# Hypertextové schéma HTTP - hlavičky, HTTP autentifikace, HTTP2 - rozdíly oproti HTTP1.1. Webové služby

## HTTP (Hypertext Transfer Protocol)

je internetový protokol určený pro komunikaci s WWW servery. Slouží pro přenos hypertextových dokumentů ve formátu HTML, XML, i jiných typů souborů. Používá obvykle port TCP/80, verze 1.1 protokolu je definována v RFC 2616. Společně s elektronickou poštou je HTTP nejvíce používaným protokolem, který se zasloužil o obrovský rozmach internetu.

Je však používán i pro přenos dalších informací. Pomocí rozšíření MIME umí přenášet jakýkoli soubor (podobně jako e-mail), používá se společně s formátem XML pro tzv. webové služby (spouštění vzdálených aplikací) a pomocí aplikačních bran zpřístupňuje i další protokoly, jako je např. FTP nebo SMTP.

Samotný protokol HTTP neumožňuje šifrování ani zabezpečení integrity dat. Pro zabezpečení HTTP se často používá TLS spojení nad TCP. Toto použití je označováno jako HTTPS.

### REQUEST Metody

**GET**

* Požadavek na uvedený objekt se zasláním případných dat (proměnné prohlížeče, [session](https://cs.wikipedia.org/wiki/Session) id, …). Výchozí metoda při požadavku na zobrazení hypertextových stránek, [RSS](https://cs.wikipedia.org/wiki/RSS) feedů aj. Celkově nejpoužívanější.

**HEAD**

* Metoda podobná GET, avšak nepředává data. Poskytne pouze metadata o požadovaném cíli (velikost, typ, datum změny, …).

**POST**

* Odesílá uživatelská data na server. Používá se například při odesílání formuláře na webu. S předaným objektem se pak zachází podobně jako při metodě GET. Data může odesílat i metoda GET, metoda POST se však používá pro příliš velký objem dat (víc než 512 bajtů, což je velikost požadavku GET), nebo pokud není vhodné přenášená data zobrazit jako součást [URL](https://cs.wikipedia.org/wiki/Uniform_Resource_Locator) (data předávaná metodou POST jsou obsažena v HTTP požadavku).

**PUT**

* Nahraje data na server. Objekt je jméno vytvářeného souboru. Používá se velmi zřídka, pro nahrávání dat na server se běžně používá FTP nebo SCP/SSH.

**DELETE**

* Smaže uvedený objekt ze serveru. K tomu je zapotřebí jistých oprávnění stejně jako u metody PUT.

**TRACE**

* Odešle kopii obdrženého požadavku zpět odesílateli, takže klient může zjistit, co na požadavku mění nebo přidávají servery, kterými požadavek prochází.

**OPTIONS**

* Dotaz na server, jaké podporuje metody.

| **Request method** | **RFC** | **Request has payload body** | **Response has payload body** | **Safe** | **Idempotent** | **Cacheable** |
| --- | --- | --- | --- | --- | --- | --- |
| **GET** | [RFC](https://en.wikipedia.org/wiki/RFC_(identifier)) [9110](https://datatracker.ietf.org/doc/html/rfc9110) | Optional | Yes | Yes | Yes | Yes |
| **HEAD** | [RFC](https://en.wikipedia.org/wiki/RFC_(identifier)) [9110](https://datatracker.ietf.org/doc/html/rfc9110) | Optional | No | Yes | Yes | Yes |
| **POST** | [RFC](https://en.wikipedia.org/wiki/RFC_(identifier)) [9110](https://datatracker.ietf.org/doc/html/rfc9110) | Yes | Yes | No | No | Yes |
| **PUT** | [RFC](https://en.wikipedia.org/wiki/RFC_(identifier)) [9110](https://datatracker.ietf.org/doc/html/rfc9110) | Yes | Yes | No | Yes | No |
| **DELETE** | [RFC](https://en.wikipedia.org/wiki/RFC_(identifier)) [9110](https://datatracker.ietf.org/doc/html/rfc9110) | Optional | Yes | No | Yes | No |
| **CONNECT** | [RFC](https://en.wikipedia.org/wiki/RFC_(identifier)) [9110](https://datatracker.ietf.org/doc/html/rfc9110) | Optional | Yes | No | No | No |
| **OPTIONS** | [RFC](https://en.wikipedia.org/wiki/RFC_(identifier)) [9110](https://datatracker.ietf.org/doc/html/rfc9110) | Optional | Yes | Yes | Yes | No |
| **TRACE** | [RFC](https://en.wikipedia.org/wiki/RFC_(identifier)) [9110](https://datatracker.ietf.org/doc/html/rfc9110) | No | Yes | Yes | Yes | No |
| **PATCH** | [RFC](https://en.wikipedia.org/wiki/RFC_(identifier)) [5789](https://datatracker.ietf.org/doc/html/rfc5789) | Yes | Yes | No | No | No |

## HTTP Hlavičky

Jedná se o list řetězců které obdrží jak klient tak server při http komunikaci (request a response). Headers jsou častokrát neviditelné pro klienta a jsou spracovány a logovány jen server a klientskou aplikací.

* V HTTP 1.x jsou headers posílány až za request line nebo response line. Jsou ve formátu “key:value” jako plain text ukončené za pomocí carriage return (CR) a line feed (LF) sequencí. KOnec header je označen prázdným polem což se zančí dvěmi po sobě jdoucími CR-LF páry. Následně při přenosu jsou headers nahrazeny několika pseudo headers fields každý začínající s :.
* V HTTP2 a HTTP3 které používají binární protocol jsou headers zakódovány do jodnoho HEADERS a 0 a vice CONTINUATION frames za použití HPACK (komprese v HTTP2) nebo QPACK (komprese v HTTP3)

Header názvy jsou case sensitive o jejich standardizaci se stará IETF (the Internet Engineering Task Force) v RFC 9110 and 9111. Nesatndartí header názvy by se měli označovat prefixem “X-“.

Value Headru může obsahuvat kvalitu která se následně používá při vyjednávání se serverem. Např: Accept-Language: de; q=1.0, en; q=0.5

Velikos header přímo omezuje webserver jakoje například Apache 2.3. kde je omezena velikost request na 8,190 bytesto může být maximálně 100 header fields in a single request.

### Standard request fields example

| **Name** | **Description** | **Example** | **Status** | **Standard** |
| --- | --- | --- | --- | --- |
| **Accept-Language** | List of acceptable human languages for response. See [Content negotiation](https://en.wikipedia.org/wiki/Content_negotiation). | Accept-Language: en-US | Permanent | [RFC](https://en.wikipedia.org/wiki/RFC_(identifier)) [9110](https://datatracker.ietf.org/doc/html/rfc9110) |
| **Authorization** | Authentication credentials for [HTTP authentication](https://en.wikipedia.org/wiki/Basic_access_authentication). | Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ== | Permanent | [RFC](https://en.wikipedia.org/wiki/RFC_(identifier)) [9110](https://datatracker.ietf.org/doc/html/rfc9110) |
| [**Cache-Control**](https://en.wikipedia.org/wiki/Cache-Control) | Used to specify directives that *must* be obeyed by all caching mechanisms along the request-response chain. | Cache-Control: no-cache | Permanent | [RFC](https://en.wikipedia.org/wiki/RFC_(identifier)) [9111](https://datatracker.ietf.org/doc/html/rfc9111) |
| **Content-Length** | The length of the request body in [octets](https://en.wikipedia.org/wiki/Octet_(computing)) (8-bit bytes). | Content-Length: 348 | Permanent | [RFC](https://en.wikipedia.org/wiki/RFC_(identifier)) [9110](https://datatracker.ietf.org/doc/html/rfc9110) |
| **Content-Type** | The [Media type](https://en.wikipedia.org/wiki/Media_type) of the body of the request (used with POST and PUT requests). | Content-Type: application/x-www-form-urlencoded | Permanent | [RFC](https://en.wikipedia.org/wiki/RFC_(identifier)) [9110](https://datatracker.ietf.org/doc/html/rfc9110) |
| **Cookie** | An [HTTP cookie](https://en.wikipedia.org/wiki/HTTP_cookie) previously sent by the server with [Set-Cookie](https://en.wikipedia.org/wiki/List_of_HTTP_header_fields#innerlink_set-cookie) (below). | Cookie: $Version=1; Skin=new; | Permanent: standard | [RFC](https://en.wikipedia.org/wiki/RFC_(identifier)) [2965](https://datatracker.ietf.org/doc/html/rfc2965), [6265](https://datatracker.ietf.org/doc/html/rfc6265) |
| **Date** | The date and time at which the message was originated (in "HTTP-date" format as defined by [RFC 9110: HTTP Semantics, section 5.6.7 "Date/Time Formats"](https://tools.ietf.org/html/rfc9110#section-5.6.7)). | Date: Tue, 15 Nov 1994 08:12:31 GMT | Permanent | [RFC](https://en.wikipedia.org/wiki/RFC_(identifier)) [9110](https://datatracker.ietf.org/doc/html/rfc9110) |
| Forwarded | Disclose original information of a client connecting to a web server through an HTTP proxy.[[16]](https://en.wikipedia.org/wiki/List_of_HTTP_header_fields#cite_note-16) | Forwarded: for=192.0.2.60;proto=http;by=203.0.113.43 Forwarded: for=192.0.2.43, for=198.51.100.17 | Permanent | [RFC](https://en.wikipedia.org/wiki/RFC_(identifier)) [7239](https://datatracker.ietf.org/doc/html/rfc7239) |
| From | The email address of the user making the request. | From: user@example.com | Permanent | [RFC](https://en.wikipedia.org/wiki/RFC_(identifier)) [9110](https://datatracker.ietf.org/doc/html/rfc9110) |
| **Host** | The domain name of the server (for [virtual hosting](https://en.wikipedia.org/wiki/Virtual_hosting)), and the [TCP port](https://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers) number on which the server is listening. The [port](https://en.wikipedia.org/wiki/Port_(computer_networking)) number may be omitted if the port is the standard port for the service requested.  Mandatory since HTTP/1.1.[[17]](https://en.wikipedia.org/wiki/List_of_HTTP_header_fields#cite_note-17) If the request is generated directly in HTTP/2, it should not be used.[[18]](https://en.wikipedia.org/wiki/List_of_HTTP_header_fields#cite_note-rfc9113_Request_Pseudo_Header_Fields-18) | Host: en.wikipedia.org:8080  Host: en.wikipedia.org | | |

### Autentifikace

Oba tyto dva typy by měli být použity výhradně přes HTTPS s TLS šifrování a validním SSL certifikátam. Jako odpověď pokud se neověří by měl přijít status code 401 unquthorized.

#### Basic

Můžeme zajistit zapomocí header Authentization pomocí **basic access authentication** metody. Kde můžeme posílat jméno a heslo. Header Authorization: Basic <credentials> kde credentials jsou Base64 šifrovaný ID a password spojený dohromady za pomocí “:”.

Např: Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==

#### Bearer

Jedná se o HTTP authentifikaci která využívá security token zvaná Bearer token. Můžem to taky chápat jako “dej přístup nositely tohoto tokenu”. Na poslání tohoto tokenu se takévyužívá Authorization header Authorization: Bearer <token>.

## HTTP 1.1

Jedná se o plaintextový protocol.

### Client request

GET **/** **HTTP**/1.1

Host: www.example.com

User-Agent: Mozilla/5.0

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,\*/\*;q=0.8

Accept-Language: en-GB,en;q=0.5

Accept-Encoding: gzip, deflate, br

Connection: keep-alive

### Server response

**HTTP**/1.1 200 **OK**

Date: Mon, 23 May 2005 22:38:34 GMT

Content-Type: text/html; charset=UTF-8

Content-Length: 155

Last-Modified: Wed, 08 Jan 2003 23:11:55 GMT

Server: Apache/1.3.3.7 (Unix) (Red-Hat/Linux)

ETag: "3f80f-1b6-3e1cb03b"

Accept-Ranges: bytes

Connection: close

<**html**>

<**head**>

<**title**>An Example Page</**title**>

</**head**>

<**body**>

<**p**>Hello World, this is a very simple HTML document.</**p**>

</**body**>

</**html**>

## HTTP 1.1 vs HTTP 2

| **HTTP/1.1** | **HTTP/2** |
| --- | --- |
| It works on the textual format. Plain text | It works on the binary protocol. binary |
| There is head of line blocking that blocks all the requests behind it until it the response does not receive. | It allows multiplexing so one TCP connection is used for multiple requests |
| It uses requests resource Inlining for use getting multiple pages | It uses PUSH frame by server that collects all multiple pages |
| It compresses data by itself. | It uses HPACK for data compression. |

### HTTP2 je vyvinut na SPDY protokolu:

#### Povinné funkcionality

**Multiplexované spojení**

SPDY podporuje neomezený počet souběžných toků dat skrze jediné TCP spojení. Efektivita spojení je maximalizována, protože jsou jednotlivé dotazy posílány zároveň. Sníží se tím také počet TCP spojení k jednomu web serveru.

**Priorita dotazování**

S multiplexem je spojen problém priority. Při pomalém spojení může dojít k zadržení [paketů](https://cs.wikipedia.org/wiki/Packet" \o "Packet), které klient nutně potřebuje. SPDY implementuje prioritu dotazů (úrovně 0 až 7)[[2]](https://cs.wikipedia.org/wiki/SPDY#cite_note-SPDY_protocol-2), která tento problém efektivně řeší.

**Komprimace hlaviček**

Komprimace hlaviček je vždy zapnutá a snižuje se tím počet odeslaných dat. Hlavičky jsou vždy komprimovány pomocí komprese [zlib](https://cs.wikipedia.org/wiki/Zlib" \o "Zlib).

#### Nepovinné funkcionality

**Server push**

Na rozdíl od HTTP, může sám server začít odesílat data. V hlavičce předá klientovi informaci, že začne odesílat data, která si klient ještě nevyžádal. Toto opatření může značně zrychlit načítání stránek, které klient ještě nenavštívil. Pokud má již klient data v [paměti](https://cs.wikipedia.org/wiki/Cache" \o "Cache) pak je odeslání zbytečné, rozhodnutí o odeslání dat náleží jenom serveru, jelikož protokol neposkytuje informace o datech která jsou uloženy u klienta.[[13]](https://cs.wikipedia.org/wiki/SPDY#cite_note-13)

**Server hint**

Server má možnost, místo aktivního odesílání dat, pouze informovat klienta o potřebných datech. Klient pak může rychleji zareagovat vlastním dotazem. Při pomalém spojení klient rychleji zjistí, která data potřebuje, ještě před tím než by se mu stáhl předchozí dotaz.

### HTTP2 HPACK

V HTTP/2 jsou hlavičky ve formátu binárních rámů, které obsahují název a hodnotu jednotlivých polí. HPACK je používán k redukci redundance a snížení velikosti přenášených dat při komunikaci. Při kompresi HPACK využívá následující techniky:

1. Statická tabulka: HPACK obsahuje statickou tabulku, která obsahuje nejběžněji používané hlavičky, jako například "Content-Type" nebo "Accept-Encoding". Tyto hlavičky jsou předem zakódovány a jsou k dispozici jak na straně klienta, tak na straně serveru.
2. Dynamická tabulka: HPACK udržuje také dynamickou tabulku, ve které jsou uloženy nedávno použité hlavičky. Tato tabulka je výměnná mezi klientem a serverem a umožňuje zakódování hlaviček, které nejsou obsaženy ve statické tabulce.
3. Huffmanovo kódování: HPACK používá Huffmanovo kódování pro zakódování často používaných řetězců. Tato technika snižuje velikost přenášených dat zakódováním často se vyskytujících znaků do kratších binárních kódů.

## Webové služby

* Služba nbízena elektronickým zařízením jinému elektronickému zařízení, komunikující spolu přez internet
* Služba běžící na počítači, poslouchající pro requesty na určitém portu přes internet, (např: serving web document http, JSON, XML, IMG)

Většinou obsahují object-oriented web-based interface do databazového server, využito například dalším web serverem nebo pomocí mobilní aplikace. Mnoho společnostá které poskytuje data v HTML formátu budou take poskytovat tyto data ve formě XML nebo JSON na jejich serverech.

Existují i jiné způsoby pro komunikaci s webovou službou: Je jím například komunikace podle filosofie CRUD (Create, Read, Update, Delete - vytvoř-zapiš, přečti-vrať, změň, smaž) realizované typicky jako Representational State Transfer (REST), který v provedení nad protokolem HTTP využívá právě čtyř jeho metod (POST, GET, PUT, DELETE). Aby bylo možno s daty na serveru pracovat, tato čtyři volání vystavené služby stačí.

webserver Apache, Ngnix, node expres (jenom jako proxi)

ngnix řeší šifrování apod

používá se API REST předávjí se data

http metody

SSL/TLS

## Status codes

|  |  |
| --- | --- |
| **Status code** | **Meaning** |
| [1xx Informational](https://umbraco.com/knowledge-base/http-status-codes/#1xx) |  |
| 100 | Continue |
| 101 | Switching protocols |
| 102 | Processing |
| 103 | Early Hints |
|  |  |
| [2xx Succesful](https://umbraco.com/knowledge-base/http-status-codes/#2xx) |  |
| 200 | OK |
| 201 | Created |
| 202 | Accepted |
| 203 | Non-Authoritative Information |
| 204 | No Content |
| 205 | Reset Content |
| 206 | Partial Content |
| 207 | Multi-Status |
| 208 | Already Reported |
| 226 | IM Used |
|  |  |
| [3xx Redirection](https://umbraco.com/knowledge-base/http-status-codes/#3xx) |  |
| 300 | Multiple Choices |
| 301 | Moved Permanently |
| 302 | Found (Previously "Moved Temporarily") |
| 303 | See Other |
| 304 | Not Modified |
| 305 | Use Proxy |
| 306 | Switch Proxy |
| 307 | Temporary Redirect |
| 308 | Permanent Redirect |
|  |  |
| [4xx Client Error](https://umbraco.com/knowledge-base/http-status-codes/#4xx) |  |
| 400 | Bad Request |
| 401 | Unauthorized |
| 402 | Payment Required |
| 403 | Forbidden |
| 404 | Not Found |
| 405 | Method Not Allowed |
| 406 | Not Acceptable |
| 407 | Proxy Authentication Required |
| 408 | Request Timeout |
| 409 | Conflict |
| 410 | Gone |
| 411 | Length Required |
| 412 | Precondition Failed |
| 413 | Payload Too Large |
| 414 | URI Too Long |
| 415 | Unsupported Media Type |
| 416 | Range Not Satisfiable |
| 417 | Expectation Failed |
| 418 | I'm a Teapot |
| 421 | Misdirected Request |
| 422 | Unprocessable Entity |
| 423 | Locked |
| 424 | Failed Dependency |
| 425 | Too Early |
| 426 | Upgrade Required |
| 428 | Precondition Required |
| 429 | Too Many Requests |
| 431 | Request Header Fields Too Large |
| 451 | Unavailable For Legal Reasons |
|  |  |
| [5xx Server Error](https://umbraco.com/knowledge-base/http-status-codes/#5xx) |  |
| 500 | Internal Server Error |
| 501 | Not Implemented |
| 502 | Bad Gateway |
| 503 | Service Unavailable |
| 504 | Gateway Timeout |
| 505 | HTTP Version Not Supported |
| 506 | Variant Also Negotiates |
| 507 | Insufficient Storage |
| 508 | Loop Detected |
| 510 | Not Extended |
| 511 | Network Authentication Required |