

Authentication and Authorization in Blazor, covering Blazor Server and Blazor WebAssembly (WASM)

1. Core Concepts (Very Important for Learners)

Authentication vs Authorization (Plain English)

Term	Meaning	Simple Example
Authentication	<i>Who are you?</i>	Login with username & password
Authorization	<i>What are you allowed to do?</i>	Only Admin can access /admin

👉 Authentication creates a **user identity**

👉 Authorization checks **permissions / roles / policies**

2. How Blazor Handles Security (Big Picture)

Two Hosting Models → Two Security Models

Feature	Blazor Server	Blazor WebAssembly
Execution	Runs on Server	Runs in Browser
State	Server memory	Client-side
Auth Style	Cookie-based (Identity)	Token-based (JWT)
Best For	Intranet, Admin apps	SPA, Public apps
Security	Very strong (server)	Depends on API security

PART A — BLAZOR SERVER AUTHENTICATION & AUTHORIZATION

3. Blazor Server Security Model

How it works internally

1. User opens Blazor Server app

2. User logs in
3. ASP.NET Core **Identity** validates credentials
4. Server issues **Authentication Cookie**
5. Cookie flows over **SignalR connection**
6. Blazor components read authentication state

👉 **No JWT required**

👉 **No tokens in browser storage**

👉 **Very secure**

4. Authentication in Blazor Server (Identity)

Step-by-Step Architecture

Core Components

Component	Role
ASP.NET Core Identity	User management
Cookie Authentication	Maintains login
AuthenticationStateProvider	Exposes user info
CascadingAuthenticationState	Shares auth state

5. Configure Authentication (Blazor Server)

Program.cs (Key Setup)

```
builder.Services.AddDbContext<AppDbContext>(options =>
    options.UseSqlServer(builder.Configuration.GetConnectionString("Default")));

builder.Services.AddIdentity<ApplicationUser, IdentityRole>()
    .AddEntityFrameworkStores<AppDbContext>()
    .AddDefaultTokenProviders();
```

```
builder.Services.AddAuthentication();  
builder.Services.AddAuthorization();  
  
builder.Services.AddRazorComponents()  
    .AddInteractiveServerComponents();
```

6. Enabling Authentication in UI

App.razor

```
<CascadingAuthenticationState>  
    <Router AppAssembly="@typeof(App).Assembly">  
        <Found Context="routeData">  
            <AuthorizeRouteView RouteData="@routeData"  
                DefaultLayout="@typeof(MainLayout)">  
                <NotAuthorized>  
                    <RedirectToLogin />  
                </NotAuthorized>  
            </AuthorizeRouteView>  
        </Found>  
    </Router>  
</CascadingAuthenticationState>
```

7. Authorization in Blazor Server

Route-Level Authorization

@attribute [Authorize]

Role-Based Authorization

```
@attribute [Authorize(Roles = "Admin")]
```

Policy-Based Authorization

```
builder.Services.AddAuthorization(options =>
```

```
{
```

```
    options.AddPolicy("AdminOnly",
```

```
        policy => policy.RequireRole("Admin"));
```

```
});
```

```
@attribute [Authorize(Policy = "AdminOnly")]
```

8. Access User Information in Components

```
@inject AuthenticationStateProvider AuthProvider
```

```
@code {
```

```
    protected override async Task OnInitializedAsync()
```

```
    {
```

```
        var state = await AuthProvider.GetAuthenticationStateAsync();
```

```
        var user = state.User;
```

```
        if (user.Identity.IsAuthenticated)
```

```
        {
```

```
            var name = user.Identity.Name;
```

```
        }
```

```
    }
```

```
}
```

9. Logout in Blazor Server

```
await SignInManager.SignOutAsync();
```

```
Navigation.NavigateTo("/login", true);
```

PART B — BLAZOR WEBASSEMBLY AUTHENTICATION & AUTHORIZATION

10. Why Blazor WASM Is Different

 **Blazor WASM runs in the browser**

Implications

Issue	Explanation
No server state	Browser can be refreshed
No cookies (by default) APIs are separate	
Needs token	JWT
Security	API must enforce

 **Authentication happens via API**

 **Authorization enforced at API**

11. Blazor WASM Security Architecture

Flow

1. User logs in
2. API validates credentials
3. API returns **JWT**
4. Blazor stores token (Memory / LocalStorage)
5. HTTP calls include Authorization: Bearer <token>
6. API validates token

12. Authentication in Blazor WASM (JWT)

Login Response (API)

```
{  
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXZXQ..."  
}
```

13. Store JWT in Blazor WASM

Token Service

```
public class TokenService
{
    public string? Token { get; private set; }

    public void SetToken(string token) => Token = token;

    public void Clear() => Token = null;
}
```

14. Custom AuthenticationStateProvider

```
public class JwtAuthStateProvider : AuthenticationStateProvider
{
    private readonly TokenService _tokenService;

    public JwtAuthStateProvider(TokenService tokenService)
    {
        _tokenService = tokenService;
    }
}
```

```

public override Task<AuthenticationState> GetAuthenticationStateAsync()
{
    if (string.IsNullOrEmpty(_tokenService.Token))
        return Task.FromResult(new AuthenticationState(
            new ClaimsPrincipal(new ClaimsIdentity())));

    var claims = JwtParser.ParseClaims(_tokenService.Token);

    var identity = new ClaimsIdentity(claims, "jwt");
    var user = new ClaimsPrincipal(identity);

    return Task.FromResult(new AuthenticationState(user));
}
}

```

15. Register Authentication in WASM

```

builder.Services.AddAuthorizationCore();

builder.Services.AddScoped<AuthenticationStateProvider, JwtAuthStateProvider>();

builder.Services.AddSingleton<TokenService>();

```

16. Attach JWT to HTTP Requests

```

builder.Services.AddHttpClient("API", client =>
{
    client.BaseAddress = new Uri("https://localhost:5001");
})

```

```
.AddHttpMessageHandler(sp =>
{
    var tokenService = sp.GetRequiredService<TokenService>();
    return new AuthHeaderHandler(tokenService);
});
```

17. Protecting Pages in Blazor WASM

@attribute [Authorize]

Role-Based

@attribute [Authorize(Roles = "Admin")]

18. API-Side Authorization (MANDATORY)

```
[Authorize]
[HttpGet("products")]
public IActionResult GetProducts()
{
    return Ok();
}
```

 **Never trust Blazor WASM alone**

✓ Always secure the API

PART C — COMPARISON & BEST PRACTICES

19. Blazor Server vs Blazor WASM

Area	Server	WASM
Auth Method	Cookie	JWT
Storage	Server	Browser
Security	Strong	Depends on API
Complexity	Lower	Higher
Offline	✗	✓
