1 功能概述

聊天室模块位于`polyvLiveCommonModul`模块的`chatroom`包下，聊天室的信息接收及发送依赖于socket，即要用`PLVSocketLoginManager`完成登录后，才能正常使用聊天室的功能。聊天室模块采用mvp模式设计，将view层对sdk层的聊天室`chatroomManager`的直接调用隔离开，并将多个场景中对`chatroomManager`的共用代码抽离封装起来，使view层的逻辑变得更简洁，封装的presenter层负责`chatroomManager`的控制以及通知view层更新ui。

2 MVP模式

聊天室的mvp模式是用`IPLVChatroomContract`作为契约类，内部定义了聊天室mvp-view接口`IChatroomView`，以及聊天室mvp-presenter接口`IChatroomPresenter`。

```
public interface IPLVChatroomContract {

    // <editor-fold defaultstate="collapsed" desc="1、mvp-聊天室view接口">

    /**
     * mvp-聊天室view层接口
     */
    interface IChatroomView {
        /**
         * 设置presenter后的回调
         */
        void setPresenter(@NonNull IChatroomPresenter presenter);

        /**
         * 文本发言事件
         */
        @WorkerThread
        void onSpeakEvent(@NonNull PLVSpeakEvent speakEvent);

        /**
         * 获取聊天发言的表情图片的大小
         */
        @AnyThread
        int getSpeakEmojiSize();

        /**
         * 获取提问发言的表情图片的大小
         */
        @AnyThread
        int getQuizEmojiSize();

        /**
         * 图片事件
         */
        @WorkerThread
        void onImgEvent(@NonNull PLVChatImgEvent chatImgEvent);

        /**
         * 点赞事件
         */
        @WorkerThread
        void onLikesEvent(@NonNull PLVLikesEvent likesEvent);

        /**
         * 回答事件
         */
        @WorkerThread
        void onAnswerEvent(@NonNull PLVTAnswerEvent answerEvent);

        /**
         * 用户登录事件
         */
        @WorkerThread
        void onLoginEvent(@NonNull PLVLoginEvent loginEvent);
```

```
/**
 * 用户退出事件
 */
@WorkerThread
void onLogoutEvent(@NonNull PLVLogoutEvent logoutEvent);

/**
 * 发送公告事件
 */
@WorkerThread
void onBulletinEvent(@NonNull PolyvBulletinVO bulletinVO);

/**
 * 移除公告事件
 */
@WorkerThread
void onRemoveBulletinEvent();

/**
 * 商品上架/新增/编辑/推送事件
 */
@WorkerThread
void onProductControlEvent(@NonNull PLVProductControlEvent productControlEvent);

/**
 * 商品下架/删除事件
 */
@WorkerThread
void onProductRemoveEvent(@NonNull PLVProductRemoveEvent productRemoveEvent);

/**
 * 商品上移/下移事件
 */
@WorkerThread
void onProductMoveEvent(@NonNull PLVProductMoveEvent productMoveEvent);

/**
 * 商品库开关事件
 */
@WorkerThread
void onProductMenuSwitchEvent(@NonNull PLVProductMenuSwitchEvent productMenuSwitchEvent);

/**
 * 房间开启/关闭事件
 */
@WorkerThread
void onCloseRoomEvent(@NonNull PLVCloseRoomEvent closeRoomEvent);

/**
 * 移除信息事件
 *
 * @param id          移除单条信息的id，如果是移除所有信息，那么必定为null
 * @param isRemoveAll true：移除所有信息，false：移除单条信息
 */
@WorkerThread
void onRemoveMessageEvent(@Nullable String id, boolean isRemoveAll);
```

```
        /**
         * 自定义事件中的送礼事件
         *
         * @param userBean       送礼用户
         * @param customGiftBean 礼物数据
         */
        @WorkerThread
        void onCustomGiftEvent(@NonNull PolyvCustomEvent.UserBean userBean, @NonNull PLVCustomGiftBean
customGiftBean);

        /**
         * 自己本地发送的文本聊天信息
         */
        void onLocalSpeakMessage(@Nullable PolyvLocalMessage localMessage);

        /**
         * 自己本地发送的提问信息
         */
        void onLocalQuestionMessage(@Nullable PolyvQuestionMessage questionMessage);

        /**
         * 自己本地发送的图片信息
         */
        void onLocalImageMessage(@Nullable PolyvSendLocalImgEvent localImgEvent);

        /**
         * 违禁词触发
         *
         * @param prohibitedMessage 违规词
         * @param hintMsg           提示消息
         * @param status            状态
         */
        @MainThread
        void onSendProhibitedWord(@NonNull String prohibitedMessage, @NonNull String hintMsg, @NonNull String
status);

        /**
         * 接收到的需要添加到列表的文本发言、图片信息
         */
        @WorkerThread
        void onSpeakImgDataList(@Size(min = 1) List<PLVBaseViewData> chatMessageDataList);

        /**
         * 历史记录数据
         *
         * @param chatMessageDataList 数据列表
         * @param requestSuccessTime  请求成功的次数
         * @param isNoMoreHistory     请求成功后，是否还有未加载的历史记录
         * @param viewIndex           请求历史记录的mvp-view索引
         */
        @MainThread
        void onHistoryDataList(@Size(min = 0) List<PLVBaseViewData<PLVBaseEvent>> chatMessageDataList, int
requestSuccessTime, boolean isNoMoreHistory, int viewIndex);

        /**
```

```
         * 历史记录请求失败回调
         *
         * @param errorMsg  错误信息
         * @param t         异常
         * @param viewIndex 请求历史记录的mvp-view索引
         */
        @MainThread
        void onHistoryRequestFailed(String errorMsg, Throwable t, int viewIndex);
    }
    // </editor-fold>

    // <editor-fold defaultstate="collapsed" desc="2、mvp-聊天室presenter接口">

    /**
     * mvp-聊天室presenter层接口
     */
    interface IChatroomPresenter {
        /**
         * 注册view，可以注册多个
         */
        void registerView(@NonNull IChatroomView v);

        /**
         * 解除注册的view
         */
        void unregisterView(IChatroomView v);

        /**
         * 获取view的索引
         */
        int getViewIndex(IChatroomView v);

        /**
         * 初始化聊天室配置，该方法内部会设置聊天室的信息监听器
         */
        void init();

        /**
         * 发送聊天文本信息
         *
         * @param textMessage 要发送的信息，不能为空
         * @return true：成功提交，收到{@link IChatroomView#onLocalSpeakMessage(PolyvLocalMessage)}回调时为发送成功，收到{@link IChatroomView#onSendProhibitedWord(String, String, String)}为触发严禁词。<br/>
         * false：发送失败。
         */
        Pair<Boolean, Integer> sendChatMessage(PolyvLocalMessage textMessage);

        /**
         * 发送提问信息
         *
         * @param questionMessage 要发送的提问信息，不能为空
         */
        int sendQuestionMessage(PolyvQuestionMessage questionMessage);

        /**
         * 发送点赞信息
```

```
         */
        void sendLikeMessage();

        /**
         * 发送聊天图片信息
         */
        void sendChatImage(PolyvSendLocalImgEvent localImgEvent);

        /**
         * 发送自定义信息
         *
         * @param baseCustomEvent 自定义信息事件
         */
        <DataBean> void sendCustomMsg(PolyvBaseCustomEvent<DataBean> baseCustomEvent);

        /**
         * 发送自定义信息，示例为发送自定义送礼信息
         *
         * @param customGiftBean 自定义信息实例
         * @param tip            信息提示文案
         */
        PolyvCustomEvent<PLVCustomGiftBean> sendCustomGiftMessage(PLVCustomGiftBean customGiftBean, String tip);

        /**
         * 设置每次获取历史记录的条数，默认20条
         */
        void setGetChatHistoryCount(int count);

        /**
         * 请求聊天历史记录
         *
         * @param viewIndex 调用该方法的view的索引，没有时传0
         */
        void requestChatHistory(int viewIndex);

        /**
         * 获取历史记录成功的次数
         */
        int getChatHistoryTime();

        /**
         * 获取聊天室的数据
         */
        @NonNull
        PLVChatroomData getData();

        /**
         * 销毁，包括销毁聊天室操作、解除view操作
         */
        void destroy();
    }
    // </editor-fold>
}
```

2.1 Presenter实现逻辑

`IChatroomPresenter`的实现类为`PLVChatroomPresenter`。

其内部主要使用一下两个类来完成核心业务：

1、`IPLVLiveRoomDataManager`，直播间数据管理器

2、`PolyvChatroomManager`，sdk层的聊天室`chatroomManager`

2.1.1 初始化

`PLVChatroomPresenter`在构造器中进行了如下初始化操作：引用`IPLVLiveRoomDataManager`，创建`PLVChatroomData`，订阅聊天室信息，监听直播间数据。

```
public PLVChatroomPresenter(@NonNull IPLVLiveRoomDataManager liveRoomDataManager) {
    this.liveRoomDataManager = liveRoomDataManager;
    chatroomData = new PLVChatroomData();
    subscribeChatroomMessage();
    observeLiveRoomData();
}
```

`PLVChatroomData`是聊天室业务数据，内部保存的数据都是`MutableLiveData`类型，主要用于提供给非mvp的view监听/获取聊天室的数据，可以作用于不同业务模块间的数据监听及获取。

`subscribeChatroomMessage`方法的逻辑是定时把收集到的聊天室信息统一处理，避免收到信息后更新ui太频繁导致的一些问题。

`observeLiveRoomData`方法的逻辑是通过`IPLVLiveRoomDataManager`监听聊天室在后台配置相关的一些数据。

2.1.2 注册mvp-view

在ui层创建好mvp-view后，通过调用mvp-presenter的`registerView`方法，即可把mvp-view传给mvp-presenter，使得mvp-presenter可对mvp-view进行操作；同时mvp-presenter也会把自身传给mvp-view，使得mvp-view可以调用mvp-presenter提供的方法。

```
@Override
public void registerView(@NonNull IPLVChatroomContract.IChatroomView v) {
    if (iChatroomViews == null) {
        iChatroomViews = new ArrayList<>();
    }
    if (!iChatroomViews.contains(v)) {
        iChatroomViews.add(v);
    }
    v.setPresenter(this);
}
```

`PLVChatroomPresenter`可以注册多个`IChatroomView`。

3 SDK核心类介绍

聊天室的SDK核心类是`PolyvChatroomManager`，该类在`PLVChatroomPresenter`中被使用。

3.1 创建

`PolyvChatroomManager`是单例模式，因此通过`PolyvChatroomManager.getInstance`方法即可获取到实例对象。

```
PolyvChatroomManager chatroomManager = PolyvChatroomManager.getInstance();
```

3.2 初始化

```
/**
  * 初始化
  */
void init();
```

3.3 设置监听器

```
//设置严禁词监听器
PolyvChatroomManager.getInstance().setProhibitedWordListener(IPolyvProhibitedWordListener l);

//设置聊天信息监听器，聊天室是通过PolyvSocketWrapper接收信息的
PolyvSocketWrapper.getInstance().getSocketObserver().addOnMessageListener(PLVSocketMessageObserver.OnMessageList
ener l)
```

3.4 发送信息

```
/**
 * 发送聊天信息至聊天室
 *
 * @param localMessage 聊天信息实体
 * @param sessionId    直播或回放的场次id，没有时可以传null
 * @return {@link PolyvLocalMessage.SendValue}
 */
int sendChatMessage(PolyvLocalMessage localMessage, String sessionId);

/**
 * 发送聊天信息至聊天室
 *
 * @param localMessage   聊天信息实体
 * @param sessionId      直播或回放的场次id，没有时可以传null
 * @param needIdCallback 是否需要回调信息id
 * @param ack            ack回调
 * @return
 */
int sendChatMessage(PolyvLocalMessage localMessage, String sessionId, boolean needIdCallback, Ack ack);

/**
 * 发送提问信息
 *
 * @param questionMessage
 * @return {@link PolyvLocalMessage.SendValue}
 */
int sendQuestionMessage(PolyvQuestionMessage questionMessage);

/**
 * 发送自定义信息
 *
 * @param baseCustomEvent 自定义信息模型类
 * @param <DataBean>
 */
<DataBean> void sendCustomMsg(PolyvBaseCustomEvent<DataBean> baseCustomEvent);

/**
  * 发送点赞
  *
  * @param sessionId 直播或回放的场次id，没有时可以传null
  */
void sendLikes(String sessionId);

/**
  * 发送点赞
  *
  * @param sessionId 直播或回放的场次id，没有时可以传null
  */
void sendLikes(int count, String sessionId);

/**
  * 发送聊天图片
```

```
  */
void sendChatImage(PolyvSendLocalImgEvent localImgEvent, String sessionId);
```

3.5 销毁

```
/**
 * 销毁
 */
void destroy();
```