



Getting Started

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

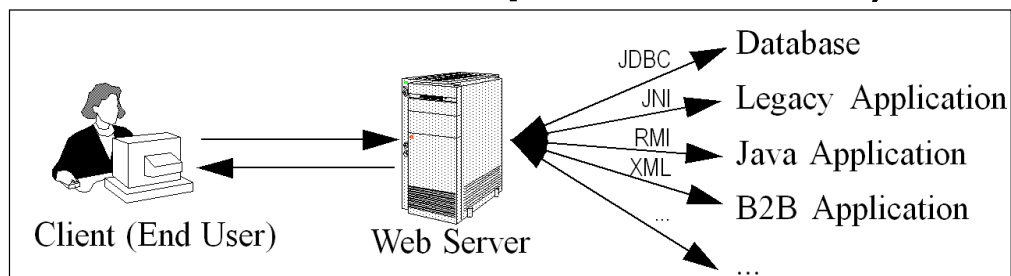
Slides © Marty Hall, <http://www.coreservlets.com>, book © Sun Microsystems Press

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)



3

Getting Started

www.coreservlets.com

Why Build Web Pages Dynamically?

- **The Web page is based on data submitted by the user**
 - E.g., results page from search engines and order-confirmation pages at on-line stores
- **The Web page is derived from data that changes frequently**
 - E.g., a weather report or news headlines page
- **The Web page uses information from databases or other server-side sources**
 - E.g., an e-commerce site could use a servlet to build a Web page that lists the current price and availability of each item that is for sale.

4

Getting Started

www.coreservlets.com

The Advantages of Servlets Over “Traditional” CGI

- **Efficient**
 - Threads instead of OS processes, one servlet copy, persistence
- **Convenient**
 - Lots of high-level utilities
- **Powerful**
 - Sharing data, pooling, persistence
- **Portable**
 - Run on virtually all operating systems and servers
- **Secure**
 - No shell escapes, no buffer overflows
- **Inexpensive**
 - There are plenty of free and low-cost servers.

5

Getting Started

www.coreservlets.com

Extending the Power of Servlets: JavaServer Pages (JSP)

- **Idea:**
 - Use regular HTML for most of page
 - Mark dynamic content with special tags
 - Details in second half of course

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD><TITLE>Welcome to Our Store</TITLE></HEAD>
<BODY>
<H1>Welcome to Our Store</H1>
<SMALL>Welcome,
<!-- User name is "New User" for first-time visitors -->
<%= coreservlets.Utils.getUserNameFromCookie(request) %>
To access your account settings, click
<A HREF="Account-Settings.html">here.</A></SMALL>
<P>
Regular HTML for rest of on-line store's Web page
</BODY></HTML>
```

6

Getting Started

www.coreservlets.com

Server-Side Java is Driving the Web

Get on board or get out of the way



7

Getting Started

www.coreservlets.com

Free Servlet and JSP Engines

- **Apache Tomcat**
 - <http://jakarta.apache.org/tomcat/>
 - See <http://www.moreservlets.com/Using-Tomcat-4.html>
- **Allaire/Macromedia JRun**
 - <http://www.macromedia.com/software/jrun/>
- **New Atlanta ServletExec**
 - <http://www.servletexec.com/>
- **Gefion Software LiteWebServer**
 - <http://www.gefionsoftware.com/LiteWebServer/>
- **Caucho's Resin**
 - <http://www.caucho.com/>

8

Getting Started

www.coreservlets.com



BREAK

Slides © Marty Hall, <http://www.coreservlets.com>, book © Sun Microsystems Press

Compiling and Invoking Servlets

- **Set your CLASSPATH**
 - Servlet JAR file (e.g., *install_dir*/common/lib/servlet.jar).
 - Top of your package hierarchy
- **Put your servlet classes in proper location**
 - Locations vary from server to server. E.g.,
 - *tomcat_install_dir*/webapps/ROOT/WEB-INF/classes
See <http://www.moreservlets.com/Using-Tomcat-4.html>
 - *jrun_install_dir*/servers/default/default-app/WEB-INF/classes
- **Invoke your servlets**
 - `http://host/servlet/ServletName`
 - Custom URL-to-servlet mapping (via web.xml)

Simple Servlet Template

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

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

        // Use "request" to read incoming HTTP headers
        // (e.g. cookies) and HTML form data (query data)

        // Use "response" to specify the HTTP response status
        // code and headers (e.g. the content type, cookies).

        PrintWriter out = response.getWriter();
        // Use "out" to send content to browser
    }
}
```

11

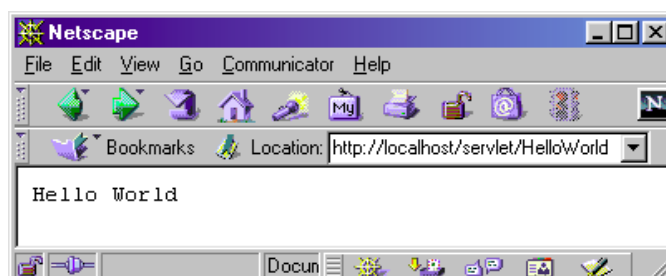
Getting Started

www.coreservlets.com

A Simple Servlet That Generates Plain Text

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

public class HelloWorld extends HttpServlet {
    public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
        throws ServletException, IOException {
        PrintWriter out = response.getWriter();
        out.println("Hello World");
    }
}
```



12

Getting Started

www.coreservlets.com

Compiling and Invoking Servlet (Tomcat 4; Class Setup)

- **Place code in C:\Servlets+JSP.**
 - R-click on source code at <http://archive.coreservlets.com>
- **CLASSPATH already set**
- **Start DOS; type "javac HelloWorld.java"**
- **Place HelloWorld.class in servlet directory**
 - C:\jakarta-tomcat-4.0\webapps\ROOT\WEB-INF\classes
 - Drag files onto shortcut in Servlets+JSP directory
- **Start server**
 - Double click startup.bat
- **Invoke servlet**
 - <http://localhost/servlet/HelloWorld>

Generating HTML

- **Set the Content-Type header**
 - Use `response.setContentType`
- **Output HTML**
 - Be sure to include the DOCTYPE
- **Use an HTML validation service**
 - <http://validator.w3.org/>
 - <http://www.htmlhelp.com/tools/validator/>
 - If your servlets are behind a firewall, you can run them, save the HTML output, and use a file upload form to validate.

A Servlet That Generates HTML

```
public class HelloWWW 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 WWW</TITLE></HEAD>\n" +
                    "<BODY>\n" +
                    "<H1>Hello WWW</H1>\n" +
                    "</BODY></HTML>");
    }
}
```

Packaging Servlets

- **Move the files to a subdirectory that matches the intended package name**
 - For example, I'll use the coreservlets 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 HelloWWW2.java:
`package coreservlets;`
- **Set CLASSPATH to top-level directory**
 - E.g., C:\Servlets+JSP.
- **Include package name in URL**
 - `http://localhost/servlet/coreservlets>HelloWWW2`

Some Simple HTML-Building Utilities

```
public class ServletUtilities {
    public static final String DOCTYPE =
        "<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 " +
        "Transitional//EN\">";

    public static String headWithTitle(String title) {
        return(DOCTYPE + "\n" +
            "<HTML>\n" +
            "<HEAD><TITLE>" + title + "</TITLE></HEAD>\n");
    }
    ...
}
```

- **Don't go overboard**

- Complete HTML generation packages usually work poorly
- The JSP framework is a better solution

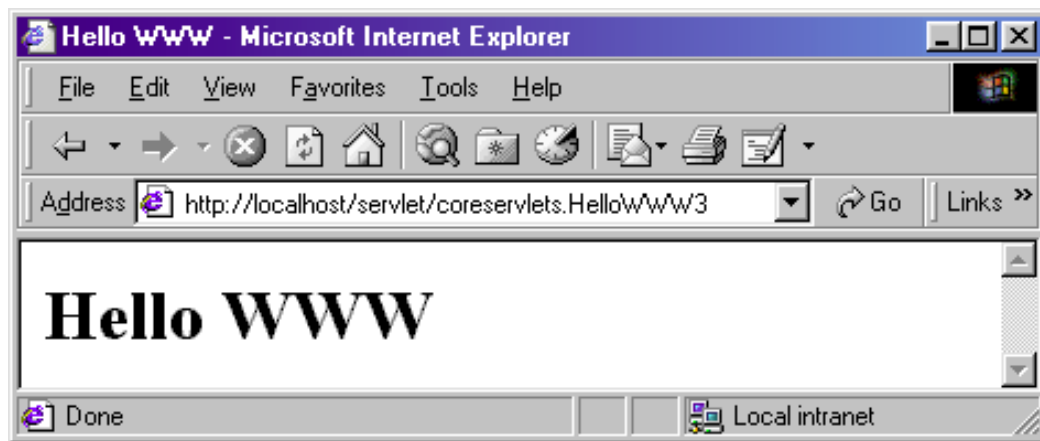
HelloWWW with ServletUtilities

```
package coreservlets;

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

public class HelloWWW3 extends HttpServlet {
    public void doGet(HttpServletRequest request,
        HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println(ServletUtilities.headWithTitle("Hello WWW") +
            "<BODY>\n" +
            "<H1>Hello WWW</H1>\n" +
            "</BODY></HTML>");
    }
}
```

HelloWWW Result



The Servlet Life Cycle

- **init**
 - Executed once when the servlet is first loaded.
Not called for each request.
- **service**
 - Called in a new thread by server for each request.
Dispatches to doGet, doPost, etc.
Do not override this method!
- **doGet, doPost, doXxx**
 - Handles GET, POST, etc. requests.
 - Override these to provide desired behavior.
- **destroy**
 - Called when server deletes servlet instance.
Not called after each request.

Why You Should Not Override service

- You can add support for other services later by adding doPut, doTrace, etc.
- You can add support for modification dates by adding a getLastModified method
- The service method gives you automatic support for:
 - HEAD requests
 - OPTIONS requests
 - TRACE requests
- **Alternative: have doPost call doGet**

Initializing Servlets

- **Common in real-life servlets**
 - E.g., initializing database connection pools.
- **Use ServletConfig.getInitParameter to read initialization parameters**
- **Set init parameters in web.xml (ver 2.2/2.3)**
 - .../WEB-INF/web.xml
 - Many servers have custom interfaces to create web.xml
- **It is common to use init even when you don't read init parameters**
 - See modification date example in *Core Servlets and JavaServer Pages* Chapter 2

Debugging Servlets

- **Use print statements; run server on desktop**
- **Use Apache Log4J**
- **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)`
- **Separate the request and response data .**
 - Request: see EchoServer at www.coreservlets.com
 - Response: see WebClient at www.coreservlets.com
- **Stop and restart the server**

23

Getting Started

www.coreservlets.com

Web Applications: A Preview

- **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 custom URLs

24

Getting Started

www.coreservlets.com

Making Custom Web Apps

1. Make a directory whose structure mirrors the structure of the default Web application.

- HTML (and, eventually, 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

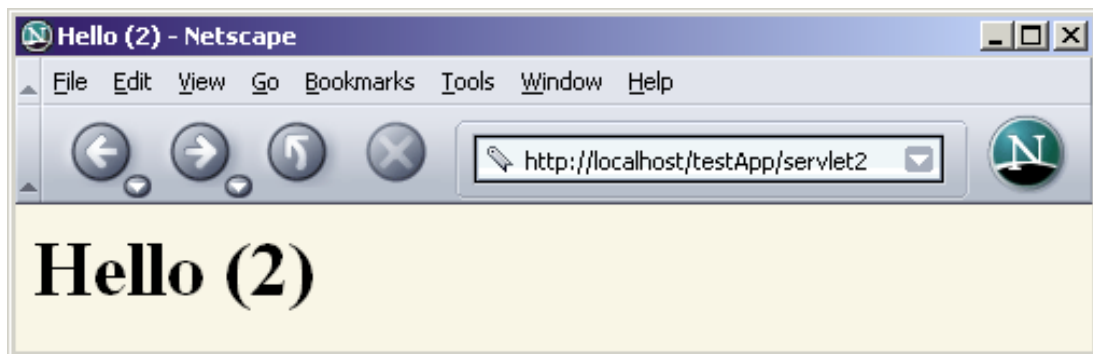
- All URLs should be of the form
http://host/webAppDir/...

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



Summary

- **Servlets are efficient, portable, powerful, and widely accepted in industry**
- **Regardless of deployment server, run a free server on your desktop for development**
- **Getting started:**
 - Set your CLASSPATH
 - Servlet JAR file
 - Top of your package hierarchy
 - Put class files in proper location
 - .../WEB-INF/classes
 - Use proper URL, usually `http://host/servlet/ServletName`
- **Download existing servlet first time**
 - Start with HelloWWW from www.coreservlets.com

Summary (Continued)

- **Main servlet code goes in doGet or doPost:**
 - The HttpServletRequest contains the incoming information
 - The HttpServletResponse lets you set outgoing information
 - Call setContentType to specify MIME type
 - Call getWriter to obtain a Writer pointing to client
- **One-time setup code goes in init**
 - Servlet gets initialized and loaded once
 - Servlet gets invoked multiple times
 - Initialization parameters set in web.xml (covered in detail in *More Servlets & JavaServer Pages* Chapter 5)



Questions?

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