



FREE eBook

LEARNING nlog

Free unaffiliated eBook created from
Stack Overflow contributors.

#nlog

Table of Contents

About	1
Chapter 1: Getting started with nlog	2
Remarks.....	2
Versions.....	2
Examples.....	2
Writing first logs.....	2
Chapter 2: Extending NLog	5
Examples.....	5
Create custom Target.....	5
Create custom Layout Renderer.....	6
Chapter 3: Troubleshooting NLog	7
Examples.....	7
Enable the internal logger in nlog.config.....	7
Enable the internal logger programmatically.....	7
Chapter 4: Write logs to files	8
Examples.....	8
Write to single file.....	8
Write one log file per day.....	8
Credits	9

About

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: [nlog](#)

It is an unofficial and free nlog ebook created for educational purposes. All the content is extracted from [Stack Overflow Documentation](#), which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official nlog.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapter 1: Getting started with nlog

Remarks

NLog is a free logging library for .NET. It is licensed under BSD-3, meaning it can be used in commercial applications.

NLog's source can be found at [GitHub](#)

Versions

Major versions:

version	release notes	release date
4	release notes	2015-06-09
3	release notes	2014-06-02
2	release notes	2011-07-18
1	release notes	2011-01-07

Examples

Writing first logs

Install the [NLog.Config](#) package. (NuGet 3 users should install [NLog](#) and add the config manually.) Write to logger:

```
using NLog;

public class MyClass
{
    private static Logger logger = LogManager.GetCurrentClassLogger();

    public void MyMethod1()
    {
        logger.Trace("Sample trace message");
        logger.Debug("Sample debug message");
        logger.Info("Sample informational message");
        logger.Warn("Sample warning message");
        logger.Error("Sample error message");
        logger.Fatal("Sample fatal error message");

        // alternatively you can call the Log() method
        // and pass log level as the parameter.
        logger.Log(LogLevel.Info, "Sample informational message");
    }
}
```

```

try
{
    SendMail();
}
catch(Exception ex)
{
    //example writing exception
    logger.Error(ex, "Error when sending mail");
}
}
}

```

This example config should be added with the NLog.Config package. Otherwise add the following to the root of the application - or next to the .dll as "nlog.config"

```

<?xml version="1.0" encoding="utf-8" ?>
<nlog xmlns="http://www.nlog-project.org/schemas/NLog.xsd"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.nlog-project.org/schemas/NLog.xsd NLog.xsd"
      autoReload="true"
      throwExceptions="false"
      internalLogLevel="Off" internalLogFile="c:\temp\nlog-internal.log">

  <!-- optional, add some variables
  https://github.com/nlog/NLog/wiki/Configuration-file#variables
  -->
  <variable name="myvar" value="myvalue"/>

  <!--
  See https://github.com/nlog/nlog/wiki/Configuration-file
  for information on customizing logging rules and outputs.
  -->
  <targets>

    <!--
    add your targets here
    See https://github.com/nlog/NLog/wiki/Targets for possible targets.
    See https://github.com/nlog/NLog/wiki/Layout-Renderers for the possible layout renderers.
    -->

    <!-- Write events to a file with the date in the filename. -->
    <target xsi:type="File" name="f" fileName="${basedir}/logs/${shortdate}.log"
           layout="${longdate} ${uppercase:${level}} ${message} ${exception}" />

  </targets>

  <rules>
    <!-- add your logging rules here -->

    <!--
    Write all events with minimal level of Debug (So Debug, Info, Warn, Error and Fatal, but
    not Trace) to "f" -->
    <logger name="*" minlevel="Debug" writeTo="f" />

  </rules>
</nlog>

```

The config writes the log messages to a file.

Read Getting started with nlog online: <https://riptutorial.com/nlog/topic/7133/getting-started-with-nlog>

Chapter 2: Extending NLog

Examples

Create custom Target

```
using NLog;
using NLog.Config;
using NLog.Targets;

namespace MyNamespace
{
    [Target("MyFirst")]
    public sealed class MyFirstTarget: TargetWithLayout //or inherit from Target
    {
        public MyFirstTarget()
        {
            //set defaults
            this.Host = "localhost";
        }

        [RequiredParameter]
        public string Host { get; set; }

        protected override void Write(LogEventInfo logEvent)
        {
            string logMessage = this.Layout.Render(logEvent);

            //TODO write to target
        }
    }
}
```

Register it under the name `MyFirst` - as soon as possible - e.g. in `main()`, `application_start()`.

```
ConfigurationItemFactory
    .Default
    .Targets
    .RegisterDefinition("MyFirst", typeof(MyNamespace.MyFirstTarget));
```

Usage:

```
<?xml version="1.0" ?>
<nlog xmlns="http://www.nlog-project.org/schemas/NLog.xsd"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

    <targets>
        <target name="target1" type="MyFirst" Host="somehost.com" />
    </targets>

    <rules>
        <logger name="*" minlevel="Debug" writeTo="target1" />
    </rules>
```

```
</nlog>
```

Create custom Layout Renderer

```
[LayoutRenderer("hello-world")]
public class HelloWorldLayoutRenderer : LayoutRenderer
{
    /// <summary>
    /// I'm option and not required or default
    /// </summary>
    public string Config1 { get; set; }

    /// <summary>
    /// I'm required option. And error will be thrown if not set.
    /// </summary>
    [RequiredParameter]
    public string Config2 { get; set; }

    /// <summary>
    /// I'm the default parameter.
    /// The first parameter value (without name) will be set to this one
    /// You can set me as required also.
    /// </summary>
    [DefaultParameter]
    public bool Caps {get;set;}

    protected override void Append(StringBuilder builder, LogEventInfo logEvent)
    {
        //TODO use options
        builder.Append("hello world!");
    }
}
```

Register - as soon as possible - e.g. in `main()`, `application_start()`.

```
//register under "hello-world"
ConfigurationItemFactory.Default.LayoutRenderers
    .RegisterDefinition("hello-world",
        typeof(MyNamespace.HelloWorldLayoutRenderer ));
```

Usage

```
 ${hello-world} - raises exception: required parameter Config2 isn't set
 ${hello-world:Config2=abc} - OK, Config2 property set
 ${hello-world:true:config2=abc} - default parameter (Caps) set to true
 ${hello-world:true:config2=abc:config1=yes} - all the three properties set.
```

Read Extending NLog online: <https://riptutorial.com/nlog/topic/7194/extending-nlog>

Chapter 3: Troubleshooting NLog

Examples

Enable the internal logger in nlog.config

In case of problems, enable the internal logger

```
<nlog internalLogFile="c:\log.txt" internalLogLevel="Trace">
  <targets>
    <!-- target configuration here -->
  </targets>
  <rules>
    <!-- log routing rules -->
  </rules>
</nlog>
```

Enable the internal logger programmatically

In case of problems, enable the internal logger.

C# example:

```
// set internal log level
InternalLogger.LogLevel = LogLevel.Trace;

// enable one of the targets: file, console, logwriter:

// enable internal logging to a file (absolute or relative path. Don't use layout renderers)
InternalLogger.LogFile = "c:\\log.txt";

// enable internal logging to the console
InternalLogger.LogToConsole = true;

// enable internal logging to the console (error stream)
InternalLogger.LogToConsoleError = true

// enable internal logging to the Trace
InternalLogger.LogToTrace = true

// enable internal logging to a custom TextWriter
InternalLogger.LogWriter = new StringWriter(); //e.g. TextWriter writer =
File.CreateText("C:\\perl.txt")
```

Read Troubleshooting NLog online: <https://riptutorial.com/nlog/topic/7190/troubleshooting-nlog>

Chapter 4: Write logs to files

Examples

Write to single file

Write all events with level: debug, info, warn, error and fatal to one file:

```
<?xml version="1.0" ?>
<nlog xmlns="http://www.nlog-project.org/schemas/NLog.xsd"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

  <targets>
    <target name="file" xsi:type="File"
            layout="${longdate} ${logger} ${message} ${exception}"
            fileName="${basedir}/logs/logfile.txt" />
  </targets>

  <rules>
    <logger name="*" minlevel="Debug" writeTo="file" />
  </rules>
</nlog>
```

Write one log file per day

Create one log file for each day. The log files will have the following names (dependent of culture)

```
2016-06-05.log
2016-06-06.log
2016-06-07.log
...
```

```
<?xml version="1.0" ?>
<nlog xmlns="http://www.nlog-project.org/schemas/NLog.xsd"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

  <targets>
    <target name="file" xsi:type="File"
            layout="${longdate} ${logger} ${message} ${exception}"
            fileName="${basedir}/${shortdate}.log" />
  </targets>

  <rules>
    <logger name="*" minlevel="Debug" writeTo="file" />
  </rules>
</nlog>
```

Read Write logs to files online: <https://riptutorial.com/nlog/topic/7192/write-logs-to-files>

Credits

S. No	Chapters	Contributors
1	Getting started with nlog	4444 , Community , Julian
2	Extending NLog	Julian
3	Troubleshooting NLog	Julian
4	Write logs to files	Julian