**实验七** **JSP+JavaBean+Servlet开发**

学生姓名： 张小锋 学 号： 8002119102 专业班级： 计软192班

实验类型：□ 验证 □ 综合 □ 设计 □ 创新 实验日期：2021.11.17 实验成绩：

1. 实验项目名称

JSP+JavaBean+Servlet开发

1. 实验目的

基于MVC模式，使用JSP+JavaBean+Servlet+JDBC技术完成之前实验的功能呢。

1. 实验基本原理
2. JSP技术编写前端页面
3. Servlet技术实现对请求的响应
4. JDBC和JavaBean实现对数据库的操作
5. 主要仪器设备及耗材

计算机、Idea、Tomcat9.0、JDK1.8

五、实验步骤（完整内容见光盘）

按照实验任务书对源代码进行修改，得到如下最终代码，

最终源代码（主要修改部分）：

main.jsp

1. <%@ page import="vo.Category" %>
2. <%@ page import="java.util.List" %>
3. <%@ page import="vo.Book" %>
4. <%@ page contentType="text/html;charset=UTF-8" language="java" %>
5. <html>
6. <head>
7. <link href="css/bootstrap.min.css" rel="stylesheet"/>
8. <link href="css/style.css" rel="stylesheet"/>
9. <title>网上书店</title>
10. </head>
11. <body onload="initAJAX();showCategory()">
12. <div class="modal fade" id="myModal">
13. <div class="modal-dialog">
14. <div class="modal-content">
15. <div class="modal-header"></div>
16. <div class="modal-body"></div>
17. <div class="modal-footer"></div>
18. </div>
19. </div>
20. </div>
21. <div class="header">
22. <div class="container">
23. <div class="row">
24. <div class="login span4">
25. <h1><a href=""> 欢迎来到<strong>我的</strong>书店</a>
26. <span class="red">.</span></h1>
27. </div>
28. <div class="links span8">
29. <a class="login" href="login.html" rel="tooltip" data-placement="bottom" data-toggle="modal" data-target="#myModal"></a>
30. <a class="register" href="" rel="tooltip" data\_placement="bottom"></a>
31. </div>
32. </div>
33. </div>
34. </div> <%--header--%>
35. <div class="row"> <%--下方左右div控制--%>
36. <div class="col-md-3"> <%--左侧菜单div控制--%>
37. <ul class="nav nav-list" id="categorylist">
38. </ul>
39. </div><%--左侧菜单div控制--%>
40. <div class="col-md-9" id="book"><%--右侧书本div控制--%>
41. </div><%--右侧书本div控制--%>
42. </div><%--下方左右div控制--%>
43. <script src="js/jquery.min.js" ></script>
44. <script src="js/bootstrap.min.js"></script>
45. <script src="js/commons.js"></script>
46. <script language="JavaScript">
47. function showCategory() {
48. xmlHttp.open("GET","/category",true);
49. xmlHttp.onreadystatechange = function () {
50. if(xmlHttp.readyState==4) {
51. var data = xmlHttp.responseText;
52. var obj = JSON.parse(data);
53. var listHtml = "<li class=\"nav-header\">书籍类别</li>\n" ;
54. for(var i in obj){
55. listHtml += "<li>\n" +
56. " <a href=\"javascript:showBook("+ obj[i].id + ")\">"+obj[i].name+"</a>\n" +
57. "           </li>"
58. }
59. document.getElementById("categorylist").innerHTML=listHtml;
60. }
61. }
62. xmlHttp.send();
63. }
64. function showBook(categoryID) {
65. xmlHttp.open("GET", "getBookByCategoryID?id="+categoryID, true);
66. xmlHttp.onreadystatechange = function () {
67. if (xmlHttp.readyState == 4) {
68. var data = xmlHttp.responseText;
69. var obj = JSON.parse(data);
70. var listbook = '';
71. for(var i in obj){
72. var bookname = obj[i].name;
73. var desc = obj[i].description;
74. if(desc.length > 20) {
75. desc = desc.substring(0,17)
76. }
77. listbook += `<div class="col-sm-9 col-md-3"><div class="thumbnail" ><img src="images/book.jpg"><div class="caption">  <h4>`
78. + bookname +
79. `</h4><p>`+desc+
80. `</p><p><a href="#" class="btn btn-primary" role="button">加入购物车</a> <a href="#" class="btn btn-default" role="button">查看详情`+
81. `</a></p></div></div> </div>`;
82. }
83. document.getElementById("book").innerHTML = listbook;
84. }
85. }
86. xmlHttp.send();
87. }
88. </script>
89. </body>
90. </html>

BookController

1. package controller;
2. import com.fasterxml.jackson.databind.ObjectMapper;
3. import service.BookService;
4. import vo.Book;
5. import javax.servlet.ServletException;
6. import javax.servlet.annotation.WebServlet;
7. import javax.servlet.http.HttpServlet;
8. import javax.servlet.http.HttpServletRequest;
9. import javax.servlet.http.HttpServletResponse;
10. import java.io.IOException;
11. import java.io.PrintWriter;
12. import java.util.List;
13. @WebServlet("/getBookByCategoryID")
14. public class BookController extends HttpServlet {
15. protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
16. }
17. protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
18. BookService bookService = new BookService();
19. int id = Integer.parseInt(request.getParameter("id"));
20. List<Book> bookList = bookService.getBookByCategoryID(id);
21. *//        request.setAttribute("books",bookList);*
22. *//        request.getRequestDispatcher("/main.jsp").forward(request,response);*
23. response.setCharacterEncoding("UTF-8");
24. ObjectMapper mapper = new ObjectMapper();
25. String jsonStr = mapper.writeValueAsString(bookList);
26. PrintWriter out = response.getWriter();
27. out.write(jsonStr);
28. }
29. }

CategoryController

1. package controller;
2. import com.fasterxml.jackson.databind.ObjectMapper;
3. import service.CategoryService;
4. import vo.Category;
5. import javax.servlet.ServletException;
6. import javax.servlet.annotation.WebServlet;
7. import javax.servlet.http.HttpServlet;
8. import javax.servlet.http.HttpServletRequest;
9. import javax.servlet.http.HttpServletResponse;
10. import java.io.IOException;
11. import java.io.PrintWriter;
12. import java.util.List;
13. @WebServlet("/category")
14. public class CategoryController extends javax.servlet.http.HttpServlet {
15. protected void doPost(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response) throws javax.servlet.ServletException, IOException {
16. }
17. protected void doGet(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response) throws javax.servlet.ServletException, IOException {
18. CategoryService categoryService = new CategoryService();
19. List<Category> list = categoryService.findAll();
20. ObjectMapper mapper = new ObjectMapper();
21. String jsonStr = mapper.writeValueAsString(list);
22. response.setCharacterEncoding("UTF-8");
23. PrintWriter out = response.getWriter();
24. out.write(jsonStr);
25. }
26. }

BookDaoImpl

1. package dao.implPackage;
2. import dao.BookDao;
3. import dao.JDBCUtil;
4. import vo.Book;
5. import java.sql.Connection;
6. import java.sql.PreparedStatement;
7. import java.sql.ResultSet;
8. import java.sql.SQLException;
9. import java.util.ArrayList;
10. import java.util.List;
11. public class BookDaoImpl implements BookDao {
12. @Override
13. public List<Book> getBookByCategoryID(int category\_id) {
14. List<Book> list = new ArrayList<Book>();
15. Connection connection = null;
16. PreparedStatement preparedStatement = null;
17. ResultSet rs = null;
18. try {
19. connection = JDBCUtil.getConnection();
20. String sql = "select id,name,author,price,image,description,category\_id from book where category\_id = ?";
21. preparedStatement = connection.prepareStatement(sql);
22. preparedStatement.setInt(1,category\_id);
23. rs = preparedStatement.executeQuery();
24. while (rs.next()) {
25. Book book = new Book();
26. book.setId(rs.getInt("id"));
27. book.setName(rs.getString("name"));
28. book.setAuthor(rs.getString("author"));
29. book.setPrice(rs.getDouble("price"));
30. book.setImage(rs.getString("image"));
31. book.setDescription(rs.getString("description"));
32. book.setCategory\_id(rs.getString("category\_id"));
33. list.add(book);
34. }
35. } catch (SQLException e) {
36. e.printStackTrace();
37. }
38. finally {
39. JDBCUtil.closeConnection(connection);
40. }
41. return list;
42. }
43. *//    其他方法*
44. }

CategoryDaoImpl

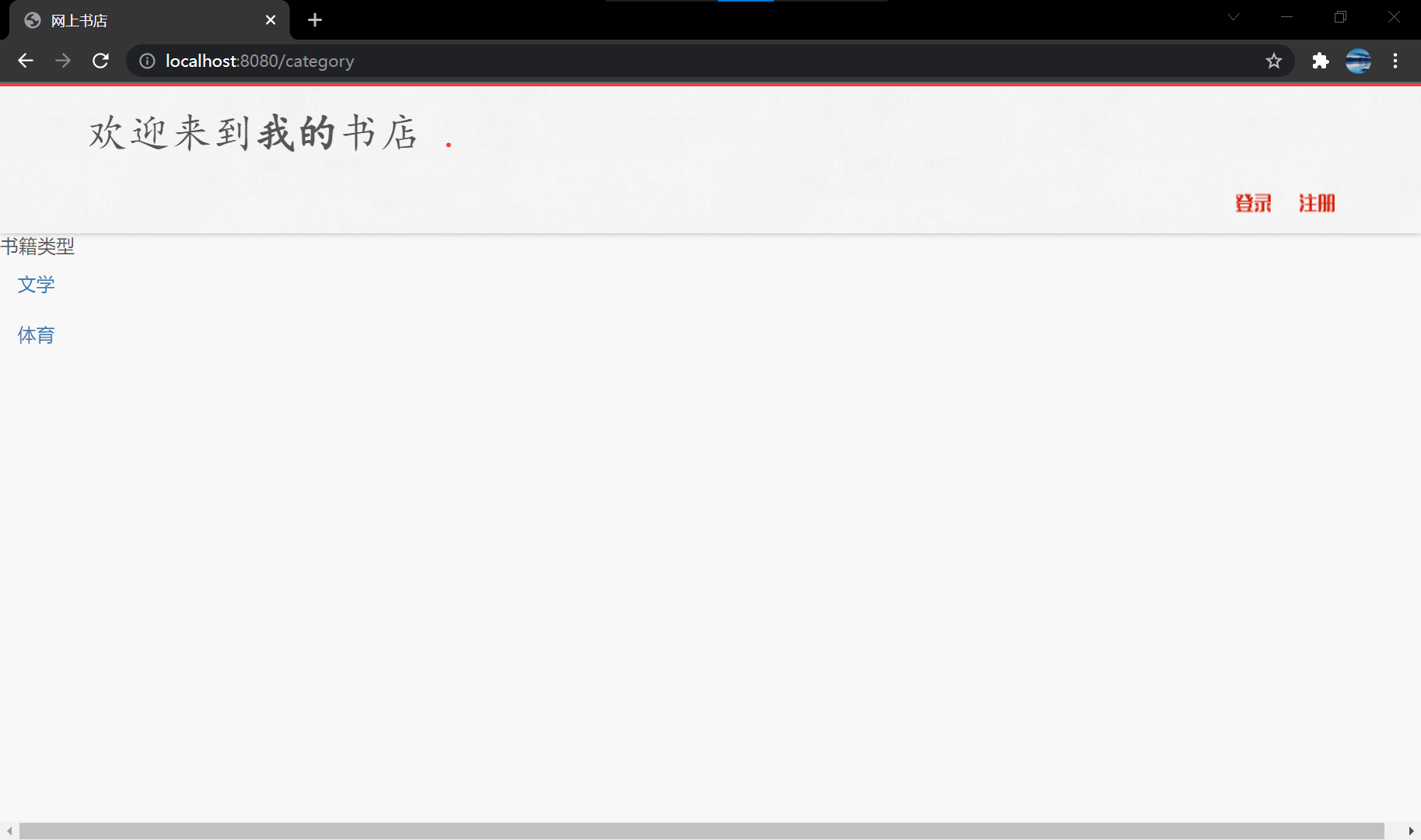
1. package dao.implPackage;
2. import dao.CategoryDao;
3. import dao.JDBCUtil;
4. import vo.Category;
5. import java.sql.Connection;
6. import java.sql.ResultSet;
7. import java.sql.SQLException;
8. import java.sql.Statement;
9. import java.util.ArrayList;
10. import java.util.List;
11. public class CategoryDaoImpl implements CategoryDao {
12. @Override
13. public void add(Category category) {
14. }
15. @Override
16. public void delete(Category category) {
17. }
18. @Override
19. public List<Category> findAll() {
20. Connection connection = null;
21. Statement stat = null;
22. List<Category> list = new ArrayList<Category>();
23. try {
24. connection = JDBCUtil.getConnection();
25. stat = (Statement)connection.createStatement();
26. String sql = "select id,name,description from category";
27. ResultSet resultSet = (ResultSet)stat.executeQuery(sql);
28. while (resultSet.next()){
29. Category category = new Category();
30. category.setId(resultSet.getInt("id"));
31. category.setName(resultSet.getString("name"));
32. category.setDescription(resultSet.getString("description"));
33. list.add(category);
34. }
35. } catch (SQLException e) {
36. e.printStackTrace();
37. }
38. finally {
39. JDBCUtil.closeConnection(connection);
40. }
41. return list;
42. }
43. }

EncodingFilter

1. package filter;
2. import javax.servlet.\*;
3. import javax.servlet.annotation.WebFilter;
4. import java.io.IOException;
5. @WebFilter(filterName = "EncodingFilter",urlPatterns = "/\*")
6. public class EncodingFilter implements Filter {
7. public void destroy() {
8. }
9. public void doFilter(ServletRequest req, ServletResponse resp, FilterChain chain) throws ServletException, IOException {
10. req.setCharacterEncoding("UTF-8");
11. resp.setContentType("text/html; charset=utf-8");
12. chain.doFilter(req, resp);
13. }
14. public void init(FilterConfig config) throws ServletException {
15. }
16. }

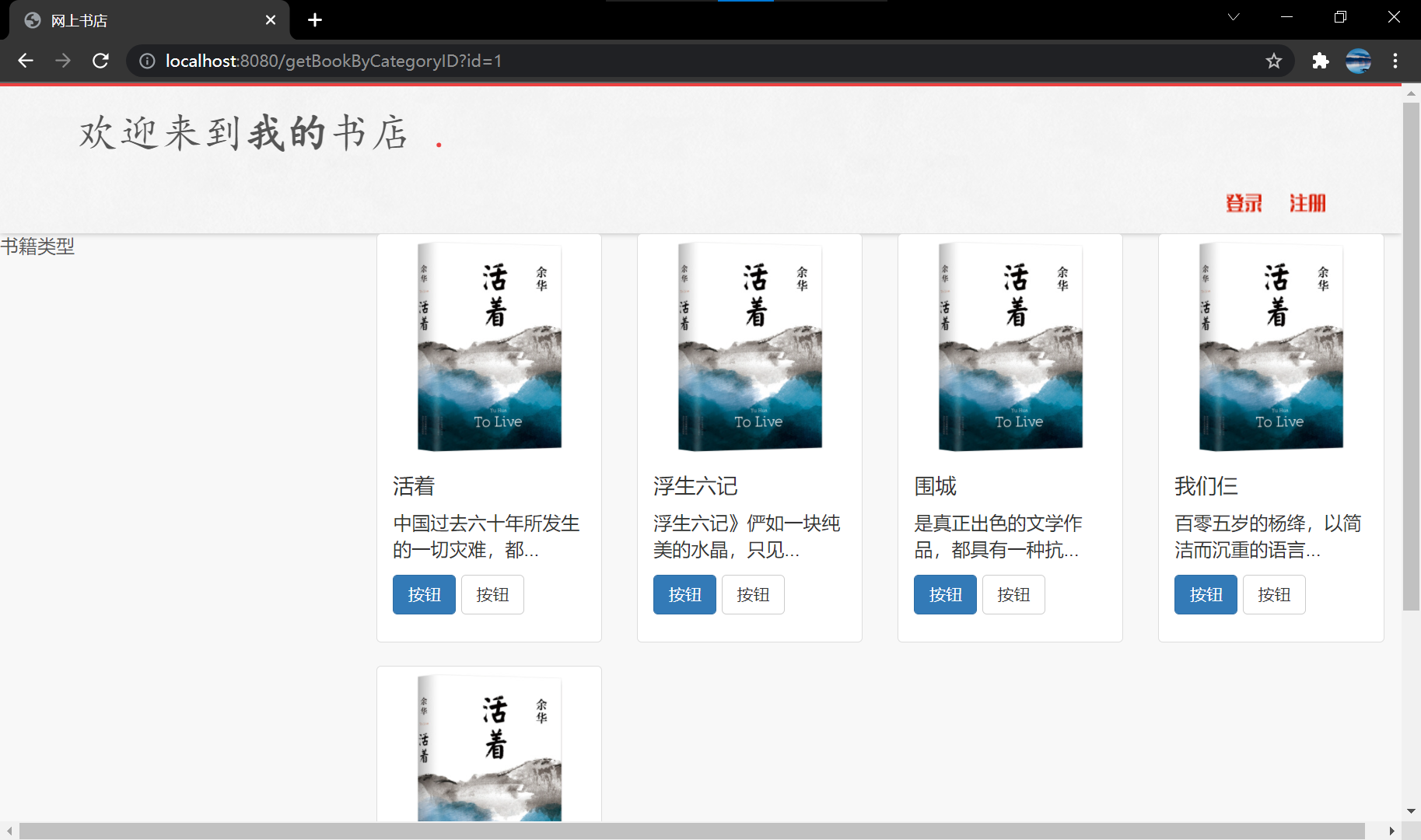
六、实验数据及处理结果

按照流程编写程序，可以得到如下运行结果：



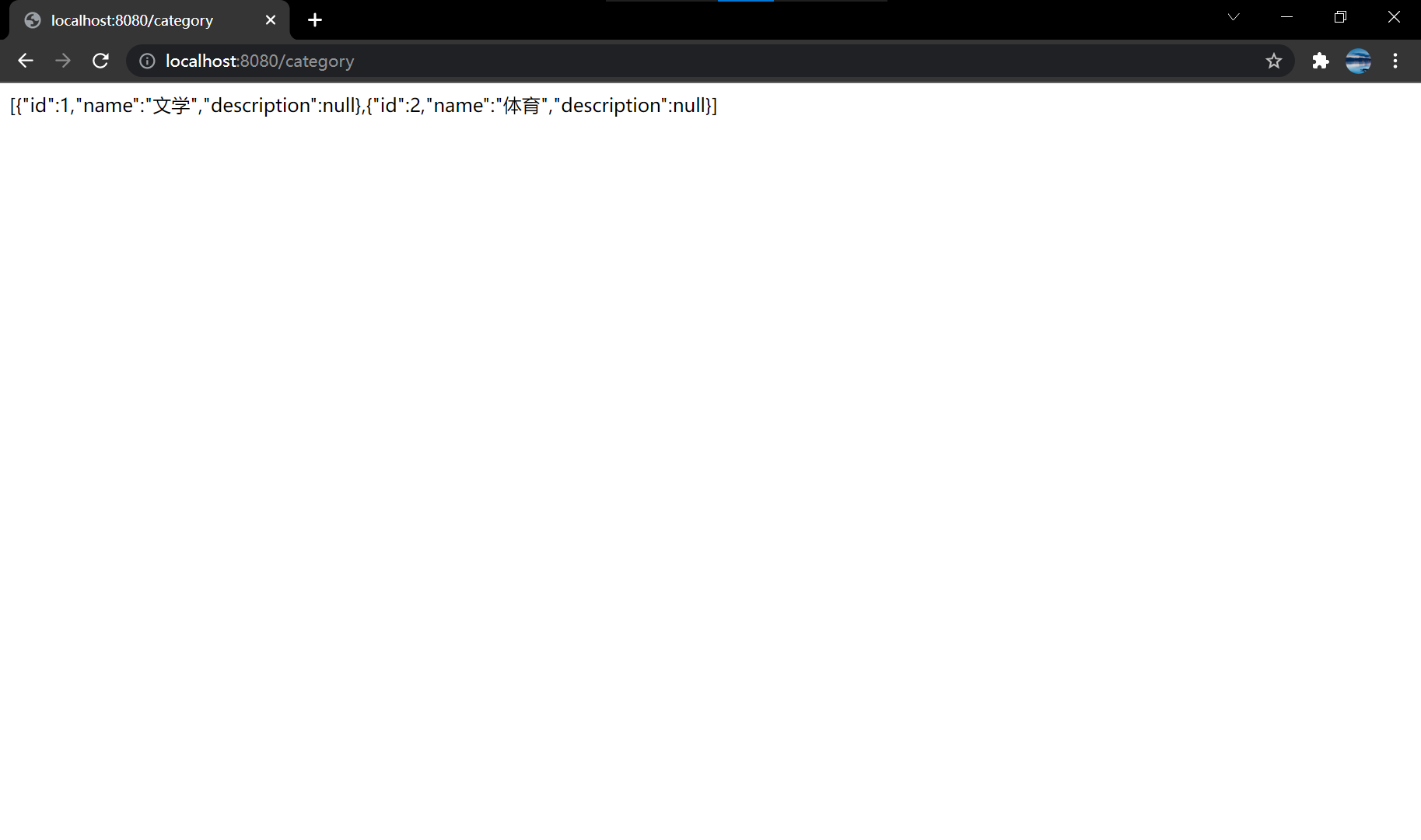
点击左侧类别链接进行测试

发现book显示正常，但是类别信息会消失，如下图所示

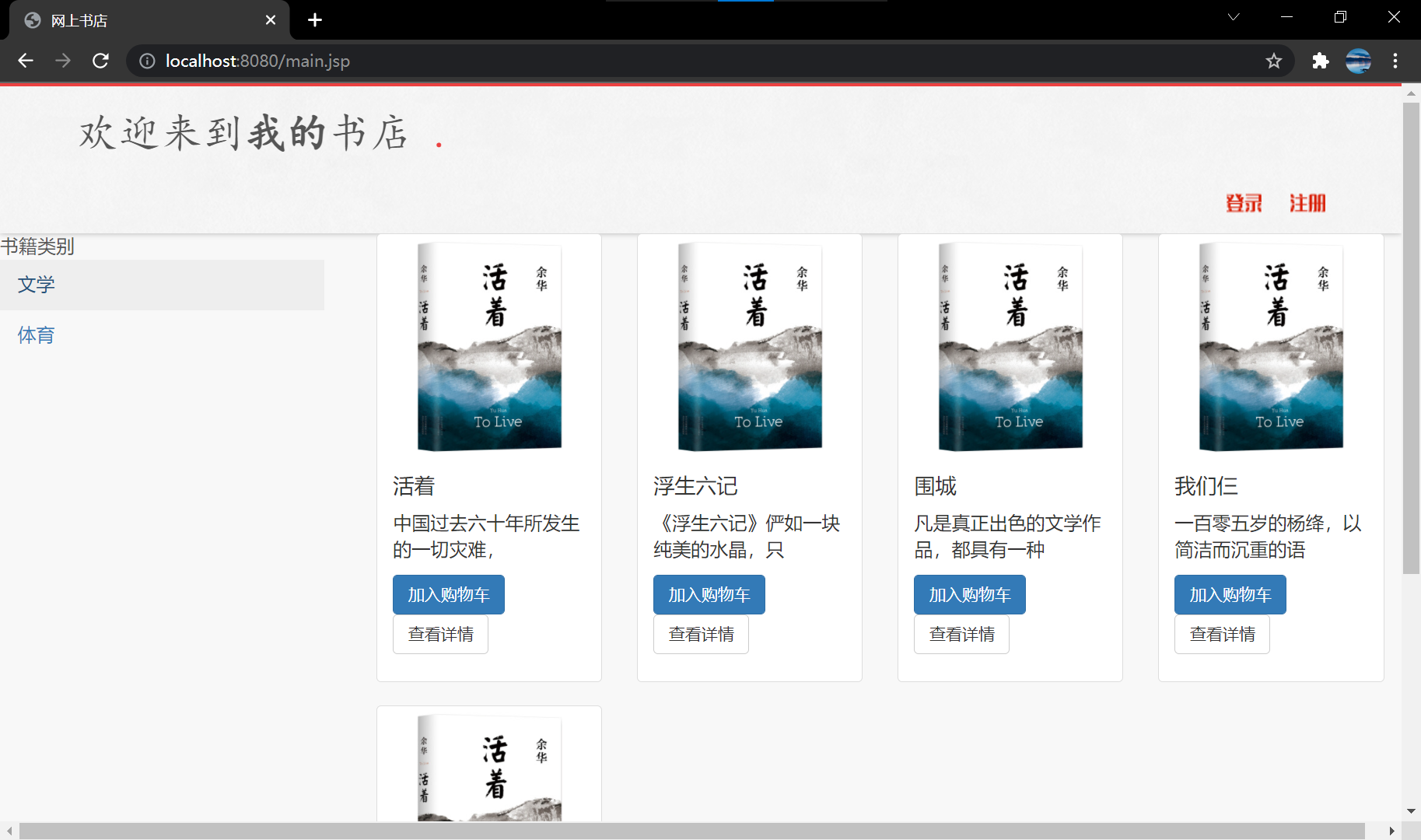


修改代码为返回JSON时的运行结果：





完成所有修改后的最终访问结果如下；



七、思考讨论题或体会或对改进实验的建议

1、JDBCUtil的作用，以及它好处和不足？

作用：将数据库连接和关闭的步骤封装成类

好处：简化了连接数据库的过程，使得在程序中需要连接数据库的时候只需要调用该代码即可，使代码得到了重用。

不足：关闭数据库的时候没有提供关闭Statement 和ResultSet 两个类的对象的功能，在关闭数据库之前应该先关闭这两个类的对象，否则会占用资源。

2、在完成book功能查询和显示时发现类别信息会丢失，如何用最小代码修改量修复此bug？

出现此现象的原因是点击超链接转到后台，后台再转发回来，这个过程之后，request对象中的内容已经不在了，request中的信息只在一次请求中有效。所以，要解决此Bug，只需要将categories信息存在session对象中即可。

修改代码如下所示：

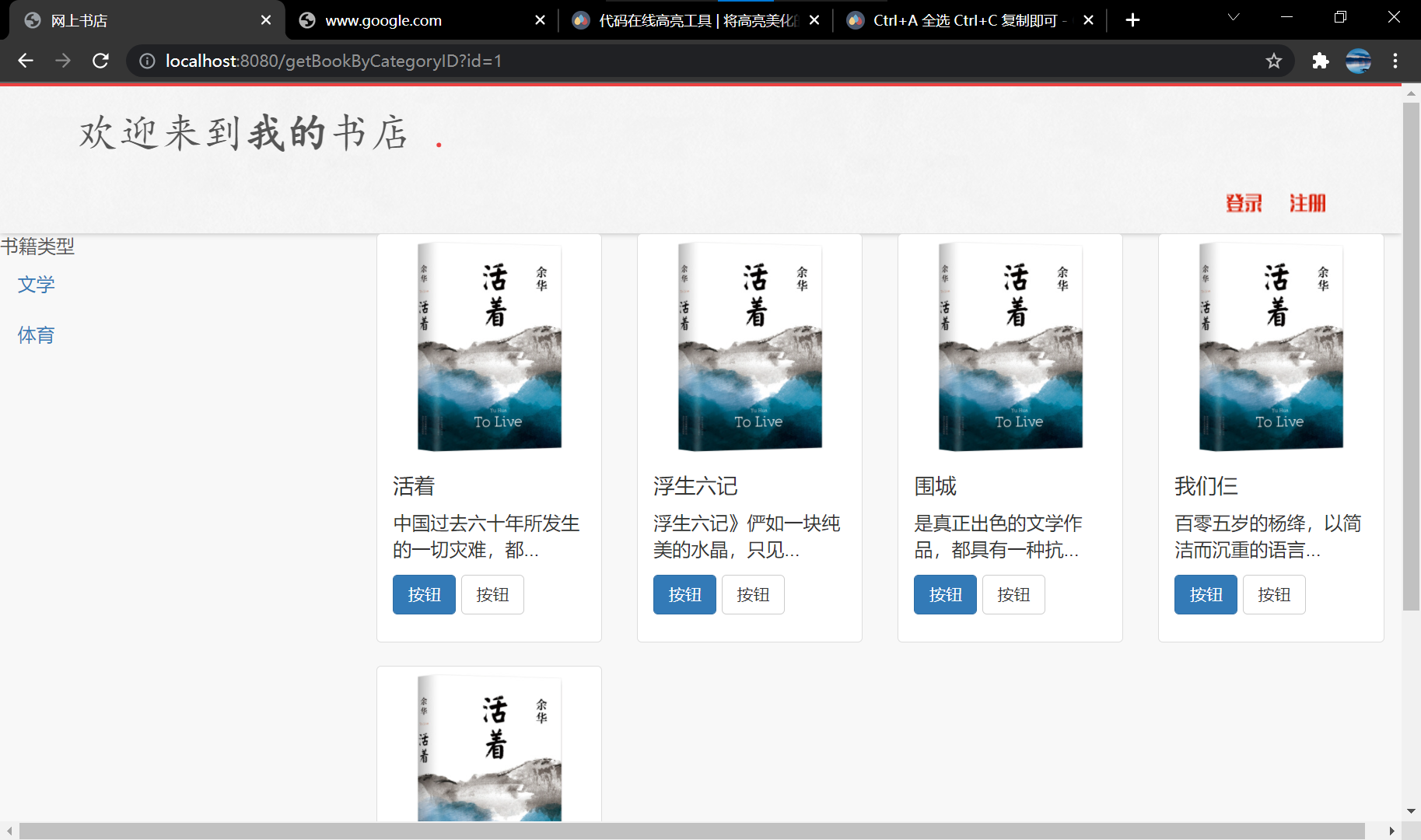
**main.jsp**

 Object categories = request.getSession().getAttribute("categories");

**CategoryController**

request.getSession().setAttribute("categories",list);

修改Bug后的运行结果如下：



3、在controller中转换java对象为json数据格式的代码可否放到service中？为什么？。

不合适，service层应该专注于业务逻辑的处理，getBookByCategoryID函数提供的是返回书籍的通用功能，如果返回的是JSON对象，那么可能会让其他需要调用该函数的地方多一步转化操作。函数的功能应该做到单一，这样才能更有利于代码的复用。

八、参考资料

1.JavaWeb程序设计