1. WebAPI1-HandsOn:

2. WebAPI-2 HandsOn:

using Microsoft.OpenApi.Models;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllers();

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddControllers();

// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new OpenApiInfo

{

Title = "Swagger Demo",

Version = "v1",

Description = "TBD",

//TermsOfService = "None",

Contact = new OpenApiContact { Name = "John Doe", Email = "john@xyzmail.com", Url = new Uri("http://localhost:5255/swagger") },

License = new OpenApiLicense { Name = "License Terms", Url = new Uri("http://localhost:5255/swagger") }

});

});

var app = builder.Build();

// Configure the HTTP request pipeline.

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI(c =>

{

c.SwaggerEndpoint("/swagger/v1/swagger.json", "Swagger Demo");

});

}

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();

3. WebAPI-3 HandsOn:

CustomExceptionFilter.cs:

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

namespace WebAPI3.Filters

{

public class CustomAuthFilter : ActionFilterAttribute

{

public override void OnActionExecuting(ActionExecutingContext context)

{

if (!context.HttpContext.Request.Headers.TryGetValue("Authorization", out var authHeader))

{

context.Result = new BadRequestObjectResult("Invalid request - No Auth token");

return;

}

if (!authHeader.ToString().Contains("Bearer", StringComparison.OrdinalIgnoreCase))

{

context.Result = new BadRequestObjectResult("Invalid request - Token present but Bearer unavailable");

}

}

}

}

CustomAuthFilter.cs:

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

namespace WebAPI3.Filters

{

public class CustomAuthFilter : ActionFilterAttribute

{

public override void OnActionExecuting(ActionExecutingContext context)

{

if (!context.HttpContext.Request.Headers.TryGetValue("Authorization", out var authHeader))

{

context.Result = new BadRequestObjectResult("Invalid request - No Auth token");

return;

}

if (!authHeader.ToString().Contains("Bearer", StringComparison.OrdinalIgnoreCase))

{

context.Result = new BadRequestObjectResult("Invalid request - Token present but Bearer unavailable");

}

}

}

}

Employee.cs:

public class Employee

{

public int Id { get; set; }

public string Name { get; set; }

public int Salary { get; set; }

public bool Permanent { get; set; }

public Department Department { get; set; }

public List<Skill> Skills { get; set; }

public DateTime DateOfBirth { get; set; }

}

public class Skill

{

public int Id { get; set; }

public string Name { get; set; }

}

public class Department

{

public int Id { get; set; }

public string Name { get; set; }

}

4. WebAPI-4 HandsOn:

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using WebAPI3.Filters;

[ApiController]

[Route("[controller]")]

[AllowAnonymous]

[CustomAuthFilter]

public class EmployeeController : ControllerBase

{

private List<Employee> \_employees;

public EmployeeController()

{

\_employees = GetStandardEmployeeList();

}

private List<Employee> GetStandardEmployeeList()

{

return new List<Employee>

{

new Employee

{

Id = 1,

Name = "Alice",

Salary = 50000,

Permanent = true,

Department = new Department { Id = 1, Name = "HR" },

Skills = new List<Skill>

{

new Skill { Id = 1, Name = "C#" },

new Skill { Id = 2, Name = "SQL" }

},

DateOfBirth = new DateTime(1990, 1, 1)

}

};

}

[HttpGet("throw")]

[ProducesResponseType(StatusCodes.Status500InternalServerError)]

public IActionResult ThrowException()

{

throw new Exception("Simulated error");

}

[HttpPut("{id}")]

public ActionResult<Employee> UpdateEmployee(int id, [FromBody] Employee updatedEmp)

{

if (id <= 0)

{

return BadRequest("Invalid employee id");

}

var employee = \_employees.FirstOrDefault(e => e.Id == id);

if (employee == null)

{

return BadRequest("Invalid employee id");

}

employee.Name = updatedEmp.Name;

employee.Salary = updatedEmp.Salary;

employee.Permanent = updatedEmp.Permanent;

employee.Department = updatedEmp.Department;

employee.Skills = updatedEmp.Skills;

employee.DateOfBirth = updatedEmp.DateOfBirth;

return Ok(employee);

}

[HttpPost]

public IActionResult AddEmployee([FromBody] Employee emp)

{

\_employees.Add(emp);

return Ok(emp);

}

}

5. WebAPI-5 HandsOn:

Program.CS:

using System.Text;

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.IdentityModel.Tokens;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllers();

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

builder.Services.AddCors(options =>

{

options.AddPolicy("AllowLocalhost",

builder => builder.WithOrigins("http://localhost:5188")

.AllowAnyHeader()

.AllowAnyMethod());

});

string key = "mysuperdupersecretkeyfordotnettraining";

var securityKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(key));

builder.Services.AddAuthentication(x =>

{

x.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;

x.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;

})

.AddJwtBearer(x =>

{

x.TokenValidationParameters = new TokenValidationParameters

{

ValidateIssuer = true,

ValidateAudience = true,

ValidateLifetime = true,

ValidateIssuerSigningKey = true,

ValidIssuer = "mySystem",

ValidAudience = "myUsers",

IssuerSigningKey = securityKey

};

});

var app = builder.Build();

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

app.UseHttpsRedirection();

app.UseCors("AllowLocalhost");

app.UseAuthentication();

app.UseAuthorization();

app.MapControllers();

app.Run();

AuthController.cs:  
using System.IdentityModel.Tokens.Jwt;

using System.Security.Claims;

using System.Text;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using Microsoft.IdentityModel.Tokens;

[ApiController]

[Route("auth")]

[AllowAnonymous]

public class AuthController : ControllerBase

{

[HttpGet("generate-token")]

public IActionResult GetToken()

{

var token = GenerateJSONWebToken(1, "Admin");

return Ok(token);

}

private string GenerateJSONWebToken(int userId, string userRole)

{

var key = new SymmetricSecurityKey(Encoding.UTF8.GetBytes("mysuperdupersecretkeyfordotnettraining"));

var creds = new SigningCredentials(key, SecurityAlgorithms.HmacSha256);

var claims = new List<Claim>

{

new Claim(ClaimTypes.Role, userRole),

new Claim("UserId", userId.ToString())

};

var token = new JwtSecurityToken(

issuer: "mySystem",

audience: "myUsers",

claims: claims,

expires: DateTime.Now.AddMinutes(10),

signingCredentials: creds

);

return new JwtSecurityTokenHandler().WriteToken(token);

}

}

EmployeeController.CS:

using System.Data;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using WebAPI3.Filters;

[ApiController]

[Route("[controller]")]

[Authorize(Roles = "POC")]

[CustomAuthFilter]

public class EmployeeController : ControllerBase

{

private List<Employee> \_employees;

public EmployeeController()

{

\_employees = GetStandardEmployeeList();

}

private List<Employee> GetStandardEmployeeList()

{

return new List<Employee>

{

new Employee

{

Id = 1,

Name = "Alice",

Salary = 50000,

Permanent = true,

Department = new Department { Id = 1, Name = "HR" },

Skills = new List<Skill>

{

new Skill { Id = 1, Name = "C#" },

new Skill { Id = 2, Name = "SQL" }

},

DateOfBirth = new DateTime(1990, 1, 1)

}

};

}

[HttpGet("throw")]

[ProducesResponseType(StatusCodes.Status500InternalServerError)]

public IActionResult ThrowException()

{

throw new Exception("Simulated error");

}

[HttpPut("{id}")]

public ActionResult<Employee> UpdateEmployee(int id, [FromBody] Employee updatedEmp)

{

if (id <= 0)

{

return BadRequest("Invalid employee id");

}

var employee = \_employees.FirstOrDefault(e => e.Id == id);

if (employee == null)

{

return BadRequest("Invalid employee id");

}

employee.Name = updatedEmp.Name;

employee.Salary = updatedEmp.Salary;

employee.Permanent = updatedEmp.Permanent;

employee.Department = updatedEmp.Department;

employee.Skills = updatedEmp.Skills;

employee.DateOfBirth = updatedEmp.DateOfBirth;

return Ok(employee);

}

[HttpPost]

public IActionResult AddEmployee([FromBody] Employee emp)

{

\_employees.Add(emp);

return Ok(emp);

}

}