HTTP
(Hypertext Transfer Protocol)

Author: Victor

The Hypertext Transfer Protocol (HTTP)

- is an application protocol for distributed, collaborative, hypermedia information systems. HTTP is the foundation of data communication for the World Wide Web.
- Hypertext is structured text that uses logical links (hyperlinks) between nodes containing text. HTTP is the protocol to exchange or transfer hypertext.
- The standards development of HTTP was coordinated by the Internet Engineering Task Force (IETF) and the World Wide Web Consortium (W3C), culminating in the publication of a series of Requests for Comments (RFCs).

http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol

A little history

- **HTTP/0.9**
  was proposed in March 1991.
- **HTTP/1.0**
  In May 1996 he was released the document RFC 1945, which served as the basis for the HTTP / 1.0.
- **HTTP/1.1**
  The current version, adopted in June 1999. TCP-connection can remain opened after sending a response to the request. The client now have to send information about the host name.
- **HTTP/2**
  February 11, 2015 published the final version of the blueprint the next version Protocol. Unlike previous versions, HTTP/2 is a binary protocol.

Request-Respons

Every requests at HTTP/1.1 consists of main two strings:
method, requested resource and protocol
(Request-Line = Method SP Request-URI SP HTTP-Version CRLF)
and Host of this resource (for example)
GET /pub/WWW/TheProject.html HTTP/1.1
Host: www.w3.org

- Every Response consists of main one string:
status-line which is information about Response
(Status-Line = HTTP-Version SP Status-Code SP Reason-Phrase CRLF)
(for example)
HTTP/1.1 200 OK

Software

All software to work with the HTTP protocol is divided into three broad categories:

- Servers as major suppliers of storage and data processing.
- Customers - end users of server services.
- Proxy to perform transport services.

Methods (part 1)

- **GET**
  method requests a representation of the specified resource.
- **POST**
  method requests that the server accept the entity enclosed in the request
- **OPTIONS**
  method returns the HTTP methods that the server supports for the specified URL.
- **HEAD**
  method asks for a response identical to that of a GET request, but without the response body.

http://tools.ietf.org/html/rfc7231#section-4

Methods (part 2)

- **PUT**
  method requests that the enclosed entity be stored under the supplied URI.
- **DELETE**
  method deletes the specified resource.
- **TRACE**
  method echoes the received request so that a client can see what (if any) changes or additions have been made by intermediate servers.
- **CONNECT**
  method converts the request connection to a transparent TCP/IP tunnel.
- **PATCH**
  method applies partial modifications to a resource.

HTTP status codes

- 1xx Informational
- 2xx Success
- 3xx Redirection
- 4xx Client Error
- 5xx Server Error
- Unofficial codes

https://en.wikipedia.org/wiki/List_of_HTTP_status_codes

http://tools.ietf.org/html/rfc7231#section-6

HTTP header fields

All header can be divided into four groups:

- General-header - used both the requests and the responses.
- Request Headers -allow the client to pass additional information about the request, and about the client itself.
- Response Headers - information about the response which cannot be placed in the Status-Line.
- Entity-header - define metainformation about the entity-body.

General Headers

- Cache-Control
- Connection
- Date
- Pragma
- Trailer
- Transfer-Encoding
- Upgrade
- Via
- Warning

https://tools.ietf.org/html/rfc2616#section-4.5

Request Headers

- Accept
- Accept-Charset
- Accept-Encoding
- Accept-Language
- Authorization
- Expect
- From
- Host
- If-Match
- If-Modified-Since
- If-None-Match
- If-Range
- If-Unmodified-Since
- Max-Forwards
- Proxy-Authorization
- Range
- Referer
- TE
- User-Agent

https://tools.ietf.org/html/rfc2616#section-5.3

Response Headers

- Accept-Ranges
- Age
- ETag
- Location
- Proxy-Authenticate
- Retry-After
- Server
- Vary
- WWW-Authenticate

https://tools.ietf.org/html/rfc2616#section-6.2

Entity Headers

- Allow
- Content-Encoding
- Content-Language
- Content-Length
- Content-Location
- Content-MD5
- Content-Range
- Content-Type
- Expires
- Last-Modified
- extension-header

https://tools.ietf.org/html/rfc2616#section-7.1

Cache-control

Headers used for cache control

- Expires: "Thu, 19 Nov 1981 08:52:00 GMT“
- Pragma: "no-cache“
- Age = 3600
- ETag: "5d2-50d275e263080"
- Last-Modified: Wed, 21 Jan 2015 10:53:38 GMT
- Cache-control: "no-store, no-cache, must-revalidate, post-check=0, pre-check=0“

https://tools.ietf.org/html/rfc2616#section-14.9

Cache-control header (request-directive)

Cache-Control = "Cache-Control" ":" cache-request-directive

cache-request-directive =

"no-cache“

| "no-store“

| "max-age" "=" delta-seconds

| "max-stale" ["=" delta-seconds]

| "min-fresh" "=" delta-seconds

| "no-transform“

| "only-if-cached“

| cache-extension

cache-extension = token ["=" (token | quoted-string)]

https://tools.ietf.org/html/rfc2616#section-14.9

Cache-control header (response-directive)

Cache-Control = "Cache-Control" ":" cache-response-directive

cache-response-directive =

"public“

| "private" ["=" <"> field-name <">]

| "no-cache" ["=" <"> field-name <">]

| "no-store“

| "no-transform"

| "must-revalidate“

| "proxy-revalidate"

| "max-age" "=" delta-seconds

| "s-maxage" "=" delta-seconds

| cache-extension

cache-extension = token ["=" (token | quoted-string)]

HTTP header (features)

- Blank PHP has function “header()”, but not all remember that this function can take three parameters:
  void header (string $string [, bool $replace = true [, int $http_response_code]])

NEVER do this:

```
header("Cache-Control: no-cache, must-revalidate");
header("Cache-Control: post-check=0,pre-check=0");
header("Cache-Control: max-age=0");
```

because header “Cache-Control” will be “max-age=0”

USE instead:

```
header("Cache-Control: no-cache, must-revalidate");
header("Cache-Control: post-check=0,pre-check=0", false);
header("Cache-Control: max-age=0", false);
```

Any questions?