1 功能概述

该模块位于文件夹 PolyvLiveCommonModule/Modules/Player 下，包含直播、回放播放器及片头广告功能。该模块将sdk层的播放器 `PLVPlayer` 和它的两个子类直播播放器 `PLVLivePlayer` 、回放播放器 `PLVLivePlaybackPlayer` 和片头广告 `PLVAdvView` 的直接调用隔离开，并将多个场景中对sdk层播放器的共用代码抽离封装起来，使多场景层的逻辑变得更简洁，封装的presenter层负责sdk层播放器的控制以及回调播放器状态到多场景层。

2 核心类介绍

该模块 `PLVPlayerPresenter` 封装定义了可供外部调用的属性、方法和播放器改变代理回调 `PLVPlayerPresenterDelegate` ，对 `PLVPlayerPresenter` 代码示例可以在 `PLVLCMediaAreaView` 、`PLVECPlayerViewController` 中找到。

2.1 初始化API

```
#pragma mark 可配置项
/// delegate
@property (nonatomic, weak) id <PLVPlayerPresenterDelegate> delegate;

/// 创建 播放器
///
/// @param videoType 视频类型
- (instancetype)initWithVideoType:(PLVChannelVideoType)videoType;

/// 设置 承载播放器画面 的父视图
///
/// @param displayView 承载播放器画面的父视图
- (void)setupPlayerWithDisplayView:(UIView *)displayView;
```

2.2 代理回调

```
@protocol PLVPlayerPresenterDelegate <NSObject>

@optional
#pragma mark - [ 代理方法 ]
#pragma mark 通用
/// 播放器‘正在播放状态’发生改变
///
/// @param playerPresenter 播放器管理器
/// @param playing 是否正在播放
- (void)playerPresenter:(PLVPlayerPresenter *)playerPresenter playerPlayingStateDidChanged:(BOOL)playing;

/// 播放器 发生错误
///
/// @param playerPresenter 播放器管理器
/// @param errorMessage 错误描述
- (void)playerPresenter:(PLVPlayerPresenter *)playerPresenter loadPlayerFailureWithMessage:(NSString
*)errorMessage;

/// 播放器‘视频大小’发生改变
///
/// @param playerPresenter 播放器管理器
/// @param videoSize 当前视频大小
- (void)playerPresenter:(PLVPlayerPresenter *)playerPresenter videoSizeChange:(CGSize)videoSize;

/// 播放器‘SEI信息’发生改变
///
/// @param playerPresenter 播放器管理器
/// @param timeStamp 附带的时间戳信息
/// @param newTimeStamp 最新时间戳
- (void)playerPresenter:(PLVPlayerPresenter *)playerPresenter seiDidChange:(long)timeStamp newTimeStamp:
(long)newTimeStamp;

/// 播放器‘频道信息’发生改变
///
/// @param playerPresenter 播放器管理器
- (void)playerPresenter:(PLVPlayerPresenter *)playerPresenter channelInfoDidUpdated:(PLVChannelInfoModel
*)channelInfo;

#pragma mark 直播相关
/// 直播‘流状态’更新
///
/// @param playerPresenter 播放器管理器
/// @param newestStreamState 当前最新的 ’流状态’
/// @param streamStateDidChanged ’流状态‘ 相对上一次 是否发生变化
- (void)playerPresenter:(PLVPlayerPresenter *)playerPresenter streamStateUpdate:
(PLVChannelLiveStreamState)newestStreamState streamStateDidChanged:(BOOL)streamStateDidChanged;

/// 直播播放器‘码率可选项、当前码率、线路可选数、当前线路‘ 发生改变
///
/// @param playerPresenter 播放器管理器
- (void)playerPresenter:(PLVPlayerPresenter *)playerPresenter codeRateOptions:(NSArray <NSString *>
*)codeRateOptions currentCodeRate:(NSString *)currentCodeRate lineNum:(NSInteger)lineNum currentLineIndex:
(NSInteger)currentLineIndex;
```

```
/// 直播播放器 需获知外部 ‘当前是否正在连麦’
///
/// @note 此回调不保证在主线程触发
///
/// @param playerPresenter 播放器管理器
- (BOOL)playerPresenterGetInLinkMic:(PLVPlayerPresenter *)playerPresenter;

/// [无延迟直播] 无延迟直播 ‘开始结束状态’ 发生改变
///
/// @param playerPresenter 播放器管理器
/// @param noDelayLiveStart 当前最新的 ’无延迟直播开始结束状态’
/// @param noDelayLiveStartDidChanged ’无延迟直播开始结束状态‘ 相对上一次 是否发生变化
- (void)playerPresenter:(PLVPlayerPresenter *)playerPresenter noDelayLiveStartUpdate:(BOOL)noDelayLiveStart
noDelayLiveStartDidChanged:(BOOL)noDelayLiveStartDidChanged;

#pragma mark 非直播相关
/// 直播回放播放器 定时返回当前播放进度
///
/// @param playerPresenter 播放器管理器
/// @param downloadProgress 已缓存进度 (0.0 ~ 1.0)
/// @param playedProgress 已播放进度 (0.0 ~ 1.0)
/// @param playedTimeString 当前播放时间点字符串 (示例 "01:23")
/// @param durationTimeString 总时长字符串 (示例 "01:23")
- (void)playerPresenter:(PLVPlayerPresenter *)playerPresenter downloadProgress:(CGFloat)downloadProgress
playedProgress:(CGFloat)playedProgress playedTimeString:(NSString *)playedTimeString durationTimeString:
(NSString *)durationTimeString;

@end
```

2.3 播放器信息

```
#pragma mark 数据
/// 当期播放器的类型
@property (nonatomic, assign, readonly) PLVChannelVideoType currentVideoType;

/// 当前直播的 可选线路数量 (暂时仅直播支持)
@property (nonatomic, assign, readonly) NSInteger lineNum;

/// 当前直播的 线路下标 (由 0 起始；暂时仅直播支持)
@property (nonatomic, assign, readonly) NSInteger currentLineIndex;

/// 当前直播的 码率/清晰度 可选项字符串数组 (暂时仅直播支持)
@property (nonatomic, strong, readonly) NSArray <NSString *> * codeRateNamesOptions;

/// 当前直播的 码率/清晰度 (暂时仅直播支持)
@property (nonatomic, copy, readonly) NSString * currentCodeRate;

/// 当前直播回放的 播放时间点 (单位:秒；仅非直播场景下有值)
@property (nonatomic, readonly) NSTimeInterval currentPlaybackTime;

/// 当前直播回放的 视频总时长 (单位:秒；仅非直播场景下有值)
@property (nonatomic, readonly) NSTimeInterval duration;

/// 广告跳转链接
@property (nonatomic, readonly) NSString *advLinkUrl;

/// 广告播放状态
@property (nonatomic, readonly) BOOL advPlaying;

/// 广告播放状态
@property (nonatomic, assign) BOOL openAdv;

#pragma mark 状态
/// 该频道是否观看 ‘无延迟直播’
@property (nonatomic, assign, readonly) BOOL channelWatchNoDelay;

/// 无延迟直播的当前 ‘开始结束状态’
@property (nonatomic, assign, readonly) BOOL currentNoDelayLiveStart;

#pragma mark UI
/// 外部传入的，负责承载播放器画面的父视图
///
/// @note 需调用 [setupPlayerWithDisplayView:] 方法来设置
@property (nonatomic, weak, readonly) UIView * displayView;
```

2.3 播放控制

```
/// 清理播放器
- (void)cleanPlayer;

/// 恢复播放
///
/// @note 对于 直播 将重新加载直播;
///       对于 回放/点播 将恢复播放;
- (BOOL)resumePlay;

/// 暂停播放
- (BOOL)pausePlay;

/// 静音 播放器
- (void)mute;

/// 取消静音 播放器
- (void)cancelMute;

#pragma mark 直播相关
/// 切换 ’当前码率‘
///
/// @note 暂时仅直播支持清晰度切换，直播回放暂时不支持
///
/// @param codeRate 目标 码率/清晰度
- (void)switchLiveToCodeRate:(NSString *)codeRate;

/// 切换 ’当前线路‘
///
/// @param lineIndex 目标 线路 (由 0 起始；比如切至线路1，则传入 0)
- (void)switchLiveToLineIndex:(NSInteger)lineIndex;

/// 开启或关闭 音频模式
- (void)switchLiveToAudioMode:(BOOL)audioMode;

#pragma mark 非直播相关
/// 跳至某个时间点 (单位：秒)
- (void)seekLivePlaybackToTime:(NSTimeInterval)toTime;

/// 切换倍速 (范围值 0.0~2.0)
- (void)switchLivePlaybackSpeedRate:(CGFloat)toSpeed;
```

3 SDK核心类介绍

播放器的SDK核心类是播放器`PLVPlayer`和它的两个子类直播播放器`PLVLivePlayer`、回放播放器`PLVLivePlaybackPlayer`，在`PLVPlayerPresenter`中被使用。

3.1 初始化

根据业务初始化配置`PLVLivePlayer`、`PLVLivePlaybackPlayer`，使用示例：

```
- (void)setupPlayer{
    PLVRoomData * roomData = [PLVRoomDataManager sharedManager].roomData;
    NSString * userIdForAccount = [PLVLiveVideoConfig sharedInstance].userId;
    if (self.currentVideoType == PLVChannelVideoType_Live) { /// 直播
        self.livePlayer = [[PLVLivePlayer alloc] initWithPolyvAccountUserId:userIdForAccount
channelId:roomData.channelId];
        self.livePlayer.delegate = self;
        self.livePlayer.liveDelegate = self;
        self.livePlayer.channelWatchNoDelay = roomData.menuInfo.watchNoDelay;
        [self.livePlayer setupDisplaySuperview:self.playerBackgroundView];

        self.livePlayer.chaseFrame = NO;
        self.livePlayer.customParam = roomData.customParam;
    }else if (self.currentVideoType == PLVChannelVideoType_Playback){ /// 回放
        self.livePlaybackPlayer = [[PLVLivePlaybackPlayer alloc] initWithPolyvAccountUserId:userIdForAccount
channelId:roomData.channelId vodId:roomData.vid vodList:NO];
        self.livePlaybackPlayer.delegate = self;
        self.livePlaybackPlayer.livePlaybackDelegate = self;
        [self.livePlaybackPlayer setupDisplaySuperview:self.playerBackgroundView];

        self.livePlayer.customParam = roomData.customParam;
    }
}
```

3.2 设置代理回调

`PLVPlayer`、`PLVLivePlayer`、`PLVLivePlaybackPlayer`提供很多设置代理以回调播放器相关的信息。

- `PLVPlayer`代理`PLVPlayerDelegate`

```
@protocol PLVPlayerDelegate <NSObject>

@optional
/// 播放器加载前，回调options配置对象
///
/// @note 播放器在加载前，将触发此回调，并附带 PLVOptions，有自定义配置需求时，可对此对象进行参数设置。
///       当集成的是 PolyvIJKMediaFramework 时，请按 PLVIJKFFOptions 来处理 options；
///       当集成的是 IJKMediaFramework 时，请按 IJKFFOptions 来处理 options；
///
/// @param player 播放器对象
/// @param options 播放配置对象
- (PLVOptions *)plvPlayer:(PLVPlayer *)player playerWillLoad:(PLVPlayerMainSubType)mainSubType withOptions:
(PLVOptions *)options;

/// 播放器 已准备好播放
///
/// @param player 播放器对象
/// @param mainSubType 播放器主副类型
- (void)plvPlayer:(PLVPlayer *)player playerIsPreparedToPlay:(PLVPlayerMainSubType)mainSubType;

/// 播放器 ’加载状态‘ 发生改变
///
/// @param player 播放器对象
/// @param mainSubType 播放器主副类型
- (void)plvPlayer:(PLVPlayer *)player playerLoadStateDidChange:(PLVPlayerMainSubType)mainSubType;

/// 播放器 ’播放状态‘ 发生改变
///
/// @param player 播放器对象
/// @param mainSubType 播放器主副类型
- (void)plvPlayer:(PLVPlayer *)player playerPlaybackStateDidChange:(PLVPlayerMainSubType)mainSubType;

/// 播放器 ’是否正在播放中‘状态 发生改变
///
/// @param player 播放器对象
/// @param mainSubType 播放器主副类型
/// @param playing 当前是否正在播放
- (void)plvPlayer:(PLVPlayer *)player playerPlayingStateDidChange:(PLVPlayerMainSubType)mainSubType playing:
(BOOL)playing;

/// 播放器 播放结束
///
/// @param player 播放器对象
/// @param mainSubType 播放器主副类型
/// @param finishReson 播放结束原因
- (void)plvPlayer:(PLVPlayer *)player playerPlaybackDidFinish:(PLVPlayerMainSubType)mainSubType finishReson:
(IJKMPMovieFinishReason)finishReson;

/// 播放器 已销毁
///
/// @param player 播放器对象
/// @param mainSubType 播放器主副类型
- (void)plvPlayer:(PLVPlayer *)player playerDidDestroyed:(PLVPlayerMainSubType)mainSubType;

/// 主播放器 ‘SEI信息’ 发生改变
```

```
///
/// @param player 播放器对象
/// @param timeStamp 附带的时间戳信息
- (void)plvPlayer:(PLVPlayer *)player playerSeiDidChanged:(long)timeStamp;

@end
```

- `PLVLivePlayer` 代理 `PLVPlayerDelegate`

```
@protocol PLVLivePlayerDelegate <NSObject>

@optional
/// 直播播放器 ‘加载状态’ 发生改变
///
/// @param livePlayer 直播播放器
/// @param loadState 当前加载状态
- (void)plvLivePlayer:(PLVLivePlayer *)livePlayer loadStateDidChanged:(PLVLivePlayerLoadState)loadState;

/// 直播播放器 ‘流状态’ 更新
///
/// @param livePlayer 直播播放器
/// @param newestStreamState 当前最新的 ’流状态’
/// @param streamStateDidChanged ’流状态‘ 相对上一次 是否发生变化
- (void)plvLivePlayer:(PLVLivePlayer *)livePlayer streamStateUpdate:(PLVChannelLiveStreamState)newestStreamState
streamStateDidChanged:(BOOL)streamStateDidChanged;

/// 直播播放器 发生错误
///
/// @param livePlayer 直播播放器
/// @param error 错误信息对象 (可能为nil; error.code 可详见 PLVFPlayErrorCodeGenerator.h)
- (void)plvLivePlayer:(PLVLivePlayer *)livePlayer loadMainPlayerFailureWithError:(NSError * _Nullable)error;

/// 直播播放器 ‘频道信息’ 发生改变
///
/// @param livePlayer 直播播放器
/// @param channelInfo 当前最新 ’频道信息‘ 对象
- (void)plvLivePlayer:(PLVLivePlayer *)livePlayer channelInfoDidUpdated:(PLVChannelInfoModel *)channelInfo;

/// 直播播放器 ‘码率可选项、当前码率、线路可选数、当前线路‘ 发生改变
///
/// @param livePlayer 直播播放器
/// @param codeRateOptions 码率/清晰度 可选项字符串数组
/// @param currentCodeRate 当前 码率/清晰度
/// @param lineNum 线路可选数
/// @param currentLineIndex 当前线路下标 (由 0 起始)
- (void)plvLivePlayer:(PLVLivePlayer *)livePlayer codeRateOptions:(NSArray <NSString *> *)codeRateOptions
currentCodeRate:(NSString *)currentCodeRate lineNum:(NSInteger)lineNum currentLineIndex:
(NSInteger)currentLineIndex;

/// 直播播放器 需获知外部 ‘当前是否正在连麦’
///
/// @note 此回调不保证在主线程触发
///
/// @param livePlayer 直播播放器
///
/// @return BOOL 需返回 ‘当前外部是否正在连麦’ 的状态布尔值
- (BOOL)plvLivePlayerGetInLinkMic:(PLVLivePlayer *)livePlayer;

/// 直播播放器 需展示暖场图片
///
/// @param livePlayer 直播播放器
/// @param show 是否展示 (YES:展示 NO:隐藏)
/// @param warmUpImageURLString 暖场图片链接地址
- (void)plvLivePlayer:(PLVLivePlayer *)livePlayer showWarmUpImage:(BOOL)show warmUpImageURLString:(NSString *
```

```
_Nullable)warmUpImageURLString;

@end
```

- `PLVLivePlaybackPlayer` 代理 `PLVPlayerDelegate`

```
@protocol PLVLivePlaybackPlayerDelegate <NSObject>

@optional
/// 直播回放播放器 发生错误
///
/// @param livePlaybackPlayer 直播回放播放器
/// @param error 错误信息对象 (可能为nil；error.code 可详见 PLVFPlayErrorCodeGenerator.h)
- (void)plvLivePlaybackPlayer:(PLVLivePlaybackPlayer *)livePlaybackPlayer loadMainPlayerFailureWithError:
(NSError * _Nullable)error;

/// 直播回放播放器 定时返回当前播放进度
///
/// @param livePlaybackPlayer 直播回放播放器
/// @param downloadProgress 已缓存进度 (0.0 ~ 1.0)
/// @param playedProgress 已播放进度 (0.0 ~ 1.0)
/// @param playedTimeString 当前播放时间点字符串 (示例 "01:23")
/// @param durationTimeString 总时长字符串 (示例 "01:23")
- (void)plvLivePlaybackPlayer:(PLVLivePlaybackPlayer *)livePlaybackPlayer downloadProgress:
(CGFloat)downloadProgress playedProgress:(CGFloat)playedProgress playedTimeString:(NSString *)playedTimeString
durationTimeString:(NSString *)durationTimeString;

/// 直播回放播放器‘频道信息’发生改变
///
/// @param livePlaybackPlayer 直播回放播放器
/// @param channelInfo 当前最新 ’频道信息‘ 对象
- (void)plvLivePlaybackPlayer:(PLVLivePlaybackPlayer *)livePlaybackPlayer channelInfoDidUpdated:
(PLVChannelInfoModel *)channelInfo;

@end
```

设置代理回调的示例可以在 `PLVPlayerPresenter` 中找到。

3.3 开始播放

创建播放器后需调用父类 `PLVPlayer` 方法 [setupDisplaySuperview:] 进行画面添加，才可正常显示播放内容，可参考3.1 初始化代码。

3.4 播放控制

`PLVPlayer`、`PLVLivePlayer`、`PLVLivePlaybackPlayer` 提供了播放器操作控制API。

- `PLVPlayer` 提供的播放控制方法：

```
/// 设置 承载播放器画面 的父视图
///
/// @param displayeSuperview 承载播放器画面的父视图
- (void)setupDisplaySuperview:(UIView *)displayeSuperview;

/// 清理 主播放器
- (void)clearMainPlayer;

/// 清理 副播放器
- (void)clearSubPlayer;

/// 清理 全部播放器
- (void)clearAllPlayer;

/// 暂停 主播放器 播放
- (void)pause;

/// 静音 主播放器
- (void)mute;

/// 取消静音 主播放器
- (void)cancelMute;

#pragma mark 非直播相关
/// 开始 主播放器 播放
- (void)play;

/// 主播放器 视频跳至某个时间点
- (void)seekToTime:(NSTimeInterval)toTime;

/// 主播放器 切换倍速 (范围值 0.0~2.0)
- (void)switchSpeedRate:(CGFloat)toSpeed;
```

- `PLVLivePlayer` 提供的播放控制方法：

```
/// 重新加载直播 （或暂停后的恢复直播）
- (void)reloadLivePlayer;

/// 切换 ’当前线路‘ 或 ’当前码率‘
///
/// @note 传入的 targetLineIndex、targetCodeRate 不一定就是最终的选定值。因为内部将判断传入值是否合法；
///       最终播放的选定值，可读取 channelInfo 的 currentLineIndex、currentDefinition 属性；
///
/// @param targetLineIndex 目标线路 (由 0 起始；比如切至线路1，则传入 0)
/// @param targetCodeRate 目标码率/清晰度 (对应字符串，可在代理方法
[plvLivePlayer:codeRateOptions:currentCodeRate:lineNum:currentLineIndex:] 中取得)
- (void)switchToLineIndex:(NSInteger)targetLineIndex codeRate:(NSString *)targetCodeRate;

/// 开启或退出 音频模式
///
/// @param audioMode 是否开启音频模式
- (void)switchToAudioMode:(BOOL)audioMode;
```

- `PLVLivePlaybackPlayer` 提供的播放控制方法：

```
/// 跳至某个时间点（单位：秒）
- (void)seekLivePlaybackToTime:(NSTimeInterval)toTime;

/// 切换倍速（范围值 0.0~2.0）
- (void)switchLivePlaybackSpeedRate:(CGFloat)toSpeed;
```

4 片头广告类介绍

片头广告播放器模块位于 `PolyvLiveCommonModule` 模块的 `GeneralUI` 的 `PLVAdvView` 目录下， `PLVAdvView` 是播放后台配置的片头广告视频或图片，该类在 `PLVPlayerPresenter` 中被使用。此功能在直播拉到流后、回放开始下载后和在播放广告中断网后重新联网情况，开始播放。

4.1 初始化

初始化配置广告信息，使用示例：

```
PLVPlayerPresenter.m

- (void)showAdv {
    /** 不显示片头广告(此开关应对云课堂暂时不显示片头广告情况，后面云课程支持片头广告，需去掉openAdv) */
    if (! self.openAdv) {
        return;
    }

    PLVRoomData *roomData = [PLVRoomDataManager sharedManager].roomData;

    // 存在片头广告
    if ((roomData.liveState == PLVChannelLiveStreamState_Live ||
         roomData.videoType == PLVChannelVideoType_Playback) && // 直播开播了或者是点播
        ([PLVFdUtil checkStringUseable:roomData.channelInfo.advertImage] ||
         [PLVFdUtil checkStringUseable:roomData.channelInfo.advertFlvUrl])) { // 存在片头图片或视频
        [self pausePlay];

        if (! self.advView) { // 广告播放器初始化
            _advView = [[PLVAdvView alloc] init];
            self.advView.delegate = self;
            [self.advView setupDisplaySuperview:self.playerBackgroundView];
        }

        if ([PLVFdUtil checkStringUseable:roomData.channelInfo.advertImage]) {
            [self.advView showImageWithUrl:roomData.channelInfo.advertImage
time:roomData.channelInfo.advertDuration];
        } else {
            [self.advView showVideoWithUrl:roomData.channelInfo.advertFlvUrl
time:roomData.channelInfo.advertDuration];
        }
    }
}
```

4.2 代理回调

片头广告代理 `PLVAdvViewDelegate` 定义在 `PLVAdvView.h` 中，提供片头广告状态信息。

```
PLVAdvView.h

/// 播放器态类型
typedef NS_ENUM(NSInteger, PLVAdvViewStatus) {
    PLVAdvViewStatusUnkown = 0, // 未知类型（初始状态）
    PLVAdvViewStatusPlay, // 展示中
    PLVAdvViewStatusFinish, // 展示完成
};

/// 广告状态回调
- (void)advView:(PLVAdvView *)advView status:(PLVAdvViewStatus)status;
```

设置代理回调的示例可以在 `PLVPlayerPresenter` 中找到。

4.3 片头播放信息、控制

`PLVAdvView`提供的播放状态和控制方法：

```
@interface PLVAdvView : UIView

@property (nonatomic, assign, readonly) BOOL playing; // 是否正在播放中
@property (nonatomic, assign, readonly) PLVAdvViewStatus status; // 播放器状态
@property (nonatomic, weak) id<PLVAdvViewDelegate> delegate;

/// 设置广告外部容器
- (void)setupDisplaySuperview:(UIView *)displayeSuperview;

/// 展示图片url
- (void)showImageWithUrl:(NSString *)url time:(NSInteger)time;

/// 播放url
- (void)showVideoWithUrl:(NSString *)url time:(NSInteger)time;

/// 销毁 广告
- (void)distroy;

@end
```