# RESTful webszolgáltatások és API tesztelés Postmannel

## Webes alkalmazások

* Böngésző a kliens, mely a felhasználói felületet jeleníti meg
  + Microsoft Internet Explorer, Microsoft Edge
  + Mozilla Firefox
  + Google Chrome
* Mivel böngésző mindenütt telepített, megszűnnek a telepítési/frissítési problémák
* Vékony kliensnek is nevezik, utalva arra, hogy a logika főleg szerver oldalon van
  + Elegendő csak ott frissíteni
* Háttérben a HTTP, HTML, CSS, JavaScript technológiák

## URL

* Uniform Resource Locator
* Interneten található erőforrások (szöveges tartalmak, képek, hangfájlok, videók) egyedi azonosítására
* Felépítése
  + Protokoll
  + Tartománynév/ip-cím
  + Port
  + Elérési út

## HTTP(S) protokoll

* 1999-ben kiadott RFC 2616 definiálja a HTTP/1.1-et (W3C szervezet)
* 2015-ben leváltott a HTTP/2.0-ás verzió, amit az RFC 7540 definiál
* Kliens-szerver kommunikáció
* Kliens tipikusan böngésző
* Kérés-válasz alapú protokoll
* Szöveges
* Fejléc, törzs
* Állapotmentes
* (S): secure - SSL/TLS alapú titkosítás
* Eszközök: Böngésző Fejlesztői eszköztár (F12 - DevTools)

## HTTP kérés

GET / HTTP/1.1  
Host: www.pontsystems.eu  
User-Agent: curl/7.58.0  
Accept: \*/\*

## HTTP válasz

HTTP/1.1 200 OK  
Date: Tue, 10 Sep 2019 20:32:24 GMT  
Server: Apache  
X-Powered-By: PHP/5.5.9-1ubuntu4.20  
Vary: Accept-Encoding  
Content-Type: text/html  
Transfer-Encoding: chunked  
  
<!DOCTYPE html>  
<html lang="en">  
 <head>  
 <!-- ... -->  
 </head>  
 <body>  
 <!-- ... -->  
 </body>  
</html>

## Hivatkozott erőforrások

* CSS
* JavaScript
* Képek (tipikusan gif, jpg, svg formátumban)
  + Formátumok közötti különbségek

## Státuszkódok

* 200 OK
* 302 Found
* 304 Not Modified
* 400 Bad Request
* 401 Unauthorized
* 403 Forbidden
* 404 Not Found
* 500 Internal Server Error

## HTTP paraméterek

* Kérdőjellel (?) elválasztva
* Kulcs és érték párok
* Egymástól & karakterrel elválasztva
* URL encoding

http://example.com/search?name=John&from=2010-01-01

## Mime-type

* Content-Type header
* Néhány: text/plain, text/html, text/javascript, text/CSS, image/gif, image/jpeg, application/vnd.ms-excel (vnd - vendor - nem szabványos)

## Űrlapok

* POST metódus

POST / HTTP/1.1  
Host: www.pontsystems.eu  
User-Agent: curl/7.58.0  
Accept: \*/\*  
  
cod=hu&sub=inq&name=John Doe

## JSON

* JavaScript Object Notation
* XML leváltására, JavaScript elterjedésével, adatok átvitelére
* Tipikusan böngészős frontend és backend közötti kommunikációra
* Ember és számítógép számára értelmezhető szöveges, kiterjeszthető formátum
  + Érvényes JavaScript kifejezés
* Kulcs-érték párokat (object) és tömböket (array) tartalmaz
* Adattípusok: szám, karakterlánc, logikai
* Speciális üres érték: null
* Fa hierarchia

## Példa JSON dokumentum

[  
 {  
 "id": 1,  
 "name": "Budapest",  
 "lat": 47.497912,  
 "lon": 19.040235,  
 "interestingAt": "2019-01-01T05:00:00",  
 "tags": [  
 "capital",  
 "favourite"  
 ]  
 },  
 {  
 "id": 2,  
 "name": "Debrecen",  
 "lat": 47.5316049,  
 "lon": 21.6273124,  
 "tags": []  
 }  
]

## JSON értelmezése JavaScriptből

data[0]['name']  
  
data[0]['tages'][1]

## AJAX

* Asynchronous JavaScript and XML
* Technológia a HTML dokumentum (DOM fa) módosítására, lap váltása nélkül, szerver oldali adatok alapján
* Megnövelt felhasználói élmény
* Kisebb adatforgalom, csökkenő szerver terhelés
* XML helyett már JSON
* Aszinkron: callback akkor hívódik meg, ha megérkezett a válasz a szervertől

## Webszolgáltatások

* W3C definíció: hálózaton keresztüli gép-gép együttműködést támogató szoftverrendszer
* Platform független
* Szereplők
  + Szolgáltatást nyújtó
  + Szolgáltatást használni kívánó

## RESTful webszolgáltatás

* Roy Fielding: Architectural Styles and the Design of Network-based Software Architectures, 2000
* Representational state transfer
* Egyedileg címezhető erőforrások (resource)
* Uniform, constrained interface for manipulate resources (crud)
* Létező technológiák: URI, HTTP, XML, JSON
* Web Application Description Language (WADL) – nem elterjedt
* AJAX világ segítette az elterjedését

## További HTTP metódusok

* PUT - feltölti a megadott erőforrást, POST működéséhez hasonlatos
* DELETE - törli a megadott erőforrást

## Postman

* A Postman egy eszköz és szolgáltatáscsomag API fejlesztéshez
* https://www.getpostman.com/
* Árazás
  + Postman: ingyenes
  + Postman Pro: kereskedelmi termék
    - https://www.getpostman.com/pricing
    - Magasabb limitek a szolgáltatások használatakor (pl. megosztás)
* Multiplatform: Windows, macOS, Linux

## Teljes életciklus

* Magas szintű csoportmunka (nem csak szoftver, hanem szolgáltatás)
* Tervezés és mock (szimuláció)
* Hibakeresés
* Manuális és automatizált tesztelés
* Dokumentálás
* Monitorozás

## Telepítés

* Natív alkalmazás
* Telepítőkészlet

## Első kérés elküldése - GET (gyakorlat)

* http://www.learnwebservices.com/locations/api/locations
* http://www.learnwebservices.com/locations/api/locations/1

## Második kérés - POST (gyakorlat)

* /api/locations
* Body: raw, JSON (application/json)

{  
 "name":"Budapest",  
 "coords":"47.497912,19.040235",   
 "interestingAt": "2019-01-01T05:00:00",   
 "tags": "capital,favourite"  
}

## Hibakezelés (gyakorlat)

* Helytelen JSON dokumentum

## Validáció (gyakorlat)

* Helyes JSON, üzletileg helytelen adatmezők

## Lapozás (gyakorlat)

* page és size paraméterek

## Egymásba ágyazott objektumok (gyakorlat)

* tags kezelése

## Dátum kezelése (gyakorlat)

* 2019-01-01T05:00:00 formátumban

## Example

* Example(s)
* Save example
* Megjelenik a dokumentációban

## Postman Echo

* Példa REST webszolgáltatások pl. eszközök tesztelésére
* https://docs.postman-echo.com/

## Postman Echo kérések (gyakorlat)

* GET
* POST

## History, Collection

* History: kisebb projektek esetén
* Collection: legmagasabb szintű szervezésre
  + Benne mappák

## Collection létrehozása (gyakorlat)

* Locations collection

## Postman felépítése

* Header
* Sidebar
* Builder
* Console (View / Show Postman Console v. Status bar)
* Status bar

## Workspace

* Felhőbe szinkronizálás: Collections, Requests, History, stb. egy nézete
* Pl. projektenként
* Közös munkára: team workspace

## Environments

* Különböző környezetek
* Variable:
  + Initial value (megosztott)
  + Current value (csak lokálisan)

## Environments (gyakorlat)

* Saját környezet telepítése
* tadev környezet
* localhost környezet
* url változó létrehozása

## Tesztesetek

* JavaScript
* Tests tab (Assert és After fixture)
* Code snippets
* Test Results tab
* Pre-request Script (Before fixture)
* Collection szinten is
* Postman Sandboxon belül futnak

## Assert

* Status code
* Response body: contains string
* Response body: JSON response check

## Teszteset (gyakorlat)

* Assert

## Változók scope-ja

* Global
* Collection
* Environment
* Data (külső adatforrásból importált)
* Local
* Fentről lefele erősebb prioritás

## Változók hozzáférése

* Builder: url, paraméter, header, body
* Kódból

pm.global.set("username", "John Doe");  
pm.global.get("username");

## Collection run

* Több lépésből álló tesztesetet külön collectionbe
* Runner (Collection Runner)

## Futtatás parancssorban

* Node.js
* npm install -g newman
* Export collection
* newman run Testapp.postman\_collection.json

## Tesztelés (gyakorlat)

* Új kedvenc hely felvétele
* Kedvenc hely módosítása
* Kedvenc hely törlése

## Data

* Collection Runner / Data
* Iteration
* CSV formátum

## Data (gyakorlat)

* https://mockaroo.com/

locations.csv:

name,lat,lon  
Budapest,47.497912,19.040235

## Verziókezelés

* Fork/merge

## Postman account

* Felhőbe szinkronizálás: Collections, Requests, History, stb.

## Postman account (gyakorlat)

* Saját account készítése
* Team workspace készítése
* Egymás meghívása

## Dokumentálás

* Markdown formátum
* Collection
* Request
* Collection / View in web

## Saját dokumentáció (gyakorlat)

* Collection, Request szinten

## Mock server

## Monitoring

## Ismétlő kérdések

* Mire való a Postman?
* Milyen verziói vannak, milyen különbség van közöttük?
* A szoftverfejlesztési életciklus mely lépéseit támogatja?
* Hogyan kell egy kérést megfogalmazni?
* Mibe lehet szervezni a kéréseket?
* Hogyan épül fel a felhasználói felület?
* Mi az a Workspace?
* Milyen csoportmunka eszközöket ismersz?

## Ismétlő kérdések 2.

* Hogyan lehet Postmannel dokumentálni az API-t?
* Hogyan támogatja a Postman a környezeteket?
* Hogyan írhatsz asserteket?
* Hogyan hozhatsz létre több lépésből álló teszteseteket?
* Hogyan lehet parancssorban futtatni a teszteseteket?
* Hogyan lehet változót deklarálni és használni?
* Hogyan lehet adatvezérelt tesztelést megvalósítani?