ASP.NET Core MVC Application Notes

[ASP.NET Core MVC Application Notes 2](#_Toc121053137)

[ASP.NET Core 6 MVC Project with Individual Accounts Authentication 3](#_Toc121053138)

[Project Template 3](#_Toc121053139)

[Database Connection String 3](#_Toc121053140)

[Initial Project Folders 4](#_Toc121053141)

[Launching the Project for the First Time 4](#_Toc121053142)

[SSL Certificate Warning when Launching the Project for the First Time 4](#_Toc121053143)

[Register New User 6](#_Toc121053144)

[Apply Migration to Create Database 6](#_Toc121053145)

[Adding Northwind Data 7](#_Toc121053146)

[CRUD in ASP.NET Core MVC 8](#_Toc121053147)

[Scaffold Entity Model and Data-Context from the Database 8](#_Toc121053148)

[Register NorthwindDbContext with Dependency Injection Container 9](#_Toc121053149)

[Add Customer Controller 9](#_Toc121053150)

[Customers CRUD Functions 10](#_Toc121053151)

[Add Customers to Navigation Menu 11](#_Toc121053152)

[Table Border CSS 11](#_Toc121053153)

[Adding TypeScript Support to ASP.NET Core MVC Project 12](#_Toc121053154)

[Nuget Packages 12](#_Toc121053155)

[TypeScript Folder 12](#_Toc121053156)

[TypeScript Configuration File: tsconfig.json 12](#_Toc121053157)

[Testing TypeScript Project Configuration 13](#_Toc121053158)

# ASP.NET Core MVC Application Notes

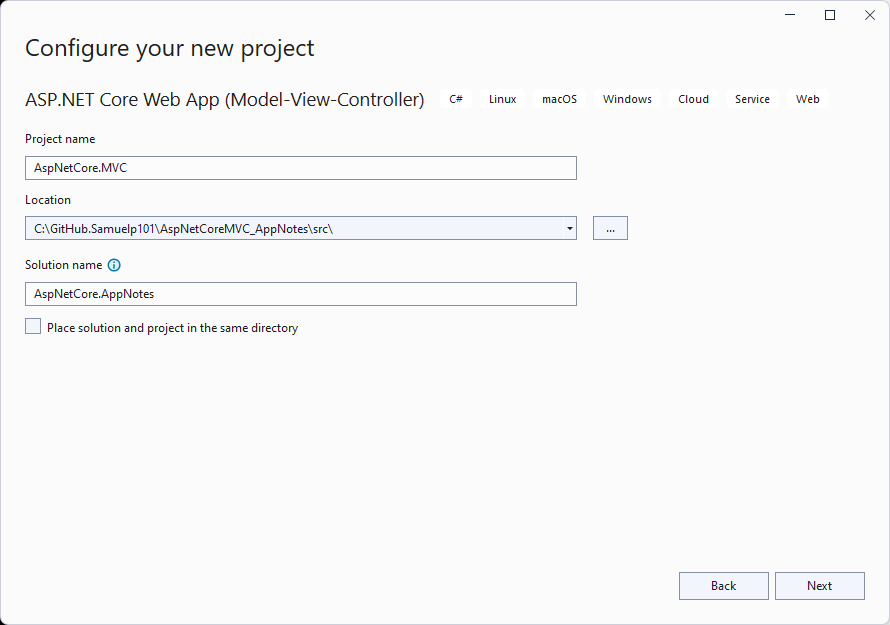
This repos contains sample projects and application notes about ASP.NET Core MVC. While the sample project is created for a specific version, each of the subjects covered is generally applicable from version 3.1 to the latest version. The latest version as of December 2022 is ASP.NET Core 6.

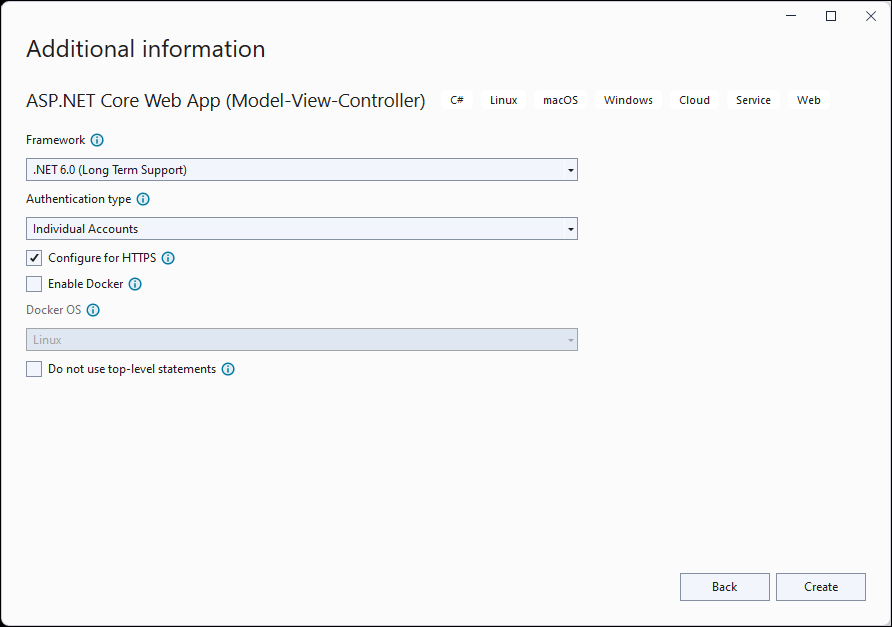
ASP.NET Core 6 MVC with Individual Accounts Authentication

# ASP.NET Core 6 MVC Project with Individual Accounts Authentication

### Project Template

Project created with the “ASP .NET Core Web App (Model-View-Controller)” template





Project Name: AspNetCore.MVC

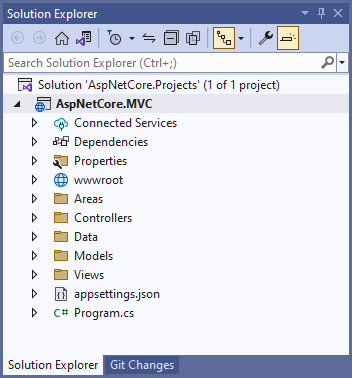
### Database Connection String

"NorthwindDB2021": "Server=ENVY101;Database=NorthwindDB2021;Trusted\_Connection=True;MultipleActiveResultSets=true"

Update connection string in Program.cs

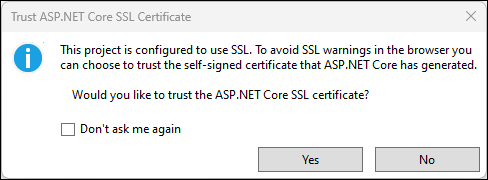
var connectionString = builder.Configuration.GetConnectionString("NorthwindDB2021");

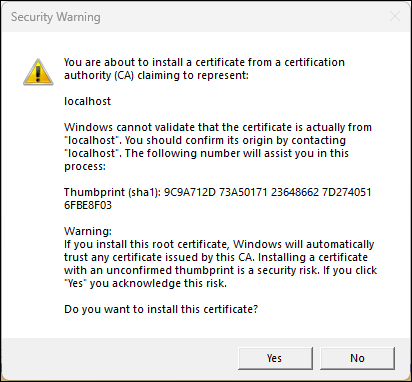
### Initial Project Folders

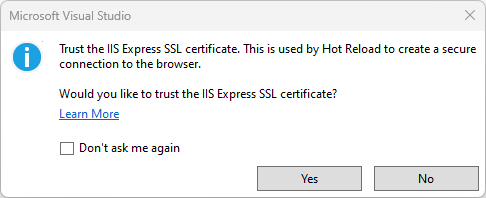


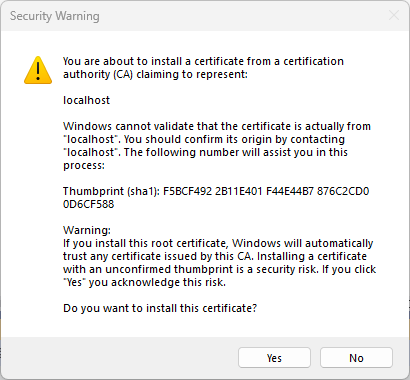
### Launching the Project for the First Time

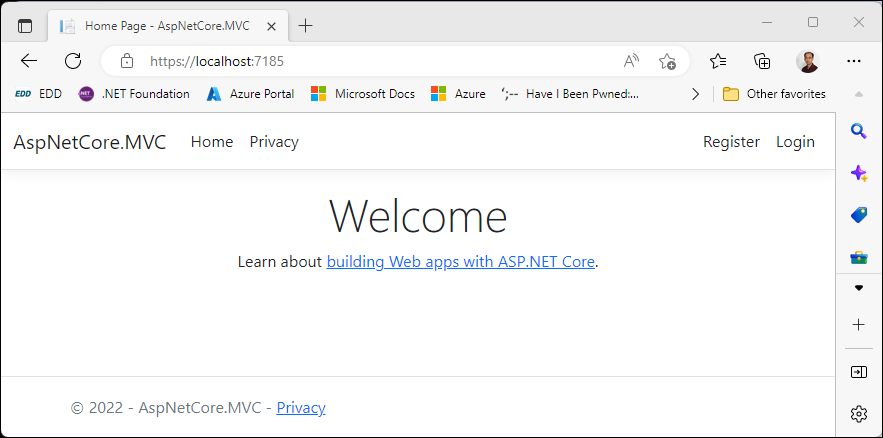
#### SSL Certificate Warning when Launching the Project for the First Time





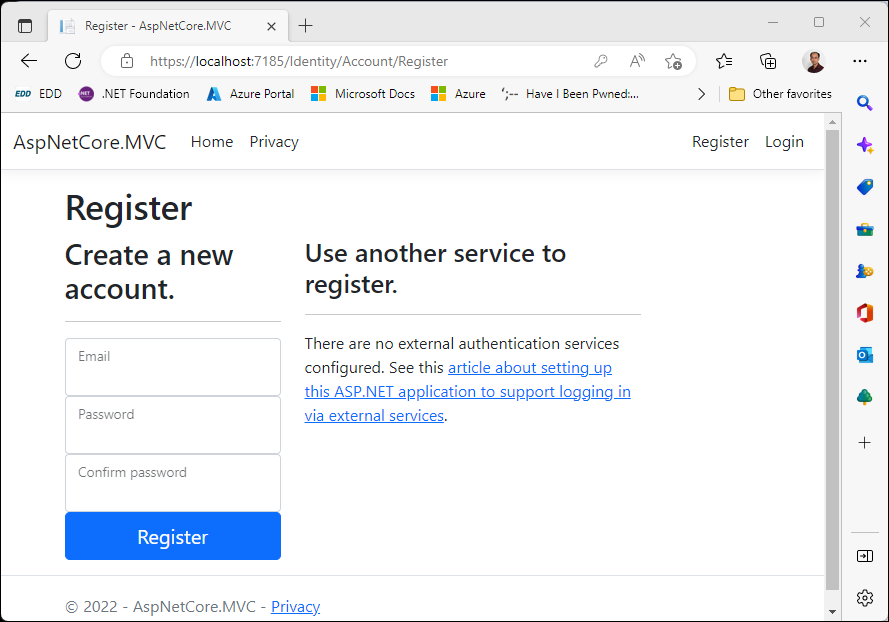






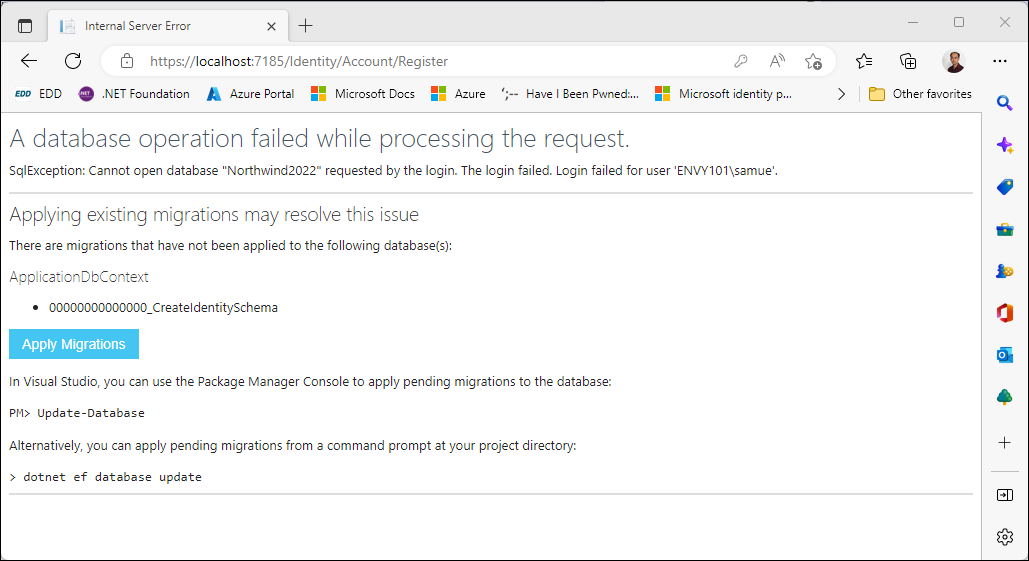
### Register New User

From the initial application screen, click on Register to register a new user.



Since the database has not been created, the “database operation failed while processing the request” is shown.

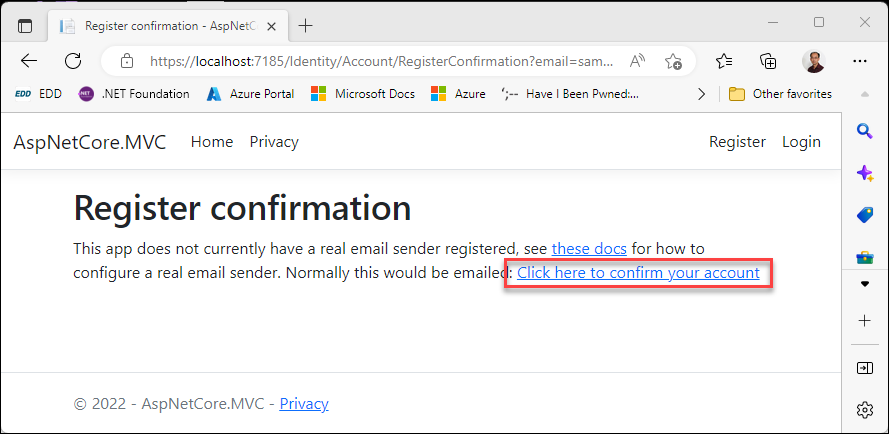
#### Apply Migration to Create Database



Click on “Apply Migrations” to generate the database, and refresh browser to continue.

After new user is added to the database, the “Register confirmation” screen is shown to provide a simulated environment to configure the new user’s email.

Click on “Click here to confirm your account”.



Note:

The initial database can also be created via the following option:

* Execute “Update-Database” from the “Package Manager Console”
* Execute the “dotnet ef database update” from CLI

### Adding Northwind Data

Data from the Northwind database is added to the database via the following script in the project’s database folder:

NorthwindDB2021\_Create\_Database\_Script(v1.0).sql

# CRUD in ASP.NET Core MVC

Continue with the AspNetCore.MVC project created in the previous section, go thru the steps to create CRUD functions targeting the Customers table in the database.

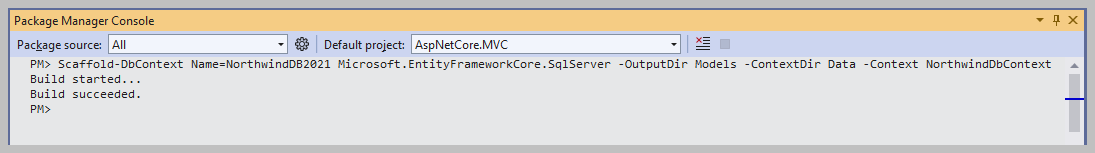
### Scaffold Entity Model and Data-Context from the Database

In the previous section, the NorthwindDB2021 database is created and populated with data from the Northwind database.

From the “Package Manager Console”, execute the following to scaffold entity models and data-context.

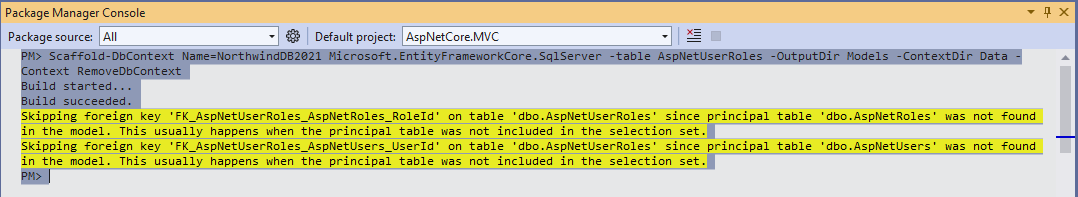
Scaffold-DbContext Name=NorthwindDB2021 Microsoft.EntityFrameworkCore.SqlServer -OutputDir Models -ContextDir Data -Context NorthwindDbContext

Note: The above is a single line of command, outputting the generated data-context to the “\Data” folder and entity-models to the “\Models” folder.



Dbo.AspNetUserRoles

Scaffold-DbContext Name=NorthwindDB2021 Microsoft.EntityFrameworkCore.SqlServer -table AspNetUserRoles -OutputDir Models -ContextDir Data -Context RemoveDbContext



Skipping foreign key 'FK\_AspNetUserRoles\_AspNetRoles\_RoleId' on table 'dbo.AspNetUserRoles' since principal table 'dbo.AspNetRoles' was not found in the model. This usually happens when the principal table was not included in the selection set.

Skipping foreign key 'FK\_AspNetUserRoles\_AspNetUsers\_UserId' on table 'dbo.AspNetUserRoles' since principal table 'dbo.AspNetUsers' was not found in the model. This usually happens when the principal table was not included in the selection set.

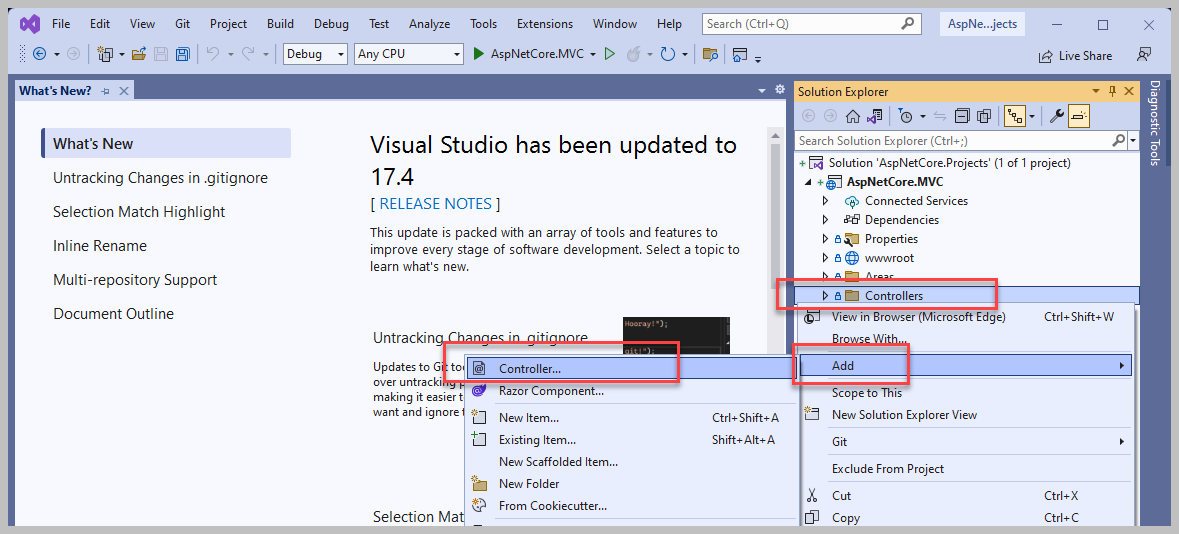
Note: Delete RemoveDbContext.cs from the project.

### Register NorthwindDbContext with Dependency Injection Container

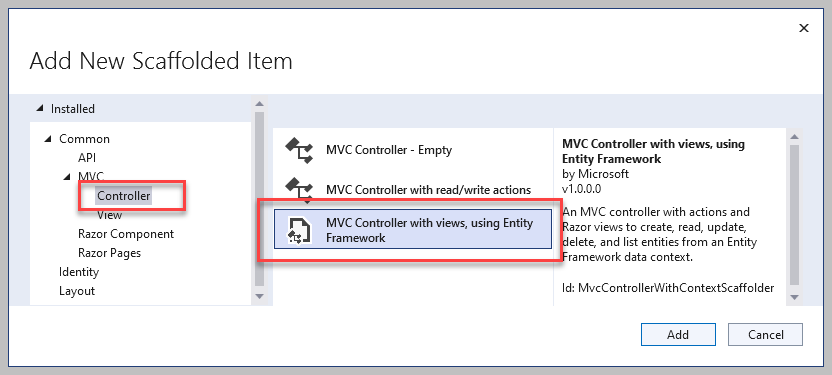
In Program.cs, add the following:

builder.Services.AddDbContext<NorthwindDbContext> (options => options.UseSqlServer(connectionString));

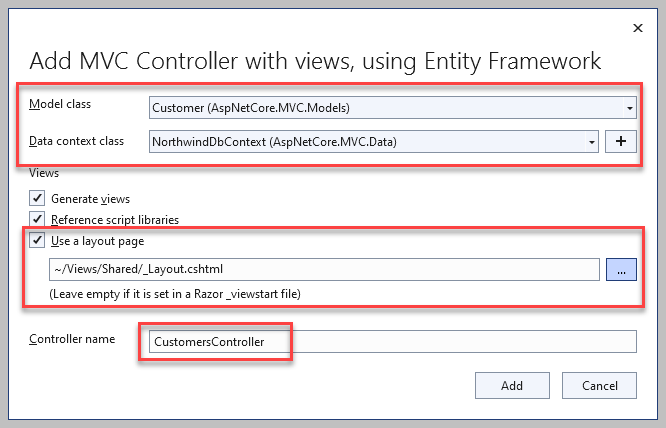
### Add Customer Controller



From “Solution Explorer”, righ click on “\Controller” folder, click “Add” and select “Controller”, to bring up the following “Add New Scaffolded Item” screen.



Select “Controller” on the left pane. Selec “MVC Controller with views, using Entity Framework” and then click “Add”.



Select Customer class, NorthwindDbContext, \_Layout, keep the default controller name and click Add.

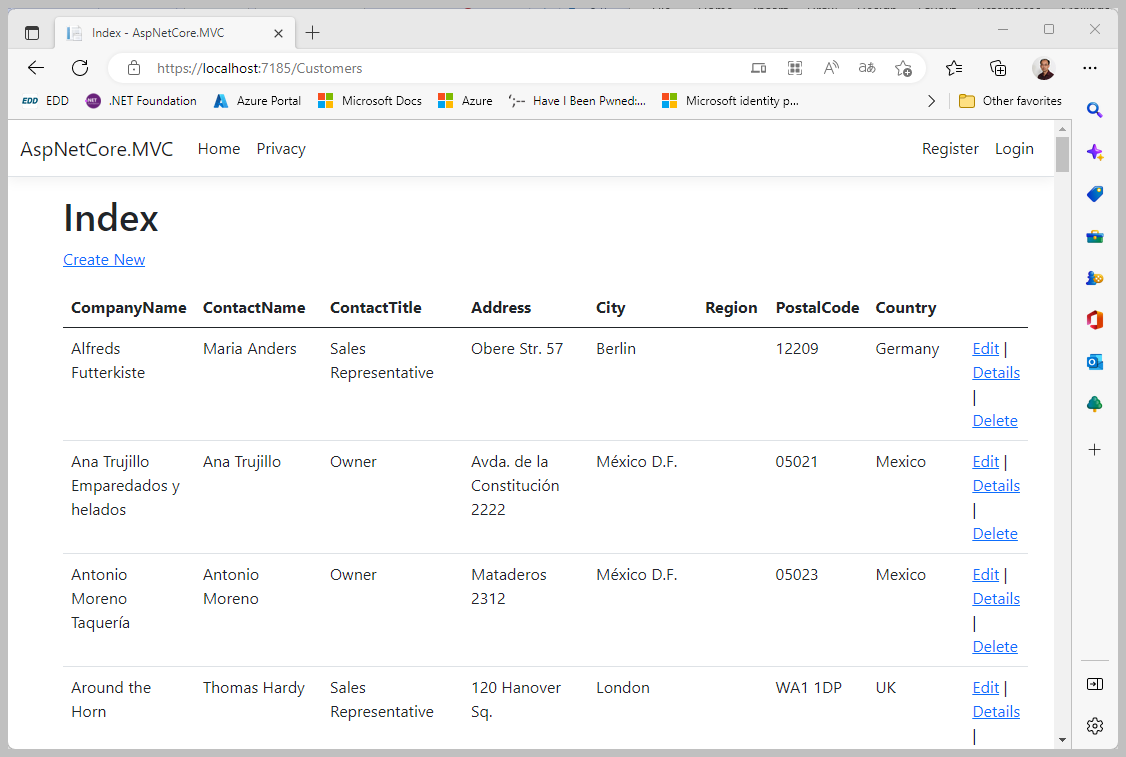
Visual Studio ‘s scaffolding engine will add the necessary Nuget package and generate the following codes to support CRUD functions for the Customers table:

* \Controllers\CustomersController.cs
* \Views\Customers\Create.cshtml
* \Views\Customers\Delete.cshtml
* \Views\Customers\Details.cshtml
* \Views\Customers\Edit.cshtml
* \Views\Customers\Index.cshtml

### Customers CRUD Functions

Launch the app and navigate to the following URL to review the newly created CRUD functions for the Customers table.

<https://localhost:7185/Customers>



Note:

Without having to write any code, Visual Studio generated CRUD function codes to Create, Read, Update and Delete records for the Customers table are fully function.

While these codes may not serve 100% of we needed to accomplish, it’s a good starting point.

One of the key benefits is that we can use the scaffolding engine to generate entity class for each of the tables in the database, which is especially useful in the scenarios where we have large number of tables. For this example, the scaffolding engine entity models from 20 tables in the NorthwindDB2021 database, of which 7 of these tables are related to Identity data and 13 of them are data from the sample Northwind database.

Reviewing entity models in the “\Models” folder, we can find 34 entity models were generated, where the number of models is greater than the actual number of tables. The addition models are generated for the database ‘s Views.

### Add Customers to Navigation Menu

In \_Layout.cshtml, add the following to the navigation menu:

<li class="nav-item">

<a class="nav-link text-dark" asp-area="" asp-controller="Customers" asp-action="Index">Customers</a>

</li>

### Table Border CSS

To improve the look a bit, add “table-bordered” css class to the table element to the Customers’ Index.cshtml

<table class="table table-bordered">

# Adding TypeScript Support to ASP.NET Core MVC Project

Continue with the AspNetCore.MVC project from the previous section, add TypeScript support to the project.

### Nuget Packages

Add the following Nuget package:

* Microsoft.TypeScript.MSBuild 4.9.3

### TypeScript Folder

Create the following project folder for TypeScript codes:

* \wwwroot\typescript\

All TypeScript codes are placed in this folder. Transpiled JavaScript codes for TypeScript code files created in a sub-folder under the “\wwwroot\typescript\” will be created in corresponding sub-folder under the “wwwroot\js\” folder.

### TypeScript Configuration File: tsconfig.json

Add TypeScript configuration file (tsconfig.json), to the “\wwwroot\typescript|” folder, and modify the content as follow:

{

"compilerOptions": {

"noImplicitAny": false,

"noEmitOnError": true,

"removeComments": false,

"sourceMap": true,

"target": "es5",

"outDir": "../js"

},

"compileOnSave": true,

"exclude": [

"node\_modules",

"wwwroot"

]

}

### Testing TypeScript Project Configuration

Add the following TypeScript file to the “\wwwroot\typescript\” folder:

HelloWorld.ts

function HelloWorld() {

document.getElementById("msg")

.innerHTML = "<h1>Hello World!</h1>";

}

Update the “\Views\Home\Index.cshtml” file with the following:

Index.cshtml

@{

ViewData["Title"] = "Home Page";

}

<div class="text-center">

<h1 class="display-4">Welcome</h1>

<p>

Learn about

<a href="https://docs.microsoft.com/aspnet/core">

building Web apps with ASP.NET Core

</a>.

</p>

</div>

<br />

<button onclick="HelloWorld()">Say Hello</button>

<h3>Content from TypeScript</h3>

<div id="msg"></div>

<script src="~/js/helloworld.js"></script>

<script>

window.onload = OnLoad;

function OnLoad() {

console.log("Index page loaded!");

}

</script>

Launch the project and check to make sure that TypeScript is transpiled to the “\wwwroot\js” folder and the code is able to run without error.