第 8 章　JSP的内置对象

上一章介绍了 JSP 的页面构成以及标准动作的使用。这一章将重点向读者介绍 JSP 中预定义的内置对象，这些对象可以在 JSP 页面中调用，而不需要事先定义，这使得 JSP 编程更加的方便和快捷。其实这些内置对象都对应着某个 Servlet 类，在 JSP 被翻译成 Servlet 之后，这些内置对象会相应转换成对应的类实例。

JSP 中使用到的内置对象主要有如下多个：

- request 内置对象
- tesponse 内置对象
- page 内置对象
- session 内置对象
- application 内置对象
- out 内置对象
- exception 内置对象
- config 内置对象
- pageContext 内置对象

这里的内置对象在实际开发过程中会经常使用，希望读者能比较熟练的掌握它们的使用方法。

8.1　request内置对象

request 内置对象是最常用的对象之一，它代表的是 java.servlet.HttpServletRequest 类的对象。request 内置对象中包含了有关浏览器请求的信息，并提供了多个用于获取 cookie、header 以及 session 内数据的方法。

8.1.1　request对象常用方法

request 对象主要用于客户端请求处理，其中，该对象中所包含的方法有：

- getMethod()：返回 HTTP 请求信息中所使用到的方法名称；
- getServletPath()：返回请求信息中调用 Servlet 的 URL 部分；
- getQueryString()：返回 HTTP GET 请求信息中 URL 之后的查询字符串；
- getContentType()：返回请求实体的 MIME 类型；
- getProtocol()：返回请求信息中的协议名名字与版本号；
- getPathInfo()：返回有关任何路径信息；
- getServletName()：返回接受请求的服务器主机；
- getServletPort()：返回服务器的端口号；
- getRemoteHost()：返回提交请求的客户机的规范名字；
- getRemoteAddr()：返回提交请求的客户机的 IP 地址；

- getScheme()：返回请求使用的模式（协议）名字；
- getParameter()：返回客户端通过表单提交过来的参数值。例如 request.getParameter(“myname”)，通过该语句来获取客户端传递过来的 myname 参数。
- getContextPath()：返回 HTTP 请求中指示请求上下文的部分。
- getHeaderNames()：返回一个枚举类型，此枚举集合中包含了请求所含有的所有请求名。
- getAuthType()：返回用于保护 Servlet 的认证模式的名字。例如，BASIC，SSL 或者 NULL（没有保护）。
- gtRequestURL()：返回 HTTP 请求信息中的第一行从协议名开始直至查询字符串之间的 URL 部分。例如，对 HTTP GET 请求 http://www.zzbsite.com/helloworld?name=johnson&age=20，这个方法将返回 http://www.zzbsite.com/helloworld 字符串。
- gtCountLength()：返回整数，表示请求实体的长度（以字节为单位）。
- gtUestPrincipal()：返回 java.security 类的 Principal 对象，其中包含有目前授权用户的名字。
- iUserInRole(String role)：返回一个布尔值，指示某个授权用户是否包含在某个具体的逻辑角色 role 中。
- gtRemoteHost()：如果用户已经被授权，则返回提交请求的用户的注册名字，否则返回一个 NULL。

8.1.2　request常用方法的实例

下面通过一个实例讲解来让读者了解有关 request 内置对象中的常见调用方法。创建一个 request.jsp 文件，该文件的详细源代码如下：

```
<%@ page contentType="text/html;charset=GBK" %>
<html>
<head><title> request 内置对象的实例 </title></head>
<body>
      <form aciton="request.jsp">
            <br>Get request results:
            <br><input type="text" name="myname">
            <br><input type="submit" name="get value">
     </form>
返回 HTTP 请求信息中使用的方法名称:<%=request.getMethod()%>
<br>
返回请求信息中调用 Servlet 的 URL 部分:<%=request.getServletPath()%>
<br>
返回 HTTP GET 请求信息中 URL 之后的查询字符串:<%=request.getQueryString()%>
<br>
返回请求实体的 MIME 类型:<%=request.getContentType()%>
<br>
返回请求信息中的协议名名字和版本号:<%=request.getProtocol()%>
<br>
有关任何路径信息:<%=request.getPathInfo()%>
<br>
返回接受请求的服务器主机:<%=request.getServerName()%>
<br>
返回服务器的端口号:<%=request.getServerPort()%>
<br>
```

```
返回提交请求的客户机的规范名字:<%=request.getRemoteHost()%>
<br>
返回提交请求的客户机的 IP 地址:<%=request.getRemoteAddr()%>
<br>
返回请求中使用的模式（协议）名字:<%=request.getScheme()%>
<br>
返回这个 request 值，提交过来的值:<%=request.getParameter("myname")%>
</body>
</html>
```

程序说明：request 中的 getParameter()方法是最为常用的，使用此方法获取到上一页面所提交的参数值。此处，页面通过<form>提交了一个 myname 参数给本页面，并调用 request.getParameter("myname")获取到这个参数值。页面中的其他 request 方法是用来获取各种请求信息。

在 IE 浏览器中输入 http://localhost:8080/helloworld/request.jsp 地址，request.jsp 的页面效果如图 8.1 所示。

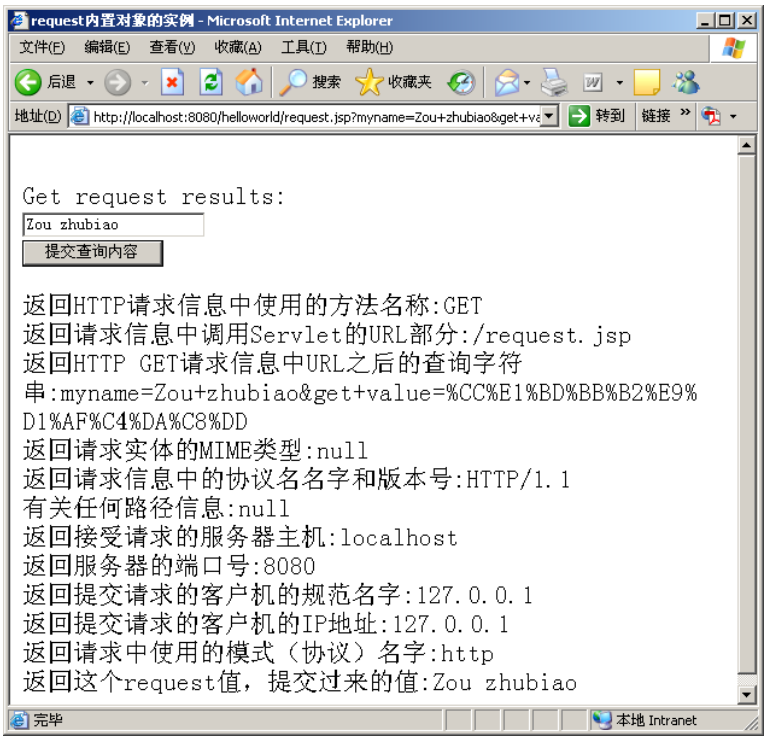

图 8.1　request.jsp 页面效果

有关 request 对象用于获取表单数据的方法将在第十二章实例中详细介绍。

8.2　response内置对象

response 对象与 request 对象相对应，它是用于响应客户请求，向客户端输出信息。response 是 javax.servlet.HttpServletResponse 类的对象。

8.2.1　response对象的常用方法

response 对象提供了多个方法用来处理 HTTP 响应，可以调用 response 中的方法修改 ContentType 中的 MIME 类型以及实现页面的跳转等等，比较常用的方法如下：

- setContentLength(int len)：此方法用于设置响应头的长度。
- setContentType(String type)：用于设置 HTTP 响应的 contentType 中的 MIME 类型，其中可以包含字符编码的规则。例如可以把 contentType 设置为“text/html;charset=GBK”。在 Servelet 编写过程中，需要调用此方法进行设置，但是在 JSP 中一般都是使用 page 指令直接指定 contentType 的属性。
- getOutputStream()：此方法返回一个 Servlet 的输出流。用于在响应中写入二进制数据。Servlet 容器不对二进制数据进行编码。
- getWriter()：此方法返回一个 PrintWriter 对象，在 Servlet 编写过程使用的比较频繁，而在 JSP 文件中，由于 out 是用 getWriter()创建的 PrintWriter 对象的隐含对象，所以可以直接调用 out 对象作输出响应。
- getCharacterEncoding()：该方法获得此时响应所采用的字符编码类型。
- sendError(int status)：使用指定错误状态码向客户机发送相应的错误信息。
- sendError(int status, String message)：使用自定义的错误状态码以及描述信息向客户机发送错误的提示信息。
- sendRedirect(String location)：将请求重新定位到一个不同的 URL（页面）上。此方法在实际开发过程中会经常使用到。
- setDateHeader(String headername, long date)：把指定的头名称以及日期设置为响应头信息。其中日期是用 long 值表示的，这是按照从新纪元开始算起的毫秒数。
- ContainsHeader(String name)：检测指定的头信息是否存在。返回一个布尔类型。
- setHeader(String headername, String value)：此方法使用指定的头名字以及相应的值来设置头信息。如果此头信息已经设置，则新的值会覆盖掉旧的值。如果头信息已经被发送出去，则此方法的设置将被忽略。
- addheader(String headername, String value)：把指定的头名字以及相应值添加到头信息当中去。
- addIntHeader(String headername, int value)：把指定的头名字以及整数值设置为头信息。如果头信息已经设置了，则新的设置值将覆盖掉以前的值。
- setStatus(int sc)：给响应设置状态的代码。
- setStatus(int sc, String sm)：为响应设置状态代码以及信息。这是在没有错误的时候使用的。

这些方法中，getWriter()和 sendRedirect(String location)在实际开发中使用的最为频繁。getWriter()常出现在 Servlet 编写中。

8.2.2　response对象的getWriter()方法实例

在服务器端的 Servlet 类文件中，会经常使用 getWriter()方法来获取一个 PrintWriter 对象，从而调用其中的 println()方法来向客户端输出内容。下面一段 Servlet 的代码实例：

```
package com.helloworld;
import java.io.PrintWriter;    //引入 PrintWriter 类
import javax.servlet.http.HttpServletResponse;        //引入 HttpServletResponse 类
```

```
public class PrintHTML {                                    //打印出 HTML 代码的方法
    public static void printHTML(HttpServletResponse response) throws Exception{
        PrintWriter out = response.getWriter();     //调用 HttpServletResponse 类中的 getWriter()方法。
        out.println("<table border='0' cellpadding='0' cellspacing='0' width='150' align='center'>");
        out.println("<tr><td height='5'>这是 HttpServletResponse 类中的 getWriter()方法的例子</td></tr>");
        out.println("</table>");
    }
}
```

程序说明：该 Java 代码动态地向客户端返回一个简单的 HTML 页面。

注意：在 JSP 页面中，response 就是 HttpServletResponse 类的一个对象，可以直接使用 response 在 JSP 页面中调用 HttpServletResponse 类中所有方法。

8.2.3　页面重定向实例

下面再通过一个例子来加深对用于页面重定向的 sendRedirect(String location)方法的理解。

在 helloworld 模块下创建一个 index4.jsp 文件：

```
<%@ page language="java" contentType="text/html;charSet=GBK" %>
<html>
<body>
<center><h3>response.sendRedirect()使用例子</h3></center>
<form action="index4.jsp">
    <table border=1><tr><td>
        <select name="pg">
            <option value=0>本页</option>
            <option value=1>hello 页面</option>
            <option value=2>goodbye 页面</option>
        </select>
    </td></tr>
    <tr><td><input type="submit" value="提交"></td></tr></table>
</form>
<%
    String pg = request.getParameter("pg");                    //获取传递参数 pg
    if("1".equals(pg))                                         //如果 pg 等于 1
        response.sendRedirect("hello.jsp");                    //则页面重定向为 hello.jsp
    else if("2".equals(pg))                                    //如果 pg 等于 2
        response.sendRedirect("goodbye.jsp");                  //则页面重定向为 goodbye.jsp
    else                                                       //否则不进行页面重定向，即还显示本页
        out.println("没有进行页面重定向");
%>
</body>
</html>
```

程序说明：页面中有个下拉菜单，选择需要跳转的页面。request 内置对象通过 getParameter()方法获取到传递过来的参数值，response 对象再根据参数值不同调用 sendRedirect()方法进行页面跳转。

重定向的 hello.jsp 页面代码如下：

```
<html>
    <body><%out.print("<center><h2>Hello!</h2></center>");%></body>
</html>
```

Goodbye.jsp 页面代码为：

```
<html>
    <body><%out.println("<center><h2>Goodbye!</h2></center>");%></body>
</html>
```

在 IE 浏览器中输入 http://localhost:8080/helloworld/index4.jsp 地址，显示的效果如图 8.2 所示，选择下拉菜单中的重定向页面，然后单击“提交”按钮就可以实现页面的跳转。

图 8.2　页面重定向

8.3　page内置对象

page 对象有点类似于 Java 编程中的 this 指针，就是指当前 JSP 页面本身。page 是 java.lang.Object 类的对象。

8.3.1　page对象的常用方法

比较常用的 page 内置对象的方法有：

- ❑ getClass()：返回当时 Object 的类。
- ❑ hashCode()：返回此 Object 的哈希代码。
- ❑ toString()：把此时的 Object 类转换成字符串。
- ❑ equals(Object o)：比较此对象是否和指定的对象是否相等。
- ❑ copy（Object o）：把此对象复制到指定的对象当中去。
- ❑ clone()：对此对象进行克隆。

由于 page 内置对象在实际开发过程并不经常使用，所以 page 对象的其他方法在这里就不一一列举出来了。

8.3.2　page的常用方法实例

下面举一个实例来加深对 page 内置对象使用的理解。创建一个 page.jsp 文件，其详细源代码如下：

```
<%@ page language="java" contentType="text/html;charSet=GBK" %>
<%@ page import="java.lang.Object" %>
<html>
    <body>
```

```
    <center><h3>Page 内置对象的实例</h3></center>
    <%!Object obj; %>                                    <! -- 对象申明 -->
    getClass:<%=page.getClass() %>
    <br>hashCode:<%=page.hashCode()%>
    <br>toString:<%=page.toString()%>
    <br>equals:<%=page.equals(obj) %>
    <br>equlas2:<%=page.equals(this) %>
  </body>
</html>
```

在浏览器中显示 page.jsp 页面，效果如图 8.3 所示。

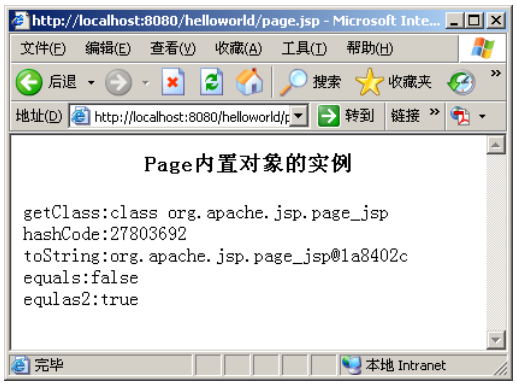

图 8.3　page.jsp 页面效果图

8.4　session内置对象

session 是与请求有关的会话期，它是 java.servlet.http.HttpSession 类的对象，用来表示和存储当前页面的请求信息。

在实际的 Web 应用开发过程会经常遇到这样的一个问题：会话状态的维持。当然有很多种方法可以用来解决这个问题，例如：Cookies、隐藏的表单输入域或者将状态信息直接附加到 URL 当中去，但是这些方法使用非常不便。

Java Servlet 提供了一个可以在多个请求之间持续有效的会话对象 HttpSession，此对象允许用户存储和提取会话状态的信息。JSP 同样也支持了 Servlet 中的这个概念。JSP 中的 session 内置对象就是对应于 Servlet 中的 HttpSession 对象。当 Web 应用系统希望通过多个页面完成一个事务的时候，session 的使用是非常有用和方便的。

8.4.1　session对象的常用方法

session 内置对象中的常用方法如下：

- ❑ getId()：此方法返回唯一的标识，这些标识为每个 session 而产生。当只有一个单一的值与一个 session 联合时，或当日志信息与先前的 sessions 有关时，它被当作键名用。
- ❑ getCreationTime()：返回 session 被创建的时间。最小单位为千分之一秒。为得到一个对打印输出很有用的值，可将此值传给 Date constructor 或者 GregorianCalendar 的方法 setTimeInMillis。
- ❑ getLastAccessedTime()：返回 session 最后被客户发送的时间。最小单位为千分之一秒。

- getMaxInactiveInterval()：返回总时间（秒），负值表示 session 永远不会超时。
- getAttribute(String key)：通过给定的关键字获取一个存储在 session 中相对应的信息。例如，Integer item = (Integer) session.getAttrobute("item")。
- setAttribute(String key, Object obj)：提供一个关键词和一个对象值，然后存在 session 当中。例如，session.setAttribute("ItemValue", itemName)。

session 一般在服务器上设置了一个 30 分钟的过期时间，当客户端停止操作后 30 分钟，session 中存储的信息会自动失效。

另外读者要非常注意的，session 中保存和查找的信息不能是基本的类型，如 int、double 等，而必须是 Java 相对应的对象，例如 Integer、Double 等。

8.4.2　问题回答操作实例

接下来本书将创建三个页面来模拟一个多页面的 Web 应用，使得读者能够对 session 的使用有深入的了解。第一个页面（session1.jsp）仅仅包含了一个要求输入用户名的 HTML 表单，代码如下：

```
<%@ page contentType="text/html;charSet=GBK" %>
<html>
    <body>
    <center><h3>用户名输入页面</h3></center>
    <!—提交表单 -->
    <form action="session2.jsp">
    <table border="1" align="center">
        <tr><td>用户名：<input type="text" name="username" size="10"></td></tr>
        <tr><td align="center"><input type="submit" value="提交"></td></tr>
    </table>
    </form>
</body>
</html>
```

程序说明：通过<form>把参数提交给 session2.jsp 页面进行处理。这一页面的效果如图 8.4 所示。

第二个页面（session2.jsp)需要通过 request 对象获取 session1.jsp 页面中的 username 参数值，并把它保存在 session 中。session 对象是以哈希表存储信息的。session2.jsp 的另外一个操作是询问第二个问题，具体的代码如下：

```
<%@ page contentType="text/html;charSet=GBK" %>
<html>
    <body>
    <center><h3>回答问题页面</h3></center>
<%
    String username = request.getParameter("username");     //获得传递参数 username
    session.setAttribute("theusername",username);           //把用户名保存在 session 中，String 可以当着对象
%>
    <p>您的用户名为：<%=username%><p>
    <!—提交表单 -->
    <form action="session3.jsp">
    <table border="1" align="center">
        <tr><td>您喜欢吃什么：<input type="text" name="food" size="10"></td></tr>
        <tr><td align="center"><input type="submit" value="提交"></td></tr>
    </table>
```

```
    </form>
</body>
</html>
```

程序说明：使用 request 内置对象中的 getParameter()方法获取到 session1.jsp 页面传递过来的参数值，并使用 session 对象中的 setAttribute()方法把用户名当着对象存储在 session 的哈希表中，这里需要指定一个关键字 theusername。另外页面使用<form>向 session3.jsp 页面递交了另外一个参数 food。这一页面的浏览器效果如图 8.5 所示。

图 8.4　session1.jsp 页面图

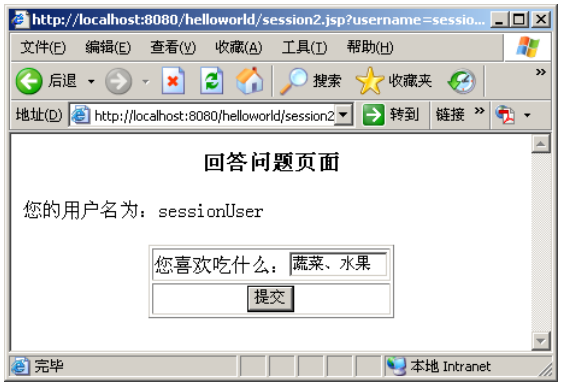

图 8.5　session2.jsp 页面图

第三个页面（session3.jsp）主要任务是显示回答结果。具体代码如下：

```
<%@ page contentType="text/html;charSet=GBK" %>
<html>
    <body>
    <center><h3>显示答案</h3></center>
    <%! String food="";%>
    <%
    food = request.getParameter("food");                    //取得 food 参数值
    String name = (String)session.getValue("theusername");  //从 session 取出关键字为 theusername 的对象
    %>
    您的用户名：<%=name%>
    <p>您喜欢吃：<%=food%>
</body>
</html>
```

程序说明：通过关键字 theusername 使用 session 对象中的 getAttribute(String key)方法获取到用户名，并把用户名和第二个问题的答案显示出来。session3.jsp 页面的浏览器效果如图 8.6 所示。

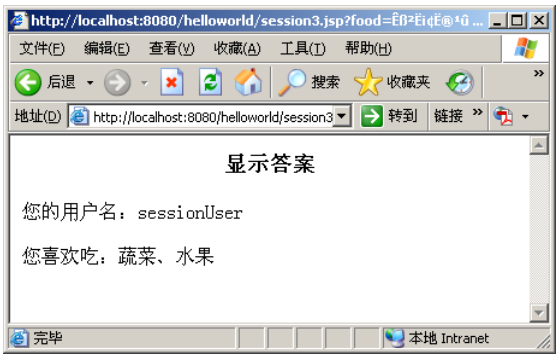

图 8.6　session3.jsp 页面图

session 内置对象的使用非常频繁，例如，使用 session 来存储用户的信息，并可以根据 session 中的用户对象是否为空来判断用户是否已经登陆。所以读者对此对象的使用要熟练掌握。

8.5　application内置对象

application 是 javax.servlet.ServletContext 类对象的一个实例，用于实现用户之间的数据共享（多使用于网络聊天系统）。

8.5.1　application对象与session对象的区别

它的作用有点类似于前一节介绍的 session 内置对象。但是它们之间还是有区别的，一般来说，一个用户对应着一个 session，并且随着用户的离开 session 中的信息也会消失，所以不同客户之间的会话必须要确保某一时刻至少有一个客户没有终止会话；而 applicat 则不同，它会一直存在，类似于系统的“全局变量”，而且只有一个实例。

8.5.2　application对象的常用方法

application 内置对象的常用方法如下：

- getAttribute(String key)：通过一个关键字返回用户所需要的信息，返回类型为对象（Object），类似于 session 中的 getAttribute(String key)方法。
- getAttributeNames()：返回所有可用的属性名，返回类型为枚举（Enumeration）。
- setAttribute(String key, Object obj)：保存一个对象信息，并指定给一个关键字。
- removeAttribute(String key)：通过关键字来删除一个对象信息。
- getServletInfo()：返回 JSP 引擎的相关信息。
- getRealPath(String path)：返回虚拟路径的真实路径。
- getContext(String URLPath)：返回执行 Web 应用的 application 对象。
- getMajorVersion()和 getMinorVersion()：返回服务器所支持的 Servlet API 最大和最小版本号。
- getMineType(String file)：返回指定文件的 MIME 类型。
- getResource(String path)：返回指定资源的 URL 路径。
- getResourceAsStream(String path)：返回指定资源的输入流。
- getRequestDispatcher(String URLPath)：返回指定资源的 RequestDispatcher 对象。
- getServlet(String name)：返回指定名称的 Servlet。
- getServlets()：返回所有的 Servlet，返回类型为枚举型。
- getServletNames()：返回所有的 Servlet 名称，返回类型为枚举型。
- log(String msg)：把指定信息写入到 Servlet 的日志文件中。
- log(String msg, Throwable throwable)：把栈轨迹以及给出的 Throwable 异常的说明信息写入 Servlet 的日志文件。

8.5.3　网站计数器实例

同样，下面将通过一个实例来讲解 Application 内置对象中常用方法的使用。

在模块 helloworld 中创建一个 setappattr.jsp 页面，用于获取 application 内置对象中的信息以及设置计数初始值，详细代码如下：

```
<%@ page contentType="text/html;charSet=GBK" %>
<html>
<h4>获得 application 信息</h4>
<br>ServletInfo:<%=application.getServerInfo()%>
<br>application.jsp real path: <%=application.getRealPath("/application.jsp")%>
<br>HelloServlet Real Path: <%=application.getRealPath("/servletsample/HelloServlet")%>
<br>Major Version: <%=application.getMajorVersion()%>
<br>get MIME: <%=application.getMimeType("/servletsample/demo.htm")%>
<br>getResource: <%=application.getResource("/HelloJSP.jsp")%>
<%   out.println("<br><h4>设置数值</h4>");
     application.setAttribute("name","zzb");                    //把字符串“zzb”对象保存在 application 中
     application.setAttribute("counter","1");                   //把字符串值“1”保存在 application 中
     out.println("set name=zzb");
     out.println("<br>set counter=1");
%>
</body>
</html>
```

程序说明：此处调用了 application 内置对象中的 setAttribute()方法来存储用户名信息以及计数初始值。

另外在相同目录下创建另外一个 getappattr.jsp 文件，用于获取计数值。具体的代码如下：

```
<%@ page contentType="text/html;charSet=GBK"%>
<html>
<body>
<br>获得用户名：<%=application.getAttribute("name")%>
<br>计数值：
<%
        //将保存在 application 中的关键字为 counter 的字符串对象取出，然后强制转化成整数型
   int mycounter = Integer.valueOf(application.getAttribute("counter").toString()).intValue();
   out.println(mycounter);
        //将数值加一，然后用新的值来更新保存再 application 中的 counter 对象
   application.setAttribute("counter",Integer.toString(mycounter+1));
%>
</body>
</html>
```

程序说明：和 session 对象一样，application 存储的是对象类型而不是普通的数值类型。此处调用了 application 对象中的 getAttribute()方法来获取前一个页面所存储的信息，并把读取出的计数值加一，然后重新存储在 application 当中去。

这两个页面所运行的效果如图 8.7 和 8.8 所示。

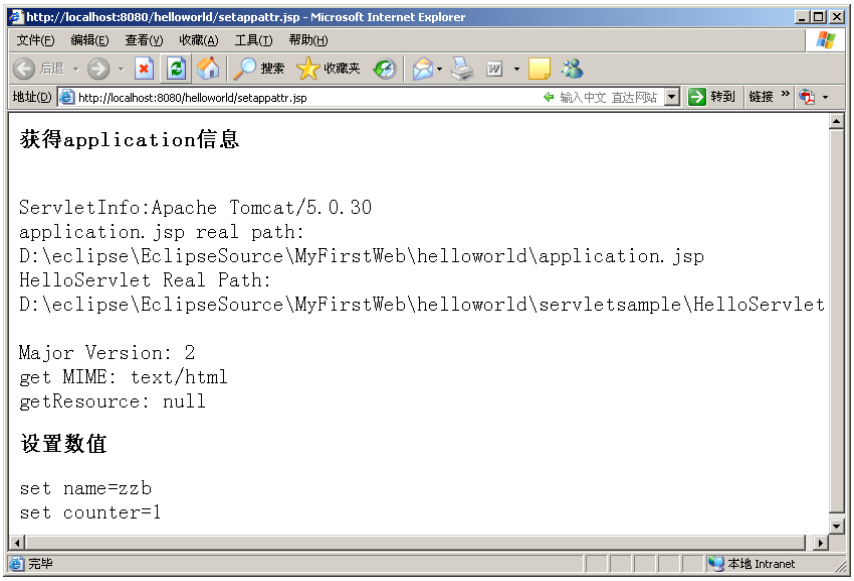

图 8.7　setappattr.jsp 运行页面

图 8.8　getappattr.jsp 运行页面

当关闭以上运行的两个浏览器窗口，再多次交替地打开和关闭 getappattr.jsp 窗口，会看到计数值一直在递增，只用 tomcat 服务不关闭。而 session 对象存储的信息会随着窗口的关闭而释放。

8.6　out内置对象

out 对象是在 JSP 开发过程中使用得最为频繁的对象，但使用也是最为简单的。

8.6.1　out对象的常用方法

out 对象的常用方法如下：

- print()：在页面中打印出字符串信息，不换行；
- println()：在页面中打印出字符串信息，并且换行；
- clear()：清除掉缓冲区中尚存在的内容。
- clearBuffer()：此方法清除掉当前缓冲区中尚存在的内容。
- flush()：清除掉数据流。
- getBufferSize()：返回缓冲区的内存大小，单位为字节流。如果不进行缓冲区的设置，大小为 0。
- getRemaining()：此方法返回缓冲区还剩下多少字节数可以使用。

- isAutoFlush()：检查当前缓冲区是设置为自动清空，还是满了就抛出异常。
- close()：关闭输出流。

其中 print()与 println()两个方法是使用最为频繁的。

8.6.2　数据输出实例

下面，同样将举一个例子来讲解 out 内置对象的使用方法。创建一个 out.jsp 文件，详细代码如下：

```
<%@ page buffer=”1kb” autoFlush=”true” contentType=”text/html;charSet=GBK” %>
<html>
<body>
<%   for(int i=0;i<135;i++)                          //迭代输出
        Out.println(“Hello world, “+i+”   “);
%>
<br>BufferSize: <%=out.getBufferSize() %>
<br>BufferRemain: <%=out.getRemaining() %>
<br>AutoFlush: <%=out.isAutoFlush() %>
<% out.clearBuffer(); %>
</body>
</html>
```

程序说明：page 指令中的 buffer 属性用来设置缓冲区的大小。autoFlush 属性为 true 表示缓冲区是自动清空的。在浏览器中运行这个 JSP 页面将会发现，程序只能输出到 i=106，后面的数字以及内容将全部被清空了，显示不出来。这是因为 out 对象调用的 clearBuffer()方法把缓冲区当前内容全部清除掉了。

把程序中的 clearBuffer()方法换成 clear()方法，再运行会报错。这是因为在调用 clear()方法之前，缓冲区已经自动清除过了（autoFlush=true）。如果把程序中的循环次数改小一点，则不管程序调用的是 clear()还是 clearBuffer()方法，浏览器上将什么也不显示。因为内容已经被 clear()或者 clearBuffer()方法清空掉了。但这时使用 clear()方法却不会出错，因为缓冲区这时还没有满，autoFlush 没有起到作用。

8.7　exception内置对象

exception 内置对象是用来处理页面出现的异常错误，它是 java.lang.Throwable 类的一个对象。前面已经讲过，在实际 JSP 网站开发过程中，通常是在其页面中加入 page 指令的 errorPage 属性来将其指向一个专门处理异常错误的页面。如果这个错误处理页面已经封装了这个页面收到的错误信息，并且错误处理页面页面含有的 isErrorpage 属性设置为 true，则这个错误处理页面可以使用以下方法来访问错误的信息：

- getMessage()和 getLocalizedMessage()：这两种方法分别返回 exception 对象的异常消息字符串和本地化语言的异常错误。
- printStackTrace()：显示异常的栈跟踪轨迹。
- toString()：返回关于异常错误的简单消息描述。
- fillInStackTrace()：重写异常错误的栈执行轨迹。

异常错误一般都是开发人员无法避免的，所以对各种可能的异常进行后期的处理和提示是非常必要的。读者要养成及时处理各种异常错误的习惯。

8.8　config内置对象

config 内置对象是 ServletConfig 类的一个实例。在 Servlet 初始化的时候，JSP 引擎通过 config 向它传递信息。这种信息可以是属性名/值匹配的参数，也可以是通过 ServletContext 对象传递的服务器的有关信息。config 内置对象中常用的方法如下。

- getServletContext()：此方法将返回一个含有服务器相关信息的 ServletContext 对象。
- getInitParameter(String name)：返回初始化参数的值。
- getInitParameterNames()：返回包含了 Servlet 初始化所需要的所有参数，返回类型是枚举型。

一般在 JSP 开发过程很少使用到 config 内置对象。只有在编写 Servlet 时，需要重载 Servlet 的 init() 方式时才用到。

8.9　pageContext内置对象

pageContext 对象是一个比较特殊的对象。它相当于页面中所有其他对象功能的最大集成者，即使用它可以访问到本页面中所有其他对象，例如前面已经描述的 request、response 以及 application 对象等。

8.9.1　pageContext对象的常用方法

这个对象中常使用的方法如下：

- getRequest()：返回当前页面中的 request 对象。
- getResponse()：使用此方法将返回当前页面中的 response 对象。
- getPage()：此方法返回当前页面中的 page 对象。
- getSession()：返回当前页面中的 session 对象。
- getOut()：返回当前页面中的 out 对象。
- getException()：返回当前页面中的 exception 对象。
- getServletConfig()：返回当前页的 config 对象。
- getServletContext()：返回当前页中的 application 对象。
- setAttribute(String name)：给指定的属性名设置属性值。
- getAttribute(String name)：根据属性名称找到相应的属性值。
- setAttribute(String name, Object obj, int scope)：在给定的范围内设置相应的属性值。
- getAttribute(String name, int scope)：在指定的范围内获取到相应的属性值。
- findAttribute(String name)：寻找一个属性并返回，如果没有找到则返回一个 null。
- removeAttribute(String name)：通过属性名删除掉某个属性。
- removeAttribute(String name, int scope)：在指定的某个范围内删除某个属性。
- getAttributeScope(String scope)：返回某属性的作用域。
- getAttributeNamesInScope(int scope)：返回指定范围内的所有属性名的枚举。
- release()：释放掉 pageContext()所占的所有资源。
- forward(String relativeURLpath)：使用当前页面重导到另一个页面。
- include(String relativeURLpath)：使用当前位置包含的另一个页面。

之上提到的 scope 范围的取值含义如表 8.1 所示。

表 8.1　scope取值及其含义

取值	含义
1	Page scope
2	Request scope
3	Session scope
4	Application scope

8.9.2　pageContext对象的简单实例

下面为应用 pageContext 对象的示范例子：

```
<html>
<body>
<%   request.setAttribute(“MyName”,”zzb1”);                    //把 MyName 保存在 request 范围中
     session.setAttribute(“MyName”,”zzb2”);                    //将 MyName 再保存再 session 范围中
     application.setAttribute(“MyName”,”zzb3”);                //将 MyName 保存在 application 范围中
%>
request: <%=pageContext.getRequest().getAttribute(“MyName”)%>
<br>session: <%=pageContext.getSession().getValue(“MyName”)%>
<br>application: <%=pageContext.getServletContext().getAttribute(“MyName”)%>
</body>
</html>
```

pageContext 对象在实际 JSP 开发过程中很少使用，因为像 request 和 response 等对象本来就可以直接调用方法进行使用，如果通过 pageContext 来调用其他对象就有点舍近求远。

8.10　本章小结

这一章节主要介绍了 JSP 中比较常用的内置对象，内置对象的可直接调用特性使得 JSP 开发更加的方便和快捷。

request 内置对象主要用于客户端请求处理，例如调用 getParameter()方法来获取客户端传递过来的参数值；response 对象与 request 对象相对应，它是用于响应客户请求，向客户端输出信息，例如调用 sendRedirect()方法进行页面重定向。page 对象类似于 this 指针，就是指当前 JSP 页面本身；session 是与请求有关的会话期，它非常有用，常用来保存用户登录状态；类似于 session 内置对象，但它的生命周期更长，常实现聊天室功能；out 对象是在 JSP 开发过程中使用得最为频繁的对象，用来在页面输出数据；exception 内置对象用来处理页面出现的异常错误；config 内置对象在实际开发过程很少使用，它用来在 Servlet 初始化传递信息。pageContext 对象是一个比较特殊的对象。它相当于页面中所有其他对象功能的最大集成者，即使用它可以访问到本页面中所有其他对象。

结合前几章的学习，至此，读者应该对网站的构架以及 JSP 文件的编写再也不感到陌生了。