Message in HTTP

HTTP is built on the basis of the Client-Server structure model and Stateless Request / Response protocol, which is governed by the exchange of messages (Message) along a TCP / IP connection.

HTTP is built on the basis of the Client-Server structure model and Stateless Request / Response protocol, which is governed by the exchange of messages (Message) along a TCP / IP connection.

A Client is a program (a browser or any Client) that establishes a connection to a Server for the purpose of sending one or more HTTP request messages. An HTTP "Server" is a program (generally referred to as a Web Server such as Apache Web Server or Internet Information Services - IIS .) that accepts connections to serve HTTP requests by sending counter messages. HTTP recovery.

HTTP uses the URI to identify a given source and to establish a connection. Once a connection is established, **HTTP Messages** are transmitted in a format similar to that used in Internet Mail [RFC5322] and MIME (Multipurpose Internet Mail Extensions) [RFC2045]. These messages include **Requests** from the Client to the Server and **Feedbacks** from the Server to the Client, which will follow the following format:

```
HTTP - message  =  Request > |  Response > ;  HTTP / 1.1  messages
```

HTTP requests and HTTP responses use a common message format of RFC 822 for required data spread. This general notification format includes 4 sections:

1. A first line
2. No or more Header fields followed by CRLF.
3. A blank line (for example, a line with nothing before CRLF), only the end of the Header field.
4. An arbitrary message body

In the following sections, we will explain each item used in the HTTP message.

Start-line (start-line)

The first line will have the following syntax:

```
start-line = Request-Line |  Status-Line
```

We will discuss Request-Line and Status-Line while discussing the HTTP Request and HTTP Response messages accordingly. Now, we look at some examples of starting lines in case of requests and responses.

```
GET /hello.jsp HTTP / 1.1 (This is Request-Line sent by the client)
```

```
HTTP / 1.1 200 OK (This is Status-Line sent by the server)
```

Header fields

Header fields provide required information about requests or responses, or about objects sent in the notification body. There are four types of Headers in HTTP messages:

General-Header : These Header fields have general application capabilities for both request and response notifications.

Request Type (Request-Header) : These Header fields are only applicable to required notifications.

Response Type (Response-Header) : These Header fields are only applicable for response messages.

Entity-Header : These fields define the body-entity information or, if no body is present, about the source identified by the request.

All Headers are mentioned above in a general format and each Header field includes a name followed by a colon (:) and the field value as follows:

```
message-header = field-name ":" [field-value]
```

Below is an example of the various Header fields:

```
User-Agent: curl / 7.16.3 libcurl / 7.16.3 OpenSSL / 0.9.7l zlib / 1.2.3
Host: www.example.com
Accept-Language: en, mi
Date: Mon, 27 Jul 2009 12:28:53 GMT
Server: Apache
Last-Modified: Wed, 22 Jul 2009 19:15:56 GMT
ETag: "34aa387-d-1568eb00"
Accept-Ranges: bytes
Content-Length: 51
Vary: Accept-Encoding
Content-Type: text / plain
```

Notification body

The message body is optional for an HTTP message but if it is available, then it is used to carry the body associated with the request or response. If the entity body is linked, then usually Content-Type and Content-Length lines determine the nature of the linked body.

A notification body is the part that carries the actual HTTP request data (including sample data and uploaded, etc.) and HTTP response data from the Server (including files, images, etc.). Below is the simple content of a notification body:

```
Hello, World!
```

In the next two chapters we will discuss how to use the concepts explained to prepare HTTP Requests and HTTP Responses.

According to Tutorialspoint

Previous post: Parameters in HTTP

Next lesson: Request (Request) in HTTP