Aman Raj

Superset ID: 6358186

1. **ASP.NET Core 8.0 Web API**

**Employee.cs**

using System;

using System.Collections.Generic;

namespace EmployeeWebAPI.Models

{

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; }

}

}

**EmployeeController.cs**

using Microsoft.AspNetCore.Mvc;

using EmployeeWebAPI.Models;

using EmployeeWebAPI.Filters;

namespace EmployeeWebAPI.Controllers

{

[ApiController]

[Route("api/[controller]")]

[CustomAuthFilter]

public class EmployeeController : ControllerBase

{

private readonly List<Employee> \_employees;

public EmployeeController()

{

\_employees = GetStandardEmployeeList();

}

private List<Employee> GetStandardEmployeeList()

{

return new List<Employee>

{

new Employee

{

Id = 1,

Name = "Alice",

Salary = 60000,

Permanent = true,

DateOfBirth = new DateTime(1990, 5, 12),

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

Skills = new List<Skill>

{

new Skill { Id = 1, Name = "Communication" },

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

}

}

};

}

[HttpGet]

[ProducesResponseType(typeof(List<Employee>), 200)]

[ProducesResponseType(500)]

public ActionResult<List<Employee>> Get()

{

// throw new Exception("Forced exception for test"); // Uncomment to test exception filter

return Ok(\_employees);

}

[HttpPost]

public ActionResult<Employee> Post([FromBody] Employee emp)

{

emp.Id = \_employees.Max(e => e.Id) + 1;

\_employees.Add(emp);

return CreatedAtAction(nameof(Get), new { id = emp.Id }, emp);

}

[HttpPut("{id}")]

public IActionResult Put(int id, [FromBody] Employee emp)

{

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

if (existing == null) return NotFound();

existing.Name = emp.Name;

existing.Salary = emp.Salary;

existing.Permanent = emp.Permanent;

existing.DateOfBirth = emp.DateOfBirth;

existing.Department = emp.Department;

existing.Skills = emp.Skills;

return NoContent();

}

}

}

**CustomAuthFilter.cs**

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

using System.Linq;

namespace EmployeeWebAPI.Filters

{

public class CustomAuthFilter : ActionFilterAttribute

{

public override void OnActionExecuting(ActionExecutingContext context)

{

var headers = context.HttpContext.Request.Headers;

if (!headers.ContainsKey("Authorization"))

{

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

return;

}

var token = headers["Authorization"].ToString();

if (!token.Contains("Bearer"))

{

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

}

}

}

}

**CustomExceptionFilter.cs**

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

using System.IO;

namespace EmployeeWebAPI.Filters

{

public class CustomExceptionFilter : IExceptionFilter

{

public void OnException(ExceptionContext context)

{

var exception = context.Exception;

var logPath = Path.Combine(Directory.GetCurrentDirectory(), "logs.txt");

File.AppendAllText(logPath, $"{DateTime.Now} - {exception.Message}{Environment.NewLine}");

context.Result = new ObjectResult("Internal Server Error. Please contact support.")

{

StatusCode = 500

};

}

}

}

**Program.cs**

using Microsoft.OpenApi.Models;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllers();

// Add Swagger configuration here

builder.Services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new OpenApiInfo

{

Title = "Employee API",

Version = "v1"

});

// Add Authorization header support

c.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme

{

Name = "Authorization",

Type = SecuritySchemeType.ApiKey,

Scheme = "Bearer",

BearerFormat = "JWT",

In = ParameterLocation.Header,

Description = "Enter 'Bearer {your\_token}'"

});

c.AddSecurityRequirement(new OpenApiSecurityRequirement

{

{

new OpenApiSecurityScheme

{

Reference = new OpenApiReference

{

Type = ReferenceType.SecurityScheme,

Id = "Bearer"

}

},

Array.Empty<string>()

}

});

});

var app = builder.Build();

// Configure middleware

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();

**Output:-**



