**.NET Logging Platforms**

**NLog**

Free logging platform for .NET with rich log routing and management capabilities. Process diagnostic messages emitted from any .NET language.

* contextual information (date/time, severity, thread, process, env vars)
* formatting preferences
* multiple output (file, database, event log, network, console, email, MSMQ)
* .NET, .NET CF, C, C++, COM – API support
* programmatic config file
* advanced routing using buffering, async logging, load balancing, failover
* Complex Config File

**Clog**

A customizable log provider system that allows you to harness your existing logging system to log client side messages to your server using WCF.

**CuttingEdge.Logging**

Implements the .NET 2.0 Provider Model for tracing messages.

* multiple output formats
* .NET 2.0 provider model design (same configuration and extensibility model)
* pluggable into an existing app without recompilation
* .NET CE, Silverlight not supported

**SimpleLoggingFacade**

Provides you with a common interface that decouples the logging framework of your choice (log4net, EntLib, NLog) from your code. This eliminates dependencies on a given framework, thus allowing you to switch / combine frameworks at any time. Furthermore, SLF’s modular architecture allows you to plug-in custom logging strategies very easily.

Basically it provides you with a common interface that decouples the used logging framework from your code:

1. ILogger logger = LoggerService.GetLogger();
2. logger.Info( "hello world" );

* Configurable through code or configuration files (app.config).
* Support for named loggers with hierarchical fallback mechanisms.
* Support for common logging frameworks such as log4net, NLog and EntLib out of the box.
* Silverlight support.
* Writing your own pluggable logging strategy is easy.
* You can exchange the underlying logging mechanism at any time, even at runtime.
* Extensible and modular architecture. Writing your own façade can be as simple as overriding one single method, but you can cover complex scenarios, should you have to.
* Robust: In case of invalid configurations or logging instructions, SLF rather outputs debug warnings instead of crashing your application at runtime.
* A ton of copy-and-paste ready samples for different use cases.
* .NET 2.0 compatible binaries.

**MS Labs – Simple Logging Library**

* simple and small logging library
* configure the entire library via code or configuration file.
* sensible defaults
* takes advantage of .NET 3.5 functionality
* .NET delegates instead of interfaces as an extensibility mechanism

**ASP.NET Weberror Logging - Server Exceptions and Client side error logger**

Log and report unhandled/handled exceptions in ASP.NET web applications on server side as well as on client side for Ms-Ajax application. Inspired by ELMAH but with completely different architecture, design and capabilities.

* Logging of nearly all unhandled exceptions
* All errors are serialized and logged in the App\_Data folder
* A web page to remotely view the entire log of recoded exceptions.
* A web page to remotely view the full details of any one logged exception. In many cases, you can review the original yellow screen of death that ASP.NET generated for a given exception, even with customErrors mode turned off.

**Yet Another Exception Logging System Description**

In this DLL you can find a LogHelper, that can help you to log the Exceptions in your applications. The LogHelper use the .net provider pattern to facilitate extend it or make your own helper. With a simple call you recive a email or write a log file with all the exception information, that can help you to trace it.

**CodePlex.Diagnostics**

Designed to provide support for exception management and logging within projects written to target the Microsoft .NET platform.

* Microsoft .NET 2.0 Provider design pattern

**TraceExtension**

This project is based on Microsoft’s sample application called ‘TraceExtension’ + following improvements:

* Support for multi-thread
* Extended control in Web.config to turn logging on and off
* Exception handling. You may want tailor the exception handling to your needs

**DotNetLoggingProvider**

This is generic provider based loggin provider for dotnet which allows you to log exception or messages from application using 3 providers.

* simple text based files
* Log4net
* Enterprise library

**NTrace**

A library that provides flexible, high-performance, and zero-config trace logging for your C# applications. NTrace is a combination of a runtime framework and a code preprocessor that provides an API for using Event Tracing for Windows (ETW) in your managed code. In practice, it's a lot like the Windows Trace Preprocessor (WPP).

There are two high-level reasons why you'd want to use NTrace. The first is that it allows you to have trace instrumentation that can be enabled and disabled at runtime without having to modify any configuration files. The second is that the tracing capabilities provided by the ETW subsystem are very efficient and flexible. We've tried to preserve as much of the simplicity of using Trace.WriteLine while providing access to all of the features of WPP/ETW. The end result is that you have a great way to troubleshoot your code when it's out in the field.

**Realtime Trace**

This library has been designed to have almost zero-impact on your source code, zero-impact on your deployment, and zero-bug on your deployed software.

* Track any error at execution time in your .Net application;
* Track any error in the release build without the Trace being enabled ;
* Store any error and its context (ie : the values of any members,variables or objects used around the Exception Event) in a data storage provider for later analysis ;
* Have a tool to diagnose and locate the error in source code, even when the application is deployed and running on a production machine ;
* Track and correct any bug that cannot be reproduced on a development machine ;
* Have a tool to measure how buggy or how stable your application is ;
* Deliver an error-free application at the first release ;
* Build industrial application that will need almost no support at all.

**DistriLog**

Set of libraries that allow you to handle distributed logging. This is aimed at applications that are installed on multiple machines and instead of having a central log server (that may slow down the application due to network latency), a local log is created. A synchronization process then unifies these logs into a central SQL database.

The basic idea for this was due to the need of applications that may be distributed over multiple computers, but you do not want to use a central server/web service based logging infrastructure due to performance issues. Distrilog logs to a local database so there is no network latency, but then it also synchronizes with a central database so that you can have a unified view of all the log information. Can be used for windows application or load balanced web applications.

**NetLib.Log**

An easy to use library that utilizes extension methods to provide a clean and easy to use API to log Exceptions. You'll no longer to have to wonder why after compiling you have all the empty catch blocks.