JSP-COMPATIBLE WEB SERVERS

You must have access to a JSP-compatible Web server before beginning to develop JavaServer Pages code. There are several JSP-compatible Web servers to choose from and most of the server software is available in both Windows and UNIX versions. Some of the available Web servers are written in the Java programming language and can be installed on any computer that has a Java Runtime Environment installed.

A computer running a Windows operating system with the Tomcat Web server installed was used to process the JSP pages created in this book.

CHOOSING A WEB SERVER

When choosing a Web server, you should evaluate the strengths and weaknesses of several Web servers to determine which server best suits your needs and meets your level of expertise. For example, while a Web server may be capable of processing thousands of JSP pages per hour, the server may be very complicated to set up.

Feature List

Some Web servers support a wider range of features than other Web servers. For example, some Web servers support detailed logging and error reporting, while others do not. You should examine the feature list for a Web server to determine whether the server meets your current and future needs.

Cost

Web server software is often available free of charge for development purposes, but you may have to pay for the software if it will be used in a commercial application. Some Web servers can be used free of charge for a specific period of time, after which a fee must be paid to continue using the Web server.

Terms and Conditions

Each Web server has its own specific terms and conditions of use. Prior to using any Web server software, you should carefully review the terms and conditions to ensure that the way you intend to use the software complies with the conditions of use.

POPULAR JSP-COMPATIBLE WEB SERVERS

The following are examples of JSP-compatible Web servers.

Jigsaw

Jigsaw is a Java-based Web server developed by the World Wide Web Consortium (W3C), which is the body responsible for many Web standards. The Jigsaw Web server is better suited to development than deployment. For more information about the Jigsaw Web server, visit the www.w3c.org/Jigsaw Web site.

LiteWebServer

LiteWebServer is a small, robust Web server developed by Gefion Software. This Web server is suitable for developing JSP pages and for making pages available on an intranet. For more information about LiteWebServer, visit the www.gefionsoftware.com Web site.

JSP-COMPATIBLE WEB SERVERS (Continued)

Nexus

Nexus is written entirely in Java and can be used either as a stand-alone Web server or as part of a larger application. To download the Nexus Web server, visit the www-uk.hpl.hp.com/people/ak/java/nexus Web site.

Orion Application Server

Orion Application Server is a popular JSP-compatible Web server that is suitable for most commercial needs. Orion Application Server supports clustering, which allows a large Web site to be stored on multiple Web servers. To learn more about Orion Application Server, visit the www.orionserver.com Web site.

Servertec Internet Server

Servertec Internet Server is an easy-to-use Web server that can be used as a stand-alone Web server or can be integrated with the Apache Web server. Apache is the most popular Web server on the Internet. Servertec Internet Server is written entirely in the Java programming language. For more information about Servertec Internet Server, visit the www.servertec.com Web site.

Tomcat

The Tomcat Web server can be used as a stand-alone Web server or can be integrated with the Apache Web server. The Tomcat Web server is available at the jakarta.apache.org Web site. For information about installing the Tomcat Web server, see page 60.

WebSphere Application Server

WebSphere Application Server is a Java-based Web server developed by IBM that can be used to develop JSP pages and deploy large-scale e-business applications. For more information about WebSphere Application Server, visit the www-4.ibm.com/software/webservers Web site.

WEB HOSTING SERVICES

If you do not want to install your own Web server software to develop JSP pages, you can use a Web hosting service that offers JSP-compatible Web servers. A Web hosting service is a company that allows individuals to store Web sites they create on the hosting service's Web servers. A Web hosting service may also offer access to other technologies related to JSP, such as JDBC, the Java technology used to enable JSP pages to retrieve information from a database.

Web hosting fees are typically inexpensive, making a Web hosting service an affordable alternative to installing a Web server on your own computer. A Web hosting service usually charges a monthly fee for their services. The fees vary depending on the amount of space, bandwidth and other resources required by the Web site.

When developing JSP pages using a Web hosting service, you will need a text editor to write the code and an ftp program to transfer your pages to the Web hosting service's Web servers.

Examples of Web hosting services that offer JSP-compatible Web servers include:

SpinWeb
www.spinweb.net

Reinvent Technologies, Inc.
www.reinventinc.com

iMagine Internet Services
www.imagineis.com

Colossus, Inc.
www.colossus.net

MyServletHosting.com
www.myservlethosting.com

INSTALL THE TOMCAT WEB SERVER

Before you can create interactive and dynamic JSP pages, you must install a Web server that can interpret and process JavaServer Pages code. Tomcat is a fully-functional Web server that you can install on your computer to create and test JSP pages.

You need to install the Java Software Development Kit before installing the Tomcat Web server. You may need to change certain Tomcat settings to specify the location of the Java SDK. Consult the documentation included with Tomcat for information about changing these settings.

Tomcat is constantly being updated. A recent release of the Tomcat Web server is included on the CD-ROM disc that accompanies this book, but you should make sure you install the latest version of the server. The latest version of Tomcat is available at the jakarta.apache.org/tomcat Web site.

The version of Tomcat you install may be an unfinished, or beta, version. Although Tomcat is a very stable application, some difficulties, such as system crashes, should be expected when using any beta software.

To install Tomcat, you simply copy the Tomcat Web server files to your computer. Once Tomcat is installed, it can be started using a program called startup, which is located in the bin directory. The bin directory also stores the shutdown program, which you can execute to stop Tomcat when you have finished displaying Web and JSP pages.

To confirm that Tomcat has been installed and started properly, you can have the server display a page in a Web browser. After starting the Web browser you want to use, you enter the name or IP number of your computer, as well as the port number used by Tomcat. The default port number used by Tomcat is 8080. If you do not know the IP number of your computer, you can use 127.0.0.1, which is the IP number that computers running TCP/IP use to refer to themselves.

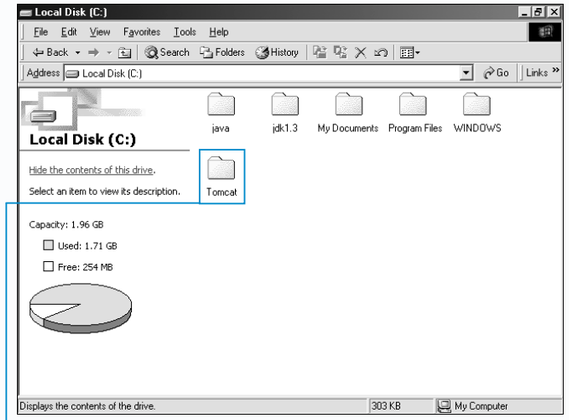
A Web hosting service may also allow you to access their Web server to test your JSP pages. For a list of Web hosting services, see page 59.

Extra

When the Tomcat Web server is installed, several directories are automatically created. These directories can be found in the main Tomcat directory.

DIRECTORY NAME:	DESCRIPTION:
bin	Stores programs for starting and shutting down the Tomcat Web server.
conf	Stores configuration files for the Tomcat Web server.
doc	Stores miscellaneous documents.
lib	Stores JAR (Java ARchive) files. The JAR file format is used to compress all the components of a Java program into a single file.
logs	Stores log files.
src	Stores the servlet Application Program Interface (API) source files used by the Tomcat Web server.
webapps	Stores sample Web applications.
work	Stores intermediate files, such as compiled JSP files. This directory may not have been created when you installed the Tomcat Web server.

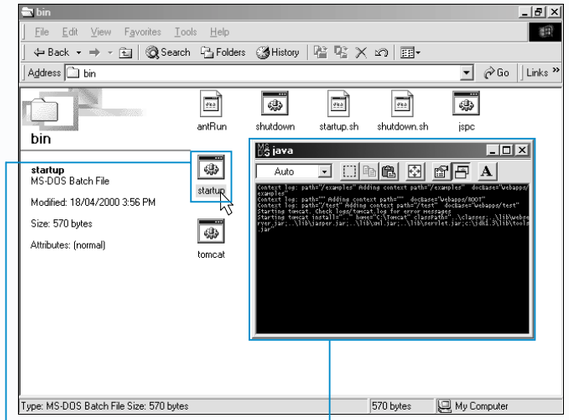
INSTALL THE TOMCAT WEB SERVER



INSTALL TOMCAT

1 Copy the Tomcat Web server files to your computer.

■ In this example, we copy the Tomcat Web server files to the C: drive. The files are stored in a directory called Tomcat.



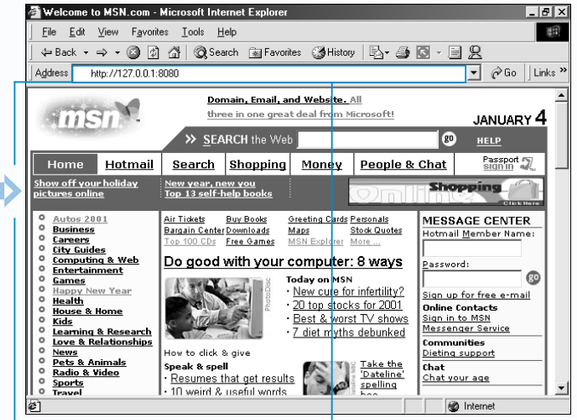
START TOMCAT

1 Display the contents of the bin directory, which is located in the Tomcat directory.

2 Double-click the appropriate startup file to start the Tomcat Web server.

Note: The startup.bat file is used to start Tomcat on Windows platforms. The startup.sh file is used to start Tomcat on non-Windows platforms.

■ A command prompt window appears.



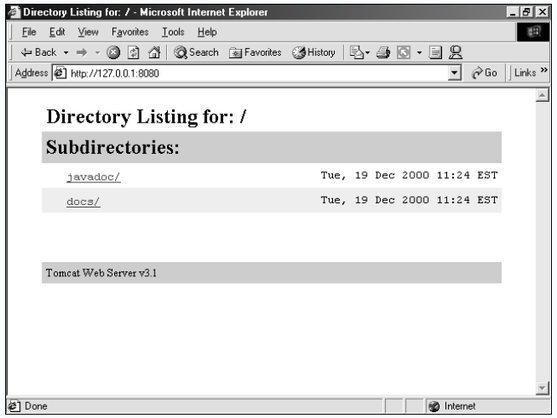
TEST TOMCAT

3 Start the Web browser you want to use to test the Tomcat Web server.

4 Click this area to highlight the current Web page address and then type **http://**.

5 Type the name or IP number of your computer followed by a colon.

6 Type the Tomcat port number and then press Enter.



■ The Web browser displays a Web page generated by the Tomcat Web server.

■ If the Tomcat Web server is not working properly, the Web browser will display an error message.

DISPLAY A WEB PAGE USING TOMCAT

After installing the Tomcat Web server, you can create a Web page and store the page on the server. You can then use a Web browser to display the Web page. Displaying a Web page allows you to confirm that Tomcat is installed properly and that you are storing pages in the correct directory.

Make sure you add the .html extension to the name of a Web page you save. Some text editors do not recognize the .html extension, so you may have to enclose the Web page name in quotation marks, such as "index.html".

When installed on a Windows platform, Tomcat uses the webapps directory as the root directory. All of the Web and JSP pages you want to display must be stored in the root directory or its subdirectories. When a Web server receives a request for a Web page, the server looks for the page in the root directory if no other directory is specified in the request. If another directory is specified, the Web server expects it to be a subdirectory of the root directory. For example, when a Web server with a root directory named docs receives the request www.server.com/work/sale.html, the server displays the document sale.html stored in the C:\docs\work directory.

If Tomcat is installed on a computer running a non-Windows operating system, such as Linux, the root directory may be different. The root directory could also change with newer versions of Tomcat. Always check the documentation included with Tomcat to verify the name and location of the root directory.

The webapps directory contains a number of directories that can be used to store your Web pages. If you are only using the Tomcat Web server to test Web and JSP pages, you may want to store the pages in the examples directory.

Before displaying Web pages, you must ensure that the Tomcat Web server is running. To start the Tomcat Web server, see page 60.

Extra

You can create your own directory within the webapps directory and then store pages you want to display in that directory. Creating your own directories is useful when you want to use Tomcat to make your pages available to others. Before you can display pages saved in a directory you created, you must change settings in the server.xml file. For example, if you created a directory named pages, you would add the following code to the server.xml file between the `<ContextManager>` tags:

```
<Context path="/pages" docBase="/pages" debug="0"
reloadable="true" >
</Context>
```

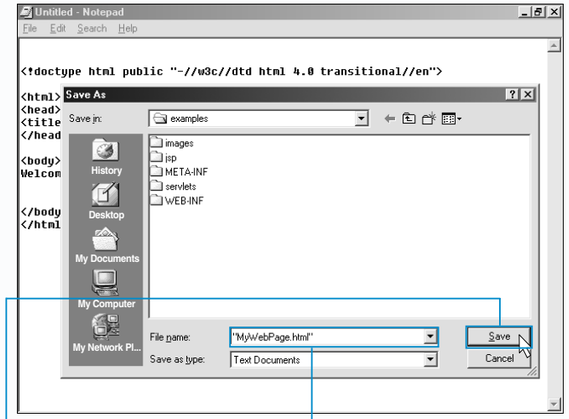
By default, the port number used by Tomcat is 8080. You must specify this number in the addresses you type when accessing pages generated by Tomcat. To change the port number used by Tomcat, open the server.xml file and look for the following section of code:

```
<Connector className="org.apache.tomcat.service.SimpleTcpConnector">
 <Parameter name="handler"
 value="org.apache.tomcat.service.http.HttpConnectionHandler"/>
 <Parameter name="port" value="8080"/>
</Connector>
```

In the line that specifies the port number, replace the existing port number with the port number you want to use. After you change the port number, you must specify the new number in Web page addresses.

The configuration settings for the Tomcat Web server are stored in the server.xml file located in the conf subdirectory of the main Tomcat directory. You can adjust the settings for Tomcat by changing or adding information to the server.xml file. To edit the file, you can open the file in a text editor. You should consult the user documentation included with Tomcat before changing any settings in the server.xml file.

DISPLAY A WEB PAGE USING TOMCAT



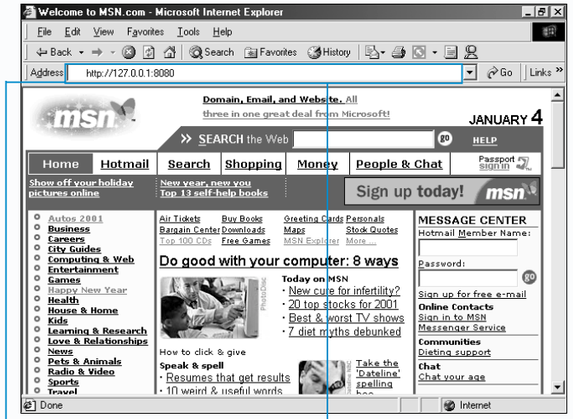
CREATE A WEB PAGE

1 In a text editor, create a Web page.

2 Save the Web page in the Tomcat root directory or one of its subdirectories.

Note: In this example, we save the Web page in the C:\Tomcat\webapps\examples directory.

■ The filename must have the .html extension and may need to be enclosed in quotation marks.



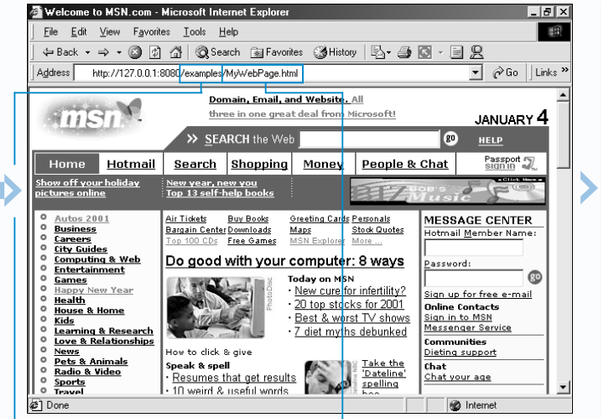
VIEW A WEB PAGE

1 Start the Web browser you want to use to display a Web page.

2 Click this area to highlight the current Web page address and then type **http://**.

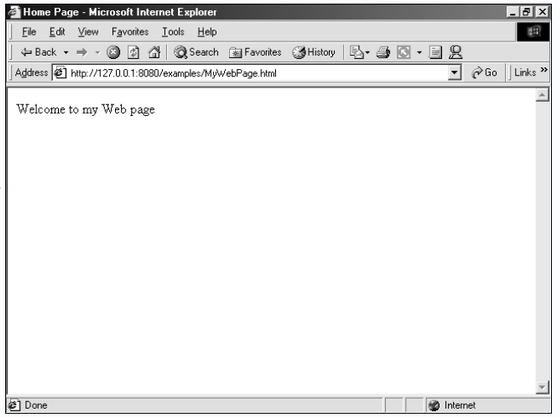
3 Type the name or IP number of your computer followed by a colon.

4 Type the Tomcat port number.



5 If the Web page is stored in a subdirectory of the root directory, type / followed by the name of the subdirectory that stores the page.

6 Type / followed by the filename of the Web page and then press Enter.



■ The Web page appears in the Web browser.

ADD A COMMENT TO A JSP PAGE

Adding comments to your HTML or JSP code is good programming practice and can help clarify the code. For example, you may use a variable named totalCost in your code. You could use a comment to explain whether the variable stores the total cost of all the products in a database or only some of the products. Comments can also contain information such as the author's name or the date the code was created. You can also use comments for debugging a program or as reminders to remove or update sections of code.

You can include HTML comments within the HTML code of a JSP page by enclosing the comments between the `<!--` and `-->` delimiters. Any text enclosed in these delimiters will be sent to the Web browser, but will not be displayed on the page. The information may be displayed by users who view the HTML source code however.

You can also add comments within the HTML code of a JSP page using the hidden comment tags, `<%--` and `--%>`.

Any code or information within the hidden comment tags will be discarded before any processing of the JSP page takes place on the Web server and will not be sent to the Web browser.

You can add comments to JSP code the same way you add comments to a Java application. The `//` notation can be used to create single-line comments and the `/*` and `*/` delimiters can be used to create multi-line comments. For information about adding comments to Java code, see page 15.

You should be very careful about where you place comments in a JSP page, especially when the comments are placed within the JSP code. The Web server expects to find only valid Java code within your JSP expressions, scriptlets and declarations. Any HTML comments or hidden comments in the JSP code will cause an error to occur.

Apply It

You may include JSP code within an HTML comment. This allows you to include dynamically generated comments in your JSP page. Embedding JSP code within HTML comments can help you to determine the state of various aspects of your JSP code without affecting the display of the Web page and can be a useful troubleshooting technique.

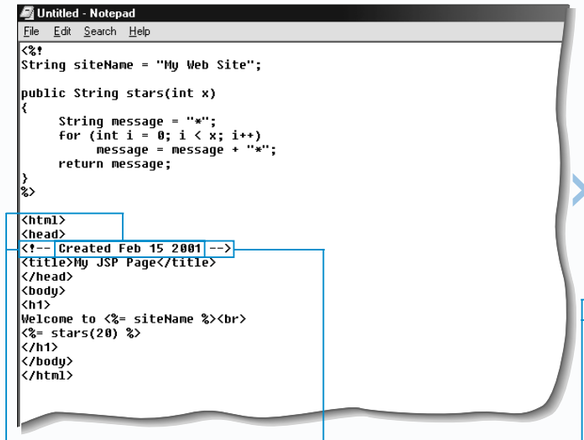
TYPE THIS:

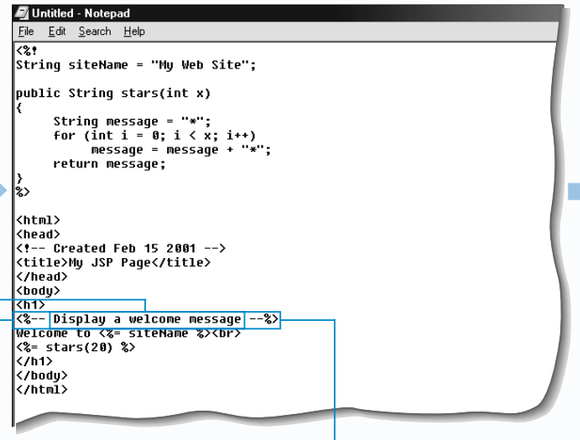
```
<%! String siteName = "My Web Site"; %>
<html>
<head>
<!-- The variable siteName has a value of
"<%= siteName %>" -->
<title>My JSP Page</title>
</head>
<body>
<% siteName = "* " + siteName + " *"; %>
<!-- The variable siteName has a value of
"<%= siteName %>" -->
Welcome to <%= siteName %><br>
</body>
</html>
```

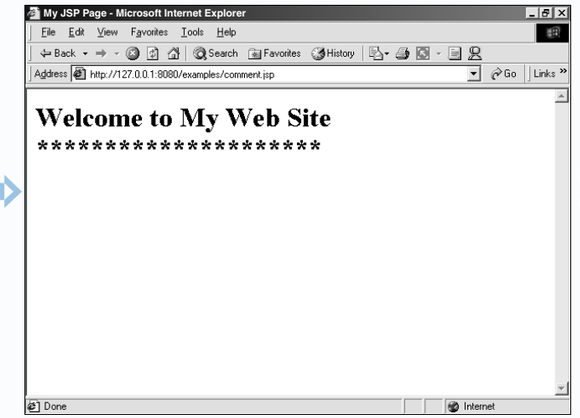
HTML SOURCE CODE:

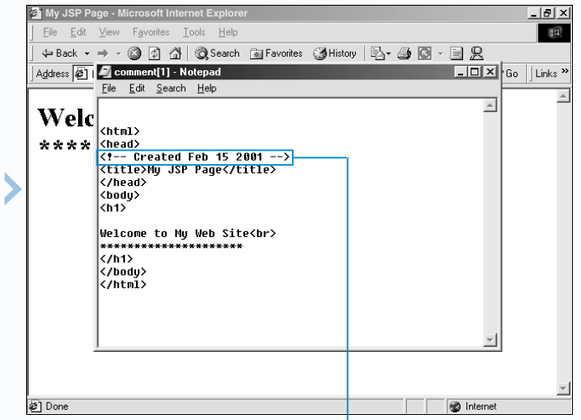
```
<html>
<head>
<!-- The variable siteName has a value of
"My Web Site" -->
<title>My JSP Page</title>
</head>
<body>

<!-- The variable siteName has a value of
"* My Web Site *" -->
Welcome to * My Web Site *<br>
</body>
</html>
```

ADD A COMMENT TO A JSP PAGE









1 To add an HTML comment, type **<!--**.

2 Type the comment.

3 Type **-->** to complete the HTML comment.

4 To add a hidden comment, type **<%--**.

5 Type the comment.

6 Type **--%>** to complete the hidden comment.

7 Save the page with the .jsp extension and then display the page in a Web browser.

■ The comments do not affect the display of the JSP page.

8 Display the source code of the JSP page.

■ The HTML comment is included in the HTML source code, but the hidden comment does not appear.

CREATE AN EXPRESSION

An expression is a scripting element that allows you to generate output on a JSP page. You can use expressions to insert information into a Web page rather than using scriptlets with `out.print()` or `out.println()` statements. This reduces the amount of code you have to type and can make your scripts easier to read.

The Web server processes the code within the expression and converts the results to a string. The results of the expression are then inserted into the HTML code in the same manner as the result of `out.print()` and `out.println()` statements in scriptlets.

A simple expression can be used to display a string enclosed in quotation marks or the value of a variable. The Web server simply inserts the string or value into the HTML code. Variables must be declared and initialized in scriptlets or declarations in the same JSP page.

You can also use calculations and method calls in your expressions. The expression processes the calculation or method and inserts the result into the HTML code. Methods used in expressions must be declared in the same JSP page and return a printable value. If a method does not return a value, the Web server displays an error message.

You can use string concatenation to join different types of information in a single expression. For example, you can create an expression such as <%= "Date of Birth: " + getDOB() %>, which generates a string followed by the value returned by a method.

An expression cannot end with a semicolon, as is customary with most Java statements. If a semicolon is included in an expression, an error will occur.

Users viewing the source code of the Web page from within a Web browser will not be able to view the contents of the expression. They will see only the information generated by the expression.

Apply It

If you need to include the " or \ character in a string, you will have to place a backslash (\) before the character. The " and \ characters have special meanings in Java and can be misinterpreted by the Web server, causing errors to occur.

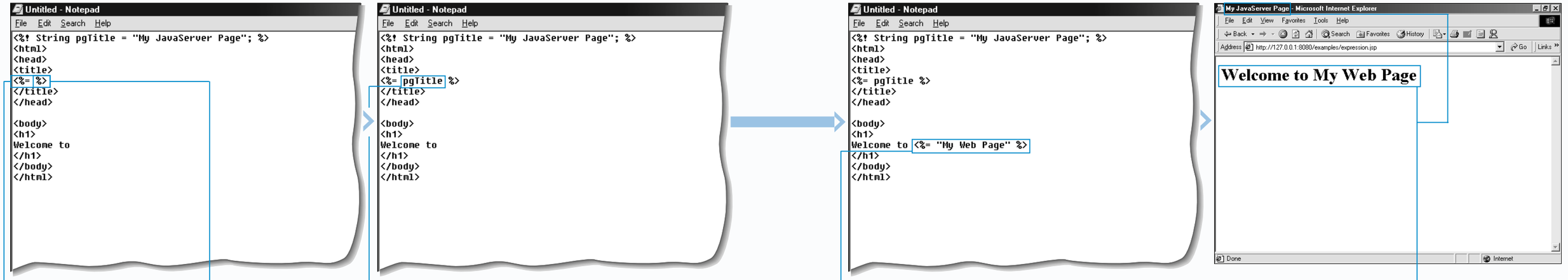
TYPE THIS:

```
<%
String fontFace = "comic";
%>

<%= "<font face=\"" + fontFace + "\">My Web Page</font>" %>
<%= "<br>c:\\Tomcat" %>
```

HTML SOURCE CODE:

```
<font face="comic">My Web Page</font>
<br>c:\Tomcat
```

CREATE AN EXPRESSION



1 Type **<%=** where you want to insert an expression in the JSP page.

2 Type **%>** where you want the expression to end.

3 Between the opening and closing delimiters (<%= and %>), type the code to be evaluated and included in the HTML code.

4 Repeat steps 1 to 3 for each expression you want to create.

5 Save the page with the .jsp extension and display the JSP page in a Web browser.

■ The Web browser displays the results of processing the expressions.

CREATE A DECLARATION

A declaration is a scripting element that allows you to define variables and methods that will be used throughout a JSP page. You must define variables and methods in a JSP page before you can use the variables and methods in the page. Although variables can also be defined within a scriptlet, using a declaration is the preferred method for defining variables.

To create a declaration, you place the code for the declaration between the `<%!` opening delimiter and the `%>` closing delimiter. Although a JSP page can include multiple declarations, this is not typically required. There is no limit to the amount of code you can include in a declaration, so you can define multiple variables and methods within the same declaration. Each line of code in a declaration must end with a semicolon, if a semicolon is required according to Java programming syntax.

Since declarations do not generate any output, they can be placed anywhere on a JSP page without interfering with the HTML code. Declarations are typically placed at the top of a page.

When defining a method in a declaration, you can use the `public`, `private` or `protected` access modifier to specify how the method will be accessed. For more information about access modifiers, see page 17. The access modifier you use becomes important when you import class files and other JSP pages into your code. For more information about including external files in a JSP page, see page 76.

If a declaration you want to add to a JSP page will contain many variables and methods, you may want to use another method of including the code, such as JavaBeans. JavaBeans allow you to store code in an external file so the source code of your JSP page is easier to understand and manage. For information on JavaBeans, see page 122.

Apply it

Using a declaration to define a method within a JSP page is similar to defining a method within a Java class. Once you define a method, you can access the contents of the method from anywhere in the JSP page. For example, you can use a scriptlet in the body of the page to pass a value to the method.

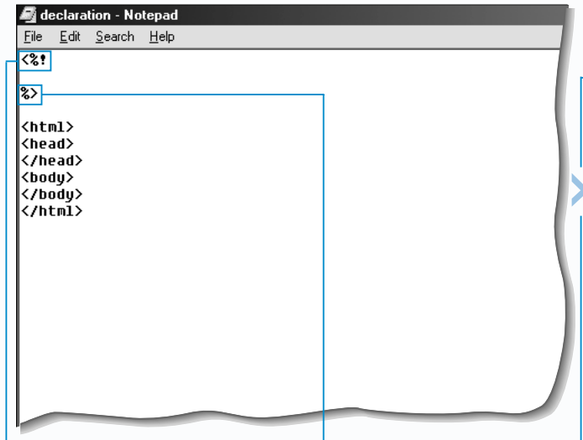
TYPE THIS:

```
<%!
String siteName = "My Web Server";
public String stars(int x)
{
      String message = "*";
      for (int i = 0; i < x; i++)
            message = message + "*";
      return message;
}
%>

<html>
<body>
Welcome to <%= siteName %><br>
<%= stars(22) %>
</body>
</html>
```

RESULT:

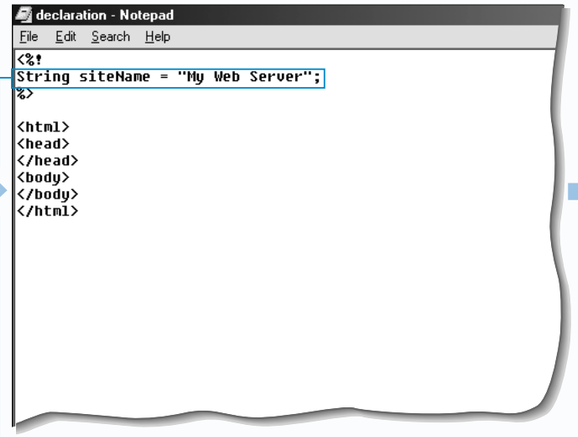
Welcome to My Web Server
****

CREATE A DECLARATION

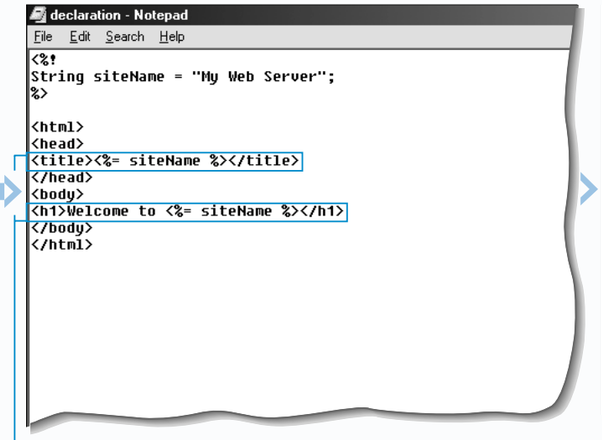


1 Type **<%!** where you want to add a declaration to a JSP page.

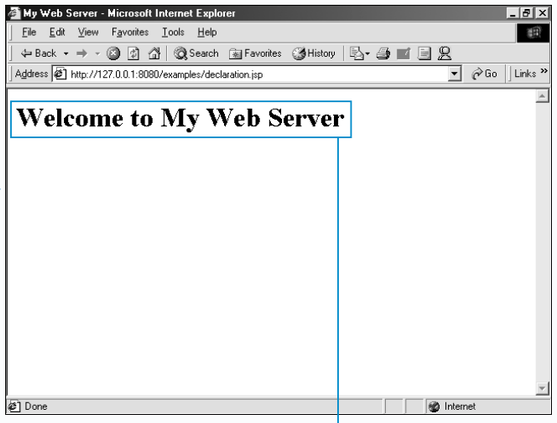
2 Type **%>** where you want the declaration to end.



3 Between the opening and closing delimiters (`<%!` and `%>`), type the code that defines a variable or method.



4 Type the code that uses the variable or method.



5 Save the page with the .jsp extension and then display the JSP page in a Web browser.

■ The Web browser displays the result of using the variable or method you defined in the declaration.

GENERATE TEXT USING SCRIPTLETS

You can use scriptlets to generate text on a JSP page. A scriptlet is a block of code embedded within a page. The code for a scriptlet is almost always written in the Java programming language, though some Web servers support scriptlets written in other languages. A page containing a scriptlet is often referred to as a template.

To add a scriptlet to a page, you place the code you want to embed in the page between the `<%` opening delimiter and the `%>` closing delimiter. A scriptlet can be used to generate text ranging from a simple message to the entire content of a JSP page.

Within a scriptlet, you can use the `out` object with the `print` or `println` member to generate text for a JSP page. The `out` object sends output to a Web browser, while the `print` and `println` members generate the text that is to be inserted into the page.

The placement of a scriptlet within a JSP page is important. If a scriptlet is used to generate output, it must be placed in the body of the page rather than in an area that does not display content, such as between the `<head>` and `</head>` tags.

Before a JSP page containing scriptlets is generated, a Web server processes the code in the scriptlets. The information generated by the code is inserted into the page before the page is displayed. Users who visit the JSP page will not be able to see the code for a scriptlet, even if they display the source code for the page, since the source code will contain only the output generated by the scriptlet. Although scriptlets are relatively secure, you should avoid including sensitive information, such as passwords, in the code.

Apply It

When you use the `print` member with the `out` object, the text you output does not include any line breaks. If you use the `print` member several times in a row, all the text you enter will appear on one long line in the source code. If you want to split text into separate lines, you can use the `println` member. The `println` member inserts a line break at the end of each line of text.

TYPE THIS:

```
<%
out.print("<h1>");
out.print("Welcome to my Web Page");
out.print("</h1>");
out.print("<hr>");
%>
```

SOURCE CODE:

```
<h1>Welcome to my Web Page</h1><hr>
```

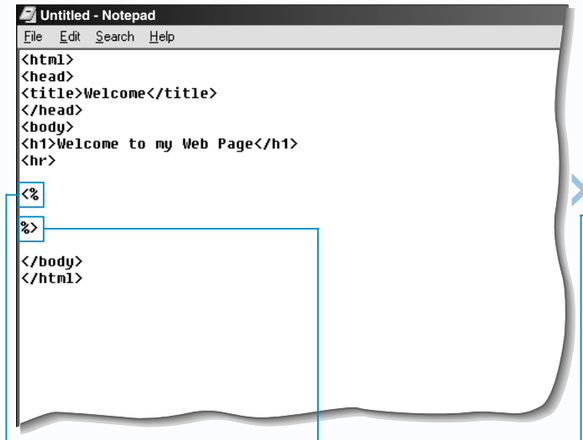
TYPE THIS:

```
<%
out.println("<h1>");
out.println("Welcome to my Web Page");
out.println("</h1>");
out.println("<hr>");
%>
```

SOURCE CODE:

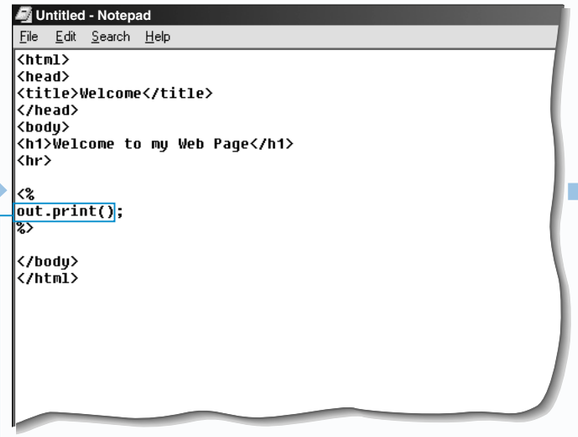
```
<h1>
Welcome to my Web Page
</h1>
<hr>
```

GENERATE TEXT USING SCRIPTLETS

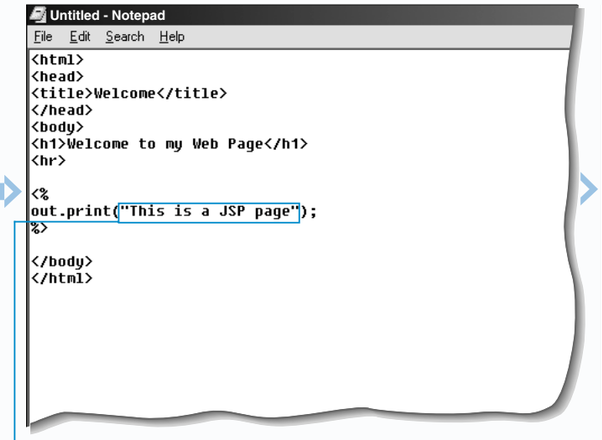


1 Type **<%** where you want to add a scriptlet to a JSP page.

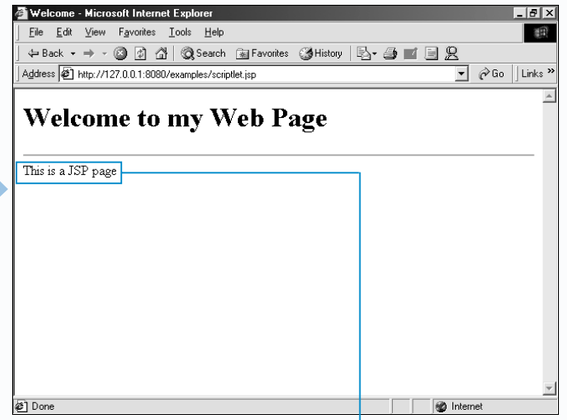
2 Type **%>** where you want the scriptlet to end.



3 Between the opening and closing delimiters (`<%` and `%>`), type **out.print( )** or **out.println( )** to generate text on the page.



4 Between the parentheses, type the text you want to output, enclosed in quotation marks.



5 Save the page with the .jsp extension and then display the JSP page in a Web browser.

The Web browser displays the result of generating text using a scriptlet.

WORK WITH MULTIPLE SCRIPTLETS

You can use multiple scriptlets within a single JSP page. This enables you to place dynamically created information in multiple locations throughout your Web page.

Any code used in one scriptlet can be accessed by other scriptlets in the same JSP page. For example, you can declare a variable in one scriptlet and then access the variable in another scriptlet that is in the same JSP page. Scriptlets will be processed in the order they appear on the JSP page, so you should consider the order of processing when creating scriptlets that use information from other scriptlets. For example, a scriptlet at the top of a JSP page will not be able to access variables declared in a scriptlet further down the page.

You must ensure that only valid Java code is included between the <% and %> delimiters. When using multiple scriptlets within HTML code, it is a common mistake to leave some HTML code between the scriptlet delimiters. HTML code included between the scriptlet delimiters must be included in valid Java statements so the Web server can dynamically generate the HTML code. If the Web server finds any raw HTML code in a scriptlet, an error will occur.

Using many large scriptlets in a JSP page can cause your code to be difficult to read and troubleshoot, should an error occur. Scriptlets are suitable for small amounts of code and for development and learning purposes. For other purposes, you should convert your scriptlets into a more manageable format, such as JavaBeans. For information about creating JavaBeans, see page 122.

Apply It

As well as displayable content for Web pages, scriptlets can also be used to generate non-displayable elements, such as attributes for HTML tags. This is useful if you also want to dynamically format your page. Before the HTML code is sent to the Web browser, the Web server replaces any scriptlets with the information generated by the scriptlets. As long as the combination of scriptlet output and HTML code in the page is valid, no errors will be generated.

TYPE THIS:

```
<%
int fontSize = 5;
String fontColor = "blue";
String fontFace = "Courier";
%>

<font face="<%
      out.print(fontFace);
%>" size="<%
      out.print(fontSize);
%>" color="<%
      out.print(fontColor);
%>">My Web Page</font>
```

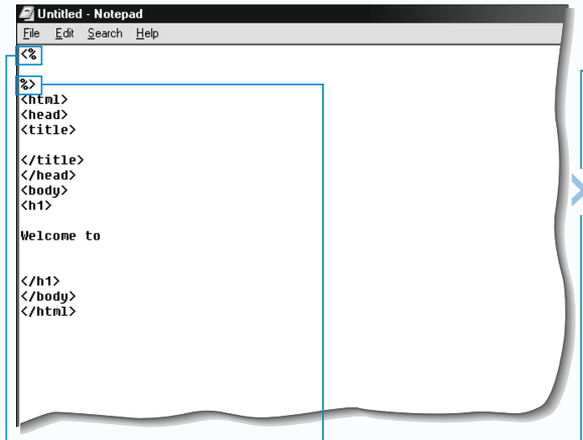
HTML SOURCE CODE:

```
<font face="Courier" size="5"
color="blue">My Web Page</font>
```

WEB BROWSER:

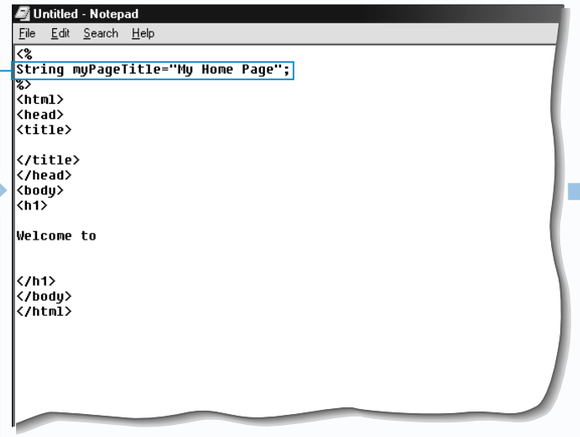
My Web Page

WORK WITH MULTIPLE SCRIPTLETS



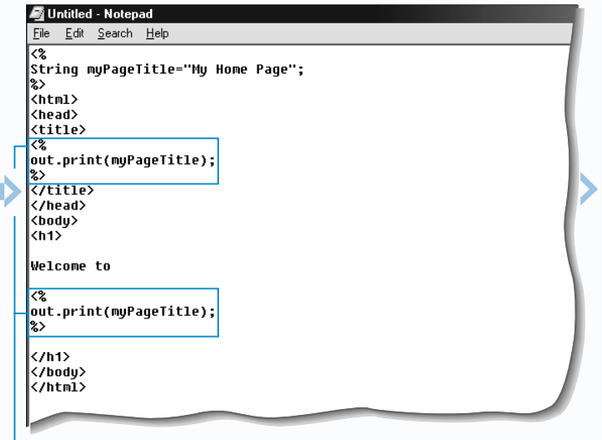
1 Type <% where you want to add a scriptlet to a JSP page.

2 Type %> where you want the scriptlet to end.

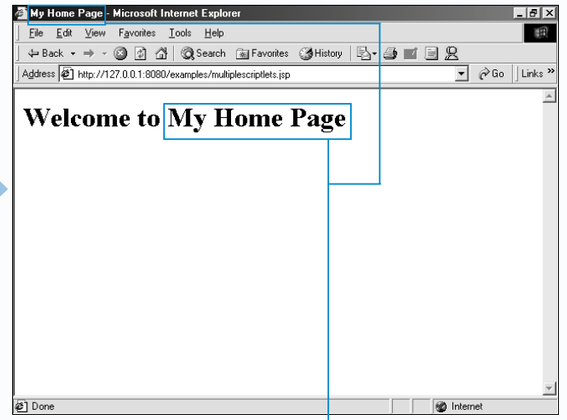


3 Between the opening and closing delimiters (<% and %>), type the code for the scriptlet.

■ In this example, code that assigns a value to a variable is inserted.



4 Repeat steps 1 to 3 for each scriptlet you want to add to your Web page.



5 Save the page with the .jsp extension and then display the page in a Web browser.

■ The Web browser displays the results of processing the scriptlets.

USING THE PAGE DIRECTIVE

Directives provide information about a JSP page to the software that processes the page. This software is often referred to as a *JSP engine* and is part of a Web server. Directives are sometimes called *JSP engine directives*. Directives do not produce visible output, but rather provide instructions and settings that determine how a JSP page is processed.

There are three JSP directives available–`page`, `include` and `taglib`. Each directive has attributes that can be assigned specific values. For example, the `page` directive offers the `autoFlush` attribute, which can be assigned a value of `true` or `false`. For a complete list of attributes that can be used with each directive, see page 241.

To add a directive to a JSP page, you place the directive statement between the `<%@` opening delimiter and the `%>` closing delimiter. The directive statement includes the name of the directive, followed by the attribute and value pairs you want to use. An attribute and its corresponding value are separated by an equal sign. A directive will only affect the JSP page containing the directive.

The `page` directive is the most commonly used directive. The `page` directive allows you to specify information about the configuration of a JSP page, such as the type of content you want the page to display. For example, you can use the `contentType` attribute with the `text/plain` value to specify that you want the information generated by a JSP page to be displayed as plain text.

A JSP page can contain multiple `page` directives. It is good programming practice to place `page` directives at the beginning of a JSP page, before any HTML or JavaServer Pages code on the page.

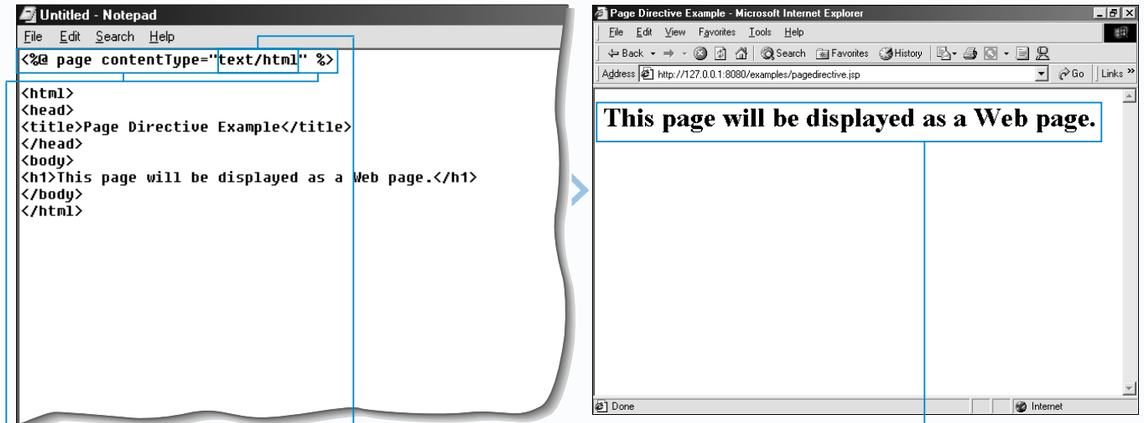
Extra

If you do not use any directives in a JSP page, the Web server's JSP engine will use its own default settings when processing the page. For example, if you do not use the `page` directive with the `contentType` attribute to specify how you want the information in a JSP page to be generated, a JSP engine will automatically display the information as a Web page.

Although you can add more than one `page` directive to a JSP page and each directive can contain more than one attribute, you usually cannot use the same attribute more than once on a page. For example, you cannot use the `contentType` attribute several times on a JSP page, since the information in the JSP page can only be generated one way at a time.

Some JSP engines do not support all of the attributes and values offered by the `page` directive. Before using the `page` directive, you should view the latest documentation for your Web server to determine whether the JSP engine will support the attributes and values you want to use.

USING THE PAGE DIRECTIVE



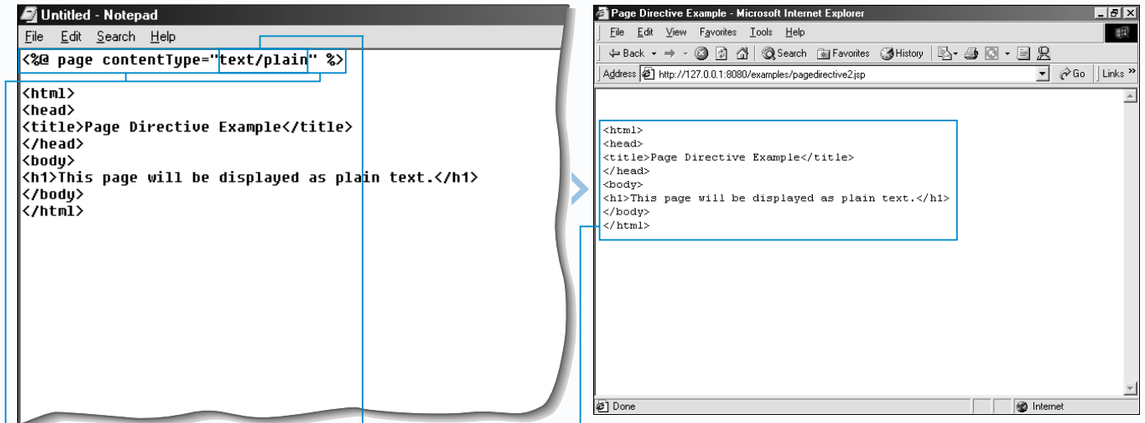
DISPLAY A JSP PAGE AS A WEB PAGE

1 In the first line of code in the JSP page, type **<%@ page contentType="" %>**.

2 To specify that you want to display the JSP page as a Web page, type **text/html** between the quotation marks.

3 Save the page with the .jsp extension and then display the JSP page in a Web browser.

■ The Web browser displays the result of using the `page` directive to display the JSP page as a Web page.



DISPLAY A JSP PAGE AS PLAIN TEXT

1 In the first line of code in the JSP page, type **<%@ page contentType="" %>**.

2 To specify that you want to display the JSP page as plain text, type **text/plain** between the quotation marks.

3 Save the page with the .jsp extension and then display the JSP page in a Web browser.

■ The Web browser displays the result of using the `page` directive to display the JSP page as plain text.

Note: Some Web browsers may automatically display a plain-text JSP page as a Web page.

USING THE INCLUDE DIRECTIVE

The `include` directive allows you to use one file in several different JSP pages. This can save you time when you need to include the same information in multiple pages. For example, if you have a copyright notice you want to display on all your JSP pages, you can create a file that contains the copyright notice and then use the `include` directive to include the information on all your pages. The file must be stored on the Web server and be accessible from all the JSP pages that you want to include the file.

You must first create the file you want to include. The file can contain plain text or HTML code, such as a table, header or footer. If the file contains plain text, you can save the file with the .txt extension. If the file contains HTML code, you can save it with the .html extension. The include file should not contain any JavaServer Pages code. The Web server will ignore any JavaServer Pages code included in the file.

To include a file in a JSP page, you add an `include` statement to the page. The `include` statement must be enclosed between the `<%@` opening delimiter and the `%>` closing delimiter. The filename specified in the `include` statement must be enclosed in quotation marks. The filename must be a fixed value, such as "footer.html". You cannot use a variable that represents the name of a file in an `include` statement.

If you change the code in the file, all of the JSP pages that include the file will be updated. You may have to clear the Web server's buffer before your JSP pages will display the changes made to the file. For information about clearing the Web server's buffer, see page 92.

Extra

The `include` directive allows you to include a file that is stored in the same directory as the JSP page that includes the file or in a subdirectory of that directory. In this example, the JSP page is stored in a directory called test and the footer.html file is stored in a subdirectory called test/pages.

Example:

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

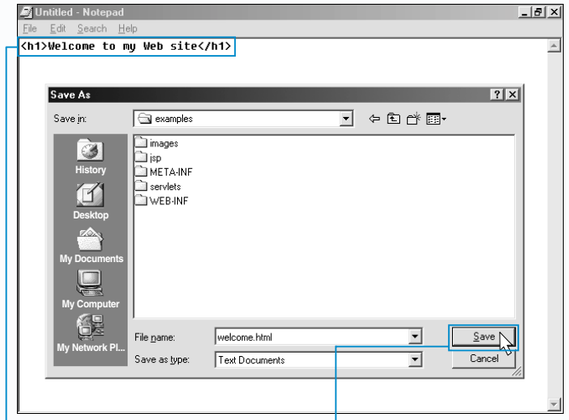
You can also include a file that is located in the parent directory of the directory that stores the JSP page. To do so, you use the double dot notation to represent the name of the parent directory. In this example, the JSP page is instructed to look for the welcome.html file in the parent directory.

Example:

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

Using the `include` directive allows you to break code into manageable sections and then include the code in JSP pages as needed. Each include file should contain code specific to only one task. If you create a file that contains code for many tasks, the JSP pages may not use all the code and the Web server's resources will be wasted.

USING THE INCLUDE DIRECTIVE

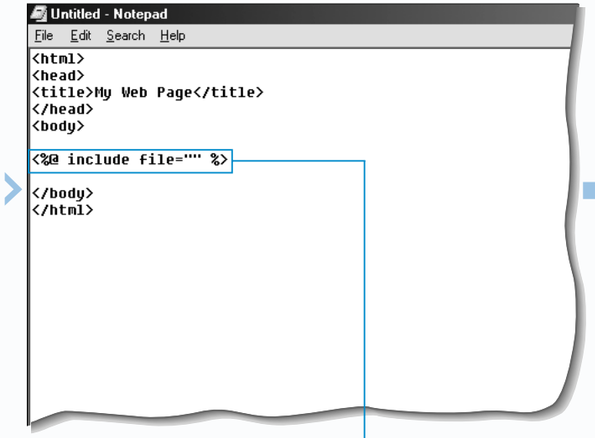


CREATE A FILE TO INCLUDE

1 In a text editor, create the file you want to include in several JSP pages.

2 Save the file.

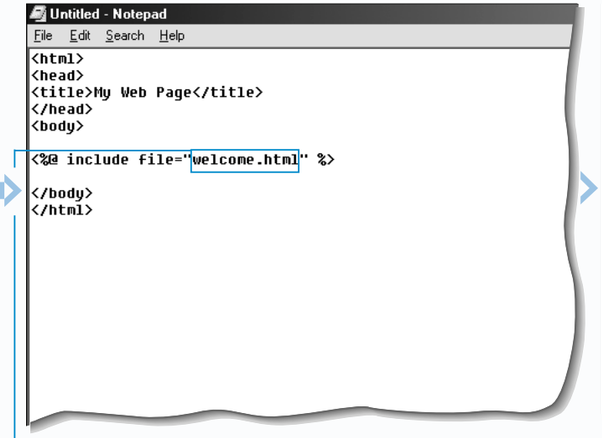
Note: If the file contains plain text, save the file with the .txt extension. If the file contains HTML code, save the file with the .html extension.



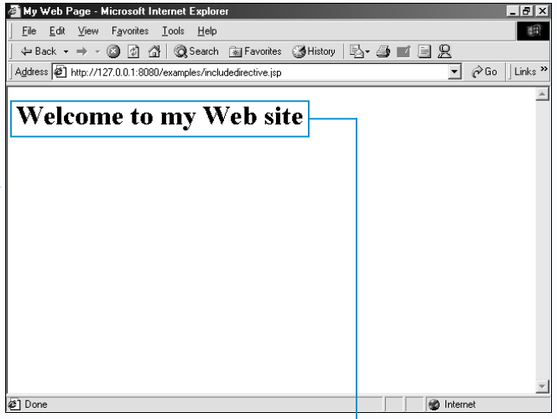
INCLUDE A FILE

1 Display the code for the JSP page in which you want to include a file.

2 Between the `<body>` and `</body>` tags, type **<%@ include file="" %>**.



3 Between the quotation marks, type the name of the file you want to include.



4 Save the page with the .jsp extension and then display the JSP page in a Web browser.

■ The Web browser displays the result of using the `include` directive.