1 功能概述

播放器模块位于`polyvLiveCommonModul`模块的`player`包下，包含直播以及回放播放器。该模块采用mvp模式设计，将view层对sdk层的播放器`videoView`的直接调用隔离开，并将多个场景中对`videoView`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`videoView`的控制以及通知view层更新ui。

2 MVP模式

直播播放器的mvp模式是用`IPLVLivePlayerContract`作为契约类，内部定义了直播播放器mvp-view接口`ILivePlayerView`，以及直播播放器mvp-presenter接口`ILivePlayerPresenter`。

```
public interface IPLVLivePlayerContract {

    // <editor-fold defaultstate="collapsed" desc="1、mvp-直播播放器view层接口">

    /**
     * mvp-直播播放器view层接口
     */
    interface ILivePlayerView {
        /**
         * 设置presenter后的回调
         */
        void setPresenter(@NonNull ILivePlayerPresenter presenter);

        /**
         * 获取主播放器view
         */
        PolyvLiveVideoView getLiveVideoView();

        /**
         * 获取暖场播放器view
         */
        PolyvAuxiliaryVideoview getSubVideoView();

        /**
         * 获取播放器缓冲视图
         */
        View getBufferingIndicator();

        /**
         * 获取暂无直播显示的视图
         */
        View getNoStreamIndicator();

        /**
         * 子播放器开始播放回调
         *
         * @param isFirst 每次加载完成后是否是第一次start播放
         */
        void onSubVideoViewPlay(boolean isFirst);

        /**
         * 子播放器点击事件
         *
         * @param mainPlayerIsPlaying 主播放器是否在播放中
         */
        void onSubVideoViewClick(boolean mainPlayerIsPlaying);

        /**
         * 主播放器播放失败回调
         *
         * @param error 失败数据
         * @param tips  错误提示
         */
        void onPlayError(PolyvPlayError error, String tips);
```

```
        /**
         * 暂无直播回调
         */
        void onNoLiveAtPresent();

        /**
         * 直播推流暂停回调
         */
        void onLiveStop();

        /**
         * 直播结束回调
         */
        void onLiveEnd();

        /**
         * 准备完成回调
         *
         * @param mediaPlayMode 音视频播放模式
         */
        void onPrepared(@PolyvMediaPlayMode.Mode int mediaPlayMode);

        /**
         * 线路切换回调
         *
         * @param linesPos 线路索引
         */
        void onLinesChanged(int linesPos);

        /**
         * 获取跑马灯回调
         *
         * @param marqueeVo  跑马灯数据
         * @param viewerName 观看用户名
         */
        void onGetMarqueeVo(PolyvLiveMarqueeVO marqueeVo, String viewerName);

        /**
         * 重新开始播放回调
         */
        void onRestartPlay();

        /**
         * 手势触发的亮度改变事件
         *
         * @param changeValue 亮度值，范围：[0,100]
         * @param isEnd       手势是否结束
         * @return 是否要改变亮度
         */
        boolean onLightChanged(int changeValue, boolean isEnd);

        /**
         * 手势触发的音量改变事件，changeValue：，return：是否要改变音量
         *
         * @param changeValue 音量值，范围：[0,100]
         * @param isEnd       手势是否结束
```

```
         * @return 是否要改变音量
         */
        boolean onVolumeChanged(int changeValue, boolean isEnd);

        /**
         * 该频道直播的服务端弹幕开关
         *
         * @param isServerDanmuOpen true：开启了弹幕，false：关闭了弹幕
         */
        void onServerDanmuOpen(boolean isServerDanmuOpen);

        /**
         * 根据频道的类型，决定是否要显示ppt
         *
         * @param visible {@link View#VISIBLE}
         */
        void onShowPPTView(int visible);

        /**
         * 断网重连
         *
         * @return true表示不使用播放器内部重连逻辑，false表示使用。
         */
        boolean onNetworkRecover();
    }
    // </editor-fold>

    // <editor-fold defaultstate="collapsed" desc="2、mvp-直播播放器presenter层接口">

    /**
     * mvp-直播播放器presenter层接口
     */
    interface ILivePlayerPresenter {
        /**
         * 注册view
         */
        void registerView(@NonNull ILivePlayerView v);

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

        /**
         * 初始化播放器配置
         */
        void init();

        /**
         * 开始播放
         */
        void startPlay();

        /**
         * 重新开始播放
         */
```

```
        void restartPlay();

        /**
         * 暂停播放
         */
        void pause();

        /**
         * 恢复播放
         */
        void resume();

        /**
         * 停止播放
         */
        void stop();

        /**
         * 是否在播放中
         */
        boolean isPlaying();

        /**
         * 获取可以切换的线路数量
         */
        int getLinesCount();

        //获取当前线路可以切换的码率(清晰度)信息
        @Nullable
        List<PolyvDefinitionVO> getBitrateVO();

        /**
         * 获取播放模式
         *
         * @return @{@link PolyvMediaPlayMode.Mode}
         */
        int getMediaPlayMode();

        /**
         * 改变播放模式
         */
        void changeMediaPlayMode(@PolyvMediaPlayMode.Mode int mediaPlayMode);

        /**
         * 切换线路
         *
         * @param linesPos 线路索引
         */
        void changeLines(int linesPos);

        /**
         * 切换码率
         *
         * @param bitRate 码率索引
         */
        void changeBitRate(int bitRate);
```

```
    /**
     * 截图
     *
     * @return 截图的图片
     */
    @Nullable
    Bitmap screenshot();

    /**
     * 获取视频信息
     *
     * @return 视频信息数据
     */
    @Nullable
    PolyvLiveChannelVO getChannelVO();

    /**
     * 获取当前线路索引
     */
    int getLinesPos();

    /**
     * 获取当前码率(清晰度)索引
     */
    int getBitratePos();

    /**
     * 设置系统音量
     *
     * @param volume 音量值，范围：[0,100]
     */
    void setVolume(int volume);

    /**
     * 获取系统音量
     *
     * @return 音量值，范围：[0,100]
     */
    int getVolume();

    /**
     * 设置播放器音量
     *
     * @param volume 音量值，范围：[0,100]
     */
    void setPlayerVolume(int volume);

    /**
     * 是否需要手势
     *
     * @param need true：需要，false：不需要
     */
    void setNeedGestureDetector(boolean need);

    /**
```

```
         * 获取直播播放器数据
         */
        @NonNull
        PLVLivePlayerData getData();

        /**
         * 销毁，包括销毁播放器、解除view
         */
        void destroy();
    }
    // </editor-fold>
}
```

2.1 Presenter实现逻辑

`ILivePlayerPresenter`的实现类为`PLVLivePlayerPresenter`。

其内部主要使用以下两个类来完成核心业务：

1、`IPLVLiveRoomDataManager`，直播间数据管理器

2、`PolyvLiveVideoView`，sdk层的播放器`videoView`

2.1.1 初始化

`PLVLivePlayerPresenter`在构造器中进行了如下初始化操作：引用`IPLVLiveRoomDataManager`，创建`PLVLivePlayerData`。

```
public PLVLivePlayerPresenter(@NonNull IPLVLiveRoomDataManager liveRoomDataManager) {
    this.liveRoomDataManager = liveRoomDataManager;
    livePlayerData = new PLVLivePlayerData();
}
```

`PLVLivePlayerData`是直播播放器数据，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/获取播放器的数据，可以作用于不同业务模块间的数据监听及获取。

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 IPLVLivePlayerContract.ILivePlayerView v) {
    this.vWeakReference = new WeakReference<>(v);
    v.setPresenter(this);
}
```

3 SDK核心类介绍

直播播放器的SDK核心类是`PolyvLiveVideoView`，该类在`PLVLivePlayerPresenter`中被使用。

3.1 创建

`PolyvLiveVideoView`是一个view，因此可以在布局中创建。

```
<com.easefun.polyv.livescenes.video.PolyvLiveVideoView
    android:id="@+id/live_video_view"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
</com.easefun.polyv.livescenes.video.PolyvLiveVideoView>
```

通过`findViewById`找到`PolyvLiveVideoView`后，可以通过播放器mvp-view的`getLiveVideoView`方法，将`PolyvLiveVideoView`传给播放器mvp-presenter使用。

3.2 初始化

可以根据业务给`PolyvLiveVideoView`配置一些view，使用示例：

PLVLCLiveMediaLayout

```
private void initVideoView() {
    //设置子播放器view
    videoView.setSubVideoView(subVideoView);
    //设置音频模式view
    videoView.setAudioModeView(audioModeView);
    //设置loadingView
    videoView.setPlayerBufferingIndicator(loadingView);
    //设置无直播时显示的view
    videoView.setNoStreamIndicator(noStreamView);
    //设置直播停止时显示的view
    videoView.setStopStreamIndicator(stopStreamView);
    //设置控制栏
    videoView.setMediaController(mediaController);
    //设置跑马灯
    videoView.post(new Runnable() {
        @Override
        public void run() {
            marqueeView = ((Activity) getContext()).findViewById(R.id.plvlc_marquee_view);//after videoLayout
add, post find
            videoView.setMarqueeView(marqueeView, marqueeItem = new PolyvMarqueeItem());
        }
    });
}
```

3.3 设置监听器

`PolyvLiveVideoView` 提供很多设置监听器以监听播放器相关的信息。

```
/**
 * 设置视频播放完成回调，只有url播放才会触发该事件
 *
 * @param l
 */
public void setOnCompletionListener(IPolyvVideoViewListenerEvent.OnCompletionListener l);

/**
 * 设置视频已准备好马上进入播放回调
 *
 * @param l
 */
public void setOnPreparedListener(IPolyvVideoViewListenerEvent.OnPreparedListener l);

/**
 * 设置视频播放器内部错误回调，只有url播放才会触发该事件
 *
 * @param l
 */
public void setOnErrorListener(IPolyvVideoViewListenerEvent.OnErrorListener l);

/**
 * 设置视频播放器信息有变更回调
 *
 * @param l
 */
public void setOnInfoListener(IPolyvVideoViewListenerEvent.OnInfoListener l);

/**
 * 设置seek完成回调，只有url播放才会触发该事件
 *
 * @param l
 */
public void setOnSeekCompleteListener(IPolyvVideoViewListenerEvent.OnSeekCompleteListener l);

/**
 * 设置视频尺寸改变回调
 *
 * @param l
 */
public void setOnVideoSizeChangedListener(IPolyvVideoViewListenerEvent.OnVideoSizeChangedListener l);


/**
 * 设置视频缓存更新回调，只有url播放才会触发该事件
 *
 * @param l
 */
public void setOnBufferingUpdateListener(IPolyvVideoViewListenerEvent.OnBufferingUpdateListener l);

/**
 * 设置视频播放回调
 *
 * @param l
 */
```

```
public void setOnVideoPlayListener(IPolyvVideoViewListenerEvent.OnVideoPlayListener l);

/**
 * 设置视频暂停回调
 *
 * @param l
 */
public void setOnVideoPauseListener(IPolyvVideoViewListenerEvent.OnVideoPauseListener l);


/**
 * 设置暖场图片弹出监听回调
 *
 * @param l
 */
public void setOnCoverImageOutListener(IPolyvVideoViewListenerEvent.OnCoverImageOutListener l);

/**
 * 设置手势左向上回调
 *
 * @param l
 */
public void setOnGestureLeftUpListener(IPolyvVideoViewListenerEvent.OnGestureLeftUpListener l);

/**
 * 设置手势左向下回调
 *
 * @param l
 */
public void setOnGestureLeftDownListener(IPolyvVideoViewListenerEvent.OnGestureLeftDownListener l);

/**
 * 设置手势右向上回调
 *
 * @param l
 */
public void setOnGestureRightUpListener(IPolyvVideoViewListenerEvent.OnGestureRightUpListener l);

/**
 * 设置手势右向下回调
 *
 * @param l
 */
public void setOnGestureRightDownListener(IPolyvVideoViewListenerEvent.OnGestureRightDownListener l);

/**
 * 设置手势左滑回调
 *
 * @param l
 */
public void setOnGestureSwipeLeftListener(IPolyvVideoViewListenerEvent.OnGestureSwipeLeftListener l);

/**
 * 设置手势右滑回调
 *
 * @param l
```

```
     */
    public void setOnGestureSwipeRightListener(IPolyvVideoViewListenerEvent.OnGestureSwipeRightListener l);

    /**
     * 设置手势单击回调
     *
     * @param l
     */
    public void setOnGestureClickListener(IPolyvVideoViewListenerEvent.OnGestureClickListener l);

    /**
     * 设置手势双击回调
     *
     * @param l
     */
    public void setOnGestureDoubleClickListener(IPolyvVideoViewListenerEvent.OnGestureDoubleClickListener l);

    /**
     * 设置PPT显示回掉
     *
     * @param l
     */
    public void setOnPPTShowListener(IPolyvVideoViewListenerEvent.OnPPTShowListener l);

    /**
     * 设置视频重新加载
     *
     * @param l
     */
    public void setOnVideoViewRestartListener(IPolyvVideoViewListenerEvent.OnVideoViewRestart l);

    /**
     * 设置获取直播后台设置的跑马灯样式的监听器
     *
     * @param l
     */
    public void setOnGetMarqueeVoListener(IPolyvVideoViewListenerEvent.OnGetMarqueeVoListener l);

    /**
     * 设置SEI信息回调
     *
     * @param l
     */
    void setOnSEIRefreshListener(IPolyvVideoViewListenerEvent.OnSEIRefreshListener l);

    /**
     * 设置网络状态监听器回调
     *
     * @param l
     */
    void setOnNetworkStateListener(IPolyvVideoViewListenerEvent.OnNetworkStateListener l);
```

设置监听器的示例可以在`PLVLivePlayerPresenter`中找到。

3.4 开始播放

`PolyvLiveVideoView`提供了开始播放视频的方法。

```
/**
 * 点播与直播的播放入口
 *
 * @param params 请求数据实体的结构 通过这个结构可以统一点播与直播的播放入口
 * @param mode   播放的类型，确定需要解析的参数
 */
@MainThread
void playByMode(PolyvBaseVideoParams params, @PLVPlayOption.PlayMode int mode);
```

该方法在`PLVLivePlayerPresenter`做了封装，播放器mvp-view调用`PLVLivePlayerPresenter`的`startPlay`方法即可开始播放视频。

```
@Override
public void startPlay() {
    PolyvLiveVideoParams liveVideoParams = new PolyvLiveVideoParams(
            getConfig().getChannelId(),
            getConfig().getAccount().getUserId(),
            getConfig().getUser().getViewerId()
    );
    liveVideoParams.buildOptions(PolyvBaseVideoParams.WAIT_AD, true)
            .buildOptions(PolyvBaseVideoParams.MARQUEE, true)
            .buildOptions(PolyvBaseVideoParams.PARAMS2, getConfig().getUser().getViewerName());
    if (videoView != null) {
        videoView.playByMode(liveVideoParams, PLVPlayOption.PLAYMODE_LIVE);
    }
    startPlayProgressTimer();
}
```

3.5 播放控制

`PolyvLiveVideoView`提供了以下的播放控制方法。

```
//开始
void start();
//暂停
void pause();
//停止
void stopPlay();
```

以上方法也在`PLVLivePlayerPresenter`中做了封装，更多播放器相关方法的调用都可以参考`PLVLivePlayerPresenter`实现的接口`ILivePlayerPresenter`。