



Servlet and JSP Review

A Recap of the Basics

JSP, Servlet, Struts, JSF, AJAX, & Java 5 Training: <http://courses.coreservlets.com>
J2EE Books from Sun Press: <http://www.coreservlets.com>



For live J2EE training, see training courses on JSP, servlets, Struts, JSF, AJAX, and Java 5 at <http://courses.coreservlets.com/>.



Taught by the author of *Core Servlets and JSP*, *More Servlets and JSP*, and this tutorial. Available at public venues, or customized versions can be held on-site at your organization.
Additional topics available upon request.

Agenda

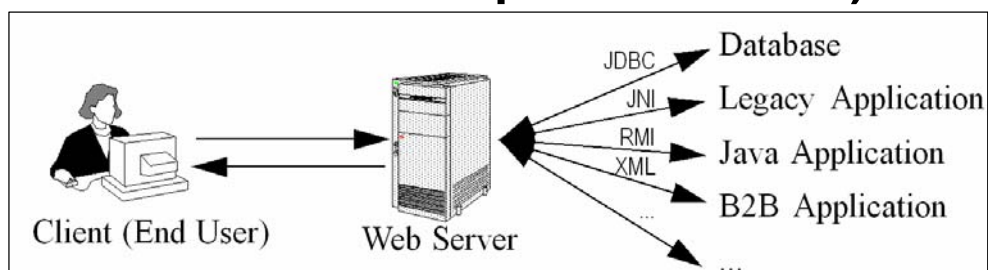
- **The Invoker Servlet**
- **The Default Web App**
- **Packageless Servlets**
- **Packaged Servlets**
- **Form Data**
- **JSP Scripting**
 - JSP Expressions
 - JSP Scriptlets
 - JSP Declarations

4

J2EE training: <http://courses.coreservlets.com>

A Servlet's Job

- **Read explicit data sent by client (form data)**
- **Read implicit data sent by client (request headers)**
- **Generate the results**
- **Send the explicit data back to client (HTML)**
- **Send the implicit data to client (status codes and response headers)**



5

J2EE training: <http://courses.coreservlets.com>

Class Setup

- **Main development directory**
 - C:\Servlets+JSP
- **Starting Tomcat**
 - Double click startup.bat
 - Enter `http://localhost/` to verify that it is running
 - See popup window for error messages and `System.out.println` output
- **Stopping Tomcat**
 - Double click shutdown.bat
- **Shortcuts to Tomcat directories**
 - In C:\Servlets+JSP
 - Recommend copying onto shortcuts, not going into the directories

6

J2EE training: <http://courses.coreservlets.com>

Using the Default Web App and Invoker Servlet on Tomcat

- **Packageless Servlets**
 - Code: `tomcat_dir/webapps/ROOT/WEB-INF/classes`
 - URL: `http://localhost/servlet/ServletName`
 - See shortcut in C:\Servlets+JSP
- **Packaged Servlets**
 - Code: `tomcat_dir/webapps/ROOT/WEB-INF/classes/subdirectoryMatchingPackageName`
 - URL: `http://localhost/servlet/packageName.ServletName`
- **HTML and JSP Files**
 - Code: `tomcat_dir/ROOT`
 - URL: `http://localhost/filename`
 - See shortcut in C:\Servlets+JSP

7

J2EE training: <http://courses.coreservlets.com>

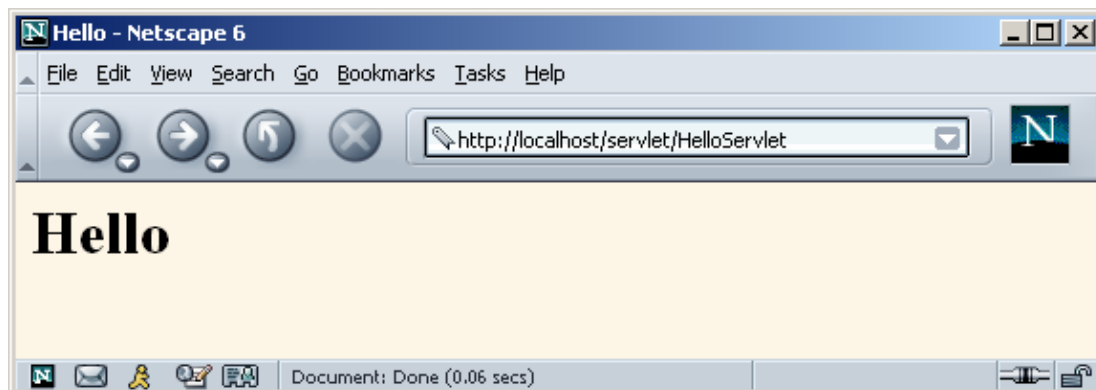
A Packageless Servlet (from CSAJSP)

```
public class HelloServlet extends HttpServlet {
    public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        String docType =
            "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0 \"+
            \"Transitional//EN\">\n";
        out.println(docType +
                    "<HTML>\n" +
                    "<HEAD><TITLE>Hello</TITLE></HEAD>\n"+
                    "<BODY BGCOLOR=\"#FDF5E6\">\n" +
                    "<H1>Hello</H1>\n" +
                    "</BODY></HTML>");
    }
}
```

8

J2EE training: <http://courses.coreservlets.com>

A Packageless Servlet



9

J2EE training: <http://courses.coreservlets.com>

Packaging Servlets

- **Move the files to a subdirectory that matches the intended package name**
 - For example, I'll use the `coreservlets` or `moreservlets` package for most of the rest of the servlets in this course. So, the class files need to go in a subdirectory called `coreservlets`.
- **Insert a package statement in the class file**
 - E.g., top of `HelloServlet2.java`:
`package coreservlets;`
- **Keep CLASSPATH referring to top-level dir**
 - E.g., `C:\Servlets+JSP`. (No changes to CLASSPATH!)
- **Include package name in URL**
 - `http://localhost/servlet/coreservlets>HelloServlet2`

10

J2EE training: <http://courses.coreservlets.com>

Packaging Servlets: HelloServlet2 (Code)

```
package coreservlets;

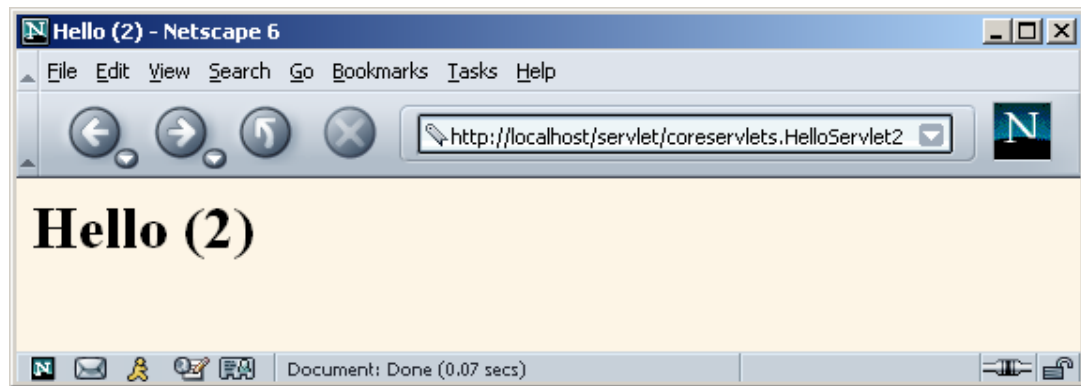
...

public class HelloServlet2 extends HttpServlet {
    public void doGet(HttpServletRequest request,
        HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        String docType =
            "<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 "+
            "Transitional//EN">\n";
        out.println(docType +
            "<HTML>\n" +
            "<HEAD><TITLE>Hello (2)</TITLE></HEAD>\n"+
            "<BODY BGCOLOR=\"#FDF5E6\">\n" +
            "<H1>Hello (2)</H1>\n" +
            "</BODY></HTML>");
    }
}
```

11

J2EE training: <http://courses.coreservlets.com>

Packaging Servlets: HelloServlet2 (Result)



12

J2EE training: <http://courses.coreservlets.com>

Using Form Data

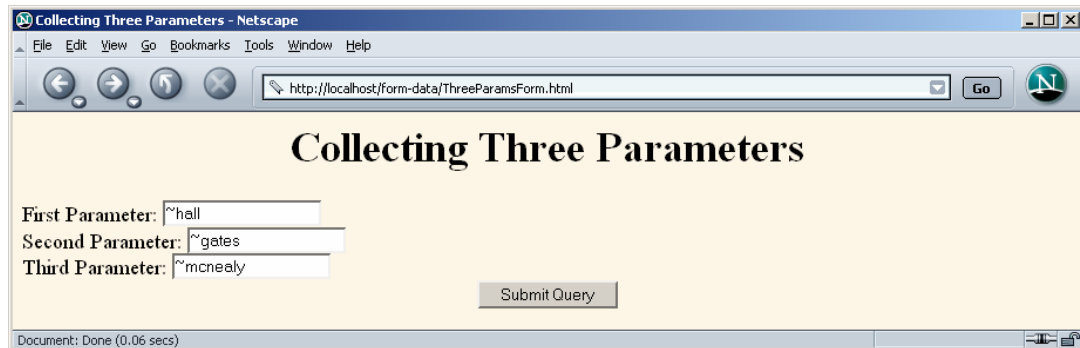
- **HTML form**
 - Should have ACTION referring to servlet
 - Should have input entries with NAMES
 - Should be installed in top-level Web app directory (e.g., ROOT) or any subdirectory other than WEB-INF
- **Servlet**
 - Calls request.getParameter with name as given in HTML
 - Return value is entry as entered by end user
 - Missing values
 - null if no input element of that name was in form
 - Empty string if form submitted with empty textfield

13

J2EE training: <http://courses.coreservlets.com>

An HTML Form With Three Parameters

```
<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>
```



- Form installed in ROOT/form-data/ThreeParamsForm.html

14

J2EE training: <http://courses.coreservlets.com>

Reading the Three Parameters

```
public class ThreeParams extends HttpServlet {  
  public void doGet(HttpServletRequest request,  
                    HttpServletResponse response)  
    throws ServletException, IOException {  
    ...  
    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>");  
  }  
}
```

15

J2EE training: <http://courses.coreservlets.com>

Reading Three Parameters: Result



- Servlet installed in `ROOT/WEB-INF/classes/coreservlets/ThreeParams.class`

16

J2EE training: <http://courses.coreservlets.com>

JSP Scripting: Uses of JSP Constructs

Simple
Application



Complex
Application

- Scripting elements calling servlet code directly
- Scripting elements calling servlet code indirectly (by means of utility classes)
- Beans
- Servlet/JSP combo (MVC)
- MVC with JSP expression language
- Custom tags

17

J2EE training: <http://courses.coreservlets.com>

JSP Scripting Design Strategy: Limit Java Code in JSP Pages

- **You have two options**
 - Put 25 lines of Java code directly in the JSP page
 - Put those 25 lines in a separate Java class and put 1 line in the JSP page that invokes it
- **Why is the second option *much* better?**
 - **Development.** You write the separate class in a Java environment (editor or IDE), not an HTML environment
 - **Debugging.** If you have syntax errors, you see them immediately at compile time. Simple print statements can be seen.
 - **Testing.** You can write a test routine with a loop that does 10,000 tests and reapply it after each change.
 - **Reuse.** You can use the same class from multiple pages.

18

J2EE training: <http://courses.coreservlets.com>

JSP Expressions

- **Format**
 - `<%= Java Expression %>`
- **Result**
 - Expression evaluated, converted to String, and placed into HTML page at the place it occurred in JSP page
 - That is, expression placed in `_jspService` inside `out.print`
- **Examples**
 - Current time: `<%= new java.util.Date() %>`
 - Your hostname: `<%= request.getRemoteHost() %>`
- **XML-compatible syntax**
 - `<jsp:expression>Java Expression</jsp:expression>`
 - You cannot mix versions within a single page. You must use XML for *entire* page if you use `jsp:expression`.

19

J2EE training: <http://courses.coreservlets.com>

Predefined Variables

- **request**
 - The `HttpServletRequest` (1st argument to `service/doGet`)
- **response**
 - The `HttpServletResponse` (2nd arg to `service/doGet`)
- **out**
 - The `Writer` (a buffered version of type `JspWriter`) used to send output to the client
- **session**
 - The `HttpSession` associated with the request (unless disabled with the `session` attribute of the page directive)
- **application**
 - The `ServletContext` (for sharing data) as obtained via `getServletContext()`.

20

J2EE training: <http://courses.coreservlets.com>

JSP Scriptlets

- **Format**
 - `<% Java Code %>`
- **Result**
 - Code is inserted verbatim into servlet's `_jspService`
- **Example**
 - `<%
String queryData = request.getQueryString();
out.println("Attached GET data: " + queryData);
%>`
 - `<% response.setContentType("text/plain"); %>`
- **XML-compatible syntax**
 - `<jsp:scriptlet>Java Code</jsp:scriptlet>`

21

J2EE training: <http://courses.coreservlets.com>

JSP Declarations

- **Format**
 - `<%! Java Code %>`
- **Result**
 - Code is inserted verbatim into servlet's class definition, outside of any existing methods
- **Examples**
 - `<%! private int someField = 5; %>`
 - `<%! private void someMethod(...) { ... } %>`
- **Design consideration**
 - Fields are clearly useful. For methods, it is usually better to define the method in a separate Java class.
- **XML-compatible syntax**
 - `<jsp:declaration>Java Code</jsp:declaration>`

22

J2EE training: <http://courses.coreservlets.com>

Debugging Servlets and JSP

- **Use print statements; run server on desktop**
- **Integrated debugger in IDE**
- **Look at the HTML source**
- **Return error pages to the client**
 - Plan ahead for missing or malformed data
- **Use the log file**
 - `log("message")` or `log("message", Throwable)`
- **Look at the request data separately.**
 - See EchoServer at www.moreservlets.com
- **Look at the response data separately**
 - See WebClient at www.moreservlets.com
- **Stop and restart the server**

23

J2EE training: <http://courses.coreservlets.com>



Questions?

JSP, Servlet, Struts, JSF, AJAX, & Java 5 Training: <http://courses.coreservlets.com>
J2EE Books from Sun Press: <http://www.coreservlets.com>