参考文章：https://www.cnblogs.com/artech/p/inside-asp-net-core-1.html

嗯，不用说，也很常见

**简单示例**

public static void Main(string[] args)

{

    //注册EncodingProvider实现对中文编码的支持

    Encoding.RegisterProvider(CodePagesEncodingProvider.Instance);

    // 日志过滤器

Func<string, LogLevel, bool> filter = (category, level) => level >= LogLevel.Warning;

    // 新建日志工厂

    ILoggerFactory loggerFactory = new LoggerFactory();

    // 添加日志提供者

    loggerFactory.AddProvider(new ConsoleLoggerProvider(filter, false));

loggerFactory.AddProvider(new DebugLoggerProvider(filter));

    // 生成日志

ILogger logger = loggerFactory.CreateLogger(nameof(Program));

    int eventId = 3721;

    // 写入消息日志

    logger.LogInformation(eventId, "升级到最新.NET Core版本({version})", "1.0.0");

    // 写入警告日志

    logger.LogWarning(eventId, "并发量接近上限({maximum}) ", 200);

    // 写入错误日志

    logger.LogError(eventId, "数据库连接失败(数据库：{Database}，用户名：{User})", "TestDb", "sa");

}

**ILogger日志**

ILogger是 .net core 的日志，我们可以使用该对象记录各种日志

**ILogger的生成**

ILogger由ILoggerFactory生成

    public interface ILoggerFactory : IDisposable

    {

        void AddProvider(ILoggerProvider provider);

        ILogger CreateLogger(string categoryName);

    }

**ILoggerFactory如何生成ILogger**

我们可以向ILoggerFactory注册多个ILoggerProvider，ILoggerFactory利用ILoggerProvider提供的ILogger合并为一个ILogger

    public interface ILoggerProvider : IDisposable

    {

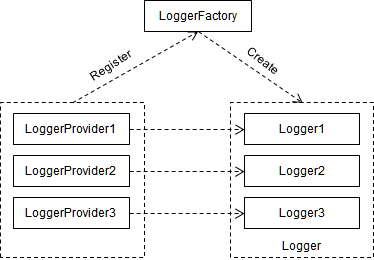
        ILogger CreateLogger(string categoryName);

    }

要想自定义日志，我们需要实现ILoggerProvider和对应的ILogger

**设计图**

来源：https://www.cnblogs.com/artech/p/inside-asp-net-core-1.html



**log4net日志**

log4net日志是比较常用的第三方日志，.net core中整合了log4net

包：Microsoft.Extensions.Logging.Log4Net.AspNetCore

示例

项目结构：



Program.cs：

static void Main (string[] args) {

    // 新建日志工厂

    ILoggerFactory loggerFactory = new LoggerFactory ();

    // 添加日志提供者，使用扩展方法 AddLog4Net 或 AddProvider 添加日志提供者

    loggerFactory.AddLog4Net ("log4net.xml");

// loggerFactory.AddProvider(new Log4NetProvider("log4net.xml"));

    // 生成 Logger

ILogger logger = loggerFactory.CreateLogger (nameof (Program));

    // 写入日志消息

logger.LogInformation ("这是一条log4net日志消息");

    Console.WriteLine ("Hello World!");

}

在log4net可以提供log4net的日志配置文件，这里我们提供了log4net.xml配置文件

Log4net.xml（记得设置文件始终复制）：

<?xml version="1.0" encoding="utf-8"?>

<log4net>

    <appender name="RollingFileAppender" type="log4net.Appender.RollingFileAppender">

        <file value="App\_Data/Logs/Logs.txt" />

        <appendToFile value="true" />

        <rollingStyle value="Size" />

        <maxSizeRollBackups value="10" />

        <maximumFileSize value="10000KB" />

        <staticLogFileName value="true" />

        <!-- 日志记录格式 -->

        <layout type="log4net.Layout.PatternLayout">

            <conversionPattern value=" 【时间】:%d%n 【级别】:%p%n 【类名】:%c%n 【线程ID】: %thread %n 【日志内容】:%m%n 【日记详细】：%exception %n---------------------------------------------------------------------------------------------------------------%n" />

        </layout>

        <!-- 级别过滤 -->

        <filter type="log4net.Filter.LevelRangeFilter">

            <levelMin value="INFO" />

            <levelMax value="ERROR" />

        </filter>

        <!-- 字符串匹配过滤 -->

        <!-- <filter type = "log4net.Filter.StringMatchFilter" >

      <stringToMatch value = "IELog" />

    </filter>

    <filter type = "log4net.Filter.DenyAllFilter" /> -->

    </appender>

    <!-- 处理DEBUG级别以上日志 -->

    <root>

        <appender-ref ref="RollingFileAppender" />

        <level value="DEBUG" />

    </root>

    <logger name="NHibernate">

        <level value="WARN" />

    </logger>

</log4net>