CoreData

Trinh Minh Cuong

Many ways to store data

- As text file
- As XML file
- As property list
- Use SQLite directly
- CoreData
- iCloud

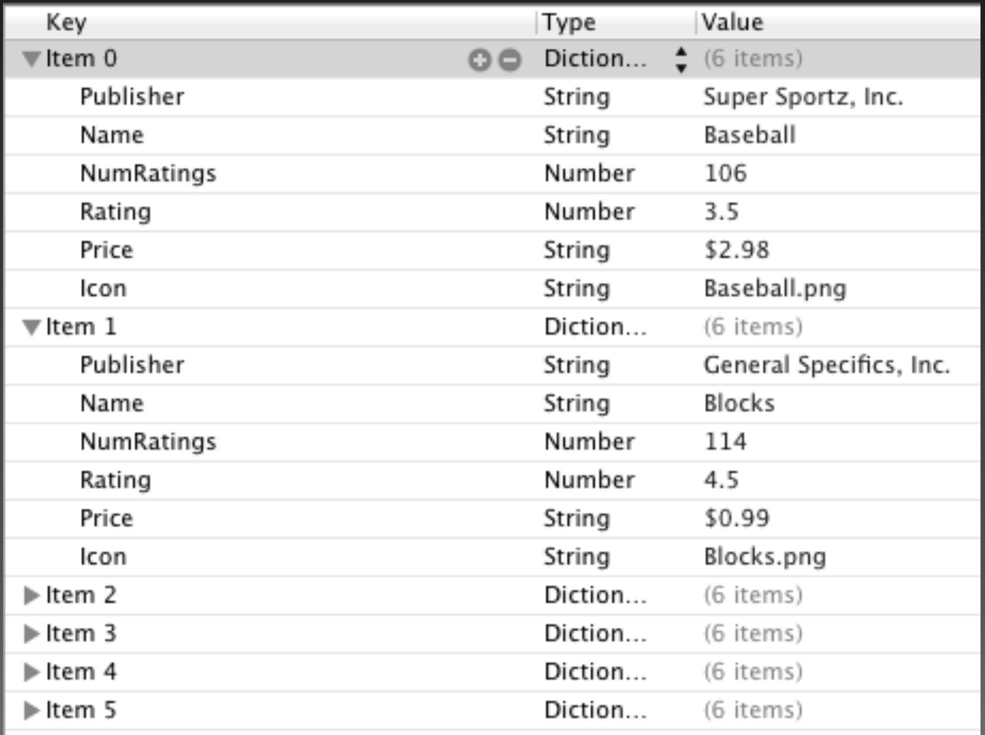
Read text file from main bundle

```
NSString *pathname = [[NSBundle mainBundle]
pathForResource:@"crayons" ofType:@"txt" inDirectory:@"/"];

NSArray *rawCrayons = [[NSString
stringWithContentsOfFile:pathname encoding:NSUTF8StringEncoding
error:nil] componentsSeparatedByString:@"\n"];
```

See Day 06: 05-Sectioned Tables

Read Property List

```
NSString *dataPath = [[NSBundle mainBundle]
pathForResource:@"Data" ofType:@"plist"];
self.data = [NSArray arrayWithContentsOfFile:dataPath];
```



Key	Type	Value
▼ Item 0	Diction...	(6 items)
Publisher	String	Super Sportz, Inc.
Name	String	Baseball
NumRatings	Number	106
Rating	Number	3.5
Price	String	$2.98
Icon	String	Baseball.png
▼ Item 1	Diction...	(6 items)
Publisher	String	General Specifics, Inc.
Name	String	Blocks
NumRatings	Number	114
Rating	Number	4.5
Price	String	$0.99
Icon	String	Blocks.png
▶ Item 2	Diction...	(6 items)
▶ Item 3	Diction...	(6 items)
▶ Item 4	Diction...	(6 items)
▶ Item 5	Diction...	(6 items)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<array>
        <dict>
                <key>Publisher</key>
                <string>Super Sportz, Inc.</string>
                <key>Name</key>
                <string>Baseball</string>
                <key>NumRatings</key>
                <integer>106</integer>
                <key>Rating</key>
                <real>3.5</real>
                <key>Price</key>
                <string>$2.98</string>
                <key>Icon</key>
                <string>Baseball.png</string>
        </dict>
```

Data.plist

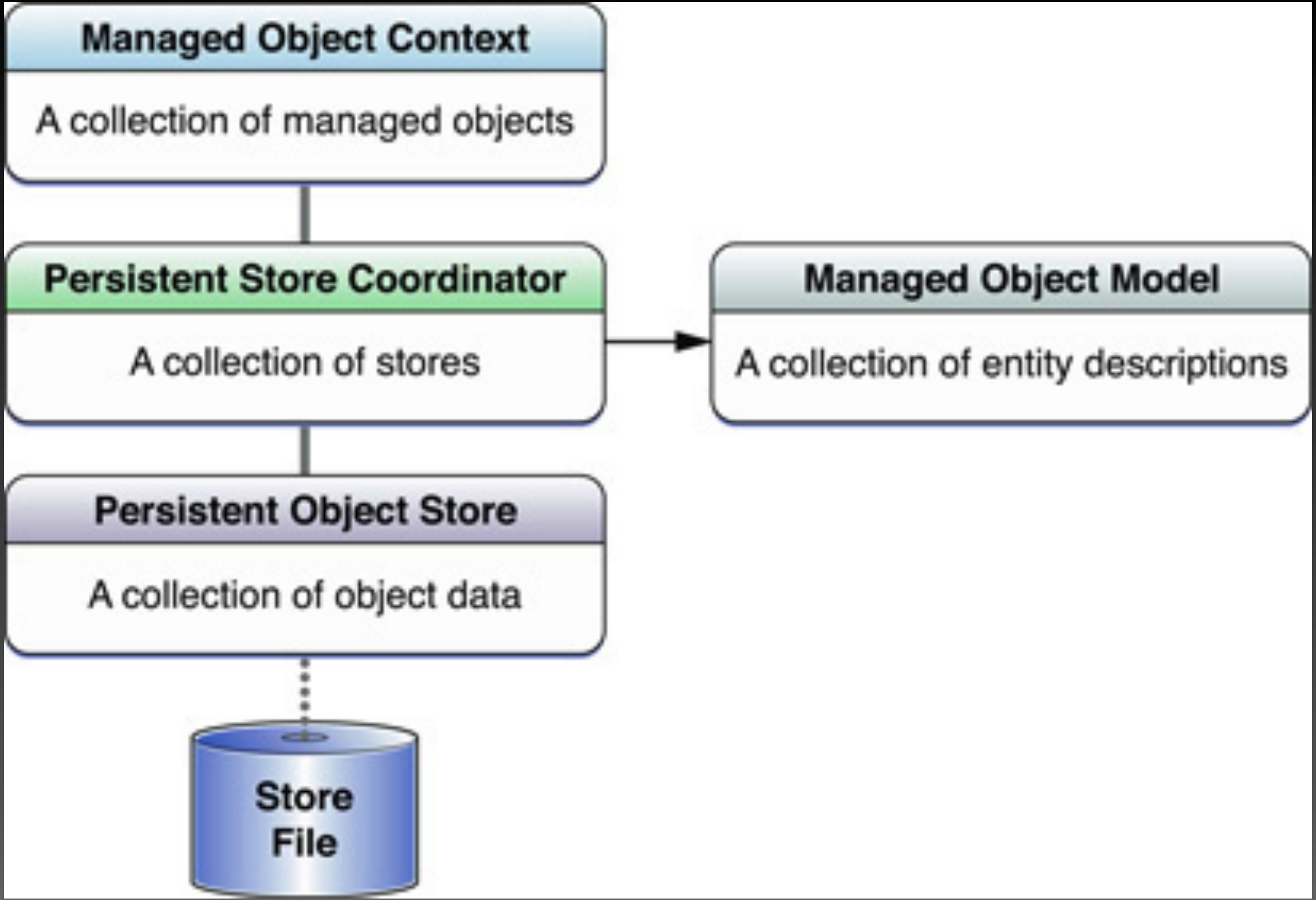
Core Data features #1

- Store objects in external storage
- Relationship maintenance
- Lazy loading
- Validation of property values

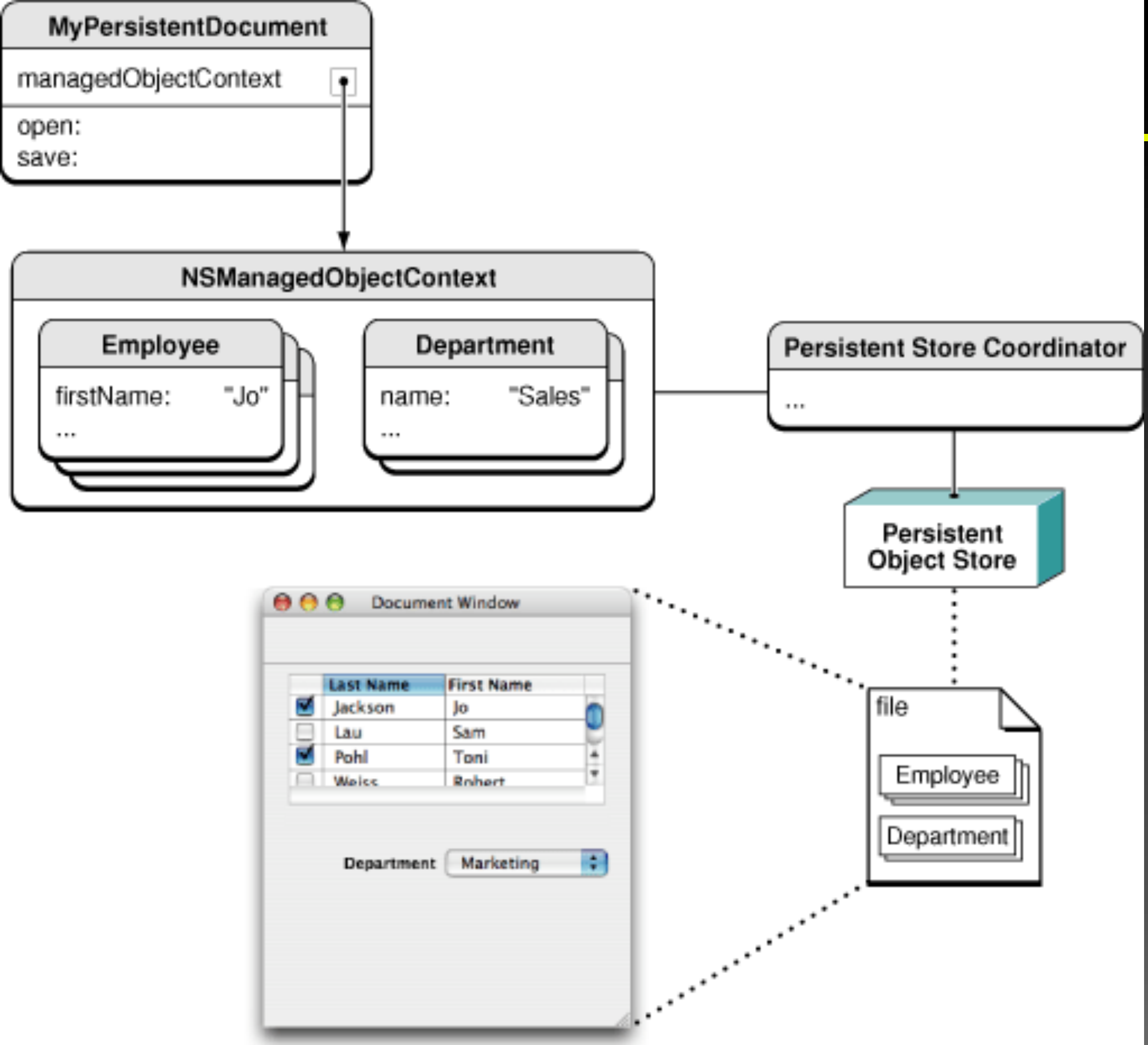
Core Data helps applications on all our platforms manage their data

Core Data features #2

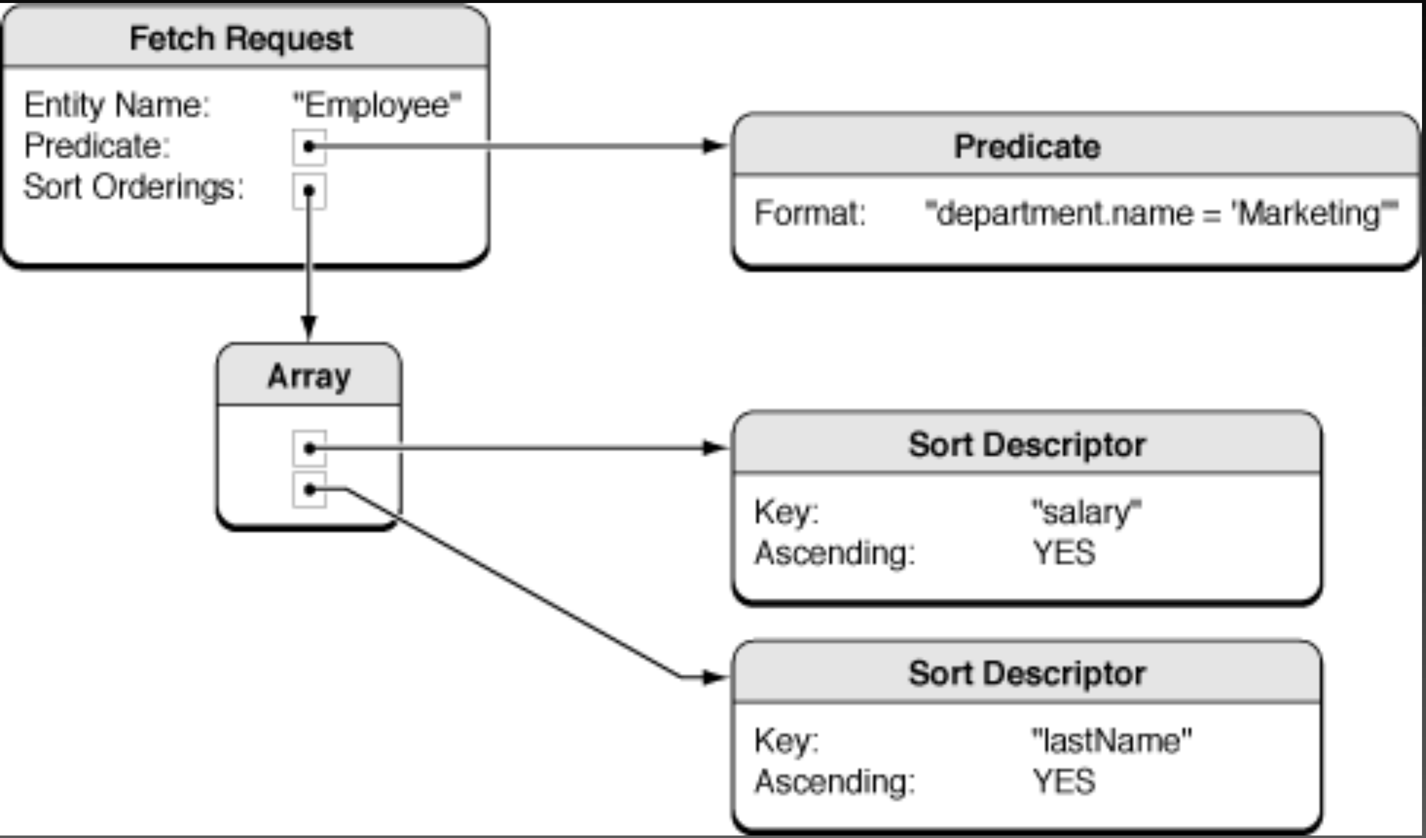
- Schema Integration
- Support key-value coding and key-value observing
- Grouping, filtering, organizing
- Change tracking and undo support
- Query with NSPredicate

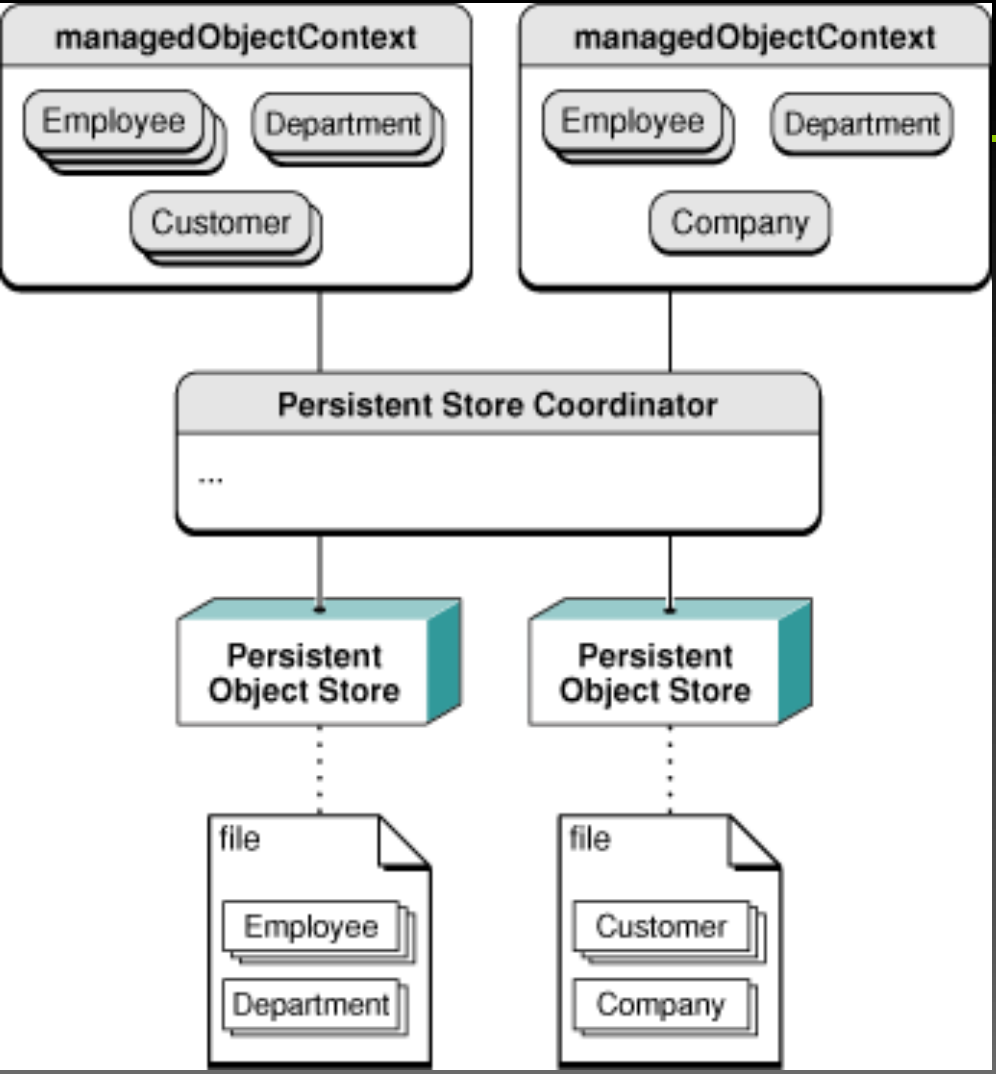

Managed Object Context
A collection of managed objects
Persistent Store Coordinator
A collection of stores
Managed Object Model
A collection of entity descriptions
Persistent Object Store
A collection of object data
Store File

Managed Object Context



Fetch Requests





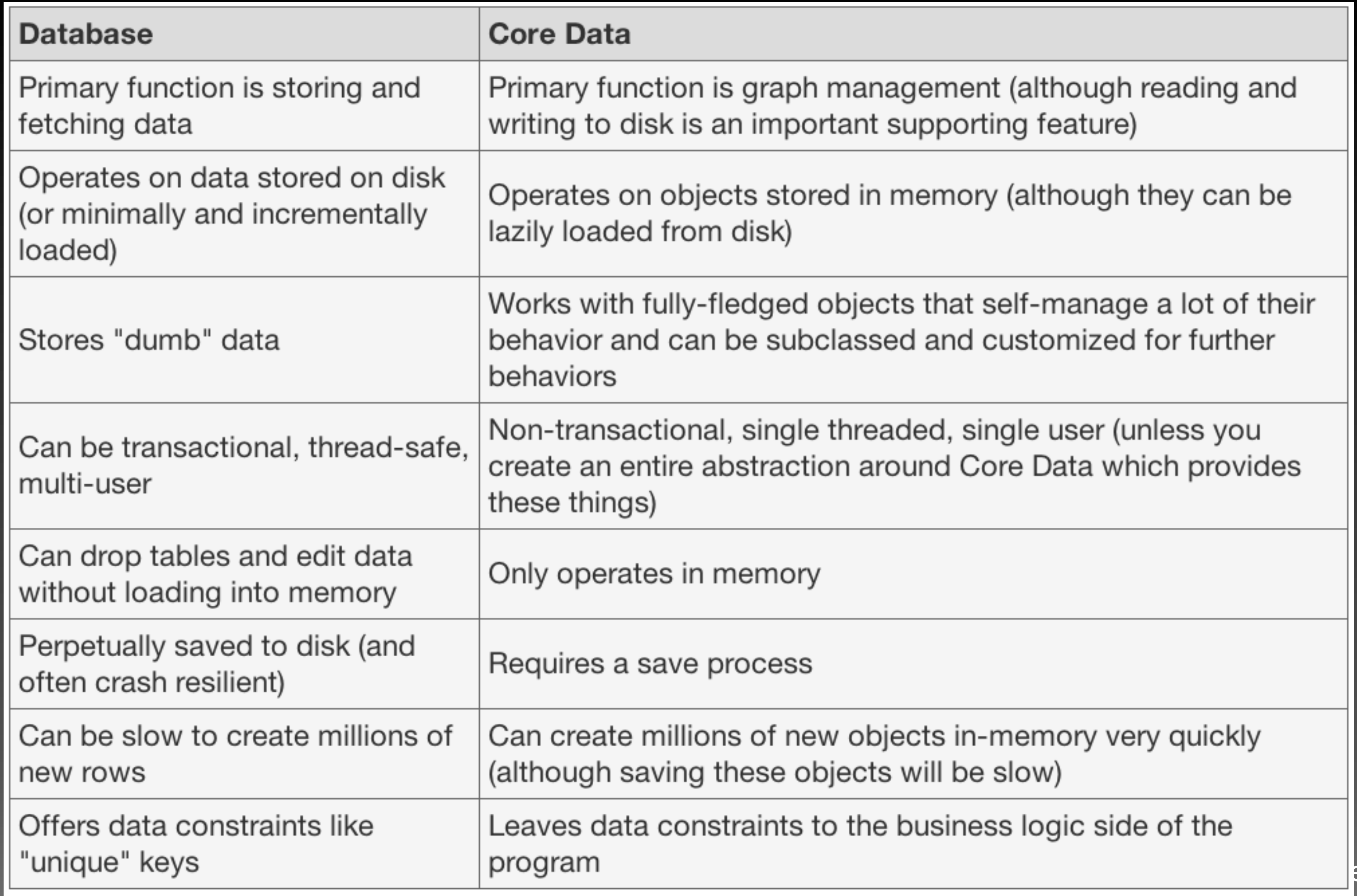
Persistent Store Coordinator

A persistent store coordinator associates persistent object stores and a managed object model

CoreData supports 4 types of persistance storage

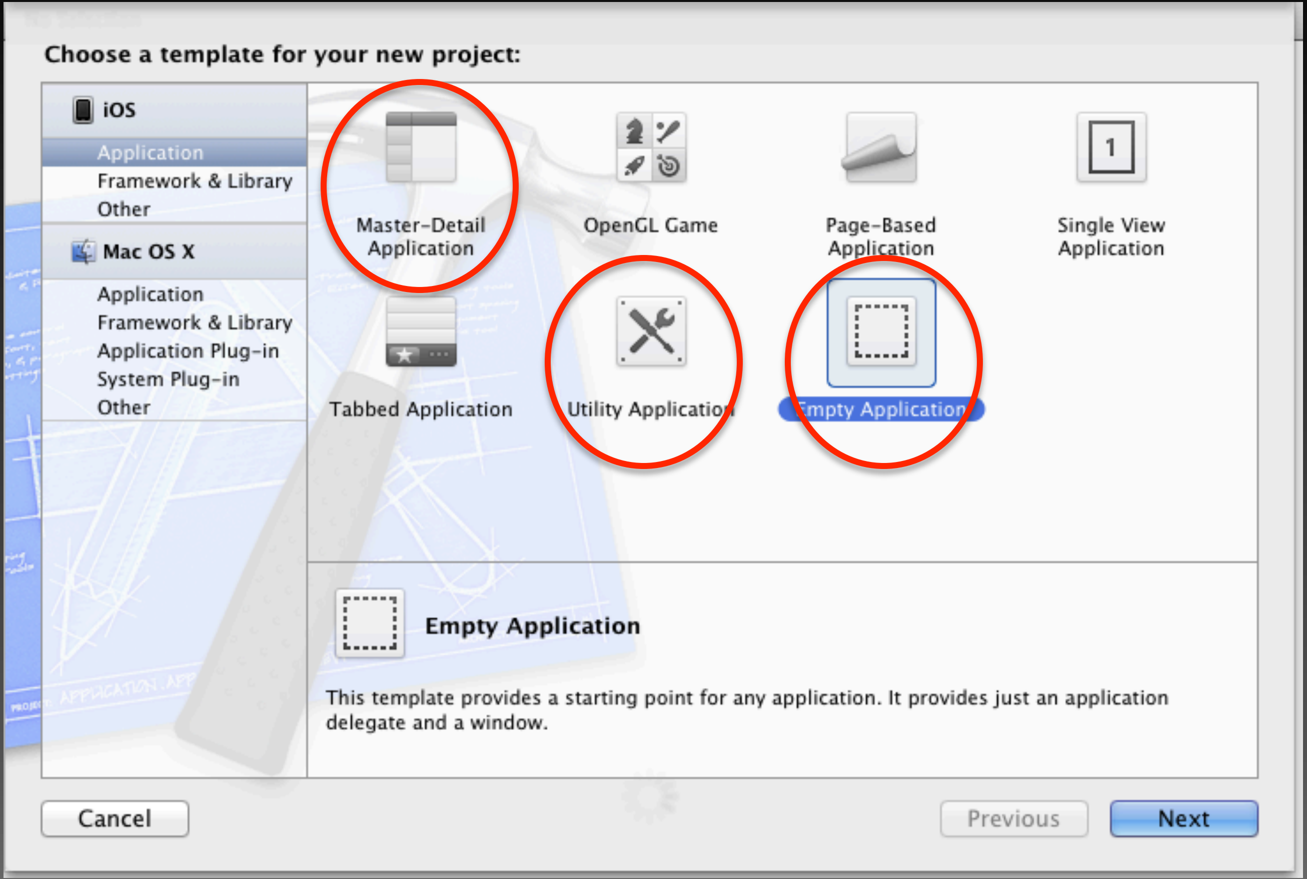
```
COREDATA_EXTERN NSString * const NSSQLiteStoreType
NS_AVAILABLE(10_4, 3_0);

COREDATA_EXTERN NSString * const NSXMLStoreType
NS_AVAILABLE(10_4, NA);

COREDATA_EXTERN NSString * const NSBinaryStoreType
NS_AVAILABLE(10_4, 3_0);

COREDATA_EXTERN NSString * const NSInMemoryStoreType
NS_AVAILABLE(10_4, 3_0);
```

http://cocoawithlove.com/2010/02/differences-between-core-data-and.html



Database	Core Data
Primary function is storing and fetching data	Primary function is graph management (although reading and writing to disk is an important supporting feature)
Operates on data stored on disk (or minimally and incrementally loaded)	Operates on objects stored in memory (although they can be lazily loaded from disk)
Stores "dumb" data	Works with fully-fledged objects that self-manage a lot of their behavior and can be subclassed and customized for further behaviors
Can be transactional, thread-safe, multi-user	Non-transactional, single threaded, single user (unless you create an entire abstraction around Core Data which provides these things)
Can drop tables and edit data without loading into memory	Only operates in memory
Perpetually saved to disk (and often crash resilient)	Requires a save process
Can be slow to create millions of new rows	Can create millions of new objects in-memory very quickly (although saving these objects will be slow)
Offers data constraints like "unique" keys	Leaves data constraints to the business logic side of the program

Three project templates support CoreData



Empty Application

This template provides a starting point for any application. It provides just an application delegate and a window.

Initialize CoreData in AppDelegate

AppDelegate.h

```
@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
```

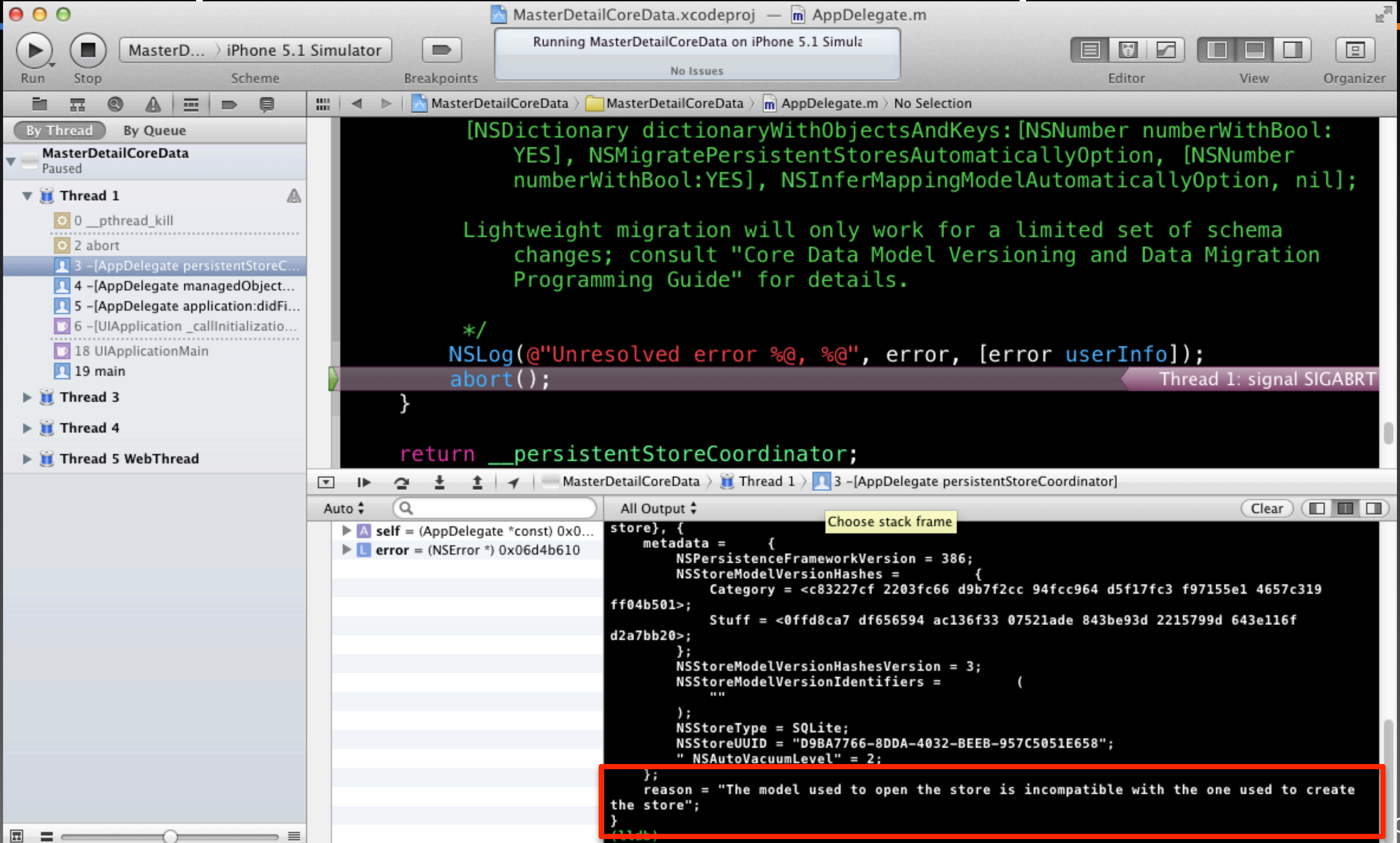
Some getter functions

```
- (NSManagedObjectContext *)managedObjectContext

- (NSManagedObjectModel *)managedObjectModel

- (NSPersistentStoreCoordinator
*)persistentStoreCoordinator
```

-(NSPersistentStoreCoordinator*) persistentStoreCoordinator

```
NSURL *storeURL = [[self applicationDocumentsDirectory]
URLByAppendingPathComponent:@"MasterDetailCoreData.sqlite"];

__persistentStoreCoordinator = [[NSPersistentStoreCoordinator
alloc] initWithManagedObjectModel:[self managedObjectModel]];

[__persistentStoreCoordinator
addPersistentStoreWithType:NSSQLiteStoreType configuration:nil
URL:storeURL options:nil error:&error]
```

Lỗi phát sinh khi model và sqlite vênh nhau



MasterDetailCoreData.xcodeproj — AppDelegate.m
Run
Stop
MasterD... iPhone 5.1 Simulator
Scheme
Breakpoints
Running MasterDetailCoreData on iPhone 5.1 Simula
No Issues
Editor
View
Organizer
MasterDetailCoreData › MasterDetailCoreData › AppDelegate.m › No Selection
By Thread
By Queue
MasterDetailCoreData
Paused
Thread 1
0 __pthread_kill
2 abort
3 -[AppDelegate persistentStoreC...
4 -[AppDelegate managedObject...
5 -[AppDelegate application:didFi...
6 -[UIApplication _callInitializatio...
18 UIApplicationMain
19 main
Thread 3
Thread 4
Thread 5 WebThread
Thread 1: signal SIGABRT
MasterDetailCoreData › Thread 1 › 3 -[AppDelegate persistentStoreCoordinator]
Auto
All Output
Clear
Choose stack frame
self = (AppDelegate *const) 0x0...
error = (NSError *) 0x06d4b610

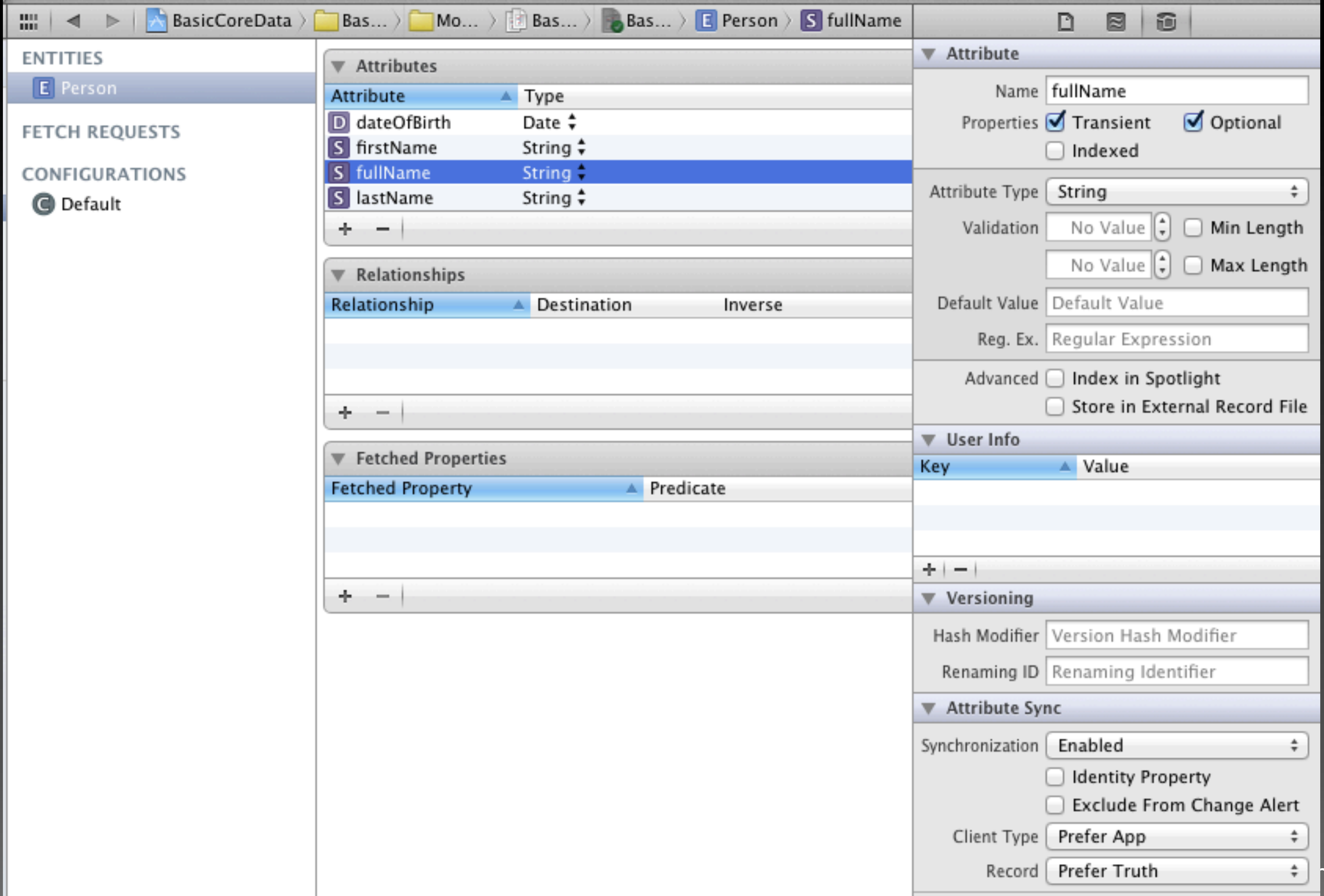
```
        [NSDictionary dictionaryWithObjectsAndKeys:[NSNumber numberWithBool:
            YES], NSMigratePersistentStoresAutomaticallyOption, [NSNumber
            numberWithBool:YES], NSInferMappingModelAutomaticallyOption, nil];

        Lightweight migration will only work for a limited set of schema
            changes; consult "Core Data Model Versioning and Data Migration
            Programming Guide" for details.

        */
       NSLog(@"Unresolved error %@, %@", error, [error userInfo]);
       abort();
    }

    return __persistentStoreCoordinator;
```

```
store}, {
    metadata =     {
        NSPersistenceFrameworkVersion = 386;
        NSStoreModelVersionHashes =         {
            Category = <c83227cf 2203fc66 d9b7f2cc 94fcc964 d5f17fc3 f97155e1 4657c319
ff04b501>;
            Stuff = <0ffd8ca7 df656594 ac136f33 07521ade 843be93d 2215799d 643e116f
d2a7bb20>;
        };
        NSStoreModelVersionHashesVersion = 3;
        NSStoreModelVersionIdentifiers =         (
            ""
        );
        NSStoreType = SQLite;
        NSStoreUUID = "D9BA7766-8DDA-4032-BEEB-957C5051E658";
        " NSAutoVacuumLevel" = 2;
    };
    reason = "The model used to open the store is incompatible with the one used to create
the store";
}
(lldb)
```

Xử lý lỗi này

```
[[NSFileManager defaultManager]
removeItemAtURL:storeURL error:nil];
```

Design Model



BasicCoreData › Bas… › Mo… › Bas… › Bas… › Person › fullName

ENTITIES
Person

FETCH REQUESTS

CONFIGURATIONS
Default

Attributes

Attribute	Type
dateOfBirth	Date
firstName	String
fullName	String
lastName	String

Relationships

Relationship	Destination	Inverse

Fetched Properties

Fetched Property	Predicate

Attribute

Name: fullName
Properties: [✓ Transient] [✓ Optional] Indexed
Attribute Type: String
Validation: No Value, Min Length; No Value, Max Length
Default Value: Default Value
Reg. Ex.: Regular Expression
Advanced: Index in Spotlight; Store in External Record File

User Info

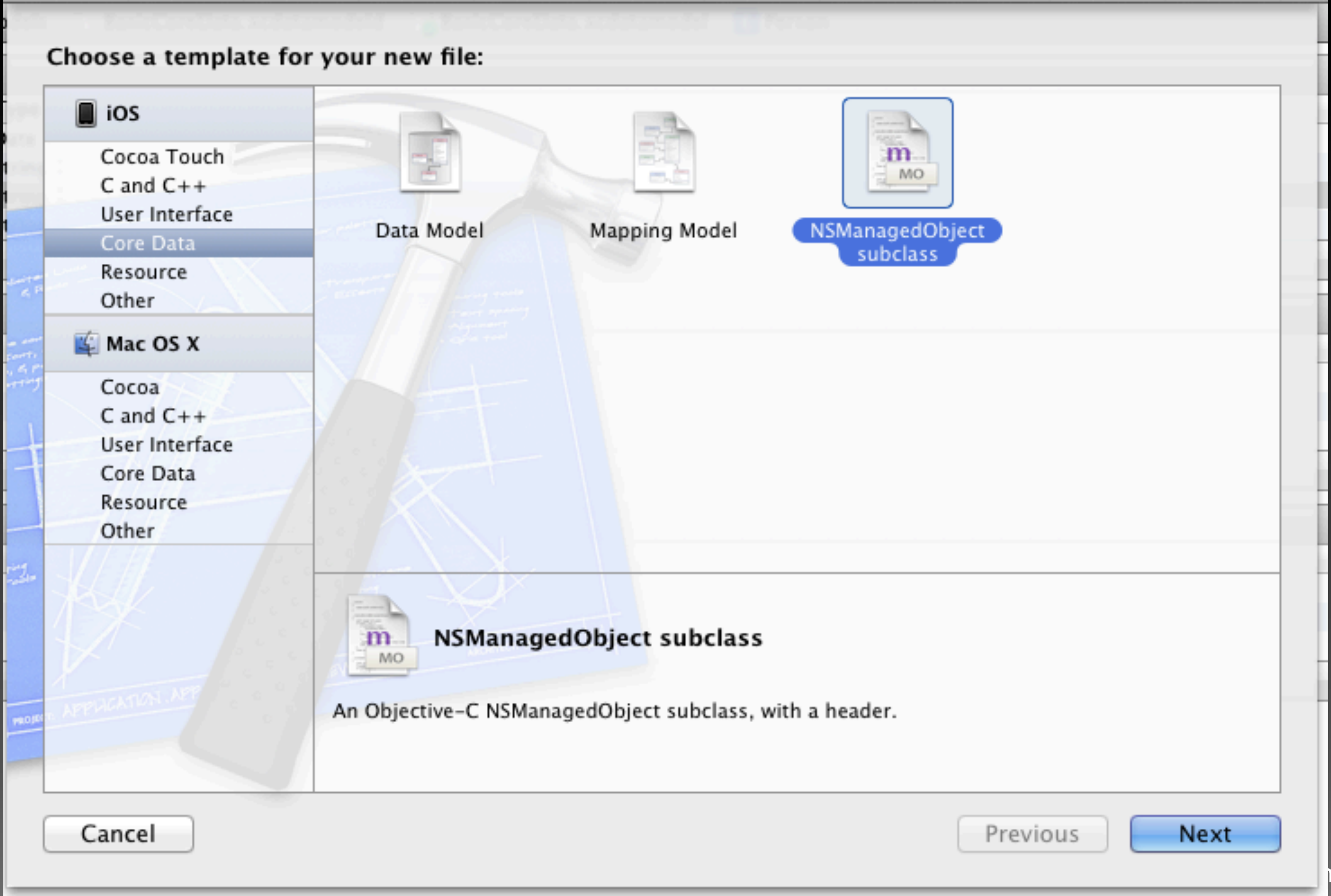
Key	Value

Versioning

Hash Modifier: Version Hash Modifier
Renaming ID: Renaming Identifier

Attribute Sync

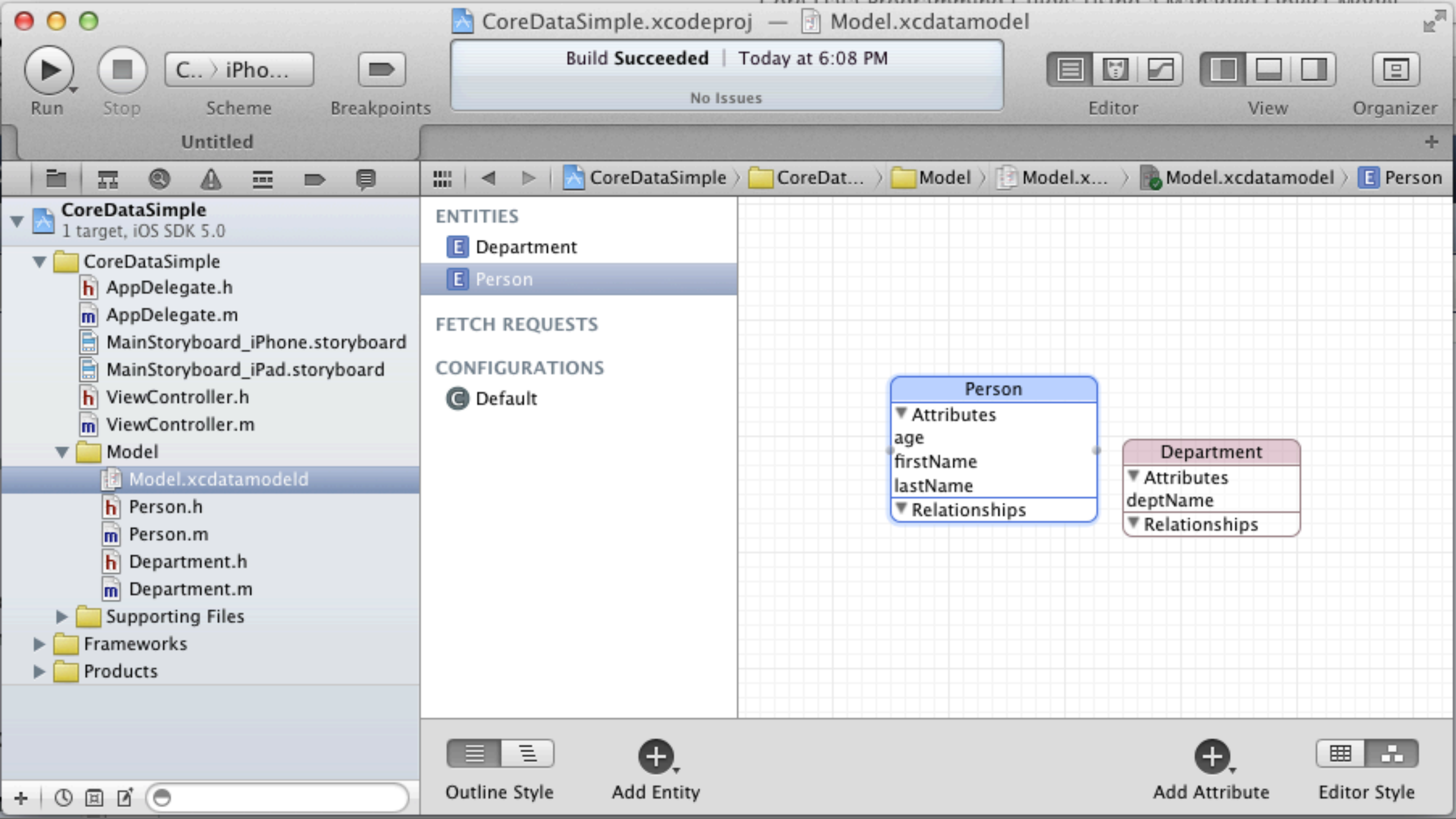
Synchronization: Enabled
Identity Property
Exclude From Change Alert
Client Type: Prefer App
Record: Prefer Truth



Choose a template for your new file:

NSManagedObject subclass

An Objective-C NSManagedObject subclass, with a header.



CoreDataSimple
1 target, iOS SDK 5.0

- CoreDataSimple
  - AppDelegate.h
  - AppDelegate.m
  - MainStoryboard_iPhone.storyboard
  - MainStoryboard_iPad.storyboard
  - ViewController.h
  - ViewController.m
  - Model
    - Model.xcdatamodeld
    - Person.h
    - Person.m
    - Department.h
    - Department.m
  - Supporting Files
- Frameworks
- Products

ENTITIES
- Department
- Person

FETCH REQUESTS

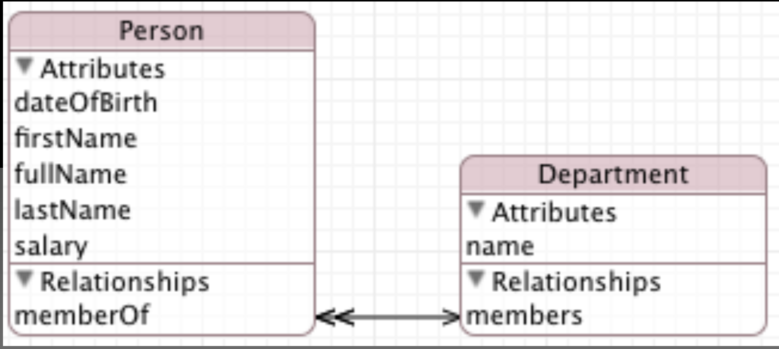
CONFIGURATIONS
- Default

Person
- Attributes
  - age
  - firstName
  - lastName
- Relationships

Department
- Attributes
  - deptName
- Relationships

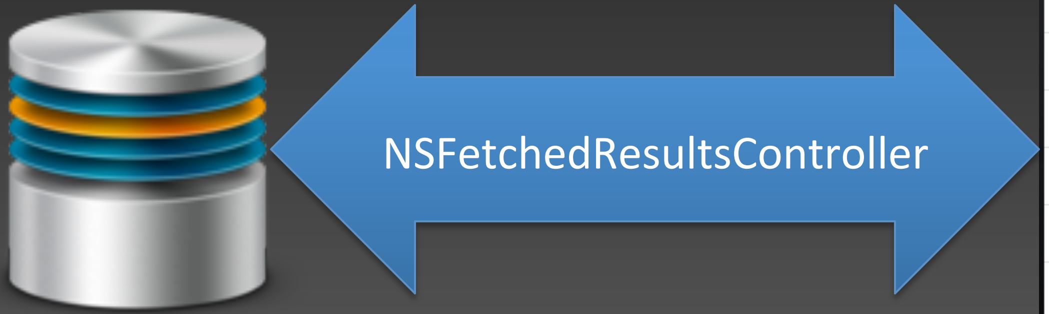
Join query

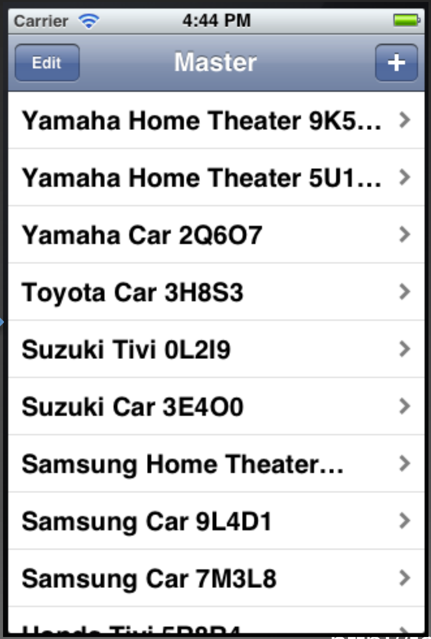
```
NSFetchRequest *request = [[NSFetchRequest alloc]
 initWithEntityName:@"Department"];

NSPredicate *predicate = [NSPredicate predicateWithFormat:
@"ANY members.firstName LIKE 'Duong'"];

 [request setPredicate:predicate];

NSError *error = nil;
NSArray *array = [managedObjectContext
executeFetchRequest:request
error:&error];
```



NSFetchedResultsController

Fetched results controller to efficiently manage the results returned from a Core Data fetch request to provide data for a UITableView object.





Yamaha Home Theater 9K5...
Yamaha Home Theater 5U1...
Yamaha Car 2Q6O7
Toyota Car 3H8S3
Suzuki Tivi 0L2I9
Suzuki Car 3E4O0
Samsung Home Theater...
Samsung Car 9L4D1
Samsung Car 7M3L8

Cool features of NSFetchedResultsController

- Monitors changes to objects in its associated managed object context, and reports changes in the results set to its delegate
- Caches the results of its computation so that if the same data is subsequently re-displayed, the work does not have to be repeated