



# Implement Pagination in .NET Core MVC Application with Custom HtmlHelpers

**mural3**

14 Feb 2022 CPOL

How to build custom HtmlHelpers to provide pagination in .NET Core MVC app

In this article, we achieve pagination in a .NET Core MVC application by simply creating a custom HtmlHelper. To keep it simple, we just use numbers to represent data.

## Introduction

In this article, let us try to build custom **HtmlHelpers** to provide pagination in .NET Core MVC application.

This article expects the user to have basic knowledge on **HtmlHelpers**.

For those who do not have an idea of HTML helpers, here is a brief explanation.

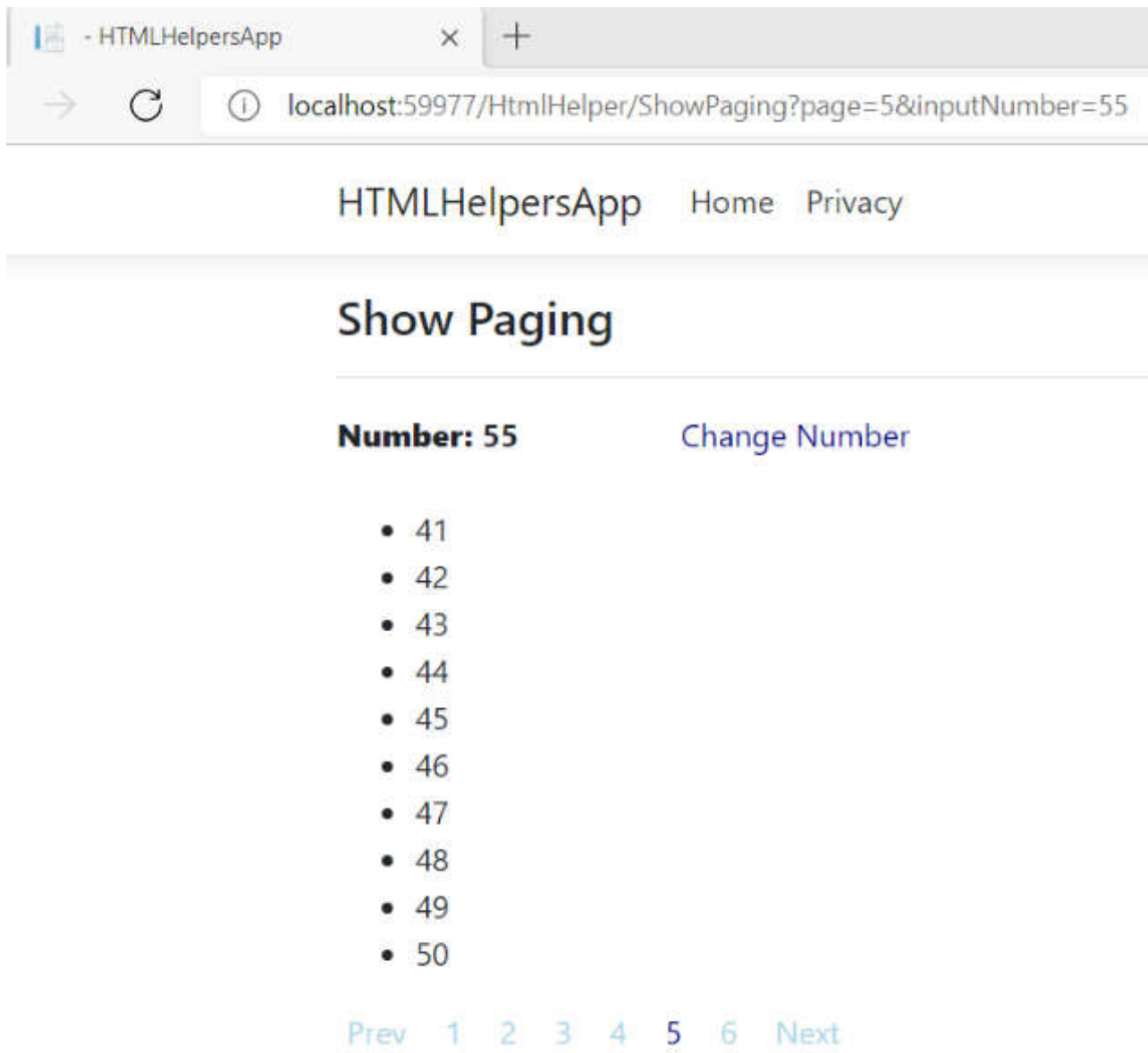
- HTML helpers make it easy and fast for developers to create HTML pages.
- In most cases, an HTML helper is just a method that returns a **string**.
- MVC comes with built in HTML helpers like `@Html.TextBox()`, `@Html.CheckBox()`, `@Html.Label`, etc.
- **Htmlhelpers** render HTML controls in razor view. For example, `@Html.TextBox()` renders `<input type="textbox">` control, `@Html.CheckBox()` renders `<input type="checkbox">` control, etc.

Please go through the [Microsoft documentation on HTML helpers](#).

## Background

In a web application, if huge number of records are to be displayed, it is required to provide pagination. In this article, we achieve pagination in a .NET Core MVC application by simply creating

a custom **HtmlHelper**. To keep it simple, we just use numbers to represent data.



Say we need to display 55 records in multiple pages with 10 items on each page. We can display all the items in 6 pages as shown above. In real time, the data could be a number of records that are fetched from a database, file, etc.

Let us see how it can be done.

- Open Visual Studio 2019 > Create .NET Core MVC Application as shown below:

## Create a new project

### Recent project templates

ASP.NET Core Web App

C#

Search for templates (Alt+S)

All languages

All platforms

All project types

**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

**Blazor Server App**

A project template for creating a Blazor server app that runs server-side inside an ASP.NET Core app and handles user interactions over a SignalR connection. This template can be used for web apps with rich dynamic user interfaces (UIs).

C#

Linux

macOS

Windows

Cloud

Web

**ASP.NET Core Web API**

A project template for creating an ASP.NET Core application with an example Controller for a RESTful HTTP service. This template can also be used for ASP.NET Core MVC Views and Controllers.

C#

Linux

macOS

Windows

Cloud

Service

Web

Back

Next

- Name your project as **HTMLHelpersApp**.

## Configure your new project

ASP.NET Core Web App

C#

Linux

macOS

Windows

Cloud

Service

Web

Project name

HTMLHelpersApp

Location

C:\Users\Kmdhar\source\repos

Solution name ⓘ

HTMLHelpersApp

☐ Place solution and project in the same directory

Back

Next

- Select .NET Framework Version:

## Additional information

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

Target Framework [i](#)

.NET Core 3.1 (Long-term support)

Authentication Type [i](#)

None

☐ Configure for HTTPS [i](#)

☐ Enable Docker [i](#)

Docker OS [i](#)

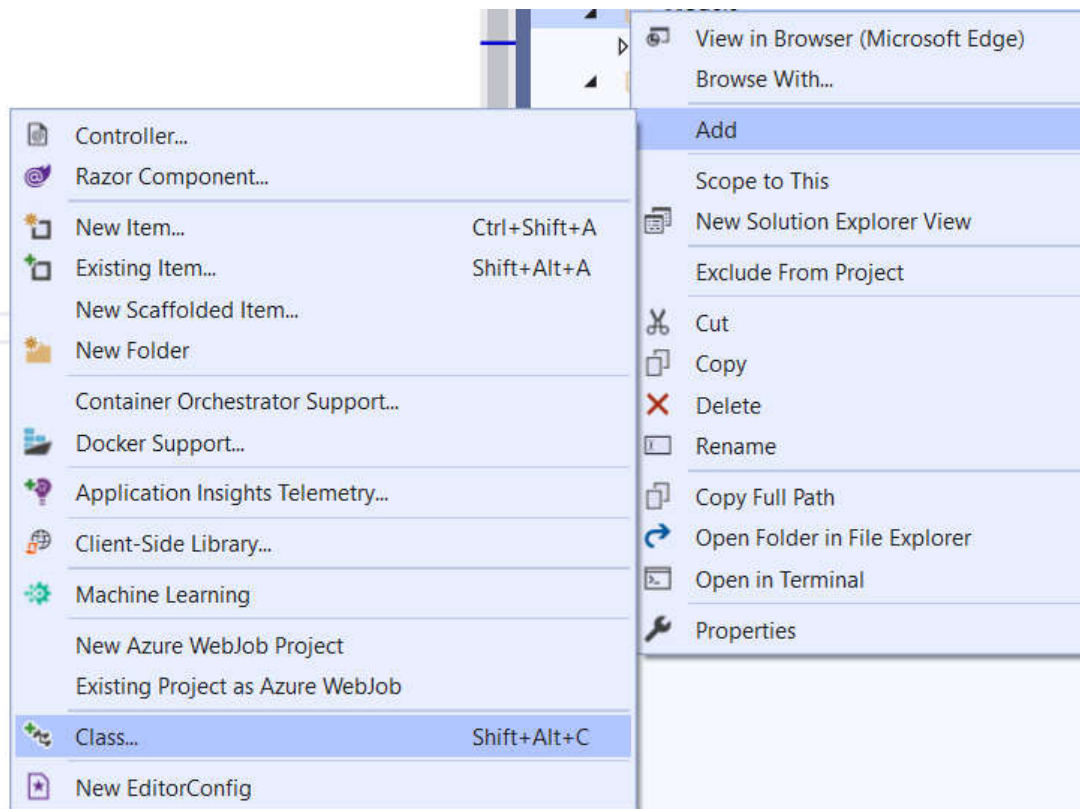
Linux

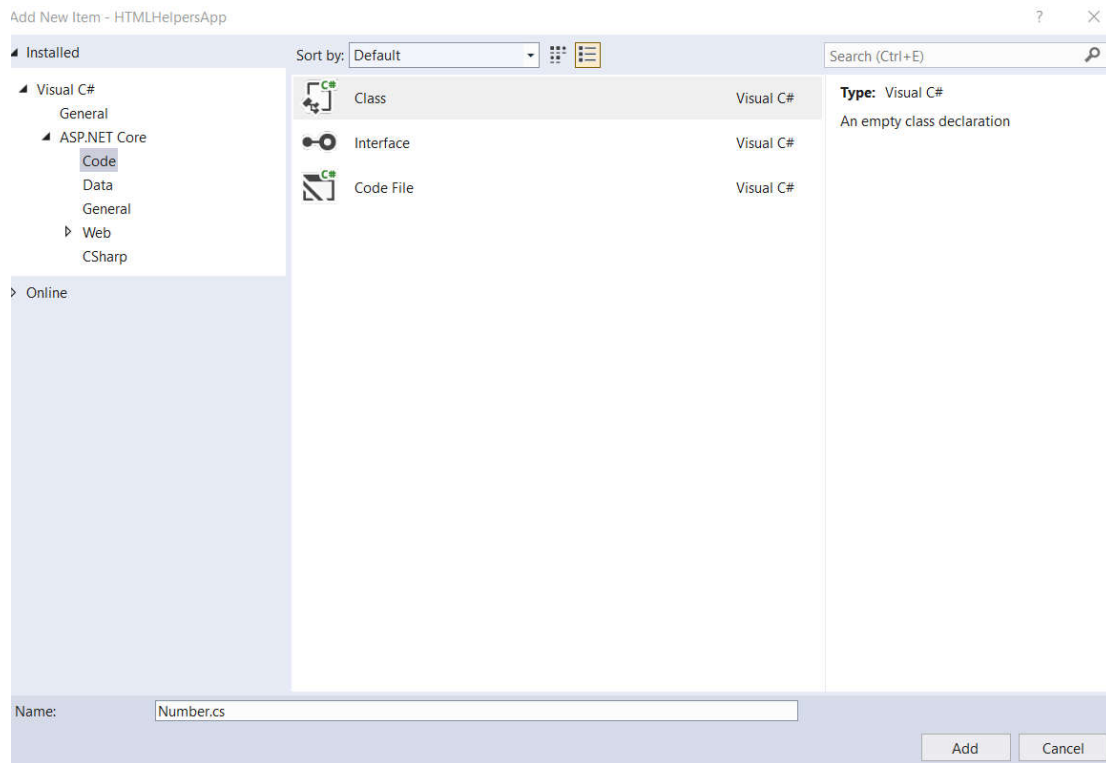
☐ Enable Razor runtime compilation [i](#)

Back

Create

- First, let us create the required models and helper files.
  - Create a new model '**Number**'.
  - Right click on *Model* folder and add '**Number**' class:





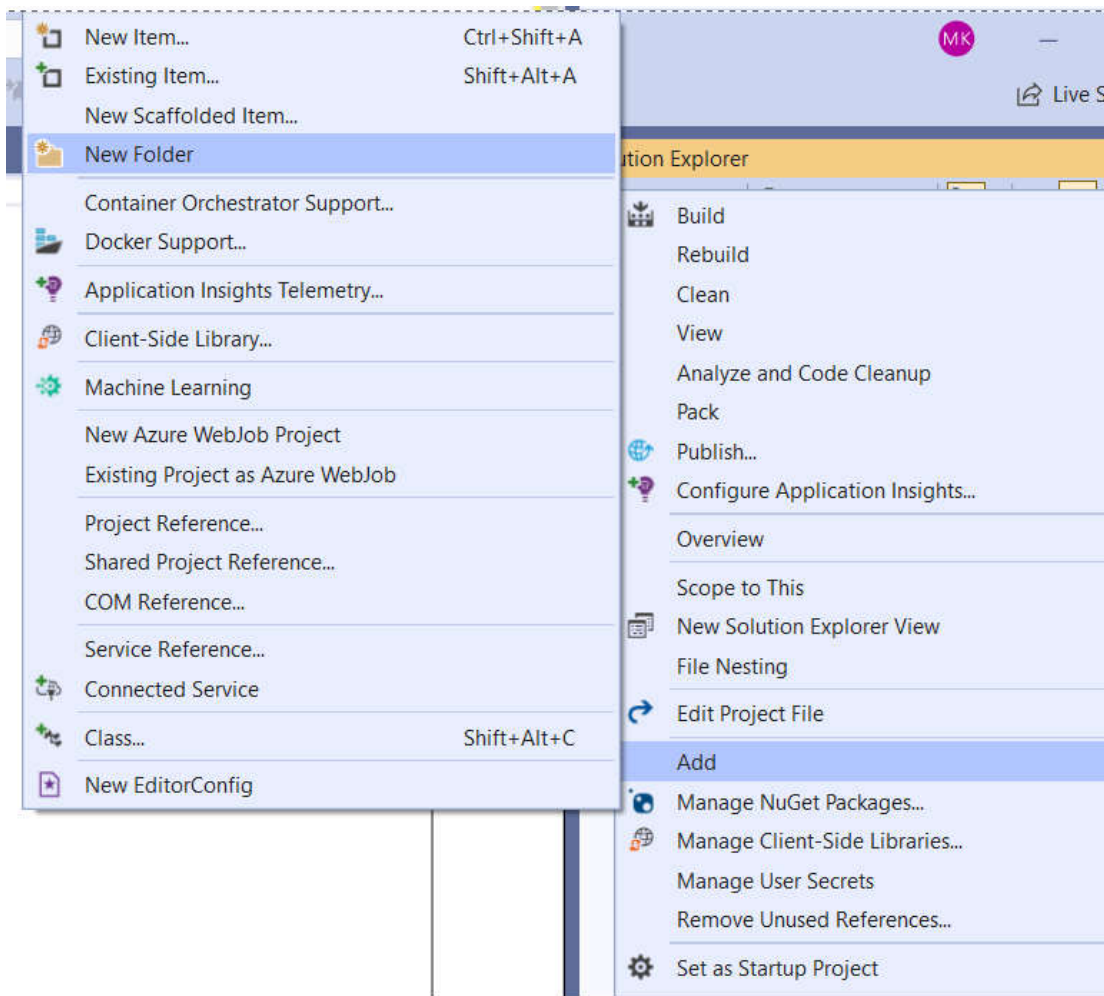
- Add code in *Number.cs*. This model captures the user input. It has only one property '*InputNumber*'.

C#

```
using System;
using System.ComponentModel.DataAnnotations;

namespace HTMLHelpersApp.Models
{
    public class Number
    {
        //validation for required, only numbers, allowed range-1 to 500
        [Required(ErrorMessage = "Value is Required!.
            Please enter value between 1 and 500.")]
        [RegularExpression(@"^\d+$",
            ErrorMessage = "Only numbers are allowed.
            Please enter value between 1 and 500.")]
        [Range(1, 500, ErrorMessage = "Please enter value between 1 and 500.")]
        public int InputNumber = 1;
    }
}
```

- Now let us add a common class *PageInfo.cs*. Create new folder *Common* and add *PageInfo.cs* class.
  - Right click on the project folder and add a new folder:



- Add code in *PageInfo.cs*
  - Page Start indicates the first item in the current page.
  - Page End indicates the last item in the current page.
  - Items per page indicated the number of items to be displayed in a page.
  - Last Page indicates the number of pages / last page number.
  - Total Items indicate the total number of items.

Based on total items and items per page, the total number of pages, the first item and last items in a page are calculated.

C#

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace HTMLHelpersApp.Common
{
    public class PageInfo
    {
        public int TotalItems { get; set; }
        public int ItemsPerPage { get; set; }
        public int CurrentPage { get; set; }

        public PageInfo()
        {
            CurrentPage = 1;
        }
    }
}
```

```

    }
    //starting item number in the page
    public int PageStart
    {
        get { return ((CurrentPage - 1) * ItemsPerPage + 1); }
    }
    //Last item number in the page
    public int PageEnd
    {
        get
        {
            int currentTotal = (CurrentPage - 1) * ItemsPerPage + ItemsPerPage;
            return (currentTotal < TotalItems ? currentTotal : TotalItems);
        }
    }
    public int LastPage
    {
        get { return (int)Math.Ceiling((decimal)TotalItems / ItemsPerPage); }
    }
}

```

- Now we come to the most important part, that is, creating the custom **Htmlhelpers**.
  - Create custom **Htmlhelper PageLinks** which render the page numbers, previous and next links.
  - Add a new class '*PagingHtmlHelpers.cs*' in '*Common*' folder.
- Add code in '*PagingHtmlHelpers.cs*':
  - Extend **HtmlHelper** class and add new functionality to add Page links:
    - **public static IHtmlContent PageLinks(this IHtmlHelper htmlHelper, PageInfo pageInfo, Func<int, string> PageUrl)**
    - Takes two parameters:
      - **pageInfo**
        - to add the page numbers
        - delegate to a function which takes an integer and string as parameters
          - to add the parameters required in controller action method
    - Use tag builders to create anchor tags:
      - **TagBuilder tag = new TagBuilder("a");**
      - Add attributes:
        - **tag.MergeAttribute("href", hrefValue);**
        - **tag.InnerHtml.Append(" " + innerHtml + " ");**
        - Styles also can be applied as attributes.

C#

```

using Microsoft.AspNetCore.Html;
using Microsoft.AspNetCore.Mvc.Rendering;
using System;
using System.Text;

```



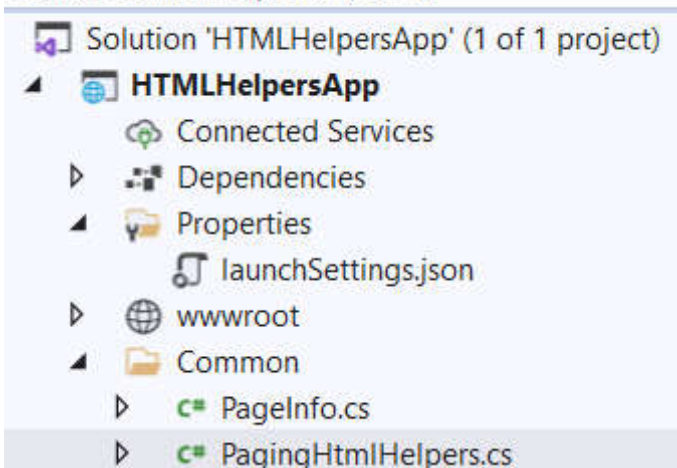
```

namespace HTMLHelpersApp.Common
{
    public static class PagingHtmlHelpers
    {
        public static IHtmlContent PageLinks
        (this IHtmlHelper htmlHelper, PageInfo pageInfo, Func<int, string> PageUrl)
        {
            StringBuilder pagingTags = new StringBuilder();
            //Prev Page
            if (pageInfo.CurrentPage > 1)
            {
                pagingTags.Append(GetTagString
                    ("Prev", PageUrl(pageInfo.CurrentPage - 1)));
            }
            //Page Numbers
            for (int i = 1; i <= pageInfo.LastPage; i++)
            {
                pagingTags.Append(GetTagString(i.ToString(), PageUrl(i)));
            }
            //Next Page
            if (pageInfo.CurrentPage < pageInfo.LastPage)
            {
                pagingTags.Append(GetTagString
                    ("Next", PageUrl(pageInfo.CurrentPage + 1)));
            }
            //paging tags
            return new HtmlString(pagingTags.ToString());
        }

        private static string GetTagString(string innerHtml, string hrefValue)
        {
            TagBuilder tag = new TagBuilder("a"); // Construct an <a> tag
            tag.MergeAttribute("class", "anchorstyle");
            tag.MergeAttribute("href", hrefValue);
            tag.InnerHtml.Append(" " + innerHtml + " ");
            using (var sw = new System.IO.StringWriter())
            {
                tag.WriteTo(sw, System.Text.Encodings.Web.HtmlEncoder.Default);
                return sw.ToString();
            }
        }
    }
}

```

Search Solution Explorer (Ctrl+;)





- Add a new class '*ShowPaging.cs*' in '*Models*' folder.
  - **DisplayResult** will display the list of numbers in each page
  - **PageInfo** will capture all the page details like number of pages, total items, start item and last item in each page, etc.

C#

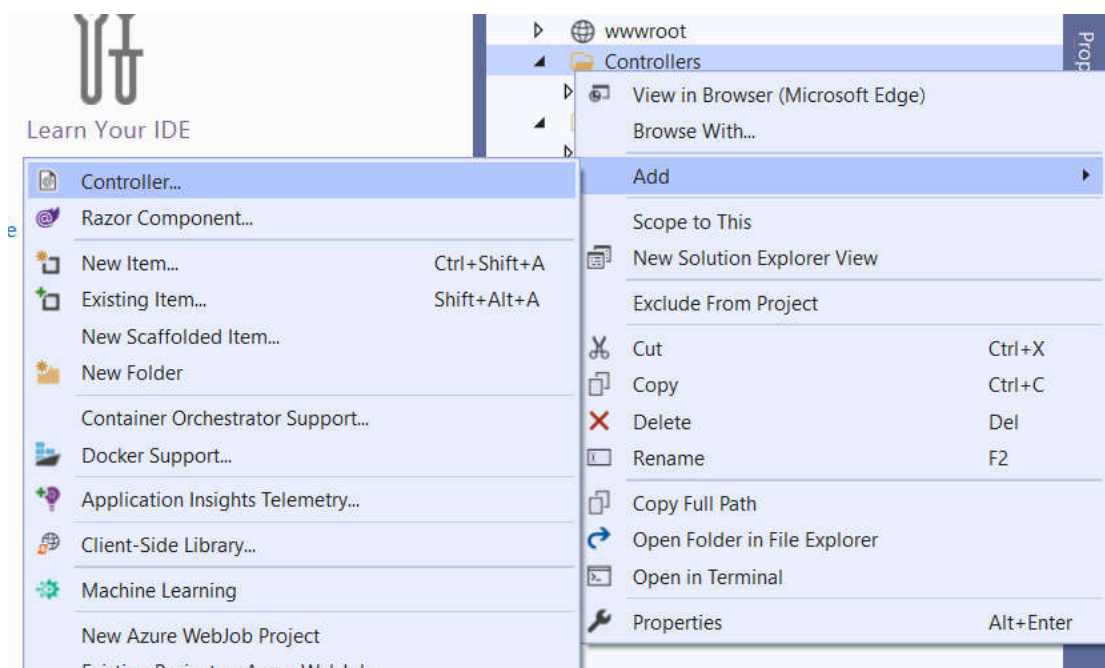
```
using HTMLHelpersApp.Common;
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;

namespace HTMLHelpersApp.Models
{
    public class ShowPaging
    {
        //validation for required, only numbers, allowed range-1 to 500
        [Required(ErrorMessage = "Value is Required!.
            Please enter value between 1 and 500.")]
        [RegularExpression(@"^\d+$", ErrorMessage = "Only positive numbers are
allowed.
            Please enter value between 1 and 500.")]
        [Range(1, 500, ErrorMessage = "Please enter value between 1 and 500.")]
        public int InputNumber { get; set; }

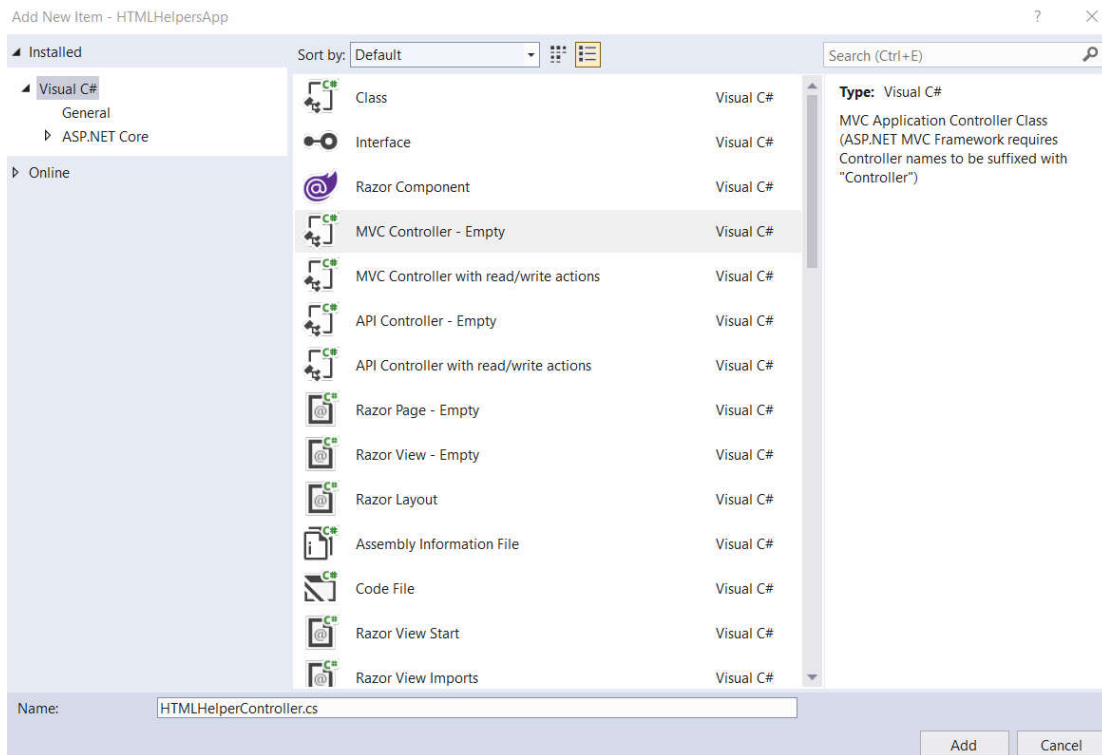
        public List<string> DisplayResult { get; set; }

        public PageInfo PageInfo;
    }
}
```

- Now that we have the required models and helpers, let us create controller and views.
- Add a new controller – '**HTMLHelperController**'.
  - Right click on the *controller* folder and select **Controller** in the context menu.



- Select '**MVCCController-Empty**'.



- Add code in '**HTMLHelperController**':
  - Number action is called for the user to enter the input number
  - **ShowPaging** action is called to display the data in multiple pages:
    - Called when page links are clicked:

C#

```
using HTMLHelpersApp.Common;
using HTMLHelpersApp.Models;
using Microsoft.AspNetCore.Mvc;
using System.Collections.Generic;

namespace HTMLHelpersApp.Controllers
{
    public class HTMLHelperController : Controller
    {
        private const int PAGE_SIZE = 10;

        public IActionResult Number()
        {
            return View();
        }

        public IActionResult ShowPaging>ShowPaging(model,
                                                    int page = 1, int inputNumber = 1)
        {
            if (ModelState.IsValid)
            {
                var displayResult = new List<string>();
                string message;

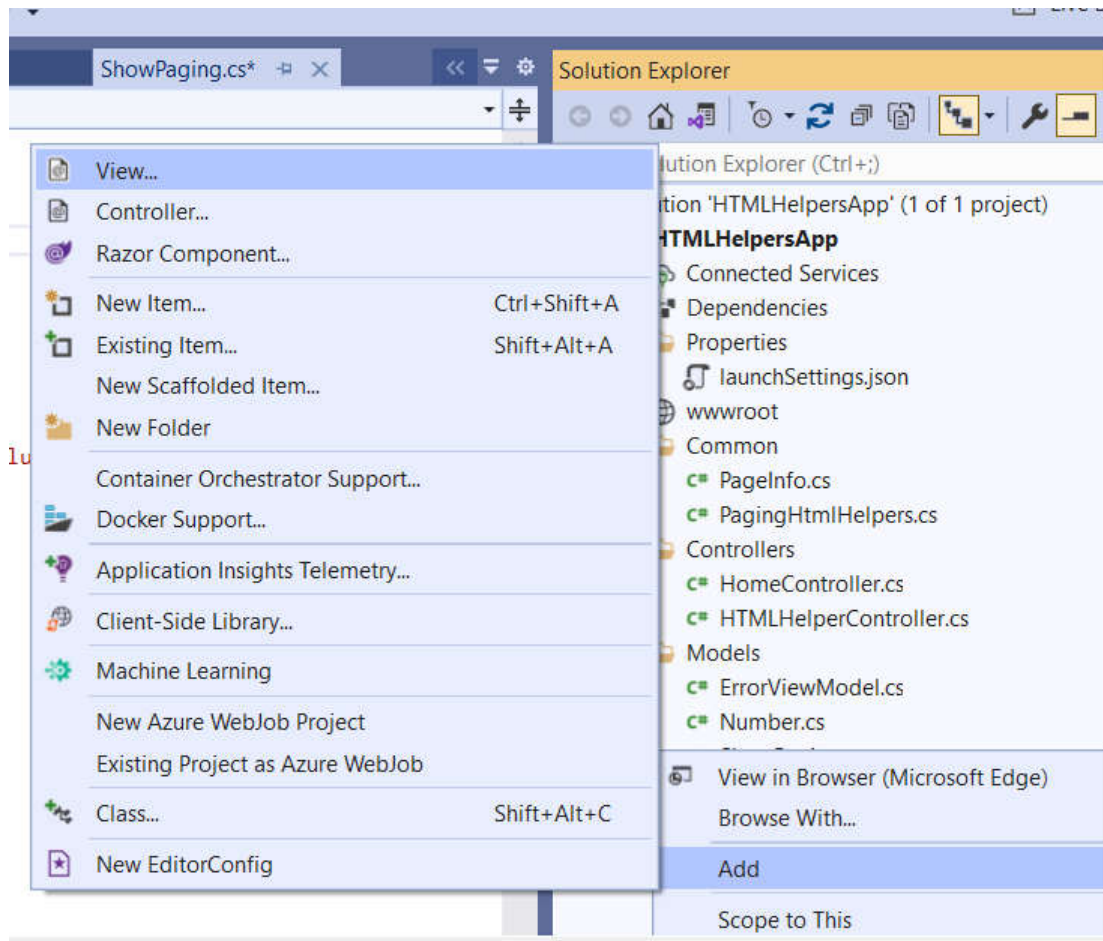
                //set model.pageinfo
                model.PageInfo = new PageInfo();
                model.PageInfo.CurrentPage = page;
                model.PageInfo.ItemsPerPage = PAGE_SIZE;
                model.PageInfo.TotalItems = inputNumber;
            }
        }
    }
}
```

```

        //Set model.displayresult - numbers list
        for (int count = model.PageInfo.PageStart;
            count <= model.PageInfo.PageEnd; count++)
        {
            message = count.ToString();
            displayResult.Add(message.Trim());
        }
        model.DisplayResult = displayResult;
    }
    //return view model
    return View(model);
}
}
}

```

- Create a new folder '*HTMLHelper*' in *Views* folder:
  - Follow the same steps mentioned above to add a new folder.
- Create a new view '*Number.cshtml*':



- Right click on **Views** > *HTMLHelper* folder and add a new view.
- Add code in '*Number.cshtml*':

C#

```

@model HTMLHelpersApp.Models.Number

<h4>Number</h4>
<hr />
<div class="row">

```

```

<div class="col-md-4">
    <form asp-action="ShowPaging" method="get">
        <div asp-validation-summary="ModelOnly" class="text-danger"></div>
        <div class="form-group">
            <input asp-for="InputNumber" class="form-control"/>
        </div>
        <div class="form-group">
            <input type="submit" value="Submit" class="btn btn-primary" />
        </div>
    </form>
</div>
</div>

```

- Similarly, create a new view 'ShowPaging.cshtml':
  - Follow the same steps as mentioned above:
- Add code in 'ShowPaging.cshtml':

C#

```

@model HTMLHelpersApp.Models.ShowPaging
@using HTMLHelpersApp.Common

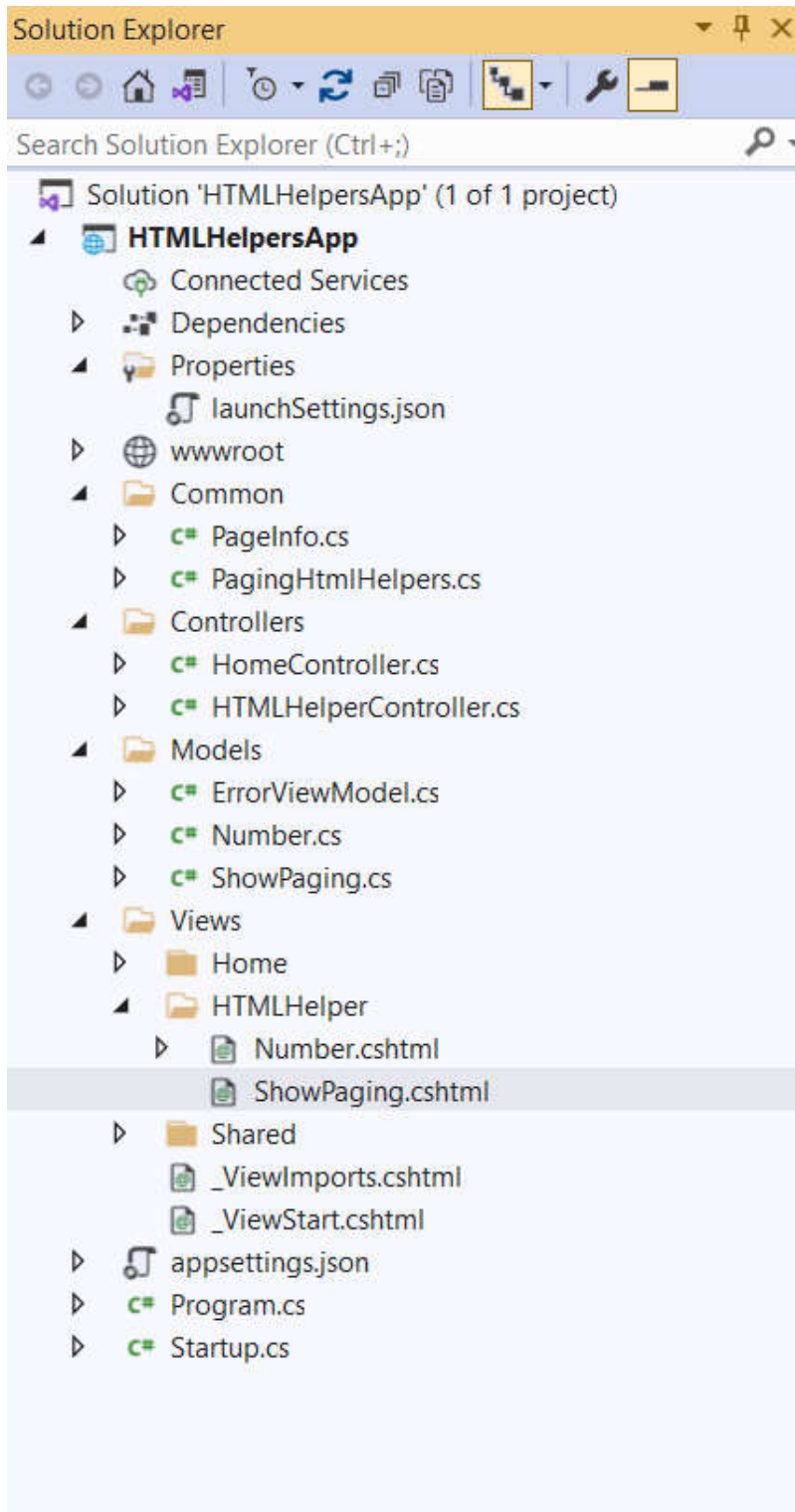
<link rel="stylesheet" href = "~/css/anchorstyles.css"/>
<form>
    <h4>Show Paging</h4>
    <hr />
    <div asp-validation-summary="All" class="text-danger"></div>
    <dl class="row">
        <dt class="col-sm-2">
            <b>Number: </b> @Html.DisplayFor(model => model.InputNumber)
        </dt>
        <dd>
            <a asp-action="Number">Change Number</a>
        </dd>
    </dl>

    <div>
        @if (Model != null && Model.DisplayResult != null)
        {
            <ul>
                @foreach (var item in Model.DisplayResult)
                {
                    <li>@Html.Raw(item)</li>
                }
            </ul>
            <div>
                @Html.PageLinks(@Model.PageInfo, x => Url.Action("ShowPaging",
                    new { page = x.ToString(), inputNumber = @Model.InputNumber }))
            </div>
        }
    </div>
</form>

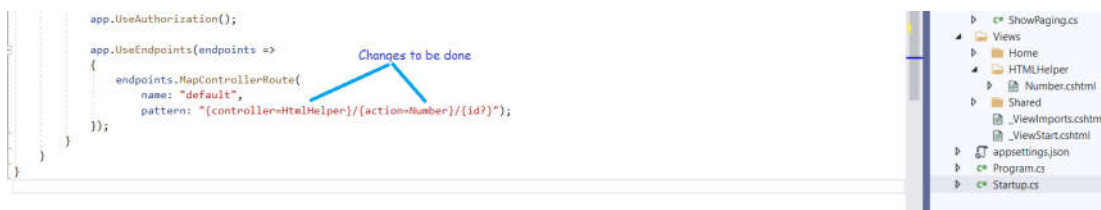
```

**Note:** `@Html.PageLinks` is the custom `HttpHelper`.

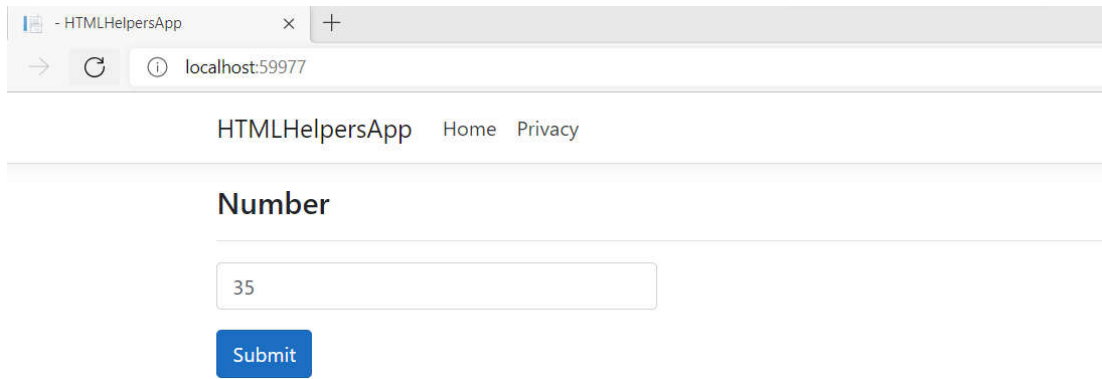
- Solution Explorer will be as follows:



- We added all the required files. Before we run, let us configure the default controller and action in 'startup.cs'.



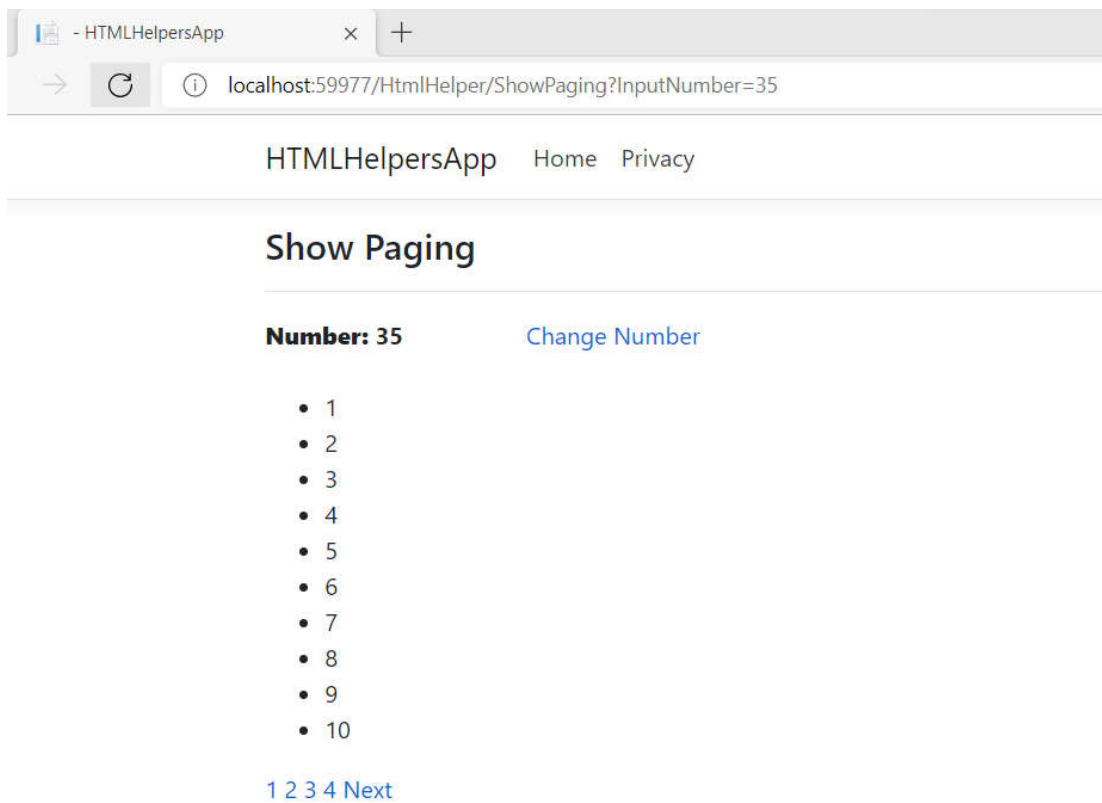
- Compile and run the application.
- Enter the input number as **35**.



HTMLHelpersApp Home Privacy

## Number

- Click on **Submit**:



HTMLHelpersApp Home Privacy

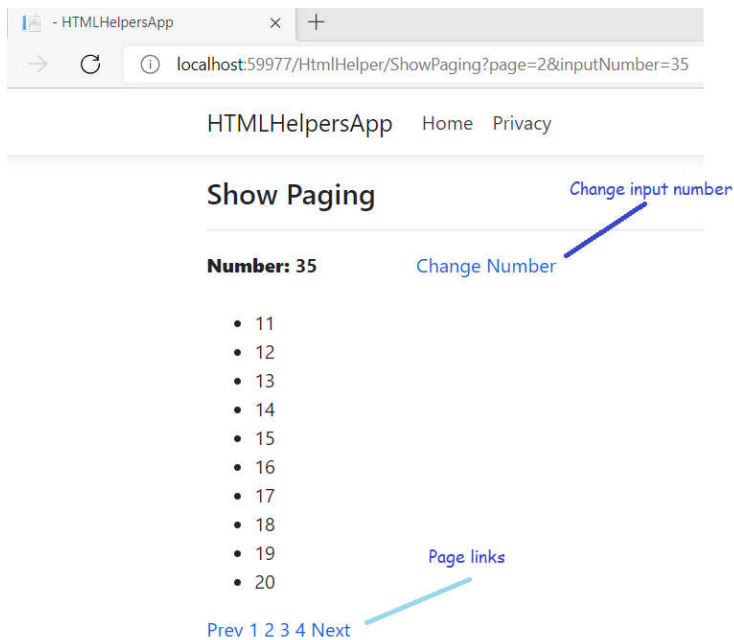
## Show Paging

**Number: 35** [Change Number](#)

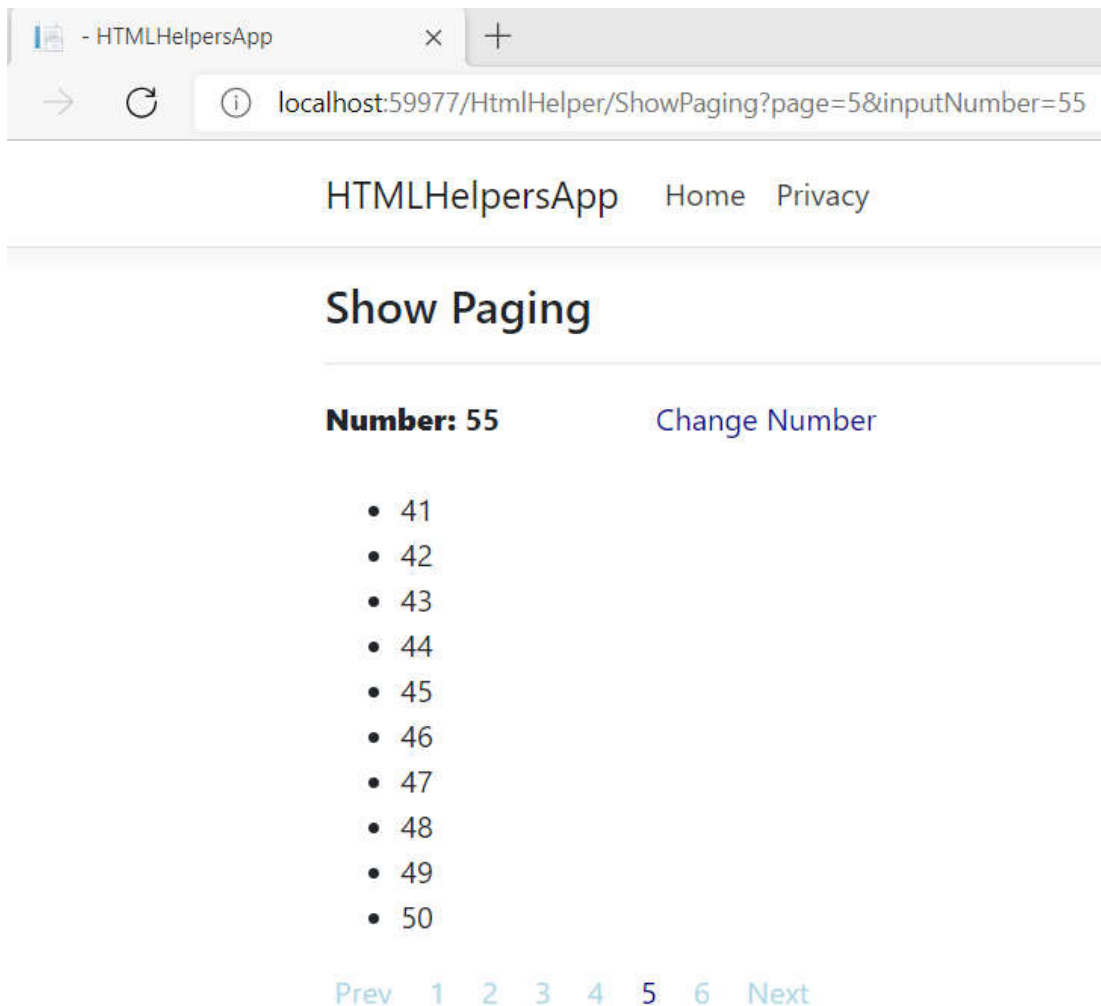
- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10

[1](#) [2](#) [3](#) [4](#) [Next](#)

- You see the paging at the bottom. 10 numbers are displayed on each page. So, there will be 4 pages to display 35 numbers. Click on page 2 link to see the second page.



- You can apply stylings to the page links as required.



## History



- 13<sup>th</sup> February, 2022: Initial version

## License

This article, along with any associated source code and files, is licensed under [The Code Project Open License \(CPOL\)](#)


Written By

**mural3**

 United States

This member has not yet provided a Biography. Assume it's interesting and varied, and probably something to do with programming.

## Comments and Discussions

 **1 message** has been posted for this article Visit <https://www.codeproject.com/Articles/5324839/Implement-Pagination-in-NET-Core-MVC-Application-w> to post and view comments on this article, or click [here](#) to get a print view with messages.

[Permalink](#)

[Advertise](#)

[Privacy](#)

[Cookies](#)

[Terms of Use](#)

Article Copyright 2022 by mural3

Everything else Copyright ©

[CodeProject](#), 1999-2023

Web04 2.8:2023-05-13:1