Storage

Storage

There are several ways to *persist* data – meaning to retain information across multiple invocations of your app. Essentially, this means storing data to the hard drive of your phone.

- User Defaults
- Core Data
- File Manager
- Keychain
- Firebase and Firestore

User Defaults

- User defaults are a way to store simple key/value pairs on the hard drive of a device
  - Similar to a dictionary, but with persistence
  - Useful for small amounts of data (<100 KB)
- Very easy mechanism, but limited capabilities
- Typically used for application configuration data
- Data types you can store in User Defaults:
  - NSData
  - NSString; String
  - NSNumber; UInt/Int/Float/Double/Bool
  - NSDate
  - NSArray; Array
  - NSDictionary; Dictionary

User Defaults

- Typically used to remember user preferences, or to store simple values, such as how many times the app has been launched, or when the user first launched the app.
- Not for storing large amounts of data, images, or other large data structures. Booleans, numbers, and strings are usually okay because they're small and very well-suited for key-value storage.
- Never store sensitive data in User Defaults, because it's not encrypted, and space is shared across apps.
- Reading from User Defaults is fast, because the data is automatically loaded into memory at app launch.

Core Data

Core Data is an iOS framework that provides powerful functionality to query and persist non-trivial data.

- Like User Defaults, used to store data in a database on the hard drive of the device
- Makes it simple and fast to work with large amounts of data (> 100KB)
- Useful for storing non-trivial data structures
- Technically not a true “database”, but you can do database-like stuff.
- Lets developers manage data in a database in an object-oriented way

Core Data (cont.)

- With Core Data, you can easily map the objects in your apps to the table records in the database without even knowing any SQL.
- Multiple choices for the “backing store”:
  - Sqlite (default)
  - XML
  - Binary
- That said, it is much more than a storage mechanism:
  - Manages data object life cycles
  - Tracks changes to the data
  - Effortless undo support
  - Saves data to disk

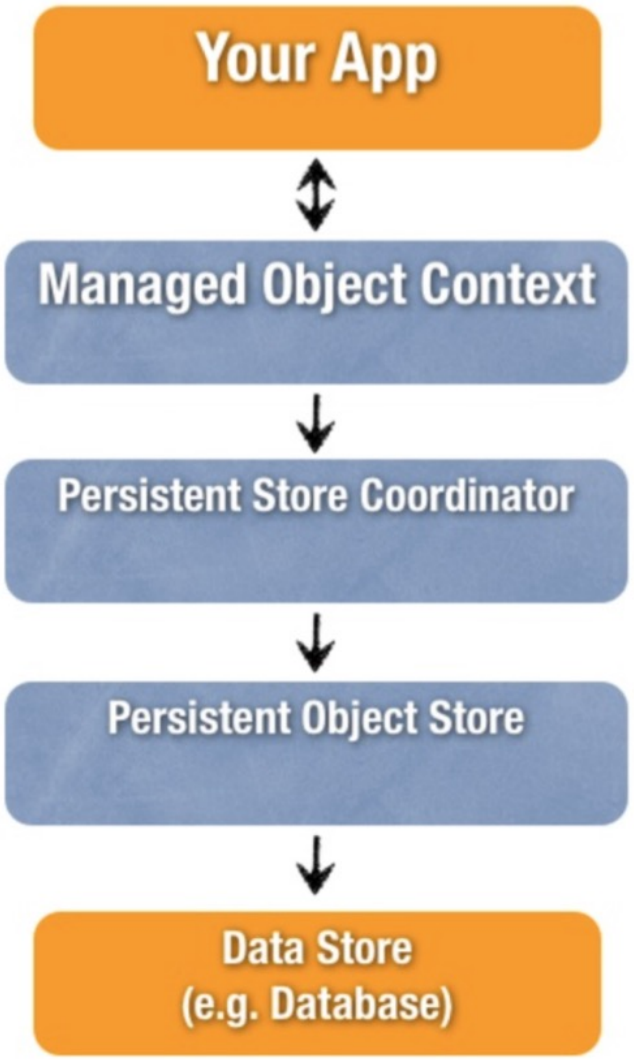
Core Data Architectural Overview

Managed Object Model

- Defines the structure of the data – the data schema (data type, relationships)
- Use the Data Model Design Tool in Xcode to define the models: it's a way of creating an object graph backed by a database.

Managed Object Context

- Acts as a temporary scratch space
- Objects fetched from the persistent store are stored in the context, where we can manipulate them
- Changes are monitored here



File Manager

- The file system typically has a large amount of storage available, but it can be relatively slow to read or write large amounts of data. If you need something quicker, you should consider SQLite or in-memory storage.
- Disk storage is typically used for larger amounts of data than User Defaults, such as:
  - images, videos, or large JSON files: binary data that has no properties that you might want to query
  - text files or PDFs
  - data structures that you might make conform to `Codable` so they can be easily converted to `Data`, which can then be written to a file on the file system.

File Manager (cont.)

- Disk storage is not as secure as you might think. Apple encrypts devices when they are locked, but as soon as the device is unlocked and your app becomes active, the encryption no longer applies. Consequently, you should avoid storing sensitive data on disk.

Keychain

- The keychain is the place where iOS stores all kinds of sensitive user data, such as usernames, passwords, certificates, `OAuth` tokens, etc. It is probably the most secure place for storing data.
- The keychain APIs are very non-trivial to navigate and use. Fortunately, there are frameworks available on CocoaPods, Carthage, and SwiftPM that make it much easier.

Firebase

Firebase is a BaaS (Backend-as-a-Service) platform which is a NoSQL database that is better and conventional compared to relational databases.

Note that this means Firebase is a database that's stored on a server somewhere, *not* on the iOS device.

- Firebase is for shared data. It's the storage mechanism of choice when data needs to be accessed by multiple devices.
- Remember that data in User Defaults and Core Data is stored on the hard drive of ONE device, and is therefore not accessible from other devices.

Firebase vs. Firestore

There are two types of noSQL databases used for storing data in Firebase:

Realtime Database

- Stores data as one large JSON tree.
- Simple data is very easy to store.
- Complex, hierarchical data is harder to organize at scale.

Cloud Firestore

- Stores data as collections of *documents*, which are very similar to JSON.
- Complex, hierarchical data is easier to organize at scale, using subcollections within documents.

User Defaults

User Defaults

You use the singleton object `UserDefaults.standard` to access the user defaults store.

1. Associate the singleton object with a variable:

```
let defaults = UserDefaults.standard
```

2. Call a method to save the data you want to store, providing key and value

```
defaults.set(<value>, forKey:<keyName>)
```

`<value>` is the data you want to store (most any type)
`<keyName>` is a String that you want to use to identify the data

Writing to User Defaults

```
// Get a reference to the global user defaults object
    let defaults = UserDefaults.standard

// Store various values
    defaults.set(900, forKey: "userId")
    defaults.set(1275.55, forKey: "total")
    defaults.set("University of Texas", forKey:
        "name")
```

Reading From User Defaults

There are several convenience methods for retrieving values from User Defaults:

```
// Retrieve the previously stored values
    let retrievedUserId =
         defaults.integer(forKey: "userId")

    let retrievedTotal =
         defaults.double(forKey: "total")

    let retrievedName =
         defaults.string(forKey: "name")
```

etc.

Removing From User Defaults

```
// Remove the objects from User Defaults
    defaults.removeObject(forKey: "userId")

    defaults.removeObject(forKey: "total")

    defaults.removeObject(forKey: "name")
```

User Defaults

Some caveats:

- The methods that return Bool, Int, Float, and Double do not return Optionals. In those cases, they return sentinel values appropriate to their types which are false (or NO), 0, 0.0, and 0.0 respectively.
- Methods that return AnyObject indicates you'll need to cast to the correct type before using