第 4 章　JSP 内建对象

一、单元概述

本章主要介绍了 JSP 常用的内建对象。本章结束学生应该完成购物车项目的开发。

JSP 内建对象是指不需要声明，也不需要专门的代码创建其实例，就可以直接在 JSP 页面中使用的对象，所以称为“内置对象”或“内建对象”。JSP 有 9 大内建对象，书中着重介绍了常用的 5 个内建对象，其中最重要的应属 request 对象和 session 对象。

本章通过理论教学、演示教学、练习教学和实践教学法，循序渐进地向学生介绍和演示，并通过练习和实践的方式使学生具备使用 JSP 内建对象处理 HTTP 请求和响应的能力。

二、知识要点及掌握程度

4.1 JSP 内建对象简介：记忆

4.2 out 对象：运用

4.3 request 对象：运用

4.4 response 对象：运用

4.5 session 对象：运用

4.6 application 对象：运用

三、能力要点及重要程度

本章培养能力指标	重要程度
掌握编程基本思想	重要
软件实现过程	重要

四、教学重点与难点

重点：

(1)request 对象实现客户端与服务器的交互；

(2)session 对象实现多页面间数据的共享。

难点：

(1)session 的原理；

(2)session 实现购物车项目。

重点、难点的解决办法：

(1)讲授环节从程序的运行过程和运行原理深入剖析，帮助学生理解代码；

(2)采用多媒体教学，演示程序实现过程；

(3)实践环节是检验教学效果的最好方法。

五、教学设计与实施方法

本单元主要采用讲授教学法、演示教学法、练习教学法和实践教学法。讲授教学是教师课堂讲授理论知识；演示教学是教师课堂演示源代码并演示运行结果；练习教学是学生完成课后作业；实践教学是学生完成课堂实践环节。

六、实践环节设计

项目编号	项目名称	项目类型	项目内容	项目成果
项目 1	JSP 的内建对象 1	单元项目	使用 request 对象和 response 对象处理 HTTP 请求和响应信息	1. 能够处理客户端的请求信息；2. 实现页面的跳转
项目 2	JSP 的内建对象 2	单元项目	使用 session 对象开发简单购物车程序	理解 session 实现多页面数据共享的原理，能够开发购物车程序

【项目导引】

本章学习使用 JSP 内建对象完成相对复杂的动态网页的功能，如表 4-1 所示。本章结束需完成电子商务网上书城系统的第 2 个子项目：购物车模块。

表 4-1　网上商城子项目时间分配表

序号	项目单元名称	教学周															
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	网上商城首页搭建																
2	购物车模块																
3	商品展示功能																
4	商品管理模块																
5	会员管理模块																
6	三级项目 ——网上商城系统																

4.1　JSP 内建对象简介

JSP 内建对象是指不需要声明，也不需要专门的代码创建其实例，就可以直接在 JSP 页面中使用的对象，所以称为“内置对象”或“内建对象”。

这些对象由容器实现和管理，是 Servlet API 接口的实例，在服务器运行时自动生成。通过存取这些内置对象实现与 JSP 页面的 Servlet 环境的相互访问。在 JSP 2.0 的规范中预定义好了 9 个内部对象，分别是 request、response、out、session、application、config、pageContext、page 和 exception，如表 4-2 所示。

表 4-2　JSP 的 9 个内建对象

对象名	类型	作用域	描述
request	javax.servlet.http.HttpServletRequest	request	提供对 HTTP 请求数据的访问，同时还提供用于加入特定请求数据的上下文
response	javax.servlet.http.HttpServletResponse	page	返回用户端的响应
session	javax.servlet.http.HttpSession	session	为请求的客户创建的 session 对象
application	javax.servlet.ServletContext	application	该对象代表应用程序上下文，它允许 JSP 页面与包括在同一应用程序中的任何 Web 组件共享信息

对象名	类型	作用域	描述
out	javax. servlet. jsp. JspWriter	page	该对象提供对输出流的访问
exception	java. lang. Throwable	page	该对象含有只能由指定的 JSP 页面访问的异常数据
pageContext	javax. servlet. jsp. PageContext	page	JSP 页面本身的上下文,它提供了唯一一组方法来管理具有不同作用域的属性,这些 API 在实现 JSP 自定义标签处理程序时非常有用
config	javax. servlet. ServletConfig	page	该对象允许将初始化数据传递给一个 JSP 页面
page	java. lang. Object	page	该对象代表 JSP 页面对象的 Servlet 类实例

4.2 out 对象

out 对象代表 JSP 页面的输出流,用来向客户端输出数据,并且管理应用服务器上的输出缓冲区。

4.2.1 输出信息的方法

1.print()和 println()

这两种方法用于输出数据,其中 print()与 println()方法的区别是:print()方法在输出完毕后并不换行,而 println()方法在输出完毕后会自动换行。当然,println()方法并不会真的在网页上产生换行的效果(字串长度超过浏览器视窗的宽度时会自动换行),只是当你在查看源文件时才会看到换行的效果。如果希望网页上有换行的效果,必须使用 HTML 标签
,即可以直接在 HTML 文件中加入
,也可以使用 print()方法或 println()方法输出
。

【例 4-1】 使用 print 方法输出 print. jsp。

```
<%@ page contentType="text/html; charset=UTF-8"  pageEncoding="UTF-8"%>
<html>
<body>
<%
  for(int i = 0;i<10;i++){
    out.print("i="+i+" ");
}
%>
</body>
</html>
```

运行结果如图 4.1 所示:

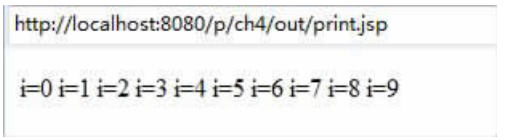

图 4.1　print.jsp 运行结果

使用 println 方法输出 println.jsp。

```
<%@ page contentType="text/html; charset=UTF-8"  pageEncoding="UTF-8"%>
<html>
<body>
<%
  for(int  i = 0;i<10;i++) {
    out.println("i="+i);
  }
%>
</body>
</html>
```

运行结果如图 4.2 所示：

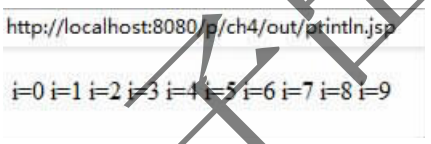

图 4.2　println.jsp 运行结果

观察 print.jsp 和 println.jsp 的运行结果，均不会换行。解决的办法是使用输出语句输出“
”，原因是 JSP 页面最终以 HTML 代码显示在客户端，而客户端浏览器只有见到
标签才会真正的换行。

2.newLine()

该方法用于输出一个换行符号，和 println()方法一样，这种换行效果在页面上是看不出来的，你需要查看源文件才能看到换行的效果。

4.2.2　管理缓冲区的方法

out 对象还可以实现对应用服务器上的输出缓冲区的管理。表 4-3 是 out 对象一些常用的与管理缓冲区有关的函数。

表 4-3　　管理缓冲区的方法

方法	描述
close	关闭输出流，从而可以强制终止当前页面的剩余部分向浏览器输出
flush	输出缓冲区里的数据
clear	清除缓冲区内容 ，但不把数据写到客户端去
clearBuffer	清除缓冲区里的数据，并且把数据写到客户端去

方法	描述
getBufferSize	获得缓冲区的大小
getRemaining	获得缓冲区中未使用空间的大小
isAutoFlush	返回布尔值,true 表示缓冲区自动刷新

【例 4-2】 缓冲区测试 bufferManage.jsp

```
<%@page buffer="1kb" contentType="text/html;charset=gbk" pageEncoding="UTF-8" %>
<html>
<body>
缓存大小:<%=out.getBufferSize() %><br>
<%
   for(int i=0;i<10;i++)
   out.print(i+"{"+out.getRemaining()+"}"+"<br>");
 %>
剩余缓存大小:<%=out.getRemaining() %><br>
自动刷新:<%=out.isAutoFlush() %><br>
</body>
</html>
```

运行结果如图 4.3 所示:

图 4.3 bufferManage.jsp 运行结果

缺省情况下服务端要输出到客户端的内容,不直接写到客户端,而是先写到一个输出缓冲区中。只有在下面三种情况下,才会把该缓冲区的内容输出到客户端上:

(1)该 JSP 网页已完成信息的输出。

(2)输出缓冲区已满。

(3)JSP 中调用了 out.flush()或 response.flushbuffer() 。

4.3 request 对象

request 对象是 javax.servlet.http.HttpServletRequest 的实例。该对象包括了从客户端

传来的请求信息，客户端通过 HTTP 请求提交的信息被 Servlet 容器封装在 request 对象中。获取客户端请求参数必须使用该对象。

4.3.1　访问请求参数

在 Web 应用程序中，经常需要完成用户与网站的交互。例如，用户填写表单，将数据提交给服务器处理，服务器获取到这些信息并进行处理。

1.读取单值参数的值

可以 request 对象的 getParameter()方法获取单值参数的值。单值参数是指变量的值最多只有一个，例如文本框、密码框以及单选按钮等。

语法：String parameterValue= request.getParameter("parameterName");

注意：如果参数不存在则返回 null。

【例 4-3】 获取登录信息。

login.html 代码：

```
<form name="myForm" action="login.jsp" method="POST">
    user: <input  type="text" name="user"><br>
    psw:<input  type="password" name="psw"><br>
    <input type="submit"  name="submit" value="submit" />
</form>
```

login.jsp 代码：

```
<%@ page language="java" pageEncoding="UTF-8" %>
<%
    String user= request.getParameter("user");
    String psw=request.getParameter("psw");
 %>
user:<%=user %><br>
psw:<%=psw %><br>
```

运行 login.html 文件，在表单中填写 user 的值为“qq”，psw 的值为“qq”，如图 4.4 所示。

图 4.4　login.html 运行结果

点击 submit 提交按钮之后，客户端数据由 login.jsp 页面进行处理，运行结果如图 4.5 所示。

图 4.5　login.jsp 运行结果

2.读取多值参数的值

可以 request 对象的 getParameterValues()方法获取多值参数的值。多值参数是指变量的值可能有多个。例如,复选框以及可以多选的下拉列表。

语法:String[] parValue= request.getParameterValues("parName");

【例 4-4】 获取用户爱好。

register.html 代码:

```
<html>
<body>
<form action="register.jsp" method="post">
  user:<input type="text" name="user"><br>
  psw:<input type="password" name="psw"><br>
  性别:
  <input type="radio" name="gender" value="男">男
  <input type="radio" name="gender" value="女">女<br>
  爱好:
  <input type="checkbox" name="hobby" value="游泳">游泳
  <input type="checkbox" name="hobby" value="睡觉">羽毛球
  <input type="checkbox" name="hobby" value="学习">阅读<br>
  <input type="submit" name="ok" value="ok">
</form>
</body>
</html>
```

register.jsp 代码:

```
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
<html>
<body>
<%
    request.setCharacterEncoding("UTF-8");
    String user=request.getParameter("user");
    String psw=request.getParameter("psw");
    String gender=request.getParameter("gender");
%>
用户名:<%=user%><br>
密码:<%=psw%><br>
性别:<%=gender%><br>
爱好:
<%
    String[] h=request.getParameterValues("hobby");
    if(h! =null){
       for(int i=0;i<h.length;i++){
      out.print(h[i]+" ");
        }
```

```
    }else{
        out.print("请选择爱好");
    }
%>
</body>
</html>
```

运行 register.html 文件，在表单中输入注册信息，其中 user 和 psw 的值均为“qq”，如图 4.6所示。

图 4.6　register.html 运行结果

点击“ok”提交按钮，客户端数据由 register.jsp 文件处理，运行结果如图 4.7 所示。

图 4.7　register.jsp 运行结果

3.读取参数的名称

使用 getParameterNames()方法可以获得客户端传送给服务器端的所有的参数名字。这个方法的返回值是一个枚举对象(Enumeration)。

语法：Enumeration names=request.getParameterNames();

【例 4-5】　获取客户端参数名称。

request_param.html 代码：

```
<form action="request_param.jsp" method="post">
    姓名:<input type="text" name="user"><br>
    兴趣:<input type="text" name="love"><br>
    <input type="submit" name="mysubmit" value="subimit"><br>
</form>
```

request_param.jsp 代码：

```
<%@ page contentType="text/html; charset=gbk" pageEncoding="UTF-8" %>
<%@page import="java.util.*" %>
<%
  request.setCharacterEncoding("gbk");
  Enumeration names=request.getParameterNames();
  while(names.hasMoreElements()){
    String n= (String)names.nextElement();
```

```
        String v=request.getParameter(n);
        out.print(n+"="+v+"<br>");
    }
%>
</body>
</html>
```

设有如下一组语 50 运行 request_param.html 文件，在表单中输入如图 4.8 所示数据。

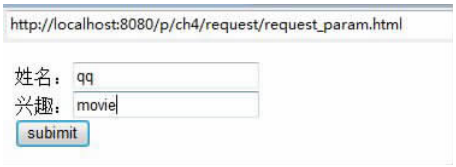

图 4.8 request_param.html 运行结果

运行结果显示三个客户端参数的名字分别是“mysubmit”、“user”和“love”，对应的参数值为“submit”、“qq”和“movie”，如图 4.9 所示。

图 4.9 request_param.jsp 运行结果

4.3.2 读取系统信息

request 对象可以读取客户端和服务器的系统信息，相关方法见表 4-4。

表 4-4 request 对象的常用方法

方法	描述
getProtocol	返回请求的协议和版本号
getRealPath	返回运用替换规则从虚拟地址中获得的实际地址
getRemoteAddr	返回发送请求的接口程序的 IP 地址
getRemoteHost	返回发送请求的接口程序的完全限定的主机名
getScheme	返回发出请求的 url 的机制 ，如 http、https 等。
getServerName	返回接收请求的服务器的主机名
getServerPort	返回接收请求的端口

【例 4-6】 获取系统信息。

request_method.html 代码：

```
<html>
<body>
    <form name="myForm" action="request_method.jsp" method="POST" >
```

```
        <input  type="text" name="search"  />
        <input type="submit"  name="submit" value="submit" />
      </form>
</body>
</html>
```

request_method.jsp 代码：

```
<%@ page contentType="text/html; charset=GBK" %>
<html>
<head>
<title>
</title>
</head>
<body>
<font size="3">
<br>客户使用的协议是<%=request.getProtocol() %>
<br>获取接受客户提交信息的页面<%=request.getServletPath() %>
<br>获取客户提交信息的长度<%=request.getContentLength() %>
<br>客户提交信息的方式<%=request.getMethod() %>
<br>获取客户的 ip 地址<%=request.getRemoteAddr() %>
<br>获取客户机的名称<%=request.getRemoteHost() %>
<br>获取客户端与服务器连接的端口号<%=request.getRemotePort() %>
<br>获取 contentType 的值<%=request.getContentType() %>
<br>获得本机地址<%=request.getLocalAddr() %>
<br>获得本机名称<%=request.getLocalName() %>
<br>获得请求的URI<%=request.getRequestURI() %>
<br>获得服务器的名称<%=request.getServerName() %>
<br>获得服务器的端口号<%=request.getServerPort() %>
</font>
</body>
</html>
```

运行 request_method.html 文件，在表单中任意输入数据，点击提交按钮后，页面的输出结果如下图 4.10 所示。

图 4.10　request_method.jsp 运行结果

4.4 response 对象

response 对象和 request 对象相对应，代表服务器对客户端的响应。该对象是 javax.servlet.http.HttpServletResponse 的实例。通常很少使用该对象直接响应，而是使用 out 对象，除非需要生成非字符响应。

使用 response 对象的 sendRedirect()方法，可以使当前页面重定向到另外的 JSP 程序或者 HTML 文件中。例如，将客户请求转发到 admin.jsp 页面代码如下：

```
<% response.sendRedirect("admin.jsp"); %>
```

使用 response 的 sendRedirect()方法跳转，具有如下特点：

(1)客户端跳转。浏览器先向服务器发送一次请求，遇到 sendRedirect 将目的 url 返回到浏览器，浏览器再去请求目的 url，目的 url 再返回 response 到浏览器。浏览器和服务器两次请求响应。因此，浏览器 url 地址变为目的 url 地址。

(2)当前页面的代码全部被执行完毕之后，才跳转到目标页面。

(3)使用该方法能跳转到任何页面，甚至是外网地址。

【例 4-7】 用户登录页面，如果是管理员，则跳转到 admin.jsp 页面。

login.html

```
<html>
<body>
  <form  action="response.jsp" method="post">
    user:<input type="text" name="user">
    <input type="submit" name="ok" value="ok">
  </form>
</body>
</html>
```

response.jsp

```
<%
    String u=request.getParameter("user");
    if(u.equals("admin")){
     response.sendRedirect("admin.jsp");
    }
%>
```

admin.jsp

```
<html>
<body>
    This is my admin page. <br>
  </body>
</html>
```

运行 login.html 文件，在表单中输入 user 的值为“admin”，如图 4.11 所示。点击“ok”提交

按钮，客户端数据交由 response.jsp 进行处理。

图 4.11　login.html 运行结果

response.jsp 根据客户端输入的不同数据，将页面转发到不同页面。本例中，客户端输入的数据如果为“admin”，则页面跳转到 admin.jsp，如图 4.12 所示。

图 4.12　页面跳转以后的结果

4.5　session 对象

4.5.1　session 概述

session 对象是 javax.servlet.http.HttpSession 的实例化对象。当一个客户访问一个服务器时，可能会在这个服务器的几个页面之间切换，服务器应当通过某种办法知道这是一个客户，就需要 session 对象。

session 何时创建和终止呢？当客户首次访问服务器上的一个 JSP 页面时，JSP 引擎产生一个 session 对象，同时分配一个 String 类型的 ID 号。当客户关闭浏览器离开这个服务器或者 session 的有效期(默认 30 分钟)结束时，服务器端将该客户的 session 对象取消。session 的默认有效期修改方法如下：

```
tomcat 安装路径\conf\web.xml 文件中
    <session-config>
    <session-timeout>30</session-timeout>
    </session-config>
```

在 Web 应用技术中，session 用户一般表现为一个浏览器窗口。每一个客户端都有一个 session 对象用来存放与这个客户端相关的数据，并且各个访问者的 session 对象互不干扰。服务器端每生成一个 session 对象，都会赋予它一个独一无二的编号，这个编号不会重复，服务器就依赖此编号来鉴别不同的客户，getId()方法就返回当前 session 的编号。

【例 4-8】 获取 session id 号。

session_id1.jsp

```
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
<%=session.getId() %>
    <a href="session_id2.jsp"> 下一页 </a>
```

运行结果如图 4.13 和图 4.14 所示。

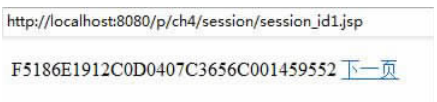

图 4.13　session_id1.jsp 运行结果

session_id2.jsp

```
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
<%=session.getId() %>
```

图 4.14 session_id2.jsp 运行结果

运行该程序,发现两个页面运行结果一致。说明对于一个客户端用户,服务器为其分配唯一的 session id 号。重新打开一个浏览器,发现两个页面的运行结果仍然一致,但是与刚才的运行结果不同。这是因为,一个浏览器窗口代表客户端的一个用户,两次打开浏览器,代表两个用户,服务器为不同的客户端分配不同的 session id 号。

4.5.2 session 的常用方法介绍

1.setAttribute(String name,Java.lang.Object)

该方法用于设定指定名字的属性的值,并且将它添加到 session 对象中,如果这个属性存在,则更改该属性的值。

2.getAttribute(String name)

该方法用于获取指定名字的属性的值,如果该属性不存在,则返回 null。

【例 4-9】 session 实现数据在多页面内共享。

addProduct.jsp

```
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
<html>
<body>
<%
  session.setAttribute("pro_name","喜羊羊");
  session.setAttribute("pro_price","90");
 %>
商品名称:<%=session.getAttribute("pro_name") %><br>
商品价格:<%=session.getAttribute("pro_price") %>
</body>
</html>
```

showProduct.jsp

```
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
<html>
<body>
  商品名称:<%=session.getAttribute("pro_name") %><br>
  商品价格:<%=session.getAttribute("pro_price") %>
</body>
</html>
```

打开 web 浏览器,运行 addProduct.jsp 文件,运行结果如图 4.15 所示。不要关闭浏览器,运行 showProduct.jsp 文件,运行结果如图 4.16 所示。此例说明,在同一个浏览器下,即在同

一个客户端下，存储于 session 中的数据在多页面内共享。

图 4.15　addProduct.jsp 运行结果

图 4.16　showProduct.jsp 运行结果

3.removeAttribute(String name)

该方法用于移除指定名称的 session 属性。

4. setMaxInactiveInterval(int interval)

设置会话的最大持续时间，单位是秒，负数表明会话永不失效。

5. invalidate()

使用该方法使 session 失效，也就是删除已保存在 session 中所有对象。由于会有越来越多的用户访问服务器，因此 session 也会越来越多。为防止内存溢出，服务器会把长时间内没有活跃的 session 从内存删除。这个时间就是 session 的超时时间。如果超过了超时时间没访问服务器，session 就自动失效了。session 的超时时间为 maxInactiveInterval 属性，可以通过对应的 getMaxInactiveInterval()获取，通过 setMaxInactiveInterval(long interval)修改。

【例 4-10】 使用 session 保存用户登录信息。

用户登录相关页面，login.html 和 login.jsp 代码如下。

login.html

```
<form action="login.jsp" method="post">
  user:<input type="text" name="user"><br>
  psw: <input type="password" name="psw"><br>
       <input type="submit" name="ok" value="ok">
</form>
```

login.jsp

```
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
<%
    String u=request.getParameter("user");
    session.setAttribute("userName",u);
    response.sendRedirect("main.jsp");
%>
```

用户登录后，进入主页 main.jsp，代码如下。

main.jsp

```
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
<html>
<body>
<%
   String u=(String)session.getAttribute("userName");
   if(u==null||u.equals("")){
     response.sendRedirect("error.jsp");
   }
```

```
%>
<%=u %>
欢迎您!
<a href="logoff.jsp">注销</a>
<hr>
  后台管理主页<br>
  <a href="news.jsp">新闻管理页面</a> <br>
  <a href=" shop.jsp">商城管理页面</a> <br>
  <a href="book.jsp">书城管理页面</a> <br>
</body>
</html>
```

如果是非法客户登录,由 error.jsp 页面返回错误信息,5 秒后自动返回登录页面,error.jsp 代码如下。

error.jsp

```
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
<html>
<body>
你还没有登录,请登录! <br>
5 秒钟以后返回登录页面<br>
<%
  response.setHeader("refresh","5;url=login.html");
%>
</body>
</html>
```

用户点击超级链接“新闻管理页面”,将进入新闻管理页面 news.jsp,代码如下。

news.jsp

```
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
<%
    String u=(String)session.getAttribute("userName ");
    if(! u.equals("admin")){
    response.sendRedirect("login.html");
  }
%>
<%=u %>
欢迎您!
<a href="logoff.jsp">注销</a>
<hr>
在这里你可以维护新闻,进行增加、修改和删除等操作。
```

用户可以注销本次登录,代码如 logoff.jsp 所示。

logoff.jsp

```
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
<%
```

```
    session.invalidate();
%>
注销成功!
```

运行 login.html，输入用户名和密码均为“admin”，如图 4.17 所示。

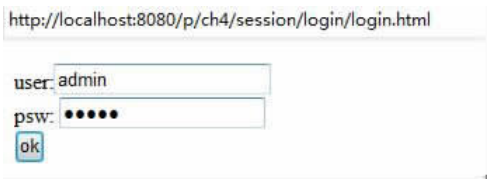

图 4.17　用户登录页面

如果用户没有输入用户名或输入的用户名为空，则页面跳转到 error.jsp 页面，如图 4.18 所示。

图 4.18　error.jsp 运行结果

用户成功登录后，进入后台管理主页面，点击不同栏目的超级链接将进入栏目管理页面，如图 4.19 所示。

图 4.19　用户登录后的主页面

用户可以随时注销本次登录，点击后台管理主页面的超级链接“注销”，就能完成该操作，如图 4.20 所示。

图 4.20　登录注销页面

【例 4-11】 简单的购物车。

shop1.jsp

```
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
<%
   if(session.getAttribute("shop")==null){
     session.setAttribute("shop","toy1");
   }else{
  session.setAttribute("shop",session.getAttribute("shop")+","+"toy1");
   }
%>
```

```
<%=session.getAttribute("shop") %>
<br>
<a href="shop2.jsp">下一页</a>
```

shop2.jsp

```
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
<%
   session.setAttribute("shop",session.getAttribute("shop")+","+"toy2");
%>
  <%=session.getAttribute("shop") %>
   <a href="shop3.jsp">下一页</a>
```

shop3.jsp

```
<%@ page contentType="text/html; charset=UTF-8"  pageEncoding="UTF-8" %>
<%
   session.setAttribute("shop",session.getAttribute("shop")+","+"toy3");
%>
  <%=session.getAttribute("shop") %>
   <a href="shop1.jsp">下一页</a>
```

清空购物车代码：

clear.jsp

```
<%@page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
<%session.removeAttribute("shop"); %>
```

运行 shop1.jsp 文件，结果如图 4.21 所示。

图 4.21 shop1.jsp 运行结果

点击“下一页”，运行 shop2.jsp，结果如图 4.22 所示。

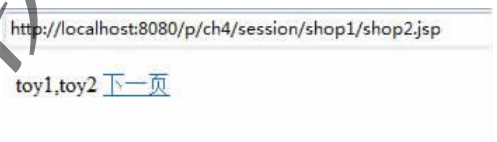

图 4.22 shop2.jsp 运行结果

点击“下一页”，运行 shop3.jsp，结果如图 4.23 所示。

图 4.23 shop3.jsp 运行结果

【例 4-11】 购物车的实现。

商品展示页面 product.html

```
<html>
<body>
```

```
<form  action="shop.jsp" method="post">
  请选择商品<br>
  <input type="radio" name="p" value="苹果">苹果
  <input type="radio" name="p" value="香蕉">香蕉
  <input type="radio" name="p" value="桔子">桔子
  <br>
  <input type="submit" name="ok" value="添加到购物车">
</form>
</body>
</html>
```

购物车处理页面 shop.jsp

```
<%
    request.setCharacterEncoding("UTF-8");
    //本次的购买记录
    String p=request.getParameter("p");
    //取出之前购物车数据

    if(session.getAttribute("shop")==null){
    session.setAttribute("shop",p);
    }else{
    String p1=(String)session.getAttribute("shop");
    //本次+以往购买记录放入购物车
    session.setAttribute("shop",p1+","+p);
    }

    %>
    <%=session.getAttribute("shop") %>
```

【例 4-12】 改进的购物车。

商品展示页面 product.html

```
<html>
<body>
  <form  action="shop1.jsp" method="post">
    请选择商品<br>
    <input type="radio" name="p" value="苹果">苹果
    <input type="radio" name="p" value="香蕉">香蕉
    <input type="radio" name="p" value="桔子">桔子
    <br>
    <input type="submit" name="ok" value="添加到购物车">
  </form>
  </body>
</html>
```

购物车处理页面 shop1.jsp

```
<%@ page language="java" import="java.util.*" pageEncoding="UTF-8"%>
<html>
<body>
<%   request.setCharacterEncoding("UTF-8");
      //本次购买的商品
      String p=request.getParameter("p");
      //数组初始化
      ArrayList al=(ArrayList)session.getAttribute("shop");
      if(al==null){
       al=new ArrayList();
      }
      //向数组中添加数据
      al.add(p);
      //加入到购物车
      session.setAttribute("shop",al);
      response.sendRedirect("show.jsp");
     %>

</body>
</html>
```

显示购物车信息页面 show.jsp

```
<%@ page language="java" import="java.util.*" pageEncoding="UTF-8"%>
<html>
<body>
    <%
    //获取购物车中数据
      ArrayList al=(ArrayList)session.getAttribute("shop");
    //为数组变量 al 创建一个 iterator
     Iterator iterator= al.iterator();
     //读取游标指向的数据
      int i=0;
     while(iterator.hasNext()){
       i++;
      String product=(String)iterator.next();
      //输出商品信息
      out.print(i+"、"+product);

       %>
      <a href="CH4/session/shop1/del.jsp? id=<%=i%>">删除</a><br>
      <%
     }
     %></body></html>
```

删除购物车信息页面

```
<%@ page language="java" import="java.util.*" pageEncoding="UTF-8"%>
<html>
<body>
<%
    String id2=request.getParameter("id");
    int id3=Integer.parseInt(id2);
    //out.print(id3+"<br>");
    ArrayList al=(ArrayList)session.getAttribute("shop");
    al.remove(id3-1);

    response.sendRedirect("show.jsp");
   %>
  </body>
</html>
```

运行商品展示页面 product.html，选择将要购买的商品，如图 4.24 所示。

图 4.24　商品展示页面

点击“添加到购物车”提交按钮，程序由“shop1.jsp”处理后重定向到 show.jsp 页面，运行结果见图 4.25.

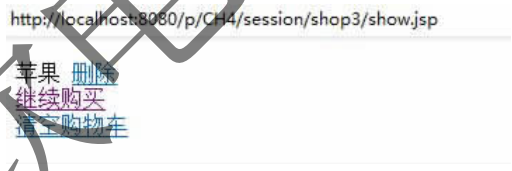

图 4.25　购物车管理页面

4.6　application 对象

application 对象是 javax.servlet.ServletContext 的实例。application 对象最常被用来存取 Web 应用上下文环境的信息。因为环境的信息通常都存储在 ServletContext 中，所以常利用 application 对象来存取 ServletContext 中的信息。服务器一旦启动，就会自动产生一个 application 对象，除非服务器被关闭，否则这个 application 对象将一直保持下去。在 application 对象的生命周期中，在当前服务器上运行的每一个 JSP 程序都可以任意存取和这个 application 对象绑定的参数(或者 Java 对象)的值。application 对象的这些特性为我们在多个 JSP 程序中、多个用户共享某些全局信息提供了方便。由此我们可以不借助数据库就实现聊天室的功能。

application 对象常用的方法有 getAttribute(String name)、setAttribute(String name, Object value)和 removeAttribute(String name)等。

【例 4-13】 application 实现网站计数器。

count.jsp

```
<%@page contentType="text/html;charset=gbk" %>
<% Integer number=(Integer)application.getAttribute("Count");
   if(number==null){
         number=new Integer(1);
         application.setAttribute("Count",number);
     }else{
number=new Integer(number.intValue()+1);
application.setAttribute("Count",number);
     }
     out.print("您是第"+number+"个本站访问者");
%>
```

运行结果如图 4.26 所示：

图 4.26 count.jsp 运行结果

4.7 小结

JSP 2.0 的规范中预定义好了 9 个内部对象，分别是 request、response、out、session、application、config、pageContext、page 和 exception。

其中 request 和 response 分别处理 HTTP 的请求和响应信息；session 对象解决了 HTTP 协议的无链接的缺点，实现了同一个客户端用户的数据在多个页面间共享的问题；application 提供了所有用户共享数据的解决方案。本章学习的重点是 request、response 和 session 对象。

4.8 单元项目——购物车模块

【项目目标】

开发功能完备的购物车模块。使用 session 对象完成购物车关键代码的开发，能够实现添加和维护购物车。

商品展示页面 products.jsp，如图 4.27 所示，请完成相应代码。

图 4.27　商品展示页面效果图

购物车页面 shop.jsp，如图 4.28 所示。

图 4.28　购物车效果图

【关键代码】

shop.jsp 页面关键代码如下：

(1)添加商品部分关键代码：

```
……
//商品编号
int pro_id=Integer.parseInt(request.getParameter("pro_id"));
   //根据商品编号获取商品信息
   Product_do product_do=new Product_do();
   Product product=product_do.getById(pro_id);

   //添加到购物车
   ArrayList al=(ArrayList)session.getAttribute("shop");
```

```
if(al==null){
     al=new ArrayList();
   }
     al.add(product);
     session.setAttribute("shop",al);
……
```

(2)显示购物车的关键代码：

```
……
//显示购物车数据
     ArrayList list=(ArrayList)session.getAttribute("shop");
     Iterator iterator=list.iterator();
     while(iterator.hasNext()){
……
}
……
```