

# 编程快客

永久域名 <http://code.javaeye.com>**jiasongmao**

浏览: 46534 次

性别:

来自: 重庆

[详细资料](#)[留言簿](#)

搜索本博客

最近访客  
[客](#)[>>更多访客](#)[yangtsefisher](#)[chenm\\_bj](#)[filter\\_tip](#)[zjun101](#)

博客分类

2009-11-06

[.NET使用NLog记录日志信息 \(二\)](#)| [ling to sql 操作存储过程、函数总结 \(二\)](#)

## [.NET使用NLog记录日志信息 \(一\)](#)

应用程序跟踪介绍

很久以前，在那个没有调试器，软件也大都基于控制台的年代里，开发者习惯于使用`printf()`语句输出跟踪调试信息。而现在，世界发生了翻天覆地般的变化——`printf()`被`Console.WriteLine()`代替了……

我们都曾经书写过类似如下的代码：

```
static void Main()
{
    Console.WriteLine("SuperApp started.");
    DoSomething();
    Console.WriteLine("SuperApp finished.");
}
```

上面这段代码中的`Console.WriteLine()`就是所谓的“跟踪”语句，因为这个语句除了输出当前程序的执行状态之外，并没有什么其他用处。由`Console.WriteLine()`的输出也叫做应用程序跟踪。在上面的例子中，这两条跟踪语句就用来告诉我们`DoSomething()`方法是否执行完毕。

在开发以及测试完成之后，我们可能会想要删除这些跟踪代码，以提高程序执行效率（因为跟踪调试语句执行效率很低）。通常我们将这些跟踪语句注释掉，以便今后需要的时候可以方便地再次启用。可是不幸的是，将跟踪语句注释掉之后，我们还需要重新编译一遍程序。

或许有一天，在经历过第N次将无数的调试语句注释和取消注释之后，你会依稀感觉到这样并不是一个好的解决方案，并期待着这样的功能：

1. 能够通过简单的方法控制显示哪些等级的跟踪信息（例如只显示警告和错误级别的跟踪信息，或是显示所有级别的跟踪信息等）。
2. 将控制跟踪信息的显示与否的逻辑与应用程序的视线代码分开，跟踪信息的显示与否并不需要重新编译应用程序。
3. 将跟踪信息写至文件、系统日志、消息队列……
4. 能够将一些极为重要的信息通过Email发送给指定收信人，或是存放在数据库中。
5. 更多你能想到的……

- [全部博客 \(200\)](#)
- [JAVA \(34\)](#)
- [.NET \(51\)](#)
- [JAVASCRIPT \(20\)](#)
- [Hibernate \(1\)](#)
- [CSS+DIV \(5\)](#)
- [编程工具 \(2\)](#)
- [JSF \(4\)](#)
- [数据库 \(1\)](#)
- [struts \(12\)](#)
- [ibatis \(2\)](#)
- [junit \(4\)](#)
- [CentOS \(35\)](#)
- [WPF \(20\)](#)
- [设计模式 \(3\)](#)
- [silverlight \(1\)](#)
- [自己的示例 \(2\)](#)

我的相册



bb1

[共 26 张](#)

我的留言簿 [>>更多留言](#)

- 你真狠,连code你都注册!!! 你个奸商
- by [jaikuai](#)

有些朋友可能觉得，在图形化调试器大行其道的当今软件开发环境中，这种书写日志的跟踪调试方式似乎用处非常有限。不过，当你的程序每一秒都被成千上万人同时访问，哪怕是停机一分钟都不能接受的时候，你就会知道这些调试信息对定位Bug来说意味着什么了（即所谓的“Live Site Debugging”）。

**NLog**是什么？

NLog (<http://www.nlog-project.org>) 是一个基于.NET平台编写的类库，我们可以使用NLog在应用程序中添加极为完善的跟踪调试代码。NLog完全实现了我们上面的期望目标，并且还远远不止这些.....

NLog允许我们自定义从跟踪消息的来源（**source**）到记录跟踪信息的目标（**target**）的规则（**rules**）。记录跟踪信息的目标（**target**）可以为如下几种形式：

1. 文件
2. 文本控制台
3. Email
4. 数据库
5. 网络中的其它计算机（通过TCP或UDP）
6. 基于MSMQ的消息队列
7. Windows系统日志
8. 其他形式，请参考<http://www.nlog-project.org/targets.html>

除此之外，每一条跟踪消息都可以自动带有上下文信息（**contextual information**），并将其发送给记录跟踪信息的目标。这些上下文信息可以包含如下内容：

1. 当前的日期和时间（多种格式）
2. 记录等级
3. 来源名称
4. 输出跟踪消息的方法的堆栈信息
5. 环境变量的值
6. 异常的详细信息
7. 计算机、进程和线程名称
8. 其他，请参考：<http://www.nlog-project.org/layoutrenderers.html>

每条跟踪信息都包含一个记录等级（**log level**）信息，用来描述该条信息的重要性。NLog支持如下几种记录等级：

1. **Trace** - 最常见的记录信息，一般用于普通输出
2. **Debug** - 同样是记录信息，不过出现的频率要比Trace少一些，一般用来调试程序
3. **Info** - 信息类型的消息
4. **Warn** - 警告信息，一般用于比较重要的场合
5. **Error** - 错误信息
6. **Fatal** - 致命异常信息。一般来讲，发生致命异常之后程序将无法继续执行。

NLog是一个免费的、基于[BSD license](#)发布的开源类库。即使将其应用于商业使用中，也基本上不会有任何的限制。NLog的二进制可执行文件以及原文件均可在<http://www.nlog-project.org/download.html>页面中下载。我们同时还为NLog提供了图形界面的安装程序，您可以选择NLog的安装路径，并可在安装过程中添加如下内容到Visual Studio集成开发环境中（同样支持Express版本）：

1. 配置文件模板

- JSF中如何使用EXT树 这个不全啊！能补充全吗？

-- by [sdml007](#)

- 问个问题 怎么刷新modelpanel里datatable的值？

-- by [zh\\_666](#)

其他分类

- [我的收藏](#) (1)
- [我的论坛主题贴](#) (200)
- [我的所有论坛贴](#) (4)
- [我的精华良好贴](#) (0)

最近加入圈子

存档

- [2010-02](#) (2)
- [2010-01](#) (11)
- [2009-12](#) (9)
- [更多存档...](#)

最新评论

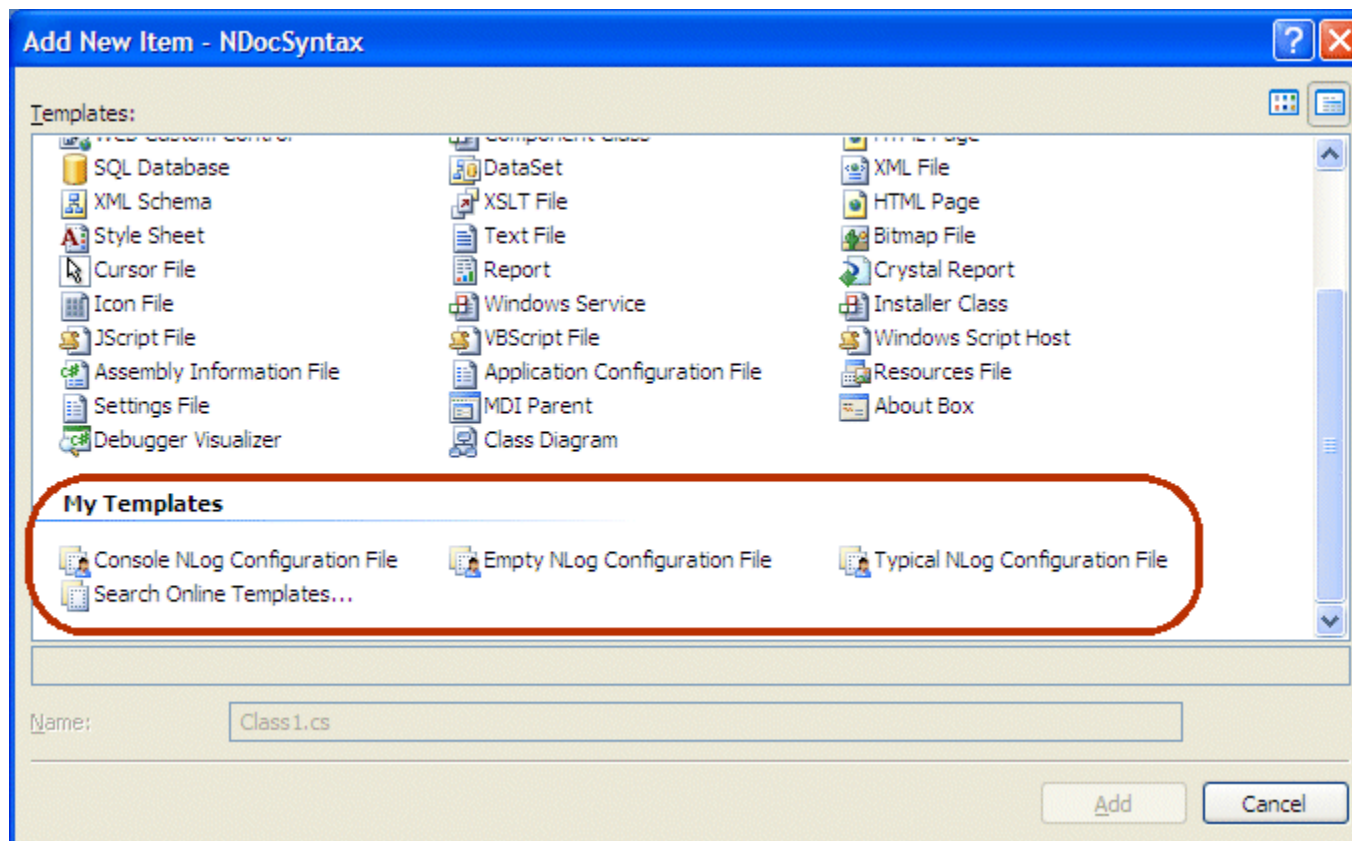
- [数据库建模工具: PowerDes ...](#)  
呵呵，谢谢前辈的文章，我是一个大三JAVA学生，UML还没入门，刚好现在想为数据库 ...  
-- by [liangrockman](#)
- [RichFace标签学习笔记](#)  
well done  
-- by [成溪先生](#)
- [struts2中使用jquery进行...](#)

2. NLog配置文件智能感知支持
3. 代码片断 (code snippet)
4. 集成至“Add Reference...”对话框

我们的第一个**NLog**应用程序

接下来让我们用**Visual Studio 2005**创建第一个使用**NLog**的应用程序。该示例程序将从把日志输出到控制台开始，并不断添加新的功能，以演示在**NLog**中对日志进行配置的方法。

首先在**Visual Studio 2005**中创建一个新项目（本示例程序将使用**C#**演示），然后在“Add New Item...”对话框中为该程序添加一个**NLog**配置文件。这里我们选择“Empty NLog Configuration File”，并命名为“NLog.config”：



注意到**NLog.dll**的引用被自动添加到了我们的项目中。刚刚添加的**NLog.config**文件的内容如下，这也正是我们在本示例程序中将要配置的：

```
<nlog xmlns="http://www.nlog-project.org/schemas/NLog.xsd"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <targets>
```

我觉得这种方法不好.很怪异,我们在action里定义的对象,不是专门给struts ...

-- by [猴猴儿的翅膀](#)

■ [struts2中的文件上传和下 ...](#)

下载文件为什么不动态指定?  
而是硬编码到配置文件中?

-- by [asdf93945](#)

■ [在CentOS下安装SVN](#)

感谢~~~~

-- by [madbird](#)

评论排行榜

■ [在CentOS下安装SVN](#)

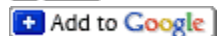
■ [数据库建模工](#)

具: [PowerDesigner与Rose详](#)  
[解 ...](#)

■ [NLog处理异常使用说明书](#)

■ [WPF中的依赖属性](#)

■ [centos下安装java.jdk](#)



[\[什么是RSS?\]](#)

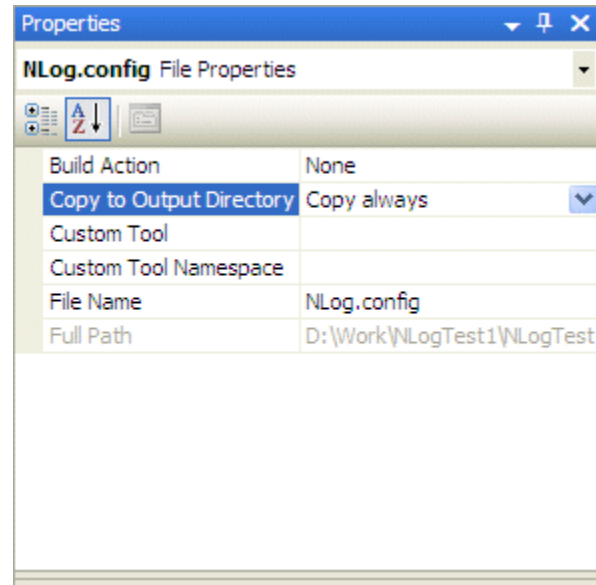
```
</targets>
```

```
<rules>
```

```
</rules>
```

```
</nlog>
```

接下来需要一个额外的步骤: 设置该配置文件的“Copy To Output Directory”选项为“Copy always”。这样该配置文件将自动被部署到\*.exe所在的目录下。这样以后, 无需任何设置, NLog即可自动搜寻到并加载该配置文件。



接下来让我们配置一下日志的输出, 在<targets />节中, 添加一个新条目让日志输出到控制台中, 并添加必要的输出布局 (layout) :

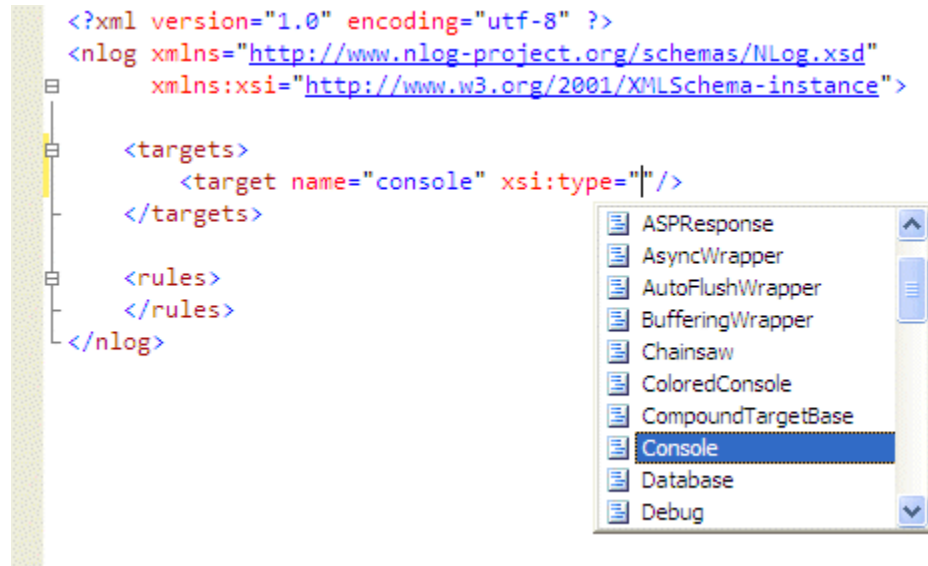
```
<targets>
```

```
<target name="console" xsi:type="Console"
```

```
layout="${longdate} | ${level} | ${message}" />
```

```
</targets>
```

在输入上述代码时, 可以看到Visual Studio的智能感知功能发挥了作用: 输入xsi:之后, 将得到当前可用的输出目标列表:



然后，我们要在<rules />节中添加必要的规则，将所有记录等级等于或高于**Debug**的信息输出至控制台。这段XML配置文件字解释能力足够强，无需多言：

```
<rules>

  <logger name="*" minlevel="Debug" writeTo="console" />

</rules>
```

若希望生成并输出诊断信息，我们还需要添加一个Logger对象。Logger对象的方法名和记录等级的名称一样（Debug()、Info()、Fatal().....）。Logger对象是通过LogManager对象创建的。建议Logger对象的名称和程序的类名保持一致。调

用LogManager的GetCurrentClassLogger()方法即可自动为当前类创建一个Logger对象。

接下来开始修改Visual Studio自动生成的这个C#文件。首先在文件头部添加“using NLog”语句，引入NLog程序集的命名空间。然后添加创建Logger对象的代码，注意在这里我们可以使用随NLog的安装而添加的Visual Studio的代码片断：输入“nlogger”，然后按两次Tab键即可。

```
using System;

using System.Collections.Generic;

using System.Text;

using NLog;

namespace NLogExample

{

    class Program
```

```

{

    private static Logger logger = LogManager.GetCurrentClassLogger();


    static void Main(string[] args)

    {

        logger.Debug("Hello World!");

    }

}
}

```

运行该程序，将看到一条日志信息输出到了控制台，该信息包含当前的时间，记录等级，以及“Hello World”消息。

最后，让我们总结一下实现该功能的步骤：

1. 用[LogManager.GetCurrentClassLogger\(\)](#)创建了一个[Logger](#)对象。该Logger对象代表与当前类相关联的日志消息的来源。
2. 通过调用Logger对象的Debug()方法，发出一条Debug记录等级的诊断信息。
3. 因为记录等级和消息来源符合配置文件中的<rules />声明，所以该消息将以指定的布局（layout）格式化后输出到控制台中。

稍微复杂一些的场景

接下来让我们将这些日志信息，包括其中的一些上下文信息（例如堆栈信息等），输出到文件和命令行两个地方。要实现这个需求，我们只要修改NLog的配置文件，并添加一个类型为“File”的目标即可。

```

<nlog xmlns="http://www.nlog-project.org/schemas/NLog.xsd"

    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

    <targets>

        <target name="console" xsi:type="ColoredConsole"

            layout="${date:format=HH\:mm\:ss} | ${level} | ${stacktrace} | ${message}" />

        <target name="file" xsi:type="File" fileName="${basedir}/file.txt"

            layout="${stacktrace} ${message}" />

    </targets>

    <rules>

        <logger name="*" minlevel="Trace" writeTo="console,file" />

    </rules>

```

</nlog>

接下来这段C#代码生成了更多的日志信息，还使用了NLog提供的一些其他方法用来输出堆栈信息。

```
static void C()
{
    logger.Info("Info CCC");
}

static void B()
{
    logger.Trace("Trace BBB");
    logger.Debug("Debug BBB");
    logger.Info("Info BBB");
    C();
    logger.Warn("Warn BBB");
    logger.Error("Error BBB");
    logger.Fatal("Fatal BBB");
}

static void A()
{
    logger.Trace("Trace AAA");
    logger.Debug("Debug AAA");
    logger.Info("Info AAA");
    B();
    logger.Warn("Warn AAA");
    logger.Error("Error AAA");
    logger.Fatal("Fatal AAA");
}
```

```

static void Main(string[] args)
{
    logger.Trace("This is a Trace message");

    logger.Debug("This is a Debug message");

    logger.Info("This is an Info message");

    A();

    logger.Warn("This is a Warn message");

    logger.Error("This is an Error message");

    logger.Fatal("This is a Fatal error message");
}

```

运行该程序，如下日志信息将被写入应用程序所在目录中的“file.txt”文件中。

```

Program.Main This is a Trace message

Program.Main This is a Debug message

Program.Main This is an Info message

Program.Main => Program.A Trace AAA

Program.Main => Program.A Debug AAA

Program.Main => Program.A Info AAA

Program.Main => Program.A => Program.B Trace BBB

Program.Main => Program.A => Program.B Debug BBB

Program.Main => Program.A => Program.B Info BBB

Program.A => Program.B => Program.C Info CCC

Program.Main => Program.A => Program.B Warn BBB

Program.Main => Program.A => Program.B Error BBB

Program.Main => Program.A => Program.B Fatal BBB

Program.Main => Program.A Warn AAA

Program.Main => Program.A Error AAA

```



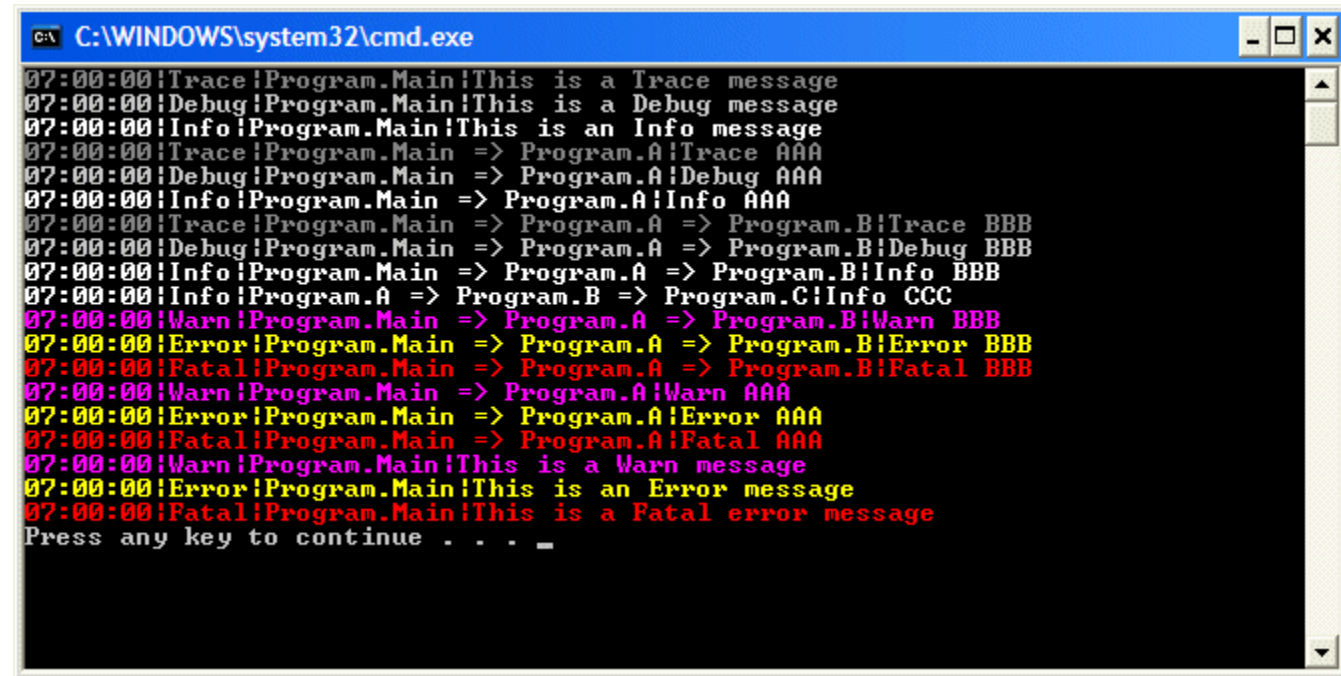
```
Program.Main => Program.A Fatal AAA

Program.Main This is a Warn message

Program.Main This is an Error message

Program.Main This is a Fatal error message
```

同时，控制台中也输出了如下漂亮的日志信息。

A screenshot of a Windows command prompt window titled "C:\WINDOWS\system32\cmd.exe". The window displays a series of log messages from NLog, each prefixed with a timestamp "07:00:00". The messages are color-coded: Trace (grey), Debug (blue), Info (green), Warn (magenta), Error (red), and Fatal (dark red). The messages show a sequence of calls between Program.Main, Program.A, Program.B, and Program.C, with various log levels. The output ends with "Press any key to continue . . . \_".

```
C:\WINDOWS\system32\cmd.exe
07:00:00!Trace!Program.Main!This is a Trace message
07:00:00!Debug!Program.Main!This is a Debug message
07:00:00!Info!Program.Main!This is an Info message
07:00:00!Trace!Program.Main => Program.A!Trace AAA
07:00:00!Debug!Program.Main => Program.A!Debug AAA
07:00:00!Info!Program.Main => Program.A!Info AAA
07:00:00!Trace!Program.Main => Program.A => Program.B!Trace BBB
07:00:00!Debug!Program.Main => Program.A => Program.B!Debug BBB
07:00:00!Info!Program.Main => Program.A => Program.B!Info BBB
07:00:00!Info!Program.A => Program.B => Program.C!Info CCC
07:00:00!Warn!Program.Main => Program.A => Program.B!Warn BBB
07:00:00!Error!Program.Main => Program.A => Program.B!Error BBB
07:00:00!Fatal!Program.Main => Program.A => Program.B!Fatal BBB
07:00:00!Warn!Program.Main => Program.A!Warn AAA
07:00:00!Error!Program.Main => Program.A!Error AAA
07:00:00!Fatal!Program.Main => Program.A!Fatal AAA
07:00:00!Warn!Program.Main!This is a Warn message
07:00:00!Error!Program.Main!This is an Error message
07:00:00!Fatal!Program.Main!This is a Fatal error message
Press any key to continue . . . _
```

再来修改一下配置文件——开发中一个很常见的需求就是让不同记录等级的日志输出到不同的目标中。例如，让记录等级等于或高于**Info**的信息输出至控制台，同时将任意记录等级的信息都存放在文件中保存。在NLog中，实现这个需求只要修改配置文件的<rules />节即可，无需修改应用程序。

```
<rules>

  <logger name="*" minlevel="Info" writeTo="console" />

  <logger name="*" minlevel="Trace" writeTo="file" />

</rules>
```

再次运行程序，可以看到**Trace**和**Debug**等级的信息只出现在了文件中，而并不显示在控制台中。

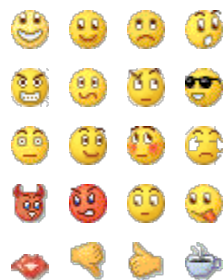
转载地址：<http://www.cnblogs.com/dflying/archive/2006/12/05/583071.html>

15:51 | 浏览 (84) | 评论 (0) | 分类: [.NET](#) | [相关推荐](#)

评论

发表评论

表情图标



字体颜色:   字体大小:   对齐:

提示: 选择您需要装饰的文字, 按上列按钮即可添加上相应的标签

---

声明：**JavaEye**文章版权属于作者，受法律保护。没有作者书面许可不得转载。若作者同意转载，必须以超链接形式标明文章原始出处和作者。

© 2003-2009 JavaEye.com. All rights reserved. 上海炯耐计算机软件有限公司 [ 沪ICP备05023328号 ]