Lecture 4
Handling the Client Request: HTTP Request Headers

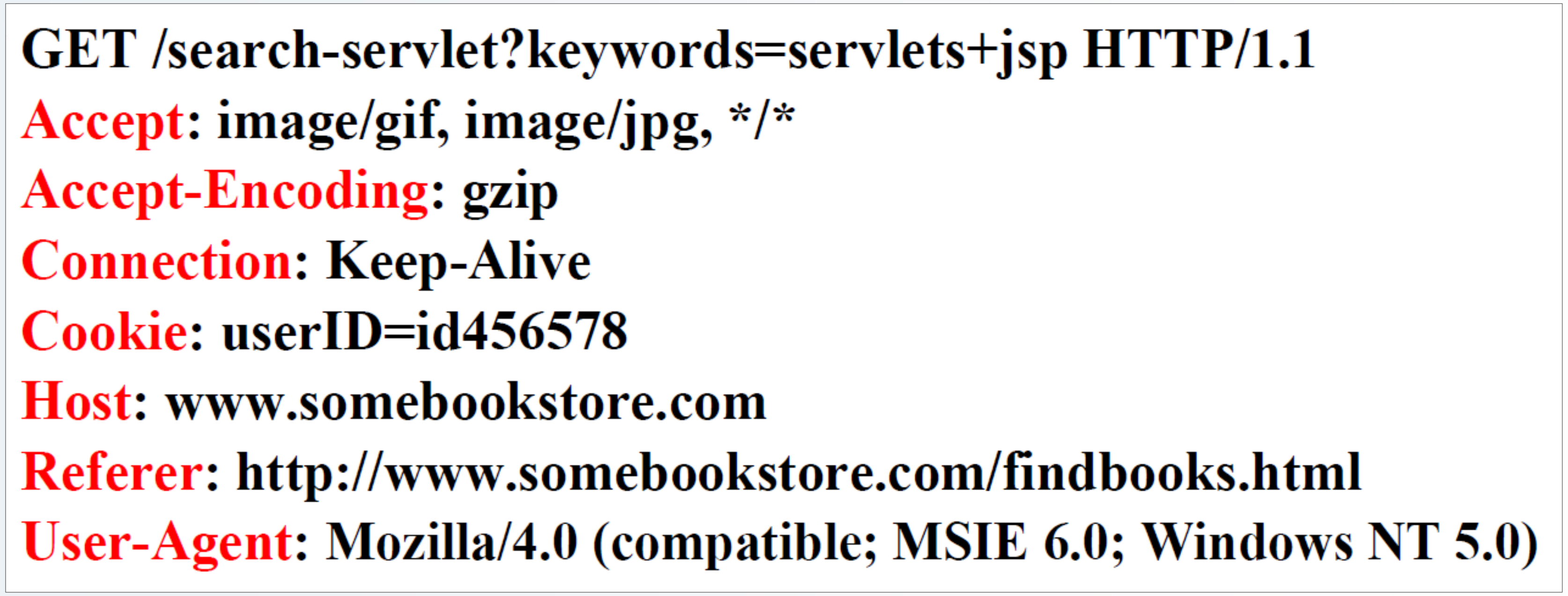
Lecture Agenda
Applied

1. Reading HTTP Request Headers.
2. Building a table of all the request headers.
3. Understanding the various request headers.
4. Reducing download times by compressing pages.
5. Differentiating among types of browsers.

A Typical HTTP Request

A Typical Request Header

Analysis of the Request Header



```
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)
```

- To excel in Java Web Development, a developer needs to understand HTTP to be effective with servlets and JSPs.
- A developer needs to be aware that implicit data (header information) is passed with each HTT Request.
- When needed, a developer should know how to access header information.

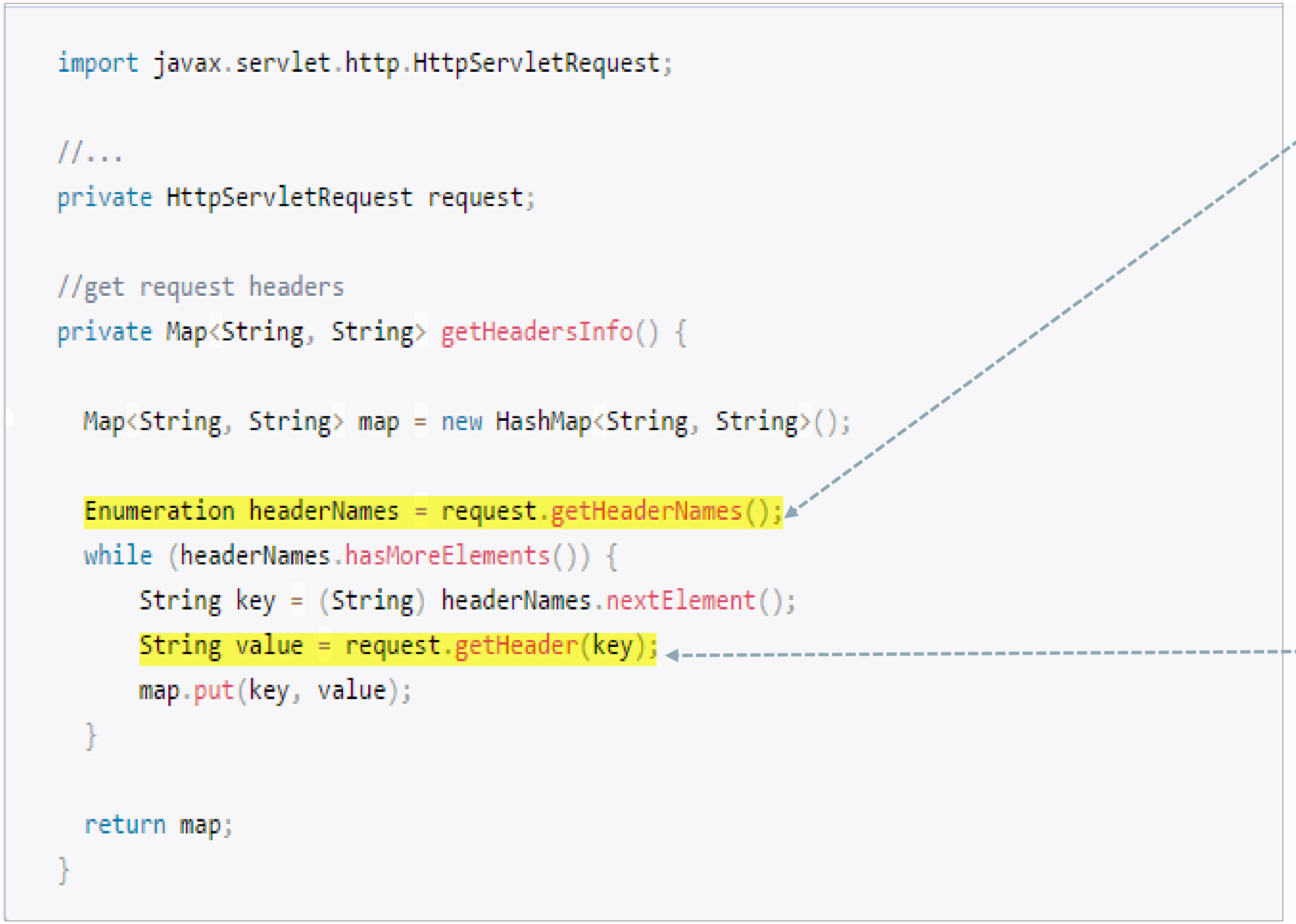
Reading Request Headers

Methods in HttpServletRequest

- General
  - getHeader() (header name is not case sensitive)
    - returns the value of the specified request header element as a string.
  - getHeaders()
    - returns all the values of the specified request header as a Enumeration (collection) of String objects.
  - getHeaderNames()
    - Returns an Enumeration of all the header names this request contains.

Reading Request Headers

Example 1: Loop through request header names



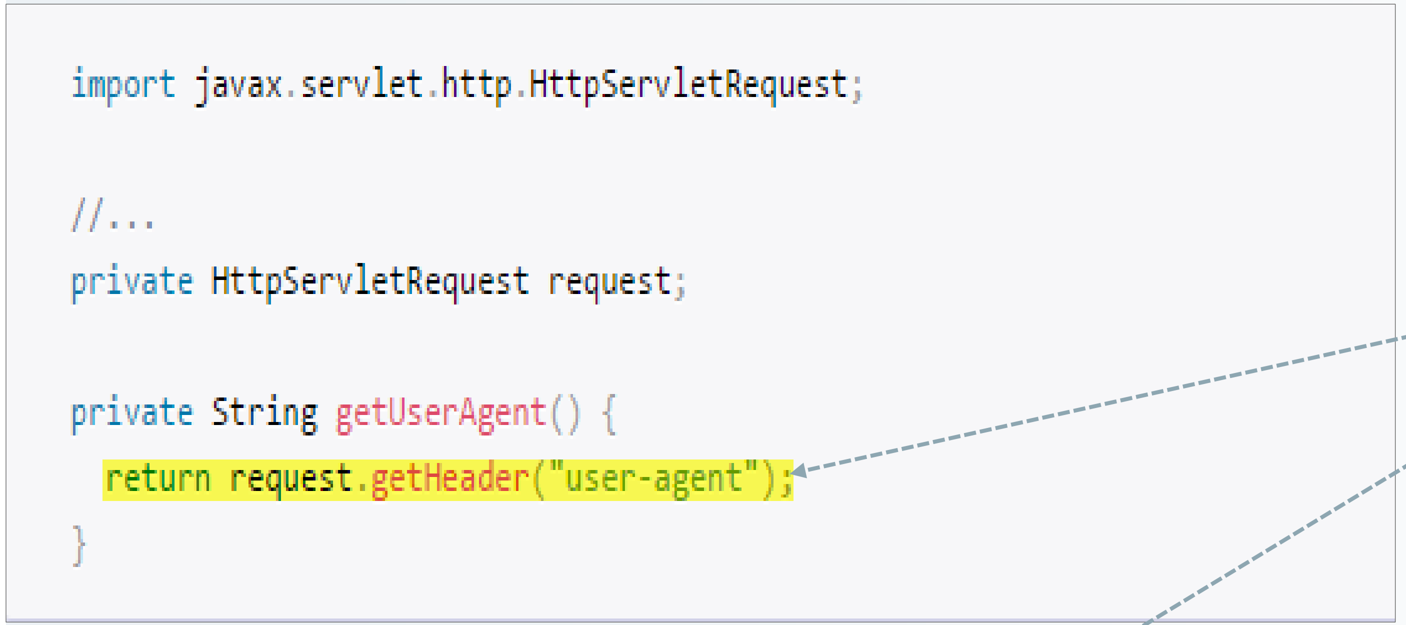
```
import javax.servlet.http.HttpServletRequest;

//...
private HttpServletRequest request;

//get request headers
private Map<String, String> getHeadersInfo() {

  Map<String, String> map = new HashMap<String, String>();

  Enumeration headerNames = request.getHeaderNames();
  while (headerNames.hasMoreElements()) {
      String key = (String) headerNames.nextElement();
      String value = request.getHeader(key);
      map.put(key, value);
  }

  return map;
}
```

All request header names are returned in the Enumeration.

Reading single value for header name.

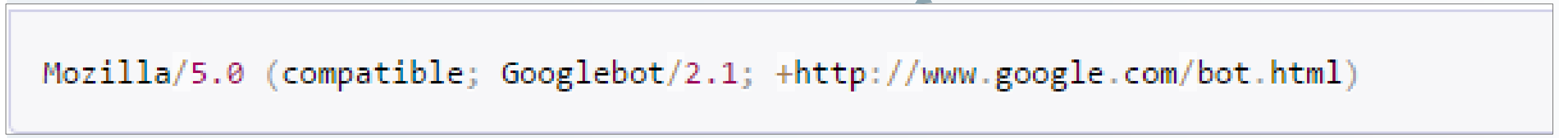
Reading Request Headers

Example 2: Get the “user-agent” header only



```
import javax.servlet.http.HttpServletRequest;


//...
private HttpServletRequest request;


private String getUserAgent() {
  return request.getHeader("user-agent");
}
```

Retrieve header value
for header “user-agent”
(client software agent)



```
Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)
```

Reading Request Headers

Methods in HttpServletRequest

- Specialized
  - getCookies()
    - Contains an array containg all the cookie objects.
  - getAuthType()
    - Returns the name of the authentication scheme used to protect the servlet (BASIC, SSL, Digest, Kerberos …).
  - getRemoteUser()
    - Returns the login of the user making the request if the user has been authenticated, null otherwise.
  - getContentLength()
    - Return length (in bytes) of the request body, or -1 if the length is not known.
  - getContentType()
    - Returns the MIME type (text/plain, application/pdf etc …)of the body of the request.
  - getDateHeader()
    - Returns value of specified date header of request

Reading Request Headers

Methods in HttpServletRequest

- Other Useful Methods
  - getMethod()
    - Returns the name of the HTTP method with which the request was made (for example: GET, POST, PUT …).
  - getRequestURI()
    - Returns URI path associated with the request (ex: /seach-servlet).
  - getQueryString()
    - Returns the query string that is contained in the request URL after the path.
  - getProtocol()
    - Returns the name and version of the protocol the request uses in the form (ex: HTTP/1.1)

Validate Missing Headers

HTTP 1.0 vs. HTTP 1.1

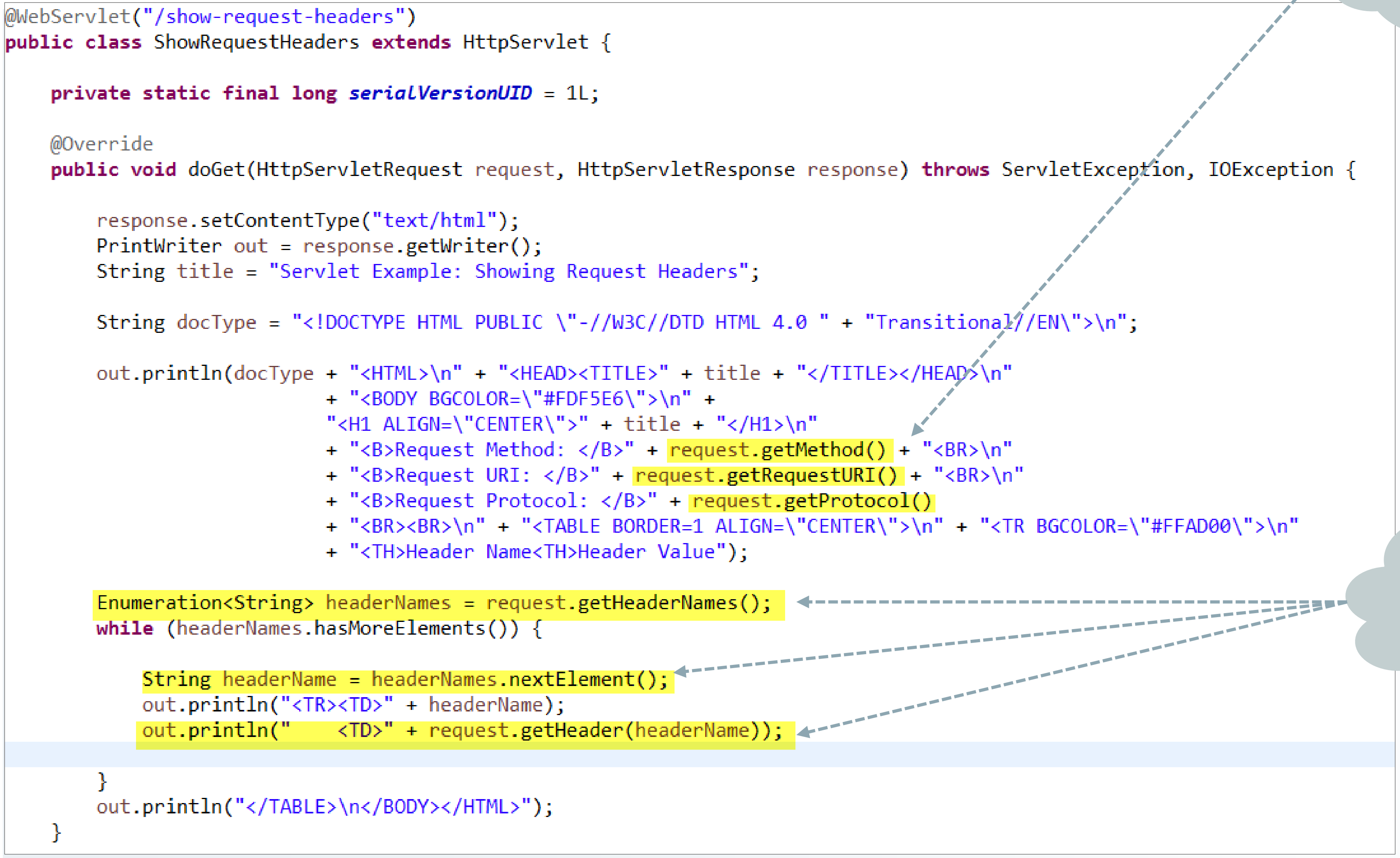
- HTTP 1.0
  - All request headers are optional
- HTTP 1.1
  - Only ***Host*** is required
- Conclusion
  - Always check that request.getHeader() is non-null before trying to use it.

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

Check if null request header before using.

Table of Request Headers

Construct a table of all Request Headers



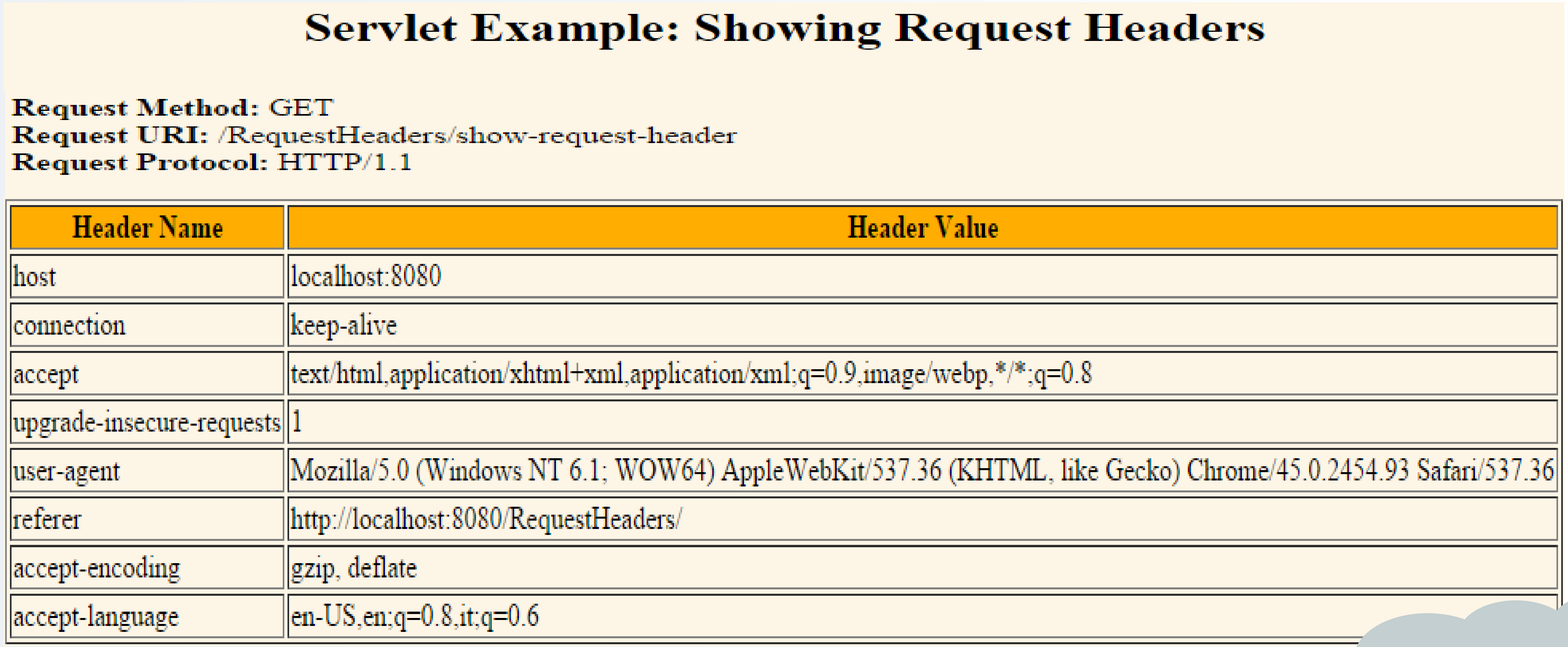
```
@WebServlet("/show-request-headers")
public class ShowRequestHeaders extends HttpServlet {

    private static final long serialVersionUID = 1L;

    @Override
    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";

        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" +
                            "<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" + "<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>");
    }
```

Read request headers (**HTTP Method, Request URI** and **Protocol/Version**)

Read all request headers, iterate and display.

Table of Request Headers

Result



Servlet Example: Showing Request Headers

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

Header Name	Header Value
host	localhost:8080
connection	keep-alive
accept	text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
upgrade-insecure-requests	1
user-agent	Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/45.0.2454.93 Safari/537.36
referer	http://localhost:8080/RequestHeaders/
accept-encoding	gzip, deflate
accept-language	en-US,en;q=0.8,it;q=0.6

Chrome Browser

Common HTTP 1.1 Headers

Header	Description

Accept	• Indicates MIME types (text/plain, text/html etc..) a browser can handle.
	• Can send different content to different clients.
	• For example PNG files have good compression, but not widely browser supported. A developer could check this:
	
	IF PNG supported THEN
	    send <img src="picture.png">
	ELSE send <img src="picture.gif">

Accept-Encoding	Indicates encoding (ex. gzip or compressed) browser can handle.
Authorization	• User identification for password-protected pages. • Instead of HTTP authorization, use HTML forms to send username/password and store information in session object. • Please note servers generally have a high-level way of setting up password-protected pages without any explicit programming in a servlet.

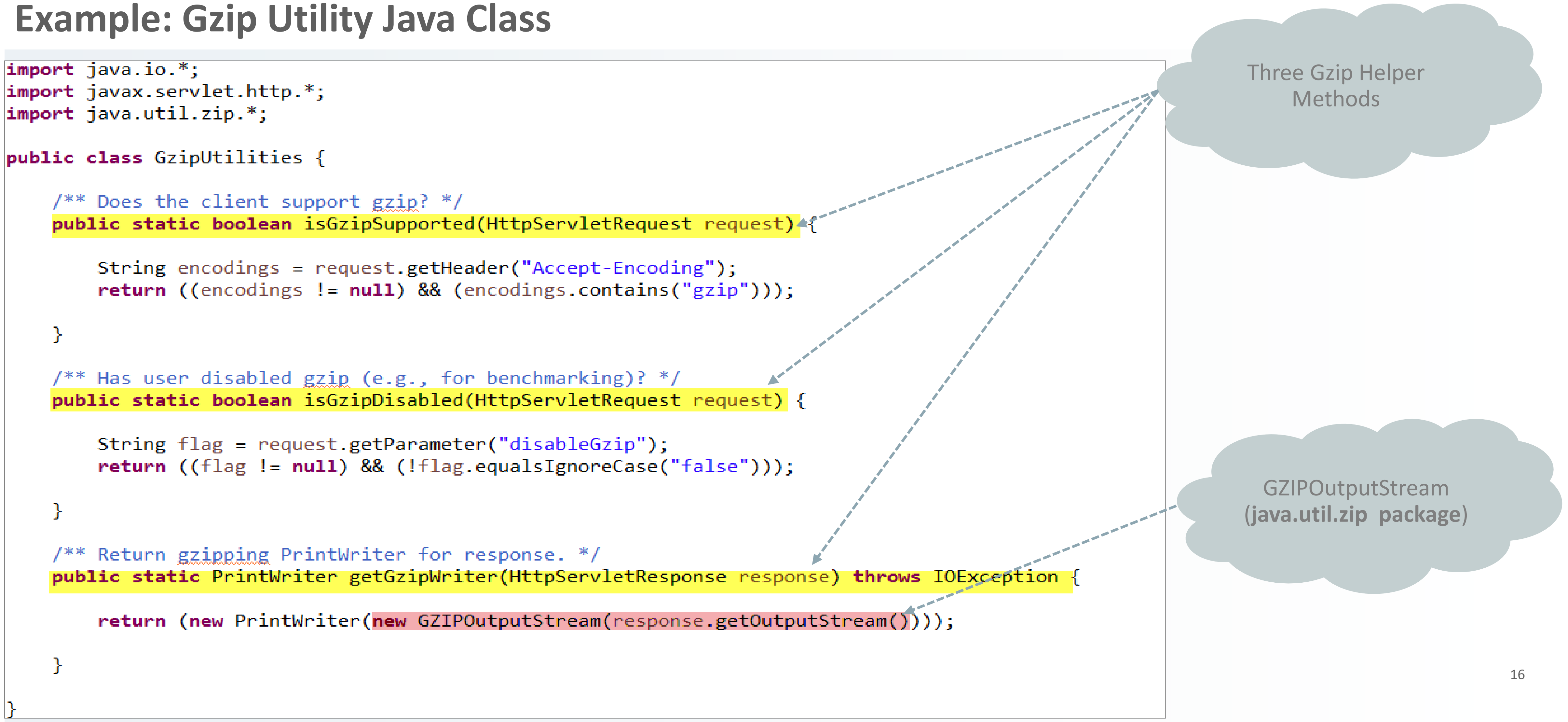
The Directories and Files for a Web Application

Directory	Description
Referer	• URL of referring web page • Useful for tracking traffic, logged by many servers.
User-Agent	• Client software agent identifying itself. • Best used for determining *category* of client (ex: browser, iphone etc ..)
Host	• Indicates host given in original URL • This is a required header in HTTP 1.1. This is important should you ever desire to write a custom HTTP client.
Connection	• In HTTP 1.0, keep-alive means browser can handle persistent connection. • In HTTP 1.1, persistent connection is default. • Persistent connections, means that the server can reuse the same socket over again for requests very close togeather.
Cookie	• Gives cookies previously sent to client. Use getCookies() **not** getHeader()

Sending Compressed Web Pages

Sending compressed Web Pages

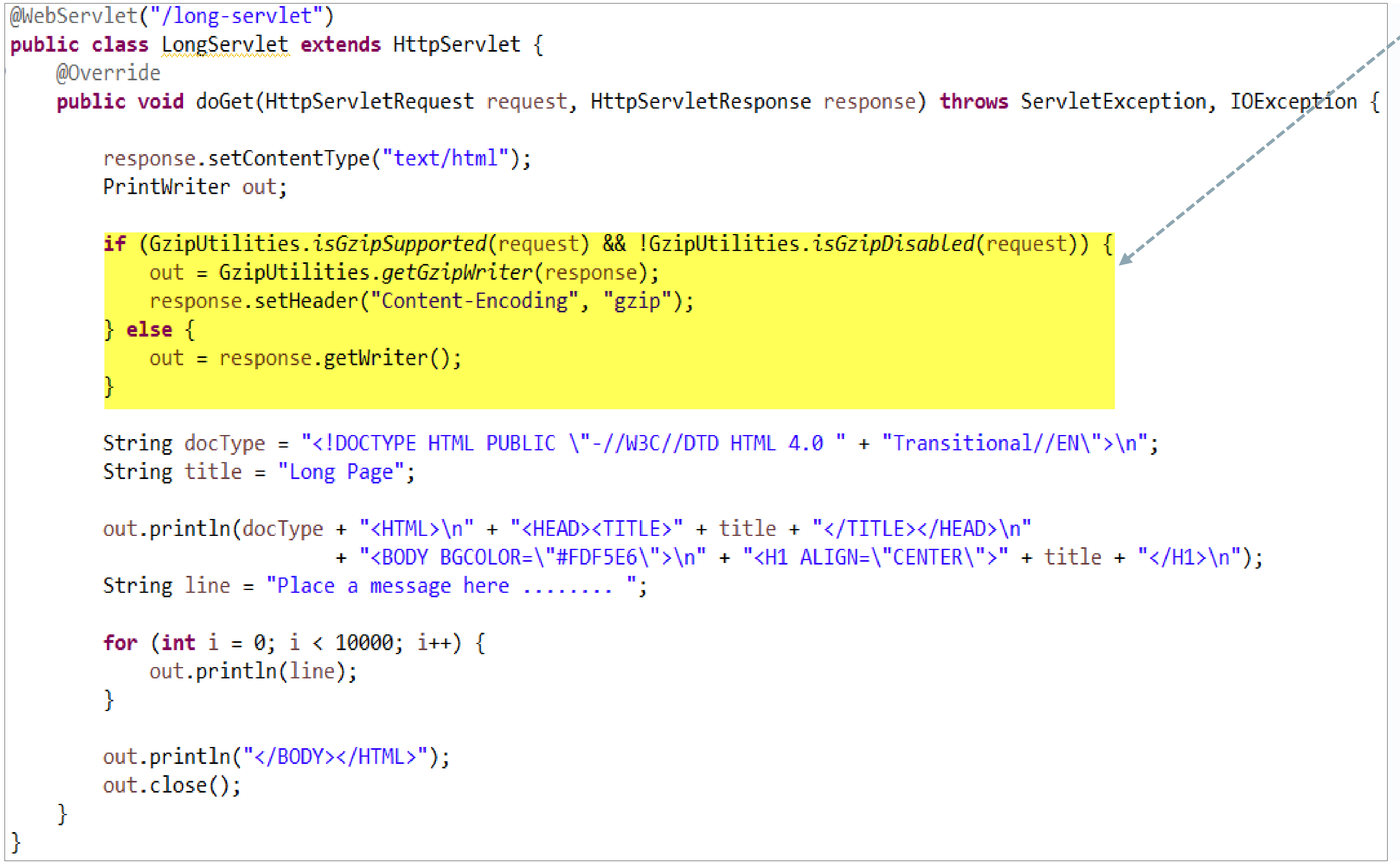
Example: Gzip Utility Java Class


Three Gzip Helper Methods
GZIPOutputStream (**java.util.zip package**)

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

public class GzipUtilities {

    /** Does the client support gzip? */
    public static boolean isGzipSupported(HttpServletRequest request) {

        String encodings = request.getHeader("Accept-Encoding");
        return ((encodings != null) && (encodings.contains("gzip")));

    }

    /** Has user disabled gzip (e.g., for benchmarking)? */
    public static boolean isGzipDisabled(HttpServletRequest request) {

        String flag = request.getParameter("disableGzip");
        return ((flag != null) && (!flag.equalsIgnoreCase("false")));

    }

    /** Return gzipping PrintWriter for response. */
    public static PrintWriter getGzipWriter(HttpServletResponse response) throws IOException {

        return (new PrintWriter(new GZIPOutputStream(response.getOutputStream())));

    }
}
```

Sending compressed Web Pages

Example: Using Gzip Utility Java Class



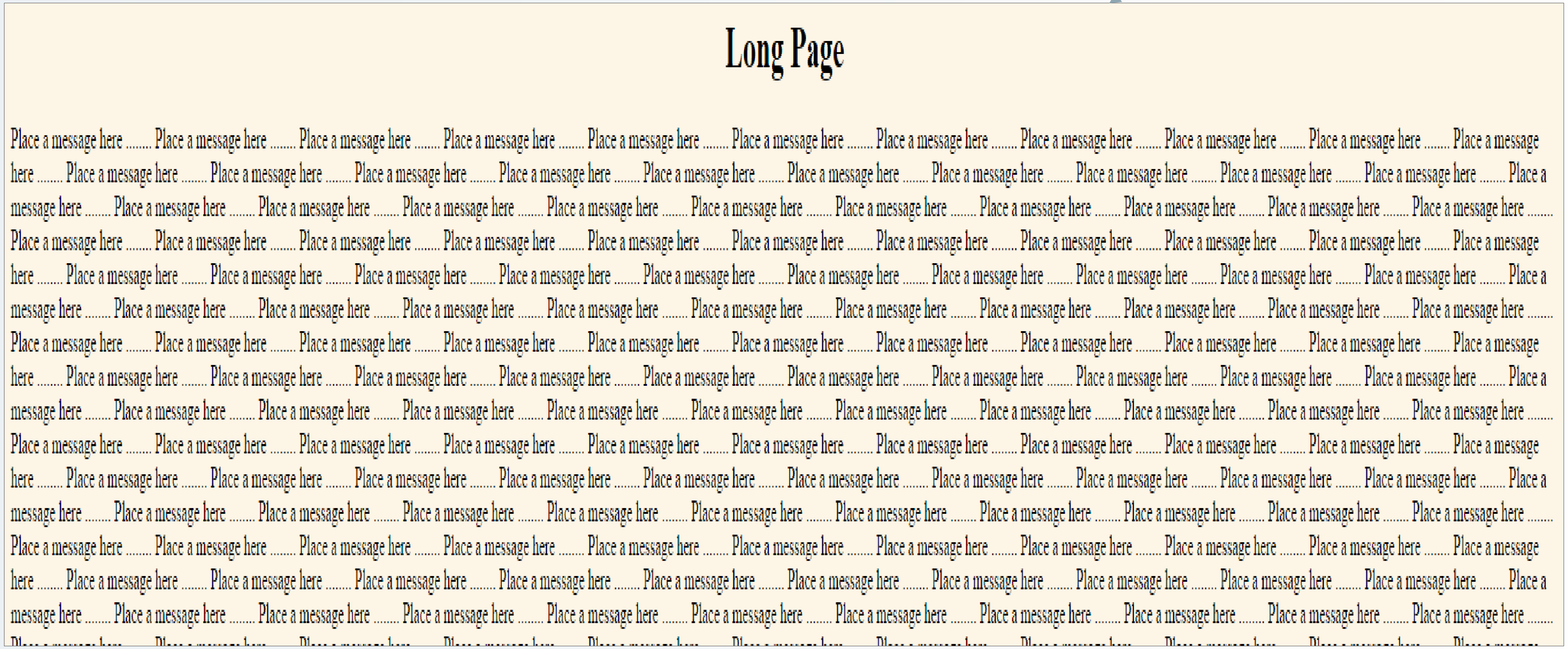
```java
@WebServlet("/long-servlet")
public class LongServlet extends HttpServlet {
    @Override
    public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

        response.setContentType("text/html");
        PrintWriter out;

        if (GzipUtilities.isGzipSupported(request) && !GzipUtilities.isGzipDisabled(request)) {
            out = GzipUtilities.getGzipWriter(response);
            response.setHeader("Content-Encoding", "gzip");
        } else {
            out = response.getWriter();
        }

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

        out.println(docType + "<HTML>\n" + "<HEAD><TITLE>" + title + "</TITLE></HEAD>\n"
                    + "<BODY BGCOLOR=\"#FDF5E6\">\n" + "<H1 ALIGN=\"CENTER\">" + title + "</H1>\n");
        String line = "Place a message here ........ ";

        for (int i = 0; i < 10000; i++) {
            out.println(line);
        }

        out.println("</BODY></HTML>");
        out.close();
    }
}
```

Returns a gzip print writer if request supported

Sending compressed Web Pages

Result

Generally at least 10% faster



Long Page

Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message
here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a
message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here ....
Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message
here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a
message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here ....
Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message
here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a
message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here ....
Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message
here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a
message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here ....
Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message
here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a
message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here .... Place a message here ....

Web Browsers

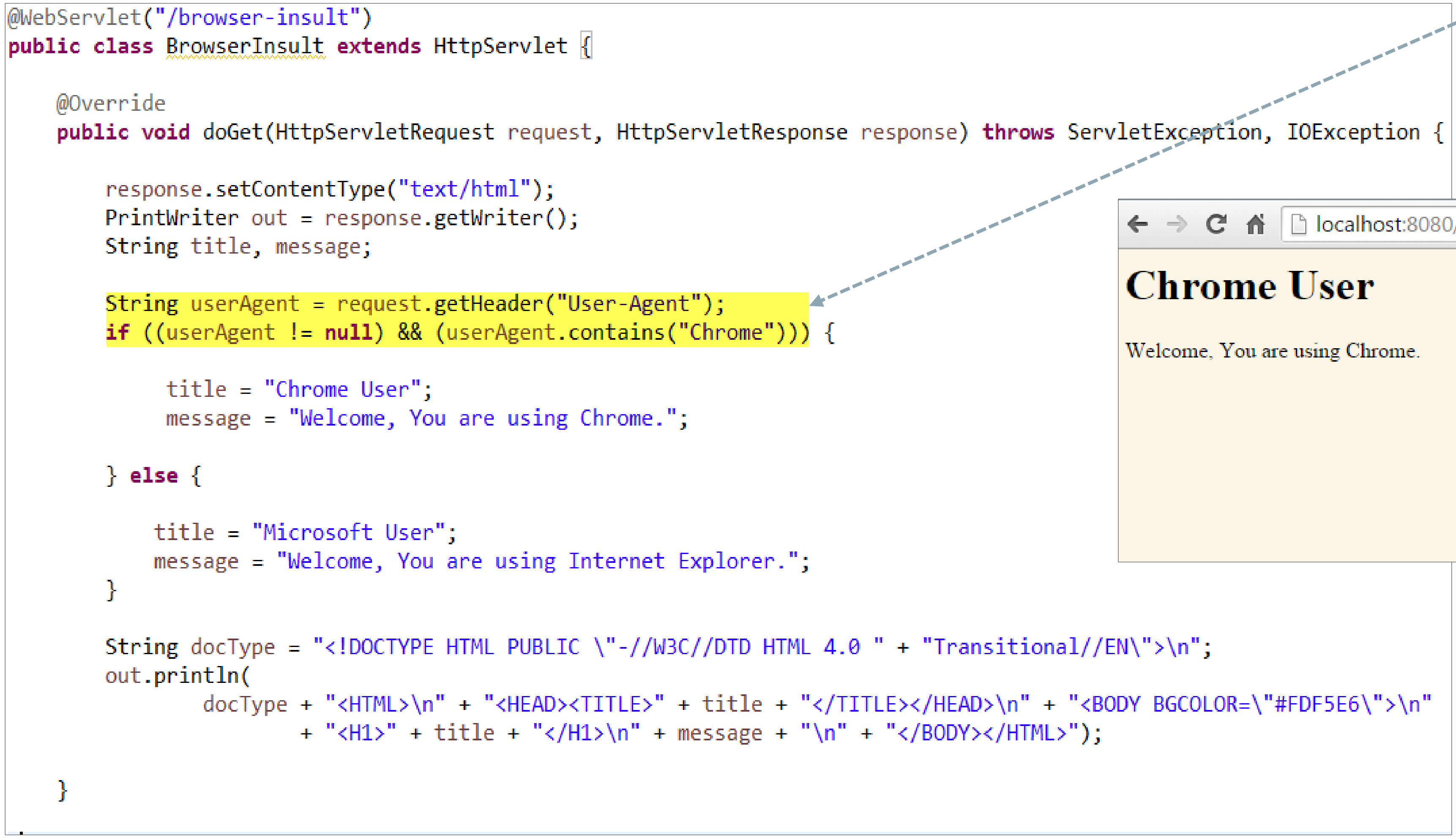
Differentiating Among Different Browser Types

- Use User-Agent only when necessary
  - Otherwise you code can be difficult-to-maintain code that consists of tables of browser versions and associated capabilities.
- Check for null
  - The header is **<u>not</u>** required by HTTP 1.1 specification, some, browser let you disable it.
- Differentiating Among Clients
  - To differentiate among browsers (FireFox, Internet Explorer, Chrome, Safari) check UserAgent
    - userAgent.contains("Chrome")   //Chrome
    - userAgent.contains("Firefox")   //FireFox
    - userAgent.contains("MSIE")   //Internet Explorer
    - (userAgent.contains("rv")   //Internet Explorer 11
    - (userAgent.contains(“Safari")   // Safari
- Header can be faked
  - If a client fakes this header, the servlet cannot tell the difference.

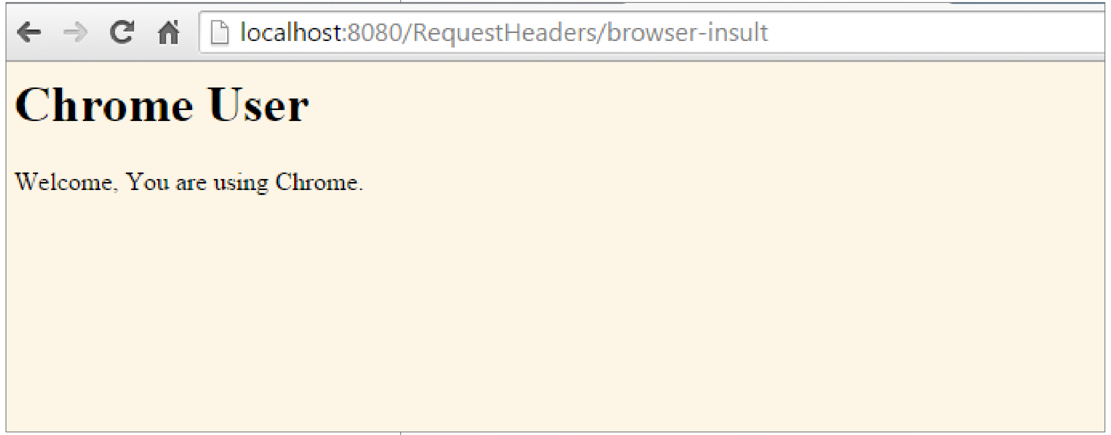
The Remaining HTTP methods

HTTP methods beside GET and POST

Validate "User-Agent" to determine browser type.



```
@WebServlet("/browser-insult")
public class BrowserInsult extends HttpServlet {

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

        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        String title, message;

        String userAgent = request.getHeader("User-Agent");
        if ((userAgent != null) && (userAgent.contains("Chrome"))) {

             title = "Chrome User";
             message = "Welcome, You are using Chrome.";

        } else {

            title = "Microsoft User";
            message = "Welcome, You are using Internet Explorer.";
        }

        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"
                        + "<H1>" + title + "</H1>\n" + message + "\n" + "</BODY></HTML>");

    }
```



localhost:8080/RequestHeaders/browser-insult

Chrome User

Welcome, You are using Chrome.