腾讯云音视频多人会话解决方案服务端

项目结构

```
server
├── README.md
├── app.js
├── controllers
|   ├── index.js
|   ├── get_im_login_info.js
|   ├── get_push_url.js
|   ├── create_room.js
|   ├── destroy_room.js
|   ├── add_pusher.js
|   ├── delete_pusher.js
|   ├── get_pushers.js
|   ├── pusher_heartbeat.js
|   └── get_room_list.js
├── logic
|   ├── im_mgr.js
|   ├── live_util.js
|   └── room_mgr.js
├── middlewares
|   ├── bodyparser.js
|   └── response.js
├── config.js
├── log.js
├── log_config.js
├── package.json
├── process.json
├── nodemon.json
├── qcloud.js
└── routes
    └── index.js
```

`app.js` 是 Demo 的主入口文件，Demo 使用 Koa 框架，在 `app.js` 创建一个 Koa 实例并响应请求。

`routes/index.js` 是 Demo 的路由定义文件

`controllers` 存放 Demo 所有业务逻辑的目录，`index.js` 不需要修改，他会动态的将 `controllers` 文件夹下的目录结构映射成 modules 的 Object，例如 Demo 中的目录将会被映射成如下的结构：

```
// index.js 输出
{
  get_im_login_info: require('get_im_login_info'),
  get_push_url: require('get_push_url'),
  create_room: require('create_room'),
  destroy_room: require('destroy_room'),
  add_pusher: require('add_pusher'),
  delete_pusher: require('delete_pusher'),
  get_pushers: require('get_pushers'),
  pusher_heartbeat: require('pusher_heartbeat'),
  get_room_list: require('get_room_list')
}
```

`qcloud.js` 导出了一个 SDK 的单例，包含了所有的 SDK 接口，之后使用的时候只需要 `require` 这个文件就行，无需重复初始化 SDK。

`config.js` 主要的配置如下：

```
{
  port: '5757',
  rootPathname: '',

  // 微信小程序 App ID
  appId: '',

  // 微信小程序 App Secret
  appSecret: '',

  // 是否使用腾讯云代理登录小程序
  useQcloudLogin: true,

  /**
   * MySQL 配置，用来存储 session 和用户信息
   * 若使用了腾讯云微信小程序解决方案
   * 开发环境下，MySQL 的初始密码为您的微信小程序 appid
   */
  mysql: {
    host: 'localhost',
    port: 3306,
    user: 'root',
    db: 'cAuth',
    pass: 'xxx',
    char: 'utf8mb4'
  },

  cos: {
    /**
```

```
      * 区域
      * 华北：cn-north
      * 华东：cn-east
      * 华南：cn-south
      * 西南：cn-southwest
      * 新加坡：sg
      * @see https://www.qcloud.com/document/product/436/6224
      */
    region: 'cn-south',
    // Bucket 名称
    fileBucket: 'wximg',
    // 文件夹
    uploadFolder: ''
  },

  /**
   * 需要开通云直播服务
   * 参考指引
@https://cloud.tencent.com/document/product/454/7953#1.-.E8.A7.86.E9.A2.91.
E7.9B.B4.E6.92.AD.EF.BC.88lvb.EF.BC.89
   * 有介绍bizid 和 pushSecretKey的获取方法。
   */
  live: {
    // 云直播 bizid
    bizid: 0,

    // 云直播 推流防盗链key
    pushSecretKey: '',

    // 云直播 推流有效期单位秒 默认7天
    validTime: 3600*24*7
  },

  /**
   * 需要开通云通信服务
   * 参考指引
@https://cloud.tencent.com/document/product/454/7953#3.-.E4.BA.91.E9.80.9A.
E8.AE.AF.E6.9C.8D.E5.8A.A1.EF.BC.88im.EF.BC.89
   * 有介绍appid 和 accType的获取方法。以及私钥文件的下载方法。
   */
  im: {
    // 云通信 sdkappid
    sdkAppID: 0,

    // 云通信 账号集成类型
    accountType: "",

    // 云通信 管理员账号
    administrator: "",
```

```
    // 云通信 派发usersig的RSA 私钥
    privateKey: ""
  },

  /**
   * 多人音视频房间相关参数
   */
  room: {
    // 房间容量上限
    maxMembers: 6,

    // 心跳超时 单位秒
    heartBeatTimeout: 20,

    // 空闲房间超时 房间创建后一直没有人进入，超过给定时间将会被后台回收，单位秒
    maxIdleDuration: 30
  },

  /**
   * 辅助功能 后台日志文件获取相关 当前后台服务的访问域名。
   */
  selfHost:"https://xxxxxxxx.qcloud.la",

  // 微信登录态有效期
  wxLoginExpires: 7200
}
```

`logic/im_mgr.js` 云通信相关的处理，主要功能有：

```
module.exports = {
  getSig,                         // 计算云通信 账号登录IM所需要的userSig票据
  createGroup,                    // 创建IM聊天室，通过云通信提供的服务端对接用的
RestFul API实现
  destroyGroup,                   // 销毁IM聊天室，通过云通信提供的服务端对接用的
RestFul API实现
  notifyPushersChange             // IM聊天室成员进入和退出系统消息通知，通过云通信提供
的服务端对接用的RestFul API实现
}
```

`logic/room_mgr.js` 实时音视频房间管理模块，负责视频房间的创建，销毁，增加成员，删除成员，获取房间列表，获取房间成员列表等功能函数；另外也负责房间成员的心跳检查，对超时的成员进行删除处理。

`logic/live_util.js` 云直播辅助函数，负责生成推流地址以及播放地址。外加一些用户ID分配和房间ID分配的功能函数。

`log.js` 后台日志模块，主要记录请求响应和错误两大类日志。请求响应日志按小时存储在`logs/response/`目录下，错误日志按小时存储在`logs/error/`目录下。最多存储7天日志。以上默认配置可以通过修改`log_config.js`来调整。 日志配置：

```
{
  appenders:
  {
    //错误日志
    errorLogger:{
      type: "dateFile",                    //日志类型
      filename: errorLogPath,              //日志输出位置
      alwaysIncludePattern: true,          //是否总是有后缀名
      pattern: "-yyyy-MM-dd-hh.log",       //后缀，每小时创建一个新的日志文件
      daysToKeep: 7                        //自定义属性，错误日志的根目录
    },
    //响应日志
    resLogger:{
      type: "dateFile",
      filename: responseLogPath,
      alwaysIncludePattern: true,
      pattern: "-yyyy-MM-dd-hh.log",
      daysToKeep: 7
    },
    //控制台输出
    consoleLogger:{
      type: "console"
    }
  },
  categories:                                  //设置logger名称对应的的日志等
级
  {
    default:{
      appenders: ['consoleLogger'],
      level:"info"
    },
    errorLogger:{
      appenders:["errorLogger"],
      level:"error"
    },

    resLogger:{
      appenders:["resLogger"],
      level:"info"
    }
  }
}
```