Chapter

15

SECTION III: JAVA SERVER PAGES

Action Elements Of Java Server Pages

Actions are specific tags that:

- Affect the runtime behavior of the JSP
- Affect the response sent back to the client
- Allow the transfer of control between pages

Actions use constructs in XML syntax. These constructs control the behavior of the Servlet engine. They are typically used to:

- Dynamically insert a file
- Reuse Java Bean components
- Forward the user to another page
- Access parameters sent with the request to do a database lookup

- Generate HTML such as a table populated with information retrieved from an external system

An action tag looks like a regular HTML tag and does not follow the <% ... %> syntax conventions.

The standard action types are:

Including Other Files

This action [<jsp:include> element] allows a static or dynamic resource to be included in the current JSP at request time. In other words, it allows inserting files [a Servlet or another JSP page] into the page being generated.

The included page can therefore be a static document, CGI Script, Servlet or another JSP page.

Syntax:

```
<jsp:include page="<FileName>" flush="<true|false>" />
```

Where,

- page: Is a string or an expression representing the relative URL of the component to which the request is forwarded. The component can be either another JSP page, a Servlet or any other object that can respond to a request
- The **flush** attribute is optional. It indicates whether the output buffer should be flushed before the included file is called. The default value of flush attribute is false

If the JSP page requires passing parameters to the included file, then the <jsp:param> element can be put to use, where one element has to be used for each parameter. To use <jsp:param> element, both the **name** and **value** attributes are required. The included JSP page can access the parameters using the getParameter() and getParameterValues() methods of the request object.

Syntax:

```
<jsp:include page="<FileName>" flush="<true|false>" />
   <jsp:param name="<ParamName1>" value="<ParamValue1>" />
   <jsp:param name="<ParamName2>" value="<ParamValue2>" />
</jsp:include>
```

Where,

- page: Is a string or an expression representing the relative URL of the component to which the request is forwarded. The component can either be another JSP page, a Servlet or any other object that can respond to a request
- The **flush** attribute is optional. It indicates whether the output buffer should be flushed before the included file is called. The default value of flush attribute is false
- The <jsp:param> element is used to send one or more **name=value** pairs as parameters to a dynamic resource. The target resource should be dynamic **i.e.** a JSP page, a Servlet or other resources, which can process the data that is sent to it as parameters

The following table shows the technical difference between the **include directive** and **include action element:**

Include	**Done When**	**Included Content**	**Parsing**
DIRECTIVE	Compilation time	Static	Parsed by engine
ACTION	Request processing time	Static or Dynamic	Not parsed but included in place

The <jsp:include> action inserts the file at the time the page is requested and not at the time the JSP page is translated into a Servlet which is exactly what the include directive does. In <jsp:include> method, there is a small percentage of efficiency loss but however it gains significantly in flexibility.

The **<jsp:include>** element is processed when a JSP page is executed. The include action allows including either a static or dynamic resource in a JSP file. If the resource is static, its content is inserted into the calling JSP file. If the resource is dynamic, the JSP page stops processing the current request the moment it encounters **<jsp:include>** element and passes the request to the included **action**. The included page is executed and then the result is included in the response from the calling JSP page.

The include **directive** is processed when the JSP page is translated into a Servlet class. The effect of the directive is to insert the text contained in another file which can be either static content or another JSP page in the including JSP page. This directive can be used to include banner content, copyright information or any chunk of content that can be reused in another page.

For example, a JSP page inserts three different snippets into a Web page. Each time the content in those snippets require a change, page authors only need to update the three files and can leave the main JSP page unchanged.

Both i.e. the <jsp:include> element and the include directive have their pros and cons and, as always, it just depends on what the requirements are. Following are the advantages of using <jsp:include>:

- Guarantees automatic recompilation
- Smaller class sizes
- Can make use of parameters
- JSP expressions can be used in attribute values and
- Can include text from any source

HINT

One of the main benefits of using the **include directive** is that local page variables can be shared between the two files. It also has slightly better run time efficiency and does not restrict output buffering.

REMINDER

<jsp:include> is useful in a newspaper scenario where the articles change daily and its desired, to avoid, if possible having to recompile JSPs every day/hour. Using the <jsp:include> method, the HTML article files that are included are built on a daily/hourly basis with the JSP page loading in the articles by their article IDentities.

Example:

```
<jsp:include page="/index.html" />
```

Explanation:
In the above example, the <jsp:include> element includes a static HTML file name **index.html** into the JSP page. The <jsp:include> element does not have a body and therefore a shorthand notation is used where the start tag ends with '/>' instead of '>'.

Included files can access all information in the request object and therefore, the included files have access to any form variables passed from the Web browser. To pass parameters to the included file, the <jsp:param> element is used.

Example:

```
<jsp:include page="/index.jsp">
   <jsp:param name="greeting" value="Hello" />
</jsp:include>
```

Explanation:
Here, the <jsp:include> element includes a dynamic JSP file named index.jsp into the current JSP page. The <jsp:param> element is used to pass value to the included dynamic JSP file.

The <jsp:param> element does not have a body and therefore a shorthand notation is used where the start tag ends with '/>' instead of '>'. The <jsp:include> element has a body consisting of the <jsp:param> element and therefore it has a closing tag </jsp:include>, which indicates the end of the <jsp:include> element.

The included file [i.e. index.jsp] uses request.getParameter() and request.getParameterValues() methods to fetch the parameters just as if the parameters were passed from the Web browser as form variables.

The values from the <jsp:param> element take precedence over the parameters already in the request. In short, if the getParameter() method is used to retrieve the parameter value, then the value specified in the <jsp:param> element is retrieved and if the getParameterValues() method is used, then both the value specified by the <jsp:param> element and the value passed from the Web browser are retrieved.

HINT

The parameters passed by the <jsp:param> element are visible to the included page only. They are not visible to the original page i.e. they do not affect the original set of parameters.

Example:
The following example demonstrates the <jsp:include> action element using two files.

The **first file** is named included.jsp. This file accepts a parameter named **name**. The value held by this parameter is printed as shown in diagram 15.1. If the parameter holds a blank value i.e. this file does not receive a parameter value, the message as shown in diagram 15.2 is printed.

The **second file** is named index.jsp. This file using the <jsp:include> action element includes the first file i.e. included.jsp and passes it a parameter value using the <jsp:param> action element.

Create a JSP named included.jsp with the following code spec, which is included in another JSP named index.jsp:

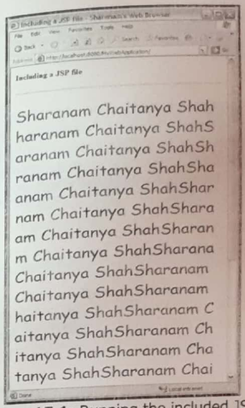
```
<%
/* Fetching the value held in the parameter passed by the
index.jsp file */
   String strName = request.getParameter("name");
```

```
/* Validating the value held inside the parameter */
   if((strName==null) || (strName.equals("")))
      strName = "Parameter not provided";
%>
<P>
   <FONT FACE="Comic Sans MS" SIZE="18" COLOR="blue">
      <%
      /* Getting the length of string */
      int intLength = strName.length();
      /* A loop that will display the value i.e. name */
      for(int i=0; i < intLength; i++)
      {
         out.println(strName + "<BR>");
         /* Getting the first character */
         char chrFirst = strName.charAt(0);
         /* Moving the first character to end of string */
         strName = strName.substring(1) + chrFirst;
      }
      %>
   </FONT>
</P>
```

Create a JSP named index.jsp with the following code spec:

```
<HTML>
   <HEAD>
      <TITLE>Including a JSP file</TITLE>
   </HEAD>
   <BODY>
      <H2>Including a JSP file</H2>
      <HR>
   <!-- Including the included.jsp page and passing it a value -->
      <jsp:include page="included.jsp" flush="true">
         <jsp:param name="name" value="Sharanam Chaitanya Shah"/>
      </jsp:include>
   </BODY>
</HTML>
```

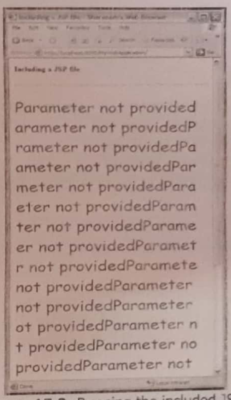
Execute the example. The output is as shown in diagram 15.1.



```
Including a JSP file

Sharanam Chaitanya Shah
haranam Chaitanya ShahS
aranam Chaitanya ShahSh
ranam Chaitanya ShahSha
anam Chaitanya ShahShar
nam Chaitanya ShahShara
am Chaitanya ShahSharan
m Chaitanya ShahSharana
Chaitanya ShahSharanam
Chaitanya ShahSharanam
haitanya ShahSharanam C
aitanya ShahSharanam Ch
itanya ShahSharanam Cha
tanya ShahSharanam Chai
```

Diagram 15.1: Running the included JSP file with parameter



```
Including a JSP file

Parameter not provided
arameter not providedP
rameter not providedPa
ameter not providedPar
meter not providedPara
eter not providedParam
ter not providedParame
er not providedParamet
r not providedParamete
not providedParameter
not providedParameter
ot providedParameter n
t providedParameter no
providedParameter not
```

Diagram 15.2: Running the included JSP file without parameter

In the index.jsp page make the following modifications and execute the example. The output is as shown in diagram 15.2.

```
<!-- Including the included.jsp page and passing it a value -->
   <jsp:include page="included.jsp" flush="true">
      <jsp:param name="name" value="Sharanam Chaitanya Shah"/>
   </jsp:include>
```

TO

```
<!-- Including the included.jsp page -->
   <jsp:include page="included.jsp" flush="true" />
```

Forwarding JSP Page To Another Page

The current JSP page can be transferred to another page without returning back to the current JSP page. This transferring of JSP page to another page is known as **forwarding**.

The **<jsp:forward>** action element allows forwarding i.e. it permanently transfers processing from one JSP to another on the local server. Any content generated by the original page is discarded and the processing begins fresh at the second JSP.

Using **<jsp:forward>** element, the request can be forwarded to another JSP, a Servlet or a static resource.

HINT

<jsp:forward> is different from the <jsp:include> element. The <jsp:include> element can call the included file at any time during a response. The <jsp:forward> element must forward the request prior to writing any output to the OutputStream. If the output stream is not buffered and some output has been written to it, a <jsp:forward> element throws a java.lang.IllegalStateException.

Syntax: [Without passing parameters]

```
<jsp:forward page="<url>" />
```

Where,

- **page:** Is a string or an expression representing the relative URL of the component to which the request is forwarded. The component can be either another JSP page, a Servlet or any other object that can respond to a request

Example:

```
<jsp:forward page="/servlet/register" />
```

Explanation:
In the above example, the <jsp:forward> element forwards the request to a Servlet named register.

Parameters can be passed to a forwarded page in the same manner as were passed to an included page whilst using <jsp:include>. While working with JSP the <jsp:param> element is used and while working with a Servlet the parameters are added to the end of the URL when the **getRequestDispatcher** method is called.

Syntax: [With passing parameters]

```
<jsp:forward page="<urlSpec>">
   <jsp:param name="<ParameterName>" value="<ParameterValue>" />
</jsp:forward>
```

Where,

- **page:** Is a string or an expression representing the relative URL of the component to which the request is forwarded. The component can be either another JSP page, a Servlet or any other object that can respond to a request

- The <jsp:param> element is used to send one or more **name=value** pairs as parameters to a dynamic resource. The target resource should be dynamic i.e. a JSP page, a Servlet or other resources, which can process the data that is sent to it as parameters

HINT

More than one <jsp:param> element can be used, in case more than one parameter needs to be send to the target resource. The **name** attribute of the <jsp:param> element specifies the parameter name and takes a case-sensitive literal string as a value, whereas the **value** attribute specifies the parameter value and takes either a case-sensitive literal string or an expression that is evaluated at request time.

The moment, JSP encounters **<jsp:forward>:**

- Execution in the current JSP halts
- Buffer is cleared
- Request is modified to argument any additionally specified parameters

Example:

```
<jsp:forward page="/servlet/register">
   <jsp:param name="username" value="vaishali" />
</jsp:forward>
```

Explanation:
In the above example, the <jsp:forward> element forwards the request to the Servlet named register. Additionally, it also passes a parameter to the Servlet using the <jsp:param> element.

WARNING

Be careful while using the <jsp:forward> element with un-buffered output. If the page directive with buffer="none" is used for specifying that the output of the JSP page should not be buffered and if the JSP page has any data in the output stream object, then using the <jsp:forward> element causes an **IllegalStateException**.

Example:
The following example demonstrates using the <jsp:forward> action element. Following files are created to demonstrate the <jsp:forward> element:

- **login.html:** This file is the user interface that accepts the username and the password from the user via an HTML FORM

- doLogin.jsp: This file is the server side processing file. The username and password captured by login.html file is passed on for further processing to this file. doLogin.jsp validates the username and password and performs the following actions:
  - If the username is found invalid then using the **<jsp:forward>** action element the control is shifted to the showMsg.jsp page with a parameter value **"Wrong Username"**
  - If the password is found invalid then using the **<jsp:forward>** action element the control is shifted to the showMsg.jsp page with a parameter value **"Wrong Password"**
- showMsg.jsp: This file is the file that takes control from the server side processing file. This file accepts a parameter named **failReason** and based on the value held in this parameter an appropriate message is displayed. This file also links to the login.html file to allow the user to try again

Create an HTML form named login.html that accepts username and password and passes the username and the password captured to the server-side processing file named doLogin.jsp with the following code spec:

```
<HTML>
  <HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=UTF-8">
    <TITLE>Login</TITLE>
  </HEAD>

  <BODY BGCOLOR="pink">
  <!-- Initializing a form object. -->
    <FORM NAME="frmLogin" ACTION="/MyWebApplication/MyJSP/Login"
            METHOD="post">
      <TABLE ALIGN="center" BGCOLOR="PINK" BORDER="0" CELLPADDING="0"
              CELLSPACING="0" NAME="tblOuter" WIDTH="50%">
        <TR HEIGHT="300" VALIGN="top">
          <TD ALIGN="center" BORDER="1" COLSPAN="10">
          <BR>
            <TABLE ALIGN="center" BORDER="1"
                    BORDERCOLOR="SKYBLUE" CELLPADDING="2"
                    CELLSPACING="0" NAME="tblFirstChild" WIDTH="100%">
            <!-- First Child Table: First Row: Column to
            hold the Form Title . -->
            <!-- Inserting table row to display the form
            title -->
              <TR BGCOLOR="BLACK">
                <TD ALIGN="left" COLSPAN="2">
                <!-- Formatting the font of title -->
                  <FONT SIZE="2" COLOR="white"><B>Login</B>
```

```
                  </FONT>
                </TD>
              </TR>
            <!-- Inserting table row that holds table data
            label and text box of Username -->
              <TR>
                <TD ALIGN="right" WIDTH="25%">User Name:</TD>
                <TD ALIGN="left">
                  <INPUT MAXLENGTH="35" NAME="txtUsername"
                          TYPE="text" SIZE="10">
                </TD>
              </TR>
            <!-- Inserting table row that holds table data
            label and text box of Password -->
              <TR>
                <TD ALIGN="right" WIDTH="25%">Password:</TD>
                <TD ALIGN="left">
                  <INPUT MAXLENGTH="35" NAME="txtPassword"
                          TYPE="password" SIZE="10">
                </TD>
              </TR>
            <!-- Inserting table row that holds inputs for
            Submit button -->
              <TR>
                <TD COLSPAN="2" ALIGN="center">
                  <INPUT NAME="cmdSubmit" TYPE="submit"
                          VALUE="Submit">
                </TD>
              </TR>
            </TABLE>
          </TD>
        </TR>
      </TABLE>
    </FORM>
  </BODY>
</HTML>
```

Create a JSP named doLogin.jsp. This page validates the login information received from login.html. If the login information is invalid then the control is transferred to a JSP named showMsg.jsp along with an appropriate error message. If the login information is valid then a welcome message is delivered by doLogin.jsp:

```
<%@ page language="java" import="java.util.*" %>
<HTML>
  <HEAD>
    <TITLE>The Error Message</TITLE>
  </HEAD>
```

```
<BODY>
    <%
        String userName = request.getParameter("txtUsername");
        String userPasswd = request.getParameter("txtPassword");
        if(!userName.equals("sharanam"))
        {
    %>
            <jsp:forward page="/MyJSP/ShowMessage">
                <jsp:param name="failReason" value="Wrong Username"/>
            </jsp:forward>
    <%
        }
        if(!userPasswd.equals("shah"))
        {
    %>
            <jsp:forward page="/MyJSP/ShowMessage">
                <jsp:param name="failReason" value="Wrong Password"/>
            </jsp:forward>
    <%
        }
    %>
        <H2>Welcome! &nbsp; <%=userName%></H2>
        The date and time of login is<%= (new java.util.Date()).toLocaleString() %>
  </BODY>
</HTML>
```

Create a JSP named showMsg.jsp. This page displays an appropriate error message, based on the value held by the parameter received from doLogin.jsp:

```
<%@ page language="java" import="java.util.*"%>
<HTML>
  <HEAD>
      <TITLE>The Error Message</TITLE>
  </HEAD>
  <BODY>
      <H2>Error Message: &nbsp; <%= request.getParameter("failReason")%>
      </H2>
      Click <A HREF="/MyWebApplication/Login.html">here</A> to try again.
  </BODY>
</HTML>
```

Execute login.html, enter the username as **sharanam** and password as **shah**. This is because the username / password pair has been hard coded in doLogin.jsp for demonstration purpose.

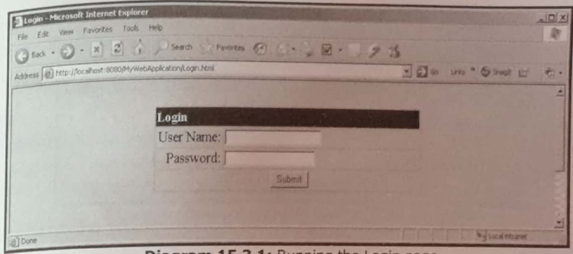
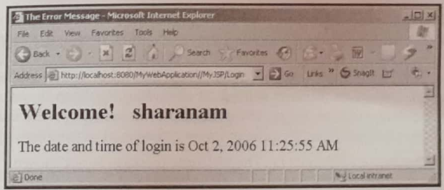

Login - Microsoft Internet Explorer
http://localhost:8080/MyWebApplication/Login.html
Login
User Name:
Password:
Submit

Diagram 15.3.1: Running the Login page

Since the login information is valid a welcome message is displayed.

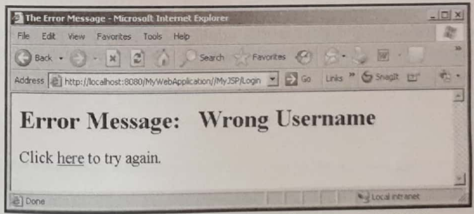

The Error Message - Microsoft Internet Explorer
http://localhost:8080/MyWebApplication//MyJSP/Login

Welcome! sharanam

The date and time of login is Oct 2, 2006 11:25:55 AM

Diagram 15.3.2: Welcome message for valid login information

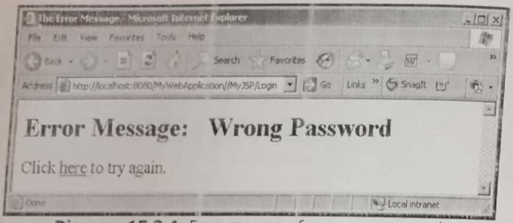
Enter a wrong username/password. This forwards the request to showMsg.jsp, which in turn displays an appropriate error message.


The Error Message - Microsoft Internet Explorer
http://localhost:8080/MyWebApplication//MyJSP/Login

Error Message: Wrong Username

Click here to try again.

Diagram 15.3.3: Error message for wrong username



The Error Message - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8080/MyWebApplication/MyJSP/Login
Go Links SnagIt
Done
Local intranet

Error Message: Wrong Password

Click here to try again.

Diagram 15.3.4: Error message for wrong password

Explanation:

As soon as the username and the appropriate password is submitted using login.html, control shifts to doLogin.jsp. doLogin.jsp extracts the username and password into memory variables.

If the memory variables hold the correct values then a welcome message is displayed.

If the memory variable holds a value other than the hard coded values then using **<jsp:forward>** the control is further shifted to showMsg.jsp with a parameter named **failReason** that holds the failure reason. showMsg.jsp simply displays the value held inside the parameter received. This shows that the moment doLogin.jsp encounters **<jsp:forward>:**

- Execution in the current JSP halts i.e. The welcome message is not shown
- Buffer is cleared
- Request is modified to argument any additionally specified parameters

Passing Parameters For Other Actions

The <jsp:param> element is used to specify additional request parameters for the target resource by appending parameters to the request object. This element is used in the body of a

- <jsp:forward>

OR

- <jsp:include>

Syntax:

```
<jsp:param name="<ParameterName>" value="<ParameterValue>" />
```

Where,

- name: Is the name of the parameter
- value: Is the value the parameter holds

Example:

```
<jsp:param name="username" value="vaishali" />
```

The <jsp:param> element sends a **name=value** pair as parameters to a dynamic resource. In the above example, name=value pair is **username=vaishali**.

The Plug-Ins

Applets are used for providing a rich user experience through a Web browser by allowing the user to interact with the GUI. Applets are normally used for client side processing.

The JSP environment provides a special element, the <jsp:plugin> element, for embedding an applet or a Java Bean component in a Web application. This element automatically detects the browser type and inserts the appropriate HTML elements [either <embed> or <object>].

In other words, the **<jsp:plugin>** element is used to generate client browser specific HTML tags [OBJECT or EMBED] that ensures that the Java Plug-in software is available, followed by execution of the applet or the Java Beans component specified in the tag.

The following are the attributes associated with <jsp:plugin> element:

- type: Identifies the type of the component such as **bean** or **applet** that is embedded. This attribute is mandatory
- code: Holds the fully qualified name of the component class. This attribute is mandatory
- align: Holds the alignment such as **bottom**, **middle** or **top** of the component
- codebase: Is the relative URL pointing to the directory that holds the class file. The directory must be a subdirectory to the directory holding the page according to the HTML 4.0 spec
- height: Is the height of the applet area in pixels / percentage. This attribute is not mandatory, but some browsers do not allow an object of zero height due to security issues
- hspace: Is the amount of white space to be inserted to the left and right of the component area in pixels
- archive: Is a comma-separated list of URIs to the archives containing classes and other resources that is preloaded