getHeaderNames() getHeader() Example

❖ Retrieve Header Information with getHeaderNames() getHeader()

When a client clicks a hyper link or a submit button, we know that the data entered by the user in the Form fields is sent to server and is the default behavior of submit button. Ofcourse, right. But along with a lot of extra information goes to server on the name of headers (attached to request object) like what browser client is using, its supported languages etc. Servlet Programmer can know this extra information using getHeaderNames() getHeader() methods of HttpServletRequest interface.
The methods used are getHeaderNames() getHeader() defined in javax.servlet.http.HttpServletRequest interface.

➢ Let us see first what Servlet API says about these methods:

1. Enumeration getHeaderNames()
   Returns an enumeration of all the header names this request contains. If the request has no headers, this method returns an empty enumeration.
   Some servlet containers do not allow servlets to access headers using this method, in which case this method returns null.
2. String getHeader(String name)
   Returns the value of the specified request header as a String. If the request did not include a header of the specified name, this method returns null. If there are multiple headers with the same name, this method returns the first head in the request. The header name is case insensitive. You can use this method with any request header.

➢ Following code gives a very simple program of extracting header information from request object using getHeaderNames() getHeader().

Client HTML file: Headers.html

```
<body>

Would you like to see the <a  href="http://localhost:8888/india/head"> Headers</a> please?

</body>
```

web.xml entry for Headers.java servlet

```
<servlet>
  <servlet-name>snrao1</servlet-name>
  <servlet-class>Headers</servlet-class>
</servlet>

<servlet-mapping>
  <servlet-name>snrao1</servlet-name>
  <url-pattern>/head</url-pattern>
</servlet-mapping>
```

Server Servlet File: Headers.java

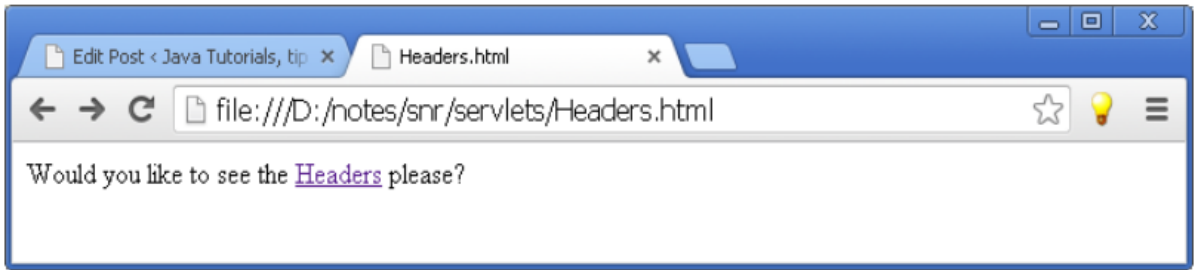
```
1  import javax.servlet.*;
2  import javax.servlet.http.* ;
3  import java.io.* ;
4  import java.util.*;
5  public class Headers extends HttpServlet
6  {
7    public void service( HttpServletRequest req, HttpServletResponse res ) throws ServletException, IOException
8    {
9      res.setContentType("text/html");
10     PrintWriter out = res.getWriter();
11
12     Enumeration e = req.getHeaderNames();      // gets all headers information
13
14     out.println("<H3>Following are the Headers coming from the Client<BR></H3>");
15
16     out.println("<table border=2 bordercolor=blue>");
17
18     out.println("<tr><th>Header Name</th><th>Header Value</th></tr>");
19
20      while(e.hasMoreElements())
21      {
22        String name = (String) e.nextElement();
23        String value = req.getHeader(name);      // gets each header information separately
24        out.println("<tr><th>"+name + "</th><th>" + value + "</th></tr>");
25      }
26      out.println("</table>");
27      out.close();
28    }
29 }
```

```
Enumeration e = req.getHeaderNames();
```

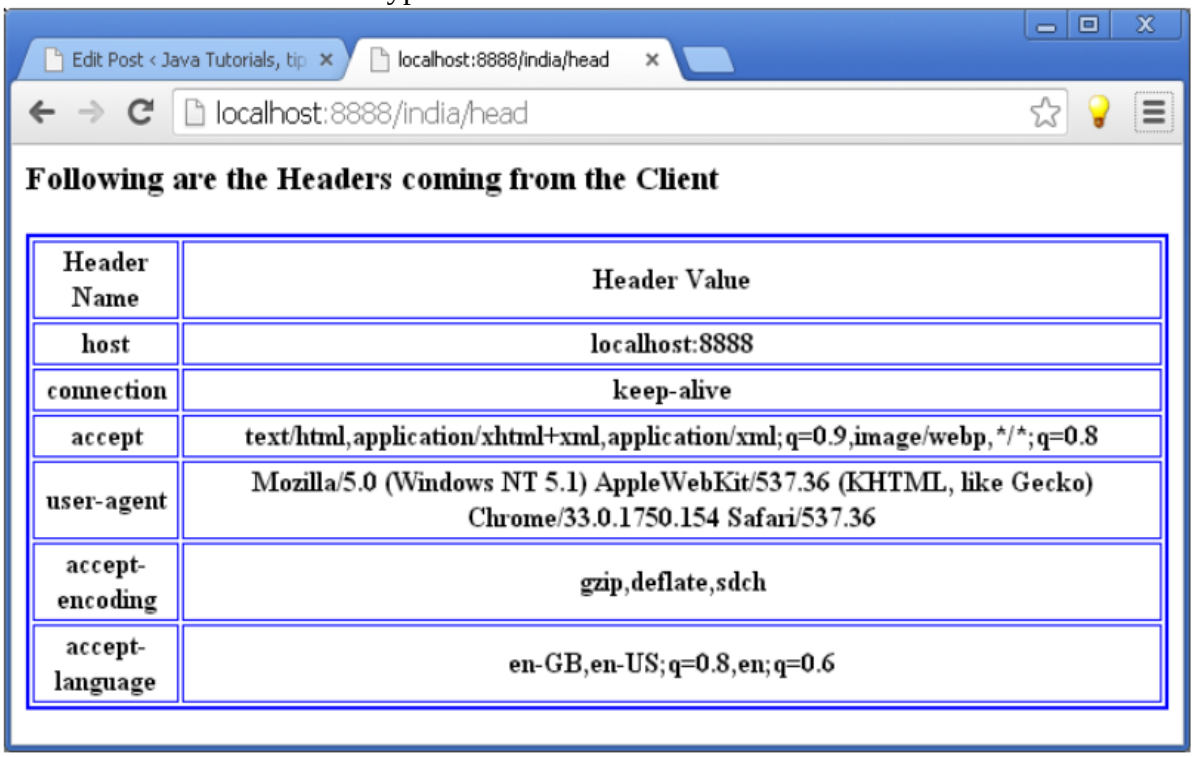
The **getHeaderNames()** method of HttpServletRequest returns an object of Enumeration. The Enumeration object **e** contains all the header names but not their associated values. In the code, value associated with each header is read and printed.

```
String name = (String) e.nextElement();
String value = req.getHeader(name);
```

The **getHeaderNames()** returns the header names in the form of strings. For this reason, the object returned by **nextElement()** is casted to String. Pass the header name "name" to **getHeader()** and the method returns the associated value as a string. In the while loop, with each iteration, one header name and its value are printed. Observe the output screen.



Client HTML Screen with a Hyperlink



In the above screenshot, the first column of table gives the header names and the second column gives the values associated with each header.

- **Following list gives the meaning of each header name.**

1. host
   This header gives the host (server) **name and port number** as written in the URL (in ACTION attribute).
2. connection
   It specifies whether the client's browser can handle the **persistent HTTP connections**. "keep-alive" indicates that the browser can handle persistent connections. **On a persistent connection, the browser can retrieve multiple files on a single request (or single connection)**.
3. accept
   This header informs the Server (or Servlet) what **MIME** types (like **text/html**, **image/gif** etc.) the browser supports (so that the Servlet can send response in the supported MIME type only).

4. user-agent
   Returns the **name of the browser and its version** etc. This information is used by the Servlet to send data specific to that browser. Suppose, the Servlet would like to send <MARQUEE> tag and if client is having Netscape Navigator browser (which does not support MARQUEE tag), it does not send and instead it may send the same information in <BLINK> tag supported by Netscape. Or a Servlet sends a response to the browser that requires an **ActiveX control** which the browser may not have. Microsoft IE version 4 does not come with **ActiveX** controls supported by version 5. For this reason, the Servlet requires the **user-agent** information.
5. accept-encoding
   This header tells the Servlet what type of **compression algorithms** are supported by the browser like **gzip** etc. so that Servlet sends an appropriate compressed file to the browser.
6. Accept-language
   This informs the Server that what **type of languages the browser support** like English, Spanish or Japanese etc., so that the Servlet may send response in other languages supported by the browser. Java and Servlets support **Internationalization**.