Java Web 课程综合训练

软件 43 班 潘戈 2141601048

软件 43 班 许林松 2141601052

日期：2016 年 2 月 29 日

联系电话：15771952903

一、正文内容

1、项目简介：使用 Jsp+Bean+Servlet，设计一个简单的网上购书系统,其中实现了用户注册功能、用户登录功能、信息查询功能、信息浏览功能。由于数据库设置外键出错，最终未实现购物车功能，项目开发过程中使用了 JSP、JavaBean、Servlet、JDBC、MySql、动态表格技术、分页技术，由于时间原因暂时未加入 Filter、Cookies、正则表达式和验证码等功能。

开发工具：myeclipse、MySql、NaviCat、Git

2、实验名称：网上购书系统的简易实现

3、实验内容：

（1）页面布局解释

注册页面：

```
<%
    String error = (String)request.getAttribute("error");
    if(error != null && error.equals("1")){
 %>
 <script>alert("用户名重复，请更换用户名！");</script>
 <%} %>
```

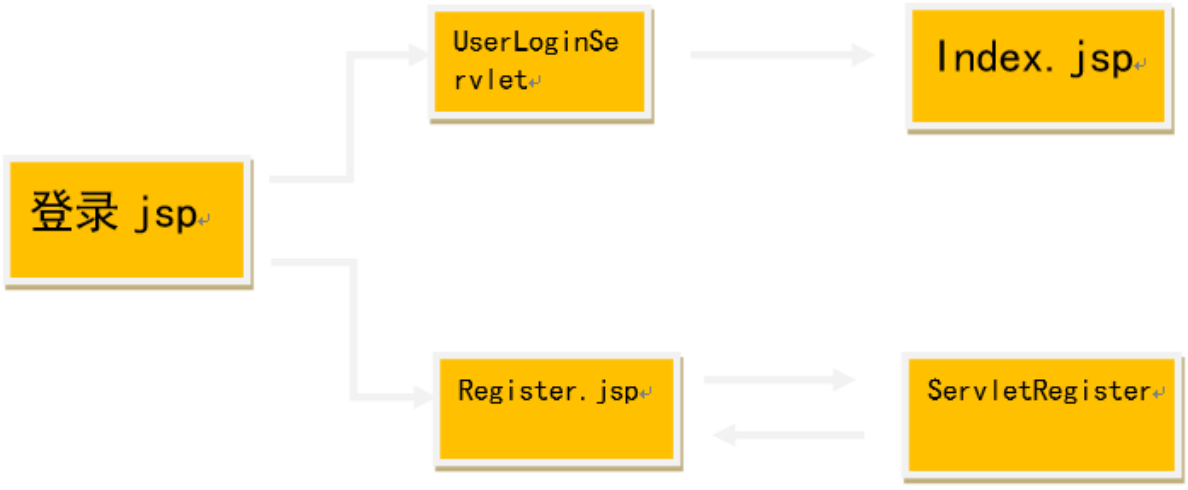
```
<%if(request.getAttribute("ok") != null){ %>
    <tr>
        <td colspan="2" align="center"><font size="3" color="red">恭喜您注册成功</font></td>
    </tr>
<%} %>
```

书籍列表页面：

```
<%
    ArrayList bookSearch = (ArrayList)request.getAttribute("search");
    ArrayList al = null;
    if(bookSearch != null){
        for(int i = 1; i < bookSearch.size(); i++){
            al = (ArrayList)bookSearch.get(i);
 %>
 <tr>
    <td>
        <a href="servlet/BookDetails?bookId=<%=al.get(0)%>" title="单击查看详细信息"><%=al.get(1) %></a>
    </td>
    <td><%=al.get(2) %></td>
    <td><%=al.get(3) %></td>
    <td><%=al.get(5) %></td>
    <td><%=al.get(4) %></td>
    <td></td>
 </tr>
 <%}}%>
```

```
<%
    Integer currentPage = (Integer)request.getAttribute("currentPage");//当前页数
    Integer pageCount = (Integer)request.getAttribute("pageCount");//总页数
    Integer totalCount = (Integer)request.getAttribute("totalCount");//总页数
    if(currentPage == null || pageCount == null){
        currentPage = pageCount = 1;
    }
 %>
<table>
    <tr>
        <td><a href="servlet/SearchServlet?page=1">首页</a></td>
        <td><a href="servlet/SearchServlet?page=<%=currentPage - 1 %>">上一页</a></td>
        <td>第 <%=currentPage %> 页</td>
        <td><a href="servlet/SearchServlet?page=<%=currentPage + 1 %>">下一页</a></td>
        <td><a href="servlet/SearchServlet?page=<%=pageCount%>">最后一页</a></td>
        <td>共<%=totalCount %> 条记录 共 <%=pageCount %> 页</td>
    </tr>
</table>
```

（2）页面关系图



（3）控制层程序和模型层程序代码解释

1、控制层（Controller）：

a. 连接数据库层（Dao）

配置文件：

name	value
jdbc.username	root
jdbc.password	xulinsong
jdbc.driver	com.mysql.jdbc.Driver
jdbc.url	jdbc:mysql://localhost:3306/homework

连接数据库：

```
public class DBConnection {

        private static String driver;
        private static String url;
        private static String user;
        private static String password;

        private Statement st = null;
        private ResultSet rs = null;
        //加载配置文件
        static{
            loadConfig();
        }

        //得到配置文件中的参数
        public static void loadConfig(){

            try {
                InputStream inStream = DBConnection.class.getResourceAsStream("/jdbc.properties");
                Properties prop = new Properties();
                prop.load(inStream);
                user = prop.getProperty("jdbc.username");
                password = prop.getProperty("jdbc.password");
```

```
            driver = prop.getProperty("jdbc.driver");
            url = prop.getProperty("jdbc.url");
        } catch (Exception e) {
            // TODO Auto-generated catch block
            throw new RuntimeException("读取数据库文件异常",e);
        }
    }

    //返回数据库链接对象
    public static Connection getConnection(){
        try {
            //反射
            Class.forName(driver);
            Connection conn = DriverManager.getConnection(url, user, password);
            if(conn == null){
                conn = DriverManager.getConnection(url, user, password);
                return conn;
            }
            return conn;
        } catch (ClassNotFoundException e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
            return null;
          } catch (SQLException e1) {
              // TODO Auto-generated catch block
              e1.printStackTrace();
              return null;
          }

      }
      //关闭
      public static void close(Connection conn, Statement st, ResultSet rs){
          if(rs != null){
              try {
                  rs.close();
              } catch (SQLException e) {
                  // TODO Auto-generated catch block
                  e.printStackTrace();
              }
          }
          if(st != null){
              try {
                  st.close();
              } catch (SQLException e) {
                  // TODO Auto-generated catch block
                  e.printStackTrace();
              }

              }
              if(conn != null){
                  try {
                      conn.close();
                  } catch (SQLException e) {
                      // TODO Auto-generated catch block
                      e.printStackTrace();
                  }
              }
          }
}
```

数据库操作：

```
public class DBOperation {
    //数据库连接对象
    private Connection conn = null;
    private Statement st = null;
    private PreparedStatement pst = null;
    private ResultSet rs = null;

    //数据转换器
    public void DataChange(ArrayList param, PreparedStatement pst) throws SQLException{
        for(int i = 0, j = 1; i < param.size(); i++, j++){
            if(param.get(i).getClass().getName().equals("java.lang.Integer")){
                Integer temp = (Integer)(param.get(i));
                pst.setInt(j, temp.intValue());
            }
            else if(param.get(i).getClass().getName().equals("java.lang.Long")){
                Long temp = (Long)(param.get(i));
                pst.setLong(j, temp.longValue());
            }
            else if(param.get(i).getClass().getName().equals("java.lang.Float")){
                Float temp = (Float)(param.get(i));
                pst.setFloat(j, temp.floatValue());
            }

          else if(param.get(i).getClass().getName().equals("java.lang.String"))
              pst.setString(j, (String)(param.get(i)));
          else
              System.out.println("error");
      }
  }

  //插入删除修改

  public boolean insertDeleteUpdate(String sql, ArrayList param){
      boolean flag = true;
      conn = DBConnection.getConnection();
      if(conn == null){
          System.out.println("数据库没有连接");
          return false;}
      try {
          pst = conn.prepareStatement(sql);
          DataChange(param, pst);
          pst.executeUpdate();
      } catch (SQLException e) {
          // TODO Auto-generated catch block
          e.printStackTrace();
          flag = false;
      } finally{
```

```
            DBConnection.close(conn, st, rs);
        }
        return flag;
    }

    //成批数据的修改操作
    public boolean UpdateByBatch(String sql, ArrayList param){
        boolean flag = false;
        conn = DBConnection.getConnection();
        try {
            pst = conn.prepareStatement(sql);
            conn.setAutoCommit(false);
            //若不出现异常则继续执行到try语句完，否则跳转到catch语句
            for(int i = 0; i < param.size(); i++){
                DataChange((ArrayList)param.get(i), pst);
                pst.addBatch();
            }
            pst.executeBatch();
            //commit:若成功执行完所有插入操作，则正常结束
            conn.commit();
            flag = true;
        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
            //若出现异常，则对数据库中所有已完成的操作全部撤销
            try {
                conn.rollback();
            } catch (SQLException e1) {
                // TODO Auto-generated catch block
                e1.printStackTrace();
            }
        } finally{
            try {
                conn.setAutoCommit(true);
            } catch (SQLException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
                DBConnection.close(conn, pst, rs);
            }
        }
        return flag;
    }

    //返回结果集和结果集中列名的查询，用于普通查询
    public ArrayList queryReturnList(String sql){
        ArrayList al = new ArrayList();
        ResultSetMetaData rsmd = null;
        String colname[];
        int columns;
```

```
    conn = DBConnection.getConnection();
    try {
        st = conn.createStatement();
        rs = st.executeQuery(sql);//得到查询结果一个数据集
        if(rs == null)
            al = null;
        else{
            rsmd = rs.getMetaData();//得到结果集的结构信息，比如字段数、字段名等。
            columns = rsmd.getColumnCount();//得到数据集的列数
            colname = new String[columns];
            for(int i = 1; i <= columns; i++)
                colname[i-1] = rsmd.getColumnName(i);//数据库列的名字

            ArrayList al_colname = new ArrayList();
            for(int i = 1; i <= columns; i++)
                al_colname.add(colname[i-1]);
            al.add(al_colname);//把结果集中各条记录依次放入返回list中的其他行
            //al的第一行是列名的数组
            while(rs.next()){
                ArrayList alRow = new ArrayList();
                for(int i = 1; i <= columns; i++)
                    alRow.add(rs.getString(colname[i-1]));//alRow是类的集合
                al.add(alRow);
            }
        }
    } catch (SQLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
        al = null;
    } finally{
        DBConnection.close(conn, st, rs);
    }
    return al;
}

//返回Boolean的查询操作，用于login等服务
public boolean queryReturnboolean(String sql){
    boolean flag = true;
    conn = DBConnection.getConnection();
    try {
        st = conn.createStatement();
        rs = st.executeQuery(sql);
        if(!rs.next())
            flag = false;
    } catch (SQLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
        flag = false;
    } finally{
```

```
            DBConnection.close(conn, st, rs);
        }
        return flag;
    }

    //计算表的总记录数
    public int getCount(String sql){
        conn = DBConnection.getConnection();
        int num = 0;
        try {
            st = conn.createStatement();
            rs = st.executeQuery(sql);
            rs.next();//rs是结果集。查询出的记录是一个列表，初始时指针指向的是第一条记录之前的。
            num = rs.getInt(1);//得到第一个记录，即列表总数
        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } finally{
            DBConnection.close(conn, st, rs);
        }
        return num;

    }
}
```

b. 登录控制层

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

    //设置字符编码
    request.setCharacterEncoding("utf-8");
    //获得输入信息
    String name = request.getParameter("name");
    String pwd = request.getParameter("pwd");

    DBOperation dbo = new DBOperation();
    //生成sql语句
    String sql1 = "select * from users where name='"+name+"' and pwd='"+pwd+"'";
    //如果登陆成功，则将用户名保存在session中
    if(dbo.queryReturnboolean(sql1)){
        HttpSession session = request.getSession();
        session.setAttribute("login", name);
    }
    else//登陆不成功则把失败信息保存在“loginError”Attribute中
        request.setAttribute("loginError", "error");
    //页面跳转到主页面
    RequestDispatcher rd = request.getRequestDispatcher("/index.jsp");
    rd.forward(request, response);

}
```

c. 注册控制层

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

    //设置字符编码
    request.setCharacterEncoding("utf-8");

    UserBean ub = new UserBean();

    //获得用户输入的注册信息
    String name = request.getParameter("logname");
    String pwd = request.getParameter("password");
    String sex = request.getParameter("sex");
    String age = request.getParameter("age");
    String address = request.getParameter("address");
    String phone = request.getParameter("phone");

    //创建ArrayList，把用户信息保存到ArrayList中
    ArrayList param = new ArrayList();
    param.add(name);
    param.add(pwd);
    param.add(sex);
    param.add(age);
    param.add(address);
    param.add(phone);

    //生成sql语句
    String sql = "insert into users(name,pwd,sex,age,address,phone) values(?,?,?,?,?,?)";

    //设置flag变量查看是否注册成功
    boolean flag = ub.updateUser(sql,param);
    //如果注册成功，将“ok”Attribute设置为1
    if(flag == true){
        request.setAttribute("ok", "1");
    }
    else//否则将“error”Attribute设置为1
        request.setAttribute("error", "1");
    RequestDispatcher dispatcher = null;
    //页面跳转回注册页面
    dispatcher = request.getRequestDispatcher("/register.jsp");
    dispatcher.forward(request, response);
}
```

d. 搜索查询控制层

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

    /**实现查询功能时，使用了分页技术，当数据量很大时，
     * 可以每次只从数据库查询一定数量的信息，
     * 而不是查询所有，提高性能。
     * 使用sql语句中的limit语句来实现分页
     */
    //设置字符编码
    request.setCharacterEncoding("utf-8");
    //获得用户查询使用的关键字
    String name = request.getParameter("name");
    String book1 = request.getParameter("book1");

    int totalCount = 0;//总记录数
    int currentPage = 1;//当前页数

    DBOperation db = new DBOperation();

    //每页显示5条记录，共totalCount条记录
    PageUtil pu = new PageUtil(5, totalCount);

    //得到每页保存的记录数
    int pageSize = pu.getPageSize();


//获得当前页数
if(request.getParameter("page") != null && !request.getParameter("page").equals("0"))
    currentPage = Integer.parseInt(request.getParameter("page"));

//将当前页数set
pu.setCurrentPage(currentPage);

//设置查询数据库时的起点
int fromIndex = (currentPage - 1) * pageSize;

//初始化sql语句
String sql = "select * from books";

//获得记录总条数
String sqlCount = "select count(*) from books";

//生成sql语句
if(name != null && !name.equals("")){//如果用户输入了书名关键字
    sql += " where name like '%"+name+"%' limit " + fromIndex + "," + pageSize;
    sqlCount += " where name like '%"+name+"%'";
    //如果用户也输入了出版社关键字
```

```
        if(book1 != null && !book1.equals("")){
            sql += "and publishing like '%"+book1+"%'";
            sqlCount += "and publishing like '%"+book1+"%'";
        }
    }
    //如果用户没有输入书名关键字
    else if(name == null || name.equals("")){
        //如果用户输入了出版社关键字
        if(book1 != null && !book1.equals("")){
            sql += " where publishing like '%"+book1+"%' limit " + fromIndex + "," + pageSize;
            sqlCount += " where publishing like '%"+book1+"%'";
        }
        else if(book1 == null || book1.equals("")){
            sql += " limit " + fromIndex + "," + pageSize;
        }
    }

    //获得总记录数，并set
    totalCount = db.getCount(sqlCount);
    pu.setRecordCount(totalCount);

       //保存当前页数
       request.setAttribute("currentPage", currentPage);
       //保存总记录数
       request.setAttribute("totalCount", totalCount);
       //保存总页数
       request.setAttribute("pageCount", pu.getPageCount());
       //保存查询结果
       request.setAttribute("search", db.queryReturnList(sql));
       //返回书籍列表界面
       RequestDispatcher rd = request.getRequestDispatcher("/booklist.jsp");
       rd.forward(request, response);
    }
```

e. 浏览信息控制层

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

        //设置字符编码
        request.setCharacterEncoding("utf-8");
        //得到从url中传入的id参数
        int id = Integer.parseInt(request.getParameter("bookId"));

        DBOperation db = new DBOperation();
        //生成sql语句
        String sql = "select * from books where id='"+id+"'";
        //通过id查询书籍
        request.setAttribute("bookinf", db.queryReturnList(sql));
        RequestDispatcher rd = request.getRequestDispatcher("/bookDetail.jsp");
        rd.forward(request, response);
    }
```

2、模型层：

a. 分页处理组件

```
public class PageUtil {
    private int pageSize;//每页显示数
    private int recordCount;//总记录数
    private int currentPage;//当前页数
    private int pageCount;//总页数
    public PageUtil(int pageSize, int recordCount, int currentPage) {
        this.pageSize = pageSize;
        this.recordCount = recordCount;
        this.currentPage = currentPage;
    }

    public PageUtil(int pageSize, int recordCount){
        this(pageSize, recordCount, 1);
    }//当前页数默认为1

    //获得总页数
    public int getPageCount(){
        int size = recordCount / pageSize;//页数=总记录数/每页显示记录数
        int mod = recordCount % pageSize;
        if(mod != 0)//如果有余数，则页数+1
            size++;
        return recordCount == 0 ? 1 : size;//如果没有记录，则默认页数为1
    }

//设置当前页数
public void setCurrentPage(int currentPage) {
    int vp = currentPage <= 0 ? 1 : currentPage;//如果当前页数<=0，设为1，否则为设置值
    vp = vp > getPageCount() ? getPageCount() : vp;//如果vp>总页数，则设置为总页数
    this.currentPage = vp;
}
```

b. 用户模型层

```
public class UserBean {

    private String name;
    private String password;
    private String sex;
    private String address;
    private String age;
    private String phone;
    private Integer id;
```

```java
//获得登入用户信息
public ArrayList getLoginUser(String sql){
    DBOperation op = new DBOperation();
    ArrayList al = op.queryReturnList(sql);
    if(al.size() > 0)
        return (ArrayList) al.get(0);
    return null;
}

//更新添加删除用户信息
public boolean updateUser(String sql, ArrayList param){
    DBOperation op = new DBOperation();
    return op.insertDeleteUpdate(sql,param);
}
```

c. 商品信息层

保存商品 id，name，price

d. 购物车层

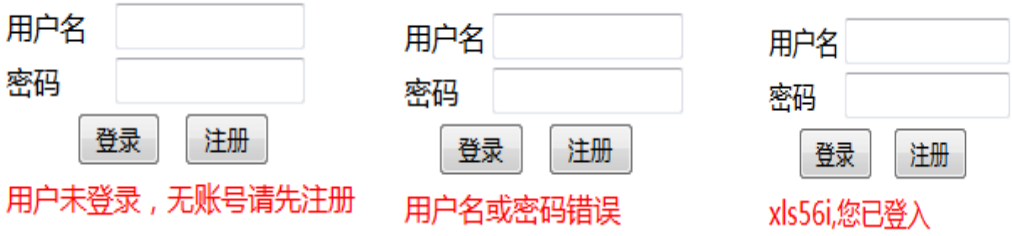
```java
public class Cart {
    private ArrayList<CartItem>cart;
    public Cart(){
        cart = new ArrayList<CartItem>();
    }

    public void addCartItem(CartItem item){
        CartItem oldItem = null;
        if(item != null){
            for(int i = 0; i < cart.size(); i++){
                oldItem = cart.get(i);
                if(oldItem.getId().equals(item.getId())){
                    oldItem.setQuantity(oldItem.getQuantity() + item.getQuantity());
                    return;
                }
            }
            cart.add(item);
        }
    }
```

```
public boolean updateCartItem(String id, int quantity){
    CartItem oldItem = null;
    for(int i = 0; i < cart.size(); i++){
        oldItem = cart.get(i);
        if(oldItem.getId().equals(id)){
            oldItem.setQuantity(quantity);
            return true;
        }
    }
    return false;
}

public boolean removeCartItem(String id){
    CartItem oldItem = null;
    for(int i = 0; i < cart.size(); i++){
        oldItem = cart.get(i);
        if(oldItem.getId().equals(id)){
            cart.remove(i);
            return true;
        }
    }
    return false;
}

public boolean clearCart(){
    cart.clear();
    return true;
}

public double getTotalMoney(){
    Iterator<CartItem> it = cart.iterator();
    double sum = 0.0;
    CartItem item = null;
    while(it.hasNext()){
        item = it.next();
        sum += item.totalPrice();
    }
    return sum;
}

public int getTotalQuantity(){
    Iterator<CartItem> it = cart.iterator();
    int sum = 0;
    CartItem item = null;
    while(it.hasNext()){
        item = it.next();
        sum += item.getQuantity();
    }
```

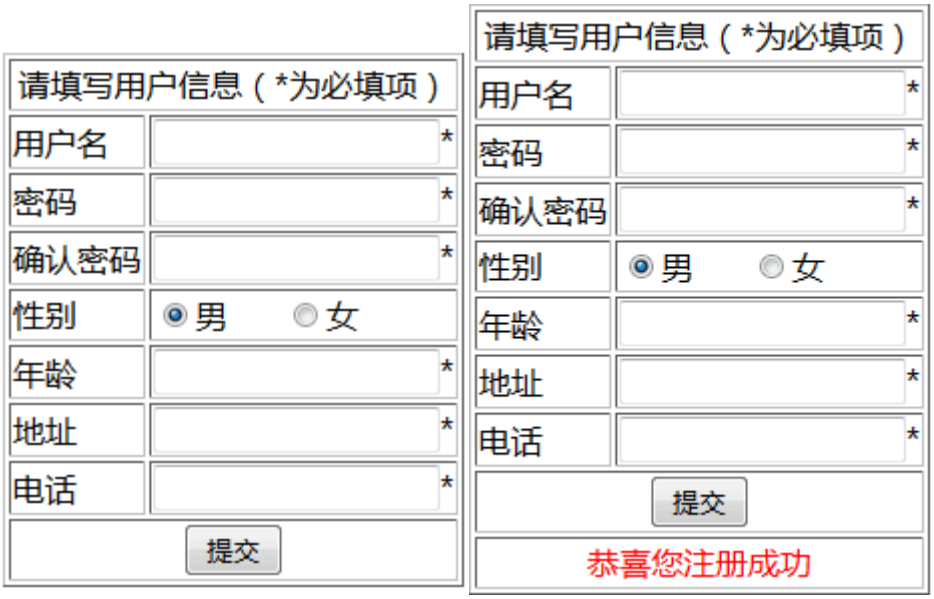
```
        return sum;
    }

    public ArrayList<CartItem> getCart(){
        return this.cart;
    }
}
```

（4）程序运行结果

a. 登录部分



b. 注册部分



c. 搜索查询部分

书名
出版社 清华
搜索

图书列表

书籍	作者	出版社	价格	ISBN	购买
数据结构	严蔚敏，吴伟民	清华大学出版社	29.00	978-7-02-010673-8	
Java程序设计	马素霞	清华大学出版社	39.00	978-7-02-010673-8	
C程序设计	谭浩强	清华大学出版社	32.00	9787506116	
Java Web	马素霞	清华大学出版社	36.00	9434612145	
php	谭浩强	清华大学出版社	31.00	9436421264	

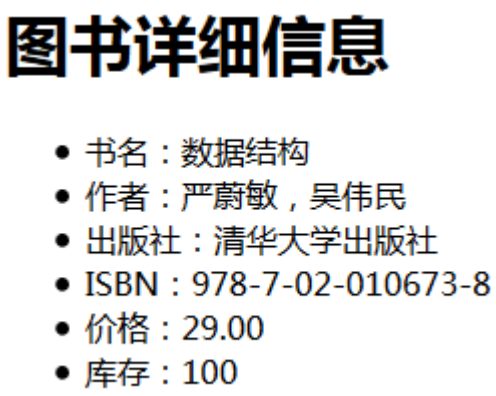
首页 上一页 第 1 页 下一页 最后一页 共7 条记录 共 2 页

图书列表

书籍	作者	出版社	价格	ISBN	购买
Java Web	马素霞	清华大学出版社	36.00	9434612145	
php	谭浩强	清华大学出版社	31.00	9436421264	
Android	谭浩强	清华大学出版社	34.00	9678221563	
数学建模实验	姜启源	清华大学出版社	26.00	93482123354	

首页 上一页 第 2 页 下一页 最后一页 共9 条记录 共 2 页

d.浏览详细信息部分



4、实验总结

通过本次试验，学习了有关 JSP 与数据库的一些知识，对上学期学的 JAVA 有了更深一层的理解，深刻认识了面向对象方法在开发项目时的重要性和简便性，虽然最后有些功能不能实现，但是这也是团队努力的结果。通过这次实验。我们进一步增加了自己的动手能力，虽然在编写代码的过程中有着各种各样的困难，但是最后还是克服了一些，做出了一个简单的系统。今后我们还要进一步的学习计算机技术，让自己的知识越来越丰富，操作也越来越熟练。希望以后可以有更多这样的练习，我们才会不断进步。

5、致谢词

感谢原盛老师在上学期中对我们的帮助，让我们在对计算机技术的学习中又近了一步，虽然技术还很差，跟一些同学相比还有很大的差距，但是我们也在一步一步的进步，感谢原盛老师的教诲，让我们受益匪浅，希望我们今后会学到更多。