

MVC Models

- Models
- Scaffolding
- Entity Framework
- Code First
- DbContext
- Scaffolding Template
- Data Context
- Udførelse af Scaffolding
- Database startværdier
- Seeding a Database
- ViewModel Redux
- Model binding

Models

- Models kan bruges til at sende data til Databasen
- Bruges til logik
- Beskrive hvordan dele af et view skal fungerer
- Beskrive hvad der skal være i et view
- Models er dem der bliver vist, gemt, skabt, redigeret og slettet

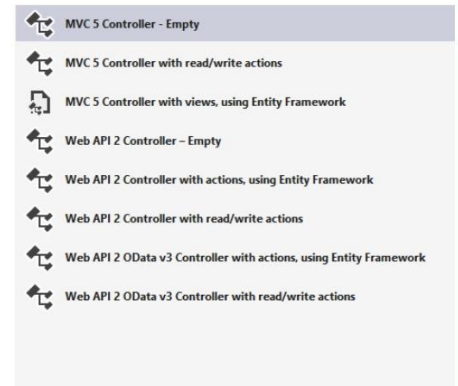
```
public class User
{
    public int ID { get; set; }
    public string Name { get; set; }
    public string Password { get; set; }
}
```

Scaffolding

- Boilerplate kode for at lave CROD - create, read, update og delete
- Den tager imod en model og laver en Controller og views
- Scaffolding hjælper med at lave det trivielle arbejde, efterfølgende skal man rette til i de nødvendige områder
- Der er mulighed for at udskifte det nuværende scaffolding system

Scaffolding

MVC 5 Controller - Empty	Laver en tom controller med en Index action
MVC 5 Controller with read/write Actions	Laver en controller med adgang til CRUD*. Koden i controller er ikke færdig og views er ikke lavet.
MVC 5 Controller with Views, Using Entity Framework	Denne template laver alle dele i en CRUD*. Controller, Views og koden der henter og sender til databasen
Web API 2	Arbejde med data over ajax. Der er flere templates til denne del. Du vil komme nærmere ind på det senere i forløbet.



*CRUD = Create, read, update and delete

Scaffolding og Entity Framework

- Object-relational mapping (ORM) framework
- EF kan arbejde med en relationsdatabase
- EF kan arbejde med database-first, model-first og code-first
- MVC scaffolders bruger code-first
 - Det betyder at man behøves kun at lave C# Klasser, og EF finder ud af resten
- Virtual properties hjælper EF med at holde øje med en klasse og kan så se om databasen skal opdateres

EF Code First konventioner

- Objektets navne vil blive til tabel navne hvis ikke andet er angivet
- Hvis en property har et navn ID vil EF se det som en primary key (identity)
- EF vil også så se på foreign key relationships, database names og meget andet
- Code First virker fantastisk godt når man starter et nyt projekt

DbContext Class

EF's code-first approach bruger DbContext til at få adgang til C# Klasser ved at bruge DbSet<T>.

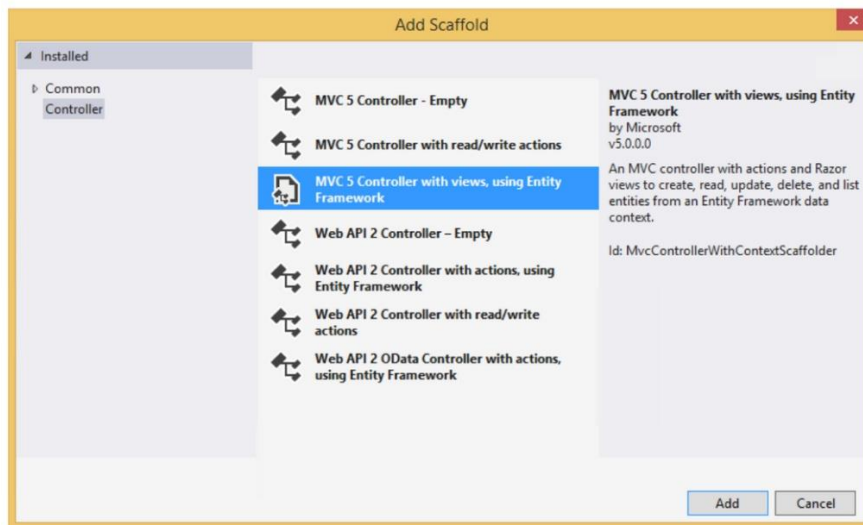
Dette vil åbne op for at kunne hente og gemme data ved brug af klassen

```
public class MusicStoreDB : DbContext
{
    public DbSet<Album> Albums { get; set; }
    public DbSet<Artist> Artists { get; set; }
    public DbSet<Genre> Genres { get; set; }
}

var db = new MusicStoreDB();
var allAlbums = from album in db.Albums
                orderby album.Title ascending
                select album;
```

Udførelse af Scaffolding Template

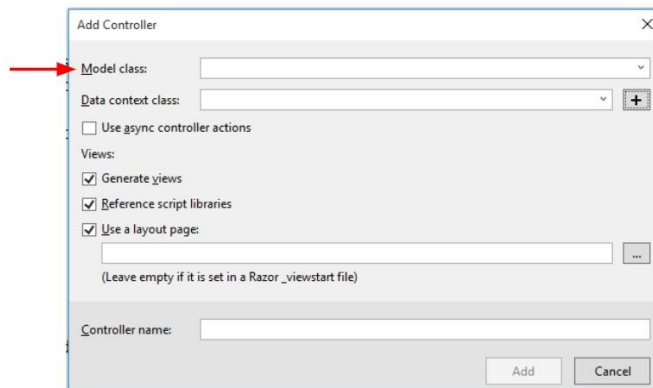
Højreklik på mappen
Controllers og vælg add>
Controller...



Udførelse af Scaffolding Template


Vælg den Model klasse
som scaffolding skal
arbejde med

En god ide er også at
vælge Reference script
libraries



Udførelse af Scaffolding Template

Du kan enten vælge en eksisterende Data context class eller få lavet en ny

Data context class: 

New Data Context ×

New data_context type:

Udførelse af Scaffolding Template

Udfyld Controller name



Add Controller

Model class:

Data context class: +

☐ Use async controller actions

Views:

☒ Generate views

☒ Reference script libraries

☒ Use a layout page:

...

(Leave empty if it is set in a Razor _viewstart file)

Controller name:

Add

Cancel

Data Context

Her er den genereret
DbContext.

Det er også muligt at skrive
dette selv.

Læg mærke til
:Base("name=MusicStoreDB")
Dette vil blive database navnet

```
public class MusicStoreDB : DbContext
{
    // You can add custom code to this file. Changes will not be overwritten
    //
    // If you want Entity Framework to drop and regenerate your database
    // automatically whenever you change your model schema,
    // please use data migrations.
    // For more information refer to the documentation:
    // http://msdn.microsoft.com/en-us/data/jj591621.aspx

    public MusicStoreDB() : base("name=MusicStoreDB")
    {
        public DbSet<MvcMusicStore.Models.Album> Albums { get; set; }

        public DbSet<MvcMusicStore.Models.Artist> Artists { get; set; }

        public DbSet<MvcMusicStore.Models.Genre> Genres { get; set; }
    }
}
```

Scaffolding

Genereret...

- StoreManagerController
- Views
- Data Context

Udførelse af Scaffolding Code

Ved at besøge siden /StoreManager/ , vil EF skabe en database.

Du kan se database strukturen ved at åbne Server Explorer i VS2015

EF vil også lave en tabel med migration historie (versions)

Database startværdier

Når der bliver kørt Code first, kan man sætte EF til at enten
DropCreateDatabaseAlways
el.
DropCreateDatabaseIfModelChanges

```
protected void Application_Start()  
{  
    Database.SetInitializer(  
        new DropCreateDatabaseAlways<MusicStoreDB>());  
}
```

Seeding a Database

Man kan kalde en Seed for at give databasen nye data.

Lav den nye klasse i models mappen

Fortæl så SetInitializer at den skal bruge MusicStoreDbInitializer

Global.asax.cs

```
public class MusicStoreDbInitializer
    : System.Data.Entity.DropCreateDatabaseAlways<MusicStoreDB>
{
    protected override void Seed(MusicStoreDB context)
    {
        context.Artists.Add(new Artist {Name = "Al Di Meola"});
        context.Genres.Add(new Genre { Name = "Jazz" });
        context.Albums.Add(new Album
            {
                Artist = new Artist { Name="Rush" },
                Genre = new Genre { Name="Rock" },
                Price = 9.99m,
                Title = "Caravan"
            });
        base.Seed(context);
    }
}
```

Global.asax.cs

```
protected void Application_Start() {
    Database.SetInitializer(new MusicStoreDbInitializer());
}
```


MusicStoreDbInitializer

Genereret...

- En ny database
- Nye data

ViewModel Redux (returned)

“view specific model”

Hvis vi gerne vil bruge Strongly typed views, vil det være en fordel at bruge en view specific model.

ViewModel kan bruges som en model og kan så indeholde data fra et View

ViewModel behøves ikke at være en fordel frem for en ViewBag

```
ViewBag.ArtistId =  
    new SelectList(db.Artists, "ArtistId", "Na  
ViewBag.GenreId =  
    new SelectList(db.Genres, "GenreId", "Name
```

```
public class AlbumEditViewModel  
{  
    public Album AlbumToEdit { get; set; }  
    public SelectList Genres { get; set; }  
    public SelectList Artists { get; set; }  
}
```

Model binding

Når der er et post til edit, vil systemet prøve at matche form-post-data med en model hvis ikke andet er angivet.

For at sikre at der ikke sker en Overposting, som kan blive en svaghed i systemet. Kan man angive hvilken værdier vi tillader at modtage.

```
[HttpPost]
public ActionResult Edit(Album album)
{
    // ...
}
```

```
public ActionResult Edit
([Bind(
    Include="AlbumId,GenreId,ArtistId,Title,Price,AlbumArtUrl")]
Album album)
```