

自动生成 APP 代码说明

机智云

编制人	翁丹丽	审核人		批准人	
产品名称		产品型号		文档编号	
会签日期			版本	2.5.1	

修订记录

修改时间	修改内容	版本	修改人	备注
2018.06.27	内容更新	2.5.1	翁丹丽	
2018.06.29	内容更新	2.5.1	沈显乐	

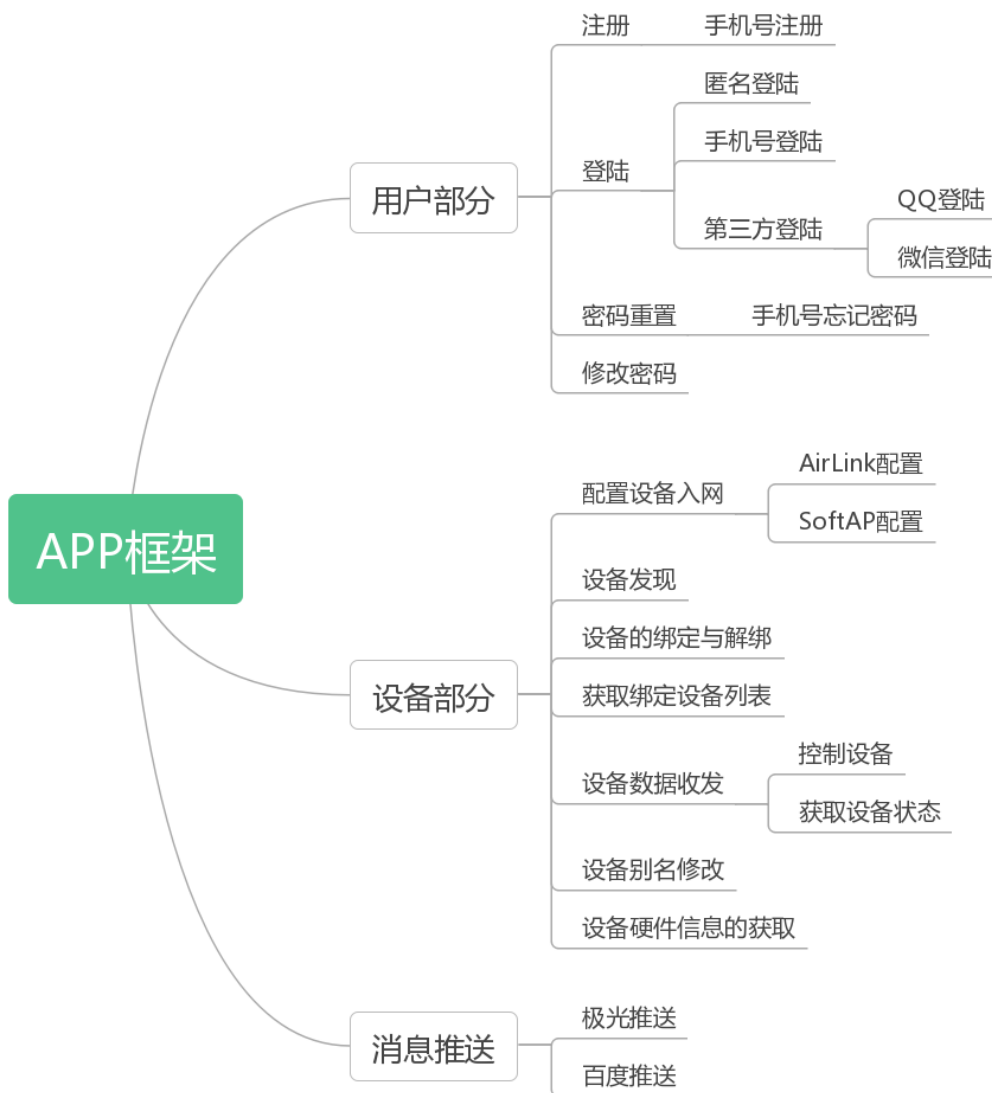
目录

1. 概述	4
2. 部署源码并控制虚拟设备	5
3. 源码说明	10
3.1. ios 目录结构说明	10
3.2. Android 目录结构说明	12
3.2.1. Android Studio 目录结构说明	12
3.2.2. Eclipse 目录结构说明	15
3.3. APP 配置文件说明	17
3.4. 控制界面入口	20
3.4.1. ios 控制界面入口	20
3.4.2. Android 控制界面入口	21
4. 第三方登陆	21
4.1. QQ 登陆接入	21
4.1.1. ios 登录配置	22
4.1.2. Android 登录配置	22
4.2. 微信登陆接入	24
4.2.1. ios 登录配置	24
4.2.2. Android 登录配置	25
5. 消息推送	26
5.1. 极光推送	26
5.2. 百度推送	26

1.概述

自动生成 APP 代码集成了用户注册、登陆、密码重置、设备入网配置、设备发现、设备绑定与解绑、设备控制等物联网控制 APP 的常用功能，并集成了第三方登陆和消息推送等可配置的功能，开发者可以通过在配置文件中配置来选择性地使用这些功能。

下面的思维导图罗列了 APP 所具备的所有功能模块：



2.部署源码并控制虚拟设备

如不需要第三方登陆与消息推送功能，则工程无需修改直接可以部署到手机上面运行。

下面将逐步说明如何使用源码控制云端虚拟设备。

第一步：部署源码到手机上。



点击注册新用户，按照流程注册好账户后登录 APP，进入如下界面：



第二步：启动虚拟设备。

如图所示，开发者下载的源码对应云端创建产品《测试自动生成 APP》，点击“在线调试设备”按钮。



进入虚拟设备页面，点击“启动虚拟设备”。



进入虚拟设备控制界面，点击显示二维码。



第三步：扫码控制虚拟设备。

点击设备列表左上角的扫码按钮，在跳转的扫码界面中扫描虚拟设备的二维码。



扫码成功后，APP 将绑定扫码的设备。



点击上图红框条目，进入 APP 控制页面。



点击开关开启按钮。

模拟设备上报数据

布尔可写 (bool_w)

1

枚举可写 (enum_w)

四

数值可写 (data_w)

0 = 1 * 0 + 0

0 999

扩展可写 (extend_w)

000000000000

布尔只读 (bool_r)

0

枚举只读 (enum_r)

推送

通信日志

JSON

接收app发送的数据

2017-03-24 17:19:29

```
{
  "bool_w": "1",
  "enum_w": "四",
  "data_w": 0,
  "extend_w": "00000000000000000000",
  "bool_r": "0",
  "enum_r": "-",
  "data_r": 0,
  "extend_r": "00000000000000000000",
  "alert_": "0",
  "fault_1": "0"}

```

虚拟设备上报数据

2017-03-24 17:19:18

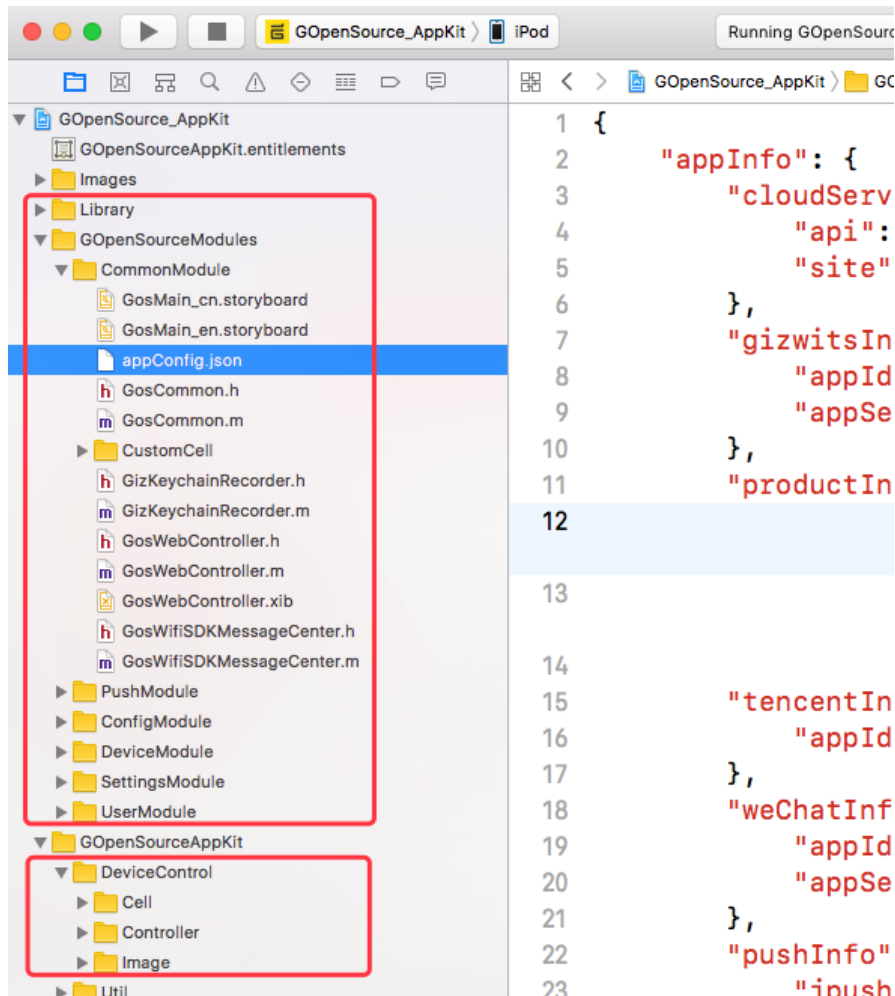
```
{
  "bool_w": "0",
  "enum_w": "四",
  "data_w": 0,
  "extend_w": "00000000000000000000",
  "bool_r": "0",
  "enum_r": "-",
  "data_r": 0,
  "extend_r": "00000000000000000000",
  "alert_": "0",
  "fault_1": "0"}

```

云端虚拟设备成功收到控制指令，表示 APP 控制成功。

3.源码说明

3.1. ios 目录结构说明



- Library: 包括 GizWifiSDK 在内的的第三方库目录
- GOpenSourceModules 组成模块 // 实现控制以外逻辑的模块
 - CommonModule // 公共方法类、资源文件读取类
 - PushModule // 推送模块, 包含 百度和极光的推送 SDK 集成封装
 - ConfigModule // 设备配置模块, 包含 AirLink 及 SoftAP
 - DeviceModule // 设备模块, 包含 设备列表
 - SettingsModule // 设置模块, 包含 消息中心, 设备分享, 修改密码, 注

销登录，以及关于等功能

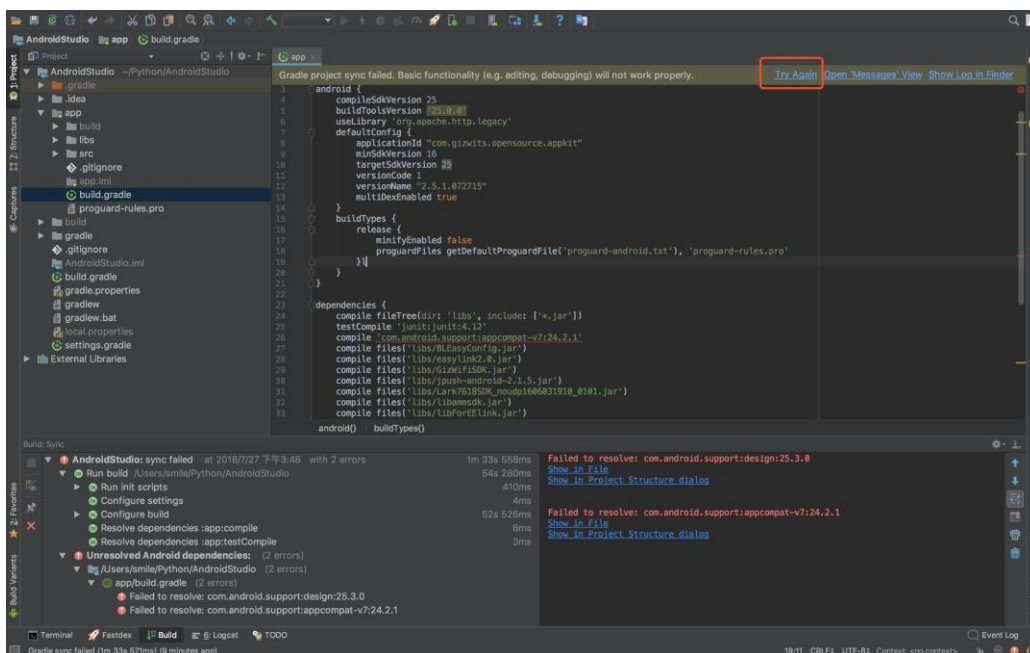
- UserModule // 用户模块，包含 用户登录（包含 QQ、微信等）、用户注册、找回密码



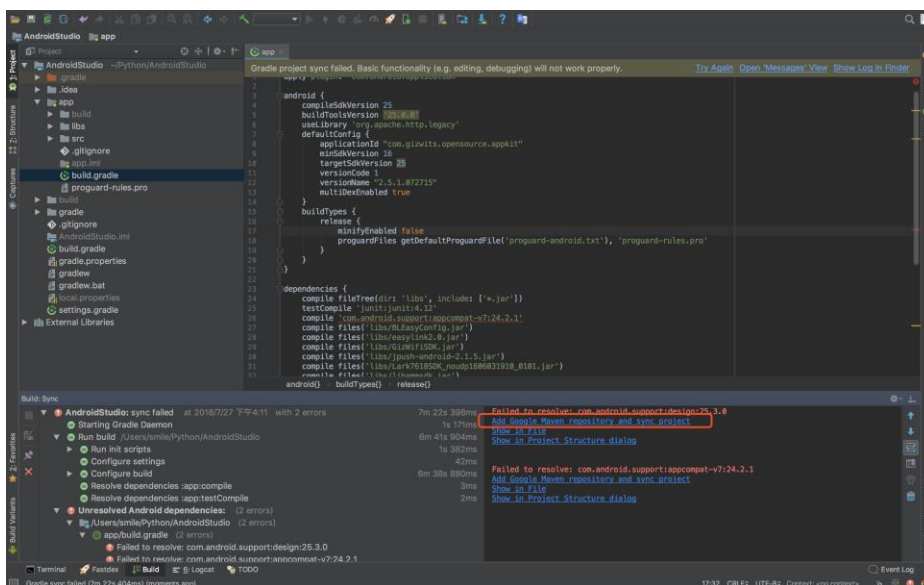
- GOpenSourceAppKit 组成模块 // 实现控制逻辑的模块
 - GosDeviceControl // 设备读写工具类，设备数据的读写通过这里的方法调用来实现
 - GosDeviceViewController // 设备控制界面

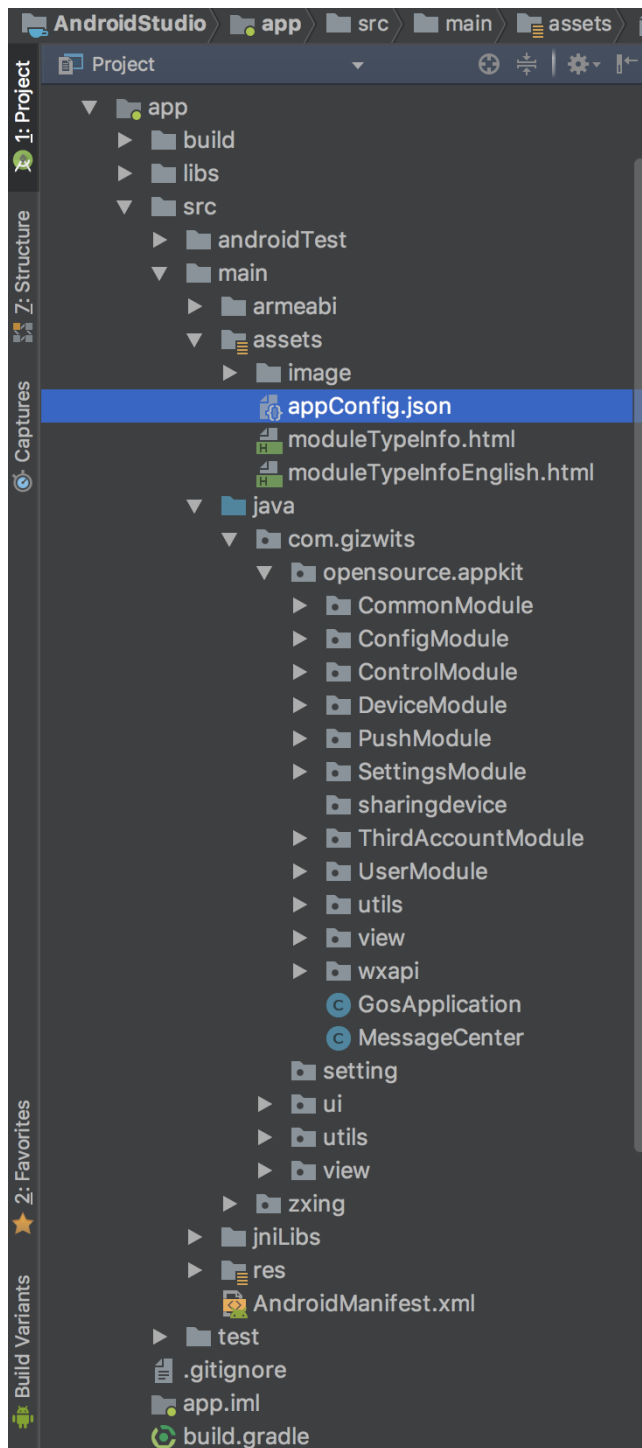
3.2. Android 目录结构说明

3.2.1. Android Studio 目录结构说明



打开 Android Studio，点击 File->Import Project 导入项目，打开 app 的 build.gradle 文件，需要加载两个远程依赖库。点击右上角红框的 Try again,进行编译。然后点击右下角 Add Google Maven repository and sync project，将远程依赖给添加上，这样就可执行了。

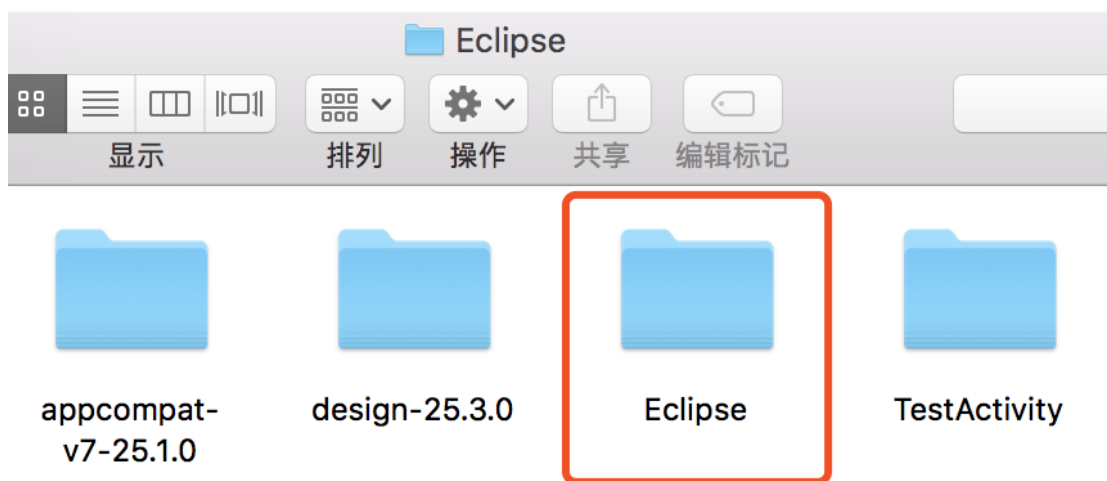




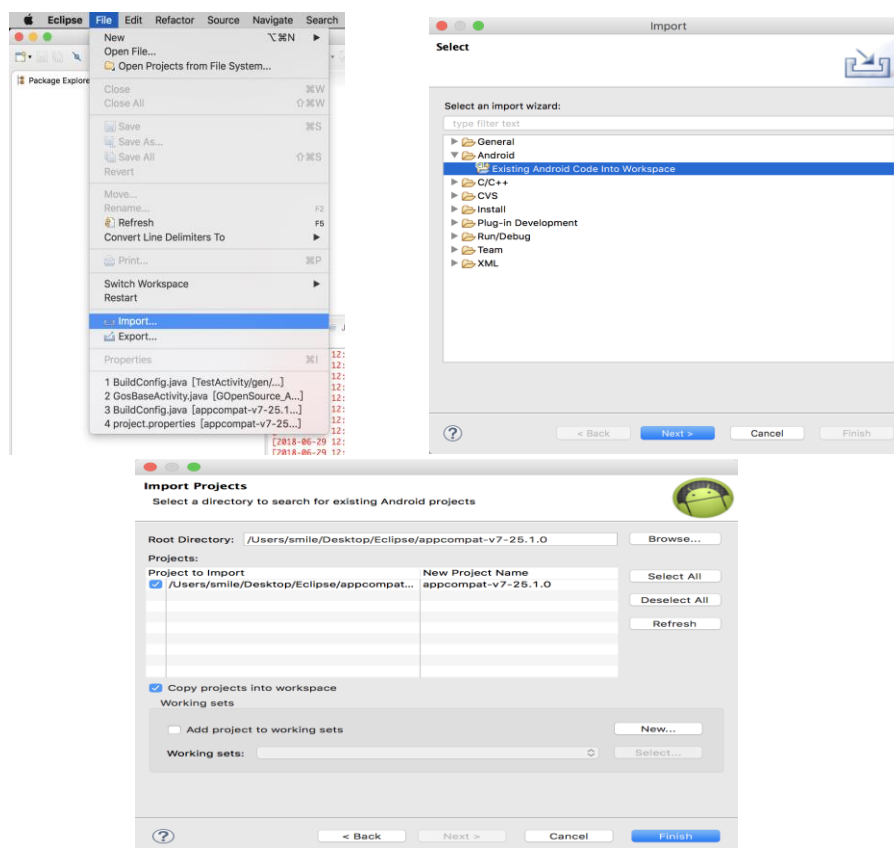
- **libs:** 包括 GizWifiSDK 在内的第三方库目录
- **assets** 资源文件 // 不可编译的原生资源文件
- **java** 文件 // 代码逻辑控制部分
 - **CommonModule** // 公共方法类、资源文件读取类
 - **ConfigModule** // 设备配置模块, 包含 AirLink 及 SoftAP

- ControlModule // 设备控制模块, 包含设备控制
- DeviceModule // 设备模块, 包含 设备列表
- PushModule // 推送模块, 包含 百度和极光的推送 SDK 集成封装
- SettingModule // 设置模块, 包含 关于
- Sharingdevice // 分享模块, 包含分享设备
- ThirdAccountModule // 第三方账号登录模块 包含 QQ 微信
- UserModule // 用户模块, 包含 用户登录、用户注册、找回密码
- Utils //工具模块
- View // 自定义控件模块
- Wxapi //微信第三方登录
- Zxing // 扫码模块
- res 文件 // 可编译的资源文件
- jniLibs 文件 // 加载 .so 的文件夹
- AndroidManifest 文件 // 清单文件 清单文件为 Android 系统提供有关您的应用的基本信息

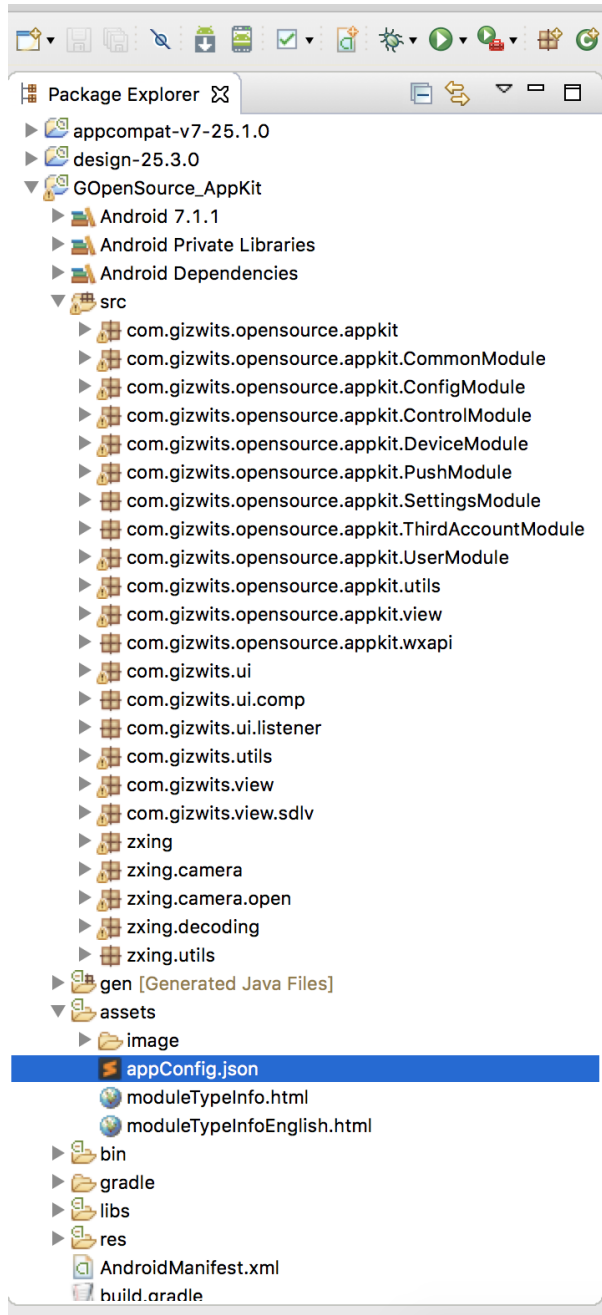
3.2.2. Eclipse 目录结构说明



Eclipse 源代码文件下有一个 Eclipse 工程，是自动生成的 Eclipse 源码，而 appcompat-v7-25.1.0 、 design-25.3.0、TestActivity 是需要用的的第三方库。



按照上图 点击 File 选择 Import,选择 Existing Android Code into Workspace 点击 next, 以次选择 Eclipse 下的项目，拷贝到本地的工程目录，点击 finish 进行导入。



- libs: 包括 GizWifiSDK 在内的第三方库目录 加载 .so 的文件
- assets 资源文件 // 不可编译的原生资源文件
- src 文件 // 代码逻辑控制部分
 - CommonModule // 公共方法类、资源文件读取类
 - ConfigModule // 设备配置模块, 包含 AirLink 及 SoftAP
 - ControlModule // 设备控制模块, 包含设备控制

- DeviceModule // 设备模块, 包含 设备列表
- PushModule // 推送模块, 包含 百度和极光的推送 SDK 集成封装
- SettingModule // 设置模块, 包含 关于
- Sharingdevice // 分享模块, 包含分享设备
- ThirdAccountModule // 第三方账号登录模块 包含 QQ 微信
- UserModule // 用户模块, 包含 用户登录、用户注册、找回密码
- Utils //工具模块
- View // 自定义控件模块
- Wxapi //微信第三方登录
- Zxing // 扫码模块
- res 文件 // 可编译的资源文件
- AndroidManifest 文件 // 清单文件 清单文件为 Android 系统提供有关您的应用

的基本信息

3.3. APP 配置文件说明

下面是对 appConfig.json 文件的参数说明:

注意: Android 每次修改完 appConfig.json 的内容, 都要卸载当前的 APK, 然后重新编译安装, 以保证当前 appConfig.json 的内容确实进行了修改。

appConfig 分为四大模块, 下面将逐一描述各个模块的功能

模块一: appInfo (结构如下图), 主要用来设置 APP 连接的域名, 机智云上申请的应用信息, 产品信息, QQ、微信以及第三方推送申请到的 appId 和 appSecret 等信息。

- cloudService: 设置云端服务器域名, 若是全球化的产品, 这里不需要设置, APP 会根据时区自动选择连接对应的服务器
 - api: 填写 api 域名信息, 如: api.gizwits.com 或 api.gizwits.com:80
 - site: 填写 site 域名信息, 如: site.gizwits.com 或 api.gizwits.com:80

- gizwitsInfo: 设置在机智云开发者中心申请到的应用信息
 - appId: 填写机智云应用的 appId
 - appSecret: 填写机智云应用的 appSecret
- productInfo: 设置产品信息
 - productKey: 填写在机智云开发者中心创建的产品的 PK
 - productSecret: 填写在机智云开发者中心创建的产品的 productSecret
- tencentInfo: 设置在腾讯开发者中心申请到的应用信息
 - appId: 填写腾讯应用 appId
- weChatInfo: 设置在微信开发者中心申请到的应用信息
 - appId: 填写微信应用 appId
 - appSecret: 填写微信应用 appSecret
- pushInfo: 设置推送的应用信息
 - jpushAppKey: 填写在极光开发者中心申请到的推送应用 appKey
 - bpushAppKey: 填写在百度开发者中心申请到的推送应用 appKey

注意：在这里设置完 QQ、微信、极光和百度的应用信息之后，并不表示 APP 就已经打开了这些功能了，还需要在 `appConfig.json` 下的 `<functionConfig>` 模块进行设置。

```

2    "appInfo": {
3      "cloudService": {
4        "api": "api.gizwits.com",
5        "site": "site.gizwits.com"
6      },
7      "gizwitsInfo": {
8        "appId": "your_app_id",
9        "appSecret": "your_app_secret"
10     },
11     "productInfo": [{
12       "productKey": "your_product_key",
13       "productSecret": "your_product_secret"
14     }],
15     "tencentInfo": {
16       "appId": "your_tencent_app_id"
17     },
18     "weChatInfo": {
19       "appId": "your_wechat_app_id",
20       "appSecret": "your_wechat_app_id"
21     },
22     "pushInfo": {
23       "jpushAppKey": "your_jpush_app_key",
24       "bpushAppKey": "your_bpush_app_key"
25     }
26   },

```

模块二：templateSelect，这块主要是用来配置设备列表界面的

```

27     "templateSelect": {
28         "deviceList": {
29             "unbindDevice": true,
30             "displayMac": true,
31         }
32     },

```

- deviceList: 设置设备列表界面功能
 - unbindDevice: 设置设备列表界面左滑解绑设备按钮, true: 有左滑解绑功能, false: 没有左滑解绑功能
 - displayMac: 设置设备列表界面是否要显示设备的 MAC 地址, true: 显示, false: 不显示

模块三: functionConfig, 主要作用, 设置各个功能模块是要启动还是关闭

```

30     "functionConfig": {
31         "bindDevice_qrcode": true,
32         "deviceOnboarding": {
33             "config_softap": true,
34             "config_airlink": true,
35             "wifiModuleType": []
36         },
37         "login_anonymous": true,
38         "login_qq": false,
39         "login_weChat": false,
40         "register_phoneUser": true,
41         "resetPassword_phoneUser": true,
42         "personalCenter_changePassword": true,
43         "push_baidu": false,
44         "push_jiguang": false
45     },

```

- bindDevice_qrcode: 是否开启扫码绑定设备的功能, true: 设备列表左上方的扫码按钮显示, 扫码功能可用; false: 不显示按钮, 屏蔽扫码功能
- deviceOnboarding: 配置支持的配网方式以及调用哪个配网接口进行配网
 - config_softap: true: 支持 softAP 配置; false: 不支持
 - config_airlink: true: 支持 Airlink 配置; false: 不支持
 - wifiModuleType: 在这里填写默认配置的模组类型
- login_anonymous: true: 支持匿名登录; false: 不支持
- login_qq: true: 支持 qq 登录; false: 不支持
- login_weChat: true: 支持 weChat 登录; false: 不支持
- register_phoneUser: true: 支持手机号注册; false: 不支持
- resetPassword_phoneUser: true: 支持手机号重置密码 (忘记密码); false: 不支持

- personalCenter_changePassword: true: 支持修改密码功能; false: 不支持
- push_baidu: true: 支持百度推送功能; false: 不支持
- push_jiguang: true: 支持极光推送功能; false: 不支持

模块四: viewConfig

```

56     "viewConfig": {
57         "viewColor": {
58             "background": "FBDA51",
59             "contrast": "333333"
60         },
61         "statusBarStyle": "default"
62     }

```

- viewColor: 配置界面的颜色样式
 - background: 用来设置导航栏、TabBar、按钮和各类图片的背景色
 - contrast: 设置导航栏、TabBar、按钮上的文字和图片的颜色
- statusBarStyle: 设置导航栏的格式, 包含 default 和 colored 两个值; default: 是文字为黑色的样式; colored: 文字为白色的样式。

3.4. 控制界面入口

在自动生成代码中若要将控制界面切换为用户自定义的控制界面, 而非源码自带的控制界面, 则可删除下图右边红框中的代码, 再做如下两步操作:

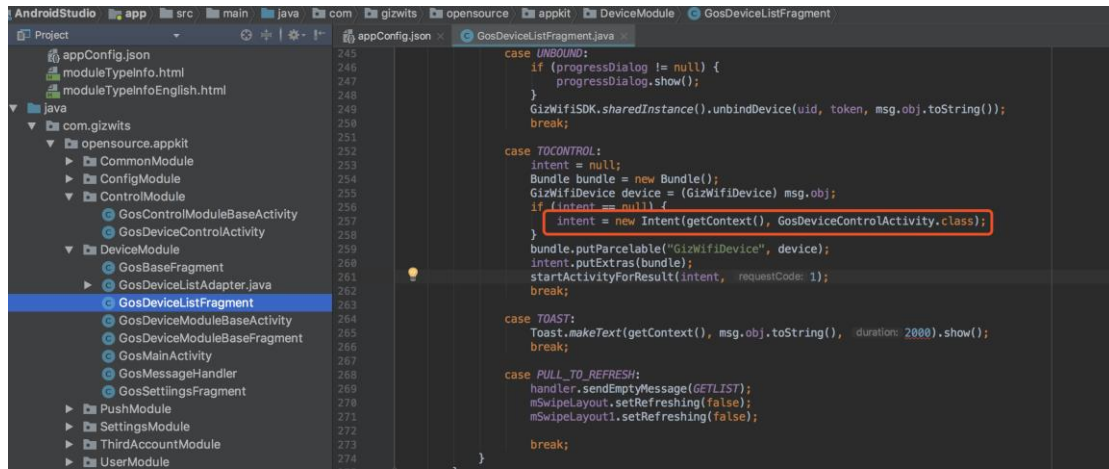
3.4.1. ios 控制界面入口

1) 初始化自定义的控制界面

2) 从 deviceListVC (设备列表界面) 跳转到自定义控制界面



3.4.2. Android 控制界面入口



- 1、初始化自定义的控制界面
- 2、不管是 Android Studio 还是 Eclipse 哪个 IDE, 都是从 GosDeviceListFragment.java (设备列表界面) 跳转到自定义控制界面

4. 第三方登陆

4.1. QQ 登陆接入

APP 要支持 QQ 登陆, 需要先到腾讯开放平台创建一个应用, 获取应用的 APPID, 并设置到 appConfig.json 文件中。

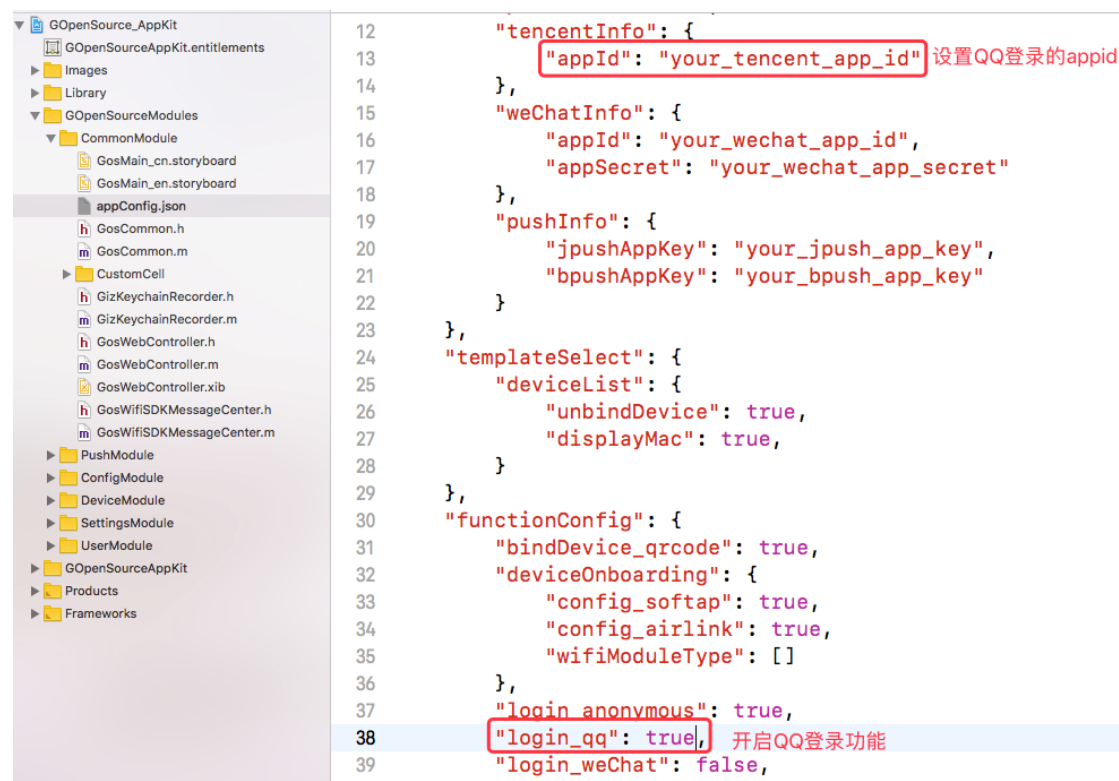
获取腾讯 APPID 并绑定到机智云应用

查看 [QQ 开放平台应用申请教程](#) 获取 APPID 和 APPSecret

开启 QQ 登陆功能, 并将 APPID 填入源码中

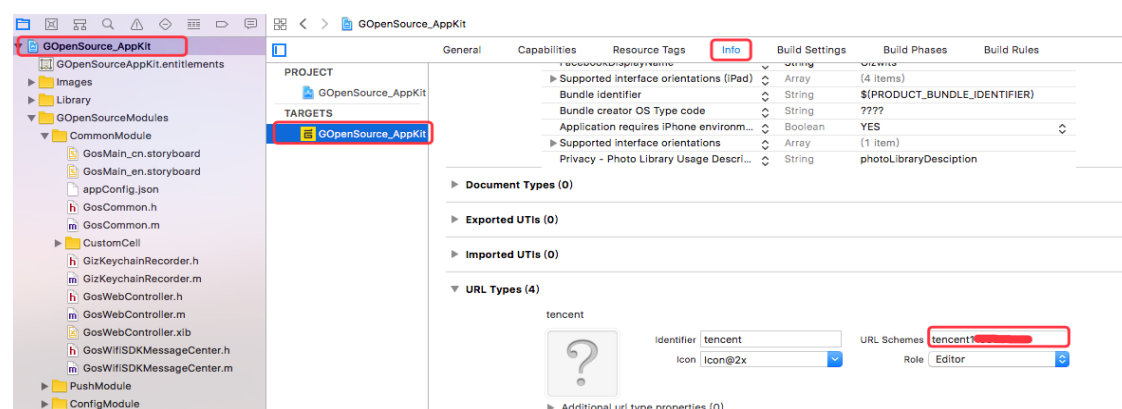
4.1.1. ios 登录配置

在 appConfig.json 文件中配置



配置 URL Schemes

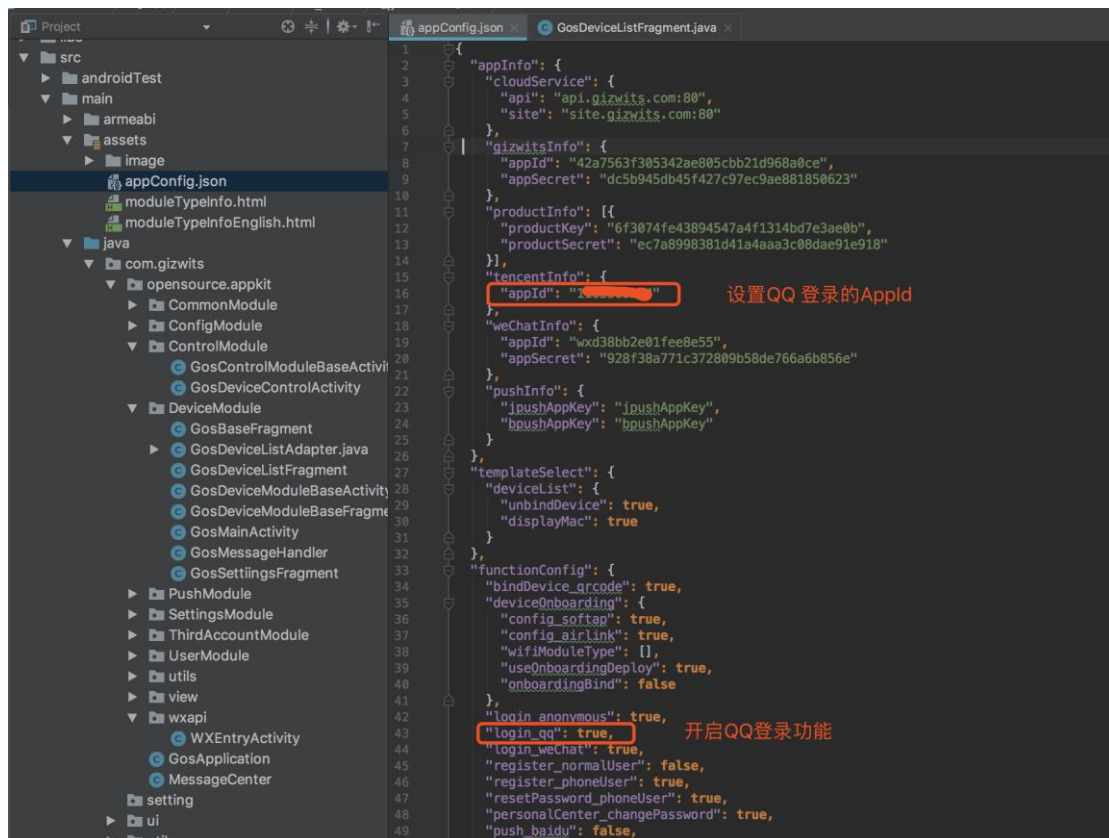
选择 QQ 登陆跳转到 QQ 授权登陆界面, 登陆成功后需要跳转回我们的 APP, 此时需要给 QQ 一个跳转路径, 这个路径就是通过 URL Schemes 来设置的, 如下图:



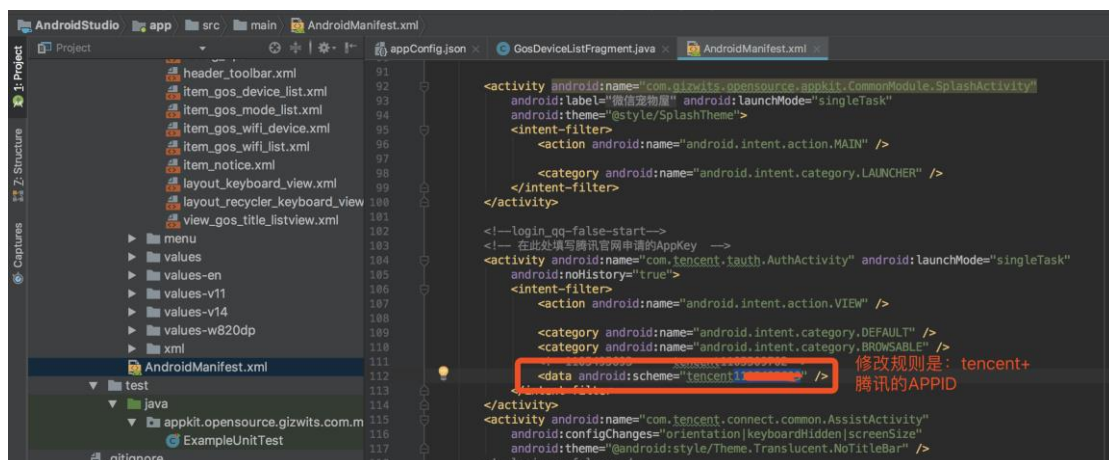
URL Schemes 的修改规则是: tencent+腾讯的 APPID。

4.1.2. Android 登录配置

在 appConfig.json 文件中配置。



配置 AndroidManifest.xml 清单文件，



以上的修改是在 Android Studio 中进行的，同样的，Eclipse 也是根据上面的两个操作进行就可以了。**注意：Android 每次修改完 appConfig.json 的内容，都要卸载当前的 APK,然后重新编译安装，以保证当前 appConfig.json 的内容确实进行了修改。**

4.2. 微信登陆接入

APP 要实现微信登陆，必须前往微信开放平台申请一个应用，获取应用的 APPID 和 APPSecret，并设置到 appConfig.json 中。

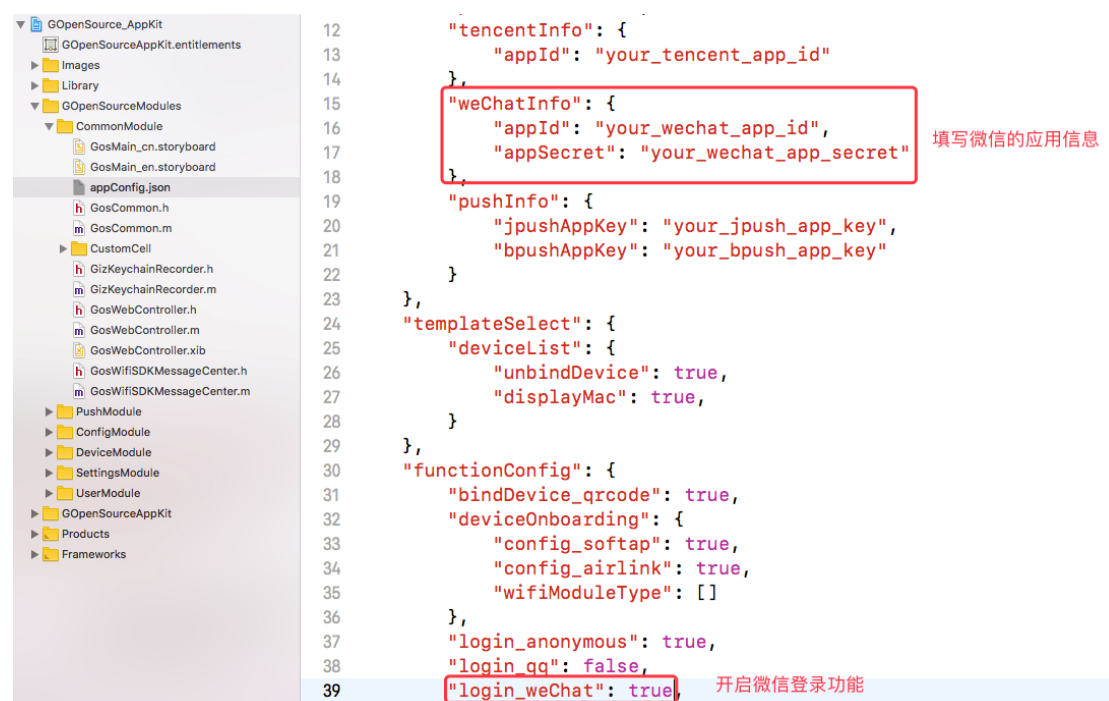
获取微信 APPID 和 APPSecret

查看[微信开放平台应用申请教程](#)获取 APPID 和 APPSecret

开启微信登陆功能，并将 APPID 和 APPSecret 填入源码中

4.2.1. ios 登录配置

在 appConfig.json 文件中配置



The image shows a screenshot of an Xcode project interface. On the left, the project navigator displays the file structure, with 'appConfig.json' selected under the 'CommonModule' folder. On the right, the content view shows the JSON configuration for 'appConfig.json'. The configuration includes fields for 'tencentInfo', 'weChatInfo', 'pushInfo', 'templateSelect', and 'functionConfig'. The 'weChatInfo' section is highlighted with a red box, and the 'login_weChat' field in the 'functionConfig' section is also highlighted with a red box. A red text annotation '填写微信的应用信息' (Fill in WeChat application information) points to the 'weChatInfo' section. A blue text annotation '开启微信登录功能' (Enable WeChat login function) points to the 'login_weChat' field.

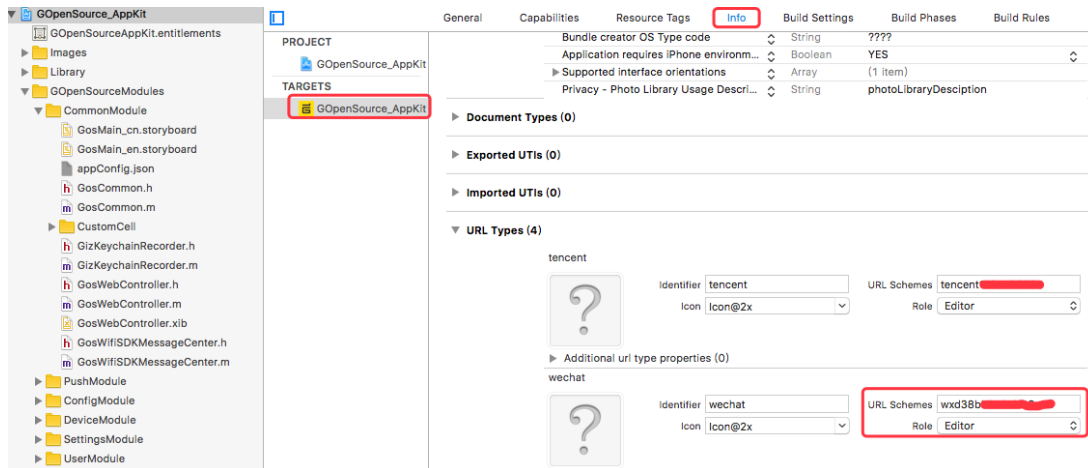
```
12  "tencentInfo": {
13    "appId": "your_tencent_app_id"
14  },
15  "weChatInfo": {
16    "appId": "your_wechat_app_id",
17    "appSecret": "your_wechat_app_secret"
18  },
19  "pushInfo": {
20    "jpushAppKey": "your_jpush_app_key",
21    "bpushAppKey": "your_bpush_app_key"
22  },
23  },
24  "templateSelect": {
25    "deviceList": {
26      "unbindDevice": true,
27      "displayMac": true,
28    }
29  },
30  "functionConfig": {
31    "bindDevice_qrcode": true,
32    "deviceOnboarding": {
33      "config_softap": true,
34      "config_airlink": true,
35      "wifiModuleType": []
36    },
37    "login_anonymous": true,
38    "login_qq": false,
39    "login_weChat": true
```

填写微信的应用信息

开启微信登录功能

1) 配置 URL Schemes

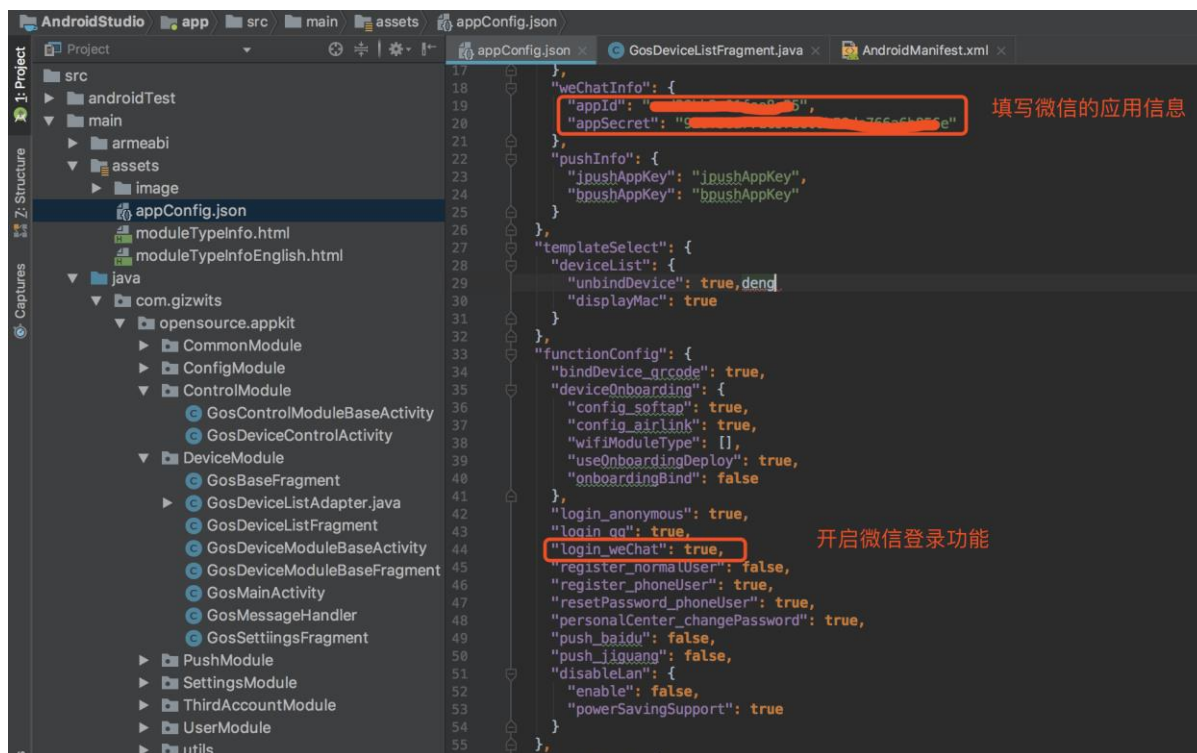
选择微信登陆跳转到微信授权登陆界面，登陆成功后需要跳转回我们的 APP，此时需要给微信一个跳转路径，这个路径就是通过 URL Schemes 来设置的，如下图：



URL Schemes 的修改规则是：微信的 APPID。

4.2.2. Android 登录配置

在 appConfig.json 文件中配置

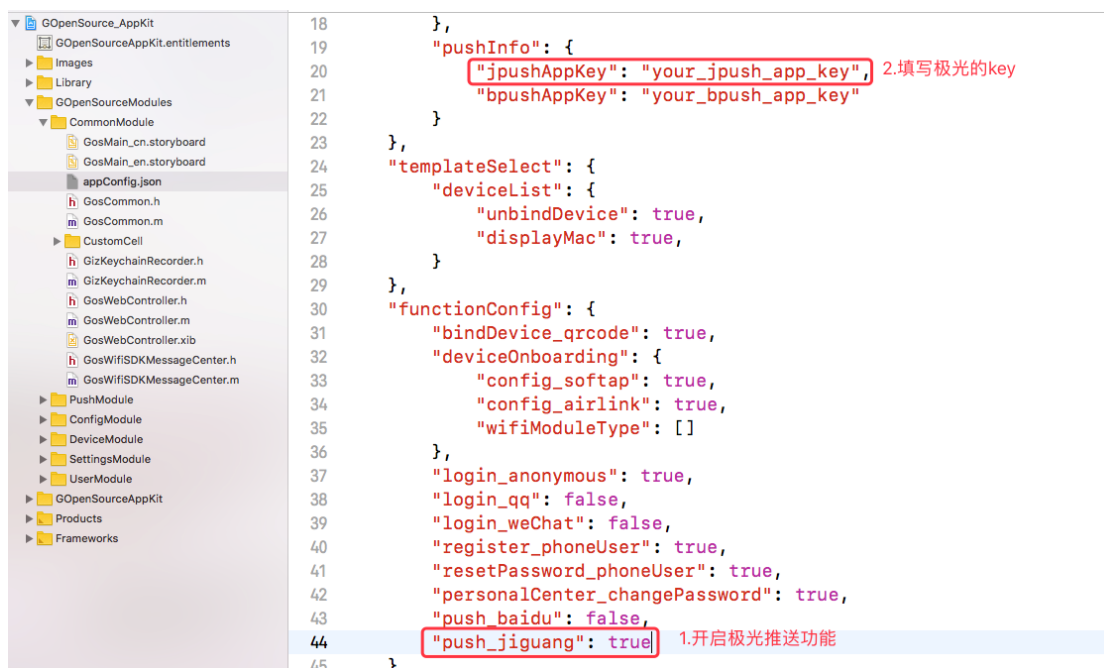


以上的修改是在 Android Studio 中进行的，同样的，Eclipse 也是根据上面的操作进行就可以了。**注意：Android 每次修改完 appConfig.json 的内容，都要卸载当前的 APK,然后重新编译安装，以保证当前 appConfig.json 的内容确实进行了修改。**

5.消息推送

5.1. 极光推送

参考 [iOS App 消息推送集成指南](#)以及 [Android 消息推送集成指南](#)，其中第三部分<修改 UIConfig.json>不可作为参考，在自动生成代码中是要通过配置 appConfig.json 让代码支持极光推送，如下图：



5.2. 百度推送

参考 [iOS App 消息推送集成指南](#)以及 [Android 消息推送集成指南](#)，其中第三部分<修改 UIConfig.json>不可作为参考，在自动生成代码中是要通过配置 appConfig.json 让代码支持百度推送，如下图：

```

18      },
19      "pushInfo": {
20          "jpushAppKey": "your_jpush_app_key",
21          "bpushAppKey": "your_bpush_app_key"
22      },
23      },
24      "templateSelect": {
25          "deviceList": {
26              "unbindDevice": true,
27              "displayMac": true,
28          }
29      },
30      "functionConfig": {
31          "bindDevice_qrcode": true,
32          "deviceOnboarding": {
33              "config_softap": true,
34              "config_airlink": true,
35              "wifiModuleType": []
36          },
37          "login_anonymous": true,
38          "login_qq": false,
39          "login_weChat": false,
40          "register_phoneUser": true,
41          "resetPassword_phoneUser": true,
42          "personalCenter_changePassword": true,
43          "push_baidu": true,
44          "push_jiguang": false
45      },

```

2.填写百度推送APPkey

1.开启百度推送功能

注意：Android 每次修改完 appConfig.json 的内容，都要卸载当前的 APK，然后重新编译安装，以保证当前 appConfig.json 的内容确实进行了修改。