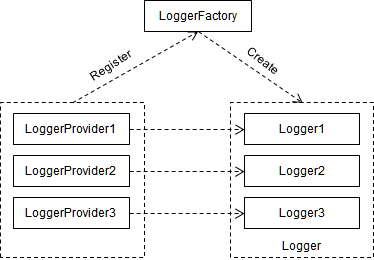
Logger、LoggerFactory和LoggerProvider是日志的核心对象，器三者关系如下：



我们向日志工厂（LoggerFactory）注册日志提供者（LoggerProvider），使用日志工厂生成日志（Logger），这个日志实际上是日志集合的封装，当我们向Logger写入日志是，他会调用集合的每一项去写入日志

日志核心三对象定义在包“Microsoft.Extensions.Logging”中

控制台日志提供者定义在包“Microsoft.Extensions.Logging.Console”中

Debug日志提供者定义在包“Microsoft.Extensions.Logging.Debug”中

Microsoft.Extensions.Logging中提供了一个依赖注入容器的扩展，用于快速将日志工厂注册到依赖注入容器中

new ServiceCollection()

    .AddLogging()

    .BuildServiceProvider()

    .GetService<ILoggerFactory>()

**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

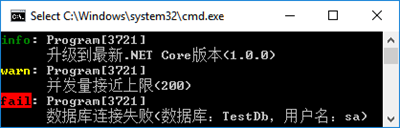
    logger.LogWarning(eventId, "并发量接近上限({maximum}) ", 200);

    // 写入错误日志

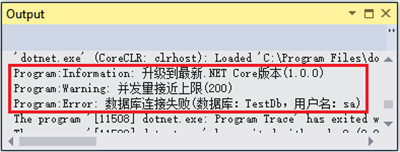
    logger.LogError(eventId, "数据库连接失败(数据库：{Database}，用户名：{User})", "TestDb", "sa");

}

以是示例会向控制台输出消息，因为我们注册了控制台日志提供者

[](http://images2015.cnblogs.com/blog/19327/201608/19327-20160817083321312-672861705.png)

以是示例会向Debug窗口输出消息，因为我们注册了Debug日志提供者

[](http://images2015.cnblogs.com/blog/19327/201608/19327-20160817083322390-129361540.png)

**日志等级**

日志等级用于日志提供者，一般用于过滤消息

public enum LogLevel

{

    Trace           = 0,

    Debug           = 1,

    Information     = 2,

    Warning         = 3,

    Error           = 4,

    Critical        = 5,

    None            = 6

}