HTTP működése – HTTP/1.1 alapjai

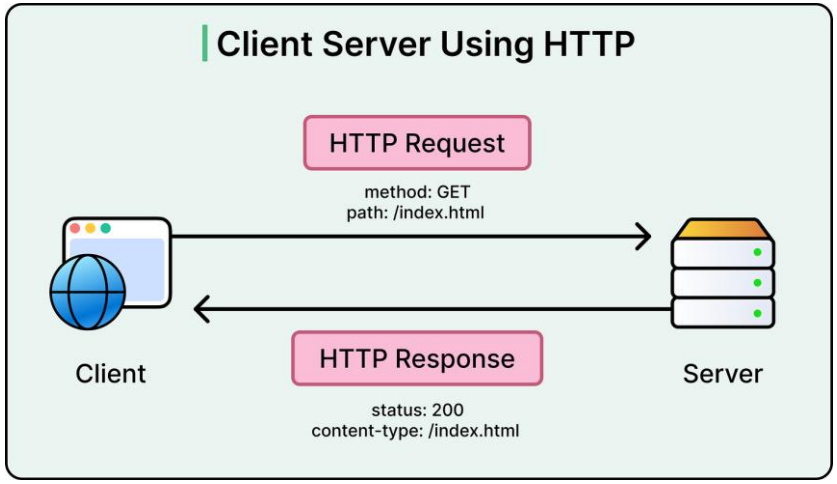
Tartalom:

- HTTP metódusok
- Státuszkódok
- Headerek
- Request–Response folyamat
- Content Negotiation
- Cache kezelés alapjai

Mi az a HTTP?

HyperText Transfer Protocol

- Kliens–szerver alapú protokoll
- TCP felett működik (alkalmazási rétegben)
- Állapotmentes (minden kérés független)
- Szöveges (Human-readable)
- HTTP/1.1 a legelterjedtebb verzió



Hol helyezkedik el a HTTP?

HTTP a TCP/IP modellben:

- ***HTTP → Application layer***
- TCP → Transport layer
- IP → Internet layer
- HTTP az alkalmazás logikával foglalkozik
- Ezért fejlesztőként nekünk ez a legfontosabb réteg.

HTTP üzenetek felépítése

Két fő elem:

- Kérés (Request)
- Válasz (Response)

Request tartalma:

- Request line (pl. GET /index.html HTTP/1.1)
- Fejlécek (Headerek)
- Törzs (Body), ez opcionális

Response tartalma:

- Status line
- Fejlécek (Headerek)
- Törzs (Body): tipikusan HTML, JSON, kép stb.

HTTP Request példa

GET /products HTTP/1.1

Host: example.com

User-Agent: Chrome/123

Accept: text/html

HTTP Response példa

HTTP/1.1 200 OK

Content-Type: text/html

Content-Length: 532

<html>...</html>

HTTP/1.1 újdonságai

A korábbi 1.0-hoz képest:

- Persistent (**keep-alive**) kapcsolatok
- **Host** header kötelező
- Fejlettebb **cache**-elés támogatása
- **Chunked** Transfer Encoding
- Content-Encoding

HTTP metódusok áttekintése

- **GET -** adatlekérés
- **POST** – adat küldés/létrehozás (pl. űrlapok)
- **PUT** – adat módosítás
- **PATCH** – részleges adat módosítás
- **DELETE** – adat törlés
- **HEAD** – fejlécek lekérése (tartalom nélkül)
- **OPTIONS** – elérhető opciók és műveletek lekérdezése

GET metódus

- Olvasási művelet (lekérdezés)
- Általában nincs body
- Biztonságos (safe)
- Idempotens
- Gyorsítótárazható (Caching)
- URL query paramétereket használhat

POST metódus

- Új erőforrás létrehozása
- Body szükséges az adatokkal
- Nem idempotens művelet
- Formok, API-k tipikus metódusa

PUT és PATCH metódusok

- **PUT**
- Teljes erőforrás cseréje/felülírása
- Idempotens

- **PATCH**
- Részleges módosítás
- Nem feltétlen idempotens

DELETE metódus

- Erőforrás törlése
- Idempotens
- API hívások ezt használják törlésre

HTTP státuszkódok - kategóriák

- 1xx – Információs
- 2xx – Siker
- 3xx – Átirányítás
- 4xx – Kliens hiba
- 5xx – Szerver hiba

Gyakoribb 2xx és 3xx kódok

- **2xx**
- 200 OK
- 201 Created
- 204 No Content

- **3xx**
- 301 Moved Permanently
- 302 Found (Moved Temporarily)
- 304 Not Modified

Gyakoribb 4xx és 5xx kódok

- **4xx – klienshibák**
- 400 Bad Request
- 401 Unauthorized
- 403 Forbidden
- 404 Not Found

- **5xx – szerverhibák**
- 500 Internal Server Error
- 503 Service Unavailable

HTTP headers (fejlécek) szerepe

- **Két fő típus:**
- Request headerek
- Response headerek

- **Tartalmazhatnak:**
- Metaadatokat
- Tartalom (content) leírás
- Nyelv, tömörítés információk
- Cache-információk
- Hítelesítési adatok
- Kapcsolatkezelés (keep-alive)

Fontos request és response headerek

- **Request:**
- Host
- User-Agent
- Accept
- Accept-Language
- Authorization
- Cache-Control

- **Response:**
- Content-Type
- Content-Length
- Set-Cookie
- Cache-Control
- Connection: keep-alive
- Keep-Alive: timeout=30, max=100

Content Negotiation

„Milyen formátumban szeretné a kliens az adatot?”

- Accept: text/html, application/json
- Accept-Language: hu-HU, en-US
- Accept-Encoding: gzip

Caching alapjai

- **Példák a Cache-Control fejlécre:**
- max-age
- no-cache
- no-store
- public / private

Cache-Control: public, max-age=3600

vagy

Cache-Control: no-store, no-cache

Összefoglalás

- **HTTP** egyszerű, szöveges, stateless
- **Metódusok** → szabályozzák a műveleteket
- **Státuszkódok** → kommunikálnak a klienssel
- **Headerek** → metaadatok, tömörítés, hitelesítés, cache
- **Content negotiation** → a kliens igényeihez igazított válasz
- **Caching** → gyorsabb web, kevesebb terhelés