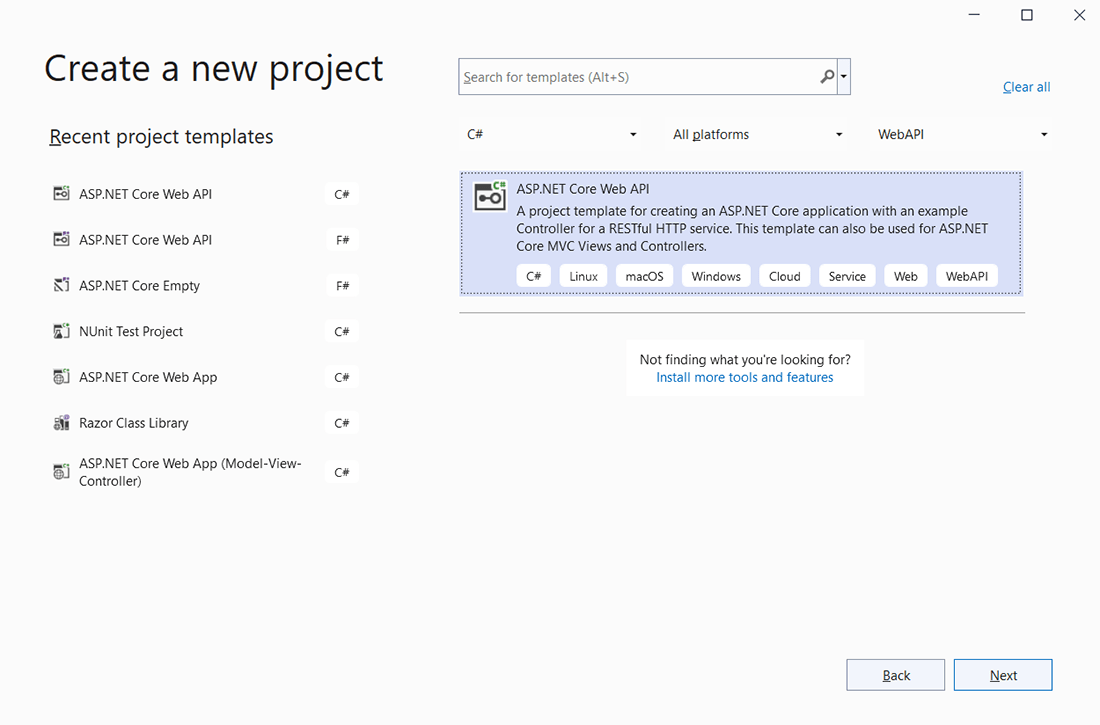
**Prerequisites: Web API in .NET 6.0**

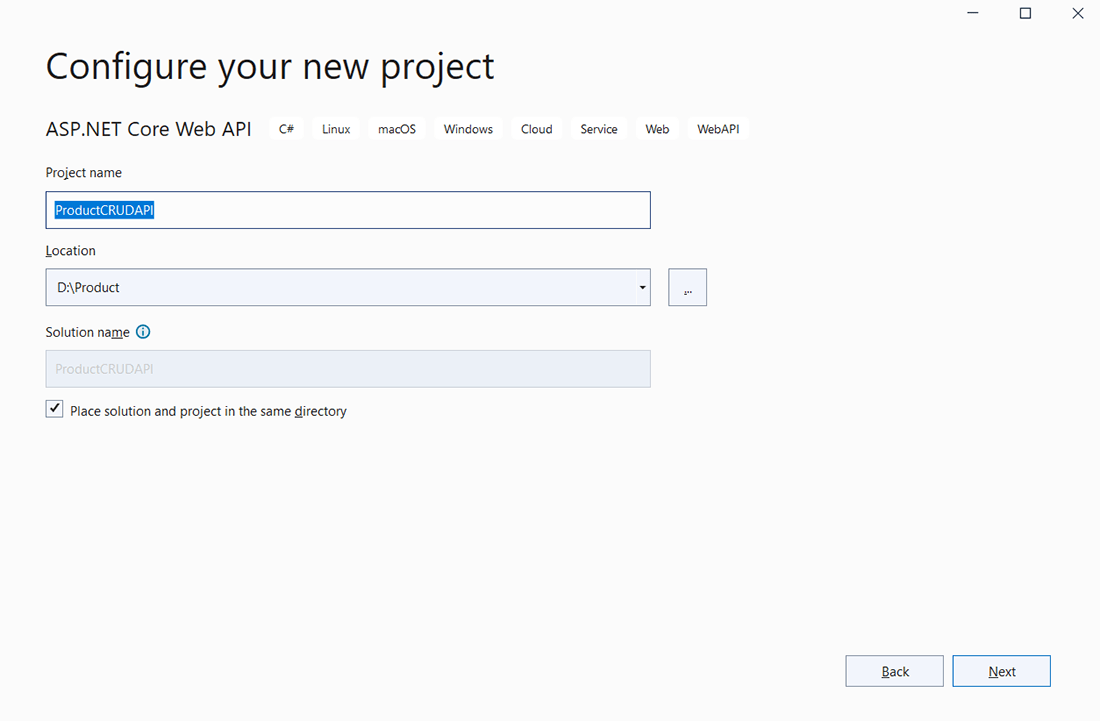
* Visual Studio 2022.
* .NET SDK 6.0.
* Sql-Server.

**Create Project**

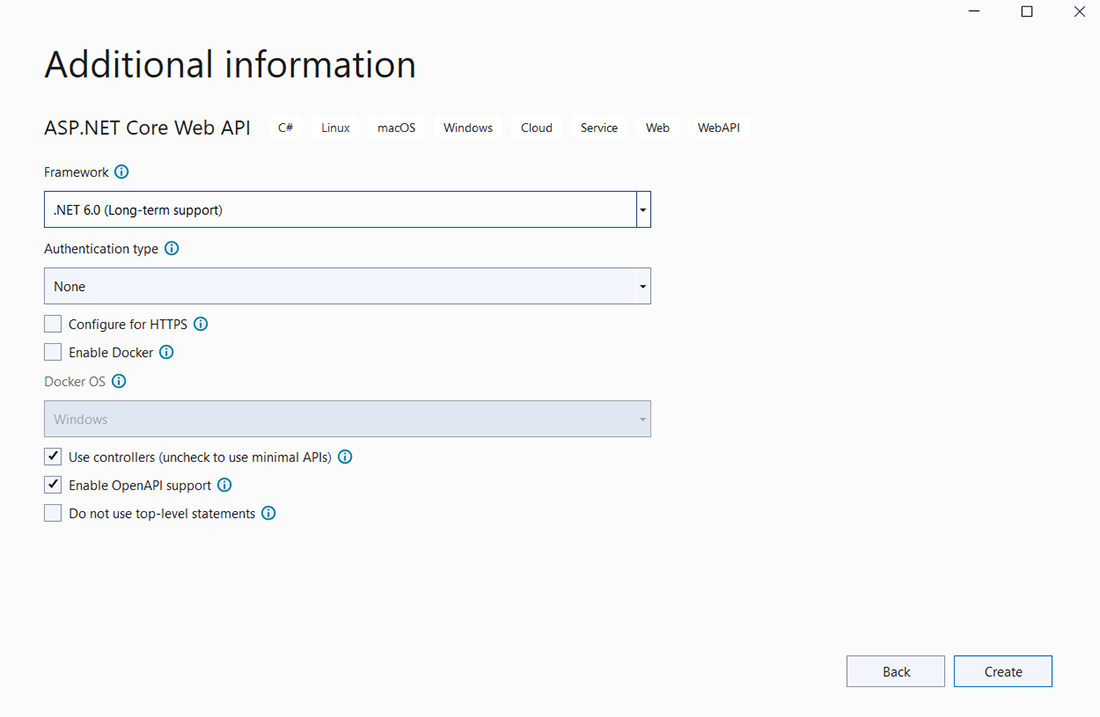
Start Visual Studio and create a new project with the type **ASP.NET Core Web API** and click Next.



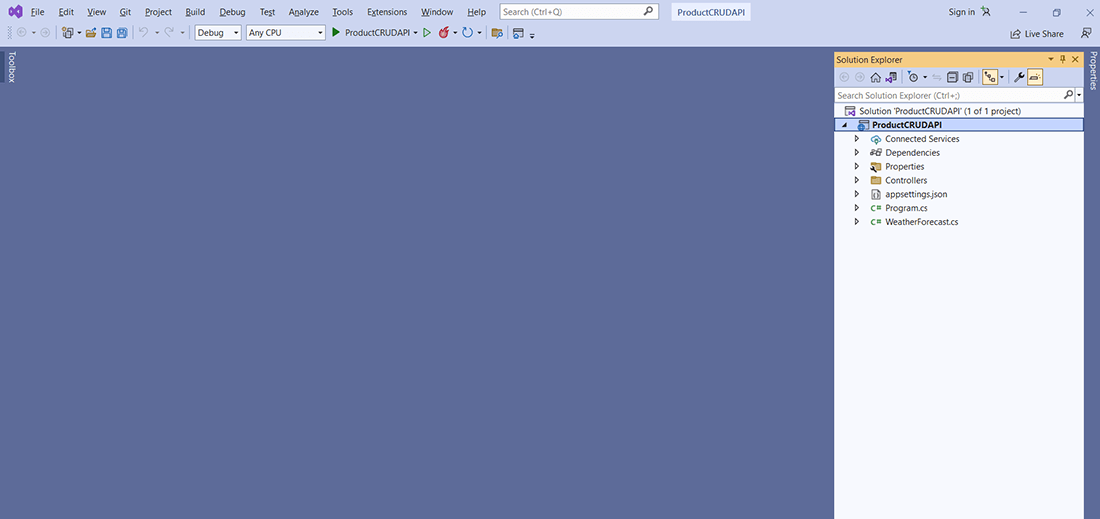
Enter the project name **ProductCrudAPI,** select the location where you want to save your project, and click **Next.**



Select **.Net 6.0** (Long-term support) as a framework. Fill in the required information as shown in the below image, and click on **Create.**



Once you click on Create, a Web API project will be created.

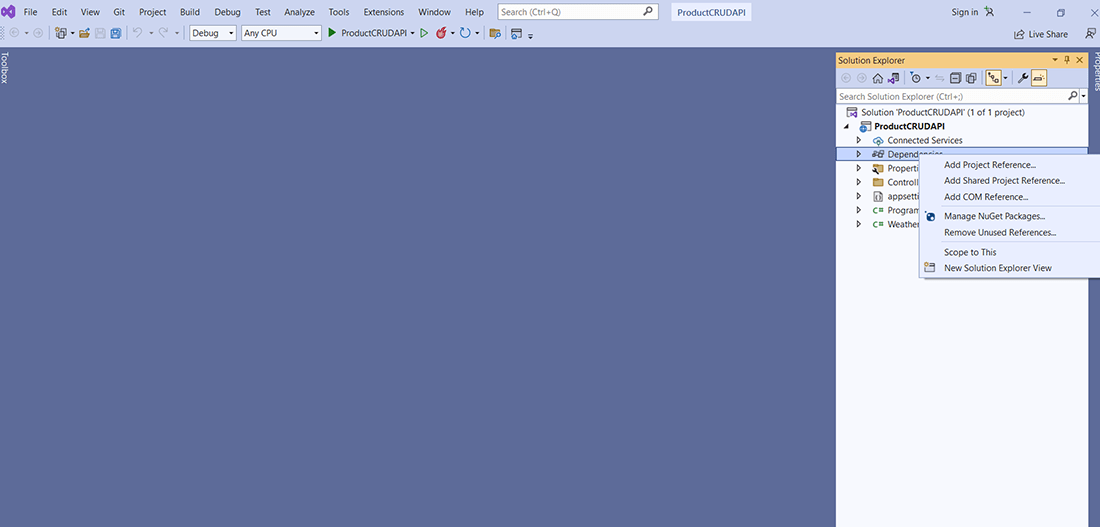


**Add NuGet Packages**

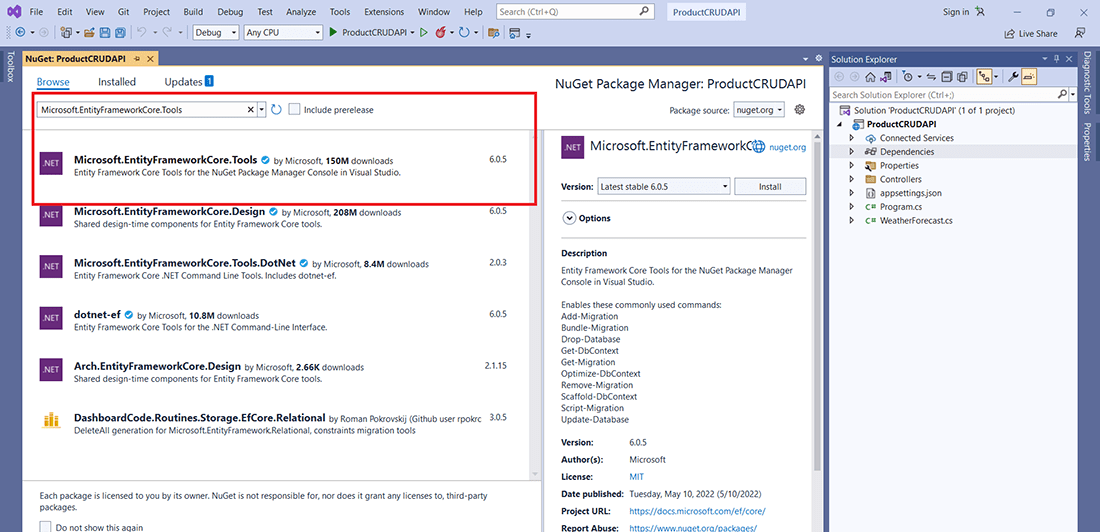
To use the entity framework in our project, we need to **install two NuGet packages:**

* + - Microsoft.EntityFrameworkCore.Tools
    - Microsoft.EntityFrameworkCore.SqlServer

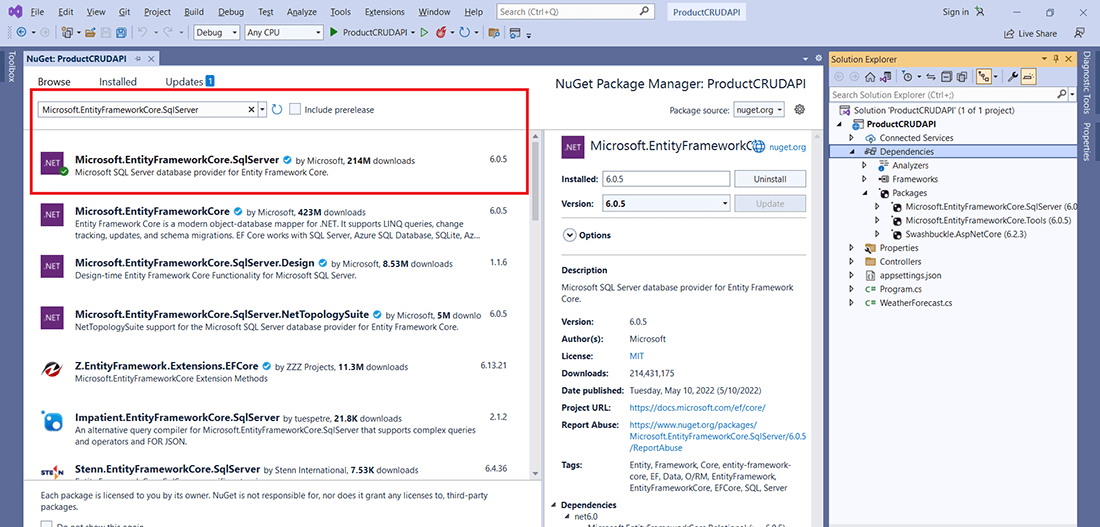
Follow the below instructions to install **NuGet packages.**  
Right-click on Dependencies and select **Manage NuGet Packages.**



**Microsoft.EntityFrameworkCore.Tools**  
Select the **Browse** tab and search for **Microsoft.EntityFrameworkCore.Tools** and install its latest stable version.



**Microsoft.EntityFrameworkCore.SqlServer**  
Once the above package is installed, Search for **Microsoft.EntityFrameworkCore.SqlServer** and install its latest stable version.



**Create SQL Database and Table.**

Moving to the next section of the Web API in .NET 6.0 Tutorial, create New Database **ProductDB** in SQL, and execute the below script to create a **Product** table.

USE [ProductDB]

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Products](

[Id] [int] IDENTITY(1,1) NOT NULL,

[Name] [varchar](50) NOT NULL,

[Description] [varchar](250) NULL,

[Price] [decimal](18, 2) NOT NULL,

PRIMARY KEY CLUSTERED

(

[Id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

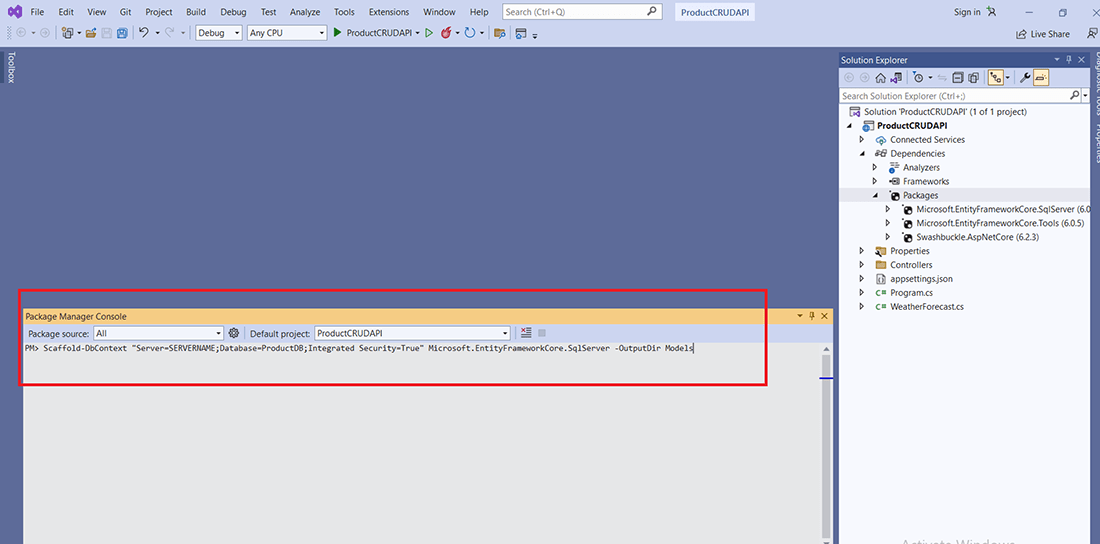
**Create DB Context and Model**

We are using the database first approach of entity framework.  
We have created a database table, and using the Scaffold-DbContext command of the entity framework; we will create the required class in the C# project.

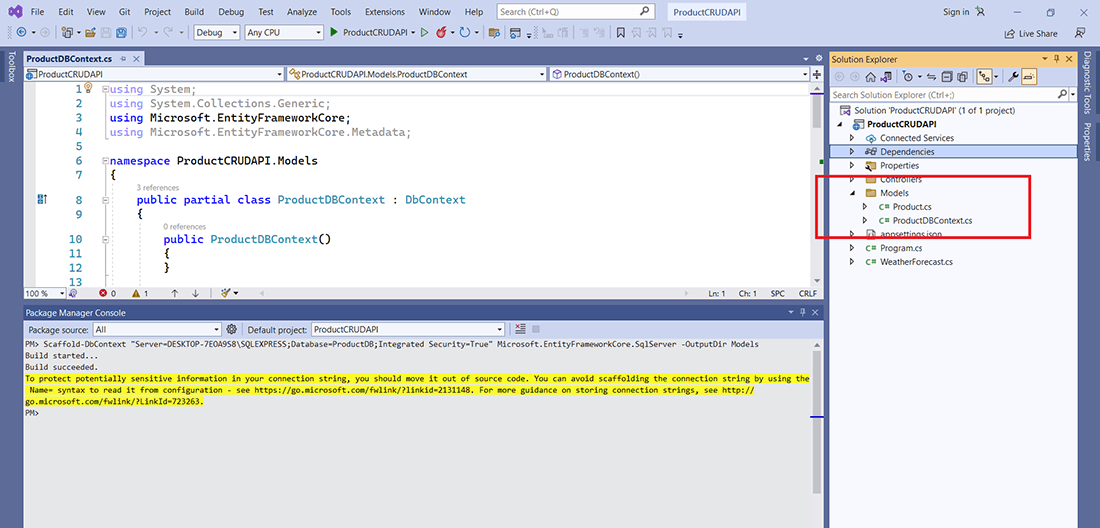
Open **Package Manager Consol** (Tool => Package Manager => Package Manager Consol) and run below command:

Scaffold-DbContext "Server=SERVERNAME;Database=ProductDB;Integrated Security=True" Microsoft.EntityFrameworkCore.SqlServer -OutputDir Models

Replace **SERVERNAME** with your database server name.



Once this command is executed, the Model folder is created in the project solution. Model folder contains two files, **ProductDBContext.cs** and **Product.cs.**



**ProductDBContext.cs**is responsible for database interaction, and **Product.cs** is a model of the Products table.

Remove **OnConfiguring()** method from ProductDBContext.cs; it contains a database connection string and is not a good practice. We will add the connection string in the appsettings.json file.

Also remove **ProductDBContext()** constructor from this file.

**Configure DB Connection**

Add database connection string in **appsettings.json**file.

{

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft.AspNetCore": "Warning"

}

},

"AllowedHosts": "\*",

"ConnectionStrings": {

"ProductDB": "Server=SERVERNAME;Database=ProductDB;Integrated Security=True;"

}

}

Replace **SERVERNAME** with your database server name.

As we are using the .Net 6 version, we need to make the required configuration changes in **Program.cs** file. Microsoft eliminates **Startup.cs** in .Net 6. In the previous .Net version, **Startup.cs** was used for configurations.

Add below lines in **Program.cs.** Please refer to the below image for this.

var connectionString = builder.Configuration.GetConnectionString("ProductDB");

builder.Services.AddDbContextPool(option =>

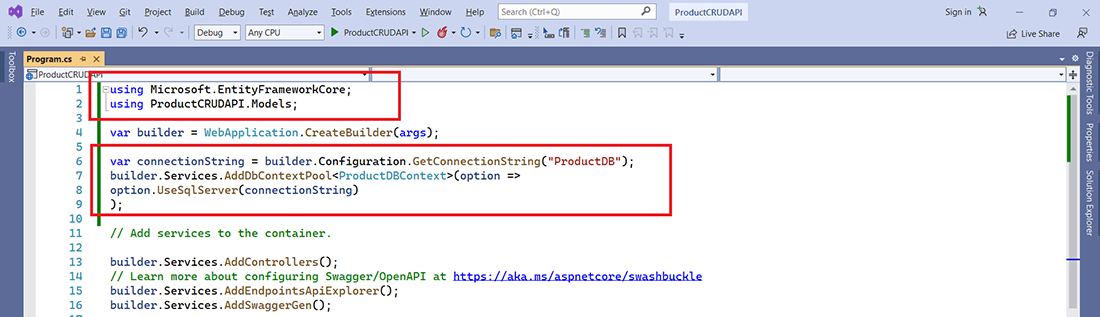
option.UseSqlServer(connectionString)

);

Also, add the below lines at the top of the **Program.cs.**

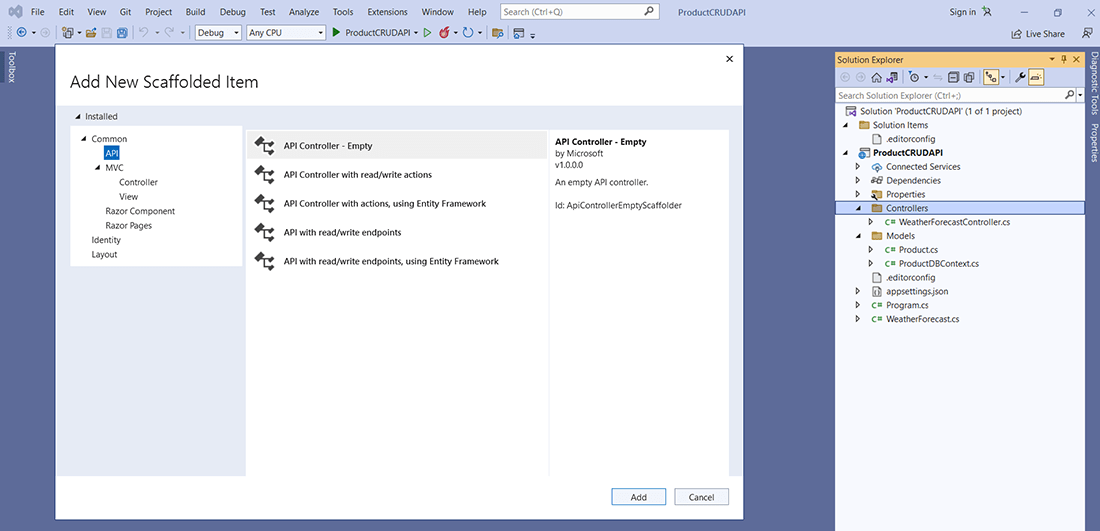
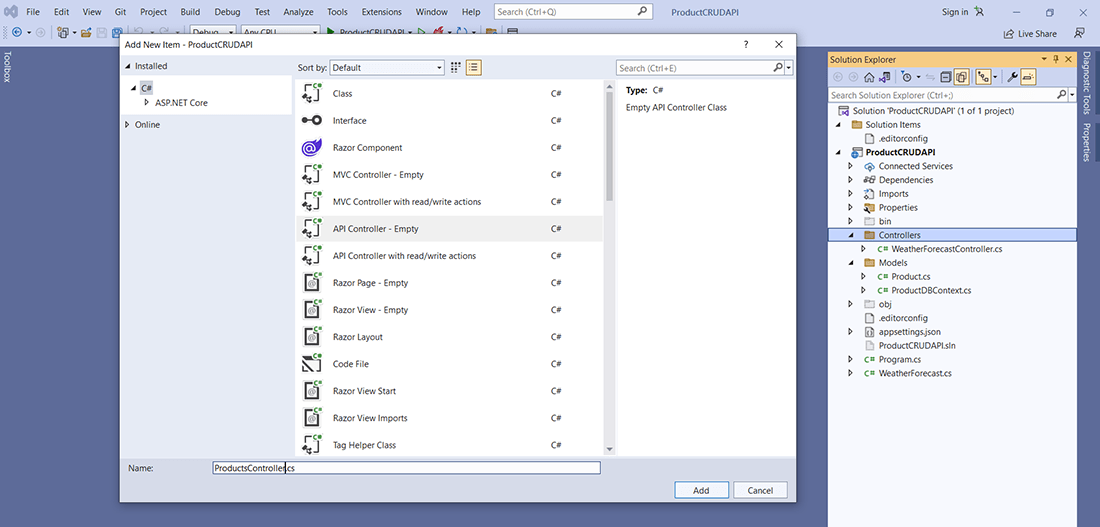
using Microsoft.EntityFrameworkCore;

using ProductAPI.Models;



**Add Products Controller**

Add a new empty API controller **ProductsController.cs** under the controller folder.

**Add Methods in ProductsController**

In **ProductsController.cs,** we will add GET, POST, PUT, and DELETE endpoints to achieve CRUD operations.

Please use the below code in your **ProductsController.cs.**

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using ProductCRUDAPI.Models;

namespace ProductCRUDAPI.Controllers

{

[Route("api/[controller]")]

[ApiController]

public class ProductsController : ControllerBase

{

private readonly ProductDBContext \_context;

public ProductsController(ProductDBContext context)

{

\_context = context;

}

[HttpGet]

public async Task<ienumerable> Get()

{

return await \_context.Products.ToListAsync();

}

[HttpGet("{id}")]

public async Task Get(int id)

{

var product = await \_context.Products.FirstOrDefaultAsync(m => m.Id == id);

if (product == null)

return NotFound();

return Ok(product);

}

[HttpPost]

public async Task Post(Product product)

{

\_context.Add(product);

await \_context.SaveChangesAsync();

return Ok();

}

[HttpPut]

public async Task Put(Product productData)

{

if (productData == null || productData.Id == 0)

return BadRequest();

var product = await \_context.Products.FindAsync(productData.Id);

if (product == null)

return NotFound();

product.Name = productData.Name;

product.Description = productData.Description;

product.Price = productData.Price;

await \_context.SaveChangesAsync();

return Ok();

}

[HttpDelete("{id}")]

public async Task Delete(int id)

{

var product = await \_context.Products.FindAsync(id);

if (product == null) return NotFound();

\_context.Products.Remove(product);

await \_context.SaveChangesAsync();

return Ok();

}

}

}

**Launch API**

Finally, we are done with Web API in .NET 6.0 tutorial. Now, it’s time to launch this API, press F5. As we are using Swagger UI, we can execute API directly.

We can see GET, POST, PUT AND DELETE under Products. We can execute different API methods from this page itself.

