# 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 + 自己设计的前端系统

## 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 |
| 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（后续实现） |
| 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 |
| 前端前台模板 | nuxt |

## 后端项目工程搭建

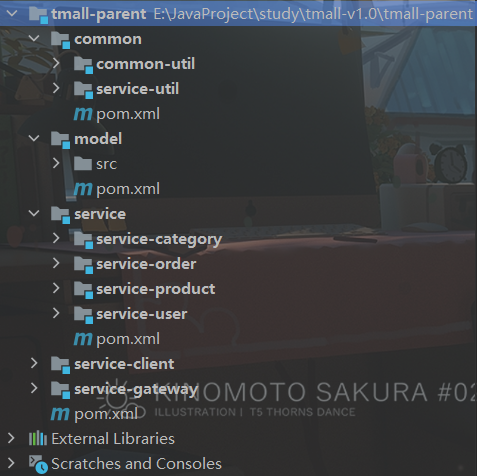
### 核心框架版本

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

<**modelVersion**>4.0.0</**modelVersion**>  
  
  
<**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**>  
 <!--<mysql.version>5.1.47</mysql.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**>  
</**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**>${druid.version}</**version**>  
 </**dependency**>  
 <!--mybatis-plus-->  
 <**dependency**>  
 <**groupId**>com.baomidou</**groupId**>  
 <**artifactId**>mybatis-plus-boot-starter</**artifactId**>  
 <**version**>${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**>  
 </**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**>  
</**dependencies**>

### common-util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**>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**>  
</**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*);  
 }  
  
}

**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"),  
  
 *LOGIN\_AUTH*(208, "未登陆"),  
 *PERMISSION*(209, "没有权限"),  
  
 *CODE\_ERROR*(210, "验证码错误"),  
 *LOGIN\_DISABLED\_ERROR*(212, "该用户已被禁用"),  
 *REGISTER\_MOBLE\_ERROR*(213, "手机号已被使用"),  
 *REGISTER\_EMAIL\_ERROR*(214, "邮箱已被使用"),  
 *LOGIN\_AURH*(215, "需要登录"),  
 *LOGIN\_ACL*(216, "没有权限"),  
  
 *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\_FAIL*(225, "取消订单失败"),  
 *CANCEL\_ORDER\_NO*(225, "不能取消预约"),  
 ;  
  
 **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

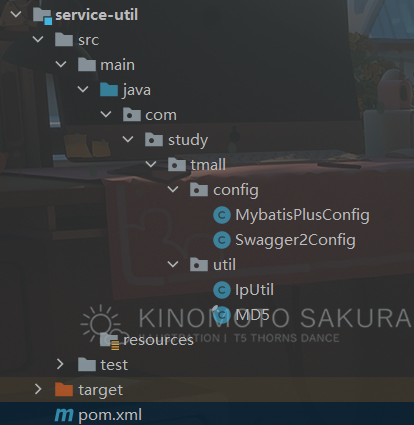
**JwtHelper**

**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  
  
 */\*\*  
 \* 根据参数生成token  
 \** ***@param*** *userId  
 \** ***@param*** *userName  
 \** ***@return*** *\*/* **public static** String createToken(String userId, String userName) {  
 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("userName", userName)  
 .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) {  
 **if**(StringUtils.*isEmpty*(token)) **return** "";  
 Jws<Claims> claimsJws = Jwts.*parser*().setSigningKey(*tokenSignKey*).parseClaimsJws(token);  
 Claims claims = claimsJws.getBody();  
 **return** (String) claims.get("userId");  
 }  
  
 */\*\*  
 \* 根据token取出userName  
 \** ***@param*** *token  
 \** ***@return*** *\*/* **public static** String getUserName(String token) {  
 **if**(StringUtils.*isEmpty*(token)) **return** "";  
 Jws<Claims> claimsJws = Jwts.*parser*().setSigningKey(*tokenSignKey*).parseClaimsJws(token);  
 Claims claims = claimsJws.getBody();  
 **return** (String)claims.get("userName");  
 }  
}

**TokenUtil**

**public class** TokenUtil {  
  
 */\*\*  
 \* 获取token中的用户名  
 \** ***@param*** *request  
 \** ***@return*** *\*/* **public static** String getUserName(HttpServletRequest request){  
 // 获取请求头中的token  
 String token = request.getHeader("token");  
 // 使用JwtHelper工具类获取token中的用户名  
 **return** JwtHelper.*getUserName*(token);  
 }  
  
 */\*\*  
 \* 获取token中的用户ID  
 \** ***@param*** *request  
 \** ***@return*** *\*/* **public static** String getUserId(HttpServletRequest request){  
 // 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  
 **return** JwtHelper.*getUserId*(token);  
 }  
}

### service-util模块



#### pom.xml

暂无添加

#### 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;  
 }  
}

**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

**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加盐转码  
 \** ***@param*** *strSrc  
 \** ***@return*** *\*/* **public static** String encrypt(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];  
 }  
 **return new** String(chars);  
 } **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**>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**>  
</**dependencies**>

#### enums

**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;  
 }  
}

**ImageTypeEnum：图片类型枚举类**

@ApiModel(description = "图片类型枚举类")  
**public enum** ImageTypeEnum {  
 *THUMBNAIL*(0, "缩略图"),  
 *DETAILS\_FIGURE*(1, "详情图"),  
 ;  
  
 **private** Integer type;  
 **private** String name;  
  
 ImageTypeEnum(Integer type, String name){  
 **this**.type = type;  
 **this**.name = name;  
 }  
  
 **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;  
 }  
}

**OrderStatusEnum：订单状态枚举类**

@ApiModel(description = "订单状态枚举类")  
**public enum** OrderStatusEnum {  
 *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")  
 ;  
  
 **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** "";  
 }  
  
 **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 {  
 *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;  
 }  
}

**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;  
 }  
}

#### 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)")  
 @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 = "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")  
 **private** Date createDate;  
  
 @ApiModelProperty(value = "买家支付日期")  
 @TableField("pay\_date")  
 **private** Date payDate;  
  
 @ApiModelProperty(value = "卖家发货日期")  
 @TableField("delivery\_date")  
 **private** Date deliveryDate;  
  
 @ApiModelProperty(value = "买家确认收货日期")  
 @TableField("confirm\_date")  
 **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;  
}

**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** Long 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** Long 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 promote\_price;  
  
 @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 = "微信号")  
 @TableField("openid")  
 **private** String openid;  
  
 @ApiModelProperty(value = "用户昵称")  
 @TableField("nick\_name")  
 **private** String nickName;  
  
 @ApiModelProperty(value = "用户认证姓名")  
 @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 = "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 = "用户登录限制表")  
@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

##### 用到的时候再写入文档

### service依赖

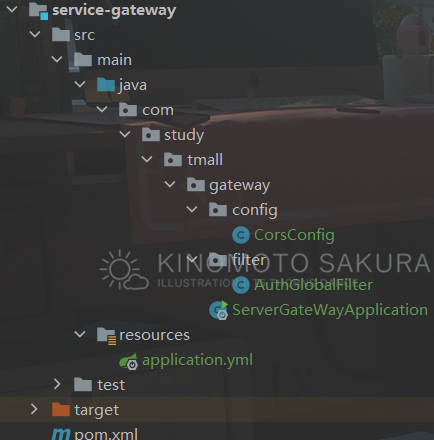
<**parent**>  
 <**artifactId**>tmall-parent</**artifactId**>  
 <**groupId**>com.study.tmall</**groupId**>  
 <**version**>1.0-SNAPSHOT</**version**>  
</**parent**>  
<**modelVersion**>4.0.0</**modelVersion**>  
  
<**artifactId**>service</**artifactId**>  
<**packaging**>pom</**packaging**>  
  
<**dependencies**>  
 <**dependency**>  
 <**groupId**>com.study.tmall</**groupId**>  
 <**artifactId**>common-util</**artifactId**>  
 <**version**>1.0-SNAPSHOT</**version**>  
 </**dependency**>  
 <**dependency**>  
 <**groupId**>com.study.tmall</**groupId**>  
 <**artifactId**>service-util</**artifactId**>  
 <**version**>1.0-SNAPSHOT</**version**>  
 </**dependency**>  
 <**dependency**>  
 <**groupId**>com.study.tmall</**groupId**>  
 <**artifactId**>model</**artifactId**>  
 <**version**>1.0-SNAPSHOT</**version**>  
 </**dependency**>  
 <!--web-->  
 <**dependency**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-starter-web</**artifactId**>  
 </**dependency**>  
 <!--mysql-->  
 <**dependency**>  
 <**groupId**>mysql</**groupId**>  
 <**artifactId**>mysql-connector-java</**artifactId**>  
 </**dependency**>  
 <!--mybatis-plus-->  
 <**dependency**>  
 <**groupId**>com.baomidou</**groupId**>  
 <**artifactId**>mybatis-plus-boot-starter</**artifactId**>  
 </**dependency**>  
 <!-- 服务调用feign -->  
 <**dependency**>  
 <**groupId**>org.springframework.cloud</**groupId**>  
 <**artifactId**>spring-cloud-starter-openfeign</**artifactId**>  
 </**dependency**>  
 <!-- 服务注册 -->  
 <!--<dependency>  
 <groupId>com.alibaba.cloud</groupId>  
 <artifactId>spring-cloud-starter-alibaba-nacos-discovery</artifactId>  
 </dependency>-->  
</**dependencies**>  
  
<**build**>  
 <**plugins**>  
 <**plugin**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-maven-plugin</**artifactId**>  
 </**plugin**>  
 </**plugins**>  
 <**resources**>  
 <**resource**>  
 <**directory**>src/main/java</**directory**>  
 <**includes**>  
 <**include**>\*\*/\*.yml</**include**>  
 <**include**>\*\*/\*.properties</**include**>  
 <**include**>\*\*/\*.xml</**include**>  
 </**includes**>  
 <**filtering**>false</**filtering**>  
 </**resource**>  
 <**resource**>  
 <**directory**>src/main/resources</**directory**>  
 <**includes**> <**include**>\*\*/\*.yml</**include**>  
 <**include**>\*\*/\*.properties</**include**>  
 <**include**>\*\*/\*.xml</**include**>  
 </**includes**>  
 <**filtering**>false</**filtering**>  
 </**resource**>  
 </**resources**>  
</**build**>

**server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-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**>

### 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);  
 }  
}

**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)) {  
 **return** JwtHelper.*getUserId*(token);  
 }  
 **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
    - 添加分类
    - 编辑分类
    - 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
* 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）