



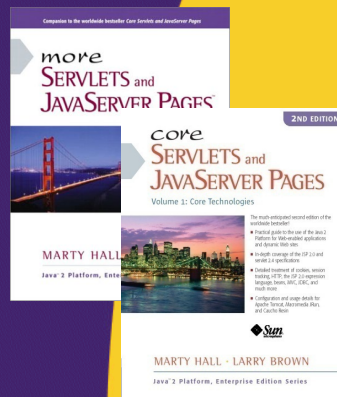
# Handling the Client Request: HTTP Request Headers

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](mailto: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

4

## 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

5

# 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`

6

# 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) {  
    ...  
}
```

7

# Making a Table of All Request Headers

```
@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>\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" +
```

8

# 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<String> headerNames =
            request.getHeaderNames();
        while(headerNames.hasMoreElements()) {
            String headerName = headerNames.nextElement();
            out.println("<TR><TD>" + headerName);
            out.println("    <TD>" + request.getHeader(headerName));
        }
        out.println("</TABLE>\n</BODY></HTML>");
    }

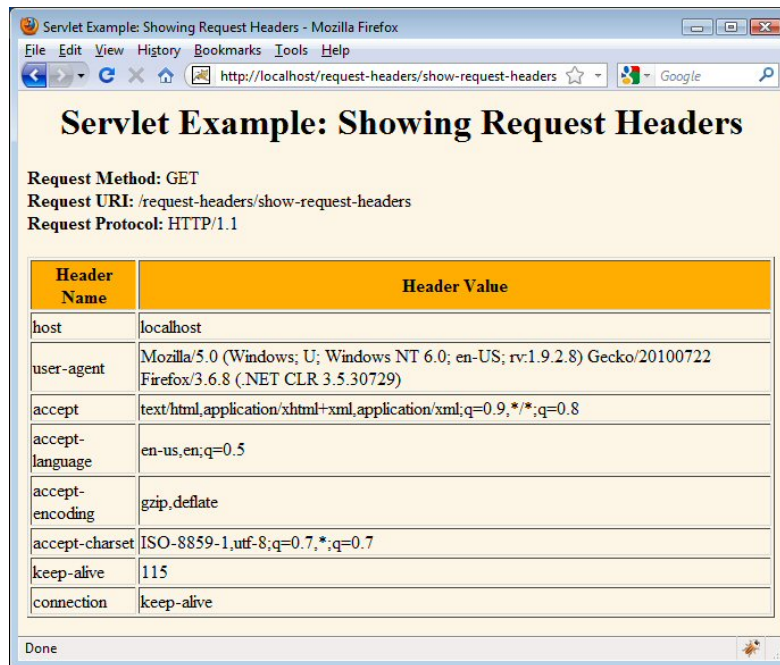
    /** Since this servlet is for debugging, have it
     *  handle GET and POST identically. */

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

9



# Making a Table of All Request Headers (Firefox)



Servlet Example: Showing Request Headers

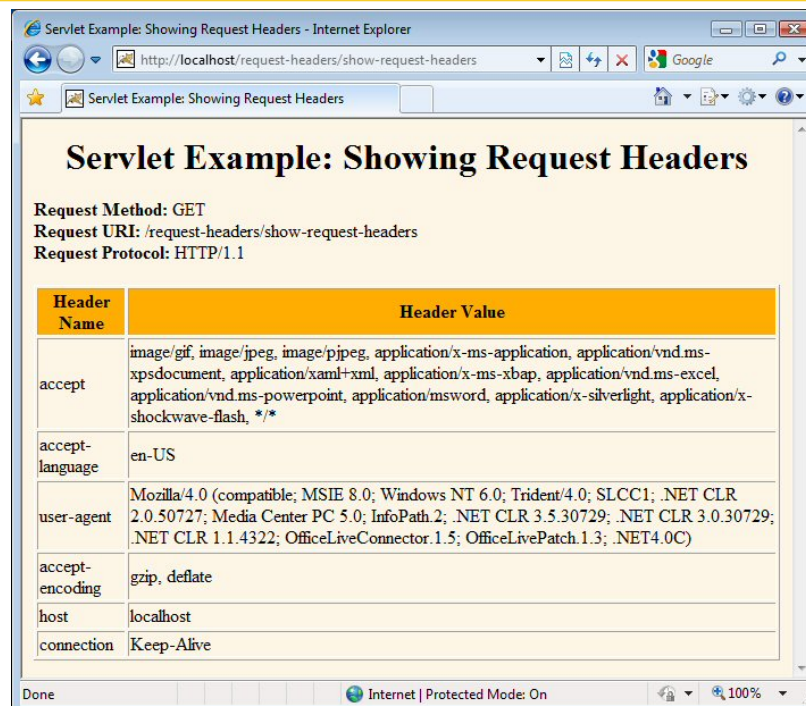
Request Method: GET  
Request URI: /request-headers/show-request-headers  
Request Protocol: HTTP/1.1

Header Name	Header Value
host	localhost
user-agent	Mozilla/5.0 (Windows; U; Windows NT 6.0; en-US; rv:1.9.2.8) Gecko/20100722 Firefox/3.6.8 (.NET CLR 3.5.30729)
accept	text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
accept-language	en-us,en;q=0.5
accept-encoding	gzip,deflate
accept-charset	ISO-8859-1,utf-8;q=0.7,*;q=0.7
keep-alive	115
connection	keep-alive

Done

As of summer 2010, Firefox was the #1 browser for visitors to coreservlets.com, accounting for 41.8% of the traffic.

# Making a Table of All Request Headers (Internet Explorer)



Servlet Example: Showing Request Headers

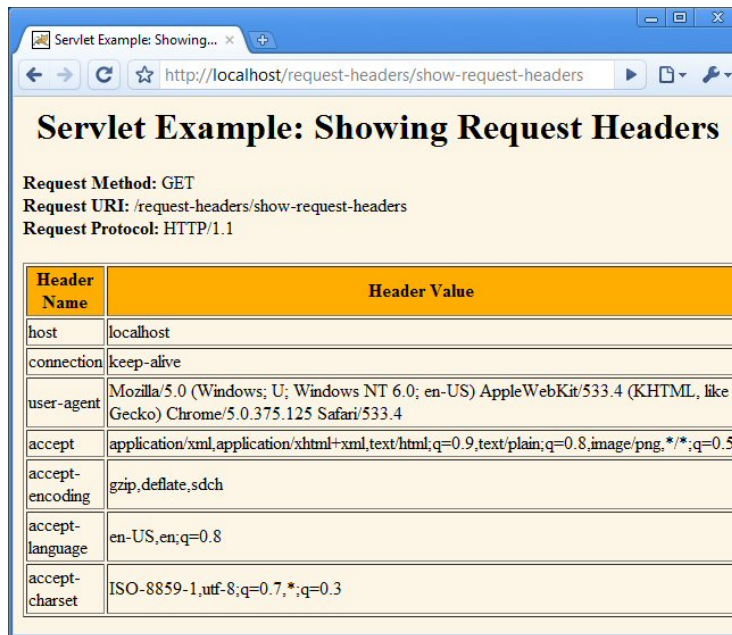
Request Method: GET  
Request URI: /request-headers/show-request-headers  
Request Protocol: HTTP/1.1

Header Name	Header Value
accept	image/gif, image/jpeg, image/pjpeg, application/x-ms-application, application/vnd.ms-xpsdocument, application/xaml+xml, application/x-ms-xbap, application/vnd.ms-excel, application/vnd.ms-powerpoint, application/msword, application/x-silverlight, application/x-shockwave-flash, */*
accept-language	en-US
user-agent	Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.0; Trident/4.0; SLCC1; .NET CLR 2.0.50727; Media Center PC 5.0; InfoPath.2; .NET CLR 3.5.30729; .NET CLR 3.0.30729; .NET CLR 1.1.4322; OfficeLiveConnector.1.5; OfficeLivePatch.1.3; .NET4.0C)
accept-encoding	gzip, deflate
host	localhost
connection	Keep-Alive

Done

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)



Header Name	Header Value
host	localhost
connection	keep-alive
user-agent	Mozilla/5.0 (Windows; U; Windows NT 6.0; en-US) AppleWebKit/533.4 (KHTML, like Gecko) Chrome/5.0.375.125 Safari/533.4
accept	application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
accept-encoding	gzip,deflate,sdch
accept-language	en-US,en;q=0.8
accept-charset	ISO-8859-1,utf-8;q=0.7,*;q=0.3

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%

12

## 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

13

## 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](http://www.moreservlets.com).

14

## 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.

15

## 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).

16

## 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

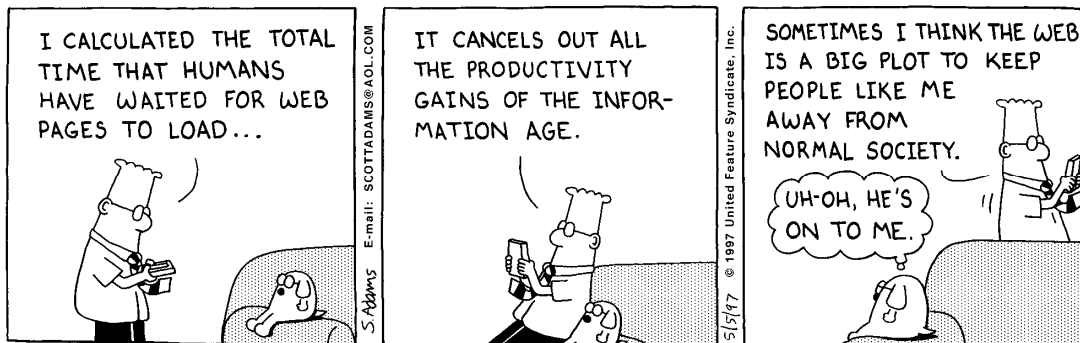
- **User-Agent**

- 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

17



# Sending Compressed Web Pages



Dilbert used with permission of United Syndicates Inc.

18

# Sending Compressed Pages: GzipUtilities.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();
        }
    }
}
```

20

## 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.";
for(int i=0; i<10000; i++) {
    out.println(line);
}
out.println("</BODY></HTML>");
out.close();
}
}
```

21

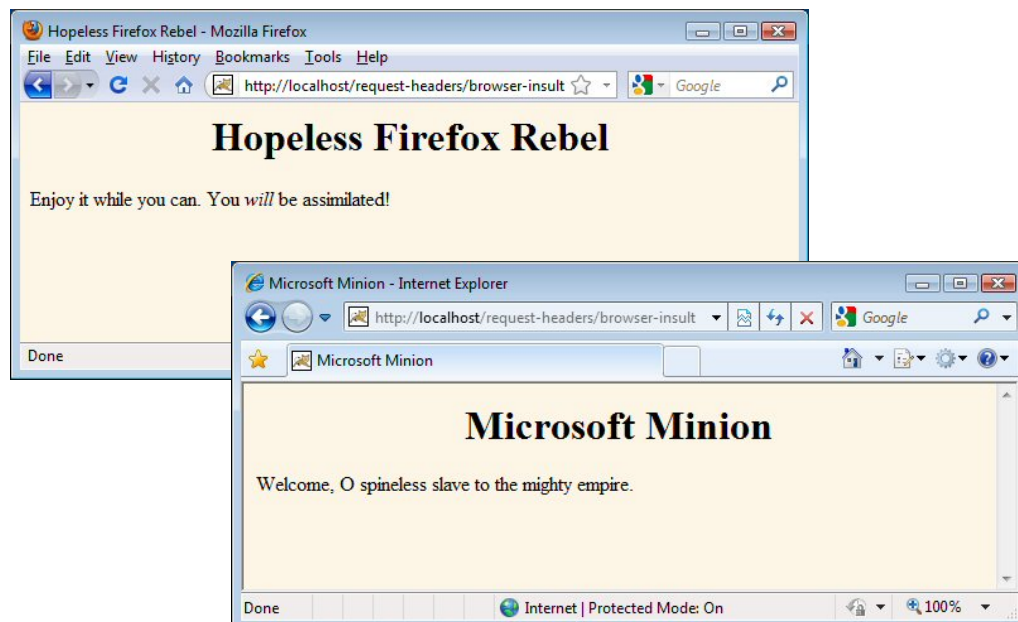


# Differentiating Among Different Browser Types (Code)

```
@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!";
        }
    }
}
```

24

# Differentiating Among Browser Types (Result)



25

# 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

26

© 2012 Marty Hall



## Questions?

[JSF 2, PrimeFaces, Java 7, Ajax, jQuery, Hadoop, RESTful Web Services, Android, Spring, Hibernate, Servlets, JSP, GWT, and other Java EE training](#)

**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.

27