

REST. Benyt Database til en REST API

Tidligere opgaver

- [Design REST API](#)
- [Simpel Rest Service](#)
- [Rest med Cors](#)
- [Test Først af ManagerKlasse \(TDD\)](#)
- [RoutningURI-StatusKoder](#)
- [SøgningOgPaging](#)
- [Swagger/hjælpe-sider \(ekstra\)](#)
- [Test din REST API med Postman](#)

Opgave 1: En simple Car REST API

Du starter med at lave en REST API til biler (en slags billet-system)

Opgave 1.1: Lav en enkelt tabel.

Du skal i en database (lokalt eller i Azure) lave en tabel med følgende informationer:

Tabel navn: **Car**

Entity informationer (columns):

- Registrerings nummer (string/nvarchar)
- Længde (double/float)
- Anhænger (bool/bit)

Indsæt nogle værdier i din tabel.

Opgave 1.2: A Installer værktøjet: EF Core Power Tool

Hvis du ikke har installeret **EF Core Power Tool**, så gør følgende

Du skal downloade værktøjet '**EF Core Power Tool**' fra

<https://marketplace.visualstudio.com/items?itemName=ErikEJ.EFCorePowerTools>

Du skal installere vsix-filen. Du bliver muligvis nødt til at lukke for Visual Studio for at installere vsix-filen.

Opgave 1.3: Lav forbindelse til database

Du skal åbne 'EF Core power tool' dvs. højre klik på dit projekt -> EF Core Power Tools -> Reverse Engineer.

Nu skal du angive din database server (find den i din 'SQL server object explorer'), angive database, samt login (dvs. windows, eller sql hvis du benytter Azure).

Så er det bare at klikke OK, vælg dine tabeller.

Derefter skal du navngive din DbContext (her i eksemplet CarDbContext, *men find selv et navn*) og angiv hvor dine modelklasser og dbContext skal ligge – passende i mappen 'model' se nedenunder:

Generate EF Core Model in Project StudentAPI

Context name: CarDbContext

Namespace: StudentAPI

EntityTypes path (f.ex. Models) - optional: model

EntityTypes sub-namespace (overrides path) - optional:

DbContext path (f.ex. Data) - optional: model

DbContext sub-namespace (overrides path) - optional:

What to generate: EntityTypes & DbContext

Naming

- Pluralize or singularize generated object names (English)
- Use table and column names directly from the database
- Use DataAnnotation attributes to configure the model
- Customize code using templates: C# - Handlebars
- Include connection string in generated code
- Install the EF Core provider package in the project

Help ★ Rate Advanced... OK Cancel

Du skulle nu i model have to filer en CarDbContext samt en model klasse Car.

Tag og kig på dem, hvordan er der defineret?

Du får desuden en 'PowerToolReadMe'-fil følg de to anvisninger der.

Opgave 1.4: Lav en CarManager

Lav et interface + en manager til at støtte CRUD til din 'Car' – tabel

Det er stort set som i [Simple REST](#) opgaven, dog skal du **IKKE** benytte en statisk liste men benyt din CarDbContext.

Lav et objekt af CarDBContext fx CarDBContext db = new CarDBContext();

Fx i metoden GetAll

```
Return db.Cars
```

Eller Metoden Create(Car car)

```
db.Add(car);
```

```
db.SaveChanges(); // Husk dette – ellers sker der ikke noget i databasen !!!
```

Opgave 1.5: Lav en CarsController

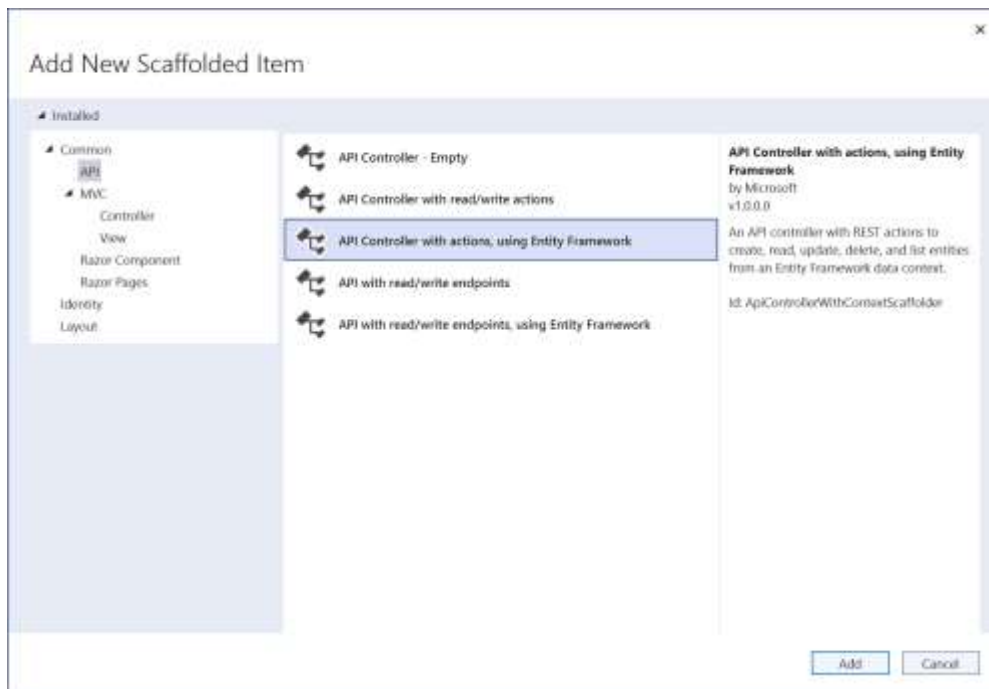
Lav en alm. CarsController (API) med retur status koder.

Opgave 1.6 Prøv din REST API

Kør sin applikation og se om opdateringer også gemmes i database tabellen.

Opgave ekstra A: Prøv den indbyggede controller med EF

I stedet for at lave trin 1.4 og 1.5, lav en controller med **'API Controller with actions using Entity Framework'**



Prøv din applikation – hvor dan virker den?

Opgave ekstra B: Benyt dine egne modelklasser og lav selv dit DbContext

Kig i denne opgave og lav part 3 + part 4 – du skal **IKKE benytte EF Core Power Tools**

Du kan evt lave det på din REST API du har arbejdet med hidtil.