

# 陕西科技大学

## 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： 201406060128

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 14 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 .....	I
ABSTRACT .....	II
1 绪论 .....	1
1.1 课题研究背景及意义 .....	1
1.2 国内外发展状况 .....	1
1.3 本文研究内容 .....	1
1.3.1 web 前端语言 .....	2
1.3.2 Java Web 整合开发 .....	2
1.3.3 数据库相关研究 .....	2
2 基础理论及相关技术 .....	3
2.1 SSM 框架 .....	3
2.2 Servlet 和 JSP .....	4
2.3 MySQL 数据库 .....	4
2.4 JDK 和 Tomcat 服务器 .....	5
2.5 IntelliJ IDEA .....	6
3 系统分析及设计 .....	7
3.1 可行性分析 .....	7
3.1.1 技术可行性 .....	7
3.1.2 经济可行性 .....	7
3.2 需求分析 .....	7
3.3 系统总体设计 .....	8
3.3.1 系统功能结构 .....	8
3.4 数据库设计与实现 .....	9
3.4.1 数据库概念设计 .....	9
3.4.2 数据库表设计 .....	12
3.4.3 数据库逻辑结构设计 .....	18
4 系统详细设计 .....	19
4.1 系统环境配置 .....	19
4.1.1 软件环境 .....	19
4.1.2 硬件环境 .....	19
4.2 系统框架实现 .....	19
4.3 系统公共模块设计 .....	21
4.3.1 数据库连接和事务管理器 .....	21

4.3.2 系统安全控制 .....	23
4.4 系统主要模块功能的实现 .....	23
4.4.1 登录注册功能 .....	23
4.4.2 商品展示功能 .....	26
4.4.3 购物车的设计与实现 .....	29
4.4.4 用户订餐的设计与实现 .....	32
4.4.5 订单操作的设计与实现 .....	34
4.4.6 用户评论的设计与实现 .....	34
4.4.7 管理员商品管理的设计与实现 .....	36
4.4.8 管理员订单状态管理的设计与实现 .....	40
4.5 总结 .....	42
5 系统测试与性能分析 .....	43
5.1 测试简介 .....	43
5.2 功能测试 .....	43
5.2.1 登录功能测试 .....	43
5.2.2 注册功能测试 .....	44
6 总 结 .....	46
致 谢 .....	47
参考文献 .....	48

## 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。提升了服务水平，让消费者享受到更高质量的服务<sup>[1]</sup>。

### 1.2 国内外发展状况

在现在，中国餐饮 O2O 首要模式可以分为四大种类：团购类，以美团、Baidu 糯米、大众点评为代表，全部引进美国 Groupon 模式，进而全面接入本地餐户，团购模式由于起步较早，是目前餐饮 O2O 的主要模式；对于点评类，目前是以大众点评为代表，通过很长时间的积累，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、美团外卖为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按照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物安全保障，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

首先，一个完备的在线订餐系统的实现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。而对于后台数据库则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<sup>[2]</sup>。

### 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<sup>[3]</sup>。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<sup>[4]</sup>。

### 1.3.3 数据库相关研究

数据库的选择因为此系统是全部用户使用的网站系统，是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，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、控制层、业务逻辑层、数据持久层，以帮助开发人员在短期内搭建结构清晰、可复用性好、维护方便的 Web 应用程序。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。

此系统基于 SSM 框架的基本业务流程是：在表示层中，首先通过 JSP 页面实现交互界面，负责接收请求(Request)和传送响应(Response)，然后 SpringMVC 通过 dispatchServlet 将接收到的 Request 分派给注解配置的路径（requestMapping）的 Controller 层中的函数中进行处理。在业务层中，对象生命周期管理组件的 Spring IoC 容器向业务逻辑层提供数据模型(Model)组件以及与操作数据模型的对象数据处理(DAO)组件共同处理完成业务逻辑，并切 SpringAOP 还提供了事务处理用来保证数据操作的完整性、缓冲池容器组件用以节省开发时间和提升系统性能，业务处理层将处理结果返回给 Controller 层，再通过 DispatchServlet 传递到相应的页面。在 Mybatis 管理的数据持久层中，则是通过 Model 对象和数据库进行交互，DAO 层只提供接口，其实现运行期通过与 XML 文件中的 SQL 语句映射完成，开发者只需要关心 SQL 语句的填写。

对上述开发模型的使用，在 SpringMVC 中实现了视图（view）、控制器（Controller）与模型（Model）的彻底分离，在 Spring 中将所有对象的管理，实现了业务逻辑层与持久层的分离，Mybatis 使持久层消失，只剩下 SQL 编写。

这样在前端无论如何变化，模型层只需很少的改动，并且对数据库操作的变更也不会让前端感知到，使系统中底层模块的可复用性大大提高。而且由于不同层之间耦合度小，有利于团队成员并行工作，使开发效率也有所提高。

## 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 程序的性能和系统的限制。其主要功能在于交互式地浏览和修改网站数据，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 协议的 Web 服务器<sup>[5]</sup>。

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 更出色，除了系统之外，代码不用做任何更改就可以完成。

（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进行开发，在任意环境中进行系统的部署，在任意环境中进行扩展。相比 ASP/PHP 的系统 and 平台局限性是显而易见的。

（c）强大的可伸缩性。从只有一个单独的 Jar 文件就可以运行 Servlet/JSP，到由多台服务器进行集群部署和负载均衡，到多台 Application 进行并行事务处理，消息处理，单台服务器到 n 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，而且许多工具组件都是开源免费的并且可以随时下载更改，并且其中许多工具已经运行在我们使用中的软件当中。

## 2.3 MySQL 数据库

关系型数据库中 MySQL 现下最流行关系型数据库之一，由瑞典公司开发，目前是 Oracle 旗下的一款数据库产品。MySQL 聚集了一切数据，种类非常全面，并且集成度高，该系统数据的稳定性和安全性都经过业界检验，可以广泛的应用于 WEB 应用的数据管理，其标准的 SQL 语法使得可操作性非常高，并且不需要额外的学习，管理程序大大简化，在程序中使用的难度也有效降低。

MySQL 关系数据库将数据保存在不同的表中，每张表中的数据都会进行分页，根据整页数据查询的速度就会大大提高，并且提供了丰富的数据类型，提高了灵活性。

MySQL 所使用的 SQL 语言是大多数数据库使用的最常用的标准结构化查询语言。MySQL 软件采用了双授权政策，分为社区版和商业版，其体积小、速度快、总体使用成本低，其是开放源码可以使商业公司有能力和进行二次开发，所以大部分网站的开发都选择 MySQL 作为后台网站数据库<sup>[6]</sup>。

MySQL 是一种开放源代码的关系型数据库管理系统（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，所以在嵌入式开发或者是群集系统的部署中，还有在基于 Web 的应用程序领域，MySQL 数据库往往是开发人员的第一选择。

#### （b）简单性

MySQL 易学易用。在开发方面和支持方面，现在有许多好用的工具可以选择。对于新手开发者也利用相关的工具就可以轻松地使用 MySQL 数据库进行日常的开发。那么对于一个有经验的 Windows 管理者也可以轻松部署并开始学习它，而你无需投入一分钱来了解这个数据库。MySQL 进行调优就可以运行的更快速，MySQL 中没有多余的功能来拖累 CPU 或占用内存。

#### （c）低成本

开源版本的数据库中 MySQL 是对硬件的较低要求是其最大的优势之一，MySQL 易学、易用、易部署、易管理和易维护。一个现存的业务，可以很容易的移植到 MySQL 上。MySQL 部署迅速，所以移植过程不会导致生产中断。而且其相对容易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

MySQL 由于其广大的社区群体，开发了如此众多的额外功能，诸如存储引擎等，这个特点使得 MySQL 拥有相当大的灵活性，可以根据你当前的系统的需要来进行调整对应的参数来更好的适应服务。

## 2.4 JDK 和 Tomcat 服务器

开放源代码的 Web 应用服务器中 Tomcat 服务器是 Apache 软件基金会（Apache Software Foundation）的 Jakarta 项目中的一个顶级项目，由 Apache 基金会和其他一些公司及开源社区共同维护开发，是一个 Web 容器，可以运行 Servlet/JSP。由于 Tomcat 安全可靠、性能不错，而且免费开源，因而较多使用在 Web 应用服务器中，深受 Java 爱好者的喜爱并得到了部分软件开发商的认可。

JDK（Java Development Kit）是 Sun 公司针对 Java 平台开发的基础核心类库，具

有性能不错，安全稳定的特点。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 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 平台的 Jrocket 在服务端运行效率也要比 Sun JDK 好很多。从 Sun 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，同时 Java 虚拟机也在不断的改进，其运行效率得到了非常大的提高<sup>[7]</sup>。

## 2.5 IntelliJ IDEA

IDEA 全称 IntelliJ IDEA，是一个集成开发环境，可以进行 java 应用程序的开发，其在智能代码助手、代码自动提示、重构、J2EE 支持、JUnit、CVS 整合、代码审查方面尤其突出。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 Maven，Spring4，Mybatis 等组件。

## 3 系统分析及设计

### 3.1 可行性分析

#### 3.1.1 技术可行性

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，并得到广泛的应用。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相应的解决方案。

#### 3.1.2 经济可行性

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。大大减少了出行，是一种经济环保的生活方式；对商家来说，减少了人力物力，更少的生活垃圾污染，对环境友好，是可持续发展的模式。

### 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，然后生成订单，用户可以查看购买的商品，可以对已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。最后，用户确认订购，填写送货地址信息，收货人联系方式，支付方式（仅线下支付）。最后用户点击提交，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，可以让用户看见订单的配送进度<sup>[8]</sup>。

### 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。用户在登录页面进行登录时，后台会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，以供用户选择所需产品。在用户选择购买某个菜品时，cookie会将其价格、数量记录到购物车中。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。确认完毕后点击提交，生成订单数据。后台主要是对用户管理、产品管理和订单管理，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增删改查。数据库存在的产品则可以在前台的首页展示供用户选择，产品可以在后台直接编辑、修改、删除。订单管理部分主要是订单信息的展示。订单管理分为准备中和已完成的订单，准备中的订单是未送出的订单，送出的订单将变为派送中的订单。用户查看订单时状态会显示卖家已送出。系统设计功能结构图如图 3-1 所示。

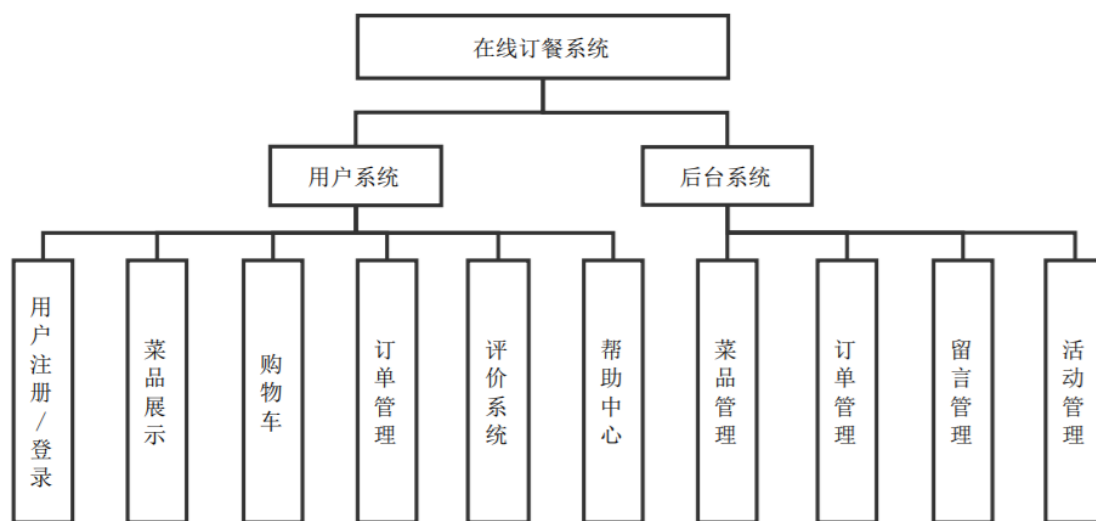


图 3-1 系统设计功能结构图

#### 3.3.1 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

本系统被分成多个模块，每个模块需要有一定的数据模型，各模块之间的联系也需要有相应的关系来支撑。下面即为本系统中所用到的数据模型概念设计。

(a) 用户信息表

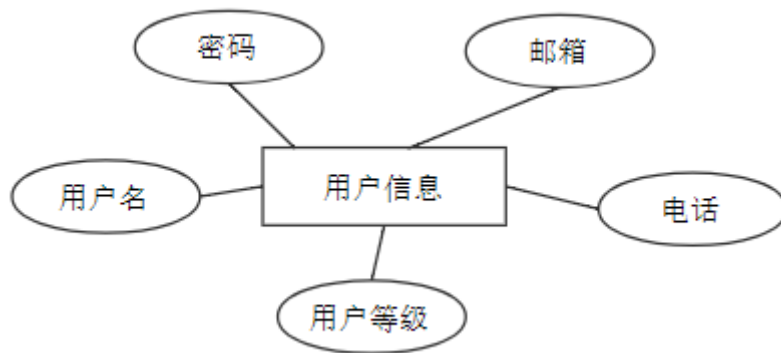


图 3-2 用户信息表实体

(b) 用户等级表



图 3-3 用户等级表实体

(c) 企业信息表

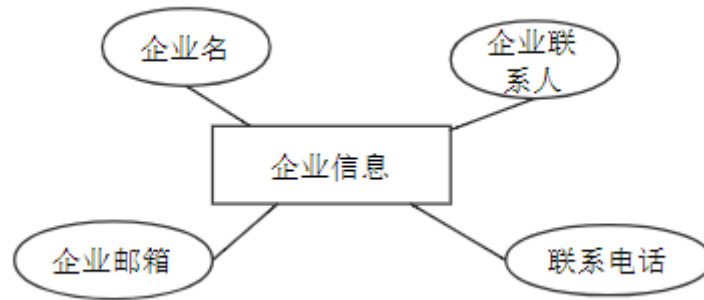


图 3-4 企业信息表实体

(d) 用户展示的商家表



图 3-5 用户展示的商家表实体

(e) 商家配送费用及安排表

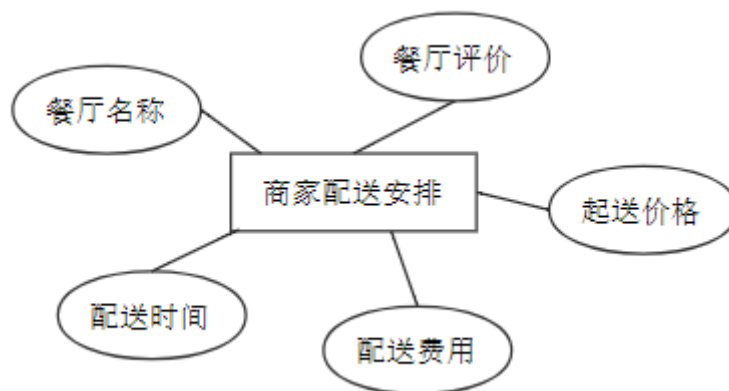


图 3-6 商家配送费用及安排表实体



(f) 菜品分类表

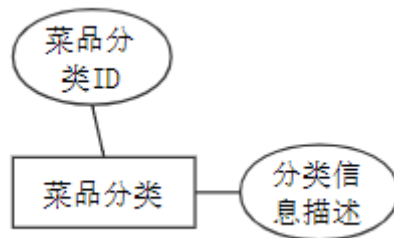


图 3-7 菜品分类表实体

(g) 菜品表



图 3-8 菜品表实体

(h) 订单表



图 3-8 订单表实体

(i) 用户与订单关系表

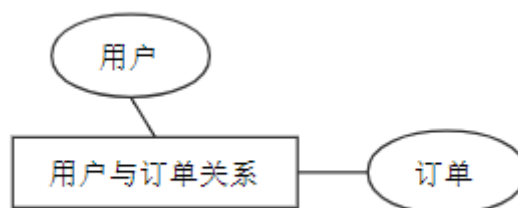


图 3-9 用户与订单关系表实体

(j) 用户配送地址表

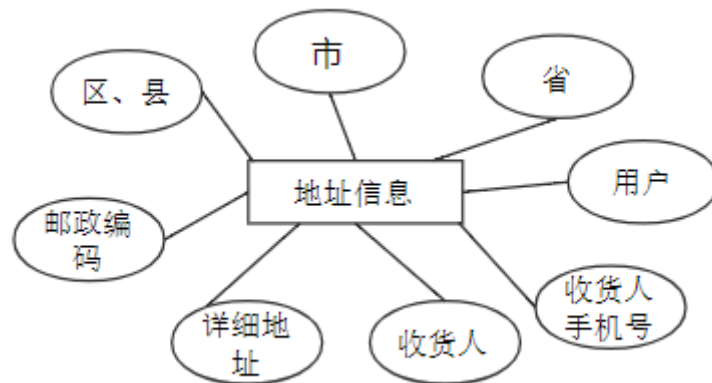


图 3-10 用户配送地址表实体

(k) 订单留言表



图 3-11 订单留言表实体

(l) 订单评价表

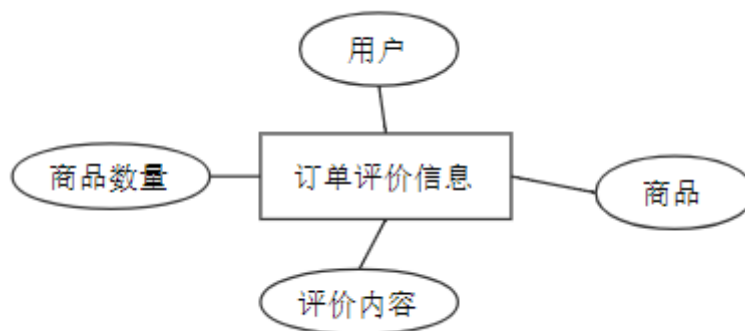


图 3-12 订单评价表实体

### 3.4.2 数据库表设计

(a) 用户信息表

表 3-1 用户信息表

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

## (b) 用户等级表

表 3-2 用户等级表

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

## (c) 企业信息表

表 3-3 企业信息表

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_LINKMAN	VARCHAR(20)	No	企业联系人
ENTERPRISE_PHONE_NUM	VARCHAR(20)	No	企业联系电话

续表 3-3

列名	数据类型	可为空	说明
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

(d) 用户展示的商家表

表 3-4 用户展示的商家表

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

(e) 商家配送费用及安排表

表 3-5 商家配送费用及安排表

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

## (f) 菜品分类表

表 3-6 菜品分类表

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_DESCRIPTION	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

## (g) 菜品表

表 3-7 菜品表

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

续表 3-7

列名	数据类型	可为空	说明
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

(h) 订单表

表 3-8 订单表

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_CONTE NT	VARCHAR(500)	No	订单内容
ORDER_PRICE	DECIMAL(5,2)	No	订单价格
DISPATCH_ADD RESS	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

(i) 用户与订单关系表

表 3-9 用户与订单关系表

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

(j) 用户配送地址表

表 3-10 用户地址表

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

## (k) 订单留言表

表 3-11 订单留言表

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

## (l) 订单评价表

表 3-12 订单评价表

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

续表 3-12

列名	数据类型	可为空	说明
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企业、用户、菜品、订单等主要的数据库对象，这些数据库对象是数据库表结构的基本模型，最终的数据模型都要实施到数据库中，形成整体的数据库结构。在线订餐系统的数据库模型如图 3-13 所示。

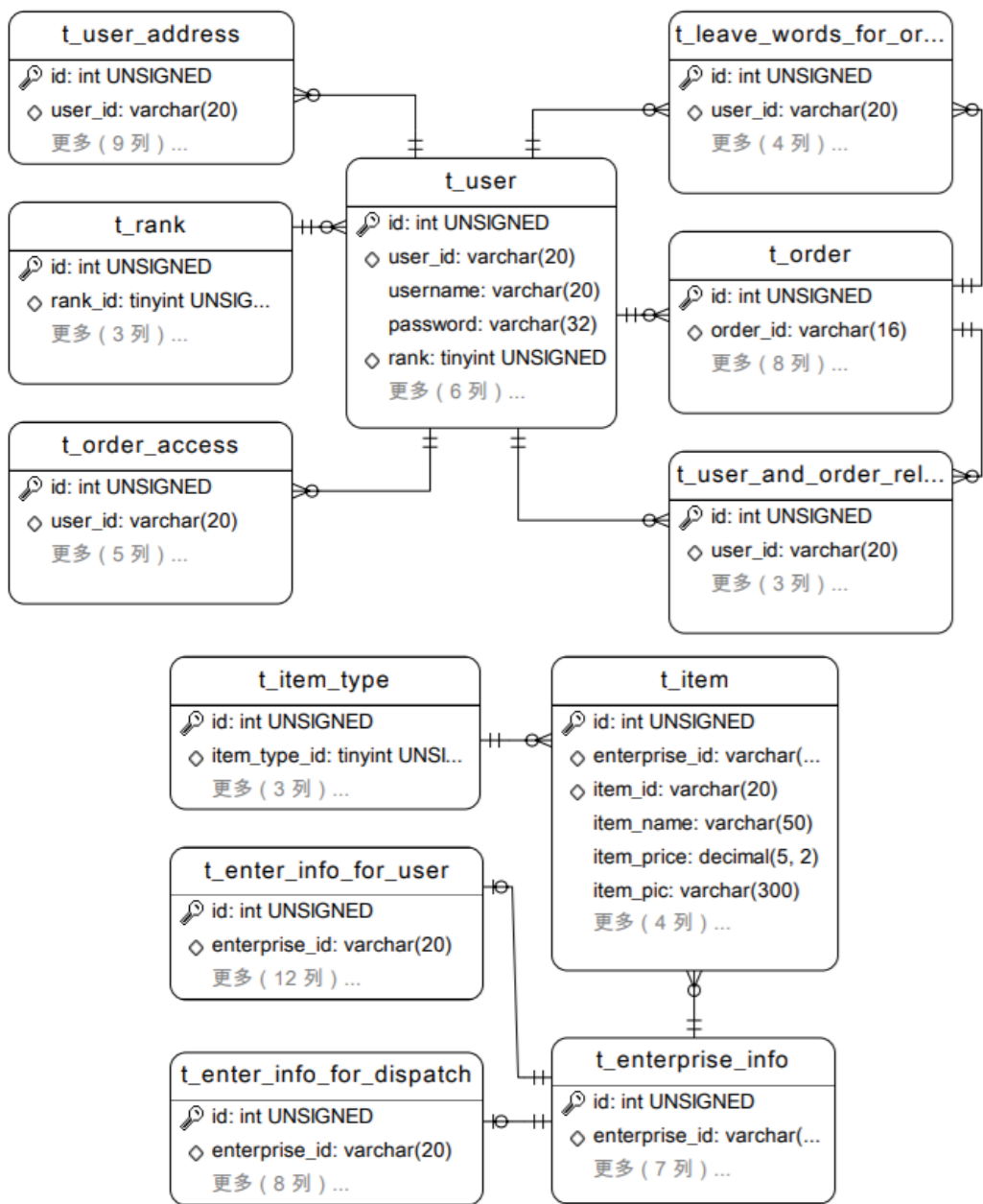


图 3-13 数据库逻辑结构



## 4 系统详细设计

### 4.1 系统环境配置

#### 4.1.1 软件环境

本系统服务器采用 Apache 下的开源项目 Tomcat7.0 作为，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：

- (a) Java 基础开发工具包：jdk-8u130-windows-x64。JDK 环境变量的配置如下：  
变量名 JAVA\_HOME：C:\Program Files\Java\jdk1.8.0\_45（为 JDK 的安，装目录）  
变量名 PATH：%JAVA\_HOME%\bin; %JAVA\_HOME%\jre\bin  
变量名  
CLASSPATH：.;%JAVA\_HOME%\lib\tools.jar;%JAVA\_HOME%\lib\dt.jar
- (b) Web 服务器采用 Tomcat-9.0.1。Tomcat-9.0.1 环境变量配置如下：  
变量名 CATALINA\_HOME：F:\apache-tomcat-9.0.1-windows-x64\apache-tomcat-9.0.1（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;
- (c) 数据库服务器采用 MYSQL5.7。
- (d) 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，软件会自动下载相应版本的 jar 包。
- (e) 使用 Dreamwear8.0，EditPlus 等软件来美化页面。

#### 4.1.2 硬件环境

- (a) 计算机型号 Hasee。
- (b) CPU 处理器，Intel(R) Core(TM) i7-4710MQ CPU @ 2.50GHz。
- (c) 磁盘容量，硬盘 1T, 内存 8.00GB。
- (d) 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
----	-----	----

续表 4-1

类名	映射表	说明
TEnterpriseInfo	t_enterprise_info	企业信息表
TUser	t_user	用户表
TItem	t_item	菜品表
TEnterInfoForDispatch	t_enter_info_for_dispatch	商家配送安排表
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	无	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 Controller 层

类名	说明
AdminOrderController	管理员订单操作
AdminUserController	管理员用户操作

续表 4-3

类名	说明
EnterpriseInfoController	企业信息操作
CartController	购物车操作
DetailspController	菜品详情操作
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：

#### (a) 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>
```

#### (b) 配置会话工厂 SessionFactory，并将数据源 dataSource 通过属性注入带 Spring 容器中，并配置对应的 SQL 文件的路径

```
<bean id="sqlSessionFactory"
    class="org.mybatis.spring.SqlSessionFactoryBean">
    <property name="dataSource" ref="dataSource"/>
    <property name="configLocation"
        value="classpath:mybatis/mybatisConfig.xml"/>
</bean>
```

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

```
    </bean>
```

(c) 配置 DAO 接口注入路径

```
    <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
        <property name="basePackage" value="com.sust.dao"/>
        <property name="sqlSessionFactoryBeanName"
            value="sqlSessionFactory"/>
    </bean>
```

```
    </bean>
```

(d) 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>
```

(e) 配置事务属性，增删改操作都需要是原子操作

```
    <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>
```

- (f) 配置事务切入点，再把事务属性和事务切入点关联起来，形成一个完整的事务配置

```
<aop:config>
    <aop:advisor advice-ref="txAdvice"
        pointcut="execution(* com.sust.service..*(..))" />
</aop:config>
```

#### 4.3.2 系统安全控制

系统除了首页不存在访问控制之外，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并且一致，若输入无误，则可成功进行浏览其他页面信息和进行下单等操作，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，且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和密码才可登录成功，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，其实现代码是在程序运行时动态生成的。

- (a) 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；当在数据库中查询不到相应的用户名和密码时，返回到错误页面并提示详情。

#### (b) 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。

#### (c) 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<sup>[10]</sup>，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>
```

(d) 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



The image shows a login form with a light gray header. Below the header, there are two input fields: the first is labeled '账号:' (Account) and the second is labeled '密码:' (Password). Below these fields are two buttons: '登录' (Login) and '注册' (Register).

图 4-1 系统登录页面



The image shows a registration form with a light gray header. Below the header, there are five input fields: '用户名:' (Username), '密码:' (Password), '再次确认:' (Confirm Password), '电子邮件:' (Email), and '手机号码:' (Mobile Number). Below these fields is a single orange button labeled '注册' (Register).

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。如下图4-3 的商品列表展示，4-4 的商品详情查看。



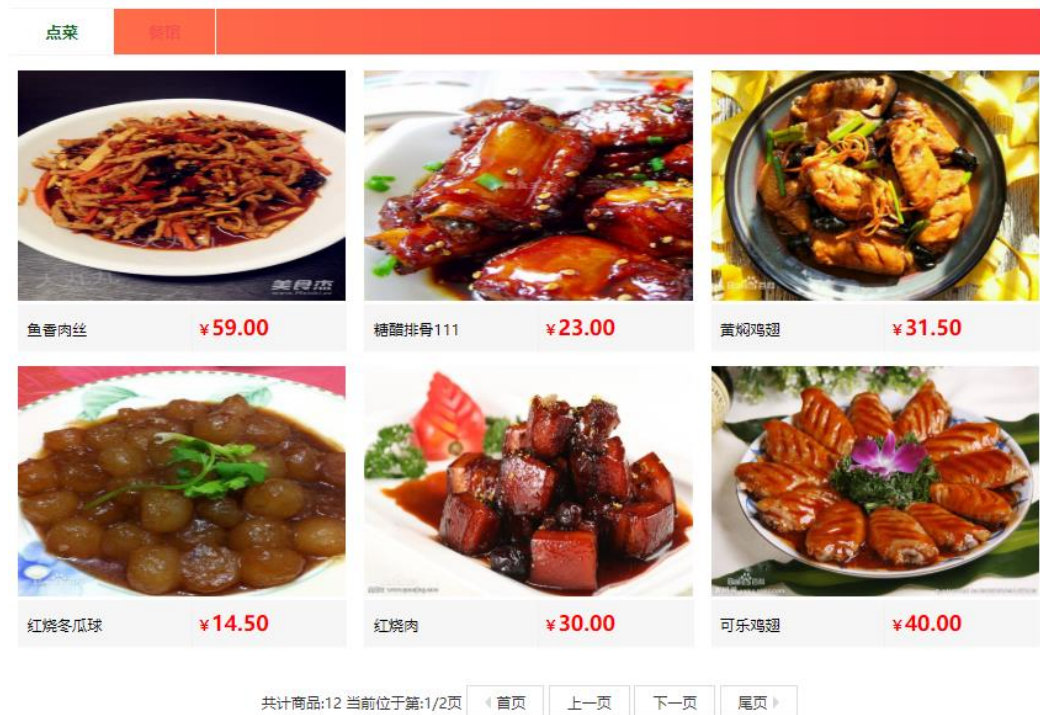


图 4-3 首页商品展示



图 4-4 商品详情页面

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>标签在页面上循环显示即可。

//确定当前页数的合法性

```
public void validCurrentPageNum(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：

```
{
    //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);
```

然后 Mybatis 就会在实际的查询之前先完成一次数量上的查询，然后再进行实际数据的查询操作，最终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 方法就不一一列举在此
}
```

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，并且显示加入购物车的商品，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 4-5 所示



图 4-5 购物车

主要代码实现如下：

//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 4-6 所示：

收货地址:

浙江省 杭州市 余杭区 航海路1588号 (孙先生收收) 183092\*\*\*73

浙江省 杭州市 余杭区 航海路1588号 (孙先生收收) 183092\*\*\*73

浙江省 宁波市 A区 bcd路1234号 (李先生收) 183092\*\*\*73

使用新地址

配送方式:

配送方式	运费	说明
送货上门	¥ 6	根据配送距离和每个订单来计算

支付方式:

线下支付

商品清单

名称	数量	价格	小计
红烧肉	1	¥ 30.00	¥ 30.00

订单附言: 多加米饭

配送费用: ¥ 6+ 商品费用: ¥ 30  
合计: **¥ 36**

提交订单

图 4-6 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 4-7 所示：



图 4-7 订单生成成功界面

用户订单信息确认界面主要实现代码解释如下，Controller 层，得到订单数据和用户的地址信息：

```
@RequestMapping(value = "/confirm_order")
public String confirm_order(HttpServletRequest request,
                           HttpServletResponse response,
                           Model model) {

    String cookieValue =
        CookieUtils.getCookieValue(request, CookieConstant.SHOPPING_CART_NAME);
    //获取商品信息
    List<ItemDetailDto> itemDetailDtos = null;
    if(cookieValue != null) {
        itemDetailDtos = getItemFromCart(cookieValue);
    }
    BigDecimal total = null;
    double val = 0.0;
    //计算总价格
    for (ItemDetailDto itemDetailDto : itemDetailDtos) {
        val+=itemDetailDto.getItemPrice().doubleValue()*
            itemDetailDto.getCnt();
    }
    total = new BigDecimal(val);
    //获取地址信息
    String userId =CookieUtils.
        getCookieValue(request,TUserConstant.USER_COOKIE_NAME);
    List<TUserAddress> userAddressList =
        userService.queryAddressByUserId(userId);
```

```
//获取配送价格信息
TenterInfoForDispatch infoForDispatch=
    enterInfoForDispatchService.
        queryEnterInfoForDispatchByenterpriseId(
            itemDetailDtos.get(0).getEnterpriseId());
Short dispatchPrice = infoForDispatch.getDispatchPrice();
model.addAttribute("userAddressList", userAddressList);
model.addAttribute("total", total);
model.addAttribute("itemDetailDtos", itemDetailDtos);
model.addAttribute("dispatchPrice",dispatchPrice);
return "front/confirm_order";
}
```

#### 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。

用户中心首页	订单编号	下单时间	收件人	订单总金额	订单状态	操作
我的订单	9999124738951905	Fri May 25 15:18:05 CST 2018	李先生	36.00	正在准备中	取消订单
收货地址	9952186798535557	Sat May 12 16:26:38 CST 2018	孙先生收	37.50	订单已送达	查看详情
我的留言	0000000000000005	Wed May 09 13:38:55 CST 2018	李先生	94.50	订单已送达	查看详情
我的优惠券	0000000000000001	Wed May 09 08:39:36 CST 2018	李先生	123.99	正在派送中	查看详情
我的收藏						
账户管理						
安全退出						

« 首页 1 2 最后一页 »

图 4-8 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 4-9 所示：



订单编号: 9999124738951905

订单信息:

订单产品	订购数量	单价	小计
红烧肉	1	¥ 30.00	¥ 30.00

配送费用: ¥6

共计金额: ¥ 36.00

收件地址: 浙江省 宁波市 A区 bcd路1234号

收件人: 李先生

电话: 183092\*\*\*73

来给个评价吧: 来说说你的感受

评价

图 4-9 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.queryOrderDetailById(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 4-10 所示:

ID	菜品编号	菜品名称	菜品价格	修改日期	操作
9	00000000000000000001	鱼香肉丝	59.00	Wed May 02 17:50:31 CST 2018	<a href="#">编辑</a> <a href="#">删除</a>
10	00000000000000000011	糖醋排骨111	23.00	Mon May 21 16:00:08 CST 2018	<a href="#">编辑</a> <a href="#">删除</a>
11	00000000000000000021	黄焖鸡翅	31.50	Sat Apr 21 22:29:22 CST 2018	<a href="#">编辑</a> <a href="#">删除</a>
12	00000000000000000031	红烧冬瓜球	14.50	Sat Apr 21 22:29:42 CST 2018	<a href="#">编辑</a> <a href="#">删除</a>
17	00000000000000000100	糖醋排骨2	23.00	Mon May 21 14:14:44 CST 2018	<a href="#">编辑</a> <a href="#">删除</a>
18	00000000000000000101	糖醋排骨3	23.00	Mon May 21 14:14:52 CST 2018	<a href="#">编辑</a> <a href="#">删除</a>
19	00000000000000000102	糖醋排骨4	23.00	Mon May 21 14:14:58 CST 2018	<a href="#">编辑</a> <a href="#">删除</a>
20	999623999023509504	肉蛋	3.00	Thu May 24 20:11:48 CST 2018	<a href="#">编辑</a> <a href="#">删除</a>

图 4-10 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 4-11 所示：

菜品 / 详情/修改

菜品详情信息

菜品编号	999623999023509504
菜品名	卤蛋
菜品价格	3.00
菜品描述	卤蛋 (Marinated Egg), 又名卤水蛋。是用各种调料或肉汁加工成的熟制蛋。卤蛋是熟食店经营禽蛋品中的一个大众化食品, 普遍受到人民的欢迎
所属类型	自制食材
菜品配图	

提交保存 放弃保存

图 4-11 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。如下图 4-12 所示。

图 4-12 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 4-13,4-14,4-15 所示。

订单列表 / 准备中

ID	订单编号	订单价格	订购人	修改日期	操作
25	9999124738951905	36.00	00000000000000000001	Fri May 25 15:18:05 CST 2018	<a href="#">去派送</a>

共 1 条记录

注: .....

图 4-13 管理员操作准备中订单

订单列表 / 派送中

ID	订单编号	订单价格	订购人	修改日期	操作
25	9999124738951905	36.00	00000000000000000001	Fri May 25 16:56:16 CST 2018	<a href="#">派送完成</a>

共 1 条记录

注: .....

图 4-14 管理员操作派送中订单

已完成 / 订单列表

ID	订单编号	订单价格	订购人	修改日期	操作
1	0000000000000001	123.99	00000000000000000001	Fri May 25 16:56:27 CST 2018	<a href="#">查看详情</a>
14	0000000000000005	94.50	00000000000000000001	Wed May 09 16:23:15 CST 2018	<a href="#">查看详情</a>
16	0000000000000003	123.99	00000000000000000003	Mon May 21 15:01:30 CST 2018	<a href="#">查看详情</a>
17	0000000000000004	123.99	00000000000000000004	Wed May 09 16:20:34 CST 2018	<a href="#">查看详情</a>
21	9951401106033008	52.00	00000000000000000002	Mon May 21 15:02:07 CST 2018	<a href="#">查看详情</a>
22	9952186798535557	37.50	00000000000000000001	Mon May 21 15:02:16 CST 2018	<a href="#">查看详情</a>
25	9999124738951905	36.00	00000000000000000001	Fri May 25 16:59:06 CST 2018	<a href="#">查看详情</a>

共 7 条记录

图 4-15 管理员查看已完成订单

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;
}
```

## 4.5 总结

整个系统只完成了最基本的功能，用户只是可以完成整体的流程操作，但是一些额外的功能还未完成，比如说异常控制，当后台出现不可预知的错误的时候，那么前台页面应该显示什么样的错误信息给用户，都是未考虑到的。编码的难点在于怎么样的逻辑是合理的，什么时候该做合适的操作，我觉得这是比较难掌握的部分。



## 5 系统测试与性能分析

### 5.1 测试简介

软件测试是为了及时发现应用程序中的存在的错误和不合理的地方而执行程序的过程，在软件开发周期中是非常重要的步骤，一个好的测试方案有很大的可能发现尚未发现的错误。正确理解测试的目的是为了找出软件中存在的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出软件中留存的未发现的错误。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<sup>[15]</sup>。一般来说对系统测试的方法有两种：黑盒测试和白盒测试。其中，黑盒测试是主要从用户的角度去看待整个程序，整个程序就是一个黑盒，从这个黑子外部输入数据到黑盒，再得出结果。全部测试都是建立在用户要求的基础上，检查程序能否满足需求说明书中的要求，对应地该测试不需要了解系统的结构和具体实现代码。白盒测试主要是可以了解程序系统内部的具体情况，还可以查阅具体的实现源码帮助你完成测试，使用者能全面了解用例的内部代码，容易找出系统中隐藏的缺陷，有效的提升代码的鲁棒性，这种方式也有一定的缺陷，系统越大，代码中工作状态的路径就越复杂，不容易全路径覆盖，深入细节就有可能忽略部分的用户需求。通过比较，黑盒测试就很能很好的满足在线订餐系统的检测，所以在检测网上订餐系统的各种功能的属性的时候，就是使用黑盒测试法，检测每一项功能是否都可以正常的使用，只从用户角度来看，是否正确的完成了需求说明中功能，对于需要输入的项，需要检查程序能否适当地接收输入数据，并且保证输出的各项结果都是合理的。所以主要是针对应用程序的页面及提供的交互功能来进行验证，黑盒测试主要能验证以下几种错误：系统界面出现差错；系统功能失效、缺失、错误；数据库访问出错；性能太差导致用户体验极差。通常情况下，这种方法的使用要将系统中错误都检测出来，需要考虑每种输入的边界情况，需要对输入输出都实施检测，为此只对系统相关的功能做简明扼要的测试。

### 5.2 功能测试

#### 5.2.1 登录功能测试

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 HTML5 的函数检测出来，不需要提交到后台进行处理，节省系统的开销和往返传送数据的时间消耗。

登录测试用例的测试过程如图 5-1 所示。

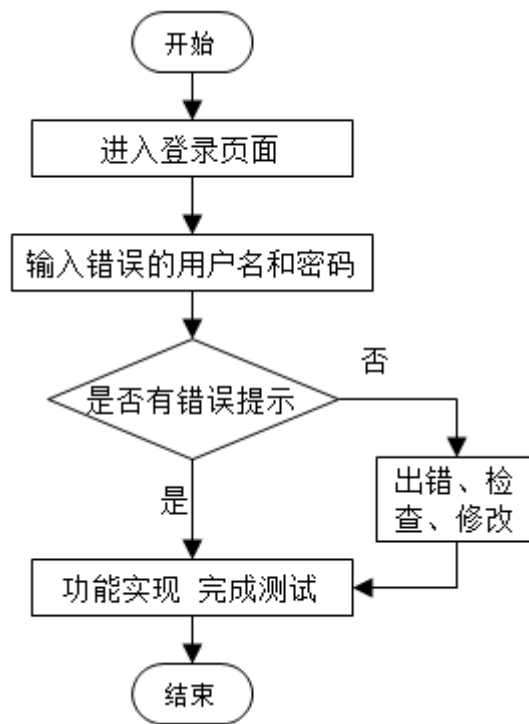


图 5-1 登录模块测试图

系统中，通过 login.html 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首页。输入错误的用户名或者密码之后在登录页面提示相应的错误信息。

### 5.2.2 注册功能测试

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。

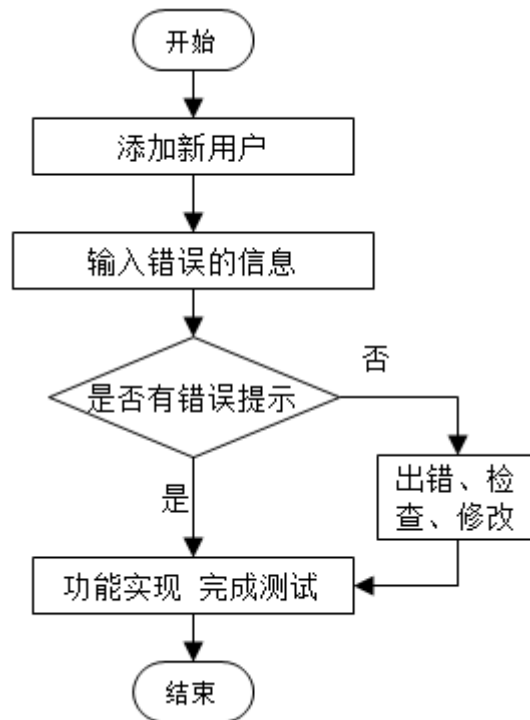


图 5-2 添加新用户测试图

## 6 总 结

在本次毕业设计中，收获了许多，也有许多不完善的地方。首先，在确定题目后，我觉得挺简单的，从用户角度看来，无非就是少数的步骤就可以完成整个流程，浏览、加入购物车、结算、查看订单等，但是在完成整个项目的需求分析后，结合自身所掌握的熟练度和所了解的多少，我感觉自己一个人的工作量是非常大的，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，我本身学过 Java 编程语言，对 SpringMVC,Mybatis3, Spring 有些许学习和了解，早期的 Servlet 开发有很多不足并且开发进度缓慢，所以选择成熟的 SSM 三大框架整合来完成这个项目。而一个完整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。所以决定先把后台的框架完成起来，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，遇到过很多问题，其实最重要的一点就是查找问题出现的原因，因为只有找到原因才能知道解决办法，很多时候不知道原因就只能一点点排查问题，去搜索引擎去搜索类似的问题，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，我意识到自己对项目的整体把控度不高，应该在最初设计系统的时候就把该有的功能都添加上去，页面中有哪些元素都需要提前设计好，而不是最后写到哪里再去设计，会浪费很多时间；并且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，毕业设计已经基本完成。通过本次的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。整个项目的设计编码都是由自己独立完成，或多或少对自己的技术和思维都有所锤炼。使自己对技术的掌握更加深入，对未来的学习和工作有着乐观的态度。通过对网站项目的设计与完成，让我对完成一个项目从最初的需求分析设计和系统框架的搭建更加熟悉，对自己掌握的技术要更加精通，对掌握不熟练的技术要有快速学习上手的能力。当然这些都归功于学校，指导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有用的知识来丰富自己，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，同时感谢学姐张珍珍对我的帮助，让我在诸多磕碰的技术道路上迈的更远。

最后，我要感谢陕西科技大学对我四年的培养，让我学到了在生活中，学习上，面对难题要有积极的态度，从问题中寻找解决它的办法。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] 徐孝凯.计算机专业毕业设计宝典[M].北京:中央广播电视大学出版社, 2005:36-58.
- [2] 蔺华.JavaScript 高级程序设计[M].北京:电子工业出版社,2018:32-110.
- [3] 许令波.深入分析 Java Web 技术内幕 [M].北京:电子工业出版社,2015:78-123.
- [4] Craig Walls. Spring 实战（第三版）[M].北京:人民邮电出版社, 2016,(01):17-22.
- [5] 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.
- [6] Tapio Lahdenmaki. 数据库索引设计与优化[M].北京:电子工业出版社,2016:42-84.
- [7] 周志明.深入理解 Java 虚拟机:JVM 高级特性与最佳实践[M].北京:机械工业出版社,2015:23-56.
- [8] 邓尧.柏联酒店网络在线订餐系统的设计与实现[D]. 成都:电子科技大学,2017:14-27.
- [9] Brian Goe. Java 并发编程实战[M].北京:机械工业出版社华章公司,2015:10-68.
- [10] Joshua Bloch. Effective Java 中文版（第二版）[M].北京:机械工业出版社, 2015:3-10.
- [11] 耿祥义,张跃平. java 设计模式[M].北京:清华大学出版社, 2017:16-95.
- [12] 张海藩.软件工程导论（第 5 版）[M].北京:清华大学出版社,2018:20-79.
- [13] 谢希仁.计算机网络[M].北京:电子工业出版社, 2015:20-98.
- [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:浙江工业大学,2017:22-36.
- [15] 吉根林,顾韵华. Web 程序设计（第 3 版）[M].北京:电子工业出版社, 2016:37-84.