Génération de documentation avec Sandcastle Help File Builder

par Philippe Vialatte (ma page DVP)

Date de publication : 02/07/2008

Dernière mise à jour : 05/08/2008

Dans ce tutoriel, nous allons voir comment générer simplement une documentation pour notre code, en se basant sur les commentaires de notre code, et sur Sandcastle Help File Builder.
De plus, nous allons voir comment intégrer cette génération à notre cycle d'intégration continue.

I - Introduction

Comment faire pour bien gérer la documentation du code ?

Dans un de mes livres favoris (the pragmatic programmer, lisez-le, j'insiste ;)), une des applications du principe DRY (don't repeat yourself) est de ne pas séparer la documentation technique du code.

En effet, si la documentation doit être mise à jour en dehors de l'environnement contenant le code source, non seulement nout répètons l'effort de documentation en mettant à jour la documentation dans le code et la documentation externe au code, mais en plus, nous allons vite voir apparaître des écarts entre les deux jeux de documentation. Historiquement, en .net, NDoc était l'outil de choix pour générer sa documentation à partir des commentaires. Malheureusement, NDoc ne supporte pas entièrement la version 2.0 (ne parlons même pas de la 3.5), et n'est plus actif depuis 2006.

Heureusement (car tout ne peut pas être toujours tout noir...), Microsoft travaille depuis quelques années (Juillet 2006 pour la première CTP) sur un équivalent à NDoc...Sandcastle

II - Comment documenter mon code ?

Plutôt que de réécrire un article complet sur ce sujet (ce que je comptais originalement faire), je vais, dans un excès de fainéantise, vous rediriger vers ce très bon article, qui, malgré sa date de publication, reste entièrement valide.

Bien commenter et documenter son code en C#
De façon générale, nous admettrons que dans le tag de commentaire summary de nos fonctions, nous trouverons les explications fonctionnelles de notre code, ainsi que les points importants de nos algorithmes.

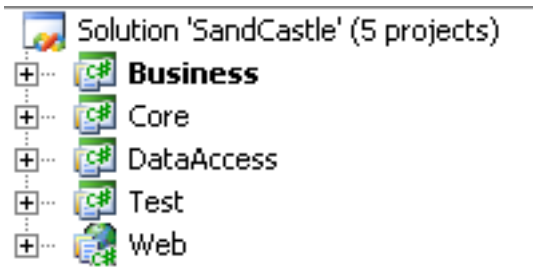
III - Utilisation de Sandcastle Help File Builder

Présentation du projet

Sandcastle Help File Builder est une interface utilisateur permettant de gérer la plomberie interne de Sandcastle (relativement lourde à mettre en #uvre) facilement, et surtout, rapidement. Dans notre exemple, nous allons prendre un projet assez standard, web dans ce cas.

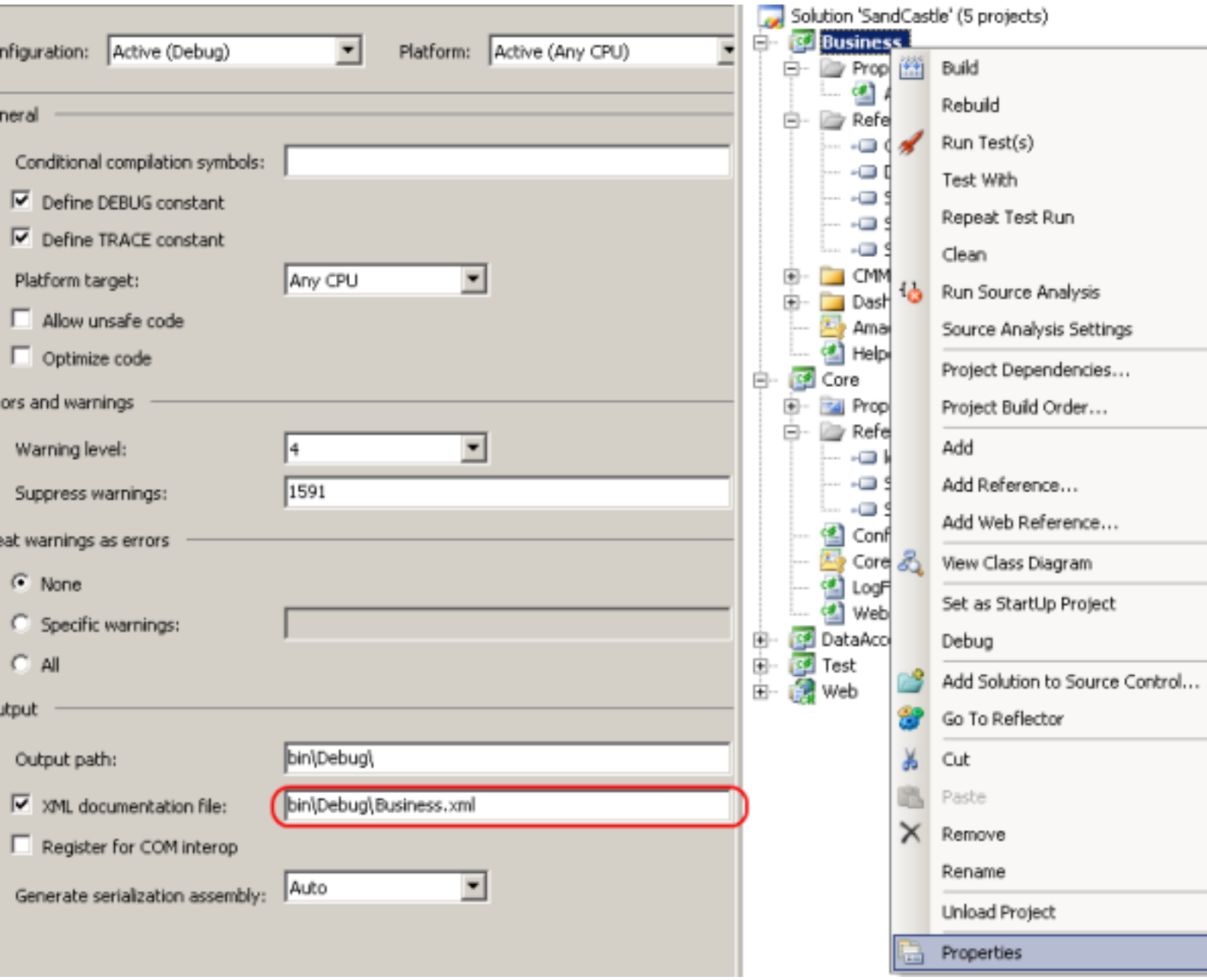
Si vous voulez générer une documentation pour un site web, le meilleur moyen sera de faire un site de type "application web". En effet, dans le cas d'un site web "standard", il n'est pas possible de générer de documentation XML sans ajouter un certain nombre de tags dans le Web.Config, et je préfère limiter les modifications de mon Web.Config à la configuration des outils.

Notre projet aura donc la structure suivante :



A noter, le projet Core possède une dépendance sur Log4Net, ce qui aura son importance plus tard...

Avant tout, pour chaque projet, je vais devoir vérifier que la génération de documentation XML est bien sélectionnée. Pour cela, je vais aller dans les propriétés de mon projet, cocher "Fichier de documentation XML".



Cocher ceci sur chaque projet me permets donc de produire un fichier xml contenant toute ma documentation. Pour les besoins de l'article, nous allons surtout nous concentrer sur le projet Core (qui a l'intérêt d'avoir peu de classes). Le code suivant:

version C#

```
using System;
using log4net;

namespace Core {
    /// <summary>
    /// Summary description for LogFactory
    /// </summary>
    public static class LogFactory {
        private static ILog log = LogManager.GetLogger(typeof(LogFactory));
```

version C#

```
        /// <summary>
        /// Logs a debug message
        /// </summary>
        /// <param name="message">The message.</param>
        public static void LogDebug(string message) {
            log.Debug(message);
        }

        /// <summary>
        /// Logs an informational message.
        /// </summary>
        /// <param name="message">The message.</param>
        public static void LogInfo(string message) {
            log.Info(message);
        }

        /// <summary>
        /// Logs an error message, with the corresponding exception.
        /// </summary>
        /// <param name="message">The message.</param>
        /// <param name="ex">The ex.</param>
        public static void LogError(string message, Exception ex) {
            log.Error(message, ex);
        }
    }
}
```

version VB.Net

```
Imports System
Imports log4net

Namespace Core
    ''' <summary>
    ''' Summary description for LogFactory
    ''' </summary>
    public shared class LogFactory
        private shared log As ILog = LogManager.GetLogger(typeof(LogFactory))

        ''' <summary>
        ''' Logs a debug message
        ''' </summary>
        ''' <param name="message">The message.</param>
        public shared Sub LogDebug(string message) {
            log.Debug(message);
        }

        ''' <summary>
        ''' Logs an informational message.
        ''' </summary>
        ''' <param name="message">The message.</param>
        public shared Sub  LogInfo(string message) {
            log.Info(message);
        }

        ''' <summary>
        ''' Logs an error message, with the corresponding exception.
        ''' </summary>
        ''' <param name="message">The message.</param>
        ''' <param name="ex">The ex.</param>
        public shared Sub LogError(string message, Exception ex) {
            log.Error(message, ex);
        }
    End Class
End NameSpace
```

Générera, dans les deux cas, le fichier de commentaire suivant:

```
<?xml version="1.0"?>
```

```
<doc>
    <assembly>
        <name>Core</name>
    </assembly>
    <members>
        <member name="T:Core.WebServiceException">
            <summary>
            Exception used to track the web service issues
            </summary>
        </member>
        <member name="M:Core.WebServiceException.#ctor">
            <summary>
            Initializes a new instance of the <see cref="T:Core.WebServiceException"/> class.
            </summary>
        </member>
        <member name="M:Core.WebServiceException.#ctor(System.String)">
            <summary>
            Initializes a new instance of the <see cref="T:Core.WebServiceException"/> class.
            </summary>
            <param name="message">The message.</param>
        </member>
        <member name="T:Core.ConfigurationWrapper">
            <summary>
            Summary description for ConfigurationWrapper
            </summary>
        </member>
        <member name="T:LogFactory">
            <summary>
            Summary description for LogFactory
            </summary>
        </member>
        <member name="M:LogFactory.LogDebug(System.String)">
            <summary>
            Logs a debug message
            </summary>
            <param name="message">The message.</param>
        </member>
        <member name="M:LogFactory.LogInfo(System.String)">
            <summary>
            Logs an informational message.
            </summary>
            <param name="message">The message.</param>
        </member>
        <member name="M:LogFactory.LogError(System.String,System.Exception)">
            <summary>
            Logs an error message, with the corresponding exception.
            </summary>
            <param name="message">The message.</param>
            <param name="ex">The ex.</param>
        </member>
    </members>
</doc>
```

Notre premier fichier d'aide

Dans un premier temps, nous allons commencer par télécharger et installer les pré-requis.

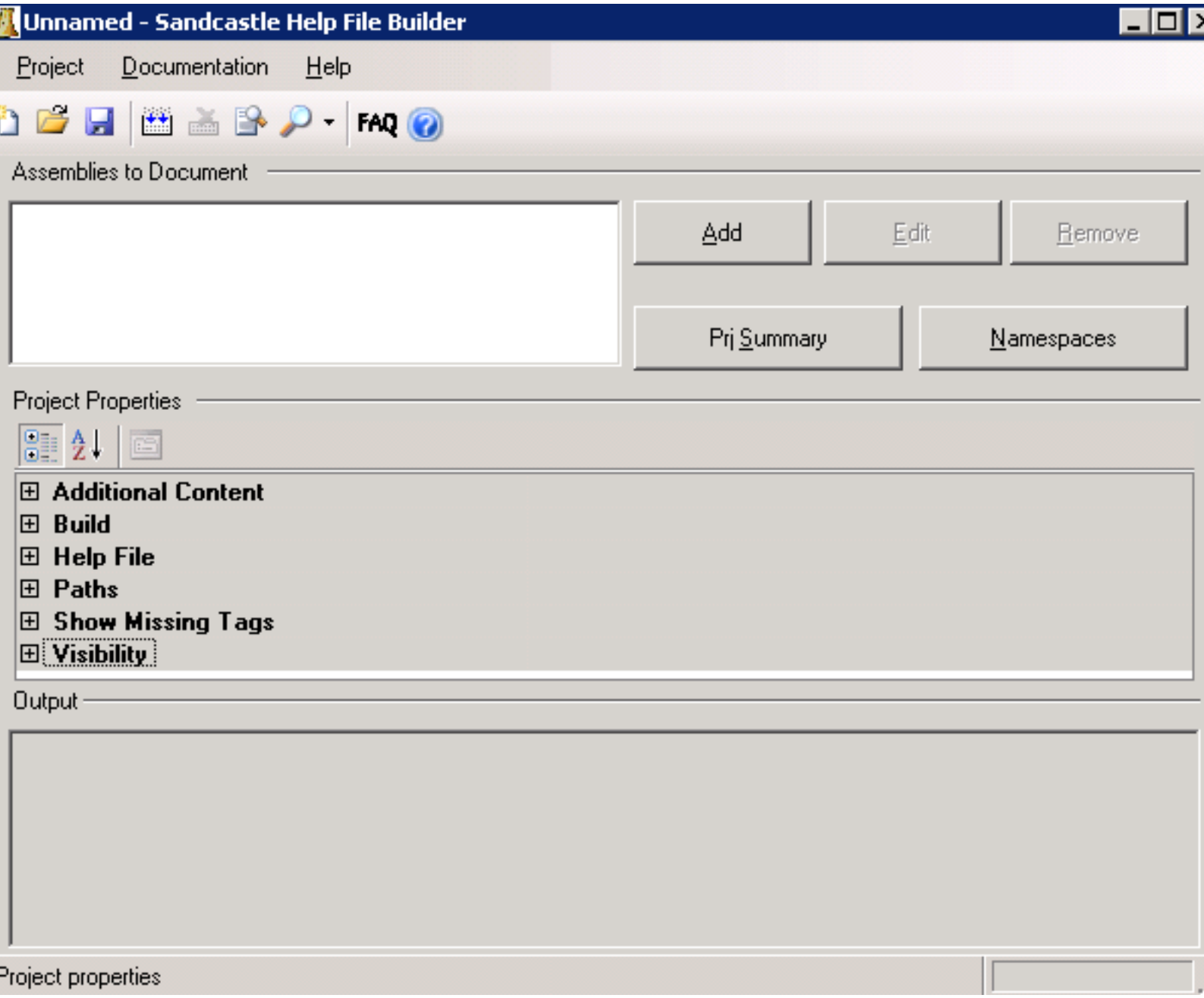
- L'installeur de SandCastle: Il se trouve sur le site de Microsoft
  SandCastle
- L'installeur de SandCastle Help File Builder : il se trouve directement sur le site codeplex du projet
  SandCastle Help File Builder
- HTML Help Workshop : Pour construire des fichiers d'aide au format 1.x
  HTML Help Workshop
- HTML Help 2.0 coProgi1984 mpiler : Pour construire des fichiers d'aide au format 2.0 -> dans un des SDK visual studio

Visual Studio 2005 SDK
Visual Studio 2005 SDK

Une fois tout ce petit monde installé, nous allons pouvoir (enfin, il était temps) passer à la construction de notre fichier d'aide.

*Durant la relecture de l'article, **Progi1984** m'a informé de la possibilité d'installer le "Visual Studio .NET Help Integration Kit" plutôt que d'installer tout le SDK.*
*Je n'ai pas pu tester cette manipulation, mais si vous voulez essayer de vous épargner les quelques 120 mo de téléchargement du SDK, allez voir **ici**.*

Sans plus attendre, je vais donc lancer SandCastle Help File Builder...

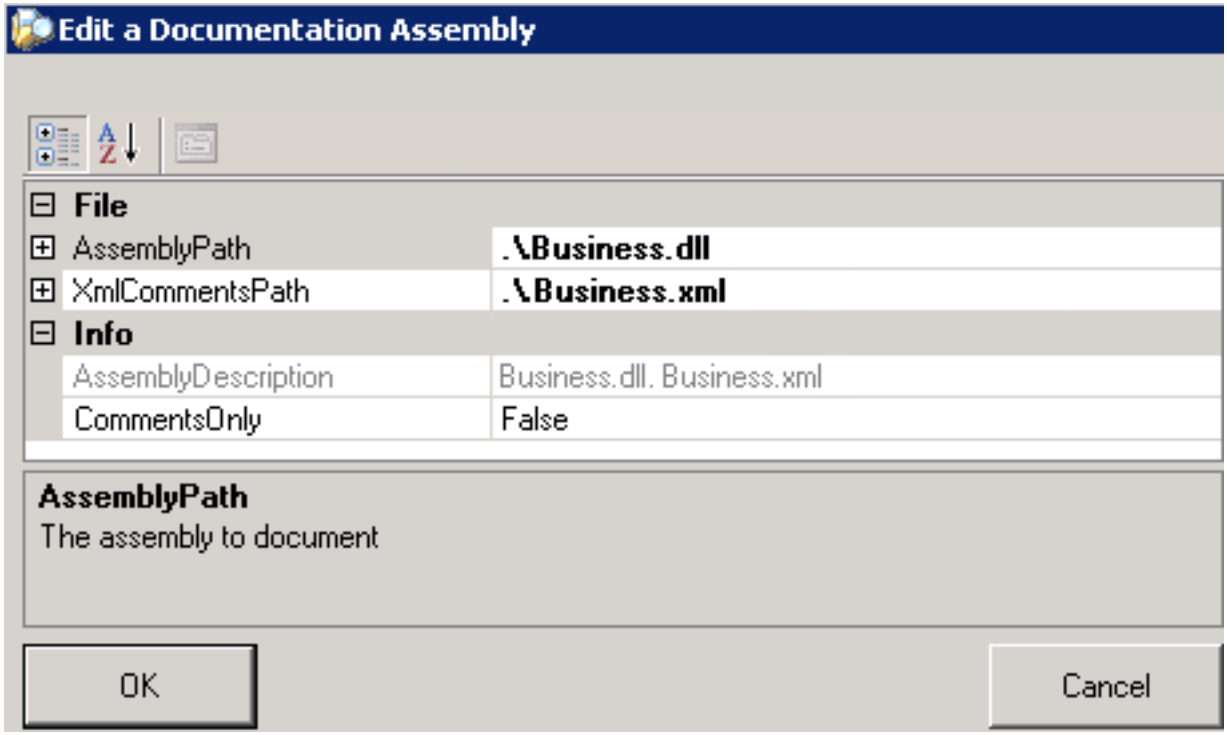


A première vue, nous voyons trois grandes zones

- Assemblies to document : Liste de tous les assemblages que nous allons inclure dans la documentation générée.
- Project Properties : Dans cette zone, nous allons trouver toutes les propriétés de notre projet.
- Output : Ici, nous trouverons tous les messages renvoyés par Sandcastle lors de la compilation de notre projet.

Ajout des assemblages

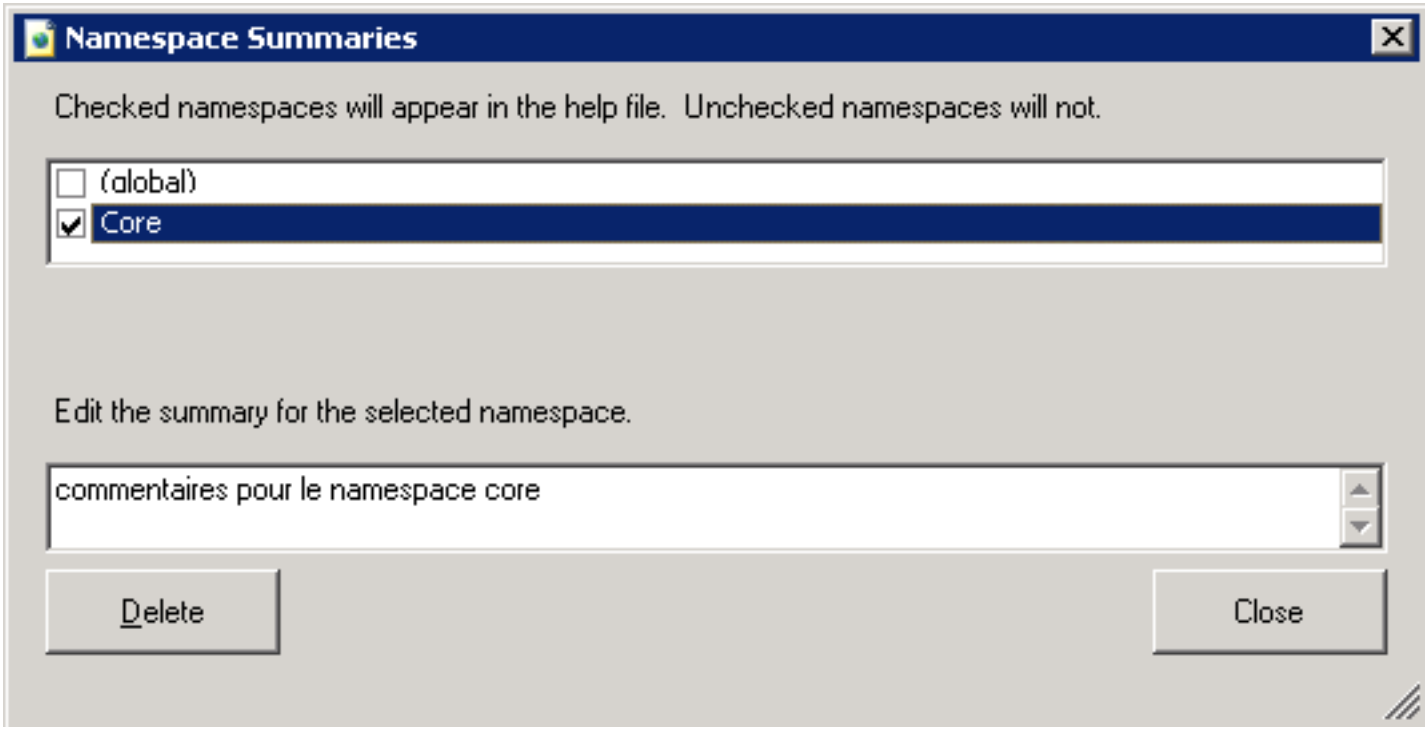
L'ajout d'un assemblage est assez simple, nous allons simplement cliquer sur Add, et sélectionner, dans le répertoire de compilation, à la fois le fichier dll et le fichier xml. Si nous ne sélectionnons qu'un seul des deux (ou si ils ont été générés dans des répertoires différents), il suffit de sélectionner l'assemblage, et cliquer sur Edit pour faire apparaître la fenêtre suivante:



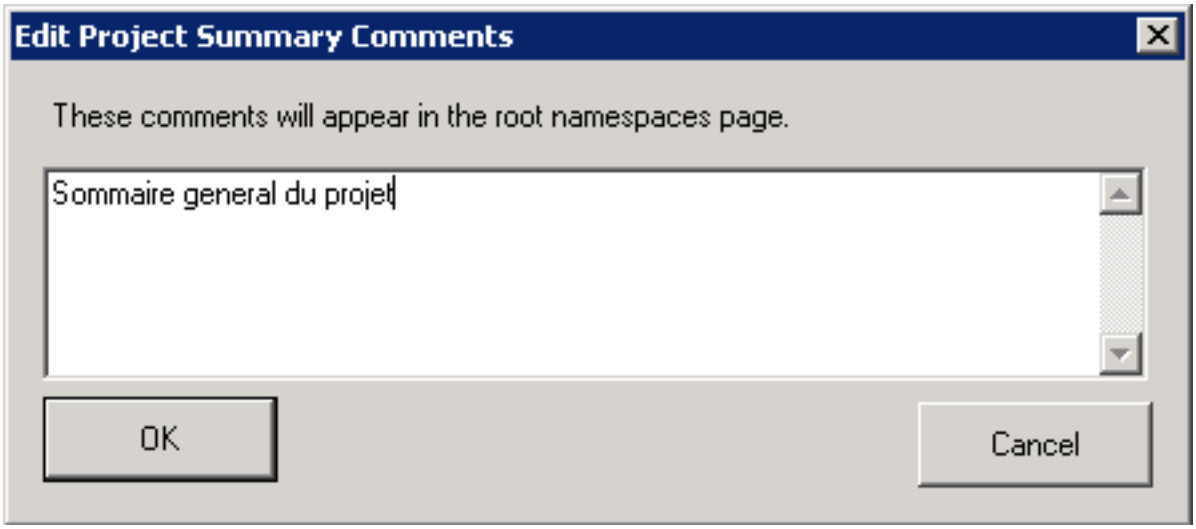
Nous ne nous intéresserons pas à l'option CommentsOnly dans ce tutoriel. Elle permet en fait principalement de partager entre plusieurs projets le sommaire et la liste des namespaces à documenter, dans le cas, par exemple, de deux jeux de documentation à générer, un en intranet documentant l'intégralité du code, l'autre externe ne documentant que les propriétés et méthodes publiques.

Sélection des namespaces et sommaire du projet

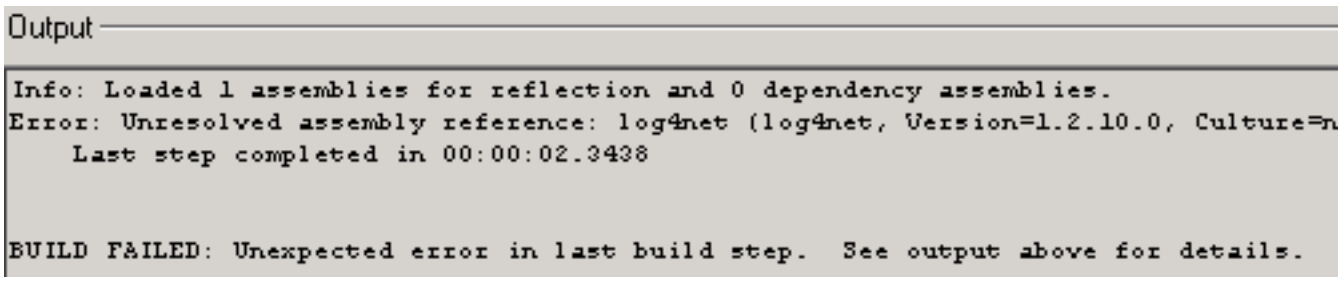
SandCastle nous permet de ne documenter qu'un sous-ensemble des espaces de nom de notre projet.
Pour sélectionner les espaces de nom en question, il suffit de cliquer sur le bouton Namespaces, et de cocher/décocher les espaces de noms désirés.



De plus, ce dialogue nous permet de renseigner les sommaires des namespaces, la documentation des namespaces n'étant pas gérée par le framework.
Le bouton Prj Summary, lui, nous permet de renseigner un sommaire pour le projet.



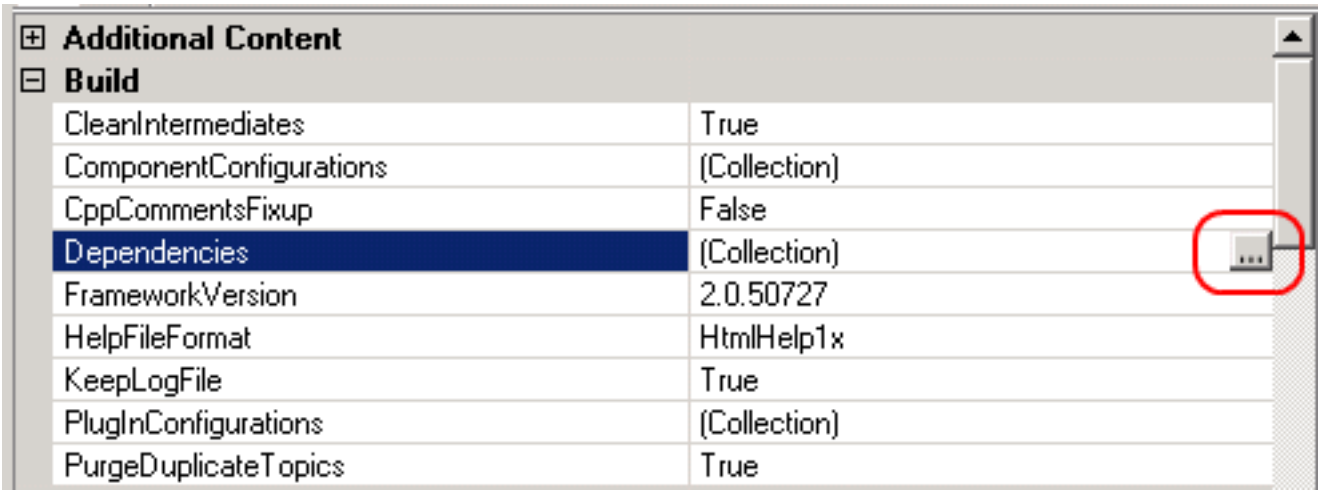
Pour tester ces sections, nous allons ajouter un commentaire court...et faire un test de génération.
Après quelques secondes de génération, le processus s'arrête avec une erreur...



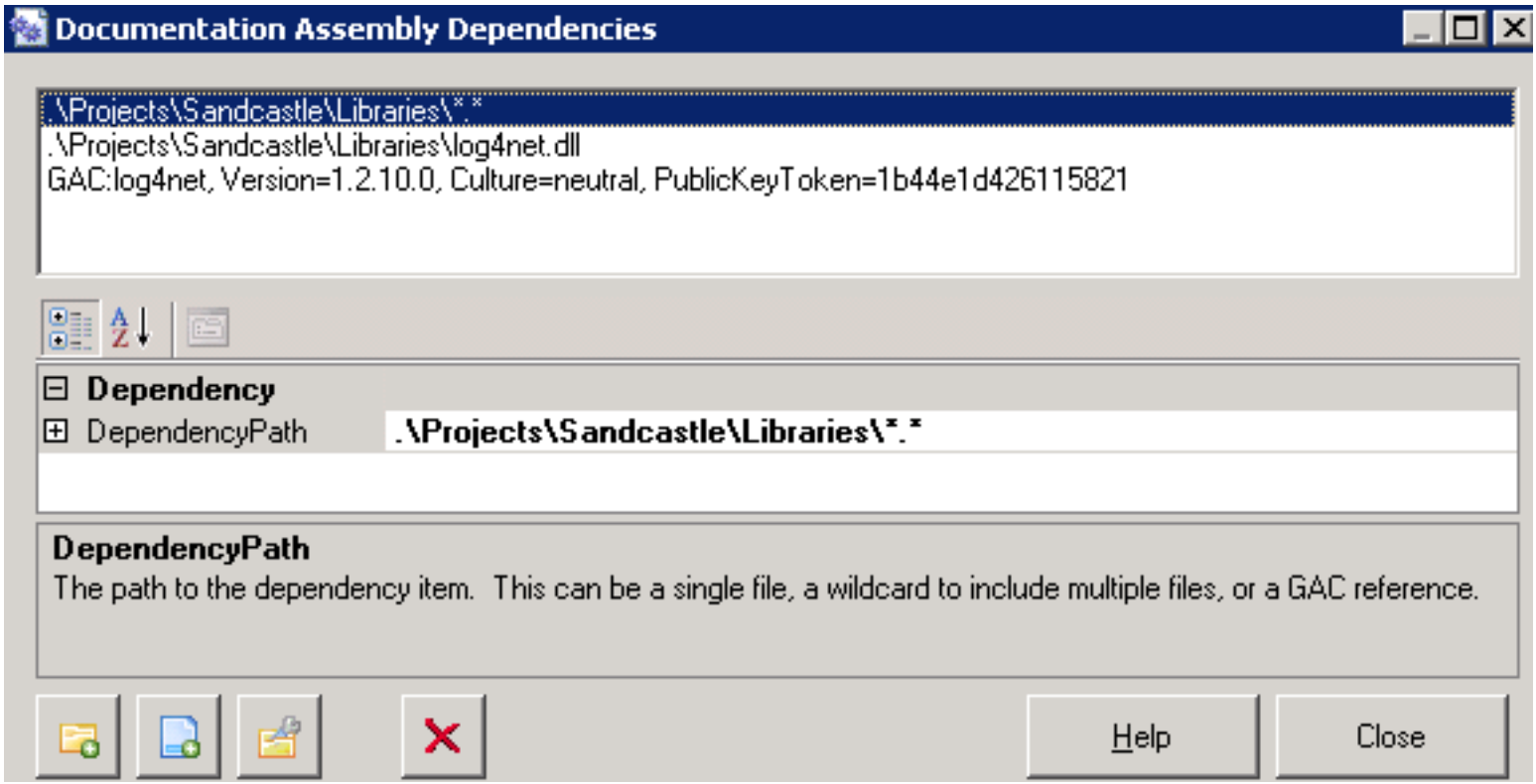
En effet, comme mentionné précédemment, nous avons, dans le namespace Core, une dépendance sur log4net...que nous n'avons pas rajoutée.
Ce que nous allons faire tout de suite...

Sélection des dépendances des projets

Pour sélectionner les dépendances, il suffit de les ajouter dans les propriétés de projet, dans la rubrique build, à la ligne dependencies.

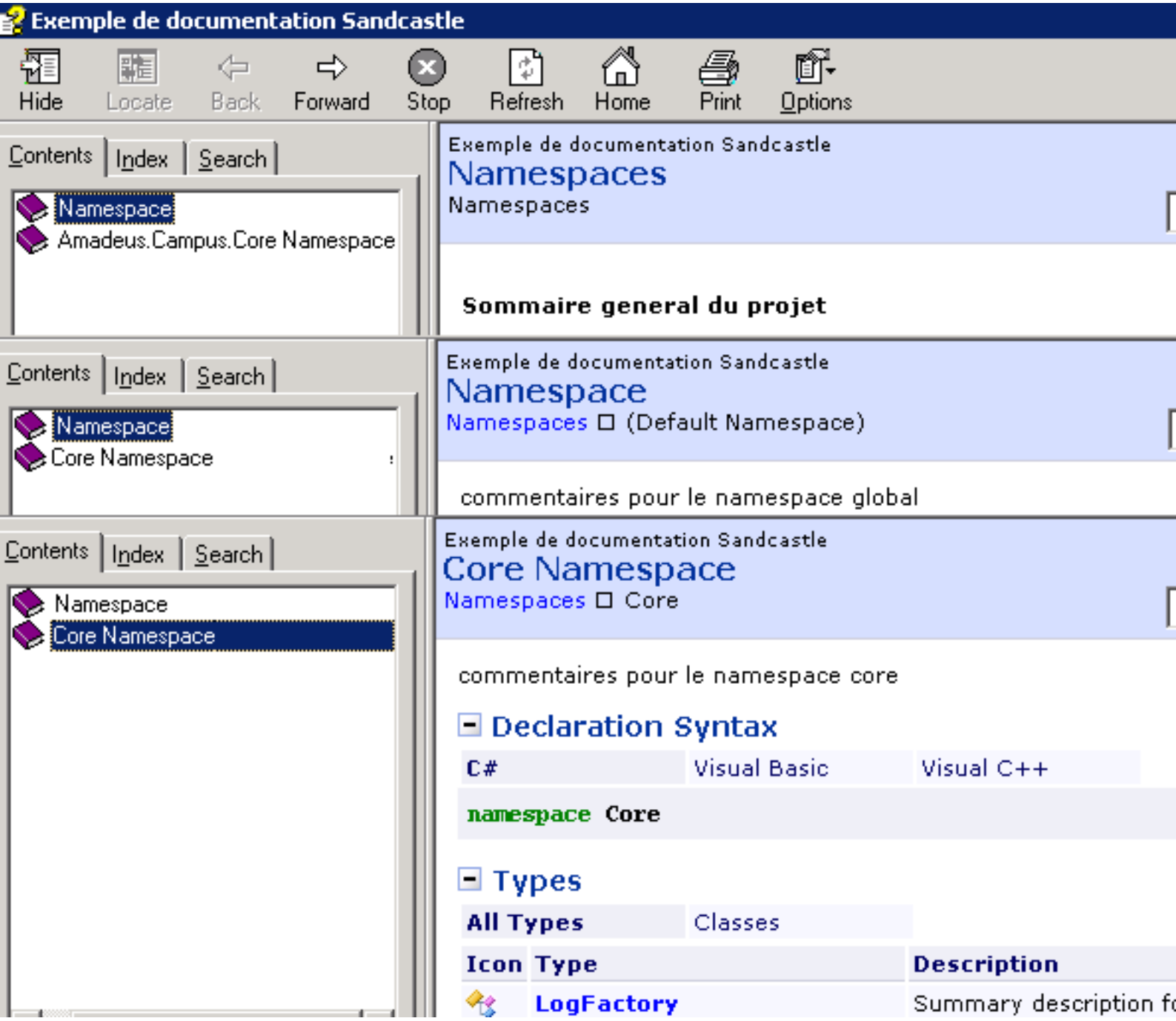


Dans cette fenêtre, nous pouvons, au choix, ajouter des dépendances sur des fichiers, des répertoires, ou des dll dans le GAC.



Nous pouvons dans ce dialogue, choisir soit de stocker le chemin de façon absolue, soit de façon relative.

Une fois cette action d'ajout de dépendances effectuée, nous pouvons désormais faire notre première génération de documentation. Pour l'exemple, je ne vais générer que le namespace Core et le namespace global.



Enrichir notre fichier d'aide

Maintenant que le fichier d'aide peut être généré, nous allons, dans l'ordre :

- Ajouter un pied de page
- Ajouter un en-tête
- Ajouter des exemples de code
- Enrichir la page d'accueil, et ajouter d'autres pages

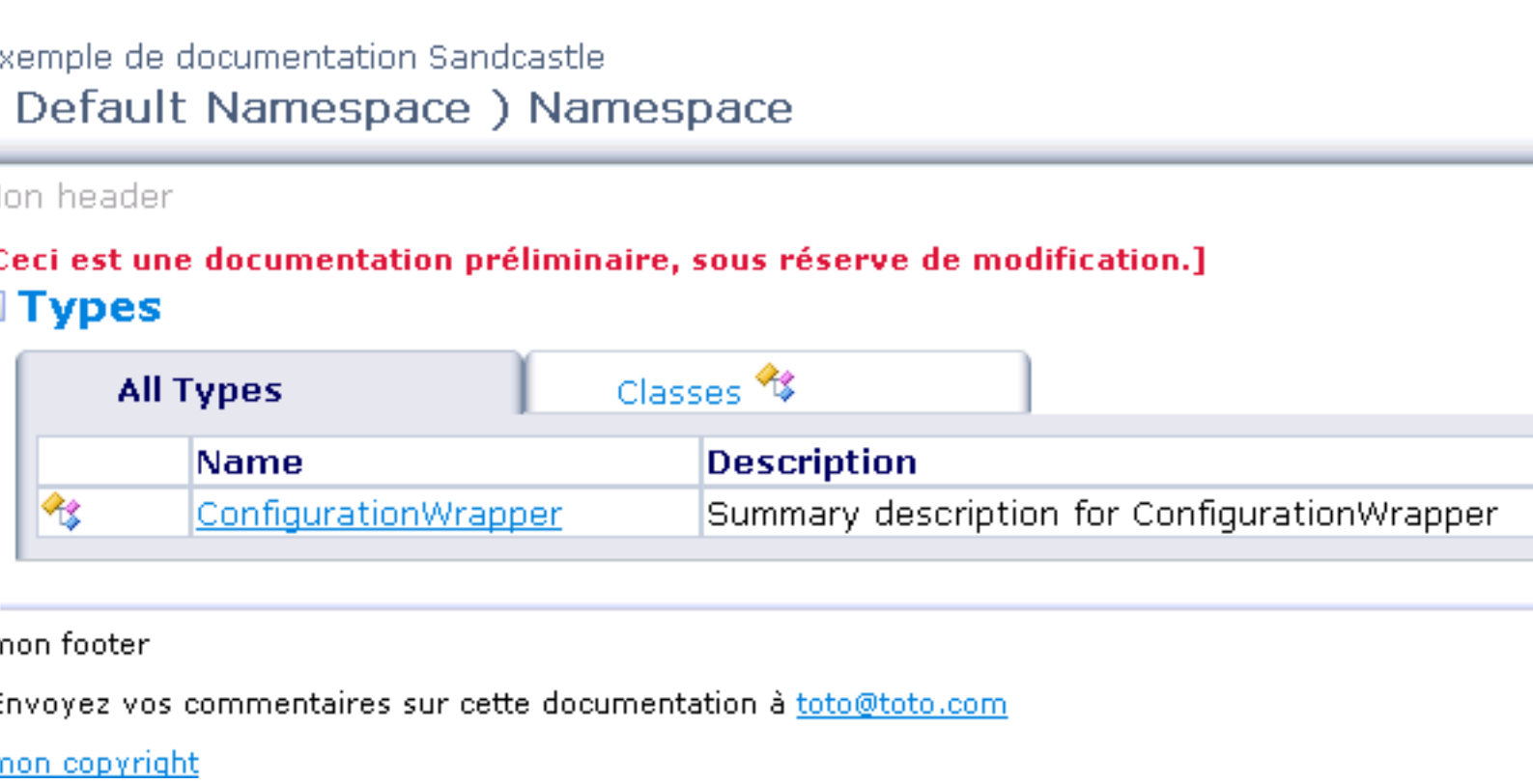
Ajout d'un en-tête et d'un pied de page

Nous avons bien sur la possibilité de modifier les en-têtes et pieds de page des pages générées. Pour cela, nous avons des champs tout prêts, dans la section "Help File".

Help File	
BinaryTOC	True
CopyrightHref	**http://localhost/copyleft**
CopyrightText	**mon copyright**
FeedbackEMailAddress	**toto@toto.com**
FooterText	**mon footer**
HeaderText	**Mon header**
HelpTitle	**Exemple de documentation Sandcastle**
HtmlHelpName	Documentation
IncludeFavorites	**True**
Language	**French (France)**
NamingMethod	Guid
Preliminary	**True**
PresentationStyle	**hana**

Ces champs nous permettent de modifier, dans l'ordre:

- **CopyRightHref** : Le lien du copyright
- **CopyRightText** : Le texte du copyright
- **FeedBackEmailAddress** : L'adresse e-mail de contact
- **FooterText** : Le texte d'information du pied de page
- **HeaderText** : Le texte d'information de l'en-tête
- **HelpTitle** : Le titre du fichier
- **HTMLHelpName** : Le nom du fichier
- **Language** : La page de langue à utiliser
- **Preliminary** : Si vrai, affiche un texte indiquant que la doc n'est pas définitive
- **PresentationStyle** : le style du fichier



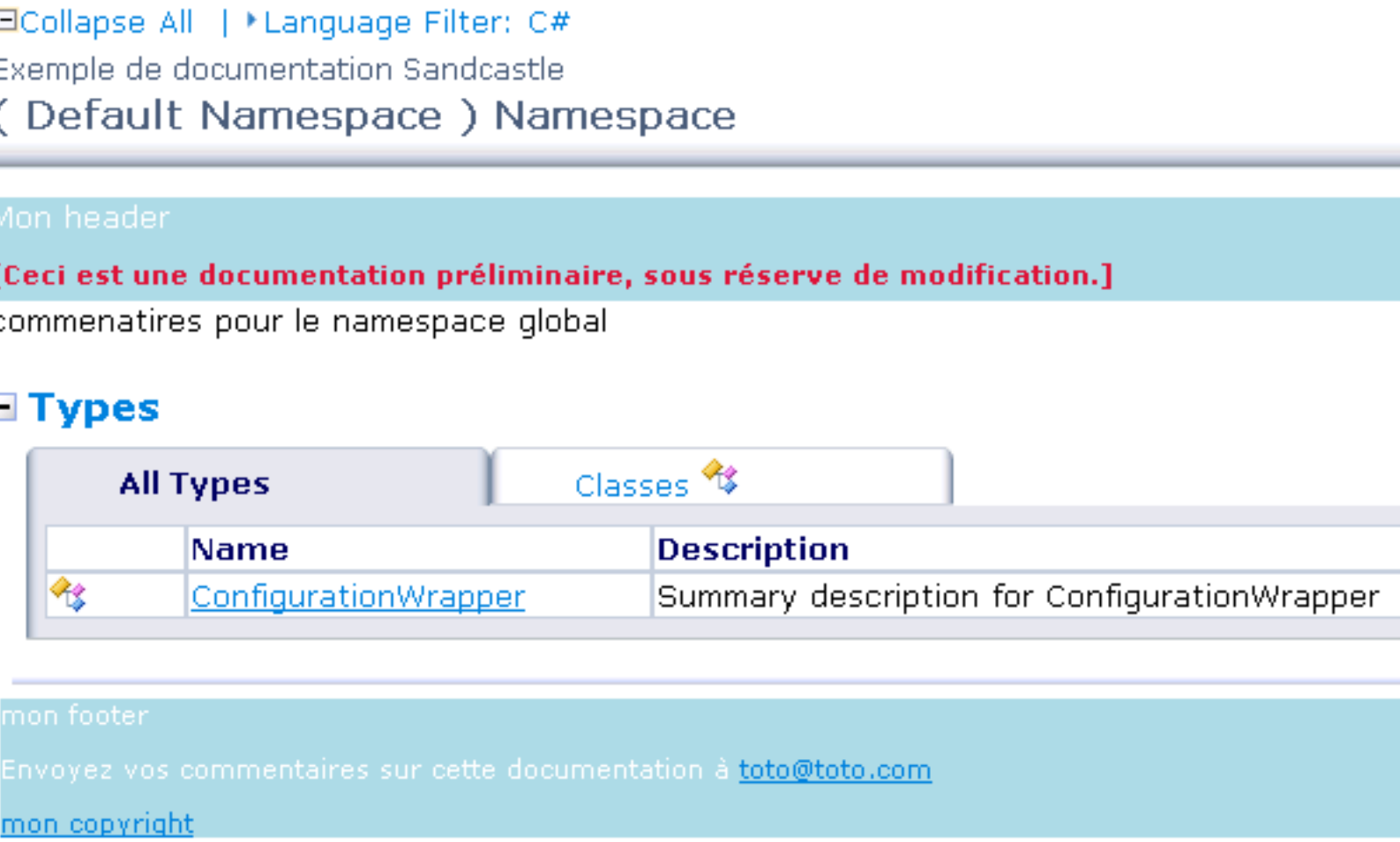
Nos en-têtes et pieds de pages sont bien visibles, mais manquent un peu de "punch".
Qu'à cela ne tienne, nous allons modifier les fichiers de templates.
Pour cela, nous allons nous rendre dans le répertoire d'installation de SandCastle Help File Builder, dans le sous répertoire SharedContent. Nous y trouverons tous les contenus partagés, et localisés, sous forme de fichiers xml.
Dans mon cas, j'utilise le Style Hana, en langue française. Je vais donc éditer le fichier **HanaBuilderContent_fr-FR.xml**.

```
<?xml version="1.0" encoding="utf-8"?>
<content xml:space="preserve">
    <!-- NOTE: This file must appear LAST in the list of content files
               so that these items override the matching items in the
               stock content files! -->

    <item id="header"><span style="color: DarkGray">{@HeaderText} {@Preliminary}</span></item>

    <item id="footer">{@FooterText}{@Comments} {@Copyright}</item>

</content>
```

Dans ce fichier, nous voyons bien que le header a comme propriétés d'avoir un texte gris, compose des éléments **HeaderText** et **Preliminary**, et que le Footer est la concatenation des éléments **FooterText**, **Comments** et **Copyright**.
Dans un premier temps, je vais changer mon header et mon footer de façon à ce qu'ils soient affichés en blanc sur fond bleu (je sais, je ne suis pas un designer ;))

```
<?xml version="1.0" encoding="utf-8"?>
<content xml:space="preserve">
    <!-- NOTE: This file must appear LAST in the list of content files
               so that these items override the matching items in the
               stock content files! -->

    <item id="header"><span style="width:100%;background-color:LightBlue;color: White">
            {@HeaderText} {@Preliminary}</span></item>

    <item id="footer"><span style="width:100%;background-color:LightBlue;color: White">
```

```
            {@FooterText}{@Comments} {@Copyright}</span></item>

</content>
```



C'est beau, non ???
...Non, parce que mon cher chef de projet veut absolument voir le texte préliminaire en jaune...
Et que je ne trouve pas cette donnée dans mon fichier xml...
C'est normal, elle se trouve en fait dans **SharedBuilderContent_fr-FR.xml**

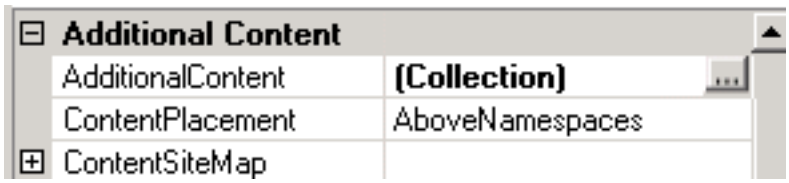
```
<?xml version="1.0" encoding="utf-8"?>
<!-- Translations provided by Thierry Huguet and his colleagues -->
<content xml:space="preserve">
    <!-- NOTE: This file must appear LAST in the list of content files
               so that these items override the matching items in the
               stock content files! -->

    <!-- reference_content.xml overrides -->
    <item id="rootTopicTitle">{@RootNamespaceTitle}</item>
    <item id="rootTopicTitleLocalized">Espaces de noms</item>
    <item id="rootLink"><referenceLink target="R:Project">{@RootNamespaceTitle}</referenceLink></item>
    <item id="productTitle">{@HtmlEncHelpTitle}</item>
    <item id="runningHeaderText">{@HtmlEncHelpTitle}</item>

    <item id="locationInformation">Assembly: {0} (Module: {1}) Version: {2}</item>
    <item id="assemblyNameAndModule">{0} (in {1})<br/><b>Version:</b> {2}</item>

    <!-- shared_content.xml overrides -->
    <item id="copyCode">Copy</item>

    <item id="locale">{@Locale}</item>
```

```
    <item id="preliminary"><p style="color: #dc143c; font-size: 8.5pt; font-weight: bold;">
    [Ceci est une documentation préliminaire, sous réserve de modification.]</p></item>

    <item id="comments"><p />Envoyez vos commentaires sur cette documentation à
        <a id="HT_MailLink" href="mailto:{@UrlEncFeedbackEMailAddress}?Subject={@UrlEncHelpTitle}">
        {@HtmlEncFeedbackEMailAddress}</a>
        <script type="text/javascript">
        var HT_mailLink = document.getElementById("HT_MailLink");
        var HT_mailLinkText = HT_mailLink.innerHTML;
        HT_mailLink.href += ": " + document.title;
        HT_mailLink.innerHTML = HT_mailLinkText;
        </script></item>

    <!-- To format the copyright HREF and/or copyright text into a message of
         your choosing, you can specify @HtmlEncCopyrightHref and/or
         @HtmlEncCopyrightText in braces -->
    <item id="copyright">{@HtmlEncCopyrightInfo}</item>

</content>
```

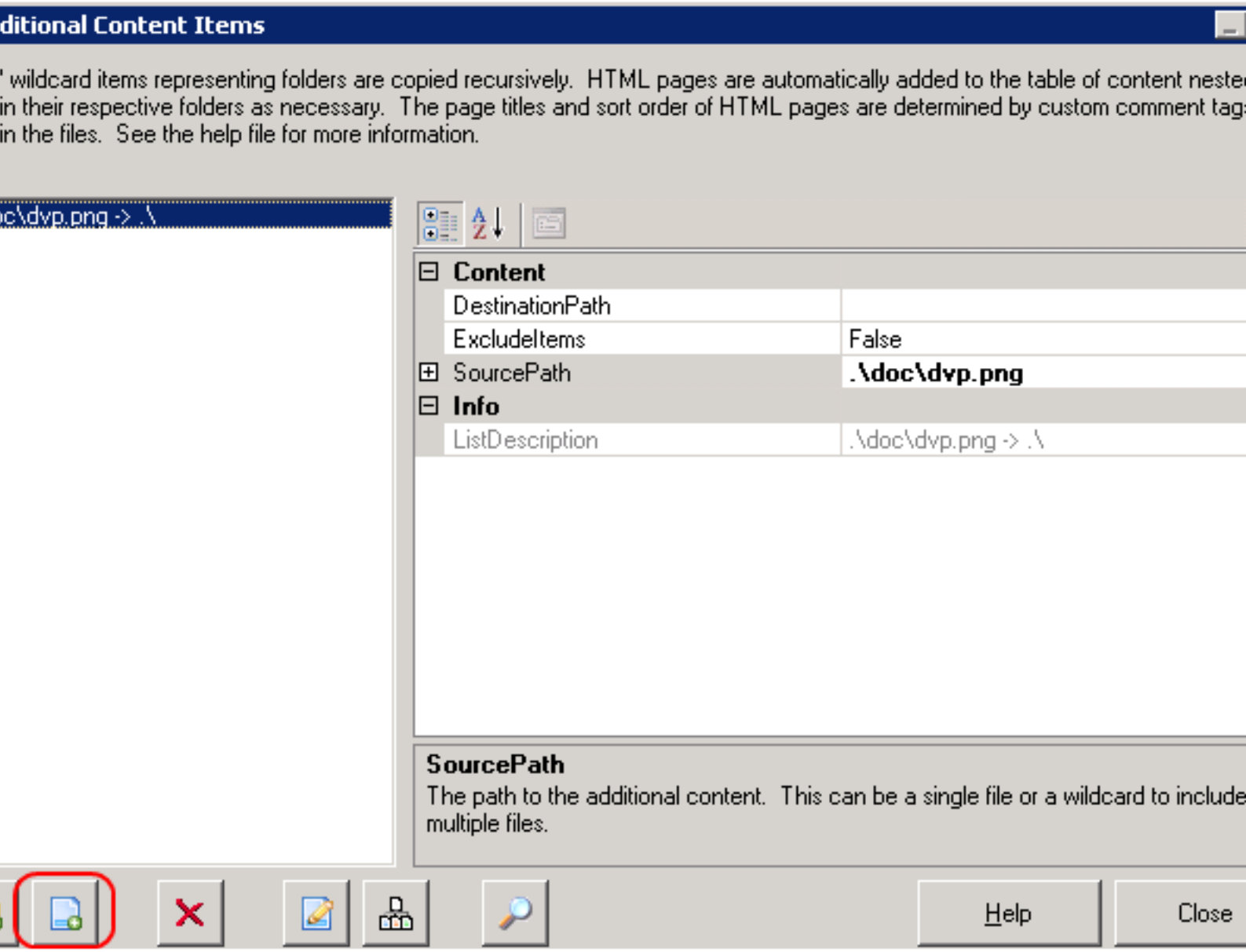
Ca y est, je n'ai plus qu'à mettre à jour l'item **preliminary**

```
 <item id="preliminary"><p style="color: Yellow; font-size: 8.5pt; font-weight: bold;">
 [Ceci est une documentation préliminaire, sous réserve de modification.]</p></item>
```

Ok, mon chef de projet est content. Mais il manque encore une chose, à savoir un joli logo dans l'en-tête, pour faire plus pro.
Cette manip n'est heureusement pas beaucoup plus difficile. Il suffit de modifier, une fois de plus, le fichier **HanaBuilderContent_fr-FR.xml**.
Sauf que là, se pose la question, comment ajouter une image ?
Pour cela, dans l'éditeur, nous irons dans la rubrique "Additional Content", sur la ligne "Additional Content", cliquer sur le bouton "..."



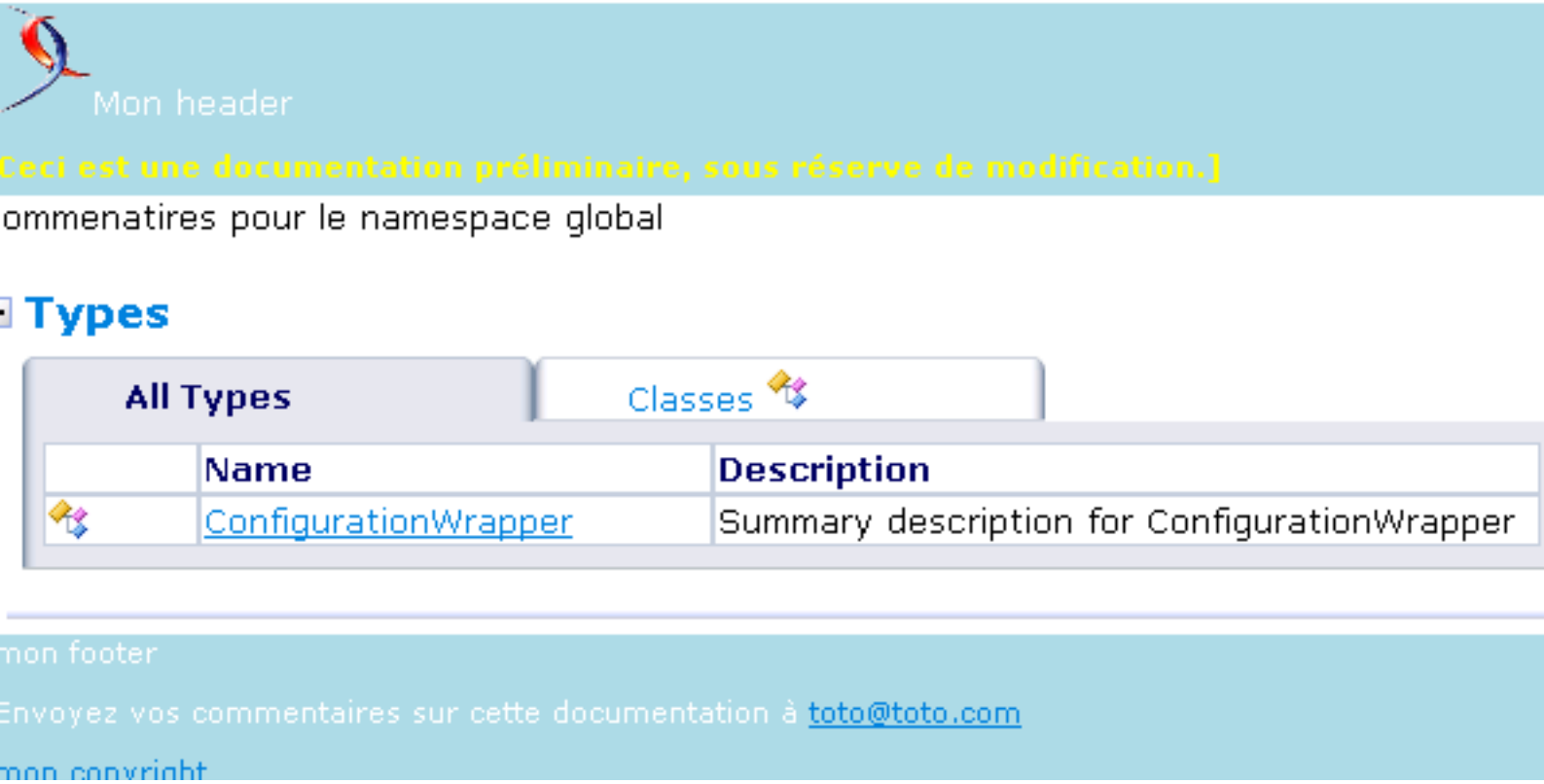
Une fois dans l'éditeur de contenus, je vais aller chercher le logo que je veux ajouter, ici, dvp.png. Ce logo va pouvoir être accédé depuis le fichier chm comme si il était à la racine d'un site.



Je vais ensuite modifier mon fichier pour obtenir ceci

```
<item id="header"><span style="width:100%;background-color:LightBlue;color: White">
      <img src="dvp.png"/>  {@HeaderText} {@Preliminary}</span></item>
```

Et, après génération, mon fichier d'aide aura dorénavant le look suivant.



Notez bien que cette manipulation va affecter TOUS les fichiers de documentation générés depuis le style Hana, en francais...

Ajout d'exemples de code multi-langages

Notre fichier d'aide commence à ressembler à quelque chose de sympathique.
Par contre, il reste un peu maigre en informations, et plutôt inutile à fournir dans l'état à un client (interne ou externe), dans le cas d'une API.
Nous allons donc ajouter des exemples de code dans notre fichier.
Le tag code existe déjà, avec un tag example, dans les tags de base de la documentation Visual studio. Le problème est que ces tags ne font pas de différence entre C# et vb.net.
Heureusement, SandCastle Help File Builder fournit un composant de compilation supplémentaire, nommé "code blocks", et dont le but est justement de résoudre ce problème. Pour cela, nous modifierons notre code de cette façon :

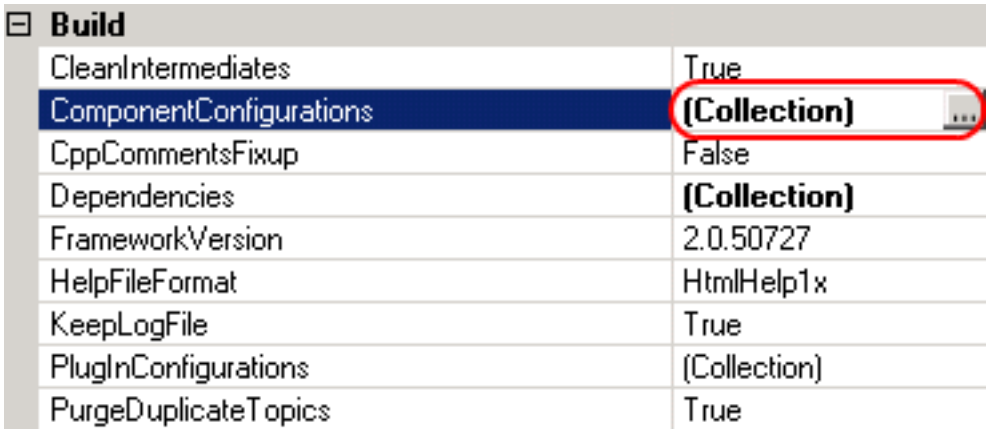
```
/// <summary>
       /// Logs a debug message
       /// </summary>
       /// <example>The debug messages can be logged using the following code
       /// <code lang="vbnet">LogFactory.LogDebug(10)</code>
       /// <code lang="c#">LogFactory.LogDebug(10);</code>
       /// </example>
       /// <param name="message">The message.</param>
       public static void LogDebug(string message) {
           log.Debug(message);
       }
```

Il sera ensuite traduit, dans le fichier XML généré, par:

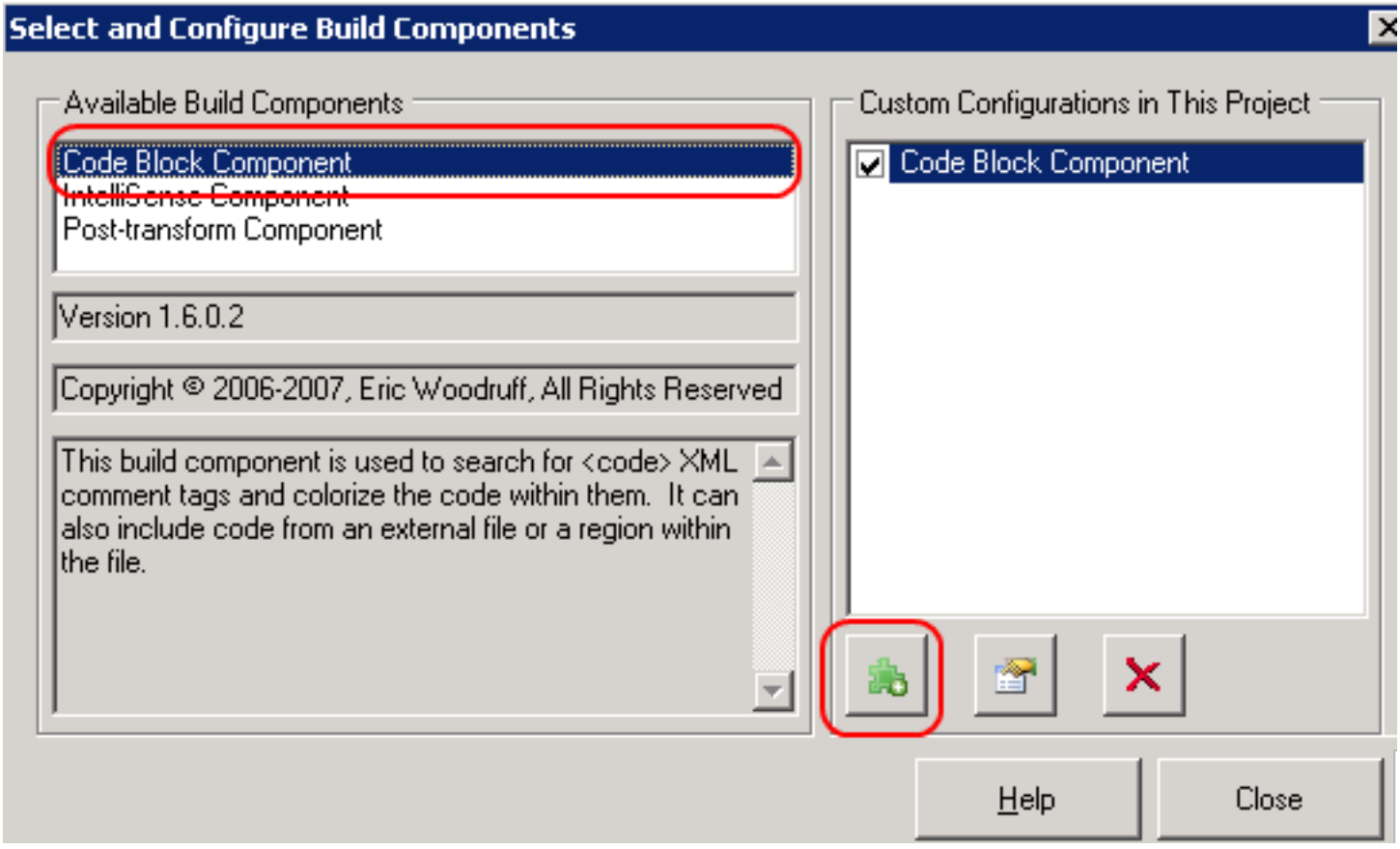
```
           <example>The debug messages can be logged using the following code
           <code lang="vbnet">LogFactory.LogDebug(10)</code>
           <code lang="c#">LogFactory.LogDebug(10);</code>
```

```
            </example>
```

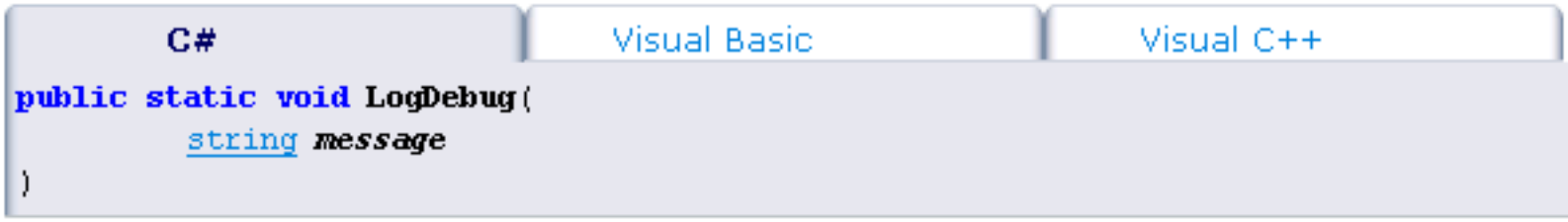
Ensuite, nous allons modifier, dans SandCastle Help File Builder, la liste des composants de compilation utilisés. Pour cela, dans la rubrique build, nous allons cliquer sur le bouton "..." de la ligne Componentconfigurations.



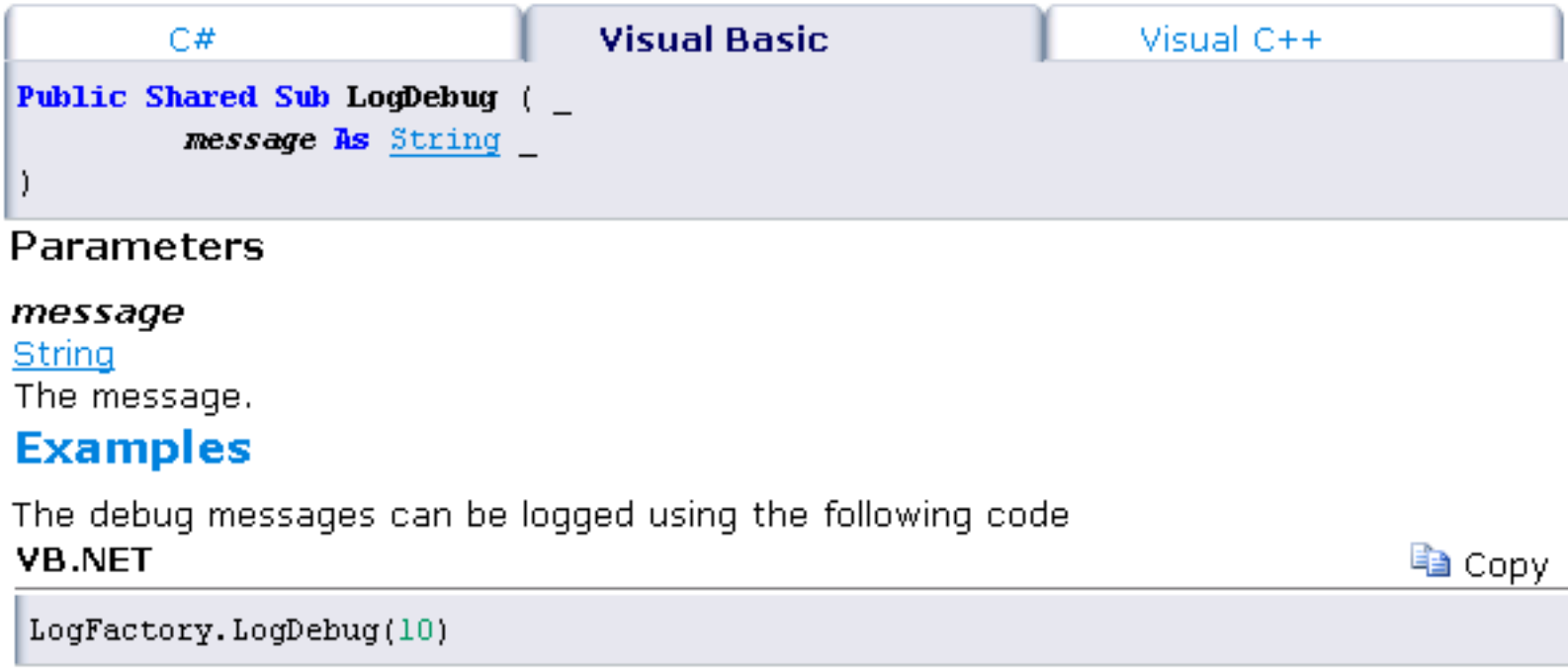
Puis, dans la rubrique Available build components, nous allons sélectionner "Code Block Component", et cliquer sur le bouton d'ajout, en bas, à gauche, de façon à obtenir l'écran suivant :



Et avec ces simples actions, nous aurons dorénavant les exemples dans notre langage de choix dans le fichier d'aide, comme ci-dessous:



Commentaire en C#

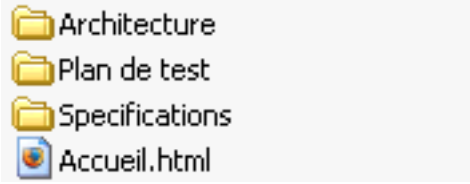


Commentaire en VB

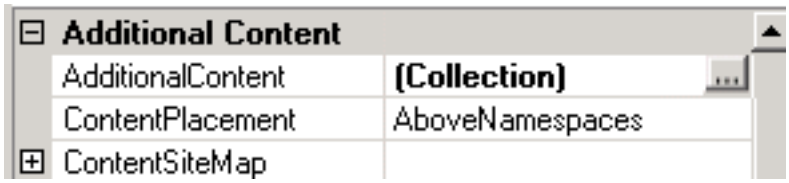
Ajout de pages supplémentaires

Pour cette partie, nous allons admettre que mon projet est un projet modèle, avec toute la documentation qui va bien. Maintenant qu'nous pouvons identifier facilement ma société, mon nom de projet, et que j'ai ajouté un moyen pour contacter l'équipe de projet depuis la doc, je vais vouloir étendre encore ma documentation. Nous pourrions imaginer deux cas "métier" où cela serait utile.
Le premier est celui de la fourniture de dll tierce partie, en externe ou en interne.
Si vous avez déjà utilisé des composants fournis par un revendeur, vous vous êtes sûrement aperçu qu'une grande partie de la documentation contient des exemples d'utilisation des bibliothèques fournies.

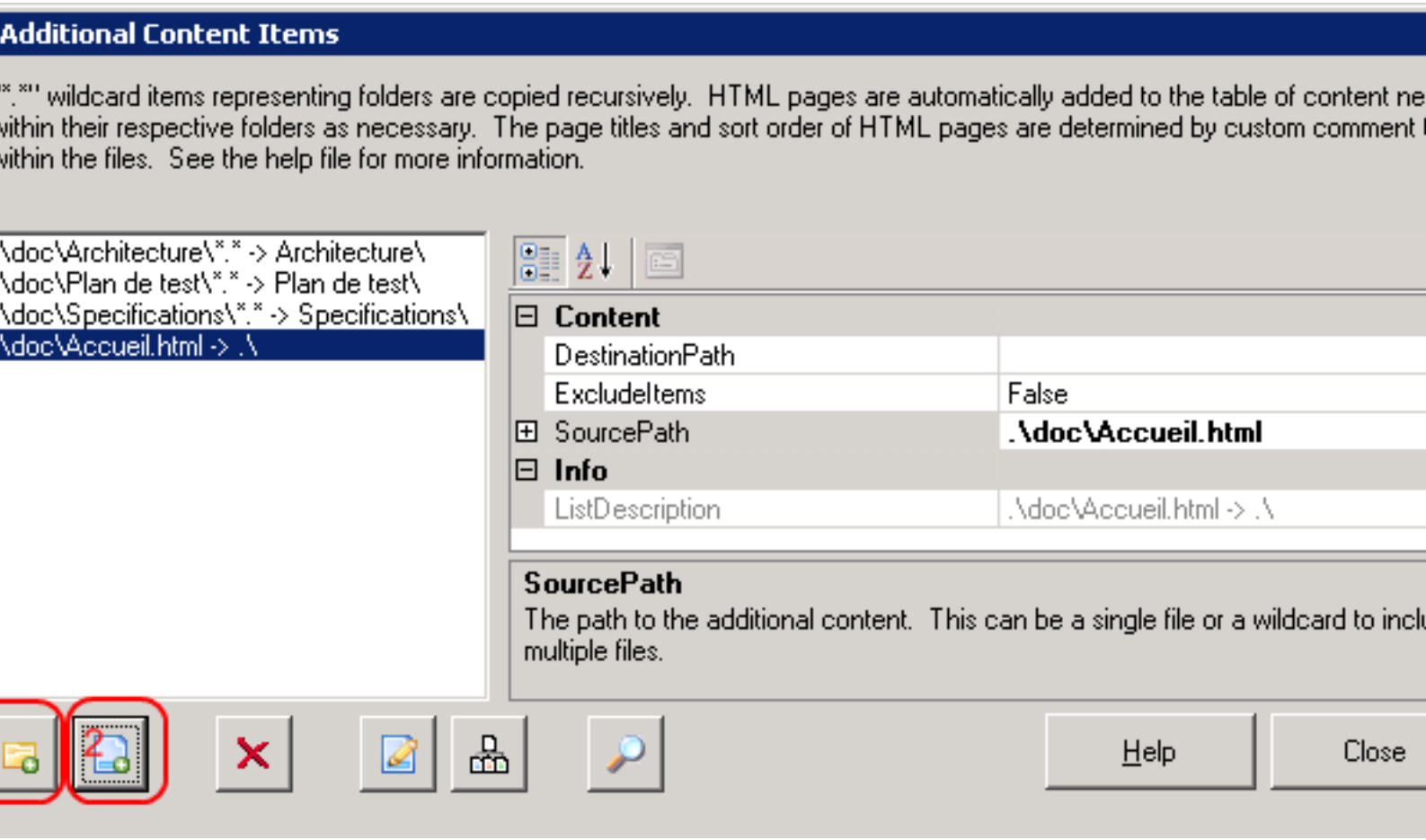
Le second cas, plus particulièrement répandu en interne, est celui d'une documentation projet un peu plus extensive, qui va contenir en plus de la documentation des assemblages, des classes, et des fonctions, une partie de description des choix d'architectures, et qui va être fournie en "package d'accueil" aux nouveaux venus.
nous allons utiliser des documents au format html, qui vont être inclus dans notre fichier d'aide. Dans un premier temps, nous allons devoir évidemment écrire ces fichiers, puis les stocker dans un répertoire de notre choix. Dans notre exemple, nous allons ajouter une partie "Spécification", une partie "Architecture", une partie "Plan de test", et une page d'accueil. Après préparation des différentes pages, mon répertoire de documentation html va ressembler à cela :



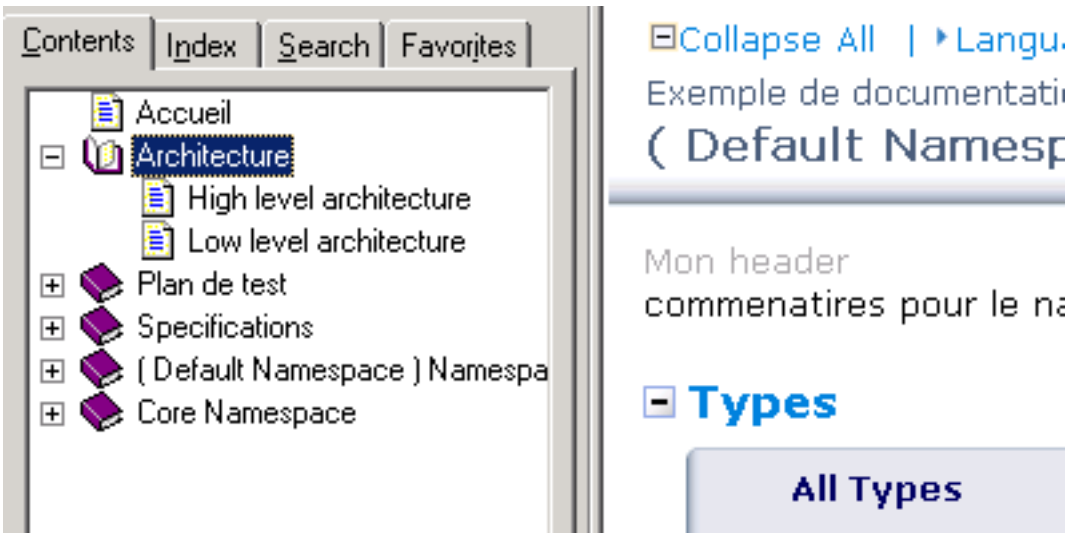
Une fois les fichiers préparés, nous allons utiliser Sandcastle Help file Builder pour les inclure à notre documentation. Pour cela, nous allons aller dans la rubrique "Additional Content", sur la ligne "Additional Content", cliquer sur le bouton "..."
(En passant, on constate que le placement du contenu additionnel peut se faire soit au dessus, soit au dessous des espaces de nom.)



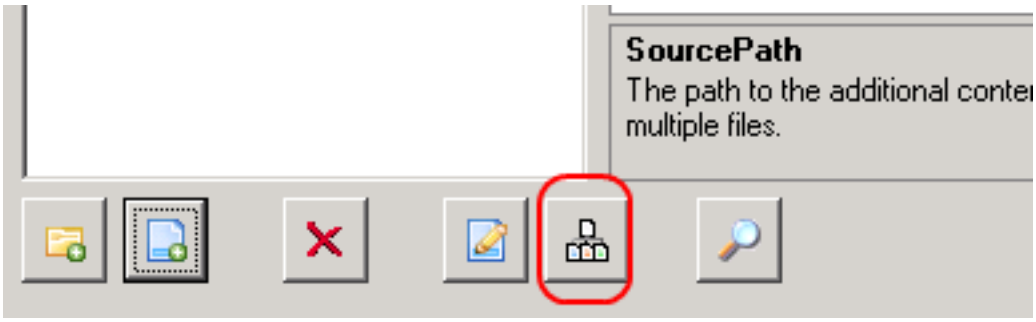
nous allons, dans un premier temps, ajouter les trois dossiers, en cliquant sur le bouton "dossier", puis ajouter le fichier Accueil en cliquant sur le bouton "fichiers". Notez qu'il existe un bouton éditer, qui permet une édition du source du fichier html. Nous obtennons donc l'écran suivant:



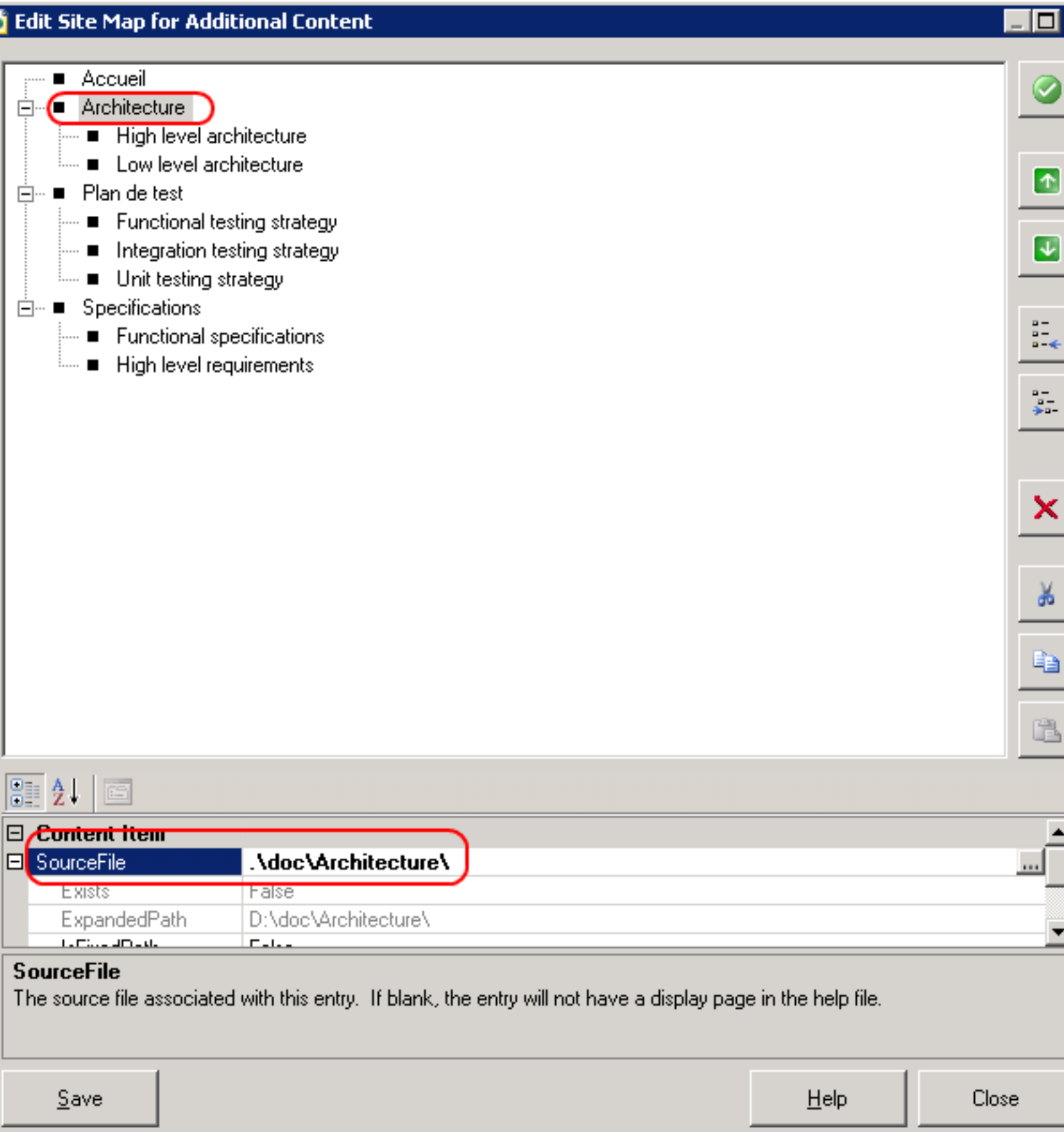
Après une génération du contenu, nous nous rendons néanmoins compte de l'oubli d'un détail, à savoir que les pages "Spécification", "Architecture", et "Plan de test" n'ont pas de descriptif.



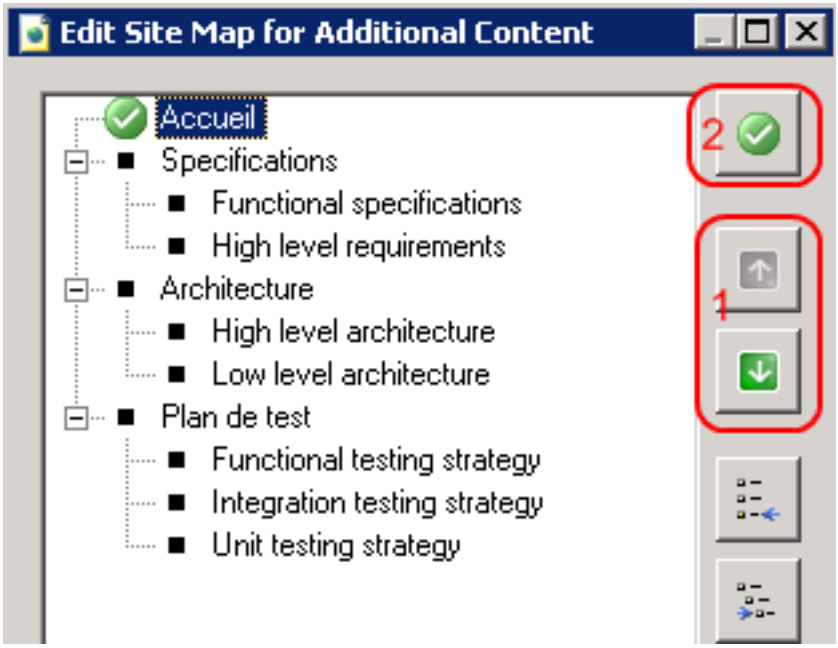
En effet, il faut associer à chaque n#ud de l'arbre une feuille html pour pouvoir avoir un affichage quelconque dans notre fichier de doc. nous allons donc éditer le sitemap du fichier chm pour pouvoir associer à chaque rubrique une page d'index. Pour cela, nous allons retourner dans la fenêtre d'édition du contenu, cliquer sur le bouton d'édition du sitemp



Dans l'éditeur de sitemap, nous allons ensuite sélectionner les n#uds auxquels nous voulons ajouter une page d'accueil, ici, Architecture, et nous allons faire pointer la page vers une page existante.



Tant que nous y sommes, nous allons aussi réarranger le menu, en mettant en premier Accueil, puis Spécifications, Architecture et plan de test, en utilisant les boutons haut et bas. nous allons aussi sélectionner la page d'accueil comme page par défaut, en la cliquant, puis en cliquant le bouton de sélection de page d'accueil, en haut à droite. Le nouveau sitemap aura la représentation suivante:



Sandcastle Help File builder génère alors automatiquement un fichier sitemap, et, à la prochaine génération nous obtennons le fichier de nos rêves ;).

IV - Intégration de Sandcastle à un système d'intégration continue

Ca y est, le fichier d'aide peut enfin être généré facilement, maintenant, il ne reste plus qu'à l'automatiser.
Pour cela, nous allons voir comment l'intégrer avec deux systèmes d'intégration continue, Cruise Control et Team system.
En fait, dans un cas comme dans l'autre, nous allons utiliser la fonctionnalité de génération en mode console de Sandcastle Help File Builder.

Cruise control.Net

```
<cruisecontrol>
  <project name="SandCastle">
    <workingDirectory>D:\Projects\Sandcastle</workingDirectory>
    <artifactDirectory>D:\Projects\Sandcastle\Tmp\</artifactDirectory>
    <modificationDelaySeconds>0</modificationDelaySeconds>
    <sourcecontrol>
      <!-- on passe ici les details de recuperation du code source... -->
    </sourcecontrol>
    <triggers>
      <scheduleTrigger time="22:00:00" name="NightlyTrigger" />
    </triggers>
    <tasks>
      <nant>
        <executable>d:\Tools\nant\nant-0.85\bin\NAnt.exe</executable>
        <baseDirectory>D:\Projects\Sandcastle\Source</baseDirectory>
        <buildFile>ArticleSandCastle.build</buildFile>
        <nologo>False</nologo>
        <buildTimeoutSeconds>1200</buildTimeoutSeconds>
      </nant>
      <exec>
        <executable>C:\Program Files\EWSoftware\Sandcastle Help File Builder
\SandcastleBuilderConsole.exe</executable>
        <baseDirectory>D:\Projects\Sandcastle\</baseDirectory>
        <buildArgs>"D:\Projects\Sandcastle\Documentation.shfb"</buildArgs>
        <buildTimeoutSeconds>10800</buildTimeoutSeconds>
      </exec>
```

```
    </tasks>
    <publishers>
      <xmllogger logDir="c:\Program Files\CruiseControl.NET\Log" />
    </publishers>
  </project>
</cruisecontrol>
```

Si vous n'avez pas l'habitude de voir du code de configuration CruiseControl, le code ci-dessus va appeler une tache de build nant (pour compiler le projet), puis va exécuter SandCastle Help File Builder en mode console, avec comme répertoire de base D:\Projects\Sandcastle\, ouvrir Documentation.shfb, et laisser trois heures (10800) secondes au processus pour s'achever.

Team system

Intégrer la génération de documentation à VSTS Team Build se fait très facilement.
Pour cela, il suffit de faire un check out du fichier de build TFSbuild.proj, et d'y ajouter les lignes suivantes, de façon à créer une nouvelle cible de build :

```
<Target Name="AfterCompile">
    <Exec Command="&amp;quot;C:\Program Files\EWSoftware\Sandcastle Help File Builder
\SandcastleBuilderConsole.exe&amp;quot;
    &amp;quot;D:\Projects\Sandcastle\Documentation.shfb&amp;quot;" />
</Target>
```

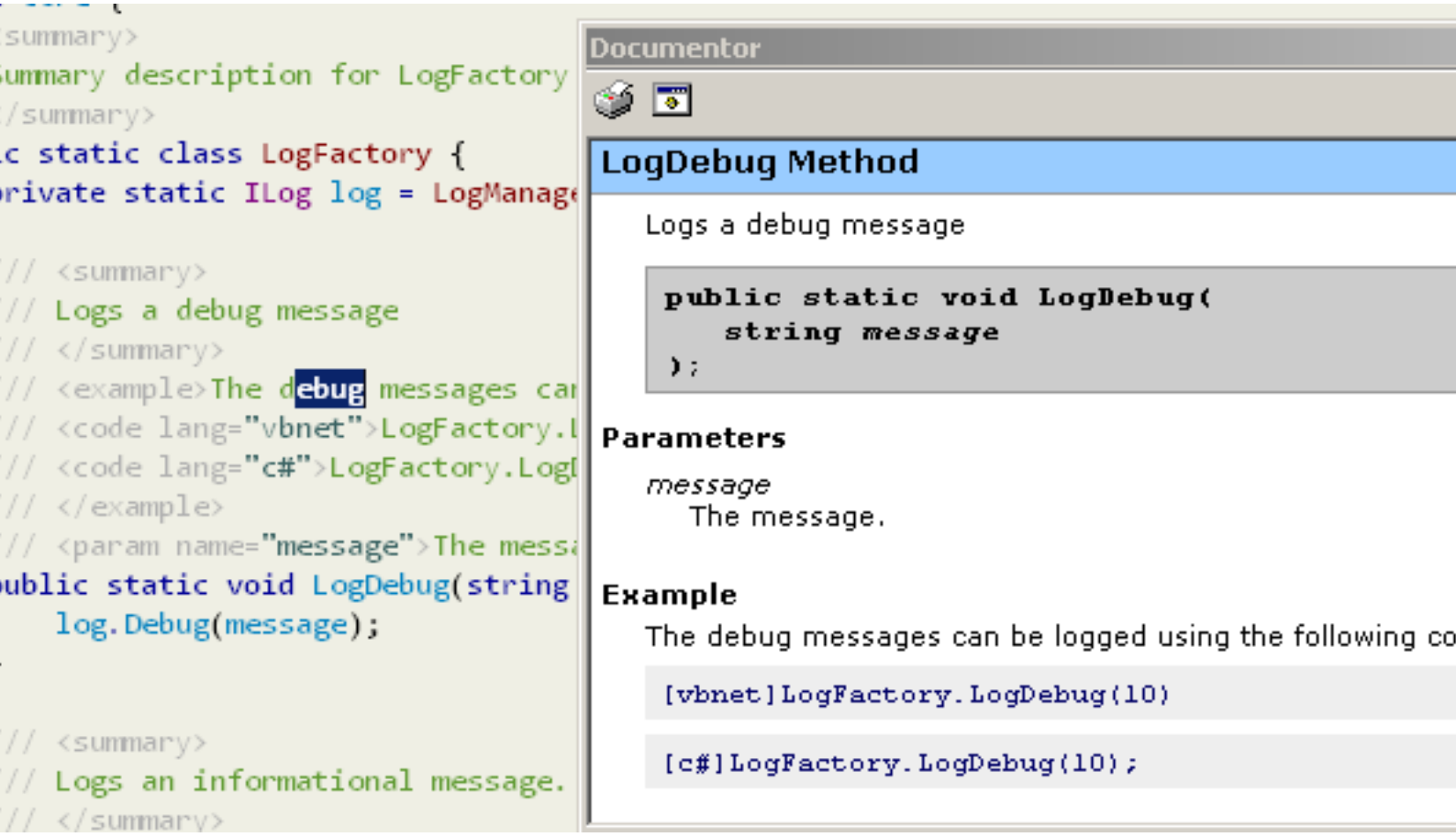
Dans ce cas, la documentation sera générée dans le répertoire spécifié dans le fichier .shfb.

Dans les deux cas, nous admettrons que le fichier shfb est rapatrié par le contrôleur de code source (ou déjà hébergé) sur le serveur.

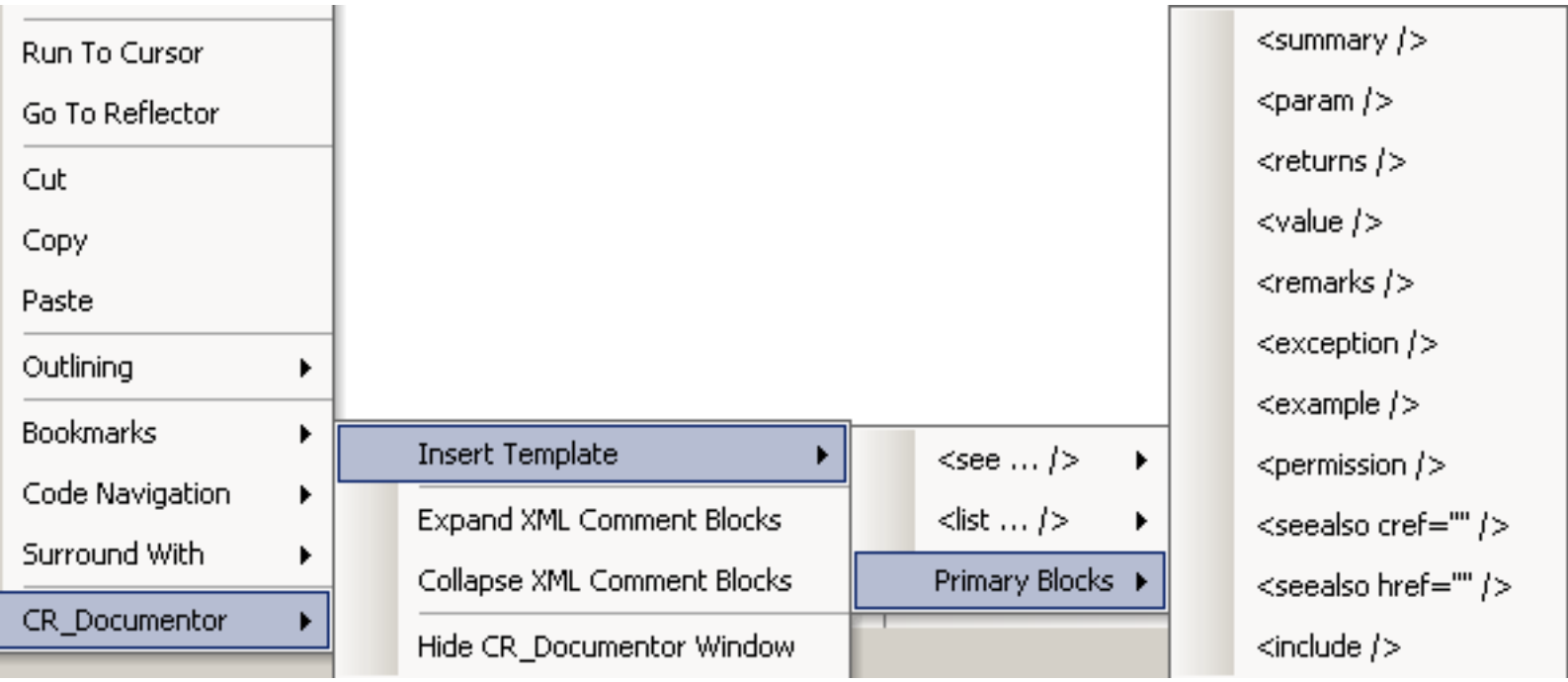
V - Outils Annexes

CR Documentor

CR Documentor est un plug-in gratuit dont le but est d'aider à la construction et à la prévisualisation des commentaires de méthodes et de classes.
Il permet de visualiser le rendu de la documentation directement depuis Visual Studio. Pour cela, il est nécessaire de télécharger **le framework DXCore** (un peu lourd...20 mo) et la dernière version de **CR Documentor**. Une fois ces fichiers téléchargés, et installés, vous pourrez pré-visualiser vos documentations d'un simple clic...



De plus, cet outil propose un ensemble de modèles pour insérer facilement des listes, ou les tags de base de documentation.

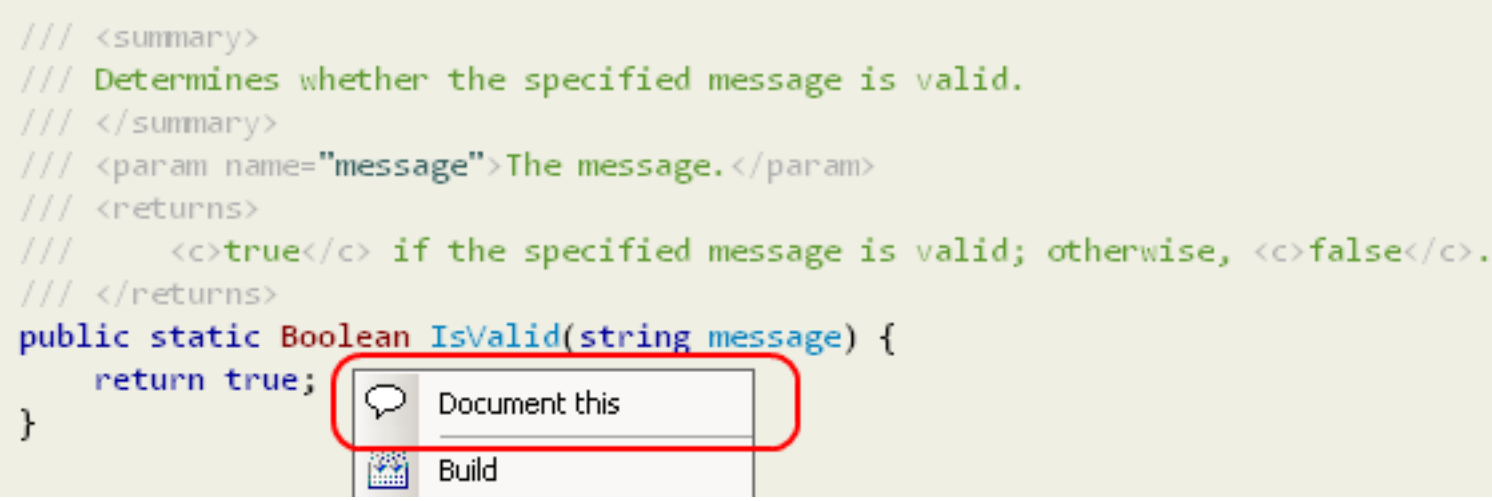


GhostDoc

GhostDoc est un outil de génération gratuit pour Visual Studio (2003 et 2005) qui permet de générer automatiquement les commentaires XML pour les méthodes, propriétés, etc.
Cette documentation est générée à partir du nom et du type de la méthode, et peut éventuellement récupérer des informations supplémentaires des classes de base pour enrichir la documentation.
Dans l'exemple de la classe Core, si nous ajoutions une nouvelle fonction hypothétique validant un message:

```
public static Boolean IsValid(string message) {
  return true;
}
```

Un simple clic droit, puis un clic sur l'icone de GhostDoc, va nous générer automatiquement un squelette de documentation assez abouti, qu'il ne restera plus qu'à retravailler.
Attention, cet outil m'est très utile, car je travaille dans un environnement...anglais...
Ceci dit, il semble possible de mettre à jour les règles syntaxiques de GhostDoc, avis aux courageux ;).



Nous pouvons trouver les fichiers d'installation pour GhostDoc sur le site suivant **GhostDoc**

VI - Remerciements

Merci à toute l'équipe de Developpez.net, et à **Lou Pitchoun** pour sa relecture de l'article.