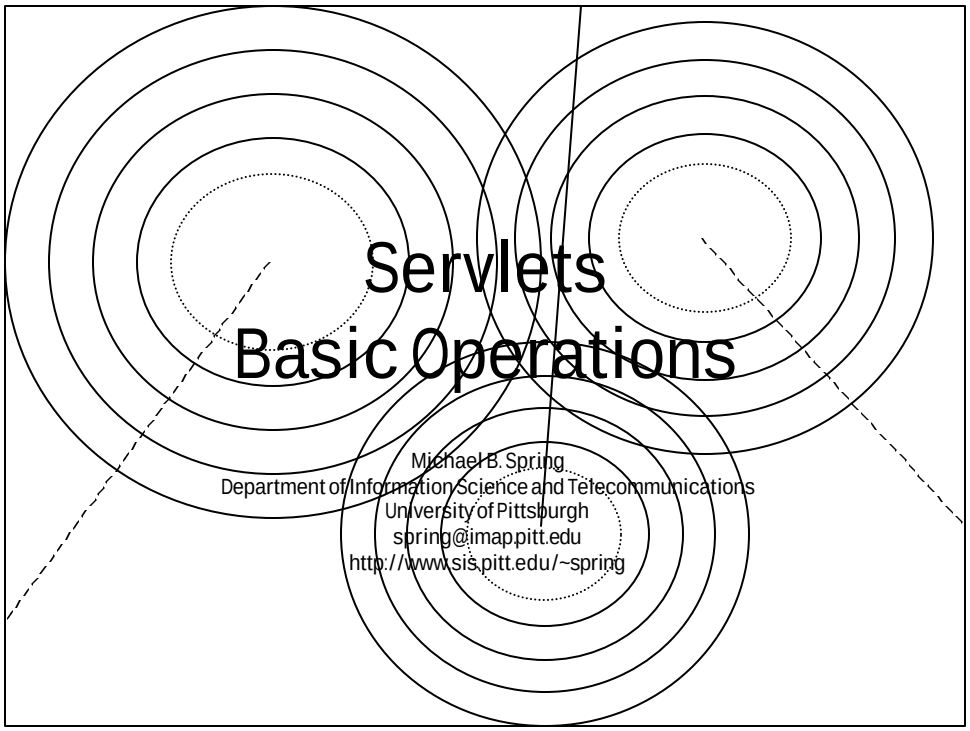


Servlets
Basic Operations

Michael B. Spring
Department of Information Science and Telecommunications
University of Pittsburgh
spring@imap.pitt.edu
http://www.sis.pitt.edu/~spring

Overview

- Preparing to develop servlets on your PC
- Writing and running an “Hello World” servlet
- Servlet Basics
- The Servlet API
- Loading and Testing Servlets

Preparing your System

- Locate the file "jswdk1_0_1-win.zip"
- Using winzip, extract all the files to the JDK directory. By default the directory will be "jdk.1.2.2"
- Set an environment variable called JAVA_HOME to the JDK directory for JDK. Use a fully qualified (complete) pathname in setting the variable
  - In NT, right click on my computer, select the environment tab, and all a new environment variable and value
  - In Win 95/98, edit the autoexec.bat file to include the command "set JAVA_HOME path"
- Add the full path (including filename) of the servlet.jar file in the jswdk-1.0.1/lib subdirectory to your CLASSPATH variable definition

Preparing your System(continued)

- In the subdirectory jswdk-1.0.1, you will find two batch files:
  - startserver.bat is used to start a local servlet server
  - stopserver.bat is used to stop the local servlet server
- If double clicking on the startserver.bat file doesn't work, run a DOS window and issue the command manually.
- Using Netscape or Internet Explorer, specify the following location – http://localhost:8080/
- If you have followed the directions, you will get a page with a set of sample servlets and java server pages.

A Simple Servlet

A Simple Servlet

- Servlets are basically Java programs that do input and output in response to contact from web servers.
- Servlets have several advantages over CGI scripts which will be addressed in time.
- For now we look at the structure of a simple servlet.
  - Import the correct packages
  - Extend the HttpServlet class from the package, overwriting four methods
    - "init" to set resources
    - "service" to provide the basic service
    - "getServletInfo" to provide a description of the servlet to the server – the description is a string
    - "destroy" to free any allocated resources

The Code for the Servlet

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.PrintWriter;
import java.io.IOException;
// a simple servlet
public class simpservlet extends HttpServlet
{
   public void init(ServletConfig config)throws ServletException
   {super.init(config);//just accept the parent init}

   public void service(HttpServletRequest req, HttpServletResponse
   rsp)
        throws ServletException, IOException
   {String ms[]={"one","two","three","four","five"};
   rsp.setContentType("text/html");
   PrintWriter out = rsp.getWriter();
   out.println("<HTML><HEAD><TITLE>Simple</TITLE></HEAD><BODY>");
   for (int i=1;i<=5;i++)
        {out.println("<P>"+i+". This is line " +ms[i-1]);}
   out.close();}

   public String getServletInfo(){return "A little demo";}

   public void destroy(){ //free resources}
}
```

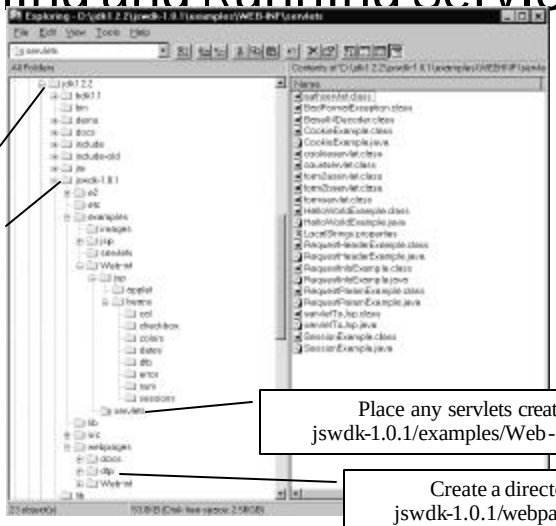

How the Servlet works

- The HTTPServlet class is constructed to minimize the amount of effort you need to put into a client server exchange
- Note that the service method provides both a request and a response.
  - The request consists of a header and a potentially a body neither of which we will deal with here
  - The response will also consist of a header and a body. The superclass will take care of most of that for us, but we will need to identify the ContentType of the response, and there is a method to allow us to do that.
  - Finally, we can obtain access to an IOStream that will allow us to specify the body of the message. When done we simply close the stream

Positioning/Running a Servlet

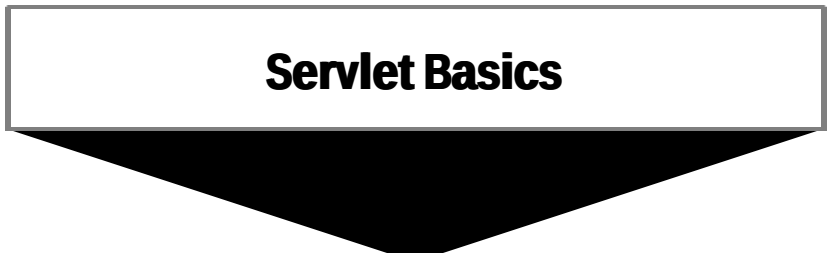
- **For the purposes of this demo, we will not do any further adjustments of the CLASSPATH, but adjustments will be required for more complex servlets and for JSPs**
- **Compile the Java servlet provided as exercise 1.**
- **Copy the class file to the following subdirectory of the JDK directory:**
  - **jswdk-1.0.1\examples\WEB-INF\servlets**
- **Start up the servlet test engine using the batch file in jswdk-1.0.1**
- **Run Netscape or Internet Explorer using the following URL:**
  - **http://localhost:8080/examples/servlet/simpservlet**

Positioning and Running Servlets



An Exercise

- Produce a new servlet based on the example provided above.
- The ideal servlet for this first example will do three things:
  - Expand the string array to specify sayings, tips, or fortunes and select one based on a random number function for display.
  - Present a result document that includes a form with at least a listbox(<SELECT> and <OPTION> and a submit button.
    - The listbox will include as options some number of items that a person might want to buy.
    - You can submit the form to a dummy URL.
    - The form should be a "get" method so you can see the arguments on the request



Rationale for Servlets

- Can be run as threads in the server process requiring less startup overhead.
  - They can be started at server start saving more time
- As bytecodes, servlets generally run faster than scripts
- While slower than binaries, they are more robust, operate cross platform, and cross server
- While servlets support http by explicit subclass, they are protocol independent by design
- They extend the capability and range of Java code bases and expertise.

Servlet Basics

- The servlet lifecyle (init, service, destroy) is as follows:
  - Load (at server startup or at first request) and instantiate one or more copies
    - Creates a ServletConfig object
    - Pass the ServletConfig object to each init process
  - The server processes a request(s)
    - Forming request and response objects based on the connection
    - Calling the service method passing these objects (The service method will be expanded to specific services)
    - The service examines the request, either directly or indirectly(based on the specific service called) and formulates a response passed through the response object
  - The server unloads the servlet when directed to do so and calls the detroy method

The Load/Service Relation

- This servlet will count the number of hits

```
public class countservlet extends HttpServlet
{static int NReq=0;
public void service(HttpServletRequest req,
      HttpServletResponse rsp) throws
      ServletException,IOException
{rsp.setContentType("text/html");
PrintWriter out = rsp.getWriter();
NReq++;
out.println("<HTML><HEAD><TITLE>”)
out.println(“Counted Request</TITLE></HEAD><BODY>");
out.println("<P>Reuest" + NReq + "since start</P>");
out.println("</BODY></HTML>");
out.close();}
```


The Servlet API

The Servlet API

- The servlets are derived as follows:

  java.lang.Object

  javax.servlet.GenericServlet

  javax.servlet.http.HttpServlet

- This discussion focuses primarily on the http servlet, but it should be made clear that there is also a protocol neutral servlet
- The HttpServlet class is supported by an HttpServletRequest interface and the HttpServletResponse interface. (the generic class has similar interfaces)
- The HttpServlet class is an abstract class – it must be extended

Methods required for the HttpServlet

- The methods init(), service(), and destroy() have been discussed. The service method normally passes the request to a specific methods – each of which responds to one of the htttp protocols
- In the examples to date, the overridden service method has provided a monolithic response – which might be very confusing to the client
- Some additional methods exist for communication between the servlet and server, one example being getServletInfo() which has been previously discussed

Specific HTTP Methods

- doGet() responds to normal http requests(GET) and forms actions submitted with the GET method
- doHead() responds to the http HEAD request which asks for header information about a file – type, size, last modification date, etc.
- doPost() responds to the POST request which includes form information in the body of the request
- doPut() responds to the PUT request which includes a document to be stored on the server in the body and placement and authorization information in the header.
- doDelete() responds to the DELETE request which removes a document given appropriate authorization.
- doTrace() responds to the TRACE request which is used for diagnostic purposes – it sends the entire request back as the body of the response.
- doOptions() responds to the OPTIONS request which is used to ask a server what kinds of methods the server will responds to.

Http Headers

- The htttp message header has a number of lines
- The first line takes the form
  - GET (or PUT or POST) path/filename HTTP/1.1
- The other headers are related among other things to
  - Security and authentication
  - Content
  - Client Capability
- The content headers include:
  - Content-Type: The mime type of the body
  - Content-Encoding: Additional coding beyond that assumed for the Type
  - Content-Length: the size of the body, whether or not it is returned

HttpServletRequest Interface

- The HttpServletRequest Interface has several methods. The more interesting of these methods include:
  - getHeaderNames() returns an Enumeration of the message headers
  - getHeader(String name) returns the value of the named header
  - getintHeader(String name) returns the value of the named header as an integer – or –1 if it does not exist
  - getRemoteUser() returns the username submitted by http authentication (when a server requests authentication, the client, if enabled, responds with an "Authorization:" header in the next and subsequent requests that provides the data necessary to authenticate the user.
  - getQueryString() returns the raw string of parameters and values passed from the server to the servlet

ServletRequest Interface

- The HttpServletRequest Interface inherits all of the methods of the ServletRequest Interface
- Several of the ServletRequest methods are of import
  - getReader() returns a BufferedReader we can use to read the body
  - getRemoteHost() returns the client's fully qualified name as a string
  - getParameter(String name) returns the value of the specified parameter. Beware that this is dangerous if the parameter allows multiple values as might be the case in a select.
  - getParameterNames() which returns an Enumeration of parameter names
  - getParameterValues() which returns an array of strings
  - Parameters, are the values from the GET or POST request already parsed.

HttpServletResponse Interface

- The HttpServletResponse Interface has a number of methods. Among the more centeral and interesting are:
  - setHeader(String name, String value) which sets the header of the name given with the value specified. Care should be taken as the validate the correct form of the header or value
  - sendError(int statuscode) sends an error message (in contrast to the standard Ok (200) code. Defined values include:
    - SC_BAD_REQUEST – bad syntax in request
    - SC_CONFLICT – conflict in accessing resource
    - SC_FORBIDDEN – refusal to provide access
    - SC_GONE – resource is no longer here

ServletResponse Interface

- The HttpServletResponse Interface inherits all of the methods of the ServletResponse Interface
- There are only four ServletResponse methods, but they are all important
  - getWriter() returns a PrintWriter which support a println() method that can be used to print text output
  - setContentLength(int length) is critical for setting the ContentLength: header of the response, without which the client will ignore the body of the return message
  - setContentType(String name) is critical in defining for the client the mime type of the body of the message
  - getOutputStream() returns a ServletOutputStream which is used primarily to support the writing of binary data – such as an image.

Loading and Running Servlets

- The first step in loading a servlet is to write and compile the servlet
- Be sure that your CLASSPATH variable includes the fully qualified filename (drive:\path\servlet.jar) for the servlet.jar file
- Move the compiled classfile to the directory where servlets are stored by default
  drive:\JDKLOCATION\jswdk-1.0.1\examples\WEB-INF\servlets\
- Start the JSDK servlet runner using the startserver.bat file in jswdk-1.0.1
- Start your browser and specify the name of the servlet using the following:
  http://localhost:8080/examples/servlet/classfilename
- If you have started the JSDK Servlet Runner on a machine accessible to the internet you may also access the servlet from other machines by specifying the domain name of ip address of the machine running the Servlet Runner.

More on Servlet Runner

- Location of normal HTML files
  drive:\JDKLOCATION\jswdk-1.0.1\webpages\
  Directory depth may be develop as you see fit
- Changing parameters
  - Accessing drive:\JDKLOCATION\jswdk-1.0.1\webserver.xml
- Reloading servlets
  - Need to restart the server
- Using the console for output
  - Use of System.out.println()

Debugging a Servlet

- Many IDEs provide debugging capabilities – including debugging for servlets
- FreeJava offers no direct support for debugging
- You can achieve crude debugging by
  - Including System.out.println() in your source
  - Watching the results in the Servlet Runner Console

Exercise

- Build two servlets.
- Servlet one will be a revision of the servlet you produced in the previous exercise. The Servlet will produce a form that has five text inputs, two sets of radio buttons and two select boxes.
  - One select will be ten linen products(face cloth, bath towel, sheet, etc) and the other color
  - Three text inputs will be: Name, Street Address, and City,
  - One Radio button inputs will be for credit card type
  - Two text inputs will be Credit Card Number and Expiration date
  - The Second Radio Button input will be delivery mode
- The second servlet will process the form return. It will use the doPost or doGet in accord with how you write the form produced by the first resend the form partially filled out if there is missing data, or it will display page confirming the users selection and thanking them if all