认识jsp的9个内置对象，4中属性范围：

**内置对象**

我们在之前的操作中，是不是经常会使用到request.getParameter()来接收表单传递的参数，在java中要使用一个对象，必须实例化这个对象，但是在jsp中request可以直接使用，类似于这样的对象，我们称之为jsp的内置对象，内置对象总共有九个。

|  |  |  |
| --- | --- | --- |
| NO | 内置对象 | 类型 |
| **1** | pageContext | javax.servlet.jsp.PageContext |
| **2** | request | javax.servlet.http.HttpServletRequest |
| **3** | response | javax.servlet.http.HttpServletResponse |
| **4** | session | javax.servlet.http.HttpSession |
| **5** | application | javax.servlet.ServletContext |
| **6** | config | javax.servlet.ServletConfig |
| 7 | out | javax.servlet.jsp.jspWriter |
| 8 | page | java.lang.Object |
| 9 | exception | java.lang.Throwable |

**属性范围：**

所谓四种属性范围，就是指在一个页面设置了属性之后，可以再其他页面是否仍然可以访问此属性，四种属性范围分别指的是以下四种：

·属性范围只在当前页：一个属性只能在设置属性的页面取得，跳转到其页面无法访问

·属性范围在一次服务器的请求范围内：在一个页面设置属性之后，只要经过了服务器端跳转，无论跳转多少次，只要是服务端跳转，跳转后的页面仍然可以访问属性

·一次会话：一个用户（浏览器）设置属性，无论经过服务端还是客户端跳转，都可以访问属性

·上下文中：指的是服务器的上下文中，就是说属性是在服务器上设置的，所有人都可以访问

属性的操作方法：

提供的四种属性范围都有如下的属性操作方法：

|  |  |  |
| --- | --- | --- |
| NO | 方法 | 描述 |
| 1 | public void setAttribute(String name,Object value) | 设置属性 |
| 2 | public Object getAttribute(String name) | 取得属性 |
| 3 | public void removeAttribute(String name) | 删除属性 |

第一种属性范围：page属性范围：（pageContext）

page 的属性范围相比后面三种属性范围相当的好理解，就是指一个页面设置了属性之后，本页面可以访问，其他页面无法取得属性。

要注意的一点：我们所说的page属性范围，指的并不是page，而是页面上下文（pageContext）

pageContext的属性操作流程如下图：

服务端跳转

取得属性

设置属性

其他页面，无法取得

本页面

下面通过代码观察：

范例1：在本页中取得设置的属性：

|  |
| --- |
| <%@page contentType=*"text/html; charset=UTF-8"*%>  <%@page import=*"java.util.\*"* %>  <%  //此时通过pageContext设置两个属性  pageContext.setAttribute("name","WANCZY");  pageContext.setAttribute("date",**new** Date());  %>  <%  //取得设置的属性  String name = (String)pageContext.getAttribute("name");  Date date = (Date)pageContext.getAttribute("date");  %>  <h1><%=name %></h1>  <h1><%=date %></h1> |

在本页中确实可以取得设置的page Context的范围属性，下面我们使用服务端跳转，观察跳转之后是否还可以取得属性

范例：经过服务端跳转：

|  |
| --- |
| <%@page contentType=*"text/html; charset=UTF-8"*%>  <%@page import=*"java.util.\*"* %>  <%  //此时通过pageContext设置两个属性  pageContext.setAttribute("name","WANCZY");  pageContext.setAttribute("date",**new** Date());  %>  <jsp:forward page=*"pageContext03.jsp"*></jsp:forward> |
| <%@page contentType=*"text/html; charset=UTF-8"*%>  <%@page import=*"java.util.\*"* %>  <%  //取得设置的属性  String name = (String)pageContext.getAttribute("name");  Date date = (Date)pageContext.getAttribute("date");  %>  <h1><%=name %></h1>  <h1><%=date %></h1> |

我们发现经过服务端跳转之后，无法再通过pavaContext取得设置的属性，现在继续观察客户端跳转是否可以取得设置的属性：

范例3：经过客户端跳转，观察属性是否可以取得

|  |
| --- |
| <%@page contentType=*"text/html; charset=UTF-8"*%>  <%@page import=*"java.util.\*"* %>  <%  //此时通过pageContext设置两个属性  pageContext.setAttribute("name","WANCZY");  pageContext.setAttribute("date",**new** Date());  %>  <a href=*"pageContext05.jsp"*>客户端跳转</a> |
| <%@page contentType=*"text/html; charset=UTF-8"*%>  <%@page import=*"java.util.\*"* %>  <%  //取得设置的属性  String name = (String)pageContext.getAttribute("name");  Date date = (Date)pageContext.getAttribute("date");  %>  <h1><%=name %></h1>  <h1><%=date %></h1> |

通过观察我们发现，经过客户端跳转，也是不能取得pageContext设置的属性

如果现在希望通过服务器跳转到其他页面中，也可以取得属性，则可以扩大属性的范围，使用request属性范围就可以

request 属性范围：

request属性范围就表示在服务端跳转中生效，只要是服务端跳转，则设置的request属性就可以一直访问

request属性范围流程：

取得属性

取得属性

页面（可以取得属性）

页面（可以取得属性）

服务端跳转

服务端跳转

设置属性

页面

还是通过代码来观察request的属性范围：

范例1：本页面设置和在本页面取得属性

|  |
| --- |
| <%@page contentType=*"text/html; charset=UTF-8"*%>  <%@page import=*"java.util.\*"* %>  <%  //此时通过pageContext设置两个属性  request.setAttribute("name","WANCZY");  request.setAttribute("date",**new** Date());  %>  <%  //取得设置的属性  String name = (String)request.getAttribute("name");  Date date = (Date)request.getAttribute("date");  %>  <h1><%=name %></h1>  <h1><%=date %></h1> |

通过效果我们可以知道，request在本页面设置属性之后是可以在本页面取得的

范例2：设置属性后通过服务端跳转后再取得属性

|  |
| --- |
| <%@page contentType=*"text/html; charset=UTF-8"*%>  <%@page import=*"java.util.\*"* %>  <%  //此时通过pageContext设置两个属性  request.setAttribute("name","WANCZY");  request.setAttribute("date",**new** Date());  %>  <jsp:forward page=*"request02.jsp"*></jsp:forward> |
| <%@page contentType=*"text/html; charset=UTF-8"*%>  <%@page import=*"java.util.\*"* %>  <%  //取得设置的属性  String name = (String)request.getAttribute("name");  Date date = (Date)request.getAttribute("date");  %>  <h1><%=name %></h1>  <h1><%=date %></h1> |

通过观察效果可以发现，request属性范围和pageContext的属性范围不同，request属性范围可以通过服务端跳转仍然可以取得属性

范例3：客户端跳转，查看是否能够取得属性

|  |
| --- |
| <%@page contentType=*"text/html; charset=UTF-8"*%>  <%@page import=*"java.util.\*"* %>  <%  //此时通过pageContext设置两个属性  request.setAttribute("name","WANCZY");  request.setAttribute("date",**new** Date());  %>  <a href=*"request05.jsp"*>客户端跳转</a> |
| <%@page contentType=*"text/html; charset=UTF-8"*%>  <%@page import=*"java.util.\*"* %>  <%  //取得设置的属性  String name = (String)request.getAttribute("name");  Date date = (Date)request.getAttribute("date");  %>  <h1><%=name %></h1>  <h1><%=date %></h1> |

观察效果，发现request属性范围在通过客户端跳转后不能取得设置的属性。

注意一点，request属性范围不管经过多少次服务端跳转都可以取得设置的request属性。

session属性：

只要设置的属性，不管如何跳转，只要同一个浏览器中，都是可以访问属性的。前提：session只针对一个用户

session属性范围的流程：

跳转（服务端、客户端）

取得属性

页面（可以取得属性）

页面

设置属性

还是通过代码来观察：

范例1：服务器端跳转

|  |
| --- |
| <%@page contentType=*"text/html; charset=UTF-8"*%>  <%@page import=*"java.util.\*"* %>  <%  //此时通过session设置两个属性  session.setAttribute("name","WANCZY");  session.setAttribute("date",**new** Date());  %>  <jsp:forward page=*"session02.jsp"*></jsp:forward> |
| <%@page contentType=*"text/html; charset=UTF-8"*%>  <%@page import=*"java.util.\*"* %>  <%  //取得设置的属性  String name = (String)session.getAttribute("name");  Date date = (Date)session.getAttribute("date");  %>  <h1><%=name %></h1>  <h1><%=date %></h1> |

观察结构发现，服务端跳转和request属性一样，同样可以取得；

范例2：通过客户端跳转

|  |
| --- |
| <%@page contentType=*"text/html; charset=UTF-8"*%>  <%@page import=*"java.util.\*"* %>  <%  //此时通过session设置两个属性  session.setAttribute("name","WANCZY");  session.setAttribute("date",**new** Date());  %>  <a href=*"session04.jsp"*>客户端跳转</a> |
| <%@page contentType=*"text/html; charset=UTF-8"*%>  <%@page import=*"java.util.\*"* %>  <%  //取得设置的属性  String name = (String)session.getAttribute("name");  Date date = (Date)session.getAttribute("date");  %>  <h1><%=name %></h1>  <h1><%=date %></h1> |

观察结果可以发现，客户端跳转同样可以取得设置的属性，这一点可request属性范围不同

但是从新开一个浏览器的话，直接访问跳转后的页面都不能取得属性。如果设置的属性想让所有的用户都可以访问，则可以使用最后一种属性范围：application范围

application属性范围：

application属性范围是指在服务器上设置一个属性，一旦设置，所用用户都可以访问了吧？

流程：

服务器

其他页面都可以取得

设置属性

页面

下面我们来观察效果：

范例：

|  |
| --- |
| <%@page contentType=*"text/html; charset=UTF-8"*%>  <%@page import=*"java.util.\*"* %>  <%  //取得设置的属性  String name = (String)application.getAttribute("name");  Date date = (Date)application.getAttribute("date");  %>  <h1><%=name %></h1>  <h1><%=date %></h1> |
| <%@page contentType=*"text/html; charset=UTF-8"*%>  <%@page import=*"java.util.\*"* %>  <%  //此时通过session设置两个属性  application.setAttribute("name","WANCZY");  application.setAttribute("date",**new** Date());  %>  <%  //取得设置的属性  String name = (String)application.getAttribute("name");  Date date = (Date)application.getAttribute("date");  %>  <h1><%=name %></h1>  <h1><%=date %></h1> |

通过以上的效果可以发现，即使浏览器关闭，同样可以取得设置的属性。但是如果服务器已关闭，在开启属性肯定消失

关于pageContext属性范围的进步一补充：

之前我们讲解的四种属性范围，实际上都是通过pageContext属性范围一起设置上的。

在 pageContext中，setAttribute()方法被重载了多次，

public void

此setAttribute(String name,Object value,int scope)方法在设置属性的时候加上了一个int的变量，此变量就表示设置的这个属性的属性范围

|  |
| --- |
| <%@page contentType=*"text/html; charset=UTF-8"*%>  <%@page import=*"java.util.\*"* %>  <%  //此时通过pageContext设置两个属性  pageContext.setAttribute("name","WANCZY",PageContext.REQUEST\_SCOPE);  pageContext.setAttribute("date",**new** Date(),PageContext.REQUEST\_SCOPE);  %>  <jsp:forward page=*"pageContext06.jsp"*></jsp:forward> |
| <%@page contentType=*"text/html; charset=UTF-8"*%>  <%@page import=*"java.util.\*"* %>  <%  //取得设置的属性  String name = (String)request.getAttribute("name");  Date date = (Date)request.getAttribute("date");  %>  <h1><%=name %></h1>  <h1><%=date %></h1> |

总结：

知道九个内置对象

充分掌握四种属性范围