Разработка iOS приложений

Лекция 9.

CoreData

Core Data – Работа с данными

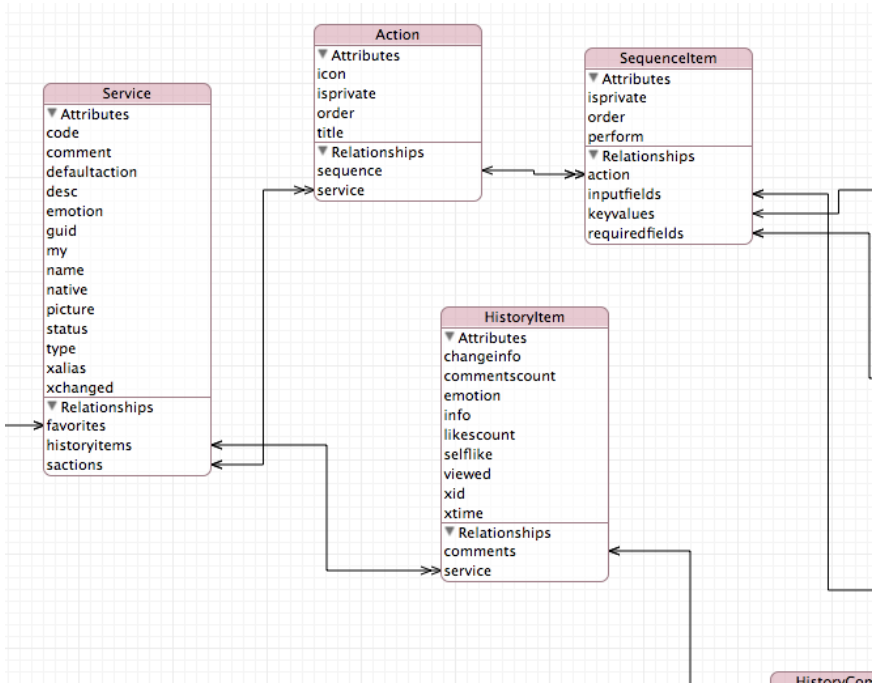
- Модель
- Хранилище
- Контекст

Guides:

- CoreData Programming Guide

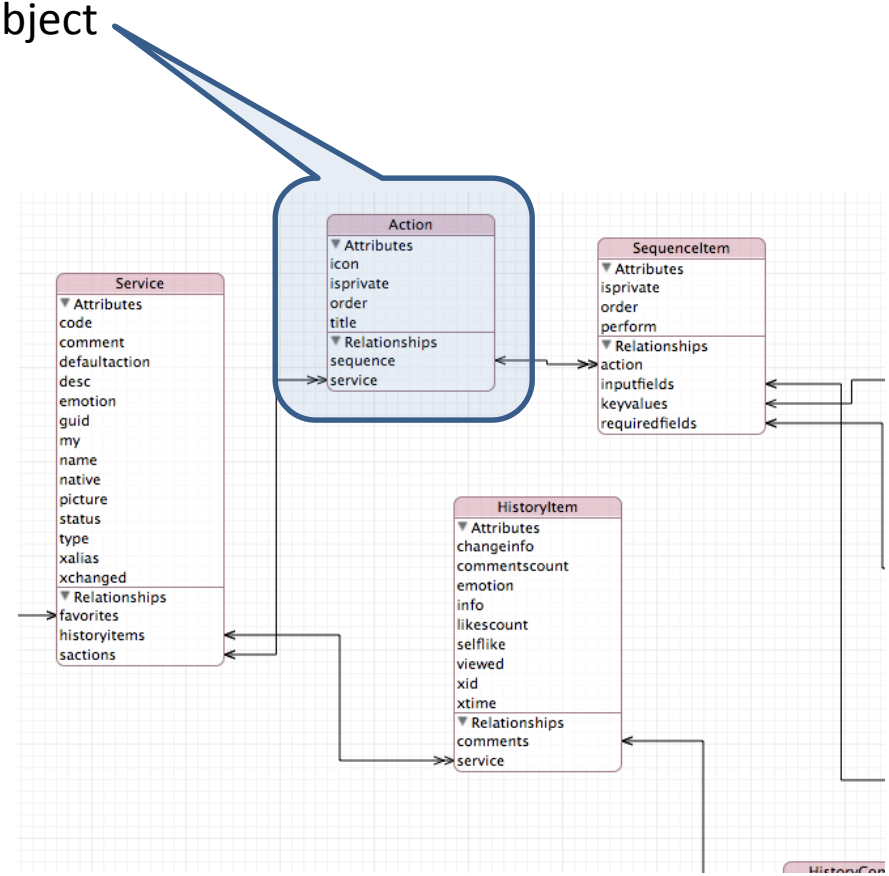
Core Data – Модель

- NSManagedObjectModel



Core Data – Модель

- NSManagedObject



Core Data – Модель

- NSManagedObject

```
@interface InputField : NSManagedObject

@property ( nonatomic, retain ) NSString* content;
@property ( nonatomic, retain ) NSString* namekey;
@property ( nonatomic, retain ) SequenceItem* sequenceitem;

@end
```

Core Data – Модель

- NSManagedObject

- ВАЖНО!
- NSManagedObject’ы – генерируемые классы. Поэтому ваши дополнительные поля и методы потеряются при регенерации.
- Дополнительные поля и методы целесообразно реализовывать через категории

Core Data – Хранилище

- NSPersistentStore
- NSPersistentStoreCoordinator

- SQLite – NSSQLiteStoreType
- XML – NSXmlStoreType
- Бинарный файл – NSBinaryStoreType
- Оперативная память - NSInMemoryStoreType

Core Data – Контекст

- NSManagedObjectContext

Core Data – Контекст, потокобезопасность

- **Контекст потоконебезопасен!**
- **На один поток – один контекст!**
- **Обновление контекстов – через нотификации!**

Core Data – Запрос данных

- NSFetchRequest

- [request setPredicate: …]
- [request setEntity:]
- [request setSortDescriptors:]

- [request executeFetchRequest: …]

Core Data – UNDO/REDO

- NSManagedObjectContext

- [context undo]
- [context redo]

- [context save]
- [context rollback]

Практика

- Перевести классы RSS, RSSChannel и RSSItem на CoreData (нарисовать модель и сделать их NSManagedObject), дополнительные поля и методы реализовать через категории
- При загрузке данных из сети сохранять их в хранилище CoreData, затем уведомлять GUI, которое через другой (!) контекст загрузить новые данные
- ВАЖНО! GUI работает только с данным в хранилище CoreData!

Итак…

- Это последнее занятие из основного цикла…
- Дальше я буду рассказывать некоторые темы, но практики по ним не будет