Announcements

- Lab 3 is due on Wednesday by 11:59 PM
- Lab 4 is due on Monday October 23rd

Lab 4

Today's Topics

- Property Lists
- iPhone's File System
- Archiving Objects
- SQLite
- Web Services

Storage on the iPhone

Property Lists

Property Lists

- **Convenient way to store a small amount of data**
  - Arrays, dictionaries, strings, numbers, dates, raw data
  - Human-readable XML or binary format
- **UserDefaults class uses property lists under the hood**

When Not to Use Property Lists

- **More than a few hundred KB of data**
  - Loading a property list is all-or-nothing
- **Complex object graphs**
- **Custom object types**

Property List Demo

iPhone's File System

Keeping Applications Separate

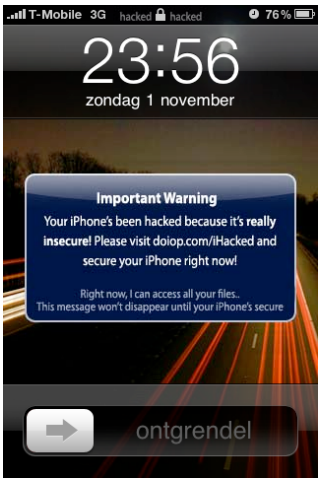
Why Keep Applications Separate?

- Security
- Privacy
- Cleanup after deleting an app

Home Directory Layout

- **Each app has its own set of directories**
- **<Application Home>**
  - MyApp.app
    - MyApp
    - MainViewController.storyboard
    - SomeImage.png
  - Documents
  - Library
    - Caches
    - Preferences
- **Applications only read and write within their home directory**

Demo



File Paths in Your Application

```
// Basic directories
let homeDir = NSHomeDirectory()
let tmpDir = NSTemporaryDirectory()

// Documents directory
let paths =
NSSearchPathForDirectoriesInDomains(. documentDirectory,
                                                    .userDomainMask, true)
let documentsPath = paths[0]

// <Application Home>/Documents/foo.plist
let fooPath = docsDir + "/foo.plist"
```


Including Writable Files with Your App

- **Many applications want to include some starter data**
- **But application bundles are code signed**
  - You can't modify the contents of your app bundle
- **To include a writable data file with your app...**
  - Build it as part of your app bundle
  - On first launch, copy it to your Documents directory

Archiving Objects

Archiving Objects

- **Next logical step from property lists**
  - Include arbitrary classes
  - Complex object graphs
- **Used by Storyboard for objects such as buttons and sliders, etc.**

Archiving Objects Example Class

```
class Book: NSObject, NSCoding {
  var title: String
  var author: String
  var pageCount: Int

   init(title: String, author: String, pageCount: Int) {
       self.title = title; self.author = author; self.pageCount = pageCount
   }

    required convenience init?(coder aDecoder: NSCoder)  {
      …
    }

    func encode(with aCoder: NSCoder) {
      …
    }
}
```


Archiving Objects Example Class

```
func encode (with aCoder: NSCoder) {

  aCoder.encode(title, forKey: "title")
  aCoder.encode(author, forKey: "author ")
  aCoder.encode(pageCount, forKey: "pageCount")

}
```

Archiving Objects Example Class

```
required convenience init?(coder aDecoder: NSCoder)  {
    guard let title = aDecoder.decodeObject(forKey: "title") as? String,
          let author = aDecoder.decodeObject (forKey: "author")  as? String
    else {
        return nil
    }
    self.init( title: title, author: author,
       pageCount: aDecoder.decodeInteger(forKey: "pageCount"))
}
```

init? Declares a "failable initialzer" which may return nil

guard statements transfer program control out of a scope if one or more conditions aren't met.

An init method marked as convenience must call the default init method

Archiving & Unarchiving Object Graphs

- **Creating an archive**

```
NSKeyedArchiver.archiveRootObject(book, toFile:"path/to/archive")
```

- **Decoding an archive**

```
guard let book = NSKeyedUnarchiver.unarchiveObject(withFile:
    "path/to/archive") as? Book else { return }
```

Archiving Objects Demo

SQLite

SQLite

- Complete SQL database in an ordinary file
- Simple, compact, fast, reliable
- No server
- Great for embedded devices
  - Included on the iPhone platform

When Not to Use SQLite

- Multi-gigabyte databases
- High concurrency (multiple writers)
- Client-server applications
- "Appropriate Uses for SQLite" http://www.sqlite.org/whentouse.html

More on SQLite

- "SQLite in 5 Minutes Or Less"
  - http://www.sqlite.org/quickstart.html
- "Intro to the SQLite C Interface"
  - http://www.sqlite.org/cintro.html
- FMDB SQLite Wrapper
  - https://github.com/ccgus/fmdb

SQLite Demo - FMDB

Core Data

- **Object-graph management and persistence framework**
  - Makes it easy to save and load model objects
    - Properties
    - Relationships
  - Higher-level abstraction than SQLite or property lists

Web Services

Your Application & The Cloud

- Store & access remote data
- May be under your control or someone else's
- Many Web 2.0 apps/sites provide developer API

Integrating with Web Services

- Non-goal of this class: teach you all about web services
  - Plenty of tutorials accessible, search on Google
- Many are exposed with XML or JSON
- High level overview of parsing these types of data

XML

Options for Parsing XML

- **XMLParser**
  - Event-driven API
  - https://developer.apple.com/reference/foundation/xmlparser
- **SWXMLHash**
  - Simple Swift XML Parsing
    - https://github.com/drmohundro/SWXMLHash

JSON

JavaScript Object Notation

- More lightweight than XML
- Looks a lot like a property list
  - Arrays, dictionaries, strings, numbers
- Open source json-framework wrappers for Swift and Objective-C

What does a JSON string look like?

```
{
   “instructor” : “Todd Sproull”,
   “students” : 20,
   “itunes-u” : true,
   “midterm-exam” : null,
   “assignments” : [ “WhatATool”,
                     “HelloPoly”]
}
```


More on JSON

- “Introducing JSON”
  - http://www.json.org/
- SwiftyJSON
  - https://github.com/SwiftyJSON/SwiftyJSON
- JSON Editor
  - http://www.jsoneditoronline.org/

Recap

- **Property lists**
  - Quick & easy, but limited
- **Archived objects**
  - More flexible, but require writing a lot of code
- **SQLite and Core Data**
  - Elegant solution for many types of problems
- **XML and JSON**
  - Low-overhead options for talking to "the cloud"