Илья Князьков

iOS Developer

Storage in iOS application

Persistent

Базы данных

1. SQL (реляционные)
2. NoSQL (не реляционные)

SQL	NoSQL
SQLite CoreData	Realm

Persistent

CoreData

```swift
import Foundation
import CoreData

@objc(Entity)
public class Entity: NSManagedObject {

    @NSManaged public var justAttribute: String?

    @nonobjc public class func fetchRequest() -> NSFetchRequest<Entity> {
        return NSFetchRequest<Entity>(entityName: "Entity")
    }

}

// MARK: - Identifiable

extension Entity: Identifiable { }
```

```swift
import CoreData

public struct AccessToCoreDataAdapter {

    // MARK: - Private Properties

    private let persistentContainer: NSPersistentContainer

    // MARK: - Internal Properties

    static let shared = AccessToCoreDataAdapter()
    var mainContext: NSManagedObjectContext {
        return persistentContainer.viewContext
    }
    var backgroundContext: NSManagedObjectContext {
        persistentContainer.newBackgroundContext()
    }

    // MARK: - Initialization

    private init() {
        let container = NSPersistentContainer(name: "SimpleModel")
        container.loadPersistentStores { _, error in
            _ = error.map { fatalError("Unresolved initialization storage error \($0)") }
        }
        self.persistentContainer = container
    }

}
```

```swift
func writeAndLoadFromStorageContext() throws {
    let request = Entity.fetchRequest()
    let entity = Entity(context: AccessToCoreDataAdapter.shared.mainContext)
    entity.justAttribute = "Attribute for test"

    try AccessToCoreDataAdapter.shared.mainContext.save()

    let entities = try AccessToCoreDataAdapter.shared.mainContext.fetch(request)

    debugPrint(entities.first?.justAttribute)
}
```

Persistent

Key-value storage

1. UserDefaults API
2. KeyChain API
3. FileManager API

Persistent UserDefaults

```
UserDefaults.standard.set("ExampleValue", forKey: "ExampleKey")
UserDefaults.standard.value(forKey: "ExampleKey") /// return `"ExampleValue"`
```

UserDefaults, путь до *.plist файла в симуляторе

Bundle ID приложения

ID запущенного симулятора

/Users/{username}/Library/Developer/CoreSimulator/Devices/{simulatorId}/data/Containers/Data/Application/{applicationId}/Library/Preferences/{bundleId}.plist

Имя пользователя

ID установленного приложения

Persistent KeyChain

```
public func setValue(for key: String, value: Value?) throws {
    try removeValue(for: key)
    guard let value = value else {
        return
    }
    let queryOnSave: [String: AnyObject] = [
        kSecAttrService as String: Constants.allocatedkeyChainScopeKey as AnyObject,
        kSecAttrAccount as String: key as AnyObject,
        kSecClass as String: kSecClassGenericPassword,
        kSecValueData as String: try startEncodeProcess(from: value) as AnyObject
    ]

    let status = SecItemAdd(queryOnSave as CFDictionary, nil)
    try throwError(from: status)
}`
```

KeyChain, путь до *.keychain-db(SQLite) файла в системе

/Users/{username}/Library/Keychains/login.keychain-db

Имя пользователя

Persistent FileManager

```
enum FileManagerError: Swift.Error {
    case fileExist
}

func saveFile(_ file: Data, urlPath: URL) throws {
    guard !isFileExist(on: urlPath) else {
        throw FileManagerError.fileExist
    }
    try FileManager.default.createDirectory(at: urlPath, withIntermediateDirectories: true)
    try file.write(to: urlPath)
}
```

Temporary

1. NSCache API
2. URLCache API

Temporary

NSCache

```
final class TestValueWrapper {
    var value = 2
}

let cache = NSCache<NSString, TestValueWrapper>()
cache.setObject(TestValueWrapper(), forKey: "ExampleKey")
cache.object(forKey: "ExampleKey") /// `TestValueWrapper()`
```

URLCache

```
let urlCache = URLCache()

let response = URLResponse()
let cachedResponse = CachedURLResponse(response: response, data: Data())
let cachableRequest = URLRequest(url: try URL("htttps://www.google.com"))
urlCache.storeCachedResponse(cachedResponse, for: cachableRequest)

urlCache.cachedResponse(for: cachableRequest) /// return instanse of `CachedURLResponse`
```

Перейдем к практике

1. Закешируем токен
2. Напишем логику проверки и перезапроса токена
3. Закешируем запросы в приложении


EDUCATION
Surf