1) Create MyExceptionFilter.cs file and add below code

namespace CustomExceptionNetCore

{

public class MyExceptionFilter : IExceptionFilter

{

public void OnException(ExceptionContext context)

{

HttpStatusCode status = HttpStatusCode.InternalServerError;

var message = "Server error occurred.";

var exceptionType = context.Exception.GetType();

message = context.Exception.Message;

//You can enable logging error

context.ExceptionHandled = true;

HttpResponse response = context.HttpContext.Response;

response.StatusCode = (int)status;

response.ContentType = "application/json";

context.Result = new ObjectResult(new { Message = message});

}

}

}

==================================================================

2) Register custom exception filter in startup.cs file

namespace CustomExceptionNetCore

{

public class Startup

{

public Startup(IConfiguration configuration)

{

Configuration = configuration;

}

public IConfiguration Configuration { get; }

// This method gets called by the runtime. Use this method to add services to the container.

public void ConfigureServices(IServiceCollection services)

{

services.AddControllers();

services.AddMvc(config =>

{

config.Filters.Add(typeof(MyExceptionFilter));

});

}

// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

{

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

}

app.UseRouting();

app.UseAuthorization();

app.UseEndpoints(endpoints =>

{

endpoints.MapControllers();

});

}

}

}

================================================================

3) controller.cs

namespace CustomExceptionNetCore.Controllers

{

[ApiController]

[Route("[controller]")]

public class WeatherForecastController : ControllerBase

{

private static readonly string[] Summaries = new[]

{

"Freezing", "Bracing", "Chilly", "Cool", "Mild", "Warm", "Balmy", "Hot", "Sweltering", "Scorching"

};

private readonly ILogger<WeatherForecastController> \_logger;

public WeatherForecastController(ILogger<WeatherForecastController> logger)

{

\_logger = logger;

}

//[HttpGet]

//public IEnumerable<WeatherForecast> Get()

//{

// var rng = new Random();

// return Enumerable.Range(1, 5).Select(index => new WeatherForecast

// {

// Date = DateTime.Now.AddDays(index),

// TemperatureC = rng.Next(-20, 55),

// Summary = Summaries[rng.Next(Summaries.Length)]

// })

// .ToArray();

//}

[HttpGet]

public int Get()

{

int number1 = 10, number2 = 0;

var result = number1 / number2;

return result;

}

}

}