

DS3103 Webutvikling interface, model, controller

Rolando Gonzalez, 2022

Innhold

- Hensikten med denne slideserien
- Opprettelsen av boilerplate
- Interface
- Model
- Controller
- Kjøre i nettleser

Hensikten med denne slideserien

- Hensikten med denne slideserien er å gi et eksempel på hvordan man kan jobbe med C#-klasser og objekter i et Web Api.
- Det simulerer samtidig hvordan man jobber med klasser og objekter når man får informasjon fra en database.
- Når man jobber med databaser (SQL-database eller Dokumentdatabase som MongoDB) henter man inn informasjon fra databasen inn i Web Apiet. Informasjonen i databasen blir gjort om til objekter basert på klasser man har i Models-mappen som «mapper» til informasjonen i databasen.

Opprettelsen av boilerplaten

- Finner egnet sted å opprette Web Api boilerplate
- Oppretter mappe
- Går inn i mappen
- Kjører kommandoen for å skape boilerplate: `dotnet new webapi`

```
Node.js command prompt
Your environment has been set up for using Node.js 16.13.1 (x64) and npm.

C:\Users\rogo001>cd desktop

C:\Users\rogo001\Desktop>mkdir ComposerApi

C:\Users\rogo001\Desktop>cd composerApi

C:\Users\rogo001\Desktop\ComposerApi>dotnet new webapi
The template "ASP.NET Core Web API" was created successfully.

Processing post-creation actions...
Running 'dotnet restore' on C:\Users\rogo001\Desktop\ComposerApi\ComposerA
  Determining projects to restore...
  Restored C:\Users\rogo001\Desktop\ComposerApi\ComposerApi.csproj (in 193
Restore succeeded.
```

Interface

- Et interface angir hvordan en klasse skal se ut, en form for regel som skal følges når utviklere lager klasse av noe: “An interface defines a contract. Any class or struct that implements that contract must provide an implementation of the members defined in the interface.” (Microsoft, 2022, <https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/interface>)

Icomposer.cs

- Setter namespace
- Deklarer public interface
- Definerer properties

```
C# IComposer.cs X C# Composer.cs C# ComposerController.cs
Interfaces > C# IComposer.cs > ...
1 namespace ComposerApi.Interfaces;
2
3     1 reference
4 public interface IComposer
5 {
6     5 references
7     | int Id {get; set;}
8     | 4 references
9     | string Name {get; set;}
10 }
```

Composer.cs

- Henter inn IComposer-tilgang
- Setter namespace
- Deklarerer klassen
- Setter properties som angitt i interfacet
- Setter default-verdi på Name i tilfelle det ikke settes

```
C# IComposer.cs  C# Composer.cs X  C# ComposerController.cs
Models > C# Composer.cs > ...
1  using ComposerApi.Interfaces;
2
3  namespace ComposerApi.Models;
4
5  9 references
   public class Composer : IComposer
6  {
7      5 references
       public int Id {get; set;}
8      4 references
       public string Name {get; set;} = "Not set"; // "Not set" er default-verdi
9  }
```

ComposerController.cs

- Henter inn Mvc-namespacet
- Henter inn tilgang til Model-klassene
- Definerer namespace
- Dekorert med ApiController og setter Route
- Arver fra ControllerBase
- Oppretter List med Composer-objekter
- Oppretter 2 endepunkter: 1 for å hente alle, 1 for å hente etter id
- ActionResult gjør at vi kan returnere statuskoder (200, 400 osv.)

```
C# IComposer.cs  C# Composer.cs  C# ComposerController.cs X
Controllers > C# ComposerController.cs > {} ComposerApi.Controllers > ComposerApi.Controllers.ComposerControll
1  using Microsoft.AspNetCore.Mvc;
2  using ComposerApi.Models;
3
4  namespace ComposerApi.Controllers;
5
6  [ApiController]
7  [Route("[controller]")]
0 references
8  public class ComposerController : ControllerBase
9  {
10     // Oppretter List<> som er en type array. Denne simulerer her en database.
11     private List<Composer> composers = new List<Composer>
12     {
13         new Composer { Id = 1000, Name = "Johan Sebastian Bach" },
14         new Composer { Id = 1001, Name = "Ludwig Van Beethoven" },
15         new Composer { Id = 1002, Name = "Edvard Grieg" },
16         new Composer { Id = 1003, Name = "Antonio Vivaldi" }
17     };
18
19     [HttpGet]
20     0 references
21     public List<Composer> Get()
22     {
23         return composers;
24     }
25
26     [HttpGet("{id}")]
27     0 references
28     public ActionResult<Composer> Get(int id)
29     {
30         Composer? chosenComposer = composers.Find( composer => composer.Id == id );
31
32         if( chosenComposer!= null )
33         {
34             return Ok(chosenComposer);
35         }
36         else
37         {
38             return NotFound();
39         }
40     }
}
```


Kjøre i nettleser

- Kjører løsningen: dotnet watch run
- Går inn i url og taster inn url for de 2 endepunktene:
 - localhost:xxxx/composer
 - localhost:xxxx/composer/yyyy

