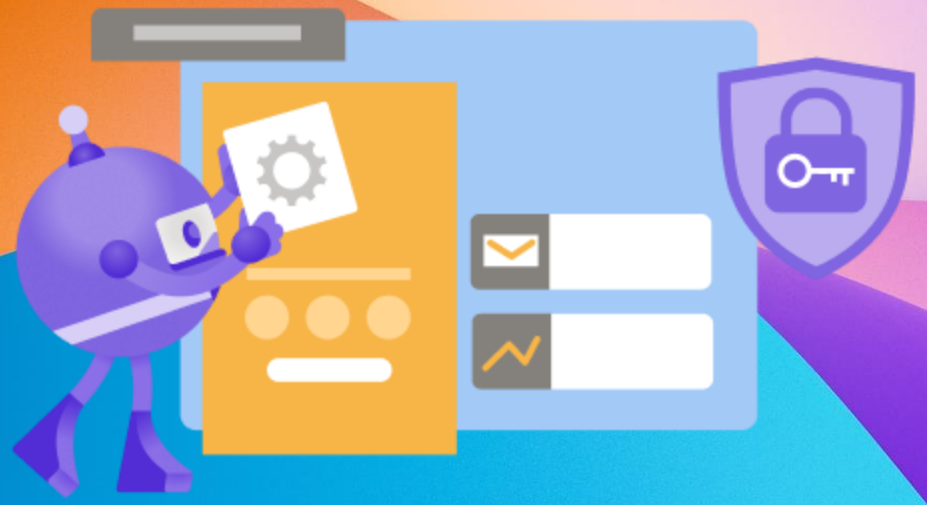
ENTERPRISE SOFTWARE DEVELOPMENT

Join the Conversation #AspNetCore @JasonTaylorDev



Developing Flexible Authorization Capabilities with ASP.NET Core

JasonTaylorDev | .NET User Group | May 2022

Join the Conversation #AspNetCore @JasonTaylorDev

Jason Taylor

SSW Solution Architect

Developer, Mentor, Speaker & Trainer

Brisbane, Australia

Working with .NET since 1.0 in 2002

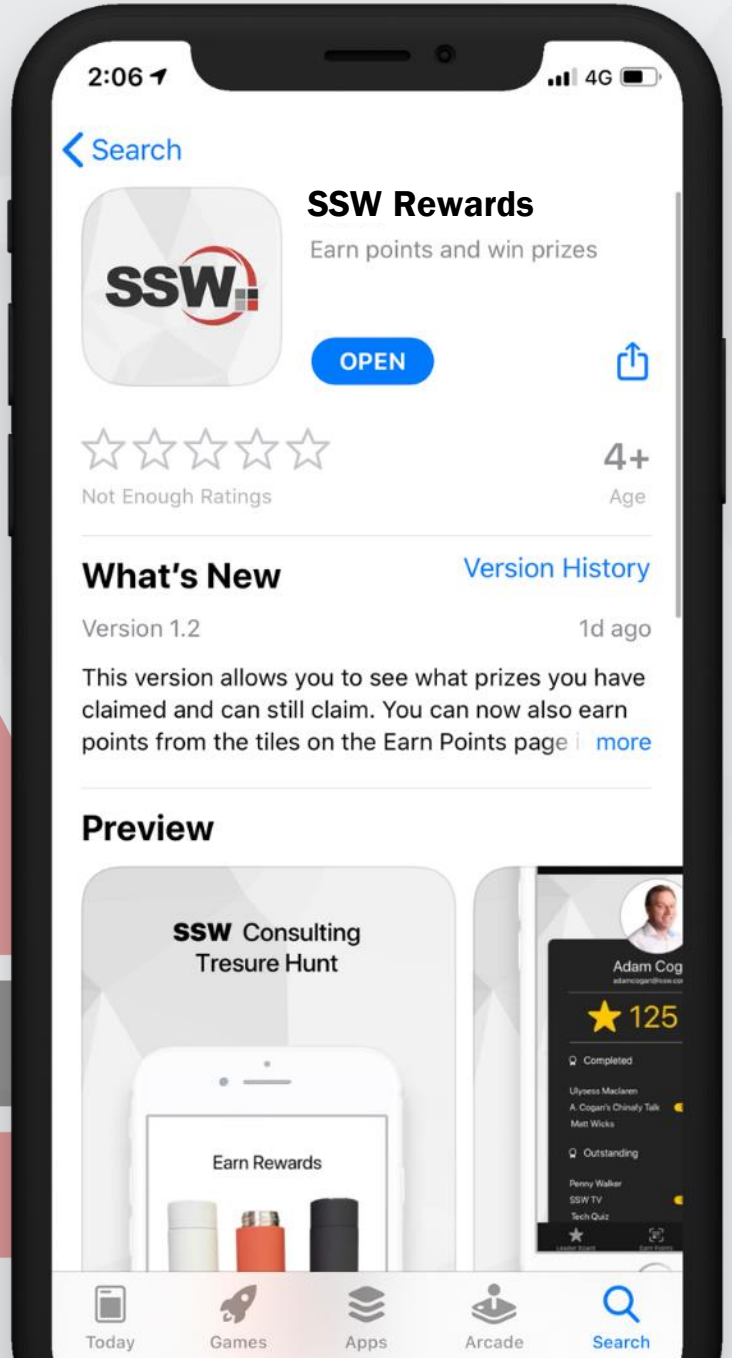
jasontaylordev

github.com/jasontaylordev

jasontaylor.dev

youtube.com/jasontaylordev

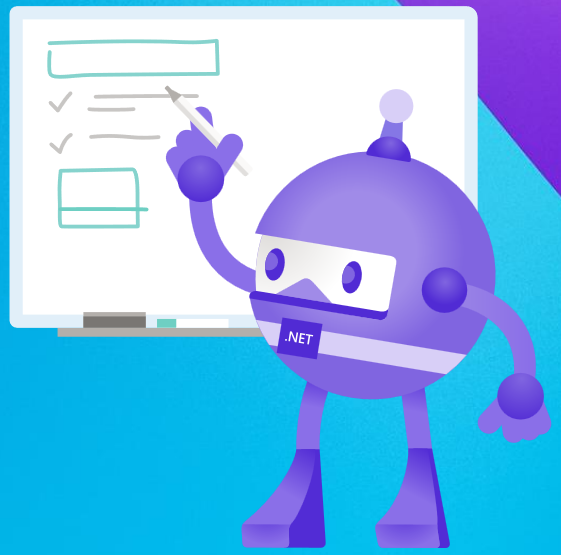
Join the Conversation #AspNetCore @JasonTaylorDev



1. Scan QR Code to install app
2. Scan SSW codes at talks and events
3. Earn **SSW Points** ★
   claim rewards, and win prizes

Join the Conversation | #AspNetCore | @JasonTaylorDev

Agenda



- Background
- Authorization in ASP.NET Core
- Authorization in Blazor WebAssembly
- Flexible Authorization with ASP.NET Core
- Code Walkthrough
- Summary

Introduction

Authentication and Authorization

Authorization determines access to resources

Includes defining and enforcing access control policies

The focus for of this talk?

Authorization in ASP.NET Core

Supports simple authorization capabilities

Includes sophisticated authorization capabilities such as role-based, claims-based, and policy-based

Easy to create custom authorization policies

Authorization middleware also customizable

Authorization in ASP.NET Core

Apply **`[Authorize]`** to a controller, action, or Razor page:

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

        public ActionResult Logout()
        {
        }
}
```

Authorization in ASP.NET Core

Apply **`[Authorize]`** to a controller, action, or Razor page:

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

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

Authorization in ASP.NET Core

Apply **`[Authorize]`** to a controller, action, or Razor page:

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

        public ActionResult Logout()
        {
        }
}
```

Authorization in ASP.NET Core

Role-based authorization:

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

Authorization in ASP.NET Core

Role-based authorization:

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

Authorization in ASP.NET Core

Role-based authorization:

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

Authorization in ASP.NET Core

Policy-based authorization:

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

Authorization in ASP.NET Core

Policy-based authorization:

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

Authorization in Blazor WebAssembly

Supports simple authorization capabilities

Includes sophisticated authorization capabilities such as role-based, claims-based, and policy-based

Includes authorization components AuthorizeView and AuthorizeRouteView

Authorization is only used for UI/UX

Authorization in Blazor WebAssembly

Apply **`[Authorize]`** to a routable page:

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

<PageTitle>Home</PageTitle>
```

Authorization in Blazor WebAssembly

Apply **`[Authorize]`** to a routable page:

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

<PageTitle>Counter</PageTitle>
```

Authorization in Blazor WebAssembly

Apply **`[Authorize]`** to a routable page:

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

<PageTitle>Users</PageTitle>
```

Authorization in Blazor WebAssembly

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>
```

Authorization in Blazor WebAssembly

Apply **`<AuthorizeView>`** to selectively display content:

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

Demo: Adding a new role

Adding a new role to an existing application.

Flexible Authorization with ASP.NET Core

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

Demo: Adding a new role

Adding a new role to an existing application.

Standard vs Flexible Approach

Standard

- ✗ Code changes
- ✗ Testing
- ✗ Build & deploy
- ✗ Documentation
- ✗ Bug fixes (maybe)

Flexible

- ✓ Free time 

Flexible Authorization - Permissions

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

Flexible Authorization - Roles

Role define a logical grouping of users

Administrators create new roles as required

Administrators assign permissions to roles

Role-based authorization should be avoided

Flexible Authorization - Engine

Built entirely using policy-based Authorization

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

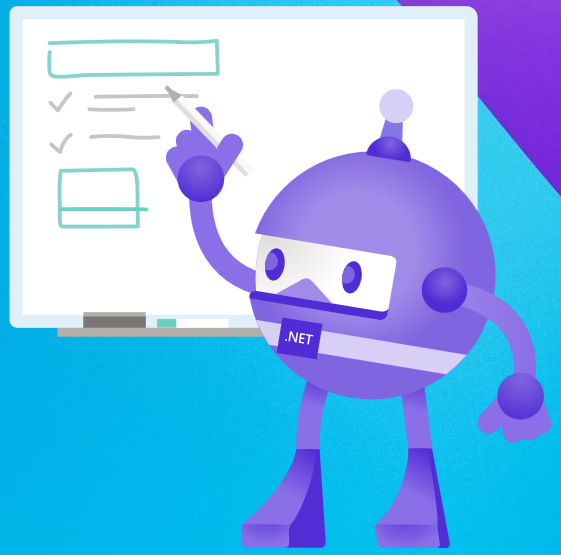
Code Walkthrough

Let's take a look at how this works under the hood.

Demo: Adding a new permission

Adding a permission to support a new requirement.

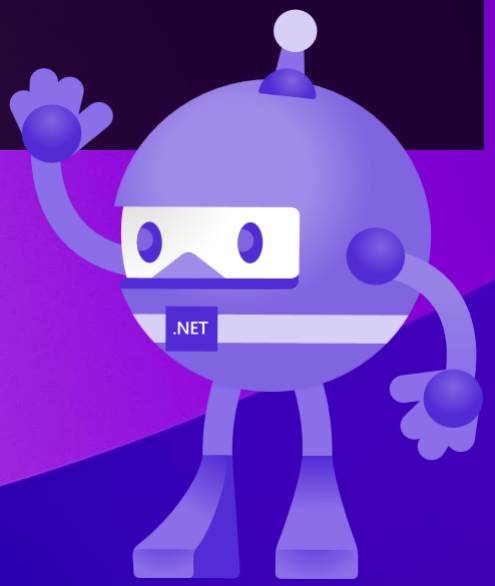
Summary



- Background
- Authorization in ASP.NET Core
- Authorization in Blazor WebAssembly
- Flexible Authorization with ASP.NET Core
- Code Walkthrough
- Summary

Find the code and slides GitHub!

github.com/jasontaylordev/flexible-aspnetcore-authorization



Scan the QR code below and earn **SSW Points** ★ for watching this talk!

Developing Flexible Authorization Capabilities in ASP.NET Core

Join the Conversation #AspNetCore @JasonTaylorDev

Thank you!

info@ssw.com.au

www.ssw.com.au

Sydney | Melbourne | Brisbane

Got feedback?

Scan the QR code below to fill our survey.

Developing Flexible Authorization Capabilities in ASP.NET Core

Join the Conversation #AspNetCore @JasonTaylorDev