

JAVA SPRING MICROSERVICES

5. forduló



A kategória támogatója: Vodafone Magyarország
Zrt.

Ismertető a feladathoz

Azért véletlenül előfordulhat, hogy dinoszauruszt is látunk? – HÁLÓ?! – Dr. Ian Malcolm

Sok külsős kliens felfigyel az InGen Sol-ra és érkeznek is az e-mailek "hivatalos" integrációs pontokról, kíváncsiak, hogy az eddig felépített projektet külső felek is használhatják-e vagy sem. Sok-sok kérés érkezik egy publikus API iránt és a Ti csapatotok is érzi, hogy bizonyos szolgáltatások vagy bizonyos adatok akár nyilvánosak is lehetnének az érdeklődők számára.

A legtöbb felkérés a dinoszauruszok helyzetének és adatainak a publikussá tételéről szól. Ötletesebbnél ötletesebb designerek jelennek meg, hogy már csak egy API-ra várnak, hogy az Ő oldalukon követhessük a kedvenc dinoszauruszaunkat, csak az adatok hiányoznak.

Mire lesz szükséged a fordulóban?

- Fejlesztői környezet (ajánlott)

Ha egy adott feladat egy annotációt vár válaszként, akkor oda csak az annotáció nevét add meg @ előjellel, nem szükséges teljes minősített nevet megadni, például: @NotNull.

Felhasznált idő: 00:00/25:00

Elért pontszám: 0/12

1. feladat 0/2 pont

A következő REST API interfész módosítási lehetőségek közül melyik töri meg a visszafelé kompatibilitást?

Válaszok

- ☐ Új attribútum hozzáadása a válaszbba
- ☒ Új kötelező attribútum hozzáadása a kérésbe
Ez a válasz helyes, de nem jelölted meg.
- ☒ Egy operáció elérési útjának megváltoztatása
Ez a válasz helyes, de nem jelölted meg.
- ☐ Új operáció hozzáadása

Magyarázat

Interfészek tervezésekor figyelnie kell a fejlesztőknek arra, hogy a megfelelő be és kimeneti típusokról gondoskodjanak, és ha a lehetőség és szándék adott akkor az interfészek operációit verziózzák, ezzel biztosítva, hogy jövőbeli változások amelyek

visszafelé kompatibilitási problémát okoznának, egy újabb verzióval kerüljenek az interfészre és a meglévő klienseket amelyek az interfész régebbi operációin dependálnak ne törjenek meg.

Egy meglévő operáció kérésébe egy kötelező attribútumot beépíteni vagy az operáció elérési útjának (REST esetén URL-jének) megváltoztatása megtöri a kliensek kommunikációját.

A másik kettő lehetőség úgymond biztonságos, ezekben az esetekben a kliensnek nem lenne szabad megtörnie.

2. feladat 0/1 pont

Csapatotok azt a feladatot kapta, hogy a szolgáltatást amely a dinoszauruszokat kezeli, dokumentálja le a végpontokat és a hozzá tartozó objektumokat példákkal együtt.

Erre a feladatra az OpenAPI 3.0 specifikációt szeretné a csapat használni, a kódban pedig annotációk segítségével szeretnék elérni a végső megoldást.

A következő feladatokban Java interfészeket és osztályokat fogsz találni, a megfelelő annotációt keressük a következő könyvtárból:

- <https://springdoc.org/> amely az OpenAPI v3 specifikációt használja.
- io.swagger.core.v3:swagger-annotations
- io.swagger.core.v3:swagger-models

```
@RequestMapping(path = "/dinosaurs")
interface DinosaurController {

    @PostMapping
    DinosaurResponse postDinosaur(
        //1
        CreateDinosaurRequest dinosaurRequest,
        // 2
        String apiKey
    );

    @GetMapping
    DinosaurListResponse getDinosaurs(
        // 3
        GetDinosaursRequest getDinosaursRequest
    );
}
```

Melyik annotációt kell írunk a //1-es kommentel jelölt helyre, ahhoz, hogy az OpenAPI generálás folyamán mint bemeneti objektum szerepeljen? (Kérlek a válaszodba írd be a @ jelet is!)

Válaszok

A helyes válasz:

@RequestBody

@Parameter

@Parameter()

Magyarázat

A **@RequestBody** vagy a **@Parameter** nevű annotációt kell használnunk az OpenAPI függőségek közül, hogy az OpenAPI dokumentációban **requestBody**-ként jelenjen meg.

3. feladat 0/1 pont

Melyik annotációt írhatjuk a //2-es kommentel jelölt helyre, ahhoz, hogy az OpenAPI generálás folyamán mint bemeneti HTTP fejléc jöjjön létre?

Válasz

- ☐ @Parameter(type = ParameterType.HEADER)
- ☒ @Parameter(in = ParameterIn.HEADER)
Ez a válasz helyes, de nem jelölted meg.
- ☐ @Schema(format = SchemaFormat.HEADER)
- ☐ @Header

Magyarázat

A **@Parameter(in = ParameterIn.HEADER)** annotáció használatával tudunk paraméter alapú bemeneti HTTP fejléctet generálni. Az utolsó **@Header** annotáció, csak másik annotáción belül használható. A másik kettő pedig helytelen.

4. feladat 0/1 pont

Melyik annotációt kell írunk a //3-as kommentel jelölt helyre, ahhoz, hogy az OpenAPI generálás folyamán mint paramétereket tároló objektum szerepeljen? (Kérlek a válaszodba írd be a @ jelet is!)

Válaszok

A helyes válasz:

@ParameterObject
@ParameterObject()
@Parameter
@Parameter()

Magyarázat

A `@ParameterObject` nevű annotációt kell használnunk, hogy a megadott osztályt mint paramétereket tartalmazó bemenetként ismerje fel.

5. feladat 0/1 pont

```
class GetDinosaursRequest {  
  
    //1  
    private String type;  
  
    //2  
    private Double minWeight;  
  
    //3  
    private Double maxWeight;  
  
    //4  
    private Double minHeight;  
  
    //5  
    private List<String> dangerLevel;  
  
    //... konstruktorok, getterek, setterek  
}
```

Melyik annotációt írhatjuk a //1-es kommentel jelölt helyre, ahhoz, hogy az OpenAPI generálás a következő reguláris kifejezés kerüljön be a mezőhöz: `[A-Z]*`?

Válaszok

- ☐ `@Parameter(pattern = "[A-Z]*")`
- ☒ `@Schema(pattern = "[A-Z]*")`
Ez a válasz helyes, de nem jelölted meg.
- ☒ `@Pattern(regexp = "[A-Z]*")`
Ez a válasz helyes, de nem jelölted meg.
- ☐ `@RegExp("[A-Z]*")`

Magyarázat

A `@Parameter` annotáció nem rendelkezik `pattern` attribútummal, a `@RegExp` annotáció pedig nem létezik, a másik kettővel sikeresen tudjuk teljesíteni a követelményt.

6. feladat 0/1 pont

Melyik annotációt írhatjuk a //2-es kommentel jelölt helyre, ahhoz, hogy az OpenAPI generálás folyamán a mező neve **min-weight** legyen?

Válaszok

☐ @QueryParam(name = "min-weight")

☒ @Schema(name = "min-weight")
Ez a válasz helyes, de nem jelölted meg.

☒ @Parameter(name = "min-weight")
Ez a válasz helyes, de nem jelölted meg.

Magyarázat

Mivel egy GET-es végponton használjuk a feladatban lévő osztályt, ott mint paraméterek jönnek majd be és nem mint kérés objektum (request body), így a @Parameter annotáció **name** nevű attribútumával tudjuk ezt megtenni. A másik kettő megoldás nem működik.

7. feladat 0/1 pont

Melyik annotációt írhatjuk a //3-as kommentel jelölt helyre, ahhoz, hogy az OpenAPI generálás folyamán a mező minimális értéke ne legyen kisebb 0-nál?

Válasz

☒ @Schema(minimum = "0")
Ez a válasz helyes, de nem jelölted meg.

☐ @Schema(minLength = 0)

☐ @Schema(minimum = 0)

☐ @Parameter(min = 0)

Magyarázat

A @Schema annotáció sok attribútummal rendelkezik amelyek megtevesztőek is lehetnek, a feladat teljesítéséhez a **minimum** attribútumot kell használni, ami pedig egy String értéket vár. A többi megoldás nem működik.

8. feladat 0/1 pont

Melyik annotációt írhatjuk a //4-as kommentel jelölt helyre, ahhoz, hogy az OpenAPI generálás folyamán az 1.5 legyen egy példa érték?

Válaszok

☒ @Schema(example = "1.5")
Ez a válasz helyes, de nem jelölted meg.

☐ @Schema(example = 1.5)

☒ @Parameter(example = "1.5")
Ez a válasz helyes, de nem jelölted meg.

☐ @ExampleObject(value = "1.5")

Magyarázat

A **@Schema** és a **@Parameter** annotáció rendelkezik `example` nevű attribútummal ami egy `String` értéket vár, így csak ezek lehetnek jók. Az **@ExampleObject** nevű annotáció nem rakható közvetlenül mezőre.

9. feladat 0/1 pont

Melyik annotációt írhatjuk a //5-as kommentel jelölt helyre, ahhoz, hogy az OpenAPI generálás folyamán a mező engedélyezett értékei a következők legyenek: **LOW, MEDIUM, HIGH**?

Válasz

☐ @Parameter(allowableValues = {"LOW", "MEDIUM", "HIGH"})

☒ @Schema(allowableValues = {"LOW", "MEDIUM", "HIGH"})

Ez a válasz helyes, de nem jelölted meg.

☐ @Values({"LOW", "MEDIUM", "HIGH"})

Magyarázat

Szebb esetben itt egy `ENUM`-ot kellene használnunk, akkor egyértelműbb lenne az engedélyezett értékek listája, itt viszont kézzel kell megadnunk, azt pedig a **@Schema** annotáció **allowableValues** nevű attribútumával tudjuk megadni. A másik kettő lehetőség nem működik.

10. feladat 0/1 pont

Az interfészbe belekerült egy végpont, ami a dinoszauruszok létrehozásáért felelős, viszont ezt a publikum számára semmiféleképpen nem szeretnénk elérhetővé tenni, ez egy belső művelet, amelyet csak megadott biztonsági körülmények között lehetséges meghívni.

```
//1
@RequestMapping(path = "/dinosaurs")
interface DinosaurController {

    @PostMapping
    //2
    DinosaurResponse postDinosaur(
        CreateDinosaurRequest dinosaurRequest,
        String apiKey
    );

    @GetMapping
    DinosaurListResponse getDinosaurs(
        GetDinosaursRequest getDinosaursRequest
    );
}
```

Melyik annotációt kell írunk a //1-es kommentel jelölt helyre, ahhoz, hogy az OpenAPI generálás folyamán egy biztonsági lehetőség létrejöjjön? (Kérlek a válaszodba írd be a @ jelet is!)

Válaszok

A helyes válasz:

`@SecurityScheme`
`@SecurityScheme()`

Magyarázat

A `@SecuritySchema` annotációval hozhatunk létre biztonsági sémákat, amelyeket később tudjuk alkalmazni különböző operációkra.

11. feladat 0/1 pont

Melyik annotációt írhatjuk a //2-es kommentel jelölt helyre, ahhoz, hogy az OpenAPI generálás folyamán, hogy az első kérdésben definiált biztonsági lehetőség által legyen védett a végpontunk?

Válaszok

☒ `@Operation(security = @SecurityRequirement(name = "ingen-sol-api-key"))`
 Ez a válasz helyes, de nem jelölted meg.

☒ **@SecurityRequirement(name = "ingen-sol-api-key")**
Ez a válasz helyes, de nem jelölted meg.

☐ **@SecurityScheme(name = "ingen-sol-api-key")**

☐ **@Server(variables = @ServerVariable(name = "ingen-sol-api-key", defaultValue = "X-API-KEY"))**

Magyarázat

A biztonsági követelményt kettő helyen lehet defininálni, vagy magán a metóduson a **@SecurityRequirement** annotáció használatával, vagy egy már meglévő **@Operation** annotációban a **security** attribútumban defininálhatjuk az előbb említett annotációval.



[Legfontosabb tudnivalók](#)

[Kapcsolat](#)

[Versenyszabályzat](#)

[Adatvédelem](#)

© 2023 Human Priority Kft.

KÉSZÍTETTE **cone**

Megjelenés

Világos