Lab 3 – Razor and Request/Session

Description

This lab is designed to give you a basic understanding of the Razor Rendering Engine and an overview of MVC's routing feature.

Estimated Time

This lab will take an estimated 4 hours to complete

Deliverable

- Push your code to GitHub and submit the link to Brightspace.
- Deploy your website to Windows Azure and submit the link to Brightspace.

See "Brightspace -> Course Content -> Extra Materials -> Microsoft Azure Web Application" for information about deploying Azure Web Apps.

Notes

- Be sure Visual Studio is up to date.
- Follow along closely to the instructions!
- When using the <input> tag be sure to set the type to collect the proper values
- Use the example code 'https://github.com/aarad-ac/IntroToASP.NETMVC' for help and tips.
- Demo can be found at: http://afrasialab3.azurewebsites.net/

Step 1: Create a new MVC Core project called 'Lab3'

1. Open Visual Studio 2022
2. Click: Create a new project
3. Select: C# -> Windows -> Web
4. Select 'ASP.NET Core Empty'
5. Name the application 'Lab3', save the project in your desired location and click the button 'Next'
6. Select .NET 6.0 for Target Framework
7. Uncheck configure for HTTPS
8. Check Do not use top-level statements

9. Click the ‘Create’ button.

Step 2: Configure your new Web Application

1. Replace the body of your main method with:

```
var builder = WebApplication.CreateBuilder(args);

// Add services to the container.
builder.Services.AddControllersWithViews();
builder.Services.AddSession();

var app = builder.Build();

// Configure the HTTP request pipeline.
if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/Home/Error");
}
app.UseStaticFiles();

app.UseRouting();

app.UseSession();

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

app.Run();
```

Step 3: Create the ‘Controllers’ and ‘Views’

1. Create a folder in your project called ‘Controllers’
    a. Right click on the project in solution explorer, then ‘Add’, then ‘New Folder’
2. Create a new Controller in this folder called ‘Home’ – Note that your class should be named ‘HomeController.cs’
    a. To do so:
        i. Right click on the ‘Controllers’ folder
        ii. Click ‘Add’

iii. Click ‘Controller…’
iv. Select the ‘MVC Controller - Empty’, and click ‘Add’
v. Select ‘MVC Controller – Empty’, type the right name, and click ‘Add’

b. Add the following code to the body of your controller:

```
public IActionResult SongForm() => View();

[HttpPost]
public IActionResult Sing()
{
   // you will complete this
}

public IActionResult CreatePerson() => View();

[HttpPost]
public IActionResult DisplayPerson(Person person)
{
   // you will complete this
}
public IActionResult Error()
{
   return View();
    }
```

3. There will be two syntax errors, on Sing and DisplayPerson methods. Ignore them for now. You will fix them later.
4. Create a folder in your project called ‘Views’
5. At the root of this new ‘Views’ folder, create a file called ‘_ViewImports.cshtml’. You can use the following steps:
   a. Right click on the ‘Views’ folder
   b. Click ‘Add’
   c. Click ‘New Item…’
   d. Select ‘Razor View Imports’, and click ‘Add’
6. Add the following lines of code to the file ‘_ViewImports.cshtml’

```
@using Lab3
@addTagHelper "*, Microsoft.AspNetCore.Mvc.TagHelpers"
```

7. Follow similar steps and add ‘_ViewStart.cshtml’, and do not change the content of the created file

Step 4: You are on your own (Part 1)

1. Create wwwroot directory on the root of the project
2. Create your own StyleSheet.css, or reuse what you find in my sample code
    a. You can find it in my sample code under, wwwroot->css
3. Create a _Layout.cshtml file
    a. Check the lecture to decide where it must go
    b. Then you can add the file that by doing Add->New Item…->Razor Layout
4. Inside the layout file:
5. Add a header and footer
6. In header, put:
    a. Your name and your welcome message
7. In footer, use HTML tags/style of your choice to explain steps you took to add css file and _Layout. Should include the following:
    a. Short description of what layout does
    b. Why you put the files in those directories. E.g. CSS in directory x, and why. Layout in directory Y and why. Etc.
    c. You should not include the content of css or how you defined your css
    a. Open _ViewStart.cshtml and specify the layout you created as default layout for all views (you are on your own)

Step 5: You are on your own (Part 2)

1. Create a folder called Home under Views
2. Create a View in Views/Home called ‘Index.cshtml’
    a. Right click on Home, then Add->View…->Razor View – Empty
    b. Click Add
    c. Select Razor View – Empty and type he right name
    d. Click Add
    e. Add some text or HTML welcome message to this view
3. Create a View in Views/Home called ‘SongForm.cshtml’
4. Create a loosely typed view called ‘Sing.cshtml’
5. SongForm view should contain an MVC form that gets the number of bottles from user and passes them down to the action called ‘Sing’ in HomeController
    a. Make sure you limit the user to the inclusive range of 50 to 100 bottles
6. Action Sing in HomeController, must store the number of buttles in session or ViewData/ViewBag
7. Action Sing in HomeController, will call the view ‘Sing’ with the number of buttles still stored in session or ViewData/ViewBag

8. In Sing view, use Razor to draw the lyrics to the children's song '10 Green Bottles…' (see: https://en.wikipedia.org/wiki/Ten_Green_Bottles)
    a. Use the value user entered for the bottles in SongForm view, and you already stored in session or ViewData/ViewBag
    b. Note that the plural/singular forms of 'bottle' should be correct when you count down to 1, and when thee is no bottle left
9. Add a MVC Form in Sing view, with a button to let the user play again
    a. Play Again, should take the user back to the SongForm view
10. Add a hyperlink in the layout header that points to the SongForm view

```
@Html.ActionLink("Show the song!", "SongForm", "Home")
```

11. Add a hyperlink in the layout header that points to the Index view

```
@Html.ActionLink("Home", "Index", "Home")
```

Step 6: You are on your own (Part 3)

1. Create a folder in your project called 'Models'
2. Create a model for Person, called Person.cs, under the 'Models' folder
    a. You can do that by right clicking on Models and adding a new 'Class'
3. Person has the following properties:
    a. First Name -- string
    b. Last Name -- string
    c. Age -- int
    d. Email Address -- string
    e. Password -- string
    f. Description of person -- string
4. Create a view called 'CreatePerson.cshtml' under Views->Home
5. Add a MVC Form in CreatePerson and collect the following information from the user
    a. First Name
    b. Last Name
    c. Age
    d. Email Address
    e. Password (should be masked as you type, and it should not be shown to the user on the DisplayPerson.cshtml – see below)
    f. Description of person
6. Once collected, post the data back to DisplayPerson action in HomeController
7. Create a view called 'DisplayPerson.cshtml'

    a. Make this view a strongly typed view, using the Person model
8. Pass the person data from DisplayPerson action to DisplayPerson view
9. Display the collected results on DisplayPerson view.
10. Add a hyperlink in the layout header that points to the CreatePerson view

```
@Html.ActionLink("Persons", "CreatePerson", "Home")
```

Step 7: You are on your own (Part 4)

Create a view called ‘Error’ under Views->Home and put a custom error message (some text or HTML) there