JSP - ACTIONS

JSP actions use constructs in XML syntax to control the behavior of the servlet engine. You can dynamically insert a file, reuse JavaBeans components, forward the user to another page, or generate HTML for the Java plugin.

There is only one syntax for the Action element, as it conforms to the XML standard:

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

Action elements are basically predefined functions and there are following JSP actions available:

Syntax	Purpose
jsp:include	Includes a file at the time the page is requested
jsp:useBean	Finds or instantiates a JavaBean
jsp:setProperty	Sets the property of a JavaBean
jsp:getProperty	Inserts the property of a JavaBean into the output
jsp:forward	Forwards the requester to a new page
jsp:plugin	Generates browser-specific code that makes an OBJECT or EMBED tag for the Java plugin
jsp:element	Defines XML elements dynamically.
jsp:attribute	Defines dynamically defined XML element's attribute.
jsp:body	Defines dynamically defined XML element's body.
jsp:text	Use to write template text in JSP pages and documents.

Common Attributes:

There are two attributes that are common to all Action elements: the **id** attribute and the **scope** attribute.

- **Id attribute:** The id attribute uniquely identifies the Action element, and allows the action to be referenced inside the JSP page. If the Action creates an instance of an object the id value can be used to reference it through the implicit object PageContext

- **Scope attribute:** This attribute identifies the lifecycle of the Action element. The id attribute and the scope attribute are directly related, as the scope attribute determines the lifespan of the object associated with the id. The scope attribute has four possible values: a page, b request, $c$session, and d application.

The <jsp:include> Action

This action lets you insert files into the page being generated. The syntax looks like this:

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

Unlike the **include** directive, which inserts the file at the time the JSP page is translated into a servlet, this action inserts the file at the time the page is requested.

Following is the list of attributes associated with include action:

Attribute	Description
page	The relative URL of the page to be included.
flush	The boolean attribute determines whether the included resource has its buffer flushed before it is included.

Example:

Let us define following two files *a*date.jps and *b* main.jsp as follows:

Following is the content of date.jsp file:

```
<p>
   Today's date: <%= (new java.util.Date()).toLocaleString()%>
</p>
```

Here is the content of main.jsp file:

```
<html>
<head>
<title>The include Action Example</title>
</head>
<body>
<center>
<h2>The include action Example</h2>
<jsp:include page="date.jsp" flush="true" />
</center>
</body>
</html>
```

Now let us keep all these files in root directory and try to access main.jsp. This would display result something like this:

```
The include action Example

   Today's date: 12-Sep-2010 14:54:22
```

The <jsp:useBean> Action

The **useBean** action is quite versatile. It first searches for an existing object utilizing the id and scope variables. If an object is not found, it then tries to create the specified object.

The simplest way to load a bean is as follows:

```
<jsp:useBean   />
```

Once a bean class is loaded, you can use **jsp:setProperty** and **jsp:getProperty** actions to modify and retrieve bean properties.

Following is the list of attributes associated with useBean action:

Attribute	Description
class	Designates the full package name of the bean.
type	Specifies the type of the variable that will refer to the object.

beanName	Gives the name of the bean as specified by the instantiate method of the java.beans.Beans class.

Let us discuss about **jsp:setProperty** and **jsp:getProperty** actions before giving a valid example related to these actions.

The <jsp:setProperty> Action

The **setProperty** action sets the properties of a Bean. The Bean must have been previously defined before this action. There are two basic ways to use the setProperty action:

You can use jsp:setProperty after, but outside of, a jsp:useBean element, as below:

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

In this case, the jsp:setProperty is executed regardless of whether a new bean was instantiated or an existing bean was found.

A second context in which jsp:setProperty can appear is inside the body of a jsp:useBean element, as below:

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

Here, the jsp:setProperty is executed only if a new object was instantiated, not if an existing one was found.

Following is the list of attributes associated with setProperty action:

Attribute	**Description**
name	Designates the bean whose property will be set. The Bean must have been previously defined.
property	Indicates the property you want to set. A value of "*" means that all request parameters whose names match bean property names will be passed to the appropriate setter methods.
value	The value that is to be assigned to the given property. The the parameter's value is null, or the parameter does not exist, the setProperty action is ignored.
param	The param attribute is the name of the request parameter whose value the property is to receive. You can't use both value and param, but it is permissible to use neither.

The <jsp:getProperty> Action

The **getProperty** action is used to retrieve the value of a given property and converts it to a string, and finally inserts it into the output.

The getProperty action has only two attributes, both of which are required ans simple syntax is as follows:

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

Following is the list of required attributes associated with setProperty action:

Attribute	Description
name	The name of the Bean that has a property to be retrieved. The Bean must have been previously defined.
property	The property attribute is the name of the Bean property to be retrieved.

Example:

Let us define a test bean which we will use in our example:

```
/* File: TestBean.java */
package action;

public class TestBean {
   private String message = "No message specified";

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

Compile above code to generated TestBean.class file and make sure that you copied TestBean.class in C:\apache-tomcat-7.0.2\webapps\WEB-INF\classes\action folder and CLASSPATH variable should also be set to this folder:

Now use the following code in main.jsp file which loads the bean and sets/gets a simple String parameter:

```
<html>
<head>
<title>Using JavaBeans in JSP</title>
</head>
<body>
<center>
<h2>Using JavaBeans in JSP</h2>

<jsp:useBean   />

<jsp:setProperty name="test"
                    property="message"
                    value="Hello JSP..." />

<p>Got message....</p>

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

</center>
</body>
</html>
```

Now try to access main.jsp, it would display following result:

```
Using JavaBeans in JSP
```

```
Got message....

Hello JSP...
```

The <jsp:forward> Action

The **forward** action terminates the action of the current page and forwards the request to another resource such as a static page, another JSP page, or a Java Servlet.

The simple syntax of this action is as follows:

```
<jsp:forward page="Relative URL" />
```

Following is the list of required attributes associated with forward action:

Attribute	Description
page	Should consist of a relative URL of another resource such as a static page, another JSP page, or a Java Servlet.

Example:

Let us reuse following two files *a* date.jps and *b* main.jsp as follows:

Following is the content of date.jsp file:

```
<p>
   Today's date: <%= (new java.util.Date()).toLocaleString()%>
</p>
```

Here is the content of main.jsp file:

```
<html>
<head>
<title>The include Action Example</title>
</head>
<body>
<center>
<h2>The include action Example</h2>
<jsp:forward page="date.jsp" />
</center>
</body>
</html>
```

Now let us keep all these files in root directory and try to access main.jsp. This would display result something like as below. Here it discarded content from main page and displayed content from forwarded page only.

```

   Today's date: 12-Sep-2010 14:54:22

```

The <jsp:plugin> Action

The **plugin** action is used to insert Java components into a JSP page. It determines the type of browser and inserts the <object> or <embed> tags as needed.

If the needed plugin is not present, it downloads the plugin and then executes the Java component. The Java component can be either an Applet or a JavaBean.

The plugin action has several attributes that correspond to common HTML tags used to format

Java components. The <param> element can also be used to send parameters to the Applet or Bean.

Following is the typical syntax of using plugin action:

```
<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>
```

You can try this action using some applet if you are interested. A new element, the <fallback> element, can be used to specify an error string to be sent to the user in case the component fails.

The <jsp:element> Action

The <jsp:attribute> Action

The <jsp:body> Action

The <jsp:element>, lt;jsp:attribute> and <jsp:body> actions are used to define XML elements dynamically. The word dynamically is important, because it means that the XML elements can be generated at request time rather than statically at compile time.

Following is a simple example to define XML elements dynamically:

```
<%@page language="java" contentType="text/html"%>
<html xmlns="http://www.w3c.org/1999/xhtml"
      xmlns:jsp="http://java.sun.com/JSP/Page">

<head><title>Generate XML Element</title></head>
<body>
<jsp:element name="xmlElement">
<jsp:attribute name="xmlElementAttr">
   Value for the attribute
</jsp:attribute>
<jsp:body>
   Body for XML element
</jsp:body>
</jsp:element>
</body>
</html>
```

This would produce following HTML code at run time:

```
<html xmlns="http://www.w3c.org/1999/xhtml"
      xmlns:jsp="http://java.sun.com/JSP/Page">

<head><title>Generate XML Element</title></head>
<body>
<xmlElement xmlElementAttr="Value for the attribute">
   Body for XML element
</xmlElement>
</body>
</html>
```

The <jsp:text> Action

The <jsp:text> action can be used to write template text in JSP pages and documents. Following is the simple syntax for this action:

```
<jsp:text>Template data</jsp:text>
```

The body fo the template cannot contain other elements; it can only contain text and EL expressions $Note: ELexpressionsareexplainedinsubsequentchapter$. Note that in XML files, you cannot use expressions such as $whatever > 0, becausethegreaterthansignsareillegal. Instead, usethegtform, suchas${whatever gt 0} or an alternative is to embed the value in a CDATA section.

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

If you need to include a DOCTYPE declaration, for instance for XHTML, you must also use the <jsp:text> element as follows:

```
<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>
```

Try above example with and without <jsp:text> action.