Comment faire pour tracer et déboguer dans Visual C#

Pour obtenir une version Microsoft Visual Basic .NET de cet article, reportez-vous à la section.
313417 .

Cet article fait référence à l'espace de noms bibliothèque de classes Microsoft.NET Framework suivant :

- System.Diagnostics

DANS CETTE TÂCHE.

Résumé

Cet article décrit comment utiliser les classes Trace et Debug . Ces classes sont disponibles dans le .NET Framework de Microsoft. Vous pouvez utiliser ces classes pour fournir des informations sur les performances d'une application lors du développement de l'application, ou après le déploiement en production. Ces classes ne sont qu'une partie des fonctionnalités d'instrumentation qui sont disponibles dans le.NET Framework.

Retour au début

Configuration requise

La liste suivante met en évidence le matériel recommandé, logiciel, infrastructure réseau et les service packs dont vous avez besoin :

- Microsoft Windows 2000 ou Microsoft Windows XP ou Microsoft Windows Server 2003
- Microsoft Visual C#

Cet article suppose également que vous êtes familiarisé avec le programme de débogage.

Retour au début

Description de la Technique

Les étapes de la section Création d'un exemple avec la classe Debugmontrent comment créer une application console qui utilise la classe Debug pour fournir des informations sur l'exécution du programme.

Lorsque le programme est exécuté, vous pouvez utiliser les méthodes de la classe Debug pour produire des messages qui vous permettent de contrôler l'ordre d'exécution de programme, afin de détecter les dysfonctionnements, ou pour fournir des informations de mesure des performances. Par défaut, les messages de la classe Debug produit s'affichent dans la fenêtre de sortie de l'environnement de développement intégré (IDE) Visual Studio.

L'exemple de code utilise la méthode WriteLine afin de produire un

message qui est suivi d'un terminateur de ligne. Lorsque vous utilisez cette méthode pour produire un message, chaque message s'affiche sur une ligne distincte dans la fenêtre Sortie.

Lorsque vous utilisez la méthode Assert de la classe Debug , la fenêtre Sortie affiche un message uniquement si une condition spécifiée a la valeur false. Le message s'affiche également dans la boîte de dialogue modale à l'utilisateur. La boîte de dialogue inclut le message, le nom du projet et le numéro de relevé Debug.Assert . La boîte de dialogue inclut également les boutons de trois commande suivants :

- **Abandonner :** Arrêt de l'application.
- **Réessayer :** L'application passe en mode débogage.
- **Ignorer :** L'application se poursuit.

L'utilisateur doit cliquer sur un de ces boutons avant que l'application puisse continuer.

Vous pouvez également diriger la sortie à partir de la classe Debug pour les destinations de la fenêtre Sortie. La classe Debug a une collection nommée d' écouteurs qui inclut des objets écouteur .

Chaque objet de l'écouteur surveille la sortie de débogage et dirige la sortie vers une cible spécifiée.

Chaque écouteur dans la collection de l'écouteur reçoive la sortie qui génère de la classe Debug . Utilisez la classe TextWriterTraceListener pour définir des objets écouteur . Vous pouvez spécifier la cible pour une classe d'écouteur TextWriterTraceListener via son constructeur.

Certaines cibles de sortie possibles sont les suivantes :

- La fenêtre de Console à l'aide de la propriété System.Console.Out .
- Un fichier texte (.txt) à l'aide de l'instruction System.IO.File.CreateText("FileName.txt") .

Après avoir créé un objet TextWriterTraceListener , vous devez ajouter l'objet à la collection Debug.Listeners doit recevoir la sortie de débogage.

Retour au début

Créer un exemple avec la classe Debug

1. Démarrez Visual Studio ou Visual C# Express Edition.
2. Créer un nouveau projet d'Application Console Visual C# nommé conInfo. Class1 est créé dans Visual Studio .NET. Program.cs est créé dans Visual Studio 2005.
3. Ajouter l'espace de noms suivante dans Class1 ou Program.cs en haut.

```
using System.Diagnostics;
```

4. Pour initialiser des variables pour contenir des informations sur un produit, ajouter les instructions de déclaration suivant à la méthode Main :

```
string sProdName = "Widget";
int iUnitQty = 100;
double dUnitCost = 1.03;
```

5. Spécifier le message qui produit de la classe en tant que premier paramètre d'entrée de la méthode WriteLine . Appuyez sur la combinaison de touches CTRL + ALT + O pour vous assurer que la fenêtre sortie est visible.

```
Debug.WriteLine("Debug Information-Product Starting ");
```

6. Pour une meilleure lisibilité, utilisez la méthode de retrait pour mettre en retrait les messages suivants dans la fenêtre Sortie :

```
Debug.Indent();
```

7. Pour afficher le contenu de variables sélectionnées, utilisez la méthode WriteLine comme suit :

```
Debug.WriteLine("The product name is " + sProdName);
Debug.WriteLine("The available units on hand are" + iUnit
Debug.WriteLine("The per unit cost is " + dUnitCost.ToStr
```

8. Vous pouvez également utiliser la méthode WriteLine pour afficher l'espace de noms et le nom de classe d'un objet existante. Par exemple, le code suivant affiche l'espace de noms System.Xml.XmlDocument dans la fenêtre Sortie :

```
System.Xml.XmlDocument oxml = new System.Xml.XmlDocument(
Debug.WriteLine(oxml);
```

9. Pour organiser la sortie, vous pouvez inclure un paramètre de catégorie comme facultatifs, deuxième entrée de la méthode WriteLine . Si vous spécifiez une catégorie, le format du message de fenêtre sortie est « catégorie : message. » Par exemple, la première ligne du code suivant affiche « champ : le nom du produit est Widget » dans la fenêtre Sortie :

```
Debug.WriteLine("The product name is " + sProdName,"Field
Debug.WriteLine("The units on hand are" + iUnitQty,"Field
Debug.WriteLine("The per unit cost is" + dUnitCost.ToStri
Debug.WriteLine("Total Cost is  " + (iUnitQty * dUnitCost
```

10. La fenêtre sortie peut afficher des messages uniquement si une condition spécifiée a la valeur true à l'aide de la méthode WriteLineIf de la classe Debug . La condition à évaluer est le premier paramètre d'entrée de la méthode WriteLineIf . Le deuxième paramètre de

WriteLineIf est le message qui s'affiche uniquement si la condition dans le premier paramètre a la valeur true.

```
Debug.WriteLineIf(iUnitQty > 50, "This message WILL appea
Debug.WriteLineIf(iUnitQty < 50, "This message will NOT a
```

11. Utilisez la méthode Assert de la classe Debug afin que la fenêtre Sortie affiche le message uniquement si une condition spécifiée a la valeur false :

```
Debug.Assert(dUnitCost > 1, "Message will NOT appear");
Debug.Assert(dUnitCost < 1, "Message will appear since dU
```

12. Créez les objets écouteur TextWriterTraceListener pour la fenêtre de la Console (tr1) et un fichier texte nommé Output.txt (tr2) et puis ajoutez chaque objet à la collection Listeners de débogage :

```
TextWriterTraceListener tr1 = new TextWriterTraceListener
Debug.Listeners.Add(tr1);

TextWriterTraceListener tr2 = new TextWriterTraceListener
Debug.Listeners.Add(tr2);
```

13. Pour une meilleure lisibilité, utilisez la méthode Unindent pour supprimer la mise en retrait pour les messages suivants du Génère de la classe Debug . Lorsque vous utilisez le tiret ainsi que les méthodes Unindent ensemble, le lecteur peut distinguer la sortie en tant que groupe.

```
Debug.Unindent();
Debug.WriteLine("Debug Information-Product Ending");
```

14. Pour vous assurer que chaque objet de l'écouteur reçoit toutes les sa sortie, appelez la méthode Flush pour les mémoires tampons de classe de débogage :

```
Debug.Flush();
```

Retour au début

À l'aide de la classe Trace

Vous pouvez également utiliser la classe Trace pour produire des messages ce moniteur, l'exécution d'une application. Les classes Trace et Debug partagent la plupart des méthodes pour produire une sortie, y compris les éléments suivants :

- **WriteLine**
- **WriteLineIf**
- **Indent**
- **Unindent**

- **Assert**
- **Flush**

Vous pouvez utiliser les classes Debug et la Trace séparément ou ensemble dans la même application. Dans un projet de Configuration de Solution Debug, la sortie de Trace et de débogage sont actifs. Le projet génère la sortie à partir de ces deux classes à tous les objets écouteur . Toutefois, un projet de Configuration de Solution version génère uniquement sortie à partir d'une classe de Trace . Le projet de Configuration de Solution version ignore les appels de méthodes de classe Debug .

```
Trace.WriteLine("Trace Information-Product Starting ");
Trace.Indent();

Trace.WriteLine("The product name is "+sProdName);
Trace.WriteLine("The product name is"+sProdName,"Field" );
Trace.WriteLineIf(iUnitQty > 50, "This message WILL appear");
Trace.Assert(dUnitCost > 1, "Message will NOT appear");

Trace.Unindent();
Trace.WriteLine("Trace Information-Product Ending");

Trace.Flush();

Console.ReadLine();
```

Retour au début

Vérifiez que tout fonctionne

1. Assurez-vous que le débogage est la configuration de solution en cours.
2. Si la fenêtre de **L'Explorateur de solutions** n'est pas visible, appuyez sur la combinaison de touches CTRL + ALT + L pour afficher cette fenêtre.
3. Cliquez sur **conInfo**, puis cliquez sur **Propriétés**.
4. Dans le volet gauche de la page de propriétés conInfo, sous le Dossier de **configuration** , assurez-vous que la flèche pointe vers Le **débogage**.

   Remarque Dans Visual C# 2005 et Visual C# 2005 Express Edition, cliquez sur **débogage** dans la page **conInfo** .
5. Au-dessus du dossier **Configuration** , dans la Zone de liste déroulante de **configuration** , cliquez sur **Active (Debug)** ou **Déboguer**, puis cliquez sur **OK**. Dans Visual C# 2005 et Visual C# 2005 Express Edition, cliquez sur **Active (Debug)** ou **de débogage** dans la zone de liste déroulante de **Configuration** dans la page **Déboguer** et puis cliquez sur **Enregistrer** dans le menu **fichier** .

6. Appuyez sur CTRL + ALT + O pour afficher la fenêtre Sortie.
7. Appuyez sur la touche F5 pour exécuter le code. Lorsque la **Échec de l'assertion** la boîte de dialogue s'affiche, cliquez sur **Ignorer**.
8. Dans la fenêtre de la Console, appuyez sur ENTRÉE. Le programme devrait se terminer et la fenêtre sortie doit afficher la sortie semblable à la suivante

```
    Debug Information-Product Starting
    The product name is Widget
    The available units on hand are100
    The per unit cost is 1.03
    System.Xml.XmlDocument
    Field: The product name is Widget
    Field: The units on hand are100
    Field: The per unit cost is1.03
    Calc: Total Cost is  103
    This message WILL appear
    ---- DEBUG ASSERTION FAILED ----
---- Assert Short Message ----
Message will appear since dUnitcost  < 1 is false
---- Assert Long Message ----


    at Class1.Main(String[] args)  <%Path%>\class1.cs(34)

    The product name is Widget
    The available units on hand are100
    The per unit cost is 1.03
Debug Information-Product Ending
Trace Information-Product Starting
    The product name is Widget
    Field: The product name isWidget
    This message WILL appear
Trace Information-Product Ending
```

9. La fenêtre de la Console et le fichier Output.txt doivent afficher le résultat suivant :

```
The product name is Widget
    The available units on hand are 100
    The per unit cost is 1.03
Debug Information-Product Ending
Trace Information-Product Starting
    The product name is Widget
    Field: The product name is Widget
    This message WILL appear
Trace Information-Product Ending
```

Remarque Le fichier Output.txt se trouve dans le même répertoire que l'exécutable conInfo (conInfo.exe). En général, il s'agit du dossier \bin, où se trouve la source du projet. Par défaut, il s'agit de C:\Documents and Settings**connexion utilisateur**\My Documents\Visual Studio Projects\conInfo\bin. Dans Visual C# 2005 et Visual C# 2005 Express Edition, le fichier Output.txt se trouve dans le dossier suivant :

C:\Documents and Settings**connexion utilisateur**\My Documents\Visual Studio 2005\Projects\conInfo\conInfo\bin\Debug

Retour au début

Listing complet du Code

```
using System;
using System.Diagnostics;

class Class1
{
   [STAThread]
   static void Main(string[] args)
   {
      string sProdName = "Widget";
      int iUnitQty = 100;
      double  dUnitCost = 1.03;
      Debug.WriteLine("Debug Information-Product Starting
      Debug.Indent();
      Debug.WriteLine("The product name is "+sProdName);
      Debug.WriteLine("The available units on hand are"+iU
      Debug.WriteLine("The per unit cost is "+ dUnitCost.T

      System.Xml.XmlDocument oxml = new System.Xml.XmlDocu
      Debug.WriteLine(oxml);

      Debug.WriteLine("The product name is "+sProdName,"Fi
      Debug.WriteLine("The units on hand are"+iUnitQty,"Fi
      Debug.WriteLine("The per unit cost is"+dUnitCost.ToS
      Debug.WriteLine("Total Cost is  "+(iUnitQty * dUnitC

      Debug.WriteLineIf(iUnitQty > 50, "This message WILL
      Debug.WriteLineIf(iUnitQty < 50, "This message will

      Debug.Assert(dUnitCost > 1, "Message will NOT appear
      Debug.Assert(dUnitCost < 1, "Message will appear sin

      TextWriterTraceListener tr1 = new TextWriterTraceLis
      Debug.Listeners.Add(tr1);

      TextWriterTraceListener tr2 = new TextWriterTraceLis
      Debug.Listeners.Add(tr2);


      Debug.WriteLine("The product name is "+sProdName);
      Debug.WriteLine("The available units on hand are"+iU
      Debug.WriteLine("The per unit cost is "+dUnitCost);
      Debug.Unindent();
      Debug.WriteLine("Debug Information-Product Ending");
      Debug.Flush();

      Trace.WriteLine("Trace Information-Product Starting
      Trace.Indent();
```

```
        Trace.WriteLine("The product name is "+sProdName);
        Trace.WriteLine("The product name is"+sProdName,"Fie
        Trace.WriteLineIf(iUnitQty > 50, "This message WILL
        Trace.Assert(dUnitCost > 1, "Message will NOT appear


        Trace.Unindent();
        Trace.WriteLine("Trace Information-Product Ending");

        Trace.Flush();

        Console.ReadLine();
    }
}
```

Retour au début

Résoudre les problèmes

- Si le type de configuration de solution est Release, la classe Debug de sortie est ignorée.
- Après avoir créé une classe d'écouteur TextWriterTraceListener pour une cible particulière, l'écouteur TextWriterTraceListener reçoit la sortie de la Trace et les classes Debug . Dans ce cas que vous utilisiez la méthode Add de la Trace ou la classe Debug pour ajouter l'écouteur TextWriterTraceListener à la classe d'écouteurs .
- Si vous ajoutez un objet écouteurs pour la même cible dans la Trace et les classes Debug , chaque ligne de sortie est en double, quel que soit
  Debug ou Trace génère la sortie.

```
TextWriterTraceListener myWriter = new Textl
Debug.Listeners.Add(myWriter);

TextWriterTraceListener myCreator = new Text
Trace.Listeners.Add(myCreator);
```

Retour au début

Références

Pour plus d'informations, consultez les rubriques suivantes dans la documentation de la bibliothèque de classes.NET Framework :

Trace (classe)
http://msdn2.microsoft.com/en-us/library/system.diagnostics.trace(vs.71).aspx

Classe de débogage
http://msdn2.microsoft.com/en-us/library/system.diagnostics.debug(vs.71).aspx

Retour au début

Dernière mise à jour : 19 avr. 2018