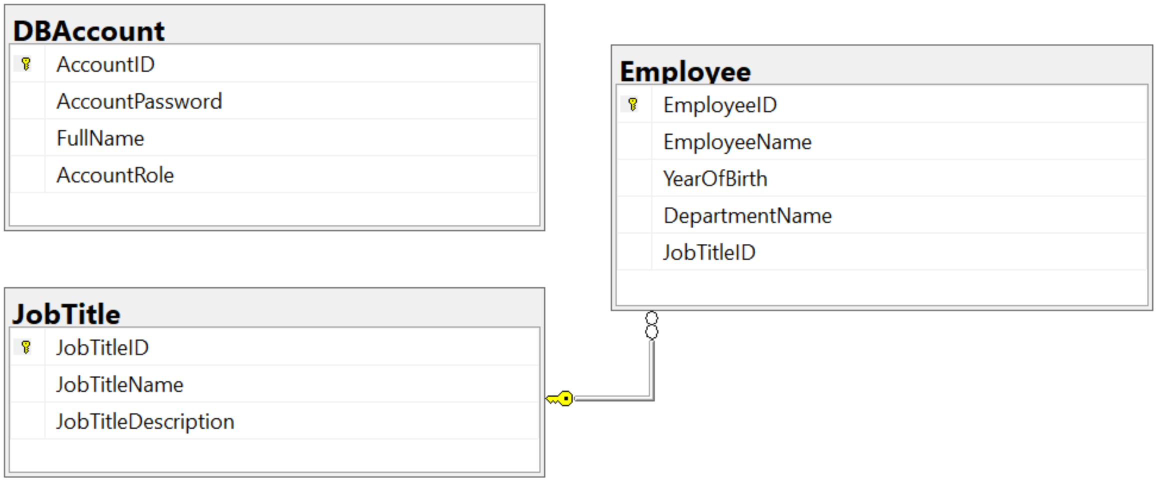
Bài 01. Windows Forms + EntityFrameworkCore or ADO.NET Core

Thiết kế form và thao tác với CSDL dùng EntityFrameworkCore hoặc ADO.NET – Không kết nối trực tiếp, kết nối thông qua Repositoty và ADO.NET.

1. You must **use Windows Forms application using 3-Layer architecture.**
2. Create your MS SQL database named **EmployeeJobTitle** by running code in script **EmployeeJobTitle.sql.**
3. Set the default form for your project as **Login** form.

Create an application using Windows Forms with .NET Core, C#, and ADO.NET (or Entity Framework Core). A MS SQL Server database will be created to persist the data and it will be used for reading and managing data.

EmployeeJobTitle management database



Account role: Administrator = 1; Manager = 2; Staff = 3

1. Design Login form includes UI controls for login function.
2. Design Management form, this form includes UI controls for CRUD actions with employee information. Note: The JobTitleID/Name will come from the JobTitle table. Design a form which allows you to view the list of records, create a new item, update the existing item, and delete a specific record.
3. Authentication function

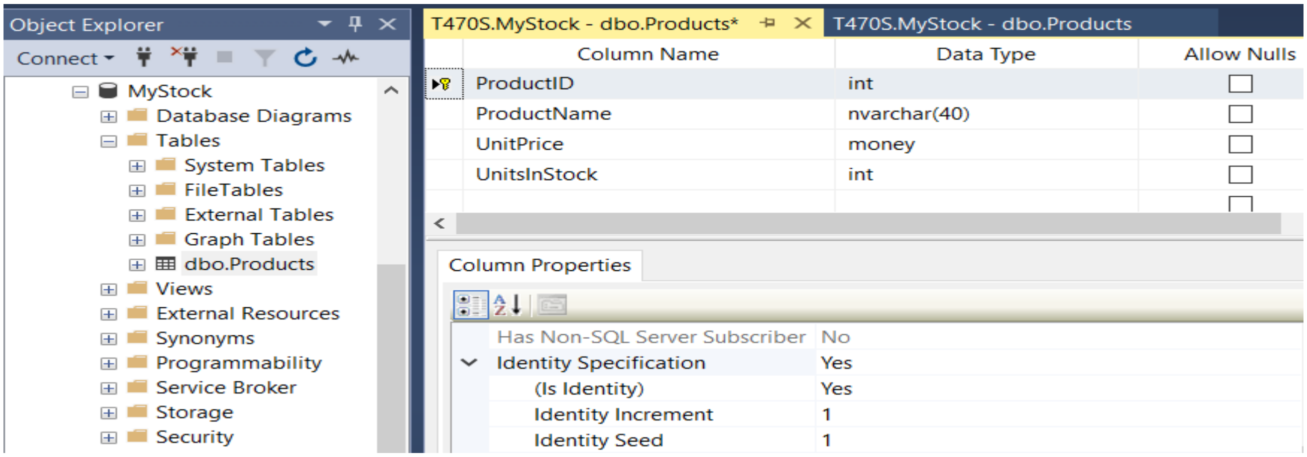
If user with a *administrator role* logs in successfully, save this information to a temporary parameter. All CRUD actions are required with authentication. In the case login unsuccessfully, display *“You are not allowed to access this function!”*.

4. Check if login successfully, list all employees in **Employee** table.
5. Check if login successfully, search employees by Department Name.
6. Check if login successfully, delete the selected item with the confirmation then update the list of employees.
7. Check if login successfully, add new item with the r*equirements:*
    - The JobTitleID/Name will come from the *JobTitle* table (you can use ComboBox UI control in this case)
    - All fields are required.
    - Value for Year of birth from 1960 - 2002.
    - Value for Employee Name is greater than 10 characters. Each word of the Employee Name must begin with the capital letter.
8. Check if login successfully, update an existing item.
    - You can use your different form or use the data grid for this update function. With update function, the *same validation requirements with Add* function.

Bài 02. ASP.NET Core Web App MVC + EntityFrameworkCore

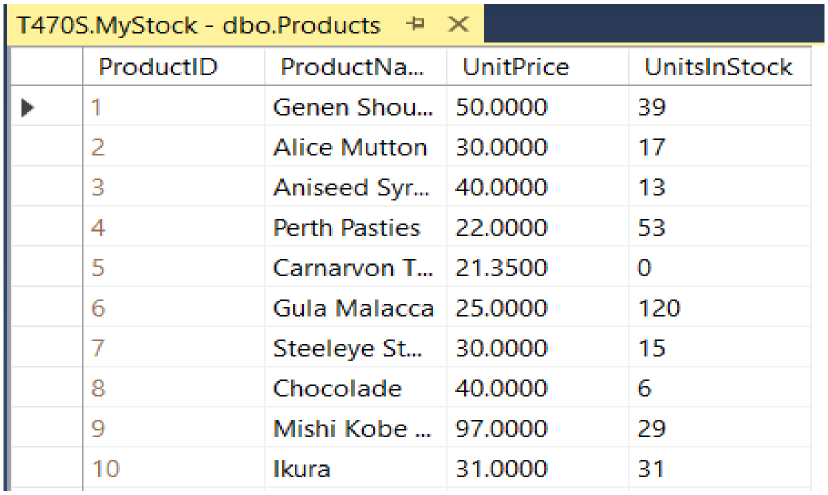
(Kết nối trực tiếp với CSDL. Sau khi hoàn tất có thể sử dụng kết nối CSDL qua Repository, DAO)

Create a sample database named **MyStock** for demonstrations



Column Name	Data Type	Allow Nulls
ProductID	int	☐
ProductName	nvarchar(40)	☐
UnitPrice	money	☐
UnitsInStock	int	☐

Has Non-SQL Server Subscriber: No
Identity Specification: Yes
(Is Identity): Yes
Identity Increment: 1
Identity Seed: 1



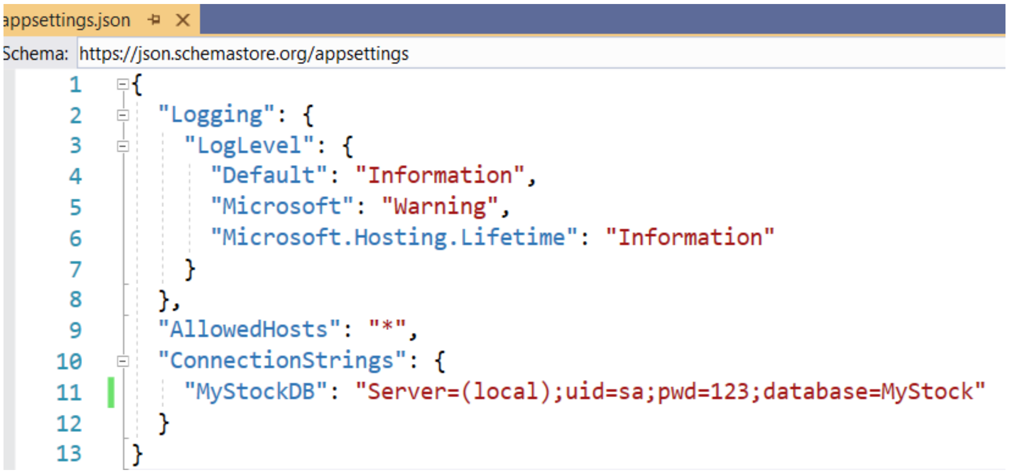
ProductID	ProductNa...	UnitPrice	UnitsInStock
1	Genen Shou...	50.0000	39
2	Alice Mutton	30.0000	17
3	Aniseed Syr...	40.0000	13
4	Perth Pasties	22.0000	53
5	Carnarvon T...	21.3500	0
6	Gula Malacca	25.0000	120
7	Steeleye St...	30.0000	15
8	Chocolade	40.0000	6
9	Mishi Kobe ...	97.0000	29
10	Ikura	31.0000	31

1.Create a ASP.NET Core MVC application named **MyStockApp**

2. Install the following packages from Nuget:

```
dotnet add package Microsoft.EntityFrameworkCore.SqlServer  --version 5.0.12
dotnet add package Microsoft.EntityFrameworkCore.Design --version 5.0.12
```

3.Open **appsettings.json ,** click **Add** and add connection string as follows:



Schema: https://json.schemastore.org/appsettings

```json
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft": "Warning",
      "Microsoft.Hosting.Lifetime": "Information"
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "MyStockDB": "Server=(local);uid=sa;pwd=123;database=MyStock"
  }
}
```

4.Right-click on the project, select **Open in Terminal.** On **Developer PowerShell** dialog**,** execute the following commands to generate model (Copy and Paste the below command):

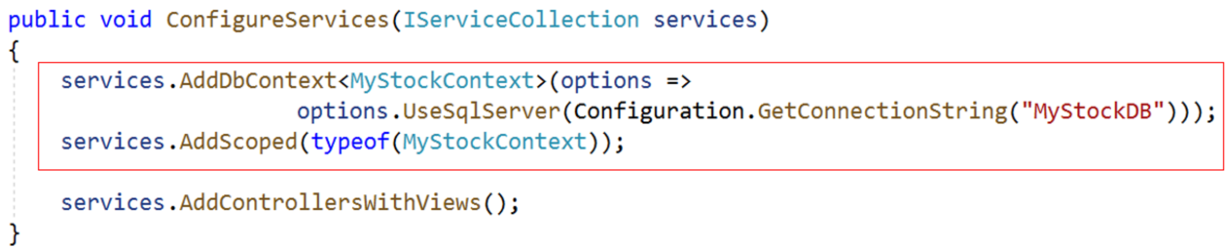
dotnet ef dbcontext scaffold Name=ConnectionStrings:MyStockDB Microsoft.EntityFrameworkCore.SqlServer --output-dir ***Models***

5.Open **Startup.cs** and write codes for **ConfigureService** method as follows:

//add two namespaces

using StockWebApp.Models;

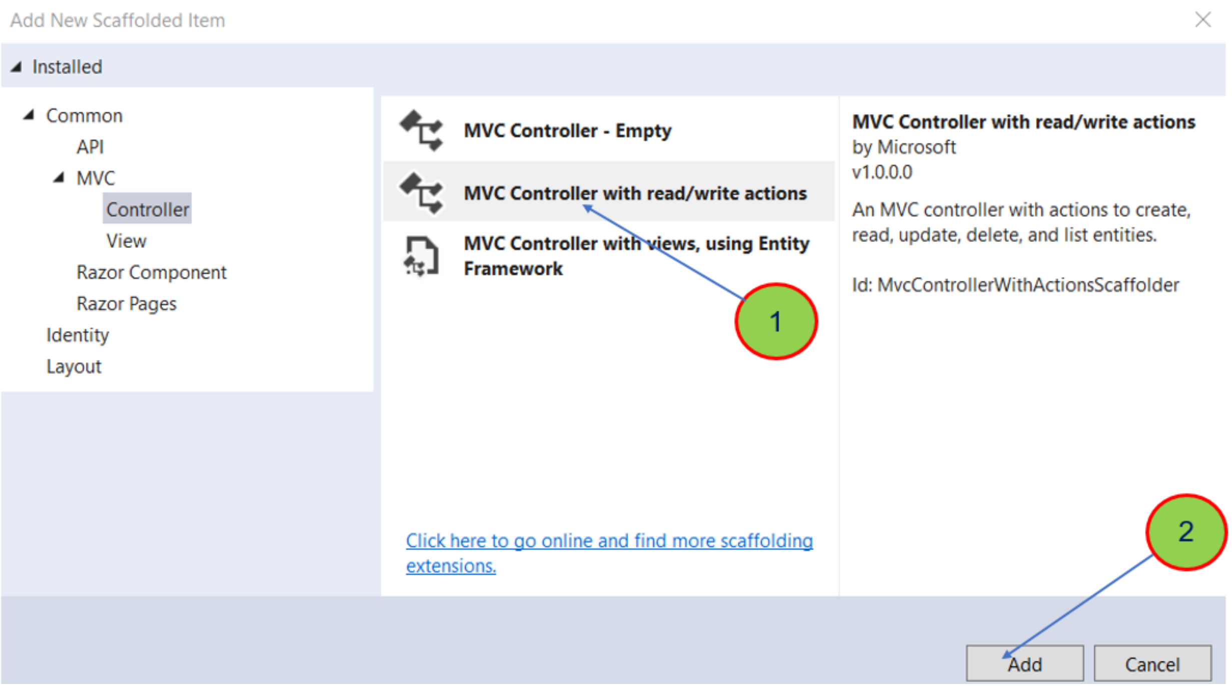
using Microsoft.EntityFrameworkCore;



```csharp
public void ConfigureServices(IServiceCollection services)
{
    services.AddDbContext<MyStockContext>(options =>
                      options.UseSqlServer(Configuration.GetConnectionString("MyStockDB")));
    services.AddScoped(typeof(MyStockContext));

    services.AddControllersWithViews();
}
```

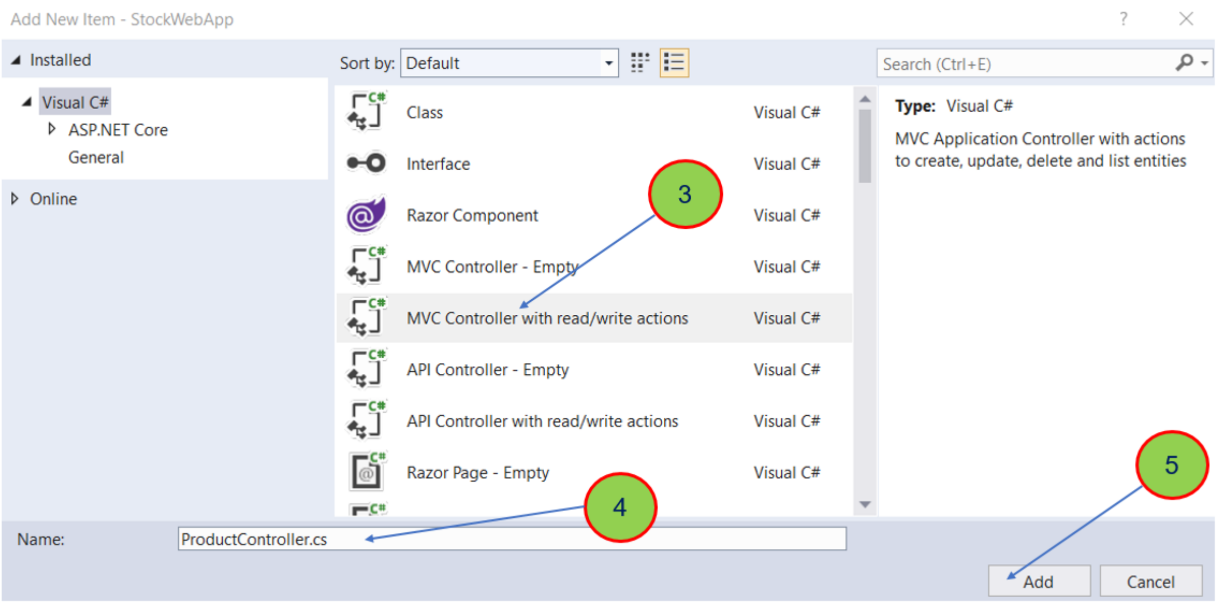
6.Right-click on **Controller** folder | Add | Controller then setup as follows:


Add New Scaffolded Item
Installed
Common
API
MVC
Controller
View
Razor Component
Razor Pages
Identity
Layout
MVC Controller - Empty
MVC Controller with read/write actions
MVC Controller with views, using Entity Framework
Click here to go online and find more scaffolding extensions.
1
2
Add
Cancel

MVC Controller with read/write actions
by Microsoft
v1.0.0.0

An MVC controller with actions to create, read, update, delete, and list entities.

Id: MvcControllerWithActionsScaffolder

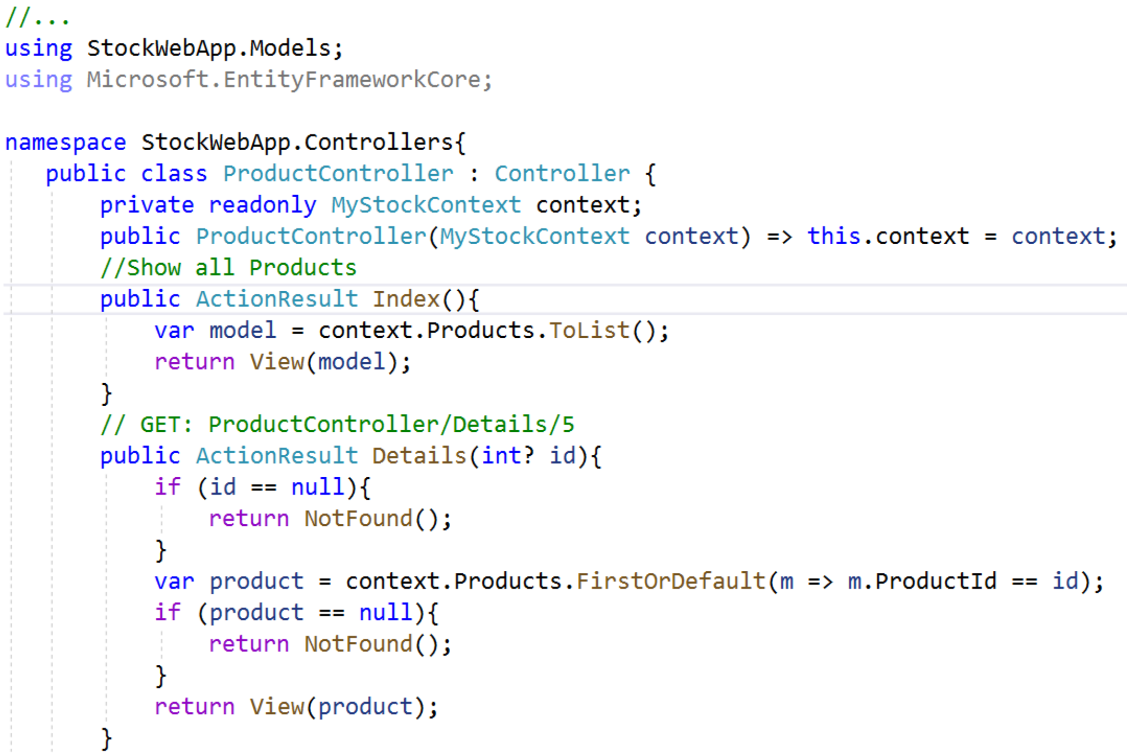


Type: Visual C#

MVC Application Controller with actions to create, update, delete and list entities

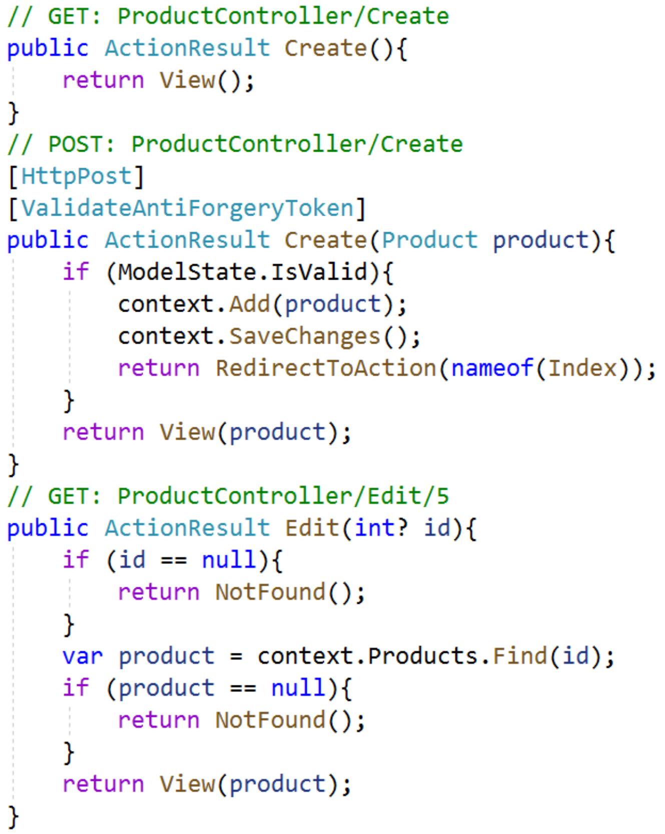
Name: ProductController.cs

7.Write codes for action methods of **ProductController.cs** as follows

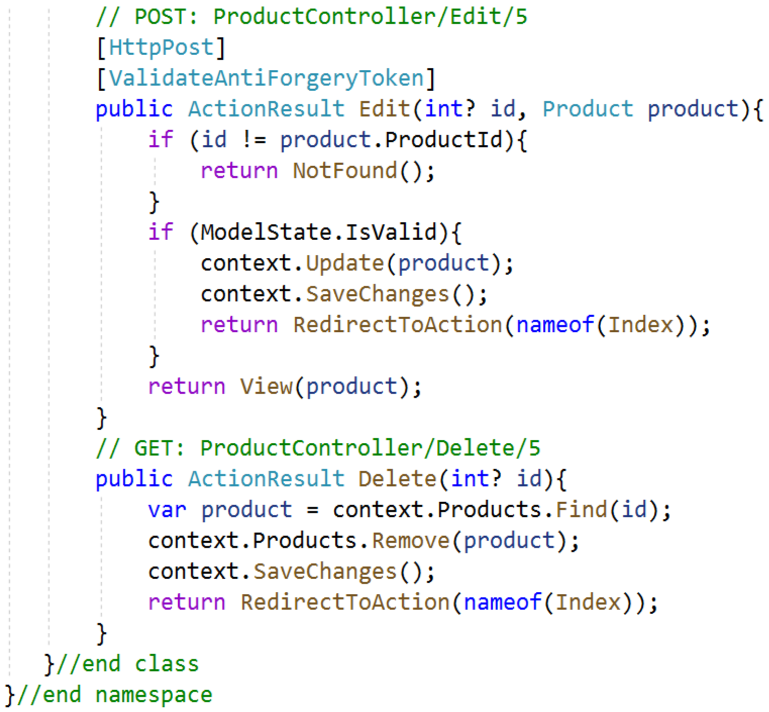


```
//...
using StockWebApp.Models;
using Microsoft.EntityFrameworkCore;

namespace StockWebApp.Controllers{
    public class ProductController : Controller {
        private readonly MyStockContext context;
        public ProductController(MyStockContext context) => this.context = context;
        //Show all Products
        public ActionResult Index(){
            var model = context.Products.ToList();
            return View(model);
        }
        // GET: ProductController/Details/5
        public ActionResult Details(int? id){
            if (id == null){
                return NotFound();
            }
            var product = context.Products.FirstOrDefault(m => m.ProductId == id);
            if (product == null){
                return NotFound();
            }
            return View(product);
        }
```



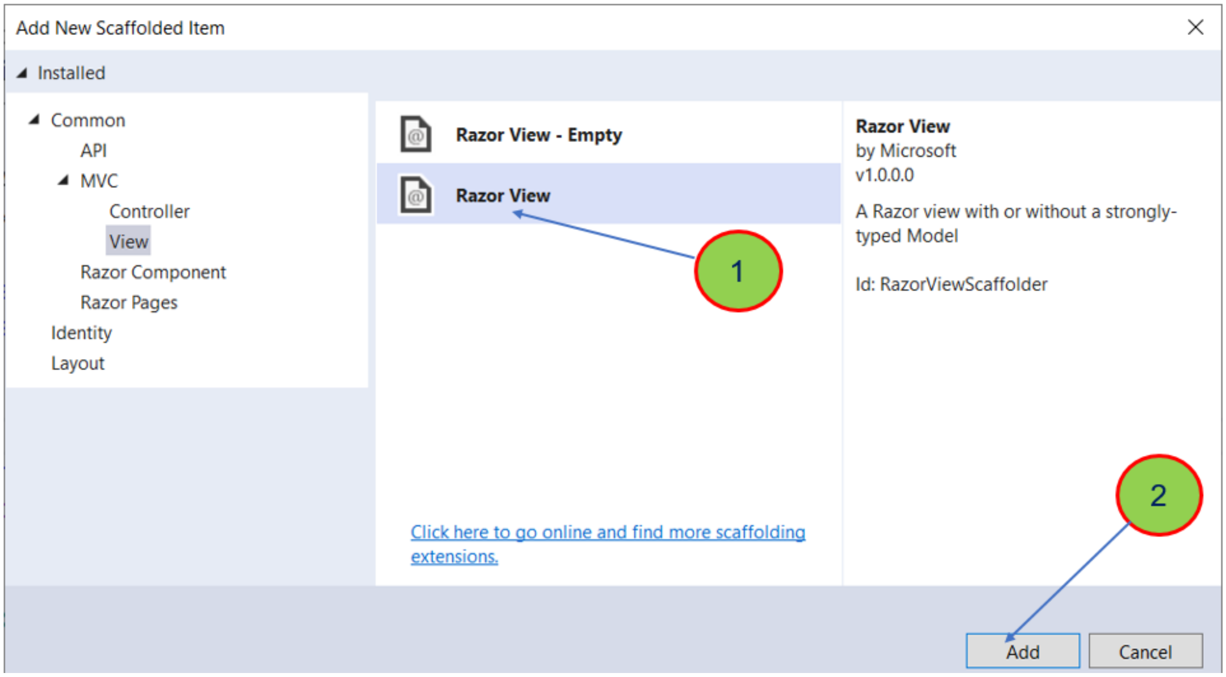
```
        // GET: ProductController/Create
        public ActionResult Create(){
            return View();
        }
        // POST: ProductController/Create
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Create(Product product){
            if (ModelState.IsValid){
                context.Add(product);
                context.SaveChanges();
                return RedirectToAction(nameof(Index));
            }
            return View(product);
        }
        // GET: ProductController/Edit/5
        public ActionResult Edit(int? id){
            if (id == null){
                return NotFound();
            }
            var product = context.Products.Find(id);
            if (product == null){
                return NotFound();
            }
            return View(product);
        }
```



```
        // POST: ProductController/Edit/5
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Edit(int? id, Product product){
            if (id != product.ProductId){
                return NotFound();
            }
            if (ModelState.IsValid){
                context.Update(product);
                context.SaveChanges();
                return RedirectToAction(nameof(Index));
            }
            return View(product);
        }
        // GET: ProductController/Delete/5
        public ActionResult Delete(int? id){
            var product = context.Products.Find(id);
            context.Products.Remove(product);
            context.SaveChanges();
            return RedirectToAction(nameof(Index));
        }
    }//end class
}//end namespace
```

8.Right-click on **View** folder | **Add** | **New Folder** named **Product**

9.Right-click on **Product** folder | **Add** | **View** named **Index** as follows


Add New Scaffolded Item

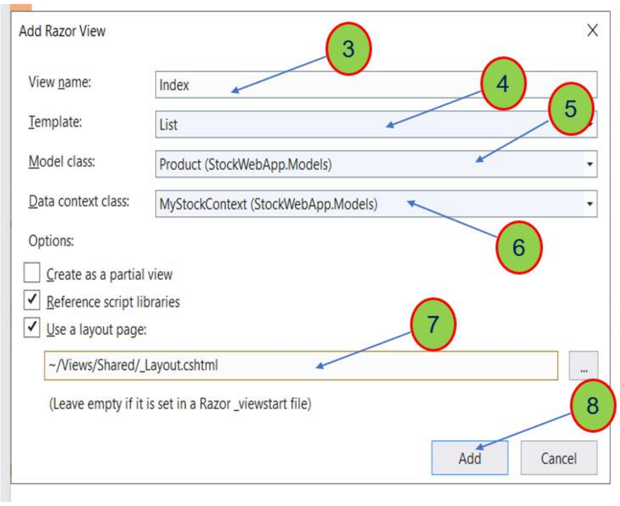
Razor View
by Microsoft
v1.0.0.0

A Razor view with or without a strongly-typed Model

Id: RazorViewScaffolder

Click here to go online and find more scaffolding extensions.

Index view (show product list)



View name: Index

Template: List

Model class: Product (StockWebApp.Models)

Data context class: MyStockContext (StockWebApp.Models)

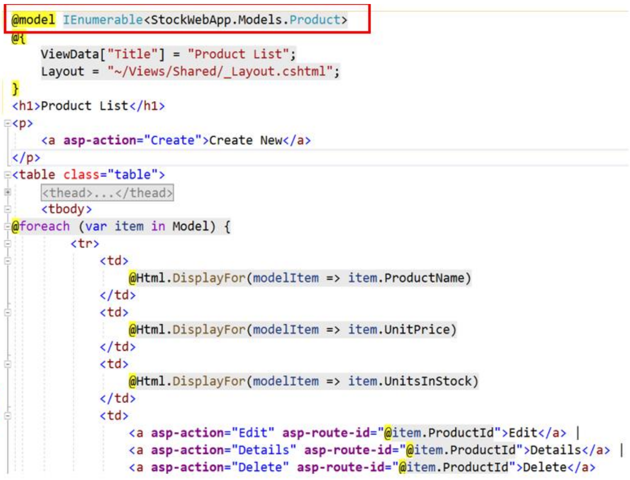
□ Create as a partial view

☑ Reference script libraries

☑ Use a layout page:

~/Views/Shared/_Layout.cshtml

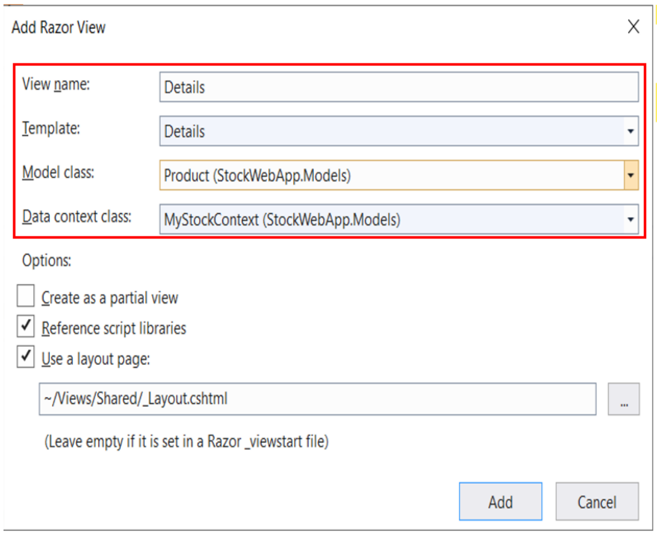
(Leave empty if it is set in a Razor _viewstart file)



```
@model IEnumerable<StockWebApp.Models.Product>
@{
    ViewData["Title"] = "Product List";
    Layout = "~/Views/Shared/_Layout.cshtml";
}
<h1>Product List</h1>
<p>
    <a asp-action="Create">Create New</a>
</p>
<table class="table">
    <thead>...</thead>
    <tbody>
@foreach (var item in Model) {
        <tr>
            <td>
                @Html.DisplayFor(modelItem => item.ProductName)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.UnitPrice)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.UnitsInStock)
            </td>
            <td>
                <a asp-action="Edit" asp-route-id="@item.ProductId">Edit</a> |
                <a asp-action="Details" asp-route-id="@item.ProductId">Details</a> |
                <a asp-action="Delete" asp-route-id="@item.ProductId">Delete</a>
```

Repeat this step to add views: **Details**, **Create** and **Edit** as the next figures

Details view



View name: Details

Template: Details

Model class: Product (StockWebApp.Models)

Data context class: MyStockContext (StockWebApp.Models)

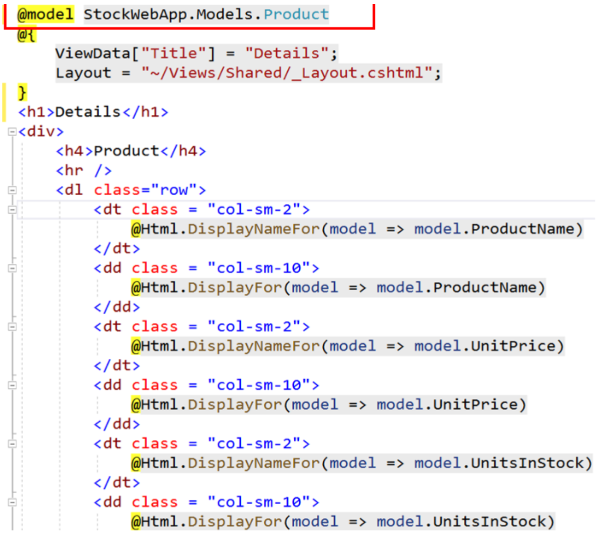
□ Create as a partial view

☑ Reference script libraries

☑ Use a layout page:

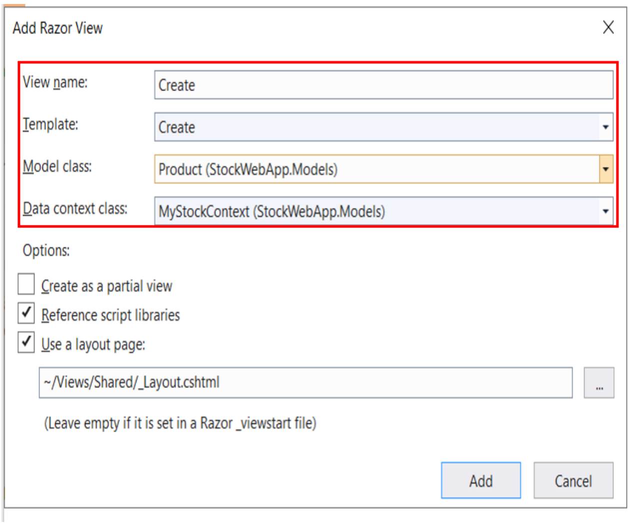
~/Views/Shared/_Layout.cshtml

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



```
@model StockWebApp.Models.Product
@{
    ViewData["Title"] = "Details";
    Layout = "~/Views/Shared/_Layout.cshtml";
}
<h1>Details</h1>
<div>
    <h4>Product</h4>
    <hr />
    <dl class="row">
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.ProductName)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.ProductName)
        </dd>
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.UnitPrice)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.UnitPrice)
        </dd>
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.UnitsInStock)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.UnitsInStock)
```

Create view



View name: Create

Template: Create

Model class: Product (StockWebApp.Models)

Data context class: MyStockContext (StockWebApp.Models)

Options:

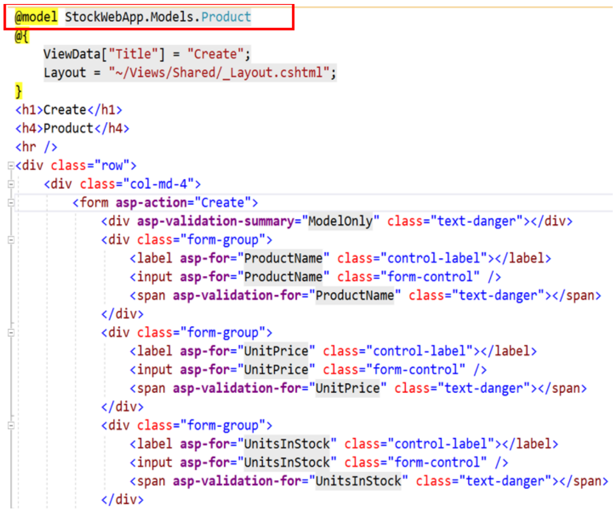
☐ Create as a partial view

☑ Reference script libraries

☑ Use a layout page:

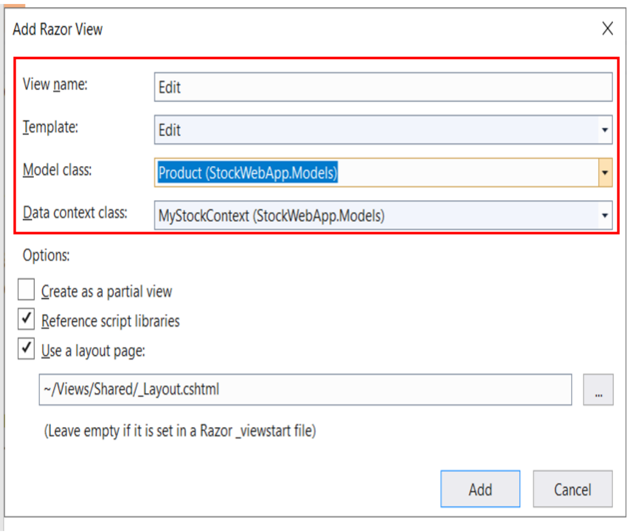
~/Views/Shared/_Layout.cshtml

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



```
@model StockWebApp.Models.Product
@{
    ViewData["Title"] = "Create";
    Layout = "~/Views/Shared/_Layout.cshtml";
}
<h1>Create</h1>
<h4>Product</h4>
<hr />
<div class="row">
    <div class="col-md-4">
        <form asp-action="Create">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <div class="form-group">
                <label asp-for="ProductName" class="control-label"></label>
                <input asp-for="ProductName" class="form-control" />
                <span asp-validation-for="ProductName" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="UnitPrice" class="control-label"></label>
                <input asp-for="UnitPrice" class="form-control" />
                <span asp-validation-for="UnitPrice" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="UnitsInStock" class="control-label"></label>
                <input asp-for="UnitsInStock" class="form-control" />
                <span asp-validation-for="UnitsInStock" class="text-danger"></span>
            </div>
```

Edit view



View name: Edit

Template: Edit

Model class: Product (StockWebApp.Models)

Data context class: MyStockContext (StockWebApp.Models)

Options:

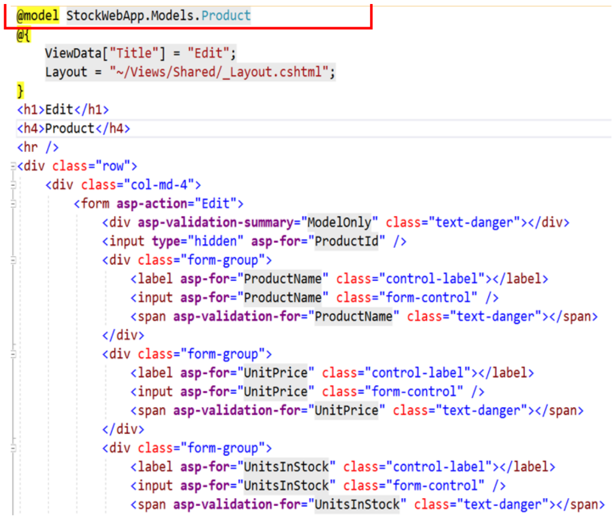
☐ Create as a partial view

☑ Reference script libraries

☑ Use a layout page:

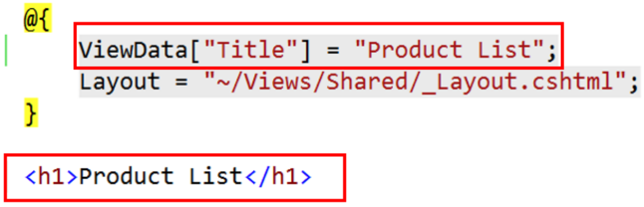
~/Views/Shared/_Layout.cshtml

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



```
@model StockWebApp.Models.Product
@{
    ViewData["Title"] = "Edit";
    Layout = "~/Views/Shared/_Layout.cshtml";
}
<h1>Edit</h1>
<h4>Product</h4>
<hr />
<div class="row">
    <div class="col-md-4">
        <form asp-action="Edit">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <input type="hidden" asp-for="ProductId" />
            <div class="form-group">
                <label asp-for="ProductName" class="control-label"></label>
                <input asp-for="ProductName" class="form-control" />
                <span asp-validation-for="ProductName" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="UnitPrice" class="control-label"></label>
                <input asp-for="UnitPrice" class="form-control" />
                <span asp-validation-for="UnitPrice" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="UnitsInStock" class="control-label"></label>
                <input asp-for="UnitsInStock" class="form-control" />
                <span asp-validation-for="UnitsInStock" class="text-danger"></span>
```

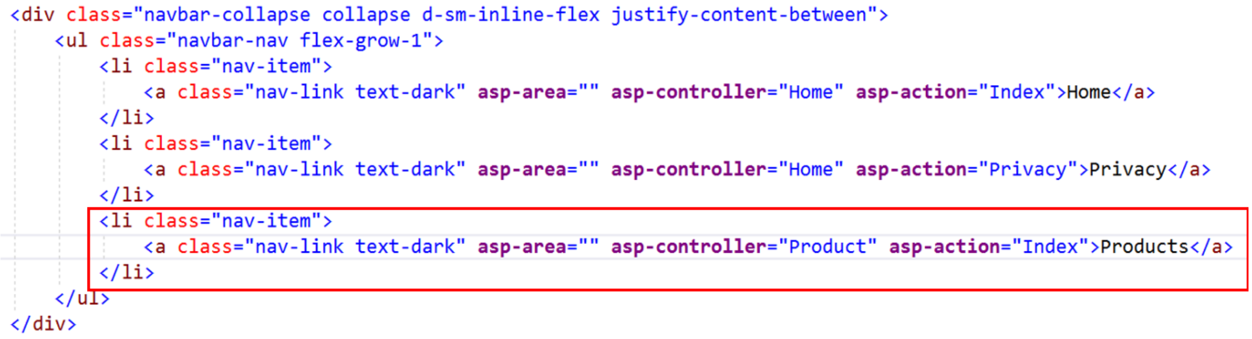
10.Open **index.cshtml** view of **Product** folder and update contents as follows



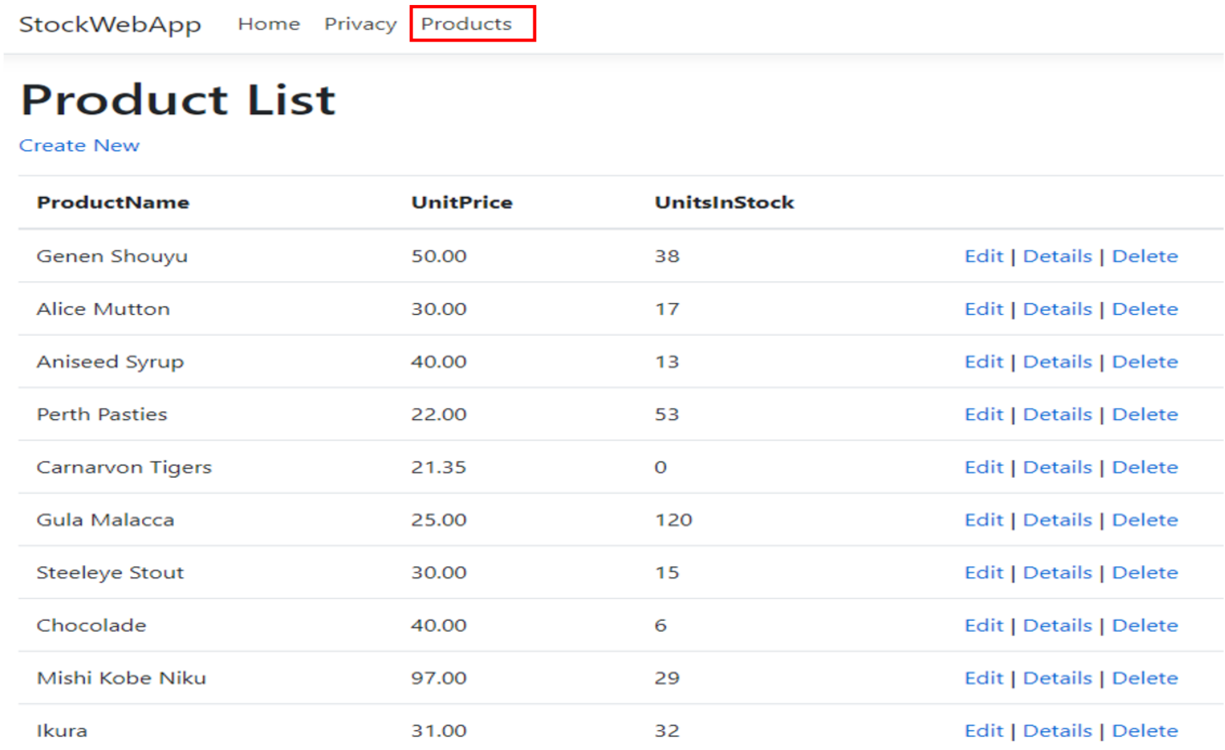
```
@{
    ViewData["Title"] = "Product List";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h1>Product List</h1>
```

11.Open **_Layout.cshtml** view in the View | Shared folder, add tags to navigate to **Index** view of **Product** controller as follows then run project:



```
<div class="navbar-collapse collapse d-sm-inline-flex justify-content-between">
    <ul class="navbar-nav flex-grow-1">
        <li class="nav-item">
            <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Index">Home</a>
        </li>
        <li class="nav-item">
            <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Privacy">Privacy</a>
        </li>
        <li class="nav-item">
            <a class="nav-link text-dark" asp-area="" asp-controller="Product" asp-action="Index">Products</a>
        </li>
    </ul>
</div>
```



Product List

Create New

ProductName	UnitPrice	UnitsInStock	
Genen Shouyu	50.00	38	Edit | Details | Delete
Alice Mutton	30.00	17	Edit | Details | Delete
Aniseed Syrup	40.00	13	Edit | Details | Delete
Perth Pasties	22.00	53	Edit | Details | Delete
Carnarvon Tigers	21.35	0	Edit | Details | Delete
Gula Malacca	25.00	120	Edit | Details | Delete
Steeleye Stout	30.00	15	Edit | Details | Delete
Chocolade	40.00	6	Edit | Details | Delete
Mishi Kobe Niku	97.00	29	Edit | Details | Delete
Ikura	31.00	32	Edit | Details | Delete