Top of Form

**JSP Declaration tag**

Declaration tag is a block of java code for declaring class wide variables, methods and classes. Whatever placed inside these tags gets initialized during JSP initialization phase and has class scope. JSP container keeps this code outside of the service method (\_jspService()) to make them class level variables and methods.

As we know that variables can be initialized using [**scriptlet**](https://beginnersbook.com/2013/05/jsp-tutorial-scriptlets/) too but those declaration being placed inside \_jspService() method which doesn’t make them class wide declarations. On the other side, **declaration tag** can be used for defining class level variables, methods and classes.

**Syntax of declaration tag:**

<%! Declaration %>

**Example 1: Variables declaration**

In this example we have declared two variables inside declaration tag and displayed them on client using [**expression tag**](https://beginnersbook.com/2013/11/jsp-expression-tag/).

<html>

<head>

<title>Declaration tag Example1</title>

</head>

<body>

<%! String name="Chaitanya"; %>

<%! int age=27; %>

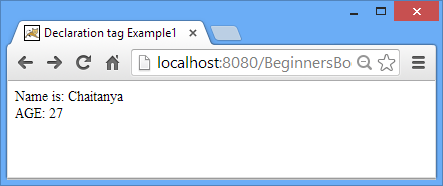
<%= "Name is: "+ name %><br>

<%= "AGE: "+ age %>

</body>

</html>

**Output**:



**Example 2: Methods declaration**

In this example we have declared a method **sum** using **JSP declaration tag**.

<html>

<head>

<title>Methods Declaration</title>

</head>

<body>

<%!

int sum(int num1, int num2, int num3){

return num1+num2+num3;

}

%>

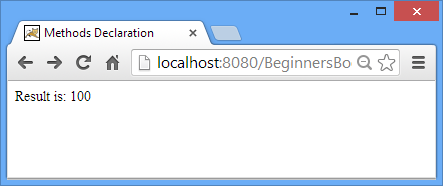
<%= "Result is: " + sum(10,40,50) %>

</body>

</html>

**Output:**

Sum of all three integers gets displayed on the browser.



Bottom of Form

Top of Form

**JSP Scriptlets**

**Scriptlets are nothing but java code** enclosed within **<% and %> tags**. JSP container moves the statements enclosed in it to **\_jspService()** method while generating servlet from JSP. The reason of copying this code to service method is: For each client’s request the \_jspService() method gets invoked, hence the code inside it executes for every request made by client.

**Syntax of Scriptlet:**

[code language=”java”]<% Executable java code%>[/code]

**JSP to Servlet transition for Scriptlet –**

As I stated in my previous tutorials that JSP doesn’t get executed directly, it first gets converted into a Servlet and then Servlet execution happens as normal. Also, I explained in first para that while translation from JSP to servlet, the java code is copied from scriptlet to \_jspService() method. Lets see how that happens.

**Sample JSP code:**

[code language=”html”]

<H3> Sample JSP </H3>

<% myMethod();%>

[/code]

**Note:** Semicolon at the end of scriptlet.

**Corresponding translated Servlet code for above JSP code:**

[code language=”java”]

public void \_jspService(HttpServletRequest request,

HttpServletResponse response)

throws ServletException, IOException {

response.setContentType("text/html");

HttpSession session = request.getSession();

JspWriter out = response.getWriter();

out.println("<H2>Sample JSP</H2>");

myMethod();

}[/code]

**An example to learn about Scripting elements:**

[code language=”java”]

<%– A jsp example to learn the JSP scripting elements–%>

<%String string1 ="JSP scriptlet";%>

<%!String string2 ="";%>

<html>

<head>

<title> JSP page: Welcome </title>

</head>

<body>

<h1>

<%–This is an Expression statement–%>

Welcome to <%=string1%>

</h1>

<%–sciptlet example–%>

<%if(localstring.equals("JSP scriptlet")){%>

Hi

<%}

else {%>

hello

<%} %>

<%–same thing can be done in this way also–%>

<%if(localstring.equals("JSP scriptlet"))

out.println("Hi"+string2);

else

out.println("hello");

%>

</body>

</html>[/code]

In the above example there are many type of JSP elements present such as Expression, JSP comment, Declaration element etc. We will see each one of them in upcoming JSP tutorials but as of now you can only focus on Scriptlets. The below are the scriptlets statements used in above example –

[code language=”java”]

<%if(localstring.equals("JSP scriptlet"))

out.println("Hi"+string2);

else

out.println("hello");

%>[/code]

The above code is a JSP scriptlet (notice starting <% and ending %> tags). If you analyze above piece of code then you would find that the code inside tags is a pure java code so in order to execute java code in JSP we use scriptlets.

[code language=”java”]<%String string1 ="JSP scriptlet";%>[/code]

Like above set of statements this statement is a java initialization code which is enclosed within tags.

Apart from above two set of scriptlets there are many other scriptlet tags present in above example (notice if-else control flow logic). To use the if-else control flow statements of java, we have used scriptlet in above example. As this is the main advantage of using scriptlet so lets make it more clear with the help of an example – You must be aware how important are our If – else control statements.

**An example to show use of if -else using scriptlet –**

Suppose there is a variable num and you want to display “hi” on your webpage if it is greater than 5 otherwise you wanna display a message. Consider the below code for this scenario –

**If you wanna write a code in java for above situation then it would look like this –**

[code language=”java”]

if (num > 5)

{

out.println("hi");

}

else

{

out.println("num value should not be less than 6");

}[/code]

**To write the similar code in JSP we need to use JSP scriptlets – Code would be like this –**

[code language=”html”]

<! DOCTYPE HTML PUBLIC "-//W3C//DTD HTML4.0 translation //EN">

<HTML>

<HEAD>

<TITLE> MY JSP PAGE </TITLE>

</HEAD>

<BODY>

<% if (num > 5) { %>

<H3> hi </H3>

<%} else {%>

<h3> num value should not be less than 6 </h3>

<% } %>

</BODY>

</HTML>

[/code]

**Important Point to remember:**Since the code inside it is a java code it **must end with a semicolon(;).**Now notice all the statements – you may find that ~~all~~ few scriptlet where we give semicolon in java, needs it here too and ends with a semicolon.

Bottom of Form

Top of Form

**JSP Directives – Page, Include and TagLib**

Directives control the processing of an entire JSP page. It gives directions to the server regarding processing of a page.

**Syntax of Directives:**

<%@ directive name [attribute name=“value” attribute name=“value” ........]%>

**There are three types of Directives in JSP:**

1) Page Directive

2) Include Directive

3) TagLib Directive

**1) Page Directive**

There are several attributes, which are used along with Page Directives and these are –

1. import
2. session
3. isErrorPage
4. errorPage
5. ContentType
6. isThreadSafe
7. extends
8. info
9. language
10. autoflush
11. buffer

**1. import:**

This attribute is used to import packages. While doing coding you may need to include more than one packages, In such scenarios this page directive’s attribute is very useful as it allows you to mention more than one packages at the same place separated by commas (,). Alternatively you can have multiple instances of page element each one with different package.

**Syntax of import attribute –**

<%@page import="value"%>

Here value is package name.

**Example of import-**The following is an example of how to import more than one package using import attribute of page directive.

<%@page import="java.io.\*%>

<%@page import="java.lang.\*%>

<%--Comment: OR Below Statement: Both are Same--%>

<%@page import="java.io.\*, java.lang.\*"%>

**2. session:**

Generally while building a user interactive JSP application, we make sure to give access to the user to get hold of his/her personal data till the session is active. Consider an example of logging in into your bank account, we can access all of your data till we signout (or session expires). In order to maintain session for a page the session attribute should be true.

This attribute is to handle HTTP sessions for JSP pages. It can have two values: true or false. Default value for session attribute is true, which means if you do not mention this attribute, server may assume that HTTP session is required for this page.

**Default value for this attribute:**true

**Syntax of session attribute:**

<%@ page session="value"%>

here value is **either true OR false**

**Examples of session:**

<%@ page session="true"%>

The above code would allow a page to have session implicit objects.

<%@ page session="false"%>

If this code is specified in a JSP page, it means session objects will not be available for that page. Hence session cannot be maintained for that page.

**3. isErrorPage:**

This attribute is used to specify whether the current JSP page can be used as an error page for another JSP page. If value of isErrorPage is true it means that the page can be used for exception handling for another page. Generally these pages has error/warning messages OR exception handling codes and being called by another JSP page when there is an exception occurred there.

There is another use of isErrorPage attribute – The [**exception implicit object**](https://beginnersbook.com/2013/11/jsp-implicit-object-exception-with-examples/) can only be available to those pages which has isErrorPage set to true. If the value is false, the page cannot use exception implicit object.

**Default value:**false

**Syntax of isErrorPage attribute:**

<%@ page isErrorPage="value"%>

Here value is either true OR false.

**Example of isErrorPage:**

<%@ page isErrorPage="true"%>

This makes a JSP page, a exception handling page.

**4. errorPage:**

As I stated above, when isErrorPage attribute is true for a particular page then it means that the page can be called by another page in case of an exception. errorPage attribute is used to specify the URL of a JSP page which has isErrorPage attrbute set to true. It handles the un-handled exceptions in the page.

**Syntax of errorPage attribute:**

<%@ page errorPage="value"%>

Here value is a JSP page name which has exception handling code (and isErrorPage set to true).

**Example of errorPage:**

<%@ page errorPage="ExceptionHandling.jsp"%>

This means if any exception occurs on the JSP page where this code has been placed, the ExceptionHandling.jsp (this page should have isErrorPage true) page needs to be called.

**5. contentType:**

This attribute is used to set the content type of a JSP page.

**Default value**: text/html

**Syntax of contentType attribute:**

<%@ page contentType="value"%>

here value of content type can be anything such as: text/html, text/xml etc.

**Example of contentType:**

Below code can be used for text/html pages.

<%@ page contentType="text/html"%>

for text/xml based pages:

<%@ page contentType="text/xml"%>

**6. isThreadSafe:**

Lets understand this with an example. Suppose you have created a JSP page and mentioned isThreadSafe as true, it means that the JSP page supports multithreading (more than one thread can execute the JSP page simultaneously). On the other hand if it is set to false then JSP engine won’t allow multithreading which means only single thread will execute the page code.

**Default value for isThreadSafe attribute:** true.

**Syntax of isThreadSafe attribute:**

<%@ page isThreadSafe="value"%>

here value can be true OR false.

**Example of isThreadSafe:**

<%@ page isThreadSafe="false"%>

Only one thread will be responsible for JSP page execution.

**7. buffer:**

This attribute is used to specify the buffer size. If you specify this to none during coding then the output would directly written to Response object by JSPWriter. And, if you specify a buffer size then the output first written to buffer then it will be available for response object.

**Syntax of buffer attribute:**

<%@ page buffer="value"%>

value is **size in kb** or **none**.

**Example of buffer:**

No buffer for this page:

<%@ page buffer="none"%>

5 kb buffer size for the page, which has below code:

<%@ page buffer="5kb"%>

**8. extends:**

Like java, here also this attribute is used to extend(inherit) the class.

**Syntax of extends attribute:**

<%@ page extends="value"%>

Value is package\_name.class\_name.

**Example of extends:**

The below code will inherit the SampleClass from package: mypackage

<%@ page extends="mypackage.SampleClass"%>

**9. info:**

It provides a description to a JSP page. The string specified in info will return when we will call getServletInfo() method.

**Syntax of info:**

<%@ page info="value"%>

here value is Message or Description

**Example of info attribute:**

<%@ page info="This code is given by Chaitanya Singh"%>

**10. language:**

It specifies the scripting language( underlying language) being used in the page.

**Syntax of language:**

<%@ page language="value"%>

value is scripting language here.

**Example of language attribute:**

<%@ page language="java"%>

**11. autoFlush:**

If it is true it means the buffer should be flushed whenever it is full. false will throw an exception when buffer overflows.

**Default value**: True

**Syntax of autoFlush:**

<%@ page autoFlush="value"%>

value can be true or false.

**Example of autoFlush attribute:**

Buffer will be flushed out when it is full –

<%@ page autoFlush="true"%>

It will throw an exception when buffer is full due to overflow condition

<%@ page autoFlush="true"%>

**12. isScriptingEnabled:**

It has been dropped and not in use.

**13. isELIgnored:**

This attribute specify whether expressions will be evaluated or not.

**Default value**: true

**Syntax of isELIgnored:**

<%@ page isELIgnored="value"%>

value can be true or false.

**Example of isELIgnored attribute:**

Any expression present inside JSP page will not be evaluated –

<%@ page isELIgnored="false"%>

Expression will be evaluated (true is a default value so no need to specify)-

<%@ page isELIgnored="true"%>

**2) Include Directive**

Include directive is used to copy the content of one JSP page to another. It’s like including the code of one file into another.

**Syntax of Include Directive:**

<%@include file ="value"%>

here value is the JSP file name which needs to be included. If the file is in the same directory then just specify the file name otherwise complete URL(or path) needs to be mentioned in the value field.

**Note: It can be used anywhere in the page.**

**Example:**

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

You can use the above code in your JSP page to copy the content of myJSP.jsp file. However in this case both the JSP files must be in the same directory. If the myJSP.jsp is in the different directory then instead of just file name you would need to specify the complete path in above code.

**3) Taglib Directive**

This directive basically allows user to use Custom tags in JSP. we shall discuss about Custom tags in detail in coming JSP tutorials. Taglib directive helps you to declare custom tags in JSP page.

**Syntax of Taglib Directive:**

<%@taglib uri ="taglibURI" prefix="tag prefix"%>

Where URI is uniform resource locator, which is used to identify the location of custom tag and tag prefix is a string which can identify the custom tag in the location identified by uri.

**Example of Targlib:**

<%@ taglib uri="http://www.sample.com/mycustomlib" prefix="demotag" %>

<html>

<body>

<demotag:welcome/>

</body>

</html>

As you can see that uri is having the location of custom tag library and prefix is identifying the prefix of custom tag.

Note: In above example – <demotag: welcome> has a prefix demotag.

Bottom of Form

Top of Form

**Video Lectures :**

**Reference Links:**

https://beginnersbook.com/2013/05/jsp-tutorial-directives/

https://docs.oracle.com/javaee/5/tutorial/doc/bnaou.html

https://beginnersbook.com/2013/05/jsp-tutorial-scriptlets/

https://www.javatpoint.com/jsp-scriptlet-tag

https://beginnersbook.com/2013/05/jsp-tutorial-directives/

https://beginnersbook.com/2013/11/jsp-declaration-tag/

Bottom of Form