# Shield模块

## 概述

### 承担的需求（主要秘密）

商品屏蔽模块的职责是屏蔽与搜索关键字匹配的商品列表中的负面商品，因为负面商品可以灵活的添加和修改，且屏蔽规则必须符合法律法规，故要求屏蔽规则具有高度的可修改性，可扩展性和灵活性。

**质量属性要求**：商品搜索要求具有很高的可修改性、可扩展性，而变更的部分主要是待屏蔽商品信息的变化。因此该模块的设计就要着重考虑词库的可扩展性和可修改性，可以灵活变更又不影响其他模块的调用。同时，由于搜索返回结果往往比较庞大，待屏蔽商品库也较庞大，因此商品过滤效率也直接影响到用户体验，需要尽量采用高效的过滤算法。用户总共进行商品查询的时间不能多于2s。

**商品屏蔽用例图**



**用例描述**

1. 负面商品信息增删改

系统管理员可以在后台管理界面灵活地增加、修改、删除负面商品。

1. 进行链接过滤

当一个商品被加入负面商品信息表中后，网页上指向该商品的连接必须失效，可以通过url内容比对结合访问控制来实现。

1. 进行内容过滤

当用户输入搜索关键字后，系统将检测搜索到的商品列表，并屏蔽其中的负面商品。

可能会修改的实现（次要秘密）

负面商品信息表

商品过滤规则

**角色**

**向外提供接口**

|  |  |
| --- | --- |
| **接口名** | **职责** |
| Public GoodsList search(Keywords keywords) | 根据输入的关键词，先调用底层模块找到符合关键词的商品列表，再进行过滤，返回符合规则的商品列表。 |
| Public URL filter(URL goodsURL) | 根据输入的URL判断是否指向被屏蔽的商品，若被屏蔽，则重定向到另一个URL，否则返回原URL。 |
| Public void addShieldGoods(Goods goods) | 增加一条负面商品信息 |
| Public void updateShieldGoods(Goods goods) | 修改一条负面商品信息 |
| Public void deleteShieldGoods(Goods goods) | 删除一条负面商品信息 |

## 类图

## 类描述

### XX类

|  |  |
| --- | --- |
| 方法名 |  |
| 前置条件 |  |
| 后置条件 |  |
| 作用 |  |

## 重要协作

## 设计模式

# PriceParity模块

## 概述

### 承担的需求（主要秘密）

商品比价模块的职责是对同一个商品，比较其在不同购物网站上的价格，并可以按照价格进行升序和降序排序。要求可以灵活地定义参与比价的网站，在未来可能会增加不同的价格计算规则，因此比价模块应该具有高度的可修改性，可扩展性和灵活性。

**商品比价用例图**



**用例描述**

1. 展示不同购物网站上的商品价格

用户查看某一商品详细信息时，系统应显示该商品在不同网站上的价格，且用户可以自定义参与比价的购物网站（从本系统提供的购物网站中选择）。

1. 根据价格排序

用户在查看某一商品时，可以点击“按价格排序”，系统将不同购物网站按商品价格升序或降序排序。

可能会修改的实现（次要秘密）

参与比价的购物网站。

价格计算规则，如是否包括运费，是否可以使用优惠券等等。

**角色**

**向外提供接口**

|  |  |
| --- | --- |
| **接口名** | **职责** |
| Public GoodsList findNamesake(Goods goods) | 根据该商品查找不同购物网站上的同名商品，返回同名商品列表 |
| Public GoodsPriceList sort(GoodsPriceList goodsPriceList,Order order) | 根据Order为升序还是降序，将参数中的goodsList按价格排序 |
| Public void addShoppingSite (ShoppingSite site) | 对当前用户，增加一个参与比价的比价网站 |
| Public void deleteShoppingSite (ShoppingSite site) | 对当前用户，删除一个参与比价的比价网站 |

## 类图



## 类描述

### PriceParity接口

|  |  |
| --- | --- |
| 方法名 | Public GoodsList findNamesake(Goods goods) |
| 前置条件 | goods不为null,goods已经存在于数据库中，用户已经登陆 |
| 后置条件 | GoodsPriceList不为null |
| 作用 | 通过goods名字，在当前用户默认的购物网站中找到同名商品列表 |

|  |  |
| --- | --- |
| 方法名 | Public GoodsPriceList sort(GoodsList goodsList, Order order，PriceCalcuStrategy strategy) |
| 前置条件 | goodsList, order, strategy不为null, order的排序依据不为null，用户已经登陆 |
| 后置条件 | 返回的goodsPriceList已经按order的排序依据(asc或desc)排好序 |
| 作用 | 根据order的排序依据对goodsPriceList中的项排序 |

|  |  |
| --- | --- |
| 方法名 | Public void addShoppingSite (ShoppingSite site) |
| 前置条件 | Site不为null，site存在于数据库中，用户已经登陆 |
| 后置条件 | 系统将site添加到用户默认购物网站列表中 |
| 作用 | 给用户比价时增加一个参与比价的购物网站 |

|  |  |
| --- | --- |
| 方法名 | Public void deleteShoppingSite (ShoppingSite site) |
| 前置条件 | Site不为null，site存在于数据库中，用户已经登陆 |
| 后置条件 | 系统将site从用户默认购物网站列表中删除 |
| 作用 | 给用户比价时减少一个参与比价的购物网站 |

### PriceParityImpl类

|  |  |
| --- | --- |
| 方法名 | Public GoodsList findNamesake(Goods goods) |
| 前置条件 | goods不为null,goods已经存在于数据库中，用户已经登陆 |
| 后置条件 | GoodsPriceList不为null |
| 作用 | 通过goods名字，在当前用户默认的购物网站中找到同名商品列表 |

|  |  |
| --- | --- |
| 方法名 | Public GoodsPriceList sort(GoodsList goodsList, Order order，PriceCalcuStrategy strategy) |
| 前置条件 | goodsList, order, strategy不为null, order的排序依据不为null，用户已经登陆 |
| 后置条件 | 返回的goodsPriceList已经按order的排序依据(asc或desc)排好序 |
| 作用 | 根据order的排序依据对goodsPriceList中的项排序 |

|  |  |
| --- | --- |
| 方法名 | Public void addShoppingSite (ShoppingSite site) |
| 前置条件 | Site不为null，site存在于数据库中，用户已经登陆 |
| 后置条件 | 系统将site添加到用户默认购物网站列表中 |
| 作用 | 给用户比价时增加一个参与比价的购物网站 |

|  |  |
| --- | --- |
| 方法名 | Public void deleteShoppingSite (ShoppingSite site) |
| 前置条件 | Site不为null，site存在于数据库中，用户已经登陆 |
| 后置条件 | 系统将site从用户默认购物网站列表中删除 |
| 作用 | 给用户比价时减少一个参与比价的购物网站 |

### NamesakeFinder类

|  |  |
| --- | --- |
| 方法名 | Public GoodsList findNamesake(Goods goods,UserShoppingSiteList siteList) |
| 前置条件 | goods不为null，siteList不为null |
| 后置条件 | goodsPriceList中包含在siteList项对应的购物网站中与goods同名的商品，也包括同义词表中包含的商品 |
| 作用 | 给用户比价时减少一个参与比价的购物网站 |

### PriceSort类

|  |  |
| --- | --- |
| 方法名 | Public GoodsPriceList sort(GoodsPriceList goodsPriceList,Order order) |
| 前置条件 | goodsPriceList,order不为null,order的排序依据不为null |
| 后置条件 | 返回的goodsPriceList已经按order的排序依据排好序 |
| 作用 | 根据order的排序依据和顺序对goodsPriceList中的项排序 |

### PriceCalcuStrategy类

|  |  |
| --- | --- |
| 方法名 | Public GoodsPrice calcuPrice(Goods goods) |
| 前置条件 | goods不为null |
| 后置条件 | GoodsPrice不为null |
| 作用 | 计算商品价格，不同的PriceCalcuStrategy子类实现不同的价格计算策略 |

### UserShoppingSiteController类

|  |  |
| --- | --- |
| 方法名 | Public UserShoppingSiteList getShoppingSiteList(User user) |
| 前置条件 | user不为null |
| 后置条件 | UserShoppingSiteList不为null |
| 作用 | 根据用户返回用户设置的参与比价的购物网站的列表 |

|  |  |
| --- | --- |
| 方法名 | Public void addShoppingSite(ShoppingSite site,User user) |
| 前置条件 | user不为null, site 不为null |
| 后置条件 | 系统将site添加到用户默认购物网站列表中 |
| 作用 | 调用持久化模块，保存用户增加的购物网站到用户设置的参与比价的购物网站列表中 |

|  |  |
| --- | --- |
| 方法名 | Public void deleteShoppingSite(ShoppingSite site,User user) |
| 前置条件 | user不为null, site 不为null |
| 后置条件 | 系统将site从用户默认购物网站列表中删除 |
| 作用 | 调用持久化模块，将用户选择的购物网站从用户设置的参与比价的购物网站列表中删除 |

## 重要协作

### 寻找同名商品



### 计算价格



### 排序



### 增删购物网站

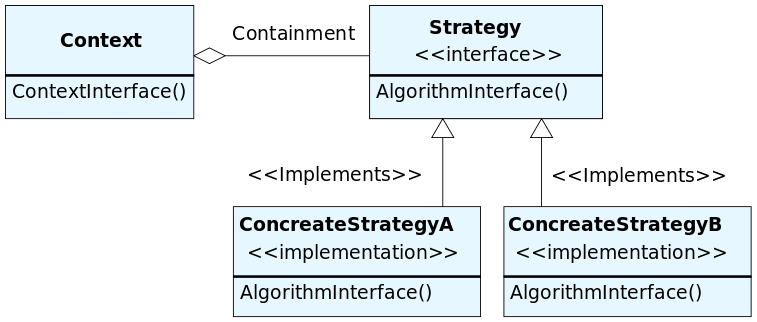


## 设计模式

### 策略模式

策略模式定义了一系列的算法，并将每一个算法封装起来，而且使它们之间可以相互替换。策略模式让算法独立于使用它的客户而独立变化。它提供了一种替代继承的方法，而且既保持了继承的优点(代码重用)，还比继承更灵活(算法独立，可以任意扩展)。同时避免程序中使用多重条件转移语句，使系统更灵活，并易于扩展。策略模式还遵守大部分GRASP 原则和常用设计原则，高内聚、低偶合。

如下图所示：



在价格计算模块中，因为要求考虑到不同计算模式下总价的差异，因此必须将各个价格计算算法封装起来，实现calcuPrice()接口后，就可以方便的修改，如果要新增算法，增加一个算法类，实现上述接口，并在声明的时候将引用指向新的算法类即可；如果要移除算法，移除对应的算法类并更新声明的引用即可；如果要修改算法，可在不修改接口的前提下修改算法类的内部即可，保证了高内聚和低耦合。

类图体现：

