Inlämningsuppgift 2

Lager API

Patrik Nygren

Webbutveckling i Java

Lager API

Bakgrund

I denna inlämningsuppgift ska ni självständigt ta fram ett REST API (backendapplikation) i SpringBoot. Applikationen kommer presentera REST-endpoints för skapa, lista, editera och ta bort (CRUD) varor i ett lager. Applikationen testas med en REST-klient som Postman. Lagerdata/varor ska sparas i och hämtas från en MySQL instans.

Syfte

- Studenten ska få utökad kunskap om att använda grundläggande SpringBoot komponenter för att strukturera en REST-applikation (Controllers, Services, Repositories och Filters).
- Studenten ska få utökad kunskap om användning av JPA (ORM) för databas hantering.
- Studenten ska få utökad kunskap om användning av Dependency Injection i Spring.
- Studenten ska få utökad kunskap om skiktad arkitektur i en webbapplikation och DTOs.
- Studenten ska få utökad kunskap om användande av Exceptions i Spring för att returnera olika HTTP felkoder.

Specifikation

Ni ska bygga ett REST API som ger tillgång till en lagerdatabas. Databasen ska ha endast en tabell (products). En produkt har följande attribut:

- **id**: (ett numeriskt värde)
- **product_id**: (en text/sträng)
- **name**: (en text/sträng)
- **cost**: (ett numeriskt värde)
- **category**: (en text/sträng)

REST API:et ska ha följande endpoint som tar emot HTTP metoderna: GET, POST, PUT, DELETE. Om en fel uppstår i applikationen som exempelvis att en eftersökt produkt inte finns ska ett lämpligt HTTP svar skickas tillbaka till klienten (404 Not Found).

När ni startar er SpringBoot applikation ska applikationen ta emot HTTP requests på url:en http://localhost:8080/products.

Exempel på användning

Skapa en produkt
Request
POST http://localhost:8080/products
{ name: ”Schampo”, cost: 20, category: ”Hair” }

Response
201 CREATED
{ product_id: hs46jfh8012nhdh, name: ”Schampo”, cost: 20, category: ”Hair” }

Hämta alla produkter i databasen
Request
GET http://localhost:8080/products

Response
200 OK
[
{ product_id: hs46jfh8012nhdh, name: ”Schampo”, cost: 20, category: ”Hair” },
… andra produkter
]

Hämta en enskild produkt
Request
GET http://localhost:8080/products/hs46jfh8012nhdh

Response
200 OK
{ product_id: hs46jfh8012nhdh, name: ”Schampo”, cost: 20, category: ”Hair” }

Uppdatera en produkt
Request
PUT http://localhost:8080/products/hs46jfh8012nhdh
{ cost: 23, category: ”Body” }

Response
200 OK

Ta bort en product
Request
DELETE http://localhost:8080/products/hs46jfh8012nhdh

Response
200 OK

Försök att hämta obefintlig produkt
Request
GET http://localhost:8080/products/a29zjxher812c0

Response
404 Not Found

Försök att skapa en produkt med fel indata
Request
POST http://localhost:8080/products
{ name: ”Schampo”, cost: -1, place: ”Hair" }

Response
400 Bad Request

Avgränsningar

Ni ska använda MySQL som databas och SpringBoot för att implementera API:et.

Inlämning

Inlämning av projektfiler sker via Ping Pong senast **11/4** kl **23:59**. Gör en zip för att ladda upp projektet. Ni kan också ge en länk till er GitHub om ni har valt att ha ert projekt där.

Bedömning och återkoppling

Redovisning av uppgiften görs med en kort demonstration under 5-10 min **15/4** under eftermiddagen då vi vanligtvis har övning.

Betygskriterierna för Godkänt är

Godkänt

- Studenten har uppvisat kunskap om att använda grundläggande SpringBoot komponenter för att strukturera en REST-applikation (Controllers, Services, Repositories och Filters).
- Studenten har uppvisat kunskap om användning av JPA (ORM) för databas hantering.
- Studenten har uppvisat kunskap om användning av Dependency Injection i Spring.
- Studenten har uppvisat kunskap om skiktad arkitektur i en webbapplikation och DTOs.
- Studenten har uppvisat kunskap om användande av Exceptions i Spring för att returnera olika HTTP felkoder.

Återkoppling sker inom en vecka efter inlämning.