МГТУ
им. Н.Э. Баумана
ТЕХНОПАРК
Mail.Ru Group

Хранение данных

Лекция №11

Павел Тихонов, Артур Сардарян,
Павел Носов

Организационная часть

1. Отметиться
2. После занятия оставить отзыв

Хранение данных в приложении

Как (где) можно хранить данные в приложении?

О чем будем говорить

- UserDefaults
- FileManager
- CoreData
- Realm
- SQLite

Хранение данных

- Кеширование
- Оффлайн работа
- Классный user experience

UserDefaults

UserDefaults

- примитивы
- классы, которые позволяет хранить plist
- потокобезопасен

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

UserDefaults

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
```

NSFileManager

Менеджер для работы с файловой системой

NSFileManager

- список файлов
- удаление
- перемещение
- копирование
- потокобезопасность

NSFileManager

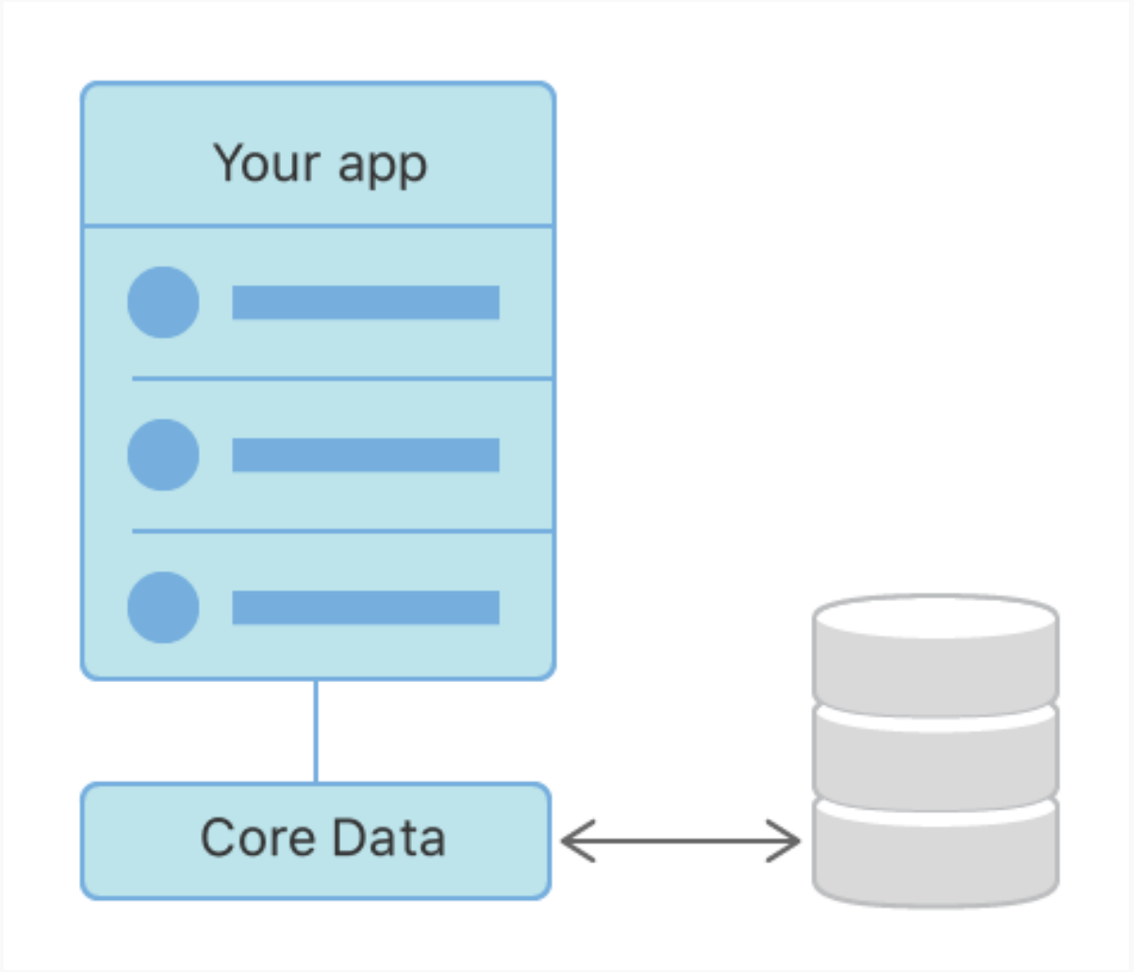

```
// сохранение

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)
}
```

CoreData

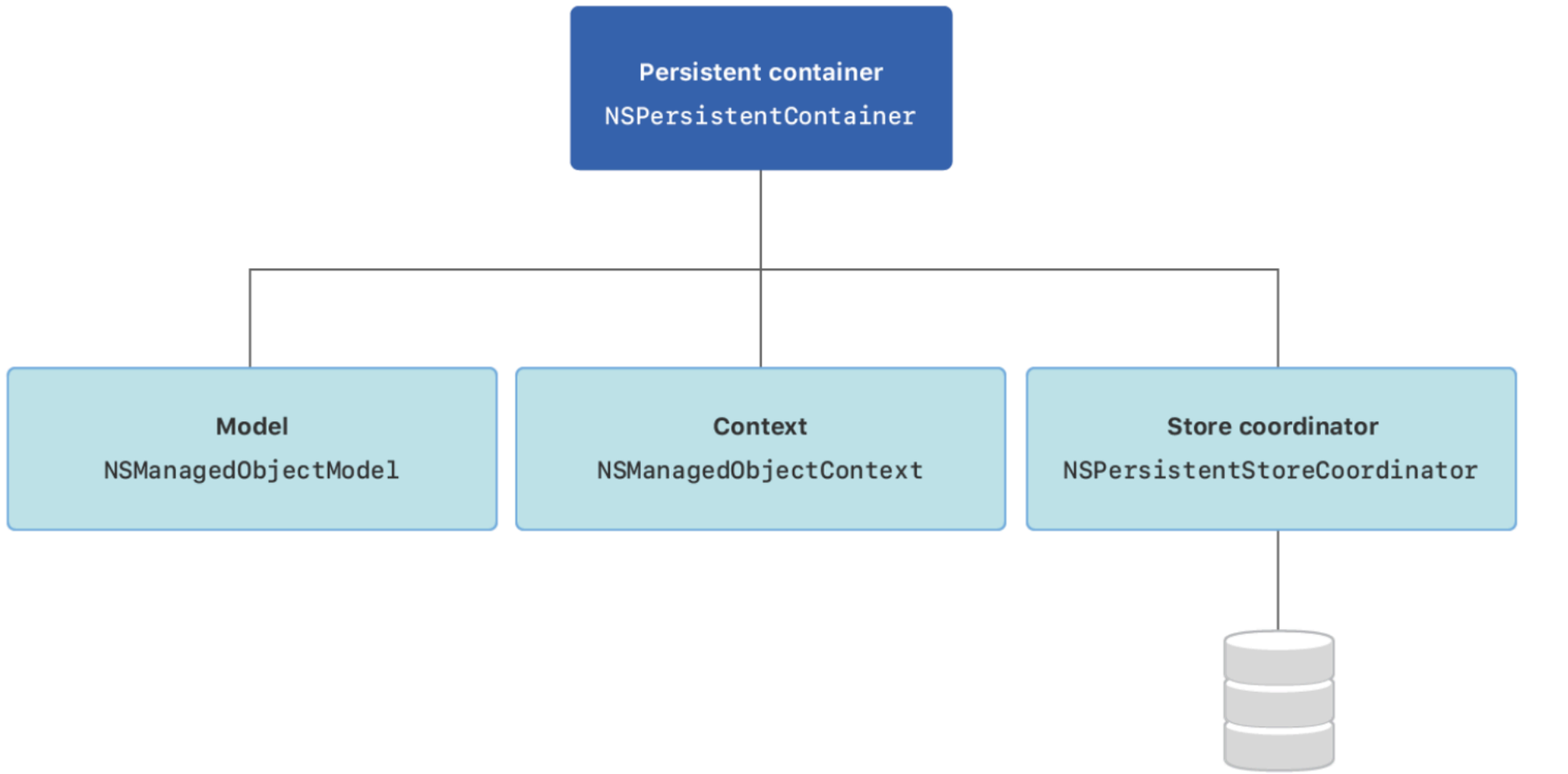
Хранение деревьев объектов

- хранилище
- не является реляционной БД
- несколько типов хранилища:
  - Sqlite
  - Xml
  - память

CoreData



CoreData


Persistent container
NSPersistentContainer
Model
NSManagedObjectModel
Context
NSManagedObjectContext
Store coordinator
NSPersistentStoreCoordinator

CoreData: основные понятия

- **NSManagedObjectModel** — описание структуры данных (объекты, их атрибуты, связи)
- **NSPersistentStore** — конкретная разновидность хранилища (**SQLite**, Binary, In-Memory + свои)
- **NSPersistentStoreCoordinator**
  - отвечает за создание/извлечение существующих данных 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

- **NSManagedObject** — базовый класс для объектов
- **NSFetchRequest** — запрос на выборку объектов
- Все операции делаются через контекст
- Объекты нельзя передавать между потоками напрямую
  - Можно только через ID объекта (objectID)

CoreData

NSFetchedResultsController

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

NSPredicate

Класс, предназначенный для задания требуемых условий фильтрации

Его можно создать:

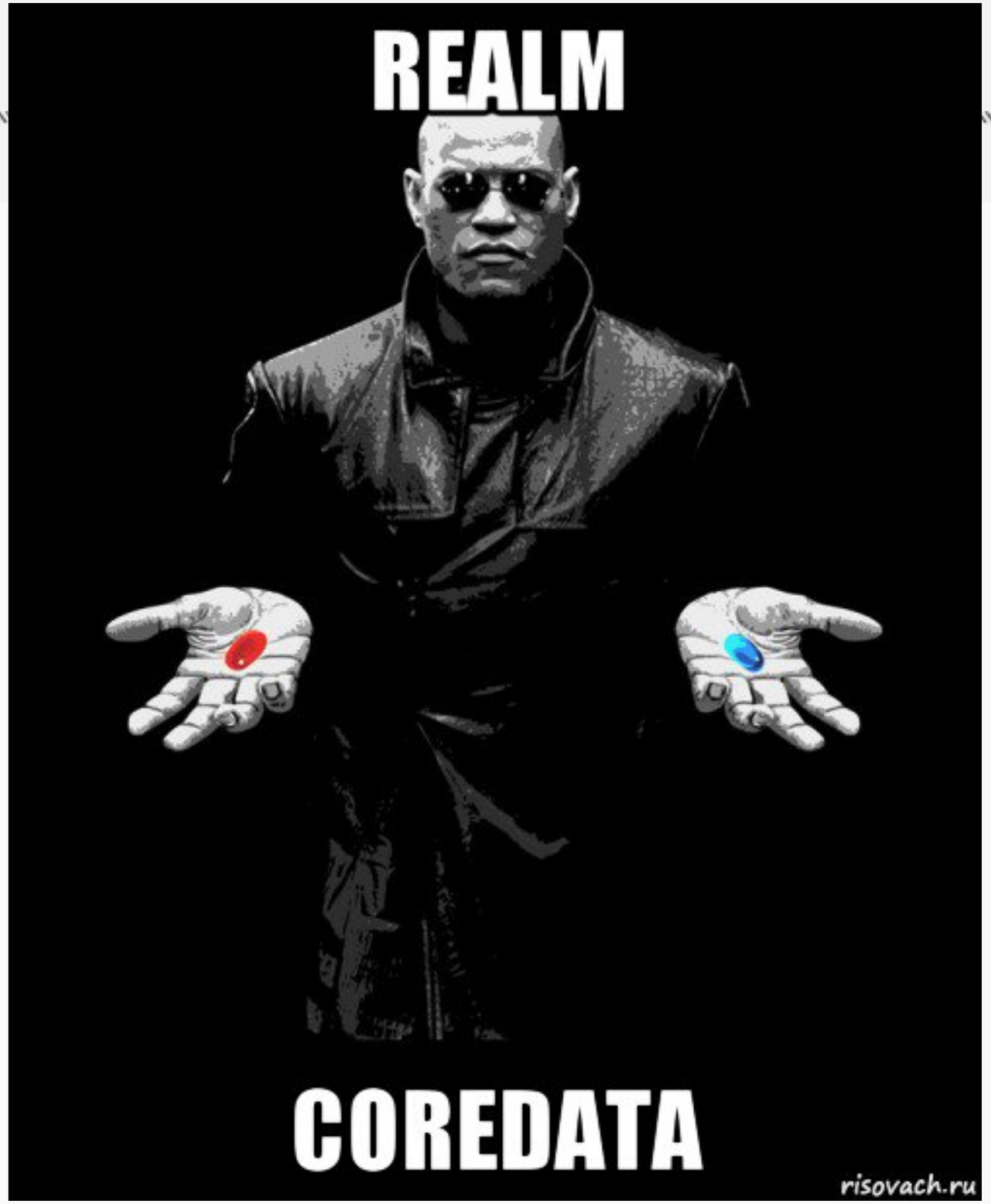
- из строки с условием
- блоком кода, который отдаёт Bool в зависимости от того, подходит объект или нет
- как логическую комбинацию других предикатов
- Подробнее см. ссылки

Realm

- 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)

Realm

- **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 **NSFetchedResultController** это нотификации


REALM
COREDATA
risovach.ru

SQLite

- 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())")
    }
```

Полезные ссылки

- https://developer.apple.com/library/archive/documentation/Cocoa/Conceptual/CoreData/index.html#//apple_ref/doc/uid/TP40001075-CH2-SW1 — Apple CoreData guide
- https://www.hackingwithswift.com/read/12/2/reading-and-writing-basics-userdefaults — User Defaults basics
- https://realm.io/docs — документация Realm
- https://www.raywenderlich.com/7569-getting-started-with-core-data-tutorial — CoreData tutorial
- https://www.raywenderlich.com/9220-realm-tutorial-getting-started — Realm tutorial
- https://github.com/ccgus/fmdb — SQLite Obj-C wrapper
- https://github.com/stephencelis/SQLite.swift — SQLite Swift wrapper
- https://github.com/yapstudios/YapDatabase — YapDatabase
- https://nshipster.com/nspredicate/ — статья про NSPredicate