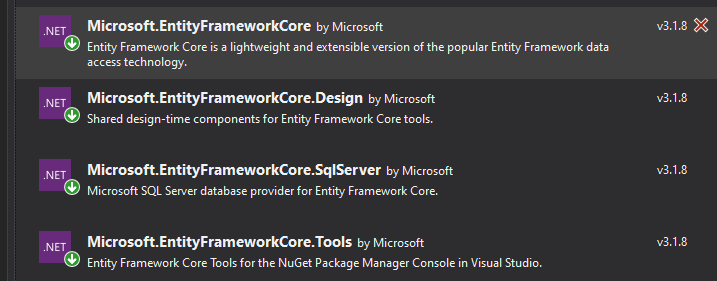


Installed packages.

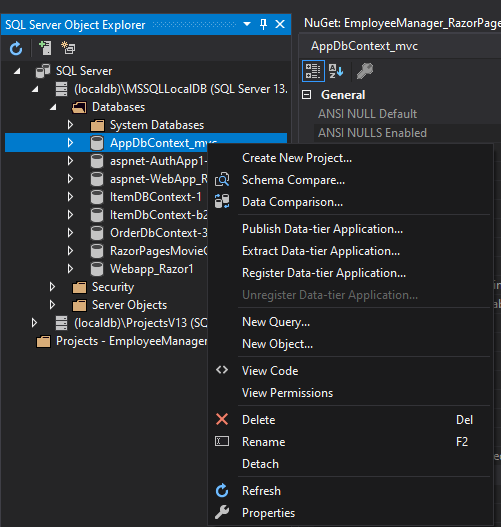


Ok so.. I had some issues getting this to work the way intended.

I previously used Scaffold-dbcontext (which I believe is part of the .tool) but at any rate is I guess different than dotnet ef dbcontext-scaffold. I was able to reverse engineer and scaffold but I wasn’t able to use the -c, and I think maybe the -t options that the book uses with the dotnet ef dbcontext-scaffold.

After some frustration I was able to follow the books method.

1. Start new project. ASP.NET Razor Page project with empty template.
2. Install nuget packages pictured above. (Microsoft.EntityFrameworkCore, Microsoft.EntityFrameworkCore.Design, Microsoft.EntityFrameworkCore**.**Tools, Microsoft.EntityFrameworkCore.SqlServer) I’m still kind of confused. I think you may not need the tools package. That may have come from Scaffold-DbContext command. Still confused on the differences. The book mentions you need the design package. I’ll have to verify this later, for now I will keep all 4 packages.
3. Now will use the “dotnet ef dbcontext scaffold” command It is in form “dotnet ef dbconext scaffold <connection> <provider> [options]
4. Pull up the SQL Server Object Explorer (view drop menu if not visible), find your database, right click and view properties.



1. Find the connection string in the properties window.



You may not be able to read it, but basically it lists the connection string as.. Data Source=(localdb)\MSSQLLocalDB;Initial Catalog=AppDbContext\_mvc;Integrated Security=True;Connect Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFailover=False

This connection string is different than what I used. I used a few different sources. Not sure exactly how I ended up using the connection string I used, but it worked. Using Source instead of Server and initial catalog instead of database. I’m not sure if it will matter but I used the bold command below, and it worked for me.

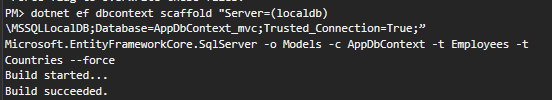
1. My provider was Microsfot.EntityFrameworkCore.SqlServer.
2. So finally with the packages installed, our dotnet ef dbcontext scaffold command read simply enter the command into the PMC.

**dotnet ef dbcontext scaffold "Server=(localdb)\MSSQLLocalDB;Database=AppDbContext\_mvc;Trusted\_Connection=True;” Microsoft.EntityFrameworkCore.SqlServer -o Models -c AppDbContext -t Employees -t Countries --data-annotations**

use the above dotnet ef dbcontext scaffold command. It has the data-annotations, which is what I have been using so that what is good for me at this time.

The below uses Fluent API. Not exactly sure the exact differences.

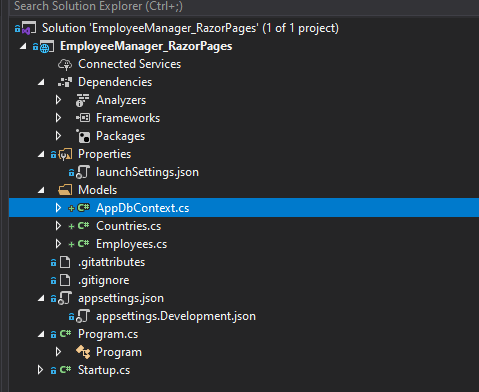
IF YOU DO USE FLUENT API. You must find another way to validate at the UI-level (which data annotations does do). Use Fluent Validation or some other third part validation library.



**dotnet ef dbcontext scaffold "Server=(localdb)\MSSQLLocalDB;Database=AppDbContext\_mvc;Trusted\_Connection=True;” Microsoft.EntityFrameworkCore.SqlServer -o Models -c AppDbContext -t Employees -t Countries**

The bold test above is the command I used. The picture shows the actual run on the command.

And it generated the following..



AFTER YOUR CLASSES ARE GENERATED YOU WILL PROBABLY NEED TO MAKE CHANGES TO THE GENERATED CLASS (I DID FOR THIS PROJECT).

The changes I made were changes to the Countries and Employees classes generated. Change to the singular ie Country, and Employee. Also added all the data annotations from EmployeeManager\_MVC. Made changes in DBContext class. Removed OnConfiguring(), and OnModelCreating().