Politechnika Wrocławska

Dokumentacja
Platformy programistyczne .NET i Java

19 maja 2024

Filip Mirski nr indeksu 264267

1 Cel zadania

Celem zadania było nauczenie się tworzyć aplikację w środowisku ASP.NET Core w frameworku Blazor oraz publikacja aplikacji na platformie Azure.

2 Zadania

2.1 Zadanie 1

W pierwszym zadaniu należało zmodyfikować fragmenty kodu istniejącej aplikacji tak, aby w podstronie "Weather" ilość elementów wynosiła 10 oraz, żeby można było je filtrować, a także wyświetlić ilość ciepłych dni.

```
    <p><em>Number of warm days: @warmDays</em></p>

    <button class="btn btn-primary" @onclick="ShowWarmDays">Filter Warm Days</
        button>
    <button class="btn btn-primary" @onclick="ResetForecast">Reset</button>

    <input class="form-control" @oninput="InputChange" placeholder="Filter by
        summary" />
}

@code {
    private WeatherForecast[]? forecasts;
    private WeatherForecast[]? originalForecasts;
    private int warmDays = 0;

    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
        {
            Date = startDate.AddDays(index),
            TemperatureC = Random.Shared.Next(-20, 55),
            Summary = summaries[Random.Shared.Next(summaries.Length)]
        }).ToArray();

        warmDays = forecasts.Count(f => f.TemperatureC > 15);
    }

    private void ShowWarmDays()
    {
        originalForecasts = forecasts;
        forecasts = forecasts?.Where(f => f.TemperatureC > 15).ToArray();
        }

    private void ResetForecast()
        {
                forecasts = originalForecasts;
        }

    private void InputChange(ChangeEventArgs e)
        {
                var value = e.Value?.ToString();
                if (value == null)
                {
```

```
                forecasts = originalForecasts;
                return;
            }

            forecasts = originalForecasts?.Where(f => f.Summary?.Contains(
                value, StringComparison.OrdinalIgnoreCase) == true).ToArray()
                ;
        }
```

2.2 Zadanie 2

W zadaniu drugim należało zrobić aplikację bazodanową we wcześniej poznanym frameworku. Z rzeczy istotnych, które nie zostały zawarte w instrukcji to trzeba było zmodyfikować głównie plik Index.razor, efekty są widoczne poniżej:

```
<AuthorizeView>
    <Authorized>
        <PageTitle>Index</PageTitle>

        <h1>Index</h1>

        <p>
            <a href="movies/create">Create New</a>
        </p>

        <QuickGrid Class="table" Items="@(DB.Movie)">
            <PropertyColumn Property="@(movie => movie.Title)"
                Sortable="true" />
            <PropertyColumn Property="@(movie => movie.RelaseDate)"
                Sortable="true" />
            <PropertyColumn Property="@(movie => movie.Rate)"
                Sortable="true" />

            <TemplateColumn Context="movie">
                <a href="@($"movies/edit?id={movie.Id}")">Edit</
                    a> |
                <a href="@($"movies/details?id={movie.Id}")">
                    Details</a> |
                <a href="@($"movies/delete?id={movie.Id}")">
                    Delete</a>
            </TemplateColumn>
        </QuickGrid>
    </Authorized>
    <NotAuthorized>
        <p>You are not authorized to view this page.</p>

        <h1>Mapa Google</h1>
        <iframe src="https://www.google.com/maps/embed?pb=!1m18!1m12!1m3
            !1d31059.17395634519!2d-122.41941552503628!3d37
            .77492951526593!2m3!1f0!2f0!3f0!3m2!1i1024!2i768!4f13.1!3m3!1
            m2!1s0x80859a6d00690021%3A0x4a501367f076adff!2sSan%20
            Francisco%2C%20CA%2C%20USA!5e0!3m2!1sen!2suk!4v1628674395887
            !5m2!1sen!2suk" width="600" height="450" style="border:0;"
            allowfullscreen="" loading="lazy"></iframe>

    </NotAuthorized>
</AuthorizeView>
```

Z ciekawszych motywów, sortowanie kolumn robimy przy pomocy Sortable=true, oraz jest potrzebny rendermode InteractiveServer.
Jeszcze dodatkowo została przeprowadzona modyfikacja pliku Details.razor, ponieważ tam ma się wyświetlać zdjęcie oraz mamy dodawać oceny.

```
        <input type="number" id="rate" @bind-value="rate" />
        <button @onclick="AddRate">Add Rate</button>
```

```
    private async void AddRate()
        {
                movie.Rate = (movie.Rate + rate) / 2;
                await DB.SaveChangesAsync();
        NavigationManager.NavigateTo("/movies");
        }
```

2.3 Zadanie 3

W zadaniu trzecim należało zgodnie z instrukcją zamieścić aplikację w serwisie Azure, efekt jest widoczny pod zamieszczonym linkiem: Hosting Azure: `https://lab422.azurewebsites.net/movies`

Repozytorium github: `https://github.com/MrSpokeMan/.Net-and-Java/tree/master/Laboratorium4`
`https://github.com/MrSpokeMan/.Net-and-Java/tree/master/Lab4_2`