Early ou Late Binding

par (site) (Blog)

Date de publication :

Dernière mise à jour :

"Les fiches VBA" sont une série de petits tutoriels rapides regroupant les informations utiles sur un sujet donné. Elles concernent le langage VBA dans son ensemble et ne sont pas spécifiques à un logiciel donné. Vous y trouverez donc des informations valables pour Access, Excel, Word, Outlook, PowerPoint, ... Bonne lecture !

I - Qu'abordons-nous ici ?

L'objet spécifique de cette fiche est de permettre de comprendre et de différencier le Late Binding (Liaison tardive) de l'Early Binding (Liaison anticipée). Ces deux techniques vous permettront de sortir de certaines situation périlleuses dans vos développements de solutions pour des parcs hétérogènes.

II - Late ou Early Binding ???

II-A - Binding... kèsako ?

Le compilateur Visual Basic exécute un processus appelé "Liaison" lorsqu'un objet est assigné à une variable d'objet, autrement dit, lorsqu'il lit une ligne telle que celle-ci :

Exemple PowerPoint

```
Set oSlide = Presentations("Présentation1").Slides(1)
```

II-B - Early Binding

On parle d'Early Binding (ou Liaison anticipée) lorsque la variable objet est déclarée dans un type objet précis.

Ainsi, les exemples suivant permettront une liaison anticipée :

Exemple Word

```
Dim bkm as Word.BookMark  'Je veux un objet BookMark de la bibliothèque Word
```

Exemple DAO

```
Dim rst as DAO.Recordset  'Je veux un objet Recordset de la bibliothèque DAO
```

Exemple Excel

```
Dim rng as Excel.Range  'Je veux un objet Range de la bibliothèque Excel
```

exemple Access

```
Dim frm as Access.Form  'Je veux un objet Form de la bibliothèque Access
```

Dans la mesure du possible, il est préférable d'utiliser la liaison anticipée (early binding), car ce genre de liaison permet au compilateur d'allouer de la mémoire et de réaliser d'autres optimisations importantes avant même qu'une application ne s'exécute. Ces optimisations font que les objets en liaison anticipée sont sensiblement plus rapides que ceux en liaison tardives.

D'autre part, le fait d'utiliser des variable objet en liaison anticipée rend votre code beaucoup plus lisible, et vous facilitera énormément la maintenance, puisque cette solution expose les membres (propriétés, méthodes, ...) de vos objets au fur et à mesure de l'écriture de votre code (complétion - CTRL+Espace), et vous donne accès à l'aide dynamique sur la syntaxe.

Tous ces avantages font qu'il est préférable de toujours utiliser la liaison anticipée (Early Binding) !

Bien évidemment, pour faire une liaison anticipée, il faut que la bibliothèque concernée soit ajoutée aux références du projet en cours (Outils/références...)

II-C - Late Binding

On parle de Late Binding (Laison tardive) lorsque la variable objet est dclarée en tant qu'Object.

Ainsi, les déclaration suivantes impliquent une déclaration tardive :

exemple quelconque n°1
```
Dim bkm as Object  'Je veux un objet
```

exemple quelconque n°2
```
Dim rst as Object  'Je veux un objet
```

exemple quelconque n°3
```
Dim rng as Object  'Je veux un objet
```

exemple quelconque n°4
```
Dim frm as Object  'Je veux un objet
```

Comme vous le constatez, aucune information n'est fournie au compilateur sur cette ligne. Il sait seulement qu'il va avoir un objet. Mais quel objet ? Issu de quelle classe ? Cela, il l'ignore.

Tous les avantages de l'Early Binding disparaissent donc. Pas de complétion, pas d'aide interactive, et la compilation ne peut exécuter les optimisations du code auxquelle on pourrait s'attendre, ce qui résulte en un code plus lent, forcément.

Comment affecter une variable Object ?

exemple Word
```
Dim oApp as Object        'Je veux un objet
set oDApp = CreateObject("Word.Application")     'Créer une nouvelle instance de la classe
Application de Word
            'Et pointer la variable oAPP vers cet objet
```

exemple PowerPoint
```
Dim oApp as Object  'Je veux un objet
set oApp = GetObject(, "PowerPoint.Application") 'récupère une instance existante de l'application
powerpoint
            'Et pointe la variable oApp vers cet objet
```

II-C-1 - CreateObjet ou GetObject ?

II-C-1-a - GetObject

GetObject est une fonction particulièrement intéressante, car elle permet de renvoyer une référence à un objet fourni par un composant ActiveX.

Examinons d'abord la syntaxe

En provenance de l'aide Office 2007 :

```
GetObject([pathname] [, class])

'### La syntaxe de la fonction GetObject comprend les arguments nommés suivants :
'### Élément Description
'pathname  Facultatif. Variable de type Variant (String). Chemin d'accès complet et nom du fichier
 contenant l'objet à extraire.
'   Si l'argument pathname est omis, l'argument class est obligatoire.
'class   Facultatif. Variable de type Variant (String). Chaîne représentant la classe de l'objet.
```

Donc, cette fonction permet de récupérer la référence à un objet.

Seul le PathName est mentionné :

```
Sub proc_PremierGet()
    Dim x As Object
    Set x = GetObject("d:\oula.doc")
    Debug.Print TypeName(x)
    Set x = Nothing
End Sub
'Affiche ceci dans la fenêtre d'exécution (CTRL+G) :
'
'   Document
```

Le PathName est la Class sont mentionnés :

```
Sub proc_DeuxiemeGet()
    Dim x As Object
    Set x = GetObject("d:\oula.doc","Word.Document")
    Debug.Print TypeName(x)
    Set x = Nothing
End Sub
'Affiche ceci dans la fenêtre d'exécution (CTRL+G) :
'
'   Document
```

Seule la Class est mentionnée :

```
Sub proc_TroisiemeGet()
    Dim x As Object
    Set x = GetObject(,"Word.Application")
    Debug.Print TypeName(x)
    Set x = Nothing
End Sub
'Affiche ceci dans la fenêtre d'exécution (CTRL+G) :
'
'   Application
```

Utilisez la fonction GetObject lorsqu'il existe une instance en cours de l'objet ou si vous souhaitez créer l'objet avec un fichier déjà chargé. S'il n'existe aucune instance en cours et si vous ne voulez pas démarrer l'objet en chargeant un fichier, utilisez la fonction CreateObject

II-C-1-b - CreateObject

III - Remerciements

Je tiens à remercier le travail d'équipe de tous les bénévoles qui participent à ce site, et, dans le cadre des observations et corrections apportées à ce tutoriel, mes remerciements vont tout particulièrement à

Correction

- xxx

Commentaires

- xxx