彩管家iOS IM说明文档

1.项目整体结构 3

2.项目目录说明 3

2.1 彩管家IM所有页面都处于Pages文件夹中。 3

2.2 Category存放所有的类别文件 4

2.3 Resources存放所有的资源文件 4

2.4 Library存放了所有的第三方库 4

2.5 Public存放所欲的公用文件 4

2.6 Chat存放IM核心代码和库 4

2.7 Header 存放头文件 4

3 接入步骤 5

3.1 将chat 文件夹add到项目中,并且修改项目build Setting ->Other Linker Flags –ObjC –all\_load 5

3.2 初始化IM 5

3.3 IM 发送消息 6

3.4 IM 接收消息 7

3.5 群功能操作 7

4 数据库操作 9

# 1.项目整体结构

**彩管家IM功能整体采用了MVC设计模式，im核心功能部分使用c++实现，以 lib\_im.lib包的形式集成到项目中的。**

**彩管家所有的页面实现代码都存于彩管家项目YouMai文件夹下。**



# 2.项目目录说明

#### 2.1 彩管家IM所有页面都处于Pages文件夹中。

****

#### 2.2 Category存放所有的类别文件

#### 2.3 Resources存放所有的资源文件

#### 2.4 Library存放了所有的第三方库

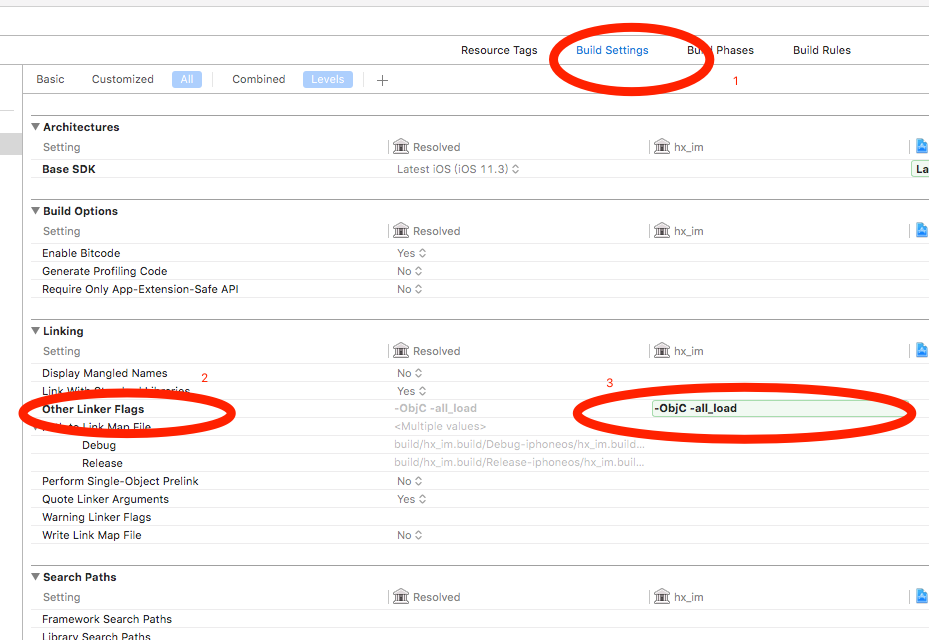
#### 2.5 Public存放所欲的公用文件

#### 2.6 Chat存放IM核心代码和库

#### 2.7 Header 存放头文件

# 3 接入步骤

#### 3.1 将chat 文件夹add到项目中,并且修改项目build Setting ->Other Linker Flags –ObjC –all\_load



#### 3.2 初始化IM

在app登录成功的位置初始化IM和登录IM。

初始化IM 在需要初始化的位置导入CZYIMManager.h 文件

[IMMannger initializeIp:服务器Ip andPort:服务器端口];

[IMMannger login:用户id andSessid: appid];

退出IM登录调用

[IMMannger IMOutLogin];

#### 3.3 IM 发送消息

在app中发送消息统一调用

[IMMannger sendMessageWithModel:<#(MessageModel \*)#>];

方法 ，IM支持的消息类型将封装成MessageModel从这个接口发送。

判断消息是否发送成功，在发送消息的位置设置回调

[IMMannger setSendCompletion:^(NSInteger error, int local\_id, NSInteger msgid) {

// error 为状态 0 成功 其余为失败，

// local\_id 回调对应的本地id

// msgid 消息发送成功后返回的这条消息的msgid，发送的消息本身是没有msgid

//在发送消息时我们每一条消息都对应有一个local\_id 我们只需要判断local\_id是否相等就知道这条消息的发送状态！

//如：model 为你发送消息设置的model

if (model.local\_id == local\_id)

{

if(error == 0)

{

//消息发送成功

}else{

//消息发送失败

}

}

}];

//退出页面时需要把这个回调置为nil

#### 3.4 IM 接收消息

在需要接收消息的地方设置回调

[IMMannger setReceiveCompletion:^(MessageModel \*model) {

//model 为收到的消息

}];

//退出页面时需要把这个回调置为nil

#### 3.5 群功能操作

所有的群操作我们都需要把要请求的数据封装成CZYGroupModel 类型进行请求

3.5.1创建群

[IMMannger CreateGroup:<#(CZYGroupModel \*)#>];

CZYGroupModel 中我封装有我们创建群需要的必要信息

创建群的回调

[IMMannger setReceiveGroupCompletion:^(CZYGroupModel \*model1, int error) {

// error=1 为成功回调

// model1 为新创建群的信息

}];

3.5.2 获取群列表

[IMMannger getGroup:<#(CZYGroupModel \*)#>];

获取群列表回调

[IMMannger setGroupCompletion:^(int local\_id, int error) {

// error=1 为成功回调 成功回调后内部将做更新数据库操作

//请求成功后 更新数据库

}];

3.5.3 获取群成员信息

[IMMannger getGroup\_member:<#(CZYGroupModel \*)#>];

获取群成员信息回调

[IMMannger setGroupCompletion:^(int local\_id, int error) {

// error=1 为成功回调 成功回调后内部将做更新数据库操作

//请求成功后 更新数据库

}];

3.5.4 获取群信息

[IMMannger getGroup\_info:<#(CZYGroupModel \*)#>];

获取群信息回调

[IMMannger setGroupCompletion:^(int local\_id, int error) {

// error=1 为成功回调 成功回调后内部将做更新数据库操作

//请求成功后 更新数据库

}];

3.5.5 增加或者删除群成员

[IMMannger ChangeGroup:<#(CZYGroupModel \*)#>];

[IMMannger setGroupCompletion:^(int local\_id, int error) {

// error=1 为成功回调 成功回调后内部将做更新数据库操作

//请求成功后 更新数据库 }];

3.5.6 修改群信息

[IMMannger Group\_info\_change:<#(CZYGroupModel \*)#> with:(GroupInfoModifyType)type];

typedef NS\_ENUM(NSUInteger,GroupInfoModifyType){

MODIFY\_NAME =1,//修改群名字

MODIFY\_AVATAR=2,//修改群头像

MODIFY\_TOPIC=3,//修改群公告

MODIFY\_OWNER=4,//修改群主

};

[IMMannger setGroupCompletion:^(int local\_id, int error) {

// error=1 为成功回调 成功回调后内部将做更新数据库操作

//请求成功后 更新数据库

}];

注：以上所有回调使用完成后，都要将其置为nil

# 4 数据库操作

在使用数据库的地方需导入 #import "CZYDBManager.h"

/\*\*插入消息\*/

-(void)insertMessageWithMessageModel:(MessageModel \*)msg\_model;

/\*\*查询一个用户的所有消息\*/

-(NSMutableArray<MessageModel \*>\*)QueryWithUserID:(NSString \*)dest\_id;

/\*\*查询消息列表\*/

-(NSMutableArray<MessageModel \*>\*)QueryWithMessageList;

-(int)QueryWithMAXLocal\_id;

/\*\*修改消息的发送状态\*/

-(BOOL)UpdateMessageStatusWhitModel:(MessageModel \*)model;

/\*\*删除某一个用户的所有消息\*/

-(void)DeleteMessageWithDestID:(NSString \*)dest\_id;

/\*\*删除某一条消息\*/

-(void)DeleteMessageWithModel:(MessageModel \*)model;

/\*\*删除消息列表\*/

-(void)DeleteMessageList;

/\*\*修改一个目标id未读标记为已读标记\*/

-(void)UpdateMessageisReadStatus:(NSString \*)dest\_id;

-(void)UpdateMessageModelisReadStatus:(MessageModel \*)model;

/\*\*修改一条语音消息的是否已经播放的标记\*/

-(void)UpdateAudioPlayerStatus:(MessageModel \*)model;

/\*\*缓存图片的Url 因为获取图片的url根据时间戳一直在变 所有只能缓存url\*/

-(void)UpdateMessageLocalUrl:(MessageModel \*)model;

/\*\*查询一个用户未读消息的条数\*/

-(NSInteger )QueryWithUserNotReadMessage:(NSString \*)dest\_id;

/\*\*查询全部未读消息的条数\*/

-(NSInteger )QueryWithAllNotReadMessage;

/\*\*查询消息列表是否存在某一个用户消息\*/

-(BOOL)QueryWithMessageWithDest\_id:(NSString \*)dest\_id;

#pragma mark 群列表数据库操作

/\*\*获取群列表\*/

-(NSMutableArray \*)QueryWithGroupList;

/\*\*更新群信息\*/

-(void)UpdataGroupInfo:(GroupInfoItem \*)group\_model;

/\*\*删除群\*/

-(void)DeleteGroupId:(int)groupid;

/\*\*获取群信息\*/

-(GroupInfoItem \*)QueryWithGroupid:(NSString \*)group\_id;

//-(void)DeletegroupList;

#pragma mark 群成员数据库操作

/\*\*更新群成员信息\*/

-(void)UpdataGroupMembers:(GroupMembers \*)group\_model;

/\*\*获取群成员信息\*/

-(GroupMembers \*)QueryGroupMembersModelWithGroupid:(NSString \*)group\_id;

#pragma mark 置顶数据库操作

/\*\*插入置顶用户id\*/

-(void)insertMessageTop:(NSString \*)dest\_id;

/\*\*删除置顶用户id\*/

-(void)DeleteMessageTop:(NSString \*)dest\_id;

/\*\*查询用户是否置顶\*/

-(BOOL)QueryUserIsTop:(NSString \*)dest\_id;

#pragma mark 免打扰

/\*\*插入免打扰用户id\*/

-(void)insertMessageDND:(NSString \*)dest\_id;

/\*\*删除免打扰用户id\*/

-(void)DeleteMessageDND:(NSString \*)dest\_id;

/\*\*查询用户是否免打扰\*/

-(BOOL)QueryUserIsDND:(NSString \*)dest\_id;