

# CS6004NT

# Application Development

***WEEK - 07***

# ASP.NET Core Identity

Authentication and Authorization

# ASP.NET Core Identity

## Custom User Table/Entity

### 1. Create a IdentityUser derived class

```
2. using Microsoft.AspNetCore.Identity;

3. namespace BookReview.Api.Infrastructure.Identity;

4. public class AppUser: IdentityUser
5. {
6.     public string Name { get; set; } = string.Empty;
7.     public string Bio { get; set; } = string.Empty;
8.     public string ProfilePicture { get; set; } = string.Empty;
9.     public IEnumerable<IdentityRole> Roles { get; set; } = Enumerable.Empty<IdentityRole>();
10. }
```

AspNetUsers	
ABC	Id
ABC	UserName
ABC	NormalizedUserName
ABC	Email
ABC	NormalizedEmail
<input checked="" type="checkbox"/>	EmailConfirmed
ABC	PasswordHash
ABC	SecurityStamp
ABC	ConcurrencyStamp
ABC	PhoneNumber
<input checked="" type="checkbox"/>	PhoneNumberConfirmed
<input checked="" type="checkbox"/>	TwoFactorEnabled
	LockoutEnd
<input checked="" type="checkbox"/>	LockoutEnabled
123	AccessFailedCount
ABC	Bio
ABC	Name
ABC	ProfilePicture

### 11. Replace all occurrence of *IdentityUser* with *AppUser*

BookReviewDbContext.cs, Program.cs, SeedIdentityData.cs, etc.

### 1. Generate and apply the migration

# ASP.NET Core Identity

## Auth in Blazor Wasm

### AuthController.cs:

```
1. [Authorize]
2. [HttpGet("/api/auth/profile")]
3. public async Task<ActionResult<UserResponse>> GetProfileAsync()
4. {
5.     var userId = User.FindFirstValue(ClaimTypes.NameIdentifier);
6.     var user = await _authService.GetProfileAsync(userId);
7.     var userResponse = new UserResponse
8.     {
9.         Id = user.Id,
10.        Email = user.Email,
11.        EmailConfirmed = user.EmailConfirmed,
12.        Name = user.Name,
13.        Bio = user.Bio,
14.        ProfilePicture = user.ProfilePicture,
15.        Roles = User.FindAll(ClaimTypes.Role).Select(x => x.Value)
16.    };
17.    return Ok(userResponse);
18. }
```

# ASP.NET Core Identity

## Auth in Blazor Wasm

×

Create new bucket

Bucket name

BookReview

Storage Network

Sia (Private by default) ▾

Cancel

Create Bucket

```
19.// Allow cookie credentials to be in request and response
20.app.UseCors(
21.    policy =>
22.        policy
23.            .WithOrigins("http://localhost:3000", "https://localhost:3001")
24.            .AllowAnyMethod()
25.            .AllowAnyHeader()
26.            .AllowCredentials()
27.);

28.// In both Blazor and API to reduce the attack surface of Cross Site Scripting (XSS) attacks
29.// https://content-security-policy.com/
30.app.Use(async (ctx, next) =>
31.{
32.    ctx.Response.Headers.Add("Content-Security-Policy", "default-src 'self' ws: wss:
    https://localhost:3001 https://localhost:5001; script-src 'self' 'unsafe-inline' 'unsafe-eval'; img-src
    'self' data:; style-src 'self' 'unsafe-inline';");
33.    await next();
34.});
```

# ASP.NET Core Identity

## Auth in Blazor Wasm

### HttpClientExtension.cs:

```
1. namespace BookReview.BlazorWasm.Services;

2. public static class HttpClientExtension
3. {
4.     private static HttpContent Serialize(object data) => new
        StringContent(JsonSerializer.Serialize(data, new JsonSerializerOptions(JsonSerializerDefaults.Web)),
            Encoding.UTF8, "application/json");

5.     public static Task<HttpResponseMessage> AuthGetAsync(this HttpClient httpClient, string
        requestUri)
6.     {
7.         var request = new HttpRequestMessage(HttpMethod.Get, requestUri);
8.         request.SetBrowserRequestCredentials(BrowserRequestCredentials.Include);
9.         return httpClient.SendAsync(request);
10.    }
```

```

15.     public static Task<HttpResponseMessage> AuthPostAsync(this HttpClient httpClient, string requestUri,
        HttpContent? content)
16.     {
17.         var request = new HttpRequestMessage(HttpMethod.Post, requestUri);
18.         request.Content = content;
19.         request.SetBrowserRequestCredentials(BrowserRequestCredentials.Include);
20.         return httpClient.SendAsync(request);
21.     }

22.     public static Task<HttpResponseMessage> AuthPostAsJsonAsync<T>(this HttpClient httpClient, string
        requestUri, T data) where T : class
23.     {
24.         var request = new HttpRequestMessage(HttpMethod.Post, requestUri);
25.         request.Content = Serialize(data);
26.         request.SetBrowserRequestCredentials(BrowserRequestCredentials.Include);

27.         return httpClient.SendAsync(request);
28.     }
29.     ...
30. }

```



# ASP.NET Core Identity

## Auth in Blazor Wasm

### AuthService.cs (Blazor):

```
1. namespace BookReview.BlazorWasm.Services;
2. public class AuthService : IAuthService
3. {
4.     // DI and constructor code ...
5.     public async Task LoginAsync(LoginRequest loginRequest)
6.     {
7.         var response = await _httpClient.AuthPostAsJsonAsync("/api/auth/login", loginRequest);
8.         await CheckForErrorResponse(response);
9.     }
10.    public async Task<UserResponse?> GetProfileAsync()
11.    {
12.        var response = await _httpClient.AuthGetAsync("/api/auth/profile");
13.        await CheckForErrorResponse(response);
14.        var result = await response.Content.ReadFromJsonAsync<UserResponse>();
15.        return result;
16.    }
17.    ...
18. }
```

# ASP.NET Core Identity

## Auth in Blazor Wasm

### AppStateService.cs:

```
1. // Service to share state globally
2. namespace BookReview.BlazorWasm.Services;

3. public class AppStateService
4. {
5.     public UserResponse? CurrentUser { get; private set; }
6.     public void SetCurrentUser(UserResponse? value)
7.     {
8.         CurrentUser = value;
9.         OnStateChange?.Invoke();
10.    }
11.    public event Action? OnStateChange;
12.}
```

# ASP.NET Core Identity

## Auth in Blazor Wasm

### Program.cs (Blazor):

```
1. // Add AppStateService and AuthService to DI
2. builder.Services.AddSingleton<AppStateService>(); // Singleton to share same instance across the app
3. builder.Services.AddScoped<IAuthService, AuthService>();
```

### Login.razor:

```
1. @page "/login"
2. @inject IAuthService AuthService
3. @inject NavigationManager NavManager
4. @inject AppStateService StateService

5. <div class="login-body">
6.     <div class="form-login">
7.         <form class="text-center" @onsubmit="LoginHandler">
8. // Rest of the html for login form...

9. @code {
10. // properties for username, password, error message etc.
```

```

13. private async Task LoginHandler()
14. {
15.     try
16.     {
17.         _errorMessage = "";
18.         if (Username != null && Password != null)
19.         {
20.             var loginRequest = new LoginRequest
21.             {
22.                 Username = Username,
23.                 Password = Password,
24.                 RememberMe = RememberMe
25.             };
26.             await AuthService.LoginAsync(loginRequest);
27.             var currentUser = await AuthService.GetProfileAsync();
28.             StateService.SetCurrentUser(currentUser);
29.             NavManager.NavigateTo("/");
30.         }
31.     }
32.     catch (Exception e)
33.     {
34.         _errorMessage = e.Message;
35.     }
36. }
37. }

```

## Index.razor:

```

1. @page "/"
2. @inject NavigationManager NavManager

3. @code {
4.     protected override void OnInitialized()
5.     {
6.         NavManager.NavigateTo("/books");
7.     }
8. }

```

# ASP.NET Core Identity

## Auth in Blazor Wasm

### MainLayout.razor:

```
1. @inherits LayoutComponentBase
2. @inject AppStateService StateService
3. @inject IAuthService AuthService

4. // layout html codes...
5. @code {
6.     protected override async Task OnInitializedAsync()
7.     {
8.         try
9.         { // If user is logged-in, fetch profile and set it to currentUser
10.             var currentUser = await AuthService.GetProfileAsync();
11.             StateService.SetCurrentUser(currentUser);
12.             StateHasChanged(); // Notifies the component that its state has changed
13.         }
14.         catch
15.         {
16.             // ignored
17.         }
18.     }
19. }
```

# ASP.NET Core Identity

## Auth in Blazor Wasm

### Use of OnStateChange Delegate:

```
1. // In a page where auth is required. Example: create/edit book page

2. protected override void OnInitialized()
3. {
4.     StateService.OnStateChange += StateService_StateChanged; // When StateService.SetCurrentUser is invoked
5. }

6. private void StateService_StateChanged()
7. {
8.     StateHasChanged(); // Notifies the component that its state has changed.
9.     if (StateService.CurrentUser == null)
10.    {
11.        NavManager.NavigateTo("/books");
12.    }
13. }
```

# ASP.NET Core Identity

## Auth in Blazor Wasm

### Role-based Access in Blazor:

```
1. @if (StateService.CurrentUser?.Roles.Contains("Admin") ?? false)
2. {
3.     <a class="edit" title="Edit" role="button" href="/edit-book/@(book.Id)">
4.         <span class="oi oi-pencil"></span>
5.     </a>
6.     <button class="delete" title="Delete" type="button" @onclick="() => OnDeleteBook(book)">
7.         <span class="oi oi-delete"></span>
8.     </button>
9. }
```

# ASP.NET Core Identity

## Auth in Blazor Wasm

### Role-based Access in API Controller:

```
1. [Authorize(Roles = "Admin")]
2. [HttpDelete("/api/books/{id:int}")]
3. public async Task<IActionResult> DeleteBookAsync(int id)
4. {
5.     await _bookService.DeleteBookAsync(id);
6.     return NoContent();
7. }
```



# ASP.NET Core Identity

## Auth in Blazor Wasm

→ ↺ https://localhost:3001

Book Reviews

Please

Username  
admin@user.cc

Password  
.....

☒ Remember me

→ ↺ https://localhost:3001/books

Book Reviews

Books

Title	Author	Year	Actions
Dracula	Bram Stoker	1897	0 ★
Dune (Dune, #1)	Frank Herbert	1965	0 ★
East of Eden	John Steinbeck	1952	0 ★
Emma	Jane Austen	1815	0 ★
Fahrenheit 451	Ray Bradbury	1953	0 ★
Five Little Pigs (Hercule Poirot, #25)	Agatha Christie	1942	0 ★
Flowers for Algernon	Daniel Keyes	1959	0 ★

→ ↺ https://localhost:3001/books

Book Reviews

Home Login

Books

Title	Author	Year	Actions
Dracula	Bram Stoker	1897	0 ★
Dune (Dune, #1)	Frank Herbert	1965	0 ★
East of Eden	John Steinbeck	1952	0 ★
Emma	Jane Austen	1815	0 ★
Fahrenheit 451	Ray Bradbury	1953	0 ★
Five Little Pigs (Hercule Poirot, #25)	Agatha Christie	1942	0 ★
Flowers for Algernon	Daniel Keyes	1959	0 ★

# EF Core

Entity Framework Core

# EF Core

## Pagination

### Book Reviews

[Home](#) [Login](#)

### Books

Title	Author	Year	Actions
Around the World in Eighty Days	Jules Verne	1872	0 ★
As a Man Thinketh	James Allen	1902	0 ★
Brave New World	Aldous Huxley	1932	0 ★
Breakfast at Tiffany's and Three Stories	Truman Capote	1958	0 ★
Candide	Voltaire	1759	0 ★

[Previous](#)[1](#)[2](#)[3](#)[4](#)[5](#)[Next](#)

Per Page:

[5](#)[10](#)[15](#)[20](#)

# EF Core

## Pagination

### PaginatedResponse.cs:

```
1. namespace BookReview.Contracts;

2. public class PaginatedResponse<T>
3. {
4.     public int PageNumber { get; set; }
5.     public int PageSize { get; set; }
6.     public int TotalPages { get; set; }
7.     public int TotalRecords { get; set; }
8.     public IEnumerable<T> Data { get; set; } = Enumerable.Empty<T>();
9. }
```

### PaginationFilter.cs:

```
1. namespace BookReview.Contracts;

2. public class PaginationFilter
3. {
4.     public int PageNumber { get; set; } = 1;
5.     public int PageSize { get; set; } = 10;
6. }
```

# EF Core

## Pagination

### BookController.cs:

```
1. [HttpGet("/api/books")]
2. public async Task<ActionResult<PaginatedResponse<BookResponse>>> GetAllBooksAsync([FromQuery] PaginationFilter filter)
3. {
4.     var totalRecords = await _bookService.CountAsync();
5.     var books = await _bookService.GetAllBooksAsync(filter.PageNumber, filter.PageSize);
6.     var response = new PaginatedResponse<BookResponse>
7.     {
8.         PageNumber = filter.PageNumber,
9.         PageSize = filter.PageSize,
10.        TotalPages = (int)Math.Ceiling((double)totalRecords / filter.PageSize),
11.        TotalRecords = totalRecords,
12.        Data = books.Select(x => new BookResponse
13.        {
14.            // Property Mapping Id, Title, Year, AuthorName etc.
15.        })
16.    };
17.    return Ok(response);
18. }
```

# EF Core

## Pagination

### BookRepository.cs:

```
1. public async Task<IEnumerable<Book>> GetAllWithDetailsPaginatedAsync(int pageNumber, int
   pageSize)
2. {
3.     var result = await _dbContext.Books
4.         .Include(x => x.Author)
5.         .Include(x => x.Reviews)
6.         .Skip((pageNumber - 1) * pageSize)
7.         .Take(pageSize)
8.         .ToListAsync();
9.     return result;
10. }
```

# EF Core

## Order By

```
1 var methodSyntax = _context.Books
2     .OrderBy(x => x.PublishedOn)
3     .ThenBy(x => x.Title);
4
5 var querySyntax = from b in _context.Books
6     orderby b.PublishedOn, b.Title
7     select b;
```

	123 BookId 🔽↑	ABC Title 🔽↑	🕒 PublishedOn 🔽↑	ABC Description
1	4	If Beale Street Could Talk	1974-01-01	In this honest and stunning novel, J
2	2	Surreal Numbers	1974-01-01	Nearly 30 years ago, John Horton C
3	3	Where the Sidewalk Ends	1974-01-01	Where the Sidewalk Ends turns fort
4	1	The Da Vinci Code	2003-03-18	While in Paris on business, Harvard

# EF Core

## Order By

```
1 var methodSyntax = _context.Books
2     .OrderByDescending(x => x.PublishedOn)
3     .ThenByDescending(x => x.Title);
4
5 var querySyntax = from b in _context.Books
6     orderby b.PublishedOn descending, b.Title descending
7     select b;
```





	123 BookId 🔼🔼	ABC Title 🔼🔼	🕒 PublishedOn 🔼🔼	ABC Description
1	1	The Da Vinci Code	2003-03-18	While in Paris on business, Harvard
2	3	Where the Sidewalk Ends	1974-01-01	Where the Sidewalk Ends turns fo
3	2	Surreal Numbers	1974-01-01	Nearly 30 years ago, John Horton
4	4	If Beale Street Could Talk	1974-01-01	In this honest and stunning novel,



# EF Core

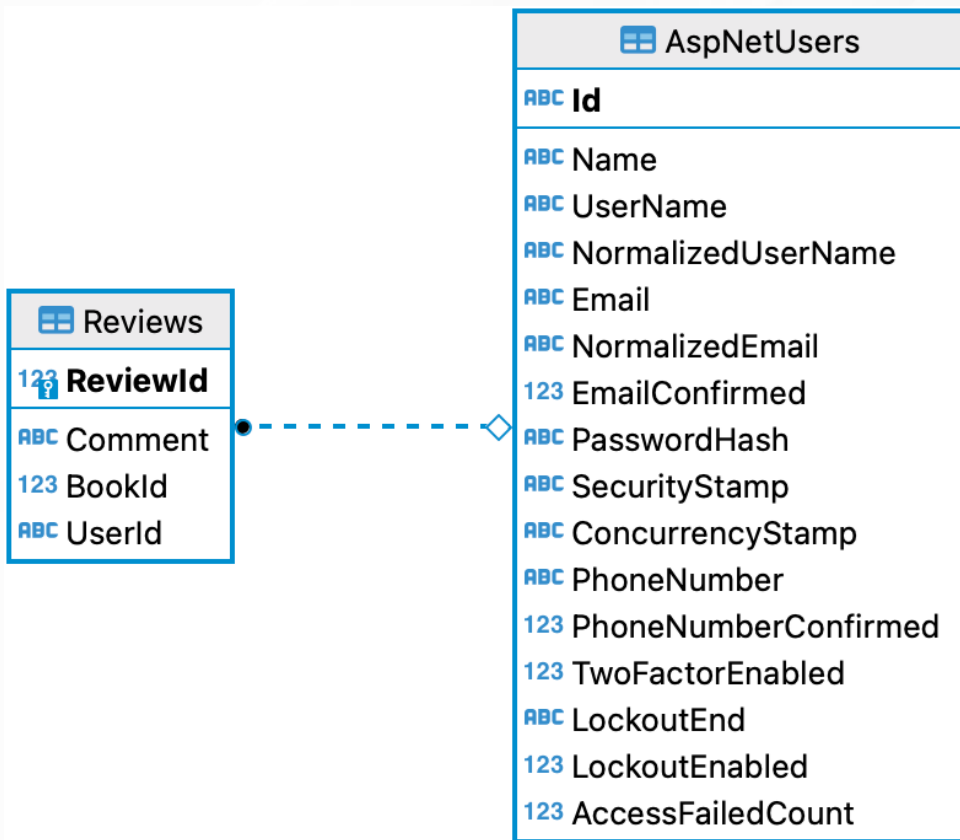
## Group By

```
1 var methodSyntax = _context.Books
2     .GroupBy(x => x.PublishedOn)
3     .Select(x => new {
4         PublishedOn = x.Key,
5         Count = x.Count()
6     });
7
8 var querySyntax = from b in _context.Books
9     group b by b.PublishedOn into x
10    select new {
11        PublishedOn = x.Key,
12        Count = x.Count()
13    };
```

	 PublishedOn 	 Count 
1	1974-01-01	3
2	2003-03-18	1

# EF Core

## Join



```
1 var methodSyntax = _context.Reviews
2     .Join(_context.Users, r => r.UserId, u => u.Id, (r, u) => new {
3         UserName = u.UserName,
4         Comment = r.Comment
5     });
6
7 var querySyntax = from r in _context.Reviews
8     join u in _context.Users on r.UserId equals u.Id
9     select new {
10         UserName = u.UserName,
11         Comment = r.Comment
12     };
```

	ABC UserName	ABC Comment
1	John Doe	You have a good head on
2	Himalay Sunuwar	On a scale from 1 to 10,
3	John Doe	There's ordinary, and the
4	Himalay Sunuwar	You should be proud of y
5	John Doe	You could survive a Zom
6	Himalay Sunuwar	On a scale from 1 to 10,
7	Himalay Sunuwar	You are making a differ

# EF Core

## Join with Multiple Columns

```
1 var methodSyntax = _context.Books
2     .Join( context.Reviews,
3         b => new { b.BookId, b.User.Id },
4         r => new { r.Book.BookId, r.User.Id },
5         (b, r) => new {
6             b.BookId,
7             b.Title,
8             r.User.Name,
9             r.Comment
10        });
11
12 var querySyntax = from b in _context.Books
13     join r in _context.Reviews
14     on new { b.BookId, b.User.Id }
15     equals new { r.Book.BookId, r.User.Id }
16     select new {
17         b.BookId,
18         b.Title,
19         r.User.Name,
20         r.Comment
21     };
```

# Thank You