# HTTP szolgáltatások

## A HTTP alapjai

A HTTP (HyperText Transfer Protocol) egy alkalmazás réteg protokoll, amely kliens–szerver modellben működik. Elsősorban webes erőforrások (HTML, CSS, JS, képek, API válaszok stb.) továbbítására használják.

Alapvető jellemzők:

- Állapotmentes (stateless)
- Kérés-válasz alapú (request–response)
- TCP (HTTP/1.x, HTTP/2) vagy QUIC/UDP (HTTP/3) felett működik
- Szöveges (HTTP/1.x) illetve bináris (HTTP/2, HTTP/3)

## HTTP működési modell

A kommunikáció menete:

Kliens kapcsolatot nyit a szerverhez

Kliens HTTP kérést küld

Szerver HTTP választ küld

Szerver lezárja a kapcsolatot (HTTP 0.9, HTTP 1.0) vagy nyitva marad ha szükséges (HTTP 1.1, HTTP 2)

## HTTP protokol verziók

### HTTP/0.9 (1991)

A HTTP/0.9 volt a legelső, kísérleti változata a protokollnak, amelyet Tim Berners-Lee fejlesztett ki a CERN-ben a World Wide Web kezdeti időszakában.

Jellemzői:

Rendkívül egyszerű

Csak egyetlen metódus létezett: GET

Nem léteztek HTTP headerek

Nem volt státuszkód

Nem volt verziószám a kérésben

A válasz kizárólag HTML tartalom volt

#### Kérés

```
# telnet szervernév 80
GET /index.html
```

#### Válasz

A szerver egyszerűen visszaküldi a HTML dokumentumot:

```
<html>
<body>Hello World</body>
</html>
```

Nincs:

státuszsor header mező Content-Type Content-Length

#### Korlátok

Nem támogatta a képeket, CSS-t, külön erőforrásokat Nem volt hibakezelési mechanizmus Nem alkalmas komplex webalkalmazásokhoz

#### Összegzés

A HTTP/0.9 egy rendkívül egyszerű, csak GET-et támogató, header nélküli protokoll volt, amely kizárólag HTML dokumentum visszaküldésére szolgált. Ez tekinthető a modern webkommunikáció kiindulópontjának.

## HTTP/1.0 (1996)

#### Jellemzők:

Minden kérés külön TCP kapcsolat Nincs alapértelmezett persistent connection Egyszerű szöveges protokoll Nincs Host header kötelezően

#### Problémák:

Sok TCP handshake Magas latency Nem hatékony párhuzamos erőforrás-letöltés

Példa kommunikáció:

Egyszerű kérés felépítése:

```
$ telnet server 80
Trying N.N.N.N...
Connected to szerver.
Escape character is '^]'.
GET / HTTP/1.0

HTTP/1.1 200 OK
Date: Sun, 15 Feb 2026 03:32:17 GMT
Server: Apache/2
Last-Modified: Tue, 27 Feb 2024 06:32:03 GMT
ETag: "0-612572f2e4ba2"
Accept-Ranges: bytes
Content-Length: 0
Connection: close
Content-Type: text/html; charset=UTF-8

Connection closed by foreign host.
```

## HTTP/1.1 (1997)

**Fejlesztések:**

- Persistent connection (Keep-Alive)
- Host header kötelező
- Chunked transfer encoding
- Pipeline támogatás

Példa a hagyományos kapcsolatra:

```
$ telnet server 80
Trying N.N.N.N...
Connected to server.
Escape character is '^]'.
GET / HTTP/1.1
Host: server

HTTP/1.1 403 Forbidden
Date: Sun, 15 Feb 2026 03:47:36 GMT
Server: Apache/2.4.62 (AlmaLinux)
Last-Modified: Mon, 24 Mar 2025 16:15:24 GMT
ETag: "1680-63118e9567f00"
Accept-Ranges: bytes
Content-Length: 130
Content-Type: text/html; charset=UTF-8
```

```
<!DOCTYPE html>
<html>
<body>
Hello world!
</body>
</html>
Connection closed by foreign host.
```

Példa persistent kapcsolatra:

```
$ telnet server 80
Trying N.N.N.N...
Connected to server.
Escape character is '^]'.
GET / HTTP/1.1
Host: xxx
Connection: keep-alive

TARTALOM
GET / HTTP/1.1
Host: xxx
Connection: close

TARTALOM

Connection closed by foreign host.
```

Előny:

- Kevesebb TCP kapcsolat
- Jobb teljesítmény

Probléma:

Head-of-line (HOL) blokkolás (több kérés esetén). Mivel a kérések sorrendjében válaszol a szerver, ezért egy lassú feldolgozási művelet blokkolja a többi kérés kiszolgálását.

# HTTP/2 (2015)

Alapja: SPDY protokoll

Fő jellemzők:

Bináris protokoll

Multiplexing (több stream egy TCP kapcsolaton)

Header compression (HPACK)

Server Push

Előny:

Megszűnik az alkalmazás-szintű head-of-line blocking

Jelentősen jobb teljesítmény

Hátrány:

TCP head-of-line blocking továbbra is fennáll csomagszinten

## HTTP/3 (2022)

Alapja: QUIC (UDP felett)

Jellemzők:

TCP helyett UDP

TLS 1.3 integrált

Nincs TCP head-of-line blocking

Gyorsabb kapcsolatfelépítés (0-RTT)

Előny:

Jelentősen alacsonyabb latency

Mobil hálózatokon stabilabb

## HTTP verziók összehasonlítása

| Verzió | Szállítási réteg | Formátum | Multiplexing | Head-of-line blocking |
|---|---|---|---|---|
| HTTP/1.0 | TCP | Szöveges | Nem | Igen |
| HTTP/1.1 | TCP | Szöveges | Korlátozott (pipeline) | Igen |
| HTTP/2 | TCP | Bináris | Igen | TCP szinten igen |
| HTTP/3 | QUIC (UDP) | Bináris | Igen | Nem |

## Telnet-es HTTP példa (HTTP/1.0)

Kapcsolódás:

```
telnet example.com 80
```

Kérés kézi begépelése:

```
GET / HTTP/1.0
```

(Fontos: az üres sor zárja a header részt.)

# Telnet-es HTTP/1.1 példa

```
telnet example.com 80
```

```
GET / HTTP/1.1 Host: example.com Connection: close
```

Megfigyelhető:

Kötelező Host header

Válasz státuszkód

Header mezők

Üres sor

Body

# HTTP státuszkódok

| Tartomány | Jelentés |
|---|---|
| 1xx | Információ |
| 2xx | Siker |
| 3xx | Átirányítás |
| 4xx | Kliens hiba |
| 5xx | Szerver hiba |

Példák:

200 OK

301 Moved Permanently

400 Bad Request

404 Not Found

500 Internal Server Error

# HTTPS

A HTTPS nem külön protokoll, hanem:

HTTP + TLS titkosítás

Alapértelmezett port:

80 → HTTP

443 → HTTPS

Telnet helyett HTTPS teszteléshez:

```
openssl s_client -connect example.com:443
```

# Összegzés

HTTP/1.x: egyszerű, szöveges, TCP-alapú

HTTP/2: bináris, multiplexelt, gyorsabb

HTTP/3: QUIC-alapú, modern, alacsony latency

Telnet kiváló eszköz a HTTP/1.x működés szemléltetésére

From:
https://wiki.r-l.hu/ - **Reverse-Logic wiki**

Permanent link:
**https://wiki.r-l.hu/doku.php?id=webszerver&rev=1771128157**

Last update: **2026/02/15 04:02**