# WEEK 4 API WEB (3)

Problem Statement:

**Objectives:**

* Demonstrate creation of an Action method to return list of custom class entity
  + Model class creation, Use AllowAnonymous attribute, Use HttpGet action method
* Explain the usage of FromBody attribute
  + Read the model object from request, other than the query string parameter
* Demonstrate Custom filter
  + Usage of ActionFilterAttribute, OnActionExecuting method to intercept the request, Create filter for Custom exception - Need to install Microsoft.AspNetCore.Mvc.WebApiCompatShim package

1. **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

1. **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.

1. **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

# THE Code:

Program.cs:

using EmployeeApi.Filters;

using Microsoft.AspNetCore.Builder;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Hosting;

using Microsoft.OpenApi.Models;

using WebApplication4.Filters;

var builder = WebApplication.CreateBuilder(args);

// Add controller services with exception filter globally

builder.Services.AddControllers(options =>

{

options.Filters.Add<CustomExceptionFilter>();

});

// Register custom authorization filter

builder.Services.AddScoped<CustomAuthFilter>();

// Swagger configuration

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new OpenApiInfo

{

Title = "Employee API",

Version = "v1",

Description = "API to demonstrate custom filters and model handling"

});

c.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme

{

In = ParameterLocation.Header,

Description = "Enter 'Bearer' followed by your token",

Name = "Authorization",

Type = SecuritySchemeType.ApiKey

});

c.AddSecurityRequirement(new OpenApiSecurityRequirement

{

{

new OpenApiSecurityScheme

{

Reference = new OpenApiReference

{

Type = ReferenceType.SecurityScheme,

Id = "Bearer"

}

},

Array.Empty<string>()

}

});

});

var app = builder.Build();

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI(c =>

{

c.SwaggerEndpoint("/swagger/v1/swagger.json", "Employee API V1");

c.RoutePrefix = string.Empty;

});

}

app.UseAuthorization();

app.MapControllers();

app.Run();

CustomAuthFLiter:

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

namespace WebApplication4.Filters

{

public class CustomAuthFilter : ActionFilterAttribute

{

public override void OnActionExecuting(ActionExecutingContext context)

{

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

{

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

return;

}

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

{

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

}

}

}

}

CustomExceptionFilter:

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

namespace EmployeeApi.Filters

{

public class CustomExceptionFilter : IExceptionFilter

{

public void OnException(ExceptionContext context)

{

var errorMessage = $"{DateTime.Now}: {context.Exception.Message}{Environment.NewLine}";

File.AppendAllText("errors.txt", errorMessage);

context.Result = new ObjectResult("An unexpected error occurred.")

{

StatusCode = 500

};

}

}

}

Employee.cs:

namespace EmployeeApi.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; }

}

public class Department

{

public int Id { get; set; }

public string Name { get; set; }

}

public class Skill

{

public int Id { get; set; }

public string Name { get; set; }

}

}

OUTPUT:



