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=ErikEJ.EFCorePowerTools

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

Trin3: Genere modelklasser og adgang til Database

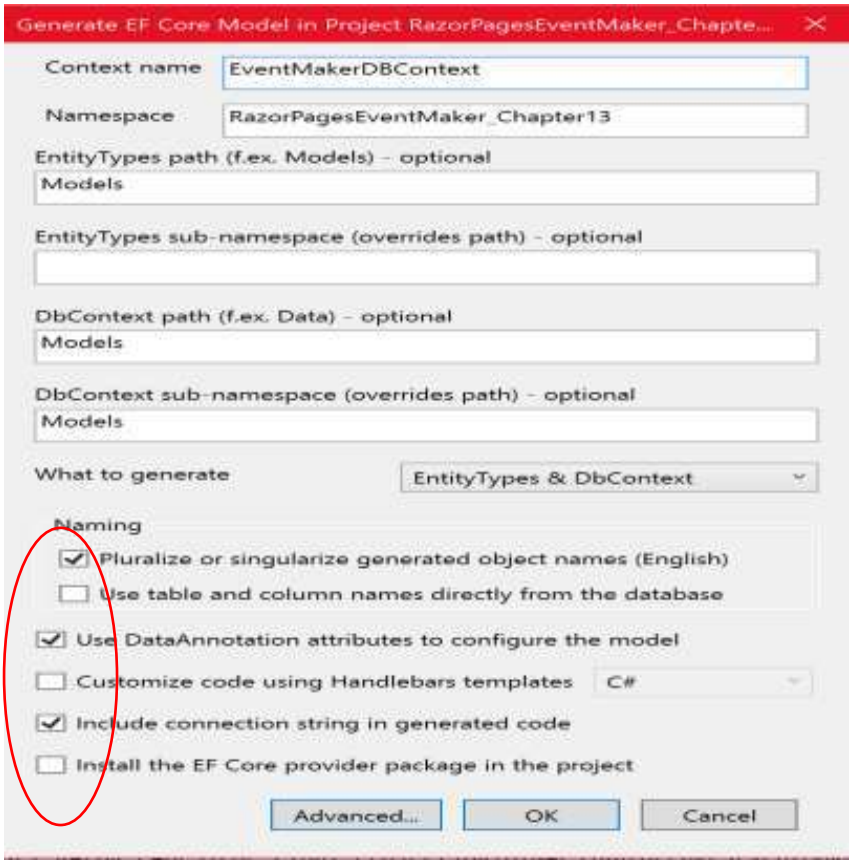
I dit projekt lav to foldere (mapper) 'model' og 'services'.

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



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 IUserStoryPersistsence

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

I service lave en klasse UserStoryPersistence der implementerer IUserStoryPersistsence

Klassen skal have et instansfelt af DBContext klassen fx navn ’db’.

Eksemple på GetAll

return db.UserStorys.ToList()

Eksemple 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.