REST. Styring af Router samt Status Koder i din REST API

Tidligere opgaver

- Design REST API
- Simpel Rest Service
- Rest med Cors
- Test Først af ManagerKlasse (TDD)

Opgave 1: Styring af Router (URI) i din REST API

Opgave 1.1: Generel URI til din REST API

Du skal i denne opgave undersøge den generelle URI (route) til din XXXXs**Controller** (fra simple Rest Service).

Når du har åbnet din XXXXsController, så vil der før klasse være en annotation
'`[Route("api/[controller]")]`' – Forklar hvad den gør?
Og hvorfor din URI ser ud som den gør (se bl.a. i hjælpesiden/swagger når du kører programmet)?

Hvordan kan du ændre din generelle URI/Route? Forklar de to måder.
Prøv de to måder.

Opgave 1.2: Opdel metode og URI(Route)

Du skal nu splitte metode fra URI (route) i de eksisterende 5 Services (funktioner) i din REST API.

Fx ser find element ud fra et id således ud ' `[HttpGet("{id}")]`'

Du skal dele den op i to annotations én til metoden og en til URI'en (routen), hvorved den vil således ud:

```
[HttpGet]
[Route("{id}")]
```

Gør det samme for Put og Delete.

Prøv om din REST API stadig virker.

Opgave 1.3: Opret en ny metode/funktion til dit REST API

Du har hidtil lavet 5 services/funktioner til dit REST API.

Du skal nu tilføje en funktion/service til dit REST API.
Det kan for eksempel være at finde flere data med fælles træk (søg på navn, søg på hoteller i en by eller lign)

Du skal definere http-metode, samt URI for din nye Funktion/Service, samt implementere den i XXXXsControlleren samt i din bagved liggende XXXXManager.

Opgave 1.4: Prøv din REST-service med den nye service/funktion

Kør din REST-service og prøv de forskellige metoder igennem den brugergrænseflade (swagger), der popper op.

Opgave 2: Styring af Status Koder i din REST API

I http protokollen sendes der status koder med tilbage i http-response. Du skal i denne opgave sørge for at sende sigende status koder tilbage.

For at se status koder i REST API:
https://restfulapi.net/http-status-codes/

Eller se her for generelt status koder i http:
https://en.wikipedia.org/wiki/List_of_HTTP_status_codes

Opgave 2.1 Refakturer din XXXXManager

Du skal i din XXXXManager refakturer dine metoder så de kaster en fornuftig exception, hvis fx et objekt ikke kan oprettes eller et objekt med en given id ikke kan findes.

Opgave 2.2 Refakturer din XXXXsController

Du skal ændre dine returtyper i dine REST-API-metoder, så de alle returnere følgende retur type:
'IActionResult'

Så i stedet for

```
public IEnumerable<Bil> Get()
```

Så

```
public IActionResult Get()
```

Dette betyder at du i controller-metoden skal ændre din retur værdi

I ovenstående eksempel:

return Ok(mgr.Get())

Hvis Listen er tom kunne man også sende

return NoContent();

PS! Hvis en status kode ikke direkte understøttes kan du altid benytte:
return StatusCode(HttpStatusCode.<anycode>);

Opgave 2.3 Prøv dit REST API

Kør din REST service.
Læg mærke til hvad der sker i 'hjælpeside' (swagger)

Opgave 2.4 Indsæt dokumentation af status koder

Du skal inden hver controller-metode angive i annotations hvilke status-koder denne metode vil sende (M.Hammel ville sige hvilke status koder slut brugeren ønsker).

Dette gøres ved fx:

[ProducesResponseType(StatusCodes.Status200OK)]
[ProducesResponseType(StatusCodes.Status404NotFound)]

Opgave 2.5 Prøv dit REST API - igen

Kør din REST service.
Læg mærke til hvad der sker i 'hjælpeside' (swagger)

Nogen forskel fra før (opgave 2.3)?

Hvis du ønsker en mere Tutorial tilgang se denne opgave der har en Item-model-klasse