.net下优秀的日志框架Serilog，你用上了吗？强烈推荐

作者：微信公众号：【架构师老卢】 11-14 7:10 1277

概述：Serilog 是一个功能丰富、可扩展且易于使用的.NET日志库。它支持多种日志记录场景，包括控制台、文件、数据库和第三方日志服务，具有强大的结构化日志记录功能，使日志数据更容易管理和分析。在本文中，我们详细介绍了 Serilog 的优秀之处和使用方法，包括基础使用、高级应用和不同的持久化方案

在 .NET 开发中，Serilog 是一款广受欢迎的日志库，它提供了强大的日志记录功能，具有丰富的特性和高度的可扩展性。Serilog 的优秀之处包括：

1. **可扩展性**: Serilog 可以轻松扩展以满足不同的日志记录需求，例如日志存储、格式化和过滤。它支持各种插件和自定义扩展，让你可以根据项目的具体要求定制日志记录功能。
2. **结构化日志**: Serilog 支持结构化日志，允许你以键值对的形式记录信息，这使得日志数据更容易分析和查询。这对于在日志中存储复杂的数据非常有用。
3. **异步日志记录**: Serilog 可以异步记录日志，这有助于提高应用程序的性能，减少因日志记录而引起的延迟。
4. **多种输出目标**: Serilog 支持多种输出目标，包括控制台、文件、数据库、第三方日志服务等。你可以将日志记录到不同的目标，以满足不同的需求。
5. **过滤器和级别控制**: Serilog 允许你使用过滤器来选择哪些日志消息应该被记录，以及记录的级别。这有助于减少日志的噪音，并仅记录关键信息。
6. **内建支持**: Serilog 支持各种 .NET 技术栈，包括 ASP.NET Core、Entity Framework、Xamarin 和其他常见的 .NET 应用程序框架。

在本文中，我们将详细介绍 Serilog 的各种优秀之处，并提供示例代码来演示其使用方法，包括高级应用和持久化方案。

Serilog 的基本使用

1. 安装 Serilog

首先，你需要在项目中安装 Serilog 包，可以使用 NuGet 包管理器或 .NET CLI 进行安装。

```
dotnet add package Serilog
dotnet add package Serilog.Sinks.Console
```

上述命令将安装 Serilog 的核心包和一个输出到控制台的插件。

2. 配置 Serilog

在应用程序中配置 Serilog，你可以在 `Program.cs` 文件中进行配置。以下是一个简单的配置示例，将日志记录到控制台。

```
using Serilog;
using Serilog.Sinks.Console;

public class Program
{
    public static void Main(string[] args)
    {
        Log.Logger = new LoggerConfiguration()
            .WriteTo.Console()
            .CreateLogger();

        // 启动应用程序
        CreateWebHostBuilder(args).Build().Run();
    }

    public static IWebHostBuilder CreateWebHostBuilder(string[] args) =>
        WebHost.CreateDefaultBuilder(args)
            .UseSerilog() // 将 Serilog 集成到 ASP.NET Core
            .UseStartup<Startup>();
}
```

3. 记录日志

现在，你可以在应用程序中记录日志。以下是一个简单的示例，在控制器中记录日志：

```
using Microsoft.AspNetCore.Mvc;
using Serilog;

public class MyController : ControllerBase
{
    public IActionResult Index()
    {
        Log.Information("访问了首页");
        return View();
    }
}
```

上述代码使用 `Log.Information` 方法记录信息级别的日志消息。

4. 结构化日志记录

Serilog 支持结构化日志记录，这允许你以键值对的形式记录信息。以下是一个示例：

```
Log.Information("用户登录 {@User}", new { Username = "john", UserId = 123 });
```

这种结构化的日志记录对于存储和查询复杂的数据非常有用。

Serilog 高级应用

1. 自定义输出目标

Serilog 允许你将日志记录到不同的输出目标，如文件、数据库或第三方日志服务。以下是一个示例，将日志记录到文件中。

首先，安装 Serilog 的文件输出插件：

```
dotnet add package Serilog.Sinks.File
```

然后，配置 Serilog 以将日志记录到文件：

```
Log.Logger = new LoggerConfiguration()
    .WriteTo.Console()
    .WriteTo.File("log.txt") // 将日志记录到文件
    .CreateLogger();
```

2. 异步日志记录

使用 Serilog 异步记录日志可以提高性能，特别是在高负载应用程序中。以下是一个示例：

```
Log.Logger = new LoggerConfiguration()
    .WriteTo.Console()
    .WriteTo.Async(a => a.File("log.txt")) // 异步将日志记录到文件
    .CreateLogger();
```

3. 过滤器和级别控制

Serilog 允许你使用过滤器来选择哪些日志消息应该被记录，以及记录的级别。以下是一个示例，只记录信息级别的日志消息：

```
Log.Logger = new LoggerConfiguration()
    .WriteTo.Console()
    .MinimumLevel.Information() // 只记录信息级别的日志
    .CreateLogger();
```

你还可以使用过滤器来更精确地控制哪些消息应该被记录。

Serilog 持久化方案

Serilog 支持各种持久化方案，允许你将日志数据存储在不同的地方，如文件、数据库或第三方日志服务。以下是一些常见的持久化方案。

1. 文件持久化

你可以使用 Serilog 的文件插件将日志记录到文件中，如前面所示。这是一个简单的持久化方案，适用于小型应用程序。

2. 数据库持久化

如果你希

望将日志数据存储在数据库中，你可以使用 Serilog 的数据库插件，如 Serilog.Sinks.MSSqlServer 或 Serilog.Sinks.PostgreSQL。

首先，安装适当的数据库插件：

```
dotnet add package Serilog.Sinks.MSSqlServer
```

然后，配置 Serilog 以将日志记录到数据库：

```
Log.Logger = new LoggerConfiguration()
    .WriteTo.Console()
    .WriteTo.MSSqlServer("connectionString", "tableName") // 将日志记录到数据库
    .CreateLogger();
```

3. Elasticsearch 持久化

如果你使用 Elasticsearch 作为日志存储后端，你可以使用 Serilog.Sinks.Elasticsearch 插件将日志记录到 Elasticsearch。

首先，安装 Elasticsearch 插件：

```
dotnet add package Serilog.Sinks.Elasticsearch
```

然后，配置 Serilog 以将日志记录到 Elasticsearch：

```
Log.Logger = new LoggerConfiguration()
    .WriteTo.Console()
    .WriteTo.Elasticsearch(new ElasticsearchSinkOptions(new Uri("http://localhost:9200"))) // 将日志记录到 Elasticsearch
    .CreateLogger();
```

这个插件将日志数据导入 Elasticsearch，你可以使用 Elasticsearch 强大的搜索和分析功能来查询日志数据。

Serilog 是一个功能丰富、可扩展且易于使用的.NET日志库。它支持多种日志记录场景，包括控制台、文件、数据库和第三方日志服务，具有强大的结构化日志记录功能，使日志数据更容易管理和分析。在本文中，我们详细介绍了 Serilog 的优秀之处和使用方法，包括基础使用、高级应用和不同的持久化方案。希望这个指南有助于你更好地理解和利用 Serilog 来提高应用程序的日志记录质量和性能。如果需要更多细节或示例代码，请随时留言。