Core Networking

Leo Font

Networking Libraries options

- Foundation Layer
- Core Foundation Layer

Foundation Layer

- Protocol streams: NSStream
- URL Loading: NSURLConnection / NSURLSession and NSURLRequest
- Service discovery: NSNetService

Core Foundation Layer

- Protocol streams: CFStream
- URL Loading: CFHTTPMessage
- Service discovery: CFNetService

Which one?

- **Always the most high level one**: Foundation Layer
- Exceptions:
  - Writing server code
  - Special needs

Peer to peer networking (games)

- Game Kit framework
- Support for peer-to-peer communication
  - Globally (over the Internet)
  - Locally (using a Bluetooth personal area network or a Wi-Fi LAN)

Peer-to-peer networking (for other apps)

- Multipeer Connectivity framework
  - Support for peer-to-peer communication
    - Infrastructure Wi-Fi
    - Peer-to-peer Wi-Fi
    - Bluetooth.

Connect to a web server

The preferred way to send and receive short pieces of information is over a standard protocol such as HTTP or HTTPS

NSURLSession vs NSURLConnection

NSURLConnection: OLD Foundation URL Loading System

- NSURLRequest
- NSURLResponse
- NSURLProtocol
- NSURLCache
- NSHTTPCookieStorage
- NSURLCredentialStorage

NSURLSession

- Apple preferred way to consume REST since iOS7 /OSX 10.9
- Improved architecture replaces NSURLConnection.
- Rich closures/delegate model.
- Can configure per-session cache, protocol, cookie, and credential policies, rather than sharing them across the app

Code

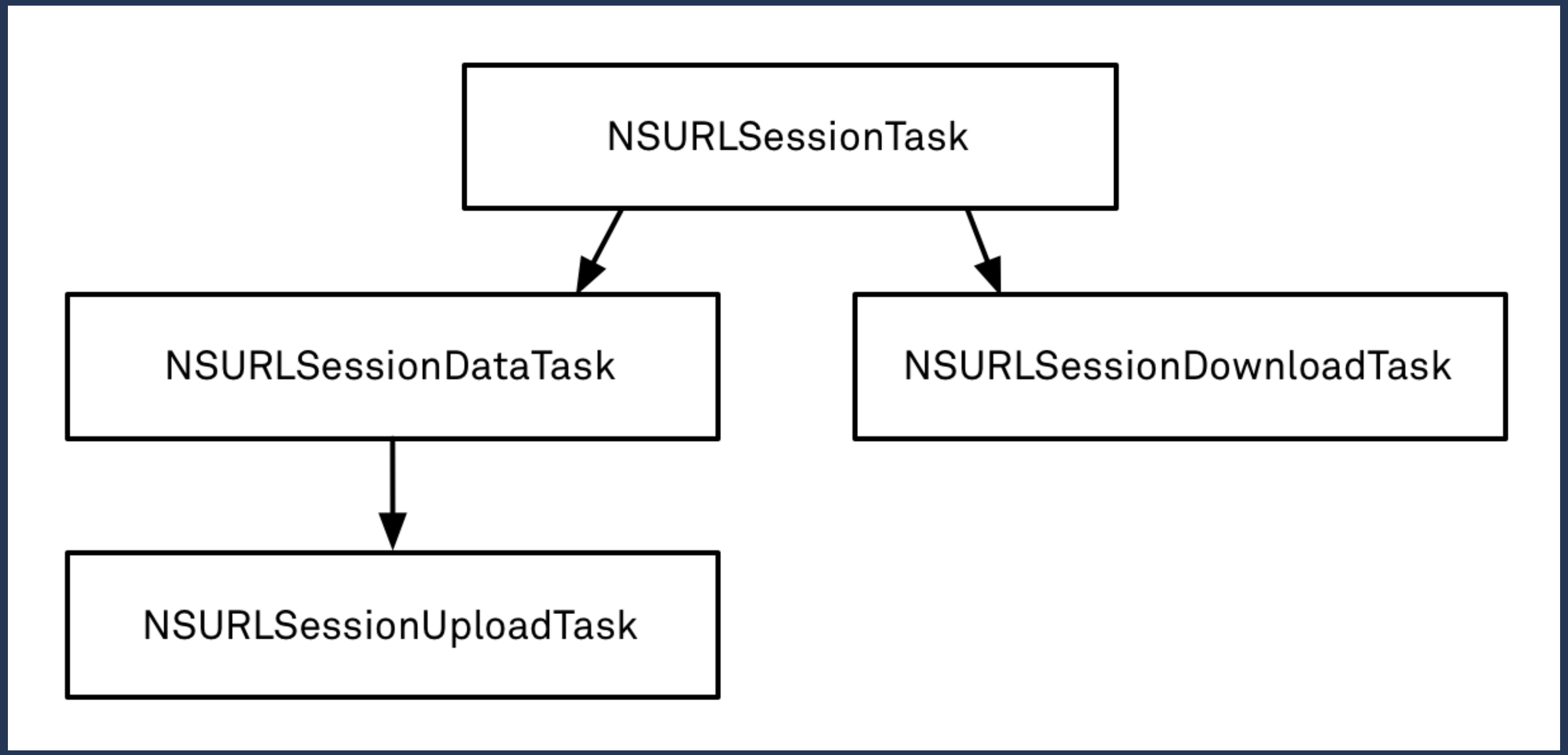
```
NSString *weatherURL = @"http://api.openweathermap.org/data/2.5/find?q=Madrid&units=metric";

NSURLSession *session = [NSURLSession sharedSession];
[[session dataTaskWithURL:[NSURL URLWithString:weatherURL]
        completionHandler:^(NSData *data, NSURLResponse *response, NSError *error) {

            NSLog(@"received %@",data);

        }] resume];
```

An NSURLSession is made using an NSURLSessionConfiguration with an optional delegate. After you create the session you then satisfy you networking needs by creating NSURLSessionTask's.

NSURLSession vs AFNetworking

- Is NSURLSession better than AFNetworking?
- They are different! One comes with your system (Apple). The other is useful in terms of object serialisation and some convenience utilities but generates an external dependency.

Anatomy of NSURLSessionTask



NSURLSessionConfiguration

- DefaultSessionConfiguration – creates a configuration object that uses the global cache, cookie and credential storage objects.
- EphemeralSessionConfiguration – this configuration is for “private” sessions and has no persistent storage for cache, cookie, or credential storage objects.
- BackgroundSessionConfiguration – use it to make networking calls from remote push notifications or while app suspended.

Once you create a NSURLSessionConfiguration, you can set various properties on it like this:

```
NSURLSessionConfiguration *sessionConfig =

[NSURLSessionConfiguration defaultSessionConfiguration];

sessionConfig.allowsCellularAccess = NO;

[sessionConfig setHTTPAdditionalHeaders:@{@“Accept": @"application/json"}];
```

```
sessionConfig.timeoutIntervalForRequest = 30.0;

sessionConfig.timeoutIntervalForResource = 60.0;

sessionConfig.HTTPMaximumConnectionsPerHost = 1;
```

NSUrlSessionDataTask without delegate (using completion handler)

```
NSURL *URL = [NSURL URLWithString:@"http://example.com"];
NSURLRequest *request = [NSURLRequest requestWithURL:URL];

NSURLSession *session = [NSURLSession sharedSession];
NSURLSessionDataTask *task = [session dataTaskWithRequest:request
                                        completionHandler:
^(NSData *data, NSURLResponse *response, NSError *error) {
    // ...
}];

[task resume];
```


KEEP
CALM
IT'S
YOUR
TURN NOW

NSURLSessionDataTask With Delegate

```
NSURLSessionConfiguration *sessionConf = [NSURLSessionConfiguration defaultSessionConfiguration];

NSURLSession *session = [NSURLSession sessionWithConfiguration: sessionConf delegate: self delegateQueue: nil];

NSURL * url = [NSURL URLWithString:@"http://example.com”];

NSURLSessionDataTask *dataTask = [session dataTaskWithURL:url];

[dataTask resume];
```

Some of the delegates available:

```
-(void)URLSession:(NSURLSession *)session dataTask:(NSURLSessionDataTask *)dataTask didReceiveData:(NSData *)data
```

```
-(void)URLSession:(NSURLSession *)session task:(NSURLSessionTask *)task didCompleteWithError:(NSError *)error
```

- Others available to handle redirections,authentication…


KEEP
CALM
IT'S
YOUR
TURN NOW

NSURLSessionUploadTask

```
NSURL *URL = [NSURL URLWithString:@"http://example.com/upload"];
NSURLRequest *request = [NSURLRequest requestWithURL:URL];
NSData *data = ...;

NSURLSession *session = [NSURLSession sharedSession];
NSURLSessionUploadTask *uploadTask = [session uploadTaskWithRequest:request
                                                           fromData:data
                                                  completionHandler:
    ^(NSData *data, NSURLResponse *response, NSError *error) {
        // ...
    }];

[uploadTask resume];
```

NSURLSessionDownloadTask

```
NSURL *URL = [NSURL URLWithString:@"http://example.com/file.zip"];

NSURLRequest *request = [NSURLRequest requestWithURL:URL];


NSURLSession *session = [NSURLSession sharedSession];

NSURLSessionDownloadTask *downloadTask = [session downloadTaskWithRequest:request
                                                         completionHandler:
   ^(NSURL *location, NSURLResponse *response, NSError *error) {
       NSString *documentsPath = [NSSearchPathForDirectoriesInDomains(NSDocumentDirectory, NSUserDomainMask, YES) firstObject];
       NSURL *documentsDirectoryURL = [NSURL fileURLWithPath:documentsPath];
       return [documentsDirectoryURL URLByAppendingPathComponent:[[response URL] lastPathComponent]];
   }];

[downloadTask resume];
```

Download image

```
NSURLSessionDownloadTask *getImageTask =
[session downloadTaskWithURL:[NSURL URLWithString:imageUrl]

    completionHandler:^(NSURL *location,
                        NSURLResponse *response,
                        NSError *error) {
                UIImage *downloadedImage =
            [UIImage imageWithData:
                [NSData dataWithContentsOfURL:location]];

      dispatch_async(dispatch_get_main_queue(), ^{
        // do stuff with image
        _imageWithBlock.image = downloadedImage;
      });
}];


[getImageTask resume];
```

Delegates also available

-(void)URLSession:(NSURLSession *)session downloadTask:(NSURLSessionDownloadTask *)downloadTask didFinishDownloadingToURL:(NSURL *)location

-(void)URLSession:(NSURLSession *)session downloadTask:(NSURLSessionDownloadTask *)downloadTask didWriteData:(int64_t)bytesWritten totalBytesWritten:(int64t)totalBytesWritten totalBytesExpectedToWrite:(int64t)totalBytesExpectedToWrite


KEEP
CALM
IT'S
YOUR
TURN NOW

Post requests

```
NSURLSessionConfiguration *defaultConfigObject = [NSURLSessionConfiguration defaultSessionConfiguration];

NSURLSession *defaultSession = [NSURLSession sessionWithConfiguration: defaultConfigObject delegate: self delegateQueue: nil];

NSURL * url = [NSURL URLWithString:@"http://posttestserver.com/post.php"];

NSMutableURLRequest * urlRequest = [NSMutableURLRequest requestWithURL:url];

NSString * params =@"&name=Leo&loc=Madrid&age=42";

[urlRequest setHTTPMethod:@"POST"];

[urlRequest setHTTPBody:[params dataUsingEncoding:NSUTF8StringEncoding]];

NSURLSessionDataTask * dataTask = [defaultSession dataTaskWithRequest:urlRequest];

[dataTask resume];
```

Delegates

-(void)connection:(NSURLConnection *)connection didReceiveData:(NSData *)data

-(void)connection:(NSURLConnection *)connection didFailWithError:(NSError *)error

-(void)connectionDidFinishLoading:(NSURLConnection *)connection


KEEP
CALM
IT'S
YOUR
TURN NOW

Caching Policies

- URLCache is the cache used by the session. By default, NSURLCache +sharedURLCache is used, which is the same as NSURLConnection.
- requestCachePolicy specifies when a cached response should be returned for a request. This is equivalent to NSURLRequest -cachePolicy.

Caching Policies

- If our server uses Cache-Control HTTP headers and sets the maximum age for its responses, both the shared NSURLSession and NSURLConnection will respect this and return cached responses before they expire.
- By default, NSURLSession and NSURLConnection will not use an expired response if there isn’t internet connectivity

Caching Policies

```
NSURLRequest *request;
if (reachability.isReachable)
{
        request = [NSURLRequest requestWithURL:url];
}
else
{
     request = [NSURLRequest requestWithURL:url
                                cachePolicy:NSURLRequestReturnCacheDataElseLoad
                                timeoutInterval:60];
}
```

- If this is not enough: Last choice -> Force response caching locally…

Parsing JSON Data

- JSON = JavaScript Object Notation.
- Represents structured data in a human-readable format.
- Works with any programming language.
- Easier to read & parse than XML.

NSJSONSerialization

- Available since iOS5.
- Converts a JSON file to a NSDictionary/NSArray of dictionaries.
- Also allows to convert NSDictionary/NSArray to a JSON format.

NSJSONSerialization

```
NSError *error;

NSDictionary *json = [NSJSONSerialization JSONObjectWithData:data options:kNilOptions error:&error];

NSDictionary *results = [[json valueForKeyPath:@“key”]];
```

- If there’s an error, nil is returned.


KEEP
CALM
IT'S
YOUR
TURN NOW

Evening Challenge

Iron Weather App (maybe next Apple's bundled weather App)

Requirements:

- Main screen should show current weather in your actual city including a pictogram representing weather condition (in this case will be Madrid - hardcoded as we already don't know Core Location yet)
- Current weather forecast should be updated every 10 minutes.

- Secondary screen zone should show weather forecast for your city’s next 5 days, including a pictogram representing weather condition.
- Extra points: During week add actual real location using reverse geocode.
- Extra points: During week localize weather description depending on device language.
- Very important: Don’t lock user interface while fetching data from the API!!!