Le debugging en .NET

A quoi correspond le Debug en .NET

Il m'a souvent été demandé à quoi servait le fait de passer une compilation de projet .NET en release ou en debug. Dans le même sens, quel est la différence entre ces deux options. Je vais donc dans cet article essayer d'éclairer un peu la lanterne de tous à ce sujet.

Introduction

Nous avons déjà un article sur ce même site qui explique comment débugger une application ASP.NET dans Visual Studio .NET :

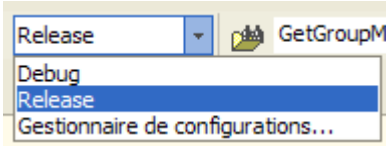
- **Débuguer une application WEB avec VS.NET (FR)**

Nous n'avons en revanche pas précisé la différence entre la compilation en mode Release et celle en mode Debug. Car la différence est conséquente et mérite que l'on s'y arrête.

Présentation

Que ce soit en développement avec Visual Studio .NET, un autre éditeur (Delphi ou SharpDevelop) ou via NotePad, lors de la compilation on peut spécifier une option de compilation parmi les suivantes (en développement WinForm ou WebForm) :

- Debug
- Release



Lorsque l'on compile un projet via le FrameWork (cs.exe par exemple), on passe des fichiers sources vers le fichier précompilé utilisable par le FrameWork (Code MSIL). On peut donc ajouter une option (/debug) permettant de rajouter des paramètres dans ce résultat.

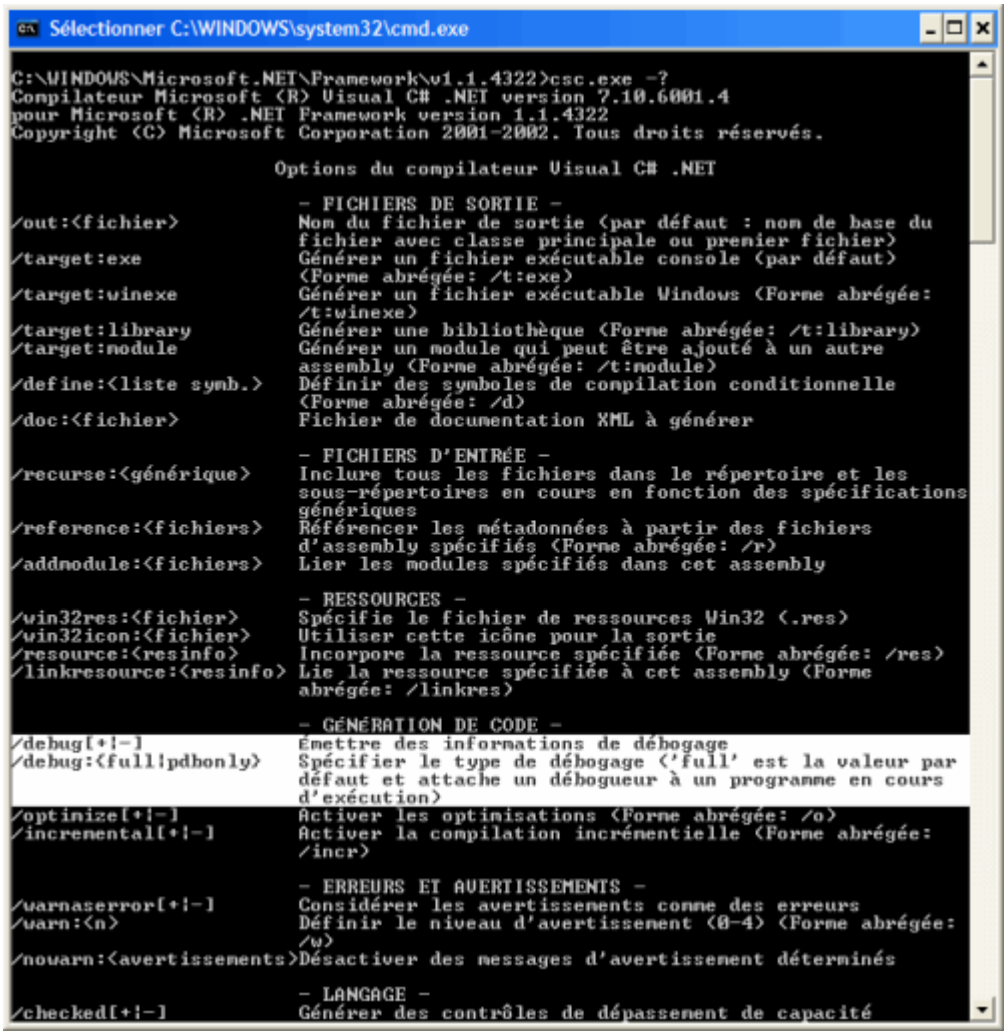
Voyons donc les principales différences.

Les Différences

Lors du développement d'un projet .NET, quelque soit l'éditeur choisi, au moment de la génération de la DLL, un appel sera fait vers le compilateur lui spécifiant le mode voulu pour cette DLL.

Ainsi pour un projet C#, cette compilation est faite via CSC.EXE. On a donc les options de ce compilateur en tapant en ligne de commande :

- csc.exe -?

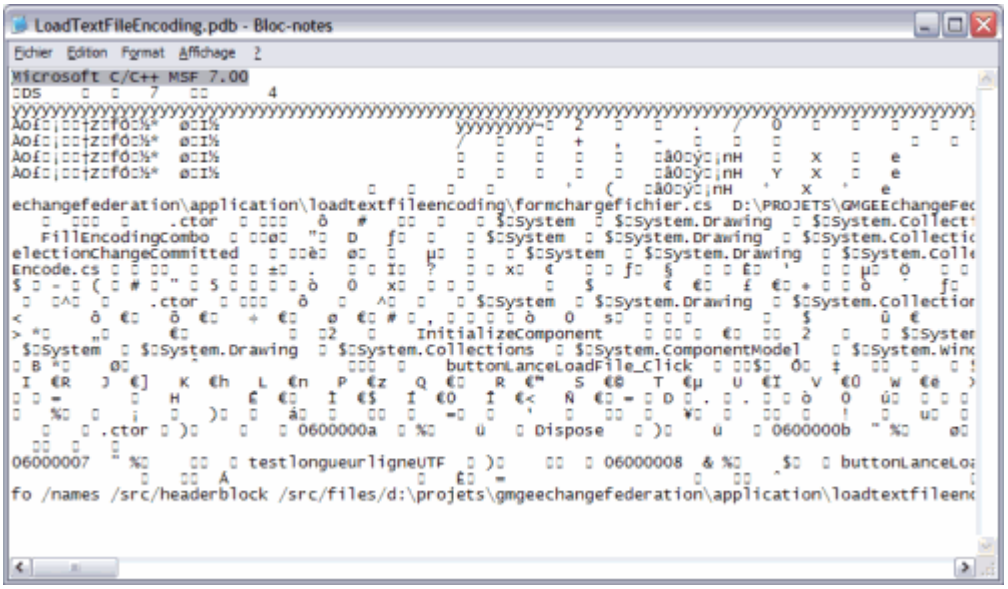


Nous prendrons Visual Studio .NET pour référence. Ainsi lorsque l'on compile un projet avec l'option "Debug", le compilateur crée un fichier PDB :

- **PDB Files**

Ce fichier PDB (Program DataBase) permet de conserver les informations et les symboles utilisés lors de cette compilation. Ce fichier est utilisé par Visual Studio .NET afin de surveiller les valeurs de variables utilisées dans le développement. Il contient aussi les références vers les points d'arrêt utilisés lors du développement.

Ce fichier PDB est un fichier binaire qui peut, lorsqu'il est ouvert avec un éditeur, nous permettre de voir les références vers les fichiers sources, les classes utilisées, les noms des variables utilisées, ...



Ainsi, ce choix de compilation va indiquer au FrameWork que celui-ci doit charger la classe "Debug" afin de suivre l'exécution du programme. Cette classe contient tout le nécessaire permettant de savoir à tout instant les valeurs des variables du code, les états de ces variables. Elle permet aussi de suivre l'exécution et de stopper celle-ci dans le cas où on se trouve sur un point d'arrêt.

Le fichier PDB est aussi très utile pour la décompilation. En effet, le fichier PDB contient tous les noms des variables locales. Celles-ci ne sont pas conservées lors de la compilation en Release, de ce fait, si on décompile la version Release, le décompilateur donnera des noms de variable du type "Text1" ou "TextBox1" au lieu de prendre celui que le développeur a éventuellement donné "MonText" ou "MonTextBox".

Voyons maintenant comment identifier le mode de compilation utilisé pour une DLL ou un EXE que l'on a.

Comment savoir l'option utilisée

Notre situation type est la suivante, nous avons une DLL (il en serait de même pour un EXE) dont nous ne connaissons pas l'origine, nous ne savons pas non plus comment celle-ci a été compilée et il est indispensable de le savoir avant de l'utiliser ou la placer sur un serveur de production.

Ceci se fait facilement avec un outil fourni avec le SDK du FrameWork .NET (et donc avec Visual Studio .NET):

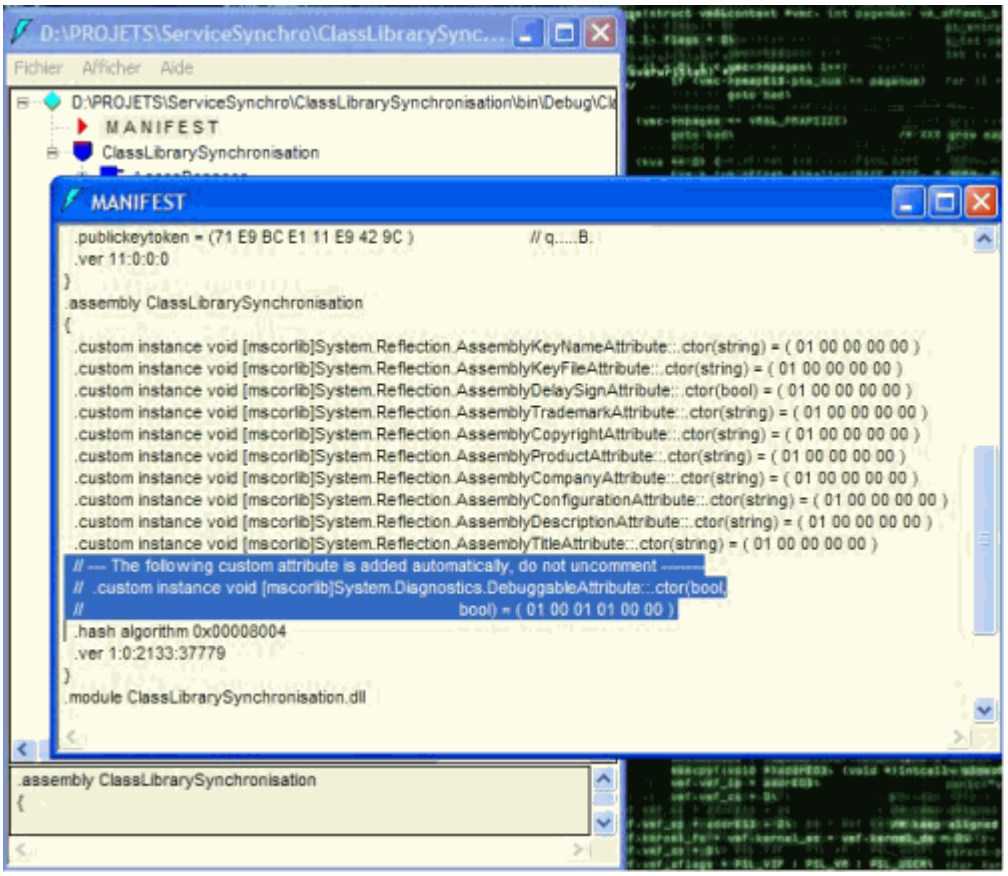
- **MSIL Disassembler (Ildasm.exe) (FR)**

ILDASM est l'inverse de l'**Assembleur MSIL (Ilasm.exe) (FR)** (qui est fourni directement avec le FrameWork .NET). ILDASM permet d'extraire le fichier Texte MSIL à partir d'une DLL ou d'un exécutable .NET.

Il nous faut donc lancer la commande suivante :

- CheminDuSDK\v1.1\Bin>ildasm.exe CheminDeLaDLL\LaDll.dll

Donc nous avons dans l'écran suivant la capture de la sortie ILDASM sur une DLL en mode debug :



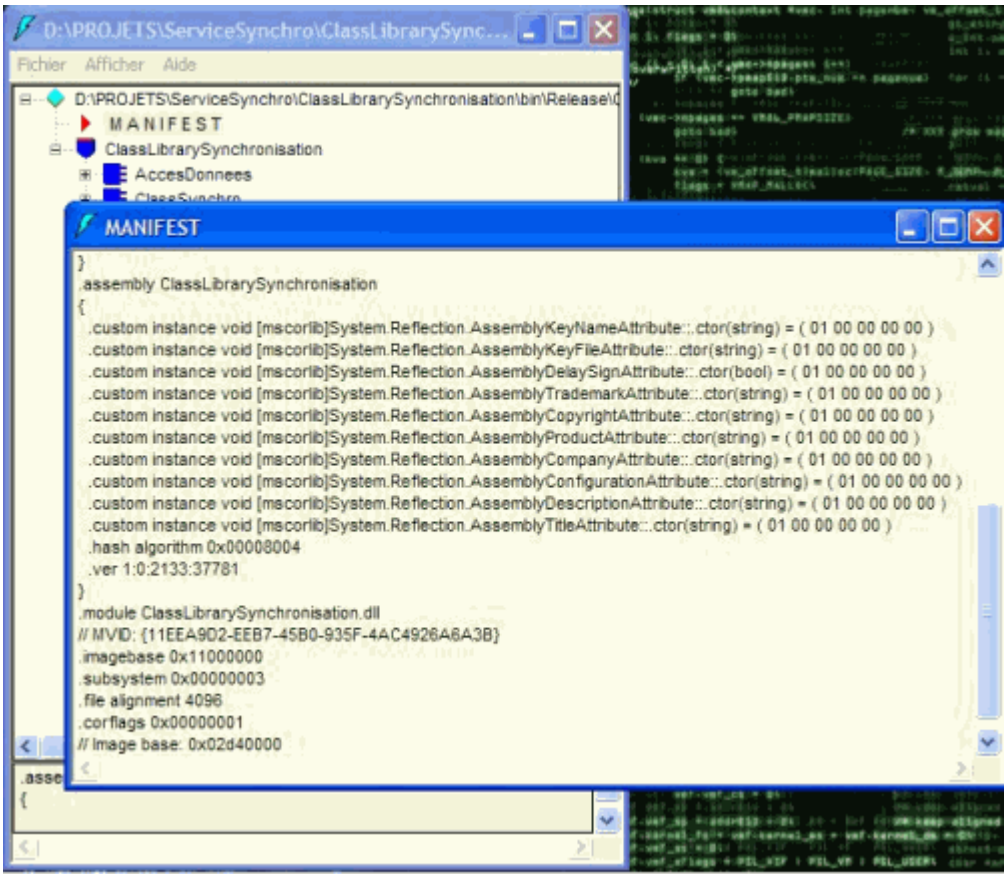
Dans cette sortie d'ILDASM, nous avons une première ligne "MANIFEST" symbolisée en rouge. Si nous ouvrons ce Manifest (en double-cliquant dessus), nous obtenons le contenu de la déclaration faite au FrameWork lors du chargement de cette DLL.

Au centre, nous voyons la ligne de déclaration de cette assemblée qui contient au bas un groupe de lignes concernant ce type de compilation :

```
.Assembly LeNomDeLaClasse
{
...

  // --- The following custom attribute is added automatically, do not uncomment -------
  //  .custom instance void [mscorlib]System.Diagnostics.DebuggableAttribute::.ctor(bool,
  //                                                                                 bool) = ( 01 00 01 01 00 00 )
...
}
```

Si on ouvre cette même DLL, cette fois compilée en mode Release, dans ILDASM et que l'on regarde le MANIFEST, ces lignes ne sont alors pas présentes :



Vous avez donc une méthode rapide pour détecter le mode utilisé pour la compilation d'un programme .NET, ceci quelque soit son langage de développement, car on teste le fichier MSIL.

Conclusion

Maintenant, vous en savez un peu plus sur ces modes de compilation qui peuvent nous paraître un peu triviaux de premier abord. La compréhension de la différence de chacun permet de livrer à vos éventuels clients des versions finales qui ne pourront pas être remis en cause sur cette partie. Il est en effet dommage de faire tomber les performances d'un outil juste du fait d'une compilation en mode Debug.

Si vous souhaitez en savoir plus, je vous invite à consulter les liens suivant :

- **PDB Files (US)**
- **MSIL Disassembler (Ildasm.exe) (FR)**
- **Assembleur MSIL (Ilasm.exe) (FR)**
- **Débogueur Runtime (Cordbg.exe) (FR)**
- **Project Settings for a C# Debug Configuration (US)**
- **Project Settings for a Visual Basic Debug Configuration (US)**
- **Inside the .NET Application (US)**
- **Debugging (US)**
- **Le Débugueur sous Visual Studio .NET (FR)**
- **Le Débug en ASP.NET (FR)**
- **How to Debug at design Time (FR)**
- **Déploiement asp.net - Debug ou Release ? (FR)**
- **Chapter 6 — Improving ASP.NET Performance (US)**
- **Using Assemblies in Microsoft .NET and C# (US)**

En vous souhaitant de bons projets de développement.

Romelard Fabrice (alias F___)
Consultant Technique **ILEM**