目录

[1 项目简介 4](#_Toc48814947)

[2 安装教程 4](#_Toc48814948)

[2.1 本地开发环境启动步骤 4](#_Toc48814949)

[2.2 生产环境部署 5](#_Toc48814950)

[2.2.1 后台部署 5](#_Toc48814951)

[2.2.2 前端部署 6](#_Toc48814952)

[3 小程序端扫码进入店铺 7](#_Toc48814953)

[3.1 生成店铺小程序码 7](#_Toc48814954)

[3.2 微信扫码进入店铺 8](#_Toc48814955)

[4 上传图片 8](#_Toc48814956)

[4.1 阿里云配置 8](#_Toc48814957)

[4.2 腾讯云配置 10](#_Toc48814958)

[4.3 七牛云配置 11](#_Toc48814959)

[4.4 服务器配置 11](#_Toc48814960)

[5 短信平台 12](#_Toc48814961)

[5.1 登录腾讯云平台 12](#_Toc48814962)

[5.2 创建短信签名 13](#_Toc48814963)

[5.3 创建短信内容 13](#_Toc48814964)

[6 小程序接口文档 13](#_Toc48814965)

[6.1 请求路径 13](#_Toc48814966)

[6.2 使用说明 13](#_Toc48814967)

[7 后台源码分析 15](#_Toc48814968)

[7.1 技术架构图 15](#_Toc48814969)

[7.2 项目介绍 16](#_Toc48814970)

[7.3 代码生成工具的使用 16](#_Toc48814971)

[8 后端源码分析 17](#_Toc48814972)

[8.1 platform-admin模块 17](#_Toc48814973)

[8.1.1 操作日志（切面编程） 17](#_Toc48814974)

[8.1.2 当前登录用户 18](#_Toc48814975)

[8.1.3 数据权限实现（mybatis拦截器实现） 19](#_Toc48814976)

[8.2 platform-api模块 21](#_Toc48814977)

[8.2.1 当前登录会员 21](#_Toc48814978)

[8.2.2 不需要token验证的接口 22](#_Toc48814979)

[8.3 统一异常处理 23](#_Toc48814980)

[8.3.1 后台异常处理 23](#_Toc48814981)

[8.3.2 前端统一异常处理 25](#_Toc48814982)

[8.4 限流 25](#_Toc48814983)

[8.5 同步锁 26](#_Toc48814984)

[8.6 定时任务 26](#_Toc48814985)

[9 前端源码分析 27](#_Toc48814986)

[9.1 技术栈 27](#_Toc48814987)

[9.2 项目结构 28](#_Toc48814988)

[9.3 主题定制 28](#_Toc48814989)

[9.4 菜单图标 29](#_Toc48814990)

[9.5 挂载全局的公共方法 29](#_Toc48814991)

[9.6 按钮权限控制 29](#_Toc48814992)

[9.7 页面组件 29](#_Toc48814993)

[10 物流查询 30](#_Toc48814994)

[11 微信公众号配置 31](#_Toc48814995)

[12 物流相关 32](#_Toc48814996)

[13 项目所需材料 35](#_Toc48814997)

[13.1 微信注册认证 35](#_Toc48814998)

[13.2 微信支付所需材料 35](#_Toc48814999)

[13.3 服务器 35](#_Toc48815000)

[13.4 其他材料 36](#_Toc48815001)

[14 系统所需配置参数 36](#_Toc48815002)

[14.1 wx.appId、wx.secret 36](#_Toc48815003)

[14.2 wx.pay.mchId、wx.pay.mchkey 36](#_Toc48815004)

[14.3 wx.pay.basenotifyurl 36](#_Toc48815005)

[14.4 wx.spbillcreateip 36](#_Toc48815006)

[14.5 设置退款回调地址 37](#_Toc48815007)

[14.6 业务域名 37](#_Toc48815008)

[14.7 订阅消息 37](#_Toc48815009)

[14.8 飞鹅打印机配置 40](#_Toc48815010)

[14.8.1 购买 40](#_Toc48815011)

[14.8.2 注册 40](#_Toc48815012)

[14.8.3 实名认证 41](#_Toc48815013)

[14.8.4 打票机上线 41](#_Toc48815014)

[15 添加插件 41](#_Toc48815015)

[15.1 “物流轨迹”插件 41](#_Toc48815016)

[15.2 “直播”插件 42](#_Toc48815017)

# 项目简介

本项目支持客户到店下单、和线上直接下单。

小程序首页展示店铺、商品列表，在店铺下单取货方式为店铺自取，不需要运费，也不需要使用收货地址。

体验：



# 安装教程

## 本地开发环境启动步骤

\* 创建数据库 platform-plus

\* 字符集 : utf8mb4 -- UTF-8 Unicode

\* 排序规则 : utf8mb4\_general\_ci

\* 导入\_sql/platform-plus.sql

\* 推荐环境（nginx、jdk1.8、mysql5.7、redis ）

\* 导入项目到IDE中

\* 导入支付证书至/platform-api/src/main/resources/目录下

\* 导入支付证书至/platform-admin/src/main/resources/目录下

\* 修改开发环境配置/platform-api/src/main/resources/application.yml

\* 修改开发环境配置/platform-api/src/main/resources/application-dev.yml

\* 修改开发环境配置/platform- admin /src/main/resources/application.yml

\* 修改开发环境配置/platform- admin /src/main/resources/application-dev.yml

\* 使用IDE启动项目

\* 将wx-mall导入微信web开发者工具填写appId

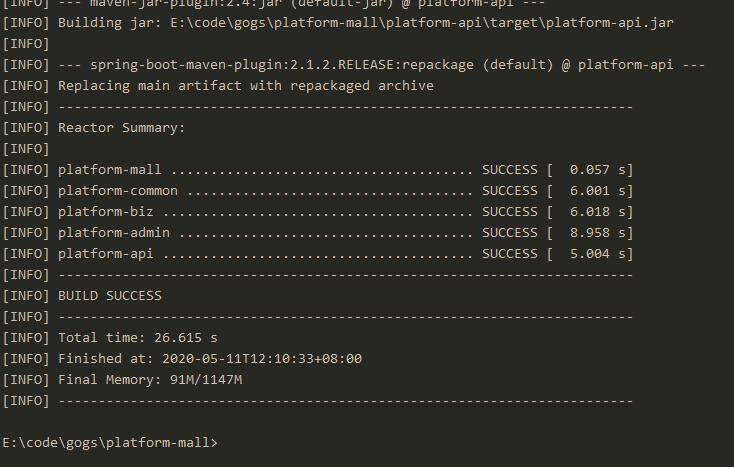
\* 修改 /wx-mall/utils/utils.js里API\_BASE\_URL的值

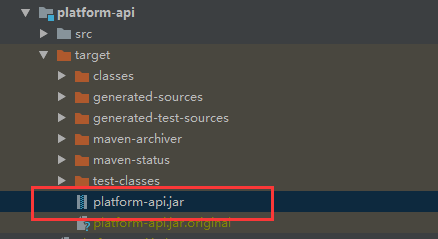
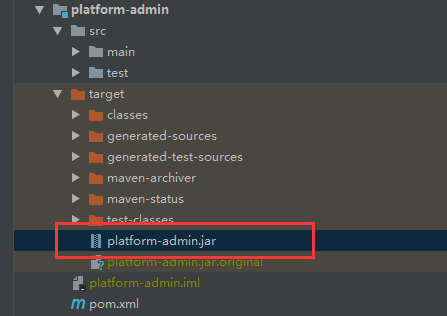
## 生产环境部署

### 后台部署

platform-mall> mvn clean

platform-mall> mvn package -P prod





将platform-admin.jar、platform-api.jar拷贝到服务器，下面两个启动脚本放在同一目录，分别执行启动两个服务（小程序接口服务、后台管理系统服务）。需要配置权限

chmod 755 startPlatformApiService.sh

chmod 755 startPlatformAdminService.sh



### 前端部署

进入到platform-admin-ui目录，修改/static/config/index-prod.js文件

window.SITE\_CONFIG['baseUrl'] = 'http://你的域名/platform-admin' // 后台接口请求地址

//如果使用第三方文件存储，将dist/1907180922 上传至第三方文件存储，然后填写cdn地址

window.SITE\_CONFIG['domain'] = '静态资源cdn地址';

*# 安装依赖*

*npm install*

*# 构建生产环境*

*npm run build*

完成之后将dist目录上传到服务器，然后配置nginx

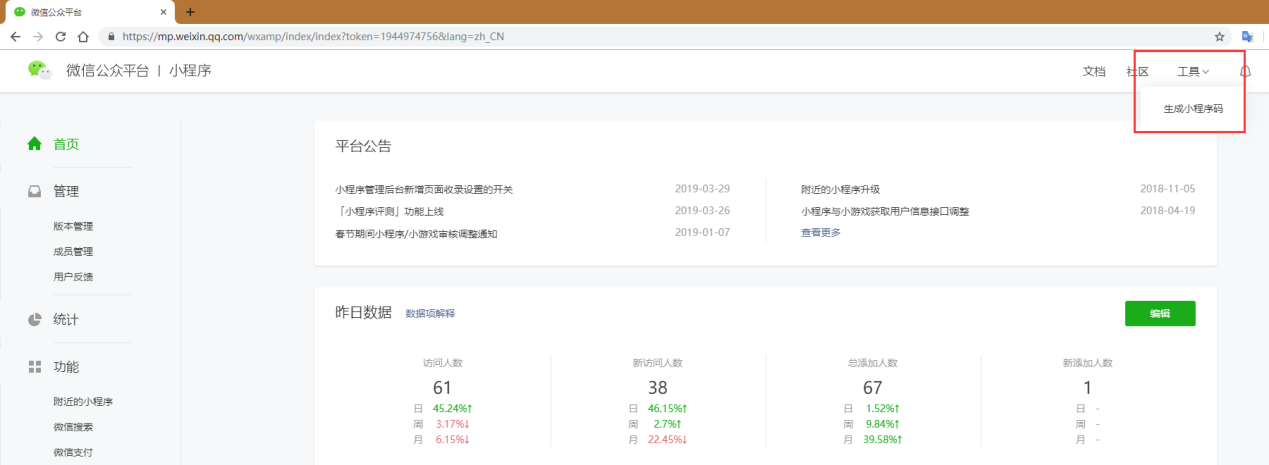
*location / {  
 # 指向我们打包后上传的前端文件  
 root /usr/local/nginx/dist;  
 index index.html;  
 }  
 location /platform-admin {  
 # 转发请求到后端  
 proxy\_pass http://你的域名:8888/platform-admin;  
 proxy\_set\_header Host $host;  
 proxy\_set\_header X-Real-IP $remote\_addr;  
 proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;  
 }  
 location /platform-api {  
 # 转发请求到后端  
 proxy\_pass http://你的域名:8889/platform-api;  
 proxy\_set\_header Host $host;  
 proxy\_set\_header X-Real-IP $remote\_addr;  
 proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;  
 }*

重启nginx，现在就可以正式访问了。

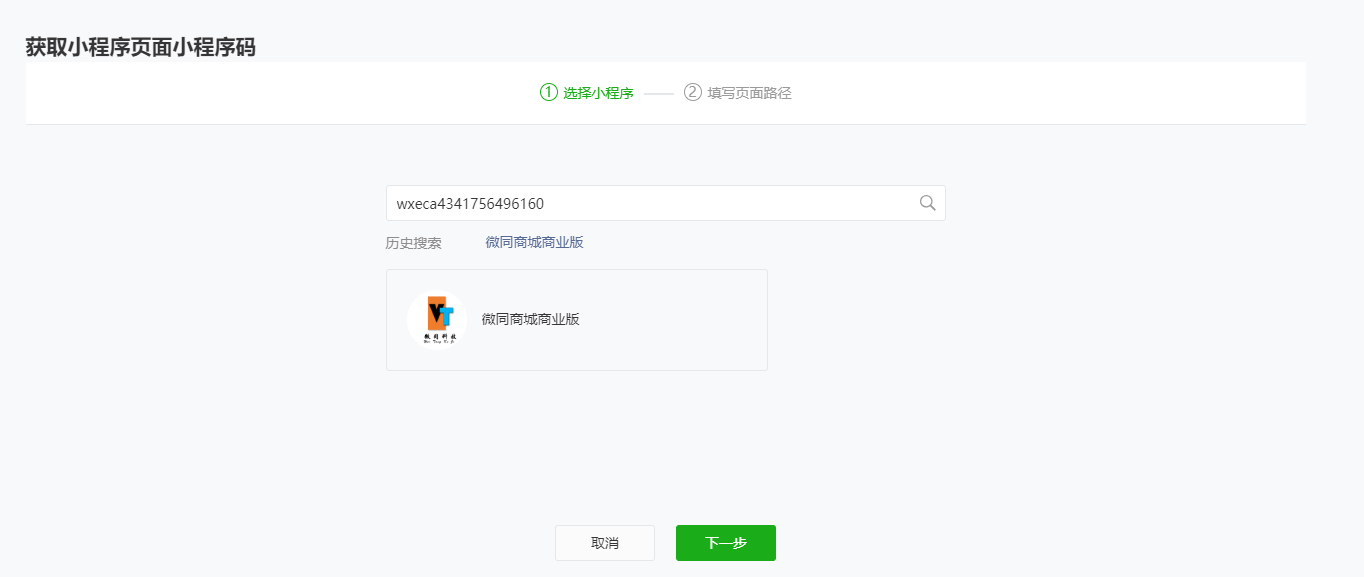
小程序访问接口需要SSL证书，可以参照一下<http://fly2you.cn/jie/page/130>

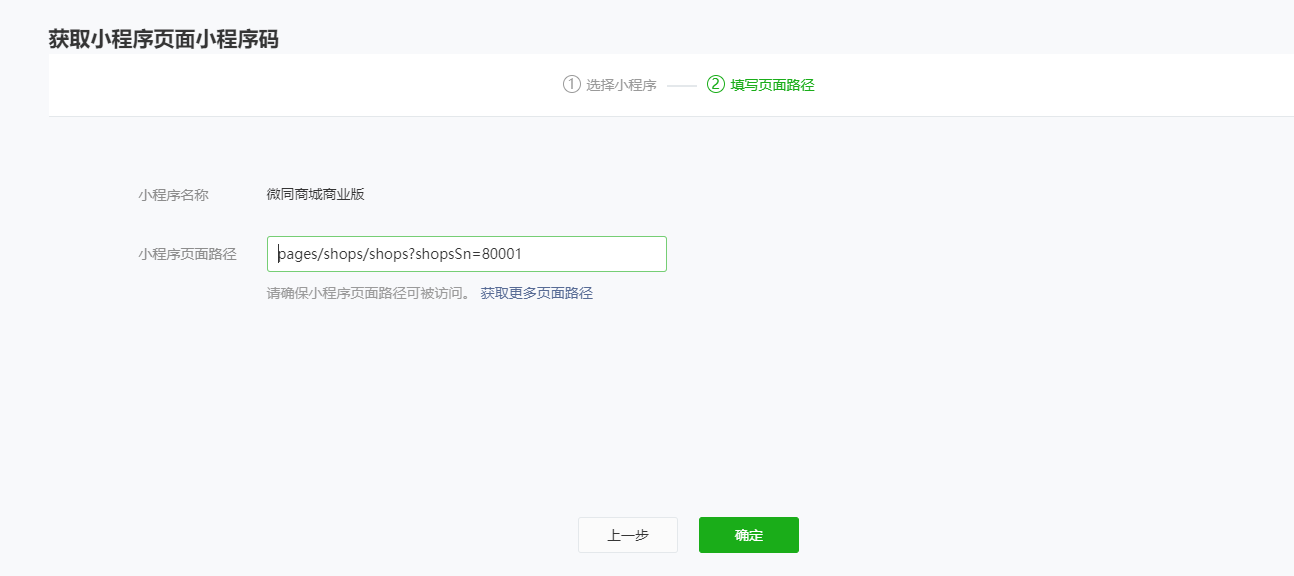
# 小程序端扫码进入店铺

## 生成店铺小程序码



获取小程序页面小程序码：





生成完成，保存下图中的小程序码，即可扫码进入店铺首页



## 微信扫码进入店铺

使用微信扫码上一步生成的小程序码即可进入店铺首页

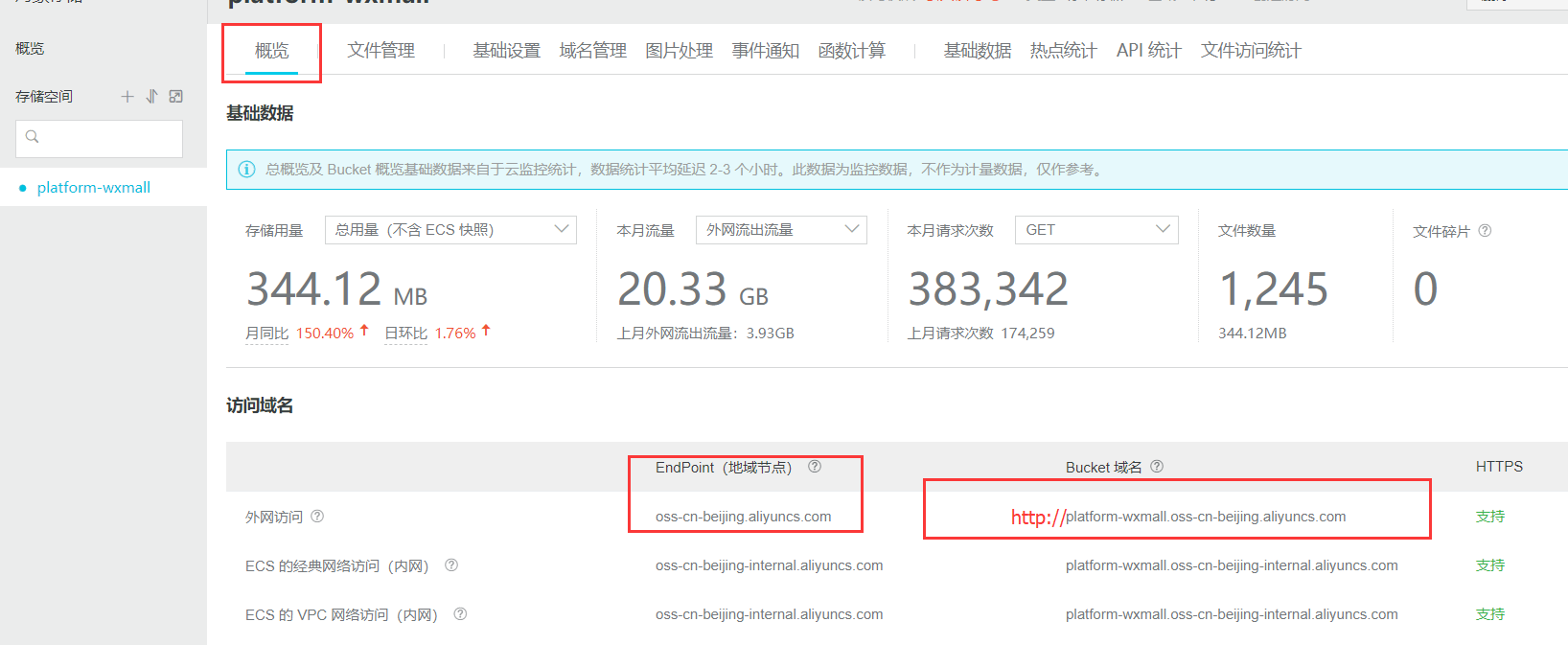
# 上传图片

需要申请阿里云、腾讯云、七牛云存储

## 阿里云配置

登录阿里云控制台，获取配置信息

域名（Bucket域名）、EndPoint：



路径前缀：自己定义（如upload）

Bucket：

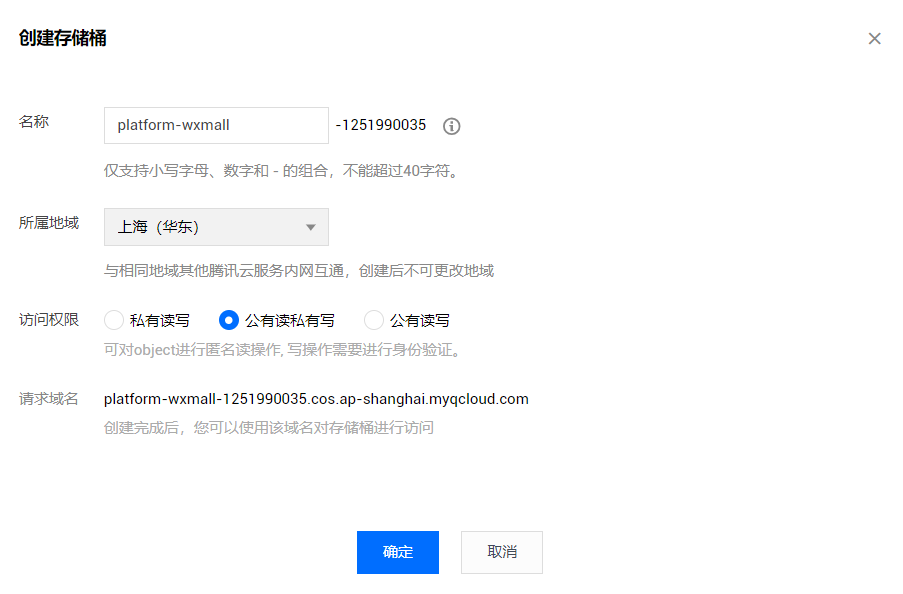
AccessKeyId、AccessKeySecret：





## 腾讯云配置

登录腾讯云控制台创建存储桶



域名、Bucket所属地区、BucketName



AppId、SecretId、SecretKey



## 七牛云配置

登录七牛云控制台，创建公开空间

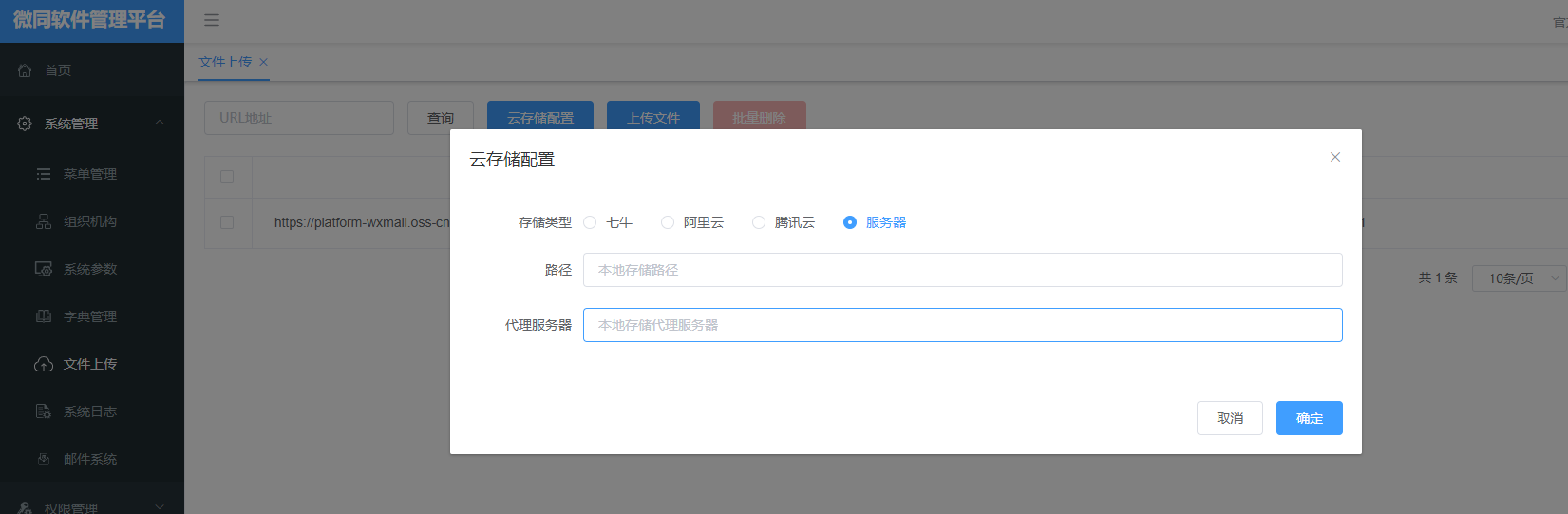
空间名、域名：



AccessKey、SecretKey



## 服务器配置

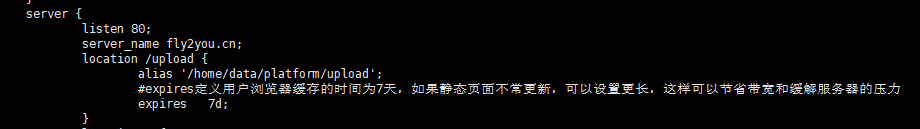


存储路径：服务器存放图片的路径。egg：/home/data/platform/upload

代理服务器：该配置是代理服务器的地址。egg：<http://fly2you.cn/upload>

代理服务器访问的其实就是我们服务器上的目录 /home/data/platform/upload

该例使用nginx代理，下面贴上nginx配置：



# 短信平台

本系统已集成腾讯云短信平台功能

## 登录腾讯云平台

<https://console.cloud.tencent.com/>



点击应用名称

## 创建短信签名



## 创建短信内容

待签名审核通过后再添加短信内容，egg：

您的验证码是{1}，请于{2}分钟内填写。如非本人操作，请忽略本短信。



将此ID写到application.yml文件tx.templateId

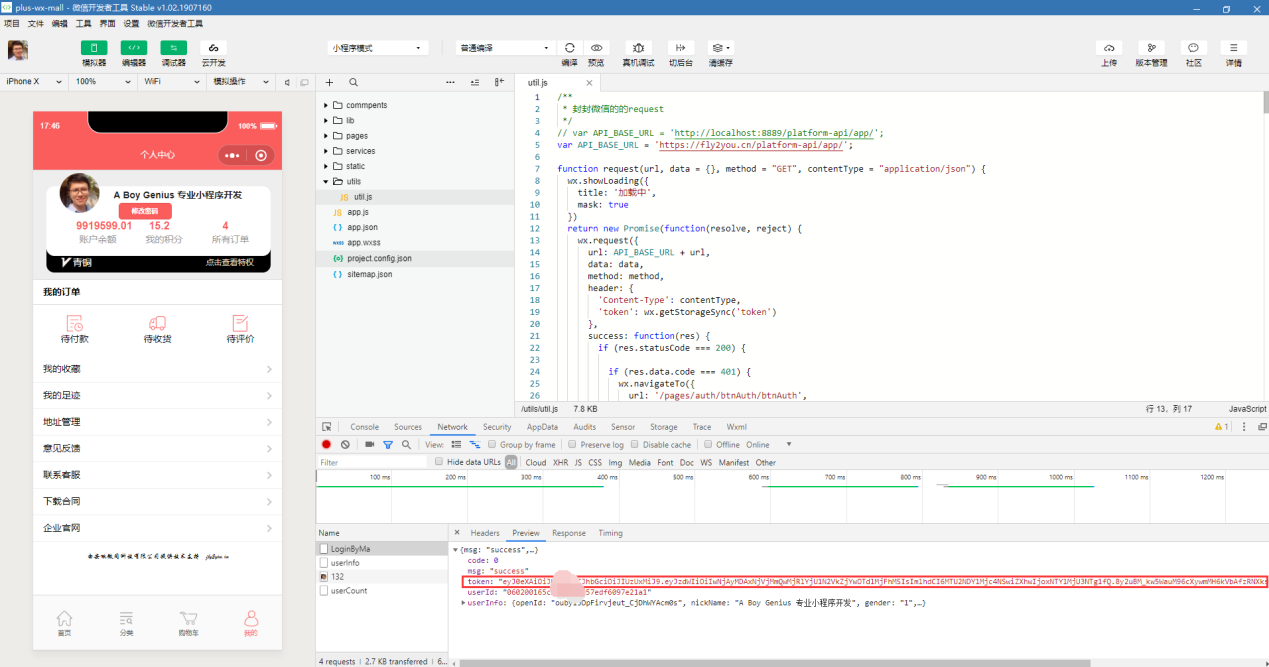
# 小程序接口文档

## 请求路径

<http://fly2you.cn/platform-api/doc.html>

## 使用说明

先使用微信web开发者工具登录，查看返回的token，如下图所示

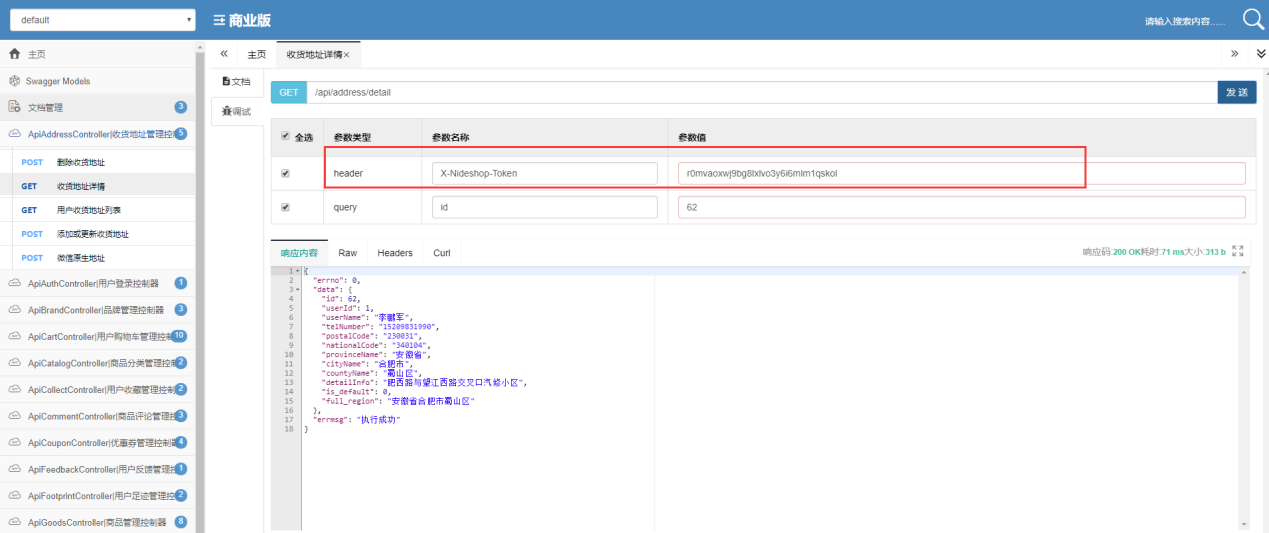


记录token的值，登录接口文档地址，进入文档管理->全局参数设置，添加一条参数参数名：token，参数值：token，参数类型：header

如下图所示

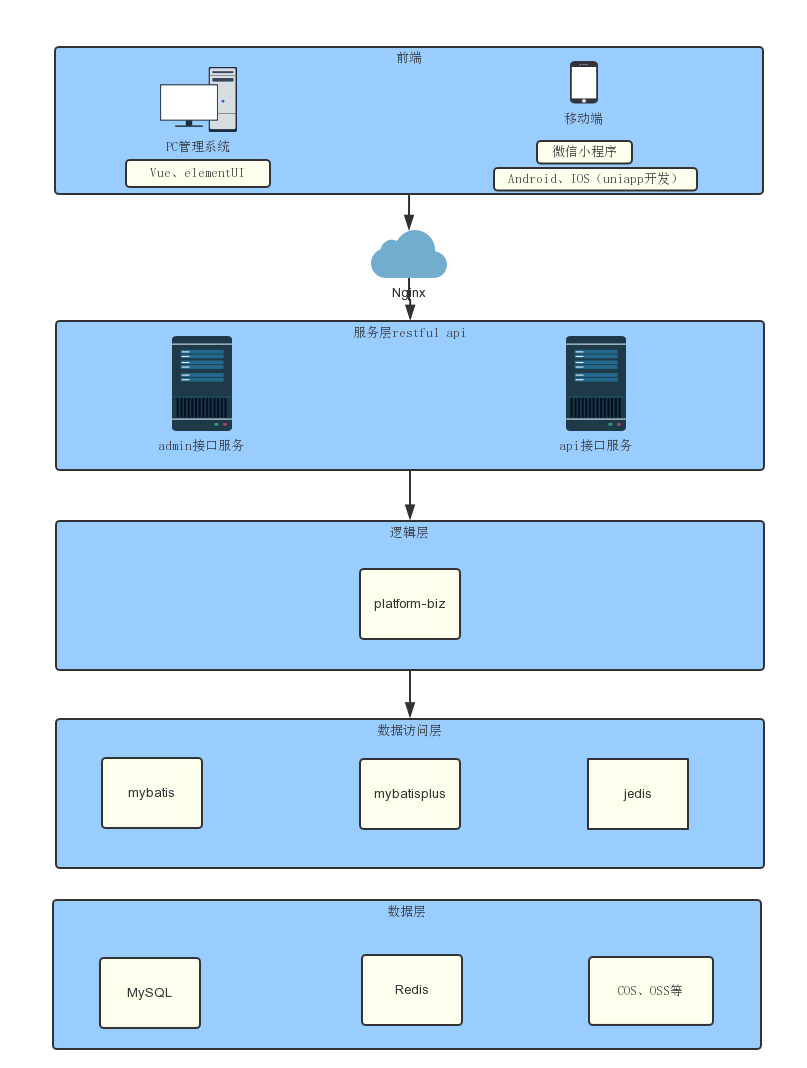


这样就可以调用需要用户登录的接口，不需要每次发起接口请求都填写token值了。如下图所示



# 后台源码分析

## 技术架构图



## 项目介绍

platform-admin：后台管理系统接口服务，端口：8888

platform-admin-ui：后台管理系统前端页面

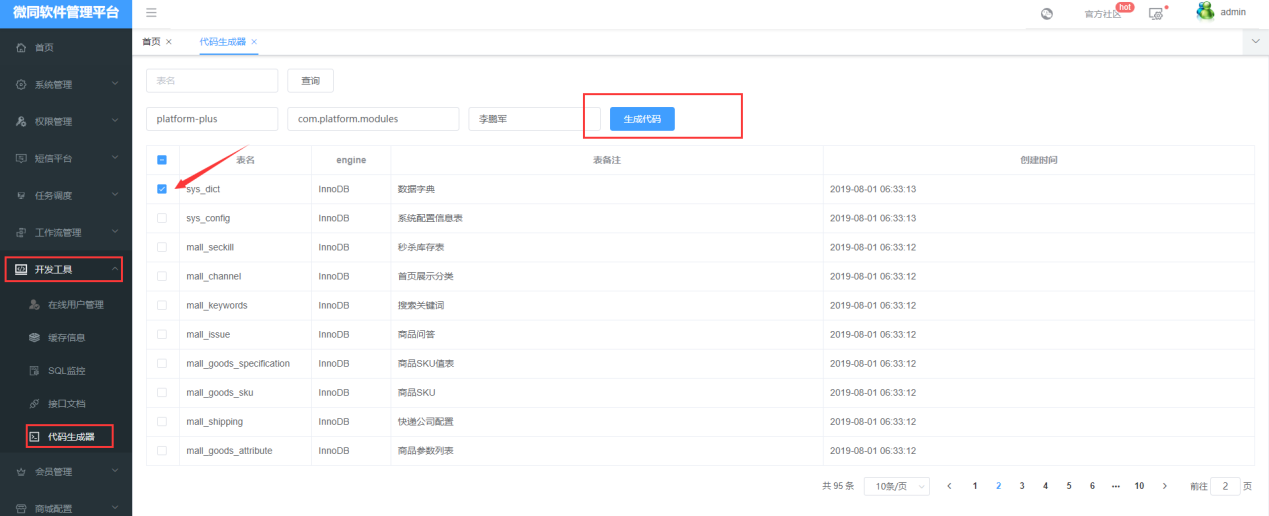
platform-api：移动端接口服务，端口：8889

platform-biz：业务处理层（admin、api共用biz层）

platform-common：通用工具

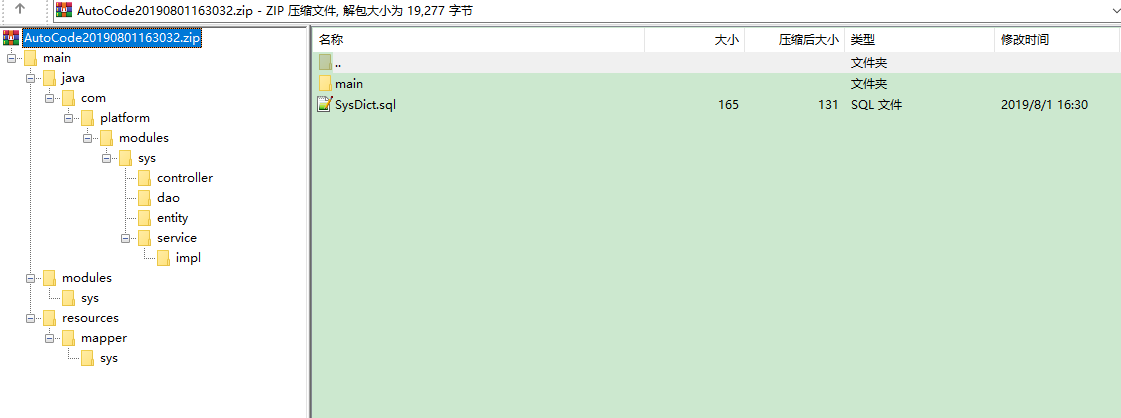
wx-mall：微信小程序代码

## 代码生成工具的使用



在系统打开代码生成器页面，选中表（支持多选，一次生成多张表的代码），点击生成代码即可。

生成的代码结构如下：



解压，然后将

main/java/com/platform/modules/sys/controller复制到platform-admin\src\main\java\com\platform\modules\sys\controller

main/java/com/platform/modules/sys/dao复制到platform-biz\src\main\java\com\platform\modules\sys\dao

main/java/com/platform/modules/sys/entity复制到platform-biz\src\main\java\com\platform\modules\sys\entity

main/java/com/platform/modules/sys/service复制到platform-biz\src\main\java\com\platform\modules\sys\service

main/modules/sys复制到platform-admin-ui/src/views/modules

main/resources/mapper/sys/复制到platform-biz\src\main\ resources \mapper\sys

在菜单管理新增菜单，菜单路由、授权标识参照生成SysDict.sql

# 后端源码分析

## platform-admin模块

### 操作日志（切面编程）

#### 实现类

/\*\*  
 \* 系统日志，切面处理类  
 \*  
 \* @author 李鹏军  
 \*/  
@Aspect  
@Component  
public class SysLogAspect {  
 @Autowired  
 private SysLogService sysLogService;  
  
 @Pointcut("@annotation(com.platform.common.annotation.SysLog)")  
 public void logPointCut() {  
  
 }  
  
 @Around("logPointCut()")  
 public Object around(ProceedingJoinPoint point) throws Throwable {  
 long beginTime = System.currentTimeMillis();  
 //执行方法  
 Object result = point.proceed();  
 //执行时长(毫秒)  
 long time = System.currentTimeMillis() - beginTime;  
  
 //保存日志  
 saveSysLog(point, time);  
  
 return result;  
 }  
  
 private void saveSysLog(ProceedingJoinPoint joinPoint, long time) {  
 MethodSignature signature = (MethodSignature) joinPoint.getSignature();  
 Method method = signature.getMethod();  
  
 SysLogEntity sysLog = new SysLogEntity();  
 SysLog syslog = method.getAnnotation(SysLog.class);  
 if (syslog != null) {  
 //注解上的描述  
 sysLog.setOperation(syslog.value());  
 }  
  
 //请求的方法名  
 String className = joinPoint.getTarget().getClass().getName();  
 String methodName = signature.getName();  
 sysLog.setMethod(className + "." + methodName + "()");  
  
 //请求的参数  
 Object[] args = joinPoint.getArgs();  
 try {  
 String params = new Gson().toJson(args);  
 sysLog.setParams(params);  
  
 //获取request  
 HttpServletRequest request = HttpContextUtils.getHttpServletRequest();  
 //设置IP地址  
 sysLog.setIp(IpUtils.getIpAddr(request));  
  
 //用户名  
 String userName = ((SysUserEntity) SecurityUtils.getSubject().getPrincipal()).getUserName();  
 sysLog.setUserName(userName);  
  
 sysLog.setTime(time);  
 sysLog.setCreateTime(new Date());  
 //保存系统日志  
 sysLogService.save(sysLog);  
 } catch (Exception ignored) {  
  
 }  
 }  
}

#### 使用方法

/\*\*  
 \* 保存  
 \*  
 \* @return RestResponse  
 \*/  
@SysLog("保存数据字典")  
@RequestMapping("/save")  
@RequiresPermissions("sys:dict:save")  
public RestResponse save(@RequestBody SysDictEntity sysDict) {  
 ValidatorUtils.validateEntity(sysDict, AddGroup.class);  
 sysDictService.add(sysDict);  
  
 return RestResponse.success();  
}

### 当前登录用户

#### 实现类：

/\*\*  
 \* Shiro工具类  
 \*  
 \* @author 李鹏军  
 \*/  
public class ShiroUtils {  
  
 public static Session getSession() {  
 return SecurityUtils.getSubject().getSession();  
 }  
  
 public static Subject getSubject() {  
 return SecurityUtils.getSubject();  
 }  
  
 public static SysUserEntity getUserEntity() {  
 return (SysUserEntity) SecurityUtils.getSubject().getPrincipal();  
 }  
  
 public static String getUserId() {  
 return getUserEntity().getUserId();  
 }  
  
 public static void setSessionAttribute(Object key, Object value) {  
 getSession().setAttribute(key, value);  
 }  
  
 public static Object getSessionAttribute(Object key) {  
 return getSession().getAttribute(key);  
 }  
  
 public static boolean isLogin() {  
 return SecurityUtils.getSubject().getPrincipal() != null;  
 }  
  
 public static String getKaptcha(String key) {  
 Object kaptcha = getSessionAttribute(key);  
 if (kaptcha == null) {  
 throw new BusinessException("验证码已失效");  
 }  
 getSession().removeAttribute(key);  
 return kaptcha.toString();  
 }  
}

#### 使用方法

ShiroUtils.getUserEntity();

### 数据权限实现（mybatis拦截器实现）

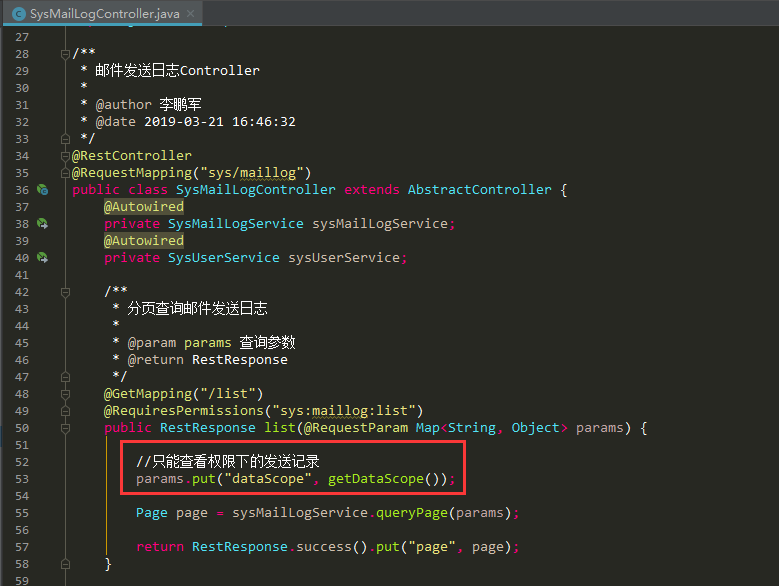
#### 实现类

/\*\*  
 \* @author 李鹏军  
 \* <p>  
 \* mybatis 数据权限拦截器  
 \*/  
@Slf4j  
@Intercepts({@Signature(type = StatementHandler.class, method = "prepare", args = {Connection.class, Integer.class})})  
public class DataScopeInterceptor extends SqlExplainInterceptor implements Interceptor {  
  
 @Override  
 public Object intercept(Invocation invocation) throws Throwable {  
 StatementHandler statementHandler = PluginUtils.realTarget(invocation.getTarget());  
 MetaObject metaObject = SystemMetaObject.forObject(statementHandler);  
 this.sqlParser(metaObject);  
  
 // 不是SELECT操作直接返回  
 MappedStatement mappedStatement = (MappedStatement) metaObject.getValue("delegate.mappedStatement");  
 if (!SqlCommandType.***SELECT***.equals(mappedStatement.getSqlCommandType())) {  
 return invocation.proceed();  
 }  
  
 BoundSql boundSql = (BoundSql) metaObject.getValue("delegate.boundSql");  
 StringBuilder filterSql = new StringBuilder(boundSql.getSql());  
 Object parameterObject = boundSql.getParameterObject();  
  
 //参数中DataScope类型的参数  
 DataScope dataScope = findDataScopeObject(parameterObject);  
  
 if (dataScope == null) {  
 return invocation.proceed();  
 } else {  
 SysUserEntity user = ShiroUtils.getUserEntity();  
 if (null != user) {  
 //如果不是超级管理员，则只能查询本机构及子机构数据  
 if (!Constant.***SUPER\_ADMIN***.equals(user.getUserId())) {  
 String userAlias = dataScope.getUserAlias();  
 String orgAlias = dataScope.getOrgAlias();  
 String alias = dataScope.getOrgNos();  
 boolean self = dataScope.getSelf();  
  
 if (StringUtils.isNotBlank(alias)) {  
 filterSql.append(" and (").append(orgAlias).append(" in (").append(alias).append(")");  
 if (self) {  
 filterSql.append(" or ").append(userAlias).append("='").append(user.getUserId()).append("' ");  
 }  
 filterSql.append(" ) ");  
 } else if (self) {  
 filterSql.append(" and ").append(userAlias).append("='").append(user.getUserId()).append("' ");  
 }  
 }  
 metaObject.setValue("delegate.boundSql.sql", filterSql.toString());  
 }  
 return invocation.proceed();  
 }  
 }  
  
 /\*\*  
 \* 生成拦截对象的代理  
 \*  
 \* @param target 目标对象  
 \* @return 代理对象  
 \*/  
 @Override  
 public Object plugin(Object target) {  
 if (target instanceof StatementHandler) {  
 return Plugin.wrap(target, this);  
 }  
 return target;  
 }  
  
 /\*\*  
 \* mybatis配置的属性  
 \*  
 \* @param properties mybatis配置的属性  
 \*/  
 @Override  
 public void setProperties(Properties properties) {  
  
 }  
  
 /\*\*  
 \* 查找参数是否包括DataScope对象  
 \*  
 \* @param parameterObj 参数列表  
 \* @return DataScope  
 \*/  
 private DataScope findDataScopeObject(Object parameterObj) {  
 if (parameterObj instanceof DataScope) {  
 return (DataScope) parameterObj;  
 } else if (parameterObj instanceof Map) {  
 for (Object val : ((Map<?, ?>) parameterObj).values()) {  
 if (val instanceof DataScope) {  
 return (DataScope) val;  
 } else {  
 if (val instanceof Map) {  
 for (Object v : ((Map<?, ?>) val).values()) {  
 if (v instanceof DataScope) {  
 return (DataScope) v;  
 }  
 }  
 }  
 }  
 }  
 }  
 return null;  
 }  
}

#### 使用方法

在查询的参数中添加一行代码即可实现

//只能查看权限下的发送记录  
params.put("dataScope", getDataScope());



## platform-api模块

### 当前登录会员

在controller层使用@LoginUser即可。例如：



实现类：

/\*\*  
 \* 有@LoginUser注解的方法参数，注入当前登录用户  
 \*  
 \* @author 李鹏军  
 \*/  
@Component  
public class LoginUserHandlerMethodArgumentResolver implements HandlerMethodArgumentResolver {  
 @Autowired  
 private MallUserService userService;  
  
 @Override  
 public boolean supportsParameter(MethodParameter parameter) {  
 return parameter.getParameterType().isAssignableFrom(MallUserEntity.class) && parameter.hasParameterAnnotation(LoginUser.class);  
 }  
  
 @Override  
 public Object resolveArgument(MethodParameter parameter, ModelAndViewContainer container,  
 NativeWebRequest request, WebDataBinderFactory factory) {  
 //获取用户ID  
 Object object = request.getAttribute(AuthorizationInterceptor.***USER\_KEY***, RequestAttributes.***SCOPE\_REQUEST***);  
 if (object == null) {  
 return null;  
 }  
  
 //获取用户信息  
  
 return userService.getById((String) object);  
 }  
}

### 不需要token验证的接口

在controller使用@IgnoreAuth即可。例如：



实现类：

/\*\*  
 \* 权限(Token)验证  
 \*  
 \* @author 李鹏军  
 \*/  
@Component  
public class AuthorizationInterceptor extends HandlerInterceptorAdapter {  
 @Autowired  
 private JwtUtils jwtUtils;  
  
 public static final String ***USER\_KEY*** = "LOGIN\_USER\_KEY";  
  
 @Override  
 public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler) {  
  
 //支持跨域请求  
 response.setHeader("Access-Control-Allow-Methods", "GET, POST, PUT, DELETE, OPTIONS");  
 response.setHeader("Access-Control-Max-Age", "3600");  
 response.setHeader("Access-Control-Allow-Credentials", "true");  
 response.setHeader("Access-Control-Allow-Headers", "x-requested-with,content-type");  
 response.setHeader("Access-Control-Allow-Origin", request.getHeader("Origin"));  
  
 IgnoreAuth annotation;  
 if (handler instanceof HandlerMethod) {  
 annotation = ((HandlerMethod) handler).getMethodAnnotation(IgnoreAuth.class);  
 } else {  
 return true;  
 }  
  
 //如果有@IgnoreAuth注解，则不验证token  
 if (annotation != null) {  
 return true;  
 }  
  
 //获取用户凭证  
 String token = request.getHeader(jwtUtils.getHeader());  
 if (StringUtils.isBlank(token)) {  
 token = request.getParameter(jwtUtils.getHeader());  
 }  
  
 //凭证为空  
 if (StringUtils.isBlank(token)) {  
 throw new BusinessException(jwtUtils.getHeader() + "不能为空", HttpStatus.***UNAUTHORIZED***.value());  
 }  
  
 Claims claims = jwtUtils.getClaimByToken(token);  
 if (claims == null || jwtUtils.isTokenExpired(claims.getExpiration())) {  
 throw new BusinessException(jwtUtils.getHeader() + "失效，请重新登录", HttpStatus.***UNAUTHORIZED***.value());  
 }  
  
 //设置userId到request里，后续根据userId，获取用户信息  
 request.setAttribute(***USER\_KEY***, claims.getSubject());  
  
 return true;  
 }  
}

## 统一异常处理

### 后台异常处理

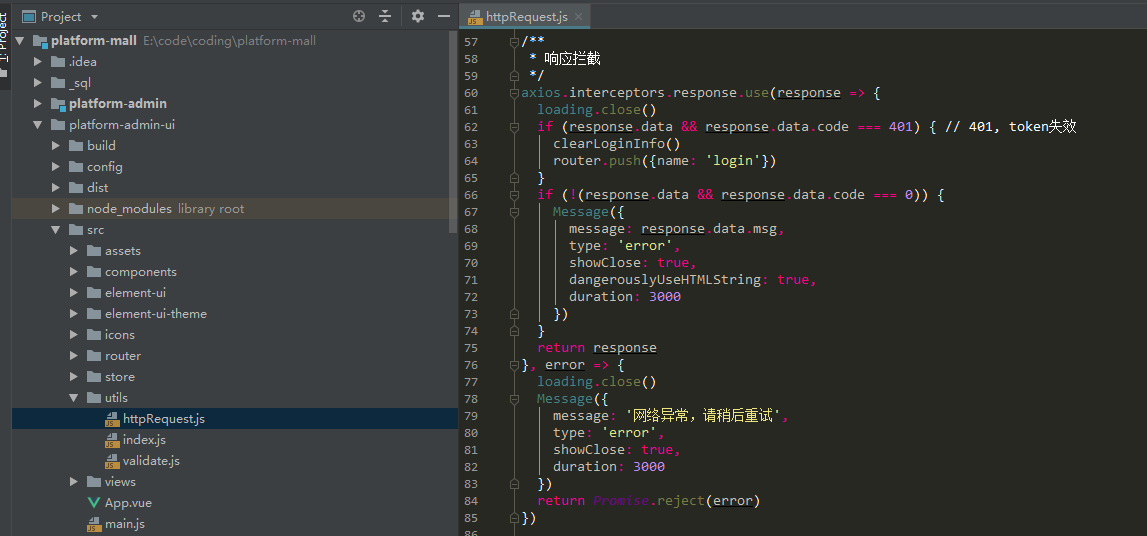
本项目通过 BusinessException 异常类，抛出自定义异常，BusinessException继承 RuntimeException，不能继承 Exception，如果继承 Exception，则 Spring 事务不会回滚。

/\*\*  
 \* 自定义异常  
 \*  
 \* @author 李鹏军  
 \*/  
public class BusinessException extends RuntimeException {  
 private static final long ***serialVersionUID*** = 1L;  
  
 private String msg;  
 private int code = HttpStatus.***INTERNAL\_SERVER\_ERROR***.value();  
  
 public BusinessException(String msg) {  
 super(msg);  
 this.msg = msg;  
 }  
  
 public BusinessException(String msg, Throwable e) {  
 super(msg, e);  
 this.msg = msg;  
 }  
  
 public BusinessException(String msg, int code) {  
 super(msg);  
 this.msg = msg;  
 this.code = code;  
 }  
  
 public BusinessException(String msg, int code, Throwable e) {  
 super(msg, e);  
 this.msg = msg;  
 this.code = code;  
 }  
  
 public String getMsg() {  
 return msg;  
 }  
  
 public void setMsg(String msg) {  
 this.msg = msg;  
 }  
  
 public int getCode() {  
 return code;  
 }  
  
 public void setCode(int code) {  
 this.code = code;  
 }  
}

我们定义了 BusinessExceptionHandler 类，并加上注解@RestControllerAdvice，就可以处理所有抛出的异常，并返回 JSON 数据。

/\*\*  
 \* 异常处理器  
 \*  
 \* @author 李鹏军  
 \*/  
@Slf4j  
@RestControllerAdvice  
public class BusinessExceptionHandler {  
  
 /\*\*  
 \* 处理自定义异常  
 \*/  
 @ExceptionHandler(BusinessException.class)  
 public RestResponse handleBusinessException(BusinessException e) {  
 RestResponse restResponse = new RestResponse();  
 restResponse.put("code", e.getCode());  
 restResponse.put("msg", e.getMessage());  
  
 return restResponse;  
 }  
  
 @ExceptionHandler(NoHandlerFoundException.class)  
 public RestResponse handlerNoFoundException(Exception e) {  
 ***log***.error(e.getMessage(), e);  
 return RestResponse.error(HttpStatus.***NOT\_FOUND***.value(), "路径不存在，请检查路径是否正确");  
 }  
  
 @ExceptionHandler(DuplicateKeyException.class)  
 public RestResponse handleDuplicateKeyException(DuplicateKeyException e) {  
 ***log***.error(e.getMessage(), e);  
 return RestResponse.error("数据库中已存在该记录");  
 }  
  
 @ExceptionHandler(AuthorizationException.class)  
 public RestResponse handleAuthorizationException(AuthorizationException e) {  
 ***log***.error(e.getMessage(), e);  
 return RestResponse.error("没有权限，请联系管理员授权");  
 }  
  
 @ExceptionHandler(Exception.class)  
 public RestResponse handleException(Exception e) {  
 ***log***.error(e.getMessage(), e);  
 return RestResponse.error();  
 }  
}

### 前端统一异常处理

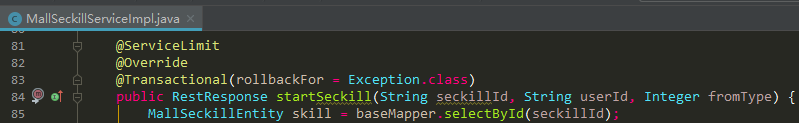


## 限流

采用令牌桶算法实现，单进程服务的限流，每秒发5个令牌

/\*\*  
 \* 限流  
 \*  
 \* @author 李鹏军  
 \*/  
@Component  
@Scope  
@Aspect  
public class LimitAspect {  
 /\*\*  
 \* 每秒只发出5个令牌，此处是单进程服务的限流,内部采用令牌捅算法实现  
 \*/  
 private static RateLimiter rateLimiter = RateLimiter.create(5.0);  
  
 /\*\*  
 \* Service层切点 限流  
 \*/  
 @Pointcut("@annotation(com.platform.common.annotation.ServiceLimit)")  
 public void ServiceAspect() {  
 }  
  
 @Around("ServiceAspect()")  
 public Object around(ProceedingJoinPoint joinPoint) {  
 Boolean flag = rateLimiter.tryAcquire();  
 Object obj = null;  
 try {  
 if (flag) {  
 obj = joinPoint.proceed();  
 }  
 } catch (Throwable e) {  
 e.printStackTrace();  
 }  
 return obj;  
 }  
}

使用案例：秒杀接口（在接口上使用@ServiceLimit）



## 同步锁

进程内锁，程序部署多节点会失效

/\*\*  
 \* 同步锁 AOP  
 \*  
 \* @author 李鹏军  
 \*/  
@Component  
@Scope  
@Aspect  
@Order(1)  
public class LockAspect {  
 /\*\*  
 \* 不用synchronized service 默认是单例的，并发下lock只有一个实例  
 \* <p>  
 \* 互斥锁 参数默认false，不公平锁  
 \*/  
 private static Lock lock = new ReentrantLock(true);  
  
 /\*\*  
 \* Service层切点  
 \*/  
 @Pointcut("@annotation(com.platform.common.annotation.ServiceLock)")  
 public void lockAspect() {  
  
 }  
  
 @Around("lockAspect()")  
 public Object around(ProceedingJoinPoint joinPoint) {  
 lock.lock();  
 Object obj;  
 try {  
 obj = joinPoint.proceed();  
 } catch (Throwable e) {  
 e.printStackTrace();  
 throw new RuntimeException();  
 } finally {  
 lock.unlock();  
 }  
 return obj;  
 }  
}

## 定时任务

本系统使用开源框架 Quartz，实现的定时任务，已实现分布式定时任务，可部署多台服务器，不重复执行，以及动态增加、修改、删除、暂停、恢复、立即执行定时任务。

新增一个定时任务，其实很简单，只要定义一个普通的 Spring Bean，然后在页面添加定时任务

/\*\*  
 \* 订单定时任务  
 \* orderTask为spring bean的名称  
 \*  
 \* @author 李鹏军  
 \*/  
@Slf4j  
@Component("orderTask")  
public class OrderTask {  
 private final MallOrderService orderService;  
 private final MallGoodsService goodsService;  
  
 @Autowired  
 public OrderTask(MallGoodsService goodsService, MallOrderService orderService) {  
 this.goodsService = goodsService;  
 this.orderService = orderService;  
 }  
  
 /\*\*  
 \* 过期自动取消订单  
 \*/  
 @SuppressWarnings(value = "unused")  
 @Transactional(rollbackFor = Exception.class)  
 public void expireOrder() {  
 ***log***.info("--------------------------开始执行过期自动取消订单任务--------------------------");  
  
 //等待付款的订单  
 List<MallOrderEntity> orderList = orderService.lambdaQuery().eq(MallOrderEntity::getOrderStatus, Constant.OrderStatus.***DFK***.getValue()).list();  
 if (null != orderList && orderList.size() > 0) {  
 ***log***.info("待付款订单数：" + orderList.size());  
  
 List<MallOrderEntity> list = new ArrayList<>();  
 orderList.forEach(order -> {  
 //当前时间大于过期时间  
 if (new Date().after(order.getExpireTime())) {  
 order.setOrderStatus(Constant.OrderStatus.***YQX***.getValue());  
 list.add(order);  
  
 //取消的订单释放商品库存  
 goodsService.backGoodsNumber(order);  
 }  
 });  
 if (list.size() > 0) {  
 orderService.updateBatchById(list);  
 }  
 }  
 ***log***.info("--------------------------结束过期自动取消订单任务--------------------------");  
 }  
  
}



# 前端源码分析

前端使用了vue+elementUI实现

## 技术栈

你需要在本地安装 nodejs。

* [nodejs](http://nodejs.org/)
* [ES6](http://es6.ruanyifeng.com/)
* [vue-cli](https://github.com/vuejs/vue-cli)
* [vue](https://cn.vuejs.org/index.html)
* [vue-router](https://github.com/vuejs/vue-router)
* [vuex](https://github.com/vuejs/vuex)
* [axios](https://github.com/axios/axios)
* [vue-cookie](https://github.com/alfhen/vue-cookie)
* [element-ui](https://github.com/ElemeFE/element)
* [iconfont](http://www.iconfont.cn/)

## 项目结构

├── build // 构建相关

├── config // 构建配置相关

├── src // 源代码

│ ├── assets // 静态资源

│ ├── components // 全局公用组件

│ ├── element-ui // element-ui组件配置

│ ├── element-ui-theme // element-ui组件主题配置

│ ├── icons // 所有 svg icons

│ ├── router // 路由

│ ├── store // 全局 store管理

│ ├── utils // 全局公用方法

│ ├── views // view

│ ├── App.vue // 入口组件

│ ├── main.js // 入口

├── static // 第三方不打包资源

│ ├── config // 全局变量配置

│ ├── img // favicon图标

│ ├── plugins // 插件

├── .babelrc // babel-loader 配置

├── eslintrc.js // eslint 配置项

├── .gitignore // git 忽略项

├── index.html // html模板

└── package.json // package.json

## 主题定制

提供13套颜色主题，进行element-ui和整站主题切换。具体切换方法如下：

1. 修改[/src/element-ui-theme/index.js](https://github.com/daxiongYang/renren-fast-vue/blob/master/src/element-ui-theme/index.js)文件中import './element-[#17b3a3]/index.css'[]中括号中的值，值可以在同文件中list属性中取即可。**（注意：这里只是修改element-ui组件主题）**
2. 修改[/src/assets/scss/\_variables.scss](https://github.com/daxiongYang/renren-fast-vue/blob/master/src/assets/scss/_variables.scss)文件中$--color-primary: [#17b3a3];[]中括号中的值，值与第一步值同步即可。**（注意：这里只是修改站点主题，不包括element-ui组件主题）**

主题定制具体实现方法是：

1. 先通过element-ui官方提供的[在线主题生成工具](https://elementui.github.io/theme-chalk-preview/#/zh-CN)，进行切换主题色，再下载解压文件（保留fonts目录中文件和index.css即可）放置/src/element-ui-theme/目录中，使用同目录中的index.js进行统一配置管理。
2. 再设置修改站点主题，使整站主题色统一一致。

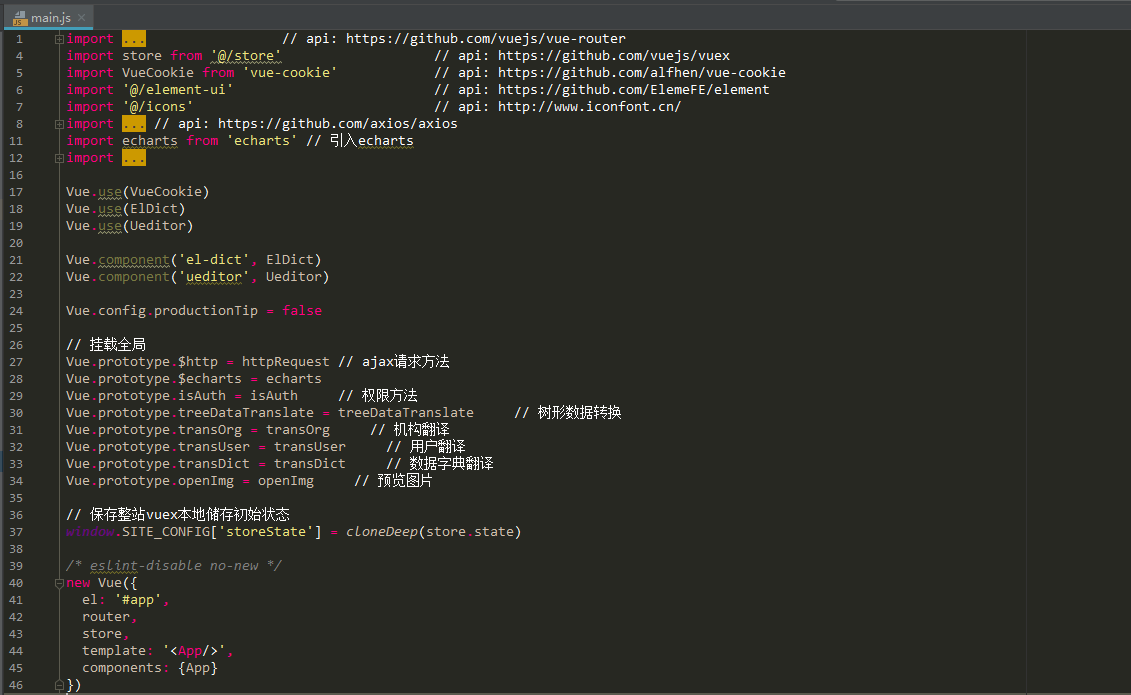
## 菜单图标

<https://www.iconfont.cn/search/index>

在阿里巴巴矢量图标库下载svg格式图标，以icon-开头命名，然后复制到platform-admin-ui\src\icons\svg下即可在修改菜单时选择

## 挂载全局的公共方法

具体实现与使用请参照源码



## 按钮权限控制

使用全局方法isAuth()，案例：

<el-button v-if="*isAuth*('sys:config:save')" type="primary" @click="addOrUpdateHandle()">新增</el-button>

## 页面组件

更多页面开发请参照elementUI官方提供文档

<https://element.eleme.cn/#/zh-CN/component/installation>

# 物流查询

登录微信公众平台

左侧菜单 ->设置 -> 第三方设置

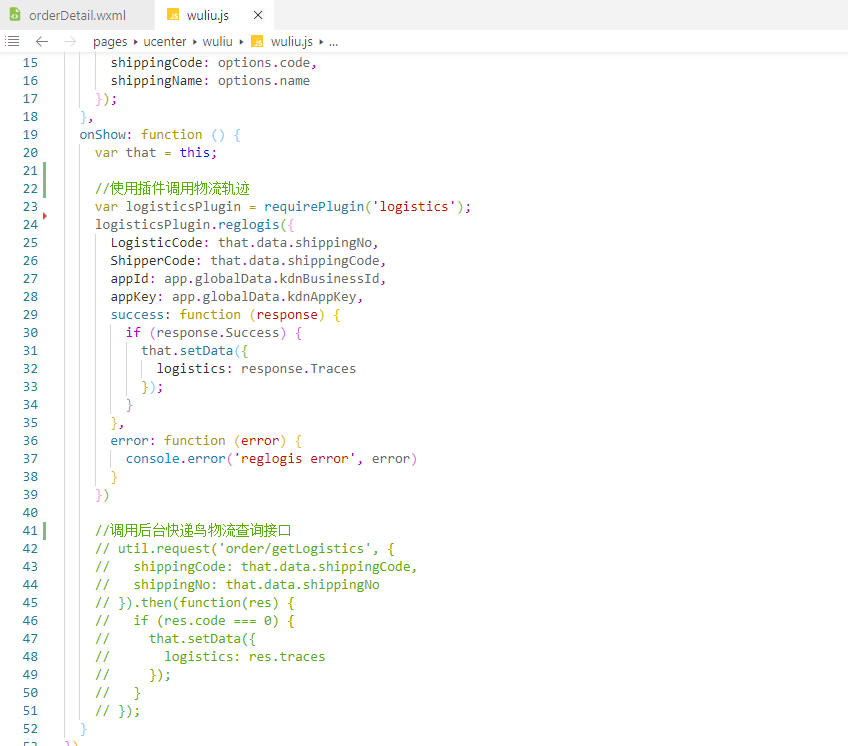
添加插件《物流轨迹》



注意有两类快递查询方式



1. 使用快递鸟接口，需要自己申请账户，有两种调用方式。
   1. 使用插件（配置app.js里globalData. kdnBusinessId、kdnAppKey）
   2. 调用后台接口（配置application.yml里businessId、appKey）



1. 直接跳转快递100小程序查询物流轨迹（无需任何配置）

# 微信公众号配置

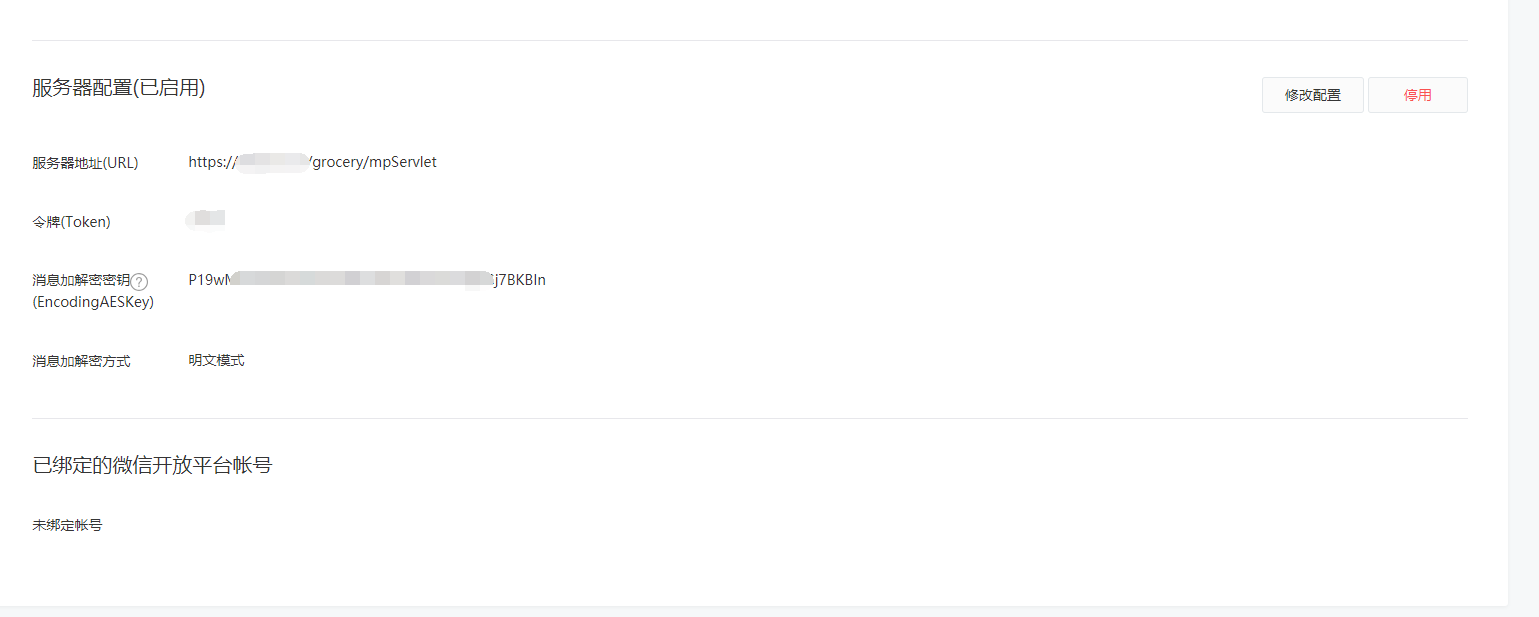
目前已经实现的功能

* 支持WxJava所有功能
* 生成公众号菜单
* 接收普通消息、事件推送

1. 填写服务器配置

登录微信公众平台官网后，在公众平台官网的开发-基本设置页面，勾选协议成为开发者，点击“修改配置”按钮，填写服务器地址（URL）、Token和EncodingAESKey，其中URL是开发者用来接收微信消息和事件的接口URL。Token可由开发者可以任意填写，用作生成签名（该Token会和接口URL中包含的Token进行比对，从而验证安全性）。EncodingAESKey由开发者手动填写或随机生成，将用作消息体加解密密钥。

另请注意，微信公众号接口必须以http://或https://

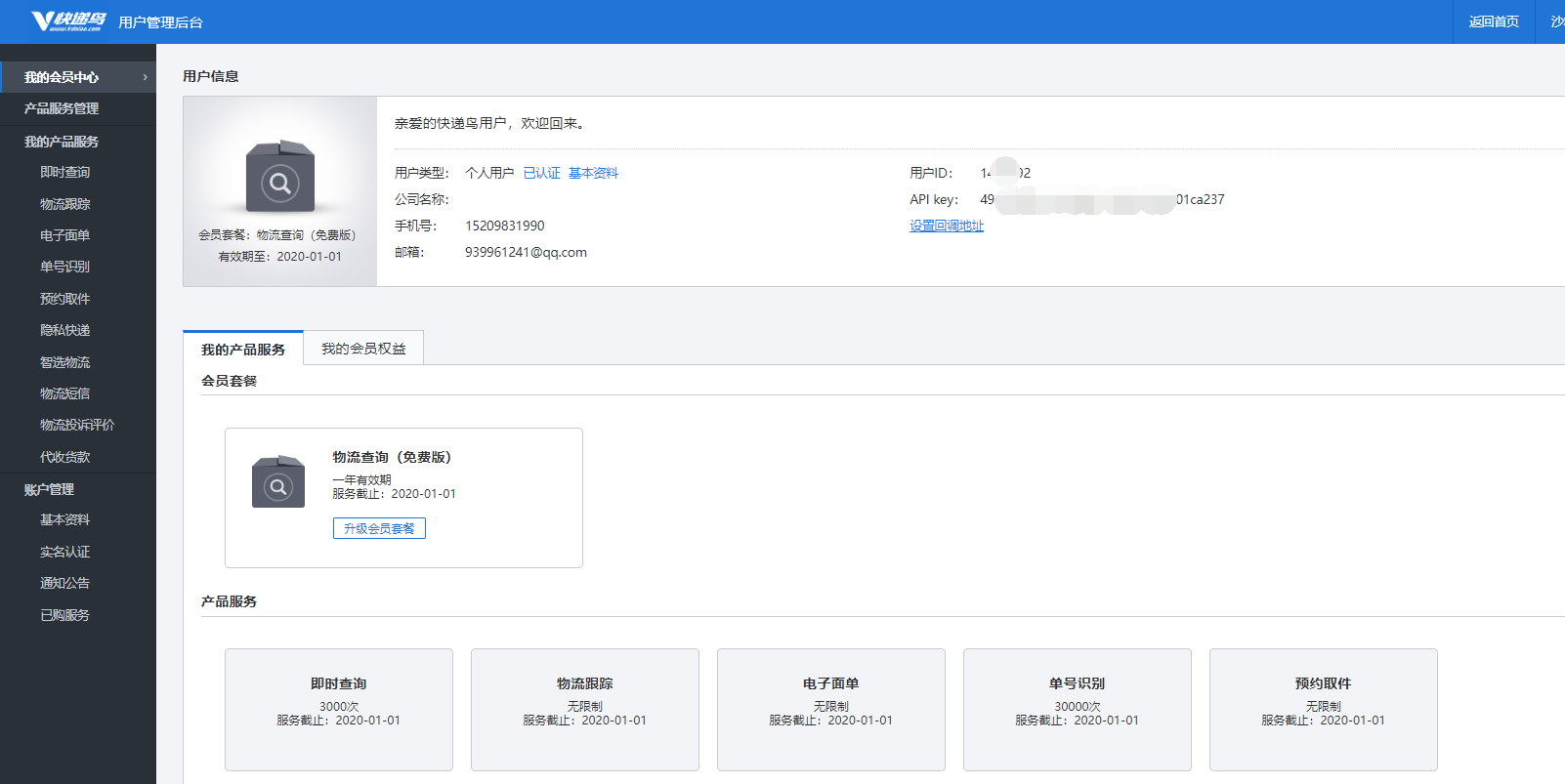


1. 服务器的IP添加到白名单
2. 将公众号appId、secret、token、aesKey保存到wx\_mp\_config表
3. 启用服务器配置后，用户关注后，信息会存入系统中

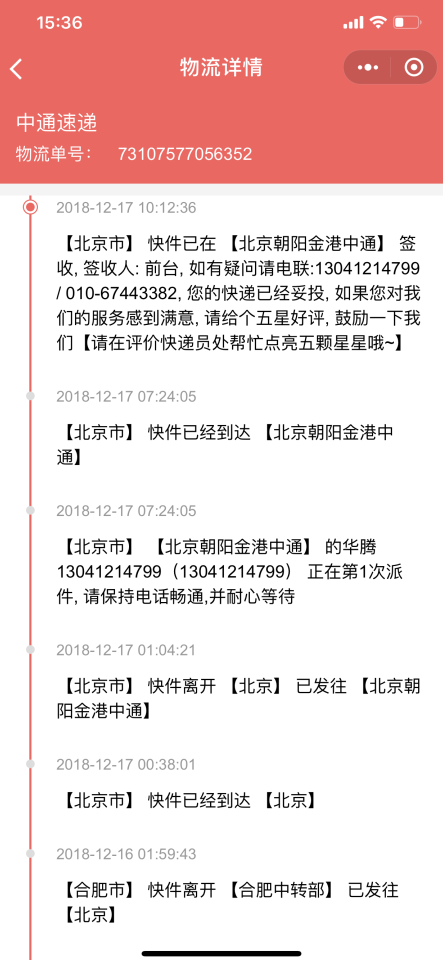


# 物流相关

申请快递鸟账户：<http://www.kdniao.com/reg>



修改wx-mall-> app.js里的值kdnBusinessId、kdnAppKey。



# 项目所需材料

## 微信注册认证

因为微信每次认证都需要300元，所以建议先申请公众号。公众号认证成功后，再申请微信小程序。申请小程序后复用公众号的认证资质。

微信申请步骤：<https://kf.qq.com/faq/120911VrYVrA151013MfYvYV.html>

## 微信支付所需材料

* 企业：

1、营业执照：彩色扫描件或数码照片

2、组织机构代码证：彩色扫描件或数码照片，若已三证合一，则无需提供

3、对公银行账户：包含开户行省市信息，开户账号

4、法人身份证：彩色扫描件或数码照片

* 个体工商户

1、营业执照：彩色扫描件或数码照片

2、对公银行账户/法人对私账户：包含开户行省市信息，开户账号

3、法人身份证：彩色扫描件或数码照片

* 政府及事业单位、民办非企业、社会团体、基金会
* 1、营业执照：彩色扫描件或数码照片
* 2、组织机构代码证：彩色扫描件或数码照片，若已三证合一，则无需提供
* 3、对公银行账户：包含开户行省市信息，开户账号
* 4、法人身份证：彩色扫描件或数码照片

## 服务器

Linux或windows服务器，安装软件：jdk1.8、redis、nginx

Mysql数据库

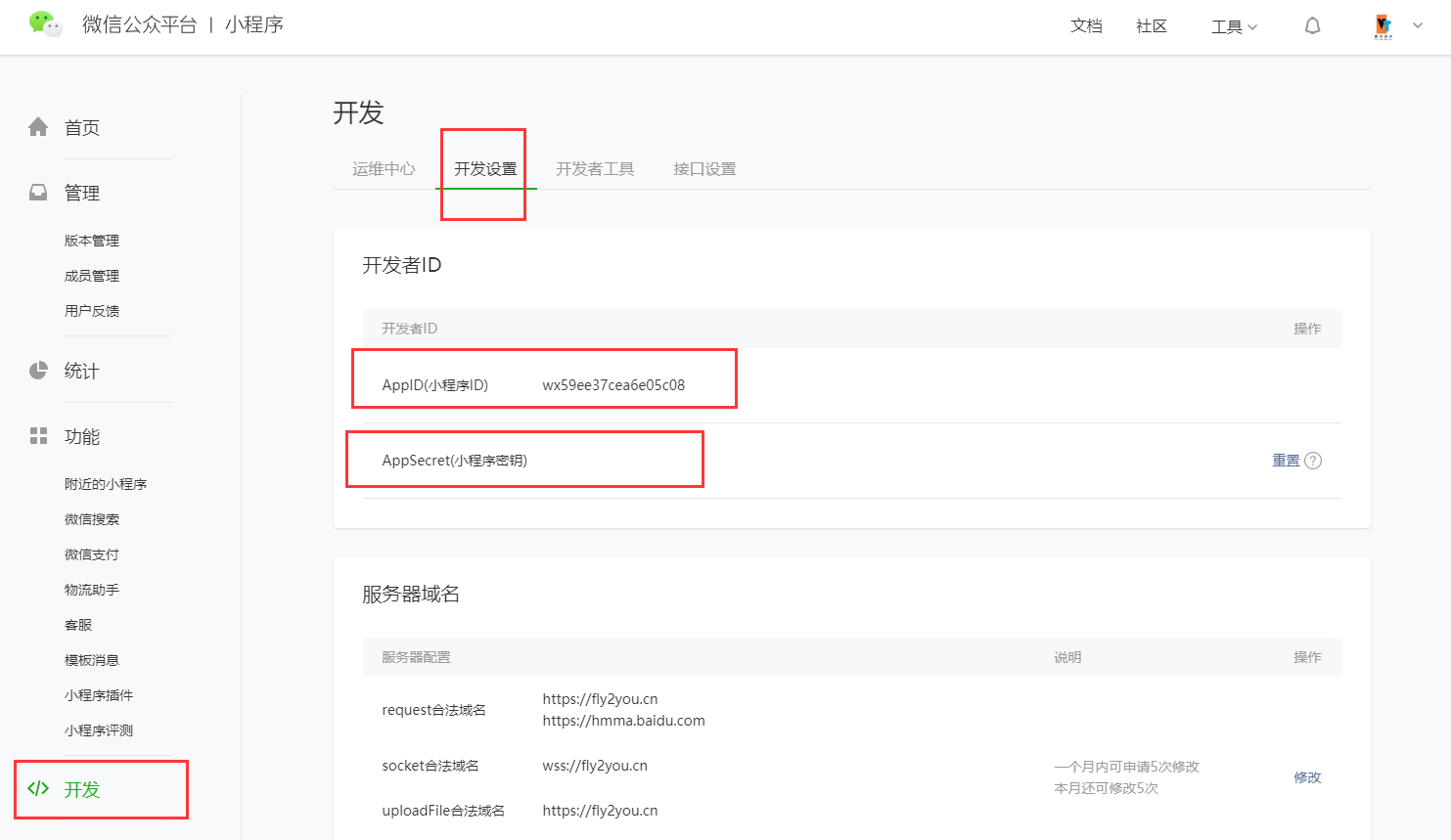
## 其他材料

申请域名、域名备案、SSL证书

# 系统所需配置参数

## wx.appId、wx.secret

小程序ID、小程序密钥



## wx.pay.mchId、wx.pay.mchkey

商户号、支付签名

微信商户平台(pay.weixin.qq.com)-->账户设置-->API安全-->密钥设置

## wx.pay.basenotifyurl

支付回调地址，换成线上的域名

## wx.spbillcreateip

服务器所在的IP

## 设置退款回调地址

接口位置在AppPayController类parseRefundNotifyResult方法



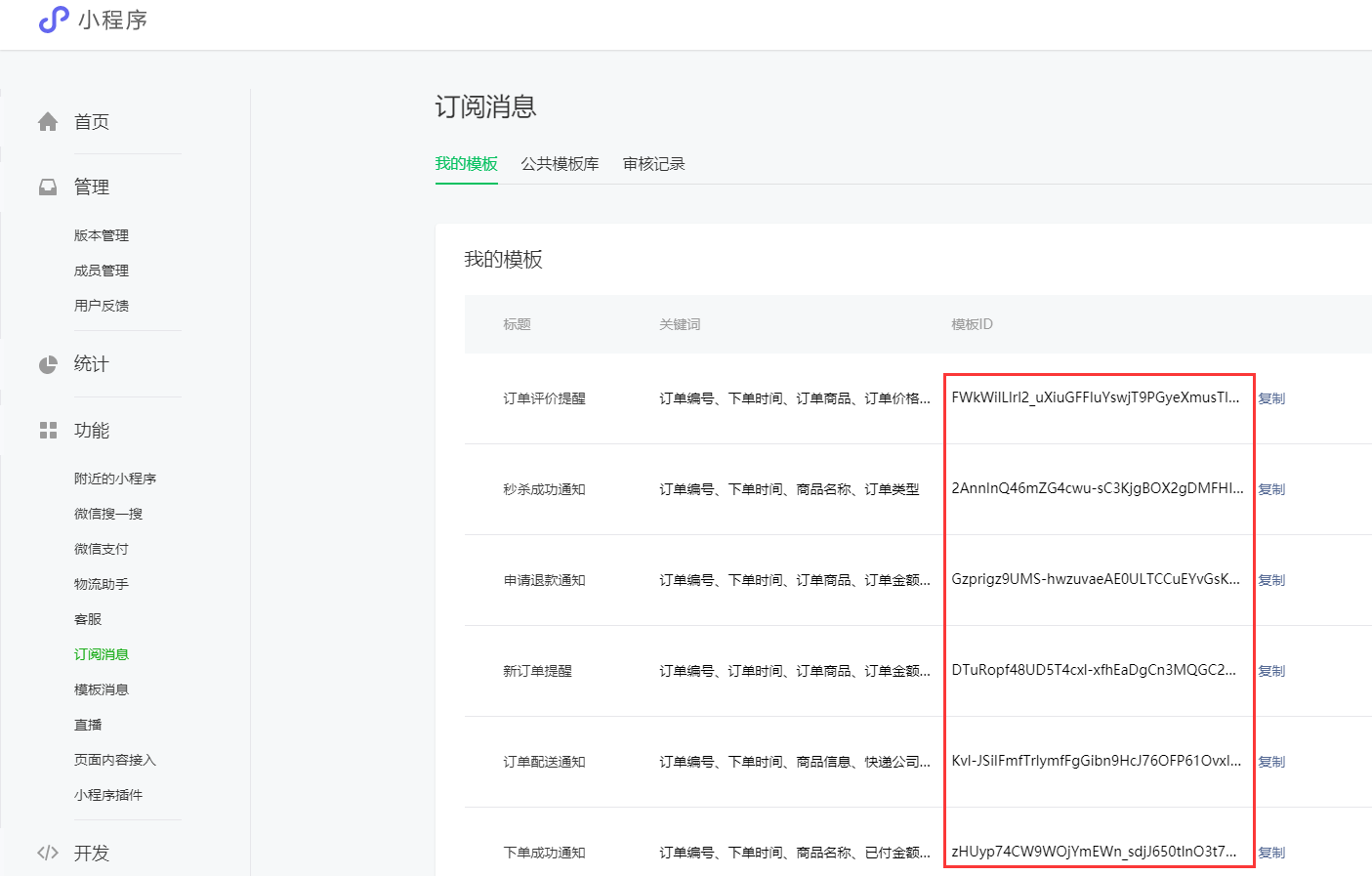
## 业务域名

小程序上线一定要设置业务域名和服务器域名，域名要备案。

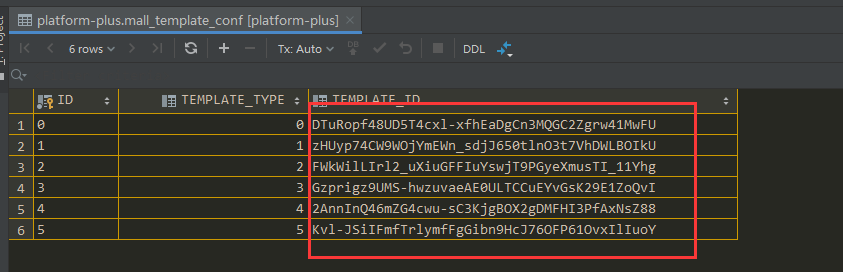


## 订阅消息

登录微信公众平台添加模板

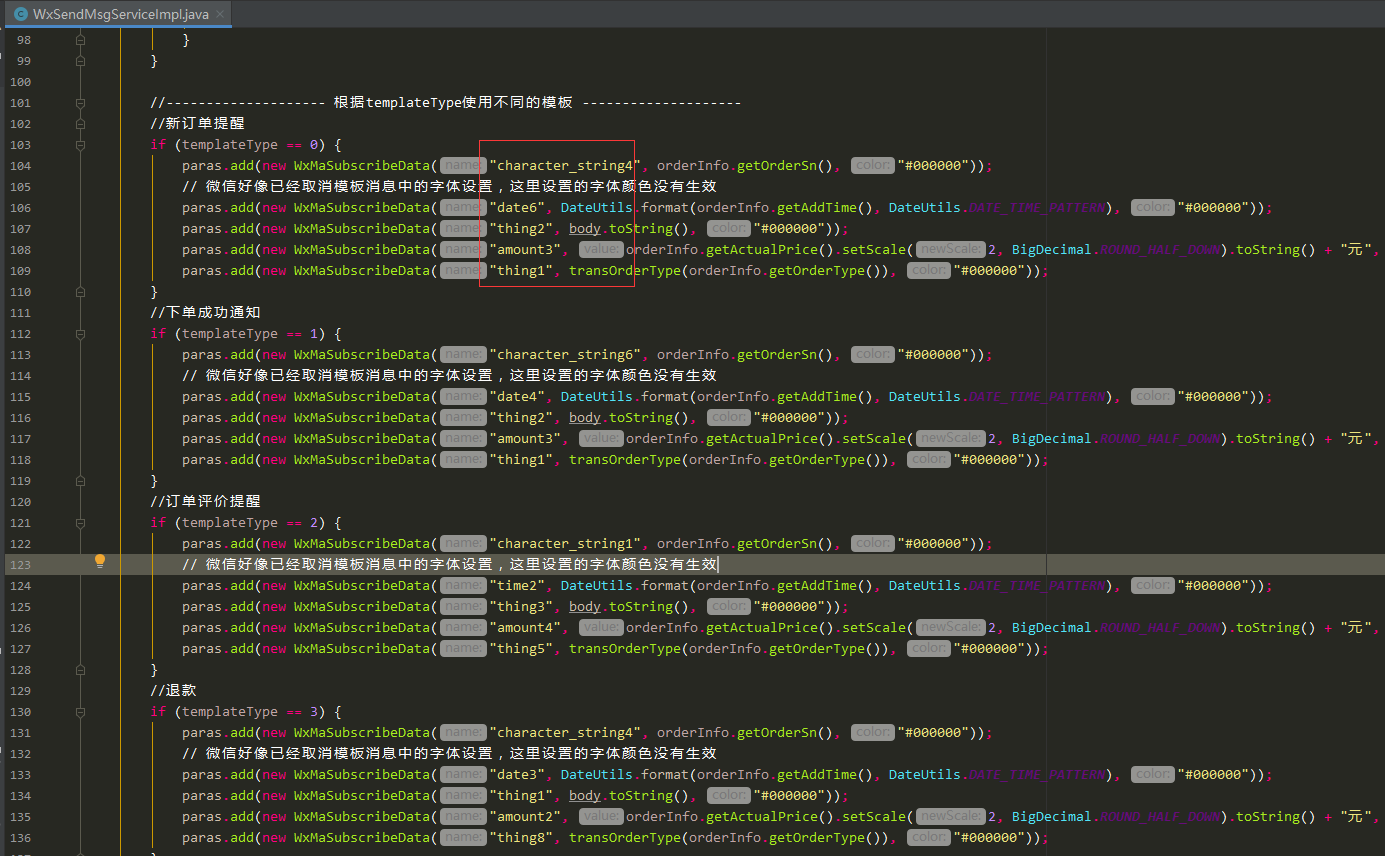


添加后将模板ID替换到数据库中



这里需要注意的是，下面红框的值一致





## 飞鹅打印机配置

### 购买



### 注册

<http://admin.feieyun.com/regist.php>

### 实名认证

实名认证之后查看USER、UKEY，如下图



将这两个参数配置到PrintUtils.java文件中

然后配置店铺打票机（每个店铺只配置一个打印机！每次店铺有新订单会打印两份，一份客户、一份存根，所以要配置存根打印机，这里每个店铺的存根打印机可以设置为同一个由总管理员维护。如果没有设置存根打印机，两份会都由店铺打印机打出来。）



打印机编号可以在每个打票机底部查看

### 打票机上线

支持GPRS、WIFI、蓝牙、USB

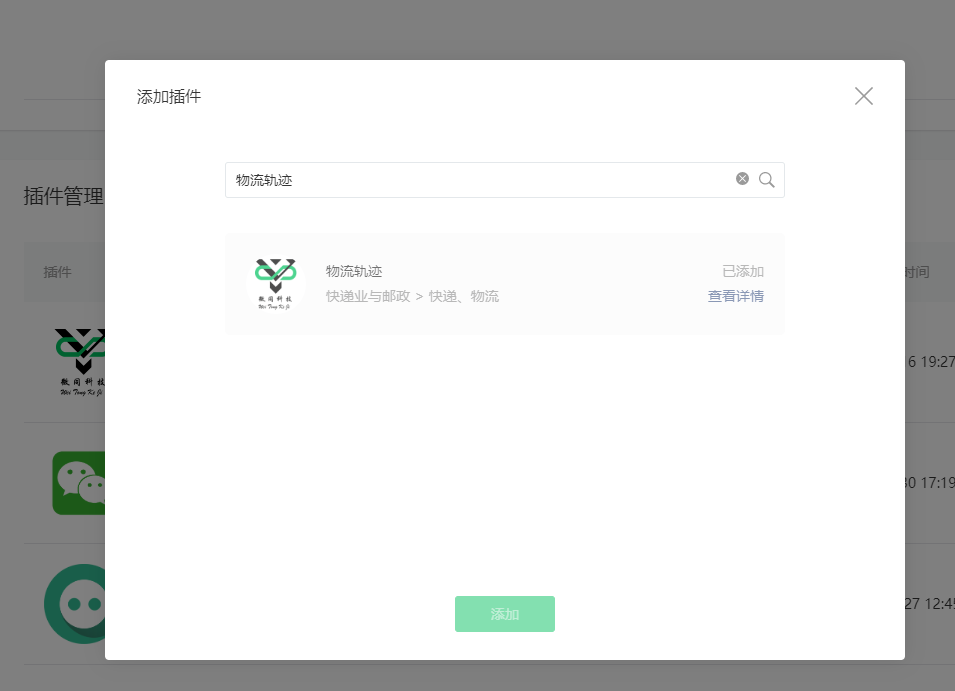
每个型号的说明书下载：<http://feieyun.com/open/index.html?name=2>

# 添加插件

## “物流轨迹”插件

登录微信公众平台小程序账号，在菜单：设置->第三方设置->插件管理

添加“物流轨迹”插件



申请之后联系客服审核通过。

## “直播”插件

**商户准入要求**

满足以下条件的电商平台、自营商家，即有机会被邀请到小程序直播公测中来：

（同时满足以下1、2、3条件，加上4、5、6条件的其中之一即可。）

**1. 满足小程序18个开放类目**（包括：电商平台、商家自营-百货、食品、初级食用农产品、酒/盐、图书报刊/音像/影视/游戏/动漫、汽车/其他交通工具的配件、服装/鞋/箱包、玩具/母婴用品(不含食品)、家电/数码/手机、美妆/洗护、珠宝/饰品/眼镜/钟表、运动/户外/乐器、鲜花/园艺/工艺品、家居/家饰/家纺、汽车内饰/外饰、办公/文具、机械/电子器件）

**2. 主体下小程序近半年没有严重违规**

**3. 小程序近90天存在支付行为**

**4. 主体下公众号累计粉丝数大于100**

**5. 主体下小程序近7日dau大于100**

**6. 主体在微信生态内近一年广告投放实际消耗金额大于1w**

**服务商接入指引**

具体接入指引请参考[《【小程序直播】服务商接入指引》](https://docs.qq.com/doc/DZHJ6Z0hNWWRUVk94)，以下为服务商接入步骤。

**1. 权限申请**

1） 在问卷[《服务商“小程序直播”接入申请》](https://wj.qq.com/s2/5542558/8d04/)填写相关信息并等待权限开通，发送申请后7个工作日内，可登陆微信开放平台查看第三方平台权限集并勾选 “小程序直播” 能力；

2） 开通后，即可登陆 “[微信开放平台](https://open.weixin.qq.com/)” （open.weixin.qq.com）勾选**“小程序直播” 第三方权限集并全网发布**；

**2.功能开发**

小程序直播需要实现【直播组件】与【后台API】两个部分，其中组件部分需要在小程序中进行配置开发。

具体开发文档，请参考[《小程序直播组件接入指引》](https://docs.qq.com/doc/DZHhzV0FiYXRQV01i)。