http://www.javatpoint.com/jsp

Directive options in jsp

Defines attributes that apply to an entire JSP page.

#### JSP Syntax

<%@ page   
[ language="**java**" ]   
[ extends="package.class" ]   
[ import="{package.class | package.\*}, ..." ]   
[ session="**true** | false" ]   
[ buffer="none | **8kb** | sizekb" ]   
[ autoFlush="**true** | false" ]   
[ isThreadSafe="**true** | false" ]   
[ info="text" ]   
[ errorPage="relativeURL" ]   
[ contentType="mimeType [ ;charset=characterSet ]" | "**text/html ; charset=ISO-8859-1**" ]   
[ isErrorPage="true | **false**" ]   
%>

#### Examples

<%@ page import="java.util.\*, java.lang.\*" %>   
<%@ page buffer="5kb" autoFlush="false" %>   
<%@ page errorPage="error.jsp" %>

#### Description

The <%@ page %> directive applies to an entire JSP file and any of its static include files, which together are called a translation unit. A static include file is a file whose content becomes part of the calling JSP file. The <%@ page %> directive does not apply to any dynamic include files; see [<jsp:include>](http://java.sun.com/products/jsp/tags/11/syntaxref11.fm11.html#8828) for more information.

You can use the <%@ page %> directive more than once in a translation unit, but you can only use each attribute, except import, once. Because the import attribute is similar to the import statement in the Java programming language, you can use a <%@ page %> directive with import more than once in a JSP file or translation unit.

No matter where you position the <%@ page %> directive in a JSP file or included files, it applies to the entire translation unit. However, it is often good programming style to place it at the top of the JSP file.

#### Attributes

* language="**java**"

The scripting language used in scriptlets, declarations, and expressions in the JSP file and any included files. In this release, the only allowed value is java.

* extends="package.class"

The fully qualified name of the superclass of the Java class file this JSP file will be compiled to. Use this attribute cautiously, as it can limit the JSP container's ability to provide a specialized superclass that improves the quality of the compiled file.

* import="{package.class | package.\* }, ..."

A comma-separated list of Java packages that the JSP file should import. The packages (and their classes) are available to scriptlets, expressions, and declarations within the JSP file. If you want to import more than one package, you can specify a comma-separated list after import or you can use import more than once in a JSP file.

The following packages are implicitly imported, so you don't need to specify them with the import attribute:

java.lang.\*   
javax.servlet.\*   
javax.servlet.jsp.\*   
javax.servlet.http.\*

You must place the import attribute before the element that calls the imported class.

* session="**true** | false"

Whether the client must join an HTTP session in order to use the JSP page. If the value is true, the session object refers to the current or new session.

If the value is false, you cannot use the session object or a <jsp:useBean> element with scope=session in the JSP file. Either of these usages would cause a translation-time error.

The default value is true.

* buffer="none | **8kb** | sizekb"

The buffer size in kilobytes used by the out object to handle output sent from the compiled JSP page to the client Web browser. The default value is 8kb. If you specify a buffer size, the output is buffered with at least the size you specified.

* autoFlush="**true** | false"

Whether the buffered output should be flushed automatically when the buffer is full. If set to true (the default value), the buffer will be flushed. If set to false, an exception will be raised when the buffer overflows. You cannot set autoFlush to false when buffer is set to none.

* isThreadSafe="**true** | false"

Whether thread safety is implemented in the JSP file. The default value is true, which means that the JSP container can send multiple, concurrent client requests to the JSP page. You must write code in the JSP page to synchronize the multiple client threads. If you use false, the JSP container sends client requests one at a time to the JSP page.

* info="text"

A text string that is incorporated verbatim into the compiled JSP page. You can later retrieve the string with the Servlet.getServletInfo() method.

* errorPage="relativeURL"

A pathname to a JSP file that this JSP file sends exceptions to. If the pathname begins with a /, the path is relative to the JSP application's document root directory and is resolved by the Web server. If not, the pathname is relative to the current JSP file.

* isErrorPage="true | **false**"

Whether the JSP file displays an error page. If set to true, you can use the exception object in the JSP file. If set to false (the default value), you cannot use the exception object in the JSP file.

* contentType="mimeType [ ;charset=characterSet ]" | "**text/html;charset=ISO-8859-1**"

The MIME type and character encoding the JSP file uses for the response it sends to the client. You can use any MIME type or character set that are valid for the JSP container. The default MIME type is text/html, and the default character set is ISO-8859-1.

## <jsp:useBean>

Locates or instantiates a Bean with a specific name and scope.

#### JSP Syntax

<jsp:useBean   
id="beanInstanceName"   
scope="**page** | request | session | application"   
{   
class="package.class" |   
type="package.class" |   
class="package.class" type="package.class" |   
beanName="{package.class | <%= expression %>}" type="package.class"   
}   
{   
/> |   
> other elements </jsp:useBean>   
}

#### Examples

<jsp:useBean id="cart" scope="session" class="session.Carts" />   
<jsp:setProperty name="cart" property="\*" />

<jsp:useBean id="checking" scope="session" class="bank.Checking" >   
<jsp:setProperty name="checking" property="balance" value="0.0" />   
</jsp:useBean>

#### Description

The <jsp:useBean> element locates or instantiates a JavaBeans component. <jsp:useBean> first attempts to locate an instance of the Bean. If the Bean does not exist, <jsp:useBean> instantiates it from a class or serialized template.

To locate or instantiate the Bean, <jsp:useBean> takes the following steps, in this order:

1. Attempts to locate a Bean with the scope and name you specify.
2. Defines an object reference variable with the name you specify.
3. If it finds the Bean, stores a reference to it in the variable. If you specified type, gives the Bean that type.
4. If it does not find the Bean, instantiates it from the class you specify, storing a reference to it in the new variable. If the class name represents a serialized template, the Bean is instantiated by java.beans.Beans.instantiate.
5. If <jsp:useBean> has instantiated (rather than located) the Bean, and if it has body tags or elements (between <jsp:useBean> and </jsp:useBean>), executes the body tags.

The body of a <jsp:useBean> element often contains a <jsp:setProperty> element that sets property values in the Bean. As described in [Step 5](http://java.sun.com/products/jsp/tags/11/syntaxref11.fm14.html#9843), the body tags are only processed if <jsp:useBean> instantiates the Bean. If the Bean already exists and <jsp:useBean> locates it, the body tags have no effect.

In this release, you can use a <jsp:useBean> element to locate or instantiate a Bean, but not an enterprise bean. To create enterprise beans, you can write a <jsp:useBean> element that calls a Bean that in turn calls the enterprise bean, or you can write a custom tag that calls an enterprise bean directly.

#### Attributes and Usage

* id="beanInstanceName"

A variable that identifies the Bean in the scope you specify. You can use the variable name in expressions or scriptlets in the JSP file.

The name is case sensitive and must conform to the naming conventions of the scripting language used in the JSP page. If you use the Java programming language, the conventions in the Java Language Specification. If the Bean has already been created by another <jsp:useBean> element, the value of id must match the value of id used in the original <jsp:useBean> element.

* scope="**page** | request | session | application"

The scope in which the Bean exists and the variable named in id is available. The default value is page. The meanings of the different scopes are shown below:

* + page - You can use the Bean within the JSP page with the <jsp:useBean> element or any of the page's static include files, until the page sends a response back to the client or forwards a request to another file.
  + request - You can use the Bean from any JSP page processing the same request, until a JSP page sends a response to the client or forwards the request to another file. You can use the request object to access the Bean, for example, request.getAttribute(beanInstanceName).
  + session - You can use the Bean from any JSP page in the same session as the JSP page that created the Bean. The Bean exists across the entire session, and any page that participates in the session can use it. The page in which you create the Bean must have a <%@ page %> directive with session=true.
  + application - You can use the Bean from any JSP page in the same application as the JSP page that created the Bean. The Bean exists across an entire JSP application, and any page in the application can use the Bean.
* class="package.class"

Instantiates a Bean from a class, using the new keyword and the class constructor. The class must not be abstract and must have a public, no-argument constructor. The package and class name are case sensitive.

* type="package.class"

If the Bean already exists in the scope, gives the Bean a data type other than the class from which it was instantiated. If you use type without class or beanName, no Bean is instantiated. The package and class name are case sensitive.

* class="package.class" type="package.class"

Instantiates a Bean from the class named in class and assigns the Bean the data type you specify in type. The value of type can be the same as class, a superclass of class, or an interface implemented by class.

The class you specify in class must not be abstract and must have a public, no-argument constructor. The package and class names you use with both class and type are case sensitive.

* beanName="{package.class | <%= expression %>}" type="package.class"

Instantiates a Bean from either a class or a serialized template, using the java.beans.Beans.instantiate method, and gives the Bean the type specified in type. The Beans.instantiate method checks whether a name represents a class or a serialized template. If the Bean is serialized, Beans.instantiate reads the serialized form (with a name like package.class.ser) using a class loader. For more information, see the JavaBeans API Specification.

The value of beanName is either a package and class name or an [Expression](http://java.sun.com/products/jsp/tags/11/syntaxref11.fm4.html#11258) that evaluates to a package and class name, and is passed to Beans.instantiate. The value of type can be the same as beanName, a superclass of beanName, or an interface implemented by beanName.

The package and class names you use with both beanName and type are case sensitive.

#### See Also

* [<jsp:setProperty>](http://java.sun.com/products/jsp/tags/11/syntaxref11.fm13.html" \l "8856)
* [<jsp:getProperty>](http://java.sun.com/products/jsp/tags/11/syntaxref11.fm10.html#8820)
* Javadoc API reference for java.beans.Beans
* JavaBeans API Specification

There are six JSP Actions:

<jsp:include/>

<jsp:forward/>

<jsp:plugin/>

<jsp:usebean/>

<jsp:setProperty/>

<jsp:getProperty/>

**Question:** What is the difference between <jsp:include page = ... > and   
<%@ include file = ... >?.  
**Answer:** Both the tag includes the information from one page in another. The differences are as follows:  
**<jsp:include page = ... >:** This is like a function call from one jsp to another jsp. It is executed ( the included page is executed and the generated html content is included in the content of calling jsp) each time the client page is accessed by the client. This approach is useful to for modularizing the web application. If the included file changed then the new content will be included in the output.   
  
**<%@ include file = ... >**: In this case the content of the included file is textually embedded in the page that have <%@ include file=".."> directive. In this case in the included file changes, the changed content will not included in the output. This approach is used when the code from one jsp file required to include in multiple jsp files.

**Question:** What is the difference between <jsp:forward page = ... > and   
response.sendRedirect(url),?.  
**Answer:**The <jsp:forward> element forwards the request object containing the client request information from one JSP file to another file. The target file can be an HTML file, another JSP file, or a servlet, as long as it is in the same application context as the forwarding JSP file.   
***sendRedirect*** sends HTTP temporary redirect response to the browser, and browser creates a new request to go the redirected page. The response.sendRedirect kills the session variables.

**Question:** Identify the advantages of JSP over Servlet.  
  
a) Embedding of Java code in HTML pages  
b) Platform independence  
c) Creation of database-driven Web applications  
d) Server-side programming capabilities  
  
**Answer :- Embedding of Java code in HTML pages**

Write the following code for a JSP page:  
<%@ page language = "java" %>   
  
<HTML>  
<HEAD><TITLE>RESULT PAGE</TITLE></HEAD>  
<BODY>  
<%  
  
PrintWriter print = request.getWriter();  
print.println("Welcome");  
  
%>  
</BODY>  
</HTML>  
Suppose you access this JSP file, Find out your answer.  
a) A blank page will be displayed.  
b) A page with the text Welcome is displayed  
c) An exception will be thrown because the implicit out object is not used  
d) An exception will be thrown because PrintWriter can be used in servlets only  
  
**Answer :- A page with the text Welcome is displayed**

**Question:** What are implicit Objects available to the JSP Page?  
**Answer:** Implicit objects are the objects available to the JSP page. These objects are created by Web container and contain information related to a particular request, page, or application. The JSP implicit objects are:

|  |  |  |
| --- | --- | --- |
| **Variable** | **Class** | **Description** |
| Application | javax.servlet.ServletContext | The context for the JSP page's servlet and any Web components contained in the same application. |
| Config | javax.servlet.ServletConfig | Initialization information for the JSP page's servlet. |
| Exception | java.lang.Throwable | Accessible only from an error page. |
| Out | javax.servlet.jsp.JspWriter | The output stream. |
| Page | java.lang.Object | The instance of the JSP page's servlet processing the current request. Not typically used by JSP page authors. |
| pageContext | javax.servlet.jsp.PageContext | The context for the JSP page. Provides a single API to manage the various scoped attributes. |
| Request | Subtype of javax.servlet.ServletRequest | The request triggering the execution of the JSP page. |
| Response | Subtype of javax.servlet.ServletResponse | The response to be returned to the client. Not typically used by JSP page authors. |
| Session | javax.servlet.http.HttpSession | The session object for the client. |

**Question:** What are all the different scope values for the <jsp:useBean> tag?  
**Answer:**<jsp:useBean> tag is used to use any java object in the jsp page. Here are the scope values for <jsp:useBean> tag:  
a) page  
b) request  
c) session and  
d) application

**Question:** What is JSP Output Comments?  
**Answer:** JSP Output Comments are the comments that can be viewed in the HTML source file.  
Example:   
<!-- This file displays the user login screen -->  
and   
<!-- This page was loaded on  
<%= (new java.util.Date()).toLocaleString() %> -->

**Question:** What is expression in JSP?  
**Answer:** Expression tag is used to insert Java values directly into the output. Syntax for the Expression tag is:   
<%= expression %>  
An expression tag contains a scripting language expression that is evaluated, converted to a String, and inserted where the expression appears in the JSP file. The following expression tag displays time on the output:  
<%=new java.util.Date()%>

**Question:** What types of comments are available in the JSP?  
**Answer:** There are two types of comments are allowed in the JSP. These are *hidden* and *output* comments. A hidden comments does not appear in the generated output in the html, while output comments appear in the generated output.  
Example of hidden comment:  
<%-- This is hidden comment --%>  
Example of output comment:  
<!-- This is output comment -->

**Question:** What is JSP declaration?  
**Answer:** JSP Decleratives are the JSP tag used to declare variables. Declaratives are enclosed in the <%! %> tag and ends in semi-colon. You declare variables and functions in the declaration tag and can use anywhere in the JSP. Here is the example of declaratives:

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

<html>

<body>

<%!  
int cnt=0;  
private int getCount(){  
//increment cnt and return the value  
cnt++;  
return cnt;  
}  
%>

<p>Values of Cnt are:</p>

<p><%=getCount()%></p>

</body>

</html>

**Question:** What is JSP Scriptlet?  
**Answer:** JSP Scriptlet is jsp tag which is used to enclose java code in the JSP pages. Scriptlets begins with **<%** tag and ends with **%>** tag. Java code written inside scriptlet executes every time the JSP is invoked.   
Example:  
<%  
//java codes  
String userName=null;  
userName=request.getParameter("userName");  
%>

**Question:** What are the life-cycle methods of JSP?  
**Answer:** Life-cycle methods of the JSP are:  
a) **jspInit()**: The container calls the jspInit() to initialize the servlet instance. It is called before any other method, and is called only once for a servlet instance.  
b)**\_jspService():** The container calls the \_jspservice() for each request and it passes the request and the response objects. \_jspService() method cann't be overridden.   
c) **jspDestroy():** The container calls this when its instance is about to destroyed.  
The jspInit() and jspDestroy() methods can be overridden within a JSP page.

Which of the following is server side programming languages?

1.HTML

2.JavaScript

3.JSP

4.Sevlets

A.1,2,3,4

B.1,2

C 3,4

D.None

What JSP stand for

1.Java Server Pages

2.Java Server Programming

3.Java Service Pages

4.Java Service Programming

What JSP is basically used today

1.To make controller

2.To make GUI

3.In writing business logic

4.Deprecated

5.None

We must have configure and map all jsp files into web.xml

1.YES

2.NO

JSP is part of BEA WebLogic?

1.Yes

2.No

JSP is part of Apache Tomcat?

1.Yes

2.No

JSP is part of Apache Tomcat?

1.Yes

2.No

JSP is part of Apache Tomcat?

1.Yes

2.No

JSP can run by

1.Apache Tomact

2.Bea WebLogic

3.Servlets

4.JSP Engine

5.ALL

6.None

IBM WebSphere is?

1. A full-featured Web Application Server

2 It provides supports to the Servlet and JSP

3. It includes pooled database access for DB2 and Oracle Relational Databases,

4. Prtovide support for EJB .

Choose correct one

A. 1,2,3,4

B. 2,3,4

C. 1,3,4

D. None

Syntax of Scriptlet Element

1.<% code %>

2. <@jsp code %>

3. <jsp: scriptlet code />

4 <%jsp code %>

Three scripting elements of JSP

1.Scriptlets

2.Expressions

3.Declarations

4.None

Syntax of Declaration Element

1.<%! code !%>

2.<%! code %>

3.<jsp: scriptlet code />

4.<%jsp code !%>

Syntax of comments

1 <%-- comment --%>

2.<!-- comment -->

3.<% // comment %>

4.<% /\* comment \*/ %>

5.<% /\*\* comment \*/ %>

Choose correct scopes into JSP:-

1.page, request, session, application

2.Page, response, session, application

3.Page, request, response, session

4.None

Choose JavaBean Actions:-

1.useBean

2.setProperty

3.getProperty

4.All

5.None

In <jsp:useBean> which two attributes is necessary

1.id and scope

2.class or beanName.

3.type and id

4 None

Correct syntax of useBean Action

1.<jsp:useBean id=" " class=" "> </jsp:useBean>

2.<jsp:useBean id=" " property= =" "> </jsp:useBean>

3.<jsp:useBean name=="newBean" class=" "> </jsp:useBean>

4.<jsp:useBean beanName =" " class=" "> </jsp:useBean>

Correct syntax of setProperty Action

1.

<jsp:useBean id="login" class="r4r.co.in.R4RLogin">

<jsp:setProperty name="login" property="username" value="rajesh">

</jsp:useBean>

2.

<jsp:useBean id="login" class="r4r.co.in.R4RLogin">

</jsp:useBean>

<jsp:setProperty name="login" property="username" value="rajesh">

3 Both

4.None

In <jsp:getProperty /> which two attributes is necessary

1.id and scope

2.class or beanName,name,property,value.

3.type and id

4.name and property

5.None

In <jsp:include> which attributes are used

1.id and scope

2.type and id

3.file and flush

4 page and flush

Choose correct syntax to pass parameter into jsp include action

1.

<jsp:include page="/login.jsp" flush="yes">

<jsp:param name="user" value="rajesh" />

</jsp:include>

2.

<jsp:include page="/login.jsp" flush="yes" param=”user” value=”rajesh”>

</jsp:include>

3.

<jsp:include page="/login.jsp" flush="yes" property=”user” value=”rajesh”>

</jsp:include>

4.All of the above.

5.None

MORE …………….Questions……………………………..

1. Which of these are the methods of lifecycle of servlets.

A init() B start C service d.destroy e all these.

2 can init() be overridden -----true / false

3 What paramter does overloaded init() method take.

4 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_represents an interface for the entire context of the web application.

A Servlet config

B Servletcontext

C DeploymentDescriptor

D ApplicationContext.

5 Number of web.xml that can exist in a web application.

A one for every servlet

B one for every web application

C any number .

6 Which of the following are valid elements of web.xml.

init-param , servlet ,servlet-mapping, welcome-file-list,listener, context-param,application-param,

url-mapping,.

7 The process of loading a servlet before any request comes in is called preloading. (true/ false)

8 Servlet Container initializes a servlet when it receives a request for that servlet first time. This is called \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (early loading / lazy loading)

9 Which of these are not Http methods that can be defined in servlet.

A doput()

B doget()

C doHead()

D none

10 Query string is displayed as part of url in which of these methods. doGet() and doPost().

11 Tick all the jsp implicit objects.

* response
* pageContext
* session
* application
* out
* pagecontext
* writer
* context
* config
* page
* exception
* request

12 What is the difference between request.sendRedirect() and forward().

13 What is the default scope for jsp:useBean.

14 What is the difference between <%! Int x=10; %> and <% int x=12; %>

15 Which package provides interfaces and classes for writing servlets?

16 When a httpservlet accepts a call from a client, it receives two objects. What are they?

17 Which of these are classes and which are interfaces.

A ServletContext

B GenericServlet

C HttpServlet

D HttpSession

E HttpRequest

F HttpResponse

G RequestDispatcher

18 getInitParameter() is used for accessing which of these element values.

A init-param

B context-param

C both

19 Servlets are loaded based on single threaded model. True/false

20 Which are the various ways of session tracking.

21 What is basic difference between include file=”demo.jsp” and jsp:include.

22 Which of the following are reasons for session closing.

A calling invalidate()

B timeout

C power shut down.

D Closing of browser

23 setMaxInactiveInterval() is a method to set \_\_\_\_\_\_\_\_

24 in setMaxInactiveIntervals(3000), 3000 is in \_\_\_\_\_\_\_\_\_\_\_ unit.

25 Which of the following methods creates a new session,if the session is not present.

A request.getSession()

B request.getSession(true)

C request.getSession(false)

D All the above

26 Which interfaces define getSession() method

A ServletRequest

B ServletResponse

C HttpServletRequest

D HttpServletResponse

27 What is the signature of setAttribute().

A setAttribute(String s,Object o)

B setAttribute(Object a,Object b)

C setAttribute(String s,String p)

D setAttribute(Object o,String s)

28 getAttribute() returns a value of type\_\_\_\_\_

29 Which JSP expression tag will print the context initialization parameter named “javax.sql.DataSource”?

A <%= application.getAttribute(“javax.sql.DataSource”) %>

B <%= application.getInitParameter(“javax.sql.DataSource”) %>

C <%= request.getParameter(“javax.sql.DataSource”) %>

D <%= contextParam.getAttribute(“javax.sql.DataSource”) %>

30 Which JSP lifecycle step is out of order?

A Translate The JSP into java source code

B compile source code and create a class

C call \_jspService()

D Instantiate the servlet class

E call jspInit()

F call jspDestroy()

G load and initialize as a servlet

31 Which of the following methods has a limit to the size of data.(GET/POST)

32 What will request.getParameter(“name”) return if name is not a valid field in the request.

A throwds NullPointerException

B prints null

C no output

D compilation error.

33

In which of the folders is web.xml stored during deployment of the web application.

A lib

B WEB-INF

C classes

D ROOT

34

Which of the following are valid expression tags in jsp

A <%= x+y; %>

B <%= x+y %>

C <% = out.println(x+y) %>

35

Which of the following attribute elements are not valid in useBean tag.

A id

B class

C scope

D property

1. With which object you call getParameter(“username”) inside a servlet/jsp to get value from a query string.

A response

B request

C session

D all the above

2 If the request url is:

<http://localhost:8080/webdemos/MyServlet?name=sam>

inside the servlet the statement request.getParameter(“username”)

prints \_\_\_\_\_\_\_\_\_\_\_\_

a null

b sam

c error

d username

3 in html <form action=”h1”></form>

Executes which of the following methods

A doPost

B doGet

C doTrace

D doDelete

4 Which of the following are not HTTP methods.

A trace

B delete

C options

D none

5

<% int x=10; %> ------1

<% ++x; %> -------2

<%= x%> ------------3

What is the output if the above code in a jsp is executed thrice.

A 11 , 11, 11

B 11,12,13

C x is not accessible in statements 2 and 3

D Statement 3 has error, there should be a semicolon.

6

<%! int x=10; %> ------1

<% ++x; %> -------2

<%= x%> ------------3

What is the output if the above code in a jsp is executed thrice.

A 11 , 11, 11

B 11,12,13

C x is not accessible in statements 2 and 3

D Statement 3 has error, there should be a semicolon.

7

<% int x=12; %>

<%! int f1(){

out.println(x);

%>

Output: ?

A 12

B error because x is not accessible

C error because out is not accessible

D error because out not declared

8

<% int x=12;

out.println(x);

%> gets translated to.

A public void \_jspService(some code){

int x=12;

PrintWriter out=response.getWriter();

out.print(x);}

B int x=12;

public void \_jspService(some code){

PrintWriter out=response.getWriter();

out.print(x);}

C

int x=12;

public void jspInit(){

PrintWriter out=response.getWriter();

out.print(x);}

}