JSP - CLIENT REQUEST

When a browser requests for a web page, it sends lot of information to the web server which can not be read directly because this information travel as a part of header of HTTP request. You can check HTTP Protocol for more information on this.

Following is the important header information which comes from browser side and you would use very frequently in web programming:

Header	Description
Accept	This header specifies the MIME types that the browser or other clients can handle. Values of **image/png** or **image/jpeg** are the two most common possibilities.
Accept-Charset	This header specifies the character sets the browser can use to display the information. For example ISO-8859-1.
Accept-Encoding	This header specifies the types of encodings that the browser knows how to handle. Values of **gzip** or **compress** are the two most common possibilities.
Accept-Language	This header specifies the client's preferred languages in case the servlet can produce results in more than one language. For example en, en-us, ru, etc.
Authorization	This header is used by clients to identify themselves when accessing password-protected Web pages.
Connection	This header indicates whether the client can handle persistent HTTP connections. Persistent connections permit the client or other browser to retrieve multiple files with a single request. A value of **Keep-Alive** means that persistent connections should be used
Content-Length	This header is applicable only to POST requests and gives the size of the POST data in bytes.
Cookie	This header returns cookies to servers that previously sent them to the browser.
Host	This header specifies the host and port as given in the original URL.
If-Modified-Since	This header indicates that the client wants the page only if it has been changed after the specified date. The server sends a code, 304 which means **Not Modified** header if no newer result is available.
If-Unmodified-Since	This header is the reverse of If-Modified-Since; it specifies that the operation should succeed only if the document is older than the specified date.
Referer	This header indicates the URL of the referring Web page. For example, if you are at Web page 1 and click on a link to Web page 2, the URL of Web page 1 is included in the Referer header when the browser requests Web page 2.
User-Agent	This header identifies the browser or other client making the request and can be used to return different content to different types of browsers.

The HttpServletRequest Object:

The request object is an instance of a javax.servlet.http.HttpServletRequest object. Each time a client requests a page the JSP engine creates a new object to represent that request.

The request object provides methods to get HTTP header information including form data, cookies, HTTP methods etc.

There are following important methods which can be used to read HTTP header in your JSP program. These method are available with *HttpServletRequest* object which represents client request to webserver.

S.N.	Method & Description
1	**Cookie[] getCookies** Returns an array containing all of the Cookie objects the client sent with this request.
2	**Enumeration getAttributeNames** Returns an Enumeration containing the names of the attributes available to this request.
3	**Enumeration getHeaderNames** Returns an enumeration of all the header names this request contains.
4	**Enumeration getParameterNames** Returns an Enumeration of String objects containing the names of the parameters contained in this request.
5	**HttpSession getSession** Returns the current session associated with this request, or if the request does not have a session, creates one.
6	**HttpSession getSession*booleancreate*** Returns the current HttpSession associated with this request or, if if there is no current session and create is true, returns a new session.
7	**Locale getLocale** Returns the preferred Locale that the client will accept content in, based on the Accept-Language header
8	**Object getAttribute*Stringname*** Returns the value of the named attribute as an Object, or null if no attribute of the given name exists.

9

ServletInputStream getInputStream

Retrieves the body of the request as binary data using a ServletInputStream.

10

String getAuthType

Returns the name of the authentication scheme used to protect the servlet, for example, "BASIC" or "SSL," or null if the JSP was not protected

11

String getCharacterEncoding

Returns the name of the character encoding used in the body of this request.

12

String getContentType

Returns the MIME type of the body of the request, or null if the type is not known.

13

String getContextPath

Returns the portion of the request URI that indicates the context of the request.

14

String getHeader*Stringname*

Returns the value of the specified request header as a String.

15

String getMethod

Returns the name of the HTTP method with which this request was made, for example, GET, POST, or PUT.

16

String getParameter*Stringname*

Returns the value of a request parameter as a String, or null if the parameter does not exist.

17

String getPathInfo

Returns any extra path information associated with the URL the client sent when it made this request.

18

String getProtocol

Returns the name and version of the protocol the request.

19

String getQueryString

Returns the query string that is contained in the request URL after the path.

20

String getRemoteAddr

Returns the Internet Protocol *IP* address of the client that sent the request.

21

String getRemoteHost

Returns the fully qualified name of the client that sent the request.

22

String getRemoteUser

Returns the login of the user making this request, if the user has been authenticated, or null if the user has not been authenticated.

23

String getRequestURI

Returns the part of this request's URL from the protocol name up to the query string in the first line of the HTTP request.

24

String getRequestedSessionId

Returns the session ID specified by the client.

25

String getServletPath

Returns the part of this request's URL that calls the JSP.

26

String[] getParameterValues*Stringname*

Returns an array of String objects containing all of the values the given request parameter has, or null if the parameter does not exist.

27

boolean isSecure

Returns a boolean indicating whether this request was made using a secure channel, such as HTTPS.

28

int getContentLength

Returns the length, in bytes, of the request body and made available by the input stream, or -1 if the length is not known.

29

int getIntHeader*Stringname*

Returns the value of the specified request header as an int.

30

int getServerPort

Returns the port number on which this request was received.

HTTP Header Request Example:

Following is the example which uses **getHeaderNames** method of HttpServletRequest to read the HTTP header infromation. This method returns an Enumeration that contains the header information associated with the current HTTP request.

Once we have an Enumeration, we can loop down the Enumeration in the standard manner, using *hasMoreElements* method to determine when to stop and using *nextElement* method to get each parameter name.

```
<%@ page import="java.io.*,java.util.*" %>
<html>
<head>
<title>HTTP Header Request Example</title>
</head>
<body>
<center>
<h2>HTTP Header Request Example</h2>
<table width="100%" border="1" align="center">
<tr bgcolor="#949494">
<th>Header Name</th><th>Header Value(s)</th>
</tr>
<%
   Enumeration headerNames = request.getHeaderNames();
   while(headerNames.hasMoreElements()) {
      String paramName = (String)headerNames.nextElement();
      out.print("<tr><td>" + paramName + "</td>\n");
      String paramValue = request.getHeader(paramName);
      out.println("<td> " + paramValue + "</td></tr>\n");
   }
%>
</table>
</center>
</body>
</html>
```

Now put the above code in main.jsp and try to access it. This would produce result something as follows:

HTTP Header Request Example

Header Name	Header Values
accept	*/*
accept-language	en-us
user-agent	Mozilla/4.0 *compatible*; *MSIE*7.0; *WindowsNT*5.1; *Trident*/4.0; *InfoPath*.2; *MS* − *RTCLM*8
accept-encoding	gzip, deflate
host	localhost:8080
connection	Keep-Alive
cache-control	no-cache

To become more comfortable with other methods you can try few more above listed methods in the same fashion