COMP4701 – Fall 2025 Web Application Development

3-Developing ASP.Net Core Web Applications

Dr. Abdullah Al-Hamdani

Razor Pages

- Razor Pages are ASP.NET Core’s equivalent of the older ASP.NET Web Forms.
- Both approaches use a “code-behind” file that is tied to a front end.
- As compared to WebForms, Razor Pages
  - Focus on writing HTML instead of web controls
  - Don’t store data in ViewState collection
  - Don’t have a page life cycle as in WebForms

Razor Syntax

- Razor Pages contains both HTML and server-side code (C# code).
- Razor syntax is simpler and requires fewer keystrokes.
- Razor uses the @ character as a transition character.
  - Starting with @, C# code begins.
- Razor automatically detects the end when C# code finishes.
- Data can be shared using Model or ViewData

Example – Data Sharing

P1.cshtml.cs

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

namespace WebApplication21.Pages {
    public class IndexModel : PageModel {
        private readonly ILogger<IndexModel> _logger;

        public List<int> list =
            new List<int>(){ 10, 20, 30, 40, 50, 60 };

        public IndexModel(ILogger<IndexModel> logger){
            _logger = logger;
        }
        public void OnGet() {
            ViewData["a"] = 1234;
        }
    }
}
```

P1.cshtml

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



@foreach(int x in @Model.list)
{
  <h1>@x</h1>
}
<p>Value for A is @ViewData["a"]</p>
```

Razor Statements

- With Razor you need to differentiate statements that return a value and methods that don’t.
  - A value that is returned can be used directly.
    - For example, Model.MyData returns a string. This string is put directly between the HTML div tags:

```
<div>@Model.MyData</div>
```

  - Invoking methods that return void, or specifying some other statements that don’t return a value, a Razor code block is needed.
    - The following code block defines a string variable:

```
@{
    string name = “Muza";
  }
```

  - A foreach statement defines a Razor code block as well:

```
@foreach(var item in list) {
    <li>The item name is @item.</li>
}
```

Exercise

- Create a Razor Page that displays the numbers from 1 to 10 using h1 element using C# loop

```
<h1>1</h1>
<h1>2</h1>
………
<h1/>10</h1>
```

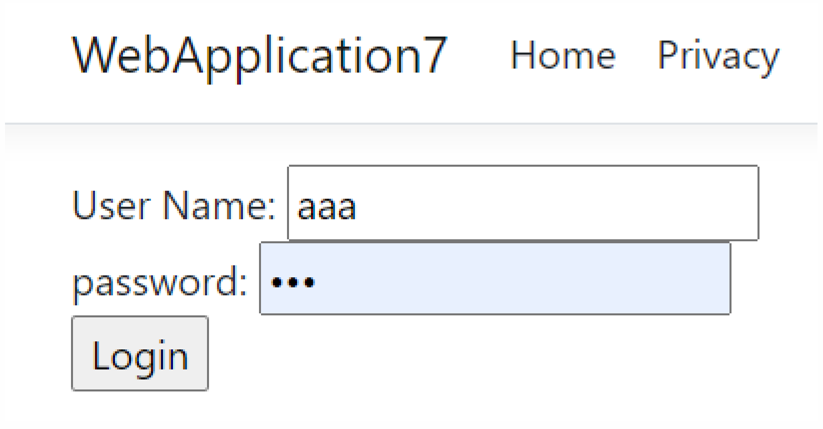
HTML Form

- There are two types of form methods
  - Get
  - Post

- What is the difference between the two types?

Razor Page

- Create new Web Application using Razor pages
- Get Web Form to send data to another web page


WebApplication7 Home Privacy

User Name: aaa

password: •••

Login

```
@page
@model WebApplication1.Pages.formModel
@{
}
<div>

 <form action="processform“ method=“get">
  User Name: <input name="uName" type="text"/><br/>
  password: <input name="pass" type="password" /> <br/>
  <input type="submit" value="Login" />
</form>

</div>
```

Getting information using Query String

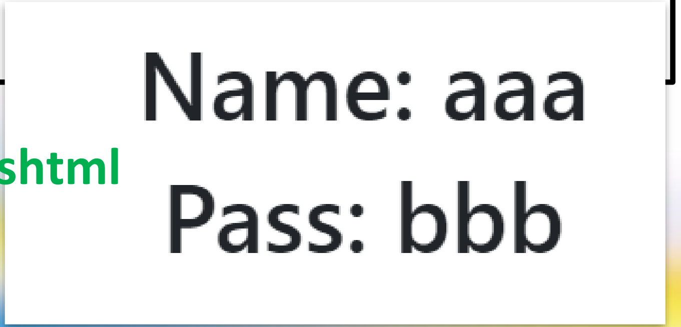
ProcessForm.cshtml

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

<div class="text-center">
    <h1 class="display-4">Welcome</h1>


    <h1>Name: @Model.name</h1>
    <h1>Pass: @Model.pass</h1>

</div>
```

ProcessForm.cshtml.cs

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

namespace WebApplication7.Pages {
    public class IndexModel : PageModel {
        private readonly ILogger<IndexModel> _logger;
        public string name = "";
        public string pass = "";

        public IndexModel(ILogger<IndexModel> logger){
            _logger = logger;
        }
        public void OnGet() {
            if (HttpContext.Request.Query["Uname"].Count > 0){
                name = HttpContext.Request.Query["Uname"];
                pass = HttpContext.Request.Query["pass"];
            }
        } } }
```

Processform.cshtml Page

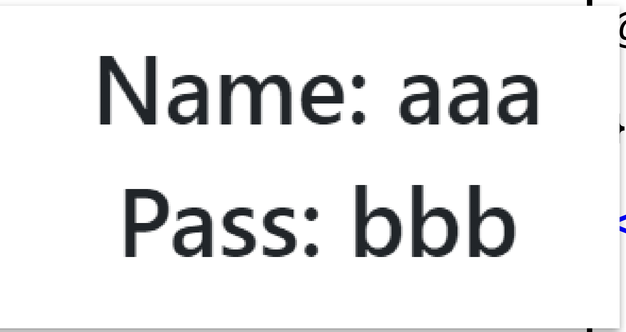


```
Name: aaa
Pass: bbb
```

Processing Form – using paramenters

ProcessForm.cshtml

Processform.cshtml Page



Name: aaa

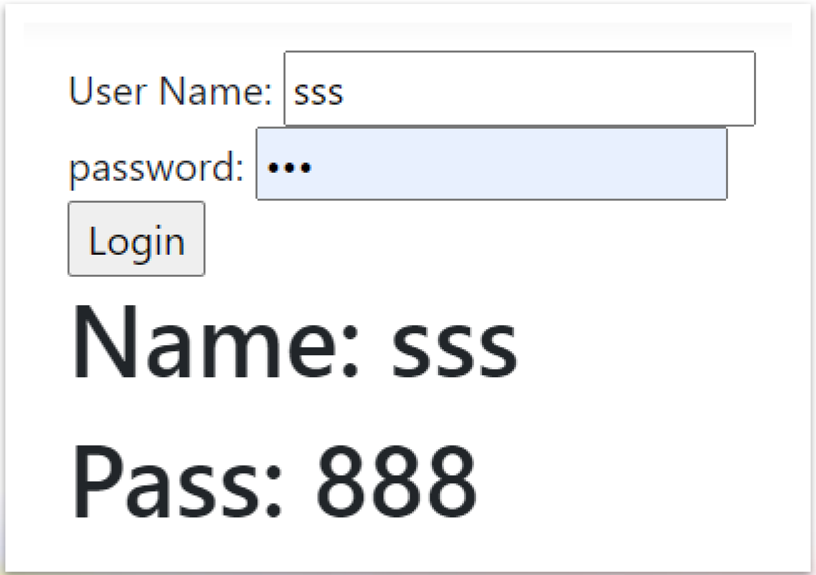
Pass: bbb

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

<div class="text-center">
    <h1 class="display-4">Welcome</h1>


    <h1>Name: @Model.name</h1>
    <h1>Pass: @Model.pass</h1>

</div>
```

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

namespace WebApplication69.Pages {
  public class processformModel:PageModel {
     public string uName;
     public string Pass;
     public void OnGet(string uName,  string Pass)
        {
            this.uName = uName;
            this.Pass = Pass;
        }
    }
}
```

Exercise

- Use one page to receive the form information and display the result
- Using GET or POST methods

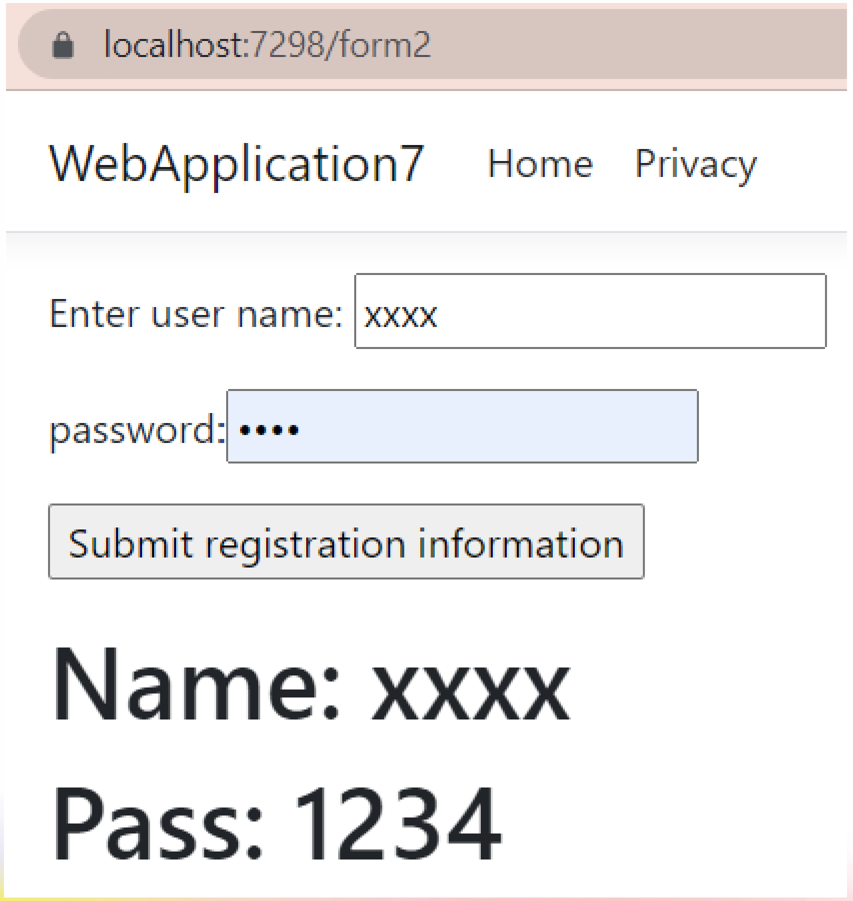

User Name: sss
password: •••
Login

Name: sss

Pass: 888

Using Model to Process Forms

- Create a new Razor page using Model and Razor form


localhost:7298/form2

Enter user name: xxxx

password: ••••

Name: xxxx

Pass: 1234

- Use

  @using (Html.BeginForm(FormMethod.Get))

  { ...form elements............}

- Form elements use the following format

```
@Html.ElementNameFor(x => x.ParmeterName)
```

Examples:

```
@Html.TextBoxFor(x => x.Email)
    Equivalent to <input type="text" name="email" />
@Html.PasswordFor(x => x.pass)
    Equivalent to <input type="password" name="pass" />
```

Form.cshtml

```
@page
@model WebApplication7.Pages.form2Model

<div>
        @using (Html.BeginForm(FormMethod.Get) )
         {
          <p>Enter user name: @Html.TextBoxFor(x => x.uname) </p>
          <p>password:@Html.PasswordFor(x => x.pass)</p>
          <p><input type="submit" value="Submit registration information"/></p>
        }

   <h1>Name: @Model.uname</h1>
   <h1>Pass: @Model.pass</h1>
 </div>
```

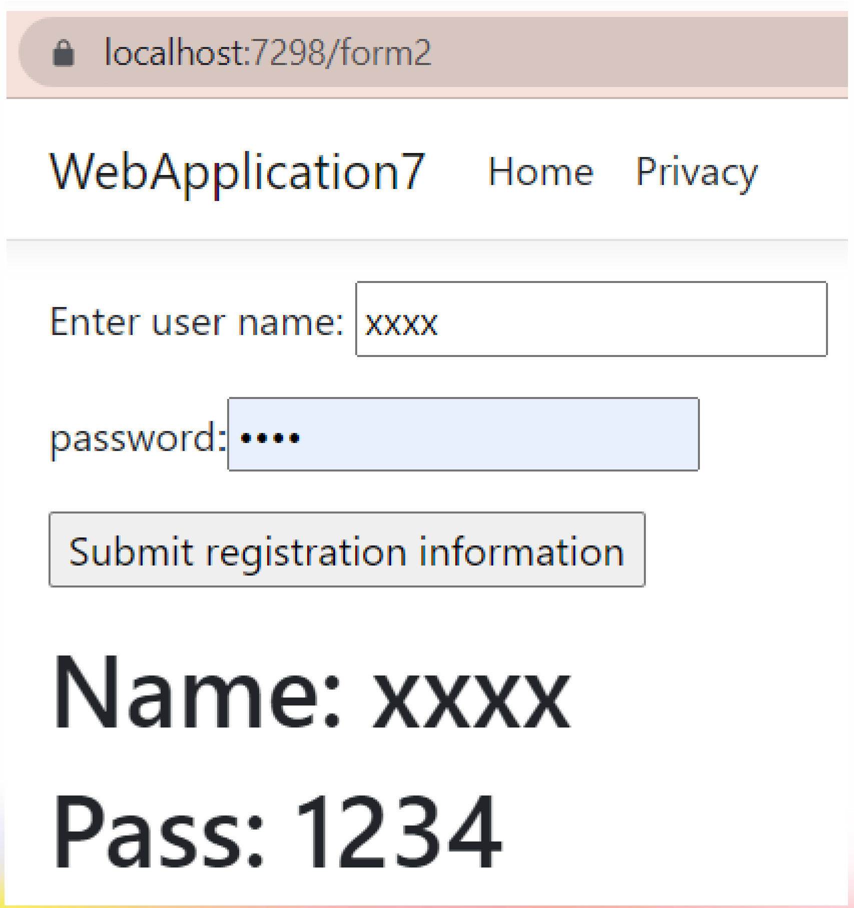
Retrieving Data from Form

Form.cshtml.cs

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

namespace WebApplication7.Pages {
    public class IndexModel : PageModel {
        private readonly ILogger<IndexModel> _logger;
        public string name = "";
        public string pass = "";

        public IndexModel(ILogger<IndexModel> logger){
            _logger = logger;
        }
        public void OnGet() {
            if (HttpContext.Request.Query["Uname"].Count > 0){
                name = HttpContext.Request.Query["Uname"];
                pass = HttpContext.Request.Query["pass"];
            }
        }
    }
}
```

Post Form

- Post method is used to processing the data in the same page


localhost:7298/form2

WebApplication7 Home Privacy

Enter user name: xxxx

password: ••••

Submit registration information

Name: xxxx

Pass: 1234

Form using Razor Syntax

Form.cshtml

```
@page
@model WebApplication7.Pages.form2Model

<div>
  @using (Html.BeginForm(FormMethod.Post) )
   {
     <p>Enter user name: @Html.TextBoxFor(x => x.uname) </p>
     <p>password:@Html.PasswordFor(x => x.pass)</p>
     <p><input type="submit" value="Submit registration information"/></p>
  }

    <h1>Name: @Model.uname</h1>
    <h1>Pass: @Model.pass</h1>
 </div>
```

Request.Form

Form.cshtml.cs

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

namespace WebApplication7.Pages {
    public class IndexModel : PageModel {
        private readonly ILogger<IndexModel> _logger;
        public string name = "";
        public string pass = "";

        public IndexModel(ILogger<IndexModel> logger){
            _logger = logger;
        }
        public void OnPost() {
            if (HttpContext.Request.Query["Uname"].Count > 0){
                name = Request.Form["Uname"];
                pass = Request.Form["pass"];
            }
        }
    }
}
```

parameters

Form.cshtml.cs

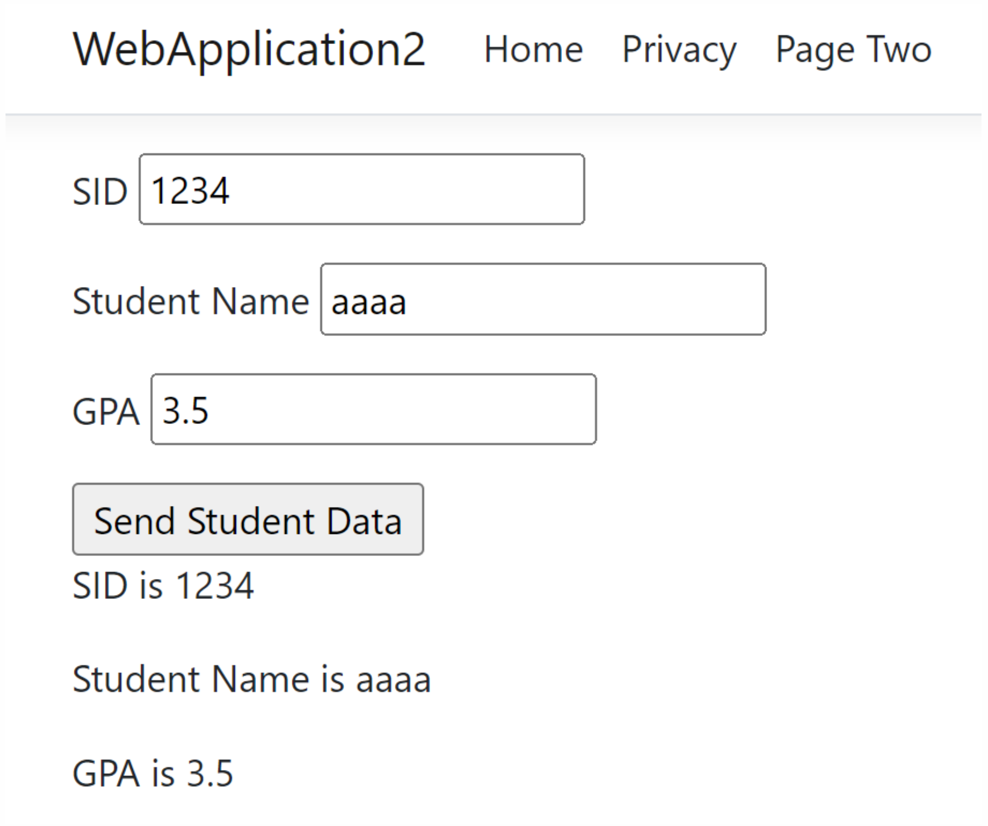
```
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;

namespace WebApplication7.Pages {
    public class IndexModel : PageModel {
        private readonly ILogger<IndexModel> _logger;
        public string name = "";
        public string pass = "";

        public IndexModel(ILogger<IndexModel> logger){
            _logger = logger;
        }
        public void OnPost (string uname,string pass)
        {
            this.uname = uname; this.pass=pass;
        }
    }
}
```

- Example

```
public class Student
{
    public int sid { get; set; }
    public string name { get; set; }
    public double GPA { get; set; }
}
```

HTML File

```
@page
@model WebApplication2.Pages.StudentModel
@{  ViewData["Title"] = "Bing Class to Form"; }

@using (Html.BeginForm(FormMethod.Get)){
    <p>SID: @Html.TextBoxFor(m => m.s1.sid)</p>
    <p>Student Name: @Html.TextBoxFor(m => m.s1.name)</p>
    <p>GPA: @Html.TextBoxFor(m => m.s1.GPA)</p>
    <input type="submit" value="Send Student Data"/>
}

<p>SID is @Model.s1.sid</p>
<p>Student Name is @Model.s1.name</p>
<p>GPA is @Model.s1.GPA</p>
```



SID 1234

Student Name aaaa

GPA 3.5

SID is 1234

Student Name is aaaa

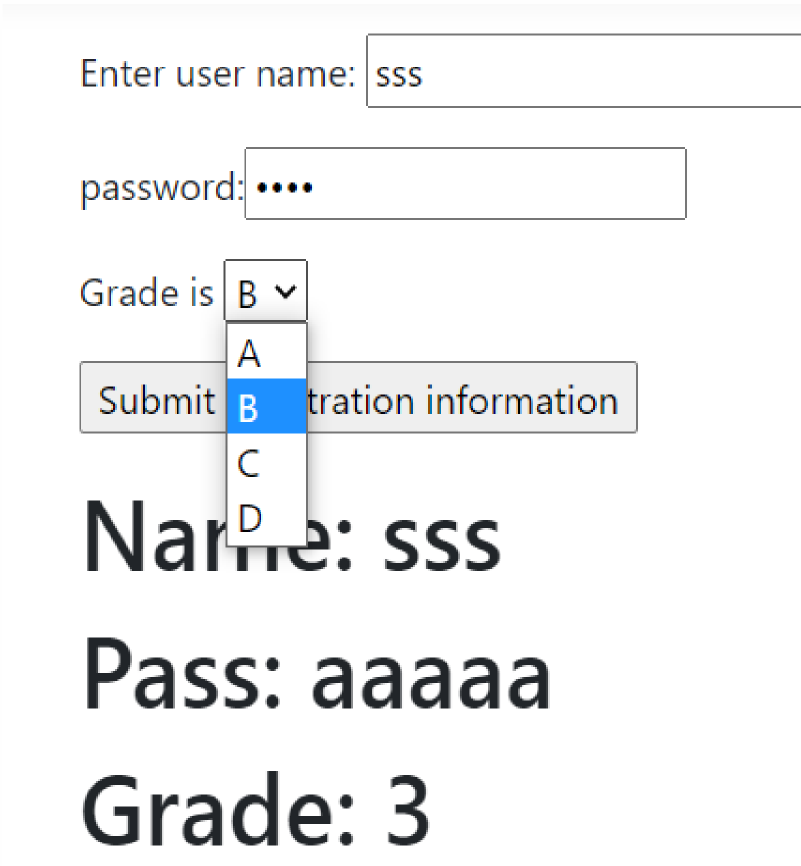
GPA is 3.5

C# Page

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

namespace WebApplication2.Pages
{

    public class StudentModel : PageModel
    {
        public Student s1 = new Student();
        public void OnGet(Student s1)
        {
            this.s1 = s1;
        }
    }
}
```

DropDownList with Options

```
@page
@model IndexModel
@{
   ViewData["Title"] = "Home page";
}
<div> @using (Html.BeginForm(FormMethod.Post) )
   {
     <p>Enter user name: @Html.TextBoxFor(x => x.uname) </p>
     <p>password:@Html.PasswordFor(x => x.pass)</p>
     <p>Grade is @Html.DropDownListFor(x => x.grade,
      new List<SelectListItem>(){new SelectListItem("A","4"),
      new SelectListItem("B","3"),new SelectListItem("C","2"),
      new SelectListItem("D","1")})</p>
     <p><input type="submit" value="Submit registration information"/></p>
   }
   <h1>Name: @Model.uname</h1>
   <h1>Pass: @Model.pass</h1>
   <h1>Grade: @Model.grade</h1>
 </div>
```



Enter user name: sss

password: ••••

Grade is B

A
B
C
D

Submit ...tration information

Name: sss

Pass: aaaaa

Grade: 3

- using Microsoft.AspNetCore.Mvc.Rendering;
- In C# page, Define List of elements

```
public List<SelectListItem> grades = new List<SelectListItem>() {
            new SelectListItem{ Text="A",Value="4"},
            new SelectListItem { Text = "B", Value = "3" },
            new SelectListItem{ Text = "C", Value = "2" },
            new SelectListItem{ Text = "D", Value = "1" }};
```

- In CSHTML page, use the list from Model parameter

```
<p>Grade is @Html.DropDownListFor(x =>x.grade,Model.grades)</p>
```

Exercise

- Use the following elements to display the language in the previous forms
  - Html.ListBoxFor
  - Html.CheckBoxFor
  - Html.RadioButtonFor
- Add more form elements using the following
  - Html.TextAreaFor
  - Html.HiddenFor

Form.cshtml.cs

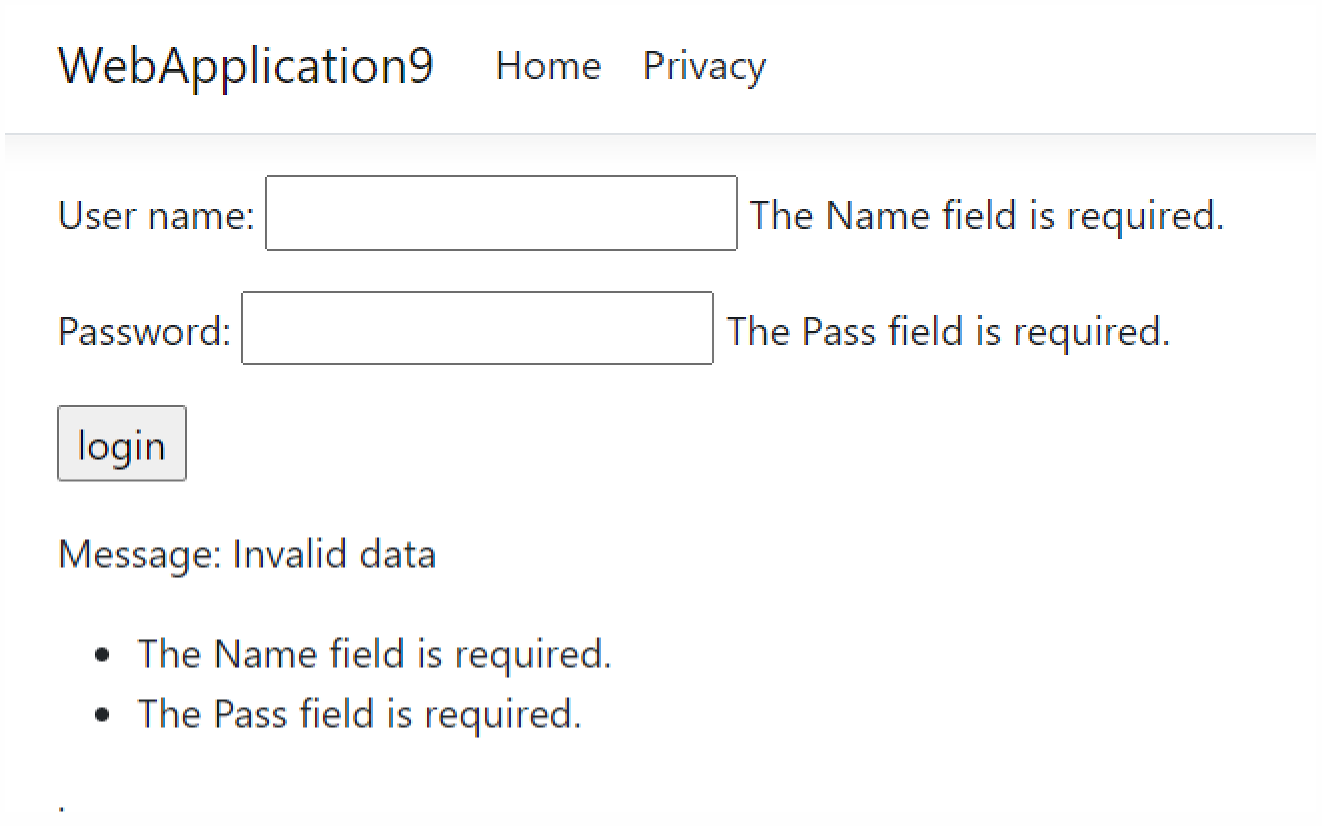
```
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;

namespace WebApplication7.Pages {
    public class IndexModel : PageModel {
        private readonly ILogger<IndexModel> _logger;
        public string name = "";
        public string pass = "";

        public IndexModel(ILogger<IndexModel> logger){
            _logger = logger;
        }
        public void OnPost(string name,string pass) {
            ViewData["Message"] = $"loginID={name} and pass={pass}";
        }
    }
}
```

Form Validation

- **Built-in validation attributes:**
  - **[CreditCard]: Validates that the property has a credit card format.**
  - **[Compare]: Validates that two properties in a model match.**
  - **[EmailAddress]: Validates that the property has an email format.**
  - **[Phone]: Validates that the property has a telephone number format.**
  - **[Range]: Validates that the property value falls within a specified range.**
  - **[RegularExpression]: Validates that the property value matches a specified regular expression.**
  - **[Required]: Validates that the field is not null.**
  - **[StringLength]: Validates that a string property value doesn't exceed a specified length limit.**
  - **[Url]: Validates that the property has a URL format.**

- **A complete list of validation attributes can be found in the System.ComponentModel.DataAnnotations namespace**

Form Validation



User name: The Name field is required.

Password: The Pass field is required.

Message: Invalid data

- The Name field is required.
- The Pass field is required.

.

Form Razor Page

```
@page
@model IndexModel
@{
    ViewData["Title"] = "Home page";
}
@using (Html.BeginForm(FormMethod.Get)){
    <p>User name: @Html.TextBoxFor(x => x.Name) @Html.ValidationMessageFor(x =>
x.Name)</p>
    <p>Password: @Html.PasswordFor(x => x.Pass) @Html.ValidationMessageFor(x =>
x.Pass)</p>
    <p><input type="submit" value="login"/></p>
}

<p>Message: @ViewData["Message"]</p>
<p>@Html.ValidationSummary();</p>
```

C# Page

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema; //sometimes needed

namespace WebApplication9.Pages {
    public class IndexModel : PageModel {
        private readonly ILogger<IndexModel> _logger;
        [Required, StringLength(10)]
        public string Name = "";
        [Required]
        public string Pass = "";
        public IndexModel(ILogger<IndexModel> logger)  {  _logger = logger;       }

        public void OnGet(string Name,string Pass){
            if (ModelState.IsValid)
             { ViewData["Message"] = $"loginID={Name} and pass={Pass}"; }
            else { ViewData["Message"] = "Invalid data"; }
        }  }  }
```

Using Class for Student with Validation

```
public class Student
{
    [Range(minimum:100000,maximum:999999,ErrorMessage ="Incorrect SID")]
    public int sid {  get; set; }
    [Required(ErrorMessage = "*"),
     MaxLength(10,ErrorMessage ="Too long name!")]
    public string name { get; set; }
    [
        Range(minimum:0,maximum:4,ErrorMessage ="GPA should be in [0,4]")]
    public float GPA { get; set; }
    [Required(ErrorMessage = "*"),
        EmailAddress(ErrorMessage ="Invalid email")]
    public string email { get; set; }

}
```

C# Class for Web Page

```
using Microsoft.AspNetCore.Mvc.RazorPages;
namespace WebApplication5.Pages {
    public class IndexModel : PageModel {
        public Student s1=new Student();
        private readonly ILogger<IndexModel> _logger;
        public IndexModel(ILogger<IndexModel> logger) { _logger = logger; }

        public void OnGet(Student s1)
        {
            this.s1 = s1;
        }
    }
}
```

Cshtml Page – Form with Validation

```
@using (Html.BeginForm(FormMethod.Get)){
    <p>SID: @Html.TextBoxFor(a=> a.s1.sid) @Html.ValidationMessageFor(x =>x.s1.sid)  </p>
    <p>Name: @Html.TextBoxFor(a => a.s1.name)
        @Html.ValidationMessageFor(x => x.s1.name)</p>
    <p>GPA: @Html.TextBoxFor(a => a.s1.GPA) @Html.ValidationMessageFor(x => x.s1.GPA)</p>
    <p>Email: @Html.TextBoxFor(a => a.s1.email)
        @Html.ValidationMessageFor(x => x.s1.email)</p>
    <input type="submit" value="Add student" />
}
<p> Validation: @Html.ValidationSummary()</p>
@if (ModelState.IsValid){
    <hr />  <p>SID: @Model.s1.sid</p>  <p>Name: @Model.s1.name</p>
    <p>GPA: @Model.s1.GPA</p> <p>Email: @Model.s1.email</p>
}
else{
  <p>Invalid form data</p>
}
```

Regular Expression Characters

Character Class	Description
.	Matches one character of any kind except line endings
\d	Matches one digit from 0 to 9
\w	Matches one letter digit or underscore
\s	Matches one whitespace character like, for example, tabs, spaces or new lines
\D	Matches one character that is not a digit
\W	Matches one character that is not a letter digit or underscore
\S	Matches one character that is not whitespace

Quantifier	Description
?	Match zero or one time
*	Match zero or more times
+	Match one or more times
{n}	Being n a number match exactly n times
{n,}	Being n a number match n or more times
{n,m}	Being both n and m numbers match between n and m times

Commonly Used Regular Expressions

Content	Regular Expression	Description
Email Addess	\S+@\S+\.\S+ \w+@\w+\.\w+ \w{3,20}@\w{3,20}\.\w{3,10}\.\w{2,5}	Characters (one or more non whitespace) followed by @ followed by one or more characters, followed by dot and followed by characters
Password (one or more letters)	\w+	One or more word letters
Specific length Password	\w{4, 10}	4 to 10 word letters
Limited length field	\S{4,10}	4 to 10 non-white space characters
US Social Security number	\d{3}-\d{2}-\d{4}	ddd-dd-dddd
Time	\d{1,2}:\d{2}:\d{2}	dd:dd:dd
Date	\d{3,8}\s\d{2},\d{4}	e.g. November 18,2009

```
public class GuestResponse
{
   [Required(ErrorMessage ="Please enter your name")]
   public string Name { get; set; }
   [Required(ErrorMessage = "Please enter your EMail")]
   [RegularExpression(@"\w+@\w+\.\w+",
           ErrorMessage = "Please enter your EMail")]
   public string Email { get; set; }
   [Required(ErrorMessage = "Please enter your Phone")]
   public string Phone { get; set; }
   [Required(ErrorMessage = "Please specify if you will attend")]
   public bool? WillAttend { get; set; } //can have null has value

}
```

Exercise

- Add a new Razor page “Registration” to enable users to register as new user
- The page should contain a form that enables the user needs to specify the username, password, name, email address.
- Use appropriate validation for each attribute
- Write code to retrieve the data from the form.

- Similar to Master Page in MS PowerPoint, used to control the layout for multiple pages
  - Project tab
  - Add new Item ➔ Razor Layout
  - Specify Layout name
  - Specify the code for the layout
    - E.g., Using table or divisions with bootstrap framework.
  - Use it to control the content of Razor pages

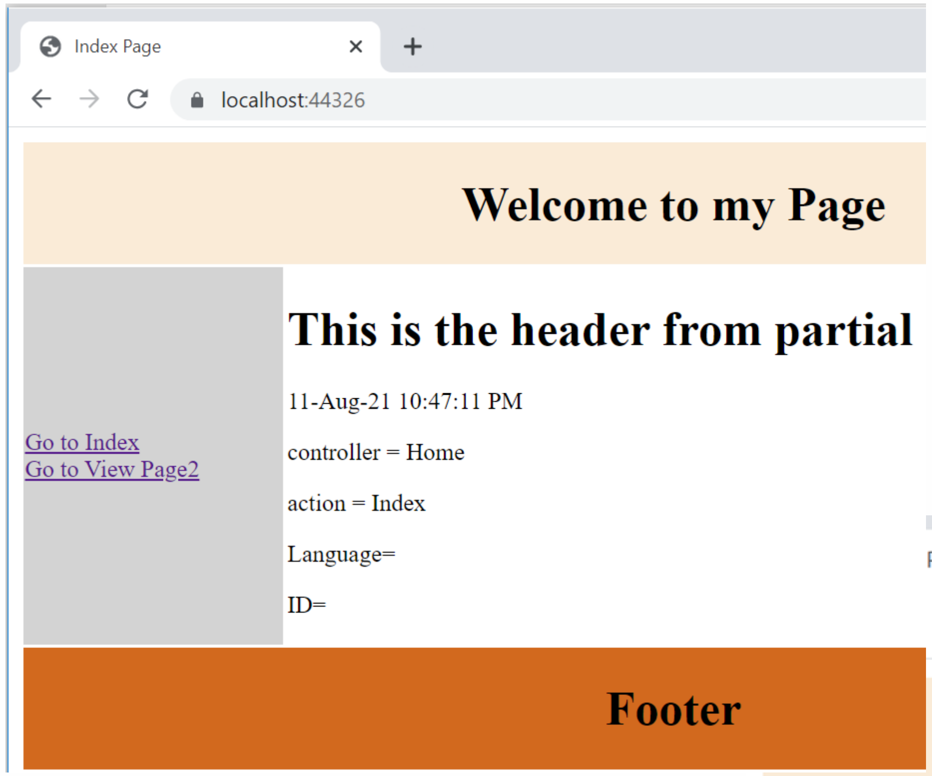
```
<!DOCTYPE html>
<html>
<head> <title>@ViewBag.Title</title>
        @RenderSection("head", required: false)
</head>
<body> <table width="100%">
        <tr><td colspan="2" align="center" style="background-color:antiquewhite">
                <h1>Welcome to my Page</h1></td></tr>
        <tr><td width="20%" style="background-color:lightgray">
                <a href="Index">Go to Index</a><br />
                <a href="Privacy">Go to View Page2</a>
            </td>
            <td>
                @RenderBody()
            </td>
        </tr>
        <tr> <td colspan="2" align="center" style="background-color:chocolate">
                <h1>Footer</h1>
                @RenderSection("foot", required: false)
            </td>
        </tr>
    </table>
</body></html>
```

Using Layout in Razor Pages

```
@{
    ViewBag.Title = “My View2 Page";
    Layout = "Shared/ABD_Layout1.cshtml";

}

<h2>This is my View2 Page</h2>
<p>@DateTime.Now.ToLongTimeString()</p>
```

$\Big\}$ RenderBody Section

Using Layout in different views



Index Page

localhost:44326

Welcome to my Page

Go to Index
Go to View Page2

This is the header from partial

11-Aug-21 10:47:11 PM

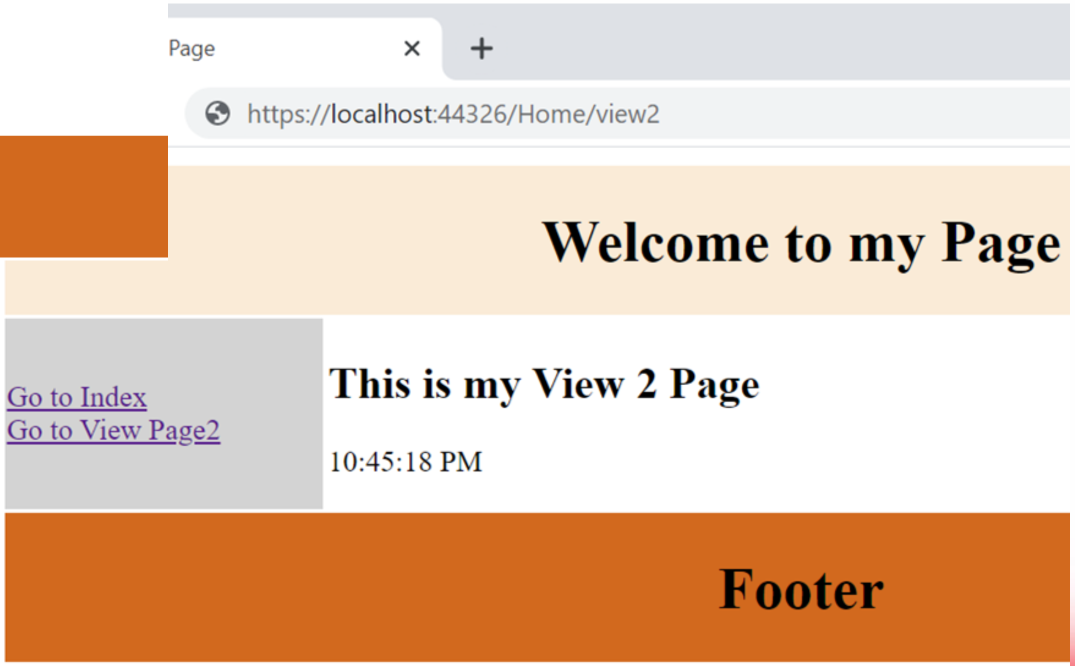
controller = Home

action = Index

Language=

ID=

Footer



Page

https://localhost:44326/Home/view2

Welcome to my Page

Go to Index
Go to View Page2

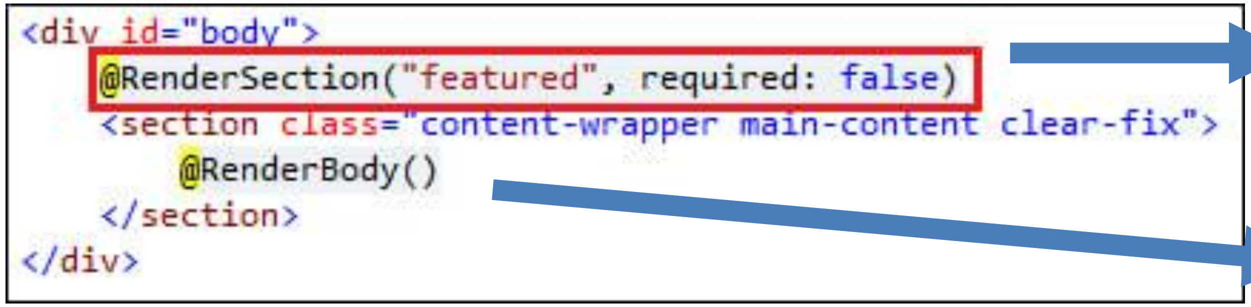
This is my View 2 Page

10:45:18 PM

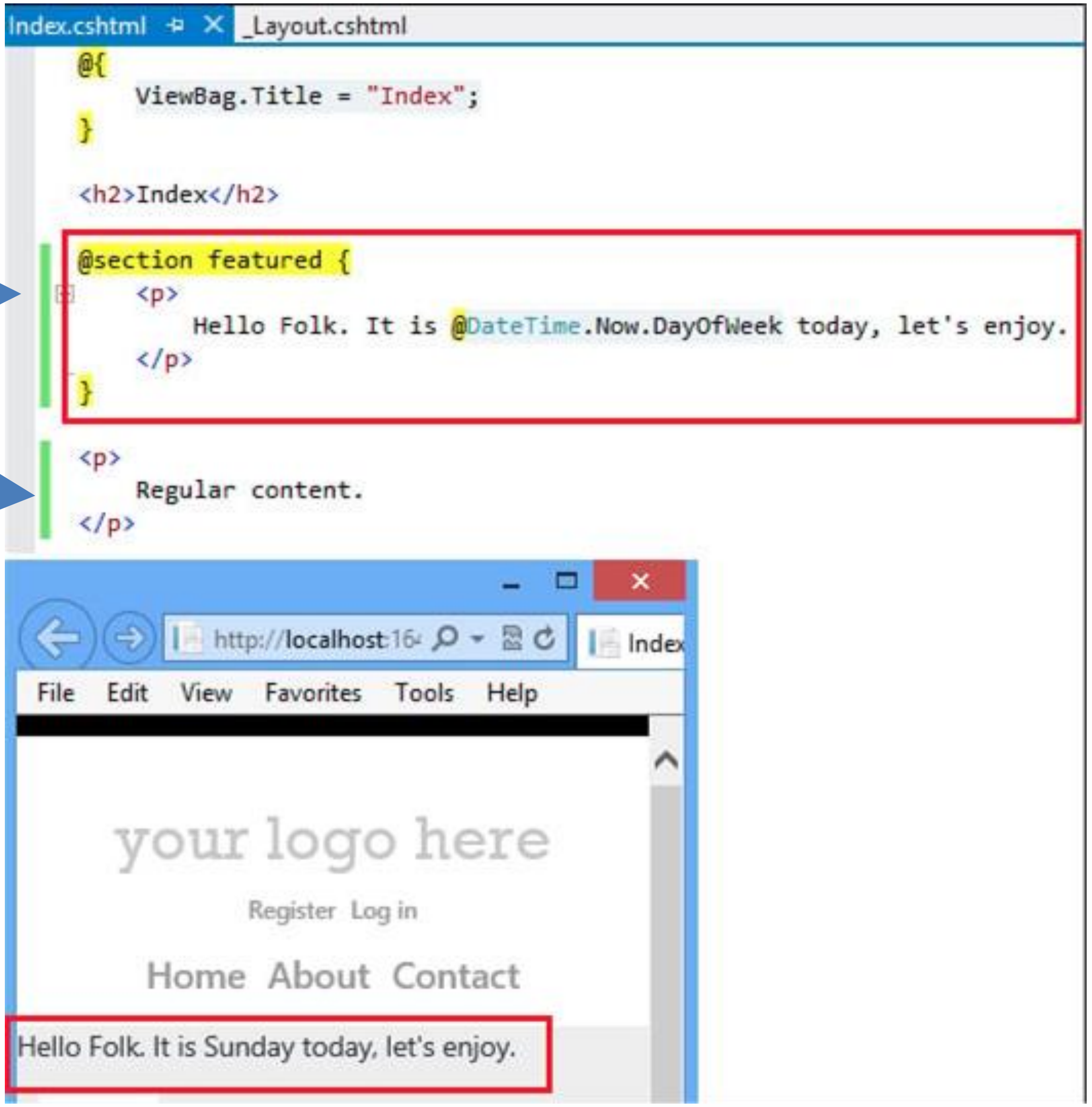
Footer

Razor Page

Layout Page



```
<div id="body">
    @RenderSection("featured", required: false)
    <section class="content-wrapper main-content clear-fix">
        @RenderBody()
    </section>
</div>
```



Index.cshtml _Layout.cshtml

```
@{
    ViewBag.Title = "Index";
}

<h2>Index</h2>

@section featured {
    <p>
        Hello Folk. It is @DateTime.Now.DayOfWeek today, let's enjoy.
    </p>
}

<p>
    Regular content.
</p>
```

Hello Folk. It is Sunday today, let's enjoy.

Example

```
<!DOCTYPE html>
<html lang="en">
   <head>
      <title>@ViewBag.Title</title>
   </head>
   <body>
      @RenderSection("header", required: false)
      @RenderBody()
      @RenderSection("footer", required: false)
   </body>
</html>
```

```
@{
   Layout = "~/Shared/_Layout.cshtml";
}

<p>Hello from Razor layout </p>

@section header{
   <p>
      header information goes here.
   </p>
}


@section footer{
   <p>
      footer information goes here.
   </p>
}
```