CoreData

Part II

План лекции

- Predicate
- FetchResultController
- N-Context
- Как лучше проектировать хранилище в приложении?

NSPredicate

- Объект для создание запроса наподобие Query аналогов
- Может сортировать и фетчить данные из коллекций
- Имеет свой синтаксис

- https://nshipster.com/nspredicate/

NSFetchResultController

- Controller слоя M - Model в MVC
- Является одним из Model Controller’ов в iOS SDK (UIDocument - его собрат)
- Умеет кэшировать запросы для оптимизации работы с данными
- Является идеальным инструментом для работы с коллекциями UITable/CollectionView

NSFetchResultController как работает

- delegate = nil и cache = nil - данные каждый раз берутся после performFetch() из базы
- delegate != nil и cache = nil - позволяет мониторить изменение данных, работы с кэшом нет
- delegate != nil и cache != nil - режим мониторинга с кэшированием объектов

NSFetchResultControllerDelegate

- Оповещает об изменениях
- Нужен для обновления UI, если изменилась Model
- sectionIndexTitleForSectionName - для изменений названий секций
- controllerWillChangeContent - начало изменений данных
- didChangeSection - изменение в конкретных секциях
- didChangeObject - изменение конкретного объекта
- controllerDidChangeContent - конец изменений данных

Работа с N-Context’ами

- Parent -> Child - контексты зависят друг от друга на уровне API NSManagedObjectContext, один из них всегда будет главным. (Есть вероятность словить deadlock, если вдруг из child будет обращение в parent)
- Independent with merge on notification - оба контекста независимы, но ссылаются на один и тот же persistentStoreCoordinator, синхронизация данных осуществляется по нотификации

Как лучше проектирование хранилище?

- Изолировать, но тогда не будет FetchResultController
- Работать через import CoreData 🤮

- Пример в проекте