# JSP (JAVA SERVER PAGES)

JSP technology is used to create web application just like Servlet technology. It can be thought of as an extension to servlet because it provides more functionality than servlet such as expression language, jstl etc. A JSP page consists of HTML tags and JSP tags. The jsp pages are easier to maintain than servlet because we can separate designing and development. It provides some additional features such as Expression Language, Custom Tag etc.

## ADVANTAGE OF JSP OVER SERVLET

There are many advantages of JSP over servlet. They are as follows:

### 1) Extension to Servlet

JSP technology is the extension to servlet technology. We can use all the features of servlet in JSP. In addition to, we can use implicit objects, predefined tags, expression language and Custom tags in JSP, that makes JSP development easy.

### 2) Easy to maintain

JSP can be easily managed because we can easily separate our business logic with presentation logic. In servlet technology, we mix our business logic with the presentation logic.

### 3) Fast Development: No need to recompile and redeploy

If JSP page is modified, we don't need to recompile and redeploy the project. The servlet code needs to be updated and recompiled if we have to change the look and feel of the application.

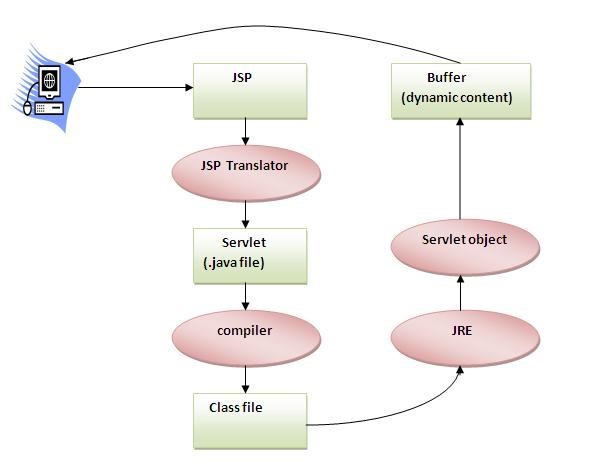
### 4) Less code than Servlet

In JSP, we can use a lot of tags such as action tags, jstl, custom tags etc. that reduces the code. Moreover, we can use EL, implicit objects etc.

## LIFE CYCLE OF A JSP PAGE

The JSP pages follows these phases:

* Translation of JSP Page
* Compilation of JSP Page
* Classloading (class file is loaded by the classloader)
* Instantiation (Object of the Generated Servlet is created).
* Initialization ( jspInit() method is invoked by the container).
* Reqeust processing ( \_jspService() method is invoked by the container).  Destroy ( jspDestroy() method is invoked by the container).



As depicted in the above diagram, JSP page is translated into servlet by the help of JSP translator. The JSP translator is a part of webserver that is responsible to translate the JSP page into servlet. Afterthat Servlet page is compiled by the compiler and gets converted into the class file. Moreover, all the processes that happens in servlet is performed on JSP later like initialization, committing response to the browser and destroy.

## CREATING A SIMPLE JSP PAGE

To create the first jsp page, write some html code as given below, and save it by .jsp extension. We have save this file as index.jsp. Put it in a folder and paste the folder in the web-apps directory in apache tomcat to run the jsp page.

index.jsp

Let's see the simple example of JSP, here we are using the scriptlet tag to put java code in the JSP page. We will learn scriptlet tag later.

<html>

<body>

<% out.print(2\*5); %>

</body>

</html>

It will print 10 on the browser.

### **THE JSP API**

The JSP API consists of two packages:

1. javax.servlet.jsp
2. javax.servlet.jsp.tagext

#### JAVAX.SERVLET.JSP PACKAGE

The javax.servlet.jsp package has two interfaces and classes.The two interfaces are as follows:

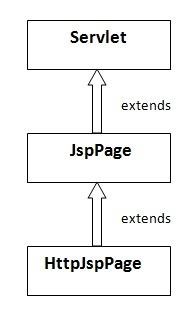
1. JspPage
2. HttpJspPage

The classes are as follows:

* JspWriter
* PageContext
* JspFactory
* JspEngineInfo
* JspException
* JspError

#### THE JSPPAGE INTERFACE

According to the JSP specification, all the generated servlet classes must implement the JspPage interface. It extends the Servlet interface. It provides two life cycle methods.



##### Methods of JspPage interface

1. public void jspInit(): It is invoked only once during the life cycle of the JSP when JSP page is requested firstly. It is used to perform initialization. It is same as the init() method of Servlet interface.
2. public void jspDestroy(): It is invoked only once during the life cycle of the JSP before the JSP page is destroyed. It can be used to perform some clean up operation.

#### THE HTTPJSPPAGE INTERFACE

The HttpJspPage interface provides the one life cycle method of JSP. It extends the JspPage interface.

Method of HttpJspPage interface:

1. public void \_jspService(): It is invoked each time when request for the JSP page comes to the container. It is used to process the request. The underscore \_ signifies that you cannot override this method.

We will learn all other classes and interfaces later.

## JSP SCRIPTLET TAG (SCRIPTING ELEMENTS)

In JSP, java code can be written inside the jsp page using the scriptlet tag. Let's see what are the scripting elements first.

### Scripting elements

The scripting elements provides the ability to insert java code inside the jsp. There are three types of scripting elements:

* scriptlet tag
* expression tag
* declaration tag

### **JSP SCRIPTLET TAG**

A scriptlet tag is used to execute java source code in JSP. Syntax is as follows:

1. <% java source code %>

#### Simple Example of JSP scriptlet tag

In this example, we are displaying a welcome message.

<html>

<body>

<% out.print("welcome to jsp"); %>

</body>

</html>

#### Example of JSP scriptlet tag that prints the user name

In this example, we have created two files index.html and welcome.jsp. The index.html file gets the username from the user and the welcome.jsp file prints the username with the welcome message.

##### index.html

<html>

<body>

<form action="welcome.jsp">

<input type="text" name="uname">

<input type="submit" value="go"><br/>

</form>

</body>

</html>

##### welcome.jsp

<html>

<body>

<%

String name=request.getParameter("uname");

out.print("welcome "+name);

%>

</form>

</body>

</html>

### **JSP EXPRESSION TAG**

The code placed within expression tag is written to the output stream of the response. So you need not write out.print() to write data. It is mainly used to print the values of variable or method.

Syntax of JSP expression tag

1. <%= statement %>

#### Example of JSP expression tag

In this example of jsp expression tag, we are simply displaying a welcome message.

|  |
| --- |
| <html>  <body>  <%= "welcome to jsp" %>  </body>  </html> |

Note: Do not end your statement with semicolon in case of expression tag.

#### Example of JSP expression tag that prints current time

To display the current time, we have used the getTime() method of Calendar class. The getTime() is an instance method of Calendar class, so we have called it after getting the instance of Calendar class by the getInstance() method.

index.jsp

<html>

<body>

Current Time: <%= java.util.Calendar.getInstance().getTime() %>

</body>

</html>

#### Example of JSP expression tag that prints the user name

In this example, we are printing the username using the expression tag. The index.html file gets the username and sends the request to the welcome.jsp file, which displays the username. index.html

|  |
| --- |
| <html>  <body>  <form action="welcome.jsp">  <input type="text" name="uname"><br/>  <input type="submit" value="go">  </form>  </body>  </html> |

##### welcome.jsp

<html>

<body>

<%= "Welcome "+request.getParameter("uname") %>

</form>

</body>

</html>

### **JSP DECLARATION TAG**

The JSP declaration tag is used to declare fields and methods.

The code written inside the jsp declaration tag is placed outside the service() method of auto generated servlet.

So it doesn't get memory at each request.

#### Syntax of JSP declaration tag

The syntax of the declaration tag is as follows:

1. <%! field or method declaration %>

Difference between the jsp scriptlet tag and jsp declaration tag ?

|  |  |
| --- | --- |
| Jsp Scriptlet Tag | Jsp Declaration Tag |
| The jsp scriptlet tag can only declare variables not methods. | The jsp declaration tag can declare variables as well as methods. |
|  |  |
| The declaration of scriptlet tag is placed inside the \_jspService() method. | The declaration of jsp declaration tag is placed outside the \_jspService() method. |

#### Example of JSP declaration tag that declares field

In this example of JSP declaration tag, we are declaring the field and printing the value of the declared field using the jsp expression tag.

##### index.jsp

|  |
| --- |
| 1. <html>  2. <body>  3.   1. <%! int data=50; %> 2. <%= "Value of the variable is:"+data %>   6. |

###### 7. </body>

###### 8. </html>

#### Example of JSP declaration tag that declares method

In this example of JSP declaration tag, we are defining the method which returns the cube of given number and calling this method from the jsp expression tag. But we can also use jsp scriptlet tag to call the declared method.

##### index.jsp

|  |
| --- |
| 1. <html> 2. <body> 3.   1. <%! 2. int cube(int n){ 3. return n\*n\*n\*; 4. } 5. %> 9.   10. <%= "Cube of 3 is:"+cube(3) %>  11.   1. </body> 2. </html> |

### **JSP IMPLICIT OBJECTS**

There are 9 jsp implicit objects. These objects arecreated by the web container that are available to all the jsp pages.

The available implicit objects are out, request, config, session, application etc.

A list of the 9 implicit objects is given below:

|  |  |
| --- | --- |
| Object | Type |
| out | JspWriter |
| request | HttpServletRequest |
|  |  |
| response | HttpServletResponse |
| config | ServletConfig |
| application | ServletContext |
| session | HttpSession |
| pageContext | PageContext |
| page | Object |
| exception | Throwable |

#### 1) out implicit object

For writing any data to the buffer, JSP provides an implicit object named out. It is the object of JspWriter. In case of servlet you need to write:

1. PrintWriter out=response.getWriter();

But in JSP, you don't need to write this code.

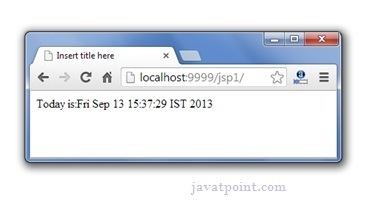
#### Example of out implicit object

In this example we are simply displaying date and time.

##### index.jsp

1. <html>
2. <body>
3. <% out.print("Today is:"+java.util.Calendar.getInstance().getTime()); %>
4. </body>
5. </html>

##### Output



### **JSP DIRECTIVES**

The jsp directives are messages that tells the web container how to translate a JSP page into the corresponding servlet.

There are three types of directives:

* page directive
* include directive
* taglib directive

#### Syntax of JSP Directive

1. <%@ directive attribute="value" %>

#### **JSP PAGE DIRECTIVE**

The page directive defines attributes that apply to an entire JSP page.

Syntax of JSP page directive

1. <%@ page attribute="value" %>

##### Attributes of JSP page directive

* import
* contentType
* extends
* info
* buffer
* language
* isELIgnored
* isThreadSafe
* autoFlush
* session
* pageEncoding
* errorPage
* isErrorPage

##### 1)import

The import attribute is used to import class,interface or all the members of a package.It is similar to import keyword in java class or interface.

##### Example of import attribute

|  |
| --- |
| 1. <html> 2. <body> 3.  4. <%@ page import="java.util.Date" %> 5. Today is: <%= new Date() %> 6.   1. </body> 2. </html> |

###### 2)contentType

The contentType attribute defines the MIME(Multipurpose Internet Mail Extension) type of the HTTP response.The default value is "text/html;charset=ISO-8859-1".

##### Example of contentType attribute

|  |
| --- |
| 1. <html>  2. <body>  3.  4. <%@ page contentType=application/msword %> 5. Today is: <%= new java.util.Date() %>  6.   1. </body> 2. </html> |

###### 3)extends

The extends attribute defines the parent class that will be inherited by the generated servlet.It is rarely used.

###### 4)info

This attribute simply sets the information of the JSP page which is retrieved later by using getServletInfo() method of Servlet interface.

##### Example of info attribute

|  |
| --- |
| 1. <html> 2. <body> 3.  4. <%@ page info="composed by hks" %> 5. Today is: <%= new java.util.Date() %>  6.   1. </body> 2. </html> |

The web container will create a method getServletInfo() in the resulting servlet.For example:

1. public String getServletInfo() {
2. return "composed by Hks"; 3. }

###### 5)buffer

The buffer attribute sets the buffer size in kilobytes to handle output generated by the JSP page.The default size of the buffer is 8Kb.

##### Example of buffer attribute

|  |
| --- |
| 1. <html> 2. <body> 3.   1. <%@ page buffer="16kb" %> 2. Today is: <%= new java.util.Date() %>   6.   1. </body> 2. </html> |

###### 6)language

The language attribute specifies the scripting language used in the JSP page. The default value is "java".

###### 7)isELIgnored

We can ignore the Expression Language (EL) in jsp by the isELIgnored attribute. By default its value is false i.e. Expression Language is enabled by default. We see Expression Language later.

1. <%@ page isELIgnored="true" %>//Now EL will be ignored

###### 8)isThreadSafe

Servlet and JSP both are multithreaded.If you want to control this behaviour of JSP page, you can use isThreadSafe attribute of page directive.The value of isThreadSafe value is true.If you make it false, the web container will serialize the multiple requests, i.e. it will wait until the JSP finishes responding to a request before passing another request to it.If you make the value of isThreadSafe attribute like:

<%@ page isThreadSafe="false" %>

The web container in such a case, will generate the servlet as:

1. public class SimplePage\_jsp extends HttpJspBase
2. implements SingleThreadModel{
3. .......
4. }

###### 9)errorPage

The errorPage attribute is used to define the error page, if exception occurs in the current page, it will be redirected to the error page.

##### Example of errorPage attribute

|  |
| --- |
| 1. //index.jsp 2. <html> 3. <body>   5. <%@ page errorPage="myerrorpage.jsp" %> 6.  7. <%= 100/0 %> 8.   1. </body> 2. </html> |

###### 10)isErrorPage

The isErrorPage attribute is used to declare that the current page is the error page.

*Note: The exception object can only be used in the error page.*

##### Example of isErrorPage attribute

1. //myerrorpage.jsp
2. <html>
3. <body>
4. <%@ page isErrorPage="true" %>
5. Sorry an exception occured!<br/>
6. The exception is: <%= exception %>
7. </body>
8. </html>

**JSP INCLUDE DIRECTIVE**

The include directive is used to include the contents of any resource it may be jsp file, html file or text file. The include directive includes the original content of the included resource at page translation time (the jsp page is translated only once so it will be better to include static resource).

##### Advantage of Include directive

Code Reusability

Syntax of include directive

1. <%@ include file="resourceName" %>

##### Example of include directive

In this example, we are including the content of the header.html file. To run this example you must create an header.html file.

|  |
| --- |
| 1. <html> 2. <body> 3.  4. <%@ include file="header.html" %> 5.  6. Today is: <%= java.util.Calendar.getInstance().getTime() %>  7.   1. </body> 2. </html> |

#### **JSP TAGLIB DIRECTIVE**

The JSP taglib directive is used to define a tag library that defines many tags. We use the TLD (Tag Library Descriptor) file to define the tags. In the custom tag section we will use this tag so it will be better to learn it in custom tag.

##### Syntax JSP Taglib directive

1. <%@ taglib uri="uriofthetaglibrary" prefix="prefixoftaglibrary" %>

##### Example of JSP Taglib directive

In this example, we are using our tag named currentDate. To use this tag we must specify the taglib directive so the container may get information about the tag.

|  |
| --- |
| 1. <html> 2. <body> 3.  4. <%@ taglib uri="http://www.javatpoint.com/tags" prefix="mytag" %> 5.  6. <mytag:currentDate/> 7.   1. </body> 2. </html> |

## 1.8. EXCEPTION HANDLING IN JSP

The exception is normally an object that is thrown at runtime. Exception Handling is the process to handle the runtime errors. There may occur exception any time in your web application. So handling exceptions is a safer side for the web developer. In JSP, there are two ways to perform exception handling:

1. By errorPage and isErrorPage attributes of page directive
2. By <error-page> element in web.xml file

### Example of exception handling in jsp by the elements of page directive

In this case, you must define and create a page to handle the exceptions, as in the error.jsp page. The pages where may occur exception, define the errorPage attribute of page directive, as in the process.jsp page.

There are 3 files:

* index.jsp for input values
* process.jsp for dividing the two numbers and displaying the result
* error.jsp for handling the exception

#### index.jsp

|  |
| --- |
| 1. <form action="process.jsp"> 2. No1:<input type="text" name="n1" /><br/><br/> 3. No1:<input type="text" name="n2" /><br/><br/> 4. <input type="submit" value="divide"/> 5. </form> |

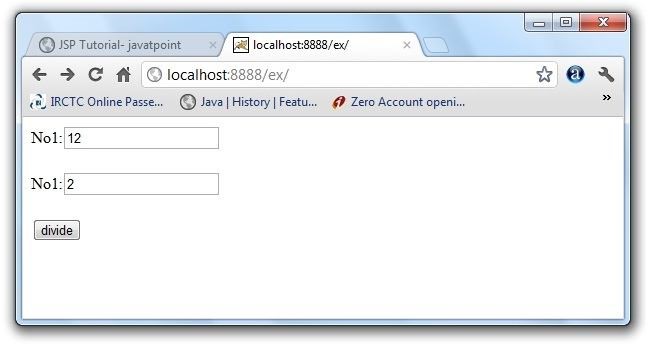
#### process.jsp

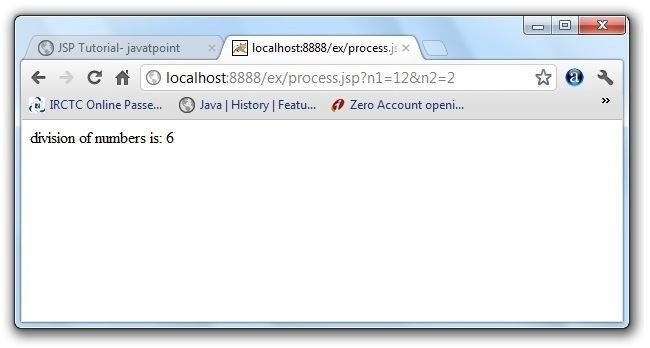
1. <%@ page errorPage="error.jsp" %>
2. <%
3. String num1=request.getParameter("n1");
4. String num2=request.getParameter("n2");
5. int a=Integer.parseInt(num1);
6. int b=Integer.parseInt(num2);
7. int c=a/b;
8. out.print("division of numbers is: "+c); 9. %>

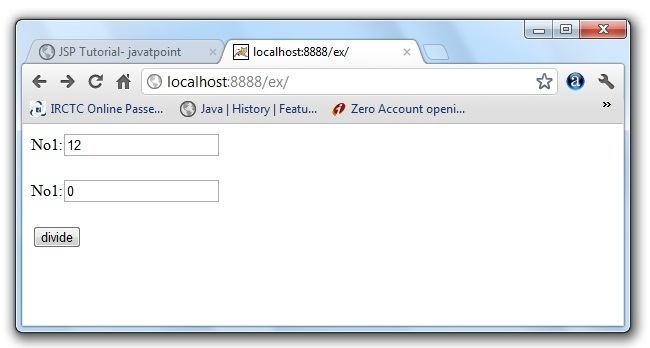
#### error.jsp

1. <%@ page isErrorPage="true" %>
2. <h3>Sorry an exception occured!</h3>
3. Exception is: <%= exception %>

Output of this example:









### Example of exception handling in jsp by specifying the error-page element in web.xml file

This approach is better because you don't need to specify the errorPage attribute in each jsp page. Specifying the single entry in the web.xml file will handle the exception. In this case, either specify exception-type or error-code with the location element. If you want to handle all the exception, you will have to specify the java.lang.Exception in the exception-type element.

Let's see the simple example:

There are 4 files:

* web.xml file for specifying the error-page element
* index.jsp for input values
* process.jsp for dividing the two numbers and displaying the result
* error.jsp for displaying the exception

#### 1) web.xml file if you want to handle any exception

|  |
| --- |
| 1. <web-app> 2.   1. <error-page> 2. <exception-type>java.lang.Exception</exception-type> 5. <location>/error.jsp</location> |

6. </error-page> 7.

8. </web-app>

This approach is better if you want to handle any exception. If you know any specific error code and you want to handle that exception, specify the error-code element instead of exceptiontype as given below:

#### 1) web.xml file if you want to handle the exception for a specific error code

|  |
| --- |
| 1. <web-app> 2.   1. <error-page> 2. <error-code>500</error-code> 3. <location>/error.jsp</location> 6. </error-page> 7.   8. </web-app> |

2) index.jsp file is same as in the above example

#### 3) process.jsp

Now, you don't need to specify the errorPage attribute of page directive in the jsp page.

|  |
| --- |
| 1. <%@ page errorPage="error.jsp" %> 2. <% 3.   1. String num1=request.getParameter("n1"); 2. String num2=request.getParameter("n2"); 6. 3. int a=Integer.parseInt(num1); 4. int b=Integer.parseInt(num2); 5. int c=a/b; 6. out.print("division of numbers is: "+c); 11.   12. %> |