**NetCore-CustomAuthorizationFilter-And-JWT-Middleware-To-Validate-Token**

1. **Jwtmiddledware.cs**

namespace WebApplication1.Filters

{

public class JwtMiddleware

{

private readonly RequestDelegate \_next;

private readonly AppSettings \_appSettings;

public JwtMiddleware(RequestDelegate next, IOptions<AppSettings> appSettings)

{

\_next = next;

\_appSettings = appSettings.Value;

}

public async Task Invoke(HttpContext context, IUserService userService)

{

var token = context.Request.Headers["Authorization"].FirstOrDefault()?.Split(" ").Last();

if (token != null)

attachUserToContext(context, userService, token);

await \_next(context);

}

private void attachUserToContext(HttpContext context, IUserService userService, string token)

{

try

{

var tokenHandler = new JwtSecurityTokenHandler();

var key = Encoding.ASCII.GetBytes(\_appSettings.Secret);

tokenHandler.ValidateToken(token, new TokenValidationParameters

{

ValidateIssuerSigningKey = true,

IssuerSigningKey = new SymmetricSecurityKey(key),

ValidateIssuer = false,

ValidateAudience = false,

// set clockskew to zero so tokens expire exactly at token expiration time (instead of 5 minutes later)

ClockSkew = TimeSpan.Zero

}, out SecurityToken validatedToken);

var jwtToken = (JwtSecurityToken)validatedToken;

var userId = int.Parse(jwtToken.Claims.First(x => x.Type == "id").Value);

// attach user to context on successful jwt validation

context.Items["User"] = userService.GetById(userId);

}

catch

{

// do nothing if jwt validation fails

// user is not attached to context so request won't have access to secure routes

}

}

}

}

**=================================================================**

1. **CustomAuthorizationFilterAttribute.cs**

namespace WebApplication1.Filters

{

public class CustomAuthorizationFilterAttribute : Attribute, IAuthorizationFilter

{

public void OnAuthorization(AuthorizationFilterContext context)

{

var user = (User)context.HttpContext.Items["User"];

if (user == null)

{

// not logged in

context.Result = new JsonResult(new { message = "Unauthorized" }) { StatusCode = StatusCodes.Status401Unauthorized };

}

}

}

}

=============================================================================

1. **Startup.cs**

namespace WebApplication1

{

public class Startup

{

public Startup(IConfiguration configuration)

{

Configuration = configuration;

}

public IConfiguration Configuration { get; }

// This method gets called by the runtime. Use this method to add services to the container.

public void ConfigureServices(IServiceCollection services)

{

services.AddControllers();

services.AddMvc(options =>

{

options.Filters.Add(typeof(CustomAuthorizationFilterAttribute));

});

}

// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

{

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

}

app.UseHttpsRedirection();

app.UseRouting();

app.UseMiddleware<JwtMiddleware>();

app.UseAuthorization();

app.UseEndpoints(endpoints =>

{

endpoints.MapControllers();

});

}

}

}

**=================================================================**

1. **Controller.cs**

namespace WebApplication1.Controllers

{

//[TypeFilter(typeof(CustomAuthorizationFilterAttribute))]

[CustomAuthorizationFilterAttribute]

[ApiController]

[Route("[controller]")]

public class WeatherForecastController : ControllerBase

{

private static readonly string[] Summaries = new[]

{

"Freezing", "Bracing", "Chilly", "Cool", "Mild", "Warm", "Balmy", "Hot", "Sweltering", "Scorching"

};

private readonly ILogger<WeatherForecastController> \_logger;

public WeatherForecastController(ILogger<WeatherForecastController> logger)

{

\_logger = logger;

}

[HttpGet]

public IEnumerable<WeatherForecast> Get()

{

var rng = new Random();

return Enumerable.Range(1, 5).Select(index => new WeatherForecast

{

Date = DateTime.Now.AddDays(index),

TemperatureC = rng.Next(-20, 55),

Summary = Summaries[rng.Next(Summaries.Length)]

})

.ToArray();

}

}