Core Data

Запазени данни с ORM

/data persistence/

лекция, представена от Г.Пенков в рамките на курса ios2011 към ФМИ на СУ

Core Data ORM

- ENTER : Core Data
- По-лесно за управление на паметта
- Де факто стандартния начин за работа със запазени данни (persistent data)
- По-удобно спрямо NSCoding, когато става дума за комплексни данни (NSCoding е протоколът за сериализиране & архивиране на данни).
- Предоставя механизъм за Object Relational Mapping - спестява всичкото писане на SQL и дава като резултат по-сигурен и стабилен код.

Класически подход при работа с DB

(...който ние ще избягваме)

1. Установяване на връзка, получаване на обект-идентификатор на връзката (connect / connection handle(r))

2. Подготвяне на SQL заявка като текст (prepare)
   - ...което води до обработка (parse) от страна на системата за връзка с базата (db driver)

3. Изпълняване на заявката (execute)
   - ... от страна на системата за бази данни
   -
4. Получаване на обект за извличане на резултата (result set)

5. Обхождане на резултата (fetch)

образно казано : досадно, бавно, несигурно, етц.

ORM ?

- Механизъм за автоматизиране достъпа до базата през междинен "виртуален" слой
- Обикновено включва и (най-често автоматизирано) създаване на помощни класове
- Помощните класове "управляват" достъпа и работата с запазени в базата данни.
- Капсулация на достъпа и операциите
- Кеширане на данните
- Прозрачен ООП подход при достъп до полетата на записите, с които искаме да работим.
- XCode предоставя вградена подсистема за създаване на object relational mapping - т.нар. entity редактор

Core Data основни функции

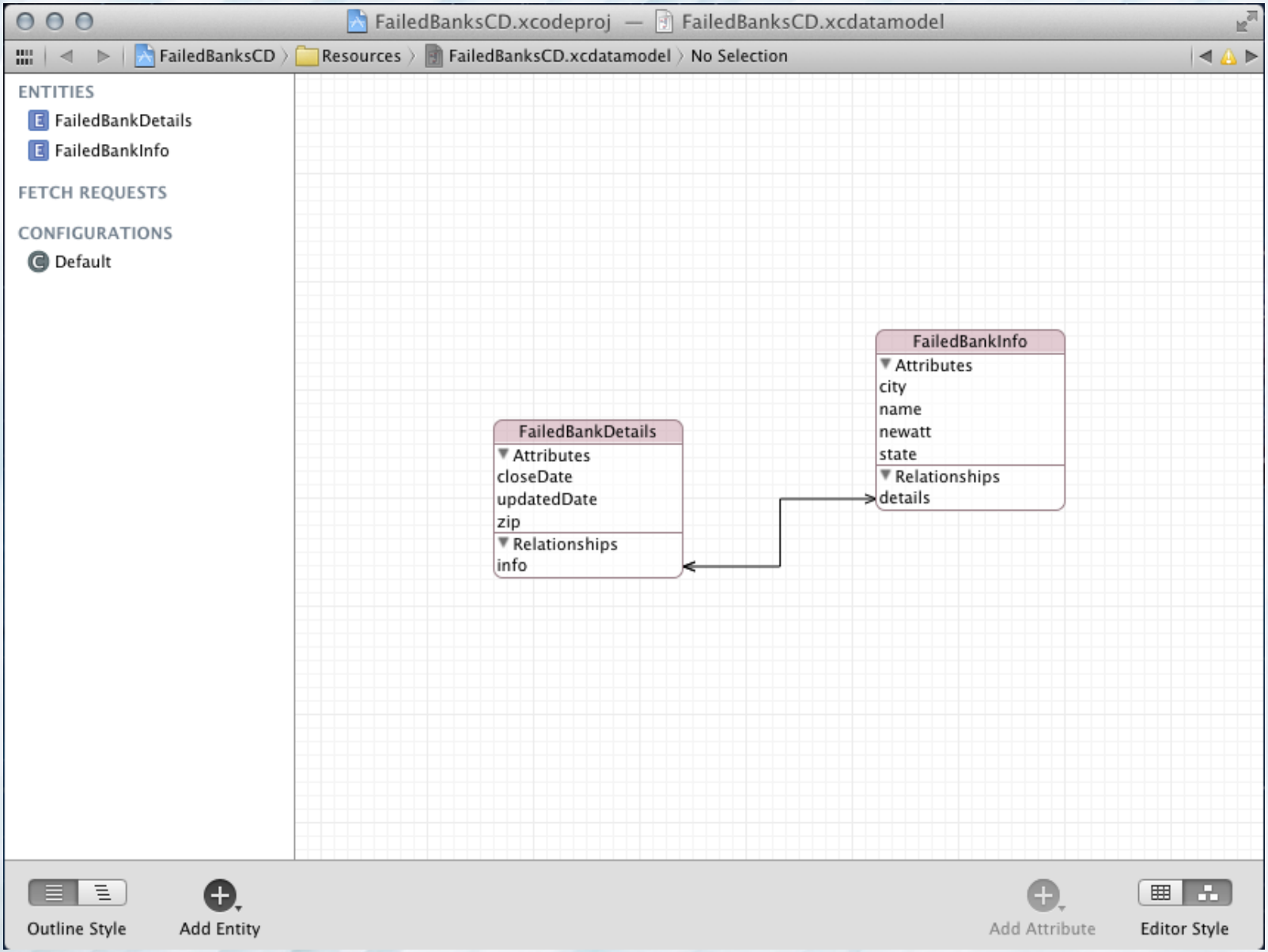
- **Проследяване на промени и възможност за връщане (undo)**
  - Всеки буфер от обекти пази промените и може да ги връща
- Поддържане на връзки между същности
  - Core Data управлява проследяването на промени по записите, върху които оперира и по-специално поддържане състоянието на отношенията сред обектите
- **Лениво зареждане на обекти (faulting)**
  - Намалява използваната памет - данните се зареждат само при нужда (faulting).
  - Поддържа се и копиране при запис (copy-on-write) за данни с общ произход
- Автоматична валидация на стойностите на характеристиките на обектите (property validation)
  - Управляваните от Core Data’s обекти (managed objects) допълват стандартния подход при работа с ключ-стойност, така че да може да се задават ограничения към една или комбинация от характеристики в дадено подмножество.

Core Data основни функции

- **Промени по схемата в модела на данните** (schema migration)
  - Възможно е проследяването на промени и автоматизирана миграция към обновена версия на схемата
- **Автоматизирана доставка на данни към UI**
  - Резултати, получени чрез NSFetchedResultsController, може да бъдат синхронизирани автоматично в съответните полета от потребителския интерфейс
- **Автоматично изграждане на съответния SQL код**
  - Дори и сложни заявки са възможни без писане на SQL, но с използване на NSPredicate инстанции за уточняване изискванията към заявените обекти
- **Групиране, филтриране и поддръжка на различни системи за бази данни** (SQLite по подразбиране)

Редактор на същности

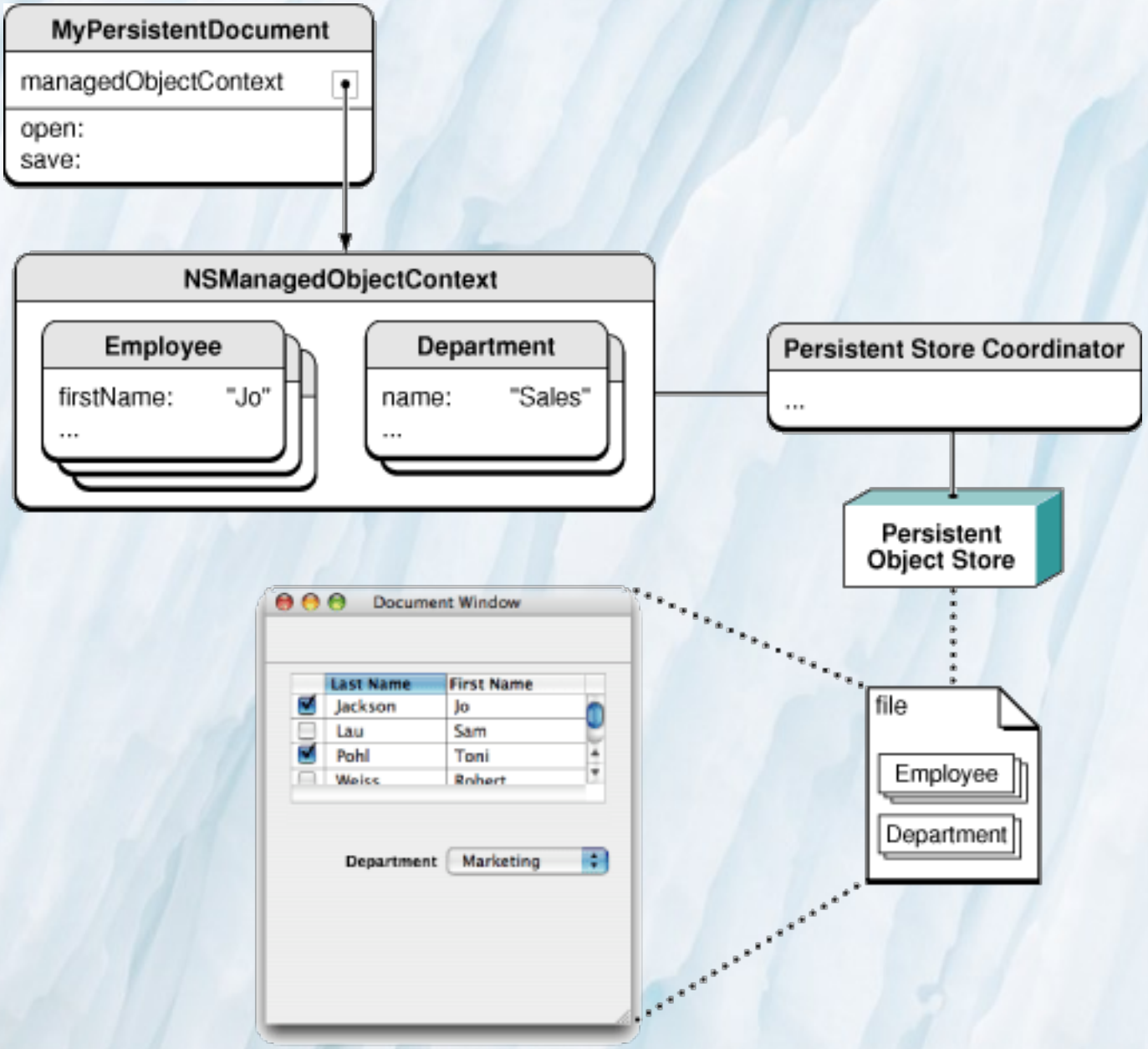
(entity editor - XCode 4.x)



XCode и ORM с Objective C

- При iPhone/iPad хранилището, което управлява запазването на данните по подразбиране се управлява от SQLite
- При създаване на проекта указваме "use core data for storage"
- Получаваме автоматична имплементация на managed object context, managed object model, persistent store coordinator. След малко за тях.
- Описанието на entity-relationship модела става :
  - директно.... с диаграма
  - или в табличен вид
  - се запазва в .xcdatamodel ресурс
  - води до автоматичното генериране на класовете, които управляват (manage) работата със същностите (entities) описани по този начин

Принцип на работа с Core Data

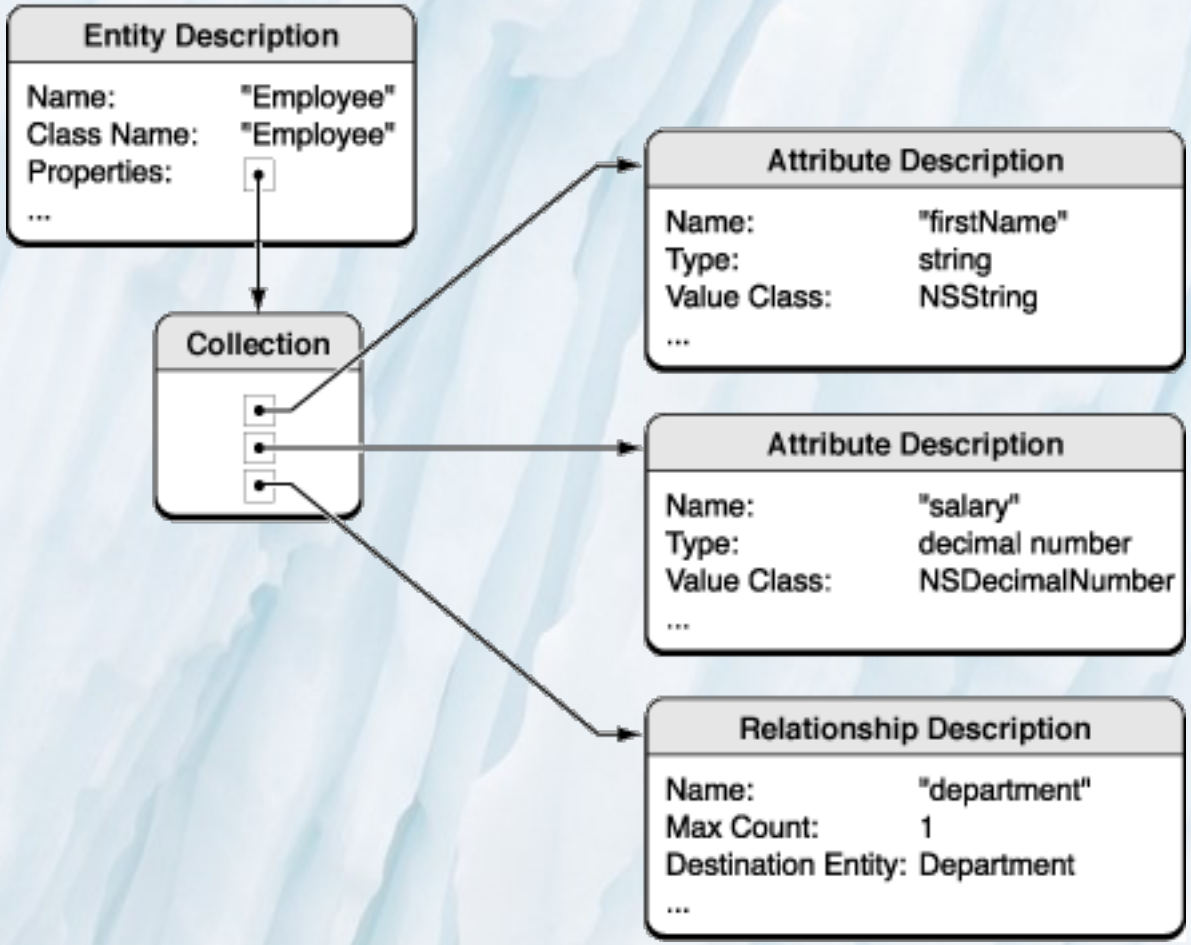


Основни понятия

- ***Managed Object Model*** - модел на същностите.
  - *може да си мислим за него като за схемата на базата*
  - клас, който съдържа дефинициите на всички същности (entities) и техните характеристики (properties)
  - обикновено е резултат от операциите по описанието на модела във визуалния редактор
  - ...но може да се допълва и ръчно (т.е. с писане на код по наше усмотрение)
  - съдържа множество инстанции на описания на същности (NSEntityDescription)

Основни понятия: Модел на същностите

/примерно описание на същност/

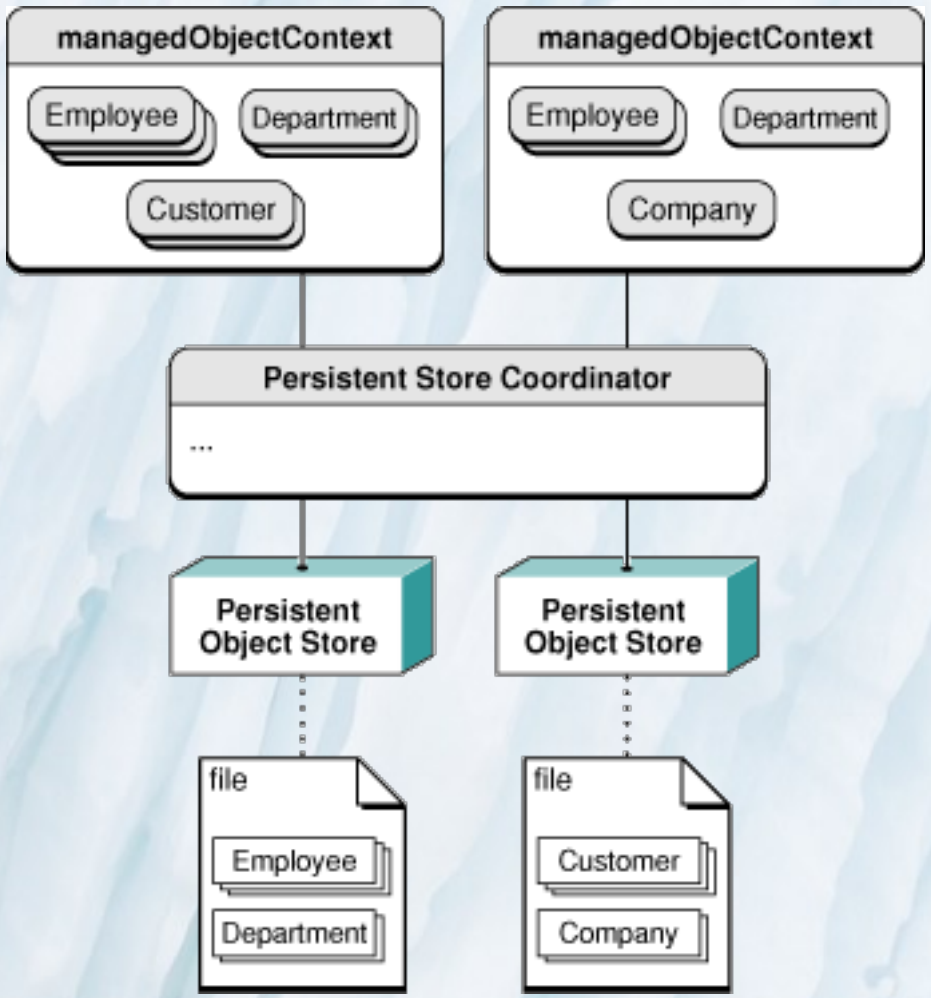


Основни понятия

- ***Persistent Store Coordinator*** - идентификатор на връзката към базата
  - Можем да си мислим за него като за "database handle"
  - Тук се дефинира къде реално се запазват данните
  - ...и как се казва базата
  - Обръщенията към базата "минават" през този координатор обект
  - Даден Persistent Store (външно хранилище за данни) може да се реализира от файл, SQLite подсистема или друг вид външна подсистема

Основни понятия: Координаторен стек

/ Advanced persistence stack/

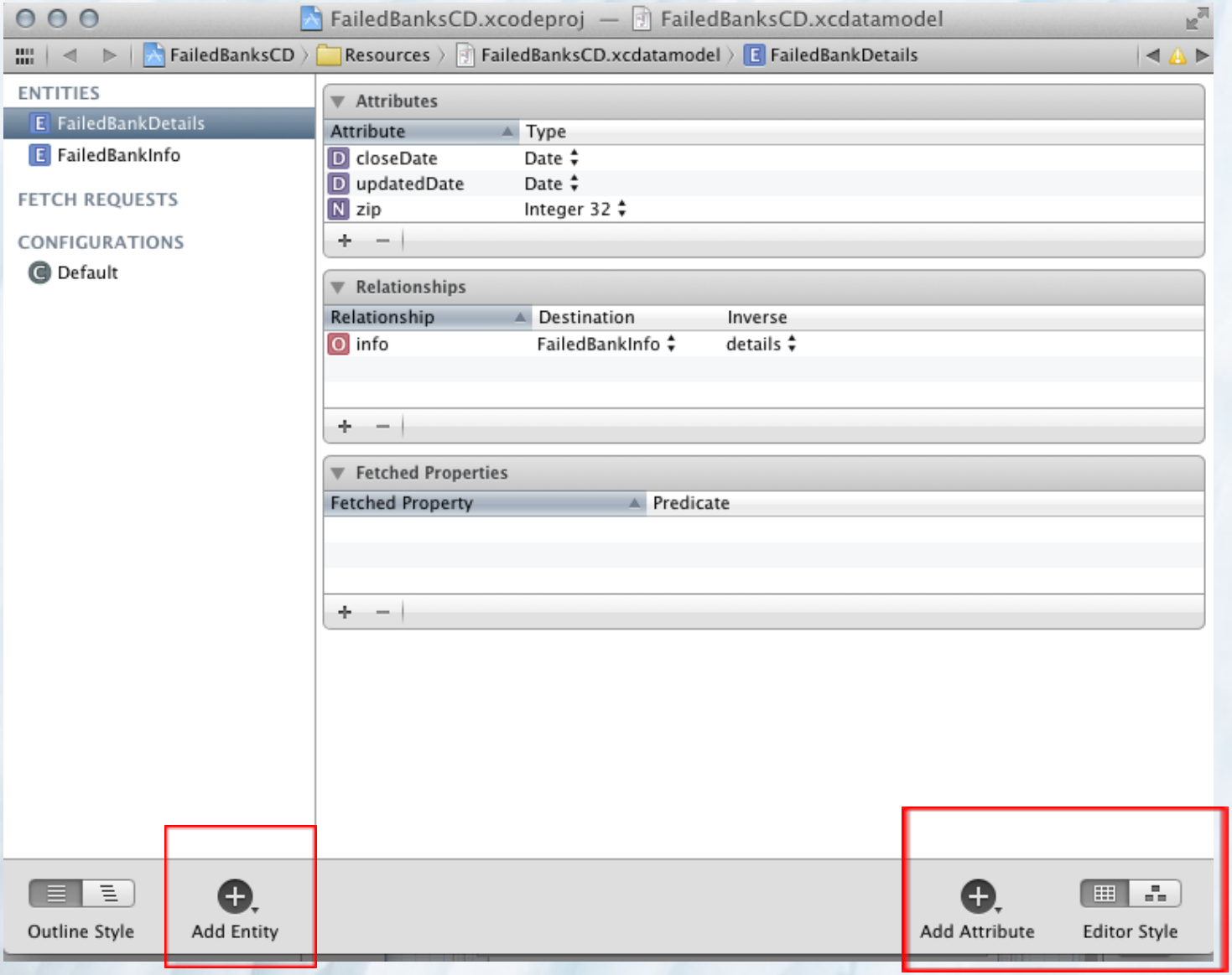


Основни понятия

- ***Managed Object Context*** - нещо като буфер или чернова, в която се пазят данните за managed обектите
  - междинното звено, което съдържа данните преди да достигнат до базата
  - спестява операции по четене ако данните не са се променили
  - всички insert / delete / select операции се извършват през такава инстанция
  - позволява връщане на стойности (undo)

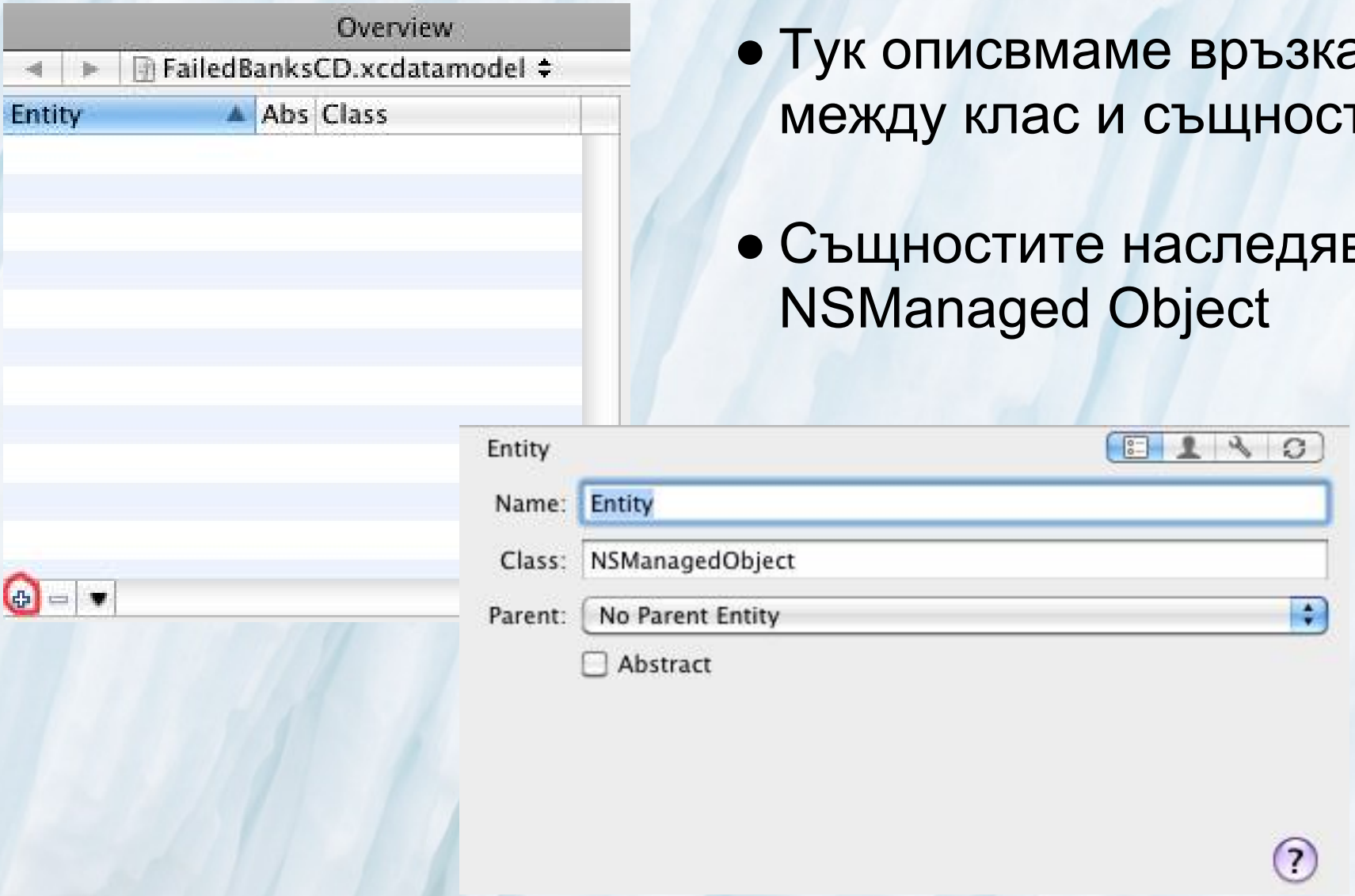
Визуален редактор :

(Снимка от XCode 4.X)



Добавяне на същности

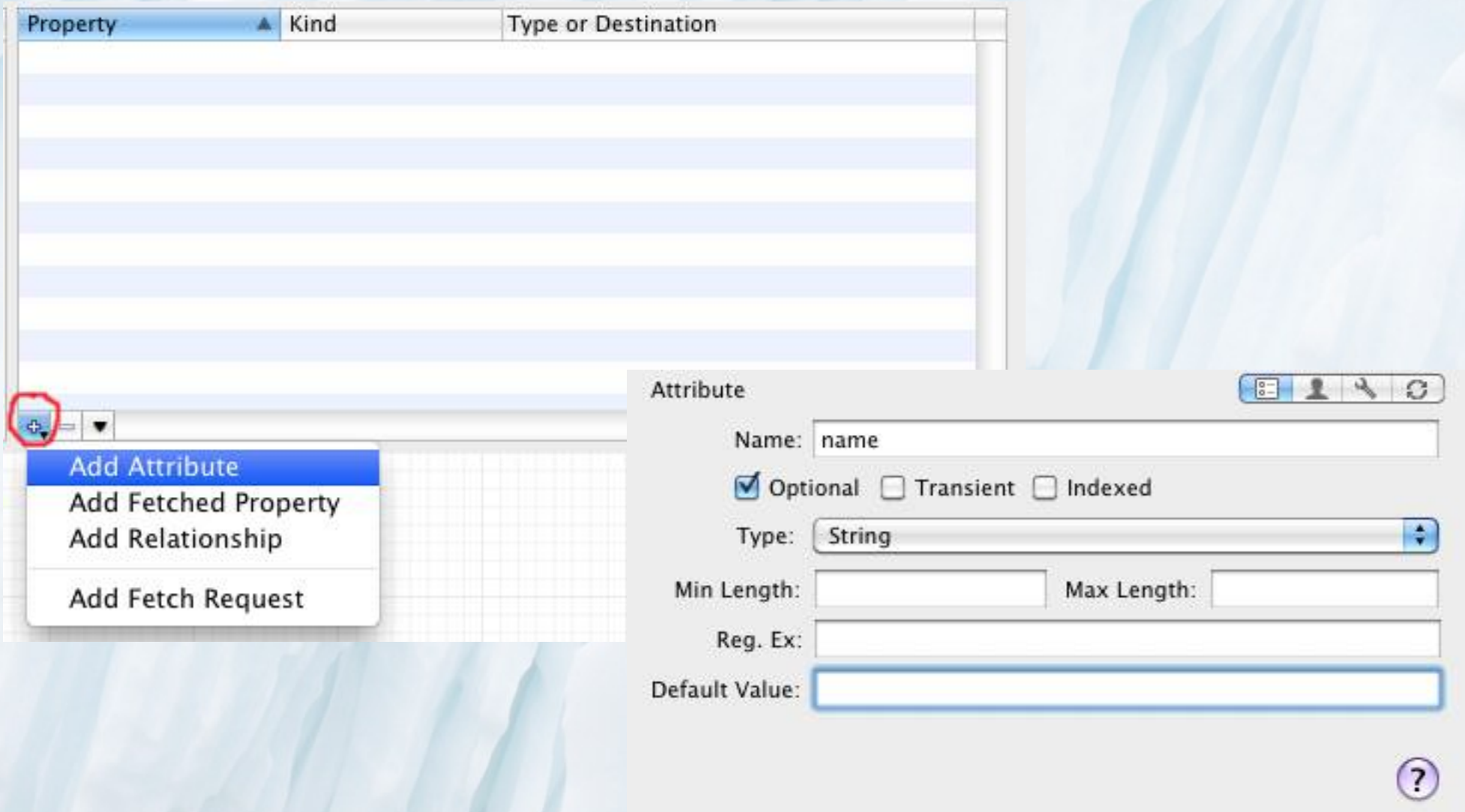
(Снимка от XCode 3.X)



- Тук описвмаме връзката между клас и същност
- Същностите наследяват NSManaged Object

Добавяне на атрибути

(Снимка от XCode 3.X)



Добавяне на връзки

(Снимка от XCode 4.X)

- може да укажем кой е ключа
- може да се укаже и обратната връзка
- автоматично се генерират съответните setter/getter

Основни обекти

- NSManagedObject - Всички управлявани от Core Data ORM обекти (managed objects) следва да са инстанция на този или наследяващ го клас. Той предоставя всички нужни методи и реализира вътрешните структури нужни за постигане на абстракцията.(вкл. описанията на същностите са от този тип).
- NSManagedObjectContext - Представя контекстът (виртуалния слой), в който попадат данните преди запис и след четене. Всички заявки и записи на данни стават през него.
- NSEntityDescription - Обект описващ характеристиките на дадена същност.
- NSFetchRequest - Обект, който обозначава заявка към базата. Може да приеме предикати (NSPredicate), които да конкретизират изискванията към заявените обекти.

Пример

/да създадем малко данни/

```
// (в applicationDidFinishLoading...)

// получаване на управляван обект
NSManagedObjectContext *context = [self managedObjectContext];

// нова инстанция от същността FailedBankInfo
NSManagedObject *failedBankInfo = [NSEntityDescription insertNewObjectForEntityForName:@"FailedBankInfo"
inManagedObjectContext:context];

[failedBankInfo setValue:@"Test Bank" forKey:@"name"];
[failedBankInfo setValue:@"Testville" forKey:@"city"];
[failedBankInfo setValue:@"Testland" forKey:@"state"];

// нова инстанция от същността FailedBankDetails
NSManagedObject *failedBankDetails = [NSEntityDescription insertNewObjectForEntityForName:@"FailedBankDetails"
inManagedObjectContext:context];

// задаване на стойности за дадени характеристики на обекта
[failedBankDetails setValue:[NSDate date] forKey:@"closeDate"];
[failedBankDetails setValue:[NSDate date] forKey:@"updatedDate"];
[failedBankDetails setValue:[NSNumber numberWithInt:12345] forKey:@"zip"];
[failedBankDetails setValue:failedBankInfo forKey:@"info"];
[failedBankInfo setValue:failedBankDetails forKey:@"details"];

// явен запис на натрупаните в контекста промени
NSError *error;
if (![context save:&error]) {
    NSLog(@"Whoops, couldn't save: %@", [error localizedDescription]); }
```

Какво се случва ?

- Първо взимаме указател към контекст на управлявани обекти (managed object context) - методът managedObjectContext
- Създаваме нова инстанция на NSManagedObject за същността FailedBankInfo като извикваме insertNewObjectForEntityForName.
- Всеки обект, който се записва от Core Data наследява NSManagedObject.
- Веднъж след като е направена инстанция на този обкет може да се извика setValue за всеки атрибут дефиниран във визуалния редактор
- Създаваме още обекти, но до момента те са променени само в паметта - в managedObjectContext. Т.е. трябва да извикаме "save"
- След save (което е като commit в други езици) данните са запазени

Отношения (relationships)

/достъп до атрибути и един-към-един/

В Objective C 2.0 може директно да се напише :
NSString *firstName = [anEmployee firstName];
Employee *manager = anEmployee.manager;

Но може да се ползва и традиционен KVC (key-value-coding) подход за достъп по ключ-стойност :

```
newEmployee.firstName = @"Stig";
[newEmployee setManager:manager];
```

Отношения (relationships)

/един към много/

В Objective C 2.0 може директно да се напише за достъп до множество от обекти :

```
NSSet *managersPeers = [managersManager directReports];
NSSet *departmentsEmployees = aDepartment.employees;
```

Но може да се ползва и традиционен KVC (key-value-coding) подход за достъп по ключ-стойност :

```
NSSet *newEmployees = [NSSet setWithObjects:employee1, employee2, nil];
[aDepartment setEmployees:newEmployees]; NSSet *newDirectReports =
[NSSet setWithObjects:employee3, nil]; manager.directReports =
newDirectReports;
```

Зареждане на обекти

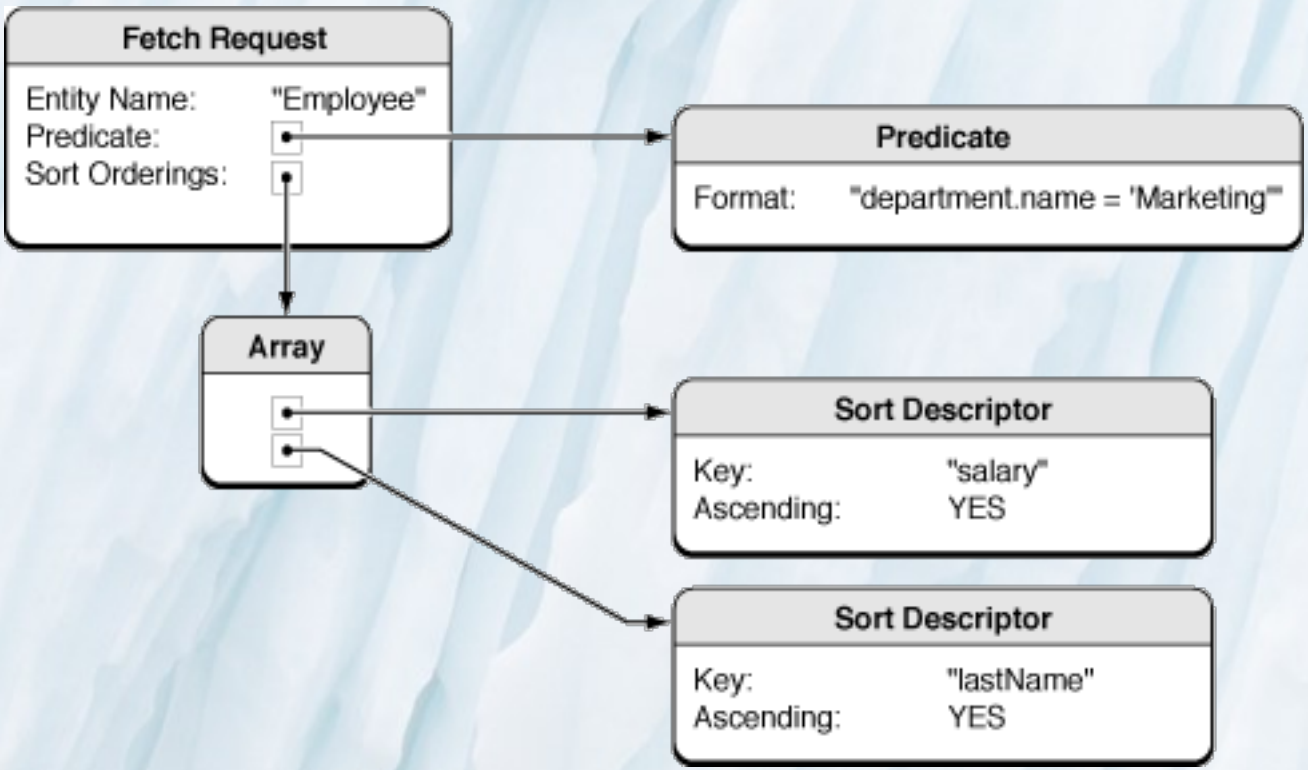
(заб: всичко цветно е анотирано с връзки към съответните помощни страници)

```
// създаване на обект "заявка"
NSFetchRequest *fetchRequest = [[NSFetchRequest alloc] init];

// определяме какъв обект търсим
NSEntityDescription *entity = [NSEntityDescription entityForName:@"FailedBankInfo"
inManagedObjectContext:context];
// изпълняваме заявката
[fetchRequest setEntity:entity];
NSArray *fetchedObjects = [context executeFetchRequest:fetchRequest error:&error];
// обхождаме резултатите
for (NSManagedObject *info in fetchedObjects) {
    NSLog(@"Name: %@", [info valueForKey:@"name"]);
    NSManagedObject *details = [info valueForKey:@"details"];
    NSLog(@"Zip: %@", [details valueForKey:@"zip"]); }[fetchRequest release];
}
```

Заявка на данни

/принципна схема/



Какво се случва в примера ?

- Създаваме нов обект, който е ***fetch request***. Може да си мислим за него като за SELECT клауза (...но на този етап с неопределени параметри)
- Създаваме нов обект от тип "описание на същност, с който указваме, че ще заявяваме "FailedBankInfo" същности :
  - ...и после setEntity за да свържем заявката с желаните същности
  - Тук може да създадем и укажем и NSPredicate, за рафиниране на заявката
- Изпълняваме executeFetchRequest (метод на обекта-контекст), за да заредим желаните обектите в контекста
- После обхождаме получените NSManagedObject в резултата и използваме valueForKey за да четем отделните му характеристики

А какво се случва на практика...?

Може да проследим с включване на

-com.apple.CoreData.SQLDebug 1

Което генерира следната помощна информация за генерираните SQL заявки

SELECT Z_VERSION, Z_UUID, Z_PLIST FROM Z_METADATA SELECT Z_MAX FROM Z_PRIMARYKEY WHERE Z_ENT = ?

UPDATE Z_PRIMARYKEY SET Z_MAX = ? WHERE Z_ENT = ? AND Z_MAX = ?

INSERT INTO ZFAILEDBANKDETAILS(Z_PK, Z_ENT, Z_OPT, ZINFO, ZUPDATEDDATE, ZZIP, ZCLOSEDATE) VALUES(?, ?, ?, ?, ?, ?, ?)

INSERT INTO ZFAILEDBANKINFO(Z_PK, Z_ENT, Z_OPT, ZDETAILS, ZNAME, ZSTATE, ZCITY) VALUES(?, ?, ?, ?, ?, ?, ?)

SELECT 0, t0.Z_PK, t0.Z_OPT, t0.ZNAME, t0.ZSTATE, t0.ZCITY, t0.ZDETAILS FROM ZFAILEDBANKINFO t0

SELECT 0, t0.Z_PK, t0.Z_OPT, t0.ZUPDATEDDATE, t0.ZZIP, t0.ZCLOSEDATE, t0.ZINFO FROM ZFAILEDBANKDETAILS t0 WHERE t0.Z_PK = ?

Какво се случва ? Faulting

- Когато се заяви Core Data поле, което не е заредено вече в контекста...
- се случва "fault", който предизвиква зареждане на данните от хранилището
- Ние не се грижим за "fault" операциите при зареждане на данни. това се случва по автоматизиран начин
- Трябва явно да укажем "save" за да синхронизираме (запазим) новосъздадените в контекста данни в хранилището

Речник

- entity - същност
- database system - хранилище за данни, (кратко : база)
- database schema - схема в хранилището
- (object) property - характеристика (на обект)
- (object) instance - инстанция, екземпляр от даден клас
- execute/call method - изпълняване на метод на даден обект
- managed object - управляван обект

Полезни връзки

- ORM... накратко
  - http://en.wikipedia.org/wiki/Object-relational_mapping

- Introduction to Core Data Programming Guide
  - http://developer.apple.com/library/mac/#documentation/cocoa/conceptual/coredata/cdprogrammingguide.html

- Ръководство за миграция на схеми - http://www.timisted.net/blog/archive/core-data-migration/

- Core Data Tutorial - източник на някои от примерите в изложението
http://www.raywenderlich.com/934/core-data-tutorial-getting-started

- Работа със свързани (referenced) обекти - http://developer.apple.com/library/mac/#documentation/cocoa/conceptual/coredata/Articles/cdRelationships.html