**自营电商详细设计**

# 自营电商详细设计

## 用户模块

### 注册

前端或客户端提交 HTTP POST 请求到 customerController

的 add方法:

方法签名：public HashMap add(相关参数)

该方法返回 JSON 格式数据：

方法请求映射为路径（URL）：api/customer/register

参数：

Email 用户邮箱 String

Password 用户密码 String

方法流程：add方法调用 customerService中的 register业务方法，

若注册成功则返回用户对象并添加对象（调用customerMapper）

若捕获 RegisterException 异常则返回：用户名已存在

### 登录

前端或客户端提交 HTTP POST 请求到 customerController的 login方法:

方法签名：public HashMap login(相关参数)

该方法返回 JSON 格式数据：

方法请求映射为路径（URL）：api/customer/Signin

参数：

Phone 用户手机号 String

Password 用户密码 String

方法流程：login方法调用 customerService中的 Signin业务方法，

若返回true则登录成功（调用customerMapper）

若返回false则捕获 SigninException异常则返回：用户名或密码错误

### 客户信息编辑

前端或客户端提交 HTTP POST 请求到 customerController的 update方法:

方法签名：public List<> update(相关参数)

该方法返回 JSON 格式数据：

方法请求映射为路径（URL）：api/customer/info

参数：

Netname 用户昵称 String

phone手机号 String

Password 用户密码 String

方法流程：update方法调用 customerService中的 info业务方法，

若新旧密码一致则修改成功（调用customerMapper）

若新旧密码不一致则捕获 RegisterException 异常则返回：密码验证错误

### 添加收货地址

前端或客户端提交 HTTP POST 请求到 customerController的 adds方法:

方法签名：public List<> adds(相关参数)

该方法返回 JSON 格式数据：

方法请求映射为路径（URL）：api/customer/info

参数：

name 名字 String

phone手机号 String

Address 地址 String

方法流程：adds方法调用 customerService中的 Address业务方法，

添加地址（调用customerMapper）

### 足迹

前端实现点击商品获得用户ID，商品对象，浏览时间提交 HTTP POST 请求到 HomeController 的 foots方法:

方法签名：public list<History > foots(相关参数)

该方法返回 JSON 格式数据：

方法请求映射为路径（URL）：api/customer/footprint

参数：

ID 用户ID Int

Commodity 商品对象 Object

Time 浏览时间 Data

方法流程：foots方法调用 customerService中的 foot业务方法，

添加到足迹（调用customerMapper）

方法签名：public list <History > foot (int id )

查看客户浏览过的商品

该方法返回 list<History >：

参数：ID 用户ID

异常：无

### 收藏

前端实现点击商品获得用户ID，商品对象，浏览时间提交 HTTP POST 请求到 customerController的 Collections 方法:

方法签名：public list<Collection > Collections(相关参数)

该方法返回 JSON 格式数据：

方法请求映射为路径（URL）：api/customer/footprint

参数：

ID 用户ID Int

Commodity 商品对象 Object

Price 收藏时商品价格 BigDecimal

Time 浏览时间 Data

方法流程：Collections 方法调用 customerService中的 Collection 业务方法，

添加到收藏（调用customerMapper）

public list <Collection > Collection (int id )

查看客户收藏的商品

参数：ID 用户ID

返回值：list集合

异常：无

## 商品模块

### 查询不同类别的商品

#### Mapper

方法签名 public List<Commodity> selectCommodityCategory(int categoryid)

方法描述 返回某个类别的产品用于商品的分类显示

使用表： 商品表 commodity

使用字段 title (商品名称)

description (商品描述)

price(商品价格)

stock(商品库存)

类别表 category

参数 int categoryid 商品类别id

返回值 List<Commodity> 根据传入参数返回一个指定类别的商品集合

异常 无

#### Service

方法签名 public PageInfo<Commodity> displayCommodityCategory(int categoryid)

方法描述 获得该类别所有商品进行分页操作

方法流程 1.根据前端发送过来的具体的类别categoryid调用CommunityMapper

中的selectCommodityCategory()方法查询商品信息

2.使用PageHelper中的startPage 设置分页信息

3.调用PageInfo<Commdtity>的构造方法将查询信息传入

4.将Pagehelper打包好的PageInfo<Commdtity>返回

参数 int categoryid 类别id

返回值 PageInfo<Commodity>

异常 无

#### Controller

方法签名 public PageInfo<Commodity> displayCommodity (int categoryid)

方法描述 返回商品分类的列表

请求类型 GET

接口路径 api/commodity/getcategory

方法流程 1.根据前端发送过来的具体的类别categoryid调用CommunityService

中的displayCommodityCategory ()方法查询商品信息

2.将返回回来的PageInfo返回给前端

参数 int categoryid 类别id

返回值 PageInfo<Commodity>

异常 无

### 查询被推荐的商品

#### Mapper

方法签名 public List<Commodity> selectCommodityRecommed()

方法描述 返回所有被推荐的商品

使用表： 商品表 commodity

使用字段 id 商品标识

title (商品名称)

description (商品描述)

price(商品价格)

stock(商品库存)

recommend(是否推荐)

参数 无

返回值 List<Commodity> 所有被推荐商品的集合

异常 无

#### Service

方法签名 public PageInfo<Commodity>displayCommodityRecommed()

方法描述 获得所有被推荐的商品进行分页操作

业务流程 1.调用CommunityMapper的selectCommodityRecommed()方法

返回所有被推荐的商品

2.使用PageHelper中的startPage 设置分页信息

3.调用PageInfo<Commdtity>的构造方法将查询信息传入

4. 将Pagehelper打包好的PageInfo<Commdtity>返回

参数 无

返回值 PageInfo<Commodity>

异常 无

#### Contoller

方法签名 public PageInfo<Commodity>displayRecommed()

方法描述 获得所有被推荐的商品

请求类型 GET

接口路径 api/commodity/getrecommed

业务流程 1.根据前端发送过来的具体的类别categoryid调用CommunityService

中的displayCommodityRecommed ()方法查询商品信息

2.将返回回来的PageInfo<Commodity>返回给前端

参数 无

返回值 PageInfo<Commodity>

异常 无

### 查询限量折扣商品

#### Mapper

方法签名 public List<Commodity> selectCommoditySpike()

方法描述 返回限量折扣商品

使用表： 秒杀活动表spike

使用字段 community\_id 折扣商品id

start\_time 活动开始时间

end\_time 活动结束时间

stock 活动商品数量

price 秒杀价格

商品表 commodity

使用字段 title (商品名称)

description (商品描述)

price(商品价格)

stock(商品库存)

参数 无

返回值 List<Commodity> 限量折扣商品集合

异常 无

#### Service

方法签名 public PageInfo<Commodity> selectCommoditySpike()

方法描述 获得限量折扣商品进行分页

业务流程 1.调用CommunityMapper中的selectCommoditySpike()返回所有限量折扣

的商品集合

2.使用PageHelper中的startPage 设置分页信息

3.调用PageInfo<Commdtity>的构造方法将查询信息传入

4.将PageHelper打包好的PageInfo<Commdtity>返回

参数 无

返回值 PageInfo<Commodity>

异常 无

#### Controller

方法签名 public PageInfo<Commodity> querySpike()

方法描述 获得限量折扣商品

请求类型 GET

接口路径 api/commodity/getspike

业务流程 1.调用CommunityService中的selectCommoditySpike ()方法查询

商品信息

2.将返回回来的PageInfo<Commodity>返回给前端

参数 无

返回值 PageInfo<Commodity>

异常 无

### 查询限时折扣商品

#### Mapper

方法签名 public List<Commodity> selectCommodityDiscount ()

方法描述 返回限时折扣商品

使用表： 秒杀活动表 discount

community\_id(活动商品id)

start\_time(开始时间)

end\_time(结束时间)

price(折扣价格)

商品表 commodity

使用字段 title (商品名称)

description (商品描述)

price(商品价格)

stock(商品库存)

参数 无

返回值 List<Commodity> 限时折扣商品价格

异常 无

#### Service

方法签名 public PageInfo<Commodity> selectCommodityDiscount()

方法描述 获得限量折扣商品进行分页

方法流程 1.调用调用CommunityMapper中的selectCommodityDiscount()返回

所有限时折扣

2.使用PageHelper中的startPage 设置分页信息

3.调用PageInfo<Commdtity>的构造方法将查询信息传入

4.将PageHelper打包好的PageInfo<Commdtity>返回

参数 无

返回值 PageInfo<Commodity>

异常 无

#### Controller

方法签名 public PageInfo<Commodity> queryCommodityDiscount()

方法描述 获得限量折扣商品

请求类型 GET

接口路径 api/commodity/getdiscount

业务流程 1.调用CommunityService中的queryDiscount ()方法查询

商品信息

2.将返回回来的PageInfo<Commodity>返回给前端

参数 无

返回值 PageInfo<Commodity>

异常 无

### 根据品牌查询商品

#### Mapper

方法签名 public List<Commodity> selectCommodityByBrand(int brandid)

方法描述 返回该品牌的所有商品

使用表： 商品表commodity

使用字段 title (商品名称)

description (商品描述)

price(商品价格)

stock(商品库存)

品牌表brand

使用字段 id(品牌id)

参数 int brandid 品牌id

返回值 List<Commodity>

异常 无

#### Service

方法签名 public PageInfo<Commodity> queryCommodityByBrand(int brandid)

方法描述 获得该品牌的所有商品进行分页

业务流程 1.根据前端发送过来的具体的类别brandid调用CommunityMapper

中的selectCommodityByBrand ()方法查询商品信息

2.使用PageHelper中的startPage 设置分页信息

3.调用PageInfo<Commdtity>的构造方法将查询信息传入

3.将Pagehelper打包好的PageInfo<Commdtity>返回

参数 int brandid 品牌id

返回值 PageInfo<Commodity>

异常 无

#### Controller

方法签名 public PageInfo<Commodity> queryCommodityByBrand(int brandid)

方法描述 获得该品牌的所有商品

请求类型 GET

接口路径 api/commodity/getbrand

业务流程 1.根据前端发送过来的具体的类别brandid调用CommunityService

中的queryCommodityByBrand ()方法查询商品信息

2.将返回回来的PageInfo<Commodity>返回给前端

参数 int brandid 品牌id

返回值 PageInfo<Commodity>

异常 无

### 根据关键字查询商品

#### Mapper

方法签名 public List<Commodity> selectCommodityByName

方法描述 返回带有关键字的所有商品

使用表： 商品表 Commodity

使用字段 title (商品名称)

description (商品描述)

price(商品价格)

stock(商品库存)

参数 String Commodityname 商品关键字

返回值 List<Commodity> 带有关键字的商品

异常 无

#### Service

方法签名 public PageInfo<Commodity> queryCommodityByName(String commodityname)

方法描述 获得带有关键字的所有商品进行分页

业务流程 1.根据前端发送来的commodityname调用CommunityMapper

中的selectCommodityByName()方法查询带有关键字的商品信息

2.使用PageHelper中的startPage 设置分页信息

3.调用PageInfo<Commdtity>的构造方法将查询信息传入

3.将Pagehelper打包好的PageInfo<Commdtity>返回

参数 String Commodityname 商品关键字

返回值 PageInfo <Commodity>

异常 无

#### Controller

方法签名 public PageInfo<Commodity> queryCommodityByName(String commodityname)

方法描述 获得带有关键字的所有商品

请求类型 GET

接口路径 api/commodity/getkeyword

业务流程 1.根据前端发送过来的具体的类别commodityname

调用CommunityService

中的queryCommodityByName ()方法查询商品信息

2.将返回回来的PageInfo<Commodity>返回给前端

参数 String Commodityname 商品关键字

返回值 PageInfo <Commodity>

异常 无

### 查询商品的详细信息

#### Mapper

方法签名 public Commodity selectCommodity(int Commodityid)

方法描述 返回指定商品的详细信息

使用表： 商品表 commodity

使用字段 title (商品名称)

description (商品描述)

price(商品价格)

stock(商品库存)

参数 int Commodityid 商品id

返回值 Commodity 商品详细信息

异常 无

#### Service

方法签名 public Commodity queryCommodity(int commodityid)

方法描述 获得该商品的详细信息

业务逻辑 1.根据前端发送来的commodityid调用CommunityMapper

中的selectCommodity ()方法查询带有关键字的商品信息

2.将查询的信息返回

参数 int Commodityid 商品id

返回值 Commodity

异常 无

#### Controller

方法签名 public Commodity queryCommodity(int commodityid)

方法描述 获得该商品的详细信息

请求类型 GET

接口路径 api/commodity/getcommodity

业务逻辑 1.根据前端发送来的commodityid调用CommunityService

中的queryCommodity ()方法查询带有关键字的商品信息

2.将查询的信息返回给前端

参数 int Commodityid 商品id

返回值 Commodity

异常 无

### 往购物车项表中插入数据

#### Mapper

方法签名 public void addCart(int uid,int commodityid,int quantity,String param)

方法描述 添加购物车中新增商品信息

使用表： 购物车表cart

使用字段 uid(用户id)

community\_id(商品id)

quantity(数量)

param(商品参数)

参数 cart

返回值 void

异常 无

#### Service

方法签名 public void addCart(int commodityid,int quantity,String param)

方法描述 将商品添加至购物车

业务流程 1.接收参数commodity

2.从session中拿到当前用户的uid和Cart<Integer,CartItem>

3.根据commodityid在判断Cart中是存在相同商品。

4.如果不存在将商品信息插入Cart中，如果存在则调用CartItem的get

Param获得购物车中同类商品的Spec的Map，如果商品参数相同数量加

1调用CartService 中的changeQuantity方法修改商品数量，如果参数

不相同则在调用CommodityMapper 中的addCart方法

map中插入新数据

参数 Cart

返回值 void

异常 无

#### Controller

方法签名 public List<CartItem> addCart(int commodityid,int quantity,String param)

方法描述 将商品添加至购物车

请求类型 GET

接口路径 api/commodity/addcart

业务逻辑 1.根据前端发送来的commodityid，quantity，param调用CommunityService

中的addCart ()方法插入更新购物车信息

2.调用CartService 中的checkCart()方法返回最新的购物车列表

参数 Cart

返回值 List<CartItem>

异常 无

### 查询商品评论

#### Mapper

方法签名 public List<Comment> selectCommoditycomment(int commodityid)

方法描述 返回该商品的评论

使用表： 评论表comment

使用字段 commodity\_id(评论商品id)

uid(评论用户id)

content(用户评论)

time(评论时间)

score(评分)

参数 int commodityid 商品id

返回值 List< Comment > 该商品所有评论

异常 无

#### Service

方法签名 public PageInfo<Comment> queryCommoditycomment（int commodityid）

方法描述 获得该商品的所有评论进行分页

业务流程 1.根据前端发送来的commodityid调用CommunityMapper

中的selectCommoditycomment ()方法查询商品的评论信息 2.使用PageHelper中的startPage 设置分页信息

3.调用PageInfo< Comment >的构造方法将查询信息传入

3.将Pagehelper打包好的PageInfo< Comment >返回

参数 int commodityid 商品id

返回值 PageInfo< Comment >

异常 无

#### Controller

方法签名 public PageInfo<Comment> queryCommoditycomment（int commodityid）

方法描述 获得该商品的所有评论

请求类型 GET

接口路径 api/commodity/getcomment

业务逻辑 1.根据前端发送来的commodityid调用CommunityService

中的queryCommoditycomment ()方法查询商品的所有评论

2.将PageInfo< Comment >返回给前端

参数 int commodityid 商品id

返回值 PageInfo< Comment >

异常 无

## 购物车模块

### 查询购物车列表

##### CartController控制层：

前端发送GET请求到CartController控制器的checkCart方法

方法签名：public List<CartItem> checkCart (Int id)

方法请求映射路径：api/cart/checkCart

返回json格式数据

参数: id 客户id

调用服务层CartService的方法checkCart返回List<CartItem>

##### Cartservice服务层：

方法签名： public List<CartItem> checkCart (Int id)

方法描述： 返回购物车列表

参数： 客户id

返回值： List<CartItem>

异常： 无

方法中的业务流程：

调用接口CartMapper中checkCart方法

返回接收的List<CartItem>对象

##### CartMapper:

方法签名： public List<CartItem> checkCart (Int id)

方法描述： 返回购物车列表

使用表： 商品表，购物车表

参数： 客户id

返回值： List<CartItem>

异常： 无

方法中的业务流程：

在方法签名前加 @Select 注解

在注解里放编写好的查询 SQL 语句

给返回到结果与实体类建立映射

给参数加 @pram 注解

### 修改一个商品的数量

##### CartController控制层:

前端发送POST请求到CartController控制器的changeQuantity方法

方法签名：public void changeQuantity(int id,int sid,int quantity)

方法请求映射路径：api/cart/changeQuantity

返回json格式数据

参数: id 客户id，

sid 商品规格id,

quantity 数量

调用服务层CartService的方法changeQuantity

##### CartService服务层:

方法签名： public void changeQuantity(int id,int sid,int quantity)

方法描述： 修改商品的数量

参数： id 客户id，

sid 商品规格id,

quantity 数量

返回值： 无

异常： 无

方法中的业务流程：

从cartItem里获取商品id和购物车项参数对象集合List<CartItemParam>

遍历购物车项参数对象集合

调用接口CartMapper中的changeQuantity方法

##### CartMapper:

方法签名： public void changeQuantity(int id,int sid,int quantity)

方法描述： 修改一个商品的数量

使用表： 购物车表

参数： id 客户id，

sid 商品规格id,

quantity 数量

返回值： 无

异常： 无

方法中的业务流程：

在方法签名前加 @Update 注解

在注解里放编写好的 SQL 语句

给参数加 @pram 注解

### 删除一个购物车项

##### CartController控制层:

前端发送POST请求到CartController控制器的changeQuantity方法

方法签名：public void delete(int uid ,int sid)

方法请求映射路径：api/cart/delete

返回json格式数据

参数: id 客户id，

sid 商品规格id

调用服务层CartService的方法delete

##### CartService服务层:

方法签名： public void delete(int uid ,int sid)

方法描述： 删除一个购物车项

参数： uid 客户id

Sid 商品规格id

返回值： 无

异常： 无

方法中的业务流程：

调用接口CartMapper中的三个参数的delete方法

##### CartMapper:

方法签名： public void delete(int uid ,int sid)

方法描述： 删除一个购物车项

使用表： 购物车表

参数： uid 客户id

Sid 商品规格id

返回值： 无

异常： 无

方法中的业务流程：

在方法签名前加 @Delect 注解

在注解里放编写好的 SQL 语句

给参数加 @pram 注解

### 删除一个商品

##### CartController控制层:

前端发送POST请求到CartController控制器的deleteCustomer方法

方法签名：public void delete(int uid ,int cid)

方法请求映射路径：api/cart/deleteCustomer

返回json格式数据

参数: uid 客户id，

cid 商品id

调用服务层CartService的方法deleteCustomer

##### CartService服务层:

方法签名： public void delete(int uid ,int cid)

方法描述： 删除一个商品

参数： uid 客户id

cid 商品id

返回值： 无

异常： 无

调用接口CartMapper中的两个参数的deleteCustomer方法

##### CartMapper:

方法签名： public void delete(int uid ,int cid)

方法描述： 删除一个商品

使用表： 购物车表

参数： uid 客户id

cid 商品id

返回值： 无

异常： 无

方法中的业务流程：

在方法签名前加 @Delect 注解

在注解里放编写好的 SQL 语句

给参数加 @pram 注解

### 根据商品id查找购物车商品

##### CartController控制层:

前端发送GET请求到CartController控制器的findCartById方法

方法签名：public CartItem findCartById (int uid ,int cid)

方法请求映射路径：api/cart/findCartById

返回json格式数据

参数: uid 客户id

cid 商品id

调用服务层CartService的方法findCartById返回CartItem

##### CartService服务层:

方法签名： public CartItem findCartById (int uid ,int cid)

方法描述： 根据商品id查找购物车商品

使用表： 购物车表

参数： uid 客户id ,

cid 商品id

返回值： CartItem 购物车项

异常： 无

方法中的业务流程：

调用接口CartMapper中的findCartById 方法

返回接收到 CartItem对象

##### CartMapper:

方法签名： public CartItem findCartById (int uid ,int cid)

方法描述： 根据商品id查找购物车商品

使用表： 购物车表

参数： 客户uid ,商品 cid

返回值： CartItem 购物车项

异常： 无

方法中的业务流程：

在方法签名前加 @Select 注解

在注解里放编写好的 SQL 语句

给返回到结果与实体类建立映射

给参数加 @pram 注解

### 根据商品关键字查找购物车商品

##### CartController控制层:

前端发送GET请求到CartController控制器的findCartByName方法

方法签名：public CartItem findCartByName (int uid ,String name)

方法请求映射路径：api/cart/findCartById

返回json格式数据

参数: uid 客户id

name 商品关键字

调用服务层CartService的方法findCartById返回CartItem

##### CartService服务层:

方法签名： public CartItem findCartById (int uid ,String name)

方法描述： 根据商品id查找购物车商品

使用表： 购物车表

参数： uid 客户id ,

name 商品关键字

返回值：CartItem 购物车项

异常： 无

方法中的业务流程：

调用接口CartMapper中的findCartById 方法

返回接收到 CartItem对象

##### CartMapper:

方法签名： public CartItem findCartById (int uid ,String name)

方法描述： 根据商品id查找购物车商品

使用表： 购物车表

参数： uid 客户id ,

name 商品关键字

返回值：CartItem 购物车项

异常： 无

方法中的业务流程：

在方法签名前加 @Select 注解

在注解里放编写好的 SQL 语句

给返回到结果与实体类建立映射

给参数加 @pram 注解

## 订单模块

### CustomerOrder

#### addOrder

方法描述: 用户下订单

方法流程: 1. 前端提交 HTTP POST 请求 **CustomerOrderController 的 addOrder**方法

方法签名: public boolean addOrder(Orders order, CartItem cartItem)

返回: true 成功 , false 失败

方法请求映射路径: api/order/addorder

参数: order 订单对象

cartItem 购物车项

控制器流程: **addOrder**方法调用**CustomerOrderService 中的 addOrder** 业务方法,下单成功则返回:true,失败则返回false

2. 业务逻辑 **CustomerOrderService** 依赖 **CustomerOrderMapper** ,并定义启用事务的**addOrder** 方法

方法签名: public boolean addOrder(Orders order, CartItem cartItem)

方法中的业务流程:

1. 根据传来的order 对象 插入订单表中生成新的订单调用**CustomerOrderMapper**
2. 根据传来的cartItem 对象,新生成的订单id组合生成新的订单项 调用**CustomerOrderMapper**

3. 基于MyBaits 在 **CustomerOrderMapper** 中实现基于注解的数据操作

方法签名: public void addOrder(Orders order)

方法描述:插入订单表

方法签名:public int findOid(int uid)

方法描述:查询客户最新生成的订单id

返回:订单id

方法签名:public void addItem(int oid, CartItem cartItem)

方法描述:插入新的订单项

4. 使用表: 订单表 orders

(1) 订单表中字段

uid 用户id

name 收件人姓名

phone 收件人电话

address 收货地址

time 下单时间

payway 支付方式

total 订单总价

订单项表 item

(2) 订单项表中所有字段

#### findItem

方法描述: 查询订单

方法流程: 1. 前端提交 HTTP POST 请求 **CustomerOrderController 的 findItem**方法

方法签名: public PageInfo<Item>(int uid)

返回: item集合

方法请求映射路径:api/order/ finditem

参数:uid 客户id

控制器流程: **findItem**方法调用**CustomerOrderService 中的 findItem** 业务方法,返回订单项分页集合

2. 业务逻辑: **CustomerOrderService** 依赖 **CustomerOrderMappe,**并定义**findItem**方法

方法签名: public PageInfo<Item> findOrders(int uid)

方法中的业务流程:

1. 根据客户的客户id查询其下的所有订单,调用**CustomerOrderMappe**,并返回PageInfo<Item>

3. 基于MyBaits 在 **CustomerOrderMapper** 中实现基于注解的数据操作

方法签名:public list<Item> findItem(int uid)

方法描述:查询客户的订单项 ,且hidden字段为0的订单项

4. 使用表:

1) 订单项表中所有字段

#### searchItem

方法描述: 搜索指定条件订单

方法流程: 1. 前端提交 HTTP POST 请求 **CustomerOrderController 的 searchItem**方法

方法签名: public PageInfo<Item> serarchItem(String conditions,int uid)

返回:PageInfo分页数据

方法请求映射路径:api/order/searchitem

参数:condition 条件 uid 客户id

控制器流程: searchItem方法调用**CustomerOrderService** 中的 searchItem业务方法,并返回PageInfo<Item>分页数据

2. 业务逻辑:**CustomerOrderService** 依赖 **CustomerOrderMapper** ,并定义的searchItem方法

方法中的业务流程:

1. 根据传过来的条件conditions 和客户的uid去**调用CustomerOrderMappe**r中的searchItem方法,取到相应的数据

3．基于MyBaits 在 **CustomerOrderMapper** 中实现基于注解的数据操作

方法签名:public list<item> searchItem(String conditions,int uid)

方法描述:根据条件搜索相关的订单,使用模糊查询等

返回:订单项集合

4. 使用的表: item 订单项表

使用的字段:订单项所有字段

使用表: 订单表 和 订单项表

参数: conditions 条件 , uid 用户id

返回值: Orders集合

异常: 无

#### dltOrder

方法签名: public boolean dltOrder(int oid)

方法描述: 根据订单id进行修改操作

方法流程: 1. 前端提交 HTTP POST 请求 **CustomerOrderController 的 dltOrder**方法

方法签名: public boolean dltOrder(int oid)

返回:true 成功 flase 失败

参数: oid 订单id

控制器流程: **dltOrder**方法调用**CustomerOrderService 中的 dltOrder** 业务方法,下单成功则返回:true,成功则重新发送请求失败则返回false

2. **业务逻辑: CustomerOrderService 依赖 CustomerOrderMappe,并定义dltOrder方法**

方法签名: public boolean dltOrder (List<Integer> oids)

方法中的业务流程:

1. 根据传来的订单id,去**调用CustomerOrderMapper**中的upOrder方法

3. 基于MyBaits 在 **CustomerOrderMapper** 中实现基于注解的数据操作

方法签名: public boolean upOrder(int oid)

方法描述: 根据订单id去修改订单表中的hidden字段成功成功就修改成1

返回: true 成功 flase 失败

4. 使用表:Oders表

使用字段: hidden 是否删除状态

Id 订单id

#### addComment

方法描述: 用户对已到货的商品进行评论

方法流程: 1. 前端提交 HTTP POST 请求 **CustomerOrderController 的 addComment**方法

方法签名: public void addComment(Comment comment)

返回:无

方法请求映射路径: api/order/addcomment

参数: comment 评论实体

控制器流程: **addComment**方法调用**CustomerOrderService 中的 addComment** 业务方法,去实现新一条评论记录

2.业务逻辑**CustomerOrderService** 依赖 **CustomerOrderMapper** ,并定义启用事务的**addComment** 方法

方法签名:public void addComment(Comment comment)

方法中的业务流程:

1. 根据传来的评论实体comment为基本插入到数据库的评论表中**调用CustomerOrderMapper**中addComment

3.基于MyBaits 在 **CustomerOrderMapper** 中实现基于注解的数据操作

方法签名:public void addComment(Comment comment)

方法描述:插入新的用户评论

4.使用表: comment

使用字段: Commodity\_id 商品 id

uid 客户id

content 评论内容

score 评分

### AdminOrder

#### findAllItems

方法描述: 查询订单

方法流程: 1. 前端提交 HTTP POST 请求 **AdminOrderController的 findAllItems**方法

方法签名: public PageInfo<Item>()

返回: item集合

方法请求映射路径:api/order/ findallitems

参数:无

控制器流程: **findOrders**方法调用**AdminOrderService 中的 findAllItems** 业务方法,返回订单项分页集合

2. 业务逻辑: **CustomerOrderService** 依赖**AdminOrderMappe,**并定义**findAllItems**方法

方法签名: public PageInfo<Item> findOrders(int uid)

方法中的业务流程:

1. 根据客户的客户id查询其下的所有订单,**调用AdminOrderMappe**,并返回PageInfo<Item>

3. 基于MyBaits 在 **AdminOrderMappe**中实现基于注解的数据操作

方法签名:public list<Item> findAllItems(int uid)

方法描述:查询客户的订单项 ,且hidden字段为0的订单项

4. 使用表:

1) 订单项表中所有字段

## 管理员模块

### 获取商品列表

1.前端提交HTTP GET 请求 请求到 commodityManngeController 的 loadPage 方法

方法签名: public PageInfo<Commodity> loadPage(int page)

该方法返回JSON格式数据

方法请求映射路径：api/admin/commoditys/show/{page}

参数：

page 页数 int

方法流程：loadPage方法中调用 CommodityMangeService 中的 getCommoditys 业务方法。

2.业务逻辑CommodityManngeService 依赖 CommodityMangeMapper

方法签名：public PageInfo<Commodity> getCommoditys(int page)

方法中的业务流程：

1） 获取全部的商品列表【调用CommodityMangeMapper】

2） 根据传入的参数page 使用PageHelper 架包 获取PageInfo<Commodity> 对象

3） 返回PageInfo<Commodtiy> 对象

3.基于 MyBatis 在 CommodityMangeMapper 中实现基于注解的数据操作

CommodityMangeMapper

方法签名：public List<Commodity> loadCommoditys()

描述：获取全部商品

参数：无

返回值：List<Commodity> 所有商品集合

异常：无

表（视图）名：Commodity

字段：

条件：无

SQL：select \* from Commodity

### 添加商品

1.前端提交HTTP POST 请求 请求到 commodityManngeController 的 addCommodity 方法

方法签名: public Map<String,Obejct> addCommodity (Commodity com)

该方法返回JSON格式数据

方法请求映射路径：api/admin/commodity/add

参数：

com 商品对象 Commodity

方法流程：addCommodity方法中调用 CommodityMangeService 中的 createCommodity 业务方法。

2.业务逻辑CommodityMangeService 依赖 CommodityMangeMapper

方法签名：public void createCommodity(Commodity c) throws SQLException

业务流程：

1）将参数c插入到数据库【调用CommodityMangeMapper】

3.基于 MyBatis 在 CommodityMangeMapper 中实现基于注解的数据操作

CommodityMangeMapper

方法签名：public void createCommodity(Commodity c);

添加一个商品，存入数据库。

参数： c 商品对象

放回值：无

异常：无

表（视图）名：Commodity

字段：title, description, price, category\_id, brand\_id

条件：无

SQL：insert into Commodity(title, description, price, category\_id,brand\_id) values(#{title},#{description},#{price},#{category.id},#{brand.id})

### 商品上下架

1.前端提交HTTP POST 请求 请求到 commodityManngeController 的 commodityState 方法

方法签名: public Map<String, Obejct> commodityState (int id,int option)

该方法返回JSON格式数据

方法请求映射路径：api/admin/commodity/state

参数：

id 商品id int

option 商品上下架状态 int

方法流程：commodityState方法中调用 CommodityMangeService 中的 upDown 业务方法。

2.业务逻辑CommodityManngeService 依赖 CommodityMangeMapper

方法签名：public void upDown(int id,int type) throws SQLException

方法中的业务流程：

1）根据参数id修改指定商品上下架状态(type 2下架 1上架)【调用CommodityMangeMapper】

3.基于 MyBatis 在 CommodityMangeMapper 中实现基于注解的数据操作

CommodityMangeMapper

方法签名：public void upDown(int id,int type);

描述：修改商品上架状态

参数：id 商品id

type 上架(1) 下架(2)

返回值：无

异常：无

表（视图）名：Commodity

字段：shelf

条件：商品id 等于 参数 id

SQL：update commodity set shelf = #{type} where id = #{id}

### 到货通知

1.业务逻辑CommodityManngeService 依赖 CommodityMangeMapper

方法签名：public void notice()

方法中的业务流程：

1）监听数据库表Commodity stock 字段变动

2）查询数据库中notice表获取需要通知用户的邮箱【调用CommodityMangeMapper】

3）通过spring内置邮件服务 给客户发送邮件

2.基于 MyBatis 在 CommodityMangeMapper 中实现基于注解的数据操作

CommodityMangeMapper

方法签名：public List<String> loadEmail();

描述：查询订阅到货通知客户的邮箱

参数：无

返回值：邮箱集合

异常：无

表（视图）名：notice

字段：无

条件：iNotice 字段 等于0

SQL：select email from notice where iNotice = 0

### 库存管理

1.前端提交HTTP POST 请求 请求到 commodityManngeController 的 stockMange 方法

方法签名: public Map<String, Obejct> stockMange (int id,int num,int option)

该方法返回JSON格式数据

方法请求映射路径：api/admin/commodity/stock

参数：

id 商品id int

num 商品库存数量 int

option 操作类型 int

方法流程：

stockMange方法中调用 CommodityMangeService 中的updateStock\addAndDelStock业务方法。

2.业务逻辑CommodityManngeService 依赖 CommodityMangeMapper

方法签名：public void updateStock(int id, int num)

方法中的业务流程：

1）根据参数id修改指定商品的库存数量（num）【调用CommodityMangeMapper】

方法签名：public void addAndDelStock (int id, int num) throws SQLException

方法中的业务流程：

1）根据参数id添加或减少指定商品的库存数量【调用CommodityMangeMapper】

3.基于 MyBatis 在 CommodityMangeMapper 中实现基于注解的数据操作

CommodityMangeMapper

方法签名：public void updateStock(int id, int num);

描述：修改指定商品的库存数量

参数：id 商品id

num 数量

返回：无

异常：无

表（视图）名：Commodity

字段：stock

条件：商品id 等于 参数 id

SQL：update commodity set stock = #{num} where id = #{id}

方法签名：public boolean addAndlessStock(int id,int num);

增加或减少指定商品的库存

参数：id 商品id

num 正数添加 负数减少

返回：数据库操作结果

异常：无

表（视图）名：Commodity

字段：stock

条件：商品id 等于 参数 id

SQL：update commodity set stock = stock + #{num} where id = #{id}

### 推荐商品

1.前端提交HTTP POST 请求 请求到 commodityManngeController 的 recommend 方法

方法签名: public Map<String, Obejct> recommendMange (int id,int code)

该方法返回JSON格式数据

方法请求映射路径：api/admin/commodity/recommend

参数：

id 商品id int

code 推荐类型 int 0 不推荐 1 商家推荐 2 热卖商品

方法流程：recommend方法中调用 CommodityMangeService 中的 recommend 业务方法。

2.业务逻辑CommodityManngeService 依赖 CommodityMangeMapper

方法签名：public void recommend(int id, int code) throws SQLException

方法中的业务流程：

1）根据参数修改指定商品的推荐状态【调用CommodityMangeMapper】

3.基于 MyBatis 在 CommodityMangeMapper 中实现基于注解的数据操作

CommodityMangeMapper

方法签名：public void updateRec(int id, int type);

修改商品的推荐类型

参数：id 商品id

type 推荐类型 0 不推荐 1 商家推荐 2热卖推荐

返回值：无

异常：无

表（视图）名：Commodity

字段：recommend

条件：商品id 等于 参数 id

SQL：update commodity set recommend = #{code} where id = #{id}

### 商品活动

1.前端提交HTTP POST 请求 请求到 commodityManngeController 的 recommend 方法

方法签名: public Map<String, Obejct> activityMange (int id,int type,Activity act)

该方法返回JSON格式数据

方法请求映射路径：api/admin/commodity/activity

参数：

id 商品id int

type 活动类型 int

act 活动参数对象 Activity

方法流程：activityMange方法中调用 CommodityMangeService 中的activity 业务方法。

2.业务逻辑CommodityManngeService 依赖 CommodityMangeMapper

方法签名：public void activity(int id, int type,Activity act) throws SQLException

方法中的业务流程：

1）根据传入的type判断活动类型

2）根据参数往指定活动类型表中插入活动数据【调用CommodityMangeMapper】

3.基于 MyBatis 在 CommodityMangeMapper 中实现基于注解的数据操作

CommodityMangeMapper

方法签名：public void createSpikeActivity(int id,Activity act);

为一个商品创建一个秒杀活动

参数：id 商品id，

act 活动对象

返回值：无

异常：无

表（视图）名：spike

字段：commodity\_id, start\_time, end\_time, stock, price

条件：无

SQL：insert into spike(commodity\_id, start\_time, end\_time, stock, price) values(#{id},#{ startTime },#{ endTime},#{stock},#{price});

方法描述：public void createDiscountActivity(int id,Activity act);

为一个商品创建一个秒杀活动

参数：id 商品id，

act 活动对象

返回值：无

异常：无

表（视图）名：discount

字段：commodity\_id, start\_time, end\_time, price

条件：无

SQL：insert into discount (commodity\_id, start\_time, end\_time, stock, price) values(#{id},#{ startTime },#{ endTime},#{price});

## 统计模块

### 根据时间段统计收入信息

1. 前端或客户端提交GET请求到StatisticsController的timeStatistics方法：

方法签名：public Map<String,Object> timeStatistics(Date startTime,Date endTime);

该方法返回JSON格式数据；

方法请求映射路径（URL）：/api/admin/statistics/revence

参数：

startTime 起始时间

endTime 结束时间

方法流程：timeStatistics方法调用StatisticsService中的revenceStatistics方法，返回方法结果。

1. 业务逻辑StatisticsService依赖StatisticsMapper类

方法签名：public Map<String,Object> revenceStatistics(Date startTime,Date endTime);

方法中的业务流程：调用StatisticsMapper中的searchRevence的方法。并返回统计结果。

3.基于MyBatis在StatisticMapper中实现基于注解的数据操作（orders）

方法签名：public Map<String,Object> searchRevence(Date start,Date end);

方法描述：按时间段统计收入

参数：

start 开始时间

end 结束时间

异常：无

返回值：Map<String,Object> 所有订单的total之和

表名：orders

字段：total

条件：orders的paystatus等于1，orders 的datetime大于start小于end

相关参数：start 开始时间，end结束时间

### 获取某个商品的购买人数

1.前端或客户端提交GET请求到StatisticsController的commodityCount方法：

方法签名：public Mapper<String,Object> commodityCount(int id);

该方法返回JSON格式数据；

方法请求映射路径（URL）：/api/statistics/commodity\_count

参数：

id:商品id

方法流程：

1. commodityCount方法调用Statistics的getCommodityCount方法
2. 向map中put查询到的数据，code为200。
3. 捕获NoCommodityException异常，向map中put数据code为500。

2.业务逻辑StatisticsService依赖StatisticsMapper类和CommodityMpper类

方法签名：public int getCommodityCount (int id) throws NoCommodityException;

方法中的业务流程：

1. 调用CommodityMpper类的selectCommodity，若无商品返回则抛出NoCommodityException异常。
2. 调用StatisticsMapper中的searchCommodityCount的方法。并返回统计结果。

异常：NoCommodityException无此商品

3.基于MyBatis在StatisticMapper中实现基于注解的数据操作（orders，item）

方法签名：public int searchCommodityCount (int id);

方法描述：统计某个商品的购买人数

参数：

id：商品id

异常：无

返回值：int 返回这个商品的所有订单数量

表名：orders ，item

字段：quantity

条件：orders中paystatus为1且item中commodity\_id的值为id之和。

相关参数：id商品id

### 统计某个品牌或类别的所有商品销

1.前端或客户端提交GET请求到StatisticsController的salesStatistic方法：

方法签名：public Map<String,Object> salesStatistic(String type,int id);

该方法返回JSON格式数据；

方法请求映射路径（URL）：/api/admin/statistics/commodity\_sales

参数：

type:品牌或类别（category、或brand）

id:品牌或类别的id

方法流程：

1. salesStatistic方法调用StatisticsService中的salesStatistics方法，返回方法结果。
2. 捕获salesStatistics方法的异常，若发生异常则返回code 500。
3. 返回方法返回的数据。

2.业务逻辑StatisticsService依赖StatisticsMapper类

方法签名：public Map<String,Object> salesStatistics(String type,int id);

方法中的业务流程：

1. 去除type字符串中的空格
2. 判断type是否是category或brand
3. 如果是则调用StatisticsMapper的searchSales方法来统计该类别或品牌的信息。否则抛出NoTypeException。
4. 返回销量统计的信息。调用StatisticsMapper中的searchSales的方法。并返回统计结果。

3.基于MyBatis在StatisticMapper中实现基于注解的数据操作（orders，item，commodity）

方法签名：public Map<String,Object> searchSales(String type,int id);

方法描述：统计某个品牌或类别的所有商品销量

参数：

type: 类型 （category、brand）

id:商品或类别的id

异常：无

返回值：Map<String,Object> 统计某个品牌或类别的商品销量

表名：orders，item,commodity

字段：sum(quantity),title

条件：orders表中的paystatus为1且sendstatus为2，如果type为category,category\_id为参数id，如果type为brand，brand\_id为参数id，以商品id分组查询quantity之和与title

相关参数：type ，id