Deadline: Zajęcia 9
Punkty: 1

S P.N E T Web Pages

Celem ćwiczenia jest zapoznanie studenta z technologią ASP.NET Web Pages, silnikiem widoków Razor oraz podstawami języka C#. Jest ono wprowadzeniem do świata .NET i jednocześnie rozgrzewką przed technologią docelową jaką będzie ASP.NET MVC. Jeśli nie masz systemu operacyjnego Windows (i nie chcesz mieć), wykonaj tutorial z oficjalnej strony firmy Microsoft:
(https://learn.microsoft.com/en-us/aspnet/core/tutorials/razor-pages/?view=aspnetcore-7.0).

Alternatywnie możesz wykonać poniższy tutorial (wymagane jest jednak środowisko Visual Studio).

1. Uruchom środowisko Visual Studio i wybierz opcję **Plik -> Nowy -> Projekt -> Aplikacja internetowa platformy ASP.NET (.NET Framework) C#**, podaj nazwę projektu, a następnie z dostępnych opcji wybierz pusty szablon.

2. Dodaj do projektu nową stronę (**Strona sieci Web** z kategorii **Razor**) o nazwie *Name.cshtml*. W sekcji skryptu zadeklaruj zmienną `name` i przypisz jej wartość `"Robert Paulson"` (https://www.w3schools.com/asp/razor_syntax.asp). Następnie w sekcji `<body>` dokumentu dodaj dziesięcioelementową nieuporządkowaną listę (skorzystaj z pętli `for`). Każdy element listy to ciąg znaków `"His name is"` skonkatenowany z wartością zmiennej `name`. Uruchom stronę [**Ctrl + F5**].

3. Stwórz nowy plik *Default.cshtml*. Dodaj formularz, który pozwoli na przesłanie do strony *Name.cshtml* metodą **GET** imienia w polu `name`. Przejdź z powrotem do pliku *Name.cshtml* i przypisz do zmiennej `name` wartość przesłaną w polu `name` żądania (`Request["name"]`). Jeśli wartość nie została przesłana lub jest pusta pozostaw wartość `"Robert Paulson"`. Przetestuj działanie strony, a następnie zamień metodę **GET** na **POST** i ponownie uruchom witrynę. Jak widzisz zmienna `Request` przechowuje w tablicy wartości niezależnie od sposobu ich przesłania. Aby jednoznacznie wskazać metodę **GET** lub **POST** należy skorzystać odpowiednio z pól `QueryString` lub `Form` zmiennej `Request`. Można również sprawdzić rodzaj żądania wykorzystując np. `IsPost` lub `Request.HttpMethod`.

4. Zmienna `Request` oferuje również dostęp do mechanizmu ciasteczek (pole `Cookies`). Sesja dostępna jest poprzez osobną zmienną `Session`. Dodaj do projektu nową stronę o nazwie *Rules.cshtml* przechowującą reguły stowarzyszenia Fight Club. Strona, podobnie jak *Default.cshtml*, ma zawierać formularz z jednym polem typu `<input>` oraz przyciskiem umożliwiającym zgłoszenie jego zawartości, a także numerowaną listę zgłoszonych do tej pory reguł. Skorzystaj w tym celu z listy (np. `var rules = new List<string>();`) przechowywanej w sesji (`Session["rules"]`). Formularz ma być przesyłany metodą **POST**. Zastosuj również wzorzec POST -> REDIRECT -> GET (`Response.Redirect("Rules.cshtml");`).

5. Celem kolejnego zadania jest ilustracja działania mechanizmu układów, pozwalającego zapewnić spójny wygląd w obrębie całej witryny. Dodaj do projektu trzy katalogi – *Suits*, *Content* oraz *Images* – a następnie stwórz w katalogu *Content* nowy plik o nazwie *Style.css* i umieść w nim poniższe reguły.

```
.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;
}
```

Do katalogu *Images* dodaj obrazki http://www.cs.put.poznan.pl/mpiernik/pai/Arc1.png oraz http://www.cs.put.poznan.pl/mpiernik/pai/Arc2.png, a w folderze *Suits* stwórz trzy nowe pliki: *_Layout.cshtml*, *Mark3.cshtml* oraz *Mark6.cshtml*. Plik *_Layout.cshtml* posłuży jako szablon układu dla pozostałych dwóch stron. Umieść w nim poniższy kod.

```
<!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>
```

Pliki *Mark3.cshtml* oraz *Mark6.cshtml* będą zawierały jedynie zawartość, która ma zostać wyświetlona w układzie zdefiniowanym w pliku wskazanym przez zmienną `Layout`. Umieść w nich poniższe bloki kodu.

```
@{
    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>
}
```

- W jaki sposób ustawiana jest informacja o układzie?
- Jak przekazywany jest tytuł każdej strony? Czy pole przekazujące tytuł strony musi nazywać się `Title`?

- Spróbuj wyświetlić w przeglądarce plik *_Layout.cshtml*. Taki efekt zawdzięczamy umieszczeniu znaku podkreślenia na początku nazwy pliku.
- Zwróć uwagę na sposób wyświetlenia ciała dokumentu oraz sekcji. Mechanizm sekcji umożliwia podział dokumentu na więcej części. Usuń istniejące linki z układu, dodaj sekcję odpowiedzialną za wyświetlenie linków do podstron i na każdej z podstron wstaw do tej sekcji link do drugiej podstrony.

6. Zdarza się, że przed wyświetleniem każdej strony chcemy ustawić pewne wspólne wartości. Na przykład aby zapewnić spójność wyglądu w poprzednim przykładzie każda strona musiała mieć tę samą linijkę kodu: `Layout = "_Layout.cshtml";`. Podobnie może być w przypadku konfiguracji konta email lub połączenia z bazą danych. Aby uniknąć kopiowania tych samych fragmentów kodu pomiędzy różnymi stronami można skorzystać ze specjalnych plików *_AppStart.cshtml* oraz *_PageStart.cshtml*. Kod pliku *_AppStart.cshtml* wykonywany jest przed pierwszym żądaniem do witryny, natomiast kod pliku *_PageStart.cshtml* jest wykonywany przy każdym żądaniu. Poprzez umieszczenie pliku *_PageStart.cshtml* w podkatalogach aplikacji można utworzyć hierarchię wywołań. Dodaj nowy plik *_PageStart.cshtml* do katalogu *Suits* i umieść w nim skrypt ustawiający układ *_Layout.cshtml* jednocześnie usuwając kod odpowiedzialny za ustawienie układu z podstron. Przetestuj działanie rozwiązania. Jeśli chcesz wykonać wspólny fragment kodu po wyświetleniu strony dodaj w pliku *_PageStart.cshtml* polecenie `RunPage();`. Kod znajdujący się po tym poleceniu zostanie wykonany po wyświetleniu strony.

7. Ćwiczenie podsumowujące [**niewymagane do zaliczenia**]
   Ostatnim zadaniem jest napisanie prostego sklepu internetowego w technologii ASP.NET Web Pages. Wybierz dowolną tematykę, której ma dotyczyć stworzony przez Ciebie sklep. Sklep powinien posiadać następujące komponenty:

   - strona powitalna,
   - lista produktów,
   - koszyk,
   - strona, za pomocą której będzie można dodać nowe produkty do sklepu,
   - spójny wygląd,
   - stały nagłówek i stopka.

   Z poziomu listy produktów użytkownik może dodać dowolny produkt do koszyka. Dodanie produktu do koszyka nie powoduje jego zniknięcia z listy produktów – jest on nadal dostępny dla innych użytkowników. Użytkownik w każdej chwili może przejść do swojego koszyka i anulować zakup wybranych produktów lub sfinalizować zakup. Dopiero finalizacja zakupu powoduje usunięcie produktu z listy. Cały sklep ma być zrealizowany w oparciu o sesję (bez bazy danych).