Source/repos/.netcore/DBFirstApproachEFCore

DBFirst Approach

Creating a Model for an Existing Database in Entity Framework Core

* Creating entity & context classes for an existing database is called Database-First approach.(quite opposite to EF6)
* EF Core does not support visual designer for DB model
* So do reverse engineering using the Scaffold-DbContext
* This reverse engineering command creates entity and context classes (by deriving DbContext)

Let’s see how to create it:

Tools -> NuGet Package Manger -> Package Manger Console and run the following command:

PM> Install-Package Microsoft.EntityFrameworkCore.Tools

PM> Install-Package Microsoft.EntityFrameworkCore.SqlServer

PM>Scaffold-DbContext "Server=(localdb)\MSSQLLocalDB;Database=BookStore;Trusted\_Connection=True;" Microsoft.EntityFrameworkCore.SqlServer -OutputDir Model

* first parameter is a connection string
* second parameter is the provider name
* -OutputDir parameter specifies the directory where we want to generate all the classes which is the Models folder in this case.

**Note:** EF Core creates entity classes only for tables and not for StoredProcedures or Views.

**Note:** Once you have created the model, you must use the Migration commands whenever you change the model to keep the database up to date with the model.

That’s it. The above codings created Model class and context class inside Model folder.

Now add controller by right click controller and select add->New scaffold item->MVC with Views using EF

Now add connection string in appsettings.json

{

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft": "Warning",

"Microsoft.Hosting.Lifetime": "Information"

}

},

"AllowedHosts": "\*",

"ConnectionStrings": {

"BookStoreContext": "Server=(localdb)\\mssqllocaldb;Database=BookStore;Trusted\_Connection=True;MultipleActiveResultSets=true"

}

}

Then add it in startip.cs

public void ConfigureServices(IServiceCollection services)

{

services.AddControllersWithViews();

services.AddDbContext<BookStoreContext>(options =>

options.UseSqlServer(Configuration.GetConnectionString("BookStoreContext")));

}

Project is created

Code First Approach:

## Add a data model class

public class Movie

{

public int Id { get; set; }

public string Title { get; set; }

[DataType(DataType.Date)]

public DateTime ReleaseDate { get; set; }

public string Genre { get; set; }

public decimal Price { get; set; }

}

## Add NuGet packages

Install-Package Microsoft.EntityFrameworkCore.Design

This will install package in Dependencies folder

## Scaffold movie pages

* In **Solution Explorer**, right-click the Controllers folder **> Add > New Scaffolded Item**.
* In the **Add Scaffold** dialog, select **MVC Controller with views, using Entity Framework > Add**.
* In the **Model class** drop down, select **Movie (MvcMovie.Models)**.
* In the **Data context class** row, select the **+** (plus) sign.
  + In the **Add Data Context** dialog, the class name *MvcMovie.Data.MvcMovieContext* is generated.
  + Select **Add**.
* **Views** and **Controller name** : Keep the default.
* Select **Add**.

Scaffolding updates the following:

* Inserts required package references in the *MvcMovie.csproj* project file.
* Registers the database context in Startup.ConfigureServices of the *Startup.cs* file.
* Adds a database connection string to the *appsettings.json* file.

Scaffolding creates the following:

* A movies controller: *Controllers/MoviesController.cs*
* Razor view files for Create, Delete, Details, Edit, and Index pages: Views/Movies/\*.cshtml
* A database context class: *Data/MvcMovieContext.cs*

The automatic creation of these files and file updates are known as *scaffolding*.

## Initial migration

Add-Migration InitialCreate

Update-Database

## Test the app

Run the app