INTRODUCTION TO SERVLETS

A servlet is a module of Java code that adds functionality to a Web server. Servlets use text to communicate with a Web server and have no graphical user interface.

The Java Servlet API

The construction of servlets is governed by a rigid specification. This specification is known as the Java Servlet Application Programming Interface, or more simply, the Java Servlet API. Detailed information about the Java Servlet API can be found at the java.sun.com Web site.

The Java Class Library

Servlets have access to all the class files that make up the Java class library, which is also called the Java API. This enables servlets to perform a wide variety of complex tasks, such as working with databases, reading and writing to files on local and remote computers and manipulating data passed by forms.

Object-Oriented Programming

Servlets make use of the object-oriented approach to programming. Object-oriented programming provides increased flexibility when maintaining and modifying code. While beginners may not benefit greatly from the object-oriented approach, this style of programming is vital to developers of large, complex Web sites.

Portable

Since servlets have to conform to the rigid Java Servlet API, a Web server that supports this specification will be able to run any servlet, regardless of the platform the servlet was developed on. This allows the same servlet to run on a Web server that uses the Windows operating system and a Web server that uses the UNIX operating system.

Efficient

When a servlet is used for the first time, the servlet is compiled into bytecode. Bytecode is a set of instructions that a Web server can use to perform the tasks specified in the servlet. Once a servlet is compiled into bytecode, the bytecode is retained in the Web server's memory. When the servlet is requested again, the Web server can use the bytecode stored in memory, without having to recompile the servlet. This dramatically speeds up the processing of servlets. A servlet will only need to be recompiled if it is modified.

Dynamic Content

Servlets can be used to enable a Web server to generate dynamic content. Dynamic content is Web page content that is generated depending on a set of changing parameters. For example, a servlet can generate a Web page that displays weather information obtained from a database. Each time the Web page is requested by a client, the servlet instructs the Web server to retrieve the latest weather information from the database and place the information in the Web page.

Multitasking

Servlets are able to handle multiple processes at once. As a result, servlets are useful for creating complex multi-user applications, such as chat room programs and file swapping programs. Besides handling multiple instructions at once, servlets can, when necessary, locate and use other servlets to share the workload.

SERVLETS AND CGI PROGRAMMING

Servlets provide an alternative to more traditional Common Gateway Interface (CGI) programming. While both servlets and CGI programs add functionality to a Web server, servlets give a Web server more control over the processing of information.

With CGI programming, a Web server does not have control over the way a CGI program processes information. When information is passed to the Web server, the server simply passes the information on to the CGI program and then waits for the program to finish processing the information. When the CGI program completes its tasks, the program passes the information back to the Web server, typically in the form of HTML code that is then sent to the client.

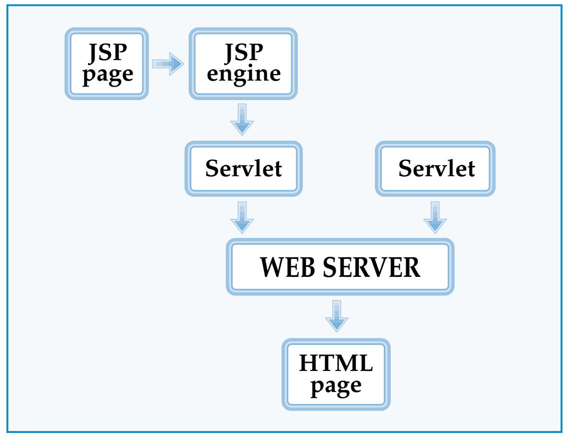
Servlets are more tightly integrated with a Web server than CGI programs, allowing more communication between the Web server and a servlet. This gives a Web server more control over the processing of information and allows for more advanced processing features, such as maintaining session information across multiple servlets.

SERVLETS AND JSP PAGES

Since JavaServer Pages technology is built on servlet technology, JSP pages and servlets are closely related. When a JSP page is processed, the page is converted into a servlet by the JSP engine on the Web server. The Web server can use the servlet generated from the JSP page as it would use any servlet. A Web server must support servlets before it can process JSP pages.

The translation of a JSP page into a servlet is transparent to the developer. Although it is an asset, developers do not need to understand servlet programming in order to work with JSP pages.

The relationship between JSP pages and servlets:



CREATE A SERVLET

You can create a simple servlet that uses Java to generate text that will be displayed in a Web browser.

To create a servlet, you must first import the `javax.servlet` and `javax.servlet.http` servlet packages. Before the servlet packages can be imported, they must be installed on your computer and accessible to your Web server software. The packages are located in the Java Servlet API class files, which are available at the java.sun.com/products/servlet Web site. For more information about obtaining and installing the Java Servlet API class files, see the top of page 199.

You will also need to import any additional packages needed by the servlet you want to create. For example, a servlet that will output text requires the `java.io` package.

You can make a servlet you create a sub-class of the `HttpServlet` class, which contains the code a Web server needs to run servlets. This saves you from having to type all the code needed by a Web server in each servlet. The `HttpServlet` class is part of the `javax.servlet` package.

The `doGet` method is the main method of a servlet. This method is processed each time a client uses a Web browser to connect to a servlet. The `doGet` method takes two arguments. The first argument is an `HttpServletRequest` object, which will contain information passed from the client. The second argument is an `HttpServletResponse` object, which will contain information to be sent to the client. The `doGet` method may also throw two errors—`ServletException` and `IOException`.

To generate text that will be sent to the client, you must create a `PrintWriter` object using the `getWriter` method of the `HttpServletResponse` object. The methods of the `PrintWriter` object can then be used to generate output.

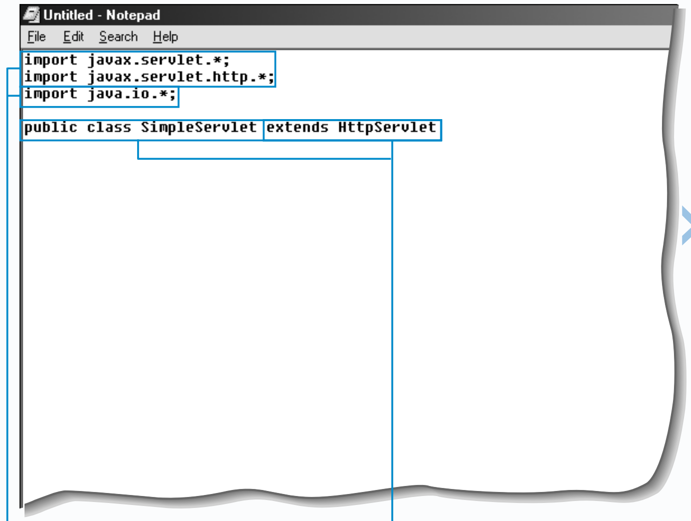
Extra

Servlets are platform independent. This means that a servlet you create can be stored and processed by any Web server that supports servlets, regardless of the platform on which you developed the servlet. Most Web servers that are capable of running Java programs are also able to run servlets.

The first time you attempt to display the results of using a servlet you created, the results may take a few moments to appear in the Web browser. This delay will occur only the first time the servlet is accessed. When a client requests a servlet for the first time, the Web server processes the servlet and then stores the servlet in memory. Subsequent requests for the servlet will be much faster, since the servlet will not need to be processed each time.

JavaServer Pages technology is built on servlet technology. JSP pages you create are converted into servlet code by the JSP engine on a Web browser when the pages are processed. A familiarity with servlet programming can help improve your ability to develop JSP pages, since you will have an increased understanding of how JSP pages are processed by a Web server.

CREATE A SERVLET



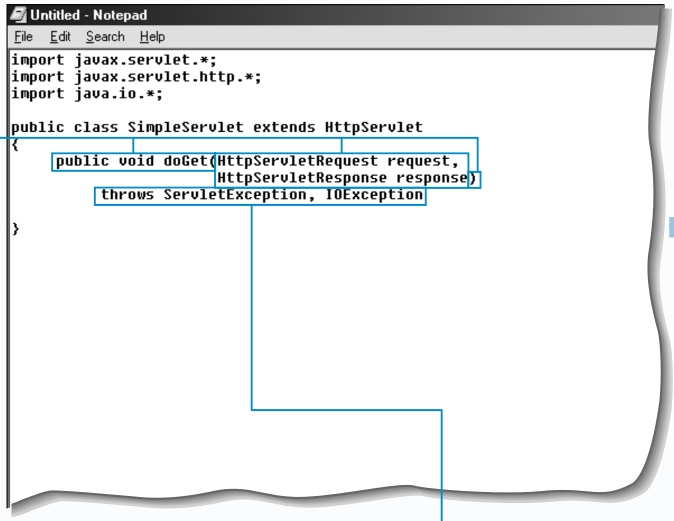
```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class SimpleServlet extends HttpServlet
```

1 Type the code that imports the `javax.servlet` and `javax.servlet.http` packages.

2 Type the code that imports any additional packages needed by the servlet.

3 To declare the class for the servlet, type **public class** followed by a name for the class.

4 To make the class a sub-class of the `HttpServlet` class, type **extends HttpServlet**.

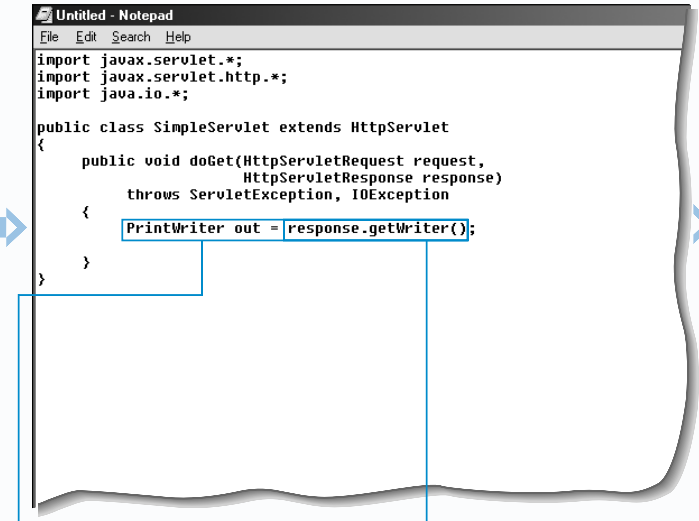


```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class SimpleServlet extends HttpServlet
{
     public void doGet(HttpServletRequest request,
                       HttpServletResponse response)
          throws ServletException, IOException

}
```

5 To create a `doGet` method, type **public void doGet()**.

6 Between the parentheses, type **HttpServletRequest** followed by a name for the `HttpServletRequest` object. Then type **HttpServletResponse** followed by a name for the `HttpServletResponse` object.

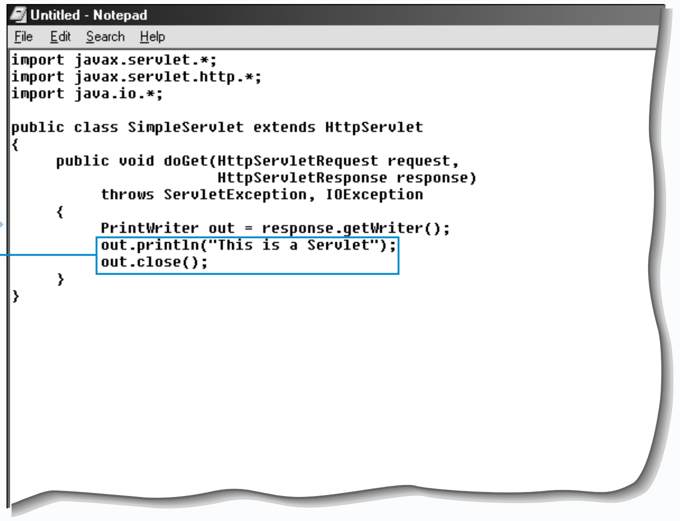
7 To specify the exception errors that may be thrown by the method, type **throws ServletException, IOException**.



```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class SimpleServlet extends HttpServlet
{
     public void doGet(HttpServletRequest request,
                       HttpServletResponse response)
          throws ServletException, IOException
     {
          PrintWriter out = response.getWriter();

     }
}
```

8 To create a `PrintWriter` object you can use to generate text that will be output to the client, type **PrintWriter** followed by a name for the object. Then type **=**.

9 Type the name of the `HttpServletResponse` object followed by a dot. Then type **getWriter()**.



```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class SimpleServlet extends HttpServlet
{
     public void doGet(HttpServletRequest request,
                       HttpServletResponse response)
          throws ServletException, IOException
     {
          PrintWriter out = response.getWriter();
          out.println("This is a Servlet");
          out.close();
     }
}
```

10 Type the code that uses the `PrintWriter` object to generate text output.

11 Save the servlet with the .java extension.

■ You can now compile and execute the servlet.

COMPILE AND EXECUTE A SERVLET

Once you have created a servlet, you can compile the Java code for the servlet. A servlet must be compiled before it can be executed.

A Java compiler is required to compile a servlet. The Java SDK includes a Java compiler called javac. The javac compiler can only be executed from the command prompt. If you are using a Windows operating system, you will need to open an MS-DOS Prompt or Command Prompt window to use javac. For more information about using javac, see page 20.

Before a servlet is compiled, the Java compiler checks the servlet code for errors. If an error is found, an error message will be generated. If no errors are found, the code will be successfully compiled and stored in a class file. The class file will have the same name as the source file, but will use the .class extension.

Once a servlet has been compiled, the resulting class file must be copied to the appropriate directory on the Web server. Consult the documentation for your Web server to determine where to save servlets. Most Web servers require executable files to be stored in a specific directory. For example, if you are using the Tomcat Web server version 3.1 and you use the examples directory for development, you should store your servlets in the examples/WEB-INF/classes directory.

Depending on the configuration of your Web server and the operating system you use, you may need to modify the permissions for a servlet class file. Refer to your Web server and operating system documentation for information about setting servlet permissions.

To execute a servlet, enter the URL of the servlet in a Web browser window.

Extra

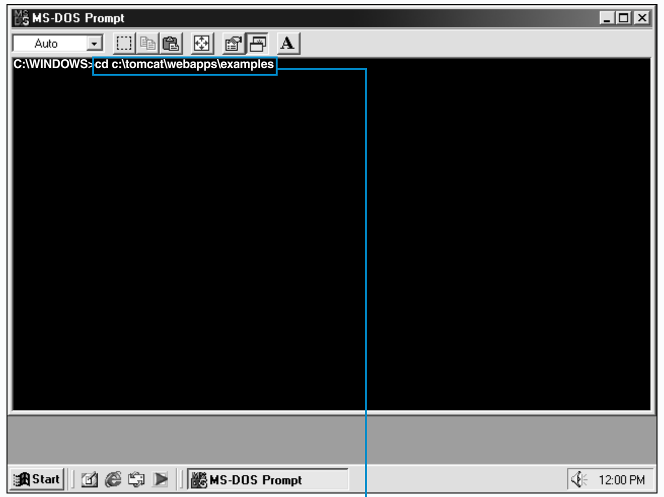
If you use the Tomcat Web server version 3.1, the URL you enter to execute a servlet will be determined by the settings in the web.xml file. You must edit the web.xml configuration file to specify information about a servlet you want to execute. If you use the examples directory for development, the web.xml file you must edit is located in the examples/WEB-INF directory.

To edit the web.xml file, you enter tags that specify information about the servlet between the `<web-app>` and `</web-app>` tags. The `<servlet>` tag allows you to specify the class name of the servlet, while the `<servlet-mapping>` tag lets you specify the URL that will be used to execute the servlet. For example, to edit the web.xml file for a servlet named SimpleServlet that you want to access by typing **/SimpleServlet** in the URL, enter the following code after the `<web-app>` tag in the web.xml file.

Example:

```
<servlet>
      <servlet-name>SimpleServlet</servlet-name>
      <servlet-class>SimpleServlet</servlet-class>
</servlet>
<servlet-mapping>
      <servlet-name>SimpleServlet</servlet-name>
      <url-pattern>/SimpleServlet</url-pattern>
</servlet-mapping>
```

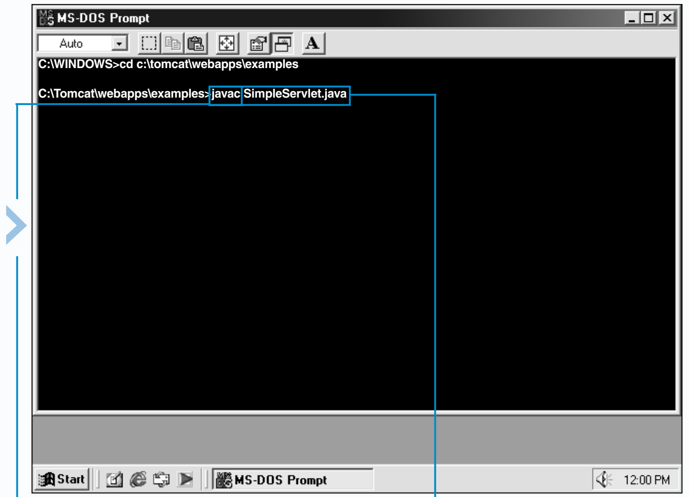
COMPILE AND EXECUTE A SERVLET



```
C:\WINDOWS>cd c:\tomcat\webapps\examples
```

1 Open the window that allows you to work at the command prompt.

2 Move to the directory that stores the servlet you want to compile.

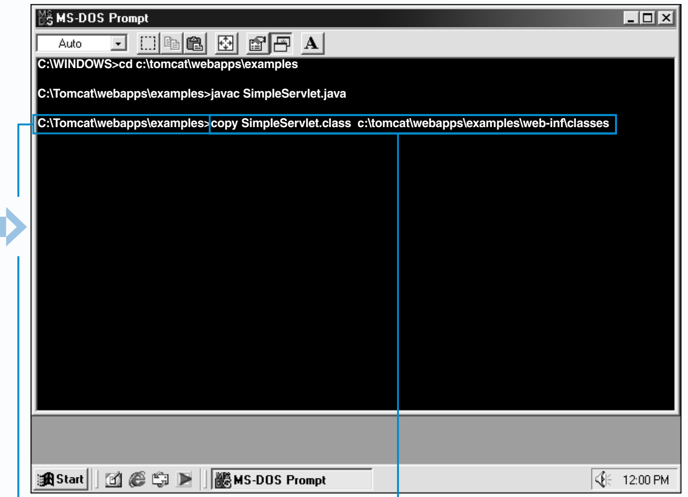


```
C:\WINDOWS>cd c:\tomcat\webapps\examples

C:\Tomcat\webapps\examples>javac SimpleServlet.java
```

3 To compile the servlet using the javac compiler, type **javac**.

4 Type the name of the servlet you want to compile, including the .java extension.

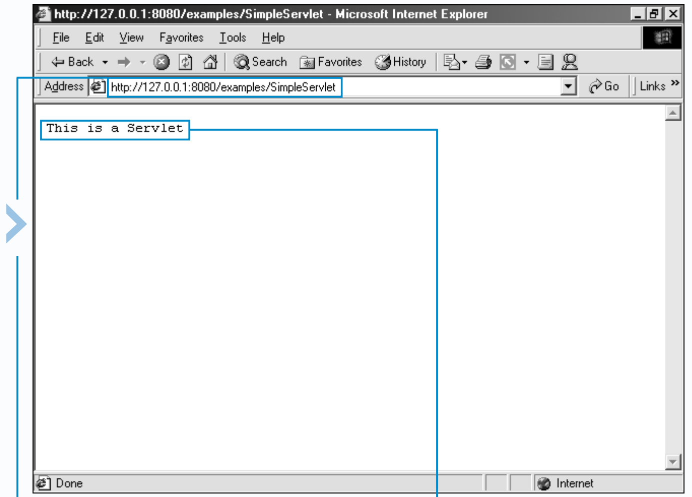
5 Press Enter to compile the Java code.



```
C:\WINDOWS>cd c:\tomcat\webapps\examples

C:\Tomcat\webapps\examples>javac SimpleServlet.java

C:\Tomcat\webapps\examples>copy SimpleServlet.class  c:\tomcat\webapps\examples\web-inf\classes
```

■ If the Java code was successfully compiled, the command prompt re-appears.

6 Type the code that copies the servlet class file to the appropriate directory on your Web server.



```
Address: http://127.0.0.1:8080/examples/SimpleServlet

This is a Servlet
```

7 In a Web browser window, enter the URL of the servlet you want to execute.

■ The Web browser displays the results of executing the servlet.

GENERATE A WEB PAGE USING A SERVLET

A servlet can be used to generate HTML code that is rendered as a Web page when viewed in a Web browser. Generating HTML code from a servlet allows you to use features of the Java programming language, such as variables, methods and objects to create Web pages. Information for the Web pages can be retrieved from files, databases or other Web pages. To generate Web pages from a servlet, you simply send strings of data containing valid HTML code to the client using the `print` method of the `out` object.

Before sending HTML data to a client, the `response` object can be instructed to send an HTTP header. HTTP headers provide instructions and information to the client application, usually a Web browser. For example, the `setContentType` method of the `response` object can be used to specify the content type of information to be sent to the client. The `"text/html"` argument can be passed to the `setContentType` method to inform the client to expect HTML information.

HTML pages typically start with a document type declaration. The document type declaration allows the Web browser to determine what type of document is being viewed by specifying which version of the HTML specification the Web browser should use to render the Web page. While most browsers will display Web pages that do not have a document type declaration, some applications, such as search engines, rely on the document type declaration to better classify Web pages. You can find information about the document type declaration for the version of HTML you use at the www.w3.org Web site.

When generating a Web page using a servlet, the `setContentType` method and the document type declaration should be sent before any other HTML code is sent to the client.

Apply It

You can split long strings of data over multiple lines by using the concatenation operator (+) to break up the strings into multiple parts.

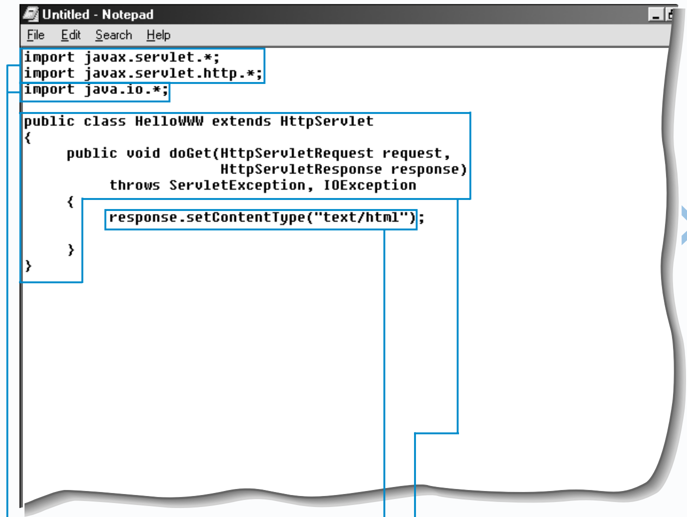
Example:

```
out.print("<!doctype html public" +
          " \"-//w3c//dtd html 4.0 " +
          "transitional//en\">" +
          "<html><head>" +
          "<title>Servlet Generated Web Page</title>" +
          "</head><body>" +
          "<h1>Servlet Generated Web Page</h1>" +
          "</body></html>");
```

You can use the `setContentType` method to indicate that a page contains other types of content besides HTML information. This chart displays the most common content types.

CONTENT TYPE:	DESCRIPTION:
`text/html`	Page contains HTML code.
`text/plain`	Page contains only plain text.
`audio/basic`	Page contains an audio file.
`video/mpeg`	Page contains a video file.
`image/gif`	Page contains a GIF image.
`image/jpeg`	Page contains a JPEG image.
`application/pdf`	Page contains a PDF file.
`application/msword`	Page contains a Word document.

GENERATE A WEB PAGE USING A SERVLET



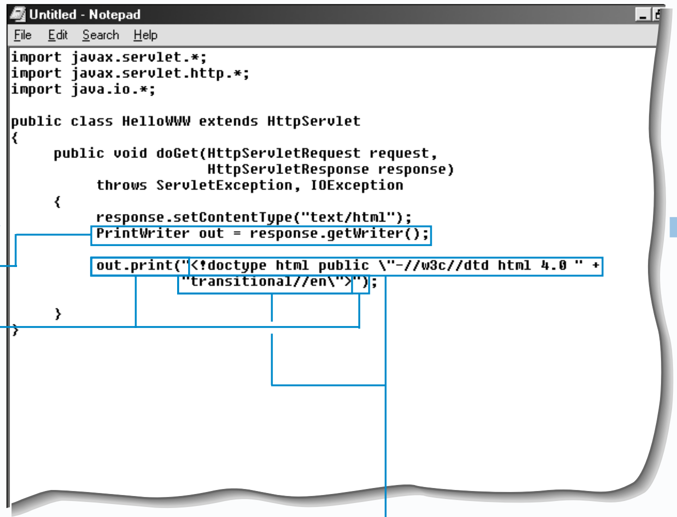
```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class HelloWWW extends HttpServlet
{
     public void doGet(HttpServletRequest request,
                       HttpServletResponse response)
          throws ServletException, IOException
     {
          response.setContentType("text/html");

     }
}
```

1 Type the code that imports the `javax.servlet` and the `javax.servlet.http` packages.

2 Type the code that imports any additional packages needed by the servlet.

3 Type the code that creates the servlet class and the `doGet` method.

4 To specify that the document will contain HTML information, type **response.setContentType ("text/html")**.

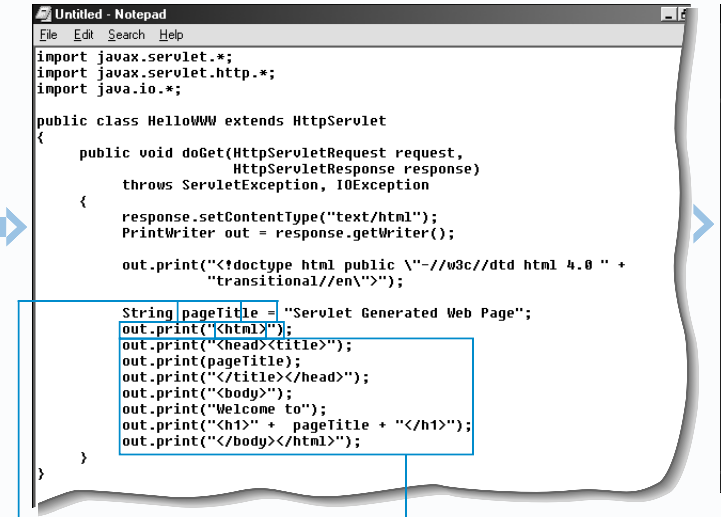


```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class HelloWWW extends HttpServlet
{
     public void doGet(HttpServletRequest request,
                       HttpServletResponse response)
          throws ServletException, IOException
     {
          response.setContentType("text/html");
          PrintWriter out = response.getWriter();

          out.print("<!doctype html public \"-//w3c//dtd html 4.0 " +
                    "transitional//en\">");

     }
}
```

5 Type the code that creates a `PrintWriter` object from the `getWriter` method of the `response` object.

6 To send a document type declaration to the client application, type **out.print("")**.

7 Between the quotation marks, type the appropriate document type declaration for the version of the HTML specification the Web browser should use to render the Web page.



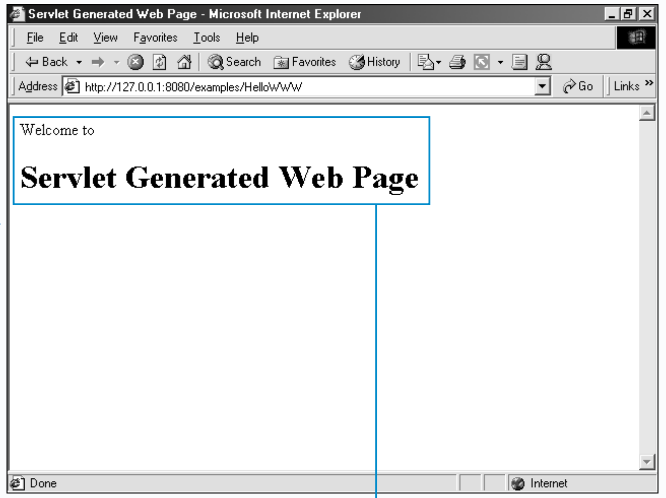
```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class HelloWWW extends HttpServlet
{
     public void doGet(HttpServletRequest request,
                       HttpServletResponse response)
          throws ServletException, IOException
     {
          response.setContentType("text/html");
          PrintWriter out = response.getWriter();

          out.print("<!doctype html public \"-//w3c//dtd html 4.0 " +
                    "transitional//en\">");

          String pageTitle = "Servlet Generated Web Page";
          out.print("<html>");
          out.print("<head><title>");
          out.print(pageTitle);
          out.print("</title></head>");
          out.print("<body>");
          out.print("Welcome to");
          out.print("<h1>" +  pageTitle + "</h1>");
          out.print("</body></html>");
     }
}
```

8 To generate HTML code, type **out.print("")**.

9 Between the quotation marks, type the HTML code you want to use for the Web page.

10 Repeat steps 8 and 9 to generate the HTML code needed to render the Web page.

■ You can use variables, methods and objects to generate dynamic HTML code.


Servlet Generated Web Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://127.0.0.1:8080/examples/HelloWWW
Go Links
Done
Internet

Welcome to

Servlet Generated Web Page

11 Save the file with the .java extension and then compile the source code for the file.

12 Copy the compiled class file to the appropriate directory on your Web server.

13 Display the servlet in a Web browser.

■ The Web browser displays the Web page generated by the servlet.

CREATE A SERVLET THAT ACCESSES CGI VARIABLES

The Common Gateway Interface (CGI) is a standard used to create interactive Web sites. Servlets are often used to replace CGI applications on a Web server.

Like the CGI applications they replace, servlets often need to determine information about the environment in which the servlet is running. For example, a servlet may need to determine the type of Web server on which it is running before performing an action such as writing to a database. Information about the environment is typically stored in CGI, or environment, variables, which store information about the computer, Web server and operating system that is running the servlet.

When you create servlets to replace CGI applications, the servlets will need to access the CGI variables. Because servlets are created using the Java programming language, which was not developed specifically for use on Web servers, servlets do not contain a built-in method for accessing CGI variables. Servlets can access CGI variables only by using objects, such as the `HttpServletRequest` object, to retrieve information.

Methods of the `HttpServletRequest` object, which handles the request information passed between a client and a servlet, are the most common methods used to access CGI variable information. For example, to access the address of the client that made a request, you use the `getRemoteAddr()` method of the `HttpServletRequest` object.

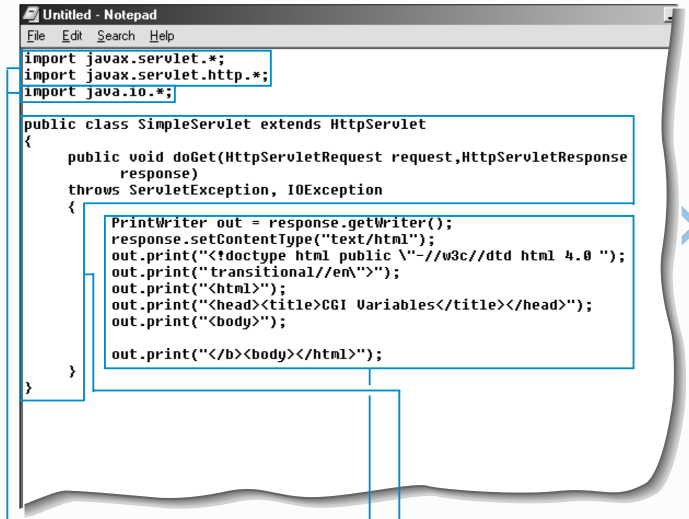
Methods of the `ServletContext` object, such as the `getServerInfo()` method, can be used to obtain information about the Web server on which the servlet is running, such as the name and version of the Web server software and the operating system running the server.

Extra

The following is a list of some commonly used CGI variables and the methods used to access the variables in a servlet.

CGI VARIABLE:	METHOD REQUIRED:
AUTH_TYPE	request.getAuthType()
CONTENT_TYPE	request.getContentType()
DOCUMENT_ROOT	getServletContext().getRealPath("/")
PATH_INFO	request.getPathInfo()
QUERY_STRING	request.getQueryString()
REMOTE_ADDR	request.getRemoteAddr()
REMOTE_HOST	request.getRemoteHost()
REMOTE_USER	request.getRemoteUser()
SCRIPT_NAME	request.getServletPath()
SERVER_NAME	request.getServerName()
SERVER_PORT	request.getServerPort()
SERVER_SOFTWARE	getServletContext().getServerInfo()

CREATE A SERVLET THAT ACCESSES CGI VARIABLES



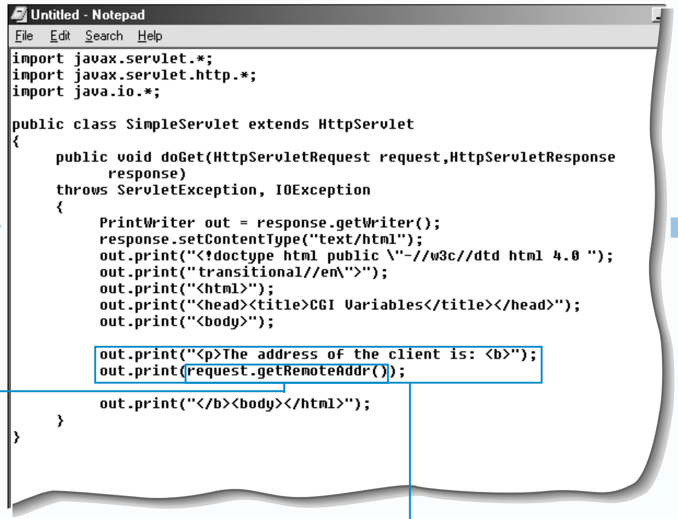
```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class SimpleServlet extends HttpServlet
{
     public void doGet(HttpServletRequest request,HttpServletResponse
          response)
     throws ServletException, IOException
     {
          PrintWriter out = response.getWriter();
          response.setContentType("text/html");
          out.print("<!doctype html public \"-//w3c//dtd html 4.0 ");
          out.print("transitional//en\">");
          out.print("<html>");
          out.print("<head><title>CGI Variables</title></head>");
          out.print("<body>");

          out.print("</b><body></html>");
     }
}
```

1 Type the code that imports the `javax.servlet` and `javax.servlet.http` packages.

2 Type the code that imports any additional packages needed by the servlet.

3 Type the code that creates the servlet class and the `doGet` method.

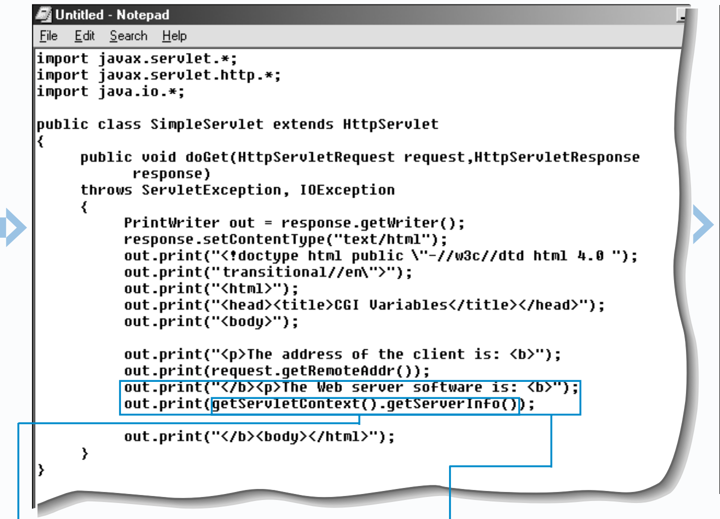
4 Type the code that sends HTML data to the client. For information about generating a Web page, see page 232.



```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class SimpleServlet extends HttpServlet
{
     public void doGet(HttpServletRequest request,HttpServletResponse
          response)
     throws ServletException, IOException
     {
          PrintWriter out = response.getWriter();
          response.setContentType("text/html");
          out.print("<!doctype html public \"-//w3c//dtd html 4.0 ");
          out.print("transitional//en\">");
          out.print("<html>");
          out.print("<head><title>CGI Variables</title></head>");
          out.print("<body>");

          out.print("<p>The address of the client is: <b>");
          out.print(request.getRemoteAddr());

          out.print("</b><body></html>");
     }
}
```

5 To access the address of the client that made a request, type **request.getRemoteAddr()**.

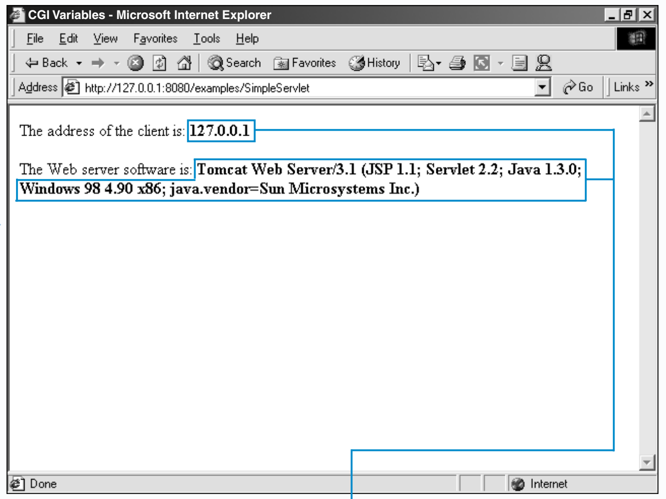
6 Type the code that uses the information from the CGI variable.



```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class SimpleServlet extends HttpServlet
{
     public void doGet(HttpServletRequest request,HttpServletResponse
          response)
     throws ServletException, IOException
     {
          PrintWriter out = response.getWriter();
          response.setContentType("text/html");
          out.print("<!doctype html public \"-//w3c//dtd html 4.0 ");
          out.print("transitional//en\">");
          out.print("<html>");
          out.print("<head><title>CGI Variables</title></head>");
          out.print("<body>");

          out.print("<p>The address of the client is: <b>");
          out.print(request.getRemoteAddr());
          out.print("</b><p>The Web server software is: <b>");
          out.print(getServletContext().getServerInfo());

          out.print("</b><body></html>");
     }
}
```

7 To access information about the Web server, type **getServletContext().getServerInfo()**.

8 Type the code that uses the information from the CGI variable.



CGI Variables - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://127.0.0.1:8080/examples/SimpleServlet
Go Links
Done
Internet

The address of the client is: **127.0.0.1**

The Web server software is: **Tomcat Web Server/3.1 (JSP 1.1; Servlet 2.2; Java 1.3.0; Windows 98 4.90 x86; java.vendor=Sun Microsystems Inc.)**

9 Save the file with the .java extension and then compile the source code for the file.

10 Copy the compiled class file to the appropriate directory on your Web server.

11 Display the servlet in a Web browser.

■ The Web browser displays the result of accessing CGI variable information using the servlet.

PROCESS FORM DATA USING A SERVLET

Using a servlet is a very effective method of processing data passed by a form. Servlets process data faster and are more efficient than CGI applications.

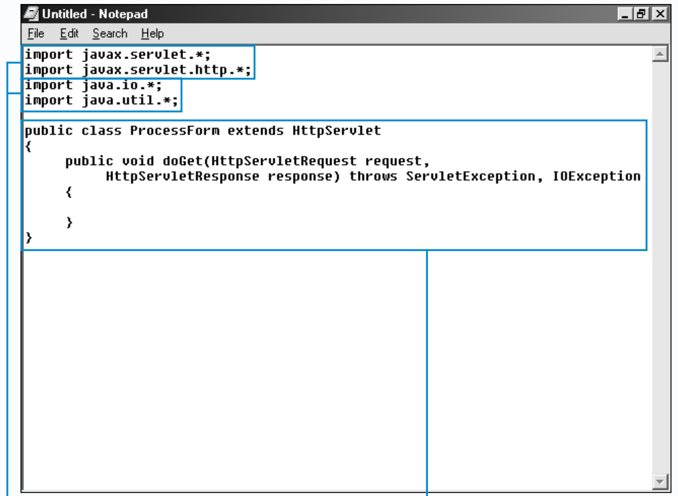
In addition to data passed by a form, a servlet can also be used to process data submitted by a query string. A query string is one or more name and value pairs appended to a URL. To submit a query string to a servlet, you enter the URL of the servlet in a Web browser, followed by a question mark. You then enter a name followed by an equal sign and a value for the name. To enter multiple name and value pairs, separate each pair with an ampersand (&), such as ?userName=Martine&id=123. A query string should not contain spaces.

In order for a servlet to process data passed by a form using the `get` method or a query string, a `doGet` method must be created in the servlet. The `getParameter` method of the `HttpServletRequest` object can be used in the `doGet` method to access form or query string data. The argument for the `getParameter` method is the name of the form element you want to access or a name specified in the query string.

Once the servlet has retrieved the data from a form or a query string, the data can be displayed in a Web browser or stored in a variable for later use. A servlet may also perform a more complex task, such as placing the data in a database or writing the data to file.

Before the servlet file can be used to process form data, the file must be saved with the .java extension and then compiled. After saving the servlet, you should review the code for the form to verify that the `action` attribute of the `<form>` tag displays the correct filename and location of the servlet.

PROCESS FORM DATA USING A SERVLET



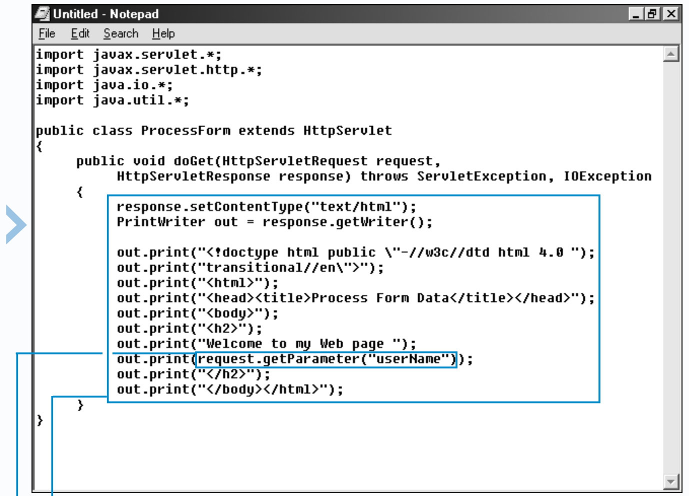
Untitled - Notepad

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;

public class ProcessForm extends HttpServlet
{
     public void doGet(HttpServletRequest request,
          HttpServletResponse response) throws ServletException, IOException
     {

     }
}
```

1 Type the code that imports the `javax.servlet` and `javax.servlet.http` packages.

2 Type the code that imports any additional packages needed by the servlet.

3 Type the code that creates the servlet class and the `doGet` method.



Untitled - Notepad

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;

public class ProcessForm extends HttpServlet
{
     public void doGet(HttpServletRequest request,
          HttpServletResponse response) throws ServletException, IOException
     {
          response.setContentType("text/html");
          PrintWriter out = response.getWriter();

          out.print("<!doctype html public \"-//w3c//dtd html 4.0 ");
          out.print("transitional//en\">");
          out.print("<html>");
          out.print("<head><title>Process Form Data</title></head>");
          out.print("<body>");
          out.print("<h2>");
          out.print("Welcome to my Web page ");
          out.print(request.getParameter("userName"));
          out.print("</h2>");
          out.print("</body></html>");
     }
}
```

4 Type the code that sends HTML data to the client. For information about generating a Web page, see page 232.

5 Type the code that accesses data passed by a form.

6 Save the file with the .java extension and then compile the source code for the file.

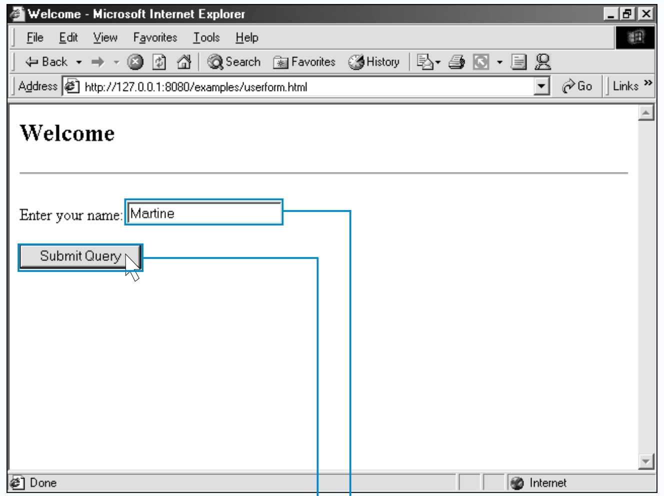
7 Copy the compiled class file to the appropriate directory on your Web server.

Extra

There are two methods a form can use to pass information to a servlet—`get` and `post`. The `get` method is faster than the `post` method and is suitable for small forms. The `post` method is suitable for large forms that will send more than 2000 characters to the servlet. When the `post` method is used by a form, a `doPost` method must be created in the servlet. You can then pass the information from the `doPost` method to the `doGet` method. This creates a servlet that can handle information passed using either the `post` or `get` method.

Example:

```
public void doGet(HttpServletRequest request,
     HttpServletResponse response) throws ServletException, IOException
{
     response.setContentType("text/html");
     PrintWriter out = response.getWriter();

     out.print("<!doctype html public \"-//w3c//dtd html 4.0 ");
     out.print("transitional//en\">");
     out.print("<html>");
     out.print("<head><title>Process Form Data</title></head>");
     out.print("<body>");
     out.print("Welcome to my Web page");
     out.print(request.getParameter("userName"));
     out.print("</body></html>");
}
public void doPost(HttpServletRequest request, HttpServletResponse response)
     throws ServletException, IOException
{
     doGet(request, response);
}
```



Welcome - Microsoft Internet Explorer

Address http://127.0.0.1:8080/examples/userform.html

Welcome

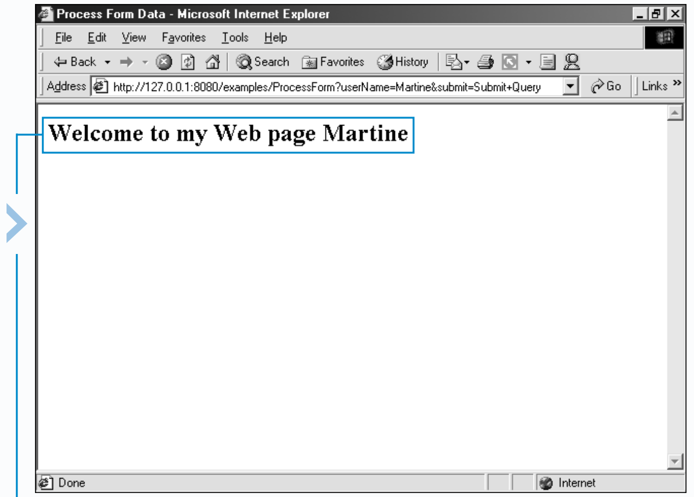
Enter your name: Martine

PROCESS FORM DATA

1 In a Web browser, display the Web page containing the form you want to process.

2 Enter data into the form.

3 Click the Submit button to pass the data in the form to the servlet.



Process Form Data - Microsoft Internet Explorer

Address http://127.0.0.1:8080/examples/ProcessForm?userName=Martine&submit=Submit+Query

Welcome to my Web page Martine

■ The Web browser displays the result of processing form data using a servlet.