# 第 3 章　JSP 内置对象

有如下代码片段：

```
<%@page language="java" pageEncoding="GBK" %>
<%request.setAttribute("aa","<h3>揭开 JSP 神秘面纱!</h3>");
   out.print(request.getAttribute("aa"));
%>
```

在代码中用到名为 out 的对象，但在整个页面中没有出现 new 关键字。也就是说没有实例化 out 对象，但在 JSP 程序中却可以使用它，同样，request 对象也是如此。这是什么原因呢？

原来 out 和 request 都是 JSP 的内置对象。所谓内置对象指在 JSP 页面中已经默认内置的 Java 对象。它们在 JSP 页面初始化时生成，由容器实现和管理。这些对象可以直接在 JSP 页面使用。经常使用的 JSP 内置对象有 out、request、response、session、application、config、pageContext 和 exception，下面将分别介绍。

## 3.1　out 对象

out 对象用来向客户端输出数据，被封装为 javax. servlet. jsp. JspWriter 类对象，通过 JSP 容器变换为 java. io. PrintWriter 类对象。Servlet 使用 java. io. PrintWriter 类对象向网页输出数据。

**例 3-1**

b. jsp：

```
<%@page language="java" pageEncoding="GB 2312"%>
<html>
  <body>
  <%
  out.print("<font size=4 color=red>");
  out.print("北大方正");
  out.print("</font>");
  %><!--在网页上输出"北大方正",字体 4 号,颜色红色-->
  </body>
</html>
```

运行结果如图 3.1 所示。

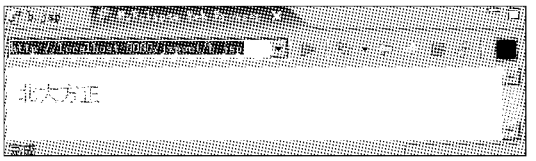


图 3.1　b.jsp 的运行结果

**注意**：out.print("北大方正");在浏览器上显示“北大方正”。

## 3.2　request 对象

HTTP 协议是客户与服务器之间提交请求信息(request)与响应信息(response)的通信协议。request 对象是从客户端向服务器发出请求，代表客户端请求信息，主要用于接收客户端通过 HTTP 协议传送给服务器的数据。该对象继承 ServletRequest 接口，被包装成 HttpServletRequest 接口。

request 对象常用方法如表 3.1 所示。

**表 3.1　request 对象常用方法**

| 方法名称 | 说　明 |
|---|---|
| String getParameter(String name) | 用来获取用户提交的数据 |
| String[] getParameterValues(String name) | 返回指定参数所有值 |
| setCharacterEncoding(String charset) | 设置响应使用字符编码格式 |
| void setAttribute(String name,java.lang.Object value) | 在请求转发时，经常要把一些数据传到转发后的页面处理，使用该方法 |
| Object getAttribute(String name) | 在请求转发后的页面使用该方法获取属性值 |
| removeAttribute(String attName) | 把设置在 request 范围内的属性删除 |
| getRemoteAddr() | 获得客户端 IP 地址 |
| Cookie[] getCookies() | 返回客户端 Cookie 对象，结果是一个 Cookie 数组 |

**例 3-2**

requesta.jsp：

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=GBK">
</head>
<body>
<form action="get.jsp" method="post" name="form1">
please enter your school name
```

```
<Input type="text" name="schoolname">
<Input type="submit" name="submit" value="提交">
</form>
</body>
</html>
```

get.jsp：

```
<%@page language="java" contentType="text/html; charset=GBK" %>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=GBK">
</head>
<body bgcolor="yellow">
<p>获取文本框提交的信息：
<%String textContent=new
String(request.getParameter("schoolname").getBytes("ISO 8859-1")); %>
<%=textContent %>
<p>获取按钮的标题：
<%String buttonName=new
String(request.getParameter("submit").getBytes("ISO 8859-1")); %>
<%=buttonName %>
</body>
</html>
```

运行结果如图 3.2 所示。

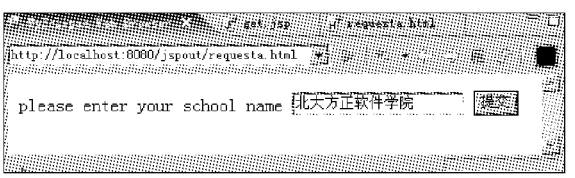


图 3.2　requesta.jsp 的运行结果

在文本框输入北大方正软件学院，单击“提交”按钮，运行结果如图 3.3 所示。

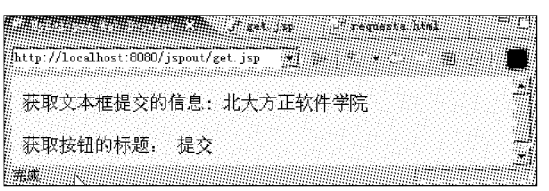


图 3.3　输入信息提交后结果

**注意**：在 requesta.jsp 中，文本框名字为 schoolname，在该文本框中输入“北大方正软件学院”，单击“提交”按钮，由表单 action 即 get.jsp 处理，request.getParameter("schoolname")得到用户提交的数据，在这里，

```
new String(
request.getParameter("schoolname").getBytes("ISO 8859-1"));
```

是用来处理汉字输入后显示为乱码的一种解决方法。

这样，在浏览器中显示用户在 requesta.jsp 的文本框 schoolname 中输入的“北大方正软件学院”。

**例 3-3**

third_example1.jsp：

```
<%@page language="java" contentType="text/html; charset=gb2312"%>
<html>
<script language="javascript">
function checkEmpty(form)
{
for(i=0;i<form.length;i++)
{
if(form.elements[i].value=="")
{
alert("表单信息不能为空");
return false;
}
}
}
</script>
<body>
  <form name="form1" method="post" action="a.jsp" onSubmit=
  "return checkEmpty(form1)">
    <table border="0">
      <tr>
        <td>输入姓名:</td>
        <td><input type="text" name="textOne"></td>
      </tr>
      <tr>
        <td>选择性别:</td>
        <td><INPUT type="radio" name="sex" value="男" checked="default">男</td>
        <td><INPUT type="radio" name="sex" value="女">女</td>
      </tr>
      <tr>
        <td>选择您喜欢的专业:</td>
        <td><input type="checkbox" name="item" value="NIIT">NIIT</td>
        <td><input type="checkbox" name="item" value="对日软件">对日软件</td>
        <td><input type="checkbox" name="item" value="中加合作">中加合作</td>
        <td></td>
      </tr>
```

```
        <tr>
          <td>隐藏域的值为:</td>
          <td><input  type="hidden" name="major" value="对日软件">对日软件</td>
        </tr>

      </table>
      <input type="submit" name="Submit" value="提交">
      <INPUT TYPE="reset" value="重置">
    </form>
</body>
</html>
```

a.jsp:

```
<%@page contentType="text/html; charset=GB 2312" %>
<html>
<%request.setCharacterEncoding("GB 2312");%>
<body><div align="center">
    <table border="0">
      <tr>
        <td>您的姓名:</td>
        <td><%=request.getParameter("textOne")%></td>
      </tr>
      <tr>
        <td>您的性别:</td>
        <td><%=request.getParameter("sex")%></td>
      </tr>
      <tr>
        <td>您喜欢的专业:</td>
        <%String itemName[]=request.getParameterValues("item");
        if(itemName==null)
       {  out.print("<td>"+"都不喜欢"+"</td>");
          out.print("</tr>");
       }
       else
       {  for(int k=0;k<itemName.length;k++)
          {  out.print("<td>"+itemName[k]+"</td>");
          }
           out.print("</tr>");
       }
       %>

      <tr>
        <td>获取隐藏域的值为:</td>
        <td><%=request.getParameter("major")%></td>
```

```
    </tr>
  </table>

  <a href="third_example1.jsp">返回</a>
  </div>
</body>
</html>
```

运行结果如图 3.4 所示。

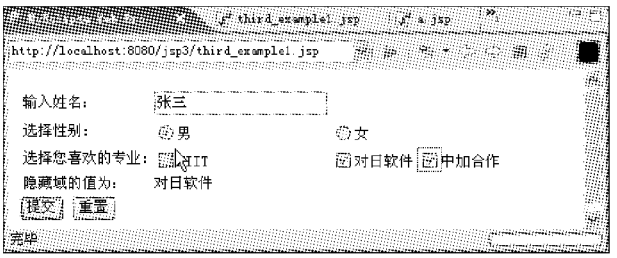


图 3.4　third_example1.jsp 的运行结果

输入“张三”，单击“提交”按钮，结果如图 3.5 所示。

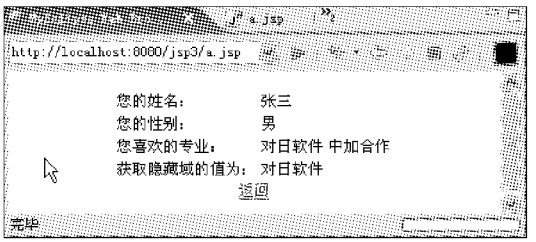


图 3.5　输入信息提交后结果

**注意**：在 a.jsp 中，request.setCharacterEncoding("GB 2312")；设置响应使用字符编码为 GB 2312。

在 a.jsp 中，String itemName[]＝request.getParameterValues("item")，返回 third_example1.jsp 中名字为 item 的检查框的所有值，它是一字符串数组。

**例 3-4**

third_example2.jsp：

```
<%@page contentType="text/html;charset=GB 2312"%>
<%
    request.setAttribute("name", "北大方正软件学院");
    request.setAttribute("stucount", "7000 人");
    request.setAttribute("tel", "010-51108888");
    request.setAttribute("city", "北京");
%>
<jsp:forward page="b.jsp" />
```

b.jsp：

```
<%@page contentType="text/html;charset=GB 2312"%>
<%request.removeAttribute("city"); %>
<table border="1">
    <tr>
        <td>学院名称:</td>
        <td><%=request.getAttribute("name")%></td>
    </tr>
    <tr>
        <td>学生人数:</td>
        <td><%=request.getAttribute("stucount")%></td>
    </tr>
    <tr>
        <td>电话:</td>
        <td><%=request.getAttribute("tel")%></td>
    </tr>
    <tr>
        <td>所在城市:</td>
        <td><%=request.getAttribute("city")%></td>
    </tr>
    <tr>
        <td>客户端 IP:</td>
        <td><%=request.getRemoteAddr()%></td>
    </tr>
</table>
```

运行结果如图 3.6 所示。

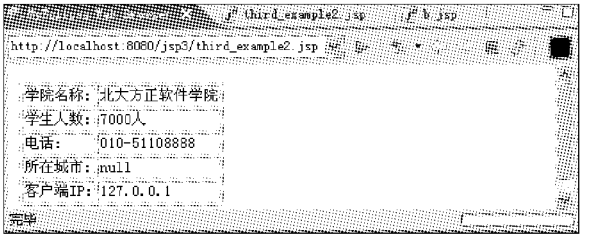


图 3.6　third_example2.jsp 的运行结果

**注意**：在 third_example2.jsp 中，request.setAttribute("name", "北大方正软件学院")；把数据“北大方正软件学院”设定在 request 范围内，转发后页面 b.jsp 使用 request.getAttribute("name")，得到数据“北大方正软件学院”。

request.getRemoteAddr()，返回提交数据的客户端 IP，本例为 127.0.0.1。

**例 3-5**

f.jsp：

```
<%@page contentType="text/html;charset=GB 2312"%>
```

```
<%
    String uName="John";
    String uSex="man";
    request.setAttribute("name", uName);
    request.setAttribute("sex", uSex);
%>
<jsp:forward page="rmAttribute.jsp"/>
```

rmAttribute.jsp：

```
<%@page contentType="text/html;charset=GB 2312"%>
<%
    request.removeAttribute("name");
    request.removeAttribute("sex");
%>
<table border="1">
    <tr>
        <td>姓名:</td>
        <td><%=request.getAttribute("name")%></td>
    </tr>
    <tr>
        <td>性别:</td>
        <td><%=request.getAttribute("sex")%></td>
    </tr>
</table>
```

运行结果如图 3.7 所示。

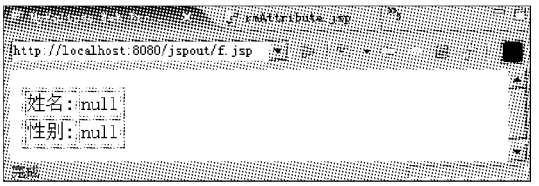


图 3.7　f.jsp 的运行结果

**注意**：在 rmAttribute.jsp 中，request.removeAttribute("name")；把设置在 request 中的属性 name 删除，所以 request.getAttribute("name")，显示 null。

## 3.3　response 对象

response 对象与 request 对象正好相反，所包含的是服务器向客户端作出的应答信息。response 被包装成 HttpServletResponse 接口，它封装了 JSP 的响应，被发送到客户端以响应客户端请求。因输出流是缓冲的，所以可以设置 HTTP 状态码和 response 头。

response 对象常用方法如表 3.2 所示。

表 3.2 response 对象常用方法

| 方法名称 | 说明 |
| --- | --- |
| addCookie(Cookie cookie) | 添加一个 Cookie 对象，用来保存客户端用户信息。用 request 对象的 getCookies()方法可以获得这个 Cookie |
| setContentType ( String contentType) | 设置响应 MIME 类型。例如：<br>response.<br>setContentType('application/msword;charset=GB 2312') |
| setCharacterEncoding (String charset) | 设置响应使用字符编码格式 |
| setHeader ( String name, String value) | 设定指定名字的 HTTP 文件头的值，如该值存在，会被新值覆盖。例如，在线聊天室，当 refresh 值为 5 时，就表示页面每 5 秒就要刷新一次 response. setHeader("refresh","5") |
| sendRedirect(URL) | 将用户重定向到一个不同的页面 URL。调用此方法，终止以前的应答，更改浏览器内容为一个新的 URL。注意：使用 sendRedirect 重定向是没办法通过 request. setAttribute 来传递对象到另外一个页面的 |
| String encodeURL ( String url) | 将 url 予以编码，回传包含 sessionId 的 URL。用 response. sendRedirect( response. encodeURL( url)) 的好处就是它能将用户的 session 追加到网址的末尾，也就是能够保证用户在不同的页面时的 session 对象是一致的。这样做的目的是防止某些浏览器不支持或禁用了 Cookie 导致 session 跟踪失败 |
| String encodeRedirectURL (String url) | 对于使用 sendRedirect()方法的 URL 进行编码 |

**例 3-6**

refresh. jsp：

```
<%@page language="java" contentType="text/html; charset=GBK" %>
<html>
<body>
<p>response 自动刷新</p>
当前时间为：
<%response.setHeader("Refresh","10");
out.println(""+new java.util.Date());%>
</body>
</html>
```

运行结果如图 3.8 所示。

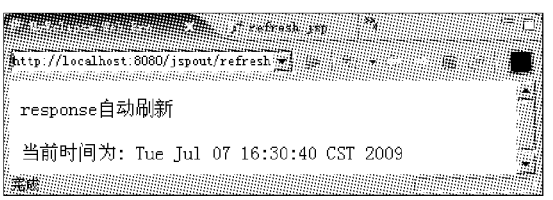


图 3.8 refresh. jsp 的运行结果

**注意**：response.setHeader("Refresh","10")指10秒钟后会重新加载页面本身，通过该方法可以设置页面自动刷新时间间隔。

**例 3-7**

cookie.jsp：

```
<%@page language="java" contentType="text/html; charset=GBK"%>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=GBK">
<title>保存数据到 cookie</title>
</head>
<%
request.setCharacterEncoding("GBK");
String name=request.getParameter("Name");
String major=request.getParameter("major");
Cookie cookies[]=request.getCookies();
//存取 name 变量
 if(cookies!=null)
{
    for(int i=0;i<cookies.length;i++)
    {if(cookies[i].getName().equals("name"))

        name=cookies[i].getValue();
    }
}
else if(name!=null)
    {
        Cookie c=new Cookie("name",name);
        c.setMaxAge(50);
        response.addCookie(c);
    }
//存取 major 变量
if(cookies!=null)
{
    for(int i=0;i<cookies.length;i++)
    {if(cookies[i].getName().equals("major"))

        major=cookies[i].getValue();
    }
}
else if(name!=null)
    {
        Cookie c=new Cookie("major",major);
```