**CMPG 323**

**Project 2 – ASP.NET Core**

**guidance documentation**

**18 August 2022**

1. **Prepare your database and connection string (local DB)**

# You need to install the OLE DB Driver for SQL Server, SQL Server 2019 and SQL Server Management Studio if you have not already done so.

# <https://www.tutorialsteacher.com/sqlserver/install-sql-server>

# <https://www.microsoft.com/en-us/download/details.aspx?id=56730>

# You will create the database from a script at found at <https://www.dropbox.com/s/umo29a0vavaicfg/Database.sql?dl=0>. We assume that you know and understand how to create and work with a database, therefore this is not a focus of the project.

# Run the script to create the database in SSMS or paste it into new query and run it.

# Get the database connection string to be used in the project as follows:

# We are working with a local database for now, the final project must be hosted on the cloud, meaning that security will be required (username and password).

# Right click in a folder on your computer and create a new text file (.txt)

# Ensure that - Show file extensions and change the file to a .udl file

# Double click on the file to see the following pop up

# 

# Enter the server’s name (according to your computer account name)

# For the local DB there is no security set up, choose Windows NT integrated security. When deploying the DB to Azure, you will need to select the username/password option and allow password saving.

# Select the database you created and OK

# Rename the .udl file back to .txt

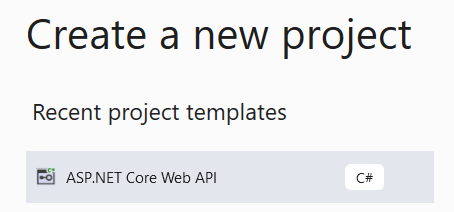
# Open the file to view the connection string

# Note – for project 2, you will have to use a database on Azure. A different script will be provided in the class of 25 August, when Azure is introduced.



1. **Create a new project – ASP.NET Core Web API**

Choose the template as shown below

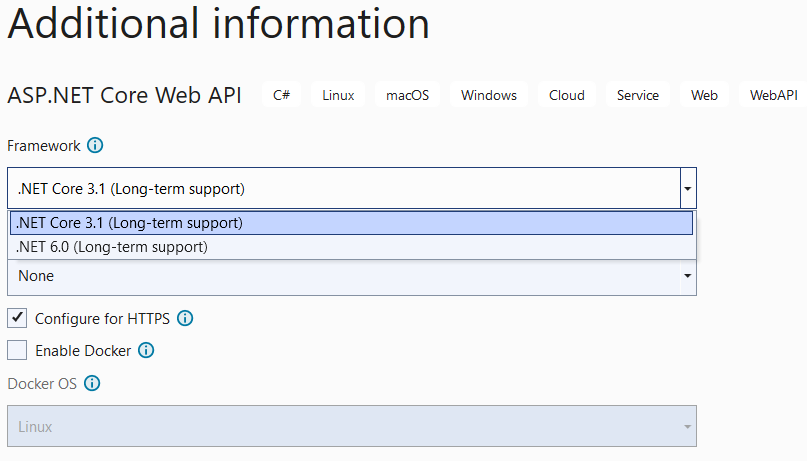


**Install ASP.NET Core 3.1.28** if it is not available. In this case, it can be selected as it has been installed. This is a very stable version of the software and is used in industry for that reason.

If you decide to work on any later version, we will not be able to assist you with questions.

<https://dotnet.microsoft.com/en-us/download/dotnet/3.1>

* SDK 3.1.422
* ASP.NET Core Runtime 3.1.28



**Select the Framework - .NET Core 3.1**

1. **Apply the Database First Approach**

This is an alternative to the Code First and Model First approaches to the Entity Data Model. It creates model codes (classes, properties, DbContext etc.) from the database in the project and those classes become the link between the database and controller. It creates the entity framework from an existing database.

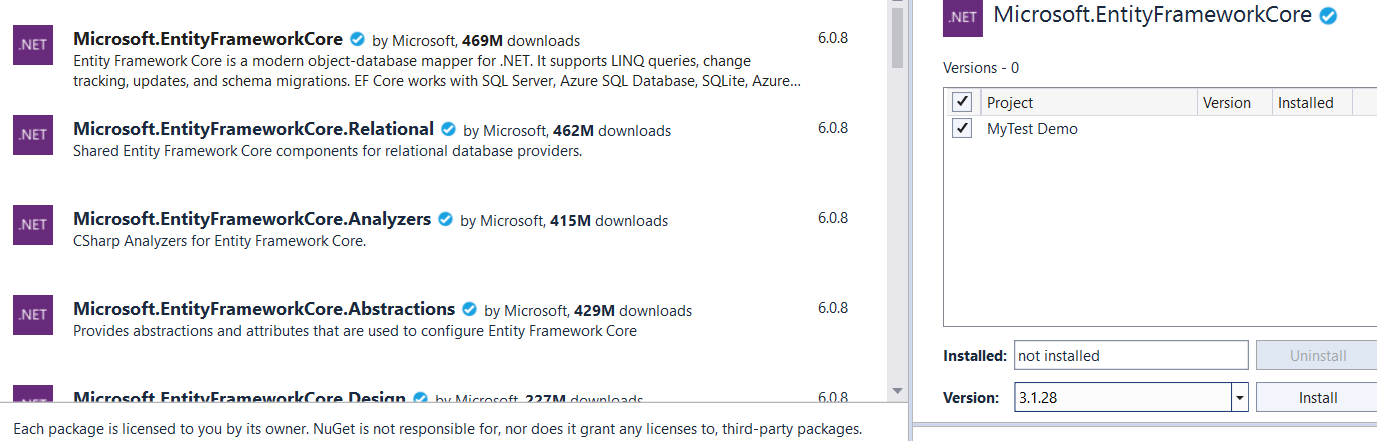
**Install the Entity Framework**

Tools / NuGet Package Manager / Manage NuGet Packages for Solution

Select browse and install the following packages:

* Microsoft.EntityFrameworkCore
* Microsoft.EntityFrameworkCore.SqlServer
* Microsoft.EntityFrameworkCore.Design
* Microsoft.EntityFrameworkCore.Tools

Note that for each install **the correct version** **(3.1.28)** must be selected.



Select the **Package Manager Console** at the bottom right.

Read <https://docs.microsoft.com/en-us/ef/core/cli/dotnet>

Type: dotnet ef to verify that EF Core CLI tools are correctly installed.

**How to scaffold the database**

Scaffolding a database produces an Entity Framework model from an existing database. The resulting entities are created and mapped to the tables in the specified database.

The commanddbcontext scaffold - Generates code for a DbContext and entity types for a database. In order for Scaffold-DbContext to generate an entity type, the database table must have a primary key.

Type in the the Package Manager Console:

dotnet ef dbcontext scaffold "<>" Microsoft.EntityFrameworkCore.SqlServer -o Models

And replace <> with your connection string.

For my example I use - Integrated Security=SSPI;Persist Security Info=False;Initial Catalog=**ProductsDB**;Data Source=**NWUUSER**

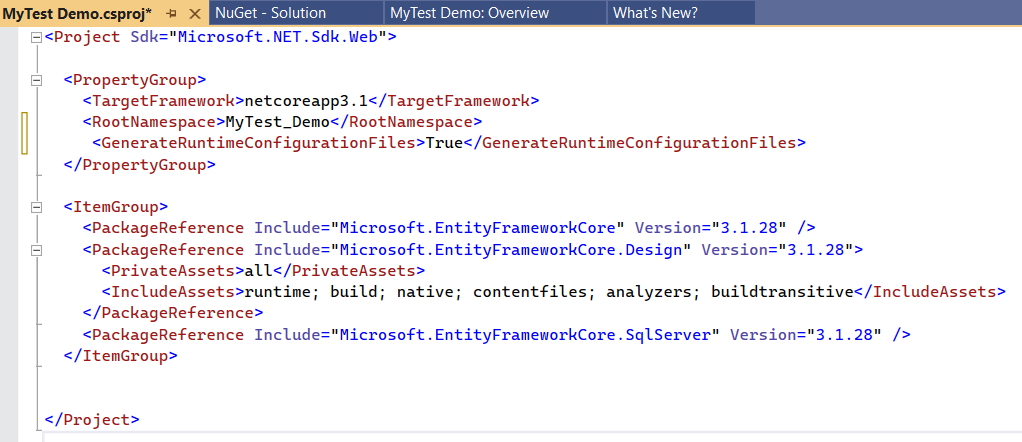
If the following error occurs do the following:

Your startup project 'MyTest Demo' doesn't reference Microsoft.EntityFrameworkCore.Design. This package is required for the Entity Framework Core Tools to work. Ensure your startup project is correct, install the package, and try again.

Click on the solution to see the .csproj file

Add the following line

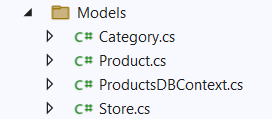
<GenerateRuntimeConfigurationFiles>True</GenerateRuntimeConfigurationFiles>



If it still happens, open your startup.cs and add the following if it not there

using Microsoft.EntityFrameworkCore;

If you open the Models on the solution, you should see the tables as well as the DBContext files. A A **DbContext** instance represents a session with the database and can be used to query and save instances of your entities. Open the files to view the content



# Update the appsettings.json file with the DB connectionstring to be used later

"ConnectionStrings": {

"DefaultConnection": "Integrated Security=SSPI;Persist Security Info=False;Initial Catalog=ConnectedOffice;Data Source=NWUUSER"

}

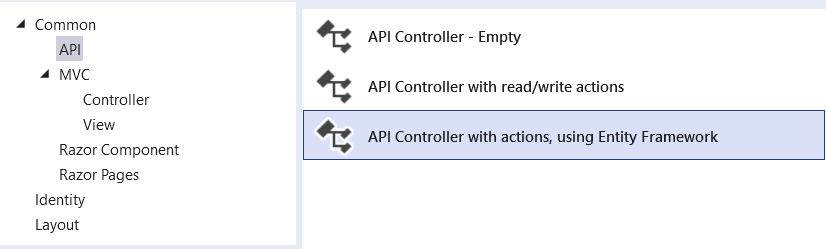
# 

# 

1. **Create controller to update database tables**

Delete the default WeatherForecastController.cs and WeatherForecast.cs

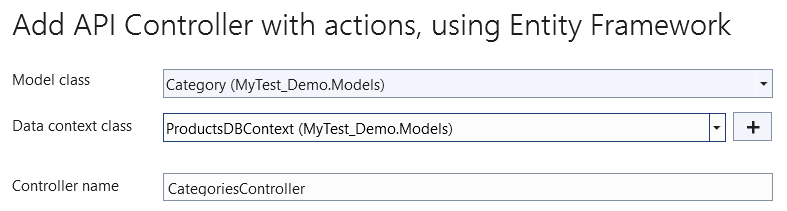
Right-click on the Controller folder and choose the following:



Make sure that the Models are referenced in the startup.cs

using MyTest\_Demo.Models; where MyTest\_Demo is the name of the solution

Choose a table to add and the DBContext (here the Category table is used) and the name of the controller. Default options are fine.



Your controller is now ready to use to update the table (resource) using HTTP commands



Execute the project – should get an error

1. **Install Swagger**

Swagger helps users build, document, test and consume RESTful web services

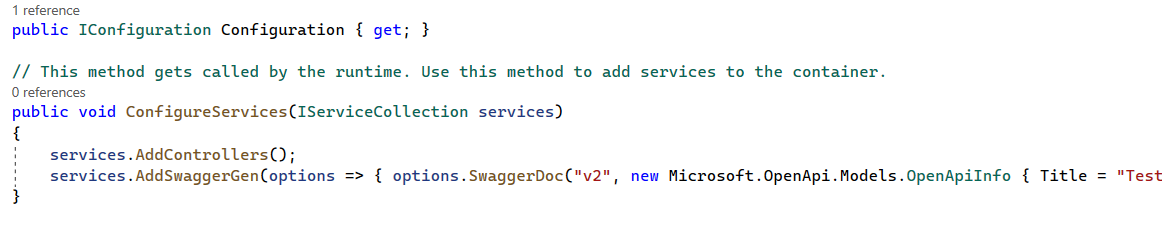
<https://www.c-sharpcorner.com/blogs/implement-swagger-in-asp-net-core-31-web-api>

Install the following packages

* Microsoft.OpenApi (latest version)
* Swashbuckle.AspNetCore (latest version)

Add the following code **to the beginning** of the startup.cs

services.AddSwaggerGen(options => {options.SwaggerDoc("v2", new Microsoft.OpenApi.Models.OpenApiInfo {Title = "MyTest Demo API", Version = "v2", Description = "Test demo for students",}); });



Add the following code **to the end** of the startup.cs

app.UseSwagger(); app.UseSwaggerUI(options =>options.SwaggerEndpoint("/swagger/v2/swagger.json",  "MyTest Demo"));



Search for the launchsettings.json file on solutions explorer

Delete the whole file and replace with: **( note** MyTest\_Demo is the name of the solution)

{

"$schema": "http://json.schemastore.org/launchsettings.json",

"iisSettings": {

"windowsAuthentication": false,

"anonymousAuthentication": true,

"iisExpress": {

"applicationUrl": "http://localhost:60696",

"sslPort": 44320

}

},

"profiles": {

"IIS Express": {

"commandName": "IISExpress",

"launchBrowser": true,

"launchUrl": "swagger",

"environmentVariables": { "ASPNETCORE\_ENVIRONMENT": "Development" }

},

"MyTest\_Demo": {

"commandName": "Project",

"launchBrowser": true,

"launchUrl": "swagger",

"applicationUrl": "https://localhost:5001;http://localhost:5000",

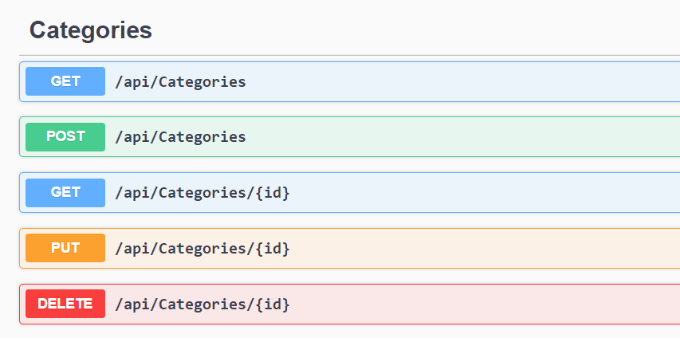
"environmentVariables": { "ASPNETCORE\_ENVIRONMENT": "Development" }

}

}

}

You can now test all the HTTP commands using the Categories table

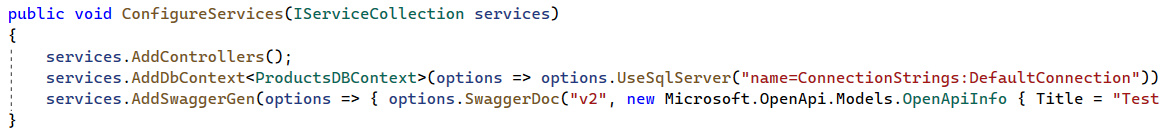




When you do your first GET, you may receive an error message relating to dependency injection.

Add the following to the startup.cs file

services.AddDbContext<ProductsDbContext>(options => options.UseSqlServer("name=ConnectionStrings:DefaultConnection"));



Look in the Models folder for the name of your dbContext – here it is ProductsDbContext

Right-click on ProductsDbContext in the line of code, choose Quick actions and refactoring and pull the dbContext in related to the Model

If all went well, you should be able to test all the HTTP commands exposed by the API, otherwise – **GOOGLE IS YOU FRIEND**, as always.

