ICON OF JAVA

ADV. JAVA
JDBC

9.Transaction Management

Mr. Nagoor Babu M.Tech

[ICON of JAVA]
(Sun certified & Realtime Expert)

Ex. HCL Employee

Trained Lakhs of Students
for last 14 years across INDIA

India's No.1 Software Training Institute

DURGASOFT

www.durgasoft.com Ph: 9246212143 ,8096969696

JSP Actions

In Jsp technology, by using scripting elements we are able to provide java code inside the Jsp pages, but the main theme of Jsp technology is not to allow java code inside Jsp pages.

To eliminate java code from Jsp pages we have to eliminate scripting elements, to eliminate scripting elements from Jsp pages we have to provide an alternative i.e. Jsp Actions.

In case of Jsp Actions, we will define a scripting tag in Jsp page and we will provide a block of java code w.r.t. scripting tag.

When container encounters the scripting tag then container will execute respective java code, by this an action will be performed called as Jsp Action.

In Jsp technology, there are 2 types of actions.

1. Standard Actions
2. Custom Actions

DURGASOFT Means JAVA
JAVA Means DURGASOFT
India's No.1 JAVA Trainer
More Than 1000 Students in a Single Class Room.
Weapon of DURGASOFT
Mr.Nagoor Babu M.Tech
Trained Lakhs of Students for Last 13years (Realtime Expert & Java Certified)

1. Standard Actions:

Standard Actions are Jsp Actions, which could be defined by the Jsp technology to perform a particular action.

Jsp technology has provided all the standard actions in the form of a set of predefined tags called Action Tags.

1. <jsp:useBean---->
2. <jsp:setProperty---->
3. <jsp:getProperty---->
4. <jsp:include---->
5. <jsp:forward---->
6. <jsp:param---->
7. <jsp:plugin---->
8. <jsp:fallback---->
9. <jsp:params---->
10. <jsp:declaration---->
11. <jsp:scriptlet---->
12. <jsp:expression---->

Java Beans:

Java Bean is a reusable component.

Java Bean is a normal java class which may declare properties, setter and getter methods in order to represent a particular user form at server side.

If we want to prepare Java Bean components then we have to use the following rules and regulations.

1. Java Bean is a normal java class, it is suggestible to implement Serializable interface.
2. Always Java Bean classes should be public, non-abstract and non-final.
3. In Java Bean classes, we have to declare all theproperties w.r.t. the properties define in the respective user form.
4. In Java Bean classes, all the properties should be private.
5. In Java Bean classes, all the behaviours should be public.
6. If we want to declare any constructor in Java Bean class then that constructor should bepublic and zero argument.

Ex:
```
public class Employee implements Serializable {
private String eno;
private String ename;
private float esal;
public void setEno(String eno) {
                this.eno=eno;
           }
public void setEname(String ename) {
this.ename=ename;
             }
public void setEsal(String esal) {
this.esal=esal;
             }
public String getEno() {
return eno;
             }
public String getEname() {
return ename;
             }
public float getEsal() {
return esal;
             }
         }
```

1. <jsp:useBean>:

The main purpose of <jsp:useBean> tag is to interact with bean object from a particular Jsp page.

Syntax:<jsp:useBean id=”--” class=”--” type=”--” scope=”--”/>

Where id attribute will take a variable to manage generated Bean object reference.

Where class attribute will take the fully qualified name of Bean class.

Where type attribute will take the fully qualified name of Bean class to define the type of variable in order to manage Bean object reference.

Where scope attribute will take either of the Jsp scopes to Bean object.

Note: In <jsp:useBean> tag, always it is suggestible to provide either application or session scope to the scope attribute value.

Ex: <jsp:useBean id=”e” class=”Employee” type=”Employee” scope=”session”/>

When container encounters the above tag then container will pick up class attribute value i.e. fully qualified name of Bean class then container will recognize Bean class .class file and perform Bean class loading and instantiation.

After creating Bean object container will assign Bean object reference to the variable specified as value to id attribute.

After getting Bean object reference container will store Bean object in a scope specified as value to scope attribute.

2. <jsp:setProperty>:

The main purpose of <jsp:setProperty> tag is to execute a particular setter method in order to set a value to a particular Bean property.

Syntax:<jsp:setProperty name="--" property="--" value="--"/>

Where name attribute will take a variable which is same as id attribute value in <jsp:useBean> tag.

Where property attribute will take a property name in order to access the respective setter method.

Where value attribute will take a value to pass as a parameter to the respective setter method.

3. <jsp:getProperty>:

The main purpose of <jsp:getProperty> tag is to execute a getter method in order to get a value from Bean object.

Syntax:<jsp:getProperty name="--" property="--"/>

Where name attribute will take a variable which is same as id attribute value in <jsp:useBean> tag.

Where property attribute will take a particular property to execute the respective getter method.

--------------Application2--------------

usebeanapp:

empform.html:

```
<html>
	<body bgcolor="lightblue"><br><br><br><br>
	<center><h1>Employee Details Form</h1></center>
	<form method="get" action="display.jsp">
	<pre><h2>
		Employee Id :<input type="text" name="eid"/>
		Employee Name : <input type="text" name="ename"/>
		Employee Salary : <input type="text" name="esal"/>
		<input type="submit" value="Display"/>
	</h2></pre>
	</form>
	</body>
</html>
```

Employee.java:

```
package comm.dss;
publicclass Employee implements java.io.Serializable {
	privateint eid;
	private String ename;
	privatefloat esal;
	publicint getEid() {
		return eid;
	}
	publicvoid setEid(int eid) {
		this.eid = eid;
	}
	public String getEname() {
		return ename;
	}
	publicvoid setEname(String ename) {
		this.ename = ename;
	}
	publicfloat getEsal() {
		return esal;
	}
	publicvoid setEsal(float esal) {
		this.esal = esal;
	}
}
```

display.jsp:

```
<%!
        int eid;
        String ename;
        float esal;
%>
<%
        try {
                eid=Integer.parseInt(request.getParameter("eid"));
                ename=request.getParameter("ename");
                esal=Float.parseFloat(request.getParameter("esal"));
        }
        catch(Exception e){
                e.printStackTrace();
        }
 %>
<jsp:useBean id="e" class="com.dss.EmployeeBean" type="com.dss.EmployeeBean"
scope="session">
<jsp:setProperty name="e" property="eid" value='<%=eid %>'/>
<jsp:setProperty name="e" property="ename" value='<%=ename %>'/>
<jsp:setProperty name="e" property="esal" value='<%=esal %>'/>
<html>
        <body>
                <center><h1>Employee Details</h1></center>
                <center>
                        Employee Id : <jsp:getProperty name="e"
property="eid"/><br><br>
                        Employee Name : <jsp:getProperty name="e"
property="ename"/><br><br>
                        Employee Salary : <jsp:getProperty name="e"
property="esal"/><br><br>
                </center>
        </body>
</html>
</jsp:useBean>
```

Note: In case of <jsp:useBean> tag, in general we will provide a separate <jsp:setProperty> tag to set a particular value to the respective property in Bean object.

In case of <jsp:useBean> tag, it is possible to copy all the request parameter values directly onto the respective Bean object.

To achieve this we have to provide "*" as value to property attribute in <jsp:setProperty> tag.

Ex: <jsp:setProperty name="e" property="*"/>

If we want to achieve the above requirement then we have to maintain same names in the request parameters i.e. form properties and Bean properties.

Note: The above "*" notation is not possible with <jsp:getProperty> tag.

DURGASOFT Means JAVA
JAVA Means DURGASOFT
India's No.1 JAVA Trainer
More Than 1000 Students in a Single Class Room
Weapon of DURGASOFT
Mr.Nagoor Babu M.Tech
Trained Lakhs of Students for Last 13years (Realtime Expert & Java Certified)

4. <jsp:include>:

Q: What are the differences between include directive and <jsp:include> action tag?

Ans: 1. In Jsp applications,**include directive** can be used to include the content of the target resource into the present Jsp page.

In Jsp pages, **<jsp:include>** action tag can be used to include the target resource response into the present Jsp page response.

2. In general include directive can be used to include static resources where the frequent updations are not available.

<jsp:include> action tag can be used to include dynamic resources where the frequent updations are available.

3. In general directives will be resolved at the time of translation and actions will be resolved at the time of request processing. Due to this reason include directive will be resolved at the time of translation but <jsp:include> action tag will be resolved at the time of request processing.

If we are trying to include a target Jsp page into present Jsp page by using include directive then container will prepare only one translated servlet.

To include a target Jsp page into the present Jsp page if we use <jsp:include> action tag then container will prepare 2 separate translated servlets.

In Jsp applications, include directives will provide static inclusion, but <jsp:include> action tag will provide dynamic inclusion.

In Jsp technology, <jsp:include> action tag was designed on the basis of Include Request Dispatching Mechanism.

Syntax:<jsp:include page="--" flush="--"/>

Where page attribute will take the name and location of the target resource to include its response.

Where flush is a boolean attribute, it can be used to give an intimation to the container about to autoFlush or not to autoFlush dynamic response to client when JspWriter buffer filled with the response at the time of including the target resource response into the present Jsp page.

--------------Application3---------------

includeapp:

addform.html:

```
<html>
	<body bgcolor="lightgreen">
	<form action="add.jsp">
		<pre>
			<u>Product Details</u>
			Product Id : <input type="text" name="pid"/>
			Product Name : <input type="text" name="pname"/>
			Product Cost : <input type="text" name="pcost"/>
			<input type="submit" value="ADD"/>
		</pre>
	</form>
	</body>
</html>
```

add.jsp:

```
<%@page import="java.sql.*"%>
<%!
	String pid;
	String pname;
	int pcost;
	static Connection con;
	static Statement st;
	ResultSet rs;
	ResultSetMetaData rsmd;
	static{
		try{
			Class.forName("oracle.jdbc.driver.OracleDriver");
	con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "system",
"durga");
			st=con.createStatement();
		}
		catch(Exception e){
			e.printStackTrace();
		}
	}
%>
<%
	try{
		pid=request.getParameter("pid");
		pname=request.getParameter("pname");
		pcost=Integer.parseInt(request.getParameter("pcost"));
		st.executeUpdate("insert into product
values('"+pid+"','"+pname+"',"+pcost+")");
		rs=st.executeQuery("select * from product");
		rsmd=rs.getMetaData();
```

```
                int count=rsmd.getColumnCount();
%>
        <html><body><center>
        <table border="1" bgcolor="lightyellow">
        <tr>
<%
        for (int i=1;i<=count;i++){
%>
        <td><b><font size="6" color="red">
        <center><%=rsmd.getColumnName(i) %></center>
        </font></b></td>
<%
        }
%>
        </tr>
<%
        while(rs.next()){
%>
        <tr>
<%
        for(int i=1;i<=count;i++){
%>
        <td><b><font size="6">
<%=rs.getString(i) %>
        </font></b></td>
<%
        }
%>
        </tr>
<%
        }
%>
        </table></center></body></html>
<%
        }
        catch(Exception e){
                e.printStackTrace();
        }
%>
        <hr>
        <jsp:include page="addform.html" flush="true"/>
```

5. <jsp:forward>:

Q: What are the differences between<jsp:include> action tag and <jsp:forward> action tag?

Ans: 1. **<jsp:include>** action tag can be used to include the target resource response into the present Jsp page.

<jsp:forward> tag can be used to forward request from present Jsp page to the target resource.

2. <jsp:include> tag was designed on the basis of Include Request Dispatching Mechanism.

<jsp:forward> tag was designed on the basis of Forward Request Dispatching Mechanism.

3. When Jsp container encounter <jsp:include> tag then container will forward request to the target resource, by executing the target resource some response will be generated in the response object, at the end of the target resource container will bypass request and response objects back to first resource, at the end of first resource execution container will dispatch overall response to client. Therefore, in case of <jsp:include> tag client is able to receive all the resources response which are participated in the present request processing.

When container encounters<jsp:forward> tag then container will bypass request and response objects to the target resource by refreshing response object i.e. by eliminating previous response available in response object, at the end of target resource container will dispatch the generated dynamic response directly to the client without moving back to first resource. Therefore, in case of <jsp:forward> tag client is able to receive only target resource response.

Syntax:<jsp:forward page="--"/>

Where page attribute specifies the name and location of the target resource.

6. <jsp:param>:

This action tag can be used to provide a name value pair to the request object at the time of by passing request object from present Jsp page to target page either in include mechanism or in forward mechanism or in both.

This tag must be utilized as a child tag to <jsp:include> tag and <jsp:forward> tags.

Syntax:<jsp:param name=”--” value=”--”/>

--------------Application4---------------

forwardapp:

registrationform.html:

```
<html>
        <body bgcolor="lightgreen">
        <form action="registration.jsp">
                <pre>
                        <u>Registration Form</u>
                        Name : <input type="text" name="pname"/>
                        Age : <input type="text" name="uage"/>
                        Address : <input type="text" name="uaddr"/>
                        <input type="submit" value="Registration"/>
                </pre>
        </form>
        </body>
</html>
```

existed.jsp:

```
<center><h1>User Existed</h1></center>
```

success.jsp:

```
<center><h1>Registration Success</h1></center>
```

failure.jsp:

```
<center><h1>Registration Failure</h1></center>
```

registration.jsp:

```
<%@page import="java.sql.*"%>
<%!
        String uname;
        int uage;
        String uaddr;
        static Connection con;
        static Statement st;
        ResultSet rs;
        static{
                try{
                        Class.forName("oracle.jdbc.driver.OracleDriver");
                con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe",
"system", "durga");
                        st=con.createStatement();
                }
                catch(Exception e){
                        e.printStackTrace();
                }
        }
%>
<%
        try{
                uname=request.getParameter("uname");
                uage=Integer.parseInt(request.getParameter("uage"));
                uaddr=request.getParameter("uaddr");
                rs=st.executeQuery("select * from reg_users where uname='"+uname+"'");
                boolean b=rs.next();
                if(b==true)
                {
%>
        <jsp:forward page="existed.jsp"/>
<%
                }
                else
                {
                        int rowCount=st.executeUpdate("insert into reg_users values
('"+uname+"',"+uage+",'"+uaddr+"')");
                        if(rowCount == 1)
                        {
%>
        <jsp:forward page="success.jsp"/>
<%
                        }
                        else
                        {
%>
        <jsp:forward page="failure.jsp"/>
<%
                        }
                }
        }
        catch(Exception e){
%>
        <jsp:forward page="failure.jsp"/>
<%
                e.printStackTrace();
        }
%>
```

7. <jsp:plugin>:

This tag can be used to include an applet into the present Jsp page.

Syntax:<jsp:plugin code="--" width="--" height="--" type="--"/>

Where code attribute will take fully qualified name of the applet.

Where width and height attributes can be used to specify the size of applet.

Where type attribute can be used to specify which one we are going to include whether it is applet or bean.

Ex:<jsp:plugin code="Logo" width="1000" height="150" type="applet"/>

8. <jsp:params>:

In case of the applet applications, we are able to provide some parameters to the applet in order to provide input data.

Similarly if we want to provide input parameters to the applet from <jsp:plugin> tag we have to use <jsp:param> tag.
<jsp:param> tag must be utilized as a child tag to <jsp:params> tag.
<jsp:params> tag must be utilized as a child tag to <jsp:plugin> tag.

Syntax:

```
<jsp:plugin>
<jsp:params>
<jsp:param name="--" value="--"/>
<jsp:param name="--" value="--"/>
           ----------
</jsp:params>
     ----------
</jsp:plugin>
```

If we provide any input parameter to the applet then that parameter value we are able to get by using the following method from Applet class.
public String getParameter(String name)

Ex: String msg=getParameter("message");

--------------Application5---------------

pluginapp:

LogoApplet.java:

```
import java.awt.*;
import java.applet.*;
public class LogoApplet extends Applet
{
                String msg;
                public void paint(Graphics g)
                {
                msg=getParameter("message");
                Font f=new Font("arial",Font.BOLD,40);
                g.setFont(f);
                this.setBackground(Color.blue);
                this.setForeground(Color.white);
                g.drawString(msg,150,70);
                }
}
```

logo.jsp:

```
<jsp:plugin code="LogoApplet" width="1000" height="150" type="applet">
                <jsp:params>
                <jsp:param name="message" value="durga software solutions"/>
                </jsp:params>
</jsp:plugin>
```

DURGASOFT Means JAVA
JAVA Means DURGASOFT
India's No.1 JAVA Trainer
More Than 1000 Students in a Single Class Room.
Weapon of DURGASOFT
Mr.Nagoor Babu M.Tech
Trained Lakhs of Students for Last 13years (Realtime Expert & Java Certified)

9. <jsp:fallback>:

The main purpose of <jsp:fallback> tag is to display an alternative message when client browser is not supporting <OBJECT---> tag and <EMBED---> tag.

Syntax:<jsp:fallback>-------Description---------</jsp:fallback>

In Jsp applications, we have to utilize <jsp:fallback> tag as a child tag to <jsp:plugin> tag.

Ex:<jsp:plugin code=”LogoApplet” width=”1000” height=”150” type=”applet”>

<jsp:fallback>Applet Not Allowed</jsp:fallback>

</jsp:plugin>

DURGASOFT Means JAVA
JAVA Means DURGASOFT
India's No.1 JAVA Trainer
More Than 1000 Students in a Single Class Room.
Weapon of DURGASOFT
Mr.Nagoor Babu M.Tech
Trained Lakhs of Students for Last 13years (Realtime Expert & Java Certified)

10. <jsp:declaration>:

This tag is almost all same as the declaration scripting element, it can be used to provide all the Java declarations in the present Jsp page.

Syntax:<jsp:declaration>

```
            --------
            --------   }  Java Declarations
            --------
</jsp:declaration>
```

11. <jsp:scriptlet>:

This tag is almost all same as thescripting element scriptlets, it can be used to provide a block of Java code in Jsp pages.

Syntax:<jsp:scriptlet>

```
--------
            --------   }  Block of Java code
--------
</jsp:scriptlet>
```

12. <jsp:expression>:

This tag is almost all same as the scripting element expression, it can be used to provide a Java expression in the present Jsp page.

Syntax:<jsp:expression> Java Expression </jsp:expression>

Ex:

```
<%@page import="java.util.*"%>
<jsp:declaration>
Date d=null;
String date=null;
</jsp:declaration>
<jsp:scriptlet>
d=new Date();
date=d.toString();
</jsp:scriptlet>
<html>
<body bgcolor="lightyellow">
<center><b><font size="6" color="red"><br><br>
Today Date : <jsp:expression>date</jsp:expression>
</font></b></center></body>
</html>
```