Archiving
&
Core Data Relationships

My Core Data Video Part 1

http://bit.ly/2zshMXp

Download & Install Important Tools

- SimSim for opening files on the simulator http://bit.ly/2zqLsnu . (SimPholders paid).
- You need a SQLite viewer (can use command line). Here's a free one http://sqlitebrowser.org

Data Persistence Options on iOS

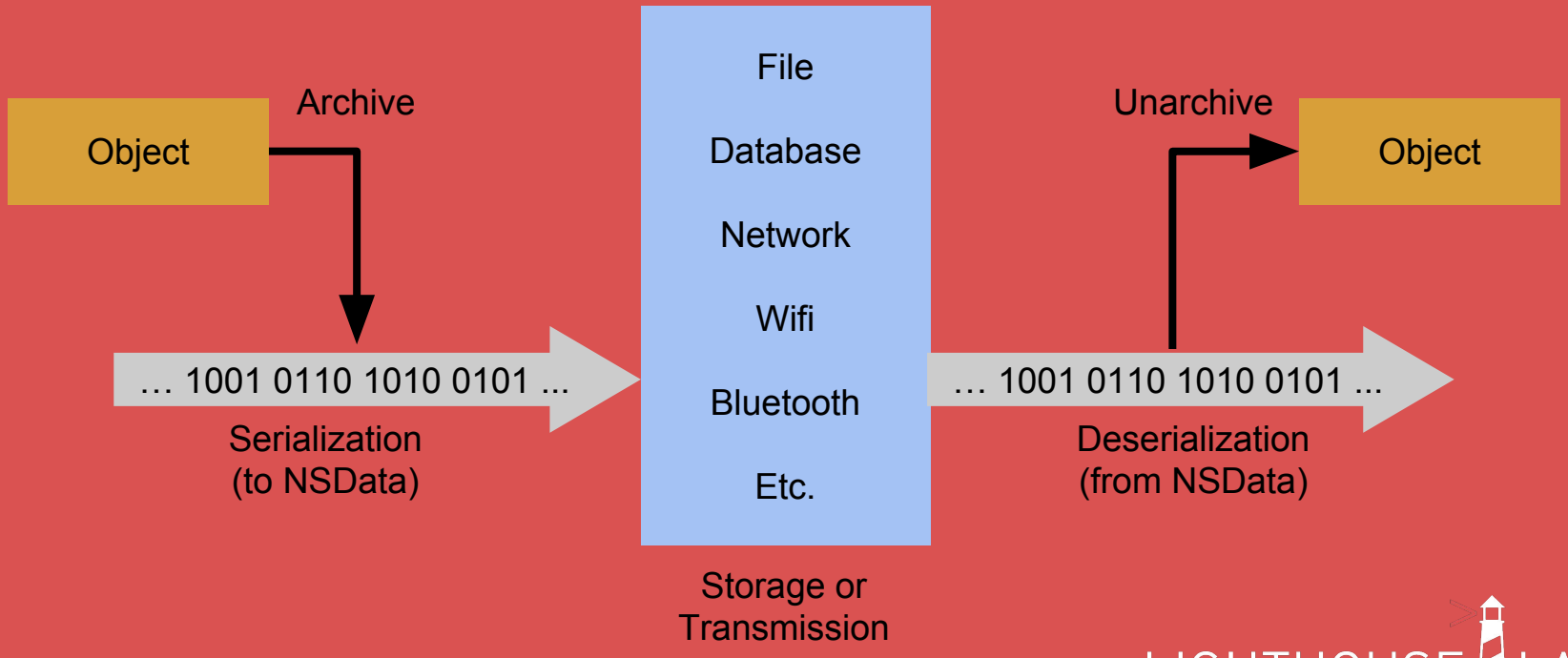
- Files
- NSUserDefaults
- Keychain
- plist files
- Direct SQL or SQL Wrapper
- Archiving NSCoding
- Core Data

LIGHTHOUSE LABS

Archiving/Unarchiving

- Store custom objects (including relationships) to disk and then instantiate them again later.
- aka serialization/deserialization.
- IB also uses this technique for storing your view hierarchies.
- Objects are responsible for knowing how to archive/unarchive themselves.
- They must conform to the `NSCoding` protocol.

Archiving/Unarchiving



LIGHTHOUSE LABS

Archiving/Unarchiving

```
@interface Person : NSObject <NSCoding>

@property (nonatomic, copy) NSString *firstName;
@property (nonatomic, copy) NSString *lastName;
@property (nonatomic) NSNumber *id;

@end
```

LIGHTHOUSE LABS

Archiving

```
@implementation Person

-(void)encodeWithCoder:(NSCoder *)aCoder {
    [coder encodeObject:self.firstName forKey:@"firstNameKey"];
    [coder encodeObject:self.lastName forKey:@"lastNameKey"];
    [coder encodeObject:self.id forKey:@"idKey"];
}

@end
```

Unarchiving

```
-(id)initWithCoder:(NSCoder *)aDecoder {
    self = [super init];
    if (self == nil) { return nil };
    _firstName = [aDecoder decodeObjectForKey:@"firstNameKey"];
    _lastName  = [aDecoder decodeObjectForKey:@"lastNameKey"];
    _id  = [aDecoder decodeObjectForKey:@"idKey"];
    return self;
}
```

Archiving/Unarchiving

- `NSCoder` is an "abstract" class.
- Apple also provides `NSKeyedArchiver` (encoding) and `NSKeyedUnarchiver` (decoding) concrete subclasses.
- Use `NSKeyedArchiver` to archive objects into `NSData` objects.
- Write NSData to file, plist, or transmit over a wire.
- Use `NSKeyedUnarchiver` to convert back to objects.

Archiving - 2 Places

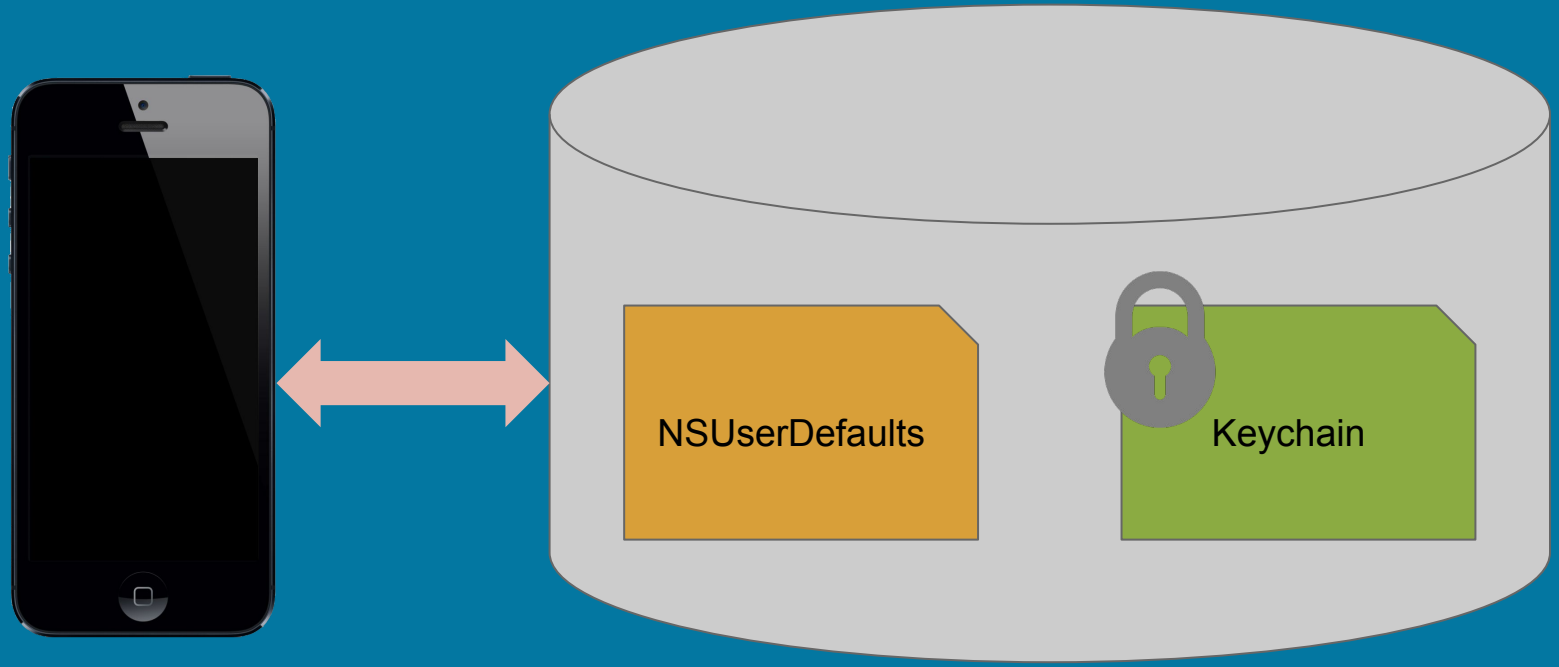
```
NSArray<Person *> *persons;   // An array of Person objects


// Saving to file
[NSKeyedArchiver archiveRootObject:persons
                            toFile:@"/path/to/archive"];



// NSUserDefaults
NSData *data = [NSKeyedArchiver archivedDataWithRootObject:persons];
[[NSUserDefaults standardUserDefaults] setObject:data
                                          forKey:@"persons"];
```

User Defaults & Keychain



LIGHTHOUSE LABS

User Defaults

- Not encrypted
- Simpler
- Survives application termination & reboot
- Does not survive application removal
- Stored in app sandbox

Keychain

- Encrypted
- More complex
- Survives application termination & reboot
- Survives application removal
- Only access app's keychain items

State Restoration

- UIKit provides functionality to help you preserve the UI of your app and restore it the next time the user enters your app.
- Use this to resume exactly where the user left off.

DEMO (Archiver)

Persistence Recap

- Saving data to files somewhere on disk doesn't scale well.
- Eg. How would you search/filter/query many archived items?
- Requires loading all data into memory and looping.

Persistence Recap

- The solution is using a relational database (SQLite is the only real choice on mobile).
- Permits complex data sets that can be quickly queried in advanced ways.
- But databases work with data, not objects.
- You'd have to write a lot of boilerplate code to ensure the integrity of your data relationships at the object layer.

Archiving/Unarchiving vs Core Data

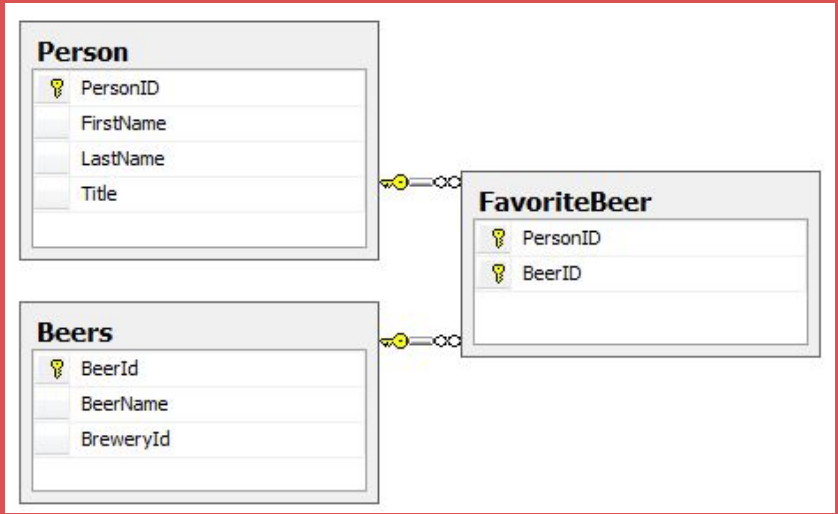
	Core Data	NSCoding / NSKeyedArchiver
Persists State	Yes	Yes
Entity Modelling	Yes	No
Querying	Yes	No
Speed	Fast	Slow
Serialization Format	SQLite, XML, or NSData	NSData
Migrations	Automatic*	Manual
Undo Manager	Automatic	Manual
Difficult Learning Curve	Maybe	No

Back to Core Data

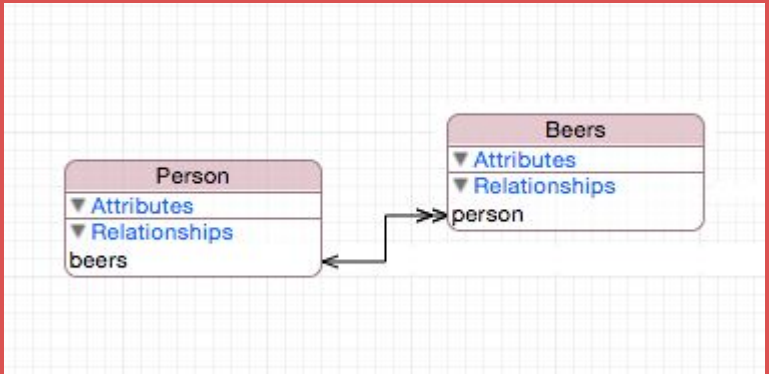
- Core Data is an “Object Graph” and object lifecycle management framework, not a database. It *can* persist to a database.
- What does this mean?

Core Data != Database

SQL Database Join Table



Core Data



LIGHTHOUSE LABS

Resources

- Apple's Serialization Guide: http://apple.co/2zoMzEn
- Core Data Programming Guide: http://apple.co/2zs2txN
- Core Data Relationships: http://apple.co/2mTIN0Z

LIGHTHOUSE LABS

DEMO