

# 目录

前言	1.1
APP开发简介	1.2
APP相关项目常见模式	1.2.1
常见APP模式	1.2.2
移动设备类型	1.2.3
APP开发总结	1.3
消息推送	1.3.1
服务商	1.3.1.1
极光推送JPush	1.3.1.1.1
开发总结	1.3.1.2
iOS消息推送	1.3.1.2.1
Android消息推送	1.3.1.2.2
调试移动端Webview/H5页面	1.3.2
调试iOS的Webview	1.3.2.1
调试Android的Webview	1.3.2.2
通话录音	1.3.3
iOS通话录音	1.3.3.1
Android通话录音	1.3.3.2
APP托管总结	1.4
APP托管平台	1.4.1
fir.im	1.4.1.1
蒲公英pgyer	1.4.1.2
APP上架总结	1.5
iOS上架	1.5.1
内测：AdHoc	1.5.1.1
公开市场：AppStore	1.5.1.2
内部发布：企业版	1.5.1.3
OTA版	1.5.1.3.1
Android上架	1.5.2
APP安装和使用总结	1.6
iOS的APP的安装和使用	1.6.1
Android的APP的安装和使用	1.6.2
APP涉及后期运维总结	1.7
数据统计	1.7.1
TalkingData	1.7.1.1
友盟UMeng	1.7.1.2
Bugly	1.7.1.3
崩溃日志收集	1.7.2
Bugly	1.7.2.1

---

应用内问题反馈	1.7.3
蒲公英pgyer	1.7.3.1
子教程	1.8
附录	1.9
参考资料	1.9.1

---

# 移动端APP开发总结

- 最新版本: v0.9
- 更新时间: 20231213

## 简介

之前折腾过少量的Android开发和一段时间的iOS开发，也了解过混合App的开发，也涉及过基于H5页面加原生壳的APP开发等等，所以此处去整理出这些关于移动端APP开发的各种方面的技术和经验心得，供参考。

## 源码+浏览+下载

本书的各种源码、在线浏览地址、多种格式文件下载如下：

### HonKit源码

- [crifan/mobile\\_app\\_summary](#): 移动端APP开发总结

### 如何使用此HonKit源码去生成发布为电子书

详见：[crifan/honkit\\_template](#): demo how to use crifan honkit template and demo

### 在线浏览

- 移动端APP开发总结 [book.crifan.org](http://book.crifan.org)
- 移动端APP开发总结 [crifan.github.io](https://crifan.github.io)

### 离线下载阅读

- 移动端APP开发总结 PDF
- 移动端APP开发总结 ePUB
- 移动端APP开发总结 MOBI

## 版权和用途说明

此电子书教程的全部内容，如无特别说明，均为本人原创。其中部分内容参考自网络，均已备注了出处。如发现有侵权，请通过邮箱联系我 [admin 艾特 crifan.com](mailto:admin@crifan.com)，我会尽快删除。谢谢合作。

各种技术类教程，仅作为学习和研究使用。请勿用于任何非法用途。如有非法用途，均与本人无关。

## 鸣谢

感谢我的老婆陈雪的包容理解和悉心照料，才使得我 [crifan](#) 有更多精力去专注技术专研和整理归纳出这些电子书和技术教程，特此鸣谢。

## 其他

### 作者的其他电子书

本人 crifan 还写了其他 150+ 本电子书教程，感兴趣可移步至：

[crifan/crifan\\_ebook\\_readme: Crifan的电子书的使用说明](#)

## 关于作者

关于作者更多介绍，详见：

[关于CrifanLi李茂 – 在路上](#)

crifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新：2023-12-13 15:21:33

# APP开发简介

在详细介绍移动端APP开发的各种细节之前，先对移动端APP的基本情况做个简介。

crifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新：2018-02-05 22:41:33

## APP相关项目常见模式

APP相关的项目的开发，目前常见有如下几种方案/模式：

### 纯Web模式

PC端：Web页面 手机端：（移动端）Web页面，页面适配手机/Pad等屏幕大小

方案：

- Web页面
  - JS框架：
    - jQuery
    - Angular JS
    - Vue.js
    - React JS
  - UI框架
    - Bootstrap

### PC端Web+App混合模式

PC端：一套Web页面，适配PC端 移动端：一套Web页面，适配移动端 + APP端打包一个壳，内置Webview，加载显示web页面

方案：

- Web页面：
  - JS框架：
    - jQuery
    - Angular JS
    - Vue.js
    - React JS
- APP端
  - iOS
    - 开发工具：Android Studio
      - 只是开发个壳，嵌入chrome的webView，用webView加载web页面
  - Android
    - 开发工具：(Mac) Xcode
      - 只是开发个壳，嵌入safari的UIWebView，用UIWebView加载web页面
      - 另外一种，虽然和Webview+原生的壳不同，但是也属于广义上的混合APP的模式：
      - React Native的混合模式
      - 用React Native写Javascript代码，一套代码
      - -> 生成iOS和Android的原生的app

### PC端Web + App端：原生APP

- Web页面
  - JS框架
    - `jQuery`
    - `Angular JS`
    - `Vue.js`

- React JS

- APP端

- Android

- 开发工具: Android Studio = AS
  - 开发完整的独立的app的完整功能

- iOS

- 开发工具: (Mac中的) Xcode
  - 开发完整的独立的app的完整功能

# 常见APP模式

正如前面的介绍，一般项目涉及到的APP模式，可以简化立即为有三种：

- 纯Web=纯Web页面适配移动端
- Web App=移动端混合APP
- Native App=纯原生移动端APP

而其中的 纯Web的，指的是一套Web页面，完全适配移动端浏览器，所以还是属于基于浏览器的页面，不算传统意义上的移动端的APP，所以不做详细解释。

接下来主要解释：混合式的Web App和纯的Native原生APP。

## Web App对比Native App

详细解释之前，对于典型的Web的APP和原生的Native的APP进行一个简单的对比：

Native App	Web App	说明
偏交互	偏浏览	交互指复杂操作，输入/选择什么的
已稳定	试错中	H5页面用来做低成本验证很好
访问硬件	信息展示	指手机里的各种传感器什么的
核心功能	周边辅助	把工作量多投在刀刃上

## 原生Native

原生的APP，概念上其实很简答，就是指的是：

- iOS平台：典型的指的是，在Mac的Xcode下，用纯的之前的 Objective-C 或最新的 Swift 的代码开发和调试，最终生成的原生的iOS的ipa文件
- Android：在 Mac / Windows 系统中，用之前的 Eclipse+ADT 或最新的 Android Studio 工具，写之前的 Java 或最新的 Kotlin 代码，去开发调试出来的Android原生的apk文件

## 混合APP

关于混合APP开发，往往指的是：用一套框架或技术，最终得到的app。

### H5+原生的壳

其中，用原生的app写个基本的启动界面，加上一个WebView，然后用webview去加载所有的app的内容，即一堆的HTML+CSS+JS，从而实现app的效果。

所以，就分成两部分：

- H5页面=Webview页面
- 原生的壳=原生代码(iOS/Android)去写对应的app客户端
  - 可能还包括一些其他相关的基本功能：
    - 启动页面
    - 自动保存密码
      - 用于实现自动登录
    - 消息推送

- 纯Web页面无法（很难）做到消息推送，只能靠原生

如此，H5+原生的壳，就实现了一个移动端的APP，如果效果做的还可以，基本上可以以假乱真，看起来和原生的APP没多大区别。

对于的这类移动端的Web页面的设计，也有专门的逻辑和注意事项，比如：

[Mobile Web Development - Web developer guides | MDN](#)

## 混合APP框架

另外一种实现一般所谓的混合APP的做法就是：使用某个混合APP框架去实现

而混合式APP框架，又可以分为两类：

- 传统混合APP框架
- 新出的一些特殊的框架

下面分别来介绍。

## 传统混合APP框架

传统的框架，有很多，其中一些比较出名的有：

### **cordova(旧称PhoneGap)**

之前叫PhoneGap，后被Adobe收购了改名为cordova。

主页：[Apache Cordova](#)

### **ionic**

还有一个混合APP框架 `ionic`，好像用的也不少

主页：[Build Amazing Native Apps and Progressive Web Apps with Ionic Framework and Angular](#)

## 新出现的基于JavaScript的框架

后来又出现一些，从广义上也可以被称为混合式APP框架的：

### **React Native**

`React Native = RN`

Facebook出的，可以用一套 `React Native` 框架，写JS代码，最终生成iOS和Android两端的App。

性能总体上比纯Web要好。

主页：[React Native · A framework for building native apps using React](#)

### **weex**

阿里的出的，目标是支持一套代码，生成iOS、Android和Web端的程序。

常常被拿来和RN对比。

目前流行度好像一般。

主页：[Weex](#)

### **HBuilder**

还有个叫做 `HBuilder`，好像不是很流行，但是之前简单尝试过，没具体折腾。

好像也可以生成APP，但是感觉是一个官网的原生的APP壳，加上内部的Webview的效果。

也算作其中一个框架吧，虽然可能是从功能上来说是比较弱的。

主页：[DCloud - HBuilder、5+、mui、流应用、HTML5专家](#)

crifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook 最后更新：2022-11-03 17:22:08

# 移动设备类型

- 按操作系统分
  - iOS
    - 公司: Apple苹果
    - 产品
      - iPhone
      - iPad
      - 等
  - Android
    - 公司: Google谷歌
- 按设备尺寸分
  - 相对大屏的 平板 = Pad
  - 想对小屏的 手机 = Phone

crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新: 2022-11-03 17:32:54

# APP开发总结

接下来介绍，对于移动端，包括iOS端和Android端，相对来说算是通用的开发相关的知识。

crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：2018-02-06 21:19:01

## 消息推送

iOS和Android端，都会涉及到，消息推送。

典型的场景有：

系统发送一个通知到某个用户，需要对应消息提醒，且希望在App没有运行的情况下，就能收到iOS和Android的系统的消息推送

比如，iOS手机中收到一条消息推送：



对于这类场景的消息推送，一般也叫做 离线消息推送。

下面来详细解释消息推送方面的知识。

crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：2018-02-06 23:13:27

## 消息推送服务商

能否实现消息推送的功能的话，一般从功能上，至少包括两部分：

- 消息推送的服务器端：实现消息队列的管理，选择合适的消息发送的时机，且要考虑到APP端可能离线，以及消息重发等等内部复杂逻辑判断。
  - 一般来说，如果不是有特殊需要，往往都是采用使用第三方的消息推送的服务
  - 否则完全自己实现消息推送的后台服务，还是需要一定端技术成本和服务器采购成本的
    - 尤其是把消息推送做到让移动端APP方便调用，且消息推送的效果好，那需要更大的成本
- 消息推送的APP端：
  - 在iOS和Android端，利用系统提供的消息推送端方面的接口，实现消息的接收和处理

对于第三方面端消息推送服务，做的比较好，主要是：极光推送 = JPush，下面就来详细介绍一下极光推送。

crifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新：2018-02-06 21:39:30

# 极光推送JPush

极光推送 = JPush

主页: 首页 - 极光 | 数据改变世界

## 极光推送的价格

默认是有免费的服务使用的。限制是：所有的使用免费的极光推送的APP去共享使用20万/秒的服务。

- -> 换句话说：如果你的APP的消息推送正好赶上其他所有的免费的APP使用极光的服务的高峰期，则消息推送可能会稍微延迟一段时间。
  - 其实一般情况下，多数情况下，延迟也还是很低的，总体的感觉是80%以上的时间，服务器端消息推送后，APP端收到消息的延迟，都是在1秒~5秒，足够一般的APP使用了，不会觉得多慢的
- -> 换另外一句话说：如果真的是你的APP对于消息推送的推送的及时率有很高的要求，那么就应该去购买收费的极光推送的服务了
  - 收费的激光服务的价格：最少每月也要2000多元
    - 所以收费服务，对于小公司，还是挺贵的
    - 所以对于一般的APP来说，如前面所说的极光的免费的消息推送的服务，也就够用了，不需要额外花钱买收费的

另外，对于APP用户规模大的，需要购买收费的极光的服务的，附上价格表，供参考：

The screenshot shows a PDF document titled "极光推送付费服务报价表" (JPush Push Service Price List). The document is a table with columns: 版本类型 (Version Type), APP 用户规模 (APP User Scale), 在线峰值用户 (Online Peak User), 月费 (元) (Monthly Fee (CNY)), Push API 频率 (次/分钟) (Push API Frequency (Per Minute)), and 增加1万在线用户费用增加 (元) (Additional fee for increasing 10,000 online users (CNY)). The table includes the following data:

版本类型	APP 用户规模	在线峰值用户	月费 (元)	Push API 频率 (次/分钟)	增加1万在线用户费用增加 (元)
初级版	10万-20万	2万	2,499	12,00	1000
基本版	50万-100万	10万	9,999	48,00	1000
中级版	100万-250万	30万	25,999	120,00	750
专业版	250万-500万	50万	35,999	240,00	750
高级版	500万-1000万	100万	49,999	480,00	500
中型企业版	1000万-2000万	200万	89,999	1200,00	500
大型企业版	2000万-5000万	500万	199,999	4800,00	500

计费原则：每月根据App当月在线用户峰值加超出部分作为当月扣费标准，当此费用超过下一版本月费时系统自动按下一版本费用扣费。如16万月在线用户峰值当月实际付费为  $9999+6000 = 15999$  元；27万月在线用户峰值实际付费为  $9999+17000 = 26999$ ，则当月付费按下一版本25999计费。

注：日在线用户数：一天之内登录JPush服务器的独立用户数  
月在线用户数峰值：自然月内“日在线用户数”最高的一天的数值

此付费服务报价表解释权归极光推送所有

## 极光推送消息基本知识

## iOS的证书

iOS的消息推送，支持开发环境和测试环境。

需要分别去上传对应的证书才可以的。

The screenshot shows the 'jiandao' application settings page. At the top, there are tabs for Push, Statistics, IM, and Application Settings. The current tab is Application Settings. Below the tabs, the application name is listed as 'com.daryun.jiandao'. A button labeled 'Download Android Example' is visible. The main section is titled 'iOS' and contains the following information:

- Bundle ID:** com.daryun.jiandao
- APNS推送环境:** 生产环境 (Production Environment)
- APNs证书文件:** 开发环境: 已验证 (Development Environment: Verified), 生产环境: 已验证 (Production Environment: Verified)
- 证书有效期至:** 开发环境: 2017-03-10 14:46:49, 生产环境: 2017-05-08 20:22:53

Below the iOS section is a 'WinPhone' section with the status '启用 WinPhone: 否' (Enable WinPhone: No). At the bottom is a blue '修改应用' (Edit Application) button.

## 标签tag和别名Alias

可以通过tag或alias，去实现，给某类用户设置同一个tag或alias，然后消息推送的时候，给同一类的用户批量推送相同的消息。

比如，牛只管理app中，对于牛只的管理员，当发生异常信息，比如牛只设备掉落时，给所有的同一组的管理员，都发送预警消息。

另外一个典型的使用方式是：

比如对于一个APP，内部有一套自己的用户系统，其中每个用户有自己的userid，比如类似于UUID的这种： user-39ee1299-4e29-43cf-904f-d8826ce1b899

而如果移动端想要实现接收到服务器推送不同的消息给每个用户，其中一种实现方式就是：

每个用户都设置一个，借用userid的独一无二的别名，比如：

user-39ee1299-4e29-43cf-904f-d8826ce1b899

后续服务器端给每个用户推送消息时，就直接给对应别名去推送消息，对应的用户即可收到对应的消息推送了。

其中几点需要说明的是：

- APP中用户登录后，APP初始化时（在init初始化JPush的API后）去注册register对应的自己用户ID对应的别名alias
  - 符合正常的逻辑：用户登录后，应该能收到消息推送
- APP中用户退出登录时，去注销对应的JPush的对应的别名

- 这样后续该用户就不会收到消息提醒了
- 符合正常的逻辑：用户注销后，不应该继续收到消息推送

## JPush的服务不错

### 后台管理页面中可以方便测试消息推送

可以方便的去测试消息发送

支持对于：

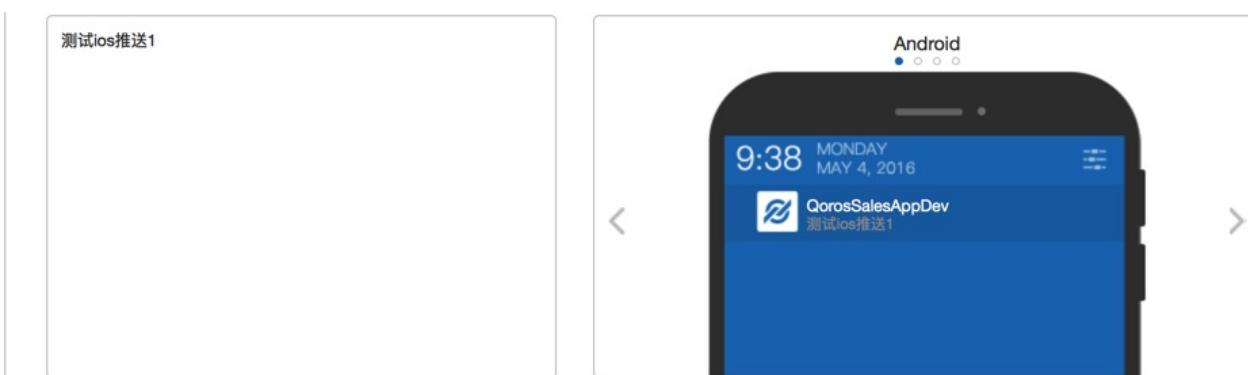
- iOS
  - 生产环境
  - 测试环境
- Android

还可以指定发送对象

- 具体的某个设备
- 某个tag
- 某个alias
- 广播所有设备

等等。

截图供参考：



**推送对象**

**目标平台 (必选)**

iOS开发环境	iOS生产环境	Android	WinPhone

**目标人群**

广播 (所有人)	设备标签 (Tag)	设备别名 (Alias) <b>(selected)</b>	Registration ID	用户分群推送
----------	------------	--------------------------------	-----------------	--------

手动输入  文件上传

10000027

推送后，可以方便的看到推送的结果：

The screenshot shows the JPush control panel interface. On the left, there's a sidebar with options like '自定义消息', '富媒体消息', '推送历史' (which is selected), and '定时消息'. The main area is titled '推送历史' (Push History) and shows a summary for a push sent on '2016-07-25 16:49' with the message 'test push to produ...'. The summary includes statistics: 2 targets, 2 successes, and 0 clicks. Below this, detailed log entries are listed for each target, including fields like '发送编号' (Delivery Number), 'Message ID', '推送方式' (Delivery Method), '接受对象' (Recipient), '推送内容' (Content), '推送平台' (Delivery Platform), 'Extras', '应用角标(badge)', '通知声音(sound)', and '推送结果' (Delivery Result). The results show 5 successes and 4 failures.

## iOS证书将要过期会及时提醒

极光推送在证书快要过期前一个月会发邮件通知管理员的，需要你及时去更新证书：

极光

发给

尊敬的极光推送开发者，您好：

您AppKey为c31...d48,名为...的应用iOS开发证书到期时间为：2018-01-10 16:20:48。

为保证推送功能不受影响，请于证书到期前：

1. 于*IOS Dev Center*创建并下载新的push证书，并保证该证书与之前的具有相同的App ID；
2. 通过极光推送控制台的*应用设置*，将新的证书上传至该应用的APNS证书。

如对APNS证书不太了解，请参考*IOS证书设置指南*；

如对极光的控制台如何设置应用不太了解，请参考相关的*iOS SDK集成指南*。

极光推送技术支持团队

[support@jpush.cn](mailto:support@jpush.cn)

然后去Apple的开发者网站*Apple Developer*中去：

- 撤销旧证书
- 重新生成新证书
- 重新到JPush后台上传新证书

即可。

## iOS中JPush相关示例代码

此处附上之前的iOS的app：简道中的JPush的消息推送的iOS端的相关代码，供参考：

## APP初始化部分的JPush相关配置

文件：`AppDelegate.swift`

代码：

```

let jpushAppKey:String = "64abxxxxxxxxx47"
let jpushChannel:String = "ios-AppStore"
//let JpushIsProduction:Bool = false
let JpushIsProduction:Bool = true

func application(application: UIApplication, didFinishLaunchingWithOptions launchOptions: [NSObject: AnyObject]?) -> Bool {
    gLog.debug("didFinishLaunchingWithOptions launchOptions=\(launchOptions)")

    //1. register device
    JPUSHService.registerForRemoteNotificationTypes(UIUserNotificationType.Badge.rawValue | UIUserNotificationType.Sound.rawValue | UIUserNotificationType.Alert.rawValue, categories: nil)
    JPUSHService.setupWithOption(launchOptions, appKey: jpushAppKey, channel: jpushChannel, apsForProduction: JpushIsProduction)
    listenRemotePush()

    return true
}

func application(application: UIApplication, didRegisterForRemoteNotificationsWithDeviceToken deviceToken: NSData) {
    //2. receive device token for register
    gLog.debug("deviceToken=\(deviceToken)")
    //deviceToken=<77300f0d c4a4f2f4 74fe24de 5db46132 c725ee1a 8e41b1fc a9a511bc 832c0113>
    NSNotificationCenter.defaultCenter().postNotificationName("DidRegisterRemoteNotification", object: deviceToken)
}

//3. send device token to push server
JPUSHService.registerDeviceToken(deviceToken)

func application(application: UIApplication, didFailToRegisterForRemoteNotificationsWithError error: NSError) {
    gLog.debug("error=\(error.localizedDescription)")
}

// //for iOS <= 6.0
// func application(application: UIApplication, didReceiveRemoteNotification userInfo: [NSObject : AnyObject]) {
//     //4. receive the remote push message from (APNS -> JPUSH) server
//     print("didReceiveRemoteNotification")
//     print("userInfo=\(userInfo)")
//     JPUSHService.handleRemoteNotification(userInfo)
//     NSNotificationCenter.defaultCenter().postNotificationName("AddNotificationCount", object: nil)
// }

//for iOS >= 7.0
func application(application: UIApplication, didReceiveRemoteNotification userInfo: [NSObject : AnyObject], fetchCompletionHandler completionHandler: (UIBackgroundFetchResult) -> Void) {
    gLog.debug("userInfo=\(userInfo)")
    JPUSHService.handleRemoteNotification(userInfo)
    NSNotificationCenter.defaultCenter().postNotificationName("AddNotificationCount", object: nil)
    NSNotificationCenter.defaultCenter().postNotificationName("handleRemoteNotification", object: userInfo)
    completionHandler(UIBackgroundFetchResult.NewData)
}

func application(application: UIApplication, didReceiveLocalNotification notification: UILocalNotification) {
    gLog.debug("notification=\(notification)")

    //notification=<UIConcreteLocalNotification: 0x15f150b00>{fire date = 2016年3月11日 星期五 中国标准时间 10:33:53, time zone = Asia/Shanghai (GMT+8) offset 28800, repeat interval = 0, repeat count = UILocalNotificationInfiniteRepeatCount, next fire date = (null), user info = (null)}
    JPUSHService.showLocalNotificationAtFront(notification, identifierKey: nil)

    if let userInfo = notification.userInfo {
        let newMessage = userInfo["newMessage"] as! Message
        debugPrintMsg(newMessage)
    }
}

```

```

@available(iOS 7, *)
func application(application: UIApplication, didRegisterUserNotificationSettings notificationSettings: UIUserNotificationSettings) {
    gLog.debug("notificationSettings=\(notificationSettings)")
    //notificationSettings=<UIUserNotificationSettings: 0x70070130; types: (UIUserNotificationTypeAlert UIUserNotificationTypeBadge UIUserNotificationTypeSound);>
    /*
     notificationSettings=<UIUserNotificationSettings: 0x12e6529a0; types: (UIUserNotificationTypeAlert UIUserNotificationTypeBadge UIUserNotificationTypeSound);>
     2016-03-11 16:22:05.110 | JPUSH | W - [JPUSHClientController] Not get deviceToken yet. Maybe: your certificate not configured APNs? or current network is not so good so APNs registration failed? or there is no APNs register code? Please refer to JPUSH docs.

     2016-03-11 16:22:05.113 | JPUSH | I - [JPUSHSessionController] sis is not on protect
     2016-03-11 16:22:05.116 | JPUSH | I - [JPUSHAddressController] Action - sendSisRequest
    */
}

@available(iOS 7, *)
func application(application: UIApplication, handleActionWithIdentifier identifier: String?, forLocalNotification notification: UILocalNotification, completionHandler: () -> Void) {
    gLog.debug("handleActionWithIdentifier=\(identifier), forLocalNotification=\(notification)")
}

@available(iOS 7, *)
func application(application: UIApplication, handleActionWithIdentifier identifier: String?, forRemoteNotification userInfo: [NSObject : AnyObject], withResponseInfo responseInfo: [NSObject : AnyObject], completionHandler: () -> Void) {
    gLog.debug("handleActionWithIdentifier=\(identifier), forRemoteNotification=\(userInfo), withResponseInfo=\(responseInfo)")
}

func networkDidLogin(notification:NSNotification) {
    gLog.debug("已登陆 networkDidLogin notification=\(notification)")

    if let registrationID = JPUSHService.registrationID() {
        gLog.debug("registrationID=\(registrationID)")
    }
}

func listenRemotePush(){
    gLog.debug("")
    let defaultCenter:NSNotificationCenter = NSNotificationCenter.defaultCenter()
    defaultCenter.addObserver(self, selector: #selector(AppDelegate.networkDidSetup(_:)), name:kJPFFNetworkDidSetupNotification, object: nil)
    defaultCenter.addObserver(self, selector: #selector(AppDelegate.networkDidClose(_:)), name:kJPFFNetworkDidCloseNotification, object: nil)
    defaultCenter.addObserver(self, selector: #selector(AppDelegate.networkDidRegister(_:)), name:kJPFFNetworkDidRegisterNotification, object: nil)
    defaultCenter.addObserver(self, selector: #selector(AppDelegate.networkDidLogin(_:)), name:kJPFFNetworkDidLoginNotification, object: nil)
    defaultCenter.addObserver(self, selector: #selector(AppDelegate.networkDidReceiveMessage(_:)), name:kJPFFNetworkDidReceiveMessageNotification, object: nil)
    defaultCenter.addObserver(self, selector: #selector(AppDelegate.serviceError(_:)), name:kJPFFServiceErrorNotification, object: nil)

    defaultCenter.addObserver(self, selector: #selector(AppDelegate.didRegisterRemoteNotification(_:)), name:"Did RegisterRemoteNotification", object: nil)

    defaultCenter.addObserver(self, selector: #selector(AppDelegate.handleRemoteNotification(_:)), name:"handleRemoteNotification", object: nil)
}

func handleRemoteNotification(remoteNotification:NSNotification) {
    gLog.debug("remoteNotification=\(remoteNotification)")
    if let userInfoDict = remoteNotification.object {
        gLog.debug("userInfoDict=\(userInfoDict)")

        if let aps = userInfoDict["aps"] {
            if let badge = aps["badge"] as? Int {
                gLog.debug("badge=\(badge)")
                UIApplication.sharedApplication().applicationIconBadgeNumber = badge
            }
        }
    }

    if UIApplication.sharedApplication().applicationState == UIApplicationState.Active {

```

```

        gLog.debug("not handle remote notification for app is active running")
    } else {
        gLog.debug("applicationState=\\"$APPLICATION.sharedApplication().applicationState\\")

        SingletonConversationTVC().updateAllConversation()
    }
}

```

## 登录后去初始化调用

文件: LoginViewController.swift

代码:

```

func doAfterLogin(){
    dispatchBackground_async({
        //init message related
        SingletonMainvc().initWebSocket()

        ...
    })
}

```

文件: MainViewController.swift

代码:

```

func jpushSetAlias() {
    gLog.debug("")
    var userIdAlias = gCurUserItem.id
    userIdAlias = userIdAlias.replace("user-", to: "usr-")
    userIdAlias = userIdAlias.replace("-", to: "_")
    gLog.debug("userIdAlias=\\"$userIdAlias\\"")
    //user-e6882cab-cedf-4335-9b7c-612cd5b4d37d
    //usr_e6882cab_cedf_4335_9b7c_612cd5b4d37d

    jpushSetAlias(userIdAlias)
}

func jpushSetAlias(userIdAlias:String) {
    gLog.debug("userIdAlias=\\"$userIdAlias\\"")

    JPUSHService.setAlias(userIdAlias, callbackSelector: #selector(MainviewController.aliasCallBack(_:tags:alias:))
), object: self)
}

func jpushClearAlias() {
    gLog.debug("")
    jpushSetAlias("")
}

func aliasCallBack(resCode:Int, tags:NSSet, alias:NSSet) {
    gLog.debug("resCode=\\"$resCode\\", tags=\\"$tags\\", alias=\\"$alias\\\"")
    //resCode=6004, tags=nil, alias=user-e6882cab-cedf-4335-9b7c-612cd5b4d37d
    //resCode=6003, tags=nil, alias=e6882cab-cedf-4335-9b7c-612cd5b4d37d
    //resCode=0, tags=nil, alias=usr_e6882cab_cedf_4335_9b7c_612cd5b4d37d
    if resCode != 0 {
        let failMsg = "设置JPUSH推送的别名失败: alias=\\"$alias\\\""
        gLog.warning(failMsg)
    }
}

func initWebSocket(){
    dispatchUserInitiated_async{
        gLog.debug("gCurUserItem.wsUrl=\\"$gCurUserItem.wsUrl\\\"")

        //ws://jiandao.im/message/user-972b6796-cc82-4058-b29f-9007115116b9/918fu0uooq18tunpo9h6grp6gk
        self.webSocket = WebSocket(url: NSURL(string: gCurUserItem.wsUrl)!)
        self.webSocket.delegate = self
        self.webSocket.pongDelegate = self
        self.webSocket.connect()
    }
}

```

```

gCurUserItem.wsInitied = true

MainViewController.wsFirstInit = true

gLog.debug("websocket \$(self.webSocket) has initied and connected")

self.jpushSetAlias()
})
}

```

## 正常注册JPush后的log日志

```

didFinishLaunchingWithOptions
2016-03-12 10:35:48.933 | JPUSH | I - [JPUSHService]
----- JPush Log -----
-----JPush SDK Version:2.1.0--build:346-----
-----AppKey:3a1a0d7c1caa422ee5548da3-----

2016-03-12 10:35:48.942 | JPUSH | I - [JPUSHClientController] Action - setup
prevStoredUser Optional("150xxxxxxxx")
prevStotedPassword Optional("111111")
didRegisterForRemoteNotificationsWithDeviceToken
deviceToken<- 89b266fc 31bb2513 20a35909 03a31e0f c8127fef 4946138c 66df7769 617b1c0c>
已注册远程通知 didRegisterRemoteNotification (Function)
deviceTokenStr Optional( 89b266fc 31bb2513 20a35909 03a31e0f c8127fef 4946138c 66df7769 617b1c0c )
didRegisterUserNotificationSettings
notificationSettings UIUserNotificationSettings: 0x14dd5cf00; types: (UIUserNotificationTypeAlert UIUserNotificationTypeBadge UIUserNotificationTypeSound);
2016-03-12 10:35:49.107 | JPUSH | I - [JPUSHSessionController] sis is not on protect
2016-03-12 10:35:49.109 | JPUSH | I - [JPUSHAddressController] Action - sendSisRequest
已连接 networkDidSetup notification NSConcreteNotification 0x14dda9c20 {name = kJPUSHNetworkDidSetupNotification}
已登陆 networkDidLogin notification NSConcreteNotification 0x14de7f470 {name = kJPUSHNetworkDidLoginNotification}
registrationID 13165ffa4e0b5fb6ba3
2016-03-12 10:35:50.104 | JPUSH | I - [JPUSHDeviceTokenReport] upload device token success

```

crifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新: 2023-12-13 14:39:28

# 消息推送开发总结

下面对消息推送相关的知识，进行详细的介绍。

crifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新：2018-02-07 22:34:02

## iOS消息推送

iOS中的消息推送，相对来说，比较特殊一点。

因为iOS中的消息推送，首先是从你的消息推送服务器，发送到苹果官网的服务器，叫做APNS，然后再发送到你的iOS设备，比如iPhone，中的。

也因此，消息推送的时间，往往没法完全的保证，因为还要取决于APNS到你的iPhone的手机中的推送时间，虽然多数情况下较快，但是也还是有时候会比较慢的。

## iOS的消息推送证书

iOS的app要支持消息推送的话，需要：

- (Xcode中) 设置APP支持消息推送
- 申请相关的消息推送的证书
- 在消息推送服务器（比如JPush的后台管理页面）中（上传）配置对应的推送证书

且需要注意的是，iOS的消息推送证书分两种：

- 开发环境的消息推送证书：用于平时开发调试期间使用
- 正式（生产）环境的消息推送证书：用于生产环境，已上线的APP使用
  - 理论上来说：生产环境的消息推送证书，在你创建到时候，就提示了，其实也可以被用于开发环境
    - 只不过平时还是很少混用，以免开发环境推送的消息影响到线上的环境

## 生成消息推送所需证书：.p12 文件

对于想要去申请获得消息推送所需的证书，简要概括步骤是：

- 登录苹果开发者网站后台管理页面
- 选择是Develop开发环境还是Production生产环境
- 选择APP ID
- 创建和上传CSR文件
- 生成 .cer 证书文件
- 用Mac中的 钥匙串 打开 .cer 证书文件
- 导出为 .p12 推送证书文件
  - 期间需要设置证书的密码，记得保存好，以后万一给别人使用时，需要密码才能用

详细步骤如下：

### 登录苹果开发者网站后台管理页面

进入：[苹果开发者 后台管理页面](#)

会自动跳转到：

Sign in with your Apple ID - Apple Developer

<https://idmsa.apple.com/xxx>

之类的地址，输入Apple ID和密码即可登录

### 选择是Develop开发环境还是Production生产环境

点击Certificates右上角的加号+：

The screenshot shows the 'Certificates, Identifiers & Profiles' section of the Apple Developer portal. On the left, there's a sidebar with a dropdown for 'iOS, tvOS, watchOS'. Under 'Certificates', the 'Development' option is selected. The main area is titled 'iOS Certificates (Development)' and shows a single certificate entry:

Name	Type	Expires
[Redacted]	iOS Development	Apr 08, 2017

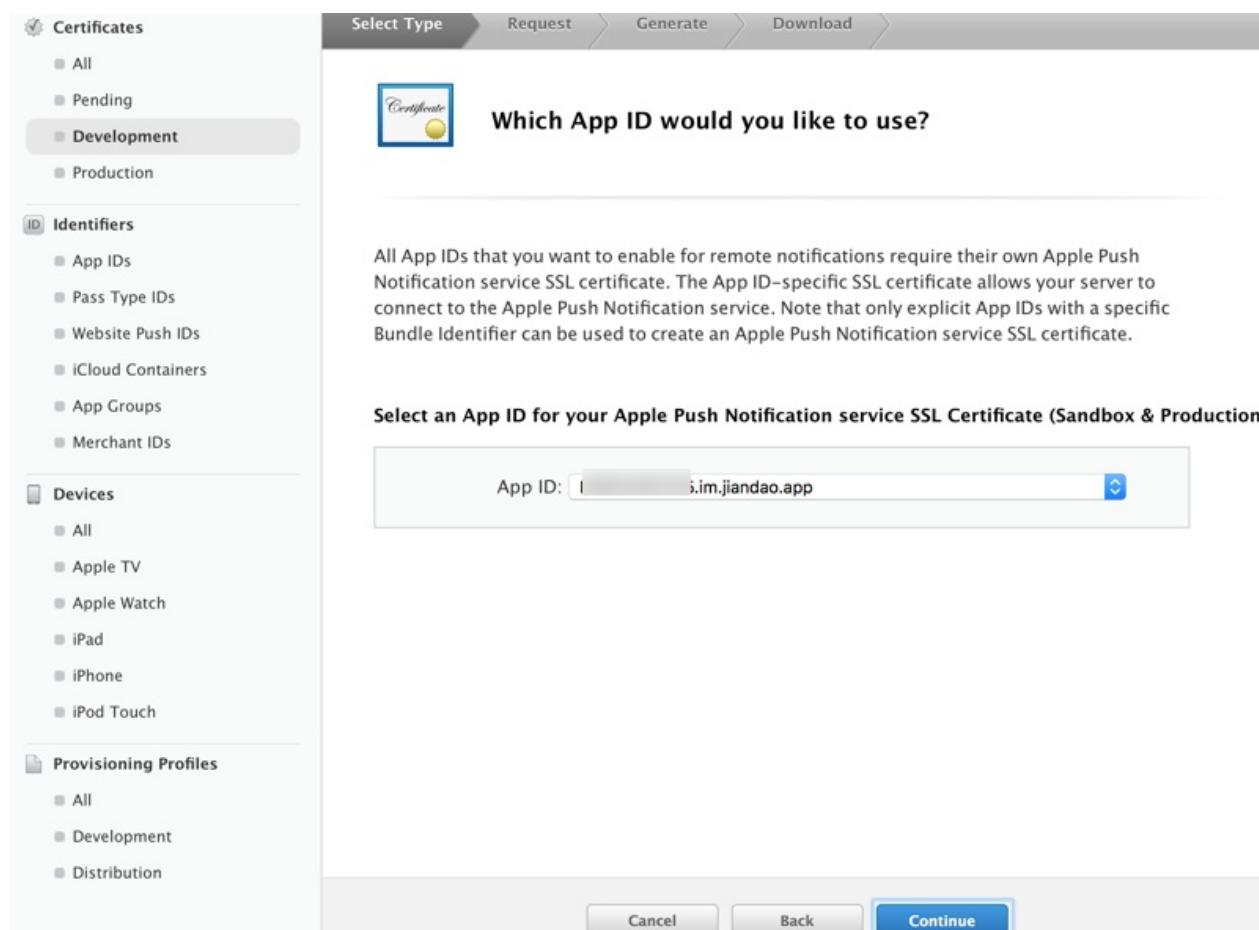
然后选择对应的环境：

- 开发环境：选择 Development -> Apple Push Notification service SSL (Sandbox)
  - 用于平时在开发环境测试消息推送使用
- 生产环境：选择 Production -> Apple Push Notification service SSL (Sandbox & Production)

◦

## **选择对应的APP ID**

然后选择对应的APP ID：



表示这个消息推送证书是给哪个APP用的。

## 创建和选择CSR文件

**About Creating a Certificate Signing Request (CSR)**

To manually generate a Certificate, you need a Certificate Signing Request (CSR) file from your Mac. To create a CSR file, follow the instructions below to create one using Keychain Access.

**Create a CSR file.**

In the Applications folder on your Mac, open the Utilities folder and launch Keychain Access.

Within the Keychain Access drop down menu, select Keychain Access > Certificate Assistant > Request a Certificate from a Certificate Authority.

- In the Certificate Information window, enter the following information:
  - In the User Email Address field, enter your email address.
  - In the Common Name field, create a name for your private key (e.g., John Doe Dev Key).
  - The CA Email Address field should be left empty.
  - In the "Request is" group, select the "Saved to disk" option.
- Click Continue within Keychain Access to complete the CSR generating process.

**Cancel** **Back** **Continue**

点击Continue继续：

**Generate your certificate.**

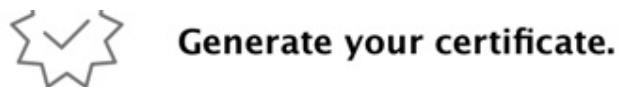
When your CSR file is created, a public and private key pair is automatically generated. Your private key is stored on your computer. On a Mac, it is stored in the login Keychain by default and can be viewed in the Keychain Access app under the "Keys" category. Your requested certificate is the public half of your key pair.

**Upload CSR file.**  
Select .certSigningRequest file saved on your Mac.

**Choose File...**

**Cancel** **Back** **Continue**

然后选择之前创建好的CSR文件：



---

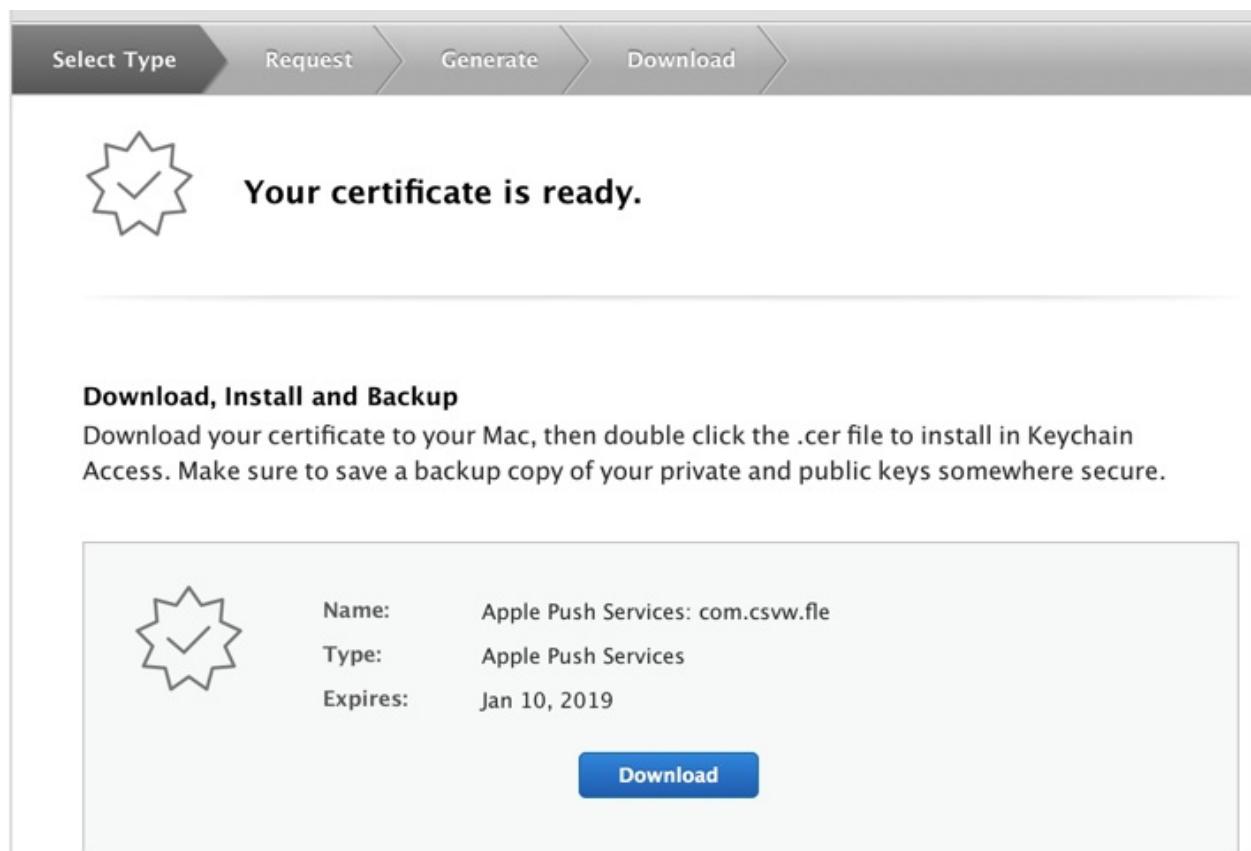
When your CSR file is created, a public and private key pair is automatically generated. Your private key is stored on your computer. On a Mac, it is stored in the login Keychain by default and can be viewed in the Keychain Access app under the "Keys" category. Your requested certificate is the public half of your key pair.

**Upload CSR file.**

Select .certSigningRequest file saved on your Mac.



然后即可成功创建对应的证书：

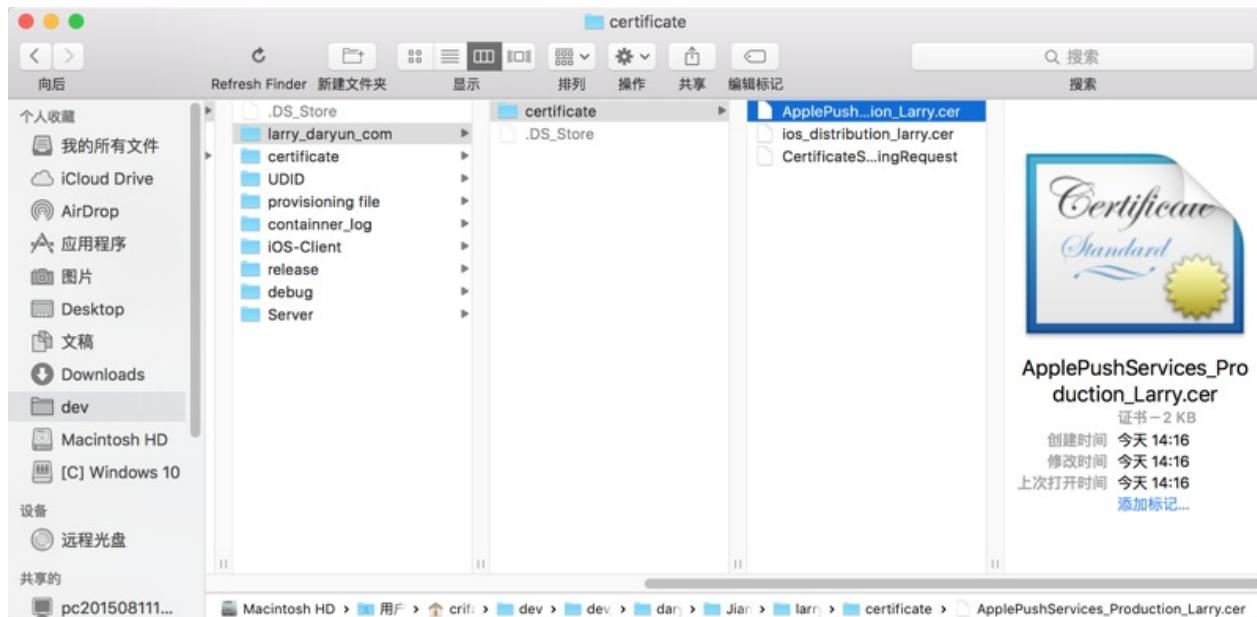


## Documentation

For more information on using and managing your certificates read:

[App Distribution Guide](#)

然后可以点击下载得到 .cer 文件:

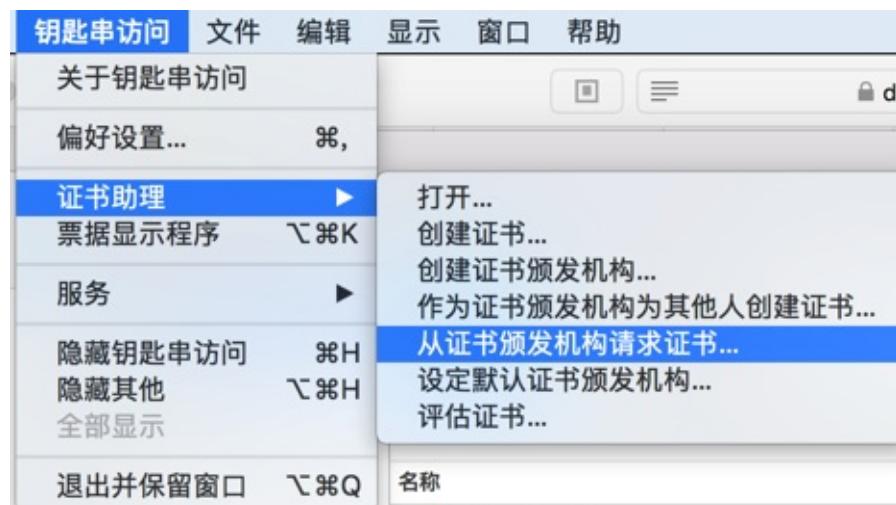


## Mac中创建CSR文件

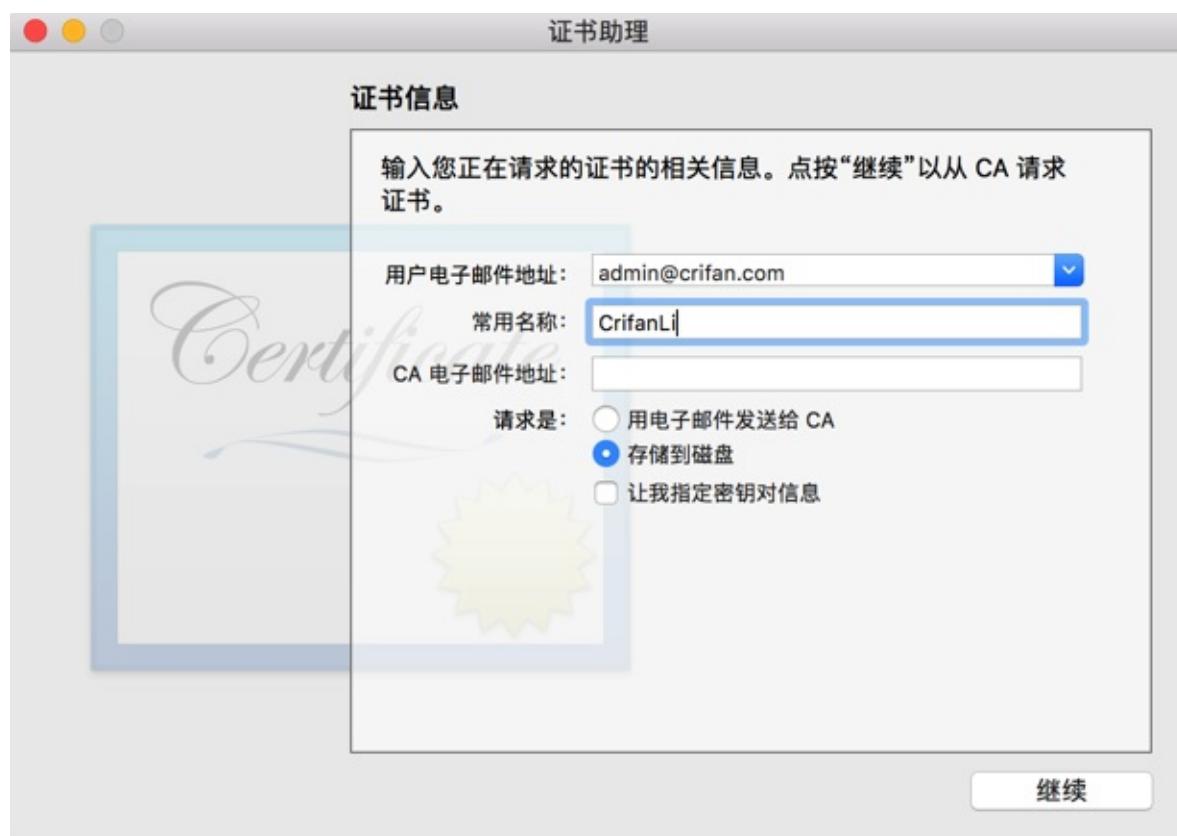
如果没有CSR文件，则去Mac中创建：

去Mac中打开 钥匙串，然后：

钥匙串访问 -> 证书助理 -> 从证书颁发机构请求证书 :



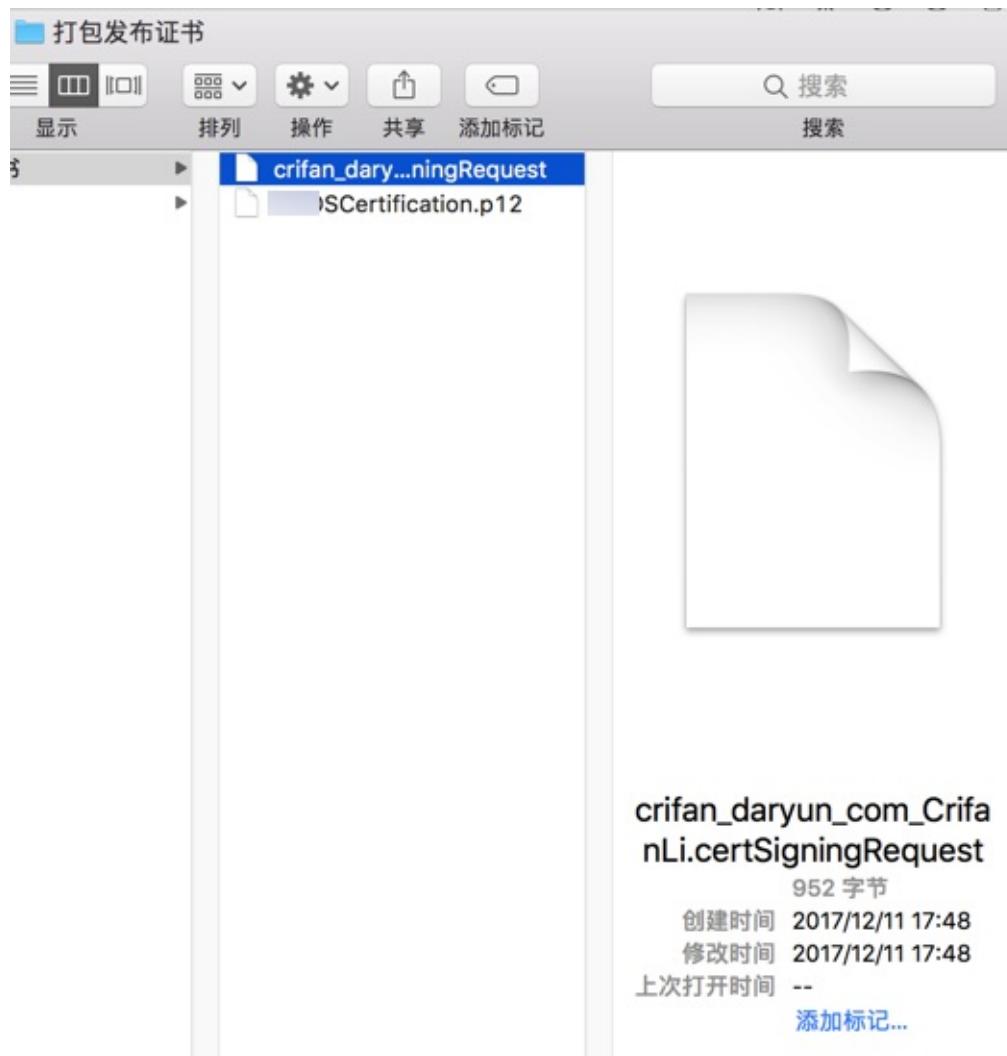
继续去在证书助理中，填写信息：



然后继续，就可以创建成功了：

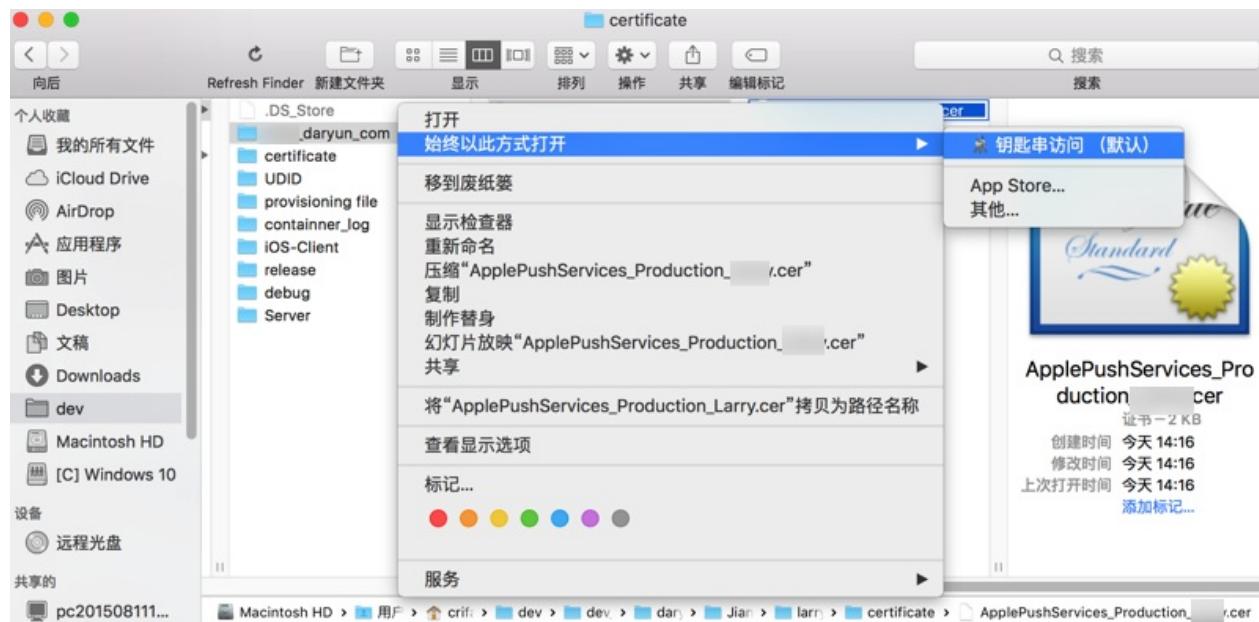


然后找到刚创建好的 .certSigningRequest 文件，比如 crifan\_daryun\_com\_CrifanLi.certSigningRequest



## 用钥匙串把 .cer 证书转换导出为 .p12 消息推送证书文件

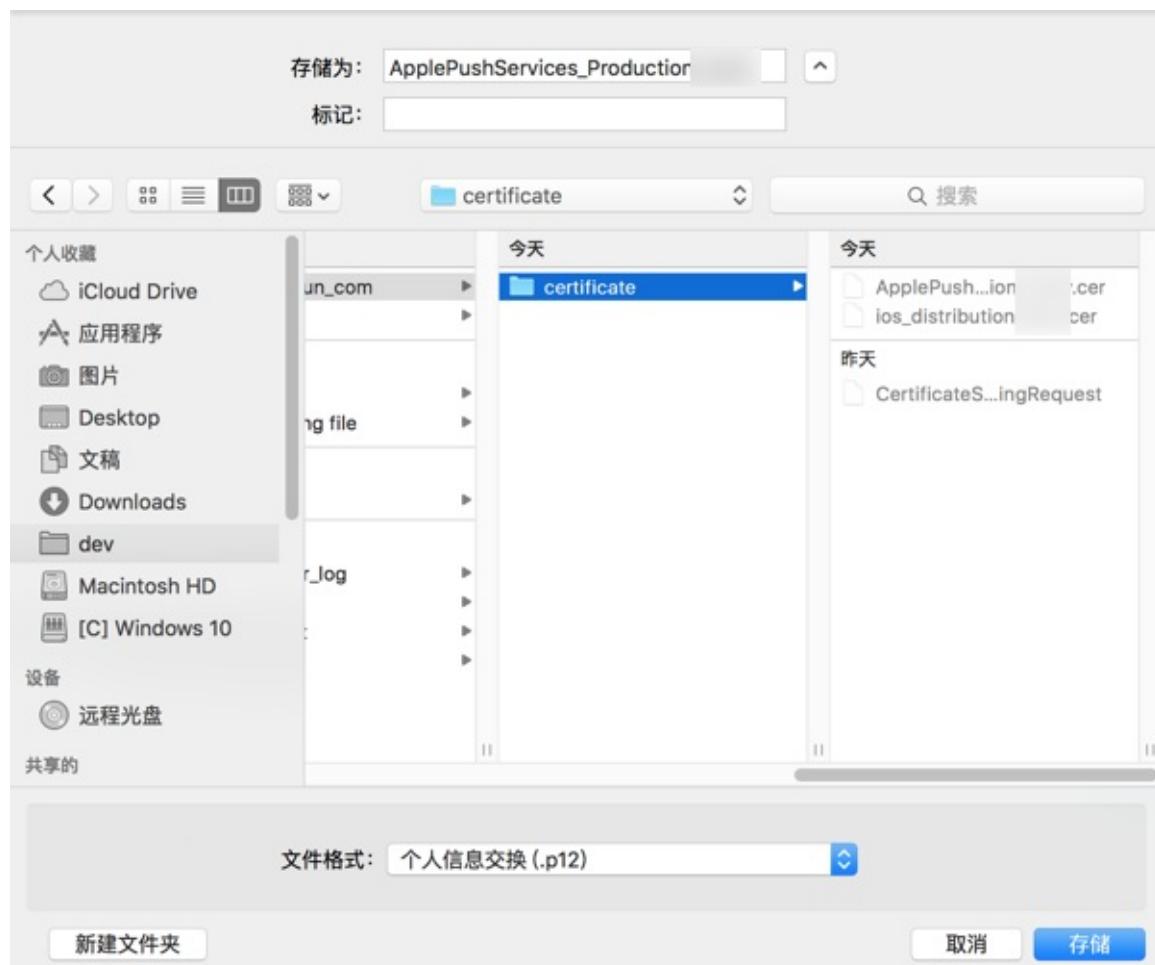
Mac中用 钥匙串访问 打开 .cer 文件：



然后右击该证书，选择 导出xxxx：



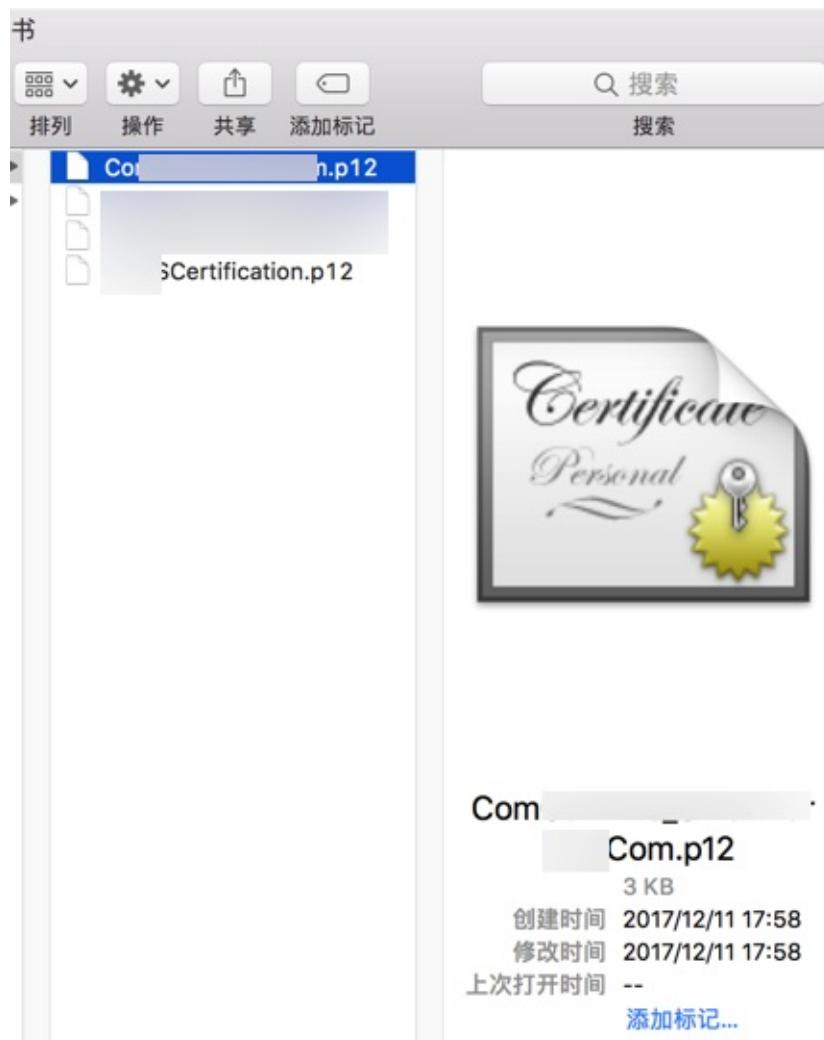
即可去导出 .p12 文件：



导出时需要设置密码:



然后即可得到消息推送所需要的 p12 证书文件:



crifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新: 2018-02-08 22:13:09

## Android消息推送

### Android的离线推送

不同的品牌手机中的Android系统，有的还针对消息离线推送进行了特殊的定制，比如：

- 华为
- 小米

在系统级别上支持：app没有运行的情况下，也能收到离线消息。

所以总结Android端的离线消息推送，以及相关的应用保活：

注：应用保活=应用（在被系统杀掉后，仍然还能在后台运行）保（持）活（跃状态）

- 离线推送：
  - 优点：目前仅支持小米手机和华为手机，可以（借助小米 / 华为的androdi系统本身）实现app被杀掉仍能收到推送。
  - 缺点：覆盖度有限，其他机型没法支持
- 应用保活：
  - 优点：对于普通的用户的杀掉正在运行的app时，基本上可以实现app
  - 缺点：第三方管理软件理论上应该还是可以把即使用了保活的app杀掉

-> 如果想要实现Android端的离线推送和应用保活的话，需要引导：

- 用户：把app加入系统和第三方管理软件的白名单
- 系统：用户自己把app加到系统的app白名单中，app进入后台后不会被杀掉
- 第三方管理软件：360管家，腾讯管家等，把app加入白名单，则清理时不会把app杀掉

### 华为的Android的离线推送

华为的Android端的离线推送叫做：HMS推送服务

根据官网教程去配置完后，即可。

在满足条件的华为设备上就可以使用华为推送接收离线推送通知了；这里的满足条件是指：华为设备必须安装2.4以上的华为移动服务，以及开启当前 app 的自启动权限；

注意：华为EMUI 4.0需要把app自动启动权限打开才能收到推送。

详见：[GCM、华为推送 \[环信开发文档\]](#)

### 小米的Android的离线推送

通过 MIUI 系统的支持，实现Android端的离线消息推送

详见：[小米开放平台 - 消息推送](#)

crifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新：2018-02-08 21:13:54

# 调试移动端Webview/H5页面

在移动端开发期间，可能会遇到：

对于APP来说，里面的内容是HTML5的web页面。

而想要调试手机端/移动端的APP内部的H5的页面。

此处就来介绍，如何在电脑上调试手机端的APP内的H5页面。

crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：2018-02-08 21:28:58

## 调试iOS的Webview

把iOS设备，比如iPhone手机连接到Mac上之后：

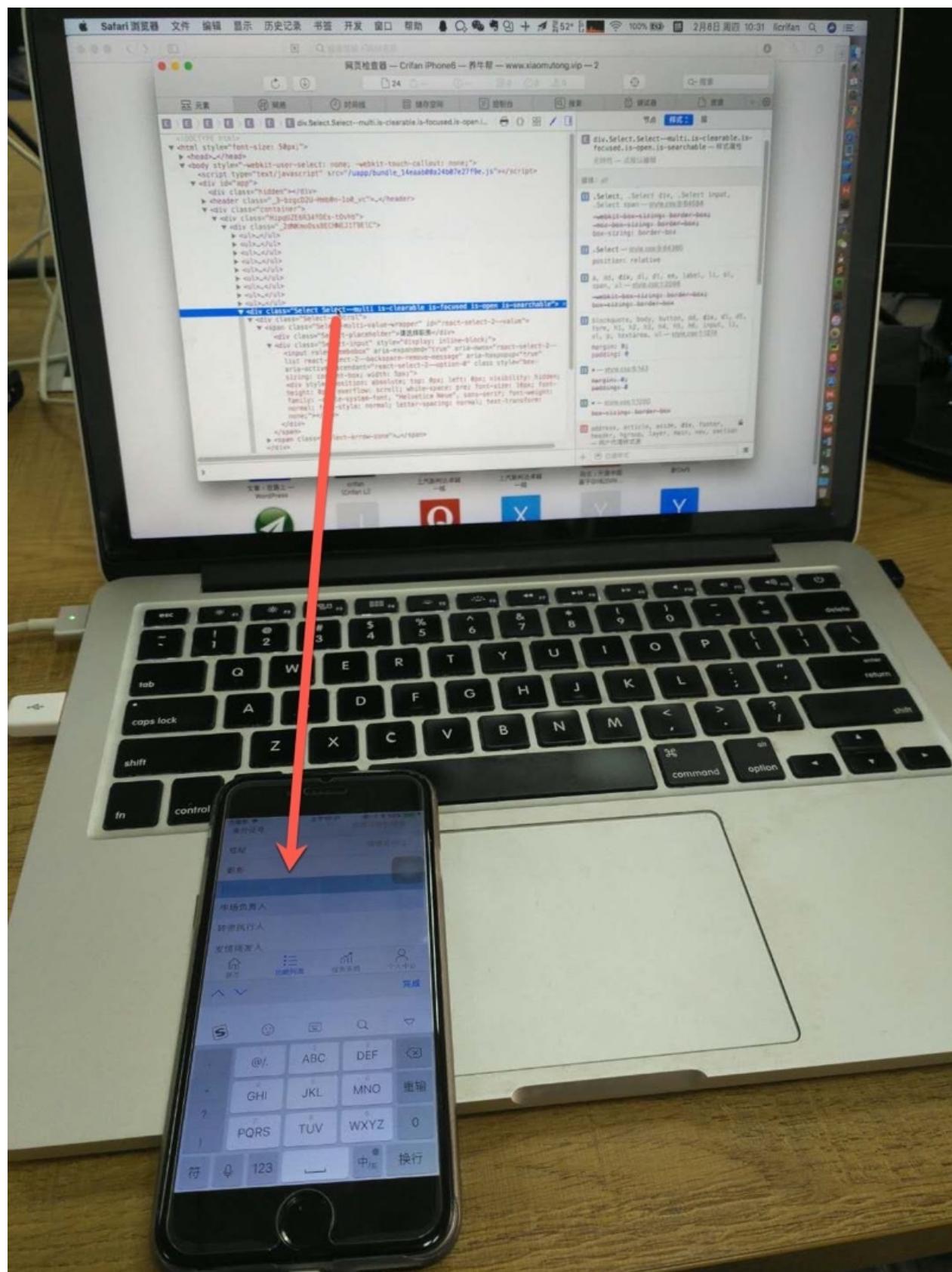


打开Mac中的Safari浏览器，在已经开启开发模式（显示出开发菜单）后，可以在：

开发 -> xxx iPhone -> 选择对应的页面：

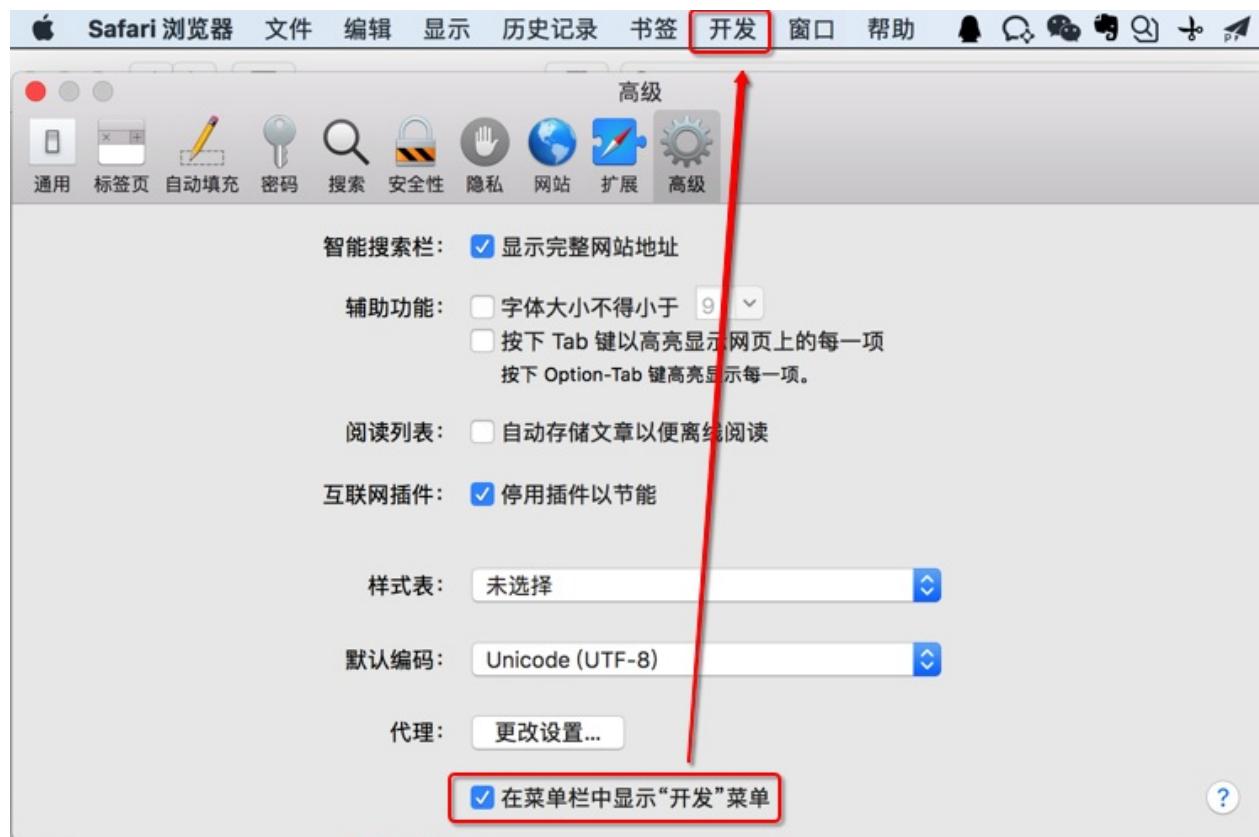


即可去调试手机移动端中的（WebKit内核的）页面。鼠标移动到对应的html的元素上之后，手机端的页面的对应部分即可实时显示出背景色了：



## Safari中显示 开发 菜单

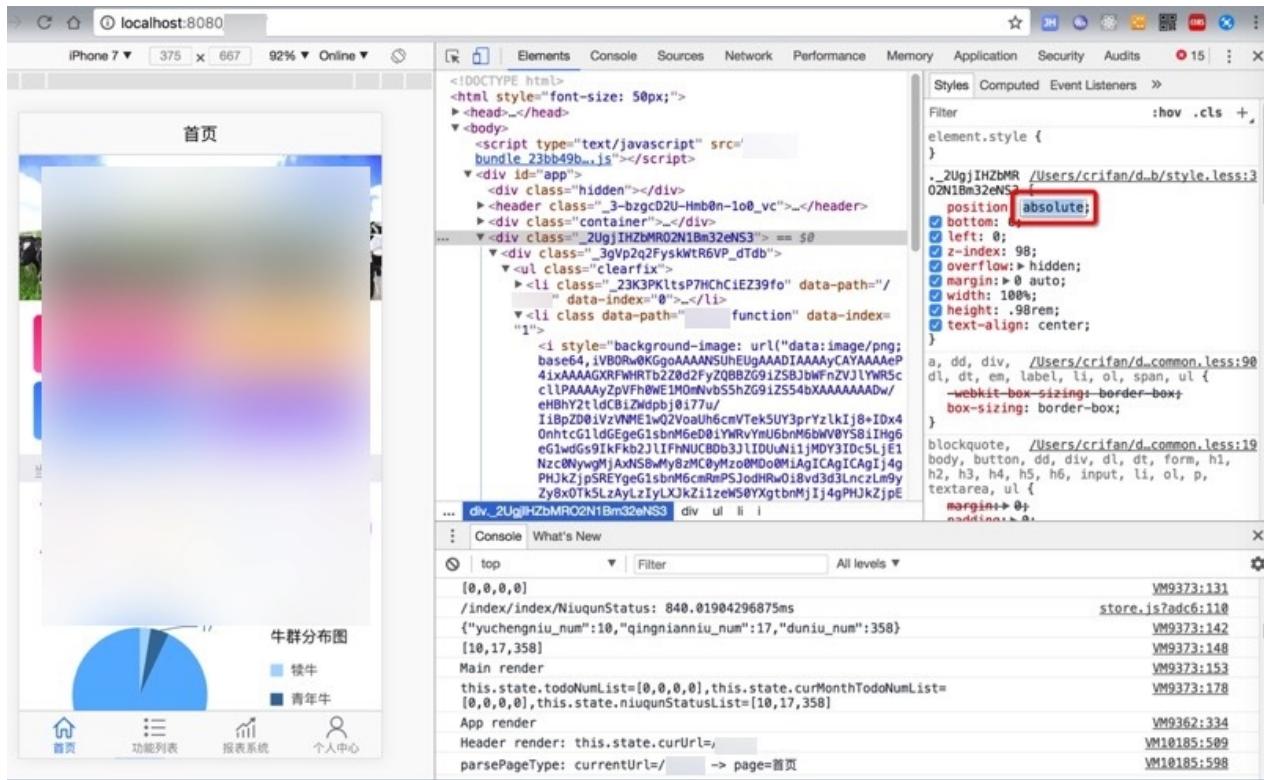
Safari -> 配置 -> 勾选 在菜单中显示"开发"菜单 -> 菜单中就可以看到 开发 菜单了：



crifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新: 2018-02-08 10:50:06

## 调试Android的Webview

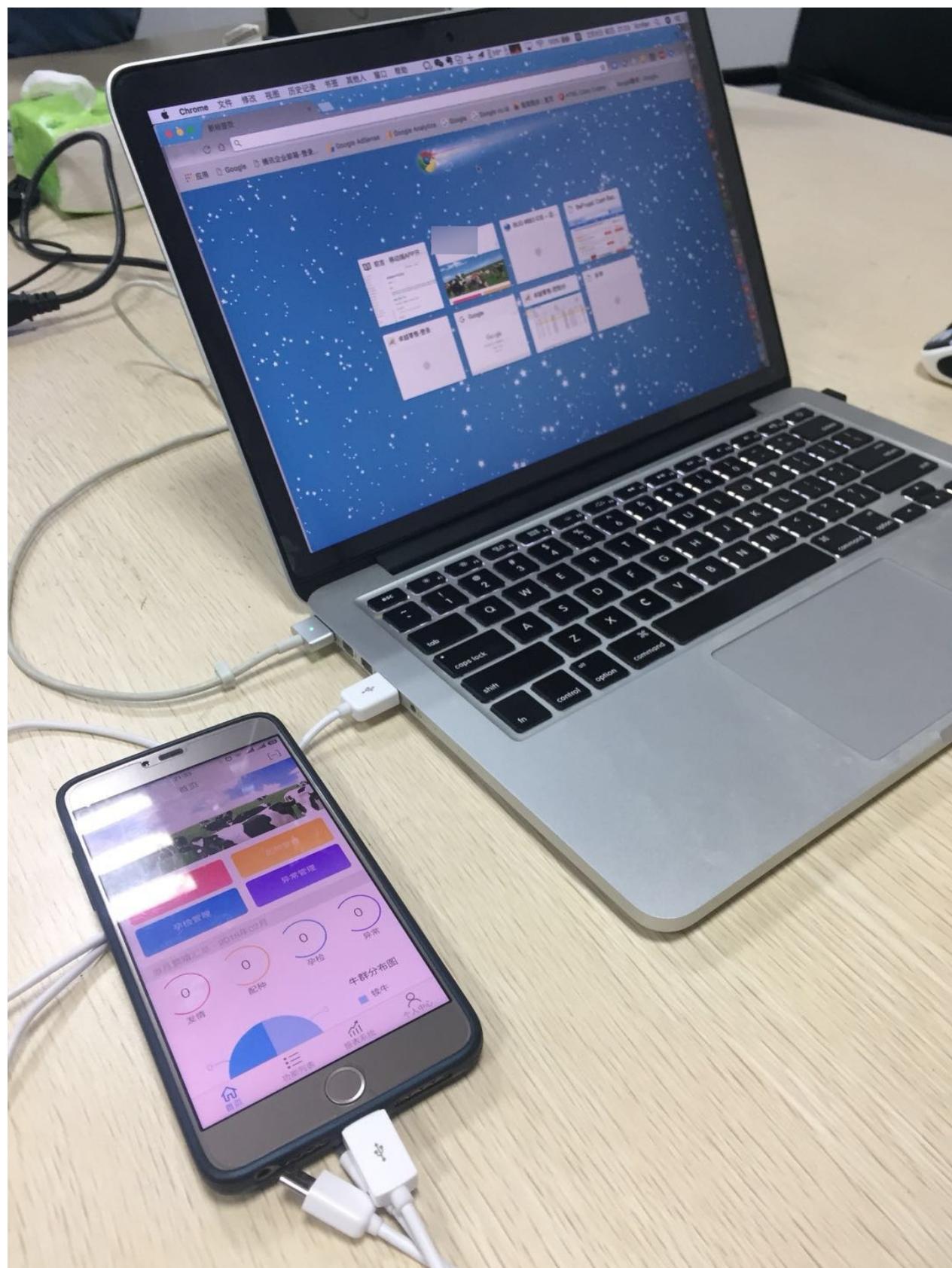
对于Chrome来说，除了在电脑上用其调试网页，包括适配了移动端的H5页面，比如：



还可以用其来调试手机端的H5页面。

## 用Chrome在电脑端调试手机端APP内的Webview页面

比如在Mac上，也当然先要把Android的手机用USB数据线连接到电脑上：

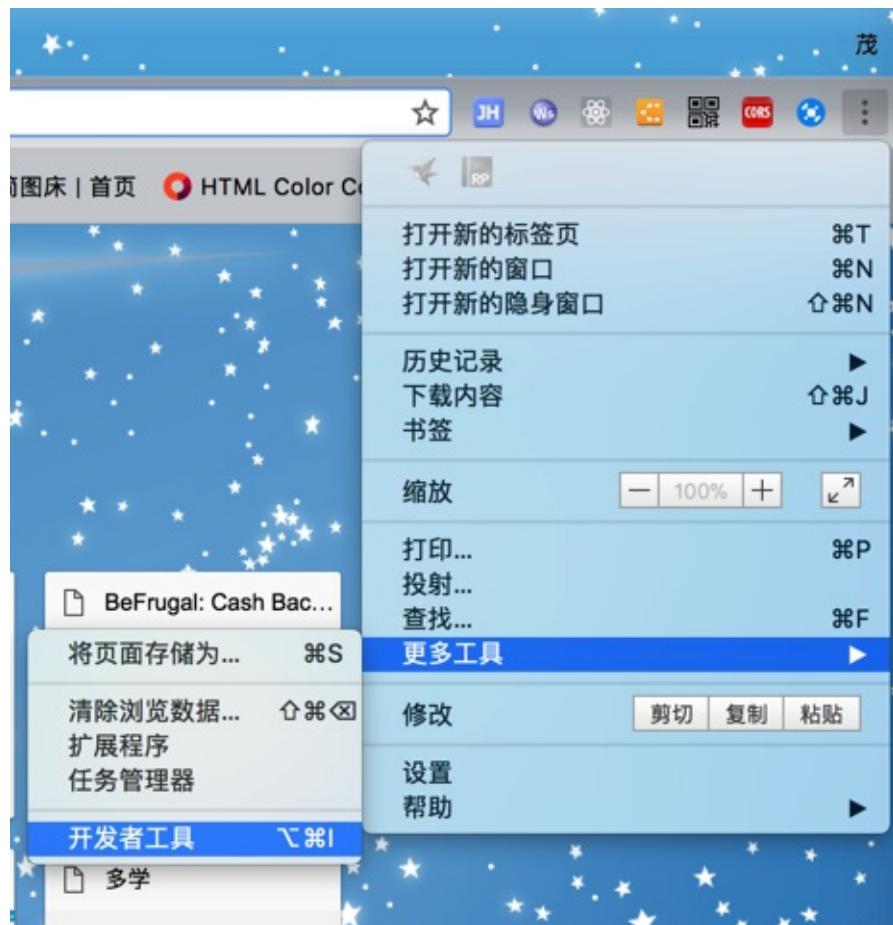


然后先确保Android端打开你要调试的APP，把对应的包含了网页的界面打开，其内部是调用了Webview的内核打开的页面，比如：



然后去Mac中，打开：

Chrome -> 更多工具 -> 开发者工具：



然后即可打开调试页面：

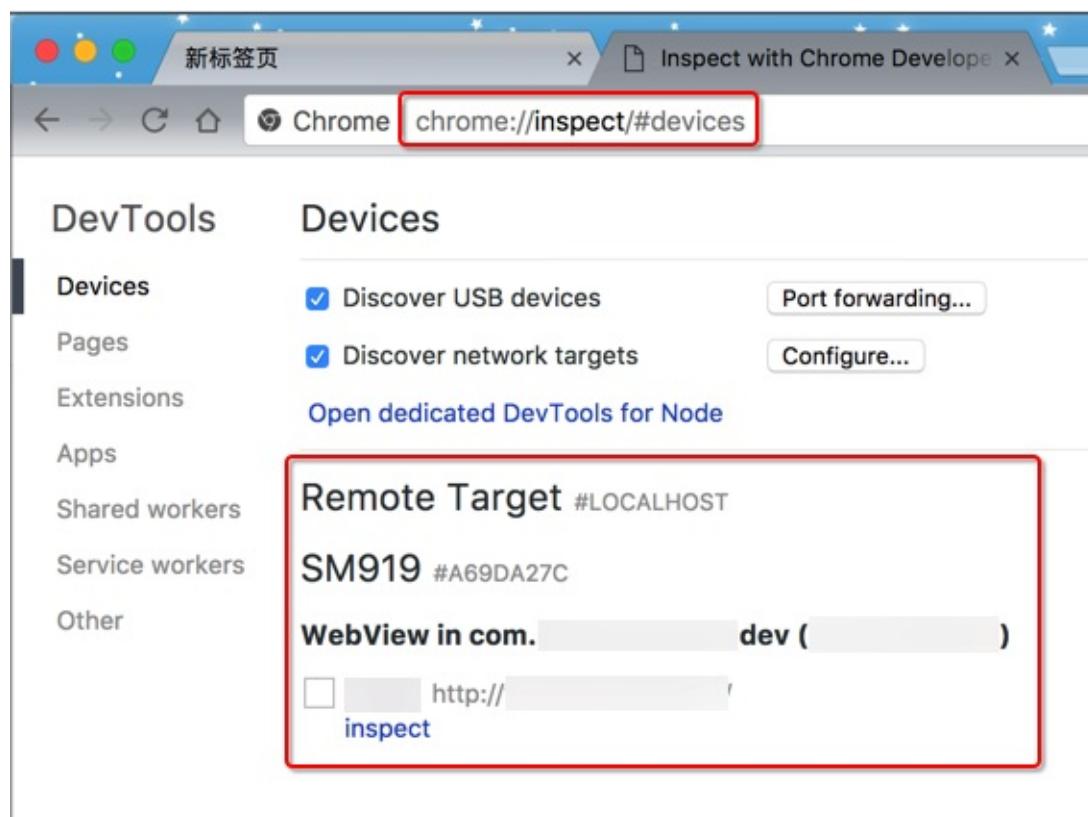
然后对于打开远程设备去调试页面，有两种方式：

### chrome://inspect中找到远程Android设备

直接Chrome中输入：

```
chrome://inspect
```

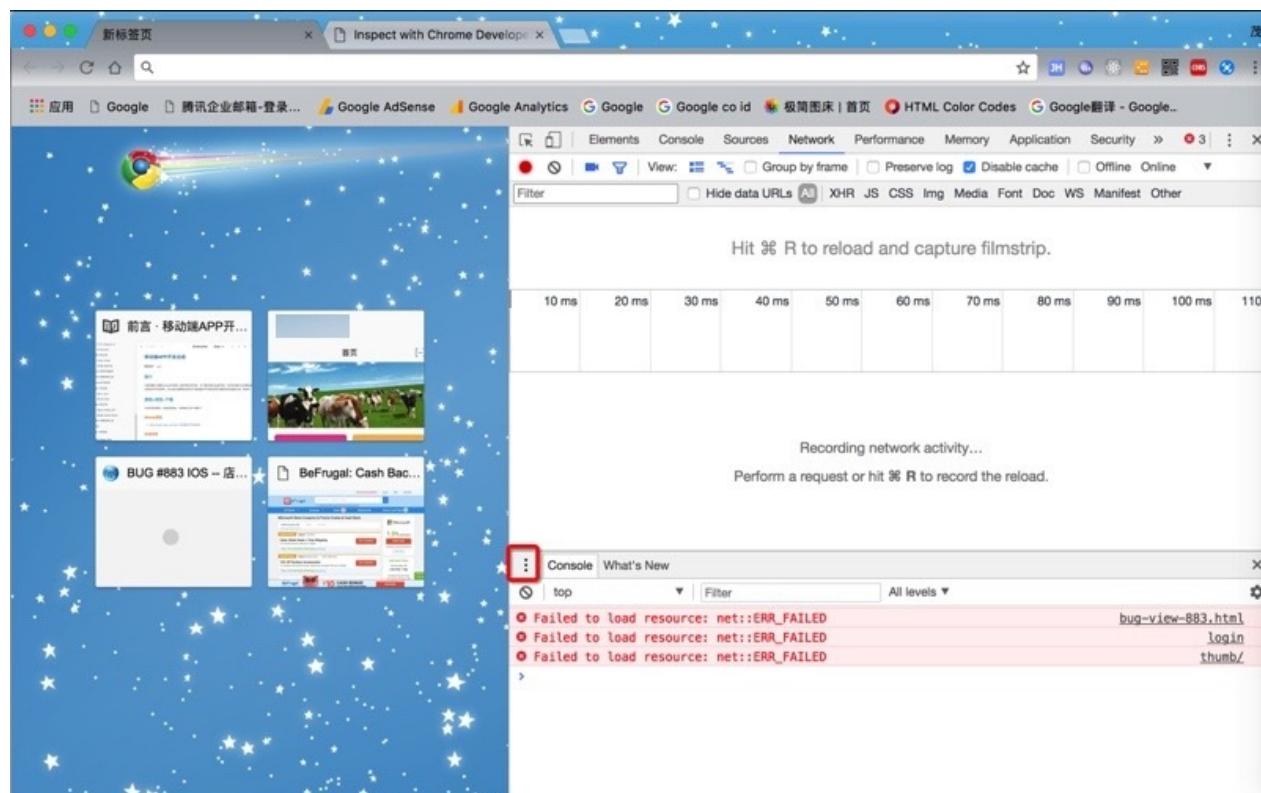
即可看到已插入电脑的Android手机中的App，包含对应的页面



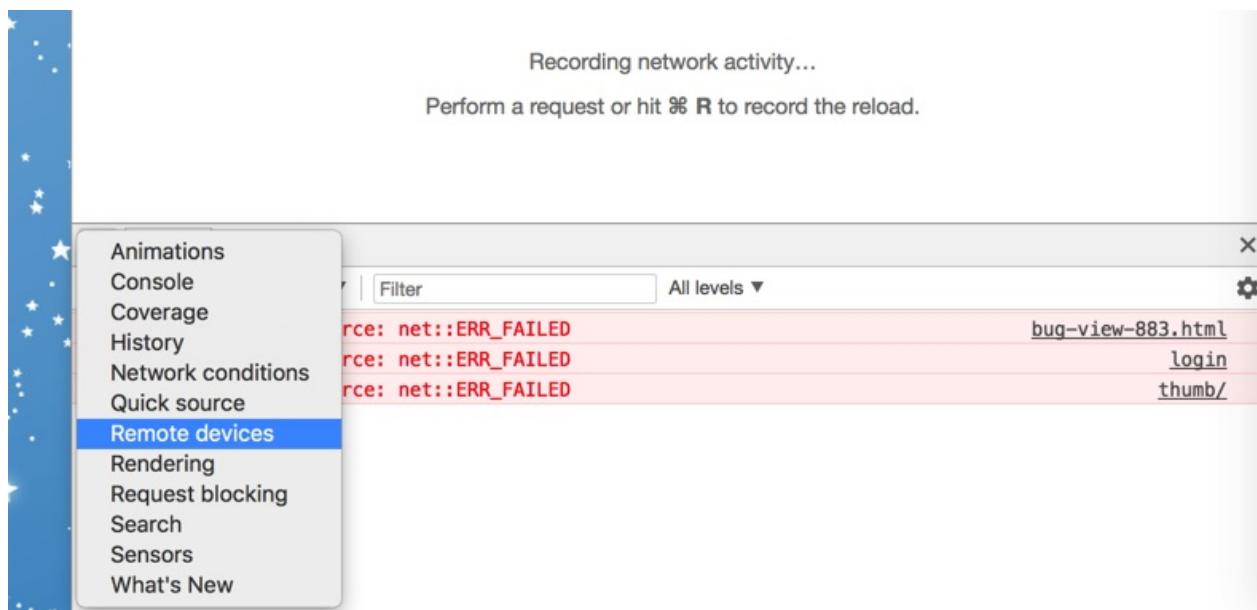
点击 `Inspect` 即可开始调试。

## console栏中Remote Devices

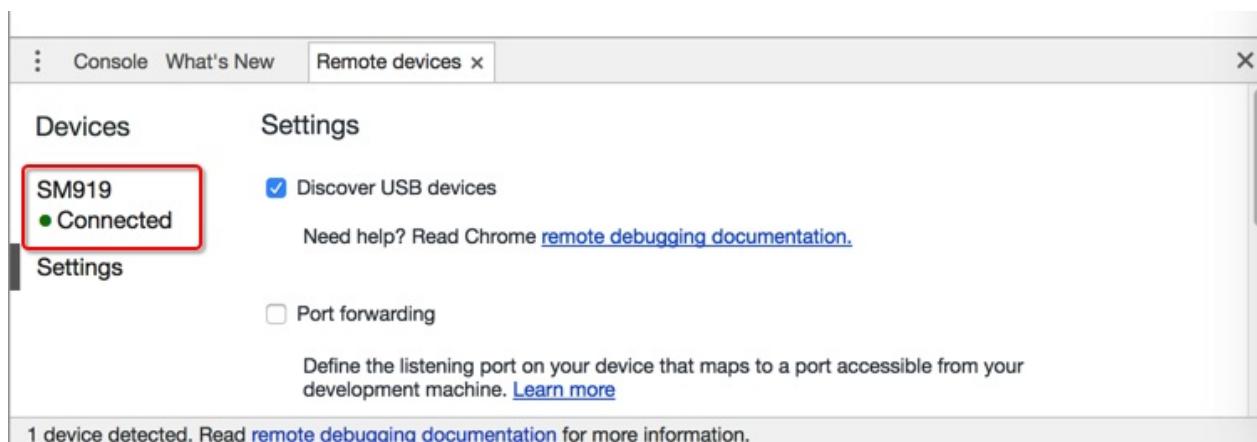
在开发者工具的Console栏中，点击三个点：



然后选择 Remote devices :

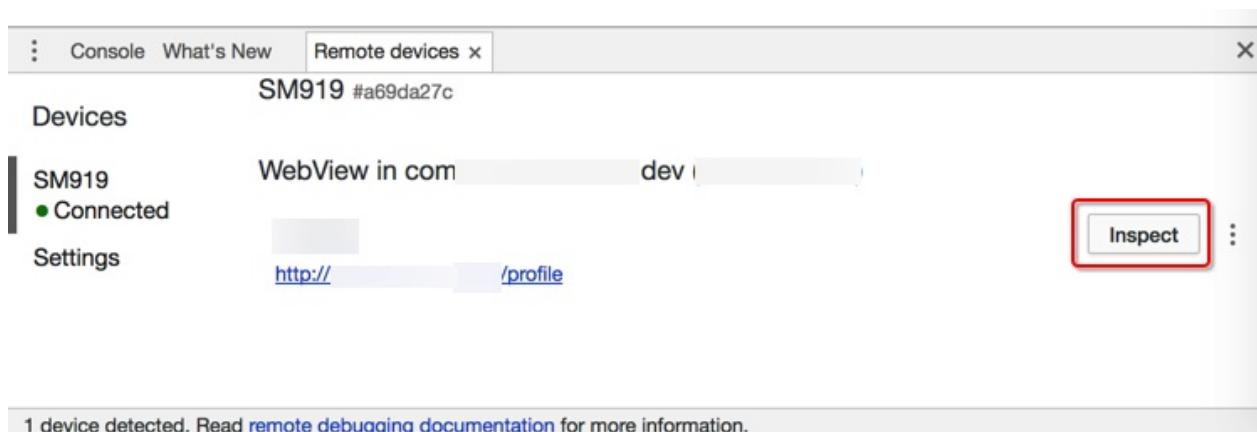


然后，如果本身Android手机已经插入电脑了，则就可以看到：xxx Connected，比如此处的SM919 Connected：



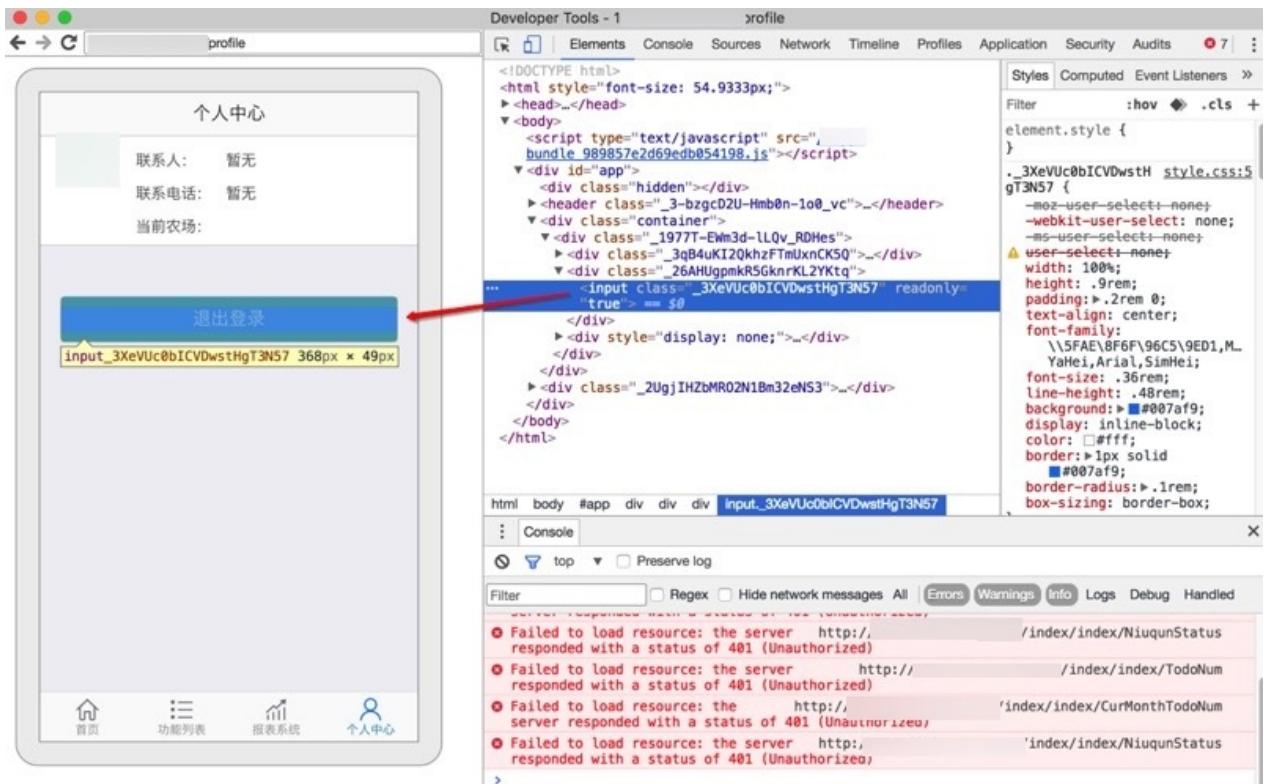
其中SM919是锥子M1L的内部设备编号。

点击了connected的Android设备后，即可看到对应的设备和页面，点击Inspect：



即可进去调试页面：

可以鼠标移动到html的元素，以便于查看详细参数，看看是否符合期望：



且调试效果是实时的：

- 当手机端切换页面时，PC端的页面内容也会立刻刷新

- 此APP被切换到后台时，对应的页面会显示Inactive，表示处于没有激活的状态：

◦

如此，就可以愉快的，在电脑端(Windows/Mac)去调试(通过数据线连接到电脑的)远程Android手机端的APP内部的网页（内部是基于Webview的页面）了。

crifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新: 2018-02-08 22:11:47

# 通话录音

下面介绍一下移动端的通话录音方面的经验总结。

crifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新: 2018-06-18 16:02:57

# iOS通话录音

ios 由于系统本身的限制，出于不收集用户数据，起到保护隐私的安全考虑，所以不提供，也无法实现在通话期间直接的支持录音。

所以iOS中的通话录音，都是第三方，通过网络或网络+运营商的方式，实现通话录音的。

下面总结一下已知的一些方案提供商：

## iOS通话录音方案提供商

### 网易云信

之前也发现过[网易云信](#)支持iOS的通话录音，但是后来却（好像是因为 20-1=? 大的缘故）而停售，停止服务了。

### 上海飞语网络

#### 上海飞语网络的联系方式

上海飞语网络科技有限公司

公司地址：上海市浦东新区博霞路22号106

联系总机：012-51954962

商务合作：021-51954962转802

QQ交流群：470250528

## 用飞语云平台实现iOS通话录音

### 使用场景

app内部点击拨号按钮，经服务器发起请求相应参数，其中【是否录音参数】必须传值为true，通话才能够进行录音，通话完成会返回该通话已录音，根据会返回一个录音下载地址，飞语需进行转码处理（一般地址下发时间通知为几分钟到几十分钟不等，视通话时长而定），但飞语不提供后台管理页面，需我们自行采取定时任务请求相应的下载相应录音，录音保存有效期为1个月左右，故我们需自行开发后台管理界面对返回的录音文件及参数进行保存管理，方便之后查询管理。

### 基本流程

关于使用飞语网络实现iOS录音的话，内部其实是利用WiFi网络+第三方网络运行商，实现的录音。

其中关于第三方服务，简称：[PSTN落地线路](#)，[落地线路](#)

飞语：

- 想要测试PSTN落地线路，直接打手机号
- 需要提供自己的APPID，然后让飞语后台开通这个功能的

整个逻辑是：

- 主叫：是用网络，装了比如某iOS的app，调用飞语SDK，内部是VOIP的UDP语音，就像微信的语音
  - 利用的是网络+飞语的SDK
- 被叫：无需网络，无需安装任何app，就是自己的手机号，正常接听就像别人打给你一样的通话了

- 即可主叫呼叫被叫，被叫显示的号，是主叫的飞语SDK中可以（任意）设置的
- 利用的是（飞语的合作伙伴，香港的一家，类似于移动，电信等网络提供商提供的）落地线路

基本流程：

1. 主叫方：138，调用 `[engineKit dialPeer:@"calleeUid" CallerUid:@"callerUid" OptionData:nil];` 类似于微信：点击语音通信
2. 确保被叫方app处于前台
  - 一般是：138发送离线消息推送push给139，139点击离线消息，启动app，确保处于前台
  - 类似于被叫方启动微信确保微信在线且处于前台
3. 被叫方139，调用 `[engineKit calleePrepare:@"uid" prepareSuccess:nil];` 类似于被叫方微信中点击接听

## 调用第三方网络服务举例

我15012345678 呼叫你 18656781234

我知道你的号 就可以在飞语sdk中调用时写上：

+8618656781234

就可以呼叫到你

你看到的我的号码 我是可以随意设置的，比如+8613900001111

当然我也可以设置你看到的号码 就是我的真实的手机号 比如+8615012345678

其中，被叫方看到的电话号码，有两种：

1. 方案1:主叫方138打电话给被叫方139，139看到的电话号码是某个公司的座机号码
  - 前提：主叫方和被叫方都是网络在线-》所以才能通过UDP实现VOIP的语音通信-》相当于两人微信都在线
  - 费用：飞语自己：1000分钟 / 5元
2. 方案2:主叫方138打电话给被叫方139，139看到的电话号码是138的号码
  - 前提：主叫方需要网络在线-》相当于主叫方的微信在线
  - 被叫方手机可以没有网络-》因为是通过第三方服务去实现拨打被叫方的电话的-》所以被叫方才能看到主叫方的电话号码（而不是固定的座机号码之类的）

## 注意

- 飞语只提供技术，不提供落地线路。
- 如果采用方案2，需要再去联系 提供落地线路的 国外公司
  - 可以问 简工程师 18917930061 要对方QQ号去咨询细节
  - 具体费用见下面的总结

## 具体实现方式

由于飞语官网没有足够完善的demo

具体方案可参考，之前自己在折腾后，写的demo：

[crifan/feiyuiOSDemo: 飞语云平台iOS点对点通话录音Demo](#)

## 计费

关于网络流量和第三方服务，都是需要一定费用的。

下面就来解释一下大概费用是如何计算的：

A网络(APP) 通过 飞语SDK 直接拨打 B手机号/座机等，的费用是：约**8.5分钱/分钟**

具体包含2部分：

- 飞语本身的费用： $5元/1000分钟 = 0.5分钱/分钟$ 
  - 飞语平台自带送了5元，可以测试1000分钟通话
  - 如果费用不足，可以在飞语后台管理页面充值
- 落地线路收费： $0.012美元/分钟 = 大概8分钱/分钟$ 
  - 合作公司是一家新加坡公司
  - 和飞语合作，有送10分钟供测试
    - 但是需要提供APPID，让飞语后台开通此PSTN功能才行
  - 费用不足，需要充值，最低500美元起

## 其他技术细节

### mp3录音文件

关于录音的问题，情况是：目前飞语的SDK正在调整和优化期间

在调用接口之前，option参数中设置了通话要录音（MP3文件）

在通话结束后，想要获得录音文件，暂时有2种方式：

1. 联系飞语工程师，提供APPID和被叫号码，对方可以帮你从后台找出来，发给你录音文件
2. 配合飞语的流程，提供自己的服务器，自己服务器实现两个接口
  - 录音成功通知：飞语调自己服务器，用于获取callID等参数
  - 获取录音文件下载地址：自己服务器调用飞语API：
    - <http://api.feiyucloud.com/api/getRecordDownUrl>
    - 获取录音文件下载地址

具体详情和接口描述，请咨询飞语官方，要具体文档。

此处由于只是技术可行性研究，就暂时用第一种办法，要了录音MP3文件听听效果，就可以了。

### app上线

之后正式上线时，需要用到苹果的 VOIP 的 push

在app的项目配置中，开启 VOIP 的 Push，估计也要去苹果开发者中心，后台创建对应的证书

具体技术自己去找，比如：[PushKit的使用 - 简书](#)

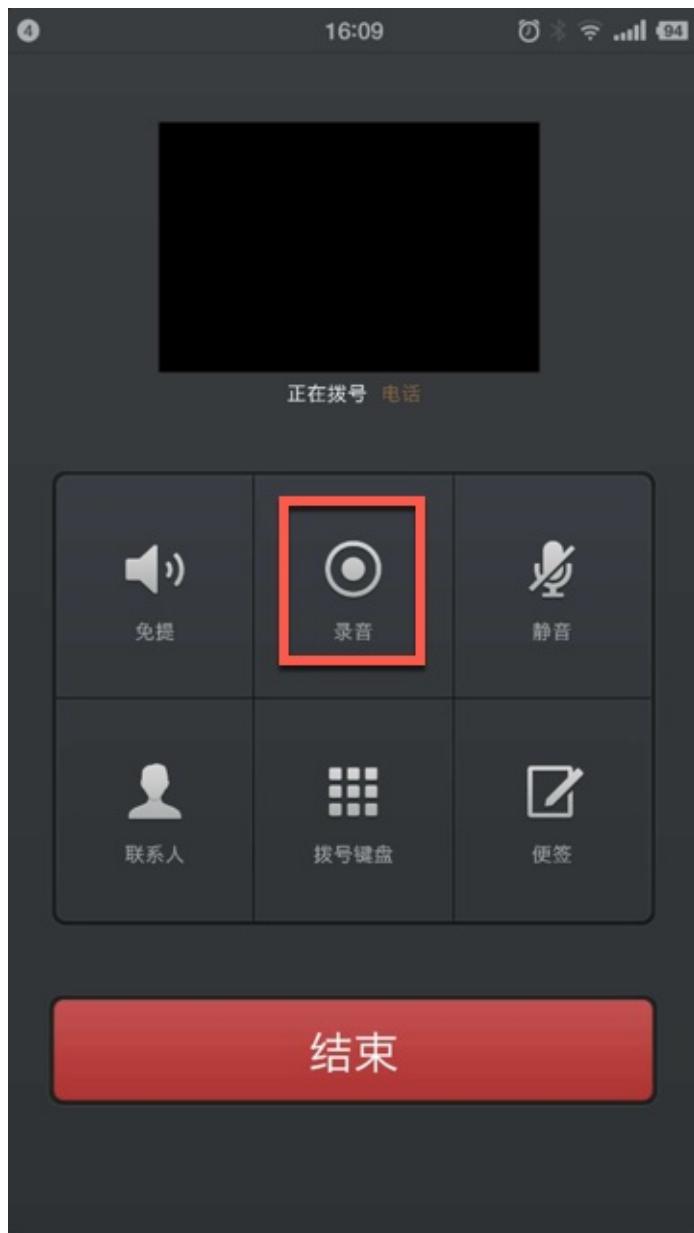
crifan.org，使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：2018-06-18 16:50:12

# Android通话录音

android端实现电话录音，相对比较容易，因为系统没有限制通话时录音的权限。

相对来说，具体实现方案，往往可以找到很多，此处不赘述。

所以，自带通话录音的的android手机也很多，比如我的 锤子M1L：



# APP托管总结

在发布到其他app托管平台期间，会涉及到给对于页面的命名

## 项目发布时页面命名规则

比如要将一个项目发布到 `fir.im` 时，给页面的短写的名字，命令，就和项目起包名，很类似。

建议 `fir.im` 中路径命名规则为： **项目名+平台+环境**

Android: <https://fir.im/XxxYyyiOS>

iOS: <https://fir.im/XxxYyyAndroid>

比如：

- 项目名：驼峰法命名 (camel casing)
  - `Xxx`
- 平台：iOS或Android等
- 环境：
  - 开发版= `dev`
  - 测试版= `test = tst`
  - 正式版
    - `production = prd`
    - `release = rls`

crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新: 2023-12-13 14:27:27

## APP托管平台

某个版本的app调试完毕后，往往为了协同工作，比如：

- 别的测试人员测试该版本app
- 别的客户人员需要下载并内部测试

所以需要发布到托管平台，便于相关人员下载。

目前常见的app托管平台很多，现总结如下。

crifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新：2018-06-18 17:06:25

**fir.im**

## 注册fir.im账号

去

[http://account.fir.im/users/sign\\_up](http://account.fir.im/users/sign_up)

注册，去填写注册信息：

The screenshot shows the sign-up page for fir.im. At the top, there's a header bar with the URL 'account.fir.im/users/sign\_up' and several navigation links: 'this week's t...', 'Dev Tools -...', '应用动态 | fir.i...', and 'iOS - fir.im'. Below the header, the word '注册' (Register) is centered above a form. The form includes fields for 'Sign up requirement:' (Email: crifan@webonn.com, Password: masked, CAPTCHA: 1, Verification Code: 719914), a '获取验证码' (Get Verification Code) button, and a large orange '注册' (Register) button. At the bottom of the form is a link '我是老用户，要登录' (I'm an old user, log in).

然后提示注册成功：

你可以使用 fir.im 帐号登录以下多个网站

**fir.im**

BugHD

inCode

## 注册成功



一封带有确认链接的邮件已经发送至您的邮箱，请检查邮箱（包括垃圾邮箱），并点击该链接激活您的账号。

之后会收到确认邮件：

### 确认信息 ☆ ☰

发件人: **notice@mail.fir.im** <notice@mail.fir.im>  自动归档

时 间: 2016年7月13日(星期三) 下午5:31

收件人: **crifan** <crifan@webonn.com>



Hi  
crifan@webonn.com

感谢你选择了 fir.im, 点击下面链接验证邮箱：

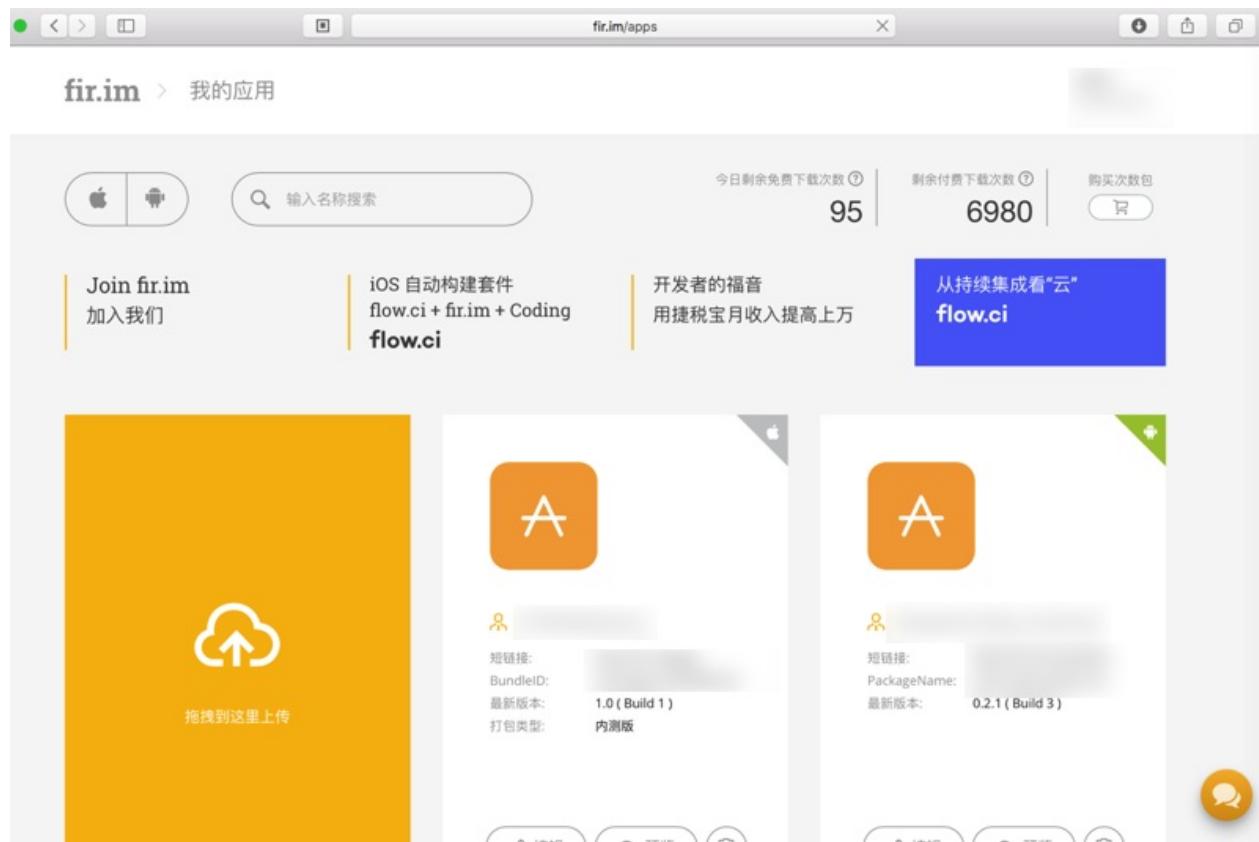
[http://account.fir.im/users/confirmation?  
confirmation\\_token=7WDxm3xuLUySyBqyo5af](http://account.fir.im/users/confirmation?confirmation_token=7WDxm3xuLUySyBqyo5af)

然后即可用新注册的账号去的登录了：



## 上传并发布app

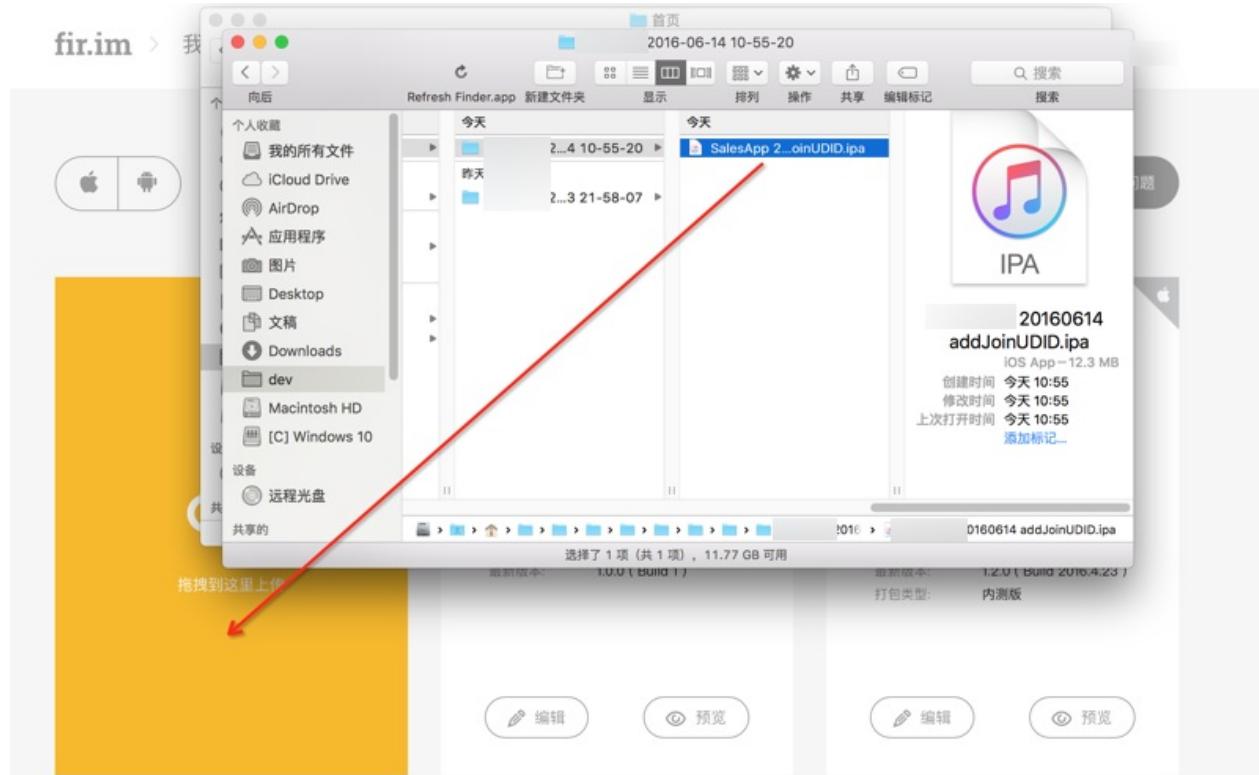
之后，即可进入fir.im管理页面：



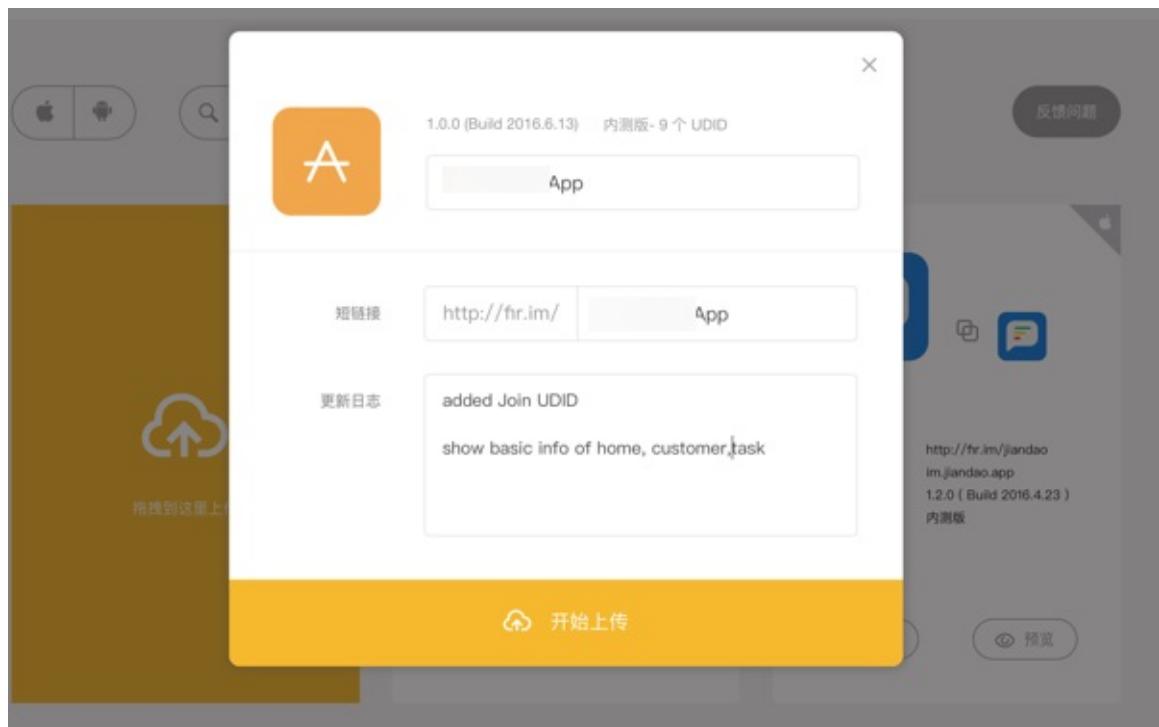
然后即可上传相应的app:

## iOS的app

比如此处上传的是iOS的app:



然后fir.im会自动检测出版本号，日期，版本类型等信息，接着输入对应的要发布的app的地址等信息：



点击开始上传后，显示进度：



上传完毕后，即可自动跳转到后台管理页面，看到已上传的app的各种信息：

http://fir.im app

http://fir.im app iOS BundleID com. app iOS 9.0 或者高版本

上传新版本 预览

基本信息 权限控制 应用合并 高级统计 设备列表 集成

版本更新 商店地址：未填写 修改

1.0.0 (Build 2016.6.13)  
2016-06-14 11:05:13 · 内测版  
added Join UDID  
show basic info of home, customer,task  
编辑 预览 标记上线

对于iOS来说，点击设备列表还可以看到已经注册的设备：

- 设备列表仅识别当前版本可安装的设备UDID，设备UDID要在苹果开发者网站添加：[苹果开发者后台](#)
- UDID添加方法可参照文档：[使用Ad-Hoc方式打包](#)

#### UDID列表及用户信息

用户	邮箱	设备型号	系统版本	UDID
			ab8e	i274
			090c	i70c
		iPhone 5s (China)	9.2.1	iada4
		iPhone 6	13E238	i579
			9a0a	i95
			5c59	i82
			c0et	i90
			222e	i774
			f261	i8f

相应的，去打开对应的fir.im的地址：

XxxYyyApp – fir.im

<http://fir.im/XxxYyyApp>

后，可以看到对应的app下载页面的信息：



扫描二维码下载  
或用手机浏览器输入这个网址: <http://fir.im/>

App

内测版 - 1.0.0 (Build 2016.6.13) - 11.71 MB  
更新于: 2016-06-14 11:05

通过iPhone等去打开后，可以点击去下载并安装。

## Android的app

Android的app的上传和信息管理，也是类似的，就不重复介绍了。

列出部分截图供参考：

信息管理：

**fir.im > 我的应用 > 简道**

http://fir.im/jiandaoandroid | Android | PackageName | im.jiandao.app

上传新版本 | 预览

基本信息 | 权限控制 | 应用合并 | 高级统计 | 集成

版本更新  
商店地址：未填写 | 修改

1.0.0 (Build 1)  
2016-05-04 17:38:09

1.实时高效信息沟通和协作  
2.支持多种聊天类型：讨论组，话题，同事，好友  
3.支持多种聊天消息：文本，图片，文件，通知  
4.支持新版本更新

(编辑) (4.09 MB) (预览) (标记上线)

基本信息：

The screenshot shows the 'Basic Information' tab of the application management interface. At the top, there's a navigation bar: 'fir.im > 我的应用 > 简道'. To the right is a blue circular icon with a white power symbol. Below the navigation, there's a toolbar with tabs: '基本信息' (selected), '权限控制', '应用合并', '高级统计', and '集成'. On the far right are two buttons: '上传新版本' (Upload New Version) and '预览' (Preview). The main area displays the following details:

- 应用 ID:** 57
- 应用名称:** 简道
- 短链接:** http://fir.im/jiandaoandroid
- 应用描述:** (empty field)

## 其他管理功能

此处再去看看其他一些额外的功能：

权限控制：

The screenshot shows the 'Permission Control' tab of the application management interface. At the top, there's a navigation bar: 'fir.im > 我的应用 > 简道'. To the right is a blue circular icon with a white power symbol. Below the navigation, there's a toolbar with tabs: '基本信息' (selected), '权限控制' (selected), '应用合并', '高级统计', and '集成'. On the far right are two buttons: '上传新版本' (Upload New Version) and '预览' (Preview). The main area displays the following settings:

- 下载页对所有人可见:** A toggle switch is set to 'On'.
- 开启访问密码:** A toggle switch is set to 'Off'.
- 成员列表:** Shows a single member named 'larry' with the email 'larry@daryun.com'. There's also a '邀请成员' (Invite Member) button.

应用合并：

fir.im > 我的应用 > 简道



http://fir.im/jiandaoandroid Android PackageName im.jiandao.app

基本信息 权限控制 应用合并 高级统计 集成

上传新版本 预览

选择已有的应用进行合并 或输入需要合并的应用的短链接

输入短链接并查询

简道

主要是用来实现，对于同一个app的不同平台，比如iOS和Android的话，可以合并在一起，便于管理：

fir.im > 我的应用 > 简道



http://fir.im/jiandaoandroid Android PackageName im.jiandao.app

基本信息 权限控制 应用合并 高级统计 集成

上传新版本 预览

已经与 简道 合并

iOS Android

解除合并

高级统计：

fir.im > 我的应用 > 简道



The screenshot shows the 'Integration' tab selected in the navigation bar. On the left, there's a sidebar with '版本统计' (Version Statistics) highlighted. In the center, it says '暂无统计数据, 请先生成统计链接' (No statistics data, please generate a statistics link first). At the bottom, there are links for '关于我们' (About us), '博客' (Blog), '合作伙伴' (Partners), 'Android 客户端' (Android Client), '用户协议' (User Agreement), '隐私政策' (Privacy Policy), and '联系我们' (Contact us).

集成：

fir.im > 我的应用 > 简道



The screenshot shows the 'Integration' tab selected. It features sections for '消息推送' (Message Push) and '自定义 WebHooks' (Custom WebHooks). Under '消息推送', there's a note about using WebHook to POST to a specified address. Below that, there's information about BearyChat, a communication tool. At the bottom, there's a note about handling download errors on iOS devices.

## 下载fir.im上的app

用手机端去打开对应页面，然后按照提示去点击下载即可。

### iOS的错误：无法下载应用程序 此时无法安装

iPhone等iOS设备中，有时候去下载fir.im上的app时，会提示出错：无法下载应用程序 此时无法安装



其根本原因是：此APP开发时配置是最低只支持 iOS 9.0，而此处iPhone系统版本是 iOS 8.3，由于版本太低而无法安装

吐槽：还是iPhone系统的提示不够智能，对于低版本的iOS系统，应该提示 已下载但无法安装，原因是当前系统版本太低，这样用户就清楚错误原因了。

对此问题，[fir.im 常见问题](#)中也有总结：

第三种：打包时选择支持的 iOS 系统版本过高，低于设置的系统版本的 iOS 设备无法安装 解决方法：降低打包时支持的 iOS 系统的最低版本。

crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：2023-12-13 14:44:50

# 蒲公英pgyer

crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新: 2018-02-02 15:19:45

# APP上架总结

crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新: 2018-02-02 15:22:44

# iOS上架

- app上架的地方：
  - 公开版本：苹果官网的唯一的App Store
  - 企业版：无需上架，找个app应用发布的地方供用户下载即可，比如
    - fir.im
    - OTA版：自己提供对应的服务器用于存放ipa和对应的plist
      - 注意必须要支持https才可以
  - AdHoc版本：临时发布供测试
    - 在用Xcode打包之前就要设置好对应的profile，添加iPhone等设备等GUID，发布后，加了GUID的iPhone等设备才能安装和使用
- 需要提供App的测试账号和密码
  - 苹果官方审核人员会登录你的app
  - 测试你的基本功能
    - 看看app内容是否有违规内容
    - 且要确保你的app的稳定性
      - 不会随便就崩溃了
      - 否则影响用户体验，就会被拒

crifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新：2018-02-02 21:42:47

# iOS小范围测试版: AdHoc

crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新: 2018-02-02 15:23:25

## iOS上架公开市场：AppStore

crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：2018-02-02 15:23:35

# iOS内部发布：企业版

crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：2018-02-02 15:24:21

## OTA版

### 坑：打包OTA包时的设置的图片地址也要正确否则无法下载

iOS打包OTA的安装包时，即使只是图片有问题（而ipa没问题），也会导致无法正常下载和安装

详见：[\[已解决\] 企业版的iOS的app去In House打包和OTA发布后有时候无法下载和安装：无法下载应用 此时无法下载 完成 重试](#)

### 坑：打包OTA包时plist中的ipa文件或图片的地址中如果包含特殊字符也会导致无法正常下载

对于ipa或图片的文件地址中如果包含特殊不可见字符的话，也是会导致无法下载的。

详见：

[\[已解决\] 企业版iOS的ipa通过OTA发布后还是无法下载和安装](#)

crifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新：2018-02-02 22:09:31

## Android上架

- app上架的地方：自己选择，有很多
  - Android官方的：Google Play
  - 国内的：
    - 应用宝
    - 360
    - 百度
    - 等等
- 无需提供测试账号

crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：2018-02-02 21:41:19

# APP安装和使用总结

此处整理和介绍，和APP开发有关系的，关于APP的安装和使用方面的知识。

crifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新：2018-02-08 10:58:32

## iOS的APP的安装和使用

此处介绍iOS的APP的安装和使用的基本知识，以便于让iOS的APP开发和iOS的APP的使用人员，对于APP的不同版本，安装方法，有个基本的了解。

## iOS的APP的发布模式

如之前已经说的，其实有几种模式：

- AdHoc
- 企业版
  - OTA
- AppStore

下面分别介绍具体安装方式

### AdHoc

iOS打包AdHoc的目的是用于小范围的内部测试。

所以苹果要去打包AdHoc时，需要实现把相关的每个iOS设备的UDID，添加进去，打包后，只有加了UDID的iOS设备，才能安装AdHoc版本。

打包出来的也是ipa，上传到别的地方，即可下载。

### fir.im上的AdHoc版本

比如上传到fir.im上，比如：

xxx Production生产版 iOS

<https://fir.im/XxxiOSProd>

然后别人去用iPhone中的Safari去打开对应地址



然后点击下载按钮，会弹出提示：



### 弹出提示会显示出域名

对于此处弹出的提示中的 `download.fir.im`，其实就是对应的之前提到的 `plist` 文件所在的 `https` 的服务器的域名/IP。

点击确定后，页面上会提示：正在安装，请按Home键在桌面查看：



内测版 - 0.2.3 (Build 1) - 12.17 MB

更新于: 2018-01-18 15:06

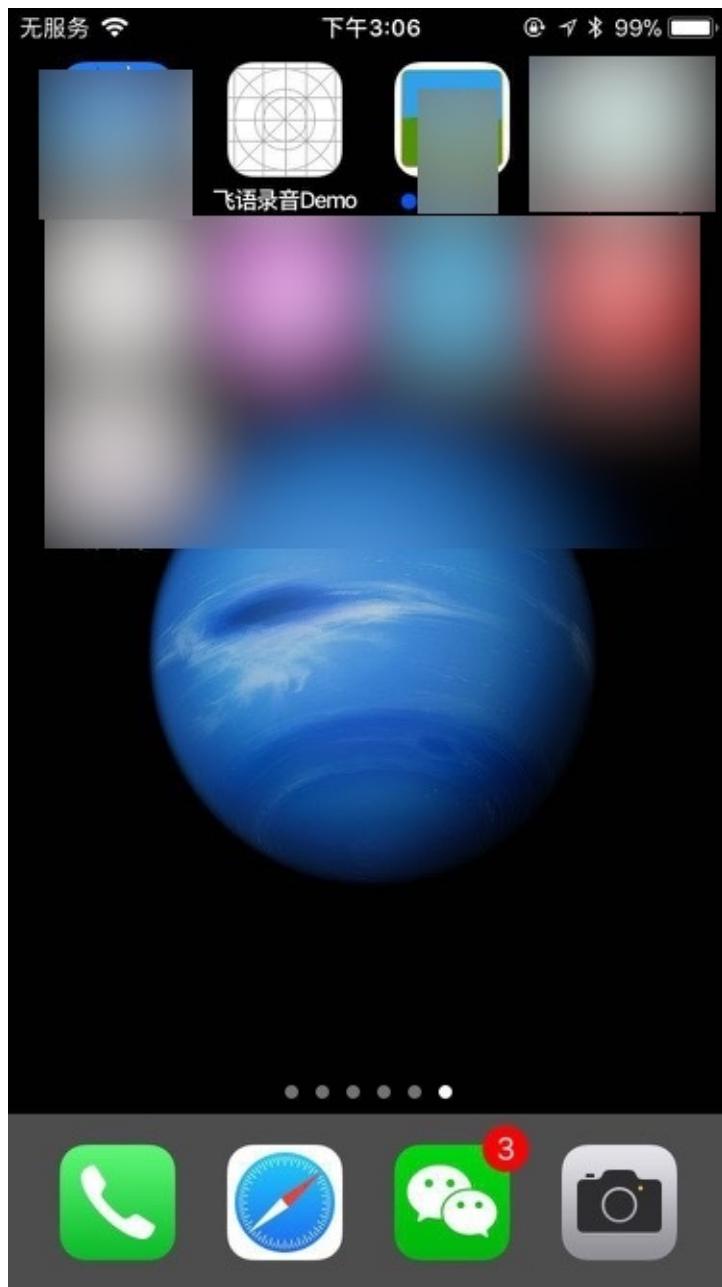
正在安装, 请按 Home 键在桌面查看



然后iPhone中按Home键, 回到桌面, 找到正在安装的APP:



稍等片刻，即可安全完毕：



## 下载速度取决于实际情况

对于此处的fir.im中的，是针对国内网络优化过的速度，所以下载很快的。

但是如果是其他地方的下载这种ipa的话，有时候会比较慢，往往是国外服务器的话，速度一般不是很理想。

## 企业版

### fir.im上的企业版

### 自己的公司官网的企业版的OTA版

其实也可以让自己的公司：

- 弄一个Web服务器
  - 把iOS打包出来的plist和ipa和logo图片，都放到对应的目录下
    - 就有了对应的plist文件的地址了

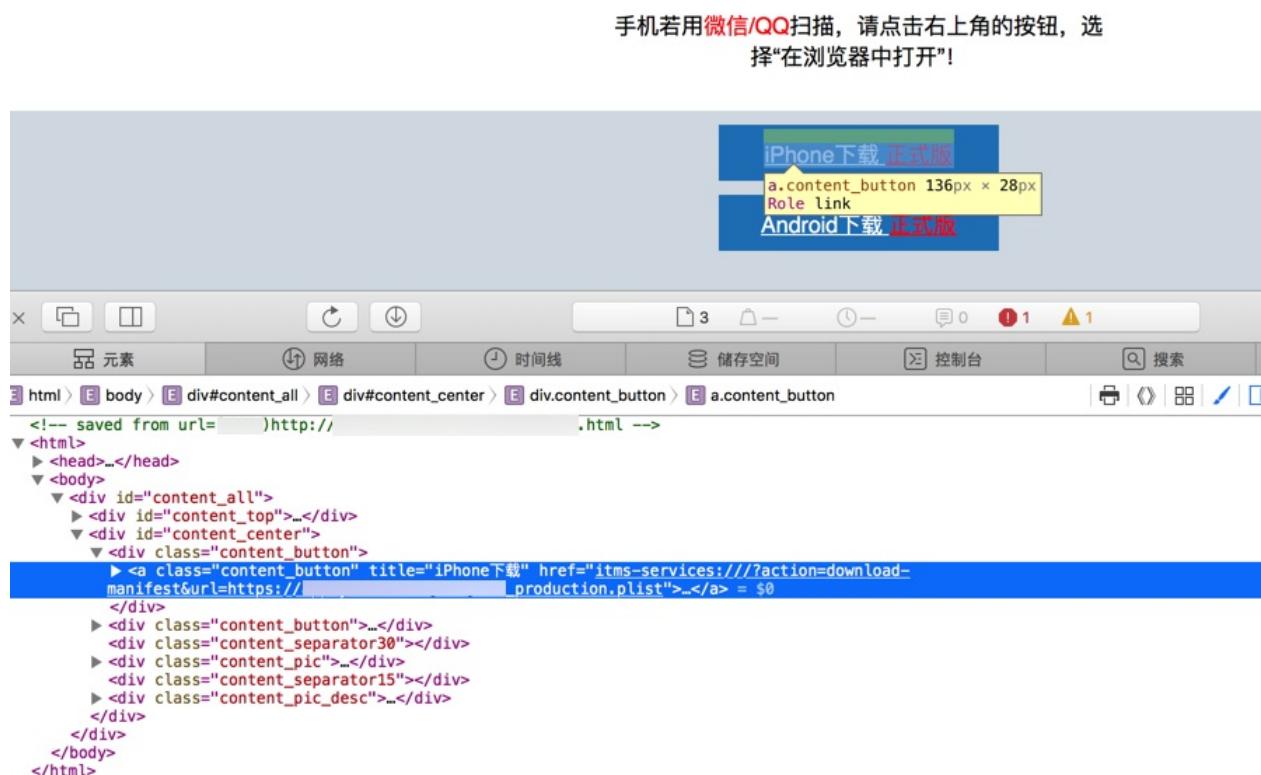
- 然后

- 再去弄一个HTML页面
  - 其中包含对应的：以 `itms-services` 开头的、`plist` 结尾的地址
  - 再弄一个按钮 `button` 或 `(a的href)` 的链接，点击后去打开上面这个地址
- 或者是iOS的app代码内部直接open以这个 `itms-services` 开头的、`plist` 结尾的地址
  - 也会直接调用iOS的Safari去打开此地址
    - 从而实现自动下载ipa并安装的效果
    - 即：实现自动升级iOS的app的效果 = 不用打开浏览器而自动下载ipa并安装的效果
  - 相关参考代码：
    - `[[UIApplication sharedApplication]openURL:[NSURL URLWithString:[NSString stringWithFormat:@"itms-services://?action=download-manifest&url=https://xxx/yyy.plist"]]];`

## https的Web服务器的ipa+plist+html页面实现发布企业版的OTA版的示例

示例1: html中点击按钮跳转href的itms-services地址

如图：



其实内部都是一个 `itms-services` 开头的、`plist` 结尾的地址，比如：

```
itms-services:///?action download-manifest url https://xxx.yyy/zzz/aaa_production.plist
```

由此，点击对应的按钮后，即可直接或间接的调用iOS的Safari浏览器去解析并验证，通过后，才会直接去下载对应的ipa，安装对应的iOS的APP。

示例2: html中点击按钮执行js去打开itms-services的地址

如图：



内部的web服务器中存放这对应的logo图片、Android的apk文件、iOS的ipa文件等等：

```

[xxx@yyy html]$ pwd
/usr/share/nginx/html
[xxx@yyy html]$ ll
total 3864
-rwxrwxrwx. 1 root      root      537 May 31 2016 50x.html
drwxrwxrwx. 2 useradmin useradmin 4096 Jan 17 11:45 android
-rw-r--r--. 1 useradmin useradmin   61 Aug 15 2016 index1.html
drwxrwxrwx. 2 useradmin useradmin 4096 Jan 17 11:43 ios
drwxrwxrwx. 2 useradmin useradmin   85 Aug 10 2016 logo
-rwxrwxrwx. 1 useradmin useradmin 3369375 Aug 10 2016 manual.pdf
drwxrwxr-x. 2 useradmin useradmin   40 Jan 24 2017 test
-rwxrwxrwx. 1 useradmin useradmin 324069 Aug 10 2016 user-guide1.pdf
-rw-r--r--. 1 useradmin useradmin 172166 Aug 10 2016 user-guide.pdf
[xxx@yyy html]$ ll ios/manifest.plist
-rw-r--r--. 1 useradmin useradmin 1111 Jan  9 05:46 ios/manifest.plist
[xxx@yyy html]$ ll ios/zzz.ipa
-rw-r--r--. 1 useradmin useradmin 23998958 Jan  9 05:46 ios/zzz.ipa
[xxx@yyy html]$ ll logo/
total 38
-rwxrwxrwx. 1 useradmin useradmin 10474 Aug 10 2016 zzz_180x180.png
-rwxrwxrwx. 1 useradmin useradmin 19555 Jul 12 2016 zzz_512x512.png
-rwxrwxrwx. 1 useradmin useradmin 2537 Jul 12 2016 zzz_57x57.png
[xxx@yyy html]$ ll android/zzz.apk
-rw-r--r--. 1 useradmin useradmin 4773193 Jan 17 03:25 android/zzz.apk

```

## AppStore

## Android的APP的安装和使用

相对于iOS的app来说，Android的app的下载和安装，限制比较少，都很方便：

只要能下载到对应的apk文件，就可以正常安装了。

### app托管平台：fir.im

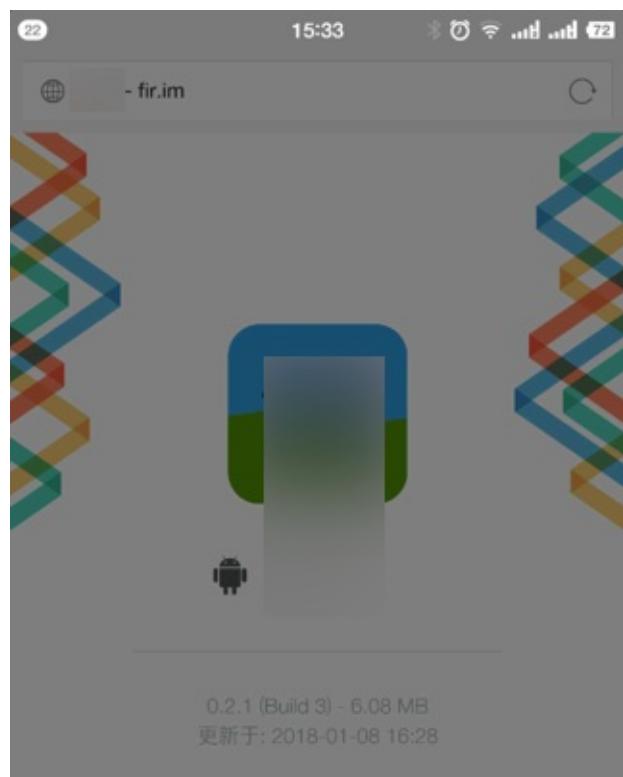
比如把apk文件上传到fir.im上：

[Xxx - fir.im](#)

然后去下载：



点击下载按钮，去下载：



来源: - fir.im

文件转存



apk  
6.1MB V1.0.0

下载

下载后，即可正常安装：





安装完毕后：



点击打开去打开APP。

## 各大应用市场

还有一种更方便的方式是：

在APP的应用市场去搜索并下载某个APP，即可。

比如：

我的锤子手机的应用市场是 应用商店：



进去后，搜索自己要的app，比如：企查查



然后点击去安装即可：



### 应用简介

[问题反馈 >](#)

企查查是一款企业信用信息查询工具，帮助用户快速了解企业状况。

[更多](#) ▾



然后点击打开：



即可打开APP：



crifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新: 2023-12-13 14:46:47

# APP涉及后期运维总结

crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新: 2018-02-02 15:25:39

## 移动APP数据统计

### 数据统计服务提供商

#### 国外

- flurry
- loclytics
- mixpanel
- Google Analytics for Mobile

#### 国内

- TalkingData
- 友盟统计umeng
- LeanCloud
- 魔方
- Bugly: 能起到部分的收集数据（比如多少用户安装了，每日日活如何等基本的数据的）效果

### 自己搭建数据统计后台

开源自行搭建的选择 == 开源解决方案

- [count.ly](#)

### 总结

单独比较数据统计的服务的效果，目前的看法是：

TalkingData > 友盟 > 腾讯云

crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新: 2018-02-02 21:49:50

# TalkingData

crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新: 2018-02-02 15:26:27

# 友盟UMeng

crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新: 2018-02-02 15:27:06

## Bugly

腾讯的崩溃日志收集的Bugly，也可以起到部分的数据统计功能：

- 运营概览
- 用户分析
- 版本分布
- 留存分析
- 使用频率/时长
- 渠道分析
- 渠道分布

crifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新：2018-02-02 21:45:20

# 崩溃日志收集

崩溃日志收集 = Crash Log Collection = 崩溃检测

crifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook 最后更新: 2018-02-02 15:30:05

# Bugly

crifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新: 2018-02-02 15:28:25

# 应用内问题反馈

crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新: 2018-02-02 15:31:26

# 蒲公英pgyer

crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新: 2018-02-02 15:31:55

## 子教程

- iOS
  - [iOS开发心得](#)
- Android
  - [Android开发总结](#)

crifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新: 2022-11-03 17:35:58

## 附录

下面列出相关参考资料。

crifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新: 2018-02-02 14:51:49

## 参考资料

- [调研] iOS和Android的移动端应用数据统计
- Mobile Web Development - Web developer guides | MDN
- Web App 和 Native App, 哪个是趋势? - 赵瀚卿的回答 - 知乎
- [已解决] 企业版的iOS的app去In House打包和OTA发布后有时候无法下载和安装: 无法下载应用 此时无法下载 完成重试
- [已解决] 企业版iOS的ipa通过OTA发布后还是无法下载和安装
- [基本解决] iOS页面切图的尺寸距离, 图片, 字体的大小和代码中所使用的iOS设备真实的尺寸的关系
- 【整理】利用极光推送JPush实现消息推送
- [记录] 购买JPUSH极光推送收费版 – 在路上
- [已解决] iOS的JPUSH生产环境无法推送 – 在路上
- [记录] 到极光推送中设置iOS的Development推送 – 在路上
- [记录] 极光推送JPUSH更换生产证书和Bundle ID – 在路上
- [记录] 手动测试JPUSH极光去企业账号生产环境的app中的消息推送
- [记录] 极光推送: AppKey的应用iOS开发证书到期时间
- [记录] 创建远程推送APNS的证书certificate: Development和Distribution – 在路上
- [已解决] 钥匙串中创建证书出错: 您输入的用户名和密码短语不正确
- [记录] 极光推送: AppKey的应用iOS开发证书到期时间
- [调研] iOS android app被杀掉后 如何实施上报地址位置信息 和 离线推送 – 在路上
- [已解决] 使用飞语云平台实现iOS的电话录音
- [已解决] 用飞语云平台实现网络电话拨打手机号且带录音
- 【整理】iOS 电话录音 sdk 方案
- [已解决] 用飞语FYRtcEngineKit去实现基本的iOS间的语音通话
- [记录] 注册app发布平台fir.im账号 – 在路上
- [记录] 给iOS app添加新的UDID后再重新打包ipa并上传fir.im – 在路上
- [记录] 把android的apk上传到fir.im中 – 在路上
- 

crifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新: 2018-06-18 17:10:05