Backend programozás

HTTP Request

Rostagni Csaba

2025. szeptember 28.

Tartalom

Tartalom

Az URL felépítése

URL eleje

`http://example.com:80/mappa/index.php`

- **Protokoll**: `http`, `https`, `ftp`, ...
- **Host**:
  - `localhost`
  - `127.0.0.1` IP
  - `example.com` (domain + TLD)
  - `pelda.example.com` (aldomain + domain + TLD)
- **Port**: 80 (ez az alapértelmezett, így gyakran elhagyjuk)
- **Path** documentroot-tól számítva tartalmazza
  - mappaszerkezetet
  - script nevét

Az URL felépítése

- Query String
  - Az URL-ben a ? utáni rész
  - Felépítése: `kulcs=érték`
  - Az elválasztó kerekter: `&`
- Fragment
  - Az URL-ben a # utáni rész
  - Az oldalon belüli ugrópont (horgony)

Linkek:

- Query string (wikipedia)
- RFC-3986 szabvány

Az URL felépítése

URL vége

```
index.php?nev=Tóth Béla&kor=26&sport=hoki
```

index.php

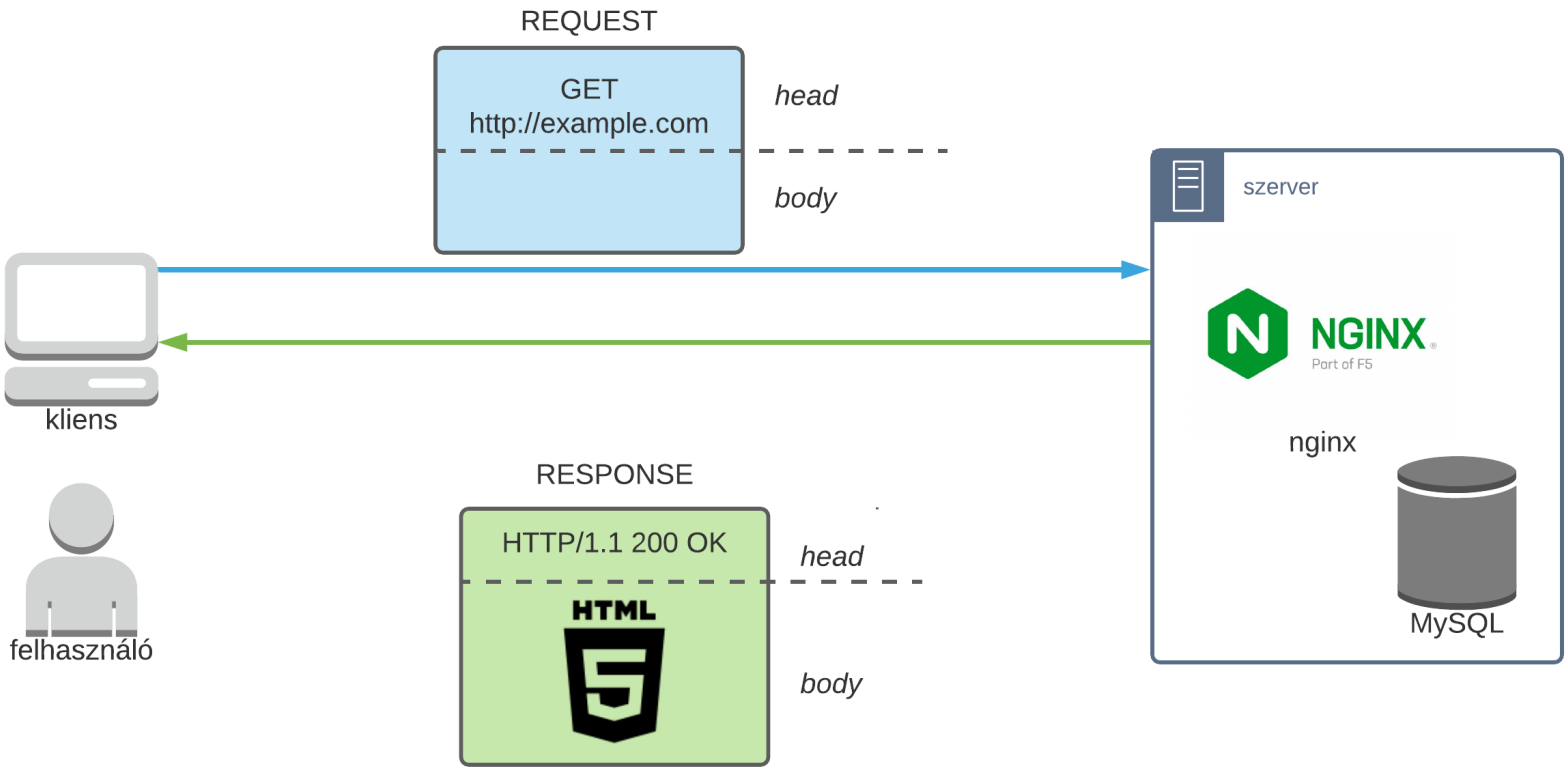
```php
<?php
    var_dump($_GET);
?>
```

Eredmény

```
array(3) {
    ["nev"]     =>  string(11) "Tóth Béla"
    ["kor"]     =>  string(2)  "26"
    ["sport"]   =>  string(4)  "hoki"
}
```

Tartalom

HTTP kérés (nginx szerver)



Hello world request

https://example.com/hello-world.html

Request - header

```
GET /hello-world.html HTTP/1.1
Host: example.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:96.0)
↪  Gecko/20100101 Firefox/96.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,
↪  image/avif,image/webp,*/*;q=0.8
Accept-Language: hu-HU,hu;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate, br
DNT: 1
Connection: keep-alive
Referer: https://example.com/
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Pragma: no-cache
Cache-Control: no-cache
```

Response

Response - header

```
HTTP/1.1 200 OK
Server: nginx/1.21.3
Date: Mon, 24 Jan 2022 08:50:26 GMT
Content-Type: text/html
Content-Length: 69
Connection: keep-alive
Last-Modified: Mon, 24 Jan 2022 08:49:26 GMT
ETag: "61ee6816-45"
Accept-Ranges: bytes
```

A Response body tartalma

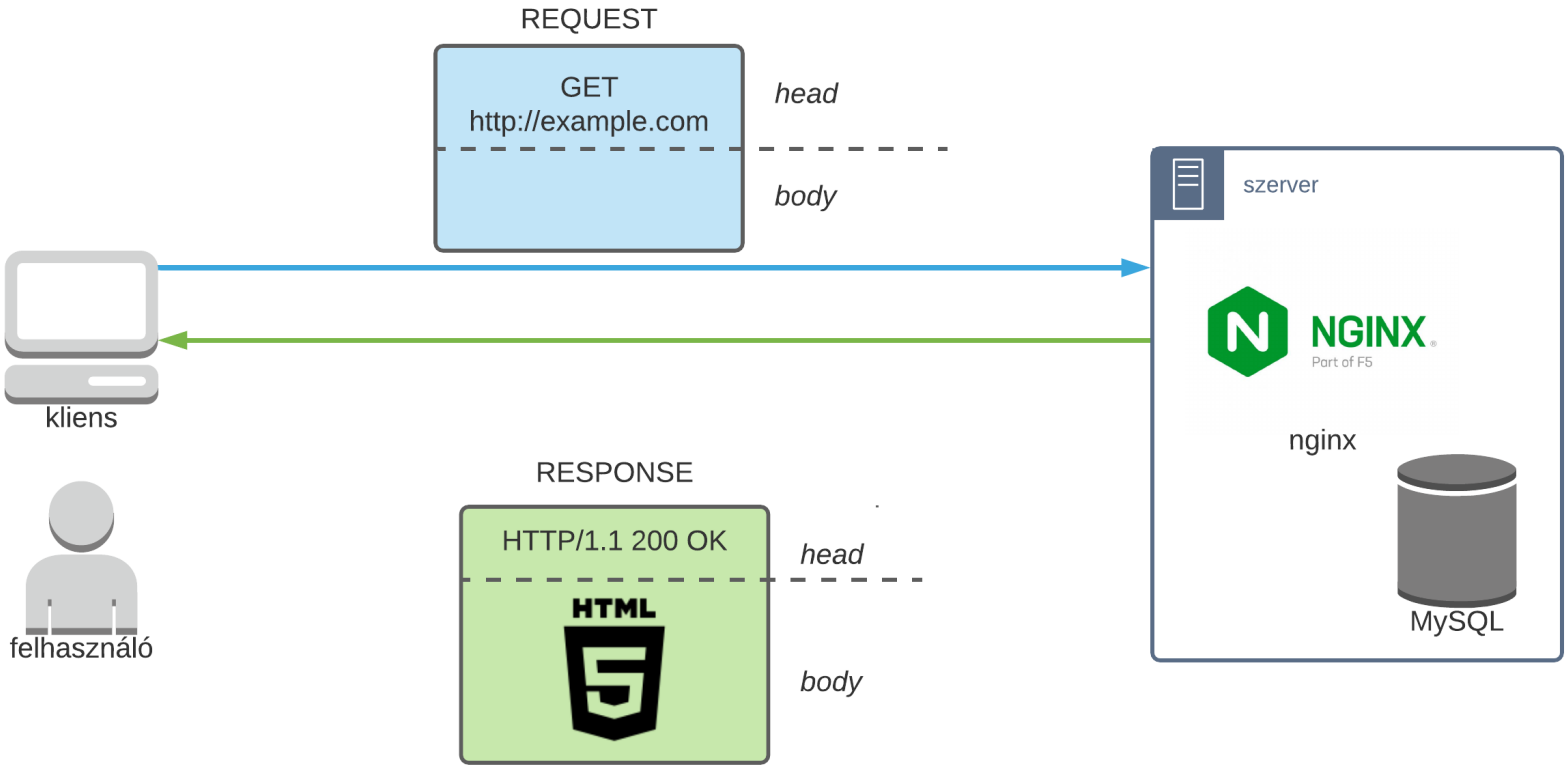
Response - body

```
<!DOCTYPE html>
<html>
<body>

<h1>Hello World</h1>

</body>
</html>
```

HTTP kérés (nginx szerver)



HTTP állapotkódok

- **1xx** - Info
- **2xx** - Sikeres kérés
  - **200** Sikeres kérés
- **3xx** - Átirányítás
  - **301** Végleges áthelyezés
  - **302** Ideiglenes áthelyezés
- **4xx** - Klienshiba
  - **403 Forbidden** A felhasználónak nincs jogosultsága elérni az oldalhoz
  - **404 Not found** Az oldal nem található
- **5xx** - Szerverhiba
  - **500** Általános szerverhiba.

Linkek:

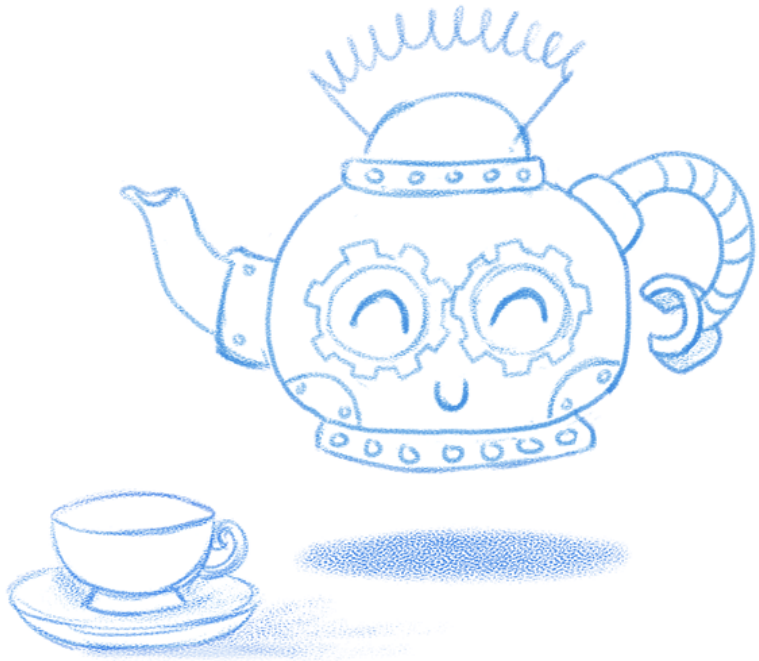
- Wikipedia: HTTP állapotkódok
- https://http.cat

418 - I’m a teapot

Google

418. I’m a teapot.

The requested entity body is short and stout.
Tip me over and pour me out.



Linkek:

- MDN: Status 418
- http.cat 418

Tartalom

Request

- Az `Illuminate\Http` névtérben található `Request` osztály segítségével dolgozható fel egy HTTP kérés.
  - Input (GET, POST, ...)
  - Cookies
  - Files

A Request használata

routes/web.php

```php
use \App\Http\Controllers\CarController;

Route::get('/cars', [CarController::class,"index"]);
```

app/Http/Controllers/CarController.php

```php
use Illuminate\Http\Request;

class CarController extends Controller {

    public function index(Request $request) {
        dd($request->all());
    }
}
```

- Az action **első** paramétere legyen egy `Request` típusú változó!

A Request használata paraméterrel

routes/web.php

```php
use \App\Http\Controllers\CarController;

Route::get('/cars/{rendszam}', [
    CarController::class,
    "show"
]);
```

app/Http/Controllers/CarController.php

```php
use Illuminate\Http\Request;

class CarController extends Controller {

    public function show(Request $request, string $rendszam) {
        dd($request->all());
    }
}
```

- Az action **első** paramétere legyen egy `Request` típusú változó!
- A route paramétere(i) utána jöhet(nek)
- A `dd()` kiírja a kapott paramétert, majd lehal

all()

PHP
```php
public function index(Request $request) {
    dd($request->all());
}
```

- Az `all()` segítségével az összes bemeneti adatot megkapjuk egy tömbként
- A `dd()` kiírja a kapott paramétert, majd lehal

collect()

```php
public function index(Request $request) {
    dd($request->collect());
}
```

- A `collect()` segítségével az összes bemeneti adatot megkapjuk egy gyűjteményként

Linkek:

- Laravel dokumentáció: Gyűjtemények (Collections)

input()

PHP

```php
public function index(Request $request) {
    $name = $request->input("name");
    $age = $request->input("age",0);

    $data = $request->input();

    $bestGame = $request->input("games.0.title");
}
```

- Az `input()` segítségével tetszőleges bemenetből (GET, POST, . . .) megkaphatjuk az adatot
- A második paraméter az alapértelmezett érték
- Paraméter nélkül az összes adatot megkapjuk
- Tömb bemenet esetén a pont jelöléssel érhető el az érték. (<input name="games[0][title]" . . . >)

query()

PHP

```php
public function index(Request $request) {
    $name = $request->query("name");
    $age = $request->query("age",0);

    $data = $request->query();
}
```

- Az `query()` segítségével a Query String adatait (GET) kaphatjuk meg
- A második paraméter az alapértelmezett érték
- Paraméter nélkül az összes adatot megkapjuk