Data Storage

Data Storage

Data can be stored:

- As **unsecured** file in device folder
- As **unsecured** data using **User Defaults**
- As **secured** data in **Keychain Acces**s
- In Database (for big amount of data)

Saving In Folder

To save some object in folder, it must:

- Conform to **Codable** protocol
- Convert object to data using one of multiple options (**JSONEncoder**, **NSKeyArchiver**)
- Create some specific file url using **FileManager**

```
let archiveURL = FileManager.default.urls(for: .documentDirectory,
                                          in: .userDomainMask).first

archiveURL!.appendingPathComponent("file")
```

- Save state to your url

```
data.write(to: url, options: .noFileProtection)
```

User Defaults

Used to store info that don’t need any security: app settings, language, theme etc.

- **Do not** use to store any passwords or emails.
- Class provides convenience methods for types - **Float, Double, Integer, Bool, URL.**
- A default object must be - **NSData, NSString, NSNumber, NSDate, NSArray, or NSDictionary.**

If you want to store any other type of object, you should archive it to create an instance of **NSData**.

Keychain

The keychain is the best place to store small secrets, like passwords and cryptographic keys.

Keychain is used to store:

- Sensitive data
- Passwords
- Emails
- Tokens
- App security keys etc.

Database

Database is used to store content like users, images, posts, videos etc.

Here are the most popular databases in iOS:

- SQLite - low-level database
- Core Data - native Apple’s database, ORM for SQLite
- CoreStore - wrapper over Core Data
- Realm - no sql database

Comparison

SQLite:
- Safe access from multiple processes and threads
- Operates on data, stored on disk.
- Can Drop table and Edit data without loading them in memory.

Core Data:
- Uses more memory than SQLite
- Uses more storage space than SQLite
- Faster in fetching records than SQLite.
- Operates on in memory.(data needs to be loaded from disk to memory)
- Need to load entire data if we need to drop table or update
- Faster than SQLite
- ORM

CoreStore:
- Same as Core Data, but easier and cleaner API.

Realm:
- Speed - faster than SQLite and CoreData.
- Scalability **-** works with large data in no time.

CoreStore vs Core Data samples

Create new object in database

Core Data

```swift
// create new user
container?.performBackgroundTask { context in
    let user = User(context: context)
    user.name = "dron"
    user.age = 22

    try? context.save()
}
```

CoreStore

```swift
// create new user
try? CoreStore.perform (synchronous: { (transaction) -> Void in
    let person = transaction.create(Into<User>())

    person.name = "Dron"
    person.age = 22
})
```

Fetch all objects

Core Data

```swift
// get all
let request: NSFetchRequest<User> = User.fetchRequest()
let users = try? container?.viewContext.fetch(request)
```

CoreStore

```swift
// get all
let users = try? CoreStore.fetchAll(From<User>())
```