Turistguide

Del 1 (CRUD)

1. Introduktion

I denne opgave skal I lave API endpoints i en Spring Boot applikation, der kan anvendes til at oprette, søge og ændre information om turistattraktioner.

I første om gang skal CRUD operationerne være uden brugergrænseflade, dvs. uden en tilknyttet html side. Hver CRUD operation skal i stedet returnere et **`ResponseEntity`**.

I skal som det sidste i opgaven arbejde med HTML og CSS.

2. Læringsmål

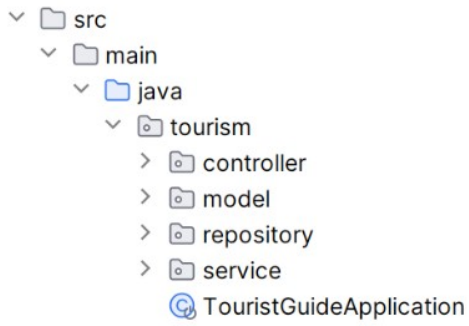
- ✓ At forstå HTTP request og response
- ✓ At forstå opbygningen af en URL
- ✓ At udarbejde endpoints i Spring Boot applikation
- ✓ At teste endpoints vha. HTTP Client værktøj i IntelliJ
- ✓ At udarbejde en HTML side, der styles med CSS

3. Spring Boot projekt

Opret vha. Spring Initializr et Spring Boot projekt, der hedder: **`TouristGuideAPI`**.

- ✓ Inkludér Spring Web
- ✓ Husk at vælge Maven

4. Packages i projektet



5. Klasser i projektet

a. Opret klassen **`TouristAttraction`** i model *package*.
Klassen skal (i første omgang) indeholde to `String` attributter **`name`** og **`description`**.

b. Opret klassen **`TouristRepository`** i repository *package* med annoteringen `@Repository`.

Tilføj en `ArrayList` til opbevaring af data (om ikke så længe skal I arbejde med en rigtig database). Opret et par `TouristAttraction` objekter, som tilføjes til denne `ArrayList`.

Klassen skal desuden indeholde CRUD metoder, der arbejder på ovenstående `ArrayList`. Vent evt. med den endelige metodesignatur for CRUD metoderne til I har set beskrivelsen af Controller klassens endpoints nedenfor.

c. Opret klassen **`TouristService`** i service *package* med annoteringen `@Service`.
Klassen skal indeholde CRUD metoder svarende til `TouristRepository` klassen og delegere kald til relevante metoder i denne.

d. Opret en **`TouristController`** klasse i controller *package* med annoteringen `@Controller`, samt **`@RequestMapping("attractions").`**

Lav nedenstående CRUD endpoints med funktionalitet, der alle returnerer et ResponseEntity.

6. Test af CRUD endpoints

Afprøv alle endpoints med HTTP Client i IntelliJ.

Afprøv ligeledes endpoints i browseren. Hvis du bruger Chrome kan du med fordel installere extension "JSON formatter", der gør, at JSON objekter vises pænere.

Endpoints skal kunne tilgås på følgende måde:

http://localhost:8080/attractions
- viser oplysninger om alle turistattraktioner

http://localhost:8080/attractions/tivoli
- viser oplysninger om attraktionen 'tivoli'

Oplysninger fra et Java objekt vises i JSON format, dvs. omgivet af { } og med attributværdier adskilt med komma. F.eks.:

```
{
   "name": "Tivoli",
   "description": "Forlystelsespark midt i København centrum"
}
```

7. Velkomstside (HTML og CSS)

Opret en velkomstside `index.html`, der vises med flg. endpoint: http://localhost:8080[1]

Teksten på velkomstsiden skal som minimum:

- Beskrive turistguiden - hvad kan den bruges til. Dækker den f.eks. en enkelt by, hele Danmark, eller er det en global bucket liste? Er den rettet imod en bestemt målgruppe (f.eks. singler eller naturelskere) eller andre relevante informationer.
- Vise en liste af udvalgte turistattraktioner (f.eks. via links til eksterne sites)
- Have links til de 2 GET endpoints i applikationen, dvs. links som kan
  - Vise alle de hardkodede turistattraktioner fra repository klassen
  - vise en af turistattraktionerne
- En footer med kontaktinfo

Sidens layout skal formateres med CSS for at se indbydende ud.

[1] Spring Boot finder selv automatisk index.html i resources/static mappen.

Krav til HTML

Brug som minimum disse tags[2]:

- head
- title
- nav
- footer
- img[3]
- ul or ol

[2] I finder en HTML Tag oversigt her https://www.w3schools.com/tags/default.asp
[3] Images placeres i projektet under resources/static/images.