# 天猫项目工程搭建与业务分析

v 1.0

2Executioner

## 项目架构分析

项目为b2c模式（后台管理员系统 + 前台用户系统），前后端分离，基于微服务架构

后端：

为了练习开发方便管理，把前台与后台融合在一起

前端：

后台管理员系统使用vue-admin-template模板

前台用户系统使用 nuxt.js + 自己设计的前端样式

## 使用技术

### 后端

|  |  |
| --- | --- |
| 开发框架 | springboot |
| 微服务框架 | spring cloud alibaba |
| 分布式文件管理 | fastDFS |
| 关系型数据库 | mysql |
| 缓存数据库 | redis |
| 安全框架 | （后续实现） |
| 分布式事务 | seata（预计下个版本使用） |
| 网关 | geteWay |
| 服务注册与配置中心 | nacos |
| 服务降级、限流 | sentinel |
| 搜索引擎 | elastic search（预计下个版本使用） |
| 存储层框架 | mybatis-plus |
| 消息中间件 | spring cloud stream（rabbitMQ） |
| 动静分离、服务器负载均衡 | nginx |
| 接口测试 | swagger |

### 前端

|  |  |
| --- | --- |
| 前端服务器 | node.js |
| 前端后台模板 | vue-admin-template |
| 前端前台框架 | nuxt |

## 后端项目工程搭建

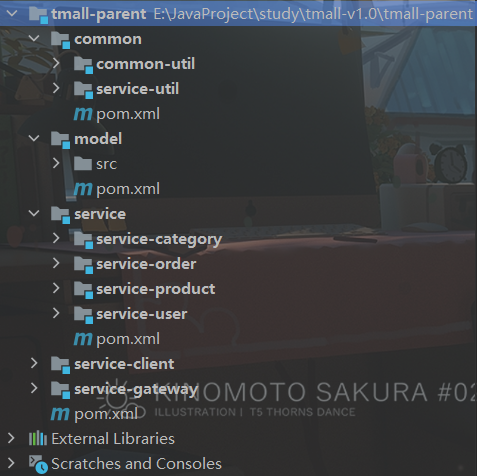
### 核心框架版本

spring boot：2.3.2.RELEASE

spring cloud alibaba：2.2.5.RELEASE

spring cloud：Hoxton.SR8

### 工程项目目录结构



* tmall-parent：maven父工程
* common：通用类父模块
* common-util：通用普通模块（exception、result、util等）
* service-util：通用服务模块（config、helper、util等）
* model：model类模块
* service：微服务父模块
* service-category：分类微服务
* service-order：订单微服务
* service-product：产品微服务
* service-user：用户微服务
* service-client：openFeign实现接口父模块
* service-gateway：gate way网关模块

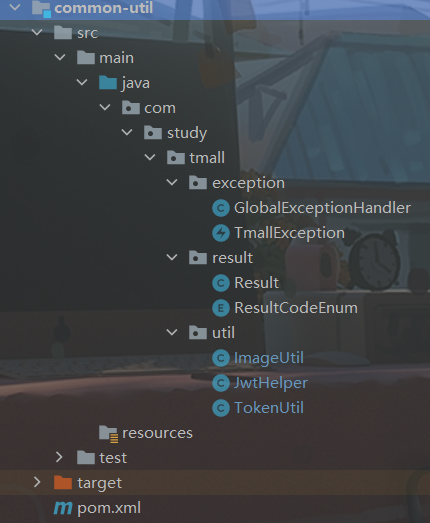
### 父工程依赖

<**modules**>  
 <**module**>common</**module**>  
 <**module**>model</**module**>  
 <**module**>service</**module**>  
 <**module**>service-client</**module**>  
 <**module**>service-gateway</**module**>  
</**modules**>  
  
<**parent**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-starter-parent</**artifactId**>  
 <**version**>2.3.2.RELEASE</**version**>  
</**parent**>  
  
<**groupId**>com.study.tmall</**groupId**>  
<**artifactId**>tmall-parent</**artifactId**>  
<**version**>1.0-SNAPSHOT</**version**>  
<**packaging**>pom</**packaging**>  
  
<!--统一管理jar包版本-->  
<**properties**>  
 <**project.build.sourceEncoding**>UTF-8</**project.build.sourceEncoding**>  
 <**java.version**>1.8</**java.version**>  
 <**maven.compiler.source**>1.8</**maven.compiler.source**>  
 <**maven.compiler.target**>1.8</**maven.compiler.target**>  
 <**junit.version**>4.12</**junit.version**>  
 <**log4j.version**>1.2.17</**log4j.version**>  
 <**lombok.version**>1.16.18</**lombok.version**>  
 <**druid.version**>1.1.16</**druid.version**>  
 <**project.version**>1.0-SNAPSHOT</**project.version**>  
 <**hutool.version**>5.6.2</**hutool.version**>  
 <**swagger.version**>2.7.0</**swagger.version**>  
 <**mybatis-plus.version**>3.4.2</**mybatis-plus.version**>  
 <**spring.cloud.version**>Hoxton.SR8</**spring.cloud.version**>  
 <**spring.cloud.alibaba.version**>2.2.5.RELEASE</**spring.cloud.alibaba.version**>  
 <**fastjson.version**>1.2.29</**fastjson.version**>  
 <**jwt.version**>0.7.0</**jwt.version**>  
 <**fastdfs.version**>1.27-SNAPSHOT</**fastdfs.version**>  
 <**easyexcel.version**>2.2.0-beta2</**easyexcel.version**>  
</**properties**>  
  
<!--子模块继承之后，提供作用：锁定版本+子模块不用写groupId和version-->  
<**dependencyManagement**>  
 <**dependencies**>  
 <!--spring cloud Hoxton.SR8-->  
 <**dependency**>  
 <**groupId**>org.springframework.cloud</**groupId**>  
 <**artifactId**>spring-cloud-dependencies</**artifactId**>  
 <**version**>${spring.cloud.version}</**version**>  
 <**type**>pom</**type**>  
 <**scope**>import</**scope**>  
 </**dependency**>  
 <!--spring cloud 阿里巴巴-->  
 <**dependency**>  
 <**groupId**>com.alibaba.cloud</**groupId**>  
 <**artifactId**>spring-cloud-alibaba-dependencies</**artifactId**>  
 <**version**>${spring.cloud.alibaba.version}</**version**>  
 <!--排除掉内置的seata，官方推荐用最新的，为了方便后续复习，这个排除放到了子模块的pom文件中-->  
 <**exclusions**>  
 <**exclusion**>  
 <**groupId**>io.seata</**groupId**>  
 <**artifactId**>seata-spring-boot-starter</**artifactId**>  
 </**exclusion**>  
 </**exclusions**>  
 <**type**>pom</**type**>  
 <**scope**>import</**scope**>  
 </**dependency**>  
 <!-- druid-->  
 <**dependency**>  
 <**groupId**>com.alibaba</**groupId**>  
 <**artifactId**>druid</**artifactId**>  
 <**version**>${druid.version}</**version**>  
 </**dependency**>  
 <**dependency**>  
 <**groupId**>com.alibaba</**groupId**>  
 <**artifactId**>easyexcel</**artifactId**>  
 <**version**>${easyexcel.version}</**version**>  
 </**dependency**>  
 <!--mybatis-plus-->  
 <**dependency**>  
 <**groupId**>com.baomidou</**groupId**>  
 <**artifactId**>mybatis-plus-boot-starter</**artifactId**>  
 <**version**>${mybatis-plus.version}</**version**>  
 </**dependency**>  
 <!--junit-->  
 <**dependency**>  
 <**groupId**>junit</**groupId**>  
 <**artifactId**>junit</**artifactId**>  
 <**version**>${junit.version}</**version**>  
 </**dependency**>  
 <!--log4j-->  
 <**dependency**>  
 <**groupId**>log4j</**groupId**>  
 <**artifactId**>log4j</**artifactId**>  
 <**version**>${log4j.version}</**version**>  
 </**dependency**>  
 <!--lombok-->  
 <**dependency**>  
 <**groupId**>org.projectlombok</**groupId**>  
 <**artifactId**>lombok</**artifactId**>  
 <**version**>${lombok.version}</**version**>  
 <**optional**>true</**optional**>  
 </**dependency**>  
 <!--swagger-->  
 <**dependency**>  
 <**groupId**>io.springfox</**groupId**>  
 <**artifactId**>springfox-swagger2</**artifactId**>  
 <**version**>${swagger.version}</**version**>  
 </**dependency**>  
 <!--swagger ui-->  
 <**dependency**>  
 <**groupId**>io.springfox</**groupId**>  
 <**artifactId**>springfox-swagger-ui</**artifactId**>  
 <**version**>${swagger.version}</**version**>  
 </**dependency**>  
 <!--hutool工具包-->  
 <**dependency**>  
 <**groupId**>cn.hutool</**groupId**>  
 <**artifactId**>hutool-all</**artifactId**>  
 <**version**>${hutool.version}</**version**>  
 </**dependency**>  
 <!--fastjson 阿里出品的json转对象，对象转json工具-->  
 <**dependency**>  
 <**groupId**>com.alibaba</**groupId**>  
 <**artifactId**>fastjson</**artifactId**>  
 <**version**>${fastjson.version}</**version**>  
 </**dependency**>  
 <!--json web token-->  
 <**dependency**>  
 <**groupId**>io.jsonwebtoken</**groupId**>  
 <**artifactId**>jjwt</**artifactId**>  
 <**version**>${jwt.version}</**version**>  
 </**dependency**>  
 <!-- fastDFS文件管理 -->  
 <**dependency**>  
 <**groupId**>org.csource</**groupId**>  
 <**artifactId**>fastdfs-client-java</**artifactId**>  
 <**version**>${fastdfs.version}</**version**>  
 <**exclusions**>  
 <**exclusion**>  
 <**groupId**>org.slf4j</**groupId**>  
 <**artifactId**>slf4j-log4j12</**artifactId**>  
 </**exclusion**>  
 </**exclusions**>  
 </**dependency**>  
 </**dependencies**>  
</**dependencyManagement**>

### common依赖

<**parent**>  
 <**artifactId**>tmall-parent</**artifactId**>  
 <**groupId**>com.study.tmall</**groupId**>  
 <**version**>1.0-SNAPSHOT</**version**>  
</**parent**>  
<**modelVersion**>4.0.0</**modelVersion**>  
  
<**artifactId**>common</**artifactId**>  
<**packaging**>pom</**packaging**>  
<**modules**>  
 <**module**>common-util</**module**>  
 <**module**>service-util</**module**>  
</**modules**>  
  
<**dependencies**>  
 <**dependency**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-starter-web</**artifactId**>  
 <**scope**>provided </**scope**>  
 </**dependency**>  
  
 <!--mybatis-plus-->  
 <**dependency**>  
 <**groupId**>com.baomidou</**groupId**>  
 <**artifactId**>mybatis-plus-boot-starter</**artifactId**>  
 <**scope**>provided </**scope**>  
 </**dependency**>  
  
 <!--lombok用来简化实体类：需要安装lombok插件-->  
 <**dependency**>  
 <**groupId**>org.projectlombok</**groupId**>  
 <**artifactId**>lombok</**artifactId**>  
 </**dependency**>  
  
 <!--swagger-->  
 <**dependency**>  
 <**groupId**>io.springfox</**groupId**>  
 <**artifactId**>springfox-swagger2</**artifactId**>  
 </**dependency**>  
 <**dependency**>  
 <**groupId**>io.springfox</**groupId**>  
 <**artifactId**>springfox-swagger-ui</**artifactId**>  
 </**dependency**>  
  
 <!--hutool-all-->  
 <**dependency**>  
 <**groupId**>cn.hutool</**groupId**>  
 <**artifactId**>hutool-all</**artifactId**>  
 </**dependency**>  
  
 <!--fastdfs文件管理客户端-->  
 <**dependency**>  
 <**groupId**>org.csource</**groupId**>  
 <**artifactId**>fastdfs-client-java</**artifactId**>  
 </**dependency**>  
</**dependencies**>

### common-util模块



#### pom.xml

<**parent**>  
 <**artifactId**>common</**artifactId**>  
 <**groupId**>com.study.tmall</**groupId**>  
 <**version**>1.0-SNAPSHOT</**version**>  
</**parent**>  
<**modelVersion**>4.0.0</**modelVersion**>  
  
<**artifactId**>common-util</**artifactId**>  
  
<**dependencies**>  
 <**dependency**>  
 <**groupId**>com.study.tmall</**groupId**>  
 <**artifactId**>model</**artifactId**>  
 <**version**>1.0-SNAPSHOT</**version**>  
 </**dependency**>  
  
 <**dependency**>  
 <**groupId**>org.apache.httpcomponents</**groupId**>  
 <**artifactId**>httpclient</**artifactId**>  
 </**dependency**>  
  
 <**dependency**>  
 <**groupId**>com.alibaba</**groupId**>  
 <**artifactId**>fastjson</**artifactId**>  
 </**dependency**>  
  
 <**dependency**>  
 <**groupId**>io.jsonwebtoken</**groupId**>  
 <**artifactId**>jjwt</**artifactId**>  
 </**dependency**>  
  
 <!-- jwt -->  
 <**dependency**>  
 <**groupId**>io.jsonwebtoken</**groupId**>  
 <**artifactId**>jjwt</**artifactId**>  
 </**dependency**>  
  
 <**dependency**>  
 <**groupId**>org.apache.httpcomponents</**groupId**>  
 <**artifactId**>httpclient</**artifactId**>  
 </**dependency**>  
</**dependencies**>

#### exception

GlobalExceptionHandler

@ControllerAdvice // 全局增强注解  
**public class** GlobalExceptionHandler {  
 */\*\*  
 \* 全局异常处理controller  
 \** ***@param*** *e  
 \** ***@return*** *\*/* @ExceptionHandler(value = Exception.**class**)  
 @ResponseBody  
 **public** Result error(Exception e){  
 // 捕获异常的语句  
 e.printStackTrace();  
 // 返回结果返回失败  
 **return** Result.*fail*();  
 }  
  
 */\*\*  
 \* 自定义全局异常处理  
 \** ***@param*** *e  
 \** ***@return*** *\*/* @ExceptionHandler(value = TmallException.**class**)  
 @ResponseBody  
 **public** Result error(TmallException e){  
 // 返回自定义异常  
 **return** Result.*build*(e.getCode(), e.getMessage());  
 }  
}

TmallException

@Data  
@ApiModel(description = "自定义全局异常类")  
**public class** TmallException **extends** RuntimeException{  
 @ApiModelProperty(value = "异常状态码")  
 **private** Integer code;  
  
 */\*\*  
 \* 通过状态码和消息创建异常对象  
 \** ***@param*** *code  
 \** ***@param*** *message  
 \*/* **public** TmallException(Integer code, String message){  
 **super**(message);  
 **this**.code = code;  
 }  
  
 */\*\*  
 \* 接收枚举对象  
 \** ***@param*** *resultCodeEnum  
 \*/* **public** TmallException(ResultCodeEnum resultCodeEnum){  
 **super**(resultCodeEnum.getMessage());  
 **this**.code = resultCodeEnum.getCode();  
 }  
}

#### result

Result

@Data  
@ApiModel(description = "控制层统一返回结果")  
**public class** Result<T> {  
 @ApiModelProperty(value = "状态码")  
 **private** Integer code;  
  
 @ApiModelProperty(value = "返回消息")  
 **private** String message;  
  
 @ApiModelProperty(value = "返回数据")  
 **private** T data;  
  
  
 */\*\*  
 \* 创建一个Result  
 \** ***@param*** *data  
 \** ***@param*** <*T*>  
 *\** ***@return*** *\*/* **protected static** <T> Result<T> build(T data){  
 Result<T> result = **new** Result<>();  
 **if** (data != **null**)  
 result.setData(data);  
 **return** result;  
 }  
  
 */\*\*  
 \* 创建一个有返回状态码的Result  
 \** ***@param*** *data  
 \** ***@param*** *resultCodeEnum  
 \** ***@param*** <*T*>  
 *\** ***@return*** *\*/* **public static** <T> Result<T> build(T data, ResultCodeEnum resultCodeEnum){  
 Result<T> result = *build*(data);  
 result.setCode(resultCodeEnum.getCode());  
 result.setMessage(resultCodeEnum.getMessage());  
 **return** result;  
 }  
  
 */\*\*  
 \* 创建一个自定义状态码和返回消息的Result  
 \** ***@param*** *code  
 \** ***@param*** *message  
 \** ***@param*** <*T*>  
 *\** ***@return*** *\*/* **public static** <T> Result<T> build(Integer code, String message){  
 Result<T> result = *build*(**null**);  
 result.setCode(code);  
 result.setMessage(message);  
 **return** result;  
 }  
  
 **public static** <T> Result<T> ok(){  
 **return** Result.*ok*(**null**);  
 }  
  
 */\*\*  
 \* 返回ok  
 \** ***@param*** *data  
 \** ***@param*** <*T*>  
 *\** ***@return*** *\*/* **public static** <T> Result<T> ok(T data){  
 **return** *build*(data, ResultCodeEnum.*SUCCESS*);  
 }  
  
 **public static** <T> Result<T> fail(){  
 **return** *fail*(**null**);  
 }  
  
 */\*\*  
 \* 返回失败  
 \** ***@param*** <*T*>  
 *\** ***@return*** *\*/* **public static** <T> Result<T> fail(T data){  
 **return** *build*(data, ResultCodeEnum.*FAIL*);  
 }  
  
 **public** Result<T> message(String msg) {  
 **this**.setMessage(msg);  
 **return this**;  
 }  
  
 **public** Result<T> code(Integer code) {  
 **this**.setCode(code);  
 **return this**;  
 }  
}

ResultCodeEnum

@ApiModel(description = "返回状态码枚举类")  
**public enum** ResultCodeEnum {  
 *SUCCESS*(200,"成功"),  
 *FAIL*(201, "失败"),  
 *PARAM\_ERROR*( 202, "参数不正确"),  
 *SERVICE\_ERROR*(203, "服务异常"),  
 *DATA\_ERROR*(204, "数据异常"),  
 *DATA\_UPDATE\_ERROR*(205, "数据版本异常"),  
  
 *URL\_ENCODE\_ERROR*( 217, "URL编码失败"),  
 *ILLEGAL\_CALLBACK\_REQUEST\_ERROR*( 218, "非法回调请求"),  
 *FETCH\_ACCESSTOKEN\_FAILD*( 219, "获取accessToken失败"),  
 *FETCH\_USERINFO\_ERROR*( 220, "获取用户信息失败"),  
  
 *PAY\_RUN*(221, "支付中"),  
 *CANCEL\_ORDER*(222, "订单已取消"),  
 *CANCEL\_ORDER\_FAIL*(225, "取消订单失败"),  
 *CANCEL\_ORDER\_NO*(225, "不能取消预约"),  
  
 *DELETE\_FAIL*(301, "删除失败"),  
 *PROPERTY\_NULL*(302, "未录入分类属性"),  
  
 *PAY\_FAIL*(400, "支付失败"),  
 *DELIVER\_GOODS\_FAIL*(401, "发货失败"),  
 *CONFIRM\_GOODS\_FAIL*(402, "确认收货失败"),  
  
 *NOT\_AUTH*(507, "用户需要认证"),  
  
 *LOGIN\_AUTH*(508, "未登陆"),  
 *PERMISSION*(509, "没有权限"),  
  
 *CODE\_ERROR*(510, "验证码错误"),  
 *LOGIN\_DISABLED\_ERROR*(512, "该用户已被禁用"),  
 *REGISTER\_MOBLE\_ERROR*(513, "手机号已被使用"),  
 *REGISTER\_EMAIL\_ERROR*(514, "邮箱已被使用"),  
 *LOGIN\_AURH*(515, "需要登录"),  
 *LOGIN\_ACL*(516, "没有权限"),  
 *LOGIN\_NAME\_FAIL*(517, "用户名或密码错误"),  
 *LOGIN\_PHONE\_FAIL*(518, "手机号或密码错误"),  
 *LOGIN\_EMAIL\_FAIL*(519, "邮箱或密码错误"),  
  
 *EMAIL\_UNBIND*(520, "邮箱未绑定"),  
 *EMAIL\_BIND*(521, " 邮箱已绑定")  
 ;  
  
 **private** Integer code;  
 **private** String message;  
  
 ResultCodeEnum(Integer code, String message) {  
 **this**.code = code;  
 **this**.message = message;  
 }  
  
 **public** Integer getCode() {  
 **return** code;  
 }  
  
 **public void** setCode(Integer code) {  
 **this**.code = code;  
 }  
  
 **public** String getMessage() {  
 **return** message;  
 }  
  
 **public void** setMessage(String message) {  
 **this**.message = message;  
 }  
}

#### util

ImageUtil

**public class** ImageUtil {  
  
 **private** ImageUtil(){}  
  
 */\*\*  
 \* 传入保存图片后fastUtil返回的字符数组  
 \* 返回融合后的图片url地址  
 \** ***@param*** *fileArray  
 \** ***@return*** *\*/* **public static** String compoundUrl(String[] fileArray){  
 // 最终访问地址  
 // http://nginx的Ip/组名/存储路径  
 // imageUrl = "http://192.168.123.130/" + fileArray[0] + "/" + fileArray[1]  
 // http://192.168.123.130/group1/M00/00/00/wKh7gmBrTfeEAyKkAAAAAFQAsAw989.jpg  
 String url = fileArray == **null** ? "" : ImageUrlEnum.*NGINX\_IP*.getStr() + "/" + fileArray[0] + "/" + fileArray[1];  
  
 **return** url;  
 }  
  
 */\*\*  
 \* 拆分数据库中存入的图片url  
 \** ***@param*** *imageUrl  
 \** ***@return*** *\*/* **public static** String[] splitUrl(String imageUrl){  
 // imageUrl = http://192.168.123.130/group1/M00/00/00/wKh7gmBrTfeEAyKkAAAAAFQAsAw989.jpg  
 // str[0] = http://192.168.123.130  
 // str[1] = group1  
 // str[2] = M00/00/00/wKh7gmBrTfeEAyKkAAAAAFQAsAw989.jpg  
  
 **int** ipLength = ImageUrlEnum.*NGINX\_IP*.getStr().length();  
 **int** groupLength = ImageUrlEnum.*GROUP\_1*.getStr().length();  
 String[] str = **new** String[3];  
 str[0] = imageUrl.substring(0, ipLength);  
 str[1] = imageUrl.substring(ipLength + 1, ipLength + groupLength + 1);  
 str[2] = imageUrl.substring(ipLength + groupLength + 2);  
 **return** str;  
 }  
  
 */\*\*  
 \* 获得文件后缀名  
 \** ***@param*** *file  
 \** ***@return*** *\*/* **public static** String getFileExtName(MultipartFile file){  
 String filename = file.getOriginalFilename(); // 获得文件名  
 String fileExtName = "jpg"; // 默认为jpg，防止出错  
 **if** (filename.contains(".")) {  
 fileExtName = filename.substring(filename.lastIndexOf(".") + 1); // 取得文件拓展名  
 }  
 **return** fileExtName;  
 }  
  
 */\*\*  
 \* 获得文件后缀名  
 \** ***@param*** *filename  
 \** ***@return*** *\*/* **public static** String getFileExtName(String filename){  
 String fileExtName = "jpg"; // 默认为jpg，防止出错  
 **if** (filename.contains(".")) {  
 fileExtName = filename.substring(filename.lastIndexOf(".") + 1); // 取得文件拓展名  
 }  
 **return** fileExtName;  
 }  
}

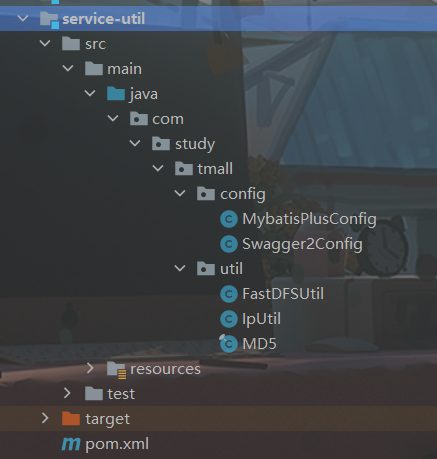
JwtHelper

**public class** JwtHelper {  
 **private static long** *tokenExpiration* = 24\*60\*60\*1000; // token过期时间  
 **private static** String *tokenSignKey* = "123456"; // token 签名密钥，可以自己设置  
  
 **private** JwtHelper(){}  
 */\*\*  
 \* 根据参数生成token  
 \** ***@param*** *userId  
 \** ***@param*** *password  
 \** ***@return*** *\*/* **public static** String createToken(String userId, String password) {  
 String token = Jwts.*builder*()  
 .setSubject("TMALL-USER") // 组名  
 .setExpiration(**new** Date(System.*currentTimeMillis*() + *tokenExpiration*)) // 设置token过期时间，毫秒单位  
 .claim("userId", userId)  
 .claim("password", password)  
 .signWith(SignatureAlgorithm.*HS512*, *tokenSignKey*)  
 .compressWith(CompressionCodecs.*GZIP*)  
 .compact();  
 **return** token;  
 }  
  
 */\*\*  
 \* 根据token取出userId  
 \** ***@param*** *token  
 \** ***@return*** *\*/* **public static** String getUserId(String token) **throws** ExpiredJwtException{  
 **if**(StringUtils.*isEmpty*(token)) **return** "";  
 String userId = **null**;  
 **try** {  
 Jws<Claims> claimsJws = Jwts.*parser*().setSigningKey(*tokenSignKey*).parseClaimsJws(token);  
 Claims claims = claimsJws.getBody();  
 userId = (String) claims.get("userId");  
 } **catch** (ExpiredJwtException e) {  
 // token过期  
 System.*out*.println("token过期：" + e);  
 **throw** e;  
 }  
 **return** userId;  
 }  
  
 */\*\*  
 \* 根据token取出password  
 \** ***@param*** *token  
 \** ***@return*** *\*/* **public static** String getPassword(String token) **throws** ExpiredJwtException {  
 **if**(StringUtils.*isEmpty*(token)) **return** "";  
 String password = **null**;  
 **try** {  
 Jws<Claims> claimsJws = Jwts.*parser*().setSigningKey(*tokenSignKey*).parseClaimsJws(token);  
 Claims claims = claimsJws.getBody();  
 password = (String)claims.get("password");  
 } **catch** (ExpiredJwtException e) {  
 // token过期  
 System.*out*.println("token过期：" + e);  
 **throw** e;  
 }  
 **return** password;  
 }  
}

TokenUtil

**public class** TokenUtil {  
  
 **private** TokenUtil(){}  
  
 */\*\*  
 \* 根据参数生成token  
 \** ***@param*** *userId  
 \** ***@param*** *password  
 \** ***@return*** *\*/* **public static** String createToken(String userId, String password) {  
 **return** JwtHelper.*createToken*(userId, password);  
 }  
  
 */\*\*  
 \* 获取token中的用户密码  
 \** ***@param*** *request  
 \** ***@return*** *\*/* **public static** String getPassword(HttpServletRequest request){  
 **try** {  
 // 获取请求头中的token  
 String token = request.getHeader("token");  
 // 使用JwtHelper工具类获取token中的用户密码  
 String password = JwtHelper.*getPassword*(token);  
 **return** password;  
 } **catch** (ExpiredJwtException e) {  
 **throw new** TmallException(ResultCodeEnum.*LOGIN\_AUTH*);  
 }  
 }  
  
 */\*\*  
 \* 获取token中的用户密码  
 \** ***@param*** *token  
 \** ***@return*** *\*/* **public static** String getPassword(String token){  
 **try** {  
 // 使用JwtHelper工具类获取token中的用户密码  
 String password = JwtHelper.*getPassword*(token);  
 **return** password;  
 } **catch** (ExpiredJwtException e) {  
 **throw new** TmallException(ResultCodeEnum.*LOGIN\_AUTH*);  
 }  
 }  
  
 */\*\*  
 \* 获取token中的用户ID  
 \** ***@param*** *request  
 \** ***@return*** *\*/* **public static** String getUserId(HttpServletRequest request){  
 **try** {  
 // 获取请求头中的token  
 String token = request.getHeader("token");  
 // 使用JwtHelper工具类获取token中的用户ID  
 String userId = JwtHelper.*getUserId*(token);  
 **return** userId;  
 } **catch** (ExpiredJwtException e) {  
 **throw new** TmallException(ResultCodeEnum.*LOGIN\_AUTH*);  
 }  
 }  
  
 */\*\*  
 \* 获取token中的用户ID  
 \** ***@param*** *token  
 \** ***@return*** *\*/* **public static** String getUserId(String token){  
 **try** {  
 // 使用JwtHelper工具类获取token中的用户ID  
 String userId = JwtHelper.*getUserId*(token);  
 **return** userId;  
 } **catch** (ExpiredJwtException e) {  
 **throw new** TmallException(ResultCodeEnum.*LOGIN\_AUTH*);  
 }  
 }  
}

### service-util模块



#### pom.xml

<**parent**>  
 <**artifactId**>common</**artifactId**>  
 <**groupId**>com.study.tmall</**groupId**>  
 <**version**>1.0-SNAPSHOT</**version**>  
</**parent**>  
<**modelVersion**>4.0.0</**modelVersion**>  
  
<**artifactId**>service-util</**artifactId**>  
  
<**dependencies**>  
 <**dependency**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-starter-data-redis</**artifactId**>  
 </**dependency**>  
 <!-- spring2.X集成redis所需common-pool2-->  
 <**dependency**>  
 <**groupId**>org.apache.commons</**groupId**>  
 <**artifactId**>commons-pool2</**artifactId**>  
 <**version**>2.6.0</**version**>  
 </**dependency**>  
</**dependencies**>

#### config

MybatisPlusConfig：MP配置类

@Configuration  
@MapperScan("com.study.tmall.\*.mapper")  
@EnableTransactionManagement // 开启事务支持  
**public class** MybatisPlusConfig {  
 // 注册乐观锁插件(旧版：3.0.5)  
 /\*@Bean  
 public OptimisticLockerInnerInterceptor optimisticLockerInnerInterceptor(){  
 return new OptimisticLockerInnerInterceptor();  
 }\*/  
  
 //注册分页插件(旧版：3.0.5)  
 /\*  
 @Bean  
 public PaginationInterceptor paginationInterceptor(){  
 return new PaginationInterceptor();  
 }  
 \*/  
  
 // 注册乐观锁和分页插件(新版：3.4.0)  
 @Bean  
 **public** MybatisPlusInterceptor mybatisPlusInterceptor(){  
 MybatisPlusInterceptor interceptor = **new** MybatisPlusInterceptor();  
 interceptor.addInnerInterceptor(**new** OptimisticLockerInnerInterceptor()); // 乐观锁插件  
 // DbType：数据库类型(根据类型获取应使用的分页方言)  
 interceptor.addInnerInterceptor(**new** PaginationInnerInterceptor(DbType.*MYSQL*)); // 分页插件  
 **return** interceptor;  
 }  
}

RedisConfig：redis配置类

@Configuration  
@EnableCaching  
**public class** RedisConfig {  
 */\*\*  
 \* 自定义key规则  
 \** ***@return*** *\*/* @Bean  
 **public** KeyGenerator keyGenerator() {  
 **return new** KeyGenerator() {  
 @Override  
 **public** Object generate(Object target, Method method, Object... params) {  
 StringBuilder sb = **new** StringBuilder();  
 sb.append(target.getClass().getName());  
 sb.append(method.getName());  
 **for** (Object obj : params) {  
 sb.append(obj.toString());  
 }  
 **return** sb.toString();  
 }  
 };  
 }  
  
 */\*\*  
 \* 自定义适用于分页的key规则  
 \* 根据类名，方法名，分页参数（current，limit）来区分key  
 \** ***@return*** *\*/* @Bean  
 **public** KeyGenerator keyGeneratorPage() {  
 **return new** KeyGenerator() {  
 @Override  
 **public** Object generate(Object target, Method method, Object... params) {  
 StringBuilder sb = **new** StringBuilder();  
 sb.append(target.getClass().getName());  
 sb.append(method.getName());  
 **for** (Object obj : params) {  
 **if** (obj **instanceof** Page) {  
 Page page = (Page)obj;  
 sb.append("@page");  
 sb.append(":current=" + page.getCurrent());  
 sb.append(":limit=" + page.getSize());  
 }  
 }  
 **return** sb.toString();  
 }  
 };  
 }  
  
 */\*\*  
 \* 自定义适用于分页的key规则  
 \* 根据类名，方法名，分页参数（current，limit） , id号来区分key  
 \** ***@return*** *\*/* @Bean  
 **public** KeyGenerator keyGeneratorIdPage() {  
 **return new** KeyGenerator() {  
 @Override  
 **public** Object generate(Object target, Method method, Object... params) {  
 StringBuilder sb = **new** StringBuilder();  
 sb.append(target.getClass().getName());  
 sb.append(method.getName());  
 **for** (Object obj : params) {  
 **if** (obj **instanceof** Page) {  
 Page page = (Page)obj;  
 sb.append("@page");  
 sb.append(":current=" + page.getCurrent());  
 sb.append(":limit=" + page.getSize());  
 } **else if** (obj **instanceof** String) {  
 String id = (String) obj;  
 sb.append("@id=" + id);  
 }  
 }  
 **return** sb.toString();  
 }  
 };  
 }  
  
 */\*\*  
 \* 只适用于产品查询且分页  
 \* 根据类名，方法名，分页参数（current，limit），keyword 来区分key  
 \** ***@return*** *\*/* @Bean  
 **public** KeyGenerator keyGeneratorKeywordPage() {  
 **return new** KeyGenerator() {  
 @Override  
 **public** Object generate(Object target, Method method, Object... params) {  
 StringBuilder sb = **new** StringBuilder();  
 sb.append(target.getClass().getName());  
 sb.append(method.getName());  
 **for** (Object obj : params) {  
 **if** (obj **instanceof** Page) {  
 Page page = (Page)obj;  
 sb.append("@page");  
 sb.append(":current=" + page.getCurrent());  
 sb.append(":limit=" + page.getSize());  
 }  
 **if** (obj **instanceof** String) {  
 String keyword = (String)obj;  
 sb.append("@keyword=" + keyword);  
 }  
 }  
 **return** sb.toString();  
 }  
 };  
 }  
  
 */\*\*  
 \* 设置RedisTemplate规则  
 \** ***@param*** *redisConnectionFactory  
 \** ***@return*** *\*/* @Bean  
 **public** RedisTemplate<Object, Object> redisTemplate(RedisConnectionFactory redisConnectionFactory) {  
 RedisTemplate<Object, Object> redisTemplate = **new** RedisTemplate<>();  
 redisTemplate.setConnectionFactory(redisConnectionFactory);  
 Jackson2JsonRedisSerializer jackson2JsonRedisSerializer = **new** Jackson2JsonRedisSerializer(Object.**class**);  
  
 //解决查询缓存转换异常的问题  
 ObjectMapper om = **new** ObjectMapper();  
 // 指定要序列化的域，field,get和set,以及修饰符范围，ANY是都有包括private和public  
 om.setVisibility(PropertyAccessor.*ALL*, JsonAutoDetect.Visibility.*ANY*);  
 // 指定序列化输入的类型，类必须是非final修饰的，final修饰的类，比如String,Integer等会跑出异常  
 om.enableDefaultTyping(ObjectMapper.DefaultTyping.*NON\_FINAL*);  
 jackson2JsonRedisSerializer.setObjectMapper(om);  
  
 //序列号key value  
 redisTemplate.setKeySerializer(**new** StringRedisSerializer());  
 redisTemplate.setValueSerializer(jackson2JsonRedisSerializer);  
 redisTemplate.setHashKeySerializer(**new** StringRedisSerializer());  
 redisTemplate.setHashValueSerializer(jackson2JsonRedisSerializer);  
  
 redisTemplate.afterPropertiesSet();  
 **return** redisTemplate;  
 }  
  
 */\*\*  
 \* 设置CacheManager缓存规则  
 \** ***@param*** *factory  
 \** ***@return*** *\*/* @Bean  
 **public** CacheManager cacheManager(RedisConnectionFactory factory) {  
 RedisSerializer<String> redisSerializer = **new** StringRedisSerializer();  
 Jackson2JsonRedisSerializer jackson2JsonRedisSerializer = **new** Jackson2JsonRedisSerializer(Object.**class**);  
  
 //解决查询缓存转换异常的问题  
 ObjectMapper om = **new** ObjectMapper();  
 om.setVisibility(PropertyAccessor.*ALL*, JsonAutoDetect.Visibility.*ANY*);  
 om.enableDefaultTyping(ObjectMapper.DefaultTyping.*NON\_FINAL*);  
 jackson2JsonRedisSerializer.setObjectMapper(om);  
  
 // 配置序列化（解决乱码的问题）,过期时间600秒  
 RedisCacheConfiguration config = RedisCacheConfiguration.*defaultCacheConfig*()  
 .entryTtl(Duration.*ofSeconds*(600))  
 .serializeKeysWith(RedisSerializationContext.SerializationPair.*fromSerializer*(redisSerializer))  
 .serializeValuesWith(RedisSerializationContext.SerializationPair.*fromSerializer*(jackson2JsonRedisSerializer))  
 .disableCachingNullValues();  
  
 RedisCacheManager cacheManager = RedisCacheManager.*builder*(factory)  
 .cacheDefaults(config)  
 .build();  
 **return** cacheManager;  
 }  
}

Swagger2Config：Swagger配置类

@Configuration  
@EnableSwagger2  
**public class** Swagger2Config {  
 @Bean  
 **public** Docket webApiConfig(){  
  
 **return new** Docket(DocumentationType.*SWAGGER\_2*)  
 .groupName("webApi")  
 .apiInfo(webApiInfo())  
 .select()  
 //只显示api路径下的页面  
 .paths(Predicates.*and*(PathSelectors.*regex*("/api/.\*")))  
 .build();  
  
 }  
  
 @Bean  
 **public** Docket adminApiConfig(){  
  
 **return new** Docket(DocumentationType.*SWAGGER\_2*)  
 .groupName("adminApi")  
 .apiInfo(adminApiInfo())  
 .select()  
 //只显示admin路径下的页面  
 .paths(Predicates.*and*(PathSelectors.*regex*("/admin/.\*")))  
 .build();  
  
 }  
  
 **private** ApiInfo webApiInfo(){  
  
 **return new** ApiInfoBuilder()  
 .title("网站-API文档")  
 .description("本文档描述了网站微服务接口定义")  
 .version("1.0")  
 .contact(**new** Contact("2Executioner", "http://localhost.com", "1205878539@qq.com"))  
 .build();  
 }  
  
 **private** ApiInfo adminApiInfo(){  
  
 **return new** ApiInfoBuilder()  
 .title("后台管理系统-API文档")  
 .description("本文档描述了后台管理系统微服务接口定义")  
 .version("1.0")  
 .contact(**new** Contact("2Executioner", "http://localhost.com", "1205878539@qq.com"))  
 .build();  
 }  
}

#### util

Base64

**public class** Base64 {  
 **public static final** String *base64Slot* = "2Executioner";  
 **private static final** String *slotEncrypt*;  
 **private static final** org.apache.tomcat.util.codec.binary.Base64 *base64*;  
  
 **static** {  
 // 对盐进行加密  
 *slotEncrypt* = MD5.*base64SlotEncrypt*(*base64Slot*);  
 *base64* = **new** org.apache.tomcat.util.codec.binary.Base64();  
 }  
  
 **private** Base64() {}  
  
 */\*\*  
 \* base64编码  
 \** ***@param*** *data  
 \** ***@return*** *\*/* **public static** String encode(String data) {  
 data = **new** String(*base64*.encode(data.getBytes()));  
 data += *slotEncrypt*;  
 data = **new** String(*base64*.encode(data.getBytes()));  
 **return** data;  
 }  
  
 */\*\*  
 \* 对前端的Base64编码进行解码  
 \** ***@param*** *data  
 \** ***@return*** *\*/* **public static** String decode(String data) {  
 data = **new** String(*base64*.decode(data));  
 data = data.substring(0, data.length() - *slotEncrypt*.length());  
 data = **new** String(*base64*.decode(data));  
 **return** data;  
 }  
}

FastDFSUtil

**public class** FastDFSUtil {  
 **static** {  
 **try** {  
 // 读取FastDFS配置文件，将配置文件中所有的tracker地址读取到内存中  
 ClientGlobal.*init*("fastdfs.conf");  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 } **catch** (MyException e) {  
 e.printStackTrace();  
 }  
 }  
  
 **private** FastDFSUtil(){}  
  
 */\*\*  
 \* 上传文件  
 \** ***@param*** *fileBuff 为需要上传的文件的字节数组  
 \** ***@param*** *fileExtName 为需要上传的文件的扩展名  
 \** ***@return*** *返回一个String数组，这个数据十分重要，需要妥善保管，建议存入数据库  
 \* 数组中的第一个元素为文件所在的组名  
 \* 数组中的第二个元素为文件所在远程路径名  
 \*/* **public static** String[] upload(**byte**[] fileBuff, String fileExtName){  
 TrackerServer ts = **null**;  
 StorageServer ss = **null**;  
  
 **try** {  
 TrackerClient trackerClient = **new** TrackerClient();  
 ts = trackerClient.getConnection();  
 ss = trackerClient.getStoreStorage(ts);  
 // 定义Storage的客户端对象，需要用这个对象来上传 下载和删除文件  
 StorageClient storageClient = **new** StorageClient(ts, ss);  
  
 String[] result = storageClient.upload\_appender\_file(fileBuff, fileExtName, **null**);  
 **return** result;  
  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 } **catch** (MyException e) {  
 e.printStackTrace();  
 } **finally** {  
 **if** (ss != **null**){  
 **try** {  
 ss.close();  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 **if** (ts != **null**){  
 **try** {  
 ts.close();  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
 **return null**;  
  
 }  
  
 */\*\*  
 \* 下载文件  
 \** ***@param*** *groupName 需要下载文件所在组名  
 \** ***@param*** *remoteFilename 需要下载文件的远程文件名  
 \** ***@return*** *返回字节数组作响应数据  
 \*/* **public static byte**[] download(String groupName, String remoteFilename){  
 TrackerServer ts = **null**;  
 StorageServer ss = **null**;  
 **try** {  
 TrackerClient trackerClient = **new** TrackerClient();  
 ts = trackerClient.getConnection();  
 ss = trackerClient.getStoreStorage(ts);  
 StorageClient storageClient = **new** StorageClient(ts, ss);  
  
 **byte**[] result = storageClient.download\_file(groupName, remoteFilename);  
 **return** result;  
  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 } **catch** (MyException e) {  
 e.printStackTrace();  
 } **finally** {  
 **if** (**null** != ss){  
 **try** {  
 ss.close();  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 **if** (**null** != ts) {  
 **try** {  
 ts.close();  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
 **return null**;  
  
 }  
  
 */\*\*  
 \* 删除文件  
 \*/* **public static** Integer delete(String groupName, String remoteFilename){  
 TrackerServer ts = **null**;  
 StorageServer ss = **null**;  
 **try** {  
 TrackerClient trackerClient = **new** TrackerClient();  
 ts = trackerClient.getConnection();  
 ss = trackerClient.getStoreStorage(ts);  
 StorageClient storageClient = **new** StorageClient(ts, ss);  
 */\*\*  
 \* 参数1： 要删除文件的组名  
 \* 参数2： 要删除文件的远程文件名  
 \* 返回一个int类型的数据，返回 0 表示删除成功，返回其它任何非0值都表示删除失败  
 \*  
 \*/* Integer result = storageClient.delete\_file(groupName, remoteFilename);  
 **return** result;  
  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 } **catch** (MyException e) {  
 e.printStackTrace();  
 } **finally** {  
 **if** (**null** != ss){  
 **try** {  
 ss.close();  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 **if** (**null** != ts){  
 **try** {  
 ts.close();  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
 **return null**;  
 }  
}

IpUtil

**public class** IpUtil {  
 */\*\*  
 \* 获取请求用户的真实ip  
 \** ***@param*** *request  
 \** ***@return*** *\*/* **public static** String getIpAddr(HttpServletRequest request){  
 **if** (request.getHeader("x-forwarded-for") == **null**) {  
 **return** request.getRemoteAddr();  
 }  
 **return** request.getHeader("x-forwarded-for");  
 }  
}

MD5

**public final class** MD5 {  
 */\*\*  
 \* MD5转码  
 \** ***@param*** *strSrc  
 \** ***@return*** *\*/* **public static** String encrypt(String strSrc) {  
 **try** {  
 **char** hexChars[] = { 'f', 'e', 'd', 'c', 'b', 'a', '9', '8', '7',  
 '6', '5', '4', '3', '2', '1', '0' };  
 **byte**[] bytes = strSrc.getBytes();  
 MessageDigest md = MessageDigest.*getInstance*("MD5");  
 md.update(bytes);  
 bytes = md.digest();  
 **int** j = bytes.length;  
 **char**[] chars = **new char**[j \* 2];  
 **int** k = 0;  
 **for** (**int** i = 0; i < bytes.length; i++) {  
 **byte** b = bytes[i];  
 chars[k++] = hexChars[b >>> 4 & 0xf];  
 chars[k++] = hexChars[b & 0xf];  
 }  
 **return new** String(chars);  
 } **catch** (NoSuchAlgorithmException e) {  
 e.printStackTrace();  
 **throw new** RuntimeException("MD5加密出错！！+" + e);  
 }  
 }  
  
  
 */\*\*  
 \* MD5加盐转码（对base64编码中的盐加密）  
 \** ***@param*** *strSrc  
 \** ***@return*** *\*/* **public static** String base64SlotEncrypt(String strSrc) {  
 **try** {  
 **char** hexChars[] = { '0', '1', '2', '3', '4', '5', '6', '7', '8',  
 '9', 'a', 'b', 'c', 'd', 'e', 'f' };  
 **byte**[] bytes = strSrc.getBytes();  
 MessageDigest md = MessageDigest.*getInstance*("MD5");  
 md.update(bytes);  
 bytes = md.digest();  
 **int** j = bytes.length;  
 **char**[] chars = **new char**[j \* 2];  
 **int** k = 0;  
 **for** (**int** i = 0; i < bytes.length; i++) {  
 **byte** b = bytes[i];  
 chars[k++] = hexChars[b >>> 4 & 0xf];  
 chars[k++] = hexChars[b & 0xf];  
 }  
 // 加盐  
 String strChars = **new** String(chars);  
 StringBuilder result = **new** StringBuilder();  
 **for** (**int** i = 0; i < strChars.length(); i++) {  
 // 转换为二进制  
 String context = Integer.*toBinaryString*(strChars.charAt(i));  
 Integer binary = Integer.*valueOf*(context);  
 // 根据位运算从盐中选取颗粒  
 result.append(hexChars[binary >>> 4 & 0xf]);  
 result.append(hexChars[binary & 0xf]);  
 }  
 **return** result.toString();  
 } **catch** (NoSuchAlgorithmException e) {  
 e.printStackTrace();  
 **throw new** RuntimeException("MD5加密出错！！+" + e);  
 }  
 }  
}

### model模块



#### pom.xml

<**parent**>  
 <**artifactId**>tmall-parent</**artifactId**>  
 <**groupId**>com.study.tmall</**groupId**>  
 <**version**>1.0-SNAPSHOT</**version**>  
</**parent**>  
<**modelVersion**>4.0.0</**modelVersion**>  
  
<**artifactId**>model</**artifactId**>  
<**packaging**>jar</**packaging**>  
  
<**dependencies**>  
 <!-- 阿里excel -->  
 <**dependency**>  
 <**groupId**>com.alibaba</**groupId**>  
 <**artifactId**>easyexcel</**artifactId**>  
 <**scope**>provided</**scope**>  
 </**dependency**>  
 <**dependency**>  
 <**groupId**>org.projectlombok</**groupId**>  
 <**artifactId**>lombok</**artifactId**>  
 </**dependency**>  
 <!--mybatis-plus-->  
 <**dependency**>  
 <**groupId**>com.baomidou</**groupId**>  
 <**artifactId**>mybatis-plus-boot-starter</**artifactId**>  
 <**scope**>provided </**scope**>  
 </**dependency**>  
 <!--swagger-->  
 <**dependency**>  
 <**groupId**>io.springfox</**groupId**>  
 <**artifactId**>springfox-swagger2</**artifactId**>  
 <**scope**>provided </**scope**>  
 </**dependency**>  
 <**dependency**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-starter-data-mongodb</**artifactId**>  
 <**scope**>provided </**scope**>  
 </**dependency**>  
 <**dependency**>  
 <**groupId**>com.alibaba</**groupId**>  
 <**artifactId**>fastjson</**artifactId**>  
 <**scope**>provided</**scope**>  
 </**dependency**>  
</**dependencies**>

#### dto

DealNotify：交易通知传输类

@Data  
@ApiModel(description = "交易通知传输类")  
**public class** DealNotify {  
 @ApiModelProperty("订单对象")  
 **private** OrderInfo orderInfo;  
  
 @ApiModelProperty("接收者邮箱")  
 **private** String receiverEmail;  
  
 @ApiModelProperty("订单项集合")  
 **private** List<OrderItem> orderItemList;  
}

ProductStock：商品库存传输类

@Data  
@ApiModel(description = "商品库存传输类")  
**public class** ProductStock {  
 @ApiModelProperty("订单项id")  
 **private** String id;  
  
 @ApiModelProperty("商品id")  
 **private** String productId;  
  
 @ApiModelProperty("更新数量")  
 **private** Integer number;  
  
 @ApiModelProperty("更新规则（加 还是 减）")  
 **private** ArithmeticTypeEnum type;  
}

TimerTask<T> ：计时器任务传输对象

@Data  
@ApiModel(description = "计时器任务传输对象")  
**public class** TimerTask<T> {  
 @ApiModelProperty("结束时间，当前时间(ms)+到执行时间的等待时间(ms)")  
 **private** Long executeTime;  
  
 @ApiModelProperty("携带的数据")  
 **private** T data;  
  
 @ApiModelProperty("任务类型")  
 **private** TaskTypeEnum type;  
  
 */\*\*  
 \* 设置多少ms后执行  
 \** ***@param*** *time 多少ms后  
 \*/* **public void** setExecuteTime(Long time) {  
 **this**.executeTime = System.*currentTimeMillis*() + time;  
 }  
}

#### enums

ArithmeticTypeEnum

@ApiModel(description = "算术运算类型枚举类")  
**public enum** ArithmeticTypeEnum {  
 *ADD*(0), // 加法  
 *SUBTRACT*(1), // 减法  
 *MULTIPLY*(2), // 乘法  
 *DIVIDE*(3), // 除法  
 *REMAINDER*(4), // 取余  
 ;  
  
 **private** Integer type;  
  
 ArithmeticTypeEnum(Integer type) {  
 **this**.type = type;  
 }  
  
 **public** Integer getType() {  
 **return** type;  
 }  
  
 **public void** setType(Integer type) {  
 **this**.type = type;  
 }  
}

AuthStatusEnum：认证状态枚举类

@ApiModel(description = "认证状态枚举类")  
**public enum** AuthStatusEnum {  
  
 *NO\_AUTH*(0, "未认证"),  
 *AUTH\_RUN*(1, "认证中"),  
 *AUTH\_SUCCESS*(2, "认证成功"),  
 *AUTH\_FAIL*(-1, "认证失败"),  
 ;  
  
 **private** Integer status;  
 **private** String name;  
  
 AuthStatusEnum(Integer status, String name){  
 **this**.status = status;  
 **this**.name = name;  
 }  
  
  
 */\*\*  
 \* 根据认证status获得认证name  
 \** ***@param*** *status  
 \** ***@return*** *\*/* **public static** String getStatusNameByStatus(Integer status){  
 AuthStatusEnum[] values = AuthStatusEnum.*values*();  
 **for** (AuthStatusEnum obj : values) {  
 **if**(status.intValue() == obj.getStatus().intValue()){  
 **return** obj.getName();  
 }  
 }  
 **return** "";  
 }  
  
 **public** Integer getStatus() {  
 **return** status;  
 }  
  
 **public void** setStatus(Integer status) {  
 **this**.status = status;  
 }  
  
 **public** String getName() {  
 **return** name;  
 }  
  
 **public void** setName(String name) {  
 **this**.name = name;  
 }  
}

EmailCodeTypeEnum：验证码类型

@ApiModel(description = "邮箱验证码类型枚举")  
**public enum** EmailCodeTypeEnum {  
 *LOGIN\_CODE*(1, "login#"), // 登录验证码，#为了和邮箱地址进行分隔  
 *REGISTER\_CODE*(2, "register#"), // 注册验证码  
 ;  
  
 **private** Integer number;  
 **private** String type;  
  
 EmailCodeTypeEnum(Integer number, String type) {  
 **this**.number = number;  
 **this**.type = type;  
 }  
  
 // 根据number获取type  
 **public static** String getTypeByNumber(Integer number) {  
 EmailCodeTypeEnum[] values = EmailCodeTypeEnum.*values*();  
 **for** (EmailCodeTypeEnum obj : values) {  
 **if** (number.intValue() == obj.getNumber().intValue()) {  
 **return** obj.getType();  
 }  
 }  
 **return** "";  
 }  
  
 **public** Integer getNumber() {  
 **return** number;  
 }  
  
 **public void** setNumber(Integer number) {  
 **this**.number = number;  
 }  
  
 **public** String getType() {  
 **return** type;  
 }  
  
 **public void** setType(String type) {  
 **this**.type = type;  
 }  
}

ImageTypeEnum：图片类型枚举类

@ApiModel(description = "图片类型枚举类")  
**public enum** ImageTypeEnum {  
 *THUMBNAIL*(0, "thumbnail"), // 缩略图 thumbnail  
 *DETAILS\_FIGURE*(1, "details"), // 详情图 details  
 ;  
  
 **private** Integer type;  
 **private** String name;  
  
 ImageTypeEnum(Integer type, String name){  
 **this**.type = type;  
 **this**.name = name;  
 }  
  
 **public static** String getNameByType(Integer type){  
 ImageTypeEnum[] values = ImageTypeEnum.*values*();  
 **for** (ImageTypeEnum obj : values) {  
 **if** (type.intValue() == obj.getType().intValue()){  
 **return** obj.getName();  
 }  
 }  
 **return** "";  
 }  
  
 **public** Integer getType() {  
 **return** type;  
 }  
  
 **public void** setType(Integer type) {  
 **this**.type = type;  
 }  
  
 **public** String getName() {  
 **return** name;  
 }  
  
 **public void** setName(String name) {  
 **this**.name = name;  
 }  
}

ImageUrlEnum：图片url枚举类

**public enum** ImageUrlEnum {  
 *NGINX\_IP*("http://192.168.123.130"), // nginx的ip  
 *GROUP\_1*("group1"),  
 ;  
  
 **private** String str; // 图片服务器的ip  
  
 ImageUrlEnum(String str) {  
 **this**.str = str;  
 }  
  
 **public** String getStr() {  
 **return** str;  
 }  
  
 **public void** setStr(String str) {  
 **this**.str = str;  
 }  
}

OrderStatusEnum：订单状态枚举类

@ApiModel(description = "订单状态枚举类")  
**public enum** OrderStatusEnum {  
 *TRADING\_CLOSED*(-1, "交易关闭"),  
 *WAIT\_PAY*(0, "待付款"),  
 *WAIT\_SHIPMENTS*(1, "待发货"),  
 *WAIT\_TAKE\_GOODS*(2, "待收货"),  
 *WAIT\_REVIEW*(3, "待评价"),  
 *COMPLETE\_TRANSACTION*(4, "交易完成"), // 确认了收货就完成了交易  
 ;  
  
 **private** Integer status;  
 **private** String name;  
  
 OrderStatusEnum(Integer status, String name) {  
 **this**.status = status;  
 **this**.name = name;  
 }  
  
 */\*\*  
 \* 返回所有枚举值  
 \** ***@return*** *返回list集合，方便前端显示  
 \*/* **public static** List<Map<String, Object>> getStatusList(){  
 List<Map<String, Object>> list = **new** ArrayList<>();  
 OrderStatusEnum[] values = OrderStatusEnum.*values*();  
 **for** (OrderStatusEnum obj : values) {  
 Map<String, Object> map = **new** HashMap<>();  
 map.put("status", obj.getStatus());  
 map.put("name", obj.getName());  
 list.add(map);  
 }  
 **return** list;  
 }  
  
 **public static** String getStatusNameByStatus(Integer status){  
 OrderStatusEnum[] values = OrderStatusEnum.*values*();  
 **for** (OrderStatusEnum obj : values) {  
 **if** (status.intValue() == obj.getStatus().intValue()){  
 **return** obj.getName();  
 }  
 }  
 **return** "";  
 }  
  
 // 判断status状态是否在枚举值中  
 **public static boolean** exist(Integer status) {  
 OrderStatusEnum[] values = OrderStatusEnum.*values*();  
 **for** (OrderStatusEnum obj : values) {  
 **if** (status.intValue() == obj.getStatus().intValue()){  
 **return true**;  
 }  
 }  
 **return false**;  
 }  
  
 **public** Integer getStatus() {  
 **return** status;  
 }  
  
 **public void** setStatus(Integer status) {  
 **this**.status = status;  
 }  
  
 **public** String getName() {  
 **return** name;  
 }  
  
 **public void** setName(String name) {  
 **this**.name = name;  
 }  
}

PaymentStatusEnum：支付状态枚举类

@ApiModel(description = "支付状态枚举类")  
**public enum** PaymentStatusEnum {  
 *FAIL*(0, "支付失败"),  
 *UNPAID*(1,"支付中"),  
 *PAID*(2, "已支付")  
 ;  
  
 **private** Integer status;  
 **private** String name;  
  
 PaymentStatusEnum(Integer status, String name) {  
 **this**.status = status;  
 **this**.name = name;  
 }  
  
 **public static** String getStatusNameByStatus(Integer status){  
 PaymentStatusEnum[] values = PaymentStatusEnum.*values*();  
 **for** (PaymentStatusEnum obj : values) {  
 **if** (status.intValue() == obj.getStatus().intValue()){  
 **return** obj.getName();  
 }  
 }  
 **return** "";  
 }  
  
 **public** Integer getStatus() {  
 **return** status;  
 }  
  
 **public void** setStatus(Integer status) {  
 **this**.status = status;  
 }  
  
 **public** String getName() {  
 **return** name;  
 }  
  
 **public void** setName(String name) {  
 **this**.name = name;  
 }  
}

PaymentTypeEnum：支付类型枚举类

@ApiModel(description = "支付类型枚举类")  
**public enum** PaymentTypeEnum {  
 *ALIPAY*(1, "支付宝"),  
 *WEIXIN*(2, "微信");  
 ;  
  
 **private** Integer status;  
 **private** String comment;  
  
 PaymentTypeEnum(Integer status, String comment) {  
 **this**.status = status;  
 **this**.comment = comment;  
 }  
  
 **public** Integer getStatus() {  
 **return** status;  
 }  
  
 **public void** setStatus(Integer status) {  
 **this**.status = status;  
 }  
  
 **public** String getComment() {  
 **return** comment;  
 }  
  
 **public void** setComment(String comment) {  
 **this**.comment = comment;  
 }  
}

ProductSortStatusEnum

@ApiModel(description = "前台系统商品排序状态枚举类")  
**public enum** ProductSortStatusEnum {  
 *DEFAULT*(0, ""), // 默认排序  
 *POPULAR*(1, "reviews"), // 按人气（评价数）降序排序  
 *NEW\_PRODUCTS*(2, "create\_time"), // 新品，按创建时间(从新到旧)降序排序  
 *SALES\_VOLUME*(3, "monthly\_sales"), // 销量，按月销量降序排序  
 *PRICE*(4, "promote\_price"), // 按价格排序，  
 ;  
  
 **private** Integer number; // 排序编号  
 **private** String field; // 字段名  
  
 ProductSortStatusEnum(Integer number, String field) {  
 **this**.number = number;  
 **this**.field = field;  
 }  
  
 // 根据排序编号取得排序字段  
 **public static** String getFieldByNumber(Integer number) {  
 **if** (number == **null**) {  
 **return** "";  
 }  
 ProductSortStatusEnum[] values = ProductSortStatusEnum.*values*();  
 **for** (ProductSortStatusEnum obj : values) {  
 **if** (number.intValue() == obj.getNumber().intValue()) {  
 **return** obj.field;  
 }  
 }  
 **return** "";  
 }  
  
 **public** Integer getNumber() {  
 **return** number;  
 }  
  
 **public void** setNumber(Integer number) {  
 **this**.number = number;  
 }  
  
 **public** String getField() {  
 **return** field;  
 }  
  
 **public void** setField(String field) {  
 **this**.field = field;  
 }  
}

RefundStatusEnum：退款状态枚举类

@ApiModel(description = "退款状态枚举类")  
**public enum** RefundStatusEnum {  
 *UNREFUND*(1,"退款中"),  
 *REFUND*(2,"已退款"),  
 ;  
  
 **private** Integer status;  
 **private** String name;  
  
 RefundStatusEnum(Integer status, String name) {  
 **this**.status = status;  
 **this**.name = name;  
 }  
  
 **public static** String getStatusNameByStatus(Integer status){  
 RefundStatusEnum[] values = RefundStatusEnum.*values*();  
 **for** (RefundStatusEnum obj : values) {  
 **if** (status.intValue() == obj.getStatus().intValue()){  
 **return** obj.getName();  
 }  
 }  
 **return** "";  
 }  
  
 **public** Integer getStatus() {  
 **return** status;  
 }  
  
 **public void** setStatus(Integer status) {  
 **this**.status = status;  
 }  
  
 **public** String getName() {  
 **return** name;  
 }  
  
 **public void** setName(String name) {  
 **this**.name = name;  
 }  
}

RegularEnum：常用正则表达式枚举类

@ApiModel(description = "常用正则表达式实例枚举")  
**public enum** RegularEnum {  
 *USER\_NAME\_LOGIN*("^[a-zA-Z0-9\_]{3,15}$"), // 用户名登录  
 *USER\_EMAIL\_LOGIN*("^\\w+([-+.]\\w+)\*@\\w+([-.]\\w+)\*\\.\\w+([-.]\\w+)\*$"), // 用户邮箱登录  
 *USER\_PHONE\_LOGIN*("^(13[0-9]|14[5|7]|15[0|1|2|3|5|6|7|8|9]|18[0|1|2|3|5|6|7|8|9])\\d{8}$"), // 用户手机号登陆  
 ;  
  
 **private** String regex;  
  
 RegularEnum(String regex) {  
 **this**.regex = regex;  
 }  
  
 **public** String getRegex() {  
 **return** regex;  
 }  
  
 **public void** setRegex(String regex) {  
 **this**.regex = regex;  
 }  
}

ReviewEnum：是否评价枚举类

@ApiModel(description = "订单项商品是否评价枚举类")  
**public enum** ReviewEnum {  
 *NOT\_REVIEW*(0), // 未评价  
 *REVIEW*(1), // 已评价  
 ;  
  
 **private** Integer status;  
  
 ReviewEnum(Integer status) {  
 **this**.status = status;  
 }  
  
 **public** Integer getStatus() {  
 **return** status;  
 }  
  
 **public void** setStatus(Integer status) {  
 **this**.status = status;  
 }  
}

SortTypeEnum：排序状态枚举类

@ApiModel(description = "排序状态枚举类")  
**public enum** SortTypeEnum {  
 *ASC*(0, "asc"), // 升序  
 *DESC*(1, "desc"), // 降序  
 ;  
  
 **private** Integer sort;  
 **private** String type;  
  
 SortTypeEnum(Integer sort, String type) {  
 **this**.sort = sort;  
 **this**.type = type;  
 }  
  
 // 根据排序代号取得排序类型  
 **public static** String getTypeBySort(Integer sort) {  
 **if** (sort == **null**) {  
 **return** "";  
 }  
 SortTypeEnum[] values = SortTypeEnum.*values*();  
 **for** (SortTypeEnum obj : values) {  
 **if** (sort.intValue() == obj.getSort().intValue()) {  
 **return** obj.type;  
 }  
 }  
 **return** "";  
 }  
  
 **public** Integer getSort() {  
 **return** sort;  
 }  
  
 **public void** setSort(Integer sort) {  
 **this**.sort = sort;  
 }  
  
 **public** String getType() {  
 **return** type;  
 }  
  
 **public void** setType(String type) {  
 **this**.type = type;  
 }  
}

TaskTypeEnum：任务类型枚举类

@ApiModel(description = "任务类型")  
**public enum** TaskTypeEnum {  
 *PAY\_OVERTIME*(1), // 商品支付超时  
 *REVIEW\_OVERTIME*(2), // 评价超时  
 ;  
  
 **private** Integer type;  
  
 TaskTypeEnum(Integer type) {  
 **this**.type = type;  
 }  
  
 **public** Integer getType() {  
 **return** type;  
 }  
  
 **public void** setType(Integer type) {  
 **this**.type = type;  
 }  
}

UserLockStatusEnum：用户锁定状态枚举类

**public enum** UserLockStatusEnum {  
 *LOCK*(0, "锁定"),  
 *UNLOCK*(1, "正常")  
 ;  
  
 **private** Integer status;  
 **private** String name;  
  
 UserLockStatusEnum(Integer status, String name) {  
 **this**.status = status;  
 **this**.name = name;  
 }  
  
 */\*\*  
 \* 根据状态获取状态名  
 \** ***@param*** *status  
 \** ***@return*** *\*/* **public static** String getStatusNameByStatus(Integer status) {  
 UserLockStatusEnum[] values = UserLockStatusEnum.*values*();  
 **for** (UserLockStatusEnum obj : values) {  
 **if** (status.intValue() == obj.getStatus().intValue()) {  
 **return** obj.getName();  
 }  
 }  
 **return** "";  
 }  
  
 **public** Integer getStatus() {  
 **return** status;  
 }  
  
 **public void** setStatus(Integer status) {  
 **this**.status = status;  
 }  
  
 **public** String getName() {  
 **return** name;  
 }  
  
 **public void** setName(String name) {  
 **this**.name = name;  
 }  
}

WeChatRedirectTypeEnum：微信回调类型枚举类，映射回调地址地址

@ApiModel(description = "微信回调类型枚举类")  
**public enum** WeChatRedirectTypeEnum {  
 *LOGIN*(0), // 微信登录  
 *BINDING*(1), // 微信绑定  
 ;  
  
 **private** Integer type;  
  
 WeChatRedirectTypeEnum(Integer type) {  
 **this**.type = type;  
 }  
  
 **public** Integer getType() {  
 **return** type;  
 }  
  
 **public void** setType(Integer type) {  
 **this**.type = type;  
 }  
}

#### model

##### base

BaseEntity

@Data  
@ApiModel(description = "mysql实体类的基类")  
**public class** BaseEntity **implements** Serializable {  
  
 @ApiModelProperty(value = "id")  
 @TableId(type = IdType.*ASSIGN\_ID*)  
 **private** String id;  
  
 @ApiModelProperty(value = "创建时间")  
 @JsonFormat(pattern="yyyy-MM-dd HH:mm:ss",timezone="GMT+8")  
 @TableField(value = "create\_time")  
 **private** Date createTime;  
  
 @ApiModelProperty(value = "更新时间")  
 @JsonFormat(pattern="yyyy-MM-dd HH:mm:ss",timezone="GMT+8")  
 @TableField(value = "update\_time")  
 **private** Date updateTime;  
  
 @ApiModelProperty(value = "逻辑删除(1:已删除，0:未删除)")  
 @TableLogic // 逻辑删除  
 @TableField(value = "is\_deleted")  
 **private** Integer isDeleted;  
  
 @ApiModelProperty(value = "其他参数")  
 @TableField(exist = **false**) // 让mp忽略这个不属于数据库中的补充参数  
 **private** Map<String, Object> params = **new** HashMap<>();  
  
}

##### category

CategoryInfo

@Data  
@ApiModel(description = "分类信息表")  
@TableName("category\_info")  
**public class** CategoryInfo **extends** BaseEntity {  
 @ApiModelProperty(value = "序列化版本")  
 **private static final long** *serialVersionUID* = 1L;  
  
 @ApiModelProperty(value = "分类名称")  
 @TableField("name")  
 **private** String name;  
  
 @ApiModelProperty(value = "分类图片url地址")  
 @TableField("image\_url")  
 **private** String imageUrl;  
}

Property

@Data  
@ApiModel(description = "分类属性名")  
@TableName("property")  
**public class** Property **extends** BaseEntity {  
 @ApiModelProperty(value = "序列化版本")  
 **private static final long** *serialVersionUID* = 1L;  
  
 @ApiModelProperty(value = "分类id，逻辑外键")  
 @TableField("category\_id")  
 **private** String categoryId;  
  
 @ApiModelProperty(value = "分类属性名称")  
 @TableField("name")  
 **private** String name;  
}

##### order

OrderInfo

@Data  
@ApiModel(description = "订单信息表")  
@TableName("order\_info")  
**public class** OrderInfo **extends** BaseEntity {  
 @ApiModelProperty(value = "序列化版本")  
 **private static final long** *serialVersionUID* = 1L;  
  
 @ApiModelProperty(value = "用户id，逻辑外键")  
 @TableField("user\_id")  
 **private** String userId;  
  
 @ApiModelProperty(value = "商家订单号")  
 @TableField("out\_trade\_no")  
 **private** String outTradeNo;  
  
 @ApiModelProperty(value = "收货地址")  
 @TableField("address")  
 **private** String address;  
  
 @ApiModelProperty(value = "邮政编码")  
 @TableField("post")  
 **private** String post;  
  
 @ApiModelProperty(value = "收件人")  
 @TableField("receiver")  
 **private** String receiver;  
  
 @ApiModelProperty(value = "收件人手机号")  
 @TableField("mobile")  
 **private** String mobile;  
   
 @ApiModelProperty(value = "支付金额")  
 @TableField("amount")  
 **private** BigDecimal amount;  
  
 @ApiModelProperty(value = "用户留言")  
 @TableField("user\_message")  
 **private** String userMessage;  
  
 @ApiModelProperty(value = "订单状态(0:待付款 1:待发货 2:待收货 3:待评价)")  
 @TableField("order\_status")  
 **private** Integer orderStatus;  
  
 @ApiModelProperty(value = "订单生成日期") // 记录生成日期在数据库中用的timestamp，订单创建日期用的datetime  
 @TableField("create\_date")  
 @JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss", timezone="GMT+8")  
 **private** Date createDate;  
  
 @ApiModelProperty(value = "买家支付日期")  
 @TableField("pay\_date")  
 @JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss", timezone="GMT+8")  
 **private** Date payDate;  
  
 @ApiModelProperty(value = "卖家发货日期")  
 @TableField("delivery\_date")  
 @JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss", timezone="GMT+8")  
 **private** Date deliveryDate;  
  
 @ApiModelProperty(value = "买家确认收货日期")  
 @TableField("confirm\_date")  
 @JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss", timezone="GMT+8")  
 **private** Date confirmDate;  
}

OrderItem

@Data  
@ApiModel(description = "订单项（购物车）表")  
@TableName("order\_item")  
**public class** OrderItem **extends** BaseEntity {  
 @ApiModelProperty(value = "序列化版本")  
 **private static final long** *serialVersionUID* = 1L;  
  
 @ApiModelProperty(value = "订单id，逻辑外键")  
 @TableField("order\_id")  
 **private** String orderId;  
  
 @ApiModelProperty(value = "商品id，逻辑外键")  
 @TableField("product\_id")  
 **private** String productId;  
  
 @ApiModelProperty(value = "用户id，逻辑外键")  
 @TableField("user\_id")  
 **private** String userId;  
  
 @ApiModelProperty(value = "商品数量")  
 @TableField("number")  
 **private** Integer number;  
  
 @ApiModelProperty(value = "是否评价(0:未评价 1:已评价)")  
 @TableField("is\_review")  
 **private** Integer isReview;  
}

PaymentInfo

@Data  
@ApiModel(description = "订单支付信息表")  
@TableName("payment\_info")  
**public class** PaymentInfo **extends** BaseEntity {  
 @ApiModelProperty(value = "序列化版本")  
 **private static final long** *serialVersionUID* = 1L;  
  
 @ApiModelProperty(value = "对外业务编号")  
 @TableField("out\_trade\_no")  
 **private** String outTradeNo;  
  
 @ApiModelProperty(value = "订单编号")  
 @TableField("order\_id")  
 **private** String orderId;  
  
 @ApiModelProperty(value = "支付类型（微信 支付宝）")  
 @TableField("payment\_type")  
 **private** Integer paymentType;  
  
 @ApiModelProperty(value = "交易编号")  
 @TableField("trade\_no")  
 **private** String tradeNo;  
  
 @ApiModelProperty(value = "支付金额")  
 @TableField("total\_amount")  
 **private** BigDecimal totalAmount;  
  
 @ApiModelProperty(value = "交易内容")  
 @TableField("subject")  
 **private** String subject;  
  
 @ApiModelProperty(value = "支付状态")  
 @TableField("payment\_status")  
 **private** Integer paymentStatus;  
  
 @ApiModelProperty(value = "回调时间")  
 @TableField("callback\_time")  
 **private** Date callbackTime;  
  
 @ApiModelProperty(value = "回调信息")  
 @TableField("callback\_content")  
 **private** String callbackContent;  
}

RefundInfo

@Data  
@ApiModel(description = "退款记录表")  
@TableName("refund\_info")  
**public class** RefundInfo **extends** BaseEntity {  
 @ApiModelProperty(value = "序列化版本")  
 **private static final long** *serialVersionUID* = 1L;  
  
 @ApiModelProperty(value = "对外业务编号")  
 @TableField("out\_trade\_no")  
 **private** String outTradeNo;  
  
 @ApiModelProperty(value = "订单编号")  
 @TableField("order\_id")  
 **private** String orderId;  
  
 @ApiModelProperty(value = "支付类型（微信 支付宝）")  
 @TableField("payment\_type")  
 **private** Integer paymentType;  
  
 @ApiModelProperty(value = "交易编号")  
 @TableField("trade\_no")  
 **private** String tradeNo;  
  
 @ApiModelProperty(value = "退款金额")  
 @TableField("total\_amount")  
 **private** BigDecimal totalAmount;  
  
 @ApiModelProperty(value = "交易内容")  
 @TableField("subject")  
 **private** String subject;  
  
 @ApiModelProperty(value = "退款状态")  
 @TableField("refund\_status")  
 **private** Integer refundStatus;  
  
 @ApiModelProperty(value = "回调时间")  
 @TableField("callback\_time")  
 **private** Date callbackTime;  
  
 @ApiModelProperty(value = "回调信息")  
 @TableField("callback\_content")  
 **private** String callbackContent;  
}

##### product

ProductImage

@Data  
@ApiModel(description = "商品图片表")  
@TableName("product\_image")  
**public class** ProductImage **extends** BaseEntity {  
 @ApiModelProperty(value = "序列化版本")  
 **private static final long** *serialVersionUID* = 1L;  
  
 @ApiModelProperty(value = "产品id，逻辑外键")  
 @TableField("product\_id")  
 **private** String productId;  
  
 @ApiModelProperty(value = "图片类型(0: 缩略图 1:详情图)")  
 @TableField("type")  
 **private** Integer type;  
  
 @ApiModelProperty(value = "图片的url地址")  
 @TableField("url")  
 **private** String url;  
}

ProductInfo

@Data  
@ApiModel(description = "商品信息表")  
@TableName("product\_info")  
**public class** ProductInfo **extends** BaseEntity {  
 @ApiModelProperty(value = "序列化版本")  
 **private static final long** *serialVersionUID* = 1L;  
  
 @ApiModelProperty(value = "分类id，逻辑外键")  
 @TableField("category\_id")  
 **private** String categoryId;  
  
 @ApiModelProperty(value = "商品名称")  
 @TableField("name")  
 **private** String name;  
  
 @ApiModelProperty(value = "商品小标题")  
 @TableField("sub\_title")  
 **private** String subTitle;  
  
 @ApiModelProperty(value = "商品原价")  
 @TableField("orignal\_price")  
 **private** BigDecimal orignalPrice;  
  
 @ApiModelProperty(value = "商品实际售价")  
 @TableField("promote\_price")  
 **private** BigDecimal promotePrice;  
  
 @ApiModelProperty(value = "库存")  
 @TableField("stock")  
 **private** Long stock;  
  
 @ApiModelProperty(value = "总销售量")  
 @TableField("sales\_volume")  
 **private** Long salesVolume;  
  
 @ApiModelProperty(value = "月销售量")  
 @TableField("monthly\_sales")  
 **private** Long monthlySales;  
}

PropertyValue

@Data  
@ApiModel(description = "商品属性值")  
@TableName("property\_value")  
**public class** PropertyValue **extends** BaseEntity {  
 @ApiModelProperty(value = "序列化版本")  
 **private static final long** *serialVersionUID* = 1L;  
  
 @ApiModelProperty(value = "产品id，逻辑外键")  
 @TableField("product\_id")  
 **private** String productId;  
  
 @ApiModelProperty(value = "分类属性id，逻辑外键")  
 @TableField("property\_id")  
 **private** String propertyId;  
  
 @ApiModelProperty(value = "商品属性值")  
 @TableField("value")  
 **private** String value;  
}

Review

@Data  
@ApiModel(description = "商品评价表")  
@TableName("review")  
**public class** Review **extends** BaseEntity {  
 @ApiModelProperty(value = "序列化版本")  
 **private static final long** *serialVersionUID* = 1L;  
  
 @ApiModelProperty(value = "用户id，逻辑外键")  
 @TableField("user\_id")  
 **private** String userId;  
  
 @ApiModelProperty(value = "商品id，逻辑外键")  
 @TableField("product\_id")  
 **private** String productId;  
  
 @ApiModelProperty(value = "评价内容")  
 @TableField("content")  
 **private** String content;  
}

##### user

UserInfo

@Data  
@ApiModel(description = "用户信息表")  
@TableName("user\_info")  
**public class** UserInfo **extends** BaseEntity {  
 @ApiModelProperty(value = "序列化版本")  
 **private static final long** *serialVersionUID* = 1L;  
  
 @ApiModelProperty(value = "用户密码")  
 @TableField("password")  
 **private** String password;  
  
 @ApiModelProperty(value = "微信号")  
 @TableField("openid")  
 **private** String openid;  
  
 @ApiModelProperty(value = "用户昵称")  
 @TableField("nick\_name")  
 **private** String nickName;  
  
 @ApiModelProperty(value = "用户名")  
 @TableField("name")  
 **private** String name;  
  
 @ApiModelProperty(value = "用户手机号")  
 @TableField("phone")  
 **private** String phone;  
  
 @ApiModelProperty(value = "用户邮箱地址")  
 @TableField("email")  
 **private** String email;  
  
 @ApiModelProperty(value = "用户头像url")  
 @TableField("avatar")  
 **private** String avatar;  
  
 @ApiModelProperty(value = "证件类型：身份证，永久居留证等")  
 @TableField("certificates\_type")  
 **private** String certificatesType;  
  
 @ApiModelProperty(value = "证件编号")  
 @TableField("certificates\_no")  
 **private** String certificatesNo;  
  
 @ApiModelProperty(value = "证件图片url地址")  
 @TableField("certificates\_url")  
 **private** String certificatesUrl;  
  
 @ApiModelProperty(value = "认证状态（0：未认证 1：认证中 2：认证成功 -1：认证失败）")  
 @TableField("auth\_status")  
 **private** Integer authStatus;  
  
 @ApiModelProperty(value = "锁定状态（0：锁定 1：正常）")  
 @TableField("status")  
 **private** Integer status;  
}

UserLoginRecord

@Data  
@ApiModel(description = "用户登录记录表")  
@TableName("user\_login\_record")  
**public class** UserLoginRecord **extends** BaseEntity {  
 @ApiModelProperty(value = "序列化版本")  
 **private static final long** *serialVersionUID* = 1L;  
  
 @ApiModelProperty(value = "用户id，逻辑外键")  
 @TableField("user\_id")  
 **private** String userId;  
  
 @ApiModelProperty(value = "用户ip记录")  
 @TableField("ip")  
 **private** String ip;  
}

#### vo

##### category

CategoryQueryVo：分类查询vo

@Data  
@ApiModel(description = "分类查询vo类")  
**public class** CategoryQueryVo {  
  
 @ApiModelProperty(value = "分类名称")  
 **private** String name;  
  
 @ApiModelProperty(value = "创建日期开始")  
 @DateTimeFormat(pattern = "yyyy-MM-dd")  
 **private** Date createTimeBegin;  
  
 @ApiModelProperty(value = "创建日期结束")  
 @DateTimeFormat(pattern = "yyyy-MM-dd")  
 **private** Date createTimeEnd;  
}

PropertyQueryVo：属性查询vo

@Data  
@ApiModel(description = "属性查询vo类")  
**public class** PropertyQueryVo {  
 @ApiModelProperty(value = "分类id")  
 **private** String categoryId;  
  
 @ApiModelProperty(value = "属性名称")  
 **private** String name;  
  
 @ApiModelProperty(value = "创建日期开始")  
 @DateTimeFormat(pattern = "yyyy-MM-dd")  
 **private** Date createTimeBegin;  
  
 @ApiModelProperty(value = "创建日期结束")  
 @DateTimeFormat(pattern = "yyyy-MM-dd")  
 **private** Date createTimeEnd;  
}

##### front

ProductReviewReturnVo

@Data  
@ApiModel(description = "商品评价返回vo类")  
**public class** ProductReviewReturnVo {  
 @ApiModelProperty(value = "评价id")  
 **private** String id;  
  
 @ApiModelProperty(value = "评价内容")  
 **private** String content;  
  
 @ApiModelProperty(value = "用户id")  
 **private** String userId;  
  
 @ApiModelProperty(value = "匿名用户名")  
 **private** String anonymity;  
  
 @ApiModelProperty(value = "用户名")  
 **private** String username;  
  
 @ApiModelProperty(value = "评价日期")  
 @JsonFormat(pattern = "yyyy-MM-dd")  
 **private** Date createTime;  
}

EmailCodeVo

@Data  
@ApiModel(description = "邮箱验证码vo类")  
**public class** EmailCodeVo {  
 @ApiModelProperty("邮箱地址")  
 **private** String email;  
  
 @ApiModelProperty("验证码类型")  
 **private** Integer type;  
}

UserInfoVo

@Data  
@ApiModel(description = "前端页头用户基本信息")  
**public class** UserInfoVo {  
 @ApiModelProperty("用户id")  
 **private** String id;  
  
 @ApiModelProperty("用户名")  
 **private** String name;  
  
 @ApiModelProperty("购物车商品数量")  
 **private** Integer productNumber;  
}

##### order

OrderQueryVo：订单查询vo

@Data  
@ApiModel(description = "订单查询vo类")  
**public class** OrderQueryVo {  
 @ApiModelProperty(value = "订单编号")  
 **private** String outTradeNo;  
  
 @ApiModelProperty(value = "用户id")  
 **private** String userId;  
  
 @ApiModelProperty(value = "订单状态")  
 **private** Integer orderStatus;  
  
 @ApiModelProperty(value = "创建日期开始")  
 @DateTimeFormat(pattern = "yyyy-MM-dd")  
 **private** Date createDateBegin;  
  
 @ApiModelProperty(value = "创建日期结束")  
 @DateTimeFormat(pattern = "yyyy-MM-dd")  
 **private** Date createDateEnd;  
}

SettlementVo：下单vo类

@Data  
@ApiModel(description = "下单vo类")  
**public class** SettlementVo {  
 @ApiModelProperty("收货地址")  
 **private** String address;  
  
 @ApiModelProperty("邮政编码")  
 **private** String post;  
  
 @ApiModelProperty("收货人姓名")  
 **private** String receiver;  
  
 @ApiModelProperty("用户留言")  
 **private** String userMessage;  
  
 @ApiModelProperty("手机号")  
 **private** String mobile;  
  
 @ApiModelProperty("订单项id集合")  
 **private** List<String> orderItemIdList;  
  
}

##### product

ProductImageReturnVo：产品图片返回vo

@Data  
@ApiModel(description = "商品图片返回到前端的vo")  
**public class** ProductImageReturnVo {  
 @ApiModelProperty(value = "id")  
 **private** String id;  
  
 @ApiModelProperty(value = "图片访问地址")  
 **private** String url;  
}

ProductInfoEeVo

@Data  
@ApiModel(description = "excel文档中的商品信息vo类")  
**public class** ProductInfoEeVo {  
 @ExcelProperty(value = "id", index = 0)  
 **private** String id;  
  
 @ExcelProperty(value = "分类id", index = 1)  
 **private** String categoryId;  
  
 @ExcelProperty(value = "商品名称", index = 2)  
 **private** String name;  
  
 @ExcelProperty(value = "商品小标题", index = 3)  
 **private** String subTitle;  
  
 @ExcelProperty(value = "商品原价", index = 4)  
 **private** BigDecimal orignalPrice;  
  
 @ExcelProperty(value = "商品实际售价", index = 5)  
 **private** BigDecimal promotePrice;  
  
 @ExcelProperty(value = "库存", index = 6)  
 **private** Long stock;  
}

ProductInfoFrontQueryVo：前台商品查询vo类

@Data  
@ApiModel(description = "前台系统商品查询vo类")  
**public class** ProductInfoFrontQueryVo {  
 @ApiModelProperty(value = "分类id")  
 **private** String categoryId;  
  
 @ApiModelProperty(value = "商品最低价格")  
 **private** BigDecimal lowPrice;  
  
 @ApiModelProperty(value = "商品最高价格")  
 **private** BigDecimal highPrice;  
  
 @ApiModelProperty(value = "排序字段") // 0:默认 1:评价数 2:新品 3:月成交量 4:价格  
 **private** Integer sortField;  
  
 @ApiModelProperty(value = "上一次排序方式")  
 **private** Integer lostSortField;  
  
 @ApiModelProperty(value = "排序方式")  
 **private** Integer sortType;  
}

ProductQueryVo：产品查询vo

@Data  
@ApiModel(description = "商品查询vo类")  
**public class** ProductQueryVo {  
 @ApiModelProperty(value = "分类id")  
 **private** String categoryId;  
  
 @ApiModelProperty(value = "商品名称 or 商品小标题")  
 **private** String keyword;  
  
 @ApiModelProperty(value = "商品最低价格")  
 **private** BigDecimal lowPrice;  
  
 @ApiModelProperty(value = "商品最高价格")  
 **private** BigDecimal highPrice;  
  
 @ApiModelProperty(value = "创建日期开始")  
 @DateTimeFormat(pattern = "yyyy-MM-dd")  
 **private** Date createTimeBegin;  
  
 @ApiModelProperty(value = "创建日期结束")  
 @DateTimeFormat(pattern = "yyyy-MM-dd")  
 **private** Date createTimeEnd;  
}

PropertyAndValueVo：属性名和属性值返回vo

@Data  
@ApiModel(description = "属性名和属性值的返回vo类")  
**public class** PropertyAndValueVo {  
 @ApiModelProperty(value = "属性值id")  
 **private** String propertyValueId;  
  
 @ApiModelProperty(value = "属性值")  
 **private** String propertyValue;  
  
 @ApiModelProperty(value = "属性名")  
 **private** String propertyName;  
}

ReviewVo：商品评价vo类

@Data  
@ApiModel(description = "评价vo类")  
**public class** ReviewVo {  
 @ApiModelProperty("订单项id")  
 **private** String orderItemId;  
   
 @ApiModelProperty("商品id")  
 **private** String productId;  
   
 @ApiModelProperty("评价内容")  
 **private** String content;  
}

##### user

UserLoginVo：用户登录vo类

@Data  
@ApiModel(description = "前台用户登录vo类")  
**public class** UserLoginVo {  
 @ApiModelProperty("用户登录账号（id 手机号 邮箱）")  
 **private** String account;  
  
 @ApiModelProperty("用户密码")  
 **private** String password;  
  
 @ApiModelProperty("邮箱验证码（6位）")  
 **private** String emailCode;  
}

UserQueryVo：用户查询查询vo

@Data  
@ApiModel(description = "用户查询vo类")  
**public class** UserQueryVo {  
 @ApiModelProperty(value = "用户id")  
 **private** String id;  
  
 @ApiModelProperty(value = "用户手机号 昵称 姓名")  
 **private** String keyword;  
  
 @ApiModelProperty(value = "用户认证状态")  
 **private** Integer authStatus;  
  
 @ApiModelProperty(value = "用户锁定状态")  
 **private** Integer status;  
}

UserRegisterVo：用户注册vo类

@Data  
@ApiModel(description = "用户注册vo类")  
**public class** UserRegisterVo {  
 @ApiModelProperty("用户名")  
 **private** String name;  
  
 @ApiModelProperty("用户密码")  
 **private** String password;  
  
 @ApiModelProperty("邮箱地址")  
 **private** String email;  
  
 @ApiModelProperty("验证码")  
 **private** String emailCode;  
}

### service依赖

<**parent**>  
 <**artifactId**>service</**artifactId**>  
 <**groupId**>com.study.tmall</**groupId**>  
 <**version**>1.0-SNAPSHOT</**version**>  
</**parent**>  
<**modelVersion**>4.0.0</**modelVersion**>  
  
<**artifactId**>service-notify</**artifactId**>  
  
<**dependencies**>  
 <**dependency**>  
 <**groupId**>com.study.tmall</**groupId**>  
 <**artifactId**>service\_order\_client</**artifactId**>  
 <**version**>1.0-SNAPSHOT</**version**>  
 </**dependency**>  
 <!-- 邮箱服务 -->  
 <**dependency**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-starter-mail</**artifactId**>  
 </**dependency**>  
  
 <**dependency**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-starter-thymeleaf</**artifactId**>  
 </**dependency**>  
 <!-- spring cloud stream -->  
 <**dependency**>  
 <**groupId**>org.springframework.cloud</**groupId**>  
 <**artifactId**>spring-cloud-starter-stream-rabbit</**artifactId**>  
 </**dependency**>  
 <**dependency**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-starter-test</**artifactId**>  
 </**dependency**>  
</**dependencies**>  
  
<**build**>  
 <!--静态文件编译过滤-->  
 <**resources**>  
 <**resource**>  
 <**directory**>src/main/resources</**directory**>  
 <**filtering**>false</**filtering**>  
 </**resource**>  
 <**resource**>  
 <**directory**>src/main/java</**directory**>  
 <**includes**>  
 <**include**>\*\*/\*.xml</**include**>  
 <**include**>\*\*/\*.properties</**include**>  
 </**includes**>  
 <**filtering**>false</**filtering**>  
 </**resource**>  
 </**resources**>  
</**build**>

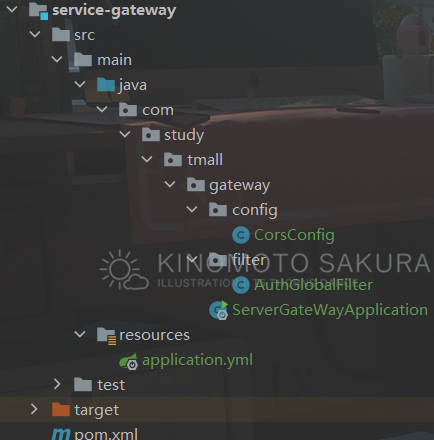
**server模块子包搭建见源码**

### service-client依赖

<**parent**>  
 <**artifactId**>tmall-parent</**artifactId**>  
 <**groupId**>com.study.tmall</**groupId**>  
 <**version**>1.0-SNAPSHOT</**version**>  
</**parent**>  
<**modelVersion**>4.0.0</**modelVersion**>  
  
<**artifactId**>service-client</**artifactId**>  
<**packaging**>pom</**packaging**>  
  
<**dependencies**>  
 <**dependency**>  
 <**groupId**>com.study.tmall</**groupId**>  
 <**artifactId**>common-util</**artifactId**>  
 <**version**>1.0-SNAPSHOT</**version**>  
 <**scope**>provided </**scope**>  
 </**dependency**>  
  
 <**dependency**>  
 <**groupId**>com.study.tmall</**groupId**>  
 <**artifactId**>model</**artifactId**>  
 <**version**>1.0-SNAPSHOT</**version**>  
 <**scope**>provided </**scope**>  
 </**dependency**>  
  
 <**dependency**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-starter-web</**artifactId**>  
 <**scope**>provided </**scope**>  
 </**dependency**>  
  
 <!-- 服务调用feign -->  
 <**dependency**>  
 <**groupId**>org.springframework.cloud</**groupId**>  
 <**artifactId**>spring-cloud-starter-openfeign</**artifactId**>  
 <**scope**>provided</**scope**>  
 </**dependency**>  
</**dependencies**>

**server-client模块子包搭建见《天猫后台管理后端开发手册》和《天猫前台管理后端开发手册》**

### service-gateway模块



#### pom.xml

<**parent**>  
 <**artifactId**>tmall-parent</**artifactId**>  
 <**groupId**>com.study.tmall</**groupId**>  
 <**version**>1.0-SNAPSHOT</**version**>  
</**parent**>  
<**modelVersion**>4.0.0</**modelVersion**>  
  
<**artifactId**>service-gateway</**artifactId**>  
  
<**dependencies**>  
 <**dependency**>  
 <**groupId**>com.study.tmall</**groupId**>  
 <**artifactId**>common-util</**artifactId**>  
 <**version**>1.0-SNAPSHOT</**version**>  
 </**dependency**>  
  
 <**dependency**>  
 <**groupId**>org.springframework.cloud</**groupId**>  
 <**artifactId**>spring-cloud-starter-gateway</**artifactId**>  
 </**dependency**>  
  
 <!-- 服务注册 -->  
 <**dependency**>  
 <**groupId**>com.alibaba.cloud</**groupId**>  
 <**artifactId**>spring-cloud-starter-alibaba-nacos-discovery</**artifactId**>  
 </**dependency**>  
</**dependencies**>

#### 启动类

@SpringBootApplication  
**public class** ServerGateWayApplication {  
 **public static void** main(String[] args) {  
 SpringApplication.*run*(ServerGateWayApplication.**class**, args);  
 }  
}

#### config

**CorsConfig**

@Configuration  
**public class** CorsConfig {  
 @Bean  
 **public** CorsWebFilter corsFilter() {  
 CorsConfiguration config = **new** CorsConfiguration();  
 config.addAllowedMethod("\*");  
 config.addAllowedOrigin("\*");  
 config.addAllowedHeader("\*");  
  
 UrlBasedCorsConfigurationSource source = **new** UrlBasedCorsConfigurationSource(**new** PathPatternParser());  
 source.registerCorsConfiguration("/\*\*", config);  
  
 **return new** CorsWebFilter(source);  
 }  
}

#### filter

**AuthGlobalFilter**

@Component  
**public class** AuthGlobalFilter **implements** GlobalFilter, Ordered {  
 // AntPathMatcher 其中match可以很方便的做url判断  
 **private** AntPathMatcher antPathMatcher = **new** AntPathMatcher();  
  
 @Override  
 **public** Mono<Void> filter(ServerWebExchange exchange, GatewayFilterChain chain) {  
 ServerHttpRequest request = exchange.getRequest();  
 String path = request.getURI().getPath();  
 System.*out*.println("==="+path);  
  
 //内部服务接口，不允许外部访问  
 **if**(antPathMatcher.match("/\*\*/inner/\*\*", path)) {  
 ServerHttpResponse response = exchange.getResponse();  
 **return** out(response, ResultCodeEnum.*PERMISSION*);  
 }  
  
 //api接口，异步请求，校验用户必须登录  
 **if**(antPathMatcher.match("/api/\*\*/auth/\*\*", path)) {  
 String userId = **this**.getUserId(request);  
 **if**(StringUtils.*isEmpty*(userId)) {  
 ServerHttpResponse response = exchange.getResponse();  
 **return** out(response, ResultCodeEnum.*LOGIN\_AUTH*);  
 }  
 }  
 **return** chain.filter(exchange);  
 }  
  
 @Override  
 **public int** getOrder() {  
 **return** 0;  
 }  
  
 */\*\*  
 \* api接口鉴权失败返回数据  
 \** ***@param*** *response  
 \** ***@return*** *\*/* **private** Mono<Void> out(ServerHttpResponse response, ResultCodeEnum resultCodeEnum) {  
 Result result = Result.*build*(**null**, resultCodeEnum);  
 **byte**[] bits = JSONObject.*toJSONString*(result).getBytes(StandardCharsets.*UTF\_8*); // 把返回的结果搞成json，再搞成UTF-8编码的byte数组  
 DataBuffer buffer = response.bufferFactory().wrap(bits); // 把byte数组打包进数据缓冲  
 // 指定编码，否则在浏览器中会中文乱码  
 response.getHeaders().add("Content-Type", "application/json;charset=UTF-8");  
 // Mono是响应式编程，层与层之间传递的对象，Mono.just就是初始化一个Mono对象  
 **return** response.writeWith(Mono.*just*(buffer));  
 }  
  
 */\*\*  
 \* 获取当前登录用户id  
 \** ***@param*** *request  
 \** ***@return*** *\*/* **private** String getUserId(ServerHttpRequest request) {  
 String token = "";  
 List<String> tokenList = request.getHeaders().get("token");  
 **if**(**null** != tokenList) {  
 token = tokenList.get(0);  
 }  
 **if**(!StringUtils.*isEmpty*(token)) {  
 **try** {  
 String userId = JwtHelper.*getUserId*(token);  
 **return** userId;  
 } **catch** (ExpiredJwtException e) {  
 System.*out*.println("token已过期");  
 **return null**; // 返回空  
 }  
  
 }  
 **return null**;  
 }  
}

## 后端业务功能

### 后台管理员系统

* + 分类管理
    - 分页条件显示分类
    - 批量删除分类
    - 删除分类
    - 上传分类图片
    - 添加分类
    - 编辑分类
    - 查询单个分类
    - 属性管理
* 分页条件显示属性
* 批量删除属性
* 删除属性
* 添加属性
* 编辑属性
  + - 产品管理
* 分页条件显示产品
* 批量删除产品
* 删除产品
* 添加产品
* excel批量导入产品
* 产品信息导出为excel
* 编辑产品
* 产品图片管理
* 显示产品图片
* 批量上传图片
* 批量删除图片
* 删除图片
* 产品属性设置
* 显示产品属性
* 异步更新产品属性值
  + 用户管理
    - 分页条件显示用户
    - 锁定\解锁 用户
    - 审批用户认证
  + 订单管理
    - 分页条件显示订单
    - 查看订单详情
    - 订单发货
  + 数据视图
* 按分类显示产品热销数据（做成统计图）

### 前台用户系统

* + 首页
    - 显示分类列表和热销商品
  + 分类页
    - 显示分类图片
    - 条件查询商品
  + 产品页
    - 显示分类图片（和分类页共用）
    - 查询商品信息
    - 分页查询用户对商品的评价
    - 模态框用户登录（和登录页共用）
    - 加入购物车
    - 立即购买
  + 搜索页
* 搜索商品
  + 登录页
* 发送验证码
* 账号登录
* 获取微信登录二维码
* 微信扫码回调
* 用户登录扫码状态轮询
  + 注册页
* 检测用户名是否被使用
* 检测邮箱是否被使用
* 发送验证码（与登录页共用，但是参数需传递此为注册验证码）
* 用户注册
* 用户邮箱绑定（如果微信扫码登录的微信号未绑定账号，则注册页实现绑定功能）
  + 页面头
* 根据token获取用户信息
  + 账户设置页
* 根据token获取用户详情信息（不同上的简略信息）
* 修改用户昵称
* 修改用户头像
* 密码修改
* 邮箱改绑
* 手机号绑定
* 手机号解绑
* 微信绑定
* 微信解绑
  + 购物车
* 获取订单项（包括订单项对应的商品信息）
* 删除订单项
* 更新订单项商品数量
  + 结算（下单）页
* 下单（把订单项关联到订单中，超时自动取消订单）
  + 支付页
* 创建native支付二维码（被取消的订单不能生成二维码）
* 支付结果轮询
* 一键支付（测试用）（被取消的订单不能支付）
  + 支付成功页
* 显示订单部分信息
  + 订单也
* 根据条件显示用户订单
* 付款（被取消的订单不能再支付）
* 发货（不做后端，前端实现页面跳转，发送发货通知邮件）
* 收货（发送签收通知邮件，且开启评论计时，超过时间的订单项不能再评论）
* 评价（不做后端，前端实现页面跳转）
  + 确认收货页
* 获取订单详情信息
* 用户确认收货
  + 交易成功页
* （只需前端显示个提示信息，无需从后端和获取数据）
  + 产品评价页
* 根据订单项id查询订单项和商品信息
* 添加评价
* 显示评价（如果该订单项已评价则显示评价，显示接口和产品页共用）