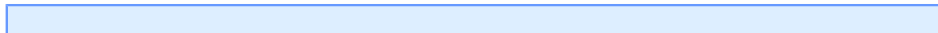


# Tutoriel 27 : Gérer des contrôles comme des variables objets

par Gilbert Miralles ([gilmir.developpez.com](http://gilmir.developpez.com))

Date de publication : Lundi 11 mars 2003

Dernière mise à jour : Lundi 4 février 2008



I - Introduction

II - Un sous programme de validation de zone de texte

III - Déclarer des variables objets

IV - Les déclarations implicites & explicites

Prochain tutoriel :

## I - Introduction

Vous pouvez non seulement utiliser des objets comme variables de tableau de contrôles, mais aussi passer ces variables objets et ces tableaux d'objets comme paramètres d'un sous programmes.

Cela peut paraître obscur pour certains, mais le but de cette méthode est tout simplement de gagner du temps à l'exécution de votre application.

Combien de fois j'ai lu dans les colonnes de certains sites ou dans les exposés de certains programmeurs que le Visual basic était lent...je ne veux en aucun cas me faire le défenseur du VB, mais je me pose une question, à savoir si toutes les possibilités d'exécution du code ont vraiment été examinées.

Or je constate que bien souvent on écrit un programme comme on le construit lors d'un premier jet, et qu'il est parfois rare que l'on se retourne sur sa réalisation pour lui apporter des modifications.

Je ne veux pas généraliser aussi je vous laisse toute liberté pour analyser le bien fondé de mes observations et vous invite à réaliser l'application que je vous propose d'écrire pour argumenter mon petit exposé.

Prenons par exemple la réalisation d'une feuille qui doit contenir une trentaine de zones de textes, chacune devant comporter un type de données bien spécifiques.

Certaines ne pourront accepter que des caractères numériques, d'autres des données alphabétiques, et certaines peuvent être limités en nombre de caractères.

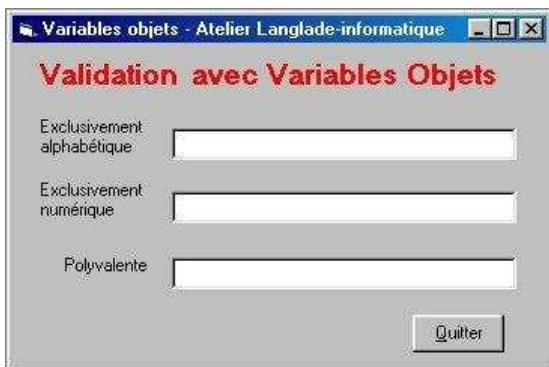
Cette configuration devrait normalement nécessiter trois sous programmes distincts, un pour chaque type de zone de texte.

## II - Un sous programme de validation de zone de texte

Nous allons écrire un sous programme de validation de texte pour démontrer le bien fondé de cette explication, ce qui nous permettra de savoir quel est le type de données requis par chaque zone de texte ainsi que la longueur maximale des données saisies.

Ce qui obligera le programme à ignorer toute saisie qui n'obéirait pas aux règles établies par le concepteur du programme.

passons aux choses sérieuses et créez un nouveau projet et positionnez sur l'interface les objets affichés ci-après.



Il ne nous reste plus qu'à rentrer le code pour faire fonctionner notre application.

Télécharger l'exo si vous ne voulez pas faire l'effort de saisir le texte.

[Télécharger l'exo](#)

**je vous laisse le soin de positionner sur la feuille les objets suivants :**

- **1 Label** - Label1 avec Propriété.Caption = "Validation avec Variables Objets"
- **3 autres Labels** pour les autres étiquettes, à savoir : Label2, Label3, Label4 avec

Description	Propriété	Valeur
Zone de texte exclusivement alphabétique	Name	txtValider
	Index	0
	Tag	A12
Zone de texte exclusivement numérique	Name	txtValider
	Index	1
	Tag	N5
Zone de texte polyvalente	Name	txtValider
	Index	2
	Tag	*4

Vous pouvez modifier la valeur de la propriété Tag qui correspond au nombre de caractères que vous avez définis dans les champs de saisie, mais assurez vous que les données affectées à ces propriétés soient écrites exclusivement en majuscules.

Nous positionnons ensuite 3 TextBox, Text1, TExt2 et Text3, les 3 textBox auront la même propriété Name "txtValider" et seront indexés de Index0 à Index2.

Il ne nous reste plus qu'à insérer le code devant faire fonctionner notre application.

```

Option Explicit                                'Nous forçons VB à déclarer les variables
Private Sub ValiderTouchePress(txtControle As TextBox, nKeyAscii As Integer)

Dim sMaxLength As String
    
```

```
Dim sKey As String * 1
'
If nKeyAscii < 32 Or nKeyAscii > 126 Then Exit Sub
sMaxLength = Right$(txtControle.Tag, Len(txtControle.Tag) - 1)
'
If Len(txtControle.Text) >= Val(sMaxLength) Then
    Beep '
    nKeyAscii = 0
    Exit Sub
End If
'
Select Case Left$(txtControle.Tag, 1)
Case "A"
    sKey = UCase(Chr$(nKeyAscii))
    '
    If Asc(sKey) < 65 Or Asc(sKey) > 90 Then
        Beep
        nKeyAscii = 0
        Exit Sub
    End If
'
Case "N"
    If nKeyAscii < 48 Or nKeyAscii > 57 Then
        Beep
        nKeyAscii = 0
        Exit Sub
    End If
End Select
'
End Sub
```

Nous insérons également le code devant provoquer l'événement lorsque nous insérons du texte dans les TextBox

```
Private Sub txtValider_KeyPress(Index As Integer, KeyAscii As Integer)
    ValiderTouchePress txtValider(Index), KeyAscii
End Sub
```

Nous constatons que la première "TextBox" n'accepte que des caractères alphabétiques, la seconde accepte des caractères numériques et la troisième qui est polyvalente accepte les deux options précédentes sans aucun problème dans la limite du nombre de caractères que nous avons fixé!

### III - Déclarer des variables objets


Pour pouvoir déclarer des variables objets comme par exemple les zones de texte que nous venons d'étudier, il faut écrire :

```
Dim txtControle As TextBox
```

Lorsque l'on déclare des variables objet explicitement, le code est plus efficace, plus facile à déboguer et s'exécute plus rapidement.

Voici la liste type de variables objet que Visual Basic accepte.

CheckBox	ComboBox	CommandButton	MDIForm
Data	DirListBox	DriveListBox	FileListBox
Grid	Frame	HScrollBar	Image
Label	Line	ListBox	Menu
OptionButton	OLE	PictureBox	Shape
TextBox	Timer	VScrollBar	Form

 *Ce sont les objets standard*

*Déclarations explicites et implicites ? Qu'est sa quo ?*

## IV - Les déclarations implicites & explicites

### Les déclarations implicites

Rendent votre code plus difficile à comprendre. Réduisent les chances de Visual Basic à piéger les erreurs Sont beaucoup plus lentes

### Les déclarations explicites

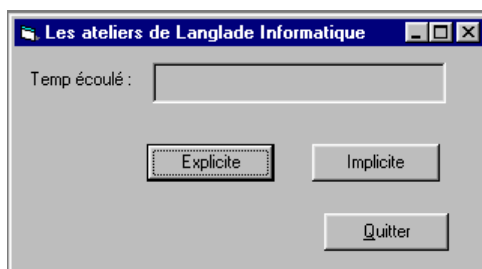
Facilitent le débogage

Lorsque l'on déclare un objet par un type spécifique, comme une zone de texte par exemple, Visual Basic sait quelles propriétés sont valables pour cette dernière

**VISUAL BASIC** ne vérifie les propriétés d'un objet contrôle qu'en phase d'exécution, tandis que dans le cas de contrôles explicites, il annoncera les erreurs de propriétés au moment de la compilation.

Nous allons essayer de comparer les deux possibilités pour constater si vraiment la différence d'exécution est flagrante.

Nous lançons un nouveau projet et nous plaçons sur la feuille les contrôles suivants :



Un Label "Temp écoulé"

Un second Label (Contrôle étiquette)

Le contrôle étiquette aura comme propriété Name : lblTemps

Trois boutons

Les boutons de commandes auront comme propriété Name: cmdExplicite, cmdImplicite, cmdQuitter

Il ne nous reste plus qu'à saisir le code devant faire fonctionner notre application

```
Private Sub cmdExplicite_Click()  
Dim VarTemps As Variant  
Dim nIndex As Integer  
|  
VarTemps = Now  
|  
For nIndex = 1 To 30000  
    Temps_Explicite cmdExplicite, nIndex  
Next  
|  
cmdExplicite.Caption = "&Explicite"  
|  
lblTemps.Caption = Minute(Now - VarTemps) & " Mins, " & Second(Now - VarTemps) & "Secs"  
End Sub  
|  
Private Sub cmdImplicite_Click()  
Dim VarTemps As Variant
```

```
Dim nIndex As Integer
'
VarTemps = Now
'
For nIndex = 1 To 30000
    Temps_Explicite cmdExplicite, nIndex
Next
'
cmdExplicite.Caption = "&Explicite"
'
lblTemps.Caption = Minute(Now - VarTemps) & " Mins, " & Second(Now - VarTemps) & " Secs"
End Sub
'
```

### Voici les deux sous programmes à saisir dans la section "Général" de la fenêtre de code:

```
Private Sub Temps_Explicite(cmdCommande As CommandButton, nNumero As Integer)
    cmdCommande.Caption = nNumero
End Sub
'
Private Sub Temps_Implicite(cmdCommande As Control, nNumero As Integer)
    cmdCommande.Caption = nNumero
End Sub
```

Il ne nous reste plus qu'à comparer le temps écoulé par les deux instructions implicites et explicites.

Nous constatons que la première génère :

Explicite = 0 minute et 2 secondes

Implicite = 0 minute et 1 seconde

L'essai n'a été pratiqué que sur une portion infime de texte, imaginez un programme avec des millions voire des milliards de caractères ?

Néanmoins, parmi les divers essais que j'ai pu réaliser, j'ai constaté que bien souvent le temps entre les deux instructions étaient identiques et que parfois il est vrai la valeur affichée était de 50% inférieure à la valeur initiale.

Je vous laisserai le soin de tirer les conclusions que vous voudrez bien formuler sur l'exposé et l'exemple pratique que je viens de vous proposer.

**Vous pouvez télécharger l'exo complet de cette réalisation.**

Bon travail et merci de vos commentaires.

Prochain tutoriel :

 ***Les bases de données***

