

*Core Servlets and JavaServer Pages / 2e*

*Volume 1: Core Technologies*

*Marty Hall • Larry Brown*

**Handling the Client  
Request: HTTP  
Request Headers**

# 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**

# HTTP Request Header

- **This information is not the same as the form (query) data**
- **Form data result from user input and is sent via GET or POST**
- **Request headers are sent indirectly by the browser**

# 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)

- **The server needs to explicitly read the request headers to make use of this information**

# Reading Request Headers (Methods in HttpServletRequest)

- **General** <http://java.sun.com/j2ee/1.4/docs/api/index.html>
  - **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

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

# 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" +
```

# Making a Table of All Request Headers (Continued)

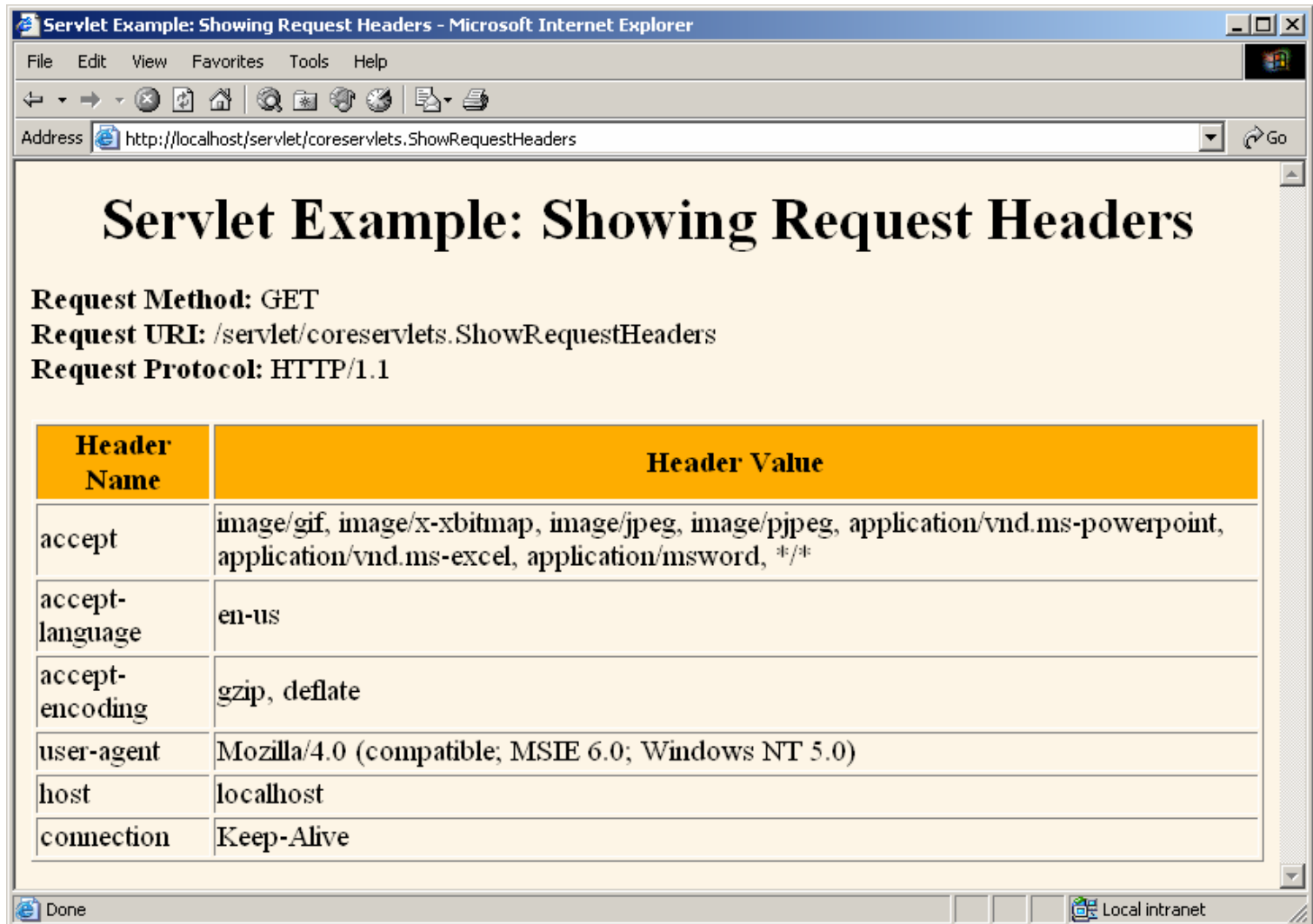
```
        "<TABLE BORDER=1 ALIGN=\"CENTER\">\n" +
        "<TR BGCOLOR=\"#FFAD00\">\n" +
        "<TH>Header Name<TH>Header Value");
Enumeration headerNames = request.getHeaderNames();
while(headerNames.hasMoreElements()) {
    String headerName = (String)headerNames.nextElement();
    out.println("<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);
}
```



# Making a Table of All Request Headers (Result 1)

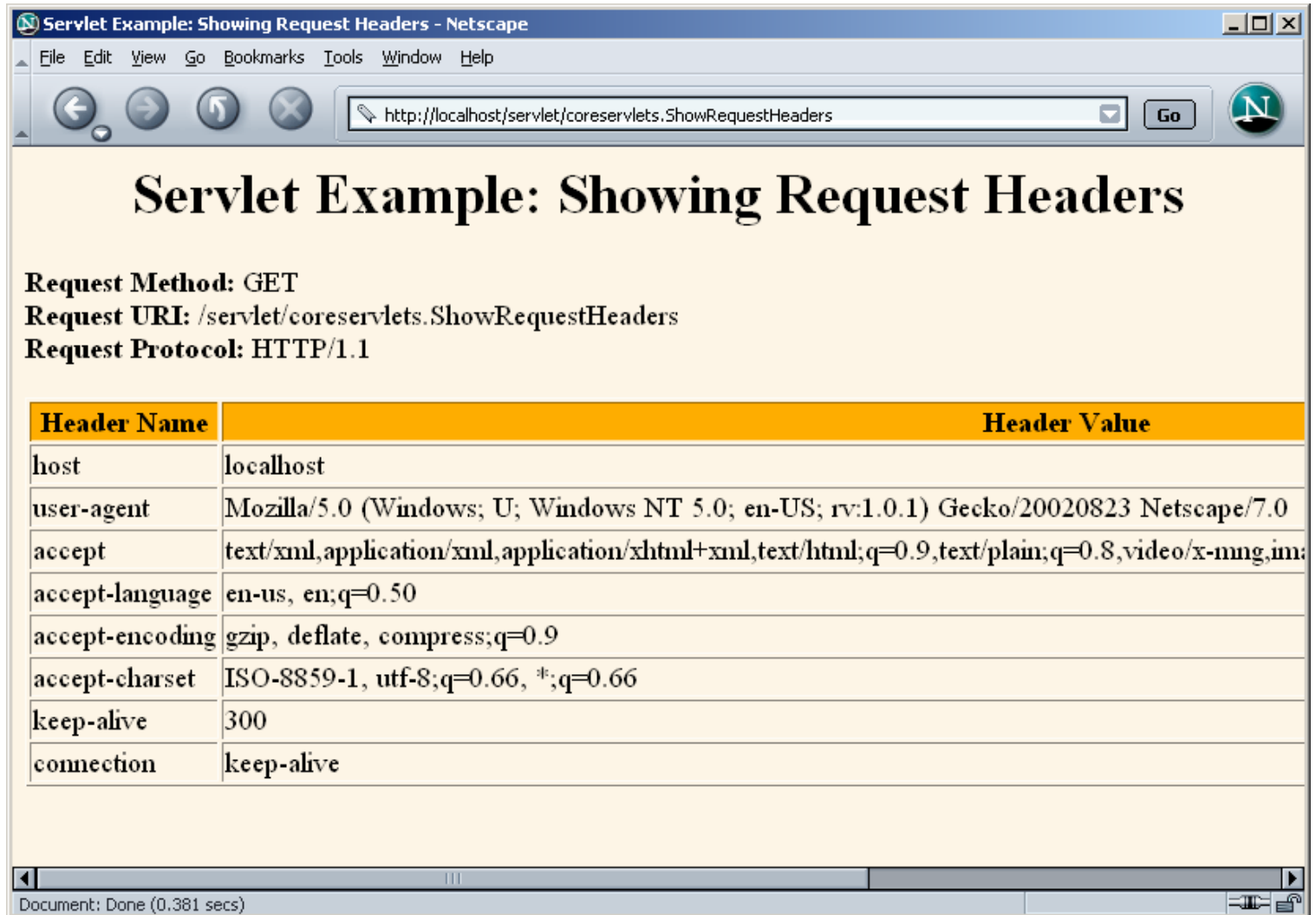


**Servlet Example: Showing Request Headers**

**Request Method:** GET  
**Request URI:** /servlet/coreservlets.ShowRequestHeaders  
**Request Protocol:** HTTP/1.1

| Header Name     | Header Value  |
|-----------------|---|
| accept          | image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/vnd.ms-powerpoint, application/vnd.ms-excel, application/msword, */* |
| accept-language | en-us   |
| accept-encoding | gzip, deflate   |
| user-agent      | Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)  |
| host            | localhost   |
| connection      | Keep-Alive  |

# Making a Table of All Request Headers (Result 2)



**Servlet Example: Showing Request Headers**

**Request Method:** GET  
**Request URI:** /servlet/coreservlets.ShowRequestHeaders  
**Request Protocol:** HTTP/1.1

| Header Name     | Header Value   |
|-----------------|--|
| host            | localhost  |
| user-agent      | Mozilla/5.0 (Windows; U; Windows NT 5.0; en-US; rv:1.0.1) Gecko/20020823 Netscape/7.0                        |
| accept          | text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,video/x-mng,image/jpeg;q=0.8 |
| accept-language | en-us, en;q=0.50   |
| accept-encoding | gzip, deflate, compress;q=0.9  |
| accept-charset  | ISO-8859-1, utf-8;q=0.66, *;q=0.66   |
| keep-alive      | 300  |
| connection      | keep-alive   |

Document: Done (0.381 secs)

# Common HTTP 1.1 Request Headers

- **Accept**

- Indicates MIME types browser can handle
  - Multipurpose internet mail extension

`text/html`      `application/pdf`      `image/gif`

- 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 5 & 6 incorrectly set this header when you hit the Refresh button. They set it correctly on original request.

- **Accept-Encoding**

- Indicates encodings (e.g., gzip or compress) browser can handle.
- gzip can reduce download time by a factor of 10

# Common HTTP 1.1 Request Headers (Continued)

- **Authorization**

- User identification for password-protected pages.
- 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).

# 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).
  - This saves the overhead of negotiating several independent connections
- Servlets need to cooperate with the server by making it possible for the server to use persistent connections.
  - This is done by the servlet by setting the Content-Length response header

- **Cookie**

- Returns cookies to servers that previously sent them to the browser. Use **.getCookies**, not **getHeader**.

# Common HTTP 1.1 Request Headers (Continued)

- **Host**

- Indicates host and port 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
  - Server sends a 304 (Not Modified) header if no newer result is available
- Server should not handle this situation directly; implement getLastModified instead to have the system handle modification dates automatically

# Common HTTP 1.1 Request Headers (Continued)

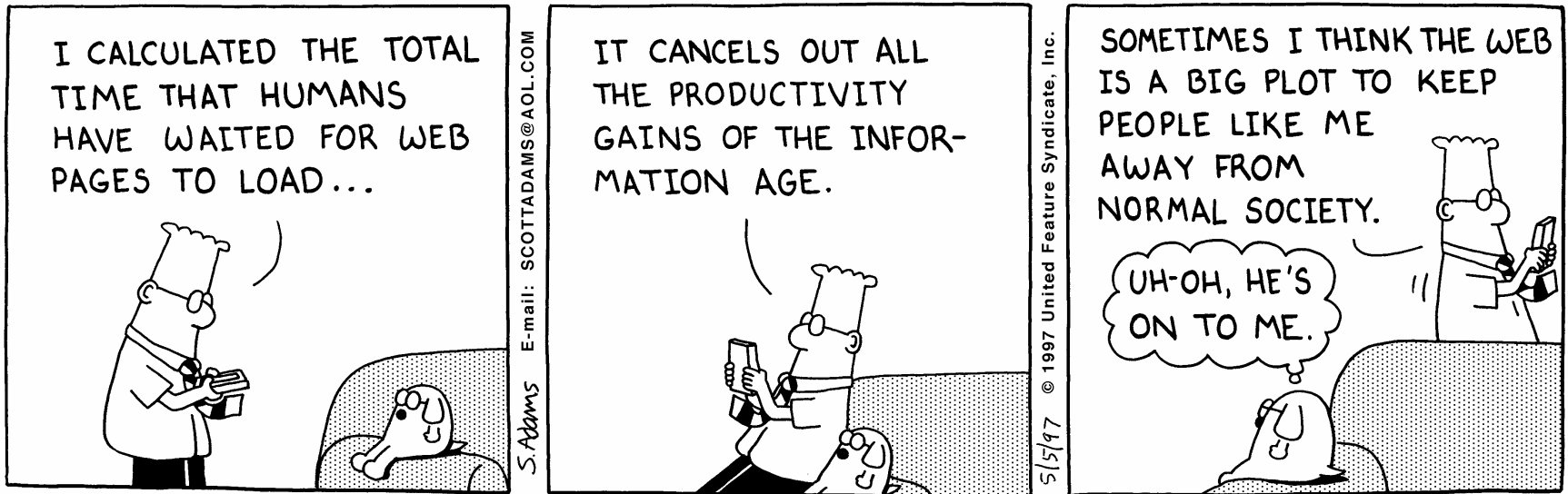
- **Referer**

- URL of referring Web page; sent when user clicks on link
- 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

- **User-Agent**

- Best used for identifying *category* of client so that different content can be returned to different browser types
  - Web browser vs. I-mode cell phone, etc.
- Most IE versions list Mozilla with the real browser listed in parentheses - this is for Javascript compatibility.
- Again, can be easily spoofed

# Sending Compressed Web Pages



Dilbert used with permission of United Syndicates Inc.



# Sending Compressed Pages

- **Gzip**

- A text compression scheme that can dramatically reduce the size of HTML or plain text pages
- The server compresses the document, sends the smaller file over the network and the browser automatically reverses the compression
- Particularly useful over dialup connections
- Must check the Accept-Encoding parameter in the request header to verify that the browser supports gzip

- **Class GzipUtilities**

- Requires import of `java.util.zip.*`
- `isGzipSupported` checks request header Accept-Encoding
- `isGzipDisabled` checks parameter to see if gzip is disabled
- `getGzipWriter` instantiates a `PrintWriter` that can use gzip

# 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
        (HttpServletRequest response) throws IOException {
        return(new PrintWriter
            (new GZIPOutputStream
                (response.getOutputStream())));
    }
}
```

# Sending Compressed Pages: LongServlet.java

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

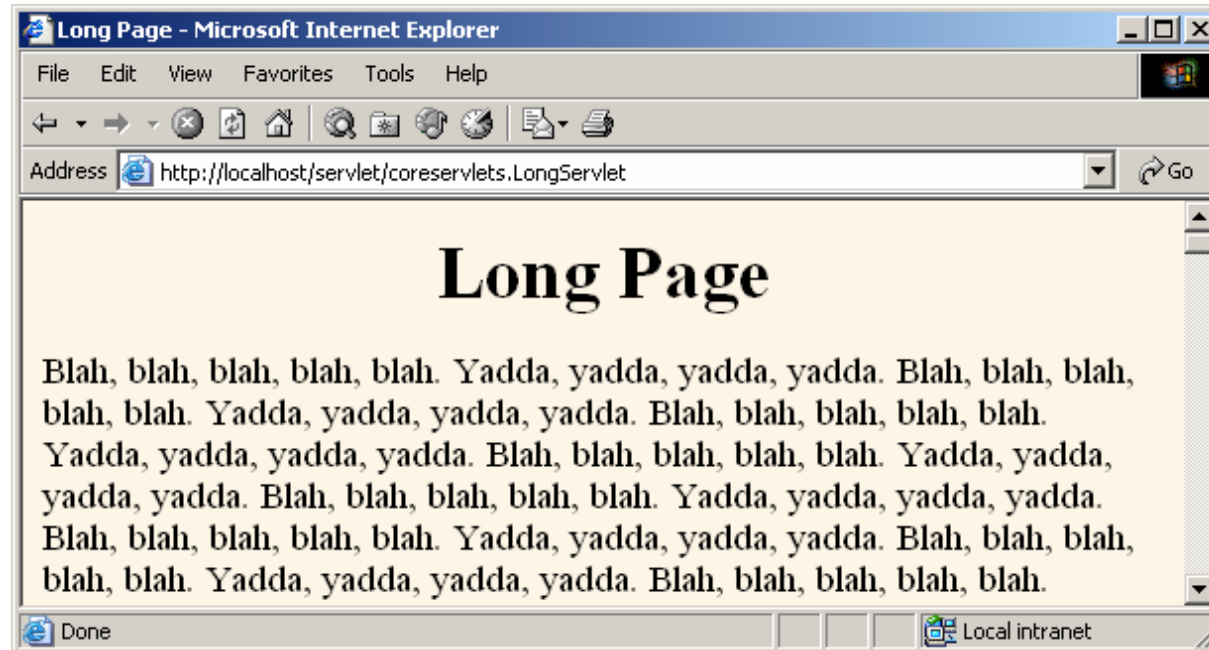
        // Change the definition of "out" depending on
        // whether or not gzip is supported.
        PrintWriter out;
        if (GzipUtilities.isGzipSupported(request) &&
            !GzipUtilities.isGzipDisabled(request)) {
            out = GzipUtilities.getGzipWriter(response);
            response.setHeader("Content-Encoding",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

# 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(); //Needed for gzip; always recommended
}
}
```

# 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



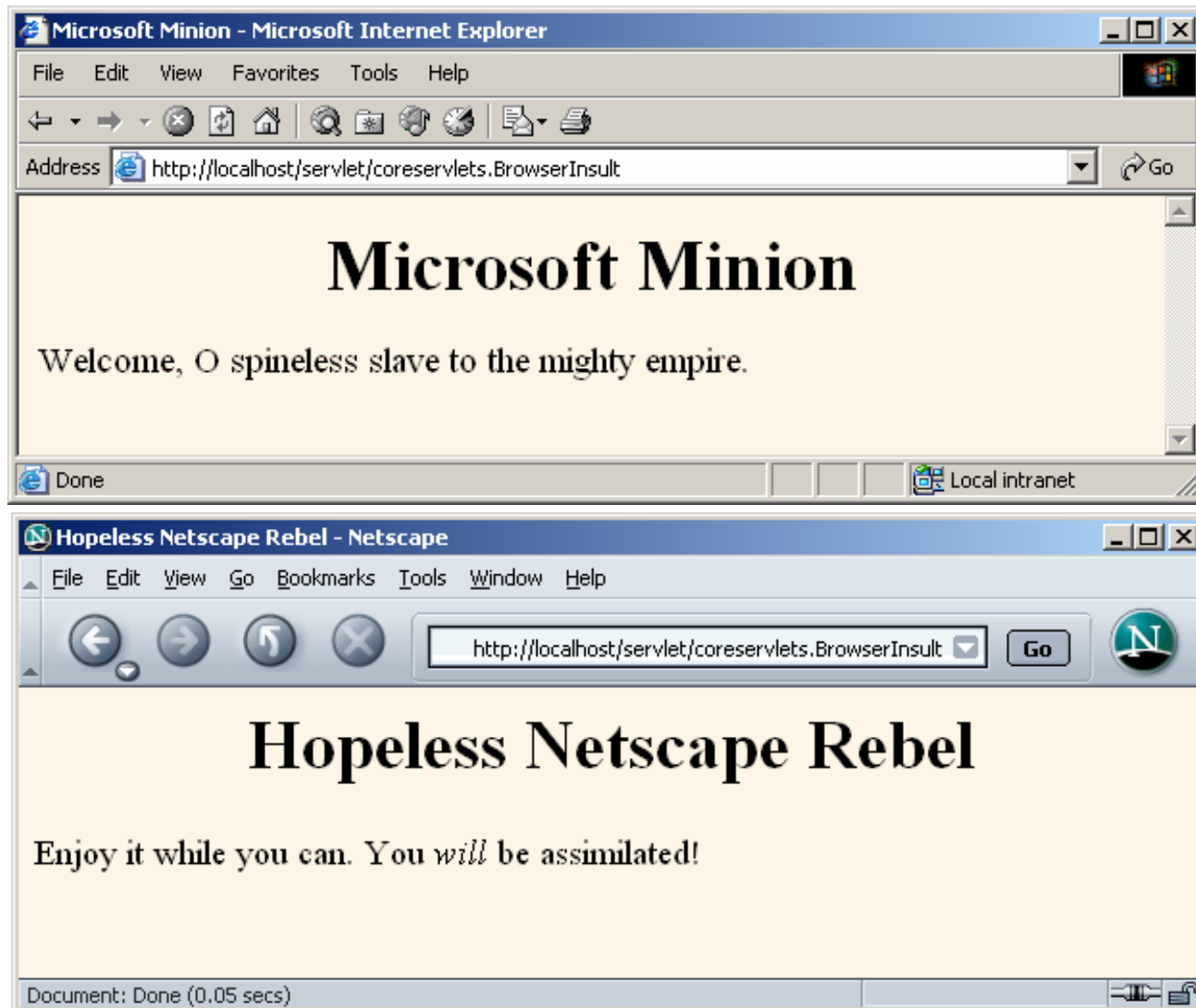
# Differentiating Among Different Browser Types

- **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.

# 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!";
        }
    }
}
```

# Differentiating Among Browser Types (Result)





# Where you came from

- **The Referer header tells you the location of the page users were on when they clicked a link to get to the current page**
  - If the user typed in the address, the `request.getHeader("Referer")` returns `null`
  - This enables you to customize the current page based on how the user got here
  - Also, you can create a link to take the user back to the page he came from
  - Using the default Tomcat directory structure, HTML files are put under `ROOT/request-headers/` while images are stored under `ROOT/request-headers/images`

# Where you came from

```
public class CustomizeImage extends HttpServlet {

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

        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        String referer = request.getHeader("Referer");
        if (referer == null) { referer = "<I>none</I>"; }

        String title = "Referring page: " + referer;
        String imageName;
        if (contains(referer, "JRun"))
            { imageName = "jrun-powered.gif"; }
        else if (contains(referer, "Resin"))
            { imageName = "resin-powered.gif"; }
        else { imageName = "tomcat-powered.gif"; }
    }
}
```

# Where you came from

```
String imagePath = "../request-headers/images/" +
imageName;
```

```
String docType = "<!DOCTYPE HTML PUBLIC \\"-
//W3C//DTD HTML 4.0 \" + \"Transitional//EN\\\">\n\";
```

```
out.println(docType + "<HTML>\n" + "<HEAD><TITLE>\"
+ title + "</TITLE></HEAD>\n" + "<BODY
BGCOLOR=\\\"#FDF5E6\\\">\n" + "<CENTER><H2>\" + title +
"</H2>\n" + "<IMG SRC=\\\"\" + imagePath + "\\\"> \n" +
"</CENTER></BODY></HTML>\" );
} // end doGet
```

```
private boolean contains(String mainString, String
subString)
{
    return(mainString.indexOf(subString) != -1);
}
}
```