

## UserStories Part 12

### (EntityFramework i console applikation)

For at isolere EntityFramework lave du et isoleret eksperiment, hvor du afprøver EntityFramework i en ConsoleApplikation. Du kan så senere flytte det til en Razor Applikation.

#### Del 1: Lav Console Applikation

Lav en alm. console applikation, gerne med en worker klasse så du ikke kalder direkte fra den statiske main metode.

#### Del 2: Forbered EntityFramework

Trin1: Installer NuGet pakker

- Microsoft.EntityFrameWorkCore.Tools
- Microsoft.EntityFrameWorkCore.SqlServer

Trin2: Download og installer 3 parts hjælpe tool

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

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

Du bliver vist nok nødt til at lukke for Visual Studio for at installere vsix-filen.

Trin3: Genere modelklasser og adgang til Database

I dit projekt lav to foldere (mapper) 'model' og 'services'.

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 (lokal eller i azure er ligegyldigt), samt angive database, samt login (hvis lokalt windows ellers sql username+password)

Så er det bare at klikke OK, vælg dine tabeller og angiv dine modelklasser og dbContext skal ligge i model

Husk at tjekke

Generate EF Core Model in Project RazorPagesEventManager\_Chapte...

Context name: EventMakerDbContext

Namespace: RazorPagesEventManager\_Chapter13

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

EntityTypes sub-namespace (overrides path) - optional:

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

DbContext sub-namespace (overrides path) - optional: Models

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 Handlebars templates: C#
- Include connection string in generated code
- Install the EF Core provider package in the project

Advanced... OK Cancel

Du skulle nu i model have to filer en DbContext (hedder måske noget lidt andet) samt en model klasse UserStory

Trin4: Lav adgang til database gennem DbContext

I service lav et Interface IUserStoryPersistence

- GetAll
- GetOne(Id)
- Create
- Delete
- Update

I service lave en klasse UserStoryPersistence der implementerer IUserStoryPersistence

Klassen skal have et instansfelt af DbContext klassen fx navn 'db'.

Eksempel på GetAll

```
return db.UserStories.ToList();
```

Eksempel på Create

```
db.UserStory.Add(student);  
db.SaveChanges(); -- HUSK
```

Trin 5: Prøv din service

I din worker klasse lav et objekt af UserStoryPersistence og prøv de fem forskellige metoder.

Se i databasen om der sker de rette opdateringer.

