iOS 直播带货 UI SDK

git 链接：**https://git2.baijiashilian.com/open-ios/BaijiaYun**

功能简介

带货 UI SDK 在直播 Core SDK 的基础上提供了一个针对带货场景下的模板，包含直播间主播视频采集和聊天。购物车ui需要在自定义，可以参考demo。

1. 直播相关功能

参考 iOS 直播 Core SDK

2. UI

直播主界面	功能
采集视图	用于显示摄像头采集到的视频
播放视图	用于显示主播的视频
聊天视图	用于发送聊天消息、 显示直播间内的聊天消息列表
购物车弹窗	主播端和观众端的购物车弹窗， 需要用户自定义
送礼物功能	显示礼物动画，

	账户体系相关的需要App层实现
点赞功能	显示点赞动画, 赞的数量累加
分享功能	sdk只提供了ui, 回调了按钮的点击事件, 需要APP层集成分享的SDK已经完善APP id等

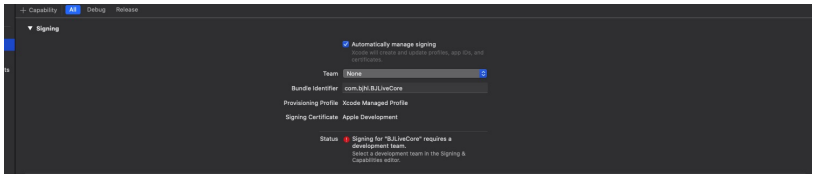
Demo

1. Demo 源文件

在 git 上下载最新的 SDK，demo 源文件在 `BJLSellUI/demo/BJLSellUI` 文件夹中。

2. Demo 编译、运行

- 在 demo 的工程目录下执行 `pod install`
- 使用 Xcode 打开 demo 文件夹下的 `BJLSellUI.xcworkspace` 文件
- 选择运行设备：**模拟器运行 demo 时无法采集音视频**；真机运行时，需要设置好 `development team`：



- 使用 Xcode 运行 demo

3. Demo 体验

- demo 运行成功后将进入如下登录界面，需要输入机构代码（参考 专属域名说明），六位参加码及用户名才能进入教

室。其中参加码通过使用 百家云后台 或者 API 创建一个教室获得，用户名可自定义。

- 教室加载成功之后进入后如下主界面，包含主播画面、聊天列表、购物车页面等部分。***其中购物车页面在带货UI SDK中提供了一个简单的示例页面 BJLSellListViewController，数据来源基于百家云管理后台，如果客户需要更丰富的自定义界面，需要自己实现。***

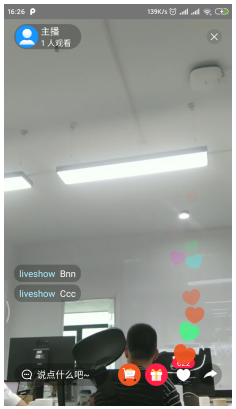
- 购物车


fy
0 人观看
28
共 7 件商品
商品 1 商品 1 商品 1 商品 1 商品 1 商品 1 商品 1 商品 1 商品 1
¥1.00
上架
查看
今天吃什么
¥6.00
下架
查看
今天不想吃萝卜白菜胡萝卜白萝卜芹菜翻鱼俩人吧, 不想吃了不
¥1.00
下架
查看
shdjkshjf
¥9999.00
下架
查看

- 礼物


主播
1 人观看
主播不在, 请稍等
选择礼物
已选
发送

- 点赞


主播
1 人观看
liveshow Bnn
liveshow Ccc
说点什么吧~

引入 SDK

1. 支持的设备和系统

- i. SDK 支持 **iOS 9.0 及以上的系统，iPhone、iPad 等设备，集成 1.0.0 或以上版本的 SDK 要求 Xcode 的版本至少为 9.0，集成 2.2.0 或以上版本的 SDK 要求 Xcode 的版本至少为 11.0**。如果进入教室时，`enterRoomFailureWithError:` 返回错误码 `BJLErrorCode_enterRoom_unsupportedDevice`，表示不支持当前设备。参考 iOS Device Summary。

- iPad: 1、2、3、4、mini 1 是 32-bit，其它都是 64-bit

- iPhone: 5、5C 之前的设备是 32-bit，5S 开始是 64-bit。
- iPod Touch: 1、2、3、4、5 是 32-bit，目前只有 6 是 64-bit。

2. 集成方式

SDK 依赖一些第三方库，建议使用 CocoaPods 方式引入：

- `Podfile` 设置 `source`

```
source 'https://github.com/CocoaPods/Specs.git'
source 'http://git2.baijiashilian.com/open-ios/specs.git'
```

- `Podfile` 中引入 `BJSellUI`

```
pod 'BaijiaYun/BJLSellUI'

# 用于动态引入 Framework，避免冲突问题
script_phase \
:name => '[BJLiveCore] Embed Frameworks',
:script => 'Pods/BJLiveCore/frameworks/EmbedFrameworks.sh
:execution_position => :after_compile
```

```
8.
9. # 用于清理动态引入的 Framework 用不到的架构，避免发布 AppStore 时发生错误，需要写在动态引入 Framework 的 script 之后
10. script_phase \
11. :name => '[BJLiveBase] Clear Archs From Frameworks',
12. :script => 'Pods/BJLiveBase/script/ClearArchsFromFrameworks "BJHLMediaPlayer.framework"',
13. :execution_position => :after_compile
```

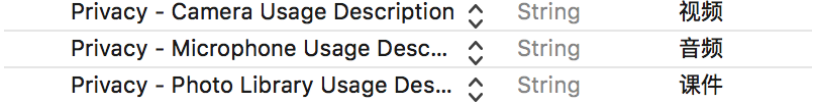
- 注意如果先集成了直播，后集成点播回放的时候，再引入点播回放 SDK 所需的 script 会导致工程 build phases 中顺序不对的情况，需要在 build phases 删除一下已经设置好的 script，重新 pod install 来保证直播和点播回放拷贝 framework 的前二个 script 先运行，清理 framework 的模拟器架构的 script 后运行。

- 在工程目录下执行 `pod install` ，初次集成需要执行 `pod update` 更新 CocoaPods 的索引。

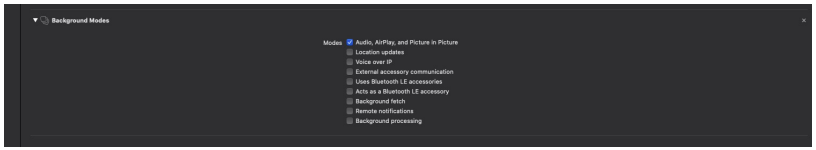
工程设置

- 隐私权限：在 `Info.plist` 中添加麦克风、摄像头、相册访问描述

1. Privacy - Microphone Usage Description 用于语音上课、发言
2. Privacy - Camera Usage Description 用于视频上课、发言，拍照传课件、聊天发图
3. Privacy - Photo Library Usage Description 用于上传课件、聊天发图



- 后台任务（**打开这一选项之后，在 App 提交审核时，强烈建议录制一个视频，说明 App 确实用到了后台播放，否则审核很有可能不通过**）：在 `Project > Target > Capabilities` 中打开 `Background Modes` 开关，选中 `Audio,Airplay, and Picture in Picture`



Hello World

可参考 demo 中的 BJLoginViewController

集成的整体流程如下：

- 在自己定义的相关文件中定义一个 `BJLSellViewController` 的属性 `roomViewController`，用于管理教室视图
- 使用教室相关信息（通过百家云后台或者 API 创建教室后获取）将 `roomViewController` 属性实例化
- 为教室的进入、退出等事件添加监听和相应的回调处理,回调处理可以根据自身需求进行自定义，为教室视图的管理做好准备
- 添加需要显示在直播间页面的购物车弹窗
- 将 `roomViewController` 的 `view` 显示出来（可使用 present、addChildViewController 等方式）

1. 准备创建教室

- 引入头文件

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

- 定义 `roomViewController`

```
@property (nonatomic) BJLSellViewController *roomViewController;
```

- 设置专属域名前缀，需要在创建 BJLRoom 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。

```
NSString *domainPrefix = @"yourDomainPrefix";
[BJLRoom setPrivateDomainPrefix:domainPrefix];
```

2. 创建、进入教室

可通过教室 ID 或参加码两种方式进行

- 教室 ID 方式：教室ID通过使用 百家云后台 或者 API 创建一个教室获得；签名参数通过 签名参数 sign 计算方法 获得

```
/**
教室ID方式
@param userNumber   用户编号，合作方账号体系下的用户ID号，必须是数字
@param userName     用户名
@param userAvatar   用户头像 URL(nullable)
@param userRole     用户角色:老师、学生等
@param roomID       教室 ID
@param groupID      分组 ID, 不分组传0
@param apiSign      签名
*/
// 创建用户实例
BJLUser *user = [self userWithNumber:number
                                name:name
                             groupID:groupID
```

```
                                avatar:avatar
                                  role:role];
/**
通过 ID 创建教室
@param roomID          教室 ID
@param user            用户
@param apiSign         API sign
@return                教室
*/
self.roomViewController = [BJLSellViewController
instanceWithID:roomID

apiSign:apiSign
                                                user:user];
```

- 参加码方式：参加码同样通过使用 百家云后台 或者 API 创建一个教室获得

```
/**
通过参加码创建教室
@param roomSecret      教室参加码
@param userName        用户名
@param userAvatar      用户头像 URL
@return                教室
*/
   self.roomViewController =
[BJLSellViewController
instanceWithSecret:roomSecret

userName:userName

userAvatar:userAvatar];
```

- 进入教室界面

```
[self presentViewController:self.roomViewController animated:YES completion:nil];
```

3. 监听进出教室事件

进出教室事件的监听可以即时获取教室的动态变化，便于添加自定义的回调处理。实现监听可以通过 `BJLSellViewControllerDelegate` 的代理回调 和 监听 `BJLSellViewController` 的方法调用的回调 两种方式实现。

3.1 BJLSellViewControllerDelegate

- 设置 `delegate`

```
self.roomViewController.delegate = self;
```

- `BJLSellViewControllerDelegate` 方法

```
/// 点击购物车按钮, 展示商品列表
/// @param sellViewController sellViewController
/// @param superview 商品列表的父view
/// @param closeCallback 关闭商品列表vc的回调
- (void)roomViewController:(BJLSellViewController *)sellViewController openListFromView:(UIView *)superview closeCallback:(BJLSCloseCallback)closeCallback {
    NSLog(@"[%@ %@ %@]", NSStringFromSelector(_cmd), roomViewController, superview);
}
```

```
/** 进入教室 - 成功 */
```

```
- (void)roomViewControllerEnterRoomSuccess:(BJLSellViewController *)roomViewController {
    NSLog(@"[%@ %@]", NSStringFromSelector(_cmd), roomViewController);
}
```

```
/** 进入教室 - 失败 */
- (void)roomViewController:(BJLSellViewController *)roomViewController
enterRoomFailureWithError:(BJLError *)error {
    NSLog(@"[%@ %@, %@]", NSStringFromSelector(_cmd), roomViewController, error);
}
```

```
/**
即将退出教室 - 正常/异常
正常退出 `error` 为 `nil`，否则为异常退出
参考 `BJLErrorCode` */
- (void)roomViewController:(BJLSellViewController *)roomViewController
willExitWithError:(nullable BJLError *)error {
    NSLog(@"[%@ %@, %@]", NSStringFromSelector(_cmd), roomViewController, error);
}
```

```
/**
 退出教室 - 正常/异常
 正常退出 `error` 为 `nil`，否则为异常退出
 参考 `BJLErrorCode` */
```

```
- (void)roomViewController:(BJLSellViewController *)roomViewController
        didExitWithError:(nullable BJLError *)error {
    NSLog(@"[%@ %@, %@]", NSStringFromSelector(_cmd), roomViewController, error);
}
```

3.2 监听 BJLScRoomViewController 的方法调用

- 进入教室成功

```
[self bjl_observe:BJLMakeMethod(self.roomViewController, roomViewControllerEnterRoomSuccess:)
        observer:^BOOL(BJLSellViewController *roomVC) {
NSLog(@"roomViewControllerEnterRoomSuccess");
            return YES;
        }];
```

- 进入教室失败

```
[self bjl_observe:BJLMakeMethod(self.roomViewController, roomViewController:enterRoomFailureWithError:)
        observer:^BOOL(BJLSellViewController *roomVC, BJLError *error) {
NSLog(@"roomViewControllerEnterRoomFailureWith error);
            return YES;
        }];
```

- 即将退出教室

```
[self bjl_observe:BJLMakeMethod(roomViewController, roomViewController:willExitWithError:)
        observer:^BOOL(BJLSellViewController *roomVC, BJLError *error) {
            if (error) {
                NSLog(@"roomViewControllerWillExitWithError:%@", error);
            }
            else {
                NSLog(@"roomViewControllerWillExit");
            }
            return YES;
        }];
```

- 退出教室

```
[self bjl_observe:BJLMakeMethod(roomViewController, roomViewController:didExitWithError:)
        observer:^BOOL(BJLSellViewController *roomVC, BJLError *error) {
            if (error) {
                NSLog(@"roomViewControllerDidExitWithError:%@", error);
            }
            else {
                NSLog(@"roomViewControllerDidExit");
            }
```

```
        return YES;
    }];
```

4. 添加自定义购物车的列表控制器

```
!!!: 需要先设置代理
```

```
/// 点击购物车按钮, 展示商品列表
/// @param sellViewController sellViewController
/// @param superview 商品列表的父view
/// @param closeCallback 关闭商品列表vc的回调
- (void)roomViewController:(BJLSellViewController *)sellViewController openListFromView:(UIView *)superview closeCallback:(BJLSCloseCallback)closeCallback {
    if (!self.sellListViewController) {
        self.sellListViewController = [[BJLSellListViewController alloc] initWithRoom:sellViewController.room];
    }

    [sellViewController bjl_addChildViewController:self.sellListViewController superview:superview];
    [self.sellListViewController.view bjl_makeConstraints:^(BJLConstraintMaker * _Nonnull make) {
        make.left.right.bottom.equalTo(superview);

        make.height.equalTo(superview).multipliedBy(0.5);
    }];
}
```

5. 退出教室

```
1. [self.roomViewController exit];
```

版本升级

版本号格式为 `大版本.中版本.小版本[-alpha(测试版本)/beta(预览版本)]` ：

- 测试版本和预览版本可能很不稳定，请勿随意尝试；
- 小版本升级只改 BUG、UI 样式优化，不会影响功能；
- 中版本升级、修改功能，更新 UI 风格、布局，会新增 API、标记 API 即将废弃，但不会导致现有 API 不可用；
- 大版本任何变化都是有可能的；

首次集成建议选择最新正式版本(版本号中不带有 `alpha` 、 `beta` 字样)，**版本升级后请仔细阅读 ChangeLog**，指定版本的方式有一下几种：

- 固执型： `pod update` 时不会做任何升级，但可能无法享受到最新的 BUG 修复，建议用于 0.x 版本；

```
1. pod 'BaijiaYun/BJLSellUI', '1.0.0'
```

- 稳妥型(**推荐**)： `pod update` 时只会升级到更稳定的小版本，而不会升级中版本和大版本，不会影响功能和产品特性，升级后需要 **适当测试**；

```
1. pod 'BaijiaYun/BJLSellUI', '~> 1.0.0'
```

- 积极型： `pod update` 时会升级中版本，但不会升级大版本，及时优化，但不会导致编译出错不可用，升级后需要 **全面测试**；

```
1. pod 'BaijiaYun/BJLSellUI', '~> 1.0'
```

- 激进型(**不推荐**)：`pod update` 时会升级大版本，可能导致编译出错、必须调整代码，升级后需要 **严格测试**；

```
1. pod 'BaijiaYun/BJLSellUI'
```

附录

1. 变更记录

- ChangeLog

2. 源码集成方式参考 BJLiveUI 的源码集成文档

- 源码集成参考文档

下载为pdf格式