

Uniwersytet Bielsko-Bialski

LABORATORIUM

Programowanie dla Internetu w technologii ASP.NET

Sprawozdanie nr 7

Uproszczona autoryzacja

Wprowadzenie

Celem tego ćwiczenia jest konfiguracja uwierzytelniania i autoryzacji w aplikacji ASP.NET Core z wykorzystaniem Microsoft.AspNetCore.Identity oraz Razor Pages. Uwierzytelnianie pozwala na identyfikację użytkowników, a autoryzacja decyduje o dostępie do poszczególnych części aplikacji w zależności od uprawnień użytkownika.

Przebieg ćwiczenia

Modyfikacja klasy kontekstu

Pierwszym krokiem jest modyfikacja klasy kontekstu, która dziedziczy po `IdentityDbContext`. Poniżej przedstawiono zmodyfikowaną klasę `TripContext`:

```
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore;
using TripApp.Models;

namespace TripApp.Data
{
    public class TripContext : IdentityDbContext<IdentityUser>
    {
        public TripContext(DbContextOptions<TripContext> options) : base(options)
        {
        }

        public DbSet<Trip> Trips { get; set; }
        public DbSet<Client> Clients { get; set; }
        public DbSet<Reservation> Reservations { get; set; }

        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            base.OnModelCreating(modelBuilder);

            modelBuilder.Entity<Trip>().ToTable("Trip");
            modelBuilder.Entity<Client>().ToTable("Client");
            modelBuilder.Entity<Reservation>().ToTable("Reservation");
        }
    }
}
```

Kluczowe jest, aby w metodzie `OnModelCreating` wywołać `base.OnModelCreating(modelBuilder);`. Dzięki temu framework Identity poprawnie skonfiguruje encje, takie jak `IdentityUserLogin<string>`.

Dodanie stron Razor Pages

Aby uniknąć konieczności ręcznego tworzenia wszystkich widoków, skorzystamy z gotowych stron Razor Pages, które oferują podstawowe funkcje uwierzytelniania takie jak:

- Logowanie
- Rejestracja
- Przypomnienie hasła
- Wylogowanie
- Potwierdzenie adresu email

Konfiguracja aplikacji

Następnym krokiem jest dodanie middleware do obsługi uwierzytelniania i autoryzacji oraz mapowanie stron Razor Pages w metodzie `Configure` w pliku `Startup.cs`:

```
app.UseAuthentication();
app.UseAuthorization();
app.MapRazorPages();
```

Dodawanie autoryzacji do kontrolerów

Możemy teraz dodać autoryzację do naszych kontrolerów za pomocą dekoratorów. Na przykład:

- `[AllowAnonymous]` – Pozwala na dostęp do kontrolera każdemu użytkownikowi.
- `[Authorize]` – Wymaga zalogowania się przez użytkownika.

Przykładowy kod kontrolera:

```
[Authorize]
public class ClientController : Controller
{
    // ...
}

[AllowAnonymous]
public class HomeController : Controller
{
    // ...
}
```

Widok dla niezalogowanego użytkownika przy dekoratorze `[Authorize]` może wyglądać tak:

Log in

Use a local account to log in.

Email

Password

☐ Remember me?

Log in

[Forgot your password?](#)

[Register as a new user](#)

[Resend email confirmation](#)

Use another service to log in.

There are no external authentication services configured. See this [article about setting up this ASP.NET application to support logging in via external services](#).

Wnioski

Konfiguracja uwierzytelniania i autoryzacji w ASP.NET Core za pomocą Microsoft.AspNetCore.Identity oraz Razor Pages znacząco upraszcza zarządzanie użytkownikami i ich uprawnieniami. Poprzez odpowiednie dziedziczenie po **IdentityDbContext** i zastosowanie właściwych migracji, możemy zapewnić poprawne działanie mechanizmów logowania i rejestracji. Dodatkowo, dekoratory **[Authorize]** i **[AllowAnonymous]** pozwalają na precyzyjne określenie, które części aplikacji są dostępne dla poszczególnych użytkowników, co zwiększa bezpieczeństwo i kontrolę nad dostępem do zasobów.