HTTP protokoll

A HTTP *(**H**yper**T**ext **T**ransfer **P**rotocol)* egy TCP feletti, alkalmazási rétegbeli protokoll, melyet széleskörben használnak webalkalmazások integrációja során.

Működése

- A HTTP egy **kliens-szerver** protokoll, amely kapcsolatot teremt egy kliens (például egy webböngésző) és egy szerver (például egy webszerver) között.
- A HTTP **kérés-válasz modellt** használ. A kliens küld egy kérést a szervernek, majd a szerver választ küld a kliensnek.
    - A HTTP **kérések** egy metódusból, egy URL-ből, opcionális fejlécekből és törzsből állnak. A leggyakoribb HTTP metódusok a GET, POST, PUT és DELETE.
    - A HTTP **válaszok** egy állapotkódból, valamint opcionális fejlécekből és választörzsből állnak. Az állapotkód jelzi, hogy a kérés sikeres volt-e, és a válasz törzse tartalmazza a kért adatokat (vagy adott esetben a hiba okát).
- A HTTP egy **állapotmentes** protokoll, ami azt jelenti, hogy nem emlékszik a korábbi kommunikációra a kliens és a szerver között. Minden kérés/válasz csere független az előző adatcseréktől.

Kérés

Egy példa kérés felépítése a következő:

```
GET /template/web/img/logo-uni-miskolc.png HTTP/1.1
User-Agent: curl/7.35.0
Host: www.uni-miskolc.hu
Accept: */*
```

A kérés első sora jelöli meg az elérni kívánt erőforrást. Ezt a kéréshez tartozó, tetszőleges számú **fejléc** (header) követi, `Fejléc: érték` alakban.

Ezt követi az opcionális **törzs** (body) rész, melyben a kérés teljesítéséhez szükséges adatokat helyezhetünk el tetszőleges formátumban (pl. egy új felhasználó adatait JSON formátumban).

Metódusok

A leggyakrabban alkalmazott metódusok és szimbolikus jelentésük:

- GET: Adatok lekérdezése
- POST: Új adat hozzáadása
- PUT: Meglévő adat módosítása

- DELETE: Meglévő adat törlése

Query paraméterek

A GET kérés speciális, mert törzs részt nem tartalmazhat. Helyette - amennyiben szükséges - a kéréshez tartozó paramétereket ún. query paraméterekként adhatjuk meg.

Példa: `/template/web/img/logo-uni-miskolc.png?time=2020-05-01&thumb=true`

A paraméterek `név=érték` formában adhatóak meg. A paramétereket az útvonaltól `?`, egymástól `&` jel választja el.

Path paraméterek

A HTTP szabványnak ugyan nem része, de a szerverek gyakran támogatják a paraméterek átadását a megjelölt útvonal részeként:

Példa: GET `/api/users/`**`Bela123`**`/repos/`**`my-awesome-project`**

A szerver ebben az esetben a kérésben szereplő útvonalból olvas ki paramétereket. A fenti példában ilyen paraméter a felhasználó (`Bela123`) és a keresett repository elnevezése (`my-awesome-project`). Az útvonal ezen részei a keresett repository-nak megfelelően módosíthatók tetszőleges értékekre.

Kérés törzsében szereplő paraméterek

POST, PUT, DELETE kéréseknél opcionálisan a törzs rész is kitölthető.

Például a következő kérés egy felhasználó létrehozására szolgál:

```
POST /users HTTP/1.1
User-Agent: curl/7.35.0
Host: api.example.com
Accept: */*
Content-Type: application/json
Content-Length: 28

{ "name": "feri", "gender": "male", "age": 15 }
```

A fenti üzenet tartalmazza a megfelelő HTTP metódust (`POST`) és útvonalat (`/users`). Ezek együttesen azonosítják a szerver számára, hogy a kliens milyen műveletet szeretne végrehajtani (új felhasználó létrehozása). Emellett láthatunk még néhány szükséges fejlécet, és a kérés törzsében az újonnan létrehozni kívánt felhasználó adatait, JSON formátumban.

Válasz

A kliens kéréseire a szerver válaszokat ad. Egy kéréshez legfeljebb egy válasz tartozhat.

A HTTP válasz a következőképpen épül fel:

```
HTTP/1.1 200 OK
Date: Thu, 18 Mar 2021 13:02:06 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 29
Connection: keep-alive
X-Powered-By: Express
X-Ratelimit-Limit: 1000
X-Ratelimit-Remaining: 999
X-Ratelimit-Reset: 1616072536
Cache-Control: no-cache
Pragma: no-cache

{ "id": 101, "name": "feri", "gender": "male", "age": 15 }
```

A válasz egyebek mellett tartalmaz egy **státuszkódot** (a fenti példában 200), ami a művelet sikerességéről ad információt.

Ezt követik a válaszhoz tartozó fejlécek.

A törzs rész opcionális: a fenti válaszban a szerver visszaküldte az adatbázishoz hozzáadott felhasználó adatait, köztük egy azonosítóval, amire a későbbiekben a kliens hivatkozhat, ha szeretné elérni a létrehozott felhasználót.

HTTP státuszkódok

Az egyes HTTP státuszkódok jelentését a HTTP specifikációja tartalmazza, általánosságban a következők írhatóak le róluk:

- 1xx: Informatív üzenet (pl. 101 Switching Protocols)
- 2xx: A kérést a szerver sikeresen megkapta, értelmezte, teljesítette (pl. 200 OK, 201 Created)
- 3xx: Átirányítás (pl. 301 Moved Permanently)
- 4xx: Kliens hiba (pl. 400 Bad Request, 404 Not found)
- 5xx: Szerver hiba (pl. 503 Service Unavailable)

(Elérhető státuszkódok)

Példa - JSONPlaceholder API

A JSONPlaceholder egy ingyenes, nyilvánosan elérhető API, amelyet elsősorban kezdő webfejlesztők

használnak HTTP API-k tesztelésére és gyakorlására. Az API különböző adatokkal rendelkezik, például felhasználókkal, bejegyzésekkel, kommentekkel és egyéb információkkal, amelyeket HTTP kérésekkel kérhetünk le vagy módosíthatunk.

- **Ingyenes**: Bármikor használhatjuk, nem szükséges regisztráció vagy autentikáció.
- **Fiktív adatok**: Az API-ban szereplő adatok nem valódiak, hanem példák, amelyek segítenek a fejlesztőknek a gyakorlásban.
- **Könnyen használható**: Az API-t könnyen elérhetjük GET, POST, PUT, DELETE HTTP kérésekkel.

A JSONPlaceholder API könnyen használható különböző eszközökkel, például:

- **Böngésző**: A GET kéréseket közvetlenül a böngésző címsorába írva is megnézheted.
- **Postman**: A Postman segítségével POST, PUT, DELETE kéréseket is küldhetsz.
- **Fetch API**: JavaScript segítségével lekérheted az adatokat és dolgozhatsz velük egy webalkalmazásban.

HTTP kérések feldolgozása fejlesztői eszközökkel

HTTP kérések küldésére és válaszok fogadására számos módszer áll rendelkezésre, amelyek az egyszerű parancssori alkalmazásoktól (pl. curl, wget) kezdve a grafikus felülettel rendelkező eszközökön (pl. webböngészők, Postman) át a programozói könyvtárakig (pl. Fetch API, Axios, OkHttp) terjednek.

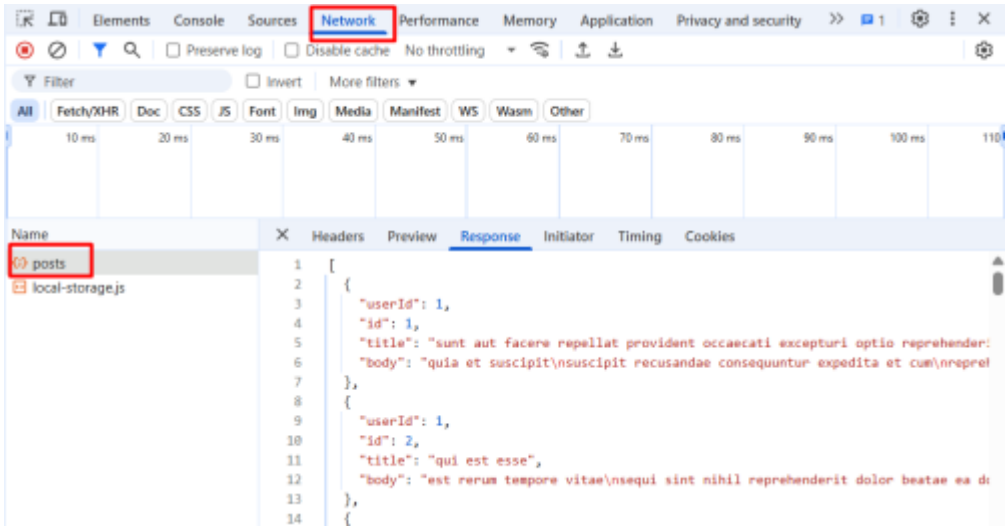
Míg a programkönyvtárakat általában webalkalmazások fejlesztésére használjuk, addig a parancssori eszközök és a GUI-val rendelkező alkalmazások elsősorban tesztelési célokat szolgálnak, lehetővé téve a HTTP kérések gyors ellenőrzését és elemzését.

Google Chrome

A Chrome fejlesztői eszközeivel (DevTools) könnyen nyomon követhetők a HTTP kérések:

Gyors teszteléshez nyiss egy új lapot, nyomd le az F12 gombot, majd töltsd be a https://jsonplaceholder.typicode.com/posts URL-t!

A DevTools-ban válts át a Network fülre, majd keresd meg az indított kérést:



Az itt megjelenő eszköztár segítségével elemezhető a HTTP forgalom:

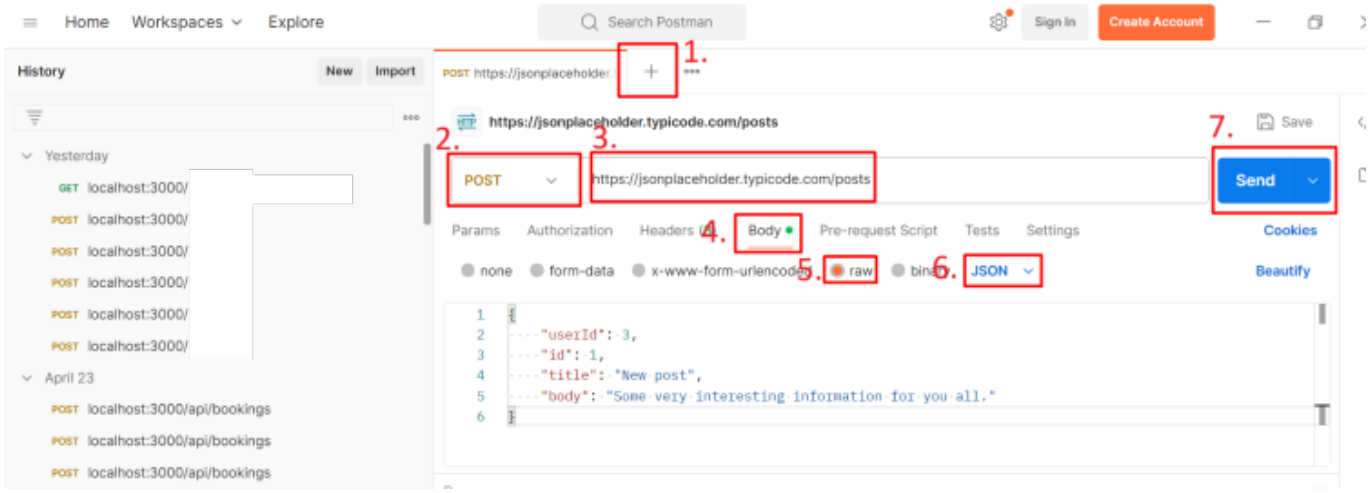
A legfontosabb funkciók:

- **Headers** fül: Kérés és válasz fejléceinek, valamint a válasz státuszkódjának a megtekintése.
- **Preview** fül: A válasz törzsének megtekintése, elemzése.
- **Response** fül: A válasz törzsének megtekintése nyers szövegként.

A DevTools segítségével a webalkalmazásunk által kezdeményezett HTTP kommunikáció is megtekinthető. Fontos azonban, hogy POST, PUT, és DELETE kéréseket közvetlenül csak programkódból tudunk küldeni. Ahhoz, hogy ilyen kéréseket gyorsan, manuálisan tudjunk előállítani, szükség lehet valamilyen külső alkalmazás (pl. Postman) használatára.

Postman

A Postman egy API tesztelő alkalmazás, amellyel HTTP kéréseket lehet küldeni és válaszokat elemezni. Tegyük fel, hogy a korábban megtekintett JSONPlaceholder API segítségével egy új bejegyzést szeretnénk létrehozni. Ehhez egy POST kérést kell küldenünk a `/posts` útvonalra.



Ennek menete:

1. Nyissunk új lapot a Postman-ben!
2. Állítsuk be a POST HTTP metódust!
3. Állítsuk be a szükséges útvonalat: `https://jsonplaceholder.typicode.com/posts`
4. A kérés törzsének beállításához váltsunk a Body fülre!
5. A kérés formátumának beállításához válasszuk a **raw** opciót!
6. Majd válasszuk ki a JSON formátumot!
7. A kérés törzsét állítsuk be a következő JSON objektumra, majd nyomjuk meg a **Send** gombot:

```
{
    "userId": 3,
    "id": 1,
    "title": "New post",
    "body": "Some very interesting information for you all."
}
```

A szerver válasza az ablak alján jelenik meg. A lenti screenshoton az látható, hogy a szerver **201 Created** státuszkóddal nyugtázta az új bejegyzés létrehozását, a válasz törzsében pedig a létrehozott bejegyzést küldte vissza. A további fülek segítségével megtekinthetők még egyebek mellett a válasz fejlécei, valamint a válaszban szereplő cookie-k is:

HTTP kérések feldolgozása JavaScript segítségével

A kész webalkalmazásban természetesen JavaScript kód segítségével szükséges összeállítani a szerver felé küldött HTTP kéréseket. Erre a célra egyebek mellett a Fetch API használható.

A Fetch API a kéréseket aszinkron módon, Promise-ok segítségével kezeli, emiatt speciális szintaktikát kell használnunk! A `then` metódusnak a sikeres kérés esetén lefutó függvényt, míg a `catch` metódusnak a hibakezelő függvényt szükséges paraméterként átadni.

A következő kód az 1-es azonosítójú bejegyzés tartalmát kérdezi le a szerverről:

```
fetch("https://jsonplaceholder.typicode.com/posts/1")
  .then(response => response.json())
  .then(data => console.log(data))
```

```
  .catch(error => console.error("Hiba:", error));
```

A szerver válaszát először JSON objektummá (bizonyos esetekben egyszerű szöveggé) szükséges konvertálni (2. sor), ezt követően elérhetők a válaszban szereplő adatok. A 3. sorban ezeket a konzolra íratjuk, de ehelyett akár a felhasználói felületen is megjeleníthetnénk, a DOM módosításával.

A hibakezelés a `catch` metódusban történik. Hiba akkor fordulhat elő, ha a szerver nem küld választ, vagy a válasz nem JSON formátumú.

Előző kérésünk egyszerű GET kérés volt, mely nem tartalmazott komplex adatokat. A következő kód POST kérés küldését mutatja be, új bejegyzés létrehozásához. Ebben az esetben a kérés `Content-Type` fejlécét is szükséges beállítanunk, jelezve a szervernek, hogy JSON formátumban küldjük a kérés törzsét. A törzset egyszerű szövegként szükséges elküldenünk, így a JavaScript objekumunkból a `JSON.stringify` függvény meghívásával JSON sztringet szükséges generálnunk:

```
fetch("https://jsonplaceholder.typicode.com/posts", {
  method: "POST",
  headers: {
    "Content-Type": "application/json"
  },
  body: JSON.stringify({ title: "Teszt", body: "Ez egy teszt.", userId: 1 })
})
  .then(response => response.json())
  .then(data => console.log(data))
  .catch(error => console.error("Hiba:", error));
```

A válasz kezelése ebben az esetben is a korábban ismertetett módon történik.