Asp.Net MVC and Security

unit4

Middleware Request Pipeline

In ASP.NET Core, the request pipeline is constructed using middleware. Middleware components are small, independent components that handle specific tasks in the request processing pipeline. The request pipeline consists of a series of middleware components, each responsible for processing an incoming HTTP request and possibly generating a response. Here's an overview of how the request pipeline works in ASP.NET Core:

Middleware is software that sits between an operating system and the applications running on it. In the context of web development, middleware is code that acts as a bridge between the client and the application logic. It intercepts requests and responses, allowing developers to perform various tasks such as logging, authentication, authorization, error handling, and more.

Middleware

- ✓ Asp.Net Core create an HTTP application pipeline that processes the request.
- ✓ This Http Pipeline is configures in **Configure method** of **Startup.cs**
- ✓ All the request to the application goes through the HTTP pipeline
- ✓ An middleware is a piece of code (component) which is used in Http pipeline.

Middleware are software components that are assembled into an application pipeline to handle requests and responses.

Each component chooses whether to pass the request on to the next component in the pipeline, and can perform certain actions before and after the next component is invoked in the pipeline.

Request delegates are used to build the request pipeline. The request delegates handle each HTTP request.

Each piece of middleware in ASP.NET Core is an object, and each piece has a very specific, focused, and limited role.

Ultimately, we need many pieces of middleware for an application to behave appropriately.

Uses of Middleware

Here are a few common use cases for middleware:

Logging: Middleware can log information about incoming requests, outgoing responses, errors, and other events, helping developers monitor and debug their applications.

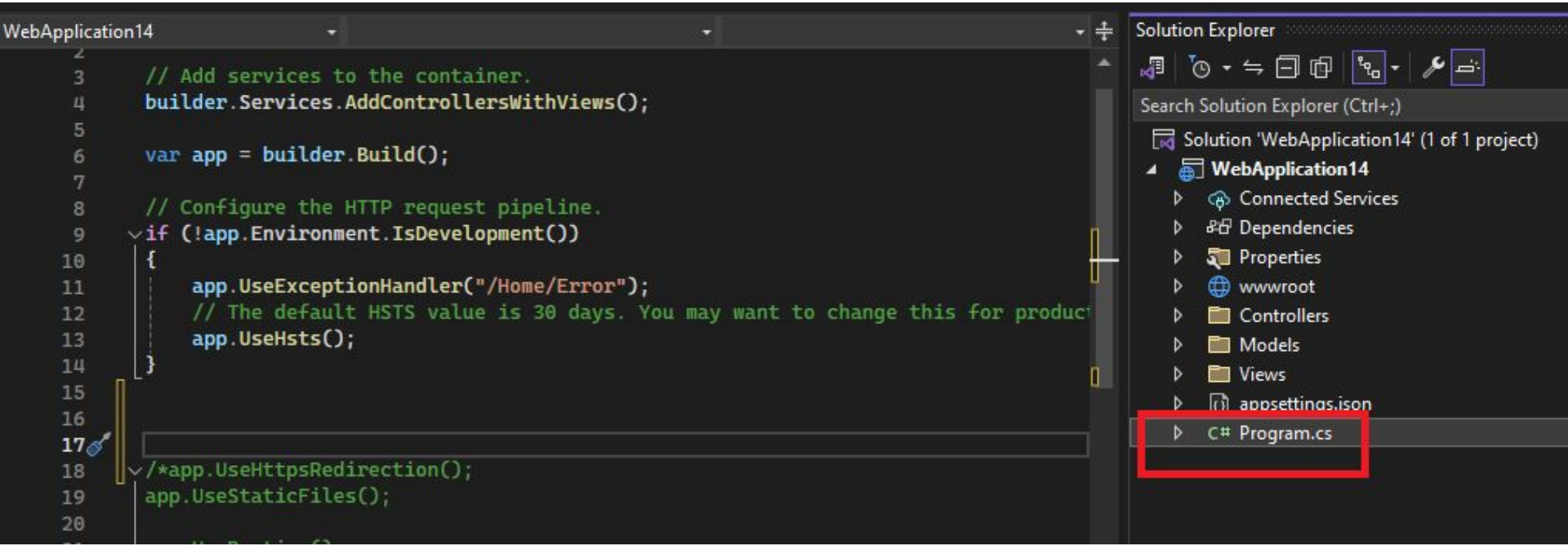
Authentication and Authorization: Middleware can enforce authentication requirements by verifying user credentials or access tokens. It can also enforce authorization rules by checking whether users have the necessary permissions to access certain resources.

Error Handling: Middleware can catch errors thrown by other parts of the application and return appropriate error responses to clients. It can also log error details for troubleshooting purposes.

Caching: Middleware can cache responses to improve performance and reduce server load. It can store responses in memory, on disk, or in an external caching service.

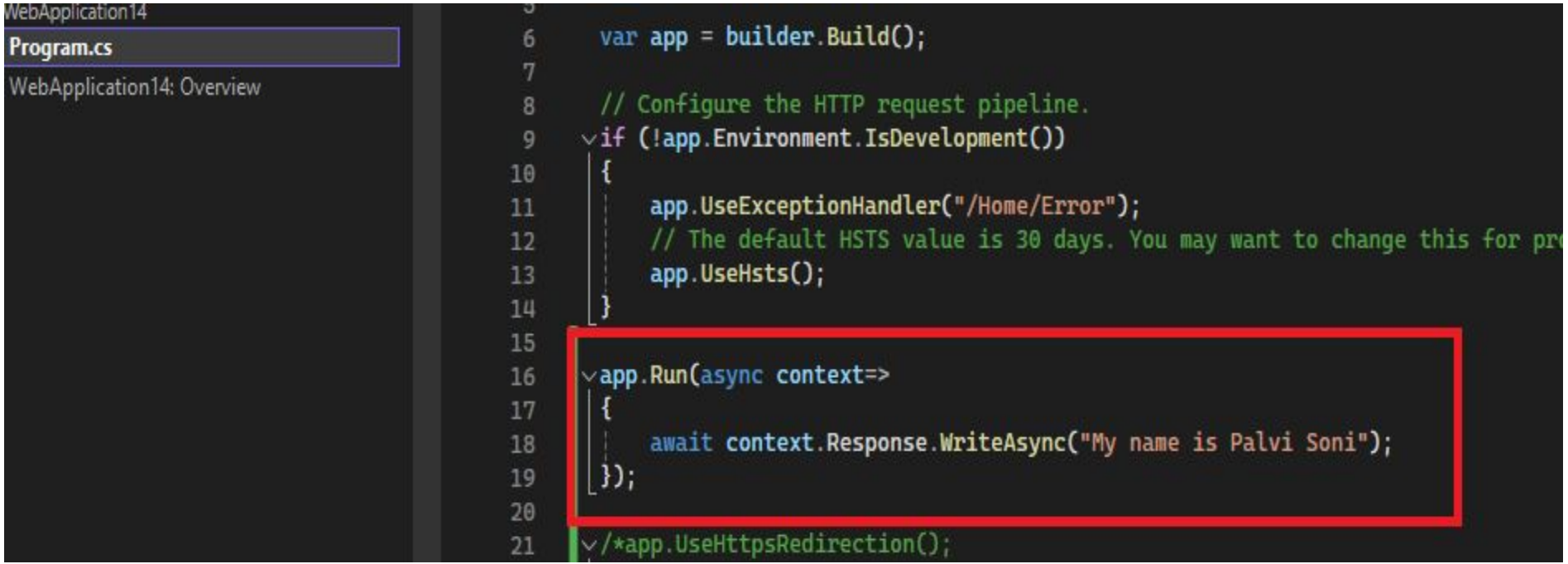
Middleware provides a flexible and modular way to extend the functionality of web applications without cluttering the main application logic. Developers can compose middleware functions into chains or pipelines, allowing them to mix and match different pieces of functionality as needed. This promotes code reuse, maintainability, and separation of concerns.

Built-In Middleware



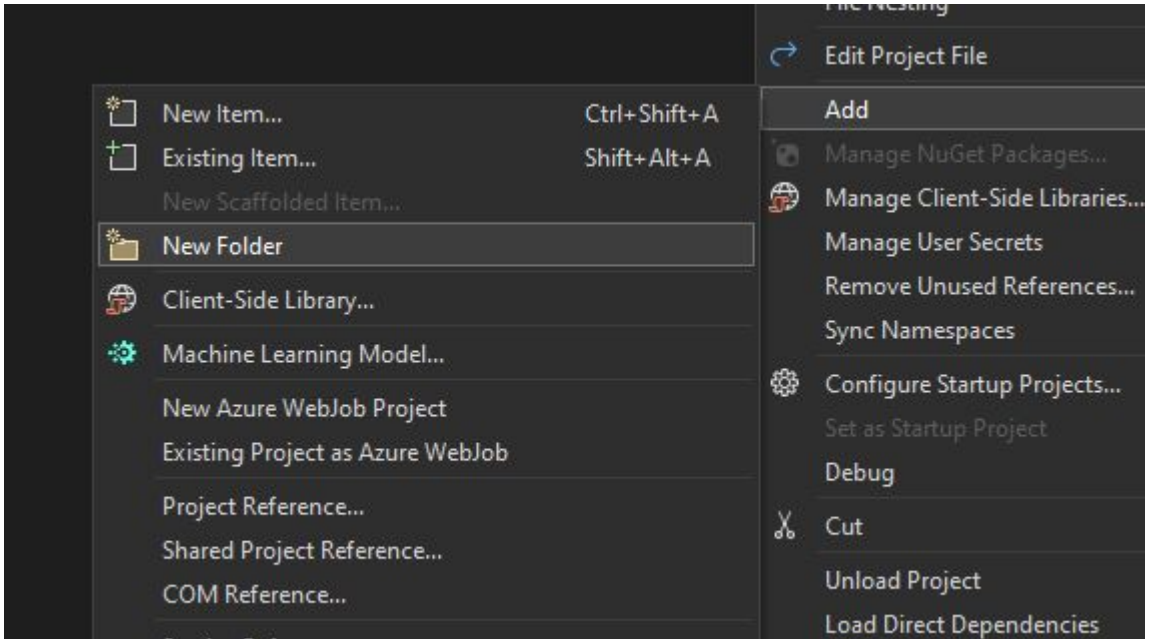
Built-in Middleware

Open program.cs file



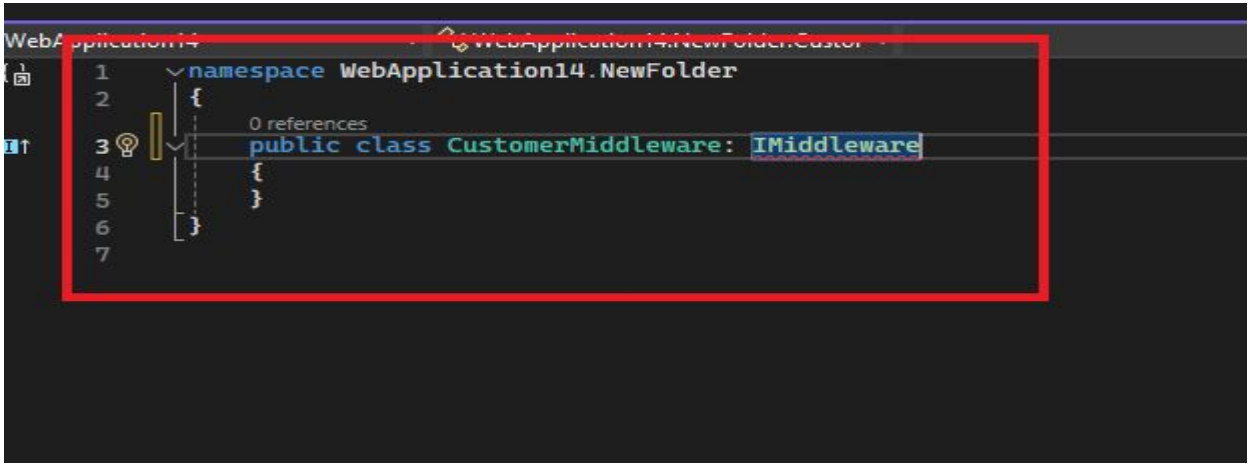
Custom Middleware

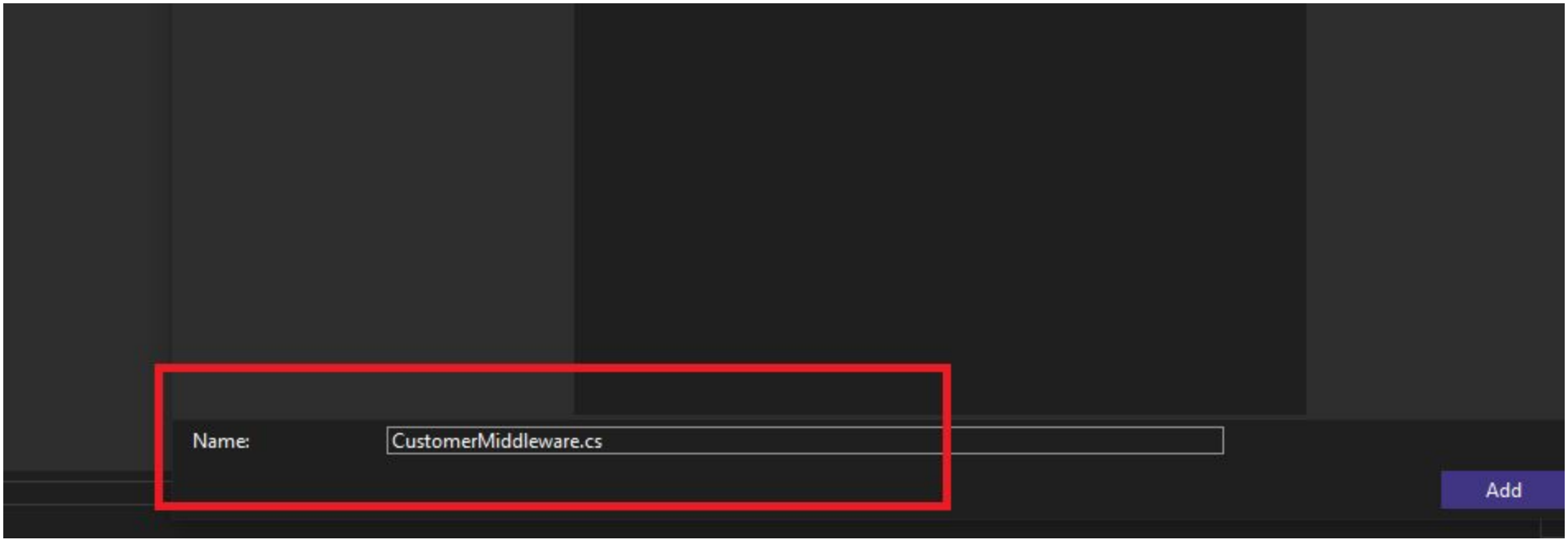
Create one folder first

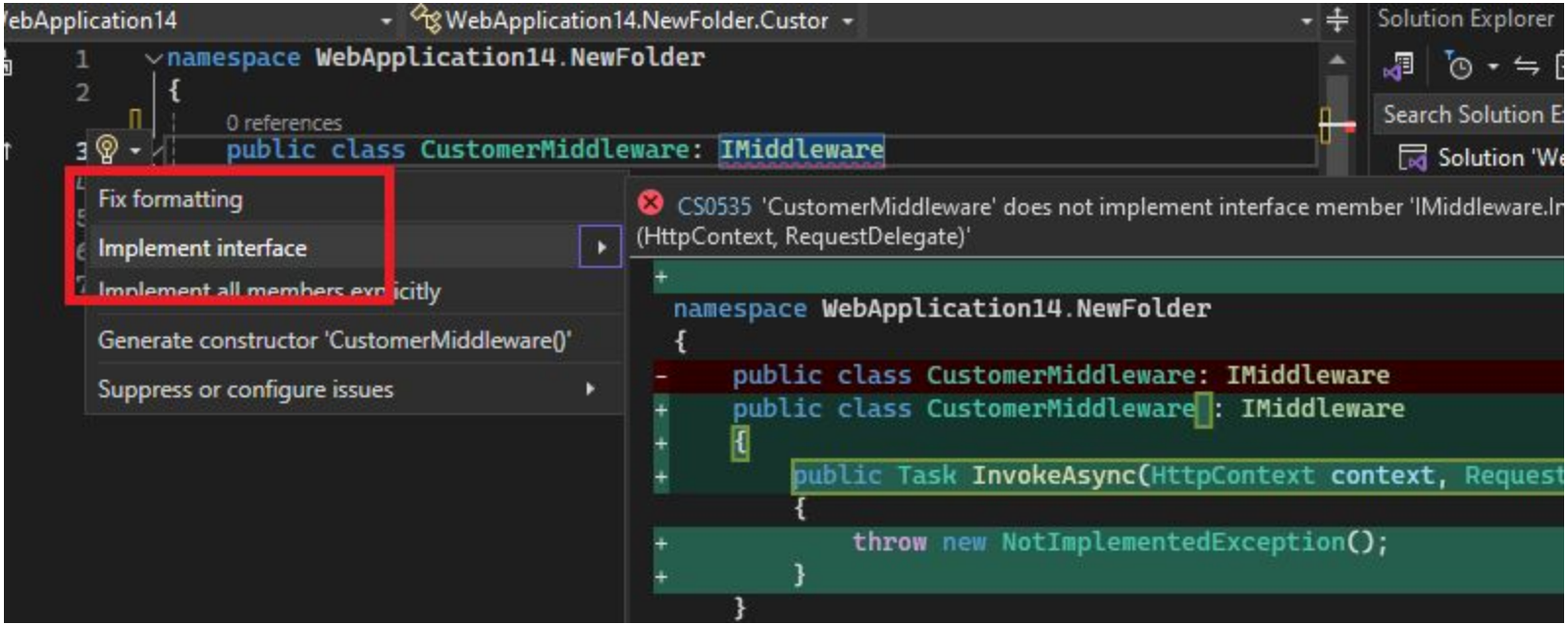


Custom Middleware

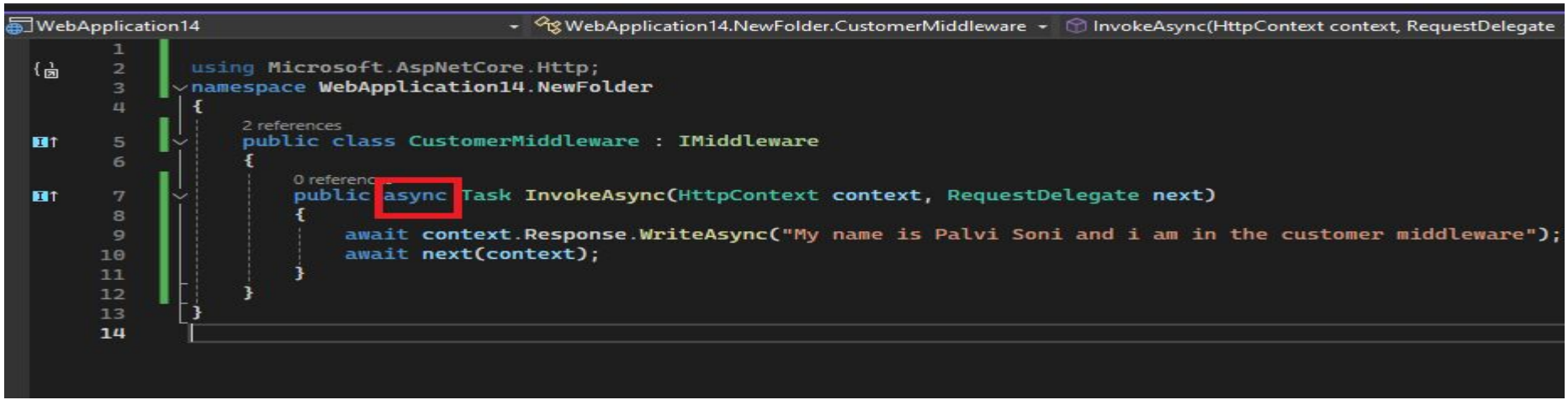
Create one model class inside that with name “CustomMiddleware” and inherit IMiddleware



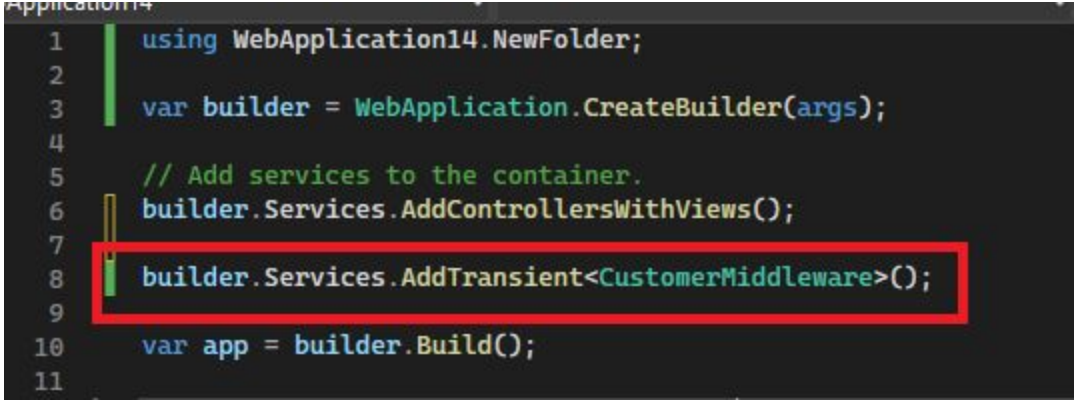

Name:
CustomerMiddleware.cs
Add


ebApplication14
WebApplication14.NewFolder.Custor
Solution Explorer
Search Solution E
Solution 'We
1 namespace WebApplication14.NewFolder
2 {
0 references
3 public class CustomerMiddleware: IMiddleware
Fix formatting
Implement interface
Implement all members exp icitly
Generate constructor 'CustomerMiddleware()'
Suppress or configure issues
CS0535 'CustomerMiddleware' does not implement interface member 'IMiddleware.In
(HttpContext, RequestDelegate)'
namespace WebApplication14.NewFolder
{
- public class CustomerMiddleware: IMiddleware
+ public class CustomerMiddleware : IMiddleware
+ {
+ public Task InvokeAsync(HttpContext context, Request
{
+ throw new NotImplementedException();
+ }
}

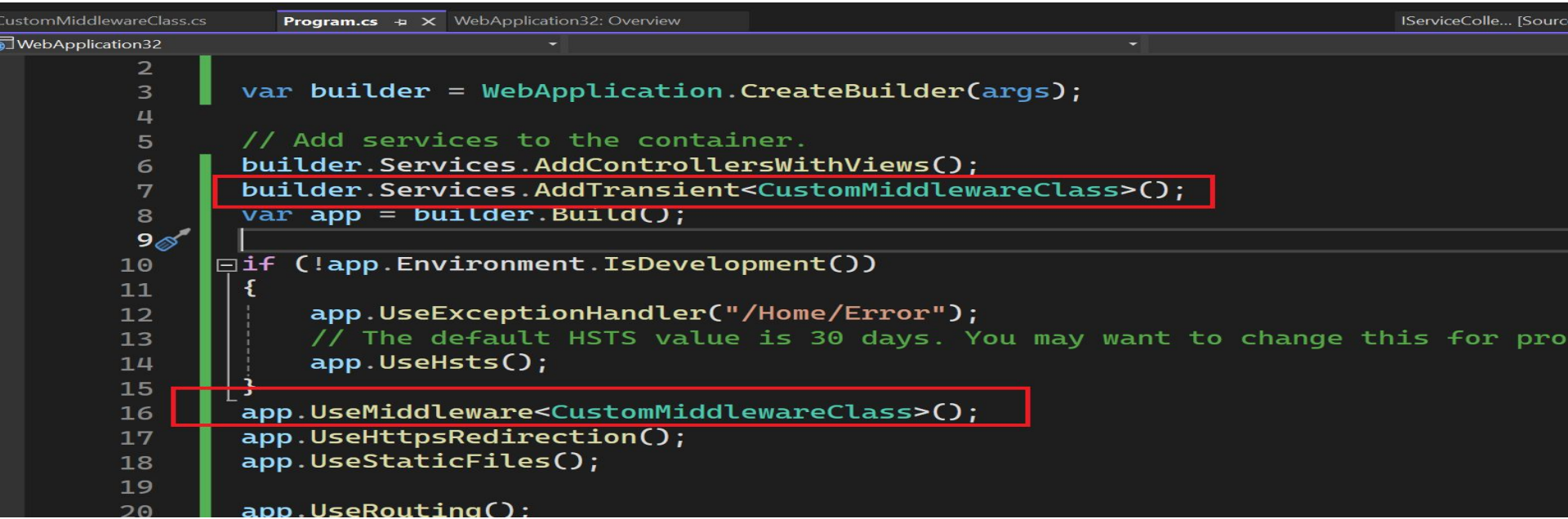
Invoke the request delegate, make sure you don’t have to use await next(context), when you are not sending request to other middleware. You can comment that

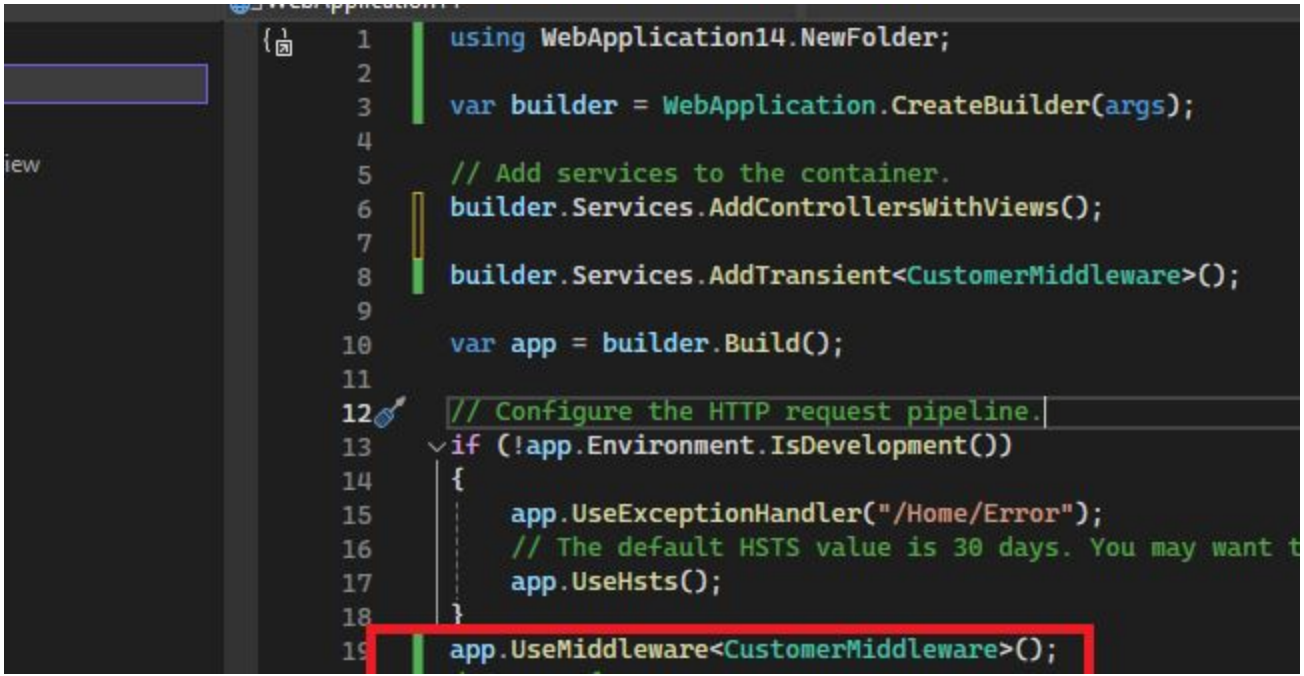


Add Transient for your customermiddleware

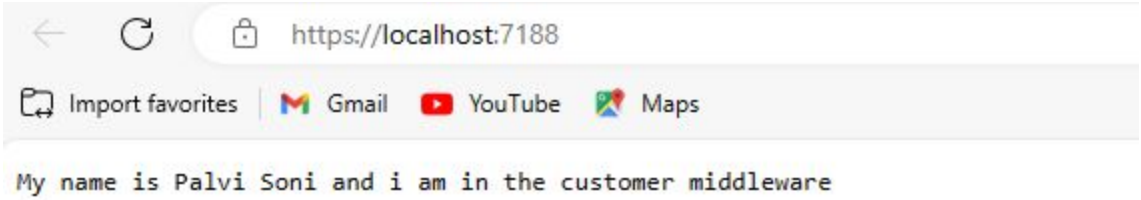


Finally make use of app.UseMiddleware for Customermiddleware class in program.cs

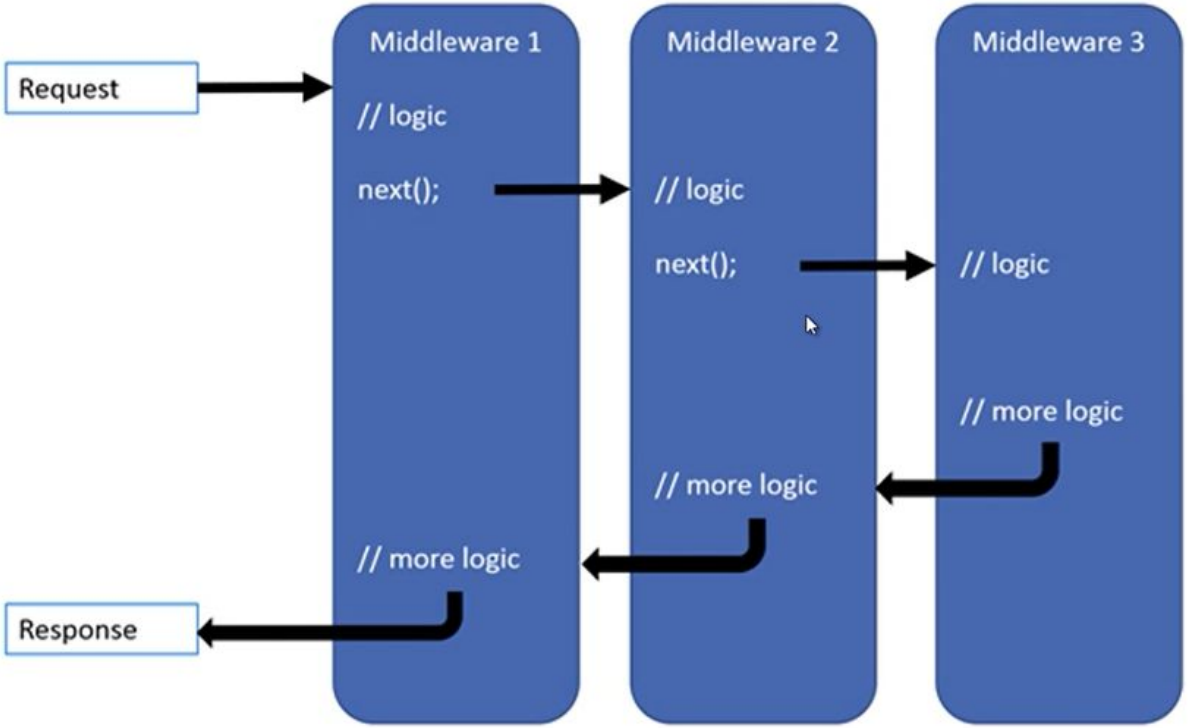



using WebApplication14.NewFolder;
var builder = WebApplication.CreateBuilder(args);
// Add services to the container.
builder.Services.AddControllersWithViews();
builder.Services.AddTransient<CustomerMiddleware>();
var app = builder.Build();
// Configure the HTTP request pipeline.
if (!app.Environment.IsDevelopment())
{
app.UseExceptionHandler("/Home/Error");
// The default HSTS value is 30 days. You may want t
app.UseHsts();
}
app.UseMiddleware<CustomerMiddleware>();

Output for customer middleware- (Custom Middleware)



DotNet Core Application Pipeline



Dependency Injection

Dependency Injection (DI) is a design pattern used in software development to achieve inversion of control (IoC) between classes and their dependencies. In simpler terms, it's a technique where one object supplies the dependencies of another object, rather than the dependent object creating them itself. This promotes loose coupling between classes, making the code more modular, testable, and maintainable.

Here's how dependency injection typically works:

Dependencies: In any application, there are classes or components that depend on other classes or components to perform certain tasks. These dependencies can be other classes, services, or configurations.

Injection: Rather than hardcoding the creation of these dependencies within the dependent class, they are passed to the dependent class from an external source. This external source could be a container, framework, or manually written code.

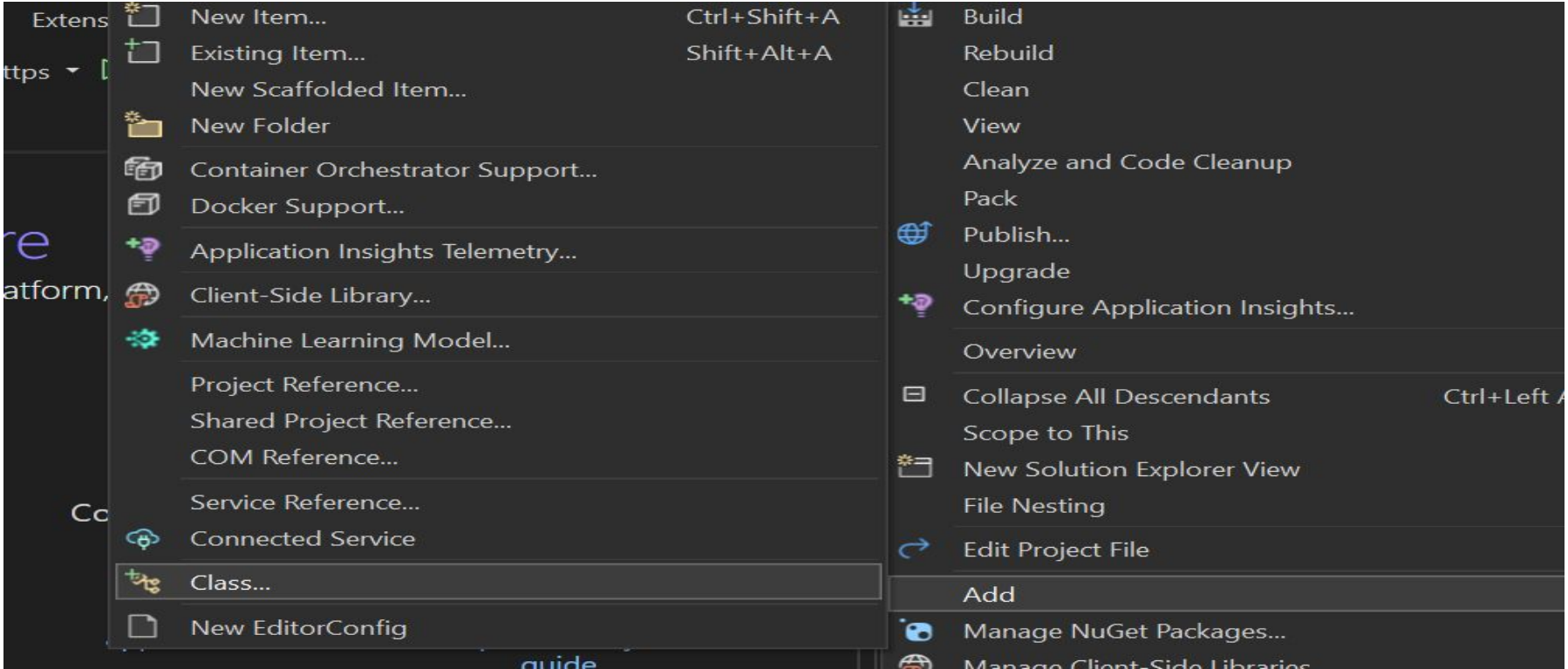
Inversion of Control (IoC): With DI, the control over the creation and lifecycle of dependencies is inverted. Instead of the dependent class creating its dependencies, it receives them from an external source, thus decoupling it from the specifics of how those dependencies are created.

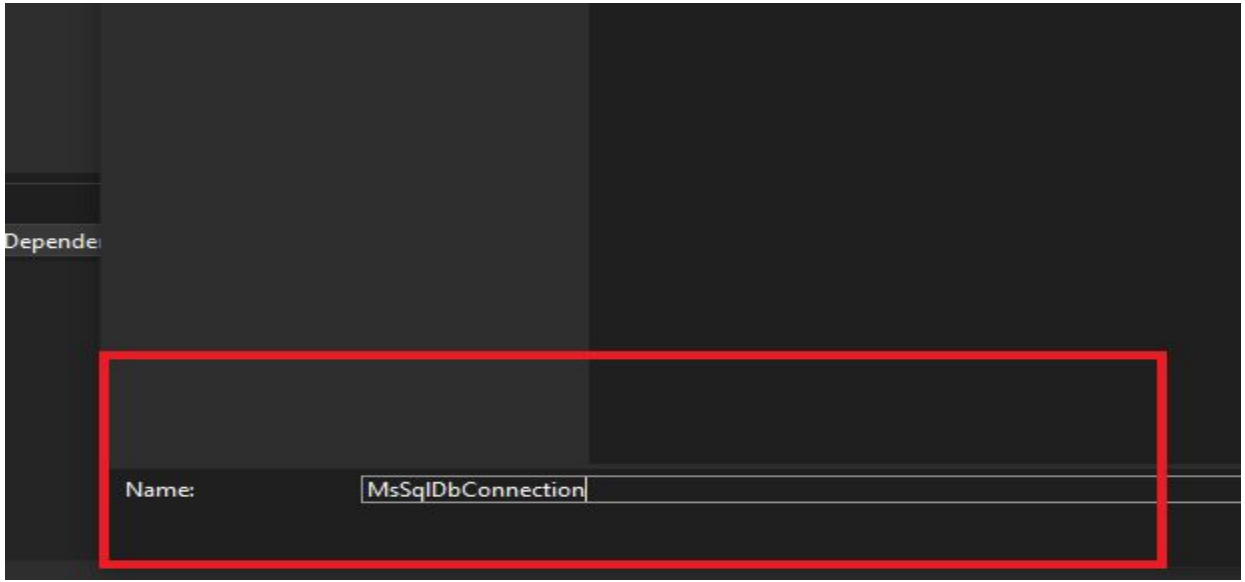
Implementation Ways

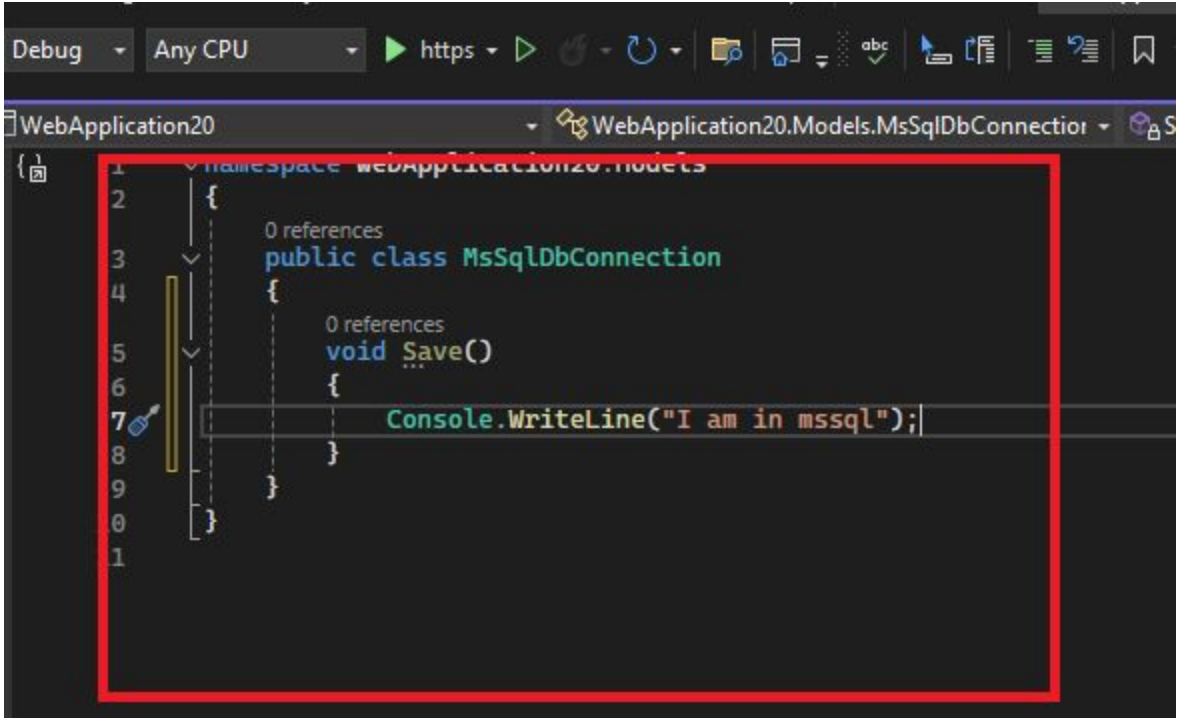
Dependency injection can be implemented in several ways:

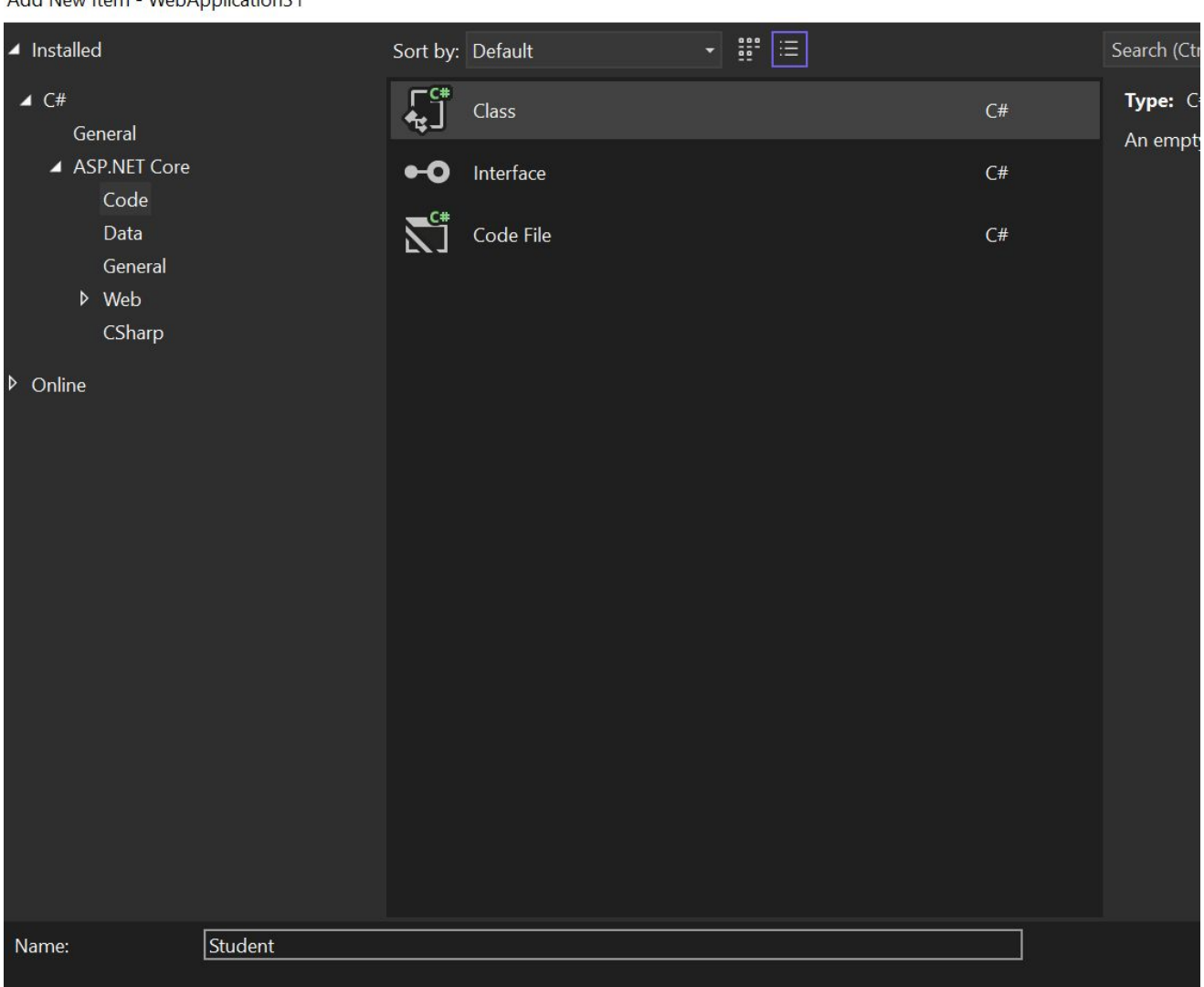
- Constructor Injection: Dependencies are provided to the dependent class via its constructor. This is one of the most common forms of DI.
- Setter Injection: Dependencies are provided via setter methods on the dependent class. This allows for optional dependencies and can be useful in certain scenarios.
- Interface Injection: The dependent class implements an interface which defines methods for setting dependencies. This is less common and more explicit than other forms of DI.

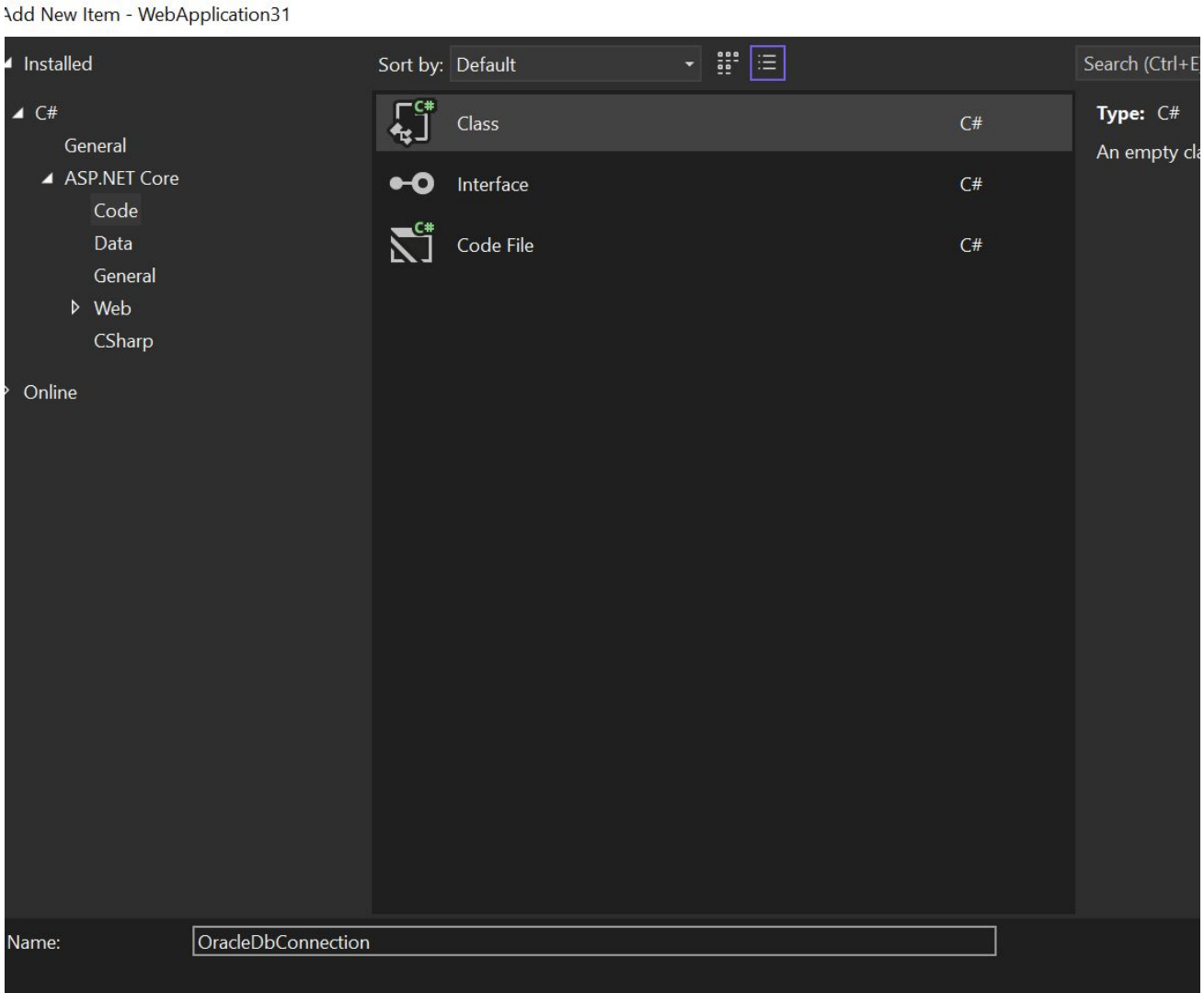
For dependency,create three model classes,student,MySqlDbConnection,OracleDbConnection

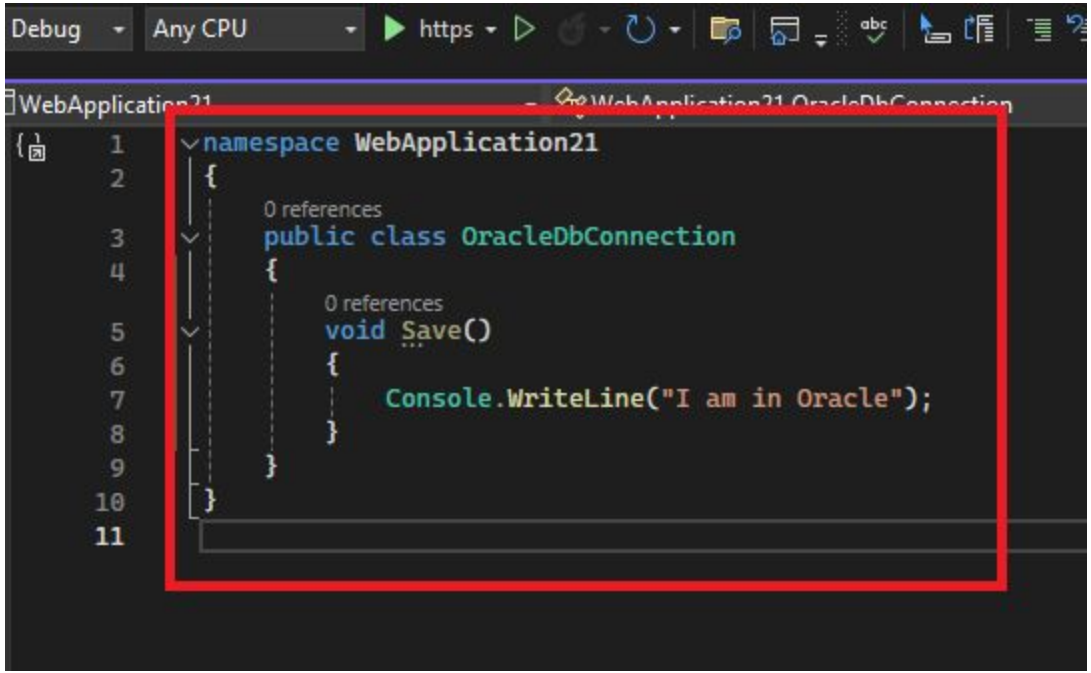



Depende
Name:
MsSqlDbConnection

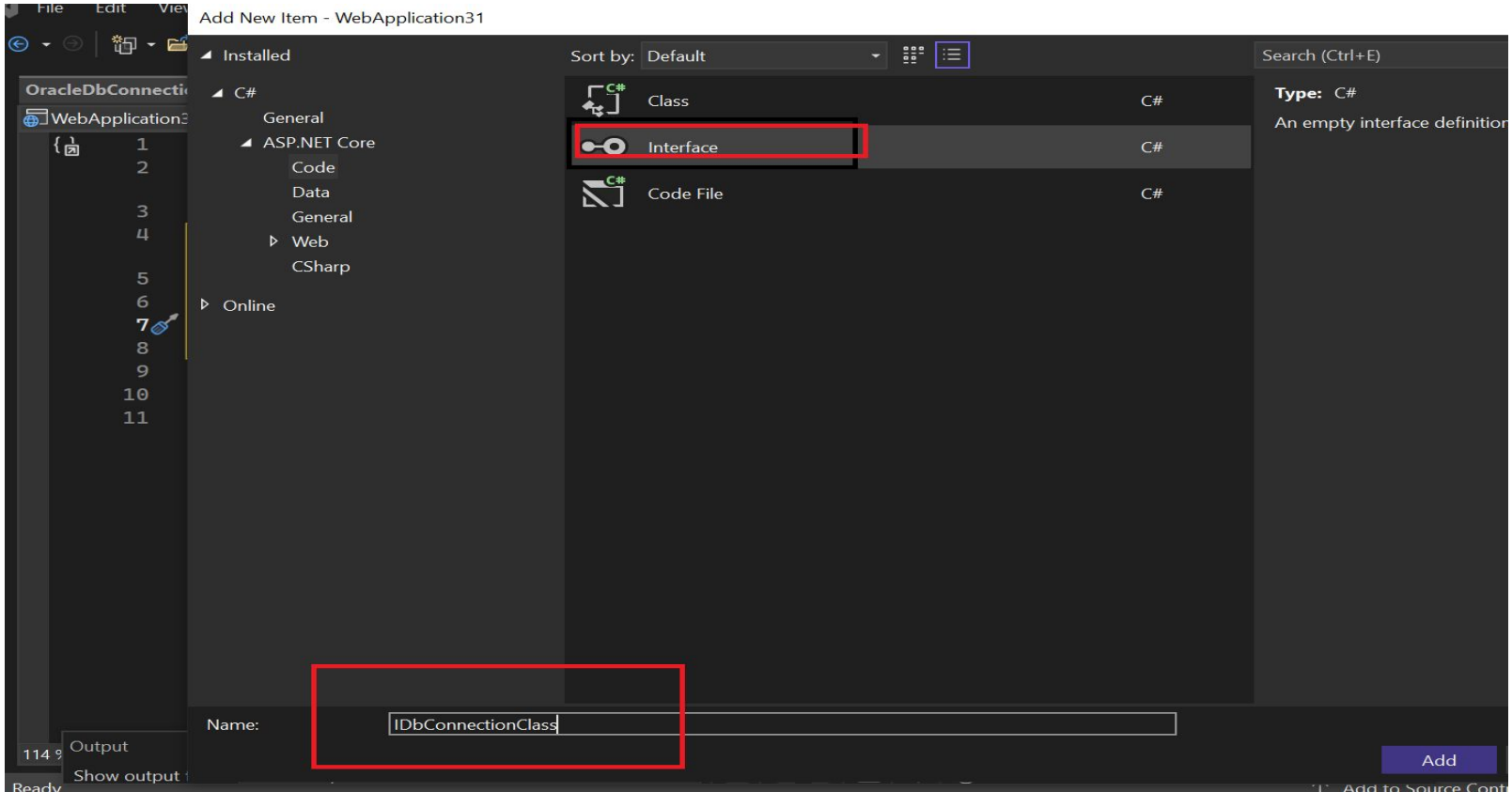

Debug
Any CPU
https
WebApplication20
WebApplication20.Models.MsSqlDbConnectio
{
0 references
public class MsSqlDbConnection
{
0 references
void Save()
{
Console.WriteLine("I am in mssql");
}
}
}


Add New Item - WebApplication31
Installed
C#
General
ASP.NET Core
Code
Data
General
Web
CSharp
Online
Sort by: Default
Class C#
Interface C#
Code File C#
Type: C
An empty
Name: Student

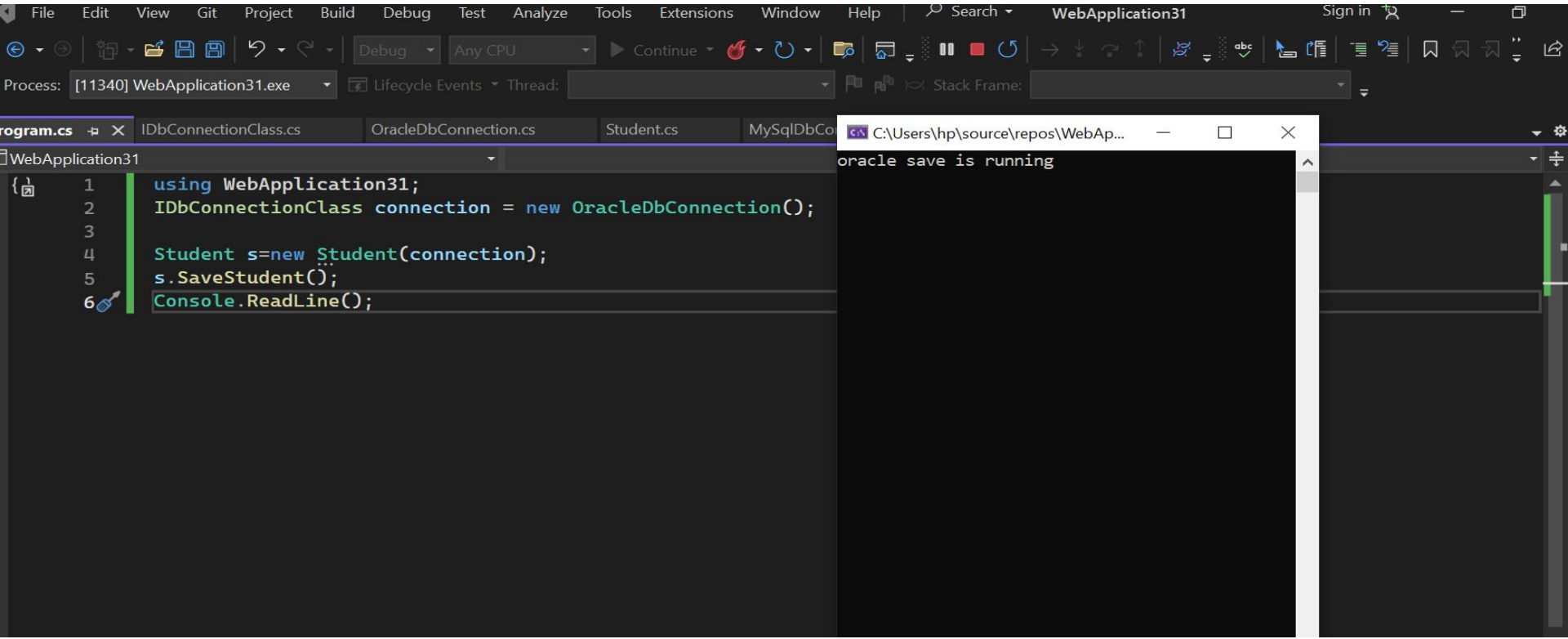

Add New Item - WebApplication31
Installed
C#
General
ASP.NET Core
Code
Data
General
Web
CSharp
Online
Sort by:
Default
Search (Ctrl+E
Class
C#
Interface
C#
Code File
C#
Type: C#
An empty cl
Name:
OracleDbConnection

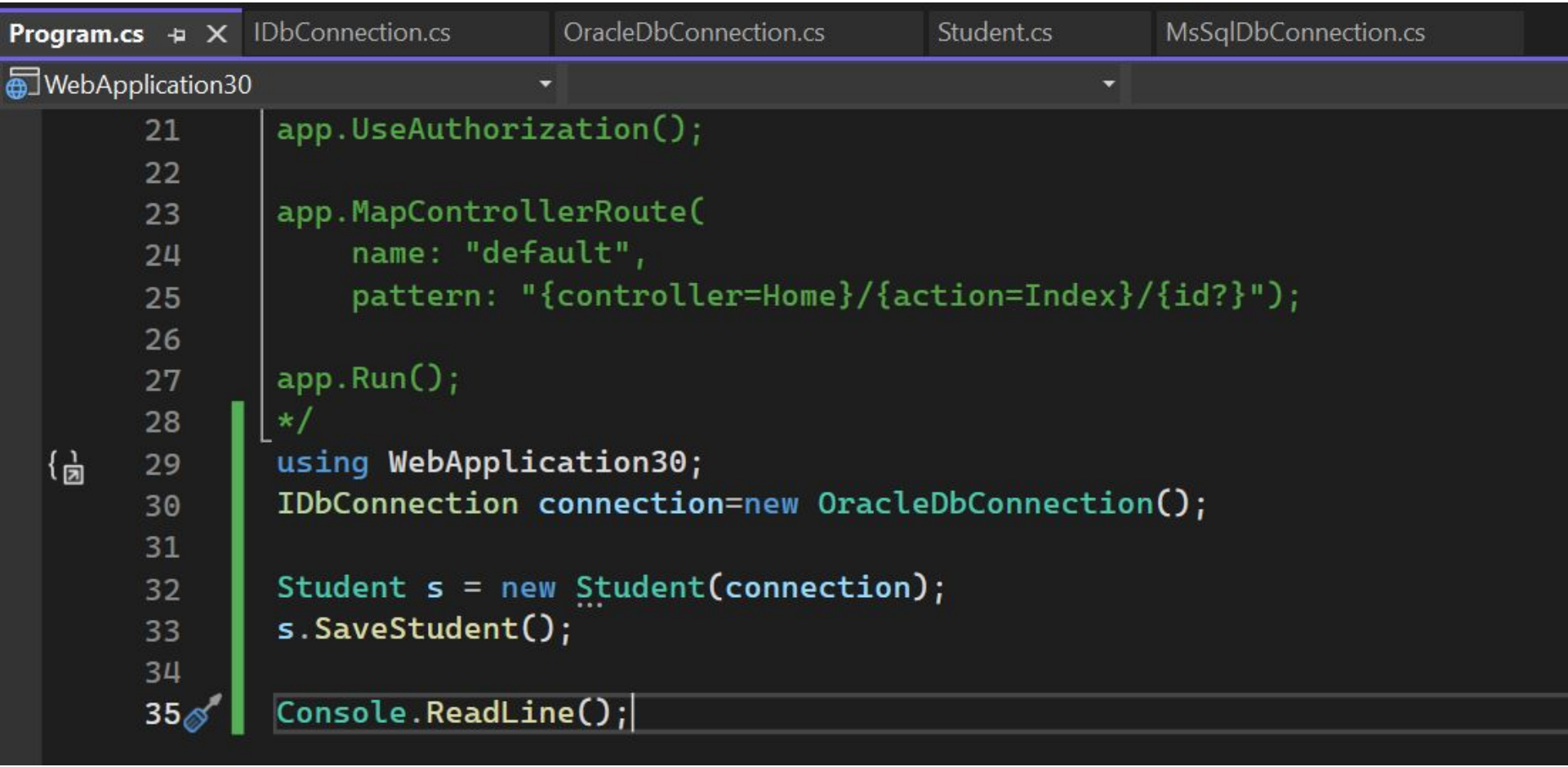

Debug
Any CPU
https
WebApplication21
namespace WebApplication21
{
0 references
public class OracleDbConnection
{
0 references
void Save()
{
Console.WriteLine("I am in Oracle");
}
}
}

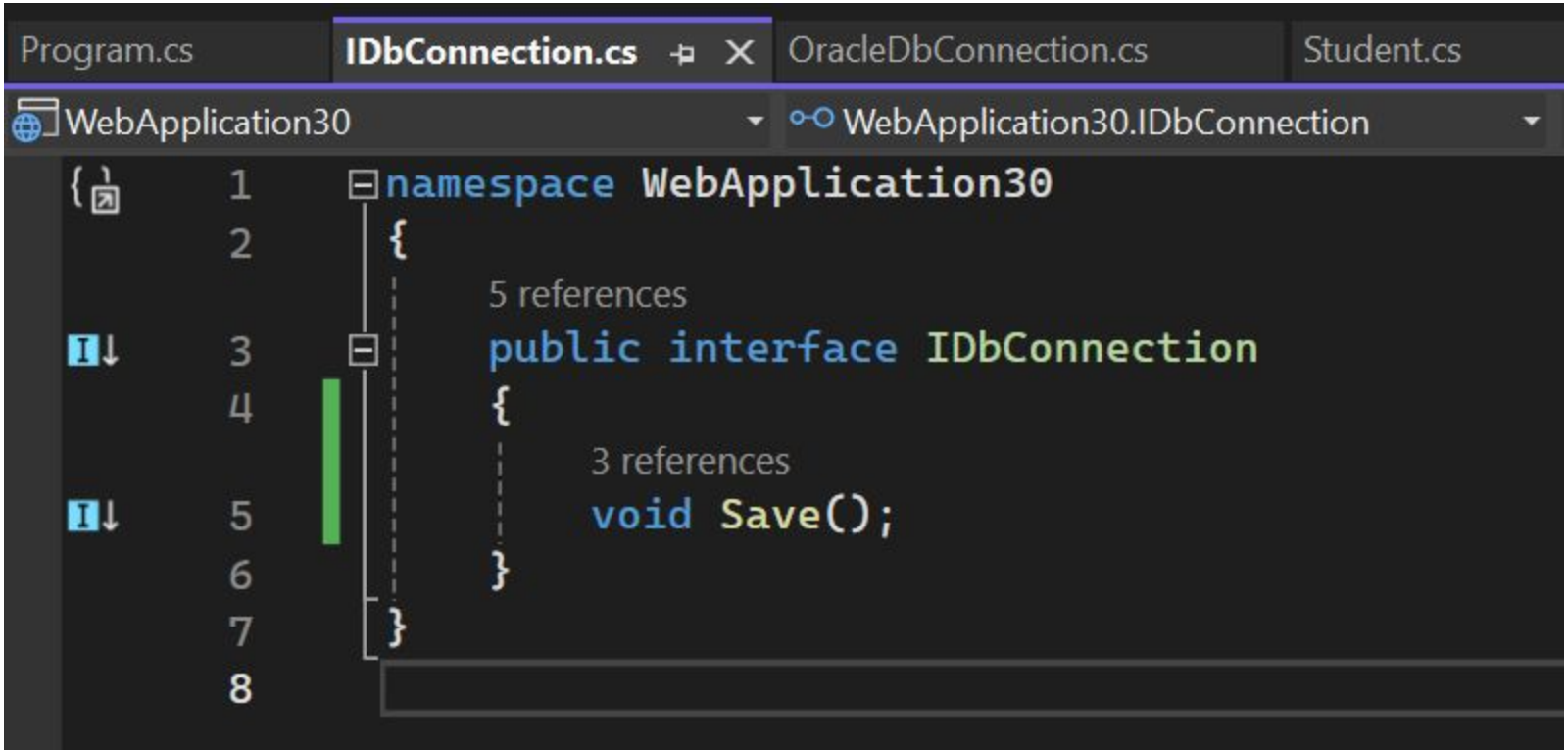
Finally,create one interface with name- IDbConnection

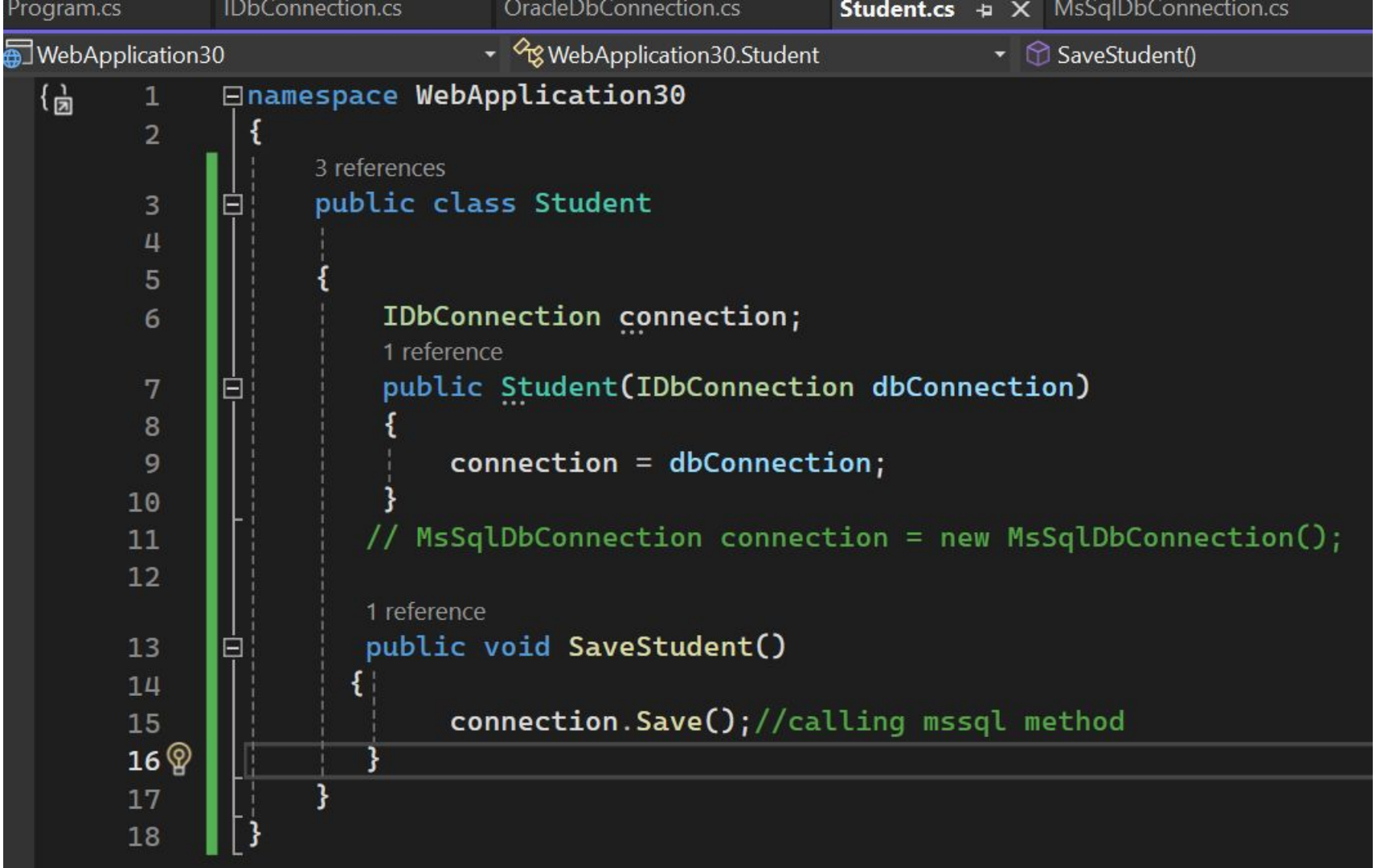


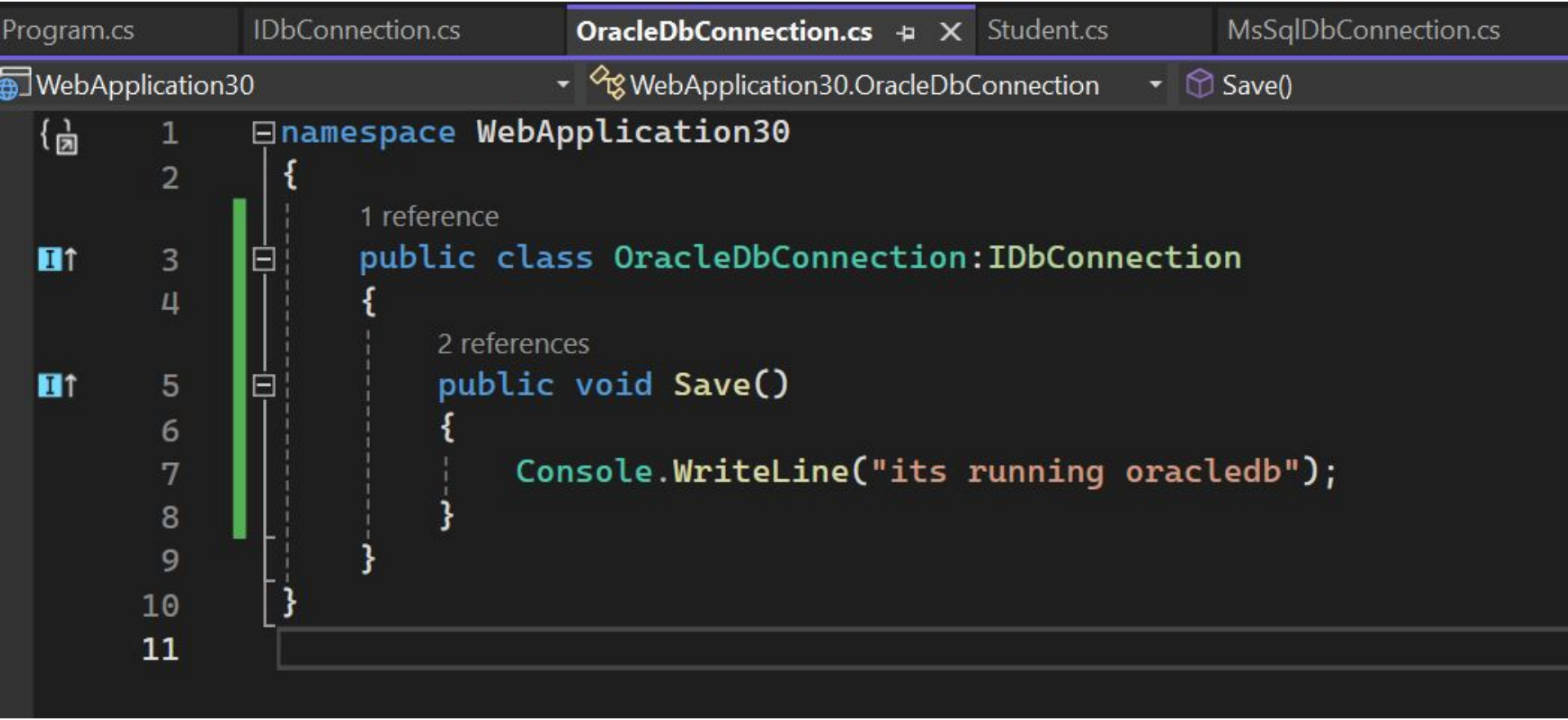
Finally,open program.cs

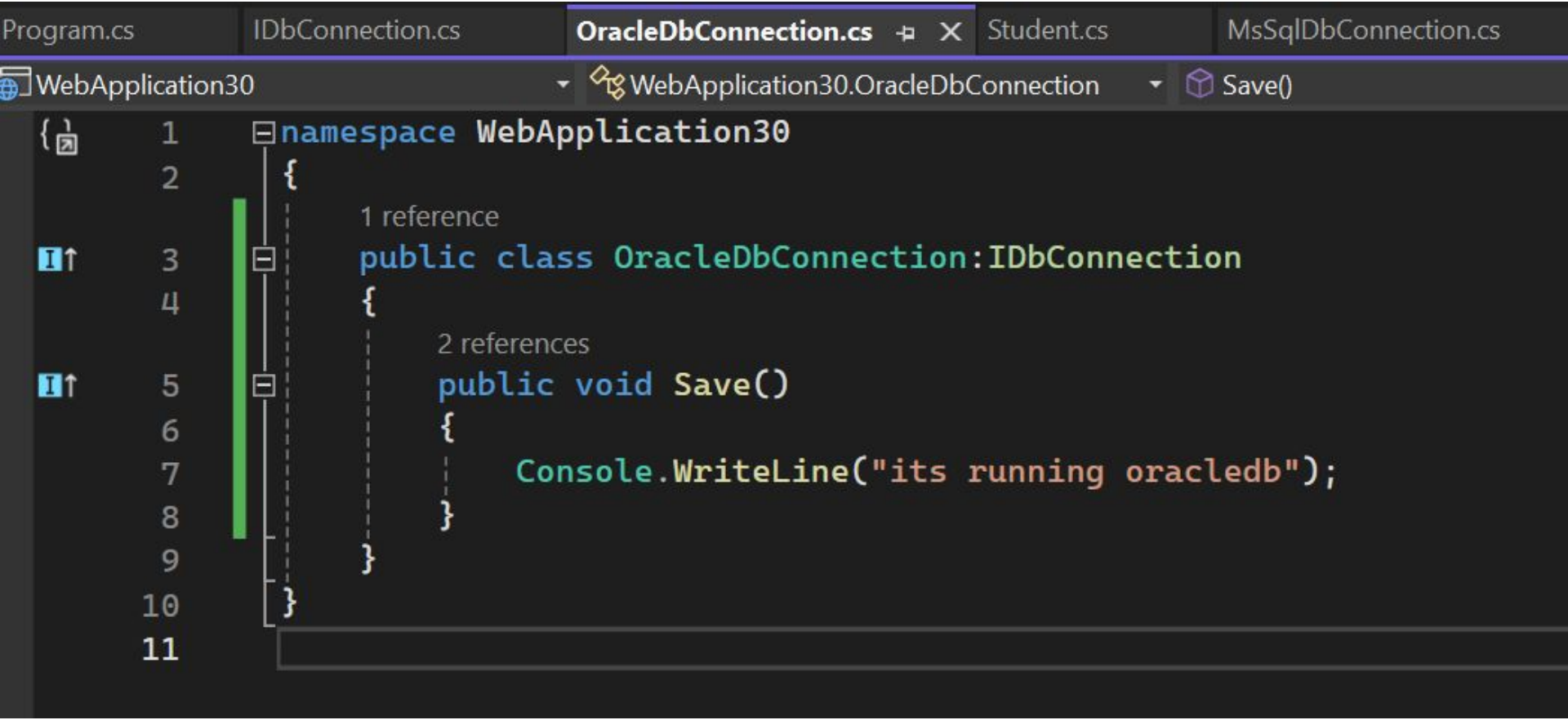



Program.cs
IDbConnection.cs
OracleDbConnection.cs
Student.cs
MsSqlDbConnection.cs
WebApplication30
app.UseAuthorization();
app.MapControllerRoute(
name: "default",
pattern: "{controller=Home}/{action=Index}/{id?}");
app.Run();
*/
using WebApplication30;
IDbConnection connection=new OracleDbConnection();
Student s = new Student(connection);
s.SaveStudent();
Console.ReadLine();


Program.cs
IDbConnection.cs
OracleDbConnection.cs
Student.cs
WebApplication30
WebApplication30.IDbConnection
namespace WebApplication30
{
5 references
public interface IDbConnection
{
3 references
void Save();
}
}


Program.cs
IDbConnection.cs
OracleDbConnection.cs
Student.cs
MsSqlDbConnection.cs
WebApplication30
WebApplication30.Student
SaveStudent()
namespace WebApplication30
{
3 references
public class Student
{
IDbConnection connection;
1 reference
public Student(IDbConnection dbConnection)
{
connection = dbConnection;
}
// MsSqlDbConnection connection = new MsSqlDbConnection();
1 reference
public void SaveStudent()
{
connection.Save();//calling mssql method
}
}
}


Program.cs
IDbConnection.cs
OracleDbConnection.cs
Student.cs
MsSqlDbConnection.cs
WebApplication30
WebApplication30.OracleDbConnection
Save()
namespace WebApplication30
{
1 reference
public class OracleDbConnection:IDbConnection
{
2 references
public void Save()
{
Console.WriteLine("its running oracledb");
}
}
}


Program.cs
IDbConnection.cs
OracleDbConnection.cs
Student.cs
MsSqlDbConnection.cs
WebApplication30
WebApplication30.OracleDbConnection
Save()
namespace WebApplication30
{
1 reference
public class OracleDbConnection:IDbConnection
{
2 references
public void Save()
{
Console.WriteLine("its running oracledb");
}
}
}

INTRODUCTION TO ENTITY FRAMEWORK

INTRODUCTION TO ORM

- ORM stands for object-relational mapping.
- Object-relational mapping (ORM) is a way to align programming code with database structure
- ORM uses metadata descriptors to create a layer between the programming language and a relational database.
- It thus connects OOPs code with the database,
- and simplifies the interaction between relational databases and OOP languages.

How ORM works?

HOW ORM WORKS?

- Suppose one is using to link python application with oracle database.
- In that case, the developer will have to choose the most relevant ORM as per the business requirements.
- This can be well understood by the diagram below:

Intreoduction to EF Core

INTRODUCTION TO EF CORE

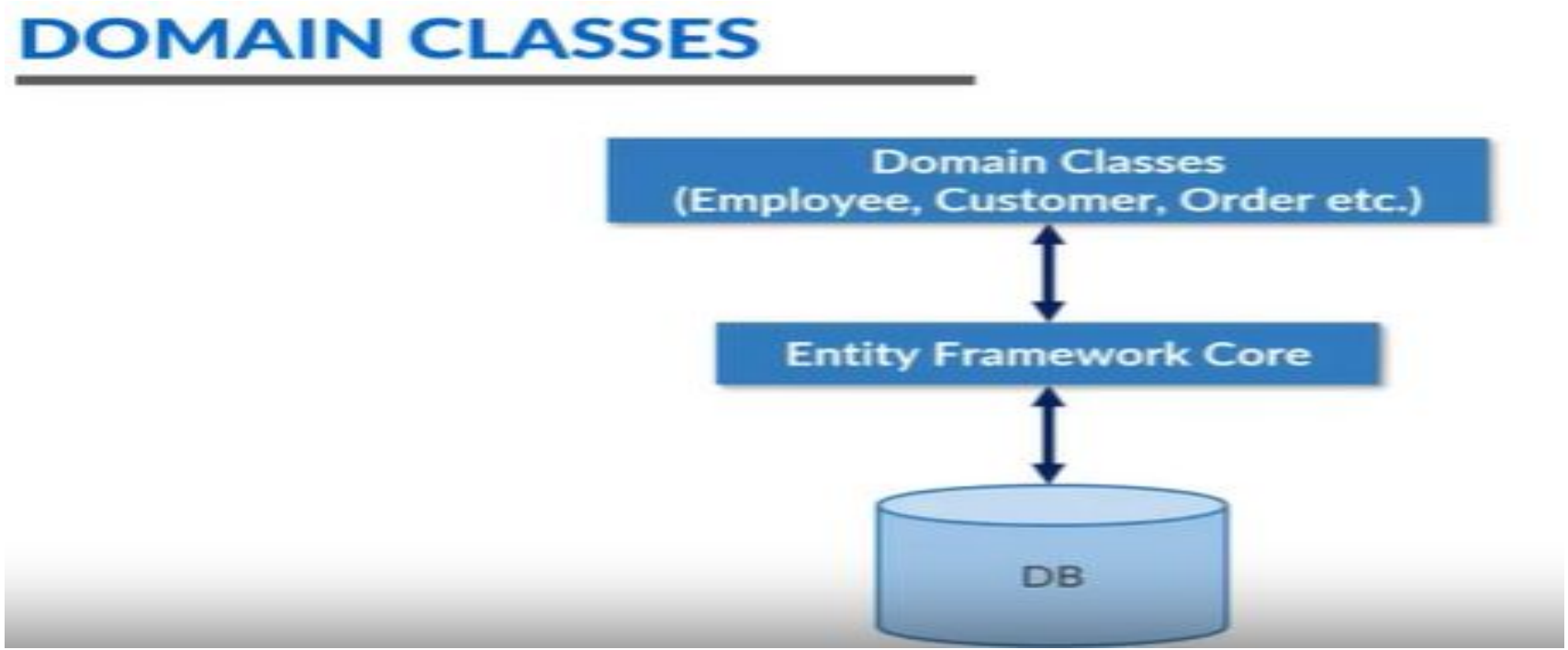
- Entity Framework is an Object Relational Mapping (ORM) framework.
- Entity Framework Core is a lightweight and extensible version of Entity Framework.
- It enables us to work with databases using .NET objects.
- EF Core provides a set of APIs for querying and manipulating data in relational databases.
- It supports a variety of database providers, including Microsoft SQL Server, SQLite, MySQL, PostgreSQL, Oracle, and others.

Domain Classes

DOMAIN CLASSES

- In our application, we usually write classes like Employee, Customer, Member, and Courses for CRUD operations.
- These classes are called Domain Classes.
- Without ORM, we will need to write a lot of code to implement the CRUD operations or map data after received from the Database.
- In the EF Core, Domain classes can be used in Code First Approach and DB First Approach.
- DB First Approach has limited access to EF Core.

Diagram


DOMAIN CLASSES
Domain Classes
(Employee, Customer, Order etc.)
Entity Framework Core
DB

Types of EF Core

TYPES OF EF CORE

1. Code First Approach:

Code First Approach works based on Domain Driven Design. In Code First approach,

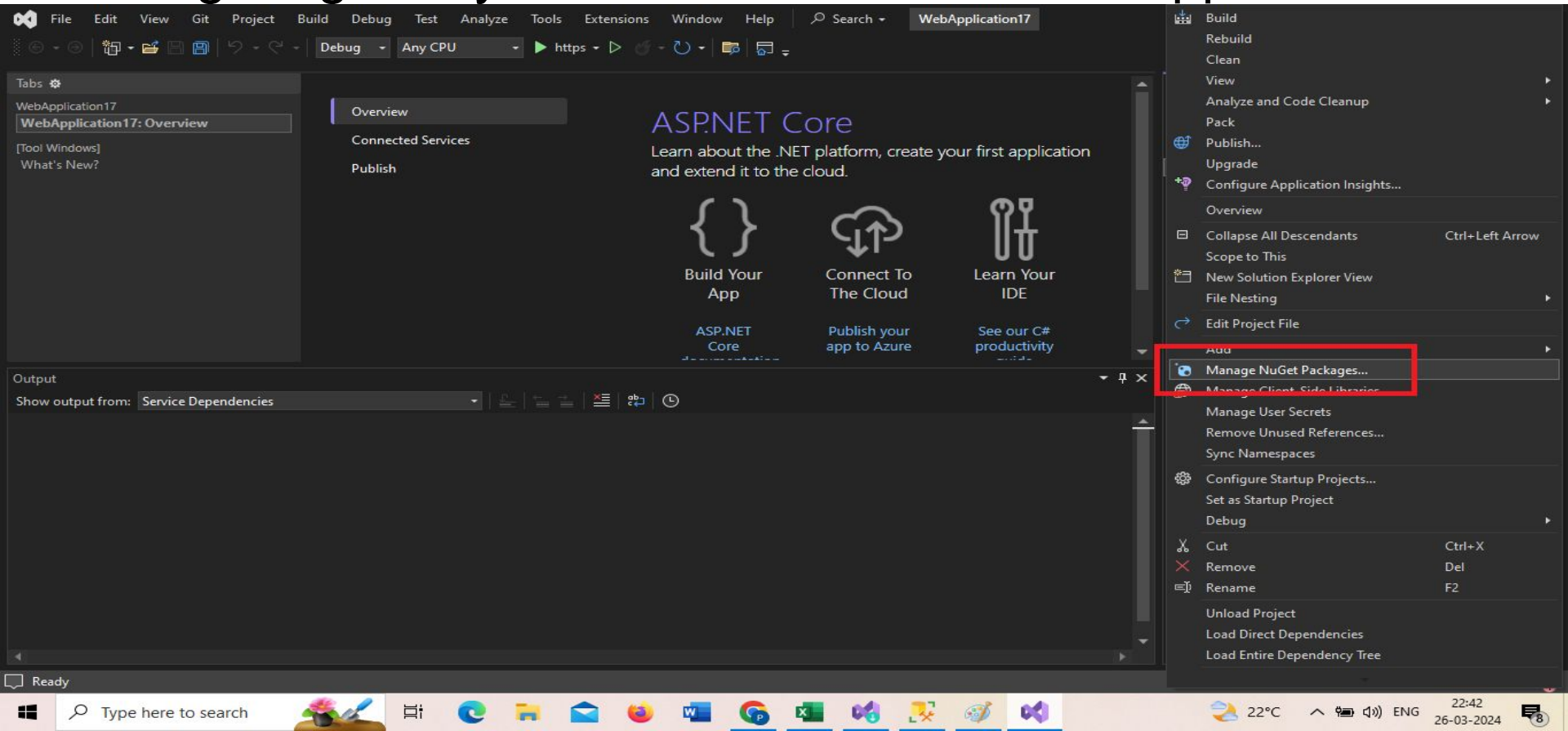
- Create Domain Classes.
- Create DB Context class derived from EF Core Db context classes.
- EF Core creates Db and Tables using a default configuration.
- You can change the EF Core default configuration.

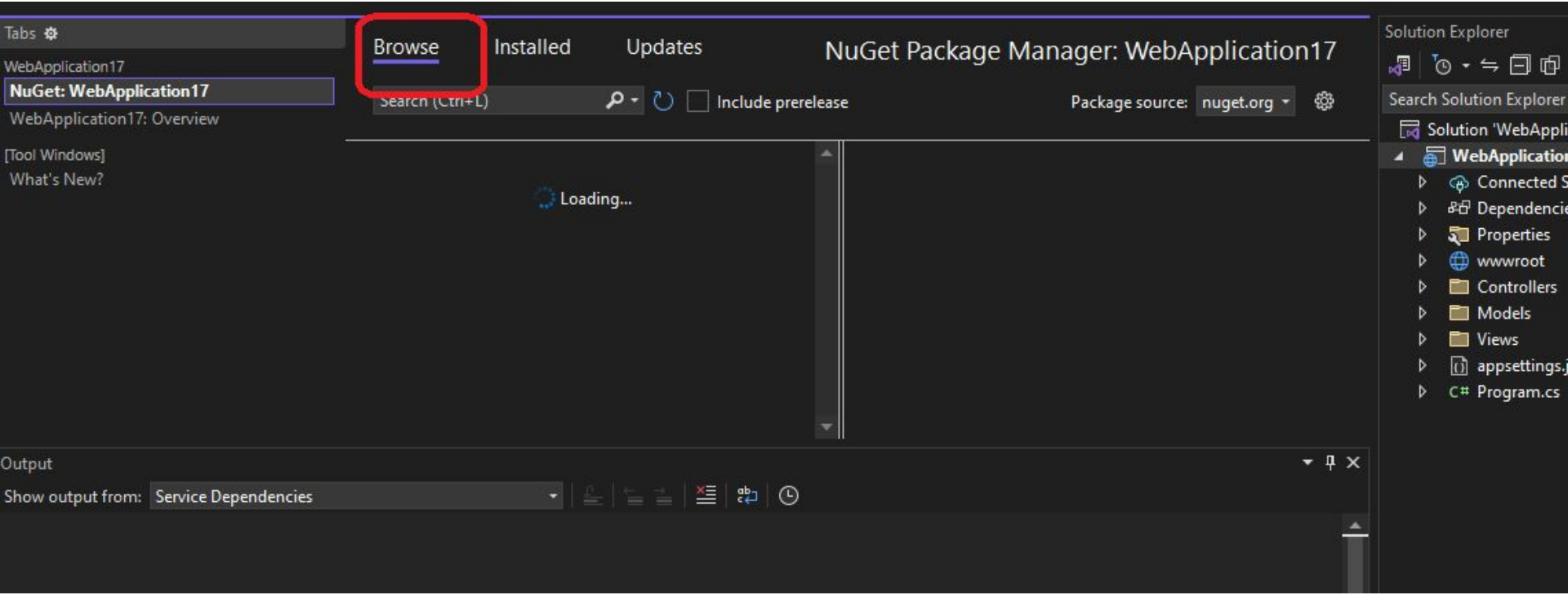
TYPES OF EF CORE

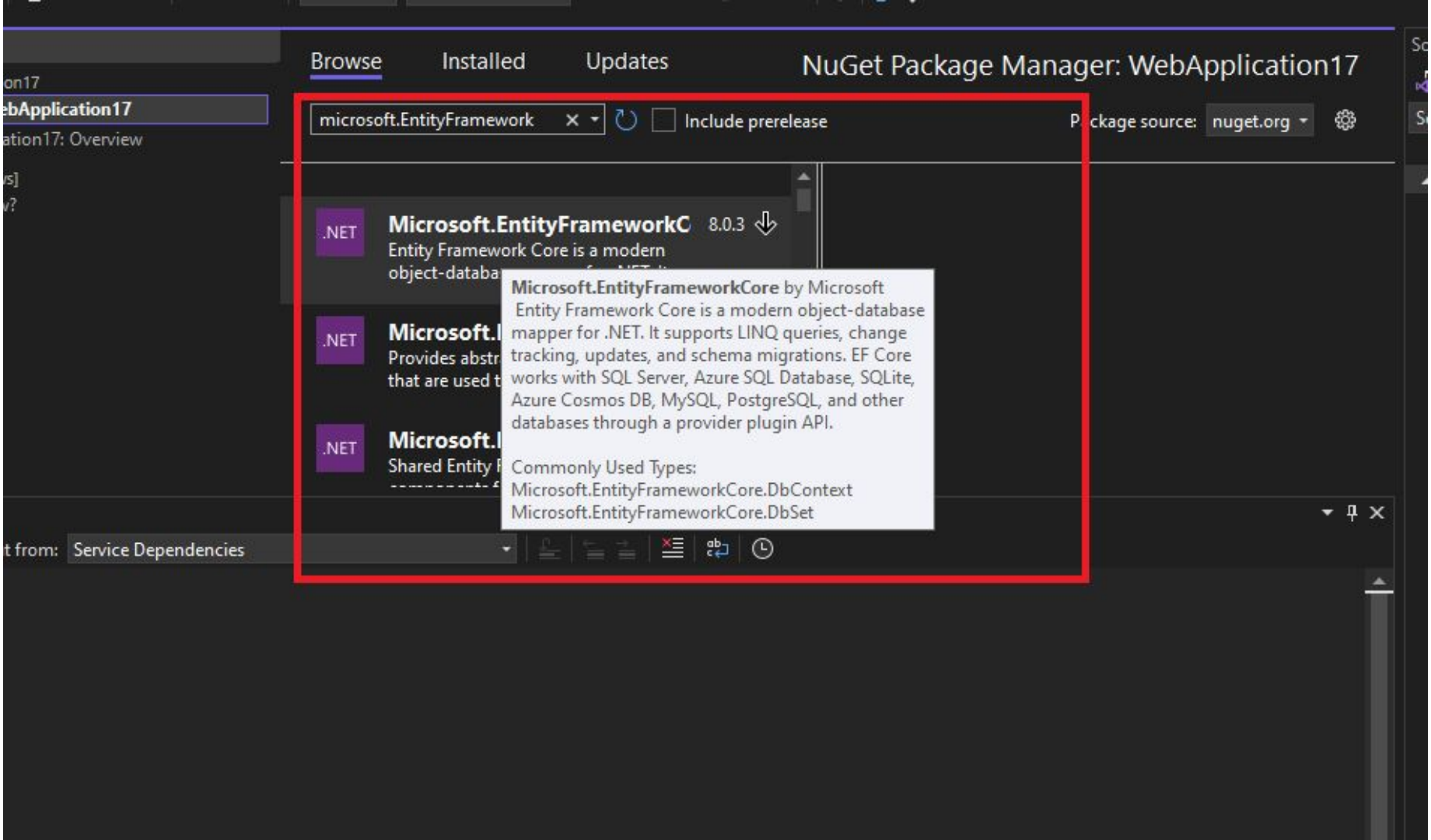
2. Database First Approach:

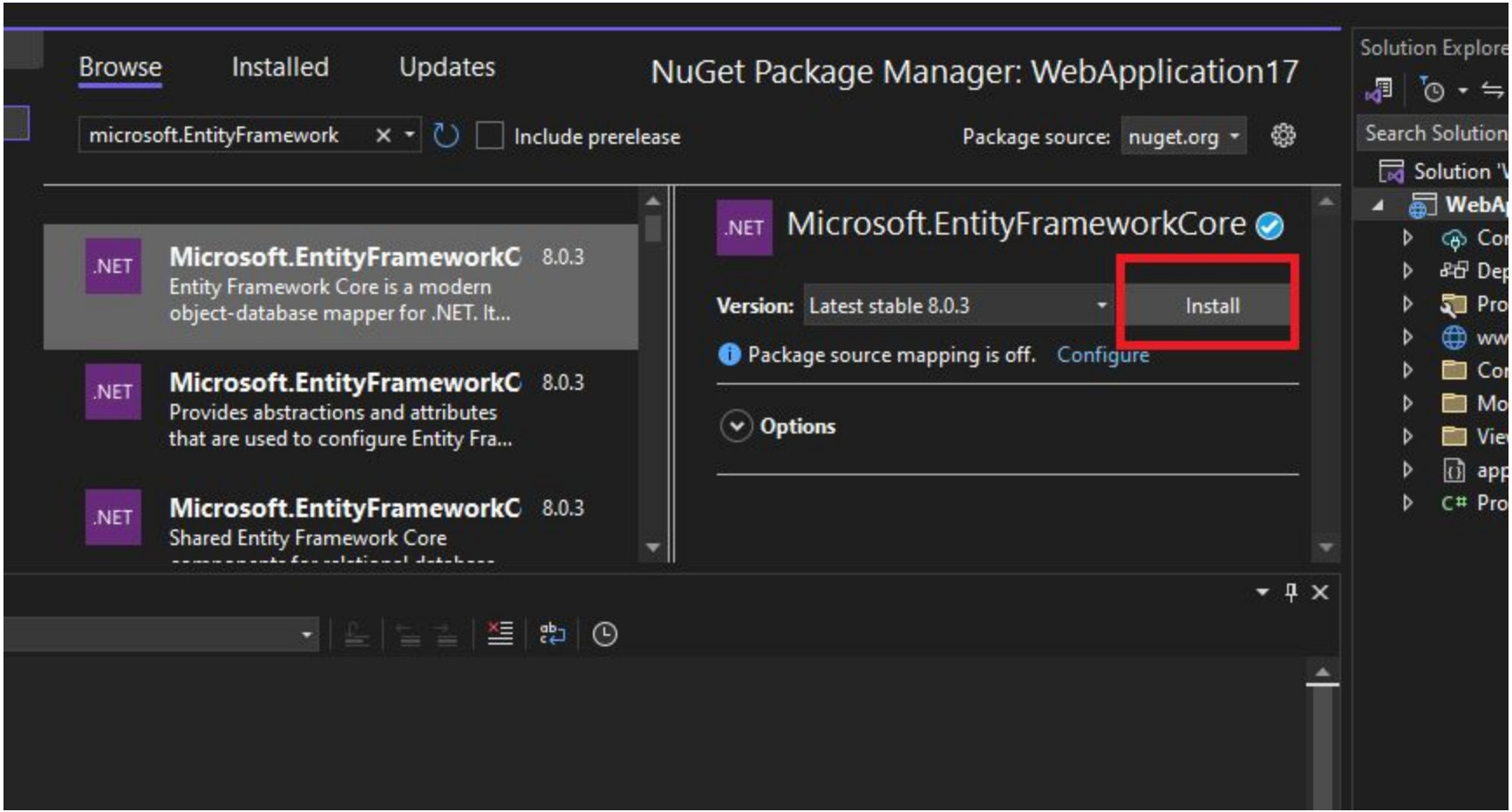
- The database approach will help us to create Domain classes and DB Context classes from an existing database using EF Core.
- If you have an existing Database then we should use this approach to work with ORM.

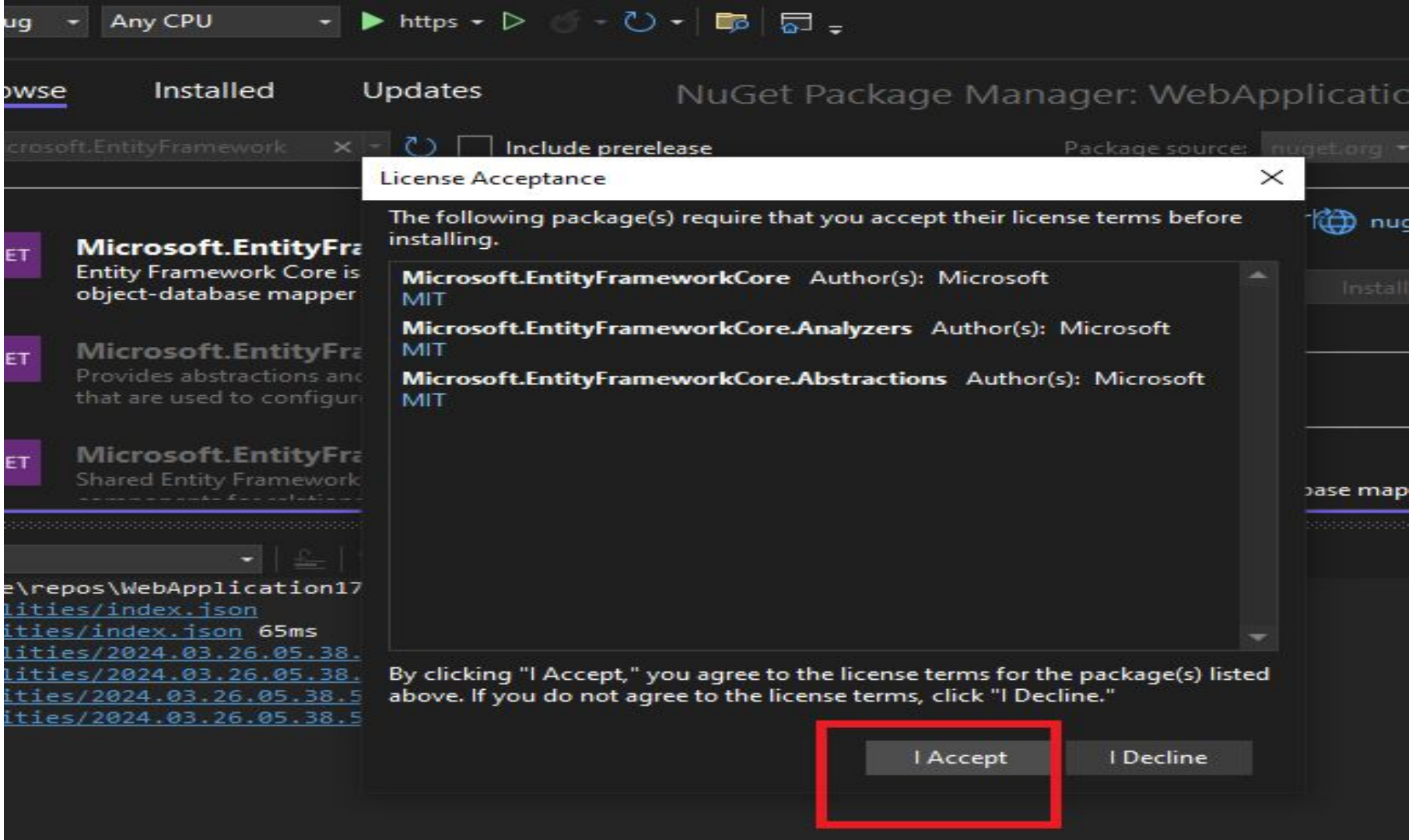
Configuring Entity Framework Core In MVC Application

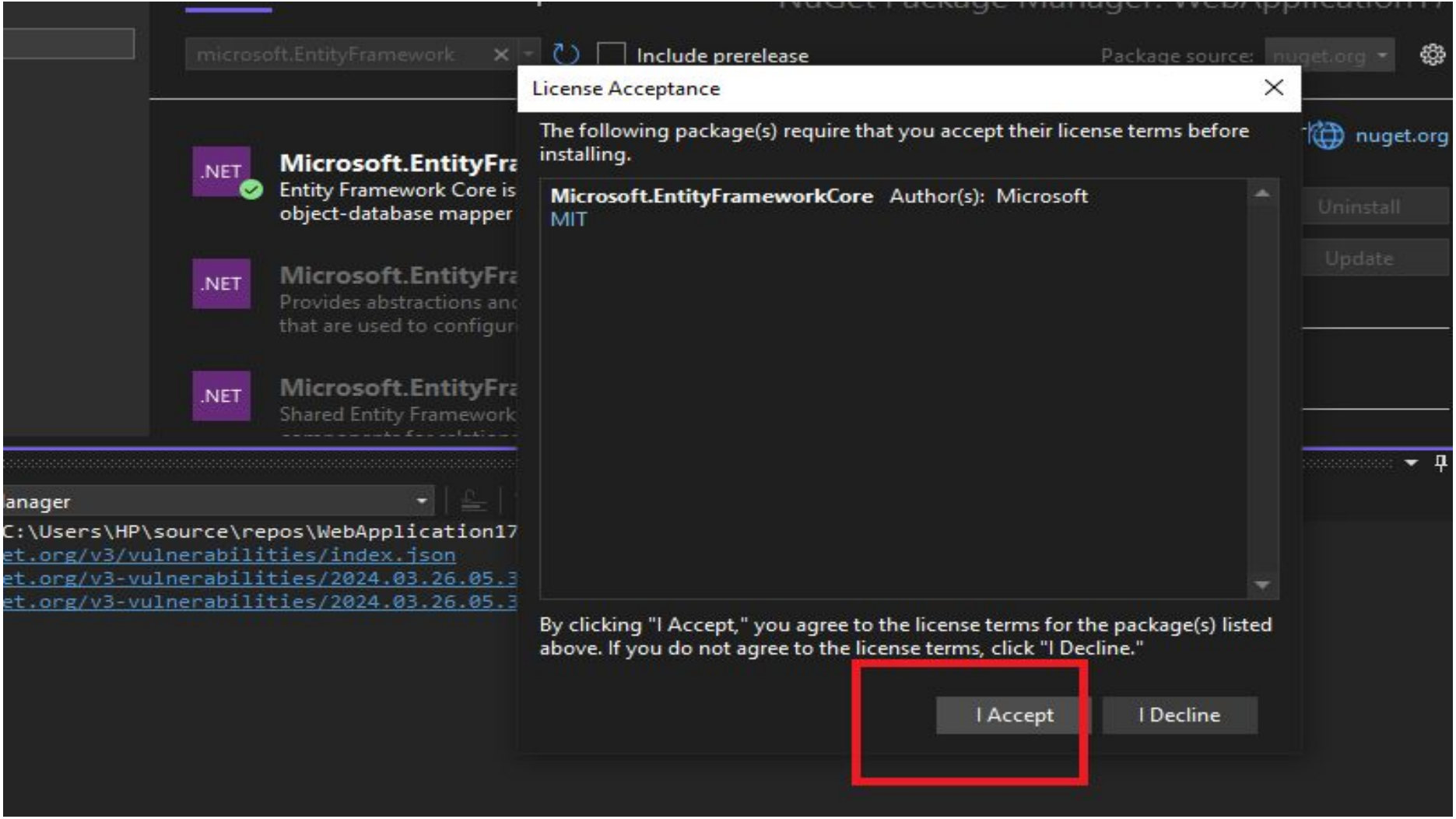


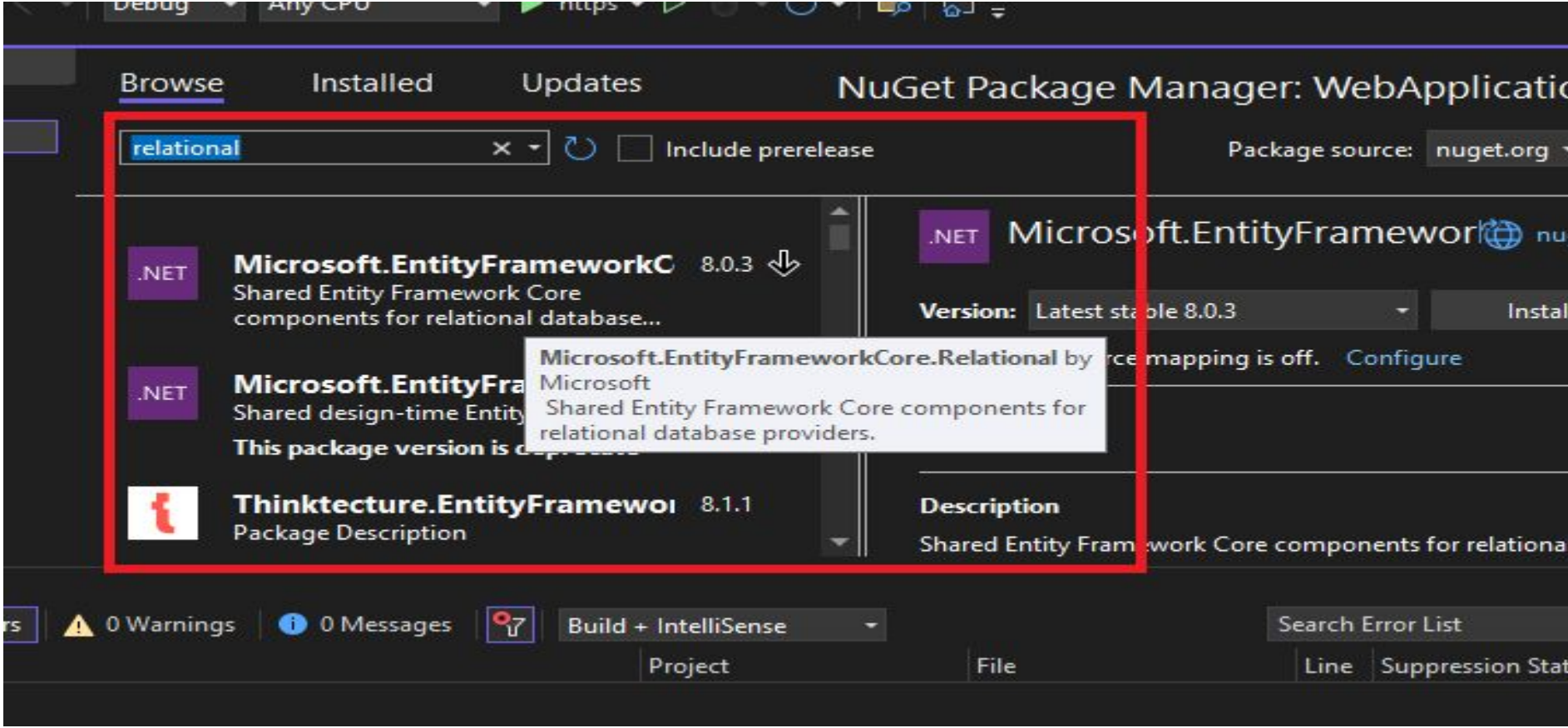

Tabs
WebApplication17
NuGet: WebApplication17
WebApplication17: Overview
[Tool Windows]
What's New?
Browse
Installed
Updates
NuGet Package Manager: WebApplication17
Search (Ctrl+L)
Include prerelease
Package source:
nuget.org
Loading...
Output
Show output from:
Service Dependencies
Solution Explorer
Search Solution Explorer
Solution 'WebAppli
WebApplicatio
Connected S
Dependencie
Properties
wwwroot
Controllers
Models
Views
appsettings.j
Program.cs

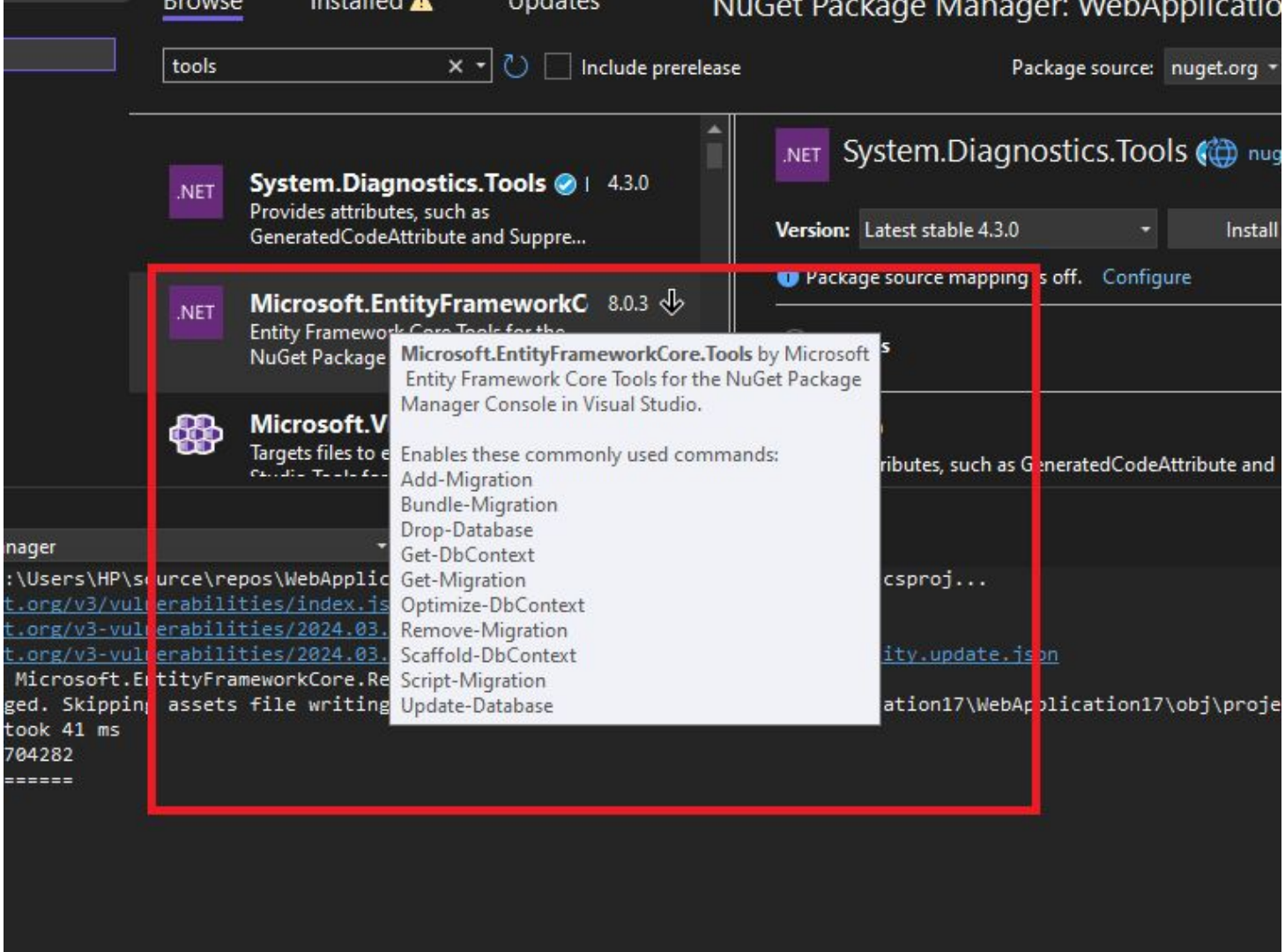

Browse
Installed
Updates
NuGet Package Manager: WebApplication17
microsoft.EntityFramework
Include prerelease
Package source:
nuget.org
.NET
Microsoft.EntityFrameworkC
8.0.3
Entity Framework Core is a modern
object-databa
Microsoft.EntityFrameworkCore by Microsoft
Entity Framework Core is a modern object-database mapper for .NET. It supports LINQ queries, change tracking, updates, and schema migrations. EF Core works with SQL Server, Azure SQL Database, SQLite, Azure Cosmos DB, MySQL, PostgreSQL, and other databases through a provider plugin API.
Commonly Used Types:
Microsoft.EntityFrameworkCore.DbContext
Microsoft.EntityFrameworkCore.DbSet
Service Dependencies

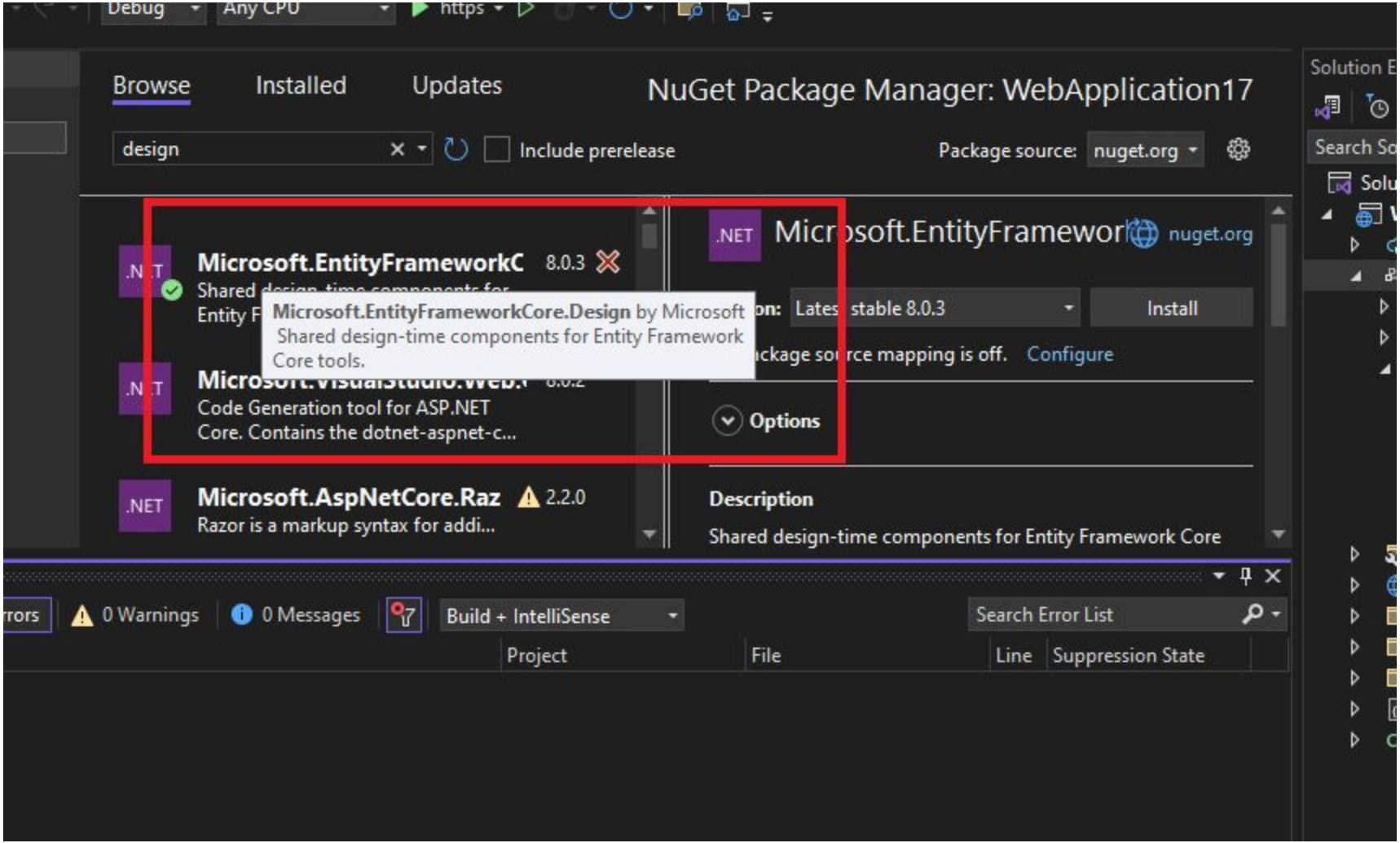

Browse
Installed
Updates
NuGet Package Manager: WebApplication17
microsoft.EntityFramework
Include prerelease
Package source:
nuget.org
Microsoft.EntityFrameworkC
8.0.3
Entity Framework Core is a modern object-database mapper for .NET. It...
Microsoft.EntityFrameworkC
8.0.3
Provides abstractions and attributes that are used to configure Entity Fra...
Microsoft.EntityFrameworkC
8.0.3
Shared Entity Framework Core
Microsoft.EntityFrameworkCore
Version:
Latest stable 8.0.3
Install
Package source mapping is off.
Configure
Options
Solution Explore
Search Solution

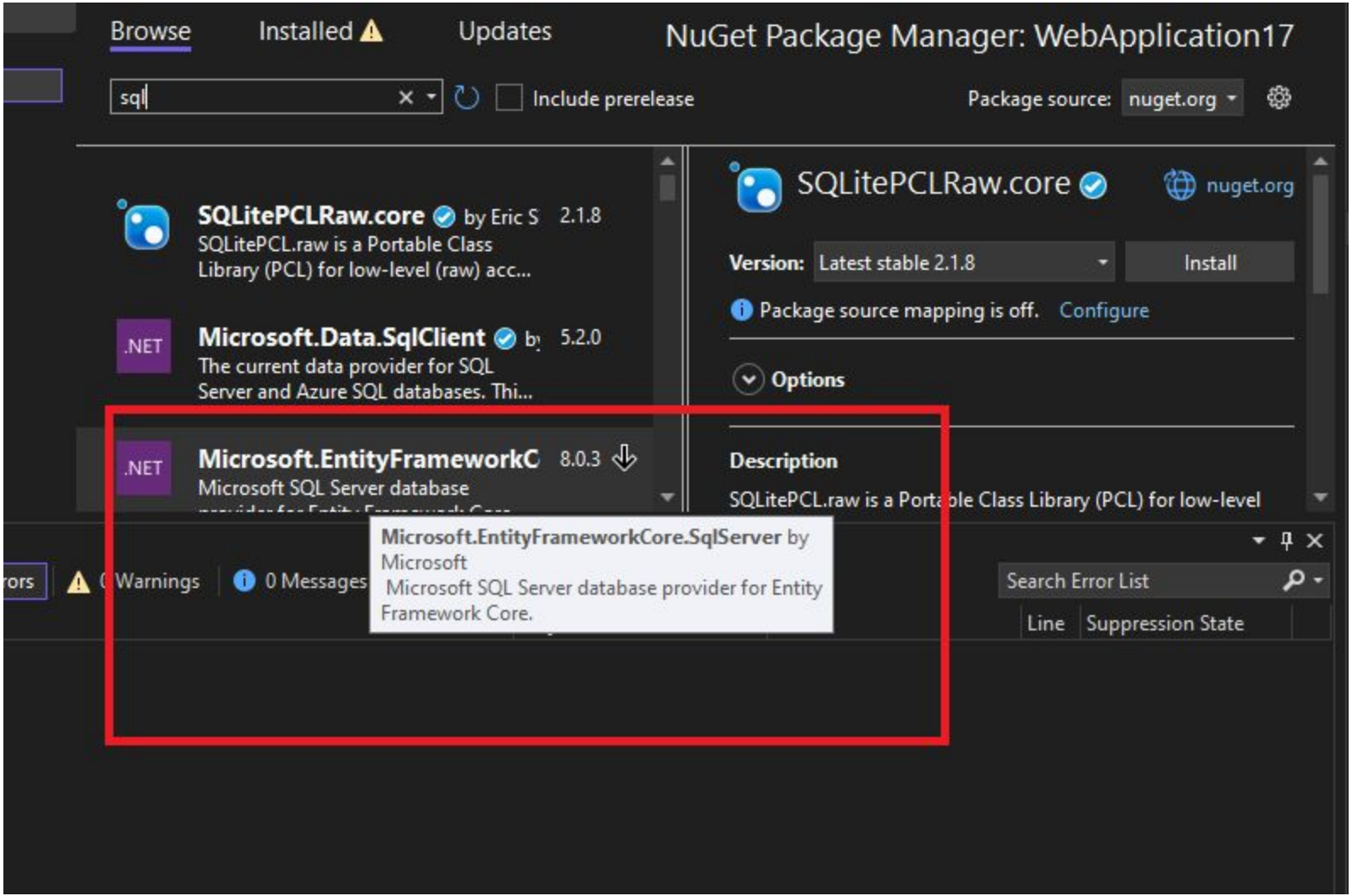

Any CPU
https
Installed
Updates
NuGet Package Manager: WebApplicatio
Include prerelease
Package source:
nuget.org
License Acceptance
The following package(s) require that you accept their license terms before installing.
Microsoft.EntityFrameworkCore Author(s): Microsoft
MIT
Microsoft.EntityFrameworkCore.Analyzers Author(s): Microsoft
MIT
Microsoft.EntityFrameworkCore.Abstractions Author(s): Microsoft
MIT
By clicking "I Accept," you agree to the license terms for the package(s) listed above. If you do not agree to the license terms, click "I Decline."
I Accept
I Decline
Microsoft.EntityFra
Entity Framework Core is
object-database mapper
Microsoft.EntityFra
Provides abstractions anc
that are used to configur
Microsoft.EntityFra
Shared Entity Framework
e\repos\WebApplication17
lities/index.json
ities/index.json 65ms
lities/2024.03.26.05.38.
lities/2024.03.26.05.38.
ities/2024.03.26.05.38.5
ities/2024.03.26.05.38.5


microsoft.EntityFramework
Include prerelease
Package source:
nuget.org
License Acceptance
The following package(s) require that you accept their license terms before installing.
Microsoft.EntityFrameworkCore Author(s): Microsoft
MIT
By clicking "I Accept," you agree to the license terms for the package(s) listed above. If you do not agree to the license terms, click "I Decline."
I Accept
I Decline
Microsoft.EntityFra
Entity Framework Core is
object-database mapper
Microsoft.EntityFra
Provides abstractions anc
that are used to configur
Microsoft.EntityFra
Shared Entity Framework
nuget.org
Uninstall
Update
anager
C:\Users\HP\source\repos\WebApplication17
et.org/v3/vulnerabilities/index.json
et.org/v3-vulnerabilities/2024.03.26.05.3
et.org/v3-vulnerabilities/2024.03.26.05.3


Debug
Any CPU
https
Browse
Installed
Updates
NuGet Package Manager: WebApplicatio
relational
Include prerelease
Package source:
nuget.org
.NET
Microsoft.EntityFrameworkC
8.0.3
Shared Entity Framework Core
components for relational database...
.NET
Microsoft.EntityFra
Shared design-time Entity
This package version is
Thinktecture.EntityFramewo
8.1.1
Package Description
Microsoft.EntityFrameworkCore.Relational by
Microsoft
Shared Entity Framework Core components for
relational database providers.
.NET
Microsoft.EntityFramewor
nu
Version:
Latest stable 8.0.3
Instal
mapping is off.
Configure
Description
Shared Entity Framework Core components for relationa
0 Warnings
0 Messages
Build + IntelliSense
Search Error List
Project
File
Line
Suppression Stat


Browse
Installed
Updates
NuGet Package Manager: WebApplicatio
tools
Include prerelease
Package source: nuget.org
.NET
System.Diagnostics.Tools | 4.3.0
Provides attributes, such as GeneratedCodeAttribute and Suppre...
.NET
Microsoft.EntityFrameworkC 8.0.3
Entity Framework Core Tools for the NuGet Package
Microsoft.V
Targets files to e
System.Diagnostics.Tools
Version: Latest stable 4.3.0
Install
Package source mapping s off. Configure
ributes, such as GeneratedCodeAttribute and
Microsoft.EntityFrameworkCore.Tools by Microsoft
Entity Framework Core Tools for the NuGet Package Manager Console in Visual Studio.
Enables these commonly used commands:
Add-Migration
Bundle-Migration
Drop-Database
Get-DbContext
Get-Migration
Optimize-DbContext
Remove-Migration
Scaffold-DbContext
Script-Migration
Update-Database
nager
:\Users\HP\s urce\repos\WebApplic
csproj...
t.org/v3/vul erabilities/index.js
t.org/v3-vul erabilities/2024.03.
t.org/v3-vul erabilities/2024.03.
ity.update.json
Microsoft.E tityFrameworkCore.Re
ged. Skippin assets file writing
ation17\WebA plication17\obj\proje
took 41 ms
704282
======


Debug
Any CPU
https
Browse
Installed
Updates
NuGet Package Manager: WebApplication17
design
Include prerelease
Package source:
nuget.org
Microsoft.EntityFrameworkC
8.0.3
Shared
Entity F
Microsoft.EntityFrameworkCore.Design by Microsoft
Shared design-time components for Entity Framework Core tools.
Code Generation tool for ASP.NET
Core. Contains the dotnet-aspnet-c...
Microsoft.AspNetCore.Raz
2.2.0
Razor is a markup syntax for addi...
.NET
Micr osoft.EntityFramewor
nuget.org
Lates stable 8.0.3
Install
ckage so rce mapping is off.
Configure
Options
Description
Shared design-time components for Entity Framework Core
0 Warnings
0 Messages
Build + IntelliSense
Search Error List
Project
File
Line
Suppression State
Solution E
Search So
Solu


Browse
Installed
Updates
NuGet Package Manager: WebApplication17
sql
Include prerelease
Package source: nuget.org
SQLitePCLRaw.core by Eric S 2.1.8
SQLitePCL.raw is a Portable Class Library (PCL) for low-level (raw) acc...
Microsoft.Data.SqlClient by 5.2.0
The current data provider for SQL Server and Azure SQL databases. Thi...
Microsoft.EntityFrameworkC 8.0.3
Microsoft SQL Server database
SQLitePCLRaw.core
nuget.org
Version: Latest stable 2.1.8
Install
Package source mapping is off. Configure
Options
Description
SQLitePCL.raw is a Portable Class Library (PCL) for low-level
Microsoft.EntityFrameworkCore.SqlServer by Microsoft
Microsoft SQL Server database provider for Entity Framework Core.
Warnings
0 Messages
Search Error List
Line
Suppression State

CODE FIRST APPROACH

dotnet ef migrations add InitialMigrate --project WebApplication22

dotnet ef database update --project WebApplication17

Install-Package Microsoft.EntityFrameworkCore.SqlServer

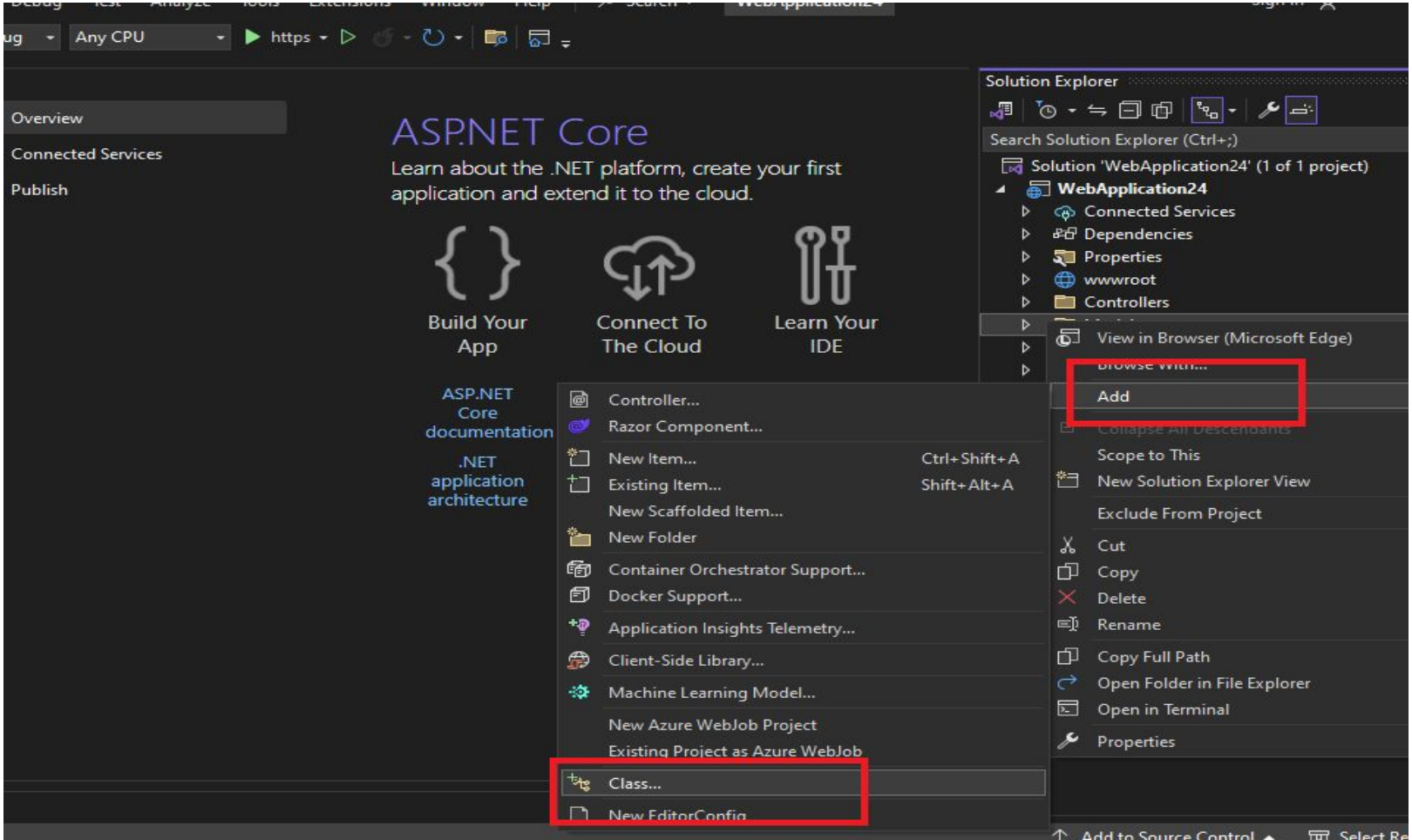
Code First Approach

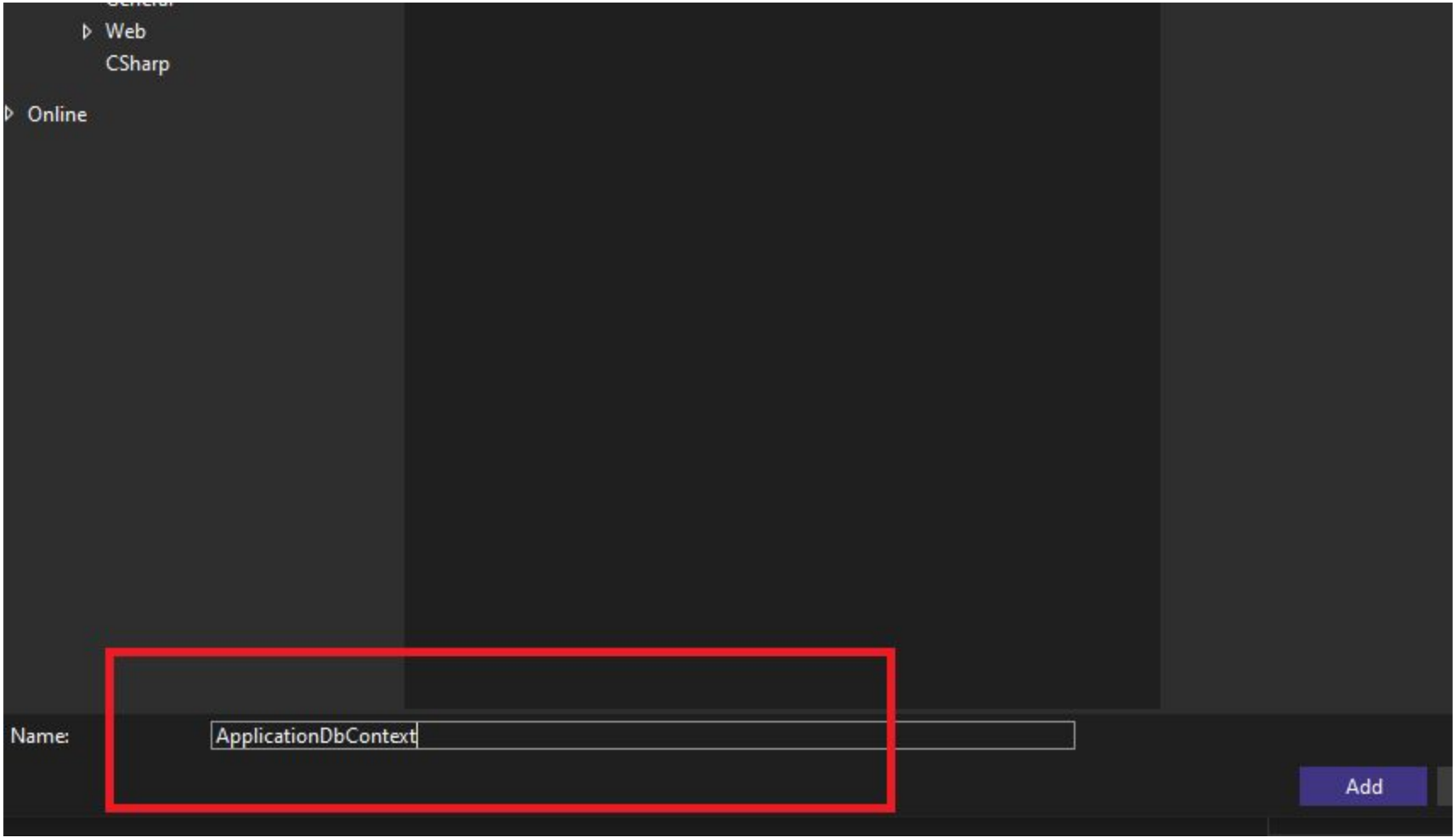
1. Code First Approach:

Code First Approach works based on Domain Driven Design. In Code First approach,

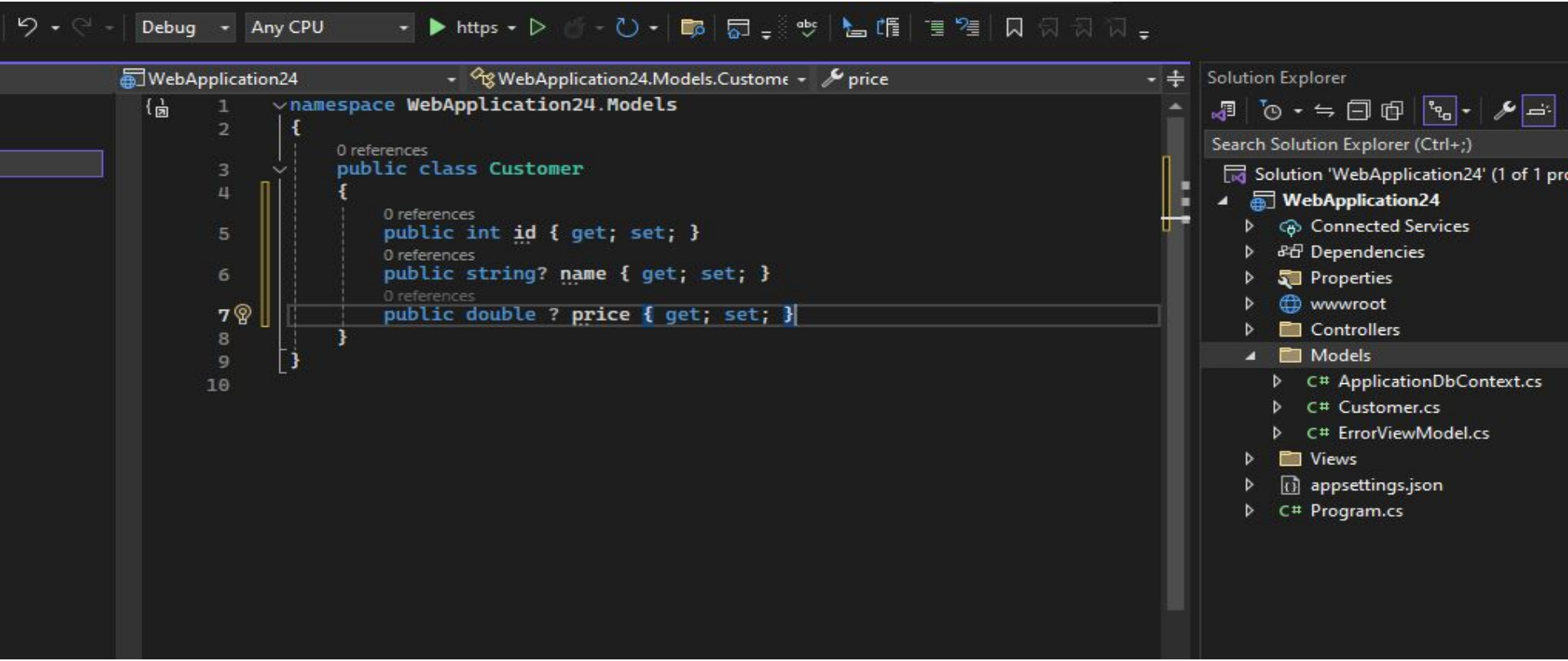
- Create Domain Classes.
- Create DB Context class derived from EF Core Db context classes.
- EF Core creates Db and Tables using a default configuration.
- You can change the EF Core default configuration.

Create a Model Class




Web
CSharp
Online
Name:
ApplicationDbContext
Add

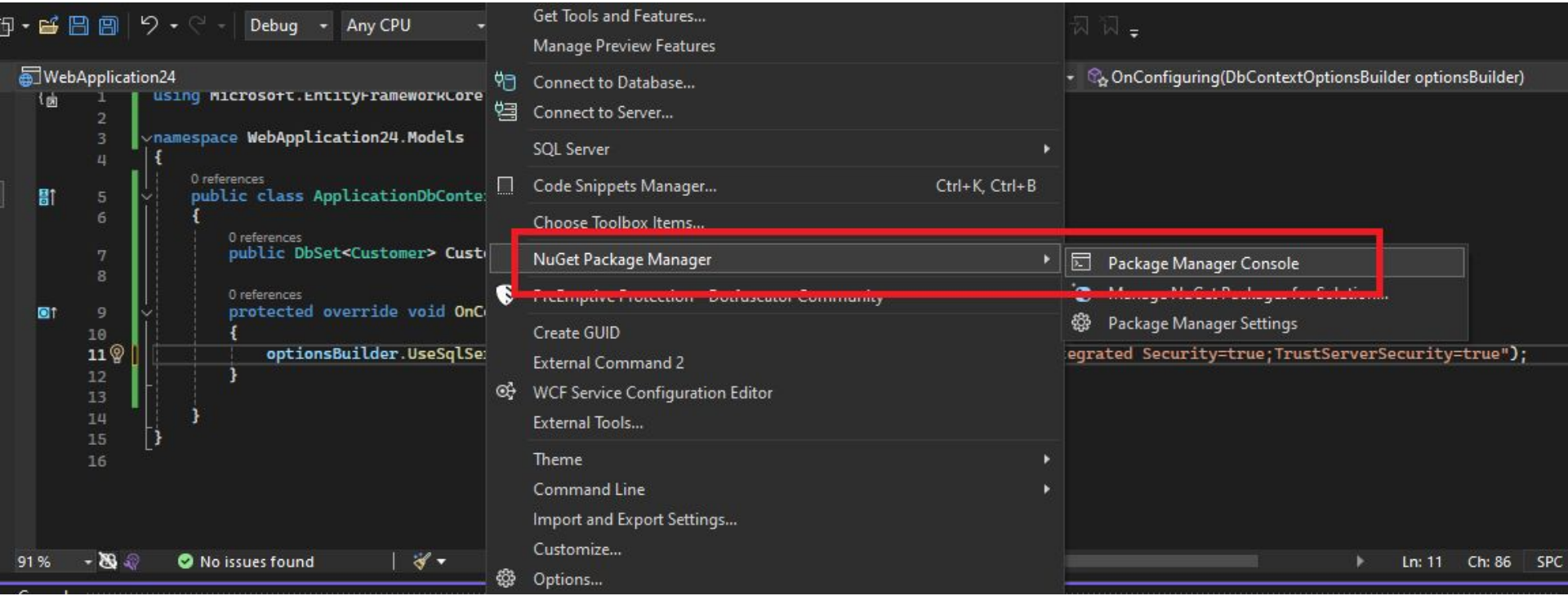
Add one more model class called Customer.cs



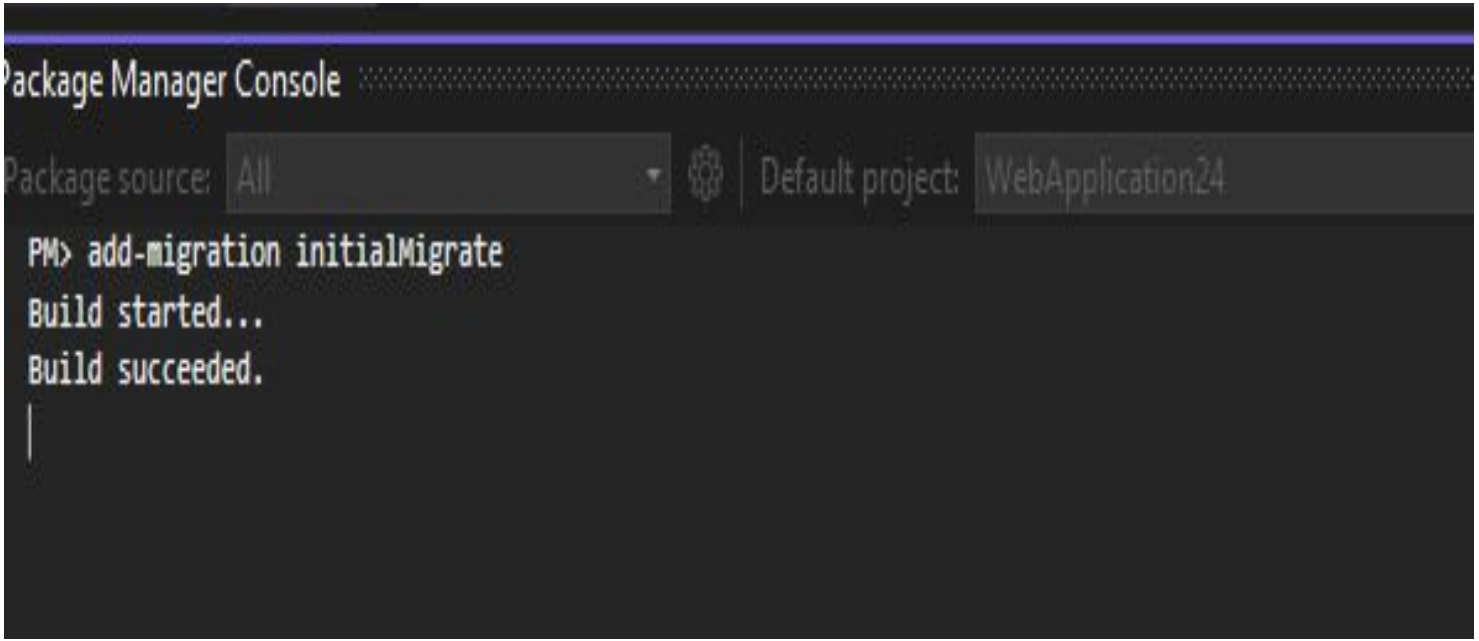
Now, in applicationdbcontext file, inherit the dbcontext file

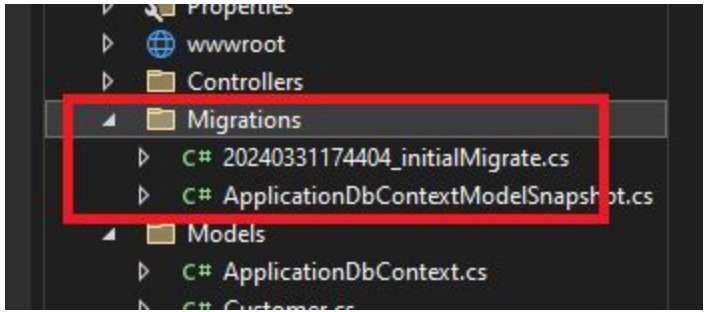
```
using Microsoft.EntityFrameworkCore;

namespace WebApplication24.Models
{
    2 references
    public class ApplicationDbContext:DbContext
    {
        0 references
        public DbSet<Customer> Customers { get; set; }

        0 references
        protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
        {
            optionsBuilder.UseSqlServer("Server=DESKTOP-6O0KGAO\\SQLEXPRESS;Database=palvidb1;Integrated Security=true;TrustServerCertificate=true");
        }

    }
}
```

Open tools then Nuget Package manager,then package manager console

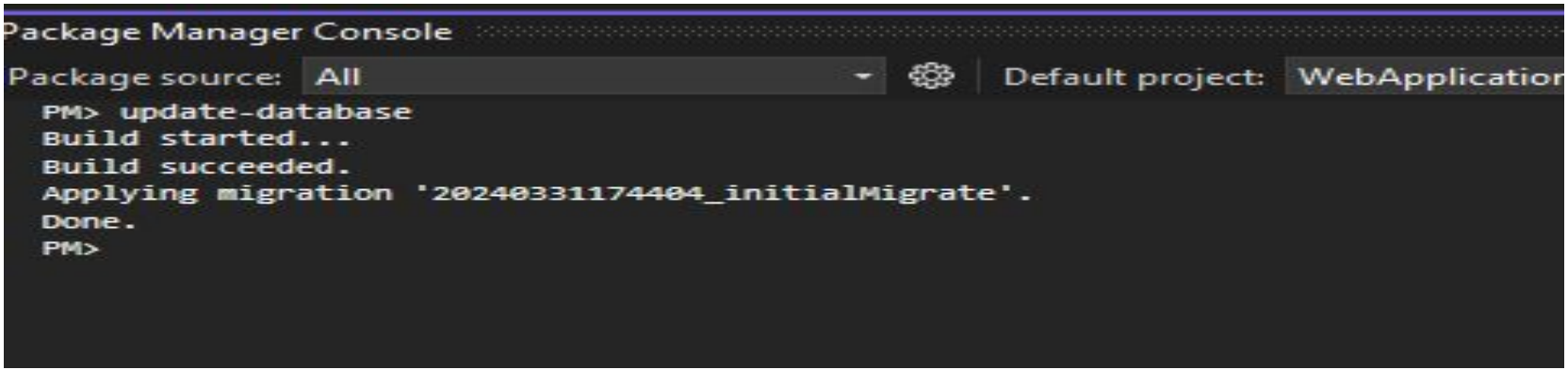


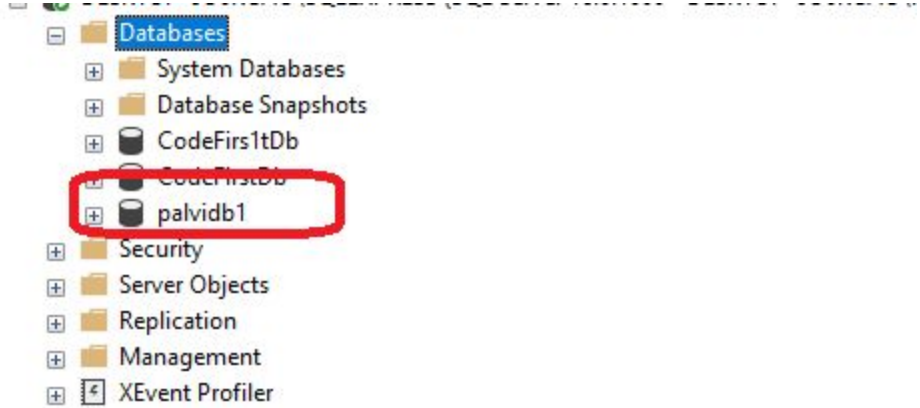
Now, we need to migrate the file, and convert it to database




wwwroot
Controllers
Migrations
20240331174404_initialMigrate.cs
ApplicationDbContextModelSnapsh
ot.cs
Models
ApplicationDbContext.cs

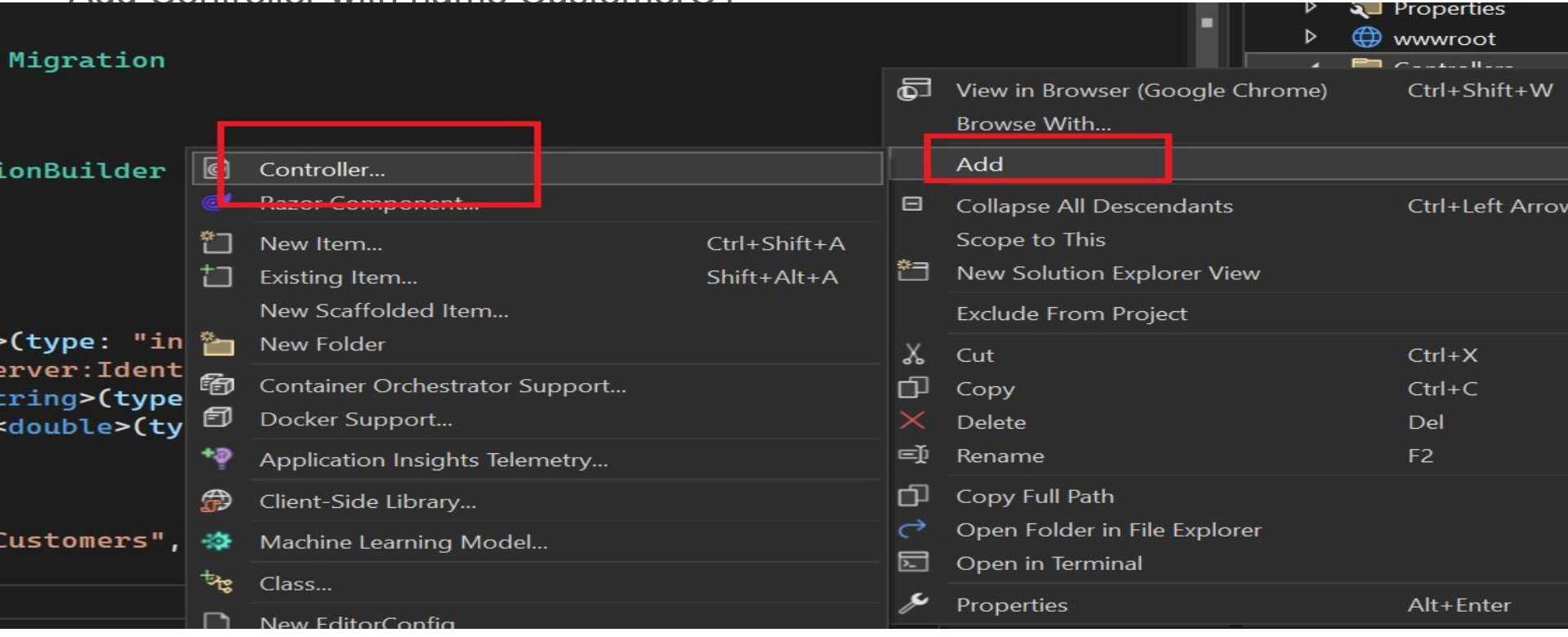
After migration, update database using update-database command

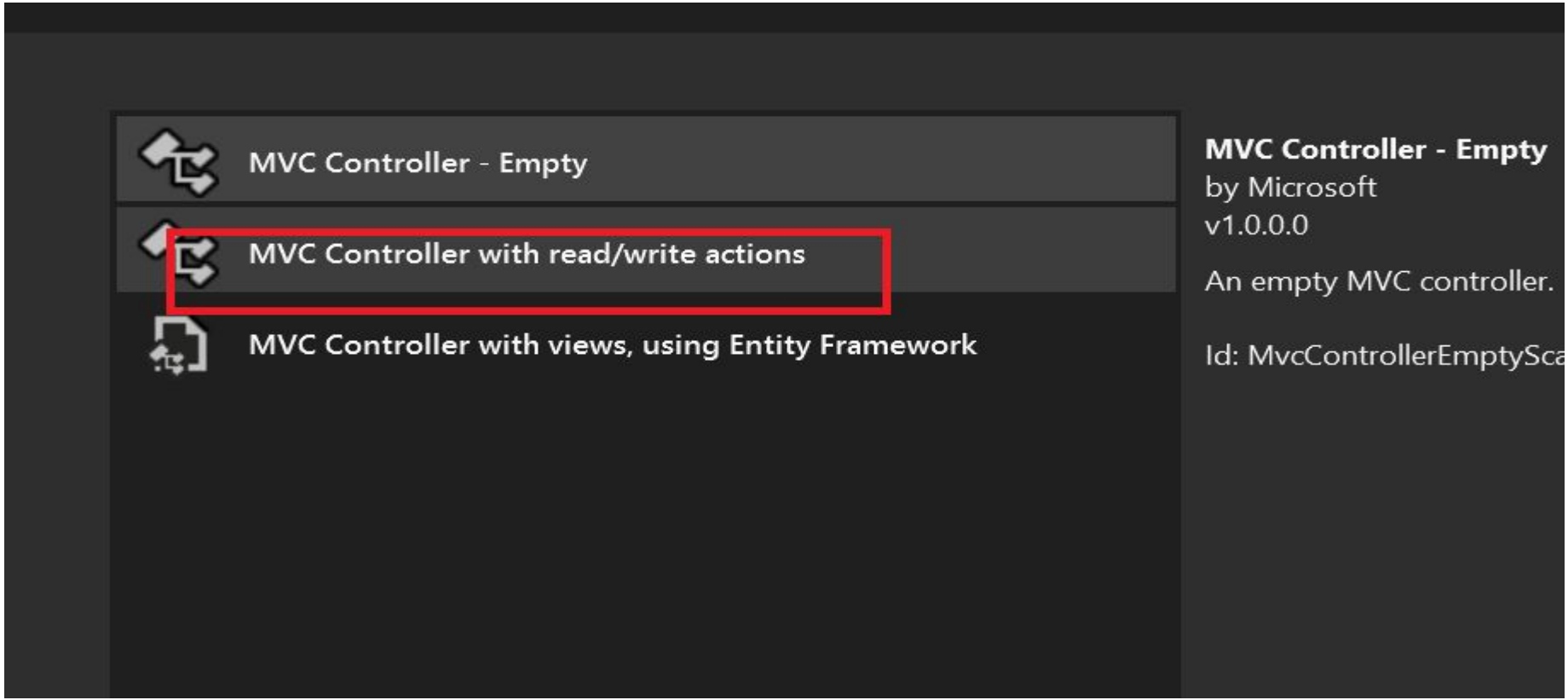


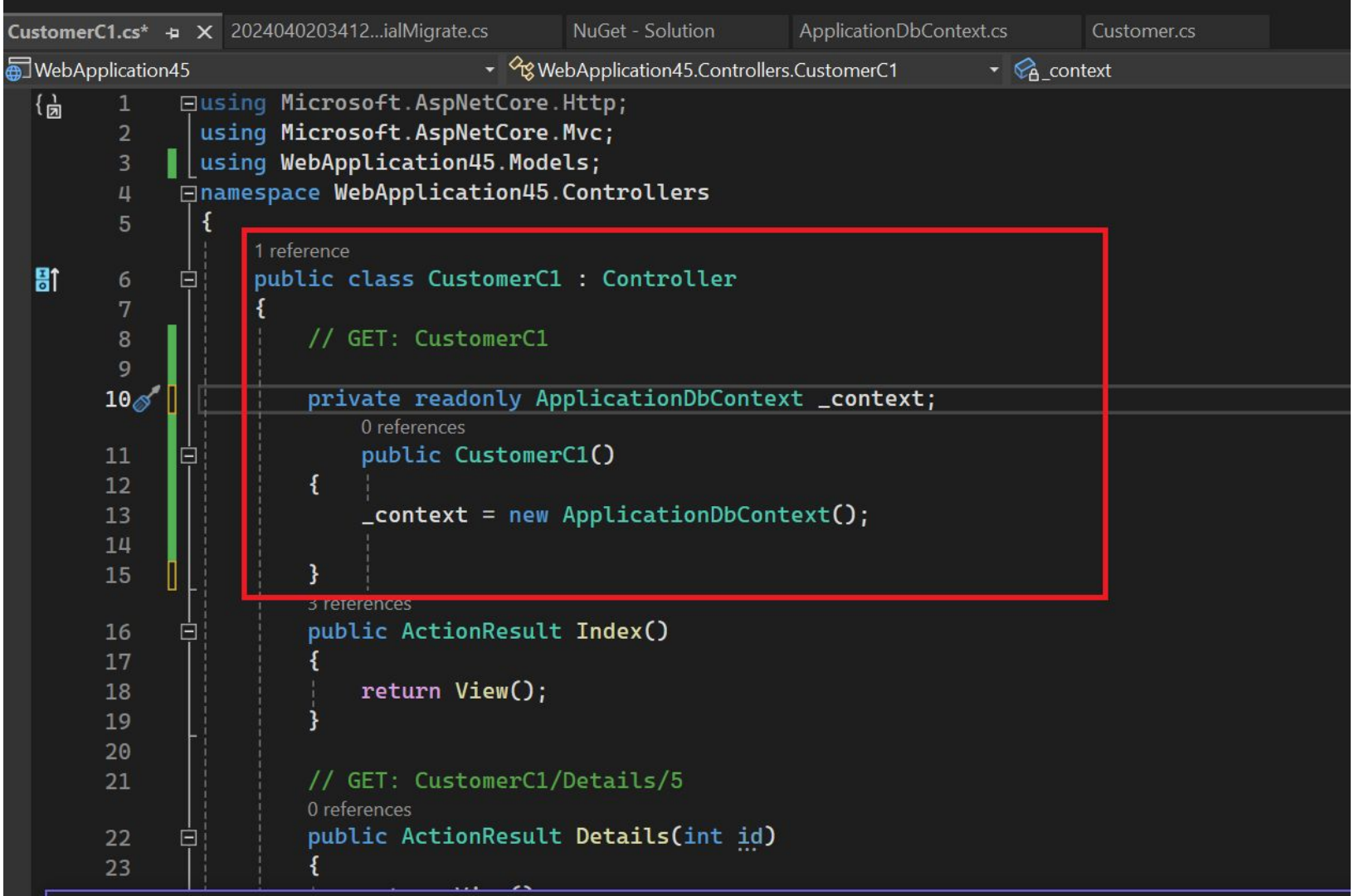

Databases
System Databases
Database Snapshots
CodeFirs1tDb
palvidb1
Security
Server Objects
Replication
Management
XEvent Profiler

CRUD OPERATIONS IN CODE FIRST APPROACH

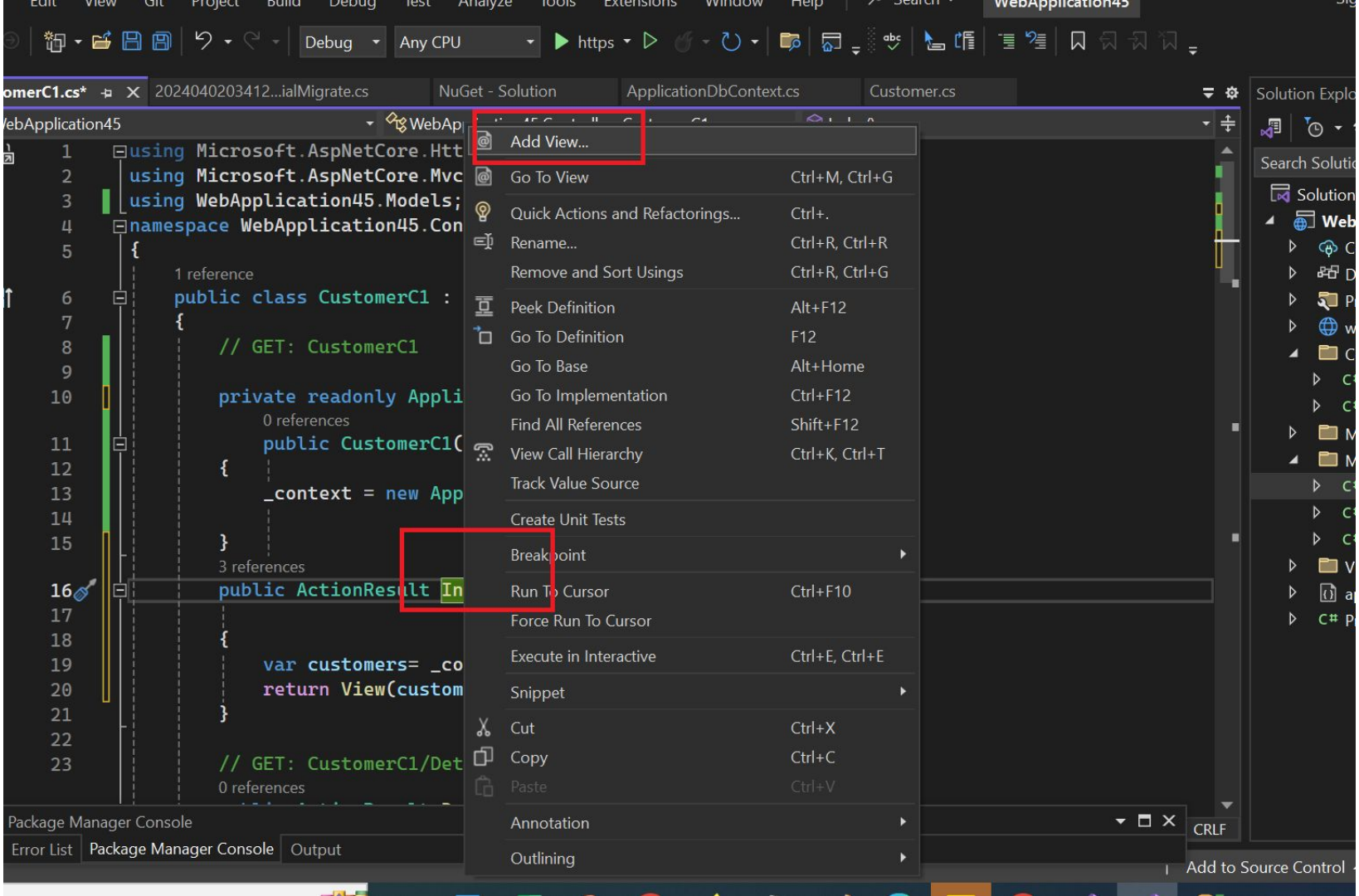
Add Controller with name CustomerC1

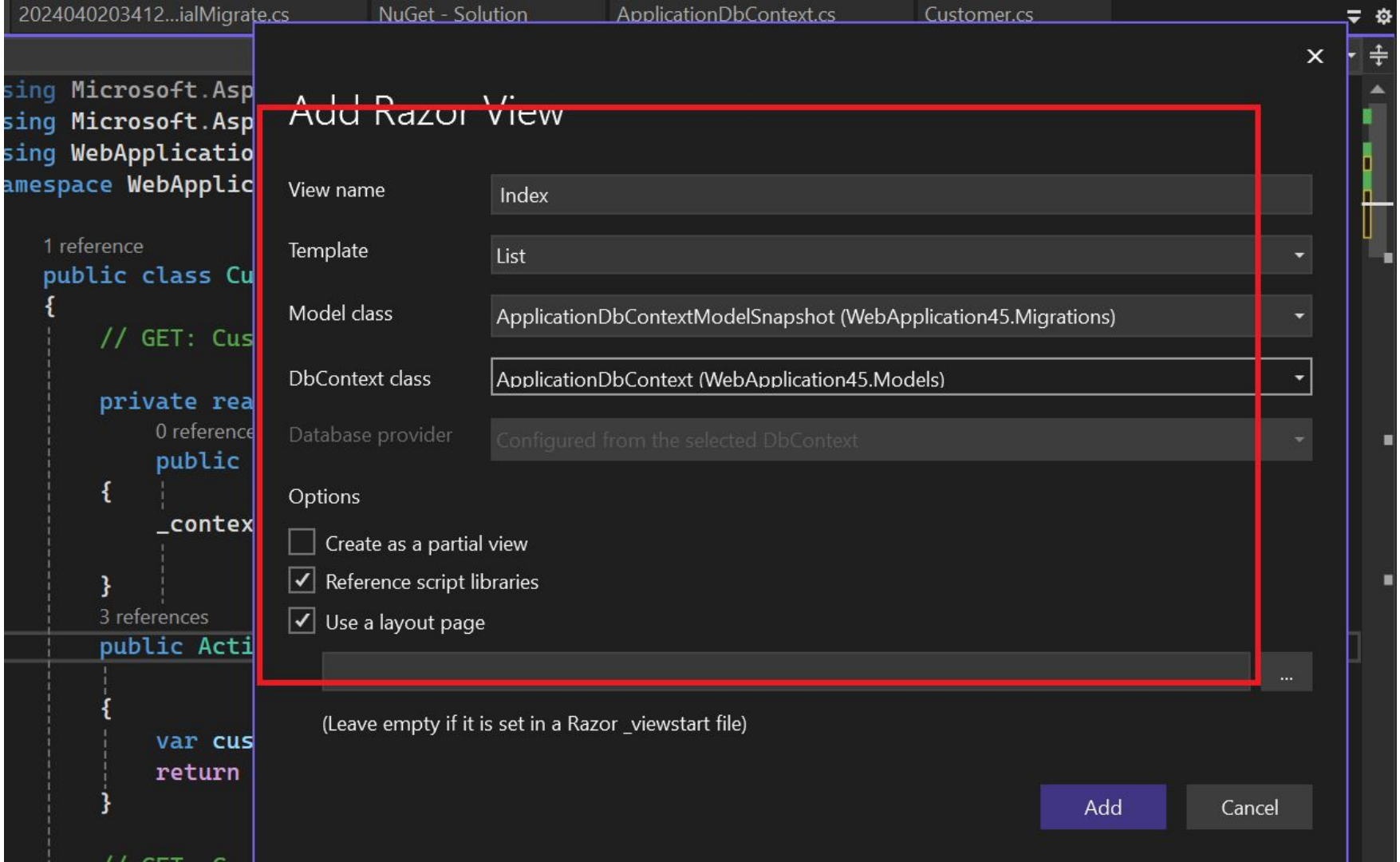



MVC Controller - Empty
MVC Controller with read/write actions
MVC Controller with views, using Entity Framework
MVC Controller - Empty
by Microsoft
v1.0.0.0
An empty MVC controller.
Id: MvcControllerEmptySca


CustomerC1.cs*
2024040203412...ialMigrate.cs
NuGet - Solution
ApplicationDbContext.cs
Customer.cs
WebApplication45
WebApplication45.Controllers.CustomerC1
_context
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using WebApplication45.Models;
namespace WebApplication45.Controllers
{
1 reference
public class CustomerC1 : Controller
{
// GET: CustomerC1
private readonly ApplicationDbContext _context;
0 references
public CustomerC1()
{
_context = new ApplicationDbContext();
}
3 references
public ActionResult Index()
{
return View();
}
// GET: CustomerC1/Details/5
0 references
public ActionResult Details(int id)
{

```
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using WebApplication45.Models;
namespace WebApplication45.Controllers
{
    1 reference
    public class CustomerC1 : Controller
    {
        // GET: CustomerC1

        private readonly ApplicationDbContext _context;
            0 references
            public CustomerC1()
            {
                _context = new ApplicationDbContext();

            }
            3 references
            public ActionResult Index()

            {
                var customers= _context.Customers.ToList();
                return View(customers);
            }

        // GET: CustomerC1/Details/5
        0 references
```

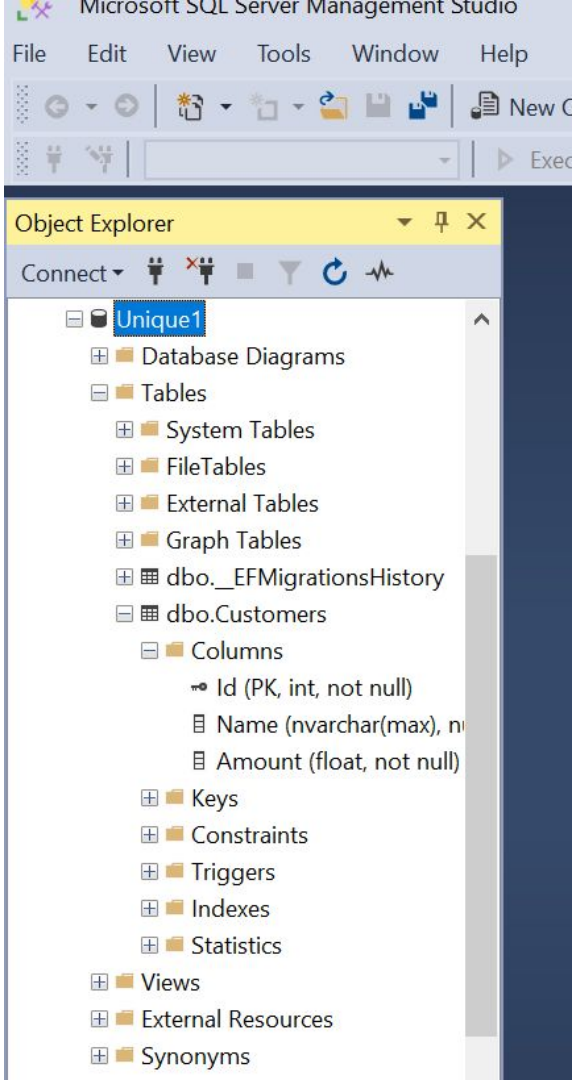

Debug
Any CPU
https
NuGet - Solution
ApplicationDbContext.cs
Customer.cs
WebApplication45
Add View...
Go To View Ctrl+M, Ctrl+G
Quick Actions and Refactorings... Ctrl+.
Rename... Ctrl+R, Ctrl+R
Remove and Sort Usings Ctrl+R, Ctrl+G
Peek Definition Alt+F12
Go To Definition F12
Go To Base Alt+Home
Go To Implementation Ctrl+F12
Find All References Shift+F12
View Call Hierarchy Ctrl+K, Ctrl+T
Track Value Source
Create Unit Tests
Breakpoint
Run To Cursor Ctrl+F10
Force Run To Cursor
Execute in Interactive Ctrl+E, Ctrl+E
Snippet
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Annotation
Outlining
Package Manager Console
Error List
Output
Solution Explorer
Search Solution
Add to Source Control

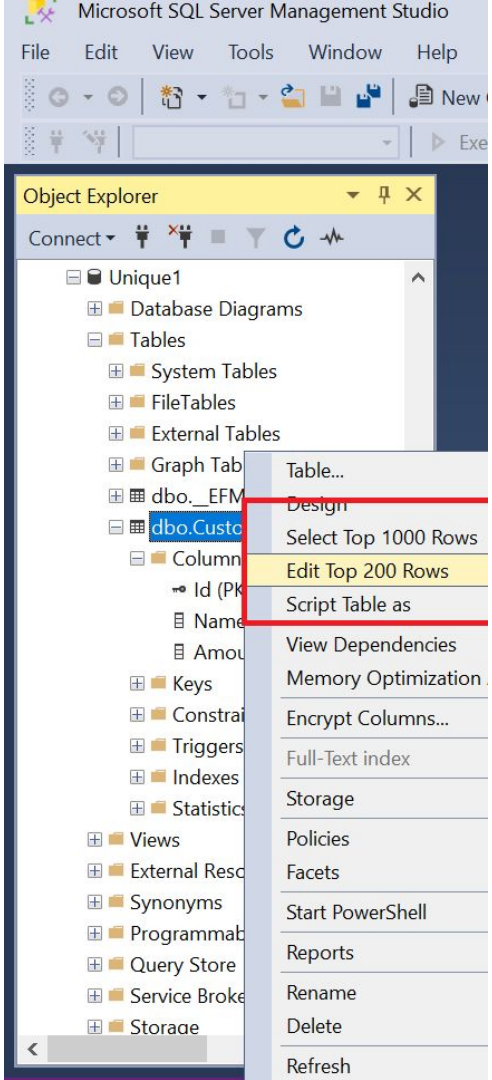

2024040203412...ialMigrate.cs
NuGet - Solution
ApplicationDbContext.cs
Customer.cs
sing Microsoft.Asp
sing Microsoft.Asp
sing WebApplicatio
amespace WebApplic
1 reference
public class Cu
{
// GET: Cus
private rea
0 reference
public
{
_contex
}
3 references
public Acti
{
var cus
return
}
Add Razor View
View name
Index
Template
List
Model class
ApplicationDbContextModelSnapshot (WebApplication45.Migrations)
DbContext class
ApplicationDbContext (WebApplication45.Models)
Database provider
Configured from the selected DbContext
Options
Create as a partial view
Reference script libraries
Use a layout page
...
(Leave empty if it is set in a Razor _viewstart file)
Add
Cancel

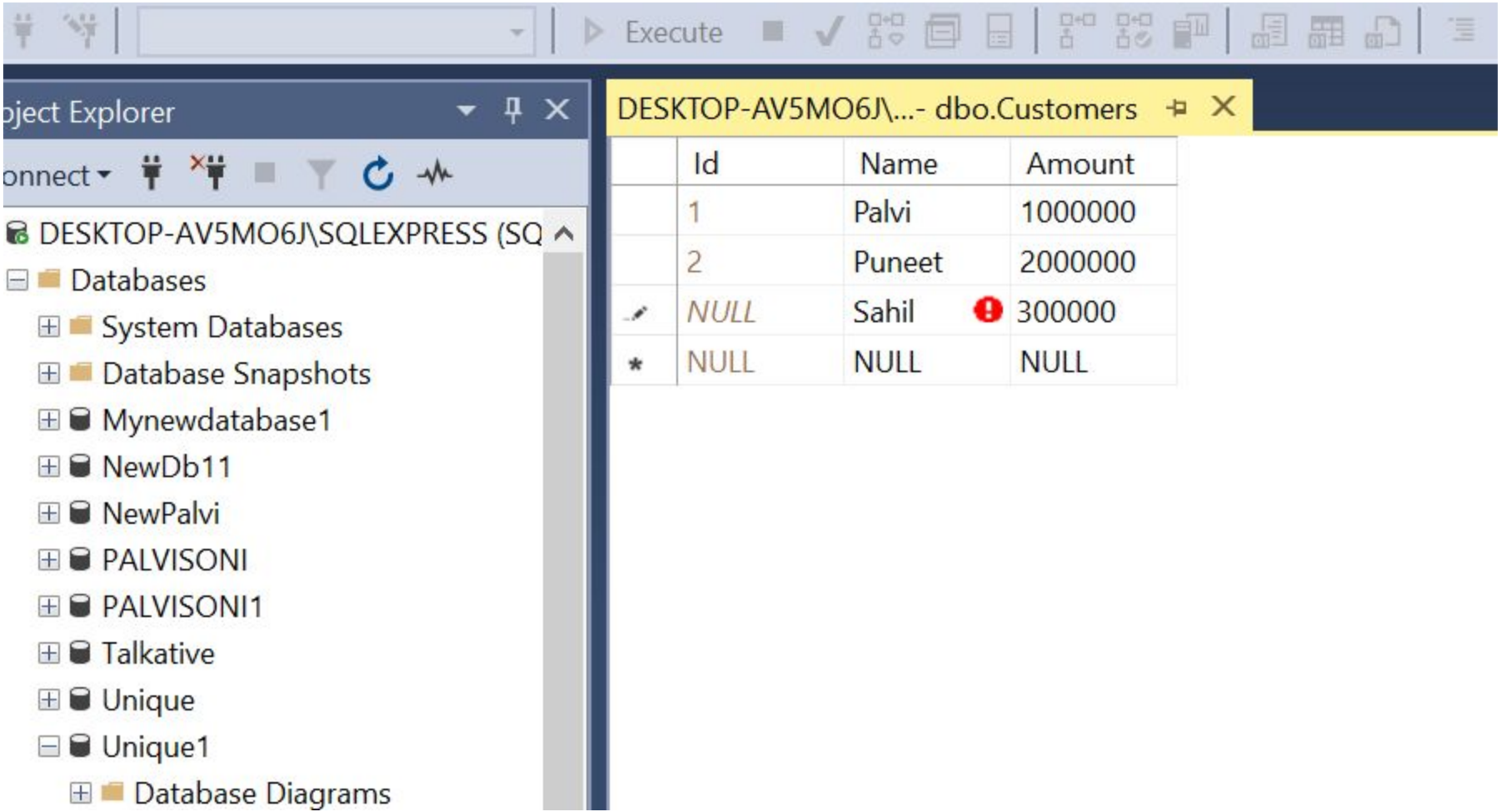
Index

Create New

Name	Amount


Microsoft SQL Server Management Studio
File
Edit
View
Tools
Window
Help
New
Exec
Object Explorer
Connect
Unique1
Database Diagrams
Tables
System Tables
FileTables
External Tables
Graph Tables
dbo.__EFMigrationsHistory
dbo.Customers
Columns
Id (PK, int, not null)
Name (nvarchar(max), n
Amount (float, not null)
Keys
Constraints
Triggers
Indexes
Statistics
Views
External Resources
Synonyms


Microsoft SQL Server Management Studio
File
Edit
View
Tools
Window
Help
New
Object Explorer
Connect
Unique1
Database Diagrams
Tables
System Tables
FileTables
External Tables
Graph Tab
dbo._EFM
dbo.Custo
Column
Id (PK
Name
Amou
Keys
Constrai
Triggers
Indexes
Statistics
Views
External Resc
Synonyms
Programmab
Query Store
Service Broke
Storage
Table...
Design
Select Top 1000 Rows
Edit Top 200 Rows
Script Table as
View Dependencies
Memory Optimization
Encrypt Columns...
Full-Text index
Storage
Policies
Facets
Start PowerShell
Reports
Rename
Delete
Refresh


Execute
bject Explorer
onnect
DESKTOP-AV5MO6J\SQLEXPRESS (SQ
Databases
System Databases
Database Snapshots
Mynewdatabase1
NewDb11
NewPalvi
PALVISONI
PALVISONI1
Talkative
Unique
Unique1
Database Diagrams
DESKTOP-AV5MO6J\...- dbo.Customers
Id
Name
Amount
1
Palvi
1000000
2
Puneet
2000000
NULL
Sahil
300000
NULL
NULL
NULL

localhost:7119/CustomerC1/index

WebApplication45 Home Privacy

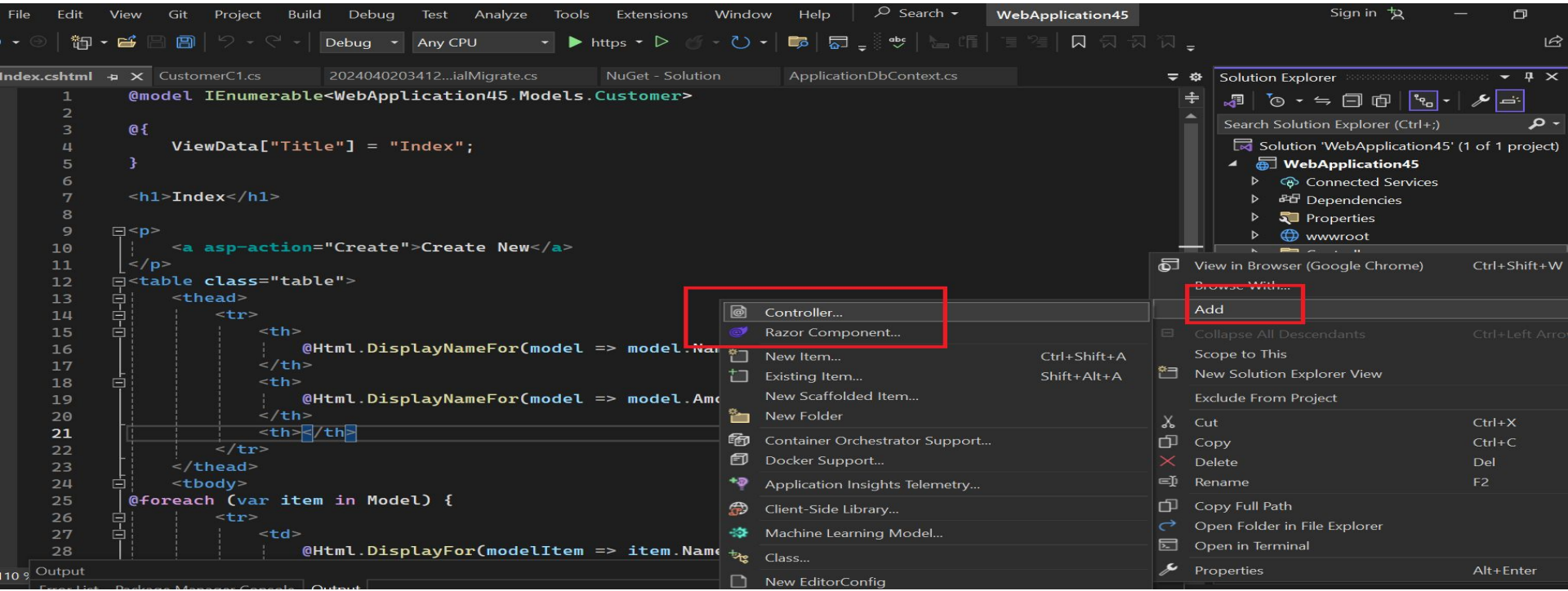
Index

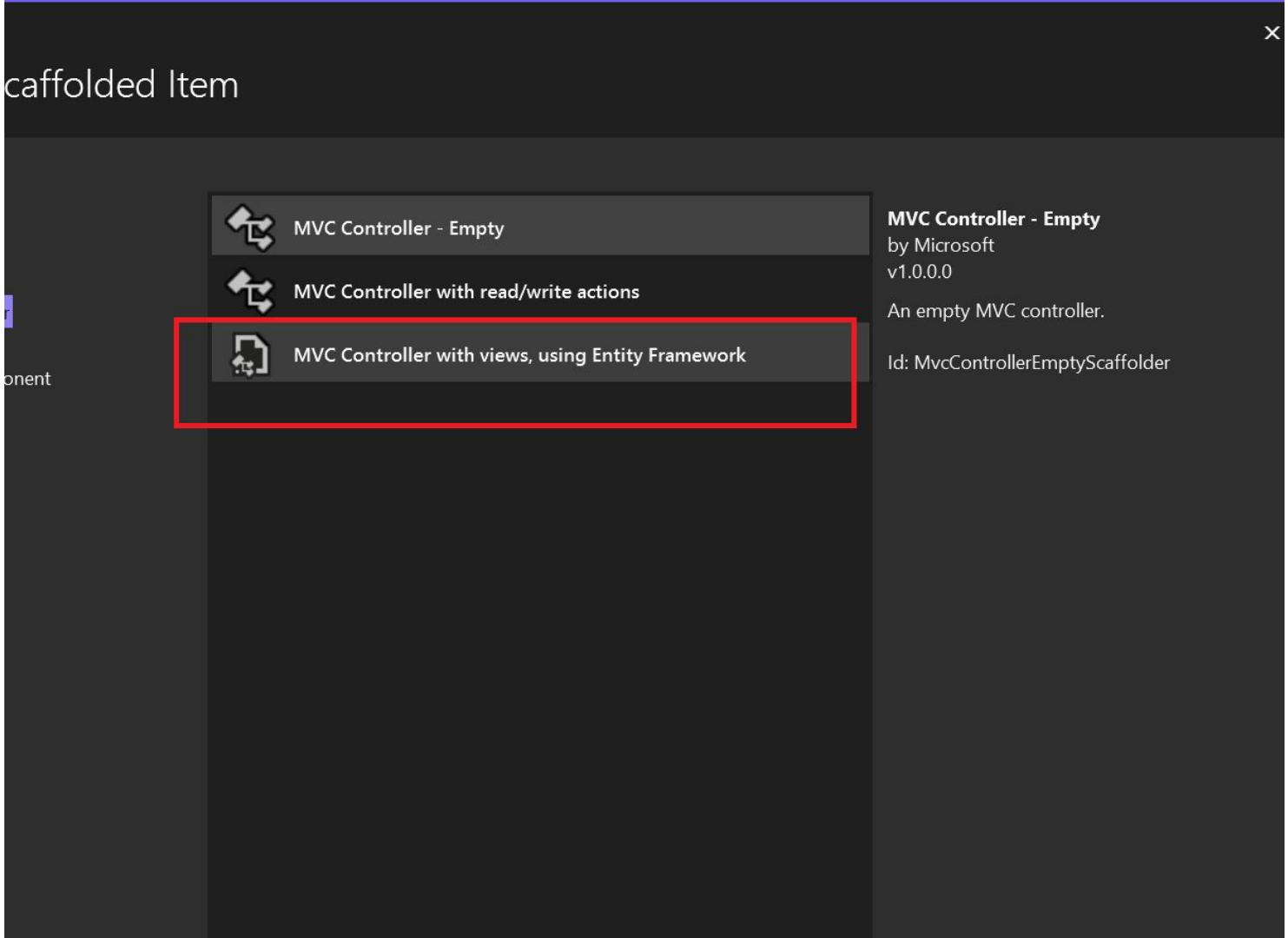
Create New

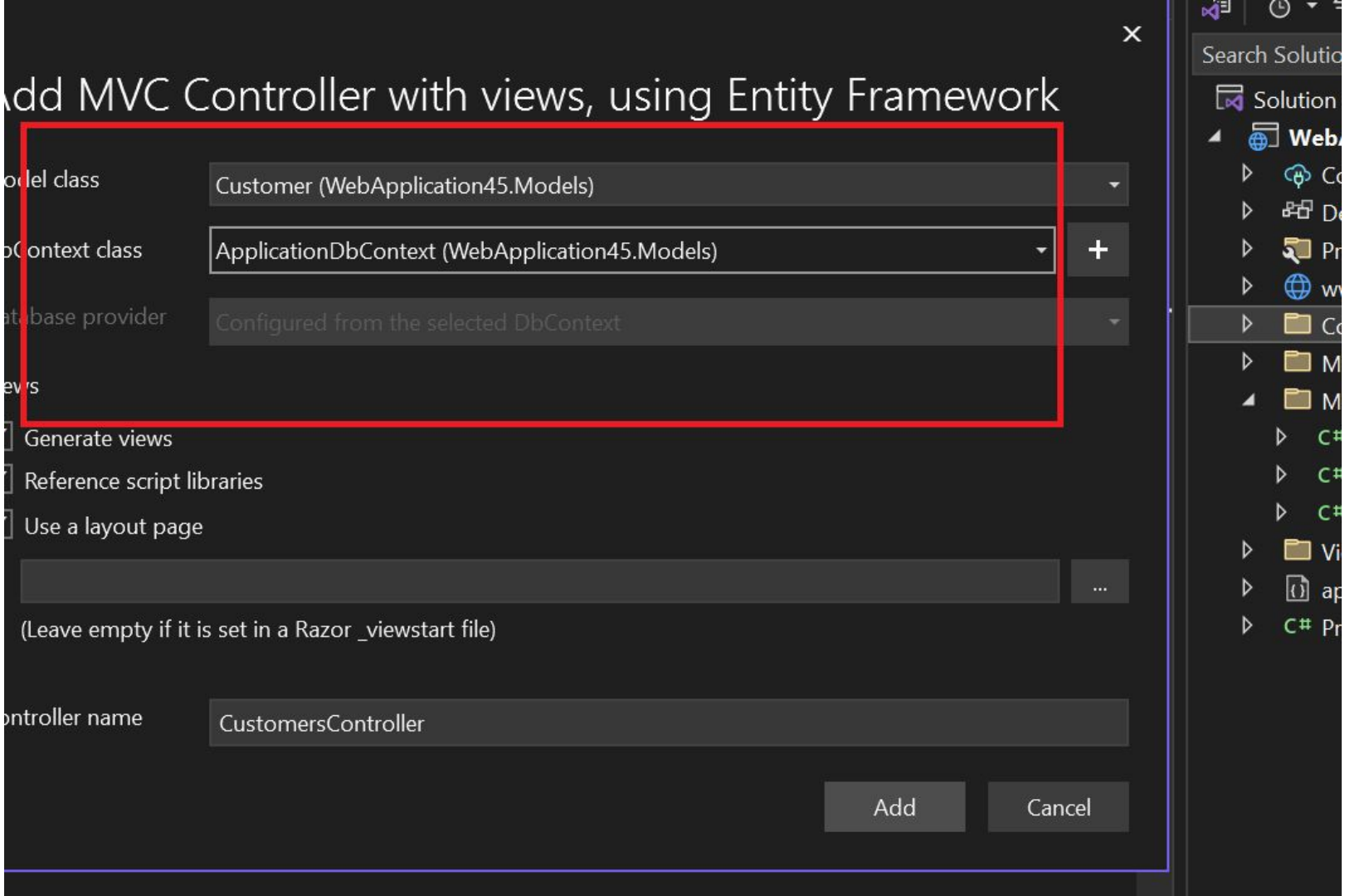
Name	Amount	
Palvi	1000000	Edit | Details | Delete
Puneet	2000000	Edit | Details | Delete

Now, we have to use ef for CRUD OPERATIONS, CREATE ONE MORE CONTROLLER

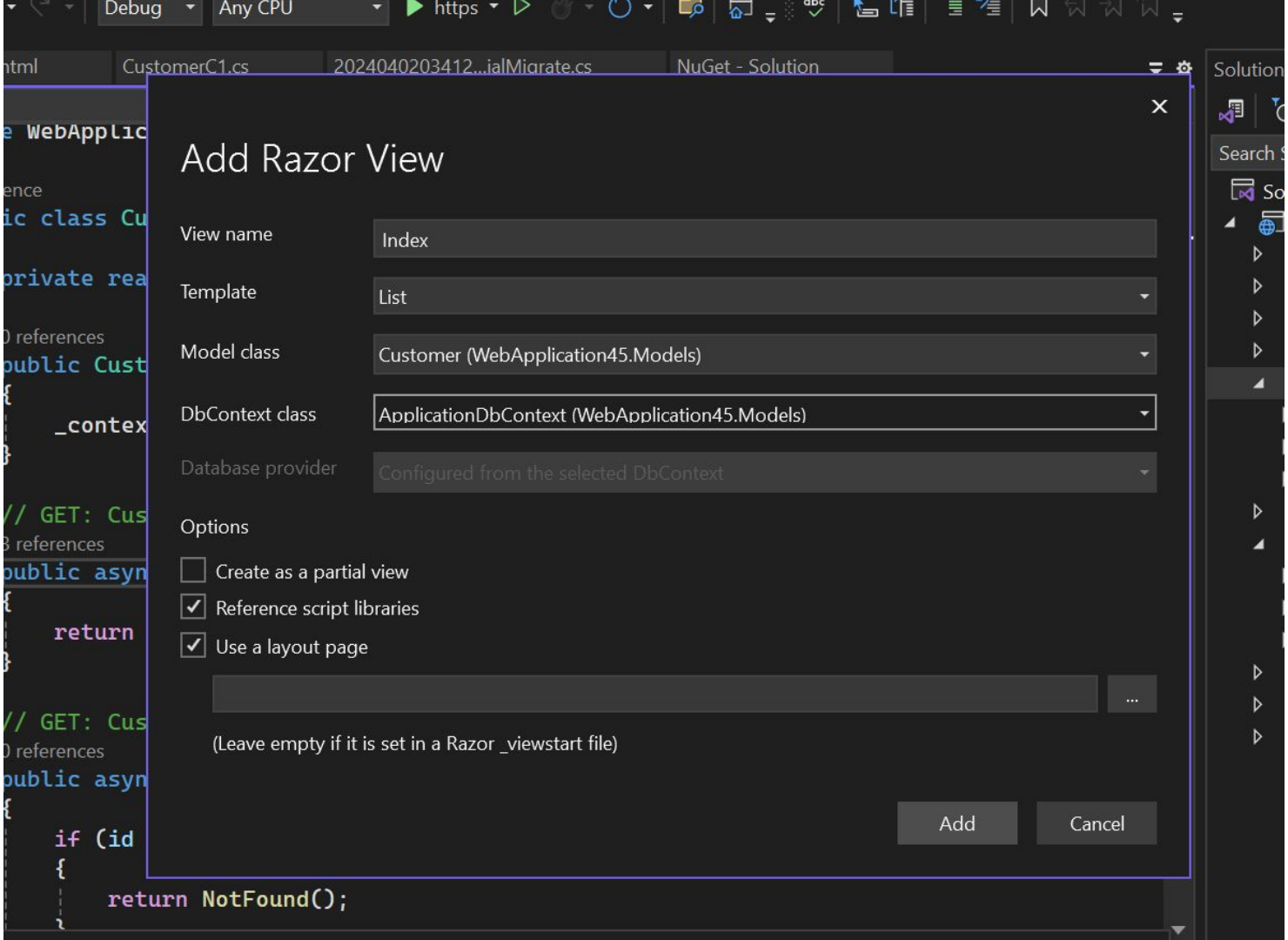
Controller name - CustomerC2




caffolded Item
MVC Controller - Empty
MVC Controller with read/write actions
MVC Controller with views, using Entity Framework
MVC Controller - Empty
by Microsoft
v1.0.0.0
An empty MVC controller.
Id: MvcControllerEmptyScaffolder


dd MVC Controller with views, using Entity Framework
del class
Customer (WebApplication45.Models)
Context class
ApplicationDbContext (WebApplication45.Models)
base provider
Configured from the selected DbContext
ews
Generate views
Reference script libraries
Use a layout page
...
(Leave empty if it is set in a Razor _viewstart file)
ntroller name
CustomersController
Add
Cancel
Search Solutio
Solution

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using WebApplication45.Models;

namespace WebApplication45.Controllers
{
    1 reference
    public class CustomersC2 : Controller
    {
        private readonly ApplicationDbContext _context;

        0 references
        public CustomersC2()
        {
            _context = new ApplicationDbContext();
        }

        // GET: CustomersC2
```


Debug
Any CPU
https
CustomerC1.cs
2024040203412...ialMigrate.cs
NuGet - Solution
Solution
Add Razor View
View name
Index
Template
List
Model class
Customer (WebApplication45.Models)
DbContext class
ApplicationDbContext (WebApplication45.Models)
Database provider
Configured from the selected DbContext
Options
Create as a partial view
Reference script libraries
Use a layout page
...
(Leave empty if it is set in a Razor _viewstart file)
Add
Cancel
return NotFound();

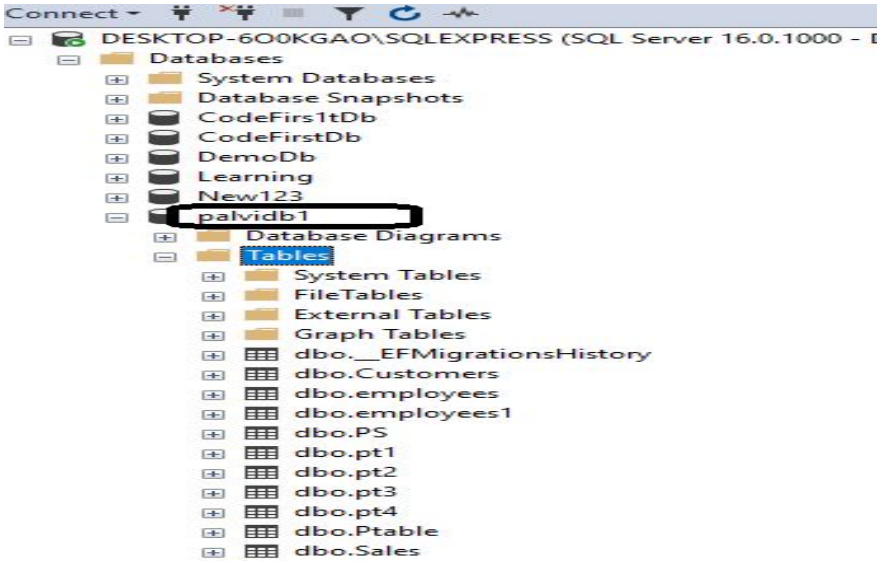
Database First Approach

TYPES OF EF CORE

2. Database First Approach:

- The database approach will help us to create Domain classes and DB Context classes from an existing database using EF Core.
- If you have an existing Database then we should use this approach to work with ORM.

Database first approach works if you have already created some database in your sql server, that database you want to access in your application.Here I want to migrate palvidb1 to mvc

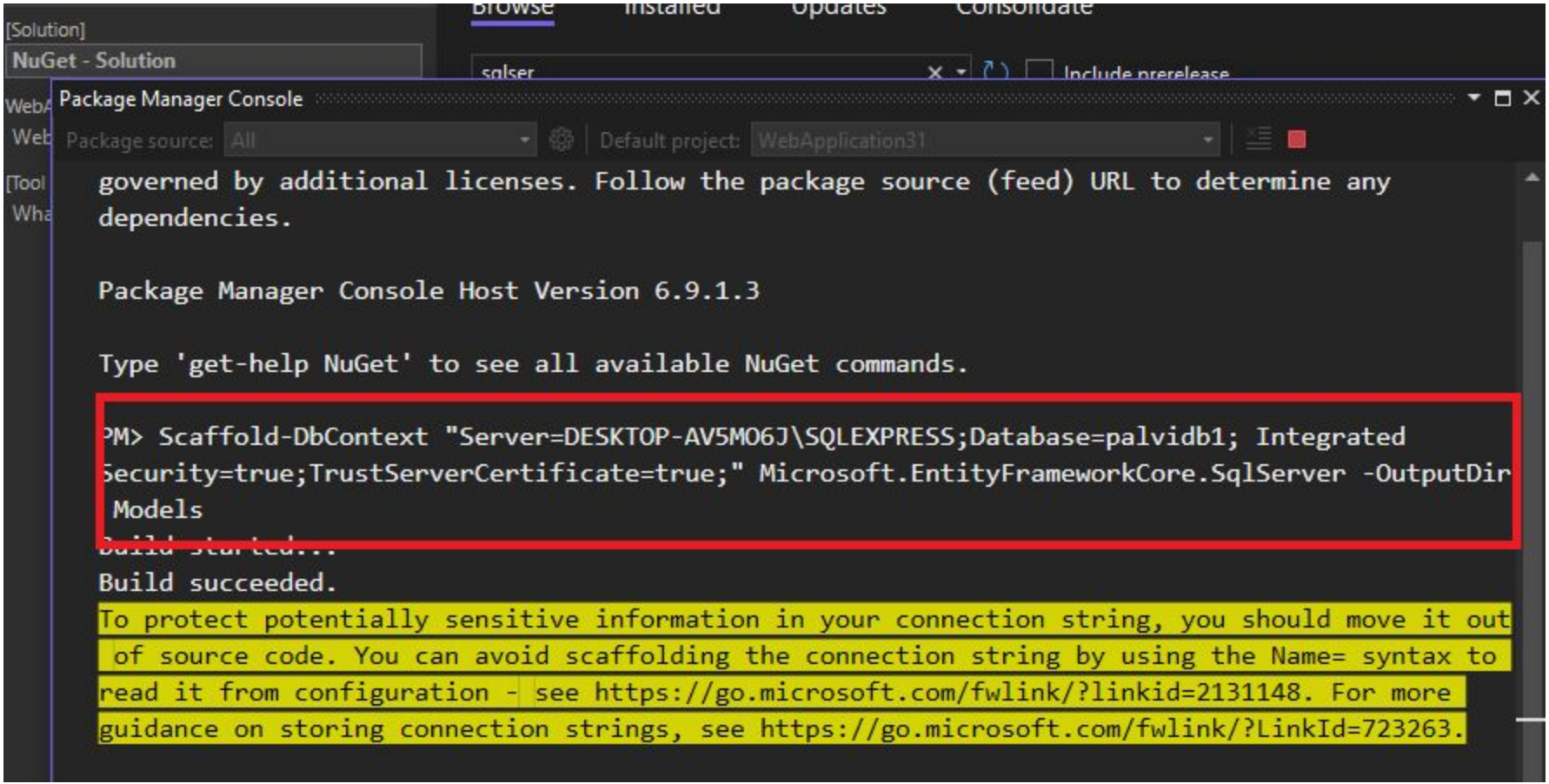


Now create a simple mvc project

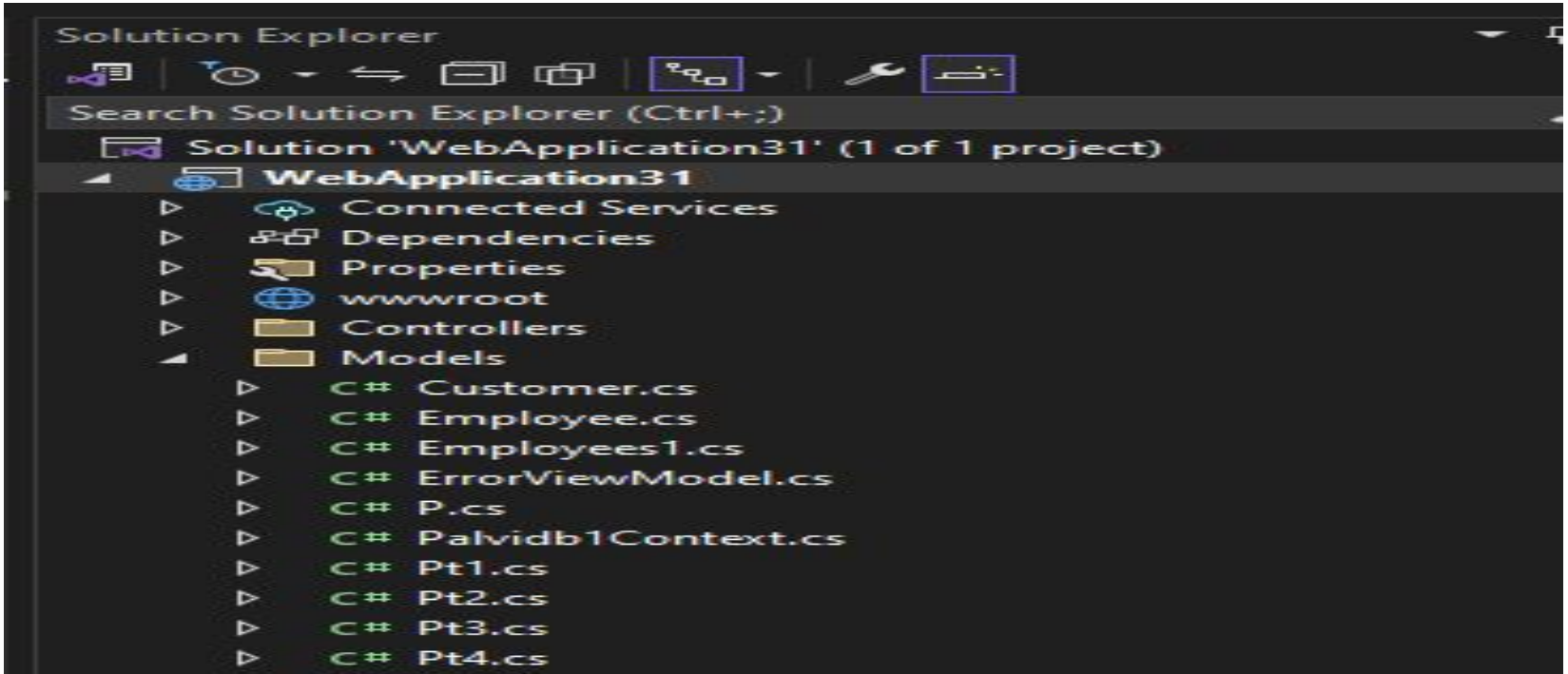
Install 3 packages-
Microsoftentityframeworkcore,Microsoftentityframeworkcore.tools,Microsoftentityframeworkcore.sqlserver

Then open package manager console and write following commands

Scaffold-DbContext
"Server=DESKTOP-AV5MO6J\SQLEXPRESS;Database=palvidb1; Integrated Security=true;TrustServerCertificate=true;"
Microsoft.EntityFrameworkCore.SqlServer -OutputDir Models


[Solution]
NuGet - Solution
Browse
Installed
Updates
Consolidate
Include prerelease
Package Manager Console
Package source: All
Default project: WebApplication31
governed by additional licenses. Follow the package source (feed) URL to determine any dependencies.
Package Manager Console Host Version 6.9.1.3
Type 'get-help NuGet' to see all available NuGet commands.
PM> Scaffold-DbContext "Server=DESKTOP-AV5MO6J\SQLEXPRESS;Database=palvidb1; Integrated Security=true;TrustServerCertificate=true;" Microsoft.EntityFrameworkCore.SqlServer -OutputDir Models
Build started...
Build succeeded.
To protect potentially sensitive information in your connection string, you should move it out of source code. You can avoid scaffolding the connection string by using the Name= syntax to read it from configuration - see https://go.microsoft.com/fwlink/?linkid=2131148. For more guidance on storing connection strings, see https://go.microsoft.com/fwlink/?LinkId=723263.

Now you will see palvidb1 has migrated here in mvc from sql server



Authentication in asp.net core

AUTHENTICATION

- Authentication of user means verifying the identity of the user.
- You might need to present your application only to the authenticated users for obvious reasons
- In other words, we can say that it is a process to validate someone against some data source.

WINDOWS AUTHENTICATION

- This authentication provider is the default provider for asp.net.
- It authenticates the users based on the user's windows accounts.
- Windows authentication relies on the IIS to do the authentication.
- IIS can be configured so that only users on the Windows domain can log in.
- If users attempt to access a page and is not authenticated,
- then the user will be shown a dialogue box that asks the user to enter their username and password.
- Then this information is passed to webserver and checked against the list of users in domain.

FORMS AUTHENTICATION

- It provides a way to handle the authentication using your own custom logic within the application.
- When the user requests a page for the application, asp.net checks for the presence of a special session cookie.
- If the cookie is present, asp.net assumes that the user is authenticated and processes the request.
- If the cookie is not present, asp.net redirects the user to a web form you provide.
- When user is authenticated process the request and indicates this to asp.net by setting a property,
- which creates the special cookie to handle the subsequent requests.

PASSPORT AUTHENTICATION

- It allows using Microsoft's passport service to authenticate users of your application.
- If your users have signed up with a passport and you are having the authentication mode of the application as passport authentication,
- then all authentication duties are offloaded to the passport servers.
- It uses an encrypted cookie mechanism to indicate the authenticated users.
- If users have already signed into passports when they visit the site,
- then they will be considered as authenticated by asp.net.
- Otherwise, they will be redirected to the passport servers to log in.

Security in MVC

AUTHORIZATION

- Authorization is the process of checking whether user has access to resources they requested.
- In asp.net, there is two forms of authorization available,
- one is file authorization and another is URL authorization.
- **Syntax:**

```
<system.web>
<authorization>
<allow users = "Sakshi Dhameja"/>
<deny users = "*"/>
</authorization>9
</system.web>
```

AUTHORIZATION

- This code will allow user SakshiDhameja and deny all other users to access that application.
- If you want to give permission for more users then just add usernames separated with a comma like SakshiDhameja, coursera, edu, etc.
- and if you want to allow only admin roles to access the application
- and deny permission for all the roles, then write the following code in web.config.

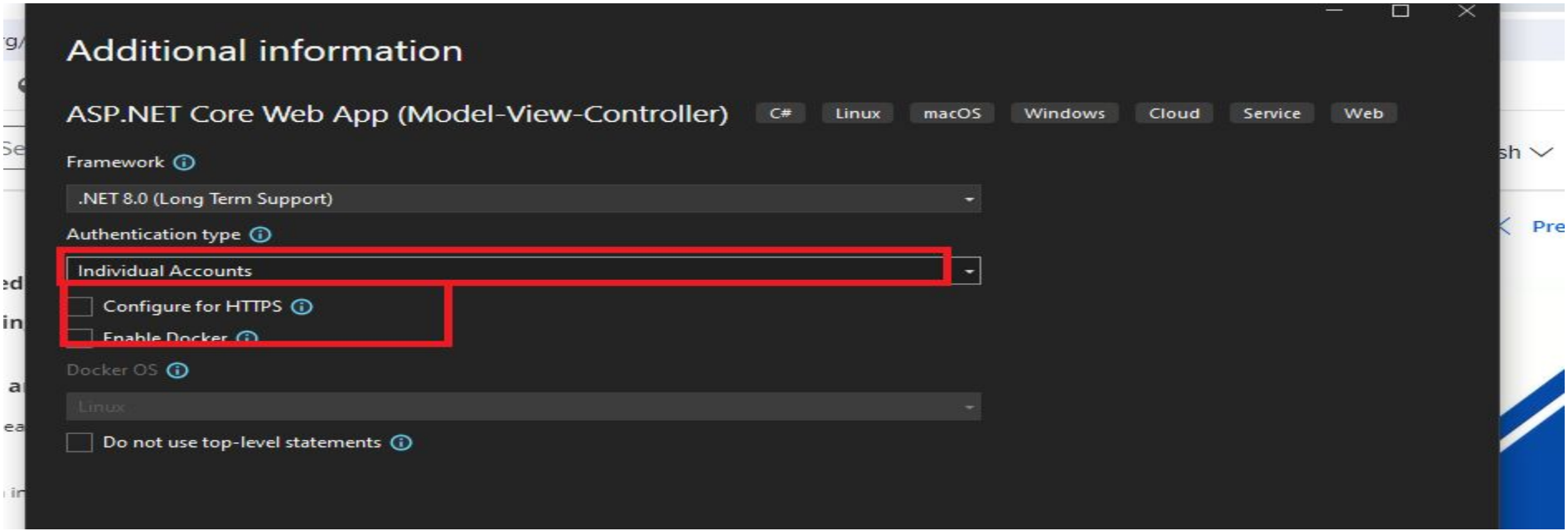
```
<system.web>
<Authorization>
<allow roles = "Admin"/>
<deny users = "*"/>
</Authorization>
</system.web>
```

AUTHORIZATION

- **File authorization:**
  - File authorization is performed by the FileAuthorizationModule.
  - Uses Control List of web page to resolve whether a user should have access or not.
  - ACL permissions are confirmed of the user's windows identity.
- **URL authorization:**
  - In configuration file you can specify the authorization rules for various directories or files using <authorization> element.

Security in Asp.net core

Create a new mvc project, while creating select highlighted options



Run your code without doing any changes, at left side, there is a option of register, register two emails randomly with some password. Then you will be redirected to the page which is appearing in the next slide.

There you will see apply migrations, make a click on that .

A database operation failed while processing the request.

SqlException: Cannot open database "aspnet-WebApplication30-76f8db85-ee69-4122-9e17-5953830a94c5" requested by the login. The log

Applying existing migrations may resolve this issue

There are migrations that have not been applied to the following database(s):

ApplicationDbContext

- 00000000000000_CreateIdentitySchema

Apply Migrations

In Visual Studio, you can use the Package Manager Console to apply pending migrations to the database:

```
PM> Update-Database
```

Alternatively, you can apply pending migrations from a command prompt at your project directory:

```
> dotnet ef database update
```

A database operation failed while processing the request.

SqlException: Cannot open database "aspnet-WebApplication30-76f8db85-ee69-4122-9e17-5953830a94c5" requested by the login. The log

Applying existing migrations may resolve this issue

There are migrations that have not been applied to the following database(s):

ApplicationDbContext

- 00000000000000_CreateIdentitySchema

Applying Migrations...

In Visual Studio, you can use the Package Manager Console to apply pending migrations to the database:

```
PM> Update-Database
```

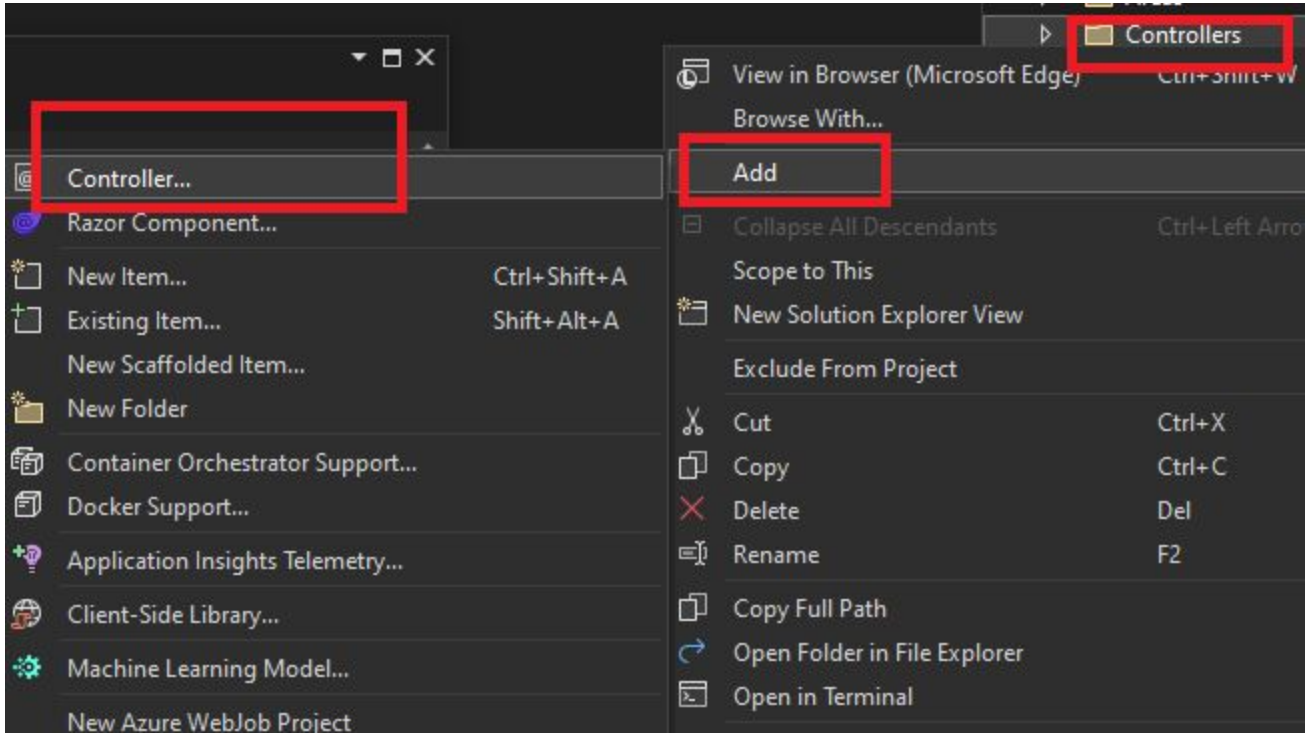
Register confirmation

This app does not currently have a real email sender registered, see these docs for how to configure a real email sender. Normally this would b

here to confirm your account

Confirm email

Thank you for confirming your email.


Controllers
View in Browser (Microsoft Edge)
Browse With...
Add
Collapse All Descendants
Ctrl+Left Arro
Scope to This
New Solution Explorer View
Exclude From Project
Cut
Ctrl+X
Copy
Ctrl+C
Delete
Del
Rename
F2
Copy Full Path
Open Folder in File Explorer
Open in Terminal
Controller...
Razor Component...
New Item...
Ctrl+Shift+A
Existing Item...
Shift+Alt+A
New Scaffolded Item...
New Folder
Container Orchestrator Support...
Docker Support...
Application Insights Telemetry...
Client-Side Library...
Machine Learning Model...
New Azure WebJob Project

After creating controller, you have to create three methods index,policy and details. Create corresponding views also. Then make some changes in program.cs as “name of the controller” and corresponding index method.

Then when you load the page, index page will be opened, there you will see the login page, add some random details, you will be able to see, it says invalid login, so you have to enter only that emails which you have registered in the beginning.