


```
% page [attributeName = attributeValue] %
```

```
isErrorPage = true
errorPage   = name of jsp file
```

`errorPage` => This attribute is used to specify if exception occurs to which page the exception object should be delegated.

isErrorPage => This attribute takes a default value as false, meaning the exception object would not be available inside this page to handle, to make the exception object available to jsp page we need to set the boolean value to true.

Servlet =====>	Processing logic	JSP ==> frontend	Main intention of JSP is to bring frontend developers(HTML/CSS) also to build webapplication using Java(with zero knowledge of Java).
JSP =====>	Presentation logic(in dynamic fashion)	 Java	JSP => No Java, but to bring dynamic nature we need to write Java code To resolve this problem they used a new approach called "ExpressionLanguage"

```
isElIgnored = "true" => syntax won't be processed rather it treats as Template Text.
isElIgnored = "false" => syntax will be processed and it prints the value
```

Note: default value is false.

<pre>%@ page isIgnored = "false"%> <h1> The userName is :: \${param.user}

 The password is :: \${param.password}

 </h1></pre>	<pre>http://localhost:9999/SecondApp/index.jsp?user=sachin&password=tendulkar output The userName is sachin The password is tendulkar</pre>
---	---

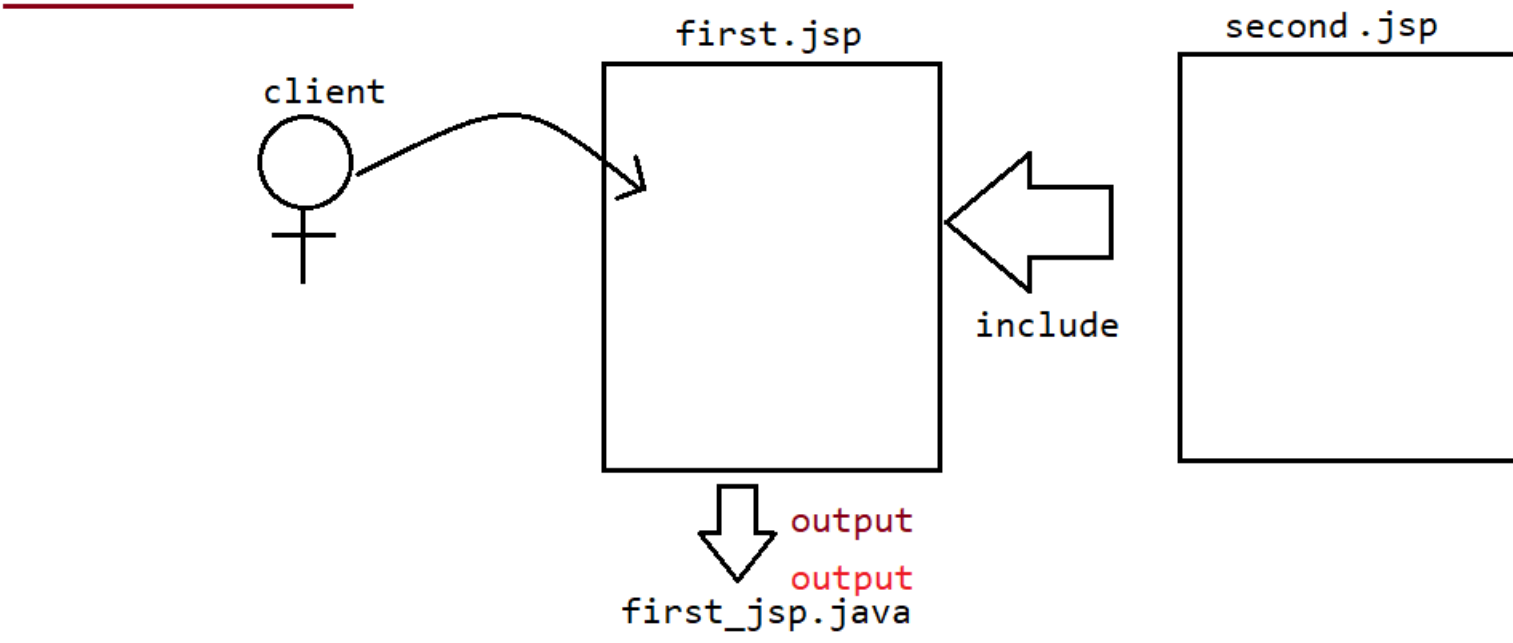
```

session
    ↓ false means in the current jsp page session object is not accessible,
    <% page session = "false" %> default value is true

    <session.setAttribute("Name", "iNeuron");
    session.setAttribute("Java", "NavinReddy");
%>
<h1>The name of the company is :: <% session.getAttribute("Name") %></h1><br>
<h1>The trainer name of java is :: <% session.getAttribute("Java") %></h1>

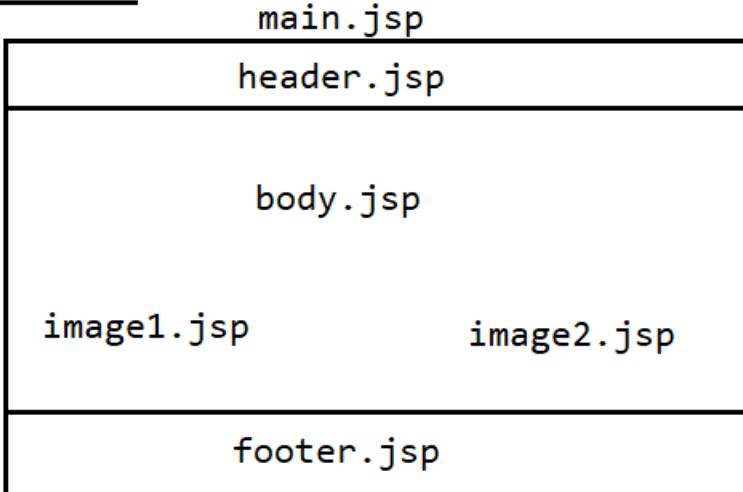
```

include directive



The content of second.jsp will be included in the current jsp during translation phase. since the inclusion is happening at the translation phase we call such inclusion as "static inclusion"

Assignment

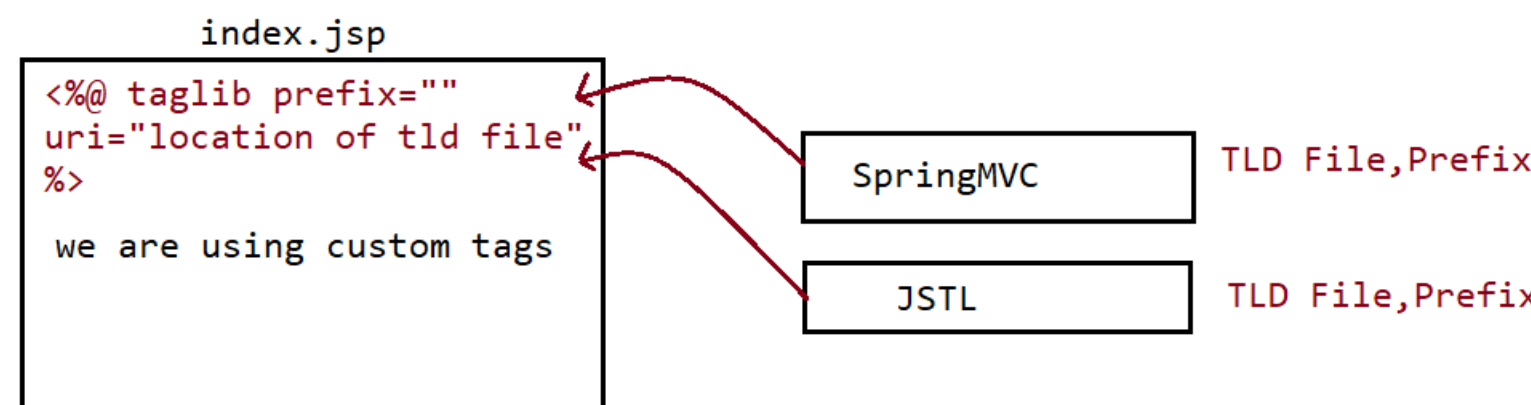


taglib directive

customization

<pre><mine:iNeuron> //body of the tag </mine:iNeuron></pre>	<ol style="list-style-type: none">1. TLD file(Tag library descriptor)2. Prefix
---	---

For few operations to be made easy for developers, the customized tags are already been coded by 3rd party vendor like SpringPivotal team, JSTL libraries by 3rd parties and so on....



Scripting Elements

- a. Declartive tag
- b. Scriptlet tag
- c. Expression tag
- d. comments
 - a. html comments
 - b. jsp comments
 - c. java based comments


```
Comments
a. JSP      comments <%-- JSP COMMENTS --%>
b. HTML     comments <!-- HTML COMMENTS -->
c. Java     comments

1. // single line
2. /*
   *      multiline
   */
3. /**
   *      java doc
```

Comments visibility


1. **JSP Comments**
Visible only in jsp page
Translated servlet not visible
2. **HTML Comments**
Visible in JSP page
Visible in Translated servlet also
3. **Java Comments**
Visible in JSP page
Visible in Translated servlet also

Expression Tag

Syntax:  `_jspservice()`
`<%= x %>`
`{`
`out.print(x);`
`}`

Scriptlet Tag

Syntax:

<pre><% //write java code %></pre>		<pre>_jspservice() { //java code }</pre>
--	---	--

Delcartive Tag

Syntax:

```
<%!  
//Any java declarations  
%>
```

⇒ The logic will be placed inside the servlet but outside `_jspService(...)`

Note: Inside JSP, there are 9 implicit objects available and these objects are local to `_jspService()` so these variables directly can't be used inside Declarative Tag.

Implicit Objects

- ```

1. request => HttpServletRequest
2. response => HttpServletResponse
3. out => JspWriter(AC)
4. config => ServletConfig
5. application => ServletContext
6. exception => java.lang.Throwable
7. session => HttpSession
8. page => java.lang.Object
9. pageContext => java.servlet.jsp.PageContext(AC)

```

Note:

```

<h1>
 <%!
 int i = 0;
 %>
 <%
 i++;
 out.println(i);
 %>
</h1>

 output
 i = 1,2,3,4,5,...

```

### Program to demonstrate the usage of all Scripting elements in JSP

```

<%@ page import = "java.util.Date" %>
<%!
 Date d = null;
 String date = "";
%>
%>
<%
 d = new Date();
 date = d.toString();
%>
<ch1 style = 'color:red;'>
 Today date is <%= date%>
</h1>

```

```

import java.util.Date;
public class index_jsp extends
{
 Date d = null;
 String date = "";

 public void _jspService(...)
 {
 d = new Date();
 date = d.toString();

 out.write("<h1 style='color:red;'>");
 out.write("Today date is ");
 out.print(date);
 out.write("<\/h1>");
 }
}

```

Diagram annotations:

- Declarative tag**: points to `<%@ page import = "java.util.Date" %>`
- scriptlet**: points to `<%!`
- expression**: points to `<%= date%>`

web.xml

```
<web-app>
 <display-name>JSP Implicit Object Application</display-name>
 <context-param>
 <param-name>User</param-name>
 <param-value>root</param-value>
 </context-param>
 <context-param>
 <param-name>Password</param-name>
 <param-value>root123</param-value>
 </context-param>
</web-app>
```

Context Object data

index.jsp

```
<h1>
 The context parameter username is ::
 <%= application.getInitParameter("User")%>

 The context parameter password is ::
 <%= application.getInitParameter("Password")%>

 The Application name is ::
 <%= application.getServletContextName()%>
</h1>
```

web.xml

```

<web-app>
 <servlet>
 <servlet-name>Demo</servlet-name>
 <jsp-file>/config.jsp</jsp-file>
 <init-param>
 <param-name>IPL</param-name>
 <param-value>BCCI</param-value>
 </init-param>
 </servlet>
 <servlet-mapping>
 <servlet-name>Demo</servlet-name>
 <url-pattern>/test</url-pattern>
 </servlet-mapping>
</web-app>

```

```

<h1>
 The Init param is :: <%= config.getInitParameter("IPL")%>

 The logical name of Servlet is :: <%= config.getServletName()%>

 <%
 java.util.Enumeration<String> params=config.getInitParameterNames();
 while(params.hasMoreElements()){
 String data = (String)params.nextElement();
 System.out.println(data + "::" + config.getInitParameter(data));
 }
 %>
 Context Name is <%=config.getServletContext()%>
</h1>

```

pageContext

1. We can get remaining implicit object using `pageContext` object
2. To perform `requestDispatching` mechanism
  - a. `include(req,resp)`
  - b. `forward(req,resp)`
3. To perform attribute management in any scope

**Note:**

```
request = pageContext.getRequest()
response = pageContext.getResponse()
config = pageContext.getServletConfig()
application = pageContext.getServletContext()
session = pageContext.getSession()
out = pageContext.getOut()
exception = pageContext.getException()
page = pageContext.getPage()
```

The diagram illustrates the `forward()` method in JSP. It shows a client sending a request to `first.jsp`. `first.jsp` then forwards the request to `second.jsp`. The response from `second.jsp` is sent back to the client. The diagram shows the flow of request and response between the client, `first.jsp`, and `second.jsp`, with the `forward()` method being the key action.