Handling the Client Request: HTTP Request Headers

Originals of Slides and Source Code for Examples:
http://courses.coreservlets.com/Course-Materials/csajsp2.html

Customized Java EE Training: http://courses.coreservlets.com/
Java, JSF 2, PrimeFaces, Servlets, JSP, Ajax, jQuery, Spring, Hibernate, RESTful Web Services, Hadoop, Android.
Developed and taught by well-known author and developer. At public venues or onsite at your location.

For live Java EE training, please see training courses at http://courses.coreservlets.com/.
JSF 2, PrimeFaces, Servlets, JSP, Ajax (with jQuery), GWT,
Android development, Java 6 and 7 programming,
SOAP-based and RESTful Web Services, Spring, Hibernate/JPA,
XML, Hadoop, and customized combinations of topics.

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. Contact hall@coreservlets.com for details.
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

A Typical HTTP Request

GET /search-servlet?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
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

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

```java
String val = request.getHeader("Some-Name");
if (val != null) {
    ...
}
```
Making a Table of All Request Headers

```java
@WebServlet("/show-request-headers")
public class ShowRequestHeaders extends HttpServlet {
    public void doGet(HttpServletRequest request,
                        HttpServletResponse response)
                        throws ServletException, IOException {
        ... // Content-Type, PrintWriter, docType defn, etc.
        out.println
        (docType +
          "<HTML>
          "+title+"</TITLE></HEAD>
          +""+"<BODY BGCOLOR="#FDF5E6">
          +""+"<H1 ALIGN="CENTER">" + title + "</H1>
          +""+"<B>Request Method: </B>" +
          request.getMethod() + "<BR>
          +""+"<B>Request URI: </B>
          +request.getRequestURI() + "<BR>
          +""+"<B>Request Protocol: </B>
          +request.getProtocol() + "<BR>

        "+"<TABLE BORDER=1 ALIGN="CENTER">" +
          "<TR BGCOLOR="#FFAD00">
          "+"<TH>Header Name</TH>" +
          "<TH>Header Value</TH>");
        Enumeration headerNames =
                        request.getHeaderNames();
        while (headerNames.hasMoreElements()) {
            String headerName = headerNames.nextElement();
            out.println("<TR><TD>" + headerName);
            out.println("<TD>" + request.getHeader(headerName));
        }
        out.println("</TABLE></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);
    }
}
```
As of summer 2010, Firefox was the #1 browser for visitors to coreservlets.com, accounting for 41.8% of the traffic.

As of summer 2010, IE was the #2 browser for visitors to coreservlets.com, accounting for 41.6% of the traffic.
Making a Table of All Request Headers (Chrome)

As of summer 2010, Chrome was the #3 browser for visitors to coreservlets.com, accounting for 12.7% of the traffic. Safari was fourth at 1.2%.

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 `<IMG SRC="picture.png" ...>` if it is supported, and `<IMG SRC="picture.gif" ...>` 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
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.

• **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.
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).

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
    • Regular Web browser vs. iPhone, etc.
  – For Web applications, use other headers if possible
  – Again, can be easily spoofed
  – See following example
Sending Compressed Web Pages

I calculated the total time that humans have waited for web pages to load...

It cancels out all the productivity gains of the information age.

Sometimes I think the web is a big plot to keep people like me away from normal society.

Uh-oh, he's on to me.

Dilbert used with permission of United Syndicates Inc.

18

Sending Compressed Pages: GzipUtilities.java

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

    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())));
    }
}
```

19
Sending Compressed Pages: LongServlet.java

@WebServlet("/long-servlet")
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();
        }

        String line = "Blah, blah, blah, blah, blah. " +
                      "Yadda, yadda, yadda, yadda.";
        for(int i=0; i<10000; i++) {
            out.println(line);
        }
        out.println("</BODY></HTML>");
        out.close();
    }
}
Sending Compressed Pages: Results

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

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.
- Differentiating among clients
  - To differentiate among Firefox, Netscape, and Internet Explorer, check for “MSIE,” not “Mozilla.”
    - Both Firefox and Internet Explorer say “Mozilla” at the beginning of the header. (For JavaScript compatibility.)
  - Detecting Safari on the iPhone
    - Mozilla/5.0 (iPhone; U; CPU like Mac OS X; en) AppleWebKit/420+(KHTML, like Gecko) Version/3.0 Mobile/1A543a Safari/419.3
- The header can be faked.
  - If a client fakes this header, the servlet cannot tell the difference.
Differentiating Among Different Browser Types (Code)

```java
@WebServlet("/browser-insult")
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 Firefox and IE are
        // the only two browsers.
        String userAgent = request.getHeader("User-Agent");
        if ((userAgent != null) &&
            (userAgent.contains("MSIE"))) {
            title = "Microsoft Minion";
            message = "Welcome, O spineless slave to the " +
                      "mighty empire."
        } else {
            title = "Hopeless Firefox Rebel";
            message = "Enjoy it while you can. " +
                      "You <I>will</I> be assimilated!";
        }
```
Summary

• **HTTP is important**
  – 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`

• **Shortcuts discussed later**
  – Cookies, authorization info, content length, and content type have shortcut methods

• **Most important headers you read directly**
  – Accept
  – Accept-Encoding
  – Connection
  – Referer
  – User-Agent