JSP程序设计教程

第4 章　JSP内置对象

第 4 章 JSP内置对象

- 4.1 JSP内置对象概述 √
- 4.2 request对象 √
- 4.3 response对象 √
- 4.4 session对象 √
- 4.5 application对象 √
- 4.6 out对象 √
- 4.7 其他内置对象 √

4.1 JSP内置对象概述

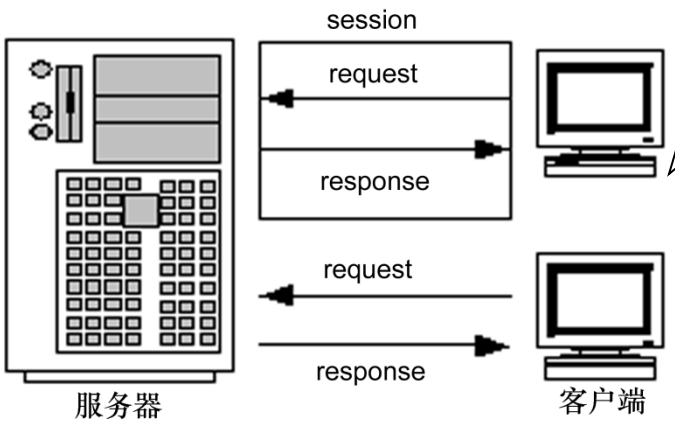
为了Web应用程序开发的方便，在JSP页面中内置了一些默认的对象，这些对象不需要预先声明就可以在脚本代码和表达式中随意使用。JSP提供的内置对象共有9个，如下表所示。

4.1 JSP内置对象概述

内置对象名称	所 属 类 型	有效范围	说　明
application	javax.servlet.ServletContext	application	该对象代表应用程序上下文，它允许JSP页面与包括在同一应用程序中的任何Web组件共享信息
config	javax.servlet.ServletConfig	page	该对象允许将初始化数据传递给一个JSP页面
exception	java.lang.Throwable	page	该对象含有只能由指定的JSP“错误处理页面”访问的异常数据
out	javax.servlet.jsp.JspWriter	page	该对象提供对输出流的访问
page	javax.servlet.jsp.HttpJspPage	page	该对象代表JSP页面对应的Servlet类实例
pageContext	javax.servlet.jsp.PageContext	page	该对象是JSP页面本身的上下文，它提供了唯一一组方法来管理具有不同作用域的属性，这些API在实现JSP自定义标签处理程序时非常有用
request	javax.servlet.http.HttpServletRequest	request	该对象提供对HTTP请求数据的访问，同时还提供用于加入特定请求数据的上下文
response	javax.servlet.http.HttpServletResponse	page	该对象允许直接访问HttpServletReponse对象，可用来向客户端输入数据
session	javax.servlet.http.HttpSession	session	该对象可用来保存在服务器与一个客户端之间需要保存的数据，当客户端关闭网站的所有网页时，session变量会自动消失

4.1 JSP内置对象概述

request、response和session是JSP内置对象中重要的3个对象，这3个对象体现了服务器端与客户端（即浏览器）进行交互通信的控制，如下图所示。



从该图可以看出，当客户端打开浏览器，在地址栏中输入服务器Web服务页面的地址后，就会显示Web服务器上的网页。客户端的浏览器从Web服务器上获得网页，实际上是使用HTTP协议向服务器端发送了一个请求，服务器在收到来自客户端浏览器发来的请求后要响应请求。JSP通过request对象获取客户浏览器的请求，通过response对客户浏览器进行响应。而session则一直保存着会话期间所需要传递的数据信息。

4.2 request对象

request对象是从客户端向服务器发出请求，包括用户提交的信息以及客户端的一些信息。客户端可通过HTML表单或在网页地址后面提供参数的方法提交数据，然后通过request对象的相关方法来获取这些数据。request的各种方法主要用来处理客户端浏览器提交的请求中的各项参数和选项。

4.2 request对象

4.2.1 访问请求参数 √

4.2.2 在作用域中管理属性 √

4.2.3 获取Cookie √

4.2.4 获取客户信息 √

4.2.5 访问安全信息 √

4.2.6 访问国际化信息 √

4.2.1 访问请求参数

在Web应用程序中，经常还需要完成用户与网站的交互。例如，当用户填写表单后，需要把数据提交给服务器处理，服务器获取到这些信息并进行处理。request对象的getParameter()方法，可以用来获取用户提交的数据。

访问请求参数的方法如下：

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

参数name与HTML标记name属性对应，如果参数值不存在，则返回一个null值，该方法的返回值为String类型。

【例4-1】 访问请求参数示例

4.2.2 在作用域中管理属性

有时，在进行请求转发时，需要把一些数据带到转发后的页面进行处理。这时，就可以使用request对象的setAttribute()方法设置数据在request范围内存取。

设置转发数据的方法使用如下：

```
request.setAttribute("key", Object);
```

参数key是键，为String类型。在转发后的页面取数据时，就通过这个键来获取数据。参数object是键值，为Object类型，它代表需要保存在request范围内的数据。

获取转发数据的方法如下：

```
request.getAttribute(String name);
```

4.2.2 在作用域中管理属性

参数name表示键名。

在页面使用request对象的setAttribute("name",obj)方法，可以把数据obj设定在request范围内。请求转发后的页面使用使用“getAttribute("name");”就可以取得数据obj。

【例4-2】 在作用域中管理属性示例

使用request对象的setAttribute()方法设置数据，然后在请求转发后取得设置的数据。

4.2.3 获取Cookie

Cookie为Web应用程序保存用户相关信息提供了一种有用的方法。Cookie是一小段文本信息，伴随着用户请求和页面在Web服务器和浏览器之间传递。

用户每次访问站点时，Web应用程序都可以读取Cookie包含的信息。例如，当用户访问站点时，可以利用Cookie保存用户首选项或其他信息，这样当用户下次再访问站点时，应用程序就可以检索以前保存的信息。

在JSP中，可以通过request对象中的getCookies()方法获取Cookie中的数据。获取Cookie的方法如下：

```
Cookie[] cookie = request.getCookies();
```

4.2.3 获取Cookie

request对象的getCookies()方法，返回的是Cookie[]数组。

【例4-3】 获取Cookie示例

使用request对象的addCookie()方法实现记录本次及上一次访问网页的时间。

4.2.4 获取客户信息

request对象提供了一些用来获取客户信息的方法，如下表所示。

方　法	说　明
getHeader(String name)	获得Http协议定义的文件头信息
getHeaders(String name)	返回指定名字的request Header的所有值，其结果是一个枚举的实例
getHeadersNames()	返回所有request Header的名字，其结果是一个枚举的实例
getMethod()	获得客户端向服务器端传送数据的方法，如get，post，header，trace等
getProtocol()	获得客户端向服务器端传送数据所依据的协议名称
getRequestURI()	获得发出请求字符串的客户端地址
getRealPath()	返回当前请求文件的绝对路径
getRemoteAddr()	获取客户端的IP地址
getRemoteHost()	获取客户端的机器名称
getServerName()	获取服务器的名字
getServerPath()	获取客户端所请求的脚本文件的文件路径
getServerPort()	获取服务器的端口号

4.2.4 获取客户信息

【例4-4】 获取客户信息示例

使用request对象的相关方法获取客户信息，

4.2.5 访问安全信息

request对象提供了对安全属性的访问，如下表所示。

方　法	说　明
isSecure()	返回布尔类型的值，它用于确定这个请求是否使用了一个安全协议，例如HTTP
isRequestedSessionIdFromCookie()	返回布尔类型的值，表示会话是否使用了一个Cookie来管理会话ID
isRequestedSessionIdFromURL()	返回布尔类型的值，表示会话是否使用URL重写来管理会话ID
isRequestedSessionIdFromValid()	检查请求的会话ID是否合法

例如，可以通过使用request对象来确定当前请求是否使用了一个类似HTTP的安全协议：

用户安全信息：<%=request.isSecure()%>

4.2.6 访问国际化信息

浏览器可以通过accept-language的HTTP报头向Web服务器指明它所使用的本地语言。request对象中的getLocale()和getLocales()方法允许JSP开发人员获取这一信息，获取的信息属于java.util.Local类型。java.util.Local类型的对象封装了一个国家和一种国家所使用的语言。使用这些信息，JSP开发者就可以使用语言所特有的信息作出响应。使用这个报头的代码如下：

4.2.6 访问国际化信息

```
<%
java.util.Locale locale=request.getLocale();
if(locale.equals(java.util.Locale.US)){
out.print("Welcome to BeiJing");
}
if(locale.equals(java.util.Locale.CHINA)){
out.print("北京欢迎您");
}
%>
```

这段代码，如果所在区域为中国，将显示“北京欢迎您”，而所在区域为英国，则显示“Welcome to BeiJing”。

4.3 response对象

response对象和request对象相对应，用于响应客户请求，向客户端输出信息。response对象是javax.servlet.http.HttpServletResponse接口类的对象，它封装了JSP产生的响应，并发送到客户端以响应客户端的请求。请求的数据可以是各种数据类型，甚至是文件。

4.3 response对象

4.3.1 重定向网页 √

4.3.2 设置HTTP响应报头 √

4.3.3 缓冲区配置 √

4.3.1 重定向网页

在JSP页面中，可以使用response对象中的sendRedirect()方法将客户请求重定向到一个不同的页面。例如，将客户请求转发到login_ok.jsp页面的代码如下：

```
response.sendRedirect("login_ok.jsp");
```

在JSP页面中，还可以使用response对象中的sendError()方法指明一个错误状态。该方法接收一个错误以及一条可选的错误消息，该消息将在内容主体上返回给客户。例如，代码“response.sendError(500,"请求页面存在错误")”将客户请求重定向到一个在内容主体上包含了出错消息的出错页面。

4.3.1 重定向网页

上述两个方法都会中止当前的请求和响应。如果HTTP响应已经提交给客户，则不会调用这些方法。
response对象中用于重定向网页的方法如下表所示。

方　法	说　明
sendError(int number)	使用指定的状态码向客户发送错误响应
sendError(int number,String msg)	使用指定的状态码和描述性消息向客户发送错误响应
sendRedirect(String location)	使用指定的重定向位置URL想客户发送重定向响应，可以使用相对URL

【例4-5】 重定向网页示例

使用request对象的相关方法重定向网页。

4.3.2 设置HTTP响应报头

response对象提供了设置HTTP响应报头的方法，如下表所示。

方　法	说　明
setDateHeader(String name,long date)	使用给定的名称和日期值设置一个响应报头，如果指定的名称已经设置，则新值会覆盖旧值
setHeader(String name,String value)	使用给定的名称和值设置一个响应报头，如果指定的名称已经设置，则新值会覆盖旧值
setHeader(String name,int value)	使用给定的名称和整数值设置一个响应报头，如果指定的名称已经设置，则新值会覆盖旧值
addHeader(String name,long date)	使用给定的名称和值设置一个响应报头
addDateHeader(String name,long date)	使用给定的名称和日期值设置一个响应报头
containHeader(String name)	返回一个布尔值，它表示是否设置了已命名的响应报头
addIntHeader(String name,int value)	使用给定的名称和整数值设置一个响应报头
setContentType(String type)	为响应设置内容类型，其参数值可以为text/html，text/plain，application/x_msexcel或application/msword
setContentLength(int len)	为响应设置内容长度
setLocale(java.util.Locale loc)	为响应设置地区信息

4.3.2 设置HTTP响应报头

技巧：通过设置HTTP头可实现禁用缓存功能，具体代码如下：

```
<%response.setHeader("Cache-Control","no-store");
response.setDateHeader("Expires",0);%>
```

需要注意的是，上面的代码必须在没有任何输出发送到客户端之前使用。

【例4-6】 设置HTTP响应报头示例

将JSP页面保存为word文档。

4.3.3 缓冲区配置

缓冲可以更加有效地在服务器与客户之间传输内容。HttpServletResponse对象为支持jspWriter对象而启用了缓冲区配置。response对象提供了配置缓冲区的方法，如下表所示。

方　法	说　明
flushBuffer()	强制把缓冲区中内容发送给客户
getBufferSize()	返回响应所使用的实际缓冲区大小，如果没使用缓冲区，则该方法返回0
setBufferSize(int size)	为响应的主体设置首选的缓冲区大小
isCommitted()	返回一个boolean，表示响应是否已经提交；提交的响应已经写入状态码和报头
reset()	清除缓冲区存在的任何数据，同时清除状态码和报头

4.3.3 缓冲区配置

【例4-7】 缓冲区配置示例

输出缓冲区的大小并测试强制将缓冲区的内容发送给客户。

4.4 session对象

HTTP协议是一种无状态协议。也就是说，当一个客户向服务器发出请求，服务器接收请求，并返回响应后，该连接就被关闭了，此时服务器端不保留连接的有关信息，因此当下一次连接时，服务器已没有了以前的连接信息，此时将不能判断这一次连接和以前的连接是否属于同一客户。为了弥补这一缺点，JSP提供了一个session对象，这样服务器和客户端之间的连接就会一直保持下去，但是在一定时间内（系统默认在30min内），如果客户端不向服务器发出应答请求，session对象就会自动消失。不过在编写程序时，可以修改这个时间限定值，使session对象在特定时间内保存信息。保存的信息可以是与客户端有关的，也可以是一般信息，这可以根据需要设定相应的内容。

4.4 session对象

4.4.1 创建及获取客户的会话√

4.4.2 从会话中移除指定的对象 √

4.4.3 销毁session √

4.4.4 会话超时的管理 √

4.4.1 创建及获取客户的会话

JSP页面可以将任何对象作为属性来保存。session内置对象使用setAttribute()和getAttribute()方法创建及获取客户的会话。

setAttribute()方法用于是设置指定名称的属性值，并将其存储在session对象中，其语法格式如下：

```
session.setAttribute(String name,String value);
```

参数name为属性名称，value为属性值。

getAttribute()方法用于是获取与指定名字name相联系的属性，其语法格式如下：

```
session.getAttribute(String name);
```

4.4.1 创建及获取客户的会话

参数name为属性名称。

【例4-8】 创建及获取客户会话示例

通过setAttribute()方法将数据保存在session中，并通过getAttribute()方法取得数据的值。

4.4.2 从会话中移除指定的对象

JSP页面可以将任何已经保存的对象进行移除。session内置对象使用removeAttribute()方法将所指定名称的对象移除，也就是说，从这个会话删除与指定名称绑定的对象。removeAttribute()方法的语法格式如下：

```
session.removeAttribute (String name);
```

参数name为session对象的属性名，代表要移除的对象名。

【例4-9】 从会话中移除指定对象示例

通过setAttribute()方法将数据保存在session中，然后通过removeAttribute()方法移除指定对象。

4.4.3 销毁session

JSP页面可以将已经保存的所有对象全部删除。session内置对象使用invalidate()方法将会话中的全部内容删除。invalidate()方法的语法格式如下：

```
session.invalidate();
```

4.4.4 会话超时的管理

在一个Servlet程序或JSP文件中，确保客户会话终止的唯一方法使用超时设置。这是因为Web客户在进入非活动状态时不以显示的方式通知服务器。为了清除存储在session对象中的客户申请资源，Servlet程序容器设置一个超时窗口。当非活动的时间超出了窗口的大小时，JSP容器将使session对象无效并撤销所有属性的绑定，从而管理会话的生命周期。

4.4.4 会话超时的管理

session对象用于管理会话生命周期的方法如下表所示。

方　法	说　明
getLastAccessedTime()	返回客户端最后一次发送与这个会话相关联的请求时间
getMaxInactiveInterval()	以秒为单位返回一个会话内两个请求的最大时间间隔，Servlet容器在客户访问期间保存这个会话处于打开状态
setMaxInactiveInterval(int interval)	以秒为单位指定在服务器小程序容器使该会话无效之前的客户请求之间的最长时间，也就是超时时间

4.5 application对象

application对象用于保存所有应用程序中的公有数据，服务器启动并且自动创建application对象后，只要没有关闭服务器，application对象将一直存在，所有用户可以共享application对象。application对象与session对象有所区别，session对象和用户会话相关，不同用户的session是完全不同的对象，而用户的application对象都是相同的一个对象，即共享这个内置的application对象。

4.5 application对象

4.5.1 访问应用程序初始化参数 √

4.5.2 管理应用程序环境属性 √

4.5.1 访问应用程序初始化参数

通过application对象调用的ServletContext对象提供了对应用程序环境属性的访问。对于将安装信息与给定的应用程序关联起来而言，这是非常有用的。例如，通过初始化信息为数据库提供了一个主机名，每一个Servlet程序客户和JSP页面都可以使用它连接到该数据库并检索应用程序数据。为了实现这个目的，Tomcat使用了web.xml文件，它位于应用程序环境目录下的WEB-INF子目录中。

4.5.1 访问应用程序初始化参数

访问应用程序初始化参数的方法如下表所示。

方　法	说　明
getInitParameter(String name)	返回一个已命名的初始化参数的值
getInitParameterNames()	返回所有已定义的应用程序初始化参数名称的枚举

【例4-10】 访问应用程序初始化参数示例

通过application对象调用web.xml文件的初始化参数。

4.5.2 管理应用程序环境属性

与session对象相同，也可以在application对象中设置属性。在session中设置的属性只是在当前客户的会话范围内容有效，客户超过保存时间不发送请求时，session对象将被回收，而在application对象中设置的属性在整个应用程序范围内是有效的，即使所有的用户都不发送请求，只要不关闭应用服务器，在其中设置的属性仍然是有效的。

4.5.2 管理应用程序环境属性

application对象管理应用程序环境属性的方法如下表所示。

方　法	说　明
removeAttribute(String name)	从ServletContext的对象中去掉指定名称的属性
setAttribute(String name,Object object)	使用指定名称和指定对象在ServletContext的对象中进行关联
getAttribute(String name)	从ServletContext的对象中获取一个指定对象
getAttributeNames()	返回存储在ServletContext对象中属性名称的枚举数据

【例4-11】 管理应用程序环境属性示例

通过application对象中的setAttribute()和getAttribute()方法实现网页计数器。

4.6 out对象

out对象主要用来向客户端输出各种数据类型的内容，并且管理应用服务器上的输出缓冲区，缓冲区默认值一般是8KB，可以通过页面指令page来改变默认值。在使用out对象输出数据时，可以对数据缓冲区进行操作，及时清除缓冲区中的残余数据，为其他的输出让出缓冲空间。待数据输出完毕后，要及时关闭输出流。out对象被封装为javax.servlet.jsp.JspWriter类的对象，在实际上应用上out对象会通过JSP容器变换为java.io.PrintWriter类的对象。

4.6 out对象

4.6.1 管理响应缓冲 √

4.6.2 向客户端输出数据 √

4.6.1 管理响应缓冲

在JSP页面中，可以通过out对象调用clear()方法清除缓冲区的内容。这类似于重置响应流，以便重新开始操作。如果响应已经提交，则会有产生IOException异常的副作用。相反，另一个种方法clearBuffer()清除缓冲区的“当前”内容，而且即使内容已经提交给客户端，也能够访问该方法。out对象用于管理响应缓冲区的方法如下表所示。

4.6.1 管理响应缓冲

out对象用于管理响应缓冲区的方法如下表所示。

方　法	说　明
clear()	清空缓冲区
clearBuffer()	清空当前区的内容
close()	先刷新流，然后关闭流
flush()	刷新流
getBufferSize()	以字节为单位返回缓冲区的大小
getRemaining()	返回缓冲区中没有使用的字符的数量
isAutoFlush()	返回布尔值，自动刷新还是在缓冲区溢出时抛出IOException异常

4.6.2 向客户端输出数据

out对象的另外一个很重要的功能就是向客户写入内容。由于JspWriter是由java.io.Writer派生而来，因此它的使用与java.io.Writer很相似。例如在JSP页面中输出一句话，代码如下：

```
<%=out.println("同一世界，同一梦想")%>
```

这句代码用于在页面中输出“同一世界，同一梦想”。

4.7 其他内置对象

在JSP内置对象中，pageContext，config，page及exception这些对象是不经常使用的，下面将对这些对象分别进行介绍。

4.7.1 获取会话范围的pageContext对象 √

4.7.2 读取web.xml配置信息的config对象 √

4.7.3 应答或请求的page对象 √

4.7.4 获取异常信息的exception对象 √

4.7.1 获取会话范围的pageContext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，使用它可以访问到本页中所有其他对象。pageContext对象被封装成javax.servlet.jsp.pageContext接口，主要用于管理对属于JSP中特殊可见部分中已经命名对象的访问，它的创建和初始化都是由容器来完成的，JSP页面里可以直接使用pageContext对象的句柄，pageContext对象的getXxx()、setXxx()和findXxx()方法可以用来根据不同的对象范围实现对这些对象的管理。

4.7.1 获取会话范围的pageContext对象

pageContext对象的常用方法如下表所示。

方　法	说　明
forward(java.lang.String relativeUtlpath)	把页面转发到另一个页面或者servlet组件上
getAttribute(java.lang.String name[,int scope])	scope参数是可选的，该方法用来检索一个特定的已经命名的对象的范围，并且还可以通过调用getAttributeNameInScope()方法，检索对某个特定范围的每个属性String字符串名称枚举
getException()	返回当前的Exception对象
getRequest()	返回当前的request对象
getResponse()	返回当前的response对象
getServletConfig()	返回当前页面的ServletConfig对象
invalidate()	返回servletContext对象，全部销毁
setAttribute()	设置默认页面范围或特定对象范围之中的已命名对象
removeAttribute()	删除默认页面范围或特定对象范围之中的已命名对象

4.7.1 获取会话范围的pageContext对象

说明：pageContext对象在实际JSP开发过程中很少使用，因为request和response等对象可以直接调用方法进行使用，如果通过pageContext来调用其他对象有些麻烦。

4.7.2 读取web.xml配置信息的config对象

config对象被封装成javax.servlet.ServletConfig接口，它表示Servlet的配置，当一个Servlet初始化时，容器把某些信息通过此对象传递给这个Servlet。开发者可以在web.xml文件中为应用程序环境中的Servlet程序和JSP页面提供初始化参数。

config对象的常用方法如下表所示。

方　法	说　明
getServletContext()	返回执行者的Servlet上下文
getServletName()	返回Servlet的名字
getInitParameter()	返回名字为name的初始参数的值
getInitParameterNames()	返回这个JSP的所有的初始参数的名字

4.7.3 应答或请求的page对象

page对象是为了执行当前页面应答请求而设置的Servlet类的实体，即显示JSP页面自身，只有在JSP页面内才是合法的。page隐含对象本质上包含当前Servlet接口引用的变量，可以看作是this变量的别名，因此该对象对于开发JSP比较有用。

page对象比较常用的方法如下表所示。

方　法	说　明
getClass()	返回当前Object的类
hashCode()	返回此Object的哈希代码
toString()	将此Object类转换成字符串
equals(Object o)	比较此对象和指定的对象是否相等
copy(Object o)	把此对象赋值到指定的对象当中去
clone()	对此对象进行克隆

4.7.4 获取异常信息的exception对象

exception内置对象用来处理JSP文件执行时发生的所有错误和异常。exception对象和Java的所有对象一样，都具有系统的继承结构，exception对象几乎定义了所有异常情况，这样的exception对象和我们常见的错误有所不同。所谓错误，指的是可以预见的，并且知道如何解决的情况，一般在编译时可以发现。

与错误不同，异常是指在程序执行过程中不可预料的情况，由潜在的错误机率导致，如果不对异常进行处理，程序会崩溃。在Java中，利用 “try/catch”关键字来处理异常情况，如果在JSP页面中出现没有捕捉到的异常，就会生成exception对象，并把这个exception对象传送到在page指令中设定的错误页面中，然后在错误提示页面中处理相应的exception对象。exception对象只有在错误页面（在页面指令里有isErrorPage=true的页面）才可以使用。

4.7.4 获取异常信息的exception对象

exception对象比较常用的方法如下表所示。

方　法	说　明
getMessage()	该方法返回异常消息字符串
getLocalizedMessage()	该方法返回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信息描述
fillInStackTrace()	重写异常错误的栈执行轨迹

【例4-12】 获取异常信息的exception对象示例

通过exception异常对象将系统出现的异常转向到其他页面。