Lab 3 - Web API using Custom Model, Authorization & Exception Filters (.NET 9)

# Objectives

• Demonstrate creation of an Action method to return list of custom class entity  
• Explain the usage of FromBody attribute to read model object from request (not query string)  
• Demonstrate Custom Authorization Filter  
• Demonstrate Custom Exception Filter using IExceptionFilter and WebApiCompatShim (if applicable)

# 1. Model Class Creation

Inside the Models folder, create the following classes with proper namespaces:

Skill.cs

namespace FirstWebApi.Models;  
  
public class Skill  
{  
 public int Id { get; set; }  
 public string Name { get; set; }  
}

Department.cs

namespace FirstWebApi.Models;  
  
public class Department  
{  
 public int Id { get; set; }  
 public string Name { get; set; }  
}

Employee.cs

namespace FirstWebApi.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; }  
}

# 2. Employee Controller with Action Methods

In the Controllers folder, create EmployeeController.cs and implement the following code:

using Microsoft.AspNetCore.Mvc;  
using FirstWebApi.Models;  
using FirstWebApi.Filters;  
  
namespace FirstWebApi.Controllers;  
  
[ApiController]  
[Route("api/[controller]")]  
[ServiceFilter(typeof(CustomAuthFilter))]  
public class EmployeeController : ControllerBase  
{  
 private static List<Employee> \_employees;  
  
 public EmployeeController()  
 {  
 if (\_employees == null || !\_employees.Any())  
 \_employees = GetStandardEmployeeList();  
 }  
  
 [HttpGet]  
 [ProducesResponseType(StatusCodes.Status200OK)]  
 [ProducesResponseType(StatusCodes.Status500InternalServerError)]  
 public ActionResult<List<Employee>> GetStandard()  
 {  
 // throw new Exception("Simulated exception");  
 return Ok(\_employees);  
 }  
  
 [HttpPost]  
 public IActionResult AddEmployee([FromBody] Employee emp)  
 {  
 \_employees.Add(emp);  
 return Ok("Employee added successfully");  
 }  
  
 [HttpPut("{id}")]  
 public IActionResult UpdateEmployee(int id, [FromBody] Employee emp)  
 {  
 var existing = \_employees.FirstOrDefault(e => e.Id == id);  
 if (existing == null) return NotFound("Employee not found");  
  
 existing.Name = emp.Name;  
 existing.Salary = emp.Salary;  
 existing.Permanent = emp.Permanent;  
 existing.Department = emp.Department;  
 existing.Skills = emp.Skills;  
 existing.DateOfBirth = emp.DateOfBirth;  
  
 return Ok("Employee updated successfully");  
 }  
  
 private List<Employee> GetStandardEmployeeList()  
 {  
 return new List<Employee>  
 {  
 new Employee  
 {  
 Id = 1,  
 Name = "Siva",  
 Salary = 60000,  
 Permanent = true,  
 Department = new Department { Id = 101, Name = "Development" },  
 Skills = new List<Skill>  
 {  
 new Skill { Id = 1, Name = "C#" },  
 new Skill { Id = 2, Name = "SQL" }  
 },  
 DateOfBirth = new DateTime(1999, 12, 31)  
 }  
 };  
 }  
}

# 3. Custom Authorization Filter

Inside a folder named Filters, add the following:

CustomAuthFilter.cs

using Microsoft.AspNetCore.Mvc;  
using Microsoft.AspNetCore.Mvc.Filters;  
  
namespace FirstWebApi.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"))  
 {  
 context.Result = new BadRequestObjectResult("Invalid request - Token present but Bearer unavailable");  
 return;  
 }  
  
 base.OnActionExecuting(context);  
 }  
}

# 4. Custom Exception Filter

CustomExceptionFilter.cs

using Microsoft.AspNetCore.Mvc;  
using Microsoft.AspNetCore.Mvc.Filters;  
  
namespace FirstWebApi.Filters;  
  
public class CustomExceptionFilter : IExceptionFilter  
{  
 public void OnException(ExceptionContext context)  
 {  
 string path = Path.Combine(Directory.GetCurrentDirectory(), "error\_log.txt");  
 File.AppendAllText(path, $"{DateTime.Now}: {context.Exception.Message}{Environment.NewLine}");  
  
 context.Result = new ObjectResult("An internal server error occurred")  
 {  
 StatusCode = StatusCodes.Status500InternalServerError  
 };  
 }  
}

# 5. Program.cs Configuration

Register filters and enable Swagger:

var builder = WebApplication.CreateBuilder(args);  
  
builder.Services.AddControllers(options =>  
{  
 options.Filters.Add<CustomExceptionFilter>();  
});  
  
builder.Services.AddScoped<CustomAuthFilter>();  
builder.Services.AddScoped<CustomExceptionFilter>();  
  
builder.Services.AddEndpointsApiExplorer();  
builder.Services.AddSwaggerGen();  
  
var app = builder.Build();  
  
app.UseHttpsRedirection();  
app.UseAuthorization();  
app.MapControllers();  
app.UseSwagger();  
app.UseSwaggerUI(c =>  
{  
 c.SwaggerEndpoint("/swagger/v1/swagger.json", "Employee API V1");  
});  
  
app.Run();

# 6. Screenshots





