



WEB TECHNOLOGIES 1

HTTP REQUEST AND RESPONSE

WHAT IS HTTP?

- HTTP stands for Hyper Text Transfer Protocol.
- This is a basis for data communication in the internet. The data communication starts with a request sent from a client and ends with the response received from a web server.
- A website URL starting with “http://” is entered in a web browser from a computer (client). The browser can be a Chrome, Firefox, Edge, Safari, Opera or anything else.
- Browser sends a request sent to the web server that hosts the website.
- The web server then returns a response as a HTML page or any other document format to the browser.
- Browser displays the response from the server to the user.



WHAT IS HTTP?



HTTP

- The operation of Hypertext Transfer Protocol (HTTP) involves the communication between a Hypertext Transfer Protocol (HTTP) client application (Usually web browser) and a Hypertext Transfer Protocol (HTTP) server application (Web servers).
- Hypertext Transfer Protocol (HTTP) uses Transmission Control Protocol (TCP) as the Transport Layer Protocol at Well Known port number 80.
- Once the TCP connection is established, the two steps in Hypertext Transfer Protocol (HTTP) communication are
 - HTTP Client Request
 - HTTP Server Response




HTTP

- HTTP Client Request:

- Hypertext Transfer Protocol (HTTP) client sends an Hypertext Transfer Protocol (HTTP) Request to the Hypertext Transfer Protocol (HTTP) Server according to the HTTP standard, specifying the information the client like to retrieve from the Hypertext Transfer Protocol (HTTP) Server.

- HTTP Server Response:

- Once the Hypertext Transfer Protocol (HTTP) Request arrived at the Hypertext Transfer Protocol (HTTP) server, it will process the request and creates an Hypertext Transfer Protocol (HTTP) Response message.
 - The Hypertext Transfer Protocol (HTTP) response message may contain the resource the Hypertext Transfer Protocol (HTTP) Client requested or information why the Hypertext Transfer Protocol (HTTP) request failed.
- 

HTTP

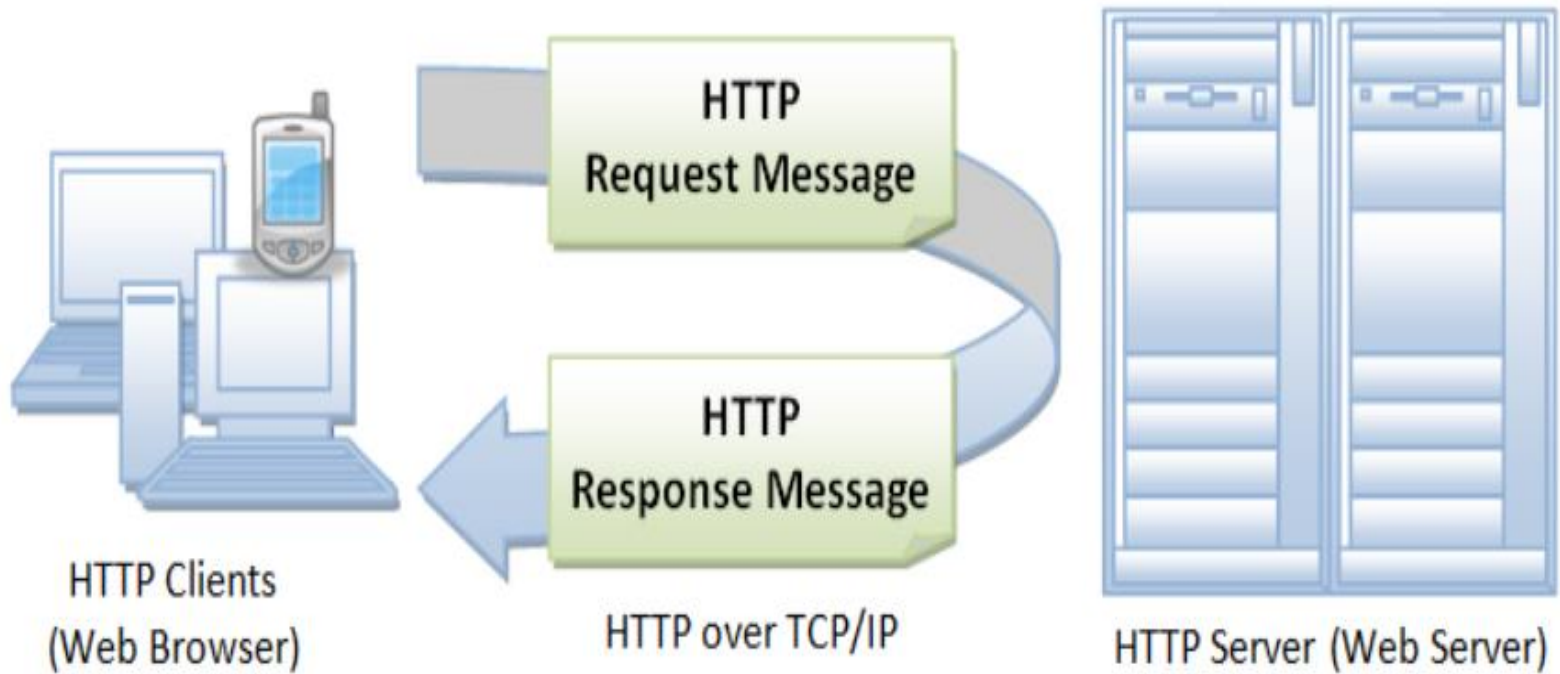
- Multiple Host Name Support:

HTTP/1.1 supports specifying a Hostname in header. This allows running multiple websites using a single IP address helps preventing the depletion of IPv4 addresses. Any combination of IP address, Port number and Hostname can be used to identify a website.

- Persistent Connections:

- In HTTP/1.1 a keep-alive-mechanism was introduced, where a TCP connection could be reused for more than one request. The default working principle of HTTP was not changed and the difference is that the TCP connection is kept open after each HTTP Request/HTTP Response pair.
- Persistent connections reduce delay remarkably, because the client does not need to re-negotiate the TCP connection if it want to retrieve any resource immediate.

HTTP



CLIENT REQUEST DATA

- When a user submits a browser request to a web server, it sends two categories of data:
 - *Form Data*: Data that the user explicitly typed into an HTML form.
 - For example: registration information.
 - *HTTP Request Header Data*: Data that is automatically appended to the HTTP Request from the client.
 - For example: cookies, browser type, etc,



WHAT IS HTTPS?

- HTTPS is the secured HTTP protocol required to send and receive information securely over internet.
- Nowadays it is mandatory for all websites to have HTTPS protocol to have secured internet.
- Browsers like Google Chrome will show an alert with “Not Secure” message in the address bar if the site is not served over HTTPS.
- Besides the security and encryption, the communication structure of HTTPS protocol remains same as HTTP protocol as explained above.
- It is highly recommended not to use confidential information like credit card details on HTTP sites. Ensure the financial transactions happens through HTTPS protocol.

HEADER FIELDS

- HTTP header fields provide required information about the request or response, or about the object sent in the message body. There are four types of HTTP message headers:

- **General-header:**

These header fields have general applicability for both request and response messages.

- **Request-header:**

These header fields have applicability only for request messages.

- **Response-header:**

These header fields have applicability only for response messages.

- **Entity-header:**

These header fields define meta information about the entity-body or, if no body is present, about the resource identified by the request.



HEADER FIELDS

- All the above mentioned headers follow the same generic format and each of the header field consists of a name followed by a colon (:) and the field value as follows:

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

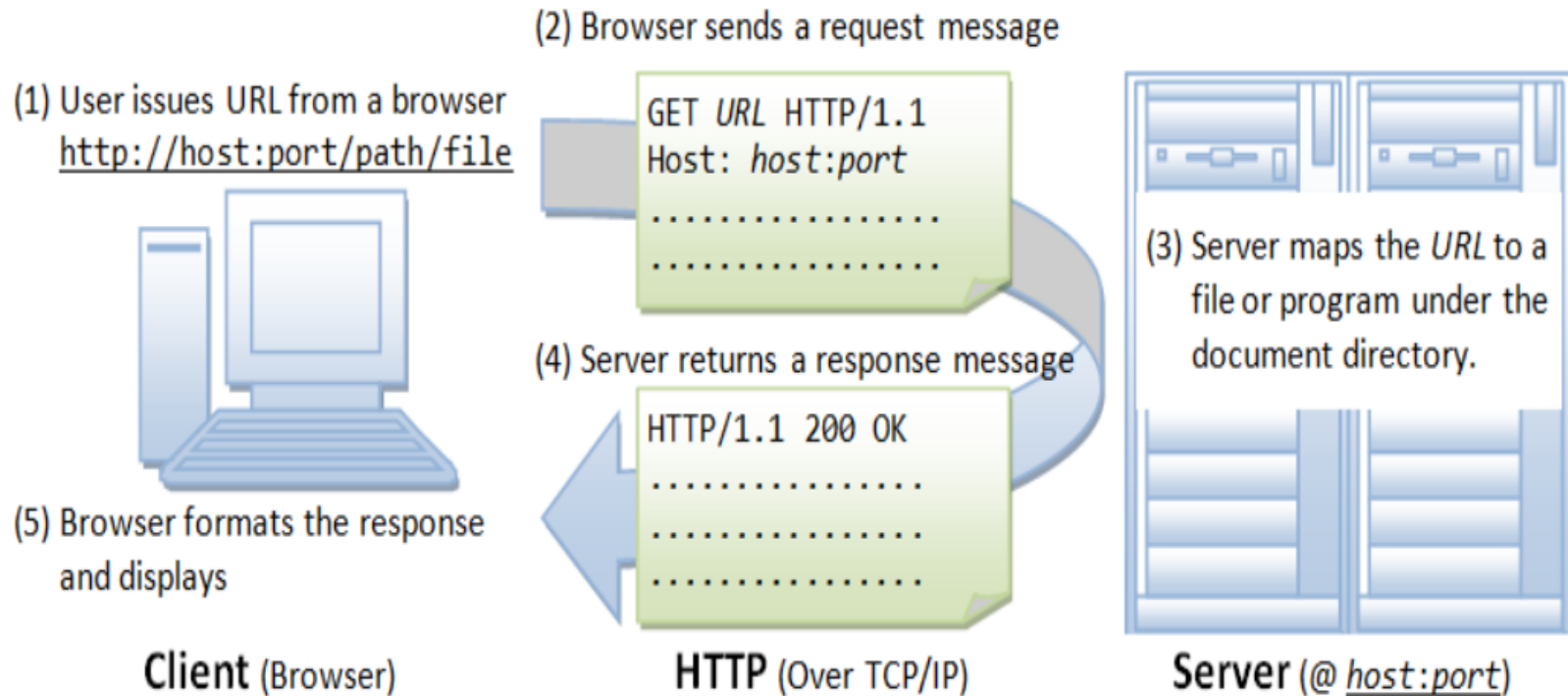
```
User-Agent: curl/7.16.3 libcurl/7.16.3 OpenSSL/0.9.7l zlib/1.2.3
Host: www.example.com
Accept-Language: en, mi
Date: Mon, 27 Jul 2009 12:28:53 GMT
Server: Apache
Last-Modified: Wed, 22 Jul 2009 19:15:56 GMT
ETag: "34aa387-d-1568eb00"
Accept-Ranges: bytes
Content-Length: 51
Vary: Accept-Encoding
Content-Type: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



HOW CLIENTS AND SERVERS USE URLS

- Example URL: **http://www.gmail.com:80/index.html**
- Clients use *prefix* (http://www.gmail.com:80) to infer:
 - What kind of server to contact (Web server)
 - Where the server is (www.gmail.com)
 - What port it is listening on (80)
- Servers use *suffix* (/index.html) to:
 - Determine if request is for static or dynamic content.
 - No hard and fast rules for this.
 - Convention: executables reside in cgi-bin directory
 - Find file on file system.
 - Initial “/” in suffix denotes home directory for requested content.
 - Minimal suffix is “/”, which all servers expand to some default home page (e.g., index.html).

HOW CLIENTS AND SERVERS USE URLS



UNIFORM RESOURCE LOCATOR

- A URL (Uniform Resource Locator) is used to uniquely identify a resource over the web.

protocol://hostname:port/path-and-file-name

There are 4 parts in a URL:

- ❑ **Protocol:** The application-level protocol used by the client and server, e.g., HTTP, FTP, and telnet.
- ❑ **Hostname:** The DNS domain name (e.g., www.nowhere123.com) or IP address (e.g., 192.128.1.2) of the server.
- ❑ **Port:** The TCP port number that the server is listening for incoming requests from the clients.
- ❑ **Path-and-file-name:** The name and location of the requested resource, under the server document base directory.



UNIFORM RESOURCE LOCATOR

- For Example, In the URL : **http://www.nowhere123.com/docs/index.html**
- The communication protocol is HTTP;
- The hostname is **www.nowhere123.com**.
- The port number was not specified in the URL, and takes on the default number, which is **TCP port 80** for HTTP.
- The path and file name for the resource to be located is **"/docs/index.html"**.
- *URLs for static content:* **http://www.cs.cmu.edu:80/index.html**
 - Identifies a file called **index.html**, managed by a Web server at **www.cs.cmu.edu** that is listening on port **80**.
- *URLs for dynamic content:* **http://www.cs.cmu.edu:8000/cgi-bin/adder?15000&213**
 - Identifies an executable file called **adder**, managed by a Web server at **www.cs.cmu.edu** that is listening on port **8000**, that should be called with **two** argument strings: **15000** and **213**.

URL NOTATION

- Read a URL and print its parts
- Requirements:

Element	Required/Optional
Protocol	Required
Host	Required
Port	Required (default value for http - 80 , for https - 443)
Path	Required (default value: /)
Query Strings	Optional (multiple query strings are separated by &)
Fragment	Optional

```
https://softuni.bg:447/search?  
Query=pesho&Users=true#go
```



```
Protocol: https  
Host: softuni.bg  
Port: 447  
Path: /search  
Query: Query=pesho&Users=true  
Fragment: go
```


VALID AND INVALID URL

■ Some valid URLs:

```
http://www.google.bg/search?sourceid=navclient&ie=UTF-8&r1z=1T4GGLL_enBG369BG369&q=http+get+vs+post
```

```
http://bg.wikipedia.org/wiki/%D0%A1%D0%BE%D1%84%D1%82%D1%83%D0%B5%D1%80%D0%BD%D0%B0_%D0%B0%D0%BA%D0%B0%D0%B4%D0%B5%D0%BC%D0%B8%D1%8F
```

■ Some invalid URLs:

```
http://www.google.bg/search?&q=C# .NET 4.0
```

Should be:
C%23+.NET+4.0

```
http://www.google.bg/search?&q=бипа
```

Should be: %D0%B1
%D0%B8%D1%80%D0%B0

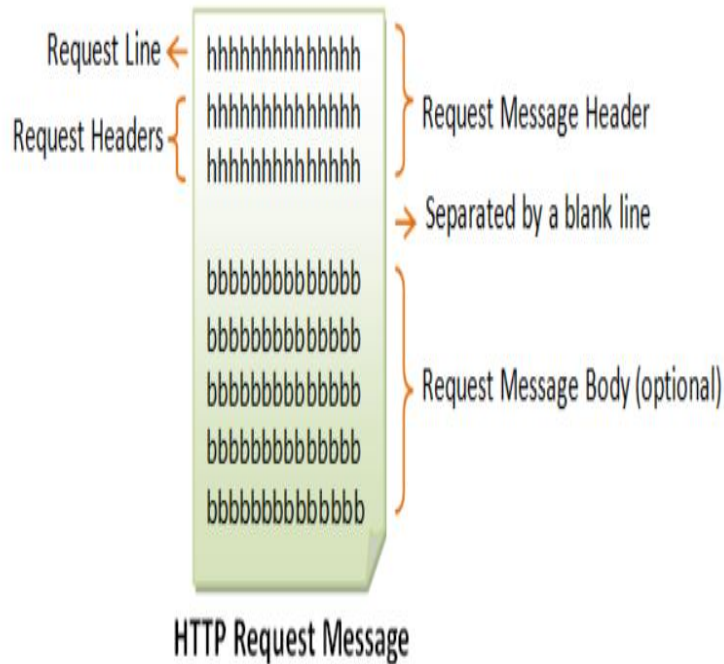


HTTP REQUESTS

- HTTP request is a *request line*, followed by zero or more *request headers*.
- Request headers: **<header name>: <header data>**
 - Provide additional information to the server.
- Request line: **<method> <uri> <version>**
 - <version> is HTTP version of request (HTTP/1.0 or HTTP/1.1)
 - <uri> is typically URL for proxies, URL suffix for servers.
 - <method> is either GET, POST, OPTIONS, HEAD, PUT, DELETE, or TRACE.
- Major differences between HTTP/1.1 and HTTP/1.0
 - HTTP/1.0 uses a new connection for each transaction.
 - HTTP/1.1 also supports *persistent connections*
 - multiple transactions over the same connection
 - Connection: Keep-Alive
 - HTTP/1.1 requires HOST header, HTTP/1.1 adds additional support for caching
 - Host: kittyhawk.cmcl.cs.cmu.edu



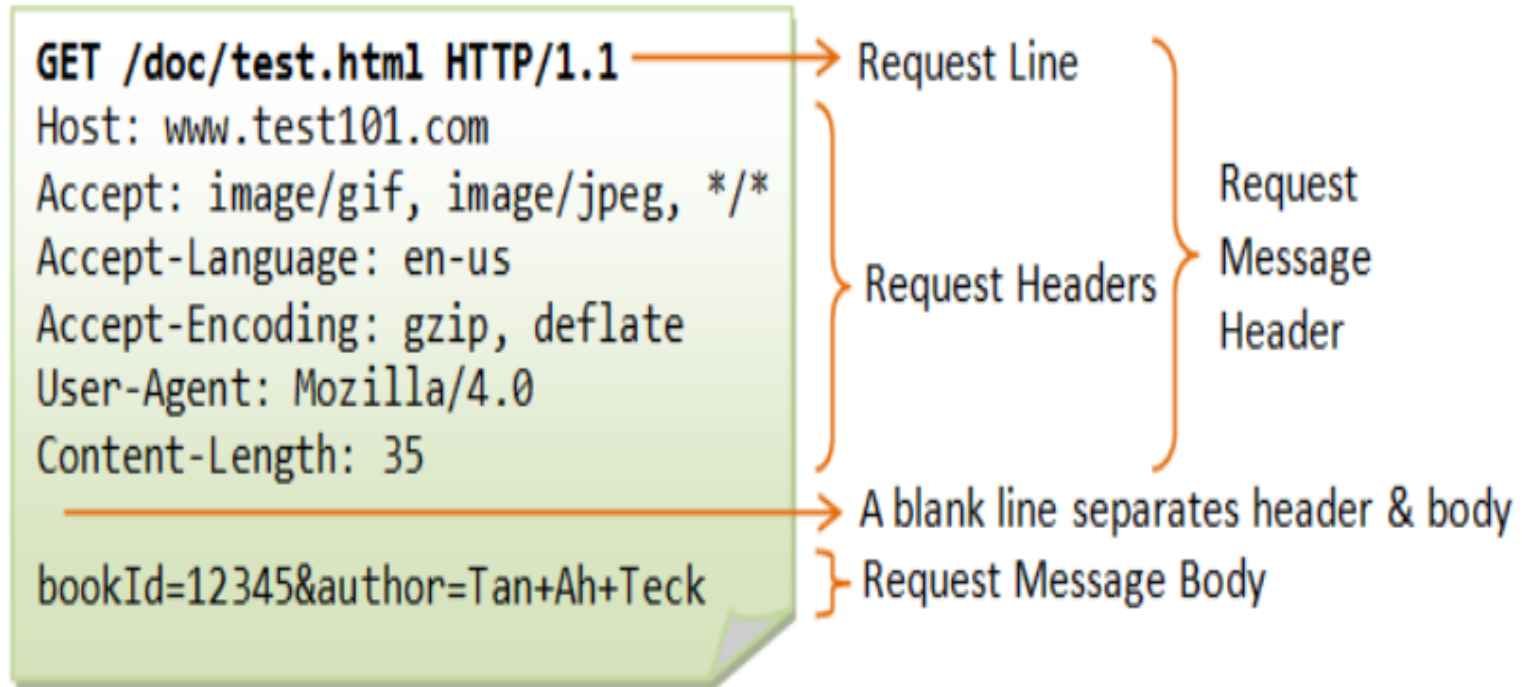
HTTP REQUEST MESSAGE



- The first line of the header is called the *request line*, followed by optional *request headers*.
- The request line has the following syntax:
request-method-name request-URI HTTP-version
- ***request-method-name***: HTTP protocol defines a set of request methods, e.g., GET, POST, HEAD, and OPTIONS. The client can use one of these methods to send a request to the server.
- ***request-URI***: specifies the resource requested.
- ***HTTP-version***: Two versions are currently in use: HTTP/1.0 and HTTP/1.1.



HTTP REQUEST MESSAGE



HTTP REQUESTS

- HTTP methods:
 - **GET:** Retrieve static or dynamic content
 - Arguments for dynamic content are in URI
 - Workhorse method (99% of requests)
 - **POST:** Retrieve dynamic content
 - Arguments for dynamic content are in the request body
 - **OPTIONS:** Get server or file attributes
 - **HEAD:** Like GET but no data in response body
 - **PUT:** Write a file to the server!
 - **DELETE:** Delete a file on the server!
 - **TRACE:** Echo request in response body
 - Useful for debugging.




HTTP REQUESTS

- For example, the browser translates the following URL into:

<http://www.nowhere123.com/doc/index.html>

```
GET /docs/index.html HTTP/1.1
Host: www.nowhere123.com
Accept: image/gif, image/jpeg, */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
(blank line)
```

- When this request message reaches the server, the server can take either one of these actions:
 - The server interprets the request received, maps the request into a file under the server's document directory, and returns the file requested to the client.
 - The server interprets the request received, maps the request into a program kept in the server, executes the program, and returns the output of the program to the client.
 - The request cannot be satisfied, the server returns an error message
- 

HTTP RESPONSE MESSAGE

- The browser receives the response message, interprets the message and displays the **contents of the message** on the browser's window according to the **media type** of the response (as in the Content-Type response header).
- Common media type include,
"text/plain", "text/html", "image/gif", "image/jpeg", "audio/mpeg", "video/mpeg",
"application/msword", and "application/pdf".



HTTP RESPONSE MESSAGE

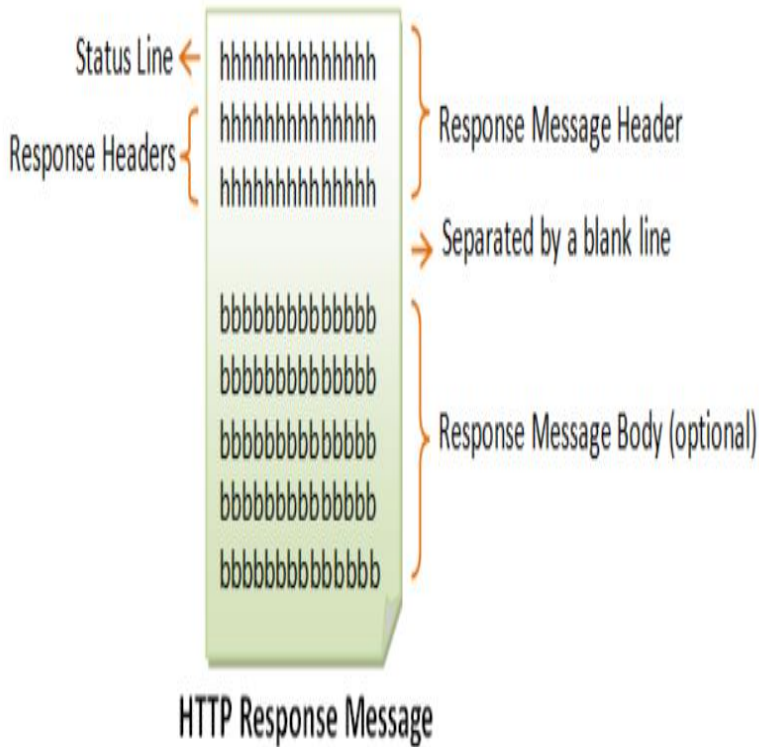
- HTTP response is a *response line* followed by zero or more *response headers*.
- Response line:

<version> <status code> <status msg>

- <version> is HTTP version of the response.
- <status code> is numeric status.
- <status msg> is corresponding English text.
 - 200 OK Request was handled without error
 - 403 Forbidden Server lacks permission to access file
 - 404 Not found Server couldn't find the file.
- Response headers: **<header name>: <header data>**
 - Provide additional information about response
 - Content-Type: MIME type of content in response body.
 - Content-Length: Length of content in response body.



HTTP RESPONSE MESSAGE



○ Format:

Status line

Response header fields

blank line

Response body

- Status line format:

- | HTTP version | status code | explanation |
|--------------|-------------|-----------------------|
| 1.0 | 200 | OK |
| 1.0 | 404 | Not Found |
| 1.0 | 500 | Internal Server Error |
| 1.1 | 200 | OK |
| 1.1 | 404 | Not Found |
| 1.1 | 500 | Internal Server Error |
| 2.0 | 200 | OK |
| 2.0 | 404 | Not Found |
| 2.0 | 500 | Internal Server Error |
| 3.0 | 200 | OK |
| 3.0 | 404 | Not Found |
| 3.0 | 500 | Internal Server Error |

- Example: HTTP/1.1 200 OK

(Current version is 1.1)

- The header field, Content-type, is required

HTTP RESPONSE MESSAGE

HTTP/1.1 200 OK

Date: Sun, 18 Oct 2009 08:56:53 GMT

Server: Apache/2.2.14 (Win32)

Last-Modified: Sat, 20 Nov 2004 07:16:26 GMT

ETag: "10000000565a5-2c-3e94b66c2e680"

Accept-Ranges: bytes

Content-Length: 44

Connection: close

Content-Type: text/html

X-Pad: avoid browser bug

<html><body><h1>It works!</h1></body></html>



GET REQUEST METHOD – EXAMPLE

```
<form method="get">  
  Name: <input type="text" name="name" />  
  Age: <input type="text" name="age" />  
  <input type="submit" />  
</form>
```

HTTP request line

GET /HTTP/1.1

Host: localhost

<CRLF>

HTTP request headers

The request body is empty



POST REQUEST METHOD – EXAMPLE

- The **POST** method transfers data in the HTTP body
- **POST** can send text and binary data e.g. upload files

```
POST /login HTTP/1.1
```

```
Host: localhost
```

```
Content-Length: 59
```

```
<CRLF>
```

```
username=mente&password=top*secret!
```

```
<CRLF>
```

HTTP request line

HTTP request headers

The request body holds
the submitted form data



GET REQUEST TO APACHE SERVE FROM BROWSER

```
GET /test.html HTTP/1.1
Accept: */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 4.01; Windows
98)
Host: euro.ecom.cmu.edu
Connection: Keep-Alive
CRLF (\r\n)
```



GET RESPONSE FROM APACHE SERVER

```
HTTP/1.1 200 OK
Date: Thu, 22 Jul 1999 04:02:15 GMT
Server: Apache/1.3.3 Ben-SSL/1.28 (Unix)
Last-Modified: Thu, 22 Jul 1999 03:33:21 GMT
ETag: "48bb2-4f-37969101"
Accept-Ranges: bytes
Content-Length: 79
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Content-Type: text/html
CRLF
<html>
<head><title>Test page</title></head>
<body>
<h1>Test page</h1>
</html>
```



DETAILS OF HTTP STATUS CODES

- The HTTP status codes are developed as per the Internet standards defined by Internet Engineering Task Force (IETF).
- They are classified into five different categories as below:
 - 1xx series – Informational Message
 - 2xx – Success Message
 - 3xx – Redirection Message
 - 4xx – Error Messages Related to Client
 - 5xx – Error Messages Related to Server




DETAILS OF HTTP STATUS CODES

Status Line

- The first line is called the *status line*, followed by optional response header(s).
- The status line has the following syntax:

HTTP-version status-code reason-phrase

- **HTTP-version:** The HTTP version used in this session. Either HTTP/1.0 and HTTP/1.1.
 - **status-code:** a 3-digit number generated by the server to reflect the outcome of the request.
 - **reason-phrase:** gives a short explanation to the status code.
 - **Common status code** and reason phrase are "200 OK", "404 Not Found", "403 Forbidden", "500 Internal Server Error".
- 

1XX: INFORMATION

Message:	Description:
100 Continue	The server has received the request headers, and the client should proceed to send the request body
101 Switching Protocols	The requester has asked the server to switch protocols
103 Checkpoint	Used in the resumable requests proposal to resume aborted PUT or POST requests



2XX-SUCCESSFUL

Message:	Description:
200 OK	The request is OK (this is the standard response for successful HTTP requests)
201 Created	The request has been fulfilled, and a new resource is created
202 Accepted	The request has been accepted for processing, but the processing has not been completed
203 Non-Authoritative Information	The request has been successfully processed, but is returning information that may be from another source
204 No Content	The request has been successfully processed, but is not returning any content
205 Reset Content	The request has been successfully processed, but is not returning any content, and requires that the requester reset the document view
206 Partial Content	The server is delivering only part of the resource due to a range header sent by the client

3XX-REDIRECTION

Message:	Description:
300 Multiple Choices	A link list. The user can select a link and go to that location. Maximum five addresses
301 Moved Permanently	The requested page has moved to a new URL
302 Found	The requested page has moved temporarily to a new URL
303 See Other	The requested page can be found under a different URL
304 Not Modified	Indicates the requested page has not been modified since last requested
306 Switch Proxy	<i>No longer used</i>
307 Temporary Redirect	The requested page has moved temporarily to a new URL
308 Resume Incomplete	Used in the resumable requests proposal to resume aborted PUT or POST requests



4XX-CLIENT ERROR

Message:	Description:
400 Bad Request	The request cannot be fulfilled due to bad syntax
401 Unauthorized	The request was a legal request, but the server is refusing to respond to it. For use when authentication is possible but has failed or not yet been provided
402 Payment Required	<i>Reserved for future use</i>
403 Forbidden	The request was a legal request, but the server is refusing to respond to it
404 Not Found	The requested page could not be found but may be available again in the future
405 Method Not Allowed	A request was made of a page using a request method not supported by that page
406 Not Acceptable	The server can only generate a response that is not accepted by the client



5XX-SERVER ERROR

Message:	Description:
500 Internal Server Error	A generic error message, given when no more specific message is suitable
501 Not Implemented	The server either does not recognize the request method, or it lacks the ability to fulfill the request
502 Bad Gateway	The server was acting as a gateway or proxy and received an invalid response from the upstream server
503 Service Unavailable	The server is currently unavailable (overloaded or down)
504 Gateway Timeout	The server was acting as a gateway or proxy and did not receive a timely response from the upstream server
505 HTTP Version Not Supported	The server does not support the HTTP protocol version used in the request
511 Network Authentication Required	The client needs to authenticate to gain network access



TCP/IP

- TCP/IP specifies how data is exchanged over the internet by providing end-to-end communications.
- It identify how it should be broken into packets, addressed, transmitted, routed and received at the destination.
- TCP/IP requires little central management, and it is designed to make networks reliable, with the ability to recover automatically from the failure of any device on the network.



TCP/IP

- The two main protocols in the internet protocol suite serve specific functions.
- TCP defines how applications can create channels of communication across a network.
- It also manages how a message is assembled into smaller packets before they are then transmitted over the internet and reassembled in the right order at the destination address.
- IP defines how to address and route each packet to make sure it reaches the right destination.
- Each gateway computer on the network checks this IP address to determine where to forward the message.

PORT 80

- Port 80 is one of the most commonly used port numbers in the Transmission Control Protocol (TCP) suite.
- Any Web/HTTP client, such as a Web browser, uses port 80 to send and receive requested Web pages from a HTTP server.
- It manages all HTTP-based requests that originate from a computer, regardless of the number of requests and initiating Web clients.
- Similarly, the HTTP server responds to all requests received at port 80.
- Alternatively, HTTP may use port 8080, rather than port 80, typically to deploy a caching or proxy server.

