Basic Web

MVC, ASP.NET Core, Razor

SoftUni

SoftUni Team

Technical Trainers

Software University

Software University

https://softuni.bg

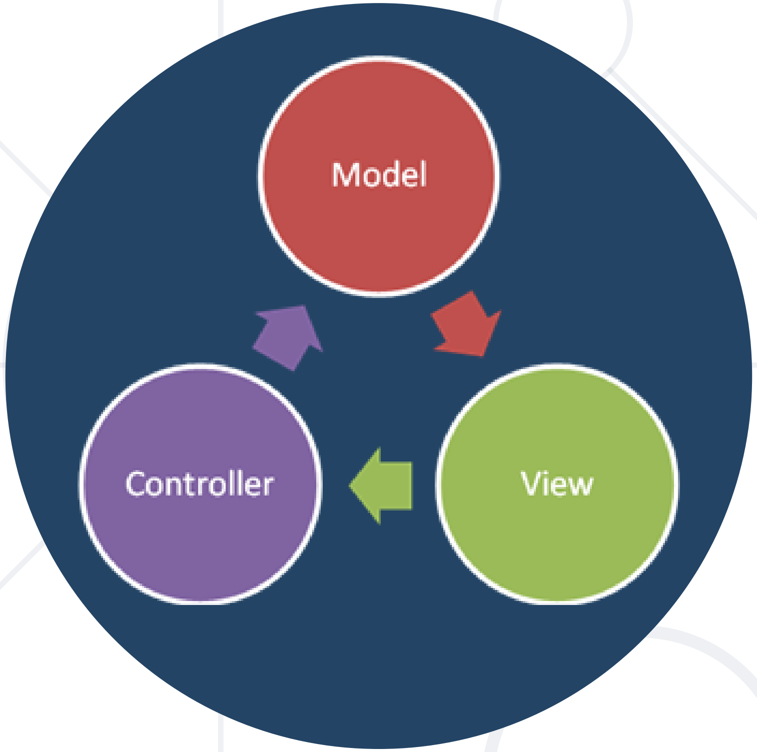
Table of Contents

1. Model-View Controller (MVC)
2. ASP.NET Core Framework
   - Introduction
   - Controllers and Routing
   - Views and Razor View Engine
   - Models in ASP.NET Core
   - Processing Requests
3. ASP.NET Core Web App – Live Demo

Have a Question?

sli.do

#fund-csharp



Model – View – Controller

MVC

MVC

- **Model-View-Controller** (MVC) is an architectural pattern
- Separates an application into **three main groups**
  - **Views, controller, models**
- Helps to achieve **separation of concerns**
- Delineation of responsibilities makes the application
  - Easy to read and understand the logic (better structure)
  - Easy to implement new functionalities and extensions
  - Easy to test and debug

Controller (Logic)

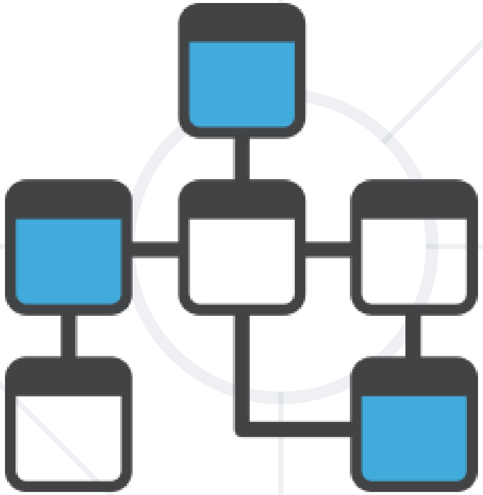
- The **core** MVC component – holds the **logic**
- Processes the requests and renders the views
- A set of classes that handles
  - **Handling data** submitted by the user
  - Overall application **flow**
  - Application-specific **logic** (business logic)
- Every controller has one or more **"actions"**

View (User Interface)

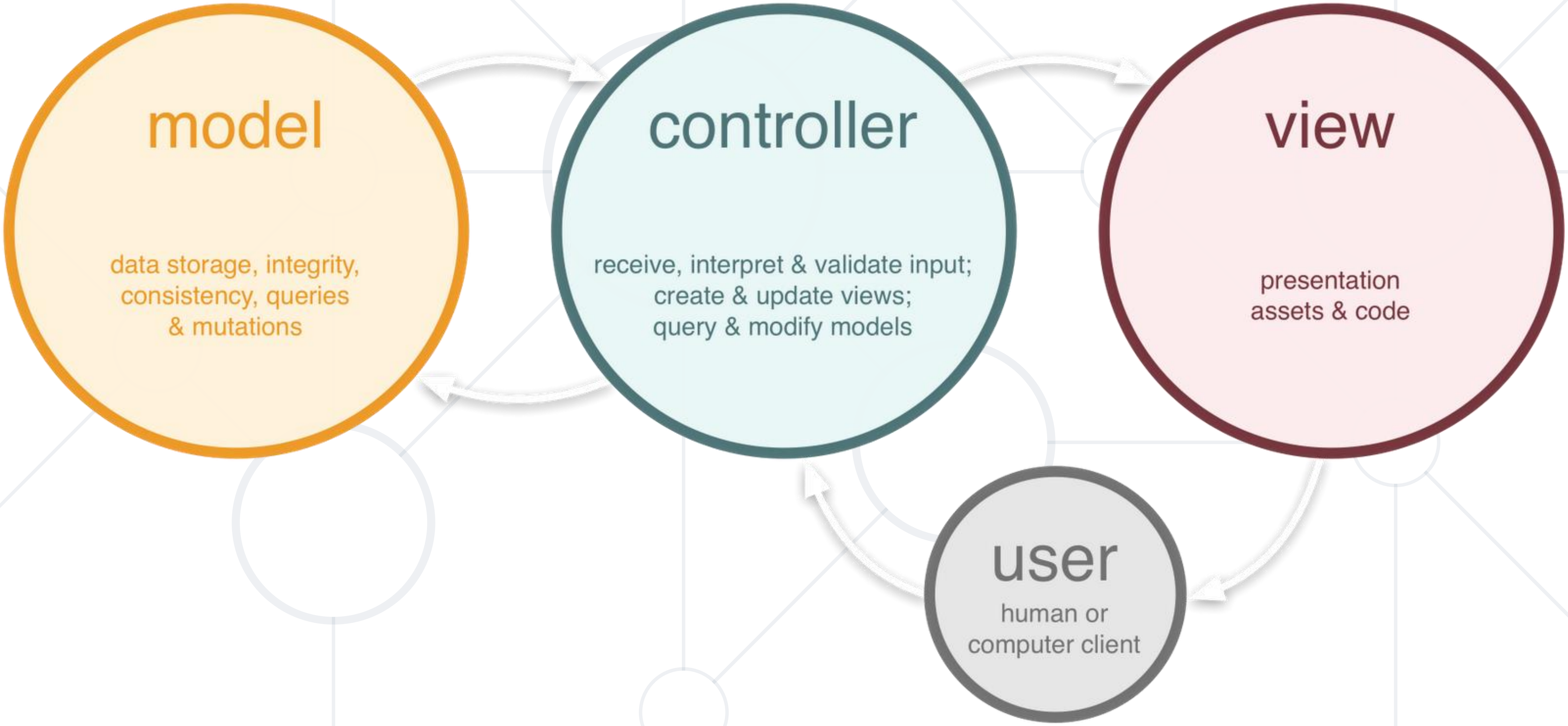
- Defines how the application's **user interface** (UI) will be displayed
- May support master views (**layouts**)
- May support sub-views (**partial views** or controls)
- May use **templates** to **dynamically generate** HTML

Model (Data)

- Set of **classes** that describes the **data** we are working with
- Rules for **how** the data can be **changed** and **manipulated**
- May contain **data validation rules**
- Often **encapsulates** data stored in a database



MVC


model
data storage, integrity, consistency, queries & mutations
controller
receive, interpret & validate input; create & update views; query & modify models
view
presentation assets & code
user
human or computer client

ASP.NET Core Framework

Web Application MVC Framework for C# and .NET

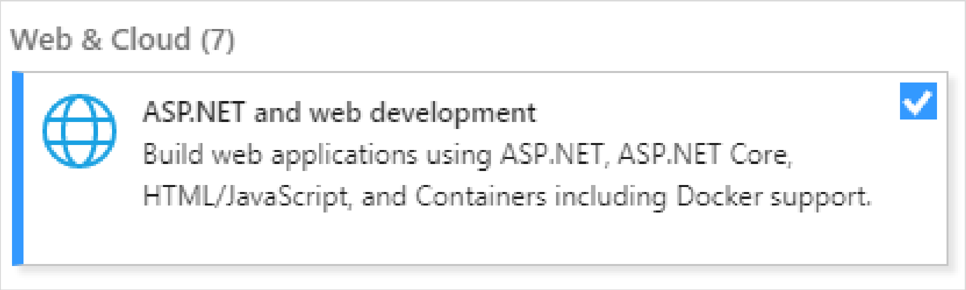
Web Framework

- **Framework** == set of resources and tools, used as base for building a software system
- **Web application framework** – provides a standard way to build and deploy **Web applications**
- Designed to support the development of
  - Web services
  - Web resources
  - Web APIs

ASP.NET Core MVC

- Lightweight, open-source and highly testable Web application framework
- Uses the **Model-View-Controller** design pattern
- Cross-platform – targeting the .NET Core platform
  - Runs on multiple operating systems

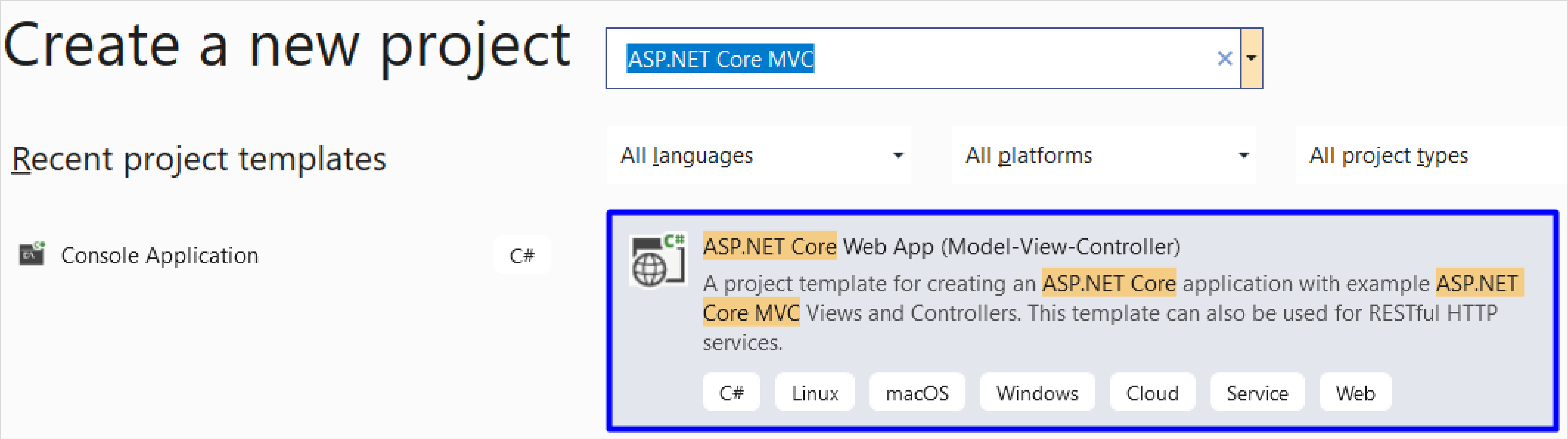
Create ASP.NET Core MVC App: Project Type



Web & Cloud (7)

☑ ASP.NET and web development
Build web applications using ASP.NET, ASP.NET Core, HTML/JavaScript, and Containers including Docker support.

Install this in **Visual Studio**!



Create a new project

ASP.NET Core MVC

Recent project templates

Console Application C#

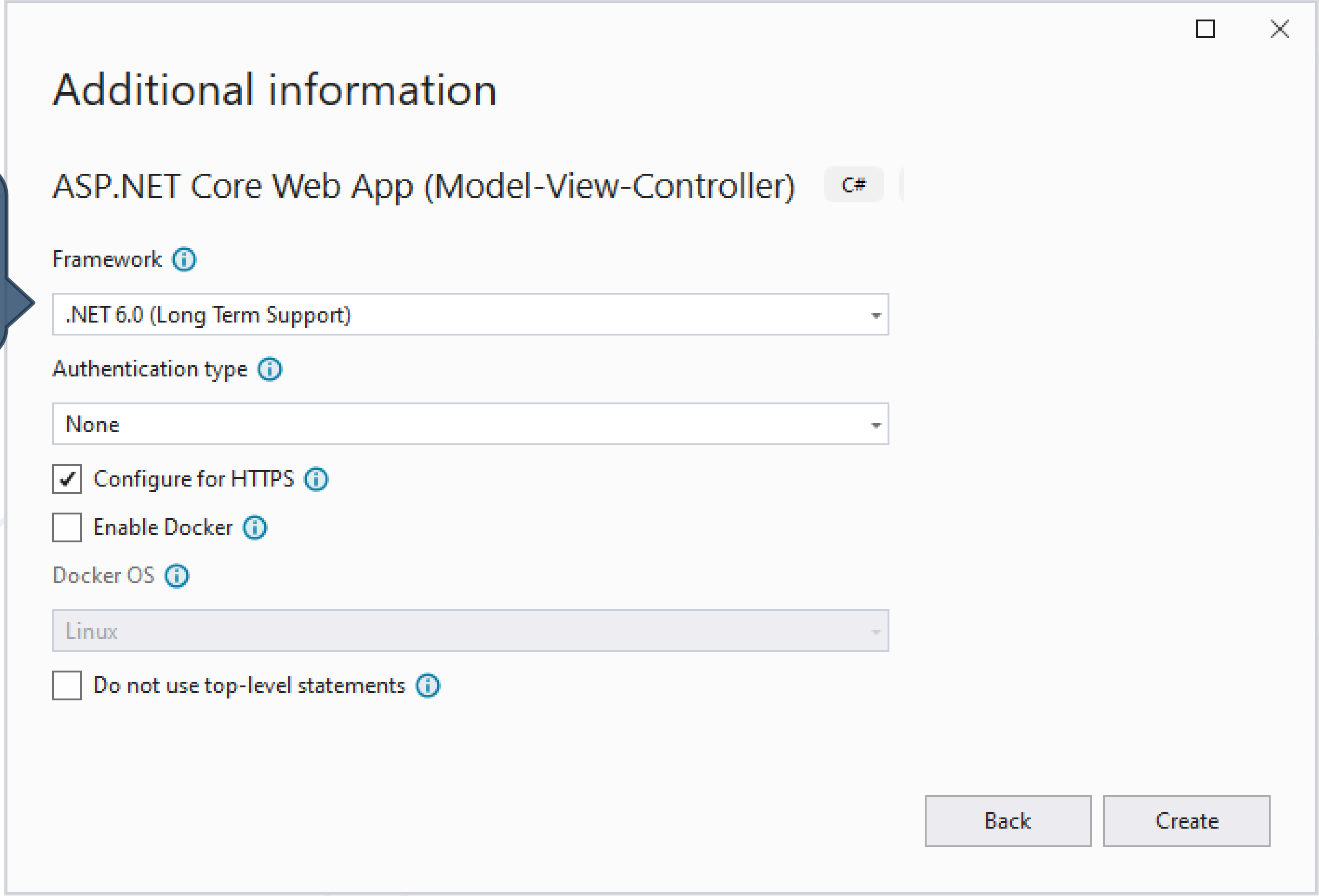
All languages | All platforms | All project types

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

Create ASP.NET MVC App: Choose Template

Choose the .NET version



Additional information

ASP.NET Core Web App (Model-View-Controller) C#

Framework: .NET 6.0 (Long Term Support)

Authentication type: None

☑ Configure for HTTPS

□ Enable Docker

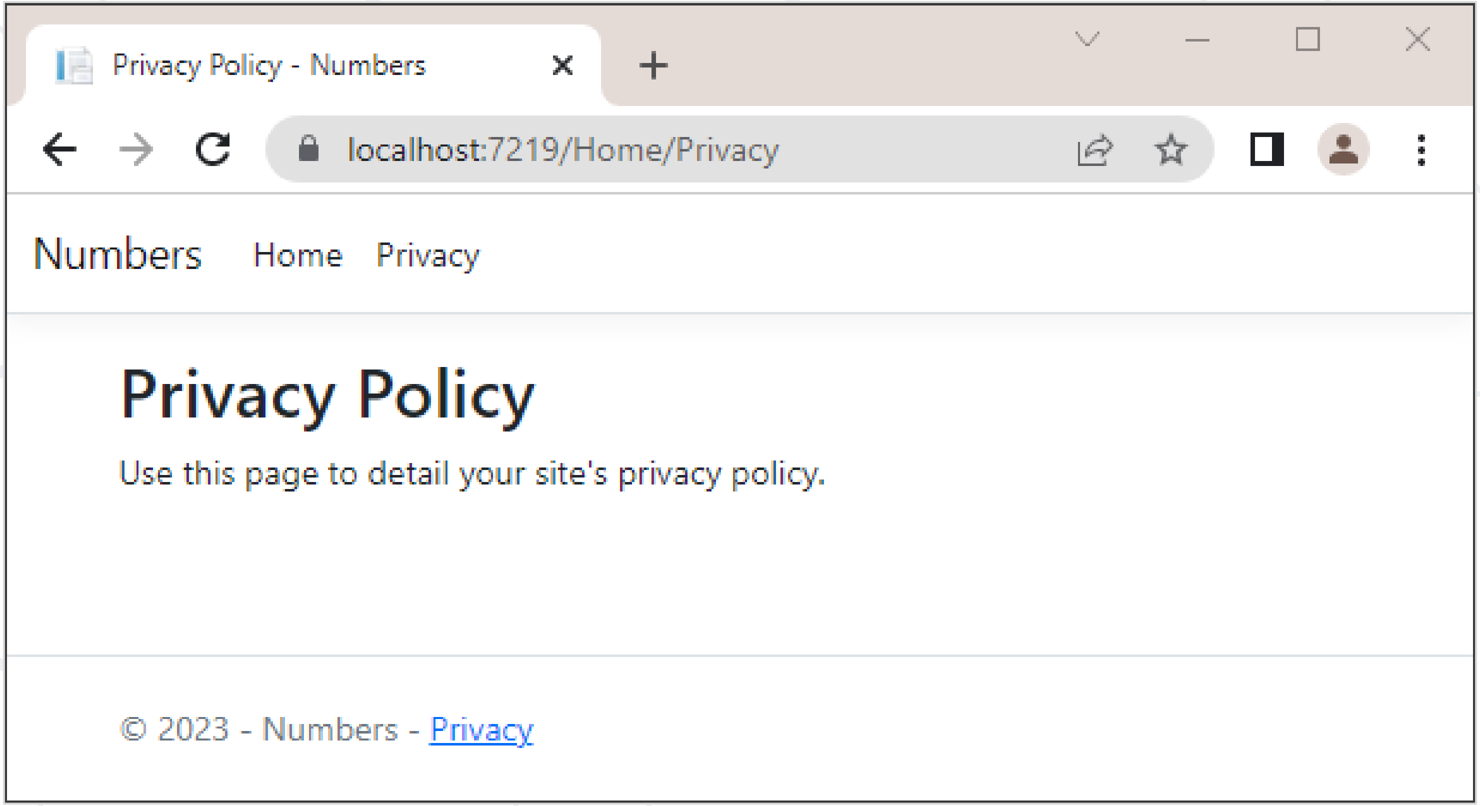
Docker OS: Linux

□ Do not use top-level statements

The "Privacy" Page in the Browser

- Run the app, by pressing **[Ctrl + F5]**
  - Open the "**Privacy**" page on https://localhost:7219/Home/Privacy

The **port number** is auto-generated



Privacy Policy

Use this page to detail your site's privacy policy.

© 2023 - Numbers - Privacy

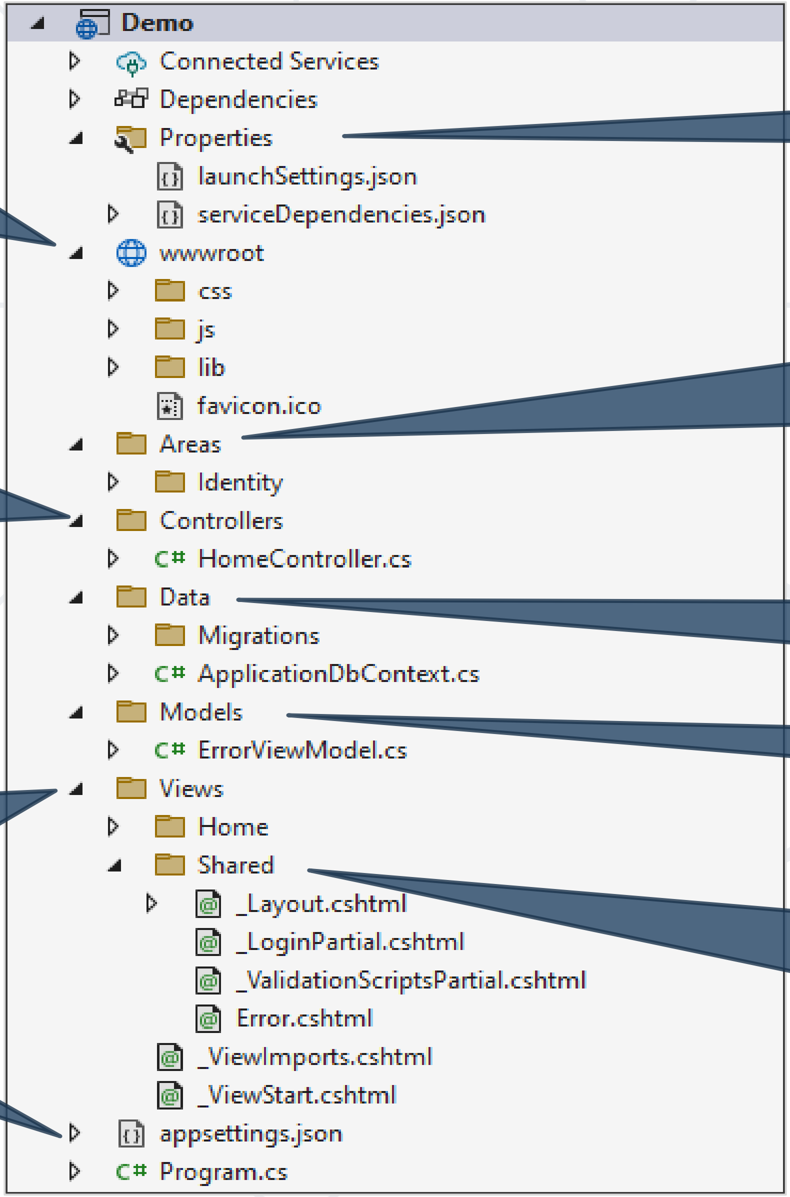
MVC App: What's Inside?

Static files: CSS styles images, fonts, …

Controller classes holding actions

Views: HTML templates for the pages

App start files



```
Demo
  Connected Services
  Dependencies
  Properties
    launchSettings.json
    serviceDependencies.json
  wwwroot
    css
    js
    lib
    favicon.ico
  Areas
    Identity
  Controllers
    HomeController.cs
  Data
    Migrations
    ApplicationDbContext.cs
  Models
    ErrorViewModel.cs
  Views
    Home
    Shared
      _Layout.cshtml
      _LoginPartial.cshtml
      _ValidationScriptsPartial.cshtml
      Error.cshtml
    _ViewImports.cshtml
    _ViewStart.cshtml
  appsettings.json
  Program.cs
```

NuGet packages

Areas: physically partition a web app in separate units

Data: EF models + DB context + migrations

Models: view models

Shared views: layout for all pages + partial views

Controllers in ASP.NET Core

- MVC controllers hold logic to process user interactions
- The URL **/Home/Privacy** invokes **HomeController** → **Privacy()**

\Controllers\HomeController.cs

```
public class HomeController : Controller
{
  public IActionResult Privacy()
  {
     return View();
  }
}
```

Controller's methods are called actions

Renders Views\Home\Index.cshtml

ASP.NET Core App Routing

- Routing is configured within Configure Method in the **Program** Class

```
app.MapControllerRoute(
    name: "default",
    pattern: "{controller=Home}/{action=Index}/{id?}");
});
```

id route parameter is optional

Route values are determined by splitting the URL into segments

- Template Matches URL path like
  - **/Orders/Details/17**
  - **/Orders/All**
- Also match the URL path **"/"**, **{controller}** and **{action}** parameters have default values **Home** and **Index**

Views in ASP.NET Core

- Views render the HTML code for the **invoked action**
- Views combine **HTML** and **C# code**
- ASP.NET MVC uses **Razor** view engine
  - **Markup syntax** for embedding server-based code into webpages
  - Syntax consists of **Razor markup**, **C#**, and **HTML**
  - Files generally have a `.cshtml` file extension
- By convention **Action** and **View** name are identical

The "Razor" Syntax (Templating Engine)

```
\Views\Home\Index.cshtml
```

```
@{
   ViewData["Title] = "Home Page";
}
<h2>@title</h2>
<h3>@ViewBag.Message</h3>
<p>Use this page to detail your site's info.</p>
```

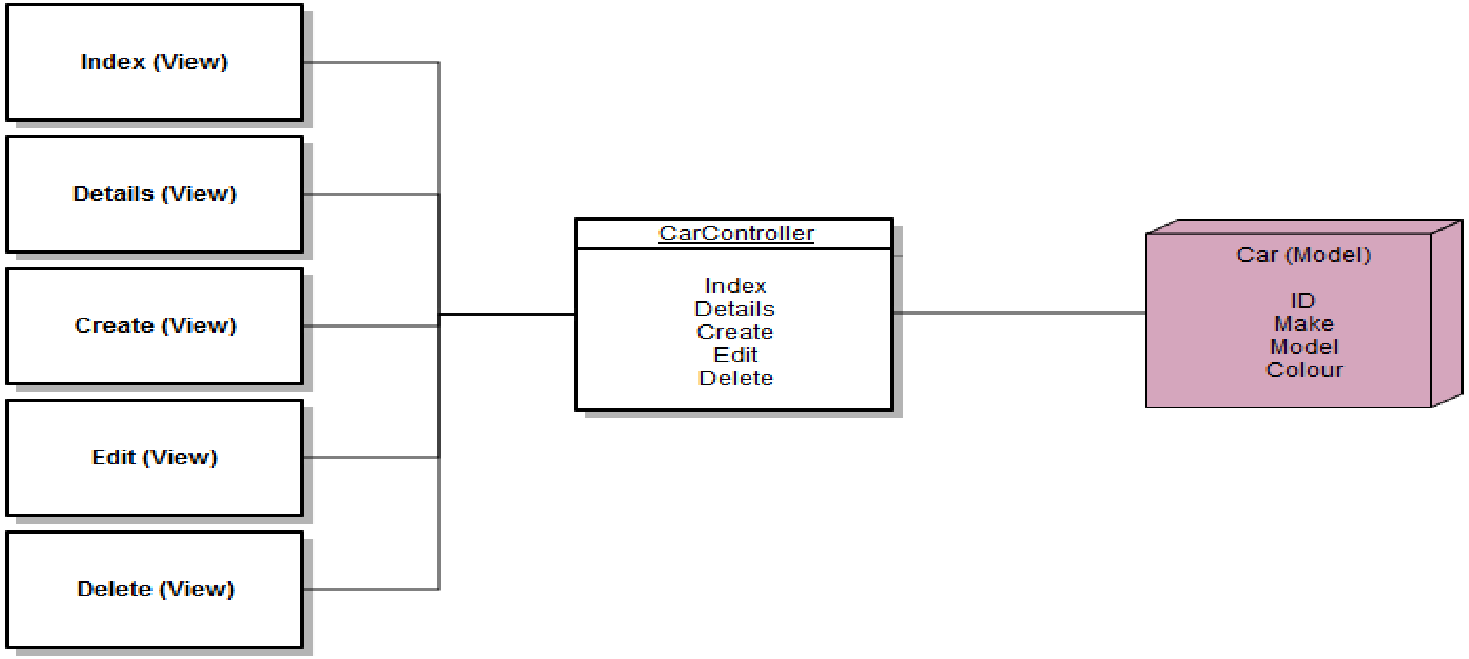
@ { … } inserts C# code block

@Something
prints a C# variable

Everything else
is HTML code

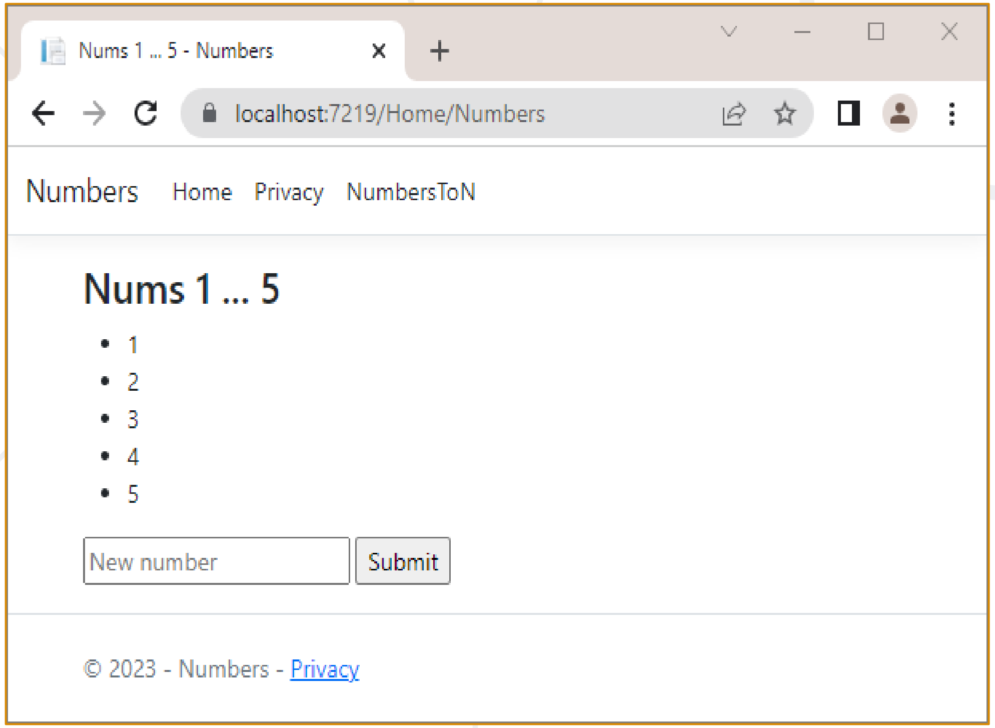
Models in ASP.NET Core

- Represent the state of the application
- May be used by controllers to pass data to Views
- Determine how the data will be stored



Example: Number Generator

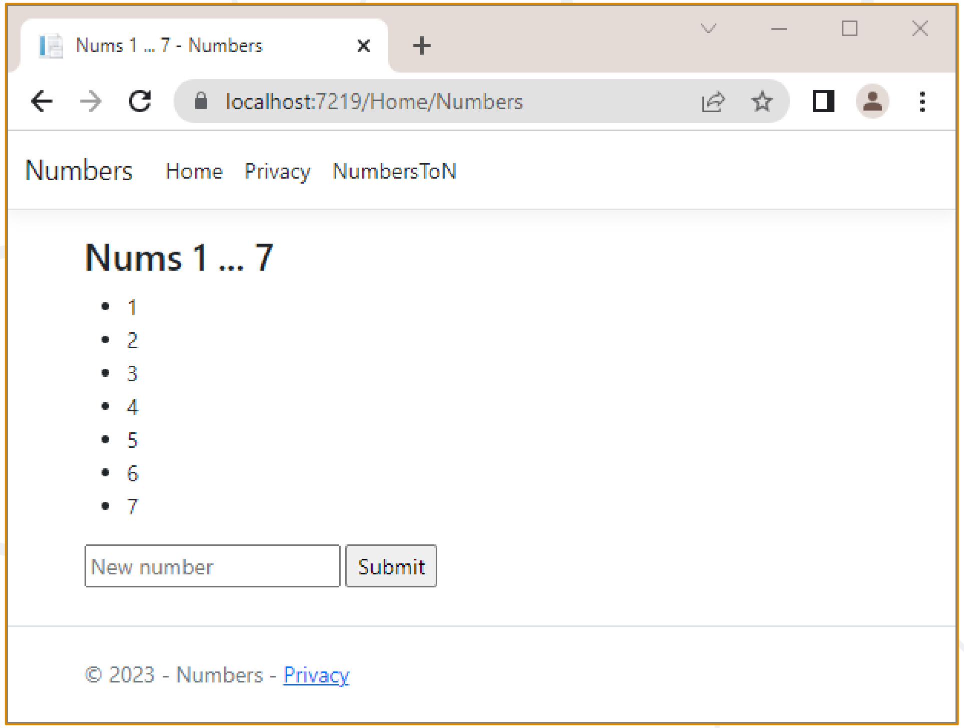
- Let's create an application that displays the numbers in range [1…n], where n will be given by the user via a form input



Nums 1 ... 5

- 1
- 2
- 3
- 4
- 5

© 2023 - Numbers - Privacy



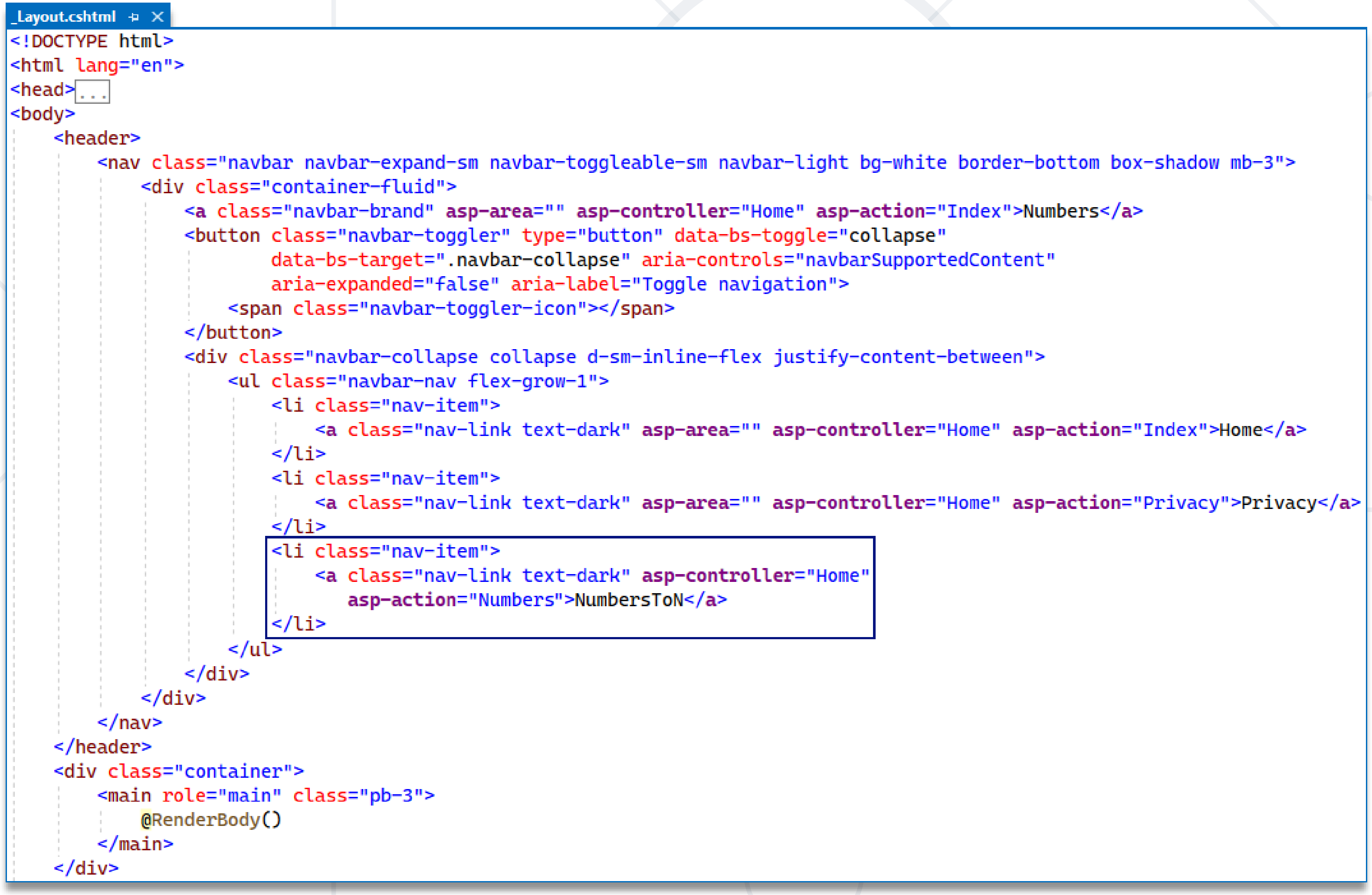
Nums 1 ... 7

- 1
- 2
- 3
- 4
- 5
- 6
- 7

© 2023 - Numbers - Privacy

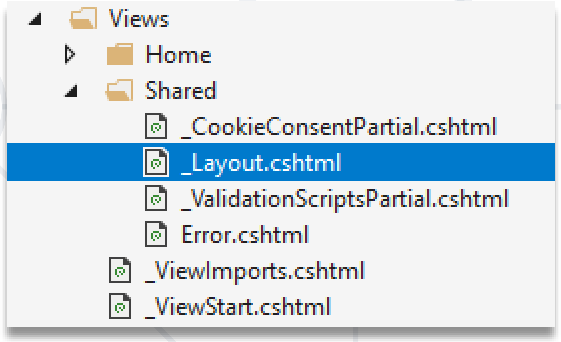
Adding Navigation

- Change the `_Layout.cshtml` file in your project



_Layout.cshtml

```
<!DOCTYPE html>
<html lang="en">
<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">Numbers</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>
                        <li class="nav-item">
                            <a class="nav-link text-dark" asp-controller="Home"
                               asp-action="Numbers">NumbersToN</a>
                        </li>
                    </ul>
                </div>
            </div>
        </nav>
    </header>
    <div class="container">
        <main role="main" class="pb-3">
            @RenderBody()
        </main>
    </div>
```



Process GET Request

- We need a **functionality** to handle our interaction
- Create new action method **Numbers** in **HomeController.cs**
- The **id** from **URL** will be passed as **parameter**

IActionResult represents the view result

```
public IActionResult Numbers(int id)
{
    int numbersRange = id;
    ViewBag.numbersRange = numbersRange;
    return this.View();
}
```

ViewBag is used to pass data to the **View**

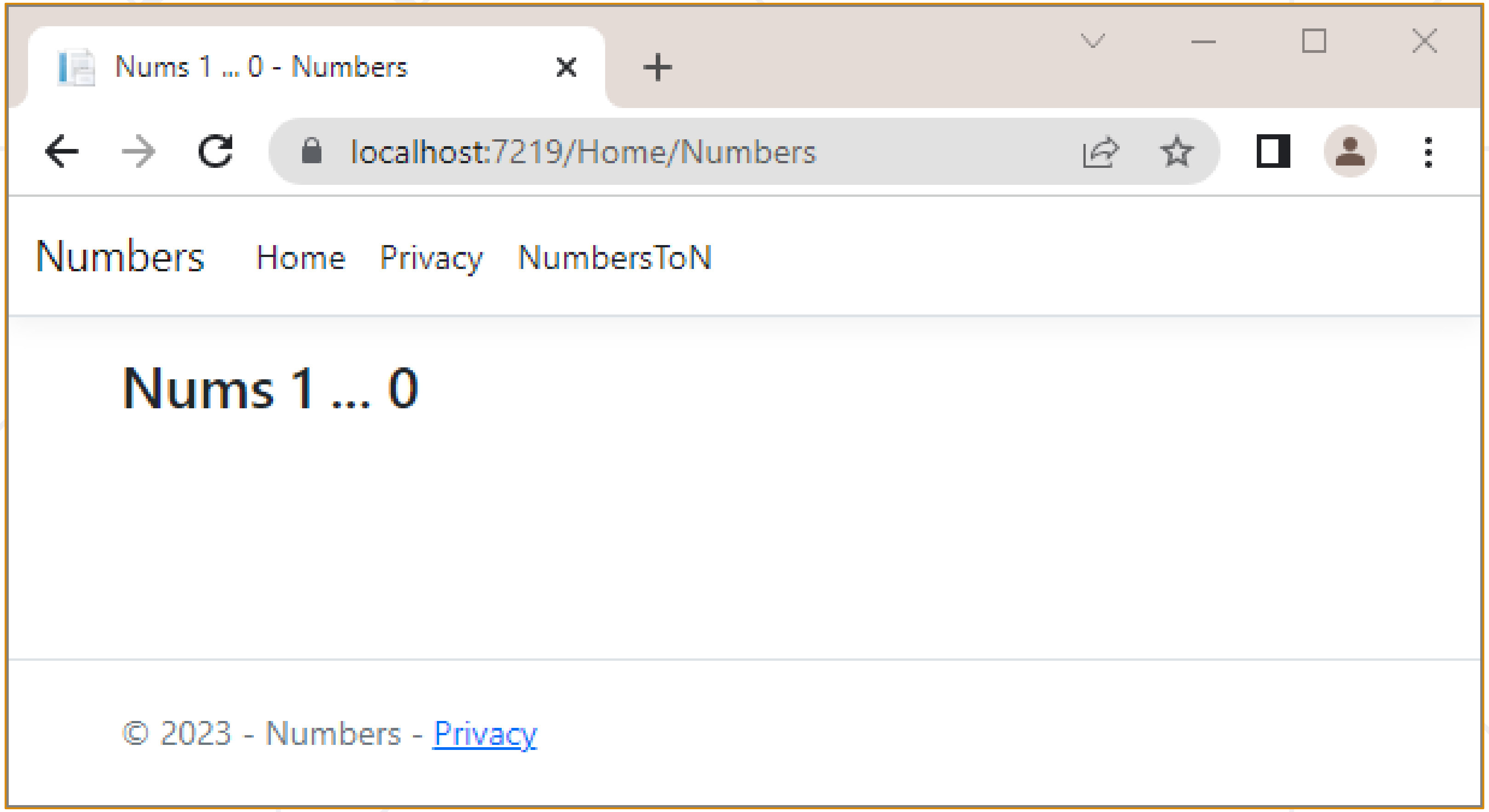
Create View

- Create a new Razor View **Numbers.cshtml** in **Views\Home** folder
- Use **Razor View Engine** to generate appropriate title and render unordered HTML list of numbers

```
@{
  ViewBag.Title ="Nums 1 .. " + ViewBag.Count;
}
<h2>@ViewBag.Title</h2>
<ul>
     @for (int i = 1; i <= ViewBag.Count; i++)
     {
        <li>@i</li>
     }
</ul>
```

Nums 1 … 10

- After clicking on **`NumbersToN`** we should be able to see this



Nums 1 … 0

© 2023 - Numbers - Privacy

Process POST Request (1)

- Add **Form** at the end of the **Numbers.cshtml**, so we can post the number range we want to generate

We must specify the **request method**

The **name** must be exactly the same as the **action parameter name**

```
<form method="POST">
    <input type="text" placeholder="New number" name="number" />
    <button type="submit">Submit</button>
</form>
```

Process POST Request (2)

- Create new **Action** method in **HomeController**

If we want to procces POST request, we must use **[HttpPost] attribute**

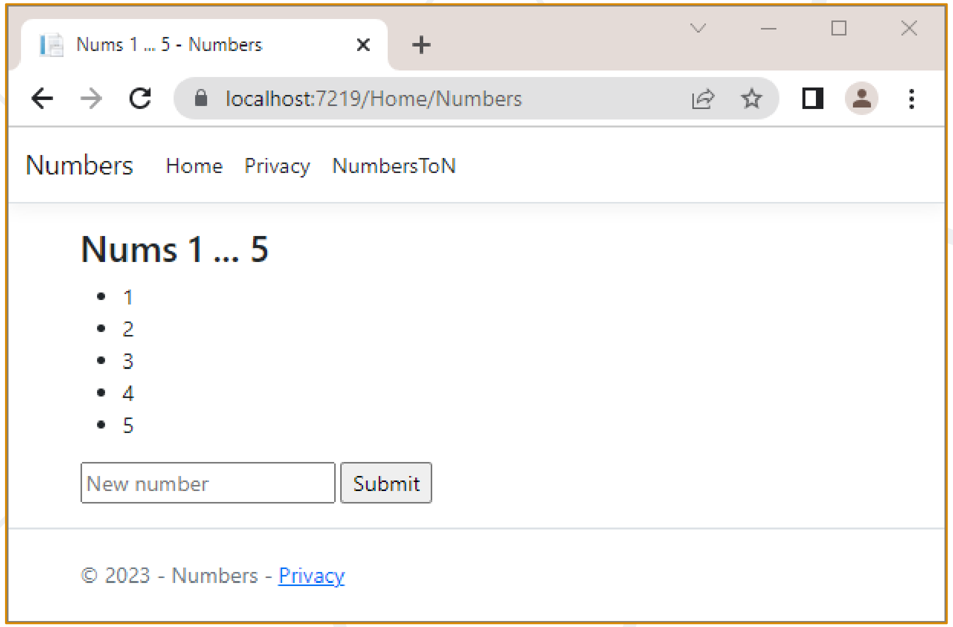
Both actions have identical names, but parameter types are different

```
[HttpPost]
public IActionResult Numbers(string number)
{
   int numbersRange = int.Parse(number);
   ViewBag.numbersRange = numbersRange;
   return this.View();
}
```

Use the same **ViewBag** property name to pass the data

Number Generator

- Now the app should be able to generate custom range



Nums 1 ... 5 - Numbers

localhost:7219/Home/Numbers

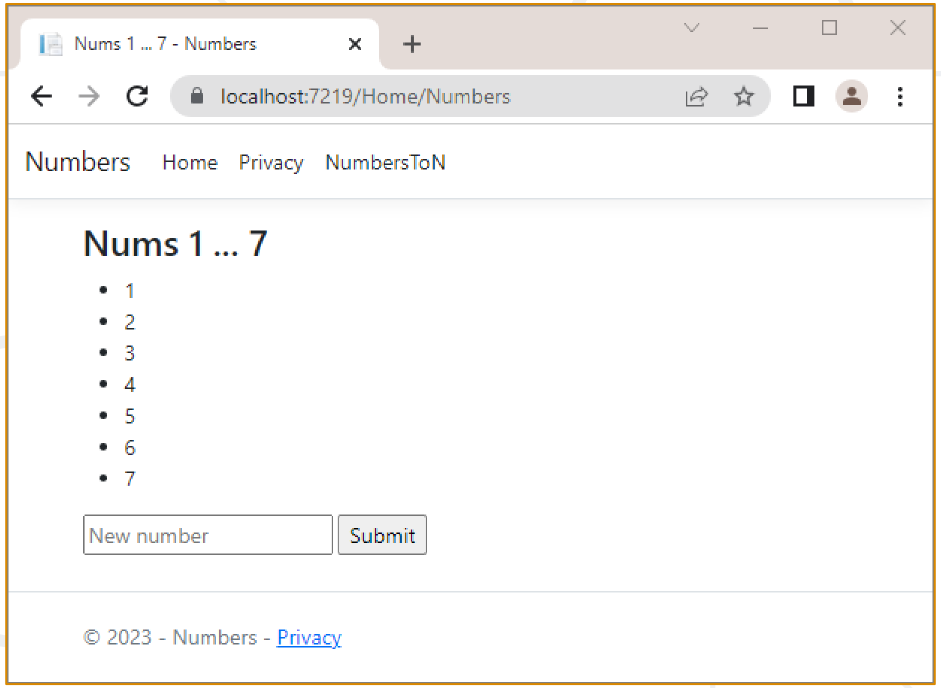
Numbers Home Privacy NumbersToN

Nums 1 ... 5

- 1
- 2
- 3
- 4
- 5

New number | Submit

© 2023 - Numbers - Privacy



Nums 1 ... 7 - Numbers

localhost:7219/Home/Numbers

Numbers Home Privacy NumbersToN

Nums 1 ... 7

- 1
- 2
- 3
- 4
- 5
- 6
- 7

New number | Submit

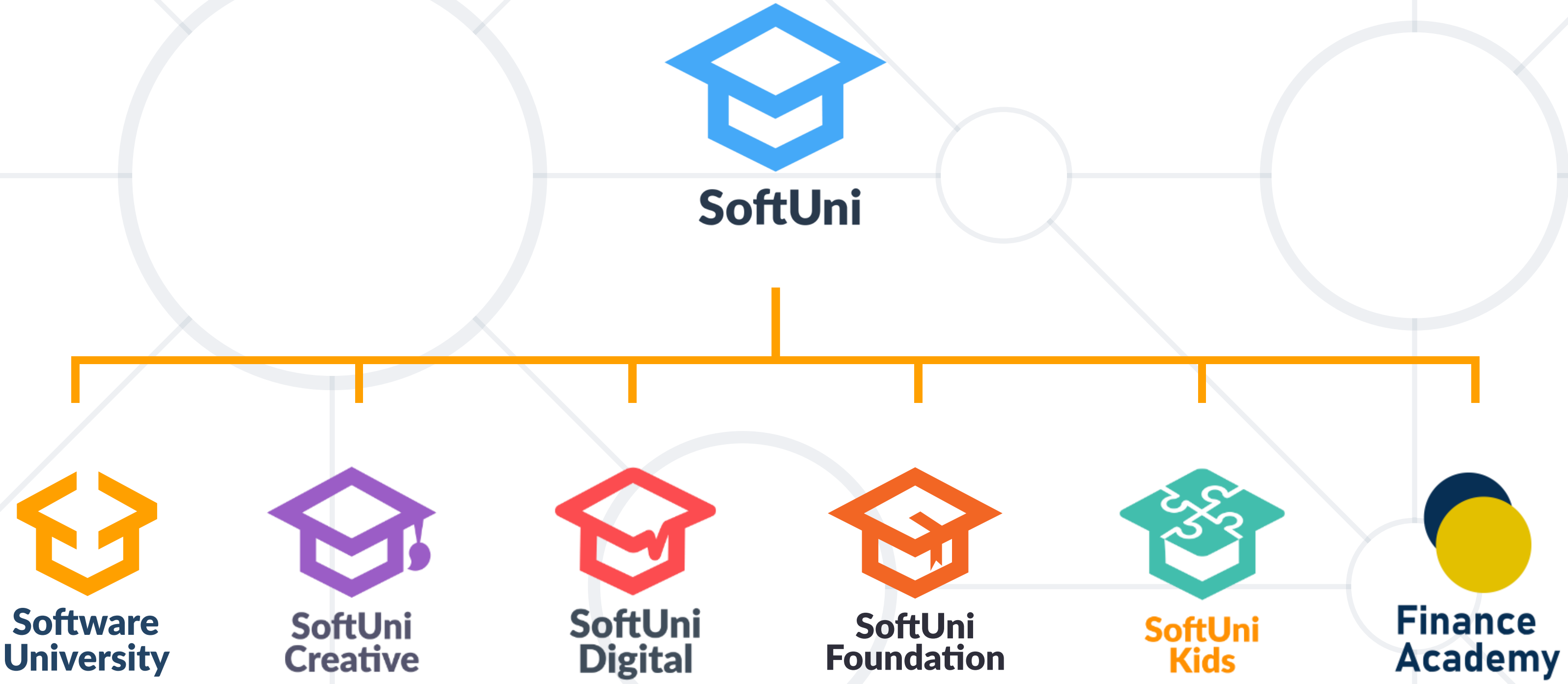
© 2023 - Numbers - Privacy

Summary

- **Framework** is a set of resources and tools for app building
- **MVC** is an architectural pattern
- **Views** and **Controllers** functionalities
- **View Engine** (like Razor) is used to create dynamic Web pages

Software University

Questions?



SoftUni

Software University

SoftUni Creative

SoftUni Digital

SoftUni Foundation

SoftUni Kids

Finance Academy

SoftUni Diamond Partners

SUPER HOSTING .BG

Coca-Cola HBC Bulgaria

Flutter™ International

INDEAVR
Serving the high achievers

AMBITIONED

DRAFT KINGS

SOFTWARE GROUP

BOSCH

Postbank
Решения за твоето утре

PHAR VISION

SmartIT

DXC TECHNOLOGY

createX

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg, about.softuni.bg
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University Forums
  - forum.softuni.bg

License