Table of Contents

Demo: Shop MVC 2, HomeView, shared _layout, navigation

Author: Baifan Zhou

Some explanations are created with the help of ChatGPT.

Introduction

This demo walks you through the creation and setup of a simple MVC web application using ASP.NET Core. The project demonstrates how to build views, controllers, and shared layouts, as well as how to configure the application for a smooth development experience. By the end of this demo, you'll have a functional MVC web application with a clean and organised structure.

HomeView and HomeController

In this section, you will create a HomeView and a HomeController to handle requests to the home page of your application. You'll learn how to:

- Create a subfolder Home under the Views folder and add an Index.cshtml file.
- Define the HomeController class to handle incoming HTTP requests and return the appropriate views.
- Understand how controllers and views work together in the MVC pattern, handling requests, executing actions, and rendering views.

Create Shared _Layout

Here, you'll set up a shared layout for your application:

- Create a subfolder Shared under the Views folder.
- Add a _Layout.cshtml file to define the common HTML structure for your pages, including headers, footers, and navigation.
- Use the @RenderBody() placeholder to inject content from individual views into the layout.
- Copy and modify layout and CSS files to maintain a consistent look and feel across your application.

Simplify Table.cshtml

This section guides you in simplifying the Table.cshtml view:

- Remove unnecessary headers and other elements to streamline the view.

Reset default launchURL

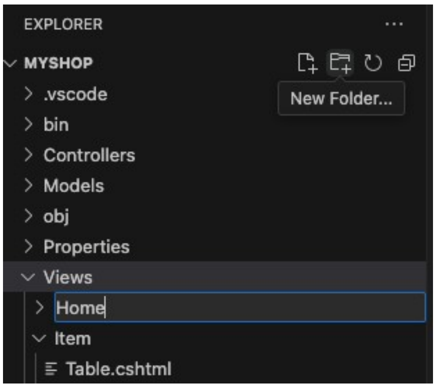
Learn how to reset the default launch URL for your application:

- Modify the .vscode/launch.json file to change the launch URI, ensuring the application starts on the desired page.

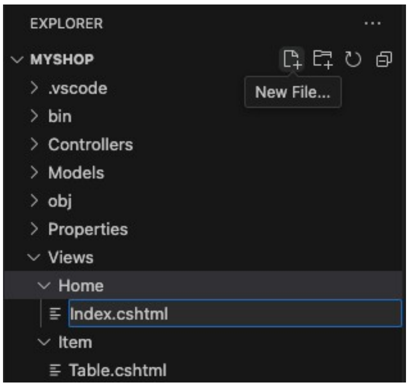
By following this demo, you will gain a solid understanding of setting up and configuring an ASP.NET Core MVC application, including creating views and controllers, setting up shared layouts, and simplifying view templates. Happy coding!

HomeView and HomeController

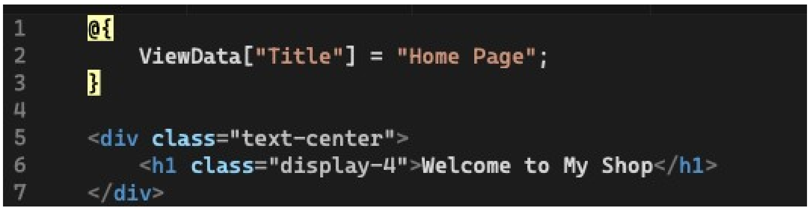
Create a subfolder “Home” under the folder Views



Create a file Index.cshtml under the Home folder

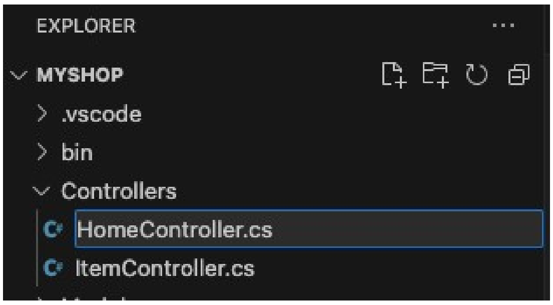


Change the Index.cshtml to this:

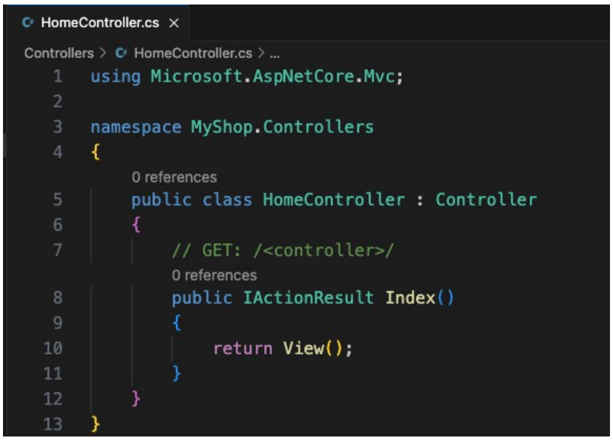


```
@{
    ViewData["Title"] = "Home Page";
}

<div class="text-center">
    <h1 class="display-4">Welcome to My Shop</h1>
</div>
```

Create a new Controller class “HomeController.cs” under the folder Controller



Change the file to:



```
using Microsoft.AspNetCore.Mvc;

namespace MyShop.Controllers
{
    public class HomeController : Controller
    {
        // GET: /<controller>/
        public IActionResult Index()
        {
            return View();
        }
    }
}
```

Detailed explanation:

MVC Architecture

The MVC pattern is a design pattern used to separate concerns within an application. It divides an application into three main components:

1. **Model**: Represents the data and the business logic.
2. **View**: Represents the presentation layer (UI).
3. **Controller**: Handles user input and interacts with the Model to render the appropriate View.

Controllers

A **Controller** is a class that handles incoming HTTP requests, processes them, and returns the appropriate response. Controllers are the heart of the MVC pattern, acting as the intermediary between the Model and the View.

- **Naming Convention**: Controllers typically end with the suffix "Controller". For example, HomeController, ItemController.
- **Location**: They are usually stored in the Controllers folder.

Actions

An **Action** is a method within a Controller that handles a specific request. Each Action method typically corresponds to a user interaction, such as viewing a list of items, displaying details of an item, or submitting a form.

- **Return Types**: Action methods usually return an IActionResult, which could be a View, JSON, a Redirect, or other result types.

HomeController.cs

The HomeController in this ASP.NET Core MVC application is responsible for handling requests to the home page of the website.

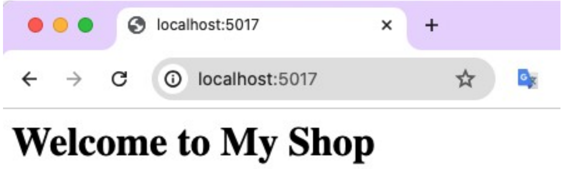
- **Controller Definition**: The HomeController class inherits from Controller, which is a base class in ASP.NET Core MVC that provides methods and properties for handling HTTP requests and returning responses.
- **Index Action Method**: The Index method is an action method. It's mapped to handle HTTP GET requests directed at the root URL (e.g., /Home/Index or / if routed as the default controller).

- **Returning a View**: When the Index action method is called, it returns a View result. This instructs the MVC framework to render a view named Index located in the Views/Home folder.

How the Controller and View Work Together

1. **Request Handling**: When a user navigates to the root URL or /Home/Index, the routing system directs the request to the Index action method of HomeController.
2. **Action Execution**: The Index action method executes and prepares any necessary data or simply decides which view to render. In this case, it just returns a View() without any model data.
3. **View Selection**: The MVC framework looks for a view file named Index.cshtml in the Views/Home directory.
4. **View Rendering**: The Index.cshtml view is rendered into HTML and sent back as the HTTP response to the user's browser.

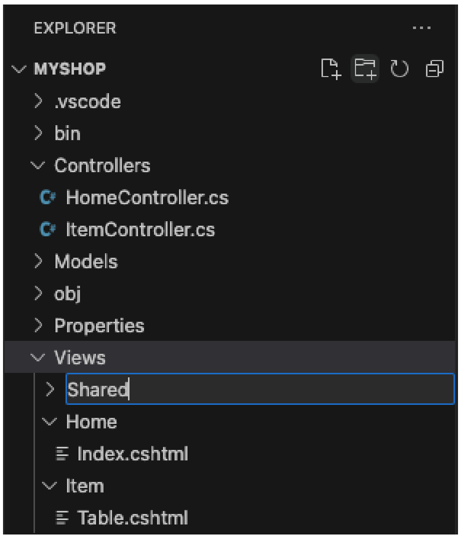
Now if you press F5, or click Menu Run → Start Debugging, and go to the home address, you will see a simple home view. In the next, we are going to create navigation.



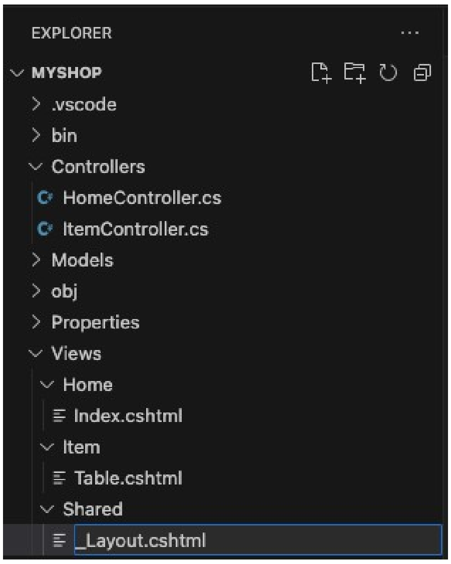
Welcome to My Shop

Create Shared _Layout

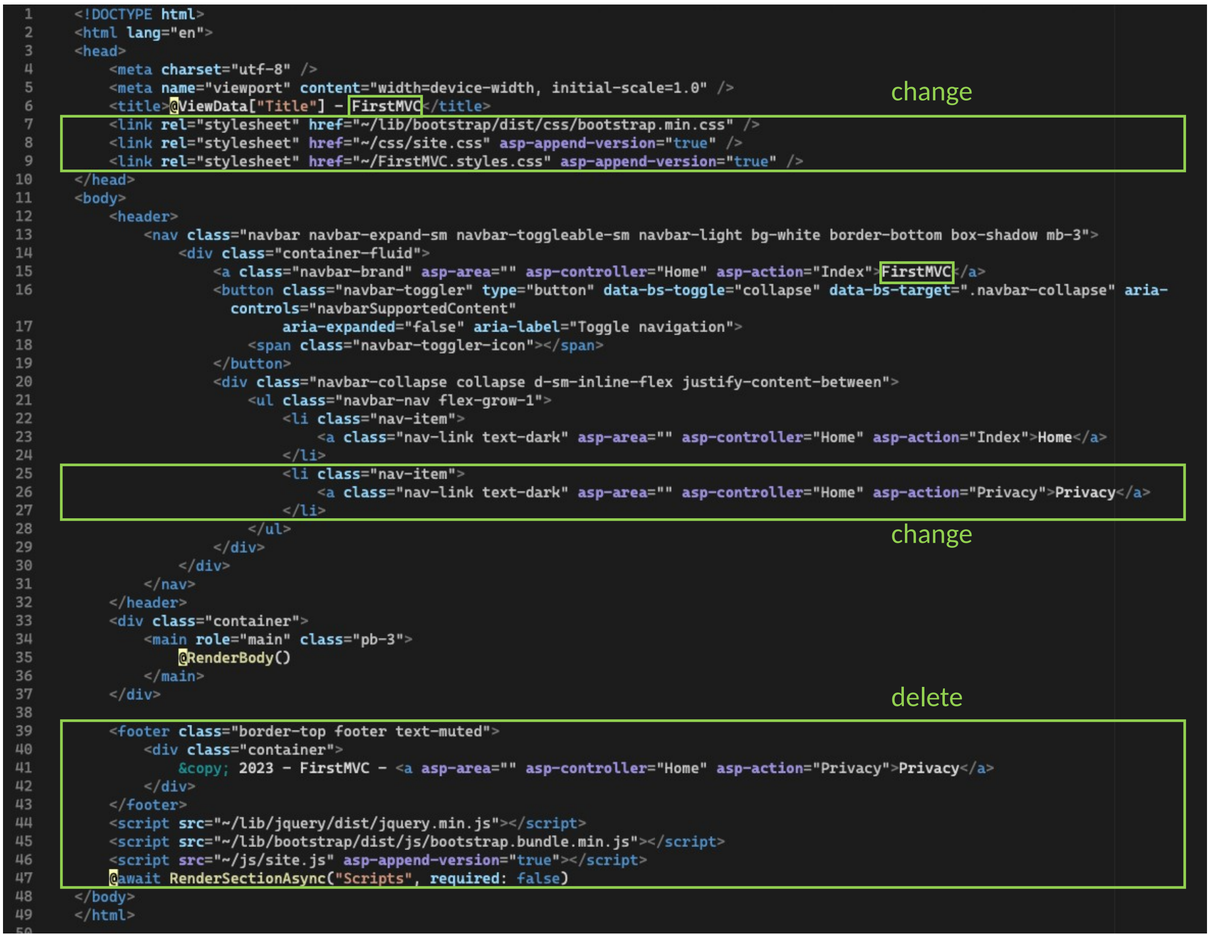
Create a subfolder "Shared" under the folder Views.



Create a file Razor Layout with the name "_Layout.cshtml" under the Shared folder



Copy the code in _Layout.cshtml from the FirstMVC project into the _Layout.cshtml file and Modify the code: change the stylesheet reference, change the nav-item, change the view title, delete the footer part



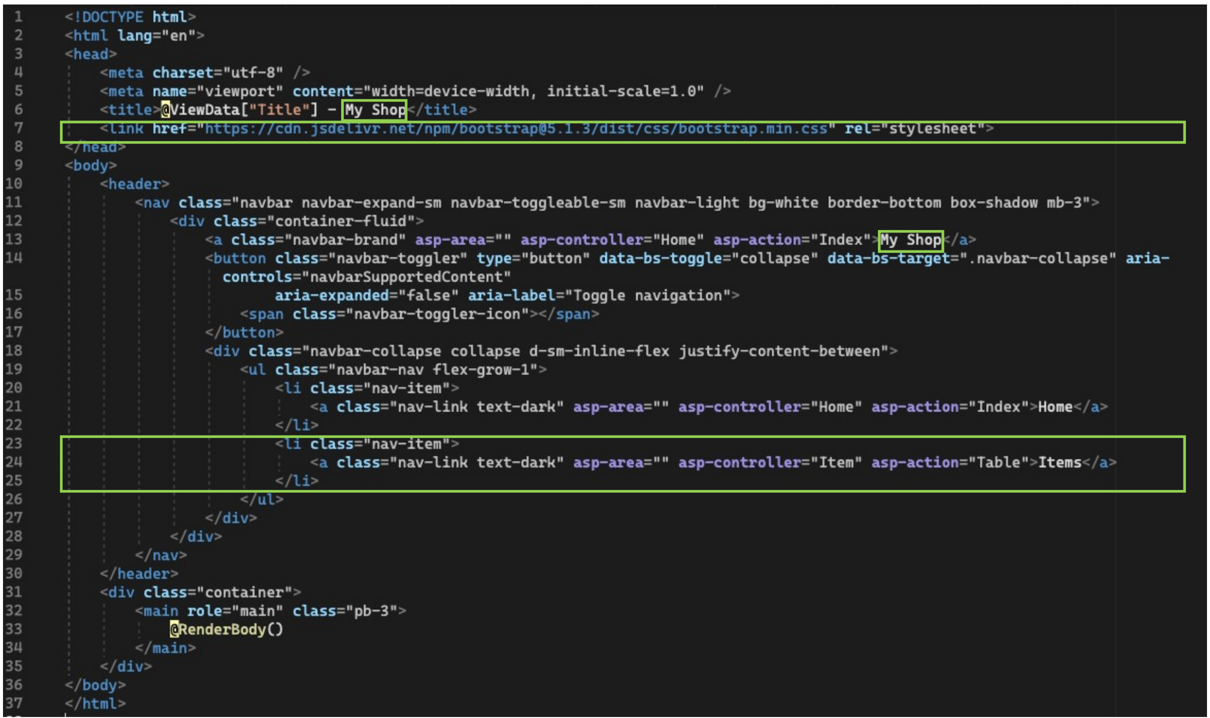
change

change

delete

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>@ViewData["Title"] - FirstMVC</title>
    <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />
    <link rel="stylesheet" href="~/css/site.css" asp-append-version="true" />
    <link rel="stylesheet" href="~/FirstMVC.styles.css" asp-append-version="true" />
</head>
<body>
    <header>
        <nav class="navbar navbar-expand-sm navbar-toggleable-sm navbar-light bg-white border-bottom box-shadow mb-3">
            <div class="container-fluid">
                <a class="navbar-brand" asp-area="" asp-controller="Home" asp-action="Index">FirstMVC</a>
                <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target=".navbar-collapse" aria-controls="navbarSupportedContent"
                        aria-expanded="false" aria-label="Toggle navigation">
                    <span class="navbar-toggler-icon"></span>
                </button>
                <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>
                    </ul>
                </div>
            </div>
        </nav>
    </header>
    <div class="container">
        <main role="main" class="pb-3">
            @RenderBody()
        </main>
    </div>

    <footer class="border-top footer text-muted">
        <div class="container">
            &copy; 2023 - FirstMVC - <a asp-area="" asp-controller="Home" asp-action="Privacy">Privacy</a>
        </div>
    </footer>
    <script src="~/lib/jquery/dist/jquery.min.js"></script>
    <script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>
    <script src="~/js/site.js" asp-append-version="true"></script>
    @await RenderSectionAsync("Scripts", required: false)
</body>
</html>
```

Change to this:

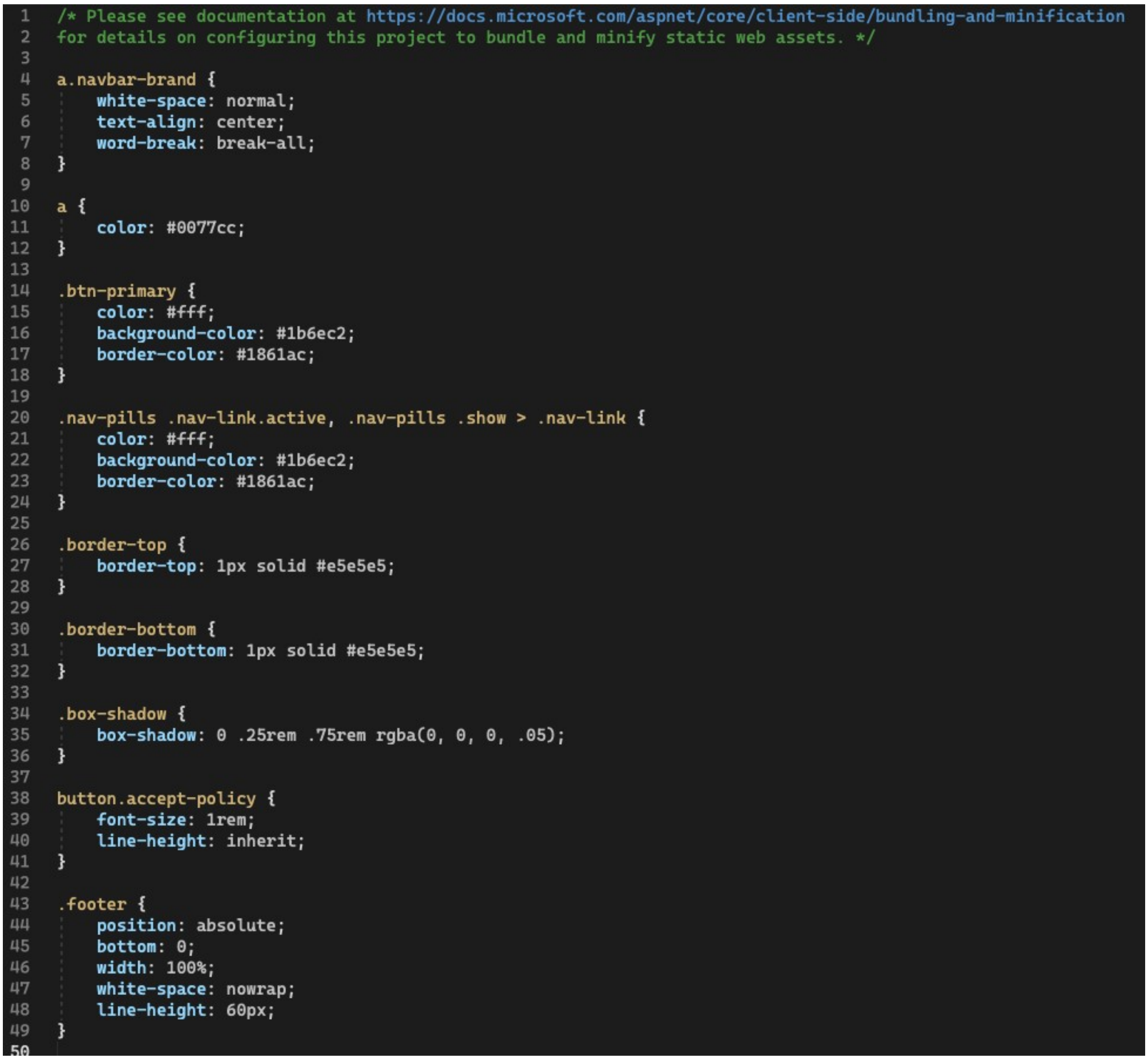


```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>@ViewData["Title"] - My Shop</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet">
</head>
<body>
    <header>
        <nav class="navbar navbar-expand-sm navbar-toggleable-sm navbar-light bg-white border-bottom box-shadow mb-3">
            <div class="container-fluid">
                <a class="navbar-brand" asp-area="" asp-controller="Home" asp-action="Index">My Shop</a>
                <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target=".navbar-collapse" aria-controls="navbarSupportedContent"
                        aria-expanded="false" aria-label="Toggle navigation">
                    <span class="navbar-toggler-icon"></span>
                </button>
                <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="Item" asp-action="Table">Items</a>
                        </li>
                    </ul>
                </div>
            </div>
        </nav>
    </header>
    <div class="container">
        <main role="main" class="pb-3">
            @RenderBody()
        </main>
    </div>
</body>
</html>
```

Note: the routing and navigation follows the default pattern “Controller/Action” as specified as the default routing pattern in the Program.cs.

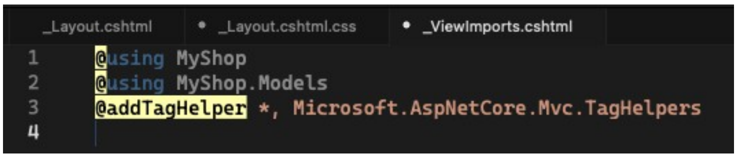
@RenderBody() is a placeholder where the content of the individual views will be injected.

Copy the file _Layout.cshtml.css from the FirstMVC project into the “Shared” folder



```css
/* Please see documentation at https://docs.microsoft.com/aspnet/core/client-side/bundling-and-minification
for details on configuring this project to bundle and minify static web assets. */

a.navbar-brand {
    white-space: normal;
    text-align: center;
    word-break: break-all;
}

a {
    color: #0077cc;
}

.btn-primary {
    color: #fff;
    background-color: #1b6ec2;
    border-color: #1861ac;
}

.nav-pills .nav-link.active, .nav-pills .show > .nav-link {
    color: #fff;
    background-color: #1b6ec2;
    border-color: #1861ac;
}

.border-top {
    border-top: 1px solid #e5e5e5;
}

.border-bottom {
    border-bottom: 1px solid #e5e5e5;
}

.box-shadow {
    box-shadow: 0 .25rem .75rem rgba(0, 0, 0, .05);
}

button.accept-policy {
    font-size: 1rem;
    line-height: inherit;
}

.footer {
    position: absolute;
    bottom: 0;
    width: 100%;
    white-space: nowrap;
    line-height: 60px;
}

```

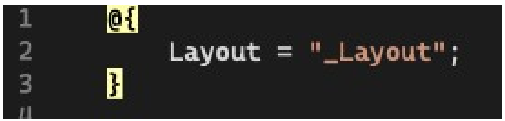
Create file “_ViewImports. cshtml” under the “Views” folder (not under “Shared”)
Add the following code into _ViewImports.cshtml



```
1   @using MyShop
2   @using MyShop.Models
3   @addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers
4
```

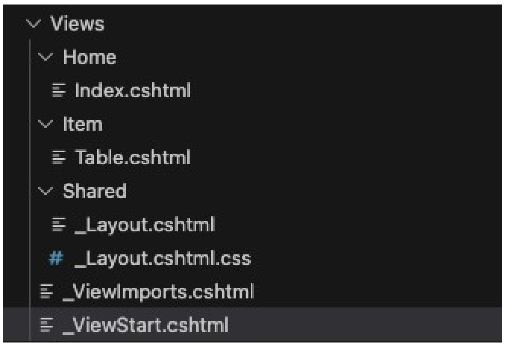
Note: It is special Razor view file allows you to define common statements applied to all other views in the application. It acts as a global import file for your views.
(Line3) is used to add tag helpers, which enables to create custom HTML elements (tags). Here it enables routing and navigation, e.g., the pattern of “Controller/Action”.

Create file Razor View Start under the Views folder
With the following code in the _ViewStart.cshtml (specifying the default layout):
It is a special Razor view file applied to all other views in the application. It acts as a global layout page for your views



```
@{
    Layout = "_Layout";
}
```

Now we have the following files in the Views folder



```
Views
  Home
    Index.cshtml
  Item
    Table.cshtml
  Shared
    _Layout.cshtml
    _Layout.cshtml.css
  _ViewImports.cshtml
  _ViewStart.cshtml
```

Detailed explanations:

Shared Folder in Views

The Shared folder within the Views directory is used to store views and other Razor files that are shared across multiple controllers and views in the application. This helps in avoiding duplication and maintaining a consistent look and feel throughout the application.

_Layout.cshtml

- **Purpose**: _Layout.cshtml is a special Razor file that defines a common layout (template) for your web pages. It typically includes the HTML structure, such as the <head>, <header>, <footer>, and any common scripts or styles.
- **Usage**: By using a layout, you ensure that all your pages share the same structure and design. Individual views will inject their content into the layout's @RenderBody() placeholder.
- **Location**: Views/Shared/_Layout.cshtml.

_ViewImports.cshtml

- **Purpose**: _ViewImports.cshtml is used to define common directives, namespaces, and tag helpers that should be available to all Razor views within the folder hierarchy where it is placed.
- **Usage**: This file helps to avoid repeating common directives across multiple views, promoting DRY (Don't Repeat Yourself) principles.
- **Location**: Typically located at the root of the Views folder but can also be placed in specific subfolders if different imports are needed for those views.
- @using MyShop.Models: Makes the MyShop.Models namespace available to all views, so you don't have to add it individually in each view file.
- @addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers: Registers all the tag helpers provided by ASP.NET Core MVC, enabling the use of tag helpers like <partial>, <environment>, etc.

_ViewStart.cshtml

- **Purpose**: `_ViewStart.cshtml` is used to execute common code before rendering any Razor view. It typically sets the layout for the views.
- **Usage**: This file is executed before every view, allowing you to set properties like the layout dynamically.

- **Location**: Typically located at the root of the `Views` folder.

- Layout = "_Layout"; specifies that all views should use the _Layout.cshtml file in the Views/Shared folder as their layout. This can be overridden in individual views if needed.

How They Work Together

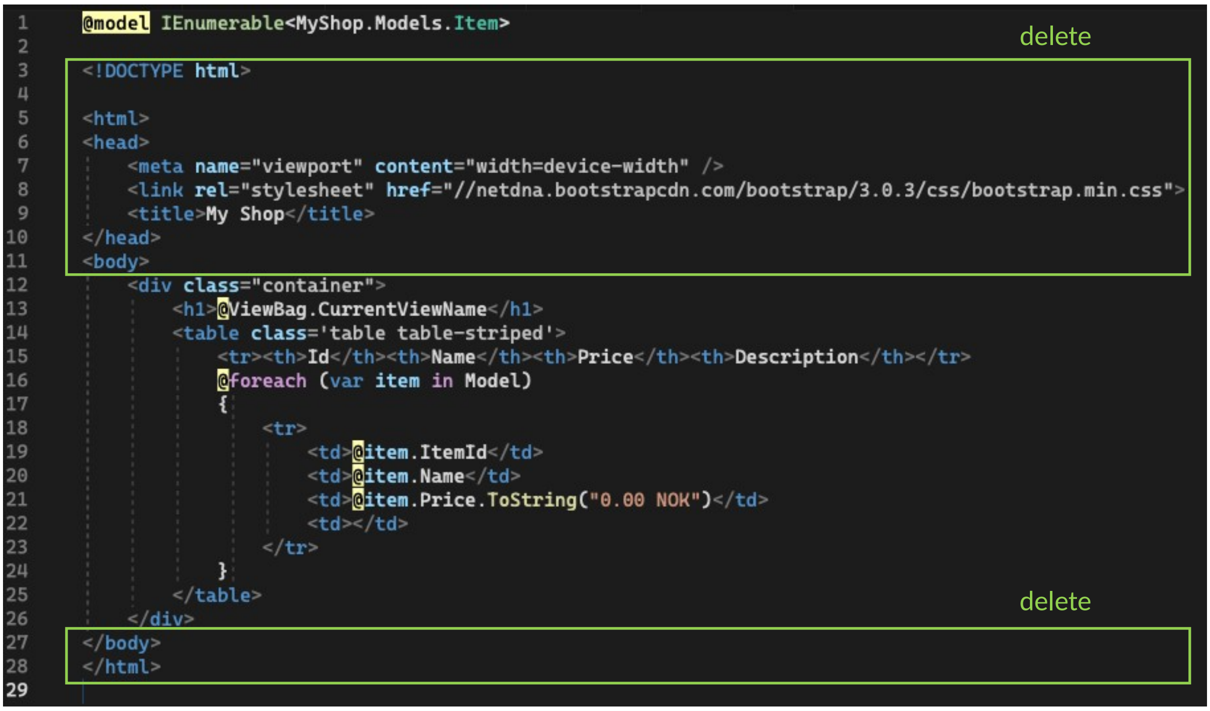
1. **Layout**:
   - The _Layout.cshtml file in the Views/Shared folder defines the common layout structure for the application.
   - Individual views will be rendered within this layout using the @RenderBody() placeholder.
2. **Imports**:
   - The _ViewImports.cshtml file specifies common namespaces and tag helpers, ensuring they are available across all views without needing to be redefined.
3. **Start**:
   - The _ViewStart.cshtml file is executed before any view, typically setting the layout that the views should use. This ensures a consistent layout is applied unless explicitly overridden.

Example Workflow

1. **Request Handling**: A request is made to an action in a controller.
2. **Action Execution**: The controller action returns a view.
3. **View Rendering**:
   - Before rendering the view, `_ViewStart.cshtml` sets the layout to `_Layout.cshtml`.
   - `_ViewImports.cshtml` is processed, making common namespaces and tag helpers available.
   - The specific view is rendered within the `@RenderBody()` section of `_Layout.cshtml`.

Simplify Table.cshtml

Change the Table.cshtml file to this (remove the head and other information):

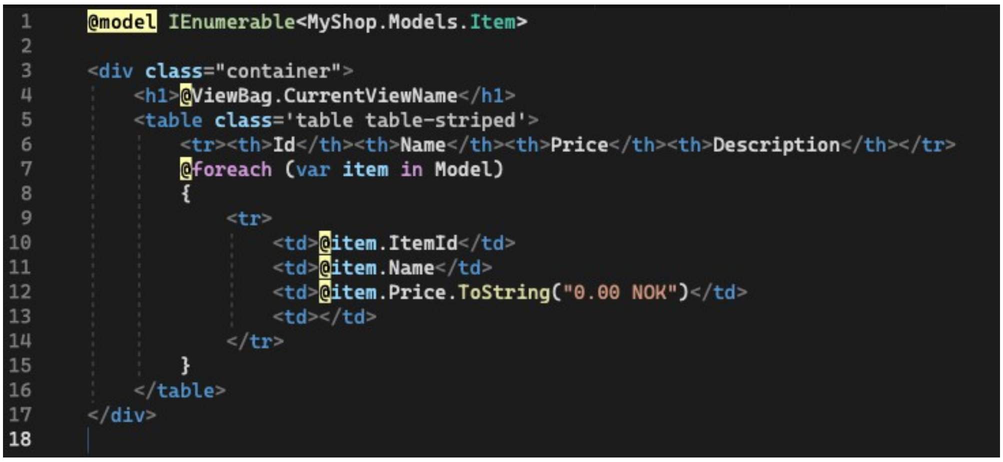


delete

delete

```
@model IEnumerable<MyShop.Models.Item>

<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <link rel="stylesheet" href="//netdna.bootstrapcdn.com/bootstrap/3.0.3/css/bootstrap.min.css">
    <title>My Shop</title>
</head>
<body>
    <div class="container">
        <h1>@ViewBag.CurrentViewName</h1>
        <table class='table table-striped'>
            <tr><th>Id</th><th>Name</th><th>Price</th><th>Description</th></tr>
            @foreach (var item in Model)
            {
                <tr>
                    <td>@item.ItemId</td>
                    <td>@item.Name</td>
                    <td>@item.Price.ToString("0.00 NOK")</td>
                    <td></td>
                </tr>
            }
        </table>
    </div>
</body>
</html>
```

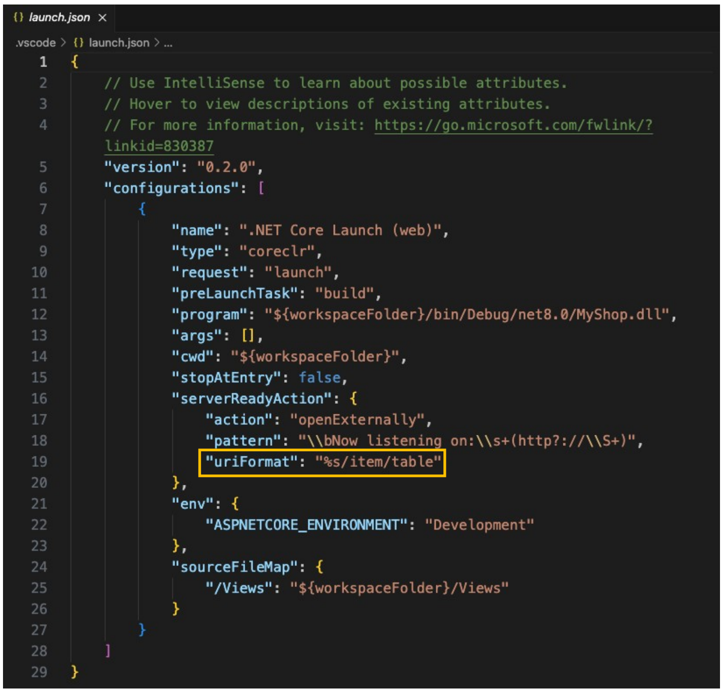
After change:



```
@model IEnumerable<MyShop.Models.Item>

<div class="container">
    <h1>@ViewBag.CurrentViewName</h1>
    <table class='table table-striped'>
        <tr><th>Id</th><th>Name</th><th>Price</th><th>Description</th></tr>
        @foreach (var item in Model)
        {
            <tr>
                <td>@item.ItemId</td>
                <td>@item.Name</td>
                <td>@item.Price.ToString("0.00 NOK")</td>
                <td></td>
            </tr>
        }
    </table>
</div>
```

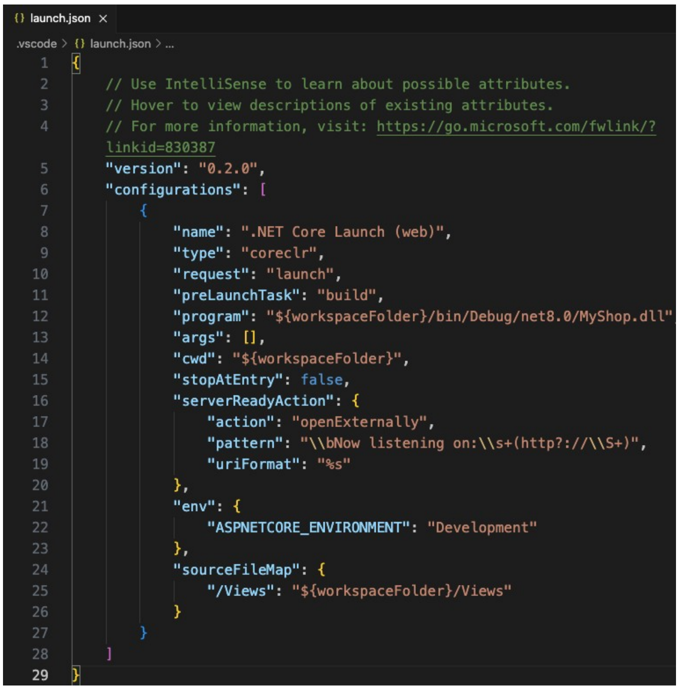
Reset default launchURL

Go to .vscode/launch.json and change the launch URI.



```json
{
    // Use IntelliSense to learn about possible attributes.
    // Hover to view descriptions of existing attributes.
    // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387
    "version": "0.2.0",
    "configurations": [
        {
            "name": ".NET Core Launch (web)",
            "type": "coreclr",
            "request": "launch",
            "preLaunchTask": "build",
            "program": "${workspaceFolder}/bin/Debug/net8.0/MyShop.dll",
            "args": [],
            "cwd": "${workspaceFolder}",
            "stopAtEntry": false,
            "serverReadyAction": {
                "action": "openExternally",
                "pattern": "\\bNow listening on:\\s+(http?://\\S+)",
                "uriFormat": "%s/item/table"
            },
            "env": {
                "ASPNETCORE_ENVIRONMENT": "Development"
            },
            "sourceFileMap": {
                "/Views": "${workspaceFolder}/Views"
            }
        }
    ]
}
```

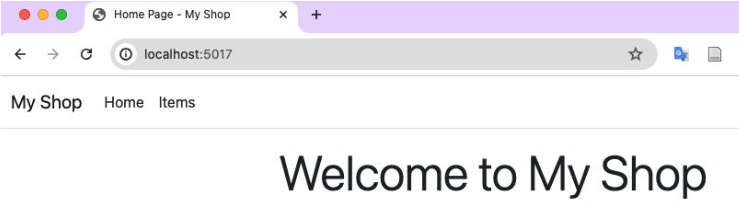
After change:



```json
{
    // Use IntelliSense to learn about possible attributes.
    // Hover to view descriptions of existing attributes.
    // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387
    "version": "0.2.0",
    "configurations": [
        {
            "name": ".NET Core Launch (web)",
            "type": "coreclr",
            "request": "launch",
            "preLaunchTask": "build",
            "program": "${workspaceFolder}/bin/Debug/net8.0/MyShop.dll"
            "args": [],
            "cwd": "${workspaceFolder}",
            "stopAtEntry": false,
            "serverReadyAction": {
                "action": "openExternally",
                "pattern": "\\bNow listening on:\\s+(http?://\\S+)",
                "uriFormat": "%s"
            },
            "env": {
                "ASPNETCORE_ENVIRONMENT": "Development"
            },
            "sourceFileMap": {
                "/Views": "${workspaceFolder}/Views"
            }
        }
    ]
}
```

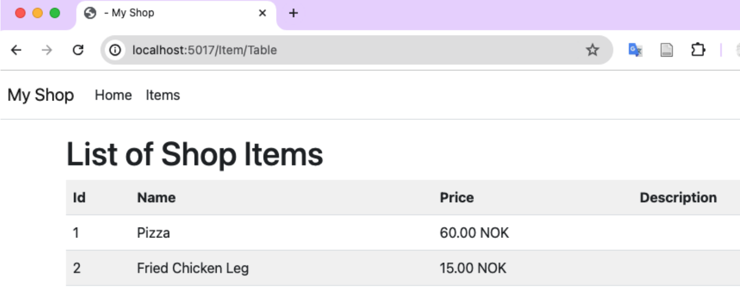
Run Project

Now start debugging (press F5) , the default page becomes /Home/Index



Welcome to My Shop

Click Items, we go to /Item/Table



List of Shop Items

Id	Name	Price	Description
1	Pizza	60.00 NOK	
2	Fried Chicken Leg	15.00 NOK	

Congratulations! You have added HomeView and Navigation to the project ☺