

Notes from Well House Consultants

These notes are written by Well House Consultants and distributed under their Open Training Notes License. If a copy of this license is not supplied at the end of these notes, please visit

*<http://www.wellho.net/net/whcotnl.html>
for details.*

1.1 Well House Consultants

Well House Consultants provides niche training, primarily but not exclusively in Open Source programming languages. We offer public courses at our training centre and private courses at your offices. We also make some of our training notes available under our "Open Training Notes" license, such as we're doing in this document here.

1.2 Open Training Notes License

With an "Open Training Notes License", for which we make no charge, you're allowed to print, use and distribute these notes provided that you retain the complete and unaltered license agreement with them, including our copyright statement. This means that you can learn from the notes, and have others learn from them too.

You are NOT allowed to charge (directly or indirectly) for the copying or distribution of these notes, nor are you allowed to charge for presentations making any use of them.

1.3 Courses presented by the author

If you would like us to attend a course (Java, Perl, Python, PHP, Tcl/Tk, MySQL or Linux) presented by the author of these notes, please see our public course schedule at

<http://www.wellho.net/course/index.html>

If you have a group of 4 or more trainees who require the same course at the same time, it will cost you less to have us run a private course for you. Please visit our onsite training page at

<http://www.wellho.net/course/otc.html>

which will give you details and costing information

1.4 Contact Details

Well House Consultants may be found online at

<http://www.wellho.net>

graham@wellho.net

technical contact

lisa@wellho.net

administration contact

Our full postal address is

404 The Spa

Melksham

Wiltshire

UK SN12 6QL

Phone +44 (0) 1225 708225

Fax +44 (0) 1225 707126

Servlets

Web applications in Java can be provided by Servlet classes running in a container on a web server. In this module, you'll learn how to write simple servlets that allow web users (whether or not they have Java in their browser) to interact with your Java applications.

<i>What is a Servlet?</i>	4
<i>A first servlet.</i>	5
<i>Reusing a servlet</i>	10

2.1 What is a Servlet?

So far, we've studied

- applications
 - a main method
 - run using a virtual machine such as "java" from JDK
 - typically inputting from STDIN (keyboard) and output to screen (via System.out)
 - each application is a class in its own right
- applets
 - extends `java.applet.Applet`
 - methods such as `init`, `start` and `paint`
 - run on the web on a client system by the browser
 - typically interacting using the AWT (abstract windowing toolkit)

Applets are ideal for client-side operations, such as dynamic graphing, local calculations, etc. An applet can contact the server from which it was downloaded but this does not provide a clean solution for core server-side applications.

If you wish to complete a form on a web page¹ and run a Java program on your server when you press the "Submit" button, you can do so using a servlet.

Servlets are run on server systems. Certain web servers run them, or they can be run on a separate program (on a different port to the main server) using the `servletrunner` program supplied with the JSDK (Java Servlet Development Kit).

Running the server

The server must be running and accepting connections before the client application calls for the service. Here's an example of how we start `servletrunner` which you might like to use as a test:

```
seal% ./bin/servletrunner &
servletrunner starting with settings:
  port = 8080
  backlog = 50
  max handlers = 100
  timeout = 5000
  servlet dir = ./examples
  document dir = ./examples
  servlet propfile = ./examples/servlet.properties
svl_right: init
```

Servers which support servlets will each have their own individual configuration files. Refer to your server administrator or the documentation provided with your server for details of this. Note that only a few servers actually provide support for servlets.²

Although as a web material provider, you're limited in the choice of servers that you use to provide servlets, they can be called up just like a normal web page from any browser. You don't have to have a special version of a browser. Thus, with the correct server, you can make servlets available to anyone on the web!

Servlets extend `javax.servlet.HttpServlet`

A servlet programmer's role is to provide methods such as `doGet` which will collect inputs from a GET request (e.g. fields from a form) and generate output (i.e. a response page) in HTML.

Other methods are available in classes such as `javax.servlet.http.*` to

¹ just a normal form written in HTML

² as of this writing

access information such as the URL that was called, and data entered onto a form.

2.2 A first servlet

This first servlet example takes data that the user has entered onto a form and echoes it back as part of the response screen. A simple calculation is made to show that the servlet is really running for demo purposes.

The HTML form

There's nothing special about the form; the same form could be used equally well for:

- a CGI (Common Gateway Interface) application with the server side programming written in C or C++ or Perl
- Active Server Pages technology, with the server performing an asp including VBScript or Perlscript.

Here's a standard piece of HTML which displays as shown in Figure 1.

```
<html>
<head>
<title>Well House Consultants - Servlet demo</title>
</head>
<body bgcolor=white>
<center>
<H1>Form to complete</H1>
<H3>Simple version</H3>
<form action=http://seal:8080/servlet/svl_right>
<table>
<tr>
<td>Identify yourself:</td>
<td><INPUT name=you></td>
</tr>
<tr>
<td>Number of persons:</td>
<td>
<select name=howmany>
<option value=1>One</option>
<option value=2>Two</option>
<option value=3>Three</option>
<option value=4>Four</option>
<option value=5>Over Four</option>
</select>
</td>
</tr>
<tr>
<td>Area:</td>
<td>
<select name=cover>
<option value=worldwide>World</option>
<option value=european>Europe</option>
</select>
</td>
</tr>
<tr>
<td>Finally select:</td>
<td><input type=submit></td>
</tr>
</table>
</form>
```

```
</center>
<p>Note - in order for this form to operate, "Servletrunner" must
be running on port 8080 on seal before you press the submit
button.</p>
<p>
Demo by Well House Consultants<br>
www.wellho.co.uk<br>
01225 708225
</body>
</html>
```

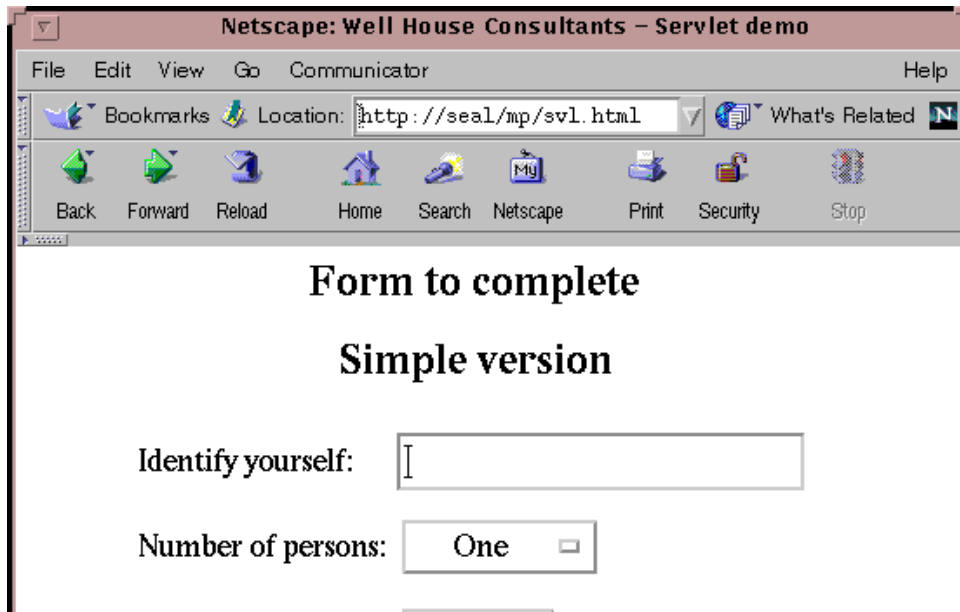


Figure 1 A sample HTML form

The servlet

When the submit button on the form is pressed, the browser contacts a machine (in this case, "seal") on service (port number) 8080 using standard http (hypertext transfer) protocols.

For the purposes of this example, we have servletrunner already waiting on that port.

When `servletrunner` receives the connection it:

- Starts the particular servlet (if it's the first call to the servlet).
- Makes information about the request (e.g. requesting client, contents of form fields) available.
- Runs the `doGet` (or `doPost` if the form was posted) method within the servlet.
- Passes the response back to the browser, adding appropriate headers so that the browser knows how to handle the response.
- When completed, it drops the connection.

Here's the code of the servlet:

```

/* Servlet to read back and echo to a second frame */

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

/* written by graham@wellho.co.uk */
/* Java and internet training and consultancy */

public class svl_right extends HttpServlet
{
    public void doGet (
        HttpServletRequest      request,
        HttpServletResponse    response
    ) throws ServletException, IOException
    {
        PrintWriter            out;
        String                  title = "Well House Consultants";

        String user_name = request.getParameter("you");
        String adults = request.getParameter("owmany");
        String coverwanted = request.getParameter("cover");
        int ac2 = 1+ Integer.parseInt(adults);

        // set content type and other response header fields first
        response.setContentType("text/html");

        // then write the data of the response
        out = response.getWriter();

        out.println("<html><head><title>");
        out.println(title);
        out.println("</title></head><body bgcolor=yellow>");
        out.println("<center><H1>Result Window</H1>");
        out.println("<p>This is your response window.<p>");
        out.println ("Called from "+HttpUtils.getRequestURL (request).toString ());
        out.println ("<p>User called "+user_name+"<br>");
        out.println ("Persons "+adults+"<br>");
        out.println ("Range "+coverwanted+"<p>");
        out.println ("<em>Addanadult gives you "+ac2+" persons!</em><br>");
        out.println("</center></body></html>");
        out.close();
    }
}

```

Here's the source of the resulting page shown in Figure 2, for reference:

```
<html>
<head>
<title>Well House Consultants</title>
</head>
<body bgcolor=yellow>
<center>
<H1>Result Window</H1>
<p>This is your response window.</p>
<p>Called from http://seal:8080/servlet/svl_right</p>
<p>
User called Graham Ellis<br>
Persons 3<br>
Range european
</p>
<p>
<em>Addanadult gives you 4 persons!</em><br>
</center>
</body>
</html>
```

Figure 2 Response window of the servlet

Comments on the servlet:

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

We're importing the io classes so that we can write an output stream back to the browser, the servlet classes so that we can extend the `HttpServlet`, and the http classes so that we can access the various parameters of the hypertext connection.

```
public class svl_right extends HttpServlet
{
```

As this is a servlet, we're extending the `HttpServlet` class which is provided as a part of the JSDK (Java Servlet Development Kit), rather than writing our own "main" application.

```
public void doGet (
    HttpServletRequest    request,
    HttpServletResponse    response
) throws ServletException, IOException
{
```

We're overriding the default `doGet` method with our own, which takes two parameters:

- An object containing information about the incoming request
- An object containing information about our response.

We are obliged to handle various exceptions ...

```
String user_name = request.getParameter("you");
String adults = request.getParameter("owmany");
String coverwanted = request.getParameter("cover");
int ac2 = 1+ Integer.parseInt(adults);
```

Here we have captured three fields from the incoming form, using the `getParameter` method. All form contents are returned to us as strings; we have elected to convert one of the incoming strings into an integer and do a simple calculation on it.

In a complete example, we would include checks that the user has entered sensible data into the form at this point, and indeed that all the fields that we need do exist, and that there are no unexpected, spurious fields. Such checking is vital because user input tends to be very unreliable, and malicious visitors to your site may try all sorts of tricks to break your server/get hold of all your data.

```
response.setContentType("text/html");
```

We have to set the response type. It doesn't have to be an HTML page; it could be text/plain, it could be an image, it could be a download to save to disk on the browser's system, or it could even be a sound!

The content type must be set before we:

```
PrintWriter out;
out = response.getWriter();
```

Opens an output stream back to the browser.

```
out.println("<html><head><title>");
out.println(title);
out.println("</title></head>
    <body bgcolor=yellow>");
out.println("<center><H1>Result Window</H1>");
out.println("<p>This is your response window.</p>");
```

We've started to generate the reply page. You must ensure that this is valid HTML and is going to look good on all the types of browsers that your users may have. On an internal intranet, you may be able to support only one flavour of browser, but on the WorldWide web any HTML codes that are browser-specific may limit your audience.

```
out.println ("Called from "+HttpUtils.getRequestURL (request).toString ());
```

Note that we can call back to the incoming request for further information; in this case, we're reporting the URL through which the servlet was accessed.

```
out.println ("<p>User called "+user_name+"<br>");
out.println ("Persons "+adults+"<br>");
out.println ("Range "+coverwanted+"</p>");
out.println ("<em>Addanadult gives you "+ac2+" persons!</em><br>");
out.println("<center></body></html>");
```

The HTML output page has now been completed.

```
out.close();
}
```

Close the connection (which causes the page to be flushed out to the browser) and end the `doGet` method. Job completed!

2.3 Reusing a servlet

A single servlet can be called up from many different web pages so if you want the same task performed with a number of front ends, that's fine.

Form and results in same window

The following example reuses the same servlet, but now places the form in the left-hand frame of a window and the results are displayed to the right.

Figure 3 Same servlet being used in HTML frames

Since we're using frames, we now have three separate HTML files:

- The Frameset document

```
<html>
<head>
<title>Well House / Servlet demo</title>
</head>
<frameset cols="*,*" frameborder=0>
<frame src="svl_l2.html" name=entry noresize scrolling=auto marginheight=0 marginwidth=0>
```

```

    <frame src="svl_temp.html" name=results scrolling=auto>
  </frameset>
</html>

```

- The Left frame (including the form)

```

<head></head>
<body bgcolor=white>
<center>
<H1>Form to complete</H1>
<H3>Batch version</H3>
<form action=http://seal:8080/servlet/svl_right target=results>
Identify yourself: <input name=you><br>
<select name=howmany>
<option value=1>One</option>
<option value=2>Two</option>
<option value=3>Three</option>
<option value=4>Four</option>
<option value=5>Over Four</option>
</select>
<p>
<select name=cover>
<option value=worldwide>World</option>
<option value=european>Europe</option>
</select>
</p>
<p>
<input type=submit>
</form>
</p>
Demo by Well House Consultants<br>
www.wellho.co.uk<br>
01225 708225
</center>
</body>
</html>

```

- The initial right frame (which is replaced by the result of running our servlet)

```

<body bgcolor=red>
<center><H1>Results will come here</H1></center>
</body>

```

Figure 4 The results after completing the form and submitting the query

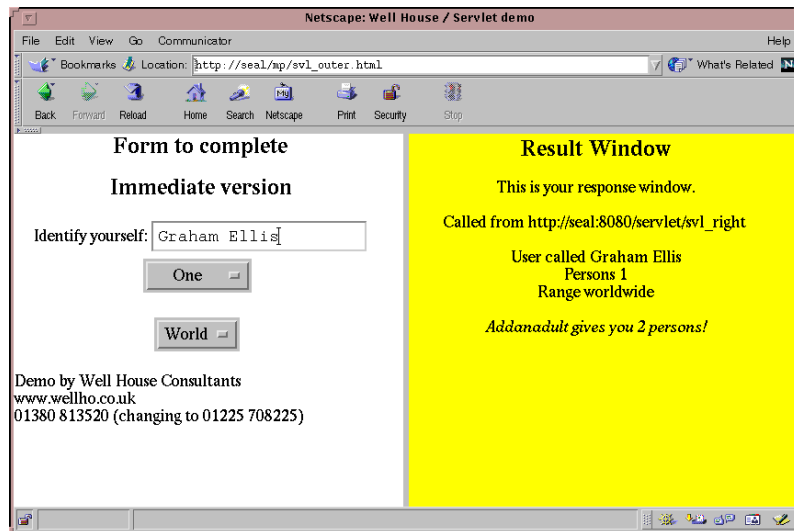
Interactive form

In that last example, our form was using a submit button to update the response information. Using the web, that's often the best approach since it keeps web traffic down (and thus the user's response speed up) and it will work with any browser.

If you're sure that your users will have a reasonable connection speed and be able to browse with a modern browser, you might choose to make the whole page appear interactive by submitting the form each time a box is completed. The only change necessary to make this work is to include some JavaScript in the form:

```
<html>
<head>
<script language="Javascript">
<!--hide
function leapnow(zing)
{ zing.submit() }
//End of hide -->
</script>
</head>
<body bgcolor=white>
<center>
<p>
<H1>Form to complete</H1>
<H3>Immediate version</H3>
<form action=http://seal:8080/servlet/svl_right target=results>
Identify yourself: <input name=you onChange=leapnow(this.form)><br>
<select name=owmany onChange=leapnow(this.form)>
<option value=1>One</option>
<option value=2>Two</option>
<option value=3>Three</option>
<option value=4>Four</option>
<option value=5>Over Four</option>
</select>
</p>
<p>
<select name=cover onChange=leapnow(this.form)>
<option value=worldwide>World</option>
<option value=european>Europe</option>
</select>
</p>
</form>
</center>
<p>
Demo by Well House Consultants<BR>
www.wellho.co.uk<BR>
01225 708225
</p>
</body>
</html>
```

Figure 5 Interactive results each time a box is completed



Of course, the best way to see how this one works is to use it, but for reference a sample screen is shown in Figure 5.

 **Exercise**

License

*These notes are distributed under the **Well House Consultants Open Training Notes License**. Basically, if you distribute it and use it for free, we'll let you have it for free. If you charge for its distribution of use, we'll charge.*

3.1 Open Training Notes License

Training notes distributed under the **Well House Consultants Open Training Notes License** (WHCOTNL) may be reproduced for any purpose PROVIDE THAT:

- This License statement is retained, unaltered (save for additions to the change log) and complete.
- No charge is made for the distribution, nor for the use or application thereof. This means that you can use them to run training sessions or as support material for those sessions, but you cannot then make a charge for those training sessions.
- Alterations to the content of the document are clearly marked as being such, and a log of amendments is added below this notice.
- These notes are provided "as is" with no warranty of fitness for purpose. Whilst every attempt has been made to ensure their accuracy, no liability can be accepted for any errors of the consequences thereof.

Copyright is retained by Well House Consultants Ltd, of 404, The Spa, Melksham, Wiltshire, UK, SN12 6QL - phone number +44 (1) 1225 708225. Email contact - Graham Ellis (graham@wellho.net).

Please send any amendments and corrections to these notes to the Copyright holder - under the spirit of the Open Distribution license, we will incorporate suitable changes into future releases for the use of the community.

If you are charged for this material, or for presentation of a course (Other than by Well House Consultants) using this material, please let us know. It is a violation of the license under which this notes are distributed for such a charge to be made, except by the Copyright Holder.

If you would like Well House Consultants to use this material to present a training course for your organisation, or if you wish to attend a public course is one is available, please contact us or see our web site - <http://www.wellho.net> - for further details.

Change log
Original Version, Well House Consultants, 2004

Updated by: _____ on _____
Updated by: _____ on _____
Updated by: _____ on _____
Updated by: _____ on _____
Updated by: _____ on _____
Updated by: _____ on _____

License Ends.