|  |
| --- |
| かずきのBlog@hatena |
| Enterprise Library 5.0入門 |
| Enterprise Libraryの基本的な使用方法 |
|  |
| **okazuki** |
| **2012/04/03** |

|  |
| --- |
| 2011年5月にリリースされたEnterprise Library 5.0の入門になります。 |

改版履歴

|  |  |
| --- | --- |
| **日時** | **改版内容** |
| 2012/04/04 | 初版作成 |

目次

[1. はじめに 3](#_Toc321344590)

[1.1. 前提条件 3](#_Toc321344591)

[2. Enterprise Libraryとは 3](#_Toc321344592)

[1.2. Enterprise Libraryの構成 3](#_Toc321344593)

[3. Hello world 4](#_Toc321344594)

# はじめに

ここでは、Microsoftのpattern & practiceチームが2011年5月にリリースしたEnterprise Library 5.0について著者の学習もかねて記載したものになります。記載内容に誤りを見つけた方はお手数をおかけしますが下記メールアドレスまたは、Twitterアカウントまで連絡ください。

* メール：[k\_ota28@hotmail.com](mailto:k_ota28@hotmail.com)
* Twitter: okazuki

## 前提条件

本書では、Visual Studio 2010を使用して.NET Framework 4(Client Profileではない)を使用しています。

## Enterprise Libraryとは

Enterprise Libraryとは、Microsoftのpattern & practiceチームが開発しているエンタープライズアプリケーションの開発のベストプラクティスを集めたライブラリです。このライブラリの特徴は、UnityというDIコンテナを軸にしてEnterprise Libraryが提供している機能(Application Blockと呼ばれる)を組み合わせて使用できるという点です。そのためEnterprise Libraryが提供している機能のうちのごく一部を使うことも、全機能使うことも問題なく出来ます。

また、Unityを使用せずに自分のプログラムから直接使用することも可能なように作成されています。本書では、UnityをDIコンテナとして使用して各Application Blockの機能を利用する方法を説明します。

### Application Blockとは

Application Blockとは、Enterprise Libraryを構成するある程度まとまった機能のことです。Application Block単体でも動作するように作成されていますが、一部のApplication Blockは他のApplication Blockに依存しているものもあります。以下にEnterprise LibraryのApplication Blockを示します。

* Caching Application Block  
  アプリケーション内でキャッシュ機能を提供します。
* Cryptography Application Block  
  データの暗号化・複合とハッシュを生成する機能を提供します。
* Data Access Application Block  
  データベースへのアクセス機能を提供します。
* Exception Handling Application Block  
  例外処理の機能を提供します。
* Logging Application Block  
  ログ出力機能を提供します。
* Policy Injection Application Block  
  機能横断的なPolicyをアプリケーションに適用する機能を提供します。
* Security Application Block  
  承認の規則（操作の許可や拒否など）を構成・管理する機能を提供します。
* Validation Application Block  
  値の妥当性検証の機能を提供します。

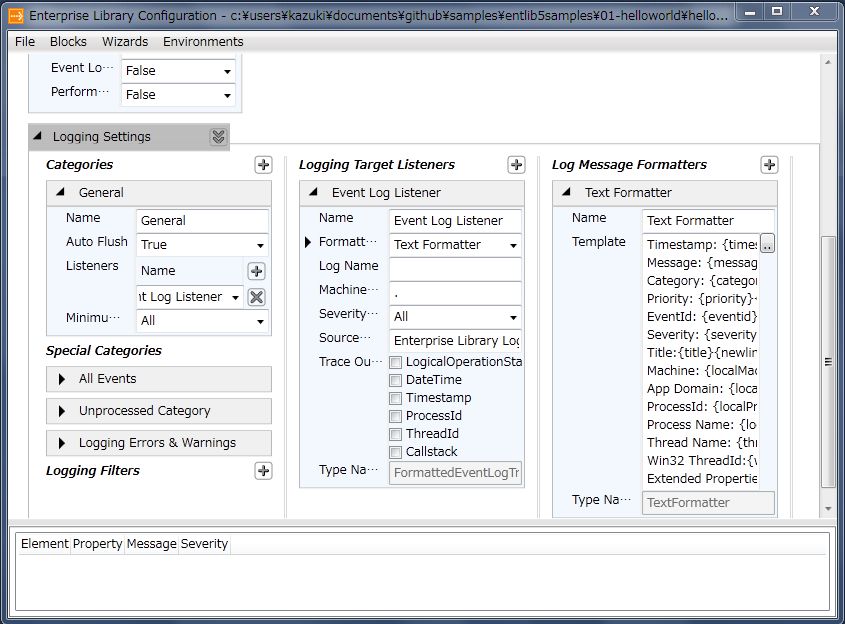
説明からもわかるように、Application Blockは特別な機能を提供しているわけではありません。一般的なアプリケーションであれば実装している機能になります。これらの機能の使い方を理解し適切に選択して使用できるようになると、アプリケーションを効率よく開発することが出来るようになると思います。（もしくは、どのような機能セットを提供しているのか、どのような点に留意して作成されているのかという参考にすることも出来ます）

## Enterprise Libraryの構成

Enterprise Libraryの基本的な使い方の流れは以下のようになります。

1. Enterprise Libraryで使用するApplication Blockを選択します。
2. 関連するアセンブリを参照に追加します。
3. Enterprise Libraryの構成を行います。以下のような方法があります。
   1. 構成ファイル(app.configやweb.config)を使用する方法
   2. 任意のファイルを使用する方法
   3. コードで構成する方法
4. Enterprise Libraryのコンテナを初期化します。
5. Enterprise Libraryのコンテナから必要な機能を取り出して使用します。

Enterprise Libraryには下図のような構成ツールが付属していて、これによりグラフィカルにapp.configやweb.configを設定することが出来ます。ここで設定した内容によって各アプリケーションブロックの機能をカスタマイズすることが出来ます。

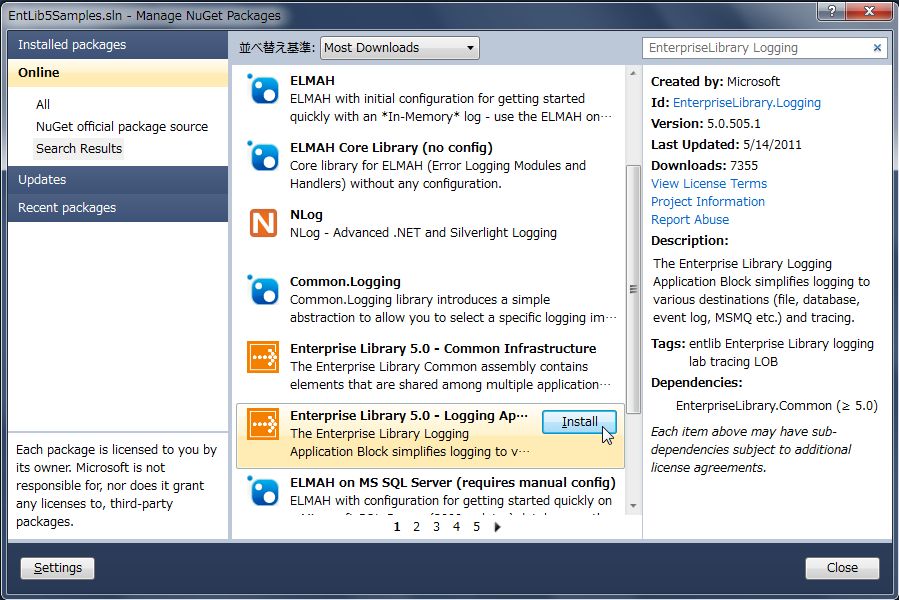


Enterprise Libraryでは、app.configなどの構成ファイル以外にもアプリケーションから構成情報を組み立てるFluent APIも提供しています。本書では、サンプルプログラムとしての見通しを優先するためFluent APIによる構成情報の組み立てを優先して使用します。

## Hello world

ここでは、Loggin Application Blockを使ってHello worldというログをファイルに出力する方法を通じて基本的なEnterprise Libraryの使い方の流れについて説明します。

HelloWorldという名前でコンソールアプリケーションを作成します。そしてNuGet Package Managerから「EnterpriseLibrary Logging」で検索をして「Enterprise Library 5.0 – Logging Application Block」をプロジェクトに追加します。

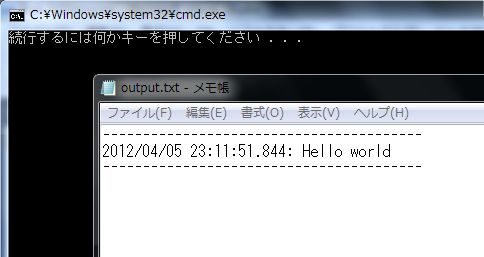


プロジェクトに参照が追加されたら、続けて参照の追加からSystem.Configurationを追加します。以上でプロジェクトの参照の設定は終了です。（昔はこの参照設定を整えるだけで一仕事だったのですが、いい時代になったものです…）Program.csを開いて下記のように編集します。

1. using System.Diagnostics;
2. using Microsoft.Practices.EnterpriseLibrary.Common.Configuration;
3. using Microsoft.Practices.EnterpriseLibrary.Logging;
4. namespace HelloWorld
5. {
6. class Program
7. {
8. static void Main(string[] args)
9. {
10. // 構成情報を組み立てる
11. var builder = new ConfigurationSourceBuilder();
12. builder.ConfigureLogging()
13. .SpecialSources
14. .AllEventsCategory
15. .SendTo
16. .FlatFile("FlatFileListener")
17. .FormatWith(
18. new FormatterBuilder()
19. .TextFormatterNamed("TextFormatter")
20. .UsingTemplate("{timestamp(local:yyyy/MM/dd HH:mm:ss.fff)}: {message}"))
21. .ToFile("output.txt");
22. // 組み立てた構成情報からConfigurationSourceを作成
23. var config = new DictionaryConfigurationSource();
24. builder.UpdateConfigurationWithReplace(config);
25. // 構成情報を元にEnterpriseLibraryのコンテナの初期化
26. EnterpriseLibraryContainer.Current = EnterpriseLibraryContainer.CreateDefaultContainer(config);
27. // EnterpriseLibraryのコンテナからLogging Application BlockのLog書き込み部品を取得
28. var logger = EnterpriseLibraryContainer.Current.GetInstance<LogWriter>();
29. // ログに出力する
30. logger.Write("Hello world");
31. // ログを表示
32. Process.Start("output.txt");
33. }
34. }
35. }

細かいAPIの利用方法は置いておいてコード全体の流れが、構成→コンテナの初期化→コンテナから部品の取り出し→使用という流れになっていることが確認できると思います。構成とコンテナの初期化をアプリケーションのエントリーポイントなどの初期化処理で行い、残りの部分で適時コンテナから必要な部品を取り出して使用するという流れになります。（もしくは、Dependency Injectionしてもらう）

このプログラムの実行結果は以下のようになります。ログが出力され、ログファイルが開かれます。



次章から、各Application Blockの使用方法を確認していきます。

# Application Block

ここでは、Enterprise Libraryで提供されるApplication Blockを１つずつ取り上げて説明します。

## Logging Application Block

ここではLogging Application Blockについて説明します。業務アプリケーションにおいて、ログは必須の構成要素です。地味な機能ですが、テスト時やリリース後の障害発生時の問題切り分けの重要な情報になります。また適切なログを出力して監視することで、障害の予兆を検知したり様々な利用方法があります。

.NETでは、System.Diagnostics.Traceを使ったログ出力機能が標準で備わってますが、