

Utiliser des classes C-Sharp dans des pages ASP

Principe

L'interopérabilité entre ASP et Dot.Net va être réalisée en passant par COM ; en effet tous les objets (ActiveX) utilisés dans les pages ASP sont des objets COM et le framework .NET supporte en standard des mécanismes d'exposition à COM simples à mettre en œuvre.

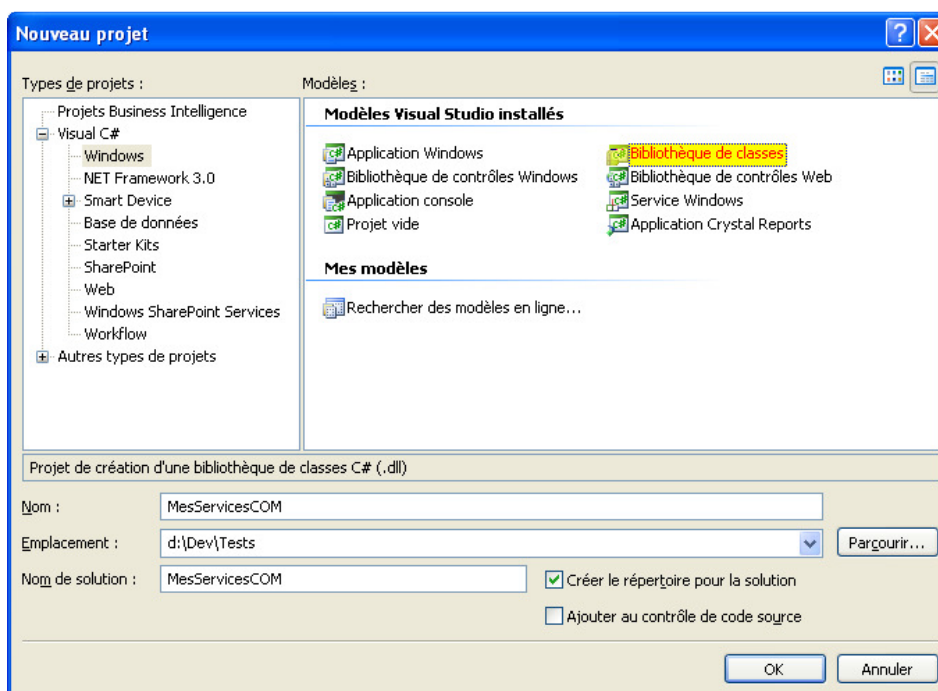
Détail important

L'interopérabilité entre .NET et COM passe par la publication d'interfaces COM implémentée par des classes .NET pour lesquelles le framework va créer dynamiquement un CCW (COM Callable Wrapper : c'est-à-dire une CoClass COM et sa factory qui permettent d'obtenir en COM des instances d'objets implémentant l'interface publiée).

Cela signifie que **les classes exposées à COM ne le sont qu'au travers de l'interface publiée** (seules les méthodes de l'interface pourront être appelées) et qu'elles seront **instanciées** par le CCW (donc **uniquement avec un constructeur par défaut**, sans paramètres).

Développement de la classe CSharp

Bien qu'un EXE puisse aussi exposer ses classes à COM, on choisira de préférence de créer un projet de type « Bibliothèque de classes ».



Une fois le projet créé, il faut commencer par définir notre interface. Pour l'exemple, nous allons mettre ici à profit les classes de traitement de texte de .NET pour proposer un service de transposition de chaîne d'un jeu de caractère à un autre.

```

namespace MesServicesCOM
{
    public interface ITranscoder
    {
        string Transcode(string originalText, int originalCp, int destinationCp);
    }
}

```

Puis nous allons l'exposer à COM avec l'attribut ComVisible, défini dans le namespace System.Runtime.InteropServices.

```

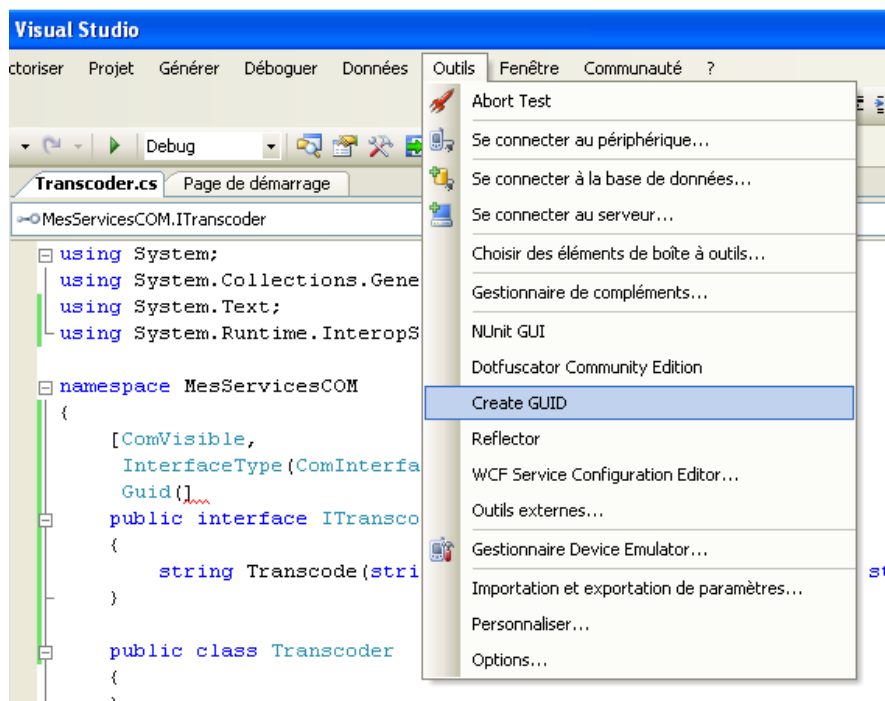
using System.Runtime.InteropServices;

namespace MesServicesCOM
{
    [ComVisible(true)]
    public interface ITranscoder
    {
        string Transcode(string originalText, int originalCp, int destinationCp);
    }
}

```

Deux attributs optionnels supplémentaires permettent respectivement de fixer l'IID (le GUID qui identifie l'interface au niveau de COM ; auto-généré si on ne le précise pas) et le type d'interface (Dual par défaut).

Pour générer un identifiant unique pour l'IID, vous pouvez utiliser l'outil « Create GUID » dans le menu Outils de Visual Studio (pensez à supprimer les accolades lorsque vous collez le GUID).



Types d'interfaces COM

COM supporte deux modes d'utilisation d'une interface : soit l'appel direct depuis un programme qui connaît sa définition et place les bons appels aux bonnes adresses avec les paramètres dans le bon ordre. Cela implique que les appels sont construits à la compilation. C'est le mécanisme de liaison anticipée ou liaison précoce (« early binding » en anglais). L'autre méthode est la liaison tardive (« late binding ») qui s'appuie sur la description de l'interface dans une bibliothèque de type (type library) COM. Celle-ci est recensée dans la base de registre lors de l'enregistrement du serveur. Pour supporter la liaison tardive une interface doit dériver d'IDispatch qui définit les fonctions d'auto-description d'une interface et de passage des paramètres par nom. Comme sont nom l'indique une interface Dual supporte deux modes d'appels, c'est-à-dire qu'elle permet les appels précompilés (plus performants) mais implémente également le support IDispatch (utilisé par les langages de script comme ASP).

Dans le cas de l'utilisation de la classe par une page ASP, le client étant un langage de script ActiveX, il ne connaît de toutes façon que la liaison tardive OLE Automation (OLE = IDispatch), donc nos interfaces devront toutes être Dual ou IDispatch.

Il reste à implémenter l'interface dans une classe,

```
public class Transcoder: ITranscoder
{
    #region ITranscoder Membres

    string ITranscoder.Transcode(string originalText, int originalCp, int destCp)
    {
        Encoding src = Encoding.GetEncoding(originalCp);
        Encoding dst = Encoding.GetEncoding(destCp);
        return dst.GetString(Encoding.Convert(src, dst, src.GetBytes(originalText)));
    }

    #endregion
}
```

puis à exposer celle-ci à COM avec l'attribut ComVisible et en indiquant par l'attribut ComDefaultInterface que l'interface principale de notre objet est ITranscoder.

L'attribut optionnel ClassInterface(ClassInterfaceType.None) indique au Framework de ne pas créer une interface implicite pour exposer les méthodes publique de la classe.

```
[ComVisible(true),
ComDefaultInterface(typeof(ITranscoder)),
ClassInterface(ClassInterfaceType.None),
Guid("FA781D27-227A-47e0-9178-E0239F28D3D8")]
public class Transcoder: ITranscoder {...}
```

Le code final de notre bibliothèque

```
using System;
using System.Text;
using System.Runtime.InteropServices;

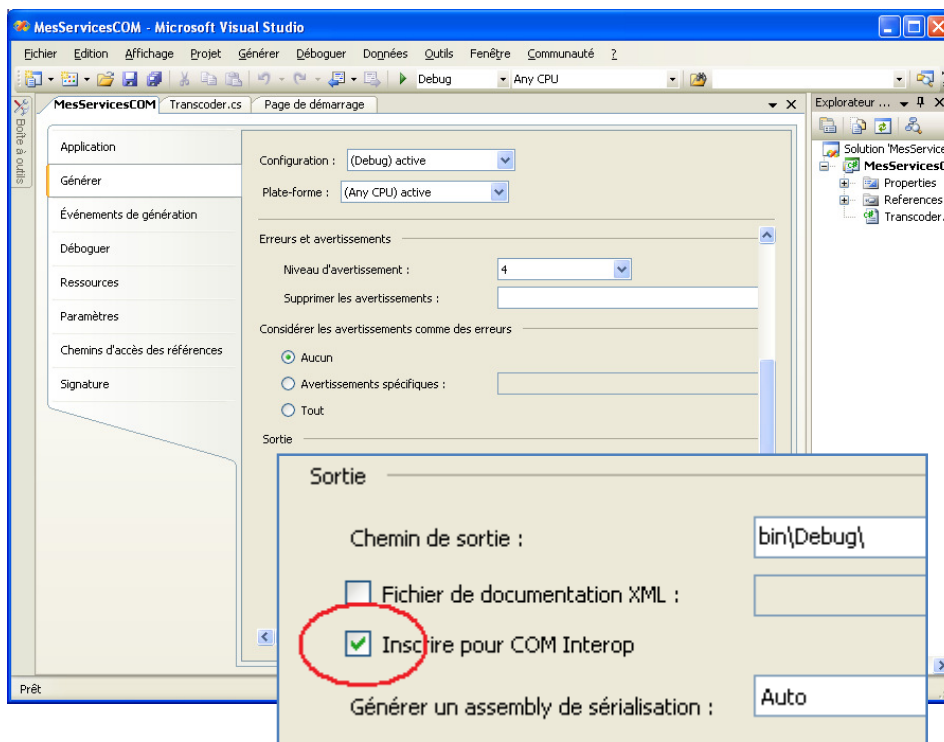
namespace MesServicesCOM
{
    [ComVisible(true),
    InterfaceType(ComInterfaceType.InterfaceIsDual),
    Guid("787A64FF-457D-408a-BCEE-ECC3D7E7B590")]
    public interface ITranscoder
    {
        string Transcode(string originalText, int originalCp, int destinationCp);
    }

    [ComVisible(true),
    ComDefaultInterface(typeof(ITranscoder)),
    ClassInterface(ClassInterfaceType.None),
    Guid("FA781D27-227A-47e0-9178-E0239F28D3D8")]
    public class Transcoder: ITranscoder
    {
        #region ITranscoder Membres

        string ITranscoder.Transcode(string originalText, int originalCp, int destCp)
        {
            Encoding src = Encoding.GetEncoding(originalCp);
            Encoding dst = Encoding.GetEncoding(destCp);
            return dst.GetString(Encoding.Convert(src, dst, src.GetBytes(originalText)));
        }

        #endregion
    }
}
```

Enfin, une option dans les pages de propriétés du projet indique à Visual Studio d'enregistrer notre classe pour COM au moment de la compilation.



Déploiement

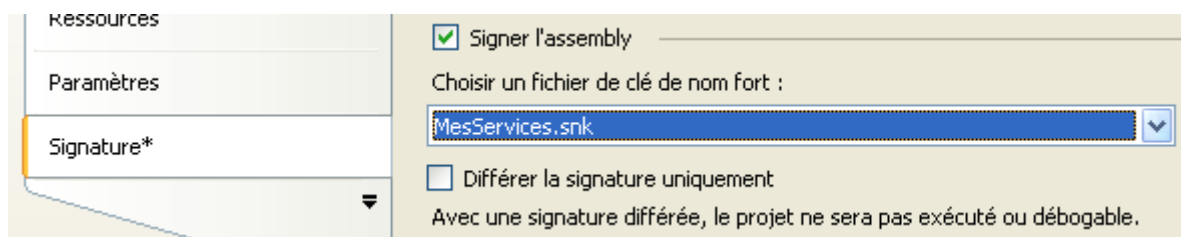
Pour que notre classe soit callable depuis COM il faut qu'elle soit enregistrée. Sur notre poste de développement Visual Studio s'en est chargé pour nous (voir ci-dessus) mais pour une utilisation sur notre serveur de production il faut quelques étapes supplémentaire:

Signature de l'assembly

En toute logique, une assembly qui est poussée en production doit être signée pour garantir l'intégrité du code et protéger contre les conflits de version. Pour cela nous allons générer un fichier de paires de clés publique/privée avec l'utilitaire SN du SDK .NET :

```
sn.exe -k "c:\dev\MesServices.snk"
```

Puis nous allons l'utiliser pour signer notre assembly (pages de propriétés du projet):



Cette opération est faite une fois pour toute en développement. Une fois recompilée avec cette option, notre DLL peut être déployée sur le(s) serveur(s) de production.

Enregistrement sur le serveur de production

L'enregistrement de notre DLL de service COM sur le serveur de production s'effectue par l'utilitaire regasm du Framework (dans C:\Windows\Microsoft.NET\Framework\v2.0.50727)

```
regasm.exe /tlb:"c:\dev\MesServicesCOM.tlb" /codebase "c:\dev\MesServicesCOM.dll" /silent
```

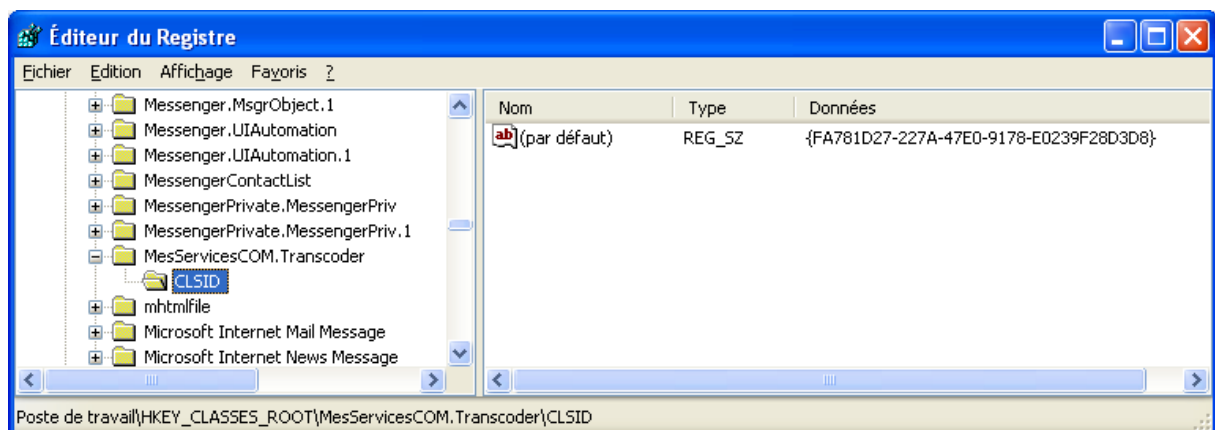
Ici les options retenues pour regasm indiquent où est située notre DLL (option /codebase).

Alternativement, la dll peut être enregistrée dans le GAC avec l'utilitaire GacUtil :

```
gacutil.exe -i "c:\dev\MesServicesCOM.dll"
```

Dans ce cas, il ne sera plus nécessaire d'indiquer à regasm le chemin de la dll.

On peut vérifier que l'objet est bien enregistré en ouvrant l'éditeur du registre :



La clé `HKEY_CLASSES_ROOT\MonAssembly.MaClasse` doit contenir une clé CLSID qui fait référence à l'entrée `HKEY_CLASSES_ROOT\CLSID\{guid de la CoClass}` qui indique quel exécutable ou DLL fournit le service.

Utilisation dans la page ASP

Une fois que notre objet est recensé sur le serveur, l'utiliser est aussi simple pour que tout autre ActiveX :

```
<%  
    dim Transcoder  
    set Transcoder = Server.CreateObject("MesServicesCOM.Transcoder")  
    dim Phrase  
    Phrase = "Une phrase où sont écrits les caractères accentués en code Windows"  
%>  
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">  
<html>  
  <head>  
    <title></title>  
  </head>  
  <body>  
    Avant: <span style="background-color:Yellow;">  
      <%= Phrase %></span>  
    <br />  
    Apres (Windows -> ASCII 7): <span style="background-color:Silver;">  
      <%= Transcoder.Transcode(Phrase, 1252, 20127) %></span>  
  </body>  
</html>
```

Voilà ! Il ne vous reste plus qu'à trouver les multiples usages que vous pouvez avoir de la puissance des classes du framework .NET dans vos sites ASP.