**Web Api using Custom Model Class**

**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

**EmployeeController.cs:**

using Microsoft.AspNetCore.Mvc;

using MyEmployeeApi.Models;

using MyEmployeeApi.Filters;

namespace MyEmployeeApi.Controllers;

[ApiController]

[Route("api/Emp")]

[ServiceFilter(typeof(CustomAuthFilter))]

public class EmployeeController : ControllerBase

{

private static List<Employee> employees;

public EmployeeController()

{

if (employees == null)

employees = GetStandardEmployeeList();

}

[HttpGet]

[ProducesResponseType(StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status500InternalServerError)]

public ActionResult<List<Employee>> Get()

{

// Uncomment to test exception handling

throw new Exception("Simulated exception");

// return Ok(employees);

}

[HttpGet("standard")]

public ActionResult<Employee> GetStandard()

{

return Ok(employees.FirstOrDefault());

}

[HttpPost]

public IActionResult Post([FromBody] Employee emp)

{

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;

return Ok(existing);

}

private List<Employee> GetStandardEmployeeList()

{

return new List<Employee>

{

new Employee

{

Id = 1,

Name = "John",

Salary = 60000,

Permanent = true,

DateOfBirth = new DateTime(1990, 5, 1),

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

Skills = new List<Skill>

{

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

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

}

}

};

}

}

**CustomAuthFilter.cs:**

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

namespace MyEmployeeApi.Filters;

public class CustomAuthFilter : ActionFilterAttribute

{

public override void OnActionExecuting(ActionExecutingContext context)

{

var hasAuth = context.HttpContext.Request.Headers.TryGetValue("Authorization", out var value);

if (!hasAuth)

{

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

return;

}

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

{

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

}

}

}

**CustomExceptionFilter.cs:**

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

namespace MyEmployeeApi.Filters;

public class CustomExceptionFilter : IExceptionFilter

{

public void OnException(ExceptionContext context)

{

var exception = context.Exception;

File.AppendAllText("logs.txt", $"{DateTime.Now}: {exception.Message}\n");

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

{

StatusCode = 500

};

}

}

**Program.cs:**

using Microsoft.OpenApi.Models;

var builder = WebApplication.CreateBuilder(args);

// Register custom filters

builder.Services.AddScoped<CustomAuthFilter>();

// Add controllers with global exception filter

builder.Services.AddControllers(options =>

{

options.Filters.Add<CustomExceptionFilter>();

});

// Enable Swagger with Authorization support

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new OpenApiInfo

{

Title = "Swagger Demo",

Version = "v1",

Description = "TBD",

TermsOfService = new Uri("https://example.com/terms"),

Contact = new OpenApiContact

{

Name = "John Doe",

Email = "john@xyzmail.com",

Url = new Uri("https://www.example.com")

},

License = new OpenApiLicense

{

Name = "License Terms",

Url = new Uri("https://www.example.com")

}

});

// 🔒 Add Authorization (Bearer token) support

c.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme

{

Name = "Authorization",

Type = SecuritySchemeType.ApiKey,

Scheme = "Bearer",

BearerFormat = "JWT",

In = ParameterLocation.Header,

Description = "Enter 'Bearer' followed by space and token"

});

c.AddSecurityRequirement(new OpenApiSecurityRequirement

{

{

new OpenApiSecurityScheme

{

Reference = new OpenApiReference

{

Type = ReferenceType.SecurityScheme,

Id = "Bearer"

}

},

Array.Empty<string>()

}

});

});

var app = builder.Build();

// Enable middleware

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run()

**Output:**









