|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 题目名称 | 基于Java Web技术实现的一个图书商城 | | | | |
| **姓名** | **陈彦宏** | **班级** | **软工191** | **学号** | **17** |
|  | | | | | |
| 具体要求 | | | | | |
| **1.功能要求**  **1）****前端功能**  图书的分类展示;  图书详情  图书搜索  订单详情(购物车)  登录与注册  注意：注册用户可以购买商品，非注册用户仅可以浏览、查询商品  **2）后端功能**  用户管理：分普通用户和管理员两类；管理员可以禁用普通用户帐号、查看普通用户的订单信息；可以添加、修改和删除其他管理员帐号；  图书管理：管理员对图书信息的添加、修改、删除和查询功能，包括图书详细信息、图片上传等。  **2.技术要求**  1）基于Java Web开发技术，并使用MVC开发模式实现；数据库使用mysql；  2）知识点应用：[1]项目创建；代码调试;[2]JSP基础语法；[3]内置对象使用；[4]合理使用Javabean；[5]理解和掌握servlet；[6]理解和掌握MVC开发模式；[7]掌握在JSP中操作数据库的方法；[8]掌握JSTL核心标签库的使用；[9]前端基础知识的综合应用。  **3.设计文档**  1）内容  整体设计（功能描述、系统结构图、使用到的技术等）；数据库设计；详细设计(代码和界面)；小结；自我评价；  2）格式  参照任务书模版。要求格式良好。  提交：  1.项目源代码（包括数据库，导出数据库结构和数据，便于阅卷）；  2.本设计文档；  3.提交日期；2021-12-24 | | | | | |

目录

[基于Java Web技术实现的一个图书商城 2](#_Toc90550920)

[一、 整体设计 2](#_Toc90550921)

[1.1功能描述 2](#_Toc90550922)

[1.2系统结构图 3](#_Toc90550923)

[1.3实现技术 3](#_Toc90550924)

[1.4运行环境 3](#_Toc90550925)

[二、 数据库设计 3](#_Toc90550926)

[三、 详细设计 5](#_Toc90550927)

[3.1 javabean类设计 5](#_Toc90550928)

[3.2 前端功能设计 8](#_Toc90550929)

[3. 3 后端功能 32](#_Toc90550930)

[3. 4 退出功能 40](#_Toc90550931)

[四、 设计小结 40](#_Toc90550932)

[五、 自我评价 40](#_Toc90550933)

# 基于Java Web技术实现的一个图书商城

## 整体设计

### 1.1功能描述

（1）前端功能实现1.图书可以通过搜索分类展示；2.图书详情；3.图书搜索；4.订单详情 5.购物车；5.登录与注册。注意要点：注册用户可以购买商品，非注册用户仅可以浏览、查询商品。只有用户才可以使用购买商品和查看我的订单功能。

（2）后端功能实现1.用户管理：分普通用户和管理员两类；管理员可以禁用普通用户帐号，可以添加、修改和删除其他用户帐号；2.图书管理：管理员对图书信息的添加、修改、删除和查询功能，包括图书详细信息、图片上传等。3.订单管理：可以查看用户的订单已经订单的详细信息，可以修改订单的状态。

**注：对于用户的输入有判空及判断输入是否合法**

### 1.2系统结构图

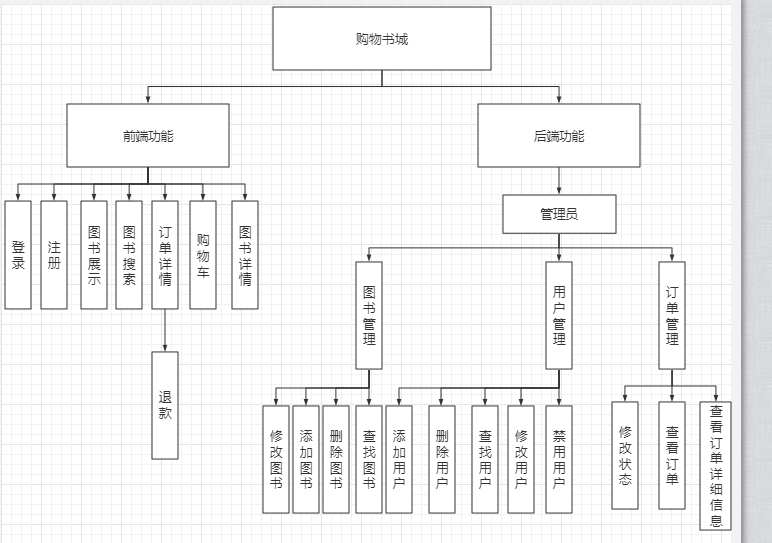


图 1.2系统功能结构图

### 1.3实现技术

基于Java Web开发技术，并使用MVC开发模式实现；数据库使用mysql；

### 1.4运行环境

Windows 7/10

## 数据库设计

用户表（user）

|  |  |  |  |
| --- | --- | --- | --- |
| id | userName | password | ban |
| Int（11）(primary\_key)  自增 | Varchar（255）  Notnull | Varchar（255）  Notnull | Int（11）默认值0  Notnull |

表格 2.1用户表

User表用于存储用户的注册信息，

Id:作为该用户的唯一标识，自动生成；userName：存储用户注册的用户名；password：存储密码；ban：用于该用户账号是否可以访问该网页的标记，若为1，则不可访问网页。

管理员表（admin）

|  |  |  |  |
| --- | --- | --- | --- |
| id | admin\_Name | admin\_password | level |
| Int（11）(primary\_key)  自增 | Varchar（255）  Notnull | Varchar（255）  Notnull | Int（11）默认值1  Notnull |

表格 2.2管理员表

Admin表用于存储管理员的注册信息

Id：作为该管理员的唯一标识，自动生成；admin\_Name:存储管理员注册的用户名；admin\_password：存储密码；level：用于区分管理员账号和用户账号的唯一标识。

图书表（booklist）

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| bookid | bookName | bookPrice | bookAuthor | bookType | bookDetail | bookIsbn | bookFile |
| Int（11）  (primary\_key)  自增 | Varchar（255）  Notnull | Double（11,2）  Notnull | Varchar（255）  Notnull | Varchar（255）  Notnull | Varchar（2550）  Notnull | int（255）  Notnull | Varchar（255）  Notnull |

表格 2.3图书表

Booklist表用于存储图书的信息

Bookid：作为该书本的唯一标识，自动生成；bookName：存储该图书的书名；bookPrice：存储图书的价钱；bookAuthor：存储图书的作者；bookType：存储图书的类型；bookDetail：存储图书的简介；bookIsbn：存储图书的Isbn码；bookFile：存储图书的图片文件路径

购物车表（cart）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| userid | bookid | bookName | bookPrice | number | bookFile |
| Int（11）  foreign key  （参照用户表：userid） Notnull | Int（11）foreign key（参照图书表：bookid） Notnull | Varchar（255）  Notnull | Double（11,2）  Notnull | Int（11）  Notnull | Varchar（255）  Notnull |

表格 2.4购物车表

Cart表用于存储用户加入到购物车中的图书信息

userId：存储当前操作用户的userId作为外键参照user表中的userid；bookid：作为该加入购物车图书的唯一标识，作为外键，参照booklist表中的bookid；bookName：存储该图书的书名；bookPrice：存储图书的价钱；number：存储该图书在购物车中的数量；bookFile：存储图书的图片文件路径

订单表（orderform）

|  |  |  |  |
| --- | --- | --- | --- |
| orderId | userId | sum | state |
| Varchar（255）  (primary\_key) | Int（11）  Notnull | Double（11,2）  Notnull | Varchar（20）  默认值：未发货  Notnull |

表格 2.5订单表

Orderform表：用于存储订单的信息

Orderid：作为该订单的唯一标识；userid：存储当前操作用户的userId作为外键参照user表中的userid；sum：存储该订单的总价钱；state：存储该订单当前的状态；

订单详情表（orderlist）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Id | orderId | userId | bookName | bookPrice | number | bookFile |
| Int（11）  (primary\_key)  自增 | Varchar（255）  foreign key  （参照订单表：orderId）  Notnull | Int（11）  foreign key  （参照用户表：userId）  Notnull | Varchar（255）  Notnull | Double（11,2）Notnull | Int（11）  Notnull | Varchar（255）  Notnull |

表格 2.6订单详情表

Orderlist表：用于存储对于订单的详细信息

Id：不用于实际的信息存储，作为主键只是为了方便在该表中插入多行相同orderId的图书信息；orderId：用于标记该图书对应的订单；作为外键，参照orderform表中的orderId；

bookName：存储该图书的书名；bookPrice：存储图书的价钱；bookFile：存储图书的图片文件路径

## 详细设计

### 3.1 javabean类设计

#### 3.1.1 User类

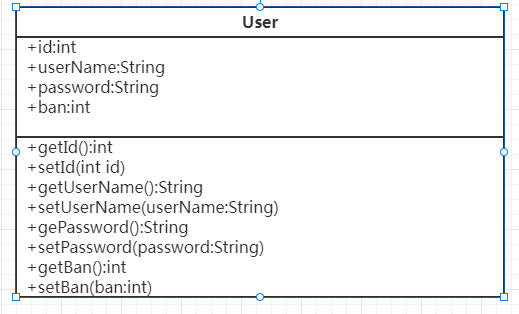


图 3.1.1 User类UML图

#### 3.1.2 Book类

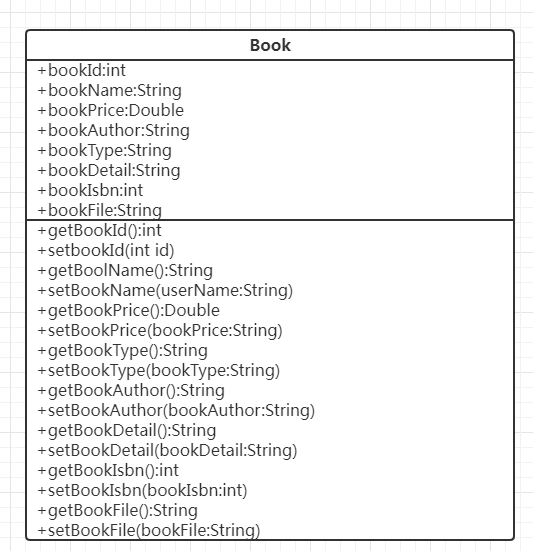


图 3.1.2 Book类UML图

#### 3.1.3 Cart类

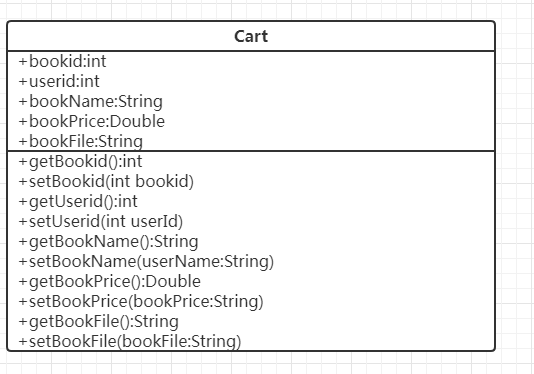


图 3.1.3 Cart类UML图

#### 3.1.4 Order类

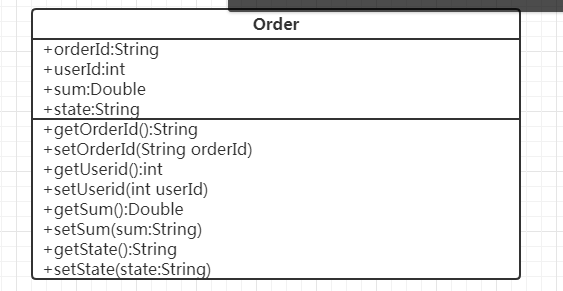


图 3.1.4 Order类UML图

#### 3.1.5 Orderllist类

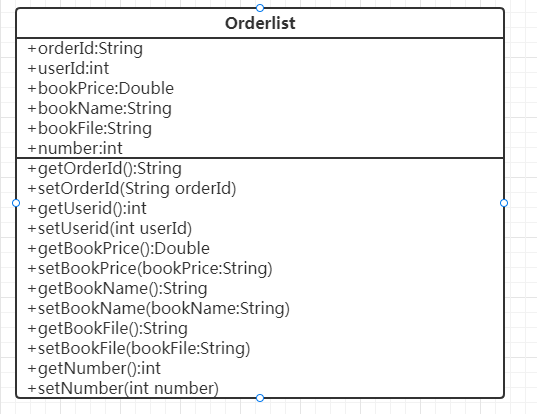


图 3.1.5 Orderlist类UML图

### 3.2 前端功能设计

前端功能设计分为:导航栏，图书展示，登录，注册，购物车，我的订单，图书详情

注册用户可以购买商品，非注册用户仅可以浏览、查询商品。只有用户才可以使用购买商品和查看我的订单功能。被禁止权限的用户无法登陆。

#### 3.2.1 导航栏设计

所有的页面包含一个导航栏，该导航栏由网页标题、注册、登录、浏览图书、我的订单、后台管理、退出登录等功能组成。为了便于维护，其他JSP页面通过使用JSP的 <[%@include...%](mailto:%25@include...%25)>标记将index.jsp（导航栏文件）嵌入到自己的页面。Index.jsp按UTF-8编码保存在Web服务目录中

Index.jsp的代码如下：

<body>

<div class=*"index"*>

<h1 >图书销售网站</h1>

<p>欢迎您：

<c:choose>

<c:when test=*"*${sessionScope.id !=null }*"*>

<c:out value=*"*${sessionScope.userName}*"*/>

<a href=*"login.jsp"*>切换用户</a>

</c:when>

<c:when test=*"*${sessionScope.id ==null }*"*>

<a href=*"login.jsp"*>登录</a>

</c:when>

</c:choose>

</p>

<div class=*"dh"*>

<ul id=*"nav"* >

<li><a href=*"register.jsp"*>注册</a></li>

<li><a href=*"bookServlet"*>浏览图书</a></li>

<li><a href=*"cart\_showServlet"*>购物车</a></li>

<li><a href=*"myOrderServlet"*>我的订单</a></li>

<li><a href=*"admin.jsp"*>后台管理</a></li>

<li><a href=*"exit.jsp"*>退出登录</a></li>

</ul>

</div>

</div>

</body>

</html>

导航栏界面如下图



图 3.2.1.1导航栏界面图

#### 3.2.2登录功能设计

登录功能界面通过login.jsp实现，功能通过loginServlet实现。

login.jsp的代码如下：

<body>

<div class="login">

<form action="loginServlet" method="post">

<h1>登录</h1>

<div class="name">用户名:&nbsp;&nbsp;<input type="text" name="userName" value="${requestScope.userName }" placeholder="用户名"/></div>

<br/>

<div class="pwd">密码:<input type="password" name="password" placeholder="密码"/></div>

<br/>

<div class="admin">管理员登录<input type="checkbox" name="admin" value="admin" /></div>

<br/>

<button class="bt-login">登录</button>

<br/><br/>

<a class="a1" href="register.jsp" >注 册</a>

<a class="a2" href="book\_view.jsp" >返回主页</a>

</form>

<p class="p1">${requestScope.msg }</p>

</div>

</body>

</html>

loginServlet的代码如下：

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

request.setCharacterEncoding("utf-8");

response.setContentType("text/html;charSet=utf-8");

String userName = request.getParameter("userName");

String password = request.getParameter("password");

String value = request.getParameter("admin");

if(userName.equals("")) {

request.setAttribute("msg", "用户名为空");

request.getRequestDispatcher("login.jsp").forward(request, response);

return;

}

if(password.equals("")) {

request.setAttribute("msg", "密码为空");

request.getRequestDispatcher("login.jsp").forward(request, response);

return;

}

Connection conn=Conn.getConn();

String sql = "select \* from User where userName=?";

String sql2 = "select \* from Admin where admin\_Name=?";

//管理员登录

if(value!=null){

String admin\_Name = userName;

try{

PreparedStatement pst = conn.prepareStatement(sql2);

pst.setString(1, admin\_Name);

ResultSet rs = pst.executeQuery();

if(rs.next()) {

if(!rs.getString("admin\_password").equals(password)){

request.setAttribute("msg", "密码错误");

request.getRequestDispatcher("login.jsp").forward(request, response);

return;

}

HttpSession se = request.getSession();

se.setAttribute("id", rs.getInt("id"));

se.setAttribute("userName", rs.getString("admin\_Name"));

se.setAttribute("level", rs.getInt("level"));

}else {

request.setAttribute("msg", "用户不存在");

request.getRequestDispatcher("login.jsp").forward(request, response);

return;

}

rs.close();

pst.close();

conn.close();

request.getRequestDispatcher("admin.jsp").forward(request, response);

}catch (SQLException e) {

// TODO: handle exception

e.printStackTrace();

}

}

//用户登录

if(value==null){

try {

PreparedStatement pst = conn.prepareStatement(sql);

pst.setString(1, userName);

ResultSet rs = pst.executeQuery();

if(rs.next()) {

if(rs.getInt("ban")==1){

request.setAttribute("msg", "该用户已被禁止访问");

request.getRequestDispatcher("login.jsp").forward(request, response);

return;

}

if(!rs.getString("password").equals(password)){

request.setAttribute("msg", "密码错误");

request.getRequestDispatcher("login.jsp").forward(request, response);

return;

}

HttpSession se = request.getSession();

se.setAttribute("id", rs.getInt("id"));

se.setAttribute("userName", rs.getString("userName"));

se.removeAttribute("level");

}else {

request.setAttribute("msg", "用户不存在");

request.getRequestDispatcher("login.jsp").forward(request, response);

return;

}

rs.close();

pst.close();

conn.close();

request.getRequestDispatcher("book\_view.jsp").forward(request, response);

}catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

}

登录界面效果图如下：



图 3.2.2.1登录界面图

#### 3.2.3 注册功能

注册功能界面通过register.jsp实现，功通过registerServlet实现。

register.jsp代码如下：

<body>

<div class="login">

<form action="registerSerlet" method="post">

<h1>注册</h1>

<div class="name">用户名:&nbsp;&nbsp;<input type="text" name="userName" value="${requestScope.userName }" /></div>

<br/>

<div class="pwd">密码:<input type="password" name="password1" /></div>

<br/>

<div class="pwd2">确认密码:<input type="password" name="password2"/></div>

<br/>

<div class="admin">管理员注册<input type="checkbox" name="admin" value="admin" /></div>

<br/>

<button class="bt-login">注册</button>

<br/><br/>

<a class="a1" href="login.jsp">返回登录</a>

<a class="a2" href="book\_view.jsp" >返回主页</a>

</form>

<p class="p1">${requestScope.msg }</p>

</div>

</body>

registerServlet的代码如下:

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// TODO Auto-generated method stub

// response.getWriter().append("Served at: ").append(request.getContextPath());

request.setCharacterEncoding("utf-8");

response.setContentType("text/html;charSet=utf-8");

String userName = request.getParameter("userName");

String password1 = request.getParameter("password1");

String password2 = request.getParameter("password2");

String value = request.getParameter("admin");

//密码不相同

if(!password1.equals(password2)) {

request.setAttribute("msg", "密码不一致");

request.getRequestDispatcher("register.jsp").forward(request, response);

return;

}

if(userName.equals("")) {

request.setAttribute("msg", "用户名为空");

request.getRequestDispatcher("register.jsp").forward(request, response);

return;

}

if(password1.equals("")) {

request.setAttribute("msg", "密码为空");

request.getRequestDispatcher("register.jsp").forward(request, response);

return;

}

Connection conn=Conn.getConn();

String sql = "select \* from User where userName=?";

String sql2 = "select \* from Admin where admin\_Name=?";

//管理员注册

if(value!=null){

String admin\_Name = userName;

try{

PreparedStatement pst = conn.prepareStatement(sql2);

pst.setString(1, admin\_Name);

ResultSet rs = pst.executeQuery();

if(rs.next()) {//存在

request.setAttribute("msg", "用户名已存在");

request.getRequestDispatcher("register.jsp").forward(request, response);

return;

}else {

sql2 = "insert into Admin values(null,?,?)";

pst = conn.prepareStatement(sql2);

pst.setString(1,admin\_Name);

pst.setString(2, password1);

if(pst.executeUpdate()>0) {

response.sendRedirect("login.jsp");

}else {

request.setAttribute("msg", "Error");

request.getRequestDispatcher("register.jsp").forward(request, response);

return;

}

}

pst.close();

rs.close();

conn.close();

}catch (SQLException e) {

// TODO: handle exception

e.printStackTrace();

}

}

//用户注册

if(value==null){

try {

PreparedStatement pst = conn.prepareStatement(sql);

pst.setString(1, userName);

ResultSet rs = pst.executeQuery();

if(rs.next()) {//存在

request.setAttribute("msg", "用户名已存在");

request.getRequestDispatcher("register.jsp").forward(request, response);

return;

}else {

sql = "insert into User values(null,?,?,0)";

pst = conn.prepareStatement(sql);

pst.setString(1,userName);

pst.setString(2, password1);

if(pst.executeUpdate()>0) {

response.sendRedirect("login.jsp");

}else {

request.setAttribute("msg", "Error");

request.getRequestDispatcher("register.jsp").forward(request, response);

return;

}

}

pst.close();

rs.close();

conn.close();

} catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

}

注册界面图如下：



图 3.2.3.1注册界面图

#### 3.2.3 图书展示功能

图书展示界面中设计了一个搜索框能够实现对图书的类型搜索，从而实现分类展示图书。

搜索框可以根据书名和类型查找到对应的图书，并能够展示出来。

图书展示界面book\_view.jsp代码如下：

<body>

<c:if test="${empty requestScope.tag }">

<c:redirect url="bookServlet" />

</c:if>

<div class="main">

<jsp:include page="index.jsp"/>

<div class="search-main">

<form action="searchServlet" method="post">

<div class="search">

<input class="input-1" type="text" name="search" placeholder="请输入关键字"/>

<button class="bt-1">查询</button>

<select class="st-1" name="select">

<option value="1" selected="selected">书名</option>

<option value="2">类别</option>

</select>

</div>

</form>

</div>

<br/>

<div class="main-show">

<ul>

<c:forEach items="${books}" var="b">

<li>

<a href="bookDetailServlet?bookid=+${b.bookId }"><img src="${b.bookFile }" width="200px" height="200px"/></a>

<br/>

<span>${b.bookName}</span>

<br/>

<span>${b.bookPrice }元</span>

<br/>

<div class="show-bt">

<button class="bt-2" onclick="sub(${b.bookId })">-</button>

<input class="input-2" type="text" id="${b.bookId }" name="num" value="1" />

<button class="bt-2" onclick="add(${b.bookId })">+</button>

<br/>

</div>

<button class="bt-3" onclick="jump(${b.bookId })">加入购物车</button>

</li>

</c:forEach>

</ul>

<p><b>${requestScope.msg }</b></p>

</div>

</div>

</body>

</html>

<script type="text/javascript">

function add(id) {

var num=document.getElementById(id).value;

num++;

document.getElementById(id).value=num;

}

function sub(id) {

var num=document.getElementById(id).value;

if(num>1){

num--;

}

document.getElementById(id).value=num;

}

function jump(id) {

var bookId=id;

var num=document.getElementById(id).value;

alert("添加成功")

window.location.href="cartServlet?bookId="+bookId+"&num="+num+"&flag="+0;

}

</script>

图书展示功能实现的bookServlet代码如下所示：

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// TODO Auto-generated method stub

//response.getWriter().append("Served at: ").append(request.getContextPath());

request.setCharacterEncoding("utf-8");

response.setContentType("text/html;charSet=utf-8");

HttpSession se = request.getSession();

Connection conn=Conn.getConn();

String sql = "select \* from Booklist ";

try{

PreparedStatement pst = conn.prepareStatement(sql);

ResultSet rs = pst.executeQuery();

List<Book> books = new ArrayList<Book>();

while(rs.next()) {

Book book = new Book();

book.setBookId(rs.getInt("bookId"));

book.setBookName(rs.getString("bookName"));

book.setBookPrice(rs.getDouble("bookPrice"));

book.setBookFile(rs.getString("bookFile"));

books.add(book);

}

request.setAttribute("books", books);

rs.close();

pst.close();

conn.close();

request.setAttribute("tag", "2");

request.getRequestDispatcher("book\_view.jsp").forward(request, response);

}catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

图书搜索同时通过搜索类别实现分类展示的searchServlet代码如下：

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// TODO Auto-generated method stub

//response.getWriter().append("Served at: ").append(request.getContextPath());

request.setCharacterEncoding("utf-8");

response.setContentType("text/html;charSet=utf-8");

String search=request.getParameter("search");

int select=Integer.parseInt(request.getParameter("select"));

Connection conn=Conn.getConn();

if(select==1){

String sql = "select \* from Booklist where bookName like '%' ? '%'";

try{

PreparedStatement pst = conn.prepareStatement(sql);

pst.setString(1, search);

ResultSet rs = pst.executeQuery();

if(rs.next()==false){

request.setAttribute("msg", "查找不到该书籍");

}

rs.absolute(0);

List<Book> books = new ArrayList<Book>();

while(rs.next()) {

Book book = new Book();

book.setBookId(rs.getInt("bookId"));

book.setBookName(rs.getString("bookName"));

book.setBookPrice(rs.getDouble("bookPrice"));

book.setBookFile(rs.getString("bookFile"));

books.add(book);

}

request.setAttribute("books", books);

rs.close();

pst.close();

conn.close();

request.setAttribute("tag", "2");

request.getRequestDispatcher("book\_view.jsp").forward(request, response);

}catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

if(select==2){

String sql = "select \* from Booklist where bookType like '%' ? '%'";

try{

PreparedStatement pst = conn.prepareStatement(sql);

pst.setString(1, search);

ResultSet rs = pst.executeQuery();

if(rs.next()==false){

request.setAttribute("msg", "查找不到该类别");

}

rs.absolute(0);

List<Book> books = new ArrayList<Book>();

while(rs.next()) {

Book book = new Book();

book.setBookId(rs.getInt("bookId"));

book.setBookName(rs.getString("bookName"));

book.setBookPrice(rs.getDouble("bookPrice"));

book.setBookFile(rs.getString("bookFile"));

books.add(book);

}

request.setAttribute("books", books);

rs.close();

pst.close();

conn.close();

request.setAttribute("tag", "2");

request.getRequestDispatcher("book\_view.jsp").forward(request, response);

}catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

}

图书展示的界面如下：



图 3.2.3.1图书展示界面

图书分类展示如下图：

小说类：



图 3.2.3.2小说类展示图

威士忌类展示：



图 3.2.3.3威士忌类展示图

图书书名搜索功能：



图 3.2.3.4图书书名搜索展示图

#### 3.2.5 购物车功能

购物车可以实现下单功能；以及删除购物车中的商品，还可以对已加入购物车中的商品进行修改数量的功能。

购物车界面book\_cart.jsp的代码如下：

<body>

<div class="index">

<h1 >图书销售网站</h1>

<p>欢迎您：

<c:choose>

<c:when test="${sessionScope.id !=null }">

<c:out value="${sessionScope.userName}"/>

<a href="login.jsp">切换用户</a>

</c:when>

<c:when test="${sessionScope.id ==null }">

<a href="login.jsp">登录</a>

</c:when>

</c:choose>

</p>

<div class="dh">

<ul id="nav" >

<li><a href="register.jsp">注册</a></li>

<li><a href="bookServlet">浏览图书</a></li>

<li><a href="cart\_showServlet">购物车</a></li>

<li><a href="myOrderServlet">我的订单</a></li>

<li><a href="admin.jsp">后台管理</a></li>

<li><a href="exit.jsp">退出登录</a></li>

</ul>

</div>

</div>

</body>

</html>

添加商品到购物车的功能使用cartServlet实现。具体代码如下：

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// TODO Auto-generated method stub

//response.getWriter().append("Served at: ").append(request.getContextPath());

request.setCharacterEncoding("utf-8");

response.setContentType("text/html;charSet=utf-8");

HttpSession se = request.getSession();

if(se.getAttribute("level")!=null){

request.setAttribute("msg", "请使用用户账号登录");

request.getRequestDispatcher("login.jsp").forward(request, response);

return;

}

if(se.getAttribute("id")==null){

response.sendRedirect("login.jsp");

return;

}

String bookId=request.getParameter("bookId");

String num = request.getParameter("num");

int flag=Integer.parseInt(request.getParameter("flag"));

int userid = (int)se.getAttribute("id");

Connection conn=Conn.getConn();

if(flag==1){

String sql = "update cart set number=? where bookid=? and userid=?";

try{

PreparedStatement pst = conn.prepareStatement(sql);

pst = conn.prepareStatement(sql);

pst.setString(1, num);

pst.setString(2, bookId);

pst.setInt(3, userid);

if(pst.executeUpdate()>0) {

pst.close();

conn.close();

response.sendRedirect("cart\_showServlet");

}

}catch (SQLException e) {

// TODO: handle exception

e.printStackTrace();

}

}

if(flag==0){

String sql = "select \* from Cart where bookid=? and userid=?";

try{

PreparedStatement pst = conn.prepareStatement(sql);

pst.setString(1, bookId);

pst.setInt(2, userid);

ResultSet rs = pst.executeQuery();

if(!rs.next()){

sql = "select \* from Booklist where bookid=?";

pst = conn.prepareStatement(sql);

pst.setString(1, bookId);

rs = pst.executeQuery();

rs.next();

int bookid=rs.getInt("bookid");

String bookName=rs.getString("bookName");

Double bookPrice=rs.getDouble("bookPrice");

String bookFile=rs.getString("bookFile");

sql = "insert into cart values(?,?,?,?,?,?)";

pst = conn.prepareStatement(sql);

pst.setInt(1, userid);

pst.setInt(2,bookid);

pst.setString(3, bookName);

pst.setDouble(4, bookPrice);

pst.setString(5, num);

pst.setString(6, bookFile);

pst.executeUpdate();

}else{

int number=rs.getInt("number")+Integer.parseInt(num);

sql = "update cart set number=? where bookid=? and userid=?";

pst = conn.prepareStatement(sql);

pst.setInt(1, number);

pst.setString(2, bookId);

pst.setInt(3, userid);

pst.executeUpdate();

}

rs.close();

pst.close();

conn.close();

request.getRequestDispatcher("bookServlet").forward(request, response);

}catch (SQLException e) {

// TODO: handle exception

e.printStackTrace();

}

}

if(flag==2){

String sql = "delete from Cart where bookid=?";

try{

PreparedStatement pst = conn.prepareStatement(sql);

pst.setString(1, bookId);

if(pst.executeUpdate()>0){

pst.close();

conn.close();

response.sendRedirect("cart\_showServlet");

}

}catch (SQLException e) {

// TODO: handle exception

e.printStackTrace();

}

}

}

将加入购物车的图书展示在购物车界面中的功能通过cart\_viewServlet实现

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// TODO Auto-generated method stub

//response.getWriter().append("Served at: ").append(request.getContextPath());

request.setCharacterEncoding("utf-8");

response.setContentType("text/html;charSet=utf-8");

HttpSession se = request.getSession();

if(se.getAttribute("level")!=null){

request.setAttribute("msg", "请使用用户账号登录");

request.getRequestDispatcher("login.jsp").forward(request, response);

return;

}

if(se.getAttribute("id")==null){

request.setAttribute("msg", "请登录");

request.getRequestDispatcher("login.jsp").forward(request, response);

return;

}

int userid=(int)se.getAttribute("id");

Connection conn=Conn.getConn();

String sql = "select \* from Cart where userid=?";

try{

PreparedStatement pst = conn.prepareStatement(sql);

pst.setInt(1, userid);

ResultSet rs = pst.executeQuery();

List<Cart> carts = new ArrayList<Cart>();

while(rs.next()) {

Cart cart = new Cart();

cart.setUserid(rs.getInt("userid"));

cart.setBookid(rs.getInt("bookid"));

cart.setBookName(rs.getString("bookName"));

cart.setBookPrice(rs.getDouble("bookPrice"));

cart.setBookNumber(rs.getInt("number"));

cart.setBookFile(rs.getString("bookFile"));

carts.add(cart);

}

request.setAttribute("carts", carts);

rs.close();

pst.close();

conn.close();

request.getRequestDispatcher("book\_cart.jsp").forward(request, response);

}catch (SQLException e) {

// TODO: handle exception

e.printStackTrace();

}

}

购物车的界面如下图：

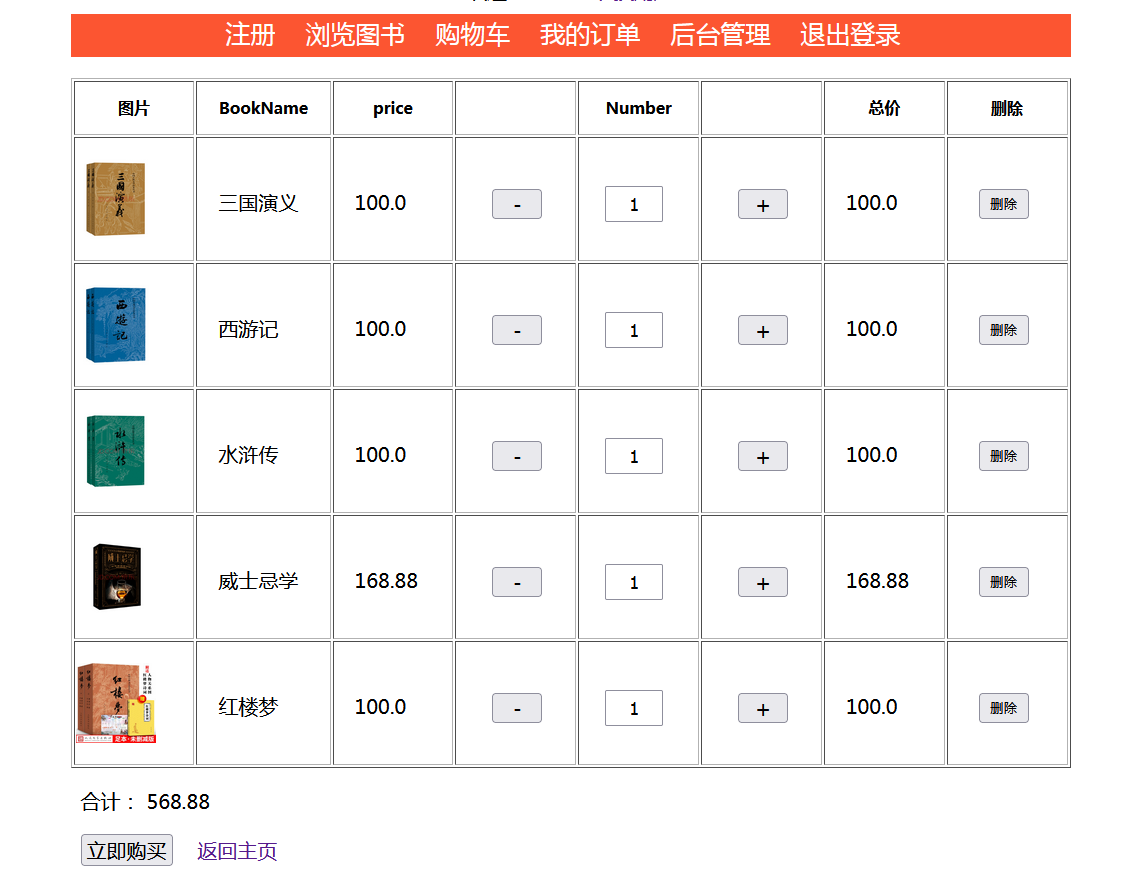


图 3.2.5.1购物车界面图

#### 3.2.6 我的订单功能

用户可以在我的订单中查看该用户已经生成的订单已经，每个订单里面商品的详细情况。

我的订单界面分成两个部分一个为总的订单界面，一个为订单的详细界面。在详情页面中可以对商品进行退款操作。如果该订单的所有商品均被退款，则订单状态显示为已取消。

总的订单界面为order.jsp代码如下：

<body>

<div class="main-order">

<jsp:include page="index.jsp"/><br/>

<a class="a1" href="book\_view.jsp">返回主页</a>

<table border="1">

<tr>

<th>OrderId</th>

<th>UserID</th>

<th>Sum</th>

<th>State</th>

<th>详情</th>

</tr>

<c:forEach items="${orders}" var="o">

<tr>

<td>${o.orderId }</td>

<td>${o.userId }</td>

<td>${o.sum }</td>

<td>${o.state }</td>

<td><button name="bt" value="1" onclick="myorder(${o.orderId })">查看详情</button></td>

</tr>

</c:forEach>

</table>

<p class="p1">${requestScope.msg }</p>

</div>

</body>

</html>

<script>

function myorder(orderId) {

var id=orderId

window.location.href="orderListServlet?orderId="+id;

}

</script>

订单详情界面为myorderlist.jsp，代码如下：

<body>

<div class="main-orderlist">

<jsp:include page="index.jsp"/><br/>

<c:set var="sum" value="0.00"/>

<c:set var="i" value="0"/>

<c:forEach items="${ols}" var="ol">

<form action="orderListServlet" method="get" >

<table border="1" >

<c:if test="${i==0 }" >

<tr>

<th>图片</th>

<th>OrderId</th>

<th>UserID</th>

<th>BookName</th>

<th>BookPrice</th>

<th>number</th>

<th>申请退款</th>

</tr>

</c:if>

<tr>

<td><a href=""><img src="${ol.bookFile }" width="80px" height="80px"/></a></td>

<td><input class="input-list" type="text" name="orderId" value="${ol.orderId }" readonly/></td>

<td><input class="input-list" type="text" name="userId" value="${ol.userId }" readonly/></td>

<td><input class="input-list" type="text" name="bookName" value="${ol.bookName }" readonly/></td>

<td><input class="input-list" type="text" name="bookPrice" value="${ol.bookPrice }" readonly/></td>

<td><input class="input-list" type="text" name="number" value="${ol.number }" readonly/></td>

<td><button name="bt" value="0">申请退款</button></td>

</tr>

</table>

</form>

<c:set var="i" value="${i+1 }"/>

</c:forEach>

<a class="a1" href="myOrderServlet">返回订单主页</a>

<p class="p1">${requestScope.msg }</p>

</div>

</body>

订单界面的展示功能通过myOrderServlet实现,查看订单详情通过orderListServlet实现具体代码与购物车以及图书展示的功能类似，因此不再给出。

我的订单界面如下图：



图3.2.6.1我的订单页面图

查看详情的界面如下图：



图 3.2.6.2订单详情页面图

下单功能通过orderServlet实现具体代码如下：

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// TODO Auto-generated method stub

//response.getWriter().append("Served at: ").append(request.getContextPath());

int tag=0;

request.setCharacterEncoding("utf-8");

response.setContentType("text/html;charSet=utf-8");

String sum=request.getParameter("sum");

HttpSession se = request.getSession();

int userId=(int)se.getAttribute("id");

Date date=new Date();

SimpleDateFormat sdf=new SimpleDateFormat("yyyyMMddHHmmss");

String orderId=sdf.format(date);

Connection conn=Conn.getConn();

String sql = "select \* from Cart where userid=?";

try{

PreparedStatement pst = conn.prepareStatement(sql);

pst.setInt(1, userId);

ResultSet rs = pst.executeQuery();

if(!rs.next()){

request.setAttribute("msg", "购物车为空");

pst.close();

conn.close();

rs.close();

request.getRequestDispatcher("cart\_showServlet").forward(request, response);

return;

}

rs.absolute(0);

if(rs.next()){

sql = "insert into orderform values(?,?,?,'未发货')";

try{

pst = conn.prepareStatement(sql);

pst.setString(1,orderId);

pst.setInt(2, userId);

pst.setString(3, sum);

if(pst.executeUpdate()>0) {

//查购物车表获取图书信息

sql = "select \* from Cart where userId=?";

pst = conn.prepareStatement(sql);

pst.setInt(1,userId);

rs = pst.executeQuery();

while(rs.next()){

String bookName=rs.getString("bookName");

String bookPrice=rs.getString("bookPrice");

String number=rs.getString("number");

String bookFile=rs.getString("bookFile");

sql = "insert into orderlist values(null,?,?,?,?,?,?)";

pst = conn.prepareStatement(sql);

pst.setString(1, orderId);

pst.setInt(2,userId);

pst.setString(3, bookName);

pst.setString(4, bookPrice);

pst.setString(5, number);

pst.setString(6, bookFile);

tag=pst.executeUpdate();

}

if(tag>0){

sql = "delete from Cart where userid=?";

pst = conn.prepareStatement(sql);

pst.setInt(1,userId);

if(pst.executeUpdate()>0){

pst.close();

rs.close();

conn.close();

request.setAttribute("msg", "购买成功");

request.getRequestDispatcher("cart\_showServlet").forward(request, response);

}

}

}else {

rs.close();

pst.close();

conn.close();

request.setAttribute("msg", "Error");

request.getRequestDispatcher("cart\_showServlet").forward(request, response);

return;

}

}catch (SQLException e) {

// TODO: handle exception

e.printStackTrace();

}

}else{

rs.close();

pst.close();

conn.close();

request.setAttribute("msg", "订单已存在");

request.getRequestDispatcher("cart\_showServlet").forward(request, response);

return;

}

}catch (SQLException e) {

// TODO: handle exception

e.printStackTrace();

}

}

退款操作是通过orderListServlet实现，即对orderlist表进行删除操作，再根据orderId重新展示订单的详情。由于代码类似，不再展示。

#### 3.2.7 图书详情页设计

图书详情界面通过bookDetail.jsp和bookDetailServlet实现，具体代码与图书展示的类似，下图为图书详情界面图：



图表 3.2.7.1图书详情界面图

### 3. 3 后端功能

后端功能实现用户管理模块：对用户进行权限禁止和开启，被禁止的用户无法登陆，无法使用购物车和订单只能浏览商品。能够对用户进行增删改查操作。

图书管理模块：可以对图书信息进行增删改查操作，可以上传图书图片。

订单管理模块：可以查看所有用户的订单以及订单详情，可以修改订单的状态。

#### 3.3.1 后台管理导航栏

实现方式与商城的导航栏相同，通过admin.jsp实现，在进入后台管理主页时可判断该用户身份是否为管理员，若否，则跳转到登录界面。由于其他部分代码类似，此处不给出。验证身份的代码如下：

<p>欢迎您：

<c:choose>

<c:when test="${sessionScope.level !=null }">

<c:out value="${sessionScope.userName}"/>

<a href="login.jsp">切换用户</a>

</c:when>

<c:when test="${sessionScope.level ==null }">

<c:redirect url="login.jsp"/>

</c:when>

</c:choose>

，效果图如下：



图 3.3.1.1管理界面导航栏图

#### 3.3.2 用户管理功能

用户管理界面通过userManage.jsp和useManageServlet实现，由于展示界面的代码具类似，此处不给出。只给出效果图：



图 3.3.2.1用户管理界面图

对于用户的搜索，使用user\_searchServlet实现，可以通过用户名和ID搜索，代码与图书搜索相同，此处不给出。只展示搜素的效果图。



图 3.3.2.2用户管理搜索结果图

对于用户的增删改查，以及权限设置通过userChangeServlet实现，其中每个按钮都对应不同的功能且有一个对应的name和value；通过点击按钮提交form到userChangeServlet中进行判断然后对数据库执行对应的操作。具体代码如下：

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// TODO Auto-generated method stub

//response.getWriter().append("Served at: ").append(request.getContextPath());

request.setCharacterEncoding("utf-8");

response.setContentType("text/html;charSet=utf-8");

String id = request.getParameter("id");

String userName = request.getParameter("userName");

String password = request.getParameter("password");

int bt =Integer.parseInt(request.getParameter("bt"));

if(userName.equals("")) {

request.setAttribute("msg", "用户名为空");

request.getRequestDispatcher("userManageServlet").forward(request, response);

return;

}

if(password.equals("")) {

request.setAttribute("msg", "密码为空");

request.getRequestDispatcher("userManageServlet").forward(request, response);

return;

}

if(bt==1){

Connection conn=Conn.getConn();

String sql = "update User set ban=1 where id=?";

try{

PreparedStatement pst = conn.prepareStatement(sql);

pst.setString(1, id);

if(pst.executeUpdate()>0) {

pst.close();

conn.close();

response.sendRedirect("userManageServlet");

}

}catch (SQLException e) {

// TODO: handle exception

e.printStackTrace();

}

}

if(bt==2){

Connection conn=Conn.getConn();

String sql = "update User set ban=0 where id=?";

try{

PreparedStatement pst = conn.prepareStatement(sql);

pst.setString(1, id);

if(pst.executeUpdate()>0) {

pst.close();

conn.close();

response.sendRedirect("userManageServlet");

}

}catch (SQLException e) {

// TODO: handle exception

e.printStackTrace();

}

}

if(bt==3){

Connection conn=Conn.getConn();

String sql = "update User set userName=? , password=? where id=?";

try{

PreparedStatement pst = conn.prepareStatement(sql);

pst.setString(1, userName);

pst.setString(2, password);

pst.setString(3, id);

if(pst.executeUpdate()>0){

pst.close();

conn.close();

response.sendRedirect("userManageServlet");

}

}catch (SQLException e) {

// TODO: handle exception

e.printStackTrace();

}

}

if(bt==4){

Connection conn=Conn.getConn();

String sql = "delete from User where id=?";

try{

PreparedStatement pst = conn.prepareStatement(sql);

pst.setString(1, id);

if(pst.executeUpdate()>0){

pst.close();

conn.close();

response.sendRedirect("userManageServlet");

}

}catch (SQLException e) {

// TODO: handle exception

e.printStackTrace();

}

}

if(bt==5){

Connection conn=Conn.getConn();

String sql = "select \* from User where id=?";

try{

PreparedStatement pst = conn.prepareStatement(sql);

pst.setString(1, id);

ResultSet rs = pst.executeQuery();

if(rs.next()) {//存在

request.setAttribute("msg", "ID已存在");

request.getRequestDispatcher("userManageServlet").forward(request, response);

return;

}else{

sql = "select \* from User where userName=?";

pst = conn.prepareStatement(sql);

pst.setString(1,userName);

rs = pst.executeQuery();

if(rs.next()) {//存在

request.setAttribute("msg", "用户名已存在");

request.getRequestDispatcher("userManageServlet").forward(request, response);

return;

}else{

sql = "insert into User values(?,?,?,0)";

pst = conn.prepareStatement(sql);

pst.setString(1,id);

pst.setString(2, userName);

pst.setString(3, password);

if(pst.executeUpdate()>0) {

request.setAttribute("msg", "添加成功");

request.getRequestDispatcher("userManageServlet").forward(request, response);

}else {

request.setAttribute("msg", "Error");

request.getRequestDispatcher("userManageServlet").forward(request, response);

return;

}

}

}

pst.close();

rs.close();

conn.close();

}catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

}

禁止用户访问的效果图如下：

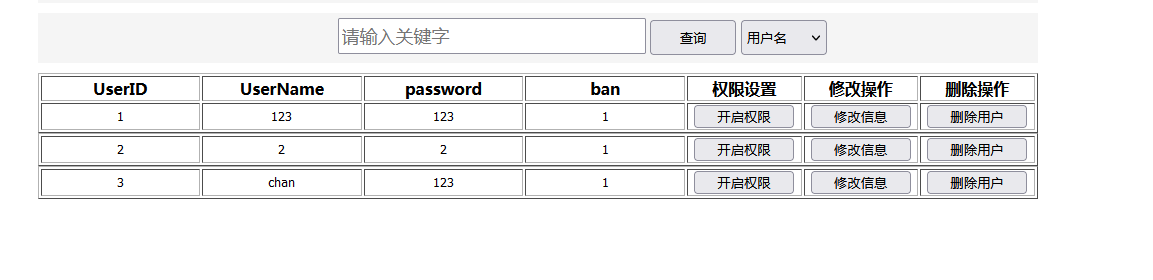


图 3.3.2.3禁止用户权限图

添加新用户：



图 3.3.2.4添加用户图

修改用户：



图 3.3.2.5修改用户信息图

删除用户：



图 3.3.2.6删除用户图

#### 3.3.3 图书管理功能

图书管理界面通过bookManage.jsp和bookManageServlet实现，由于展示界面的代码具类似，此处不给出。对于图书管理页面中的搜索，使用bookManagesearchServlet实现，可以通过书名、ID、作者以及类别搜索，代码与searchServlet相同，此处不给出。下面为图书管理页面的效果图。



图 3.3.3.1图书管理界面图

在图书管理功能模块中对于图书的增删改查bookChangeServlet实现，其中每个按钮都对应不同的功能且有一个对应的name和value；通过点击按钮提交form到bookChangeServlet中进行判断然后对数据库执行对应的操作。代码与用户管理类似此处不给出。只给出其中添加图书的文件上传功能的代码，具体代码如下：

if(bt==3){

//图片上传

Part part=request.getPart("photo");

long len =part.getSize();

String fileName=null;

if(len>0){

String s=part.getHeader("Content-Disposition");

int pos=s.lastIndexOf(".");

String ext="";//后缀名

if(pos>0){

ext=s.substring(pos,s.length()-1);

}

fileName=bookId+ext;

part.write(getServletContext().getRealPath("/")+fileName);

System.out.println(getServletContext().getRealPath("/")+fileName);

}

#### 3.3.4 订单管理功能

订单管理界面通过orderManage.jsp和orderManageServlet实现，其中在订单管理页面中查看订单详细信息也是通过orderManageServlet实现。由于展示界面的代码具类似，此处不给出。对于订单管理页面中的搜索，使用ordersearchServlet实现，可以通过订单号和用户ID进行搜索，代码与searchServlet相同，此处不给出。下面为订单管理页面的效果图。



图 3.3.4.1订单管理页面图

订单管理页面查看订单详情：



图 3.3.4.2订单管理页面订单详情图

### 3. 4 退出功能

点击导航栏的退出即可返回商场主页面同时清除所有的session。

Exit.jsp的代码如下：

<body>

<%request.getSession().invalidate(); response.sendRedirect("book\_view.jsp"); %>

</body>

## 设计小结

通过本次实验我能够总结、整理和复习这个学期所学的JSP内容。同时也更好的掌握了MVC的模式的开发，以及对数据库的操作。理解并掌握了Servlet以及JSP操作数据库的方法。还有如何使用JSTL和EL去替换JSP中的JAVA代码，能够把之前所学的内容综合起来。

## 自我评价

这次的设计我觉得自己对Servlet的运用以及对数据库的操作上做的不错，要求的功能也都能够很好的实现，还在要求的基础上完善了退款以及修改订单状态的功能。 不足的是由于时间关系没能够把界面设计的很美观。整体上看我对于本学期所学的JSP的内容掌握的还是不错。