**Inject serilog into the application**

For that I will open program.cs file.

var builder = WebApplication.CreateBuilder(args);

// Configure Serilog for logging

var logger = new LoggerConfiguration()

.WriteTo.Console()

//.MinimumLevel.Information()// Set minimum log level to Information

.MinimumLevel.Warning() // Set minimum log level to Warning

.CreateLogger();

// clear default logging providers and add Serilog

builder.Logging.ClearProviders();

// Add Serilog as the logging provider

builder.Logging.AddSerilog(logger);

Program.cs file

**Inject into controller for add logging to console**

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

[ApiController]

public class TestLoggingController : ControllerBase

{

private readonly IRegionRepository \_regionRepository;

private readonly ILogger<TestLoggingController> \_Logger;

public TestLoggingController(IRegionRepository regionRepository, ILogger<TestLoggingController> logger1)

{

\_regionRepository = regionRepository;

\_Logger = logger1;

}

[HttpGet]

[Route("Serilog")]

public async Task<IActionResult> SerilogLogging()

{

\_Logger.LogInformation($"serilog logging is working fine");

//calling repository to get all regions

var regionsDomain = await \_regionRepository.GetAllAsync();

\_Logger.LogInformation($"Finished getting all regions request total records : {regionsDomain.Count()} and records: {System.Text.Json.JsonSerializer.Serialize(regionsDomain)}");

return Ok(regionsDomain);

}

}

**Example 2 with .MinimumLevel.Warning()**

// Configure Serilog for logging

var logger = new LoggerConfiguration()

.WriteTo.Console()

//.MinimumLevel.Information()// Set minimum log level to Information

.MinimumLevel.Warning() // Set minimum log level to Warning

.CreateLogger();

// clear default logging providers and add Serilog

builder.Logging.ClearProviders();

// Add Serilog as the logging provider

builder.Logging.AddSerilog(logger);

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

[ApiController]

public class TestLoggingController : ControllerBase

{

private readonly IRegionRepository \_regionRepository;

private readonly ILogger<TestLoggingController> \_Logger;

public TestLoggingController(IRegionRepository regionRepository, ILogger<TestLoggingController> logger1)

{

\_regionRepository = regionRepository;

\_Logger = logger1;

}

[HttpGet]

[Route("Serilog")]

public async Task<IActionResult> SerilogLogging()

{

\_Logger.LogInformation($"serilog logging is working fine");

\_Logger.LogWarning($"This is a warning message from Serilog");

\_Logger.LogError($"This is an error message from Serilog");

//calling repository to get all regions

var regionsDomain = await \_regionRepository.GetAllAsync();

\_Logger.LogInformation($"Finished getting all regions request total records : {regionsDomain.Count()} and records: {System.Text.Json.JsonSerializer.Serialize(regionsDomain)}");

return Ok(regionsDomain);

}

}

**Example 3 - Log Exceptions**

// Configure Serilog for logging

var logger = new LoggerConfiguration()

.WriteTo.Console()

//.MinimumLevel.Information()// Set minimum log level to Information

.MinimumLevel.Warning() // Set minimum log level to Warning

.CreateLogger();

// clear default logging providers and add Serilog

builder.Logging.ClearProviders();

// Add Serilog as the logging provider

builder.Logging.AddSerilog(logger);

[HttpGet]

[Route("logException")]

public async Task<IActionResult> ExceptionHandling()

{

try

{

throw new Exception("This is a test exception for Serilog logging");

//calling repository to get all regions

var regionsDomain = await \_regionRepository.GetAllAsync();

return Ok(regionsDomain);

}

catch (Exception ex)

{

\_Logger.LogError(ex, ex.Message);

//\_Logger.LogError(ex, "An error occurred while processing the request in ExceptionHandling method");

//return StatusCode(500, "Internal server error");

throw; //rethrowing the exception to be handled by global exception handler

}

}

Note : instead of applying try catch on all controller we can use Globale Exception handling