**各平台iOS接口调研报告**

**一、前言**

本文档属于毕业设计课题“针对iOS应用的多平台数据封装中间件实现”的前期调研报告一部分，主要目的是为多平台的数据抽象提供参考，并产出平台接口抽象类图，结合设计模式调研确定代码架构样式。

**二、各平台接口介绍**

*（一）微信平台接口*

1.源代码分析

微信开放平台提供了一个完整的范例工程ReleaseSample

文件结构如下,省略相关图片资源文件：

/ReleaseSample

/App

//iOS应用delegate，接受并响应应用生命周期事件

AppDelegate.h

AppDelegate.m

/Control

//Api组件，封装接口Api

WXApiManager.h

WXApiManager.m

WXApiRequestHandler.h

WXApiRequestHandler.m

WXApiResponseHander.h

WXApiResponseHander.m

/Helper

//定义常量、数据结构、数据get/set方法以及弹窗（AlertView）视图

Constant.h

GetMessageFromWXResp+responseWithTextOrMediaMessage.h

GetMessageFromWXResp+responseWithTextOrMediaMessage.m

SendMessageToWXReq+requestWithTextOrMediaMessage.h

SendMessageToWXReq+requestWithTextOrMediaMessage.m

UIAlertView+WX.h

UIAlertView+WX.m

WXMediaMessage+messageConstruct.h

WXMediaMessage+messageConstruct.m

/ViewController

//视图控制器，定义Sample的业务逻辑

RespForWeChatViewController.h

RespForWeChatViewController.mm

SendMsgToWeChatViewController.h

SendMsgToWeChatViewController.m

/SEKExport

//微信终端SDK文件

libWeChatSDK.a

WechatAuthSDK.h

WXApi.h

WXApiObject.h

SDKSample.xcodeproj //iOS应用工程文件

en.lproj //.xib文件，定义UI

main.m //启动方法

SDKSample-Info.plist

SDKSample-Prefix.pch

重点分析SEKExport目录下的WXApi.h（定义接口）和WXApiObject.h（定义数据结构）。WechatAuthSDK.h主要负责登录请求，不做分析。

//WXApiObject.h

首先定义BaseReq和BaseResp作为请求和响应的基类：

1. #pragma mark - BaseReq
2. /\*! @brief 该类为微信终端SDK所有请求类的基类
3. \*
4. \*/
5. @interface BaseReq : NSObject
6. /\*\* 请求类型 \*/
7. @property (nonatomic, assign) **int** type;
8. /\*\* 由用户微信号和AppID组成的唯一标识，发送请求时第三方程序必须填写，用于校验微信用户是否换号登录\*/
9. @property (nonatomic, copy) NSString \*openID;
10. @end
12. #pragma mark - BaseResp
13. /\*! @brief 该类为微信终端SDK所有响应类的基类
14. \*
15. \*/
16. @interface BaseResp : NSObject
17. /\*\* 错误码 \*/
18. @property (nonatomic, assign) **int** errCode;
19. /\*\* 错误提示字符串 \*/
20. @property (nonatomic, copy) NSString \*errStr;
21. /\*\* 响应类型 \*/
22. @property (nonatomic, assign) **int** type;
23. @end

然后继承Base类实现业务类。

根据实际应用情况分析：

SendMessageToWXReq 第三方程序发送消息至微信终端程序的消息结构体，信息类型包括文本和多媒体消息。

1. @interface SendMessageToWXReq : BaseReq
2. /\*\* 发送消息的文本内容
3. \* @note 文本长度必须大于0且小于10K
4. \*/
5. @property (nonatomic, copy) NSString \*text;
6. /\*\* 发送消息的多媒体内容
7. \* @see WXMediaMessage
8. \*/
9. @property (nonatomic, strong) WXMediaMessage \*message;
10. /\*\* 发送消息的类型，包括文本消息和多媒体消息两种，两者只能选择其一，不能同时发送文本和多媒体消息 \*/
11. @property (nonatomic, assign) **BOOL** bText;
12. /\*\* 发送的目标场景，可以选择发送到会话(WXSceneSession)或者朋友圈(WXSceneTimeline)。 默认发送到会话。
13. \* @see WXScene
14. \*/
15. @property (nonatomic, assign) **int** scene;
16. /\*\* 指定发送消息的人
17. \* @note WXSceneSpecifiedSession时有效
18. \*/
19. @property (nonatomic, copy, nullable) NSString \*toUserOpenId;
20. @end

对应有SendMessageToWXResp，表示微信发送回来的处理结果。

其中用到了表示多媒体消息的数据类WXMediaMessage，

用于微信终端和第三方程序之间传递消息的多媒体消息内容。

1. @interface WXMediaMessage : NSObject
2. + (WXMediaMessage \*)message;
3. /\*\* 标题
4. \* @note 长度不能超过512字节
5. \*/
6. @property (nonatomic, copy) NSString \*title;
7. /\*\* 描述内容
8. \* @note 长度不能超过1K
9. \*/
10. @property (nonatomic, copy) NSString \*description;
11. /\*\* 缩略图数据
12. \* @note 大小不能超过32K
13. \*/
14. @property (nonatomic, strong, nullable) NSData \*thumbData;
15. /\*\*
16. \* @note 长度不能超过64字节
17. \*/
18. @property (nonatomic, copy, nullable) NSString \*mediaTagName;
19. @property (nonatomic, copy, nullable) NSString \*messageExt;
20. @property (nonatomic, copy, nullable) NSString \*messageAction;
21. /\*\*
22. \* 多媒体数据对象，可以为WXImageObject，WXMusicObject，WXVideoObject，WXWebpageObject等。
23. \*/
24. @property (nonatomic, strong) id mediaObject;
25. /\*! @brief 设置消息缩略图的方法
26. \*
27. \* @param image 缩略图
28. \* @note 大小不能超过64K
29. \*/
30. - (**void**)setThumbImage:(UIImage \*)image;
31. @end

WXMediaMessage具有mediaObject属性作为不同多媒体属性载体。

//WXApi.h

WXApiDelegate提供了两个关键的代理方法onReq和onResp

1. @protocol WXApiDelegate <NSObject>
2. @optional
3. /\*! @brief 收到一个来自微信的请求，第三方应用程序处理完后调用sendResp向微信发送结果
4. \*
5. \* 收到一个来自微信的请求，异步处理完成后必须调用sendResp发送处理结果给微信。
6. \* 可能收到的请求有GetMessageFromWXReq、ShowMessageFromWXReq等。
7. \* @param req 具体请求内容，是自动释放的
8. \*/
9. - (**void**)onReq:(BaseReq\*)req;
10. /\*! @brief 发送一个sendReq后，收到微信的回应
11. \*
12. \* 收到一个来自微信的处理结果。调用一次sendReq后会收到onResp。
13. \* 可能收到的处理结果有SendMessageToWXResp、SendAuthResp等。
14. \* @param resp具体的回应内容，是自动释放的
15. \*/
16. - (**void**)onResp:(BaseResp\*)resp;
17. @end

WXApi提供了一系列业务方法。

根据文档和代码可以总结微信平台接口的一般业务流程为：

S1.检查是否安装微信

1. [WXApi isWXAppInstalled]

S2.向微信终端注册id

1. [WXApi registerApp:kWXAppID];

S3.初始化请求，设置发送的目标场景及数据类型

1. SendMessageToWXReq\* req = [[SendMessageToWXReq alloc] init];
2. req.scene = WXSceneSession;
3. req.message = something;

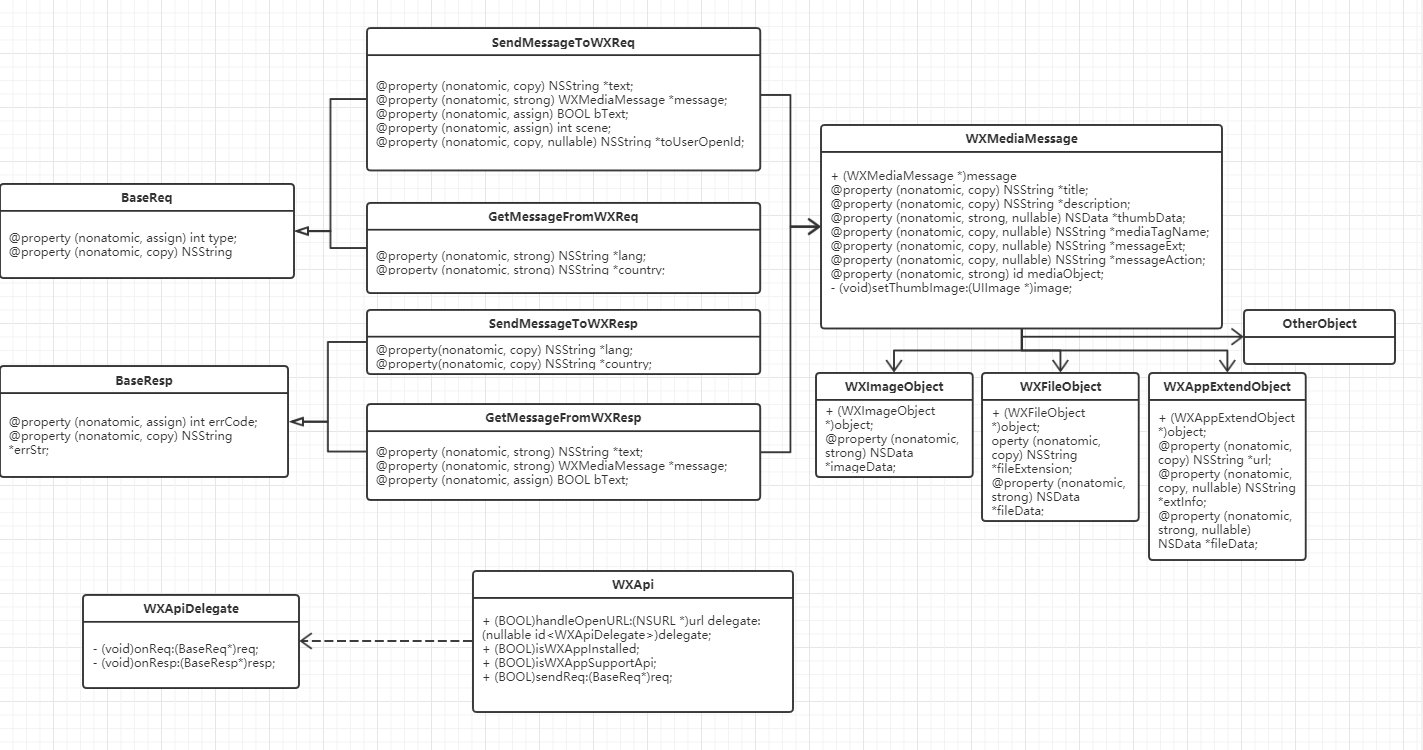
S4.发送请求

1. [WXApi sendReq:req];

S5.可选，通过设置代理并且实现代理方法接受微信的回应

1. [WXApi handleOpenURL:url delegate:delegate];

2.核心类图如下：



1. *新浪微博开放平台接口*
2. 源代码分析

新浪微博开放平台提供了一个完整的范例工程WeiboSDKDemo

文件结构类似微信范例工程ReleaseSample，同样省略资源文件：

/WeiboSDKDemo

//iOS应用delegate，接受并响应应用生命周期事件

AppDelegate.h

AppDelegate.m

//视图控制器，定义Sample的业务逻辑

LinkToWeiboViewController.h

LinkToWeiboViewController.m

SendMessageToWeiboViewController.h

SendMessageToWeiboViewController.m

/libWeiboSDK

//微博SDK文件

libWeiboSDK.a

WBHttpRequest.h

WeiboSDK.h

en.lproj //.xib文件，定义UI

main.m //启动方法

WeiboSDKDemo-Info.plist

WeiboSDKDemo-Prefix.pch

/WeiboSDKLibDemo.xcodeproj //iOS应用工程文件

同样主要分析libWeiboSDK下的定义的接口以及ViewController中的示例调用。

//WeiboSDK.h

WBDataTransferObject 微博客户端程序和第三方应用之间传输数据信息的基类

WBBaseRequest和WBBaseResponse继承WBDataTransferObject，是所有响应类和请求类的基类

1. @interface WBBaseRequest : WBDataTransferObject
2. /\*\*
3. 返回一个 WBBaseRequest 对象
4. @return 返回一个\*自动释放的\*WBBaseRequest对象
5. \*/
6. + (id)request;
7. @end
9. @interface WBBaseResponse : WBDataTransferObject
10. /\*\*
11. 对应的 request 中的自定义信息字典
12. 如果当前 response 是由微博客户端响应给第三方应用的，则 requestUserInfo 中会包含原 request.userInfo 中的所有数据
13. @see WBBaseRequest.userInfo
14. \*/
15. @property (strong, nonatomic, readonly) NSDictionary \*requestUserInfo;
16. /\*\*
17. 响应状态码
18. 第三方应用可以通过statusCode判断请求的处理结果
19. \*/
20. @property (nonatomic, assign) WeiboSDKResponseStatusCode statusCode;
21. /\*\*
22. 返回一个 WBBaseResponse 对象
23. @return 返回一个\*自动释放的\*WBBaseResponse对象
24. \*/
25. + (id)response;
26. @end

通过继承WBBaseRequest和WBBaseResponse实现了业务类。

常用的是WBSendMessageToWeiboRequest 第三方应用发送消息至微博客户端程序的消息结构体

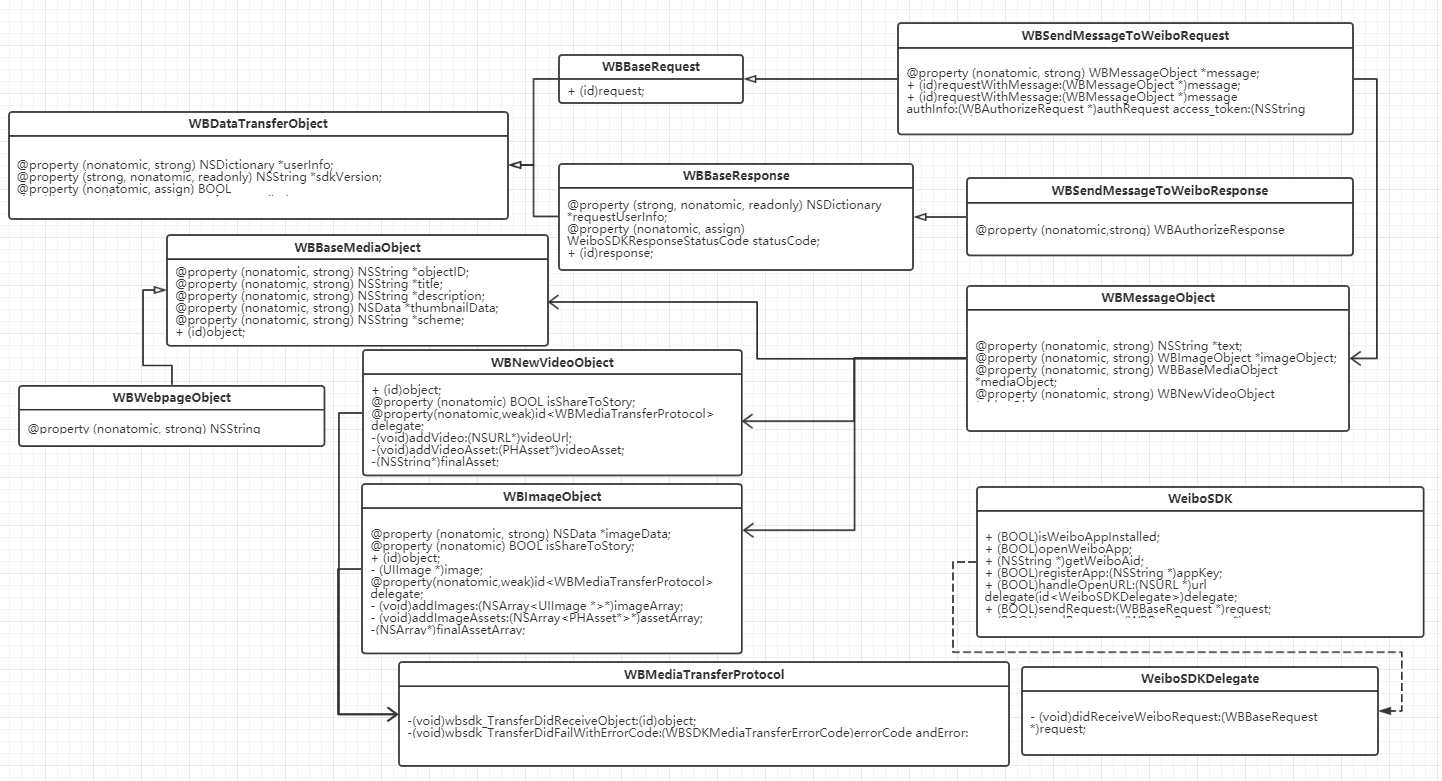
1. @interface WBSendMessageToWeiboRequest : WBBaseRequest
2. /\*\*
3. 发送给微博客户端的消息
4. \*/
5. @property (nonatomic, strong) WBMessageObject \*message;
6. /\*\*
7. 返回一个 WBSendMessageToWeiboRequest 对象
8. 此方法生成对象被[WeiboSDK sendRequest:]会唤起微博客户端的发布器进行分享，如果未安装微博客户端或客户端版本太低
9. 会根据 shouldOpenWeiboAppInstallPageIfNotInstalled 属性判断是否弹出安装/更新微博的对话框
10. @param message 需要发送给微博客户端的消息对象
11. @return 返回一个\*自动释放的\*WBSendMessageToWeiboRequest对象
12. \*/
13. + (id)requestWithMessage:(WBMessageObject \*)message;
14. /\*\*
15. 返回一个 WBSendMessageToWeiboRequest 对象
16. 当用户安装了可以支持微博客户端內分享的微博客户端时,会自动唤起微博并分享
17. 当用户没有安装微博客户端或微博客户端过低无法支持通过客户端內分享的时候会自动唤起SDK內微博发布器
18. @param message 需要发送给微博的消息对象
19. @param authRequest 授权相关信息,与access\_token二者至少有一个不为空,当access\_token为空并且需要弹出SDK內发布器时会通过此信息先进行授权后再分享
20. @param access\_token 第三方应用之前申请的Token,当此值不为空并且无法通过客户端分享的时候,会使用此token进行分享。
21. @return 返回一个\*自动释放的\*WBSendMessageToWeiboRequest对象
22. \*/
23. + (id)requestWithMessage:(WBMessageObject \*)message
24. authInfo:(WBAuthorizeRequest \*)authRequest
25. access\_token:(NSString \*)access\_token;
26. @end

另外数据类不是通过基类实现的，而是分别实现然后聚合在WBMessageObject的定义中。

1. /\*\*
2. 微博客户端程序和第三方应用之间传递的消息结构
3. 一个消息结构由三部分组成：文字、图片和多媒体数据。三部分内容中至少有一项不为空，图片和多媒体数据不能共存。(新版的多图和视频属于图片数据，并且图片和视频也不能共存)
4. \*/
5. @interface WBMessageObject : NSObject
6. /\*\*
7. 消息的文本内容
8. @warning 长度小于2000个汉字
9. \*/
10. @property (nonatomic, strong) NSString \*text;
11. /\*\*
12. 消息的图片内容
13. \*/
14. @property (nonatomic, strong) WBImageObject \*imageObject;
15. /\*\*
16. 消息的多媒体内容
17. \*/
18. @property (nonatomic, strong) WBBaseMediaObject \*mediaObject;
19. /\*\*
20. 消息的视频内容
21. \*/
22. @property (nonatomic, strong) WBNewVideoObject \*videoObject;
23. /\*\*
24. 返回一个 WBMessageObject 对象
25. @return 返回一个\*自动释放的\*WBMessageObject对象
26. \*/
27. + (id)message;
28. @end

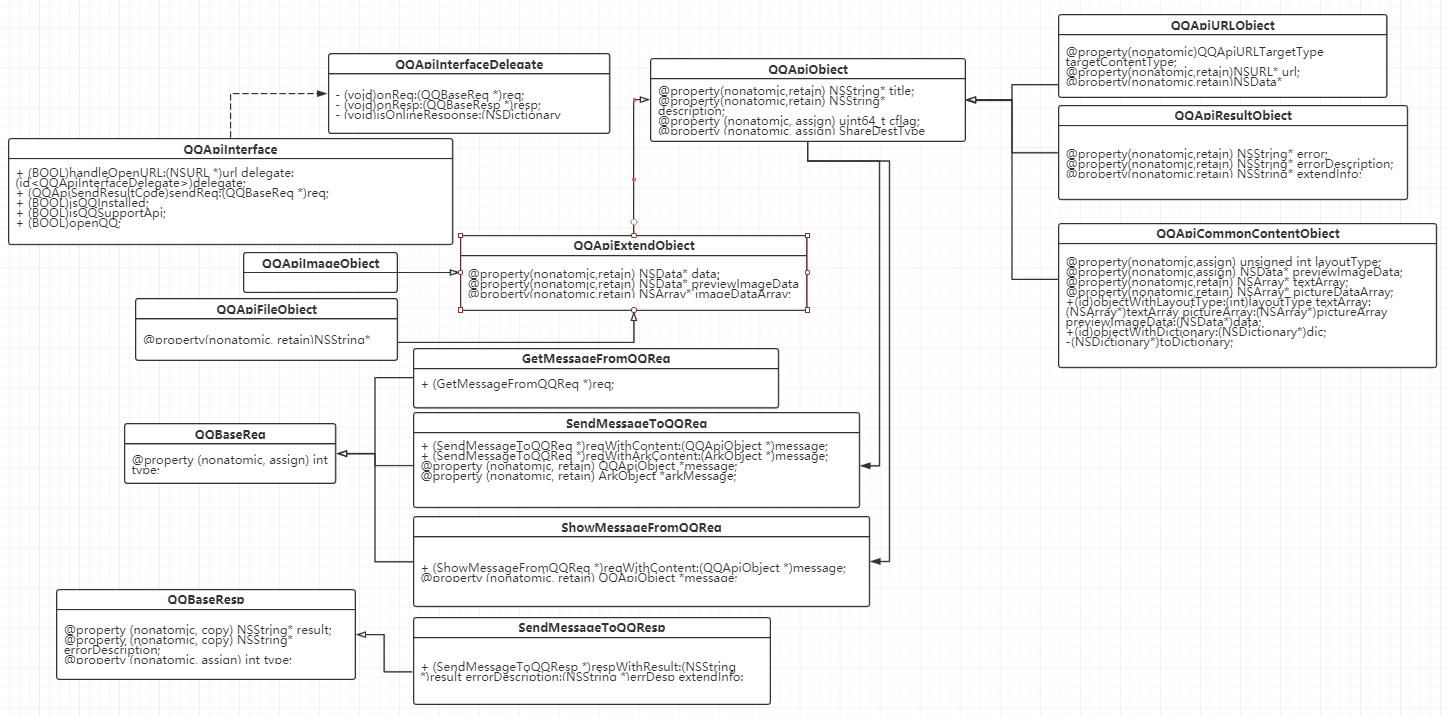
WeiboSDK定义了业务流程的方法，和微信平台类似不在赘述，具体差异可看类图

1. 核心类图



1. *腾讯（QQ）平台接口*

因为腾讯开放平台接口与微信类似，因此在这里只给出分析后的核心类图



**三、平台接口调研总结**

根据前面的调研，尤其是对类图的分析，可以总结出以下几点：

1. 业务流程上总体是一致的。即：检查平台应用安装情况，注册/登记身份信息，根据业务组装数据类，发送相应请求，实现delegate方法以接收平台response；

2.业务需要的参数类型是相似的。业务流程相似，传输的参数也具有相似性，比如标准的消息（Message）都会具有标题，简介等属性；

3.数据类型的结构是有差异的。虽然参数具有相似性，但实际采用的代码结构还是有所不同，如何用一种好的设计模式去兼容不同的数据结构，是设计模式调研的重要目标；

4.对数据的要求是不一致的。通过文档可以看到，即使是相同含义的字段，字段的限制也会有所不同，这也是抽象过程中需要注意的一点。

以上的分析将是之后数据和业务抽象的标准，对之后有分享功能设计具有重要的指导意义。

**四、资料来源以及参考文献**

1.微信开放平台iOS资源

https://open.weixin.qq.com/cgi-bin/showdocument?action=dir\_list&t=resource/res\_list&verify=1&id=open1419319164&token=&lang=zh\_CN

2.新浪微博开放平台iOS SDK

https://open.weibo.com/wiki/SDK#iOS\_SDK

3.腾讯开放平台iOS API

http://wiki.open.qq.com/index.php?title=iOS\_API%E8%B0%83%E7%94%A8%E8%AF%B4%E6%98%8E&oldid=46716