Lecture #3 - NSURL

NSURL

NSURL

- URL = “Universal Resource Locator”
  - `http://....`
  - `file://...`

NSURL

- URL = “Universal Resource Locator”
  - `http://....`
  - `file://...`
- Provides means to create / manipulate URLs

NSURL

- URL = “Universal Resource Locator”
  - `http://....`
  - `file://...`
- Provides means to create / manipulate URLs
- Composed of a base URL, and string

NSURL

- URL = “Universal Resource Locator”
  - `http://....`
  - `file://...`
- Provides means to create / manipulate URLs
- Composed of a base URL, and string
- Historically, most Foundation APIs used `NSString` paths
  - Today, most use `NSURL` as well
  - Use `NSURL` when possible – most Cocoa APIs are moving to `NSURL`

NSURL

- Useful Methods

NSURL

- Useful Methods

```
// Create a URL representing a file system path
+ (NSURL *)fileURLWithPath:(NSString *)path
                isDirectory:(BOOL)isDirectory;
```

NSURL

- Useful Methods

```
// Create a URL representing a file system path
+ (NSURL *)fileURLWithPath:(NSString *)path
                isDirectory:(BOOL)isDirectory;
```

```
// Create a URL for any scheme, e.g. “https”, “http”, ...
+ (NSURL *)urlWithScheme:(NSString *)scheme
                    host:(NSString *)host
                    path:(NSString *)path;
```

NSURL

- Useful Methods

```
// Create a URL representing a file system path
+ (NSURL *)fileURLWithPath:(NSString *)path
                isDirectory:(BOOL)isDirectory;
```

```
// Create a URL for any scheme, e.g. “https”, “http”, ...
+ (NSURL *)urlWithScheme:(NSString *)scheme
                    host:(NSString *)host
                    path:(NSString *)path;
```

```
// Get the path
- (NSString *)path;

- (BOOL)isFileURL; // Convenience to see if a file url
- (NSString *)scheme; // Get the scheme - http, file, ftp, ...
```

NSURL

- Used By Other APIs – in place of `NSString *` paths

NSURL

- Used By Other APIs – in place of `NSString *` paths

```
// NSString.h

+ (NSString *)stringWithContentsOfURL:(NSURL *)url;
- (void)writeToURL:(NSURL *)url ...;
```

NSURL

- Used By Other APIs – in place of `NSString *` paths

```
// NSString.h

+ (NSString *)stringWithContentsOfURL:(NSURL *)url;
- (void)writeToURL:(NSURL *)url ...;
```

```
// NSData.h

+ (NSString *)dataWithContentsOfURL:(NSURL *)url;
```

NSURL

```
NSURL *filePathURL = [NSURL fileURLWithPath:@"/tmp/foo.png"];
```

```
NSURL *httpURL = [NSURL URLWithString:@"http://www.apple.com/iphone"];
```

```
NSURL *httpURL = [[NSURL alloc] initWithScheme:@"http"
                                          host:@"www.apple.com"
                                          path:@"/iphone"];
```

Lecture #4 - Property Lists

Property Lists

- Example

```
<?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>Fred</key>
    <dict>
        <key>Assignment1</key>
        <integer>95</integer>
        <key>IsGraduateStudent</key>
        <true/>
    </dict>
</dict>
</plist>
```

Property Lists

- Example

```
<?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>Fred</key>
    <dict>
        <key>Assignment1</key>
        <integer>95</integer>
        <key>IsGraduateStudent</key>
        <true/>
    </dict>
</dict>
</plist>
```

Property Lists

- Example

```
<?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>Fred</key>
    <dict>
        <key>Assignment1</key>
        <integer>95</integer>
        <key>IsGraduateStudent</key>
        <true/>
    </dict>
</dict>
</plist>
```

Property Lists

- Example

```
<?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>Fred</key>
    <dict>
        <key>Assignment1</key>
        <integer>95</integer>
        <key>IsGraduateStudent</key>
        <true/>
    </dict>
</dict>
</plist>
```

Property Lists

- Example

```
<?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>Fred</key>
    <dict>
        <key>Assignment1</key>
        <integer>95</integer>
        <key>IsGraduateStudent</key>
        <true/>
    </dict>
</dict>
</plist>
```

Property Lists

- Example

```
<?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>Fred</key>
    <dict>
        <key>Assignment1</key>
        <integer>95</integer>
        <key>IsGraduateStudent</key>
        <true/>
    </dict>
</dict>
</plist>
```

Persistence

- `NSArray`, `NSDictionary`

```
// Reading
+ (id)arrayWithContentsOfURL:(NSURL *)aURL;
+ (id)dictionaryWithContentsOfURL:(NSURL *)aURL;

// Writing
- (BOOL)writeToURL:(NSURL *)aURL atomically:(BOOL)flag
```

- `NSPropertyListSerialization` Class
  - Fine control
  - Convert between plist serialization format

An Example

- Create and save a property list

```
NSDictionary *fredsInfo = [NSDictionary dictionaryWithObjectsAndKeys:
                                [NSNumber numberWithInt:95], @"Assignment1",
                                [NSNumber numberWithBool:NO], @"IsGradStudent",
                                nil];

NSDictionary *classInfo = [NSDictionary dictionaryWithObjectsAndKeys:
                                fredsInfo, @"Fred",
                                nil];

[classInfo writeToFile:@"/tmp/classInfo.plist" atomically:YES];
```

An Example

- Create and save a property list

```
NSDictionary *fredsInfo = [NSDictionary dictionaryWithObjectsAndKeys:
                                [NSNumber numberWithInt:95], @"Assignment1",
                                [NSNumber numberWithBool:NO], @"IsGradStudent",
                                nil];

NSDictionary *classInfo = [NSDictionary dictionaryWithObjectsAndKeys:
                                fredsInfo, @"Fred",
                                nil];

[classInfo writeToFile:@"/tmp/classInfo.plist" atomically:YES];
```

- Loading a property list

```
NSDictionary *classInfo =
          [NSDictionary dictionaryWithContentsOfFile:@"/tmp/classInfo.plist"];
```

Summary

- Foundation
  - Value Classes – `NSString, NSNumber, NSURL`, ...
  - Collection Classes – `NSArray, NSDictionary`, ...
  - Property Lists
- CoreFoundation – `CFTypes`
  - Toll Free Bridging
- Our First Design Pattern
  - Mutable / Immutable Pattern

Reading

- CocoaFundamentals.pdf
  - The Root Class: P.70 - 74
  - Foundation Classes: P.34 - 40
  - Object Mutability: P.94 - 98

Reading

- CocoaFundamentals.pdf
  - A Bit of History: P.57, 58
- “Collections Programming”
  - http://developer.apple.com/mac/library/documentation/Cocoa/Conceptual/Collections/Collections.pdf
- Online Class References
  - http://developer.apple.com/iphone/library/documentation/Cocoa/Reference/Foundation/ObjC_classic/index.html
  - Focus on NSArray, NSDictionary

Dates

NSDate

- Represents a single point in time
- Point in time communicated using `NSTimeInterval`

```
// Time interval in seconds
typedef double NSTimeInterval;
```

- Question becomes, what is the `NSTimeInterval` relative to?

NSDate / NSTimeInterval

- Absolute time – relative to the **“Reference Date”**
  - “Reference Date” – 1 Jan 2001 00:00:00 GMT
  - `CFAbsoluteTime`, or `NSTimeInterval` when API has “referenceDate” in it

```
// NSDate.h
+ (id)dateWithTimeIntervalSinceReferenceDate:(NSTimeInterval)secs;
+ (NSTimeInterval)timeIntervalSinceReferenceDate;
```

- Other frames of reference
  - `NSTimeInterval`

```
// NSDate.h
+ (id)dateWithTimeIntervalSinceNow:(NSTimeInterval)secs;
- (NSTimeInterval)timeIntervalSinceNow;
```

```
// NSDate.h
+ (id)dateWithTimeIntervalSince1970:(NSTimeInterval)secs;
- (NSTimeInterval)timeIntervalSince1970;
```

Calendar Dates

- When you want dates relative to a calendaring system
  - NSCalendar
  - NSDateComponents
- Example

```
// Calculate an NSDate for -- 2nd Friday in May, 2008 (start of day)
```

```
NSDateComponents *components = [[NSDateComponents alloc] init];
[components setWeekday:6];          // Friday
[components setWeekdayOrdinal:2];  // The 2nd week of the month
[components setMonth:5];            // May
[components setYear:2008];          // 2008
```

```
NSCalendar *cal = [[NSCalendar alloc] initWithCalendarIdentifier:NSGregorianCalendar];

NSDate *date = [gregorian dateFromComponents:components];
```

Date & Times

- Date and Time Programming Guide
- Foundation
  - http://developer.apple.com/mac/library/documentation/Cocoa/Conceptual/DatesAndTimes/DatesAndTimes.pdf
  - Read The Following Sections
    - “Dates” – p. 9-11
    - “Calendars, Date Components, and Calendar Units” – p.11 - 14
    - “Calendrical Calculations” – p.15 - 18
    - “Time Zones” - p. 19