

© 2003 Marty Hall



Handling the Client Request: HTTP Request Headers

JSP and Servlet Training Courses: <http://courses.coreservlets.com>
 JSP and Servlet Books from Sun Press: <http://www.coreservlets.com>

Agenda

- Reading HTTP request headers
- Building a table of all the request headers
- Understanding the various request headers
- Reducing download times by compressing pages
- Differentiating among types of browsers

JSP/Servlet training: <http://www.coreservlets.com>

A Typical HTTP Request

```
GET /servlet/Search?keywords=servlets+jsp HTTP/1.1
Accept: image/gif, image/jpg, */*
Accept-Encoding: gzip
Connection: Keep-Alive
Cookie: userID=id456578
Host: www.somebookstore.com
Referer: http://www.somebookstore.com/findbooks.html
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)
```

- It shouldn't take a rocket scientist to realize that you need to understand HTTP to be effective with servlets and JSP

JSP/Servlet training: <http://www.coreservlets.com>

Reading Request Headers (Methods in HttpServletRequest)

- **General**
 - `getHeader` (header name is not case sensitive)
 - `getHeaders`
 - `getHeaderNames`
- **Specialized**
 - `getCookies`
 - `getAuthType` and `getRemoteUser`
 - `getContentLength`
 - `getContentType`
 - `getDateHeader`
 - `getIntHeader`
- **Related info**
 - `getMethod`, `getRequestURI`, `getQueryString`, `getProtocol`

JSP/Servlet training: <http://www.coreservlets.com>

Checking For Missing Headers

- **HTTP 1.0**
 - All request headers are optional
- **HTTP 1.1**
 - Only Host is required
- **Conclusion**
 - Always check that `request.getHeader` is non-null before trying to use it

```
String val = request.getHeader("Some-Name");
if (val != null) {
    ...
}
```

JSP/Servlet training: <http://www.coreservlets.com>

Making a Table of All Request Headers

```
public class ShowRequestHeaders 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" +
        "<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" +
```

JSP/Servlet training: <http://www.coreservlets.com>

Making a Table of All Request Headers (Continued)

```

<TABLE BORDER=1 ALIGN="CENTER">\n" +
<TR BGCOLOR="\#FFAD00">\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));
}
out.println("</TABLE>\n</BODY></HTML>");
}
/** Since this servlet is for debugging, have it
* handle GET and POST identically.
*/
public void doPost(HttpServletRequest request,
HttpServletResponse response)
throws ServletException, IOException {
doGet(request, response);
}

```

ISP/servlet training: <http://www.coreservlets.com>

Making a Table of All Request Headers (Result 1)

Header Name	Header Value
accept	text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
accept-encoding	gzip
accept-language	en-us,en
accept-ranges	bytes
user-agent	Mozilla/5.0 (Windows; U; MSIE 6.0; Windows NT 5.1)
host	localhost
connection	Keep-Alive

ISP/servlet training: <http://www.coreservlets.com>

Making a Table of All Request Headers (Result 2)

Header Name	Header Value
host	localhost
user-agent	Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1.3) Gecko/20080326 Firefox/2.0
accept	text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
accept-encoding	gzip
accept-language	en-us,en
accept-ranges	bytes
user-agent	Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1.3) Gecko/20080326 Firefox/2.0
host	localhost
connection	Keep-Alive

ISP/servlet training: <http://www.coreservlets.com>

Common HTTP 1.1 Request Headers

- **Accept**
 - Indicates MIME types browser can handle
 - Can send different content to different clients. For example, PNG files have good compression characteristics but are not widely supported in browsers. A servlet could check to see if PNG is supported, sending `` if it is supported, and `` if not.
 - Warning: IE incorrectly sets this header when you hit the Refresh button. It sets it correctly on original request.
- **Accept-Encoding**
 - Indicates encodings (e.g., gzip or compress) browser can handle.
 - See following example

ISP/servlet training: <http://www.coreservlets.com>

Common HTTP 1.1 Request Headers (Continued)

- **Authorization**
 - User identification for password-protected pages.
 - See upcoming example.
 - Instead of HTTP authorization, use HTML forms to send username/password and store info in session object. This approach is usually preferable because standard HTTP authorization results in a small, terse dialog box that is unfamiliar to many users.
 - Servers have high-level way to set up password-protected pages without explicit programming in the servlets.
 - For details, see Chapter 7 (Declarative Security) and Chapter 8 (Programmatic Security) of *More Servlets and JavaServer Pages*, www.moreservlets.com.

ISP/servlet training: <http://www.coreservlets.com>

Common HTTP 1.1 Request Headers (Continued)

- **Connection**
 - In HTTP 1.0, keep-alive means browser can handle persistent connection. In HTTP 1.1, persistent connection is default. Persistent connections mean that the server can reuse the same socket over again for requests very close together from the same client (e.g., the images associated with a page, or cells within a framed page).
 - Servlets can't do this unilaterally; the best they can do is to give the server enough info to permit persistent connections. So, they should set Content-Length with `setContentLength` (using `ByteArrayOutputStream` to determine length of output).
- **Cookie**
 - Gives cookies previously sent to client. Use `getCookies`, not `getHeader`. See chapter & later class session.

ISP/servlet training: <http://www.coreservlets.com>

Common HTTP 1.1 Request Headers (Continued)

- **Host**
 - Indicates host given in original URL
 - This is a *required* header in HTTP 1.1. This fact is important to know if you write a custom HTTP client (e.g., WebClient used in book) or telnet to a server and use the HTTP/1.1 version.
- **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.
 - See lottery-number example in book (*Core Servlets & JSP (2nd Ed)* Chapter 3).

JSPServlet training: <http://www.coreservlets.com>

Common HTTP 1.1 Request Headers (Continued)

- **Referer**
 - URL of referring Web page
 - Useful for tracking traffic; logged by many servers
 - Can also be used to let users set preferences and then return to the page they came from
 - Can be easily spoofed; don't let this header be sole means of deciding how much to pay sites that show your banner ads.
 - Some browsers (Opera), ad filters (Web Washer), and personal firewalls (Norton) screen out this header
 - See example in book
- **User-Agent**
 - Best used for identifying *category* of client
 - Web browser vs. I-mode cell phone, etc.
 - For Web applications, use other headers if possible
 - Again, can be easily spoofed
 - See following example

JSPServlet training: <http://www.coreservlets.com>

Sending Compressed Web Pages



Dilbert used with permission of United Syndicates Inc.

JSPServlet training: <http://www.coreservlets.com>

Sending Compressed Pages: GzipUtilities.java

```
public class GzipUtilities {
    public static boolean isGzipSupported
        (HttpServletRequest request) {
        String encodings = request.getHeader("Accept-Encoding");
        return((encodings != null) &&
            (encodings.indexOf("gzip") != -1));
    }

    public static boolean isGzipDisabled
        (HttpServletRequest request) {
        String flag = request.getParameter("disableGzip");
        return((flag != null)&&
            (!flag.equalsIgnoreCase("false")));
    }

    public static PrintWriter getGzipWriter
        (HttpServletResponse response) throws IOException {
        return(new PrintWriter
            (new GZIPOutputStream
            (response.getOutputStream())));
    }
}
```

JSPServlet training: <http://www.coreservlets.com>

Sending Compressed Pages: LongServlet.java

```
public class LongServlet extends HttpServlet {
    public void doGet(HttpServletRequest request,
        HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html");

        // Change the definition of "out" depending on
        // whether or not gzip is supported.
        PrintWriter out;
        if (GzipUtilities.isGzipSupported(request) &&
            !GzipUtilities.isGzipDisabled(request)) {
            out = GzipUtilities.getGzipWriter(response);
            response.setHeader("Content-Encoding", "gzip");
        } else {
            out = response.getWriter();
        }
    }
}
```

JSPServlet training: <http://www.coreservlets.com>

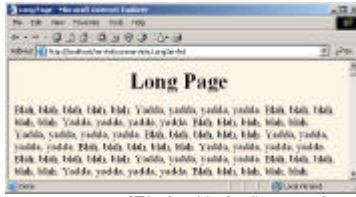
Sending Compressed Pages: LongServlet.java (Continued)

```
...
    out.println
        (docType +
            "<HTML>\n" +
            "<HEAD><TITLE>" + title + "</TITLE></HEAD>\n" +
            "<BODY BGCOLOR=\"#FDF5E6\">\n" +
            "<H1 ALIGN=\"CENTER\">" + title + "</H1>\n");
    String line = "Blah, blah, blah, blah, blah." +
        "Yadda, yadda, yadda, yadda, yadda.";
    for(int i=0; i<10000; i++) {
        out.println(line);
    }
    out.println("</BODY></HTML>");
    out.close();
}
```

JSPServlet training: <http://www.coreservlets.com>

Sending Compressed Pages: Results

- **Uncompressed (28.8K modem), Netscape and Internet Explorer: > 50 seconds**
- **Compressed (28.8K modem), Netscape and Internet Explorer: < 5 seconds**
- **Caution: be careful about generalizing benchmarks**



ISP/servlet training: <http://www.coreservlets.com>

Differentiating Among Different Browser Types

- **Use User-Agent only when necessary.**
 - Otherwise, you will have difficult-to-maintain code that consists of tables of browser versions and associated capabilities.
- **Check for null.**
 - The header is not required by the HTTP 1.1 specification, some browsers let you disable it (e.g., Opera), and custom clients (e.g., Web spiders or link verifiers) might not use the header at all.
- **To differentiate between Netscape and Internet Explorer, check for “MSIE,” not “Mozilla.”**
 - Both Netscape and Internet Explorer say “Mozilla” at the beginning of the header.
 - For JavaScript compatibility.
- **Note that the header can be faked.**
 - If a client fakes this header, the servlet cannot tell the difference.

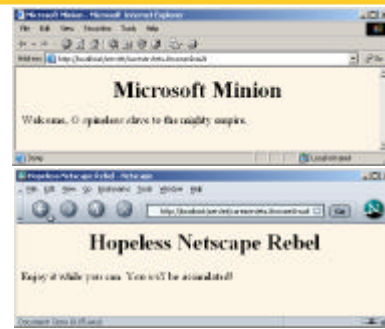
ISP/servlet training: <http://www.coreservlets.com>

Differentiating Among Different Browser Types (Code)

```
public class BrowserInsult extends HttpServlet {
    public void doGet(HttpServletRequest request,
        HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        String title, message;
        // Assume for simplicity that Netscape and IE are
        // the only two browsers.
        String userAgent = request.getHeader("User-Agent");
        if ((userAgent != null) &&
            (userAgent.indexOf("MSIE") != -1)) {
            title = "Microsoft Minion";
            message = "Welcome, O spineless slave to the " +
                "mighty empire.";
        } else {
            title = "Hopeless Netscape Rebel";
            message = "Enjoy it while you can. " +
                "You <I>will</I> be assimilated!";
        }
    }
}
```

ISP/servlet training: <http://www.coreservlets.com>

Differentiating Among Browser Types (Result)



ISP/servlet training: <http://www.coreservlets.com>

Summary

- Many servlet tasks can *only* be accomplished by making use of HTTP headers coming from the browser
- Use `request.getHeader` for arbitrary header
 - Remember to check for null
- Cookies, authorization info, content length, and content type have shortcut methods
- Most important headers you read directly
 - Accept
 - Accept-Encoding
 - Connection
 - Referer
 - User-Agent

ISP/servlet training: <http://www.coreservlets.com>



Questions?

JSP and Servlet Training Courses: <http://courses.coreservlets.com>
 JSP and Servlet Books from Sun Press: <http://www.coreservlets.com>