HƯỚNG DẪN SỬ DỤNG SERILOG TRONG ASP.NET CORE

Tài liệu này cung cấp hướng dẫn chuyên sâu và chi tiết nhất về cách tích hợp Serilog vào một dự án ASP.NET Core, phù hợp với tiêu chuẩn áp dụng tại các dự án doanh nghiệp. Serilog là thư viện logging mạnh mẽ, hỗ trợ ghi log có cấu trúc, giúp dễ dàng phân tích, tìm lỗi và giám sát hệ thống hiệu quả.

1. Giới thiệu Serilog là gì?

Serilog là một thư viện mã nguồn mở trong .NET dùng để ghi log có cấu trúc (structured logging). Khác với log text thuần, Serilog cho phép ghi log với các thuộc tính cụ thể giúp dễ dàng truy vết và phân tích tự động. Điều này rất quan trọng trong các dự án doanh nghiệp có tích hợp hệ thống log tập trung như ELK Stack, Seq hoặc Datadog.

2. Lợi ích khi dùng Serilog trong doanh nghiệp

- Ghi log có cấu trúc, hỗ trợ JSON
- Ghi vào nhiều nơi cùng lúc (console, file, DB, cloud...)
- Tích hợp dễ dàng với ASP.NET Core lifecycle
- Quản lý log theo cấp độ (Information, Error, Warning...)
- Hỗ trợ log tập trung qua Seq hoặc Elasticsearch

3. Cài đặt các package cần thiết

Bạn cần cài các gói sau bằng NuGet:

- Serilog.AspNetCore
- Serilog.Sinks.Console
- Serilog.Sinks.File
- Serilog.Sinks.Seq
- Serilog.Settings.Configuration

4. Tích hợp vào Program.cs

CSharp

```
Log.Logger = new LoggerConfiguration()
    .ReadFrom.Configuration(builder.Configuration)
    .Enrich.FromLogContext()
    .WriteTo.Console()
    .WriteTo.File("Logs/log-.txt", rollingInterval: RollingInterval.Day)
    .CreateLogger();

builder.Host.UseSerilog();
```

5. Thêm cấu hình vào appsettings.json

Bạn nên cấu hình log qua appsettings.json để dễ thay đổi giữa các môi trường:

Json

```
"Serilog": {
  "MinimumLevel": { "Default": "Information" },
  "WriteTo": [
    { "Name": "Console" },
    { "Name": "File", "Args": { "path": "Logs/log-.txt", "rollingInterval": "Day" }}
  ],
  "Enrich": ["FromLogContext"]
}
```

6. Các cấp độ log phổ biến

Cấp độ	Ý nghĩa
Verbose	Chi tiết toàn bộ hệ thống, dùng khi cần debug rất sâu.
Debug	Thông tin phục vụ lập trình viên, không cần trong production.
Information	Thông tin chung, ví dụ: user đăng nhập thành công.
Warning	Cảnh báo bất thường nhưng chưa phải lỗi.
Error	Có lỗi xảy ra, cần kiểm tra.
Fatal	Lỗi nghiêm trọng khiến hệ thống dừng.

7. Cách ghi log trong Controller hoặc Service

Csharp

```
public class SampleController : ControllerBase
{
    private readonly ILogger<SampleController> _logger;

    public SampleController(ILogger<SampleController> logger)
    {
        _logger = logger;
    }

    [HttpGet("/test")]
    public IActionResult Test()
    {
        _logger.LogInformation("Yêu cầu nhận lúc {Time}", DateTime.UtcNow);
        return Ok();
    }
}
```

8. Best Practices khi dùng Serilog

- Luôn dùng ILogger<T> để inject log thay vì gọi Log.Logger trực tiếp
- Không log thông tin nhạy cảm (mật khẩu, token...)
- Ghi log đầu vào/ra của API thông qua middleware
- Sử dụng rolling file để tránh file log quá lớn
- Dùng filter để tắt log không cần thiết của Microsoft

9. Các hệ thống log tập trung thường dùng

- **Seq**: Trình xem log mạnh mẽ, tích hợp tốt với Serilog:
- **ELK Stack (Elasticsearch, Logstash, Kibana)**: Tùy biến mạnh mẽ, phổ biến
- **Datadog / Splunk**: Dịch vụ thương mại có dashboard phân tích chuyên sâu

10. Ví dụ về Serilog và phân tích

```
  "Serilog": {
    "Using": [
      "Serilog.Sinks.Console"
    ],
    "MinimumLevel": "Error",
    "WriteTo": [
      {
        "Name": "Console",
        "Args": {
          "theme": "Serilog.Sinks.SystemConsole.Themes.AnsiConsoleTheme::Code,
Serilog.Sinks.Console",
          "outputTemplate": "{Timestamp:yyyy-MM-dd HH:mm:ss} [{Level: u3}]
{Message:lj} <s: {SourceContext}>{NewLine}{Exception}"
        }
      }
    ],
    "Enrich": [
      "FromLogContext",
      "WithMachineName",
      "WithThreadId"
    ],
    "Destructure": [
      {
        "Name": "ToMaximumDepth",
        "Args": {
          "maximumDepth": 4
        }
      },
      {
        "Name": "ToMaximumStringLength",
        "Args": {
          "maximumCollectionCount": 100
        }
      },
      {
        "Name": "ToMaximumCollectionCount",
        "Args": {
          "maximumCollectionCount": 10
        }
      }
    ],
    "Properties": {
      "Application": "Nyakko.Services.AuthAPI"
    }
  }
```

Phân Tích Cấu Hình Serilog

I. "Using"

```
"Using": ["Serilog.Sinks.Console"]
```

- Chỉ định sử dụng Serilog.Sinks.Console để ghi log ra màn hình console.

II. "MinimumLevel"

```
"MinimumLevel": "Error"
```

- Mức độ log tối thiểu là Error. Các log dưới mức này như Information, Debug, Warning sẽ bị bỏ qua.

III. "WriteTo"

```
"WriteTo": [
  {
    "Name": "Console",
    "Args": {
      "theme": "Serilog.Sinks.SystemConsole.Themes.AnsiConsoleTheme::Code,
Serilog.Sinks.Console",
      "outputTemplate": "{Timestamp:yyyy-MM-dd HH:mm:ss} [{Level: u3}]
{Message:lj} <s: {SourceContext}>{NewLine}{Exception}"
    }
  }
]
```

- Ghi log ra console với định dạng hiển thị cụ thể và sử dụng giao diện màu sắc (AnsiConsoleTheme::Code).

- outputTemplate bao gồm timestamp, cấp độ log viết tắt, nội dung message, nguồn (SourceContext), và exception nếu có.

Sink: Ghi log ra **console**.

Args:

- theme: Giao diện của log trong console. Ở đây dùng AnsiConsoleTheme::Code, giúp làm nổi bật màu sắc trong terminal.
- outputTemplate: Template hiển thị log:
    - {Timestamp:yyyy-MM-dd HH:mm:ss}: Thời gian log.
    - [{Level: u3}]: Cấp độ log viết tắt 3 chữ cái (INF, ERR...).
    - {Message:lj}: Nội dung log.
    - <s: {SourceContext}>: Nguồn phát sinh log (ví dụ: class hay namespace).
    - {NewLine}{Exception}: Dòng mới nếu có exception sẽ in ra.

IV. "Enrich"

Mục đích: Thêm thông tin bổ sung vào mỗi log.

```
"Enrich": [
  "FromLogContext",
  "WithMachineName",
  "WithThreadId"
]
```

- FromLogContext: thêm thông tin từ context (ví dụ: requestId).

- WithMachineName: thêm tên máy vào log.

- WithThreadId: thêm ID của thread thực hiện log.

V. "Destructure"

```
"Destructure": [
  { "Name": "ToMaximumDepth", "Args": { "maximumDepth": 4 } },
  { "Name": "ToMaximumStringLength", "Args": { "maximumCollectionCount": 100 }
},
  { "Name": "ToMaximumCollectionCount", "Args": { "maximumCollectionCount": 10
} }
]
```

- ToMaximumDepth: log object tối đa 4 cấp sâu.

- ToMaximumStringLength: (khai báo sai, cần sửa).

- ToMaximumCollectionCount: log tối đa 10 phần tử cho một collection.

🛠 GỢI Ý THÊM:

```
- { "Name": "ToMaximumStringLength", "Args": {
"maximumCollectionCount": 100 } }
+ { "Name": "ToMaximumStringLength", "Args": { "maximumStringLength":
100 } }
```

VI. "Properties"

```
"Properties": {
  "Application": "Nyakko.Services.AuthAPI"
}
```

Mục đích: Gắn thêm thông tin tĩnh vào log (luôn có trong mọi log).

Application: Gắn tên ứng dụng vào log – rất hữu ích khi dùng nhiều service ghi log chung một nơi (như ELK stack, Seq…).

- Gắn tên ứng dụng (Nyakko.Services.AuthAPI) vào tất cả các log.

🔚 Tổng Kết

Thành phần	Mô tả
Using	Ghi log ra console
MinimumLevel	`Error` trở lên
WriteTo	Giao diện màu, có timestamp, level, message, nguồn
Enrich	Thêm tên máy, ID thread, context
Destructure	Tối đa depth = 4, max string = 100, max collection = 10
Properties	Dùng trong microservice `AuthAPI`

11. Tài nguyên mở rộng

- https://serilog.net

- https://github.com/serilog/serilog

- https://datalust.co/seq

- https://github.com/serilog/serilog/wiki

Cập nhật: 08/04/2025

Tác giả: Nyakkon – miko@wibu.me