**Filters:**

using Microsoft.AspNetCore.Mvc.Filters;

namespace WebAPI\_Demo.Filters

{

public class MyActionFilter :Attribute, IActionFilter

{

private readonly string \_name;

public MyActionFilter(string name)

{

\_name = name;

}

public void OnActionExecuted(ActionExecutedContext context)

{

Console.WriteLine($"OnActionExecuted-{\_name}");

}

public void OnActionExecuting(ActionExecutingContext context)

{

Console.WriteLine($"OnActionExecuting-{\_name}");

}

}

}

using Microsoft.AspNetCore.Mvc.Filters;

namespace WebAPI\_Demo.Filters

{

public class MyActionFilter :Attribute, IAsyncActionFilter

{

private readonly string \_name;

public MyActionFilter(string name)

{

\_name = name;

}

public async Task OnActionExecutionAsync(ActionExecutingContext context, ActionExecutionDelegate next)

{

Console.WriteLine($"OnActionExecuted-{\_name}");

await next();

}

}

}

**WebApi Controller:**

// GET: api/Products

[HttpGet]

[MyActionFilter("GetAllProduct Action Method called")]

public async Task<ActionResult<IEnumerable<ProductModel>>> GetProduct()

{

if (\_context.Product == null)

{

return NotFound();

}

return await \_context.Product.ToListAsync();

}

Exception Handling:

Microsoft.AspNetCore.Diagnostics

**Repository Pattern:**

**Interface: (**IProductRepository**)**

using WebAPI\_Demo.Models;

namespace WebAPI\_Demo.Repository

{

public interface IProductRepository

{

Task<List<ProductModel>> GetAllProduct();

Task<ProductModel> GetProductById(int id);

Task<ProductModel> CreateProduct(ProductModel product);

Task<ProductModel> UpdateProduct(ProductModel product);

Task<ProductModel> DeleteProduct(int id);

}

}

**Class(ProductRepository):**

using WebAPI\_Demo.Models;

using Microsoft.EntityFrameworkCore;

using WebAPI\_Demo.Repository;

namespace WebAPI\_Demo.Repository

{

public class ProductRepository : IProductRepository

{

private readonly MyDbContext \_context;

public ProductRepository(MyDbContext context)

{

\_context = context;

}

public async Task<List<ProductModel>> GetAllProduct()

{

return await \_context.Product.ToListAsync();

}

public async Task<ProductModel> GetProductById(int id)

{

return await \_context.Product.FindAsync(id);

}

public async Task<ProductModel> CreateProduct(ProductModel product)

{

var data= await \_context.Product.AddAsync(product);

await \_context.SaveChangesAsync();

return data.Entity;

}

public async Task<ProductModel> UpdateProduct(ProductModel product)

{

var data = await \_context.Product.FirstOrDefaultAsync(p => p.p\_id == product.p\_id);

if (data != null)

{

data.p\_name = product.p\_name;

data.p\_price = product.p\_price;

await \_context.SaveChangesAsync();

return data;

}

else

return null;

}

public async Task<ProductModel> DeleteProduct(int id)

{

var data = await \_context.Product.FirstOrDefaultAsync(p => p.p\_id == id);

if (data != null)

{

\_context.Product.Remove(data);

await \_context.SaveChangesAsync();

}

return null;

}

}

}

**WebAPI Controller:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

using Microsoft.EntityFrameworkCore;

using WebAPI\_Demo.Models;

using WebAPI\_Demo.Filters;

using WebAPI\_Demo.Repository;

namespace WebAPI\_Demo.Controllers

{

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

[ApiController]

public class ProductsController : ControllerBase

{

private readonly IProductRepository \_product;

public ProductsController(IProductRepository product)

{

\_product = product;

}

// GET: api/Products

[HttpGet]

[MyActionFilter("GetAllProduct Action Method called")]

public async Task<ActionResult<IEnumerable<ProductModel>>> GetProduct()

{

try

{

return Ok(await \_product.GetAllProduct());

}

catch(Exception ex)

{

return StatusCode(500, ex.Message);

}

}

// GET: api/Products/5

[HttpGet("{id}")]

public async Task<ActionResult<ProductModel>> GetProductModel(int id)

{

try

{

return Ok(await \_product.GetProductById(id));

}

catch (Exception ex)

{

return StatusCode(500, ex.Message);

}

}

// PUT: api/Products/5

// To protect from overposting attacks, see https://go.microsoft.com/fwlink/?linkid=2123754

[HttpPut("{id}")]

public async Task<IActionResult> PutProductModel(int id, ProductModel productModel)

{

try

{

return Ok(await \_product.UpdateProduct(productModel));

}

catch (Exception ex)

{

return StatusCode(500, ex.Message);

}

}

// POST: api/Products

// To protect from overposting attacks, see https://go.microsoft.com/fwlink/?linkid=2123754

[HttpPost]

public async Task<ActionResult<ProductModel>> PostProductModel(ProductModel productModel)

{

try

{

return Ok(await \_product.CreateProduct(productModel));

}

catch (Exception ex)

{

return StatusCode(500, ex.Message);

}

}

// DELETE: api/Products/5

[HttpDelete("{id}")]

public async Task<IActionResult> DeleteProductModel(int id)

{

try

{

return Ok(await \_product.DeleteProduct(id));

}

catch (Exception ex)

{

return StatusCode(500, ex.Message);

}

}

//private bool ProductModelExists(int id)

//{

// return (\_context.Product?.Any(e => e.p\_id == id)).GetValueOrDefault();

//}

}

}

**Program.cs:**

builder.Services.AddScoped<IProductRepository, ProductRepository>();