AFNetworking

nate

Code

Commits

Issues 966,347

Wikis

Users 159,770

Languages

JavaScript	4,918
Java	2,300
Python	1,686
Objective-C	×
Ruby	1,132
PHP	798
Go	772
C	711
C++	688
Swift	612

AFNetworking/AFNetworking

A delightful networking framework for iOS, OS X, watchOS, and tvOS.

Objective-C ★ 28,360 8,991 Updated 6 days ago

rs/SDWebImage

Asynchronous image downloader with cache support as a UIImageView category

Objective-C ★ 16,879 4,645 Updated a day ago

BradLarson/GPUImage

An open source iOS framework for GPU-based image and video processing

Objective-C ★ 14,927 3,719 Updated on 15 Jan

SnapKit/Masonry

Harness the power of AutoLayout NSLayoutConstraints with a simplified, chainable and expressive syntax. Supports iOS ...

为什么分析这个框架

基于Foundation URL Loading System的网络框架。成千上万的APP集成。

- 版本变化
- 框架架构
- 网络连接核心部分代码
- 代码分享

NSURLConnection -> URLSession

AFNetworking Version	Minimum iOS Target	Minimum OS X Target	Minimum watchOS Target	Minimum tvOS Target	Notes
3.x	iOS 7	OS X 10.9	watchOS 2.0	tvOS 9.0	Xcode 7+ is required. `NSURLConnectionOperation` support has been removed.
2.6 -> 2.6.3	iOS 7	OS X 10.9	watchOS 2.0	n/a	Xcode 7+ is required.
2.0 -> 2.5.4	iOS 6	OS X 10.8	n/a	n/a	Xcode 5+ is required. `NSURLSession` subspec requires iOS 7 or OS X 10.9.
1.x	iOS 5	Mac OS X 10.7	n/a	n/a	
0.10.x	iOS 4	Mac OS X 10.6	n/a	n/a	

文件结构

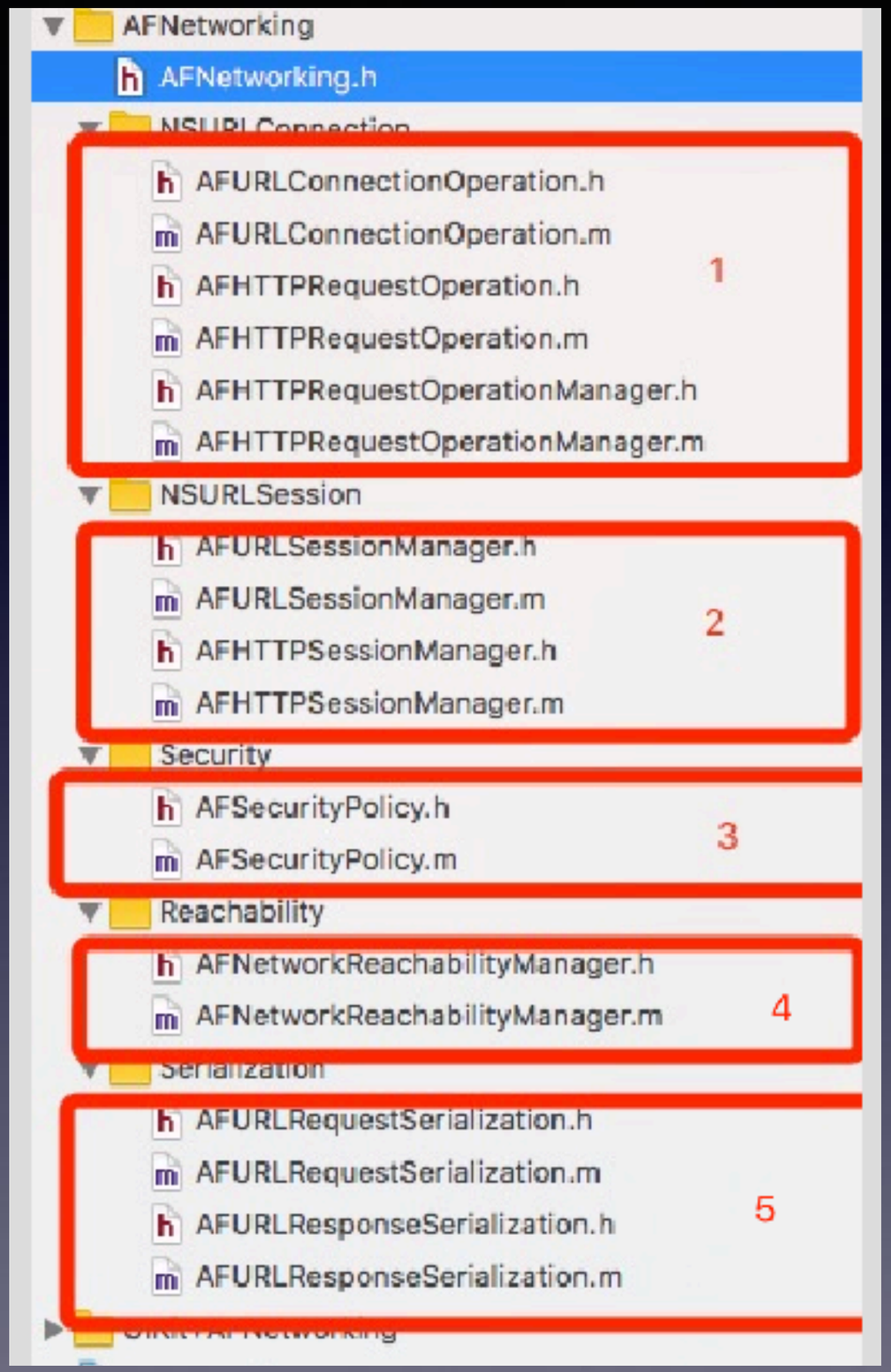
1、NSURLConnection

2、URLSession

3、安全部分

4、检测工具

5、拼装和解析



基本使用

GET Request

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

POST URL-Form-Encoded Request

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

POST Multi-Part Request

```
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<AFMultipartFo
    [formData appendPartWithFileURL:filePath name:@"image" error:nil];
} success:^(AFHTTPRequestOperation *operation, id responseObject) {
    NSLog(@"Success: %@", responseObject);
} failure:^(AFHTTPRequestOperation *operation, NSError *error) {
    NSLog(@"Error: %@", error);
}];
```

HTTP Manager Reachability

```
NSURL *baseURL = [NSURL URLWithString:@"http://example.com/"];
AFHTTPRequestOperationManager *manager = [[AFHTTPRequestOperationManager alloc] initWithBaseURL:baseURL];

NSOperationQueue *operationQueue = manager.operationQueue;
[manager.reachabilityManager setReachabilityStatusChangeBlock:^(AFNetworkReachabilityStatus status) {
    switch (status) {
        case AFNetworkReachabilityStatusReachableViaWWAN:
        case AFNetworkReachabilityStatusReachableViaWiFi:
            [operationQueue setSuspended:NO];
            break;
        case AFNetworkReachabilityStatusNotReachable:
        default:
            [operationQueue setSuspended:YES];
            break;
    }
}];

[manager.reachabilityManager startMonitoring];
```

网络流程

- NSURLConnection + NSOperation
- AFURLRequestSerializer
- NSURLResponse
- AFURLResponseSerializer
- Data

request组装

```
NSError *serializationError = nil;
NSMutableURLRequest *request = [self.requestSerializer requestWithMethod:
    method URLString:[[NSURL URLWithString:URLString relativeToURL:self.
    baseURL] absoluteString] parameters:parameters error:&
    serializationError];
```

```
NSURL *url = [NSURL URLWithString:URLString];

NSParameterAssert(url);

NSMutableURLRequest *mutableRequest = [[NSMutableURLRequest alloc]
    initWithURL:url];
mutableRequest.HTTPMethod = method;

for (NSString *keyPath in AFHTTPRequestSerializerObservedKeyPaths()) {
    if ([self.mutableObservedChangedKeyPaths containsObject:keyPath]) {
        [mutableRequest setValue:[self valueForKeyPath:keyPath] forKey:
            keyPath];
    }
}

mutableRequest = [[self requestBySerializingRequest:mutableRequest
    withParameters:parameters error:error] mutableCopy];
```

请求

```
AFHTTPRequestOperation *operation = [self
    HTTPRequestOperationWithHTTPMethod:@"GET" URLString:URLString
    parameters:parameters success:success failure:failure];

[self.operationQueue addOperation:operation];

return operation;
```

- NSOperation
- -setCompletionBlock:
- -isReady
- -isExecuting
- -isFinished
- -isConcurrent
- -start
- -operationDidStart
- -finish
- -cancel
- -cancelConnection

NSURLConnectionDelegate

- -connection:willSendRequestForAuthenticationChallenge:
- -connectionShouldUseCredentialStorage:
- -connection:willSendRequest:redirectResponse:
- -connection:didSendBodyData:totalBytesWritten:totalBytesExpectedToWrite:
- -connection:didReceiveResponse:
- -connection:didReceiveData:
- -connectionDidFinishLoading:
- -connection:didFailWithError:
- -connection:willCacheResponse:

```
    if (success) {
        dispatch_group_async(self.completionGroup ?:
            http_request_operation_completion_group(), self.
            completionQueue ?: dispatch_get_main_queue(), ^{
            success(self, responseObject);
        });
    }
}
```

解析

```
- (id)responseObject {
    [self.lock lock];
    if (!_responseObject && [self isFinished] && !self.error) {
        NSError *error = nil;
        self.responseObject = [self.responseSerializer
            responseObjectForResponse:self.response data:self.responseData
            error:&error];
        if (error) {
            self.responseSerializationError = error;
        }
    }
    [self.lock unlock];

    return _responseObject;
}
```

忽略警告

```
#pragma clang diagnostic push
#pragma clang diagnostic ignored "-Wgnu"
            dispatch_async(self.completionQueue ?: dispatc
                ^{
                    failure(nil, serializationError);
            });
#pragma clang diagnostic pop
```

线程初始化

```
static dispatch_queue_t url_request_operation_completion_queue() {
    static dispatch_queue_t af_url_request_operation_completion_queue;
    static dispatch_once_t onceToken;
    dispatch_once(&onceToken, ^{
        af_url_request_operation_completion_queue =
            dispatch_queue_create("com.alamofire.networking.operation.queue",
            DISPATCH_QUEUE_CONCURRENT );
    });

    return af_url_request_operation_completion_queue;
}
```

循环引用

```
__weak __typeof(self)weakSelf = self;
[super setCompletionBlock:^ {
    __strong __typeof(weakSelf)strongSelf = weakSelf;

ma clang diagnostic push
ma clang diagnostic ignored "-Wgnu"
    dispatch_group_t group = strongSelf.completionGroup ?:
        url_request_operation_completion_group();
    dispatch_queue_t queue = strongSelf.completionQueue ?:
        dispatch_get_main_queue();
ma clang diagnostic pop

    dispatch_group_async(group, queue, ^{
        block();
    });

    dispatch_group_notify(group,
        url_request_operation_completion_queue(), ^{
        //清理完成block
        [strongSelf setCompletionBlock:nil];
    });
}];
```

状态机和锁

```
- (void)finish {
    [self.lock lock];
    self.state = AFOperationFinishedState;
    [self.lock unlock];

    dispatch_as  c(dispatch_get_main_queue(), ^{
        [[NSN  ificationCenter defaultCenter] postNotificationName:
            AFNetworkingOperationDidFinishNotification object:self];
    });
}
```

1

2

NSRecursiveLock它可以允许同一线程多次加锁，而不会造成死锁。递归锁会跟踪它被lock的次数。每次成功的lock都必须平衡调用unlock操作。只有所有达到这种平衡，锁最后才能被释放，以供其它线程使用。

runloop

```
self.connection = [[NSURLConnection alloc] initWithRequest:self.
    request delegate:self startImmediately:NO];

NSRunLoop *runLoop = [NSRunLoop currentRunLoop];
for (NSString *runLoopMode in self.runLoopModes) {
    //所有mode都接受数据
    [self.connection scheduleInRunLoop:runLoop forMode:runLoopMode];
    [self.outputStream scheduleInRunLoop:runLoop forMode:runLoopMode]
    ;
}

[self.outputStream open];
[self.connection start];
```

delegate回调在各种runloop mode下触发

常驻线程

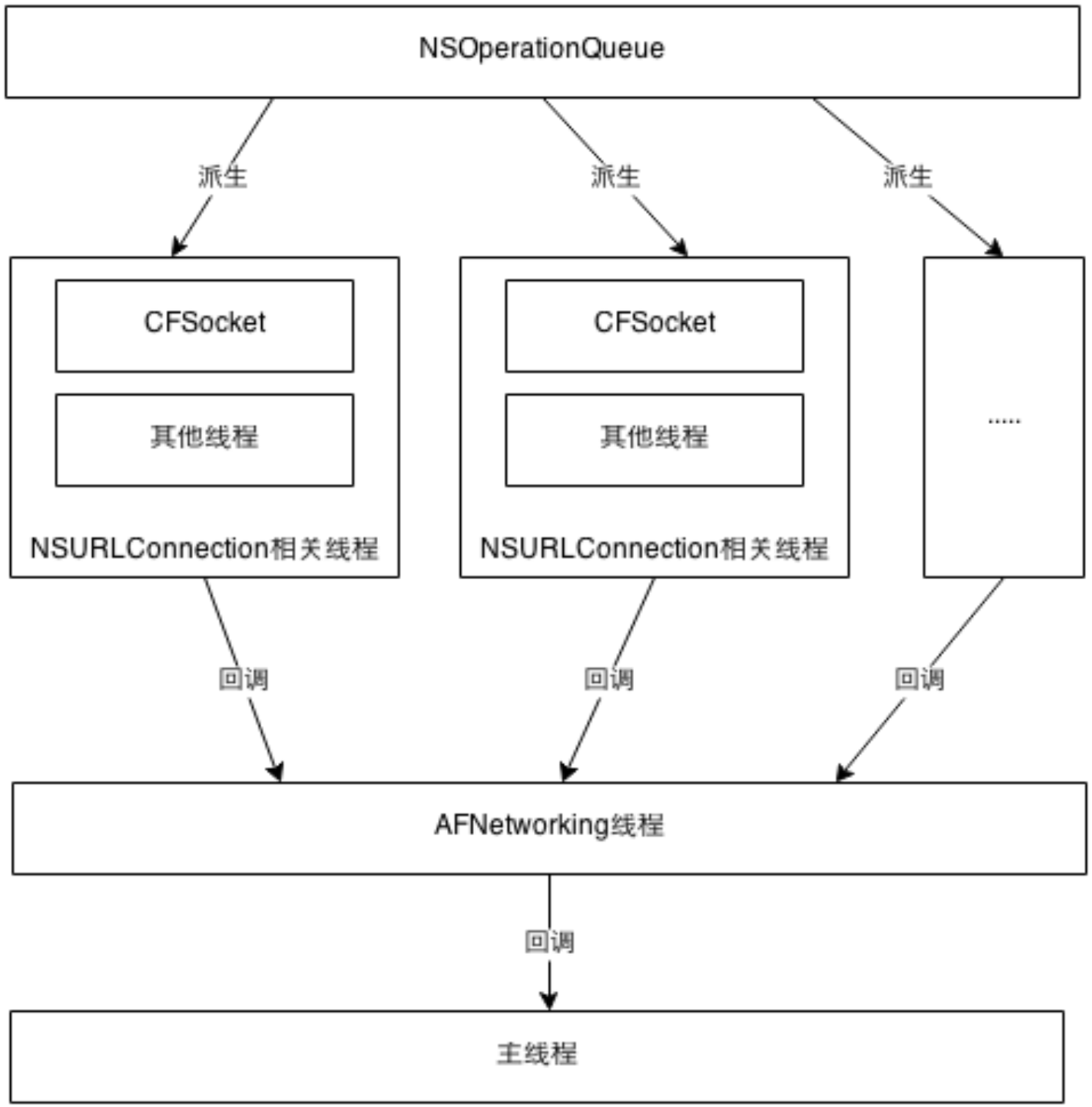
```
+ (void)networkRequestThreadEntryPoint:(id)__unused object {
    @autoreleasepool {
        [[NSThread currentThread] setName:@"AFNetworking"];

        // 修改线程runloop模式，使其常驻后台
        NSRunLoop *runLoop = [NSRunLoop currentRunLoop];
        [runLoop addPort:[NSMachPort port] forMode:NSDefaultRunLoopMode];
        [runLoop run];
    }
}

//获取常驻线程
+ (NSThread *)networkRequestThread {
    static NSThread *_networkRequestThread = nil;
    static dispatch_once_t oncePredicate;
    dispatch_once(&oncePredicate, ^{
        //nsthread方式创建
        _networkRequestThread = [[NSThread alloc] initWithTarget:self
            selector:@selector(networkRequestThreadEntryPoint:) object:nil];
        [_networkRequestThread start];
    });

    return _networkRequestThread;
}
```

delegate回调的线程


NSOperationQueue
派生
派生
派生
CFSocket
其他线程
NSURLConnection相关线程
CFSocket
其他线程
NSURLConnection相关线程
.....
回调
回调
回调
AFNetworking线程
回调
主线程

GCD的线程调度

```
clang diagnostic ignored -Wgnu
    dispatch_group_t group = strongSelf.completionGroup ?:
        url_request_operation_completion_group();
    dispatch_queue_t queue = strongSelf.completionQueue ?: dispatch_get_main_queue(
clang diagnostic pop

    dispatch_group_async(group, queue, ^{
        block();
    });

    dispatch_group_notify(group, url_request_operation_completion_queue(), ^{
        //清理完成block
        [strongSelf setCompletionBlock:nil];
    });
```

设计思想

- 易用自解释的API
- 模块化组件
- 分层设计

谢谢