**Státuszkódok és hibakezelés**

**(ha ok, @ResponseStatussal specifikusabb kódot, ha nem ok, ResponseEntitybe csomagolt problem -> hiba esetén szabványos json vissza; problem build majd ResponseEntity build)**

Ha egy művelet **sikeresen végrehajtásra** kerül, akkor **http 200-as státuszkóddal** térünk vissza;

viszont, **ha hiba keletkezik** az miből fakadhat és milyen kóddal kell visszatérni?

*Példák hibás működésre:*

A kliens nem küldött megfelelő adatot (pl. formátum nem megfelelő, vagy kötelező helyre üres értéket küldött)

a felhasználó nincs autentikálva, nincs bejelentkezve

server oldalon nem elérhető az adatbázis

**Kódok**

*200as kódok - ok*

*300as kódok*

alapvetően jó volt a kérés, de még valami hiányzik kliens oldalon, pl. át kell irányítani a kliens vagy be kell jelentkeztetni

*400as kódok*

valami rossz adatot küldött a kliens

*500as kódok*

server oldali probléma, ezzel a kliens nem tud mit tenni

**Hogyan lehet Controllerből ilyen státusz kódokat visszaadni?**

**(ResponseEntity fluent Api-val, try-cath és ResponseEntity.ok(metódushívás )=200 és ResponseEntity.notFound.build()=404; külön a metódusban implementálva)**

**amennyiben nem elég a 200as, ResponseEntityt kell visszaadni, nem pedig Dto-t**

**fluent builder APIval** lehet létrehozni ResponseEntity példányokat

ResponseEntity.ok(XXX) – 200ast választ hozunk létre, értékül a bodyba a service által visszaadott employeet állítjuk be

amennyiben nem találja az idt, catch ágon Illegalt kapunk el, ResponseEntity.notfound-al 404es hibát adunk vissza a kliensnek



**Hogyan lehet standard módon mindig ugyanazt a státuszkódot visszaadni?**

**(@ResponseStatus(HttpStatus.CREATED)=201 és @ResponseStatus(HttpStatus.No\_CONTENT)=204)**

ha nem hibás az adat, de legyen más a kód, mint a default(legyen specifikusabb)

**@ResponseStatus** annotációval

@PostMapping -> **@ResponseStatus(HttpStatus.CREATED**) -> 201est ad vissza nem 200ast, aminek jelentése: az erőforrás a server oldalon létrehozásra került

@DeleteMapping -> **@ResponseStatus(HttpStatus.No\_CONTENT)** -> 204est ad vissza nem 200ast, aminek jelentése: nem hiba miatt üres a response törzse, hanem mi akartuk így

**Hogyan történik a hibakezelés?**

Ha eldobásra kerül ez exception, azt a Spring MVC elkapja, és valami hasonló jSONt küld vissza a kliens oldalra 400as státusz kóddal



**Hibakezelés**

*Spring MVCben ha mi akarjuk kezelni a hibát, milyen lehetőségeink vannak?*

servlet szabvány szerint **web.xml** fileban lehet **deklarálni**, hogy hanyas státuszkódú hibáknál, milyen exceptionöknél milyen másik oldalra történjen az átirányítás

Springben **saját exceptionre** rá lehet tenni **a @ResponseStatus** annotációt

Globálisan lehet **ExceptionResolver osztályokat** definiálni

Egy adott **controlleren belül lokálisan** is lehet kivételeket kezelni az **@ExceptionHandlerrel**

globálisan is fel lehet venni **@ControllerAdvice-al** olyan osztályokat, amelyben @ExceptionHandler annotációval ellátott hibakezelő eljárásokat lehet definiálni és ezt a controller osztályhoz hozzá lehet rendelni

**Egy ExceptionHandler metódust hogyan kell megvalósítani?**

**(handler metódus + @ExceptionHandler + @ResponseStatus -> a controller összes metódusára érvényes lesz)**

**@ExceptionHandler** annotációt rakunk a **handler metódusra** és paraméterként megadjuk, hogy milyen exceptiont vár; amelyik **controllerbe** ezt a metódust így beletesszük, ha **bármelyik metódusában** egy illegalarg kerül kiváltásra, akkor a spring mvc ezt elkapja és ezt a handlert hívja meg és a **@ResponseStatus szerint deklarált státuszkóddal**(itt 404=NOT\_FOUND) fog visszatérni



**Szabvány a hibakezelésre – RFC 7807**

**(szolgáltatók respponsejában legyen egységes a json a hibaüzire -> RFC szabvány -> type, title, status, detail + egyedi saját mezők)**

**minden szolgáltatónál más volt a jSON amit visszakaptunk**, ha hiba történt, ezért kellett egy szabvány; ez lett az RFC 7807

Ha valami hiba van**, a választ egy application/problem+json mime-type-al kell visszaadni** és a következő mezőkkel kell rendelkeznie:

type, title, status, detail + egyedi saját mezők



type: egyedi azonosító a hiba típusával; mi magunk deklarálhatjuk

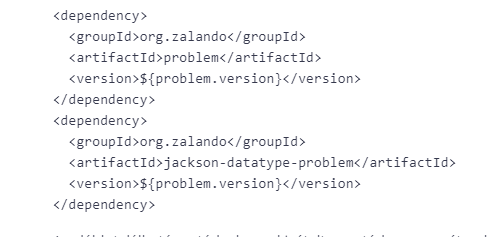
opcionálisan be lehet még tenni egy instance-t, ami a hibának az elérhetőségét tartalmazza, pl egy naplózó rendszerben egy url-t

ezen kívül még **egyedi saját mezők definiálhatók**, mégtöbb info a hibákról a kliensnek

**Problem**

**RFC 7807 szabvány java-s 3rd party implementációja, pom.xmf függőség, problem buildelés fluent APival(.WithType, .withTitle, .withStatus, .withDetail) majd problem becsomagloása ResponseEntitybe(.status, .body), ; Jacksonos Objectmapper példány létrehozása @Beannel, ami megfelelően fogja jSONbe kiírni a http responseba(=Jackson inicializálás)**

Java-ban több **implementációja** van ennek a **szabványnak**, pl. a **Problem nevű third party library,** amihez pom xml függőség felvétele kell:

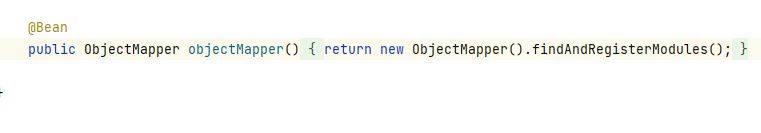


*Majd a hibát a következőképpen kezelhetjük:*



**Builder API-val fluent** módon létrehozunk egy **problem példányt**; végül ezzel a problemmel egy **ResponseEntitybe becsomagolva** térünk vissza

Ahhoz, hogy a problem működjön, fontos hogy majd megfelelően tudja az alkalmazás jSONbe kiírni; ehhez fel kell venni a jSON szerializálásnak, hogy ezt olvassa föl: ehhez egy @Configuraiton-el ellátott osztályban **@Bean-el egy objectMappert** kell definiálni és ennek meghívni a **findAndRegisterModules()** metódusát; ez észre fogja venni, hogy a classpathon ott van a Problem library és **a problem példányt megfelelően fogja jSONbe kiírni a http responseba(=Jackson inicializálás)**



**Státuszkódok és hibakezelés – gyakorlat**

1.új http kérés; azonosító alapján alkalmazott lekérés és lekérek egy nem létezőt -> látom, hogy milyen jSON-nel és 500as kóddal tér vissza; módosítsuk úgy, hogy 404es legyen

megfelelő metódus visszatérési típusa legyen ResponseEntity; try-> ResponseEntity okba csomagolva a metódushívás; catch -> illegalarg és ResponseEntity.not found build,

2.módosítsunk úgy, hogy létrehozáskor a kód http status created legyen -> createEmployee metódus és ResponseStatus(HttpStatus.CREATED) ->201est kaptunk vissza

3. törléskor legyen ResponseStatus(HttpStatus.NO\_CONTENT) ->204es kód lesz

4. több legyen is lehet, hogy nem találok adatot az adott id-val; ezért nem lokálisan minden metódusnál cserélem a státuszkódot, hanem globálisan írok rá egy handlert ->

void handleNotFound(); @ExceptionHandler(IllegalArumentException.class); érdemes erre saját exceptiont! kell még ide: ResponseStatus(HttpStatus.NOT\_FOUND); csekkolás módosításra és lekérdezésre

5.resonsebody ne legyen üres; pom dependency problemre (2 kell!); Jackson inicializálás: @Bean és objectMapper példányosítás

handler metódus térjen vissza ResponseEntitiy<Problem>-el; problem létrehozása builderrel; illegalargot kapja meg a metódus paraméterként és térjünk vissza ResponseEntitiy-el amit fluent apivel építünk fel(.status, .contentType, .body; a content type application:problem\_json!)

**Kérdések**

*Alapesetbe egy sikeres http request milyen sttuszkóddal tér vissza?*

*Mondj server- és kliens oldali lehetséges hibákat!*

*Mit jelentenek a 200- 300- 400- és 500-as kódok?*

*Milyen osztályt kelll használni, hogy az alapvetőtől eltérő státuszkódot adjunk vissza Controllerből?*

*Hogyan lehet standard módon mindig ugyanazt a státuszkódot visszaadni, abban az esetben, ha nincs hiba?*

*Mi történik a Springben hibakezeléskor default működés esetén?*

*Ha mi akarjuk kezelni a hibát, mit kell használni?*

*Egy ExceptionHandler metódust hogyan kell annotációkkal konfigurálni?*

*Milyen szabvány van a hibakezelésre és miért volt szükséges bevezetni?*

*Milyen javas 3rd party implementációja van ennek a szabványnak és hogy implementáljuk az exceptionhandler osztályunkba? Mi lesz a visszatérési értéke a handler metódusnak?*

*Ahhoz, hogy megfelelően tudja az alkalmazás jSONbe kiírni a problemet, mit és hol kell definiálnunk?*