# Creating a .NET Core MVC Project with Database First Entity Framework

## Create a .NET Core MVC Project

* File, New Project, Web, ASP.NET Core Web Application (or search in VS2019)
* Select MVC

## Get Your Models from the Database

You can use the Package Manger Console and a PowerShell cmdlet to scan your database and create the classes needed for the Entity Framework. (see https://docs.microsoft.com/en-us/ef/core/miscellaneous/cli/powershell#scaffold-dbcontext)

In the following example, update the connection string to point to your SQL Server and your database (Initial Catalog =). (The following is all one line!)

**Scaffold-DbContext** "*Data Source=****(localdb)\MSSQLLocalDB****;  
 Initial Catalog=****PRS****;  
 Integrated Security=True;  
 Connect Timeout=30;  
 Encrypt=False;  
 TrustServerCertificate=False;  
 ApplicationIntent=ReadWrite;  
 MultiSubnetFailover=False*"   
-Provider "Microsoft.EntityFrameworkCore.SqlServer"   
-OutputDir Models   
-DataAnnotations

**Notes:**

* **Scaffold-DbContext** – The PowerShell cmdlet
* **"…"** – your connection string
* **-Provider** – the provider for the data source. Usually the one listed.
* **-OutputDir** – The folder to write the files to. Typically “Models”.
* **-DataAnnotations** – You have two options:
  + Leave off -DataAnnotations and your context class will be provided with “Fluent API” statements to define details about the table, including defaults and maximum lengths.
  + Add -DataAnnotations and your model classes will be update with annotations such as “[StringLength(30)]”. SQL features that cannot be described using annotations will be added to the context class as “Fluent API” code.
  + Using DataAnnotations will help when building Razer Views!

## Review and update your EF models and DbContext.

* Review your entity model classes and the attributes that have been added to the properties.
* Add any desired attributes for Razer Views. Example: [Display(Name = "First Name")]
* Review the DbContext class.
* Comment out, or delete, the “OnConfiguring” method. (This is BAD place to store connection strings.) You may want to copy this connection string while you are here.
* Note the OnModelCreating method and the “Fluent API” code used to define the SQL Server constraints and column defaults. There is one section (modelBuilder.Entity) for each table.
* If you will not be using “UPDATE-DATABASE” to create a new database (maybe in Azure), you can delete the entire OnModelCreating method. You may want to just comment it out and keep it for documentation.
* Note that you will generally not be using Add-Migration or Update-Database when you are using “database first” patterns.

## Edit Startup.cs:

* Add a using statement: using Microsoft.EntityFrameworkCore;
* In the ConfigureServices method add your connection string. This is the same connection string used with Scaffold-DbContext.

services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version\_2\_2);

**var connection = @"Data Source=(localdb)\MSSQLLocalDB;Initial Catalog=DotNetCoreMVC.Models.MyDB;Integrated Security=True;Connect Timeout=30;Encrypt=False;TrustServerCertificate=True;ApplicationIntent=ReadWrite;MultiSubnetFailover=False";**

**services.AddDbContext<Models.MyDB>**

**(options => options.UseSqlServer(connection));**

* Replace “MyDB” with the name of the wizard generated context.

Note: Storing the connection string in the code is BAD! Store it in appsettings.json and retrieve with:  
 var connetion = Configuration["yourConnectionStringName"];

Build the project.

## Add your Controllers and Views

All done!