CSBP 461
Internet Computing:

Java Servlets (Part 2)

Dr. M. Elarbi Badidi

Topics

- Servlet in big picture of J2EE
- Servlet request & response model
- Servlet life cycle
- Servlet scope objects
- Servlet request
- Servlet response: Status, Header, Body

SCOPE OBJECTS

Scope Objects

- Enables sharing information among collaborating web components via attributes maintained in Scope objects
  - Attributes are name/object pairs
- Attributes maintained in the Scope objects are accessed with
  - getAttribute() & setAttribute()
- 4 Scope objects are defined
  - Web context, session, request, page

Four Scope Objects: Accessibility

- Web context (ServletContext)
  - Accessible from Web components within a Web context
- Session
  - Accessible from Web components handling a request that belongs to the session
- Request
  - Accessible from Web components handling the request
- Page
  - Accessible from JSP page that creates the object

Four Scope Objects: Class

- Web context
  - javax.servlet.ServletContext
- Session
  - javax.servlet.http.HttpSession
- Request
  - subtype of javax.servlet.ServletRequest: javax.servlet.http.HttpServletRequest
- Page
  - javax.servlet.jsp.PageContext

WEB CONTEXT (SERVLETCONTEXT)

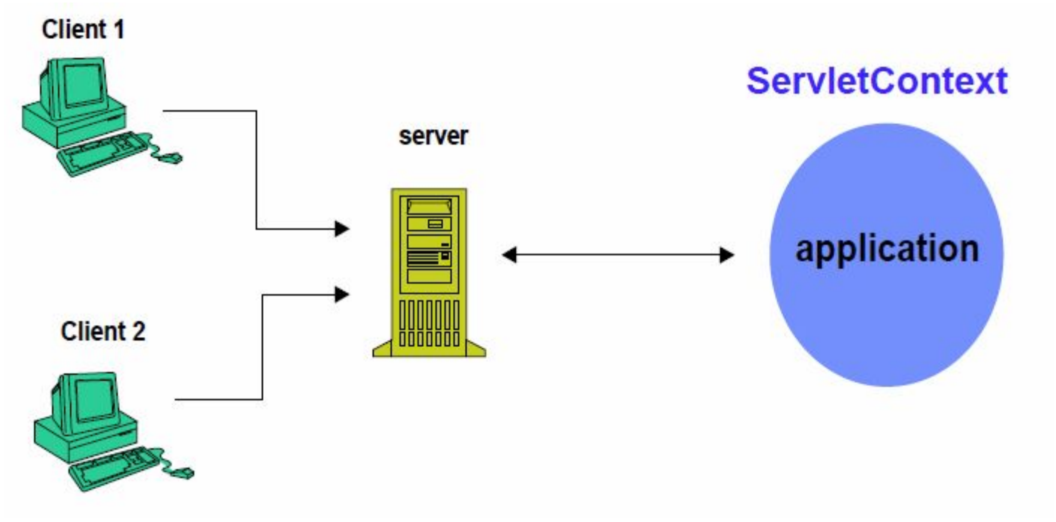
What is ServletContext For?

- Used by servlets to
  - Set and get context-wide (application-wide) object-valued attributes
  - Get request dispatcher
    - To forward to or include web component
  - Access Web context-wide initialization parameters set in the web.xml file
  - Access Web resources associated with the Web context
  - Log
  - Access other misc. information

Scope of ServletContext

- Context-wide scope
  - Shared by all servlets and JSP pages within a "web application“
    - Why it is called “web application scope”
  - A "web application" is a collection of servlets and content installed under a specific subset of the server's URL namespace and possibly installed via a *.war file
    - All servlets in BookStore web application share same ServletContext object
  - There is one ServletContext object per "web application" per Java Virtual Machine

ServletContext: Web Application Scope



How to Access ServletContext Object?

- Within your servlet code, call getServletContext()
- Within your servlet filter code, call getServletContext()
- The ServletContext is contained in ServletConfig object, which the Web server provides to a servlet when the servlet is initialized
    - init (ServletConfig servletConfig) in Servlet interface

Example: Getting Attribute Value from ServletContext

```
public class CatalogServlet extends HttpServlet {
    private BookDB bookDB;
    public void init() throws ServletException {
        // Get context-wide attribute value from
        // ServletContext object
        bookDB = (BookDB)getServletContext().
                getAttribute("bookDB");
        if (bookDB == null) throw new
            UnavailableException("Couldn't get
        database.");
    }
}
```

Example: Getting and Using RequestDispatcher Object

```
public void doGet (HttpServletRequest request,
                  HttpServletResponse response)
      throws ServletException, IOException {

      HttpSession session = request.getSession(true);
      ResourceBundle messages =
   (ResourceBundle)session.getAttribute("messages");

      // set headers and buffer size before accessing the Writer
      response.setContentType("text/html");
      response.setBufferSize(8192);
      PrintWriter out = response.getWriter();

      // then write the response
      out.println("<html>" +
                  "<head><title>" +
  messages.getString("TitleBookDescription") +
                  "</title></head>");

      // Get the dispatcher; it gets the banner to the user
      RequestDispatcher dispatcher =
                          session.getServletContext().getRequestDispatcher("/banner"
  );

      if (dispatcher != null)
            dispatcher.include(request, response);
      ...
```

Example: Logging

```
public void doGet (HttpServletRequest request,
                   HttpServletResponse response)
    throws ServletException, IOException {
    ...
    getServletContext().log(“Life is good!”);
    ...
    getServletContext().log(“Life is bad!”,
  someException);
```

SESSION (HTTPSESSION)

Why HttpSession?

- Need a mechanism to maintain client state across a series of requests from a same user (or originating from the same browser) over some period of time
    - Example: Online shopping cart
- Yet, HTTP is stateless
- HttpSession maintains client state
    - Used by Servlets to set and get the values of session scope attributes

How to Get HttpSession?

- via getSession() method of a Request object (HttpServletRequest)

Example: HttpSession

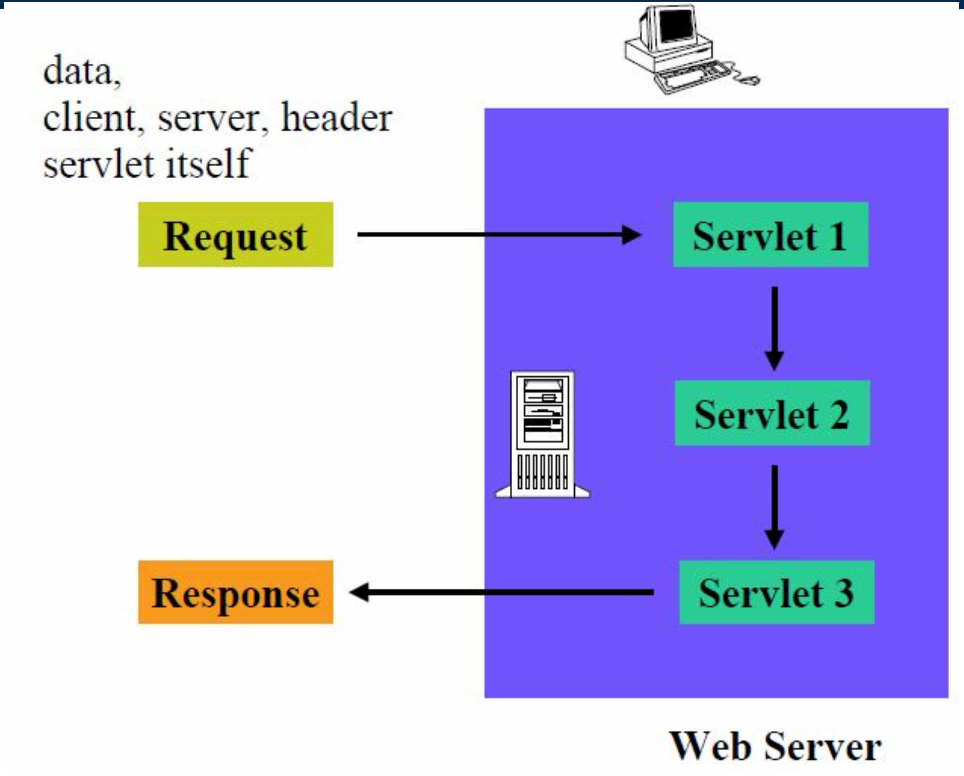
```
public class CashierServlet extends HttpServlet {
 public void doGet (HttpServletRequest request,
                    HttpServletResponse response)
       throws ServletException, IOException {

    // Get the user's session and shopping cart
    HttpSession session = request.getSession();
    ShoppingCart cart =
    (ShoppingCart)session.getAttribute("cart");
    ...
    // Determine the total price of the user's books
    double total = cart.getTotal();
```

SERVLET REQUEST (**HTTPSERVLETREQUEST**)

What is Servlet Request?

- Contains data passed from client to servlet
- All servlet requests implement ServletRequest interface which defines methods for accessing
  - Client sent parameters
  - Object-valued attributes
  - Locales
  - Client and server
  - Input stream
  - Protocol information
  - Content type
  - If request is made over secure channel (HTTPS)

Requests


data,
client, server, header
servlet itself
Request
Servlet 1
Servlet 2
Servlet 3
Response
Web Server

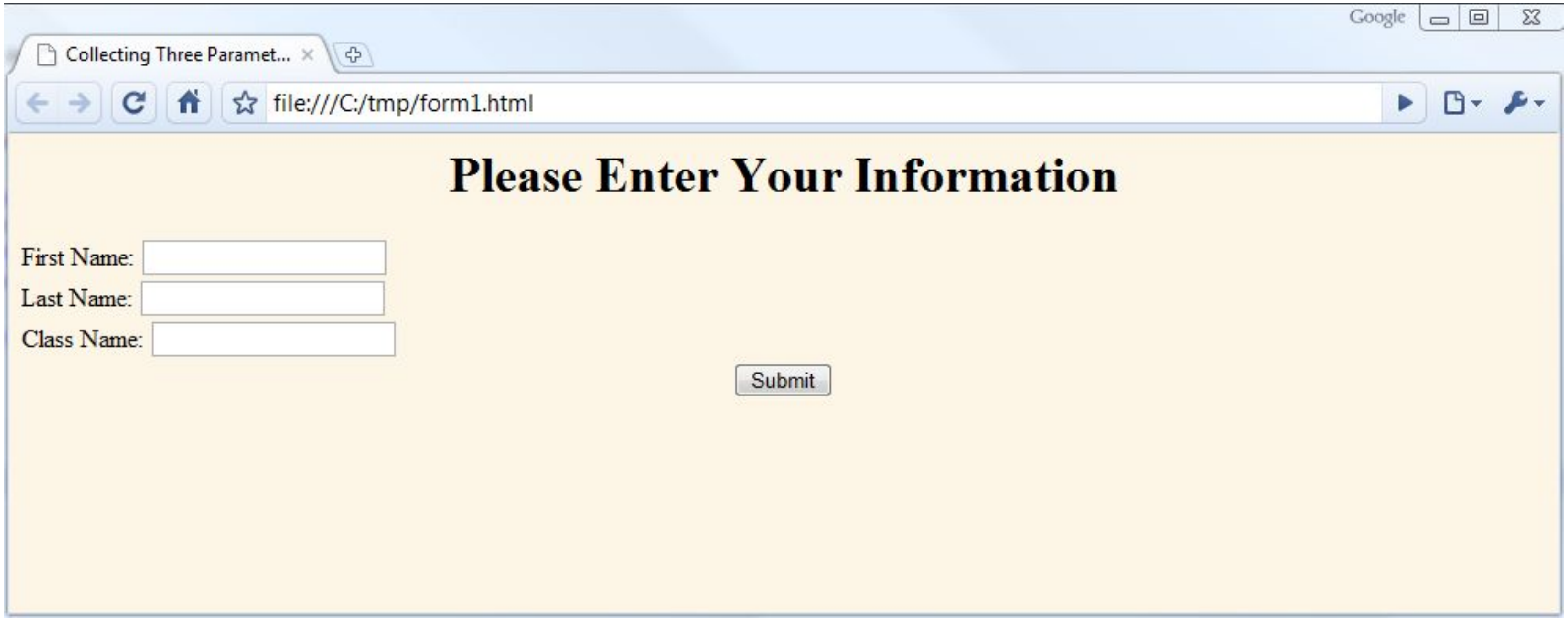
Getting Client Sent Parameters

- A request can come with any number of parameters
- Parameters are sent from HTML forms:
  - GET: as a query string, appended to a URL
  - POST: as encoded POST data, not appeared in the URL
- getParameter("paraName")
  - Returns the value of paraName
  - Returns null if no such parameter is present
  - Works identically for GET and POST requests

A Sample FORM using GET

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

  <FORM ACTION="/sample/servlet/ThreeParams">
     First Name: <INPUT TYPE="TEXT" NAME="param1"><BR>
     Last Name: <INPUT TYPE="TEXT" NAME="param2"><BR>
     Class Name: <INPUT TYPE="TEXT" NAME="param3"><BR>
     <CENTER>
          <INPUT TYPE="SUBMIT">
     </CENTER>
  </FORM>
  </BODY>
</HTML>
```

A Sample FORM using GET



Collecting Three Paramet...

file:///C:/tmp/form1.html

Please Enter Your Information

First Name:

Last Name:

Class Name:

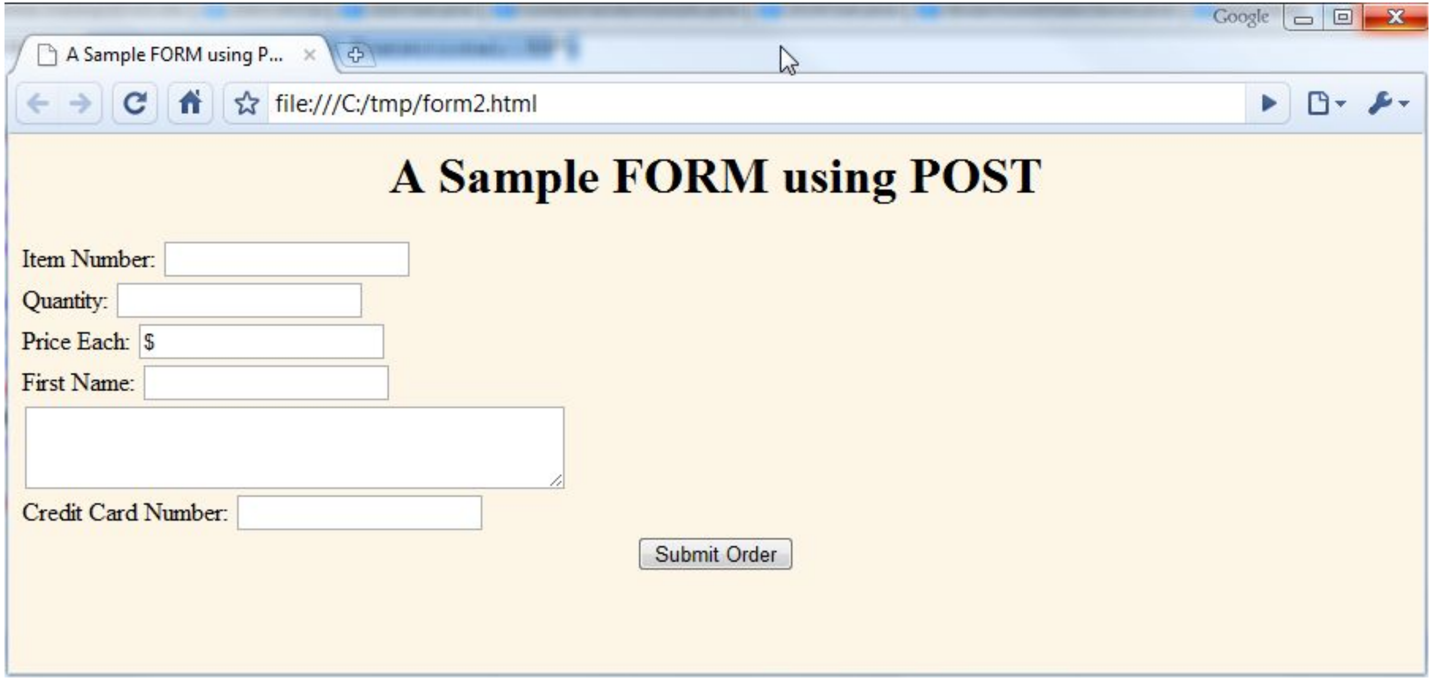
A FORM Based Servlet: Get

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

/** Simple servlet that reads three parameters from the html form */
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 = "Your Information";
        out.println("<HTML>" +
                 "<BODY BGCOLOR=\"#FDF5E6\">\n" +
                 "<H1 ALIGN=CENTER>" + title + "</H1>\n" +
                 "<UL>\n" +
                 "  <LI><B>First Name in Response</B>: "
                 + request.getParameter("param1") + "\n" +
                 "  <LI><B>Last Name in Response</B>: "
                 + request.getParameter("param2") + "\n" +
                 "  <LI><B>NickName in Response</B>: "
                 + request.getParameter("param3") + "\n" +
                 "</UL>\n" +
                 "</BODY></HTML>");
  }
}
```

A Sample FORM using POST

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
  <TITLE>A Sample FORM using POST</TITLE>
</HEAD>
<BODY BGCOLOR="#FDF5E6">
<H1 ALIGN="CENTER">A Sample FORM using POST</H1>
<FORM ACTION="/sample/servlet/ShowParameters" METHOD="POST">
  Item Number: <INPUT TYPE="TEXT" NAME="itemNum"><BR>
  Quantity: <INPUT TYPE="TEXT" NAME="quantity"><BR>
  Price Each: <INPUT TYPE="TEXT" NAME="price" VALUE="$"><BR>
  First Name: <INPUT TYPE="TEXT" NAME="firstName"><BR>
  <TEXTAREA NAME="address" ROWS=3 COLS=40></TEXTAREA><BR>
  Credit Card Number:
  <INPUT TYPE="PASSWORD" NAME="cardNum"><BR>
  <CENTER>
     <INPUT TYPE="SUBMIT" VALUE="Submit Order">
  </CENTER>
</FORM>
</BODY>
</HTML>
```

A Sample FORM using POST



A Sample FORM using POST

Item Number:

Quantity:

Price Each: $

First Name:

Credit Card Number:

Submit Order

A Form Based Servlet: POST

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

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

  public void doPost(HttpServletRequest request,
               HttpServletResponse response)
        throws ServletException, IOException {
     doGet(request, response);
  }
}
```

Who Set Object/value Attributes

- Request attributes can be set in two ways
  - Servlet container itself might set attributes to make available custom information about a request
    - example: javax.servlet.request.X509Certificate attribute for HTTPS
  - Servlet set application-specific attribute
    - void setAttribute(java.lang.String name, java.lang.Object o)
    - Embedded into a request before a RequestDispatcher call

Getting Locale Information

```
public void doGet (HttpServletRequest request,
              HttpServletResponse response)
        throws ServletException, IOException {

    HttpSession session =request.getSession();
    ResourceBundle messages =

    (ResourceBundle)session.getAttribute("messages");

    if (messages == null) {
        Locale locale=request.getLocale();
        messages = ResourceBundle.getBundle(
        "messages.BookstoreMessages", locale);
        session.setAttribute("messages", messages);
    }
```

Getting Client Information

- Servlet can get client information from the request
  - String request.getRemoteAddr()
    - Get client's IP address
  - String request.getRemoteHost()
    - Get client's host name

Getting Server Information

- Servlet can get server's information:
  - String request.getServerName()
    - e.g. www.sun.com
  - int request.getServerPort()
    - e.g. Port number "8080"

Getting Misc. Information

- Input stream
  - ServletInputStream getInputStream()
  - java.io.BufferedReader getReader()
- Protocol
  - java.lang.String getProtocol()
- Content type
  - java.lang.String getContentType()
- Is secure or not (if it is HTTPS or not)
  - boolean isSecure()

HTTP SERVLET REQUEST

What is HTTP Servlet Request?

- Contains data passed from HTTP client to HTTP servlet
- Created by servlet container and passed to servlet as a parameter of doGet() or doPost() methods
- HttpServletRequest is an extension of ServletRequest and provides additional methods for accessing
  - HTTP request URL
    - Context, servlet, path, query information
  - Misc. HTTP Request header information
  - Authentication type & User security information
  - Cookies
  - Session

HTTP Request URL

- Contains the following parts
    - http://[host]:[port]/[request path]?[query string]

HTTP Request URL: [request path]

- http://[host]:[port]/[request path]?[query string]
- [request path] is made of
  - Context: /<context of web app>
  - Servlet name: /<component alias>
  - Path information: the rest of it
- Examples
  - http://localhost:8080/hello1/greeting
  - http://localhost:8080/hello1/greeting.jsp
  - http://daydreamer/catalog/lawn/index.html

HTTP Request URL: [query string]

- http://[host]:[port]/[request path]?[query string]
- [query string] are composed of a set of parameters and values that are user entered
- Two ways query strings are generated
  - A query string can explicitly appear in a web page
    - `<a href="/bookstore1/catalog?Add=101">Add To Cart</a>`
    - `String bookId = request.getParameter("Add");`
  - A query string is appended to a URL when a form with a GET HTTP method is submitted
    - http://localhost/hello1/greeting?username=Ahmed
    - `String userName=request.getParameter(“username”)`

Context, Path, Query, Parameter Methods

- String getContextPath()
- String getQueryString()
- String getPathInfo()
- String getPathTranslated()

HTTP Request Headers

- HTTP requests include headers which provide extra information about the request
- Example of HTTP 1.1 Request:

```
GET /search? keywords= servlets+ jsp HTTP/ 1.1
Accept: image/ gif, image/ jpg, */*
Accept-Encoding: gzip
Connection: Keep- Alive
Cookie: userID= id456578
Host: www.sun.com
Referer: http:/www.sun.com/codecamp.html
User-Agent: Mozilla/ 4.7 [en] (Win98; U)
```

HTTP Request Headers

- Accept
  - Indicates MIME types browser can handle.
- Accept-Encoding
  - Indicates encoding (e. g., gzip or compress) browser can handle
- Authorization
  - User identification for password- protected pages
  - Instead of HTTP authorization, use HTML forms to send username/password and store info in session object

HTTP Request Headers

- Connection
  - In HTTP 1.1, persistent connection is default
  - Servlets should set Content-Length with setContentLength (use ByteArrayOutputStream to determine length of output) to support persistent connections.
- Cookie
  - Gives cookies sent to client by server sometime earlier. Use getCookies, not getHeader
- Host
  - Indicates host given in original URL.
  - This is required in HTTP 1.1.

HTTP Request Headers

- If-Modified-Since
  - Indicates client wants page only if it has been changed after specified date.
  - Don’t handle this situation directly; implement getLastModified instead.
- Referer
  - URL of referring Web page.
  - Useful for tracking traffic; logged by many servers.
- User-Agent
  - String identifying the browser making the request.
  - Use with extreme caution!

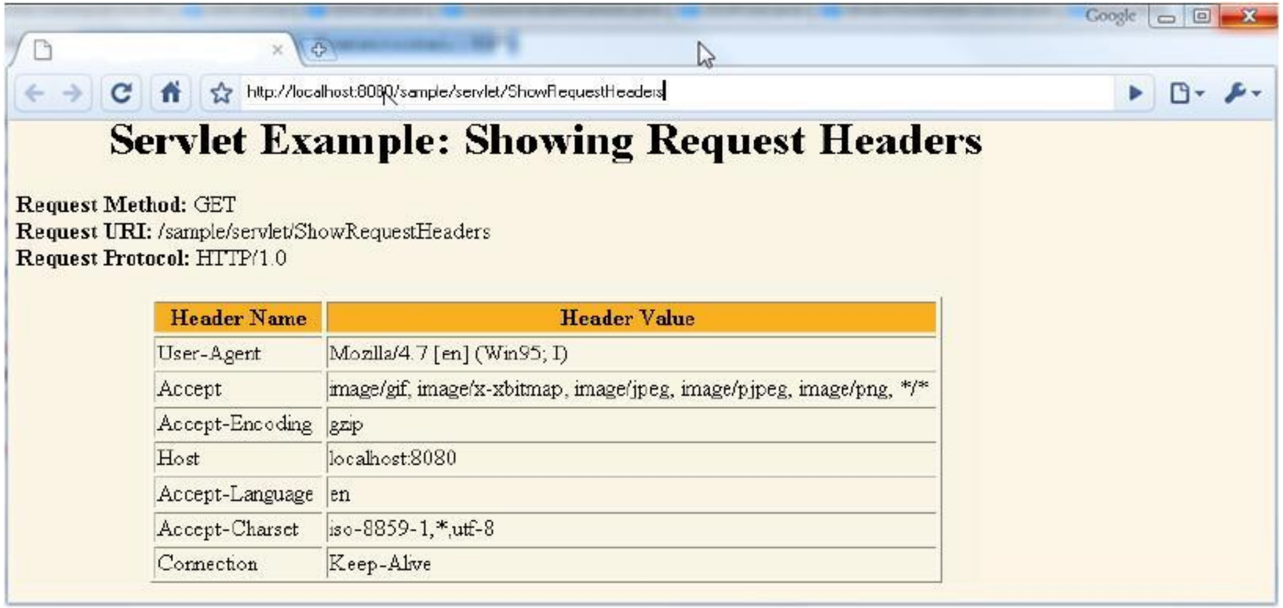
HTTP Header Methods

- String getHeader(java.lang.String name)
  - value of the specified request header as String
- java.util.Enumeration getHeaders(java.lang.String name)
  - values of the specified request header
- java.util.Enumeration getHeaderNames()
  - names of request headers
- int getIntHeader(java.lang.String name)
  - value of the specified request header as an int

Showing Request Headers

```
//Shows all the request headers sent on this particular request.
public class ShowRequestHeaders extends HttpServlet {
  public void doGet(HttpServletRequest request,
             HttpServletResponse response)
             throws ServletException, IOException {
      response.setContentType("text/html");
      PrintWriter out = response.getWriter();
      String title = "Servlet Example: Showing Request Headers";
      out.println("<HTML>" + ...
               "<B>Request Method: </B>" +
               request.getMethod() + "<BR>\n" +
               "<B>Request URI: </B>" +
               request.getRequestURI() + "<BR>\n" +
               "<B>Request Protocol: </B>" +
               request.getProtocol() + "<BR><BR>\n" +
             ...
               "<TH>Header Name<TH>Header Value");
      Enumeration headerNames = request.getHeaderNames();
      while(headerNames.hasMoreElements()) {
             String headerName = (String)headerNames.nextElement();
             out.println("<TR><TD>" + headerName);
             out.println(" <TD>" + request.getHeader(headerName));
      }
      ...
  }
}
```

Request Headers Sample



http://localhost:8080/sample/servlet/ShowRequestHeaders

Servlet Example: Showing Request Headers

Request Method: GET
Request URI: /sample/servlet/ShowRequestHeaders
Request Protocol: HTTP/1.0

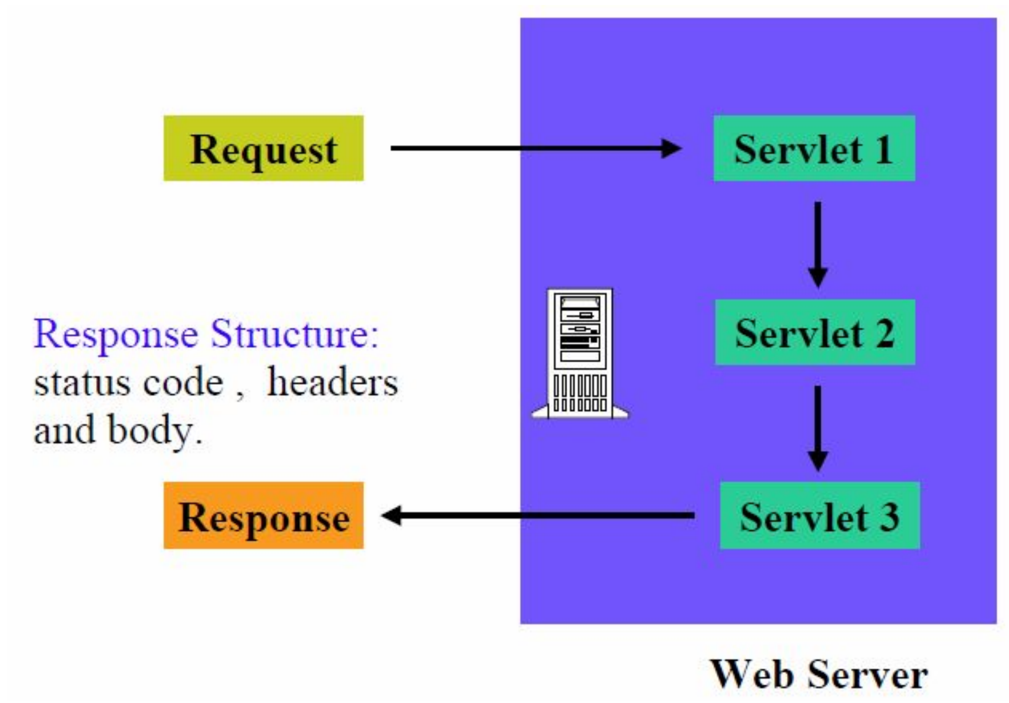
Header Name	Header Value
User-Agent	Mozilla/4.7 [en] (Win95; I)
Accept	image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, image/png, */*
Accept-Encoding	gzip
Host	localhost:8080
Accept-Language	en
Accept-Charset	iso-8859-1,*,utf-8
Connection	Keep-Alive

SERVLET RESPONSE (HTTPSERVLETRESPONSE)

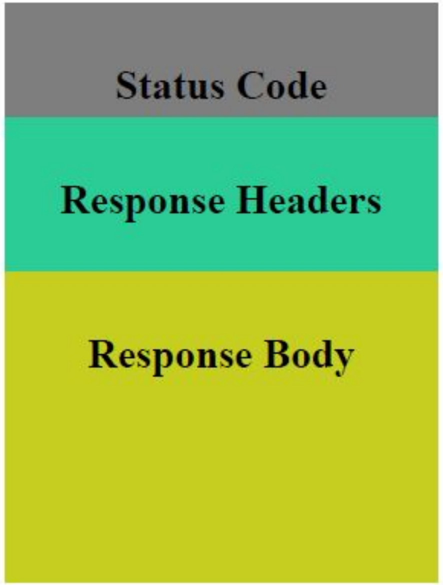
What is Servlet Response?

- Contains data passed from servlet to client
- All servlet responses implement ServletResponse interface
  - Retrieve an output stream
  - Indicate content type
  - Indicate whether to buffer output
  - Set localization information
- HttpServletResponse extends ServletResponse
  - HTTP response status code
  - Cookies

Responses


Request
Servlet 1
Servlet 2
Servlet 3
Response
Response Structure:
status code , headers
and body.
Web Server

Response Structure



HTTP Response Status Codes

- Why do we need HTTP response status code?
  - Forward client to another page
  - Indicates resource is missing
  - Instruct browser to use cached copy

Methods for Setting HTTP Response Status Codes

- public void setStatus(int statusCode)
  - Status codes are defined in HttpServletResponse
  - Status codes are numeric fall into five general categories:
    - 100-199 Informational
    - 200-299 Successful
    - 300-399 Redirection
    - 400-499 Incomplete
    - 500-599 Server Error
  - Default status code is 200 (OK)

Example of HTTP Response Status

```
HTTP/ 1.1 200 OK
Content-Type: text/ html
<! DOCTYPE ...>
<HTML
...
</ HTML>
```

Common Status Codes

- 200 (SC_OK)
  - Success and document follows
  - Default for servlets
- 204 (SC_No_CONTENT)
  - Success but no response body
  - Browser should keep displaying previous document
- 301 (SC_MOVED_PERMANENTLY)
  - The document moved permanently (indicated in Location header)
  - Browsers go to new location automatically

Common Status Codes

- 302 (SC_MOVED_TEMPORARILY)
  - Note the message is "Found“
  - Requested document temporarily moved elsewhere (indicated in Location header)
  - Browsers go to new location automatically
  - Servlets should use sendRedirect, not setStatus, when setting this header
- 401 (SC_UNAUTHORIZED)
  - Browser tried to access password- protected page without proper Authorization header
- 404 (SC_NOT_FOUND)
  - No such page

Methods for Sending Error

- Error status codes (400-599) can be used in sendError methods.
- public void sendError(int sc)
  - The server may give the error special treatment
- public void sendError(int code, String message)
  - Wraps message inside small HTML document

setStatus() & sendError()

```
try {
  returnAFile(fileName, out)
}
catch (FileNotFoundException e)
  { response.setStatus(response.SC_NOT_FOUND);
     out.println("Response body");
  }
          has same effect as
try {
  returnAFile(fileName, out)
}
catch (FileNotFoundException e)
  { response.sendError(response.SC_NOT_FOUND);
  }
```

Why HTTP Response Headers?

- Give forwarding location
- Specify cookies
- Supply the page modification date
- Instruct the browser to reload the page after a designated interval
- Give the file size so that persistent HTTP connections can be used
- Designate the type of document being generated
- Etc.

Methods for Setting Arbitrary Response Headers

- public void setHeader(String headerName, String headerValue)
  - Sets an arbitrary header.
- public void setDateHeader(String name, long millisecs)
  - Converts milliseconds since 1970 to a date string in GMT format
- public void setIntHeader(String name, int headerValue)
  - Prevents need to convert int to String before calling setHeader
- addHeader, addDateHeader, addIntHeader
  - Adds new occurrence of header instead of replacing.

Methods for Setting Arbitrary Response Headers

- setContentType
  - Sets the Content- Type header. Servlets almost always use this.
- setContentLength
  - Sets the Content- Length header. Used for persistent HTTP connections.
- addCookie
  - Adds a value to the Set- Cookie header.
- sendRedirect
  - Sets the Location header and changes status code.

Common HTTP 1.1 Response Headers

- Location
  - Specifies a document's new location.
  - Use sendRedirect instead of setting this directly.
- Refresh
  - Specifies a delay before the browser automatically reloads a page.
- Set-Cookie
  - The cookies that browser should remember. Don’t set this header directly.
  - use addCookie instead.

Common HTTP 1.1 Response Headers

- Cache-Control (1.1) and Pragma (1.0)
  - A no-cache value prevents browsers from caching page. Send both headers or check HTTP version.
- Content- Encoding
  - The way document is encoded. Browser reverses this encoding before handling document.
- Content- Length
  - The number of bytes in the response. Used for persistent HTTP connections.

Common HTTP 1.1 Response Headers

- Content- Type
  - The MIME type of the document being returned.
  - Use setContentType to set this header.
- Last- Modified
  - The time document was last changed
  - Don’t set this header explicitly.
  - provide a getLastModified method instead.

Refresh Sample Code

```
public class DateRefresh extends HttpServlet {
 public void doGet(HttpServletRequest req,
            HttpServletResponse res)
       throws ServletException, IOException {
    res.setContentType("text/plain");
    PrintWriter out = res.getWriter();
    res.setHeader("Refresh", "5");
    out.println(new Date().toString());
 }
}
```

Writing a Response Body

- A servlet almost always returns a response body
- Response body could either be a PrintWriter or a ServletOutputStream
- PrintWriter
  - Using response.getWriter()
  - For character-based output
- ServletOutputStream
  - Using response.getOutputStream()
  - For binary (image) data