# 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});