

Versionnez et Déployez vos Assemblies .NET

Mis à jour le 09/07/2003

Par Elise Dupont

Droit de diffusion:

L'ensemble ou partie de ce document ainsi que le code mis à disposition, ne peut être diffusé sur d'autres sites Web sans l'autorisation au préalable de son créateur.

Sommaire:

- 1. Introduction
- 2. Définitions
- 3. Avantages et inconvénients du Versionnage
- 4. Comment versionner son Assembly ?
 - 4.1 Donner une version
 - 4.2 Donner un Nom fort (ou strong name)
- 5. Comment déployer son Assembly après l'avoir versionnée
 - 5.1 Pour un poste client
 - 5.2 Pour un développement
 - 5.3 Rendre l'Assembly visible sous Visual Studio .NET

1. Introduction

La notion d'Assemblies a été introduite afin de simplifier le déploiement d'applications et éviter les conflits de versions (fréquents lorsque l'on développe des applications basées sur la réutilisation de composants). En effet, avant les Assemblies, la compatibilité entre les différentes versions d'une librairie, n'était assurée que par "compatibilité ascendante". Cela vous est sûrement déjà arrivé de voir d'anciens programmes ne plus fonctionner suite à l'installation d'un programme qui a modifié certaines DLL partagées par plusieurs applications.

La notion d'Assembly supprime définitivement ce problème. Le développeur peut définir des règles de versionnage sur les composants qu'il utilise. De plus le **CLR** fournit la possibilité d'exécuter plusieurs versions d'un même composant en même temps (exécution "côte à côte" ou "side-by-side" en anglais).

Ce cours a donc pour but de vous expliquer les étapes pour versionner et déployer proprement vos Assemblies. En effet, une fois que vous avez fini de développer vos objets, méthodes et autres composants, qu'ils soient destinés à être utilisés par d'autres développeurs ou par des applications, dès que vous mettrez à jour vos applications, il vous faudra peut-être jongler avec plusieurs versions. Ou alors vos objets méthodes et autres seront sur la même machine utilisés par plusieurs applications... Dans tous ces cas, vous aurez sûrement besoin de versionner vos composants, et/ou de les déployer de façon claire et propre sur les machines clientes (utilisatrices de vos Assemblies). Le Versionnage est étroitement lié au

"Strong Naming", qui vous permet de définir un Nom dur pour votre Assembly. Ces deux étapes (Versionnage et Strong Naming) sont obligatoires si vous voulez déployer vos Assemblies dans le **cache de l'assembly global** (Global Assembly Cache en anglais ou **GAC**).

2. Définitions

- **Assembly** : Bloc d'assemblage minimum pour construire une application. Cela peut être une Dll par exemple. L'assembly est donc la plus petite unité d'exécution que l'on trouve dans le fonctionnement d'une application pour le framework. Les assemblies sont donc des unités de déploiement, de contrôle de version, de réutilisabilité, de visibilité de types et enfin de contrôle de la sécurité.
- **Versionnage (*Versioning*)** : Mécanisme qui consiste à conserver la version d'une entité logicielle quelconque, de façon à pouvoir la retrouver facilement, même après l'apparition et la mise en place de versions plus récentes.
- **Déploiement (*Deployment*)** : Le déploiement peut se traduire par l'installation massive de matériels et de logiciels. De ce fait, il implique une mise à l'échelle des méthodes de gestion et une standardisation. En informatique décisionnelle notamment, un déploiement peut constituer une étape cruciale dans le développement de l'entreprise.
- **Global Assembly Cache (GAC)** : Ce cache se présente sous la forme d'un répertoire dans le système de fichiers qui est C:\WINDOWS\assembly pour la plupart des installations Windows. Il constitue une sorte de répertoire public pour les composants, dans lequel les applications .NET ont accès.

3. Avantages et inconvénients du Versionnage

Avantages

- Permet d'avoir sur une même machine plusieurs versions d'un même composant et de charger dynamiquement et facilement la version nécessaire.
- Permet d'identifier de façon sûre un composant et sa version (si vous gérez plusieurs dll que vous mettez souvent à jour, cela devient vite le chaos sans versionnage).
- Permet de publier publiquement vos composants dans le Global Assembly Cache (GAC).

Inconvénients

- Ajoute des étapes supplémentaires dans le déploiement (et prends donc un peu de temps).
- Si un de vos composant est versionné et qu'il fait référence à d'autres composants, chacun des composants référencés doit lui aussi être versionné.

4. Comment versionner son Assembly ?

Le versionnage est composé de deux parties inséparables :

- Donner un **numéro de version** à votre Assembly
- Donner un **nom fort** à votre Assembly

Ces deux étapes se définissent de façon assez simple. En effet, quand vous créez votre composant / votre Dll (en général vous créez pour cela un nouveau Projet de Type Classe Librairie ou "Class Library Project" en anglais). A la création de ce projet, Visual Studio .NET vous ajoute automatiquement un fichier qui est justement là pour gérer toutes les informations générales relatives à une Assembly ! Parfait, c'est ce qu'il nous fallait ! Ce fichier s'appellera **AssemblyInfo.vb** dans un projet VB et **AssemblyInfo.cs** dans un projet C#. C'est ce fichier que nous allons éditer.

Nota bene : Si par hasard ce fichier n'existait pas, il vous suffit tout simplement de le créer vous-même, et d'y ajouter les quelques attributs nécessaires (tout ceci est trouvable facilement dans la documentation du [SDK](#)).

Etape 1 : Donner une version

Pour spécifier un numéro de version, il suffit d'éditer l'attribut **AssemblyVersion**. Cet attribut est sûrement déjà présent avec comme numéro "1.0.*".

Comment définir ce numéro ? Et bien il est composé de 4 valeurs : *Version Majeure*, *Version Mineure*, *Numéro de génération*, *Révision*.

Nous allons définir le numéro de version qui nous convient, tout simplement comme ceci :

```
<Assembly: AssemblyVersion("1.0.1.0")>
```

Etape 2 : Donner un Nom fort (ou strong name)

Donner un nom fort est une étape obligatoire si l'on veut ajouter une Assembly dans le GAC. Traduction d'après la documentation du Framework :

Un nom fort consiste à donner une identité à une Assembly - son nom simple sous forme de texte, son numéro de version, et ses informations sur la culture (s'ils sont fournis) - plus une clé publique et une signature digitale. Ces informations sont générées à partir d'un fichier Assembly (NDLR : le fichier AssemblyInfo) en utilisant la clé privée correspondante. Microsoft® Visual Studio .NET® et d'autres outils fournis par le Framework .NET SDK peuvent assigner un nom fort à une Assembly. Les Assembly qui ont le même nom fort sont supposées être identiques.

Et en pratique ça donne quoi ? après avoir donné un numéro de version à votre Assembly, il vous suffit de générer une clé, et de lier cette clé à votre Assembly. Vous pouvez si vous le voulez définir d'autres paramètres, mais je ne les exposerai pas ici car ils sont optionnels au bon fonctionnement de notre projet : Donner un nom fort.

- Générer la clé

Avant tout, sachez que tous les outils que je cite ici sont généralement tous au même endroit : Le répertoire d'installation du Framework (en général C:\Program Files\Microsoft Visual Studio .NET\FrameworkSDK\Bin\"). Lancez l'invite de commande DOS qui est fourni avec .NET (à défaut une invite de commande simple peut fonctionner, mais vous n'aurez pas les variables d'environnement du framework chargées et vous devrez spécifier les chemins entièrement).

Entrez la commande suivante : `sn -k <file name>`

SN est le nom de l'outil qui génère pour nous une clé. Dans cette commande, file name est le nom du fichier clé que nous obtiendrons en sortie (avec l'extension "snk" à préciser). L'exemple suivant crée un fichier clé nommé sgKey.snk :

```
sn -k sgKey.snk
```

Vous devez mettre cette clé dans le même répertoire que celui de votre projet (.vbproj ou .csproj) !

- Lier la clé à votre Assembly

Pour assigner cette clé à votre Assembly, il vous suffit d'éditer l'attribut "AssemblyKeyFile", toujours dans votre fichier AssemblyInfo. Comme ceci :

```
<Assembly: AssemblyKeyFile("../..//sgKey.snk")>
```

Si vous avez mit la clé au bon endroit, tout est bon, sinon un warning vous prévient qu'il n'a pas réussi à trouver le fichier correspondant.

Si tout a été fait correctement, en cliquant droit sur la dll générée et en regardant ses propriétés vous devriez voir s'afficher le numero de version (plus le nom de l'entreprise etc.. si vous les avez rajouté aussi dans le fichier AssemblyInfo).

5. Comment déployer son Assembly après l'avoir versionnée ?

L'étape finale consiste à déployer l'Assembly. Vous avez plusieurs possibilités, en fonction de vos besoins. De manière générale, nous supposons que si vous voulez déployer une Assembly qui a besoin d'être versionnée, c'est dans le but de la déployer pour :

- - un poste client où l'on installe aussi une (ou plusieurs) application(s).
- - un développeur qui l'utilisera dans son application

En fonction de la cible, je vous conseille des cas de figure différents.

Pour un poste client

Il faudra installer la DLL dans le GAC cette fois ci, enfin je vous le recommande si vous voulez gérer plusieurs versions en même temps. Vous avez deux façons de le faire :

- Grâce à l'outil GACUTIL :
- La commande de ligne est la suivante :
- Pour installer :

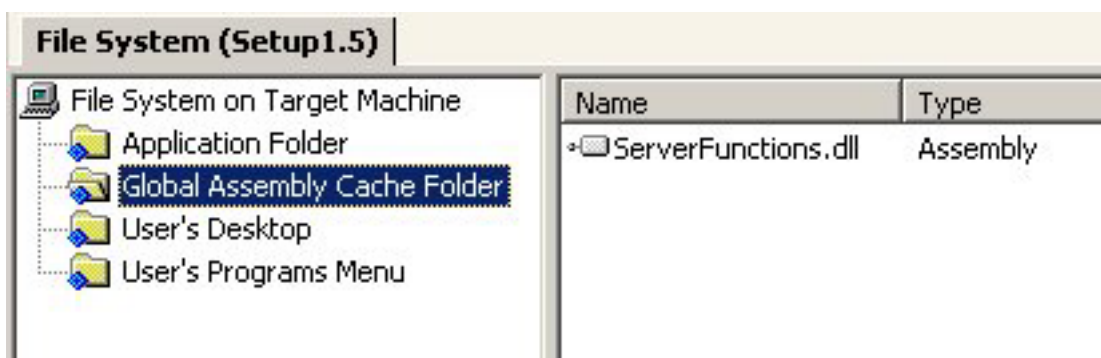
```
gacutil /i nom_de_la_dll.dll
```

Pour désinstaller :

```
gacutil /u MonAssembly
```

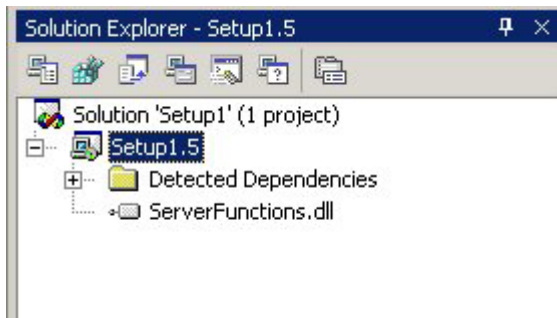
L'option /i installe. Pour désinstaller utiliser l'option /u à la place (mais au lieu de donner le nom de la DLL donnez le nom de l'Assembly).

- Vous pouvez aussi utiliser un Programme d'installation qui installera tout pour vous. Pour ce faire, créez un nouveau projet de type "Setup Project". Un onglet "File System" apparaît avec une liste prédéfinie de répertoires.



Répertoires d'installation

Vous pouvez en définir d'autres en cliquant droit dans la fenêtre. Cliquez droit, ajoutez le répertoire de type "Global Assembly Cache Folder", cliquez droit dans le répertoire ajouté et ajoutez votre Assembly là (ajouter fichier>ouvrez la dll). VS .NET détectera automatiquement les dépendances liées et les ajoutera au projet.



Dépendances trouvées automatiquement

Compilez, récupérez les fichiers (.msi en général) générés et utilisez-les pour installer vos Assemblies, comme vous le feriez pour un programme normal !

Pour un développement

- Vous pouvez tout simplement déposer les dll dans un répertoire, celui de l'application par exemple, ou un répertoire spécial pour votre Assembly + un répertoire par version. Il vous faudra donc jongler avec plusieurs répertoires.
- Vous pouvez aussi passer par un Setup, ajoutez-les dans le répertoire de votre choix (celui nommé "Application Folder" convient en général très bien, cela installera la DLL dans le chemin choisit par l'utilisateur au moment de l'installation).

Rendre l'Assembly visible sous Visual Studio .NET

Quand vous développez une Librairie, vous voudriez évidemment que Visual Studio .NET soit capable de la lister dans la boîte de dialogue "Ajouter une référence" sans que l'utilisateur ait besoin de cliquer sur "Browse" et d'indiquer le chemin de la dll. Ceci n'est pas faisable en ajoutant la librairie dans le GAC car la boîte de dialogue "Ajouter une référence" se base sur un chemin, et pas sur les composants du GAC. Pour rendre visible votre Assembly, vous devez ajouter une clé dans la base de registre comme ceci :

Sous le chemin

HKEY_CURRENT_USER\SOFTWARE\Microsoft\.NETFramework\AssemblyFolders\MyAssemblies ajoutez la clé suivante : **@="C:\MyAssemblies"** (peut-être définie dans le Setup project). Si vous ajoutez la clé sous HKEY_LOCAL_MACHINE au lieu de HKEY_CURRENT_USER cela appliquera les changements à tous les utilisateurs de la machine. Sinon cela n'affectera que les settings de l'utilisateur courant. Ensuite redémarrez VS. NET et c'est bon !



L'ensemble ou partie de ce document ainsi que le code mis à disposition, ne peut être diffusé ailleurs sans autorisation préalable

elise.dupont@europe.com - www.dotnet-tech.com - 2003