SandCastle

par Olivier DELMOTTE

Date de publication : XX/XX/2007

Dernière mise à jour : XX/XX/2007

.Net offre, depuis sa création, la possibilité de documenter son code par le biais de commentaires XML. Cette possibilité était d'abord réservée à C#, puis à VB.Net par l'intermédiaire d'un add-in pour Visual Studio. Maintenant cette fonctionnalité est disponible quelque soit votre langage préféré. Mais disposer du XML c'est bien, encore faut-il le transformer pour le rendre facilement et rapidement exploitable. C'est la mission de SandCastle.

1 - SandCastle

Comme je viens de vous le dire, le but de SandCastle est de génerer rapidement et facilement une documentation fiable et facile d'accès de son code source. Nous allons voir que ce n'est pas forcément le cas mais qu'il existe des outils pour remédier à ca.

A l'heure où j'écris ces lignes, SandCastle est encore en développement, mais est déjà utilisable. La preuve, il est utilisé en interne chez Microsoft pour générer la documentation du Framework. Pour le télécharger, rien de plus simple : rendez vous sur le site de Microsoft. La version que j'utilise pour ce tutoriel est la CTP de Décembre 2006.

> *Il existe le projet NDoc qui ressemble fortement au projet SandCastle. Cependant, NDoc, d'après la page du projet, semble laissé à l'abandon. De plus, NDoc ne supporte pas les nouveautés du Framework 2.0, dont une des plus intéressante : les Generics.*

Le gros inconvénient de SandCastle, à mon avis, est qu'il ne propose pas de mode graphique pour gérer la génération de la documentation. Tout se fait en mode console. Nous verrons qu'il existe des projets externes qui permettent de palier ce défaut, dont un sur lequel nous nous attarderons et qui me semble très prométeur.

La structure de SandCastle est relativement simple. Il est divisé en trois composants sur lesquels nous allons jeter un oeil.

1-1 - Préparation

Si vous regardez l'arborescence du dossier d'installation de SandCastle, vous remarquerez un dossier Examples. Ce dossier contient une classe toute simple que nous utiliserons comme exemple. Compilez la, c'est pas bien compliqué. Pensez à générer le XML associé.

Exemple de code fourni par SandCastle

```
namespace TestNamespace {

        /// <summary> A test class.</summary>
        public class StoredNumber {

                /// <summary>Initializes the stored number class with a starting value.</summary>
                public StoredNumber (int value) {
                        number = value;
                }

                private int number;

                /// <summary>Increments the stored number by one.</summary>
                public void Increment () {
                        number++;
                }

                /// <summary>Increments the stored number by a specified step.</summary>
                public void Increment (int step) {
                        number = number + step;
                }

                /// <summary>Gets the stored number.</summary>
                public int Value {
                        get {
                                return(number);
                        }
                }

        }

}
```

Depuis une console dans le dossier d'exemple, compilation
```
csc /target:library /doc:test.xml /recurse:*
```

Vérifiez que vous aillez bien les fichiers test.dll et test.xml dans le dossier courant.

1-2 - MrefBuilder

Maintenant voyons à quoi sert MrefBuilder. Son rôle est simple mais essentiel. Ce que va faire MrefBuilder c'est prendre chacune des assemblys spécifiées et en faire une reflexion totale et donner un résultat sous forme de XML. Son utilisation n'est pas des plus compliquées.

Liste complète des commandes de MrefBuilder
```
C:\Program Files\Sandcastle\Examples\Sandcastle>mrefbuilder /?
MrefBuilder (v2.2.61208.1447)
Copyright © Microsoft 2006
MRefBuilder [options] assemblies

/?
Show this help page.

/out:outputFilePath
Specify an output file. If unspecified, output goes to the console.

/config:configFilePath
Specify a configuration file. If unspecified, MRefBuilder.config is used

/dep:dependencyAssembly[,dependencyAssembly,dependencyAssembly,...]
Speficy assemblies to load for dependencies.

/internal+|-
Specify whether to document internal as well as externally exposed APIs.
```

- /out:outputFilePath : c'est le fichier qui contiendra le résultat de la réflexion de votre assembly et éventuellement des assemblys satellites.
- /config:configFilePath : il est possible de spécifier un fichier de configuration spécifique. Celui par défaut ira très bien dans note cas.
- /dep:dependencyAssembly[,dependencyAssembly,dependencyAssembly,...] : la liste de tous les assemblys nécessaires à faire la réflexion de notre assembly. Les assemblys de base du Framework ne sont pas à spécifier.
- /internal+|- : spécifie si les membres non visibles doivent être pris en compte (+) ou non (-)

Petit retour sur l'option /dep. Les assemblys du GAC ne sont actuellement pas prises en compte automatiquement, il faudra en faire une copie dans le meme dossier que l'assembly cible. Ceci exclu les assemblys de base du Framework (system.dll, system.windows.forms.dll, ...)

Voyons maintenant 2 exemples de génération de fichiers de référence :

Génération des références avec mrefbuilder option - pour internal (par défaut)
```
C:\Program Files\Sandcastle\Examples\Sandcastle>mrefbuilder /out:tempxml\base.xml te
st.dll
MrefBuilder (v2.2.61208.1447)
Copyright © Microsoft 2006
Info: Loaded 1 assemblies for reflection and 0 dependency assemblies.
Info: Wrote information on 1 namespaces, 1 types, and 4 members
```

Génération des références avec mrefbuilder option - pour internal (par défaut)
```
C:\Program Files\Sandcastle\Examples\Sandcastle>mrefbuilder /out:tempxml\base.xml /i
nternal+ test.dll
MrefBuilder (v2.2.61208.1447)
Copyright © Microsoft 2006
```

Génération des références avec mrefbuilder option - pour internal (par défaut)
```
Info: Loaded 1 assemblies for reflection and 0 dependency assemblies.
Info: Wrote information on 2 namespaces, 2 types, and 5 members
```

Pensez à créer le répertoire de sortie s'il n'est pas le même que celui de travail. Ca vous évitera une jolie erreur s'il n'existe pas.

Normalement, vous devriez avoir un fichier refbuild.xml dans votre dossier de travail. Si vous l'ouvrez, vous verrez tous les détails de vos assemblys. Nous l'appelerons reference.

Passons à la suite avec la partie la plus interessante.

1-3 - XslTransform

La liste des types, méthodes, propriétés, ... de notre assembly générée avec MrefBuilder est complète, mais elle ne permet pas de structurer suffisament les données pour pouvoir générer notre documentation. Il est donc nécessaire de traiter ces données et de les transformer. C'est la qu'intervient XslTransform. Nous allons générer à partir de notre liste de base d'autres listes plus structurée et qui correspondront plus à nos besoins. Pour se faire, Microsoft à mis à notre disposition plusieurs XSL Stylesheets.

- AddFriendlyFilenames.xsl
- AddGuidFilenames.xsl
- AddMemberLists.xsl
- AddOverloads.xsl
- AddRoot.xsl
- AddXamlAttachedMembers.xsl
- ApplyVSDocModel.xsl
- ReflectionToChmContents.xsl
- ReflectionToChmIndex.xsl
- ReflectionToChmProject.xsl
- ReflectionToHxSContents.xsl
- ReflectionToManifest.xsl
- RemoveExplicitImplementations.xsl
- TocToChmContents.xsl
- TocToHxSContents.xsl
- TocToManifest.xsl

Tous ne sont pas utilisés de la même façon selon le type de documentation à générer. Nous allons voir tout de suite comment les utiliser.

1-4 - BuildAssembler

BuildAssembler est le dernier qui nous permettras de générer la documentation à proprement parler.

- Générer des pages HTML
- Générer un fichier de Documentation pour Visual Studio
- Générer un fichier CHM (Compiled HTML)

Pour faire simple, et surtout pasqu'il y a moyen de faire tout ca beaucoup plus simplement, nous allons nous limiter à la création d'une documentation au format HTML.

La première étape consiste à préparer le dossier de sortie. Pour cela nous utiliserons un dossier Output dans le dossier où nous travaillons. Nous allons y copier des dossier situés dans le répertoire Presentation/vs2005 du dossier d'installation de SandCastle. Ces répertoires sont : Icons, Scripts et Styles. Comme vous vous en doutez, nous allons utiliser le style de documentation de Visual Studio 2005. Profitons-en pour créer un dossier html dans celui que nous venons de créer.

La deuxième étape consiste à complèter un peu plus notre liste que nous avons crée avec MrefBuilder. C'est la qu'intervient XslTransform. Nous allons appliquer 2 xsl à notre fichier reference. Le premier va ordonner notre reference. C'est à dire regrouper les surcharges d'une même méthodes afin de faciliter le travail par la suite. Le second va générer les noms de fichiers qui correspondront à la page de documentation de tel ou tel membre.

```
xsltransform /xsl:..\..\ProductionTransforms\AddOverloads.xsl tempxml\reference.xml
/out:tempxml\withoverloads.xml
xsltransform /xsl:..\..\ProductionTransforms\AddFriendlyFilenames.xsl tempxml\withoverloads.xml
/out:tempxml\withfriendlyfilenames.xml
```

Si vous regardez maintenant le fichier withfriendlyfilenames.xml du dossier tempxml, vous verrez que toutes les méthodes sont regroupées et qu'il y a une balise file qui apparait.

La troisième étape vas nous conduire à générer notre index. La encore c'est XmlTransform qui intervient :

```
xsltransform /xsl:..\..\ProductionTransforms\ReflectionToManifest.xsl tempxml\reference.xml
/out:tempxml\index.xml
```

La dernière étape : la génération. Avant toute chose, il faut copier le fichier de configuration sandcastle.config contenu dans le dossier Presentation\vs2005\Configuration du répertoire d'installation de SandCastle vers le dossier de travail.

2 - Liens