目录

[1 项目简介 3](#_Toc452613069)

[2 安装教程 3](#_Toc680773009)

[2.1 本地开发环境启动步骤 3](#_Toc2106574694)

[2.2 生产环境部署 3](#_Toc1785477616)

[2.2.1 后台部署 3](#_Toc1733292581)

[2.2.2 前端部署 4](#_Toc832737312)

[3 上传图片 5](#_Toc665075285)

[3.1 阿里云配置 5](#_Toc267932360)

[3.2 腾讯云配置 7](#_Toc2013450408)

[3.3 七牛云配置 9](#_Toc13697830)

[3.4 服务器配置 9](#_Toc438678581)

[3.5 华为云配置 10](#_Toc559550716)

[4 短信平台 12](#_Toc537993599)

[4.1 腾讯云短信平台 12](#_Toc1152264523)

[4.1.1 登录腾讯云平台 12](#_Toc102309415)

[4.1.2 创建短信签名 12](#_Toc1527420305)

[4.1.3 创建短信内容 12](#_Toc333549897)

[4.2 阿里云短信平台 13](#_Toc1040800209)

[4.2.1 登录阿里云平台 13](#_Toc1474807848)

[4.2.2 添加签名 13](#_Toc839247662)

[4.2.3 添加模板 13](#_Toc562861738)

[5 移动端接口文档 13](#_Toc351765531)

[5.1 请求路径 13](#_Toc100799326)

[6 后台源码分析 15](#_Toc1917158246)

[6.1 技术架构图 15](#_Toc834000934)

[6.2 项目介绍 16](#_Toc427933769)

[6.3 代码生成工具的使用 16](#_Toc360121780)

[7 后端源码分析 17](#_Toc957839214)

[7.1 platform-admin模块 17](#_Toc866251786)

[7.1.1 操作日志（切面编程） 17](#_Toc1302124289)

[7.1.2 当前登录用户 18](#_Toc1944562293)

[7.1.3 数据权限实现（mybatis拦截器实现） 19](#_Toc1852318405)

[7.2 platform-api模块 21](#_Toc1992485923)

[7.2.1 当前登录会员 21](#_Toc1998400190)

[7.2.2 不需要token验证的接口 22](#_Toc467754250)

[7.3 统一异常处理 24](#_Toc1755531730)

[7.3.1 后台异常处理 24](#_Toc943959977)

[7.3.2 前端统一异常处理 26](#_Toc1673633050)

[7.4 限流 26](#_Toc1009862944)

[7.5 同步锁 27](#_Toc1203237567)

[7.6 定时任务 28](#_Toc2107768417)

[7.7 多数据源 30](#_Toc373543607)

[8 前端源码分析 30](#_Toc1052702668)

[8.1 技术栈 30](#_Toc1803457090)

[8.2 项目结构 31](#_Toc1119117872)

[8.3 主题定制 32](#_Toc1352294278)

[8.4 菜单图标 32](#_Toc1190494145)

[8.5 挂载全局的公共方法 32](#_Toc529955916)

[8.6 按钮权限控制 33](#_Toc1354396103)

[8.7 页面组件 33](#_Toc8644921)

[9 微信公众号配置 33](#_Toc1413782898)

[10 服务器 34](#_Toc1690096278)

[10.1 要求 34](#_Toc681945477)

[10.2 申请云服务器和云数据库 34](#_Toc337407900)

# 项目简介

本项目为公众号基础版

# 安装教程

## 本地开发环境启动步骤

\* 创建数据库 platform-mp

\* 字符集 : utf8mb4 -- UTF-8 Unicode

\* 排序规则 : utf8mb4\_general\_ci

\* 导入\_sql/platform-mall-MySQL.sql

\* 推荐环境（nginx、jdk1.8、mysql8.0、redis ）

\* 导入项目到IDE中

\* 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

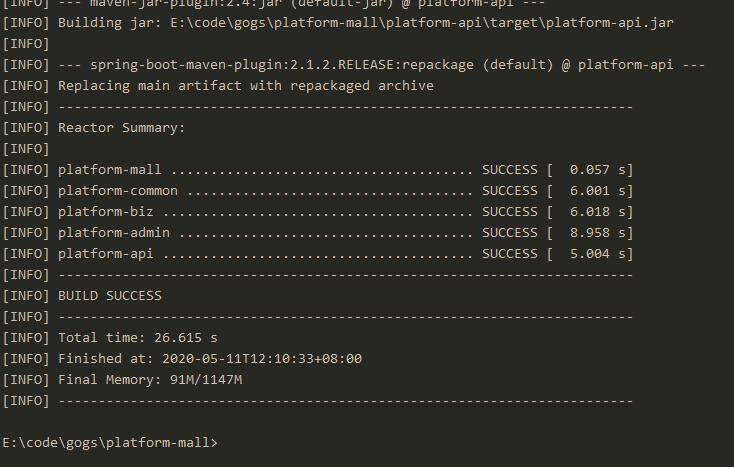
\* 修改开发环境配置/platform-admin/src/main/resources/application-dev.yml

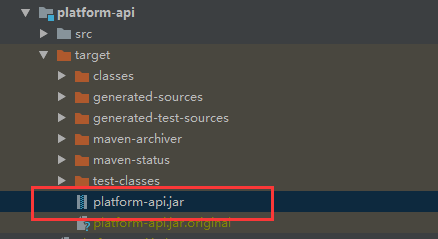
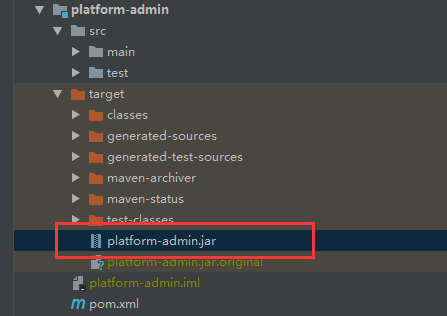
## 生产环境部署

### 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（readme.md文件有详细内容）

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。

需要注意，小程序访问接口必须要SSL证书，使用https

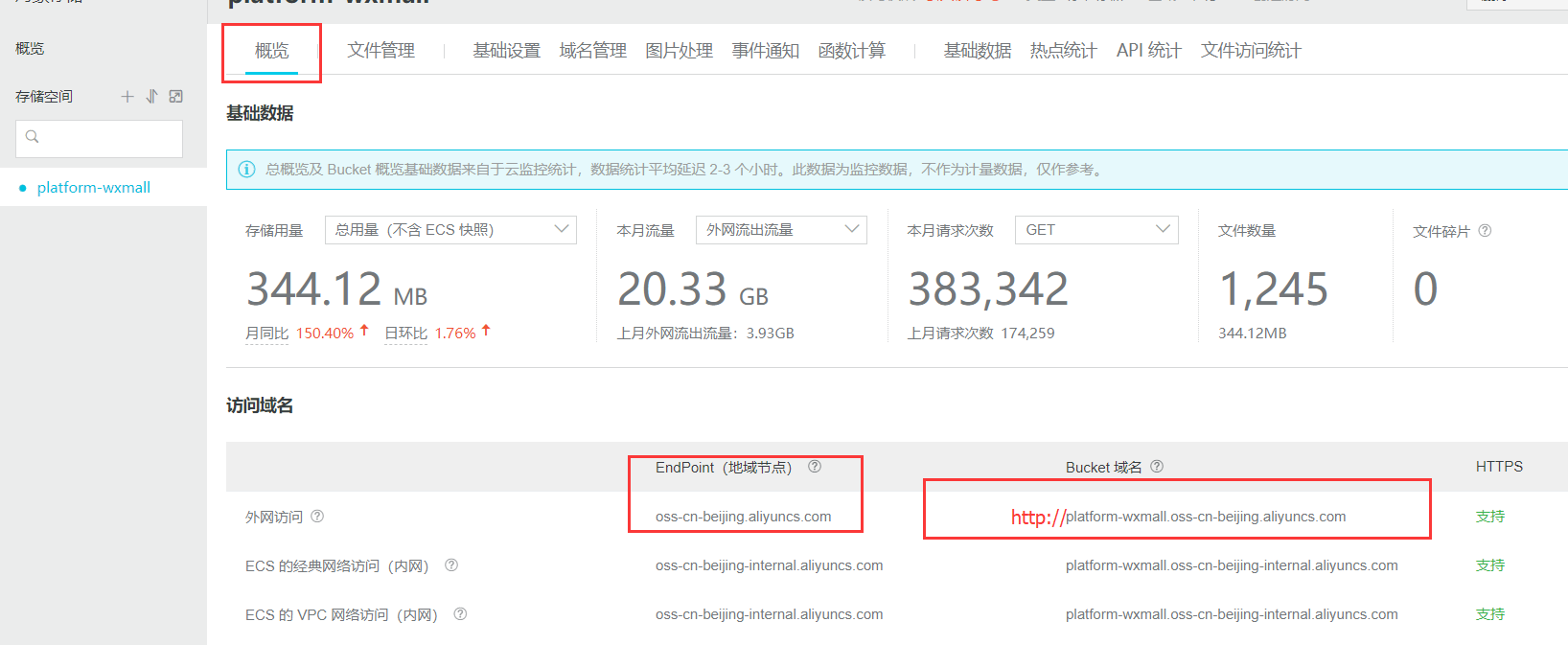
# 上传图片

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：

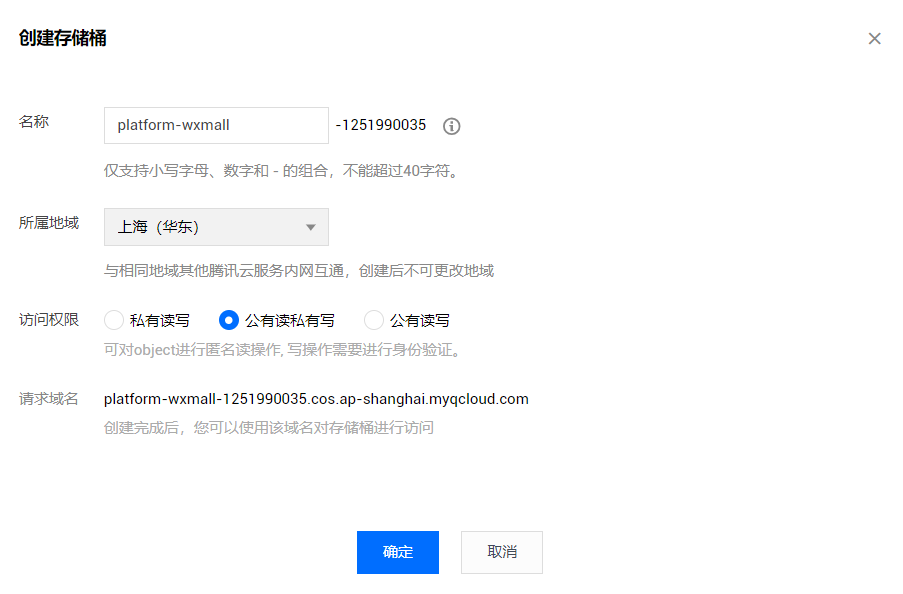
AccessKeyId、AccessKeySecret：





## 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

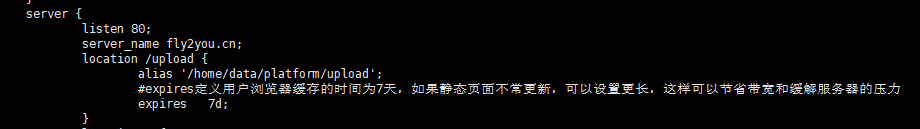


存储路径：服务器存放图片的路径。egg：/home/data/platform/upload

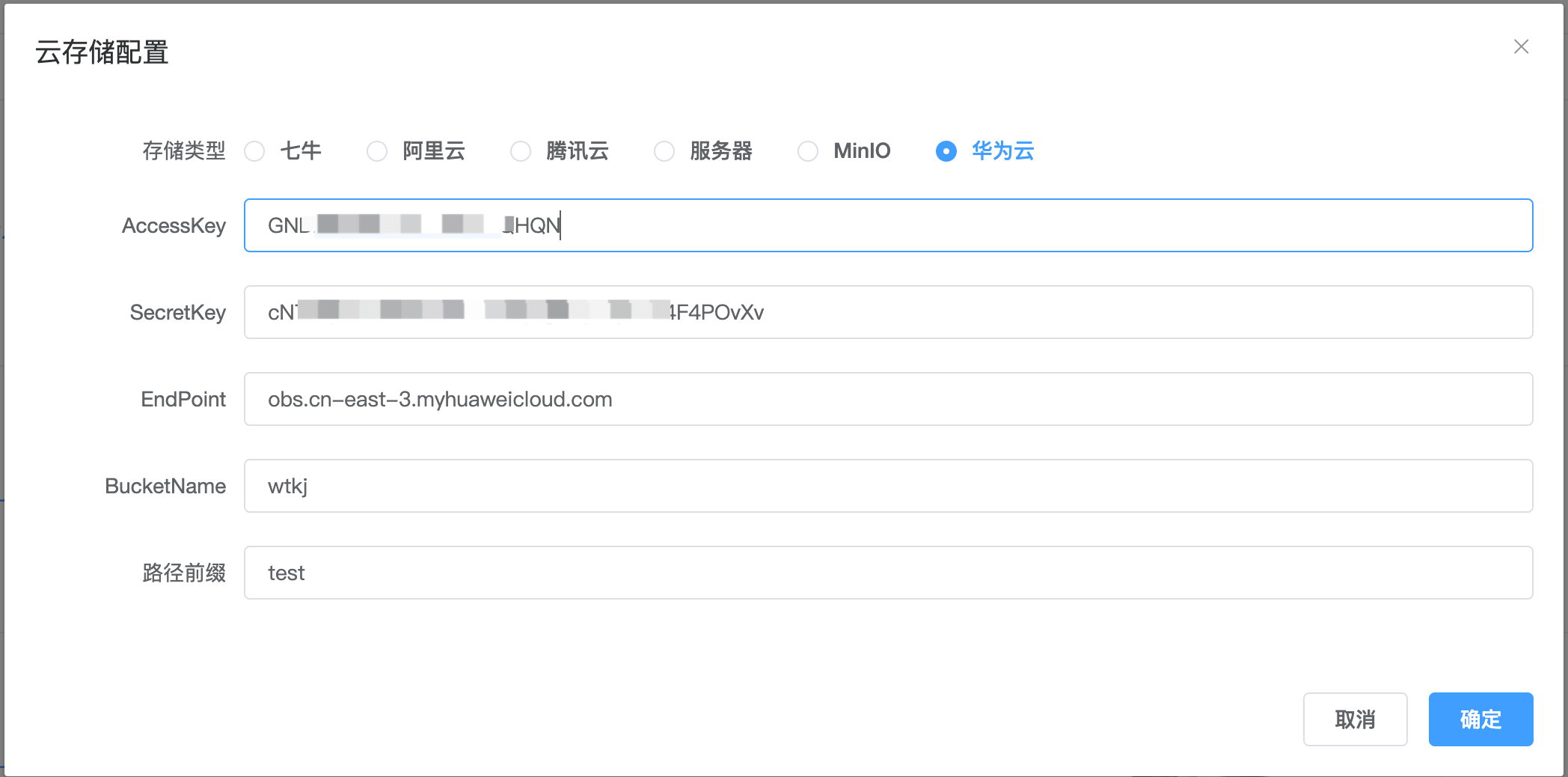
代理服务器：该配置是代理服务器的地址。egg：<http://fly2you.cn/upload>

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

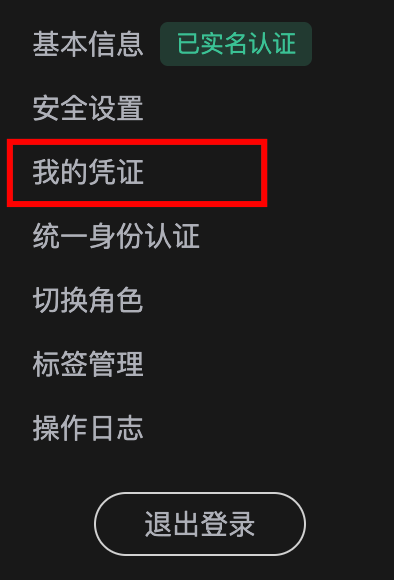
该例使用nginx代理，下面贴上nginx配置：

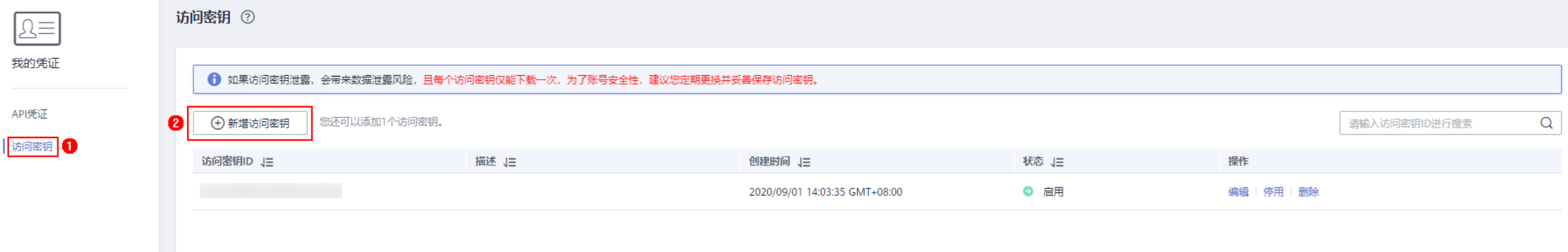


## 华为云配置

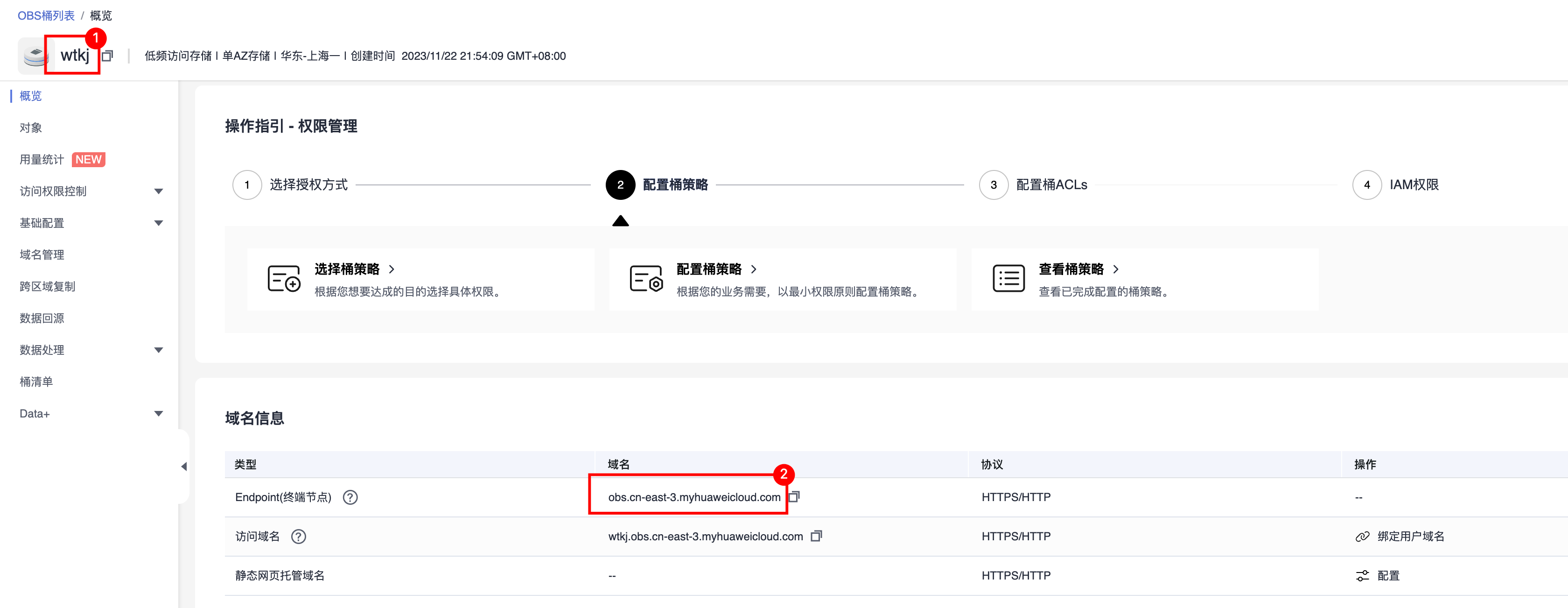


1.登录[华为云管理控制台](https://auth.huaweicloud.com/authui/login.html?service=https://console.huaweicloud.com/console/?locale=zh-cn" \l "/login" \t "/Users/pengjun/Documents\\x/_blank)，鼠标指向页面右上角的用户名，在下拉列表中单击“我的凭证”。



2.在“我的凭证”页面中选择“访问密钥”页签。单击“新增访问密钥”，按操作指引获取认证帐号的AK/SK，请妥善保管AK/SK信息。

3.下图1为BucketName，2为EndPoint。



1. 路径前缀为自定义文件存储文件夹，这里设置为test，会在OBS对象根目录创建test目录。



# 短信平台

本系统已集成腾讯云短信、阿里云短信平台功能

## 腾讯云短信平台

### 登录腾讯云平台

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

在应用管理菜单创建应用，然后点击应用名称，查看AppID和AppKey



### 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和上述中的AppID和AppKey配置到后台管理系统（菜单：短信平台->短信配置）

## 阿里云短信平台

### 登录阿里云平台

<https://dysms.console.aliyun.com/overview>

### 添加签名

一般为公司简称

### 添加模板

模版类型为验证码，内容为您的验证码为 ${code} ，该验证码5分钟内有效，请勿泄露于他人。

最后将模板CODE配置到后台管理系统（菜单：短信平台->短信配置）

# 移动端接口文档

## 请求路径

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

使用说明

先调用登录接口，记录返回的token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

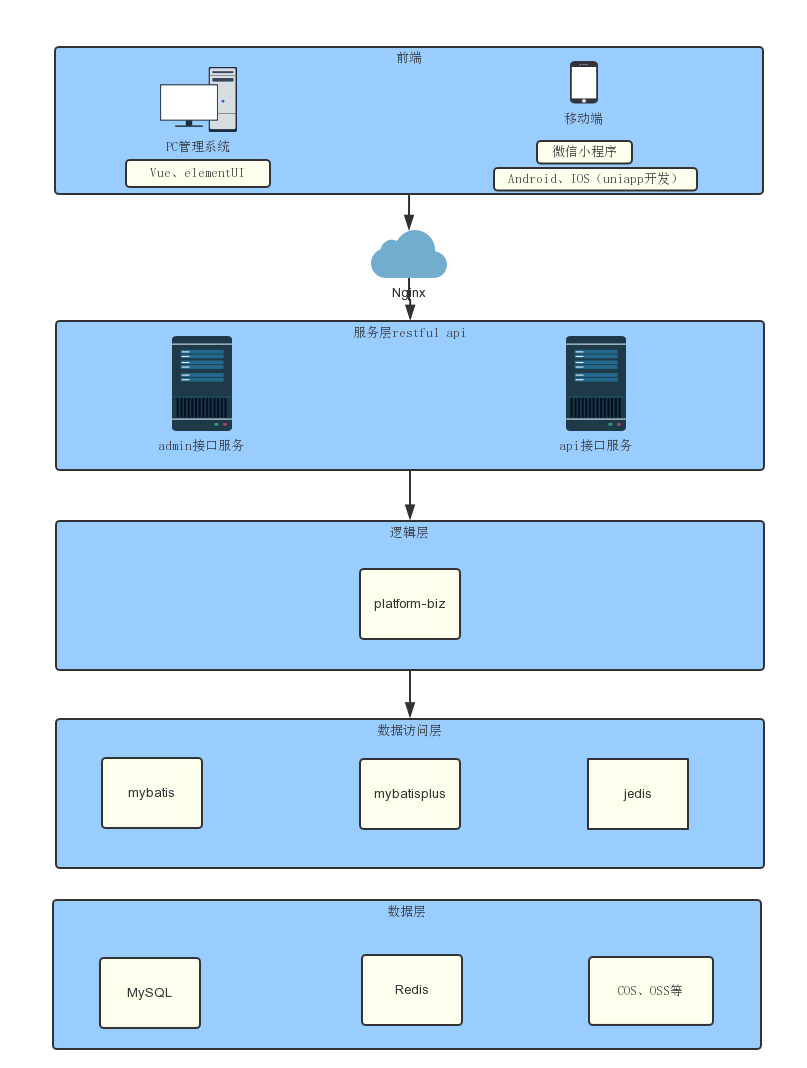


这样就可以调用需要用户登录的接口，不需要每次发起接口请求都填写token值了。如下图所示



# 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

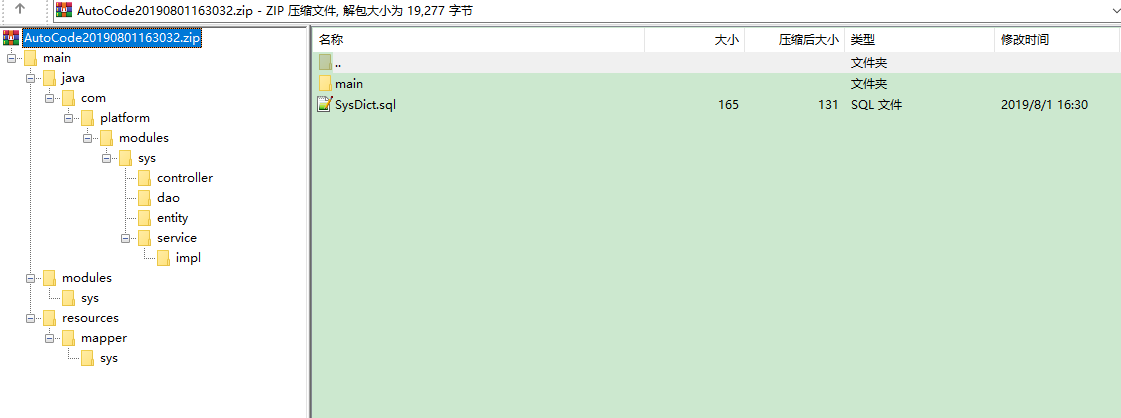
platform-biz：业务处理层（admin、api共用biz层）

platform-common：通用工具

## 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

/\*\*

\* 数据权限

\*

\* @author 李鹏军

\* @date 2021-05-18 13:30:56

\*/

@Data

@NoArgsConstructor

public class DataScopeInnerInterceptor implements InnerInterceptor {

@Override

public void beforeQuery(Executor executor, MappedStatement ms, Object parameter, RowBounds rowBounds, ResultHandler resultHandler, BoundSql boundSql) {

// 不是SELECT操作直接返回

if (!SqlCommandType.***SELECT***.equals(ms.getSqlCommandType())) {

return;

}

StringBuilder filterSql = new StringBuilder(boundSql.getSql());

//参数中DataScope类型的参数

DataScope dataScope = findDataScopeObject(parameter);

if (dataScope != null) {

SysUserEntity user = ShiroUtils.getUserEntity();

if (null != user) {

//系统管理员，拥有最高权限，不进行权限过滤

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("' ");

}

}

PluginUtils.MPBoundSql mpBoundSql = PluginUtils.mpBoundSql(boundSql);

List<ParameterMapping> mappings = mpBoundSql.parameterMappings();

mpBoundSql.sql(filterSql.toString());

mpBoundSql.parameterMappings(mappings);

}

}

}

/\*\*

\* 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;

}

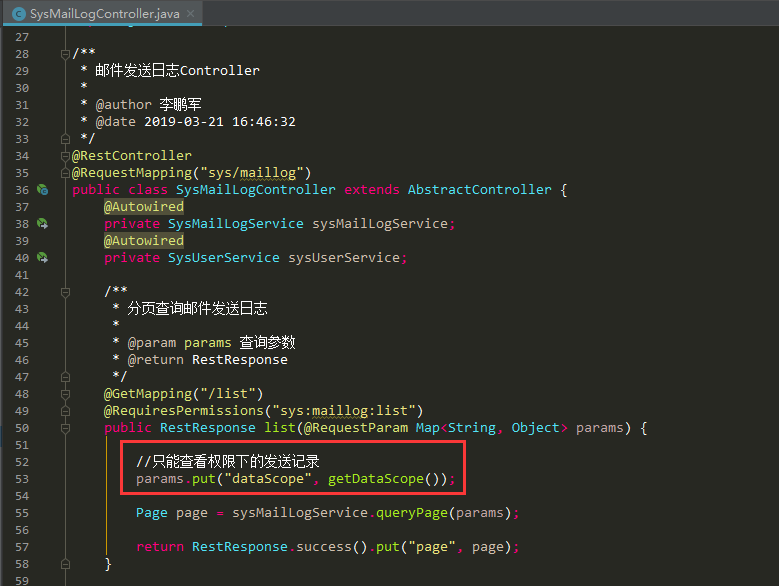
}

#### 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) {

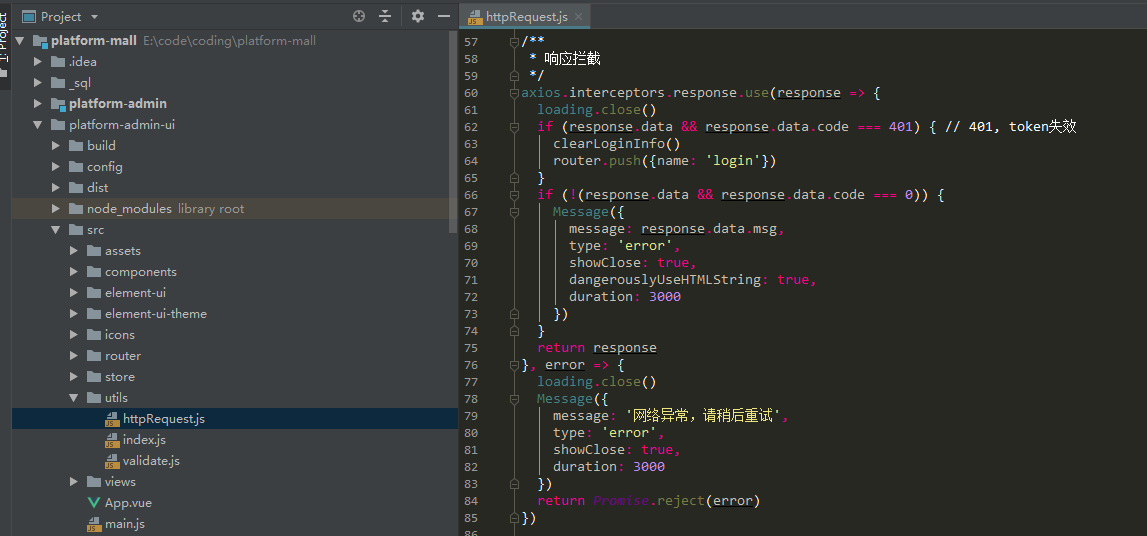
***log***.error(e.getMessage(), e);

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();

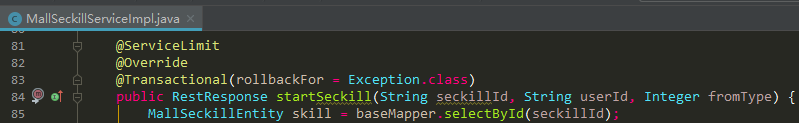
}

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--------------------------");

}

}



## 多数据源

使用第三方开源软件dynamic-datasource-spring-boot-starter，使用 @DS 切换数据源。

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

## 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" \t "_blank)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" \t "_blank)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/" \l "/zh-CN" \t "_blank)，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。

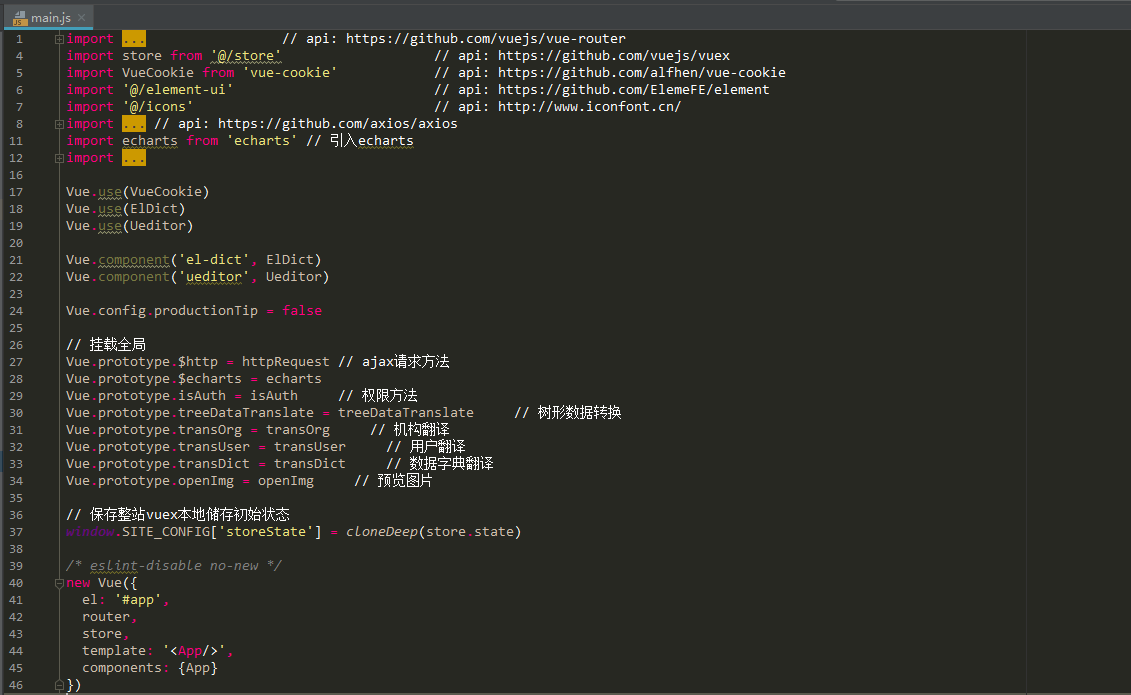
## 菜单图标

<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](https://element.eleme.cn/" \l "/zh-CN/component/installation)

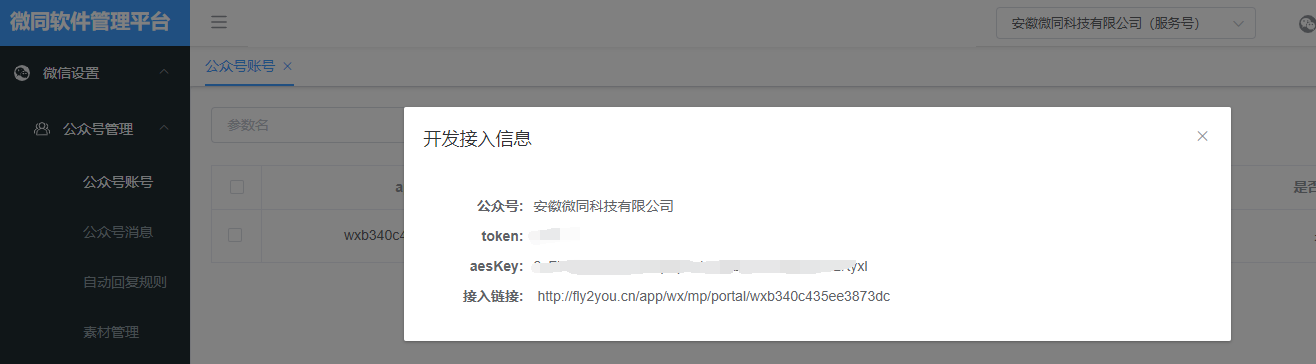
# 微信公众号配置

目前已经支持多公众号管理

* 自动回复规则
* 公众号菜单
* 接收普通消息、事件推送
* 素材管理

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。



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



# 服务器

## 要求

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

## 申请云服务器和云数据库

<https://cloud.tencent.com/act/cps/redirect?redirect=11447&cps_key=30280f92fc381dfc9e1d9e0e23d25a18>

云服务器推荐使用4核8G 5M带宽，标准型S2

MySQL高可用版1核2G 100GB SSD盘