个人介绍

- 李远超 一个更加关注生活的程序员
- iOS高级开发工程师 有信网络
- 男 28岁 本科 4年工作经验 唐山
- 18501340650
- lych0317@126.com

工作履历

- 深圳市有信网络技术有限公司 iOS高级开发工程师 2016.3～现在
- 小叶子（北京）科技有限公司 iOS中级开发工程师 2013.10～2016.3
- 东软集团 iOS初级开发工程师 2012.2～2013.9

工程经验

- 组建、建设团队
- 制定分支管理策略
- 配置Jenkins实现可持续集成
- 任务划分、安排排期
- 规范代码风格、进行代码Review
- 组织分享、提升团队技术水平
- 与其他部门沟通、优化开发流程

项目经验

- 红豆Live
  这是一个新立项目，从框架搭建到后期版本迭代都由我负责，iOS团队共4个人。上线4个月，目前iOS端日活2w以上，错误率在0.04%以下。
  - 工程架构
  - 网络架构
  - 日志管理
  - 文档管理
  - 连麦功能
  - 性能调优
- 有信电话
  这是一个iOS版日活在75w以上的老项目，其中一个模块`主题`需要优化。

- 存在问题
  - 关于颜色读取的代码过于复杂，并且不够集中，修改起来牵涉的代码比较多
  - 使用通知的地方特别多，对性能消耗严重，代码可读性差，不便于维护
  - 主题资源配置复杂，给UI工程师造成了很大的工作量
- 优化成效
  - 针对颜色、图片的读取做了统一接口处理，方便维护
  - 改用代理方式，有针对性的通知相关页面更新UI，提高代码可读性
  - 和前端工程师配合开发了个小工具，UI工程师方便配置主题资源，自动生成配置文件
  - 明显减少了该模块的bug量

- The ONE 智能钢琴
- 最全钢琴谱
- 大商所

详细介绍

分支管理
鉴于咱们时间紧、任务重，经常出现多个版本同时进行的情况。现制定如下分支管理策略，欢迎各位提出更优的解决方案

- 分支列表
  `x.x.x`代表版本号，`xx`开发人员名称
  - `master`
  - `bugfix_x.x.x`
  - `xx_bugfix`
  - `dev_x.x.x`
  - `xx_dev_x.x.x`
  - `其他`
- 分支说明
  `master`、`bugfix_x.x.x`、`dev_x.x.x`为项目统一管理分支，由管理员负责
  `xx_bugfix`、`xx_dev_x.x.x`、`其他`为开发人员管理分支，由开发人员管理，注意命名规则
  - `master` 整个项目只有一个
    稳定版本：由`bugfix_x.x.x`和`dev_x.x.x`分支提供内容。
  - `bugfix_x.x.x` 整个项目只有一个
    修复bug：从`master`上分离出来的分支，用于修复线上bug，测试通过后看产品要求。

    - 如果需要发版，则在此基础上分离出`dev_x.x.x`分支，打包提审，通过后合并到`master`以及其他所有`dev_x.x.x`分支。
    - 如果不需要发版，则将代码直接合并到所有`dev_x.x.x`分支
- `dev_x.x.x` 可以同时存在多个
  开发版本：从`master`上分离出来的分支，负责开发以及后续提测的bug修复，打包提审，通过后合并到`master`分支。
  PS：此时要更新`bugfix_x.x.x`分支（内容、名称），并在`master`上打对应`tag`。
- `xx_bugfix` 每个开发人员只有一个
  从`bugfix_x.x.x`上分离出来的分支，用于开发人员修复线上bug，修复后由相应人员`review`后合并到`bugfix_x.x.x`分支
- `xx_dev_x.x.x` 每个开发人员可以有多个
  从`dev_x.x.x`上分离出来的分支，用于开发人员开发新版本，开发完一个小颗粒后由相应人员`review`后合并到`dev_x.x.x`分支
- `其他`
  由开发人员自由定义，比如某个版本的`feature`分支。

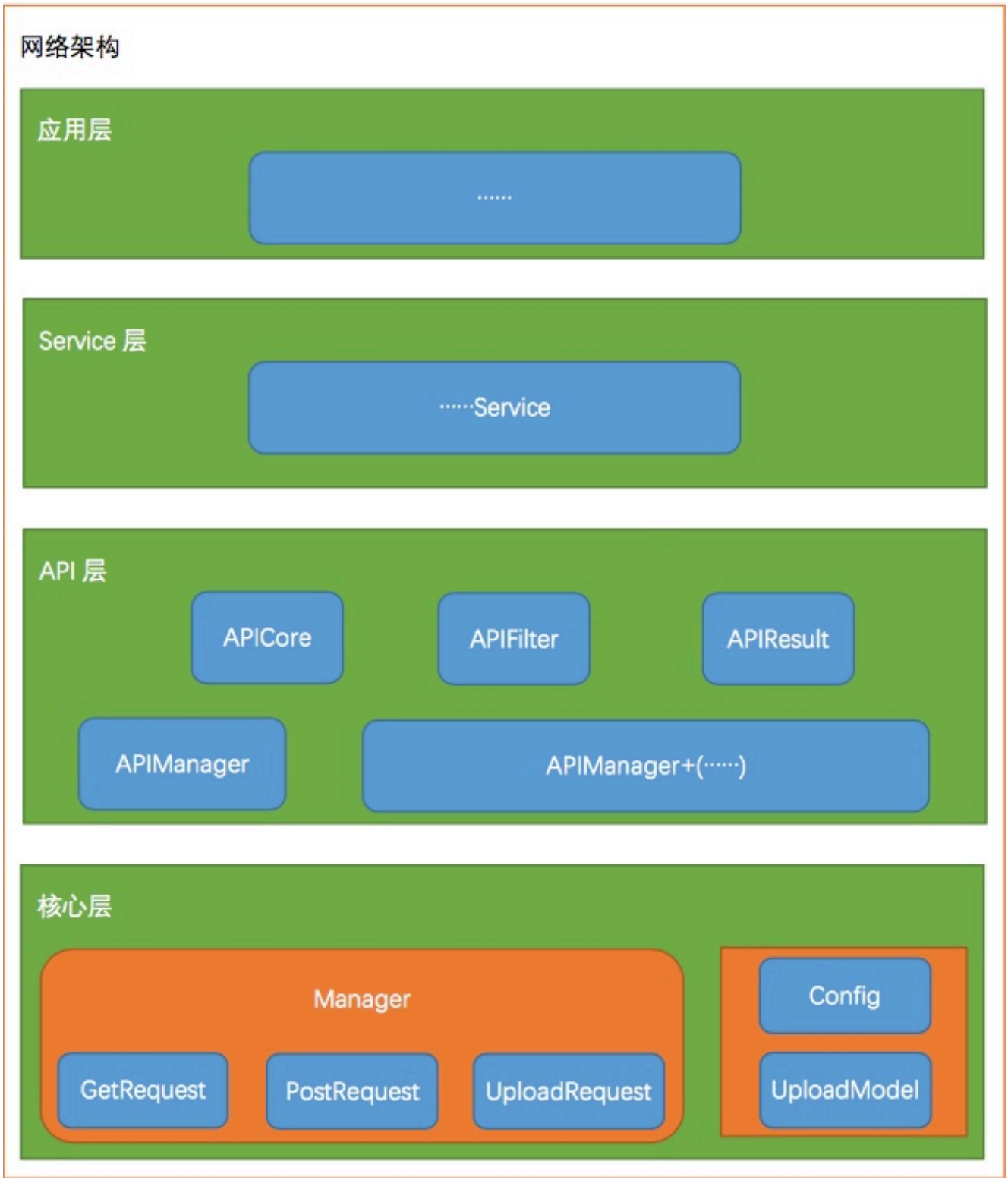
工程架构

采用分层架构设计（`表示层`、`业务逻辑层`、`核心逻辑层`）

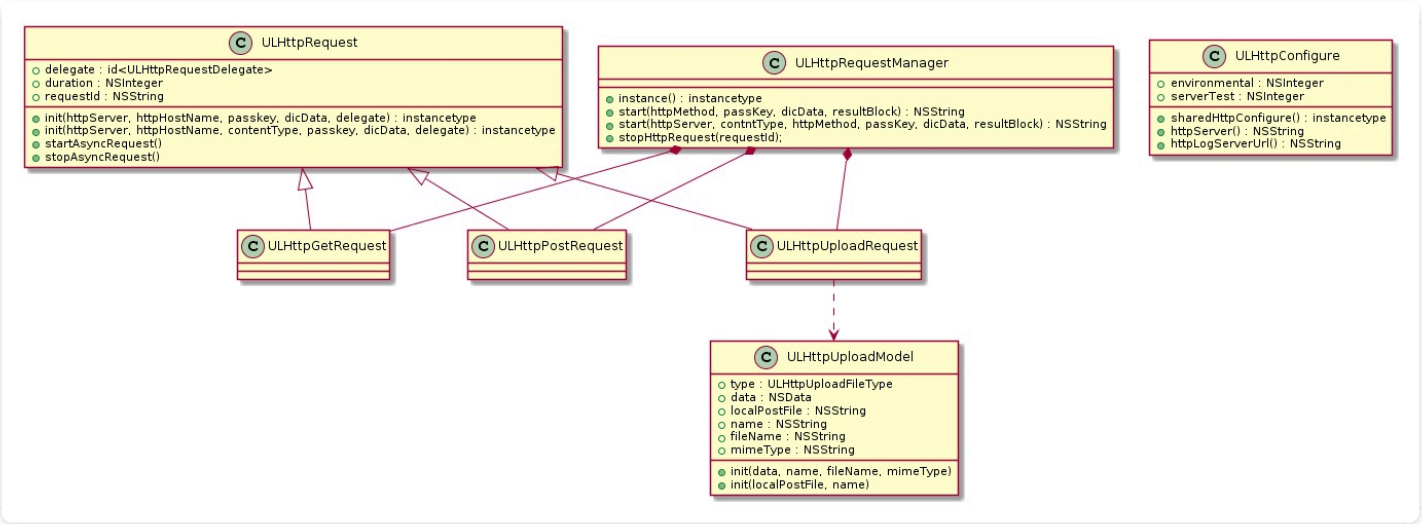
- 层与层之间严格分离
- 不可跨层访问
- 底层不可以访问上层服务

网络架构

- 架构设计


网络架构
应用层
.....
Service 层
.....Service
API 层
APICore
APIFilter
APIResult
APIManager
APIManager+(.....)
核心层
Manager
GetRequest
PostRequest
UploadRequest
Config
UploadModel

- 核心层类图



- Service层存在的意义
  一个接口（API）可能根据不同的参数实现不同的功能，那么这些逻辑如果全在API层，则会使得API层耦合严重，代码混乱，同样如果在应用层也不好。
  Service层就是为了保证API层功能单一、结构简单。进行一些业务逻辑上的处理。
  比如：

> *`/room/communicate/cancel` 的功能是移除连麦申请，但是会根据参数`opt`来判断是清空列表还是移除某个人。*
> *这时：API层只需负责根据参数实现接口的访问，具体是什么样的参数，就由Service层来提供。*

API层代码，只有一个方法

```
/**
 取消连麦申请

 @param roomID 房间ID
 @param type 类型，取消单个还是全部
 @param uid 如果取消单个，那么这个必填，用户ID
 @param block 回调
 @return 请求ID
 */
+ (NSString *)revokeMicWithRoomID:(ULLInteger)roomID type:(NSInteger)type removedU
ID:(ULLInteger)uid callback:(ULAPIResultCallback)block;
```

Service层代码，有两个方法

```
/**
 取消单人连麦申请

 @param roomID 房间ID
 @param uid 用户ID
```

```
 @param block 回调
 @return 请求ID
 */
+ (NSString *)revokeMicWithRoomID:(ULLInteger)roomID removedUID:(ULLInteger)uid ca
llback:(void (^)(BOOL isSuccess, NSString *message))block;


/**
 取消房间内所有人的连麦申请

 @param roomID 房间ID
 @param block 回调
 @return 请求ID
 */
+ (NSString *)revokeMicWithRoomID:(ULLInteger)roomID callback:(void (^)(BOOL isSuc
cess, NSString *message))block;
```

日志管理

采用`CocoaLumberjack`服务，在`核心逻辑层`统一向上暴露日志接口，方便日后更换日志服务
日志级别分为（`错误`、`一般信息`、`调试信息`）

- 测试和线上分别控制
  - 测试环境输出所有级别日志
  - 线上环境输出`错误`和`一般信息`
- 说明
  - 显示服务
  - 显示地址
  - 显示版本
  - 显示日志
  - **切换服务** （发布版本会隐藏掉）
- 打开方式

- 元素介绍

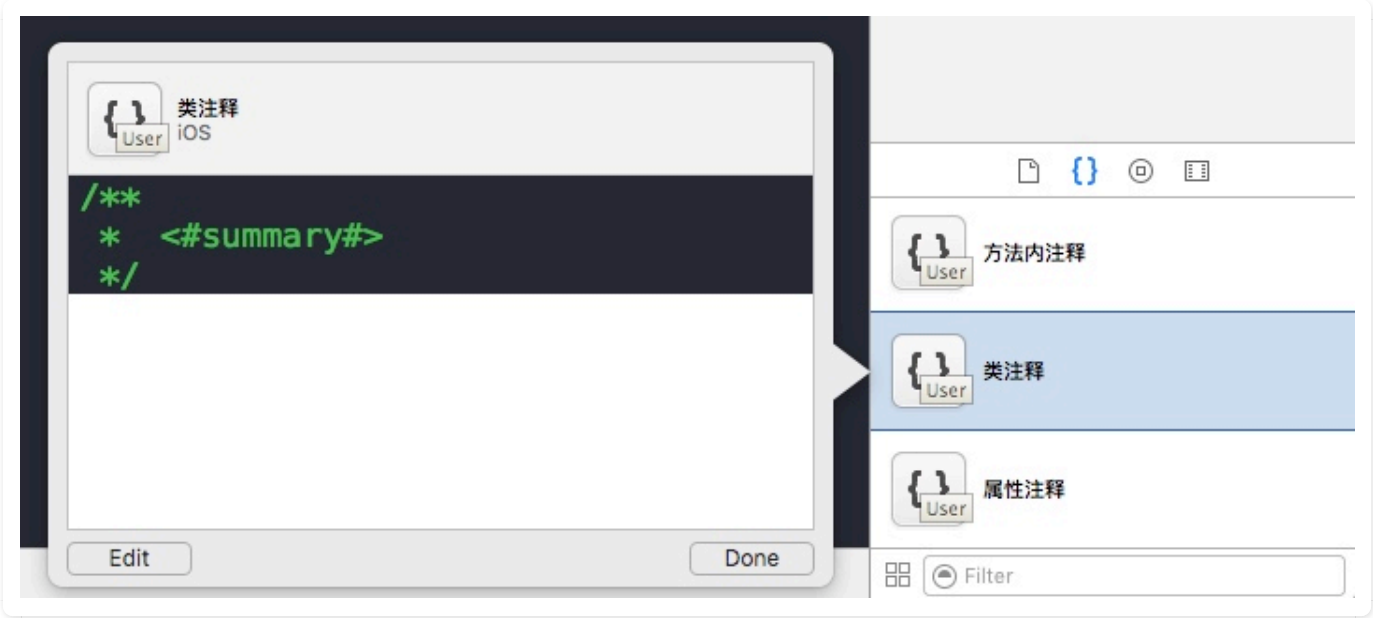

●●●●● 中国联通
16:09
服务
切换服务
日志
关闭页面
关闭
V1.2.6_10
版本号
环境：线上
地址：https://hongrenshuo.com.cn/api/
版本：企业
显示信息
2016/09/22 16:08:41:317 【友盟MobClick】 event = click_myresume_setting
2016/09/22 16:08:45:410 http request token 508adf7b-75a4-49ba-90e1-3a6df152fd24, ua os=8.3&imei=406e530fe342d4eb5a97cb0d456816ffe11945be&m=iPhone6,2&s=640x1136&c=1&vc=10&vn=1.2.6&n=红豆Live
2016/09/22 16:08:45:410 http request post data {
uid = 1721701646339;
}
2016/09/22 16:08:45:416 【友盟MobClick】 event = load_myresume
2016/09/22 16:08:45:417 get url = https://hongrenshuo.com.cn/api/v4/room/liveList?uid=1721701646339&pageSize=20&pageNo=1
2016/09/22 16:08:45:417 http request token 508adf7b-75a4-49ba-90e1-3a6df152fd24, ua os=8.3&imei=406e530fe342d4eb5a97cb0d456816ffe11945be&m=iPhone6,2&s=640x1136&c=1&vc=10&vn=1.2.6&n=红豆Live
2016/09/22 16:08:45:917
==========
日志

RequestURL:
https://hongrenshuo.com.cn/api/v4/user/queryUserInfo

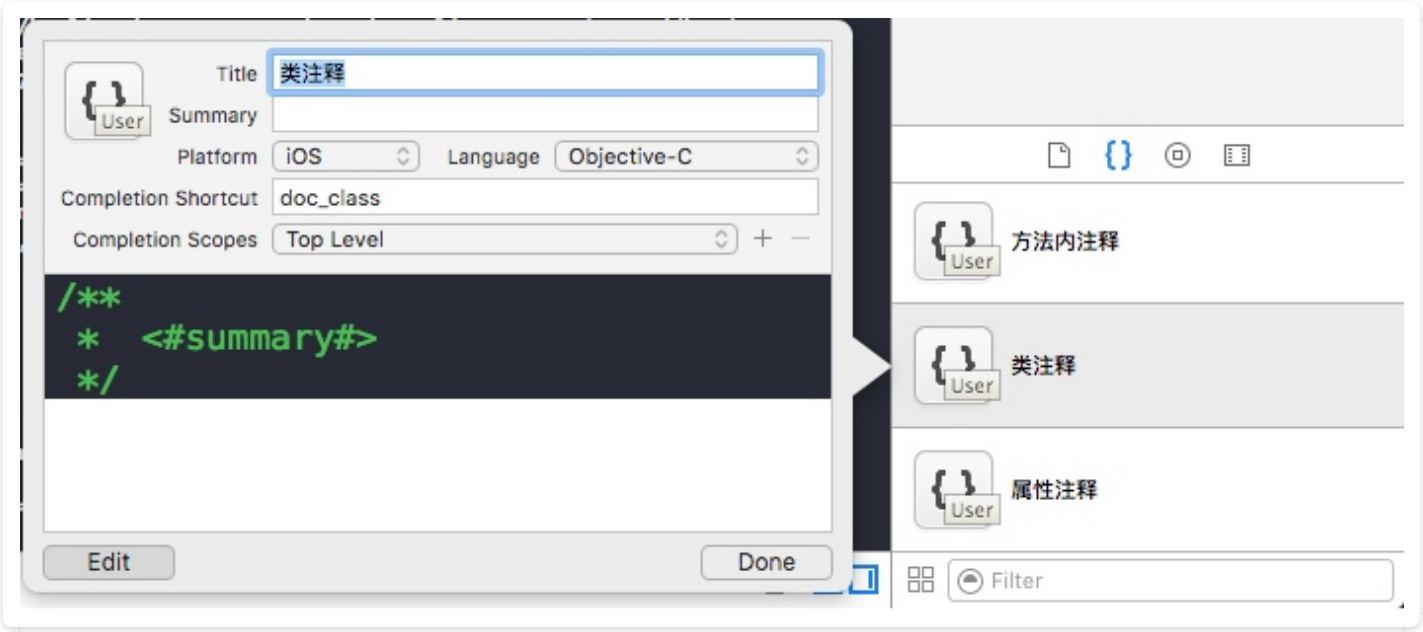
Response:
{"h":{"code":200,"msg":"操作成功","time":
1474531725849},"b":{"statisticInfo":{"diamondNumber":
0,"gold":382,"roomNumber":0,"concernNumber":

文档管理

- 了解appledoc以及安装
- 添加代码块
  - 在编辑区域编写代码
  - 拖进代码块区域



  - 做些基本设置



- 类注释

```
/**
 *  <#summary#>
 */
```

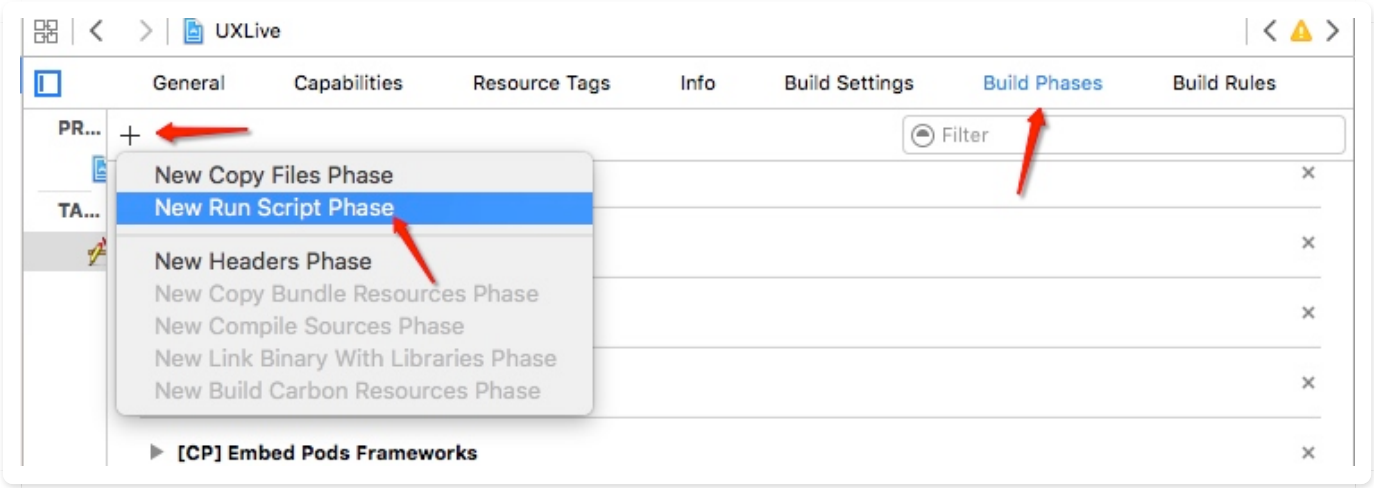
- 属性注释

```
/** <#description#> */
```

- 方法注释

```
Option+command+/
```

- 向工程中添加自动执行脚本
  - 指定`target`



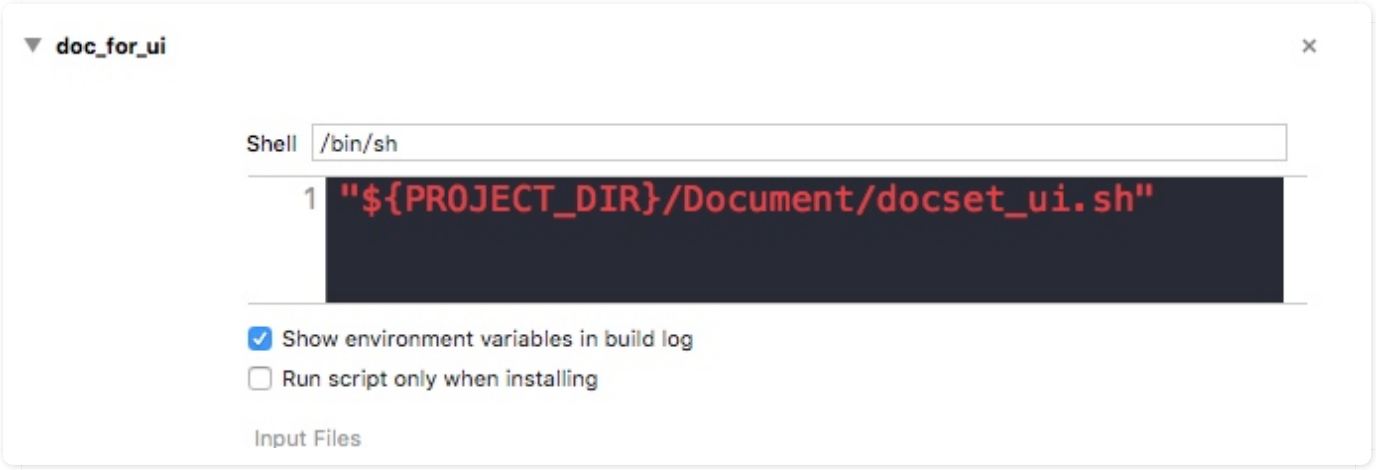
- 编写脚本

```
#!/bin/sh
#appledoc Xcode script
# Start constants
company="UXin";
companyID="com.uxin";
companyURL="http://www.uxin.com";
target="UXLive";
outputPath="${PROJECT_DIR}/Document/ULCore";
# End constants
/usr/local/bin/appledoc \
--project-name "${PROJECT_NAME}" \
--project-company "${company}" \
--company-id "${companyID}" \
--docset-atom-filename "${company}.atom" \
--docset-feed-url "${companyURL}/${company}/%DOCSETATOMFILENAME" \
--docset-package-url "${companyURL}/${company}/%DOCSETPACKAGEFILENAME" \
--docset-fallback-url "${companyURL}/${company}" \
--output "${outputPath}" \
--docset-platform-family "${target}" \
--logformat xcode \
--keep-intermediate-files \
--no-repeat-first-par \
--no-warn-invalid-crossref \
--exit-threshold 2 \
"${PROJECT_DIR}/UXLive/ULCore"
```

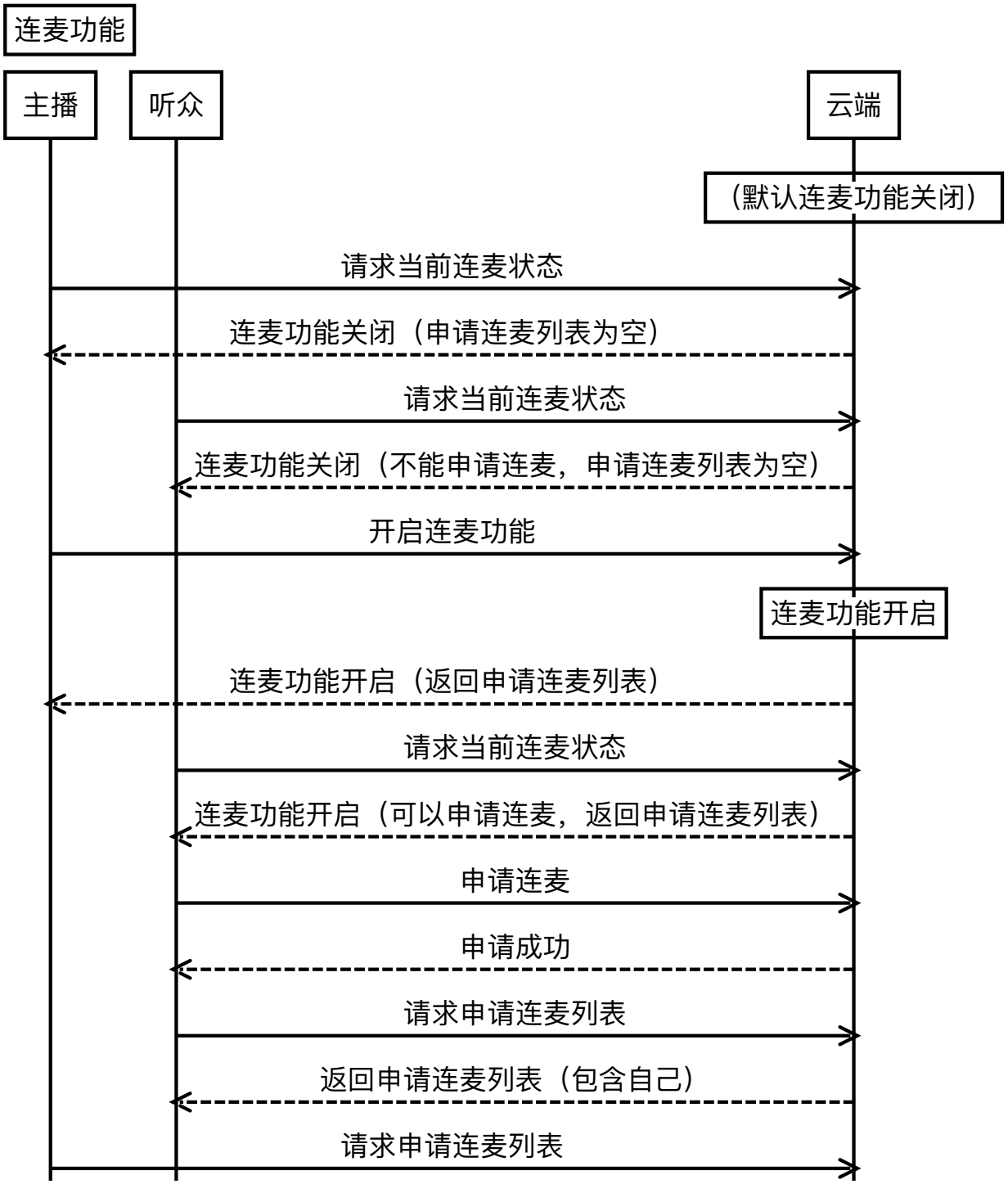
- 设置可执行权限

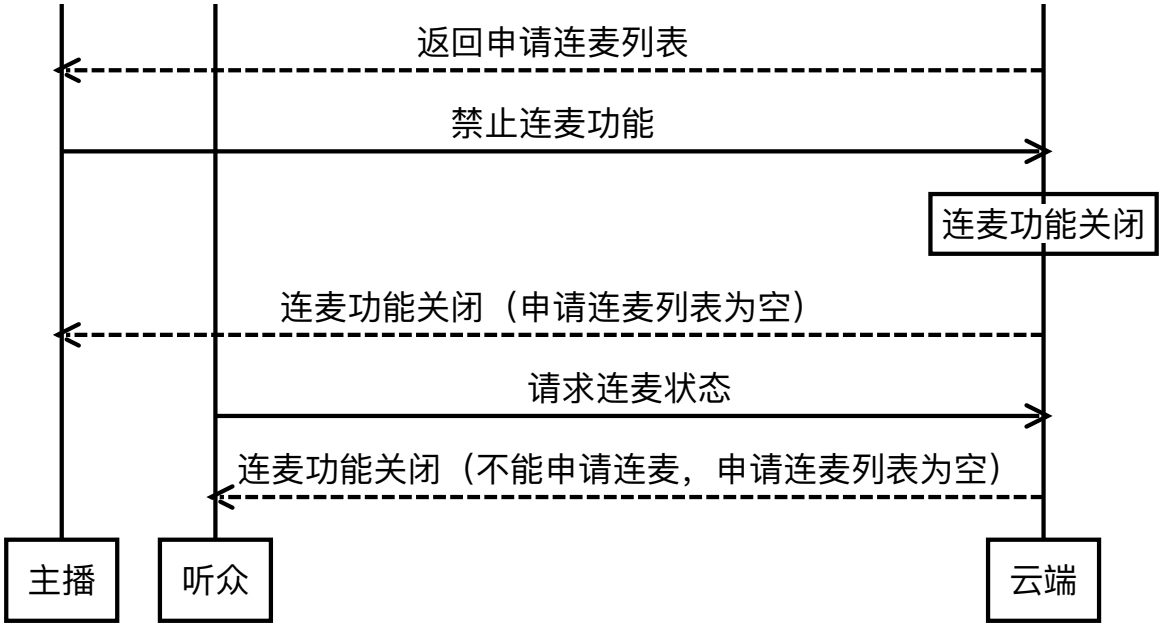
`chmod a+x xx.sh`

- 指定脚本路径

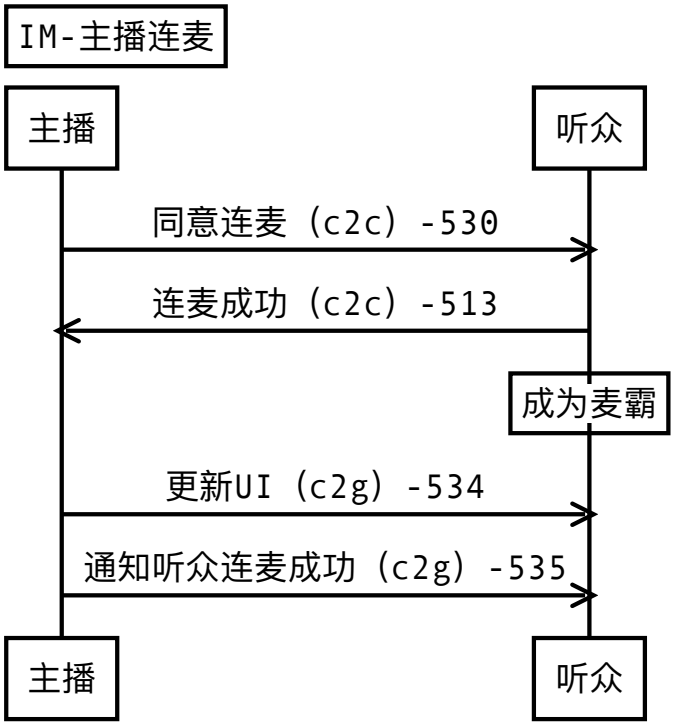

doc_for_ui
Shell
/bin/sh
1
"${PROJECT_DIR}/Document/docset_ui.sh"
Show environment variables in build log
Run script only when installing
Input Files

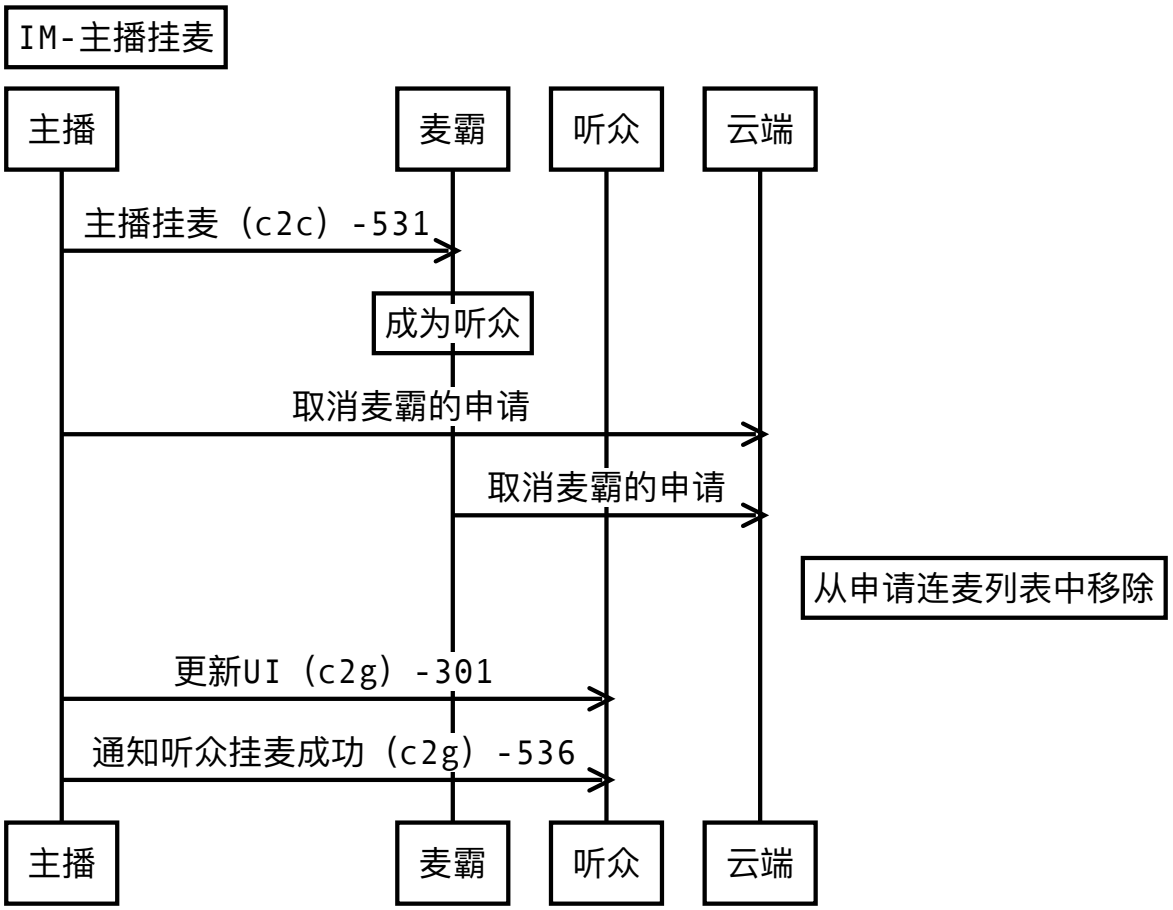
连麦功能

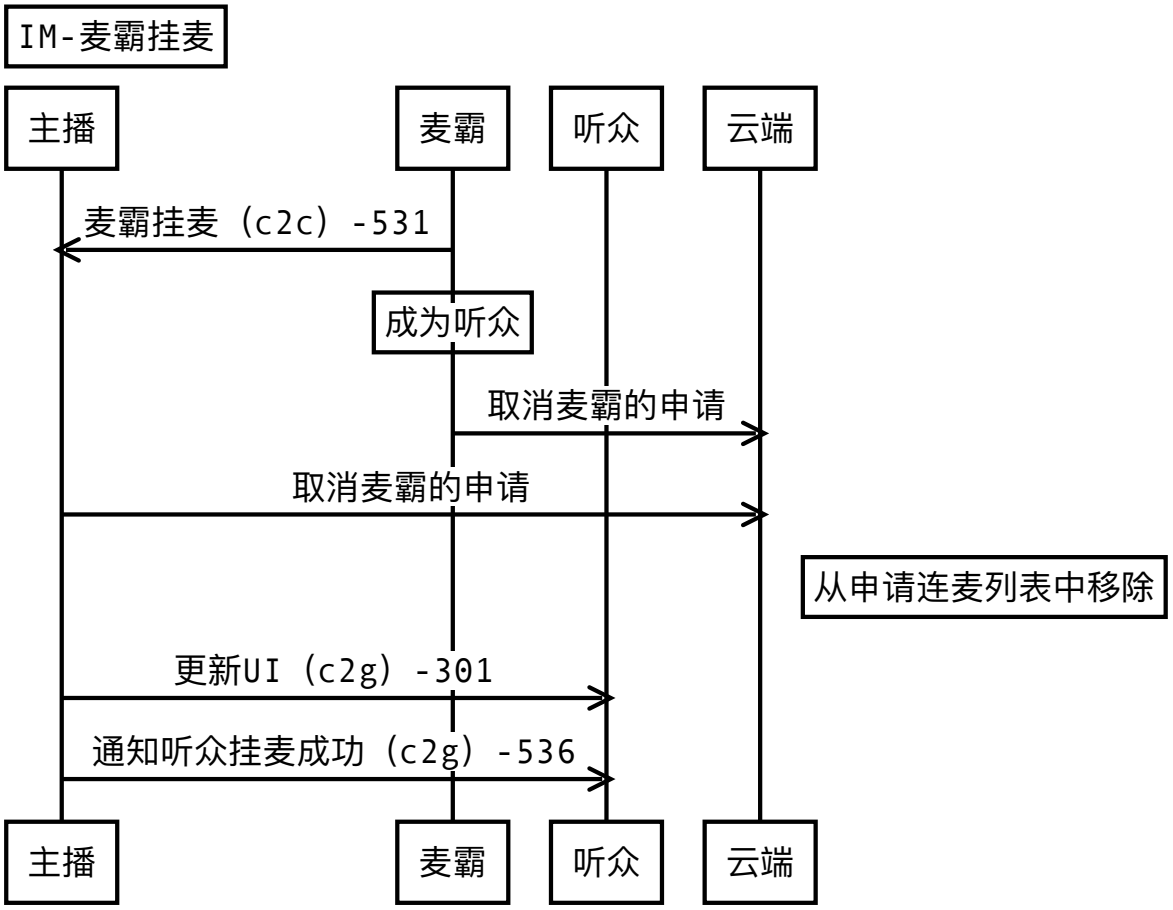

连麦功能
主播
听众
云端
(默认连麦功能关闭)
请求当前连麦状态
连麦功能关闭（申请连麦列表为空）
请求当前连麦状态
连麦功能关闭（不能申请连麦，申请连麦列表为空）
开启连麦功能
连麦功能开启
连麦功能开启（返回申请连麦列表）
请求当前连麦状态
连麦功能开启（可以申请连麦，返回申请连麦列表）
申请连麦
申请成功
请求申请连麦列表
返回申请连麦列表（包含自己）
请求申请连麦列表



c2c点对点消息
c2g点对群消息
-xxx消息类型




IM-主播挂麦
主播
麦霸
听众
云端
主播挂麦（c2c）-531
成为听众
取消麦霸的申请
取消麦霸的申请
从申请连麦列表中移除
更新UI（c2g）-301
通知听众挂麦成功（c2g）-536
主播
麦霸
听众
云端


IM-麦霸挂麦
主播
麦霸
听众
云端
麦霸挂麦（c2c）-531
成为听众
取消麦霸的申请
取消麦霸的申请
从申请连麦列表中移除
更新UI（c2g）-301
通知听众挂麦成功（c2g）-536
主播
麦霸
听众
云端

性能调优

- 接口访问
  - 减小数据包大小
  - 缓存
  - 时间控制
  - 版本控制
  - keep-alive
- 图片加载
  - 减小图片大小
  - 缓存
  - 预加载
  - 大小图组合