4/23/14 5:23 PM

AFNetWorking How To

Base on AFNetWorking 2.0.3

This document will present to you an incredibly, easy-to-use library, AFNetworking 2.0.3

This document will guide you through the major components of the AFNeworking framework. And how it works, and what is the different between AFNetWorking 2.0.3 to others version.

Thi.Duong

Sutrix Media (Vietnam) JSC.

CONTENT OWNER:

Sutrix Media

1. Signature

Originator By: Thi.Duong	Date: 16/1/2014
Prepared By: Thi.Duong	
	Date: …16.../…01..../…2014…….
Approved By: Hai Nguyen	
	Date: …27.../…02..../…2014…….
Reviewed By: Hai Nguyen	Date: …27.../…02..../…2014…….
Distributed To:	

HCMC Office: 72/4 Truong Quoc Dung,
District Phu Nhuan, HCMC, Viet Nam
☏: +84 8 39975901 🖷: +84 8 3997 5900

HK Office: Unit 706, 7/F., South Seas, Towers,
75 Mody Road, TsimShaTsui, Hong Kong
☏: +852 8197 1217

🖱: www.sutrixmedia.com

2. Revision History

***A – Added, M – Modified, D - Deleted**

Version	Date	A*, M, D	Change Description	Author	Approved By
1	16\1\2014	A	AFNetworking Architecture Document	Thi.Duong	Hai Nguyen
2	07\3\2014	M	Serializations	Thi.Duong	Hai Nguyen
2	22\4\2014	M	Cocoapod to manage Afnetworking lib	Thi.Duong	Hai Nguyen

HCMC Office: 72/4 Truong Quoc Dung,
District Phu Nhuan, HCMC, Viet Nam
☏: +84 8 39975901 🖷: +84 8 3997 5900
HK Office: Unit 706, 7/F., South Seas, Towers,
75 Mody Road, TsimShaTsui, Hong Kong
☏: +852 8197 1217
🖱: www.sutrixmedia.com

3. Table of Contents

HCMC Office: 72/4 Truong Quoc Dung,
District Phu Nhuan, HCMC, Viet Nam
☏: +84 8 39975901 🖷: +84 8 3997 5900

HK Office: Unit 706, 7/F., South Seas, Towers,
75 Mody Road, TsimShaTsui, Hong Kong
☏: +852 8197 1217

🖱: www.sutrixmedia.com

4. Overview

AFNetworking is a delightful networking library for iOS and Mac OS X. It's built on top of the Foundation URL Loading System, extending the powerful high-level networking abstractions built into Cocoa. It has a modular architecture with well-designed, feature-rich APIs that are a joy to use.

Perhaps the most important feature of all, however, is the amazing community of developers who use and contribute to AFNetworking every day. AFNetworking powers some of the most popular and critically-acclaimed apps on the iPhone, iPad, and Mac.

AFNetworking contains everything you need to interface with online resources, from web services to file downloads. It also helps you ensure that your UI is responsive even when your app is in the middle of a big download.

This document will guide you through the major components of the AFNeworking framework. And how it works, and what is the different between AFNetWorking 2.0.3 (which is the newest version) to others version.

HCMC Office: 72/4 Truong Quoc Dung,
District Phu Nhuan, HCMC, Viet Nam
☏: +84 8 39975901 🖷: +84 8 3997 5900
HK Office: Unit 706, 7/F., South Seas, Towers,
75 Mody Road, TsimShaTsui, Hong Kong
☏: +852 8197 1217
🖱: www.sutrixmedia.com

5. Requirements

AFNetworking 2.0.0 and higher requires Xcode 5, targeting either iOS 6.0 and above, or Mac OS 10.8 Mountain Lion (64-bit with modern Cocoa runtime) and above.
For compatibility with iOS 5 or Mac OS X 10.7, use the latest 1.x release.
For compatibility with iOS 4.3 or Mac OS X 10.6, use the latest 0.10.x release.

6. Architecture

CLASSES

AFHTTPRequestOperation

AFHTTPRequestOperation is a subclass of AFURLConnectionOperation for requests using the HTTP or HTTPS protocols. It encapsulates the concept of acceptable status codes and content types, which determine the success or failure of a request.

AFHTTPRequestOperationManager

AFHTTPRequestOperation is a subclass of AFURLConnectionOperation for requests using the HTTP or HTTPS protocols. It encapsulates the concept of acceptable status codes and content types, which determine the success or failure of a request.

AFHTTPSessionManager

AFHTTPSessionManager is a subclass of AFURLSessionManager with convenience methods for making HTTP requests. When a baseURL is provided, requests made with the GET / POST / et al. convenience methods can be made with relative paths; network reachability is also scoped to the host of the base URL as well.

AFNetworking

HCMC Office: 72/4 Truong Quoc Dung, District Phu Nhuan, HCMC, Viet Nam
☎: +84 8 39975901 🖷: +84 8 3997 5900
HK Office: Unit 706, 7/F., South Seas, Towers, 75 Mody Road, TsimShaTsui, Hong Kong
☎: +852 8197 1217
🖱: www.sutrixmedia.com

Just a import class that import all nessary class for project

AFNetworkReachabilityManager

AFNetworkReachabilityManager monitors the reachability of domains, and addresses for both WWAN and WiFi network interfaces.

AFSecurityPolicy

AFSecurityPolicy evaluates server trust against pinned X.509 certificates and public keys over secure connections.

Adding pinned SSL certificates to your app helps prevent man-in-the-middle attacks and other vulnerabilities. Applications dealing with sensitive customer data or financial information are strongly encouraged to route all communication over an HTTPS connection with SSL pinning configured and enabled

AFURLConnectionOperation

AFURLConnectionOperation is a subclass of NSOperation that implements NSURLConnection delegate methods.

AFURLRequestSerialization

is the new class of 2.0.3 AFNetworking, that it self contain three class:

AFHTTPRequestSerializer

AFJSONRequestSerializer

AFPropertyListRequestSerializer

AFURLResponseSerialization

HCMC Office: 72/4 Truong Quoc Dung, District Phu Nhuan, HCMC, Viet Nam
☏: +84 8 39975901 🖷: +84 8 3997 5900
HK Office: Unit 706, 7/F., South Seas, Towers, 75 Mody Road, TsimShaTsui, Hong Kong
☏: +852 8197 1217
🖱: www.sutrixmedia.com

Is a new class in AFNetWorking 2.0.3 that handle the response data type for each request that contain these class:

AFHTTPResponseSerializer

AFJSONResponseSerializer

AFXMLParserResponseSerializer

AFXMLDocumentResponseSerializer *(Mac OS X)*

AFPropertyListResponseSerializer

AFImageResponseSerializer

AFCompoundResponseSerializer

AFURLSessionManager

AFURLSessionManager creates and manages an NSURLSession object based on a specifiedNSURLSessionConfiguration object, which conforms to <NSURLSessionTaskDelegate>,<NSURLSessionDataDelegate>, <NSURLSessionDownloadDelegate>, and <NSURLSessionDelegate>

NSURLConnection

AFURLConnectionOperation

AFHTTPRequestOperation

AFHTTPRequestOperationManager

NSURLSession *(iOS 7 / Mac OS X 10.9)*

AFURLSessionManager

AFHTTPSessionManager

Serialization

<AFURLRequestSerialization>

AFHTTPRequestSerializer

AFJSONRequestSerializer

HCMC Office: 72/4 Truong Quoc Dung, District Phu Nhuan, HCMC, Viet Nam
☏: +84 8 39975901 🖷: +84 8 3997 5900
HK Office: Unit 706, 7/F., South Seas, Towers, 75 Mody Road, TsimShaTsui, Hong Kong
☏: +852 8197 1217
🖱: www.sutrixmedia.com

AFPropertyListRequestSerializer

<AFURLResponseSerialization>

AFHTTPResponseSerializer

AFJSONResponseSerializer

AFXMLParserResponseSerializer

AFXMLDocumentResponseSerializer *(Mac OS X)*

AFPropertyListResponseSerializer

AFImageResponseSerializer

AFCompoundResponseSerializer

Additional Functionality

AFSecurityPolicy

AFNetworkReachabilityManager

HCMC Office: 72/4 Truong Quoc Dung,
District Phu Nhuan, HCMC, Viet Nam
☏: +84 8 39975901 🖷: +84 8 3997 5900
HK Office: Unit 706, 7/F., South Seas, Towers,
75 Mody Road, TsimShaTsui, Hong Kong
☏: +852 8197 1217
🖱: www.sutrixmedia.com

7. Getting Started

Please use Cocoapods to download the latest version from AFNetworking if you don't have it already.

8. Usage

8.1 Based on Response Data Type

Json Request

```
//Init URL request
NSString *weatherUrl = [NSString stringWithFormat:@"%@weather.php?format=json",BaseURLString];
NSURL *url = [NSURL URLWithString:weatherUrl];
NSURLRequest *request = [NSURLRequest requestWithURL:url];

//Fetch json data across the network
AFHTTPRequestOperation *operation = [[AFHTTPRequestOperation alloc] initWithRequest:request];
operation.responseSerializer = [AFJSONResponseSerializer serializer];
[operation setCompletionBlockWithSuccess:^(AFHTTPRequestOperation *operation, id responseObject){
//Handle json data response.

}failure:^(AFHTTPRequestOperation *operation, NSError *error){
//Handle the error response like network isn't available or server have a problem, you should display the error message to let user know.

}
[operation start];
```


HCMC Office: 72/4 Truong Quoc Dung, District Phu Nhuan, HCMC, Viet Nam
☏: +84 8 39975901 🖷: +84 8 3997 5900
HK Office: Unit 706, 7/F., South Seas, Towers, 75 Mody Road, TsimShaTsui, Hong Kong
☏: +852 8197 1217
🖱: www.sutrixmedia.com

Plist Request:

```
//Init URL request
NSString *weatherUrl = [NSString stringWithFormat:@"%@weather.php?format=plist",BaseURLString];
NSURL *url = [NSURL URLWithString:weatherUrl];
NSURLRequest *request = [NSURLRequest requestWithURL:url];

//Fetch plist data across the network
AFHTTPRequestOperation *operation = [[AFHTTPRequestOperation alloc] initWithRequest:request];
operation.responseSerializer = [AFPropertyListResponseSerializer serializer];
[operation setCompletionBlockWithSuccess:^(AFHTTPRequestOperation *operation, id responseObject){
//Handle plist data response.

}failure:^(AFHTTPRequestOperation *operation, NSError *error){
//Handle the error response like network isn't available or server have a problem, you should display the
error message to let user know.

}
[operation start];
```

Notice that this code is almost identical to the JSON version, except for the change of operation type from operation.responseSerializer = [AFJSONResponseSerializer serializer];
to operation.responseSerializer = [AFPropertyListResponseSerializer serializer];

That is the magic of AFNetworking 2.0.3, all request can do in one method, but you have to know your request well, what kind of data it would make, and what kind of data it will return.

HCMC Office: 72/4 Truong Quoc Dung, District Phu Nhuan, HCMC, Viet Nam
☏: +84 8 39975901 🖷: +84 8 3997 5900
HK Office: Unit 706, 7/F., South Seas, Towers, 75 Mody Road, TsimShaTsui, Hong Kong
☏: +852 8197 1217
🖱: www.sutrixmedia.com

In this case the request return plist formats.

XML Request

```
//Init URL request
NSString *weatherUrl = [NSString stringWithFormat:@"%@weather.php?format=xml",BaseURLString];
NSURL *url = [NSURL URLWithString:weatherUrl];
NSURLRequest *request = [NSURLRequest requestWithURL:url];

//Fetch plist data across the network
AFHTTPRequestOperation *operation = [[AFHTTPRequestOperation alloc] initWithRequest:request];
operation.responseSerializer = [AFXMLParserResponseSerializer serializer];
[operation setCompletionBlockWithSuccess:^(AFHTTPRequestOperation *operation, id responseObject){
//Handle and parse XML data response
        self.xmlWeather = [NSMutableDictionary dictionary];
        NSXMLParser *XMLParser = responseObject;
        XMLParser.delegate = self;
        XMLParser setShouldProcessNamespaces:YES];
        [XMLParser parse];
} failure:^(AFHTTPRequestOperation *operation, NSError *error){
//Handle the error response.

}
[operation start];
```

The same as XML Request in the success block the response object acctualy is an NSXMLParser object.

HCMC Office: 72/4 Truong Quoc Dung,
District Phu Nhuan, HCMC, Viet Nam
☏: +84 8 39975901 🖷: +84 8 3997 5900
HK Office: Unit 706, 7/F., South Seas, Towers,
75 Mody Road, TsimShaTsui, Hong Kong
☏: +852 8197 1217
🖱: www.sutrixmedia.com

8.2 Based on HTTP Request Method

POST HTTP Request:

```
//Init URL request and request parameters
NSString *weatherUrl = [NSString stringWithFormat:@"%@weather.php?format=xml",BaseURLString];
NSURL *url = [NSURL URLWithString:weatherUrl];
NSDictionary *parameters = @{@"Key1": @"value1", @"Key2": @"value2"};
//Setup the serializer request/response and Fetch data across network
AFHTTPRequestOperationManager *manager = [AFHTTPRequestOperationManager manager];
[manager.requestSerializer setValue:@"Accept" forHTTPHeaderField:@"application/json"];
manager.securityPolicy.allowInvalidCertificates = YES;
manager.requestSerializer = [AFHTTPRequestSerializer serializer];
manager.responseSerializer = [AFJSONResponseSerializer serializer];
[manager POST:@"http://www.example.com/response.json" parameters:parameters
success:^(AFHTTPRequestOperation *operation, id responseObject){
//Handle json response
self.weather = responseObject;
}failure:^(AFHTTPRequestOperation *operation, NSError *error){
//Handle the error response.
}
```

This is another powerful way to make a request. Just passing the whole dictionary and we good to go.

GET HTTP Request

```
//Init URL request and request parameters
NSString *weatherUrl = [NSString stringWithFormat:@"%@weather.php?format=xml",BaseURLString];
NSURL *url = [NSURL URLWithString:weatherUrl];
NSDictionary *parameters = @{@"Key1": @"value1", @"Key2": @"value2"};
```


HCMC Office: 72/4 Truong Quoc Dung, District Phu Nhuan, HCMC, Viet Nam
☏: +84 8 39975901 🖷: +84 8 3997 5900
HK Office: Unit 706, 7/F., South Seas, Towers, 75 Mody Road, TsimShaTsui, Hong Kong
☏: +852 8197 1217
🖱: www.sutrixmedia.com

```
//Setup the serializer request/response and Fetch data across network
AFHTTPRequestOperationManager *manager = [AFHTTPRequestOperationManager manager];
[manager.requestSerializer setValue:@"Accept" forHTTPHeaderField:@"application/json"];
manager.securityPolicy.allowInvalidCertificates = YES;
manager.requestSerializer = [AFHTTPRequestSerializer serializer];
manager.responseSerializer = [AFJSONResponseSerializer serializer];
[manager GET:@"http://www.example.com/response.json" parameters:parameters
success:^(AFHTTPRequestOperation *operation, id responseObject){
self.weather = responseObject;
self.title = @"HTTP POST";
[self.tableView reloadData];
}failure:^(AFHTTPRequestOperation *operation, NSError *error){
//Handle the error response.
}
```

You can see it totally the same as GET Http Request.

8.3 Blocks & Serialization

Blocks

AFNetworking 2 made blocks itself. Block is known as a pointer of a function which is under a variable with executive code inside its body.

A block pointer lets us handle and store blocks, so that we can pass around blocks to functions or have functions return blocks - stuff that we normally do with variables and objects.

By thy way, AFNetworking 2 is easy to pass a bunche of data / variables without reference to a function.

Serialization

AFNetworking 2 has moved to a new model of serializers to add a bit more modularity. This means that requests can now have a request serializer that handles generating the request based on user parameters and doing tasks such as authorization and the response serializer can now generate user data from the server response.

HCMC Office: 72/4 Truong Quoc Dung, District Phu Nhuan, HCMC, Viet Nam
☏: +84 8 39975901 🖷: +84 8 3997 5900
HK Office: Unit 706, 7/F., South Seas, Towers, 75 Mody Road, TsimShaTsui, Hong Kong
☏: +852 8197 1217
🖱: www.sutrixmedia.com

<AFURLRequestSerializer> - Objects conforming to this protocol are used to decorate requests by translating parameters into either a query string or entity body representation, as well as setting any necessary headers.

<AFURLResponseSerializer> - Objects conforming to this protocol are responsible for validating and serializing a response and its associated data into useful representations

Both <AFURLRequestSerializer> & <AFURLResponseSerializer> are lightweight protocols, with a single method each:

```
- (NSURLRequest *)requestBySerializingRequest:(NSURLRequest *)request
withParameters:(NSDictionary *)parameters  error:(NSError *__autoreleasing *)error
```

```
- (id)responseObjectForResponse:(NSURLResponse *)response data:(NSData *)data
error:(NSError *__autoreleasing *)error
```

How to use

In the MVC pattern, we separate the responsibility to 3 main object :

1. Model : this contains everything that we want to do for app data (low layer). For example : create data object, process data, synchronize data functions
2. View : this contains all sub-sections could perform the layout on screen
3. Controllers : this contains all classes which are declared to init a layout, control all views to perform screen behavior. It can send a request to load data, synchronize data, display data on screen, remove all data….

Due to MVC integration in project, we have to decouple everything when our app interacts to web services directly, it make us easy to maintain code and know exactly where we should put our classes due to their responsibility.

AFNetworking make a request to server and handle the response in a object called :

id responseObject

a general object type in object-c.

HCMC Office: 72/4 Truong Quoc Dung, District Phu Nhuan, HCMC, Viet Nam
☏: +84 8 39975901 🖷: +84 8 3997 5900
HK Office: Unit 706, 7/F., South Seas, Towers, 75 Mody Road, TsimShaTsui, Hong Kong
☏: +852 8197 1217
🖱: www.sutrixmedia.com

It means the reponse data type which AFNetworking handles could be any thing : NSDictionary, NSArray,.. we can map to the type we want to have.
But, our purpose is to go deeply inside the id responseObject . Our data object should be created at this level and all its attributes have been assigned value. This data processing is completely decoupled from view controller, we don't have to add many code lines to process data from web service connection class and view controller class.
The below decription is for JSon data request as a example to know how to do
Now, we create subclass of Request Serializer and Response Serializer :

WSJSONResponseSerializer.h

```
#import "AFURLResponseSerialization.h"
/**
 * @brief
 * The WSJSONResponseSerializer class implements a specialized AFJSONResponseSerializer that encapsulates
 * a block object designed to map the JSON response to models. Its purpose is to avoid creating a class for each request.
 *
 * It overrides the `- (id)responseObjectForResponse:(NSURLResponse *)response data:(NSData *)data error:(NSError *__autoreleasing *)error`
 * method to:
 * - log response to standard output,
 * - execute the block object with the response object coming from superclass call.
 */
@interface WSJSONResponseSerializer : AFJSONResponseSerializer <AFURLResponseSerialization>
/**
 * A block object to be executed to map the JSON response to models.
 *
 * @param responseObject    The JSON response as a dictionary.
 * @param headers           The HTTP headers.
 * @param error             The error that occurred while attempting to decode the response data.
```


HCMC Office: 72/4 Truong Quoc Dung, District Phu Nhuan, HCMC, Viet Nam
☏: +84 8 39975901 🖷: +84 8 3997 5900
HK Office: Unit 706, 7/F., South Seas, Towers, 75 Mody Road, TsimShaTsui, Hong Kong
☏: +852 8197 1217
🖱: www.sutrixmedia.com

```
 */
@property (copy) void (^mapResponse)(NSDictionary *responseObject, NSDictionary *headers, NSError
**error);
@end
```

WSJSONResponseSerializer.m

```
#import "WSJSONResponseSerializer.h"

@implementation WSJSONResponseSerializer
```

```
- (id)responseObjectForResponse:(NSURLResponse *)response data:(NSData *)data error:(NSError
*__autoreleasing *)error {

    NSLog(@"%@", [[NSString alloc] initWithData:data encoding:self.stringEncoding]);
    id responseObject = [super responseObjectForResponse:response data:data error:error];
    if (self.mapResponse && responseObject && [responseObject isKindOfClass:[NSDictionary class]] &&
!*error) {
        self.mapResponse(responseObject, [(NSHTTPURLResponse *)response allHeaderFields], error);
    }
    return responseObject;
}
@end
```

By using a block (^mapResponse), we easy add executive code to process data object returned from ws request, for example :

```
- (void)loginWithEmail:(NSString *)email password:(NSString *)password {
    [self.activityIndicator startAnimating];
    WSJSONResponseSerializer *serializer = [[WSJSONResponseSerializer alloc] init];
    serializer.mapResponse = ^(NSDictionary *responseObject, NSDictionary *headers, NSError **error) {
        if (![[AppData appData] userLoggedInWithData:responseObject]) {
            // TODO: create *error
        }
```


HCMC Office: 72/4 Truong Quoc Dung, District Phu Nhuan, HCMC, Viet Nam
☏: +84 8 39975901 🖷: +84 8 3997 5900
HK Office: Unit 706, 7/F., South Seas, Towers, 75 Mody Road, TsimShaTsui, Hong Kong
☏: +852 8197 1217
🖱: www.sutrixmedia.com

```
    };
    WSRequestOperationManager *manager = [WSRequestOperationManager backendManager];
    [manager setJSONRequestSerializer:nil responseSerializer:serializer];
    NSDictionary *parameters = @{loginKey: email, passwordKey: password};
    [manager POST:[[Configuration configuration] loginURL]
      parameters:parameters
        success:^(AFHTTPRequestOperation *operation, id responseObject) {
            [self.activityIndicator stopAnimating];
            if ([self.delegate respondsToSelector:@selector(loginDidSuccess:)]) {
                [self.delegate loginDidSuccess:self];
            }
        }
        failure:^(AFHTTPRequestOperation *operation , NSError *error) {
            [self.activityIndicator stopAnimating];
            UIAlertView *alertView = [[UIAlertView alloc] initWithTitle:nil message:manager.errorMessage
delegate:self cancelButtonTitle:@"OK" otherButtonTitles:nil, nil];
            [alertView show];
        }];
}
```

As you can see [[AppData appData] userLoggedInWithData:responseObject] is created to parse user info which returned by sending a request to Login web service. The main thread will be blocked, to wait for the code in block ^mapResponse is executed completely. Then a delegate method will be called to the controller knows it's time to load data on screen.

By this way, we can send any request to server and easy process response data based on its type.

To decouple the request and reponse, we continue to create a custom class to handle all supported info for a request.

WSRequestOperationManager.h

```
#import "AFHTTPRequestOperationManager.h"
```

```
@class WSJSONResponseSerializer;
@class WSXMLResponseSerializer;
```

```
/**
 * @brief
 * The WSRequestOperationManager class implements a specialized AFHTTPRequestOperationManager
that manages
 * network and server errors.
 *
 * The class defines a singleton for backend requests and another one for CQ4 requests.
```

```
 */
@interface WSRequestOperationManager : AFHTTPRequestOperationManager
/**
 * The error message to display in case of failure or `nil` in case of success
 */
@property (nonatomic, strong) NSString *errorMessage;
```

```
/**
 * @return The singleton manager instance for all backend requests.
 */
+ (instancetype)backendManager;
/**
 * @return The singleton manager instance for all CQ4 requests.
 */
+ (instancetype)CQ4Manager;
/**
 * Tells if network is available for the base URL of this manager.
```

```
 *
 * @return `YES` if network is available for the base URL of this manager, `NO` otherwise.
 */
- (BOOL)isReachable;
/**
 * Setups the request and response serializers.
```

```
 *
 * @param requestSerializer     the specified request serializer or `nil` to use the default one.
 * @param responseSerializer    the specified response serializer or `nil` to use the default one.
```

```
 */
- (void)setJSONRequestSerializer:(AFJSONRequestSerializer *)requestSerializer
        responseSerializer:(WSJSONResponseSerializer *)responseSerializer;
/**
```

```
 * Setups the request and response serializers.
 *
 * @param requestSerializer     the specified request serializer or `nil` to use the default one.
 * @param responseSerializer    the specified response serializer or `nil` to use the default one.
 * @param authorization         `YES` if authorization must be added to the HTTP header field "Authorization",
 *                              `NO` otherwise
 */
- (void)setJSONRequestSerializer:(AFJSONRequestSerializer *)requestSerializer
        responseSerializer:(WSJSONResponseSerializer *)responseSerializer
            authorization:(BOOL)authorization;

- (void)setXMLRequestWithResponseSerializer:(WSXMLResponseSerializer *)responseSerializer;
```

```
- (void)setXMLRequestWithResponseSerializer:(WSXMLResponseSerializer *)responseSerializer
                    authorization:(BOOL)authorization;
@end
```


HCMC Office: 72/4 Truong Quoc Dung,
District Phu Nhuan, HCMC, Viet Nam
☏: +84 8 39975901 🖷: +84 8 3997 5900
HK Office: Unit 706, 7/F., South Seas, Towers,
75 Mody Road, TsimShaTsui, Hong Kong
☏: +852 8197 1217
🖱: www.sutrixmedia.com

We can store a token (authorization) or any info from response headers which we want to use.

In failure block, we handle all response code error to display message to UI. This function returns a message in NSString type, we can get it from any where via this custom webservice request class.

9. Another Usage

AFURLSessionManager

AFURLSessionManager creates and manages an NSURLSession object based on a specifiedNSURLSessionConfiguration object, which conforms to <NSURLSessionTaskDelegate>,<NSURLSessionDataDelegate>, <NSURLSessionDownloadDelegate>, and <NSURLSessionDelegate>.

Creating a Download Task

```
//Init URL request
NSURLSessionConfiguration *configuration = [NSURLSessionConfiguration defaultSessionConfiguration];
AFURLSessionManager *manager = [[AFURLSessionManager alloc]
initWithSessionConfiguration:configuration];
NSURL *URL = [NSURL URLWithString:@"http://example.com/download.zip"];
NSURLRequest *request = [NSURLRequest requestWithURL:URL];

//Starting to download package
NSURLSessionDownloadTask *downloadTask = [manager downloadTaskWithRequest:request progress:nil
destination:^NSURL *(NSURL *targetPath, NSURLResponse *response){
        NSURL *documentsDirectoryPath = [NSURL
fileURLWithPath:[NSSearchPathForDirectoriesInDomains(NSDocumentDirectory, NSUserDomainMask, YES)
firstObject]];
        return [documentsDirectoryPath URLByAppendingPathComponent:[targetPath
lastPathComponent]];
```


HCMC Office: 72/4 Truong Quoc Dung, District Phu Nhuan, HCMC, Viet Nam
☏: +84 8 39975901 🖷: +84 8 3997 5900
HK Office: Unit 706, 7/F., South Seas, Towers, 75 Mody Road, TsimShaTsui, Hong Kong
☏: +852 8197 1217
🖱: www.sutrixmedia.com

```
} completionHandler:^(NSURLResponse *response, NSURL *filePath, NSError *error) {
    NSLog(@"File downloaded to: %@", filePath);
}];
[downloadTask resume];
```

Creating a Data Task

```
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];
```

Creating an Upload Task

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


HCMC Office: 72/4 Truong Quoc Dung,
District Phu Nhuan, HCMC, Viet Nam
☏: +84 8 39975901 🖷: +84 8 3997 5900
HK Office: Unit 706, 7/F., South Seas, Towers,
75 Mody Road, TsimShaTsui, Hong Kong
☏: +852 8197 1217
🖱: www.sutrixmedia.com

```
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];
```


HCMC Office: 72/4 Truong Quoc Dung,
District Phu Nhuan, HCMC, Viet Nam
☏: +84 8 39975901 🖷: +84 8 3997 5900
HK Office: Unit 706, 7/F., South Seas, Towers,
75 Mody Road, TsimShaTsui, Hong Kong
☏: +852 8197 1217
🖱: www.sutrixmedia.com