Applying Authorization Policies

Roland Guijt

Freelance Trainer and Consultant | Microsoft MVP

@rolandguijt | roland.guijt@gmail.com

Authorization Data Sources

Claims from identity cookie/ClaimsPrincipal

User data stored on application level

Authorization API

Authentication Methods and Authorization

Authentication type doesn't matter

Pure cookies, Identity or using an OpenID Connect identity provider

All authorization is done on the client, not on the identity provider

An authorization policy is a centralized way to define authorization rules

[Authorize] Hierarchy

```
[Authorize]
public class ConferenceController: Controller

        [Authorize(Policy = "CanAddConference")]
        public Task<IActionResult> Add()
```

FallbackPolicy and DefaultPolicy

FallbackPolicy is triggered when no authorize attribute is present

DefaultPolicy is triggered when there is an authorize attribute without a policy

Razor Pages and Fine-grained Authorization

Inject IAuthorizationService and check the policy in code

Use an MVC controller for the part of the application you want finer grained control

Split the content across multiple pages and share partial views

Create a filter that does the authorization https://4sh.nl/razorfilter

Always apply authorization to the endpoints too!

A More Complex Policy

Speakers may only add a new proposal when they have more than the specified years of experience

Calculate years of experience using CareerStarted claim

Compare calculated years with specified number of years

Requirements and Handlers

YearsOfExperienceRequirement

YearsOfExperience = 5

AuthorizationHandler<YearsOfExperienceRequirement>	Succeed Fail Do nothing
AuthorizationHandler<YearsOfExperienceRequirement>	Succeed Fail Do nothing

Logic Around Multiple Handlers and/or Policies

If at least one of the handlers succeed and others don't have a response, access is granted

If no handler calls Succeed, access is denied

If one of them calls Fail, access is denied

Calling Fail explicitly denies access, no matter what other handlers and policies do

But they are still called

Multiple Handlers

Badge

Visitor sticker

Try This Out For Yourself!

Heads up: only when users have the speaker role AND a career that started more than 10 years ago, the add link will be displayed

See what happens if you change the careerstarted claim

Clear the browser cookies and login again after each change

Resource-based Policies

Authorization that depends on the state of an application object

Example: editing a proposal should only be allowed is the proposal's speaker is the logged in user and if it's not approved yet

Can be used to do these kind of checks in a centralized way

Further Experimentation

More complex requirement

Watch what happens if the speaker name is changed

Revoking Access

Default Lifetime of a Cookie

Destroyed after session ends

Can be set to persist

Lifetime: 14 days

Sliding expiration

Example: Handling the OnValidatePrincipal Event

```
var userRepository =
    context.HttpContext.RequestServices.GetRequiredService<IUserRepository>();
var userPrincipal = context.Principal;

var lastChanged = (from c in userPrincipal.Claims
                       where c.Type == "LastUpdated"
                       select c.Value).FirstOrDefault();

if (string.IsNullOrEmpty(lastChanged) ||
    !userRepository.ValidateLastChanged(userPrincipal, lastChanged))
{
    context.RejectPrincipal();
    await context.HttpContext.Authentication.SignOutAsync("Cookies");
}
```

Thanks for watching!

Roland Guijt

Freelance Trainer and Consultant | Microsoft MVP

@rolandguijt | roland.guijt@gmail.com