AFNetworking

AFNetworking 是一个能够快速使用的ios和mac os x下的网络框架，它是构建在Foundation URL Loading System之上的，封装了网络的抽象层，可以方便的使用，AFNetworking是一个模块化架构，拥有丰富api的框架。

AFNetworking官方文

档http://cocoadocs.org/docsets/AFNetworking/2.3.1/index.html

一、HTTP请求与操作： 1、AFHTTPRequestOperationManager: 该类封装与Web应用程序进行通信通过HTTP，包括要求制作，响应序列化，网络可达性监控和安全性，以及要求经营管理的常见模式。

`GET 请求：`

```
AFHTTPRequestOperationManager *manager = [AFHTTPRequestOperationManager
manager];
[manager GET:@"http://example.com/resources.json" parameters:nil
success:^(AFHTTPRequestOperation *operation, id responseObject) {
    NSLog(@"JSON: %@", responseObject);
} failure:^(AFHTTPRequestOperation *operation, NSError *error) {
    NSLog(@"Error: %@", error);
}];
```

`POST Multi-Part格式的表单文件上传请求：`

```
AFHTTPRequestOperationManager *manager = [AFHTTPRequestOperationManager
manager];
NSDictionary *parameters = @{@"foo": @"bar"};
NSURL *filePath = [NSURL fileURLWithPath:@"file://path/to/image.png"];
[manager POST:@"http://example.com/resources.json" parameters:parameters
constructingBodyWithBlock:^(id<AFMultipartFormData> formData) {
    [formData appendPartWithFileURL:filePath name:@"image" error:nil];
} success:^(AFHTTPRequestOperation *operation, id responseObject) {
    NSLog(@"Success: %@", responseObject);
} failure:^(AFHTTPRequestOperation *operation, NSError *error) {
    NSLog(@"Error: %@", error);
}];
```

`二、Session管理`：
1、AFURLSessionManager：创建和管理制定的NSURLSession对象

`创建一个上传任务：`

```
NSURLSessionConfiguration *configuration = [NSURLSessionConfiguration
defaultSessionConfiguration];
AFURLSessionManager *manager = [[AFURLSessionManager alloc]
initWithSessionConfiguration:configuration];

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

NSURL *filePath = [NSURL fileURLWithPath:@"file://path/to/image.png"];
NSURLSessionUploadTask *uploadTask = [manager uploadTaskWithRequest:request
fromFile:filePath progress:nil completionHandler:^(NSURLResponse *response,
id responseObject, NSError *error) {
    if (error) {
        NSLog(@"Error: %@", error);
    } else {
        NSLog(@"Success: %@ %@", response, responseObject);
    }
}];
[uploadTask resume];
```

创建一个数据流任务:

```
NSURLSessionConfiguration *configuration = [NSURLSessionConfiguration
defaultSessionConfiguration];
AFURLSessionManager *manager = [[AFURLSessionManager alloc]
initWithSessionConfiguration:configuration];

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

NSURLSessionDataTask *dataTask = [manager dataTaskWithRequest:request
completionHandler:^(NSURLResponse *response, id responseObject, NSError
*error) {
    if (error) {
        NSLog(@"Error: %@", error);
    } else {
        NSLog(@"%@ %@", response, responseObject);
    }
}];
[dataTask resume];
```

使用AFHTTPRequestOperation

1、AFHTTPRequestOperation是使用HTTP或HTTPS协议的AFURLConnectionOperation的子类。 它封装的获取后的HTTP状态和类型将决定请求的成功与否 。

2、虽然AFHTTPRequestOperationManager通常是最好的去请求的方式，但是AFHTTPRequestOpersion也能够单独使用。

通过GET方式：

```
NSURL *URL = [NSURL URLWithString:@"http://example.com/resources/123.json"];
NSURLRequest *request = [NSURLRequest requestWithURL:URL];
AFHTTPRequestOperation *op = [[AFHTTPRequestOperation alloc]
initWithRequest:request];
op.responseSerializer = [AFJSONResponseSerializer serializer];
[op setCompletionBlockWithSuccess:^(AFHTTPRequestOperation *operation, id
responseObject) {
    NSLog(@"JSON: %@", responseObject);
} failure:^(AFHTTPRequestOperation *operation, NSError *error) {
    NSLog(@"Error: %@", error);
}];
[[NSOperationQueue mainQueue] addOperation:op];
```

连续操作多个：

```
NSMutableArray *mutableOperations = [NSMutableArray array];
for (NSURL *fileURL in filesToUpload) {
    NSURLRequest *request = [[AFHTTPRequestSerializer serializer]
multipartFormRequestWithMethod:@"POST"
URLString:@"http://example.com/upload" parameters:nil
constructingBodyWithBlock:^(id<AFMultipartFormData> formData) {
        [formData appendPartWithFileURL:fileURL name:@"images[]" error:nil];
    }];

   AFHTTPRequestOperation *operation = [[AFHTTPRequestOperation alloc]
initWithRequest:request];

   [mutableOperations addObject:operation];
}

NSArray *operations = [AFURLConnectionOperation
batchOfRequestOperations:@[...] progressBlock:^(NSUInteger
numberOfFinishedOperations, NSUInteger totalNumberOfOperations) {
    NSLog(@"%lu of %lu complete", numberOfFinishedOperations,
totalNumberOfOperations);
} completionBlock:^(NSArray *operations) {
    NSLog(@"All operations in batch complete");
}];
[[NSOperationQueue mainQueue] addOperations:operations
waitUntilFinished:NO];
```

1. AFURLConnectionOperation类型

AFURLConnectionOperation类型直接继承自NSOperation类型，它的执行包含如下几个可以说明的地方。

1. NSOperation的状态管理

AFURLConnectionOperation内部有一个AFOperationState类型的state属性，当整个连接操作发生变化后，state属性会改变。同时AFURLConnectionOperation会改写NSOperation的状态返回属性，如isReady, isExecuting, isFinished。这些属性的返回值就是通过内部state属性来决定的。

1. NSOperation的异步执行

AFURLConnectionOperation还会改写NSOperation的isConcurrent属性，并直接返回YES，说明调用start方法后会异步执行其操作。

所有AFURLConnectionOperation的异步执行会运行到一个名称为AFNetworking的线程中，这个线程只会被创建一次，这个线程有一个无限循环来运行线程中的NSRunLoop。

2. AFHTTPRequestOperation类型
AFHTTPRequestOperation类型直接继承上面的AFURLConnectionOperation类型。其意义是在AFURLConnectionOperation类型上，加入了偏向HTTP的处理，如对于整个HTTP操作成功的判断，和暂停和继续处理等。

1. 关于操作的成功判定

核心是两个数据，一个是NSIndexSet类型的acceptableStatusCodes代表可以被认为算成功的HTTP返回码，默认就是200-299。另一个数据是NSSet类型的acceptableContentTypes。也就是需要的MIME类型。

有了这两个数据，AFHTTPRequestOperation就可以判断整个操作是否成功。于是AFHTTPRequestOperation又加入了另一个重要的方法：

- (void)setCompletionBlockWithSuccess:(void (^)(AFHTTPRequestOperation *operation, id responseObject))success

```
                        failure:( void (^)( AFHTTPRequestOperation
*operation, NSError *error))failure;
```

使用它就可以设置两个处理整个操作成功和失败的Block。

1. 关于暂停和继续

这个在AFNetworking中源码里的注释提到过。AFURLConnectionOperation的默认执行是暂停后再执行实际上就是重新开始，但是他的派生类，AFHTTPRequestOperation不会这样做，由于AFHTTPRequestOperation是HTTP导向的，所以它的暂停操作会缓存当前的数据位置，等继续操作后会利用HTTP头中的Range字段来继续上面的操作。