JSP 指令 JSP 隐式对象

# JSP 动作元素

与JSP指令元素不同的是，JSP动作元素在请求处理阶段起作用。JSP动作元素是用XML语法写成的。

利用JSP动作可以动态地插入文件、重用JavaBean组件、把用户重定向到另外的页面、为Java插件生成HTML代码。

动作元素只有一种语法，它符合XML标准：

```
<jsp:action_name attribute="value" />
```

动作元素基本上都是预定义的函数，JSP规范定义了一系列的标准动作，它用JSP作为前缀，可用的标准动作元素如下：

| 语法 | 描述 |
| --- | --- |
| jsp:include | 在页面被请求的时候引入一个文件。 |
| jsp:useBean | 寻找或者实例化一个JavaBean。 |
| jsp:setProperty | 设置JavaBean的属性。 |
| jsp:getProperty | 输出某个JavaBean的属性。 |
| jsp:forward | 把请求转到一个新的页面。 |
| jsp:plugin | 根据浏览器类型为Java插件生成OBJECT或EMBED标记。 |
| jsp:element | 定义动态XML元素 |
| jsp:attribute | 设置动态定义的XML元素属性。 |
| jsp:body | 设置动态定义的XML元素内容。 |
| jsp:text | 在JSP页面和文档中使用写入文本的模板 |

## 常见的属性

所有的动作要素都有两个属性：id属性和scope属性。

- **id属性：**
  id属性是动作元素的唯一标识，可以在JSP页面中引用。动作元素创建的id值可以通过PageContext来调用。
- **scope属性：**
  该属性用于识别动作元素的生命周期。 id属性和scope属性有直接关系，scope属性定义了相关联id对象的寿命。 scope属性有四个可能的值： (a) page, (b)request, (c)session, 和 (d) application。

# <jsp:include>动作元素

<jsp:include>动作元素用来包含静态和动态的文件。该动作把指定文件插入正在生成的页面。语法格式如下：

```
<jsp:include page="相对 URL 地址" flush="true" />
```

前面已经介绍过include指令，它是在JSP文件被转换成Servlet的时候引入文件，而这里的jsp:include动作不同，插入文件的时间是在页面被请求的时候。

以下是include动作相关的属性列表。

| 属性 | 描述 |
| --- | --- |
| page | 包含在页面中的相对URL地址。 |
| flush | 布尔属性，定义在包含资源前是否刷新缓存区。 |

## 实例

以下我们定义了两个文件 **date.jsp** 和 **main.jsp**，代码如下所示：

date.jsp文件代码：

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<p>
   今天的日期是: <%= (new java.util.Date()).toLocaleString()%>
</p>
```

main.jsp文件代码：

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>菜鸟教程(runoob.com)</title>
</head>
<body>

<h2>include 动作实例</h2>
<jsp:include page="date.jsp" flush="true" />

</body>
</html>
```

现在将以上两个文件放在服务器的根目录下，访问main.jsp文件。显示结果如下：

```
include 动作实例

今天的日期是: 2016-6-25 14:08:17
```

# <jsp:useBean>动作元素

**jsp:useBean** 动作用来加载一个将在JSP页面中使用的JavaBean。

这个功能非常有用，因为它使得我们可以发挥 Java 组件复用的优势。

jsp:useBean动作最简单的语法为：

```
<jsp:useBean id="name" class="package.class" />
```

在类载入后，我们既可以通过 jsp:setProperty 和 jsp:getProperty 动作来修改和检索bean的属性。

以下是useBean动作相关的属性列表。

| 属性 | 描述 |
| --- | --- |
| class | 指定Bean的完整包名。 |
| type | 指定将引用该对象变量的类型。 |
| beanName | 通过 java.beans.Beans 的 instantiate() 方法指定Bean的名字。 |

在给出具体实例前，让我们先来看下 jsp:setProperty 和 jsp:getProperty 动作元素：

# <jsp:setProperty>动作元素

jsp:setProperty用来设置已经实例化的Bean对象的属性，有两种用法。首先，你可以在jsp:useBean元素的外面（后面）使用jsp:setProperty，如下所示：

```
<jsp:useBean id="myName" ... />
...
<jsp:setProperty name="myName" property="someProperty" .../>
```

此时，不管jsp:useBean是找到了一个现有的Bean，还是新创建了一个Bean实例，jsp:setProperty都会执行。第二种用法是把jsp:setProperty放入jsp:useBean元素的内部，如下所示：

```
<jsp:useBean id="myName" ... >
...
   <jsp:setProperty name="myName" property="someProperty" .../>
</jsp:useBean>
```

此时，jsp:setProperty只有在新建Bean实例时才会执行，如果是使用现有实例则不执行jsp:setProperty。

jsp:setProperty动作有下面四个属性,如下表：

| 属性 | 描述 |
| --- | --- |
| name | name属性是必需的。它表示要设置属性的是哪个Bean。 |
| property | property属性是必需的。它表示要设置哪个属性。有一个特殊用法：如果property的值是"*"，表示所有名字和Bean属性名字匹配的请求参数都将被传递给相应的属性set方法。 |
| value | value 属性是可选的。该属性用来指定Bean属性的值。字符串数据会在目标类中通过标准的valueOf方法自动转换成数字、boolean、Boolean、 byte、Byte、char、Character。例如，boolean和Boolean类型的属性值（比如"true"）通过Boolean.valueOf转换，int和Integer类型的属性值（比如"42"）通过Integer.valueOf转换。 value和param不能同时使用，但可以使用其中任意一个。 |
| param | param 是可选的。它指定用哪个请求参数作为Bean属性的值。如果当前请求没有参数，则什么事情也不做，系统不会把null传递给Bean属性的set方法。因此，你可以让Bean自己提供默认属性值，只有当请求参数明确指定了新值时才修改默认属性值。 |

# <jsp:getProperty>动作元素

jsp:getProperty动作提取指定Bean属性的值，转换成字符串，然后输出。语法格式如下：

```
<jsp:useBean id="myName" ... />
...
<jsp:getProperty name="myName" property="someProperty" .../>
```

下表是与getProperty相关联的属性：

| 属性 | 描述 |
| --- | --- |
| name | 要检索的Bean属性名称。Bean必须已定义。 |
| property | 表示要提取Bean属性的值 |

## 实例

以下实例我们使用了Bean:

```
package com.runoob.main;

public class TestBean {
   private String message = "菜鸟教程";

   public String getMessage() {
      return(message);
   }
```

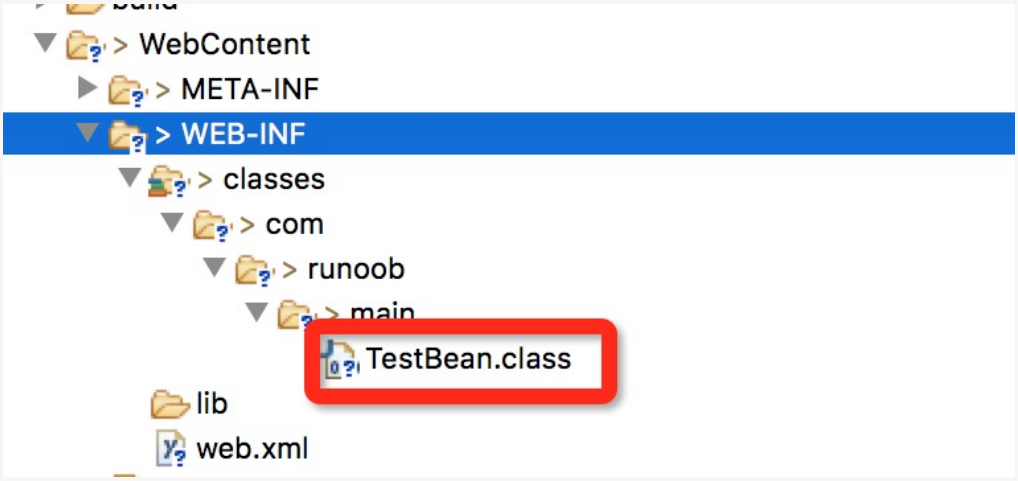
```
    public void setMessage(String message) {
       this.message = message;
    }
}
```

编译以上实例文件 TestBean.java ：

```
$ javac TestBean.java
```

编译完成后会在当前目录下生成一个 **TestBean.class** 文件， 将该文件拷贝至当前 JSP 项目的 **WebContent/WEB-INF/classes/com/runoob/main** 下( com/runoob/main 包路径，没有需要手动创建)。

下面是一个 Eclipse 中目录结构图：




下面是一个很简单的例子，它的功能是装载一个Bean，然后设置/读取它的message属性。

现在让我们在main.jsp文件中调用该Bean:

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>菜鸟教程(runoob.com)</title>
</head>
<body>

<h2>Jsp 使用 JavaBean 实例</h2>
<jsp:useBean id="test" class="com.runoob.main.TestBean" />

<jsp:setProperty name="test"
                    property="message"
                    value="菜鸟教程..." />

<p>输出信息....</p>
```

```
<jsp:getProperty name="test" property="message" />

</body>
</html>
```

浏览器访问，执行以上文件，输出如下所示：


---

# <jsp:forward> 动作元素

jsp:forward动作把请求转到另外的页面。jsp:forward标记只有一个属性page。语法格式如下所示：

```
<jsp:forward page="相对 URL 地址" />
```

以下是forward相关联的属性：

| 属性 | 描述 |
| --- | --- |
| page | page属性包含的是一个相对URL。page的值既可以直接给出，也可以在请求的时候动态计算，可以是一个JSP页面或者一个 Java Servlet. |

## 实例

以下实例我们使用了两个文件，分别是： date.jsp 和 main.jsp。

date.jsp 文件代码如下：

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<p>
   今天的日期是: <%= (new java.util.Date()).toLocaleString()%>
</p>
```

main.jsp文件代码：

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
```

```
<meta charset="utf-8">
<title>菜鸟教程(runoob.com)</title>
</head>
<body>

<h2>forward 动作实例</h2>
<jsp:forward page="date.jsp" />
</body>
</html>
```

现在将以上两个文件放在服务器的根目录下，访问main.jsp文件。显示结果如下：

```
今天的日期是: 2016-6-25 14:37:25
```

---

# <jsp:plugin>动作元素

jsp:plugin动作用来根据浏览器的类型，插入通过Java插件 运行Java Applet所必需的OBJECT或EMBED元素。

如果需要的插件不存在，它会下载插件，然后执行Java组件。 Java组件可以是一个applet或一个JavaBean。

plugin动作有多个对应HTML元素的属性用于格式化Java 组件。param元素可用于向Applet 或 Bean 传递参数。

以下是使用plugin 动作元素的典型实例:

```
<jsp:plugin type="applet" codebase="dirname" code="MyApplet.class"
                           width="60" height="80">
   <jsp:param name="fontcolor" value="red" />
   <jsp:param name="background" value="black" />

   <jsp:fallback>
      Unable to initialize Java Plugin
   </jsp:fallback>

</jsp:plugin>
```

如果你有兴趣可以尝试使用applet来测试jsp:plugin动作元素，<fallback>元素是一个新元素，在组件出现故障的错误时发送给用户错误信息。

---

# <jsp:element> 、 <jsp:attribute>、 <jsp:body>动作元素

<jsp:element> 、 <jsp:attribute>、 <jsp:body>动作元素动态定义XML元素。动态是非常重要的，这就意味着XML元素在编译时是动态生成的而非静态。

以下实例动态定义了XML元素：

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
```

```
<html>
<head>
<meta charset="utf-8">
<title>菜鸟教程(runoob.com)</title>
</head>
<body>
<jsp:element name="xmlElement">
<jsp:attribute name="xmlElementAttr">
   属性值
</jsp:attribute>
<jsp:body>
   XML 元素的主体
</jsp:body>
</jsp:element>
</body>
</html>
```

浏览器访问以下页面，输出结果如下所示：

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>菜鸟教程(runoob.com)</title>
</head>
<body>
<xmlElement xmlElementAttr="属性值">
   XML 元素的主体
</xmlElement>
</body>
</html>
```

---

# <jsp:text>动作元素

<jsp:text>动作元素允许在JSP页面和文档中使用写入文本的模板，语法格式如下：

```
<jsp:text>模板数据</jsp:text>
```

以上文本模板不能包含重复元素，只能只能包含文本和EL表达式（注：EL表达式将在后续章节中介绍）。请注意，在XML文件中，您不能使用表达式如 ${whatever > 0}，因为>符号是非法的。 你可以使用 ${whatever gt 0}表达式或者嵌入在一个CDATA部分的值。

```
<jsp:text><![CDATA[<br>]]></jsp:text>
```

如果你需要在 XHTML 中声明 DOCTYPE,必须使用到<jsp:text>动作元素，实例如下：

```
<jsp:text><![CDATA[<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"DTD/xhtml1-strict.dtd">]]>
</jsp:text>
<head><title>jsp:text action</title></head>
<body>

<books><book><jsp:text>
    Welcome to JSP Programming
</jsp:text></book></books>

</body>
</html>
```

你可以对以上实例尝试使用<jsp:text>及不使用该动作元素执行结果的区别。

← JSP 指令 JSP 隐式对象 →