iOS 回放 Core SDK

git 链接
最新版 changelog

主要功能

直播视频回放核心 SDK 提供在线及本地直播视频的回放功能(不包含 UI)，可以完整重现直播时候的ppt,画笔, 以及老师视频. 支持回放下载. 功能包括：

模块	功能	对应文件
教室管理	在线/本地的回放教室进入/退出的监听	BJPRoom.h
	聊天信息的监听	
	播放本地回放的时候获取权限	
视频管理	记忆播放	BJPPlaybackVM.h
	监听回放视频状态	
	视频控制(播放/暂停/停止/拖拽/切换清晰度/切换倍速)	
下载和在线回放信息	下载管理	PMDownloadManager.h
	回放视频信息	BJPlayerManager.h

另外还提供一个包含UI的回放UI SDK, 包含整套的直播回放UI, 集成工作量比较少, 便于快速开发.

Demo体验

Demo建议使用 BJPlaybackUI

- demo 运行成功后将进入如下登录界面. 需要输入教室ID 和 sessionId(可以为空)及token才能进入教室。这些参数由客户服务端访问百家云服务端API获取, 然后传给移动端.

- 进入后如下图, 支持倍速, 切换清晰度

集成SDK

使用CocoaPods方式集成

- Podfile 中设置 source

```
source 'https://github.com/CocoaPods/Specs.git'
source 'http://git.baijiashilian.com/open-ios/specs.git '
```

- 配置ATS. 因为视频url是http的, 需要在info.plist里面增加 NSAllowsArbitraryLoads = true
- 配置"Privacy - Media Library Usage Description" 媒体库权限
- Podfile 中引入 `BJPlaybackCore`, 默认动态集成ijk播放器

```
pod 'BJPlaybackCore', '~> 1.0'

  script_phase \
  :name => '[BJLiveCore] Embed Frameworks',
  :script => 'Pods/BJLiveCore/frameworks/EmbedFrameworks.sh',
  :execution_position => :after_compile

  script_phase \
  :name => '[BJVideoPlayer] Embed Frameworks',
  :script => 'Pods/BJVideoPlayer/BJVideoPlayer/EmbedFrameworks.sh',
  :execution_position => :after_compile

  script_phase \
  :name => '[BJLiveBase] Clear Archs From Frameworks',
  :script => 'Pods/BJLiveBase/script/ClearArchsFromFrameworks.sh "BJHLMediaPlayer.framework" "BJYIJKMediaFramework.framework"',
  :execution_position => :after_compile
```

1. 引入头文件

```
#import <BJPlaybackCore/BJPlaybackCore.h>
```

2. 创建、进入回放

创建、进入教室的整体流程如下:

- 设置专属域名前缀
- 定义一个BJPRoom的属性room,用于管理回放房间
- 使用回放相关信息将room属性实例化
- 为回放的进入, 退出, 聊天消息更新等事件添加监听
- 调用 room 定义的 enter 进入教室, 监听到进入房间成功之后, 即可开始观看回放.

2.1 设置专属域名前缀

- 1.4.5 之后的版本支持设置专属域名前缀，需要在创建 BJPRoom 实例之前设置。例如专属域名为 `demo123.at.baijiayun.com`，则前缀为 `demo123`，参考 专属域名说明。

```
// 以demo123.at.baijiayun.com为例，domainPrefix = @“demo123”
NSString *domainPrefix = @"yourDomainPrefix";
[BJPRoom setPrivateDomainPrefix:domainPrefix];
```

2.2 定义回放房间

```
@property (nonatomic) BJPRoom *room;
```

2.3 创建教室:分为在线回放和本地回放

- 创建用于在线回放的房间

```
/**
 创建在线回放的room
 创建在线视频, 参数不可传空
 创建本地room的话, 两个参数传nil

 @param classId classId
 @param sessionId sessionId, 长期房间回放参数. 如果classId对应的课程不是长期房间,可不传;
                  如果classId对应的课程是长期房间, 不传则默认返回长期房间的第一个课程
 @param token token
 @param userName 第三方用户名, 不上报则传nil
 @param userNumber 第三方用户number号, 不上报则传0
 @return room
 */
+ (instancetype)onlineVideoCreateRoomWithClassId:(NSString *)classId
                                       sessionId:(nullable NSString *)sessionId
                                           token:(NSString *)token
                                        userName:(nullable NSString *)userName
                                      userNumber:(NSInteger)userNumber;
```

- 创建用于本地回放的房间

```
/**
 创建本地视频  进入房间

 @param videoPath 本地视频的路径
 @param signalPath 本地信令, 根据isZip的值, 传信令文件的路径
 @param isZip 所传信令文件是否为压缩文件, YES:传压缩的信令文件的路径
 NO: 传解压后的信令文件, 且所传的路径的下一级目录即为all.json等数据
 @param handle 用于在iOS10以上的系统用户授权进入本地资料库, 如果版本低iOS10,不需要授权,
 即status = BJPMediaLibraryAuthorizationStatusAuthorized
 */
+ (instancetype)localVideoCreatRoomWithVideoPath:(NSString *)videoPath
                                      signalPath:(NSString *)signalPath
                                      definition:(PMVideoDefinitionType)definition
                                           isZip:(BOOL)isZip
                                          status:(void (^)(BJPMediaLibraryAuthorizationStatus status))handle;
```

- 上报用户标识符：根据实际需要选择是否上报

```
// 如果用户标识符 _userInfo 存在，上报该标识符
[self.room.playbackVM setUserInfo:_userInfo];
```

2.4 准备进入回放: 添加方法监听

- 监听进入、退出房间事件

```objc
// 监听刚进入教室成功/失败
@weakify(self);
[self bjl_observe:BJLMakeMethod(self.room, roomDidEnterWithError:)
        observer:^BOOL(BJLError *error) {
            @strongify(self);
            // error 为 nil 表示正常进入，否则输出错误信息
            if (error) {
                NSLog(@"roomDidEnterWithError ==> error = %@", error);
            }
            return YES;
        }];
```

```objc
// 即将退出房间
@weakify(self);
[self bjl_observe:BJLMakeMethod(self.room, roomWillExitWithError:)
        observer:^BOOL(BJLError *error) {
            @strongify(self);
            // error 为 nil 表示主动退出，否则输出错误信息
            if (error) {
                NSLog(@"roomWillExitWithError: ==> error = %@", error);
            }
            return YES;
        }];
```

```objc
// 退出房间
@weakify(self);
[self bjl_observe:BJLMakeMethod(self.room, roomDidExitWithError:)
        observer:^BOOL(BJLError *error) {
            @strongify(self);
            // error 为 nil 表示主动退出，否则输出错误信息
            if (error) {
                NSLog(@"roomDidExitWithError ==> error = %@", error);
            }
            return YES;
        }];
```

2.5 进出回放房间

```objc
// 进入回放
[self.room enter];

// 退出回放
[self.room exit];
```

3. 音视频管理

回放的音视频管理, 参考BJPRoom的属性`playbackVM`

创建房间之后，回放管理类 `BJPPlaybackVM` 的实例 `playbackVM` 也被初始化，可使用它进行回放管理

```objc
/** 播放器的view */
@property (nonatomic, readonly) UIView *playView;
```

播放信息可通过回放管理类的实例 `playbackVM` 直接获取，也可以通过 KVO 监听

- 通过 `playbackVM` 获取

```
// 当前播放状态：枚举类型
PMPlayState playState = self.room.playbackVM.playbackState;

// 当前视频信息，包含视频 ID、视频名称、片头片尾、清晰度列表、选集信息列表等
PMVideoInfoModel *videoInfo = self.room.playbackVM.videoInfoModel;
NSArray <PMVideoDefinitionInfoModel*> *definitionList = videoInfo.definitionList; //支持的清晰度列表

// 当前播放清晰度，包含清晰度类型、CDN 列表等
PPMVideoDefinitionInfoModel *currDefinitionInfoModel;
NSArray<PMVideoCDNInfoModel *> *cdnList = definitionInfo.cdnList; // CDN列表

// 当前播放的 CDN
PMVideoCDNInfoModel *cdnInfo = self.room.playbackVM.currCDNInfoModel;
```

- 通过 KVO 监听

```
[self bjl_kvo:BJLMakeProperty(self.room.playbackVM, currentTime)
    observer:^BOOL(NSNumber * _Nullable old, NSNumber * _Nullable now) {
        NSLog(@"currentTime: old = %@; now = %@", old, now);
//更新进度条
        return YES;
    }];
```

3.1 视频通知类型

```
// 即将播放的通知
PMPlayerWillPlayNotification

// 加载状态改变的通知
PKMoviePlayerLoadStateDidChangeNotification

// 播放状态改变的通知
PKMoviePlayerPlaybackStateDidChangeNotification

// 播放结束的通知
PKMoviePlayerPlaybackDidFinishNotification
```

- 以监听播放结束的通知为例

```
// 添加通知
[[NSNotificationCenter defaultCenter] addObserver:self
                                         selector:@selector(playbackFinish:)
                                             name:PKMoviePlayerPlaybackDidFinishNotification
                                           object:nil];
```

```objc
// 响应方法
- (void)playbackFinish:(NSNotification *)noti {
    // 获取结束原因
    PKMovieFinishReason reason = [[noti.userInfo objectForKey:PKMoviePlayerPlaybackDidFinishReasonUserInfoKey] integerValue];
    // 获取错误信息
    NSString *error = [noti.userInfo objectForKey:@"error"];
    switch (reason) {
        case PKMovieFinishReasonPlaybackEnded: // 回放结束
            [MBProgressHUD bjp_showMessageThenHide:@"Playback Ended ==> video finish" toView:self.view onHide:nil];
            break;

        case PKMovieFinishReasonPlaybackError: // 回放出错
            [MBProgressHUD bjp_showMessageThenHide:[NSString stringWithFormat:@"playback error ==> video finish, error:%@", error] toView:self.view onHide:nil];
            break;

        case PKMovieFinishReasonUserExited: // 用户退出
            [MBProgressHUD bjp_showMessageThenHide:@"User Exited ==> video finish" toView:self.view onHide:nil];
            break;

        default:
            break;
    }
}
```

3.2 视频播放控制

```objc
// 播放视频
[self.room.playbackVM playerPlay];

// 暂停播放
[self.room.playbackVM playerPause];

// 停止播放
[self.room.playbackVM playerStop];
```

3.3 跳转到指定时刻

```objc
// 以5.0秒为例
[self.room.playbackVM playerSeekToTime:5.0];
```

3.4 改变播放倍速

```objc
// 以2.0倍为例
[self.room.playbackVM changeRate:2.0];
```

3.5 切换清晰度

```
/** 切换换清晰度
typedef NS_ENUM(NSInteger, PMVideoDefinitionType){
    DT_Unknown  = -1, //未知
    DT_LOW      = 0, //标清
    DT_HIGH     = 1, //高清
    DT_SUPPERHD = 2, //超清
    DT_720p     = 3, //720p
    DT_1080p    = 4, //1080p
};
*/
[self.room.playbackVM changeDefinition:DT_HIGH];
```

4. 回放协议

协议为 `BJPMProtocol`

- 设置当前 viewController 为代理

```
self.room.playbackVM.playerControl.delegate = self;
```

- 代理方法

```
// 播放过程中出错（@required）
- (void)videoplayer:(BJPlayerManager *)playerManager throwPlayError:(NSError *)error {
    NSLog(@"video play ended with error: %@", error);
}
```

5. 显示 PPT、画笔

```
/** 课件显示 */
@property (nonatomic, readonly, nullable) UIViewController<BJLSlideshowUI> *slideshowViewController;
```

- 添加 PPT、画笔视图

```
// 将 BJPRoom 的课件视图添加到当前 viewController 的对应视图
UIView *roomSlideshowView = self.room.slideshowViewController.view;
[self.slideshowView addSubview:roomSlideshowView];
[roomSlideshowView mas_makeConstraints:^(MASConstraintMaker *make) {
    make.edges.equalTo(self.slideshowView);
}];
```

- 设置显示模式

```
/** 设置显示模式
 BJLContentMode_scaleAspectFit 完整
 BJLContentMode_scaleAspectFill 铺满
 */
self.room.slideshowViewController.contentMode = BJLContentMode_scaleAspectFit;
```

6. 获取聊天消息列表

- 显示聊天的UI界面是需要重点优化的，优化方式可以使用高度缓存，手动计算高度，不使用自动布局等方式来优化，聊天界面一直占用主线程会导致音视频和课件不同步，界面一直卡顿等问题。如果成功优化之后依旧存在卡顿，可以使用调试工具针对性能消耗较大的功能进行针对性的优化。

1.3.x版本

- SDK 传出来的聊天信息列表是增量的, 需要创建一个可变数组(比如：chatList)来接收消息, 同时需要监听'didReceiveMessageList:'方法

```
// 接收增量消息
@weakify(self);
[self bjl_observe:BJLMakeMethod(self.room, didReceiveMessageList:)
        observer:^BOOL(NSArray<BJPMessage *> *messageArray){
            @strongify(self);
            if (messageArray.count > 0) {
                for (BJPMessage *msg in messageArray) {
                    [self.chatCtrl.chatList addObject:msg];
                }
            }
            [self.chatCtrl.tableView reloadData];
            return YES;
        }];
```

- 用户回退视频时，需要将当前消息列表清空，重新添加增量数组

```
// 监听 PKMoviePlayerPlaybackStateDidChangeNotification,
// 当 state == PKMoviePlaybackStateSeekingBackward时, 清空这个可变数组
- (void)playStatusChange:(NSNotification *)noti {
    if (self.room.playbackVM.playbackState == PKMoviePlaybackStateSeekingBackward) {
        [self.chatCtrl.chatList removeAllObjects];
        [self.chatCtrl.tableView reloadData];
    }
}
```

1.4.x版本

- SDK将消息的增量更新和覆盖更新拆分成了二个不同的监听，上层维护一个消息列表，收到消息的覆盖更新的时候需要清空消息列表，然后添加到消息数组；收到消息的增量更新的时候，直接向消息列表中添加即可。

```
// 覆盖更新
    [self bjl_observe:BJLMakeMethod(self.room, didReceiveMessageList:)
            observer:^BOOL(NSArray<BJPMessage *> *messageArray){
                @YPStrongObj(self);
                dispatch_async(dispatch_get_main_queue(), ^{
                    [self.messageViewContrller.allMessages removeAllObjects];
                    if (messageArray.count > 0) {
                        for (BJPMessage *msg in messageArray) {
                            [self.messageViewContrller.allMessages addObject:msg];
                        }
                    }
                    [self.messageViewContrller.tableView reloadData];
                    [self.messageViewContrller scrollToTheEndTableView];
                });
                return YES;
            }];
```

```
[self bjl_observe:BJLMakeMethod(self.room, didReceiveNewMessageList:)
        observer:^BOOL(NSArray<BJPMessage *> *messageArray){
            dispatch_async(dispatch_get_main_queue(), ^{
                if (messageArray.count > 0) {
                    for (BJPMessage *msg in messageArray) {
                        [self.messageViewContrller.allMessages addObject:msg];
                    }
                    [self.messageViewContrller.tableView reloadData];
                    [self.messageViewContrller scrollToTheEndTableView];
                }
            });
            return YES;
        }];
```

7. 下载

7.1 初始设置

- 注意：下载前需要先设置根路径，然后调用单例方法，否则代码抛出异常

```
/* 下载根路径, folder需要是一个 存在的 文件夹, 如果不存在, 需要上层创建 */
NSString *path = @"xxx/folder";
[PMDownloadManager downloadManagerWithRootPath:path];
```

7.2 下载方法

参考头文件:PMDownloadManager.h

```
//回放下载
[[PMDownloadManager downloadManager] addDownloadWithClass:classID seesionID:sessionID token:token definionArray:defiArr showFileName:showname creatTime:@"creatTime"];
```

需要注意的是,由于我们使用了ID+token的视频下载方式, 因此是存在token过期的现象, 发生token失效或者下载url失效的问题时 , 会向上层抛出BJPMErrorCodeDownloadInvalid (1010)的错误码,此时需要上层重新获取token,并调用下面的方法来重新设置token

```
/**
 下载中的任务发生了下载的url失效的错误,需要调用此方法,内部重新请求下载地址

@param downloader 下载的任务
@param token 新的token
*/
- (void)resetDownloadWithDownloader:(PMDownloader *)downloader token:(NSString *)token;
```

7.3 主要属性

```
/** 下载事件相关的代理 */
@property (nonatomic, weak  ) id<PMDownloadDelegate> downloadDelegate;

//下载过程中的数据 和 已经完成的数据
@property (nonatomic, readonly) NSArray <PMDownloader *> *downloadingList;
@property (nonatomic, readonly) NSArray <PMDownloadModel *> *finishedList;
```

7.4 下载代理方法

```
- (void)startDownload:(PMDownloader *)downloader;//开始下载
- (void)updateProgress:(PMDownloader *)downloader;//获取下载进度
- (void)finishedDownload:(PMDownloader *)downloader;//下载完成

/**
 下载时的错误回调, 包括开始下载之前和下载过程中的错误

 @param downloader 下载过程中的下载器实例, 可能为空
 @param beforeDownloadModel 下载开始前的错误model, 可能为空
 */
- (void)downloadFail:(nullable PMDownloader *)downloader beforeDownloadError:(nullable PMBeforeDownloadModel *)beforeDownloadModel;
```

7.5 分账户下载

- 在当前账户登录成功后, 首先释放当前的下载管理的单例

```
[PMDownloadManager downloadManagerDestory];
```

- 设置根路径, 如上, 传进来的文件夹以当前userId为命名的文件夹
- 调用 `+downloadManager`

7.6 下载中的错误回调方法处理

详细参考demo中的对(void)downloadFail:(nullable PMDownloader *)downloader beforeDownloadError:(nullable PMBeforeDownloadModel *)beforeDownloadModel;代理方法的处理

8. 错误码

```
    BJPMErrorCodeLoading          = 1000,    // 视频加载中
    BJPMErrorCodeLoadingEnd       = 1001,    // 视频加载完成
    BJPMErrorCodeParse            = 1002,    // 视频播放错误
    BJPMErrorCodeNetwork          = 1003,    // 网络错误, 没有网络或是未知网络
    BJPMErrorCodeWWAN             = 1004,    // 非WIFI环境
    BJPMErrorCodeWIFI             = 1005,    // wifi
    BJPMErrorCodeServer           = 1006,    // 请求服务端返回错误
    BJPMErrorCodeApp              = 1007,    // app端调用错误
    BJPMErrorCodeDownloadInvalid  = 1010,    // 视频的url失效
    BJPMErrorCodeMissFile         = 1011,    // 下载的文件丢失
```

9. 回放加密

- 回放加密需要使用1.4.x之后的SDK版本
- 只有ijk播放器支持播放加密视频
- 播放, 下载加密视频

```
BOOL isEncrypt = YES;
BJPlayerManagerType type = BJPlayerManagerType_IJKPlayer;
// 在线回放
    BJPRoom *room = [BJPRoom onlineVideoCreateRoomWithClassId:classId
                                                    sessionId:sessionId
                                                        token:token
                                                     userName:userName
                                                   userNumber:userNumber
                                                  use_encrypt:isEncrypt
                                                PlayerManagerType:type];

// 本地回放
 BJPRoom *room = [BJPRoom localVideoCreatRoomWithVideoPath:videoPath
                                                    signalPath:signalPath
                                                    definition:definition
                                                         isZip:isZip
                                                PlayerManagerType:type];

 // 下载回放
[[PMDownloadManager downloadManager] addDownloadWithClass:classID
                                                    seesionID:sessionID
                                                        token:token
                                                  definionArray:array
                                                   showFileName:showFileName
                                                     creatTime:creatTime
                                                   need_encrypt: isEncrypt];
```

10. 纯音频

- 目前纯音频需要使用1.4.5之后的版本
- 回放默认转码了纯音频
- 回放和点播在线播放的默认清晰度可以在后台配置清晰度列表, SDK根据配置的清晰度优先级列表来匹配第一个清晰度, 如果后台配置了纯音频最高优先级, 就可以进入回放和点播的时候直接播放纯音频, 否则就通过改变清晰度的方式播放纯音频。
- 视频清晰度 PMVideoDefinitionType 增加纯音频枚举

```
typedef NS_ENUM(NSInteger, PMVideoDefinitionType){
    DT_Audio    = -2,   // 音频
    DT_Unknown  = -1,   // 未知
    DT_LOW      = 0,    // 标清
    DT_HIGH     = 1,    // 高清
    DT_SUPPERHD = 2,    // 超清
    DT_720p     = 3,    // 720p
    DT_1080p    = 4,    // 1080p
};
```

- 播放纯音频

```
// 播放音频
[self.room.playbackVM changeDefinition:DT_Audio];
```

- 下载纯音频, 同一个视频只能下载纯音频或其他的一种清晰度, 按照传入的清晰度列表匹配第一个满足条件的清

晰度

```
self.preferredDefinitionList = @[@(-2),@(0),@(1),@(2),@(3),@(4)];

[[PMDownloadManager downloadManager] addDownloadWithClass:classID seesionID:sessionID token:token definionArray:self.preferredDefinitionList showFileName:showname creatTime:@"creatTime"];
```