

毕 业 设 计

****

**题目： 在线订餐系统的设计与实现**

**学 生：** 王宗玉\_\_\_\_\_\_\_\_\_

**学 号：** 20140606012\_\_\_\_\_\_

**学 院：\_\_\_**电气与信息工程学院\_\_

**专 业：\_\_\_\_**计算机科学与技术\_\_\_

**指导教师：\_\_\_\_\_\_\_\_\_\_**杨云\_\_\_\_\_\_\_\_\_

2018 **年** 6 **月** 17 **日**

在线订餐系统的设计与实现

# 摘 要

随着社会生活水平的不断提高，人们生活的节奏加快，传统的餐饮服务已经不能满足用户的需求，网上订餐的方式受到了用户的青睐。通过在线订餐系统，为餐厅和用户搭建起一个快捷的网站服务平台，同时也为用户提供更准确更便捷的餐饮信息的平台。本系统基于B/S模式，选择面向对象的、易于扩展的Java作为编程语言，后台使用SSM三大框架和Servlet整合的形式，以MVC作为设计理念，使用MySQL数据库作为后台数据支持，通过Apache Tomcat7.0服务器运行系统。本系统主要实现了企业菜品信息的录入，菜品信息的发布，用户浏览菜品信息和下单等操作，以及评价和反馈等功能。

关键词：毕业生，SSM框架，B/S模式，订餐系统

# Design and Implementation of Online Ordering System ABSTRACT

With the continuous improvement of the living standards of society, the rhythm of people’s lives has accelerated, and traditional catering services have been unable to meet the needs of users. The way of ordering meals online has been favored by users. Through the online ordering system, a fast website service platform is set up for restaurants and users. At the same time, it also provides users with a more accurate and convenient platform for catering information. This system is based on the B/S model, and chooses object-oriented and easy-to-expand Java as the programming language. It uses the three major frameworks of SSM and Servlet integration in the background, uses MVC as the design concept, and uses MySQL database as background data support through Apache Tomcat7. .0 Server running system. This system mainly realizes the input of the company's dish information, the release of dish information, the user's operation of browsing dish information and orders, and the functions of evaluation and feedback.

**Key words：**Graduates, SSM Framework, B/S Model, Ordering system

目 录

[摘 要 II](#_Toc515114340)

[Design and Implementation of Online Ordering System ABSTRACT III](#_Toc515114341)

[1 绪论 1](#_Toc515114342)

[1.1 课题研究背景及意义 1](#_Toc515114343)

[1.2 国内外发展状况 1](#_Toc515114344)

[1.3 本文研究内容 1](#_Toc515114345)

[1.3.1 web前端语言 1](#_Toc515114346)

[1.3.2 Java Web整合开发 2](#_Toc515114347)

[1.3.3 数据库相关研究 2](#_Toc515114348)

[2 基础理论及相关技术 2](#_Toc515114349)

[2.1 SSM框架 2](#_Toc515114350)

[2.2 Servlet和JSP 3](#_Toc515114351)

[2.3 MySQL数据库 4](#_Toc515114352)

[2.4 JDK和Tomcat服务器 5](#_Toc515114353)

[2.5 IntelliJ IDEA 5](#_Toc515114354)

[3 系统分析及设计 6](#_Toc515114355)

[3.1 可行性分析 6](#_Toc515114356)

[3.2 需求分析 6](#_Toc515114357)

[3.3 系统总体设计 6](#_Toc515114358)

[3.3.2 系统功能结构 7](#_Toc515114359)

[3.4 数据库设计与实现 7](#_Toc515114360)

[3.4.1 数据库概念设计 7](#_Toc515114361)

[3.4.2 数据库表设计 11](#_Toc515114362)

[3.4.3 数据库逻辑结构设计 16](#_Toc515114363)

[4 系统详细设计 17](#_Toc515114364)

[4.1 系统环境配置 17](#_Toc515114365)

[4.1.1 软件环境 17](#_Toc515114366)

[4.1.2 硬件环境 17](#_Toc515114367)

[4.2 系统框架实现 17](#_Toc515114368)

[4.3 系统公共模块设计 19](#_Toc515114369)

[4.3.1 数据库连接和事务管理器 19](#_Toc515114370)

[4.3.2 系统安全控制 21](#_Toc515114371)

[4.4系统主要模块功能的实现 21](#_Toc515114372)

[4.4.1登录注册功能 21](#_Toc515114373)

[4.4.2 商品展示功能 24](#_Toc515114374)

[4.4.3 购物车的设计与实现 28](#_Toc515114375)

[4.4.4 用户订餐的设计与实现 31](#_Toc515114376)

[4.4.5 订单操作的设计与实现 32](#_Toc515114377)

[4.4.6用户评论的设计与实现 33](#_Toc515114378)

[4.4.7管理员商品管理的设计与实现 35](#_Toc515114379)

[4.4.8管理员订单状态管理的设计与实现 38](#_Toc515114380)

[5 系统测试与性能分析 40](#_Toc515114381)

[5.1 测试目的 40](#_Toc515114382)

[5.2 测试方法 40](#_Toc515114383)

[5.2.1 模块测试 40](#_Toc515114384)

[5.2.2 子系统测试 40](#_Toc515114385)

[5.2.3 系统测试 40](#_Toc515114386)

[5.3 功能测试 40](#_Toc515114387)

[5.3.1 登录测试 40](#_Toc515114388)

[5.3.2 注册测试 41](#_Toc515114389)

[6 总 结 43](#_Toc515114390)

[致 谢 44](#_Toc515114391)

[参 考 文 献 45](#_Toc515114392)

# 

# 1 绪论

## 1.1 课题研究背景及意义

出门在外，很少会有人自己亲自做顿饭吃，更多的是要么吃食堂，要么去饭店，去食堂免不了排队，食堂吃上一些天就顿觉食不下咽。去饭店，跑到附近的饭店发现人很多，排队变成了坐等。随着网络逐步的渗透到我们的生活中，网上购物已成为人们消费的一种方式。随之兴起的网上订餐可以使人们借助于互联网进行饭菜的选购，餐馆配有专门的人员负责送餐，这就减轻了人们的去食堂去餐馆吃饭所占据的等待吃饭时间，以及受天气等因素的影响造成出行的不便。更重要的是餐馆的经营模式变得更加的灵活，受到地域，天气等因素的影响明显减轻了，餐馆的消费人群明显的增多了。网上订餐系统一方面使得消费者多了一种选择饭菜的方式，一方面拓展的餐馆的销售途径。

## 1.2 国内外发展状况

目前，中国餐饮O2O主要模式可以分为四大种类：团购类，以美团、百度糯米、大众点评为代表，引进美国Groupon模式，全面接入本地餐户，团购起步较早，是目前餐饮O2O的主要模式；点评类，以大众点评为代表，通过长期的积累，汇集了大量餐厅和用户的信息，起到了市场培育的作用；订餐类，以订餐小秘书为代表，帮助用户通过互联网渠道预定就餐作为、甚至餐品，提升用餐质量；外卖类，以饿了么、美团外卖外代表的第三方外卖平台是餐饮O2O领域近期最受资本青睐的O2O模式，千万级美元融资频频发生，并且众多传统餐饮和互联网巨头和纷纷布局，市场在近年来快速发展。   目前中国餐饮O2O主要模式及代表商家

   根据中国在线外卖市场现状调研分析及发展趋势报告认为，现在用户选择在线外卖平台时会考虑平台的食品安全保障，还有平台的送餐速度，还有平台的优惠活动。目前

 餐饮外卖市场保持稳定增长态势，并且目前而言餐饮外卖的互联网渗透率仍然较低，随着送餐物流的不断完善、技术进步、城市扩展等因素驱动，预计互联网餐饮外卖市场在未来5年内仍将维持高速增长的态势。

## 1.3 本文研究内容

在线订餐系统的开发主要包括前端应用程序的开发以及后台数据库的建立和维护两个方面。对于前者则要求具有应用程序功能完备，易使用等特点。而对于后者要求建立起数据一致性和完整性强、数据安全性好的数据库。

### 1.3.1 web前端语言

一个功能完备的网站，网站展示页面的美观简洁非常重要，前端仅是向用户表达所要展示的信息，所以可以使用H5和CSS等来设计页面，用Javascript和JQuery增加页面特效，用ajax实现异步刷新，来增加用户体验，当然在设计时需要注意到浏览器的兼容性问题。

### 1.3.2 Java Web整合开发

网站除了给用户展示出用户所需要的内容之外，用户与页面之间的交互就显得更重要，即浏览器和后台服务器的请求，使用SSM框架来完成后台的制作[2]。其中SpringMVC完成页面之间的跳转、输入校验、文件上传以及使用jstl标签完成页面显示，Mybatis用来完成数据库中数据的持久化，Spring用来接管整个项目，创建对象并维护对象间的关系。

### 1.3.3 数据库相关研究

数据库的选择由于此系统是所有用户使用的网站系统，所以对数据的安全性和完整性要求较高，可以选择MySQL5.7数据库,结合Mybatis框架完成数据操作，可以得到较快的速度，用户的体验更好。

本系统一共分了两个系统来分别实现其功能。连个系统分别为：用户系统、后台系统。本系统设计完成后，将其安装在Tomcat服务器上，用户可以直接使用浏览器进行查询和注册，后台数据库使用MySQL，用户系统使用JSP动态网页，再辅以少量的JavaScript脚本用以提高用户的体验。后台系统和用户系统使用同一套数据库。

# 2 基础理论及相关技术

## 2.1 SSM框架

SSM是 SpringMVC+Spring+Mybatis的一个集成[框架](http://baike.baidu.com/item/%E6%A1%86%E6%9E%B6)，是现下使用较为广泛的一种Web应用程序开源框架。

集成SSM框架的系统从职责上分为四层：[表示层](http://baike.baidu.com/item/%E8%A1%A8%E7%A4%BA%E5%B1%82)、控制层、[业务逻辑层](http://baike.baidu.com/item/%E4%B8%9A%E5%8A%A1%E9%80%BB%E8%BE%91%E5%B1%82)、[数据持久层](http://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E6%8C%81%E4%B9%85%E5%B1%82)，以帮助开发人员在短期内搭建结构清晰、可复用性好、[维护](http://baike.baidu.com/item/%E7%BB%B4%E6%8A%A4)方便的[Web应用程序](http://baike.baidu.com/item/Web%E5%BA%94%E7%94%A8%E7%A8%8B%E5%BA%8F)[2]。其中使用Spring作为系统的整体基础架构，管理SpringMVC和Mybatis。SpringMVC

负责MVC的分离，控制用户与服务器之间的交互，产生的数据使用Mybatis持久化到数据库中，也是通过Mybatis来查询用户所需要的数据，在业务逻辑层控制显示的内容。 具体做法是：用[面向对象](http://baike.baidu.com/item/%E9%9D%A2%E5%90%91%E5%AF%B9%E8%B1%A1)的分析方法根据需求提取出数据模型，每个数据模型对应数据库中的一张表和一个Java模型类，然后给出DAO接口，调用DAO接口中的查询函数，Mybatis通过接口函数名映射到相应XML文件中具体的SQL语句，在运行时组装成接口的实现，然后通过JDBC查询到需要的数据，Mybatis的好处是具体的SQL语句是在XML中配置的，当业务变化时只需要重写SQL、重新加载即可， Mybatis在与数据库的交互时，可以根据数据库表反向生成对应的Java模型类，减少工作量，只要环境搭好，建立好数据库后，通过配置文件配置模型类和表之间的对应关系，运行项目后对数据库的某张表的操作统统转化为对模型类的操作、十分便于数据库的操作，并且将SQL语句放入XML配置文件中又异常灵活，便于修改。

系统的基本[业务](http://baike.baidu.com/item/%E4%B8%9A%E5%8A%A1)[流程](http://baike.baidu.com/item/%E6%B5%81%E7%A8%8B/31013)是：在[表示层](http://baike.baidu.com/item/%E8%A1%A8%E7%A4%BA%E5%B1%82)中，首先通过JSP页面实现[交互界面](http://baike.baidu.com/item/%E4%BA%A4%E4%BA%92%E7%95%8C%E9%9D%A2)，负责接收请求(Request)和传送响应(Response)，然后SpringMVC通过dispatchServlet将接受到的Request分派给注解配置的路径（requestMapping）的Controller层函数中进行处理。在业务层中，管理服务组件的Spring IoC容器负责向业务提供模型(Model)组件和该组件的协作对象数据处理(DAO)组件完成业务逻辑，并提供事务处理、缓冲池等容器组件以提升系统性能和保证数据的完整性，并且将业务处理结果返回给Controller层，再通过DispatchServlet返回到相应的页面。而在数据[持久层](http://baike.baidu.com/item/%E6%8C%81%E4%B9%85%E5%B1%82)中，则依赖于Mybatis的对象化映射和数据库交互，处理DAO组件请求的数据，并返回处理结果。

采用上述开发模型，不仅实现了视图、控制器与模型的彻底分离，而且还实现了[业务逻辑层](http://baike.baidu.com/item/%E4%B8%9A%E5%8A%A1%E9%80%BB%E8%BE%91%E5%B1%82)与持久层的分离。这样无论前端如何变化，模型层只需很少的改动，并且数据库的变化也不会对前端有所影响，大大提高了系统的可复用性。而且由于不同层之间[耦合度](http://baike.baidu.com/item/%E8%80%A6%E5%90%88%E5%BA%A6)小，有利于团队成员并行工作，大大提高了开发[效率](http://baike.baidu.com/item/%E6%95%88%E7%8E%87)。

## 2.2 Servlet和JSP

Servlet 为创建基于 web 的应用程序提供了基于组件、独立于平台的方法，可以不受 CGI 程序的性能限制。其主要功能在于[交互式](http://baike.soso.com/ShowTitle.e?sp=S%E4%BA%A4%E4%BA%92%E5%BC%8F)地浏览和修改数据，生成动态Web内容。Servlet运行于支持Java的应用服务器中。从实现上讲，Servlet可以响应任何类型的请求，但绝大多数情况下Servlet只用来扩展基于[HTTP协议](http://baike.soso.com/ShowTitle.e?sp=SHTTP%E5%8D%8F%E8%AE%AE)的Web服务器。

JSP（全称JavaServer Pages）是由Sun Microsystems公司倡导和许多公司参与共同创建的一种使软件开发者可以响应客户端请求，而动态生成HTML、XML或其他格式文档的Web网页的技术标准。JSP技术是以Java语言作为脚本语言的，JSP网页为整个服务器端的Java库单元提供了一个接口来服务于HTTP的应用程序。用JSP开发的Web应用是跨平台的，既能在Linux下运行，也能在其他操作系统上运行。

JSP相比ASP和PHP，具有的优势：

（a）一次编写，到处运行。在这一点上Java比[PHP](http://baike.baidu.com/view/99.htm)更出色，除了系统之外，代码不用做任何更改。

（b）系统的多平台支持。基本上可以在所有平台上的任意环境中开发，在任意环境中进行系统部署，在任意环境中扩展。相比[ASP](http://baike.baidu.com/view/2616.htm)/PHP的局限性是显而易见的。

（c）强大的可伸缩性。从只有一个小的[Jar](http://baike.baidu.com/view/50457.htm)文件就可以运行Servlet/JSP，到由多台服务器进行集群和负载均衡，到多台Application进行事务处理，消息处理，一台服务器到无数台服务器，Java显示了一个巨大的生命力。

（d）多样化和功能强大的开发工具支持。这一点与ASP很像，Java已经有了许多非常优秀的开发工具，而且许多可以免费得到，并且其中许多已经可以顺利的运行于多种平台之下。

## 2.3 MySQL数据库

MySQL是一个[关系型数据库管理系统](http://baike.baidu.com/item/%E5%85%B3%E7%B3%BB%E5%9E%8B%E6%95%B0%E6%8D%AE%E5%BA%93%E7%AE%A1%E7%90%86%E7%B3%BB%E7%BB%9F)，由瑞典公司开发，目前属于 [Oracle](http://baike.baidu.com/item/Oracle) 旗下产品。MySQL是最流行的[关系型数据库管理系统](http://baike.baidu.com/item/%E5%85%B3%E7%B3%BB%E5%9E%8B%E6%95%B0%E6%8D%AE%E5%BA%93%E7%AE%A1%E7%90%86%E7%B3%BB%E7%BB%9F)之一，在 WEB 应用方面，MySQL是最好的RDBMS (Relational Database Management System，关系数据库管理系统) 应用软件。

MySQL是一种关系数据库管理系统，关系数据库将数据保存在不同的表中，而不是将所有数据放在一个大仓库内，这样就增加了速度并提高了灵活性。

MySQL所使用的 SQL 语言是用于访问[数据库](http://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%BA%93)的最常用标准化语言。MySQL 软件采用了双授权政策，分为社区版和商业版，由于其体积小、速度快、总体拥有成本低，尤其是[开放源码](http://baike.baidu.com/item/%E5%BC%80%E6%94%BE%E6%BA%90%E7%A0%81)这一特点，一般中小型网站的开发都选择 MySQL 作为网站数据库。

[MySQL](http://baike.baidu.com/item/MySQL/471251)是一种[开放源代码](http://baike.baidu.com/item/%E5%BC%80%E6%94%BE%E6%BA%90%E4%BB%A3%E7%A0%81)的关系型[数据库管理](http://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%BA%93%E7%AE%A1%E7%90%86)系统（RDBMS）MySQL具有性价比高、灵活、广为使用和具有良好支持的特点。

（a）普及性

MySQL是一个可靠的数据库系统，无论是在嵌入式或大型群集系统的部署中，还是在基于Web的应用程序领域，用户时常会发现其实自己并不是第一个选用MySQL数据库的先驱者。

（b）简单性

与其他数据库相比，MySQL易学易用。对于MySQL数据库，无论是在开发方面，还是支持方面，现在有大量强大的工具可以选择。每一个新手开发者可以轻松地使用MySQL数据库进行开发。甚至一个有经验的Windows管理者也可以轻松部署并开始学习它，而你不需投入一分钱来了解这个数据库。MySQL可以运行的更快速，MySQL中没有多余的功能来拖累CPU或占用内存。

（c）低成本

MySQL是一个开源的数据库，MySQL对硬件的较低要求是其最大的优势之一，MySQL易学、易部署、易管理和易维护。对于现有业务，可以轻松移植到MySQL。MySQL部署迅速，因此移植过程不会导致生产中断。而且，较短的学习曲线可以让你的系统管理员迅速掌握它的运行和维护。

（d）灵活性和可扩展性

由于在MySQL中有如此众多的额外功能可选，诸如存储引擎等，这个特点使得MySQL可以根据你当前的系统的需要来进行调整。

## 2.4 JDK和Tomcat服务器

Tomcat [服务器](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=3850180&ss_c=ssc.citiao.link)是一个免费的[开放源代码](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=269184&ss_c=ssc.citiao.link)的Web [应用服务器](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=4341240&ss_c=ssc.citiao.link)，是Apache软件基金会（Apache Software Foundation）的Jakarta项目中的一个核心项目，由Apache、Sun和其他一些公司及个人共同开发而成,是为了使Servlet/JSP能够与Apache服务器一起运行而开发的Servlet/JSP容器。因为Tomcat 技术先进、性能稳定，而且免费，因而深受Java爱好者的喜爱并得到了部分软件开发商的认可，成为目前比较流行的Web 应用服务器。

JDK（Java Development Kit）是Sun Microsystems针对Java开发的产品。自从Java推出以来，JDK已经成为使用最广泛的Java SDK。JDK 是整个[Java](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=20980)的核心，包括了Java运行环境、Java工具和Java基础类库。JDK是学好Java的第一步。而专门运行在[x86](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=474226)平台的Jrocket在服务端[运行效率](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=7808302)也要比Sun JDK好很多。从[Sun](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=471153)的JDK5.0开始，提供了泛型等非常实用的功能，其版本也不断更新，运行效率得到了非常大的提高。

## 2.5 IntelliJ IDEA

IDEA 全称IntelliJ IDEA，是java语言开发的集成环境，尤其在智能代码助手、代码自动提示、重构、[J2EE](https://baike.baidu.com/item/J2EE)支持、[JUnit](https://baike.baidu.com/item/JUnit)、[CVS](https://baike.baidu.com/item/CVS)整合、代码审查。其本身就包含了多个框架和多种服务，还可以通过添加插件组件构建开发环境。并且附加了一部分插件集，包括，Spring4，Mybatis等组件。

# 3 系统分析及设计

## 3.1 可行性分析

随着社会的进步，人们生活节奏的加快，传统的餐馆已不能满足用户的需求，随之兴起的网上订餐受到了用户的青睐。Internet的发展普及，使得网上订餐成为可能，人们只需一台电脑，一根网线就可以足不出户的选购自己的美食。目前网站开发的技术发展的很快，基于Java语言的JSP技术为网站开发提供了良好的支持。Java语言是当今最流行的网站开发语言之一，具有卓越的通用性、高效性、平台移植性和安全性，在网站开发中更具优势。MySQL属于中型数据库，具有同时保持数据的完整性和一致性等许多高级管理功能，使得数据库管理更加直观方便，并且提供了支持Java语言的数据库驱动的jar包。本系统计划采用B/S构架，用户只需要可以进行正常的上网操作，就可以完成在本系统的操作，所以网站设计简洁，那么大多数人群都可以进行操作。在web系统设计方面业界提供了许多通用成熟的框架来解决相应的问题，如Spring用来管理应用程序中对象的生命周期等。所以在技术方面都有解决方案。

## 3.2 需求分析

本系统分前台后台两部分，前台系统需要实现用户注册，用户登录，产品展示、用户订餐、加入购物车，生成订单，用户查看购买的商品，可以对购买的商品进行评论等功能。需要实现产品信息的提示、用户信息注册、登录管理、购物车管理、订单处理、信息反馈等模块。当用户进入本系统时，在订餐首页显示可以订购的产品信息，以供用户户选择所需产品，未登录的用户只可以访问首页，在尝试访问其他页面时会重定向到登录页面，确保接下来的操作用户是在登录状态下完成的。当用户选择订购某个产品时，能够对产品价格、数量记录到购物车中。然后，用户确认订购，填写送货的信息，联系方式，送货地址，支付方式（仅线下支付）。最后用户提交，生成订单。后台系统设计的功能主要是对用户、产品和订单的管理，用户部分主要是用户信息的展示，产品管理部分主要是产品的添加，删除，修改，以及产品列表查询。产品则可以在前台的首页展示供用户选择，删除的产品只有再次添加才能在前台展示。订单管理部分主要是订单状态的改变，可以让用户可以看见订单的配送进度。

## 3.3 系统总体设计

本系统分前台后台两部分，前台功能主要是用户注册，用户登录，产品展示，用户订餐，订单生成，动态展示产品销售情况，用户查看订单等功能。当用户登录或注册时，实时的验证用户输入是否合法。当用户进入本系统时，在订餐首页显示可以订购的产品信息，以供用户户选择所需产品。当用户选择订购某个产品时，产品价格、数量将记录到购物车中。然后，用户可以确认订购，填写送货的信息，联系方式，送货地址，支付方式。最后用户提交，生成订单。后台主要是对用户、产品和订单的管理，用户部分主要是用户信息的展示，产品管理部分主要是产品的添加，修改，以及产品的上架与下架。上架的产品则可以在前台的首页展示供用户选择，下架的产品只有再次上架才能在前台展示。订单管理部分主要是订单信息的展示。订单管理分为已处理与待处理的订单，待处理的订单是未发货的订单，选择发货订单将变为以处理的订单。用户查看订单时会显示卖家已发货。

系统设计功能结构图如图3-1所示。

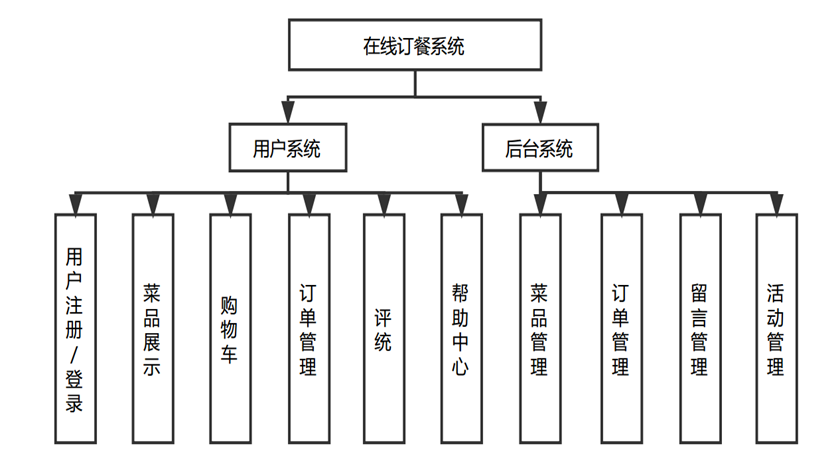


图3-1 系统设计功能结构图

### 3.3.2 系统功能结构

在线订餐系统系统功能如上图3-1所示：

其他各个模块功能介绍如下：

用户系统模块：注册登录模块、菜品展示模块、购物车模块、评价模块、查看历史订单模块、帮助中心模块；

后台系统模块：菜品管理模块、订单管理、留言管理、活动管理、

## 3.4 数据库设计与实现

订餐系统的开发过程中，对数据的操作必不可少，所以数据库的设计就尤为重要，设计过程中需要明确表之间的关联关系。

### 3.4.1 数据库概念设计

本系统有多个模块，各模块需要有相应的数据模型，各模块之间的联系也需要有相应的关系来支撑。下面主要介绍本系统中所用到的数据模型。

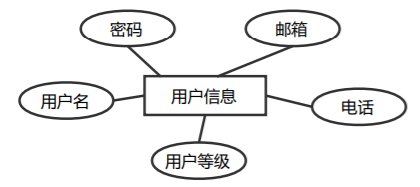
1. 用户信息表（ER图）

图3-2用户信息表实体

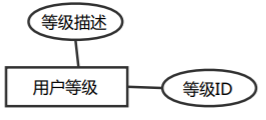
1. 用户等级表

图3-3用户等级表实体

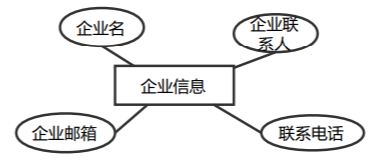
1. 企业信息表

图3-4企业信息表实体

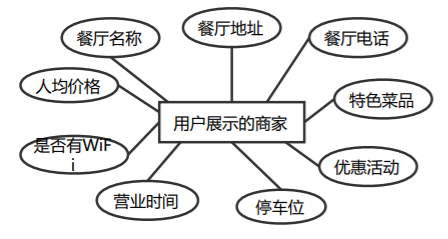
1. 用户展示的商家表

图3-5用户展示的商家表实体

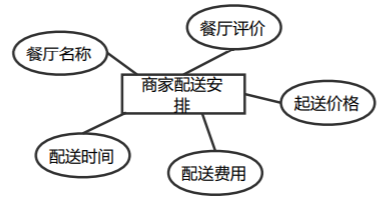
1. 商家配送费用及安排表

图3-6商家配送费用及安排表实体

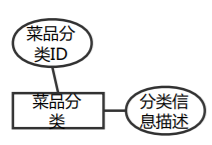
1. 菜品分类表

图3-7菜品分类表实体

1. 菜品表

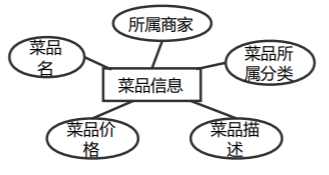


图3-8菜品表实体

1. 订单表



图3-8订单表实体

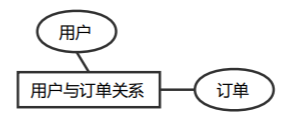
1. 用户与订单关系表

图3-9用户与订单关系表实体

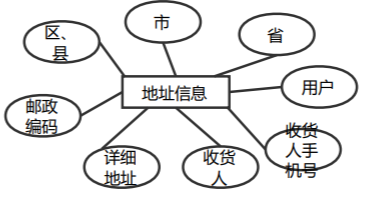
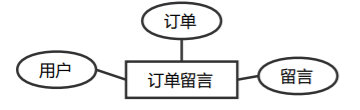
1. 用户配送地址表

图3-10用户配送地址表实体

1. 订单留言表

图3-11订单留言表实体



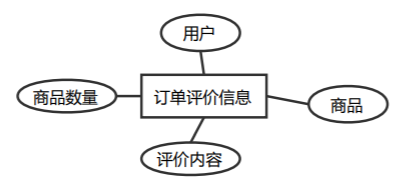
1. 订单评价表

图3-12订单评价表实体

### 3.4.2 数据库表设计

1. 用户信息表

表3-1 用户信息表 t\_user

|  |  |  |  |
| --- | --- | --- | --- |
| 列名 | 数据类型 | 可为空 | 说明 |
| ID | INT | No | 自增ID |
| USER\_ID | VARCHAR(20) | No | 用户ID |
| USERNAME | VARCHAR(20) | No | 用户名 |
| PASSWORD | VARCHAR(32) | No | 密码 |
| RANK | TINYINT | No | 用户级别 |
| AGE | TINYINT | No | 年龄 |
| EMAIL | VARCHAR(30) | No | 邮箱 |
| PHONE\_NUM | VARCHAR(20) | No | 电话 |
| USED | TINYINT | No | 是否可用 |
| CREATE\_TIME | TIMESTAMP | No | 创建时间 |
| UPDATE\_TIME | TIMESTAMP | No | 修改时间 |

1. 用户等级表

表3-2 用户等级表t\_rank

|  |  |  |  |
| --- | --- | --- | --- |
| 列名 | 数据类型 | 可为空 | 说明 |
| ID | INT | No | 自增ID |
| RANK\_ID | TINYINT | No | 等级ID |
| RANK\_DESC | VARCHAR(5) | No | 等级描述 |
| CREATE\_TIME | TIMESTAMP | No | 创建时间 |
| UPDATE\_TIME | TIMESTAMP | No | 修改时间 |

1. 企业信息表

表3-3 企业信息表t\_enterprise\_info

|  |  |  |  |
| --- | --- | --- | --- |
| 列名 | 数据类型 | 可为空 | 说明 |
| ID | INT | No | 自增ID |
| ENTERPRISE\_ID | VARCHAR(20) | No | 企业ID |
| ENTERPRISE\_COMPANY\_NAME | VARCHAR(50) | No | 企业名称 |
| ENTERPRISE\_LINK\_MAN | VARCHAR(20) | No | 企业联系人 |
| ENTERPRISE\_PHONE\_NUM | VARCHAR(20) | No | 企业联系电话 |
| ENTERPRISE\_ADDRESS | VARCHAR(100) | No | 企业地址 |
| ENTERPRISE\_EMAIL | VARCHAR(50) | No | 企业邮箱 |
| CREATE\_TIME | TIMESTAMP | No | 创建时间 |
| UPDATE\_TIME | TIMESTAMP | No | 修改时间 |

1. 用户展示的商家表

表3-4 用户展示的商家表t\_enter\_info\_for\_user

|  |  |  |  |
| --- | --- | --- | --- |
| 列名 | 数据类型 | 可为空 | 说明 |
| ID | INT | No | 自增ID |
| ENTERPRISE\_ID | VARCHAR(20) | No | 企业ID |
| COMPANY\_NAME | VARCHAR(50) | No | 商家名称 |
| ADDRESS | VARCHAR(20) | No | 商家地址 |
| PHONE\_NUM | VARCHAR(20) | No | 联系电话 |
| FEATURE\_ITEMS | VARCHAR(100) | No | 特色菜品 |
| PREFERENTIAL\_ACTIVITIES | VARCHAR(50) | No | 优惠活动 |
| PARKING\_SPACE | VARCHAR(30) | No | 停车位 |
| BUSINESS\_HOURS | VARCHAR(20) | No | 营业时间 |
| WIFI | VARCHAR(10) | No | 是否有wifi |
| AVERAGE\_PRICE | VARCHAR(5) | No | 人均价格 |
| ENTER\_IMG | VARCHAR(100) | No | 商家门面图 |
| CREATE\_TIME | TIMESTAMP | No | 创建时间 |
| UPDATE\_TIME | TIMESTAMP | No | 修改时间 |

1. 商家配送费用及安排表

表3-5 商家配送费用及安排表t\_enter\_info\_for\_dispatch

|  |  |  |  |
| --- | --- | --- | --- |
| 列名 | 数据类型 | 可为空 | 说明 |
| ID | INT | No | 自增ID |
| ENTERPRISE\_ID | VARCHAR(20) | No | 企业ID |
| ENTERPRISE\_COMPANY\_NAME | VARCHAR(50) | No | 商家名称 |
| ENTER\_IMG | VARCHAR(100) | No | 商家门面图 |
| RANK | TINYINT | No | 餐厅评价分数 |
| SEND\_PRICE | SMALLINT | No | 起送价格 |
| DISPATCH\_PRICE | SMALLINT | No | 配送费 |
| CREATE\_TIME | TIMESTAMP | No | 创建时间 |
| UPDATE\_TIME | TIMESTAMP | No | 修改时间 |

1. 菜品分类表

表3-6 菜品分类表t\_item\_type

|  |  |  |  |
| --- | --- | --- | --- |
| 列名 | 数据类型 | 可为空 | 说明 |
| ID | INT | No | 自增ID |
| ITEM\_TYPE\_ID | TINYINT | No | 菜品分类ID |
| ITEM\_TYPE\_DESC | VARCHAR(8) | No | 菜品分类信息描述 |
| CREATE\_TIME | TIMESTAMP | No | 创建时间 |
| UPDATE\_TIME | TIMESTAMP | No | 修改时间 |

1. 菜品表

表3-7 菜品表t\_item

|  |  |  |  |
| --- | --- | --- | --- |
| 列名 | 数据类型 | 可为空 | 说明 |
| ID | INT | No | 自增ID |
| ENTERPRISE\_ID | VARCHAR(20) | No | 企业ID |
| ITEM\_ID | VARCHAR(20) | No | 菜品ID |
| ITEM\_NAME | VARCHAR(50) | No | 菜品名 |
| ITEM\_PRICE | DECIMAL(5,2) | No | 菜品价格 |
| ITEM\_PIC | VARCHAR(300) | No | 菜品配图 |
| ITEM\_DESC | VARCHAR(600) | No | 菜品描述 |
| ITEM\_TYPE | TINYINT | No | 菜品所属分类 |
| CREATE\_TIME | TIMESTAMP | No | 创建时间 |
| UPDATE\_TIME | TIMESTAMP | No | 修改时间 |

1. 订单表

表3-8 订单表t\_order

|  |  |  |  |
| --- | --- | --- | --- |
| 列名 | 数据类型 | 可为空 | 说明 |
| ID | INT | No | 自增ID |
| ORDER\_ID | VARCHAR(16) | No | 订单号码 |
| USER\_ID | VARCHAR(20) | No | 用户ID |
| ORDER\_CONTENT | VARCHAR(500) | No | 订单内容 |
| ORDER\_PRICE | DECIMAL(5,2) | No | 订单价格 |
| DISPATCH\_ADDRESS | INT | No | 配送地址编号 |
| EXPECT\_TIME | VARCHAR(20) | No | 期望送达时间 |
| ORDER\_STATUS | TINYINT | No | 订单状态 |
| CREATE\_TIME | TIMESTAMP | No | 创建时间 |
| UPDATE\_TIME | TIMESTAMP | No | 修改时间 |

1. 用户与订单关系表

表3-9用户与订单关系表t\_user\_and\_order\_relation

|  |  |  |  |
| --- | --- | --- | --- |
| 列名 | 数据类型 | 可为空 | 说明 |
| ID | INT | No | 自增ID |
| USER\_ID | VARCHAR(20) | No | 用户ID |
| ORDER\_ID | VARCHAR(16) | No | 订单号码 |
| CREATE\_TIME | TIMESTAMP | No | 创建时间 |
| UPDATE\_TIME | TIMESTAMP | No | 修改时间 |

1. 用户配送地址表

表3-10 用户地址表t\_user\_address

|  |  |  |  |
| --- | --- | --- | --- |
| 列名 | 数据类型 | 可为空 | 说明 |
| ID | INT | No | 自增ID |
| USER\_ID | VARCHAR(20) | No | 用户ID |
| PROVINCE | VARCHAR(10) | No | XX省 |
| CITY | VARCHAR(10) | No | XX市 |
| DISTRICT | VARCHAR(10) | No | XX区/县 |
| POST\_CODE | VARCHAR(8) | No | 邮政编码 |
| DETAIL\_ADDR | VARCHAR(30) | No | 详细地址 |
| CONSIGNEE | VARCHAR(20) | No | 收货人 |
| PHONE\_NUM | VARCHAR(20) | No | 收货人手机号码 |
| CREATE\_TIME | TIMESTAMP | No | 创建时间 |
| UPDATE\_TIME | TIMESTAMP | No | 修改时间 |

1. 订单留言表

表3-11订单留言表 t\_leave\_words\_for\_order

|  |  |  |  |
| --- | --- | --- | --- |
| 列名 | 数据类型 | 可为空 | 说明 |
| ID | INT | No | 自增ID |
| USER\_ID | VARCHAR(20) | No | 用户ID |
| ORDER\_ID | VARCHAR(16) | No | 订单号码 |
| LEAVE\_WORDS | VARCHAR(100) | No | 用户留言 |
| CREATE\_TIME | TIMESTAMP | No | 创建时间 |
| UPDATE\_TIME | TIMESTAMP | No | 修改时间 |

1. 订单评价表

表3-12订单评价表 t\_order\_access

|  |  |  |  |
| --- | --- | --- | --- |
| 列名 | 数据类型 | 可为空 | 说明 |
| ID | INT | No | 自增ID |
| USER\_ID | VARCHAR(20) | No | 用户ID |
| ITEM\_ID | VARCHAR(16) | No | 所购商品ID |
| PAY\_COUNT | SMALLINT | No | 所购商品数量 |
| ACCESS\_WORDS | VARCHAR(100) | No | 评价内容 |
| CREATE\_TIME | TIMESTAMP | No | 创建时间 |
| UPDATE\_TIME | TIMESTAMP | No | 修改时间 |

### 3.4.3 数据库逻辑结构设计

数据库概念设计中已经分析了用户、企业、菜品、订单等主要的数据实体对象，这些实体对象是数据表结构的基本模型，最终的数据模型都要实施到数据库中，形成整体的数据结构，可以使用PowerDesigner工具完成这个数据库的建模。

物业管理系统数据库模型如图3-13所示。

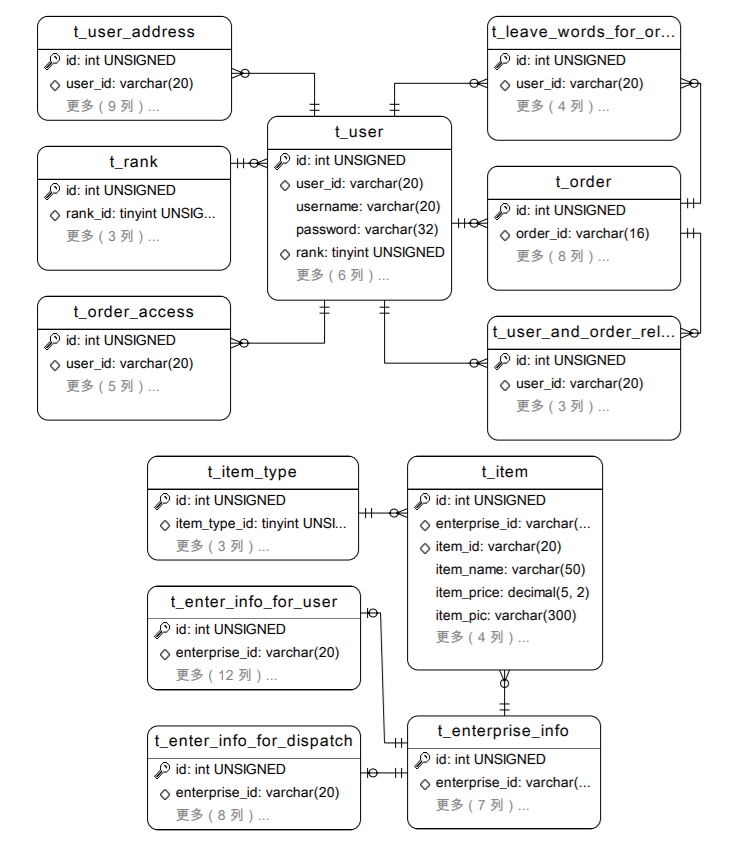


图 3-13 数据库逻辑结构

# 4 系统详细设计

## 4.1 系统环境配置

### 4.1.1 软件环境

本系统采用Apache下的开源项目Tomcat7.0作为服务器，JDK版本为8.0，使用MYSQL5.7作为后台数据库的支持，使用SpringMVC、Spring4、Mybatis3、JSP技术作为后台服务端的开发，浏览器使用谷歌。具体各部分环境配置如下：

1. Java开发工具包：jdk-8u130-windows-x64。JDK环境变量的配置如下：

变量名JAVA\_HOME：D:\Program Files\Java\jdk1.8.0（为JDK的安，装目录）

变量名Path：%JAVA\_HOME%\bin

变量名 ClassPath：.;%JAVA\_HOME%\lib\tools.jar;%JAVA\_HOME%\lib\dt.jar

1. Web服务器采用Tomcat7.0.5。Tomcat7.0环境变量配置如下：

变量名CATALINA\_HOME：D:\program\apache-tomcat-7.0.54（为Tomcat存放目录）

变量名Path：%CATALINA\_HOME%\lib;%CATALINA\_HOME%\lib\servlet-api.jar;

%CATALINA\_HOME%\lib\jsp-api.jar;

1. 数据库服务器采用MYSQL5.7。
2. 开发工具采用的是IDEA，建立Maven工程，在POM文件中配置相应框架的版本号。
3. 使用Dreamwear8.0，EditPlus美化页面。

### 4.1.2 硬件环境

1. 计算机型号 Hasee。
2. CPU处理器，Intel(R) Core(TM) i7-4710MQ CPU @ 2.50GHz。
3. 磁盘容量，硬盘1T,内存8.00GB。
4. 操作系统 Window10。

## 4.2 系统框架实现

（a）Model(VO)层

表4-1 Model层主要类

|  |  |  |
| --- | --- | --- |
| 类名 | 映射表 | 说明 |
| TUser | t\_user | 用户表 |
| TItem | t\_item | 菜品表 |
| TEnterpriseInfo | t\_enterprise\_info | 企业信息表 |
| TEnterInfoForDispatch | t\_enter\_info\_for\_dispatch | 商家配送安排表 |

续表4-1

|  |  |  |
| --- | --- | --- |
| 类名 | 映射表 | 说明 |
| TenterInfoForUser | t\_enter\_info\_for\_user | 用户展示的商家表 |
| TItemType | t\_item\_type | 菜品分类表 |
| TLeaveWordsForOrder | t\_leave\_words\_for\_order | 订单留言表 |
| TOrder | t\_order | 订单表 |
| TOrderAccess | t\_order\_access | 订单评价表 |
| TRank | t\_rank | 用户等级表 |
| TUserAddress | t\_user\_address | 用户地址表 |
| TUserAndOrderRelation | t\_user\_and\_order\_relation | 用户与订单关系表 |
| ShoppingCart |  | 购物车 |

（b）业务逻辑层

表4-2 业务逻辑主要类

|  |  |  |
| --- | --- | --- |
| 类名 | 实现类 | 说明 |
| UserService | UserServiceImpl | 用户操作 |
| EnterpriseInfoService | EnterpriseInfoServiceImpl | 企业信息操作 |
| EnterInfoForDispatchService | EnterInfoForDispatchServiceImpl | 商家派送信息操作 |
| EnterInfoForUserService | EnterInfoForUserServiceImpl | 商家显示信息操作 |
| ItemService | ItemServiceImpl | 菜品操作 |
| OrderService | OrderServiceImpl | 订单操作 |
| OrderAccessService | OrderAccessServiceImpl | 订单评价操作 |

（c）控制层类

表4-3 Action,Servlcet

|  |  |
| --- | --- |
| 类名 | 说明 |
| AdminOrderController | 管理员订单操作 |
| AdminUserController | 管理员用户操作 |
| CartController | 购物车操作 |
| DetailspController | 菜品详情操作 |
| EnterpriseInfoController | 企业信息操作 |
| ItemController | 菜品信息操作 |
| OrderController | 订单操作 |
| SearchController | 搜索项操作 |
| ShopController | 商家操作 |
| UserController | 用户操作 |

## 4.3 系统公共模块设计

### 4.3.1 数据库连接和事务管理器

本系统通过在application-dao.xml文件中配置Druid数据源连接数据库，具体代码如下：

1. 配置 Druid 数据源

<bean id="dataSource" class="com.alibaba.druid.pool.DruidDataSource"

destroy-method="close">

<property name="url" value="${jdbc.url}"/>

<property name="username" value="${jdbc.username}"/>

<property name="password" value="${jdbc.password}"/>

<property name="driverClassName" value="${jdbc.driverClassName}"/>

<property name="maxActive" value="10"/>

<property name="minIdle" value="5"/>

</bean>

1. 配置会话工厂SessionFactory，并将数据源dataSource通过属性注入进去

<bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">

<property name="dataSource" ref="dataSource"/>

<property name="configLocation" value="classpath:mybatis/mybatisConfig.xml"/>

<!-- 加载mapper.xml文件 -->

<property name="mapperLocations" value="classpath\*:mybatis/mapper/\*.xml"/>

</bean>

1. 配置DAO接口的SQL映射文件

<bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">

<property name="basePackage" value="com.sust.dao"/>

<property name="sqlSessionFactoryBeanName" value="sqlSessionFactory"/>

</bean>

1. 配置Spring 的声明式事务，配置 Mybatis的事务管理器

<bean id="transactionManager"

class="org.springframework.jdbc.datasource.DataSourceTransactionManager">

<property name="dataSource" ref="dataSource" />

</bean>

1. 配置事务属性

<tx:advice id="txAdvice">

<tx:attributes>

<tx:method name="save\*"/>

<tx:method name="insert\*"/>

<tx:method name="add\*"/>

<tx:method name="create\*"/>

<tx:method name="delete\*"/>

<tx:method name="update\*"/>

<tx:method name="find\*" propagation="SUPPORTS" read-only="true" />

<tx:method name="select\*" propagation="SUPPORTS" read-only="true" />

<tx:method name="query\*" propagation="SUPPORTS" read-only="true" />

<tx:method name="get\*" propagation="SUPPORTS" read-only="true" />

</tx:attributes>

</tx:advice>

1. 配置事务切入点, 再把事务属性和事务切入点关联起来

<aop:config>

<aop:advisor advice-ref="txAdvice"

pointcut="execution(\* com.sust.service..\*.\*(..))" />

</aop:config>

### 4.3.2 系统安全控制

系统除了首页可以任意访问之外，其他页面都需要登录才可以访问，登录操作是比较用户名和密码是否同时在数据库中存在，若输入无误，则可成功进行浏览其他信息和下单等操作，后台子系统是通过一个子路径进行访问，并且需要再次登录，切身份等级为管理员的用户才可以登录。用户系统与后台系统没有关联的路径，后台只能通过特定的URL进行登录，系统之间隔离保持了良好的安全性。并且配置了拦截器，每一次请求都会判断是否是已登录用户的请求，是则继续完成请求，否则转到登录页面进行登录，更加提高了安全性。

## 4.4系统主要模块功能的实现

### 4.4.1登录注册功能

用户在登录页面来进行登录，只有输入正确的用户名密码或工号身份证号等才可登录成功，对于新用户可进行注册，点击注册按钮进入相应的注册界面进行注册，注册成功后，会在数据库中生成对应的记录，并跳转至登录界面进行登录。如图4-1，图4-2 所示。具体实现代码如下所示，分别编写业务逻辑层，和控制层；DAO层只有接口说明，实现代码是在程序运行时动态生成的。

1. 控制层

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

public String loginAction(

@RequestParam("username") String username,

@RequestParam("password") String password,

HttpServletRequest request,

HttpServletResponse response){

logger.info("username:{}",username);

Tuser user = userService.queryUserInfoByNameAndPwd(username,password);

if (user != null){

CookieUtils.setCookie(request,response,"userId",user.getUserId());

return "redirect:/home";

}else {

return "redirect:/error";

}

}

控制层根据业务逻辑层返回的结果来判断最后跳转的页面，并设置相应的内容，此处为当在数据库中查询到相应的用户名和密码后，设置cookie用来跟踪用户后续的操作，然后跳转到首页，用户就可以接着进行下一步操作；当在数据库中没有查询到相应的用户名和密码时，返回到错误页面并提示详情。

1. 业务逻辑层

public TUser queryUserInfoByNameAndPwd(String username, String password) {

TUserExample example = new TUserExample();

TUserExample.Criteria criteria = example.createCriteria();

criteria.andUsernameEqualTo(username).andPasswordEqualTo(password);

List<TUser> users = userdao.selectByExample(example);

return users.size()>0 ? users.get(0) : null;

}

此处操作example来设置相应的查询条件，即用户名和密码，然后再调用DAO层的接口来真正执行与之对应的SQL查询。业务逻辑层根据查询结果，设置返回值。

1. DAO层

根据用户名和密码来进行登陆判断，若在数据库中查询到了唯一的结果，则说明数据库存在该用户。由于使用的SSM框架整合的方式[10]，所以以往的在代码中的单独的SQL语句已经不存在，Mybatis根据业务逻辑层的参数来设置查询的条件。

<sql id="Base\_Column\_List" >

id, user\_id, username, password, rank, birth, email, phone\_num, used, create\_time,

update\_time

</sql>

<select id="selectByExample" resultMap="BaseResultMap" parameterType="com.sust.model.TUserExample" >

select

<if test="distinct" >

distinct

</if>

<include refid="Base\_Column\_List" />

from t\_user

<if test="\_parameter != null" >

<include refid="Example\_Where\_Clause" />

</if>

<if test="orderByClause != null" >

order by ${orderByClause}

</if>

</select>

1. 表示层

SpringMVC根据控制层返回值来进行页面的跳转，返回值中的字符串会被解析成对应目录下的同名的JSP文件，JSP被编译运行后，返回到浏览器中就变成了HTML的静态页面

图4-1 系统登录页面



图4-2 系统注册页面

在Controller中首先新建一个user对象出来，然后根据insert函数的返回值来判断用户是否注册成功，若为true,说明该用户注册成功，否则返回返回到错误页面并进行错误信息的说明。

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

public String registerDo(@RequestParam("username")String username,

@RequestParam("password")String password,

@RequestParam("email")String email,

@RequestParam("phone\_num")String phone\_num){

TUser user = new TUser();

user.setUserId(IdUtils.getNextId());

user.setUsername(username);

user.setPassword(password);

user.setEmail(email);

user.setPhoneNum(phone\_num);

user.setRank(Byte.valueOf("2")); //普通用户

user.setUsed(Byte.valueOf("1")); //可用

boolean ok = userService.insertUser(user);

if(ok){

return "redirect:/home";

}else {

return "front/error";

}

}

### 4.4.2 商品展示功能

未登录时，首页展示了一部分商品，若需要查看更多的商品，就需要用户登录后才可以操作；当用户登录后，就可以查看所有的商品信息，首页只展示了简略的商品信息，点击商品的图片或者名称都可以跳转到商品的详情页面，商品都是属于商家，所以也提供了从商品跳转到商家的按钮，在商家中可以查看商家的信息和商家的菜谱。

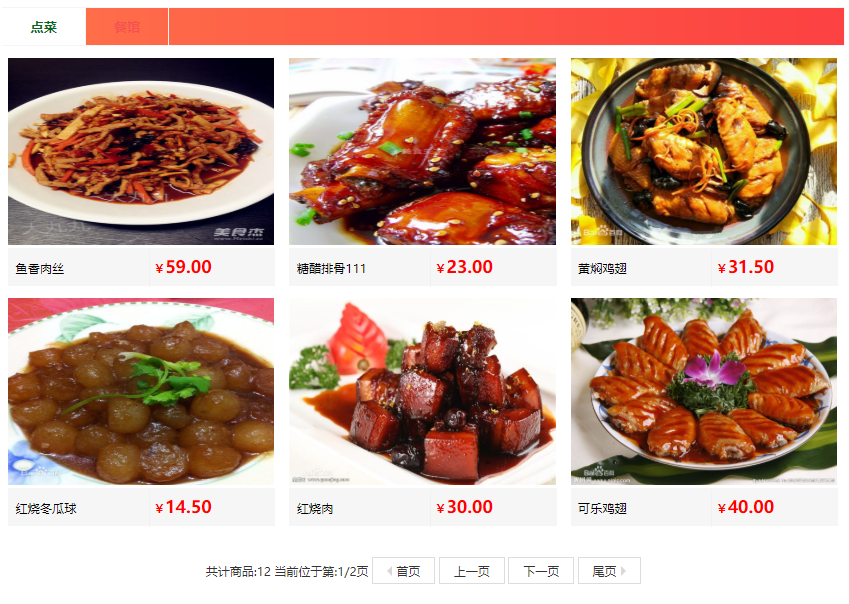


图4-3 首页商品展示



图4-4 首页店铺信息



图4-5 商品详情页面

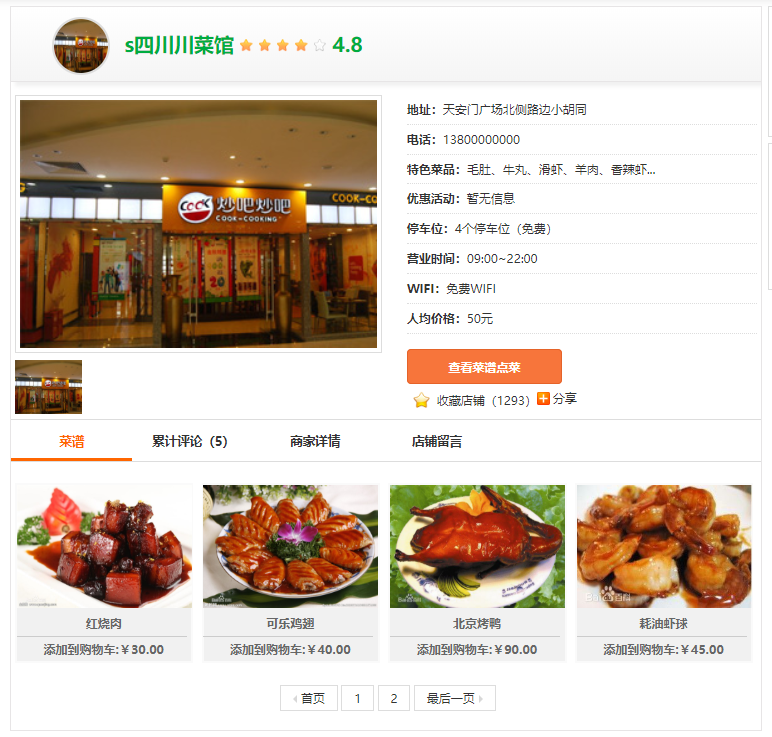


图4-6 店铺详情页面

菜品列表、商家列表都不可能在页面上全部展示出来，所以使用了分页功能，每页只显示一部分商家或者菜品的信息，用户点击下一页上一页可使用户的体验更加流畅，实现是通过创建Page对象，点击上一页下一页就会获取不同的数据，然后将数据封装成list，JSP页面通过<c:Foreach>标签显示在页面上即可。

确定当前页数的合法性

public void vaildCurrentPageNum(Integer curpage){

int curpagetmp = 1;

if (curpage == null) {

curpagetmp = 1;

} else {

curpagetmp = curpage;

}

currentPage = curpagetmp;

}

更新Page对象的内容

public <T> void updatePageInfo(List<T> list){

PageInfo<T> pageInfo = new PageInfo<>(list);

this.setTotalNumber((int)pageInfo.getTotal());

this.count();

}

具体计算操作

private void count()

{

//计算总页数

int totalPageTemp = this.totalNumber /this.pageNumber;

int plus = (this.totalNumber % this.pageNumber) == 0 ? 0 : 1;

totalPageTemp = totalPageTemp + plus;

if(totalPageTemp <= 0) {

totalPageTemp = 1;

}

this.totalPage = totalPageTemp;

//设置当前页

if(this.totalPage < this.currentPage) {

this.currentPage = this.totalPage;

}

if(this.currentPage < 1) {

this.currentPage =1 ;

}

}

按条件的分页查询的伪代码如下：

首先在Controller层中new 一个Page对象，并将其返回给页面：

public String toIndex(@RequestParam(value = "curpage", required = false) Integer curpage, Model model) {

//分页数据

Page page = new Page();

page.vaildCurrentPageNum(curpage);

//查询信息

……

page.updatePageInfo(查询到的列表);

model.addAttribute("page", page);

return "itemListPage";

}

之后在业务逻辑层查询列表代码之前添加一句代码：

PageHelper.startPage(curpagetmp, pageNumber);

之后框架就会帮你完成总数的查询，根据总数和列表就可以完成Page对象其他属性的更新操作，具体的Page对象属性如下所示，其他成员方法已经在上面列举：

public class Page

{

private int totalNumber; //总条数

private int currentPage; //当前页码

private int totalPage; //总页数

private int pageNumber = 6; //每页显示条数

//getter和setter方法很简单就不列举在此

}

在上面的代码中，页面传递过的参数仍需要进一步的判断，避免出现错误的参数，从而降低在用户页面出现错误信息的频率。

### 4.4.3 购物车的设计与实现

用户在找到自己需要的商品之后，通常还会在浏览其他的商品，这时候就需要购物车的功能，用户将商品添加到购物车中，最后一次性结算。当用户选择商品点击加入购物车按钮时，这件商品的的信息会被加入到浏览器的cookie当中，当用户点击购物车按钮时，会跳转到购物车页面，并且显示所购买的商品，数量和价格，并且在选中单选框后，计算出选中的商品的总价。如图4-7所示。

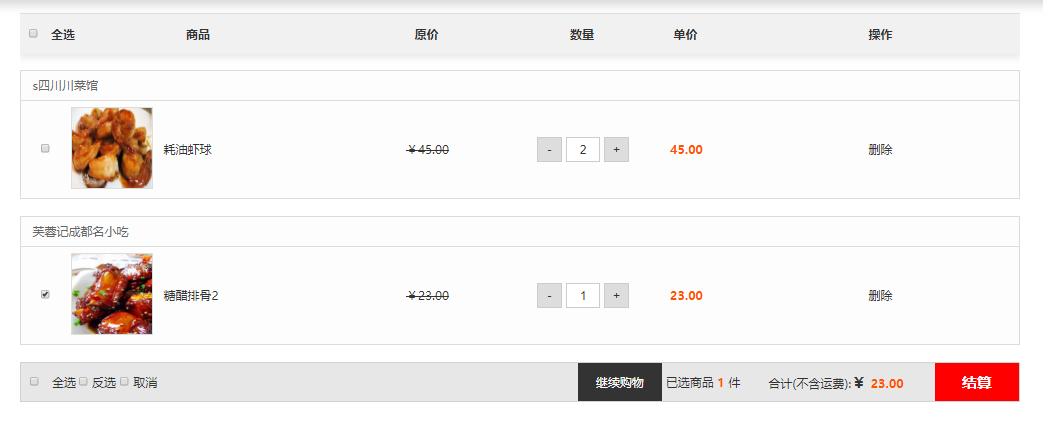


图4-7 购物车

主要实现如下：

计算在购物车中商品的总价格

public BigDecimal getTotal(HttpServletRequest request,HttpServletResponse response){

BigDecimal total = null;

String cookieValue = CookieUtils.getCookieValue(request, CookieConstant.SHOPPING\_CART\_NAME);

List<ItemDetailDto> itemDetailDtos = null;

if(cookieValue != null) {

itemDetailDtos = getItemFromCart(cookieValue);

}

//计算总价格

for (ItemDetailDto itemDetailDto : itemDetailDtos) {

double val = 0.0;

val += itemDetailDto.getItemPrice().doubleValue() \* itemDetailDto.getCnt();

total = new BigDecimal(val);

}

return total;

}

删除购物车中的商品：

private void removeItemFromCart(String itemId,

HttpServletRequest request,

HttpServletResponse response) {

String cookieVal = CookieUtils.getCookieValue(request, CookieConstant.SHOPPING\_CART\_NAME);

ShoppingCart shoppingCart = JsonUtils.jsonToPojo(cookieVal, ShoppingCart.class);

shoppingCart.removeGoods(itemId);

CookieUtils.setCookie(request, response, CookieConstant.SHOPPING\_CART\_NAME, JsonUtils.objectToJson(shoppingCart));

}

添加商品到购物车：

private void addItemToCart(String itemId,

Integer cnt,

HttpServletRequest request,

HttpServletResponse response) {

String CartCk = CookieUtils.getCookieValue(request, CookieConstant.SHOPPING\_CART\_NAME);

if (CartCk == null) {

Map<String, Integer> shoppingCart = new HashMap<>();

this.addGoods(shoppingCart, itemId, cnt);

String s = JsonUtils.objectToJson(shoppingCart);

logger.info("cookie:{}", s);

CookieUtils.setCookie(request, response, CookieConstant.SHOPPING\_CART\_NAME, s);

} else {

String cookieVal = CookieUtils.getCookieValue(request, CookieConstant.SHOPPING\_CART\_NAME);

Map<String, Integer> shoppingCart = JsonUtils.jsonToPojo(cookieVal, HashMap.class);

this.addGoods(shoppingCart, itemId, cnt);

CookieUtils.setCookie(request, response, CookieConstant.SHOPPING\_CART\_NAME, JsonUtils.objectToJson(shoppingCart));

}

}

从cookie中得到商品数据，返回给页面：

public List<ItemDetailDto> getItemFromCart(String cookieVal){

List<ItemDetailDto> itemDetailDtos = new ArrayList<>();

Map<String, Integer> cart = JsonUtils.jsonToPojo(cookieVal, HashMap.class);

List<String> itemIdList = cart.keySet().stream().collect(Collectors.toList());

List<TItem> itemList = itemService.BatchQueryByitemId(itemIdList);

for (TItem item : itemList) {

ItemDetailDto itemDetailDto = new ItemDetailDto();

TEnterpriseInfo tEnterpriseInfo = enterpriseInfoService.queryById(item.getEnterpriseId());

itemDetailDto.setItemDetailDto(item, tEnterpriseInfo);

itemDetailDto.setCnt(cart.get(item.getItemId()));

itemDetailDtos.add(itemDetailDto);

}

return itemDetailDtos;

}

### 4.4.4 用户订餐的设计与实现

用户将购物车中的物品选中之后，点击结算按钮，会跳转到结算页面，此时会显示用户订单信息，用户需要配送的地址信息，支付方式由于客观原因仅支持线下支付，用户需要选择配送地址，还可以添加订单附言。如下图4-8所示：

图4-8 用户订单信息确认界面

当用户确认完订单之后就可以点击生成订单，结果展示如图4-9所示：

图4-9 订单生成成功界面

主要实现代码如下：

### 4.4.5 订单操作的设计与实现

订单设置了准备中、派送中、已完成三个状态。

订单生成后用户可以在用户中心查看自己的订单。

当订单状态在准备状态下，用户可以取消订单。

当订单在其他状态下用户只可以查看，订单状态的更改由管理员更新。

当订单状态为已完成时，用户可以对订单进行评价。

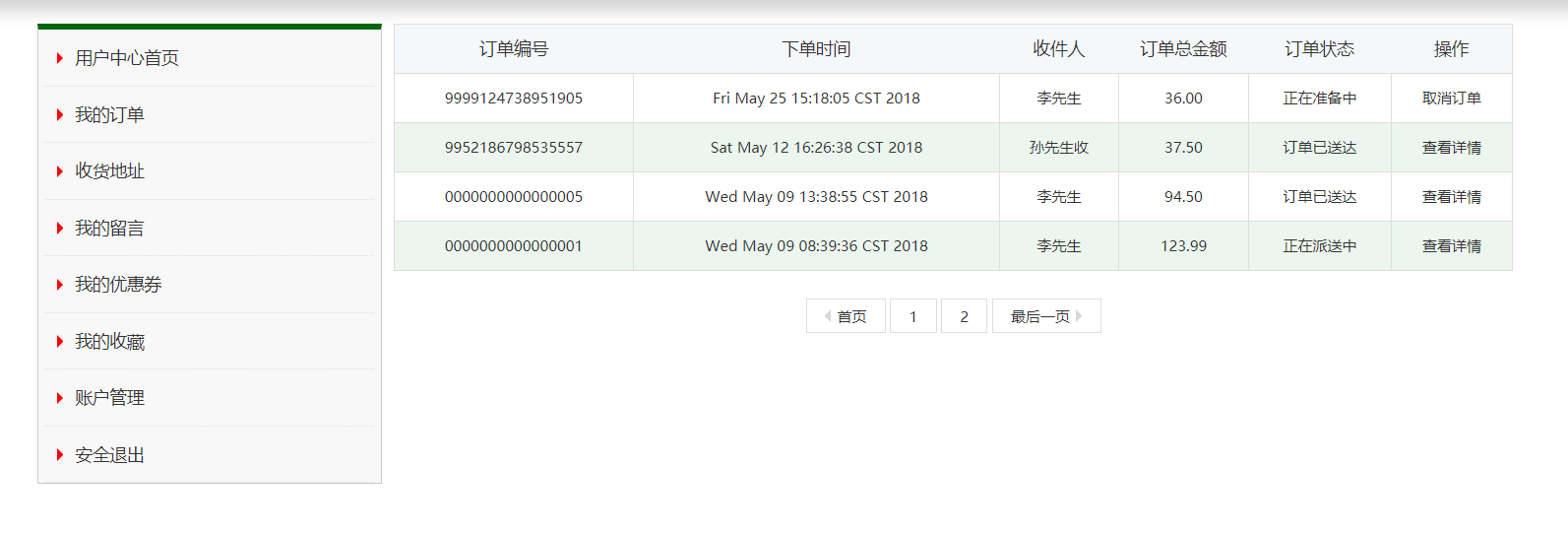


图4-10 用户中心我的订单状态

### 4.4.6用户评论的设计与实现

只有将订单状态变成已完成，此时才可以给订单添加评价，如下图4-11所示：



图4-11 用户中心订单评价

具体实现为：当查看订单详情的时候，除了将基本信息传递给页面，再向页面传递订单的状态，在JSP页面中使用<c:if >标签进行判断，当状态为已完成时，再显示评价项。

JSP页面实现如下：

<c:if test="${orderStatus == 2}">

<span class="Font14 FontW Lineheight35 Block">来给个评价吧：

<input type="text" id="access" class="input\_addr" placeholder="来说说你的感受"/>

<input type="button" id="access\_sub\_btn" class="Submit"

value="评价" onclick="access\_sub()"/>

</span>

</c:if>

Controller层实现如下：

@RequestMapping("/user/user\_orderdetail.html")

public String touser\_orderdetail(@RequestParam("orderid")String orderid,

HttpServletRequest request,

Model model){

TOrder order = userService.queryOrderDetailByOrderId(orderid);

Map<String,String> itemIdAndCnt = splitOrderContent(order.getOrderContent());

List<OrderContentDto> orderContentDtoList = new ArrayList<>();

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

String key = entry.getKey();

TItem item =itemService.queryById(Integer.valueOf(key));

OrderContentDto contentDto = new OrderContentDto();

contentDto.setItemId(item.getItemId());

contentDto.setItemName(item.getItemName());

contentDto.setItemPrice(item.getItemPrice());

contentDto.setCnt(Integer.valueOf(itemIdAndCnt.get(key)));

contentDto.setTotal(contentDto.getItemPrice().multiply(new BigDecimal(contentDto.getCnt())));

orderContentDtoList.add(contentDto);

}

String orderId = order.getOrderId();

BigDecimal dispatchPrice = DisPatchPriceConstants.DISPATCH\_PRICE;

TUserAddress address = userService.queryAddressById(order.getDispatchAddress(), order.getUserId());

BigDecimal orderPrice = order.getOrderPrice();

model.addAttribute("orderId",orderId);

model.addAttribute("orderContentDtoList",orderContentDtoList);

model.addAttribute("dispatchPrice",dispatchPrice);

model.addAttribute("orderPrice",orderPrice);

model.addAttribute("address",address);

model.addAttribute("orderStatus",order.getOrderStatus());

return "front/user\_orderdetail";

}

Service层实现如下：

public TOrder queryOrderDetailByOrderId(String orderid) {

TOrderExample example = new TOrderExample();

TOrderExample.Criteria criteria = example.createCriteria();

criteria.andOrderIdEqualTo(orderid);

List<TOrder> orderList = orderdao.selectByExample(example);

return orderList.size() > 0 ? orderList.get(0) : null;

}

### 4.4.7管理员商品管理的设计与实现

点击侧边栏的菜品管理，可以查看菜品列表，列表页面如下图4-12所示：。

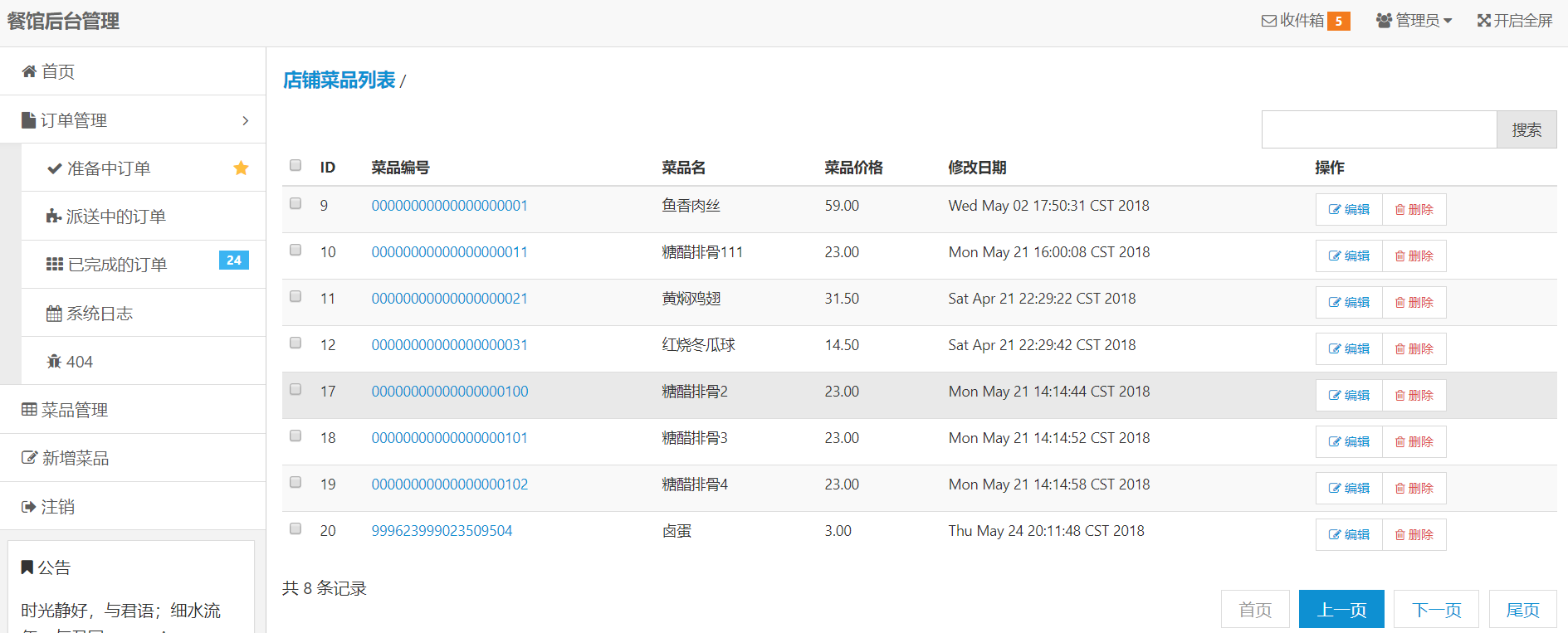


图4-12 用户中心商品列表

Controller层实现：

public String list(@RequestParam(value = "enterpriseid", required = true) String enterpriseid, Model model) {

List<TItem> items = itemService.queryByPage(enterpriseid);

model.addAttribute("items", items);

model.addAttribute("enterpriseid", enterpriseid);

return "back/admin-item-manager";

}

Service层实现：

public List<TItem> queryByPage(String enterpriseid)

{

TItemExample tItemExample = new TItemExample();

TItemExample.Criteria criteria = tItemExample.createCriteria();

criteria.andEnterpriseIdEqualTo(enterpriseid);

return itemDao.selectByExample(tItemExample);

}

点击删除按钮就可以直接删除在数据库中的记录；点击菜品列表页的编辑按钮或菜品编号，都可以进入菜品详情页面，也就是修改页面，除了编号不能更改之外，其他的都可以修改，更改完成之后点击提交按钮，提示更改成功就成功将更改保存到数据库中。具体页面展示如图4-13所示：



图4-13 用户中心商品详情修改页

Controller层为，在跳转到详情页面时，就将这个菜品的详细信息也查询出来传递到详情页面的输入框中，当用户对输入框中的信息做出修改之后，点击提交保存，此时会将输入框中的信息重新提交到后台，后台就对此菜品做出更新操作，实现如下：

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

@ResponseBody

public String listdedail(@RequestParam(value = "itemId", required = true) String itemId,

@RequestParam(value = "itemName", required = true) String itemName,

@RequestParam(value = "itemPrice", required = true) String itemPrice,

@RequestParam(value = "itemDesc", required = true) String itemDesc,

@RequestParam(value = "itemType", required = true) String itemType,

Model model) {

TItem item = new TItem();

item.setItemId(itemId);

item.setItemName(itemName);

item.setItemDesc(itemDesc);

item.setItemPrice(BigDecimal.valueOf(Double.valueOf(itemPrice)));

item.setItemType(Byte.valueOf(itemType));

logger.info("update:item:{}",item);

boolean ok = itemService.updateByItem(item);

if(ok ){

return JsonUtils.objectToJson(Result.build(0, "更新成功"));

}else {

return JsonUtils.objectToJson(Result.build(1, "更新失败"));

}

}

Service层实现条件更新操作：

public boolean updateByItem(TItem item) {

TItemExample example = new TItemExample();

TItemExample.Criteria criteria = example.createCriteria();

criteria.andItemIdEqualTo(item.getItemId());

int update = itemDao.updateByExampleSelective(item, example);

return update > 0 ? true : false;

}

新增菜品涉及到两个操作，一个是图片上传，和菜品信息上传；我的代码实现是在填写完菜品信息后，添加图片时需单独点击上传图片，待到图片上传成功后才可以点击提交按钮。



图4-14 用户中心商品新增页

上传图片首先需要得到从前台传递的图片文件，使用 MultipartFile类来获取图片文件，然后将图片内容下载存放到指定目录，最后将图片路径返回给页面。具体实现如下：

private String uploadFile(MultipartFile file, HttpServletRequest request) throws IOException {

if(!file.isEmpty()) {

String subpath = "/upload/";

String path = request.getServletContext().getRealPath(subpath);

String srcfilename = file.getOriginalFilename();

String suffix = srcfilename.substring(srcfilename.indexOf("."));

String filename = IdUtils.getNextId()+suffix;

File filepath = new File(path, filename);

if (!filepath.getParentFile().exists()) {

filepath.getParentFile().mkdirs();

}

file.transferTo(new File(path + File.separator + filename));

logger.info("imgPath : {}:{}",path,filename);

return subpath + filename;

}

return null;

}

### 4.4.8管理员订单状态管理的设计与实现

每个订单生成的时候都是准备中的状态，然后商家开始准备订单，准备完成之后点击去派送按钮，订单状态就会转变为派送中，当用户收到订单后，管理员再将订单状态更改为已完成，此时用户就可以对订单进行评价。具体如下图所示。



图4-15 管理员操作准备中订单



图4-16 管理员操作派送中订单

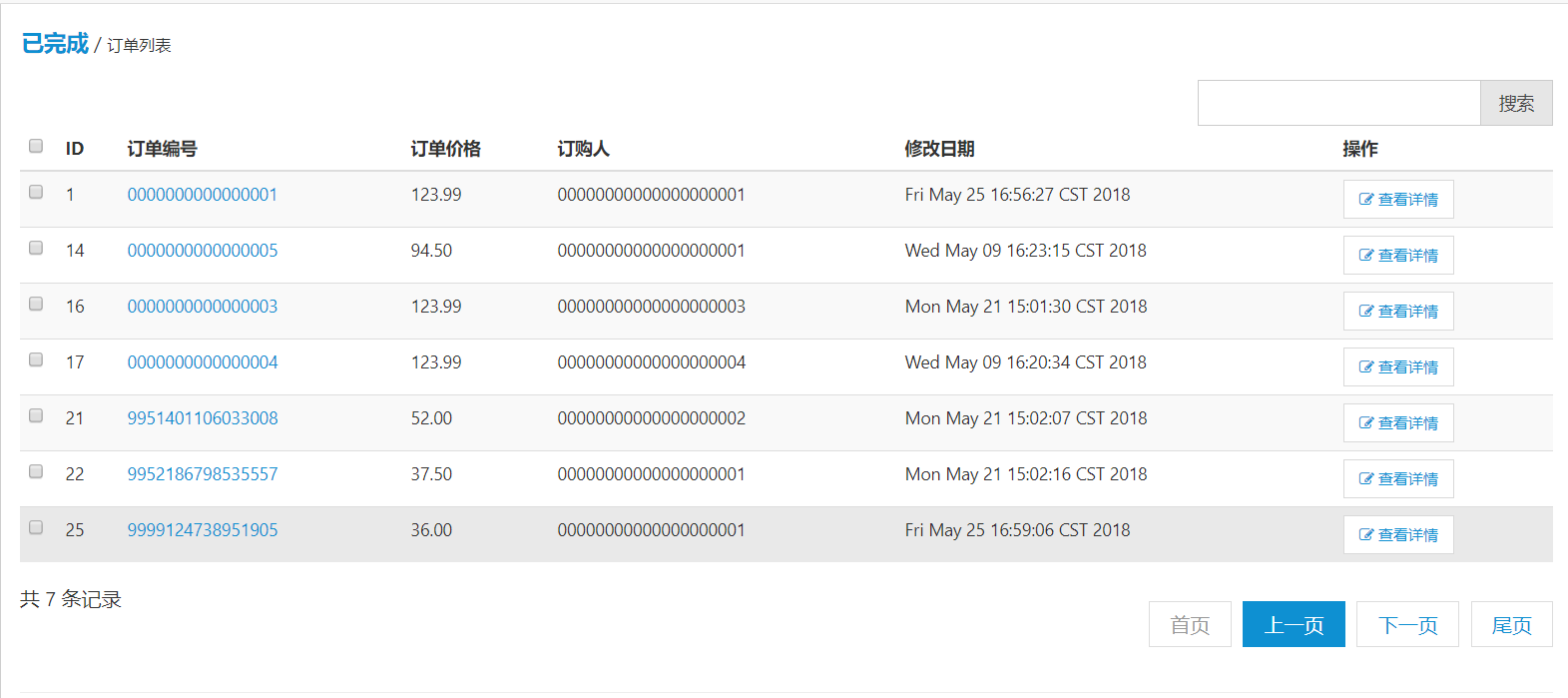


图4-17 管理员查看已完成订单

具体的代码实现分析如下：

Controller层代码示意如下

public String updatStatus(@RequestParam("orderId")String orderId){

Preconditions.checkNotNull(orderId, "订单Id不可为空");

boolean ok = orderService.updateOrderStatus(orderId,"1");

if (ok) {

return JsonUtils.objectToJson(Result.build(0, "状态更新成功"));

} else {

return JsonUtils.objectToJson(Result.build(1, "状态更新失败"));

}

}

Service层代码实现如下，首先新建一个order对象，设置好状态和orderId，再根据ID更新对应的记录，就完成了对状态的更新。

public boolean updateOrderStatus(String orderId, String status) {

TOrder order = new TOrder();

order.setOrderId(orderId);

order.setOrderStatus(Byte.valueOf(status));

TOrderExample example = new TOrderExample();

TOrderExample.Criteria criteria = example.createCriteria();

criteria.andOrderIdEqualTo(orderId);

int update = orderMapper.updateByExampleSelective(order, example);

return update > 0 ? true : false;

}

# 5 系统测试与性能分析

## 5.1 测试目的

软件测试是为了发现程序中的错误而执行程序的过程，是软件开发周期中非常重要的一步，好的测试方案是极可能发现迄今为止尚未发现的错误的测试方案。正确认识测试是为了找出错误而不是为了证明程序的正确性是非常必要的，只有这样才能真正的测试好开发的程序。测试阶段的根本目标是尽可能地发现并排除软件隐藏的错误，最终把一个高质量的软件系统交给用户使用[16]。系统的测试方法有两种：黑盒测试与白盒测试。其中，黑盒测试是不考虑代码，仅对其功能进行测试的方法；白盒测试是代码测试。本系统主要采用黑盒测试。

## 5.2 测试方法

### 5.2.1 模块测试

每个子系统按照功能划分成模块，每个模块完成一个清晰定义的子功能，而且这个功能模块和同级其他模块的功能之间相互依赖关系很小。因此把每个模块作为一个单独的实体进行测试，比较容易检验处模块是否有错误。在这个过程中发现的往往是编码与详细设计的错误。

### 5.2.2 子系统测试

子系统的测试是把经过单元测试的模块放在一起形成一个子系统来测试。模块之间的协调与通信是这个测试中的主要问题，而这部分测试着重测试模块的接口。

### 5.2.3 系统测试

把所有经过测试的子系统装配成一个完整的体系来测试。在这个过程中不近应该发现设计与编码的错误，还应该验证系统确实能提供需求说明书中指定的功能。在这部分测试中发现的往往是软件设计中的错误，也有可能发现需求说明书中的错误。

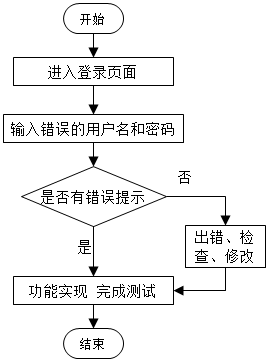
## 5.3 功能测试

功能测试，也称黑盒测试，是对产品的功能特性与操作行为进行测试的方法。以下对部分功能进行测试，列出部分测试结果。主要涉及的功能模块有：用户登录测试、添加测试。

### 5.3.1 登录测试

该测试主要用于验证系统是否能正确的处理用户登录时遇到的各种情况。用户名和密码输入为空时可通过页面上的Javascript函数检测出来，不需要提交到Action处理，节省系统的开销。

登录测试用例的测试过程如图5-1所示。

图5-1 登录模块测试图

系统中，通过login.jsp登录页面输入信息后提交到Controller处理，判断是否为合法用户，若登录不合法则给出提示信息，登录合法时根据用户的类型跳转到指定的页面。输入错误的用户名或者密码之后提示错误，并返回到登录页面。

### 5.3.2 注册测试

该测试主要用于测试添加新用户行为。测试主要集中在添加新用户信息以及修改个人信息时的测试类似此处。该测试的主要内容包括:用户名填写、密码、电话号码、邮箱格式是否正确。增加用户信息的测试内容如表5-1所示。

表5-1 增加新用户测试用例

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 输入条件 | 测试用例说明 | 测试数据 | 期望结果 | 测试结果 |
| 用户名 | 长度、格式不正确 | 用户名 | 正确不提示，错误提示 | 格式不正确无法存入数据库 |
| 密码 | 长度、格式不正确 | 密码 | 同时正确不提示，错误提示不一致 | 格式不正确无法存入数据库 |
| 电话 | 长度、格式不正确 | 电话号码 | 正确不提示，错误提示 | 格式不正确无法存入数据库 |
| 邮箱 | 长度。格式不正确 | 邮箱 | 正确不提示，错误提示 | 格式不正确无法存入数据库 |

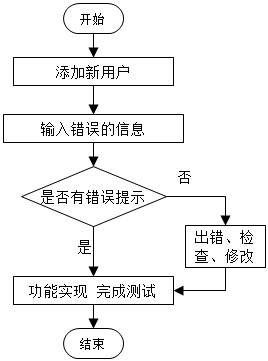
上述测试用例的测试过程如图5-2所示。

图5-2 添加新用户测试图

# 6 总 结

通过本次毕业设计，我收获了许多，也感到有好多遗憾。首先，在拿到题目时，我有点胆怯，看完了整个项目需求，结合自身所掌握的熟练度和所了解的多少，我感觉凭借自己是难以完成的，对于项目本身而言，它是一个网站类题目，对于网站可以通过多种开发语言技术进行开发，我本身学的是Java编程语言，对SpringMVC,Mybatis3,Spring有些许学习和了解，对于早前的Servlet有很多遗忘所以选择使用SSM三大框架整合类完成。而一个完美成型的网站最重要的两部分就是成熟稳定的后台技术，和美观高大上的前端技术，我只知其一，而不知其二。决定先把大体的框架完成起来，美化后面再说，通过查看相关文献资料搭起主要环境，在编写的过程中，好多东西都是以前从未深入地去学习理解过，所以开始编写时进度很慢，每遇到一个问题，就会纠结持续好久，不断地在网上查找资料和问题的导致原因，去一点一点地排错和编写，一个问题经过一天两天甚至更多时间都未能解决，好在自己坚持下来，大体完成架构功能。在后期的编写中，问题也是反反复复，不断的查BUG并解决，并适当的通过查看学习将自己所设计的页面美化了下，比起开始能好许多。通过本次毕业设计网站的开发，我意识到自己对所需知识不扎实牢固，遗忘过多，对相关技术的深度学习十分欠缺。但正因为这些因素，在编写的过程中，我通过不断的学习来解决这些自己从未解决的难点，也为自己的基础奠定了更多的积累和经验。

# 致 谢

到现在，毕业设计已经接近尾声。通过此次毕业设计让我感受颇深，因为这次是对大学四年所学知识的总结和运用。整个项目的设计编写是自己独立完成，或多或少对自己的意义很深。使自己对技术的掌握更加深入，对未来的学习和工作有着深远的意义。通过项目的设计与完成，让我对完成一个项目从开始的需求分析和框架的搭建更加熟悉，对自己还未掌握和掌握不熟练的技术有了进一步的学习和理解。当然这些都归功于学校，学院老师和师哥师姐以及身边的同学的鼓励和帮助，我才可以通过不断地思考和学习去掌握理解更多的先进的技术来丰富自己，充实自己。为此，我特别感谢我的指导老师杨云老师对我的指导，同时感谢学姐秦一梅对我的帮助，让我在诸多正能量的基石上迈的更远。

最后，我要感谢陕西科技大学对我四年的培养，让我学到了不管在生活中，学习上，遇到问题要积极的面对问题，从问题中找解决它的办法。同时感谢大学中所有传道授业于我的每一位老师，也感谢身边一直带给自己正能量的所有同学。

# 参 考 文 献

[1] 疯狂软件. Spring MVC+MYBatis企业应用实战[M]. 电子工业出版社,2017:3-14.

[2] Tapio Lahdenmaki. 数据库索引设计与优化[DB]. 电子工业出版社,2016.

[3] Craig Walls. Spring实战（第三版）[CP]. 人民邮电出版社, 2016,(01):17-22.

[4] 周志明. 深入理解Java虚拟机：JVM高级特性与最佳实践[M]. 机械工业出版社,2015 .

[5] Brian Goe. Java并发编程实战[CP]. 机械工业出版社华章公司，2015:1-6.

[6] 许令波.深入分析Java Web技术内幕 [M].北京：电子工业出版社,2015:8-11.

[7] 宋智军,邱仲潘. JSP从入门到精通（第二版）[M].北京：电子工业出版社,2017:3-8.

[8] 吉根林,顾韵华. Web程序设计（第3版）[M].北京：电子工业出版社, 2016:3-6.

[9] 蔺华. JavaScript 高级程序设计[CP].北京：电子工业出版社,2018:5-11.

[10] Joshua Bloch. Effective Java中文版（第二版） [M]. 机械工业出版社, 2015:3-10.

[11] 耿祥义.张跃平. java设计模式[M]. 清华大学出版社, 2017:2-3.

[12] 张海藩.软件工程导论（第5版）[M].北京：清华大学出版社,2018:4-7.

[13] 谢希仁.计算机网络[M].北京： 电子工业出版社, 2015.

[14]鲍耀翔. 基于ASP.NET的餐厅在线订餐系统的设计和实现[D].浙江工业大学,2017:22,36.

[15]邓尧. 柏联酒店网络在线订餐系统的设计与实现[D].电子科技大学,2017:14,27.