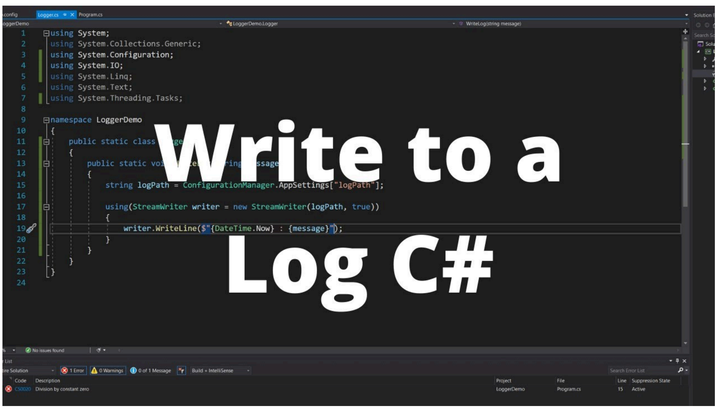
# 不要用第三方日志包了，Microsoft.Extensions.Logging功能就很强大了

作者：微信公众号：【架构师老卢】 11-14 10:41 163




**概述：**在.NET中，Microsoft.Extensions.Logging是一个广泛使用的日志库，用于记录应用程序的日志信息。它提供了丰富的功能和灵活性，使开发人员能够轻松地记录各种类型的日志，并将其输出到不同的目标，包括日志文件

在.NET中，Microsoft.Extensions.Logging是一个广泛使用的日志库，用于记录应用程序的日志信息。它提供了丰富的功能和灵活性，使开发人员能够轻松地记录各种类型的日志，并将其输出到不同的目标，包括日志文件。本文将详细介绍Microsoft.Extensions.Logging的各种基础功能以及如何按天生成日志文件。

**一、Microsoft.Extensions.Logging基础功能**

## 1. 创建Logger

首先，我们需要创建一个Logger实例，以便在应用程序中记录日志。以下是创建Logger的基本方法：

```
using Microsoft.Extensions.Logging;

var loggerFactory = LoggerFactory.Create(builder =>
{
    builder.AddConsole(); // 输出日志到控制台
});

var logger = loggerFactory.CreateLogger<Program>();
```

在上面的示例中，我们首先创建了一个LoggerFactory实例，然后通过它创建了一个Logger。我们还使用`AddConsole`方法将日志输出到控制台。

## 2. 记录日志消息

一旦有Logger实例，我们可以使用它来记录不同级别的日志消息。Microsoft.Extensions.Logging定义了以下日志级别（按严重性递增排列）：

- Trace
- Debug
- Information
- Warning
- Error
- Critical

下面是如何使用Logger记录不同级别的日志消息的示例：

```
logger.LogInformation("这是一条信息日志");
logger.LogWarning("这是一条警告日志");
logger.LogError("这是一条错误日志");
logger.LogCritical("这是一条严重错误日志");
```

## 3. 提供上下文信息

通常，我们需要将一些上下文信息记录到日志中，以帮助排查问题。可以使用`LogInformation`等方法的重载版本，传递额外的参数，如下：

```
var userId = 123;
logger.LogInformation("用户 {UserId} 登录成功", userId);
```

在上述示例中，我们将`userId`作为参数传递给日志消息，以便将其包含在日志中。

## 4. 使用日志范围

有时，我们希望在一段代码块中记录一组相关日志消息，并希望这些消息具有相同的上下文信息。这时可以使用`LogScope`，如下所示：

```
using (logger.BeginScope("交易处理 - 订单 {OrderId}", orderId))
{
    logger.LogInformation("订单处理中...");
    // 执行一些订单处理逻辑
    logger.LogInformation("订单处理完成");
}
```

在上述示例中，我们使用`BeginScope`方法创建了一个日志范围，将其包含在指定的上下文信息中。在范围内的所有日志消息都会自动包含这些上下文信息。

## 5. 配置日志级别

我们可以在应用程序中配置所需的最低日志级别，以决定哪些日志消息将被记录。通常，这是在应用程序的配置文件中完成的。以下是一个示例：

```
var loggerFactory = LoggerFactory.Create(builder =>
{
    builder.AddConsole();
    builder.SetMinimumLevel(LogLevel.Information); // 设置最低日志级别
});
```

在上述示例中，我们使用`SetMinimumLevel`方法来配置最低日志级别。只有达到或高于指定级别的日志消息才会被记录。

**二、按天生成日志文件**

现在，让我们来看看如何按天生成日志文件。通常，我们会希望将日志消息记录到文件中，并按日期对日志文件进行归档。

## 1. 安装相关包

首先，我们需要安装一些必要的包，以便实现按天生成日志文件。使用NuGet包管理器或.NET CLI可以轻松完成这一步骤：

```
dotnet add package Microsoft.Extensions.Logging.File
```

## 2. 配置文件日志提供程序

接下来，我们需要配置文件日志提供程序，以便将日志消息记录到文件中。以下是如何配置文件提供程序的示例：

```
var loggerFactory = LoggerFactory.Create(builder =>
{
    builder.AddFile("logs/myapp-{Date}.txt");
    builder.SetMinimumLevel(LogLevel.Information);
});
```

在上述示例中，我们使用`AddFile`方法配置文件日志提供程序，并指定日志文件的名称模式。`{Date}`将根据日志消息的日期自动替换为实际日期。

## 3. 示例代码

以下是一个完整的示例代码，演示了如何使用Microsoft.Extensions.Logging按天生成日志文件：

```
using System;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Logging.File;

class Program
{
    static void Main()
    {
        var loggerFactory = LoggerFactory.Create(builder =>
        {
            builder.AddFile("logs/myapp-{Date}.txt");
            builder.SetMinimumLevel(LogLevel.Information);
        });

        var logger = loggerFactory.CreateLogger<Program>();

        for (int i = 0; i < 10; i++)
        {
            logger.LogInformation("这是一条信息日志 - {LogCount}", i);
        }
    }
}
```

上述示例中，我们配置了文件日志提供程序，日志文件的名称模式包含了`{Date}`，并循环记录了10条日志消息。每天，日志文件将被归档到不同的文件中，以便跟踪和检查以前的日志。

Microsoft.Extensions.Logging是.NET中强大的日志库，可以用于记录各种类型的日志消息。通过配置文件日志提供程序，我们可以按天生成日志文件，以便更好地管理和分析日志。