

基于AFN的二次封装



作者 laitys (/u/45c32cefd5b9) + 关注

2017.03.28 17:45* 字数 1605 阅读 56 评论 1 喜欢 3

(/u/45c32cefd5b9)

一.前言

最近重构项目，遇到了很多问题，也从中总结出了几点经验，这里先讲一讲网络工具类的封装，网络请求是前端和后台沟通的桥梁，那么前端网络工具类的封装或者使用是一个项目的基础。由于本人现做的项目“年久失修、满目疮痍”，项目14年建立之后经历不下于5人之手，每人编程习惯不同，勿论对错，但从网络工具的使用来看就显得非常之乱：有用原生NSURLSession（NSURLSession是苹果在iOS7后为HTTP数据传输提供的一系列接口,比NSURLConnection强大,坑少,好用）、有用原生NSURLConnection（NSURLConnection在iOS9被宣布弃用）、有用ANFNetWorking(本人弃用ASI后就一直在用AFN它是对NSURLSession封装较好作者并持续更新的框架)、还有用Overcoat。为了统一网络请求，方便以后开发，我就封装了网络工具类，此工具类基于AFNetWorking，封装了GET、POST、Upload、Download方法，关于网络监测之前文章有写，在此不做赘述，欢迎留言批评指正！

二.封装思路

- 创建网络工具类单例，在单例创建的时候配置相关参数，例如：新增一些响应解析器能够接受的数据类型（接受数据不全导致网络请求错误是AFN的一大坑）、设置超时时间、配置请求头（这里我们把access_token验证放到了请求头里，又的项目是放到了请求体或参数里）、配置相应器和解析器等
- 基于AFHTTPSessionManager（是对NSURLSession的封装）GET、POST、Upload、Download方法二次封装
- 上传Upload方法需要上传数据的参数，所以这里建立了个参数模型：WKUploadParam

三.源码

- 网络工具类 WKNetWorkTool.h

```

/*
**此网络工具类是LWK新建网络工具类
*为统一管理降低耦合新建模块均用此工具类
*
*
*/

#import <Foundation/Foundation.h>
#import <AFNetworking/AFHTTPSessionManager.h>
#import "WKUploadParam.h"
@interface WKNetWorkTool : AFHTTPSessionManager

/**
 * 二次封装网络工具单例
 *
 */
+ (instancetype)sharedTool;

/**
 * 基于AFN二次封装GET方法
 *
 * @URLString 相对路径
 * @params 请求参数
 * @finish 完成回调
 *
 */
- (void)requestGET:(NSString *)URLString parames:(id)parames success:(void (^)(id responseObj))success failure:(void (^)(id error))failure;

/**
 * 基于AFN二次封装POST方法
 *
 * @URLString 相对路径
 * @params 请求参数
 * @finish 完成回调
 */
- (void)requestPOST:(NSString *)URLString parames:(id)parames success:(void (^)(id responseObj))success failure:(void (^)(id error))failure;

/**
 * 上传图片（单张或一组）
 *
 * @param URLString 上传图片的网址字符串
 * @param parameters 上传图片的参数
 * @param uploadParam 上传图片的信息
 * @param success 上传成功的回调
 * @param failure 上传失败的回调
 */
- (void)uploadWithURLString:(NSString *)URLString
parameters:(id)parameters
uploadParam:(NSArray <WKUploadParam *> *)uploadParams
success:(void (^)())success
failure:(void (^)(NSError *error))failure;

/**
 * 下载数据
 *
 * @param URLString 下载数据的网址
 * @param parameters 下载数据的参数
 * @param success 下载成功的回调
 * @param failure 下载失败的回调
 */
- (void)downloadWithURLString:(NSString *)URLString
parameters:(id)parameters
progerss:(void (^)())progress
success:(void (^)())success
failure:(void (^)(NSError *error))failure;

@end

```

- 网络工具类 *WKNetWorkTool.m*

```

#import "WKNetWorkTool.h"
#import "SDHSDDataCache.h"
@implementation WKNetWorkTool

#pragma mark - 单例

+ (instancetype)sharedTool{
    static dispatch_once_t onceToken;
    static WKNetWorkTool *instance;
    dispatch_once(&onceToken, ^{
        instance = [[super alloc] init];
        instance.responseSerializer = [AFJSONResponseSerializer serializer];
        AFHTTPRequestSerializer *requestSerializer = [AFHTTPRequestSerializer serial

```

```
izer];
    requestSerializer.timeoutInterval = 10;
    ///我们项目是把access_token（后台验证用户省份标识）放在了请求头里,有的项目是放在了请求体里,视实际情况而定
    [requestSerializer setValue:[SDHSDataCache getToken] forKey:@"access_token"];
    instance.requestSerializer = requestSerializer;

    //          ///1.强制更换AFN数据解析类型 只支持一下添加的数据类型 AFN自带的就没有了 如果AFN新增了数据解析类型 这里也没有变化 所以有下面2方法 向原有可解析数据类型添加较好
    //          instance.responseSerializer.acceptableContentTypes = [NSSet initWithObjects:@"application/json", @"text/json", @"text/javascript",@"text/html",@"plant/html",nil];8k9K10

    ///2.获取AFN原有的数据解析类型 然后新增一些响应解析器能够接受的数据类型
    NSMutableSet *acceptableContentTypes = [NSMutableSet initWithSet:instance.responseSerializer.acceptableContentTypes];
    [acceptableContentTypes addObjectsFromArray:@[@"application/json", @"text/json", @"text/javascript",@"text/html",@"plant/html",@"text/plain",@"text/xml"]];
    instance.responseSerializer.acceptableContentTypes = acceptableContentTypes;

});
return instance;
}

#pragma mark - 自定义GET

- (void)requestGET:(NSString *)URLString parames:(id)parames success:(void (^)(id responseObj))success failure:(void (^)(id error))failure{
    //AFN没有做UTF8转码 防止URL字符串中含有中文或特殊字符发生崩溃
    urlString = [[NSString stringWithFormat:@"%s",nstrPublicUrl,urlString] stringByAddingPercentEscapesUsingEncoding:NSUTF8StringEncoding];
    [self GET:urlString parameters:parames success:^(NSURLSessionDataTask * _Nonnull task, id _Nonnull responseObject) {
        success(responseObject);
    } failure:^(NSURLSessionDataTask * _Nullable task, NSError * _Nonnull error) {
        failure(error);
    }];
}

#pragma mark - 自定义POST

- (void)requestPOST:(NSString *)URLString parames:(id)parames success:(void (^)(id responseObj))success failure:(void (^)(id error))failure{
    //AFN没有做UTF8转码 防止URL字符串中含有中文或特殊字符 发生崩溃
    urlString = [[NSString stringWithFormat:@"%s",nstrPublicUrl,urlString] stringByAddingPercentEscapesUsingEncoding:NSUTF8StringEncoding];
    [self POST:urlString parameters:parames success:^(NSURLSessionDataTask * _Nonnull task, id _Nonnull responseObject) {

        ///将接收回来的数据转成UTF8的字符串，然后取出格式占位符 加上个转义符后才能让数据进行转换 否则转换失败
        NSString*jsonString = [[NSString alloc] initWithBytes:[responseObject bytes]length:[responseObject length]encoding:NSUTF8StringEncoding];
        jsonString = [jsonString stringByReplacingOccurrencesOfString:@"\t" withString:@"\\t"];
        NSData * jsonData = [jsonString dataUsingEncoding:NSUTF8StringEncoding];

        success(responseObject);
    } failure:^(NSURLSessionDataTask * _Nullable task, NSError * _Nonnull error) {
        failure(error);
    }];
}

#pragma mark - 上传数据

- (void)uploadWithURLString:(NSString *)URLString parameters:(id)parameters uploadParam:(NSArray<WKUploadParam *> *)uploadParams success:(void (^)(void))success failure:(void (^)(NSError *))failure {
    urlString = [NSString stringWithFormat:@"%s",nstrPublicUrl,urlString];
    [self POST:urlString parameters:parameters constructingBodyWithBlock:^(id<AFMultipartFormData> _Nonnull formData) {
        for (WKUploadParam *uploadParam in uploadParams) {
            [formData appendPartWithFileData:uploadParam.data name:uploadParam.name fileName:uploadParam.filename mimeType:uploadParam.mimeType];
        }
    } success:^(NSURLSessionDataTask * _Nonnull task, id _Nonnull responseObject) {
        if (success) {
            success(responseObject);
        }
    } failure:^(NSURLSessionDataTask * _Nullable task, NSError * _Nonnull error) {
        failure(error);
    }];
}
```

```
    } failure:^(NSURLSessionDataTask * _Nullable task, NSError * _Nonnull error) {
        if (failure) {
            failure(error);
        }
    }];
}

#pragma mark - 下载数据

- (void)downloadWithURLString:(NSString *)URLString parameters:(id)parameters
progerss:(void (^)(void))progress success:(void (^)(NSURLSessionDataTask * _Nonnull, NSError * _Nonnull))failure {
    NSString urlString = [NSString stringWithFormat:@"%s", nstrPublicUrl, URLString];
    NSURLRequest *request = [NSURLRequest requestWithURL:[NSURL URLWithString:urlString]];
    NSURLSessionDownloadTask *downloadTask = [self downloadTaskWithRequest:request progress:nil destination:^(NSURL * _Nonnull, NSURL * _Nonnull targetPath, NSURLResponse * _Nonnull response) {
        return targetPath;
    } completionHandler:^(NSURLResponse * _Nonnull response, NSURL * _Nullable filePath, NSError * _Nullable error) {
        if (failure) {
            failure(error);
        }
    }];
    [downloadTask resume];
}

@end
```

• **上传参数模型 WKUploadParam.h**

```
#import <Foundation/Foundation.h>

@interface WKUploadParam : NSObject
/**
 * 图片的二进制数据
 */
@property (nonatomic, strong) NSData *data;
/**
 * 服务器对应的参数名称
 */
@property (nonatomic, copy) NSString *name;
/**
 * 文件的名称(上传到服务器后, 服务器保存的文件名)
 */
@property (nonatomic, copy) NSString *filename;
/**
 * 文件的MIME类型(image/png, image/jpg等)
 */
@property (nonatomic, copy) NSString *mimeType;
@end
```



laitys (/u/45c32cefd5b9)
写了 11466 字, 被 28 人关注, 获得了 31 个喜欢
(/u/45c32cefd5b9)

+ 关注

如果觉得我的文章对您有用, 请随意打赏。您的支持将鼓励我继续创作!

赞赏支持

🤍 喜欢 | 3



更多分享



写下你的评论...

1条评论

只看作者

按喜欢排序 按时间正序 按时间倒序



179b0c11a717 (/u/179b0c11a717)

2楼 · 2017.03.28 19:22

(/u/179b0c11a717)

写的很好,对我太有帮助了,谢谢楼主



赞



回复

被以下专题收入，发现更多相似内容

