iOS网络请求之上传图片：从示例到源码解析 -- 以上传Face++SDK回调的图片为例（HYNetworking，AFNetworking，XMCenter）

前言

- 网络框架

本文一开始上传图片以调用HYNetworking的API为例，这个网络框架是以AFNetworking为基础进行的封装。HYNetworking内部实现上传图片的时候，其实就是采用AFNetworking关于上传图片的API，都是AFNetworking里面一个API。后面再讲XMCenter上传图片请求的操作方法，它也是基于AFNetworking上传进行的封装，不过比HYNetworking更加隐晦而已。

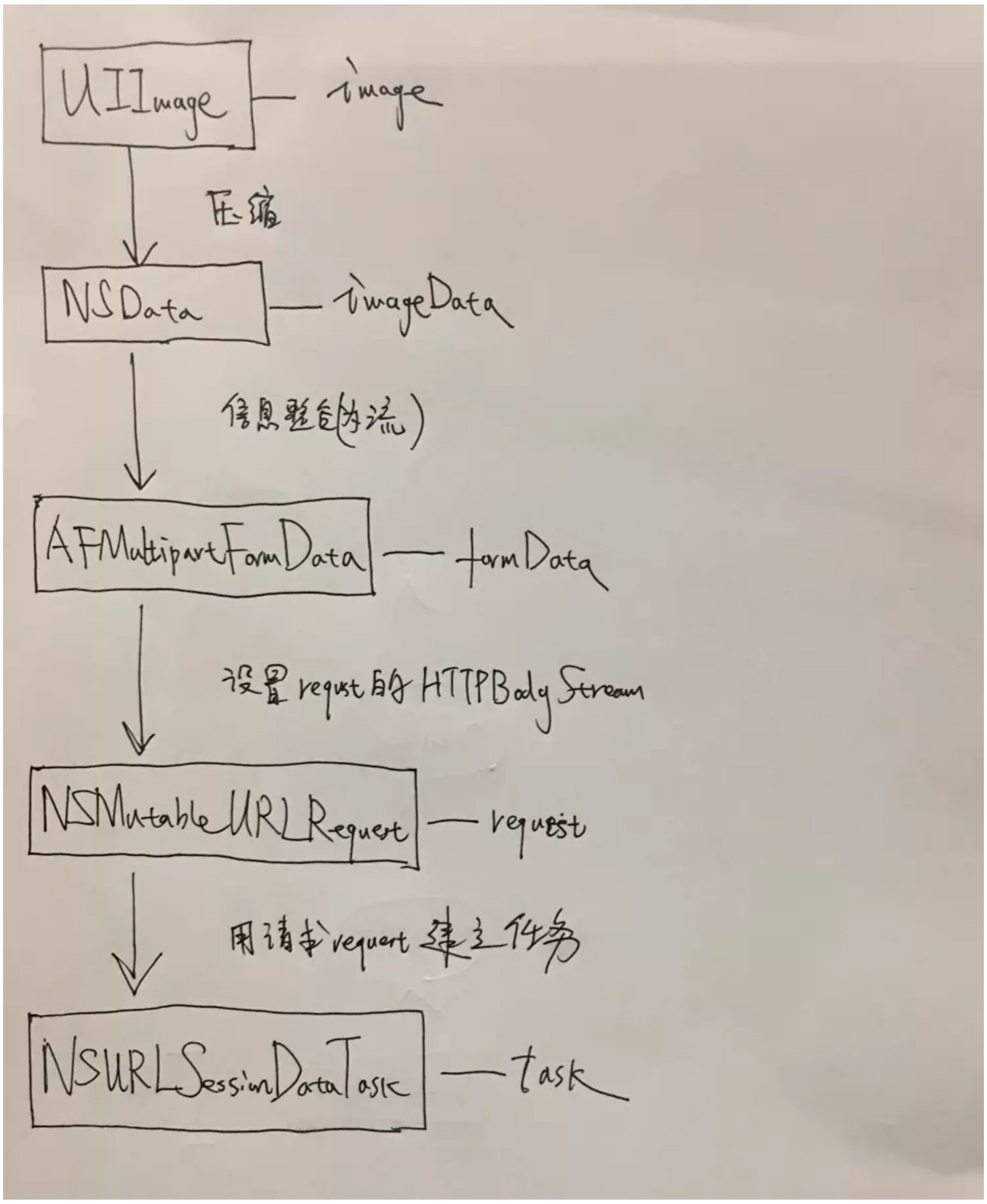
- 需求背景

这里的需求背景是，我们的app采用全球领先的AI方案提供商 -- 旷视科技 的Face++ SDK进行身份证识别：它识别到身份证后会回调一个图片数据，我们用此图片向Face++公司的服务器请求验证，该请求通过则block回调成功，接着将图片数据保存到手机本地，然后在合适的时机（比如，点击“完成”或者“下一步”按钮）把图片数据上传到自己公司的服务器。

- 先上总结

上传图片的流程图如下所示

- 点击"扫描身份证按钮"事件处理


UIImage — image
压缩
NSData — imageData
信息整合(为流)
AFMultipartFormData — formData
设置request的HTTPBodyStream
NSMutableURLRequest — request
用请求request建立任务
NSURLSessionDataTask — task

1. 获取回调图片

下面举例一个典型Face++的例子

- 点击"扫描身份证按钮"事件处理

```
// 身份证扫描正面
```

```
__weak typeof(self) weakSelf = self;
BOOL idcard = [MGIDCardManager getLicense];

if (!idcard) {
    [[[UIAlertView alloc] initWithTitle:@"提示" message:@"SDK授权失
败，请检查" delegate:self cancelButtonTitle:@"完成"
otherButtonTitles:nil, nil] show];
    return;
}

if ([CommonUtils isAllowedCamera] == NO) {

    //无权限
    UIAlertController *AlertController = [UIAlertController
alertControllerWithTitle:@"“小满APP”想访问您的相机" message:@"需要您的同
意，才能访问相机" preferredStyle:UIAlertControllerStyleAlert];

    [AlertController addAction:[UIAlertAction actionWithTitle:@"去设
置" style:UIAlertActionStyleDefault handler:^(UIAlertAction *
_Nonnull action) {

        [[UIApplication sharedApplication] openURL:[NSURL
URLWithString:UIApplicationOpenSettingsURLString]];
    }]];
    [AlertController addAction:[UIAlertAction actionWithTitle:@"取
消" style:UIAlertActionStyleCancel handler:^(UIAlertAction *
_Nonnull action) {

    }]];
    //弹出提示框;
    [self presentViewController:AlertController animated:YES
completion:nil];

    return;
}

MGIDCardManager *cardManager = [[MGIDCardManager alloc] init];

[cardManager IDCardStartDetection:weakSelf
IdCardSide:IDCARD_SIDE_FRONT
                           finish:^(MGIDCardModel *model) {

                               [_cardInfoVC sendFaceIDCardRequest:
[model croppedImageOfIDCard]];
                           } errr:^(MGIDCardError) {

                           }];
```
复制代码

```
复制代码
```

- 其中，`croppedImageOfIDCard`是为了从回调的model中取出图片：
- Face++SDK中的MGIDCardModel.mm

```
#pragma mark - Return UIImage
- (UIImage *)croppedImageOfIDCard {
#if TARGET_IPHONE_SIMULATOR
    return nil;
#else
    return [self.result croppedImageOfIDCard];
#endif
}
复制代码
```

- FaceSDK本地识别成功后，携带image向Face后台请求代码（主要检验身份证合法性等等），请求成功之后我们才将身份证图片保存到本地：

```
- (void)sendFaceIDCardRequest:(UIImage *)image
{
    [KVNProgress
showWithParameters:@{KVNProgressViewParameterStatus: @"加载中...",

KVNProgressViewParameterBackgroundType:
@(KVNProgressBackgroundTypeSolid),

KVNProgressViewParameterFullScreen: @(NO)}];

    UIImage *cardImage = image;
    NSMutableDictionary *params = [NSMutableDictionary dictionary];
    [params setObject:api_key forKey:@"api_key"];
    [params setObject:api_secret forKey:@"api_secret"];
    [params setObject:@"1" forKey:@"legality"];

    __weak typeof(self) weakSelf = self;
    [HYBNetworking uploadWithImage:cardImage url:QueryCardId
filename:@"image"
    name:@"image" mimeType:@"image/jpeg" parameters:params
progress:^(int64_t bytesWritten, int64_t totalBytesWritten) {}
    success:^(id response) {

        [KVNProgress dismiss];
        NSDictionary *dic = response;
```

```
          NSLog(@"%@",[dic mj_JSONString]);

          NSDictionary *legalityDict = [dic
objectForKey:@"legality"];
          NSString *side = [dic objectForKey:@"side"];
          if ([side isEqualToString:@"back"]) {

              // 发证机关
              self.cell.CREDITISSUEQRG = [dic
objectForKey:@"issued_by"];
              // valid_date 证件到期日
              self.cell.CREDITCERTENDTIME  = [dic
objectForKey:@"valid_date"];
              [weakSelf saveImage:image
withName:@"reverseCardImage.png"];

              weakSelf.cell.reverseCardImage = image;
              [weakSelf.cell setClipReverseCardImage:image];
              [weakSelf.cell
setReverseCardImageDataInfo:legalityDict];

          }
          else if ([side isEqualToString:@"front"]) {

              // 姓名
              weakSelf.cell.nameTextField.text = [dic
objectForKey:@"name"];
              // 性别
              //NSString *SEX = [dic objectForKey:@"race"];
              // 证件号码
              weakSelf.cell.IDCardTextField.text = [dic
objectForKey:@"id_card_number"];
              // 户籍所在地
              weakSelf.cell.HOUSEHOLDADDR = [dic
objectForKey:@"address"];

              NSString *CERTID = [CommonUtils
getValueInUDWithKey:kCERTID];
              if (![CommonUtils isStringNilOrEmpty:CERTID]) {
                  if (![CERTID isEqualToString:[dic
objectForKey:@"id_card_number"]]) {
                      [Toast showBottomWithText:@"您使用的身份证与您实名的
身份证不一致"];
                      return;
                  }
              }
              // 更新UI
              weakSelf.cell.positiveIDCardImage = image;//加载图片
```

```
            [weakSelf.cell setClipPositiveIDCardImage:image];
            [weakSelf.cell
setPositiveIDCardImageDataInfo:legalityDict];

            // 出生年月
//            NSDictionary *dict = [dic objectForKey:@"birthday"];
//            NSString *day = [dict objectForKey:@"day"];
//            NSString *month = [dict objectForKey:@"month"];
//            NSString *year = [dict objectForKey:@"year"];

            [weakSelf saveImage:image
withName:@"positiveIDCardImage.png"];
        }

    } fail:^(NSError *error) {
        [KVNProgress dismiss];
        [Toast showBottomWithText:kNetworkErrorMsg];
    }];
}
```
复制代码

- 最后面有个`saveImage: withName:`就是向Face++请求成功之后，进行保存到本地操作。

```
#pragma mark - 保存图片至沙盒

- (void)saveImage:(UIImage *)currentImage withName:(NSString
*)imageName
{
    if ([imageName isEqualToString:@"positiveIDCardImage.png"]) {
        NSString *imgPath = [CommonUtils
pathFileDocDir:@"/BorrowProcess/positiveIDCardImage.png"];
        if ([CommonUtils fileExistAtPath:imgPath]) {
            [CommonUtils fileDel:imgPath];
        }
    }
    if ([imageName isEqualToString:@"reverseCardImage.png"]) {
        NSString *imgPath1 = [CommonUtils
pathFileDocDir:@"/BorrowProcess/reverseCardImage.png"];
        if ([CommonUtils fileExistAtPath:imgPath1]) {
            [CommonUtils fileDel:imgPath1];
        }
    }

    float kCompressionQuality = 0.5;
    NSData *imageData = UIImageJPEGRepresentation(currentImage,
kCompressionQuality);  //将图片压缩

    NSString *fullPath = [[NSHomeDirectory()
stringByAppendingPathComponent:@"Documents/BorrowProcess/"]
stringByAppendingPathComponent:imageName];  //获取沙盒目录

    //创建文件夹
    [self createFile];
    //保存图片
    BOOL _isWriteToFile = [imageData writeToFile:fullPath
atomically:YES];  //将图片写入文件
}
复制代码
```

2. 上传回调图片

通过上面的保存操作，现在我们的APP到了点击下一步的情形，这时候需要我们向自己的后台（不是Face++的后台）上传图片了。

2.1 调用HYBNetworking的上传图片API

> 上传图片API

```
+ (HYBURLSessionTask *)uploadWithImage:(UIImage *)image
                                    url:(NSString *)url
                               filename:(NSString *)filename
                                   name:(NSString *)name
                               mimeType:(NSString *)mimeType
                             parameters:(NSDictionary *)parameters
                               progress:(HYBUploadProgress)progress
                                success:(HYBResponseSuccess)success
                                   fail:(HYBResponseFail)fail
```
复制代码

调用示例

```
#pragma mark - 上传身份证正面
//请求接口
-(void)httpRequestFRONT{
    NSString *LOANAPPLICATIONNO = nil;
    NSMutableArray *dataArray = [self seekPlist];
    if (dataArray) {
        NSDictionary *dataDict = nil;
        for (NSDictionary *dict in dataArray) {
            NSInteger index = [[dict objectForKey:@"index"]
integerValue];
            if (index == 0) {
                if (dict) {
                    dataDict = dict;
                }
            }
        }

        NSDictionary *dd = [dataDict
objectForKey:@"BorrowProcess"];
        LOANAPPLICATIONNO = [dd objectForKey:@"LOANAPPLICATIONNO"];
    }
    if ([CommonUtils isStringNilOrEmpty:LOANAPPLICATIONNO]) {
        LOANAPPLICATIONNO = @"";
    }

    NSArray *paths =
NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,
NSUserDomainMask, YES);
    NSString *path = [paths objectAtIndex:0];
    NSString *fileName = @"positiveIDCardImage";
    NSString *filePath = [NSString
stringWithFormat:@"%@/BorrowProcess/%@.png", path, fileName];
```

```
    UIImage *image;
    if ([self isFileExist:fileName] == YES)
    {
        //显示本地缓存
        image = [UIImage imageWithContentsOfFile:filePath];

    }
    [CommonUtils saveStrValueInUD:@"status016" forKey:kTRANSCODE];

    [KVNProgress
showWithParameters:@{KVNProgressViewParameterStatus: @"加载中...",

KVNProgressViewParameterBackgroundType:
@(KVNProgressBackgroundTypeSolid),

KVNProgressViewParameterFullScreen: @(NO)}];
    NSMutableDictionary *params = [NSMutableDictionary dictionary];

    [params setObject:[CommonUtils getStrValueInUDWithKey:kTOKEN]
forKey:@"TOKEN"];
    // 用户账号
    [params setObject:[CommonUtils getStrValueInUDWithKey:kUSERNO]
forKey:@"USERNO"];

    //    NSArray *BUSINESSTYPE = @[];//文档编号
    //    NSArray *BUSINESSNO = @[];//业务编号
    //    NSArray *DOCTYPE = @[];//文档类型
    [params setObject:@"LOAN_APPLY" forKey:@"BUSINESSTYPE"];
    [params setObject:LOANAPPLICATIONNO forKey:@"BUSINESSNO"];
    [params setObject:@"IDCARD_FRONT_APP" forKey:@"DOCTYPE"];

    __weak typeof(self) weakSelf = self;

    //上传图片
    [HYBNetworking uploadWithImage:image url:uploadImage
filename:@"IDFrontImage.jpg" name:@"FILE" mimeType:@"image/jpeg"
parameters:params progress:^(int64_t bytesWritten, int64_t
totalBytesWritten) {
        //获取进度: bytesWritten/totalBytesWritten

    } success:^(id response) {

        [KVNProgress dismiss];

        typeof(weakSelf) strongSelf = weakSelf;
        NSString *stringData = [response mj_JSONString];
        stringData = [DES3Util decrypt:stringData];
        NSLog(@"stirngData: %@", stringData);
        NSDictionary *responDict = [stringData mj_JSONObject];
        NSString *RETCODE = [responDict objectForKey:@"RETCODE"];
        if ([RETCODE isEqualToString:kSUCCESS]) {

            [strongSelf httpRequestBACK];
        }else if([RETCODE isEqualToString:kROKEN]){
            NSDictionary *processdict = @{@"index":[NSNumber
numberWithInteger:10], @"showProcessView":[NSNumber
numberWithBool:YES]};
            [[NSNotificationCenter defaultCenter]
postNotificationName:kBorrowProcessNotication object:nil
userInfo:processdict];
        }else{
            NSString *RETMSG = [responDict objectForKey:@"RETMSG"];
            if (![CommonUtils isStringNilOrEmpty:RETMSG]) {
                [Toast showBottomWithText:RETMSG];
            }
        }
    } fail:^(NSError *error) {
        [KVNProgress dismiss];
        [Toast showBottomWithText:kNetworkErrorMsg];
    }];
}
复制代码
```

2.2 调用AFNetwork整合图片的API -- 暨HYBNetworing上传图片封装源码解析

> 整合图片API

```
- (void)appendPartWithFileData:(NSData *)data
                          name:(NSString *)name
                      fileName:(NSString *)fileName
                      mimeType:(NSString *)mimeType
复制代码
```

> 调用示例 -- 上述2.1节中HYBNetworking的上传图片其实是调用AFNetworking的上传图片API。所以，HYBNetworking框架中上传图片的源码实现就是调用AFNetworking上传图片API的一个示例：

- HYBNetworking.m

```
+ (HYBURLSessionTask *)uploadWithImage:(UIImage *)image
                                   url:(NSString *)url

                              filename:(NSString *)filename
                                  name:(NSString *)name
                              mimeType:(NSString *)mimeType
                            parameters:(NSDictionary *)parameters
                              progress:(HYBUploadProgress)progress
                               success:(HYBResponseSuccess)success
                                  fail:(HYBResponseFail)fail {
  if ([self baseUrl] == nil) {
    if ([NSURL URLWithString:url] == nil) {
      HYBAppLog(@"URLString无效，无法生成URL。可能是URL中有中文，请尝试
Encode URL");
      return nil;
    }
  } else {
    if ([NSURL URLWithString:[NSString stringWithFormat:@"%@%@",
[self baseUrl], url]] == nil) {
      HYBAppLog(@"URLString无效，无法生成URL。可能是URL中有中文，请尝试
Encode URL");
      return nil;
    }
  }

  if ([self shouldEncode]) {
    url = [self encodeUrl:url];
  }

  NSString *absolute = [self absoluteUrlWithPath:url];

  AFHTTPSessionManager *manager = [self manager];
  HYBURLSessionTask *session = [manager POST:url
parameters:parameters
constructingBodyWithBlock:^(id<AFMultipartFormData>  _Nonnull
formData) {
    NSData *imageData = UIImageJPEGRepresentation(image, 1);

    NSString *imageFileName = filename;
    if (filename == nil || ![filename isKindOfClass:[NSString
class]] || filename.length == 0) {
      NSDateFormatter *formatter = [[NSDateFormatter alloc] init];
      formatter.dateFormat = @"yyyyMMddHHmmss";
      NSString *str = [formatter stringFromDate:[NSDate date]];
      imageFileName = [NSString stringWithFormat:@"%@.jpg", str];
    }

    // 整合图片，以文件流的格式
    [formData appendPartWithFileData:imageData name:name
fileName:imageFileName mimeType:mimeType];


  } progress:^(NSProgress * _Nonnull uploadProgress) {
    if (progress) {
      progress(uploadProgress.completedUnitCount,
uploadProgress.totalUnitCount);
    }
  } success:^(NSURLSessionDataTask * _Nonnull task, id  _Nullable
responseObject) {
    [[self allTasks] removeObject:task];
    [self successResponse:responseObject callback:success];

    if ([self isDebug]) {
      [self logWithSuccessResponse:responseObject
                               url:absolute
                            params:parameters];
    }
  } failure:^(NSURLSessionDataTask * _Nullable task, NSError *
_Nonnull error) {
    [[self allTasks] removeObject:task];

    [self handleCallbackWithError:error fail:fail];

    if ([self isDebug]) {
      [self logWithFailError:error url:absolute params:nil];
    }
  }];

  [session resume];
  if (session) {
    [[self allTasks] addObject:session];
  }

  return session;
}
```
复制代码

可见，整合图片的一句关键代码在

```
// 整合图片，以文件流的格式
[formData appendPartWithFileData:imageData name:name
fileName:imageFileName mimeType:mimeType];
```
复制代码

这里是设置图片的数据流，作为AFNetwork的POST请求方法的一个 `constructingBodyWithBlock` 参数的输入。

> AFNetwork的POST请求方法源码

- AFHTTPSessionManager.m

```
- (NSURLSessionDataTask *)POST:(NSString *)URLString
                    parameters:(id)parameters
     constructingBodyWithBlock:(void (^)(id <AFMultipartFormData>
formData))block
                      progress:(nullable void (^)(NSProgress *
_Nonnull))uploadProgress
                       success:(void (^)(NSURLSessionDataTask
*task, id responseObject))success
                       failure:(void (^)(NSURLSessionDataTask
*task, NSError *error))failure
{
    NSError *serializationError = nil;
    NSMutableURLRequest *request = [self.requestSerializer
multipartFormRequestWithMethod:@"POST" URLString:[[NSURL
URLWithString:URLString relativeToURL:self.baseURL] absoluteString]
parameters:parameters constructingBodyWithBlock:block
error:&serializationError];
    if (serializationError) {
        if (failure) {
            dispatch_async(self.completionQueue ?:
dispatch_get_main_queue(), ^{
                failure(nil, serializationError);
            });
        }

        return nil;
    }

    __block NSURLSessionDataTask *task = [self
uploadTaskWithStreamedRequest:request progress:uploadProgress
completionHandler:^(NSURLResponse * __unused response, id
responseObject, NSError *error) {
        if (error) {
            if (failure) {
                failure(task, error);
            }
        } else {
            if (success) {
                success(task, responseObject);
            }
        }
    }];

    [task resume];

    return task;
}
复制代码
```

> 这个方法将block传递给下一个API，并返回一个request：

- AFURLRequestSerialization.m

```
- (NSMutableURLRequest *)multipartFormRequestWithMethod:(NSString
*)method
                                              URLString:(NSString
*)URLString
                                             parameters:
(NSDictionary *)parameters
                              constructingBodyWithBlock:(void (^)
(id <AFMultipartFormData> formData))block
                                                  error:(NSError
*__autoreleasing *)error
{
    NSParameterAssert(method);
    NSParameterAssert(![method isEqualToString:@"GET"] && ![method
isEqualToString:@"HEAD"]);

    NSMutableURLRequest *mutableRequest = [self
requestWithMethod:method URLString:URLString parameters:nil
error:error];

    __block AFStreamingMultipartFormData *formData =
[[AFStreamingMultipartFormData alloc]
initWithURLRequest:mutableRequest
stringEncoding:NSUTF8StringEncoding];

    if (parameters) {
        for (AFQueryStringPair *pair in
AFQueryStringPairsFromDictionary(parameters)) {
            NSData *data = nil;
            if ([pair.value isKindOfClass:[NSData class]]) {
                data = pair.value;
            } else if ([pair.value isEqual:[NSNull null]]) {
                data = [NSData data];
            } else {
            } else {
                data = [[pair.value description]
dataUsingEncoding:self.stringEncoding];
            }

            if (data) {
                [formData appendPartWithFormData:data name:
[pair.field description]];
            }
        }
    }

    if (block) {
        block(formData);
    }

    return [formData requestByFinalizingMultipartFormData];
}
复制代码
```

- 这个方法的关键在于最后，用block回调了formData：

```
if (block) {
        block(formData);
    }
复制代码
```

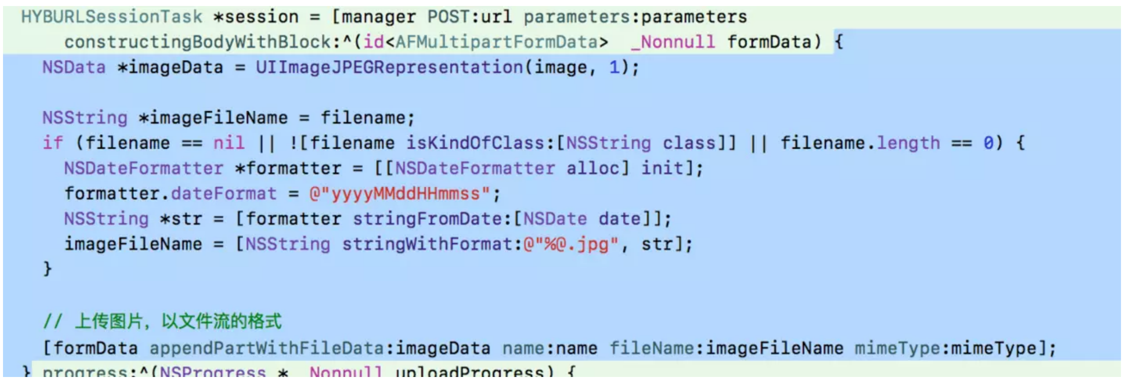
这个formData是一个AFStreamingMultipartFormData模型的对象

> 图片模型 -- AFStreamingMultipartFormData

- AFURLRequestSerialization.m

```
@interface AFStreamingMultipartFormData ()
@property (readwrite, nonatomic, copy) NSMutableURLRequest
*request;
@property (readwrite, nonatomic, assign) NSStringEncoding
stringEncoding;
@property (readwrite, nonatomic, copy) NSString *boundary;
@property (readwrite, nonatomic, strong) AFMultipartBodyStream
*bodyStream;
@end
复制代码
```

这样，就执行了前面调用时block里面的图片数据整合操作（[formData appendPartWithFileData...];）。



```
HYBURLSessionTask *session = [manager POST:url parameters:parameters
    constructingBodyWithBlock:^(id<AFMultipartFormData>  _Nonnull formData) {
  NSData *imageData = UIImageJPEGRepresentation(image, 1);

  NSString *imageFileName = filename;
  if (filename == nil || ![filename isKindOfClass:[NSString class]] || filename.length == 0) {
    NSDateFormatter *formatter = [[NSDateFormatter alloc] init];
    formatter.dateFormat = @"yyyyMMddHHmmss";
    NSString *str = [formatter stringFromDate:[NSDate date]];
    imageFileName = [NSString stringWithFormat:@"%@.jpg", str];
  }

  // 上传图片，以文件流的格式
  [formData appendPartWithFileData:imageData name:name fileName:imageFileName mimeType:mimeType];
} progress:^(NSProgress *  _Nonnull uploadProgress) {
```

> 在`multipartFormRequestWithMethod:...`的实现代码中，接着，利用block体中设置好的formData，调用下述的`requestByFinalizingMultipartFormData`方法以返回一个request，然后进行请求。`requestByFinalizingMultipartFormData`的代码如下：

- AFURLRequestSerialization.m

```
- (NSMutableURLRequest *)requestByFinalizingMultipartFormData {
    if ([self.bodyStream isEmpty]) {
        return self.request;
    }

    // Reset the initial and final boundaries to ensure correct
Content-Length
    [self.bodyStream setInitialAndFinalBoundaries];
    [self.request setHTTPBodyStream:self.bodyStream];

    [self.request setValue:[NSString
stringWithFormat:@"multipart/form-data; boundary=%@",
self.boundary] forHTTPHeaderField:@"Content-Type"];
    [self.request setValue:[NSString stringWithFormat:@"%llu",
[self.bodyStream contentLength]] forHTTPHeaderField:@"Content-
Length"];

    return self.request;
}
复制代码
```

上面代码的关键在于将图片数据整合进request，设置HTTPBodyStream，即`[self.request setHTTPBodyStream:self.bodyStream];`。

- 获取上述request之后，如前面所述“ AFNetwork的POST请求方法源码”，调用返回request的API之后，再调用POST请求方法进行请求操作，即 `__block NSURLSessionDataTask *task = [self uploadTaskWithStreamedRequest:request progress:uploadProgress completionHandler:^(NSURLResponse * __unused response, id responseObject, NSError *error) {` 这一句。

> 获取带有图片request的API

```
- (NSMutableURLRequest *)multipartFormRequestWithMethod:(NSString *)method
                                              URLString:(NSString *)URLString
                                             parameters:(NSDictionary *)parameters
                              constructingBodyWithBlock:(void (^)(id <AFMultipartFormData> formData))block
                                                  error:(NSError *__autoreleasing *)error
复制代码
```

> 上传带有图片request的API

```
- (NSURLSessionUploadTask *)uploadTaskWithStreamedRequest:(NSURLRequest *)request
                                                 progress:(void (^)(NSProgress *uploadProgress)) uploadProgressBlock
                                        completionHandler:(void (^)(NSURLResponse *response, id responseObject, NSError *error))completionHandler
复制代码
```

2.3 源码分析 -- AFNetwork整合图片数据调用栈解析

- 将描述图片的参数字符串转化头字典

```
- (void)appendPartWithFileData:(NSData *)data

                         name:(NSString *)name
                     fileName:(NSString *)fileName
                     mimeType:(NSString *)mimeType
{
    NSParameterAssert(name);
    NSParameterAssert(fileName);
    NSParameterAssert(mimeType);

    NSMutableDictionary *mutableHeaders = [NSMutableDictionary
dictionary];
    [mutableHeaders setValue:[NSString stringWithFormat:@"form-
data; name=\"%@\"; filename=\"%@\"", name, fileName]
forKey:@"Content-Disposition"];
    [mutableHeaders setValue:mimeType forKey:@"Content-Type"];

    [self appendPartWithHeaders:mutableHeaders body:data];
}
复制代码
```

- 为图片数据添加头字典 -- 数据整合

```
- (void)appendPartWithHeaders:(NSDictionary *)headers
                         body:(NSData *)body
{
    NSParameterAssert(body);

    AFHTTPBodyPart *bodyPart = [[AFHTTPBodyPart alloc] init];
    bodyPart.stringEncoding = self.stringEncoding;
    bodyPart.headers = headers;
    bodyPart.boundary = self.boundary;
    bodyPart.bodyContentLength = [body length];
    bodyPart.body = body;

    [self.bodyStream appendHTTPBodyPart:bodyPart];
}
复制代码
```

其中，数据流 `self.bodyStream` 是这样定义的：

```
@property (readwrite, nonatomic, strong) AFMultipartBodyStream
*bodyStream;
复制代码
```

其中，`AFMultipartBodyStream`是这样定义的：

```
@interface AFMultipartBodyStream : NSInputStream <NSStreamDelegate>
复制代码
```

- 再来看看怎样向数据流添加整合好的图片数据的：

```
- (void)appendHTTPBodyPart:(AFHTTPBodyPart *)bodyPart {
    [self.HTTPBodyParts addObject:bodyPart];
}
复制代码
```

- 其中，`HTTPBodyParts`是这样定义的：

```
@property (readwrite, nonatomic, strong) NSMutableArray
*HTTPBodyParts;
复制代码
```

- 另外，`AFHTTPBodyPart`的定义是这样的：

```
@interface AFHTTPBodyPart : NSObject
@property (nonatomic, assign) NSStringEncoding stringEncoding;
@property (nonatomic, strong) NSDictionary *headers;
@property (nonatomic, copy) NSString *boundary;
@property (nonatomic, strong) id body;
@property (nonatomic, assign) unsigned long long bodyContentLength;
@property (nonatomic, strong) NSInputStream *inputStream;

@property (nonatomic, assign) BOOL hasInitialBoundary;
@property (nonatomic, assign) BOOL hasFinalBoundary;

@property (readonly, nonatomic, assign, getter = hasBytesAvailable)
BOOL bytesAvailable;
@property (readonly, nonatomic, assign) unsigned long long
contentLength;
复制代码
```

3. XMCenter

XMCenter其实也是基于ANNetwork封装的，不过封装的层级比HYBNetwork多，看起来有点隐晦。这里分析一下XMCenter上传图片的API。

3.1 整合图片

- 整合图片API

```
- (void)addFormDataWithName:(NSString *)name fileName:(NSString
*)fileName mimeType:(NSString *)mimeType fileData:(NSData
*)fileData
复制代码
```

- 调用示例

```
- (NSString *_Nullable)uploadWithUploadImageModels:(nullable
NSArray<UploadImageModel *> *)imageModelArr params:(id _Nullable
)params paramsType:(ParamsType)paramsType userNo:(NSString *)userNo
APITag:(API_Tag)tag mimeType:(nullable NSString *)mimeType
       OnSuccess:(SuccessBlock)successBlock onFailure:
(FailureBlock)failureBlock
{
    NSString *identifier = [XMCenter sendRequest:^(XMRequest *
_Nonnull request) {

        request.api = [self pathURLWithAPITag:tag];

        if ([self
respondsToSelector:@selector(cys_requestOnlyLayerParameters:)]) {
            request.parameters = [self
cys_requestOnlyLayerParameters:params];
        }

        // 上传图片，以文件流的格式
        for (UploadImageModel *imageModel in imageModelArr) {
            NSData *imageData =
UIImageJPEGRepresentation(imageModel.image, 0.5);

            NSString *imageFileName = imageModel.fileName;
            if (imageModel.fileName == nil || ![imageModel.fileName
isKindOfClass:[NSString class]] || imageModel.fileName.length == 0)
{
                NSDateFormatter *formatter = [[NSDateFormatter
alloc] init];
                formatter.dateFormat = @"yyyyMMddHHmmss";
                NSString *str = [formatter stringFromDate:[NSDate
date]];
                imageFileName = [NSString
stringWithFormat:@"%@.jpg", str];
            }
            // 上传图片
            [request addFormDataWithName:imageModel.name
fileName:imageFileName mimeType:mimeType fileData:imageData];
        }

        NSLog(@"请求的参数：%@",request.parameters);
        if (imageModelArr.count > 0) {
            request.requestType = kXMRequestUpload;
        }
        request.requestSerializerType = kXMRequestSerializerJSON;
    } onSuccess:^(id  _Nullable responseObject) {
        // 省略...
复制代码
```

3.2 上传图片API源码解析

- 将整合好的图片数据formData添加到图片模型数组uploadFormDatas

```
- (void)addFormDataWithName:(NSString *)name fileName:(NSString

*)fileName mimeType:(NSString *)mimeType fileData:(NSData
*)fileData {

    XMUploadFormData *formData = [XMUploadFormData
formDataWithName:name fileName:fileName mimeType:mimeType
fileData:fileData];
    [self.uploadFormDatas addObject:formData];
}
复制代码
```

- 整合图片数据

```
+ (instancetype)formDataWithName:(NSString *)name fileName:
(NSString *)fileName mimeType:(NSString *)mimeType fileData:(NSData
*)fileData {
    XMUploadFormData *formData = [[XMUploadFormData alloc] init];
    formData.name = name;
    formData.fileName = fileName;
    formData.mimeType = mimeType;
    formData.fileData = fileData;
    return formData;
}
复制代码
```

- 图片模型数组

```
- (NSMutableArray<XMUploadFormData *> *)uploadFormDatas {
    if (!_uploadFormDatas) {
        _uploadFormDatas = [NSMutableArray array];
    }
    return _uploadFormDatas;
}
复制代码
```

- 图片模型 -- XMUploadFormData

```
/**
 `XMUploadFormData` is the class for describing and carring the
upload file data, see `AFMultipartFormData` protocol for details.
 */
@interface XMUploadFormData : NSObject

/**
 The name to be associated with the specified data. This property
must not be `nil`.
 */
@property (nonatomic, copy) NSString *name;

/**
 The file name to be used in the `Content-Disposition` header. This
property is not recommended be `nil`.
 */
@property (nonatomic, copy, nullable) NSString *fileName;

/**
 The declared MIME type of the file data. This property is not
recommended be `nil`.
 */
@property (nonatomic, copy, nullable) NSString *mimeType;

/**
 The data to be encoded and appended to the form data, and it is
prior than `fileURL`.
 */
@property (nonatomic, strong, nullable) NSData *fileData;

/**
 The URL corresponding to the file whose content will be appended
to the form, BUT, when the `fileData` is assigned, the `fileURL`
will be ignored.
 */
@property (nonatomic, strong, nullable) NSURL *fileURL;

// NOTE: Either of the `fileData` and `fileURL` should not be
`nil`, and the `fileName` and `mimeType` must both be `nil` or
assigned at the same time,
复制代码
```

- 遍历图片模型数组中的图片模型进行上传请求

```
- (void)xm_uploadTaskWithRequest:(XMRequest *)request
               completionHandler:
```

```
               completionHandler:
(XMCompletionHandler)completionHandler {

    AFHTTPSessionManager *sessionManager = [self
xm_getSessionManager:request];
    AFHTTPRequestSerializer *requestSerializer = [self
xm_getRequestSerializer:request];

    __block NSError *serializationError = nil;
    NSMutableURLRequest *urlRequest = [requestSerializer
multipartFormRequestWithMethod:@"POST"

URLString:request.url

parameters:request.parameters

constructingBodyWithBlock:^(id<AFMultipartFormData> formData) {
        [request.uploadFormDatas
enumerateObjectsUsingBlock:^(XMUploadFormData *obj, NSUInteger idx,
BOOL *stop) {
            if (obj.fileData) {
                if (obj.fileName && obj.mimeType) {
                    [formData appendPartWithFileData:obj.fileData
name:obj.name fileName:obj.fileName mimeType:obj.mimeType];
                } else {
                    [formData appendPartWithFormData:obj.fileData
name:obj.name];
                }
            } else if (obj.fileURL) {
                NSError *fileError = nil;
                if (obj.fileName && obj.mimeType) {
                    [formData appendPartWithFileURL:obj.fileURL
name:obj.name fileName:obj.fileName mimeType:obj.mimeType
error:&fileError];
                } else {
                    [formData appendPartWithFileURL:obj.fileURL
name:obj.name error:&fileError];
                }
                if (fileError) {
                    serializationError = fileError;
                    *stop = YES;
                }
            }
        }];
    } error:&serializationError];

    if (serializationError) {
        if (completionHandler) {
            dispatch_async(xm_request_completion_callback_queue(),
^{
                completionHandler(nil, serializationError);
            });
        }
        return;
    }

    [self xm_processURLRequest:urlRequest byXMRequest:request];

    NSURLSessionUploadTask *uploadTask = nil;
    __weak __typeof(self)weakSelf = self;
    uploadTask = [sessionManager
uploadTaskWithStreamedRequest:urlRequest

progress:request.progressBlock

completionHandler:^(NSURLResponse *response, id responseObject,
NSError *error) {
        __strong __typeof(weakSelf)strongSelf = weakSelf;
        [strongSelf xm_processResponse:response
                                object:responseObject
                                 error:error
                               request:request
                     completionHandler:completionHandler];
    }];

    [self xm_setIdentifierForReqeust:request
taskIdentifier:uploadTask.taskIdentifier
sessionManager:sessionManager];
    [uploadTask bindingRequest:request];
    [uploadTask resume];
}
```
复制代码

- 可见，XMCenter最后还是调用AFNetwork的POST请求及 `uploadTaskWithStreamedRequest` 方法进行上传。

```
- (NSMutableURLRequest *)multipartFormRequestWithMethod:(NSString
*)method
                                             URLString:(NSString
*)URLString
                                            parameters:
(NSDictionary *)parameters
                             constructingBodyWithBlock:(void (^)
(id <AFMultipartFormData> formData))block
                                                 error:(NSError
*__autoreleasing *)error
```
复制代码

- 同样可见，同HYBNetwork一样，调用AFNetwork的图片数据整合API `appendPartWithFileData:`进行设置

```
[formData appendPartWithFileData:obj.fileData name:obj.name
fileName:obj.fileName mimeType:obj.mimeType];
```
复制代码

4. 总结：上传图片逻辑整理

AFNetwork

- 压缩转换：UIImage实例对象通过UIImageJPEGRepresentation压缩转换为NSData，下面称之为imageData。
- 信息整合：将imageData与文件名fileName，文件路径name，类型名mimeType整合成图片模型（AFHTTPBodyPart）的一个对象bodyPart中去。
- 添加图片模型：将上面新建好的图片模型对象bodyPart，向图片输入流（AFMultipartBodyStream）的对象bodyStream的数组属性（HTTPBodyParts）添加。
- 设置requet的HTTPBodyStream属性为bodyStream：封装为 `requestByFinalizingMultipartFormData`
- 将图片模型对象formData用AFNetwork的POST请求与 `uploadTaskWithStreamedRequest` 方法进行上传。

HYBNetwork

- 压缩转换：UIImage实例对象通过UIImageJPEGRepresentation压缩转换为NSData，下面称之为imageData。
- 信息整合：利用AFNetwork的 `appendPartWithFileData` ，将imageData与文件名fileName，文件路径name，类型名mimeType整合成图片模型（AFStreamingMultipartFormData）的一个对象formData中去。
- 将图片模型对象formData用AFNetwork的POST请求与 `uploadTaskWithStreamedRequest` 方法进行上传。

XMCenter

- 压缩转换：UIImage实例对象通过UIImageJPEGRepresentation压缩转换为NSData，下面称之为imageData。
- 信息整合：利用AFNetwork的 `appendPartWithFileData` ，将imageData与文件名fileName，文件路径name，类型名mimeType整合成图片模型（XMUploadFormData）的一个对象formData中去。
- 添加图片模型：向管理器的图片模型数组uploadFormDatas添加上面新建好的图片模型对象formData。
- 遍历图片模型数组，获得图片模型，利用AFNetwork的POST请求与 `uploadTaskWithStreamedRequest` 方法进行上传。

5. 说明

本文示例只做了最简单的请求方式 -- 发起一次请求就调用一次。

其实，还有很多可以优化的点，例如，对所有request进行管理封装：建立一个请求的队列或者数组，相同请求不允许再添加，优先级低的请求先等待，异步请求的最大并发线程数，等等。这里只提醒，就不介绍了。