## 1 功能概述

common模块的ppt是使用mvp模式实现的，将PPT的一些UI无关的底层逻辑封装在了这个模块。

## 2 MVP模式

PPT模块包含了两个MVP的接口，分别是悬浮窗和PPT：

悬浮窗

```
/**
 * date: 2020/9/16
 * author: HWilliamgo
 * description: 悬浮窗业务MVP
 */
public interface IPLVLiveFloatingContract {

    /**
     * 悬浮窗View
     */
    interface IPLVLiveFloatingView {

        /**
         * 设置讲师信息
         *
         * @param nick   讲师昵称
         * @param picUrl 讲师图片Url
         */
        void updateTeacherInfo(String nick, String picUrl);
    }

    /**
     * 悬浮窗业务Presenter
     */
    interface IPLVLiveFloatingPresenter {
        /**
         * 初始化
         */
        void init(IPLVLiveFloatingView view);

        /**
         * 销毁
         */
        void destroy();
    }
}
```

PPTView

```
/**
 * date: 2020/9/16
 * author: HWilliamgo
 * description: PPT业务MVP
 */
public interface IPLVPPTContract {
    /**
     * PPTView
     */
    interface IPLVPPTView {

        /**
         * 发送消息到webView
         *
         * @param msg 消息
         */
        void sendMsgToWebView(String msg);

        /**
         * 发送消息到webView
         *
         * @param msg   消息
         * @param event 事件
         */
        void sendMsgToWebView(String msg, String event);

        /**
         * 隐藏加载中图片
         * 直播时，则聊天室登录后，收到PPT控制消息时，隐藏加载中图片。
         * 回放时，在收到PPT的prepare回调后，异常加载中图片。
         */
        void hideLoading();

    }

    /**
     * PPT Presenter
     */
    interface IPLVPPTPresenter {
        /**
         * 初始化
         *
         * @param view view
         */
        void init(IPLVPPTView view);

        /**
         * 移除消息延迟时间
         */
        void removeMsgDelayTime();

        /**
         * 恢复消息延迟时间
         */
        void recoverMsgDelayTime();
```

```
        /**
         * 发送画笔消息
         */
        void sendPPTBrushMsg(String msg);

        /**
         * 销毁
         */
        void destroy();
    }
}
```

### 2.1 Presenter实现逻辑

#### 2.1.1 PLVLiveFloatingPresenter

悬浮窗的Presenter主要的逻辑是，用socket管理器监听了聊天室发出的讲师信息事件，并将数据解析后更新到View接口：

```
PolyvSocketWrapper.getInstance().getSocketObserver().addOnMessageListener(onMessageListener = new
PLVSocketMessageObserver.OnMessageListener() {
            @Override
            public void onMessage(String listenEvent, String event, String message) {
                if (PLVEventConstant.Class.O_TEACHER_INFO.equals(event)) {
                    PLVTeacherInfoEvent teacherInfoEvent = PLVEventHelper.toMessageEventModel(message,
PLVTeacherInfoEvent.class);
                    if (teacherInfoEvent != null) {
                        String teacherNick = teacherInfoEvent.getData().getNick();
                        String picUrl = teacherInfoEvent.getData().getPic();
                        if (view != null) {
                            view.updateTeacherInfo(teacherNick, picUrl);
                        }
                    }
                }
            }
        });
```

#### 2.2.2 PLVPPTPresenter

PLVPPTPresenter中实现了对PPT事件的收发操作。

1. 监听server发来的PPT消息，解析后将消息发送到View层，由View层的webView进行处理：

```
PolyvSocketWrapper.getInstance().getSocketObserver().addOnMessageListener(
        onMessageListener = new PLVSocketMessageObserver.OnMessageListener() {
            @Override
            public void onMessage(String listenEvent, String event, String message) {
                if (PLVEventConstant.Ppt.ON_SLICE_START_EVENT.equals(event) ||
                        PLVEventConstant.Ppt.ON_SLICE_DRAW_EVENT.equals(event) ||
                        PLVEventConstant.Ppt.ON_ASSISTANT_CONTROL.equals(event) ||
                        PLVEventConstant.Ppt.ON_SLICE_OPEN_EVENT.equals(event) ||
                        PLVEventConstant.Ppt.ON_SLICE_ID_EVENT.equals(event)) {
                    //...
                } else if (PLVEventConstant.MESSAGE_EVENT_LOGIN.equals(event)) {
                    //发送login事件到ppt
                    //...
                }
            }
        }
);
```

2. 当用户操作PPT画笔时，要将画笔数据发送到server（该功能暂未实现）

```
//发送画笔事件
PolyvSocketWrapper.getInstance().emit(PLVEventConstant.MESSAGE_EVENT, message);
```

## 3 SDK核心类介绍

### 3.1 用PolyvSocketWrapper收发服务端PPT消息

common模块的PPT用到的SDK类是`PolyvSocketWrapper`,利用该类，既可以监听服务端老师发送的PPT事件，也可以主动发送PPT数据到服务端。

具体可以参考PLVLiveFloatingPresenter和PLVPPTPresenter中对该类的使用。

### 3.2 PPT使用的WebView

PPT使用的是SDK提供的`PolyvPPTWebView`类进行渲染的，对该类的使用均封装在了Scene模块的`PLVLCPPTView`类中，只是调用逻辑在presenter。

他有3类API：

1. 通用对外API

```
//注册被js调用的方法
public void registerHandler();
//加载webView
public void loadWeb();
//发送消息到JS
public void callMessage(String event, String message);
```

2. 直播对外API

```
//发送消息到JS
public void callUpdateWebView(String messages);
```

3. 回放对外API

```
//发送播放开始
public void callStart(String messages);
//发送播放暂停
public void callPause(String messages);
//发送seek
public void callSeek(String messages);
//发送PPT数据
public void callPPTParams(String messages);
```