JAVA SERVER PAGES

PART-2

Directives:

There are three directives in JSP.

1) page directive

2) include directive

3) taglib directive

1) page directive:

Syntax:

<%@ page ………………. %>

Attributes:

1) language="………"=> Java language is a default. JSP supports java language only.

2) import="……" => It is used to import packages. The following packages are implicitly imported in JSP.

i) java.lang package

ii) javax.servlet package

iii) javax.servlet.http package

iv) javax.servlet.jsp package

Examples:

1) <%@ page import=”java.util.*” %>

2) <%@ page import=”java.util.*, java.sql.*” %>

3) buffer=”………” => It is used to specify the buffer size

4) autoFlush=”…..” => true is default. Data automatically flushed.

The above two attributes are used for utilizing buffer memory efficiently.

5) errorPage=”…..” => It is used to specify the url of an error page. Whenever exception occurs in a program control jumps to specified error page.

6) isErrorPage=”…….” => Here false is default. It is used in error page only. It is used to display user friendly error message.

7) contentType=”…….”=> It is used to specify MIME(Multipurpose Internet Mail Extensions)type. Default mime type is “text/html”

8) session=”…….” => true is default.

Example: <%@ page session=”false” %> => This statement indicates web container that do not create session. It does not allow to access implicit session object also.

9) isThreadSafe=”……” => default is true.

Example: <%@ page isThreadSafe=”false” %> => This statement indicates web container to implement javax.servlet.SingleThreadModel interface in a converted servlet. If that interface implemented in a servlet, then that servlet allows to access only one thread at a time.

10) isELIgnored=”…..”=> default is false.

Example: <%@ page isELIgnored=”true” %> => This statement indicates web container that do not allow expression language in a JSP.

2) include directive:

Syntax: <%@ include file=”………..” %> => It is used to include jsp file or html file.

Example:

```
<html>

<body bgcolor=yellow text=red>

<h1><u> Include Directive Example </u></h1>

<%@ include file="test.jsp" %>

<%@ include file="time.jsp" %>

<%@ include file="table.jsp" %>

</body>

</html>
```

3) taglib directive:

Syntax: <%@ taglib prefix=”…..” uri=”……” %> => It is used to include tag libraries.

JSP Actions:

1) <jsp:forward>

2) <jsp:include>

3) <jsp:useBean>

4) <jsp:setProperty>

5) <jsp:getProperty>

6) <jsp:plugin>

7) <jsp:fallback>

8) <jsp:params>

9) <jsp:param>

1)<jsp:forward>:

It is used to forward the request and response of one jsp to another jsp.

Example:

arithmetic.html

```
<html>

<body bgcolor=green text=yellow>

<center>

<h1><u>Arithmetic Application</u></h1>

<form action=arithmetic.jsp>

First Number <input type=text name=first><br>

Second Number <input type=text name=second><br><br>

<input type=submit name=operation value=ADD>

<input type=submit name=operation value=SUB>

<input type=submit name=operation value=MUL>

<input type=submit name=operation value=DIV>

</form>

</center>
```

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

arithmetic.jsp

```
<% String s=request.getParameter("operation");
if(s.equals("ADD")) { %>
<jsp:forward page="add.jsp" />
<% } else if(s.equals("SUB")) {  %>
<jsp:forward page="sub.jsp" />
<% } else if(s.equals("MUL")) { %>
<jsp:forward page="mul.jsp" />
<% } else { %>
<jsp:forward page="div.jsp" />
<% } %>
```

add.jsp

```
<html>
<body bgcolor=red text=yellow>
<h1>
<% String s1=request.getParameter("first");
String s2=request.getParameter("second");
int a=Integer.parseInt(s1);
int b=Integer.parseInt(s2);
out.println("Output: "+(a+b));
```

```
%>
</h1>
</body>
</html>
```

sub.jsp

```
<html>
<body bgcolor=blue text=yellow>
<h1>
<% String s1=request.getParameter("first");
String s2=request.getParameter("second");
int a=Integer.parseInt(s1);
int b=Integer.parseInt(s2);
out.println("Output: "+(a-b));
%>
</h1>
</body>
</html>
```

mul.jsp

```
<html>
<body bgcolor=green text=yellow>
<h1>
<% String s1=request.getParameter("first");
```

```
String s2=request.getParameter("second");
int a=Integer.parseInt(s1);
int b=Integer.parseInt(s2);
out.println("Output: "+(a*b));
%>
</h1>
</body>
</html>
```

div.jsp

```
<html>
<body bgcolor=yellow text=red><h1>
<% String s1=request.getParameter("first");
String s2=request.getParameter("second");
int a=Integer.parseInt(s1);
int b=Integer.parseInt(s2);
out.println("Output: "+(a/b));
%>
</h1>
</body>
</html>
```

2)<jsp:include>:

It is used to include the request and response of one jsp into another jsp.

Example:

header.jsp

```
<html>
<body>
<center>
<font color=red>
<h1><u><% out.println("JSP Tutorial"); %></u></h1>
</font>
</center>
</body>
</html>
```

footer.jsp

```
<html>
<body>
<center>
<font color=green>
<h3><% out.println("By"); %></h3>
<h2><% out.println("Venkatesh Mansani"); %></h2>
<h1><% out.println("Naresh i Technologies"); %></h1>
```

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

page1.jsp

```
<html>
<body bgcolor=yellow text=blue>
<jsp:include page="header.jsp" />
<h1><u>JSP:</u></h1>
<% out.println("JSP stands for Java Server Pages. JSP is a specification
for developing web applications with Java programming language.");
%>
<jsp:include page="footer.jsp" />
</body>
</html>
```

page2.jsp

```
<html>
<body bgcolor=yellow text=blue>
<jsp:include page="header.jsp" />
<h1><u>JSTL:</u></h1>
<% out.println("JSTL stands for Java Server Pages Standard Tag Library.
JSTL introduced with JSP 2.0 version to simplify the JSP."); %>
<jsp:include page="footer.jsp" />
```

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

3)<jsp:useBean>:

It is used to create an object to a Java bean class.

4)<jsp:setProperty>:

It is used to call setter methods of a Java bean.

5)<jsp:getProperty>:

It is used to call getter methods of a Java bean.

Example:

bean.html

```
<html>
<body bgcolor=red text=yellow><h1>
<jsp:useBean id="mb" class="beanapp.MessageBean">
<jsp:setProperty name="mb" property="message" value="Welcome" />
<jsp:getProperty name="mb" property="message" />
</jsp:useBean>
</h1></body>
</html>
```

MessageBean.java

```
package beanapp;
import java.io.Serializable;
```

```
public class MessageBean implements Serializable{
        private String message;
        public String getMessage() {
                return message;
        }
        public void setMessage(String message) {
                this.message = message;
        }
}
```

By

Mr. Venkatesh Mansani

Naresh i Technologies