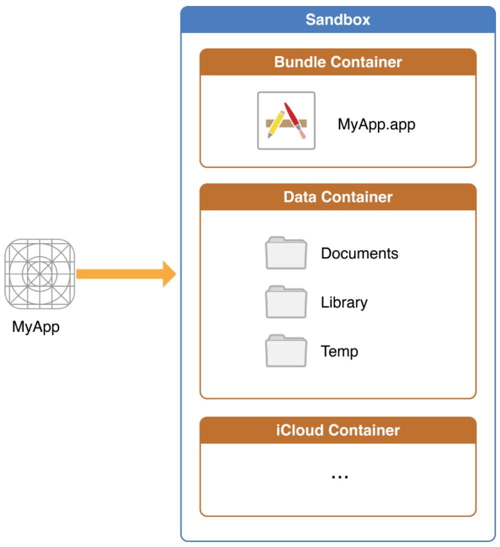
Chapter 3

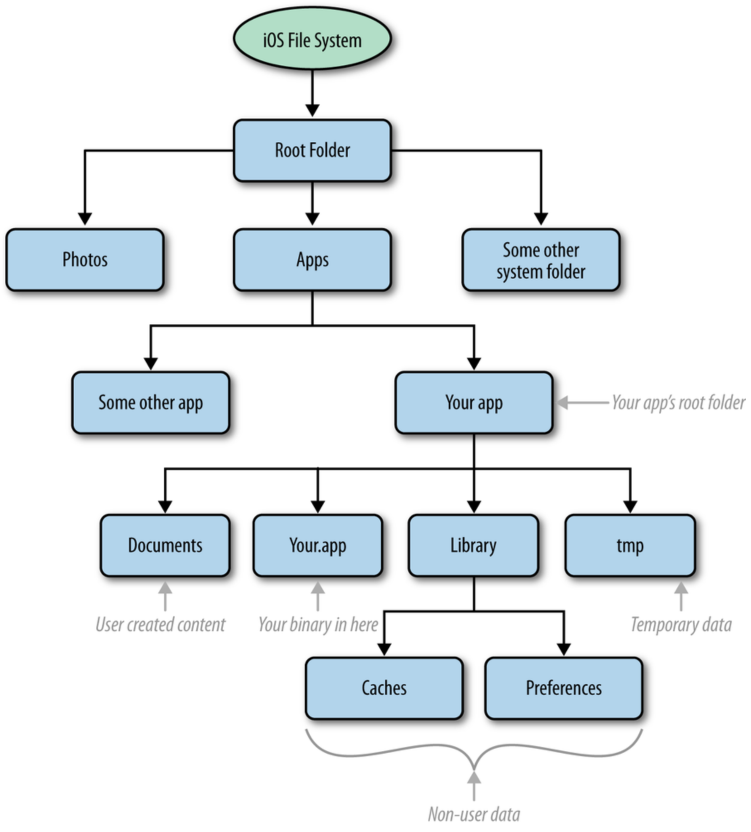
iOS Data Persistence

III.Persistance in iOS

1. iOS File System

The file system in iOS is based on the UNIX file system.The iOS file system is geared toward apps running on their own and an iOS app's interactions with the file system are limited mostly to the directories inside the app's sandbox.





1. Accepting Files and Directories

Typical File Structure

-AppName.app

-Documents/

-Library/

|-Caches

|-Application Support

|-Preferences

-tmp/

1. Using NSFileManager

Using NSFileManager object lets you examine the contents of the file system and make changes: to locate, create, copy, and move files and directories. When specifying the location of files, you can use either NSURL or NSString objects. The use of the NSURL class is generally preferred for specifying file-system items because they can convert path information to a more efficient representation internally.

eg. Using URLsForDirectory Methods of NSFileManager

```
var docPath =
FileManager.default.urls(for: .documentDirectory,
in: .userDomainMask)

var cachePath =
FileManager.default.urls(for: .cachesDirectory,
in: .userDomainMask)

var appsupportPath =
FileManager.default.urls(for: .applicationSupportDirectory,
in: .userDomainMask)
```

Note:Dot directories and files. Any file or directory whose name starts with a period (.) character is hidden automatically.When using a file manager object with a delegate, it is recommended that you create a unique instance of the NSFileManager class and use your delegate with that instance.

I. **Useful File Operations**

a) Moving and Copying Items

- copyItemAtURL:toURL:error:
- copyItemAtPath:toPath:error:
- moveItemAtURL:toURL:error:
- moveItemAtPath:toPath:error:

b) Creating and Deleting Items

-CreateDirectoryAtURL:withIntermediateDirectories:attributes:error:
-createDirectoryAtPath:withIntermediateDirectories:attributes:error:
-createFileAtPath:contents:attributes:-
-removeItemAtURL:error:
-removeItemAtPath:error:

c) Discovering Directory Contents

- contentsOfDirectoryAtURL:includingPropertiesForKeys:options:error:
- contentsOfDirectoryAtPath:error:

2. Using NSSearchPathForDirectoriesInDomains

eg .let documentsPath =
NSSearchPathForDirectoriesInDomains
(.DocumentDirectory, .UserDomainMask, true)[0]

Note: you can user Bundle.main.url(forResource or path(forResource for application resource files if you can't simply access the resource by its filename

eg.

```
let file = URL(fileURLWithPath:
NSBundle.mainBundle().pathForResource("myfile", ofType:
"txt")!)
let dataRaw = NSData(contentsOfURL: file )
let datastr = NSString(data: dataRaw!, encoding:
NSUTF8StringEncoding)
print(datastr)
```

2. Using SQLite for Persistence

To use SQLite for Database Operation, following framework are necessary to import to project file:

-libsqlite3.0.tbd, libsqlite3.tbd, Foundation and SystemConfiguration framework

and, it is necessary using third party frame work, create an objective-C header file and set the header name in the target's Objective-C Bridging Header build setting and declare objective-c header file name in it.

a) **Useful SQL Commands:** Refer to SQL Note
b) **Useful simple and lightweight Third Party SQLite wrapper for Swift:**

https://github.com/FahimF/SQLiteDB
https://github.com/ccgus/fmdb

e.g Using SQLiteDB Wrapper to display the table contents from sqlite dictionary database file, dict.sqlite

```
//set Database name in SQLiteDB.swift with let DB_NAME = "dict.sqlite"
let db = SQLiteDB.sharedInstance
let data = db.query("SELECT * FROM dictionary ")
for row in data
{
  print((row["en"] as! String)+(row["mm"] as! String))
}
```

3. Exercise

1. store the following value in app using UserDefault
   Name, age , NRIC and Address by capturing from user input using textfield
and load the value from the disk and store if the value is already stored

2. Create a projects using SQLite as database to store and retrieve data of the app

3. Personal Finance Project Demo