

Uniwersytet Bielsko-Bialski

LABORATORIUM

Programowanie dla Internetu w technologii ASP.NET

Sprawozdanie nr 7

Uproszczona autoryzacja

Cel ćwiczenia

Celem ćwiczenia było zapoznanie się z podstawami uwierzytelniania i autoryzacji użytkowników w aplikacjach tworzonych za pomocą ASP.NET Core Identity.

Wprowadzenie

Uwierzytelnienie to proces polegający na potwierdzeniu zadeklarowanej tożsamości podmiotu biorącego udział w procesie komunikacji. W praktyce odbywa się to poprzez porównanie przedstawionych przez użytkownika dowodów tożsamości z danymi przechowywanymi w systemie. Celem uwierzytelniania jest uzyskanie określonego poziomu pewności, że dany podmiot jest w rzeczywistości tym, za którego się podaje.

Autoryzacja to proces nadawania podmiotowi dostępu do zasobu. Celem autoryzacji jest kontrola dostępu, która potwierdza, czy dany podmiot jest uprawniony do korzystania z żądanego zasobu. Autoryzacja następuje dopiero po potwierdzeniu tożsamości podmiotu za pomocą identyfikacji i uwierzytelnienia.

Microsoft.AspNetCore.Identity to interfejs API, który obsługuje funkcje logowania interfejsu użytkownika. Zarządza użytkownikami, hasłami, danymi profilu, rolami, oświadczeniami, tokenami, potwierdzeniem wiadomości e-mail i nie tylko. Użytkownicy mogą utworzyć konto przy użyciu informacji logowania przechowywanych w Identity lub mogą użyć zewnętrznego dostawcy logowania. ASP.NET Core Identity dodaje funkcje logowania interfejsu użytkownika do aplikacji internetowych platformy ASP.NET Core

Wykonanie ćwiczenia

W pierwszej kolejności należy zmienić dziedziczenie klasy TripContext z DbContext (EF Core) na IdentityDbContext (ASP.NET Core Identity)

```
public class TripContext : IdentityDbContext<IdentityUser>
{
    public DbSet<Trip> Trips { get; set; }
    public DbSet<Client> Clients { get; set; }
    public DbSet<Reservation> Reservations { get; set; }
    public TripContext(DbContextOptions<TripContext> options) : base(options)
    {
    }
}
```

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
{
    base.OnModelCreating(modelBuilder);
    modelBuilder.Entity<Trip>().ToTable("Trips");
    modelBuilder.Entity<Client>().ToTable("Clients");
    modelBuilder.Entity<Reservation>().ToTable("Reservations");
}
```

Następnie za pomocą scaffoldingu wygenerować widoki dla następujących zachowań :

- 1.Rejestracja
- 2.Logowanie
- 3.Wylogowywanie
- 4.Potwierdzenie rejestracji
- 5.Przypomnienie hasła

×

Add Identity

Select an existing layout page, or specify a new one:

/Areas/Identity/Pages/Account/Manage/_Layout.cshtml

(Leave empty if it is set in a Razor _viewstart file)

☐ Override all files

Choose files to override

<input type="checkbox"/> Account\StatusMessage	<input type="checkbox"/> Account\AccessDenied	<input checked="" type="checkbox"/> Account\ConfirmEmail
<input type="checkbox"/> Account\ConfirmEmailChange	<input type="checkbox"/> Account\ExternalLogin	<input checked="" type="checkbox"/> Account\ForgotPassword
<input type="checkbox"/> Account\ForgotPasswordConfirmation	<input type="checkbox"/> Account\Lockout	<input checked="" type="checkbox"/> Account>Login
<input type="checkbox"/> Account>LoginWith2fa	<input type="checkbox"/> Account>LoginWithRecoveryCode	<input checked="" type="checkbox"/> Account\Logout
<input type="checkbox"/> Account\Manage\Layout	<input type="checkbox"/> Account\Manage\ManageNav	<input type="checkbox"/> Account\Manage\StatusMessage
<input type="checkbox"/> Account\Manage\ChangePassword	<input type="checkbox"/> Account\Manage\DeletePersonalData	<input type="checkbox"/> Account\Manage\Disable2fa
<input type="checkbox"/> Account\Manage\DownloadPersonalData	<input type="checkbox"/> Account\Manage>Email	<input type="checkbox"/> Account\Manage\EnableAuthenticator
<input type="checkbox"/> Account\Manage\ExternalLogins	<input type="checkbox"/> Account\Manage\GenerateRecoveryCodes	<input type="checkbox"/> Account\Manage\Index
<input type="checkbox"/> Account\Manage\PersonalData	<input type="checkbox"/> Account\Manage\ResetAuthenticator	<input type="checkbox"/> Account\Manage\SetPassword
<input type="checkbox"/> Account\Manage\ShowRecoveryCodes	<input type="checkbox"/> Account\Manage\TwoFactorAuthentication	<input checked="" type="checkbox"/> Account\Register
<input checked="" type="checkbox"/> Account\RegisterConfirmation	<input type="checkbox"/> Account\ResendEmailConfirmation	<input type="checkbox"/> Account\ResetPassword
<input type="checkbox"/> Account\ResetPasswordConfirmation		

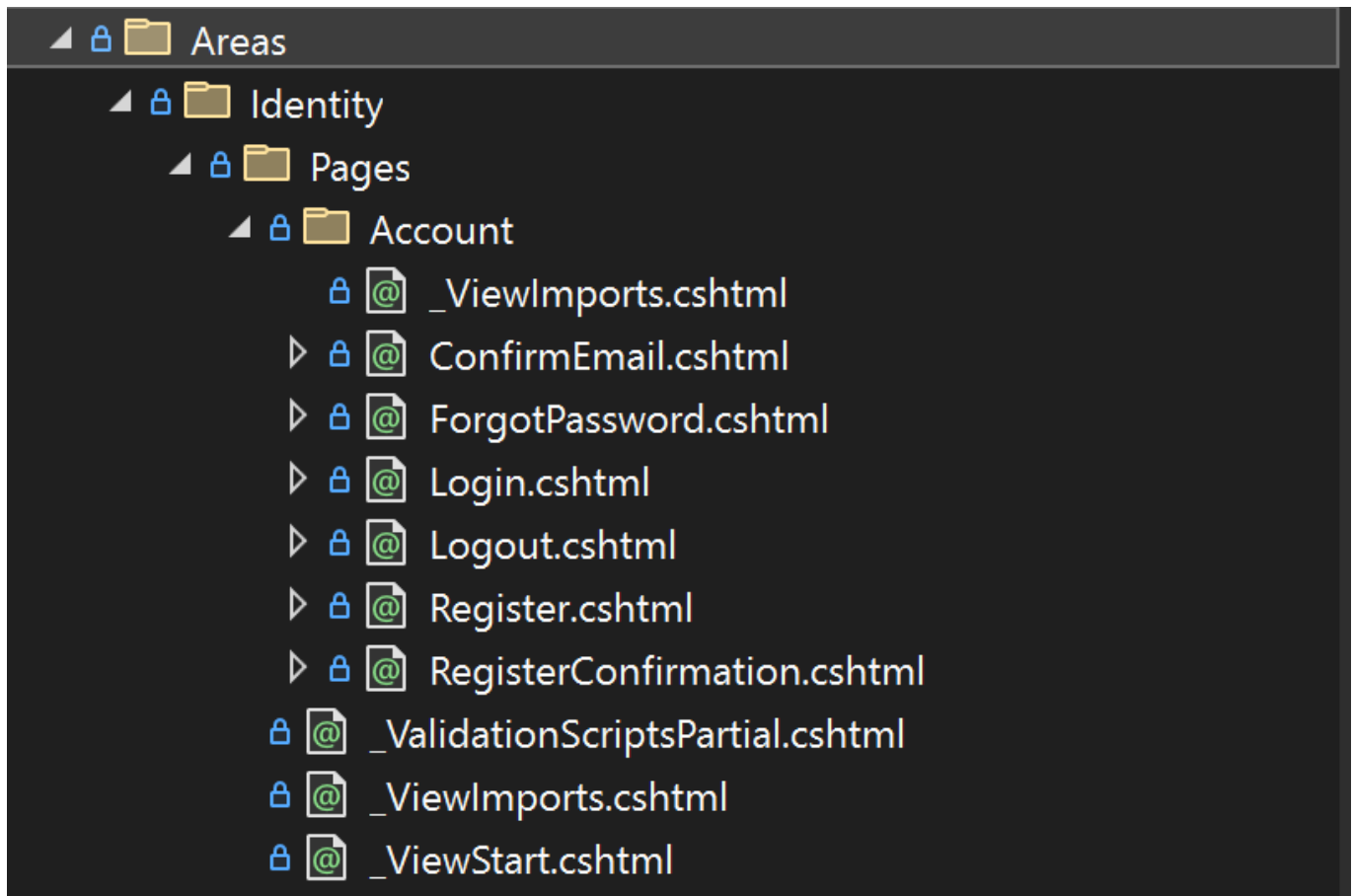
DbContext class: TripContext (Trips) +

Database provider: Configured from the selected DbContext

User class: +

Add Cancel

Co zostało wygenerowane :



Mapowanie mechanizmu uwierzytelniania, autoryzacji oraz mapowania stron blazor.

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

Przykład użycia udostępniania dostępu do danego kontrolera osobom niewierzytelnionym

```
[AllowAnonymous]  
public class TripsController : Controller
```

Index

[Create New](#)

From	To	StartTrip	EndTrip	Price	
Warszawa	Kraków	12.12.2021 00:00:00	13.12.2021 00:00:00	100	Edit Details Delete
Kraków	Warszawa	14.12.2021 00:00:00	15.12.2021 00:00:00	100	Edit Details Delete
Warszawa	Gdańsk	16.12.2021 00:00:00	17.12.2021 00:00:00	100	Edit Details Delete
Gdańsk	Warszawa	18.12.2021 00:00:00	19.12.2021 00:00:00	100	Edit Details Delete

© 2024 - Trips - [Privacy](#)

Index

[Create New](#)

From	To	StartTrip	EndTrip	Price	
Warszawa	Kraków	12.12.2021 00:00:00	13.12.2021 00:00:00	100	Edit Details Delete
Kraków	Warszawa	14.12.2021 00:00:00	15.12.2021 00:00:00	100	Edit Details Delete
Warszawa	Gdańsk	16.12.2021 00:00:00	17.12.2021 00:00:00	100	Edit Details Delete
Gdańsk	Warszawa	18.12.2021 00:00:00	19.12.2021 00:00:00	100	Edit Details Delete

© 2024 - Trips - [Privacy](#)

Jak widać użytkownik niezależnie od tego czy jest zalogowany czy nie widzi dokładnie to samo.

Przykład który niezalogowanemu użytkownikowi nie pozwoli zobaczyć rezerwacji:

```
[Authorize]
public class ReservationsController : Controller
```

Widok dla niezalogowanego użytkownika :

localhost:7133/Identity/Account/Login?ReturnUrl=%2FReservations

Wycieczki Rezerwacje Klienci Home

Register Login

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](#).

© 2024 - Trips - [Privacy](#)

Widok dla zalogowanego użytkownika :

localhost:7133/Reservations

Wycieczki Rezerwacje Klienci Home

Hello kacper@gmail.com! Logout

Index

[Create New](#)

IdClient	IdTrip	AmountOfPeople	ReservationDate	Status
244	54	2	02.06.2024 20:41:00	Edit Details Delete
1	1	123	25.05.2024 19:53:00	Edit Details Delete
33	13	33	24.05.2024 19:57:00	Edit Details Delete

© 2024 - Trips - [Privacy](#)

Wnioski

Ćwiczenie to pozwoliło na zrozumienie i praktyczne zastosowanie uwierzytelniania i autoryzacji w aplikacjach ASP.NET Core za pomocą ASPNET Core Identity. Zabezpieczenie dostępu do stron aplikacji poprzez implementację uwierzytelniania i autoryzacji jest kluczowym elementem w tworzeniu bezpiecznych aplikacji internetowych.