*3.WebApiHandson*

**Web Api using custom model class**

Create a Custom class ‘Employee’ of the below defined structure

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; }

}

Create a new controller - EmployeeController with Read Write actions

Constructor: Create few records, HTTPGet, HTTPPost/HTTPPut

Create a Private method GetStandardEmployeeList that returns a List of Employee class. Invoke this method in the Get action method of the EmployeeController that was created in the previous step.

Public ActionResult<Employee> GetStandrad()

Modify the return type of the Get action method(without parameter) to return List of Employee class object

Add ProducesResponseType to the GET action method for Status code 200

Check the Swagger description for the GET method for success status code

using Microsoft.AspNetCore.Mvc;

using EmployeeWebApi.Models;

using System;

using System.Collections.Generic;

using System.Linq;

namespace EmployeeWebApi.Controllers

{

    [ApiController]

    [Route("api/[controller]")]

    public class EmployeeController : ControllerBase

    {

        private readonly List<Employee> \_employees;

        public EmployeeController()

        {

            \_employees = new List<Employee>

            {

                new Employee

                {

                    Id = 1,

                    Name = "John Doe",

                    Salary = 50000,

                    Permanent = true,

                    Department = new Department { Id = 1, Name = "IT", Location = "NY" },

                    Skills = new List<Skill>

                    {

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

                    },

                    DateOfBirth = new DateTime(1990, 1, 1)

                }

            };

        }

        private List<Employee> GetStandardEmployeeList()

        {

            return \_employees;

        }

        [HttpGet]

        [ProducesResponseType(200)]

        public ActionResult<List<Employee>> Get()

        {

            return Ok(GetStandardEmployeeList());

        }

        [HttpGet("GetStandard")]

        [ProducesResponseType(200)]

        public ActionResult<Employee> GetStandard()

        {

            return Ok(\_employees.FirstOrDefault());

        }

        [HttpPost]

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

        {

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

            \_employees.Add(employee);

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

        }

        [HttpPut("{id}")]

        public ActionResult<Employee> Put(int id, [FromBody] Employee employee)

        {

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

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

            existing.Name = employee.Name;

            existing.Salary = employee.Salary;

            existing.Permanent = employee.Permanent;

            existing.Department = employee.Department;

            existing.Skills = employee.Skills;

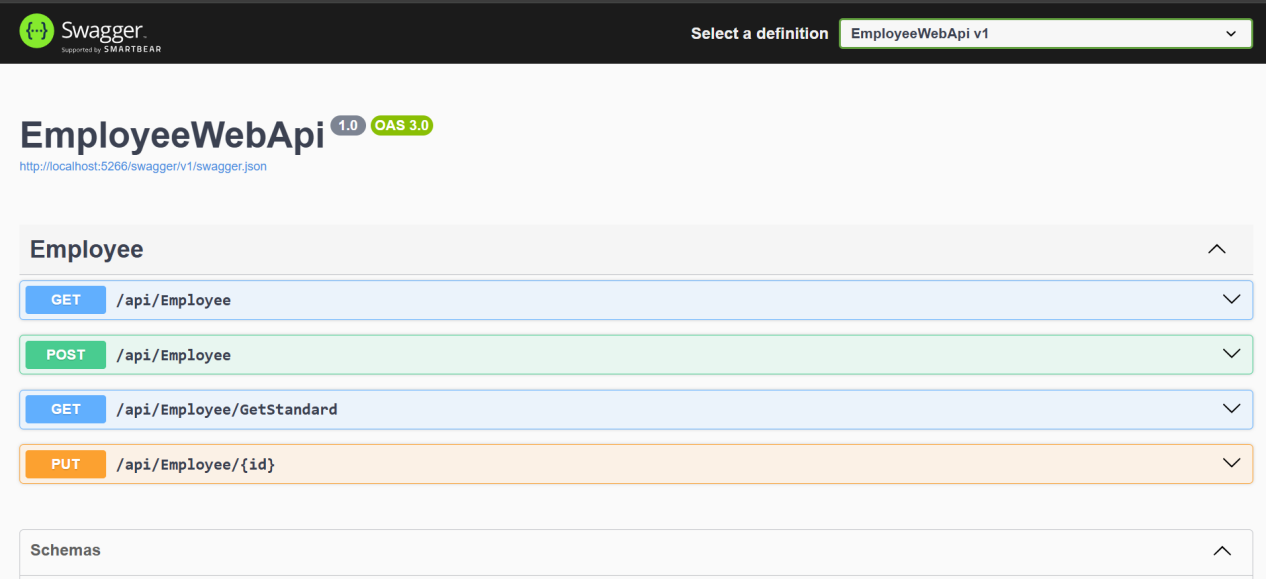
            existing.DateOfBirth = employee.DateOfBirth;

            return Ok(existing);

        }

    }

}



**Create a Custom action filter for Authorization.**

The requirement is to intercept incoming requests and check if there is a key ‘Authorization’ in the request header or not. If it is there, then to check if it contains a value ‘Bearer’ or not.

Create a folder ‘Filters’ in the application solution. Create a class ‘**CustomAuthFilter**’ to filter requests. Inherit ActionFilterAttribute. Override OnActionExecuting method to check if the request object has Header ‘Authorization’ or not. If not, throw BadRequestResult with the message

Invalid request - No Auth token

If the header is present, then check if the value contains the word ‘Bearer’. If not, throw BadRequestResult with the message

Invalid request - Token present but Bearer unavailable

Add an attribute **CustomAuthFilter** to the Employee controller to filter any request to check for the Authorization token in the request header.

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

namespace EmployeeWebApi.Filters

{

    public class CustomAuthFilter : ActionFilterAttribute

    {

        public override void OnActionExecuting(ActionExecutingContext context)

        {

            var request = context.HttpContext.Request;

*// Check if Authorization header exists*

            if (!request.Headers.ContainsKey("Authorization"))

            {

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

                return;

            }

*// Check if Authorization header contains "Bearer"*

            var authHeader = request.Headers["Authorization"].ToString();

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

            {

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

                return;

            }

            base.OnActionExecuting(context);

        }

    }

}



**Custom Exception filter**

Create a class ‘CustomExceptionFilter’ to catch the exceptions occuring the application. Implement IExceptionFilter thru the OnException method  
  
Use the exception context to fetch the exception detail. Capture that and write it to a File in the system.  
  
Set the Result property of the exception context to ExceptionResult.  
  
Throw an exception in GET action method.  
Ensure that the GET action method has ProducesResponseType for 500 - Internal server error  
  
Use Swagger to test the exception and message being thrown.  
  
Note: This needs WebApiCompatShim NuGet package installation

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

using System;

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", "exceptions.log");

            var logDirectory = Path.GetDirectoryName(logPath);

            if (!Directory.Exists(logDirectory))

            {

                Directory.CreateDirectory(logDirectory);

            }

            var logEntry = $"[{DateTime.Now}] {exception.GetType().Name}: {exception.Message}\n{exception.StackTrace}\n--------------------------\n";

            File.AppendAllText(logPath, logEntry);

            context.Result = new ObjectResult(new { error = "An error occurred while processing your request." })

            {

                StatusCode = 500

            };

            context.ExceptionHandled = true;

        }

    }

}