курс development 12 недель

iOS: разработка приложений с 0

r_d

iOS программа курса 20 занятий

1
Swift: начало

2
ООП: основы

3
Создание iOS-прило-
жения в Xcode

4
Создание интерфейса iOS-приложения

5
Динамические интерфейсы, часть 1

6
Динамические интерфейсы, часть 2

iOS программа курса 20 занятий

7

Динамические интерфейсы, часть 3

8

Навигация в приложении, часть 1

9

Навигация в приложении, часть 2

10

Анимации в iOS

11

Работа с памятью в iOS

12

Многозадачность в iOS, часть 1

iOS программа курса 20 занятий

13
Многозадачность в iOS, часть 2

14
Дебаг iOS-приложения

15
Тестирование

16
Хранение данных в приложении

17
Работа с сетью в приложении

18
Сборка приложения

iOS программа курса 20 занятий

19

Современные архитектуры для iOS приложений

20

Защита курсовых проектов

Хранение данных в iOS приложении

- Для чего хранить данные в iOS приложениях локально
- File System для хранения “больших” данных
- UserDefaults для хранения “небольших” данных
- Core Data для хранения “небольших” данных**

Для чего хранить данные в iOS приложениях локально

- Существует целый класс задач по хранению данных внутри приложения. Эти данные могут быть использованы единожды или многократно, в одном месте в приложении либо во множестве.

- Идея проста — мы хотим что-то сохранить либо: а) на этапе разработки приложения (аудио, видео, изображения, иконки) либо б) во время работы приложения (конкретные данные о вашем пользователе, ваша аватарка, сообщение от жены в месенджере).
-
- В первом случае мы хотим дать доступ к ресурсам моментально,во втором — получить ресурсы только единожды, а затем также иметь к ним моментальный доступ.

Для чего хранить данные в iOS приложениях локально

Также стоит сразу упомянуть, что большинство тех приложений, что каждый из нас использует ежедневно поддерживают режим “offline-first”.

Offline-first - режим, когда последних данных (с веб-сервера либо другого источника) ещё нет, но мы хотим дать пользователю приложение в максимально “рабочем” состоянии.

К примеру, если вы впервые скачаете и запустите Instagram, у вас подгрузится лента с изображениями ваших друзей. Затем, когда вы закроете приложение и в след.раз, когда снова зайдете в него, у вас не будет соединения с сетью (либо соединение будет ооочень медленным), вы увидите тот контент, который был загружен в прошлый раз.

Для чего хранить данные в iOS приложениях локально

Существует 3 инструмента “из коробки”, чтобы хранить данные:

- File System
- NSUserDefaults
- CoreData

File System для хранения “больших” данных

- Файловая система необходима для длительного хранения файлов, приложений и системных файлов ОС. File System доступна для всех приложений на iOS, как единый ресурс для хранения файлов различных форматов данных.

- Операционная система iOS, начиая с 10.3, работает с файловой системой формата APFS (ранее была HFS+). APFS присутствует в macOS начиная с High Sierra и выше.

File System для хранения “больших” данных

- Для работы с файловой системой вам в первую очередь необходимо импортировать библиотеку Foundation. Так у вас будет доступ к необходимому API, предоставляемый Apple.

- Файловая система умеет хранить и обрабатывать тысячи и миллионы файлов, поэтому её структура базируется на иерархии. Иерархии строятся с помощью директорий (папки), в которых могут хранится как файлы так и другие директории.

File System для хранения “больших” данных

- В таких операционных системах как iOS / macOS / etc от Apple существует специальный механизм, который не позволяет хранить данные вне контекста нашего приложения. Этот механизм называется Sandbox (“песочница”).

- Мы не можем записать наш файл в директорию, к которой у нас нет специальных прав безопастности (security privileges).

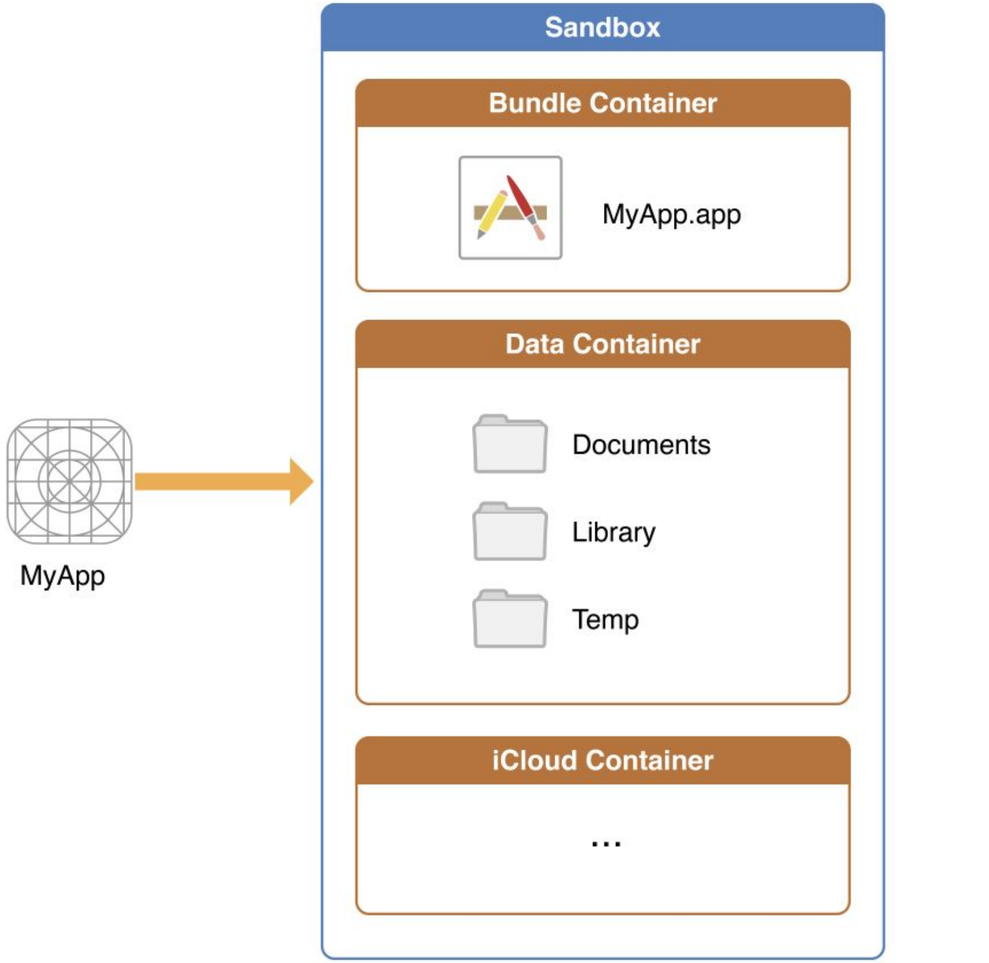
- Если вы нарушаете правило и ваше приложение обходит система Sandbox на iOS — ваше приложение не будет опубликовано в App Store.

- Для macOS действуют такие же правила, но для десктопных приложений существует ряд альтернатив для дистрибуции, кроме публикации в App Store.

File System для хранения “больших” данных

- Наше iOS приложение стандартно имеет доступ только к нескольким директориям, которые находятся внутри самого приложения.
- Во время установки приложения “установщик” создаёт набор таких папок.

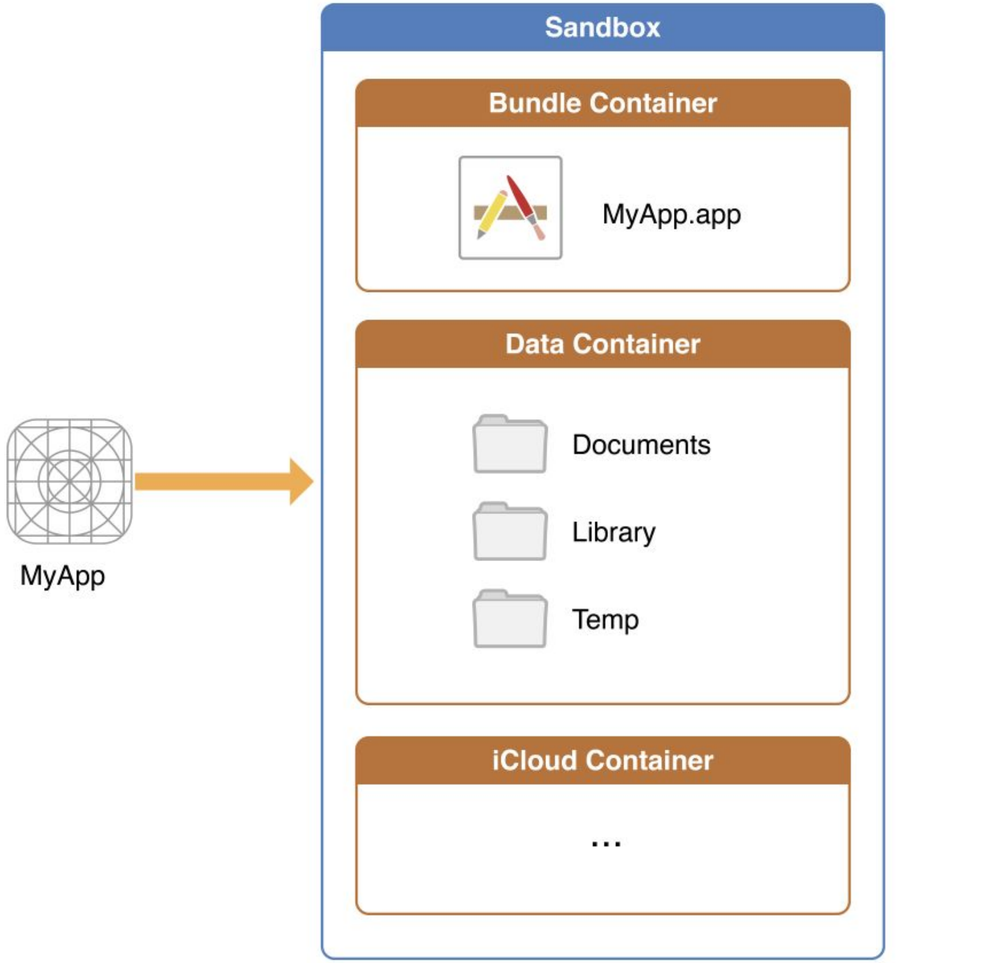
Figure 1–1 An iOS app operating within its own sandbox directory



File System для хранения “больших” данных

- Каждая директория создаётся для определенных целей.
- Директория “Bundle Container” содержит весь bundle приложения.
- Директория “Data Container” содержит набор папок, куда можно складывать скачанные файлы во время работы приложения.

Figure 1–1 An iOS app operating within its own sandbox directory

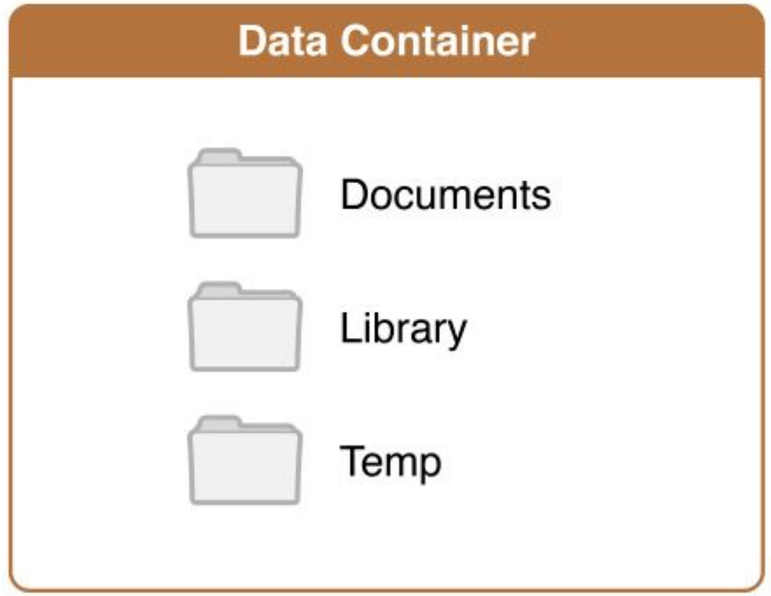


File System для хранения “больших” данных

- Данные о пользователе можно складывать в Documents/. Всё что может быть создано пользователем или скачано для корректной работы пользователя: файлы конфигурации, базы данных и прочее.

- Файлы для поддержания корректной работы программы стоит хранить в Library/Application support/. Все что стоит скрыть от пользователя, но необходимо хранить.

- Все данные, которые мы хотим временно хранить (в рамках одной рабочей сессии) стоит хранить в Temp/ (либо tmp/).



UserDefaults для хранения “небольших” данных

- UserDefaults — простоей key-value хранилище, в котором хранятся (1) статические данные для старта приложения и (2) динамические данные для работы приложения с чем-то*.

- Статические данные — это конфигурации и права для использования различных системных интерфейсов, таких как камера, GPS, контакты телефонной книги и прочие системные вещи.

- Динамические данные — то, что мы можем сохранить в User Defaults во время работы нашего приложения (после запуска). Например, мы можем получать данные геопозиции каждые 30 секунд и записывать их в UserDefaults. При перезапуске приложения у нас будут сохранены актуальные данные о месторасположении.

UserDefaults для хранения “небольших” данных

- Доступ к UserDefaults должен осуществляться только через статический объект UserDefaults.standard. Если создать собственный экземпляр класса UserDefaults и дать доступ к файлу Info.plist, то существует высокая вероятность переписать что-то в Info.plist, что повлияет на работоспособность приложения.

- Можно создавать свои экземпляры класса UserDefaults, но которые будут ссылаться на новые .plist файлы (можно так же создать и хранить в проекте).

UserDefaults для хранения “небольших” данных

- UserDefault по-умолачнию умеет хранить и работать со всеми примитивными типами данных, такими как floats, doubles, integers, Boolean values, и URLs.

- Важно: .plist файлы оперируют типами из языка Objective-C. Поддерживаются такие типы как NSString, NSData, NSNumber, NSDate, NSArray и NSDictionary. Это те типы, которые могут храниться как value в этом таком хранилище. Тип значения Key — это NSString.

- Значения, которые возвращаются из UserDefaults являются иммутабельными (т.е. неизменяемыми). Это работает как для чтения из UserDefaults (метод stringForKey:), так и для записи (метод setObject:forKey:).

Работа с UserDefaults. Чтение настроек

В обычных обстоятельствах NSUserDefaults предельно просты.

- Чтение настроек из UserDefaults
- Сохранение пользовательских настроек в UserDefaults

Работа с UserDefaults. Чтение настроек

- Если есть настройка управляющая некоторой частью кода, вы просто вызываете подходящий метод (*-objectForKey:* или один из оберточных методов для конкретного типа).

- Если обнаружилось, что вам требуется сделать что-то ещё для получения настройки, то следует сделать шаг назад и взвесить всё ещё раз.

Работа с UserDefaults. Чтение настроек

1. Кэширование значений из NSUserDefaults обычно не нужно, поскольку чтение и так предельно быстрое.

2. Действия в ответ на изменение значения почти никогда не нужны, поскольку предназначение любых настроек — контролировать что программа делает, а не заставлять действовать.

3. Написание кода для обработки случая «значение не установлено» также в общем случае не нужно, так как можно зарегистрировать значение по умолчанию.

Работа с UserDefaults. Сохранение настроек

- Аналогично, когда пользователь изменяет некоторую настройку, вы просто вызываете `-setObject:forKey:` (или одну из его типо-специфичных оберток).

- Если обнаружилось, что вам требуется что-то ещё, то, опять же, вероятно, это не нужно!

- Пользователи обычно не способны изменять настройки так быстро, чтобы «пакетирование» любого вида было бы полезно для производительности. Реальная запись на диск асинхронна, и UserDefaults производят слияние изменений в одну операцию записи автоматически.

Работа с UserDefaults

```
let defaults = UserDefaults.standard
defaults.set(25, forKey: "Age")
defaults.set(true, forKey: "UseTouchID")
defaults.set(CGFloat.pi, forKey: "Pi")

defaults.set("Paul Hudson", forKey: "Name")
defaults.set(Date(), forKey: "LastRun")
```

Работа с UserDefaults

```
let age = defaults.integer(forKey: "Age")
let useTouchID = defaults.bool(forKey: "UseTouchID")
let pi = defaults.double(forKey: "Pi")
```

CoreData для хранения “небольших” данных**

- “Core Data — фреймворк от компании Apple, встроенный в операционную систему iOS, MacOS, который позволяет разработчику взаимодействовать с базой данных. Был представлен компанией Apple с анонсом Mac OS X 10.4 Tiger и iOS с iPhone SDK 3.0.

- Позволяет данным быть организованными в Сущность-Атрибут-Значение (EAV). Управление данными может быть осуществлено с помощью манипуляций сущностей и их взаимосвязей.” (c)

- “Persist or cache data and support undo on a single device.” (c)

Core Data для хранения “небольших” данных**

- Core Data описывает данные, которые хранятся в iOS приложении, код может манипулировать для сохранения и записи данных в приложении.

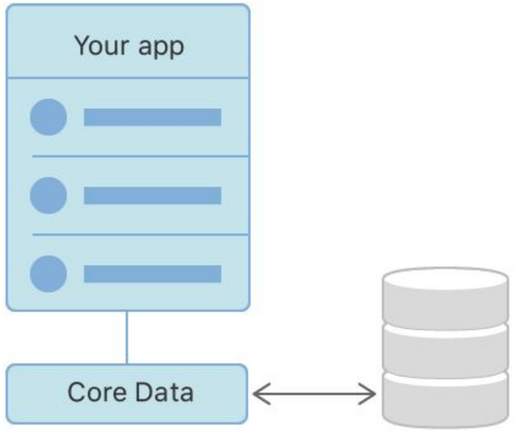
- Модель БД создается в Interface Builder. Код пишется на Objective-C или Swift. Core Data организован в огромные классы.

Core Data для хранения “небольших” данных**

- Суть работы фреймворка, как внутренней БД проста: создание модели, и при добавлении новых элементов сохранение с помощью метода saveContext(); в модель с помощью ниже приведенных методов в таблице.

- Приложение не может работать без БД, так как при выходе из него данные будут утрачены. Это и является главной целью данного фреймворка — хранение данных. Также есть аналоги Core Data — Realm и SQLite.

- Core Data может конвертировать данные в XML, бинарный код, SQLite для хранения. Core Data схемы стандартизированы.



Demo Time 🎉 🎉 🎉