Deadline: 20.12.2021
Points: 1

S P.N E T Web Pages

The goal of this exercise is to familiarize students with the ASP.NET Web Pages technology, the Razor view engine and the basics of C# language. It serves as an introduction to the world of .NET and a worm-up before our target technology which will be ASP.NET MVC. If you don't have Windows OS (and don't what to have), you can complete an alternative tutorial from Microsoft's official website (https://docs.microsoft.com/en-us/aspnet/core/tutorials/razor-pages/?view=aspnetcore-5.0).

1. Run Visual Studio and select **File -> New -> Project -> ASP.NET internet application (.NET Framework) C#** and select an empty template from the available options.

2. Add a new website to the project (**Web Page** from **Razor** category) and call the file *Name.cshtml*. In the code section declare a variable called `name` and assign it with value `"Robert Paulson"`. Next, in the `<body>` section of the document, add a 10-element unordered list (use the `for` statement). Each element of the list should display `His name is` concatenated with the value from variable `name`. Run the website.

3. Create new file *Default.cshtml*. Add a form which will allow to send a name to the *Name.cshtml* page using **HTTP GET**. Go back to the *Name.cshtml* file and assign the value passed in the **GET** request (`Request["name"]`) to the `name` variable. If the value is null or empty, leave the default value `"Robert Paulson"`. Test the website. Change the method from **GET** to **POST** and test the website again. As you can see, the `Request` variable stores the passed values regardless of the method they were sent by. If you want to unambiguously choose a **GET** or a **POST** method you should use `QueryString` or `Form` fields of the `Request` variable, respectively. You can also test for the request type, e.g., `IsPost` or `Request.HttpMethod`.

4. The `Request` variable allows also to gain access to cookies mechanism (through the `Cookies` field). Session is available through a separate `Session` variable. Add a new page called *Rules.cshtml* which will display the rules of Fight Club. The page should display a list of rules submitted thus far (np. `var rules = new List<string>();`) stored in session (`Session["rules"]`) and should have a form allowing to add new rules to this list. The form should be submitted using **POST** method. You should also implement the POST -> REDIRECT -> GET pattern (`Response.Redirect("Rules.cshtml");`).

5. The aim of the next exercise is to illustrate the mechanism of layouts, allowing to ensure a consistent appearance of our whole application. Add three directories to your projects – *Suits*, *Content* and *Images* – and in the *Content* directory create a new file *Style.css* and fill it with the following code.

```
.armour {
    width: 100%; height: 200px;
    background-color: #D7C66C;
}
.plate { text-align: center; }
.info {
    width: 300px;
    position: fixed; top: 250px; right: 20px;
    border: 2px solid black;
    padding: 10px;
    background-color: white;
}
```

Add images *Arc1.png* and *Arc2.png* to the *Images* directory and create three new files in the *Suits* folder: *_Layout.cshtml*, *Mark3.cshtml* and *Mark6.cshtml*. File *_Layout.cshtml* will serve as a template for the other two pages. Place the following code inside it.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <title>@Page.Title</title>
    <link rel="stylesheet" type="text/css" href="~/Content/Style.css">
    <style>
        body { background-color: #9B2B32; }
    </style>
</head>
<body>
    <div class="links">
        <a href="Mark3.cshtml">Mark III</a>
        <a href="Mark6.cshtml">Mark VI</a>
    </div>
    <div class="armour"></div>
    <div class="plate">@RenderBody()</div>
    <div class="armour"></div>
    <div class="info">@RenderSection("Info")</div>
</body>
</html>
```

Files *Mark3.cshtml* and *Mark6.cshtml* will present the contents of a template indicated by variable `Layout`. Place the following pieces of code inside these files.

```
@{
    Layout = "_Layout.cshtml";
    Page.Title = "Suit v3";
}
<img src="~/Images/Arc1.png" alt="Arc reactor (Palladium)" />
@section Info {
    <p>
        The Mark III, was the third suit created by Tony Stark and was the main suit Tony
used in the movie and in the game. After initial flight tests were completed on the Mark II ,
Tony built the Mark III. The strongest of the armors in Iron Man, it was designed for
customization, the Mark III armor can be equipped with a variety of incredible enhancements
and upgrades. It was heavily damaged at the end of the first Iron Man film by Iron Monger. It
was later replaced by the Mark IV armor.
    </p>
}
```

```
@{
    Layout = "_Layout.cshtml";
    Page.Title = "Suit v6";
}
<img src="~/Images/Arc2.png" alt="Arc reactor (Vibranium)" />
@section Info {
    <p>
        The Mark VI, was the sixth suit created by Tony Stark after he made a better Arc
Reactor. The Mark VI is designed to be powered by Vibranium, instead of Palladium. Made using
the same gold Titanium Alloy, it debuted with War Machine in the final fight of Iron Man 2.
    </p>
}
```

Pay attention to setting the layout.
How the title of each page is set? Does this filed need to be called title?
Try to view the *_Layout.cshtml* page in the browser. This effect is thanks to the underscore.

6. Add a new file *_PageStart.cshtml* to the *Suits* catalogue, move the line of code responsible for setting the layout of the pages to that file and test the result.

7. Summary [not obligatory]

The aim of the last assignment is to write a simple web store. The store should have the following components:

- welcome page,
- products list,
- a shopping cart,
- consistent layout,
- permanent header and footer (defined in the layout).

Remember about using POST/REDIRECT/GET.
To store the products, rely on session – you don't need to use a database at all.