## 1 功能概述

文档模块是三分屏开播端使用到的功能，它包括控制文档WebView的显示内容，频道文档的管理操作。

## 2 MVP模式

文档模块使用MVP模式，接口定义为 `IPLVDocumentContract` 类。

### 2.1 View

文档MVP模式对应的View为 `PLVAbsDocumentView` ，这是一个空实现的抽象View。在注册MVP-View时可以通过继承该View，只重写部分需要监听回调的方法。

### 2.2 Presenter

文档MVP模式对应的Presenter为 `PLVDocumentPresenter` ，它提供了许多文档操作的功能方法。在使用该Presenter前，**必须**调用 `init()` 方法对其进行初始化。

```
interface Presenter {
    /**
     * 初始化方法，必须调用一次
     *
     * @param lifecycleOwner 生命周期
     * @param liveRoomDataManager 直播间数据管理器
     * @param documentWebProcessor 封装的WebView处理器
     */
    void init(LifecycleOwner lifecycleOwner,
              IPLVLiveRoomDataManager liveRoomDataManager,
              PLVSDocumentWebProcessor documentWebProcessor);

}
```

### 2.2.1 向View层的调用

一般来说，由Presenter向View的调用都是带有回调性质的。大部分的回调都是通过监听Model层中LiveData的数据变更，然后向view层调用实现的。下面示例代码展示了在通过网络请求获取PPT列表数据后，是如何回调给view层的。

```
/**
 * 监听Model层所有PPT文档列表更新
 * 向view层回调
 *
 * @param lifecycleOwner
 */
private void observePptInfo(LifecycleOwner lifecycleOwner) {
    plvDocumentRepository.getPptInfoLiveData().observe(lifecycleOwner, new
Observer<PLVStatefulData<PLVSPPTInfo>>() {
        @Override
        public void onChanged(@Nullable PLVStatefulData<PLVSPPTInfo> plvspptInfo) {
            if (plvspptInfo == null || !plvspptInfo.isSuccess()) {
                return;
            }
            for (WeakReference<IPLVDocumentContract.View> viewWeakReference : viewWeakReferenceList) {
                IPLVDocumentContract.View view = viewWeakReference.get();
                if (view != null) {
                    view.onPptCoverList(plvspptInfo.getData());
                }
            }
        }
    });
}
```

当Model层`plvDocumentRepository`进行网络请求取到ppt列表数据后，会更新`pptInfoLiveData`中的数据。这里订阅了LiveData的数据更新，在数据有变更时会触发onChanged方法，对每一个注册的view调用`onPptCoverList()`来提醒ppt列表数据的更新。

### 2.2.2 向Model层的调用

`PLVDocumentPresenter`依赖2个Model层对象，分别是负责与Webview进行交互，以及具有文档管理功能的`PLVDocumentRepository`，负责在上传文档时进行上传进度本地缓存的`PLVPptUploadLocalRepository`。在这里，我们把PPT Webview也视为了一个远端对象，与Webview的交互视为与网络请求等价，因此由Model层进行Webview的交互处理。

下面示例代码展示了如何让Webview切换显示的ppt。

```
@Override
public void changePpt(int autoId) {
    if (!checkInitialized()) {
        return;
    }
    plvDocumentRepository.sendWebMessage(PLVSDocumentWebProcessor.CHANGEPPT, "{\"autoId\":" + autoId + "}");
```

其中autoId为后端返回的ppt的标识符，在ppt列表中会携带该数据。这里首先需要通过`checkInitialized()`检查Presenter是否已经经过初始化，随后直接调用Model层方法发送数据。

### 2.3 Model

#### 2.3.1 PLVDocumentRepository

`PLVDocumentRepository`负责向WebView发送数据，向服务器发送请求，以及接收相应的返回数据。PLVDocumentRepository不持有Presenter的引用，向presenter返回数据的方式是由presenter主动订阅LiveData。以下示例代码展示了获取ppt列表的方式。

```
/**
 * 请求更新PPT文档列表数据
 * 回调 {@link #getPptInfoLiveData()}
 */
public void requestPptCoverList() {
    PLVDocumentDataManager.getDocumentList(new PLVrResponseCallback<PLVSPPTInfo>() {
        @Override
        public void onSuccess(PLVSPPTInfo plvspptInfo) {
            cachePptInfo = plvspptInfo;
            plvsPptInfoLiveData.postValue(PLVStatefulData.success(cachePptInfo));
        }

        @Override
        public void onFinish() {
        }
    });
}
```

`PLVDocumentDataManager`是SDK内部提供的文档数据类，里面封装了获取ppt列表的http请求，直接调用`getDocumentList()`方法即可在回调的onSuccess中拿到ppt列表数据。获取到ppt列表后更新plvsPptInfoLiveData的数据，若presenter订阅了该liveData，则能够自动在订阅回调里获取到新的数据。

#### 2.3.2 PLVPptUploadLocalRepository

`PLVPptUploadLocalRepository`负责在PPT文档进行上传时，在本地缓存记录上传进度。下次重新登录进入直播间时，会检查是否存在上次上传文档未完成的情况。这个类利用SharedPreferences进行数据的本地存储。