class: inverse, center, middle

# API tesztelés Postmannel

## 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

## HTTP-ből újrafelhasznált tulajdonságok

* Kliens-szerver, kérés-válasz
* URI
* HTTP metódusok
* Hibakezelés, státuszkódok
* Header
* Content negotiation, mime-type
* Cache

## RESTful webszolgáltatásokat hívó frontend tipikus működése

* Letöltésre kerül a statikus HTML oldal, CSS és JavaScript állományok
* A HTML dokumentumot a CSS alapján lerendereli a böngésző, előáll a DOM
* A böngészőben lévő JavaScript motor (engine) futtatja a JavaScript kódot
* AJAX-szal megtörténik a (GET) hívás a szerver RESTful webszolgáltatása felé
* A szerver visszaadja a választ JSON formátumban
* Meghívásra kerül a callback függvény
* A callback függvény értelmezi a JSON formátumú adatot, és módosítja a DOM fát, ezáltal a HTML oldalon megjelennek az adatok

## RESTful módosítás

* Az AJAX hívás paraméterként egy JSON dokumentumot küld fel
* A szerver oldali RESTful webszolgáltatás értelmezi a JSON adatokat, és hatására adatot szúr be, módosít vagy töröl az adatbázisban
* Amennyiben sikeres a művelet, erről a klienst értesíti
* A callback függvény tájékoztatja a felhasználót a művelet sikerességéről

## 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://tadev.training360.com/testapp-rest
* /api/test
* /api/result

## Második kérés - POST (gyakorlat)

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

{  
 "id": 1,  
 "question": "Mi az a REST API?",  
 "answers": [  
 {  
 "text": "HTTP-re épülő webszolgáltatások",  
 "point": 1  
 },  
 {  
 "text": "Teszteszköz",  
 "point": 0  
 }  
 ],  
 "maxPoint": 1  
}

## Második kérés - POST (gyakorlat)

* /api/result

{  
 "user": {  
 "userName": "John Doe"  
 },  
 "startTime": "2019-05-01T19:11:40.470Z",  
 "endTime": "2019-05-01T19:17:29.560Z",  
 "result": 1,  
 "maxPoint": 2  
}

## 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)

* testapp collection

## Postman felépítése

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

## Postman account

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

## Workspace

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

## 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

## 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

## Collection run

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

## Tesztelés (gyakorlat)

* Kérdések törlése
* Új kérdés felvétele
* Kérdések lekérdezése
* Eredmény felvétele
* Eredmények lekérdezése

## Futtatás parancssorban

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

## Változók scope-ja

* Global
* Collection
* Environment
* Data
* Local

## Változók hozzáférése

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

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

## Data

* Collection Runner / Data
* Iteration
* CSV formátum

## Data (gyakorlat)

names.csv:

username  
John  
Jane  
Jack

## 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?