南 阳 理 工 学 院

本科生毕业设计（论文）

学院（系）：信息工程学院

专 业 ：通信工程

学 生 ：黄照光

指导 教师 ：吴冉

完成日期 2020 年 05 月

南阳理工学院本科生毕业设计（论文）

基于Java的快点餐系统的设计与实现

Design and Implementation of Quick Meal System Based on Java

总计：毕业设计（论文）39页

表格： 6 个

插图： 11 幅

南阳理工学院本科毕业设计（论文）

基于Java的快点餐系统的设计与实现

**Design and Implementation of Quick Meal System Based on Java**

学院(系)：信息工程学院

专 业：通信工程

姓 名：黄照光

学 号：1606735067

年级班级：16通信二班

指导教师：吴冉

评阅教师：

完成日期：

南阳理工学院

Nanyang Institute of Technology

基于Java的快点餐系统的设计与实现

[摘 要]：针对现代餐饮行业的传统服务模式已经不能满足实际需求的缺点，文中设计了一种在线点餐的系统。该系统后端整体框架使用SpringBoot框架，前端框架使用Jsp技术，Shiro作为该系统的登录权限认证，Tomcat作为消息中间件，Redis作数据库缓存，Jmeter作性能压力测试，Git实现版本控制，MySQL作数据库，完成点餐系统的设计和实现。该系统具备在线点餐、登录服务、热销推荐等功能，取代传统的人工方式，提高了工作效率，为商家带来了更多的利润。

[关键词]：Java；快点餐；Redis；Git；MySQL

**Design and Implementation of** **Quick Meal System Based on Java**

**Abstract:** In view of the shortcomings that the traditional service mode of modern catering industry can not meet the actual needs, this paper designs an online ordering system. The whole back-end framework of the system uses the spring boot framework, the front-end framework uses the JSP technology, Shiro as the login authority authentication of the system, Tomcat as the message middleware, Redis as the database cache, JMeter as the performance stress test, GIT realizes version control, MySQL as the database, these complete the design and implementation of the ordering system. The system has the functions of online ordering, login service, hot selling recommendation, and so on. It can replace the traditional manual mode, improve the working efficiency , and bring more profits for businesses.

**Key Words****:** Java; Quick Meal ; Redis; Git; MySQL

目 录

[1 绪论 1](#_Toc10560)

[1.1 引言 1](#_Toc18304)

[1.2 点餐系统的研究背景 1](#_Toc22342)

[1.3 点餐系统的研究目的 1](#_Toc28822)

[1.4 点餐系统的研究工作 1](#_Toc29228)

[2 点餐系统相关的技术介绍 3](#_Toc20104)

[2.1 SpringBoot技术 3](#_Toc30991)

[2.2 JSP技术 3](#_Toc20949)

[2.3 Shiro技术 3](#_Toc14560)

[2.4 Redis技术 3](#_Toc91)

[2.5 Git技术 3](#_Toc521)

[2.6 MySql技术 4](#_Toc19150)

[2.7 Jmeter技术 4](#_Toc17534)

[3 点餐系统需求的调查以及实现 5](#_Toc25379)

[3.1 需求分析 5](#_Toc3331)

[3.2 功能模块设计 5](#_Toc17189)

[3.2.1 商家管理模块 6](#_Toc30188)

[3.2.2 用户点餐模块 6](#_Toc19651)

[4 数据库设计 7](#_Toc28329)

[4.1 概念结构设计 7](#_Toc31206)

[4.2 逻辑结构设计 7](#_Toc6357)

[5 功能模块的实现 11](#_Toc32710)

[5.1 登录模块的实现 11](#_Toc11944)

[5.2 首页模块的实现 12](#_Toc10776)

[5.3 商品管理的实现 14](#_Toc17905)

[5.4 订单管理功能的实现 15](#_Toc5268)

[5.5 修改密码功能的实现 17](#_Toc7186)

[5.6 线上付款功能的实现 18](#_Toc23958)

[6 系统测试 20](#_Toc14983)

[6.1 功能测试 20](#_Toc748)

[6.2 性能测试 21](#_Toc259)

[总结 22](#_Toc25114)

[参考文献 23](#_Toc18736)

[致谢 24](#_Toc24462)

[附录 25](#_Toc14725)

# 1 绪论

## 1.1 **引言**

随着科技文明的不断发展，人们的生活水平也在逐渐提高，并且人们的消费观念和消费方式也有了明显的改变[1]。人们在购买自己想要的商品时，线上支付的方式也流行于日常生活中。这种新的行为习惯也改变着餐饮行业，但是大多数餐饮店由于模式观念转变较慢，技术支持不足或其他原因仍采用传统的消费模式，这种方式不但浪费人力物力，而且相比较其他已支持线上点餐的商家会流失大量客户[2]。因此，线上点餐的模式成了当今人们在餐厅满足自己需求的趋势。而本系统就是建立这种需求之上，使用B/S模式，商家在后台直接上架商品，其中有商品的名称、价钱、库存、口味、备注，商品上架后可在首页看到。顾客还能选择销量最高的、最新上架的或者直接根据关键字搜索自己想要的商品，大大增加了用户体验感。

## 1.2 点餐系统的研究背景

随着网络在生活中的普及，互联网已经成为人们目前最习惯使用的生活助手，人们可以足不出户的进行生活消费，甚至可以通过线上购买自己想要的商品[3]。但是在传统的餐饮模式下，餐饮店只能通过服务员根据菜单将菜品展示给用户，流程十分繁琐，而且人力物力和门面的装修也需要倾覆很大的心血，用户的体验度也并没有得到有效提升。不但如此，餐厅在对菜品的管理和维护上也大多采取手动工作的方式，其技术含量和工作效率都较低，难以适应现代化经营管理的要求[4]。线上点餐也由此成为了商家与用户迫切的需求。

## 1.3 点餐系统的研究目的

首先饮食是人们生活必不可少的一种行为，大多数用户为了提高生活水平都会到餐厅进行饮食消费。其次现代生活的快速化使得人们越来越倾向于在餐厅自由点餐，迅速浏览商家的热销菜品以及对应评价并进行线上支付[5]。另外，互联网经过持续的发展，尤其是局域网技术的日益强大，为很多餐饮行业构建综合管理系统提供了技术支持；而且随着计算机硬件的成本下降，经济适用性也得到了保障[6]。综合以上原因，开发点餐系统具有十分重要的意义，极大地推进了餐饮行业的进步与发展。

## 1.4 点餐系统的研究工作

通过搜集相关资料并进行理论的整理和分析之后，文中结合其他餐厅使用点餐系统的经验和Java技术广泛的应用领域，设计并实现一个基于SpringBoot框架的点餐系统[7]。该系统包括用户注册登录、卖家添加/修改商品、买家点餐、商品付款、商品配送、购物车管理、买家留言、更改密码、发送邮件、订单管理、信息修改等主要功能。本文一共包含6章，各个章节安排如下：

第1章为绪论。介绍了论文的引言、研究背景、研究目的、研究工作以及论文的组织结构。

第2章为相关技术概述。介绍了该系统中用到的一些前沿技术以及如何使用。

第3章为需求调查以及实现。分析该系统要实现的需求，总结功能模块和每个模块的内容。

第4章为数据库设计。从两方面阐述该系统中数据库的设计思想，一方面是概念逻辑设计：建立E-R图，分析抽象实体和实体类之间的关系；另一方面是逻辑结构设计：分析各个表的整体内容以及每个表的字段设计。

第5章为功能模块的实现。较为详细地描述一些重要功能模块的具体实现，包括设计思想、重要部分代码和页面截图。

第6章为系统测试。该章节分为功能测试和性能测试，通过编写简单的测试用例完成功能的测试；通过jmeter软件完成性能压力的测试。

# 2 点餐系统相关的技术介绍

## 2.1 SpringBoot技术

说起SpringBoot框架，就要先说一下Spring。Spring是Java企业版的轻量级代替品，通过IOC和AOP技术，用简易的Java对象就可以实现EJB的功能[8]。而Spring Boot 框架整体是基于 Spring 开发完成的，它对Spring进行了改善和优化，基于约定优于配置的思想，能够让研发人员不需要在项目配置和业务逻辑之间进行思维方式的转换，只需关注业务代码的编写，很大程度上提升了开发效率，从而缩短了项目完成的周期[9]。

## 2.2 JSP技术

JSP全称Java Server Pages，是一种动态网页开发技术。它使用JSP标签在HTML网页中插入Java代码，标签通常以<%开头，以%>结束。JSP是一种Java servlet，通常在项目中用JSP页面交互数据库，记录用户信息，还可以在不同的页面中传递一些公共的信息以减少资源的浪费[10]。

## 2.3 Shiro技术

Apache Shiro是一种基于Java的安全框架，功能强大而且使用简单，它为开发人员提供一个可视化高且全面的认证，授权，加密及会话管理的解决方案[11]。Shiro有3个核心组件分别为Subject，SecurityManager和Realms，SecurityManager是Shiro框架的核心技术，用来管理用户的安全操作。本系统用Shiro来完成项目的登录认证和基本权限控制。

## 2.4 Redis技术

Redis是C语言开发的一个开源的（遵从BSD协议）高性能键值对（key-value）的内存数据库，可以作数据库、缓存、消息中间件等[12]。它是一种NoSQL（not-only sql）的数据库。它性能十分优秀，因为数据存储在内存中，相比较其他数据库读写速度非常快，并且支持并发10万QPS。并且Redis是单进程单线程，采用IO多路复用机制来提高效率。Redis还支持数据持久化，因为它可以将内存中的部分数据保存在磁盘中，重启时就会加载[12]。

## 2.5 Git技术

Git是当前最高效的分布式版本控制系统。它是一个开源的分布式版本控制软件，可以有效、高速地处理从很小到非常大的项目版本管理。它主要用于版本回退和多人同时开发，与常用的CVS，Subversion等不同，它采用了分布式版本库的方式，不需要服务器端软件支持[13]。在本系统中，把项目代码放到GitHub上，用Git做版本控制，可以看到历史提交记录和每次提交的备注。

## 2.6 MySql技术

MySql是一种[关系型数据库](http://baike.baidu.com/view/1450387.htm" \t "http://www.360doc.com/content/14/0213/16/_blank)，它使用的是标准化数据库语言，它将数据保存在不同的数据表中，这样数据的关系将更加清晰，数据的查找也更加的方便[14]。开发者对数据处理的效率会更加高效，大大方便开发者的操作也方便于生产的实际应用。

## 2.7 Jmeter技术

Apache JMeter是Apache组织开发的基于Java的压力测试工具，用于对项目中的接口做性能测试，也叫压力测试[15]。它以线程组创建多线程的方式模拟大量的请求，在巨大的请求压力下测试系统的整体性能，通过分析结果中的平均响应时间、吞吐量和错误率来判断系统性能是否符合要求。

# 3 点餐系统需求的调查以及实现

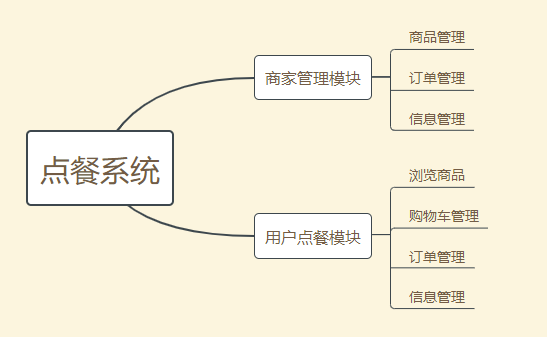
## 3.1 需求分析

该点餐系统中存在卖家和买家两种角色。点餐系统的卖家是餐饮店的老板，而买家一般为购买商品的用户。对于用户而言，会有浏览商品和购买商品的需求，还有管理与自己相关的用户信息的需求。对于卖家而言，则会有对商品基本信息管理、对商品分类管理以及对个人信息管理的需求。

考虑到系统移植的普适性、轻量级开发和减少系统搭建难度，项目整体架构采用SpringBoot框架，并且以注解相辅助的方式进行业务逻辑的开发，持久层框架使用Mybatis，它可以基本不写SQL语句并且使大部分SQL语句更加灵活，配置也相对于Hibernate框架更为简单轻便，对事物的控制更加简便灵活，也可以很大程度上的防止SQL注入攻击来提升系统的安全性。

## 3.2 功能模块设计

点餐系统的功能模块分为以下两大内容：商家管理模块和用户点餐模块。功能模块具体如图3-1所示。

图3-1 点餐系统功能模块分布图

### 3.2.1 商家管理模块

商家管理模块大致分为商品管理、订单管理和信息管理三部分。

商品管理：该模块商家可以对本店铺商品进行添加、删除、更改等，可以设置商品的数量，商品的上下架状态，上传商品的图片，还有商品的口味，商品的价格和库存，当然还可以增加商品展示时的备注说明，同时在商品展示时还有“已售数量”来表示该商品的受欢迎程度。

订单管理：该模块商家可以看到某笔订单的详情，包括商品名称、商品数量、用户的备注、下单时间、期望到达时间、订单地址、订单的支付方式以及订单状态。商家还可以进行订单状态的更新，及时将订单的处理通知给用户，比如已发货、未发货、已送达、带配送等物流信息。当订单状态变为已送达后，商家还能查看用户对该订单的评分以及评价。

信息管理：该模块商家可以对个人信息进行管理。可以修改门店的地址和商家的名称，商家手机号，以及商家登录该系统账号的密码，本系统对密码采用MD5密文处理，把密码转成32位的随机字母存储到数据库，较大程度上保证用户信息的安全性。

### 3.2.2 用户点餐模块

用户点餐模块大致分为浏览商品、购物车管理、订单管理和信息管理四部分。

浏览商品：该模块是用户进入该系统后的首页展示，用户可以看到展示的商品，可以根据默认、最新、热销三种种类查看不同的商品，同时也可以在首页按商品的名字模糊搜索想吃的商品，首页商品展示还显示已售数量供用户参考。

购物车管理：从首页可以直接对某商品进行添加购物车的操作，也可以进入商品详情页后再添加相应的数量到购物车，也有对购物车的一系列管理，包括下订单，再次加入购物车。

订单管理：用户的订单管理不同于商家，商家的订单管理只能管理顾客的订单状态，而用户的订单管理是对自己的某个或多个订单的操作。用户在下订单的时候，配送地址可以选择默认的，也可以选择新建，新建地址是在一个新的页面完成的。配送时间也有几个时间段可供选择，还可以加备注等功能。

信息管理：用户的信息管理同商家的一样。

# 4 数据库设计

## 4.1 概念结构设计

好的数据库设计能够节省整个系统的开发时间，并且也可以规避上线运行之后的很多Bug，另外对系统的扩展、延伸、移植、高并发都有很大的帮助。数据库的结构是为了显示更多抽象实体和实体类之间的关系。这种关系表现在经常提到的E-R图的形式中，其中有一对一联系（1：1），一对多联系（1：n），多对多联系（m：n）三种关系。

在该项目中，主要存在八个实体类，也就对应数据库中的八张表，分别是用户信息表、商品表、商品图片表、订单表、购物车表、订单详情表、菜单表、用户地址表。该点餐系统数据库的E-R图如图4-1所示。

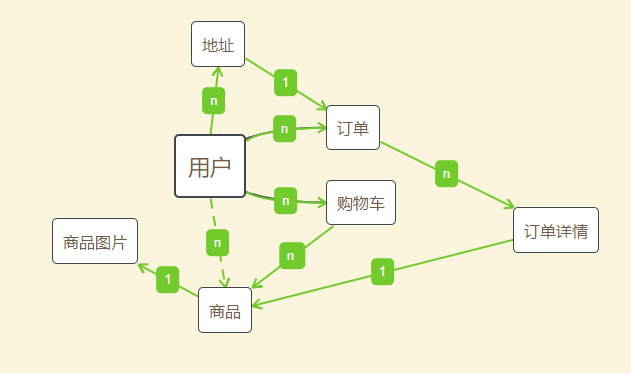


图4-1 点餐系统数据库E-R图

## 4.2 逻辑结构设计

点餐系统使用MySql数据库进行数据表单的创建，根据分析，数据库表共有8张，由于篇幅的限制，在这里以其中五张进行说明：用户信息表（user）、用户地址表（address）、商品信息表（product）、购物车表（car）、订单表（orders），具体如下：

（1）用户信息表（user），主要存储用户的基本常用信息，比如用户邮箱（账号）、账号密码、昵称、真实姓名、手机、角色、性别等。具体如表4-1所示。

表4-1 user（用户信息表）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 数据名称 | 数据的类型 | 数据大小(bit) | 数据解释 |
| 1 | user\_id | bigint | 20 | 用户id |
| 2 | user\_email | varchar | 30 | 用户邮箱 |
| 3 | user\_password | varchar | 50 | 用户密码 |
| 4 | user\_nickname | varchar | 30 | 用户昵称 |
| 5 | user\_realname | varchar | 30 | 用户真实姓名 |
| 6 | user\_phone | varchar | 20 | 用户电话 |
| 7 | user\_rol | char | 1 | 角色（0顾客、1卖家） |
| 8 | user\_sex | char | 1 | 性别（0女、1男） |

（2）用户地址表（address），主要存储用户地址的相关信息，比如用户地址详情、是否是默认地址等，具体如表4-2所示。

表4-2 address（用户地址表）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 数据名称 | 数据的类型 | 数据大小(bit) | 数据解释 |
| 1 | address\_id | bigint | 20 | 地址id |
| 2 | address\_buyer\_id | bigint | 20 | 对应用户id |
| 3 | address\_address | varchar | 200 | 用户地址详情 |
| 4 | address\_is\_default | char | 1 | 是否是默认地址 |

（3）商品信息表（product），主要存储商品的基本详情，比如商品编号、名称、价格、库存、口味、是否上架、备注、已售数量等，具体如表4-3所示。

表4-3 product（商品信息表）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 数据名称 | 数据的类型 | 数据大小(bit) | 数据解释 |
| 1 | product\_id | bigint | 20 | 商品id |
| 2 | product\_no | varchar | 30 | 商品编号 |
| 3 | product\_name | varchar | 50 | 用户地址详情 |
| 4 | product\_seller\_id | bigint | 20 | 商家id |
| 5 | product\_price | double | 10 | 商品价格 |
| 6 | product\_stock | int | 255 | 商品库存 |
| 7 | product\_taste | char | 1 | 辣不辣（ 0否1是） |
| 8 | product\_is\_sell | char | 1 | 是否上架 （0否1是） |
| 9 | product\_description | varchar | 100 | 商品描述 |
| 10 | product\_picture | varchar | 50 | 商品图片位置 |
| 11 | product\_has\_selled | int | 11 | 商品已售数量 |

（4）购物车表（car），主要存储购物车的基本详情，比如购物车对应的用户、购物车内的商品详情、商品数量、商品价格等，具体如表4-4所示。

表4-4 car（购物车表）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 数据名称 | 数据的类型 | 数据大小(bit) | 数据解释 |
| 1 | car\_id | bigint | 20 | 购物车id |
| 2 | car\_buyer\_id | bigint | 20 | 对应用户id |
| 3 | car\_product\_id | bigint | 20 | 对应商品id |
| 4 | car\_product\_num | int | 11 | 对应商品数量 |
| 5 | car\_product\_price | double | 10 | 对应商品价格 |

（5）订单表（orders），主要存储订单的基本详情，比如订单编号、要求配送地址、订单开始时间、订单结束时间、要求送达时间、订单总价格、订单支付状态、订单备注等，具体如表4-5所示。

表4-5 orders（订单表）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 数据名称 | 数据的类型 | 数据大小(bit) | 数据解释 |
| 1 | order\_id | bigint | 20 | 订单id |
| 2 | order\_buyer\_id | bigint | 20 | 用户id |
| 3 | order\_seller\_id | bigint | 20 | 商家id |
| 4 | order\_no | varchar | 30 | 订单编号 |
| 5 | order\_address | varchar | 255 | 配送地址 |
| 6 | order\_time | timestamp | 0 | 订单开始时间 |
| 7 | order\_delieve\_time | timestamp | 0 | 要求送达时间 |
| 8 | order\_finish\_time | timestamp | 0 | 订单结束时间 |
| 9 | order\_total\_price | double | 10 | 订单总价格 |
| 10 | order\_payment | varchar | 20 | 订单支付状态 |
| 11 | order\_status | char | 2 | 订单状态 |
| 12 | order\_comments | varchar | 1000 | 订单备注 |

# 5 功能模块的实现

## 5.1 登录模块的实现

网站的登录模块一直都是系统中最关键的部分，用户只有通过验证账号密码的正确性才能够进入系统网站。用户在登录页面填入自己的账号密码，随后点击提交button按钮进行登录操作。在login.jsp的文件中，共调用了四种标签，使用div标签输入并展示用户信息输入框，第二种使用p标签并加粗加红为用户提示错误信息。第三种使用input标签，在form表单里面可以使用户输入其信息，并通过第四种button按钮实现点击登录事件。

在登录功能的实现原理上，当用户点击登录的时候，系统会将用户输入或选择的内容异步（ajax）传递给后台控制器LoginController，主要是使用Subject中的login方法对账号密码进行验证，主要代码如下（完整代码详见附录1）：

Subject subject = SecurityUtils.getSubject();

UsernamePasswordToken token = new UsernamePasswordToken (user.getUser Email(), MD5Util.MD5(user.getUserPassword()));

try {

subject.login(token);

user.setUserPassword(MD5Util.MD5(user.getUserPassword()));

List<User> userList = loginService.getUser(user);

if (userList.size() <= 0) {

model.addAttribute("message", "用户名或密码不正确！");

return "login";

} else {

request.getSession().setAttribute("user", userList.get(0));

return "redirect:index";

}

 如代码所示，直接将input（text）中的value属性的值传递给后端的Model对象。之后通过UsernamePasswordToken对象将用户名密码转换成token，再通过subject的login方法进行登录认证，认证成功后根据账号查询得到用户的基本信息存储到session中以便页面显示使用，用户名密码在MD5加密后解析更具有安全性。如果身份信息校验成功则跳转首页，如若校验失败则会显示“账号或密码不正确”。另外还可以在登录页面注册新用户和更改密码操作，如下图5-1所示：



图5-1 点餐系统登录页面

## 5.2 首页模块的实现

登录后进入首页，查询商品表的全部信息，为了提高并发性，选择把商品分页显示在页面上，分页功能用的是pagehelper，它能简单高效的把查询到的数据进行二次封装。首页有最新、默认、销量最高三种分类，对应的是持久层框架mybatis的条件搜索，再加上在首页还能根据商品的关键字进行模糊搜索与价格的间距搜索，更加体现了持久层框架mybatis的灵活性。主要的代码如下（完整代码详见附录2）：

//根据输入条件，进行组合查询

ProductExample example1 = new ProductExample();

ProductExample.Criteria criteria1 = example1.createCriteria();

//确定最高价和最低价

criteria1.andProductPriceBetween(min, max);

//上架的才显示 不上架的不显示

criteria1.andProductIsSellEqualTo("1");

//判断商品的状态

if (status != 0) {

if (status == 1) {

//按照最新查询

example1.setOrderByClause("product\_id DESC");

} else {

//按照卖出最多查询

example1.setOrderByClause("product\_hass\_selled DESC");

}

}

//根据搜索的关键词在进行查询

if (word != null) {

criteria1.andProductNameLike("%" + word + "%");

}

在跳转首页的时候，就会把分页、商品分类状态、价格最小和最大、商品关键字通过from表单或者url参数传到后端的PageController控制器，再根据mybatis分页查询后紧接着跳转回首页显示。另外在首页也可以看到商品详情并直接点击加入购物车；用户信息入口，退出登录、购物车、订单列表都设计在在右上角，方便观察。点餐系统的首页如下图5-2所示。



图5-2 点餐系统首页页面

## 5.3 商品管理的实现

商品管理功能是在商家角色下进入首页才有的功能，是商家对本店菜品的添加、修改、删除等一系列操作。在一般场景中，商品列表和商品详情信息页，应该是访问很频繁的页面，对于这些页面中的记录，一般是需要频繁查询数据库的，这样对数据库会产生很大的压力，有可能导致查询很慢，这对用户的交互感非常差。因此本系统对于一些需要经常查询数据库的记录，会把这些记录放进Redis缓存中，这样一来就会减少数据库的压力，而且访问Redis也是非常的快速。商品列表页面在前端代码层面大多数都是一些点击事件和点击提交函数，对应的是JavaScript中的一些click函数，还有ajax局部刷新请求数据。在商品详情获取的后端处理器中，首先通过session获取会话中的用户的信息，如果用户对象为空则获取要查看的商品id，根据商品id查询商品的基本详情，包括图片、价格、库存、口味、和备注等信息（完整代码详见附录3）。点餐系统管理商品列表页面如下图5-3所示。



图5-3 点餐系统管理商品列表页面

在添加商品页面，其中上传图片部分，后端用的是MultipartFile，用它做上传图片的功能是非常简单的，其中file.transferTo(imgFile)代码则是重点，把图片转成io流进行解析。本系统把上传后的图片保存在本地系统中，之后再根据图片的路径在前端页面显示图片。而对于上面说的Redis缓存，还有一个小问题，就是当数据库中数据更新的时候，Redis中相应的缓存数据就不能用了，因为会造成数据不一致的问题。假如一条记录在数据库更改后，再访问数据就会去查询数据库，该系统的解决办法是先删除Redis中的数据，数据库中更新后再插入Redis缓存中。点餐系统修改商品页面如下图5-4所示。



图5-4 点餐系统修改商品页面

## 5.4 订单管理功能的实现

订单管理主要是订单状态的改变。当点击时，首先根据该商品的id查询出该商品的相关信息，同时加载当前用户的收货地址信息。汇总了该商品和用户信息之后提交订单。这时会利用代码在redis里生成一个13位的order id，同时将该id和商品信息存入订单表中。订单id为了保证不重复，该系统的做法是时间戳加5位随机数，保证订单id的唯一性。对于订单的状态总共有12种，包括0：已确定-未付款，1：已确定-已付款-未发货，2：已确定-已付款-已发货-待收货，3：已确定-已付款-已发货-已收货，4：已确定-货到付款-未发货，5：已确定-货到付款-已发货-未收货，6：已确定-货到付款-已发货-已收货，7：已经评价，8：取消订单，9：退款申请，10：已退款，11：用户不收货，12：超时10分钟未付款，订单取消。而其中的“12”状态超时10分钟未付款，订单取消，在该系统中是通过定时任务中@Scheduled(fixedRate = 600000)注解完成的，时间是10分钟。主要的代码如下：

@Scheduled(fixedRate = 600000)

public void checkOrder() {

log.info("检测未支付订单是否超时...");

OrdersExample ordersExample = new OrdersExample();

OrdersExample.Criteria criteria = ordersExample.createCriteria();

criteria.andOrderStatusEqualTo("0");

List<Orders> orderList = ordersMapper.selectByExample(ordersExample);

orderList.forEach(order->{

Date finishTime = order.getOrderFinishTime();

Date date = new Date();

if (date.getTime() - finishTime.getTime() > 600000) {

order.setOrderStatus("12");

ordersMapper.updateByPrimaryKeySelective(order);

log.info("订单已被取消：" + order.getOrderNo());

}

});

}

另外在订单付完款后，用户还能对某个订单进行评价，而且商家也能看到顾客的评价跟反馈。完成支付之后需要在后台对订单进行审核，把状态改为已发货，返回前台页面可以看到订单的信息已经发生了更新。本系统没有模拟物流系统所以当我们点击确认收货，后台处理程序则更改本次订单的收货状态字段，并将评价字段置为待评价，并且付款方式也支持线上支付和货到付款两种支付方式。

当收货之后，商品进入了待评价阶段，点击评价进入评价界面，点击发表评论按钮异步提交评论信息到后台controller控制器进行数据的插入，如果出现错误则根据返回的错误信息进行提示，如果执行成功则跳回到订单页，显示此订单的完整信息，包括订单编号、交易时间、名称、口味、数量等信息。点餐系统订单列表页面如下图5-5所示。



图5-5 点餐系统订单列表页面

## 5.5 修改密码功能的实现

修改密码功能用到了qq的邮箱系统，在qq邮箱中设置完相应的发送邮件功能的配置后，还需要在代码yml配置文件中配置如下图5-6信息：

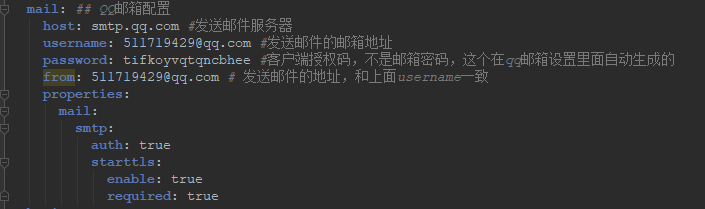


图5-6 点餐系统邮箱配置页面

当修改密码时发邮件给指定的账号，这一步是异步操作的，就是系统中修改密码页面会立即返回结果并跳转到消息成功页面，而发送邮箱跟返回结果不会有直接的关系，大概20秒后邮箱会收到邮件。这部分代码用到了Java8中的新特性Lambda表达式，stream流在书写上和观赏性上相比Java8之前有很大提高。异步主要的代码如下：

//异步

CompletableFuture.runAsync(()->{

iMailService.sendHtmlMail(email, "主题：修改密码",

"<a href='http://localhost:82/editPassword?userId=" + user.getUserId() + "'>跳转重置密码页面</a>");

);

在改密码的时候需要发邮件给原账号，主要流程是：在页面填入需要修改的账号，点击“发送邮件”，用户会在邮箱收到邮件，在邮件中点击链接时会直接跳转修改密码页面，当用户修改完密码后等5秒跳转回登录页面进行重新登录。当初选择邮箱系统的时候，之所以选择qq邮箱系统，是因为根据大数据分析到当代大学生用qq的比较多，所以用户注册的时候可能以qq账号为本系统的账号进行注册，因此选择了调用qq的邮箱系统接口。点餐系统收到修改密码邮件页面如下图5-7所示。

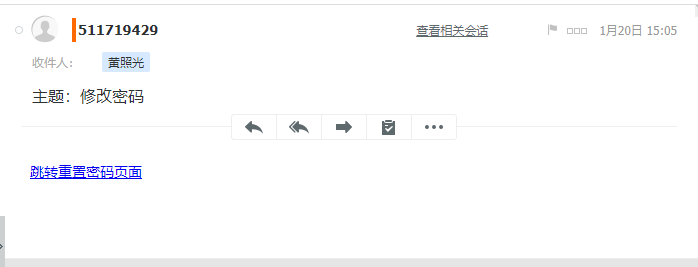


图5-7 点餐系统收到修改密码邮件页面

## 5.6 线上付款功能的实现

该系统为了模拟付款，调用了支付宝的沙箱接口来进行线上付款，沙箱环境非常接近真实的付款环境，这也是该系统选择支付宝沙箱的重要原因之一，而且操作也很简单。付款方式也采用当下最流行的扫码付款，既便捷又安全。

在Java代码中，通过new DefaultAlipayClient创建沙箱环境对象，在传的七个参数中第一个参数是沙箱环境的回调网址，第二个参数是自己在沙箱管理平台申请的唯一的识别id，第三个参数也是在沙箱管理平台申请的私钥，第四个和第五个参数是传入的格式，第六个参数是支付公共键，第七个参数的固定参数代表加密方式。主要代码如下：

//获得初始化的AlipayClient

AlipayClient alipayClient = new DefaultAlipayClient ("https://openapi.alipaydev. com/gateway.do", APP\_ID, PRIVATE\_KEY, "json", "utf-8", ALIPAY\_PUBLIC\_KEY, "RSA2");

//创建API对应的request

AlipayTradePagePayRequest alipayRequest = new AlipayTradePagePayRequest();

if (all == 1) {

//成功后回调

alipayRequest.setReturnUrl(ww + "/payok?orderSn=" + ordersn\_string + "&all=1"); alipayRequest.setNotifyUrl(ww + "/payok?orderSn=" + ordersn\_string + "&all=1");

}

alipayRequest.setBizContent("{\"out\_trade\_no\":\"" + ordersn[0] + "\","

+ "\"total\_amount\":\"" + money + "\","

+ "\"subject\":\"subject\","

+ "\"body\":\"body\","

+ "\"product\_code\":\"FAST\_INSTANT\_TRADE\_PAY\"}");//填充业务参数

String form = "";

try {

//调用SDK生成表单

form = alipayClient.pageExecute(alipayRequest).getBody();

} catch (AlipayApiException e) {

e.printStackTrace();

}

上面为支付宝沙箱环境提供的代码demo，填入自己账号的相关信息后即可直接使用，系统调用支付宝沙箱环境进行付款购买商品更加具有真实性和客观性。点餐系统付款页面如下图5-8所示。

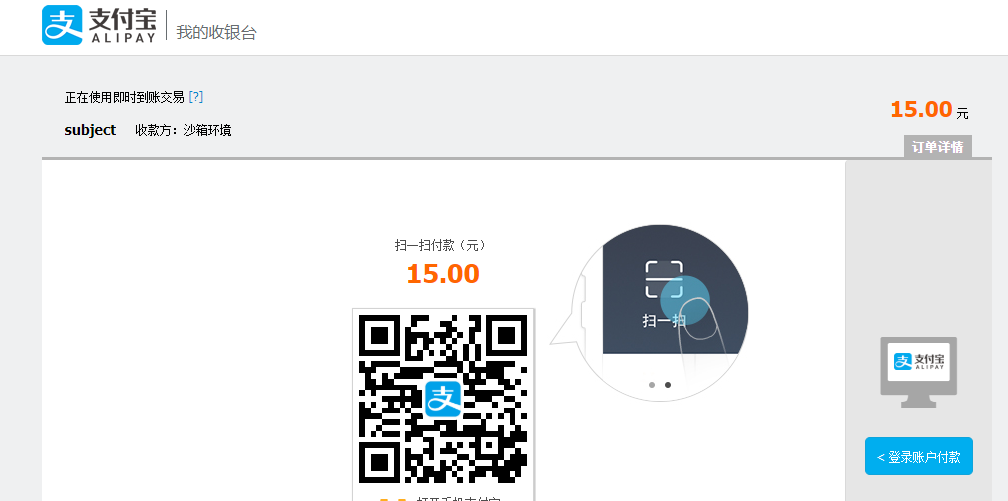


图5-8 点餐系统付款页面

# 6 系统测试

系统测试就是发现系统中存在的缺陷并将缺陷改正的过程，也是项目开发生命周期的最后一个步骤，是不能省略的一个重要步骤。系统测试主要分为功能测试和性能测试两大部分，其中功能测试是对系统中所有功能的一个基本测试，观察测试结果是否与需求设计符合；在互联网时代，并发访问是很重要的，性能测试主要是对接口的压力测试，观察功能接口能达到多大的并发访问、多高的吞吐量。

## 6.1 功能测试

在功能测试中，申请了多个商家和顾客账号用来测试每一个接口，主要是对登录、注册、商品详情、订单、购物车、付款、修改密码发邮件等接口做重点测试，测试结果如下表6-1所示，跟预期的结果大致相同。

表6-1 测试用例

|  |  |  |  |
| --- | --- | --- | --- |
| 测试模块 | 测试功能 | 测试详情 | 测试结果 |
| 公共模块 | 登录 | 账号：admin@qq.com  密码：admin | 登陆成功，跳转首页 |
| 账号：511719429@qq.com  密码：123456 | 登陆成功，跳转首页 |
| 注册 | 账号：511719429@qq.com  密码：123456 | 注册成功，跳转登录页面 |
| 修改密码 | 账号：511719429@qq.com  原密码：123456  新密码：dian123!@# | 显示修改成功后，跳转登录页面 |
| 修改个人信息 | 修改部分信息 | 修改成功后跳转首页 |
| 退出登录 | 点击退出按钮 | 退出成功后跳转首页 |
| 搜索商品 | 在首页按分类搜索和关键字模糊搜索 | 搜索返回结果正确 |
| 订单管理 | 更改订单的状态（包括商家更改和用户更改） | 更新成功 |
| 商家模块 | 添加商品 | 添加新的商品（包括上传图片） | 添加成功 |
| 修改商品 | 对商品信息进行修改 | 修改成功 |
| 删除商品 | 选择商品并点击删除 | 删除成功 |
| 用户模块 | 购物车 | 添加商品到购物车 | 添加成功 |
| 删除已选择的商品 | 删除成功 |
| 下订单 | 从购物车结算 | 下单成功 |
| 支付 | 扫码支付任意金额 | 支付成功，跳转订单列表页面 |

## 6.2 性能测试

在性能测试中，该系统采用jmeter对一些重要的接口进行压力测试，下面着重介绍一下登录接口的压力测试。

对登录进行压力测试时，在测试计划中加入线程组，加入循环控制器，加入取样器，加入http请求，在http请求中设置一个简单的http登录，它会自动获取cookie信息，然后把post请求改为跟随重定向接着填写http请求头和账号密码等信息，另外还需要在项目中修改配置文件jmeter.properties，设置CookieManager.save.cookies=true，然后点击运行就可以查看结果树了[16]。在本次结果中，如下图6-1所示，可以看到error失败率为0%，对登录接口代码做了一些调整后，平均响应时间降低了，并且吞吐量也有提高。为了做并发测试，添加了很多个线程数，并设置结果树、图形结果、聚合报告，把间隔时间设置为0，可以看作同时访问，也就是并发请求。

Jmeter还可以通过设置断言来验证结果，用来做回归测试很方便，本次系统就不做延伸了。

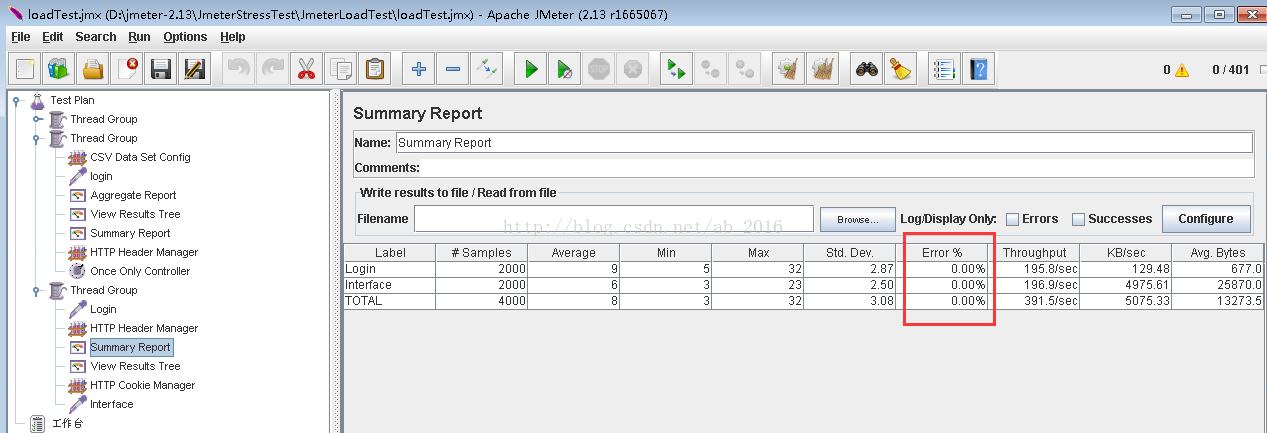


图6-1 点餐系统登录压测页面

# 总结

经过一段时间不懈努力和深入钻研，终于将点餐系统设计并开发完成。该系统面向广大有意愿将互联网与店铺营销模式进行结合的商家。该系统可以将商家的商品以互联网页面的形式排列展现给用户，告别传统的菜单，使本店特色商品、最新商品和热销商品以醒目的方式推荐给顾客。顾客还可以通过进入系统迅速定位到商家推荐商品，并选择自己喜爱的商品进行下单，支付的时候不只是柜台支付现金，也可以使用线上支付的方式进行消费，在顾客本次消费后可以对商家的商品、服务进行评价与打分，使顾客可以更好的与商家进行交互，便于商家根据用户的意见进行菜品与管理的调整。而且该系统的功能经过测试证明已经可以满足大部分想改变传统销售模式的商家的需求，具有一定的实际意义。系统也存在着少量的缺陷，如点餐环节、线上支付、配送状态还需要再做完善；另外redis缓存时数据一致性还需要更全面的考虑，遇到高并发的时候会出现脏读问题，是先删缓存还是先更新数据库这个还有待商榷，准备在下一版中通过加锁阻塞改进这些功能，会根据实际功能来优化代码以达到预期的效果。

# 参考文献

1. 周兴宇.微信自助点餐系统[D].辽宁:大连海事大学,2016.
2. 黄文.基于微信的叫号点餐系统的设计与实现[D].湖北:华中科技大学,2014.
3. 温立辉.Spring框架在模型层的应用及原理[J].福建电脑,2017,6(1):165-166.
4. 王旭.软件开发中JAVA编程的应用[J].中国管理信息化,2018,5(2):138-139.
5. 杨振华，杨明红.数据库编程中Java设计模式的应用研究[J].电脑迷,2017,10(1):245-246.
6. 张桂珠,陈爱国.Java面向对象程序设计(第2版)[M].北京:北京邮电大学出版社,2018.
7. 张光平.Java多线程的分析[J].科技风,2018,12(6):188.
8. 王咏佳.校园订餐系统的设计与实现[D].江西:江西财经大学,2017.
9. 张磊.网上订餐系统的设计与实现[D].吉林:吉林大学,2016.
10. 王东亮.网上点餐系统中购物车的设计与实现[J].电子技术与软件工程,2015,5(10):228.
11. 林子雨,赖永炫.关系数据库中的关键词查询结果动态优化[J].软件学报,2014,7(6):120.
12. Ramakrishnan Jagdish.A dynamic programming approach to adaptive fractionation[J].Physics in Medicine and Biology[J].Physics in Medicine and Biology,2016,57(5):29-33.
13. Rachit Mohan Garg,YaminiSood,Balaji Kottana.AFramework Based Approachfor the Development of Web Based Applications Waknaghat[J].Jaypee University of InformationTechnology,2015(4),p5.
14. 冯颖凌,廖子豪,严豪.基于Java的网上商城开发[J].电脑知识与技术,2016,12(24):123-124.
15. 刘睿文,吴帝李.Java的B2C电子商城网站系统设计[J].电脑编程技巧,2018(22):90-92.
16. 沈黄河.电子商务平台网上商城系统的设计与实现[J].电脑与电信,2017(22):50-51.

# 致谢

该论文是在指导老师吴冉的耐心教导和帮助下完成的。她教学态度严谨，对我在论文的写作中提供了很大的帮助，严格要求我的论文质量，力争让我做到最好。在我有疑问的时候她都会第一时间给我解答，并给我正确的指引。在论文完成的过程中，老师给了我很多指导意见，细心地在论文中写了大量的批注告诉我应该怎么改，在我遇到困难不知该怎么写的时候，老师的意见总会让我恍然大悟。还要谢谢我的学长，论文中出现的一些技术问题在询问学长后也得以解决。此时毕业论文已经接近尾声，在此我向吴冉老师表示深深的感激！

四年的大学生涯仿佛就在昨日，半年的实习经验让我学到了很多东西，也认识到在学校期间老师给我们的指导是多么的重要，让我们出校的瞬间就站在了比同龄人更高的起点上。我要感谢在这四年中给予我极大支持和帮助的老师和同学们以及父母。因为有了父母的支持，我才能有四年的时间来安心的学习，因为有老师和同学的帮助我才能在四年里学到更多的知识和技能，谢谢你们在生活和学习中对我的帮助和包容。是你们让我树立信心，技术上攻克一个又一个的难题，希望你们在以后的日子里生活和和美美。最后，请让我再次表达对你们的感谢，谢谢你们的帮助！

# 附录

项目源代码网址：https://github.com/hzg511719429/bishe

附录1：

登录模块的controller代码：

@GetMapping(value = "/login")

public String loginView() {

return "login";

}

@PostMapping(value = "/login")

public String loginForm(User user, Model model, HttpServletRequest request) {

Subject subject = SecurityUtils.getSubject();

UsernamePasswordToken token = new UsernamePasswordToken(user.getUserEmail(), MD5Util.MD5(user.getUserPassword()));

try {

subject.login(token);

user.setUserPassword(MD5Util.MD5(user.getUserPassword()));

List<User> userList = loginService.getUser(user);

if (userList.size() <= 0) {

model.addAttribute("message", "用户名或密码不正确！");

return "login";

} else {

request.getSession().setAttribute("user", userList.get(0));

return "redirect:index";

}

} catch (Exception e) {

model.addAttribute("message", "用户名或密码错误");

return "login";

}

}

登录模块的service代码：

@Override

public List<User> getUser(User user) {

//根据登录用户名，即用户邮箱和密码来查询用户，确认输入的账户和密码是否正确

UserExample userExample = new UserExample();

UserExample.Criteria criteria1 = userExample.createCriteria();

criteria1.andUserEmailEqualTo(user.getUserEmail());

criteria1.andUserPasswordEqualTo(user.getUserPassword());

//执行查询

List<User> userList = userMapper.selectByExample(userExample);

return userList;

}

登录模块的shiro配置代码：

public class UserRealm extends AuthorizingRealm {

@Autowired

private LoginService loginService;

@Override

protected AuthorizationInfo doGetAuthorizationInfo(PrincipalCollection principalCollection) {

return null;

}

@Override

protected AuthenticationInfo doGetAuthenticationInfo(AuthenticationToken authenticationToken) throws AuthenticationException {

String email = (String) authenticationToken.getPrincipal();

User user = loginService.findUserByEmail(email);

String psw = user.getUserPassword();

SimpleAuthenticationInfo info = new SimpleAuthenticationInfo(email, psw, this.getName());

return info;

}

}

附录2：

首页模块的controller代码：

@Controller

public class PageController {

@Autowired

private PageService pageService;

/\*\*

\* 主页

\*

\* @param page 当前页数，默认是1

\* @param row 一页有几个数据，默认10个

\* @param status 商品的状态：0默认、1最新、2销量，默认是0

\* @param min 输入的最低价格，默认为0

\* @param max 输入的最高价格，默认为100000

\* @param word 输入要查询的关键字，默认为空

\* @return

\*/

@RequestMapping("/")

public String index(@RequestParam(defaultValue = "1") Integer page,

@RequestParam(defaultValue = "10") Integer row,

@RequestParam(defaultValue = "0") Integer status,

@RequestParam(defaultValue = "0.00") double min,

@RequestParam(defaultValue = "100000.00") double max,

@RequestParam(defaultValue = "") String word, Model model) {

//下面是查询商品列表

//返回PageResult，其中object属性是 list集合

getIndex(page, row, status, min, max, word, model);

return "/index";

}

//转到主页的代码

private void getIndex(Integer page, Integer row, Integer status, double min,

double max, String word, Model model) {

PageResult pageResult = pageService.getProductList(page, row, status, min, max, word);

model.addAttribute("pageResult", pageResult);

model.addAttribute("status", status);

model.addAttribute("max", max);

model.addAttribute("min", min);

}

}

首页模块的service代码：

@Service

public class PageServiceImpl implements PageService {

@Autowired

private ProductMapper productMapper;

@Autowired

private ProductimgMapper productimgMapper;

@Override

public PageResult getProductList(int page, int row, int status, double min, double max, String word) {

//根据输入条件，进行组合查询

ProductExample example1 = new ProductExample();

ProductExample.Criteria criteria1 = example1.createCriteria();

//确定最高价和最低价

criteria1.andProductPriceBetween(min, max);

//上架的才显示 不上架的不显示

criteria1.andProductIsSellEqualTo("1");

//判断商品的状态

if (status != 0) {

if (status == 1) {

//按照最新查询

example1.setOrderByClause("product\_id DESC");

} else {

//按照卖出最多查询

example1.setOrderByClause("product\_hass\_selled DESC");

}

}

//根据搜索的关键词在进行查询

if (word != null) {

criteria1.andProductNameLike("%" + word + "%");

}

//执行查询

PageHelper.startPage(page, row);

List<Product> productList = productMapper.selectByExample(example1);

PageInfo pageInfo = new PageInfo(productList);

//总页数

int totalPages = pageInfo.getPages();

for (Product product : productList) {

ProductimgExample example2 = new ProductimgExample();

ProductimgExample.Criteria criteria2 = example2.createCriteria();

criteria2.andProductIdEqualTo(product.getProductId());

List<Productimg> productImgList = productimgMapper.selectByExample(example2);

if (productImgList.size() == 0) {

product.setProductPicture("static/img/defult.jpg");

} else {

//因为可能上传多张图片，这里取一张显示

product.setProductPicture(productImgList.get(0).getImgurl());

}

}

return PageResult.buid(page, productList, totalPages);

}

}

附录3：

商品管理模块的controller代码：

@Controller

public class SellProductsController {

@Autowired

private SellProductsService sellProductsService;

@Autowired

private ProductService productService;

/\*\*

\* 卖家商品列表展示

\*

\* @param model

\* @param page

\* @param row

\* @return

\*/

@RequestMapping("/adminMenu")

public String sellProducts(HttpServletRequest request, Model model,

@RequestParam(defaultValue = "1") Integer page,

@RequestParam(defaultValue = "10") Integer row) {

PageResult pageResult = sellProductsService.getUserAllProducts(request, page, row);

model.addAttribute("pageResult", pageResult);

model.addAttribute("objList", pageResult.getObjects().size());

return "sell/sellProducts";

}

/\*\*

\* 上传图片

\*

\* @param file

\* @param count

\* @param request

\* @return

\*/

@RequestMapping("/uploadImg")

@ResponseBody

public ShopResult upload(MultipartFile file, int count, HttpServletRequest request) {

if (file == null) {

return ShopResult.build(500, "请选择上传的图片");

}

try {

if (count == 0) {

request.getSession().setAttribute("uploadImgCount", null);

} else if (count > 5) {

return ShopResult.build(500, "上传图片不能超过6张");

}

String basePath = request.getSession().getServletContext().getRealPath("/");

String contentType = file.getContentType();

//如果是文件保存在/data/files 图片保存在/data/images

String path = contentType.contains("image") ? "data/images/" + DateUtils.format(new Date(), "YYYYMMdd") : "data/files/" + DateUtils.format(new Date(), "YYYYMMdd");

String fileName = System.currentTimeMillis() + ".jpg";

//新的图片

File newFile = new File(basePath + path);

if (!newFile.exists()) {

newFile.mkdirs();

}

//把内存图片写入磁盘中

File imgFile = new File(basePath + path, fileName);

file.transferTo(imgFile);

Map<Integer, String> list = (Map<Integer, String>) request.getSession().getAttribute("uploadImgCount");

if (list == null) {

list = new HashMap<>();

}

list.put(count, path + "/" + fileName);

request.getSession().setAttribute("uploadImgCount", list);

return ShopResult.build(200, count + "", path + "/" + fileName);

} catch (Exception e) {

e.printStackTrace();

return ShopResult.build(500, "上传失败");

}

}

/\*\*

\* 修改商品状态

\*

\* @param request

\* @param id

\* @param status

\* @param model

\* @return

\*/

@RequestMapping("/updateSellStatus")

public String updateSellStatus(HttpServletRequest request, Long id, String status, Model model) {

sellProductsService.updateProductStatus(request, id, status);

PageResult pageResult = sellProductsService.getUserAllProducts(request, 1, 10);

model.addAttribute("pageResult", pageResult);

return "/sell/sellProducts";

}

/\*\*

\* 修改商品

\*

\* @param request

\* @param product

\* @return

\*/

@RequestMapping(value = "/updateProduct", method = RequestMethod.POST)

public String updateGoods(HttpServletRequest request, Product product) {

sellProductsService.updateProduct(request, product);

return "redirect:adminMenu";

}

商品管理的service代码：

@Service

public class SellProcudtsServiceImpl implements SellProductsService {

@Override

public PageResult getUserAllProducts(HttpServletRequest request, int page, int row) {

//如果该用户已登录，从session中获取该user，根据userId查询该用户下的所有商品

User user = (User) request.getSession().getAttribute("user");

ProductExample productExample = new ProductExample();

ProductExample.Criteria criteria1 = productExample.createCriteria();

criteria1.andProductSellerIdEqualTo(user.getUserId());

//执行查询前，进行分页操作

PageHelper.startPage(page, row);

List<Product> productList = productMapper.selectByExample(productExample);

PageInfo pageInfo = new PageInfo(productList);

int totalPages = pageInfo.getPages();

return PageResult.buid(page, productList, totalPages);

}

@Override

public void updateProductStatus(HttpServletRequest request, long id, String status) {

//根据id获取商品，然后更新状态

Product product = productMapper.selectByPrimaryKey(id);

product.setProductIsSell(status);

productMapper.updateByPrimaryKey(product);

}

@Override

public ShopResult updatelittleProducts(HttpServletRequest request, int[] ids, int status) {

//循环数组，改变商品状态

for (int i = 0; i < ids.length; i++) {

//判断status状态

//状态2是删除操作

if (status == 2) {

//删除商品前要先删除商品图片

ProductimgExample example = new ProductimgExample();

ProductimgExample.Criteria criteria = example.createCriteria();

criteria.andProductIdEqualTo((long) ids[i]);

productimgMapper.deleteByExample(example);

//还要删除购物车对应的商品

CarExample example1 = new CarExample();

CarExample.Criteria criteria1 = example1.createCriteria();

criteria1.andCarProductIdEqualTo((long) ids[i]);

carMapper.deleteByExample(example1);

productMapper.deleteByPrimaryKey((long) ids[i]);

} else {

//此状态是改变上架或不上架。

Product product = productMapper.selectByPrimaryKey((long) ids[i]);

product.setProductIsSell(status + "");

productMapper.updateByPrimaryKey(product);

}

}

return ShopResult.build(200, "修改成功", request.getContextPath() + "/adminMenu");

}

@Override

public ShopResult insertProduct(HttpServletRequest request, Product product) {

User user = (User) request.getSession().getAttribute("user");

product.setProductSellerId(user.getUserId());

product.setProductNo(IDUtils.genItemId() + "");

product.setProductHassSelled(0);

if (product.getProductDescription() == null) {

product.setProductDescription("商家未进行介绍");

}

productMapper.insert(product);

ProductExample example = new ProductExample();

ProductExample.Criteria criteria = example.createCriteria();

criteria.andProductNoEqualTo(product.getProductNo());

List<Product> productList = productMapper.selectByExample(example);

//商品图片添加，先获取路径

Map<Integer, String> list = (Map<Integer, String>) request.getSession().getAttribute("uploadImgCount");

for (Map.Entry<Integer, String> entry : list.entrySet()) {

Productimg productimg = new Productimg();

productimg.setProductId(productList.get(0).getProductId());

productimg.setImgurl(entry.getValue());

productimgMapper.insert(productimg);

}

return ShopResult.build(200, "添加成功", request.getContextPath() + "/adminMenu");

}

}