第 3 章　JSP 内置对象

在 Web 程序设计中，用户交互是必不可少的。比如用户注册、用户登录时需要提交用户信息，程序是怎么处理这些请求的呢？本章介绍非常重要的用于处理浏览器请求的对象，通过使用内置对象，实现系统的访问控制、统计页面或网站的访问量等功能。通过本章的学习，达到如下的目标。

【技能目标】 掌握用户页面控制、在线计数器、用户访问信息跟踪等的编程处理。

【知识目标】 常见内置对象及其应用。

【关键词】 请求(request)　　响应(response)
cookie(含有 key/value 的小型文本文件)　　重定向(redirect)
会话(session)　　应用程序(application)
属性(attribute)　　时间间隔(interval)

3.1　JSP 内置对象概述

JSP 内置对象是指在 JSP 中内置的不需要定义就可以在网页中直接使用的对象。它是 Web 服务器创建的一组对象。JSP 内置对象的名称是 JSP 的保留字。

因为内置对象有些能够存储参数，有些能够提供输出，还有些能提供其他的功能，因此使用这些内置对象的频率比较高。内置对象有以下特点。

(1) 内置对象是自动载入的，因此它不需要直接实例化。

(2) 内置对象是通过 Web 容器来实现和管理的。

(3) 在所有的 JSP 页面中，直接调用内置对象都是合法的。

JSP 标准中定义了 9 个内置对象。

(1) out 对象：负责管理对客户端的输出。

(2) request 对象：负责得到客户端的请求信息。

(3) response 对象：负责向客户端发出响应。

(4) session 对象：负责保存同一客户端一次会话过程中的一些信息。

(5) application 对象：表示整个应用的环境的信息。

(6) exception 对象：表示页面上发生的异常，可以通过它获得页面异常信息。

(7) page 对象：表示当前 JSP 页面本身，就像 Java 类定义中的 this 一样。

(8) pageContext 对象：表示此 JSP 的上下文。

(9) config 对象：表示此 JSP 的 ServletConfig。

3.2 out 对 象

JSP 通过 out 对象向客户端输出各种数据类型的内容，对应用服务器上的输出缓冲区进行管理。

3.2.1 向客户端输出数据

out 对象是一个输出流，用于向客户端输出数据，out 对象基类是 javax.servlet.jsp.JspWriter 类。out 对象跟 Servlet 中由 HttpServletResponse 类得到的 PrintWriter 对象略有不同，但是 JspWriter 类和 PrintWriter 类都是从 java.io.Writer 类继承而来的，所以基本上还是一样的。out 对象的常用方法如表 3-1 所示。

表 3-1 out 对象的常用方法

方 法 名	描 述
void print()	输出数据，不换行
void println()	输出数据，换行
void newLine()	输出一个换行符
void flush()	输出缓冲区里的内容
void 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
void clear()	清除缓冲区里的数据，但不把数据写入客户端
void clearBuffer()	清除缓冲区里的数据，并且把数据写入客户端
int getBufferSize()	获得缓冲区的大小
int getRemaining()	获取缓冲区中没有被占用的空间的大小

1. print()方法

print()方法向客户端浏览器输出信息，通过该方法输出与使用 JSP 表达式输出相同。例如，下面两种方式其效果是一样的。

```
<%
  out.print("新闻动态");
 %>
<% ="新闻动态" %>
```

2. println()方法

println()方法向客户端浏览器输出信息，并在输出内容后输出一个换行符。

说明：out 内置对象的 print()和 println()方法与 Java API 中的 PrintStream(System.out 获取)提供的 print()和 println()方法作用相似。

例如，通过 println()方法向页面中输出字符串“明日科技”及“编程词典”的代码如下。

```
<%
  out.println("明日科技");
  out.println("编程词典");
 %>
```

虽然 println()方法输出了换行"\n",但在 HTML 语言中输出换行需要使用< br >标签,并且不会解析"\n"换行符。所以虽然 println()方法输出了内容,但是并未换行,如果换行要使用< pre >标签。

```
<pre>
<%
  out.println("明日科技");
  out.println("编程词典");
 %>
</pre>
```

3.2.2　管理缓冲区

out 对象不仅可以向 JSP 页面输出内容,而且可以管理页面中的缓冲区,如清理缓冲区、刷新缓冲区以及获取缓冲区大小等。

crear():用于关闭输出流,清除缓冲区中的内容,一旦输出流被关闭了,就不能再使用 out 对象进行任何操作。

crearBuffer():清除当前缓冲区中的内容。

flush():刷新流。

isAutoFlush():检测当前缓冲区已满时是自动清空还是抛出异常。

getBufferSize():获取缓冲区的大小。

一般来说,不要在 JSP 页面中直接调用 out 对象的 close()方法,否则将会抛出异常。

【例 3-1】 关闭输出流。

TestOut01.jsp 的代码如下。

```
<%@ page contentType = "text/html; charset = gb2312" %>
<html>
  <body>
    <%
      out.println("清华大学< br >");
      out.println("北京大学< br >");
      out.close(); //关闭输出流
     %>
  </body>
</html>
```

3.2.3　任务:输出用户信息

1. 需求分析

使用 out 对象按表格格式输出显示用户信息。

2. 任务实施

(1) 使用 MyEclips 创建一个新项目 task3_1。

(2) 新建一个 JSP 页面 userList.jsp,编写如下代码。

```
<%@ page language="java" import="java.util.*" pageEncoding="GBK"%>
<html>
  <head><title>用户信息</title></head>
  <body>
    <%!
      class User
      {
          private String code,String name;
          public User(String code,String name) {
            this.code=code; this.name=name;
          }
          public void setcode(String code) {
            this.code=code;
          }
          public String getcode() {
              return this.code;
          }
        public String toString() {
            return "code:"+this.code+" name:"+name;
        }
    }
    %>
    <%
      User[] s=new User[3];
      s[0]=new student("910001","张上方"); s[1]=new student("910002","历史");
      s[2]=new student("910003","王五");
      out.println("<table border="1">")
      out.println("<tr><th>学号</th><th>姓名</th></tr>")
      int i;
      for (i=0;i<3;i++) {
        out.println("<tr><td>"+s[i].code+"<\td>");
        out.println("<td>"+s[i].name+"<\td><\tr>");
      }
      out.println("</table>");
    %>
  </body>
</html>
```

(3) 启动服务器,部署项目并在浏览器上显示。

3.3 request 对象

request 对象是 HttpServletRequestWrapper 类的实例,代表着客户端的请求。request 包含客户端的信息以及请求的信息,如请求哪个文件、附带的地址栏参数等。

它的继承层次结构图如图 3-1 所示。ServletRequest 接口的唯一子接口是 HttpServletRequest，HttpServletRequest 接口的唯一实现类是 HttpServletRequestWrapper。

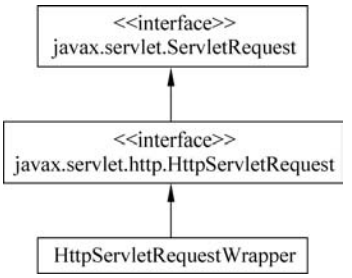

图 3-1　request 对象继承层次结构图

request 内置对象是由 Tomcat 创建的，一旦 HTTP 请求报文发送到 Tomcat 中，Tomcat 对数据进行解析，就会立即创建 request 对象，并对参数赋值，然后将其传递给对应的 JSP/Servlet。一旦请求结束，request 对象就会立即被销毁。

request 对象用来封装 HTTP 请求的参数信息、进行属性值的传递以及完成服务端跳转，其主要的方法如表 3-2 所示。

表 3-2　Request 对象的主要方法

方　法	描　述
void setAttribute(String name,Object value)	在 request 中保存一个对象。本页面内或者 forward 之后的页面中可以通过 getAttribute(String name)方法获取该对象
Object getAttribute(String name)	从 request 中获取 name 对应的对象
String getParameter(String key)	返回提交的参数
String getMethod()	返回提交方式，一般为 GET 或者 POST
String[] getParameterValues(String key)	返回提交的多个同名参数值。以数组形式返回
Enumeration getParameterNames()	返回所有提交的参数名称
Cookie[] getCookies()	返回所有的 Cookie
String getContextPath()	返回应用程序路径，例如/jstweb
String getRequestURL()"/jsp/method.jsp"	返回请求的 URI 路径，例如/jsp/method.jsp
void setCharacterEncoding(String encoding)	设置 request 的编码方式
String getHeader(String name)	获取 request 头信息
Enumeration getHeaderNames()	返回所有的 request 头名称
Dispatcher getRequestDispatcher()	返回 Dispatcher 对象。Dispatcher 对象可以执行 forward
HttpSession getSession()	返回 HttpSession 对象

3.3.1　获取客户端请求参数

request 对象封装了用户提交的信息，通过调用该对象相应的方法可以获取封装的信息，即使用该对象可以获取用户提交信息。request 对象可以用于获取客户端请求，如图 3-2 所示。

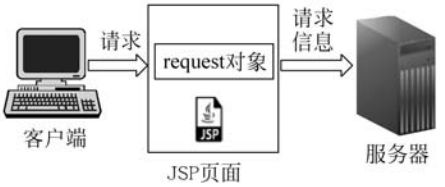

图 3-2　request 对象处理客户请求

request 对象的内存模型可以简单地划分为参数区和属性区。参数区存放的是 Tomcat 解析 HTTP 请求报文后提取出来的参数名和参数值。参数内的变量指向对象外部的相应的字符串对象，如果是多值参数，对应的就是字符串数组对象。

常见的客户端传递参数方式有以下几种。

(1) 浏览器地址栏直接输入：一定是 GET 请求。

(2) 超链接：一定是 GET 请求。

(3) 表单：可以是 GET 请求，也可以是 POST 请求，这取决于< form >的 method 属性值。

GET 请求和 POST 请求有以下区别。

(1) GET 请求。

① 请求参数会在浏览器的地址栏中显示，所以不安全。

② 请求参数长度限制长度在 1KB 之内。

③ GET 请求没有请求体，无法通过 request. setCharacterEncoding()来设置参数的编码。

(2) POST 请求。

请求参数不会显示在浏览器的地址栏中，相对安全。

请求参数长度没有限制。

下面介绍 request 的几种常见应用场合。

1. 单值参数的获取

对于单值参数，使用 String getParameter(String name)方法得到对应的字符串对象的地址引用，如果客户端请求没有输入 value 值，得到的字符串对象就是 null。

【例 3-2】　通过超链接传递参数。

```
< a href = "login. jsp?name = 张三 &sex = man&id = " >传递参数</a >
```

login. jsp 的代码如下。

```
< % = "name:" + request. getParameter("name") % >< br >
< % = "sex:" + request. getParameter("sex") % >< br >
< % = "id:" + request. getParameter("id") % >< br >
```

如果指定的参数不存在，将返回 null；如果指定了参数名，但未指定参数值，将返回空的字符串。

【例 3-3】　获取表单输入的文本信息。

```
<%@ page contentType = "text/Html;charset = GB2312" %>
<html>
<body>
  <form action = "tree.jsp" method = "post" name = "myform">
    <input type = "text" name = "boy">
    <input type = "submit" value = "提交" name = "submit">
  </form>
</body>
</html>
```

tree.jsp 的代码如下。

```
<%@ page contentType = "text/html;charset = GBK" %>
<html>
<body>
  获取文本框提交的信息:
  <% String textContent = request.getParameter("boy"); %>
  <% = textContent %>
</body>
</html>
```

注意：使用 request 对象获取信息要格外小心，要避免使用空对象，否则会出现 NullPointerException 异常。

【例 3-4】　获取单选按钮的信息。

radio.jsp 的代码如下。

```
<html>
<%@ page contentType = "text/html;charset = GBK" %>
<body>
  <P>诗人李白是中国历史上哪个朝代的人:
    <form action = "answer.jsp" method = "post" name = "form">
      <input type = "radio" name = "dynasty" value = "a">宋朝
      <input type = "radio" name = "dynasty" value = "b">唐朝
      <input type = "radio" name = "dynasty" value = "c">明朝
      <input type = "radio" name = "dynasty" value = "d" checked = "ok">元朝
    </form>
  </body>
</html>
```

answer.jsp 的代码如下。

```
<html>
<%@ page contentType = "text/html;charset = GB2312" %>
<body>
  <% String dynasty = request.getParameter("dynasty");
    if(s1 == null) s1 = "";
    if(s1.equals("b")) {
   %>
```

```
    你回答正确
    <% } %>
</body>
</html>
```

【例 3-5】 获取列表框的信息。

select.jsp 的代码如下。

```
<html>
<%@ page contentType="text/html;charset=GB2312" %>
<body>
  请选择部门
  <form action="dep.jsp" method="post" name="form">
    <p>选择部门</p>
    <select name="dep">
      <option Selected value="1">销售部
      <option value="2">开发部
      <option value="3">测试部
    </Select>
    <input type="submit" value="提交你的选择" name="submit">
  </form>
</body>
</html>
```

dep.jsp 的代码如下。

```
<html>
<%@ page contentType="text/html;charset=GBK" %>
<body>
<% String s1=request.getParameter("dep"); %>
  你选择的是:<% =dep %>
  </body>
</html>
```

2. 多值参数的获取

对于多值参数,通过 String[] getParameterValues(String name)方法可以得到这个多值参数的数组对象的地址引用。如果客户请求没有输入值,那么 request 获取的是一个 null 的数组对象。

【例 3-6】 获取复选框 Check 的信息。

```
<html>
<%@ page contentType="text/html;charset=GB2312" %>
<body>
  <form action="inte.jsp" method="post" name="form">
    <p>选择爱好
      <input type="checkBox" name="inte" value="1" />书法
      <input type="checkBox" value="2" name="inte"/>体育
      <input type="checkBox" value="3" name="inte"/>唱歌
```

```
      <input type="submit" value="提交你的选择" name="submit">
    </p>
  </form>
</body>
</html>
```

inte.jsp 的代码如下。

```
<%@ page language="java" import="java.util.*" pageEncoding="GBK"%>
<html>
  <head><title>复选框的应用</title></head>
    <body>
      <%
          request.setCharacterEncoding("GBK");
          String[] inter=request.getParameterValues("inte");
        爱好:
      <%
        for(int i=0;i<inter.length;i++){
       %>
        <%=inter[i]%>
      <%  }  %>
    </body>
</html>
```

3. 获取参数名称

通过 Enumeration getParameterNames()方法获取所有参数的名称。

【例 3-7】 获取表单中所有参数名称。

test.jsp 的代码如下。

```
<form action="param.jsp" method="post">
  参数 1:<input type="text" name="p1"/><br/>
  参数 2:<input type="text" name="p2"/><br/>
  <input type="submit" value="提交"/>
</form>
```

param.jsp 的代码如下。

```
<%
  Enumeration names = request.getParameterNames();
  while(names.hasMoreElements()){
    out.println(names.nextElement());
  }
%>
```

4. 获取参数名和参数值

还可以通过 Map getParameterMap()方法得到所有参数名和参数值组成的 Map 对象,其中 key 为参数名,value 为参数值,因为一个参数名称可能有多个值,所以参数值是

String[],而不是 String。

【例 3-8】 获取参数名和参数值。

test.jsp 的代码如下。

```
<a href = "param.jsp?p1 = v1&p1 = vv1&p2 = v2&p2 = vv2">超链接</a>
```

param.jsp 的代码如下。

```
<%
    Map<String,String[]> paramMap = request.getParameterMap();
    for(String name : paramMap.keySet()) {
        String[] values = paramMap.get(name);
        System.out.println(name + ": " + Arrays.toString(values));
    }
%>
```

5. 处理汉字信息

如果 request 对象获取客户提交的汉字字符时出现了乱码问题,就必须进行特殊处理。首先,将获取的字符串用 ISO-8859-1 进行编码,并将编码存放到一个字节数组中,然后将这个数组转化为字符串对象。例如:

```
String textContent = request.getParameter("boy");
byte b[] = textContent.getBytes("ISO - 8859 - 1");
textContent = new String(b);
```

【例 3-9】 处理乱码。

test.jsp 的代码如下。

```
<%@ page contentType = "text/html;charset = GB2312" %>
<html>
<body>
  获取文本框提交的信息:
    <% String textContent = request.getParameter("boy");
      byte b[] = textContent.getBytes("ISO - 8859 - 1");
      textContent = new String(b);
    %>
  </body>
</html>
```

也可以用下面的方法处理乱码问题。

```
<% ----以 POST 方式提交数据时----
  //设置读取请求信息的字符编码为 GBK 或者 GB2312
  request.setCharacterEncoding("GBK");
  //读取用户名和密码
  String name = request.getParameter("name");
  String pwd = request.getParameter("pwd");
%>
```