**REST webszolgáltatások POST és DELETE művelet**

Erőforrás felvitel – **POST** művelet

Erőforrás módosítás – **PUT** művelet

A **PUT idempotens**

**idempotens:** az adott művelet többszöri elvégzésre is ugyanazt az eredményt adja vissza; nincs további hatása, ha többször is meghívják ugyanazokkal a bemeneti paraméterekkel.

**pl. átutalás** mégegyszer ne történjen meg -> használjunk PUTot, és az adatbázisban csak egy módosítás fog történni a többszöri futás ellenére

**POST-nál a művelet többszöri meghívása annak többször elvégzését is jelenti**

a böngészők a POST művelet ismerik, ezért kérdeznek rá újraküldés esetén, hogy biztos?

PUT haszna jelentkezik hálózati kapcsolat szakadása esetén nem tudjuk, hogy végrehajtódott a művelet, ha újra is küldjük be, nem fog végrehajtódni

**Controller POST művelettel**

*Mi a különbség, ha create vagy update műveletet akarunk végezni? maga az adat tartalom fogja meghatározni:*

küldünk id-t -> módosítás lesz (update)

nem küldünk id-t -> új felvitel lesz új erőforrás (create)



mindkét metódus az adott erőforrás aktuális állapotával, DTO-val tér vissza

**azért, hogy adatot lehessen beküldeni a http request törzsébe a @RequestBodyt használunk a command paraméter előtt; a http válaszba kiírva a DTO jSON formátumban lesz**

**Controller Delete művelettel**

**@DeleteMapping** használatával a metódus csak a DELETE http metódussal lesz elérhető; paramétere egy azonosító, amit PathVaraibleként adunk át; ilyen metódusnál nem adunk vissza semmit

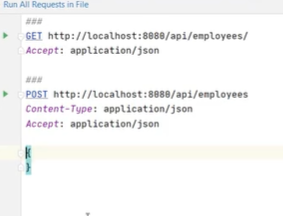
**REST webszolgáltatások POST és DELETE művelet - gyakorlat – Create**

*Úgy módosítom az alkalmazást, hogy fel lehessen venni egy alkalmazottat; ehhez egy DTO-t hozok létre; ez egy command lesz, mert módosítja az alkalmazás állapotát*

1. új osztály: CreateEmployeeCommand, String name; @Data
2. EmployeesControllerben új metódus: EmployeeDao createEmployee(CreateEmployeecommand command) -> kérés delegálása a servicenek; @PostMapping rája: ezt a metódust hívja meg a Spring, amikor a az api/empoyees/ címen (@RequestMapping definiálta az osztály előtt!) egy post kérés jön be; a Springnek a bodyban beérkező adatokból kell ezt a commandot létrehoznia -a paramétert el kell látni a @Requestbodyval
3. serviceben metódus létrehozása; azonosítót már nekem kell majd kiosztanom(incrementálnom); ehhez kell egy AtomicLong (szálbiztos)típusú idGenerator; a syncronized employeelistánál átírom az idket idGenerator.incrementAndGet()re; a createEmployyenál ugyanezt átírom és a nevét is, command.getName()re módosítom, és empoyees.add és return modelmapper.map(…
4. post metódusú kéréseket böngészővle nem tudok kiadni; de erre biztosít eszközt az IDEA; new -> httpRequest -> Employees; egyszerű szöveges állomány, amiben http kéréseket lehet felvenni; AdRequest ->GetRequest



*aztán hozzáraktunk egy postot; json formátumban adtuk meg az adatot!(name..):*

**

**Kérdések**

*Mire jó alapesetben a POST és a PUT művelet?*

*Mit jelent az, hogy a PUT művelet idempotens? Mondj rá példát!*

*Ezzel szemben a POST műveletnél mi történik többszöri végrehajtásra?*

*PL. milyen esetben lehetséges, hogy tévesen újraküldjük a requestet és miért hasznos ilyenkor a PUT?*

*Milyen annotációkkal konfiguráljuk a javas metódusainkat ha azt akarjuk, hogy POST és PUT http metódussal legyenek elérhetőek? Hogy állítjuk be, hogy a metódusok milyen URLen figyeljenek?*

*Az id küldés vagy nem küldés vonatkozásában mikor akarunk create és update műveletet végezni?*

*Mit tesz lehetővé a @RequestBody annotáció (2) és mivel együtt használjuk? (mi elé tesszük)*

*Mik a PUT és POST metódussal elérhető javas metódusok visszatérési értékei?*

*Mit jelent ha a metódust @DeleteMapping annotációvla látjuk el?*

*Hogy adjuk át ilyenkor, hogy melyik erőforrást akarjuk törölni?*

*Jellemzően mi az ilyen metódus visszatérési értéke?*

*Hogy küldünk be az adott url-re http metódusokat fejlesztő eszköz segítségével?*