1 功能概述

互动应用是多个场景都可以共用的一个功能模块，包含了如下互动应用：答题，问卷，公告，抽奖，签到。

由IPLVInteractLayout接口作为核心类来集成。

2 使用演示

```
//构造
ViewStub interactLayoutViewStub = findViewById(R.id.plvlc_ppt_interact_layout);
IPLVInteractLayout interactLayout = (IPLVInteractLayout) interactLayoutViewStub.inflate();
//初始化
interactLayout.init();
//显示公告
interactLayout.showBulletin();
//销毁
interactLayout.destroy();
```

具体的使用方法请参考PLVLCCloudClassActivity中对IPLVInteractLayout接口的调用。

3 接口介绍

IPLVInteractLayout作为互动应用的接口，定义了如下几个需要在Activity中使用的方法：

```
/**
 * date: 2020/10/9
 * author: HWilliamgo
 * 针对互动应用封装的Interface，可以在各个场景使用，定义了：
 * 1. 外部直接调用的方法
 */
public interface IPLVInteractLayout {
    // <editor-fold defaultstate="collapsed" desc="1. 外部直接调用的方法">
    /**
     * 初始化
     */
    void init();

    /**
     * 显示公告
     */
    void showBulletin();

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

    /**
     * 点击返回
     *
     * @return 返回true表示拦截事件
     */
    boolean onBackPress();
    // </editor-fold>
}
```

4 实现介绍

PLVInteractLayout是IPLVInteractLayout接口的实现类，它内部实现了互动应用的逻辑：

4.1 初始化

```
@Override
public void init() {
    if (interactWebView == null) {
        return;
    }
    //加载互动应用资源
    if (LOAD_WEB_URL) {
        interactWebView.loadWeb();
    } else {
        interactWebView.loadUrl("file:///android_asset/index.html");
    }
    //设置要显示的互动应用
    setupInteractApp();
}
```

1. 加载资源

   初始化时，根据当前的需求选择加载在线互动应用资源还是加载本地互动应用资源，即根据变量`LOAD_WEB_URL`。默认为true表示是加载在线资源，如果需要加载本地资源，则将这里的`LOAD_WEB_URL`改为false即可。

2. 设置互动应用

   调用`setupInteractApp()`方法设置具体的互动应用，将所需的互动应用添加进来。

4.2 设置互动应用

设置互动应用即将想要的互动应用添加进来。见方法：`setupInteractApp()`

```
//通用控制
PLVInteractCommonControl commonControl = new PLVInteractCommonControl();
//互动答题
PLVInteractAnswer interactAnswer = new PLVInteractAnswer();
//互动问卷调查
LVInteractQuestionnaire interactQuestionnaire = new PLVInteractQuestionnaire();
//互动抽奖
interactLottery = new PLVInteractLottery(commonControl);
//互动签到
PLVInteractSignIn interactSignIn = new PLVInteractSignIn();
//互动公告
interactBulletin = new PLVInteractBulletin();


interactWebView.addInteractApp(commonControl);
interactWebView.addInteractApp(interactAnswer);
interactWebView.addInteractApp(interactQuestionnaire);
interactWebView.addInteractApp(interactLottery);
interactWebView.addInteractApp(interactSignIn);
interactWebView.addInteractApp(interactBulletin);
```

`setupInteractApp()` 方法的逻辑分为两步：

1. 实例化具体的互动应用，并设置对应的回调。注意，通用控制不属于业务上的互动应用，他是用于控制所有互动应用的一个通用类，是必须要添加的。
2. 将所有互动应用添加到互动应用webView中。

4.3 互动应用的实现

4.3.1 互动应用类简介

互动应用的具体实现在interact/app目录下，共有5个互动应用，每个类分别表示：

1. PLVInteractAnswer：答题
2. PLVInteractBulletin：公告
3. PLVInteractLottery：抽奖
4. PLVInteractQuestionnaire：问卷
5. PLVInteractSignIn：签到

该目录的其他类：

1. PLVInteractCommonControl：通用控制类，用于每个互动应用共用的逻辑封装在这个类中
2. IPLVInteractSendServerResultToJs：互动应用发送到server的结果再发送到JS的监听器

4.3.2 具体互动应用实现逻辑和如何自定义UI

1. 实现逻辑

   我们找一个较为简单的互动应用看实现逻辑，如：互动签到。

   `PLVInteractSignIn`类的实现中关键分为两个方法。

   1. `processSocketMsg(String msg, String event)`用于处理socket消息

      根据event判断时间类型，过滤出签到消息，并处理msg变量的数据，再调用`sendMsgToJs`方法，对应事件和数据发送到webView，由webView做出相应的UI相应。

   2. `registerMsgReceiverFromJs(IPLVInteractJSBridge interactJSBridge)`用于处理JS消息

      用`IPLVInteractJSBridge`接口的方法注册JS事件监听器，在收到JS发送来的签到事件时（也就是用户点击了webView上的签到），对数据进行处理，并将结果发送到server。

   3. `sendResultToServer(PolyvSignIn2SocketVO socketVO)`用于发送结果到server。

      调用对应的方法将结果发送到服务端。

2. 自定义UI

   自定义UI的方式分为两种，我们还是以签到为例：

   1. 重写`PLVInteractSignIn`方法：
      1. 重写`processSocketMsg`方法中的逻辑，提取出服务端发送来的数据后，不再调用`sendMsgToJs()`将事件发送到webView，将数据渲染到你自定义的界面UI上即可。
      2. 将`registerMsgReceiverFromJs`的方法体清空。因为不再需要用webView渲染界面了，因此也不再监听webView发送来的事件了。
      3. 发送数据到server:当你的界面UI响应了对应的事件后，直接调用`sendResultToServer`，并发送数据到server。
   2. 继承`PLVInteractAppAbs`类，实现自己的逻辑，并自己在`PLVInteractLayout`中添加到webView。

4.4 SDK核心类介绍

4.1.1 PLVInteractWebView

互动应用WebView，负责渲染互动应用的UI,和处理维护多个互动应用。

接口如下：

```
/**
*  加载webView页面
*/
public void loadWeb() ;

/**
 * 添加互动应用
 *
 * @param interactAppAbs 互动应用
 */
public void addInteractApp(PLVInteractAppAbs interactAppAbs);


/**
 * 移除互动应用
 *
 * @param interactAppAbs 互动应用
 */
public void removeInteractApp(PLVInteractAppAbs interactAppAbs);

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

4.1.2 PLVInteractAppAbs

PLVInteractAppAbs是互动应用基础类，所有的互动应用以该类作为父类拓展，来实现各自的业务逻辑。

抽象方法：

需要具体的子类互动应用重写的方法。

```
/**
 * 处理socket消息
 *
 * @param msg   消息
 * @param event 事件
 */
protected abstract void processSocketMsg(String msg, String event);
/**
 * 注册JS发到原生的消息接收器
 */
protected abstract void registerMsgReceiverFromJs(IPLVInteractJSBridge interactJSBridge);
```

封装子类共用API:

```
/**
 * 设置显示隐藏监听器
 *
 * @param onInteractAppShowListener listener
 */
public void setOnShowListener(OnInteractAppShowListener onInteractAppShowListener);
```

子类使用的方法:

```
/**
 * 通知UI显示
 */
protected void notifyShow();

/**
 * 发送消息到JS
 *
 * @param handlerName 事件名字
 * @param data        数据
 * @param callBack    回调
 */
protected void sendMsgToJs(String handlerName, String data, CallBackFunction callBack);
```