Last week

Smarticle

Smarticle

- Networking
- Persistence

Networking

- Real-world networks
- Networking is hard
- iOS networking stacks

Types of networks

- Wide Area Networks
- Local Area Networks
- Personal Area Networks

Types of networks

- Internet
- Bonjour
- Bluetooth/Multipeer

Internet

- Layered architecture
- Client-server
- Unreliable, best-effort

Layers

- Application
- Network
- Transport
- Physical

Layers

- HTTP/HTTPS
- IP
- TCP
- Wifi/Cellular

Client-Server

- REST
- JSON

Failures

- Packet loss
- Latency
- Firewalls
- Captive networks
- Server errors

Time and Money

- Waiting for a response
- Battery life
- Bandwidth
- Data transfer cost

Networking on iOS

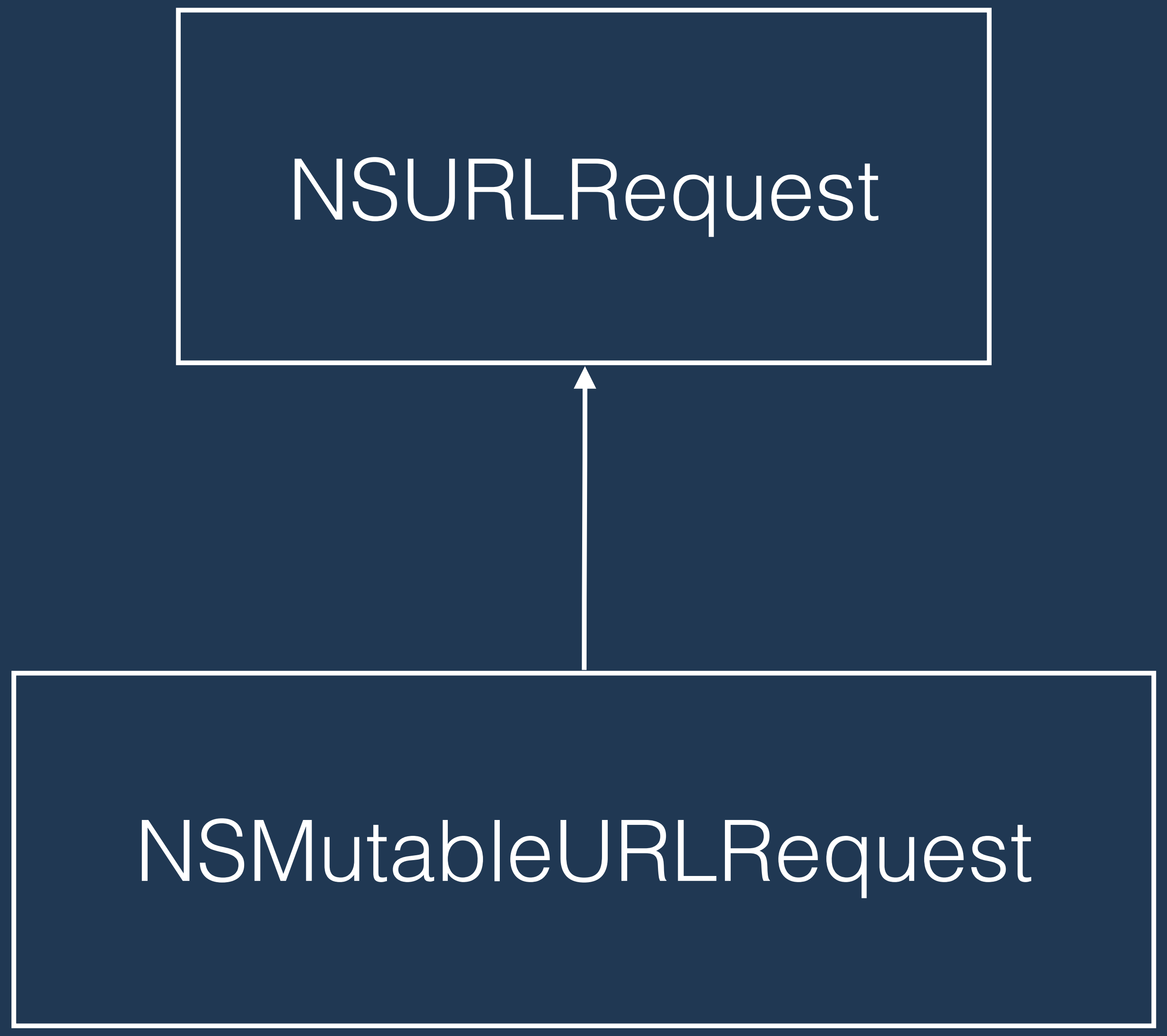
- NSURL
- NSURLRequest
- NSURLSession

NSURL

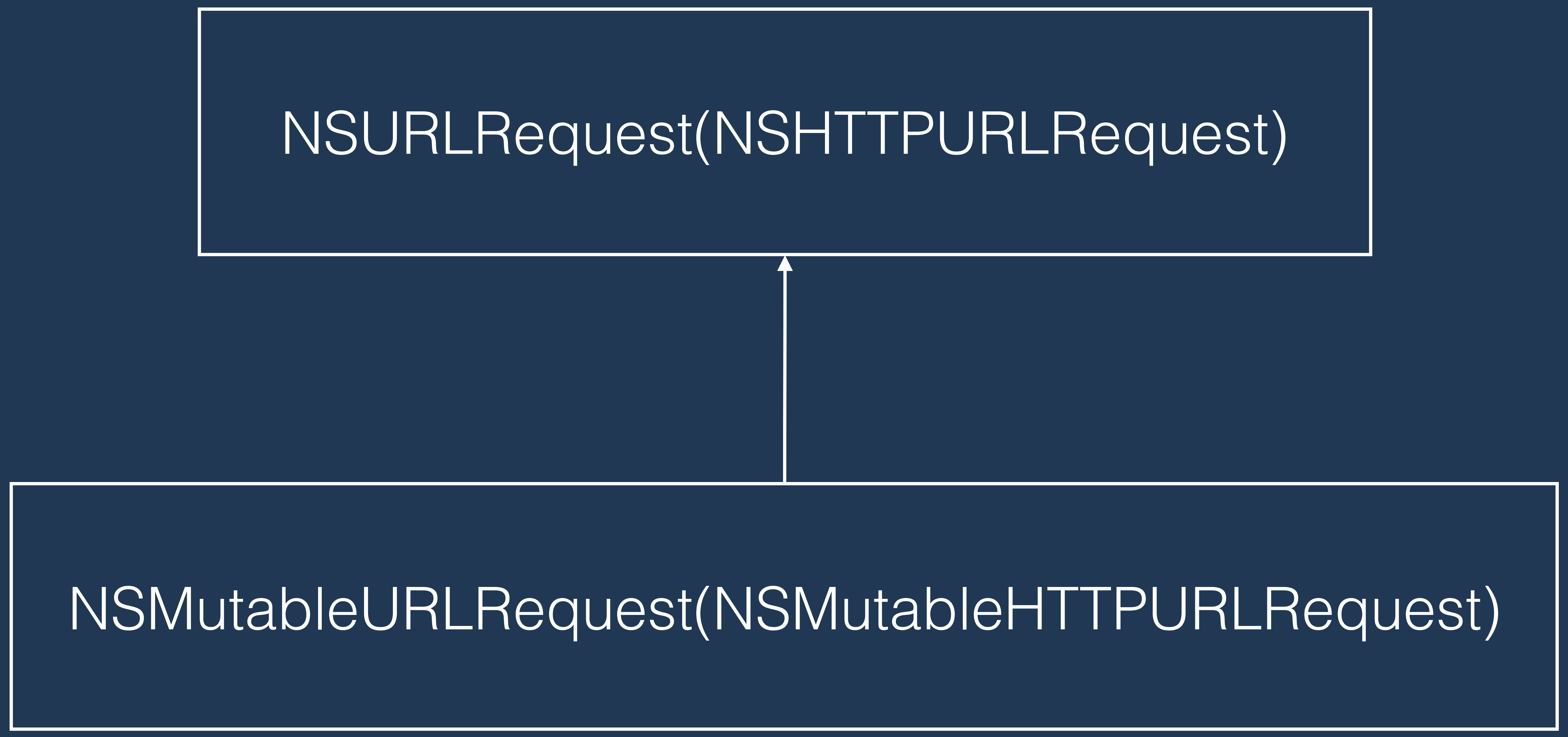
```
NSString *requestString = @"http://www.apple.com";
NSURL *url = [NSURL URLWithString:requestString];

NSString *contents = [NSString stringWithContentsOfURL:url
                                              encoding:NSUTF8StringEncoding
                                                 error:nil];

NSLog(@"%@", contents);
```

Ship it?

NSURLRequest



- Timeout interval
- Cache policy
- Allows cellular access


NSURLRequest(NSHTTPURLRequest)
NSMutableURLRequest(NSMutableHTTPURLRequest)

- Method
- Header fields
- Body
- Cookies
- Pipelining

NSURLConnection

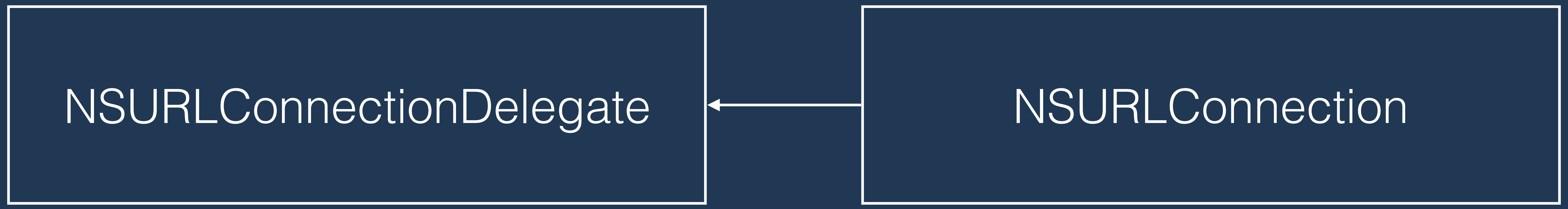
```objc
NSMutableURLRequest *request = [NSMutableURLRequest requestWithURL:url];
request.timeoutInterval = 60;

NSData *data = [NSURLConnection sendSynchronousRequest:request
                                     returningResponse:nil
                                                 error:nil];

NSString *contents = [[NSString alloc] initWithData:data
                                           encoding:NSUTF8StringEncoding];
NSLog(@"%@", contents);
```

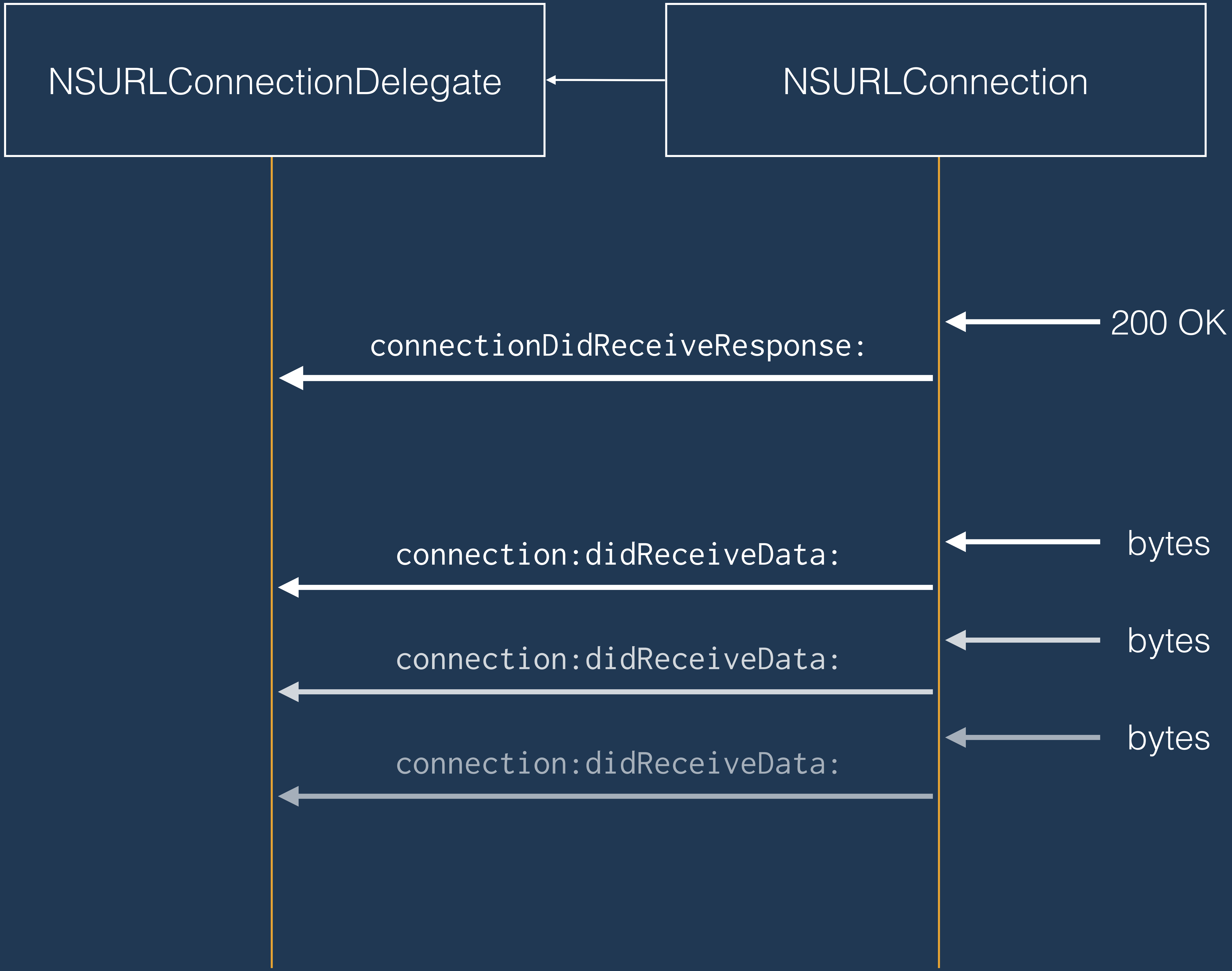
Blocks

```
[NSURLConnection sendAsynchronousRequest:request
                                   queue:[NSOperationQueue mainQueue]
                       completionHandler:^(NSURLResponse *response,
                                           NSData *data,
                                           NSError *connectionError)
 {
     NSString *contents = [[NSString alloc] initWithData:data
                                                 encoding:NSUTF8StringEncoding];
     NSLog(@"%@", contents);

 }];
```

Good, but not always
good enough…

NSURLConnectionDelegate


NSURLConnectionDelegate
NSURLConnection


NSURLConnectionDelegate
NSURLConnection
200 OK
connectionDidReceiveResponse:
bytes
connection:didReceiveData:
bytes
connection:didReceiveData:
bytes
connection:didReceiveData:

```objc
- (void)startMyConnection
{
     self.connection = [[NSURLConnection alloc] initWithRequest:request
                                                       delegate:self];
}

- (void)connection:(NSURLConnection *)connection didReceiveResponse:(NSURLResponse *)response
{
    self.receivedData = [NSMutableData data];
}

- (void)connection:(NSURLConnection *)connection didReceiveData:(NSData *)data
{
    [self.receivedData appendData:data];
}

- (void)connectionDidFinishLoading:(NSURLConnection *)connection
{
    NSString *contents = [[NSString alloc] initWithData:self.receivedData
                                               encoding:NSUTF8StringEncoding];
    NSLog(@"%@", contents);
}
```

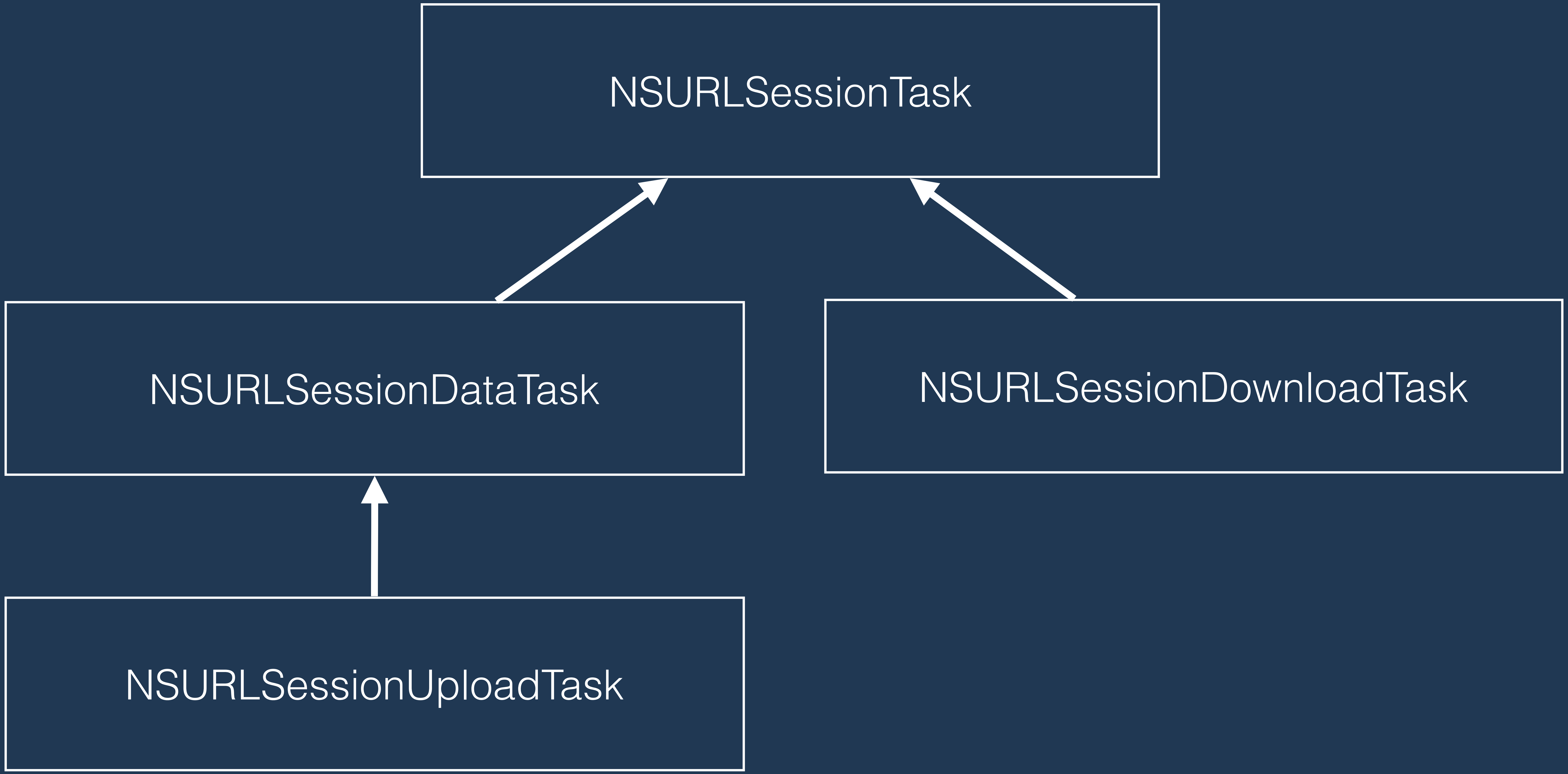
```
@protocol NSURLConnectionDelegate <NSObject>
@optional
- (void)connection:(NSURLConnection *)connection didFailWithError:(NSError *)error;
- (BOOL)connectionShouldUseCredentialStorage:(NSURLConnection *)connection;
- (void)connection:(NSURLConnection *)connection willSendRequestForAuthenticationChallenge:(NSURLAuthenticationChallenge *)challenge;
@end



@protocol NSURLConnectionDataDelegate <NSURLConnectionDelegate>
@optional
- (NSURLRequest *)connection:(NSURLConnection *)connection willSendRequest:(NSURLRequest *)request redirectResponse:(NSURLResponse *)response;
- (void)connection:(NSURLConnection *)connection didReceiveResponse:(NSURLResponse *)response;

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

- (NSInputStream *)connection:(NSURLConnection *)connection needNewBodyStream:(NSURLRequest *)request;
- (void)connection:(NSURLConnection *)connection   didSendBodyData:(NSInteger)bytesWritten
                                                 totalBytesWritten:(NSInteger)totalBytesWritten
                                         totalBytesExpectedToWrite:(NSInteger)totalBytesExpectedToWrite;

- (NSCachedURLResponse *)connection:(NSURLConnection *)connection willCacheResponse:(NSCachedURLResponse *)cachedResponse;

- (void)connectionDidFinishLoading:(NSURLConnection *)connection;
@end



@protocol NSURLConnectionDownloadDelegate <NSURLConnectionDelegate>
@optional
- (void)connection:(NSURLConnection *)connection didWriteData:(long long)bytesWritten totalBytesWritten:(long long)totalBytesWritten
expectedTotalBytes:(long long) expectedTotalBytes;
- (void)connectionDidResumeDownloading:(NSURLConnection *)connection totalBytesWritten:(long long)totalBytesWritten expectedTotalBytes:(long
long) expectedTotalBytes;

@required
- (void)connectionDidFinishDownloading:(NSURLConnection *)connection destinationURL:(NSURL *) destinationURL;
@end
```

NSURLSession

NSURLSession

- Session and task-based authentication via delegate
- Completion handler blocks
- Cancel, pause and resume

NSURLSessionTask


NSURLSessionTask
NSURLSessionDataTask
NSURLSessionDownloadTask
NSURLSessionUploadTask

```
NSURLSession *session = [NSURLSession sharedSession];

NSURLSessionDataTask *task = [session dataTaskWithRequest:request
                                        completionHandler:^(NSData *data,
                                                            NSURLResponse *response,
                                                            NSError *error) {

                                      NSString *contents = [[NSString alloc] initWithData:data
                                                            encoding:NSUTF8StringEncoding];
                                      NSLog(@"%@", contents);

                                  }];

[task resume];
```

NSURLSessionConfiguration

```
+ defaultSessionConfiguration
+ ephemeralSessionConfiguration
+ backgroundSessionConfiguration:
```

AFNetworking

AFNetworking

- Wraps both NSURLRequest and NSURLSession classes
- Request/response serializers
- Must have for iOS 6, still worthwhile in iOS 7

Lab 3.1

NY Times Most Popular

- http://api.nytimes.com/svc/mostpopular/{version}/{resource-type}/{section}[/share-types]/{time-period}[.response-format]?api-key={your-API-key}
- JSON format
- Supports paging via the offset parameter

Articles Paging

- Articles responses contain the total number of articles
- offset in multiples of 20 returns a given page of results