

Хранение данных



- Кэширование
- Оффлайн работа
- Быстродействие

UserDefaults

- примитивы
- потокобезопасен

Разделяется по доменам для кросс-доменной работы (их можно делить между extensions и вообще между приложениями)

User Defaults



Хранит Bool, Dictionary, Int, String, Data, Array

```
UserDefaults.standard.set("Peter", forKey: "name")  
let name = UserDefaults.standard.string(forKey: "name") ?? ""
```

```
UserDefaults.standard.set(true, forKey: "loggedIn")  
let loggedIn = UserDefaults.standard.bool(forKey: "loggedIn") ?? false
```

Файловая система

NSFileManager

- список файлов
- удаление
- перемещение
- копирование

FileManager



// сохранение

```
let fileManager = FileManager.default
let path =
(NSSearchPathForDirectoriesInDomains(.documentDirectory, .userDomainMask, true)
[0] as NSString).appendingPathComponent("apple.jpg")
```

```
let image = UIImage(named: "apple.jpg")
let imageData = image!.jpegData(compressionQuality: 0.5)
```

```
fileManager.createFile(atPath: path as String, contents: imageData, attributes:
nil)
```

// загрузка

```
if fileManager.fileExists(atPath: path) {
    self.imageView.image = UIImage(contentsOfFile: path)
}
```

Хранение деревьев объектов

- стор
- не является реляционной базой
- несколько бекендов
 - Sqlite
 - Xml
 - память

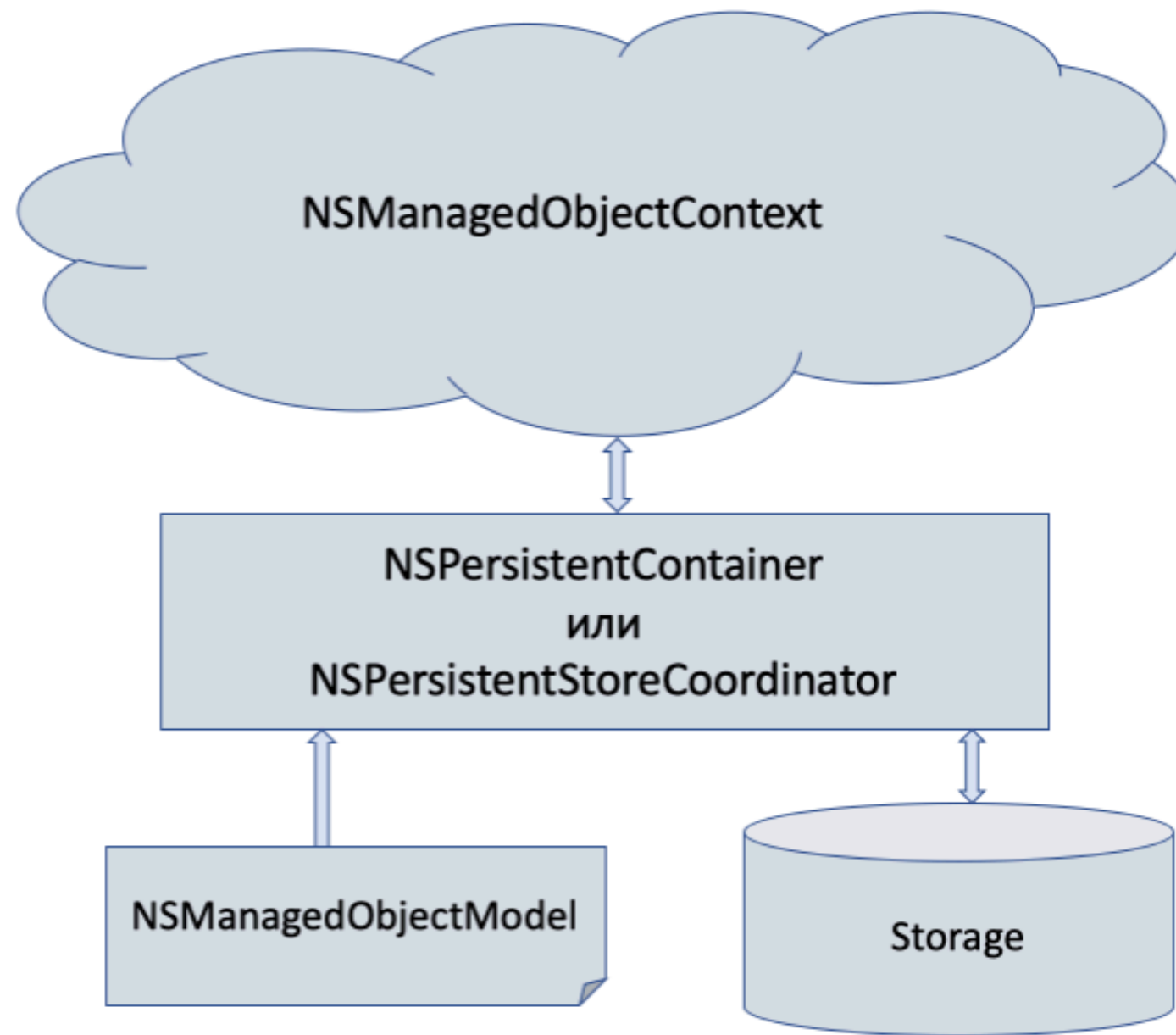
- NSManagedEntity — хранимые объекты
- NSManagedObject — базовый класс для объектов
- NSFetchRequest — запрос на выборку объектов
- Все операции делаются через контекст
- Объекты нельзя передавать между потоками напрямую
 - Можно только через ид объекта

CoreData: основные понятия



- NSManagedObjectModel — описание структуры данных (загружается в память при старте)
- NSPersistentStore — конкретная разновидность хранилища (SQLite, Binary, In-Memory + свои)
- NSPersistentStoreCoordinator
 - отвечает за создание/извлечение существующих из persistent store;
 - передаёт их запросившему контексту.
- NSManagedObjectContext
 - при извлечении объектов из persistent store создаётся объектный граф;
 - все изменения выполняются в контексте;
 - если контекст сохраняют, то изменения сохраняются в persistent store.

CoreData



NSFetchedResultsController

- Выбирает данные для представления в виде UITableView
- Поддерживает секции
- Реагирует на изменения NSFetchRequest и вызывает делегат с описанием этих изменений

Класс, предназначенный для задания каких-то условий фильтрации

Его можно создать

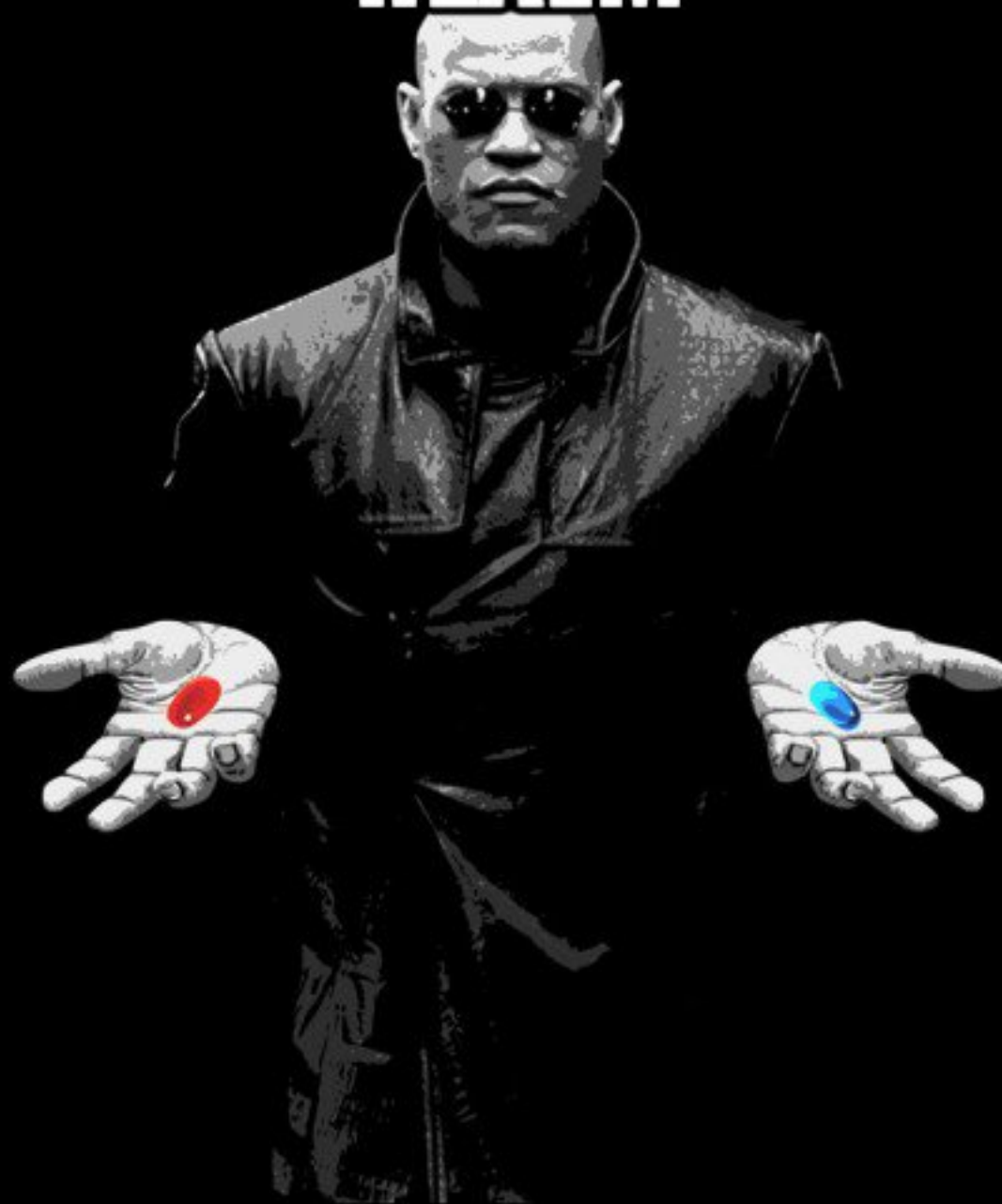
- из строки с условием
- блоком кода, который отдаёт Bool в зависимости от того, подходит объект или нет
- как логическую комбинацию других предикатов
- Подробнее см. ссылки

- Тоже объектное хранилище
- Использует свой собственный высокопроизводительный бекенд хранения
- Есть для ObjectiveC, Swift, Java, JS (если у вас swift + objc, то надо использовать objc версию)

- Object — базовый класс для моделей данных
- Связи моделируют отдельным классом List
- Условия в запросах делаются тоже через NSPredicate
- Объекты realm не должны передаваться между потоками
- Аналог NSFetchedResultsController это нотификации



REALM



COREDATA

risovach.ru

- низкий уровень
- C-API
- Реляционная база

Для облегчения работы есть несколько библиотек

FMDB - создание базы



```
let filemgr = FileManager.default
let dirPaths = filemgr.urls(for: .documentDirectory,
                             in: .userDomainMask)

databasePath = dirPaths[0].appendingPathComponent("contacts.db").path

if !filemgr.fileExists(atPath: databasePath as String) {

    let contactDB = FMDatabase(path: databasePath as String)

    if contactDB == nil {
        print("Error: \(contactDB?.lastErrorMessage())")
    }

    if (contactDB?.open())! {
        let sql_stmt = "CREATE TABLE IF NOT EXISTS CONTACTS (ID INTEGER PRIMARY
KEY AUTOINCREMENT, NAME TEXT, ADDRESS TEXT, PHONE TEXT)"
        if !(contactDB?.executeStatements(sql_stmt))! {
            print("Error: \(contactDB?.lastErrorMessage())")
        }
        contactDB?.close()
    } else {
        print("Error: \(contactDB?.lastErrorMessage())")
    }
}
```


FMDB - добавление



```
let contactDB = FMDatabase(path: databasePath as String)

if (contactDB?.open())! {

    let insertSQL = "INSERT INTO CONTACTS (name, address, phone) VALUES ('\
(name.text!)', '\(address.text!)', '\(phone.text!)')"

    let result = contactDB?.executeUpdate(insertSQL,
                                         withArgumentsIn: nil)

    if !result! {
        status.text = "Failed to add contact"
        print("Error: \(contactDB?.lastErrorMessage())")
    } else {
        status.text = "Contact Added"
        name.text = ""
        address.text = ""
        phone.text = ""
    }
} else {
    print("Error: \(contactDB?.lastErrorMessage())")
}
```

FMDB - извлечение



```
let contactDB = FMDatabase(path: databasePath as String)

if (contactDB?.open())! {
    let querySQL = "SELECT address, phone FROM CONTACTS WHERE name = '\
(name.text!)'"

    let results:FMResultSet? = contactDB?.executeQuery(querySQL,
                                                         withArgumentsIn: nil)

    if results?.next() == true {
        address.text = results?.string(forColumn: "address")
        phone.text = results?.string(forColumn: "phone")
        status.text = "Record Found"
    } else {
        status.text = "Record not found"
        address.text = ""
        phone.text = ""
    }
    contactDB?.close()
} else {
    print("Error: \(contactDB?.lastErrorMessage())")
}
```

Ссылки про хранение данных



ССЫЛКИ

Apple CoreData Guide: <https://developer.apple.com/library/content/documentation/Cocoa/Conceptual/CoreData/index.html>

Realm: <https://realm.io>

NSPredicate cheat sheet: <https://news.realm.io/news/npredicate-cheatsheet/>

SQLite obj-c wrapper: <https://github.com/ccgus/fmdb>

SQLite swift wrapper: <https://github.com/stephencelis/SQLite.swift>

Realm tutorial: <https://www.raywenderlich.com/9220-realm-tutorial-getting-started>

CoreData tutorial: <https://www.raywenderlich.com/7569-getting-started-with-core-data-tutorial>

CoreData: <https://habr.com/en/post/436510/>

Другие базы

YapDatabase: <https://github.com/yapstudios/YapDatabase>

Pencil: <https://github.com/naru-jpn/pencil>