商场促销系统

技术实现文档version 1.0

Arvin Si.Chuan / 邱依强

2017

目录

[1 介绍 1](#_Toc484691046)

[1.1 本文档的用意 1](#_Toc484691047)

[1.2 定义与缩写词 1](#_Toc484691048)

[1.3 参考资料 1](#_Toc484691049)

[1.4 阅读指南 1](#_Toc484691050)

[1.4.1 本文档已涵盖的内容 1](#_Toc484691051)

[1.4.2 本系统将要实现但尚未实现的内容 2](#_Toc484691052)

[2 构建环境 2](#_Toc484691053)

[2.1 构建所采用的软硬件环境 2](#_Toc484691054)

[2.2 本系统实现所采用的外部框架 3](#_Toc484691055)

[3 主要业务流程 5](#_Toc484691056)

[3.1 系统使用主要业务流程 5](#_Toc484691057)

[3.1.1 使用系统的主要过程 5](#_Toc484691058)

[3.1.2 进行促销的业务流程 5](#_Toc484691059)

[3.1.3 后台业务执行流程 5](#_Toc484691060)

[3.1.4 异步传输业务流程 6](#_Toc484691061)

[3.1.5 促销规则匹配的主要业务流程 6](#_Toc484691062)

[4 核心接口 6](#_Toc484691063)

[4.1 REST API 6](#_Toc484691064)

[4.1.1 员工资源 7](#_Toc484691065)

[4.1.2 商品资源 8](#_Toc484691066)

[4.1.3 订单资源 10](#_Toc484691067)

[4.1.4 促销规则资源 13](#_Toc484691068)

[4.2 业务接口类 20](#_Toc484691069)

[4.2.1 外部支持系统 20](#_Toc484691070)

[4.2.2 订单处理接口 21](#_Toc484691071)

[4.2.3 促销规则匹配接口 21](#_Toc484691072)

[4.2.4 支付接口 21](#_Toc484691073)

[5 核心功能算法实现 21](#_Toc484691074)

[5.1 订单处理器 21](#_Toc484691075)

[5.2 规则匹配 24](#_Toc484691076)

[5.2.1 规则匹配器控制器 24](#_Toc484691077)

[5.2.2 基础匹配处理器 26](#_Toc484691078)

[5.2.3 FULLFREE匹配处理器 27](#_Toc484691079)

[5.2.4 FULLCOUNT匹配处理器 29](#_Toc484691080)

[5.2.5 BUYFREE匹配处理器 31](#_Toc484691081)

[6 尚未解决的问题 33](#_Toc484691082)

# 介绍

## 本文档的用意

本文档作为商场促销系统实现部分的说明文档，主要用于在项目的开发过程中，为便于对系统的详细实现和进一步维护、对系统的实现部分进行追踪。为达到上述目标，本文档主要陈述实现系统的构建环境、主要接口、核心算法以及尚未完全解决的问题。

## 定义与缩写词

1. BUYFREE：买X件减Y元优惠类型；
2. BUYCOUNT：买X件打Y折优惠类型；
3. BUYSPECIAL：买X件享受特价优惠类型；
4. BUYPRESENT：买X件赠送Y件商品或优惠券优惠类型；
5. FULLFREE：满X元减Y元优惠类型；
6. FULLCOUNT：满X元打Y折优惠类型；
7. FULLPRESENT：满X元赠送Y件商品或优惠券优惠类型。

## 参考资料

1. AJAX与spring mvc交互（<http://blog.csdn.net/yangtang_newton/article/details/7525800>）
2. Spring security实现权限管理(<http://blog.csdn.net/zmx729618/article/details/51096593>)
3. 最新SpringMVC 4，如何解决@ResponseBody时，String类型乱码(<http://bbs.csdn.net/topics/391855872?page=1>)
4. Spring Security 4.2.2.RELEASE API(<http://docs.spring.io/spring-security/site/docs/current/apidocs/index.html?index-all.html>)

## 阅读指南

### 本文档已涵盖的内容

1. 促销系统对于商品信息、促销规则、订单信息和员工信息的REST风格Web API，其中包含有经过测试的API；
2. 系统对促销进行支持的用户接口以及其视图控制器；
3. 促销系统进行促销的业务流程；
4. 部分核心促销规则的匹配、实现算法。

### 本系统将要实现但尚未实现的内容

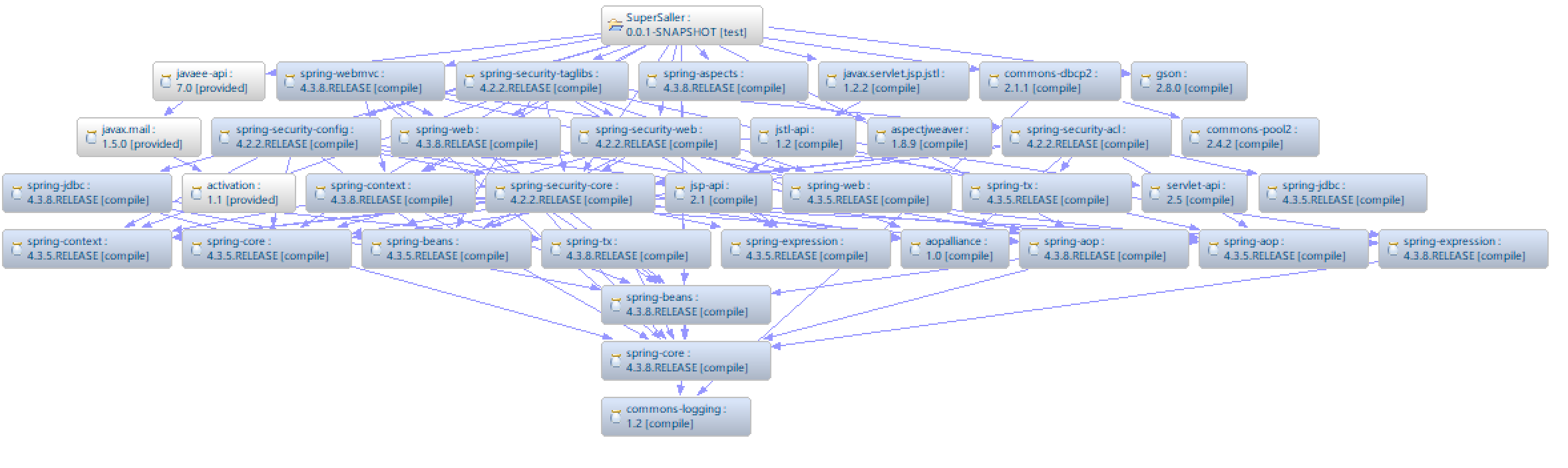
1. 促销规则的维护功能；
2. 促销侧员工的维护；
3. 取消订单的功能；
4. 重新读取非归档态（未结算且未取消）订单的功能；
5. 订单的支付功能；
6. 订单打印的支持。

# 构建环境

## 构建所采用的软硬件环境

|  |  |
| --- | --- |
| **环境项** | **内容** |
| 实现语言 | JAVA |
| JDK Version | jdk1.8.0\_91 |
| JVM Version | jre1.8.0\_91 |
| JAVA EE Version | JAVA EE 7 – Web 3.1 |
| JSTL Version | 1.2.2 |
| 服务器环境 | Apache Tomcat 8.0.39 |
| 数据库 | Oracle 12c 12.1.0 |

## 本系统实现所采用的外部框架

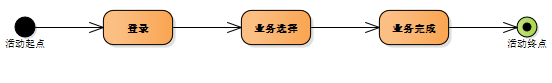


|  |  |  |  |
| --- | --- | --- | --- |
| **框架名** | **版本号** | **从属包全名** | **说明** |
| Commons-dbcp2 | 2.1.1 | org.apache.commons | DBCP连接池V2 |
| Gson | 2.8.0 | com.google.code.gson | Google json |
| Spring-core | 4.3.8-RELEASE | org.springframework | Spring 4框架 |
| Spring-context | 4.3.8-RELEASE | org.springframework |
| Spring-aop | 4.3.8-RELEASE | org.springframework |
| Spring-beans | 4.3.8-RELEASE | org.springframework |
| Spring-aspect | 4.3.8-RELEASE | org.springframework |
| Spring-jdbc | 4.3.8-RELEASE | org.springframework |
| Spring-web | 4.3.8-RELEASE | org.springframework |
| Spring-webmvc | 4.3.8-RELEASE | org.springframework |
| Spring-security-core | 4.2.2-RELEASE | org.springframework.security | Spring Security 4 框架 |
| Spring-security-web | 4.2.2-RELEASE | org.springframework.security |
| Spring-security-config | 4.2.2-RELEASE | org.springframework.security |
| Spring-security-taglibs | 4.2.2-RELEASE | org.springframework.security |
| JQuery | v3.2.1 |  |  |

# 主要业务流程

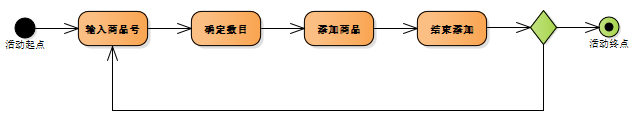
## 系统使用主要业务流程

### 使用系统的主要过程



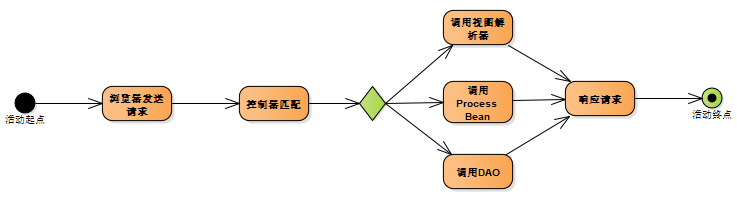
说明：用户使用系统时，应当遵循此过程。任何功能的使用都需要进行登录验证。

### 进行促销的业务流程



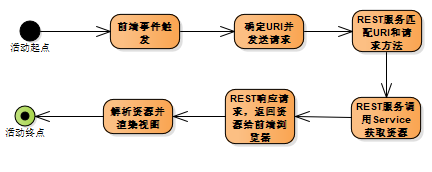
说明：收银员在使用系统时，首先输入商品号，通过商品号索引得出具体商品，再选择数目并添加到已购清单。在此过程中，促销规则的匹配会在后台服务器自动完成，匹配后的商品条目会被打上规则ID的标签。

### 后台业务执行流程



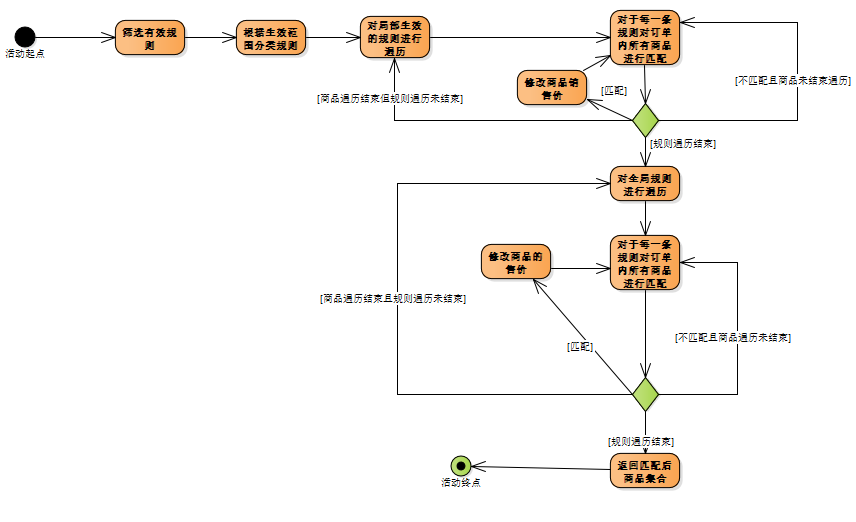
说明：前端浏览器在发送任何一个请求后，后台服务器接收并匹配其对应的资源，控制器通过其请求类型和URI确定具体业务。业务确定后调用Service进行资源的获取并及时响应请求。

### 异步传输业务流程



说明：异步传输返回数据全部采用JSON格式，服务器接收的请求不允许采用GET方法，允许表单类型和JSON类型的参数值传递。为配合Spring Security CSRF,所有AJAX请求都应当在Request Header中放入X-CSRF-TOKEN验证报头。

### 促销规则匹配的主要业务流程



# 核心接口

## REST API

位置：com.superSaller.controller.REST

注解：@RestController

约束：所有接口不允许GET方法，所有接口需要有效的X-CSRF-TOKEN报头

### 员工资源

总览：

|  |  |
| --- | --- |
| **位置** | com.superSaller.controller.REST.AdminRESTController.java |
| **URI** | hostname:8080/SuperSaller/admin/ |
| **Annotation** | @RequesetMapping”/admin”) |

详细内容：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **用途** | 获取指定员工信息 | **版本** | | 1.0 | **状态** | | 测试通过 |
| **URI** | hostname:8080/SuperSaller/admin/em/{emid} | | | | | | |
| **允许的方法** | POST | | **允许的参数类型** | | | 路径变量 | |
| **调用示例** |  | | | | | | |
| **返回值示例** | {  "emRole": "cashier",  "emID": "cashier"  } | | | | | | |
| **控制器**  **方法体** | @RequestMapping(value = "/em/{emid}", method = RequestMethod.***POST***)  **public** CashSideEmployee getCashSideEmployee(@PathVariable(value = "emid") String emid) {  **return** dao.queryEmployeeByID(emid);  } | | | | | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **用途** | 获取所有员工信息 | **版本** | | 1.0 | **状态** | | 测试通过 |
| **URI** | hostname:8080/SuperSaller/admin/em/all | | | | | | |
| **允许的方法** | POST | | **允许的参数类型** | | | null | |
| **调用示例** |  | | | | | | |
| **返回值示例** | [  {  "emRole": "quit",  "emID": "\_\_\_\_\_\_\_\_"  },  {  "emRole": "cashier",  "emID": "cashier"  },  {  "emRole": "systemAdmin",  "emID": "jimi"  }  ] | | | | | | |
| **控制器**  **方法体** | @RequestMapping(value = "/em/all", method = RequestMethod.***POST***)  **public** List<CashSideEmployee> getAllEm() {  **return** dao.queryAllEm();  } | | | | | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **用途** | 删除员工 | **版本** | | 1.0 | **状态** | | 测试通过 |
| **URI** | hostname:8080/SuperSaller/admin/em/{emid} | | | | | | |
| **允许的方法** | DELETE | | **允许的参数类型** | | | 路径变量 | |
| **调用示例** |  | | | | | | |
| **返回数据示例** | {  "status": true,  } | | | | | | |
| **控制器**  **方法体** | @RequestMapping(value = "/em/{emid}", method = RequestMethod.***DELETE***)  **public** Map<String, Boolean> deleteEmByID(@PathVariable("emid") String emID) {  Map<String, Boolean> status = **new** HashMap<String, Boolean>();  status.put("status", dao.deleteEm(emID));  **return** status;  } | | | | | | |

### 商品资源

总览：

|  |  |
| --- | --- |
| **位置** | com.superSaller.controller.REST. GoodsRESTController.java |
| **URI** | hostname:8080/SuperSaller/goods/ |
| **Annotation** | @RequesetMapping”/goods”) |

详细内容：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **用途** | 商品号模糊查询 | **版本** | | 1.0 | **状态** | | 测试通过 |
| **URI** | hostname:8080/SuperSaller/goods/fuzzyQuery | | | | | | |
| **允许的方法** | POST | | **允许的参数类型** | | | 表单类型 | |
| **调用参数示例** | {goodID:2} | | | | | | |
| **返回数据示例** | [  {  "goodID": "2119008",  "goodName": "味全活性乳酸菌饮品（芦荟味）",  "goodSpecifications": "330ml",  "goodPrice": 6.5,  "goodType": "饮料",  "goodBrand": "味全",  "goodProducer": "杭州味全食品有限公司"  },  ] | | | | | | |
| **控制器**  **方法体** | @RequestMapping(value = "/fuzzyQuery", method = RequestMethod.***POST***)  **public** List<Good> fuzzyQueryGoods(String goodID) {  **return** goodsSupport.fuzzyQuery(goodID);  } | | | | | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **用途** | 单个商品精确查询 | **版本** | | 1.0 | **状态** | | 测试通过 |
| **URI** | hostname:8080/SuperSaller/goods/query | | | | | | |
| **允许的方法** | POST | | **允许的参数类型** | | | 表单类型 | |
| **调用参数示例** | {"goodID": "2119008"} | | | | | | |
| **返回数据示例** | {  "goodID": "2119008",  "goodName": "味全活性乳酸菌饮品（芦荟味）",  "goodSpecifications": "330ml",  "goodPrice": 6.5,  "goodType": "饮料",  "goodBrand": "味全",  "goodProducer": "杭州味全食品有限公司"  } | | | | | | |
| **控制器**  **方法体** | @RequestMapping(value = "/query", method = RequestMethod.***POST***)  **public** Good queryGood(String goodID) {  Good good = goodsSupport.getGood(goodID);  **return** good;  } | | | | | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **用途** | 模糊查询商品号 | **版本** | | 1.0 | **状态** | | 测试通过 |
| **URI** | hostname:8080/SuperSaller/goods/fuzzyQueryID | | | | | | |
| **允许的方法** | POST | | **允许的参数类型** | | | 表单类型 | |
| **调用参数示例** | {goodID:"211"} | | | | | | |
| **返回数据示例** | [  "2119008",  "2119029"  ] | | | | | | |
| **控制器**  **方法体** | @RequestMapping(value = "/fuzzyQueryID", method = RequestMethod.***POST***)  **public** List<String> fuzzyQueryGoodsID(String goodID) {  List<Good> goods = goodsSupport.fuzzyQuery(goodID);  List<String> possibleID = **new** ArrayList<String>();  **for** (Good good : goods) {  possibleID.add(good.getGoodID());  }  **return** possibleID;  } | | | | | | |

### 订单资源

总览：

|  |  |
| --- | --- |
| **位置** | com.superSaller.controller.REST. OrderRESTController.java |
| **URI** | hostname:8080/SuperSaller/order/ |
| **Annotation** | @RequesetMapping”/order”) |

详细内容：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **用途** | 订单模糊查询 | **版本** | | 1.0 | **状态** | | 测试通过 |
| **URI** | hostname:8080/SuperSaller/order/{orderid} | | | | | | |
| **允许的方法** | POST | | **允许的参数类型** | | | 路径变量 | |
| **调用参数示例** |  | | | | | | |
| **返回数据示例** | [  {  "orderID": "20170608002821000000000000000285",  "orderSum": 0,  "sumMoney": 0,  "employee": {  "emRole": "systemAdmin",  "emID": "jimi"  },  "status": "inited"  }  ] | | | | | | |
| **控制器**  **方法体** | @RequestMapping(value = "/{orderid}", method = RequestMethod.***POST***)  **public** List<Order> fuzzyQueryOrders(@PathVariable("orderid") String id) {  **return** orderDAO.fuzzyQueryOrders(id);  } | | | | | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **用途** | 向指定订单添加商品 | **版本** | | 1.0 | **状态** | | 测试通过 |
| **URI** | hostname:8080/SuperSaller/order/ /good/add/{orderid}/{quantity} | | | | | | |
| **允许的方法** | PUT | | **允许的参数类型** | | | 路径变量+JSON请求体 | |
| **调用参数示例** | Request uri:  <http://localhost:8080/SuperSaller/order/good/add/20170608121226000000000000000330/1>  request body:  {"goodID":"9787508672069","goodName":"未来简史","goodSpecifications":"16开平装","goodPrice":50.9,"goodType":"图书","goodBrand":"中信出版（Citic Press）","goodProducer":"中信出版社"} | | | | | | |
| **返回数据示例** | [{"saledGood":{"goodID":"10138758607","saledDate":{"date":{"year":2017,"month":6,"day":8},"time":{"hour":12,"minute":17,"second":10,"nano":0}},"sum":1.0,"price":58.0,"orderID":"20170608121226000000000000000330","RuleID":[]},"goodID":"10138758607","goodName":"52度泸州老窖醉美泸州酒","goodSpecifications":"50ml","goodPrice":58.0,"goodType":"白酒","goodBrand":"泸州老窖系列酒","goodProducer":"泸州老窖股份有限公司"}] | | | | | | |
| **控制器**  **方法体** | @RequestMapping(value = "/good/add/{orderid}/{quantity}", method = RequestMethod.***PUT***)  **public** List<ViewSideGood> addGoodToOrder(@RequestBody ViewSideGood good, @PathVariable("orderid") String orderID,  @PathVariable("quantity") **double** quantity) {  good.setSaledDate(LocalDateTime.*now*());  good.setSaledPrice(good.getGoodPrice());  good.setOrderID(orderID);  good.setSum(quantity);  List<ViewSideGood> viewSideGoods = orderProcess.addGoodAndMatch(good);  **return** viewSideGoods;  } | | | | | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **用途** | 从指定订单中删除商品 | **版本** | | 1.0 | **状态** | | 测试通过 |
| **URI** | hostname:8080/SuperSaller/order/good/delete | | | | | | |
| **允许的方法** | DELETE | | **允许的参数类型** | | | 请求体 | |
| **调用参数示例** | Uri:  http://localhost:8080/SuperSaller/order/good/delete  Request body:  {"saledGood":{"goodID":"9787508672069","saledDate":{"date":{"year":2017,"month":6,"day":8},"time":{"hour":12,"minute":12,"second":41,"nano":0}},"sum":2,"price":50.9,"orderID":"20170608121226000000000000000330","RuleID":[]},"goodID":"9787508672069","goodName":"未来简史","goodSpecifications":"16开平装","goodPrice":50.9,"goodType":"图书","goodBrand":"中信出版（Citic Press）","goodProducer":"中信出版社"} | | | | | | |
| **返回数据示例** | [{"saledGood":{"goodID":"10138758607","saledDate":{"date":{"year":2017,"month":6,"day":8},"time":{"hour":12,"minute":17,"second":10,"nano":0}},"sum":1.0,"price":58.0,"orderID":"20170608121226000000000000000330","RuleID":[]},"goodID":"10138758607","goodName":"52度泸州老窖醉美泸州酒","goodSpecifications":"50ml","goodPrice":58.0,"goodType":"白酒","goodBrand":"泸州老窖系列酒","goodProducer":"泸州老窖股份有限公司"}] | | | | | | |
| **控制器**  **方法体** | @RequestMapping(value = "/good/delete", method = RequestMethod.***DELETE***)  **public** List<ViewSideGood> removeFromOrder(@RequestBody ViewSideGood good) {  **return** orderProcess.removeGoodAndMatch(good);  } | | | | | | |

### 促销规则资源

总览：

|  |  |
| --- | --- |
| **位置** | com.superSaller.controller.REST. RuleRESTController.java |
| **URI** | hostname:8080/SuperSaller/rule/ |
| **Annotation** | @RequesetMapping”/rule”) |

详细内容：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **用途** | 通过UUID索引规则 | **版本** | | 1.0 | **状态** | | 测试通过 |
| **URI** | hostname:8080/SuperSaller/rule/{uuid} | | | | | | |
| **允许的方法** | POST | | **允许的参数类型** | | | 路径变量 | |
| **调用示例** |  | | | | | | |
| **返回数据示例** | {  "UUID": "A99962051CC6492D98BB0C0B42AA172D",  "Type": "FULLFREE",  "Name": "满200减20",  "emID": "jimi",  "discountRate": 100,  "freeMoney": 20,  "specialPrice": -1,  "conditionValue": 200,  "datePeriodEnd": {  "date": {  "year": 2017,  "month": 6,  "day": 30  },  "time": {  "hour": 0,  "minute": 0,  "second": 0,  "nano": 0  }  },  "datePeriodStart": {  "date": {  "year": 2017,  "month": 6,  "day": 6  },  "time": {  "hour": 0,  "minute": 0,  "second": 0,  "nano": 0  }  },  "dayPeriodEnd": 1439,  "dayPeriodStart": 540,  "priority": 0,  "weekPeriodEnd": 7,  "weekPeriodStart": 1  } | | | | | | |
| **控制器**  **方法体** | @RequestMapping(value = "/{uuid}", method = RequestMethod.***POST***)  **public** DiscountRule qureyRuleByID(@PathVariable(value = "uuid") String uuid) {  **return** ruleDAO.queryRuleByID(uuid);  } | | | | | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **用途** | 新增规则 | **版本** | | 1.0 | **状态** | | 测试通过 |
| **URI** | hostname:8080/SuperSaller/rule/new | | | | | | |
| **允许的方法** | POST | | **允许的参数类型** | | | 请求体 | |
| **调用示例** | {  "UUID": "",  "Type": "FULLFREE",  "Name": "满200减20",  "emID": "jimi",  "discountRate": 100,  "freeMoney": 20,  "specialPrice": -1,  "conditionValue": 200,  "datePeriodEnd": {  "date": {  "year": 2017,  "month": 6,  "day": 30  },  "time": {  "hour": 0,  "minute": 0,  "second": 0,  "nano": 0  }  },  "datePeriodStart": {  "date": {  "year": 2017,  "month": 6,  "day": 6  },  "time": {  "hour": 0,  "minute": 0,  "second": 0,  "nano": 0  }  },  "dayPeriodEnd": 1439,  "dayPeriodStart": 540,  "priority": 0,  "weekPeriodEnd": 7,  "weekPeriodStart": 1  } | | | | | | |
| **返回数据示例** | {  "UUID": "A99962051CC6492D98BB0C0B42AA172D",  "Type": "FULLFREE",  "Name": "满200减20",  "emID": "jimi",  "discountRate": 100,  "freeMoney": 20,  "specialPrice": -1,  "conditionValue": 200,  "datePeriodEnd": {  "date": {  "year": 2017,  "month": 6,  "day": 30  },  "time": {  "hour": 0,  "minute": 0,  "second": 0,  "nano": 0  }  },  "datePeriodStart": {  "date": {  "year": 2017,  "month": 6,  "day": 6  },  "time": {  "hour": 0,  "minute": 0,  "second": 0,  "nano": 0  }  },  "dayPeriodEnd": 1439,  "dayPeriodStart": 540,  "priority": 0,  "weekPeriodEnd": 7,  "weekPeriodStart": 1  } | | | | | | |
| **控制器**  **方法体** | @RequestMapping(value = "/new", method = RequestMethod.***POST***)  **public** DiscountRule addRule(@RequestBody DiscountRule rule) {  String uuid = UUID.*randomUUID*().toString().replaceAll("-", "").toUpperCase();  rule.setUUID(uuid);  **return** ruleDAO.addRule(rule);  } | | | | | | |

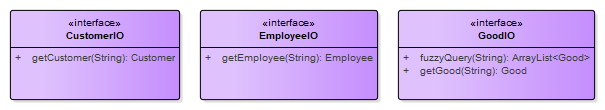
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **用途** | 向规则添加绑定的商品 | **版本** | | 1.0 | **状态** | | 测试通过 |
| **URI** | hostname:8080/SuperSaller/rule/{uuid}/good/ | | | | | | |
| **允许的方法** | PUT | | **允许的参数类型** | | | 路径变量+表单数据 | |
| **调用示例** | URI：  http://localhost:8080/SuperSaller/rule/A99962051CC6492D98BB0C0B42AA172D/good/  Form data:  {  “goodID”: “9787508672069”,  Sum:1  } | | | | | | |
| **返回数据示例** | {  “status”:true, } | | | | | | |
| **控制器**  **方法体** | @RequestMapping(value = "/{uuid}/good/", method = RequestMethod.***PUT***)  **public** Map<String, Boolean> addGoodToRule(@PathVariable("uuid") String ruleUUID, String goodID, **double** sum) {  **boolean** flag = discountGoodsDAO.addGoodToRule(ruleUUID, goodID, sum);  Map<String, Boolean> status = **new** HashMap<String, Boolean>();  status.put("status", flag);  **return** status;  } | | | | | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **用途** | 显示指定员工创建的所有规则 | **版本** | | 1.0 | **状态** | | 测试通过 |
| **URI** | hostname:8080/SuperSaller/rule/{emid}/all | | | | | | |
| **允许的方法** | POST | | **允许的参数类型** | | | 路径变量 | |
| **调用示例** |  | | | | | | |
| **返回数据示例** | [  {  "UUID": "A99962051CC6492D98BB0C0B42AA172D",  "Type": "FULLFREE",3  "Name": "满200减20",  "emID": "jimi",  "discountRate": 100,  "freeMoney": 20,  "specialPrice": -1,  "conditionValue": 200,  "datePeriodEnd": {  "date": {  "year": 2017,  "month": 6,  "day": 30  },  "time": {  "hour": 0,  "minute": 0,  "second": 0,  "nano": 0  }  },  "datePeriodStart": {  "date": {  "year": 2017,  "month": 6,  "day": 6  },  "time": {  "hour": 0,  "minute": 0,  "second": 0,  "nano": 0  }  }  ] | | | | | | |
| **控制器**  **方法体** | @RequestMapping(value = "/{emid}/all", method = RequestMethod.***POST***)  **public** List<DiscountRule> queryAllRulesByEm(@PathVariable("emid") String emID) {  System.***out***.println(emID);  **return** ruleDAO.queryRulesByEm(emID);  } | | | | | | |

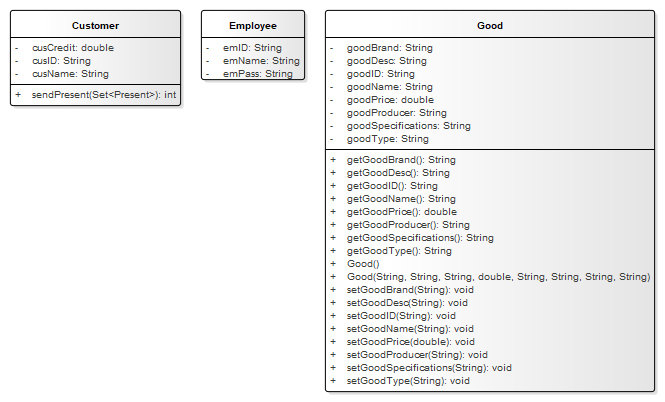
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **用途** | 获取所有规则 | **版本** | | 1.0 | **状态** | | 测试通过 |
| **URI** | hostname:8080/SuperSaller/rule/all | | | | | | |
| **允许的方法** | POST | | **允许的参数类型** | | | null | |
| **调用示例** |  | | | | | | |
| **返回数据示例** | [  {  "UUID": "A99962051CC6492D98BB0C0B42AA172D",  "Type": "FULLFREE",3  "Name": "满200减20",  "emID": "jimi",  "discountRate": 100,  "freeMoney": 20,  "specialPrice": -1,  "conditionValue": 200,  "datePeriodEnd": {  "date": {  "year": 2017,  "month": 6,  "day": 30  },  "time": {  "hour": 0,  "minute": 0,  "second": 0,  "nano": 0  }  },  "datePeriodStart": {  "date": {  "year": 2017,  "month": 6,  "day": 6  },  "time": {  "hour": 0,  "minute": 0,  "second": 0,  "nano": 0  }  }  ] | | | | | | |
| **控制器**  **方法体** | @RequestMapping(value = "/all", method = RequestMethod.***POST***)  **public** List<DiscountRule> queryAllRules() {  **return** ruleDAO.queryAllRules();  } | | | | | | |

## 业务接口类

### 外部支持系统

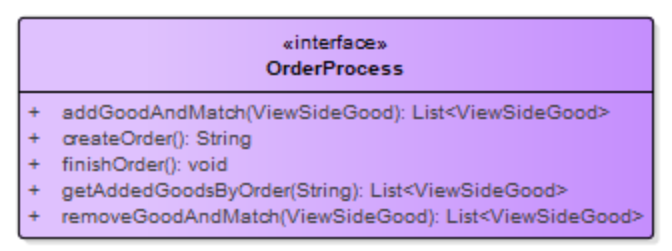


本系统正常运行所需要的外部支持系统包含有三类，主要负责非本系统边界内的资源的读写，其中最为常用的接口应当为GoodIO，应该尽可能提高其并发性能。



外部支持系统实现时，需要根据上图所示的实体类进行相关实体的构建或转换以保持与本系统的相容性。

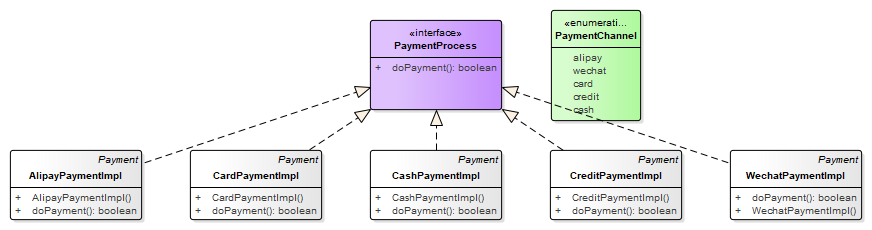
### 订单处理接口



### 促销规则匹配接口



### 支付接口



# 核心功能算法实现

## 订单处理器

|  |
| --- |
| **package** com.superSaller.beans.checkout.Impl;  **import** java.util.ArrayList;  **import** java.util.List;  **import** javax.annotation.Resource;  **import** org.springframework.stereotype.Service;  **import** com.superSaller.beans.checkout.DiscountMatcher;  **import** com.superSaller.beans.checkout.OrderProcess;  **import** com.superSaller.beans.checkout.entities.ViewSideGood;  **import** com.superSaller.beans.payment.PaymentProcess;  **import** com.superSaller.dao.OrderDAO;  **import** com.superSaller.dao.SaledGoodDAO;  @Service("orderProcess")  **public** **class** OrderProcessImpl **implements** OrderProcess {  @Resource(name = "orderDAO")  **private** OrderDAO orderDAO;  @Resource(name = "saledGoodDAO")  **private** SaledGoodDAO saledGoodDAO;  @Resource(name = "ruleMatcher")  **private** DiscountMatcher matcher;  **private** PaymentProcess paymentProcess;  @Override  **public** String createOrder() {  **return** orderDAO.createOrder();  }  @Override  **public** List<ViewSideGood> addGoodAndMatch(ViewSideGood good) {  List<ViewSideGood> viewSideGoods = addGood(good);  matcher.doDiscountRuleMatch(viewSideGoods);  // return viewSideGoods;  **return** batchUpdateGood(viewSideGoods);  }  @Override  **public** List<ViewSideGood> removeGoodAndMatch(ViewSideGood good) {  List<ViewSideGood> viewSideGoods = removeGood(good);  **if** (viewSideGoods.size() > 0) {  matcher.doDiscountRuleMatch(viewSideGoods);  // return viewSideGoods;  **return** batchUpdateGood(viewSideGoods);  } **else** {  **return** **new** ArrayList<ViewSideGood>();  }  }  **private** List<ViewSideGood> addGood(ViewSideGood good) {  **if** (orderDAO.queryOrder(good.getOrderID()) == **null**) {  good.setOrderID(createOrder());  }  saledGoodDAO.addSaledGood(good.getSaledGood());  **return** saledGoodDAO.getGoodsInsameOrder(good);  }  **private** List<ViewSideGood> removeGood(ViewSideGood good) {  saledGoodDAO.removeSaledGood(good.getSaledGood());  **return** saledGoodDAO.getGoodsInsameOrder(good);  }  **private** List<ViewSideGood> batchUpdateGood(List<ViewSideGood> goods) {  // not set rule that matched  **return** saledGoodDAO.batchUpdateGoods(goods);  }  @Override  **public** List<ViewSideGood> getAddedGoodsByOrder(String orderID) {  // **TODO** Auto-generated method stub  **return** **null**;  }  @Override  **public** **void** finishOrder() {  // **TODO** Auto-generated method stub  }  } |

## 规则匹配

### 规则匹配器控制器

|  |
| --- |
| **package** com.superSaller.beans.checkout.Impl;  **import** java.time.LocalDateTime;  **import** java.util.ArrayList;  **import** java.util.List;  **import** javax.annotation.Resource;  **import** org.springframework.stereotype.Service;  **import** com.superSaller.beans.checkout.DiscountMatcher;  **import** com.superSaller.beans.checkout.entities.DiscountRule;  **import** com.superSaller.beans.checkout.entities.Present;  **import** com.superSaller.beans.checkout.entities.ViewSideGood;  **import** com.superSaller.dao.RuleDAO;  @Service(value = "ruleMatcher")  **public** **class** DiscountRuleMatcherImpl **implements** DiscountMatcher {  **private** List<ViewSideGood> goods;  **private** List<DiscountRule> rules;  **private** List<Present> presents;  @Resource(name = "ruleDAO")  **private** RuleDAO ruleDAO;  **public** DiscountRuleMatcherImpl() {  }  @Override  **public** List<ViewSideGood> doDiscountRuleMatch(List<ViewSideGood> goods) {  **this**.goods = goods;  doRuleValidity();  rules = ruleDAO.queryAllRules(); // for test  doRuleMatch();  goods = **this**.goods;  **return** goods;  }  /\*\*  \*  \* **@param** customer  \* **@param** rules  \*/  **private** **void** doRuleMatch() {  List<DiscountRule> partlyRules = **new** ArrayList<DiscountRule>();  List<DiscountRule> globalRule = **new** ArrayList<DiscountRule>();  **for** (DiscountRule rule : rules) {  **if** (rule.getBundleGoods().size() == 0) {  globalRule.add(rule);  } **else** {  partlyRules.add(rule);  }  }  **for** (DiscountRule rule : partlyRules) {  doDetailMatch(rule);  }  **for** (DiscountRule rule : globalRule) {  doDetailMatch(rule);  }  }  **private** **void** doDetailMatch(DiscountRule rule) {  **switch** (rule.getType()) {  **case** "FULLFREE":  **new** FULLFREEMatcher().match(rule, goods);  **break**;  **case** "FULLCOUNT":  **new** FULLCOUNTMatcher().match(rule, goods);  **break**;  **case** "FULLPRESENT":  **break**;  **case** "FULLVOUCHER":  **break**;  **case** "BUYPRESENT":  **break**;  **case** "BUYPRESET":  **break**;  **case** "BUYSPECIAL":  **break**;  **case** "BUYFREE":  **new** BUYFREEMatcher().match(rule, goods);  **break**;  **case** "BUYCOUNT":  **break**;  **case** "BUYVOUCHER":  **break**;  }  }  // valid rules  **private** **void** doRuleValidity() {  List<DiscountRule> rules = ruleDAO.queryAllRules();  List<DiscountRule> filteDiscountRules = **new** ArrayList<DiscountRule>();  **for** (DiscountRule discountRule : rules) {  **boolean** flag = **true**;  LocalDateTime now = LocalDateTime.*now*();  **int** miniteOfDayOfNow = now.getHour() \* 60 + now.getMinute();  **if** (discountRule.getDatePeriodStart().isAfter(LocalDateTime.*now*())) {  // DATE NOT BEGIN  flag = **false**;  } **else** **if** (discountRule.getDatePeriodEnd().isBefore(LocalDateTime.*now*())) {  // DATE ALREADY PASSED  flag = **false**;  } **else** **if** (discountRule.getDayPeriodStart() > miniteOfDayOfNow) {  // TIME NOT BEGIN  flag = **false**;  } **else** **if** (discountRule.getDayPeriodEnd() < miniteOfDayOfNow) {  // TIME ALREADY PASSED  flag = **false**;  }  **if** (flag) {  filteDiscountRules.add(discountRule);  }  }  **this**.rules = filteDiscountRules;  }  } |

### 基础匹配处理器

|  |
| --- |
| **package** com.superSaller.beans.checkout.Impl;  **import** java.util.Map;  **import** java.util.Set;  **public** **class** BasicGoodsMatcher {  **public** **boolean** matchBundle(Map<String, Double> toCantain, Map<String, Double> toBeCotained) {  // **TODO** might happen that goods in chart is zero  **boolean** matched = **false**;  Set<String> keysToBeContained = toBeCotained.keySet();  **if** (toCantain.keySet().containsAll(keysToBeContained)) {  matched = **true**;  **for** (String string : keysToBeContained) {  **if** (toCantain.get(string) < toBeCotained.get(string)) {  matched = **false**;  **break**;  }  }  }  **return** matched;  }  } |

### FULLFREE匹配处理器

|  |
| --- |
| **package** com.superSaller.beans.checkout.Impl;  **import** java.util.ArrayList;  **import** java.util.List;  **import** java.util.Map;  **import** java.util.Set;  **import** java.util.TreeMap;  **import** com.superSaller.beans.checkout.entities.DiscountRule;  **import** com.superSaller.beans.checkout.entities.ViewSideGood;  **public** **class** FULLFREEMatcher **extends** BasicGoodsMatcher {  **public** List<ViewSideGood> match(DiscountRule rule, List<ViewSideGood> goods) {  **if** (matchedFilter(rule, goods)) {  **return** goods;  }  **double** totalMoney = 0, sum = 0, averageFreeMoney;  Map<String, Double> goodsNums = **new** TreeMap<String, Double>();  **for** (ViewSideGood viewSideGood : goods) {  sum += viewSideGood.getSum();  totalMoney += viewSideGood.getSaledPrice() \* viewSideGood.getSum();  goodsNums.put(viewSideGood.getGoodID(), viewSideGood.getSum());  }  **if** (rule.getBundleGoods().size() == 0) {  **if** (totalMoney >= rule.getConditionValue()) {  **double** discountMoney = (totalMoney - rule.getFreeMoney()) \* (100 - rule.getDiscountRate()) / 100;  averageFreeMoney = (discountMoney + rule.getFreeMoney()) / sum;  **for** (ViewSideGood viewSideGood : goods) {  viewSideGood.setSaledPrice(viewSideGood.getSaledPrice() - averageFreeMoney);  List<String> list = viewSideGood.getRuleID();  list.add(rule.getUUID());  }  }  } **else** **if** (matchBundle(goodsNums, rule.getBundleGoods())) {  Set<String> ids = rule.getBundleGoods().keySet();  totalMoney = sum = averageFreeMoney = 0;  List<ViewSideGood> bundleGoods = **new** ArrayList<ViewSideGood>();  **for** (ViewSideGood viewSideGood : goods) {  **if** (ids.contains(viewSideGood.getGoodID())) {  bundleGoods.add(viewSideGood);  }  }  **for** (ViewSideGood viewSideGood : bundleGoods) {  sum += viewSideGood.getSum();  totalMoney += viewSideGood.getSaledPrice() \* viewSideGood.getSum();  }  **double** discountMoney = (totalMoney - rule.getFreeMoney()) \* (100 - rule.getDiscountRate()) / 100;  averageFreeMoney = (discountMoney + rule.getFreeMoney()) / sum;  **for** (ViewSideGood viewSideGood : bundleGoods) {  viewSideGood.setSaledPrice(viewSideGood.getSaledPrice() - averageFreeMoney);  List<String> list = viewSideGood.getRuleID();  list.add(rule.getUUID());  }  }  **return** goods;  }  **private** **boolean** matchedFilter(DiscountRule rule, List<ViewSideGood> goods) {  // **TODO** when saled goods rules and orders dao finished, add real filter.  **boolean** flag = **false**;  **for** (ViewSideGood viewSideGood : goods) {  **if** (rule.getUUID().equals(viewSideGood.getRuleID())) {  flag = **true**;  **break**;  }  }  **return** flag;  }  } |

### FULLCOUNT匹配处理器

|  |
| --- |
| **package** com.superSaller.beans.checkout.Impl;  **import** java.util.ArrayList;  **import** java.util.List;  **import** java.util.Map;  **import** java.util.Set;  **import** java.util.TreeMap;  **import** com.superSaller.beans.checkout.entities.DiscountRule;  **import** com.superSaller.beans.checkout.entities.ViewSideGood;  **public** **class** FULLCOUNTMatcher **extends** BasicGoodsMatcher {  **public** List<ViewSideGood> match(DiscountRule rule, List<ViewSideGood> goods) {  **double** totalMoney = 0, sum = 0, averageFreeMoney;  Map<String, Double> goodsNums = **new** TreeMap<String, Double>();  **for** (ViewSideGood viewSideGood : goods) {  sum += viewSideGood.getSum();  totalMoney += viewSideGood.getSaledPrice() \* viewSideGood.getSum();  goodsNums.put(viewSideGood.getGoodID(), viewSideGood.getSum());  }  **if** (rule.getBundleGoods().size() == 0) {  **if** (totalMoney >= rule.getConditionValue()) {  **double** discountMoney = totalMoney \* (100 - rule.getDiscountRate()) / 100;  averageFreeMoney = discountMoney / sum;  **for** (ViewSideGood viewSideGood : goods) {  viewSideGood.setSaledPrice(viewSideGood.getSaledPrice() - averageFreeMoney);  List<String> list = viewSideGood.getRuleID();  list.add(rule.getUUID());  }  }  } **else** **if** (matchBundle(goodsNums, rule.getBundleGoods())) {  Set<String> ids = rule.getBundleGoods().keySet();  totalMoney = sum = averageFreeMoney = 0;  List<ViewSideGood> bundleGoods = **new** ArrayList<ViewSideGood>();  **for** (ViewSideGood viewSideGood : goods) {  **if** (ids.contains(viewSideGood.getGoodID())) {  bundleGoods.add(viewSideGood);  }  }  **for** (ViewSideGood viewSideGood : bundleGoods) {  sum += viewSideGood.getSum();  totalMoney += viewSideGood.getSaledPrice() \* viewSideGood.getSum();  }  **double** discountMoney = totalMoney \* (100 - rule.getDiscountRate()) / 100;  averageFreeMoney = discountMoney / sum;  **for** (ViewSideGood viewSideGood : bundleGoods) {  viewSideGood.setSaledPrice(viewSideGood.getSaledPrice() - averageFreeMoney);  List<String> list = viewSideGood.getRuleID();  list.add(rule.getUUID());  }  }  **return** goods;  }  } |

### BUYFREE匹配处理器

|  |
| --- |
| **package** com.superSaller.beans.checkout.Impl;  **import** java.util.ArrayList;  **import** java.util.List;  **import** java.util.Map;  **import** java.util.Set;  **import** java.util.TreeMap;  **import** com.superSaller.beans.checkout.entities.DiscountRule;  **import** com.superSaller.beans.checkout.entities.ViewSideGood;  **public** **class** BUYFREEMatcher **extends** BasicGoodsMatcher {  **public** List<ViewSideGood> match(DiscountRule rule, List<ViewSideGood> goods) {  **double** totalMoney = 0, sum = 0, averageFreeMoney = 0, freeMoney = Double.***MAX\_VALUE***;  Map<String, Double> goodsNums = **new** TreeMap<String, Double>();  **for** (ViewSideGood viewSideGood : goods) {  sum += viewSideGood.getSum();  totalMoney += viewSideGood.getSaledPrice() \* viewSideGood.getSum();  goodsNums.put(viewSideGood.getGoodID(), viewSideGood.getSum());  freeMoney = viewSideGood.getSaledPrice() < freeMoney ? viewSideGood.getSaledPrice() : freeMoney;  }  **if** (rule.getBundleGoods().size() == 0) {  **if** (sum >= rule.getConditionValue()) {  freeMoney = rule.getFreeMoney() > 0 ? rule.getFreeMoney() : freeMoney;  **double** discountMoney = (totalMoney - freeMoney) \* (100 - rule.getDiscountRate()) / 100;  averageFreeMoney = (discountMoney + freeMoney) / sum;  **for** (ViewSideGood viewSideGood : goods) {  viewSideGood.setSaledPrice(viewSideGood.getSaledPrice() - averageFreeMoney);  List<String> list = viewSideGood.getRuleID();  list.add(rule.getUUID());  }  }  } **else** **if** (matchBundle(goodsNums, rule.getBundleGoods())) {  Set<String> ids = rule.getBundleGoods().keySet();  totalMoney = sum = averageFreeMoney = 0;  freeMoney = Double.***MAX\_VALUE***;  **int** minMatchTimes = Integer.***MAX\_VALUE***;  List<ViewSideGood> bundleGoods = **new** ArrayList<ViewSideGood>();  **for** (ViewSideGood viewSideGood : goods) {  **if** (ids.contains(viewSideGood.getGoodID())) {  bundleGoods.add(viewSideGood);  }  }  **for** (ViewSideGood viewSideGood : bundleGoods) {  sum += viewSideGood.getSum();  totalMoney += viewSideGood.getSaledPrice() \* viewSideGood.getSum();  **int** min = (**new** Double(viewSideGood.getSum())).intValue();  minMatchTimes = min < minMatchTimes ? min : minMatchTimes;  freeMoney = viewSideGood.getSaledPrice() < freeMoney ? viewSideGood.getSaledPrice() : freeMoney;  }  **int** min = (**new** Double(rule.getConditionValue())).intValue();  minMatchTimes = min < minMatchTimes ? min : minMatchTimes;  System.***out***.println(freeMoney + " " + minMatchTimes);  freeMoney = freeMoney \* minMatchTimes;  **double** discountMoney = (totalMoney - freeMoney) \* (100 - rule.getDiscountRate()) / 100;  averageFreeMoney = (discountMoney + freeMoney) / sum;  **for** (ViewSideGood viewSideGood : bundleGoods) {  viewSideGood.setSaledPrice(viewSideGood.getSaledPrice() - averageFreeMoney);  List<String> list = viewSideGood.getRuleID();  list.add(rule.getUUID());  }  }  **return** goods;  }  } |

# 尚未解决的问题

1. 收银功能；
2. 规则管理功能；
3. 员工管理；
4. 规则匹配可能出现多次重复优惠；
5. 订单的取消功能；
6. 未完成订单读取功能；
7. 规则的编辑；
8. 用户账户登录次数保护。