Handling HTTP Request Headers with Servlets

Understanding HyperText Transfer Protocol (HTTP) is essential for developing efficient Web application, increasing their performance and usability. HTTP information is sent to server application from Web browsers in the form of **HTTP request headers**. These HTTP request headers are distinct from data sent from user Web forms as a part of GET and POST requests. These headers are set by Web browsers and sent immediately after GET or POST request line. The following headers are sent to the server side Web applications: `Accept`, `Accept-Encoding`, `Connection`, `Cookie`, `Host`, `Referer`, and `User-Agent`.

All these headers can be extracted within servlets by calling **`getHeader`** method of **`HttpServletRequest`** with the name of the header, returning a string if the header is supplied or **`null`** otherwise. Some important general-purpose headers have special access methods in **`HttpServletRequest`**, which can be checked in API documentation.

https://tomcat.apache.org/tomcat-9.0-doc/servletapi/index.html

For more details about HTTP see RFC documents listed in the **Literature and Links** section.

The most important headers have specific methods for extracting them from HTTP requests. Some of these methods are in details described in API documentation (see Links section):

- **`getCookies`** method is used for getting header **`Cookie`**, with all cookies sent by the client through the request.
- **`getAuthType`** and **`getRemoteUser`** methods are used for getting **`Authorization`** data about authorization. **`getAuthType`** returns the name of the authentication scheme used to protect the servlet. **`getRemoteUser`** returns the login of the user making this request
- **`getContentLength`** method returns the value of the **`Content-Length`** header as an integer value. This is number of bytes in the request body.
- **`getHeaderNames`** can be used for getting the list of headers as an **`Enumeration`** of all header names received on this particular request.
- ... see API documentation for **Interface HttpServletRequest !**

HTTP request Headers

Access to request headers allow web applications (in our case servlets) to perform number of optimization and performance improvements, which are:

- Specification of MIME types that Web browser can handle can be used for providing response in several formats that can be accepted and viewed in the client side. This can be managed by using **`Accept`** header.
- Web application can check the char set acceptable by Web browsers by extracting **`Accept-Charset`** header from the request, which enable using allowed set of characters.
- Web application can detect client's preferred language by detecting the header **`Accept-Language`**. The values are standard language code, such as **de**, **da**, **us**, **sr**, etc. Language codes are specified in ISO standard (see links).
- For identification of clients that use passwords for accessing protected Web pages is used header **`Authorization`**.
- Characteristics of HTTP connection, like persistence, can be checked with the header **`Connection`**.
- Handling **cookies** the server previously sent to clients can be done with the header **`Cookie`**. This is not standard part of HTTP, but its extension.
- Handling of the host and the port used for providing request URL can be done with the header **`Host`**. This is especially important because providers use virtual hosting (one computer with Web server that hosts several Web sites).
- Identification of the modification of Web pages is important for managing several Web sites, which is possible with the header **`If-Modified-Since`** (**`If-Unmodified-Since`**).
- Identification of the URL from which a request originates can be done with the header **`Referer`**. This is useful for tracking the use of web sites, from where the traffic comes (who refers the web site). This header can be also manipulated by clients, so the use should be careful.
- Identification of a user browser can be done with header **`User-Agent`**, which is useful for distinguishing among group of clients (users).

Checking Web browsers

The header **`User-Agent`** can be used for identifying client Web browser, but the following issues should be checked:

- Web browser can be detected in header **`User-Agent`**, but it is important to get additional information about browsers, such as support for Java or **gzip** compression.
- Before handling browser specific actions it is advisable to check whether **`request.getHeader`** does not returned **`null`**.
- Be careful since some browsers allow users to fake this information.

The main request line

The main request line is usually rendered as a URL with the following typical structure:

`[Protocol]://[Host name][URI of software application]?[Query string]`

The parts of the main request line are:

- **Method**. The main request method can be extracted by using the **`getMethod`** method, which will return either GET or POST in the most of the cases.
- **Request URI**. This is the part of the URL following the host information, but before data section of the URL. This can be extracted by using the method **`getRequestURI`**. For example, for the URL **`http://localhost/handlingforms/GetPostFormsServlet?firstName=Miki&lastName=Maus`** the URI part is the name of the servlet **`GetPostFormsServlet`**, while the host part is **`http://localhost/handlingforms/`**.
- **Query string**. The form data are extracted as a query string by using the method **`getQueryString`**. For example, for the URL **`http://localhost/handlingforms/GetPostFormsServlet?firstName=Miki&lastName=Maus`** the query string is the part following the question mark (?): **`firstName=Miki&lastName=Maus`**.
- **Protocol**. The information about the protocol can be extracted by using the method **`getProtocol`**, which can be done before specifying the response. Typical value is HTTP/1.1.

Literature and Links

[1] **The Apache Tomcat**. http://tomcat.apache.org/

[2] Marty Hall and Larry Brown. *Core Servlets and JavaServer Pages, Free Online Version of Second Edition*. http://pdf.coreservlets.com/. Chapter 5: Handling the Client Request: HTTP Request Headers.

[3] https://tomcat.apache.org/tomcat-9.0-doc/servletapi/overview-summary.html

[4] https://tomcat.apache.org/tomcat-9.0-doc/servletapi/index.html.

[5] **Interface HttpServletRequest**.
https://docs.oracle.com/javaee/6/api/javax/servlet/http/HttpServletRequest.html

[6**] RFC 7230. Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing**.
https://tools.ietf.org/html/rfc7230.

[7] **RFC 7231. Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content**.
https://tools.ietf.org/html/rfc7231.

[8] See also RFCs 7232, 7233, 7234 and 7235.

[9] **Language codes**. https://www.iso.org/iso-639-language-codes.html.