摘　要

随着网络技术的迅猛发展，人类社会已经进入了一个信息化社会。电子商务指利用简单、快捷、低成本的电子通讯方式，买卖双方通过网络进行各种商贸活动的一种商业交易模式。以及人们消费观念和生活方式的改变，这种新型的商业模式逐渐融入了人们的生活中。通过企业的门户网站，人们可以足不出户的寻找自己所需的物品，通过对不同虚拟商城的访问，“顾客”可以找出性价比最高的商品，通过下订单，你可以很快得到自己想要的物品，这给顾客节省很多时间和精力;对于厂家来说，可以极大地降低库存风险，可根据客户需要，按需采购和生产，大大地提高了效益。

本课题是设计并且实现一个基于 web 技术的在线交易系统。系统主要以 JAVAEE 作为开发基础，使用了 Spring、Spring、MVC+MyBatis+MySql 等多种协议或技术，用 IDEA 作为开发工具,基本实现了网上交易系统应有的主要功能模块.包括:管理员的登录，管理和维护;用户注册、登录、注销，个人信息的查询、修改，商品管理，购物车管理，订单管理。该系统界面简单、操作方便，容易维护。

【关键词】数码商品管理系统；MySql；JAVAEE；Spring MVC

Abstract

With the rapid development of network technology, human society has entered an information society electronic commerce refers to the use of simple, fast, low-cost electronic communication, buyers and sellers through the network to carry out a variety of business activities of a business model as well as the change of people's consumption concept and lifestyle, this new business model has gradually integrated into people's life by visiting different virtual malls, "customers" can find out the most cost-effective goods. By placing orders, you can quickly get the goods you want, which saves customers a lot of time and energy for manufacturers, can greatly reduce the inventory risk, according to customer needs, purchase and production on demand, greatly improve the efficiency.

This topic is to design and implement an online trading system based on Web technology.The system is mainly based on JavaEE development, using Spring、Spring, MVC+MyBatis、MySQL and other protocols or technologies,with IDEA as a development tool,the basic realization of the online trading system should have the mainfunctional modules Includes administrator login management and maintenance User registration, login, logout, personal information query, modification, commodity management, shopping cart management, order management The system interface is simple, easy to operate, easy to maintain.

【Key words】 Digital commodity management system; MySql; JAVAEE; Spring

目　录

1 引言

1.1 选题背景

随着信息技术的不断发展，我们现在已经步入了一个信息化的时代，现在网络已经和我们生活紧密的来联系起来了，休闲，娱乐，学习，购物等等许多我们数之不尽的事情。随着信息化的不断提升， 今天一种新的购物方式已然出现就是网上购物，针对各类大型网站的管理系统的开发技术日渐成熟。从选购到下订单，再到付款，最后确认，都可以通过软件进行统一管理。

市场上对于在线购物系统的需求量增加，因此开发一个数码商品平台系统非常必要, 本系统主要利用 Java EE 和 Spring 框架技术进行开发，具有很强的安全性、可靠性和扩展性。其前台实现了从商品选购到订单提交整个业务流程，后台实现了对整个系统的管理，就整体功能而言，能满足基本的网上购物需求，能让用户感到便捷、快速，让企业管理、维护更简单。

1.2 技术可行性

随着信息化网络的发展，网络对人们的生活影响越来越大，网上购物也逐渐成为一种趋势，这对电子商场的发展起到了很好的推进作用。本文主要通过对电子商城的应用环境和主要实现功能进行分析，叙述了本系统的设计与实现过程。该系统是基于多层企业级应用标准 JAVA EE 技术开发的 WEB 应用，以典型的 MVC 模式架构为基础。

Spring 由 Rod Johnson 创建，它是为了解决企业应用开发的复杂性而创建的。Spring 使用基本的 JavaBean 来完成以前只可能由 EJB 完成的事情。然而，Spring 的用途不仅限于服务器端的开发。从简单性、可测试性和松耦合的角度而言，任何 Java 应用都可以从 Spring 中受益。简单来说，Spring 是一个轻量级的控制反转(IOC)和面向切面(AOP)的容器框架。

MyBatis 是一款优秀的持久层框架，它支持定制化 SQL、存储过程以及高级映射。避免了几乎所有的 JDBC 代码和手动设置参数以及获取结果集，可以使用简单的 XML 或注解来配置和映射原生信息，将接口和 Java 的 POJOs(Plain Ordinary Java Object,普通的 Java 对象)映射成数据库中的记录,性能优异,具有高度的灵活性,可优化性、易于维护以及简单易学等特点。

1.2.1 开发工具介绍

主要开发工具为 IDEA 版本、Mysql8.0、Tomcat9.0
主要技术包括：Spring、Spring MVC、MyBatis

1.2.2 系统技术介绍

系统基于 B/S 模式开发的 JAVA EE 多层体系结构 WEB 应用。主要分为以下几层：视图层、service、DAO 层、持久层。业务逻辑层和 DAO 层都通过接口与其它层进行连接从而减小了各层之间的耦合度，实现高内聚、低耦合的思想。

1.3 开发目的

现今的互联网已经出现了很多在线购物系统，其中比较有名气的、影响力较大的有京东等。本项目致力于开发出一款简单、高效、用户体验高的网上在线购物系统，方便广大的购物爱好者浏览、购买商品、提供用户体验、方便人们的日常生活。

2 系统设计

本系统使用 SSM 框架实现各个模块,web 服务器使用 Tomcat9.0,数据库使用的是 MYSQL,集成开发环境为 IDEA

2.1 功能需求

本项目数码商品系统是一个销售和购物的完美结合，有对应的商品管理、订单管理、购物车管理等,数码商品系统分为两个子系统:后台管理系统和前台展示系统

2.2 系统模块划分

2.2.1 后台管理子系统

管理员登录进入后台管理主页面(main.jsp),对商品及商品类型、注册用户、用户的订单以及网站公告管理,后台管理子系统的模块划分如图 2.1 所示。

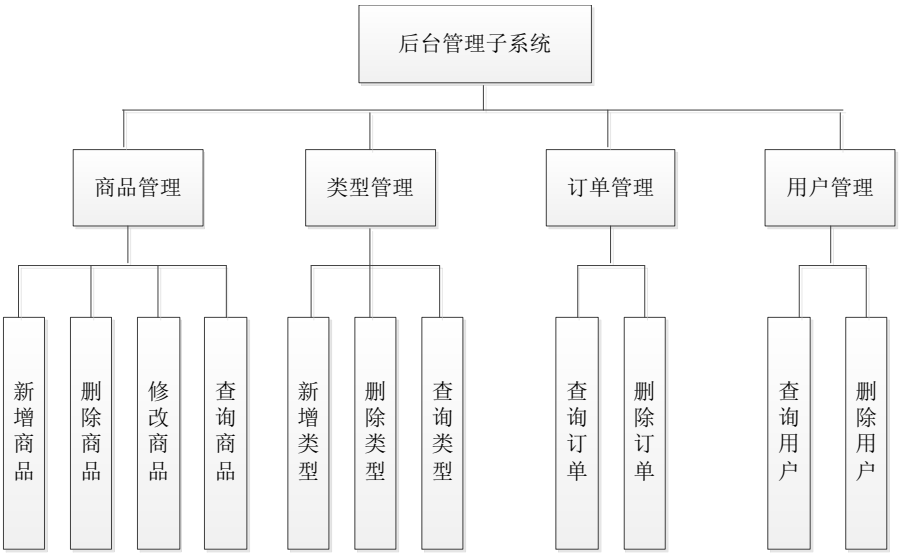

图 2.1 后台管理子系统的模块划分图

2.2.2 数码商品子系统

非注册用户只能浏览商品,不能购买商品、查看购物车和查看用户中心。成功登录的用户可以完成所有的功能,包括购买商品、支付等功能,数码商品子系统的模块划分如图 2.2 所示。

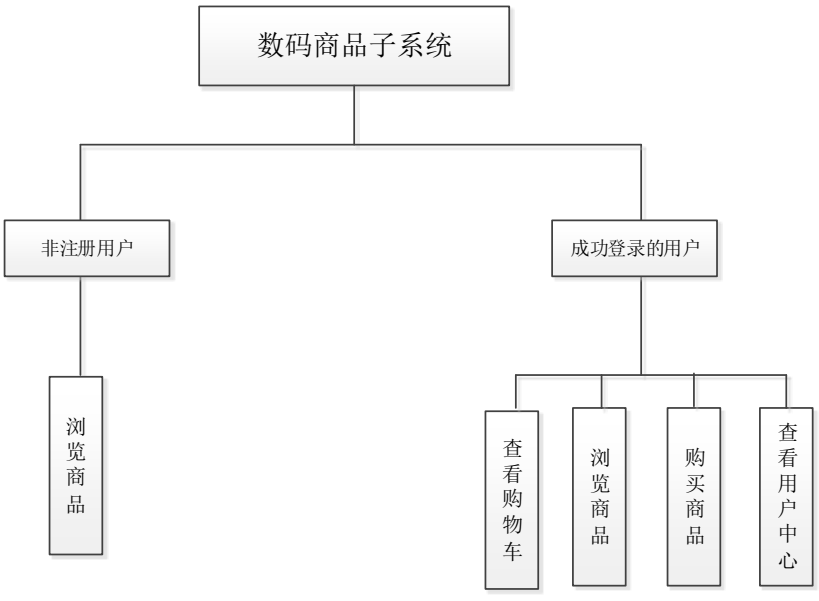

图 2.2 数码商品子系统图

2.3 系统详细设计

为了清晰的表达系统的业务功能模块，下面给出购物模块和后台管理模块。

2.3.1 后台管理模块

管理员对前台功能的控制如图 2.3 所示。

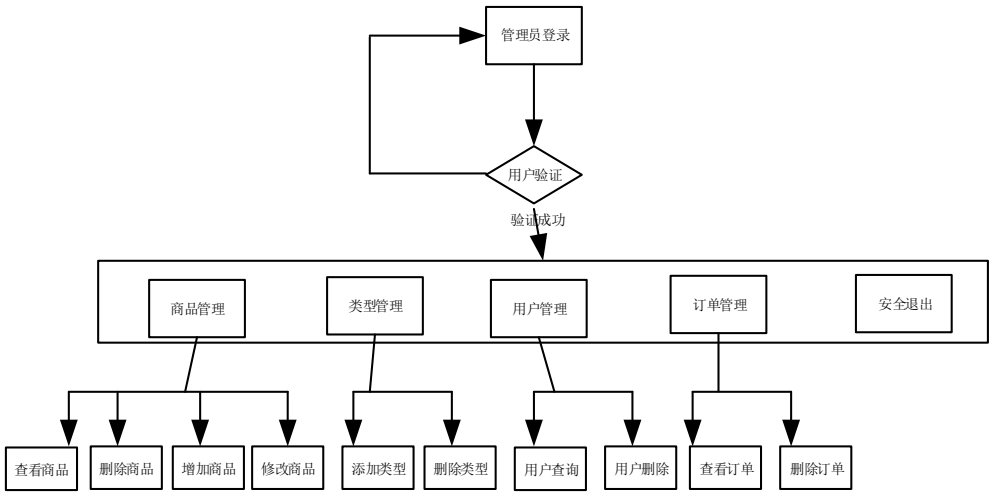

图 2.3 后台管理模块图

2.3.2 前台购物模块

前台功能的流程如图 2.4 所示。

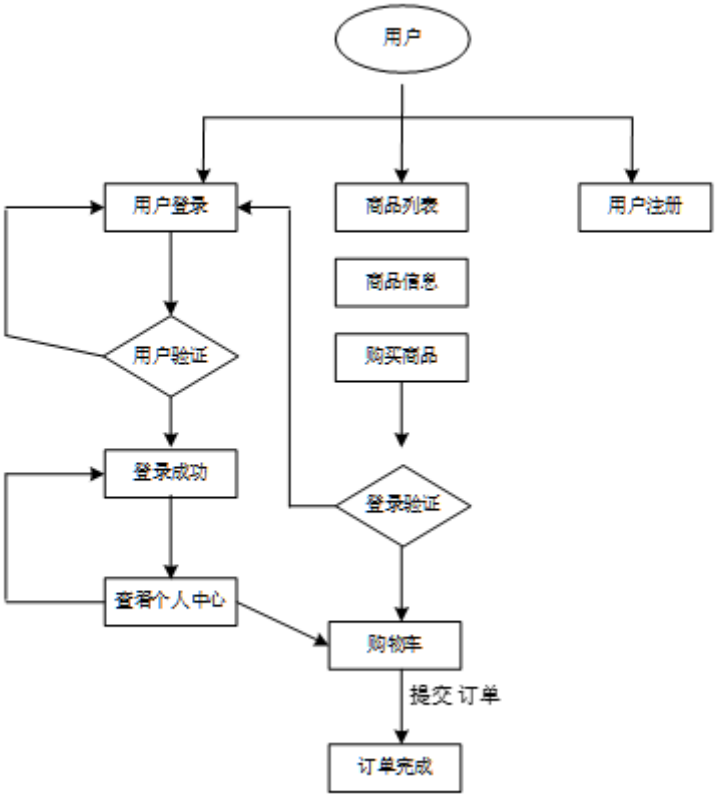

图 2.4 前台功能流程图

3 数据库设计

系统采用加载纯java数据库驱动程序的方式连接MYSQL数据库,在MYSQL中创建了数据库 shop,并在 shop 中创建 9 张与系统相关的数据表,即 ausertable、busertable、 carttable 、 goodstable 、goodstype 、orderdetail 、orderbasetable

3.1 数据库概念结构设计

根据系统设计与分析可以设计出如下数据结构:

(1) 管理员:包括用户名和密码。管理员用户和密码由数据库管理员预设,不用注册。

(2) 用户:包括用户 ID、邮箱和密码。注册用户的邮箱不能相同,用户 ID 唯一。

(3) 商品类型:包括商品 ID 和类型名称。商品类型由数据库管理员管理,包括新增和删除管理。

(4) 商品:包括商品编号、名称、原价、现价、库存、图片以及类型。其中,商品编号唯一,类型与“商品类型”关联。

(5) 购物车:包括购物车 ID、用户 ID、商品编号以及购买数量。其中,购物车 ID 唯一,用户 ID 与“用户”关联,商品编号与“商品”关联。

(6) 订单基础信息:包括订单编号、用户 ID、订单金额、订单状态以及下单时间。其中,订单编号唯一,用户 ID 与“用户”关联。

(7) 订单详情:包括订单编号、商品编号以及购买数量。其中,订单编号与“订单基础信息”关联, 商品编号与“商品”关联。

3.2 数据库逻辑结构设计

将数据库概念结构图转换为 MYSQL 数据库所支持的实际数据模型, 如图 3.2 所示。

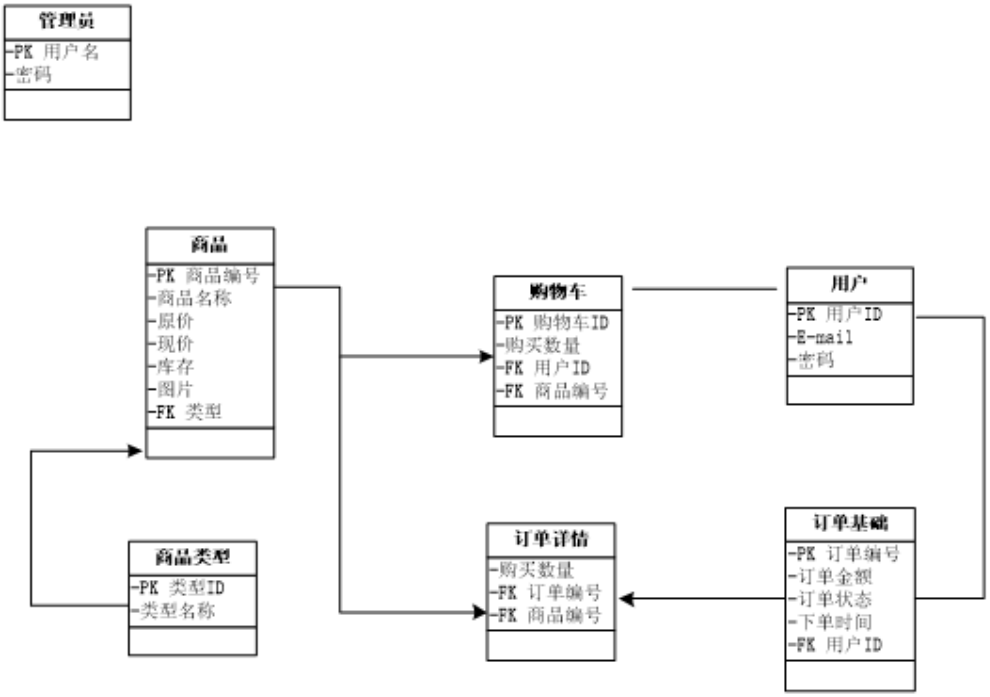

图 3.2 数据库逻辑结构图

各类详细属性如表 3.1~表 3.7 所示。

表 3. 1 管理员信息表 (ausertable)

字段	含义	类型	长度	是否为空
aname	用户名(pk)	varchar	20	No
apwd	密码	varchar	20	No

表 3.2 用户信息表 (busertable)

字段	含义	类型	长度	是否为空
id	用户名 (PK 自增)	Int	20	No
bemail	E-mail	varchar	20	No
bpwd	密码	Varchar	20	No

表 3. 3 商品类型表 (goodstype)

字段	含义	类型	长度	是否为空
id	类型 ID(pk)自增	int	20	No
Typename	类型名称	varchar	20	No

表 3.4 商品信息表 (goodstable)

字段	含义	类型	长度	是否为空
id	商品编号 (pk)自增	Int	20	No

gname	商品名称	Vhachar	20	No
goprice	原价	Double		No
grprice	现价	double		No
gstore	库存	int	20	No
gpicture	图片	Varchar	20	No
goodstype_id	类型(FK)	Int	20	No

表 3.5 购物车表 (carttable)

字段	含义	类型	长度	是否为空
id	购物车 ID (PK 自增)	Int	20	No
busertable_id	用户 ID(FK)	Int	20	No
goodstable_id	商品编号(FK)	Int	20	No
shoppingnum	购买数量	int	20	No

表 3.6 订单基础表 (orderbasetable)

字段	含义	类型	长度	是否为空
id	订单编号 (PK 自增)	Int	20	No
busertable_id	用户 ID(FK)	Int	20	No
amount	订单金额	Double		No
status	订单状态	Tinyint	5	No
orderdate	下单时间	Datetime		No

表 3.7 订单详情表 (orderdetail)

字段	含义	类型	长度	是否为空
id	订单编号 (PK 自增)	Int	20	No
orderbasetable_id	订单编号 (FK)	Int	20	No
goodstable_id	商品编号 (FK)	Int	20	No
shoppingnum	购买数量	int	20	No

3.3 类图描述

类图详细描述如图 3.3 所示。

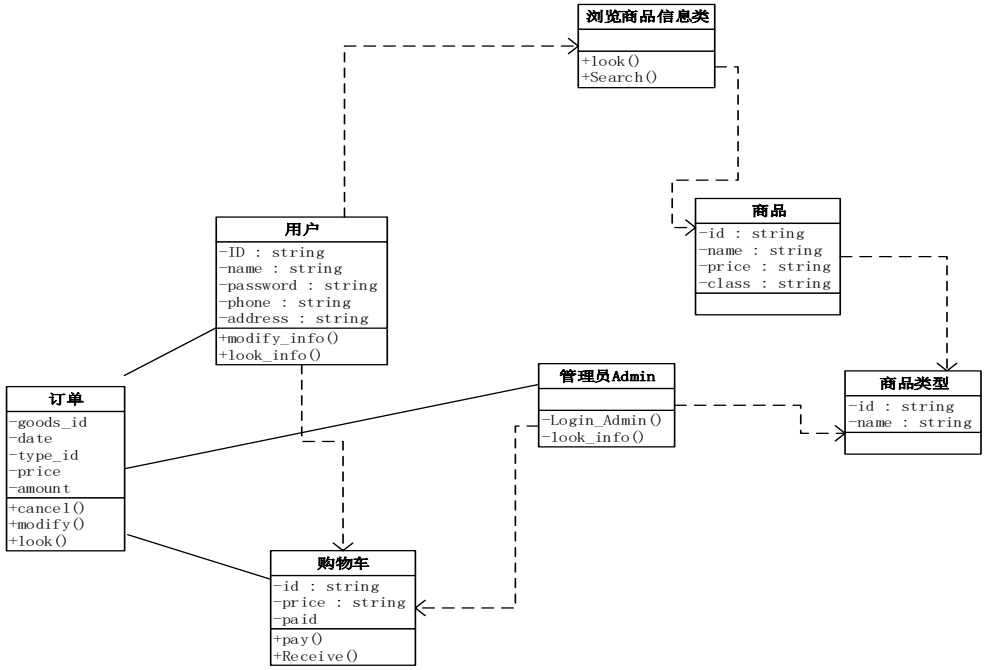

图 3.3 属性类型描述图

类图具体描述如下列表：

表 3.8 用户类

user 类
类名 customer
整体说明：注册该系统的顾客
属性说明： 用户的基本信息：id，password，name，phone，Email，address 操作说明：修改个人信息 modify_info（） 查看个人信息 look_info（）

表 3.9 管理员类

系统管理员
整体说明：来描述管理此系统的人员信息
类名：Admin
操作说明：登录管理整个系统

表 3.10 订单类

订单类
整体说明：一个基于商品和买家的关联累
属性说明：订单号 ID　总价格 ：price
订单的基本信息：订单号 order_id 数量 amount 价格 price 日期 date
操作说明：删除订单 delete(),查看订单 look()

表 3.11 商品类

商品类
整体说明：可买卖的物品
类名：commodity
属性说明：订单的基本信息：id ,name,price,date,amount,
操作说明： 关系：与商家和买家有关联关系

表 3.12 商品维护类

商品维护类
整体说明：进行商品信息修改的操作
类名_modify_com_info
操作说明：增加商品信息 add_commdity, 修改商品信息 modify_commdity 删除商品信息 delete_commdity

表 3.3 购物车类

购物车类
整体说明：账单详情查看
类名：cart
属性说明：goodsID，goodsname,goodsproce ,num
操作说明：修改购物车

3.4 顺序图

用户注册流程如图 3.4 所示。

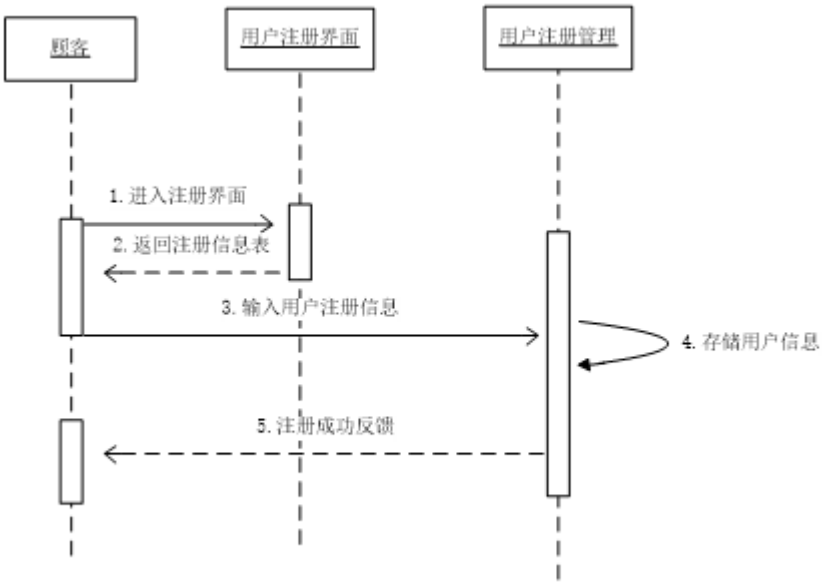

图 3.4 用户输出流程图

对象名称:用户/顾客，登录注册，注册管理;

功能：完成游客注册，分配一个唯一的账号。

用户登录流程如图 3.5 所示。

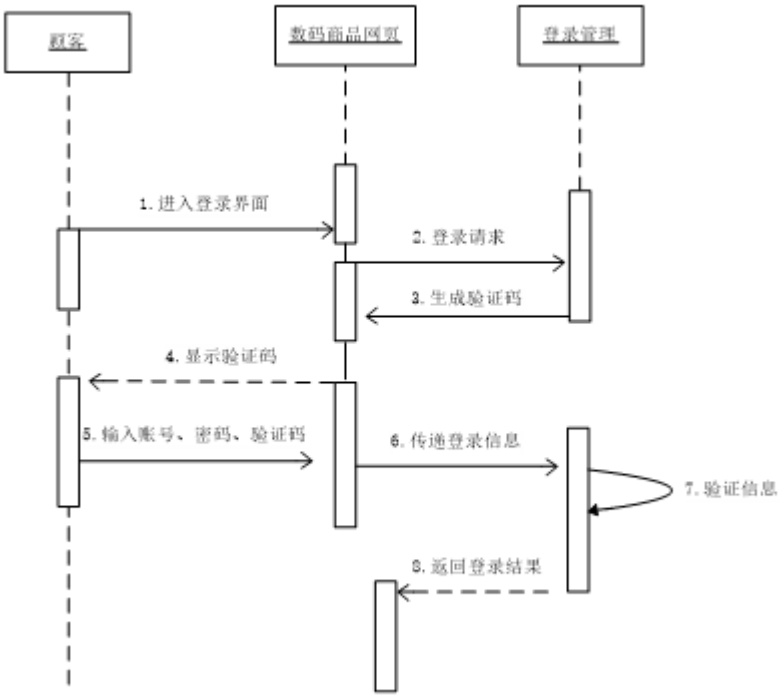

图 3.5 用户登录流程图

对象名称：用户/顾客，数码商城界面，登录管理;

功能：完成登录系统。

4 系统实现

依据 SpringMVC 框架思想，将开发类主要分为 前后台 controller 层、Pojo 持久化层、Exception 统一异常、util 工具、dao、service 逻辑包。Mybati 层，Mybatis 包含所有领域对象及相应的 ORM 映射文件，dao 包含所有的持久层抽象类，service 包含所有业务层类。

另一方面根据系统开发所需建立 filter（过滤器）包、listener（监听器）包、tool（工具）包,整个系统主要包括商品信息模块、个人中心模块、购物车模块、订单模块、商品类别、后台管理员模块。

4.1 系统目录结构

(1) controller 层

系统的控制器类在该包类中,包含前台(admin)和后端(before)的子系统。

(2) dao 层

dao 层中存放了 Java 接口程序用于实现数据库的持久化操作,每个 dao 的接口方法与 SQL 映射方法文件中的 id 相同。

(3) Exception 层

该包中的异常类有 3 个,其中,AdminExcepiton 处理管理员未登录异常:UserException 处理前台用户未登录异常:

MyExceptionHandler 对系统进行统一异常处理,包括管理员未登录异常、前台用户未登录异常和一些未知异常。

(4) mybatis 层

MyBatis 的核心配置文件 mybatis-config-xml 和 SQL 映射文件在该包中。

(5) po 层

持久化类存放在该包中

(6) service 层

service 中有两个子包,admin 和 before 存放管理相关业务的接口和实现类

(7) Util 层

存放的是系统的工具类,包括获取时间字符串方法以及获取前台用户登录 ID 方法。

4.2 登录权限控制

@ModelAttribute 注解

首页被控制器执行所以登录前先判断管理员是否登录:

```
@Controller
public class BaseController {
    @ModelAttribute    //注解
    //没有登录就调用该方法
     public void isLogin(HttpSession session, HttpServletRequest request) throws AdminException {
          if(session.getAttribute("auser") == null){
               throw new AdminException("没有登录");     } } }
```

4.3 管理员页面

4.3.1 管理员登录页面

后台管理网站通过初始化登录页面进入注册页面:

在管理员输入用户名和密码后,系统将对管理员的用户名和密码进行验证.如果用户名和密码正确,则成功登录,进入系统管理员页面(main.jsp)错误则重新跳转到当前页面控制器类请求路径为 admin/login 如图 4.3 所示。

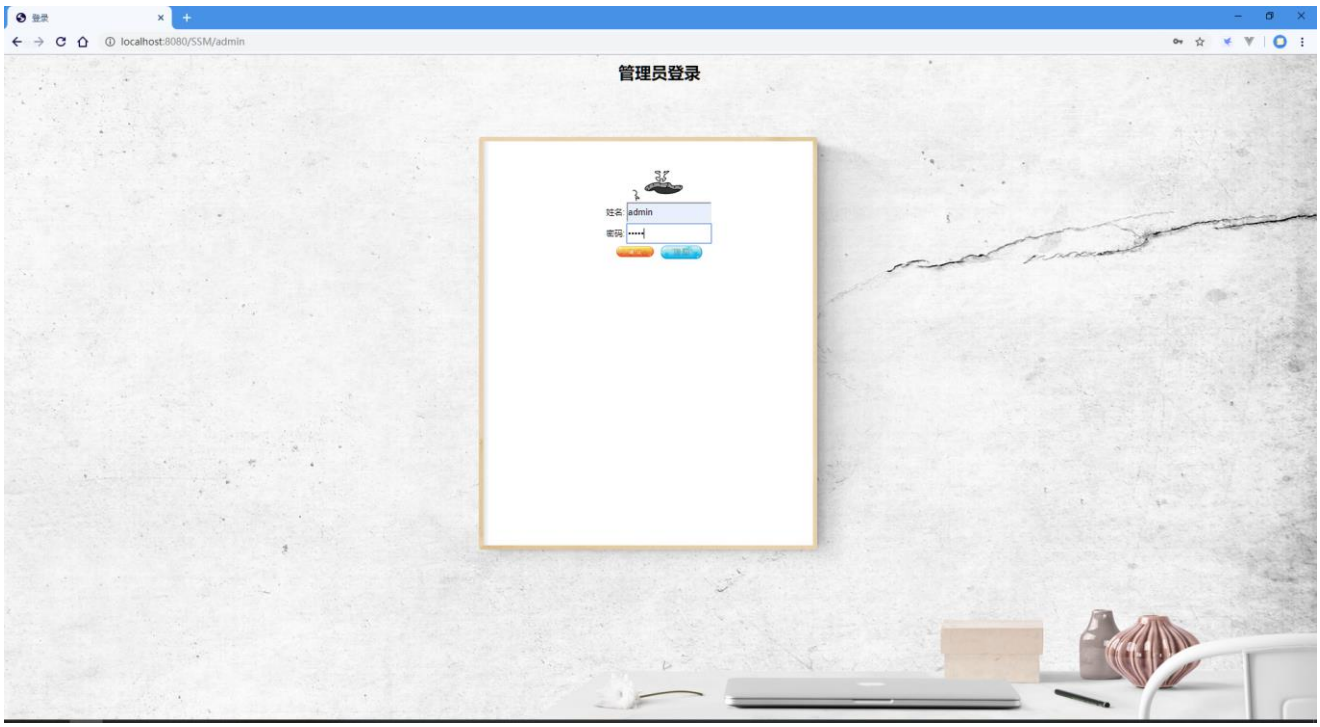

图4.3管理员登录页面图

Service 层代码:

```
@Service("adminService")
@Transactional
public class AdminServiceImpl implements AdminService{
    @Autowired
    private AdminDao adminDao;
    @Autowired
    private AdminTypeDao adminTypeDao;
    @Override
    public String login(Auser auser, Model model, HttpSession session) {
        if(adminDao.login(auser) != null && adminDao.login(auser).size() > 0) {
            session.setAttribute("auser", auser);
            //添加商品与修改商品页面
            session.setAttribute("goodsType", adminTypeDao.selectGoodsType());
            return "admin/main";
        }
        model.addAttribute("msg", "用户名或密码错误!");
        return "admin/login";}}
```

4.3.2 类型管理

类型分为添加类型和删除类型

添加类型 超链接(adminType/toAddType)打开如图 4.4 所示。

图 4.4 类型管理图

Service 层

Service 层调用 Dao 层进行数据库中的添加和删除操作

```
@Service("adminTypeService")
@Transactional
```

```
public class AdminTypeServiceImpl implements AdminTypeService{
    @Autowired
    private AdminTypeDao adminTypeDao;
    @Override
    public String toAddType(Model model) {
        model.addAttribute("allTypes", adminTypeDao.selectGoodsType());
        return "admin/addType";}
    @Override
    public String addType(String typename, Model model, HttpSession session) {
        adminTypeDao.addType(typename);
        //添加商品与修改商品页面
        session.setAttribute("goodsType", adminTypeDao.selectGoodsType());
        return "forward:/adminType/toAddType";}
    @Override
    public String toDeleteType(Model model) {
        model.addAttribute("allTypes", adminTypeDao.selectGoodsType());
        return "admin/deleteType";}@Override
    public String deleteType(Integer id, Model model) {
        //类型有关联
        if(adminTypeDao.selectGoodsByType(id).size() > 0) {
            model.addAttribute("msg", "类型关联,不能删除");
            return "forward:/adminType/toDeleteType";}
        if(adminTypeDao.deleteType(id) > 0)
            model.addAttribute("msg", "类型删除成功!");
        //回到删除页面
        return "forward:/adminType/toDeleteType";
```

4.3.3 添加、删除、修改、查询商品信息

在系统主页面(main.jsp)用户通过显示商品页面进行点击对应的添加、删除、修改、查询商品信息如图 4.5 所示。

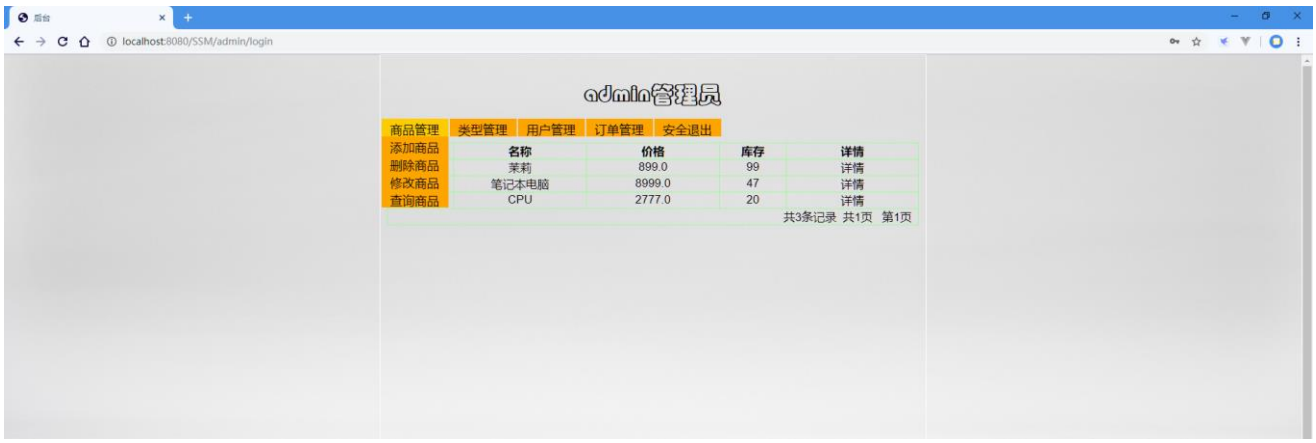

图 4.5 商品类型图

controller 层

根据@RequestMapping 注解对应的控制器类中增删改查方法请求调用处理业务

Service 层

Service 层调用 Dao 层进行数据库中的增删改查操作

Sql 映射文件代码

```
<mapper namespace="com.dao.AdminGoodsDao">
    <!-- 查询商品 -->
    <select id="selectGoods"  resultType="Goods">
        select * from goodstable
    </select>
    <!-- 分页查询商品 -->
    <select id="selectAllGoodsByPage" resultType="Goods" parameterType="map">
        select * from goodstable order by id limit #{startIndex}, #{perPageSize}
    </select>
    <!-- 添加商品 -->
    <insert id="addGoods" parameterType="Goods">
        insert into goodstable (id,gname,goprice,grprice,gstore,gpicture,goodstype_id)
        values (null, #{gname}, #{goprice}, #{grprice}, #{gstore}, #{gpicture}, #{goodstype_id})
    </insert>
    <!-- 根据 id 查询一个商品 -->
    <select id="selectGoodsById"  resultType="Goods" parameterType="Integer">
        select gt.*,gy.typename from goodstable gt,goodstype gy where gt.id=#{id} and gt.goodstype_id = gy.id
    </select>

    <!-- 删除单个商品 -->
    <delete id="deleteAGoods" parameterType="Integer">
        delete from goodstable where id=#{id}
    </delete>
    <!-- 修改一个商品 -->
    <update id="updateGoodsById" parameterType="Goods">
     update goodstable
     <set>
        <if test="gname != null">
          gname = #{gname},
        </if>
        <if test="goprice != null">
          goprice = #{goprice},
        </if>
        <if test="grprice != null">
          grprice = #{grprice},
```

```
        </if>
        <if test="gstore != null">
          gstore = #{gstore},
        </if>
        <if test="gpicture != null">
          gpicture = #{gpicture},
        </if>
        <if test="goodstype_id != null">
          goodstype_id = #{goodstype_id},
        </if>
      </set>
        where id = #{id}
  </update>
```

4.4 订单管理

单击后台管理系统中订单管理如图 4.6 所示。

图 4.6 订单管理图

Controller 层

根据@RequestMapping 注解找到对应的控制器类中的请求调用处理业务

Service 层

Service 层调用对应 Dao 层进行数据库中的订单操作

Sql 映射文件

```
<mapper namespace="com.dao.AdminOrderDao">
    <select id="orderInfo"    resultType="map" >
        select ot.id, ot.amount, ot.status,    orderdate, bt.bemail, ot.busertable_id
        from    ORDERBASETABLE    ot,    BUSERTABLE    bt    where
ot.busertable_id=bt.id
    </select>
        <delete id="deleteOrderDetail" parameterType="Integer">
        delete from orderdetail where orderbasetable_id=#{id}
    </delete>
```

```
    <delete id="deleteOrderBase" parameterType="Integer">
        delete from orderbasetable where id=#{id}
    </delete>
</mapper>
```

4.5 用户管理

单击后台管理系统中用户管理如图 4.7 所示。

图 4.7 用户管理图

SQL 映射文件

```
<mapper namespace="com.dao.AdminUserDao">
    <select id="userInfo"    resultType="Buser" >
        select * from busertable
    </select>
    <delete id="deleteuserManager"    parameterType="Integer" >
        delete from busertable where id = #{id}
    </delete>
</mapper>
```

4.6 退出系统

单击后台管理系统中订单退出系统如图 4.8 所示。

图 4.8 退出系统图

点击退出直接退出页面
执行.invalidate()方法

4.7 前端显示页面

没有登录的用户具有浏览首页,查看商品详情和查看公告等权限。成功登录的用户除了具有以上权限外,还具有购买商品、查看购物车、以及查看用户中心的权限如图 4.9 所示。

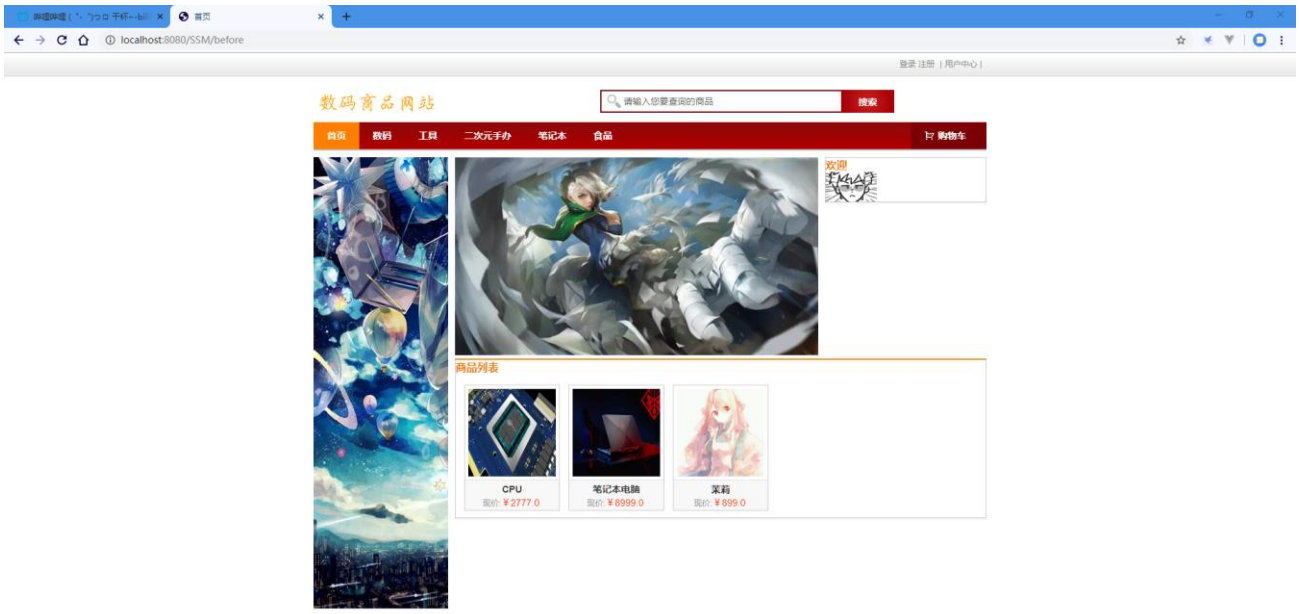

图 4.9 前端显示页面图

```
<!--搜索-->
<div class="all_zong_logo">
    <div class="all_zong_logo2">
        <img src="images/before/log.png" /></div>
    <div class="back_search">
        <div class="back_search_red">
            <form action="search" name="myForm" method="post">
                <div class="div2">
                    <input type="text" name="mykey" class="txt" value="
请输入您要查询的商品"   onfocus="clearValue()" />
                </div>
                <div class="div1">
                <input type="submit" class="an"    value="搜索" />
                </div></form></div></div></div>
<div class="skin_a">
<div class="front_daohangbj">
<div class="all_zong">
<div class="front_daohang"><ul>
  <li class="backbj"><a href="before?id=0">首页</a></li>
<!--  商品类型  -->
```

```
                    <c:forEach items="${goodsType}" var="g">
                    <li><a
href="before?id=${g.id }">${g.typename }</a></li></c:forEach>
                        <li class="buy"><p class="car">
                            <a href="cart/selectCart">购物车</a>
```

4.8 用户注册/登录页面

通过注册登录进入 index.jsp 主页面

注册页面如图 4.10 所示。

图 4.10 用户注册图

登录页面如图 4.11 所示。

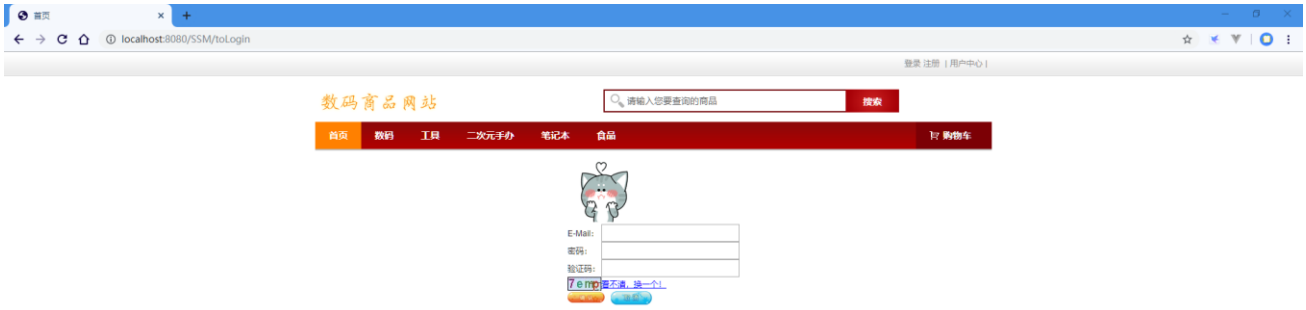

图 4.11 登录页面图

页面登录代码(login.jsp)

```
<script type="text/javascript">
    //确定按钮
    function gogo(){
        document.loginform.submit();
    }
    //取消按钮
```

```
function cancel(){
    document.loginform.action="";
}
function refreshCode(){
    document.getElementById("code").src = "validateCode?" + Math.random();
 }
</script> </head> <body>          <center>
<form:form action="user/login" method="post" modelAttribute="buser"    name =
"loginform">
<table> <br><tr>
<td colspan="2"><img src="images/admin/login.gif"></td></tr><tr>
            <td>E-Mail：</td>
            <td><input  type="text"  name="bemail"  value="${buser.bemail  }"
 class="textSize"/></td></tr><tr>
            <td>密码：</td>
            <td><input type="password" name="bpwd" class="textSize"/></td>
 </tr><tr><td>验证码：</td>
            <td><input type="text" name="code" class="textSize"/></td>
</tr><tr><td>
                <img id="code" src="validateCode"/></td>
            <td class="ared">
                <a href="javascript:refreshCode();"><font color="blue">看不清，换
一个！</font></a>
            </td></tr><tr>
            <td colspan="2">
                <input          type="image"          src="images/admin/logins.png"
onclick="gogo()">
                <input          type="image"          src="images/admin/register.png"
onclick="cancel()">
            </td></tr>   </table>
```

4.9 购物车类

4.9.1 购物车页面

单击购物车页面进行商品购买如图 4.12 所示。

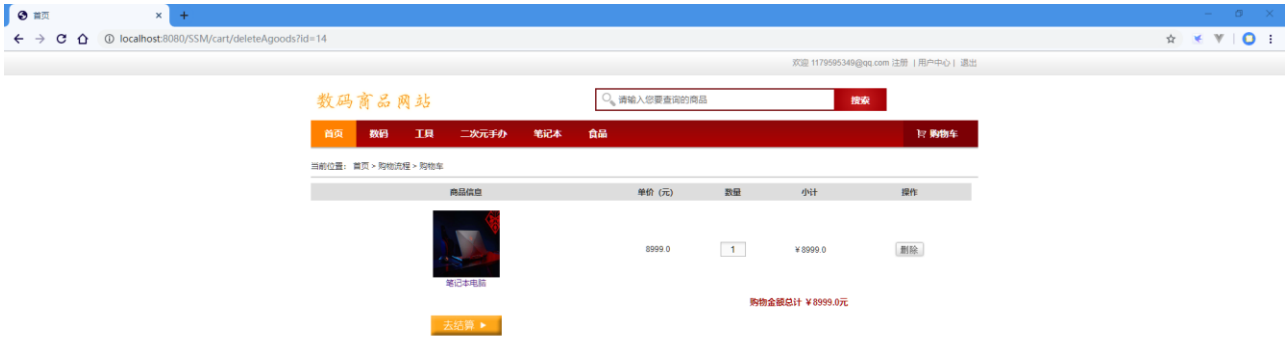

图 4.12 购物车页面图

购物车核心代码:

```
<c:forEach var="ce" items="${cartlist}">
    <tr>
    <td align="center">
      <img style="width: 100px; height: 100px;"src="logos/${ce.gpicture}" border="0"title="${ce.gname}" />
      <br/>
      <a href="goodsDetail?id=${ce.id}" >${ce.gname}</a>
  </td>
    <td width="110px" align="center">
      <span>${ce.grprice}</span>
      </td>
        <td align="center" width="115px"
          valign="middle">
          <input type="text" name="goods_number"
          value="${ce.shoppingnum}" size="4"
          style="text-align: center; width: 36px;"/>
        </td>
        <td align="center" width="115px">
        ￥<span>${ce.smallsum}</span>
        </td>
        <td align="center" width="185px">
        <a href="javaScript:deleteAgoods('${ce.id}')"
        title="删除">
        <img src="images/before/sc.png" /></a>
        </td></tr></c:forEach>
```

4.9.2 订单管理

购物车类的商品通过结算进入订单管理页面如图 4.13 所示。

图 4.13 订单管理图

SQL 映射文件

```
<mapper namespace="com.dao.OrderDao">
    <!-- 添加一个订单，成功后将主键值回填给 id（po 类的属性）-->
    <insert id="addOrder" parameterType="Order" keyProperty="id" useGeneratedKeys="true">
        insert into orderbasetable (busertable_id, amount, status, orderdate) values
        (#{busertable_id}, #{amount}, 0, now())
    </insert>
    <!-- 生成订单详情 -->
    <insert id="addOrderDetail" parameterType="map">
        insert into ORDERDETAIL (orderbasetable_id, goodstable_id, SHOPPINGNUM) select #{ordersn}, goodstable_id, SHOPPINGNUM from CARTTABLE where busertable_id = #{uid}
    </insert>
    <!-- 查询商品购买量-->
    <select id="selectGoodsShop" parameterType="Integer" resultType="map">
        select shoppingnum gshoppingnum, goodstable_id gid from carttable where busertable_id=#{uid}
    </select>
    <!-- 更新商品库存 -->
    <update id="updateStore" parameterType="map">
        update GOODSTABLE set GSTORE=GSTORE-#{gshoppingnum} where id=#{gid}
    </update>
```

4.9.3 支付页面

订单管理页面提交确认购买商品,跳转到支付页面如图 4.14 所示。

图 4.14 支付页面图

支付页面代码(orderone.jsp)

```
<script type="text/javascript">
                //确定按钮
                function gogo(){
                    document.payForm.submit();
                }
            </script>
    </head>
    <body>
        <div class="blank"></div>
        <div class="block clearfix">
         <h2 style="text-align:center; height:30px; line-height:30px;">您的订单已提交成功，
                订单号: <font style="color:red" size='5'>${ordersn}</font></h2><br/>
            <center>

            <form action="order/pay" method="post" name="payForm">
                    <input type="image" src="images/before/alipay.png" onclick="gogo()"/>
            </form>
            </center>
        </div>
    </body>
```

Controller 层

通过@RequestMapping("/pay")

Service 层

跳转到支付页面

Sql 映射文件

```
    <!-- 支付订单 -->
    <update id="pay" parameterType="Integer">
        update orderbasetable set status=1 where id=#{ordersn}
    </update>
```

</mapper>

5 测试计划

5.1 目的

这个对于数码商品网站行测试的总体安排和进度计划。

1.确定现有项目的信息和应测试软件构件。

2.标明推荐的测试需求。

3.推荐可采用的测试策略，并对这些策略加以说明。

4.确定所需的资源，并对测试的工作量进行估计。

5.列出测试项目的可交付元素。

5.2 背景说明

A.系统名称：数码商品系统

B.系统简介：该系统为一个基于 J2ee 技术的数码商品系统，旨在实现一个数码商品系统，出售各种数码产品。该系统将面向所有消费者。

5.3 测试范围

整个系统进行全面的登录验证数据测试:

表 5-3 测试范围

序号	产品描述	测试重点	备注
1	数码商品网站	客户界面的登录	
2	数码商品网站	订单管理	
3	数码商品网站	购物车管理	
4	数码商品网站	密码管理	

5.4 风险评估

1.需求风险:对软件需求理解不准确,导致测试范围存在误差。

2.测试用例风险:测试用例设计不完善,忽略了边界条件,异常处理等情况,用例没有完全覆盖需求,测试用例没有得到全部执行,有些用例被无意的漏掉。

3.缺陷风险:某些缺陷偶发,难以重现,容易被遗漏。

4.测试环境风险:测试环境与生产环境不能完全一致,导致测试结构存在误差。

5.测试技术风险:某些项目存在技术难度,项目延期

以上是测试过程中可能发生的风险,其中有的风险是难以避免的,如缺陷风险.对于难以避免的风险,我们目标是将风险降到最低水平。

5.5 测试资源需求

确保项目测试环境符合测试要求，降低严重影响测试结果真实性和正确性的风险，包括各个阶段（单元、集成、系统测试）的资源要求。

表 5.5.1 硬件资源

资源名称/类型	配置
测试 PC 机	WEB 服务器 内存 16G 硬盘 500G

表 5.5.2 软件资源

数据库管理	
应用软件	Idea Tomcat 9x
客户端前端展示	谷歌浏览器

5.6 测试数据要求

主要介绍测试范围并作概况性描述，这部分内容是测试计划的核心所在。

表 5-6 测试内容

序号	测试用例名称	测试状态	测试结果	备注
1	用户注册	已执行	测试通过	
2	注册用户登录	已执行	测试通过	
3	增删改查/商品	已执行	测试通过	
4	提交订单	已执行	测试通过	
5	查询商品信息	已执行	测试通过	
6	管理员登录	已执行	测试通过	
7	用户管理系统	已执行	测试通过	
8	订单管理系统	已执行	测试通过	
9	商品管理系统	已执行	测试通过	

管理员注册和登录页面如图 5.6 所示。

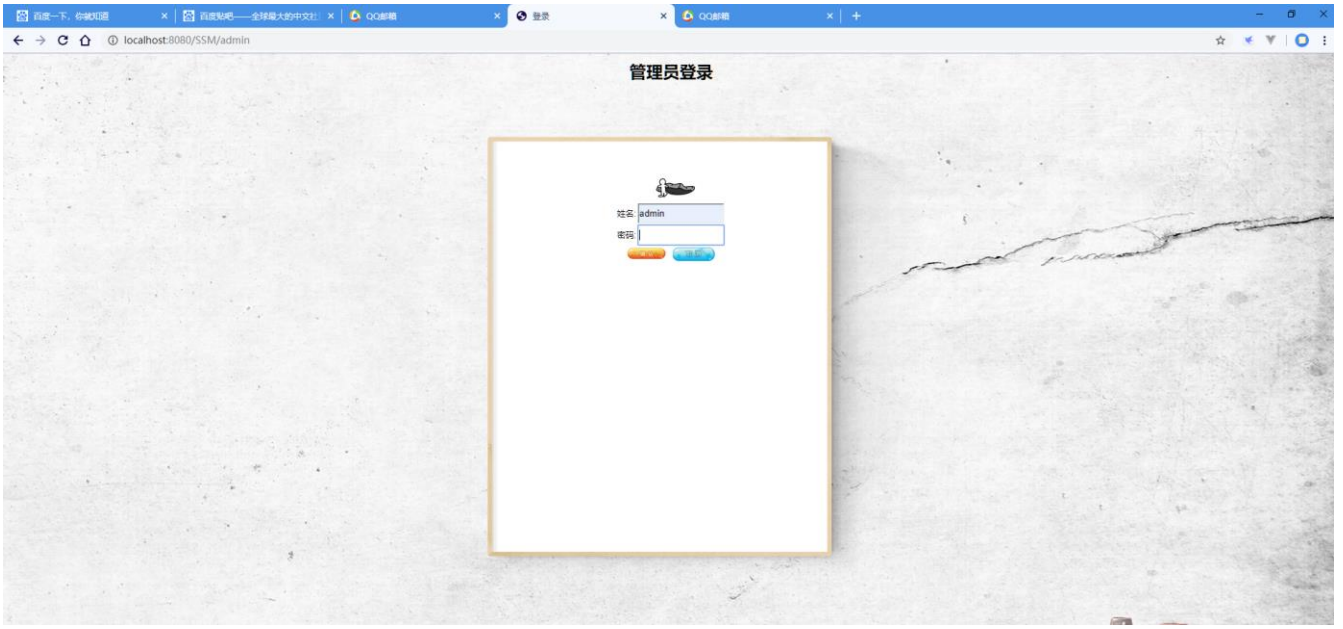

图 5.6 管理员注册和登录图

根据整个系统描述了要进行的测试活动的范围、方法、资源和进度的文档；是对整个信息系统应用软件组装测试和确认测试。

表 5.6.2 测试报告

输入条件	有效等价类	无效等价类
用户名字符组合	①字母（a-z 或 A-Z）和数字（0-9）组合	②特殊字符③纯数字组合④纯字母组合
用户名长度	⑤8-16 位数字字母组合	⑥小于 8 位的数字字母组合 ⑦大于 16 位的数字字母组合 ⑧用户名为空 ⑨用户名为空格
用户名格式	⑩字母开头的用户名	⑪数字开头的用户名 ⑫其他字符开头的用户名
密码字符组合	⑬字母（a-z 或 A-Z）和数字（0-9）组合	⑭特殊字符⑮纯数字组合 ⑯纯字母组合
密码长度	⑰8-16 位数字字母组合	⑱小于 8 位的数字字母组合 ⑲大于 16 位的数字字母组合⑳用户名为空 ㉑用户名为空格
密码格式	㉒字母开头的密码	㉓数字开头的密码㉔其他字符开头的密码

将所有可能输入数据的数据域进行划分，分成若干子集，再从其中选出少数具有

代表性的的是用例测试程序，该方法是一种常用的黑盒测试方法。

表 5.6.3 有效等价类测试

用例 ID	功能点	测试输入	预期结果	实际结果	覆盖的等价类
1	登陆功能	用户名：admin 密码：admin	登录成功	登录成功	① ⑤ ⑩ ⑬⑰ ㉒

与有效等价类的定义恰巧相反。无效等价类指对程序的规格说明是不合理的或无意义的输入数据所构成的集合。对于具体的问题，无效等价类至少应有一个，也可能有多个。

表 5.6.4 无效等价类测试

用例 ID	功能点	测试输入	预期结果	实际结果	覆盖的等价类
3	用户登录	用户名：admin 密码：（空格）	登录失败	登录失败	③ ⑪ ㉑

6 结论

开发 Web 站点，是一项复杂、繁重的工作，需要多方面的知识。此次通过设计数码商品管理系统，使我接触到许多基于 java 语言的 WEB 网络开发方面的知识，包括 HTML 语言，CSS 级联样式表，客户端开发语言 JavaScript,MVC 分层体系结构，支持 MVC 架构的 Spring 框架，开源服务器 Tomcat,mysq1 数据库，服务器端开发语言 JSP 等，这些语言(或技术)扩展了我的知识面，也使我在制作的过程中对它们的特点，应用方向等有基本的了解，并在实际的应用中大致明白如何将它们融合运用，以开发出界面美观，功能强大的 Web 站点来。

通过本次亲自动手设计系统，我更加深刻的了解系统开发的全过程，从中我熟练掌握了 B/S 结构 Web 应用软件设计的思想及其开发的全过程。首先要进行系统的需求分析，分析系统要达到什么功能，系统要划分为几个模块来设计，系统的具体功能应当如何去实现;分析完之后，考虑其涉及到的计算机专业知识，把问题细化，把大的问题划分为小的问题，然后 e 逐个进行解决。学年论文设计开始阶段，由于缺少软件架构的设计经验，所以在项目的业务流程和软件的开发

规范上犯了很的错误，所以在后期的组装过程变的非常的困难。

设计收获:

1.掌握了网站开发的基本流程，设计整个网站的功能模块图;

2.掌握了如何搭建和配置一个 Tomcat 服务器;

3.掌握了使用 JSP 编程，实现对信息的显示、修改、删除和添加等。

7 参考文献

[1]景玉建.电子商务的发展与前景[J].《职业时空》，2005，13：10-11

[2]张孝祥.深入 JavaWeb 开发内幕一一核心基础[M].北京:电子工业出版社.北京 2006.10

[3]百度百科 JavaBeans[DB/OL],http://baike.baidu.com/view/1006495.htm

[4]百度百科 J2EE[DB/OL]，http://baike.baidu.com/view/1507.htm

[5]百度百科 spring[DB/OL], http://baike.baidu.com/view/23023.htm

[6]黄杰湘制作[DB/OL]，《mysq1 中文参考手册》，网络电子书，http:www.devoinfo.com

[7]Tomcat 简介[DB/OL]，http://b1og.sina.com.cn/n/4836985010002yg

[8]百度百科 B/S[DB/OL]，http://baike.baidu.com/view/1477348.html

[9]百度百科 3-tier[DB/OL],http://baike.baidu.com/view/687468.html

[10]百度百科软件测试[DB/OL]，http://baike.baidu.com/view/16563.htm