Persistence & Core Data

Data Persistence

Almost any application you build needs to save data to be useful.

- This is called “data persistence”.
- State that outlives the process that created it.
- Can be done in many ways!

Data Persistence Options on iOS

- Files
- NSUserDefaults
- plist files
- Archiving
- Keychain
- Direct SQL
- Core Data

LIGHTHOUSE LABS

The Filesystem

Bits and bytes on disk

LIGHTHOUSE LABS

Writing to Files

- Any data can be turned into a binary representation.
- NSData is a class used to wrap binary data, and can be saved to disk easily
- This is generally a good fit for large files, with established file format standards.
- E.g. Images, movies, audio, etc.

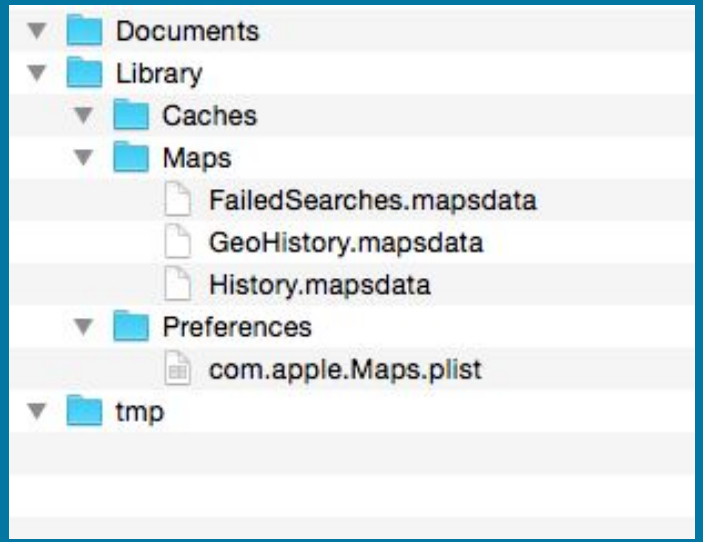
1. Get the location of a writable directory

iOS apps are “sandboxed”

- Only specific directories on disk can be read from / written to.
- Some are flushed out occasionally (temp, caches directories).

1. Get the location of a writable directory

This will be a directory inside your app's sandbox



Documents,
Temporary, or
Cache directory.

1. Get the location of a writable directory

```
NSArray *docDirectories =
            NSSearchPathForDirectoriesInDomains(
                                NSDocumentDirectory,
                                NSUserDomainMask,YES);

NSString* docPath = [docDirectories firstObject];

NSURL *docURL = [NSURL fileURLWithPath:docPath];
```

2. Append your target file’s name

```
NSString *filePath = [docPath
stringByAppendingPathComponent:@”appdata”];

NSURL *fileURL = [docURL
URLByAppendingPathComponent:@”appdata”];
```

This appends “/appdata” to the path or to the URL

3. [optional] Check if file already exists

```
NSFileManager* fm = [NSFileManager
defaultManager];

if ([fm fileExistsAtPath:filePath]){
  NSLog(@”File already exists.”);
} else {
  // write to disk ...
}
```

In some cases this is not enough...

4. Write to disk!

```
[fileManager createFileAtPath:filePath
contents:myData attribute:nil];
```

or using NSData directly

```
[myData writeToFile:filePath atomically:YES];
```

or a number of other methods on foundation classes...

User Defaults

NSUserDefaults

- Intended to store simple user preferences
- Very similar to a dictionary, but it's automatically persisted to disk
- Fairly simple to make it sync via iCloud
- Doesn't store custom objects, can be made to if you *really* want it...
- Doesn't perform well with large data sets

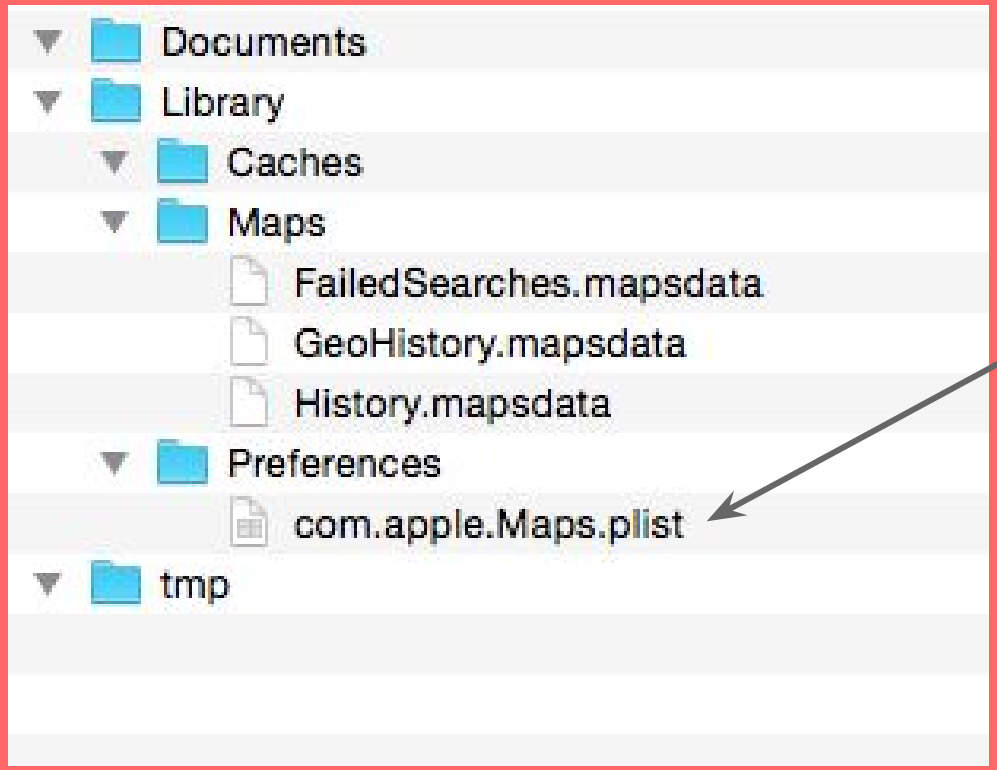
NSUserDefaults

User defaults can store `NSData`, `NSString`, `NSNumber`, `NSDate`, `NSArray`, and `NSDictionary` objects.

Using `setObject:forKey:` and `objectForKey:`

There are convenience methods for boxing and unboxing primitive types as you get/set them.

NSUserDefaults



NSUserDefaults

LIGHTHOUSE LABS

Databases

Databases

Author

id	name	age

Book

LIGHTHOUSE LABS

Databases

Author

id	name	age
1	John	44
2	Mary	34

Book

		title

LIGHTHOUSE LABS

Databases

Author

id	name	age
1	John	44
2	Mary	34

Book

id	author_id	title
0	1	Learn iOS in 4 hrs
1	1	iOS for dummies

Core Data

What is Core Data

Core Data is an object-oriented database.

- Translate objects into table rows (save data).
- Create objects from table rows (load data).
- Query our database for specific bits of data.
- Create & manage relationships between objects.
- Handle changes to our data model between versions of our app

Core Data - Components

- Managed object model
  - a collection of entity descriptions
- Persistent Store Coordinatior
  - uses the MoM to map between sqlite and objects
- Managed Object Context
  - a workspace to load objects into, modify, save.
- Managed Object
  - Our model objects, with properties and relationships

Core Data - Components

- Managed Object Context
  - a workspace to load objects into, modify, save.
- Managed Object
  - Our model objects, with properties and relationships

NSManagedObjectContext

- Groups CRUD operations
- Handles Concurrency
- Handles undo/redo
- Fetches objects from the persistent store

NSManagedObject

- YOUR DATA LIVES HERE!!
- Holds state Information (was inserted, updated, deleted, etc.)
- Lifecycle Hooks: methods called before/after important events (insertion, fetch, save)
- Data validation
- 99% of the time you'll use subclasses, rather than bare NSManagedObjects

NSFetchedResultsController

- We give it a query, a grouping, and a sort order, run it.
- It gets objects, groups and sorts them, and then watches for changes.
- We can read directly from it using NSIndexPaths
- Useful for collections/tables backed by Core Data

Lots more to come!

We’re just not ready for a relationship right now...