

# JSP tutorial

Written by the team at **Visualbuilder.com**

Last Updated : May 18, 2001

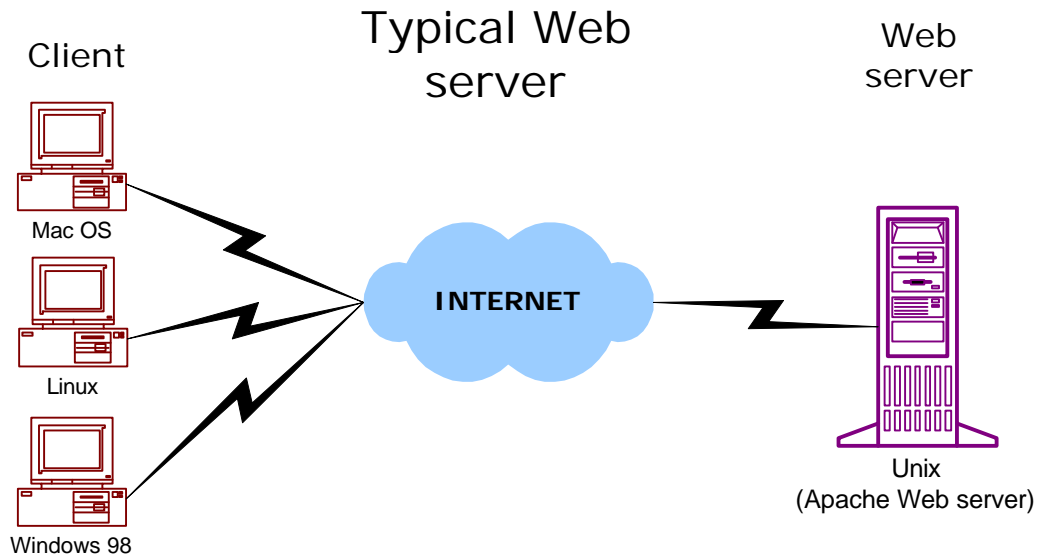
Online version available at <http://visualbuilder.com/>

This is a tutorial to help a developer understand what is meant by JavaServer Pages. The reader will learn the main concepts, setup the JSP environment on their computer and write a few JSP programs.

## Introduction to JSP

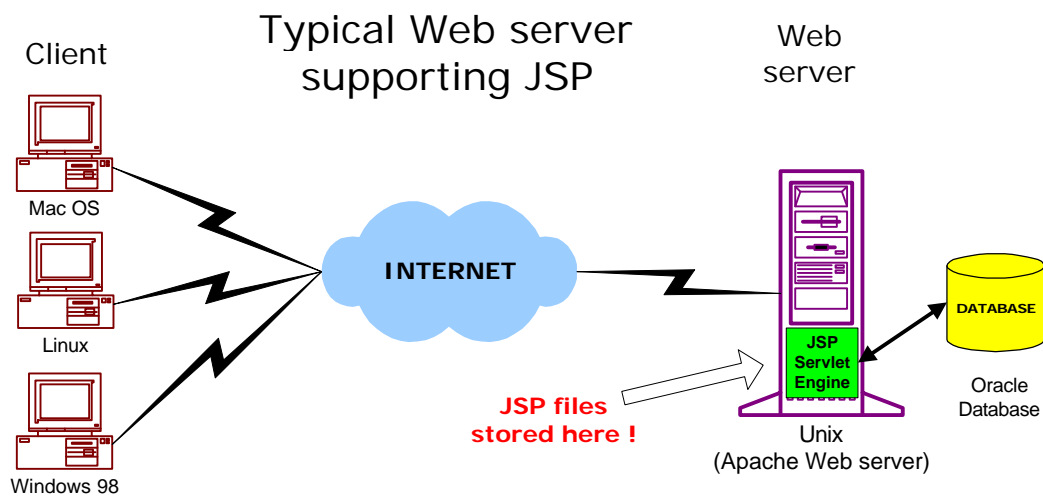
JavaServer Pages (JSP) is a technology based on the Java language and enables the development of dynamic web sites. JSP was developed by Sun Microsystems to allow server side development. JSP files are HTML files with special Tags containing Java source code that provide the dynamic content.

The following shows the Typical Web server, different clients connecting via the Internet to a Web server. In this example, the Web server is running on Unix and is the very popular Apache Web server.



First static web pages were displayed. Typically these were people's first experience with making web pages so consisted of My Home Page sites and company marketing information. Afterwards Perl and C were languages used on the web server to provide dynamic content. Soon most languages including Visualbasic, Delphi, C++ and Java could be used to write applications that provided dynamic content using data from text files or database requests. These were known as CGI server side applications. ASP was developed by Microsoft to allow HTML developers to easily provide dynamic content supported as standard by Microsoft's free Web Server, Internet Information Server (IIS). JSP is the equivalent from Sun Microsystems, a comparison of ASP and JSP will be presented in the following section.

The following diagram shows a web server that supports JSP files. Notice that the web server also is connected to a database.



JSP source code runs on the web server in the JSP Servlet Engine. The JSP Servlet engine dynamically generates the HTML and sends the HTML output to the client's web browser.

## Why use JSP?

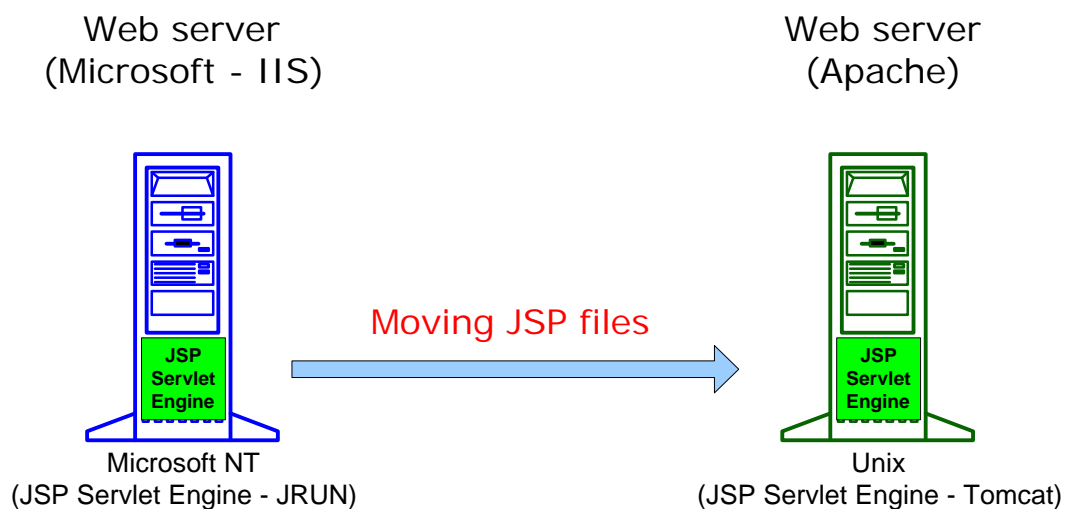
JSP is easy to learn and allows developers to quickly produce web sites and applications in an open and standard way. JSP is based on Java, an object-oriented language. JSP offers a robust platform for web development.

Main reasons to use JSP:

1. Multi platform
2. Component reuse by using Javabeans and EJB.
3. Advantages of Java.

You can take one JSP file and move it to another platform, web server or JSP Servlet engine.

## Moving JSP file from one platform to another.



HTML and graphics displayed on the web browser are classed as the presentation layer. The Java code (JSP) on the server is classed as the implementation. By having a separation of presentation and implementation, web designers work only on the presentation and Java developers concentrate on implementing the application.

## JSP compared to ASP

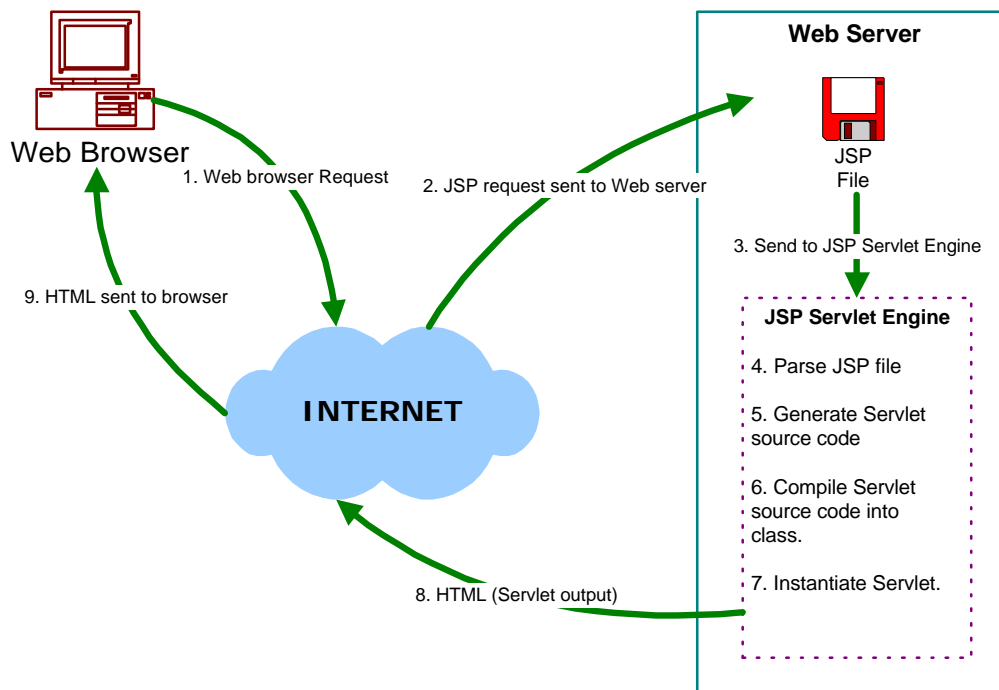
JSP and ASP are fairly similar in the functionality that they provide. JSP may have slightly higher learning curve. Both allow embedded code in an HTML page, session variables and database access and manipulation. Whereas ASP is mostly found on Microsoft platforms i.e. NT, JSP can operate on any platform that conforms to the J2EE specification. JSP allow component reuse by using Javabeans and EJBs. ASP provides the use of COM / ActiveX controls.

## JSP compared to Servlets

A Servlet is a Java class that provides special server side service. It is hard work to write HTML code in Servlets. In Servlets you need to have lots of `println` statements to generate HTML.

## JSP architecture

JSPs are built on top of SUN's servlet technology. JSPs are essentially an HTML page with special JSP tags embedded. These JSP tags can contain Java code. The JSP file extension is .jsp rather than .htm or .html. The JSP engine parses the .jsp and creates a Java servlet source file. It then compiles the source file into a class file, this is done the first time and this is why the JSP is probably slower the first time it is accessed. Any time after this the special compiled servlet is executed and it therefore returns faster.



### Steps required for a JSP request:

1. The user goes to a web site made using JSP. The user goes to a JSP page (ending with .jsp). The web browser makes the request via the Internet.
2. The JSP request gets sent to the Web server.
3. The Web server recognises that the file required is special (.jsp), therefore passes the JSP file to the JSP Servlet Engine.
4. If the JSP file has been called the first time, the JSP file is parsed, otherwise go to step 7.
5. The next step is to generate a special Servlet from the JSP file. All the HTML required is converted to println statements.
6. The Servlet source code is compiled into a class.
7. The Servlet is instantiated, calling the *init* and *service* methods.
8. HTML from the Servlet output is sent via the Internet.
9. HTML results are displayed on the user's web browser.

## Setting up a JSP environment

Before setting up the JSP environment, you must make sure you have the JDK.

Download JDK 1.3 from the following URL:

<http://java.sun.com/j2se/1.3/>

For Windows, the complete download is about 30 MB.

Run through the setup.

One of the main problems new Java developers have is setting the PATH and CLASSPATH.

For Windows 95/98/ME you edit the AUTOEXEC.BAT file with the new PATH and CLASSPATH settings and reboot your machine.

For Windows NT/2000 you edit the environment settings.

Both of these changes are described in the Java installation instructions.

### Download the JSP environment

You can download JSP environments from the web.

<http://java.sun.com/products/jsp/download.html>

The preferred option is to download the J2EE Reference Implementation, Tomcat (approximately 3MB). Tomcat is a free open source JSP and Servlet engine, developed by Apache. Instructions to download Tomcat are given below.

### For Tomcat setup

To download Tomcat (current version 3.2.1), go to the following URL:

<http://jakarta.apache.org/builds/jakarta-tomcat/release/v3.2.1/bin/>

Download your chosen compressed file, for example on Windows you would download the zip file – 3.0 MB:

<http://jakarta.apache.org/builds/jakarta-tomcat/release/v3.2.1/bin/jakarta-tomcat-3.2.1.zip>

Unzip the file into a directory and set an environment variable TOMCAT\_HOME to your main Tomcat directory:

For example,

```
set TOMCAT_HOME=c:\tomcat
```

To start the server change to the tomcat\bin directory and type:

**startup**

Open a web browser and in the address box type:

<http://localhost:8080/> - this displays the example page.



Place any new JSP files in the "webapps" directory under your installed Tomcat directory.

For example, to run "myfirst.jsp" file, copy the file into the "webapps/ROOT" directory and then open a browser to the address:

<http://localhost:8080/myfirst.jsp>

This will show you the executed JSP file.

**CONGRATULATIONS - YOUR JSP and SERVLET ENVIRONMENT IS NOW SETUP!**

## Creating your first JSP page

```
<html>
<head>
<title>My first JSP page
</title>
</head>
<body>
<%@ page language="java" %>

<% System.out.println("Hello World"); %>
</body>
</html>
```

Type the code above into a text file. Name the file helloworld.jsp.  
Place this in the correct directory on your JSP web server and call it via your browser.

## Using JSP tags

There are four main tags:

1. Declaration tag
2. Expression tag
3. Directive Tag
4. Scriptlet tag
5. Action tag

## Declaration tag ( <%! %> )

This tag allows the developer to declare variables or methods.

Before the declaration you must have <%!

At the end of the declaration, the developer must have %>

Code placed in this tag must end in a semicolon ( ; ).

Declarations do not generate output so are used with JSP expressions or scriptlets.

For Example,

```
<%!  
    private int counter = 0 ;  
    private String get Account ( int accountNo) ;  
%>
```

## Expression tag ( <%= %> )

This tag allows the developer to embed any Java expression and is short for `out.println()`.

A semicolon ( ; ) does not appear at the end of the code inside the tag.

For example, to show the current date and time.

```
Date : <%= new java.util.Date() %>
```

## Directive tag ( <%@ directive ... %>)

A JSP directive gives special information about the page to the JSP Engine.

There are three main types of directives:

- 1) page – processing information for this page.
- 2) Include – files to be included.
- 3) Tag library – tag library to be used in this page.

Directives do not produce any visible output when the page is requested but change the way the JSP Engine processes the page.

For example, you can make session data unavailable to a page by setting a page directive (session) to false.

### 1. Page directive

This directive has 11 optional attributes that provide the JSP Engine with special processing information. The following table lists the 11 different attributes with a brief description:

language	Which language the file uses.	<%@ page language = "java" %>
extends	Superclass used by the JSP engine for the translated Servlet.	<%@ page extends = "com.taglib..." %>
import	Import all the classes in a java package into the current JSP page. This allows the JSP page to use other java classes.	<%@ page import = "java.util.*" %>
session	Does the page make use of sessions. By default all JSP pages have session data available. There are performance benefits to switching session to false.	Default is set to true.
buffer	Controls the use of buffered output for a JSP page. Default is 8kb	<%@ page buffer = "none" %>
autoFlush	Flush output buffer when full.	<%@ page autoFlush = "true" %>
isThreadSafe	Can the generated Servlet deal with multiple requests? If true a new thread is started so requests are handled simultaneously.	
info	Developer uses info attribute to add information/document for a page. Typically used to add author, version, copyright and date info.	<%@ page info = "visualbuilder.com test page, copyright 2001. " %>
errorPage	Different page to deal with errors. Must be URL to error page.	<%@ page errorPage = "/error/error.jsp" %>
IsErrorPage	This flag is set to true to make a JSP page a special Error Page. This page has access to the implicit object exception (see later).	

contentType	Set the mime type and character set of the JSP.	
-------------	---	--

## 2. Include directive

Allows a JSP developer to include contents of a file inside another. Typically include files are used for navigation, tables, headers and footers that are common to multiple pages.

Two examples of using include files:

This includes the html from privacy.html found in the include directory into the current jsp page.

```
<%@ include file = "include/privacy.html" %>
```

or to include a navigation menu (jsp file) found in the current directory.

```
<%@ include file = "navigation.jsp" %>
```

Include files are discussed in more detail in the later sections of this tutorial.

## 3. Tag Lib directive

A tag lib is a collection of custom tags that can be used by the page.

```
<%@ taglib uri = "tag library URI" prefix = "tag Prefix" %>
```

Custom tags were introduced in JSP 1.1 and allow JSP developers to hide complex server side code from web designers.

This topic will be covered in the Advanced JSP tutorial at [visualbuilder.com](http://visualbuilder.com)

## Scriptlet tag ( <% ... %> )

Between <% and %> tags, any valid Java code is called a Scriptlet. This code can access any variable or bean declared.

For example, to print a variable.

```
<%  
  
    String username = "visualbuilder" ;  
    out.println ( username ) ;  
  
%>
```



## Action tag

There are three main roles of action tags :

- 1) enable the use of server side Javabeans
- 2) transfer control between pages
- 3) browser independent support for applets.

### Javabeans

A Javabeans is a special type of class that has a number of methods. The JSP page can call these methods so can leave most of the code in these Javabeans. For example, if you wanted to make a feedback form that automatically sent out an email. By having a JSP page with a form, when the visitor presses the submit button this sends the details to a Javabeans that sends out the email. This way there would be no code in the JSP page dealing with sending emails (JavaMail API) and your Javabeans could be used in another page (promoting reuse).

To use a Javabeans in a JSP page use the following syntax:

```
<jsp : usebean id = " .... " scope = "application" class = "com..." />
```

The following is a list of Javabeans scopes:

page – valid until page completes.

request – bean instance lasts for the client request

session – bean lasts for the client session.

application – bean instance created and lasts until application ends.

## Creating your second JSP page

For the second example, we will make use of the different tags we have learnt. This example will declare two variables; one string used to store the name of a website and an integer called counter that displays the number of times the page has been accessed. There is also a private method declared to increment the counter. The website name and counter value are displayed.

```
<HTML>
<HEAD>
<TITLE> JSP Example 2</TITLE>
</HEAD>
<BODY> JSP Example 2
<BR>
<%!
    String sitename = "visualbuilder.com";
    int counter = 0;

    private void increment Counter()
    {
        counter ++;
    }
%>

Website of the day is
<%= sitename %>
<BR>
page accessed
<%= counter %>
</BODY>
</HTML>
```

## Implicit Objects

So far we know that the developer can create Javabeans and interact with Java objects. There are several objects that are automatically available in JSP called implicit objects.

The implicit objects are

Variable	Of type
Request	Javax.servlet.http.HttpServletRequest
Response	Javax.servlet.http.HttpServletResponse
Out	Javax.servlet.jsp.JspWriter
Session	Javax.servlet.http.HttpSession
PageContent	Javax.servlet.jsp.PageContext
Application	Javax.servlet.http.ServletContext
Config	Javax.servlet.http.ServletConfig
Page	Java.lang.Object

### Page object

Represents the JSP page and is used to call any methods defined by the servlet class.

### Config object

Stores the Servlet configuration data.

### Request object

Access to information associated with a request. This object is normally used in looking up parameter values and cookies.

```
<% String devStr = request.getParameter("dev"); %>  
Development language = <%= devStr %>
```

This code snippet is storing the parameter "dev" in the string devStr. The result is displayed underneath.

## JSP Comments

Coming soon...

## Creating your third JSP page

Coming soon...

## JSP page directives

Coming soon...

## Static JSP includes

Coming soon...

## Dynamic JSP includes

Coming soon...



## Error pages

Coming soon...

## Using JavaBeans with JSP

Coming soon...

If you have any comments or suggestions regarding this document, please use the JSP forum at [visualbuilder.com](http://visualbuilder.com) or send us an email at [support@visualbuilder.com](mailto:support@visualbuilder.com)

Thank you,

Visualbuilder team

<http://www.visualbuilder.com>