**SETUP DEVELOPMENT ENVIRONMENT**

VSCODE <https://code.visualstudio.com/>

.Net Core SDK

Download the latest Node.js software: <https://nodejs.org/en/download/>

Type “npm” and enter command should execute successfully.

Install dotNetCore from this page:

<https://dotnet.microsoft.com/download/dotnet-core/thank-you/sdk-3.1.201-windows-x64-installer>

In command prompt run following commads (it should display appropriate versions):

*dotnet --list-sdks*

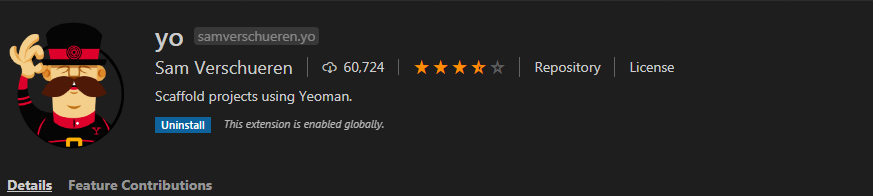
*dotnet --list-runtimes*

Install GIT from this link: <https://git-scm.com/download/win>

In VSCode press Shift+Ctrl+P and type: Install Extension and install following:

* Angular Snippets (Version 9)
* ASP.Net Helper
* C#
* C# Extension
* Rest Client
* SQL Server (MSSQL)
* Beautify

Install Yeoman to generate a template for a new project.



**npm install –g you**

**Create a new Project**

In the VSCode terminal type following:

**mkdir<*name\_of\_the\_project*>**

enter to this project (with command cd ‘project\_name’) and run the following command:

**dotnet new angular**(old command is : yoaspnetcore-spa)

Install dot net watch: <https://github.com/dotnet/AspNetCore.Docs/blob/master/aspnetcore/tutorials/dotnet-watch.md>

Add item group into your .ccproj file.

Running the command: **dotnet watch run** you will see the changes in cs files without refreshing (re-running) the application.

GIT

Commit your local changes to master. Then enter the command:

git remote add *origin* [*https://github.com/Nedzad83/VehicleFactory.git*](https://github.com/Nedzad83/VehicleFactory.git)

**Adding EF Core to our project**

Use the following .NET Core CLI command from the operating system's command line to install or update the EF Core SQL Server provider:

Dotnet add package Microsoft.EntityFrameworkCore.SqlServer

dotnet add package Microsoft.EntityFrameworkCore.SqlServer --version 3.1.3

Also, we need to add following:

dotnet add package Microsoft.EntityFrameworkCore.Tools.DotNet

**Creating a project**

First we need to create two additional models: *Make* and *Model*(add them into a new folder).

Next, we need to install EntityFramework package, add following code into .csproj file:

dotnet add package Microsoft.EntityFrameworkCore.Tools.DotNet --version 2.0.3

<PackageReference Include="Microsoft.EntityFrameworkCore.Tools.DotNet" Version="2.0.3" />

Then in the terminal window enter: *dotnet restore*. If you want to check if it is installed successfully, just enter following commands:

*dotnet-ef*

*dotnet-ef migrations –help*

**Create a DbContext**

We need to create a DBContext. In the new folder add a new class: *VehicleDbContext*, and inside of it define constructor:

        public VehicleDbContext(DbContextOptions options)

        : base(options)

        {

        }

Now we need to register the *VehicleDbContext* as a service for Dependency Injection, so later when we create the controller, we gonna pass the DbContext in the constructor of the controller. Dependency Injection framework inside ASP.Net Core will automatically create the instance of the *VehicleDbContext* and pass it to controller. So we are not going to explicitly create a new *VehicleDbContext* in a controller. Instead of that we gonna pass a dependency of a *VehicleDbContext* class in a controller (in a constructor) in order to achieve loose coupling. Also, later if we want to unit-test them, we can replase those dependencies with fake objects.

So, in the Startup class we will register our dependency:

services.AddDbContext<VehicleDbContext>(options => options.UseSqlServer(Configuration.GetConnectionString("Default")));

In the appsettings.json file add:

  "ConnectionStrings": {

    "Default": "Data Source=…\\SQLEXPRESS;Initial Catalog=VehicleFactory;Integrated Security=True;"

  },

In the Terminal console run this command:

dotnet add package Microsoft.EntityFrameworkCore.Design

dotnet ef migrations add InitialModel

If you want to remove previous migration you should not remove it manually, but you can use following commande instead: dotnet ef migrations remove

After successfully run the migrations, new ‘Migrations’ folder should be added in your project structure. So finally, we need to apply our migration to a database. We can do this with following command:

dotnet ef database update

**Populate Database**

We will create empty migration with this command:

dotnet ef migrations add SeedDatabase

Populate UP and DOWN function in SeedDatabase file with appropriate queries and then update it to database:

dotnet ef database update

**BUILD API**

In the Controller folder, add a new controller and name it: MakeController. Inside of it add a new method like this:

        [HttpGet("/api/makes")]

        public async Task<IEnumerable<Make>> GetMakes()

        {

            return await context.Makes.Include(x => x.Models).ToListAsync();

        }

dotnet add package AutoMapper

dotnet add package AutoMapper.Extensions.Microsoft.DependencyInjection --version 7.0.0

dotnet restore

In startup file register AutoMapper:

services.AddAutoMapper(typeof(Startup));

In MakesController:

 return mapper.Map<List<Make>, List<MakeResource>>(makes);

Also we need to add MapperProfile so mapper know how to cast objects.

git config --global user.email "email@example.com"