JavaScript

JSP

5 第5章 JSP内置对象

【学习目标】

- 熟练掌握JSP的9种内置对象及其常用方法。
- 熟练使用request内置对象获取用户请求信息。
- 熟练使用session内置对象实现购物车功能。
- 熟练使用application内置对象实现网站计数器功能。
- 了解Cookie的概念和用处，能使用Cookie对象实现自动登录功能。

5.1 内置对象概述

JSP提供了一些由容器实现和管理的内置对象，它们在JSP页面中可以直接使用，不需要实例化。在JSP中一共提供了9种内置对象：out、request、response、session、application、pageContext、config、page、exception。这些内置对象的类型如表5-1所示。

表5-1 JSP内置对象

对象名	类 型	对象说明
out	javax.servlet.jsp.JspWriter	HTML标准输出
request	javax.servlet.http.HttpServletRequest	封装请求信息
response	javax.servlet.http.HttpServletResponse	封装响应信息
session	javax.servlet.http.HttpSession	封装会话信息
application	javax.servlet.ServletContext	封装应用信息
pageContext	javax.servlet.jsp.PageContext	封装当前JSP页面的上下文信息
exception	java.lang.Throwable	封装异常处理信息
config	javax.servlet.ServletConfig	封装JSP页面的Servlet配置信息
page	java.lang.Object	如同Java中的this

5.2 out 对象

out 对象的主要作用是向浏览器输出数据信息,也可以通过 out 对象操作缓冲区。out 对象对应的实现类为 javax.servlet.jsp.JspWriter。

当 JSP 容器将 JSP 页面转换成 Servlet 时,所有的模板元素会使用 out.write()方法来输出。但是,在 JSP 页面的 Java 脚本段中输出模板元素或数据时,一般会使用 out.print()或 out.println()方法。

out 对象的方法主要分为两大类:向浏览器输出数据的方法和操作缓冲区的方法。

5.2.1 向浏览器输出数据的方法

out 对象向浏览器输出数据的方法如下。

(1) public abstract void print():显示各种类型的数据,该方法需要一个参数,参数类型可以是整型、浮点型、字符型、对象类型等。

(2) public abstract void println():分行显示各种类型的数据。该方法可以有一个参数,参数类型可以是整型、浮点型、字符型、对象类型等,方法无参数时,输出一个换行符。

(3) public abstract void newline():输出一个换行符。

注意:

(1) out.print(expression)等价于<%= expression %>。

(2) out.println()或 out.newLine()的换行效果是出现在页面源代码中,而不是页面中。页面的换行仍要使用
标签。

下面通过例子来进一步掌握 out 对象向浏览器输出数据的常用方法。

【例 5-1】 table.jsp,使用 out 对象输出行变色表格。

```
<%@ page contentType="text/html;charset=utf-8" %>
<html>
    <head>
      <title>使用 out 对象输出行变色表格</title>
    </head>
    <body>
    <table width="50%" align="center">
    <%
        int i=0;
        while(i<10){
            i++;
            if(i%2==0) {
                out.println("<tr bgcolor=\"#00FF00\">");
            }else{
                out.println("<tr bgcolor=\"#0055FF\">");
            }
```

```
            out.println("<td>当前行数:"+i+"</td>");
            out.println("</tr>");
        }
    %>
    </table>
    </body>
</html>
```

在浏览器地址栏中输入 URL：http://localhost:8080/ch05/out/table.jsp，得到的运行结果如图 5-1 所示。

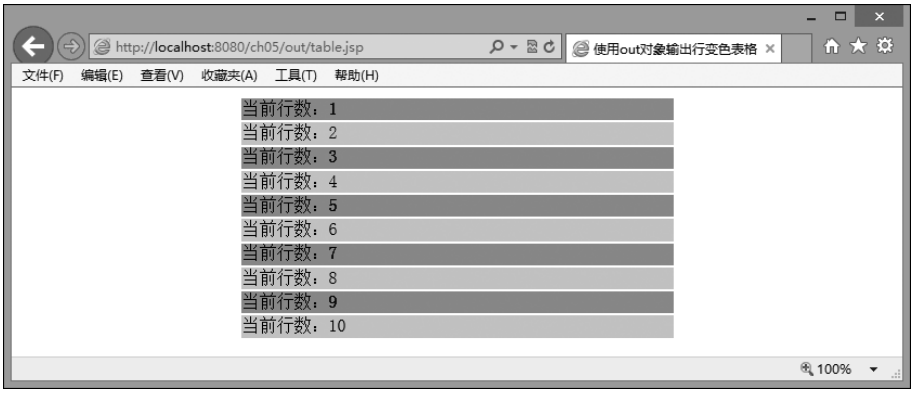

图 5-1　使用 out 对象输出行变色表格

5.2.2　操作缓冲区的方法

out 对象操作缓冲区的方法如下。

(1) public abstract void clear()：清空缓冲区中的内容，如果缓冲区中的数据已经被刷新到客户端，则会引发 IO 异常。

(2) public abstract void clearBuffer()：清空缓冲区中的内容，即使缓冲区中的数据已经被刷新到客户端，也不会引发 IO 异常。

(3) public abstract void flush()：刷新缓冲区中的数据到客户端。

(4) public abstract void close()：刷新缓冲区，并关闭输出流。

(5) public int getBuffersize()：返回缓冲区的大小，单位为字节。

(6) public abstractint getRemaining()：返回缓冲区剩余空间的大小，单位为字节。

(7) public boolean isAutoFlush()：返回 autoFlush 的取值。

下面通过例子来进一步掌握 out 对象操作缓冲区的常用方法。

【例 5-2】　out.jsp，使用 out 对象输出数据并操作缓冲区。

```
<%@ page contentType="text/html;charset=utf-8" buffer="2kb" autoFlush="false"%>
<%
    out.println("你好!");
```

```
    out.clear();
%>
<html>
<head><title>使用 out 对象输出数据并操作缓冲区</title></head>
<body>
<%
    out.println("再次说你好!"+"<br>");
    out.println("out 对象的缓冲区大小为:"+out.getBufferSize()+"字节<br>");
    out.println("缓冲区剩余大小为:"+out.getRemaining()+"字节<br>");
    out.flush();
    for(int i=0;i<2000;i++){
        out.print("A");
    }
    out.println("<br>"+out.isAutoFlush());
%>
</body>
</html>
```

在浏览器地址栏中输入 URL:“http://localhost:8080/ch05/out/out.jsp”,得到的运行结果如图 5-2 所示。

图 5-2 使用 out 对象输出数据并操作缓冲区

5.3 request 对象

request 对象封装了浏览器的请求信息,通过 request 对象的各种方法可以获取客户端以及用户提交的各项请求信息。请求信息包括请求的头部信息、请求方式、请求的参数名称和参数值等。此外,request 对象还提供了获取 cookie 和 session 等对象的方法。request 对象对应的接口为 javax.servlet.http.HttpServletRequest。

5.3.1 获取请求头部信息的方法

request 对象获取请求头部信息的方法如下:

(1) String getHeader(String name)：返回指定头部名称的字符串类型的信息值。

(2) Enumeration<String> getHeaderNames()：返回所有头部信息的名称。

(3) Enumeration<String> getHeaders(String name)：返回指定头部名称的所有信息值。

(4) int getIntHeader(String name)：返回指定头部名称的整型信息值。

(5) long getDateHeader(String name)：返回指定头部名称的代表日期对象的长整型信息值。

下面通过例子来进一步掌握 request 对象获取请求头部信息的常用方法。

【例 5-3】 header.jsp，使用 request 对象获取并显示全部请求头部信息。

```
<%@ page contentType="text/html;charset=utf-8" import="java.util.*"%>
<html>
<head>
<title>使用 request 对象获取并显示全部请求头部信息</title>
</head>
<body>
<h1>客户端发送的 HTTP 请求头包含如下信息:</h1>
<%
    Enumeration<String> headers = request.getHeaderNames();
    while (headers.hasMoreElements()) {
        String headerName = headers.nextElement();
        String headerValue = request.getHeader(headerName);
%>
        <b><%= headerName %></b>:<%= headerValue %><br>
<%
    }
%>
</body>
</html>
```

在浏览器地址栏中输入 URL："http://localhost:8080/ch05/request/header.jsp"，得到的运行结果如图 5-3 所示。

图 5-3　使用 request 对象获取全部请求头部信息

5.3.2 获取请求参数信息的方法

request 对象获取请求参数的方法如下：

(1) String getParameter(String name)：获取请求参数 name 的值，并以字符串形式返回，如果参数 name 不存在则返回 null。

例如，获取名为 name 的参数值：

```
String name=request.getParameter("name");
```

(2) String[] getParameterValues(String name)：获取请求参数 name 的所有参数值，返回字符串数组，如果参数 name 不存在则返回 null。

例如，获取名为 mybox 的复选框的所有取值：

```
String[] mybox=request.getParameterValues("mybox");
if(mybox!=null) {
    for(int i=0;i<mybox.length;i++)
        out.println(mybox[i]+ " ");
}
```

(3) Enumeration<String> getParameterNames()：获取请求中所有参数的名称。

例如：

```
String value=null;
Enumeration names=request.getParameterNames();
while(names.hasMoreElements() ){
    value=names.nextElement();
    out.println(name+"="+request.getParameter(name));
    out.println("<br>");
}
```

(4) Map<String, String[]> getParameterMap()：以 Map 类型返回请求的所有参数，其中参数名作为 Map 的 key。Map 的 value 是这个参数名对应的所有参数值的字符串数组。

5.3.3 其他方法

request 对象的其他常用方法如下：

(1) Object getAttribute(String name)：返回指定名称的属性值，如果指定的属性不存在，则返回 null。

(2) void setAttribute(String name, Object o)：在请求对象中存储名称为 name、值为 o 的属性。

(3) void removeAttribute(String name)：删除指定名称的属性。

(4) String getContextPath()：返回请求 URI 中表示请求上下文的部分。

(5) Cookie[] getCookies()：以数组形式返回客户端发送的所有 Cookie 对象。

(6) String getMethod()：获取 HTTP 请求的方式，例如 GET、POST 或 PUT。

(7) HttpSession getSession()：获取 Session 对象。

(8) void setCharacterEncoding(String encoding)：设置请求体的字符编码方式，用来解决传递非英文字符所出现的乱码问题。

注意：Tomcat 服务器默认按照 ISO-8859-1 进行 URL 编码，因此，对于以 GET 方式提交的请求参数，仅使用 setCharacterEncoding()方法设置请求体的编码方式是无效的。常见的解决方法是在 tomcat 的 server.xml 下的＜connector port＝"8080"…＞标签中增加 useBodyEncodingForURI＝"true"。

5.3.4　项目 1：读取用户的注册信息

1. 项目构思

使用 request 对象的相关方法实现在 reg.jsp 页面获取并显示 3.1.5 节项目 1 中用户在 reg.html 页面填写的注册信息。

2. 项目设计

(1) 注册页面 reg.html 使用的是 3.1.5 节项目 1 中的 reg.html，更改＜form＞标签如下：

```
<form method="post" action="reg.jsp">
```

(2) reg.jsp 页面使用 request 对象的 getParameter()方法和 getParameterValues()方法获取用户注册的全部信息，同时使用 request 对象的 setCharacterEncoding(String encoding)方法来处理表单提交的中文乱码问题。

注意：支持中文的编码方式有 GBK、GB2312、UTF-8 等。选择编码方式时，需要与提交用户信息的表单页的编码方式保持一致。

3. 项目实施

```
文件名:reg.html
<html>
<head>
<meta http-equiv="content-type" content="text/html;charset=utf-8">
<title>用户注册页面</title>
</head>
<body>
<h2 align="center">用户注册</h2>
<form method="post" action="reg.jsp">
<table align="center" border="1" width="50%">
    <tr><th width="30%">姓名</th><td><input type="text" name="name"></td>
</tr>
```

```
    <tr><th>密码</th><td><input type="password" name="pw"></td></tr>
    <tr><th>确认密码</th><td><input type="password" name="pw2"></td></tr>
    <tr><th>性别</th><td><input type="radio" name="gender" value="男" checked>男
          <input type="radio" name="gender" value="女">女</td></tr>
    <tr><th>职业</th>
        <td><select size="1" name="career">
              <option value="教育工作者">教育工作者</option>
              <option value="公司职员" selected>公司职员</option>
              <option value="自由职业者">自由职业者</option>
              <option value="其他">其他</option>
          </select></td></tr>
    <tr><th>电话号码</th><td><input type="text" name="telephone"></td></tr>
    <tr><th>电子邮箱</th><td><input type="text" name="email" size="30">
</td></tr>
    <tr><th>兴趣爱好</th><td>
        <input type="checkbox" name="fav" value="体育">体育
        <input type="checkbox" name="fav" value="看书">看书
        <input type="checkbox" name="fav" value="旅游">旅游
        <input type="checkbox" name="fav" value="美食">美食
    </td></tr>
    <tr><th>自我简介</th>
    <td><textarea name="intro" rows="5" cols="30"></textarea></td></tr>
    <tr><td colspan="2" align="center"><input type="submit" value="注册">
</td></tr>
 </table>
 </form>
 </body>
 </html>

 文件名:reg.jsp
 <%@ page pageEncoding="utf-8"%>
 <%
    request.setCharacterEncoding("utf-8");
 %>
 您的姓名是:<%=request.getParameter("name")%><br>
 您的密码是:<%=request.getParameter("pw")%><br>
 您的职业是:<%=request.getParameter("career")%><br>
 您的电话号码:<%=request.getParameter("telephone")%><br>
 您的电子邮箱:<%=request.getParameter("email")%><br>
 <%
    String[] fav = request.getParameterValues("fav");
    if(fav!=null){
        out.print("您的兴趣爱好有:");
```

```
            for(String f:fav)
            {
                out.print(f);
            }
            out.println("<br>");
        }
    %>
    您的自我简介:<%=request.getParameter("intro")%>
```

4. 项目运行

在浏览器地址栏中输入 URL：“http://localhost:8080/ch05/request/reg.html”，得到的运行结果如图 5-4 所示。

图 5-4　用户注册页面

单击“注册”按钮后，表单信息提交到 reg.jsp 页面，结果如图 5-5 所示。

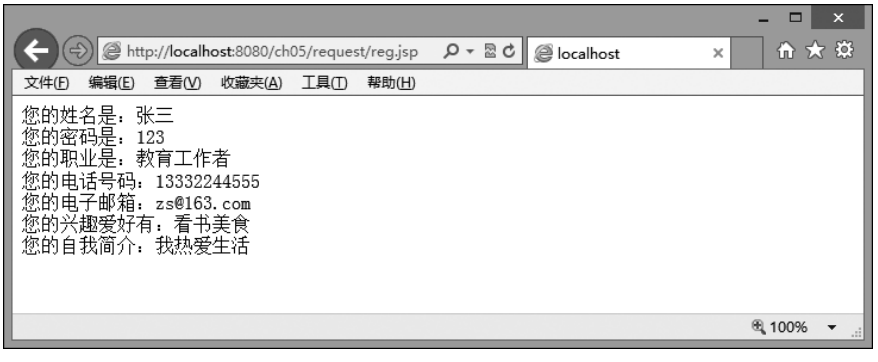

图 5-5　使用 request 对象读取并显示用户注册信息

5.4 response 对象

response 对象与 request 对象相对应,代表服务器对客户端的响应。response 对象对应的接口为 javax.servlet.http.HttpServletResponse。

当客户端访问服务器的 JSP 页面时,会发送一个 HTTP 请求。服务器收到该请求后,会建立一个 response 对象,这个对象封装服务器端处理请求时生成的响应信息,请求处理完成后,服务器将其返回给客户端。response 对象返回给客户端的信息包括响应头信息、响应的本体(HTML 内容)以及服务端的状态码等。

5.4.1 与响应头信息相关的方法

response 对象与响应头信息相关的方法如下:

(1) void addDateHeader(String name, long date):通过指定的名字和值,新增 date 类型的响应头信息。

(2) void addHeader(String name, String value):通过指定的名字和值,新增 String 类型的响应头信息。

(3) void addIntHeader(String name, int value):通过指定的名字和值,新增 int 类型的响应头信息。

(4) void setDateHeader(String name, long date):通过指定的名字和值,设置 date 类型的响应头信息。

(5) void setHeader(String name, String value):通过指定的名字和值,设置 String 类型的响应头信息。

(6) void setIntHeader(String name, int value):通过指定的名字和值,设置 int 类型的响应头信息。

(7) String getHeader(String name):返回指定头部名称的字符串类型的响应头信息,如果 name 指定的响应头信息未被设置,则返回 null。

(8) Collection<String> getHeaderNames():返回所有响应头部名称的集合。

(9) Collection<String> getHeaders(String name):返回指定头部名称的所有响应头信息的集合。

浏览器通常会将刚刚浏览过的网页内容保存在缓存中,当再次请求同一个页面时,浏览器会直接从缓存中取出缓存的内容进行显示。使用 response 对象的 setHeader()方法可以设置浏览器不缓存页面,语句如下:

```
response.setHeader("Cache-Control", "no-cache");
```

除此之外,response 对象的 setHeader()方法还可以设置页面的自动刷新频率。例如,每隔 60s 重新加载本页面的语句如下:

```
response.setIntHeader("Refresh", 60);
```