eBook Gratuit

APPRENEZ log4net

eBook gratuit non affilié créé à partir des **contributeurs de Stack Overflow.**

#log4net

Table des matières

À propos 1

Chapitre 1: Démarrer avec log4net 2

Remarques 2

Examples 2

Configurer log4net 2

Chapitre 2: Log4Net Dépannage 9

Introduction 9

Examples 9

Activer le débogage interne 9

Appenders tamponnés 10

Configurez () non appelé ou appelé plusieurs fois 10

Le composant de fichier n'écrit pas 10

Crédits 12

À propos

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: log4net

It is an unofficial and free log4net 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 log4net.

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

Chapitre 1: Démarrer avec log4net

Remarques

Cette section fournit une vue d'ensemble de ce qu'est log4net et pourquoi un développeur peut vouloir l'utiliser.

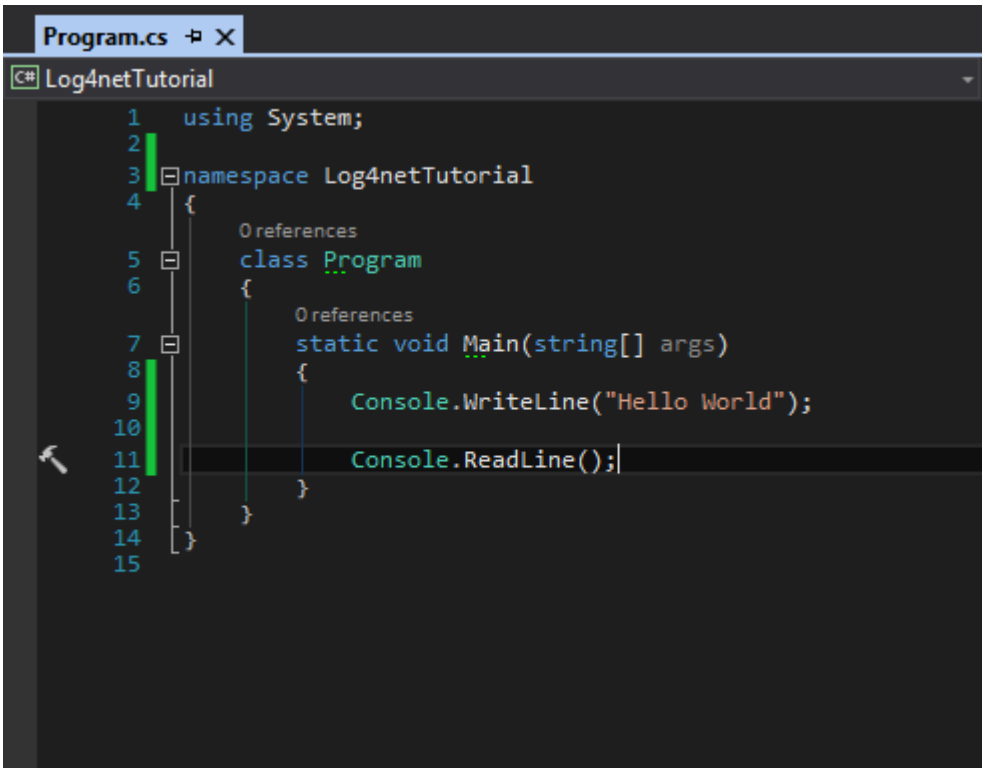
Il devrait également mentionner tous les grands sujets dans log4net, et établir un lien avec les sujets connexes. La documentation de log4net étant nouvelle, vous devrez peut-être créer des versions initiales de ces rubriques connexes.

Examples

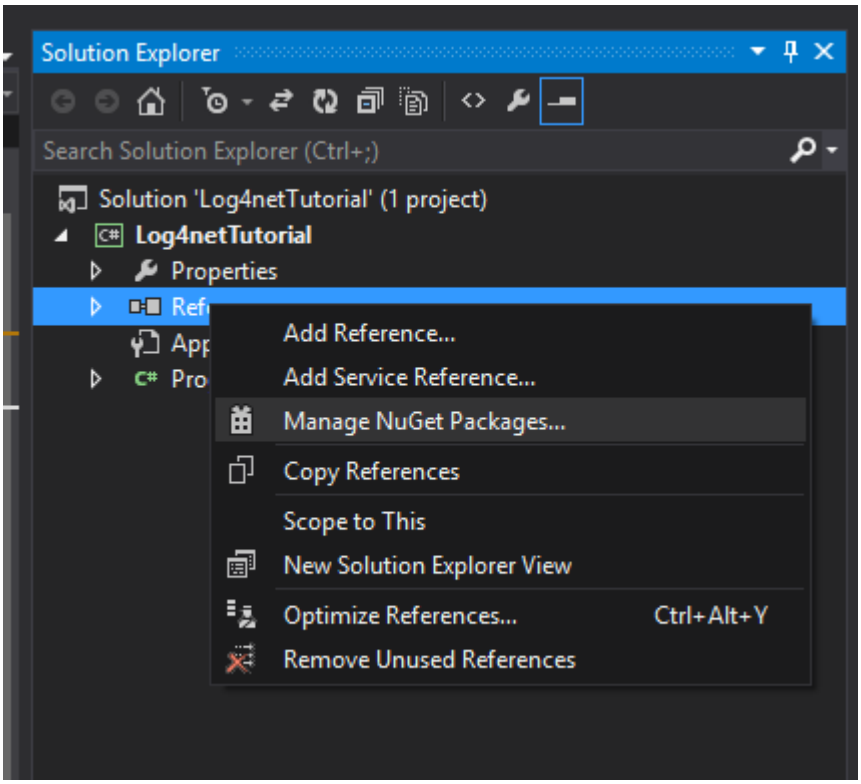
Configurer log4net

Laissez-nous créer une application console *Hello World* simple et connectez-vous quelque chose à la console à l'aide de log4net. Une fois que nous avons cette opération, nous pouvons l'étendre pour l'utiliser dans des scénarios de développement réels dans les exemples suivants. Commençons petit et construisons à partir de là.

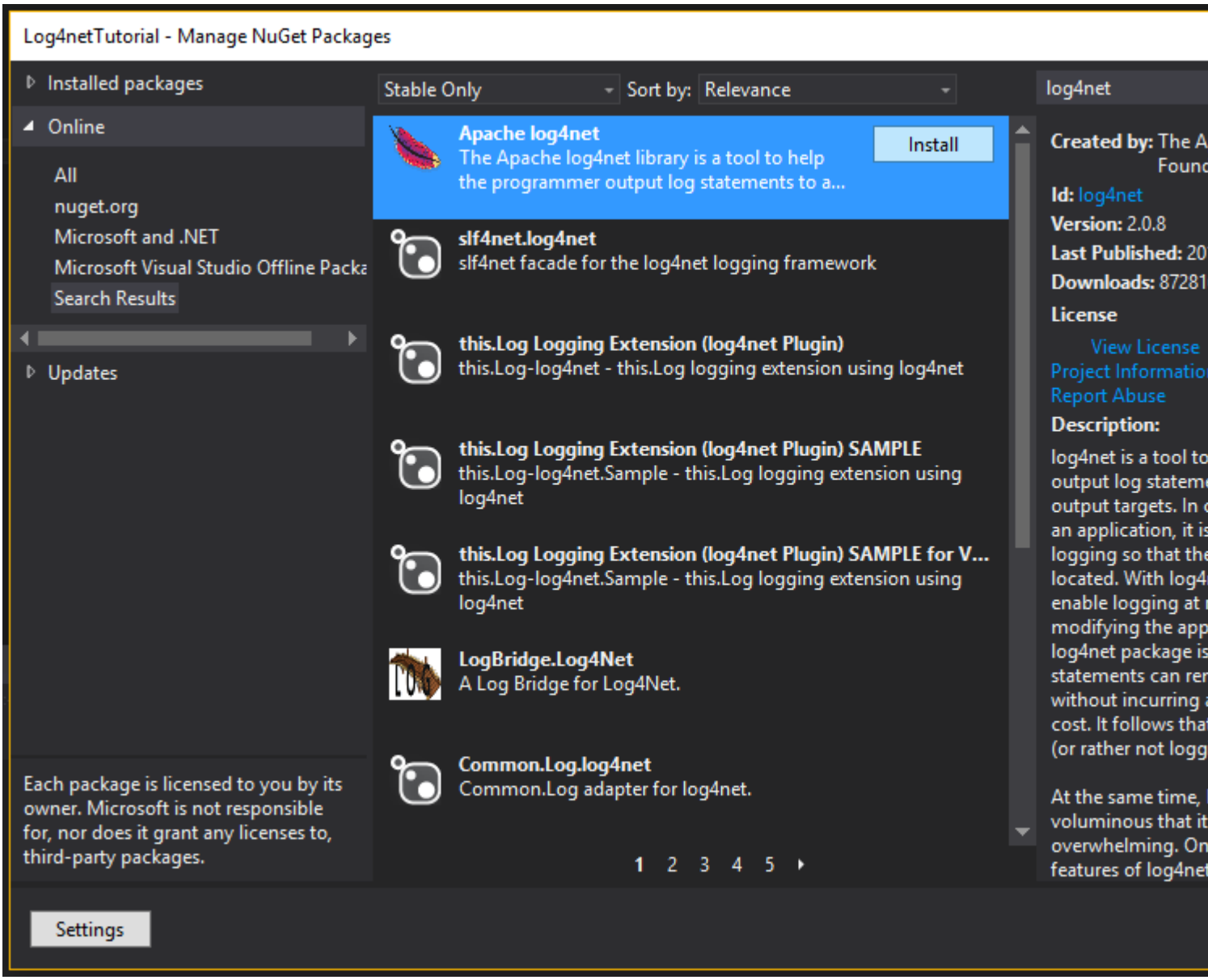
Tout d'abord, nous devons créer un projet de console simple dans Visual Studio et `WriteLine("Hello World")` comme indiqué ci-dessous.



Ensuite, nous devons ajouter la bibliothèque log4net à notre projet. Nous devons donc cliquer avec le bouton droit sur *Références* et sélectionner *Gérer les packages NuGet.*



Puis recherchez **log4net** et installez



Nous avons maintenant ajouté log4net.dll à notre projet.

Maintenant, nous devons configurer notre *app.config* (ou *web.config* si c'est une application Web). Cette partie est un peu délicate mais ne paniquez pas. Nous allons le parcourir pas à pas et nous connecter.

Sous *app.config,* nous devons d'abord créer une section de configuration sous configuration. Nous devons fournir un nom et un type comme indiqué ci-dessous:

```
<configSections>
  <section name="log4net" type="log4net.Config.Log4NetConfigurationSectionHandler,Log4net"/>
</configSections>
```

Ce que nous avons fait ici, c'est que nous disons que nous allons créer dans notre fichier de configuration une section appelée `log4net` qui doit être recherchée. Tout ce qui concerne la journalisation sera ici. De la création de notre fichier journal, de la longueur et des informations à consigner. Alors maintenant, allons-y et créez la section `log4net` .

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <configSections>
    <section name="log4net" type="log4net.Config.Log4NetConfigurationSectionHandler,Log4net"/>
  </configSections>
  <log4net>

  </log4net>
  <startup>
      <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5" />
  </startup>
</configuration>
```

La première chose que nous allons ajouter sous notre section log4net est un appender. Un appender est essentiellement un outil de journalisation. C'est ce que nous voulons enregistrer nos journaux. Il existe de nombreux ajouts disponibles pour la connexion aux fichiers, à la base de données, etc. Dans cet exemple, nous allons nous connecter à notre fenêtre de console pour créer un éditeur de console:

```
<log4net>
  <appender name="ConsoleAppender" type="log4net.Appender.ConsoleAppender">

  </appender>
</log4net>
```

Ensuite, nous devons définir une mise en page:

```
<log4net>
  <appender name="ConsoleAppender" type="log4net.Appender.ConsoleAppender">
    <layout type="log4net.Layout.PatternLayout">

    </layout>
  </appender>
</log4net>
```

Sous la présentation, nous définirons ce que nous voulons afficher sur la console et comment nous voulons l’afficher. Pour cela, nous avons besoin d'un modèle de conversion. Il existe de nombreux types de modèles que nous pouvons utiliser, pour cet exemple, nous allons nous en tenir à un modèle assez simple.

```
<log4net>
  <appender name="ConsoleAppender" type="log4net.Appender.ConsoleAppender">
    <layout type="log4net.Layout.PatternLayout">
      <conversionPattern value="%date{ABSOLUTE} [%thread] %level %logger - %message%newline"/>
    </layout>
  </appender>
</log4net>
```

Ce que nous faisons ci-dessus est de dire que nous avons besoin des informations suivantes: L'horodatage absolu, le thread en cours d'exécution qui génère l'exception et le niveau de journalisation. Il existe 7 niveaux de journalisation fournis par log4net.

- OFF - rien n'est enregistré (ne peut être appelé)

- FATAL
- ERREUR
- PRÉVENIR
- INFO
- DÉBOGUER
- ALL - tout est enregistré (ne peut pas être appelé)

Cependant, sur les 7, nous ne pouvons utiliser que 5 (DEBUG, INFO, WARN, ERROR et FATAL). Déclarer le niveau de journalisation à DEBUG signifie qu'il va tout enregistrer, à savoir DEBUG, INFO, WARN, ERROR et FATAL. Cependant, la déclaration du niveau de journalisation à WARN consignera uniquement WARN, ERROR et FATAL. J'espère que vous comprenez la hiérarchie.

La dernière chose dont nous avons besoin sous *app.config* est une section racine. La section racine héberge nos enregistreurs de haut niveau et le niveau de connexion. Il est important de comprendre que tout hérite de la racine, donc aucun appender ne se connectera à moins qu'il ne soit référencé sous la racine. Avec cela ajouté, *voici* comment votre *app.config* devrait ressembler à:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <configSections>
    <section name="log4net" type="log4net.Config.Log4NetConfigurationSectionHandler,Log4net"/>
  </configSections>
  <log4net>
    <appender name="ConsoleAppender" type="log4net.Appender.ConsoleAppender">
      <layout type="log4net.Layout.PatternLayout">
        <conversionPattern value="%date{ABSOLUTE} [%thread] %level %logger -
%message%newline"/>
      </layout>
    </appender>
    <root>
      <level value="DEBUG"/>
      <appender-ref ref="ConsoleAppender" />
    </root>
  </log4net>
  <startup>
      <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5" />
  </startup>
</configuration>
```

Ce n'est pas une mauvaise idée de copier-coller des sections du fichier de configuration, cependant, il est important de comprendre de quoi elles sont responsables. Maintenant que nous avons fini de configurer notre *app.config* , nous devons ajouter un peu de code à notre projet de console. Tout d'abord, nous devons définir une entrée unique qui doit être placée en dehors de votre classe.

```
[assembly: log4net.Config.XmlConfigurator(Watch = true)]
```

Placez-le juste en dessous de mes instructions using dans le fichier Program.cs. Ensuite, nous devons créer une instance du consignateur pour l'utiliser pour la journalisation. Ceci est fait une fois par classe.

```
private static readonly log4net.ILog log = log4net.LogManager.GetLogger
(System.Reflection.MethodBase.GetCurrentMethod().DeclaringType);
```

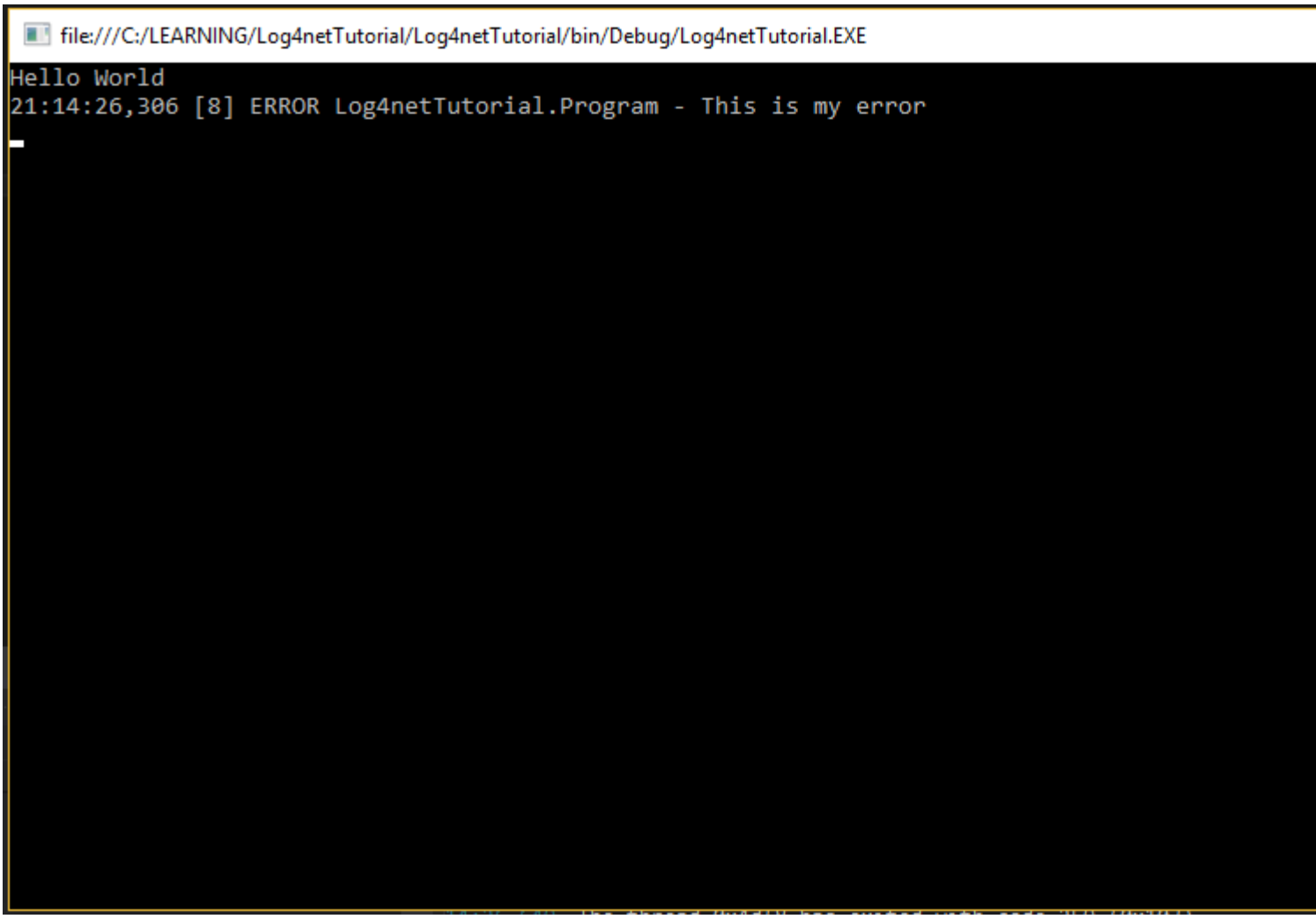
Et enfin, nous devons enregistrer quelque chose

```
log.Error("This is my error");
```

Votre fichier Program.cs devrait ressembler à ceci:

```
using System;

[assembly: log4net.Config.XmlConfigurator(Watch = true)]

namespace Log4netTutorial
{
    class Program
    {
        private static readonly log4net.ILog log =
log4net.LogManager.GetLogger(System.Reflection.MethodBase.GetCurrentMethod().DeclaringType);

        static void Main(string[] args)
        {
            Console.WriteLine("Hello World");
            log.Error("This is my error");

            Console.ReadLine();
        }
    }
}
```

Allez-y et lancez le programme, votre erreur doit être enregistrée dans la console:



Lire Démarrer avec log4net en ligne: https://riptutorial.com/fr/log4net/topic/9982/demarrer-avec-log4net

Chapitre 2: Log4Net Dépannage

Introduction

Log4net est un système de journalisation à arrêt automatique. Échec d'arrêt signifie qu'il arrête la journalisation sur une exception interne et n'interagit pas par conception avec le flux de programme. Sachant cela vous explique que le dépannage log4net n'est pas si facile. Si la journalisation échoue, votre programme ne le remarque pas. Vous voyez beaucoup de questions sur: pourquoi mon log4net logging ne fonctionne pas? Cet article explique le dépannage de base et les solutions les plus courantes.

Dans la plupart des cas, la première étape consiste à activer le débogage interne pour log4net.

Examples

Activer le débogage interne

Il existe deux manières d'activer le débogage interne dans log4net:

- Spécifiez l'option log4net.Internal.Debug dans le fichier de configuration de l'application
- Activer le débogage interne de log4net par programme

Spécifiez l'option log4net.Internal.Debug dans le fichier de configuration de l'application

C'est la méthode préférée pour activer le débogage interne, ajoutez la clé log4net.Internal.Debug au fichier app.config de votre application.

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <appSettings>
        <add key="log4net.Internal.Debug" value="true"/>
    </appSettings>
</configuration>
```

La journalisation du débogage démarre immédiatement au démarrage de l'application.

Activer le débogage interne de log4net par programme

Une deuxième manière consiste à le faire par programmation. Définissez la propriété log4net.Util.LogLog.InternalDebugging sur true:

```
log4net.Util.LogLog.InternalDebugging = true;
```

Sortie du journal de débogage interne

Les messages de débogage internes sont écrits sur la console et sur le System.Diagnostics.Trace. Lorsque vous devez consolider la sortie, vous pouvez rediriger le

System.Console.Out. Ou vous pouvez rediriger le message de trace vers un fichier:

```
<configuration>
...

<system.diagnostics>
    <trace autoflush="true">
        <listeners>
            <add
                name="textWriterTraceListener"
                type="System.Diagnostics.TextWriterTraceListener"
                initializeData="C:\tmp\log4net.txt" />
        </listeners>
    </trace>
</system.diagnostics>

...
</configuration>
```

Appenders tamponnés

Certains des appender log4net sont des appenders tamponnés. Ces utilisateurs enregistreront uniquement un certain nombre de messages enregistrés. Certains exemples sont SmtpAppender, RemotingAppender ou AdoNetAppender. Ces appenders ont un paramètre BufferSize:

```
<bufferSize value="100" />
```

Cela signifie que l'enregistreur se connectera lorsqu'il y aura 100 messages dans le tampon. Lorsque vous souhaitez tester l'appender, vous pouvez définir le bufferSize à 1.

Si vous voulez que le tampon à vider d'une erreur, vous pouvez utiliser un évaluateur:

```
<evaluator type="log4net.Core.LevelEvaluator">
     <threshold value="ERROR"/>
</evaluator>
```

Si la condition de l'évaluateur est remplie, le tampon sera vidé.

Configurez () non appelé ou appelé plusieurs fois

Si vous ne voyez aucune journalisation, vous pouvez vérifier si `Configure()` est appelé dans votre application. Le moyen le plus simple est de l'ajouter en tant qu'attribut à votre assembly:

```
[assembly: log4net.Config.XmlConfigurator(Watch = true)]
```

Ensuite, vous n'avez pas à ajouter `log4net.Config.XmlConfigurator.Configure()` n'importe où dans votre configuration. Ou vous pouvez ajouter `log4net.Config.XmlConfigurator.Configure()` dans l'une de vos méthodes de démarrage. Assurez-vous d'appeler la configuration une seule fois.

Le composant de fichier n'écrit pas

Si vous avez un appender de fichier, assurez-vous d’écrire dans un emplacement où l’utilisateur est autorisé à créer et à mettre à jour les fichiers. Sinon, la journalisation échouera. Vous pouvez le vérifier lorsque le débogage interne est activé.

Lire Log4Net Dépannage en ligne: https://riptutorial.com/fr/log4net/topic/10686/log4net-depannage

Crédits

S. No	Chapitres	Contributeurs
1	Démarrer avec log4net	Community, FortyTwo
2	Log4Net Dépannage	Peter