1 功能概述

文档（PPT&黑板）模块包括黑板、PPT展示、画笔标注等功能。文档模块的整个界面被封装在`PLVHCDocumentLayout`中。

2 使用演示

使用文档模块只需要在Activity的布局中添加进`PLVHCDocumentLayout`布局，并调用其初始化方法即可。

```
// PLVHCLiveHiClassActivity.java
private void initView() {
    // 文档布局
    plvhcDocumentLy = findViewById(R.id.plvhc_document_ly);
    // 初始化文档布局，传入初始化所需的参数
    plvhcDocumentLy.init(liveRoomDataManager);
}
```

同时，也可以设置布局的监听回调，以便在不同模块布局之间进行通信。

```
// IPLVHCDocumentLayout 设置监听器方法定义
/**
  * 设置view交互事件监听器
  *
  * @param listener 监听器
  */
void setOnViewActionListener(OnViewActionListener listener);

/**
  * view交互事件监听器
  */
interface OnViewActionListener {

    /**
      * 回调标注工具按钮状态变更
      *
      * @param showUndoButton   是否显示撤销按钮
      * @param showDeleteButton 是否显示删除按钮
      */
    void onChangeMarkToolOperationButtonState(boolean showUndoButton,
                                              boolean showDeleteButton);
}
```

3 实现介绍

3.1 IPLVHCDocumentLayout

文档布局是一个ViewGroup，它主要由一个负责显示黑板PPT的WebView - `IPLVDocumentContainerView`，和一个叠加在WebView上面负责控制标注画笔等功能的控制栏 - `PLVHCMarkToolControllerLayout`构成。

```
@Override
public void init(IPLVLiveRoomDataManager liveRoomDataManager) {
    // 初始化 MVP - View
    initMvpView();
    // 初始化文档Webview
    initContainerView(liveRoomDataManager);
}
```

3.2 PLVHCMarkToolControllerLayout

控制栏是一个皮肤布局，包括标注工具、画笔、文字、橡皮檫、移动黑板等功能。它不直接操纵黑板PPT的WebView，而是通过调用`PLVHCDocumentLayout`提供的相关方法，由`PLVHCDocumentLayout`再对文档WebView状态进行修改。下面代码展示当使用者点击箭头标注工具，是如何告诉WebView切换到箭头标注状态的。

```
// PLVHCMarkToolControllerLayout.java
//初始化标注集合
markToolViewMap = mapOf(
        // 箭头标注的view及其对应的枚举
        pair(PLVHCMarkToolEnums.MarkTool.ARROW, plvhcToolbarMarkToolArrowIv),
        // 此处省略其他代码
);
//设置标注的点击事件
foreach(markToolViewMap.entrySet(), new PLVSugarUtil.Consumer<Map.Entry<PLVHCMarkToolEnums.MarkTool,
PLVRoundImageView>>() {
    @Override
    public void accept(Map.Entry<PLVHCMarkToolEnums.MarkTool, PLVRoundImageView> entry) {
        final PLVHCMarkToolEnums.MarkTool markTool = entry.getKey();
        final PLVRoundImageView view = entry.getValue();
        view.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                if (!PLVHCMarkToolEnums.MarkTool.CLEAR.equals(markTool)) {
                    updateCurrentSelectedMarkTool(markTool);
                }
                if (onChangeMarkToolStateListener != null) {
                    // 标注工具类型变更回调触发
                    onChangeMarkToolStateListener.onChangeMarkTool(markTool);
                }
                hide();
            }
        });
    }
});


// PLVLSDocumentLayout.java
if (markTool != PLVHCMarkToolEnums.MarkTool.CLEAR) {
    lastSelectMarkTool = markTool;
    // 其他标注工具的处理逻辑
    containerView.sendEvent(new PLVChangeApplianceEvent(markTool.getAppliances()));
} else {
    // 清屏时的处理
}
```