

《Java高级应用开发》期末

课程设计

|  |  |
| --- | --- |
| 题 目 | 书城 |
|  |  |
| 专 业 | 计算机科学与技术 |
| 院 系 | 计算机科学学院 |
| 学 号 | 210015511046 |
| 姓 名 | 刘国栋 |
| 任 课 教 师 | 杨歆 |

**计算机科学学院**

**《书城》设计报告**

一、设计内容

实现书城用户购买图书，加入购物车，生成订单，商家（管理员）接受订单，发货，用户收货，登录功能和注册功能，加入验证码验证

二、设计过程

1、首页设计

（1） 效果图（成品截图）



（2）设计理念和方法

书城首页，可以看到商品，购物加入购物车，生成订单需登陆注册，使用jsp,后台绑定servlet，实现书本分页查询，以及按照价格范围进行查询，后台管理需要管理员登陆，登陆成功后可进行购买加入购物车，以及首页显示登陆姓名

## 源代码

1. **<**%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %**>**
2. **<**%@ page contentType="text/html;charset=UTF-8" language="java" %**>**
3. <!DOCTYPE html**>**
4. **<html>**
5. **<head>**
6. **<meta** charset="UTF-8"**>**
7. **<title>**书城首页**</title>**
9. **<**%-- 静态包含 base标签、css样式、jQuery文件 --%**>**
10. **<**%@ include file="/pages/common/head.jsp"%**>**
11. **<Script** type="text/javascript"**>**
12. $(function () {
13. // 给加入购物车按钮绑定单击事件
14. $("button.addToCart").click(function () {
15. /\*\*
16. \* 在事件响应的function函数 中，有一个this对象，这个this对象，是当前正在响应事件的dom对象
17. \* @type {jQuery}
18. \*/
19. var bookId = $(this).attr("bookId");
20. // location.href = "http://localhost:8080/book/cartServlet?action=addItem&id=" + bookId;
22. // 发ajax请求，添加商品到购物车
23. $.getJSON("http://localhost:8080/book/cartServlet","action=ajaxAddItem&id=" + bookId,function (data) {
24. $("#cartTotalCount").text("您的购物车中有 " + data.totalCount + " 件商品");
25. $("#cartLastName").text(data.lastName);
26. })
27. });
28. });
29. **</Script>**
31. **</head>**
32. **<body>**
34. **<div** id="header"**>**
35. **<img** class="logo\_img" alt="" src="/static/img/logo.gif" **>**
36. **<span** class="wel\_word"**>**网上书城**</span>**
37. **<div>**
38. **<**%--如果用户还没有登录，显示     【登录 和注册的菜单】 --%**>**
39. **<c:if** test="${empty sessionScope.user}"**>**
40. **<a** href="pages/user/login.jsp"**>**登录**</a>** |
41. **<a** href="pages/user/regist.jsp"**>**注册**</a>**
42. **</c:if>**
43. **<**%--如果已经登录，则显示 登录 成功之后的用户信息。--%**>**
44. **<c:if** test="${not empty sessionScope.user}"**>**
45. **<span>**欢迎**<span** class="um\_span"**>**${sessionScope.user.username}**</span>**光临书城**</span>**
46. **<a** href="orderServlet?action=queryAllOrders&userId=${sessionScope.user.id}"**>**我的订单**</a>**
47. **<a** href="userServlet?action=logout"**>**注销**</a>**
48. **</c:if>**
50. **<a** href="pages/cart/cart.jsp"**>**购物车**</a>**
51. **<a** href="pages/manager/manager.jsp"**>**后台管理**</a>**
52. **</div>**
53. **</div>**
55. **<div** id="main"**>**
56. **<div** id="book"**>**
57. **<div** class="book\_cond"**>**
58. **<form** action="client/bookServlet" method="get"**>**
59. **<input** type="hidden" name="action" value="pageByPrice"**>**
60. 价格：**<input** id="min" type="text" name="min" value="${param.min}"**>** 元 -
61. **<input** id="max" type="text" name="max" value="${param.max}"**>** 元
62. **<input** type="submit" value="查询" **/>**
63. **</form>**
64. **</div>**
65. **<div** style="text-align: center"**>**
66. **<c:if** test="${empty sessionScope.cart.items}"**>**
67. **<**%--购物车为空的输出--%**>**
68. **<span** id="cartTotalCount"**>** **</span>**
69. **<div>**
70. **<span** style="color: red" id="cartLastName"**>**当前购物车为空**</span>**
71. **</div>**
72. **</c:if>**
73. **<c:if** test="${not empty sessionScope.cart.items}"**>**
74. **<**%--购物车非空的输出--%**>**
75. **<span** id="cartTotalCount"**>**您的购物车中有 ${sessionScope.cart.totalCount} 件商品**</span>**
76. **<div>**
77. 您刚刚将**<span** style="color: red" id="cartLastName"**>**${sessionScope.lastName}**</span>**加入到了购物车中
78. **</div>**
79. **</c:if>**
80. **</div>**
82. **<c:forEach** items="${requestScope.page.items}" var="book"**>**
83. **<div** class="b\_list"**>**
84. **<div** class="img\_div"**>**
85. **<img** class="book\_img" alt="" src="${book.imgPath}" **/>**
86. **</div>**
87. **<div** class="book\_info"**>**
88. **<div** class="book\_name"**>**
89. **<span** class="sp1"**>**书名:**</span>**
90. **<span** class="sp2"**>**${book.name}**</span>**
91. **</div>**
92. **<div** class="book\_author"**>**
93. **<span** class="sp1"**>**作者:**</span>**
94. **<span** class="sp2"**>**${book.author}**</span>**
95. **</div>**
96. **<div** class="book\_price"**>**
97. **<span** class="sp1"**>**价格:**</span>**
98. **<span** class="sp2"**>**￥${book.price}**</span>**
99. **</div>**
100. **<div** class="book\_sales"**>**
101. **<span** class="sp1"**>**销量:**</span>**
102. **<span** class="sp2"**>**${book.sales}**</span>**
103. **</div>**
104. **<div** class="book\_amount"**>**
105. **<span** class="sp1"**>**库存:**</span>**
106. **<span** class="sp2"**>**${book.stock}**</span>**
107. **</div>**
108. **<div** class="book\_add"**>**
109. **<button** bookId="${book.id}" class="addToCart"**>**加入购物车**</button>**
110. **</div>**
111. **</div>**
112. **</div>**
113. **</c:forEach>**
114. **</div>**
116. **<**%--静态包含分页条--%**>**
117. **<**%@include file="/pages/common/page\_nav.jsp"%**>**



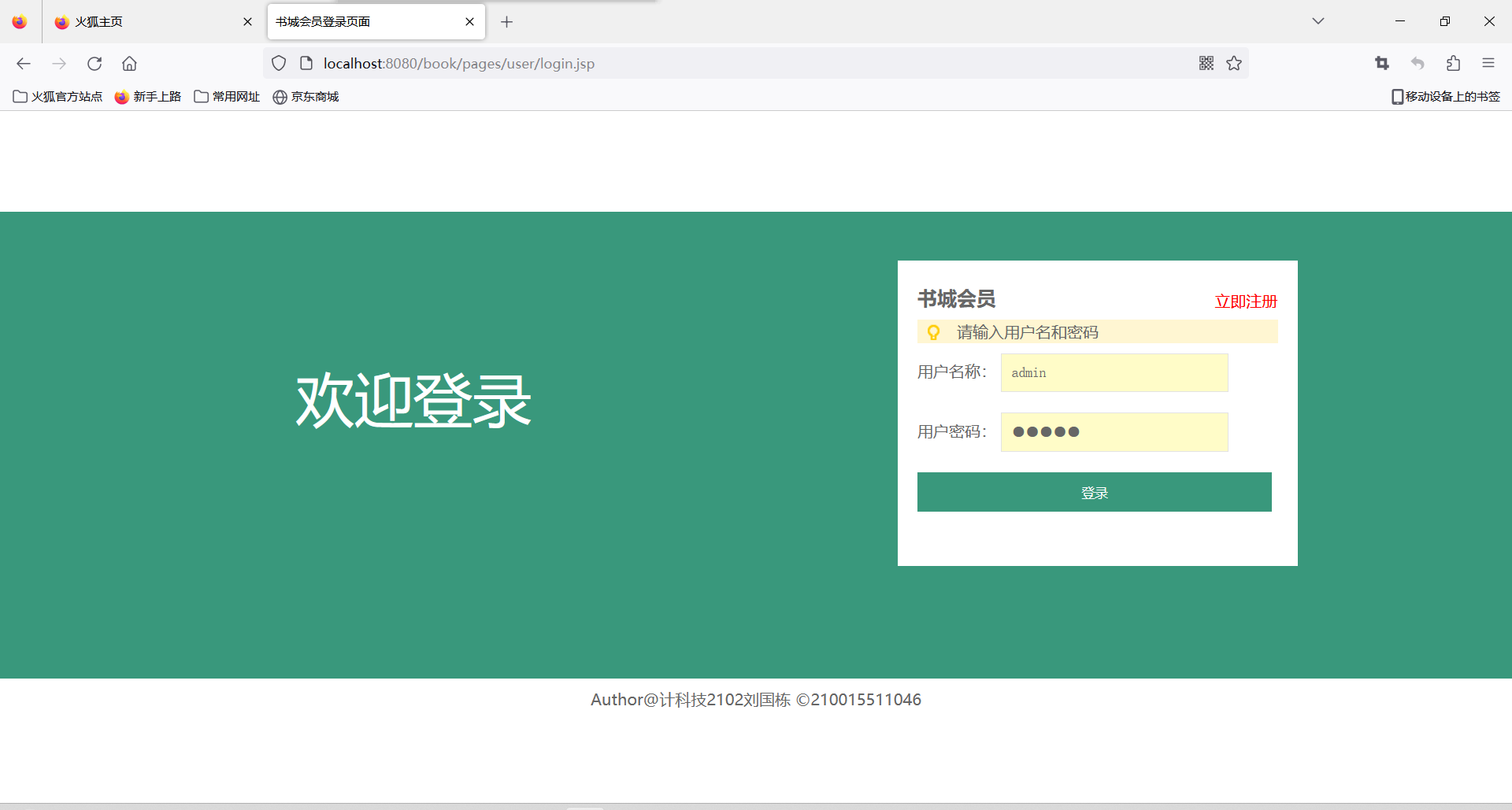

123. **</div>**
125. **<**%--静态包含页脚内容--%**>**
126. **<**%@include file="/pages/common/footer.jsp"%**>**
128. **</body>**
129. **</html>**

（4）必要的说明

该页面作为主要引入页面

2、登录页面

（1） 效果图（成品截图）



（2）设计理念和方法

表格表单绑定servlet事件，将登陆名传到session域主页面显示登录名

（3）源代码

1. **<**%@ page contentType="text/html;charset=UTF-8" language="java" %**>**
2. <!DOCTYPE html**>**
3. **<html>**
4. **<head>**
5. **<meta** charset="UTF-8"**>**
6. **<title>**书城会员登录页面**</title>**
8. **<**%-- 静态包含 base标签、css样式、jQuery文件 --%**>**
9. **<**%@ include file="/pages/common/head.jsp"%**>**

12. **</head>**
13. **<body>**
14. **<div** id="login\_header"**>**
15. **</div>**
17. **<div** class="login\_banner"**>**
19. **<div** id="l\_content"**>**
20. **<span** class="login\_word"**>**欢迎登录**</span>**
21. **</div>**
23. **<div** id="content"**>**
24. **<div** class="login\_form"**>**
25. **<div** class="login\_box"**>**
26. **<div** class="tit"**>**
27. **<h1>**书城会员**</h1>**
28. **<a** href="pages/user/regist.jsp"**>**立即注册**</a>**
29. **</div>**
30. **<div** class="msg\_cont"**>**
31. **<b></b>**
32. **<span** class="errorMsg"**>**
33. ${ empty requestScope.msg ? "请输入用户名和密码":requestScope.msg }
34. **</span>**
35. **</div>**
36. **<div** class="form"**>**
37. **<form** action="userServlet" method="post"**>**
38. **<input** type="hidden" name="action" value="login" **/>**
39. **<label>**用户名称：**</label>**
40. **<input** class="itxt" type="text" placeholder="请输入用户名"
41. autocomplete="off" tabindex="1" name="username"
42. value="${requestScope.username}" **/>**
43. **<br** **/>**
44. **<br** **/>**
45. **<label>**用户密码：**</label>**
46. **<input** class="itxt" type="password" placeholder="请输入密码"
47. autocomplete="off" tabindex="1" name="password" **/>**
48. **<br** **/>**
49. **<br** **/>**
50. **<input** type="submit" value="登录" id="sub\_btn" **/>**
51. **</form>**
52. **</div>**
54. **</div>**
55. **</div>**
56. **</div>**
57. **</div>**
59. **<**%--静态包含页脚内容--%**>**
60. **<**%@include file="/pages/common/footer.jsp"%**>**

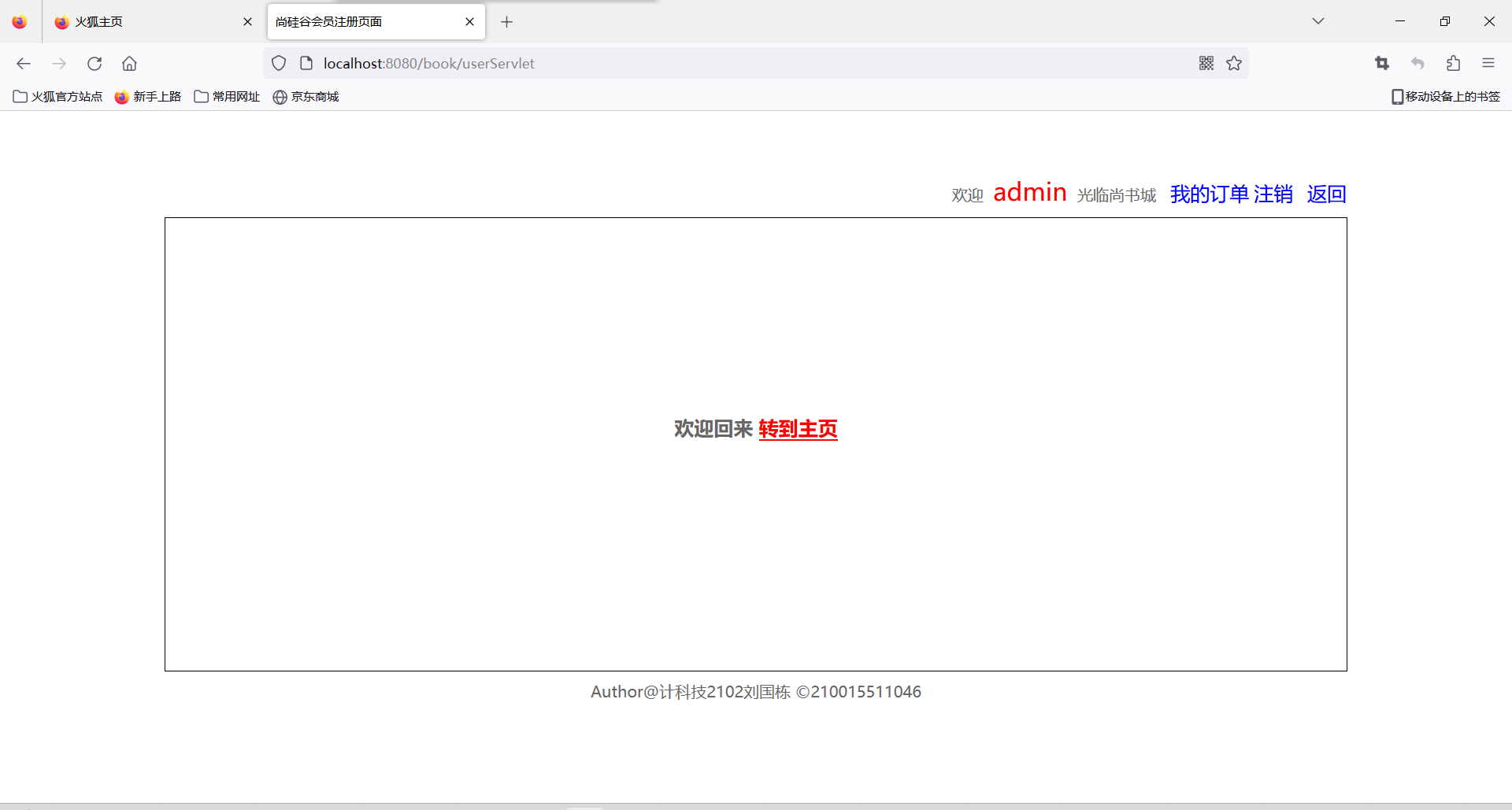
63. **</body>**
64. **</html>**

（4）必要的说明

登录页面作为用户和管理员首先都会看到的页面，如果成功跳到登录成功页面，失败则显示用户名或密码错误

1. 登陆成功

（1） 效果图



（2）设计理念和方法

中转页面，也可作为用户个人信息页面，~~还没设计~~

（3）源代码

1. **<**%@ page contentType="text/html;charset=UTF-8" language="java" %**>**
2. <!DOCTYPE html**>**
3. **<html>**
4. **<head>**
5. **<meta** charset="UTF-8"**>**
6. **<title>**会员注册页面**</title>**
8. **<**%-- 静态包含 base标签、css样式、jQuery文件 --%**>**
9. **<**%@ include file="/pages/common/head.jsp"%**>**

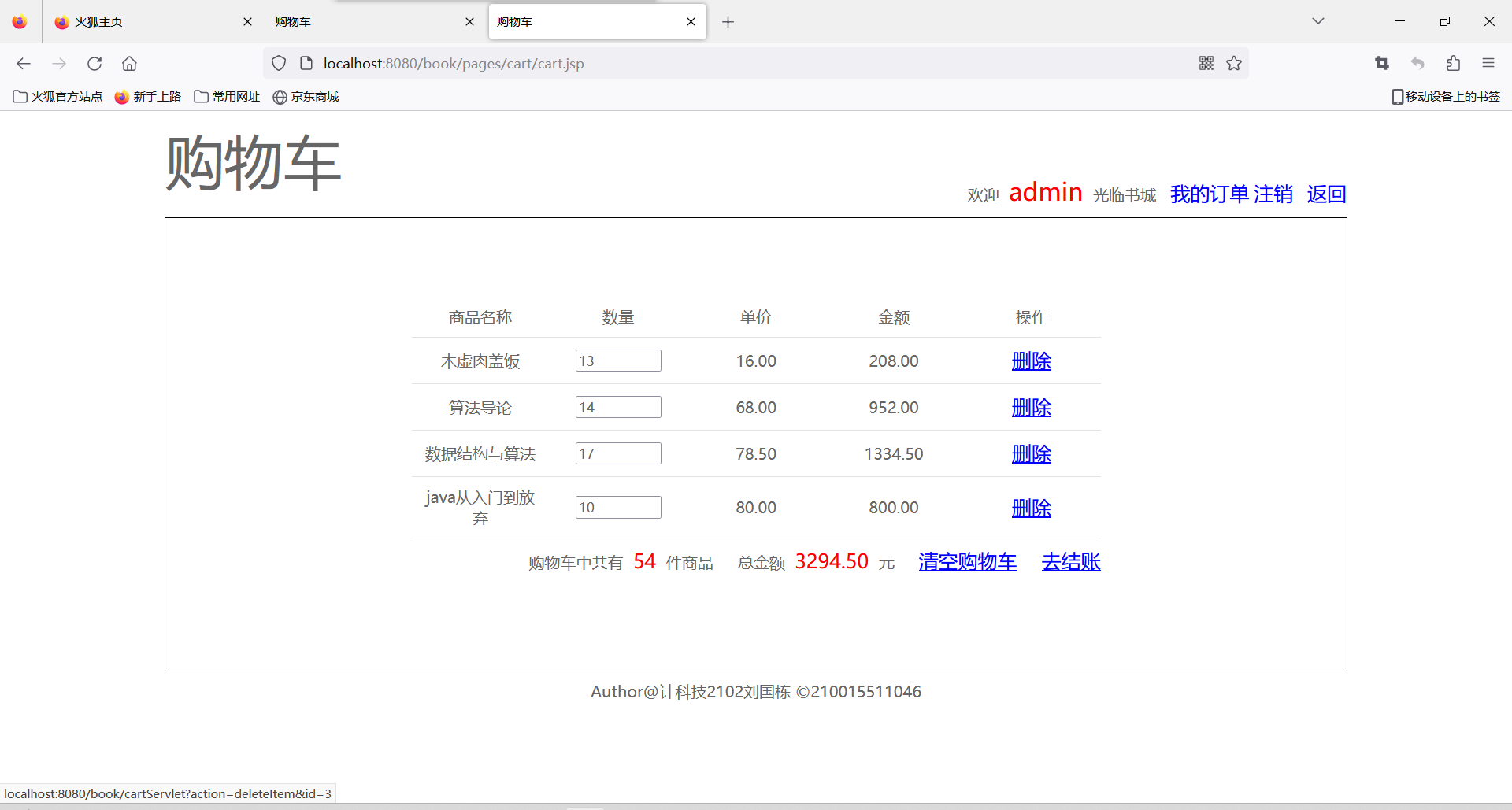
12. **<style** type="text/css"**>**
13. h1 {
14. text-align: center;
15. margin-top: 200px;
16. }
18. h1 a {
19. color:red;
20. }
21. **</style>**
22. **</head>**
23. **<body>**
24. **<div** id="header"**>**
25. **<img** class="logo\_img" alt="" src="static/img/logo.gif" **>**
27. **<**%--静态包含，登录 成功之后的菜单 --%**>**
28. **<**%@ include file="/pages/common/login\_success\_menu.jsp"%**>**

31. **</div>**
33. **<div** id="main"**>**
35. **<h1>**欢迎回来 **<a** href="index.jsp"**>**转到主页**</a></h1>**
37. **</div>**

40. **<**%--静态包含页脚内容--%**>**
41. **<**%@include file="/pages/common/footer.jsp"%**>**

44. **</body>**
45. **</html>**
46. 购物车界面

（1） 效果图



1. 设计理念和方法

主页加入图书后可在该页面显示，删除书，修改数量和清空购物车都会触发js弹窗，点击结账会生成相应的订单号，并在管理员界面显示

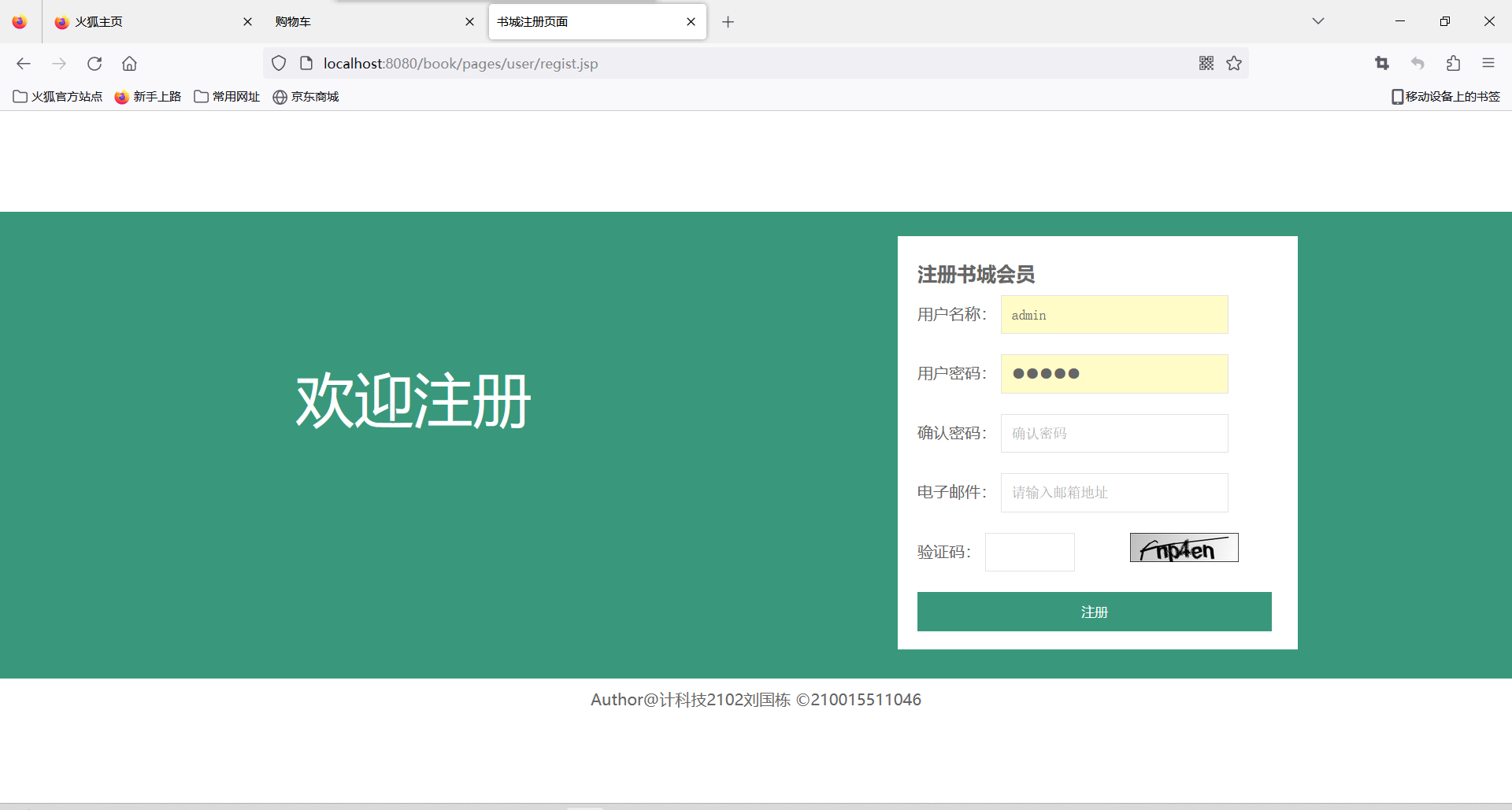
1. 源代码
2. **<**%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %**>**
3. **<**%@ page contentType="text/html;charset=UTF-8" language="java" %**>**
4. <!DOCTYPE html**>**
5. **<html>**
6. **<head>**
7. **<meta** charset="UTF-8"**>**
8. **<title>**购物车**</title>**
10. **<**%-- 静态包含 base标签、css样式、jQuery文件 --%**>**
11. **<**%@ include file="/pages/common/head.jsp"%**>**

14. **</head>**
15. **<body>**
16. **<div** id="header"**>**
17. **<span** class="wel\_word"**>**购物车**</span>**
19. **<**%--静态包含，登录 成功之后的菜单 --%**>**
20. **<**%@ include file="/pages/common/login\_success\_menu.jsp"%**>**
21. **<script** type="text/javascript"**>**
22. $(function () {
23. // 给 【删除】绑定单击事件
24. $("a.deleteItem").click(function () {
25. return confirm("你确定要删除【" + $(this).parent().parent().find("td:first").text() +"】吗?")
26. });
27. // 给清空购物车绑定单击事件
28. $("#clearCart").click(function () {
29. return confirm("你确定要清空购物车吗?");
30. })
31. // 给输入框绑定 onchange内容发生改变事件
32. $(".updateCount").change(function () {
33. // 获取商品名称
34. var name = $(this).parent().parent().find("td:first").text();
35. var id = $(this).attr('bookId');
36. // 获取商品数量
37. var count = this.value;
38. if ( confirm("你确定要将【" + name + "】商品修改数量为：" + count + " 吗?") ) {
39. //发起请求。给服务器保存修改
40. location.href = "http://localhost:8080/book/cartServlet?action=updateCount&count="+count+"&id="+id;
41. } else {
42. // defaultValue属性是表单项Dom对象的属性。它表示默认的value属性值。
43. thisthis.value = this.defaultValue;
44. }
45. });
47. });
48. **</script>**
50. **</div>**
52. **<div** id="main"**>**
54. **<table>**
55. **<tr>**
56. **<td>**商品名称**</td>**
57. **<td>**数量**</td>**
58. **<td>**单价**</td>**
59. **<td>**金额**</td>**
60. **<td>**操作**</td>**
61. **</tr>**
62. **<c:if** test="${empty sessionScope.cart.items}"**>**
63. **<**%--如果购物车空的情况--%**>**
64. **<tr>**
65. **<td** colspan="5"**><a** href="index.jsp"**>**亲，当前购物车为空！快跟小伙伴们去浏览商品吧！！！**</a>** **</td>**
66. **</tr>**
67. **</c:if>**
68. **<c:if** test="${not empty sessionScope.cart.items}"**>**
69. **<**%--如果购物车非空的情况--%**>**
70. **<c:forEach** items="${sessionScope.cart.items}" var="entry"**>**
71. **<tr>**
72. **<td>**${entry.value.name}**</td>**
73. **<td>**
74. **<input** class="updateCount" style="width: 80px;"
75. bookId="${entry.value.id}"
76. type="text" value="${entry.value.count}"**>**
77. **</td>**
78. **<td>**${entry.value.price}**</td>**
79. **<td>**${entry.value.totalPrice}**</td>**
80. **<td><a** class="deleteItem" href="cartServlet?action=deleteItem&id=${entry.value.id}"**>**删除**</a></td>**
81. **</tr>**
82. **</c:forEach>**
83. **</c:if>**
84. **</table>**
85. **<**%--如果购物车非空才输出页面的内容--%**>**
86. **<c:if** test="${not empty sessionScope.cart.items}"**>**
87. **<div** class="cart\_info"**>**
88. **<span** class="cart\_span"**>**购物车中共有**<span** class="b\_count"**>**${sessionScope.cart.totalCount}**</span>**件商品**</span>**
89. **<span** class="cart\_span"**>**总金额**<span** class="b\_price"**>**${sessionScope.cart.totalPrice}**</span>**元**</span>**
90. **<span** class="cart\_span"**><a** id="clearCart" href="cartServlet?action=clear"**>**清空购物车**</a></span>**
91. **<span** class="cart\_span"**><a** href="orderServlet?action=createOrder"**>**去结账**</a></span>**
92. **</div>**
93. **</c:if>**
95. **</div>**

98. **<**%--静态包含页脚内容--%**>**
99. **<**%@include file="/pages/common/footer.jsp"%**>**

102. **</body>**
103. **</html>**
104. 注册页面

（1） 效果图



1. 设计理念和方法

注册时会查询数据库中有没有该用户，如果有则会提示该用户已被占用，加入验证码功能

1. 源代码
2. **package** com.LGD.web;
4. **import** com.LGD.pojo.User;
5. **import** com.LGD.service.UserService;
6. **import** com.LGD.service.impl.UserServiceImpl;
7. **import** com.LGD.utils.WebUtils;
8. **import** com.google.gson.Gson;
10. **import** javax.servlet.ServletException;
11. **import** javax.servlet.http.HttpServletRequest;
12. **import** javax.servlet.http.HttpServletResponse;
13. **import** java.io.IOException;
14. **import** java.util.HashMap;
15. **import** java.util.Map;
17. **import** **static** com.google.code.kaptcha.Constants.KAPTCHA\_SESSION\_KEY;
19. **public** **class** UserServlet **extends** BaseServlet {
21. **private** UserService userService = **new** UserServiceImpl();

24. **protected** **void** ajaxExistsUsername(HttpServletRequest req, HttpServletResponse resp) **throws** ServletException, IOException {
25. // 获取请求的参数username
26. String username = req.getParameter("username");
27. // 调用userService.existsUsername();
28. **boolean** existsUsername = userService.existsUsername(username);
29. // 把返回的结果封装成为map对象
30. Map<String,Object> resultMap = **new** HashMap<>();
31. resultMap.put("existsUsername",existsUsername);
33. Gson gson = **new** Gson();
34. String json = gson.toJson(resultMap);
36. resp.getWriter().write(json);
37. }
39. /\*\*
40. \* 注销
41. \* @param req
42. \* @param resp
43. \* @throws ServletException
44. \* @throws IOException
45. \*/
46. **protected** **void** logout(HttpServletRequest req, HttpServletResponse resp) **throws** ServletException, IOException {
47. //        1、销毁Session中用户登录的信息（或者销毁Session）
48. req.getSession().invalidate();
49. //        2、重定向到首页（或登录页面）。
50. resp.sendRedirect(req.getContextPath());
51. }
52. /\*\*
53. \* 处理登录的功能
54. \*
55. \* @param req
56. \* @param resp
57. \* @throws ServletException
58. \* @throws IOException
59. \*/
60. **protected** **void** login(HttpServletRequest req, HttpServletResponse resp) **throws** ServletException, IOException {
62. //  1、获取请求的参数
63. String username = req.getParameter("username");
64. String password = req.getParameter("password");
65. // 调用 userService.login()登录处理业务
66. User loginUser = userService.login(**new** User(**null**, username, password, **null**));
67. // 如果等于null,说明登录 失败!
68. **if** (loginUser == **null**) {
69. // 把错误信息，和回显的表单项信息，保存到Request域中
70. req.setAttribute("msg", "用户或密码错误！");
71. req.setAttribute("username", username);
72. //   跳回登录页面
73. req.getRequestDispatcher("/pages/user/login.jsp").forward(req, resp);
74. } **else** {
75. // 登录 成功
76. // 保存用户登录的信息到Session域中
77. req.getSession().setAttribute("user", loginUser);
78. //跳到成功页面login\_success.html
79. req.getRequestDispatcher("/pages/user/login\_success.jsp").forward(req, resp);
80. }
82. }
84. /\*\*
85. \* 处理注册的功能
86. \*
87. \* @param req
88. \* @param resp
89. \* @throws ServletException
90. \* @throws IOException
91. \*/
92. **protected** **void** regist(HttpServletRequest req, HttpServletResponse resp) **throws** ServletException, IOException {
93. // 获取Session中的验证码
94. String token = (String) req.getSession().getAttribute(KAPTCHA\_SESSION\_KEY);
95. // 删除 Session中的验证码
96. req.getSession().removeAttribute(KAPTCHA\_SESSION\_KEY);
98. //  1、获取请求的参数
99. String username = req.getParameter("username");
100. String password = req.getParameter("password");
101. String email = req.getParameter("email");
102. String code = req.getParameter("code");
104. User user = WebUtils.copyParamToBean(req.getParameterMap(), **new** User());
106. //        2、检查 验证码是否正确  === 写死,要求验证码为:abcde
107. **if** (token!=**null** && token.equalsIgnoreCase(code)) {
108. //        3、检查 用户名是否可用
109. **if** (userService.existsUsername(username)) {
110. System.out.println("用户名[" + username + "]已存在!");
112. // 把回显信息，保存到Request域中
113. req.setAttribute("msg", "用户名已存在！！");
114. req.setAttribute("username", username);
115. req.setAttribute("email", email);
117. //        跳回注册页面
118. req.getRequestDispatcher("/pages/user/regist.jsp").forward(req, resp);
119. } **else** {
120. //      可用
121. //                调用Sservice保存到数据库
122. userService.registUser(**new** User(**null**, username, password, email));
123. //
124. //        跳到注册成功页面 regist\_success.jsp
125. req.getRequestDispatcher("/pages/user/regist\_success.jsp").forward(req, resp);
126. }
127. } **else** {
128. // 把回显信息，保存到Request域中
129. req.setAttribute("msg", "验证码错误！！");
130. req.setAttribute("username", username);
131. req.setAttribute("email", email);
133. System.out.println("验证码[" + code + "]错误");
134. req.getRequestDispatcher("/pages/user/regist.jsp").forward(req, resp);
135. }
137. }

140. }
141. 图书管理

（1） 效果图



1. 设计理念和方法

可以增加删除查看图书信息，也可分页查看图书，前端表单关联bookservlet相关方法

1. 源代码
2. **package** com.LGD.web;
4. **import** com.LGD.pojo.Book;
5. **import** com.LGD.pojo.Page;
6. **import** com.LGD.service.BookService;
7. **import** com.LGD.service.impl.BookServiceImpl;
8. **import** com.LGD.utils.WebUtils;
10. **import** javax.servlet.ServletException;
11. **import** javax.servlet.http.HttpServletRequest;
12. **import** javax.servlet.http.HttpServletResponse;
13. **import** java.io.IOException;
14. **import** java.util.List;
16. **public** **class** BookServlet **extends** BaseServlet{
18. **private** BookService bookService = **new** BookServiceImpl();
20. /\*\*
21. \* 处理分页功能
22. \* @param req
23. \* @param resp
24. \* @throws ServletException
25. \* @throws IOException
26. \*/
27. **protected** **void** page(HttpServletRequest req, HttpServletResponse resp) **throws** ServletException, IOException {
28. //1 获取请求的参数 pageNo 和 pageSize
29. **int** pageNo = WebUtils.parseInt(req.getParameter("pageNo"), 1);
30. **int** pageSize = WebUtils.parseInt(req.getParameter("pageSize"), Page.PAGE\_SIZE);
31. //2 调用BookService.page(pageNo，pageSize)：Page对象
32. Page<Book> page = bookService.page(pageNo,pageSize);
34. page.setUrl("manager/bookServlet?action=page");
36. //3 保存Page对象到Request域中
37. req.setAttribute("page",page);
38. //4 请求转发到pages/manager/book\_manager.jsp页面
39. req.getRequestDispatcher("/pages/manager/book\_manager.jsp").forward(req,resp);
40. }


44. **protected** **void** add(HttpServletRequest req, HttpServletResponse resp) **throws** ServletException, IOException {
45. **int** pageNo = WebUtils.parseInt(req.getParameter("pageNo"), 0);
46. pageNo+=1;
47. //        1、获取请求的参数==封装成为Book对象
48. Book book = WebUtils.copyParamToBean(req.getParameterMap(),**new** Book());
49. //        2、调用BookService.addBook()保存图书
50. bookService.addBook(book);
51. //        3、跳到图书列表页面
52. //                /manager/bookServlet?action=list
53. //        req.getRequestDispatcher("/manager/bookServlet?action=list").forward(req, resp);
55. resp.sendRedirect(req.getContextPath() + "/manager/bookServlet?action=page&pageNo=" + pageNo);
57. }


61. **protected** **void** delete(HttpServletRequest req, HttpServletResponse resp) **throws** ServletException, IOException {
62. //        1、获取请求的参数id，图书编程
63. **int** id = WebUtils.parseInt(req.getParameter("id"), 0);
64. //        2、调用bookService.deleteBookById();删除图书
65. bookService.deleteBookById(id);
66. //        3、重定向回图书列表管理页面
67. //                /book/manager/bookServlet?action=list
68. resp.sendRedirect(req.getContextPath() + "/manager/bookServlet?action=page&pageNo=" + req.getParameter("pageNo"));
69. }


73. **protected** **void** update(HttpServletRequest req, HttpServletResponse resp) **throws** ServletException, IOException {
74. //        1、获取请求的参数==封装成为Book对象
75. Book book = WebUtils.copyParamToBean(req.getParameterMap(),**new** Book());
76. //        2、调用BookService.updateBook( book );修改图书
77. bookService.updateBook(book);
78. //        3、重定向回图书列表管理页面
79. //        地址：/工程名/manager/bookServlet?action=list
80. resp.sendRedirect(req.getContextPath() + "/manager/bookServlet?action=page&pageNo=" + req.getParameter("pageNo"));
81. }

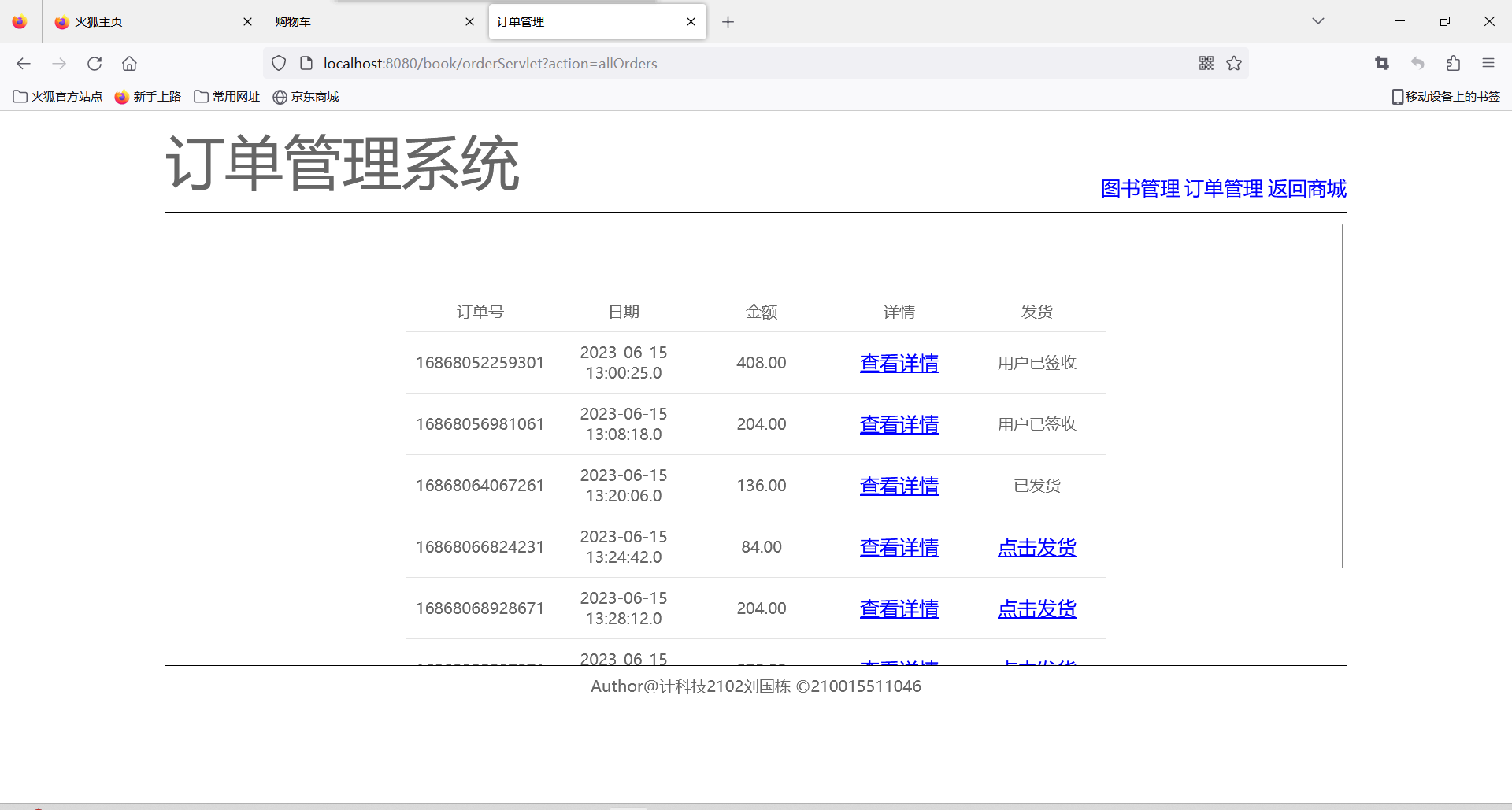

85. **protected** **void** getBook(HttpServletRequest req, HttpServletResponse resp) **throws** ServletException, IOException {
86. //1 获取请求的参数图书编号
87. **int** id = WebUtils.parseInt(req.getParameter("id"), 0);
88. //2 调用bookService.queryBookById查询图书
89. Book book = bookService.queryBookById(id);
90. //3 保存到图书到Request域中
91. req.setAttribute("book", book) ;
92. //4 请求转发到。pages/manager/book\_edit.jsp页面
93. req.getRequestDispatcher("/pages/manager/book\_edit.jsp").forward(req,resp);
94. }

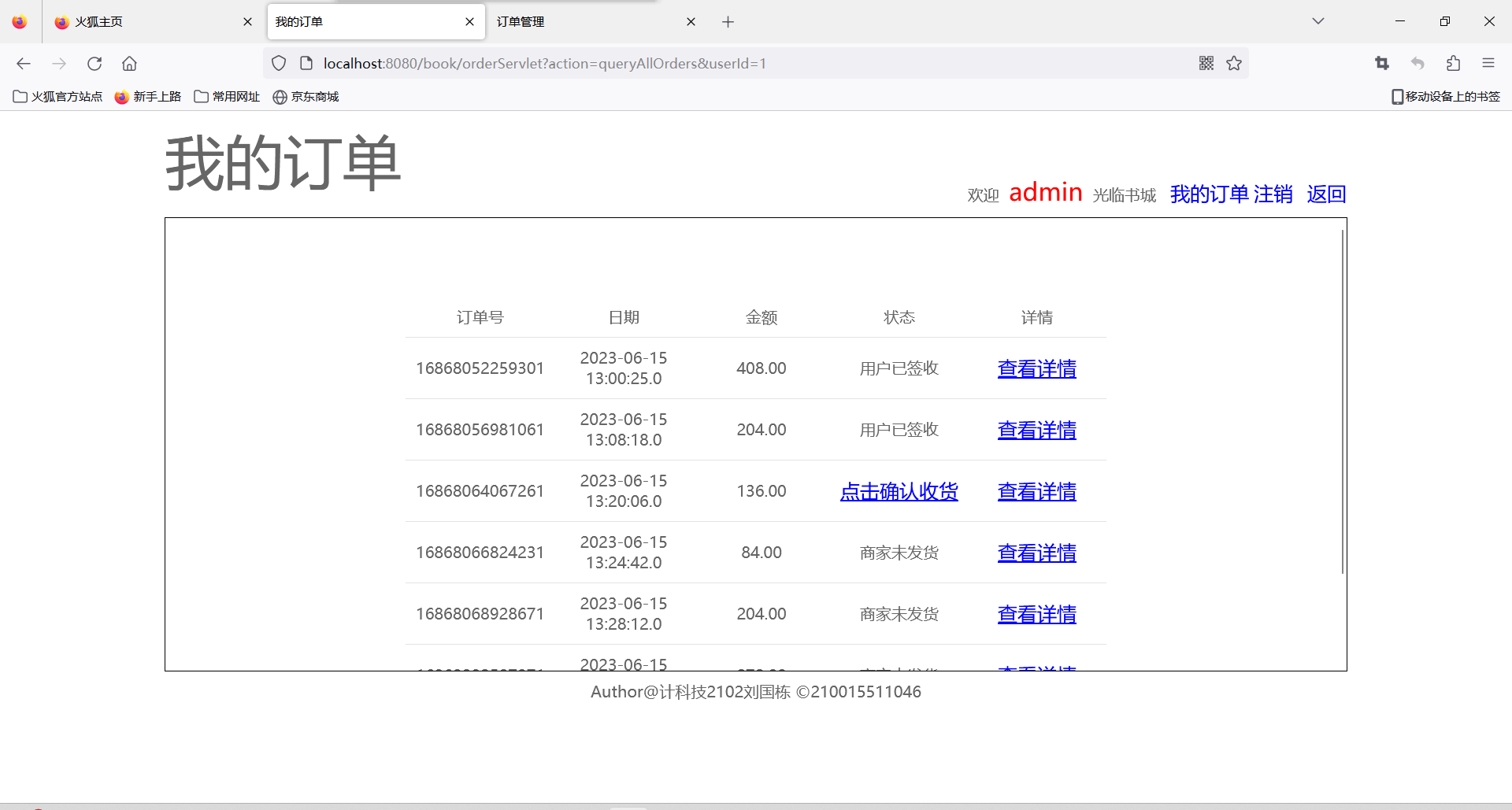

98. **protected** **void** list(HttpServletRequest req, HttpServletResponse resp) **throws** ServletException, IOException {
99. //1 通过BookService查询全部图书
100. List<Book> books = bookService.queryBooks();
101. //2 把全部图书保存到Request域中
102. req.setAttribute("books", books);
103. //3、请求转发到/pages/manager/book\_manager.jsp页面
104. req.getRequestDispatcher("/pages/manager/book\_manager.jsp").forward(req,resp);
105. }



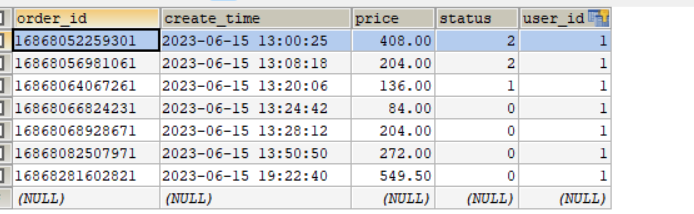
110. }
111. 订单模块

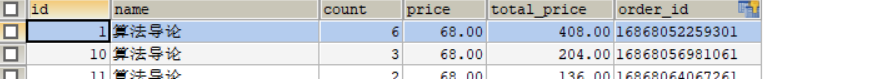
（1） 效果图





1. 设计理念和方法

用户与管理员进行交互，数据库设计：



用户在购物车页面生成订单，管理员接受，并对订单做查看或发货操作，用户可查看订单是否发货，并可选择是否收货

1. 源代码
2. **package** com.LGD.web;
4. **import** com.LGD.pojo.Cart;
5. **import** com.LGD.pojo.Order;
6. **import** com.LGD.pojo.OrderItem;
7. **import** com.LGD.pojo.User;
8. **import** com.LGD.service.OrderService;
9. **import** com.LGD.service.impl.OrderServiceImpl;
11. **import** javax.servlet.ServletException;
12. **import** javax.servlet.http.HttpServletRequest;
13. **import** javax.servlet.http.HttpServletResponse;
14. **import** java.io.IOException;
15. **import** java.util.List;
17. **public** **class** OrderServlet **extends** BaseServlet {
19. **private** OrderService orderService = **new** OrderServiceImpl();
21. /\*\*
22. \* 生成订单
23. \*
24. \* @param req
25. \* @param resp
26. \* @throws ServletException
27. \* @throws IOException
28. \*/
29. **protected** **void** createOrder(HttpServletRequest req, HttpServletResponse resp) **throws** ServletException, IOException {
30. // 先获取Cart购物车对象
31. Cart cart = (Cart) req.getSession().getAttribute("cart");
32. // 获取Userid
33. User loginUser = (User) req.getSession().getAttribute("user");
35. **if** (loginUser == **null**) {
36. req.getRequestDispatcher("/pages/user/login.jsp").forward(req,resp);
37. **return**;
38. }
40. System.out.println("OrderServlet程序在[" +Thread.currentThread().getName() + "]中");
42. Integer userId = loginUser.getId();
43. //        调用orderService.createOrder(Cart,Userid);生成订单
44. String orderId = orderService.createOrder(cart, userId);
45. //        req.setAttribute("orderId", orderId);
46. // 请求转发到/pages/cart/checkout.jsp
47. //        req.getRequestDispatcher("/pages/cart/checkout.jsp").forward(req, resp);
49. req.getSession().setAttribute("orderId",orderId);
51. resp.sendRedirect(req.getContextPath()+"/pages/cart/checkout.jsp");
52. }
53. **protected** **void** queryAllOrders(HttpServletRequest req, HttpServletResponse resp) **throws** ServletException, IOException {
54. User loginUser = (User) req.getSession().getAttribute("user");
55. **if** (loginUser==**null**){
56. req.getRequestDispatcher("/pages/user/login.jsp").forward(req,resp);
57. **return**;
58. }
59. **int** userId = loginUser.getId();
60. List<Order> order = (List<Order>) orderService.queryOrders(userId);
61. req.setAttribute("order",order);
62. req.getRequestDispatcher("/pages/order/order.jsp").forward(req,resp);
63. }
65. **protected** **void** queryOrderItems(HttpServletRequest req, HttpServletResponse resp) **throws** ServletException, IOException {
66. String orderId = req.getParameter("orderId");
67. List<OrderItem> orderItem = (List<OrderItem>) orderService.queryOrderItemsByOrderId(orderId);
68. System.out.println(orderItem);
69. req.setAttribute("orderItem",orderItem);
70. req.getRequestDispatcher("/pages/order/order\_details.jsp").forward(req,resp);
71. }

74. **protected** **void** allOrders(HttpServletRequest req, HttpServletResponse resp) **throws** ServletException, IOException {
75. List<Order> orders = (List<Order>) orderService.queryAll();
76. req.setAttribute("orders",orders);
77. req.getRequestDispatcher("/pages/manager/order\_manager.jsp").forward(req,resp);
78. }

81. **protected** **void** sendOrder(HttpServletRequest req, HttpServletResponse resp) **throws** ServletException, IOException {
82. String orderId = req.getParameter("orderId");
83. req.getSession().setAttribute("orderId",orderId);
84. Order order = orderService.queryOrderByOrderId(orderId);
85. orderService.sendOrder(orderId);
86. req.getRequestDispatcher("orderServlet?action=allOrders").forward(req,resp);
88. }
90. **protected** **void** receiveOrder(HttpServletRequest req, HttpServletResponse resp) **throws** ServletException, IOException {
91. String orderId = req.getParameter("orderId");
92. User loginUser = (User) req.getSession().getAttribute("user");
93. **if** (loginUser==**null**){
94. req.getRequestDispatcher("/pages/user/login.jsp").forward(req,resp);
95. **return**;
96. }
97. **int** userId = loginUser.getId();
98. Order order = orderService.queryOrderByOrderId(orderId);
99. orderService.receiveOrder(orderId);
100. req.getRequestDispatcher("orderServlet?action=queryAllOrders&userId=userId").forward(req,resp);
101. }
102. }
103. 总结

只用到了jsp,servlet并没有用使用大的框架技术，在功能上还可以更加改进，界面上也可以更加完善，用户分类还可以更细分