Pilotage Microsoft® OFFICE (VB.NET)

J-M RABILLOUD

Remerciements

J'adresse ici tous mes remerciements à l'équipe de rédaction de "developpez.com" et tout particulièrement à Maxence Hubiche et David Pédehourcq pour le temps qu'ils ont bien voulu passer à la correction et à l'amélioration de cet article.

Dans cet article nous allons regarder comment communiquer avec les composants d'applications MS Office comme Word ou Excel. Il n'est pas possible d'être exhaustif dans ce type d'applications tant les possibilités sont nombreuses. Je me limiterais donc à deux exemples concrets qui nous permettront de voir le fonctionnement.

Les codes sont donnés en VB.NET et en C#, les applications Offices sont Excel 2000 et Word 2000.

Bonne lecture

Introduction

Dans les langages DotNet, il est possible de continuer d'utiliser des composants COM sans avoir à réécrire ceux-ci. De même, il est aussi possible d'utiliser des composants managés dans un client COM. L'ensemble de ces mécanismes est défini par le terme interop COM. Nous étudierons en détail celui-ci dans un autre article.

Généralités

Primary Interop Assembly (PIA)

Il y a des différences importantes entre la définition des types dans une bibliothèque de type d'un composant COM et celle d'un assembly .NET. Il y a la possibilité de créer un assembly gérant cette conversion. Celui-ci fonctionne alors comme une référence et peut être redistribuée avec votre application. Cet assembly particulier est défini sous le terme de 'Primary Interop Assembly'. Certains composants COM proposent un assembly pour leur utilisation. C'est par exemple le cas pour Office XP dont les PIA sont disponibles à cette adresse :

http://support.microsoft.com/default.aspx?kbid=328912

Dans ce cas vous pouvez directement les créer. Sinon il faut gérer cette création.

La plupart du temps celle-ci est transparente.

Dans VS.NET je crée un nouveau projet. Dans le menu 'projet' je choisis 'Ajouter une référence' et dans la boite de dialogue je sélectionne l'onglet COM. Là je choisis 'Microsoft Excel Object Library X.0' (X dépendant de la version ; 8 pour Excel 97 et 9 pour Excel 2000) et je valide. Dans les références de mon projet il existe alors deux références interop.excel et interop.office. Il y a eu création dans le répertoire de mon projet d'un excel.dll et d'un office.dll, ce sont les PIA susnommés.

Cela fonctionne à l'identique avec C#Builder.

Il existe d'autres outils de création de PIA que je ne détaillerai pas ici.

Règles du pilotage

Bien qu'il s'agisse souvent d'interaction entre l'application Office et votre application, le fonctionnement intrinsèque n'en reste pas moins un pilotage d'application et comme tel doit suivre un certain nombre de règles.

Gestion des références

Les applications Office sont des serveurs de composants COM de type Out-of-process. Ceci veut dire que lors de l'appel à des objets Excel ou Word, il y aura démarrage de l'application Office correspondante. Comme votre application est responsable de ce démarrage, elle se doit de gérer clairement le fonctionnement et la fermeture de l'application Office, c'est pour cela qu'on emploie le terme de pilotage.

Les techniques employées peuvent être assez diverses selon le but à atteindre mais en général :

- Une application ne doit pas créer plus d'une instance d'une application Office.
- Votre application doit détruire explicitement l'ensemble de ses références à des composants pilotés.

Transfert des données

C'est souvent le point qui pose problème. Si le PIA contient l'ensemble des conversions de types entre les types définis par le composant et les types du Framework DotNet, il n'en est pas de même pour les types des données utilisables par les deux. Dans l'ensemble, le problème est plus sensible pour les chaînes de caractères et Word que pour Excel pour qui tout est de type 'variant'. Vous le verrez, j'utilise pour ma part beaucoup le presse-papiers qui est un moyen fiable et simple pour échanger pour de gros volumes de données.

Délégation et fonctionnalités

Plus que des règles, il s'agit là de bon sens.

On est assez souvent tenté de récupérer des fonctionnalités d'application Office vers son application sans en utiliser le meilleur parti. Je vais prendre l'exemple du deuxième cas que nous traiterons, la correction orthographique.

Fondamentalement, il existe deux approches. Soit on rapproche le module de correction de Word vers l'application soit on envoie le texte à corriger vers Word.

En première analyse, le premier choix semble plus pertinent mais c'est une erreur. Certes dans ce cas, on évite deux transferts de données, mais on doit alors écrire une boite de dialogue de correction et gérer soit même la décomposition du texte en mot et/ou en phrases. Cela revient à refaire ce que Word sait déjà faire.

Exemples de pilotage

Piloter Excel depuis une application DotNet.

Ce premier exercice est un grand classique du pilotage Excel. Nous allons faire le tracé d'une fonction Y=f(X), ce tracé devant s'afficher dans notre application. Pour cela nous allons ajouter sur une feuille 3 zones de texte et un contrôle Picture. Une zone de texte (nommée txtFormule) servira à la saisie de la fonction. Comme règle implicite, la fonction devra suivre une syntaxe compréhensible par Excel, en utilisant X au lieu d'une référence de cellule. Par exemple Log(X)/X^2.

Les deux autres zones (nommées respectivement txtMini et txtMaxi) permettront de rentrer les bornes du domaine sur lequel on souhaite le tracé.

Un bouton cmdTrace déclenchera le tracé.

Je vous donne ici le code, nous en discuterons après.

Code VB.NET

```
Private Sub cmdTrace_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmdTrace.Click

      'Appel d'Excel et attribution des variables
      Dim AppExcel As New Excel.Application
      Dim Classeur As Excel.Workbook =
AppExcel.Workbooks.Add(Excel.XlWBATemplate.xlWBATWorksheet)
      Dim Feuille As Excel.Worksheet = Classeur.ActiveSheet
      'Nom des séries
      Feuille.Cells(1, 1).value = "X"
      Feuille.Cells(1, 2).value = Me.txtFormule.Text
      Feuille.Cells(2, 1).value = CDbl(txtMini.Text)
      'création d'un tableau de 100 valeurs allant de min à max par
pas fixe
      Feuille.Cells(2,
1).resize(101).DataSeries(Excel.XlRowCol.xlColumns,
Excel.XlDataSeriesType.xlDataSeriesLinear, , (CDbl(txtMaxi.Text) -
CDbl(txtMini.Text)) / 100)
      Feuille.Cells(2, 2).resize(101).formulalocal = Replace("=" &
txtFormule.Text, "X", "LC(-1)", 1, -1, CompareMethod.Text)
      Dim MaPlage As Excel.Range
      'cherche les cellules en erreur
```

```
      MaPlage = Feuille.Cells(1,
2).resize(102).SpecialCells(Excel.XlCellType.xlCellTypeFormulas,
Excel.XlSpecialCellsValue.xlErrors)
      If Not MaPlage Is Nothing Then
         'il y a des erreurs
         If MaPlage.Count = 101 Then
            MessageBox.Show("Erreur dans la formule", "ERREUR",
MessageBoxButtons.OK, MessageBoxIcon.Error)
            Classeur.Close(False)
            MaPlage = Nothing
            Feuille = Nothing
            Classeur = Nothing
            AppExcel.Quit()
            AppExcel = Nothing
            Exit Sub
         End If
      Else
         MaPlage.Value = "#N/A"
      End If
      'Tracé de la courbe
      Dim Graphe As Excel.Chart =
Classeur.Sheets.Add(Classeur.Sheets(1), , 1,
Excel.XlWBATemplate.xlWBATChart)
      Graphe.ChartType = Excel.XlChartType.xlXYScatterLinesNoMarkers
Graphe.SetSourceData(Feuille.Cells(1, 1).Resize(102, 2),
Excel.XlRowCol.xlColumns)
      'récupération de la courbe dans le presse-papiers
      Graphe.CopyPicture(Excel.XlPictureAppearance.xlScreen,
Excel.XlCopyPictureFormat.xlBitmap,
Excel.XlPictureAppearance.xlScreen)
      'récupération du presse-papiers dans le picturebox
      Dim data As IDataObject
      data = Clipboard.GetDataObject()
      Dim bmap As Bitmap
      If data.GetDataPresent(GetType(System.Drawing.Bitmap)) Then
         bmap = CType(data.GetData(GetType(System.Drawing.Bitmap)),
Bitmap)
         Me.PictureBox1.Image = bmap
         Me.PictureBox1.SizeMode = PictureBoxSizeMode.StretchImage
      End If
      'Fermeture propre d'Excel
      Classeur.Close(False)
      MaPlage = Nothing
      Feuille = Nothing
      Classeur = Nothing
      AppExcel.Quit()
      AppExcel = Nothing

End Sub
```

```
Private Sub txtMini_KeyPress(ByVal sender As Object, ByVal e As
System.Windows.Forms.KeyPressEventArgs) Handles txtMini.KeyPress
      Dim c As Char
      c = e.KeyChar
      If Not (Char.IsDigit(c) Or Char.IsControl(c)) Then
           e.Handled = True
      End If
End Sub

Private Sub txtMaxi_KeyPress(ByVal sender As Object, ByVal e As
System.Windows.Forms.KeyPressEventArgs) Handles txtMaxi.KeyPress
      Dim c As Char
      c = e.KeyChar
      If Not (Char.IsDigit(c) Or Char.IsControl(c)) Then
          e.Handled = True
      End If
End Sub
```

Il est bien entendu que dans mes références, j'ai ajouté mon Excel.dll.

Vous noterez tout d'abord que je n'ai pas gérer un certain nombres de cas qui n'apporte rien à l'exemple mais qui doivent être normalement gérés (égalité mini / maxi, maxi>mini, etc..)

Bien que cela ne traite pas directement le sujet, j'ai ajouté la procédure de gestion de l'événement KeyPress de mes zones de texte mini / maxi afin de restreindre la saisie à des valeurs numériques.

Comme le code n'est pas toujours très explicite, pour qui n'est pas habitué au pilotage Excel, je vais expliciter quelques parties.

```
Feuille.Cells(2, 1).resize(101).DataSeries(Excel.XlRowCol.xlColumns,
Excel.XlDataSeriesType.xlDataSeriesLinear, , (CDbl(txtMaxi.Text) -
CDbl(txtMini.Text)) / 100)
Feuille.Cells(2, 2).resize(101).formulalocal = Replace("=" &
txtFormule.Text, "X", "LC(-1)", 1, -1, CompareMethod.Text)
```

Dans cette partie je crée le tableau qui va me permettre le tracé. La première ligne me crée une série de valeurs dont le premier point est la valeur mini, contenant 101 valeurs jusqu'à la valeur maxi, le pas étant fixe (maxi-mini/100).

Notez qu'il y a utilisation de la fonction Replace pour convertir X dans le système de référence d'Excel.

```
      MaPlage = Feuille.Cells(1,
2).resize(102).SpecialCells(Excel.XlCellType.xlCellTypeFormulas,
Excel.XlSpecialCellsValue.xlErrors)
```

Cette ligne crée un objet Range contenant les cellules en erreur. J'ai besoin de cette récupération pour voir si Excel sait interpréter la formule saisie. Si toutes les cellules sont en erreur, Excel ne peut pas interpréter la formule et il y a sortie de la procédure. Sinon, l'exécution se poursuit.

Je dois alors passer un Bitmap de ma courbe tracée vers mon application. Pour copier une courbe comme Bitmap, il existe la méthode CopyPicture d'Excel, mais pour récupérer l'image du presse-papiers ce n'est pas immédiat.

```
      Dim data As IDataObject
      data = Clipboard.GetDataObject()
      Dim bmap As Bitmap
      If data.GetDataPresent(GetType(System.Drawing.Bitmap)) Then
         bmap = CType(data.GetData(GetType(System.Drawing.Bitmap)),
Bitmap)
         Me.PictureBox1.Image = bmap
```

J'utilise l'interface IDataObject pour récupérer le contenu du presse-papiers. Celle-ci permet d'interpréter les données du presse-papiers dans différents formats. Je peux alors tester le presse-papiers et procéder à la récupération.

Correction orthographique à l'aide de Word

Dans ce deuxième exemple nous allons utiliser la correction orthographique de Word pour une zone de texte de mon application. Le code en est très simple.

Code VB.NET

```
Private Sub cmdCorrige_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles cmdCorrige.Click
      Dim AppWord As New Word.Application
      Dim MonDoc As Word.DocumentClass = AppWord.Documents.Add()
      Clipboard.SetDataObject(Me.TextBox1.Text)
      MonDoc.Content.Paste()
      MonDoc.Activate()
      MonDoc.CheckSpelling()
      MonDoc.Content.Copy()
      If Clipboard.GetDataObject.GetDataPresent(DataFormats.Text)
Then
            Me.TextBox1.Text =
CType(Clipboard.GetDataObject.GetData(DataFormats.Text), String)
      End If
      MonDoc.Saved = True
      MonDoc.Close(False)
      AppWord.Quit()
      MonDoc = Nothing
      AppWord = Nothing
End Sub
```

Comme je vous l'ai dis plus avant, j'utilise plutôt toutes les fonctionnalités de Word. Si j'avais voulu, j'aurais pu créer un code beaucoup plus complexe afin de ne pas transférer le Texte vers Word.

Intercepter les évènements office

Il est possible d'intercepter les événements Excel dans votre application. En VB.NET deux techniques sont accessibles.

On crée une variable globale avec le mot clé WithEvents, les événements sont alors directement accessibles dans l'IDE.

```
   Private WithEvents AppExcel As New Excel.Application

   Private Sub AppExcel_SheetSelectionChange(ByVal Sh As Object,
ByVal Target As Excel.Range) Handles AppExcel.SheetSelectionChange
      .....
   End Sub
```

Ou alors on passe par la gestion standard des évènements

```
   Private Sub Button1_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles Button1.Click
   Dim monExcel As New Excel.Application
   Dim monClasseur As Excel.Workbook = monExcel.Workbooks.Add
   AddHandler monClasseur.BeforePrint, New
Excel.AppEvents_WorkbookBeforePrintEventHandler(AddressOf
BeforeBookPrint)

        End Sub

        Private Sub BeforeBookPrint(ByRef Cancel As Boolean)

        End Sub
```

Voilà. J'espère que ce bref aperçu vous sera utile. N'hésitez pas à poser vos questions sur le forum DotNet de developpez : http://www.developpez.net/forums/viewforum.php?f=49

Bonne programmation.