https://www.halvorsen.blog

ASP.NET Core

# ASP.NET Core with Razor

Hans-Petter Halvorsen

# Contents

- Introduction
  - Introduction to ASP.NET Core.
- Getting Started using ASP.NET Core with Razor
  - How to create an ASP.NET Core Web Application in Visual Studio.
- Using Razor
  - Learn how to create and use Razor syntax.

# Introduction

# ASP.NET Core

ASP.NET Core is a framework for building Web Applications and Services with .NET and C#. ASP.NET Core supports different types, here are some examples:

- ASP.NET Core with Razor Pages The focus in this Tutorial!
- ASP.NET Core MVC
- ASP.NET Core Blazor Web Apps
- ASP.NET Core Web API with controllers
- ASP.NET Core Minimal Web APIs

# ASP.NET History (short version)

- 1996: **Active Server Pages (ASP)** was first released.
  - ASP allowed developers to create dynamic and interactive web applications using server-side scripting.
- 1997: First version of **Visual Studio** was released. **Visual Basic** and Visual C++ was the main programming languages
- 2002: First version of **.NET Framework** and **ASP.NET** was released. Visual Studio .NET was released.
  - A new way to program based on the new .NET Framework platform and the new **C#** programming language was released.
  - The development type was “ASP.NET Web Forms Pages”, which focused on making Web Apps in the same way as making Windows Forms Apps.
- 2009: **ASP.NET MVC** was released and became the new standard way to create ASP.NET Web Applications.
- 2016: **.NET Core** and **ASP.NET Core** was released. Until this .NET Framework and ASP.NET was Windows only. This was a new approach was Cross-platform.
- 2017: **ASP.NET Core with Razor Pages** was introduced. This has been the new standard way to create Server-side Web applications from Microsoft. It has many similarities to PHP but uses C# combined with Razor syntax.
- 2019: **ASP.NET Core Blazor** was introduced. This is a client-side development approach that uses C# and Razor instead of JavaScript on the client-side.
- 2020: **.NET** 5 was released, and this release unified the older .NET Framework with the newer .NET Core approach into one unified development model called just **.NET**.

The name “**ASP.NET Core**” is the name Microsoft uses today for their **web development platform**. The “ASP.NET Core” web development platform is an umbrella that now includes all the above-mentioned development types and apps. Microsoft has a long history of using confusing and changing names which makes everything more complicated.

# ASP.NET Core

- ASP.NET Core is a framework for web development.
- ASP.NET Core is based on .NET and C#.
- What is the difference between ASP.NET Core and .NET frameworks?
  - ASP.NET Core is specifically designed for web development, while the .NET framework covers a broader range of application types, including Windows desktop, mobile, and web applications.
- In ASP.NET Core Razor code and layout are separated into 2 files; The layout file has the extension “. cshtml”, and the code-behind file has the extension “. cshtml.cs” (where “cs” is short for C#).
- The layout files “. cshtml” use something called Razor syntax and are mixed with HTML.
- ASP, ASP.NET and ASP.NET Core is made by Microsoft.
- Homepage: [https://dotnet.microsoft.com/en-us/apps/aspnet](https://dotnet.microsoft.com/en-us/apps/aspnet)

# ASP.NET and Razor

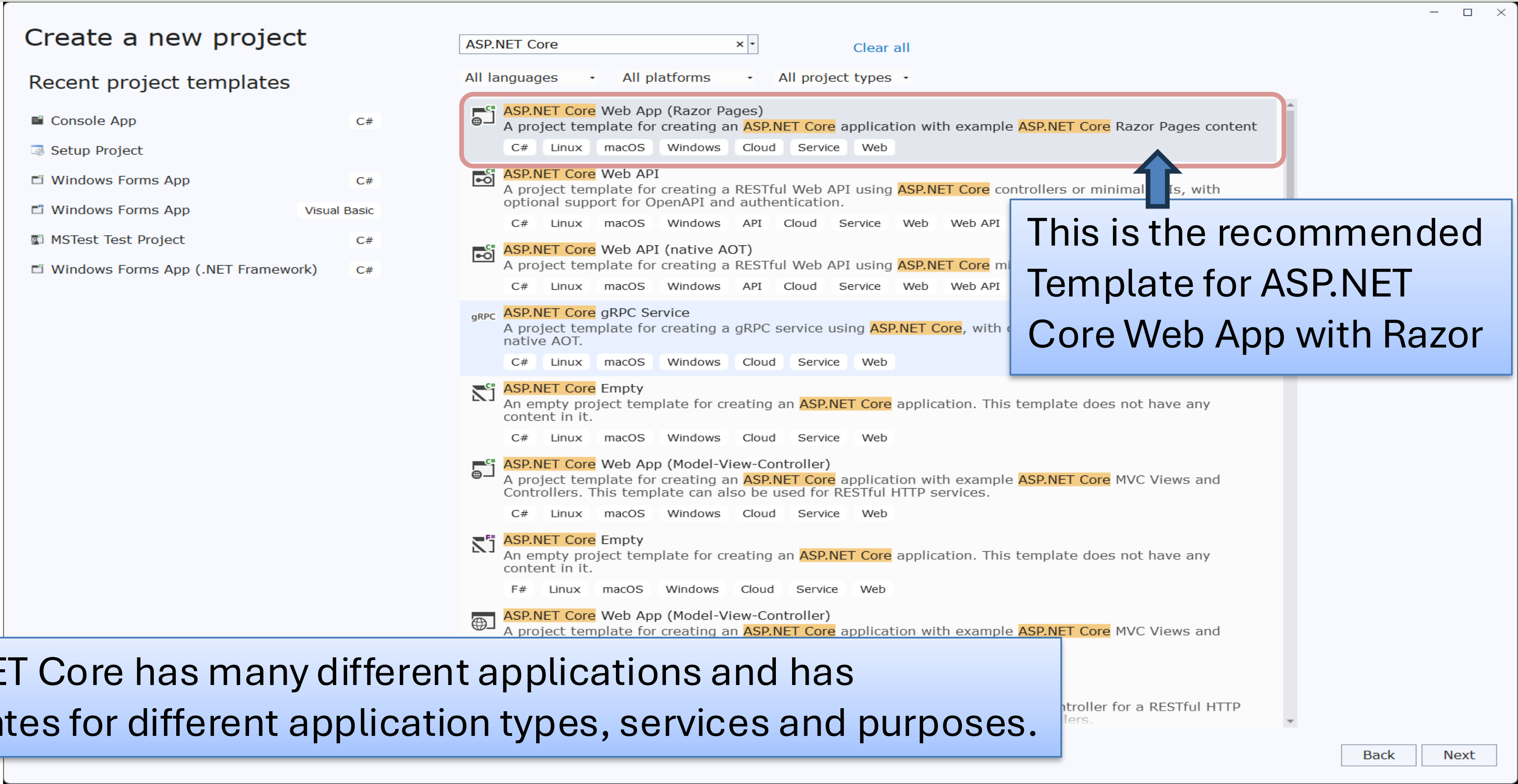
- Razor is a markup syntax for embedding server-based code into ASP.NET Core webpages.
- The Razor syntax consists of Razor markup, C#, and HTML.
- Files containing Razor code generally have a .cshtml file extension.
- The default Razor language is HTML.
- Rendering HTML from Razor markup is no different than rendering HTML from an HTML file.
- HTML markup in .cshtml Razor files is rendered by the server unchanged.

# Getting Started using ASP.NET Core with Razor

Hans-Petter Halvorsen

Table of Contents

# ASP.NET Core Web App with Razor



Create a new project

Recent project templates

| | |
|---|---|
| Console App | C# |
| Setup Project | |
| Windows Forms App | C# |
| Windows Forms App | Visual Basic |
| MSTest Test Project | C# |
| Windows Forms App (.NET Framework) | C# |

ASP.NET Core ×

Clear all

All languages · All platforms · All project types

ASP.NET Core Web App (Razor Pages)
A project template for creating an ASP.NET Core application with example ASP.NET Core Razor Pages content
C# · Linux · macOS · Windows · Cloud · Service · Web

ASP.NET Core Web API
A project template for creating a RESTful Web API using ASP.NET Core controllers or minimal Is, with optional support for OpenAPI and authentication.
C# · Linux · macOS · Windows · API · Cloud · Service · Web · Web API

ASP.NET Core Web API (native AOT)
A project template for creating a RESTful Web API using ASP.NET Core mi
C# · Linux · macOS · Windows · API · Cloud · Service · Web · Web API

ASP.NET Core gRPC Service
A project template for creating a gRPC service using ASP.NET Core, with native AOT.
C# · Linux · macOS · Windows · Cloud · Service · Web

ASP.NET Core Empty
An empty project template for creating an ASP.NET Core application. This template does not have any content in it.
C# · Linux · macOS · Windows · Cloud · Service · Web

ASP.NET Core Web App (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.
C# · Linux · macOS · Windows · Cloud · Service · Web

ASP.NET Core Empty
An empty project template for creating an ASP.NET Core application. This template does not have any content in it.
F# · Linux · macOS · Windows · Cloud · Service · Web

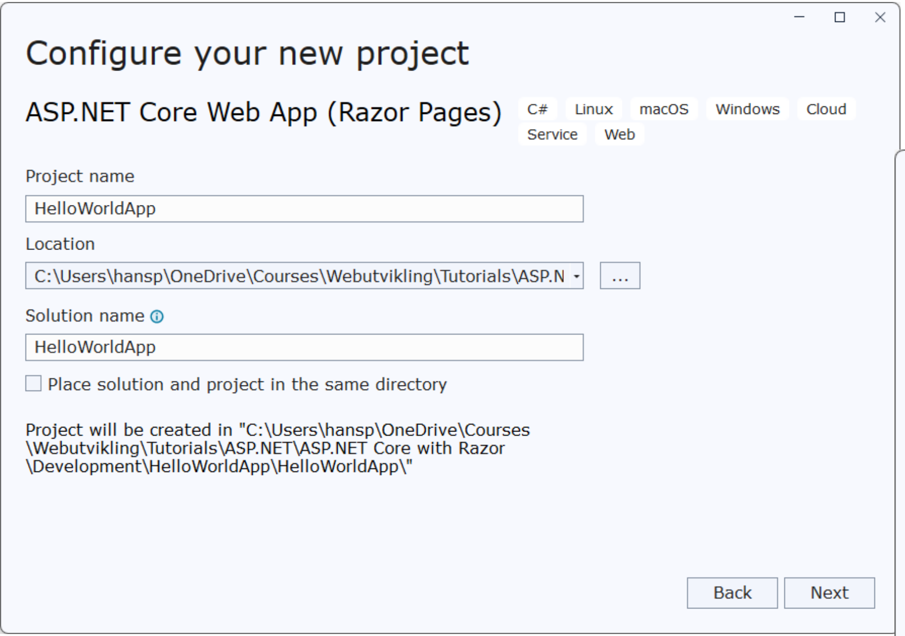
ASP.NET Core Web App (Model-View-Controller)
A project template for creating an ASP.NET Core application with example ASP.NET Core MVC Views and

ntroller for a RESTful HTTP
lers.

This is the recommended Template for ASP.NET Core Web App with Razor

ASP.NET Core has many different applications and has templates for different application types, services and purposes.

# Setting up the Project



## Configure your new project

ASP.NET Core Web App (Razor Pages)

Project name

HelloWorldApp

Location

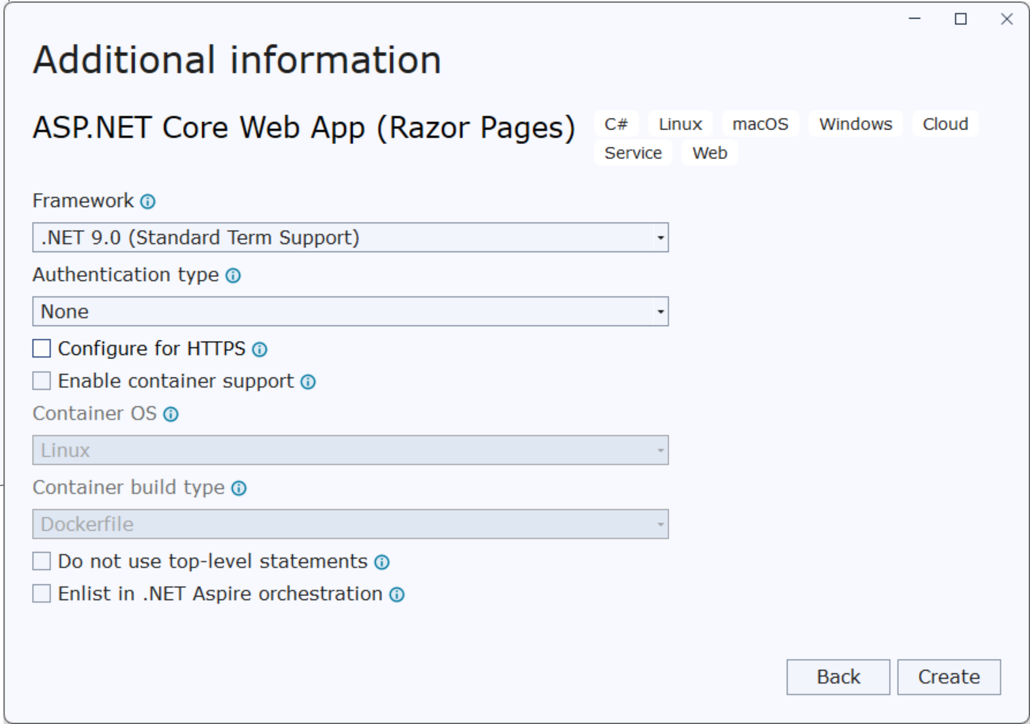
C:\Users\hansp\OneDrive\Courses\Webutvikling\Tutorials\ASP.N

Solution name

HelloWorldApp

□ Place solution and project in the same directory

Project will be created in "C:\Users\hansp\OneDrive\Courses\Webutvikling\Tutorials\ASP.NET\ASP.NET Core with Razor\Development\HelloWorldApp\HelloWorldApp\"



## Additional information

ASP.NET Core Web App (Razor Pages)

Framework

.NET 9.0 (Standard Term Support)

Authentication type

None

□ Configure for HTTPS

□ Enable container support

Container OS

Linux

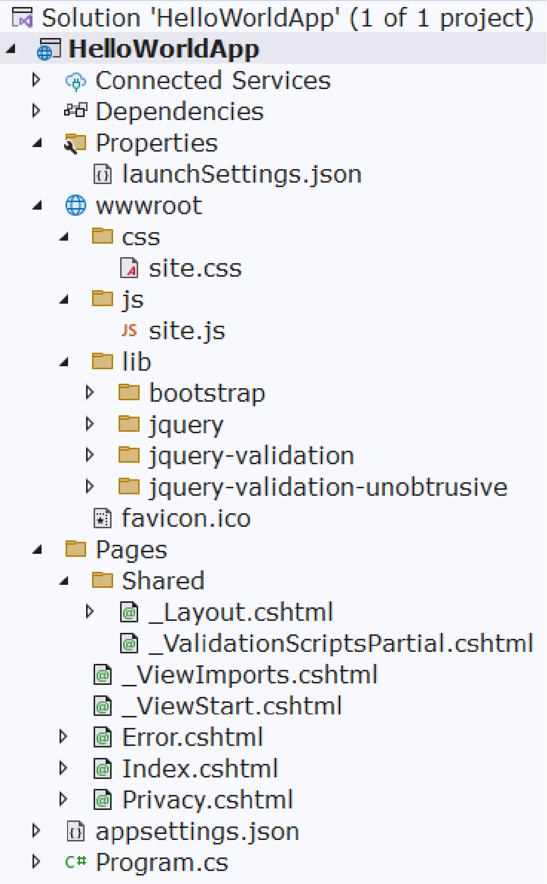
Container build type

Dockerfile

□ Do not use top-level statements

□ Enlist in .NET Aspire orchestration

# Folder and Files



```
Solution 'HelloWorldApp' (1 of 1 project)
  HelloWorldApp
    Connected Services
    Dependencies
    Properties
      launchSettings.json
    wwwroot
      css
        site.css
      js
        site.js
      lib
        bootstrap
        jquery
        jquery-validation
        jquery-validation-unobtrusive
      favicon.ico
    Pages
      Shared
        _Layout.cshtml
        _ValidationScriptsPartial.cshtml
      _ViewImports.cshtml
      _ViewStart.cshtml
      Error.cshtml
      Index.cshtml
      Privacy.cshtml
    appsettings.json
    Program.cs
```

- **appSettings.json**
  - Contains configuration data, such as, e.g., connection strings to databases, etc.
- Program.cs
  - Contains the entry point for the program.
- **wwwroot** folder
  - Contains static files, such as HTML files, JavaScript files and CSS files.
- **Pages** folder
  - You may put your ASP.NET Core web pages here.

# Supporting Files

Supporting files have names that begin with an underscore (_).

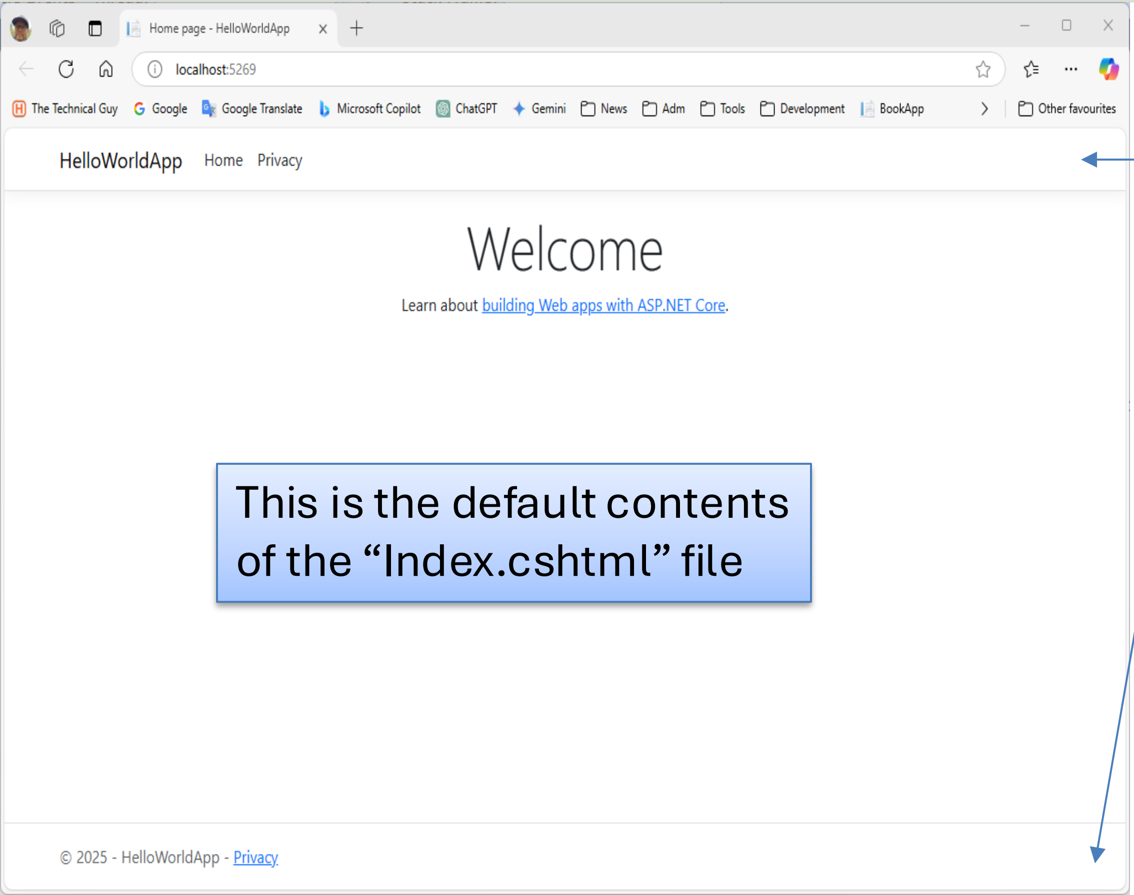
- ***_Layout.cshtml*** file configures UI elements common to all pages. You can use this file to set up the navigation menu at the top of the page, footer, etc.

# Razor Pages

Each Razor page is a pair of 2 files:

- A **.cshtml** file that contains HTML markup with C# code using Razor syntax. This file is referred to as the “Razor page”.
- A **.cshtml.cs** file that contains C# code that handles page events. This is called a “code behind” file or the “Page Model” file.

# Start the Web Application (F5)



HelloWorldApp Home Privacy

Welcome

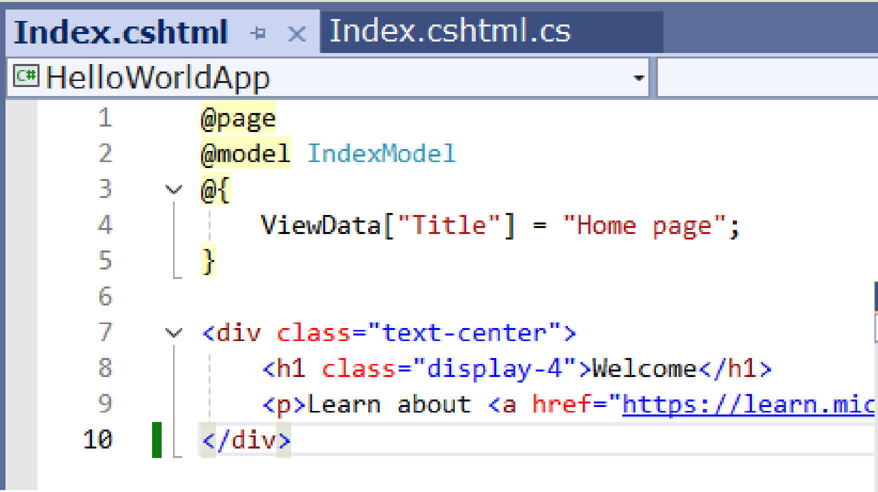
Learn about building Web apps with ASP.NET Core.

© 2025 - HelloWorldApp - Privacy

Header and footer is specified in the "_Layout.cshtml" file

This is the default contents of the "Index.cshtml" file

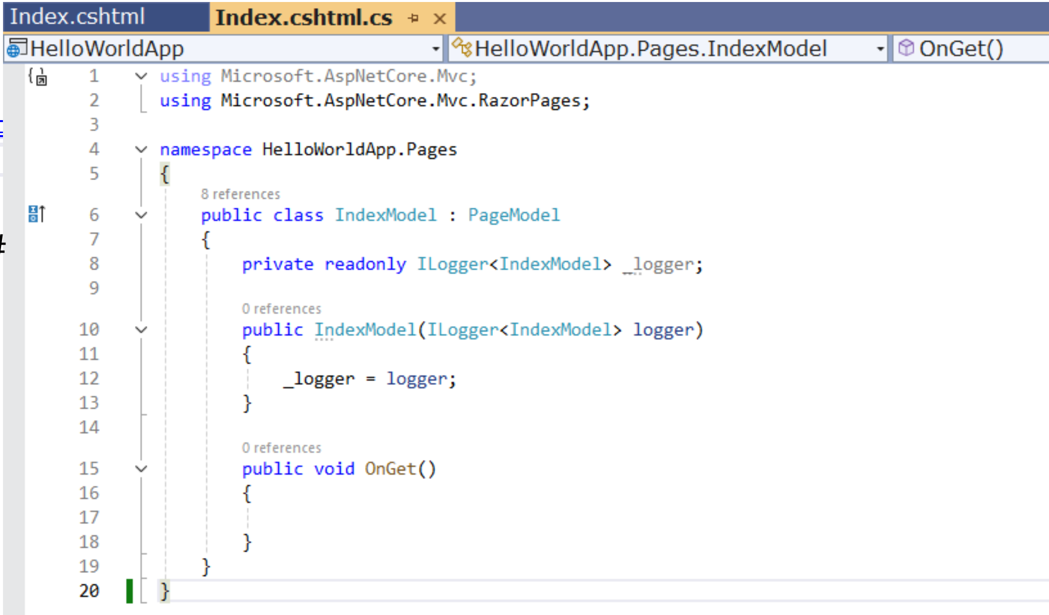
# Index.cshtml



```
@page
@model IndexModel
@{
    ViewData["Title"] = "Home page";
}

<div class="text-center">
    <h1 class="display-4">Welcome</h1>
    <p>Learn about <a href="https://learn.mic
</div>
```

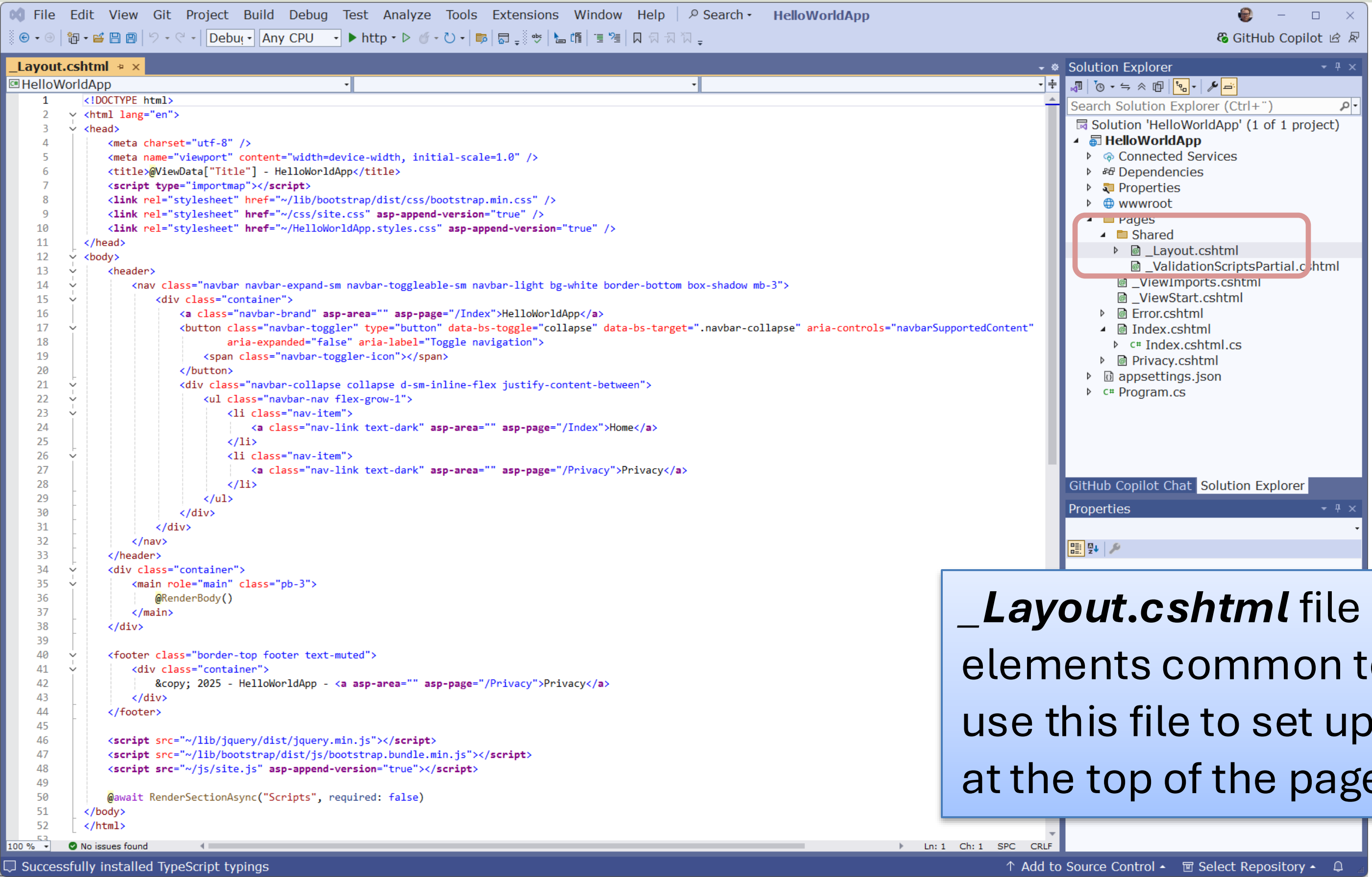
**Index.cshtml** - contains HTML markup with C# code using Razor syntax. The default example don’t have so much Razor though.

**Index.cshtml.cs** - contains C# code that handles page events. This is called a code behind file or Page Model file.



```
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;

namespace HelloWorldApp.Pages
{
    8 references
    public class IndexModel : PageModel
    {
        private readonly ILogger<IndexModel> _logger;

        0 references
        public IndexModel(ILogger<IndexModel> logger)
        {
            _logger = logger;
        }

        0 references
        public void OnGet()
        {

        }
    }
}
```

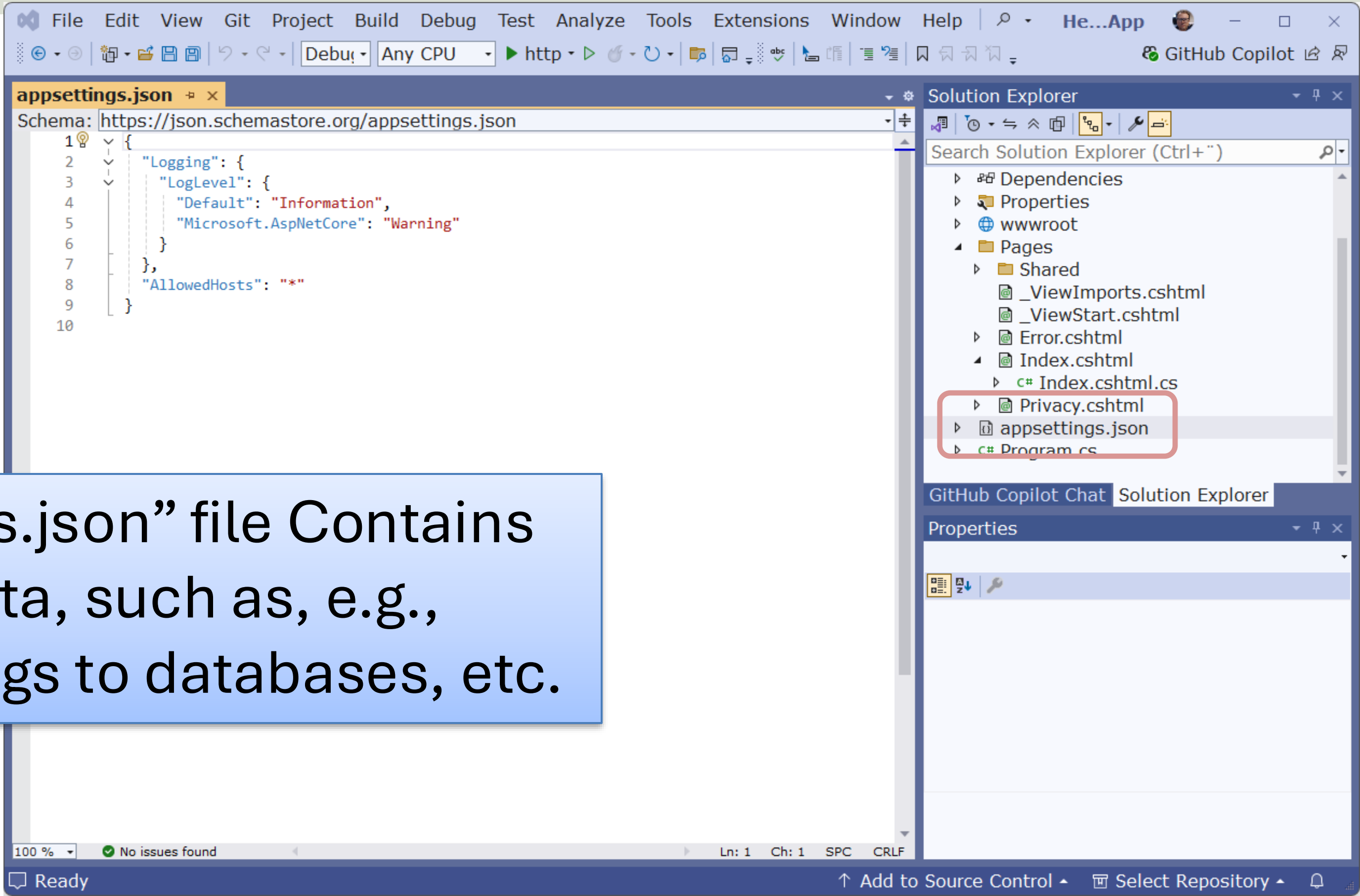
# _Layout.cshtml



```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>@ViewData["Title"] - HelloWorldApp</title>
    <script type="importmap"></script>
    <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="~/HelloWorldApp.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">
                <a class="navbar-brand" asp-area="" asp-page="/Index">HelloWorldApp</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-page="/Index">Home</a>
                        </li>
                        <li class="nav-item">
                            <a class="nav-link text-dark" asp-area="" asp-page="/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; 2025 - HelloWorldApp - <a asp-area="" asp-page="/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>
```

***_Layout.cshtml*** file configures UI elements common to all pages. You can use this file to set up the navigation menu at the top of the page (header), footer, etc.

# appSettings.json



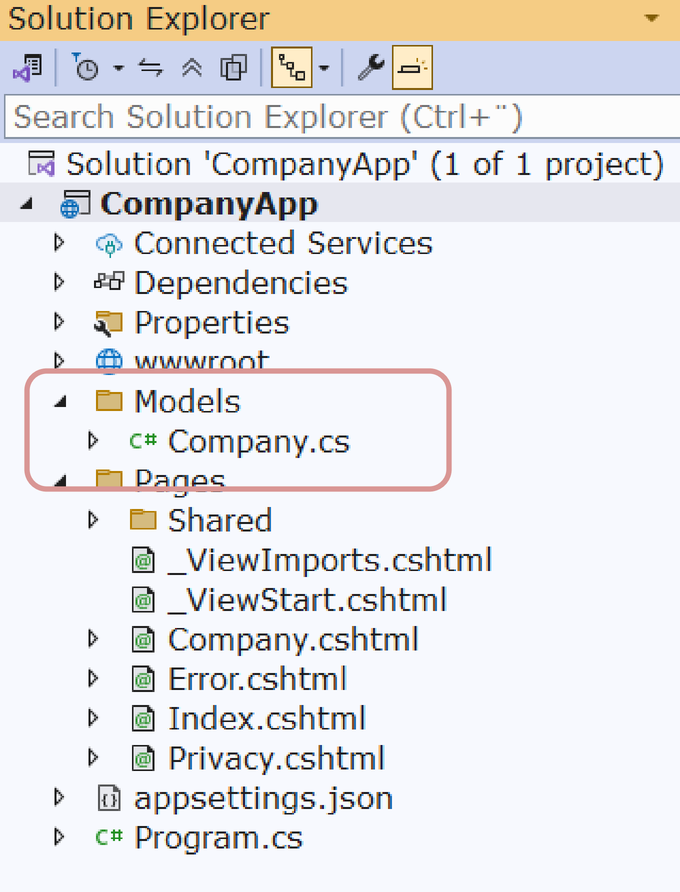
Schema: https://json.schemastore.org/appsettings.json

```json
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*"
}
```

The “appSettings.json” file Contains configuration data, such as, e.g., connection strings to databases, etc.

# Code Files and Classes

- Typically, you want to create and put most of your code into separate Classes and Methods
- You can create a folder called, e.g., “Models”.
- Then you create all your Classes and Methods inside that folder.



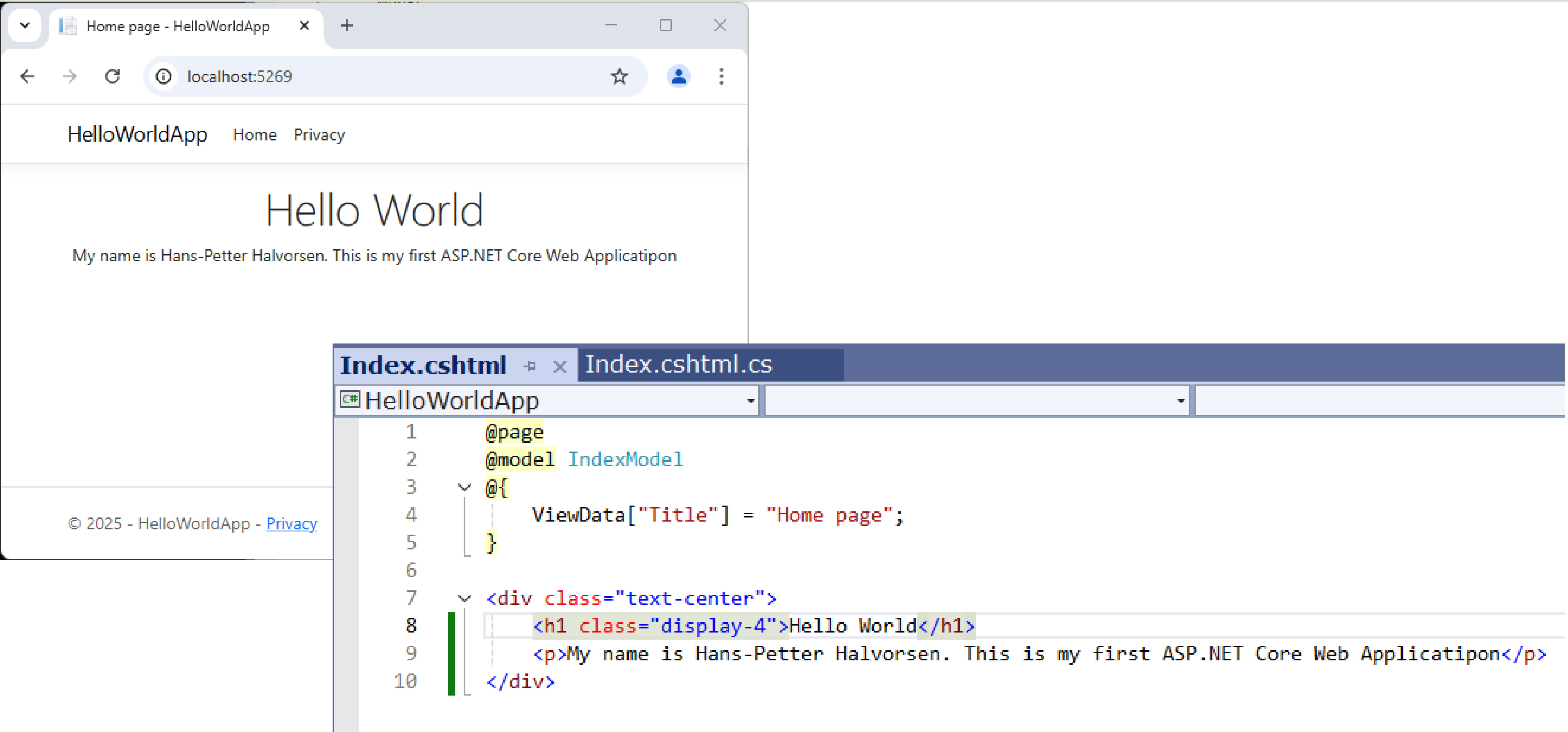
Solution Explorer

Search Solution Explorer (Ctrl+")

Solution 'CompanyApp' (1 of 1 project)

- CompanyApp
  - Connected Services
  - Dependencies
  - Properties
  - wwwroot
  - Models
    - Company.cs
  - Pages
    - Shared
    - _ViewImports.cshtml
    - _ViewStart.cshtml
    - Company.cshtml
    - Error.cshtml
    - Index.cshtml
    - Privacy.cshtml
  - appsettings.json
  - Program.cs

# Hello World



Hello World

My name is Hans-Petter Halvorsen. This is my first ASP.NET Core Web Applicatipon

© 2025 - HelloWorldApp - Privacy

```
@page
@model IndexModel
@{
    ViewData["Title"] = "Home page";
}

<div class="text-center">
    <h1 class="display-4">Hello World</h1>
    <p>My name is Hans-Petter Halvorsen. This is my first ASP.NET Core Web Applicatipon</p>
</div>
```

# Using Razor

Hans-Petter Halvorsen

Table of Contents

# Razor Pages

Each Razor page is a pair of 2 files:

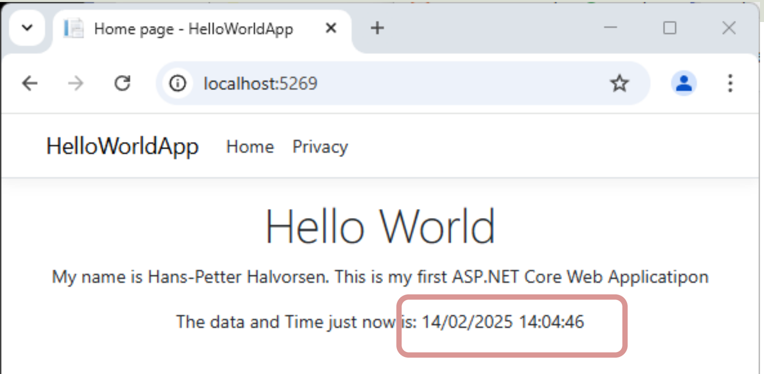
- A **.cshtml** file that contains HTML markup with C# code using Razor syntax. This file is referred to as the “Razor page”.
- A **.cshtml.cs** file that contains C# code that handles page events. This is called a “code behind” file or the “Page Model” file.

# HTML and Razor

**Razor** code is mixed with the HTML code inside the **“.cshtml”** files. Blocks with Razor code (multiple lines) is separated from the HTML using **@{** ... **}** or just **@**... for inline expressions.

```
@{
 string myName = "Hans-Petter Halvorsen";
}
<div>
<h1>Hello World, my name is @myName</h1>
</div>
```

# Example 1



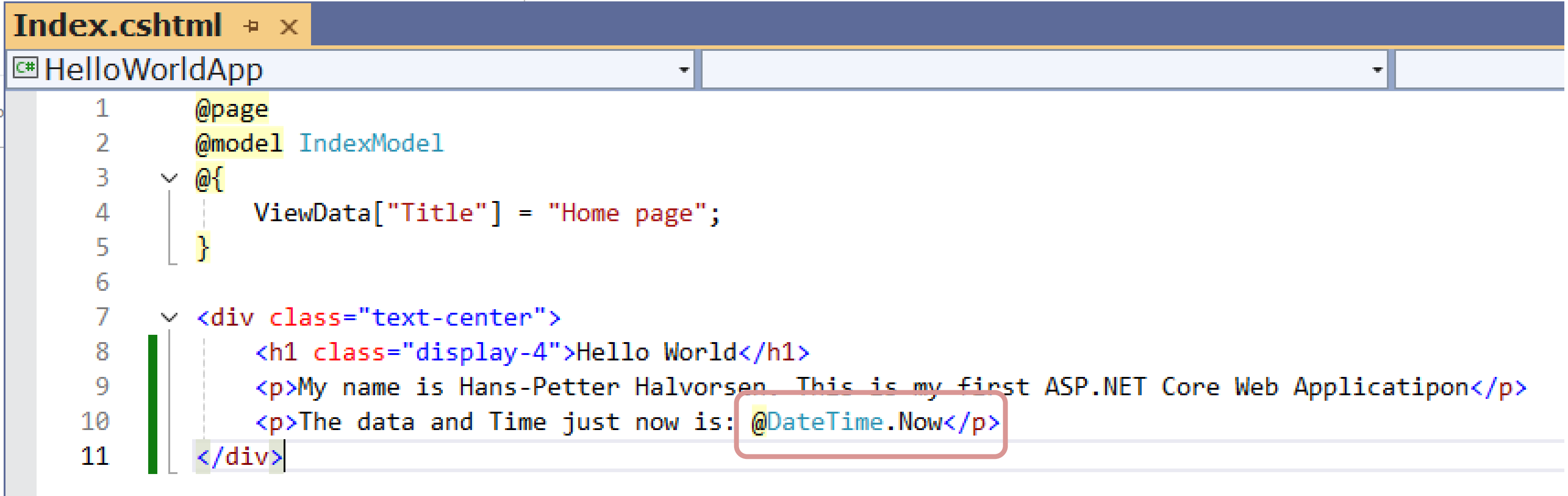
Hello World

My name is Hans-Petter Halvorsen. This is my first ASP.NET Core Web Applicatipon

The data and Time just now is: 14/02/2025 14:04:46

© 2025 - HelloWo

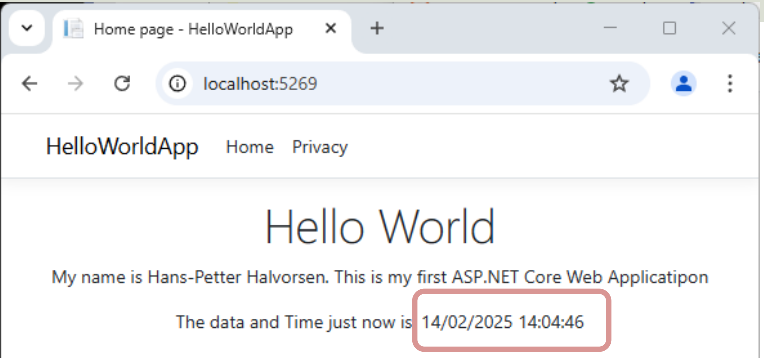
Here is the DataTime.Now used directly



```
@page
@model IndexModel
@{
    ViewData["Title"] = "Home page";
}

<div class="text-center">
    <h1 class="display-4">Hello World</h1>
    <p>My name is Hans-Petter Halvorsen. This is my first ASP.NET Core Web Applicatipon</p>
    <p>The data and Time just now is: @DateTime.Now</p>
</div>
```

# Example 2

Here we first define a variable and then use the variable inline integrated in the HTML code using the @.

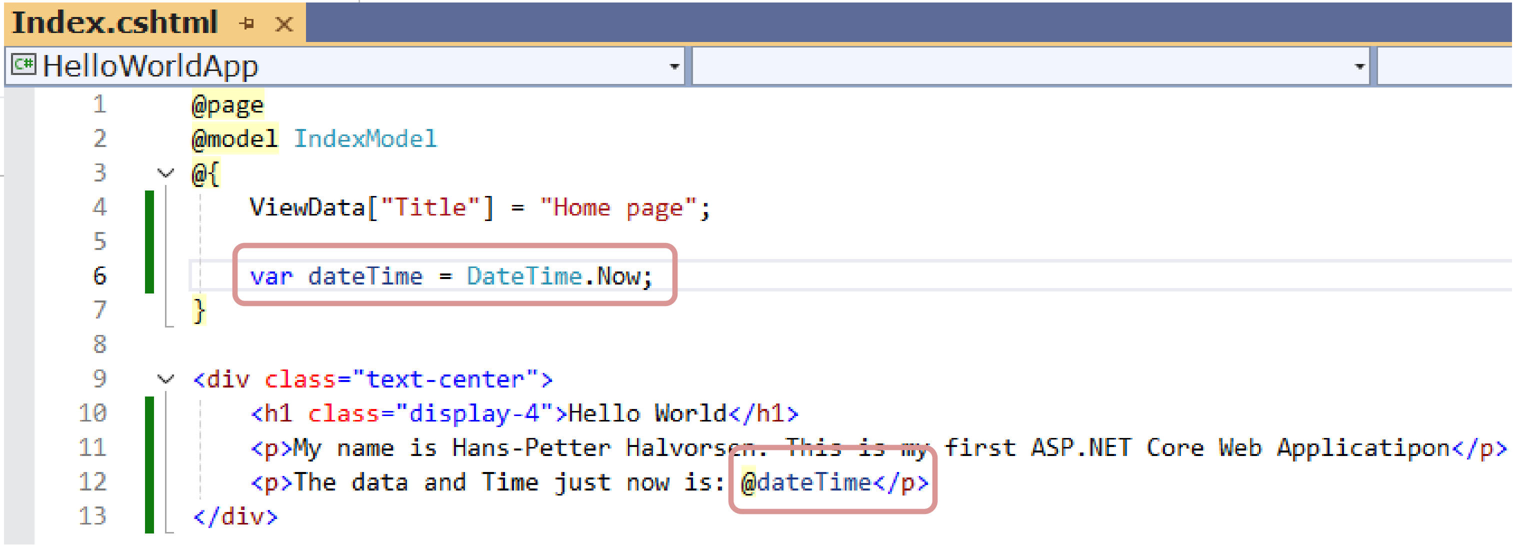


Hello World

My name is Hans-Petter Halvorsen. This is my first ASP.NET Core Web Applicatipon

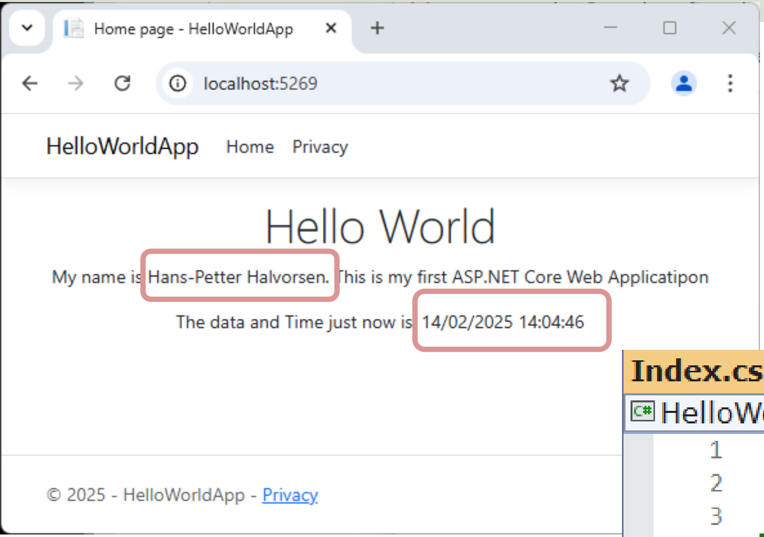
The data and Time just now is 14/02/2025 14:04:46

© 2025 - HelloWorldApp - Privacy



```
@page
@model IndexModel
@{
    ViewData["Title"] = "Home page";

    var dateTime = DateTime.Now;
}

<div class="text-center">
    <h1 class="display-4">Hello World</h1>
    <p>My name is Hans-Petter Halvorsen. This is my first ASP.NET Core Web Applicatipon</p>
    <p>The data and Time just now is: @dateTime</p>
</div>
```

# Example 3



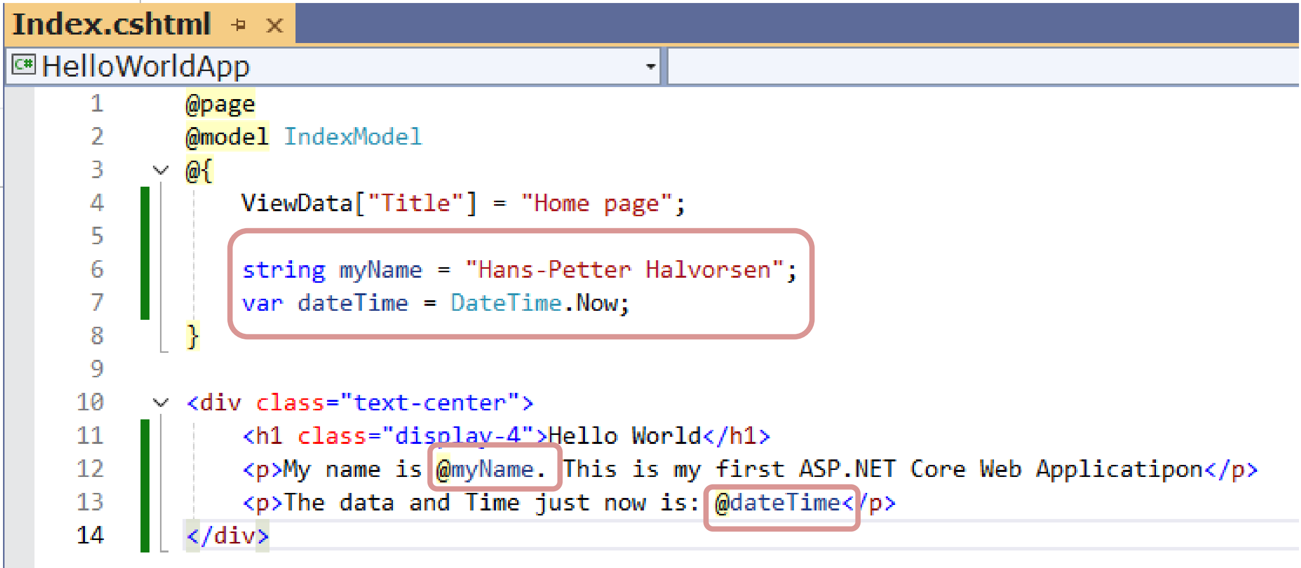
# Hello World

My name is Hans-Petter Halvorsen. This is my first ASP.NET Core Web Applicatipon

The data and Time just now is 14/02/2025 14:04:46

© 2025 - HelloWorldApp - Privacy

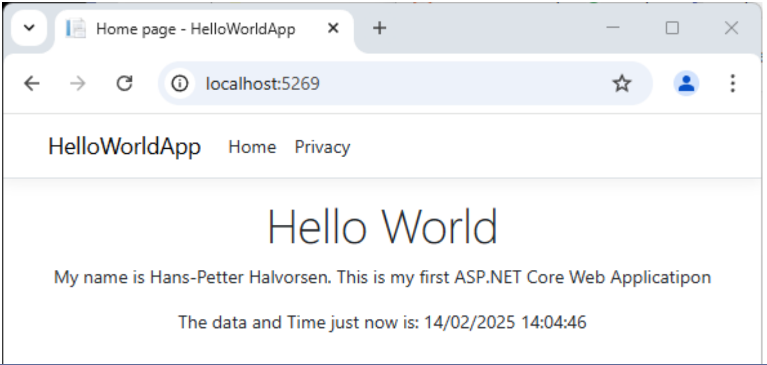
Here we create some C# code in the top and use the variables inline integrated in the HTML code using the @.



```
@page
@model IndexModel
@{
    ViewData["Title"] = "Home page";

    string myName = "Hans-Petter Halvorsen";
    var dateTime = DateTime.Now;
}

<div class="text-center">
    <h1 class="display-4">Hello World</h1>
    <p>My name is @myName. This is my first ASP.NET Core Web Applicatipon</p>
    <p>The data and Time just now is: @dateTime</p>
</div>
```

# Sending Data between Page Model and Razor page

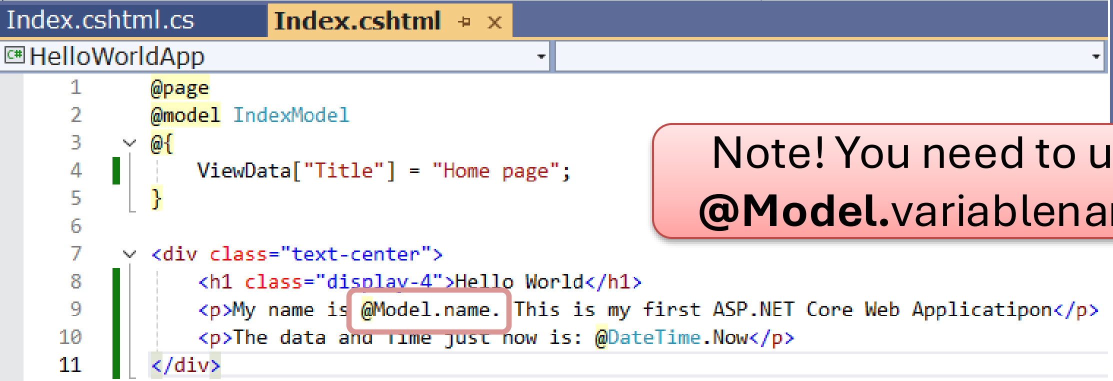
Typically, we need to send data between the Page Model (.cshtml.cs) and the Razor page (.cshtml).



Hello World

My name is Hans-Petter Halvorsen. This is my first ASP.NET Core Web Applicatipon

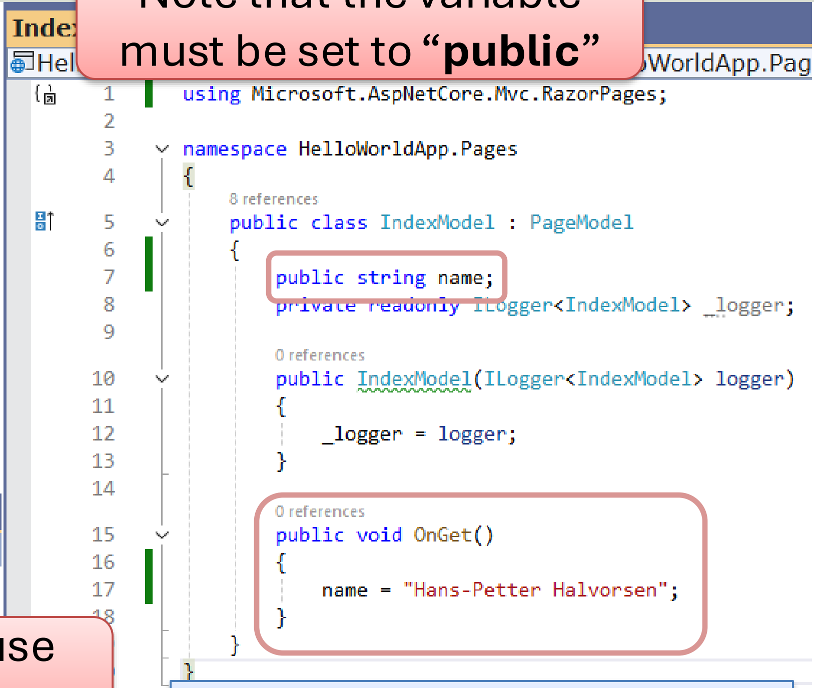
The data and Time just now is: 14/02/2025 14:04:46



```
@page
@model IndexModel
@{
    ViewData["Title"] = "Home page";
}

<div class="text-center">
    <h1 class="display-4">Hello World</h1>
    <p>My name is @Model.name. This is my first ASP.NET Core Web Applicatipon</p>
    <p>The data and Time just now is: @DateTime.Now</p>
</div>
```

Note! You need to use **@Model.**variablename

Note that the variable must be set to “**public**”



```
using Microsoft.AspNetCore.Mvc.RazorPages;

namespace HelloWorldApp.Pages
{
    8 references
    public class IndexModel : PageModel
    {
        public string name;
        private readonly ILogger<IndexModel> _logger;

        0 references
        public IndexModel(ILogger<IndexModel> logger)
        {
            _logger = logger;
        }

        0 references
        public void OnGet()
        {
            name = "Hans-Petter Halvorsen";
        }
    }
}
```

When a user request a Webpage, the code inside the **OnGet()** method are always executed on the Server before the code is sent to the Client.

# Resources and References

- Tutorial: Get started with Razor Pages in ASP.NET Core: https://learn.microsoft.com/en-us/aspnet/core/tutorials/razor-pages/razor-pages-start
- Learn Razor Pages: https://www.learnrazorpages.com
- Introduction to ASP.NET Core Razor Pages: https://www.csharp.com/article/introduction-to-asp-net-core-razor-pages/

# Hans-Petter Halvorsen

University of South-Eastern Norway

www.usn.no

E-mail: hans.p.halvorsen@usn.no

Web: https://www.halvorsen.blog