Persistence & Core Data

How to Save Data

Intro

Almost any application you build needs to save data to be useful. The storage of application data is called “persistence”.

Wikipedia definition: state that outlives the process that created it.

Data Persistence Options on iOS

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

LIGHTHOUSE LABS

Writing to Files

You can turn any kind of data you have (text, image, etc) into its binary representation (`NSData`). Once you have an `NSData` object, you can store it to a file on the file system, using `NSFileManager`.

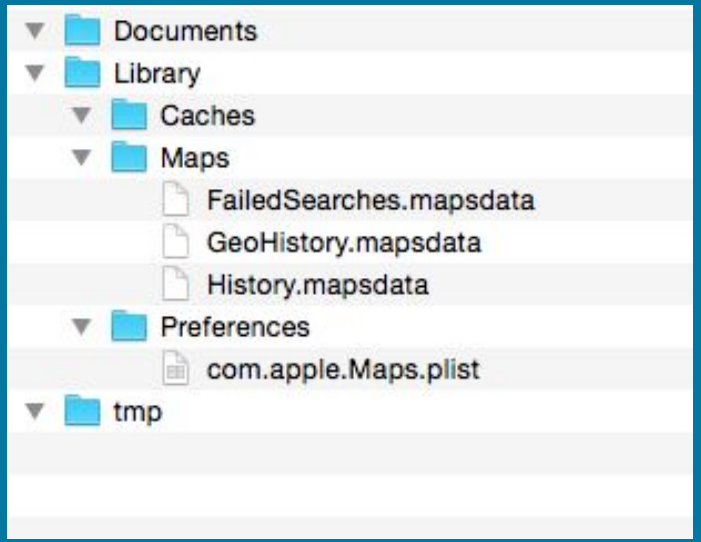
Writing to disk

1. Get the location of a writable directory
2. Append your target file’s name
3. [optional] Check if file already exists
4. Write to disk

1. Get the location of a writable directory

This is 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.

3. [optional] Check if file already exists

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

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

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...

NSUserDefaults

- Intended to store simple user preferences
- Very similar to a dictionary, but it’s automatically persisted to disk
- Very simple to have synced via iCloud
- Does not easily store custom objects, but can be coerced to if you really want it
- Doesn’t perform well if larger data sets are added

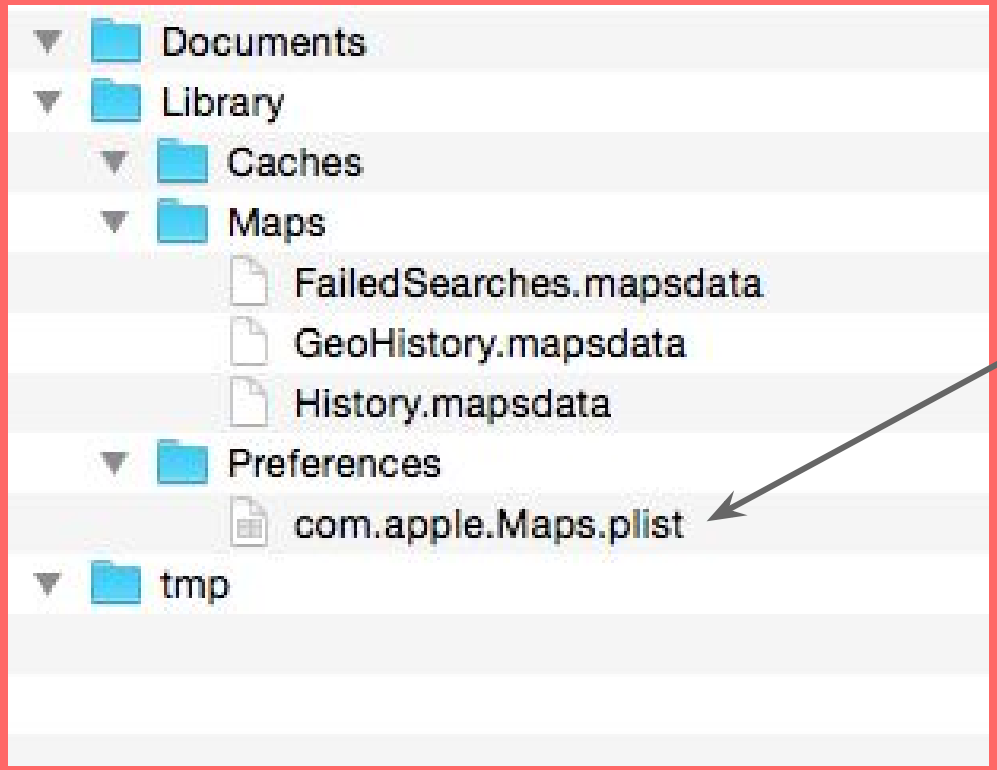
NSUserDefaults

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

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

There are convenience methods for storing primitive types (they're boxed).

NSUserDefaults



NSUserDefaults

LIGHTHOUSE LABS

DEMO - NSUserDefaults

Installing Some Tools

- SQLite Viewer https://itunes.apple.com/ca/app/datum-free/id901631046?mt=12
- OpenSim for opening simulator files: https://github.com/luosheng/OpenSim/releases

Databases

Author

id	name	age

Book

LIGHTHOUSE LABS

Databases

Author

id	name	age
1	John	44
2	Mary	34

Book

id	author_id	title

Databases

Author

id	name	age
1	John	44
2	Mary	34

Book

id	author_id	title
1	1	Learn iOS in 4 hrs
2	1	iOS for dummies

Enter Core Data

Core Data is an “Object Graph” and object lifecycle management framework. It *can* persist to a database.

That’s a mouthful. What it means is that Core Data’s primary job is keeping your model layer logically consistent. It also provides a lot of other solutions that will “reduce your model layer code by 50-70%” (according to Apple).

When we say Core Data keeps your model layer consistent, we mean it...

- Stores the relationship between your objects.
- Provides easy mechanism for validating data when you want to save, update, or delete.
- Handles object deletion (and deletion of any associated objects).
- Has change-tracking and undo support.
- Has sophisticated query mechanism.
- It also saves them to disk.

Core Data Things

- Managed object model
  - a collection of entity descriptions
- Persistent Store Coordinator
  - 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

NSManagedObjectContext

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

NSManagedObject

- State Information (was inserted, updated, deleted, etc)
- Lifecycle Hooks: methods called before/after important events (insertion, fetch, save)
- Validation
- 99% of the time you'll use subclasses, rather than bare NSManagedObjects

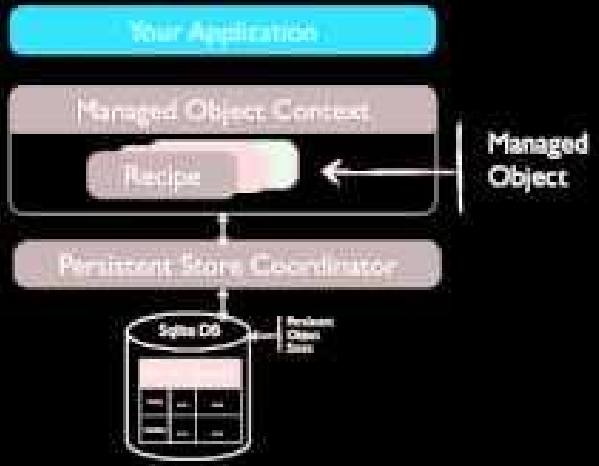
Creating Instances

We can use `NSEntityDescription` to create new instances of our `NSManagedObject`s.

```
LHLTask* task = [NSEntityDescription
insertNewObjectForEntityName:@"LHLTask"
inManagedObjectContext:context];
```

CoreData

Accessing the Persistent Store



Core Data CRUD

Food Journal App Model Layer

We’re going to see how to build the model layer for a simple personal data app using Core Data. The steps we’ll follow are:

- create the data model
- create some new instances
- use an NSFetchedResultsController to display our data in a table view.