NDC { London } 2023

Flexible Authorisation with ASP.NET Core

Jason Taylor
Solution Architect

Increment.
the δifference

Jason Taylor

Solution Architect at Increment

Developer, Mentor, Speaker & Trainer

Microsoft MVP in Developer Technologies

Working with .NET since 1.0 in 2002

jasontaylordev

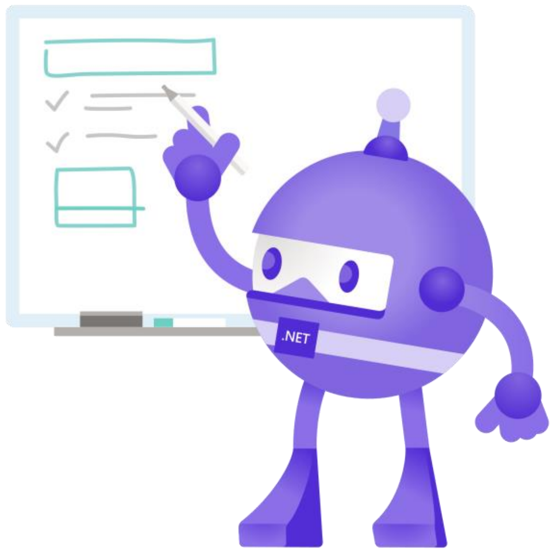
github.com/jasontaylordev

jasontaylor.dev

youtube.com/jasontaylordev

Increment.

Contents



Increment.
the δifference

Background

Introduction

- Authentication and authorisation
- Authorisation determines access to resources
- Includes defining and enforcing access control policies
- The focus for of this talk?

Increment.

Authorisation in ASP.NET Core

δ

Basic Authorisation

- Supports simple authorisation capabilities
- Includes sophisticated authorisation capabilities such as role-based, claims-based, and policy-based
- Easy to create custom authorisation policies
- Authorisation middleware also customisable

Increment.

Simple Authorisation

Basic Authorisation

- Apply [Authorize] to a controller, action, or Razor page

```
[Authorize]
public class AccountController : Controller
{
        public ActionResult Login()
        {
        }

        public ActionResult Logout()
        {
        }
}
```

Increment.

Simple Authorisation

Basic Authorisation

- Apply [Authorize] to a controller, action, or Razor page

```
public class AccountController : Controller
{
    public ActionResult Login()
    {
    }

    [Authorize]
    public ActionResult Logout()
    {
    }
}
```

Simple Authorisation

Basic Authorisation

- Apply [Authorize] to a controller, action, or Razor page

```
[Authorize]
public class AccountController : Controller
{
    [AllowAnonymous]
    public ActionResult Login()
    {
    }

    public ActionResult Logout()
    {
    }
}
```

Authorisation with Roles

Basic Authorisation

- Role-based authorisation

```
[Authorize(Roles = "Administrators")]
public class AdministratorController : Controller
{
	public ActionResult Index()
	{
	}
}
```

Increment.

Authorisation with Roles

Basic Authorisation

- Role-based authorisation

```
[Authorize(Roles = "Administrators, Accounts")]
public class AccountsController : Controller
{
	public ActionResult Index()
	{
	}
}
```

Increment.

Authorisation with Roles

δ

- Role-based authorisation

```
[Authorize(Roles = "Administrators")]
[Authorize(Roles = "Accounts")]
public class AccountsController : Controller
{
        public ActionResult Index()
        {
        }
}
```

Increment.

Authorisation with Policies

Basic Authorisation

- Policy-based authorisation

```
[Authorize(Policy = "EmployeesOnly")]
public class EmployeesController : Controller
{
        public ActionResult Index()
        {
        }
}`
```

Increment.

Authorisation with Policies

Basic Authorisation

- Policy-based authorisation

```
builder.Services.AddAuthorization(options =>
{
        options.AddPolicy("EmployeesOnly",
                policy => policy.RequireClaim("EmployeeNumber"));
}
```

Increment.

Authorisation in Blazor WebAssembly

Basic Authorisation

- Supports basic authorisation capabilities
- Includes sophisticated authorisation capabilities such as role-based, claims-based, and policy-based
- Includes helpful authorisation components AuthorizeView and AuthorizeRouteView
- Authorisation is only used for UI/UX

Increment.

Simple Authorisation

Basic Authorisation

- Apply [Authorize] to a routable page

```
@page "/"
@attribute [Authorize]

<PageTitle>Home</PageTitle>
```

Increment.

Simple Authorisation

Basic Authorisation

- Apply [AllowAnonymous] to a routable page

```
@page "/login"
@attribute [AllowAnonymous]

<PageTitle>Login</PageTitle>
```

Authorisation with Roles

Basic Authorisation

- Role-based authorisation

```
@page "/counter"
@attribute [Authorize(Roles = "Administrators, Accounts")]

<PageTitle>Counter</PageTitle>
```

Increment.

Authorisation with Policies

δ

Basic Authorisation

- Policy-based authorisation

```
@page "/users"
@attribute [Authorize(Policy = "EmployeesOnly")]

<PageTitle>Users</PageTitle>
```

Increment.

Authorisation with Views

Basic Authorisation

- Apply <AuthorizeView> to selectively display content

```
<AuthorizeView>
        <NavLink href="/">Home</NavLink>
</AuthorizeView>

<AuthorizeView Roles="Administrators, Accounts">
        <NavLink href="counter">Counter</NavLink>
</AuthorizeView>

<AuthorizeView Policy="EmployeesOnly">
        <NavLink href="users">Users</NavLink>
</AuthorizeView>
```

Increment.

Authorisation with Views

Basic Authorisation

- Apply <AuthorizeView> to selectively display content

```
<AuthorizeView>
        <NotAuthorized>
                  <NavLink href="login">Login</NavLink>
        </NotAuthorized>
</AuthorizeView>
```

Increment.

Demo: Adding a new role

Basic Authorisation

- Adding a new role to an existing application

Increment.

Flexible Authorisation with ASP.NET Core

δ

- Easily add new roles and configure access control
- Easily reconfigure access control for existing roles
- Remove roles without impacting existing access control checks
- Easily view access control policies
- Support all of the above as standard application features

Increment.

Demo: Adding a new role

Flexible Authorisation

- Adding a new role to an existing application

Increment.

Basic vs. Flexible Approach

Basic

- Code changes
- Testing
- Build & deploy
- Bug fixes
- Documentation

Flexible

- Free time 🏝

Demo: Adding a new permission

Flexible Authorisation

- Adding a permission to support a new requirement

Increment.

Overview

Flexible Authorisation

- Create necessary roles and permissions
- Assign permissions to roles as required
- Add permissions claim to authenticated user
- Add permissions-based access control as required
- Enforced by dynamic authorisation policies

Managing Roles

Flexible Authorisation

- Roles define a logical grouping of users
- Administrators create new roles as required
- Administrators assign permissions to roles
- Role-based authorisation should be avoided

Managing Permissions

Flexible Authorisation

- Permissions define granular access to resources
- Developers create permissions as necessary
- Permissions are not assigned directly to a user
- Permissions are assigned to a role
- Users inherit the permissions of any assigned roles

Increment.

Authorisation Engine

δ

Flexible Authorisation

- Built entirely using policy-based authorisation
- Using a custom [Authorize] attribute, developers specify required permissions
- Required permissions are translated into a policy name
- Policy will be dynamically created using a custom policy provider

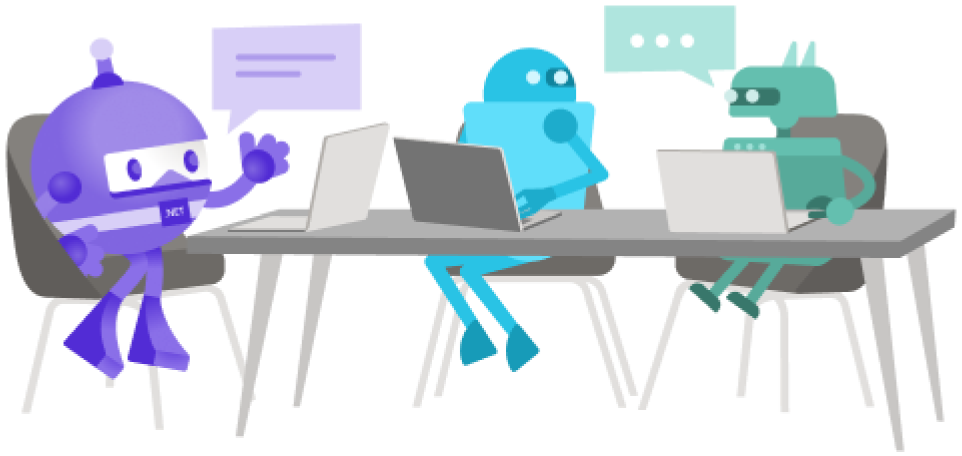
Increment.

Demo: Code Review

δ

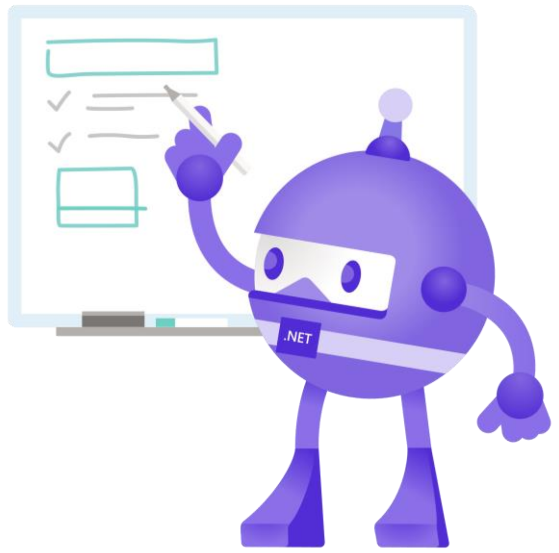
Flexible Authorisation

- An overview of the flexible authorisation engine



Increment.

Summary

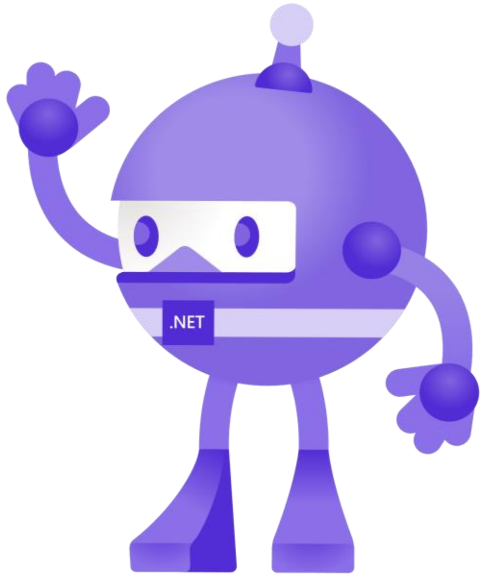


Increment.
the δifference

Next Steps

Summary

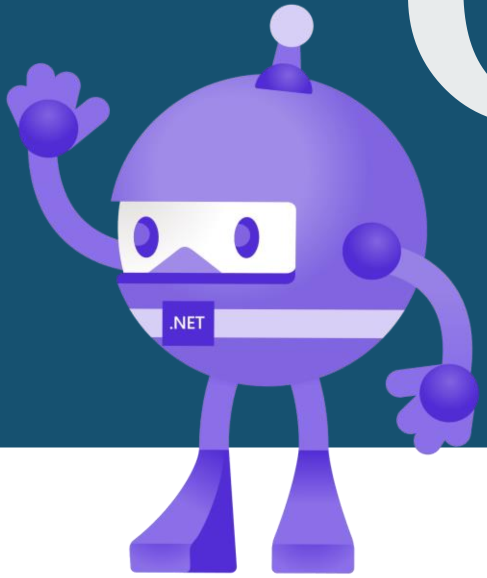
- Access the code and slides on GitHub: https://github.com/jasontaylordev/flexible-aspnetcore-authorization



Increment.

Thank you!

@JasonTaylorDev



Increment.
the δifference