## 淘气商城 详细设计

# 一、业务术语

|  |  |  |
| --- | --- | --- |
| 序号 | 术语 | 备注 |
| 01 | b2c | Business-to-Customer的缩写，意为“商对客”，是电子商务的一种模式，通常指的是直接面向消费者销售产品和服务商业零售模式。 |
| 02 | taoqi | 本电商项目的产品名 |
| 03 | Admin | 系统管理员 |
| 04 | User | 会员 or 用户 |
| 05 | Shop | 店家 |

# 二、编码命名规范

|  |  |  |
| --- | --- | --- |
| 序号 | 包、类、接口名 | 备注 |
| 01 | com.taoqi.xx | 项目包根命名 |
| 02 | User | 用户实体 |
| 03 | Product | 产品实体 |
| 04 | Role | 角色实体 |
| 05 | Spread | 推广实体 |
| 06 | Evaluate | 评论实体 |
| 07 | Type | 分类实体 |
| 08 | Order | 订单实体 |
| 09 | Shop | 店铺实体 |
| 10 | Apply | 申请实体 |
| 11 | XxxController | 系列控制器 |
| 12 | XxxService | 系列业务逻辑处理器 |
| 13 | XxxMapper | 系列持久层ORM映射器 |
| 14 | XxxUtil | 系列工具类 |
| 15 | XxxConfig | 系列配置类 |
| 16 | XxxException | 系列自定义异常 |
| 17 | XxxVO | 系列与视图关联的实体类 |

# 三、数据库字典

### 1.t\_user用户表

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 字段名 | 类型 | 长度 | 默认 | 主键？ | 非空？ | 约束？ | 外键？ | 描述 |
| id | int | 11 |  | ✔ | ✔ | 唯一 |  | 自增长，主键 |
| nick\_name | varchar | 55 |  |  | ✔ |  |  | 用户昵称 |
| user\_name | varchar | 55 |  |  | ✔ | 唯一 |  | 用户名 |
| pass\_word | char | 64 |  |  | ✔ |  |  | 用户密码 |
| salt | char | 20 |  |  | ✔ |  |  | 密码盐 |
| imgsrc | varchar | 255 |  |  | ✔ |  |  | 头像地址 |
| phone | char | 11 |  |  | ✔ |  |  | 手机号码 |
| address | varchar | 255 |  |  |  |  |  | 收货地址 |
| stat | int | 1 |  |  | ✔ | 0-2 |  | 用户标识 |

### t\_role角色表

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 字段名 | 类型 | 长度 | 默认 | 主键？ | 非空？ | 约束？ | 外键？ | 描述 |
| id | int | 11 |  | ✔ | ✔ | 唯一 |  | 自增长，主键 |
| role | int | 11 |  |  | ✔ | 唯一 |  | 角色名 |

### t\_auth权限表

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 字段名 | 类型 | 长度 | 默认 | 主键？ | 非空？ | 约束？ | 外键？ | 描述 |
| id | int | 11 |  | ✔ | ✔ | 唯一 |  | 自增长，主键 |
| user\_id | int | 11 |  |  | ✔ |  | ✔ | 用户id，外键 |
| role\_id | int | 11 |  |  | ✔ |  | ✔ | 角色id，外键 |

### t\_evaluate评论表

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 字段名 | 类型 | 长度 | 默认 | 主键？ | 非空？ | 约束？ | 外键？ | 描述 |
| id | int | 11 |  | ✔ | ✔ | 唯一 |  | 自增长，评论编号 |
| user\_id | int | 11 |  |  | ✔ |  | ✔ | 用户id |
| order\_id | int | 11 |  |  | ✔ |  | ✔ | 订单id |
| reply\_id | int | 11 |  |  | ✔ |  |  | 回复id |
| star | int | 1 |  |  |  | 0-5 |  | 评分 |
| info | varchar | 255 |  |  | ✔ |  |  | 评论/回复内容 |

### t\_product商品表

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 字段名 | 类型 | 长度 | 默认 | 主键 | 非空 | 约束 | 外键 | 描述 |
| id | int | 11 |  | **√** | **√** | 唯一 |  | 自增长，商品编号 |
| typeid | int | 11 |  |  | **√** |  | **√** | 值对应分类表t\_type |
| shopid | int | 11 |  |  | **√** |  | **√** | 值对应店铺表t\_shop |
| imgsrc | varchar | 255 |  |  | **√** |  |  | 商品图片地址 |
| name | varchar | 55 |  |  | **√** |  |  | 商品的名称 |
| info | varchar | 255 |  |  | **√** |  |  | 商品的描述信息 |
| price | float |  |  |  | **√** |  |  | 商品的价格 |
| stock | int | 11 |  |  | **√** |  |  | 商品在店铺的库存 |

### t\_shop店铺表

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 字段名 | 类型 | 长度 | 默认 | 主键 | 非空 | 约束 | 外键 | 描述 |
| id | int | 11 |  | **√** | **√** | 唯一 |  | 自增长，店铺编号 |
| name | varchar | 255 |  |  | **√** |  |  | 店铺名称 |
| user\_id | int | 11 |  |  | **√** |  |  | 店主名字 |
| info | varchar | 255 |  |  | **√** |  |  | 店铺的描述信息 |
| time | timestamp |  | now |  | **√** |  |  | 店铺的创建时间 |

### t\_car购物车表

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 字段名 | 类型 | 长度 | 默认 | 主键 | 非空 | 约束 | 外键 | 描述 |
| id | int | 11 |  | **√** | **√** | 唯一 |  | 自增长，购物车编号 |
| user\_id | int | 11 |  |  | **√** |  |  | 外键，用户编号 |
| order\_id | int | 11 |  |  | **√** |  |  | 名外键，订单编号 |
| price | float |  |  |  | **√** |  |  | 商品价格 |
| count | int | 11 | now |  | **√** |  |  | 商品数量 |
| time | timestamp |  | now |  |  |  |  | 加入时间 |

### t\_type分类表

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 字段名 | 类型 | 长度 | 默认 | 主键 | 非空 | 约束 | 外键 | 描述 |
| Id | int | 11 |  | **√** | **√** |  |  | 分类编号 |
| name | varchar | 55 |  |  | **√** |  |  | 分类名称 |
| parent | int | 11 |  |  | **√** |  |  | 父分类（0未顶层分类） |
| flag | int | 1 |  |  | **√** |  |  | 申请类型（0系统分类，其他则店铺自定义分类） |

### t\_spread推广表

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 字段名 | 类型 | 长度 | 默认 | 主键 | 非空 | 约束 | 外键 | 描述 |
| id | int | 11 |  | **√** | **√** | 唯一 |  | 自增长，推广竞价编号 |
| user\_id | int | 11 |  |  |  |  |  | 值对应用户表表t\_user |
| info | varchar | 255 |  |  |  |  |  | 推广信息，商家店铺链接 |
| imgSrc | varchar | 255 |  |  |  |  |  | 推广商家图片地址 |
| price | float |  |  |  |  |  |  | 推广竞价出价 |

### t\_order订单表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 类型 | 是否可空 | 中文名 | 描述 |
| id | int | no | 序号 | 自增长，订单编号 |
| userid | int | no | 用户编号 | 值对应用户表t\_user |
| productid | int | no | 产品编号 | 值对应产品表t\_product |
| count | int | no | 商品数量 | 商品的数量 |
| sum | double | no | 商品总价 | 商品的总价 |
| flag | int | no | 删除标记 | 用户是否已删除订单，默认为0，未删除 |
| stat | int | no | 处理标记 | 商品是否已发货，默认为0，未处理 |
| time | timestamp | no | 记录时间 | 记录的实时时间 |

# 接口设计

### 用户模块

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Url | 请求参数 | 返回值 | 逻辑概述 | 请求类型 | 访问权限 |
| user/reg | User对象 | 0或者1（插入数据影响行数） | 插入user数据 | Post | 无 |
| user/get | 分页Page、size | 获取用户信息列表 | 分页查询 | Get | Admin |
| user/get/{id} | {id} | 根据id获取用户 | 根据id查询 | Get | Admin |
| user/update | User对象 | 0或者1（插入数据影响行数） | 修改用户信息 | Post | User |
| user/delete | Userid | 0或者1（插入数据影响行数） | 删除用户 | Post | Admin |
| user/apply | Userid | 待受理，固定值 | 插入数据到商家申请表 | Post | User |

### 评论模块

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Url | 请求参数 | 返回值 | 逻辑概述 | 请求类型 | 访问权限 |
| evaluate/uget/{priductid} | productid | List<Evaluate> | 查询数据库，获取数据，将数据返回到前端 | Get | 无 |
| evaluate/sget/{orderid} | orderid | List<Evaluate> | 查询数据库，获取数据，将数据返回到前端 | Get | shop |
| evaluate/add | Evaluate | Int(0/1); 0:插入失败  1：插入成功 | 插入评论信息 | Post | User |
| evaluate/get | 无 | List<Evaluate> | 查询数据库，获取数据，将数据返回到前端 | Get | admin |
| evaluate/delete | evaluateid | Int(0/1); 0:删除失败  1：删除成功 | 删除评论/回复信息信息 | Post | admin |

### 购物车模块

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Url | 请求参数 | 返回值 | 逻辑概述 | 请求类型 | 访问权限 |
| Car/get/{userid} | userid | List<car> | 查询数据库，获取数据，将数据返回到前端 | Get | User |
| Car/add | Car | Int(0/1); 0:插入失败  1：插入成功 | 插入购物车信息 | Post | User |
| Car/update | Car | Int(0/1); 0:修改失败  1：修改成功 | 修改购物车信息 | Post | User |
| Car/delete | carid | Int(0/1); 0:删除失败  1：删除成功 | 删除购物车信息 | Post | User |

### 权限模块

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Url | 请求参数 | 返回值 | 逻辑概述 | 请求类型 | 访问权限 |
| auth/login | username,password字符串 | Null或jwt字符 | 查询数据库，获取User，如果成功则生成jwt返回 | Post | 无 |
| auth/check | jwt字符串 | Null或jwt对象 | 解析jwt字符，如果成功解析则生成新的jwt字符返回 | Post | User |

### 商品模块

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| url | 请求参数 | 返回值 | 逻辑概述 | 请求类型 | 访问权限 |
| product/get | name | Null或list<product  > | 根据商品名查询数据 | get | 无 |
| product/getlike | str商品模糊查询商品名 | null或list<product> | 根据str进行商品的模糊查询 | get | 无 |
| product/getbytype/{typeid} | {typeid} | Null或list<product  > | 根据分类查询商品列表 | get | 无 |
| product/getbyshop/{shopid} | {shopid} | Null或List<product> | 根据店铺查询数据库活动商品列表 | get | 无 |
| product/add | product对象 | fail或ok | 新增商品 | post | SHOP |
| product/delete/{id} | {id}商品id | fail或ok | 下架商品 | Post | SHOP |
| product/change | product对象 | fail或ok | 修改商品的信息 | Post | SHOP |

### 店铺模块

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| url | 请求参数 | 返回值 | 逻辑概述 | 请求类型 | 访问权限 |
| shop/add/{shop} | {shop} | fail/ok | 增加店铺 | Post | ADMIN |
| shop/delete/{id} | {id}店铺id | fail/ok | 删除店铺 | post | ADMIN |
| shop/change/{shop} | {shop} | fail/ok | 修改店铺信息 | Post | SHOP |
| shop/get/all | 无 | null/list<shop> | 获取所有店铺信息 | post | ADMIN |
| shop/get/{id} | {id}店铺id | null/shop | 获取当前店铺信息 | post | SHOP |

### 分类模块

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Url | 请求参数 | 返回值 | 逻辑概述 | 请求类型 | 访问权限 |
| type/classification | 无 | list<type> | 查询第一级分类 | Post | 无 |
| type/[subclassification](E:/%E6%9C%89%E9%81%93%E7%BF%BB%E8%AF%91/Dict/8.5.1.0/resultui/html/index.html" \l "/javascript:;)/{type：id} | type：id | list<type> | 查询第一级分类下面的第二级分类 | Post | 无 |
| type/addclassification/{type} | type | String | 插入分类 | Post | 商家/管理员 |
| type/del[subclassification](E:/%E6%9C%89%E9%81%93%E7%BF%BB%E8%AF%91/Dict/8.5.1.0/resultui/html/index.html" \l "/javascript:;)/{type：id} | type：id | String | 删除某个分类 | Post | 商家/管理员 |
| type/[merchant](E:/%E6%9C%89%E9%81%93%E7%BF%BB%E8%AF%91/Dict/8.5.1.0/resultui/html/index.html" \l "/javascript:;)/{Shop：id} | Shop：id | list<type> | 查询某个店铺分类 | Post | 无 |

### 推广模块

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| url | 请求参数 | 返回值 | 逻辑概述 | 请求类型 | 访问权限 |
| spread/getall | 无 | null或list<Spread  > | 获取参与竞价的所有商家信息 | Get | ADMIN |
| spread/getall | 无 | null或list<Spread  > | 获取参与竞价的前三的商家信息 | Get | ADMIN |
| spread/insert | userid,info,imgsrc,price字段 | fail或ok | 商家进行竞价 | Post | SHOP |
| spread/deletebyid/{id} | {id} | fail或ok | 删除序号为{id}的店铺竞价信息 | Post | ADMIN |
| spread/delete | 无 | fail或ok | 删除竞价价格前三的店铺竞价信息 | Post | SHOP |
| spread/deleteall | 无 | fail或ok | 删除所有的店铺竞价信息 | Post | SHOP |

### 订单模块

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| url | 请求参数 | 返回值 | 逻辑概述 | 请求类型 | 访问权限 |
| order/get/{id} | {id} | Null或list<order> | 根据订单编号查询所有数据 | Get | 无 |
| order/getOrder{id} | {id} | Null或list<order> | 根据订单编号查看所有未删除的所有数据 | Get | 无 |
| order/insertOrder{Order} | {Order} | void | 添加一条订单数据 | Post | 无 |
| order/delectOrder{id} | {id} | void | 用户删除订单，  系统将删除标记修改为1（已删除） | Post | 无 |
| order/updateOrder{id} | {id} | void | 系统发货，若发货，则将处理标记修改为1（已发货），用户若已 签收，则将处理标记修改为2（已签收） | Post |  |

# 五、组件设计

### 用户模块

##### com.taoqi.user

###### 启动类：UserServiceApp

###### 实体类：User

##### com.taoqi.user.mapper

###### 业务逻辑UserService依赖于UserMapper（接口名）并定义其方法

###### 接口名：UserMapper

方法：

1. getUserAll

方法签名：**List<User> getUserAll(int page,int size)**

类型：抽象方法

参数：

int page 页码

int size 大小

操作的表：t\_user

返回值：List<User> User实体集合 (用户信息集合)

方法描述：通过page和size参数进行分页查询，并将结果封装到List<User>，返回该 List<User>实体集合

1. updateUser

方法签名: **int update(User user)**

参数：User user 需要更新的User对象

类型：抽象方法

操作的表：v\_user

返回值：int 更新语句影响行数

方法描述：通过SQL更新语句更新t\_user表中特定的User信息，并将影响行数返回；

1. saveUser

方法签名：**int saveUser(User user)**

类型：抽象方法

参数：User user 所要保存的用户对象

操作的表：t\_user

返回值：int 插入语句影响行数

方法描述：通过提供的用户实体对象，使用SQL语句将其数据插入到t\_user表中，并 且返回影响行数；

##### com.taoqi.user.service（接口类）

###### 业务逻辑：UserServiceImp依赖于UserService（接口名）并定义其方法

###### 接口名：UserService

方法：

1、get

方法签名：**Map<String, Object> get(int page,int size)**

方法类型：抽象方法

方法参数：

int page 页码

int size 大小

返回值：Map<String, Object> map集合（数据、消息、状态码）状态码：200，消息：‘操作成功’ ，数据：用户实体集合数据；

方法流程：通过注入的Mapper组件调用UserMapper接口的getUserAll业务方法，将返回数据作为参数调用ResultUtil的succes方法返回map集合（数据、消息、状态）

2、update

方法签名：**Map<String, Object> update(User user)**

方法类型：抽象方法

方法参数：User user 用户实体

返回值：Map<String, Object> map集合（数据、消息、状态码）状态码：200/500，消息：‘操作成功’ /‘操作失败’，数据：无

方法流程：通过注入的Mapper组件调用UserMapper接口的updateUser业务方法，将返回数据通过判断调用ResultUtil的succes方法或者error方法返回map集合（数据、消息、状态）

3、reg

方法签名：**Map<String, Object> reg(User user)**

方法类型：抽象方法

方法参数：User user 用户实体

返回值：Map<String, Object> map集合（数据、消息、状态码）状态码：200/500，消息：‘操作成功’/‘操作失败’ ，数据：无

方法流程：通过注入的Mapper组件调用UserMapper接口的saveUser业务方法，将返 回数据通过判断调用ResultUtil的succes方法或者error方法返回map集合（数据、消 息、状态）

##### com.taoqi.user.service.Impl

###### 业务逻辑：UserController调用UserServiceImpl（实现UserService接口）的方法

###### 类名：UserServiceImp

方法

1. get

实现自UserService

1. update

实现自UserService

1. reg

实现自UserService

##### com.taoqi.user.controller

类名：UserController

###### 方法

1. get

业务逻辑：前端或客户端（游客）提交 HTTP GET 请求到 UserController的 get方法

方法签名：**public Map<String, Object> get(int page,int size)**

返回值：Map集合(状态码、消息、数据)

方法请求映射为路径URL：/user/get

方法参数：

int page 页码

int size 大小

方法流程：首先校验参数，再通过注入的Service组件调用UserService接口的get业务方法

1. update

业务逻辑：前端或客户端（店铺）提交 HTTP GET 请求到 UserController的 update方法

方法签名：**public Map<String, Object> update(User user)**

返回值：Map集合(状态码、消息、数据)

方法请求映射为路径URL: /user/update

方法参数：

User user 用户实体

方法流程：首先校验参数，再通过注入的Service组件调用UserService接口的update业务方法

1. reg

业务逻辑：前端或客户端（系统）提交 HTTP GET 请求到 UserController的 reg方法:

方法签名：**public Map<String, Object> reg(User user)**

返回值：Map集合(状态码、消息、数据)

方法请求映射为路径URL：/user/reg

方法参数：

User user 用户实体

方法流程：首先校验参数，再通过注入的Service组件调用UserService接口的reg业务方法

### 评论模块

##### com.taoqi.evaluate

###### 启动类：EvaluateServiceApp

###### 实体类：Evaluate

##### com.taoqi.evaluate.mapper

###### 业务逻辑EvaluateService依赖于EvaluateMapper（接口名）并定义其方法

###### 接口名：EvaluateMapper

方法：

1. touristfind

方法签名：**List<VEvaluate> touristfind(long priductid)**

字段：无

类型：抽象方法

参数：

long priductid 商品编号

操作的表/视图：t\_order（具体表）和v\_evaluate(视图)

返回值：List<VEvaluate> VO/DTO实体集合(评论信息集合)，该商品的所有订单所对应的所有评论信息，

描述：通过商品的编号使用SQL语句查询数据库中t\_order表和视图v\_evaluate，并且将其封装到List<VEvaluate>，然后在将List<VEvaluate>实体集合返回

1. shopfind

方法签名：**List<VEvaluate> shopfind(long orderid)**

字段：无

类型：抽象方法

参数：

orderid long 订单编号

操作的表/视图：v\_evaluate

返回值：List<VEvaluate> VO/DTO实体集合 (评论信息集合)，该订单所对应的所有评论信息

方法描述：通过订单编号使用SQL语句查询数据库视图中v\_evaluate表中该订单所对应的所有评论信息，并且封装到List<VEvaluate>，返回该List<VEvaluate>实体集合

1. adminfind

方法签名: **List<VEvaluate> adminfind()**

字段：无

参数：无

类型：抽象方法

操作的表/视图：v\_evaluate

返回值：List<VEvaluate> VO/DTO实体集合 (评论信息集合)，数据库中所有评论信息

方法描述：通过SQL查询语句查询v\_evaluate视图表中所有的评论信息，并且将所查询到的数据封装到List<VEvaluate>，然后将List<VEvaluate>实体集合返回

1. addevaluate

方法签名：**int addevaluate(Evaluate evaluate)**

字段：无

类型：抽象方法

参数：

evaluate Evaluate 所要插入的评论对象

操作的表/视图：t\_evaluate

返回值：int 转台编号1/0（1：插入数据成功；2：插入数据失败）

方法描述：通过提供的评论实体对象，使用SQL语句将其数据插入到t\_evaluate表中，并且返回1/0（1：插入数据成功；2：插入数据失败）

1. deleteevaluate

方法签名：**int deleteevaluate(long id)**

字段：无

类型：抽象方法

参数：

Id long 评论编号

操作的表/视图：t\_evaluate

方法描述：通过评论编号使用SQL语句将数据库的表t\_evaluate中所对应评论编号的所有信息从t\_evaluate中移除；

返回值：int 状态编号 1/0（1：插入成功 0:插入失败）

##### com.taoqi.evaluate.service（接口类）

###### 业务逻辑：EvaluateServiceImp依赖于EvaluateService（接口名）并定义其方法

###### 接口名：EvaluateService

方法：

1、touristServicefind

方法签名：**Map<String, Object> touristServicefind(long priductid)**

字段：无

方法类型：抽象方法

方法参数：priductid long 商品编号

返回值：Map<String, Object> map集合（数据、消息、状态码）状态码：200，消息：‘操作成功’ ，数据：商品编号对应的所有订单的所有评论信息

方法流程：touristServicefind调用EvaluateMapper接口的touristfind业务方法，将返回数据作为参数调用ResultUtil的succes方法返回map集合（数据、消息、状态）

2、shopServicefind

方法签名：**Map<String, Object> shopServicefind(long orderid)**

字段：无

方法类型：抽象方法

方法参数：orderid long 商品编号

返回值：Map<String, Object> map集合（数据、消息、状态码）状态码：200，消息：‘操作成功’ ，数据：该订单下的所有评论信息

方法流程：shopservicefind调用EvaluateMapper接口的shopfind业务方法，将返回数据作为参数调用ResultUtil的succes方法返回map集合（数据、消息、状态）

3、adminServicefind

方法签名：**Map<String, Object> adminServicefind()**

字段：无

方法类型：抽象方法

方法参数：无

返回值：Map<String, Object> map集合（数据、消息、状态码）状态码：200，消息：‘操作成功’ ，数据：数据库中所有评论信息

方法流程：adminServicefind调用EvaluateMapper接口的adminfind业务方法，将返回数据作为参数调用ResultUtil的succes方法返回map集合（数据、消息、状态）

4、addServiceEvaluate

方法签名：**Map<String, Object> addevaluate(Evaluate evaluate)**

字段：无

方法类型：抽象方法

方法参数：无

返回值：Map<String, Object> map集合（消息、状态码）状态码：200/500，消息：‘操作成功’

方法流程：adminServicefind调用EvaluateMapper接口的adminfind业务方法，将返回数据作为参数调用ResultUtil的succes方法返回map集合（数据、消息、状态）

5、deleteServiceEvaluate

方法签名：**Map<String, Object> deleteevaluate(long id)**

字段：无

方法类型：抽象方法

方法参数：id long

返回值：Map<String, Object> map集合（消息、状态码）状态码：200/500，消息：‘操作成功’ /“操作失败”

方法流程：adminServicefind调用EvaluateMapper接口的adminfind业务方法，将返回数据为1，则将字符串“操作成功”作为参数调用ResultUtil的succes方法返回map集合（数据、消息、状态）,若返回数据为0，则将字符串“操作失败”作为参数调用ResultUtil的error方法返回map集合（数据、消息、状态）

##### com.taoqi.evaluate.service.Impl

###### 业务逻辑：EvaluateController调用EvaluateServiceImp（实现EvaluateService接口）的方法

###### 类名：EvaluateServiceImp

方法

1. touristServicefind

方法签名：**public Map<String, Object>touristServicefind(long priductid)**

字段：无

方法参数：priductid long 商品编号

返回值：Map<String, Object> map集合（状态码、消息、数据）状态码：200，消息：‘操作成功’ ，数据：数据库中该商品所对应的所有订单的所有评论信息

方法描述：实现接口EvaluateService中touristServicefind方法所描述的所有业务逻辑

1. shopServicefind

方法签名：**public Map<String, Object>shopServicefind(long orderid)**

字段：无

方法参数：orderid long 订单编号

返回值：Map<String, Object> map集合（状态码、消息、数据）状态码：200，消息：‘操作成功’ ，数据：数据库中该订单的所有评论信息

方法描述：实现接口EvaluateService中shopServicefind方法所描述的所有业务逻辑

1. adminServicefind

方法签名：**public Map<String, Object>adminServicefind()**

字段：无

方法参数：无

返回值：Map<String, Object> map集合（状态码、消息、数据）状态码：200，消息：‘操作成功’ ，数据：数据库中所有评论信息

方法描述：实现接口EvaluateService中adminServicefind方法所描述的所有业务逻辑

1. addServicEevaluate

方法签名：**public Map<String, Object>addServicEevaluate(Evaluate evaluate)**

字段：msg String

方法参数：evaluate Evaluate 评论对象（POJO）

返回值：Map<String, Object> map集合（状态码、消息）状态码：200或500，消息：‘操作成功’ 或“操作失败”

方法描述：实现接口EvaluateService中addServicEevaluate方法所描述的所有业务逻辑

1. deleteServiceEvaluate

方法签名：**public Map<String, Object>deleteServiceEvaluate(long orderid)**

字段：无

方法参数：orderid long 订单编号

返回值：Map<String, Object> map集合（状态码、消息）状态码：200或500，消息：‘操作成功’ 或“操作失败”

方法描述：实现接口EvaluateService中deleteServiceEvaluate方法所描述的所有业务逻辑

##### com.taoqi.evaluate.controller

类名：EvaluateController

###### 方法

1. getTouristEvaluate

业务逻辑：前端或客户端（游客）提交 HTTP GET 请求到 EvaluateController的 getTouristEvaluate方法

方法签名：**public Map<String, Object>getTouristEvaluate(long priductid)**

字段：无

返回值：JSON格式数据(状态码、消息、数据)

方法请求映射为路径URL：/uget/{priductid}

方法参数：

priductid long 商品编号

方法流程：getTouristEvaluate方法调用 EvaluateService中的 touristServicefind业务方法返回JSON数据(状态码、消息、数据)

1. getShopEvaluate

业务逻辑：前端或客户端（店铺）提交 HTTP GET 请求到 EvaluateController的 getShopEvaluate方法

方法签名：**public Map<String, Object>getShopEvaluate(orderid long)**

字段：无

返回值：JSON格式数据(状态码、消息、数据)

方法请求映射为路径URL: /sget/{orderid}

方法参数：

orderid long 订单编号

方法流程：getShopEvaluate方法调用 EvaluateService中的 shopServicefind业务方法返回JSON数据(状态码、消息、数据)

1. getAdminEvaluate

业务逻辑：前端或客户端（系统）提交 HTTP GET 请求到 EvaluateController的 getAdminEvaluate方法:

方法签名：**public Map<String, Object>getAdminEvaluate()**

字段：无

返回值：JSON格式数据(状态码、消息、数据)

方法请求映射为路径URL：/get

方法参数：无

方法流程：getAdminEvaluate方法调用 EvaluateService中的 adminServicefind业务方法返回JSON数据(状态码、消息、数据)

1. addevaluate

业务逻辑：前端或客户端（用户）提交 HTTP GET 请求到 EvaluateController的 addevaluate方法:

字段：无

方法签名：**public Map<String, Object>addEvaluate(Evaluate evaluate)**

返回值：JSON格式数据(状态码、消息、数据)

方法请求映射为路径URL: evaluate/uget/{priductid}

方法参数：

evaluate Evaluate 评论对象（POJO）

方法流程：addEvaluate方法调用 EvaluateService中的 addServicEevaluate业务方法返回JSON数据(状态码、消息、数据)

1. deleteEvaluate

业务逻辑：前端或客户端（系统）提交 HTTP GET 请求到 EvaluateController的 deleteEvaluate方法:

字段：无

方法签名：**public Map<String, Object>deleteEvaluate(long id)**

方法返回JSON格式数据

方法请求映射为路径URL: evaluate/uget/{priductid}

方法参数：

Id long 评论编号

方法流程：deleteEvaluate方法调用 EvaluateService中的 deleteServiceEvaluate业务方法返回JSON数据

### 购物车模块

##### com.taoqi.car

###### 启动类：CarServiceApp

###### 实体类：Car

###### VO实体类： VCar

##### com.taoqi.car.mapper

###### 业务逻辑CarService依赖于CarMapper（接口名）并定义其方法

接口名:CarMapper

方法

1. findcar

方法签名：**List<Car> findcar(long userid)**

方法类型：抽象方法

方法参数：userid long 用户编号

返回值：List<Car> 该用户的所有购物车信息的集合

操作的表/视图：v\_car 视图

方法描述：通过用户编号使用SQL语句查找数据库v\_car(视图)中该用户所有的购物车信息，并且将其封装成一个List<Car>集合，然后在进行返回

1. deleteById

方法签名：**int deletebyid(long carid)**

方法类型：抽象方法

方法参数：carid long 购物车编号

返回值：int 0/1 1:删除成功；2：删除失败

操作的表/视图：t\_car

方法描述：通过购物车编号使用SQL语句删除数据库t\_car中该购物车编号所对应的所有信息，并且返回1/0，1:删除成功；2：删除失败 ；

1. deleteAllById

方法签名：**int deleteAllbyId(long userid)**

方法类型：抽象方法

方法参数：userid long 购物车编号

返回值：int 0/1 1:删除成功；2：删除失败

操作的表/视图：t\_car

方法描述：通过用户编号使用SQL语句删除数据库t\_car中该用户编号所有的购物车信息，并且返回1/0，1:删除成功；2：删除失败 ；（清空购物车）

1. addCar

方法签名：**int addCar (Car car)**

方法类型：抽象方法

方法参数：Car car

返回值：int 0/1 1:添加成功；2：添加失败

操作的表/视图：t\_car

方法描述：通过参数提供的数据对象，使用SQL语句将对象数据插入数据库中的t\_car

1. updateCar

方法签名：**int UpdateCar(int count，long carid);**

方法类型：抽象方法

方法参数：

count int 商品数量

carid long 购物车编号

返回值：int 0/1 1:更新成功；2：更新失败

操作的表/视图：t\_car

方法描述：通过提供的购物车编号和商品数量使用SQL语句更新数据库t\_car中该购物车编号将购物车的商品数量更新，并且返回 0/1 1:更新成功；2：更新失败

##### com.taoqi.car.service

###### 业务逻辑：CarServiceImp依赖于CarService（接口名）并定义其方法

###### 接口名：CarService

方法：

1、servicefindcar

方法签名：**Map<String, Object> servicefindcar(long userid)**

方法类型：抽象方法

方法参数：userid long 用户编号

返回值：Map<String, Object> map集合（消息、状态码、数据）状态码：200，消息：‘操作成功’，数据：商品编号对应的所有订单的所有评论信息

方法流程：servicefindcar调用CarMapper接口的findcar业务方法，将返回数据作为参数调用ResultUtil的succes方法然后返回map集合（数据、消息、状态）

2、deleteServiceById

方法签名：**Map<String, Object> deleteServicebyId(long carid)**

方法类型：抽象方法

方法参数：carid long 商品编号

返回值：Map<String, Object> map集合（消息、状态码）状态码：200/500，消息：‘操作成功’/‘操作失败’

方法流程：deleteServicebyId调用CarMapper接口的deleteById业务方法，若返回数据为1，则将字符串“操作成功”作为参数调用ResultUtil的succes方法返回map集合（数据、消息、状态）,若返回数据为0，则将字符串“操作失败”作为参数调用ResultUtil的error方法返回map集合（数据、消息、状态）

3、deleteServiceAll

方法签名：**Map<String, Object> deleteServiceAll(long userid)**

方法类型：抽象方法

方法参数：userid long

返回值：Map<String, Object> map集合（消息、状态码）状态码：200/500，消息：‘操作成功’/‘操作失败’

方法流程：deleteServicebyId调用CarMapper接口的deleteAllById业务方法，若返回数据为1，则将字符串“操作成功”作为参数调用ResultUtil的succes方法返回map集合（数据、消息、状态）,若返回数据为0，则将字符串“操作失败”作为参数调用ResultUtil的error方法返回map集合（数据、消息、状态）

4、addServiceCar

方法签名：**Map<String, Object> addServiceCar(Car car)**

方法类型：抽象方法

方法参数：car Car 购物车对象

返回值：Map<String, Object> map集合（消息、状态码）状态码：200/500，消息：‘操作成功’/‘操作失败’

方法流程：deleteServicebyId调用CarMapper接口的deleteAllById业务方法，若返回数据为1，则将字符串“操作成功”作为参数调用ResultUtil的succes方法返回map集合（数据、消息、状态）,若返回数据为0，则将字符串“操作失败”作为参数调用ResultUtil的error方法返回map集合（数据、消息、状态）

5、updateServiceCar

方法签名：**Map<String, Object> updateServiceCar(long carid, int count)**

方法类型：抽象方法

方法参数：

carid long 购物车编号

count int 商品数量

返回值：Map<String, Object> map集合（消息、状态码）状态码：200/500，消息：‘操作成功’ /“操作失败”

方法流程：updateServiceCar调用CarMapper接口的updateCar业务方法，将返回数据为1，则将字符串“操作成功”作为参数调用ResultUtil的succes方法返回map集合（数据、消息、状态）,若返回数据为0，则将字符串“操作失败”作为参数调用ResultUtil的error方法返回map集合（数据、消息、状态）

##### com.taoqi.car.serviceImpl

###### 业务逻辑：CarController调用CarServiceImpl（实现CarService接口）的方法

###### 类名：CarServiceImpl

方法

1、servicefindcar

方法签名：**public Map<String, Object>touristServicefind(long productid)**

方法描述：实现接口CarService中servicefindcar方法所描述的所有业务逻辑

2、deleteServiceById

方法签名：**public Map<String, Object>shopServicefind(long orderid)**

方法描述：实现接口CarService中deleteServiceById方法所描述的所有业务逻辑

3、deleteServiceAll

方法签名：**public Map<String, Object>adminServicefind()**

方法描述：实现接口CarService中deleteServiceAll方法所描述的所有业务逻辑

4、addServiceCar

方法签名：**public Map<String, Object>addServicEevaluate(Evaluate evaluate)**

方法描述：实现接口CarService中addServiceCar方法所描述的所有业务逻辑

5、updateServiceCar

方法签名：**public Map<String, Object>deleteServiceEvaluate(long orderid)**

方法描述：实现接口CarService中updateServiceCar方法所描述的所有业务逻辑

### 权限模块

##### com.taoqi.auth

###### 启动类：AuthServiceApp

###### 实体类：Auth、Role

##### com.taoqi.auth.mapper

###### 业务逻辑AuthService依赖于AuthMapper（接口名）并定义其方法

###### 接口名：AuthMapper

方法：

1. login

方法签名：**User login(String name,String pass)**

类型：抽象方法

参数：

String name 用户名

String pass 密码

操作的表：t\_user

返回值：User User实体

方法描述：通过name和pass参数调用登录的存储过程，并将结果封装到User实体 返回

1. getRolesByUserId

方法签名: **List<String> getRolesByUserId(long id)**

类型：抽象方法

参数：long id 用户id

操作的视图：v\_auth

返回值：List<String>，用户拥有的权限集合

方法描述：通过id参数调用查询语句，并将结果封装到List集合返回；

1. getUserByUserName

方法签名：**User getUserByUserName(String userName)**

类型：抽象方法

参数：String userName 用户名

操作的表：t\_user

返回值：User User实体

方法描述：通过提供的用户名调用查询语句，并将结果封装到User实体返回；

##### com.taoqi.auth.service（接口类）

###### 业务逻辑：AuthServiceImp依赖于AuthService（接口名）并定义其方法

###### 接口名：AuthService

方法：

1、login

方法签名：**Map<String, Object> login(String name,String pass)**

方法类型：抽象方法

方法参数：

String name 用户名

String pass 密码

返回值：Map<String, Object> map集合（数据、消息、状态码）状态码：200/500，消息：‘登录成功’/‘登录失败’，数据：jwt字符、User对象；

方法流程：通过注入的Mapper组件调用AuthMapper接口的login业务方法，将返回数据如果不为null则将其加密为jwt字符并作为参数调用ResultUtil的succes方法返回map集合（数据、消息、状态）

2、check

方法签名：**Map<String, Object> check(String jwt)**

方法类型：抽象方法

方法参数：String jwt jwt字符

返回值：Map<String, Object> map集合（数据、消息、状态码）状态码：200/500，消息：‘操作成功’ /‘操作失败’，数据：jwt字符、User对象；

方法流程：对传入的jwt字符进行解密，如果正常解密且没有超时，则对其中的用户信息再次进行jwt加密生成新的jwt字符并作为参数调用ResultUtil的succes方法返回map集合（数据、消息、状态）

##### com.taoqi.auth.service.Impl

###### 业务逻辑：AuthController调用AuthServiceImpl（实现AuthService接口）的方法

###### 类名：AuthServiceImp

方法

1. login

实现自AuthService

1. check

实现自AuthService

##### com.taoqi.auth.controller

类名：AuthController

###### 方法

1. login

业务逻辑：前端或客户端（游客）提交 HTTP POST请求到 AuthController的 login方法

方法签名：**public Map<String, Object> long(String name,String pass)**

返回值：Map集合(状态码、消息、数据)

方法请求映射为路径URL：/auth/login

方法参数：

String name 用户名

String pass 密码

方法流程：首先校验参数，再通过注入的Service组件调用AuthService接口的login业务方法

1. check

业务逻辑：前端或客户端（店铺）提交 HTTP POST请求到 AuthController的 check方法

方法签名：**public Map<String, Object> check(String jwt)**

返回值：Map集合(状态码、消息、数据)

方法请求映射为路径URL: /user/check

方法参数：

String jwt jwt字符

方法流程：首先校验参数，再通过注入的Service组件调用AuthService接口的check业务方法

### 商品模块

##### 包：com.taoqi.product

###### 启动类：ProductServiceApp

###### 实体类：Product

#### **com.taoqi.product.service(接口类**)

类名：ProductService

##### **1.getlike**：

方法签名：Map<String, Object> getlike (String str);

方法类型：抽象方法

方法参数：String str 商品关键字

返 回 值：Map<String, Object> map集合（数据、消息、状态码）状态码：

200，消息：‘操作成功’ ，数据：所有的商品集合信息

方法流程: 当前getlike调用ProductMapper接口的getlike实现方法，将返回

数据作为参数调用ResultUtil的succes方法返回map集合（数据、

消息、状态）

##### 2.getbytype

方法签名：Map<String, Object> getbytype (long typeid);

方法类型：抽象方法

方法参数：long typeid 商品分类编号

返 回 值：Map<String, Object> map集合（数据、消息、状态码）状态码：

200，消息：‘操作成功’ ，数据：所有的商品集合信息

方法流程: 当前getbytype调用ProductMapper接口的getbytype实现方法，，

将返回数据作为参数调用ResultUtil的succes方法返回map集合（数

据、消息、状态）

##### 3.getbyshop

方法签名：Map<String, Object> getbyshop (long shopid);

方法类型：抽象方法

方法参数：long shopid 商品店铺编号

返 回 值：Map<String, Object> map集合（数据、消息、状态码）状态码：

200，消息：‘操作成功’ ，数据：所有的商品集合信息

方法流程: 当前getbyshop调用ProductMapper接口的getbyshop实现方法，将返回数据作为参数调用ResultUtil的succes方法返回map集合（数据、消息、状态）

##### 4.upProduct

方法签名：Map<String, Object> upProduct(Product pro);

方法类型：抽象方法

方法参数：Product pro 商品实体类

返 回 值：Map<String, Object> map集合（数据、消息、状态码）状态码：

200，消息：‘操作成功’ ，数据：所有的商品集合信息

方法流程: 当前upProduct调用ProductMapper接口的upProduct实现方法，将返回数据作为参数调用ResultUtil的succes方法返回map集合（数据、消息、状态）

##### 5.downProduct

方法签名：Map<String, Object> downProduct(Long productid);

方法类型：抽象方法

方法参数：Long prodctid 商品编号

返 回 值：Map<String, Object> map集合（数据、消息、状态码）状态码：

200，消息：‘操作成功’ ，数据：所有的商品集合信息

方法流程: 当前downProduct调用ProductMapper接口的downProduct实现

方法，将返回数据作为参数调用ResultUtil的succes方法返回map集合（数据、消息、状态）

##### 6．changeProduct

方法签名：Map<String, Object> changeProduct(Product pro);

方法类型：抽象方法

方法参数：Product pro 商品实体类

返 回 值：Map<String, Object> map集合（数据、消息、状态码）状态码：

200，消息：‘操作成功’ ，数据：所有的商品集合信息

方法流程: 当前changeProduct调用ProductMapper接口的changeProduct实现方法，将返回数据作为参数调用ResultUtil的succes方法返回map集合（数据、消息、状态）

#### com.taoqi.product.service.Impl(实现类)

业务逻辑：ProductController调用ProductServiceImp（实现ProductService接口）的方法

类名: ProductServiceImp

##### 1.getlike:

方法描述：实现接口ProductService 中的getlike业务逻辑方法

##### **2.getbytype**:

方法描述：实现接口ProductService中的getbytype业务逻辑方法

##### 3.getbyshop:

方法描述：实现接口ProductService中的getbyshop业务逻辑方法

##### **4.upProduct**:

方法描述：实现接口ProductService中的upProduct业务逻辑方法

##### 5.downProduct

方法描述：实现接口ProductService中的downProduct业务逻辑方法

##### **6.changeProduct**:

方法描述：实现接口ProductService中的changeProduct业务逻辑方法

#### com.taoqi.product.mapper（接口类）

业务逻辑ProductService依赖于ProductMapper（接口名）并定义其方法

接口名：ProductMapper

方法：

##### 1.getlike:

方法签名：List<Product>getlike (String str);

类 型: 抽象方法

参 数：String str 商品关键字

操作的表：t\_product(商品表)

方法描述：通过elasticsearch*搜索引擎*查询商品并将其封装到list<product>集

合返回

返 回 值：List<Product> 商品集合

##### 2.getbytype：

方法签名：List<Product>getbytype (long typeid);

类 型: 抽象方法

参 数：long typeid 商品类型编号

操作的表：t\_product(商品表)

方法描述：通过商品表（t\_product）中分类id（type\_id）使用SQL语句查询相关商

品并将数据封装到list<product>，返回该实体集合

返 回 值：List<Product> 商品集合

##### 3.getbyshop:

方法签名：List<Product>getbyshop (long shopid);

类 型: 抽象方法

参 数：long shopid 店铺编号

操作的表：t\_product(商品表)

方法描述：通过商品表（t\_product）中店铺id（shop\_id）使用SQL语句查询相关商

品并将数据封装到list<product>，返回该实体集合

返回值：List<Product> 商品集合

##### 4.upProduct:

upProduct:

方法签名：int upProduct (Product pro);

类 型: 抽象方法

参 数：Product pro 所要插入的商品对象

操作的表：t\_product(商品表)

方法描述：将商品实体对象使用SQL语句将其数据插入到t\_product表中，并且返

回1/0（1：插入数据成功；2：插入数据失败）

返 回 值：int <0,1> (0,插入失败；，插入成功)

##### **5.downProduct**:

方法签名：int downProduct (long productid);

类 型: 抽象方法

参 数：long productid 所要删除的商品对象编号id

操作的表：t\_product(商品表)

方法描述：通过商品id使用SQL语句删除商品表（t\_product）中的商品信息

返 回 值：long 0/1 (0,删除失败；，删除成功)

##### **6.changeProduct**:

方法签名：int changeProduct (Product pro);

类 型: 抽象方法

参 数：Product pro所要修改商品的对象

操作的表：t\_product(商品表)

方法描述：根据该商品使用sql语句更新改商品信息

返 回 值：long 0/1 (0,修改失败；修改成功)

#### com.taoqi.product.controller

##### 1.getlike：

前端提交HTTP GET请求到productController的getlike方法：

方法签名：public Map<String, Object> getlike(String str)

返回值：为json格式数据

方法请求映射为路径（URL）: /getlike

参数：　String str 查询的商品名

方法流程：getlike方法调用ProductService中的getlike方法返回json数据

##### **2.getbytype**：

前端提交HTTP GET请求到productController的getbytype方法：

方法签名：public Map<String, Object>getbytype (long typeid)

返回值：为json格式数据

方法请求映射为路径（URL）: / getbytype

参数：long typeid商品类型编号

方法流程：getbytype方法调用ProductService中的getbytype方法返回json数据

##### 3.getbyshop:

前端提交HTTP GET请求到productController的getbyshop方法：

方法签名：public Map<String, Object>getbytype (long shopid)

返回值：为json格式数据

方法请求映射为路径（URL）: / getbytype

参数：　long shopid 店铺编号

方法流程：getbyshop方法调用ProductService中的getbyshop方法返回json数据

##### 4.upProduct:

前端提交HTTP POST请求到productController的upProduct方法：

方法签名：public Map<String, Object> upProduct (Product product)

返回值：为json格式数据

方法请求映射为路径（URL）: / add

参数：Product product 商品对象

方法流程：upProduct方法调用ProductService中的upProduct方法返回json数据

##### 5.downProduct:

前端提交HTTP POST请求到productController的downProduct方法：

方法签名：public Map<String, Object> downProduct (long id)

返回值：为json格式数据

方法请求映射为路径（URL）: / add/{id}

参数：long　id 商品对象编号

##### 6.changeProduct:

前端提交HTTP POST请求到productController的changeProduct方法：

方法签名：public Map<String, Object>changeProduct (Product pro)

返回值：为json格式数据

方法请求映射为路径（URL）: / change

参数：　Product pro 商品对象

### 店铺模块

#### com.taoqi.shop.service(接口类)

##### **1.addshop**：

方法签名：Map<String, Object> addshop (Shop shop);

方法类型：抽象方法

方法参数：Shop shop 店铺实体类

返 回 值：Map<String, Object> map集合（数据、消息、状态码）状态码：

200，消息：‘操作成功’ ，数据：所有的商品集合信息

方法流程: 当前addshop调用ShopMapper接口的addsho实现方法，将返回

数据作为参数调用ResultUtil的succes方法返回map集合（数据、

消息、状态）

##### **2.delshop**：

方法签名：Map<String, Object> addshop (long shopid);

方法类型：抽象方法

方法参数：long shopid 店铺编号

返 回 值：Map<String, Object> map集合（数据、消息、状态码）状态码：

200，消息：‘操作成功’ ，数据：所有的商品集合信息

方法流程: 当前delshop调用ShopMapper接口的delshop实现方法，将返回

数据作为参数调用ResultUtil的succes方法返回map集合（数据、

消息、状态）

##### **3.changeshop**：

方法签名：Map<String, Object> changeshop(Shop shop);

方法类型：抽象方法

方法参数：Shop shop店铺实体类

返 回 值：Map<String, Object> map集合（数据、消息、状态码）状态码：

200，消息：‘操作成功’ ，数据：所有的商品集合信息

方法流程: 当前changeshop调用ShopMapper接口的changeshop实现方法，

将返回数据作为参数调用ResultUtil的succes方法返回map集合（数据、消息、状态）

##### **4.getall**：

方法签名：Map<String, Object>getall();

方法类型：抽象方法

方法参数：无

返 回 值：Map<String, Object> map集合（数据、消息、状态码）状态码：

200，消息：‘操作成功’ ，数据：所有的商品集合信息

方法流程: 当前getall调用ShopMapper接口的getall实现方法，

将返回数据作为参数调用ResultUtil的succes方法返回map集合（数据、消息、状态）

#### com.taoqi. shop.serviceImpl(实现类)

##### addshop

方法描述：实现接口ShopService中的addshop业务逻辑方法

##### delshop

方法描述：实现接口ShopService中的delshop业务逻辑方法

##### changeshop

方法描述：实现接口ShopService中的changeshop业务逻辑方法

##### getall

方法描述：实现接口ShopService中的getall业务逻辑方法

#### com.taoqi. shop.controller

##### 1.**addshop**

前端提交HTTP POST请求到ShopController的addshop方法：

方法签名：public Map<String, Object> addshop(Shop shop)

返回值：为json格式数据

方法请求映射为路径（URL）: /addshop

参数：　Shop shop 店铺实体类

方法流程：addshop方法调用ShopService中的addshop方法返回json数据

##### 2.**delshop**

前端提交HTTP POST请求到ShopController的delshop方法：

方法签名：public Map<String, Object> delshop(long shopid)

返回值：为json格式数据

方法请求映射为路径（URL）: /delshop

参数：　long shopid 店铺编号

方法流程：delshop方法调用ShopService中的delshop方法返回json数据

##### 3.change**shop**

前端提交HTTP POST请求到ShopController的changeshop方法：

方法签名：public Map<String, Object> changeshop(Shop shop)

返回值：为json格式数据

方法请求映射为路径（URL）: /changeshop

参数：　Shop shop 店铺实体类

方法流程：changeshop方法调用ShopService中的changeshop方法返回json数据

##### 4.getall

前端提交HTTP GET请求到ShopController的getall方法：

方法签名：public Map<String, Object> getall()

返回值：为json格式数据

方法请求映射为路径（URL）: /getall

参数：　无

方法流程：getall方法调用ShopService中的getall方法返回json数据

#### com.taoqi. shop.mapper（接口类）

业务逻辑ShopService依赖于ShopMapper（接口名）并定义其方法

接口名：ShopMapper

方法：

##### addshop

方法签名：int addshop (Shop shop);

类 型: 抽象方法

参 数：Shop shop 店铺实体类

操作的表：t\_shop(店铺表)

方法描述：将店铺实体对象使用SQL语句将其数据插入到t\_shop表中，并且返回

1/0（1：插入数据成功；2：插入数据失败）

返 回 值：int <0,1> (0,插入失败；，插入成功)

##### 2.delshop

方法签名：int delshop (long shopid);

类 型: 抽象方法

参 数：long shopid店铺编号

操作的表：t\_shop(店铺表)

方法描述：根据店铺编号使用sql删除该店铺所有信息，并且返回0/1（1：插入数

据成功；2：插入数据失败）

返 回 值：int <0,1> (0,删除失败；，删除成功)

##### 3.changeshop

方法签名：int changeshop (Shop shop);

类 型: 抽象方法

参 数：Shop shop 店铺实体类

操作的表：t\_shop(店铺表)

方法描述：根据店铺编号使用sql语句改变该店铺所有信息，并且返回0/1（1：插

入数据成功；2：插入数据失败）

返 回 值：int <0,1> (0,修改失败；修改成功)

##### 4.getall

方法签名：List<shop> getall();

类 型: 抽象方法

参 数：无

操作的表：t\_shop(店铺表)

方法描述：用sql获取店铺表t\_shop获取所有店铺信息，并将数据封装到list<shop>

集合

返 回 值：int <0,1> (0,修改失败；修改成功)

### 分类模块

##### com.taoqi.type

###### 启动类：TypeServiceApplication

###### 实体类：Type

##### com.taoqi.type.Mapper

###### 业务逻辑TypeService依赖于TypeMapper（接口名）并定义其方法方法：

###### SelectAllTtpe

方法签名：List<Type> SelectAllTtpe();

类型：抽象方法

参数：

无

操作的表/视图：t\_type

返回值：List<Type> Type实体集合（分类集合），所有系统添加的一级分类。

方法描述：通过SQL查询语句查询t\_type表中所有系统添加的一级分类，并且将所查询到的数据封装到List<Type>，然后将List<Type>实体集合返回。

###### SelecttTtpe

方法签名：List<Type> SelecttTtpe();

类型：抽象方法

参数：

无

操作的表/视图：t\_type

返回值：List<Type> Type实体集合（分类集合），所有系统添加的二级分类。

方法描述：通过SQL查询语句查询t\_type表中所有系统添加的二级分类，并且将所查询到的数据封装到List<Type>，然后将List<Type>实体集合返回。

###### addTypem

方法签名：int addTypem(Type t);

类型：抽象方法

参数：

Type对象

操作的表/视图：t\_type

返回值：int 返回当前操作影响行数，（0:添加失败,!0:添加成功）

方法描述：通过提供的分类实体对象，使用SQL语句将其数据插入到t\_type表中，并且返回!0/0（!0：插入数据成功；0：插入数据失败）

###### delTypest

方法签名：int delTypest( Long i);

类型：抽象方法

参数：

Long i 需要要删除对象id

操作的表/视图：t\_type

返回值：int 返回当前操作影响行数，（0:添加失败,!0:添加失败）

方法描述：通过提供的需要要删除对象id，使用SQL语句将其数据从t\_type表中删除，并且返回!0/0（!0：插入数据成功；0：插入数据失败）

###### selectmer

方法签名：List<Type> selectmer( Long i);

类型：抽象方法

参数：

Long i 需要查询店铺分类

操作的表/视图：t\_type

返回值：List<Type> Type实体集合（分类集合），店铺添加的分类

方法描述：通过SQL查询语句查询t\_type表中所有店铺添加的分类，并且将所查询到的数据封装到List<Type>，然后将List<Type>实体集合返回。

##### com.taoqi.type.Service

###### Service

业务逻辑：TypeServiceImp依赖于TypeService（接口名）并定义其方法

AllTypes

方法签名：List<Type> AllTypes();

字段：无

方法参数：无

返回值：List<Type> LIst集合（状态码、消息、数据）状态码：200，消息：‘操作成功’ ，数据：数据库中第一级所有分类对象

方法流程：typeServicefind调用typemapper接口的SelectAllTtpe业务方法，将返回数据作为参数返回list集合（数据、消息、状态）

AllTypest

方法签名：List<Type> AllTypest();

字段：无

方法参数：无

返回值：List<Type> LIst集合（状态码、消息、数据）状态码：200，消息：‘操作成功’ ，数据：数据库中第二级所有分类对象

方法流程：typeServicefind调用typemapper接口的SelecttTtpe业务方法，将返回数据作为参数返回list集合（数据、消息、状态）

AddTypes

方法签名：String addTypes(Type t);

字段：无

方法参数：Type t Type对象

返回值：String String（状态码、消息、数据）状态码：200，消息：‘操作成功’ ，数据：操作成功或者操作失败

方法流程：typeServicefind调用typemapper接口的addTypem业务方法，将返回当前操作行数（数据、消息、状态）

DelTypes

方法签名：String delTypes(Long i);

字段：无

方法参数：Long i 需要删除分类的id

返回值：String String（状态码、消息、数据）状态码：200，消息：‘操作成功’ ，数据：操作成功或者操作失败

方法流程：typeServicefind调用typemapper接口的delTypest业务方法，将返回当前操作行数（数据、消息、状态）

selectmerchant2

方法签名：List<Type> selectmerchant2(Long i);

字段：无

方法参数：Long i 需要需要查询商店分类的商店id

返回值：List<Type> LIst集合（状态码、消息、数据）状态码：200，消息：‘操作成功’ ，数据：数据库中该店铺的所有分类

方法流程：typeServicefind调用typemapper接口的Selectmer业务方法，将返回数据作为参数返回list集合（数据、消息、状态）

###### serviceImp

业务逻辑：TypeController调用TypeServiceImp（实现TypeService接口）的方法

AllTypes

方法签名：List<Type> AllTypes();

字段：无

方法参数：无

返回值：List<Type> LIst集合（状态码、消息、数据）状态码：200，消息：‘操作成功’ ，数据：数据库中第一级所有分类对象

方法描述：实现接口TypeService中AllTypes方法所描述的所有业务逻辑

AllTypest

方法签名：List<Type> AllTypest();

字段：无

方法参数：无

返回值：List<Type> LIst集合（状态码、消息、数据）状态码：200，消息：‘操作成功’ ，数据：数据库中第二级所有分类对象

方法描述：实现接口TypeService中AllTypest方法所描述的所有业务逻辑

AddTypes

方法签名：String addTypes(Type t);

字段：无

方法参数：Type t Type对象

返回值：String String（状态码、消息、数据）状态码：200，消息：‘操作成功’ ，数据：操作成功或者操作失败

方法描述：实现接口TypeService中addTypes方法所描述的所有业务逻辑

DelTypes

方法签名：String delTypes(Long i);

字段：无

方法参数：Long i 需要删除分类的id

返回值：String String（状态码、消息、数据）状态码：200，消息：‘操作成功’ ，数据：操作成功或者操作失败

方法描述：实现接口TypeService中delTypes方法所描述的所有业务逻辑

selectmerchantt

方法签名：List<Type> selectmerchant2(Long i);

字段：无

方法参数：Long i 需要需要查询商店分类的商店id

返回值：List<Type> LIst集合（状态码、消息、数据）状态码：200，消息：‘操作成功’ ，数据：数据库中该店铺的所有分类

方法描述：实现接口TypeService中selectmerchantt方法所描述的所有业务逻辑

##### com.taoqi.type.Controller

###### 业务逻辑：TypeController调用TypeServiceImp（实现TypeService接口）的方法

###### AllType

业务逻辑：前端或客户端（游客）提交 HTTP GET 请求到 TypeController的 AllTypes方法

字段：无

方法签名：List<Type> AllTypes();

方法返回JSON格式数据

方法请求映射为路径URL: /type/classification

方法参数：无

方法流程：AllTypes方法调用 TypeService中的 SelectAllTtpe业务方法返回JSON数据

###### AllType2

业务逻辑：前端或客户端（游客）提交 HTTP GET 请求到 TypeController的 AllTypest方法

字段：无

方法签名：List<Type> AllTypest();

方法返回JSON格式数据

方法请求映射为路径URL: /type/[subclassification](E:/%E6%9C%89%E9%81%93%E7%BF%BB%E8%AF%91/Dict/8.5.1.0/resultui/html/index.html" \l "/javascript:;)/

方法参数：无

方法流程：AllTypes2方法调用 TypeService中的 SelectAllTtpe业务方法返回JSON数据

###### AddType

业务逻辑：前端或客户端（系统）提交 HTTP POST 请求到 TypeController的 addTypes方法:

字段：无

方法签名：String addTypes(Type t);

方法返回String数据

方法请求映射为路径URL: /type/addclassification/{type}

方法参数：type对象

方法流程：addTypes方法调用 TypeService中的 AddTypem业务方法返回String数据

###### DelType

业务逻辑：前端或客户端（系统）提交 HTTP POST 请求到 TypeController的 delTypes方法:

字段：无

方法签名：String delTypes(Type i);

方法返回String数据

方法请求映射为路径URL: /type/del[subclassification](E:/%E6%9C%89%E9%81%93%E7%BF%BB%E8%AF%91/Dict/8.5.1.0/resultui/html/index.html" \l "/javascript:;)/{type：id}

方法参数：type.id

方法流程：delTypes方法调用 TypeService中的 DelTypest业务方法返回String数据

###### Selectmerchant

业务逻辑：前端或客户端（系统）提交 HTTP GET 请求到 TypeController的 delTypes方法:

字段：无

方法签名：List<Type> selectmerchant(Long i);

方法返回String数据

方法请求映射为路径URL: /type/[merchant](E:/%E6%9C%89%E9%81%93%E7%BF%BB%E8%AF%91/Dict/8.5.1.0/resultui/html/index.html" \l "/javascript:;)/{Shop：id}

方法参数：type.id

方法流程：selectmerchantt方法调用 TypeService中的 selectmerchantt业务方法返回String数据

### 推广模块

##### com.taoqi.spread

###### 启动类：SpreadServiceApp

###### 实体类：Spread

##### com.taoqi.spread.mapper

###### 业务逻辑SpreadService依赖于SpreadMapper（接口名）并定义其方法

###### 接口名：SpreadMapper

方法：

1. listAllSpread

方法签名：**List<VSpread> listAllSpread()**

类型：抽象方法

参数：无

操作的表/视图：t\_spread（具体表）和v\_spread(视图)

返回值：List<VSpread> VO/DTO实体集合(推广信息集合)，商城所对应的所有推广信息

描述：使用SQL语句查询数据库中t\_spread表和视图v\_spread，并且将其封装到List<VSpread>，然后在将List<VSpread>实体集合返回

1. list

方法签名：**List<VSpread> list()**

类型：抽象方法

参数：无

操作的表/视图：t\_spread（具体表）和v\_spread(视图)

返回值：List<VSpread> VO/DTO实体集合(推广信息集合)，商城所对应的竞价前三的推广信息

方法描述：使用SQL语句排序后查询数据库中t\_spread表和视图v\_spread中排名前三的信息，并且将其封装到List<VSpread>，然后在将List<VSpread>实体集合返回

1. insertSpread

方法签名: **int insertSpread(int userid,String imgsrc,String info,float price)**

类型：抽象方法

参数：userid int 用户id

info String 推广信息

imgsrc String 推广图片

price float 价格

操作的表/视图：t\_spread

返回值：int 转台编号1/0（1：插入数据成功；0：插入数据失败）

方法描述：通过传入的参数，使用SQL语句将其数据插入到t\_spread表中

1. deleteThreeSpread

方法签名：**int deleteThreeSpread()**

类型：抽象方法

参数：spread Spread 所要插入的评论对象

操作的表/视图：t\_spread

返回值：int 转台编号1/0（1：删除数据成功；0：删除数据失败）

方法描述：使用SQL语句将数据库的表t\_spread中除前三名以外的推广信息从t\_spread中移除

1. deleteSpreadById

方法签名：**int deleteSpreadById(int id)**

类型：抽象方法

参数：Id int 推广编号

操作的表/视图：t\_spread

方法描述：通过评论编号使用SQL语句将数据库的表t\_spread中所对应评论编号的所有信息从t\_spread中移除

返回值：int 状态编号 1/0（1：删除数据成功；0：删除数据失败）

##### com.taoqi.spread.service（接口类）

###### 业务逻辑：SpreadServiceImp依赖于SpreadService（接口名）并定义其方法

###### 接口名：SpreadService

方法：

1、loadList

方法签名：**List<Spread> loadList()**

方法类型：抽象方法

方法参数：无

返回值：List集合（数据、消息、状态码）状态码：200，消息：‘操作成功’ ，数据：竞价排名前三的推广信息

方法流程：loadList调用SpreadMapper接口的list业务方法 ，返回对应List集合（数据）

2、loadListAll

方法签名：**List<Spread> loadListAll()**

方法类型：抽象方法

方法参数：无

返回值：List集合（数据、消息、状态码）状态码：200，消息：‘操作成功’ ，数据：所有推广信息

方法流程：loadListAll调用SpreadMapper接口的listAllSpread业务方法，返回对应List集合（数据）

3、deleteById

方法签名：**int deleteById(int id)**

方法类型：抽象方法

方法参数：int id 推广编号

返回值：int 转台编号1/0（1：删除数据成功；0：删除数据失败）

方法流程：deleteById调用SpreadMapper接口的deleteSpreadById业务方法

4、deleteThree

方法签名：**int deleteThree()**

方法类型：抽象方法

方法参数：无

返回值：int 转台编号1/0（1：删除数据成功；0：删除数据失败）

方法流程：deleteThree调用SpreadMapper接口的deleteThreeSprread业务方法

5、addSpread

方法签名：**int addSpread(int userid,String info,String imgsrc,float price)**

方法类型：抽象方法

方法参数：userid int 用户id

info String 推广信息

imgsrc String 推广图片

price float 价格

返回值：int 转台编号1/0（1：删除数据成功；0：删除数据失败）

方法流程：addSpread调用SpreadMapper接口的inserttSpread业务方法

##### com.taoqi.spread.service.Impl

###### 业务逻辑：SpreadController调用SpreadServiceImp（实现SpreadService接口）的方法

###### 类名：SpreadServiceImp

方法

1. loadList

方法签名：**public List<Spread> loadList()**

方法描述：实现接口SpreadService中loadList方法所描述的所有业务逻辑

1. LoadListAll

方法签名：**public List<Spread> loadListAll()**

方法描述：实现接口SpreadService中loadListAll方法所描述的所有业务逻辑

1. deleteById

方法签名：**public int deleteById(int id)**

方法描述：实现接口SpreadService中deleteById方法所描述的所有业务逻辑

1. deleteThree

方法签名：**public int deleteThree()**

方法描述：实现接口SpreadService中deleteThree方法所描述的所有业务逻辑

1. addSpread

方法签名：**public int addSpread(int userid,String info,String imgsrc,float price)**

方法描述：实现接口SpreadService中addSpread方法所描述的所有业务逻辑

##### com.taoqi.spread.controller

类名：SpreadController

###### 方法

1. getAllSpread

业务逻辑：前端或客户端（Admin）提交 HTTP GET 请求到 SpreadController的 getAllSpread方法

方法签名：**public List<Spread> getAllSpread()**

返回值：JSON格式数据(状态码、消息、数据)

方法请求映射为路径URL：/spread/getall

方法参数：无

方法流程：getAllSpread方法调用 SpreadService中的 loadListAll业务方法返回JSON数据(状态码、消息、数据)

1. getThreeSpread

业务逻辑：前端或客户端（Admin）提交 HTTP GET 请求到 SpreadController的 getThreeSpread方法

方法签名：**public List<Spread> getThreeSpread()**

返回值：JSON格式数据(状态码、消息、数据)

方法请求映射为路径URL: /spread/get

方法参数：无

方法流程：getShopSpread方法调用 SpreadService中的 loadList业务方法返回JSON数据(状态码、消息、数据)

1. insertSpread

业务逻辑：前端或客户端（系统）提交 HTTP GET 请求到 SpreadController的 insertSpread方法

方法签名：**public Map<String, Object> insertSpread( int userid,String info,String imgsrc,float price)**

返回值：JSON格式数据(状态码、消息、数据)

方法请求映射为路径URL： /spread/insert

方法参数：userid int 用户id

info String 推广信息

imgsrc String 推广图片

price float 价格

方法流程：insertSpread方法调用 SpreadService中的 addSpread业务方法，若返回数据为1，则将字符串“操作成功”作为参数调用ResultUtil的succes方法返回map集合（数据、消息、状态）,若返回数据为0，则将字符串“操作失败”作为参数调用ResultUtil的error方法返回map集合（数据、消息、状态）

1. deleteSpreadById

业务逻辑：前端或客户端（用户）提交 HTTP GET 请求到 SpreadController的 deleteSpreadById方法

方法签名：**public Map<String, Object> deleteSpreadById( int id)**

返回值：JSON格式数据(状态码、消息、数据)

方法请求映射为路径URL: /spread/deletebyid/{id}

方法参数：id int 推广信息编号

方法流程：deleteSpreadById方法调用 SpreadService中的 deleteById业务方法，若返回数据为1，则将字符串“操作成功”作为参数调用ResultUtil的succes方法返回map集合（数据、消息、状态）,若返回数据为0，则将字符串“操作失败”作为参数调用ResultUtil的error方法返回map集合（数据、消息、状态）

1. deleteSpread

业务逻辑：前端或客户端（系统）提交 HTTP GET 请求到 SpreadController的 deleteSpread方法:

字段：无

方法签名：**public Map<String, Object>deleteSpread(long id)**

方法返回JSON格式数据

方法请求映射为路径URL: spread/uget/{priductid}

方法参数：

方法流程：deleteSpread方法调用 SpreadService中的 deleteThree业务方法，若返回数据为1，则将字符串“操作成功”作为参数调用ResultUtil的succes方法返回map集合（数据、消息、状态）,若返回数据为0，则将字符串“操作失败”作为参数调用ResultUtil的error方法返回map集合（数据、消息、状态）

### 订单模块

##### com.taoqi.Order.mapper（接口类）

###### 业务逻辑OrderService依赖于OrderMapper（接口名）并定义其方法

方法：

###### OrderMapper

1、 List<Order> findAllOrder();

类型：抽象方法

方法描述：系统管理员查看所有订单

返回值：List<Order> VO/DTO实体集合(评论信息集合)

操作表：t\_order

2、 List<Order> findUserOrder(long id);

类型：抽象方法

方法描述：会员查看已有的订单

返回值：List<Order> VO/DTO实体集合 (评论信息集合)

操作表：t\_order

3、 int addOrder(Order order);

类型：抽象方法

方法描述：用户添加订单

返回值： int 1/0（1：插入成功 0:插入失败）

操作表：t\_order

4、 int delectUserOrder(long id)

类型：抽象方法

方法描述：用户通过订单id删除一条订单

返回值：int 1/0（1：插入成功 0:插入失败）

操作表：t\_order

字段：flag (删除标记)

1. int updateOrder(int stat,long id);

类型：抽象方法

方法描述：通过订单修改一条订单

返回值：int 2/1/0（2：已签收 1：已发货 0:未处理）

操作表：t\_order

字段：stat (处理标记)

#### com.taoqi.order.service（接口类）

###### 业务逻辑：OrderServiceImp依赖于OrderService（接口名）并定义其方法

###### OrderService

List<Order> getsSelectAll();

方法参数：

返回值：List<Order> List集合

方法流程：getOrderFindAll 调用OrderMapper接口的findAllOrder实现方法，将返回数据作文参数调用ResultUtil的succes方法返回map集合（数据、消息、状态）

List<Order> getSelectUser();

字段：无

方法参数：

返回值：List<Order> List集合

方法流程：getOrderFindUser调用OrderMapper接口的findUserOrder业务方法，将返回数据作为参数调用ResultUtil的succes方法返回map集合（数据、消息、状态）

Int getOrderAdd(Order order);

字段：无

方法参数：Order order

返回值：0 或 1 （0失败，1成功）

方法流程：getOrderAdd调用OrderMapper接口的addOrder业务方法，将返回数据为0则将“操作失败”作为参数调用ResultUtil的error方法，若返回数据为1则将“操作成功”作为参数调用ResultUtil的error方法

int getOrderDelectUser((long id);

字段：无

方法参数： id long订单编号

返回值：0 或 1 （0失败，1成功）

方法流程：getOrderDelectUser调用OrderMapper接口的delectUserOrder业务方法，将返回数据为0则将“未删除”作为参数调用ResultUtil的error方法，若返回数据为1则将“已删除”作为参数调用ResultUtil的error方法

int getOrderUpdate(long stat,long id);

字段：无

方法参数：stat long 处理标记, id long 订单编号

返回值：0 或 1 或 2 （0未处理，1已发货，2已签收）

方法流程：getOrderUpdate调用OrderMapper接口的updateOrder业务方法，将返回数据为0则将“未处理”作为参数调用ResultUtil的error方法，若返回数据为1则将“已发货”作为参数调用ResultUtil的error方法，若返回数据为2则将“已签收”作为参数调用ResultUtil的error方法

#### com.taoqi.order.service.Impl

###### OrderController调用OrderServiceImp（实现OrderService接口）的方法

###### OrderServiceImp

1. public List<Order> getOrderFindAll ()

方法参数：

返回值：List<Order> List集合

方法流程：getOrderFindAll 调用OrderMapper接口的findAllOrder实现方法，将返回数据作文参数调用ResultUtil的succes方法返回map集合（数据、消息、状态）

1. public List<Order> getOrderFindUser()

字段：无

方法参数：

返回值：List<Order> List集合

方法流程：getOrderFindUser调用OrderMapper接口的findUserOrder业务方法，将返回数据作为参数调用ResultUtil的succes方法返回map集合（数据、消息、状态）

1. public int getOrderAdd(Order order)

字段：无

方法参数：Order order

返回值：0 或 1 （0失败，1成功）

方法流程：getOrderAdd调用OrderMapper接口的addOrder业务方法，将返回数据为0则将“操作失败”作为参数调用ResultUtil的error方法，若返回数据为1则将“操作成功”作为参数调用ResultUtil的error方法

4、 public int getOrderDelectUser(long id)

字段：无

方法参数： id long订单编号

返回值：0 或 1 （0失败，1成功）

方法流程：getOrderDelectUser调用OrderMapper接口的delectUserOrder业务方法，将返回数据为0则将“未删除”作为参数调用ResultUtil的error方法，若返回数据为1则将“已删除”作为参数调用ResultUtil的error方法

5、 public int getOrderUpdate(**long** stat,**long** id)

字段：无

方法参数：stat long 处理标记, id long 订单编号

返回值：0 或 1 或 2 （0未处理，1已发货，2已签收）

方法流程：getOrderUpdate调用OrderMapper接口的updateOrder业务方法，将返回数据为0则将“未处理”作为参数调用ResultUtil的error方法，若返回数据为1则将“已发货”作为参数调用ResultUtil的error方法，若返回数据为2则将“已签收”作为参数调用ResultUtil的error方法

##### com.taoqi.order.controller

###### 类名：OrderController

###### 业务逻辑：前端或客户端（系统）提交 HTTP GET 请求到 OrderController的getSelectAll方法

字段：无

**public** List<Order> list()

方法返回JSON格式数据(状态码、消息、数据)

方法请求映射为路径URL: /order/selectAll

方法参数：

方法流程：getSelectAll方法调用 OrderService中的 selectAll业务方法返回JSON数据(状态码、消息、数据)

###### 业务逻辑：前端或客户端（用户）提交 HTTP GET 请求到OrderController的getSelectUser方法

字段：无

**public** List<Order> listOrder()

方法返回JSON格式数据

方法请求映射为路径URL: /order/ selectUser

方法参数：

Orderid long 订单编号

方法流程：getSelectUser方法调用OrderService中的 selectUser业务方法返回JSON数据

###### 业务逻辑：前端或客户端（用户）提交 HTTP GET 请求到OrderController的getOrderAdd方法:

字段：无

**public int** getAddOrder(Order order)

方法返回:0或1

方法请求映射为路径URL: /order/ addOrder

方法参数：无

方法流程：getOrderAdd方法调用OrderService中的 rderAdd业务方法返回JSON数据

###### 业务逻辑：前端或客户端（用户）提交 HTTP GET 请求到OrderController的getOrderDelectUser方法:

字段：无

**public int** updateOrder(long stat ,long id)

方法返回0或1

方法请求映射为路径URL: /order/ delectUserOrder

方法参数：

方法流程：getOrderDelectUser方法调用OrderService中的orderDelectUser业务方法返回JSON数据

###### 业务逻辑：前端或客户端（系统）提交 HTTP GET 请求到OrderController的getOrderUpdate方法:

字段：无

**public int** getOrderUpdate (long id)

方法返回0或1或2

方法请求映射为路径URL: /order/ updateOrder

方法流程：getOrderUpdate方法调用OrderService中的orderUpdate业务方法返回JSON数据

### 公共类

##### com.taoqi.utils

###### 类名：ResultUtil

方法：

1、result

方法签名：**public static Map<String, Object> result(int code, String msg, Object data)**

字段：无

方法参数：

Code int 状态码

Msg String 消息

Data Object 数据

返回值：Map<String, Object> map集合（状态码、消息、数据）

方法流程：实例化一个Map集合，将方法参数添加到Map集合之中，其中的数据

如：“Code”: code

再将数据返回

2、succes

方法签名：**public static Map<String, Object> succes()**

方法参数：无

返回值：Map<String, Object> map集合（状态码、消息）

方法描述：调用result,填入默认参数（200，”操作成功”，null）

再将其数据返回

1. succes

方法签名：public static Map<String, Object> succes(String msg)

方法参数：

msg String 消息

返回值：Map<String, Object> map集合（状态码、消息）

方法描述：调用result,填入默认参数（200，msg，null）

再将其数据返回

4、succes

方法签名：**public static Map<String, Object> succes(Object data)**

方法参数：data Object 数据

返回值：Map<String, Object> map集合（状态码、消息、数据）

方法描述：调用result,填入默认参数（200，”操作成功”，data）

再将其数据返回

5、error

方法签名：public static Map<String, Object> error()

方法参数：无

返回值：Map<String, Object> map集合（状态码、消息）

方法描述：调用result,填入默认参数（200，”操作成功”，null）

再将其数据返回

6、error

方法签名：**public static Map<String, Object> error(String msg)**

方法参数：msg String 消息

返回值：Map<String, Object> map集合（状态码、消息）

方法描述：调用result,填入默认参数（500，msg，null）

再将其数据返回

7、error

方法签名：**public static Map<String, Object> error(Object data)**

方法参数：data Object 数据

返回值：Map<String, Object> map集合（状态码、消息、数据）

方法描述：调用result,填入默认参数（500，”操作失败”，data）

再将其数据返回