REST. En Repetition

Du skal her endnu en gang lave en model klasse og et repository samt en rest service
Du burde ikke være mere end 2-2½ time om opgaven.

Øvelse til at lave en rest service til en dyrehandel

Opgave 1: Library

Opret et class library fx AnimalLib.

Opgave 1.1: Model Klasse

Du skal lave en klasse Animal med properties

- Id - et unikt heltal
- Category – en string med kategorien fx fugl, fisk, krybdyr, kat, hund …
  constraints skal være mindst 2 tegn langt
- Age – et heltal i år
- Price – en double
  constraints skal være over 10 og mindre end 10000

Lav klassen med properties (husk betingelser), konstruktører og en ToString

Opgave 1.2: Lav en unittest af din model klasse

Du skal lave en dækkende test af to properties (Category og Price).
Vis med code coverage at du har testet de to properties.

Opgave 1.3: Repository klasse

Du skal lave en klasse AnimalRepository der indeholder en liste med Animal, hvor der fra starten skal være tre dyr som mock-data. Dit Animal repository skal have følgende metoder

- **List<Animal> GetAll()** – returnere en liste med alle animals
- **Animal GetById(int id)** – kaster en exception hvis id ikke findes eller returnerer den det fundne objekt.
- **List<Animal> GetByCategory(string category)** – Returnere en liste med fundne objekter – kan være tom.
- **Animal Add(Animal animal)** – tildeler et unikt id og tilføjer objektet til listen, Returnere animal med nye id.
- **Animal Delete(int Id)** – Finder objekt med id'et og fjerner det fra listen, returnere det objekt der fjernes eller kaster en exception hvis Id'et ikke findes.
- **Animal Update(int id, Animal animal)** – Finder objektet med id'et og opdaterer det, returnere det objekt der er opdateret eller kaster en exception hvis Id'et ikke findes.
- **List<Animal> SortByCategory()** – returnerer listen med animal sorteret efter category
- **List<Animal> SortByPrice()** – returnerer listen med animal sorteret efter price

Opgave 1.4: Lav en unittest af din AnimalRepoistory klasse

Du skal lave en dækkende test af tre metoder (GetById, GetByCategory og Add).
Vis med code coverage at du har testet de tre metoder

Opgave 1.5: Lav et interface af dit AnimalRepository

Refaktorer din AnimalRepository klasse så det benytter et interface.
Du kan 'extract' et interface i visual studio ret let – højre klik på klassen , benyt quick actions and refactorings -> vælg extract interface

Opgave 2: Rest Service (API)

Opret et 'ASP.NET core web API' projekt – uncheck https
(fjern wheatherForecast model klasse og controller)

Lav en reference til din dll-fil fra opgave 1.

Opgave 2.1: Configurer Program.cs

Du skal tilføje builder.Services.AddSingleton<IAnimalRepository>(new AnimalRepository)

Opgave 2.2: Rest Controller klasse

Du skal oprette en ny API-controller AnimalsController, som får et objekt af din AnimalRepository klasse injected i konstruktøren

Tilpas de fem 'default' medtoder i controller-templaten

Opgave 2.3: Benyt DTO objekter i Add(post) og Update(put)

For at selv kan styre status koder – opret et AnimalDTO med felter svarende til modelklassen uden constraints.
Benyt DTO klassen i ADD hhv. Update metoderne.
Inden du kalder repositoriet skal du konvertere AnimalDTO objektet til Animal objekt, og evt. fange fejl.

Opgave 2.4: Udvid med statuskoder

Du skal sikre at din rest service returner fornuftige statuskoder (husk annotaions til swagger)

Opgave 2.5: Udvid med sort

Du skal lave to nye metoder med nye 'router' (URIer) til at hente animal sorteret efter category hhv. price.

Opgave 2.6: Udvid med filtrering

Du skal lave en ny metode (med route) til at søge/filtere, som modtager et Filter objekt 'from query'.
I denne opgave har filter objektet kun én property – category.

Opgave 2.7: Understøt Cors

Du skal i program.cs konfigurere din REST service til at understøtte CORS
v.hj.a. Builder.AddCors og app.UseCors

Opgave 2.8: KØR og prøv din Rest service

Du skal sikre at

Opgave 2.9: Publicer din Rest Service i Azure

Du skal publicere din Rest Service i Azure – tjek at det virker

Opgave 3: Test din Rest Service (API)

Du skal fra Postman teste din REST aervice

Opgave 3.1: Lav en test-suite i postman (Integration Test)

Du skal lave og køre en dækkende test af to API-kald (GetById (GET) samt Add (POST)).

Opgave 3.2: Kontroller at din Rest er sat rigtigt op til Cors

Du skal lav to kald fra Postman til at teste at Cors er sat rigtigt op

- Simple Request
- preflight request