**Add multiple roles in a policy**

builder.Services.AddAuthorization(options =>

{

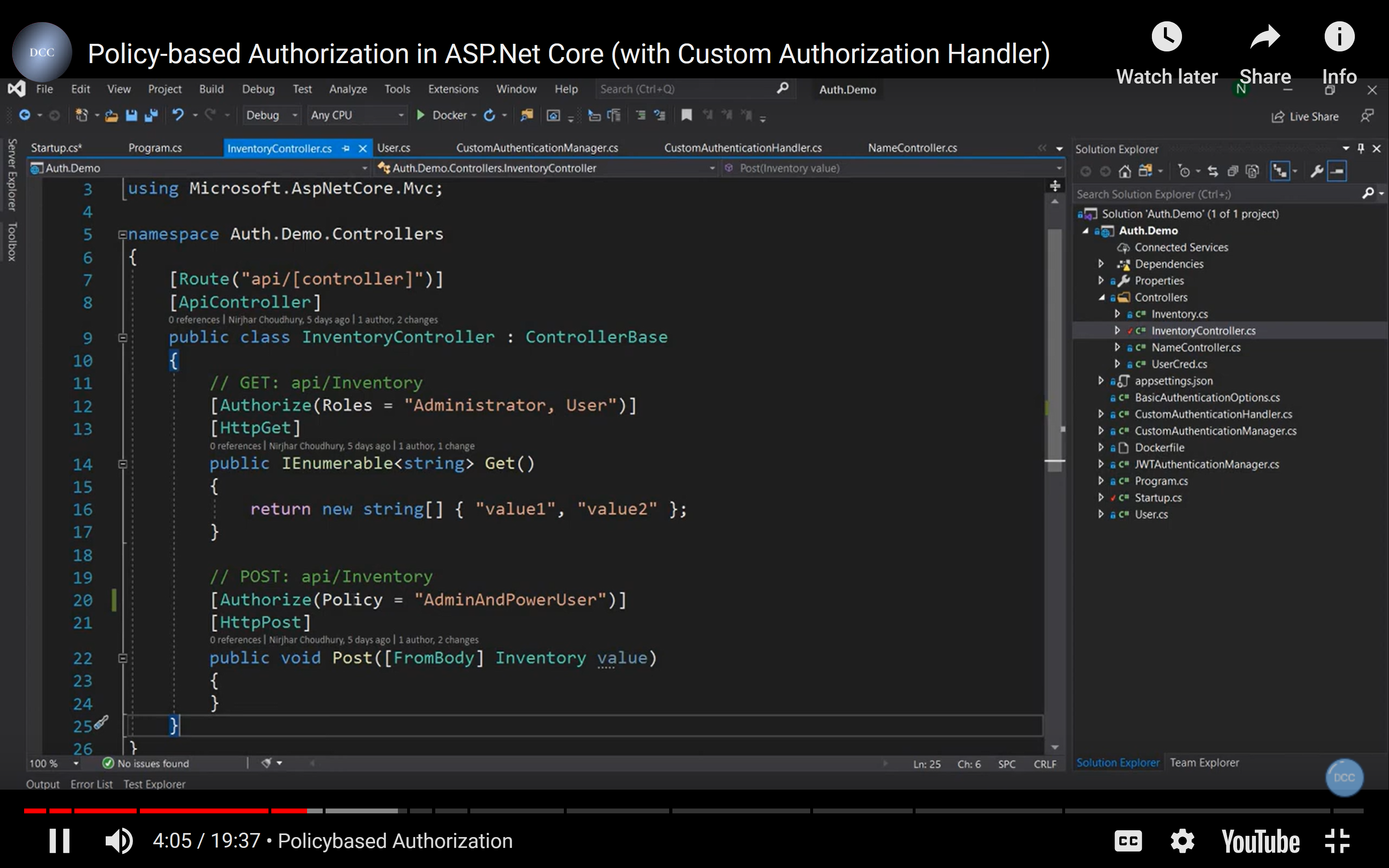
options.AddPolicy("AdminAndManager", policy =>

{

policy.RequireRole("Admin", "Manager");

});

});



// Custom Authorization Handler

First Create Requirement then create handler

Only Employees who have exp more than 20 years can do post

First > Create any requirement, that class has to derive from IAuthorizationRequirement

Add a constructor

Take num of years as a parameter

Epxose years as a property

using Microsoft.AspNetCore.Authorization;

namespace WebApi.Policy

{

public class EmployeeWIthMoreThan20YearsExperience : IAuthorizationRequirement

{

public int Years { get; set; }

public EmployeeWIthMoreThan20YearsExperience(int years)

{

Years = years;

}

}

}

Regioster in Program.cs file

builder.Services.AddAuthorization(options =>

{

options.AddPolicy("AdminAndManager", policy =>

{

policy.RequireRole("Admin", "Manager");

});

options.AddPolicy("EmployeeWIthMoreThan20YearsExperience", policy =>

{

policy.Requirements.Add(new EmployeeWIthMoreThan20YearsExperience(20));

});

});

// Add a provide who will give number of experience in years

// Just hardcoded

namespace WebApi.Policy

{

public class EmployeeNumberofYears

{

public int NumberOfyears()

{

return 20;

}

}

}

// Register this in Program.cs file

builder.Services.AddSingleton<EmployeeNumberofYears>();

Now we need AuthorizationHandler

Derive from AuthorizationHandler

using Microsoft.AspNetCore.Authorization;

namespace WebApi.Policy

{

public class EmployeeWithMoreThan20YearsHandler : AuthorizationHandler<EmployeeWIthMoreThan20YearsExperience>

{

protected override Task HandleRequirementAsync(AuthorizationHandlerContext context, EmployeeWIthMoreThan20YearsExperience requirement)

{

throw new NotImplementedException();

}

}

}

Inject provider

using Microsoft.AspNetCore.Authorization;

using System.Security.Claims;

namespace WebApi.Policy

{

public class A : AuthorizationHandler<EmployeeWIthMoreThan20YearsExperience>

{

protected override Task HandleRequirementAsync(AuthorizationHandlerContext context, EmployeeWIthMoreThan20YearsExperience requirement)

{

throw new NotImplementedException();

}

}

public class EmployeeWithMoreThan20YearsHandler : AuthorizationHandler<EmployeeWIthMoreThan20YearsExperience>

{

EmployeeNumberofYears \_emp;

public EmployeeWithMoreThan20YearsHandler(EmployeeNumberofYears emp)

{

\_emp = emp;

}

protected override Task HandleRequirementAsync(AuthorizationHandlerContext context,

EmployeeWIthMoreThan20YearsExperience requirement

)

{

var years = \_emp.NumberOfyears();

if (years >= requirement.Years)

{

context.Succeed(requirement);

}

return Task.FromResult(0);

}

}

}