Politechnika Wrocławska

Platformy programistyczne .Net i Java

Aplikacja webowa w technologii ASP.NET Core

Prowadzący: Mgr. inż. Michał Jaroszczuk

Grupa Środa, 18:55-20:40

Jan Klisowski 263485

13.05.2024

Spis treści

1. Opis Projektu - Blazor Web App

Projekt polegał na implementacji aplikacji webowej w technologii Blazor. Aplikacja umożliwia tworzenie interaktywnych stron internetowych, wykorzystując platformę .NET.

2. Opis Klas i Metod

2.1. Klasa `Program`

Klasa `Program` stanowi punkt wejścia do aplikacji Blazor. Zawiera konfigurację serwera oraz inicjalizację usług i oprogramowania pośredniczącego.

2.2. Klasa `App`

Klasa `App` jest głównym komponentem aplikacji Blazor. Odpowiada za wyświetlanie interfejsu użytkownika i zarządzanie routowaniem.

2.3. Klasa `Counter`

Klasa `Counter` reprezentuje komponent licznika w aplikacji. Pozwala użytkownikowi zwiększać licznik poprzez kliknięcie przycisku.

2.4. Klasa `WeatherForecast`

Klasa `WeatherForecast` zawiera dane o prognozie pogody, takie jak data, temperatura i podsumowanie.

2.5. Metoda `IncrementCount`

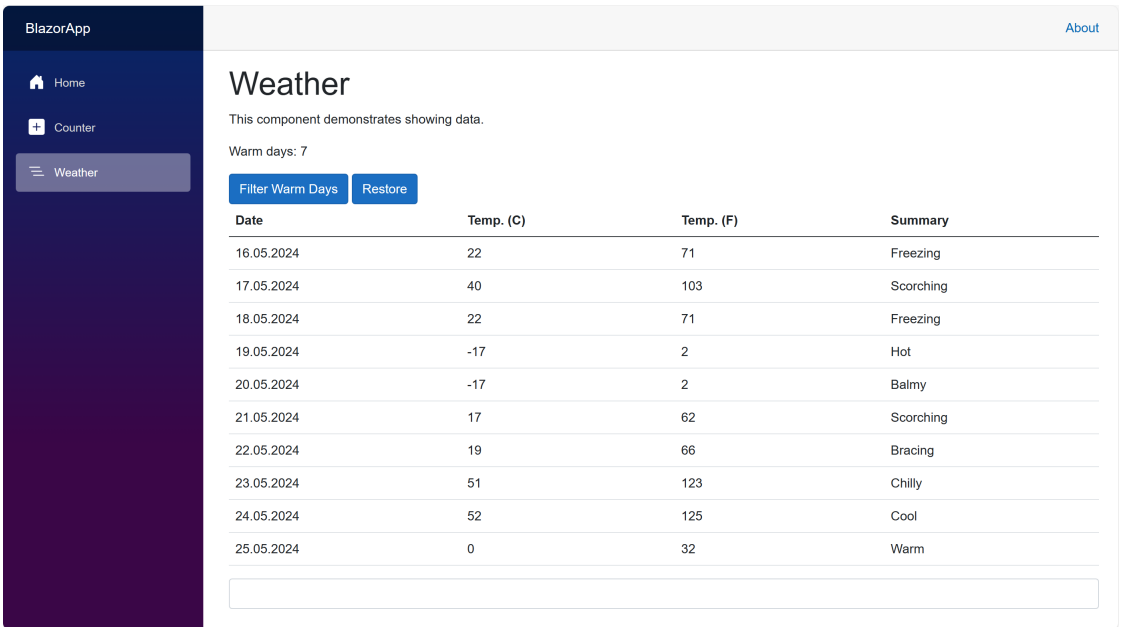
Metoda `IncrementCount` zwiększa wartość licznika o 1 po kliknięciu przycisku.

3. Repozytorium

Link do repozytorium projektu: *https://github.com/jachoofrachoo/netjavaKlis*

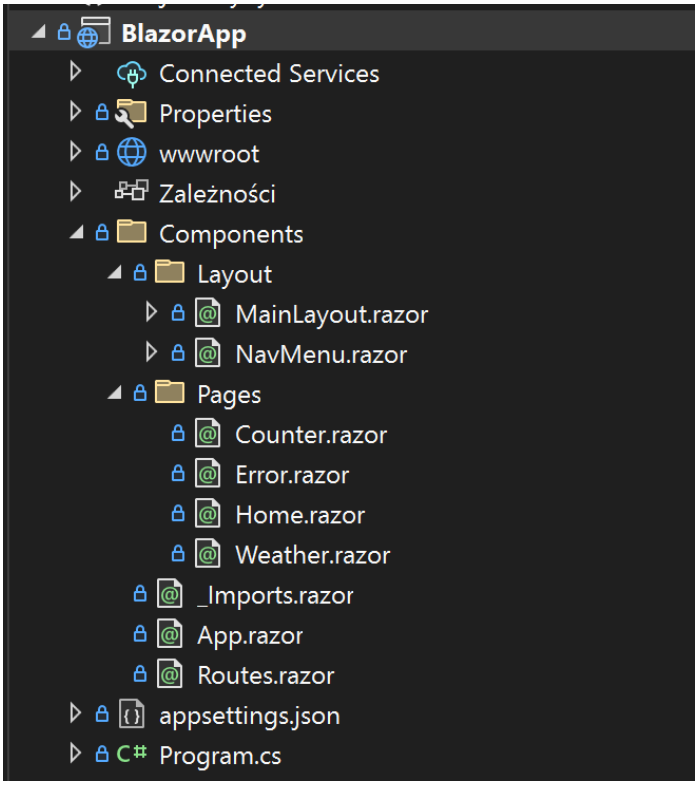
4. Zdjęcia

4.1. Interfejs Graficzny Aplikacji



Rys. 1: Interfejs Graficzny Aplikacji Blazor

4.2. Drzewo Projektu



Rys. 2: Drzewo Projektu Aplikacji Blazor

4.3. Kluczowy Fragment Kodu

```
1   @page "/weather"
2   @rendermode InteractiveServer
3   @attribute [StreamRendering]
4
5   <PageTitle>Weather</PageTitle>
6
7   <h1>Weather</h1>
8
9   <p>This component demonstrates showing data.</p>
10
11  <p role="status">Warm days: @warmDays</p>
12
13  <button class="btn btn-primary" @onclick="FilterWarmDays">Filter Warm Days</button>
14
15  <button class="btn btn-primary" @onclick="Restore">Restore</button>
```

Rys. 3: Kod Aplikacji Blazor

4.4. Kluczowy Fragment Kodu

```
@if (forecasts == null)
{
    <p><em>Loading...</em></p>
}
else
{
    <table class="table">
        <thead>
            <tr>
                <th>Date</th>
                <th>Temp. (C)</th>
                <th>Temp. (F)</th>
                <th>Summary</th>
            </tr>
        </thead>
        <tbody>
            @foreach (var forecast in filteredForecasts ?? forecasts)
            {
                <tr>
                    <td>@forecast.Date.ToShortDateString()</td>
                    <td>@forecast.TemperatureC</td>
                    <td>@forecast.TemperatureF</td>
                    <td>@forecast.Summary</td>
                </tr>
            }
        </tbody>
    </table>
}
```

Rys. 4: Kod Aplikacji Blazor

4.5. Kluczowy Fragment Kodu

```
    <input class="form-control" @oninput="@Input" />
@code {
    private WeatherForecast[]? forecasts;
    private WeatherForecast[]? filteredForecasts; // Dodanie tablicy zfiltrowanych danych
    private int warmDays = 0; // Dodanie zmiennej przechowującej liczbę ciepłych dni

    private void CountWarmDays()
    {
        warmDays = forecasts.Count(f => f.TemperatureC > 15);
    }

    // Metoda filtrowania ciepłych dni
    private void FilterWarmDays()
    {
        filteredForecasts = forecasts?.Where(f => f.TemperatureC > 15).ToArray();
    }

    // Metoda przywracająca oryginalną tabelę
    private void Restore()
    {
        filteredForecasts = null;
    }

    // Metoda filtrująca wprowadzone dane
    private void Input(ChangeEventArgs arg)
    {
        var filterText = arg.Value?.ToString()?.ToLower();
        if (string.IsNullOrWhiteSpace(filterText))
        {
            filteredForecasts = null;
            return;
        }

        filteredForecasts = forecasts?.Where(f => f.Summary?.ToLower()?.Contains(filterText) ?? false).ToArray();
    }

    protected override async Task OnInitializedAsync()
    {
        // Simulate asynchronous loading to demonstrate streaming rendering
        await Task.Delay(500);

        var startDate = DateOnly.FromDateTime(DateTime.Now);
        var summaries = new[] { "Freezing", "Bracing", "Chilly", "Cool", "Mild", "Warm", "Balmy", "Hot", "Sweltering", "Scorching" };
        forecasts = Enumerable.Range(1, 10).Select(index => new WeatherForecast // Zmiana na 10 dni prognozy
            {
                Date = startDate.AddDays(index),
                TemperatureC = Random.Shared.Next(-20, 55),
                Summary = summaries[Random.Shared.Next(summaries.Length)]
            }).ToArray();

        // Zliczanie ciepłych dni
        CountWarmDays();
    }

    private class WeatherForecast
    {
        public DateOnly Date { get; set; }
        public int TemperatureC { get; set; }
        public string? Summary { get; set; }
        public int TemperatureF => 32 + (int)(TemperatureC / 0.5556);
    }
}
```

Rys. 5: Kod Aplikacji Blazor