Servlet and JSP Review

***Core* Servlets & JSP book: www.coreservlets.com**
***More* Servlets & JSP book: www.moreservlets.com**
Servlet and JSP Training Courses: courses.coreservlets.com

For live Struts training, please see JSP/servlet/Struts/JSF training courses at http://courses.coreservlets.com/.

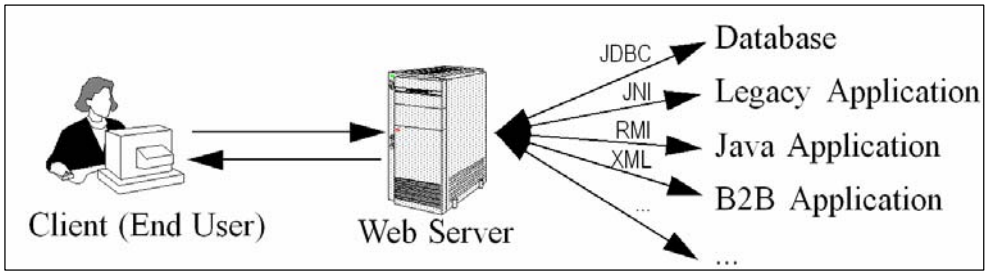
Taught by the author of *Core Servlets and JSP*, *More Servlets and JSP*, and this tutorial. Available at public venues, or customized versions can be held on-site at your organization.

Agenda

- What servlets are all about
- Advantages of servlets
- What JSP is all about
- Free servlet and JSP engines
- Compiling and invoking servlets
- Servlet structure
- A few basic servlets
- Servlet lifecycle
- Initializing servlets
- Debugging servlets

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)



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

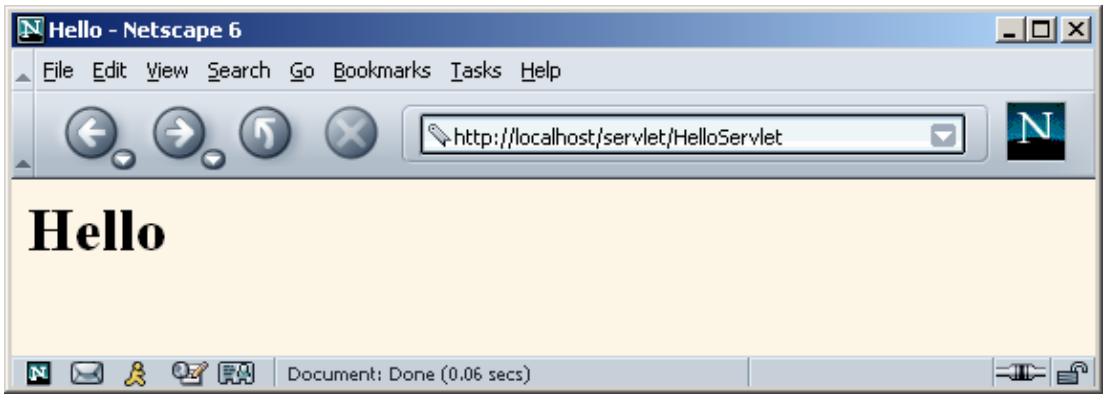
Using the Default Web App and Invoker Servlet on Tomcat

- **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

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>");
  }
}
```


A Packageless Servlet



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

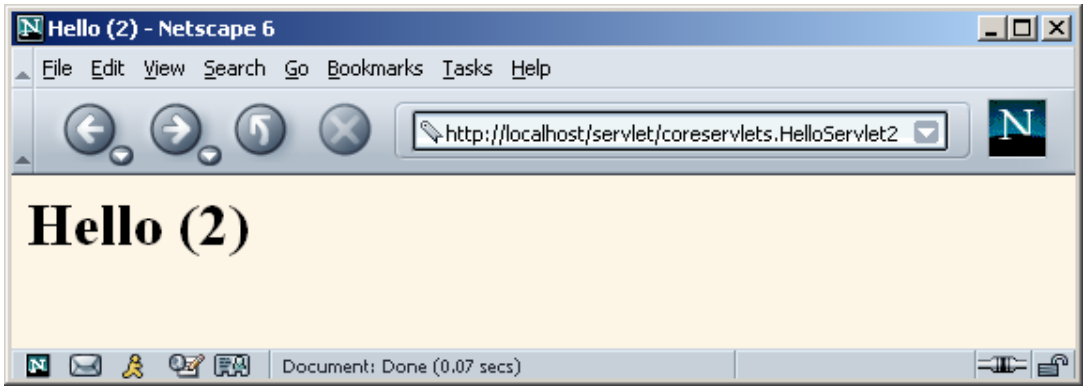
Packaging Servlets: HelloServlet2 (Code)

```
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>");
  }
}
```

Packaging Servlets: HelloServlet2 (Result)

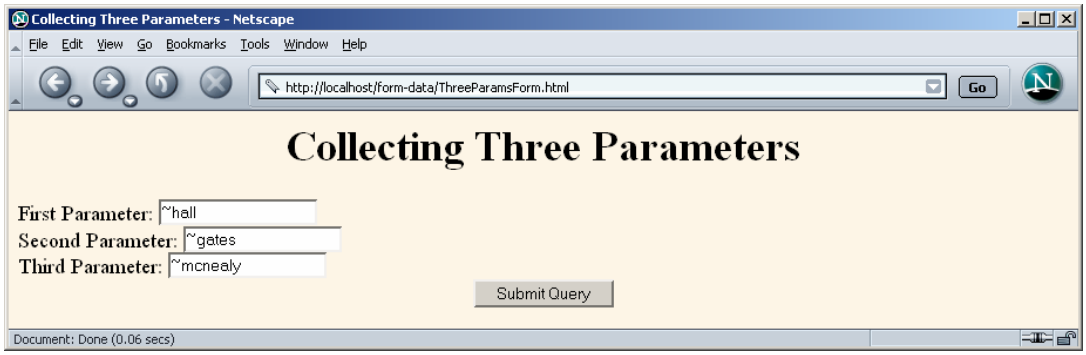


Using Form Data

- **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

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

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>");
  }
}
```

Reading Three Parameters: Result

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

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

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.

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.

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().

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>

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>

Including Files at Request Time: jsp:include

- **Format**
  - <jsp:include page="Relative URL" />
- **Purpose**
  - To reuse JSP, HTML, or plain text content
  - To permit updates to the included content without changing the main JSP page(s)
- **Notes**
  - JSP content cannot affect main page: only *output* of included JSP page is used
  - Don't forget that trailing slash
  - Relative URLs that starts with slashes are interpreted relative to the Web app, not relative to the server root.
  - You are permitted to include files from WEB-INF

jsp:include Example: A News Headline Page (Main Page)

```
…
<BODY>
<TABLE BORDER=5 ALIGN="CENTER">
  <TR><TH CLASS="TITLE">
      What's New at JspNews.com</TABLE>
<P>
Here is a summary of our three
most recent news stories:
<OL>
  <LI><jsp:include page="/WEB-INF/Item1.html" />
  <LI><jsp:include page="/WEB-INF/Item2.html" />
  <LI><jsp:include page="/WEB-INF/Item3.html" />
</OL>
</BODY></HTML>
```


A News Headline Page, Continued (First Included Page)

```
<B>Bill Gates acts humble.</B> In a startling
and unexpected development, Microsoft big wig
Bill Gates put on an open act of humility
yesterday.
<A HREF="http://www.microsoft.com/Never.html">
More details...</A>
```

- Note that the page is *not* a complete HTML document; it has only the tags appropriate to the place that it will be inserted

A News Headline Page: Result

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**

Web Applications: A Summary

- **Learning**
  - Use default Web application (ROOT on Tomcat)
  - Use default URLs (http://…/servlet/ServletName)
  - Advantages
    - Simpler
    - Can test without restarting server or editing web.xml
- **Deployment**
  - Use a custom Web application (on Tomcat, a directory in install_dir/webapps with structure similar to ROOT)
  - Register custom URLs in WEB-INF/web.xml
  - Advantages
    - URLs look better
    - Advanced features (init params, security, filters, etc.) depend on your using registered URLS

Making Custom Web Apps

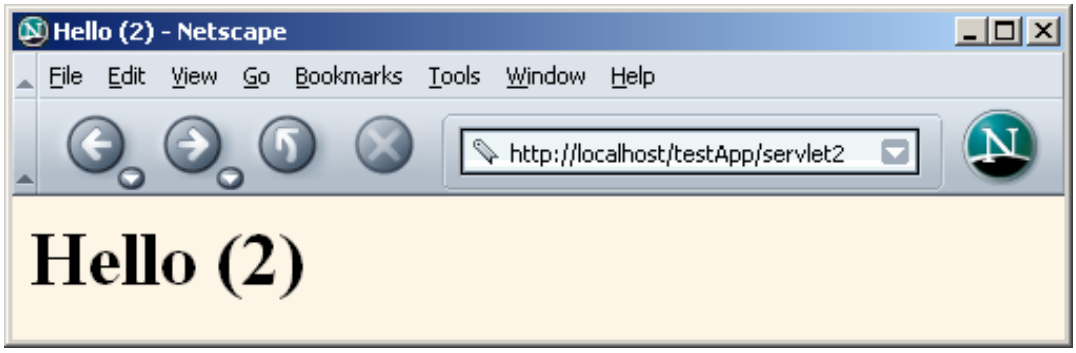
1. **Make a directory whose structure mirrors the structure of the default Web application.**
   - HTML and JSP documents go in the top-level directory
   - The web.xml file goes in the WEB-INF subdirectory
   - Servlets and other classes go either in WEB-INF/classes or a subdirectory of WEB-INF/classes that matches the package name.
   - On Tomcat, entire directory goes in *install_dir*/webapps
2. **Update your CLASSPATH.**
   - Add *webAppDir*/WEB-INF/classes to it.

Making Custom Web Apps

3. **Use the directory name in the URL**
   - All URLs should be of the form http://host/webAppDir/…
4. **Use web.xml to assign custom URLs**
   - Use the `servlet` and `servlet-mapping` elements

```
<servlet>
  <servlet-name>Servlet2</servlet-name>
  <servlet-class>
    coreservlets.HelloServlet2
  </servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>Servlet2</servlet-name>
  <url-pattern>/servlet2</url-pattern>
</servlet-mapping>
```


Making Custom Web Apps



Questions?

Core **Servlets & JSP book: www.coreservlets.com**
More **Servlets & JSP book: www.moreservlets.com**
Servlet/JSP/Struts/JSF Training: courses.coreservlets.com