Web语言程序设计

第五章： JSP内部对象

第五章： JSP内部对象

1. JSP内建对象简述
2. out对象
3. request对象
4. response对象
5. application对象

6 session对象

7 PageContext对象

8 config对象

9 page对象

10 exception对象

第五章： JSP内部对象

1 JSP内建对象简述

JSP规范中定义了9种内建对象，它们分别为request 、 response 、 out 、 session 、 application、 config、 pageContext、 page和exception，在JSP中并不需要编写任何额外的代码就可以自动使用这些内建对象。

第五章： JSP内部对象

① request对象：来自客户端的请求，此请求包括GET/POST请求方法的参数。

② response 对象：对客户端的响应。

③ session对象：与请求有关的会话。

④ application对象：代码段的运行环境。

⑤ out 对象：传送响应的输出信息流。

⑥ pageContext对象：页面的管理属性。

⑦ config对象：代码段的配置对象。

⑧ page对象：JSP页面本身。

⑨ exception对象：JSP页面运行时抛出的异常。

第五章： JSP内部对象

- JSP内部对象的方法实际上都源于Servlet API包中提供的各种类的成员方法。Servlet API包主要包括：
- ① javax.servlet；
- ② javax.servlet.http；
- ③ javax.servlet.jsp。

第五章： JSP内部对象

内部对象	所 属 类 型	用　途	作用范围
request	javax.servlet.httpServletRequest	包含了请求方的信息	request
response	javax.servlet.httpServletResponse	用以设定JSP回应信息的资料设定	page
out	javax.servlet.jsp.JspWriter	响应信息流的标准输出	page
session	javax.servlet.Http.HttpSession	在同一请求中所产生的session资料，目前只对Http协议有定义	session
application	javax.servlet.ServletContext	提供安全信息	application
config	javax.servlet.ServletConfig	提供配置信息	page
pageContext	javax.servlet.jsp.PageContext	提供当前页面属性	page
page	java.lang.Object	同于java的this	page
exception	java.lang.Throwable	异常处理	page

2 out对象

out对象是javax.servlet.jsp.JspWriter类的一个子类的对象，它的作用是把信息回送到客户端的浏览器中。在out对象中，最常用的方法就是print()和println()。在使用print()或println()方法时，由于客户端是浏览器，因此向客户端输出时，可以使用HTML中的一些标记，例如：“out.println(”<h1>Hello,JSP</h1>“);”。

2 out对象

- **clear()**：清除缓冲区中的内容，不将数据发送至客户端。
- **clearBuffer()**：清除缓冲区中的内容，并将数据发送至客户端。
- **close()**：关闭输出流。
- **flush()**：输出缓冲区中的数据。
- **getBufferSize()**：获取缓冲区的大小。
- **getRemaining()**：获取缓冲区的剩余空间。
- **newLine()**：输出一个换行字符，换一行。
- **print()**：显示各种数据类型的内容。
- **println()**：分行显示各种数据类型的内容。

2 out对象

程序5-1.jsp

```
<%@ page contentType="text/html;charset=GBK" %>
<%
        out.println("hello");
        out.newLine();
        out.write("hello");
%>
<%="hello"%>
<%
  out.close();
%>
```

3 request对象

request对象的作用是获取客户端所需要的信息。request对象被封装为javax.servlet.http.HttpServletResquest接口。

当客户端请求一个JSP页面时，JSP引擎会将客户端的请求信息包装在这个request对象中。请求信息的内容包括：请求的标头（Header）信息（如浏览器的版本名称、语言和编码方式等）、请求的方式（HTTP方法：如GET、POST和PUT，<Form>的method属性设定值）、请求的参数名称和参数值、客户端的主机名称等，通过request对象获取有关客户端的请求信息 。

第五章： JSP内部对象

- （1）getParameter() 方法
- （2）getAttribute()方法
- （3）getAttributeNames()方法
- （4）getContentLength()方法
- （5）getContentType()方法
- （6）getCharacterEncoding()方法
- （7）getInputStream()方法
- （8）getParameterNames()方法
- （9）getParameterValues()方法

第五章： JSP内部对象

- （10）getScheme()方法
- （11）getProtocol()方法
- （12）getServerPort()方法
- （13）getServerName()方法
- （14）getReader()方法
- （15）getRemoteAddr()方法
- （16）getRemoteHost()方法
- （17）getRealPath()方法
- （18）setAttribute()方法

第五章： JSP内部对象

- (19) getAuthType()方法
- (20) getCookies()方法
- (21) getDataHeader()方法
- (22) getHeader()方法
- (23) getHeaderNames()方法
- (24) getIntHeader(String str)方法
- (25) getMethod()方法
- (26) getPathInfo()方法
- (27) getPathTranslated()方法

第五章： JSP内部对象

- （28）getQueryString()方法
- （29）getRemoteUser()方法
- （30）getRequestedSessionId()方法
- （31）getRequestURI()方法
- （32）getServletPath()方法
- （33）getSession()方法
- （34）isRequestedSessionIdValid()方法
- （35）isRequestedSessionIdFromCookie()方法
- （36）isResquestedSessionIdFromURL()方法

常用的request方法

1）getParameter(String strTextName) 获取表单提交的信息。

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

2） getProtocol() 获取客户使用的协议。

String strProtocol=request.getProtocol();

3） getServletPath() 获取客户提交信息的页面。

String strServlet=request.getServletPath();

4）getMethod() 获取客户提交信息的方式，get|post。

String strMethod = request.getMethod();

5）getHeade() 获取HTTP头文件中的accept、accept-encoding和Host的值。

String strHeader = request.getHeader("accept");

6）getRemoteAddr() 获取客户的IP地址。

String strIP = request.getRemoteAddr();

常用的request方法

7）getRemoteHost() 获取客户机的名称。

```
String clientName = request.getRemoteHost();
```

8）getServerName() 获取服务器名称。

```
String serverName = request.getServerName();
```

9）getServerPort() 获取服务器的端口号。

```
int serverPort = request.getServerPort();
```

10）getParameterNames() 获取客户端提交的所有参数的名字。

```
Enumeration enum = request.getParameterNames();
while(enum.hasMoreElements()){
String s=(String)enum.nextElement();
out.println(s);
}
```

第五章： JSP内部对象

request对象的作用

3.1.获取请求参数

1）String getParameter(String paramName)：

获取paramName请求参数的值。

2）Map getParameterMap()：

获取所有请求参数名和参数值所组成的Map对象。

3）Enumeration getParameterNames()：

获取所有请求参数名所组成的Enumeration对象。

4） String[] getParameterValues(String name)：
paramName请求参数的值，当该请求参数有多个值时，该方法将返回多个值所组成的数组。

第五章： JSP内部对象

程序5-2.jsp

```
<HTML>
<BODY>
        <FORM ACTION="4-15.jsp" METHOD="POST">
<P>姓名：<INPUT TYPE="TEXT" SIZE="20" NAME="UserID"></P>
<P>密码：<INPUT TYPE="PASSWORD" SIZE="20" NAME="UserPWD"></P>
<P><INPUT TYPE="SUBMIT" VALUE="提 交"> </P>
        </FORM>
</BODY>
```

程序5-3.jsp

```
<%@ page contentType="text/html;charset=GBK" %>
<%
request.setCharacterEncoding("GBK");
String strUserName = "";
String strUserPWD = "";
strUserName = request.getParameter("UserID");
strUserPWD = request.getParameter("UserPWD");
%>
姓名：<%=strUserName%><br>
密码：<%=strUserPWD%>
```

第五章： JSP内部对象

```
<FORM method="post" action="2.jsp">
  <INPUT type=checkbox name="c1" value="beijing">北京<BR>
  <INPUT type=checkbox name="c1" value="tianjin">天津<BR>
  <INPUT type=checkbox name="c1" value="hebei">河北<BR>
  <INPUT type=submit value="提交">
  </FORM>

  <%
    String[] s=request.getParameterValues("c1");
    for(int i=0;i<s.length;i++){
          out.print(s[i]);
    }
  %>
```

Request应用

❖3.2.操作request范围的属性

❖ 1) setAttribute(String attName , Object attValue)：

作用：将attValue设置成request范围的属性值。

❖ 2) Object getAttribute(String attName)：

作用：获取request范围的属性。

Request应用

- 3.3.解决汉字输出问题
- 1）在执行获取请求参数前设置编码

```
request.setCharacterEncoding(“汉字编码”)
```

- 2）转换字符编码

```
//获取原始的请求参数值
String rawName = request.getParameter("name");
//将请求参数值使用ISO-8859-1字符串分解成字节数组
byte[] rawBytes = rawName.getBytes("ISO-8859-1");
//将字节数组重新编码成字符串
String name = new String(rawBytes , "gb2312");
```

3）获取请求参数同时转换编码

```
request.getParameter(“name”).getBytes ("ISO-8859-1");
```

第五章： JSP内部对象

Request应用

- 3.4.获取协议头、服务器、客户端等信息。
- 见程序request.jsp

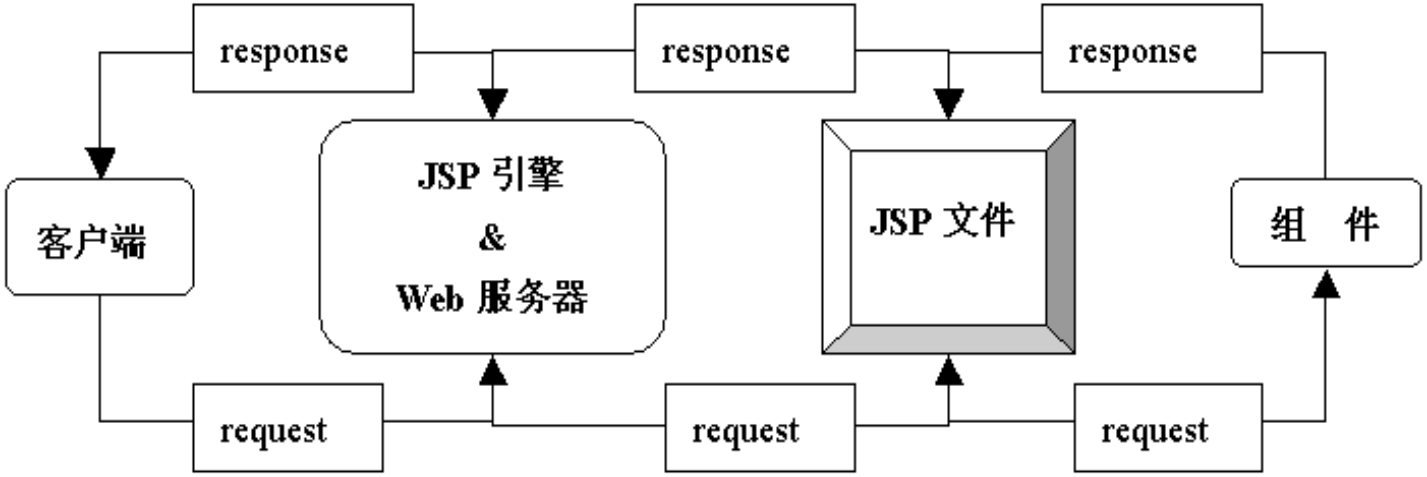
获取用户信息例子

request方法可以获取用户信息：程序5-6.jsp

```
<BR>客户使用的协议是: <% String protocol=request.getProtocol();
<BR>获取接受客户提交信息的页面：<% String path=request.getServletPath();
<BR>接受客户提交信息的长度：<% int length=request.getContentLength();
<BR>客户提交信息的方式：<% String method=request.getMethod();
<BR>获取HTTP头文件中User-Agent的值：： <% String header1=request.getHeader("User-Agent");
<BR>获取HTTP头文件中accept的值：<% String header2=request.getHeader("accept");
<BR>获取HTTP头文件中Host的值：<% String header3=request.getHeader("Host");
<BR>获取HTTP头文件中accept-encoding的值：<% String header4=request.getHeader("accept-
encoding");
<BR>获取客户的IP地址：<% String  IP=request.getRemoteAddr();
<BR>获取客户机的名称：<% String clientName=request.getRemoteHost();
<BR>获取服务器的名称：<% String serverName=request.getServerName();
<BR>获取服务器的端口号：<% int serverPort=request.getServerPort();
```

4 reponse对象

request对象是使用非常频繁的对象之一，然而request对象功能的实现离不开response对象的支持，request对象和response对象的结合可以使JSP更好地实现客户端与服务器端的信息交互，下图显示了客户端与服务器端信息交互的流程。



4 reponse对象

response对象的作用是封装JSP产生的响应，然后将其发送至客户端以响应客户的请求。response对象被封装为javax.servlet.http.HttpServletResponse接口。JSP引擎会根据客户端的请求信息建立一个预设的response回应对象。它是用来提供给客户端浏览器的参考的信息，如回应的标头、回应本体（如HTML文本的内容）以及服务器端的状态码信息 。

response 应用

4.1.相应生成图片

```
ImageIO.write(BufferedImage, "bmp",
response.getOutputStream());
```

Response应用

- 4.2.重定向页面

response.sendRedirect(“URL”)

- 4.3.HTTP文件头相应

Response.setHeader(“Refresh”, “时间”)

4.4.动态ContentType响应

response.setContentType(“文档格式”;charset=编码方式)

其中文档格式可以是text/plain(文本文件);application/x-msexcel(excel文件);application/msword(Word文件)

Response应用

❖4.5.设置清除页面缓冲区

```
response.setHeader("Pragma", "no-cache")

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

response.setDateHeader("Expires", -1);
```

4. 6网页跳转例子

response对象，最常用到的是sendRedirect()方法， 与<jsp:forward>指令的区别？

<jsp:forward>只能在站内跳转，

response.sendRedirect()可以在任意的网址跳转。

程序5-9.jsp

4.6 网页跳转例子

案例名称：网页转向
程序名称：5-9.jsp

```
<%@ page contentType="text/html;charset=GBK" %>
<%
        response.sendRedirect(“5-01.jsp”);//任意跳转
%>

<jsp:forward page=http://localhost:8080/3/3-1.jsp />
//站内跳转
```

4.6 网页跳转

解决权限设置问题。

程序5-10.jsp、5-11.jsp、5-12.jsp、5-13.jsp和5-14.jsp

5-10输入信息，5-11中间处理，5-12管理员界面，5-13普通用户界面、5-14.jsp密码错误页面。

第五章： JSP内部对象

案例名称：权限问题

程序名称：5-10.jsp：输入信息

```
<FORM method="post"  action="4-11.jsp">
  用户名：<INPUT type="text" size="20" name="userid">
  密码：<INPUT type="password" name="password">
  <INPUT type="submit" value="提交">   </FORM>
```

程序名称：5-11.jsp：中间处理

```
 <%  request.setCharacterEncoding("GBK");
 String userid=request.getParameter("userid");
 String password=request.getParameter("password");%>
   <%
if(userid.equals("admin")&&password.equals("admin")){
 response.sendRedirect(“5-12.jsp");  }
 f(userid.equals("admin1")&&password.equals("admin1")){
 response.sendRedirect(“5-13.jsp");  }
 else{  response.sendRedirect(“5-14.jsp"); } %>
```

4.6 动态contentType响应例子

当请求jsp页面的时候，page指令设置页面的contentType属性是text/html，服务器端将静态页面发给客户端，可以通过设置contentType来实现其他输出格式。

text/plain(文本文件)；application/x-msexcel(Excel文件)和application/msword（word文件）。

程序5-15.jsp

4.6 动态contentType响应例子

案例名称：显示为**Word**文档
程序名称：**5-15.jsp**

```
<%@ page contentType="text/html;charset=GBK" %>
<HTML>
        <BODY>
        <P>response对象 <BR>setContentType方法
        <P>将当前页面保存为word文档吗
        <%
        response.setContentType("application/msword;charset=GB2312");
        %>
        </BODY>
</HTML>

 // response.setContentType("application/msword;charset=GB2312");
   // response.setContentType("application/x-msexcel");
 //response.setContentType("text/plain");
```

4.6 HTTP文件头响应例子

可以利用jsp改变客户端的响应。

response.setHeader()。

程序5-16.jsp

自动刷新

4.6 HTTP文件头响应例子

案例名称：动态改变响应头
程序名称：5-16.jsp

```
<%@ page contentType="text/html;charset=GBK" %>
<%@ page import="java.util.*" %>
<P>现在的时间是：<BR>
<% out.println(""+new Date());
   response.setHeader("Refresh","5");
 %>
```

4.6 HTTP文件头响应例子

案例名称：动态改变响应头

程序名称：5-16.jsp

```
<%@ page contentType="text/html;charset=GBK" %>
<%@ page import="java.util.*" %>
<P>现在的时间是：<BR>
<% out.println(""+new Date());
   response.setHeader("Refresh","5");
 %>
```

第五章： JSP内部对象

5 application对象

application对象的主要作用是为多个应用程序保存信息，直至服务器关闭为止。application对象被封装为javax.servlet.SercletContext接口的类型。

application对象负责提供应用程序在服务器中运行时的一些全局信息，因此我们可以利用application来获取一些系统相关信息。

5 application对象

网站的所有用户公用一个application对象，当文web服务器启动时，application对象被创建，直到服务器关闭。

网络信息交流可分为在线交流和离线交流，在线交流如QQ、ICQ等；离线交流如BBS、BLOG等。应用JSP技术很容易实现这些交流工具。

5.1 application对象自定义属性

可以为application对象添加属性，application对象最常用的方法有两个：

Public void setAttribute（String key, Object obj）：将对象obj添加到application对象中，并添加索引关键字key；

Public Object getAttribute（String key）：获取application对象还有key关键字的对象。

注：任何对象都可以添加到application对象中，返回的时候需要强制转换为原来的类型

5.1 application对象自定义属性

程序4-18.jsp

```
<%String str="web语言程序设计";
  application.setAttribute("greeting",str);  %>
    <% String
strback=(String)application.getAttribute("greeting");%>
<%=strback%>
```

程序首先对application的greeting属性进行赋值，然后将其输出到浏览器上。执行完后该对象被保存到服务器上。

5.1 application对象自定义属性

程序4-19.jsp

虽然没有赋值，但是依然可以输出application对象的对象。

```
<%
String str=(String)application.getAttribute("greeting");
%>

<%=str%>
```

5.1 application对象自定义属性

注意：

Application对象不会因为某一个或是全部用户离开就消失，一旦建立application对象，它就会一直保存到网站关闭或是这个application对象被卸载。

removeAttribute()方法

形式：removeAttribute(String name1)

说明：删除指定属性的值。使用此方法后，所操作了的属性的值为空值。

程序4-20.jsp

5.1 application对象自定义属性

程序4-20.jsp

```
<% application.removeAttribute("greeting");
 String
str=(String)application.getAttribute("greeting");
 %>
 <body>
   <%=str%> <br>
 </body>
```

5.2 实现聊天室

聊天室运行多用户实时进行信息交流，所有用户可以看见彼此信息，这与application对象的特点相符，所以可以利用application对象实现聊天室功能。

程序4-21.jsp

第五章： JSP内部对象

案例名称：实现聊天室

程序名称：4-21.jsp

```
<%
if(application.getAttribute("mywords")==null||application.getAttribute("mywords").
equals("")){
application.setAttribute("mywords","开始聊天： "); }
if(request.getParameter("mywords")!=null){
String mywords=new String(request.getParameter("mywords").getBytes("ISO-
8859-1"));
System.out.println(mywords);
mywords=(String)application.getAttribute("mywords")+"<br>"+mywords;
application.setAttribute("mywords",mywords);
}
String aa=(String)application.getAttribute("mywords");
%>
<%=aa%>
<FORM ACTION="4-21.jsp" METHOD="get">
<INPUT TYPE="TEXT" SIZE="30" NAME="mywords" VALUE="I LIKE CHAT">
        <INPUT TYPE="SUBMIT" name="submit" VALUE="提交">
        </FORM>
```

5.3 网页计数器

网页计数器是application的另外一个用途，由于application是公共所有的，所以可以存储计数器的值，当有新用户访问时，计数器自动增加。程序4-22.jsp

```
<%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);}%>
```

Integer 类在对象中包装了一个基本类型 int 的值。Integer 类型的对象包含一个 int 类型的字段。

此外，该类提供了多个方法，能在 int 类型和 String 类型之间互相转换，还提供了处理 int 类型时非常有用的其他一些常量和方法。

6 session对象

session对象的作用是记录每个客户端的访问状态，以便跟踪每个客户端的操作状态。session对象被封装为javax.servlet.http.HttpSession接口，当客户端请求超过一个以上的JSP程序网页时，session对象提供有保存请求时期对象属性的方法，所保存的对象在请求过程中都是有效的。

第五章： JSP内部对象

6 session对象

session对象指的就是客户端与服务器端的一次会话。

session对象是一个非常重要的工具，应用极广，主要用来保存各个用户个人信息。

与application之间的区别？

第五章： JSP内部对象

6.1 对session对象的理解

当一个用户登录一个网站的时候，系统会对用户自动分配一个session，可以使用getID()方法获得session的ID。 程序4-23.jsp

形式：String getId()

说明：获得一个session的编号，这编号是一个session惟一标识符。每生成一个session时，服务器便会给它一个独一无二编号。

6.1 对session对象的理解

案例名称：动态改变响应头

程序名称：**4-23.jsp**

```
<%String str=session.getId(); %>
 <body>
  你的sessionID：<%=str%>
 </body>
```

6.2 自定义属性

session对象中最常用的就是setAttribute()、getAttribute()和removeAttribute()方法。

（1）getAttribute()方法

形式：String getAttribute(String attribute1)

说明：获得指定名字的属性，如果该属性不存在，则返回null。

（2）setAttribute()方法

形式：String setAttribute(String attribute1)

说明：设定指定名字的属性值，并将其存储到session对象中。

程序4-24.jsp和4-25.jsp

第五章： JSP内部对象

6.1 对session对象的理解

案例名称：自定义属性
程序名称：**4-24.jsp**

```
<%  String aa="欢迎";
  session.setAttribute("aa",aa);
  String bb=(String)session.getAttribute("aa");%>
<%=bb%>
   <a href="4-25.jsp">下一页</a>
```

程序名称：**4-25.jsp**

```
 <%
//session.removeAttribute("aa");
String bb=(String)session.getAttribute("aa");%>
<%=bb%> <br>
```

6.3 简易版本购物车

利用session保存用户选购的商品信息。

程序buy1.jsp

程序buy2.jsp

程序display.jsp

案例名称：简易版本购物车

程序名称：**buy1.jsp**

```
<%  request.setCharacterEncoding("gbk");
  if(request.getParameter("c1")!=null){
  session.setAttribute("s1",request.getParameter("c1")); }
   if(request.getParameter("c2")!=null){
  session.setAttribute("s2",request.getParameter("c2")); }
   if(request.getParameter("c3")!=null){
  session.setAttribute("s3",request.getParameter("c3"));}%>
  <body>
  各种肉类大甩卖，一律8元：
    <FORM action="buy1.jsp" method="post">
    <p> <INPUT type="checkbox" name="c1" value="猪肉">猪肉</p>
     <p> <INPUT type="checkbox" name="c2" value="牛肉">牛肉</p>
     <p> <INPUT type="checkbox" name="c3" value="羊肉">羊肉</p>
          <P><INPUT type="submit" name="b1" value="提交">
          <INPUT type="reset" name="b2" value="取消"></P>
                    <A href="buy2.jsp">买点别的</A> <A
href="display.jsp">查看购物车</A>
    </FORM>
```

第五章： JSP内部对象

程序名称：**buy2.jsp**

```
<%  request.setCharacterEncoding("gbk");
 if(request.getParameter("c4")!=null){
 session.setAttribute("s4",request.getParameter("c4")); }
  if(request.getParameter("c5")!=null){
 session.setAttribute("s5",request.getParameter("c5"));}
  if(request.getParameter("c6")!=null){
 session.setAttribute("s6",request.getParameter("c6"));} %>
 <body>
  各种球类大甩卖，一律10元：
  <FORM method="post" action="buy2.jsp">
  <P><INPUT type="checkbox" name="c4" value="足球">足球</P>
  <P><INPUT type="checkbox" name="c5" value="排球">排球</P>
  <P><INPUT type="checkbox" name="c6" value="蓝球">蓝球</P>
  <P><INPUT type="submit" name="b3" value="提交"><INPUT
type="reset" name="b4" value="取消"></P>
   <A href="buy1.jsp">买点别的</A> <A href="display.jsp">查看购物车
</A></FORM>
```

程序名称：**display.jsp**

```
<%  String str="";
if(session.getAttribute("s1")!=null){
str=(String)session.getAttribute("s1");
out.println(str+"<br>");  }
if(session.getAttribute("s2")!=null){
str=(String)session.getAttribute("s2");
 out.println(str+"<br>");  }
if(session.getAttribute("s3")!=null){
str=(String)session.getAttribute("s3");
 out.println(str+"<br>"); }
if(session.getAttribute("s4")!=null){
str=(String)session.getAttribute("s4");
 out.println(str+"<br>"); }
if(session.getAttribute("s5")!=null){
str=(String)session.getAttribute("s5");
 out.println(str+"<br>"); }
if(session.getAttribute("s6")!=null){
str=(String)session.getAttribute("s6");
out.println(str+"<br>");}%>
```

7 PageContext对象

PageContext对象的作用是取得任何范围的参数，通过PageContext对象可以获取JSP页面的out、request、response、session、application等对象，重新定向客户的请求等。

8 config对象

config对象是在一个Servlet初始化时，JSP引擎向它传递信息用的，此信息包括Servlet初始化时所要用到的参数及服务器的有关信息。

9 page对象

page对象指代JSP页面本身、代表了正在运行的由JSP文件产生的类对象。

10 exception对象

exception对象的作用是显示异常信息，它是java.lang.Throwable的一个实例，只有在包含isErrorPage="true"的页面中才可以被使用，在一般的JSP内容中使用该对象将无法编译JSP文件。

在JSP页面编写过程中常会出现如下几种错误，读者应该注意。

空指针错误：java.lang.NullPointerException格式化数字错误：java.lang.NumberFormatException类定义未找到错误：java.lang.NoClassDefFoundError

JAVA错误：java.lang.Error

11.cookie的使用

- Cookie对象是由Web服务器端产生保存到浏览器中的信息。Cookie对象可以用来在浏览器中保存一些信息。
- 创建cookie的步骤：
- 1.创建cookie

格式：Cookie 对象名=new Cookie(“变量名”，值);

- 2.写入Cookie中

格式：response.addCookie(“对象名”);

- 3.读取Cookie

Cookie[] 数据变量名=request.getCookies();

例程序4-27.jsp和4-28.jsp

第五章： JSP内部对象

案例名称：写入cookie
程序名称：4-27.jsp

```
<%String strName = "Zhourunfa";
   Cookie c = new Cookie("Name1", strName);
   response.addCookie(c);%>
写入Cookie<br><br>
<a href="4-28.jsp">查看</a>
```

案例名称：读出cookie
程序名称：4-28.jsp

```
<%@ page contentType="text/html;charset=GBK" %>
<HTML><BODY>
<%Cookie cookies[] = request.getCookies();
   for(int i=0; i<cookies.length; i++) {
       if(cookies[i].getName().equals("Name1"))
      out.print(cookies[i].getValue());   }%>
读出Cookie<br><br>
```

12 错误解决方法

HTTP常见错误代码

401：验证错误tomcat登录验证

404：访问资源不存在

405：servlet错误

500：语法编译错误

12 错误解决方法

HTTP常见错误代码

错误信息定制：编写显示错误的页面和 设置web.xml

error404.html和error500.html

```
<error-page>
<error-code>404</error-code>
<location>/error404.html</location>
</error-page>
<error-page>
<error-code>500</error-code>
<location>/error500.html</location>
</error-page>
```

本章小结

主要介绍了jsp常用的对象，掌握out、response、request、session、cookie对象。

重点掌握response、request、session和application对象的基本应用。

Web语言程序设计

下　课！！