

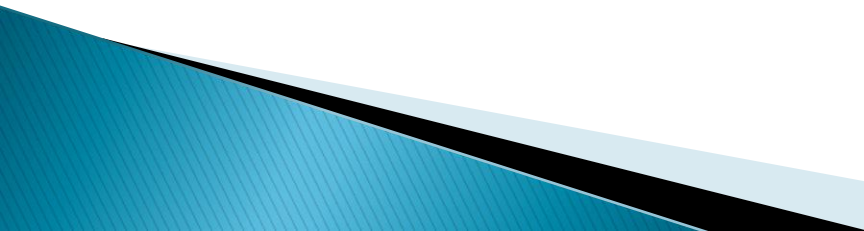
Introduction to log4net

Rob Prouse

rob@prouse.org

<http://www.alteridem.net>

Features

- ▶ **Fast and flexible**
 - ▶ **Hierarchical, named logging categories**
 - ▶ **Multiple logging levels**
 - ▶ **Output to multiple logging targets**
 - ▶ **Dynamic XML Configuration**
 - ▶ **Thread Safe**
 - ▶ **Format of logs is easily changed**
 - ▶ **Proven architecture (log4j)**
 - ▶ **Modular and extensible design**
 - ▶ **Support for multiple frameworks**
- 

Alternatives

- ▶ **Enterprise Library 3.1 Logging Application Block**
 - <http://msdn2.microsoft.com/en-us/aa480464.aspx>
 - From Microsoft Patterns & Practices
 - Possibly a good choice if you are already using Enterprise Library
 - A bit heavy-weight
 - Logging requires more code
- ▶ **NLog**
 - <http://www.nlog-project.org>
 - API is similar to log4net
 - Easy to configure
 - BSD License
 - .NET, C/C++ and COM interfaces
 - Project hasn't been updated since its 1.0 release in Sept 2006
- ▶ **.NET Logging Framework**
 - <http://www.theobjectguy.com/dotnetlog/>
 - Lightweight
 - Not as flexible
 - Very little community support
- ▶ **Roll your own**
 - We are paid to solve business problems, not to write code
 - Error prone
 - Can affect performance

License

- ▶ **Apache License, Version 2.0**
- ▶ **Free to use, modify and distribute**
- ▶ **Can use in commercial software**
- ▶ **Does not require modified versions to use same license**
- ▶ **Only requires that a notice informing recipients that Apache licensed code has been used.**
- ▶ **Two files must be put in the top directory of the software**
 - **LICENSE** – a copy of the license itself
 - **NOTICE** – a list of the licensed libraries used and their developers.
- ▶ ***** IANAL *****

Logger Hierarchies

- ▶ Can be thought of as Categories
- ▶ Organized into a named hierarchical structure
- ▶ Allow you to turn on logging for specific parts of your application
- ▶ Hierarchies are defined like namespaces in .NET
 - a.b.c
 - System.Windows.Forms
- ▶ Turning on logging for a root category, turns on logging for all children
- ▶ Root logger
 - It always exists
 - It cannot be retrieved by name
 - It always has an assigned level

Log Levels

- ▶ **5 log levels, plus two configuration options**
 - **FATAL**
 - **ERROR**
 - **WARN**
 - **INFO**
 - **DEBUG**
 - **ALL**
 - **OFF**
- ▶ **Log levels are inclusive**
 - **If log is set to INFO, all logs above that level (WARN, ERROR, and FATAL) are also logged.**

Code Example

▶ Getting Started

1. Add a reference to log4net
2. Add an XmlConfigurator attribute to your assembly
3. Add an application configuration file
4. Add a log4net using statement
5. Get a static ILog member for your class using `LogManager.GetLogger()`
6. Use the static ILog member to log

Appenders

- ▶ **Appenders are the destination for the logs**
- ▶ **Ships with most appenders you will need**
- ▶ **Easy to write your own**
- ▶ **Multiple appenders can be attached to each logger**
- ▶ **Each appender can receive logs from multiple loggers**

- AdoNetAppender
- AnsiColorTerminalAppender
- AspNetTraceAppender
- ColoredConsoleAppender
- ConsoleAppender
- EventLogAppender
- FileAppender

- LocalSyslogAppender
- MemoryAppender
- NetSendAppender
- OutputDebugStringAppender
- RemoteSyslogAppender
- RemotingAppender
- RollingFileAppender

- SmtpAppender
- TelnetAppender
- TraceAppender
- UdpAppender

Logging Contexts

- ▶ **Allows you to place data to be logged in a global, thread, or event context**
 - `log4net.GlobalContext`
 - `log4net.ThreadContext`
 - `log4net.ThreadLogicalContext`
 - `log4net.Core.LoggingEvent`
- ▶ `log4net.ThreadContext.Properties["name"] = value;`
- ▶ **PatternLayout renders the value of a named property using the `%property{name}` syntax.**
- ▶ **Context Stacks available**

Code Example

▶ Logging Contexts

- Add item that you want to log across calls to the context
- Add %property to your configuration

Configuration

- ▶ **Fully configurable programmatically**
- ▶ **Easier and more flexible to use XML configuration files**
- ▶ **Several ways to specify configuration file**
 - Application config file (*AssemblyName.config* or *web.config*)
 - `XmlConfiguratorAttribute` on the assembly
 - `XmlConfigurator.Configure(filename)`
 - `XmlConfigurator.ConfigureAndWatch(filename)`

Configuration Example

- ▶ **Specify config file in Code**
 - Add using `log4net.Config`
 - Add `XmlConfigurator.ConfigureAndWatch`
- ▶ **A simple configuration example**
- ▶ **<Logger>**
- ▶ **<root>**
- ▶ **<appender>**

Layouts

- ▶ **Allows customization of the output format**
- ▶ **PatternLayout most commonly used**
 - `%timestamp [%thread] %-5level %logger - %message%newline`
 - `176 [main] INFO Com.Foo.Bar - Located nearest gas station.`
- ▶ **See docs for other layouts including Xml, Simple and Exception layouts.**

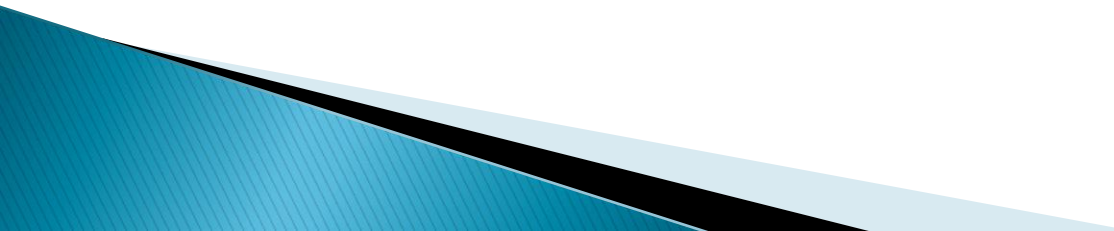
Filters

- ▶ **Allow appenders to filter messages delivered to them**
- ▶ **Allows fine control when events are logged to multiple appenders**
- ▶ **LevelRangeFilter most often used**
- ▶ **Several Filters available**
 - `log4net.Filter.DenyAllFilter`
 - `log4net.Filter.LevelMatchFilter`
 - `log4net.Filter.LevelRangeFilter`
 - `log4net.Filter.LoggerMatchFilter`
 - `log4net.Filter.PropertyFilter`
 - `log4net.Filter.StringMatchFilter`

Configuration Example

- ▶ `<layout>`
- ▶ `<filter>`

Object Renderers

- ▶ **log4net renders objects using their ToString() method**
 - ▶ **If you want more control over that, you can create and use your own renderer**
 - ▶ **Derive from IObjectRenderer interface**
- 

Best Practices

- ▶ **Logging is only useful if you use it**
- ▶ **Two styles of logging**
 - Logging by software component
 - Logging by functional area
- ▶ **Log all exceptions, not just their message strings**
- ▶ **Use the format logging methods where appropriate**
- ▶ **Short circuit expensive logging with the various Is...Enabled properties**
- ▶ **Advice on Log Levels**
 - **FATAL** - For very serious errors that may crash the program
 - **ERROR** - For errors that may corrupt data or cause improper program behaviour
 - **WARN** - For errors that have been handled but that developers may want to track
 - **INFO** - Non-verbose logging of program execution
 - startup of sub-systems,
 - timings,
 - remoting calls, etc.
 - Not for verbose output or within loops that may slow down the program.
 - **DEBUG** – For verbose or low level debugging information
 - mainly used to help the developer debug the system
 - can be used in loops, for dumping data, detailed program flow, etc.

Links

- ▶ **log4net** - <http://logging.apache.org/log4net/>
 - Download
 - Config Examples
 - FAQ
 - SDK Reference
 - Manual
- ▶ **My blog** - <http://www.alteridem.net>
- ▶ **O'Reilly – Using log4net**
<http://www.ondotnet.com/pub/a/dotnet/2003/06/16/log4net.html>

Q&A

Rob Prouse

rob@prouse.org

<http://www.alteridem.net>