sagarmatha

COLLEGE OF SCIENCE & TECHNOLOGY

TRIBHUVAN UNIVERSITY AFFILIATE

Lab Report On: Net Centric Computing

Lab Report No: 02

Submitted By:

Bijay Shrestha

Roll no: 11

Group: BSc.CSIT

Submitted To:

CSIT Department

Date: 12 Nov

1.) Create a ASP.NET Core MVC Project and perform following task:

a. Create a controller Controller1 with 3 action methods:

i) Index - this should return string

ii) Details - pass some value and show this into corresponding view page

iii) Show - use of ViewData and ViewBag

PROCESS INVOLVED

→ To create a ASP.NET Core MVC Project, "Create a Project" was selected in Visual Studio.

→ "ASP.NET Core Web Application" was selected and then "Next" was selected.

→ The project name was ~~mo~~ named to "Lab2"

→ Location path was selected for project.

→ Clicked "Create"

→ All of the options, "Web-Application (Model-View-Controller" was selected and "Create" was selected.

→ To run the application, "IIS Express" or "Ctrl+ F5" was pressed.

a.) Create a controller "Controller1"

→ In Solution Explorer, right-click Controller > Add > Controller

In a dialog box, select "Controller Class - Empty"

→ Then, enter "Controller1.Controller" and select "Add"

→ After this, 3 action methods ~~a~~ were created in Controller1 which are written below i.) Index - this should return string:

```
public string Index()
{
    return " This is my default action..." ;
}
```

ii) Details - pass some value and show this in its corresponding view page

- Action Method in Controller1

```
public IActionResult Details (string detail)
{
    ViewData ["detail"] = detail;
    return View();
}
```

- View Page of Details action method

→ To create a View page of Details, we can put cursor on Details action method in Controller1 and then right-click and "Add View" and then "Razor View", select "Add" and select the layout too, and finally select "Add". This create the folder & file as

```
Views
  ↳ controller1
      ↳ Details.cshtml
```

```
@{
    ViewData ["Title"] = "Details";
}
<h1>@ViewData ["detail"] </h1>
```

→ To run this, start server/app and in browser, write URL as:

https://localhost:44394/Controller1/Details?detail= ~~Hello~~ I+am+Bijay+Shrestha

iii) Show-use of ViewData and ViewBag

```
public IActionResult Show ()
{
    ViewData ["Message"] = "Message from ViewData";
    ViewBag.Greet = "Greet from ViewBag";
    return View ();
}
```

Views of "Show" is created similarly as of Details :-

```
@{
    View Data ["Title"] = "Show";
}
<h1> @ViewData ["Message"] </h1>
<h2> @ViewBag. Greet </h2>
```

URL : localhost: 44394/Controller1/Show

2.) In your project add new Controller Controller 2 with add one action method to send id numeric value and string value to your View Page. Then show using razor Syntax to display the string value to id numbers of times.

→ Following the similar steps as done for controller 1 and its action method and views, Here also, we created "Conroller 2" controller with "Display" action method with its corresponding View page.

→ "Display" action method in Controller 2

```
public IActionResult ~~Welcome~~ Display (int n, string greet)
{
    ViewData ["n"] = n;
    ViewBag.Greet = greet;
    return View ();
}
```

→ View Page of ~~Details~~ Display is in Views/Controller 2/Display.cshtml

```
@{
    ViewData ["Title"] = "Display";
    Layout = "~/Views/Shared/_Layout.cshtml";
}
<h1> Display </h1>
<ul>
    @for (int i = 0; i < (int) ViewData ["n"]; i++)
    {
        <li> @ViewBag.greet </li>
    }
</ul>
```

→ URL: https://localhost:44394/Controller 2/Display?n=2&&

greet = Hello + Sir

3.) Now Add model Model1 in yous project with these propety members - FullName, Email, Salary

a.) Add Data Annotation on Model

b.) Add necessary Model Validations on each property Members.

PROCESS INVOLVED

→ Go to Solution Explorer, right-click Models folder, select Add and select Class, then name it as "Model1" and press Add.

→ Create the property members as

```
public string FullName { get ; set ; }
public string Email { get ; set ; }
public decimal Salary { get ; set ; }
```

a) Add Data Annotation on Model

→ The data annotation are added as

```
[Display (Name = "Full Name")]
[Required]
public string FullName {get; set; }
```

→ Similary, for other 2 properties

b.) Add necessary Model Validations

→ Email field is validated as

```
[EmailAddress]
public string Email { get; set ; }
```

FullName is validated as

→
```
[StringLength (100, MinimumLength = 3, ErrorMessage =
    "Your name is too short")]
public string FullName { get; set; }
```

Finally, the 'Model1' becomes

```
public class Model1
{
    [Display (Name = " First Name ")]
    [Required]
    [Stringlength (100, Minimum length = 3, Error Message
      = " Your name is too short ")]
    public string FullName { get ; set ; }

    [Display (Name = " Email ")]
    [EmailAddress]
    public string Email { get ; set ; }

    [Display (Name = " Salary ")]
    [Required]
    public decimal Salary { get ; set ; }
}
```

4.) On any of the above Controller, create action methods with HttpGet and HttpPost

a.) Show how can you bind the Model
b.) Check the Model State is Valid or not
c.) Show validation message in your view page

PROCESS INVOLVED

→ In controller 2, action methods with HttpGet and HttpPost are created and model binding is done as

```
[HttpGet]
public IActionResult SimpleBinding ()
{
  return View ~~()~~; ( new Model1 () { FullName = "
         Bijay Shrestha", Email = " bijay@gmail.com",
         Salary = 11000 }) ;

}
```

```
[HttpPost]
public IActionResult SimpleBinding ( Model1 webUser)
{
    if (ModelState.IsValid)
         return Content ("Hello!"+ webUser.FullName);
    else
       return View (webUser);

}
```

→ While in httpPost method, model state is checked whether it is valid or not.

→ The view page of SimpleBinding method to show validation message when there is error is

```
Views /Controller2 /SimpleBinding.cshtml

@model Lab2.Models.Model1

@{
    ViewData ["Title"] = "SimpleBinding";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h1> SimpleBinding </h1>

@using (var from = Html.BeginForm ())
{
    <div>
        @Html.LabelFor (m => m.FullName)
        @Html.TextBoxFor (m => m.FullName)
        @Html.ValidationMessageFor (m => m.FullName)

    </div>
    <div>
        @Html.LabelFor (m => m. Email)
        @Html.TextBoxFor (m => m. Email)
        @Html.ValidationMessageFor (m => m.Email)
    </div>
    <div>
        @Html.LabelFor (m => m.Salary)
        @Html.TextBoxFor (m => m. Salary)
        @Html.ValidationMessageFor (m => m.Salary)
    </div>
  <input type ="submit" value ="Submit" />
}
```