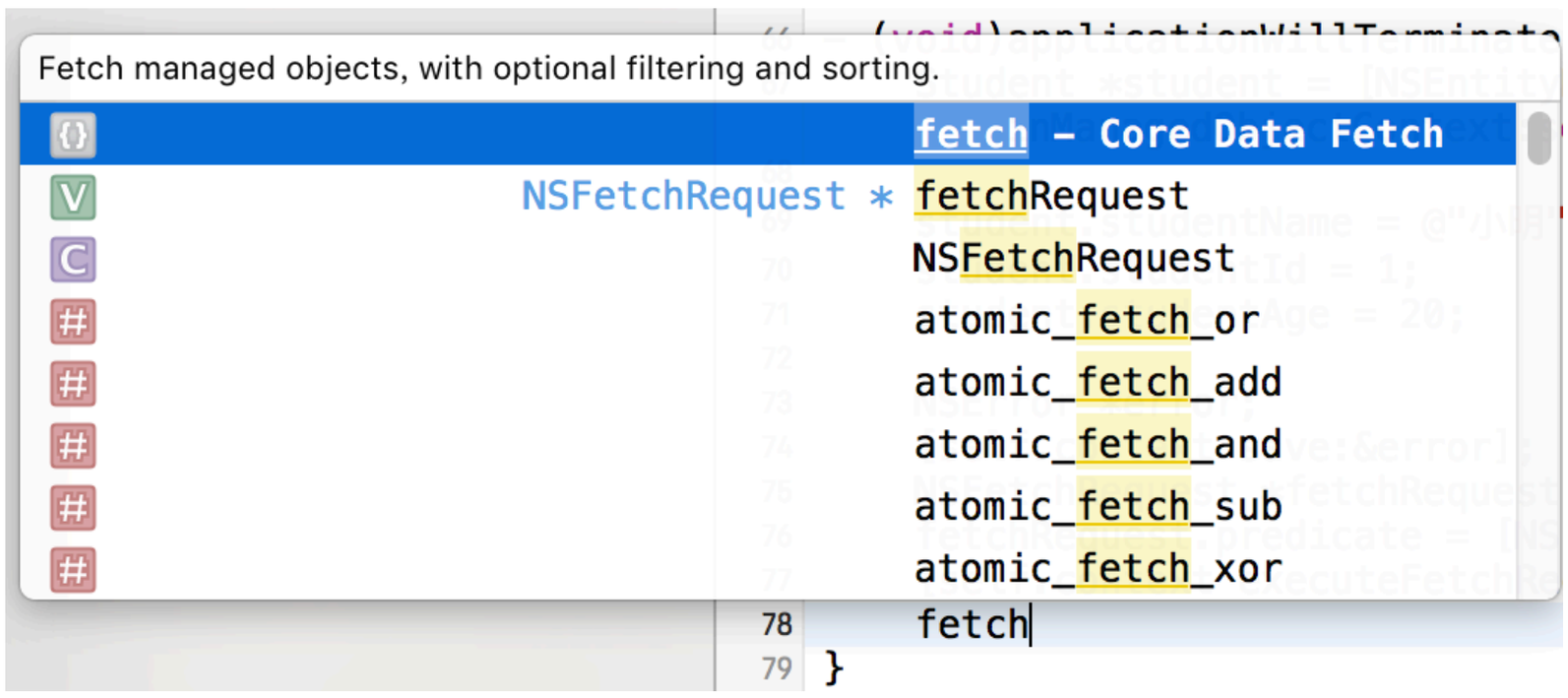
- fetchLimit — 指定结果集中数据的最大条目数，相当于 SQL 中的 LIMIT 子句
- fetchOffset — 指定查询的偏移量，默认为 0
- fetchBatchSize — 指定批处理查询的大小，设置了这个属性后，查询的结果集会分批返回
- entityName/entity — 指定查询的数据表，相当于 SQL 中的 FROM 语句
- propertiesToGroupBy — 指定分组规则，相当于 SQL 中的 GROUP BY 子句
- propertiesToFetch — 指定要查询的字段，默认会查询全部字段

配置好 `NSFetchRequest` 对象后，需要调用 `NSManagedObjectContext` 的

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

来执行查询，返回的数组就是查询出的结果集。

Xcode 中预置了用来创建 fetchRequest 的code snippet，键入 `fetch`，就会出现相应的提示：



创建出来的代码是这样子的：

```objc
NSFetchRequest *fetchRequest = [[NSFetchRequest alloc] init];
NSEntityDescription *entity = [NSEntityDescription entityForName:@"Entity name"
    inManagedObjectContext:context];
[fetchRequest setEntity:entity];
// Specify criteria for filtering which objects to fetch
NSPredicate *predicate = [NSPredicate predicateWithFormat:@"format string",
    arguments];
[fetchRequest setPredicate:predicate];
// Specify how the fetched objects should be sorted
NSSortDescriptor *sortDescriptor = [[NSSortDescriptor alloc] initWithKey:@"key"
                                                               ascending:YES];
[fetchRequest setSortDescriptors:[NSArray arrayWithObjects:sortDescriptor, nil]];

NSError *error = nil;
NSArray *fetchedObjects = [context executeFetchRequest:fetchRequest error:&error];
if (fetchedObjects == nil) {
    Error handling code
}
```

基本上常用到的代码都在这里了。当然我们也可以自己来手写出来，下面就是一个最简单的查询语句：

```objc
NSFetchRequest *fetchRequest = [Student fetchRequest]; // 自动创建的 NSMa
    // 也可以使用这种方式：[NSFetchRequest fetchRequestWithEntityName:@"Stu

fetchRequest.predicate = [NSPredicate predicateWithFormat:@"studentAge >

NSArray<NSSortDescriptor *> *sortDescriptors = @[[NSSortDescriptor sortD

fetchRequest.sortDescriptors = sortDescriptors;
//
NSArray<Student *> *students = [self.context executeFetchRequest:fetchRe
```

删除条目

简单的删除条目还是比较简单的，在上一步里查询出来后，只需调用 `NSManagedObjectContext` 的 `- (void)deleteObject:(NSManagedObject *)object;` 方法来删除一个条目。例如，将上面查询出来的 `students` 全部删除，可以这么写：

```objc

for (Student *student in students) {
    [self.context deleteObject:student];
}
[self.context save:nil]; // 最后不要忘了调用 save 使操作生效。
```

更新条目

还是在查询出的 `students` 数组的基础上，如果要更新里面的字段，可以遍历这个数组，依次修改数组里元素的字段：

```objc
for (Student *student in students) {
        student.studentName = @"newName";
    }

[self.context save:nil];
```

增删改查进阶

批量插入

简单的批量插入和插入单条数据一样，只是在所有的数据都插入只有才调用 `[context save];` 来

保存。下面是简单的示例代码：

```
for (NSUInteger i = 0; i < 1000; i++) {
    Student *newStudent = [NSEntityDescription insertNewObjectForEntityF

    int16_t stuId = arc4random_uniform(9999);
    newStudent.studentName = [NSString stringWithFormat:@"student-%d", s
    newStudent.studentId = stuId;
    newStudent.studentAge = arc4random_uniform(10) + 10;
}

[self.context save:nil];
```

批量更新

这里讲的批量更新方式，用到的是集合类型中的 KVC 特性。这是什么呢？就是在 `NSArray` 这样的集合类型里，可以调用它的 `[setValue: forKeyPath:]` 方法来更新这个数组中所有元素所对应的 keypath。例如想要将上面查询出来的 `students` 数组里所有元素的 `studentName` 属性都修改成 `@"anotherName"`，就可以这么来写：

```
[students setValue:@"anotherName" forKeyPath:@"studentName"];
```

除了这种批量更新的方式，还有下面将要讲的 `NSBatchUpdateRequest` 也可以进行批量更新，不妨接着往下看。

NSBatchUpdateRequest 批量更新

`NSBatchUpdateRequest` 是在 **iOS** 8, macOS 10.10 之后新添加的 API，它是专门用来进行批量更新的。因为用上面那种方式批量更新的话，会存在一个问题，就是更新前需要将要更新的数据，查询出来，加载到内存中；这在数据量非常大的时候，假如说要更新十万条数据，就比较麻烦了，因为对于手机这种内存比较小的设备，直接加载这么多数据到内存里显然是不可能的。解决办法就是每次只查询出读取一小部分数据到内存中，然后对其进行更新，更新完之后，再更新下一批，就这样分批来处理。但这显然不是高效的解决方案。

于是就有了 `NSBatchUpdateRequest` 这个 API。它的工作原理是不加载到内存里，而是直接对本地数据库中数据进行更新。这就避免了内存不足的问题；但同时，由于是直接更新数据库，所以内存中的 `NSManagedObjectContext` 不会知道数据库的变化，解决办法是调用

`NSManagedObjectContext` 的

```
+ (void)mergeChangesFromRemoteContextSave:(NSDictionary*)changeNotificationData i
```

方法来告诉 context，有哪些数据更新了。

下面来看一下 `NSBatchUpdateRequest` 的用法。

1. 创建

```
// 根据 entity 创建
NSBatchUpdateRequest *updateRequest = [[NSBatchRequest alloc] initW
// 根据 entityName 创建
NSBatchUpdateRequest *updateRequest = [[NSBatchUpdateRequest alloc]
```

2. predicate

`NSBatchUpdateRequest` 的 `predicate` 用来指定更新条件，例如这里指定更新 studentAge 等于 20 的学生

```
1  updateRequest.predicate = [NSPredicate predicateWithFormat:@"studer
```

3. propertiesToUpdate

`propertiesToUpdate` 属性是一个字典，用它来指定需要更新的字段，字典里的 key 就是要更新的字段名，value 就是要设置的新值。因为 **objective-c** 字典里只能存储对象类型，所以如果字段基本数据类型的的话，需要转换成 `NSNumber` 对象。

```
1  updateRequest.propertiesToUpdate = @{@"studentName" : @"anotherName
```

4. resultType

`resultType` 属性是 `NSBatchUpdateRequestResultType` 类型的枚举，用来指定返回的数据类型。这个枚举有三个成员：

- `NSStatusOnlyResultType` — 返回 BOOL 结果，表示更新是否执行成功
- `NSUpdatedObjectIDsResultType` — 返回更新成功的对象的 ID，是 NSArray\

NSBatchDeleteRequest 批量删除

`NSBatchDeleteRequest` 的用法和 `NSBatchUpdateRequest` 很相似，不同的是 `NSBatchDeleteRequest` 需要指定 `fetchRequest` 属性来进行删除；而且它是 **ios** 9 才添加进来的，和 `NSBatchUpdateRequest` 的适用范围不一样，下面看一下示例代码：

```
1  NSFetchRequest *deleteFetch = [Student fetchRequest];
2
3  deleteFetch.predicate = [NSPredicate predicateWithFormat:@"studentAge ==
4
5  NSBatchDeleteRequest *deleteRequest = [[NSBatchDeleteRequest alloc] init
6  deleteRequest.resultType = NSBatchDeleteResultTypeObjectIDs;
7
8  NSBatchDeleteResult *deleteResult = [self.context executeRequest:deleteR
9  NSArray<NSManagedObjectID *> *deletedObjectIDs = deleteResult.result;
10
11 NSDictionary *deletedDict = @{NSDeletedObjectsKey : deletedObjectIDs};
12 [NSManagedObjectContext mergeChangesFromRemoteContextSave:deletedDict in
```

好了，CoreData 中的增删改查就讲完了，下篇文章将会介绍 CoreData Model 中的 relationships 实现多表关联的用法。

- 上一篇 CoreData 从入门到精通 （一） 数据模型 + CoreData 栈的创建
- 下一篇 CoreData 从入门到精通（三）关联表的创建

相关文章推荐

- CoreData 从入门到精通 （一） 数据模型 + CoreDa…
- iOS：WKWebView与UIWebView的区别
- 数组过滤与NSPredicate类的用法
- NSPredicate 谓词
- iOS 常用公共方法（一）
- SQLite学习笔记五:Order By，Group By，Having，…
- ios开发中是CoreData的一些详细介绍。
- 让CoreData更简单些
- CoreData使用详解
- iOS常用公共方法

猜你在找

- 【直播】机器学习&数据挖掘7周实训--韦玮
- 【直播】3小时掌握Docker最佳实战-徐西宁
- 【直播】计算机视觉原理及实战--屈教授
- 【直播】机器学习之矩阵--黄博士
- 【直播】机器学习之凸优化--马博士
- 【套餐】系统集成项目管理工程师顺利通关--徐朋
- 【套餐】机器学习系列套餐（算法+实战）--唐宇迪
- 【套餐】微信订阅号+服务号Java版 v2.0--翟东平
- 【套餐】微信订阅号+服务号Java版 v2.0--翟东平
- 【套餐】Javascript 设计模式实战--曾亮

查看评论

暂无评论

您还没有登录,请[登录]或[注册]