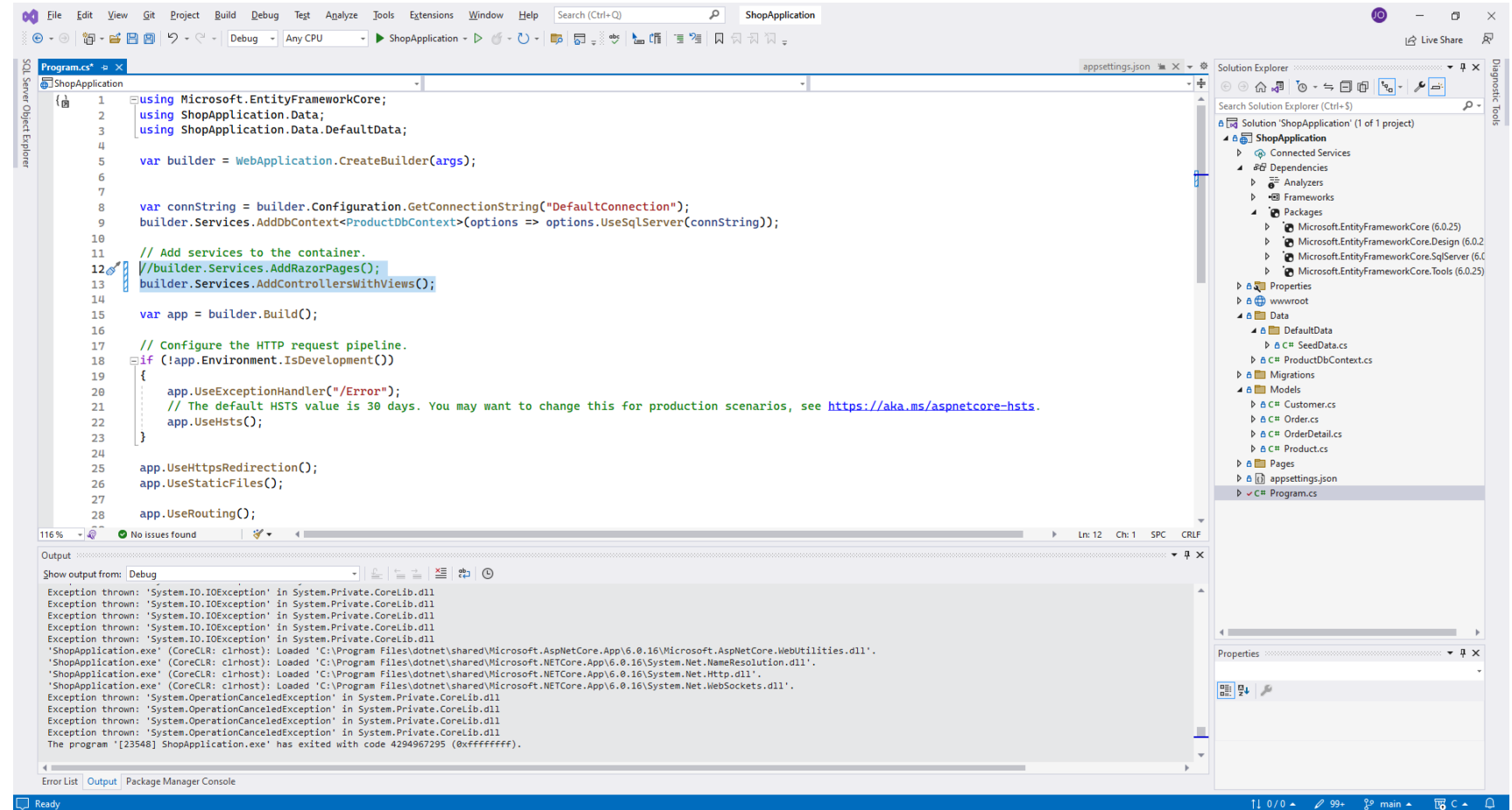


# Change Empty .NET application to MVC

Empty to Model View Component

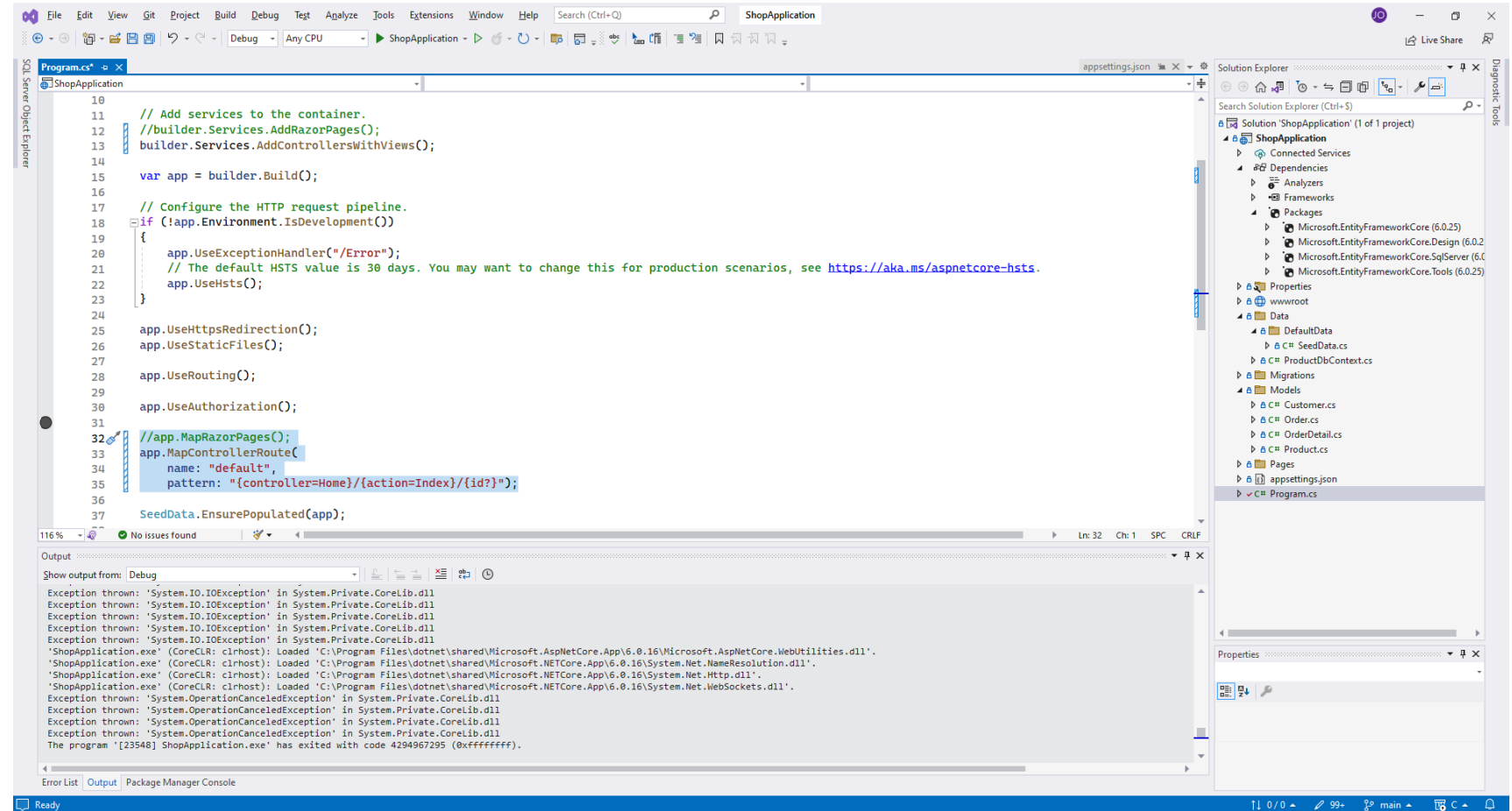
# Program.cs

- In Program.cs We Modify the services so we will use controllers with views. Telling the program we will work with MVC instead of Razor



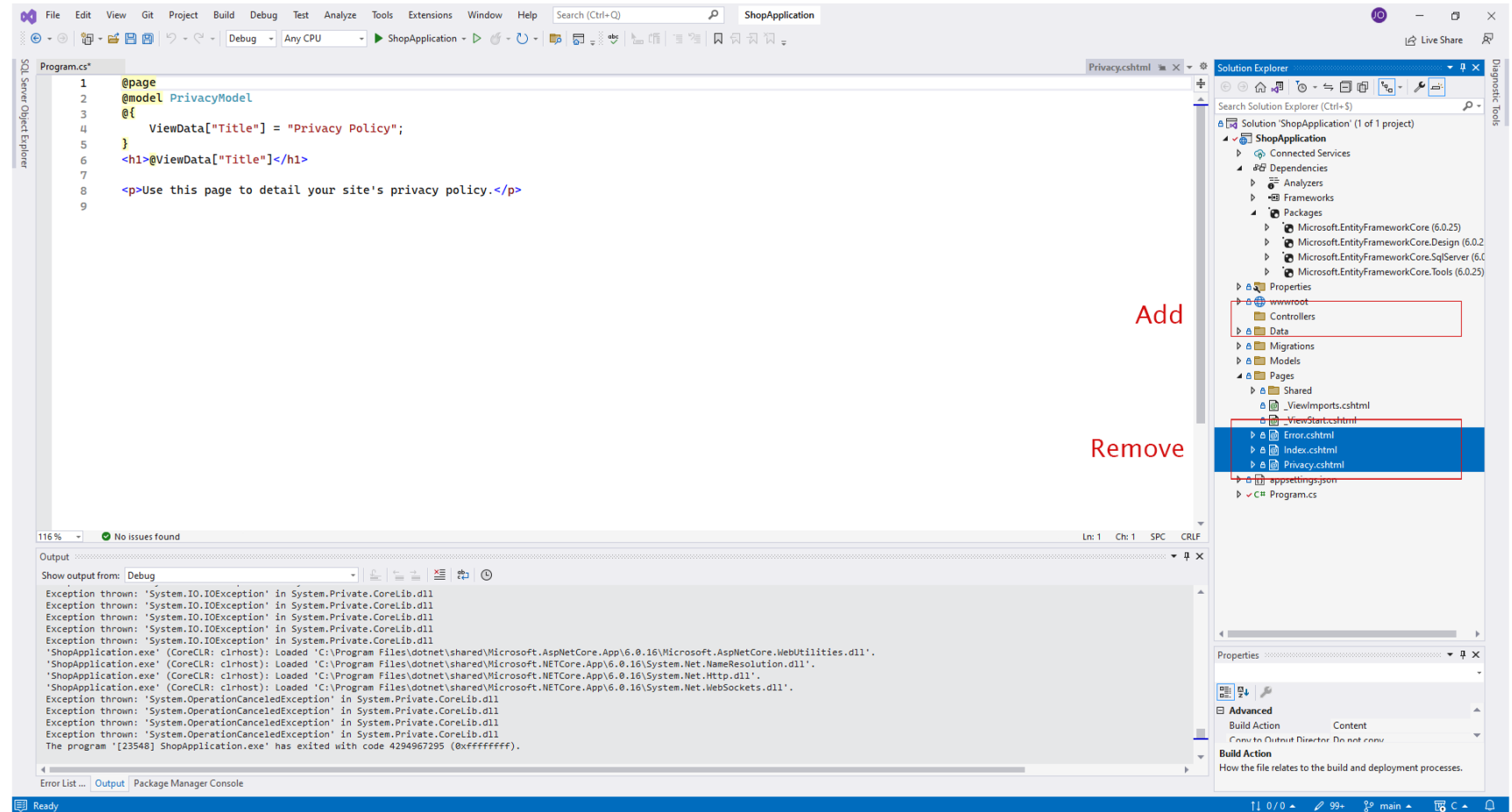
# Program.cs

- Still in Program.cs We modify the following code, since we don't need to Start Razorpages, we will also not need to set the routing for it. So we set our custom routing as followed



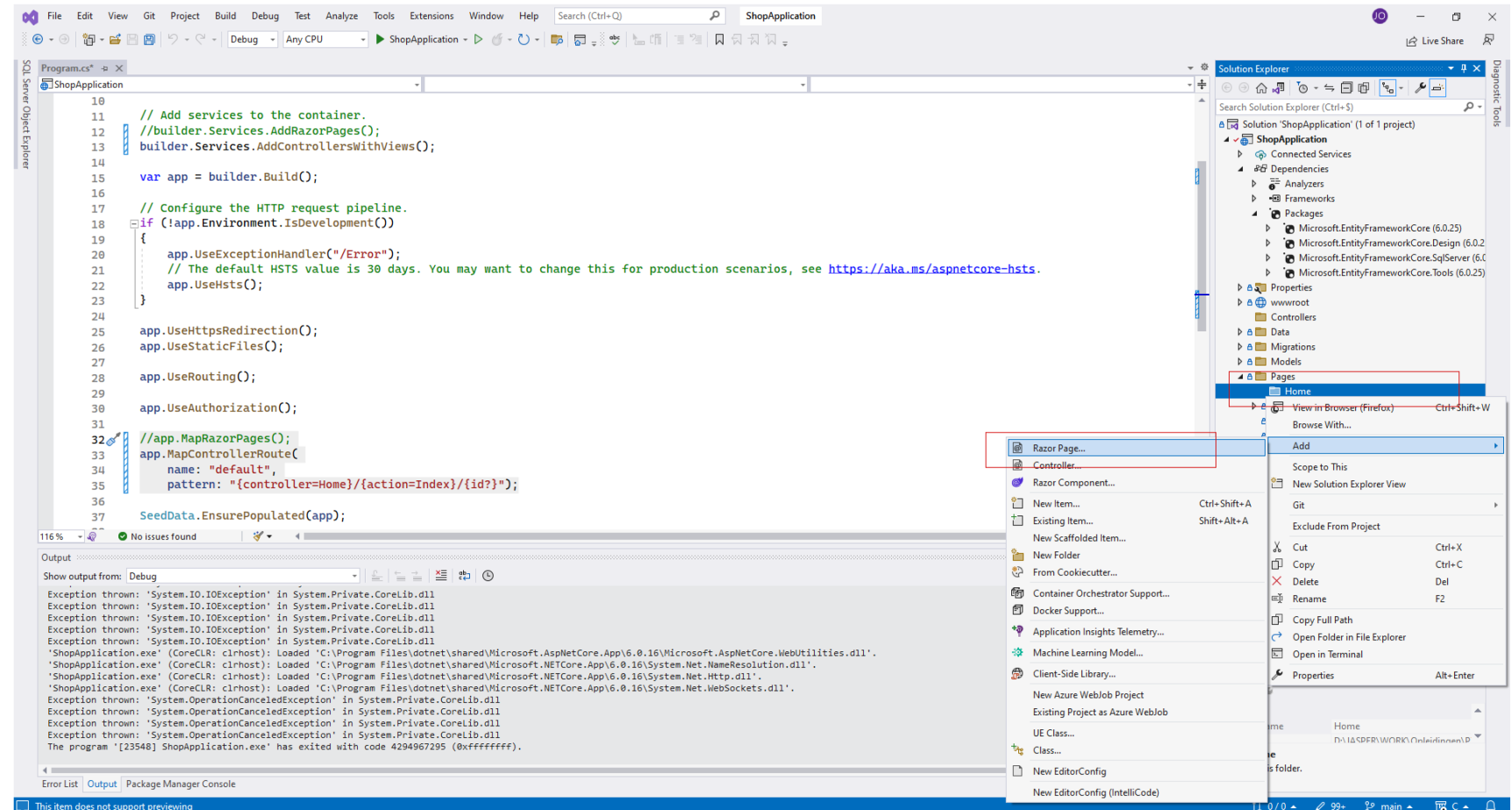
# Required folders

- Then in our solution -> Project, We will remove the basic cshtml files in the Pages folder and add a folder named 'Controllers'



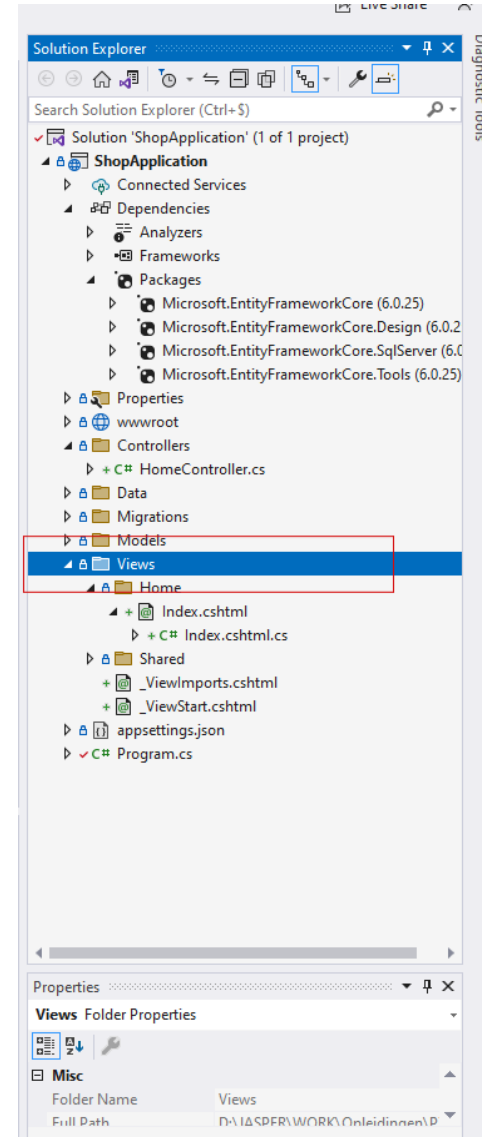
# Views

- We add a folder named 'Home' inside the Pages/View folder. This folder will be our Home page folder
- We add a Razor Page in the Home folder



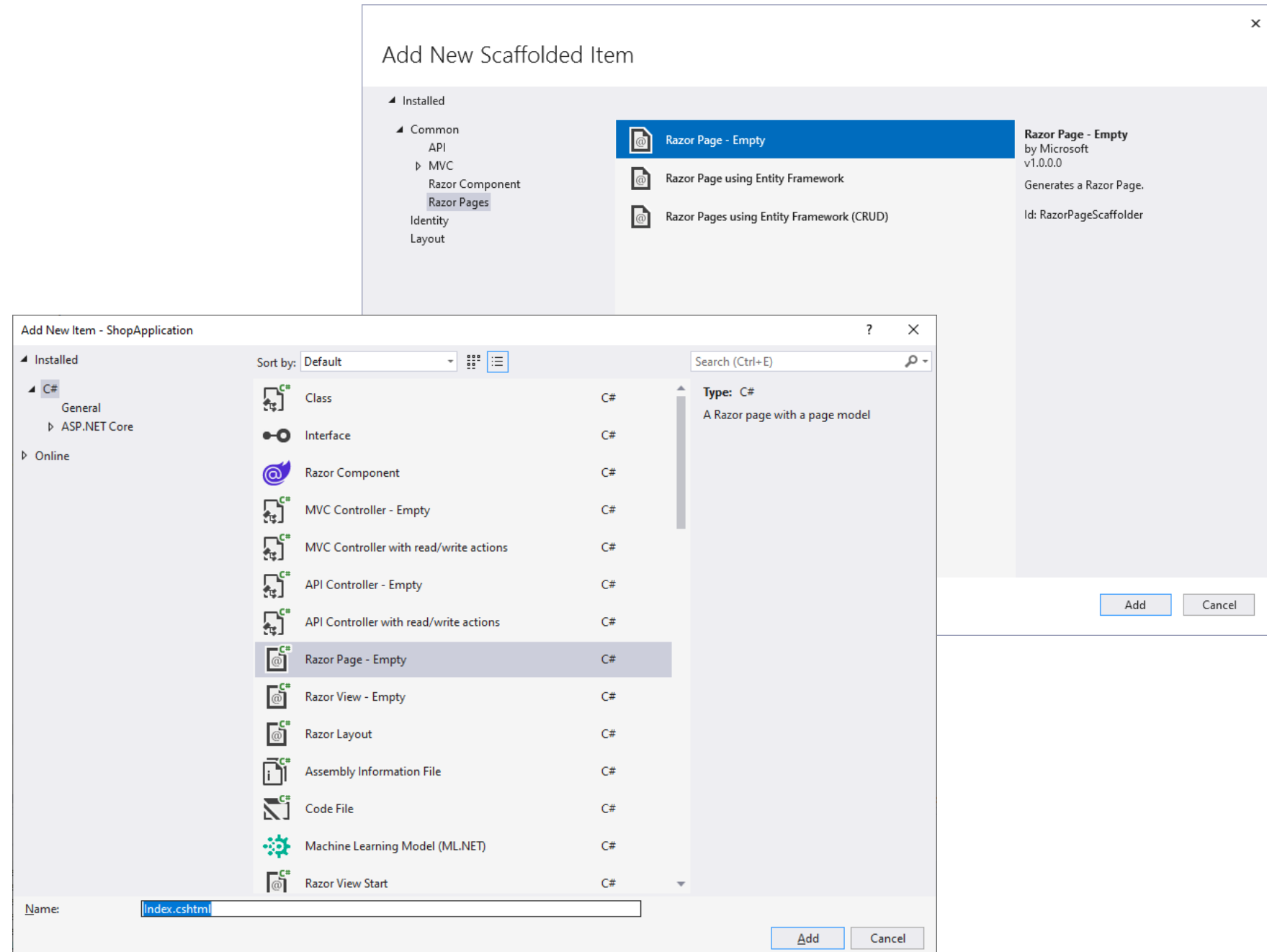
# Views

- We change the name from Pages to Views



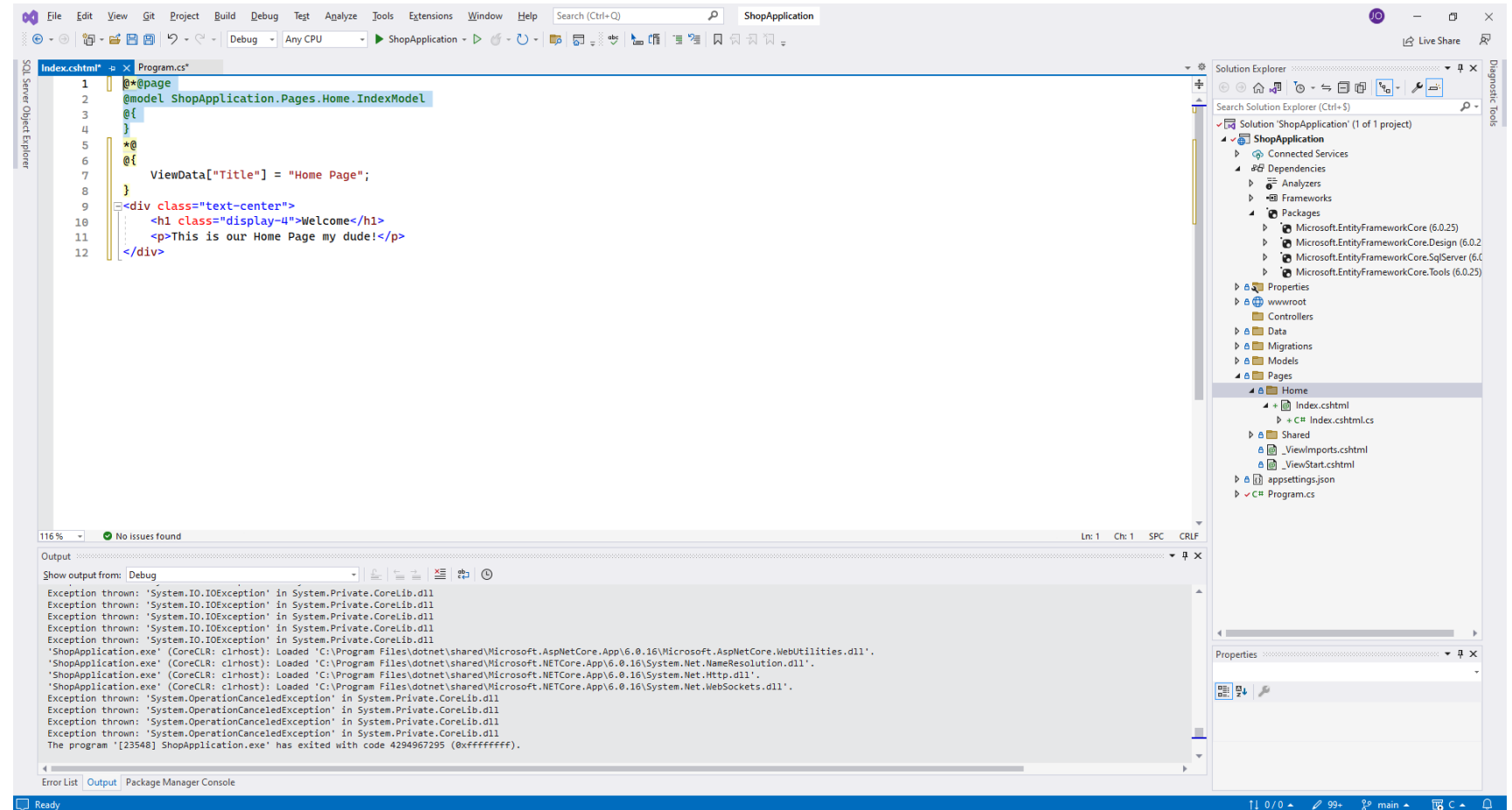
# Views

- For this example, we will add an Empty Razor Page
- We will name this Page Index



# Views

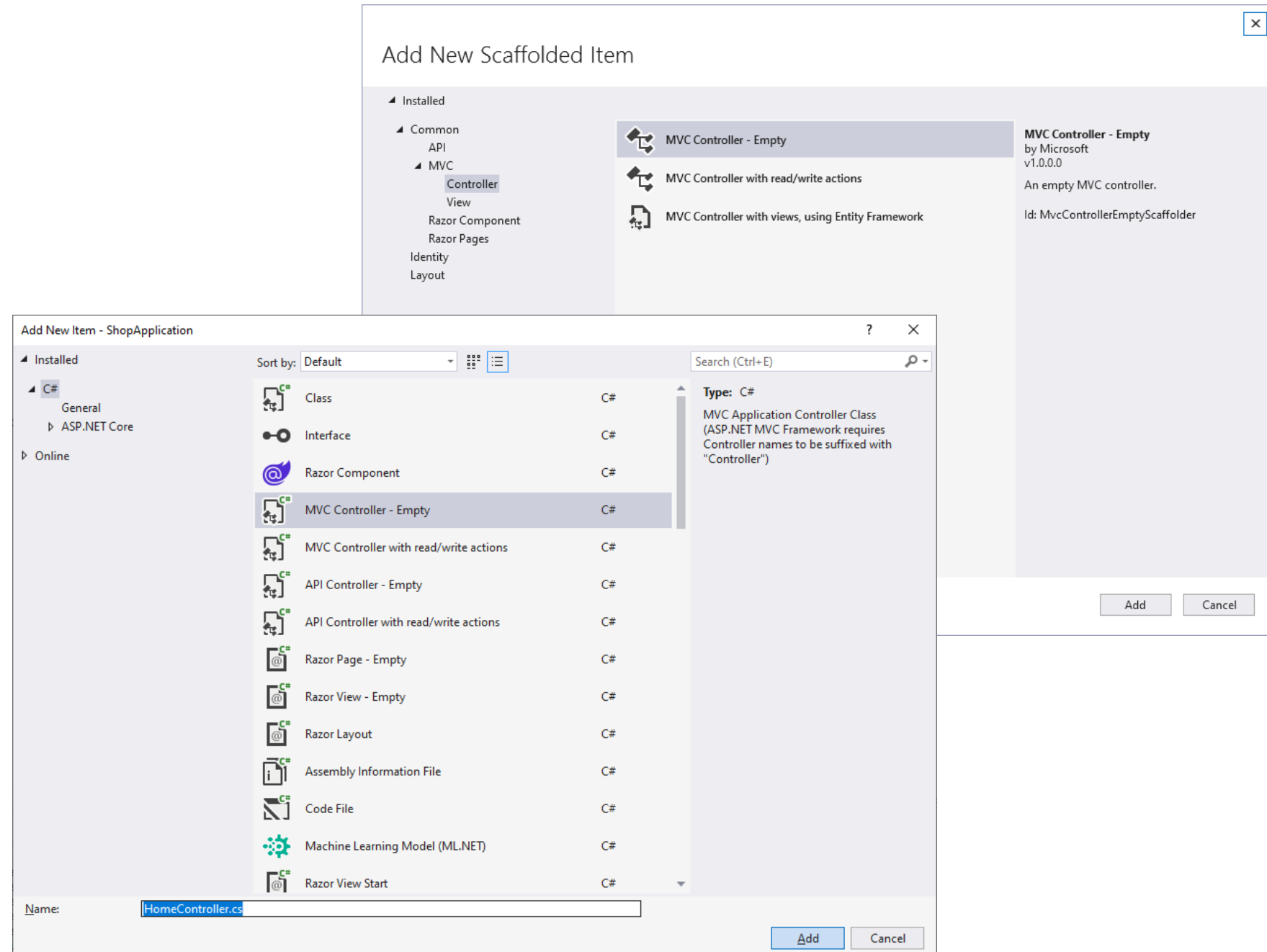
- We remove the default text and modify it like so
- Everything between '@{ }' above the project is where we place our C# code. If we bring in code or models, it happens here
- Everything below it can be standard HTML/CSS/JS





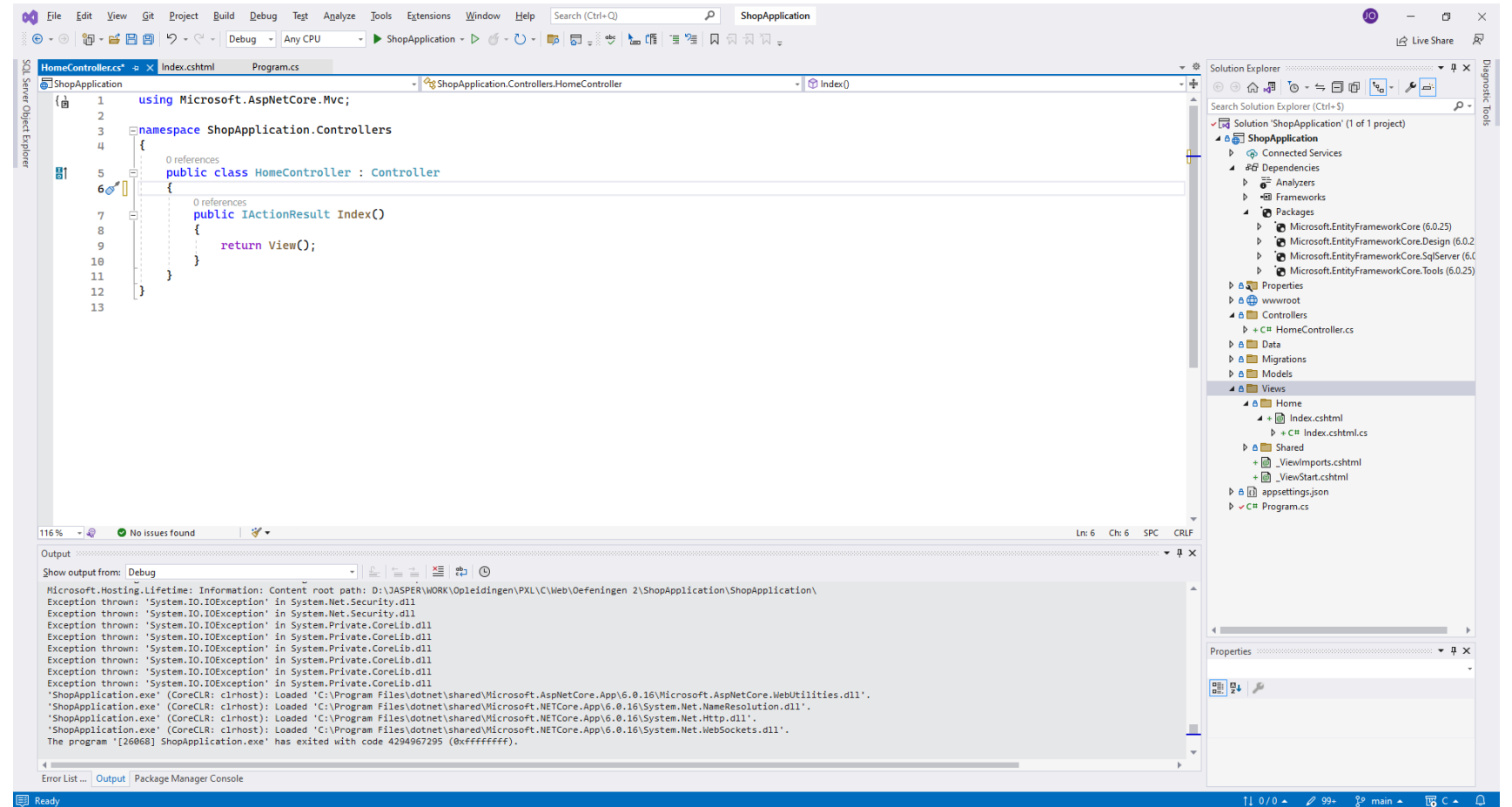
# Controllers

- Since we added one part of the Model View Controller, which is the View (Page) we will now add the Controller
- In your Controllers folder we will now add... a controller !



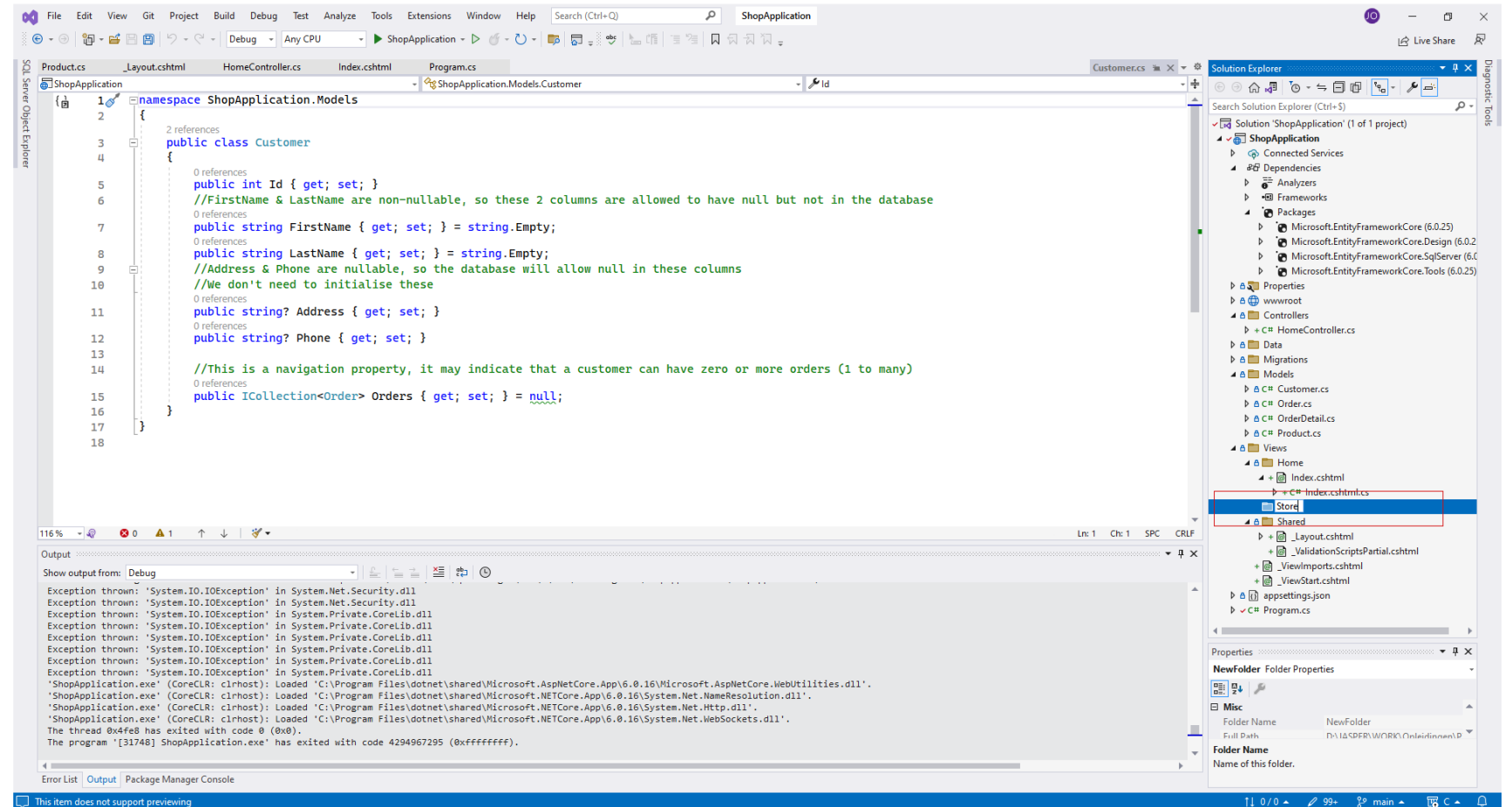
# Controllers

- In our empty Controller We only have the controller for Index



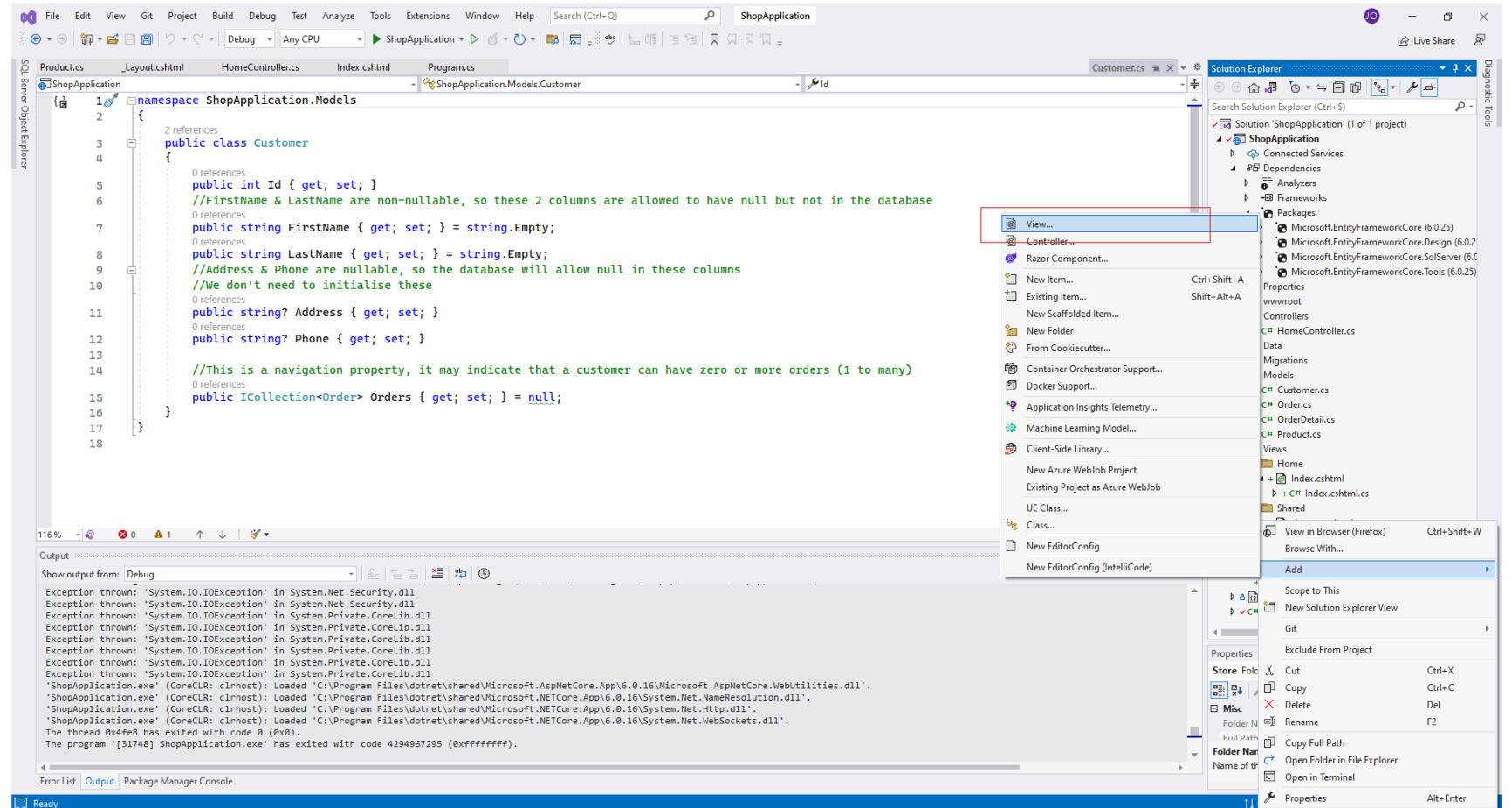
# Views

- In Views we add a new folder, this folder must have the name of which page you want to add. We want a Store navigation page now, so we add this inside Views



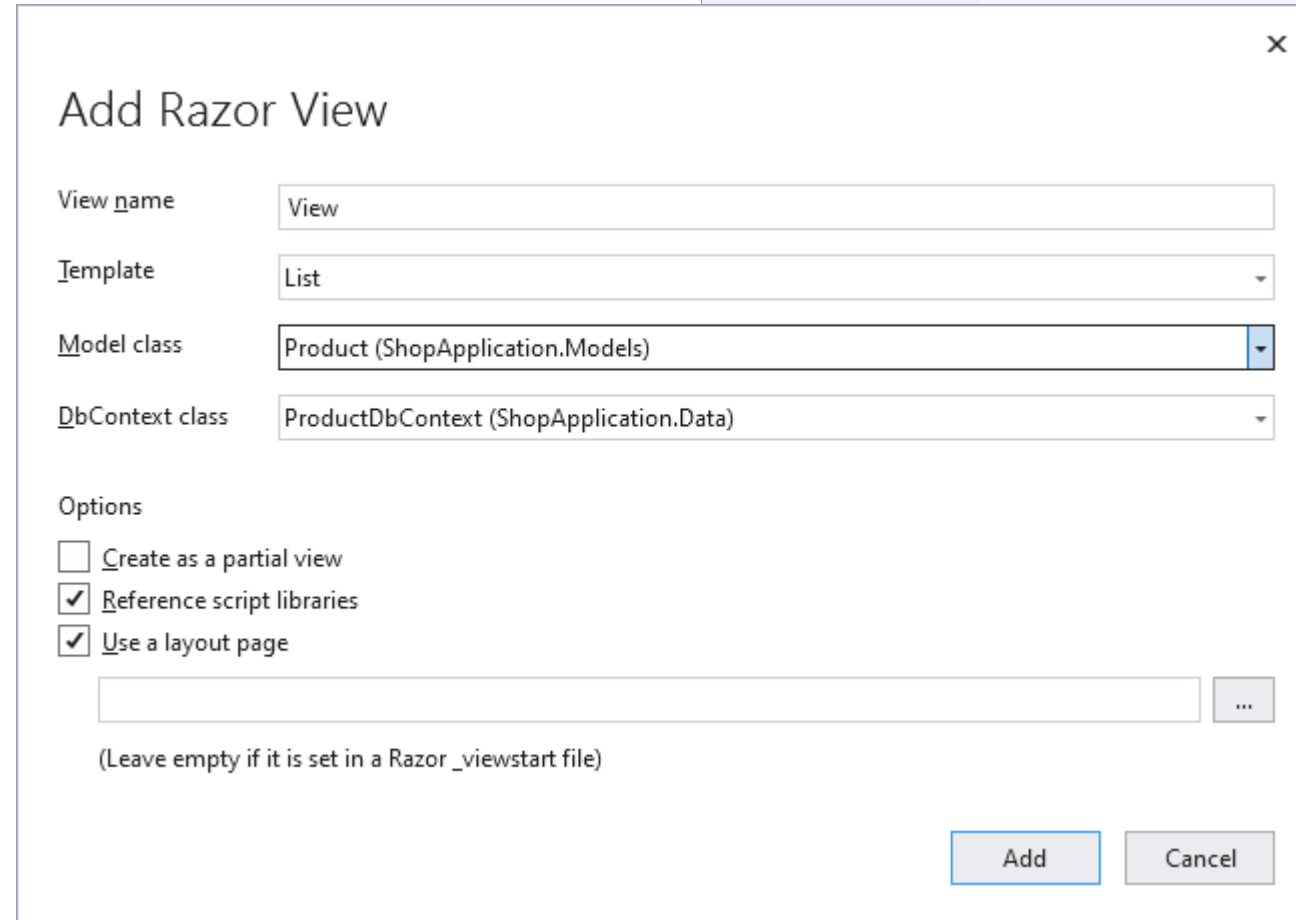
# Views

- In Views -> Store we add a new View



# Views

- This time we will not use the Empty
- We name our View 'View'
- The template we use is List, here you can choose what the action is that you will be doing on this page
- Since we will need to display products in our Store page we will choose 'Product' as our model.
- Since we want our DB to be accessible we select ProductDbContext



**Add Razor View**

View name: View

Template: List

Model class: Product (ShopApplication.Models)

DdbContext class: ProductDbContext (ShopApplication.Data)

Options

☐ Create as a partial view

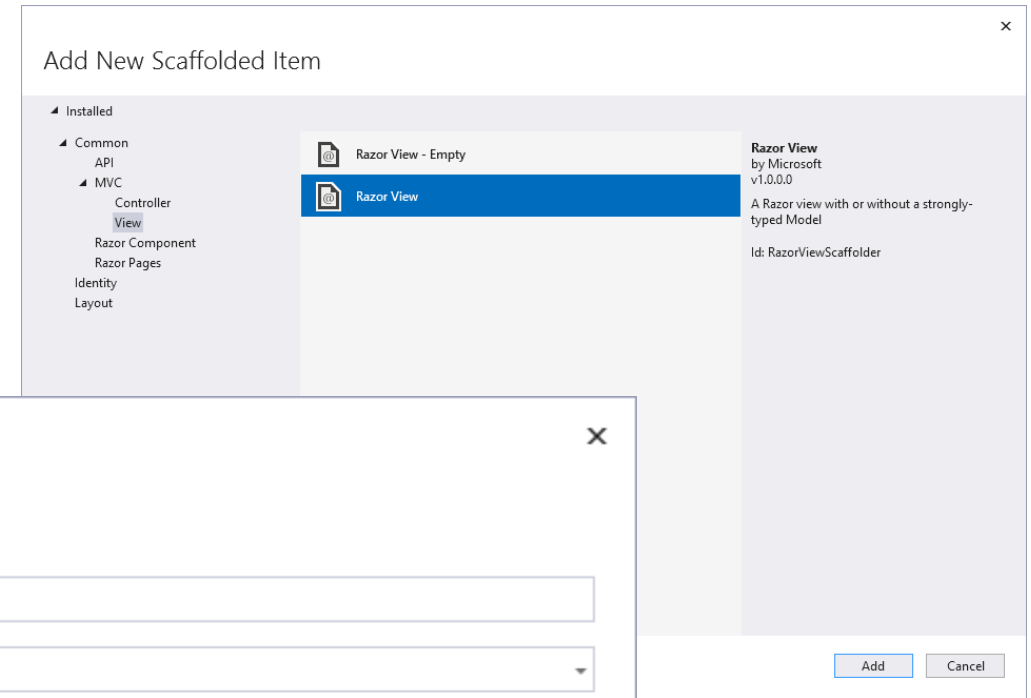
☒ Reference script libraries

☒ Use a layout page

...

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

Add Cancel



# Controllers

- For now we will leave everything as it is and add a controller for Shop
- We select our model 'Product' and leave the rest as default
- We then rename the controller to match the View we want to display. We want to use the 'Store' view so we rename this the 'StoreController'

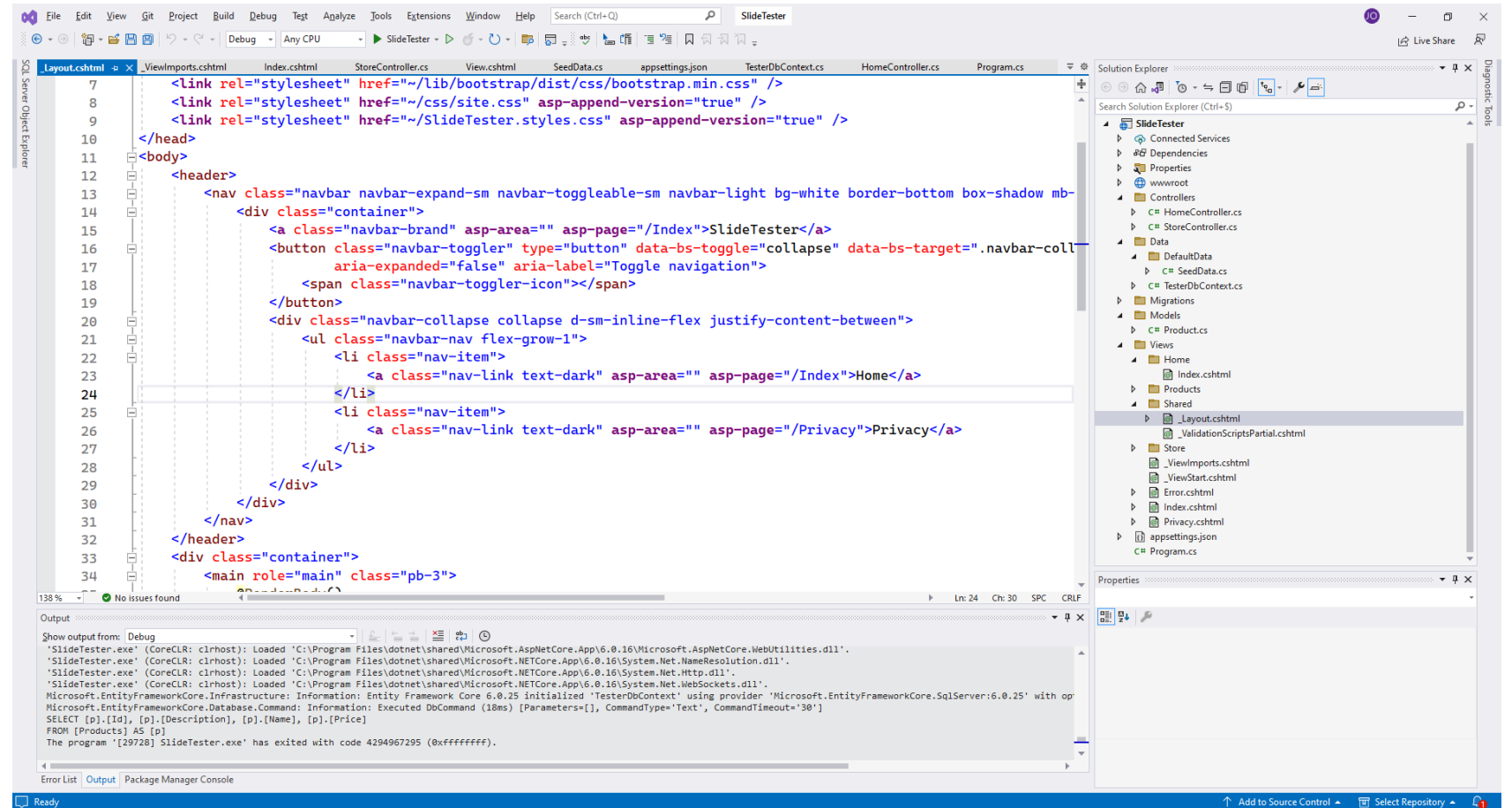
The image shows two overlapping windows from the Visual Studio IDE. The top window is titled 'Add New Scaffolded Item' and displays a tree view of installed scaffolds. The 'Controller' item under the 'MVC' category is selected. The bottom window is titled 'Add MVC Controller with views, using Entity Framework' and contains the following configuration:

- Model class:** Product (ShopApplication.Models)
- DbContext class:** ProductDbContext (ShopApplication.Data)
- Views:**
  - ☒ Generate views
  - ☒ Reference script libraries
  - ☒ Use a layout page
- Controller name:** StoreController

Buttons for 'Add' and 'Cancel' are present at the bottom of both windows.

# Shared

- We navigate to Views -> Shared -> \_Layout.cshtml



# Shared

- We alter the navigation to use 'asp-controller and asp-action

