Announcements

- **Lab 4 is due tonight at 11:59 PM**
- **Lab 5 is now posted**
  - Due March 30th
  - Do not procrastinate on this lab
- **Oscar's Sunday TA hours are cancelled**
  - Dan is still available on Sunday from 3 – 4 PM
- **Discuss final project ideas on Monday March 16th**
  - We have several guest presenters to pitch ideas
  - You may also discuss your idea and look for partners
  - Final project groups should consist of 3 or 4 people
  - No individual projects

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
- **NSUserDefaults 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

Reading & Writing Property Lists

- NSArray and NSDictionary convenience methods
- Operate recursively

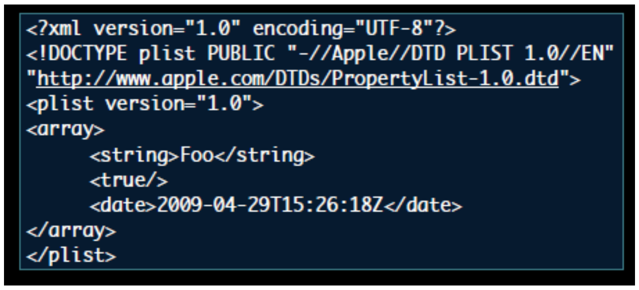
```
// Writing
- (BOOL)writeToFile:(NSString *)aPath atomically:(BOOL)flag;
- (BOOL)writeToURL:(NSURL *)aURL atomically:(BOOL)flag;

// Reading
- (id)initWithContentsOfFile:(NSString *)aPath;
- (id)initWithContentsOfURL:(NSURL *)aURL;
```

Writing an Array to Disk

```
NSArray *array = [NSArray arrayWithObjects:@“Foo”,
[NSNumber numberWithBool:YES],
[NSDate dateWithTimeIntervalSinceNow:60],nil];

[array writeToFile:@“MyArray.plist” atomically:YES];
```

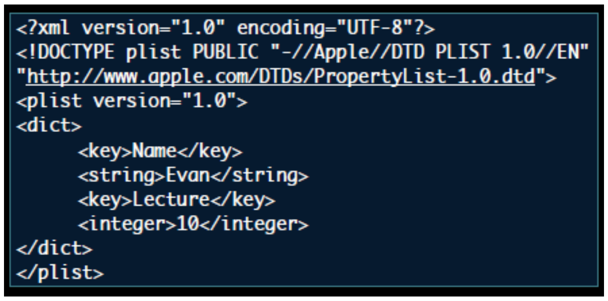


```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<array>
      <string>Foo</string>
      <true/>
      <date>2009-04-29T15:26:18Z</date>
</array>
</plist>
```

Writing a Dictionary to Disk

```
NSDictionary *dict = [NSDictionary dictionaryWithObjectsAndKeys:
@“Name”, @“Evan”,
@“Lecture”, [NSNumber numberWithInt:10], nil];

[dict writeToFile:@“MyDict.plist” atomically:YES];
```



```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
      <key>Name</key>
      <string>Evan</string>
      <key>Lecture</key>
      <integer>10</integer>
</dict>
</plist>
```

NSPropertyListSerialization

- **Allows finer-grained control**
  - File format
  - More descriptive errors
  - Mutability

```
    // Property list to NSData
+ (NSData *)dataFromPropertyList:(id)plist
                                 format:(NSPropertyListFormat)format
                         errorDescription:(NSString **)errorString;

// NSData to property list
+ (id)propertyListFromData:(NSData *)data
          mutabilityOption:(NSPropertyListMutabilityOptions)opt
                    format:(NSPropertyListFormat *)format
          errorDescription:(NSString **)errorString;
```

More on Property Lists

"Property List Programming Guide for Cocoa"

http://developer.apple.com/documentation/Cocoa/Conceptual/PropertyLists/

iPhone's File System

Keeping Applications Separate

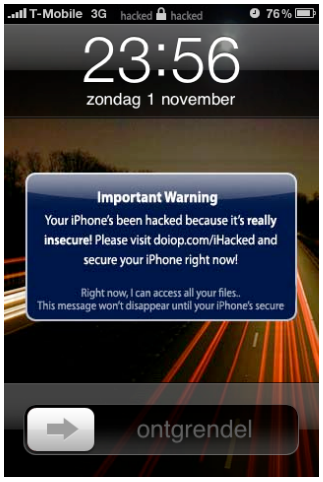
Why Keep Applications Separate?

- Security
- Privacy
- Cleanup after deleting an app

Home Directory Layout

- Each app has its own set of directories
- <Application Home>
  - MyApp.app
    - MyApp
    - MainWindow.nib
    - SomeImage.png
  - Documents
  - Library
    - Caches
    - Preferences
- Applications only read and write within their home directory
- Backed up by iTunes during sync (mostly)

Demo



Important Warning

Your iPhone's been hacked because it's really insecure! Please visit doiop.com/iHacked and secure your iPhone right now!

Right now, I can access all your files..
This message won't disappear until your iPhone's secure

File Paths in Your Application

```
// Basic directories
NSString *homePath = NSHomeDirectory();
NSString *tmpPath = NSTemporaryDirectory();

// Documents directory
NSArray *paths =
NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,
                                    NSUserDomainMask, YES);
NSString *documentsPath = [paths objectAtIndex:0];

// <Application Home>/Documents/foo.plist
NSString *fooPath = [documentsPath
    stringByAppendingPathComponent:@“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 Interface Builder for NIBs**

Making Objects Archivable

- **Conform to the <NSCoding> protocol**

```
// Encode an object for an archive
-   (void)encodeWithCoder:(NSCoder *)coder
{
  [super encodeWithCoder:coder];
  [coder encodeObject:name forKey:@“Name”];
  [coder encodeInteger:numberOfSides forKey:@“Sides”];
 }

// Decode an object from an archive
-   (id)initWithCoder:(NSCoder *)coder
{
  self = [super initWithCoder:coder];
  name = [[coder decodeObjectForKey:@“Name”] retain];
  numberOfSides = [coder decodeIntegerForKey:@“Side”];
}
```

Archiving & Unarchiving Object Graphs

- Creating an archive

```
NSArray *polygons = ...;
NSString *path = ...;
BOOL result = [NSKeyedArchiver archiveRootObject:polygons
            toFile:path];
```

- Decoding an archive

```
NSArray *polygons = nil;
NSString *path = ...;
polygons = [NSKeyedUnarchiver unarchiveObjectWithFile:path];
```

More on Archiving Objects

"Archives and Serializations Programming Guide for Cocoa"

http://developer.apple.com/documentation/Cocoa/Conceptual/Archiving/

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
- Example code available
  – http://research.engineering.wustl.edu/~todd/cse436/examples/MySQLiteTableView.zip

SQLite Demo

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
- **Available on iPhone 3.0 and up**
  - http://developer.apple.com/iphone/library/documentation/DataManagement/Conceptual/iPhoneCoreData01/Introduction/Introduction.html

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

- **libxml2**
  - Tree-based: easy to parse, entire tree in memory
  - Event-driven: less memory, more complex to manage state
  - Text reader: fast, easy to write, efficient
- **NSXMLParser**
  - Event-driven API: simpler but less powerful than libxml2

More on Parsing XML

- **Brent Simmons, "libxml2 + xmlTextReader on Macs"**
  http://inessential.com/?comments=1&postid=3489
  - Includes example of parsing Twitter XML!
- **Big Nerd Ranch, "Parsing XML in Cocoa"**
  http://weblog.bignerdranch.com/?p=48
  - Covers the basics of NSXMLReader

JSON

JavaScript Object Notation

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

What does a JSON string look like?

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

Using json-framework

- Reading a JSON string into Foundation objects

```
#import <JSON/JSON.h>

// Get a JSON string from the cloud
NSString *jsonString = ...;

// Parsing will result in Foundation objects
// Top level may be an NSDictionary or an NSArray
id object = [jsonString JSONValue];
```

Using json-framework

- Writing a JSON string from Foundation objects

```
// Create some data in your app
  NSDictionary *dictionary = ...;

// Convert into a JSON string before sending to the cloud
  jsonString = [dictionary JSONRepresentation];
```

JSON Demo

More on JSON

- **"JSON Parser/Generator for Objective-C"**
  - http://code.google.com/p/json-framework/
- **"Introducing JSON"**
  - http://www.json.org/
- **Example code available**
  - http://research.engineering.wustl.edu/~todd/cse436/examples/

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"

Parse Demo