Лекция №5

Хранение данных. Часть 1

Короткий Егор

образование

О чем будем говорить

- UserDefaults
- FileManager
- Keychain

Хранение данных в приложении

Как (где) можно хранить данные в приложении?

Хранение данных

- Кеширование
- Офлайн-работа
- Классный user experience

UserDefaults

UserDefaults

- примитивы (array, bool, data, dictionary, float, integer, object, stringArray, string, double + url)
- классы, которые позволяет хранить plist
- потокобезопасен
- безопасность (слабая)

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

Базовый кейс

Авторизация. Как сделать, чтобы мы знали, залогинен ли юзер?

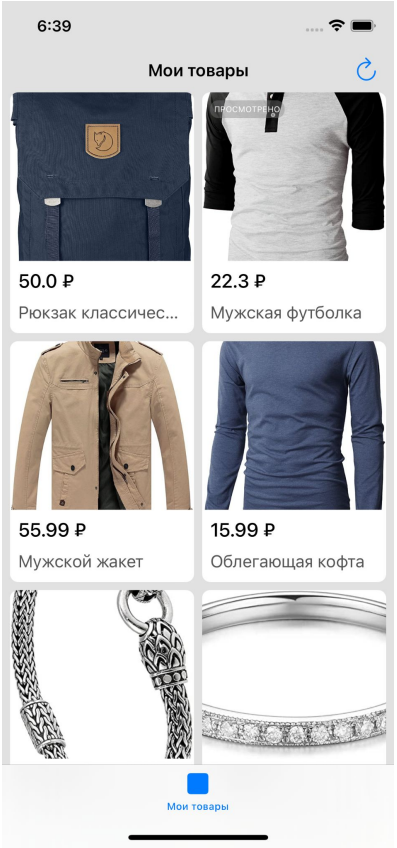
UserDefaults

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
```

UserDefaults. Кейс

Просмотренные товары



UserDefaults. Механизм уведомлений

```
func markAsViewed(with id: String) {
    let key = "\(viewedKey)\(id)"

    UserDefaults.standard.set(true, forKey: key)

    NotificationCenter.default.post(name: Self.notificationKey, object: nil)
}
```

```
func startObserveViewedProducts() {
    NotificationCenter.default.addObserver(self,
                                           selector: #selector(didMarkAsViewedProduct),
                                           name: ProductViewedManager.notificationKey,
                                           object: nil)
}
```

FileManager

Менеджер для работы с файловой системой

NSFileManager

- список файлов
- удаление
- перемещение
- копирование
- потокобезопасность

FileManager. Путь

```
private func documentsDirUrl() -> URL? {
    try? FileManager.default.url(for: .documentDirectory,
                                 in: .userDomainMask,
                                 appropriateFor: nil,
                                 create: false)
}
```

FileManager. Чтение

```
func read() {

    guard let url = documentsDirUrl() else {
        return
    }

    let fileUrl = url.appendingPathComponent(filename)

    guard let content = try? String(contentsOf: fileUrl, encoding: fileEncoding) else {
        titleLabel.text = emptyLabelText()
        return
    }

    titleLabel.text = text
}
```

FileManager. Запись

```
func save() {
    guard let url = documentsDirUrl() else {
        debugPrint("failed to get documents dir url")
        return
    }

    let fileUrl = url.appendingPathComponent(filename)
    let manager = FileManager.default

    var contents: String = "Some content VK"

    do {
        try contents.write(to: fileUrl, atomically: true, encoding: fileEncoding)
    } catch {
        debugPrint(error)
    }
}
```

FileManager. Кейс

Разработаем свой кэш для хранения картинок

FileManager. Кейс

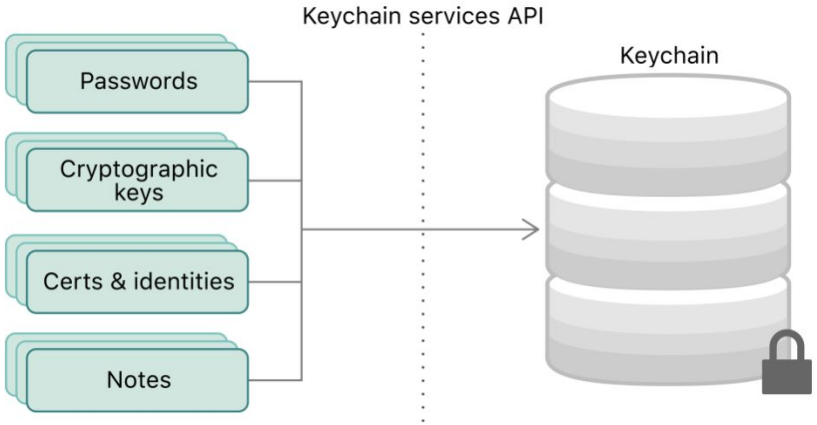
```swift
final class FileManagerImageCache {
    private let fileManager = FileManager.default
    private let saveQueue = DispatchQueue(label: "com.vk.FileManagerImageCache.saveImages", qos: .utility, attributes: [.concurrent])

    private func path() -> URL? {
        return try? fileManager.url(for: .cachesDirectory,
                                    in: .userDomainMask,
                                    appropriateFor: nil,
                                    create: true)
    }

    private func imageHash(with key: String) -> String {
        let charset = CharacterSet.alphanumerics.inverted

        return key.components(separatedBy: charset).joined()
    }
}
```

FileManager. Кейс

```
extension FileManagerImageCache: ImageCacheDescription {
    func obatin(with key: String) -> UIImage? {
        guard let url = path() else {
            return nil
        }

        let imageUrl = url.appendingPathComponent("\(imageHash(with: key))")

        guard
            fileManager.fileExists(atPath: imageUrl.path),
            let data = try? Data(contentsOf: imageUrl)
        else {
            return nil
        }

        return UIImage(data: data)
    }

    func save(object: UIImage, for key: String) {
        guard let url = path() else {
            return
        }

        let imageUrl = url.appendingPathComponent("\(imageHash(with: key))")

        saveQueue.async {
            let data = object.pngData()

            try? data?.write(to: imageUrl, options: [.atomic])
        }
    }
}
```

Keychain

Keychain — специализированная база данных для хранения метаданных и конфиденциальной информации.



https://github.com/evgenyneu/keychain-swift

https://github.com/matthewpalmer/Locksmith

https://habr.com/ru/post/351116

QA

образование