C# Création d'un objet COM

par JORDY Davide

Date de publication : 16 avril 2011

Dernière mise à jour :

Cet article va vous présenter comment il est facile de fabriquer et utiliser un objet COM en C#.

I - Définition

COM : Component Oject Model (Modèle objet de composant).
Il permet l'écriture de composant logiciels réutilisables par differentes applications. COM implemente un ensemble de fonctionnalités (interfaces) le rendant utilisable depuis n'importe quel langage de programmation supportant cette technologie (C, C++, C#, Asp, VBScript, Delphi,...). C'est un standard de communication entre composants logiciels propre à Microsoft, permettant de programmer des objets qui font abstraction du langage utilisé. Cette technologie est connu egalement sous le nom de activeX. Son predecesseur fut OLE (Object Linking and Embedding)

II - Introduction

Pour l'exemple nous allons développer un objet COM afin de faire communiquer via cookie crypté un site ASP classique et un site ASP.net hébergé sur le même serveur web.
L'objet COM aura pour but de crypter et de decrypter, la création/lecture du cookie reste à la charge des differents langages concernés.
Au lieu de faire communiquer ces 2 sites via http nous avons choisi la methode du cookie mais cette decision ne rentre pas dans le cadre de cet article.

Ce composant est développé en C# sous le framework 3.5 avec Visual studio 2008.

III - Création d'un objet COM

III-A - Partie Visual Studio 2008 et C# 3.5

1. Tout d'abord nous allons créer une nouvelle application dans Visual Studio 2008.
 a. Fichier -> Nouveau -> Projet -> Class library.
 b. Le nom donné: *MYCOM*.

2. Nous allons créer une Interface *ISecurity* regroupant toutes les fonctionnalités de securité de notre composant COM.
Pour l'exemple il s'agira que des methodes *Encrypt* et *Decrypt* et une Class *Security* pour l'implémentation.
 a. Créer une nouvelle Interface et Class :
  i. Ajouter une Interface et la nommer *ISecurity*
  ii. Ajouter une Class et la nommer *Security*.

Code Interface ISecurity

```
using System;
namespace MYCOM
{
    public interface ISecurity
    {
        /// <summary>
        /// Decrypts the specified content.
        /// </summary>
        /// <param name="content">The content.</param>
        /// <returns></returns>
        string Decrypt(string content);

        /// <summary>
        /// Encrypts the specified content.
        /// </summary>
        /// <param name="content">The content.</param>
        /// <returns></returns>
        string Encrypt(string content);
    }
}
```

Code Class ISecurity

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Security.Cryptography;
using System.Runtime.InteropServices;

namespace MYCOM
{
    [ClassInterface(ClassInterfaceType.None)]
    public class Security : MYCOM.ISecurity
    {
        private const string cryptoKey = "cryptoKey";

        // The Initialization Vector for the DES encryption routine
        private static readonly byte[] IV = new byte[8] { 240, 3, 45, 29, 0, 76, 173, 59 };

        /// <summary>
        /// Encrypts provided string parameter
        /// </summary>
        public string Encrypt(string content)
        {
            //Code d'encryption
        }

        /// <summary>
        /// Decrypts provided string parameter
        /// </summary>
        public string Decrypt(string content)
        {
            //Code de decryption
        }

    }
}
```

3. Ajouter la référence System.EnterpriseServices
http://msdn.microsoft.com/fr-fr/library/system.enterpriseservices.aspx

4. Création d'une signature à nom fort pour l'assembly afin de pouvoir exposer l'objet COM.
a. Projet *MYCOM* -> Propriété -> Signer -> cocher Signer l'assembly.
Nouvelle clé et remplir le nom de la clé et le mot de passe (*MYCOM*, *mycom*).

5. Cocher l'option Register for COM interop dans le menu Build dans les propriétés du projet *MYCOM*.

6. Ajouter 3 attributs dans la Class *AssemblyInfo* :

```
[assembly: ApplicationName("COM")] //Indique le nom de l'application
```

```
[assembly: ApplicationActivation(ActivationOption.Library)] //Indique l'option de l'activation.
```

```
[assembly: AssemblyKeyFile(@"C:\VS_Solution\MYSOLUTION
\MYCOM.pfx")] //Indique le chemin de la clé créée précédemment.
```

7. Mettre l'attribut COMVisible à true

```
[assembly: ComVisible(true)]
```

8. Ajouter un Identifiant pour que le projet soit exposé en objet COM

```
[assembly: Guid("6383a29e-fbf7-4c95-bbea-eb929e027f6a")]
```

9. Décorer la class *Security* avec

```
[ClassInterface(ClassInterfaceType.None)]
```

Value	Description
None	Using ClassInterfaceType.None is the only way to expose functionality through interfaces implemented explicitly by the class therefore it is recommended in our case.
Auto dispatch	Indicates that the class only supports late binding for COM clients.
Auto Dual	Indicates that a dual class interface is automatically generated for the class and exposed to COM. Type information is produced for the class interface and published in the type library. Using AutoDual is strongly discouraged because of the versioning limitations.

10. Compiler le projet afin de voir s'il y a des erreurs.

IV - Enregistrer l'objet Com sur le serveur

1. Copier les 3 fichiers générés par la compilation (*MYCOM.dll, MYCOM.pdb, MYCOM.tlb*) sur le serveur web.

2. Entrer cette ligne de commande ' Regasm /codebase /tlb:"C:\MYSolution \MYCOM\bin\Debug\MYCOM.tlb" " C:\MYSolution \MYCOM\bin \Debug\MYCOM.dll"

3. Enregistrer *MYCOM* dans le GAC (Global Assembly Cache) : gacutil /if "D:\ImportDLL\MYCOM.dll

V - Utilisation de MYCOM

1 - Coté asp 3.0

```
dim objCom
set objCom = Server.CreateObject("MYCOM.Security")

dim myString
myString = objCom.Encrypt("Ceci est un test !!!!")
Response.Write objCom.Decrypt(myString)
```

2 - Coté asp.net

Ajouter la référence *MYCOM* à la solution.

```
SMARTCOM.Security security = new SMARTCOM.Security();
  string myString = security.Encrypt("Hello MYCOM tu me recois.");
Label1.Text = security.Decrypt(myString);
```

Dans notre cas nous pourrons donc utiliser ce procédé pour faire communiquer un site asp classique avec un site asp.net via cookie crypté au lieu de passer par HTTP avec les paramètres en clair.

VI - Conclusion

Grâce à COM la communication entre plusieurs langages est devenu assez simple