

4.3 Example: Reading Three Parameters

[Listing 4.1](#) presents a simple servlet called `ThreeParams` that reads form parameters named `param1`, `param2`, and `param3` and places their values in a bulleted list. Although you are required to specify `response` settings (see [Chapters 6](#) and [7](#)) before beginning to generate the content, you are not required to read the `request` parameters at any particular place in your code. So, we read the parameters only when we are ready to use them. Also recall that since the `ThreeParams` class is in the `coreservlets` package, it is deployed to the `coreservlets` subdirectory of the `WEB-INF/classes` directory of your Web application (the default Web application in this case).

As we will see later, this servlet is a perfect example of a case that would be dramatically simpler with JSP. See [Section 11.6](#) (Comparing Servlets to JSP Pages) for an equivalent JSP version.

Listing 4.1 ThreeParams.java

```
package coreservlets;

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

/** Simple servlet that reads three parameters from the
 *  form data.
 */

public class ThreeParams extends HttpServlet {
    public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        String title = "Reading Three Request Parameters";
        String docType =
            "<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 " +
            "Transitional//EN\">\n";
        out.println(docType +
                    "<HTML>\n" +
                    "<HEAD><TITLE>" + title + "</TITLE></HEAD>\n" +
                    "<BODY BGCOLOR=#FDF5E6>\n" +
                    "<H1 ALIGN= CENTER>" + title + "</H1>\n" +
                    "<UL>\n" +
                    "  <LI><B>param1</B>: " +
                    "+ request.getParameter("param1") + "\n" +
                    "  <LI><B>param2</B>: " +
                    "+ request.getParameter("param2") + "\n" +
                    "  <LI><B>param3</B>: " +
                    "+ request.getParameter("param3") + "\n" +
                    "</UL>\n" +
                    "</BODY></HTML> ");
    }
}
```

[Listing 4.2](#) shows an HTML form that collects user input and sends it to this servlet. By using an `ACTION` URL beginning with a slash (`/servlet/coreservlets.ThreeParams`), you can install the form anywhere in the default Web application; you can move the HTML form to another directory or move both the HTML form and the servlet to another machine, all without editing the HTML form or the servlet. The general principle that form URLs beginning with slashes increases portability holds true even when you use custom Web applications, but you have to include the Web application prefix in the URL. See [Section 2.11](#) (Web Applications: A Preview) for details on Web applications. There are other ways to write the URLs that also simplify portability, but the most important point is to use relative URLs (no host name), not absolute ones (i.e., `http://host/...`). If you use absolute URLs, you have to edit the forms whenever you move the Web application from one machine to another. Since you almost certainly develop on one machine and deploy on another, use of absolute URLs should be

strictly avoided.

Core Approach



Use form ACTION URLs that are relative, not absolute.

Listing 4.2 ThreeParamsForm.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML><HEAD><TITLE>Collecting Three Parameters</TITLE></HEAD>
<BODY BGCOLOR="#FDF5E6">
<H1 ALIGN="CENTER">Collecting Three Parameters</H1>

<FORM ACTION="/servlet/coreservlets.ThreeParams">
  First Parameter: <INPUT TYPE="TEXT" NAME="param1"><BR>
  Second Parameter: <INPUT TYPE="TEXT" NAME="param2"><BR>
  Third Parameter: <INPUT TYPE="TEXT" NAME="param3"><BR>
  <CENTER><INPUT TYPE="SUBMIT"></CENTER>
</FORM>

</BODY></HTML>
```

Recall that the location of the default Web application varies from server to server. HTML forms go in the top-level directory or in subdirectories other than `WEB-INF`. If we place the HTML page in the `form-data` subdirectory and access it from the local machine, then the full installation location on the three sample servers used in the book is as follows:

- **Tomcat Location**

`install_dir/webapps/ROOT/form-data/ThreeParamsForm.html`

- **JRun Location**

`install_dir/servers/default/default-ear/default-war/form-data/ThreeParamsForm.html`

- **Resin Location**

`install_dir/doc/form-data/ThreeParamsForm.html`

- **Corresponding URL**

`http://localhost/form-data/ThreeParamsForm.html`

Figure 4-1 shows the HTML form when the user has entered the home directory names of three famous Internet personalities. OK, OK, only two of them are famous,^[1] but the point here is that the tilde (~) is a nonalphanumeric character and will be URL-encoded by the browser when the form is submitted. Figure 4-2 shows the result of the servlet; note the URL-encoded values on the address line but the original form field values in the output: `getParameter` always returns the values as the end user typed them in, regardless of how they were sent over the network.

[1] Gates isn't that famous, after all.

Figure 4-1. Front end to parameter-processing servlet.

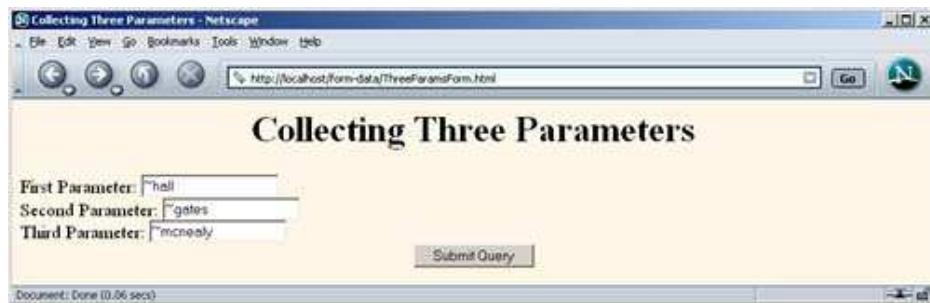


Figure 4-2. Result of parameter-processing servlet: request parameters are URL-decoded automatically.



[Team LiB]