Visual Studio .NET 2003 et VBCommenter
Utiliser la documentation automatique sous VB.NET

Le principe des commentaires automatiques apparu avec C# avec Visual Studio.NET n'existe pas de base en VB.NET.
Il faut donc utiliser un outil développé par l'équipe de **GotDotNet (US)**. Nous allons donc voir comment utiliser cet outil et donc générer une documentation standard facilement.

Introduction

Lorsque l'on veut développer une application, il est rapidement obligatoire de commenter son code. Ceci pour plusieurs raisons :

- Permettre facilement le travail en équipe
- Permettre de se replonger dans son code rapidement même plusieurs mois après
- Permettre une lecture aisée du code par toute autre personne
- ...

Comme me disait un professeur, n'importe qui ne connaissant pas forcément le langage que vous utilisez doit pouvoir ouvrir n'importe quelle fichier source et comprendre rapidement le but de chaque fonction. Ceci ne peut donc se faire que par l'utilisation 'abusive' des commentaires.

Le but de cet article est de vous présenter l'outil **VBCommenter** qui va vous permettre de générer (avec **NDoc**) les fichiers d'aide que vous pourrez conserver ou redistribuer avec votre projet.

Présentation

VBCommenter est développé par l'équipe de **GotDotNet** dans le but de permettre aux développeurs VB.NET utilisant Visual Studio.NET 2003 de travailler avec NDoc (de la même façon que les développeur C#).

En effet, le principe des commentaires automatiques a été ajouté en natif dans C# avec les **///**, mais n'existait pas sous VB.NET.

Ainsi dans tout projet C#, on retrouve au dessus de chaque classe et de chaque fonction les blocs ci-dessous :

```
/// <summary>
/// Description résumée de ClassTest.
/// </summary>
public class ClassTest
{
....


 /// <summary>
 /// Commentaire sur Mafonction.
 /// </summary>
 /// <param name="e"></param>

 public MaFonction(int e)
 {
 ...
 }
...
}
```

Le but est donc de pouvoir avoir la même chose avec VB.NET. Voyons maintenant comment faire ceci.

Installation et Utilisation de VBCommenter

Téléchargement

Il faut avant tout télécharger le fichier d'installation depuis le site officiel :

- **Télécharger VBCommenter pour Visual Studio .NET 2003**

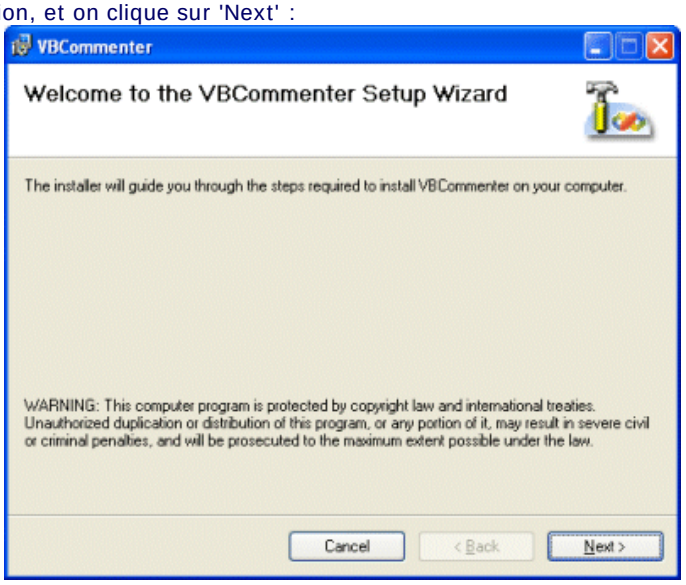
Il faut prendre dans la liste la dernière version proposée (au moment de l'écriture de cet article il s'agissait de la version 1.1.1).

Installation

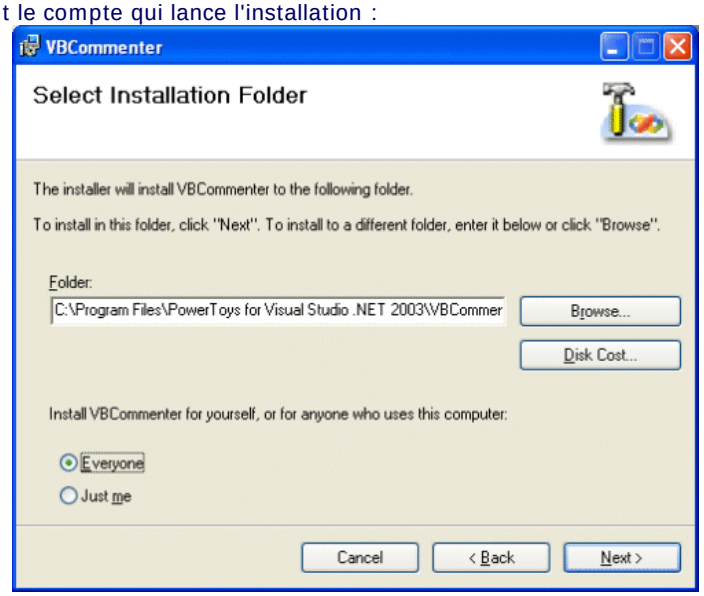
Vous obtenez alors un fichier ZIP qu'il faut décompresser. Dans ce zip, il y a le fichier MSI d'installation du produit (VBCommenterSetup.msi) qu'il faut exécuter.

Attention : il est préférable de ne pas avoir de session de Visual Studio.NET 2003 lancées au moment de l'installation.

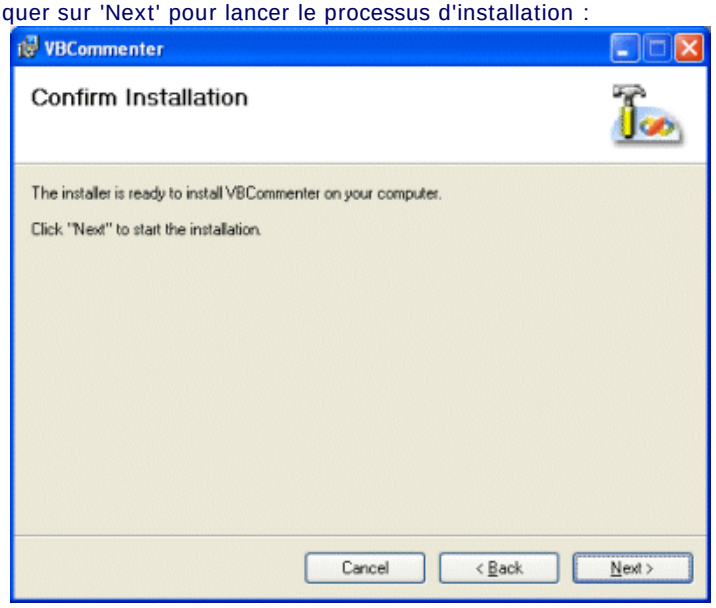
On lance alors l'installation, et on clique sur 'Next' :



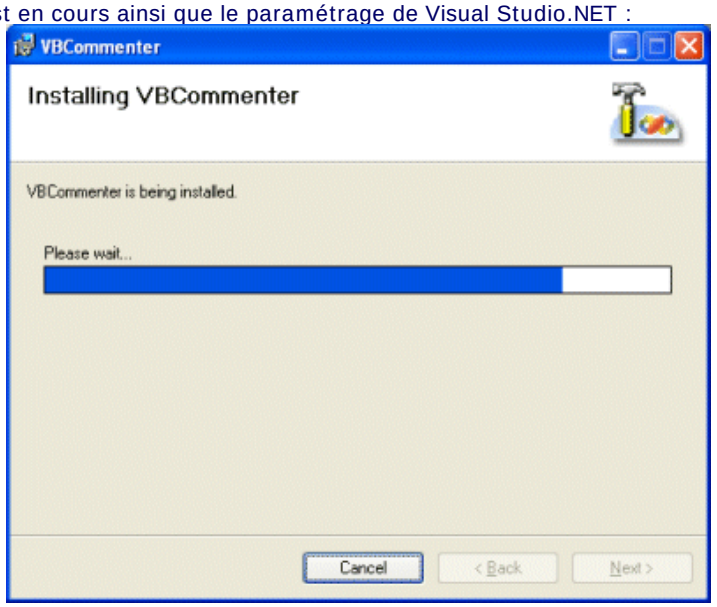
Maintenant on doit définir le répertoire d'installation de l'application et si on autorise à tout le monde de l'exécuter ou seulement le compte qui lance l'installation :



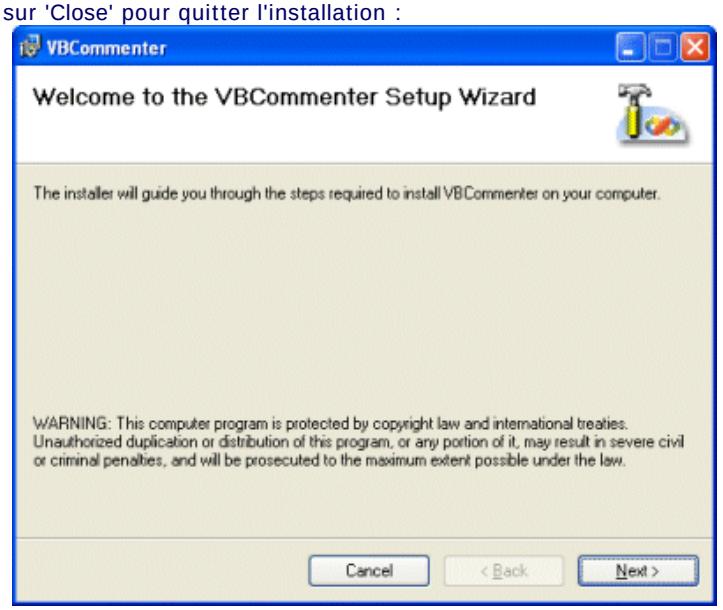
Il nous demande de cliquer sur 'Next' pour lancer le processus d'installation :



La copie des fichiers est en cours ainsi que le paramétrage de Visual Studio.NET :

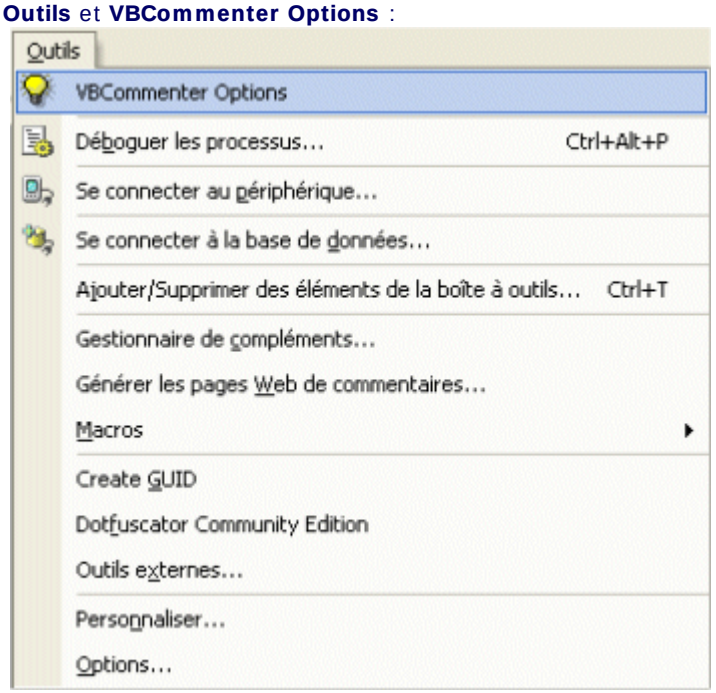


On a plus qu'à cliquer sur 'Close' pour quitter l'installation :

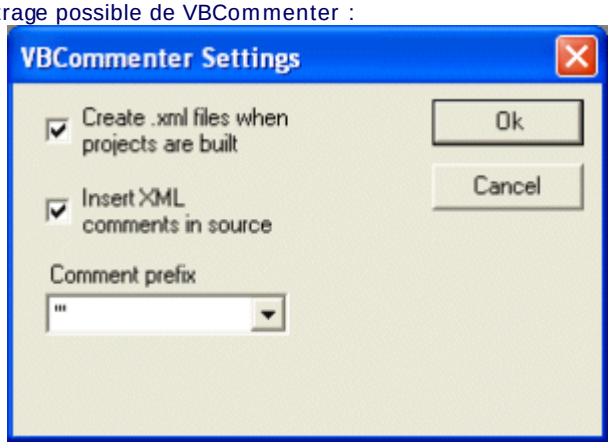


Utilisation

A ce stade VBCommenter est installé sur la machine. Pour s'en convaincre, il suffit de lancer Visual Studio.NET 2003 et de cliquer sur **Outils** et **VBCommenter Options** :



On obtient alors le paramétrage possible de VBCommenter :



Les paramétrages de base permettent d'utiliser le ''' comme préfixe des blocs de VBCommenter, de générer un fichier XML au même niveau que le fichier binaire lors de la compilation de votre projet et enfin d'insérer les bloc XML de commentaire à chaque fois que vous taperez ''' (ou un autre préfixe choisi) au dessus de votre classe, variable partagée, fonction ou procédure.

Exemple de Classe commentée

Dans le cas d'une classe on aura donc :

```
''' ------------------------------------------------------------------------
''' Project    : MonProjetTest
''' Class    : MaClasse
'''
''' ------------------------------------------------------------------------
''' <summary>
''' Classe de Test pour article VBCommenter
''' </summary>
''' <remarks>
''' </remarks>
''' <history>
'''    [fabrice69]   23/06/2004   Created
''' </history>
''' ------------------------------------------------------------------------
Public Class MaClasse

    ''' ------------------------------------------------------------------------
    ''' <summary>
    ''' Variable Partagée commentée
    ''' </summary>
    ''' <remarks>
    ''' </remarks>
    ''' <history>
    '''    [moi]   22/06/2004   Created
    ''' </history>
    ''' ------------------------------------------------------------------------
    Public Shared ReadOnly MaVariablePartagee As String = "VS.NET 2003 & VBCommenter"

     ''' ------------------------------------------------------------------------
    ''' <summary>
    ''' Commentaire de MaFonction
    ''' </summary>
    ''' <param name="Val"></param>
    ''' <returns></returns>
    ''' <remarks>
    ''' </remarks>
    ''' <history>
    '''    [moi]   23/06/2004   Created
    ''' </history>
    ''' ------------------------------------------------------------------------
    Public Function MaFonction(ByRef Val As Integer) As String
        Return "Visual Studio.NET 2003 et VBCommenter = Que du Bonheur"
    End Function

    ''' ------------------------------------------------------------------------
    ''' <summary>
    ''' Commentaire de MaProcedure
    ''' </summary>
    ''' <param name="Val"></param>
    ''' <remarks>
    ''' </remarks>
    ''' <history>
    '''    [moi]   23/06/2004   Created
    ''' </history>
    ''' ------------------------------------------------------------------------
    Public Sub MaProcedure(ByVal Val As Integer)
        HttpContext.Current.Response.Write("Visual Studio.NET 2003 et VBCommenter = Que du Bonheur")
    End Sub

    ' ------------------------------------------------------------------------
End Class
```

Maintenant que nous avons VBCommenter installé et que la compilation de notre projet nous a bien fourni le fichier XML de commentaire, voyons avec quoi et comment exploiter ce fichier.

Installation de NDoc

Téléchargement

Il faut avant tout télécharger le fichier d'installation depuis le site officiel :

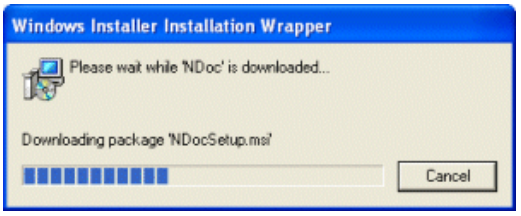
- **Télécharger NDoc**

La dernière version release au moment de l'écriture de cet article est la version 1.2 (cela peut changer à l'avenir).

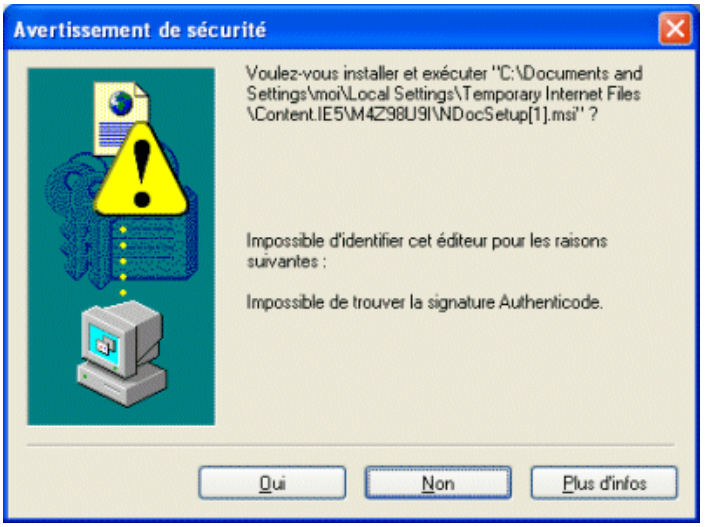
Installation

L'installation est similaire à celle de VBCommenter, on doit juste lancer l'exécutable téléchargé.

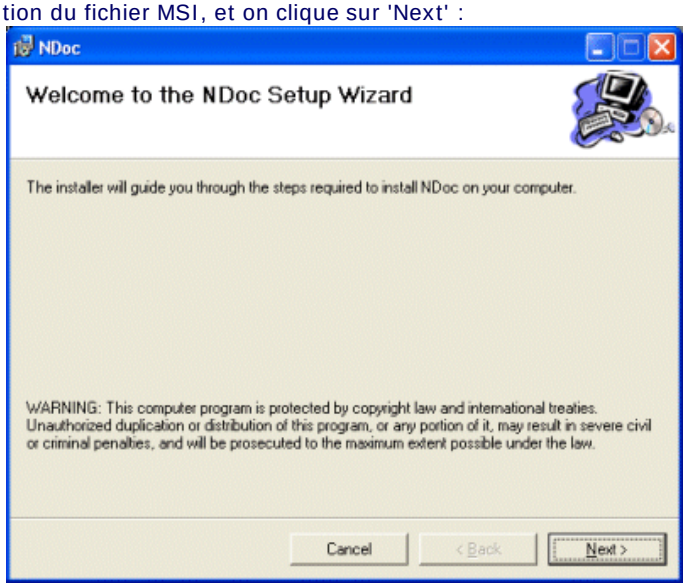
Il va décompresser automatiquement le fichier MSI qu'il contient :



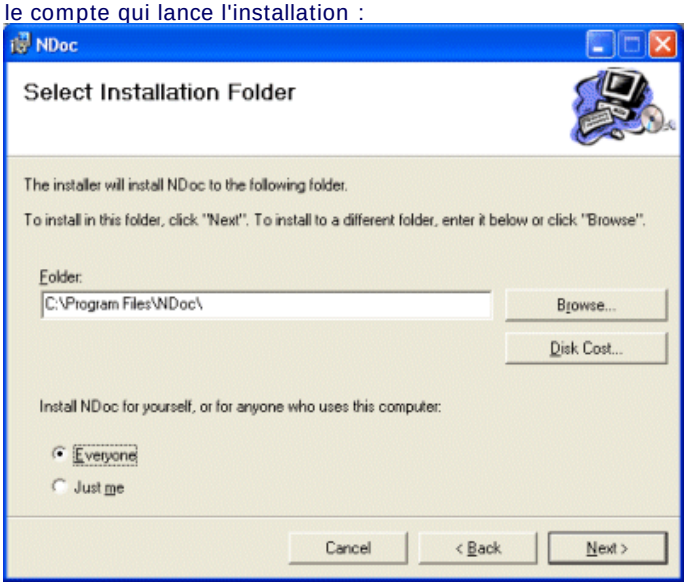
A ce moment une alerte envoyée par Windows, signale que le fichier MSI n'est pas signé, vous n'en tenez pas compte et cliquez sur OUI :



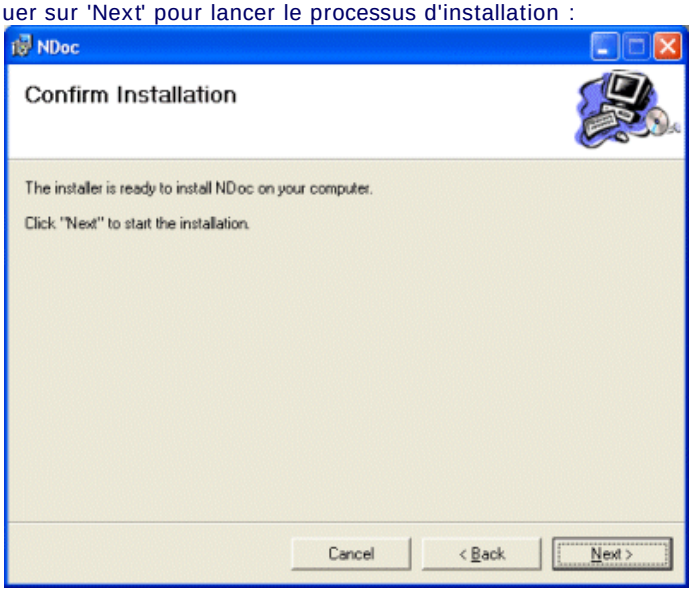
On arrive donc à l'exécution du fichier MSI, et on clique sur 'Next' :



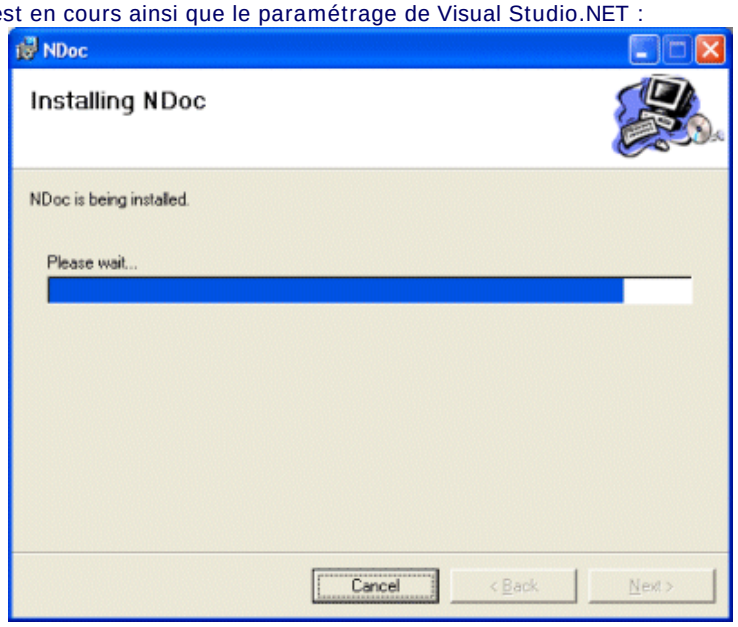
Maintenant on doit définir le répertoire d'installation de l'application et si on autorise à tout le monde de l'exécuter ou seulement le compte qui lance l'installation :



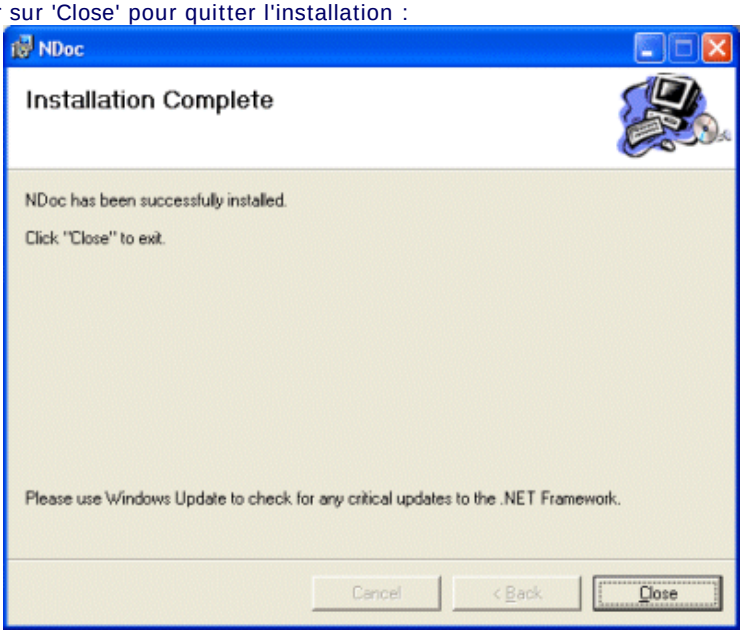
Il nous demande de cliquer sur 'Next' pour lancer le processus d'installation :



La copie des fichiers est en cours ainsi que le paramétrage de Visual Studio.NET :



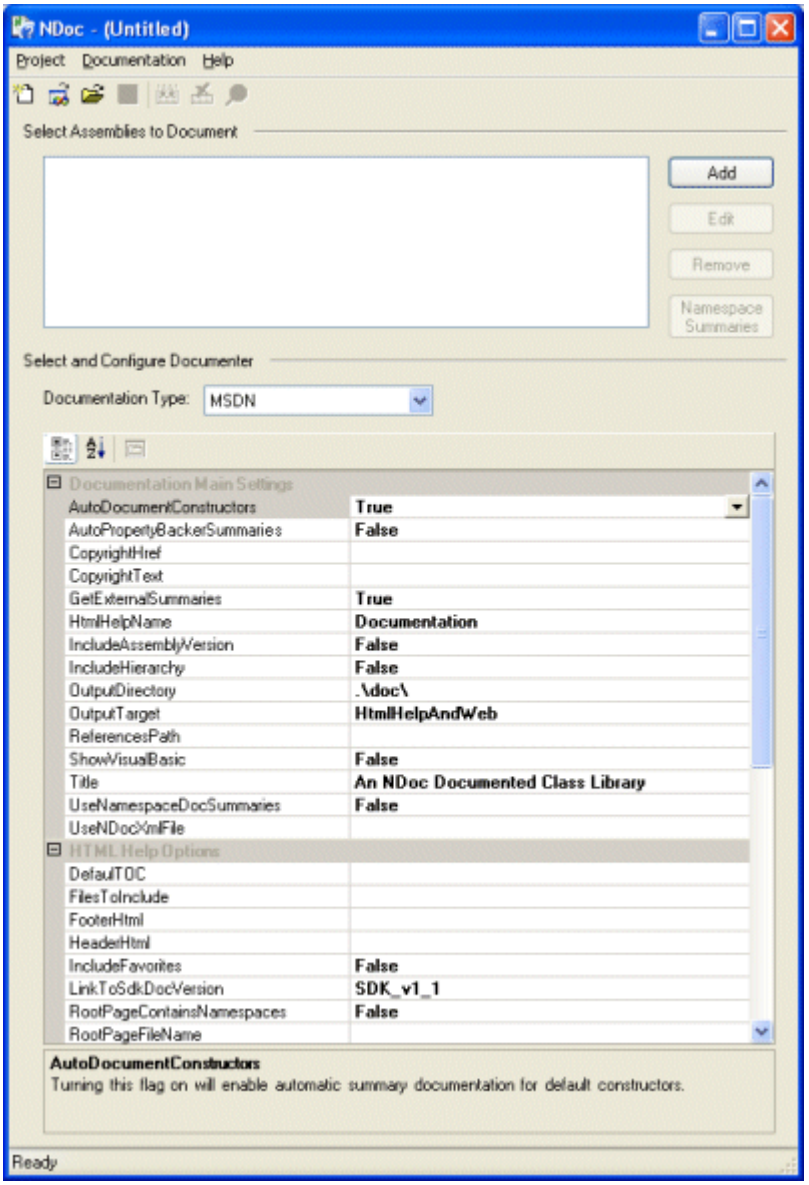
On a plus qu'à cliquer sur 'Close' pour quitter l'installation :



Utilisation

Maintenant NDoc est installé sur la machine, on peut lancer celui-ci en cliquant sur :

- **Demarrer > Programme > NDoc**

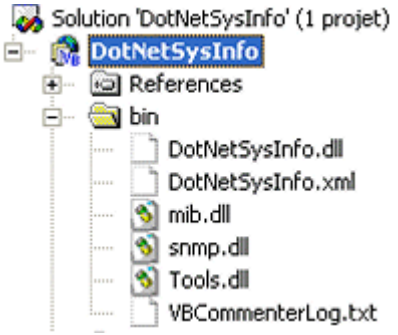


Voyons donc comment nous servir de cet outil sur une solution simple ne contenant qu'un projet :

- **Projet DotNetSysInfo - Sur DotNet Project (FR)**

Utilisation de NDoc

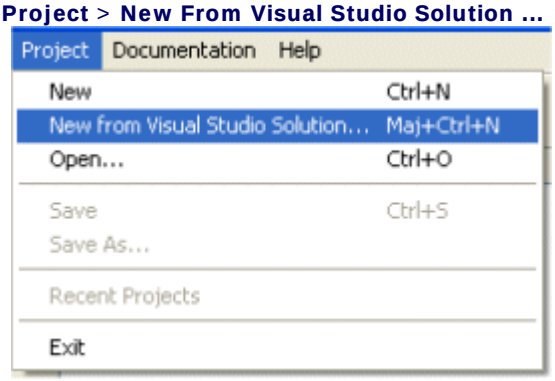
Lors de la compilation de notre Projet, nous avons obtenu le fichier XML de commentaires dans le répertoire où se trouvent les fichiers binaires du projet.



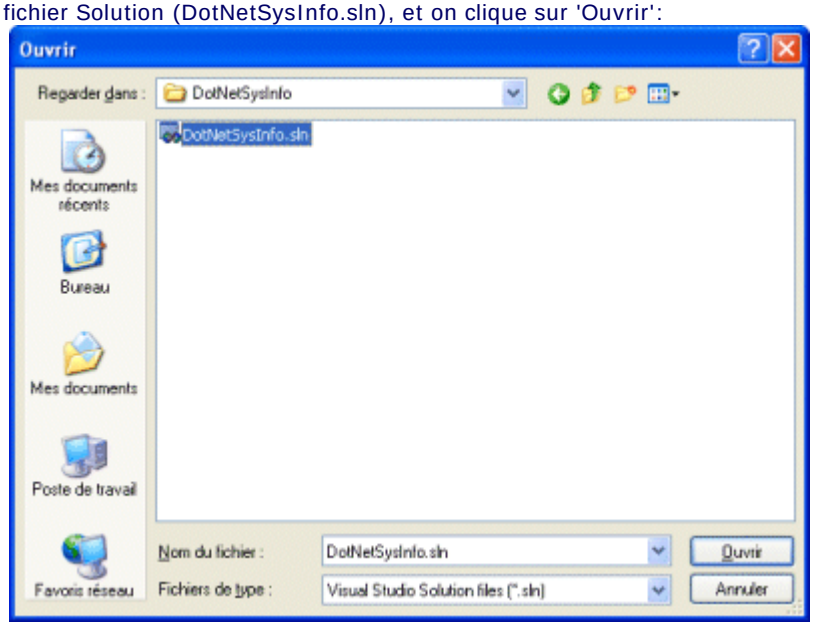
Ce fichier XML est accompagné d'un fichier de Log (VBCommenterLog.txt) qui liste l'ensemble des fichiers scannés où se trouvaient les balises et préfixes recherchés.

On va donc dans NDoc référencer le fichier SLN de notre solution Visual Studio.NET et associer la DLL avec le fichier XML. Voyons comment faire.

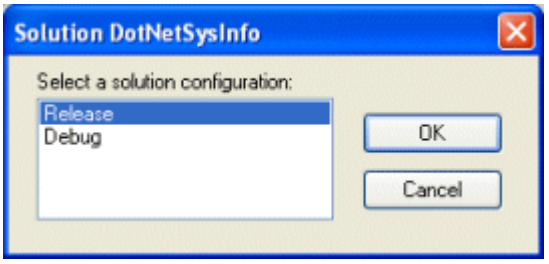
Dans NDoc, il faut cliquer sur **Project** > **New From Visual Studio Solution ...** :



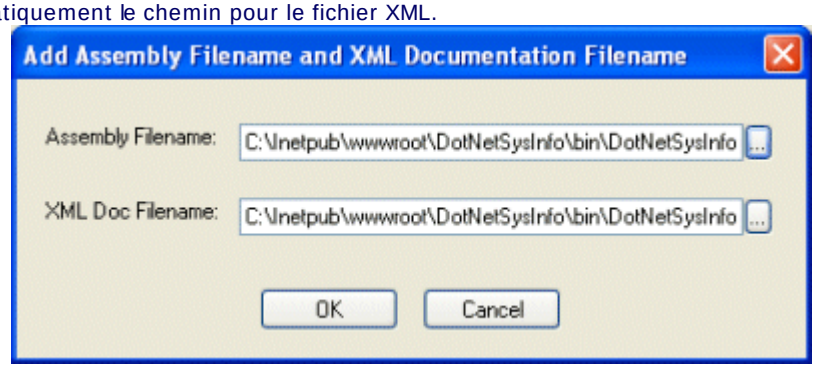
On va chercher le fichier Solution (DotNetSysInfo.sln), et on clique sur 'Ouvrir':



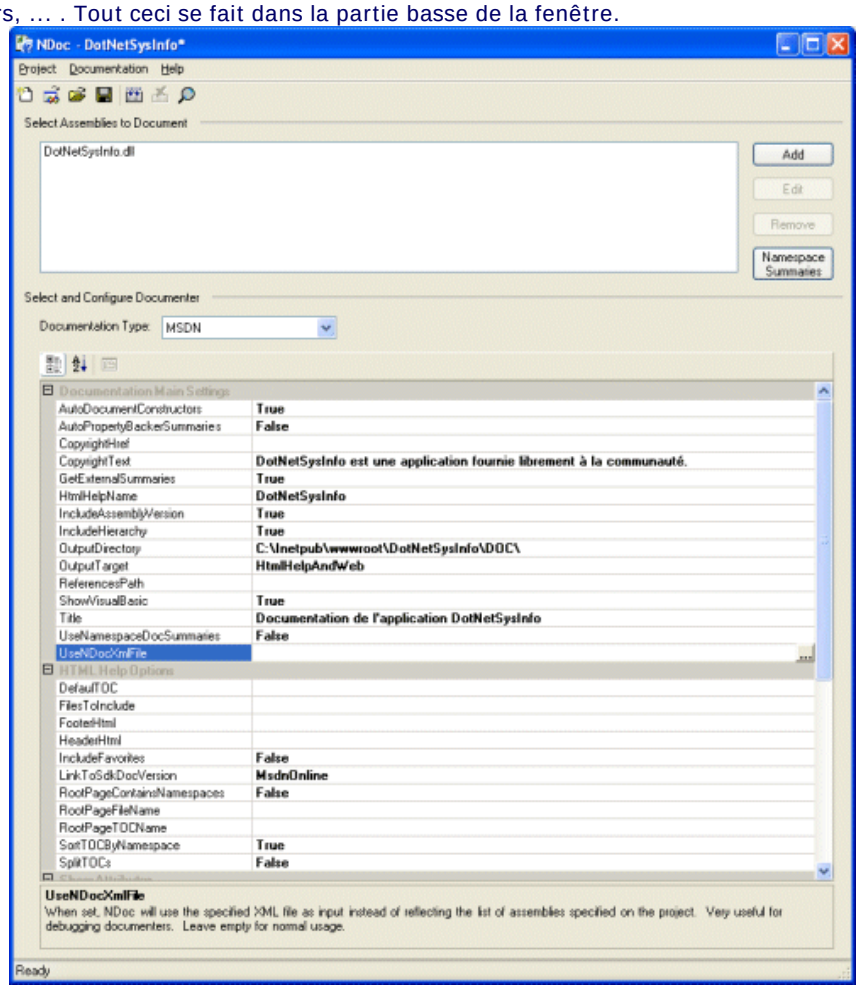
A ce moment une boite de dialogue se présente nous demandant de choisir entre la version Release ou Debug, dans notre cas, nous prendrons la version Release :



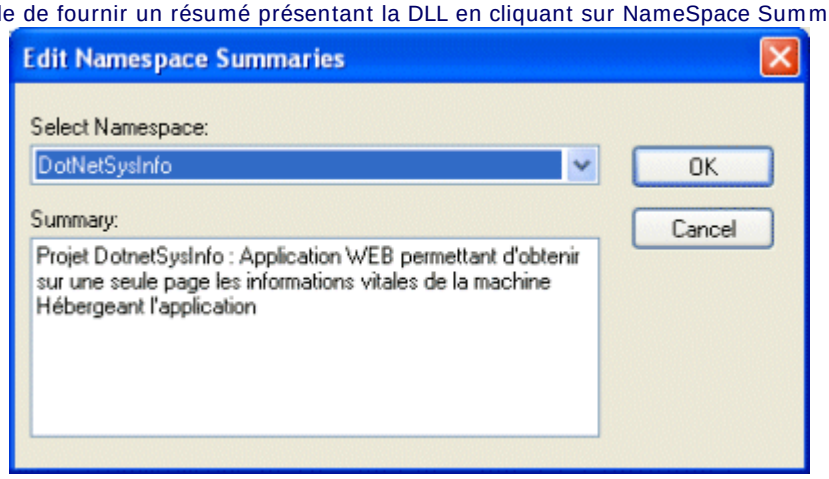
A ce stade il ne nous reste plus qu'à définir la DLL associée au fichier XML que nous voulons documenter. Ceci se fait en cliquant sur **ADD** et allant chercher le fichier DLL que nous voulons documenter, normalement il renseigne automatiquement le chemin pour le fichier XML.



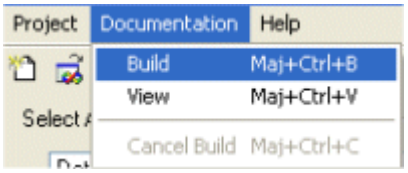
Nous pouvons aussi à ce niveau donner un nom à notre fichier, définir un titre, définir le répertoire où vont se situer les fichiers, ... Tout ceci se fait dans la partie basse de la fenêtre.



Il est aussi possible de fournir un résumé présentant la DLL en cliquant sur NameSpace Summary :



Une fois tous les renseignements complétés, nous pouvons générer la documentation, en cliquant sur **Documentation** > **Build** :



On obtient alors nos fichiers de documentations dans le répertoire choisi dans l'écran. Je vous conseille d'ailleurs de sauvegarder le fichier NDoc, car il vous permettra de remettre à jour très rapidement la documentation lors des Update de votre code.

Deux options me semblent aussi importantes dans la configuration de la génération :

- L'affichage de la version de l'Assembly dans la Doc, ce qui vous permet de savoir où vous en êtes
- Le choix du type de documentation (Web ou HtmlHelp)

Pour les autres options, je vous invite à consulter la Documentation sur le site officiel du produit.

Conclusion

Le fait de commenter son code n'est pas un luxe que le développeur peut s'offrir que lorsqu'il a un peu de temps libre (de toutes les manières il n'en a jamais), mais bien une obligation. Il faut absolument se forcer à commenter et donc ensuite à documenter son code afin de permettre une meilleure approche de son travail par les personnes extérieures. Il n'est pas non plus obligatoire de mettre un roman dans chaque commentaire, mais simplement une petite phrase présentant rapidement le but de la fonction ou de la classe.

La différence entre un développeur moyen et un bon développeur ne se résume pas seulement à la qualité de son code, mais aussi au fait de montrer aux autres son travail, et comment celui-ci a été fait.

En conclusion, il faut allier :

- Le **SAVOIR FAIRE** et le **FAIRE SAVOIR**

J'espère que cet article vous aura servi. Voila quelques articles ou exemples sur ce sujet :

- **Créer une documentation avec Ndoc (FR)**
- **Commenter et documenter son code C# avec Visual Studio .NET et XML (FR)**
- **La documentation du code en C# (FR)**
- **Liste des PowerToys disponibles pour Visual Studio .NET (US)**
- **Site Officiel de GotDotNet (US)**
- **Site Officiel de VBCommenter (US)**
- **NDoc References (US)**

En vous souhaitant de bons projets de développement.

Romelard Fabrice (alias F___)