Wydział
Informatyki
POLITECHNIKA BIAŁOSTOCKA

Programowanie aplikacji WWW w technologii .NET

Razor Pages – Struktura projektu i podstawy działania

Ireneusz Mrozek

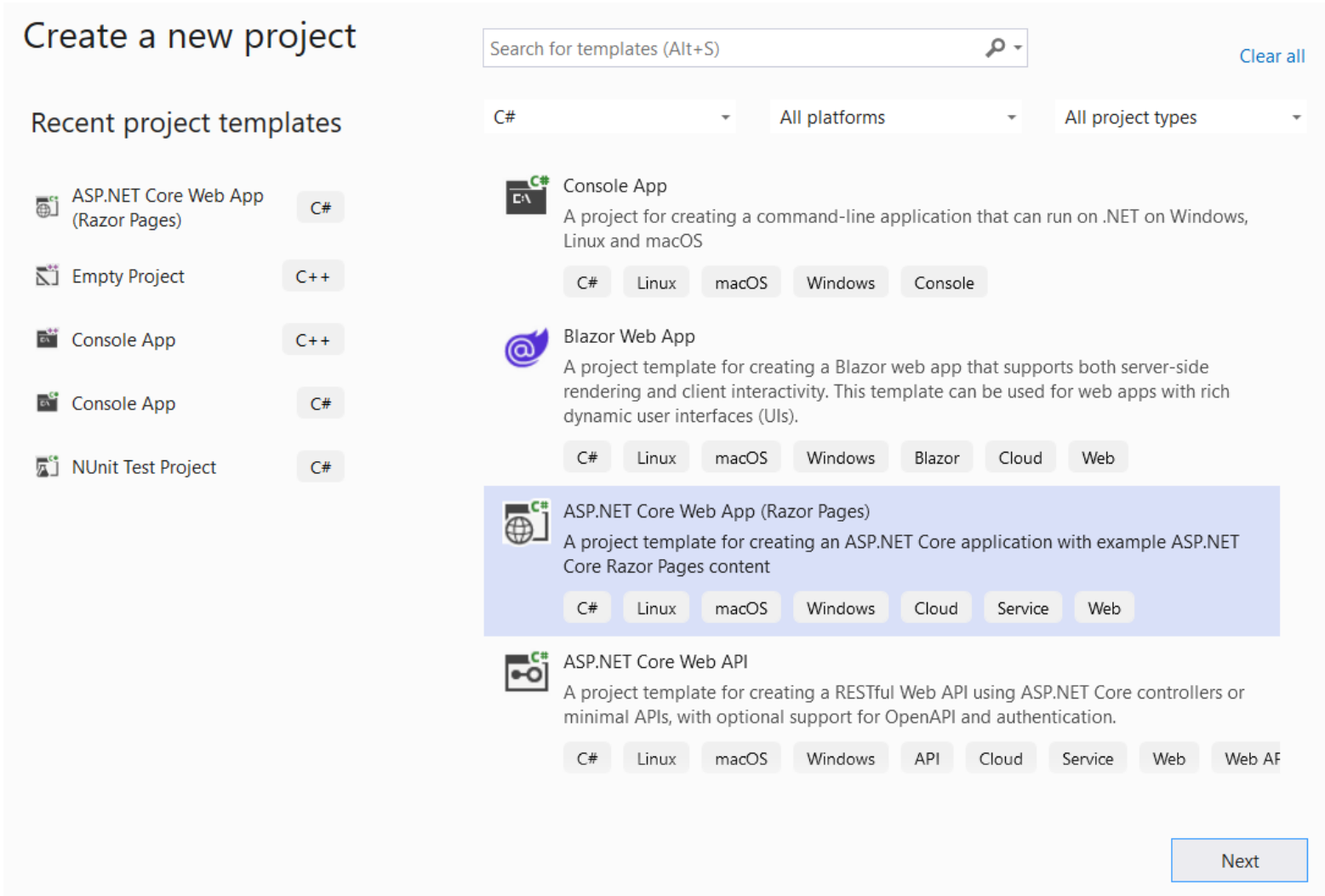
Materiały przygotowane w ramach projektu „PB 5.0 – dostosowanie oferty dydaktycznej Politechniki Białostockiej do potrzeb nowoczesnej gospodarki oraz zielonej i cyfrowej transformacji” (nr umowy FERS.01.05-IP.08-0327/23-00) w ramach programu Fundusze Europejskie dla Rozwoju Społecznego 2021-2027 współfinansowanego ze środków Europejskiego Funduszu Społecznego Plus.

Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską
NCBR Narodowe Centrum Badań i Rozwoju

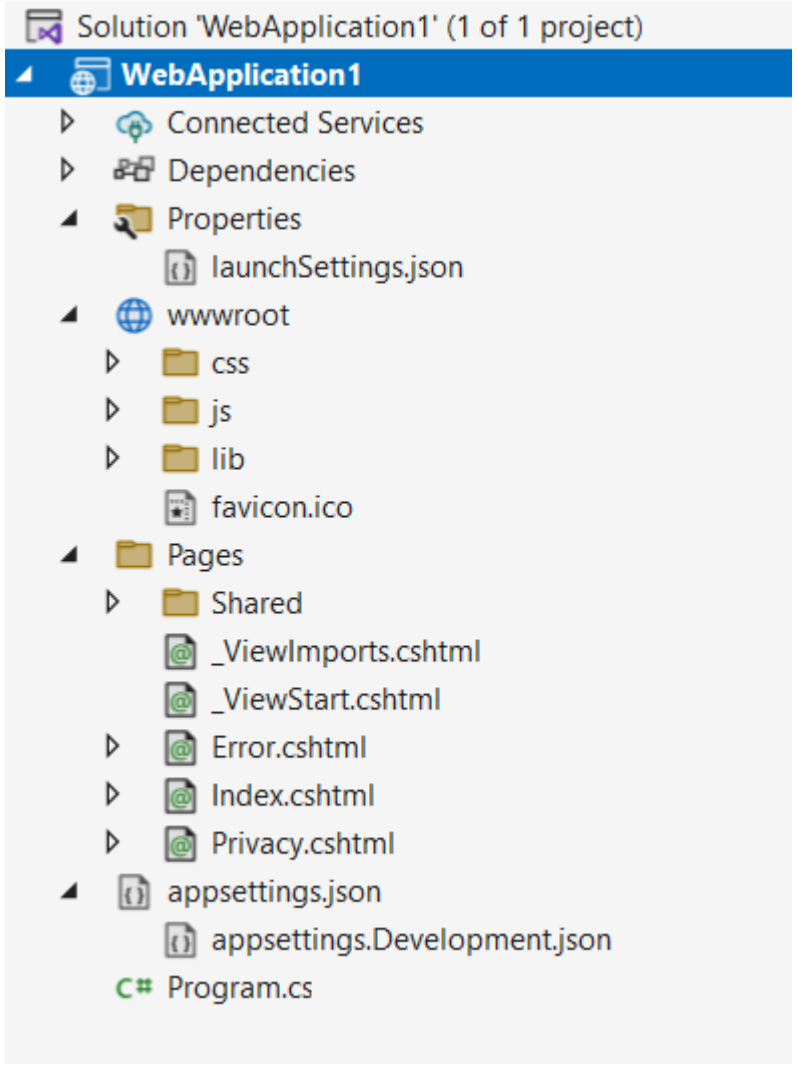
Razor Pages

- Framework do budowy stron internetowych w `ASP.NET Core`.
- Upraszcza tworzenie aplikacji webowych opartych na stronach.
- Każda strona powiązana jest z tzw. Page Model – klasą obsługującą logikę backendową.
- Nie wymaga stosowania wzorca MVC (Model-View-Controller).
- Kod HTML, C# i logika interfejsu użytkownika zintegrowane są w pliku `.cshtml`.
- Logika działania strony znajduje się w powiązanym pliku `.cshtml.cs` (Page Model).
- Idealne rozwiązanie dla prostszych aplikacji.
- Zapewnia wystarczającą elastyczność także dla bardziej złożonych scenariuszy.

Tworzenie nowego projektu ASP NET Razor Pages



Struktura projektu Razor Pages w ASP.NET



Pages/ - Główne miejsce przechowywania stron Razor Pages.

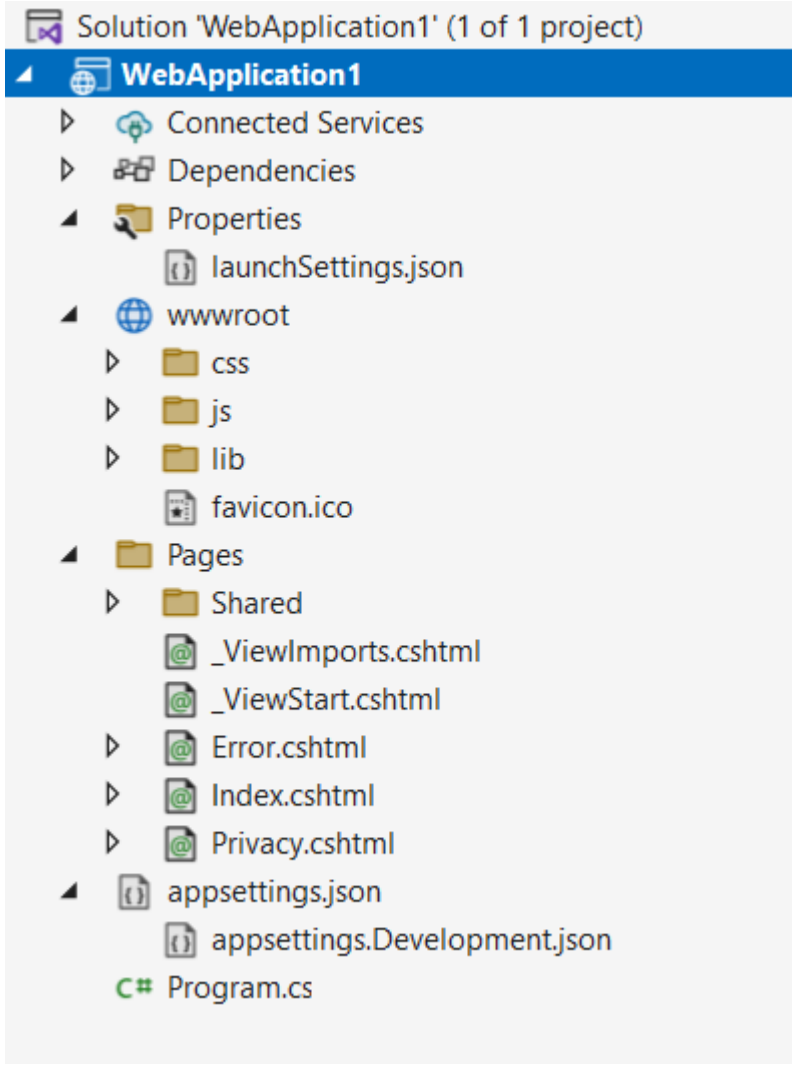
Shared/ – katalog opcjonalny, który może zawierać wspólne widoki lub komponenty, takie jak układy (_Layout.cshtml) i widoki częściowe.

_ViewImports.cshtml - importuje przestrzenie nazw i zależności (np. tagi pomocnicze) dla całego projektu.

_ViewStart.cshtml - definiuje układ (_Layout.cshtml) używany przez wszystkie strony Razor.

appsettings.json – plik konfiguracyjny dla aplikacji, w którym można ustawiać np. zmienne środowiskowe.

Struktura projektu Razor Pages w ASP.NET



wwwroot/ - Katalog na pliki statyczne, takie jak CSS, JavaScript, obrazy itp.

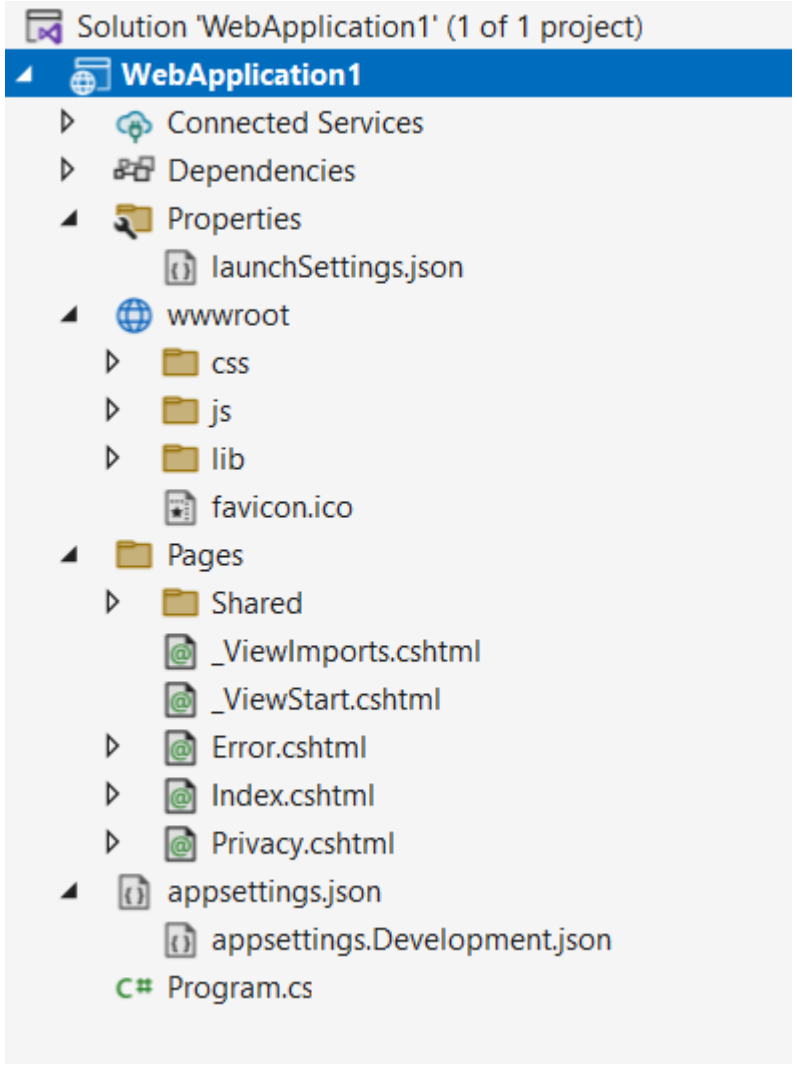
css/: pliki stylów CSS.

js/: pliki JavaScript.

lib/: zewnętrzne biblioteki, np. Bootstrap lub jQuery.

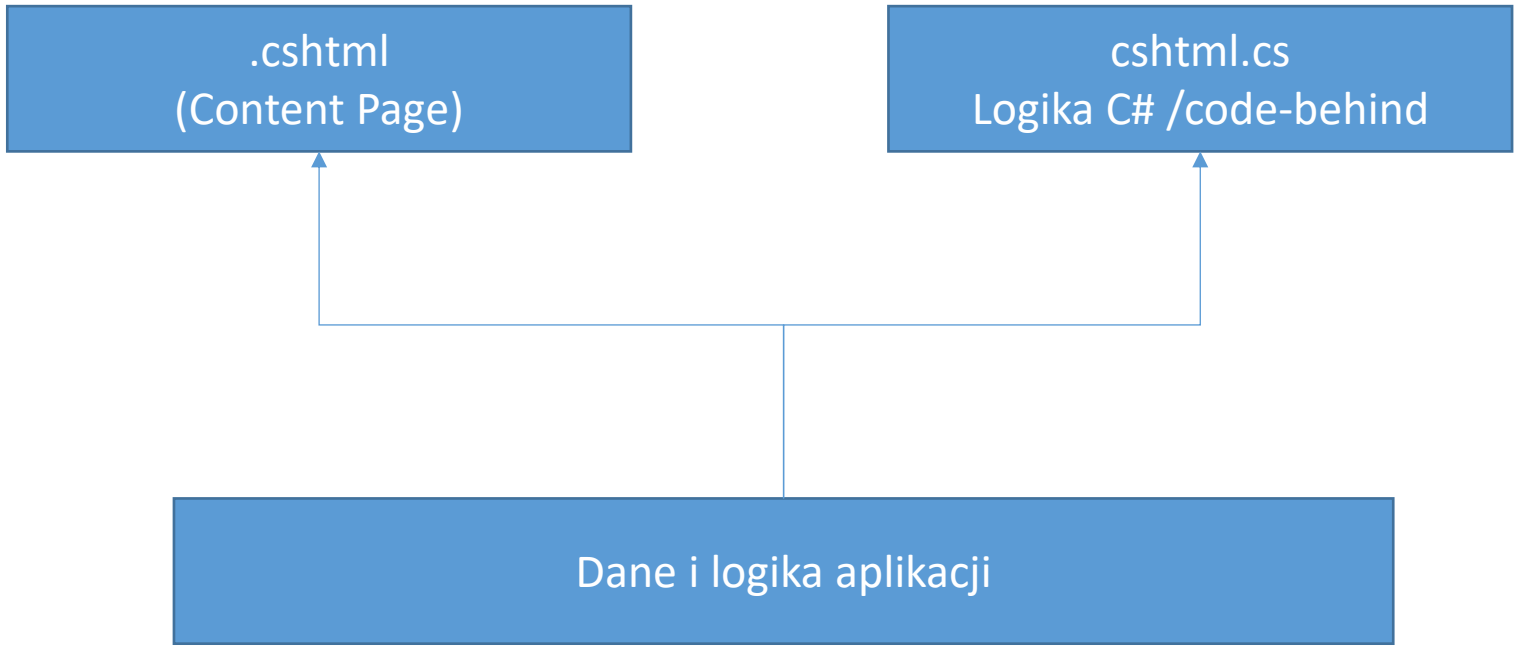
Properties/ - Zawiera plik **launchSettings.json**, definiujący konfigurację uruchamiania aplikacji (np. profile debugowania).

Struktura projektu Razor Pages w ASP.NET



Program.cs - punkt wejścia aplikacji. W tym pliku definiowane jest uruchomienie serwera i konfiguracja usług.

Struktura strony



Pliki `cshtml i cshtml.cs`

Pliki .cshtml

Definicja: Plik .cshtml to widok w technologii Razor, który łączy HTML z kodem C#.

Zastosowanie: Służy do tworzenia interfejsu użytkownika, generując dynamiczne treści na podstawie danych przekazywanych przez model.

Pliki .cshtml.cs

Definicja: Plik .cshtml.cs to plik "code-behind", który zawiera logikę C# związaną z danym widokiem.

Zastosowanie: Zawiera metody, właściwości i dane, które są używane w pliku .cshtml. Dzięki temu można zarządzać danymi wejściowymi oraz logiką przetwarzania danych.

Pliki cshtml i cshtml.cs

.cshtml

```
@page
@model ExampleModel
<div style="margin-top:30px;">
    <form method="post">
        <div>Name: <input asp-for="Name" /></div>
        <div><input type="submit" /></div>
    </form>
    @if (!string.IsNullOrEmpty(Model.Name))
    {
        <p>Hello @Model.Name!</p>
    }
</div>
```

.cshtml.cs

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;
namespace RazorPages.Pages
{
    public class ExampleModel : PageModel
    {
        [BindProperty]
        public string Name { get; set; }
    }
}
```

Struktura widoku w Razor Pages (pliki .cshtml)

```
@page
@model ContactModel

<h1>Kontakt</h1>
<p>@Model.Message</p>

<form method="post">
    <input type="text" name="userInput" />
    <button type="submit">Wyślij</button>
</form>
```

Kluczowe elementy widoku:

- HTML i Razor Syntax
- Dyrektywa @page:
- Używanie właściwości i metod z PageModel

Struktura widoku w Razor Pages (pliki .cshtml)

```
@page
@{
    var name = string.Empty;
    if (Request.HasFormContentType)
    {
        name = Request.Form["name"];
    }
}
<div style="margin-top:30px;">
    <form method="post">
        <div>Name: <input name="name" /></div>
        <div><input type="submit" /></div>
    </form>
</div>
<div>
    @if (!string.IsNullOrEmpty(name))
    {
        <p>Hello @name!</p>
    }
</div>
```

Chociaż nie jest to polecane istnieje możliwość realizacji aplikacji www wyłącznie w oparciu o strony typu content pages.

Struktura logiki w Razor Pages (pliki .cshtml.cs)

```
public class ContactModel : PageModel
{
    public string Message { get; set; }

    public void OnGet()
    {
        Message = "Witamy na stronie kontaktowej!";
    }

    public void OnPost()
    {
        // Obsługa danych z formularza.
    }
}
```

Zalety PageModel

- Jasne oddzielenie logiki aplikacji od widoku.
- Łatwiejsza organizacja kodu w porównaniu do klasycznego MVC.
- Ułatwia testowanie jednostkowe.

Obsługa żądań w Razor Pages