



C# Web – MVC

PRO/PRW - C# Web 1

**DE HOGESCHOOL
MET HET NETWERK**

Hogeschool PXL – Elfde-Liniestraat 24 – B-3500 Hasselt
www.pxl.be - www.pxl.be/facebook



Doel

- ASP.Net Core toepassing
 - MVC Web Application
 - Models
 - Views
 - Controllers
 - Custom Model Validation
 - Entity Framework Core

ASP.Net Core MVC

CUSTOM MODEL VALIDATION

```
namespace MVCModelValidation.Models
{
    public class TestData
    {
        public int? TestDataId { get; set; }
        [Required]
        [StringLength(10)]
        public string? TestText { get; set; }
        //[CustomDate]
        public DateTime? TestDate { get; set; }
    }
}
```

- Folder Data toevoegen
 - Add class LocalData

```
namespace MVCModelValidation.Data
{
    public class LocalData
    {
        public static List<TestData> TestDataList = new List<TestData>();
    }
}
```

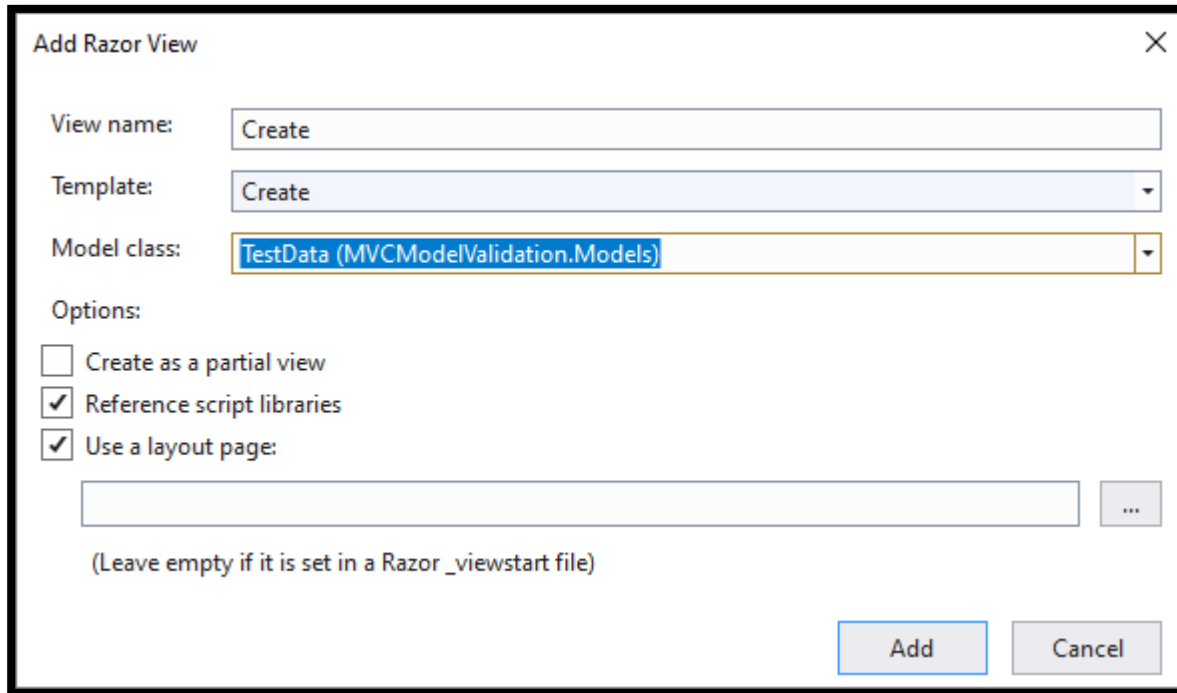
Add Folder

- CustomModelValidations
 - Add class
 - CustomDate.cs

```
using Microsoft.AspNetCore.Mvc.ModelBinding.Validation;
```

```
public class CustomDate : Attribute, IModelValidator
{
    public IEnumerable<ModelValidationResult> Validate(ModelValidationContext context)
    {
        var dtm = DateTime.Now;
        var lst = new List<ModelValidationResult>();
        if (DateTime.TryParse(context.Model.ToString(), out dtm))
        {
            if (dtm > DateTime.Now)
                lst.Add(new ModelValidationResult("", "Date of Birth cannot be in the future"));
            else if (dtm < new DateTime(1900, 1, 1))
                lst.Add(new ModelValidationResult("", "Date of Birth should not be before 1900"));
        }
        else
        {
            lst.Add(new ModelValidationResult("", "Not a valid date!"));
        }
        return lst;
    }
}
```

- Add Controller TestDataController
- Add Folder Views/TestData
 - Add Razor View (Model TestData)
 - Index – Template(List)
 - Create – Template(Create)



Add Razor View

View name:

Template:

Model class:

Options:

☐ Create as a partial view

☒ Reference script libraries

☒ Use a layout page:

...

(Leave empty if it is set in a Razor _viewstart file)

```
public class TestDataController : Controller
{
    public IActionResult Index()
    {
        return View(LocalData.TestDataList);
    }
    [HttpGet]
    public IActionResult Create()
    {
        var testData = new TestData();
        return View(testData);
    }
    [HttpPost]
    public IActionResult Create(TestData testData)
    {
        if (ModelState.IsValid)
        {
            LocalData.TestDataList.Add(testData);
            return RedirectToAction("Index");
        }
        return View(testData);
    }
}
```

Views/Home/Index.cshtml

```
@{  
    ViewData["Title"] = "Home Page";  
}  
  
<div class="text-center">  
    <h1 class="display-4">Custom Model validation</h1>  
    <a asp-controller="TestData" asp-action="Index">  
        Test Data - Custom Model Validation</a>  
</div>
```


ASP.NET CORE

MVC Web Application – Oefening07

ASP.NET CORE

ASP.Net Core MVC

ENTITY FRAMEWORK

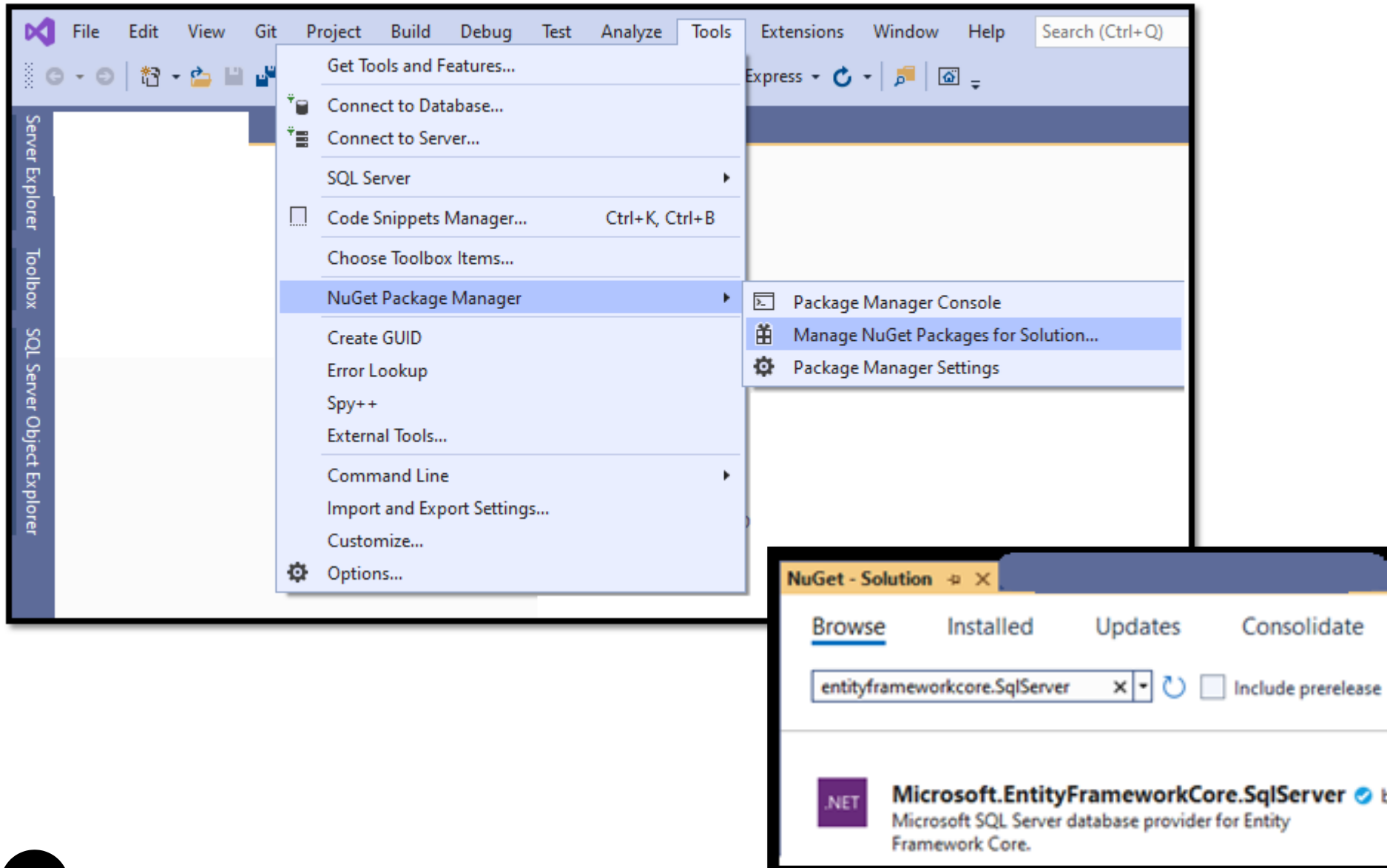
MVCTestEFCore



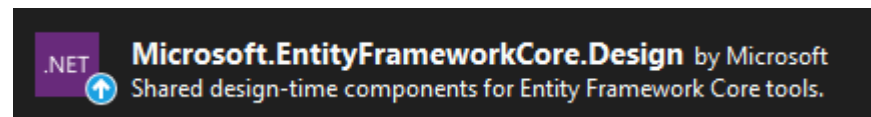
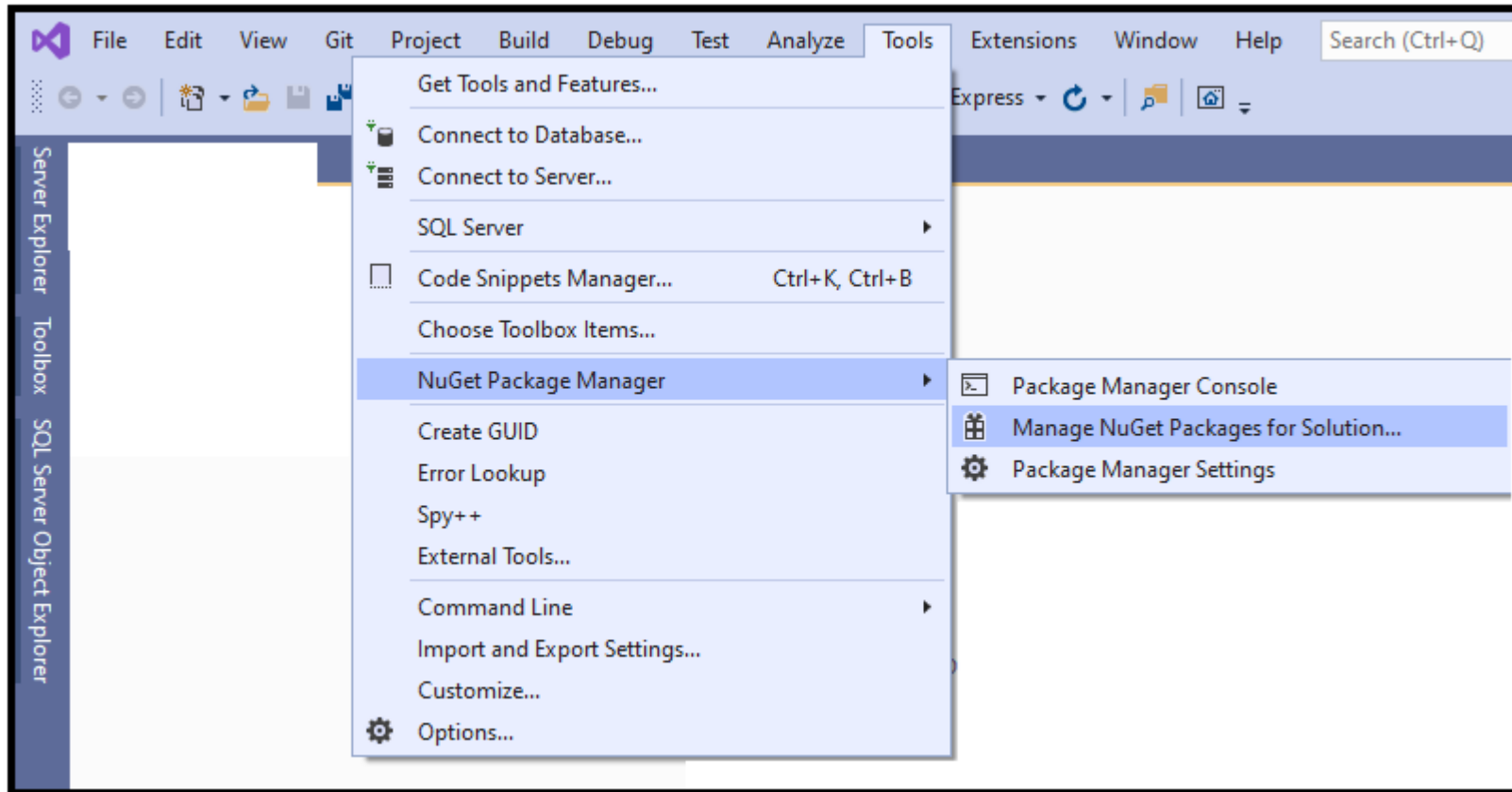
Web Application (Model-View-Controller)

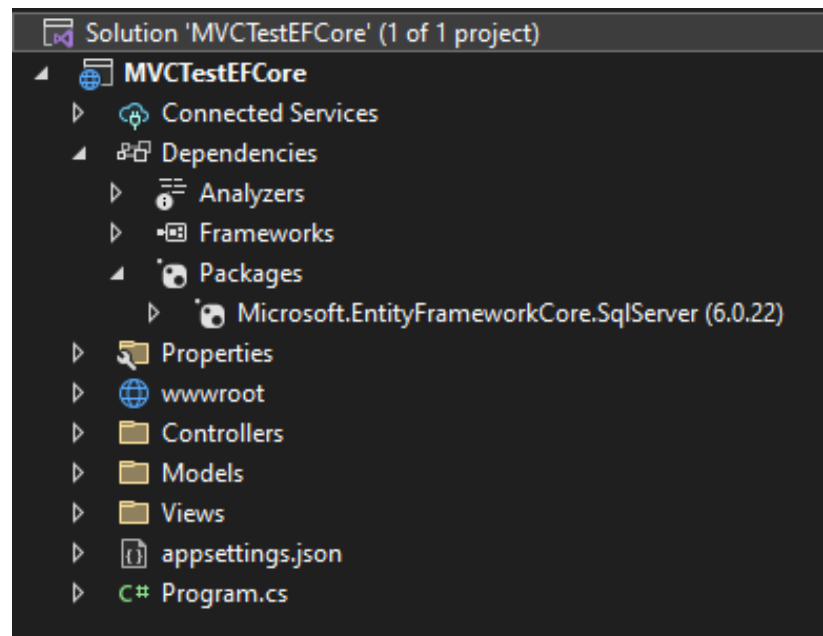
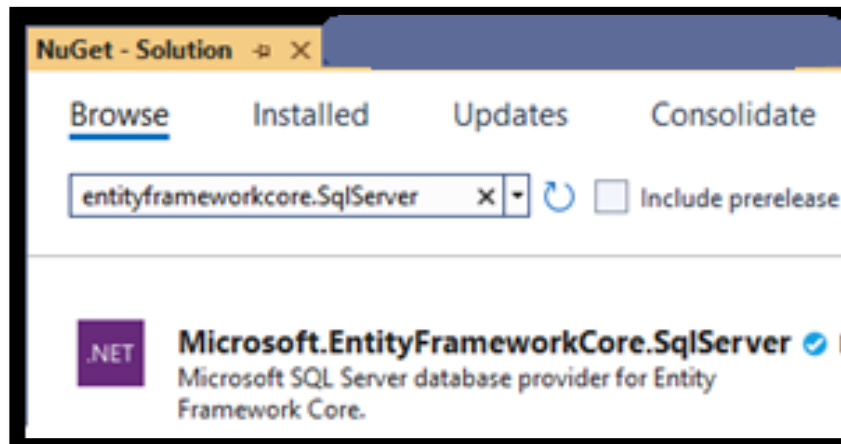
A project template for creating an ASP.NET Core application with example ASP.NET Core MVC Views and Controllers. This template can also be used for RESTful HTTP services.

Entity Framework Core – SQL Server – Nuget Package Manager



Entity Framework Core – SQL Server – Nuget Package Manager





Project MVCEFCore

- Create Folder Data
 - Add class AppDbContext.cs

```
namespace MVCTestEFCore.Data
{
    public class AppDbContext
    {
    }
}
```

DbContext

```
using Microsoft.EntityFrameworkCore;
```

```
using Microsoft.EntityFrameworkCore;
namespace MVCTestEFCore.Data
{
    public class AppDbContext : DbContext
    {
        public AppDbContext(DbContextOptions<AppDbContext> options) :
            base(options)
        {
        }
    }
}
```


Project MVCEFCore

- Folder Data
 - Add class `LocalConnectionString.cs`

```
namespace MVCTestEFCore.Data
{
    public static class LocalConnectionString
    {
        public static string ConnectionString =>
            CreateLocalConnectionString();

        private static string CreateLocalConnectionString()
        {
            var sb = new StringBuilder();
            sb.Append(@"Server=(localdb)\mssqllocaldb;");
            sb.Append("Database=mvcefcoredb;");
            sb.Append("Trusted_Connection=true;");
            sb.Append("MultipleActiveResultSets=true");
            return sb.ToString();
        }
    }
}
```

Program.cs

```
using Microsoft.EntityFrameworkCore;
using MVCTestEFCore.Data;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllersWithViews();

var connString = LocalConnectionString.ConnectionString;

builder.Services.AddDbContext<AppDbContext>(
    options => options.UseSqlServer(connString));

var app = builder.Build();
```

Models folder

- Add FirstTable.cs

```
using System.ComponentModel.DataAnnotations;

namespace MVCTestEFCore.Models
{
    public class FirstTable
    {
        public int FirstTableId { get; set; }
        [Required]
        public string TextColumn { get; set; }
    }
}

// FirstTableId = primary key
```

Controllers folder

- Add empty MVC controller
 - FirstTableController

```
namespace MVCTestEFCore.Controllers
{
    public class FirstTableController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```

Views folder

- Add folder FirstTable
 - Add Razor View
 - Index.cshtml
 - based on model FirstTable
 - List template



Microsoft.EntityFrameworkCore.Design by Microsoft
Shared design-time components for Entity Framework Core tools.

Add Razor View

View name:

Template:

Model class:

DbContext class:

Options

☐ Create as a partial view

☒ Reference script libraries

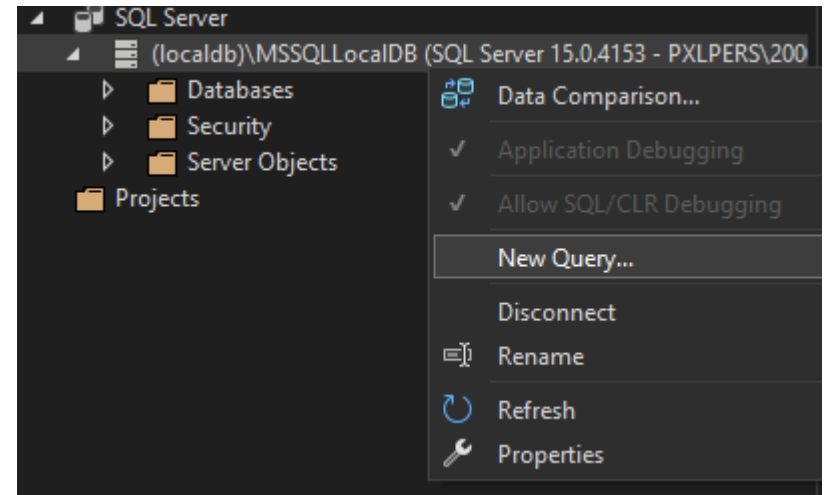
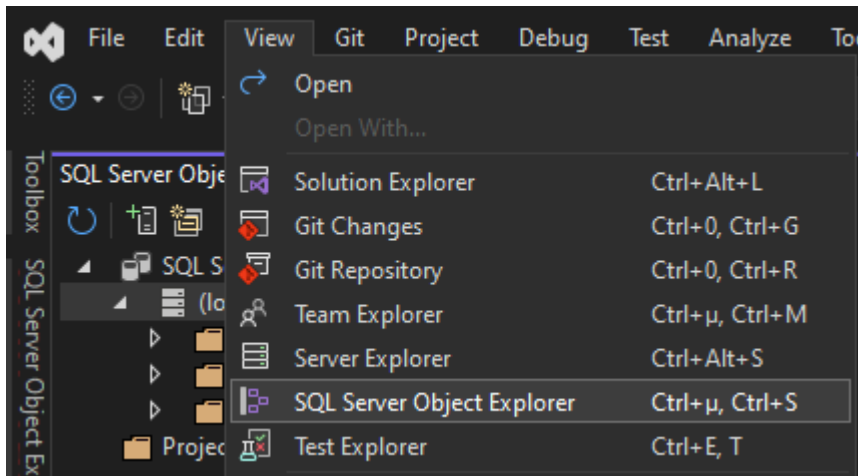
☒ Use a layout page

(Leave empty if it is set in a Razor _viewstart file)

```
@model IEnumerable<MVCTestEFCore.Models.FirstTable>
@{
    ViewData["Title"] = "Index";
}
<h1>Index</h1>
<p>
    <a asp-action="Create">Create New</a>
</p>
<table class="table">
...
```

Database script (SQL Server Object Explorer)

- Create database
 - (connection string)



Create Database mvcefcoredb;

Database script

- Create Table
 - Insert record

```
Use mvcefcoredb;
```

```
CREATE TABLE FirstTable (  
    FirstTableId int IDENTITY(1,1) PRIMARY KEY,  
    TextColumn varchar(15) NOT NULL);
```

```
Insert into FirstTable (TextColumn)  
values ('TestColumn1');
```

```
Insert into FirstTable (TextColumn)  
values ('TestColumn2');
```

```
Select * from FirstTable;
```

DbContext

- Table FirstTable

```
using Microsoft.EntityFrameworkCore;
namespace MVCTestEFCore.Data
{
    public class AppDbContext : DbContext
    {
        public AppDbContext(DbContextOptions<AppDbContext> options) :
            base(options)
        {
        }

        public DbSet<FirstTable>? FirstTable { get; set; }
    }
}
```


Controllers folder

- FirstTableController
 - Dependency Injection

```
namespace MVCTestEFCore.Controllers
{
    public class FirstTableController : Controller
    {
        AppDbContext _context;
        public FirstTableController(AppDbContext context)
        {
            _context = context;
        }

        public IActionResult Index()
        {
            return View();
        }
    }
}
```

Controllers folder

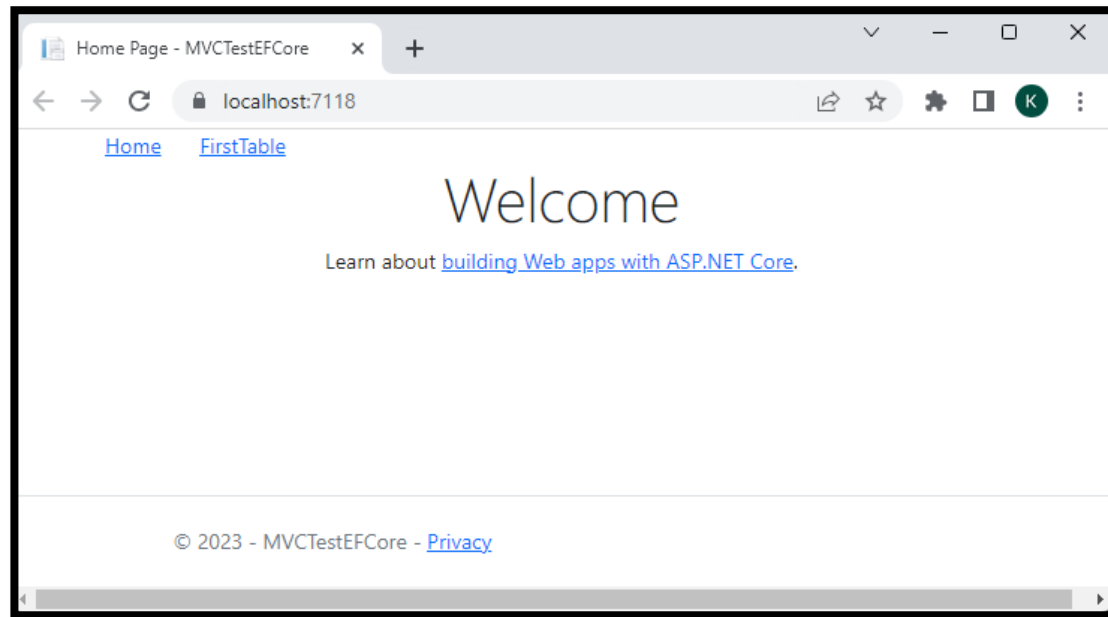
- FirstTableController
 - `Select * from FirstTable`

```
namespace MVCTestEFCore.Controllers
{
    public class FirstTableController : Controller
    {
        AppDbContext _context;
        public FirstTableController(AppDbContext context)
        {
            _context = context;
        }

        public IActionResult Index()
        {
            return View(_context.FirstTable);
        }
    }
}
```

Layout

- Views/Shared/_Layout.cshtml



```
<header>
  <div class="row">
    <div class="col-1"></div>
    <div class="col-1">
      <a asp-controller="Home" asp-action="Index">Home</a>
    </div>
    <div class="col-1">
      <a asp-controller="FirstTable" asp-action="Index">FirstTable</a>
    </div>
  </div>
</header>
```