

Date de publication :

Dernière mise à jour : 22/12/2008

Cette FAQ est le fruit de la collaboration des responsables, rédacteurs, modérateurs et autres utilisateurs du forum Visual Basic. Je les en remercie vivement. Elle a pour volonté de vous aider dans la réalisation d'applications VB6. Bien que son concepteur se tourne désormais vers une technologie .NET, ce langage n'en reste pas moins très répandu et supporte de nombreuses applications dans une multitude de domaines.

Soyez toujours nombreux à faire vivre cette FAQ soit en postant un code qui vous semble pertinent (**ici**) soit en améliorant et/ou corrigeant les codes qui vous sont déjà proposés.

Si vous avez soumis un code et qu'il n'apparaît pas dans la FAQ, soyez patient, il sera certainement dans la prochaine mise à jour !

Pour toutes questions ou tous problèmes inhérents à la FAQ, vous pouvez contacter par MP **bbil** ou **ThierryAIM**

***NB:** Les réponses aux questions sont basées sur VISUAL BASIC 6*

## Ont contribué à cette FAQ :

Khany - zazaraignée - ThierryAIM - Romain Puyfoulhoux  
- Jean-Marc Rabilloud - Abelman - Alexandre Lokchine  
- Da40 - DarkVader - Elifqaoui - hpj - Sygale - Bazoom  
- CSoldier - Khorne - Catbull - cafeine - Tofalu - ridan -  
Jacques Malatier - argyronet - shwin - Sadneth - Delphi-  
ne - odan71 - kracotte - e-steel - spacefrog - méphistopheles  
- mdriesbach - Optitech - SilkyRoad - jmfmarques -  
Xo - bbil - random - Theo - Delbeke - Cafeine - Tofalu  
- olivier] - Demco - LedZeppII - Gaël Donat - fdraven -  
nabil - Maxence HUBICHE - sovo - ProgElecT - forum -

---

1. A propos de cette FAQ (3) .....	4
2. Concepts, langage et environnement (42) .....	5
2.1. Concepts (5) .....	6
2.2. Langage (30) .....	8
2.3. Environnement de développement (7) .....	21
3. Interface (68) .....	24
3.1. Form (18) .....	25
3.2. Contrôles (35) .....	37
3.3. Divers (15) .....	61
4. Graphisme (5) .....	75
5. Système (105) .....	80
5.1. Fichiers (38) .....	116
5.2. Réseaux (18) .....	145
6. Bases de données (15) .....	166
7. Documentation et installation (8) .....	175
8. Liaison Office (19) .....	182
8.1. Excel (5) .....	183
8.2. Word (10) .....	186
8.3. Divers Office (4) .....	193
9. Divers (35) .....	196
9.1. Routines (25) .....	197
9.2. Manipulation de dates (10) .....	215

Sommaire > A propos de cette FAQ

Où puis-je trouver la dernière version de cette FAQ ?

**Auteurs : Romain Puyfoulhoux ,**

Elle se trouve à l'adresse <http://vb.developpez.com/faq/>.

Comment participer à cette FAQ ?

**Auteurs : Romain Puyfoulhoux ,**

**Une fois inscrit sur le forum de Developpez.com, vous pouvez poster vos questions/réponses dans ce sous-forum : Vos contributions VB6 .**

Où pouvons nous laisser des commentaires sur la faq VB6 ?

**Auteurs : bbil ,**

**Votre avis sur cette faq nous intéresse, une discussion à été ouverte sur le forum pour recueillir vos commentaires :**

---

## Sommaire > Concepts, langage et environnement

Sommaire > Concepts, langage et environnement > Concepts

## Que choisir entre VB6 et Visual Basic .Net ?

Auteurs : [Romain Puyfoulhoux](#) ,

VB.net est l'un des langages supportés par .Net, la nouvelle plateforme de Microsoft, dont le framework est totalement orienté objet. Il n'est donc pas une simple mise à jour comme l'était VB6 par rapport à VB5.

La question est donc moins de choisir entre deux langages qu'entre deux plateformes. .Net sera dans quelques années la nouvelle plateforme de Windows et remplacera donc l'actuelle Win32. Si vous voulez vous initier à un langage d'avenir, tournez-vous vers VB.Net: sur les prochaines versions de Windows, VB6 ne sera plus vraiment adapté.

lien : [La Faq .Net](#)

lien : [Comment migrer vos projets vers VB.NET ?](#)

lien : [De VB6 à VB.NET](#)

## Comment Obtenir ou télécharger VB6 ?

Auteurs : [bbil](#) ,

Visual basic 6, (VB6) n'étant plus commercialisé par Microsoft seul les marchés de l'occasion où autre ventes au enchères permettent encore de se procurer une licence.

lien :  [Support de Visual Basic 6.0 sous Windows® Vista](#)

## Comment choisir entre un module standard et un module de classe ?

Auteurs : [Romain Puyfoulhoux](#) ,

Un module standard contient un ensemble de fonctions et de procédures, plus toutes les déclarations qui leur sont nécessaires. Parmi ces déclarations, fonctions, et procédures, certaines peuvent être de portée privée, c'est-à-dire que l'on ne peut pas y accéder à partir d'un autre module. Sera donc de portée privée tout ce qui n'est utilisé que dans le module standard lui-même.

Un module de classe vous permettra de créer une classe. Une classe décrit un objet particulier : quelles sont ses caractéristiques (appelées propriétés) et ce qu'il peut faire (quelles sont ses méthodes). Par exemple, chaque form de votre projet correspond à une classe. Ses propriétés sont Caption, BorderStyle, etc... et ses méthodes Show, Hide, etc... Dans une classe, une méthode s'écrit sous la forme d'une fonction de portée publique. Voyons comment s'écrit une propriété, par exemple la propriété Marque de la classe Voiture :

```
'Variable privée représentant la marque, on ne peut pas y accéder depuis l'extérieur
Private m_Marque As String

'Property Get est la fonction appelée lorsque l'on veut obtenir la valeur de la propriété
Property Get Marque() As String
Marque = m_Marque
End Property

'Property Let est la procédure appelée lorsque l'on veut modifier la valeur de la propriété.
'Si elle n'existe pas, la propriété sera en lecture seule.
'Elle doit avoir un argument qui a le type de la propriété,
'soit le même que celui de la valeur renvoyée par Property Get.
Property Let Marque(value As String)
m_Marque = value
End Property
```

Et pour finir, un exemple d'utilisation de la classe Voiture :

```
vb
Dim UneVoiture As Voiture
Set UneVoiture = New Voiture
UneVoiture.Marque = "Renault"
MsgBox UneVoiture.Marque
Set UneVoiture = Nothing
```

### Comment faire un programme sans interface graphique, de type console ?

Auteurs : Romain Puyfoulhoux ,

Dans un module standard, créez une procédure Main(). Dans les propriétés du projet, Sélectionnez "Sub Main" comme objet de démarrage.

```
vb
Sub Main()
'code du programme
End Sub
```

### Quelles sont les différences principales entre les 2 modes de compilation de VB :

Auteurs : DarkVader ,

#### Natif

- Désassemblage seulement possible
- Optimisation vitesse (x5-x20 /P-Code selon type d'opération)
- Taille plus importante qu'un exécutable en P-Code (+30 à 50%)
- Possibilité de créer une table de Débogage Symbolique
- Quelques bugs à la compilation - Instabilité d'appel à certaines API (multiThreading par ex)

#### P-Code (Pseudo-Code)

- Code interprété - couche supplémentaire donc plus lent
- Dé-compilation possible à l'image de vb.net ou java (<http://www.vb-decompiler.org/>)
- Optimisation taille du code



*La compilation en code Natif ou en P-Code nécessite l'une comme l'autre MSVBVMx0.dll*

Sommaire > Concepts, langage et environnement > Langage

## A quoi sert "Option Explicit" ?

Auteurs : **Romain Puyfoulhoux** ,

La déclaration de "Option Explicit" au début d'un module indique à Visual Basic que toutes les variables utilisées à l'intérieur de ce module devront être préalablement déclarées. Ainsi si vous vous trompez sur le nom d'une variable vous aurez une erreur à la compilation. Etre prévenu d'une erreur pendant la phase de compilation est un gros avantage : elle vous évite d'avoir à la retrouver par débogage après avoir constaté un dysfonctionnement pendant l'exécution. De plus, déclarer vos variables vous permet de spécifier leur type, ce qui est préférable du point de vue des performances du programme.

Pour qu'"Option Explicit" soit ajouté automatiquement à la création d'un module, activez l'option "Déclaration des variables obligatoire" dans les options du projet (onglet "Editeur").

## Utilité du # , \$ , % , ! , @ dans le nom d'une variable ?

Auteurs : **zazaraignée** ,

il s'agit, "caractères de déclaration de type" ajoutés à une valeur pour indiquer le type de donnée qu'il faut lui attribuer. Il en existe quelques uns.

À l'origine, à l'époque du bon vieux code spaghetti, on déclarait une variable en y ajoutant ou non le suffixe \$ pour dire s'il s'agissait d'une valeur de chaîne ou d'une valeur numérique.

Au fil de l'évolution de Basic, il s'est ajouté d'autres types spécifiques avec leurs suffixes bien à eux.

Aujourd'hui, on les utilise encore dans quelques situations. En voici une liste:

caractères de déclaration de type

- \$ pour les valeurs de chaînes (variables seulement)
- % pour les entiers de type Integer
- & pour les entiers de type Long
- ! pour les réels de type Single
- # pour les réels de type Double
- @ pour les réels de type Currency

## Qu'est-ce que ByVal et ByRef ?

Auteurs : **Romain Puyfoulhoux** ,

Les arguments d'une fonction ou d'une procédure peuvent être passés de deux façons différentes : par valeur ou par référence. Si un argument est précédé de ByVal, il est passé par valeur, et s'il est précédé de ByRef, ou de ni l'un ni l'autre, il est passé par référence.

**Passage par valeur (ByVal)**

Consiste à passer la valeur de l'argument à la procédure, et non son adresse. La procédure utilise alors une copie de la variable. La valeur de la variable ne peut pas être modifiée par la procédure.

**Passage par référence (ByRef)**

Ici c'est l'adresse de la variable qui est passée à la procédure. La variable peut donc être modifiée par la procédure.

vb

```
Private Sub Echange(ByVal a As Integer, ByVal b As Integer)
    Dim temp As Integer
```



```
vb
temp = a
a = b
b = temp

End Sub

Private Sub Echange2(a As Integer, b As Integer)

Dim temp As Integer
temp = a
a = b
b = temp
End Sub

Private Sub Test()

x = 5
y = 3
Echange x, y
MsgBox x & " " & y 'affiche "5 3", les valeurs n'ont pas été modifiées par la procédure

Echange2 x, y
MsgBox x & " " & y 'affiche "3 5", parce que les valeurs ont été modifiées

End Sub
```

## Comment utiliser les énumérations. Comment créer une liste de constantes (ex : une liste de couleurs)

Auteurs : Tofalu ,

Un ensemble de constantes liées par leur sens peut être défini par une énumération à l'aide du mot clé Enum. La déclaration de l'énumération est à placer dans le haut du module (partie déclaration)

Chaque ligne de l'énumération est déclarée ainsi : <Nom de la valeur> = <Valeur : Type entier>

```
Private Enum Couleurs
Bleu=1
Vert=3
Jaune=5
End Enum
```

Ainsi, si on déclare une variable de type Couleurs, on aura par exemple :

```
Dim MaCouleur as Couleurs
MaCouleur=Couleurs.Vert
Msgbox MaCouleur
```

Ceci affichera 3.

## Comment insérer un saut de ligne ?

Auteurs : Romain Puyfoulhoux ,

Le saut de ligne est représenté par la constante vbCrLf. Exemple pour afficher un message sur 2 lignes avec MsgBox :

```
vb
```

vb

```
MsgBox "Opération terminée." & vbCrLf & "Cliquez sur OK."
```

## Comment récupérer l'emplacement de mon programme ?

**Auteurs : Romain Puyfoulhoux ,**

En utilisant La propriété Path de l'objet App. Ajoutez le caractère "\" à la fin s'il n'y ait pas déjà.

vb

```
Dim Path as String

Path = App.Path
If right(Path,1) <> "\" then Path = Path & "\"
```

## Comment passer un tableau en paramètres ?

**Auteurs : Romain Puyfoulhoux ,**

L'exemple suivant affiche les éléments d'un tableau :

vb

```
Private Sub AfficheElements(t() As Long)

Dim i As Long
For i = LBound(t) To UBound(t)
    MsgBox t(i)
Next

End Sub

Private Sub Form_Load()

Dim t(1 To 3) As Long
t(1) = 1
t(2) = 4
t(3) = 5
AfficheElements t()

End Sub
```

## Comment tester si un tableau dynamique est vide ?

**Auteurs : Romain Puyfoulhoux ,**

La fonction UBound() renvoie l'indice maximum autorisé pour un tableau donné. Si le tableau est vide, elle crée une erreur. La fonction suivante utilise ce principe. Elle renvoie vrai si le tableau passé en paramètre contient au moins un élément.

vb

```
Public Function ContientElements(ByVal tableau As Variant) As Boolean

Dim indice As Long
```

vb

```
On Error goto vide
indice = UBound(tableau)
ContientElements = True
Exit Function
```

vide:

```
End Function
```

## Comment faire une fonction qui renvoie un résultat ?

Auteurs : **Romain Puyfoulhoux**,

La fonction suivante renvoie un booléen : Vrai si le nombre reçu en paramètre est pair, et faux sinon.

vb

```
Private Function EstPaire(x As Long) As Boolean
    EstPaire = (x Mod 2 = 0)
End Function
```

## Comment faire une fonction qui renvoie un tableau ?

Auteurs : **Jean-Marc Rabilloud**,

Dans l'exemple ci-dessous, la fonction **RGBparTableau()** renvoie les composantes rouge, verte, bleue du code couleur passé en paramètre. Les composantes sont renvoyées sous la forme d'un tableau.

vb

```
Public Function RGBparTableau(ByVal couleur As Long) As Long()

Dim MonTab(0 To 2) As Long
MonTab(2) = Int(couleur / 65536)
MonTab(1) = Int((couleur - (65536 * MonTab(2))) / 256)
MonTab(0) = couleur - ((MonTab(2) * 65536) + (MonTab(1) * 256))
RGBparTableau = MonTab

End Function

Private Sub Command1_Click()

Dim TabCouleur() As Long, couleur As Long

couleur = 9550940
TabCouleur() = RGBparTableau(couleur)
MsgBox "Le code " & couleur & " correspond en RGB à " & _
    TabCouleur(0) & " " & TabCouleur(1) & " " & TabCouleur(2)

End Sub
```

## Comment faire une fonction qui renvoie une variable de type utilisateur ?

Auteurs : **Jean-Marc Rabilloud**, **Romain Puyfoulhoux**,

Une fonction peut renvoyer une variable de type utilisateur comme n'importe quel autre type simple.

Cependant vous ne pouvez pas déclarer dans un module de classe, y compris dans le module d'une form, une fonction de portée publique renvoyant un type utilisateur. Dans un module standard, si vous déclarez un type utilisateur et une fonction de portée publique qui renvoie ce type, le type doit être aussi déclaré avec une portée publique.

Dans l'exemple ci-dessous, la fonction RGBparType() renvoie les composantes rouge, verte, bleue du code couleur passé en paramètre. Les composantes sont les champs du type CompCouleur.

```

vb

Private Type CompCouleur
    Red As Long
    Green As Long
    Blue As Long
End Type

Private Function RGBparType(ByVal Couleur As Long) As CompCouleur

    RGBparType.Blue = Int(Couleur / 65536)
    RGBparType.Green = Int((Couleur - (65536 * RGBparType.Blue)) / 256)
    RGBparType.Red = Couleur - ((RGBparType.Blue * 65536) + (RGBparType.Green * 256))

End Function

Private Sub Command1_Click()

    Dim TypColor As CompCouleur

    TypColor = RGBparType(9550940)
    lblred.Caption = "Red = " & TypColor.Red
    lblgreen.Caption = "Green = " & TypColor.Green
    lblblue.Caption = "Blue = " & TypColor.Blue

End Sub
    
```

### Comment rechercher une chaîne de caractères dans une autre ?

**Auteurs : Romain Puyfoulhoux ,**

**Instr(p,ch1,ch2) recherche la chaîne ch2 dans ch1 à partir de la position p. Si ch2 est trouvée, la valeur renvoyée est la position de son premier caractère dans ch1, sinon la fonction renvoie 0.**

```

vb

pos = Instr(4, "Nous sommes au mois de Juillet.", "Juillet")      'renvoie 24
pos = Instr(27, "Nous sommes au mois de Juillet.", "Juillet")    'renvoie 0
    
```

### Convertir en majuscule la première lettre de chaque mot d'une phrase

**Auteurs : forum ,**

**La fonction VBA StrConv permet de convertir des chaînes de caractères.**

```
StrConv(String, Conversion)
```

**Le premier paramètre correspond à la chaîne de caractères à convertir, le deuxième est le masque de conversion. Utilisez la constante VbLowerCase pour convertir en minuscules, VbUpperCase pour convertir en majuscules,**

**VbProperCase** pour convertir en noms propres : la première lettre de chaque mot est alors en majuscule, le reste en minuscules.

```
MsgBox StrConv("Ceci est un essai", vbProperCase)
```

Affiche : Ceci Est Un Essai

## Comment récupérer les arguments de la ligne de commande passée à l'exécutable ?

**Auteurs : Romain Puyfoulhoux ,**

La ligne de commande et ses arguments vous est renvoyée par la fonction **Command()** sous la forme d'une chaîne de caractères. Pour transformer cette chaîne en un tableau dont chaque élément correspond à un argument, utilisez la fonction **Split()** :

vb

```
Dim args() As String  
args = Split(Command(), " ")
```

## Comment créer un nombre aléatoire ?

**Auteurs : Romain Puyfoulhoux ,**

La fonction **Rnd()** renvoie une valeur aléatoire supérieure ou égale à 0 et strictement inférieure à 1.

vb

```
Public Function NombreAleatoire(ByVal lngInf As Long, ByVal lngSup As Long) As Long  
  
'Renvoie une valeur comprise entre les limites lngInf et lngSup  
  
Randomize 'initialise le générateur pseudo-aléatoire  
NombreAleatoire = Int(Rnd() * (lngSup - lngInf + 1)) + lngInf  
  
End Function
```

## Comment obtenir la constante Pi ?

**Auteurs : Alexandre Lokchine ,**

Contrairement à d'autres langages, la constante **Pi** n'existe pas en VB. La meilleure façon de l'obtenir rapidement est d'utiliser le code suivant :

vb

```
Dim Pi as Double
```

vb

```
Pi=4*Atn(1)
```

## Comment manipuler des entiers supérieurs à la valeur d'un type Long ?

**Auteurs : ThierryAIM ,**

Dans Visual Basic, le type *Long* représente un nombre entier, codé sur 4 octets, dont la valeur est comprise entre -2 147 483 648 et 2 147 483 647.

Il est possible de manipuler des nombres dépassant ces limites, en utilisant le type *Decimal*

Ce type de donnée peut stocker des nombres dans la plage de valeurs : +/-79 228 162 514 264 337 593 543 950 335

Le type *Decimal* est un sous-type de *Variant*. Il n'est donc pas possible de l'attribuer directement à une variable.

Pour contourner le problème, il faut créer une variable de type *Variant* et utiliser la fonction *CDec*, afin de créer un sous-type *Decimal*

vb

```
Dim d As Variant
d = CDec("79 228 162 514 264 337 593 543 950 000")
MsgBox d + 335 'Maxi possible !
```

Toutes les opérations arithmétiques s'effectuent normalement, sous réserve, bien sûr, comme pour tous les types de variables, de ne pas dépasser les valeurs limites.

## Comment utiliser la compilation conditionnelle ?

**Auteurs : Romain Puyfoulhoux ,**

La compilation conditionnelle vous permet de compiler et donc d'inclure dans l'exécutable un bloc de code source uniquement si une condition est vérifiée.

**Syntaxe :**

```
#If expression Then
    statements
[#ElseIf expression-n Then
    [elseifstatements]]
[#Else
    [elsestatements]]
#End If
```

expression est ici une ou plusieurs comparaisons entre une constante de compilation conditionnelle et une valeur.

Initialiser une constante de compilation conditionnelle peut se faire de plusieurs manières. Vous pouvez utiliser la directive `#Const` :

vb

```
#Const nomconstante = expression
```

En utilisant cette directive, la constante déclarée est accessible uniquement à l'intérieur du module où elle est déclarée. Vous pouvez aussi indiquer la valeur d'une constante dans le deuxième onglet de la fenêtre des propriétés du projet.

Avec cette méthode, la constante sera accessible dans tous les modules du projet. Et pour donner la valeur True à une valeur booléenne, il faut spécifier la valeur ?1.

Voici quelques cas où la compilation conditionnelle peut être intéressante :

- enregistrer dans un fichier log tout ce qui se passe pendant l'exécution du programme, mais uniquement avec une version compilée spécialement pour le débogage ou bien tout au long de la phase de développement.
- compiler un code source différent suivant la version de Windows visée.
- compiler certains codes sources selon les fonctionnalités à inclure dans la version de l'application qui est compilée. La compilation conditionnelle permet ici de n'inclure dans l'exécutable que le code source nécessaire; un pirate ne pourra donc pas bidouiller l'exécutable afin d'activer les autres fonctionnalités.

Voici un petit exemple où l'on active les menus que si les fonctions correspondantes sont incluses dans la version compilée :

```
vb
#Const module_stats = True
#Const module_export = True

Private Sub Form_Load()

#If module_stats Then
    menuStats.Enabled = True
#End If

#If module_export Then
    menuExport.Enabled = True
#End If

End Sub
```

## Comment lire ou modifier une propriété d'un objet par son nom ?

**Auteurs : Romain Puyfoulhoux ,**

**Avec la fonction CallByName.**

```
vb
Private Sub Form_Load()

'Ici CallByName renvoie la valeur de la propriété Caption de la form
Msgbox CallByName(Me, "Caption", VbGet)

'modifie la propriété Caption de la form
CallByName Me, "Caption", VbLet, "Test de callbyname"

End Sub
```

## Comment exécuter une méthode en passant son nom en argument ?

**Auteurs : Romain Puyfoulhoux ,**

**Avec la fonction CallByName.**

vb

```
Private Sub Form_Load()  
  
    'Déplace la form à la position 5000,150  
    CallByName Me, "Move", VbMethod, "5000,150"  
  
End Sub
```

## Comment utiliser les expressions régulières ?

Auteurs : Romain Puyfoulhoux ,

La librairie Microsoft VBScript Regular Expressions permet d'utiliser des expressions régulières dans Visual Basic. Il faut l'ajouter dans les références du projet.

Le principe consiste à créer un objet RegExp, à le paramétrer en modifiant ses propriétés, puis à lancer la recherche en appelant la méthode Execute(). Vous devez passer en paramètre la chaîne de caractères dans laquelle vous souhaitez faire une recherche. La méthode Execute vous renvoie les occurrences dans une collection. Si vous avez donné la valeur True à la propriété Global de l'objet RegExp, cette collection contiendra toutes les occurrences, sinon seulement la première.

Voici comment vérifier la validité d'une adresse email. Ce code a été écrit avec la version 5.5 de la librairie.

vb

```
Public Function EmailValide(ByVal email As String) As Boolean  
  
    Dim regEx As RegExp, occurrences As MatchCollection  
  
    Set regEx = New RegExp  
    regEx.Pattern = "[a-z0-9_.-]+@[a-z0-9.-]{2,}\.[a-z]{2,3}$"  
    regEx.IgnoreCase = True 'Ne pas distinguer les minuscules des majuscules  
    regEx.Global = False 'Renvoyer seulement la première occurrence  
    Set occurrences = regEx.Execute(email)  
    EmailValide = (occurrences.Count = 1)  
  
End Function
```

## Comment interpréter une chaîne string ?

Auteurs : Sadneth , shwin ,

Si vous travaillez uniquement en VBScript :

vb

```
Dim toto As String  
  
toto = "SUPER * 10"  
MsgBox Eval(Replace(toto, "SUPER", 500))
```

Si vous travaillez en VB6, il faut cocher le composant Microsoft Script Control dans Projet -> Composants et placer ce code :



vb

```
Dim r As New ScriptControl

Dim toto As String

toto = "SUPER * 10"

toto = Replace(toto, "SUPER", 500)

r.Language = "vbscript"
Debug.Print r.Eval(toto)
```

## Comment Copier, Coller, Couper, Annuler ?

**Auteurs : ThierryAIM ,****Un petit exemple avec un RichTextBox :**

vb

```
Private Const EM_UNDO = &HC7
Private Const EM_CANUNDO = &HC6
Private Const WM_USER As Long = &H400
Private Const EM_REDO As Long = (WM_USER + 84)
Private Const EM_CANREDO As Long = (WM_USER + 85)
Private Const WM_COPY = &H301
Private Const WM_CUT = &H300
Private Const WM_CLEAR = &H303
Private Const WM_PASTE = &H302
Private Declare Function SendMessage Lib "user32" Alias "SendMessageA" (ByVal _
    hWnd As Long, ByVal wParam As Long, ByVal lParam As Any) As Long

Private Sub mnuEditSubRedo_Click()
    SendMessage txtTerm.hWnd, EM_REDO, 0, ByVal 0&
End Sub

Private Sub mnuEditSubUndo_Click()
    SendMessage txtTerm.hWnd, EM_UNDO, 0, ByVal 0&
End Sub

Private Sub mnuEditSubCopy_Click()
    Clipboard.Clear
    SendMessage txtTerm.hWnd, WM_COPY, 0, 0
End Sub

Private Sub mnuEditSubCut_Click()
    Clipboard.Clear
    SendMessage txtTerm.hWnd, WM_CUT, 0, 0
End Sub

Private Sub mnuEditSubPaste_Click()
    SendMessage txtTerm.hWnd, WM_PASTE, 0, 0
End Sub

Private Sub RichText1_Change()
    mnuEditSubUndo.Enabled = SendMessage(txtTerm.hWnd, EM_CANUNDO, 0, ByVal 0&)
    mnuEditSubRedo.Enabled = SendMessage(txtTerm.hWnd, EM_CANREDO, 0, ByVal 0&)
```

```
vb  
End Sub
```

## Comment redimensionner un tableau ?

Auteurs : **kracotte** ,

```
vb  
  
Dim Tab(longueur) As String  
'si longueur vaut 2 vous aurez Tab(0), Tab(1), Tab(2)  
'on test si le tableau est plein  
If Tab (UBound(Tab)) <> "" Then  
'on redimensionne le tableau en "préservant" ses valeurs  
  
ReDim Preserve Tab (Nouvelle Longueur)  
  
End If
```

## Comment accéder à une procédure, fonction ou variable déclarée dans une form, à partir d'un autre module ?

Auteurs : **Romain Puyfoulhoux** ,

Pour qu'une fonction, procédure, ou variable soit accessible à partir d'un autre module que celui où elle est déclarée, sa déclaration doit débuter par le mot clé **Public**. Voici les déclarations d'une variable de type long et d'une procédure publiques dans la form nommée Form1 :

```
vb  
  
Public Variable As Long  
  
Public Sub Afficher()  
    'code  
End Sub
```

Et voici comment les appeler depuis un autre module :

```
vb  
  
Form1.Variable = 3  
Form1.Afficher
```

## A quoi correspond le Me. que je vois devant le nom de certains objets dans des lignes de code ?

Auteurs : **Demco** ,

Le **Me** désigne en fait l'instance du formulaire sur lequel on se trouve.  
Ainsi la ligne suivante fait référence à une zone de texte se trouvant dans le formulaire actuel :

```
Me.txtNom = "Dupond"
```

## Comment instancier un nouvel objet ?

Auteurs : Tofalu ,

Deux syntaxes sont possibles :

```
Dim obj as new cIToto  
obj.mamethode
```

Ou

```
Dim obj as cIToto  
Set obj= New cIToto  
obj.mamethode
```

Pourtant, il existe une différence fondamentale entre les deux syntaxes. En Visual Basic, dès qu'une méthode est appelée, une vérification interne est réalisée. Celle-ci consiste à vérifier que l'objet n'est pas vide avant l'appel. Le mot clé SET permet d'outrepasser cette vérification et de dire explicitement au système que l'objet a été instancié. Ainsi, on gagne de nombreuses opérations qui auraient dues être effectuées à chaque fois que l'objet aurait été référencé. La syntaxe 2 consomme donc une ligne de code de plus mais est beaucoup plus performante.

## Comment tester si deux variables représentent le même objet ?

Auteurs : Tofalu ,

Pour tester si deux variables correspondent au même objet, il faut utiliser l'opérateur IS :

```
Dim a As Object  
Dim b As Object  
Set a = CurrentDb  
Set b = a  
If b Is a Then  
    MsgBox "Le même objet"  
Else  
    MsgBox "Objet différent"  
End If  
Set b = CurrentProject  
  
If b Is a Then  
    MsgBox "Le même objet"  
Else  
    MsgBox "Objet différent"  
End If
```

Ceci affiche respectivement :

1 Le même objet

## 2 Objet différent

### Comment faire une procédure ou une fonction qui a un ou plusieurs paramètres optionnels ?

Auteurs : Romain Puyfoulhoux ,

Un paramètre est optionnel s'il est précédé du mot clé **Optional**. Les paramètres placés après doivent être également optionnels. Vous pouvez également spécifier une valeur par défaut :

vb

```
Private Sub Affiche(Optional x As Long=10)
End Sub
```

Pour les paramètres de type **Variant** et sans valeur par défaut, vous avez le moyen de savoir si une valeur a été spécifiée lors de l'appel en utilisant **IsMissing** :

vb

```
Private Sub Affiche(Optional x As Variant)
If IsMissing(x) Then
    'pas de valeur spécifiée
Else
    'une valeur a été spécifiée pour x
End If
End Sub
```

### Exécuter un code si une variable optionnelle est passée en paramètre d'une fonction

Auteurs : Cafeine ,

Dans la déclaration de votre fonction déclarez l'argument facultatif de cette façon :

```
...Optional ByVal strArg1 as String="<empty>"...
```

Puis dans votre fonction regardez la valeur de **strArg1** :

```
Dim flagNonRenseigné As Boolean
flagNonRenseigné = False

If strArg1 = "<empty>" Then
    flagNonRenseigné = True
    strArg1 = ""
End If
```

Sommaire > Concepts, langage et environnement > Environnement de développement

Trouver des informations sur classes,méthodes, événements... d'un projet ?

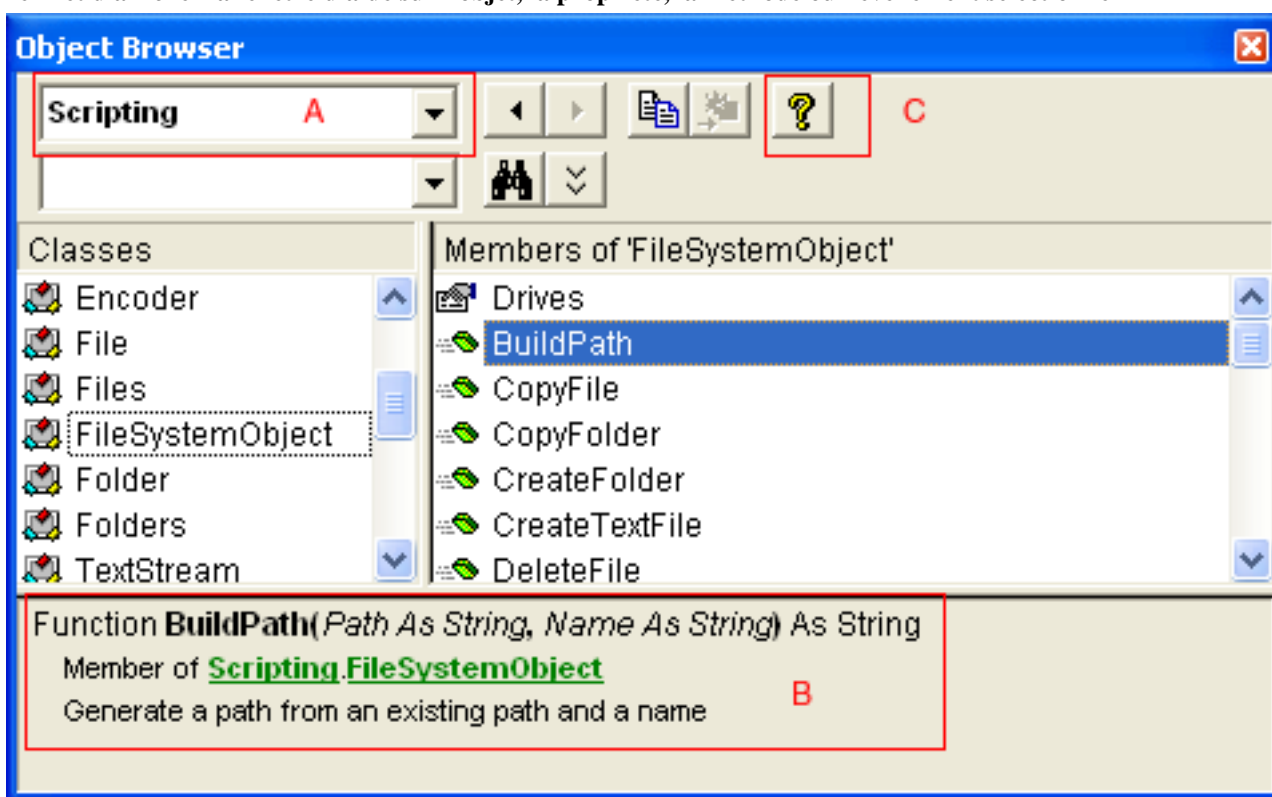
Auteurs : zazaraignée ,

On cherche souvent de l'aide directement sur le CD MSDN sans trop savoir ce que l'on cherche au juste. L'une des premières sources d'information lorsqu'on utilise un composant que l'on ne connaît pas bien et l'explorateur d'objets.

Son bouton est dans la barre d'outils. Il est accessible aussi par l'option de menu Affichage > Explorateur d'objets (View > Object browser pour ceux qui ont la version anglaise) ou encore par le raccourci F2.

Voir l'image ci-dessous:

1. Affiche une liste des bibliothèques (références) chargées
2. Donne une aide sommaire immédiate
3. Permet d'afficher la fenêtre d'aide sur l'objet, la propriété, la méthode ou l'événement sélectionné



*L'Explorateur d'objet (Touche F2)*

**Aide en ligne VB :**

*Affiche les classes, propriétés, méthodes, événements et constantes disponibles dans les bibliothèques d'objets et les procédures de votre projet.*


*Il vous permet de rechercher et d'utiliser des objets que vous créez ainsi que des objets provenant d'autres applications.*

Comment utiliser la molette de la souris dans l'éditeur de code de VB6 ?

Auteurs : ThierryAIM ,

**Par défaut, la molette de la souris n'est pas reconnue dans l'éditeur de code de Visual Basic 6.**

**Il est toutefois possible d'installer un palliatif à ce problème**

Vous trouverez toute la démarche à suivre à  cette adresse sur le site de Microsoft

## Donner le style Windows XP aux contrôles dans l'IDE de Visual Basic 6

Auteurs : **ThierryAIM** ,

Cette astuce vous permet de tester votre programme avec le style XP en mode conception, sans le compiler.

- Fermer votre environnement Visual Basic 6
- Créez un fichier manifest comme décrit au paragraphe [FAQ Comment donner le style de Windows XP à mes contrôles VB6 ?](#) et enregistrez-le sous VB6.exe.manifest dans le répertoire d'installation de Visual Basic 6 (par défaut : *C:\Program Files\Microsoft Visual Studio\VB98*).
- Relancez votre IDE VB6, et ajoutez des contrôles à votre form.

### Limites et Solutions

Certains contrôles ne s'affichent pas correctement lorsqu'ils sont disposés directement sur une form; c'est le cas notamment des `OptionButtons` ou des `Frames`

**Solution** : Insérer ces contrôles dans un conteneur `PictureBox` et le tour est joué.

Certains assistants de VB6 présentent aussi des problèmes d'affichage, car ils n'ont pas été conçus ou testés avec le mode XP

Il n'y a pas de solution dans ce cas. Si cela est vraiment gênant, supprimez ou renommez le fichier `vb6.exe.manifest` et relancez votre IDE

lien : [FAQ Comment donner le style de Windows XP à mes contrôles VB6 ?](#)

## Comment commenter un bloc de code source ?

Auteurs : **Abelman** ,

Il suffit d'ajouter les boutons 'Commenter bloc' et 'Décommenter bloc' au menu Edition.

- Pour cela, ouvrir le menu `Affichage` (troisième en partant de la gauche), puis `Barres d'outils`, et cocher "Edition" ou sélectionner "Personnaliser" (le dernier).
- Une boîte de dialogue s'ouvre. Choisir l'onglet `Commandes` (le 2ème), puis sélectionner `Edition` dans la liste de gauche.
- Dans le liste de droite se trouvent les boutons que vous pouvez ajouter à votre barre d'outils VB.
- Faire glisser les boutons 'Commenter bloc' puis 'Décommenter bloc' depuis cette liste vers la barre d'outils VB.

Il ne vous reste plus qu'à sélectionner un bloc de code et cliquer sur **Commenter Bloc** ou **Décommenter Bloc** et admirer le résultat.

Pourquoi vb s'arrête sur une erreur malgré "on error goto" ou "on error resume next" ?

**Auteurs : Romain Puyfoulhoux ,**

Pour corriger ce problème, allez dans le menu **Outils - Options**, cliquez sur l'onglet **"Général"** et pour l'option **"Récupération d'erreur"**, sélectionnez la valeur **"Arrêt sur les erreurs non gérées"**.

Pourquoi le menu qui permet de créer l'exécutable est désactivé et comment je peux le réactiver ?

**Auteurs : Romain Puyfoulhoux ,**

Ceci est généralement dû à un plantage de VB. Pour réactiver ce menu, allez dans le menu **"Affichage"**, **"Barre d'outils"**, **"Personnaliser"**, puis cliquez sur le bouton **"Rétablir..."** et validez en cliquant sur **"OK"**.

Comment savoir si l'on utilise VBA (Visual basic pour application) ou VB6 ?

**Auteurs : bbil ,**

VBA est intégré à une application tierce (Autocad, Excel, Word...) alors que l'IDE de VB6 est autonome. la fenêtre "à propos" menu "?" puis à propos permet de les différencier pour VBA :

pour VB6 :

l'on remarque la différence des icônes (le 32 pour VB6) et les différents numéros de version 6.3 pour VBA et 6.0 pour VB6.





Sommaire > Interface > Form

## Comment ajouter dynamiquement des contrôles dans une form ?

Auteurs : **Romain Puyfoulhoux** ,

Vous avez plusieurs solutions possibles. La première est de poser un contrôle sur la form et de lui donner un index, par exemple 0. Pour créer les autres contrôles à l'exécution, il faut les charger avec Load, les positionner, puis les rendre visibles. Vous pouvez ensuite détruire ces contrôles avec Unload. Pour tester l'exemple suivant, placez un textbox sur une form et donnez-lui l'index 0. Ces quelques lignes font apparaître 9 autres textbox :

vb

```
Dim i As Long
For i = 1 To 9
    Load Text1(i)
    Text1(i).Top = Text1(i - 1).Top + Text1(0).Height + 60
    Text1(i).Visible = True
Next
```

Vous pouvez ensuite les détruire :

vb

```
Dim i As Long
For i = 1 To 9
    Unload Text1(i)
Next
```

Une autre façon de procéder consiste à ajouter un élément à la collection Controls de la form. Auparavant il faut avoir déclaré votre contrôle dans la partie Déclarations du module de la form. Une fois la déclaration ajoutée, vous pourrez sélectionner votre contrôle dans la liste déroulante au-dessus de l'éditeur de code, comme si le contrôle avait été posé sur la form. Exemple pour un textbox :

vb

```
Dim WithEvents TextBoxDynamique As VB.TextBox
```

Nous pouvons alors créer le contrôle, le placer, le dimensionner et le rendre visible :

vb

```
Set TextBoxDynamique = Me.Controls.Add("VB.TextBox", "txtDynamic")
'txtDynamic est le nom du textbox
TextBoxDynamique.Left = 2000
TextBoxDynamique.Top = 1000
TextBoxDynamique.Width = 1500
TextBoxDynamique.Height = 70
TextBoxDynamique.Visible = True
```

Pour supprimer le contrôle :

```
Me.Controls.Remove "txtDynamic"
```

```
Set TextBoxDynamique = Nothing
```

## Comment empêcher la fermeture d'une form ?

Auteurs : **Romain Puyfoulhoux** ,

Quand la fermeture d'une form a été demandée, elle reçoit l'évènement QueryUnload. Dans l'exemple suivant, la form ne se ferme pas, sauf si la demande de fermeture est due à la commande Unload :

```
vb

Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)
    If UnloadMode <> vbFormCode Then Cancel = 1
End Sub
```

## Comment désactiver le bouton de fermeture sur une form ?

Auteurs : **Jean-Marc Rabilloud** ,

Placez tout d'abord ces déclarations dans le module de la form :

```
vb

Private Const SC_CLOSE = &HF060&
Private Const MF_BYCOMMAND = &H0&

Private Declare Function GetSystemMenu Lib "user32" (ByVal hwnd As Long, ByVal bRevert As Long) As Long
Private Declare Function RemoveMenu Lib "user32" (ByVal hMenu As Long, ByVal nPosition As Long, _
    ByVal wFlags As Long) As Long
```

Puis ces quelques lignes dans la procédure Form\_Load :

```
vb

Dim hSysMenu As Long
hSysMenu = GetSystemMenu(Me.hwnd, False)
RemoveMenu hSysMenu, SC_CLOSE, MF_BYCOMMAND
```

## Comment activer/désactiver les boutons "Réduire" et "Agrandir" d'une form ?

Auteurs : **ThierryAIM** ,

Les propriétés MaxButton et MinButton d'une form VB ne sont pas accessibles en mode exécution.

Cette astuce permet de les modifier par le code.

NOTA : valable uniquement à la création de la Form, dans la procédure Form\_Load. Une fois la form chargée, il n'est plus possible de modifier ses options de menu

```
vb

'-- Déclarations
Private Const GWL_STYLE As Long = -16

Private Const WS_MAXIMIZEBOX = &H10000
Private Const WS_MINIMIZEBOX = &H20000
```

vb

```
Private Const WS_SYSMENU = &H80000

Private Declare Function GetWindowLong Lib "user32" Alias "GetWindowLongA" _
    (ByVal hwnd As Long, ByVal nIndex As Long) As Long

Private Declare Function SetWindowLong Lib "user32" Alias "SetWindowLongA" _
    (ByVal hwnd As Long, ByVal nIndex As Long, _
    ByVal dwNewLong As Long) As Long
```

**Pour activer les boutons :**

vb

```
Private Sub Form_Load()
    SetWindowLong hwnd, GWL_STYLE, GetWindowLong(Me.hwnd, GWL_STYLE) Or WS_MAXIMIZEBOX
    SetWindowLong hwnd, GWL_STYLE, GetWindowLong(Me.hwnd, GWL_STYLE) Or WS_MINIMIZEBOX
End Sub
```

**Pour désactiver les boutons :**

vb

```
Private Sub Form_Load()
    SetWindowLong hwnd, GWL_STYLE, GetWindowLong(Me.hwnd, GWL_STYLE) Xor WS_MAXIMIZEBOX
    SetWindowLong hwnd, GWL_STYLE, GetWindowLong(Me.hwnd, GWL_STYLE) Xor WS_MINIMIZEBOX
End Sub
```

## Comment faire pour que ma form soit toujours au premier plan ?

**Auteurs : Romain Puyfoulhoux ,**

**Ajoutez ces déclarations dans le module de votre form :**

vb

```
Private Const SWP_NOMOVE = 2
Private Const SWP_NOSIZE = 1
Private Const HWND_TOPMOST = -1
Private Const HWND_NOTOPMOST = -2;

Private Declare Function SetWindowPos Lib "USER32" (ByVal hwnd As Long, _
    ByVal hWndInsertAfter As Long, ByVal x As Long, ByVal y As Long, _
    ByVal cx As Long, ByVal cy As Long, ByVal wFlags As Long) As Long
```

**Et placez cette ligne dans la procédure Form\_Load :**

vb

```
SetWindowPos Me.hwnd, HWND_TOPMOST, 0, 0, 0, 0, SWP_NOMOVE Or SWP_NOSIZE
```

**Pour annuler, placez cette ligne dans la procédure adéquate :**

vb

vb

```
SetWindowPos Me.hwnd, HWND_NOTOPMOST, 0, 0, 0, 0, SWP_NOMOVE Or SWP_NOSIZE
```

## Comment faire passer une MDIChild form au 1er plan en cliquant sur un bouton dans une ToolBar?

Auteurs : **ThierryAIM**,

Pour placer le nom de la MDIChild Form sur un bouton dans la ToolBar, il faut utiliser la fonction CallByName. Pour le passage au premier plan, on utilise Zorder. Info : Le contrôle ToolBar se trouve dans Microsoft Windows Common Controls 5.0 (SP2) ou Microsoft Windows Common 6.0 (SP4) suivant le Service Pack VB installé. Dans Form1, placez ce code :

vb

```
Public FormName As Object

Private Sub Form_Load()
    Set FormName = Me
    MDIForm1.Toolbar1.Buttons.Add 1, "Key1", Me.Name
End Sub
```

Dans MDIForm1, placez ce code :

vb

```
Private Sub Toolbar1_ButtonClick(ByVal Button As MSComctlLib.Button)
    Select Case Button.Key
        Case "Key1"
            CallByName FormName, "ZOrder", VbMethod, 0
    End Select
End Sub
```

Pour retirer le bouton lors de la fermeture de la MDIChild Form, ajouter ce code dans Form1 :

vb

```
Private Sub Form_Unload(Cancel As Integer)
    MDIForm1.Toolbar1.Buttons.Remove "Key1"
End Sub
```

## Comment savoir quelles forms sont chargées en mémoire ?

Auteurs : **Romain Puyfoulhoux**,

Elles sont dans la collection Forms. Ainsi cette boucle décharge toutes les forms :

vb

```
Dim f As Form
For Each f In Forms
    Unload f
```

vb

[Next](#)

## Comment redimensionner les contrôles quand la form est redimensionnée ?

Auteurs : [Romain Puyfoulhoux](#) ,

Placez ce code source dans le module de votre form. Le principe consiste ici à déplacer et redimensionner les contrôles afin qu'ils soient disposés de la même façon. Ainsi si la form a subit un agrandissement de 4 \* 3, toutes les distances sont multipliées par les mêmes facteurs.

Une autre méthode, plus souvent utilisée, consiste à agrandir seulement les contrôles principaux, et à garder les mêmes dimensions pour les contrôles plus petits (boutons, listes déroulantes, etc...). Mais dans ce cas un code source spécifique doit être écrit pour chacune de vos forms.

(La fonction *ExistProperty* est disponible dans la FAQ VB, voir le lien ci-dessous)

vb

```
Dim lar As Long, lng As Long

Private Sub Form_Load()

    lng = Me.Width
    lar = Me.Height

End Sub

Private Sub Form_Resize()

    Dim ctl As Control

    If (Me.WindowState = 1) Then Exit Sub

    For Each ctl In Me.Controls
        If TypeOf ctl Is ComboBox Then
            'Les comboboxes ont leur propriété Height en lecture seule
            ctl.Move ctl.Left * Me.Width / lng, ctl.Top * Me.Height / lar, ctl.Width * Me.Width / lng
        Else
            If ExistProperty(ctl, "Width") And ExistProperty(ctl, "Height") Then
                ctl.Move ctl.Left * Me.Width / lng, ctl.Top * Me.Height / lar, _
                    ctl.Width * Me.Width / lng, ctl.Height * Me.Height / lar
            End If
        End If
    Next

    lng = Me.Width
    lar = Me.Height

End Sub
```

lien : [FAQ](#) Comment savoir si une propriété existe pour un contrôle ?

## Comment passer des paramètres à une form lors de son ouverture ?

Auteurs : [Romain Puyfoulhoux](#) ,

La méthode Show de la classe Form ne vous permet pas d'ajouter vos propres paramètres. L'astuce consiste à passer par des variables ou des propriétés publiques. Dans l'exemple suivant, les variables privées m\_chaine1 et m\_chaine2 sont

accessibles en lecture/écriture depuis l'extérieur par l'intermédiaire des propriétés `Chaine1` et `Chaine2`. La méthode publique `Init()` peut aussi être appelée afin d'initialiser ces propriétés.

Module de la Form :

vb

```
Private m_chaine1 As String
Private m_chaine2 As String

Public Sub Init(strChaine1 As String, strChaine2 As String)
    Chaine1 = strChaine1
    Chaine2 = strChaine2
End Sub

Public Property Get Chaine1() As String
    Chaine1 = m_chaine1
End Property

Property Let Chaine1(strChaine As String)
    m_chaine1 = strChaine
End Property

Property Get Chaine2() As String
    Chaine2 = m_chaine2
End Property

Property Let Chaine2(strChaine As String)
    m_chaine2 = strChaine
End Property
```

Pour initialiser la form et l'afficher :

vb

```
Form1.Chaine1 = "La première chaîne"
Form1.Chaine2 = "La deuxième chaîne"
Form1.Show
```

Ou bien encore :

vb

```
Form1.Init "La première chaîne", "La deuxième chaîne"
Form1.Show
```

## Comment permettre le déplacement d'une form qui n'a pas de barre de titre ?

Auteurs : **Romain Puyfoulhoux** ,

Nous allons permettre à l'utilisateur de déplacer une form après avoir cliqué dessus, comme s'il avait cliqué dans la barre de titre. Pour cela, nous annulons le clic sur la form et envoyons un clic dans la barre de titre. Commencez par copier ces déclarations au début du module de la form :

vb

```
Private Declare Function SendMessage Lib "User32" Alias "SendMessageA" _
    (ByVal hWnd As Long, ByVal wParam As Long, _
    ByVal lParam As Long, lParam As Any) As Long
```

```
vb  
  
Private Declare Sub ReleaseCapture Lib "User32" ()  
  
Const WM_NCLBUTTONDOWN = &H1  
Const HTCAPTION = 2
```

Puis le code de la procédure Form\_MouseDown() :

```
vb  
  
Private Sub Form_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)  
  
If Button = 1 Then  
    Call ReleaseCapture  
    SendMessage Me.hWnd, WM_NCLBUTTONDOWN, HTCAPTION, 0&  
End If  
  
End Sub
```

### Comment énumérer les contrôles d'une form dans une boucle ?

**Auteurs : ThierryAIM ,**

Ce code imprime le nom de chaque controle de Form1 dans la fenêtre de débogage:

```
vb  
  
Dim Ctrl As Control  
  
For Each Ctrl in Form1.Controls  
    Debug.Print Ctrl.Name  
Next
```

**lien : Comment redimensionner les contrôles quand la form est redimensionnée ?**

### Comment savoir si un contrôle appartient à un groupe de contrôles ?

**Auteurs : Catbull ,**

Cette fonction reçoit en paramètre un objet *Control* et retourne :

- Vrai si le contrôle appartient à un groupe
- Faux dans le cas contraire

```
vb  
  
Private Function isGroupe(C As Control) As Boolean  
    On Error Resume Next  
    isGroupe = (C.Index >= 0)  
End Function
```

L'exemple suivant liste dans la fenêtre d'exécution tous les contrôles d'une form, avec leur nom et index pour ceux qui appartiennent à un groupe :

```
vb  
  
Private Sub Command1_Click()
```

```
vb
Dim C As Control

For Each C In Form1.Controls
    If isGroupe(C) Then
        Debug.Print "Index du controle " & C.Name & " : " & C.Index
    Else
        Debug.Print "Le controle " & C.Name & " n'a pas d'index"
    End If
Next C
End Sub
```

## Comment ouvrir une même form plusieurs fois ?

**Auteurs : Romain Puyfoulhoux , hpj ,**

Le module où se trouve le code source d'une form est en fait un module de classe. Un module de classe décrit, comme son nom l'indique, une classe, qui une fois instanciée devient un objet. Donc une form chargée en mémoire pendant l'exécution du programme est en fait une instance de classe.

Si vous avez une form nommée "Form1" dans votre projet, Visual Basic autorise cette syntaxe :

```
vb
Form1.show
```

Etonnant, quand on considère que Form1 est le nom d'une classe que l'on peut instancier normalement.

Pour ouvrir plusieurs fois une form, il suffit d'en créer plusieurs instances :

```
vb
Dim MaNouvelleForm As New Form1
Dim EncoreUneNouvelleForm As New Form1

MaNouvelleForm.Show
EncoreUneNouvelleForm.Show
```

## Comment afficher une form dont les dimensions ne dépendent pas de la résolution ?

**Auteurs : Romain Puyfoulhoux ,**

Les dimensions d'une fenêtre, en nombre de pixels, ne changent pas en fonction de la résolution de l'écran. Donc plus la résolution est importante plus une fenêtre paraît petite à l'écran. C'est le fonctionnement normal d'une interface graphique, mais si vous souhaitez afficher une form ayant proportionnellement toujours la même taille, voici comment procéder.

Affichez la fenêtre de présentation des feuilles (Menu affichage -> présentation des feuilles). Dans cette fenêtre, faites un clic droit et cochez "guides de résolution". Ajustez la taille de la feuille de telle sorte qu'elle ait les proportions souhaitées dans la plus petite résolution supportée par votre application, par exemple 640 x 480 ou bien 800 x 600. L'ajustement de ses dimensions dans les résolutions supérieures est ensuite géré par ce code source, que vous pouvez copier dans le module de la form.

```
vb
```



```
vb
Private Sub Form_Load()

'Résolution correspondant à la form telle qu'elle est en mode conception
Const ResolutionRefX As Long = 640
Const ResolutionRefY As Long = 480

'Rapport entre la résolution actuelle et celle de référence
Dim RatioX As Single
Dim RatioY As Single

'Résolution actuelle
Dim ResolutionX As Long
Dim ResolutionY As Long

ResolutionX = Screen.Width / Screen.TwipsPerPixelX
ResolutionY = Screen.Height / Screen.TwipsPerPixelY

RatioX = ResolutionX / ResolutionRefX
RatioY = ResolutionY / ResolutionRefY

'Adapte les dimensions en fonction de la résolution actuelle
ResizeForResolution RatioX, RatioY

End Sub

Private Sub ResizeForResolution(ByVal RatioX As Single, ByVal RatioY As Single)

Dim ctl As Control
Dim RatioPolices As Single

RatioPolices = (RatioX + RatioY) / 2

Me.Width = Me.Width * RatioX
Me.Height = Me.Height * RatioY

For Each ctl In Me.Controls
If TypeOf ctl Is ComboBox Then
ctl.Move ctl.Left * RatioX, ctl.Top * RatioY, ctl.Width * RatioX
Else
ctl.Move ctl.Left * RatioX, ctl.Top * RatioY, ctl.Width * RatioX, ctl.Height * RatioY
End If
If TypeOf ctl Is Label Then ctl.FontSize = ctl.FontSize * RatioPolices
Next

End Sub
```

La procédure `ResizeForResolution` redimensionne tous les contrôles de la form et agrandit les polices des labels.

## Comment détecter le mouvement d'une form ?

Auteurs : Romain Puyfoulhoux ,

En VB, aucun événement ne permet d'être averti lorsqu'une form a été déplacée. Mais pour Windows l'événement existe. Il est donc possible de l'intercepter grâce au sous classement.

Copiez ce code source dans le module de la form.

```
vb
Private Sub Form_Load()
'Remplace la procédure de fenêtre par défaut par notre propre procédure
```

```

vb
    oldWndProc = SetWindowLong(hwnd, GWL_WNDPROC, AddressOf WindowProc)
End Sub

Private Sub Form_Unload(Cancel As Integer)
    'Remet la procédure de fenêtre par défaut
    SetWindowLong hwnd, GWL_WNDPROC, oldWndProc
End Sub
    
```

Et celui-ci dans un module standard.

```

vb

Public Declare Function SetWindowLong Lib "user32" Alias "SetWindowLongA" _
    (ByVal hwnd As Long, ByVal nIndex As Long, _
    ByVal dwNewLong As Long) As Long

Public Const GWL_WNDPROC = (-4)
Public oldWndProc As Long

Private Declare Function CallWindowProc Lib "user32" Alias "CallWindowProcA" _
    (ByVal lpPrevWndFunc As Long, ByVal hwnd As Long, _
    ByVal msg As Long, ByVal wParam As Long, ByVal lParam As
    Long) As Long
Private Const WM_MOVE = &H3

Public Function WindowProc(ByVal hwnd As Long, ByVal msg As Long, _
    ByVal wParam As Long, ByVal lParam As Long) As Long

    If msg = WM_MOVE Then
        'la form a été déplacée
        Form1.Cls
        Form1.Print "Nouvelle position : " & Form1.Left & ", " & Form1.Top
    End If

    'Appelle la procédure de fenêtre par défaut pour que Windows puisse traiter l'évènement
    WindowProc = CallWindowProc(oldWndProc, hwnd, msg, wParam, lParam)

End Function
    
```

**Attention, la procédure Form\_Unload doit obligatoirement être exécutée. Si vous déboguez et cliquez sur Stop, l'éditeur VB plantera. Si vous fermez votre programme avec l'instruction End, la procédure Form\_Unload ne sera pas exécutée et votre programme plantera.**

**lien : Qu'est-ce que le sous classement ?**

### Comment centrer une feuille MDI fille par rapport à la MDI mère ?

**Auteurs : Romain Puyfoulhoux ,**

**Ces quelques lignes sont placées dans le module de la MDI mère et ouvrent la MDI fille nommée Form1.**

```

vb

Load Form1
Form1.Left = (Me.ScaleWidth - Form1.Width) / 2
Form1.Top = (Me.ScaleHeight - Form1.Height) / 2
    
```

vb

Form1.Show

## Comment régler la transparence d'une fenêtre ?

Auteurs : ridan ,

Fonctionne sous Win 2000 et supérieur. Placez ce code dans un module :

vb

```
Private Declare Function SetLayeredWindowAttributes Lib "user32.dll" ( _
    ByVal hwnd As Long, _
    ByVal crKey As Long, _
    ByVal bAlpha As Byte, _
    ByVal dwFlags As Long) As Long

Private Declare Function SetWindowLong Lib "user32.dll" Alias "SetWindowLongA" ( _
    ByVal hwnd As Long, _
    ByVal nIndex As Long, _
    ByVal dwNewLong As Long) As Long

Private Const WS_EX_LAYERED As Long = &H80000
Private Const LWA_ALPHA As Long = &H2
Private Const GWL_EXSTYLE As Long = -20

Public Sub Transparence(Forme As Form, alpha As Long)
    SetWindowLong Forme.hwnd, GWL_EXSTYLE, WS_EX_LAYERED
    SetLayeredWindowAttributes Forme.hwnd, 0, 255 * alpha / 100, LWA_ALPHA
End Sub
```

lien : [Page sources : un contrôle transparent](#)

## Comment faire réapparaître le "Underscore \_" dans les menus de ma form ?

Auteurs : argyronet ,

Depuis la version de Windows 2000 SP2, les raccourcis clavier, matérialisés par \_ ne sont pas visibles sauf lorsque l'on appuie sur ALT justement.

(Une option de configuration est disponible au niveau de Windows.)

Cette astuce permet de restaurer le processus, via le code VB6 :

vb

```
Private Declare Function SystemParametersInfo Lib "user32" Alias "SystemParametersInfoA" _
    (ByVal uAction As Long, ByVal uParam As Long, ByVal lpvParam As Any, ByVal fuWinIni As Long) As Long

Private Sub SetMenuUnderlines(ByVal ShowUnderscore As Long, Optional ByVal UpdateSystem As Boolean = False)
    Call SystemParametersInfo(SPI_SETMENUUNDERLINES, 0, ByVal ShowUnderscore, _
    SPIF_SENDWININICHANGE Or IIf(UpdateSystem, SPIF_UPDATEINIFILE, 0))
End Sub

Private Sub cmdAmpersandOperation_Click()
    Const HIDE_AMPERSAND As Integer = &H0
    Const SHOW_AMPERSAND As Integer = &H1

    Dim lngMenusAreUnderlined As Long
```

vb

```
Dim blnUpdateSystem As Boolean
Dim intWhatDoWeDo As Integer

SystemParametersInfo SPI_GETMENUUNDERLINES, 0, lngMenusAreUnderlined, 0
If lngMenusAreUnderlined Then
    If MsgBox("Les légendes de menus dotés d'un & sont affichées..." & vbCrLf & _
    "Voulez-vous les définir comme masquées pour la prochaine session ?", vbQuestion + vbYesNo) = 6 Then
        intWhatDoWeDo = HIDE_AMPERSAND
        blnUpdateSystem = MsgBox("Voulez-vous que ce soit pour de bon ?", _
    vbQuestion + vbYesNo, "Sauver dans le système") = 6
    Else
        intWhatDoWeDo = SHOW_AMPERSAND
    End If
Else
    If MsgBox("Les légendes de menus dotés d'un & sont masquées..." & vbCrLf & _
    "Voulez-vous les définir comme affichées pour la prochaine session ?", vbQuestion + vbYesNo) = 6 Then
        intWhatDoWeDo = SHOW_AMPERSAND
        blnUpdateSystem = MsgBox("Voulez-vous que ce soit pour de bon ?", _
    vbQuestion + vbYesNo, "Sauver dans le système") = 6
    Else
        intWhatDoWeDo = HIDE_AMPERSAND
    End If
End If
SetMenuUnderlines intWhatDoWeDo, blnUpdateSystem
Unload Me
End
End Sub
```

Sommaire > Interface > Contrôles

### Comment choisir entre une MSFlexgrid et une Datagrid ?

**Auteurs :** Romain Puyfoulhoux ,

La datagrid est réservée à l'utilisation d'une grille liée à une source de données. Vous pouvez modifier le format d'affichage de chaque colonne. L'utilisateur peut saisir les valeurs au clavier, ajouter et supprimer des lignes. Toutes ces modifications sont prises en compte dans la base de données, avec très peu voire aucune programmation. Cependant, les choses se compliquent très vite dès que vous voulez modifier des données portant sur plusieurs tables de la base.

Quant à la MSFlexgrid, son inconvénient majeur est la non prise en charge de saisie de données. Il reste cependant plusieurs façons de combler ce manque avec quelques lignes de code. Elle peut être liée à une source de données comme la datagrid, mais le format d'affichage n'est pas modifiable. Vous pouvez aussi remplir la grille par programmation, ligne par ligne, ce qui vous permet d'afficher exactement ce que vous voulez; par contre cela peut être moins rapide à l'exécution sur un très grand nombre de lignes. A cela, il faut ajouter les possibilités de modifier la police, la couleur du texte, la couleur de fond d'une cellule, d'afficher une image dans une cellule, et de fusionner plusieurs cellules adjacentes qui contiennent la même valeur. La MSFlexgrid est donc à mon avis préférable dans la plupart des cas.

### Comment permettre à l'utilisateur de taper du texte dans une flexgrid ?

**Auteurs :** Romain Puyfoulhoux ,

Ce programme ajoute dans la cellule courante les caractères au fur et à mesure qu'ils sont tapés. Une autre manière de faire consiste à placer un textbox qui recouvre la cellule courante, et de transférer le texte du textbox dans la cellule chaque fois que l'on quitte le textbox (évènement lostfocus).

```
vb
Dim strTexte

strTexte = MSFlexGrid1.Text
If KeyAscii = 8 Then
    'Touche d'effacement
    If Len(strTexte) > 0 Then MSFlexGrid1.Text = Left(strTexte, Len(strTexte) - 1)
ElseIf KeyAscii <> 13 Then
    MSFlexGrid1.Text = strTexte & Chr(KeyAscii)
End If
```

### Comment avoir une case à cocher dans une flexgrid (2 méthodes)?

**Auteurs :** Romain Puyfoulhoux , Da40 , ridan ,

La méthode la plus simple consiste à utiliser des images représentant chacune un des états possibles de la case à cocher. Ici nous aurons deux images: une pour représenter une case non cochée et une autre pour la case cochée. Ces images sont stockées dans deux simples contrôles images.

Dans toutes les cellules qui doivent contenir une case à cocher, nous insérons l'image correspondant à l'état courant de la case à cocher.

Les deux procédures du code source ci-dessous montrent comment afficher la case à cocher. La procédure `InitCelluleAvecCase()` initialise une cellule pour qu'elle puisse contenir une case à cocher. Elle appelle la procédure `AfficheCelluleAvecCase()` qui elle se contente d'afficher la bonne image.

```
vb
```

vb

```

Private Sub InitCelluleAvecCase(ligne As Long, colonne As Long, Optional valeur As Boolean = False)

MSFlexGrid1.Col = colonne
MSFlexGrid1.Row = ligne
MSFlexGrid1.CellPictureAlignment = flexAlignCenterCenter
AfficheCelluleAvecCase ligne, colonne, valeur

End Sub

Private Sub AfficheCelluleAvecCase(ligne As Long, colonne As Long, Optional valeur As
Boolean = False)

Set MSFlexGrid1.CellPicture = IIf(valeur, imgCaseCochee.Picture, imgCase.Picture)

End Sub

```

Voici un exemple complet qui utilise nos deux procédures. Au chargement de la form, la msflexgrid est initialisée pour afficher un tableau de commandes. Dans la troisième colonne de la grille, une case à cocher indique si la commande a été envoyée.

vb

```

Private Type commande
    reference As String
    montant As Single
    envoyee As Boolean
End Type

Dim commandes(0 To 2) As commande

Private Sub Form_Load()

Dim i As Long

'tableau de commandes
commandes(0).reference = "cmd0001": commandes(0).montant = 80: commandes(0).envoyee = True
commandes(1).reference = "cmd0002": commandes(1).montant = 150.2: commandes(1).envoyee = False
commandes(2).reference = "cmd0003": commandes(2).montant = 95.5: commandes(2).envoyee = True

'initialisation de la msflexgrid
MSFlexGrid1.Cols = 3
MSFlexGrid1.Rows = 4
MSFlexGrid1.FixedCols = 0
For i = 0 To 2
    MSFlexGrid1.TextMatrix(i + 1, 0) = commandes(i).reference
    MSFlexGrid1.TextMatrix(i + 1, 1) = commandes(i).montant
    InitCelluleAvecCase i + 1, 2, commandes(i).envoyee
Next

End Sub

Private Sub MSFlexGrid1_Click()

If MSFlexGrid1.Col = 2 And MSFlexGrid1.MouseRow > 0 Then
    commandes(MSFlexGrid1.Row - 1).envoyee = Not commandes(MSFlexGrid1.Row - 1).envoyee
    AfficheCelluleAvecCase MSFlexGrid1.Row, 2, commandes(MSFlexGrid1.Row - 1).envoyee
End If

End Sub

```

Méthode testée sous Win XP :

vb

```
Private Const Checked As Byte = 253
Private Const UnChecked As Byte = 168
Private Sub Form_Load()

    Dim CCol, CRow As Integer

    With MSFlexGrid1

        .FixedCols = 0
        .FixedRows = 1
        .Cols = 8
        .Rows = 8
        .ColAlignment(2) = flexAlignCenterCenter

        For CRow = .FixedRows To .Rows - 1

            For CCol = 0 To .Cols - 1
                .TextMatrix(CRow, CCol) = "Cel " & CRow & ", " & CCol
            Next CCol

            .Col = 2
            .Row = CRow
            .CellFontName = "Wingdings"
            .CellFontSize = 12
            .CellFontBold = False
            .Text = Chr(UnChecked)

        Next CRow

    End With

End Sub

Private Sub MSFlexGrid1_Click()

    With MSFlexGrid1

        If .Col = 2 Then
            If .Text = Chr(UnChecked) Then
                .Text = Chr(Checked)
            ElseIf .Text = Chr(Checked) Then
                .Text = Chr(UnChecked)
            End If
        End If

    End With

End Sub
```

## Comment n'autoriser la sélection que d'une seule ligne dans une MSFlexgrid ?

**Auteurs : Romain Puyfoulhoux ,**

**Il n'existe pas de propriété pour le faire mais voici une astuce possible :**

vb

```
Private Sub MSFlexGrid1_SelChange()
MSFlexGrid1.RowSel = MSFlexGrid1.Row
End Sub
```

Si vous essayez de sélectionner plusieurs lignes à la souris, vous verrez un clignotement que vous pouvez éviter en empêchant le rafraîchissement automatique de la grille tant que le bouton de la souris est enfoncé :

vb

```
Private Sub MSFlexGrid1_MouseDown(Button As Integer, Shift As Integer, x As Single, y As Single)
MSFlexGrid1.Redraw = False
End Sub

Private Sub MSFlexGrid1_MouseUp(Button As Integer, Shift As Integer, x As Single, y As Single)
MSFlexGrid1.Redraw = True
End Sub
```

## Comment exporter le contenu d'une msflexgrid dans un fichier ?

Auteurs : Sygale , Romain Puyfoulhoux ,

Cette fonction exporte le contenu de la msflexgrid passée en paramètre dans le fichier strFileName. Les colonnes sont séparées par le caractère passé dans le troisième paramètre.

vb

```
Public Function ExportGridToFile(Mygrid As MSFlexGrid, ByVal strFileName as string, _
Optional ByVal strSep As String = vbTab) As Boolean

Dim intFreeFile As Integer           'Numéro du fichier
Dim intCol As Integer, intRow As Integer 'Indices de ligne et colonne de W
Dim ligne As String                 'La ligne à écrire dans le fichier

On Error GoTo ExportGridToFile_ERR

'Prend le prochain numéro de fichier
intFreeFile = FreeFile

'Ouvre le fichier en bloquant son accès aux autres applications
Open strFileName For Output Access Write Lock Read Write As #intFreeFile

With Mygrid
'Pour chaque ligne
For intRow = .FixedRows To .Rows - 1
    ligne = ""
    'Pour chaque colonne
    For intCol = .FixedCols To .Cols - 1
        'Ajoute la valeur de la cellule
        ligne = ligne & .TextMatrix(intRow, intCol) & strSep
    Next intCol
    'Enlève le séparateur final
    If strSep <> "" Then ligne = Left(ligne, Len(ligne) - 1)
    Print #intFreeFile, ligne
Next intRow
End With

'Valide le bon fonctionnement de la fonction
ExportGridToFile = True

ExportGridToFile_FIN:
Close #intFreeFile + 1
Exit Function

ExportGridToFile_ERR:
ExportGridToFile = False
Resume ExportGridToFile_FIN
```



```
vb
End Function
```

## Comment modifier la largeur des colonnes d'une MSFlexgrid en fonction de la longueur du texte ?

Auteurs : Romain Puyfoulhoux ,

Copiez ce code dans un module standard :

```
vb

Private Type Size
    cx As Long
    cy As Long
End Type

Private Declare Function GetTextExtentPoint32 Lib "gdi32" Alias "GetTextExtentPoint32A" _
    (ByVal hdc As Long, ByVal lpsz As String, _
    ByVal cbString As Long, lpSize As Size) As Long

Public Sub ResizeColumns(hdc As Long, flexgrid As MSFlexGrid)

    Dim idxRow As Long, idxCol As Long, lngMax As Long
    Dim texte As String, taille As Size

    With flexgrid
        'Parcoure les colonnes
        For idxCol = 0 To .Cols - 1
            lngMax = 0
            'Parcoure les lignes
            For idxRow = 0 To .Rows - 1
                texte = .TextMatrix(idxRow, idxCol)
                'met la taille du texte de la cellule en pixels dans taille
                GetTextExtentPoint32 hdc, texte, Len(texte), taille
                'lngMax est la longueur du texte le plus long dans cette colonne
                If taille.cx > lngMax Then lngMax = taille.cx
            Next
            'met lngMax en twips en ajoutant 10 pixels pour les marges
            If lngMax > 0 Then lngMax = (lngMax + 10) * Screen.TwipsPerPixelX
            'Applique la largeur de colonne si besoin
            If lngMax > .ColWidth(idxCol) Then .ColWidth(idxCol) = lngMax
        Next
    End With

End Sub
```

Voici comment appeler la procédure `ResizeColumns` :

```
vb

ResizeColumns Me.hdc, MSFlexGrid1
```

La procédure attend un contexte de périphérique en premier paramètre. Le contexte de périphérique est renvoyé par la propriété `hdc`. Si la form et la `MSFlexGrid` ont la même police, vous pouvez passer le contexte de périphérique de la

form. Sinon vous pouvez ajouter un PictureBox invisible qui aura la même police que celle de la MSFlexgrid et passer sa propriété hdc en premier paramètre.

### Comment adapter les dimensions d'une image à celle d'un PictureBox ?

Auteurs : Romain Puyfoulhoux ,

vb

```
Picture1.PaintPicture Image1.Picture, 0, 0, Picture1.Width, Picture1.Height
```

### Comment contrôler les caractères qui peuvent être saisis dans un textbox ?

Auteurs : Romain Puyfoulhoux ,

Une solution consiste à utiliser l'évènement **KeyPress**, qui a lieu lorsqu'une touche correspondant à un caractère est enfoncée. Les touches comme shift, alt, control et F1 à F12 ne sont pas concernées. La procédure de cet évènement a un argument, **KeyAscii**, qui est le code du caractère à afficher. Modifiez sa valeur pour afficher le caractère que vous voulez. Donnez-lui une valeur nulle si aucun caractère ne doit être affiché. La fonction **chr()** renvoie le caractère dont le code est passé en paramètre.

L'exemple suivant interdit tout caractère autre que les chiffres et la touche d'effacement :

vb

```
Private Sub Text1_KeyPress(KeyAscii As Integer)

If KeyAscii <> 8 Then
    If Not IsNumeric(Chr(KeyAscii)) Then KeyAscii = 0
End If
End Sub
```

Une astuce souvent utilisée consiste à rechercher le caractère entré, dans une chaîne contenant tous les caractères autorisés. Si ce caractère n'est pas dans la chaîne, rien n'est affiché :

vb

```
Private Sub Text1_KeyPress(KeyAscii As Integer)

Dim allowedKeys As String
allowedKeys = "0123456789-.,." & Chr(8)
If InStr(allowedKeys, Chr(KeyAscii)) = 0 Then KeyAscii = 0
End Sub
```

### Comment annuler la dernière modification du contenu d'un textbox (undo) ?

Auteurs : Romain Puyfoulhoux ,

Une solution possible est de simuler l'appui sur les touches **ctrl + z** :

vb

```
Text1.SetFocus
SendKeys "^z"
```

Mais vous pouvez aussi envoyer le message EM\_UNDO au textbox, grâce à la fonction SendMessage() de l'Api Windows. Ajoutez tout d'abord ces déclarations :

```
vb
Private Declare Function SendMessage Lib "user32" Alias "SendMessageA" (ByVal hwnd As Long, _
    ByVal wParam As Long, ByVal lParam As Any) As Long
Private Const EM_UNDO = &HC7
```

Pour annuler la dernière modification sur le contenu de Text1 :

```
vb
SendMessage Text1.hwnd, EM_UNDO, 0, 0
```

## Comment récupérer une par une les lignes d'un textbox multilignes ?

Auteurs : Romain Puyfoulhoux ,

Une première idée serait d'utiliser la fonction Split() avec vbCrLf comme séparateur. Mais une fin de ligne n'est pas forcément due à un retour chariot. Nous allons plutôt faire appel aux API Windows.

Copiez tout d'abord ces déclarations au début du module de la form :

```
vb
Private Declare Function SendMessage Lib "user32" Alias "SendMessageA" _
    (ByVal hwnd As Long, ByVal wParam As Long, _
    ByVal lParam As Any) As Long
Private Declare Sub CopyMemory Lib "kernel32" Alias "RtlMoveMemory" (Destination As Any, Source As
    Any, _
    ByVal Length As Long)
Private Const EM_GETLINECOUNT = &HBA
Private Const EM_LINELENGTH = &HC1
Private Const EM_LINEINDEX = &HBB
Private Const EM_GETLINE = &HC4
```

La procédure ci-dessous affiche une par une les lignes du textbox dont le handle est passé en paramètre :

```
vb
Private Sub AfficheLignes(lngHandleTextBox As Long)
Dim lngNbLignes As Long, i As Long
Dim lngIndexCar As Long, lngLongueurLigne As Integer
Dim strLigne As String
'nombre de lignes
lngNbLignes = SendMessage(lngHandleTextBox, EM_GETLINECOUNT, 0, 0)
For i = 1 To lngNbLignes
    'index du premier caractère de la ligne
    lngIndexCar = SendMessage(lngHandleTextBox, EM_LINEINDEX, i - 1, 0)
    'longueur de la ligne
    lngLongueurLigne = SendMessage(lngHandleTextBox, EM_LINELENGTH, lngIndexCar, 0)
    'récupère la ligne dans la chaîne strLigne
    strLigne = Space(lngLongueurLigne)
```

```
vb
CopyMemory ByVal strLigne, intLongueurLigne, Len(intLongueurLigne)
SendMessage lngHandleTextBox, EM_GETLINE, i - 1, ByVal strLigne
MsgBox strLigne
Next
End Sub
```

## Comment afficher son propre menu popup dans un textbox ?

**Auteurs : Romain Puyfoulhoux ,**

Créez un menu dans la form et rendez-le invisible. Puis créez des sous-menus visibles. Ce sont ces derniers qui apparaîtront dans le menu popup.

Tout se passe dans la procédure **MouseDown**. Afin d'empêcher le menu contextuel de Windows de s'afficher lorsque l'utilisateur va relâcher la souris, nous désactivons le textbox puis le réactivons immédiatement.

```
vb
Private Sub Text1_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
    If Button = vbRightButton Then
        Text1.Enabled = False
        Text1.Enabled = True
        PopupMenu menuTextBox 'menuTextBox est le nom du menu invisible
    End If
End Sub
```

## Comment utilise-t-on le contrôle TabStrip ?

**Auteurs : Romain Puyfoulhoux , Jean-Marc Rabilloud ,**

Le contrôle **TabStrip** représente un ensemble d'onglets. Mais il ne permet pas d'associer des contrôles (textbox, commandbutton, etc...) à chacun des onglets. Vous devez donc gérer vous même l'affichage des contrôles en fonction de l'onglet qui est actif. Le plus simple est d'utiliser des frames indexées, et de mettre au premier plan la frame correspondant à l'onglet activé. Ici la première frame est d'index 1 :

```
vb
Private Sub TabStrip1_Click()
    frameOnglet(TabStrip1.SelectedItem.Index).ZOrder 0
End Sub
```

Toutefois, on utilise plus souvent le contrôle **Tabstrip** lorsque plusieurs onglets utilisent une même frame. Dans ce cas on met dans la propriété **tag** de chaque objet **Tab** l'index de la frame qu'il doit mettre au premier plan. Le code devient alors :

```
vb
Private Sub TabStrip1_Click()
    frameOnglet(TabStrip1.SelectedItem.Tag).ZOrder 0
```

```
vb  
End Sub
```

Il n'est pas la peine d'aligner ni de dimensionner les Frames de votre TabStrip. En général on utilise un code d'alignement dans l'événement Load de la feuille semblable à celui-ci :

```
vb  
  
For compteur = 0 To frameOnglet.Count-1  
    With TabStrip1  
        frameOnglet(compteur).Move .ClientLeft, .ClientTop, .ClientWidth, .ClientHeight  
    End With  
Next compteur
```

### Comment se connecter à une base Access 2000 avec un contrôle Data ?

Auteurs : Romain Puyfoulhoux ,

Pour cela, DAO 3.6 et Jet 4.0 doivent être installés. Dans les références de votre projet, enlevez "Microsoft DAO 3.5 Object Library" et sélectionnez "Microsoft DAO 3.6 Object Library". Puis utilisez ce code source pour vous connecter :

```
vb  
  
Dim daoDB36 As Database  
Dim rs As Recordset  
Dim sPath As String  
  
sPath = "c:\base.mdb"  
Set daoDB36 = DBEngine.OpenDatabase(sPath)  
Set rs = daoDB36.OpenRecordset("tClients")  
Set Data1.Recordset = rs
```

### Comment imprimer le contenu d'un RichTextBox ?

Auteurs : Romain Puyfoulhoux ,

Un code source vous permettant d'imprimer un RichTextBox, avec la gestion des pages, est dans  cet article de MSDN.

### Comment concaténer le contenu de deux contrôles RTF ?

Auteurs : Bazoom ,

Sur une feuille, placez 2 contrôles RichTextBox avec leur texte respectif et un bouton de commande Command1. Placez ensuite ce code dans le module de la form.

```
vb  
  
Private Function ConcatRTF(RTF1 As String, RTF2 As String) As String  
  
    ConcatRTF = Left(RTF1, InStrRev(RTF1, "}") - 1) & Mid(RTF2, 2)  
  
End Function
```

```
vb
Private Sub Command1_Click()

    With Me.RichTextBox1
        .TextRTF = ConcatRTF(.TextRTF, Me.RichTextBox2.TextRTF)
        .Refresh
    End With

End Sub
```

## Comment mettre plusieurs colonnes dans une ComboBox ?

Auteurs : [Alexandre Lokchine](#) , [Khany](#) ,

Ce n'est pas possible avec la ComboBox standard. Par contre, celle incluse dans le composant Microsoft Forms 2.0 dispose de cette fonctionnalité.

- Ajoutez le composant Microsoft Forms 2.0 Object Library dans votre projet. Cela a pour effet d'ajouter des contrôles dans la boîte à outils, dont le contrôle ComboBox (il a le même nom que la ComboBox standard de VB mais il est plus évolué).
- Ajoutez un nouveau contrôle ComboBox sur votre Form.
- Modifiez sa propriété ColumnCount à la valeur souhaitée.

L'exemple suivant montre comment remplir une ComboBox à 3 colonnes :

```
vb
Dim i As Integer

For i = 0 To 4
    cboTest.AddItem
    cboTest.List(i, 0) = "Lig " & i & " Col 0"
    cboTest.List(i, 1) = "Lig " & i & " Col 1"
    cboTest.List(i, 2) = "Lig " & i & " Col 2"
Next i
```

Pour qu'une colonne soit cachée, il suffit de mettre sa largeur à 0. Par exemple pour qu'une ComboBox ait 3 colonnes et que les deux premières soient cachées :

```
vb
Combo1.ColumnCount = 3
Combo1.ColumnWidths = "0" & ";" & "0" & ";" & "3975"
```

## Comment dérouler une ComboBox ?

Auteurs : [Romain Puyfoulhoux](#) ,

Habituellement, une ComboBox se déroule quand l'utilisateur clique dessus. Si vous voulez la dérouler avec du code, placez ces déclarations dans un module :

```
vb
Private Declare Function SendMessage Lib "user32" Alias "SendMessageA" _
    (ByVal hwnd As Long, ByVal wParam As Long, _
    ByVal lParam As Any) As Long
```

```
vb
Private Const CB_SHOWDROPDOWN = &H14F

Public Sub DerouleCombo(handle As Long)
    SendMessage handle, CB_SHOWDROPDOWN, True, ByVal 0
End Sub
```

Voici comment par exemple dérouler automatiquement une combo quand elle reçoit le focus :

```
vb

Private Sub Combo1_GotFocus()
    DerouleCombo Combo1.hwnd
End Sub
```

## Comment modifier la largeur de la zone de déroulement d'une ComboBox ?

Auteurs : Alexandre Lokchine , ThierryAIM ,

Copiez ce code dans un module standard :

```
vb

Private Const CB_GETLBTEXTLEN = &H149
Private Const CB_SHOWDROPDOWN = &H14F
Private Const CB_GETDROPPEDWIDTH = &H15F
Private Const CB_SETDROPPEDWIDTH = &H160
Private Const CB_MSGMAX = &H15B
Private Const CB_SETITEMHEIGHT = &H153

Private Declare Function SendMessage Lib "user32" Alias "SendMessageA" _
    (ByVal hwnd As Long, ByVal wParam As Long, _
    ByVal lParam As Any) As Long

Public Sub LargeurCombo(hwnd As Long, largeur As Long)

    SendMessage hwnd, CB_SETDROPPEDWIDTH, largeur, ByVal 0

End Sub
```

La procédure LargeurCombo modifie la largeur de la zone de déroulement de la combo dont le handle est passé en paramètre. Pour la tester, posez une combo sur une form et placez code dans le module de la form :

```
vb

Private Sub Form_Load()

    Dim i As Long

    For i = 1 To 20
        Combo1.AddItem "ligne " & i
    Next

    LargeurCombo Combo1.hwnd, 300

End Sub
```

```
vb  
End Sub
```

## Comment modifier la hauteur de la zone de déroulement d'une ComboBox ?

Auteurs : **ThierryAIM**,

Ceci est possible en faisant appel à l'api MoveWindow. Copiez ce code dans module standard.

```
vb  
  
Private Declare Function MoveWindow Lib "user32" (ByVal hwnd As Long, _  
                                                ByVal x As Long, ByVal y As Long, _  
                                                ByVal nWidth As Long, ByVal nHeight As Long, _  
                                                ByVal bRepaint As Long) As Long  
  
Public Sub SetCtrlHeight(vCtrl As Control, vHeight As Long)  
  
    Dim xForm As VB.Form, xCont As Object, XScaleMode As ScaleModeConstants  
  
    'Trouve où est situé le contrôle (Formulaire et Contenant)  
    Set xForm = vCtrl.Parent  
    Set xCont = vCtrl.Container  
  
    'Enlève le contrôle de son contenant pour le mettre sur la form  
    Set vCtrl.Container = xForm  
  
    'Sauvegarde ScaleMode avant de le modifier  
    XScaleMode = xForm.ScaleMode  
  
    'Met ScaleMode du formulaire en Pixels car MoveWindow utilise les pixels  
    xForm.ScaleMode = vbPixels  
  
    'Redimensionne la fenêtre du ComboBox  
    MoveWindow vCtrl.hwnd, vCtrl.Left, vCtrl.Top, vCtrl.Width, vHeight, 1  
  
    'Remet ScaleMode à sa valeur initiale  
    xForm.ScaleMode = XScaleMode  
  
    'Remet le contrôle dans son contenant initial  
    Set vCtrl.Container = xCont  
  
End Sub
```

Voici comment donner à la zone de déroulement d'une combo box une hauteur de 180 pixels :

```
vb  
  
Private Sub Form_Load()  
  
    Dim xHeight As Long  
    xHeight = 180 ' en pixel : affiche 12 lignes en Font MS sans Serif taille 8  
    Call SetCtrlHeight(Combo1, xHeight)
```



```
vb
End Sub
```

## Comment ajouter une fonctionnalité de correspondance à une ComboBox ?

Auteurs : Jean-Marc Rabilloud , Romain Puyfoulhoux ,

Cette fonctionnalité consiste à sélectionner automatiquement l'élément de la liste dont le début correspond aux caractères saisis dans la combo. Si aucun n'élément ne correspond, rien n'est sélectionné.

Copiez tout d'abord ces déclarations au début du module de la form :

```
vb

Private Declare Function SendMessage Lib "user32" Alias "SendMessageA" _
    (ByVal hwnd As Long, ByVal wParam As Long, _
    ByVal lParam As Long, lParam As Any) As Long

Private Const CB_FINDSTRING = &H14C
```

Tout le reste du code se situe dans la procédure de l'événement KeyPress, qui a lieu chaque fois qu'un caractère est entré :

```
vb

Private Sub Combol_KeyPress(KeyAscii As Integer)

    Dim Trouve As Long, Position As Integer, Taille As Integer, strTemp As String

    With Me.Combol
        If KeyAscii = 8 Then
            If .SelStart = 0 Then Exit Sub
            .SelStart = .SelStart - 1
            .SelText = ""
        Else
            Position = .SelStart
            strTemp = .Text
        End If
        .SelText = Chr(KeyAscii)
        Trouve = SendMessage(.hwnd, CB_FINDSTRING, 0, ByVal .Text)
        If Trouve = -1 Then
            'les trois lignes suivantes doivent être enlevées en cas de non correspondance possible
            .Text = strTemp
            .SelStart = Position
            .SelLength = (Len(.Text) - Position)
            KeyAscii = 0
            Exit Sub
        Else
            Position = .SelStart
            Taille = Len(.List(Trouve)) - Len(.Text)
            .SelText = .SelText & Right(.List(Trouve), Taille)
            .SelStart = Position
            .SelLength = Taille
            KeyAscii = 0
        End If
    End With
End Sub
```

```
vb
End Sub
```

## Comment ajouter un ascenseur horizontal à une ListBox ?

Auteurs : Jean-Marc Rabilloud , Khany , Romain Puyfoulhoux ,

Si l'un des éléments d'une ListBox est trop long pour pouvoir être affiché entièrement dans la liste, un ascenseur horizontal serait bien utile, mais ce contrôle n'a pas de propriété permettant de l'indiquer. Afin qu'un ascenseur soit créé quand l'un des éléments dépasse la largeur de la liste, copiez tout d'abord ce code dans un module.

```
vb

Private Declare Function DrawText Lib "user32" Alias "DrawTextA" (ByVal hdc As Long, ByVal
lpStr As String, _
                                                                    ByVal nCount As Long, lpRect As
RECT, _
                                                                    ByVal wFormat As Long) As Long
Private Declare Function GetSystemMetrics Lib "user32" (ByVal nIndex As Long) As Long
Private Declare Function SendMessage Lib "user32" Alias "SendMessageA" _
(ByVal hwnd As Long, ByVal wParam As Long, _
ByVal lParam As Long, lParam As Any) As Long

Private Const LB_SETHORIZONTALEXTENT = &H194
Private Const DT_CALCRECT = &H400
Private Const SM_CXVSCROLL = 2

Private Type RECT
Left As Long
Top As Long
Right As Long
Bottom As Long
End Type

Public Sub ApplyScrollBar(ByVal hdc As Long, MaListe As ListBox)

Dim compteur As Long, Nlargeur As Long, LargText As Long, sysScrollWidth As Long
Dim rcText As RECT

sysScrollWidth = GetSystemMetrics(SM_CXVSCROLL)
For compteur = 0 To MaListe.ListCount - 1
DrawText hdc, MaListe.List(compteur), -1&, rcText, DT_CALCRECT
LargText = rcText.Right + sysScrollWidth
If LargText >= Nlargeur Then
Nlargeur = LargText
End If
Next compteur
SendMessage MaListe.hwnd, LB_SETHORIZONTALEXTENT, Nlargeur, ByVal 0&

End Sub
```

Ensuite, appelez la procédure ApplyScrollBar chaque fois que vous ajoutez ou supprimez un élément dans la liste, ce qui aura pour effet d'afficher ou de supprimer l'ascenseur et d'actualiser sa taille s'il reste visible. Pour tester ce code, placez une listbox sur une form et placez ce code dans le module de la form :

```
vb

Private Sub Form_Load()

List1.AddItem "fjzkzlggj fjekljdsj"
List1.AddItem "fjzkzlggj fjekd jfskdlggjslkdgjklsg fjdsj"
List1.AddItem "fjzkzlggj fjekl"
```

vb

```
ApplyScrollBar Me.hdc, List1
```

```
End Sub
```

## Comment aligner des entrées multicolonnees dans une ListBox ?

Auteurs : Alexandre Lokchine ,

Si vous préférez ne pas utiliser la collection Microsoft Forms 2.0, il existe néanmoins une manière propre de présenter une entrée multicolonnees dans une ListBox :

Les items doivent être insérés dans la liste séparés par des tabulations, par exemple :

vb

```
List1.AddItem "champ1" & vbTab & "champ2" & vbTab & "champ3"
```

Si la police est à chasse fixe (Courier, par exemple), les colonnes sont alors correctement alignées, mais si la police est à chasse variable (ce qui est le cas la plupart du temps), il faut utiliser les API de Windows pour faire un alignement correct, comme suit :

Placez ce code dans un module :

vb

```
Public Declare Function SendMessage Lib "user32" Alias "SendMessageA" _  
    (ByVal hwnd As Long, ByVal wMsg As Long, _  
    ByVal wParam As Long, lParam As Any) As Long  
  
Public Const LB_SETTABSTOPS = &H192
```

Utilisez ensuite le code suivant pour produire une liste avec des champs alignés proprement :

vb

```
Private Sub Form_Load()  
  
    'ce tableau va contenir les positions des tabulations (ce qui equivaut  
    'à la largeur des colonnes  
    ReDim tabstop(0 To 2) As Long  
  
    'on positionne les tabulations  
    tabstop(0) = 90  
    tabstop(1) = 130  
    tabstop(2) = 185  
  
    'on efface puis on réinitialise les tabulations  
    Call SendMessage(List1.hwnd, LB_SETTABSTOPS, 0&, ByVal 0&)  
    Call SendMessage(List1.hwnd, LB_SETTABSTOPS, 3, tabstop(0))  
    List1.Refresh
```

```
vb
End Sub
```

## Comment lier une DataCombo ou une DataList à un champ d'une base de données ?

Auteurs : Khany , Romain Puyfoulhoux ,

Nous allons voir comment placer dans une combo ou une liste les valeurs contenues dans un champ d'une base de données en vue d'une sélection d'un enregistrement. La liaison directe avec la source de données évite la boucle de lecture du recordset ainsi que l'instruction additem et facilite la lecture des autres champs de l'enregistrement sélectionné.

Placez un contrôle DataCombo ou DataList sur une feuille et initialisez une connexion à une base de données; ensuite, insérez ce code :

```
vb

Set DataCombo1.DataSource = rst
Set DataCombo1.RowSource = rst
DataCombo1.ListField = "MonChamp" 'nom du champ
```

Si l'on désire voir l'élément sélectionné en surbrillance, il faut ajouter au code précédent :

```
vb

DataCombo1.DataField = "MonChamp"
```

Pour placer le recordset sur l'élément sélectionné et récupérer la valeur des autres champs de cet enregistrement :

```
vb

Private Sub DataCombo1_Click()

    If DataCombo1 <> "" Then
        rst.Bookmark = DataCombo1.SelectedItem
    End If

    Text1.Text = rst.Fields("AutreChamp") ' nom d'un champ quelconque dans la base
End Sub
```

## Comment tester l'existence d'un noeud dans un contrôle TreeView ?

Auteurs : Alexandre Lokchine ,

Chaque noeud d'un TreeView possède une clé qui est un identifiant unique, assigné au moment de l'insertion et dont on se sert ici afin de tester l'existence d'un noeud:

```
vb

If Treeview.Nodes(cle_a_verifier) Is Nothing Then
    'le noeud n'existe pas
```

```
vb
End If
```

## Comment définir la couleur d'arrière-plan d'un TreeView ?

Auteurs : Alexandre Lokchine , Romain Puyfoulhoux ,

Contrairement à une opinion largement répandue, il est possible de changer la couleur d'arrière-plan d'un TreeView.

Placez ces déclarations dans un module :

```
vb

Private Const GWL_STYLE As Long = (-16)
Private Const TVS_HASLINES As Long = 2
Private Const TV_FIRST As Long = &H1100
Private Const TVM_SETBKCOLOR As Long = (TV_FIRST + 29)
Private Const TVM_SETTEXTCOLOR As Long = (TV_FIRST + 30)

Private Declare Function SendMessage Lib "user32" Alias "SendMessageA" _
    (ByVal hwnd As Long, ByVal wParam As Long, _
    ByVal lParam As Any) As Long

Private Declare Function GetWindowLong Lib "user32" Alias "GetWindowLongA" _
    (ByVal hwnd As Long, ByVal nIndex As Long) As Long

Private Declare Function SetWindowLong Lib "user32" Alias "SetWindowLongA" _
    (ByVal hwnd As Long, ByVal nIndex As Long, _
    ByVal dwNewLong As Long) As Long
```

Ensuite, placez la procédure suivante dans la section générale du formulaire:

```
vb

Public Sub SetTreeViewArrPlan(ByVal treeview As MSCOMCTL.TreeView, ByVal couleur As Long)

    Dim style As Long, noeud As Node

    'Changement de l'arrière-plan du treeview
    Call SendMessage(treeview.hwnd, TVM_SETBKCOLOR, 0, ByVal couleur)

    'réinitialisation de l'arbre
    style = GetWindowLong(treeview.hwnd, GWL_STYLE)

    'Si l'arbre a des lignes, on les désactive temporairement pour que
    'l'arrière-plan se redessine proprement, puis on les réactive

    If style And TVS_HASLINES Then
        SetWindowLong treeview.hwnd, GWL_STYLE, style Xor TVS_HASLINES
        SetWindowLong treeview.hwnd, GWL_STYLE, style
    End If

    'on change la couleur de fond des noeuds
    For Each noeud In treeview.Nodes
        noeud.BackColor = couleur
    Next

End Sub
```

L'appel se fait comme suit :

vb

```
Dim couleur As Long
couleur = vbRed
SetTreeViewArrPlan TreeView1, couleur
```

## Comment donner un effet d'ombre à un contrôle ?

Auteurs : Khany ,

Dans un module standard, placez ce code :

vb

```
Public Const RELIEF = 1
Public Const ENCADREMENT = 2

Public Sub Ombrage(Feuille As Form, Ctrl As Control, Effet As Integer, OmbreLarg As Integer,
OmbreCoul As Long)

    Dim CouleurOmbre As Long
    Dim LargeurOmbre As Integer
    Dim Largeur As Integer
    Dim Taille As Integer

    LargeurOmbre = OmbreLarg
    CouleurOmbre = OmbreCoul
    Largeur = Feuille.DrawWidth
    Taille = Feuille.ScaleMode
    Feuille.DrawWidth = 1

    Select Case Effet
        Case ENCADREMENT
            Feuille.Line (Ctrl.Left + LargeurOmbre, Ctrl.Top + LargeurOmbre)- _
                Step(Ctrl.Width - 1, Ctrl.Height - 1), CouleurOmbre, BF

        Case RELIEF
            Feuille.Line (Ctrl.Left - LargeurOmbre, Ctrl.Top - LargeurOmbre)- _
                Step(Ctrl.Width - 1, Ctrl.Height - 1), CouleurOmbre, BF
    End Select

    Feuille.DrawWidth = Largeur
    Feuille.ScaleMode = Taille

End Sub
```

Posez, par exemple, un PictureBox sur votre feuille, et placez ce code dans le module de la feuille :

vb

```
Private Sub Form_Paint()

    Ombrage Me, Picture1, ENCADREMENT, 10, QBColor(5) 'contour du cadre accentué
    Ombrage Me, Picture1, RELIEF, 10, QBColor(5) 'effet d'enfoncement de l'image
```

```
vb
End Sub
```

### Comment savoir s'il y a une image dans un contrôle ?

**Auteurs : Jean-Marc Rabilloud , Alexandre Lokchine ,**

La fonction ci-dessous renvoie Vrai si le contrôle passé en paramètre contient une image.

```
vb

Public Function ImagePresente(controle As Control) As Boolean

    ImagePresente = (controle.Picture.Handle > 0)

End Function
```

### Comment savoir si une propriété existe pour un contrôle ?

**Auteurs : DarkVader , ThierryAIM ,**

Il peut être utile de savoir si une propriété existe pour un contrôle donné, dans une boucle par exemple, afin d'éviter les erreurs

Insérez ce code dans la section déclaration de votre form ou dans un module :

```
vb

Public Function ExistProperty(Obj As Object, ByVal PropertyName As String) As Boolean
    On Error Resume Next
    CallByName Obj, PropertyName, VbGet
    ExistProperty = (Err.Number = 0)
    Err.Clear
End Function
```

**Exemple d'utilisation :**

```
vb

Dim Ctrl As Control
For Each Ctrl In Me.Controls
    If Not ExistProperty(Ctrl, "ForeColor") Then
        Debug.Print "La propriété ForeColor n'existe pas pour le controle " & Ctrl.Name
    End If
Next
```

### Comment dérouler un bouton DropDown dans une ToolBar en cliquant sur l'ensemble du bouton ?

**Auteurs : Bazoom , Romain Puyfoulhoux , ThierryAIM ,**

Dans Projets -> Composants -> Cocher Microsoft Windows Common Controls 6.0 (SP4) Le code suivant ne fonctionne que sous Win 2000 et supérieur mais le deuxième code proposé palie à ce manque et fonctionne sous Win 98.

Sur une feuille, poser un contrôle ToolBar et copier le code dans la partie déclaration :

vb

```

Private Type POINTAPI
    X As Long
    Y As Long
End Type

Private Declare Function GetCursorPos Lib "user32" (lpPoint As POINTAPI) As Long
Private Declare Function SetCursorPos Lib "user32" (ByVal X As Long, ByVal Y As Long) As Long
Private Declare Sub mouse_event Lib "user32" (ByVal dwFlags As Long, ByVal dx As Long, _
    ByVal dy As Long, ByVal cButtons As Long, ByVal dwExtraInfo As Long)

Private Sub Form_Load()
    Dim i As Integer
    Dim btn As Button

    ' Ajoute un objets Button au contrôle Toolbar.
    Set btn = Toolbar1.Buttons.Add(Caption:="Test", Style:=tbrDropdown)
    ' Ajoute deux objets ButtonMenu à l'objet Button.
    btn.ButtonMenus.Add Text:="Option 1"
    btn.ButtonMenus.Add Text:="Option 2"
End Sub

Private Sub Toolbar1_ButtonClick(ByVal Button As MSCComctlLib.Button)
    Dim dx As Long
    Dim dy As Long
    Dim Point As POINTAPI

    With Button
        If .Style = tbrDropdown Then
            If .ButtonMenus.Count = 0 Then Exit Sub
            dx = (Me.Left + Toolbar1.Left + .Left + .Width) / Screen.Width * 65535
            dy = (Me.Top + Toolbar1.Top + .Top + .Height) / Screen.Height * 65535
            GetCursorPos Point
            mouse_event &H8003, dx, dy, 0, 0
            mouse_event &H8004, dx, dy, 0, 0
            SetCursorPos Point.X, Point.Y
        End If
    End With
End Sub

Private Sub Toolbar1_ButtonMenuClick(ByVal ButtonMenu As MSCComctlLib.ButtonMenu)
    Select Case ButtonMenu.Index
        Case 1
            MsgBox "Vous avez cliquer sur l'option 1"
        Case 2
            MsgBox "Vous avez cliquer sur l'option 2"
    End Select
End Sub

```

**Pour Win 98 : Avec un bouton normal et le déroulement d'un menu quand on clique dessus.**

vb

```

Private Sub Toolbar1_ButtonClick(ByVal Button As MSCComctlLib.Button)

    If Button.Index = 1 Then
        PopupMenu lemenu, , Button.Left, Button.Top + Button.Height + Toolbar1.Top + 12
    End If

End Sub

```



Le menu est créé dans l'éditeur de menu de la form.

## Comment donner le style de Windows XP à mes contrôles VB6 ?

**Auteurs :** CSoldier ,

**Créez un fichier Manifest comme suit: (dans le bloc note de windows par exemple)**

```

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<assembly xmlns="urn:schemas-microsoft-com:asm.v1" manifestVersion="1.0">
<assemblyIdentity
  type="win32"
  name="Microsoft.Windows.Editeur"
  processorArchitecture="X86"
  version="4.4.0.0"
/>
<description>Entrez la description du programme ici</description>
<dependency>
  <dependentAssembly>
    <assemblyIdentity
      type="win32"
      name="Microsoft.Windows.Common-Controls"
      language="*"
      processorArchitecture="X86"
      version="6.0.0.0"
      publicKeyToken="6595b64144ccf1df"
    />
  </dependentAssembly>
</dependency>
</assembly>

```

et sauvegardez-le avec le nom et l'extension du fichier exécutable pour lequel il est destiné. Exemple: *J'ai un programme qui s'appelle "CSBar.exe", si je veux faire un manifest pour ce fichier exécutable, le fichier manifest aura pour nom "CSBar.exe.manifest"* Pour finir, dans le module de code de votre Form principale, mettez le code suivant:

vb

```

'Déclaration des API
Private Declare Function InitCommonControls Lib "comctl32.dll" () As Long

Private Sub Form_Initialize()
  'Initialise les controles pour leur donner le style de Windows XP.
  InitCommonControls
End Sub

```

**Compilez votre programme et lancez-le, vous pourrez voir que vos contrôles on désormais le style de Windows XP.**

lien : [FAQ](#) Donner le style Windows XP aux contrôles dans l'IDE de Visual Basic 6

## Comment faire évoluer un ProgressBar en fonction d'un FileCopy?

**Auteurs :** Jean-Marc Rabilloud , ridan ,

**Mettez ce code dans un module :**

vb

```

Private Type SHFILEOPSTRUCT

```

```
vb
hWnd As Long
wFunc As Long
pFrom As String
pTo As String
fFlags As Integer
fAnyOperationsAborted As Boolean
hNameMappings As Long
lpzProgressTitle As String
End Type

Private Declare Function SHFileOperation Lib "shell32.dll" Alias "SHFileOperationA" (lpFileOp As SHFILEOPSTRUCT) As Long
Private Const FO_COPY = &H2
Private Const FOF_ALLOWUNDO = &H40

Public Sub CopyFileWindowsWay(SourceFile As String, DestinationFile As String)

Dim lngReturn As Long
Dim typFileOperation As SHFILEOPSTRUCT

With typFileOperation
.hWnd = 0
.wFunc = FO_COPY
.pFrom = SourceFile & vbNullChar & vbNullChar
.pTo = DestinationFile & vbNullChar & vbNullChar
.fFlags = FOF_ALLOWUNDO
End With
lngReturn = SHFileOperation(typFileOperation)
If lngReturn <> 0 Then
MsgBox Err.LastDllError, vbCritical Or vbOKOnly
Else
If typFileOperation.fAnyOperationsAborted = True Then
MsgBox "Operation Failed", vbCritical Or vbOKOnly
End If
End If

End Sub
```

Appel de la procédure :

```
vb
Call CopyFileWindowsWay(App.Path & "\denoxaut.732", App.Path & "\messenger\denoxaut.732")
```

## Utilise le contrôle Common Dialog pour récupérer le chemin d'un fichier

Auteurs : olivier],

Ouvrir un module (page de code) :

- menu *Projet* puis *Composants*
- Onglet *Contrôle*
- Cochez Microsoft Common Dialog Control

Dans la barre d'outils, choisissez le control Microsoft Common Dialog Control et placez le sur le formulaire. Nous le nommons Dlg.

Placez le code suivant dans un module (action click du bouton Btn\_Chercher) :

```

Private Sub Btn_Chercher_Click()
    With dlg
        .DialogTitle = "selectionner un fichier" 'titre de la boite
        .FileName="*.txt" 'on recherche un fichier d'extension txt
        .initDir="c:\ " 'repertoire par default
        .CancelError = false 'pour ne pas partir en erreur si on click sur annuler
        .ShowOpen
    End With
    'txtPath est la zone de texte recevant le chemin du fichier
    txtPath = dlg.FileName
End Sub

```

[lien : Afficher la boîte de dialogue ouvrir afin de récupérer le nom et le chemin du fichier sélectionné](#)  
[lien : Afficher la boîte de dialogue Enregistrer sous afin de récupérer le nom et le chemin du fichier sélectionné](#)  
[lien : Comment sélectionner plusieurs extensions dans le filtre d'un CommonDialog ?](#)

### Comment sélectionner plusieurs extensions dans le filtre d'un CommonDialog ?

**Auteurs : Tofalu ,**

Pour utiliser le CommonDialog, il faut cocher le composant Microsoft Common Dialog Control 6.0 (SP2) et utiliser ce code :

```

vb
CommonDialog1.Filter = "Fichier zip et rar (*.zip,*.rar)|*.zip;*.rar"

```

### Comment copier une image dans un RichTextBox ?

**Auteurs : ridan ,**

Placer un composant RichTextBox sur votre feuille (Microsoft Rich Textbox Control 6.0 (SP4)) et le code suivant :

```

vb
Private Declare Function SendMessage Lib "user32.dll" Alias "SendMessageA" ( _
    ByVal hwnd As Long, _
    ByVal wMsg As Long, _
    ByVal wParam As Long, _
    ByVal lParam As Any) As Long

Private Const WM_PASTE As Long = &H302

Private Sub Form_Load()
    Clipboard.Clear
    Clipboard.SetData LoadPicture("c:\bitmap.bmp")
    SendMessage RichTextBox1.hwnd, WM_PASTE, 0, 0
End Sub

```

### Comment faire de l'auto complétion avec une Combobox standard ?

**Auteurs : ThierryAIM ,**

Une Combobox standard en VB6 n'a pas de propriété "MatchEntry"  
 Il est toutefois possible de faire de l'auto-complétion dans une Combobox avec le code suivant :

vb

```
Private Sub Combo1_Change()  
    Dim i As Integer, start As Integer  
    start = Len(Combo1.Text)  
    For i = 0 To Combo1.ListCount - 1  
        If Left(Combo1.List(i), start) = Combo1.Text Then  
            Combo1.Text = Combo1.List(i)  
        End If  
    Next  
    Combo1.SelStart = start  
    Combo1.SelLength = Len(Combo1.Text)  
End Sub
```

Une autre méthode en utilisant l'API Windows :

vb

```
Private Declare Function SendMessage Lib "user32" Alias "SendMessageA" _  
    (ByVal hwnd As Long, ByVal wParam As Long, _  
    ByVal lParam As Long, lParam As Any) As Long  
  
Private Const CB_ERR As Long = (-1)  
Private Const CB_SELECTSTRING As Long = &H14D  
  
Private Sub Combo1_Change()  
    Dim start As Integer  
    start = Len(Combo1.Text)  
    If SendMessage(Combo1.hwnd, CB_SELECTSTRING, ByVal Combo1.ListIndex, ByVal Combo1.Text) <>  
        CB_ERR Then  
            Combo1.SelStart = start  
            Combo1.SelLength = Len(Combo1.Text)  
        End If  
    End Sub
```

## Comment faire un écran d'accueil (splash screen) ?

Auteurs : **Romain Puyfoulhoux** , **Alexandre Lokchine** ,

Si au lancement de votre application, une fenêtre ne s'affiche pas instantanément, cela donnera une impression de lourdeur et de lenteur d'exécution à l'utilisateur. Le chargement de la fenêtre principale d'un programme pouvant être relativement long, il est courant d'afficher un écran d'accueil, qui apparaît immédiatement, et qui reste visible pendant le chargement du reste du programme.

Nous allons créer une procédure Main qui sera chargée :

- d'afficher l'écran d'accueil
- d'afficher la fenêtre principale
- de fermer l'écran d'accueil après un délai à spécifier

Le délai sert à laisser l'écran d'accueil visible au moins quelques secondes, même dans le cas où l'ouverture de la fenêtre principale serait quasi instantanée.

Pour créer l'écran d'accueil, ajoutez une form au projet et nommez-la frmSplash. Enlevez sa barre de titre en donnant la valeur False à la propriété ControlBox et en ne mettant aucun texte dans la propriété Caption. Vous pouvez aussi, si vous le souhaitez, enlever la bordure de la fenêtre en donnant la valeur 0 à la propriété BorderStyle. Donnez la valeur 2 à la propriété StartUpPosition. Posez ensuite un timer et nommez-le "tmr". Puis collez ce code source dans le module de la feuille :

```
vb
Dim sec As Long
Dim delai As Long

Private Sub Form_Load()

    tmr.Enabled = False
    tmr.Interval = 1000
    sec = 0
    AuSommet Me.hwnd

End Sub

Public Sub CloseAfter(ByVal attente As Long)

    tmr.Enabled = True
    delai = attente

End Sub

Private Sub tmr_Timer()

    sec = sec + 1
    Me.Print sec
    If sec >= delai Then Unload Me

End Sub]
```

Dans la procédure Form\_Load, l'appel à la fonction AuSommet permet de spécifier que cette form doit toujours se trouver au premier plan. Ainsi elle ne sera pas cachée par la fenêtre principale. Ajoutez une autre form au projet et nommez-la frmMain. Placez enfin ce code dans un module standard :

vb

vb

```

Private Const SWP_NOMOVE = 2
Private Const SWP_NOSIZE = 1
Private Const FLAGS = SWP_NOMOVE Or SWP_NOSIZE
Private Const HWND_TOPMOST = -1

Private Declare Function SetWindowPos Lib "USER32" (ByVal hwnd As Long, ByVal hWndInsertAfter As
Long, _
                                                    ByVal x As Long, ByVal y As Long, ByVal cx As
Long, _
                                                    ByVal cy As Long, ByVal wFlags As Long) As Long

Public Function AuSommet(hwnd As Long) As Long

    AuSommet = SetWindowPos(hwnd, HWND_TOPMOST, 0, 0, 0, 0, FLAGS)

End Function

Sub Main()

    frmSplash.Show
    DoEvents
    frmMain.Show
    'fermeture au bout de 2 secondes
    frmSplash.CloseAfter 2

End Sub

```

**lien : [Projet VB à télécharger](#)**

### Peut-on changer la police ou la couleur du texte avec MsgBox ?

**Auteurs : Romain Puyfoulhoux ,**

**Non. Vous devrez donc programmer vos propres boîtes de message avec une form qui aura la valeur fixedSingle à la propriété BorderStyle pour qu'elle ne soit pas redimensionnable, et la valeur CenterScreen pour StartUpPosition.**

### Comment faire une barre de progression ?

**Auteurs : Romain Puyfoulhoux ,**

**Voici deux solutions : la faire vous même avec un PictureBox, ou utiliser le contrôle ProgressBar contenu dans le composant Microsoft Windows Common Controls 6.0. Voyons un exemple illustrant la première méthode; posez un PictureBox et un timer sur une form, et placez ce code dans le module de la form :**

vb

```

Dim niveau as long, max as long

Private Sub Form_Load()

    max = 100    'valeur maximale du niveau
    niveau = 0  'valeur initiale
    Timer1.Interval = 200
    Picture1.Width = 5000
    Picture1.Height = 280

End Sub

Private Sub Timer1_Timer()

```

```
vb
niveau = niveau + 1
If niveau = max Then Timer1.Enabled = False
Picture1.Line (0, 0)-(niveau * Picture1.ScaleWidth / max, Picture1.ScaleHeight), vbBlue, BF

End Sub
```

Et maintenant voici l'équivalent avec le contrôle **ProgressBar**. Posez sur une form un timer et un contrôle **ProgressBar**, et placez ce code dans le module de la form :

```
vb

Private Sub Form_Load()

ProgressBar1.min = 0
ProgressBar1.max = 100
ProgressBar1.Value = 0
Timer1.Interval = 200

End Sub

Private Sub Timer1_Timer()

ProgressBar1.Value = ProgressBar1.Value + 1
If ProgressBar1.Value >= ProgressBar1.max Then Timer1.Enabled = False

End Sub
```

## Comment modifier le texte d'un rptLabel dans un datareport ?

**Auteurs : Romain Puyfoulhoux ,**

**Exemple pour un label nommé "lblDateImp" qui est dans la section "sctEntete" :**

```
vb

DataReport1.Sections("sctEntete").controls("lblDateImp").caption = "Imprimé le " & date
```

## Comment afficher un DataReport en mode paysage ?

**Auteurs : Romain Puyfoulhoux ,**

Si vous n'avez pas installé le Service Pack 4 ou une version ultérieure pour Microsoft Visual Studio 6, le DataReport utilise la configuration de l'imprimante par défaut pour choisir entre le mode portrait et le mode paysage. Les Service Packs 4 et supérieurs contiennent une mise à jour de l'objet DataReport, qui possède une nouvelle propriété appelée **Orientation**. Le code suivant utilise cette propriété afin d'afficher un DataReport en mode paysage :

```
vb

DataReport1.Orientation = rptOrientLandscape
```

```
vb
```

```
DataReport1.Show
```

## Comment faire défiler un ensemble de contrôles avec un ascenseur ?

Auteurs : [Romain Puyfoulhoux](#) ,

Placez sur une form une frame nommée Frame1. Sélectionnez-la, et placez dans cette frame une autre frame, nommée Frame2.

Frame1 doit être le conteneur de Frame2 et la hauteur de Frame2 doit être supérieure à celle de Frame1. Mettez la propriété Top de Frame2 à 0. Puis placez les contrôles qui devront défiler dans Frame2. Enfin ajoutez ce code dans le module de la form :

```
vb
```

```
Private Sub Form_Load()  
  
VScroll1.Min = 0  
VScroll1.Max = Frame2.Height - Frame1.Height  
  
End Sub  
  
Private Sub VScroll1_Change()  
  
Frame2.Top = -VScroll1.Value  
  
End Sub  
  
Private Sub VScroll1_Scroll()  
  
VScroll1_Change  
  
End Sub
```

Vous pouvez aussi utiliser des PictureBox, mais les frames consomment moins de ressources.

## Comment mettre des images dans un menu ?

Auteurs : [Romain Puyfoulhoux](#) ,

L'éditeur de menus de Visual Basic ne permet pas d'ajouter une image pour un menu. Voici une façon de le faire.

Copiez ces déclarations dans un module standard :

```
vb
```

```
Public Declare Function GetMenu Lib "user32" (ByVal hwnd As Long) As Long  
Public Declare Function GetSubMenu Lib "user32" (ByVal hMenu As Long, ByVal nPos As Long) As Long  
Public Declare Function SetMenuItemBitmaps Lib "user32" (ByVal hMenu As Long, _  
ByVal nPosition As Long, _  
ByVal wFlags As Long, _  
ByVal hBitmapUnchecked As Long, _  
ByVal hBitmapChecked As Long) As Long  
Public Declare Function GetMenuItemID Lib "user32" (ByVal hMenu As Long, ByVal nPos As Long) As  
Long  
Public Const MF_BYPOSITION = &H400&
```



Vos images seront de préférence des gifs transparentes. Vous pouvez les stocker par exemple dans un contrôle ImageList. Pour cela, ajoutez le composant « Microsoft Windows Common Controls 6.0 » à votre projet. Déposez un contrôle ImageList sur votre form. Faites un clic droit sur le contrôle ImageList et cliquez sur le menu Propriétés. La fenêtre qui s'est ouverte vous permet d'insérer vos images.

Le code qui ajoute les images aux menus est à placer dans la procédure Form\_Load() de la form. L'exemple ci-dessous traite les deux premiers menus d'une form, qui sont typiquement les menus Fichier et Edition.

```
vb
Private Sub Form_Load()

Dim hMenu As Long, hSousMenu As Long, menuId As Long

hMenu = GetMenu(Me.hwnd)

hSousMenu = GetSubMenu(hMenu, 0) 'handle du menu Fichier
SetMenuItemBitmaps hSousMenu, 0, MF_BYPOSITION, ImageList1.ListImages(1).Picture, 0 'ouvrir
SetMenuItemBitmaps hSousMenu, 1, MF_BYPOSITION, ImageList1.ListImages(2).Picture, 0 'enregistrer
SetMenuItemBitmaps hSousMenu, 2, MF_BYPOSITION, ImageList1.ListImages(3).Picture, 0 'imprimer

hSousMenu = GetSubMenu(hMenu, 1) 'handle du menu Edition
SetMenuItemBitmaps hSousMenu, 0, MF_BYPOSITION, ImageList1.ListImages(4).Picture, 0 'annuler
SetMenuItemBitmaps hSousMenu, 1, MF_BYPOSITION, ImageList1.ListImages(5).Picture, 0 'couper
SetMenuItemBitmaps hSousMenu, 2, MF_BYPOSITION, ImageList1.ListImages(6).Picture, 0 'copier
SetMenuItemBitmaps hSousMenu, 3, MF_BYPOSITION, ImageList1.ListImages(7).Picture, 0 'coller

End Sub
```

Cette méthode a malheureusement un inconvénient : les couleurs sont mal respectées, et vos images auront un aspect anormalement foncé. Une autre méthode, beaucoup plus compliquée et réservée aux programmeurs avancés, est utilisée dans ce projet.

## Comment mettre les éléments de mon menu sur plusieurs colonnes ?

Auteurs : Jean-Marc Rabilloud ,

Il ne s'agit pas ici de sous-menus à un deuxième niveau, comme ceux du menu "Zoom" de l'éditeur de Visual Basic, mais bien de mettre les sous-menus sur plusieurs colonnes.

Dans le module d'une form, copiez d'abord ces déclarations dans la partie Générale :

```
vb
Private Declare Function GetMenu Lib "user32" (ByVal hwnd As Long) As Long
Private Declare Function GetSubMenu Lib "user32" (ByVal hMenu As Long, ByVal nPos As Long) As Long
Private Declare Function GetMenuItemCount Lib "user32" (ByVal hMenu As Long) As Long
Private Declare Function GetMenuString Lib "user32" Alias "GetMenuStringA" _
    (ByVal hMenu As Long, ByVal wIDItem As Long, _
    ByVal lpString As String, ByVal nMaxCount As Long, _
    ByVal wFlag As Long) As Long
Private Declare Function GetMenuItemID Lib "user32" (ByVal hMenu As Long, ByVal nPos As Long) As Long
Private Declare Function ModifyMenu Lib "user32" Alias "ModifyMenuA" _
    (ByVal hMenu As Long, ByVal nPosition As Long, _
    ByVal wFlags As Long, ByVal wIDNewItem As Long, _
    ByVal lpString As Any) As Long
```

Puis copiez cette procédure :

vb

```

Private Sub MenuSurPlusieursColonnes(handle As Long, numeroDuMenu As Integer, _
                                     nbEltsParColonnes As Integer)

Dim hMenu As Long, hSubMenu As Long
Dim mnuItemCount As Long, mnuItemID As Long, mnuItemText As String
Dim compteur As Integer, Resultat As Long, Buffer As String

If nbEltsParColonnes < 1 Then Exit Sub
If nbEltsParColonnes < 0 Then Exit Sub

hMenu = GetMenu(handle) 'Handle du menu de la feuille
hSubMenu = GetSubMenu(hMenu, numeroDuMenu) 'Handle du sous menu désiré
mnuItemCount = GetMenuItemCount(hSubMenu) 'Nombre d'éléments dans le sous menu

'On règle le pas et le compteur pour le nombre d'éléments par colonne
For compteur = nbEltsParColonnes + 1 To mnuItemCount Step nbEltsParColonnes
    Buffer = Space$(256)
    Resultat = GetMenuString(hSubMenu, compteur - 1, Buffer, Len(Buffer), &H400&)
    mnuItemText = Left$(Buffer, Resultat)
    mnuItemID = GetMenuItemID(hSubMenu, compteur - 1)
    Call ModifyMenu(hSubMenu, compteur - 1, &H400& Or &H20&, mnuItemID, mnuItemText)
Next compteur

End Sub
    
```

Appelez simplement la procédure pour chacun des menus à modifier :

vb

```

Private Sub Form_Load()

MenuSurPlusieursColonnes Me.hwnd, 0, 5 '5 éléments par colonne pour le premier menu
MenuSurPlusieursColonnes Me.hwnd, 1, 8 '8 éléments par colonne pour le deuxième menu

End Sub
    
```

## Comment faire du Drag and Drop ?

**Auteurs : Jean-Marc Rabilloud ,**

Le DragDrop doit être vu comme un artifice visuel. Pour schématiser, rien n'est déplacé lors d'une opération DragDrop. C'est votre code qui va générer le "déplacement". Cette opération existe sous deux formes :

- La forme standard pour les opérations de déplacement au sein d'un process
- La forme OLE, qui rend possible les déplacements inter process.

La forme standard est assez simple d'utilisation. On active l'opération dans l'événement MouseMove du contrôle source, on gère les icônes dans l'(es) événement(s) DragOver des contrôles de la feuille, et enfin on effectue le déplacement dans l'événement DragDrop du contrôle cible. L'exemple suivant permet d'échanger des éléments entre deux contrôles Listbox.

vb

```

Private Sub Form_DragOver(Source As Control, X As Single, Y As Single, State As Integer)
If Source Is List1 Then
    Source.DragIcon = _
        LoadPicture("C:\Program Files\Microsoft Visual Studio\Common\Graphics\Icons\Misc
        \misc06.ico")
End If
    
```

```
vb
End Sub

Private Sub List1_DragOver(Source As Control, X As Single, Y As Single, State As Integer)
If (Source Is List1) Or (Source Is List2) Then
    Source.DragIcon = _
        LoadPicture("C:\Program Files\Microsoft Visual Studio\Common\Graphics\Icons\Dragdrop
\droplpg.ico")
End If
End Sub

Private Sub List2_DragOver(Source As Control, X As Single, Y As Single, State As Integer)
If (Source Is List1) Or (Source Is List2) Then
    Source.DragIcon = _
        LoadPicture("C:\Program Files\Microsoft Visual Studio\Common\Graphics\Icons\Dragdrop
\droplpg.ico")
End If
End Sub

Private Sub List1_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
List1.Drag vbBeginDrag
End Sub

Private Sub List2_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
List2.Drag vbBeginDrag
End Sub

Private Sub List1_DragDrop(Source As Control, X As Single, Y As Single)
If Source Is List2 Then
    List1.AddItem List2.List(List2.ListIndex)
    List2.RemoveItem List2.ListIndex
End If
End Sub

Private Sub List2_DragDrop(Source As Control, X As Single, Y As Single)
If Source Is List1 Then
    List2.AddItem List1.List(List1.ListIndex)
    List1.RemoveItem List1.ListIndex
End If
End Sub
```

Sous sa forme OLE, le drag and drop peut être beaucoup plus complexe. Bien qu'identique dans la forme de programmation, les données peuvent être transmises entre les applications. Dans l'exemple suivant, vous pouvez déplacer un fichier graphique en partant de l'explorateur vers votre PictureBox.

```
vb

Private Sub Form_Load()

'autorise les opérations manuelles de dépose
Picture1.OLEDropMode = vbOLEDropManual

End Sub

Private Sub Picture1_OLEDragDrop(Data As DataObject, Effect As Long, Button As Integer, _
    Shift As Integer, X As Single, Y As Single)

Dim NomFichier As String

'vérifie le format de l'objet Data Transmis
If Data.GetFormat(vbCFFiles) = True Then
    ' vbCFFiles correspond à une liste de fichier, donc récupération du premier élément
    NomFichier = Data.Files(1)
    On Error GoTo invalidPicture
    Picture1.Picture = LoadPicture(NomFichier)
End If
```

```
vb
Exit Sub

invalidPicture:
    MsgBox "Format de fichier incorrect", vbCritical + vbOKOnly

End Sub

Private Sub Picture1_OLEDragOver(Data As DataObject, Effect As Long, Button As Integer, Shift As Integer, _
    X As Single, Y As Single, State As Integer)

'vérifie le format de l'objet Data Transmis pour valoriser le paramètre Effect
If Data.GetFormat(vbCFFiles) Then
    Effect = vbDropEffectCopy And Effect
Else
    Effect = vbDropEffectNone
End If

End Sub
```

## Comment récupérer les coordonnées de la souris ?

**Auteurs : Romain Puyfoulhoux ,**

Les procédures des événements **MouseDown**, **MouseMove** et **MouseUp** reçoivent les coordonnées de la souris dans les paramètres **X** et **Y**. L'origine utilisée est le coin haut gauche du contrôle concerné. Les coordonnées sont exprimées dans l'unité spécifiée dans la propriété **ScaleMode** pour les contrôles conteneurs, et toujours en twips pour les autres.

Pour les autres événements, par exemple **Click** ou **DoubleClick**, les coordonnées se récupèrent via les API Windows. Copiez ces déclarations au début du module de la form :

```
vb

Private Declare Function GetCursorPos Lib "user32" (lpPoint As POINTAPI) As Long
Private Declare Function ScreenToClient Lib "user32" (ByVal hwnd As Long, lpPoint As POINTAPI) As Long

Private Type POINTAPI
    X As Long
    Y As Long
End Type
```

Puis la fonction ci-dessous, qui renvoie les coordonnées de la souris exprimées en twips, par rapport au coin haut gauche de la form :

```
vb

Private Function GetMousePosition() As POINTAPI

Dim pos As POINTAPI

GetCursorPos pos
ScreenToClient Me.hwnd, pos
pos.X = Screen.TwipsPerPixelX * pos.X
pos.Y = Screen.TwipsPerPixelY * pos.Y
GetMousePosition = pos
```

```
vb  
End Function
```

## Comment simuler un clic de souris ?

Auteurs : Alexandre Lokchine ,

Placez ce code dans un module standard. La procédure Clic simule un clic de souris. Les paramètres à passer sont les coordonnées en pixels.

```
vb  
  
Declare Function ClientToScreen Lib "user32" (ByVal hwnd As Long, lpPoint As POINTAPI) As Long  
Declare Sub mouse_event Lib "user32" (ByVal dwFlags As Long, ByVal dx As Long, ByVal dy As Long, _  
    ByVal cButtons As Long, ByVal dwExtraInfo As Long)  
  
Public Const MOUSEEVENTF_MOVE = &H1  
Public Const MOUSEEVENTF_LEFTDOWN = &H2  
Public Const MOUSEEVENTF_LEFTUP = &H4  
Public Const MOUSEEVENTF_RIGHTDOWN = &H8  
Public Const MOUSEEVENTF_RIGHTUP = &H10  
Public Const MOUSEEVENTF_MIDDLEDOWN = &H20  
Public Const MOUSEEVENTF_MIDDLEUP = &H40  
Public Const MOUSEEVENTF_ABSOLUTE = &H8000  
  
Public Declare Function GetCursorPos Lib "user32" (lpPoint As POINTAPI) As Long  
Public Declare Function SetCursorPos Lib "user32" (ByVal x As Long, ByVal y As Long) As Long  
  
Public Type POINTAPI  
    x As Long  
    y As Long  
End Type  
  
Public dstX As Long  
Public dstY As Long  
  
Public Function GetX() As Long  
    Dim n As POINTAPI  
    GetCursorPos n  
    GetX = n.x  
End Function  
  
Public Function GetY() As Long  
    Dim n As POINTAPI  
    GetCursorPos n  
    GetY = n.y  
End Function  
  
Public Sub Clic(PosX As Integer, PosY As Integer)  
  
    'placer la souris par sécurité (pour processeurs lents)  
    Call SetCursorPos(PosX, PosY)  
    mouse_event MOUSEEVENTF_LEFTDOWN + MOUSEEVENTF_LEFTUP, PosX, PosY, 0, 0  
  
End Sub
```

## Comment insérer une image au format gif animée ?

Auteurs : Romain Puyfoulhoux ,

Voici un composant gratuit très simple d'emploi : gif89.zip.

Il suffit d'extraire le fichier d'extension .dll dans le répertoire windows\system, puis de l'ajouter dans les composants du projet. Vous aurez alors un nouveau contrôle dans la boîte à outils. Pour afficher une image, vous pouvez indiquer le chemin du fichier à la propriété filename en mode conception ou lors de l'exécution.

## Qu'est-ce que le sous classement ?

Auteurs : Romain Puyfoulhoux ,

Sous Windows, tous les contrôles et toutes les fenêtres sont attachés à une fonction, que l'on appelle "procédure de fenêtre", et qui est chargée de traiter tous les messages reçus par cette fenêtre. Voici sa déclaration en VB :

vb

```
Public Function WindowProc(ByVal hwnd As Long, ByVal msg As Long, _  
    ByVal wParam As Long, ByVal lParam As Long) As Long
```

Cette fonction est donc exécutée chaque fois qu'une fenêtre qui lui est attachée reçoit un message. VB utilise ce genre de fonction pour nous signaler qu'un événement a lieu en lançant la procédure événementielle que l'on a déclarée, par exemple Form\_Load.

Mais de nombreux événements pouvant parfois être utiles au programmeur n'ont pas été repris dans Visual Basic. C'est ici qu'intervient le sous classement - en anglais subclassing - : nous attachons à un contrôle ou à la form notre propre procédure afin d'intercepter l'évènement qui nous intéresse.

Exemples d'utilisation du sous classement :

- Comment détecter le mouvement d'une form ?
- Comment détecter le changement de la résolution de l'écran ?

## Comment compiler un ou plusieurs projets VB6 à partir d'une autre application VB6 ?

Auteurs : Xo , ThierryAIM ,

Un projet VB6 (.vbp, .vbg) est compilable en ligne de commande.

Le principe : créer un tableau contenant le ou les chemins complets du ou des projets à compiler. Lancer la procédure de compilation, qui va, elle-même, exécuter ces tâches successivement (Sub ExecuteTache), afin d'éviter les erreurs de compilation multiples et simultanées.

Nota : tous les projets à compiler doivent être fermés, et les fichiers exécutables associés, non utilisés (faute de quoi ils ne pourront être écrasés).

vb

```
Private Declare Function CloseHandle Lib "kernel32" (ByVal hObject As Long) As Long  
Private Declare Function GetExitCodeProcess Lib "kernel32" (ByVal hProcess As Long, lpExitCode As  
    Long) As Long  
Private Declare Function OpenProcess Lib "kernel32" (ByVal dwDesiredAccess As Long, _  
    ByVal bInheritHandle As Long, ByVal  
    dwProcessId As Long) As Long  
Private Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As Long)  
  
Private Const STILL_ACTIVE = &H103  
Private Const PROCESS_QUERY_INFORMATION = &H400  
Private Const C_COMMANDLINECOMPILE As String = ""C:\Program Files\Microsoft Visual Studio  
    \VB98\vb6.exe" /make "
```

vb

```

Private Sub ExecuteTache(ByVal Tache As String, Optional ModShell As Variant)
' ---Cette fonction crée une tache puis regarde si elle est active---
Dim hProcess As Long
Dim RetVal As Long
' ---Ouvre un programme et récupère son handle (hProcess)---
On Error GoTo errShell
hProcess = OpenProcess(PROCESS_QUERY_INFORMATION, False, _
Shell(Tache, IIf(IsMissing(ModShell), vbMinimizedNoFocus, ModShell)))
On Error GoTo 0

' ---Regarde si le processus est actif---
Do
' Retourne le status du processus en cours
GetExitCodeProcess hProcess, RetVal
' Les 2 lignes suivantes sont recommandées pour éviter de faire
' travailler le système avec GetExitCodeProcess
DoEvents
Sleep 200
Loop While RetVal = STILL_ACTIVE
CloseHandle (hProcess)
Debug.Print "Tache exécutée"
Exit Sub

errShell:
msg = "Erreur " & Err.Number & " : " & Err.Description & vbCrLf & _
"Programme concerné : " & Tache
MsgBox msg, vbOKOnly + vbCritical, "Lancement de la tâche impossible"
End Sub

Private Sub Compile(Projects() As String)
Dim i As Integer
Screen.MousePointer = vbHourglass
For i = 0 To UBound(Projects)
cmd = C_COMMANDLINECOMPILE & Projects(i)
Debug.Print cmd
ExecuteTache cmd
Next i
Screen.MousePointer = vbDefault
End Sub

Private Sub Command1_Click()
Dim ProjetACompiler(1) As String

ProjetACompiler(0) = ""C:\VB6toto\Projet1.vbp""
ProjetACompiler(1) = ""C:\VB6titi\Projet1.vbp""
Compile ProjetACompiler()
End Sub

```

## Comment détecter l'action sur une touche ?

Auteurs : spacefrog ,

Pour capturer certaines touches comme la touche "Suppr" l'utilisation de l'événement keypress, ne suffit pas, une solution est l'utilisation de l'API GetAsyncKeyState  
déclarer l'API GetAsyncKeyState :

vb

```
Public Declare Function GetAsyncKeyState Lib "user32.dll" (ByVal vKey As Long) As Integer
```

Voici en exemple, une boucle qui s'interrompt avec une action sur la touche Suppr :

vb

```
'Boucle tant que touche Suppr pas actionnée..  
While GetAsyncKeyState(&H2E) = 0  
    DoEvents  
Wend  
MsgBox "FIN"
```

Liste des constantes utilisables :

**Constante Valeur Définition**

- **VK\_F9 78** Touche f9
- **VK\_F8 77** Touche f8
- **VK\_F6 75** Touche f6
- **VK\_F7 76** Touche f7
- **VK\_F5 74** Touche f5
- **VK\_F4 73** Touche f4
- **VK\_F3 72** Touche f3
- **VK\_F2 71** Touche f2
- **VK\_F1 70** Touche f1
- **VK\_DIVIDE 6F** Touche "division".
- **VK\_DECIMAL 6E** Touche séparateur décimal.
- **VK\_SUBTRACT 6D** Touche "soustraction".
- **VK\_SEPARATOR 6C** Touche de séparation.
- **VK\_ADD 6B** Touche "addition".
- **VK\_MULTIPLY 6A** Touche "multiplication".
- **VK\_NUMPAD9 69** Touche 9 (clavier numérique).
- **VK\_NUMPAD8 68** Touche 8 (clavier numérique).
- **VK\_NUMPAD7 67** Touche 7 (clavier numérique).
- **VK\_NUMPAD6 66** Touche 6 (clavier numérique).
- **VK\_NUMPAD5 65** Touche 5 (clavier numérique).
- **VK\_NUMPAD4 64** Touche 4 (clavier numérique).
- **VK\_NUMPAD3 63** Touche 3 (clavier numérique).
- **VK\_NUMPAD2 62** Touche 2 (clavier numérique).
- **VK\_NUMPAD1 61** Touche 1 (clavier numérique).
- **VK\_NUMPAD0 60** Touche 0 (clavier numérique).
- **VK\_APPS 5D** Touche Windows applications (Microsoft Natural Keyboard).
- **VK\_RWIN 5C** Touche Windows droite (Microsoft Natural Keyboard).
- **VK\_LWIN 5B** Touche Windows gauche (Microsoft Natural Keyboard).
- **VK\_Z 5A** Touche z
- **VK\_Y 59** Touche y
- **VK\_X 58** Touche x
- **VK\_W 57** Touche w
- **VK\_V 56** Touche v
- **VK\_U 55** Touche u
- **VK\_T 54** Touche t
- **VK\_S 53** Touche s
- **VK\_R 52** Touche r
- **VK\_Q 51** Touche q
- **VK\_P 50** Touche p
- **VK\_O 4F** Touche o
- **VK\_N 4E** Touche n
- **VK\_M 4D** Touche m
- **VK\_L 4C** Touche l
- **VK\_K 4B** Touche k
- **VK\_J 4A** Touche j



- **VK\_I 49 Touche i**
- **VK\_H 48 Touche h**
- **VK\_G 47 Touche g**
- **VK\_F 46 Touche f**
- **VK\_E 45 Touche e**
- **VK\_D 44 Touche d**
- **VK\_C 43 Touche c**
- **VK\_B 42 Touche b**
- **VK\_A 41 Touche a**
- **VK\_9 39 Touche 9**
- **VK\_8 38 Touche 8**
- **VK\_7 37 Touche 7**
- **VK\_5 35 Touche 5**
- **VK\_6 36 Touche 6**
- **VK\_3 33 Touche 3**
- **VK\_4 34 Touche 4**
- **VK\_1 31 Touche 1**
- **VK\_2 32 Touche 2**
- **VK\_0 30 Touche 0**
- **VK\_HELP 2F Touche "aide".**
- **VK\_DELETE 2E Touche "Suppression".**
- **VK\_INSERT 2D Touche insertion.**
- **VK\_SNAPSHOT 2C Touche impression écran.**
- **VK\_EXECUTE 2B Touche "exécution".**
- **VK\_SELECT 29 Touche "selection".**
- **VK\_DOWN 28 Flèche curseur bas.**
- **VK\_RIGHT 27 Flèche curseur droit.**
- **VK\_UP 26 Flèche curseur haut.**
- **VK\_LEFT 25 Flèche curseur gauche.**
- **VK\_HOME 24 Touche "début".**
- **VK\_END 23 Touche "fin".**
- **VK\_NEXT 22 Touche "page bas".**
- **VK\_PRIOR 21 Touche "page haut".**
- **VK\_SPACE 20 Touche Espace.**
- **VK\_ESCAPE 1B Touche Echap.**
- **VK\_CAPITAL 14 Touche verrouillage majuscule.**
- **VK\_PAUSE 13 Touche "Pause".**
- **VK\_MENU 12 Touche "Alt".**
- **VK\_CONTROL 11 Touche "Control".**
- **VK\_SHIFT 10 Touche "Shift".**
- **VK\_RETURN 0D Touche "Entrée".**
- **VK\_CLEAR 0C Touche d'effacement.**
- **VK\_TAB 09 Touche tabulation.**
- **VK\_BACK 08 Touche retour arrière.**
- **VK\_MBUTTON 04 Bouton du milieu de la souris (le 3ème bouton).**
- **VK\_CANCEL 03 Control-break.**
- **VK\_RBUTTON 02 Bouton droit de la souris.**
- **VK\_LBUTTON 01 Bouton gauche de la souris.**
- **VK\_F10 79 Touche f10**
- **VK\_F11 7A Touche f11**
- **VK\_F12 7B Touche f12**
- **VK\_F13 7C Touche f13**
- **VK\_F14 7D Touche f14**
- **VK\_F15 7E Touche f15**

- **VK\_F16 7F Touche f16**
- **VK\_F17 80H Touche f17**
- **VK\_F18 81H Touche f18**
- **VK\_F19 82H Touche f19**
- **VK\_F20 83H Touche f20**
- **VK\_F21 84H Touche f21**
- **VK\_F22 85H Touche f22**
- **VK\_F23 86H Touche f23**
- **VK\_F24 87H Touche f24**
- **VK\_NUMLOCK 90 Touche verrouillage numérique.**
- **VK\_SCROLL 91 Touche verrouillage défilement.**
- **VK\_ATTN F6 Touche "Attn".**
- **VK\_CRSEL F7 Touche "CrSel".**
- **VK\_EXSEL F8 Touche "ExSel".**
- **VK\_PLAY FA Touche "Play".**
- **VK\_ZOOM FB Touche "Zoom".**
- **VK\_NONAME FC Reservé.**
- **VK\_PA1 FD Touche PA1.**

## Sommaire > Graphisme

### Comment retrouver les composantes rouge, verte, bleue d'un code couleur de type Long ?

Auteurs : Jean-Marc Rabilloud , jmfmarques ,

Rappelons que le code d'une couleur se calcule par les composantes RGB à l'aide de la formule :

$\text{Red} + \text{Green} * 256 + \text{Blue} * 256 * 256.$

Il nous suffit donc d'écrire la fonction inverse.

vb

```
Public Sub ComposantesRGB(ByVal Couleur As Long, Red As Long, Green As Long, Blue As Long)

Blue = Int(Couleur / 65536)
Green = Int((Couleur - (65536 * Blue)) / 256)
Red = Couleur - ((Blue * 65536) + (Green * 256))

End Sub

Private Sub Command1_Click()
Dim Rouge As Long, Vert As Long, Bleu As Long

ComposantesRGB 9550940, Rouge, Vert, Bleu
LblRed.Caption = "Red = " & Rouge
LblGreen.Caption = "Green = " & Vert
LblBlue.Caption = "Blue = " & Bleu

End Sub
```

### Une autre méthode en utilisant l'API Windows

vb

```
Private Declare Function TranslateColor Lib "olepro32.dll" Alias "OleTranslateColor" _
(ByVal clr As OLE_COLOR, ByVal palet As Long, col As Long) As Long

Private Sub TranslateRGB(coulnorm, R, G, B)
Dim RealColor As Long
TranslateColor coulnorm, 0, RealColor
R = RealColor And &HFF&
G = (RealColor And &HFF00&) / 2 ^ 8
B = (RealColor And &HFF0000) / 2 ^ 16
End Sub

Private Sub Command1_Click()
TranslateRGB Me.Label1.BackColor, R, G, B
MsgBox "R = " & R & " G = " & G & " B = " & B
End Sub
```

### Comment retrouver la couleur d'un pixel à l'écran ?

Auteurs : Alexandre Lokchine , Romain Puyfoulhoux ,

Si vous cherchez simplement à connaître la couleur d'un point dans un PictureBox, utilisez sa méthode Point(). Sinon, voici une méthode pour récupérer la couleur de n'importe quel pixel à l'écran.

Placez ce code dans un module standard :

vb

```
Private Declare Function GetDC Lib "user32" (ByVal hwnd As Long) As Long
Private Declare Function GetPixel Lib "gdi32" (ByVal hdc As Long, ByVal x As Long, ByVal y As Long) As Long

Public Type Couleur
    red As Integer
    green As Integer
    blue As Integer
End Type

Public Function CouleurPixel (ByVal x As Long, ByVal y As Long) As Couleur

    Dim pixel As Couleur, RGBPx As Long

    RGBPx = GetPixel(GetDC(0&), x, y)
    pixel.red = &HFF& And RGBPx
    pixel.green = (&HFF00& And RGBPx) \ 256
    pixel.blue = (&HFF0000 And RGBPx) \ 65536
    CouleurPixel = pixel

End Function
```

La fonction `CouleurPixel` renvoie la couleur du pixel, dont les coordonnées lui ont été passées en paramètres, dans une structure `Couleur`. Si vous voulez connaître la position du pointeur de souris, ajoutez ces déclarations dans le module standard :

vb

```
Public Declare Function GetCursorPos Lib "user32" (lpPoint As PointAPI) As Long

Public Type PointAPI
    X As Long
    Y As Long
End Type
```

Voici maintenant un exemple où l'on utilise un timer pour mettre dans un `PictureBox` la couleur du pixel survolé par la souris :

vb

```
Private Sub Timer1_Timer()

    Dim pixel As Couleur, CursPos As PointAPI

    GetCursorPos CursPos
    pixel = CouleurPixel(CursPos.X, CursPos.Y)
    Picture1.BackColor = RGB(pixel.red, pixel.green, pixel.blue)

End Sub
```

## Comment avoir la couleur inverse exacte ?

**Auteurs : Cafeine ,**

Lors d'un changement dynamique de la couleur de fond d'un contrôle, il peut s'avérer utile pour une meilleure lisibilité de modifier la police de caractères, pour cela utiliser la fonction suivante qui vous permet d'avoir en retour la couleur inverse de celle passée en paramètre

```

Option Explicit

Type Col_Sep
    red As Integer
    green As Integer
    blue As Integer
End Type

Function GetInverseColor(ByVal vbCol As Long) As Long

Dim colDecompose As Col_Sep
colDecompose = SepareColor(vbCol)
GetInverseColor = RGB(255 - colDecompose.red, 255 - colDecompose.green, 255 - colDecompose.blue)

End Function

Function SepareColor(ByVal ColRGB As Long) As Col_Sep

With SepareColor
    .red = Int(ColRGB And &HFF)
    .green = Int((ColRGB And &H100FF00) / &H100)
    .blue = Int((ColRGB And &HFF0000) / &H10000)
End With

End Function
    
```

## Comment obtenir la taille graphique d'une chaîne de caractères ?

**Auteurs : ThierryAIM , Romain Puyfoulhoux ,**

En utilisant la fonction `GetTextExtentPoint32()` de l'API Windows. Celle-ci reçoit en argument un contexte graphique, le texte dont vous voulez connaître la taille, le nombre de caractères, ainsi qu'une structure de type `POINTAPI` qui recevra la longueur et la hauteur du texte en pixels. En VB vous avez accès au contexte graphique d'une form ou d'un `PictureBox` grâce à la propriété `hdc`.

**Copiez ce code dans un module standard. La fonction `TailleDuTexte` vous renvoie une structure de type `POINTAPI` contenant la largeur et la hauteur du texte passé en paramètre.**

```

vb

Public Type POINTAPI
    X As Long
    Y As Long
End Type
Private Declare Function GetTextExtentPoint32 Lib "gdi32" Alias _
    "GetTextExtentPoint32A" (ByVal hdc As Long, ByVal lpsz As String, _
        ByVal cbString As Long, lpSize As POINTAPI) As
    Long

Public Function TailleDuTexte(ByVal hdc As Long, ByVal texte As String) As POINTAPI

    Dim taille As POINTAPI

    GetTextExtentPoint32 hdc, texte, Len(texte), taille
    taille.X = taille.X * Screen.TwipsPerPixelX
    taille.Y = taille.Y * Screen.TwipsPerPixelY
    TailleDuTexte = taille

End Function
    
```

**Voici par exemple comment afficher la largeur du texte contenu dans un label :**

vb

```
Dim taille As POINTAPI
MsgBox TailleDuTexte(Me.hdc, Label1.Caption).X
```

La fonction `GetTextExtentPoint32` calcule la taille du texte en utilisant comme police celle liée au contexte graphique passé en paramètre. Donc pour que le résultat soit exact, la police de la form doit être la même que celle du label.

## Comment afficher un texte incliné par rapport à sa ligne de base ?

Auteurs : Khany , Romain Puyfoulhoux ,

Voici comment écrire un texte incliné par rapport à la ligne de base. Le principe consiste à créer la police désirée avec la fonction `CreateFontIndirect` de l'api Win 32.

Copiez ce code dans un module standard :

vb

```
Private Declare Function CreateFontIndirect Lib "gdi32" Alias "CreateFontIndirectA" _
    (lpLogFont As LOGFONT) As Long
Private Declare Function SelectObject Lib "gdi32" (ByVal hdc As Long, _
    ByVal hObject As Long) As Long
Private Declare Function DeleteObject Lib "gdi32" (ByVal hObject As Long) As Long

Private Const LF_FACESIZE = 32
Private Const FW_NORMAL = 400
Private Const FW_BOLD = 700

Private Type LOGFONT
    lfHeight As Long
    lfWidth As Long
    lfEscapement As Long
    lfOrientation As Long
    lfWeight As Long
    lfItalic As Byte
    lfUnderline As Byte
    lfStrikeOut As Byte
    lfCharSet As Byte
    lfOutPrecision As Byte
    lfClipPrecision As Byte
    lfQuality As Byte
    lfPitchAndFamily As Byte
    lfFaceName As String * LF_FACESIZE
End Type

Public Type Police
    taille As Integer
    angle As Integer
    gras As Boolean
    italique As Boolean
    souligne As Boolean
    nom As String
End Type

Public Sub TexteIncline(device As Object, ByVal texte As String, font As Police)

    Dim fnt As LOGFONT, prevFont As Long, hFont As Long, ret As Long

    fnt.lfEscapement = font.angle * 10 'lfEscapement est en dixième de degrés
    fnt.lfFaceName = font.nom & Chr$(0) 'lfFaceName doit se terminer par un caractère nul
    fnt.lfItalic = IIf(font.italique, 1, 0)
```

vb

```
fnt.lfUnderline = IIf(font.souligne, 1, 0)
fnt.lfWeight = IIf(font.gras, FW_BOLD, FW_NORMAL)

' Windows attend la taille de caractères en pixels et en
' négatif si vous spécifiez la hauteur de caractères désirée
fnt.lfHeight = (font.taille * -20) / Screen.TwipsPerPixelY

hFont = CreateFontIndirect(fnt) 'création de la police
prevFont = SelectObject(device.hdc, hFont) 'handle sur la police actuelle
device.Print texte

ret = SelectObject(device.hdc, prevFont) 'restauration de la police d'origine
ret = DeleteObject(hFont) 'suppression de l'objet créé
```

End Sub

La procédure `TexteIncline` écrit un texte dans l'objet spécifié en paramètre. Cet objet doit avoir une méthode `print`, vous pouvez donc choisir une form, l'objet `Printer` ou un `PictureBox`. Le paramètre `font` est une variable de type `Police`, qui est un type utilisateur créé pour l'occasion, et qui est plus simple à paramétrer que le type `LOGFONT` des api Win32.

Pour tester ce code, posez un `PictureBox` et un bouton de commande sur une form et placez ce code dans le module de la form :

vb

```
Private Sub Command1_Click()

Dim font As Police

font.nom = "Arial"
font.taille = 14 'taille de 14 points
font.angle = 50 '50 degrés par rapport à l'horizontale
font.italique = True
font.gras = True
font.souligne = True

Picture1.CurrentX = Picture1.ScaleWidth / 2
Picture1.CurrentY = Picture1.ScaleHeight / 2
TexteIncline Picture1, "Texte incliné", font
```


End Sub

Seules les polices True-type peuvent être utilisées pour écrire un texte de façon inclinée.

## Sommaire > Système

### Comment déterminer l'identifiant et la langue du système d'exploitation ?

Auteurs : mdriesbach , ThierryAIM ,

La liste complète des identifiants de langage est disponible à l'adresse suivante :  [MSDN : Language Identifiers and Locales](#)

Placez ce code dans la section déclaration de votre form ou dans un module :

vb

```
Private Declare Function GetSystemDefaultLangID Lib "kernel32" () As Long
Private Declare Function VerLanguageName Lib "kernel32" Alias "VerLanguageNameA" _
(ByVal wLang As Long, ByVal szLang As String, ByVal nSize As Long) As Long

Private Enum pLang
    LangID = 1
    LangName = 2
End Enum

Private Function GetSystemLanguage(param As pLang) As String
    Dim ID As String
    Dim Buffer As String
    ID = "&H" & Right(Hex(GetSystemDefaultLangID()), 3)
    Select Case param
    Case 1
        GetSystemLanguage = ID & " / " & CStr(Val(ID))
    Case 2
        Buffer = String(255, 0)
        VerLanguageName CLng(ID), Buffer, Len(Buffer)
        Buffer = Left$(Buffer, InStr(1, Buffer, Chr$(0)) - 1)
        GetSystemLanguage = Buffer
    End Select
End Function
```

Exemple :

vb

```
MsgBox "L'ID langue est : " & GetSystemLanguage(LangID)
MsgBox "Votre système est en : " & GetSystemLanguage(LangName)
```

### Comment savoir si mon clavier est en majuscule ou pas?

Auteurs : Tofalu ,

```
Public Declare Function GetKeyState Lib "user32" (ByVal iVirtualKey As Integer) As Long

Public Function Is_Majuscule() As Boolean
    Is_Majuscule = (&H1 And GetKeyState(vbKeyCapital)) <> 0
End Function
```



La fonction `Is_Majuscule` renvoie true si le clavier est en majuscule, false sinon.

lien : [Comment Activer/Désactiver le Caps Lock du clavier](#)

## Comment Activer/Désactiver le Caps Lock du clavier

Auteurs : [Maxence HUBICHE](#) ,

```
Private Type KeyboardBytes
    kbByte(0 To 255) As Byte
End Type
Enum apiOnOff
    apiOn = 1
    apiOff = 0
End Enum
Dim kbArray As KeyboardBytes

Private Declare Function GetKeyboardState Lib "user32" (kbArray As KeyboardBytes) As Long
Private Declare Function SetKeyboardState Lib "user32" (kbArray As KeyboardBytes) As Long

Private Sub ChangerCapsLock(v As apiOnOff)
    GetKeyboardState kbArray
    kbArray.kbByte(&H14) = v
    SetKeyboardState kbArray
End Sub
```

Utiliser sous la forme pour désactiver :

```
ChangerCapsLock apiOff
```

ou, pour activer :

```
ChangerCapsLock apiOn
```

## Comment déterminer les variables d'environnement du système (Windows 2000 et +) ?

Auteurs : [DarkVader](#) ,

A partir de Windows 2000, vous pouvez retrouver facilement les variables d'environnement du système à l'aide de l'astuce suivante :

(retourne dans la fenêtre d'exécution, toutes les variables d'environnement avec leur nom et leur valeur)

```
vb

Dim i As Integer
For i = 1 To 255
    If Environ(i) <> "" Then Debug.Print Environ(i)
Next
```

Dans un projet, vous pouvez utiliser, par exemple, le nom de la variable d'environnement dans votre code pour en extraire sa valeur :

```
vb
```

vb

```
MsgBox Environ("USERNAME")
```

## Comment exécuter des commandes Dos ?

Auteurs : Romain Puyfoulhoux ,

vb

```
'le paramètre /  
c indique que la console Dos doit se fermer sitôt l'exécution de la commande terminée.  
Shell "command.com /c dir *.* > c:\liste.txt"
```

## Comment exécuter un programme ?

Auteurs : Romain Puyfoulhoux ,

Utilisez la fonction Shell. Vous pouvez indiquer dans le deuxième paramètre comment la fenêtre du programme doit s'afficher (si elle doit rester invisible, ou s'afficher normalement, ou en prenant tout l'écran, etc...).

vb

```
Dim ret As Long  
ret = Shell("notepad.exe", vbNormalFocus)
```

lien : [FAQ](#) Comment ouvrir un fichier HTML, Word ou autre en utilisant l'exécutable associé ?

## Comment fermer un programme ouvert avec la fonction Shell ?

Auteurs : Romain Puyfoulhoux ,

Vous trouverez une méthode possible dans le code source ci-dessous. La procédure KillApp() ferme le programme dont l'identifiant est passé en paramètre. Vous pouvez utiliser la valeur renvoyée par la fonction Shell().

La seule ligne contenue dans la procédure KillApp() a pour effet d'énumérer toutes les fenêtres ouvertes, et d'appeler pour chacune d'entre-elles la fonction CloseWindow().

La fonction CloseWindow() regarde si la fenêtre en cours appartient au processus que l'on doit fermer, et si elle contient un menu Système (menu qui apparaît quand on clique sur l'icône de la fenêtre). Si c'est le cas, elle ferme la fenêtre.

Insérez ce code dans un module standard:

vb

```
Private Declare Function EnumWindows Lib "user32" (ByVal lpEnumFunc As Long, ByVal lParam As Long) As Long  
Private Declare Function GetWindowThreadProcessId Lib "user32" (ByVal hwnd As Long, _  
lpdwprocessid As Long) As Long  
Private Declare Function PostMessage Lib "user32" Alias "PostMessageA" (ByVal hwnd As Long, _  
ByVal wParam As Long, ByVal lParam As Long) As Long  
Private Declare Function GetWindowLong Lib "user32" Alias "GetWindowLongA" (ByVal hwnd As Long, _  
ByVal nIndex As Long) As Long  
Private Const WM_CLOSE = &H10  
Private Const GWL_STYLE = (-16)  
Private Const WS_SYSMENU = &H80000  
  
Private Function CloseWindow(ByVal hwnd As Long, ByVal hInstance As Long) As Long
```

vb

```
Dim idproc As Long

idproc = 0

'reçoit dans idproc l'id du processus lié à cette fenêtre
GetWindowThreadProcessId hwnd, idproc
If (idproc = hInstance) And ((GetWindowLong(hwnd, GWL_STYLE) And WS_SYSMENU) = WS_SYSMENU) Then
    PostMessage hwnd, WM_CLOSE, 0, 0
End If

'obligatoire pour qu'EnumWindows continue l'énumération
CloseWindow = True

End Function

Public Sub KillApp(hInstance As Long)

EnumWindows AddressOf CloseWindow, hInstance

End Sub
```

Certains programmes n'acceptent pas d'être ouverts plusieurs fois en même temps. Pour cela, ils commencent par chercher si une instance du programme est déjà en mémoire. Si c'est le cas, ils terminent l'instance qui vient d'être créée, et si une nouvelle fenêtre doit être ouverte, c'est l'ancienne instance qui le fait.

*Avec ce code source vous ne pourrez donc pas, par exemple, fermer une fenêtre de l'explorateur Windows ouverte par Shell(), car le processus dont l'id vous est renvoyé est automatiquement fermé, et la nouvelle fenêtre est ouverte par l'instance qui était déjà en mémoire.*

lien : [FAQ](#) Comment tuer un processus en connaissant le nom de sa fenêtre ?

## Comment lister les processus actifs sans utiliser l'API Windows (Win2000 et >) ?

Auteurs : [ThierryAIM](#),

Il est possible de lister les processus actifs sur une machine en utilisant les fonctions de l'API Windows. Mais il existe une astuce beaucoup plus simple à l'aide d'un petit script WMI, utilisant la classe Win32\_process (testé sur Windows 2000, XP)

vb

```
Private Sub Command1_Click()
    Dim svc As Object
    Dim sQuery As String
    Dim oproc
    On Error GoTo Command1_Click_Error

    Set svc = GetObject("winmgmts:root\cimv2")
    sQuery = "select * from win32_process"
    For Each oproc In svc.execquery(sQuery)
        Debug.Print oproc.Name & " = " & oproc.ExecutablePath
    Next
    Set svc = Nothing
    Exit Sub

Command1_Click_Error:
    MsgBox "Error " & Err.Number & " (" & Err.Description & ")"
    Err.Clear
End Sub
```

Retrouvez toutes les informations sur la classe Win32\_process (propriétés et méthodes) sur le site de Microsoft MSDN :

 [Win32\\_Process WMI class](#)

Attention, certaines propriétés ne sont pas implémentées. Lire attentivement les descriptions et explications fournies sur le site !

lien : [FAQ Comment "tuer" un processus en mémoire \(Win2000 et >\)?](#)

lien :  [Microsoft MSDN : WMI Scripting Primer: Part 1](#)

## Comment "tuer" un processus en mémoire (Win2000 et >)?

Auteurs : [ThierryAIM](#) ,

En complément de la question précédente, il est possible d'arrêter un processus par le code, en le détruisant directement dans la liste des processus actifs de Windows

vb

```
Public Function KillProcess(ByVal ProcessName As String) As Boolean
    Dim svc As Object
    Dim sQuery As String
    Dim oproc
    Set svc = GetObject("winmgmts:root\cimv2")
    sQuery = "select * from win32_process where name='" & ProcessName & "'"
    For Each oproc In svc.execquery(sQuery)
        oproc.Terminate
    Next
    Set svc = Nothing
End Function
```

Exemple pour Acrobat Reader :

vb

```
KillProcess "AcroRd32.exe"
```

lien : [FAQ Comment lister les processus actifs sans utiliser l'API Windows \(Win2000 et >\) ?](#)

lien :  [Microsoft MSDN : WMI Scripting Primer: Part 1](#)

## Comment verrouiller la station de travail ?

Auteurs : [Tofalu](#) ,

Il faut utiliser l'API Windows et plus particulièrement la fonction LockStation  
Dans un module :

```
Private Declare Function LockWorkStation Lib "user32.dll" () As Long
Public Sub Verrouiller()
    LockWorkStation
End Sub
```

Il suffit alors d'appeler la méthode Verrouiller là où vous en avez besoin.

## Comment arrêter ou bien redémarrer le système ?

Auteurs : Tofalu ,

Pour cela, il faut utiliser l'API ExitWindowsEx.  
Dans un module placer les déclarations suivantes :

```
Public Const EWX_LOGOFF = 0
Public Const EWX_SHUTDOWN = 1
Public Const EWX_REBOOT = 2
Public Const EWX_FORCE = 4
Public Declare Function ExitWindowsEx Lib "user32" (ByVal uFlags As Long, ByVal dwReserved As Long)
```

La constante LOGOFF ferme la session, SHUTDOWN arrête la machine, REBOOT redémarre. La constante FORCE peut être utiliser en addition d'une des 3 autres afin de forcer l'arrêt des applications sans demande de confirmation de sortie.

Exemple pour arrêter l'ordinateur :

```
ExitWindowsEx(EWX_SHUTDOWN, 0)
```

La même chose en forçant l'arrêt des applications :

```
ExitWindowsEx(EWX_SHUTDOWN OR EWX_FORCE, 0)
```

## Comment réduire la fenêtre d'une application ?

Auteurs : LedZeppII ,

Pour réduire une fenêtre d'une application , il faut passer par des API

ces déclarations sont à mettre au début du module :

```
' Déclarations
Const WM_SYSCOMMAND As Long = &H112

Const SC_MINIMIZE As Long = &HF020&
Const SC_MAXIMIZE As Long = &HF030&
Const SC_RESTORE As Long = &HF120&

Private Declare Function PostMessage Lib "User32.dll" Alias "PostMessageA" ( _
ByVal hwnd As Long, ByVal MSG As Long, ByVal wParam As Long, ByVal lParam As Long) As Long

Private Declare Function FindWindow1 Lib "User32.dll" Alias "FindWindowA" ( _
ByVal lpClassName As Long, ByVal lpWindowName As String) As Long
```

Il faut utiliser la fonction suivante pour fermer la fenêtre, celle-ci fonctionne avec le titre de la fenêtre à réduire :

```
Function AppMinimize(AppTitle As String) As Boolean
Dim hwnd As Long
```

```
hwnd = FindWindow(0, AppTitle & vbNullChar)
If hwnd <> 0 Then
PostMessage hwnd, WM_SYSCOMMAND, SC_MINIMIZE, 0
AppMinimize = True
Else
AppMinimize = False ' Fenêtre pas trouvée
End If
End Function
```

Pour comprendre le fonctionnement vous pouvez faire un essai avec le Bloc-notes, ouvrez Bloc-notes, la fenêtre s'appelle : "Sans titre - Bloc-Notes".

mettez ce code sur un événement de votre choix :

```
Sub quicktest()
Dim AppTitle As String, tmr As Long

AppTitle = "Sans titre - Bloc-Notes"
AppActivate AppTitle
tmr = Timer
While ((Timer - tmr) < 2)
Wend
AppMinimize AppTitle

End Sub
```

Ouvrez votre Application en pleine fenêtre et la fenêtre Bloc-notes en niveau inférieur, sur appel du code ci-dessus la fenêtre Bloc-notes se réduira après timer écoulé.

## Comment lister et modifier les services de Windows 2000 /XP ?

Auteurs : [ThierryAIM](#) ,

Dans le même esprit que les questions précédentes, il est possible, par un script WMI, de lister et d'intervenir sur les services Windows, à l'aide des propriétés et méthodes de la classe Win32\_Service :

Exemple pour lister tous les services actifs et leur état (activé ou stoppé) :

```
vb

Dim svc As Object
Dim sQuery As String
Dim oserv
On Error GoTo Command1_Click_Error

Set svc = GetObject("winmgmts:root\cimv2")
sQuery = "select * from win32_service"
For Each oserv In svc.execquery(sQuery)
Debug.Print oserv.Name & " : " & oserv.PathName & " : " & oserv.State
Next
Set svc = Nothing
Exit Sub

Command1_Click_Error:
MsgBox "Error " & Err.Number & " (" & Err.Description & ")"
Err.Clear
```

Retrouvez toutes les informations sur la classe Win32\_Service (propriétés et méthodes) sur le site de Microsoft MSDN :

 [Win32\\_Service WMI class](#)

lien : [FAQ](#) Comment lister les processus actifs sans utiliser l'API Windows (Win2000 et >) ?

lien :  [Microsoft MSDN : WMI Scripting Primer: Part 1](#)

## Comment obtenir le numéro de série unique du processeur machine (Windows 2000 et >) ?


Auteurs : [ThierryAIM](#) ,

Toujours à l'aide d'un script WMI utilisant la classe Win32\_processor :

```
vb
Private Sub Command1_Click()
    Dim svc As Object
    Dim oproc
    On Error GoTo Command1_Click_Error

    Set svc = GetObject("winmgmts:root\cimv2")
    For Each oproc In svc.execquery("select * from Win32_Processor ")
        Debug.Print oproc.Name & " = " & oproc.ProcessorId
    Next
    Set svc = Nothing
    Exit Sub

Command1_Click_Error:
    MsgBox "Error " & Err.Number & " (" & Err.Description & ")"
    Err.Clear
End Sub
```

Retrouvez toutes les informations sur la classe Win32\_processor (propriétés et méthodes) sur le site de Microsoft MSDN :  [Win32\\_Processor WMI class](#)

lien :  [Microsoft MSDN : WMI Scripting Primer: Part 1](#)

## Comment tuer un processus en connaissant le nom de sa fenêtre ?

Auteurs : [Jean-Marc Rabilloud](#) ,

Le code suivant est assez similaire à celui de la question précédente, la différence résidant dans le fait que vous ne connaissez pas l'identifiant du processus mais juste le titre de la fenêtre.

Ce code doit être placé dans un module standard. Le principe utilisé est celui d'une énumération à l'aide d'un callback classique : la fonction EnumWindows énumère toutes les fenêtres ouvertes et appelle la fonction EnumCallback pour chacune d'entre elles. Celle-ci ferme la fenêtre si son titre contient l'expression recherchée.

```
vb
Declare Function EnumWindows Lib "user32" (ByVal wndenmproc As Long, ByVal lParam As Long) As Long
Declare Function GetWindowText Lib "user32" Alias "GetWindowTextA" (ByVal hwnd As Long, _
    ByVal lpString As String, _
    ByVal cch As Long) As Long

Declare Function SendMessage Lib "user32" Alias "SendMessageA" (ByVal hwnd As Long, ByVal wParam As
Long, _
    ByVal lParam As Long, _
    ByVal wParam As Long, _
    lParam As Any) As Long
```

```
vb
Public Const WM_CLOSE = &H10

Private AppCible As String

Public Function EnumCallback(ByVal app_hWnd As Long, ByVal param As Long) As Long

Dim buf As String * 256
Dim Titre As String
Dim Longueur As Long

'Recupère le titre de la fenêtre
Longueur = GetWindowText(app_hWnd, buf, Len(buf))
Titre = Left$(buf, Longueur)

'Vérifie si le titre de la fenêtre correspond au nom recherché
If InStr(Titre, AppCible) <> 0 Then
'Ferme la fenêtre
SendMessage app_hWnd, WM_CLOSE, 0, 0
End If

'Poursuit l'énumération
EnumCallback = 1

End Function

Public Sub KillApp(App_Cherchee As String)

AppCible = App_Cherchee
'Demande à Windows d'énumérer les fenêtres ouvertes
EnumWindows AddressOf EnumCallback, 0

End Sub
```

Un appel du type KillApp "Excel" fermera Microsoft Excel. Attention d'éviter l'utilisation de termes trop simples pouvant se trouver dans le titre d'une fenêtre d'une autre application.

lien : [FAQ](#) Comment fermer un programme ouvert avec la fonction Shell ?

## Comment lire / écrire dans un fichier .ini ?

Auteurs : [Romain Puyfoulhoux](#) ,

Les fichiers .ini sont des fichiers texte utilisés pour enregistrer les options d'un programme. Ils sont composés de sections, qui contiennent des clés auxquelles on peut donner une valeur. Par exemple :

```
[Affichage]
State=Maximized
Left=50
Top=80
[Sauvegarde]
Confirm=True
Auto=False
```

Pour pouvoir respectivement lire et écrire dans un fichier .ini, voici les déclarations que vous devez ajouter dans votre module :

```
vb
Private Declare Function GetPrivateProfileString Lib "kernel32" Alias "GetPrivateProfileStringA" _
    (ByVal lpApplicationName As String, ByVal lpKeyName As Any, ByVal lpDefault As String, _
    ByVal lpReturnedString As String, ByVal nSize As Long, ByVal lpFileName As String) As Long
```



vb

```
Private Declare Function WritePrivateProfileString Lib "kernel32" Alias "WritePrivateProfileStringA"  
    _  
    (ByVal lpApplicationName As String, ByVal lpKeyName As Any, ByVal lpString As Any, _  
    ByVal lpFileName As String) As Long
```

Voici ci-dessous la fonction qui écrira une valeur pour la clé et dans la section indiquée. Notez que vous n'avez pas besoin de créer le fichier s'il n'existe pas, car la fonction WritePrivateProfileString le fait pour vous.

vb

```
Private Function EcritDansFichierIni(Section As String, Cle As String, _  
    Valeur As String, Fichier As String) As Long  
EcritDansFichierIni = WritePrivateProfileString(Section, Cle, Valeur, Fichier)  
End Function
```

Et voyons maintenant la fonction qui nous retournera la valeur d'une clé dans une section donnée. ValeurParDefaut contient la valeur qui devra nous être retournée si le fichier n'existe pas, ou si aucune valeur n'a été spécifiée pour la clé demandée :

vb

```
Private Function LitDansFichierIni(Section As String, Cle As String, Fichier As String, _  
    Optional ValeurParDefaut As String = "") As String  
  
Dim strReturn As String  
strReturn = String(255, 0)  
GetPrivateProfileString Section, Cle, ValeurParDefaut, strReturn, Len(strReturn), Fichier  
LitDansFichierIni = Left(strReturn, InStr(strReturn, Chr(0)) - 1)  
  
End Function
```

Le code nécessaire pour écrire la section [Affichage] du fichier donné en exemple sera :

vb

```
EcritDansFichierIni "Affichage", "State", "Maximized", "c:\config.ini"  
EcritDansFichierIni "Affichage", "Left", "50", "c:\config.ini"  
EcritDansFichierIni "Affichage", "Top", "80", "c:\config.ini"
```

Et nous pourrons lire la valeur donnée à la clef "Left" avec :

vb

```
LeftParam = LitDansFichierIni("Affichage", "Left", "c:\config.ini", 100)
```

lien : [FAQ](#) Comment lister toutes les sections d'un fichier .ini ?

lien : [FAQ](#) Comment lister toutes les clés et valeurs d'une section d'un fichier .ini ?

Comment lister toutes les sections d'un fichier .ini ?

Auteurs : [Catbull](#) , [ThierryAIM](#) ,

Déclaration :

vb

```
Private Declare Function GetPrivateProfileSectionNames Lib "kernel32.dll"
Alias "GetPrivateProfileSectionNamesA" _
(ByVal lpzReturnBuffer As String, ByVal nSize As Long, ByVal lpFileName As String) As Long
```

Paramètre *lpzReturnBuffer* : adresse d'un tampon qui va recevoir la ou les sections du fichier .ini.  
Chaque nom de section est terminé par un caractère *null* (Chr(0)=vbNullChar), le dernier est suivi d'un second caractère *null*.

vb

```
Private Function ListeSectionIni(ByVal Path As String, Section() As String)
Dim strReturn As String
strReturn = String(8192, 0)

GetPrivateProfileSectionNames strReturn, Len(strReturn), Path

Section = Split(Left(strReturn, InStr(1, strReturn, vbNullChar & vbNullChar) - 1), vbNullChar)
End Function

Private Sub Command1_Click()
Dim Section() As String

ListeSectionIni "C:\test.ini", Section '-- Paramètre Section passé ByRef
For Index = LBound(Section) To UBound(Section)
Debug.Print Section(Index)
Next
End Sub
```

lien : [FAQ](#) Comment lire / écrire dans un fichier .ini ?

lien : [FAQ](#) Comment lister toutes les clés et valeurs d'une section d'un fichier .ini ?

## Comment lister toutes les clés et valeurs d'une section d'un fichier .ini ?

Auteurs : [Catbull](#), [ThierryAIM](#),

Déclaration :

vb

```
Private Declare Function GetPrivateProfileSection Lib "kernel32" Alias "GetPrivateProfileSectionA" _
(ByVal lpAppName As String, ByVal lpReturnedString As String, ByVal nSize As Long, ByVal
lpFileName As String) As Long
```

Paramètre *lpReturnedString* : adresse d'un tampon qui va recevoir la ou les clés et valeurs de la section du fichier .ini.  
Chaque clé est terminée par un caractère *null* (Chr(0)=vbNullChar), la dernière est suivi d'un second caractère *null*.

vb

```
Public Function ListeSectionKey(ByVal Path As String, ByVal Section As String, Key() As String)
Dim strReturn As String
strReturn = String(8192, 0)

GetPrivateProfileSection Section, strReturn, 8192, Path

Key = Split(Left(strReturn, InStr(1, strReturn, vbNullChar & vbNullChar) - 1), vbNullChar)
End Function

Private Sub Command1_Click()
```

vb

```
Dim Key() As String

ListeSectionKey "C:\test.ini", "SectionName1", Key '-- le paramètre Key est passé byRef
For Index = LBound(Key) To UBound(Key)
    Debug.Print Key(Index)
Next
End Sub
```

lien : [FAQ](#) Comment lister toutes les sections d'un fichier .ini ?

lien : [FAQ](#) Comment lire / écrire dans un fichier .ini ?

## Comment ouvrir un fichier HTML, Word ou autre en utilisant l'exécutable associé ?

Auteurs : [Romain Puyfoulhoux](#) ,

Placez cette déclaration dans le module d'une form :

vb

```
Private Declare Function ShellExecute Lib "shell32.dll" Alias "ShellExecuteA" _
    (ByVal hwnd As Long, ByVal lpOperation As String, ByVal lpFile As String, _
    ByVal lpParameters As String, ByVal lpDirectory As String, ByVal nShowCmd As Long) As Long
```

La ligne suivante affiche le site Developpez.com dans le navigateur par défaut, en fournissant le répertoire de votre application comme répertoire par défaut :

vb

```
ShellExecute Me.hwnd, "open", "http://www.developpez.com", "", App.Path, 1
```

lien : [FAQ](#) Comment exécuter un programme ?

## Comment connaître la résolution de l'écran ?

Auteurs : [Romain Puyfoulhoux](#) ,

vb

```
Dim x As Long, y As Long
x = Screen.Width / Screen.TwipsPerPixelX 'résolution horizontale
y = Screen.Height / Screen.TwipsPerPixelY 'verticale
```

## Comment changer la résolution de l'écran ?

Auteurs : [Romain Puyfoulhoux](#) ,

Copiez ce code source dans un module. Vous pourrez alors changer la résolution par un simple appel à la procédure ResolutionEcran(). Pour passer par exemple à une résolution de 800 x 600 :

vb

```
ResolutionEcran 800, 600
```

vb

```
Private Declare Function EnumDisplaySettings Lib "user32" Alias "EnumDisplaySettingsA" _
    (ByVal lpszDeviceName As Long, ByVal iModeNum As Long, lpDevMode As Any) As Boolean

Private Declare Function ChangeDisplaySettings Lib "user32" Alias "ChangeDisplaySettingsA" _
    (lpDevMode As Any, ByVal dwflags As Long) As Long

Private Const CCHDEVICENAME = 32
Private Const CCHFORMNAME = 32
Private Const DM_WIDTH = &H80000
Private Const DM_HEIGHT = &H100000

Private Type DEVMODE
    dmDeviceName As String * CCHDEVICENAME
    dmSpecVersion As Integer
    dmDriverVersion As Integer
    dmSize As Integer
    dmDriverExtra As Integer
    dmFields As Long
    dmOrientation As Integer
    dmPaperSize As Integer
    dmPaperLength As Integer
    dmPaperWidth As Integer
    dmScale As Integer
    dmCopies As Integer
    dmDefaultSource As Integer
    dmPrintQuality As Integer
    dmColor As Integer
    dmDuplex As Integer
    dmYResolution As Integer
    dmTTOption As Integer
    dmCollate As Integer
    dmFormName As String * CCHFORMNAME
    dmUnusedPadding As Integer
    dmBitsPerPel As Integer
    dmPelsWidth As Long
    dmPelsHeight As Long
    dmDisplayFlags As Long
    dmDisplayFrequency As Long
End Type

Public Sub ResolutionEcran(sgWidth As Long, sgHeight As Long)

Dim blTMP As Boolean, lgTMP As Long, dmEcran As DEVMODE, res As Long

lgTMP = 0
Do
    blTMP = EnumDisplaySettings(0, lgTMP, dmEcran)
    lgTMP = lgTMP + 1
Loop While blTMP <> 0

dmEcran.dmFields = DM_WIDTH Or DM_HEIGHT
dmEcran.dmPelsWidth = sgWidth
dmEcran.dmPelsHeight = sgHeight
lgTMP = ChangeDisplaySettings(dmEcran, 0)

End Sub
```

## Comment détecter le changement de la résolution de l'écran ?

**Auteurs : Romain Puyfoulhoux ,**

**Sous Windows, toutes les fenêtres des applications reçoivent le message WM\_DISPLAYCHANGE quand la résolution a changé. Le principe consiste donc à intercepter ce message grâce au sous-classement.**

Copiez ce code source dans le module de la form.

```
vb

Private Sub Form_Load()

    'Remplace la procédure de fenêtre par défaut par notre propre procédure
    oldWndProc = SetWindowLong(hwnd, GWL_WNDPROC, AddressOf WindowProc)

End Sub

Private Sub Form_Unload(Cancel As Integer)

    'Remet la procédure de fenêtre par défaut
    SetWindowLong hwnd, GWL_WNDPROC, oldWndProc

End Sub
```

Et celui-ci dans un module standard.

```
vb

Public Declare Function SetWindowLong Lib "user32" Alias "SetWindowLongA" _
    (ByVal hwnd As Long, ByVal nIndex As Long, _
    ByVal dwNewLong As Long) As Long

Public Const GWL_WNDPROC = (-4)
Public oldWndProc As Long

Private Declare Function CallWindowProc Lib "user32" Alias "CallWindowProcA" _
    (ByVal lpPrevWndFunc As Long, ByVal hwnd As Long, _
    ByVal msg As Long, ByVal wParam As Long, ByVal lParam As
    Long) As Long
Private Const WM_DISPLAYCHANGE = &H7E

Public Function WindowProc(ByVal hwnd As Long, ByVal msg As Long, _
    ByVal wParam As Long, ByVal lParam As Long) As Long

    If msg = WM_DISPLAYCHANGE Then
        'la résolution a changé
    End If

    'Appelle la procédure de fenêtre par défaut pour que Windows puisse traiter l'évènement
    WindowProc = CallWindowProc(oldWndProc, hwnd, msg, wParam, lParam)

End Function
```

Attention, la procédure Form\_Unload doit obligatoirement être exécutée. Si vous déboguez et cliquez sur Stop, l'éditeur VB plantera. Si vous fermez votre programme avec l'instruction End, la procédure Form\_Unload ne sera pas exécutée et votre programme plantera.

lien : [FAQ](#) Qu'est-ce que le sous classement ?

## Comment énumérer les polices disponibles ?

Auteurs : [Romain Puyfoulhoux](#) ,

Les polices disponibles à l'écran et à l'impression sont respectivement dans la collection Fonts de l'objet Screen et dans la collection Fonts de l'objet Printer. Le code source ci-dessous montre comment remplir une combo box avec les polices disponibles à l'écran.

vb

```
Dim i As Long

For i = 0 To Screen.FontCount - 1
    Combo1.AddItem Screen.Fonts(i)
Next
```

## Comment faire une pause pendant un temps défini ?

Auteurs : **Romain Puyfoulhoux** ,

Placez cette ligne dans la partie Déclarations d'un module :

vb

```
Private Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As Long)
```

Vous pourrez ainsi faire une pause de 2 secondes avec l'appel suivant :

vb

```
Sleep 2000
```

## Comment récupérer les paramètres régionaux, comme le séparateur décimal ou celui des milliers ?

Auteurs : **Romain Puyfoulhoux** ,

Les paramètres régionaux s'obtiennent grâce à la fonction `GetLocaleInfo()` de l'API Windows. Les paramètres de cette fonction sont :

**locale** : identifiant représentant le type d'information locale demandé (système ou utilisateur)

**LCTYPE** : valeur indiquant quel paramètre doit être retrouvé. Ce doit être une des constantes **LCTYPE**

**lpLCData** : buffer recevant la valeur du paramètre demandé

**cchData** : longueur du buffer

Voici les déclarations des deux fonctions dont vous aurez besoin, ainsi que quelques-unes des constantes **LCTYPE** disponibles :

vb

```
Private Declare Function GetLocaleInfo Lib "kernel32" Alias "GetLocaleInfoA" (ByVal locale As Long,
    ByVal LCTYPE As Long, ByVal lpLCData As String, ByVal cchData As Long) As Long

Private Declare Function GetUserDefaultLCID Lib "kernel32" () As Long

Private Const LOCALE_IDATE = &H21           'format de date courte : 0 = M-J-A, 1 = J-M-A, 2 = A-M-J
Private Const LOCALE_ILDATE = &H22         'format de date longue
Private Const LOCALE_SCountry = &H6        'pays en toutes lettres
Private Const LOCALE_SNativeLangName = &H4 'langue, en toutes lettres
Private Const LOCALE_SThousand = &HF      'séparateur des milliers
Private Const LOCALE_SDecimal = &HE       'séparateur décimal
```

La fonction ci-dessous renvoie la valeur du paramètre régional dont la constante **LCTYPE** est passée en paramètre :

vb

```
Private Function ParametreRegional(parametre As Long) As String

Dim lngResultat As Long
Dim buffer As String
Dim pos As Integer
Dim locale As Long

'recupère l'identifiant de l'information locale de type utilisateur
locale = GetUserDefaultLCID()

'renvoie le nombre de caractères nécessaire pour recevoir la valeur du paramètre demandé
lngResultat = GetLocaleInfo(locale, parametre, buffer, 0)

buffer = String(lngResultat, 0)
GetLocaleInfo locale, parametre, buffer, lngResultat
pos = InStr(buffer, Chr(0))
If pos > 0 Then ParametreRegional = Left(buffer, pos - 1)
End Function
```

## Comment récupérer le chemin UNICODE d'un lecteur ?

Auteurs : Gaël Donat ,

```
Private Const RESOURCETYPE_ANY = &H0
Private Const RESOURCE_CONNECTED = &H1
Private Type NETRESOURCE
    dwScope As Long
    dwType As Long
    dwDisplayType As Long
    dwUsage As Long
    lpLocalName As Long
    lpRemoteName As Long
    lpComment As Long
    lpProvider As Long
End Type
Private Declare Function WNetOpenEnum Lib "mpr.dll" Alias "WNetOpenEnumA" (ByVal dwScope As Long, ByVal dwType As Long, _
    ByVal dwUsage As Long, lpNetResource As Any, lpEnum As Long) As Long
Private Declare Function WNetEnumResource Lib "mpr.dll" Alias "WNetEnumResourceA" (ByVal hEnum As Long, lpCount As Long, _
    lpBuffer As Any, lpBufferSize As Long) As Long
Private Declare Function WNetCloseEnum Lib "mpr.dll" (ByVal hEnum As Long) As Long
Private Declare Function lstrlen Lib "kernel32" Alias "lstrlenA" (ByVal lpString As Any) As Long
Private Declare Function lstrcpy Lib "kernel32" Alias "lstrcpyA" (ByVal lpString1 As Any, ByVal lpString2 As Any) As Long
Function LetterToUNC(DriveLetter As String) As String
    Dim hEnum As Long
    Dim NetInfo(1023) As NETRESOURCE
    Dim entries As Long
    Dim nStatus As Long
    Dim LocalName As String
    Dim UNCName As String
    Dim i As Long
    Dim r As Long

    ' Begin the enumeration
    nStatus = WNetOpenEnum(RESOURCE_CONNECTED, RESOURCETYPE_ANY, 0&, ByVal 0&, hEnum)

    LetterToUNC = DriveLetter

    'Check for success from open enum
    If ((nStatus = 0) And (hEnum <> 0)) Then
        ' Set number of entries
```

```

entries = 1024

' Enumerate the resource
nStatus = WNetEnumResource(hEnum, entries, NetInfo(0), CLng(Len(NetInfo(0))) * 1024)

' Check for success
If nStatus = 0 Then
    For i = 0 To entries - 1
        ' Get the local name
        LocalName = ""
        If NetInfo(i).lpLocalName <> 0 Then
            LocalName = Space(lstrlen(NetInfo(i).lpLocalName) + 1)
            r = lstrcpy(LocalName, NetInfo(i).lpLocalName)
        End If

        ' Strip null character from end
        If Len(LocalName) <> 0 Then
            LocalName = Left(LocalName, (Len(LocalName) - 1))
        End If

        If UCase$(LocalName) = UCase$(DriveLetter) Then
            ' Get the remote name
            UNCName = ""
            If NetInfo(i).lpRemoteName <> 0 Then
                UNCName = Space(lstrlen(NetInfo(i).lpRemoteName) + 1)
                r = lstrcpy(UNCName, NetInfo(i).lpRemoteName)
            End If

            ' Strip null character from end
            If Len(UNCName) <> 0 Then
                UNCName = Left(UNCName, (Len(UNCName) - 1))
            End If

            ' Return the UNC path to drive
            'added the [] to separate on printout only
            LetterToUNC = UNCName

            ' Exit the loop
            Exit For
        End If
    Next i
End If
End If

' End enumeration
nStatus = WNetCloseEnum(hEnum)
End Function

```

Appelez la fonction directement comme ceci :

```
LetterToUNC("E:")
```

et elle renvoie :

```
\\hp-ux004\oracle
```

Comment récupérer les chemins complets des répertoires Windows, System, et Windows\Temp ?

Auteurs : Romain Puyfoulhoux ,

Voici deux solutions possibles. Tout d'abord, les fonctions de l'API Windows :



vb

```
Private Declare Function GetWindowsDirectory Lib "kernel32" Alias "GetWindowsDirectoryA" _
    (ByVal lpBuffer As String, ByVal nSize As Long) As Long

Private Declare Function GetSystemDirectory Lib "kernel32" Alias "GetSystemDirectoryA" _
    (ByVal lpBuffer As String, ByVal nSize As Long) As Long

Private Declare Function GetTempPath Lib "kernel32" Alias "GetTempPathA" _
    (ByVal nBufferLength As Long, ByVal lpBuffer As String) As Long
```

Ces 3 fonctions de l'API Win32 renvoient respectivement les chemins complets des répertoires Windows, System, et Windows\Temp. Elles s'utilisent toutes les 3 de la même manière :

vb

```
Function GetWindowsDir() As String

    Dim buffer As String * 256
    Dim Length As Long
    Length = GetWindowsDirectory(buffer, Len(buffer))
    GetWindowsDir = Left(buffer, Length)

End Function

Function GetSystemDir() As String

    Dim buffer As String * 256
    Dim Length As Long
    Length = GetSystemDirectory(buffer, Len(buffer))
    GetSystemDir = Left(buffer, Length)

End Function

Function GetTempDir() As String

    Dim buffer As String * 256
    Dim Length As Long
    Length = GetTempPath(Len(buffer), buffer)
    GetTempDir = Left(buffer, Length)

End Function
```

Une autre manière de procéder est de faire appel au FileSystemObject :

vb

```
Dim fso As FileSystemObject

Set fso = New FileSystemObject

MsgBox fso.GetSpecialFolder(0) 'répertoire windows
MsgBox fso.GetSpecialFolder(1) 'répertoire system
MsgBox fso.GetSpecialFolder(2) 'répertoire temp

Set fso = Nothing
```

```
vb
End Sub
```

## Comment obtenir les chemins complets des répertoires spéciaux ?

Auteurs : Romain Puyfoulhoux ,

Pour récupérer les chemins complets des répertoires du Bureau, de Mes Documents, ou du menu Démarrer, vous pouvez utiliser la fonction SHGetSpecialFolderPath de l'Api Windows :

```
vb
Private Declare Function SHGetSpecialFolderPath Lib "shell32.dll" Alias "SHGetSpecialFolderPathA" _
    (ByVal hwndOwner As Long, ByVal lpszPath As String, _
    -
    ByVal nFolder As Long, ByVal fCreate As Long) As
    Long
```

Description des paramètres :

hwndOwner : handle de la fenêtre à utiliser si une boîte de dialogue doit être affichée

lpszPath : chaîne de caractères recevant le chemin complet du répertoire demandé

nFolder : nombre indiquant le répertoire demandé

fCreate : si la valeur passée à ce paramètre n'est pas nulle, le répertoire est créé, s'il n'existe pas déjà

Sous Windows NT 4.0 et Windows 95, cette fonction n'est disponible que si Internet Explorer 4.0 ou supérieur est installé. Voyons comment l'utiliser :

```
vb
Public Function GetSpecialFolderPath(dossier As Long, hwnd As Long)
    Dim buffer As String
    buffer = Space(256)
    SHGetSpecialFolderPath hwnd, buffer, dossier, 0
    GetSpecialFolderPath = Left(buffer, InStr(buffer, Chr(0)) - 1)
End Function
```

Pour tester cette fonction, placez par exemple ces 3 lignes dans une procédure du module d'une form :

```
vb
MsgBox GetSpecialFolderPath(0, Me.hwnd) 'répertoire du Bureau
MsgBox GetSpecialFolderPath(5, Me.hwnd) 'répertoire Mes Documents
MsgBox GetSpecialFolderPath(11, Me.hwnd) 'répertoire du menu Démarrer
```

Les réfractaires aux Api Windows préféreront utiliser le Windows Script Host Object Model en ajoutant wshom.ocx aux références du projet. Les chemins complets des répertoires spéciaux sont dans la collection SpecialFolders de l'objet WshShell.

```
vb
Dim Wsh As WshShell
Set Wsh = New WshShell
```

vb

```
MsgBox Wsh.SpecialFolders.Item("Desktop")      'répertoire du Bureau
MsgBox Wsh.SpecialFolders.Item("MyDocuments")  'répertoire Mes Documents
MsgBox Wsh.SpecialFolders.Item("StartMenu")    'répertoire du menu Démarrer

Set WshShell = nothing

End Sub
```

*Nota : Avec les anciennes versions de Wshom.ocx, la classe WshShell s'appelle IWshShell\_Class.*

ou encore :

vb

```
Set WshShell = CreateObject("Wscript.Shell")
MsgBox WshShell.SpecialFolders("Desktop")
MsgBox WshShell.SpecialFolders("MyDocuments")
MsgBox WshShell.SpecialFolders("StartMenu")
Set WshShell = nothing
```

## Comment connaître la version de Windows sur laquelle mon application est exécutée ?

**Auteurs : Romain Puyfoulhoux ,**

**La fonction VersionWindows() de ce code source retourne la version de Windows et place dans le paramètre sp le service pack qui serait éventuellement installé.**

vb

```
Private Declare Function GetVersionExA Lib "kernel32" (lpVersionInformation As OSVERSIONINFO) As Integer

Private Const VER_PLATFORM_WIN32_WINDOWS = 1
Private Const VER_PLATFORM_WIN32_NT = 2

Private Type OSVERSIONINFO
    dwOSVersionInfoSize As Long
    dwMajorVersion As Long
    dwMinorVersion As Long
    dwBuildNumber As Long
    dwPlatformId As Long
    szCSDVersion As String * 128
End Type

Public Function VersionWindows(ByRef sp As String) As String

    Dim os As OSVERSIONINFO

    os.dwOSVersionInfoSize = Len(os)
    GetVersionExA os
    sp = ""

    With os
        Select Case .dwPlatformId
            Case VER_PLATFORM_WIN32_WINDOWS
                Select Case .dwMinorVersion
                    Case 0
                        VersionWindows = "95"
                    Case 10
                        VersionWindows = "98"
                    Case 90

```

```

vb
        VersionWindows = "Me"
    End Select
Case VER_PLATFORM_WIN32_NT
    Select Case .dwMajorVersion
        Case 3
            VersionWindows = "NT 3.51"
        Case 4
            VersionWindows = "NT 4.0"
        Case 5
            If .dwMinorVersion = 0 Then
                VersionWindows = "2000"
            Else
                VersionWindows = "XP"
            End If
        End Select
    End Select

    If InStr(.szCSDVersion, Chr(0)) > 0 Then
        sp = Left(.szCSDVersion, InStr(.szCSDVersion, Chr(0)) - 1)
    End If
End With
End Function

```

lien : [FAQ Comment savoir si mon application VB6 et exécutée sous Windows VISTA ?](#)

## Comment savoir si mon application VB6 et exécutée sous Windows VISTA ?

**Auteurs :** [bbil](#) , [ProgElecT](#) ,

En utilisant la méthode décrite ici :

[FAQ](#) Comment connaître la version de Windows sur laquelle mon application est exécutée ?

Grâce à la fonction `GetVersionExA`, la variable de type `OSVERSIONINFO`, retourne pour sa propriété : `dwMajorVersion = 6` lorsque le système d'exploitation est Vista où Windows Serveur 2008.

Pour obtenir plus d'informations il faut passer par la fonction "étendue" de `GetVersion`, et le type de donnée `OSVERSIONINFOEX`, dont la propriété `wProductType` prend la valeur 1 pour les systèmes d'exploitation de type station de travail (Vista, XP , 2000)

```

Private Declare Function GetVersionExA Lib "kernel32" (lpVersionInformation As OSVERSIONINFOEX) As Integer

Private Const VER_PLATFORM_WIN32_WINDOWS = 1
Private Const VER_PLATFORM_WIN32_NT = 2

'wProductType
Private Const
    VER_NT_WORKSTATION = 1 'Windows Vista, Windows XP Professional, Windows XP Home Edition, or Windows 2000 Profes
Private Const
    VER_NT_DOMAIN_CONTROLLER = 2 'Controleur de domaine sous Windows Serveur 2008,2003 ou 2000
Private Const VER_NT_SERVER = 3 'Windows Serveur 2008 , 2003 or 2000

Private Type OSVERSIONINFOEX
    dwOSVersionInfoSize As Long
    dwMajorVersion As Long
    dwMinorVersion As Long
    dwBuildNumber As Long
    dwPlatformId As Long
    szCSDVersion As String * 128
    wServicePackMajor As Integer
    wServicePackMinor As Integer
    wSuiteMask As Integer

```

```
wProductType As Byte
wReserved As Byte
End Type

Public Function VersionWindows(ByRef sp As String) As String

    Dim os As OSVERSIONINFOEX

    os.dwOSVersionInfoSize = Len(os)
    GetVersionExA os
    sp = ""

    With os
        Select Case .dwPlatformId
            Case VER_PLATFORM_WIN32_WINDOWS
                Select Case .dwMinorVersion
                    Case 0
                        VersionWindows = "95"
                    Case 10
                        VersionWindows = "98"
                    Case 90
                        VersionWindows = "Me"
                End Select
            Case VER_PLATFORM_WIN32_NT
                Select Case .dwMajorVersion
                    Case 3
                        VersionWindows = "NT 3.51"
                    Case 4
                        VersionWindows = "NT 4.0"
                    Case 5
                        If .dwMinorVersion = 0 Then
                            VersionWindows = "2000"
                        Else
                            VersionWindows = "XP"
                        End If
                    Case 6
                        If .wProductType = VER_NT_WORKSTATION Then
                            VersionWindows = "Vista"
                        Else
                            VersionWindows = "Windows Server 2008"
                        End If
                End Select
        End Select

        If InStr(.szCSDVersion, Chr(0)) > 0 Then
            sp = Left(.szCSDVersion, InStr(.szCSDVersion, Chr(0)) - 1)
        End If
    End With

End Function
```

lien :  [OSVERSIONINFOEX Structure](#)

## Comment créer un raccourci sur le Bureau ?

Auteurs : [Romain Puyfoulhoux](#) ,

Ajoutez la référence Windows Script Host Object Model (wshom.ocx) à votre projet.

```
vb
```

```
Dim Wsh As New WshShell
```

```
vb
DesktopPath = Wsh.SpecialFolders("Desktop")
Set Shortcut = Wsh.CreateShortcut(DesktopPath & "\\Test.lnk")
With Shortcut
    .TargetPath = App.EXEName
    .Description = "Mon Programme"
    .WindowStyle = 4
    .Save
End With
```

Avec les anciennes versions de Wshom.ocx, la classe WshShell s'appelle IWshShell\_Class.

## Comment récupérer le numéro de série d'un volume physique ?

**Auteurs : Romain Puyfoulhoux ,**

**Première possibilité, l'api GetVolumeInformation, dont voici la déclaration :**

```
vb
Private Declare Function GetVolumeInformation Lib "kernel32" Alias "GetVolumeInformationA" _
    (ByVal lpRootPathName As String, ByVal lpVolumeNameBuffer As String, ByVal nVolumeNameSize As
    Long, _
    lpVolumeSerialNumber As Long, lpMaximumComponentLength As Long, lpFileSystemFlags As Long, _
    ByVal lpFileSystemNameBuffer As String, ByVal nFileSystemNameSize As Long) As Long
```

Cette fonction, qui permet d'obtenir diverses informations à propos d'un lecteur, s'utilise très simplement. Ces quelques lignes affichent le numéro de série du lecteur C :

```
vb
Dim numero As Long
GetVolumeInformation "c:\", "", 0, numero, 0, 0, "", 0
MsgBox numero
```

**Mais vous pouvez obtenir cette même information avec le FileSystemObject :**

```
vb
Dim fso As FileSystemObject
Set fso = New FileSystemObject
MsgBox fso.Drives("c").SerialNumber
```

vb

```
Set fso = Nothing
```

## Comment mettre mon programme à droite dans la barre des tâches (le systray) ?

Auteurs : [Romain Puyfoulhoux](#) ,

Vous trouverez un exemple pas à pas avec le code source dans le  [Howto de MSDN](#).

## Comment lancer un exécutable et reprendre la main quand il a fini ?

Auteurs : [Abelman](#) ,

Copiez ce code dans un module standard :

vb

```
Public Declare Function CreateProcess Lib "kernel32" Alias "CreateProcessA" _
    (ByVal lpApplicationName As Long, ByVal
    lpCommandLine As String, _
    ByVal lpProcessAttributes As Long, ByVal
    lpThreadAttributes As Long, _
    ByVal bInheritHandles As Long, ByVal dwCreationFlags As Long,
    _
    ByVal lpEnvironment As Long, ByVal lpCurrentDirectory As
    Long, _
    lpStartupInfo As STARTUPINFO, _
    lpProcessInformation As PROCESS_INFORMATION) As Long
Public Declare Function CloseHandle Lib "kernel32" (ByVal hObject As Long) As Long
Public Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As Long)
Public Declare Function GetExitCodeProcess Lib "kernel32" (ByVal hProcess As Long, lpExitCode As
    Long) As Long

'Type pour gérer les lancements de processus
Type STARTUPINFO
    cb As Long
    lpReserved As String
    lpDesktop As String
    lpTitle As String
    dwX As Long
    dwY As Long
    dwXSize As Long
    dwYSize As Long
    dwXCountChars As Long
    dwYCountChars As Long
    dwFillAttribute As Long
    dwFlags As Long
    wShowWindow As Integer
    cbReserved2 As Integer
    lpReserved2 As Long
    hStdInput As Long
    hStdOutput As Long
    hStdError As Long
End Type

'Info sur un processus
Type PROCESS_INFORMATION
    hProcess As Long
    hThread As Long
    dwProcessID As Long
    dwThreadId As Long
```

```
vb
End Type

Public Const NORMAL_PRIORITY_CLASS = &H20&
Public Const STILL_ACTIVE = &H103&

Public Sub AttendreFinProcess(proc As PROCESS_INFORMATION, Optional timeout As Long = 60)

Dim Ret As Long
Dim tms As Single
Dim exitcode As Long

'Attendre la fin de la commande
tms = Timer
Ret = GetExitCodeProcess(proc.hProcess, exitcode)
Do While Ret <> 0 And exitcode = STILL_ACTIVE
    Ret = GetExitCodeProcess(proc.hProcess, exitcode)
    DoEvents
    Sleep 100
    If Timer - tms > timeout Then
        Err.Raise STILL_ACTIVE, "AttendreFinProcess", "Timeout sur l'attente de la fin d'un process"
    End If
Loop
If Ret = 0 Then
    Err.Raise Err.LastDllError, "AttendreFinProcess", "Erreur systeme N° " & Err.LastDllError
End If

End Sub

Public Sub LancerProcess(sExe As String, proc As PROCESS_INFORMATION)

Dim start As STARTUPINFO
Dim Ret As Long

'StartupInfo pour le processus qui lancera la commande
start.cb = Len(start)
'Lancement de la commande
Ret& = CreateProcess(0&, sExe, 0&, 0&, 0&, NORMAL_PRIORITY_CLASS, 0&, 0, start, proc)
If Ret = 0 Then
    Err.Raise Err.LastDllError, "LancerProcess", "Erreur systeme N° " & Err.LastDllError
End If

End Sub
```

Créez une form et placez-y un bouton nommé Command1. Puis copiez ce code dans le module de la form :

```
vb

Private Sub Command1_Click()

Dim proc As PROCESS_INFORMATION

On Error GoTo errortag
proc.hProcess = -1
proc.hThread = -1
Debug.Print "Debut du processus"
Call LancerProcess("notepad", proc)
Call AttendreFinProcess(proc)
Debug.Print "fin du processus"
Call CloseHandle(proc.hProcess)
Call CloseHandle(proc.hThread)

Exit Sub

errortag:
If proc.hProcess <> -1 Then CloseHandle proc.hProcess
```



```
vb
If proc.hThread <> -1 Then CloseHandle proc.hThread
MsgBox Err.Number & " - " & Err.Description

End Sub
```

La fonction `AttendreFinProcess()` attend la fin d'un processus tant que le temps indiqué par le paramètre `timeout` n'est pas écoulé.

Exécutez le projet et cliquez sur le bouton de commande. Le Notepad se lance. Lorsque vous le fermez, la console de débogage affiche "fin du processus".

## Comment obtenir le nom de l'utilisateur ?

Auteurs : **Abelman** , **Romain Puyfoulhoux** , **hpj** ,

Copiez cette déclaration au début d'un module standard :

```
vb

Public Declare Function GetUserName Lib "advapi32.dll" Alias "GetUserNameA" _
    (ByVal lpbuffer As String, nSize As Long) As Long
```

Copiez ensuite cette fonction dans votre module : (pour Win 98)

```
vb

Private Function NomUtilisateur() As String

Dim sUserName As String
Dim iSize As Long

'Un premier appel pour avoir le nombre de caractères nécessaire pour sUserName
GetUserName sUserName, iSize

'On met sUserName à la bonne taille
sUserName = Space(iSize)

'Appel final
GetUserName sUserName, iSize
NomUtilisateur = sUserName

End Function
```

**PS :** On aurait aussi pu déclarer `sUsername` avec une taille assez grande (`dim sUserName as string*32`). Un seul appel de `GetUserName` aurait alors suffi

Pour Windows 2000 et supérieur :

```
vb

Dim NomUtilisateur As String
```

vb

```
NomUtilisateur = Environ("USERNAME")
```

lien : [FAQ](#) Comment déterminer les variables d'environnement du système (Windows 2000 et +) ?

## Comment Vérifier si l'utilisateur courant est Administrateur ?

Auteurs : [forum](#) ,

Utilisez la fonction IsNTAdmin de l'API advpack.dll

vb

```
Private Declare Function IsNTAdmin Lib "advpack.dll" (ByVal dwReserved As Long, ByRef  
lpdwReserved As Long) As Long  
Public Sub VerifieSiAdmin()  
    If CBool(IsNTAdmin(ByVal 0&, ByVal 0&)) = True Then  
        MsgBox "Vous êtes Administrateur de ce poste"  
    Else  
        MsgBox "Vous n'êtes pas Administrateur de ce poste"  
    End If  
End Sub
```

## Comment obtenir ou modifier le contenu du Presse-papiers ?

Auteurs : [Romain Puyfoulhoux](#) ,

Vous pouvez accéder au presse-papiers via l'objet Clipboard.

## Comment permettre à l'utilisateur de sélectionner une imprimante ?

Auteurs : [Romain Puyfoulhoux](#) , [Alexandre Lokchine](#) ,

Les imprimantes installées sont contenues dans la collection Printers. L'imprimante sélectionnée pour l'impression est représentée par l'objet Printer.

Nous allons voir deux méthodes différentes permettant à l'utilisateur de sélectionner une imprimante.

La première est très simple et s'appuie sur le Common Dialog Control. Cochez-le dans les composants du projet et placez-en un sur une form. Il suffit de cette instruction pour afficher la boîte de dialogue standard de propriétés d'impression :

vb

```
CommonDialog1.ShowPrinter
```

La boîte de dialogue une fois validée, les propriétés de l'objet Printer sont automatiquement modifiées afin de correspondre aux paramètres saisis par l'utilisateur.

La deuxième méthode n'utilise aucun composant. Son principe consiste à affecter à l'objet Printer le nom d'une des imprimantes installées. Copiez ce code source dans un module standard.

vb

vb

```
Private Declare Function GetProfileString Lib "kernel32" Alias "GetProfileStringA" _
    (ByVal lpAppName As String, ByVal lpKeyName As String, _
    ByVal lpDefault As String, ByVal nSize As Long) As Long

'Renvoie l'imprimante par défaut

Public Function ImprimanteParDefaut() As String

    Dim def As String, di As Long

    def = String(128, 0)
    di = GetProfileString("WINDOWS", "DEVICE", "", def, 127)
    If di Then ImprimanteParDefaut = Left$(def, di - 1)

End Function

'Sélectionne une imprimante

Public Function SelectionneImprimante(ByVal UneImprimante As String) As Boolean

    Dim impr As Printer, ok As Boolean

    'Il faut chercher dans la boucle l'imprimante correspondante
    'et l'affecter à Printer

    For Each impr In Printers
        If impr.DeviceName = UneImprimante Then
            Set Printer = impr
            ok = True
            Exit For
        End If
    Next

    SelectionneImprimante = ok

End Function
```

La fonction SelectionneImprimante sélectionne l'imprimante dont le nom est passé en paramètre. Maintenant placez un bouton de commande et une ComboBox sur une form et placez le code ci-dessous dans le module de la form.

vb

```
Private Sub Form_Load()

    Dim ImpParDefaut As String, IdxImpParDefaut As Integer
    Dim impr As Printer, i As Long, pos As Integer

    'Récupère le nom de l'imprimante par défaut
    ImpParDefaut = ImprimanteParDefaut()

    'Enlève les informations qui suivent le nom de l'imprimante
    pos = InStr(1, ImpParDefaut, ",", vbTextCompare)
    If pos > 0 Then ImpParDefaut = Left(ImpParDefaut, pos - 1)

    'Ajoute dans la combo box les imprimantes
    IdxImpParDefaut = 0
    i = 0
    For Each impr In Printers
        Combo1.AddItem impr.DeviceName
        'Regarde si c'est celle par défaut et si oui retient son index
        If impr.DeviceName = ImpParDefaut Then IdxImpParDefaut = i
        i = i + 1
    Next

End Sub
```

```

vb
'Sélectionne dans la combo l'imprimante par défaut
If Combo1.ListCount > 0 Then Combo1.ListIndex = IdxImpParDefaut

End Sub

Private Sub Command1_Click()

    MsgBox "Avant : " & Printer.DeviceName
    SelectionneImprimante Combo1.Text
    MsgBox "Après : " & Printer.DeviceName

End Sub
    
```

La ComboBox contient la liste des imprimantes installées. Au chargement de la form, l'imprimante par défaut est sélectionnée dans la combo.

## Comment obtenir la quantité de mémoire du système ?

**Auteurs : Alexandre Lokchine , Romain Puyfoulhoux ,**

**Pour obtenir tous les paramètres relatifs à la mémoire, placez ce code dans un module standard :**

```

vb

Public Type MEMORYSTATUS
    dwLength As Long
    dwMemoryLoad As Long
    dwTotalPhys As Long
    dwAvailPhys As Long
    dwTotalPageFile As Long
    dwAvailPageFile As Long
    dwTotalVirtual As Long
    dwAvailVirtual As Long
End Type

Public Declare Sub GlobalMemoryStatus Lib "kernel32" (lpBuffer As MEMORYSTATUS)
    
```

**Voici comment utiliser la fonction GlobalMemoryStatus :**

```

vb

Dim MS As MEMORYSTATUS
Dim chaine As String

MS.dwLength = Len(MS)
GlobalMemoryStatus MS

chaine = "Pourcentage RAM utilisé: " & Format$(MS.dwMemoryLoad, "###,###,###,###") & " %" & vbCrLf
'on divise toutes les valeurs par 1024 pour les convertir en Kilots-octets
chaine = chaine & "Taille de la mémoire physique totale: " & _
    Format$(MS.dwTotalPhys / 1024, "###,###,###,###") & " Ko" & vbCrLf
chaine = chaine & "Mémoire physique disponible: " & _
    Format$(MS.dwAvailPhys / 1024, "###,###,###,###") & " Ko" & vbCrLf
chaine = chaine & "Mémoire virtuelle totale: " & _
    Format$(MS.dwTotalVirtual / 1024, "###,###,###,###") & " Ko" & vbCrLf
chaine = chaine & "Mémoire virtuelle disponible: " & _
    
```

vb

```
Format$(MS.dwAvailVirtual / 1024, "###,###,###,###") & " KO" & vbCrLf
```

## Comment éjecter le lecteur de cd-rom ?

**Auteurs : Romain Puyfoulhoux ,**

vb

```
Private Declare Function mciSendString Lib "winmm.dll" Alias "mciSendStringA" _
    (ByVal lpstrCommand As String, ByVal lpstrReturnString As
    Any, _
    Integer) As Long
    ByVal wReturnLength As Integer, ByVal hCallback As

Public Sub Ejecte()

mciSendString "Set CDAudio Door Open Wait", 0&, 0, 0

End Sub
```

## Comment afficher/masquer la barre des tâches ?

**Auteurs : ridan ,****Ajouter ces déclarations dans un module :**

vb

```
Private Declare Function SetWindowPos Lib "user32" ( _
    ByVal hwnd As Long, _
    ByVal hWndInsertAfter As Long, _
    ByVal x As Long, _
    ByVal y As Long, _
    ByVal cx As Long, _
    ByVal cy As Long, _
    ByVal wFlags As Long) As Long

Private Declare Function FindWindow Lib "user32" Alias "FindWindowA" ( _
    ByVal lpClassName As String, _
    ByVal lpWindowName As String) As Long

Private Const SWP_HIDEWINDOW As Long = &H80
Private Const SWP_SHOWWINDOW As Long = &H40

Public Sub masquer()
    Dim hwnd As Long
    hwnd = FindWindow("Shell_traywnd", "")
    SetWindowPos hwnd, 0, 0, 0, 0, 0, SWP_HIDEWINDOW
End Sub

Public Sub afficher()
    Dim hwnd As Long
    hwnd = FindWindow("Shell_traywnd", "")
    SetWindowPos hwnd, 0, 0, 0, 0, 0, SWP_SHOWWINDOW
```

```
vb
```

```
End Sub
```

## Comment lancer un élément du panneau de configuration ?

Auteurs : ridan ,

Liste non exhaustive des éléments du panneau de configuration :

- Administrateur de sources de données ODBC : Odbc32.cpl
- Ajout/Suppression de Programmes : Appwiz.cpl
- Assistant Ajout de nouveau matériel : hdwwiz.cpl
- Comptes d'utilisateurs : nusrmgr.cpl
- Connexions Réseau : Ncpa.cpl
- Contrôleurs de jeu : joy.cpl
- Options D'accessibilité : Access.cpl
- Options d'alimentation : Ups.cpl/powercfg.cpl
- Options de modem et Téléphonie : Modem.cpl/Telephon.cpl
- Options Régionales et Linguistiques : Intl.cpl
- Propriétés d'affichage : Desk.cpl
- Propriétés de Date et Heure : Timedate.cpl
- Propriétés de la souris : Main.cpl
- Propriétés Internet : Inetcpl.cpl
- Propriétés Son et Périphériques audio : Mmsys.cpl
- Propriétés système : Sysdm.cpl

```
vb
```

```
Dim NomEl As String  
NomEl = "Appwiz.cpl"  
Shell "rundll32.exe shell32.dll,Control_RunDLL " & NomEl
```

## Comment vider la corbeille ?

Auteurs : ridan ,

```
vb
```

```
Private Declare Function SHEmptyRecycleBin Lib "shell32.dll" Alias "SHEmptyRecycleBinA" ( _  
    ByVal hwnd As Long, _  
    ByVal pszRootPath As String, _  
    ByVal dwFlags As Long) As Long  
  
'Annuler la boîte de dialogue de confirmation  
Private Const SHERB_NOCONFIRMATION = &H1  
'Annuler la boîte de dialogue de progression de suppression  
Private Const SHERB_NOPROGRESSUI = &H2  
'Annuler le son à la fin de la suppression  
Private Const SHERB_NOSOUND = &H4
```

Pour appeler l'API :

```
vb
```

vb

```
SHEmptyRecycleBin 0, vbNullString, SHERB_NOCONFIRMATION
```

## Comment activer/désactiver l'économiseur d'écran ?

Auteurs : ridan ,

Déclarer les constantes comme suit :

vb

```
Private Const SPI_SETSCREENSAVEACTIVE As Long = 17
Private Const VRAI As Long = 1
Private Const FAUX As Long = 0

Private Declare Function SystemParametersInfo Lib "user32.dll" Alias "SystemParametersInfoA" ( _
    ByVal uAction As Long, _
    ByVal uParam As Long, _
    ByRef lpvParam As Any, _
    ByVal fuWinIni As Long) As Long
```

Pour activer l'économiseur :

vb

```
SystemParametersInfo spi_screensaveactive, VRAI, 0, 0
```

Pour désactiver l'économiseur :

vb

```
SystemParametersInfo spi_screensaveactive, FAUX, 0, 0
```

## Comment afficher l'écran de veille où éteindre le moniteur

Auteurs : bbil ,

En utilisant l'API SendMessage.

Pour afficher l'écran de veille

```
Private Declare Function SendMessage Lib "User32" Alias "SendMessageA" _
    (ByVal hWnd As Long, ByVal wParam As Long, _
    ByVal wParam As Long, ByVal lParam As Long) As Long

Const WM_SYSCOMMAND = &H112&
Const SC_SCREENSAVE = &HF140&
Private Sub CdEcranVeille_Click()
    Dim lRes As Long
    lRes = SendMessage(Form1.hWnd, WM_SYSCOMMAND, _
        SC_SCREENSAVE, 0&)
End Sub
```

Pour éteindre l'écran :

```
'Pour les besoin de l'exemple on utilise la fonction Sleep de l'API Kernel32
```

```
Private Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As Long)

Private Declare Function SendMessage Lib "User32" Alias "SendMessageA" _
    (ByVal hWnd As Long, ByVal wParam As Long, _
    ByVal wParam As Long, ByVal lParam As Long) As Long

Const WM_SYSCOMMAND = &H112&
Const SC_MONITORPOWER = &HF170&

Private Sub cdScreenPowerOff_Click()
    Dim lRes As Long
    Sleep (5000) 'Une pause de 5s
    ' Pour éteindre l'écran
    lRes = SendMessage(Form1.hWnd, WM_SYSCOMMAND, _
        SC_MONITORPOWER, 2&)
    Sleep (5000) 'Une pause de 5s
    ' Pour afficher à nouveau l'écran.
    lRes = SendMessage(Form1.hWnd, WM_SYSCOMMAND, _
        SC_MONITORPOWER, -1&)
    Debug.Print "Fin " & Now
End Sub
```

## Comment désactiver le gestionnaire des tâches ou la séquence de touches Ctrl-Alt-Suppr ?

Auteurs : **Khorne** ,

Pour Windows 2000 et supérieur :

Cette astuce permet de désactiver ou d'activer le gestionnaire des tâches à partir du code de votre programme et donc, d'inhiber la séquence de touches Ctrl-Alt-Suppr.

Lorsque le gestionnaire est désactivé, le système renvoie un message d'avertissement (A utiliser avec précaution) :

Désactiver :

vb

```
Set WshShell = CreateObject("WScript.Shell")
WshShell.RegWrite "HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Policies\System\DisableTaskMgr", "1"
Set WshShell = Nothing
```

Activer :

vb

```
Set WshShell = CreateObject("WScript.Shell")
WshShell.RegWrite "HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Policies\System\DisableTaskMgr", ""
Set WshShell = Nothing
```

## Comment récupérer la taille et la position de la barre des tâches ?

Auteurs : **Tofalu** ,

Dans un module :

vb



```

vb
'déclaration du type Rect
Private Type RECT
    Left As Long
    Top As Long
    Right As Long
    Bottom As Long
End Type

Public Type TaskBarInformation
    Left As Long 'Position par rapport au bord gauche de l'écran
    Top As Long 'Position par rapport au haut de l'écran
    Width As Long 'Largeur
    Height As Long 'Hauteur
    Position As Long 'Position : 1 ->En Haut
        ' 2 ->Droite
        ' 3 ->Bas
        ' 4 ->Gauche
End Type

Private Declare Function GetWindowRect Lib "user32" _
    (ByVal hwnd As Long, lpRect As RECT) As Long

Private Declare Function FindWindow Lib "user32" Alias "FindWindowA" ( _
    ByVal lpClassName As String, _
    ByVal lpWindowName As String) As Long

Public Function GetTaskBarInformation() As TaskBarInformation
    Dim rctTemp As RECT
    Dim tskTemp As TaskBarInformation
    Dim intI As Integer, intJ As Integer
    Dim hwnd As Long
    With Screen
        intI = .Width \ (.TwipsPerPixelX * 2)
        intJ = .Height \ (.TwipsPerPixelY * 2)
    End With
    'Récupère le handle de la barre des taches
    hwnd = FindWindow("Shell_traywnd", "")
    'Récupère le rectangle de la barre des taches
    GetWindowRect hwnd, rctTemp
    'Calcule les dimensions
    With tskTemp
        .Left = rctTemp.Left
        .Top = rctTemp.Top
        .Width = rctTemp.Right - rctTemp.Left
        .Height = rctTemp.Bottom - rctTemp.Top
        If .Top > intJ Then
            .Position = 3
        ElseIf .Left > intI Then
            .Position = 2
        ElseIf .Top < intJ Then
            .Position = 1
        Else
            .Position = 4
        End If
    End With
    GetTaskBarInformation = tskTemp
End Function

```

### Exemple d'utilisation :

```

vb
Dim TskTest As TaskBarInformation
TskTest = GetTaskBarInformation

```

```
vb
With TskTest
MsgBox "La position gauche est : " & .Left & vbCrLf & _
      "La position haute est : " & .Top & vbCrLf & _
      "La largeur est de : " & .Width & vbCrLf & _
      "La hauteur est de : " & .Height & vbCrLf & _
      "La position est : " & .Position
End With
```

**NB : Les dimensions sont données en pixels**

## Comment masquer le bouton démarrer avec VB ?

**Auteurs : nabil ,**

**ce code vous permet de cacher et afficher le bouton démarrer, il suffit de placer 2 boutons sur ta form et ce code :**

```
Private Declare Function ShowWindow Lib "user32" _
    (ByVal hwnd As Long, ByVal nCmdShow As Long) As Long

Private Declare Function FindWindow Lib "user32" _
    Alias "FindWindowA" (ByVal lpClassName As String, _
    ByVal lpWindowName As String) As Long

Private Declare Function FindWindowEx Lib "user32" _
    Alias "FindWindowExA" (ByVal hwnd1 As Long, _
    ByVal hwnd2 As Long, _
    ByVal lpsz1 As String, _
    ByVal lpsz2 As String) As Long

Sub StartButton(blnValue As Boolean)
    Dim lngHandle As Long
    Dim lngStartButton As Long

    lngHandle = FindWindow("Shell_TrayWnd", "")
    lngStartButton = FindWindowEx(lngHandle, 0, "Button", vbNullString)

    If blnValue Then
        ShowWindow lngStartButton, 5
    Else
        ShowWindow lngStartButton, 0
    End If
End Sub

Private Sub Command1_Click()
    StartButton (True)
End Sub

Private Sub Command2_Click()
    StartButton (False)
End Sub
```

## Comment récupérer l'exécutable associé à un fichier ?

**Auteurs : SilkyRoad ,**

**Si vous recherchez quel exécutable est associé à votre fichier, alors suivez le code ci-dessous :**

vb/vba

```
Private Declare Function FindExecutableA Lib "shell32.dll" _
    (ByVal lpFile As String, ByVal lpDirectory As String, ByVal lpResult As String) As Long

Public Const MAX_PATH = 256

Public Function FindExecutable(S As String) As String
    'trouve quel executable ouvre le fichier cible
    Dim i As Integer
    Dim S2 As String

    S2 = String(MAX_PATH, 32) & Chr$(0)
    i = FindExecutableA(S & Chr$(0), vbNullString, S2)
    If i > 32 Then
        FindExecutable = Left$(S2, InStr(S2, Chr$(0)) - 1)
    Else
        FindExecutable = ""
    End If
End Function
```

Sommaire > Système > Fichiers

## Comment savoir si un fichier existe ?

Auteurs : Romain Puyfoulhoux ,

vb

```
If Dir("c:\temp\Erreurs.tmp", vbHidden) <> "" Then
    'le fichier existe (vbHidden permet de le retrouver même s'il est caché)
End If
```

## Comment compter les fichiers dans un répertoire ?

Auteurs : random ,

Ci-joint vous trouverez une Fonction à ajouter dans un nouveau module qui vous permettra de compter le nombre de fichiers d'une ou plusieurs extensions voulues dans répertoire donné :

```
Function nbfich(chemin As String, ParamArray termin() As Variant) As Long
    Dim fichier As String
    Dim extension As Variant
    Dim compteur As Long
    For Each extension In termin
        fichier = dir(chemin & "\*." & extension)
        Do Until fichier = ""
            compteur = compteur + 1
            fichier = dir
        Loop
    Next extension
    nbfich = compteur
End Function
```

Voici comment appeler cette Fonction :

```
nbfich("c:\mesimages", "gif", "bmp", "pcx")
```

Vous pouvez mettre une ou plusieurs extensions séparées par des virgules.

## Comment savoir si un fichier est ouvert ?

Auteurs : Cafeine ,

Au moyen d'une fonction qui tente d'ouvrir un fichier en écriture, en cas d'erreur retournée, cela indique que le fichier est déjà ouvert, dans le cas contraire, on considère qu'il est fermé.

```
Function IsFileOpen(ByVal strFic As String) As Boolean
    Dim fic As Integer
    On Error Resume Next

    fic = FreeFile()
    Open strFic For Input Access Read Lock Read Write As fic

    If Err.Number = 0 Then
        IsFileOpen = False
        Close fic
    Else
```

```
        IsFileOpen = True
    End If
End Function
```

lien : [Source](#) Tester si un fichier est déjà ouvert

## Comment copier un fichier ?

Auteurs : Romain Puyfoulhoux ,

vb

```
'Copie le fichier "c:\temp\Erreurs.tmp" en "c:\temp\Erreurs.bak"
FileCopy "c:\temp\Erreurs.tmp", "c:\temp\Erreurs.bak"
```

## Comment copier un fichier actuellement ouvert par une application ?

Auteurs : odan71 ,

La méthode habituellement utilisée pour copier un fichier, FileCopy, échoue si le fichier en cours est actuellement ouvert (violation de partage). Pour contourner ce problème, il faut faire appel à une API du kernel nommée CopyFile. Dans l'exemple ci-dessous, cette méthode est utilisée pour sauvegarder une base Access alors même que celle-ci est ouverte.

vb

```
Private Declare Function CopyFile Lib "kernel32" Alias "CopyFileA" _
    (ByVal lpExistingFileName As String, ByVal lpNewFileName As String, _
    ByVal bFailIfExists As Long) As Long
' bFailIfExists doit être à false pour permettre l'overwriting

Private Sub Form_Load()
    Dim Nouvfich As String
    CommonDialog1.Filter = "Base de données (*.mdb)|*.mdb"
    On Error GoTo erreur
    CommonDialog1.ShowSave
    Nouvfich = CommonDialog1.FileName
    CopyFile App.Path & "\tests.mdb", Nouvfich, False
erreur:
    If Err = 32755 Then
        Exit Sub
    End If
End Sub
```

## Comment renommer un fichier ou un répertoire ?

Auteurs : Romain Puyfoulhoux ,

vb

```
'Renomme "c:\temp\Erreurs.tmp" en "c:\temp\Erreurs.bak"
Name "c:\temp\Erreurs.tmp" As "c:\temp\Erreurs.bak"
'Renomme le répertoire "c:\temp" en "c:\var"
```

vb

```
Name "c:\temp" As "c:\var"
```

## Comment détruire un fichier ?

Auteurs : Romain Puyfoulhoux ,

vb

```
Kill "c:\Erreurs.tmp"
```

Le fichier doit exister sinon une erreur d'exécution a lieu.

## Comment envoyer un fichier à la corbeille ?

Auteurs : Romain Puyfoulhoux ,

En utilisant les API Windows. Copiez ce code source dans un module standard :

vb

```
Private Type SHFILEOPSTRUCT
    hwnd As Long
    wFunc As Long
    pFrom As String
    pTo As String
    fFlags As Long
    fAnyOperationsAborted As Long
    hNameMappings As Long
    lpzProgressTitle As String
End Type

Private Const FO_DELETE As Long = &H3
Private Const FOF_ALLOWUNDO As Long = &H40

Private Declare Function SHFileOperation Lib "Shell32.dll" Alias "SHFileOperationA" _
    (lpFileOp As SHFILEOPSTRUCT) As Long

Public Function DansCorbeille(fichier As String, handle As Long) As Boolean

    Dim DelFileOp As SHFILEOPSTRUCT
    Dim Result As Long

    With DelFileOp
        .hwnd = handle
        .wFunc = FO_DELETE
        .pFrom = fichier & vbNullChar & vbNullChar
        .fFlags = FOF_ALLOWUNDO
    End With

    Result = SHFileOperation(DelFileOp)
    DansCorbeille = (Result = 0) And (DelFileOp.fAnyOperationsAborted = 0)

End Function
```

La fonction `DansCorbeille` renvoie `True` si l'envoi du fichier dans la corbeille a été effectué. Ses paramètres sont le chemin complet du fichier et le handle de la fenêtre utilisé pour afficher les éventuelles boîtes de dialogue d'avertissement ou de demande de confirmation. Ce deuxième paramètre peut être une valeur nulle. Voici un exemple d'utilisation de cette fonction :

vb

```
If DansCorbeille("C:\lettre.doc", Me.hwnd) Then
    MsgBox "Le fichier a été déplacé dans la corbeille"
Else
    MsgBox "Le fichier n'a pas pu être déplacé dans la corbeille"
End If
```

## Comment connaître les dates de création, de dernière modification et de dernier accès d'un fichier ?

Auteurs : Romain Puyfoulhoux ,

En passant soit par les API, soit par le FileSystemObject. En natif, VB ne donne accès qu'à la date de dernière modification, via la fonction FileDateTime(). Si vous choisissez de passer par les API, voici les déclarations nécessaires :

vb

```
Private Const MAX_PATH = 260
Private Type FILETIME
    dwLowDateTime As Long
    dwHighDateTime As Long
End Type
Private Type SYSTEMTIME
    wYear As Integer
    wMonth As Integer
    wDayOfWeek As Integer
    wDay As Integer
    wHour As Integer
    wMinute As Integer
    wSecond As Integer
    wMilliseconds As Integer
End Type
Private Type WIN32_FIND_DATA
    dwFileAttributes As Long
    ftCreationTime As FILETIME
    ftLastAccessTime As FILETIME
    ftLastWriteTime As FILETIME
    nFileSizeHigh As Long
    nFileSizeLow As Long
    dwReserved0 As Long
    dwReserved1 As Long
    cFileName As String * MAX_PATH
    cAlternate As String * 14
End Type
Private Const INVALID_HANDLE_VALUE = -1
Private Declare Function FindFirstFile Lib "kernel32" Alias "FindFirstFileA" _
    (ByVal lpFileName As String, _
    lpFindFileData As WIN32_FIND_DATA) As Long
Private Declare Function FindClose Lib "kernel32" (ByVal hFindFile As Long) As Long
Private Declare Function FileTimeToSystemTime Lib "kernel32" _
    (lpFileTime As FILETIME, _
    lpSystemTime As SYSTEMTIME) As Long
Private Declare Function FileTimeToLocalFileTime Lib "kernel32" _
    (lpFileTime As FILETIME, _
    lpLocalFileTime As FILETIME) As Long
```

Les dates d'un fichier sont récupérées par la fonction FindFirstFile qui attend en paramètres le nom du fichier et une structure WIN32\_FIND\_DATA qui reçoit les informations obtenues. Les dates de création, de dernière modification et de dernier accès sont stockées respectivement dans les champs ftCreationTime, ftLastWriteTime et ftLastAccessTime, tous de type FILETIME. Pour avoir des dates sous une forme exploitable, quelques conversions sont nécessaires. Notre fonction FileTimeToDate() convertit une date de type FILETIME en type Date.

vb

```
Private Function FileTimeToDate(ft As FILETIME) As Date

Dim datelocale As FILETIME, datesys As SYSTEMTIME

If FileTimeToLocalFileTime(ft, datelocale) = 0 Then Exit Function
If FileTimeToSystemTime(datelocale, datesys) = 0 Then Exit Function
FileTimeToDate = CDate(datesys.wDay & " " & datesys.wMonth & " " & datesys.wYear & " " & _
    datesys.wHour & ":" & datesys.wMinute & ":" & datesys.wSecond)

End Function
```

La partie principale du code est assez simple :

vb

```
Dim findData As WIN32_FIND_DATA, hFind As Long

hFind = FindFirstFile("c:\autoexec.bat", findData)
If hFind = INVALID_HANDLE_VALUE Then Exit Sub
FindClose hFind

MsgBox "Crée le : " & FileTimeToDate(findData.ftCreationTime)
MsgBox "Modifié le : " & FileTimeToDate(findData.ftLastWriteTime)
MsgBox "Accédé le : " & FileTimeToDate(findData.ftLastAccessTime)
```

Pour terminer, voici la version avec le FileSystemObject :

vb

```
Dim fso As FileSystemObject, f As File

Set fso = New FileSystemObject

On Error GoTo fin
Set f = fso.GetFile("c:\autoexec.bat")
MsgBox "Crée le : " & f.DateCreated
MsgBox "Modifié le : " & f.DateLastModified
MsgBox "Accédé le : " & f.DateLastAccessed
Set f = Nothing

fin:
Set fso = Nothing
```

lien : [FAQ](#) Quelle référence dois-je ajouter à mon projet pour pouvoir utiliser le FileSystemObject ?

## Comment obtenir le numéro de version d'un fichier (si disponible) ?

Auteurs : [ThierryAIM](#) ,

Cette fonction reçoit le chemin complet d'un fichier en paramètre et renvoie le numéro de version du fichier, s'il existe, sinon renvoie une chaîne vide :  
(concerne essentiellement les fichiers .dll, .ocx, .exe et autres fichiers du système)

Collez ce code dans un module standard :

vb

```
'-- Déclarations de structure et des fonctions de l'API Windows
```



vb

```

Private Type VS_FIXEDFILEINFO
    dwSignature As Long
    dwStrucVersionl As Integer ' e.g. = &h0000 = 0
    dwStrucVersionh As Integer ' e.g. = &h0042 = .42
    dwFileVersionMSl As Integer ' e.g. = &h0003 = 3
    dwFileVersionMSh As Integer ' e.g. = &h0075 = .75
    dwFileVersionLSl As Integer ' e.g. = &h0000 = 0
    dwFileVersionLSh As Integer ' e.g. = &h0031 = .31
    dwProductVersionMSl As Integer ' e.g. = &h0003 = 3
    dwProductVersionMSh As Integer ' e.g. = &h0010 = .1
    dwProductVersionLSl As Integer ' e.g. = &h0000 = 0
    dwProductVersionLSh As Integer ' e.g. = &h0031 = .31
    dwFileFlagsMask As Long ' = &h3F for version "0.42"
    dwFileFlags As Long ' e.g. VFF_DEBUG Or VFF_PRERELEASE
    dwFileOS As Long ' e.g. VOS_DOS_WINDOWS16
    dwFileType As Long ' e.g. VFT_DRIVER
    dwFileSubtype As Long ' e.g. VFT2_DRV_KEYBOARD
    dwFileDateMS As Long ' e.g. 0
    dwFileDateLS As Long ' e.g. 0
End Type

Private Declare Function GetFileVersionInfo Lib "Version.dll" Alias "GetFileVersionInfoA" _
    (ByVal lptstrFilename As String, ByVal dwhandle As Long, ByVal dwlen As Long, lpData As Any) As Long

Private Declare Function GetFileVersionInfoSize Lib "Version.dll" Alias "GetFileVersionInfoSizeA" _
    (ByVal lptstrFilename As String, lpdwHandle As Long) As Long

'-- Renvoie le numero de version d'un fichier au format String : x.xx.xxxx.xxxx
Public Function GetFileVerInfo(sFullPath As String) As String
    Dim rc As Long, lDummy As Long, sBuffer() As Byte
    Dim lBufferLen As Long, lVerPointer As Long, udtVerBuffer As VS_FIXEDFILEINFO
    Dim lVerbufferLen As Long

    '*** Obtient la taille du buffer ***
    lBufferLen = GetFileVersionInfoSize(sFullPath, lDummy)
    If lBufferLen < 1 Then
        GetFileVerInfo = "" ' No Version Info available!
        Exit Function
    End If

    '**** Stocke les informations dans la structure udtVerBuffer ****
    ReDim sBuffer(lBufferLen)
    rc = GetFileVersionInfo(sFullPath, 0, lBufferLen, sBuffer(0))
    rc = VerQueryValue(sBuffer(0), "\", lVerPointer, lVerbufferLen)
    MoveMemory udtVerBuffer, lVerPointer, Len(udtVerBuffer)

    '**** Renvoie les information de numéro de version ****
    GetFileVerInfo = Format$(udtVerBuffer.dwFileVersionMSh) & "." & _
        Format$(udtVerBuffer.dwFileVersionMSl) & "." & _
        Format$(udtVerBuffer.dwFileVersionLSh) & "." & _
        Format$(udtVerBuffer.dwFileVersionLSl)
End Function

```

### Exemple :

vb

vb

```
Msgbox GetFileVerInfo("C:\windows\Explorer.exe")
```

## Comment récupérer la taille d'un fichier ?

Auteurs : **Abelman** ,

vb

```
Debug.Print FileLen("nomdufichier") 'Affiche en octets la taille du fichier
```

lien : [FAQ](#) Comment convertir une taille de fichier donnée en octets en une unité adaptée ?

## Comment convertir une taille de fichier donnée en octets en une unité adaptée ?

Auteurs : **Optitech** ,

La commande FileLen vue ci-dessus, retourne la taille du fichier en octets, mais un fichier de 5368709120 octets ne nous dit rien.

On sait que :

1Ko = 1024o

1Mo = 1024Ko

1Go = 1024Mo

etc...

Pour convertir en une unité compréhensible, nous allons diviser la taille par 1024, si celle-ci est supérieure à 1024. Si le nombre obtenu est toujours supérieur à 1024, on redivise par 1024, et ainsi de suite...

La fonction suivante permet cette conversion jusqu'aux Yo (yotta-octet = 2<sup>80</sup> octets).

vb

```
Function Unite(ByVal Taille As Double) As String

    Dim TabUnite 'On déclare une variable qui va être un tableau
    Dim i As Integer 'Un compteur

    TabUnite = Array("o", "Ko", "Mo", "Go", "To", "Po", "Eo", "Zo", "Yo") 'Le tbaleau de unités
    i = 0 'Initialisation du compteur

    Do While Taille >= 1024 And i < 8 'Début de la boucle
        Taille = Taille / 1024 'On divise
        i = i + 1 'on rajoute 1 au compteur
    Loop

    'On retourne la taille convertie avec l'unité dans une string
    Unite = Round(Taille, 2) & " " & TabUnite(i)

End Function
```

Exemple :

vb

vb

```
MsgBox Unite(FileLen("chemindufichier"))
```

lien : [FAQ](#) Comment récupérer la taille d'un fichier ?

## Comment copier un répertoire ?

Auteurs : [Romain Puyfoulhoux](#) ,

Ce source copie le contenu du répertoire c:\sources dans le répertoire c:\oldsources. Passer la valeur True en troisième paramètre de CopyFolder indique que les fichiers existants devront être écrasés.

vb

```
Dim fso As FileSystemObject
Set fso = New FileSystemObject
fso.CopyFolder "c:\sources", "c:\oldsources", True
Set fso = Nothing
```

lien : [FAQ](#) Quelle référence dois-je ajouter à mon projet pour pouvoir utiliser le FileSystemObject ?

## Comment supprimer un répertoire ?

Auteurs : [Romain Puyfoulhoux](#) ,

Avec Rmdir, mais vous ne pouvez l'utiliser que pour supprimer des répertoires vides.

vb

```
Rmdir "c:\temp"
```

Pour supprimer un répertoire qui contient fichiers ou répertoires, utilisez le FileSystemObject, qui est disponible seulement si vous avez inclus la librairie Microsoft Scripting Runtime dans les références de votre projet.

vb

```
Dim fso as FileSystemObject
Set fso = New FileSystemObject
fso.DeleteFolder "c:\temp", True
```

La valeur True passée au deuxième paramètre permet de supprimer le répertoire dans le cas où il aurait l'attribut lecture seule. Ce paramètre est optionnel, et a la valeur False par défaut.

lien : [FAQ](#) Quelle référence dois-je ajouter à mon projet pour pouvoir utiliser le FileSystemObject ?

## Comment obtenir le contenu d'un répertoire ?

Auteurs : [Romain Puyfoulhoux](#) ,

vb

```
Dim rep As String
'obtient le premier fichier ou répertoire qui est dans "c:\"
rep = Dir("c:\*.*", vbDirectory)
'boucle tant que le répertoire n'a pas été entièrement parcouru
Do While (rep <> "")
```

```
vb
'teste si c'est un fichier ou un répertoire
If (GetAttr("c:\" & rep) And vbDirectory) = vbDirectory Then
    MsgBox "Répertoire " & rep
Else
    MsgBox "Fichier " & rep
End If
'passe à l'élément suivant
rep = Dir
Loop
```

## Comment scanner un répertoire et tous ses sous-répertoires ?

Auteurs : e-steel ,

**Nécessite d'activer la référence *Microsoft Scripting Runtime* (sccrun.dll)  
Il y a possibilité d'agir sur chaque fichier listé !**

```
vb

Private Sub Command1_Click()

Dim fso As FileSystemObject, dossier As Folder, sousdossier As Folder, fichier As File

Set fso = New FileSystemObject
Set dossier = fso.GetFolder("c:\essai")
scan dossier

End Sub

Public Sub scan(ByVal dossier As Folder)

For Each fichier In dossier.Files
    Debug.Print fichier
Next

For Each sousdossier In dossier.SubFolders
    Debug.Print sousdossier
    scan sousdossier
Next

End Sub
```

## Comment obtenir la taille d'un répertoire ?

Auteurs : ThierryAIM ,

**Voici une méthode simple utilisant le FileSystemObject. Placez ce code dans un module :**

```
vb

Public Enum UniteMemoire
    octets = 1
    kiloOctets = 2
    megaOctets = 3
End Enum

Public Function TailleRepertoire(f As Folder, Optional unite As UniteMemoire = 1) As Long

Dim s As Long
```

vb

```
s = f.Size
Select Case unite
  Case 2:
    TailleRepertoire = Int(s / 1024)
  Case 3:
    TailleRepertoire = Int(s / 1048576)
  Case Else:
    TailleRepertoire = s
End Select
End Function
```

Un exemple d'utilisation :

vb

```
Private Sub Test()

Dim fs As FileSystemObject, f As Folder, strTaille As String

Set fs = New FileSystemObject
Set f = fs.GetFolder("c:\windows")

strTaille = Format(TailleRepertoire(f), "##,##0 octets") & vbCrLf & _
            Format(TailleRepertoire(f, kiloOctets), "##,##0 Ko") & vbCrLf & _
            Format(TailleRepertoire(f, megaOctets), "##,##0 Mo")
MsgBox strTaille

End Sub
```

lien : [FAQ](#) Quelle référence dois-je ajouter à mon projet pour pouvoir utiliser le FileSystemObject ?

## Comment ouvrir une fenêtre de sélection de répertoire ?

Auteurs : [Romain Puyfoulhoux](#) ,

Pour cela vous devez ajouter ces déclarations au début de votre module :

vb

```
Private Const BIF_RETURNONLYFSDIRS = 1
Private Const BIF_DONTGOBELOWDOMAIN = 2

Private Declare Function SHBrowseForFolder Lib "shell32" (lpbi As BrowseInfo) As Long
Private Declare Function SHGetPathFromIDList Lib "shell32" (ByVal pidList As Long, _
    ByVal lpBuffer As String) As Long
Private Declare Function lstrcat Lib "kernel32" Alias "lstrcatA" (ByVal lpString1 As String, _
    ByVal lpString2 As String) As Long

Private Type BrowseInfo
  hWndOwner As Long
  pIDLRoot As Long
  pszDisplayName As Long
  lpszTitle As Long
  ulFlags As Long
  lpfnCallback As Long
  lParam As Long
  iImage As Long
End Type
```

La fonction suivante ouvre la fenêtre de sélection de répertoire standard de Windows et renvoie le chemin du répertoire sélectionné. Les paramètres attendus sont le titre à afficher et l'identifiant de la fenêtre parente.

```

vb
Public Function SelectFolder(Titre As String, Handle As Long) As String

Dim lpIDList As Long
Dim strBuffer As String
Dim strTitre As String
Dim tBrowseInfo As BrowseInfo

strTitre = Titre
With tBrowseInfo
.hwndOwner = Handle
.lpszTitle = lstrcat(strTitre, "")
.ulFlags = BIF_RETURNONLYFSDIRS + BIF_DONTGOBELOWDOMAIN
End With

lpIDList = SHBrowseForFolder(tBrowseInfo)

If (lpIDList) Then
strBuffer = String(260, vbNullChar)
SHGetPathFromIDList lpIDList, strBuffer
SelectFolder = Left(strBuffer, InStr(strBuffer, vbNullChar) - 1)
End If

End Function

```

Cette ligne fait appel à la fonction écrite ci-dessus pour ouvrir la fenêtre de sélection de répertoire et afficher le répertoire sélectionné :

```

vb
MsgBox SelectFolder("Sélectionnez un répertoire :", Me.hWnd)

```

## Comment modifier la fenêtre de sélection de répertoire ?

**Auteurs : bbil , ThierryAIM ,**

**Vous trouverez dans la [faq](#) : Comment ouvrir une fenêtre de sélection de répertoire ?**  
**Une solution utilisant la fonction SHBrowseForFolder de l'Api shell32,**  
**pour afficher la fenêtre de sélection de répertoire.**  
**Cette fonction prends en paramètre de type BrowseInfo, la modification de la fenêtre de sélection de répertoire est obtenu grâce à la propriété uFlag de ce paramètre .**  
**Et donc une modification de la ligne de code :**

```

.ulFlags = BIF_RETURNONLYFSDIRS + BIF_DONTGOBELOWDOMAIN

```

Nom	Valeur	Conséquence
<b>BIF_RETURNONLYFSDIRS</b>	<b>&amp;H0001</b>	<b>Autorise seulement la sélection d'éléments du</b>

		système de fichier ( pas d'imprimante...)
<b>BIF_DONTGOBELOWDOMAIN</b>	<b>&amp;H0002</b>	Limite la sélection dans réseau au nom de domaines sans ouverture possible
<b>BIF_STATUSTEXT</b>	<b>&amp;H0004</b>	Rajoute une zone état dont le texte peu ensuite être modifié par des appels à SendMessage
<b>BIF_EDITBOX</b>	<b>&amp;H0010</b>	Rajout d'une zone d'édition sur la boîte de dialogue parcourir, permettant la saisie d'un répertoire (existant)
<b>BIF_VALIDATE</b>	<b>&amp;H0020</b>	Utilisé conjointement avec <b>BIF_EDITBOX</b> , appel de la fonction callback : <b>BrowseCallbackProc</b> , avec le message : <b>BFFM_VALIDATEFAILED</b> , si répertoire saisie invalide.
<b>BIF_NEWDIALOGSTYLE</b>	<b>&amp;H0040</b>	Boîte de dialogue nouveau style, support glisser déposer, taille modifiable, bouton créer nouveau dossier, menu contextuel.
<b>BIF_BROWSEINCLUDEURL</b>	<b>&amp;H0080</b>	Permet la sélection d'un fichier par son URL, nécessite les flags : <b>BIF_USENEWUI</b> et <b>BIF_BROWSEINCLUDEFILES</b> .
<b>BIF_USENEWUI</b>	<b>&amp;H0050</b>	Nouvelle interface y compris la zone d'édition, équivalent à : <b>BIF_EDITBOX</b> OR <b>BIF_NEWDIALOGSTYLE</b> .
<b>BIF_UAHINT</b>	<b>&amp;H0100</b>	Lorsque combiné avec <b>BIF_NEWDIALOGSTYLE</b> , ajoute un texte d'aide (" <i>Pour afficher... cliquez sur +</i> "), en lieu et place de la zone

		d'édition, si BIF_EDITBOX et actif celui-ci l'emporte.
<b>BIF_NONEWFOLDERBUTTON</b>	<b>&amp;H0200</b>	Supprime l'affichage du bouton nouveau dossier.
<b>BIF_NOTRANSLATETARGETS</b>	<b>&amp;H0400</b>	Sur sélection d'un raccourci, renvoi le raccourci lui même plutôt que sa cible.
<b>BIF_BROWSEFORCOMPUTER</b>	<b>&amp;H1000</b>	Autorise seulement le choix d'un ordinateur.
<b>BIF_BROWSEFORPRINTER</b>	<b>&amp;H2000</b>	Autorise seulement le choix d'une imprimante.
<b>BIF_BROWSEINCLUDEFILES</b>	<b>&amp;H4000</b>	Permet la sélection des fichiers.
<b>BIF_SHAREABLE</b>	<b>&amp;H8000</b>	Utilisé conjointement avec BIF_NEWDIALOGSTYLE, affiche la liste des ressources partagées par les ordinateurs distants.

### implémentation

Le code vu précédemment [FAQ Comment ouvrir une fenêtre de sélection de répertoire ?](#) peut-être adapté : rajouter tout d'abords la déclaration des constantes utiles :

```
Public Const BIF_RETURNONLYFSDIRS = &H0001
Public Const BIF_DONTGOBELOWDOMAIN = &H0002
Public Const BIF_STATUSTEXT = &H0004
Public Const BIF_EDITBOX = &H0010
Public Const BIF_NEWDIALOGSTYLE = &H0040
Public Const BIF_NONEWFOLDERBUTTON = &H0200
(...)
```

### Modifier :

Dans l'entête de la fonction :

```
Public Function SelectFolder(Titre As String, Handle As Long, Optional ByVal uFlags As Long =
    BIF_RETURNONLYFSDIRS + BIF_DONTGOBELOWDOMAIN) As String
```

Dans le corps de la fonction :

```
.ulFlags = uFlags
```

et en appelant la fonction SelectFolder :

Avec affichage du bouton de création de dossier :

```
MsgBox SelectFolder ("Titre", Me.hwnd, BIF_NEWDIALOGSTYLE)
```

Sans affichage du bouton de création de dossier :

```
MsgBox SelectFolder ("Titre", Me.hwnd, BIF_NEWDIALOGSTYLE+ BIF_NONEWFOLDERBUTTON)
```



## MSDN BROWSEINFO Structure

### Quelle référence dois-je ajouter à mon projet pour pouvoir utiliser le FileSystemObject ?

Auteurs : [Romain Puyfoulhoux](#) ,

Dans le Menu Projet >> Références >> Ajoutez la librairie Microsoft Scripting Runtime. Le fichier correspondant se nomme scrrun.dll

### Comment récupérer le répertoire d'un fichier à partir de son chemin complet ?

Auteurs : [ThierryAIM](#) ,

Cette fonction reçoit le chemin complet d'un fichier en paramètre et renvoie le chemin du répertoire :

```
vb

Public Function ExtractFilePath(ByVal sFullPath As String) As String

If Right(sFullPath, 1) = "\" Then
    ExtractFilePath = sFullPath
Else
    ExtractFilePath = Left(sFullPath, InStrRev(sFullPath, "\"))
End If

End Function
```

lien : [FAQ](#) Comment récupérer le nom d'un fichier à partir d'un chemin complet ?

lien : [FAQ](#) Comment récupérer l'extension d'un fichier à partir d'un chemin complet ?

### Afficher la boîte de dialogue ouvrir afin de récupérer le nom et le chemin du fichier sélectionné

Auteurs : [shwin](#) ,

Cette fonction propose plusieurs arguments utiles pour personnaliser votre boîte de dialogue, ils sont expliqués dans le code.

Code à placer dans un module :

```
'Déclaration de l'API

Private Declare Function GetOpenFileName Lib "comdlg32.dll" Alias _
    "GetOpenFileNameA" (pOpenfilename As OPENFILENAME) As Long

'Structure du fichier
Private Type OPENFILENAME
    lStructSize As Long
    hwndOwner As Long
    hInstance As Long
    lpstrFilter As String
    lpstrCustomFilter As String
    nMaxCustFilter As Long
    nFilterIndex As Long
    lpstrFile As String
    nMaxFile As Long
    lpstrFileTitle As String
    nMaxFileTitle As Long
    lpstrInitialDir As String
```

```

lpstrTitle As String
flags As Long
nFileOffset As Integer
nFileExtension As Integer
lpstrDefExt As String
lCustData As Long
lpfnHook As Long
lpTemplateName As String
End Type

'Constantes
Private Const OFN_READONLY = &H1
Private Const OFN_OVERWRITEPROMPT = &H2
Private Const OFN_HIDEREADONLY = &H4
Private Const OFN_NOCHANGEDIR = &H8
Private Const OFN_SHOWHELP = &H10
Private Const OFN_ENABLEHOOK = &H20
Private Const OFN_ENABLETEMPLATE = &H40
Private Const OFN_ENABLETEMPLATEHANDLE = &H80
Private Const OFN_NOVALIDATE = &H100
Private Const OFN_ALLOWMULTISELECT = &H200
Private Const OFN_EXTENSIONDIFFERENT = &H400
Private Const OFN_PATHMUSTEXIST = &H800
Private Const OFN_FILEMUSTEXIST = &H1000
Private Const OFN_CREATEPROMPT = &H2000
Private Const OFN_SHAREAWARE = &H4000
Private Const OFN_NOREADONLYRETURN = &H8000
Private Const OFN_NOTESTFILECREATE = &H10000

Private Const OFN_SHAREFALLTHROUGH = 2
Private Const OFN_SHARENOWARN = 1
Private Const OFN_SHAREWARN = 0

Public Function OuvrirUnFichier(Handle As Long, _
                               Titre As String, _
                               TypeRetour As Byte, _
                               Optional TitreFiltre As String, _
                               Optional TypeFichier As String, _
                               Optional RepParDefaut As String) As String
'OuvrirUnFichier est la fonction a utiliser dans votre formulaire pour ouvrir _
'la boîte de dialogue de sélection d'un fichier.
'Explication des paramètres
'Handle = le handle de la fenêtre (Me.Hwnd)
'Titre = Titre de la boîte de dialogue
'TypeRetour (Définit la valeur, de type String, renvoyée par la fonction)
'1 = Chemin complet + Nom du fichier
'2 = Nom fichier seulement
'TitreFiltre = Titre du filtre
'Exemple: Fichier Access
'N'utilisez pas cet argument si vous ne voulez spécifier aucun filtre
'TypeFichier = Extension du fichier (Sans le .)
'Exemple: MDB
'N'utilisez pas cet argument si vous ne voulez spécifier aucun filtre
'RepParDefaut = Répertoire d'ouverture par défaut
'Exemple: C:\windows\system32

'Si vous laissé l'argument vide, par défaut il se place dans le répertoire de votre application

Dim StructFile As OPENFILENAME
Dim sFiltre As String

'Construction du filtre en fonction des arguments spécifiés
If Len(TitreFiltre) > 0 And Len(TypeFichier) > 0 Then
sFiltre = TitreFiltre & " (" & TypeFichier & ")" & Chr$(0) & ".*" & TypeFichier & Chr$(0)
End If
sFiltre = sFiltre & "Tous (*.*)" & Chr$(0) & ".*" & Chr$(0)

```

```
'Configuration de la boîte de dialogue
With StructFile
  .lStructSize = Len(StructFile) 'Initialisation de la grosseur de la structure
  .hwndOwner = Handle 'Identification du handle de la fenêtre
  .lpstrFilter = sFiltre 'Application du filtre
  .lpstrFile = String$(254, vbNullChar) 'Initialisation du fichier '0' x 254
  .nMaxFile = 254 'Taille maximale du fichier
  .lpstrFileTitle = String$(254, vbNullChar) 'Initialisation du nom du fichier '0' x 254
  .nMaxFileTitle = 254 'Taille maximale du nom du fichier
  .lpstrTitle = Titre 'Titre de la boîte de dialogue
  .flags = OFN_HIDEREADONLY 'Option de la boîte de dialogue
  If ((IsNull(RepParDefaut)) Or (RepParDefaut = "")) Then
    .lpstrInitialDir = App.path
  Else: .lpstrInitialDir = RepParDefaut
  End If
End With

If (GetOpenFileName(StructFile)) Then 'Si un fichier est sélectionné
  Select Case TypeRetour
    Case 1: OuvrirUnFichier = Trim$(Left(StructFile.lpstrFile, InStr(1,
StructFile.lpstrFile, vbNullChar)-1))
    Case 2: OuvrirUnFichier = Trim$(Left(StructFile.lpstrFileTitle, InStr(1,
StructFile.lpstrFileTitle, vbNullChar)-1))
  End Select
  End If
End Function
```

Exemple pour appeler la fonction depuis le code d'un formulaire :

```
MsgBox OuvrirUnFichier(Me.Hwnd, "Parcourir", 1, "Fichier Word", "doc")
```

lien : Afficher la boîte de dialogue Enregistrer sous afin de récupérer le nom et le chemin du fichier sélectionné

lien : Utilise le contrôle Common Dialog pour récupérer le chemin d'un fichier

lien : [FAQ](#) Comment avec l'API GetOpenFileNameA ouvrir plusieurs fichiers à la fois ?

## Comment avec l'API GetOpenFileNameA ouvrir plusieurs fichiers à la fois ?

Auteurs : shwin ,

Il suffit de mettre l'attribut flags à la valeur suivante :

```
openfile.flags = &H200 'Multiselect
```

Ensuite, la variable fichier est un string qui va contenir les noms.

```
C:\autoexec.bat nouveau.txt
```

A vous de travailler cette chaîne pour obtenir le résultat souhaité.

lien : [FAQ](#) Afficher la boîte de dialogue ouvrir afin de récupérer le nom et le chemin du fichier sélectionné

Afficher la boîte de dialogue Enregistrer sous afin de récupérer le nom et le chemin du fichier sélectionné

Auteurs : shwin ,

**Note :** Les arguments de cette fonction sont expliqués dans le code.

**Code à placer dans un module :**

```
'Déclaration de l API
Private Declare Function GetSaveFileName Lib "comdlg32.dll" _
    Alias "GetSaveFileNameA" (pOpenfilename As OPENFILENAME) _
    As Long

'Structure du fichier
Private Type OPENFILENAME
    lStructSize As Long
    hWndOwner As Long
    hInstance As Long
    lpstrFilter As String
    lpstrCustomFilter As String
    nMaxCustFilter As Long
    nFilterIndex As Long
    lpstrFile As String
    nMaxFile As Long
    lpstrFileName As String
    nMaxFileName As Long
    lpstrInitialDir As String
    lpstrTitle As String
    Flags As Long
    nFileOffset As Integer
    nFileExtension As Integer
    lpstrDefExt As String
    lCustData As Long
    lpfnHook As Long
    lpTemplateName As String
End Type

Function EnregistrerUnFichier(Handle As Long, Titre As String, _
    NomFichier As String, Chemin As String) As String

'EnregistrerUnFichier est la fonction a utiliser dans votre formulaire pour ouvrir _
la boîte de dialogue d'enregistrement d'un fichier.
'Explication des paramètres
'Handle = le handle de la fenêtre (Me.Hwnd)
'Titre = Titre de la boîte de dialogue
'NomFichier = Nom par défaut du fichier à enregistrer
'Chemin = Chemin par défaut du fichier à enregistrer

Dim structSave As OPENFILENAME

With structSave
    .lStructSize = Len(structSave)
    .hWndOwner = Handle
    .nMaxFile = 255
    .lpstrFile = NomFichier & String$(255 - Len(NomFichier), 0)
    .lpstrInitialDir = Chemin
    .lpstrFilter = "Tous (*.*)" & Chr$(0) & " *.*" & Chr$(0) 'Définition du filtre (aucun)
    .Flags = &H4 'Option de la boîte de dialogue
End With
```

```

If (GetSaveFileName(structSave)) Then
    EnregistrerUnFichier = Mid$(structSave.lpstrFile, 1, InStr(1,
    structSave.lpstrFile, vbNullChar) - 1)
End If

End Function

```

Exemple pour appeler la fonction depuis le code d'un formulaire :

```
MsgBox EnregistrerUnFichier(Me.hwnd, "Enregistrer sous", "Test.doc", "C:\")
```

lien : Afficher la boîte de dialogue ouvrir afin de récupérer le nom et le chemin du fichier sélectionné

lien : Utilise le contrôle Common Dialog pour récupérer le chemin d'un fichier

## Comment obtenir le chemin relatif d'un fichier ?

Auteurs : Cafeine , Tofalu ,

Pour obtenir le chemin relatif d'un fichier par rapport à un répertoire, vous pouvez utiliser la fonction suivante :

```

Function GetRelativePath(ByVal strPath As String, Optional ByVal strPathCurrent As String)

Dim tmpCurr() As String
Dim tmpP() As String
Dim i As Integer
Dim iIndex As Integer

' par défaut on considère que c'est relatif par rapport au chemin courant de la base
' pour VB6, on utilise le chemin de l'application :

If strPathCurrent = "" Then strPathCurrent = App.Path 'CurrentProject.Path
If Right(strPathCurrent, 1) = "\" Then
    strPathCurrent = Left(strPathCurrent, Len(strPathCurrent) - 1)

' on passe tout en minuscule pour éviter les erreurs de comparaison minuscule et majuscule
strPath = LCase(strPath)
strPathCurrent = LCase(strPathCurrent)

If Left(strPath, 1) = Left(strPathCurrent, 1) Then
    ' on recherche la partie commune aux deux chemins

    tmpP = VBA.Split(strPath, "\")
    tmpCurr = VBA.Split(strPathCurrent, "\")
    For iIndex = 0 To IIf(UBound(tmpP) > UBound(tmpCurr), UBound(tmpCurr), UBound(tmpP))
        If tmpP(iIndex) <> tmpCurr(iIndex) Then
            Exit For
        Else
            i = iIndex
        End If
    Next iIndex
    If i = UBound(tmpCurr) Then
        ' c'est un sous répertoire

        For iIndex = i + 1 To UBound(tmpP)
            GetRelativePath = GetRelativePath & tmpP(iIndex) & "\"
        Next iIndex
        GetRelativePath = Left(GetRelativePath, Len(GetRelativePath) - 1)
    Else
        ' il faut remonter de UBound(tmpCurr) - i

```

```

    For iIndex = 1 To UBound(tmpCurr) - i
        GetRelativePath = GetRelativePath & "..\"
    Next iIndex
    For iIndex = i + 1 To UBound(tmpP)
        GetRelativePath = GetRelativePath & tmpP(iIndex) & "\"
    Next iIndex
    GetRelativePath = Left(GetRelativePath, Len(GetRelativePath) - 1)
End If
Else
    ' deux lecteurs différents

    GetRelativePath = strPath
End If

End Function

```

Le premier paramètre correspond au chemin du fichier ou du dossier, le second correspond au chemin courant.

Exemple :

```

?getrelativepath("c:\toto\tata\test.xls","c:\tintin")
..\toto\tata\test.xls
?getrelativepath("c:\toto\tata\test.xls","c:\toto\tata")
\test.xls

```

### Comment récupérer le nom d'un fichier à partir d'un chemin complet ?

**Auteurs :** [ThierryAIM](#) ,

Cette fonction reçoit le chemin complet d'un fichier en paramètre et renvoie le nom du fichier :

vb

```

Public Function ExtractFileName(ByVal sFullPath As String) As String
    If InStr(sFullPath, "\") = 0 Or Right(sFullPath, 1) = "\" Then
        ExtractFileName = ""
        Exit Function
    End If
    ExtractFileName = Mid(sFullPath, InStrRev(sFullPath, "\") + 1)
End Function

```

lien : [FAQ](#) Comment récupérer le répertoire d'un fichier à partir de son chemin complet ?

lien : [FAQ](#) Comment récupérer l'extension d'un fichier à partir d'un chemin complet ?

### Comment récupérer l'extension d'un fichier à partir d'un chemin complet ?

**Auteurs :** [ThierryAIM](#) ,

Cette fonction reçoit le chemin complet d'un fichier en paramètre et renvoie l'extension du fichier, si elle existe, sinon renvoie une chaîne vide :

(Nécessite la fonction [FAQ](#) Comment récupérer le nom d'un fichier à partir d'un chemin complet ?)

vb

vb

```
Public Function ExtractFileExt(ByVal sFullPath As String) As String
    Dim sName As String
    sName = ExtractFileName(sFullPath)
    If InStr(sName, ".") = 0 Then
        ExtractFileExt = ""
    Else
        ExtractFileExt = Mid(sName, InStrRev(sName, ".") + 1)
    End If
End Function
```

lien : [FAQ](#) Comment récupérer le nom d'un fichier à partir d'un chemin complet ?

lien : [FAQ](#) Comment récupérer le répertoire d'un fichier à partir de son chemin complet ?

## Comment compresser et décompresser des fichiers ?

Auteurs : [Elifqaoui](#) , [Romain Puyfoulhoux](#) ,

VB n'inclut pas de composant permettant de compresser des fichiers. Il est possible de s'en sortir en exécutant, via la fonction Shell, un programme de type pkzip. Mais une solution plus pratique est d'utiliser une librairie ou un active-x. Ici nous utiliserons la **zlib**, qui a l'avantage d'être gratuite, open source et de créer des zips standards. Le code à écrire en VB pour la manipuler étant assez conséquent, nous allons aussi importer les classes VB d'Andrew McMillan disponibles dans les fichiers [zipclass.zip](#) et [ZipExtractionClass.zip](#) (la zlib est aussi dans ces zips).

Après avoir téléchargé ces 2 fichiers, importez les classes dans votre projet et copiez le fichier **zlib.dll** dans le répertoire de votre projet ou dans le répertoire système.

Voici comment créer un fichier zip :

vb

```
Dim z As ZipClass

Set z = New ZipClass
z.AddFile "c:\test.doc"
z.AddFile "c:\test.jpg"
z.WriteZip "c:\test.zip", True
Set z = Nothing
```

Et comment faire une extraction :

vb

```
Dim zip As ZipExtractionClass

Set zip = New ZipExtractionClass
If zip.OpenZip("C:\Test\Test.zip") Then
    If zip.Extract("C:\Test\Extract", True, True) Then
        MsgBox "Extraction terminée.", vbInformation
    End If
    zip.CloseZip
End If
Set zip = Nothing
```

lien : [La zlib](#)

lien : [La création d'un zip par Andrew McMillan](#)

**lien : L'extraction d'un zip par Andrew McMillan**

## Comment déterminer le type d'un lecteur ?

**Auteurs : Alexandre Lokchine , Romain Puyfoulhoux ,**

**Le FileSystemObject vous permet de le faire facilement :**

```
vb

Public Function TypeLecteur(ByVal drvpath) As String

    Dim fs As FileSystemObject, d As Drive, t As String

    Set fs = New FileSystemObject

    On Error GoTo fail
    Set d = fs.GetDrive(drvpath)

    Select Case d.DriveType
        Case 0: t = "Inconnu"
        Case 1: t = "Amovible"
        Case 2: t = "Fixe"
        Case 3: t = "Réseau"
        Case 4: t = "CD-ROM"
        Case 5: t = "Disque RAM"
    End Select

fin:
    TypeLecteur = t
    Exit Function

fail:
    t = "Introuvable"
    Resume fin

End Function
```

**Cette fonction attend en argument la lettre d'un lecteur et renvoie son type en toutes lettres. Par exemple :**

```
vb

MsgBox TypeLecteur("c")
```

**lien : [FAQ](#) Quelle référence dois-je ajouter à mon projet pour pouvoir utiliser le FileSystemObject ?**

## Comment savoir si un CD se trouve dans l'un des lecteurs du système (2 codes)?

**Auteurs : Romain Puyfoulhoux , Alexandre Lokchine , ridan ,**

**Voici une fonction qui vous renvoie le premier lecteur qui contient un CD, ou une chaîne vide s'il n'y en a aucun. Vous devez ajouter le FileSystemObject dans les références du projet.**

```
vb

Public Function LecteurAvecCD() As String

    Dim fso As FileSystemObject, lecteur As Drive
    Dim strPath As String, strLecteurCD As String

    Set fso = New FileSystemObject
```



vb

```
For Each lecteur In fso.Drives
    If lecteur.DriveType = 4 Then
        On Error GoTo suite
        strPath = Dir(lecteur.path)
        strLecteurCD = lecteur.path
        Exit For
    End If
suite:
    Next

    Set fso = Nothing
    LecteurAvecCD = strLecteurCD

End Function
```

Sans utilisation de FSO (FileSystemObject), placez ce code dans un module :

vb

```
Private Declare Function GetVolumeInformation Lib "kernel32.dll" Alias "GetVolumeInformationA" ( _
    ByVal lpRootPathName As String, _
    ByVal lpVolumeNameBuffer As String, _
    ByVal nVolumeNameSize As Long, _
    ByRef lpVolumeSerialNumber As Long, _
    ByRef lpMaximumComponentLength As Long, _
    ByRef lpFileSystemFlags As Long, _
    ByVal lpFileSystemNameBuffer As String, _
    ByVal nFileSystemNameSize As Long) As Long

Public Function Test_Lecteur(Lecteur As String) As Long

    Dim VolNameBuffer As String
    VolNameBuffer = String(255, ".")
    Test_Lecteur = GetVolumeInformation(Lecteur, VolNameBuffer, 255, 0, 0, 0, 0, 255)

End Function
```

Appel de la procédure :

vb

```
Dim result As Long
result = Test_Lecteur("D:\")
If result = 0 Then
    MsgBox "Lecteur Vide"
Else
    MsgBox "CD présent dans le lecteur"
End If
```

lien : [FAQ](#) Quelle référence dois-je ajouter à mon projet pour pouvoir utiliser le FileSystemObject ?

## Comment récupérer l'espace Total/Libre/Utilisé d'un disque ?

Auteurs : ridan ,

1) En utilisant l'API Windows :

vb

```
Private Declare Function GetDiskFreeSpace Lib "kernel32.dll" Alias "GetDiskFreeSpaceA" ( _
```

vb

```
ByVal lpRootPathName As String, _
ByRef lpSectorsPerCluster As Long, _
ByRef lpBytesPerSector As Long, _
ByRef lpNumberOfFreeClusters As Long, _
ByRef lpTtoalNumberOfClusters As Long) As Long
```

```
Private Function conversion(nombre As Currency) As String
```

```
Const KB As Double = 1024
Const MB As Double = KB * 1024
Const GB As Double = MB * 1024
```

```
If nombre <= 999 Then
    conversion = Str(nombre) & " bytes"
ElseIf nombre <= KB * 999 Then
    conversion = Format((nombre / KB), "0.00") & " KB"
ElseIf nombre <= MB * 999 Then
    conversion = Format((nombre / MB), "0.00") & " MB"
ElseIf nombre <= GB * 999 Then
    conversion = Format((nombre / GB), "0.00") & " GB"
End If
```

```
End Function
```

```
Public Function Espace_Disque() As String
```

```
Dim SectorsPerCluster As Long
Dim BytesPerCluster As Long
Dim NumberOfFreeClusters As Long
Dim TtoalNumberOfClusters As Long
```

```
Dim Espace_Libre As Currency
Dim Espace_Total As Currency
Dim Espace_Utilise As Currency
```

```
GetDiskFreeSpace "C:\", SectorsPerCluster, _
    BytesPerSector, _
    NumberOfFreeClusters, _
    TtoalNumberOfClusters
```

```
Espace_Libre = NumberOfFreeClusters * BytesPerSector * SectorsPerCluster
Espace_Total = BytesPerSector * TtoalNumberOfClusters * SectorsPerCluster
Espace_Utilise = Espace_Total - Espace_Libre
```

```
Espace_Disque = "Espace Libre : " & conversion(Espace_Libre) & vbCrLf & _
    "Espace Total : " & conversion(Espace_Total) & vbCrLf & _
    "Espace Utilisé : " & conversion(Espace_Utilise)
```

```
End Function
```

## 2) En utilisant le FileSystemObject :

vb

```
' Ajouter la référence à Microsoft Scripting Runtime à votre projet
```

```
Sub InfoEspaceDisques()
```

```
Dim fso As New FileSystemObject, d As Drive, txt As String
For Each d In fso.Drives
```

```
    'If d.DriveType = 2 Or d.DriveType = 5 or d.DriveType = 3 Then
```

```
    If d.DriveType = Fixed Or d.DriveType = RamDisk Or _
    d.DriveType = Remote Then
```

```
        txt = txt & "Disk " & d.DriveLetter & "(" & d.VolumeName & ")" & vbCrLf
```

```
        txt = txt & "Total : " & d.TotalSize & vbCrLf
```

```
vb
        txt = txt & "Libre : " & d.FreeSpace & vbCrLf
        Debug.Print txt
    End If
Next
Set fso = Nothing
End Sub
```

## Comment récupérer la vidéo d'une WebCam ?

Auteurs : ridan ,

Placez un contrôle Timer et ce code :

```
vb
Private Declare Function capCreateCaptureWindow Lib "avicap32.dll" Alias "capCreateCaptureWindowA"
( _
    ByVal lpszWindowName As String, _
    ByVal dwStyle As Long, _
    ByVal X As Long, _
    ByVal Y As Long, _
    ByVal nWidth As Long, _
    ByVal nHeight As Long, _
    ByVal hwndParent As Long, _
    ByVal nID As Long) As Long

Private Declare Function SendMessage Lib "user32.dll" Alias "SendMessageA" ( _
    ByVal hwnd As Long, _
    ByVal wMsg As Long, _
    ByVal wParam As Long, _
    ByVal lParam As Any) As Long

Private Const WM_USER As Long = &H400
Private Const WM_CAP_DRIVER_CONNECT As Long = WM_USER + 10
Private Const WM_CAP_DRIVER_DISCONNECT As Long = WM_USER + 11
Private Const WM_CAP_GRAB_FRAME As Long = WM_USER + 60
Private Const WM_CAP_EDIT_COPY As Long = WM_USER + 30

Private iResult As Long

Private Sub Form_Load()

    Timer1.Interval = 50
    iResult = capCreateCaptureWindow("Capture", _
        0, 0, 0, _
        Picture1.ScaleWidth, _
        Picture1.ScaleHeight, _
        Me.hwnd, 0)
    SendMessage iResult, WM_CAP_DRIVER_CONNECT, 0, 0

End Sub

Private Sub Form_Unload(Cancel As Integer)

    SendMessage iResult, WM_CAP_DRIVER_DISCONNECT, 0, 0

End Sub

Private Sub Timer1_Timer()

    Clipboard.Clear
    SendMessage iResult, WM_CAP_GRAB_FRAME, 0, 0
    SendMessage iResult, WM_CAP_EDIT_COPY, 0, 0
```

```
vb
Picture1.Picture = Clipboard.GetData

End Sub
```

## Comment lire/modifier les attributs d'un fichier/répertoire par l'API Windows ?

Auteurs : shwin , ThierryAIM ,

En complément des fonctions SetAttr et GetAttr, ces fonctions de l'API Windows permettent de modifier ou de lire les attributs d'un fichier, non accessibles par VB6 :

```
vb

Private Const FILE_ATTRIBUTE_READONLY As Long = &H1 'Fichier en lecture seule.
Private Const FILE_ATTRIBUTE_HIDDEN As Long = &H2 'Fichier caché.
Private Const FILE_ATTRIBUTE_SYSTEM As Long = &H4 'Fichier système.
Private Const FILE_ATTRIBUTE_DIRECTORY As Long = &H10 'L'élément est un répertoire.
Private Const FILE_ATTRIBUTE_ARCHIVE As Long = &H20 'Le fichier a l'attribut archive.
Private Const FILE_ATTRIBUTE_NORMAL As Long = &H80 'Le fichier n'a pas d'attribut.
Private Const FILE_ATTRIBUTE_TEMPORARY As Long = &H100 'Fichier temporaire.
Private Const FILE_ATTRIBUTE_COMPRESSED As Long = &H800 'Fichier (répertoire) compressé.
Private Const FILE_ATTRIBUTE_ENCRYPTED As Long = &H4000 ' Fichier crypté

Private Declare Function SetFileAttributes Lib "kernel32" Alias "SetFileAttributesA" _
(ByVal lpFileName As String, ByVal dwFileAttributes As Long) As Long
Private Declare Function GetFileAttributes Lib "kernel32" Alias "GetFileAttributesA" _
(ByVal lpFileName As String) As Long
```

Modifier les attributs d'un fichier :

```
vb

Private Sub Command1_Click()
MsgBox SetFileAttributes("C:\str.txt", FILE_ATTRIBUTE_SYSTEM)
End Sub
```

Lire les attributs d'un fichier :

```
vb

If GetFileAttributes("C:\test.ini") And FILE_ATTRIBUTE_ENCRYPTED Then MsgBox "Fichier crypté"
```

## Comment jouer des fichiers wav sans lecteurs (WMP ou MID) ?

Auteurs : odan71 , Tofalu ,

Ce code intègre un son wav sans utiliser Windows média player ou mdi.

```
vb

Private Const SND_APPLICATION = &H80 ' look for application specific association
Private Const SND_ALIAS = &H10000 ' name is a WIN.INI [sounds] entry
Private Const SND_ALIAS_ID = &H110000 ' name is a WIN.INI [sounds] entry identifier
Private Const SND_ASYNC = &H1 ' play asynchronously
Private Const SND_FILENAME = &H20000 ' name is a file name
Private Const SND_LOOP = &H8 ' loop the sound until next sndPlaySound
```

vb

```
Private Const SND_MEMORY = &H4           ' lpszSoundName points to a memory file
Private Const SND_NODEFAULT = &H2       ' silence not default, if sound not found
Private Const SND_NOSTOP = &H10         ' don't stop any currently playing sound
Private Const SND_NOWAIT = &H2000      ' don't wait if the driver is busy
Private Const SND_PURGE = &H40          ' purge non-static events for task
Private Const SND_RESOURCE = &H40004    ' name is a resource name or atom
Private Const SND_SYNC = &H0            ' play synchronously (default)
Private Declare Function PlaySound Lib "winmm.dll" Alias "PlaySoundA" _
    (ByVal lpszName As String, ByVal hModule As Long, ByVal dwFlags As Long) As Long
```

Arrêter le son :

vb

```
Private Sub Arreter_Click()
    PlaySound 0&, ByVal 0&, SND_FILENAME Or SND_ASYNC
End Sub
```

Jouer le son :

vb

```
Private Sub Jouer_Click()
    PlaySound "C:\WINDOWS\MEDIA\test.WAV", ByVal 0&, SND_FILENAME Or SND_ASYNC
End Sub
```

## Comment récupérer l'icône associée à un fichier ?

**Auteurs : SilkyRoad ,**

**Ce code nécessite d'activer la référence "Standard OLE Types" :**

vb/vba

```
Declare Function SHGetFileInfo Lib "shell32.dll" Alias "SHGetFileInfoA" _
    (ByVal pszPath As Any, ByVal dwFileAttributes As Long, psfi As SHFILEINFO, _
    ByVal cbFileInfo As Long, ByVal uFlags As Long) As Long

Public Declare Function OleCreatePictureIndirect _
    Lib "olepro32.dll" (PicDesc As PicBmp, RefIID As GUID, _
    ByVal fPictureOwnsHandle As Long, IPic As IPicture) As Long

Public Declare Function ShellExecute Lib "shell32.dll" Alias "ShellExecuteA" _
    (ByVal hwnd As Long, ByVal lpOperation As String, ByVal lpFile As String, _
    ByVal lpParameters As String, ByVal lpDirectory As String, ByVal nShowCmd As Long) As Long

Public Type PicBmp
    Size As Long
    tType As Long
    hBmp As Long
    hPal As Long
    Reserved As Long
End Type

Public Type GUID
    Data1 As Long
    Data2 As Integer
    Data3 As Integer
    Data4(7) As Byte
End Type
```

vb/vba

```

Public Type SHFILEINFO
    hicon As Long
    iIcon As Long
    dwAttributes As Long
    szDisplayName As String * 260
    szTypeName As String * 80
End Type

Public Function GetIconFromFile(FileName As String, IconIndex As Long, _
    UseLargeIcon As Boolean) As IPicture
    '*****
    'Necessite d'activer la reference "Standard OLE Types"
    '*****
    Dim b As SHFILEINFO
    Dim retval As Long
    Dim pic As PicBmp
    Dim IPic As IPicture
    Dim IID_IDispatch As GUID

    retval = SHGetFileInfo(FileName, 0, b, Len(b), &H100)

    With IID_IDispatch
        .Data1 = &H20400
        .Data4(0) = &HC0
        .Data4(7) = &H46
    End With

    With pic
        .Size = Len(b)
        .tType = 3 'vbPicTypeIcon
        .hBmp = b.hicon
    End With

    Call OleCreatePictureIndirect(pic, IID_IDispatch, 1, IPic)
    Set GetIconFromFile = IPic
End Function

```

## Comment lire ou écrire les propriétés avancées d'un fichier ?

**Auteurs :** DarkVader , bbil ,

**Nécessite d'activer la référence *DSO OLE Document Properties Reader 2.1* (dsofile.dll)**



**Lien Microsoft pour téléchargement dll qui encapsule les interfaces "IPropertyStorage" et "IPropertySetStorage".**

vb

```

Sub LitEcritProprieteAvancees()
    Dim dso As New DSOFfile.OleDocumentProperties
    Dim docP As DSOFfile.SummaryProperties
    Dim stFichier As String
    stFichier = App.Path & "\monfic.txt"
    dso.Open stFichier
    Set docP = dso.SummaryProperties
    docP.Comments = docP.Comments &
    IIf(docP.Comments <> "", vbCrLf, "") & "Ajout ou Modification à : " & Now
    dso.Save

```

```
vb
End Sub
```

## Comment lire un fichier avec un charset UTF 8 ?

Auteurs : forum ,

En utilisant un objet ADODB.stream, ajouter pour cela la référence à "Microsoft ActiveX Data Object " version 2.5 ou supérieure.

```
vb
' Ajouter la référence Microsoft ActiveX Data Objects 2.5 ou +
Dim stream As New ADODB.stream
stream.Charset = "UTF-8"
stream.Open
stream.LoadFromFile "c:\tmp\testUTF8.txt"
MsgBox stream.ReadText
stream.Close
```

## Comment compter les occurrences d'une chaîne dans un fichier ?

Auteurs : Cafeine ,

C'est faisable grâce aux RegExp, pensez à ajouter la référence Microsoft Regular Expressions 5.5 :

```
Function CountMatches(ByVal strFic As String, ByVal strSearch As String) As Long

    Dim reg As VBScript_RegExp_55.RegExp
    Dim Matches As VBScript_RegExp_55.MatchCollection
    Dim Fic As Integer
    Dim strBuff As String * 20000
    Dim strBorder As String

    ' instanciation

    Set reg = New VBScript_RegExp_55.RegExp

    reg.Global = True
    reg.IgnoreCase = True
    reg.Multiline = True
    reg.Pattern = "(" & strSearch & ")"

    ' gestion fichier

    Reset
    Fic = FreeFile
    Open strFic For Binary Access Read As #Fic

    Do While Not EOF(Fic)
        strBorder = Right(strBuff, Len(strSearch) - 1)
        Get #Fic, , strBuff
        strBorder = strBorder & strBuff
        Set Matches = reg.Execute(strBorder)
        CountMatches = CountMatches + Matches.Count
    Loop
    Close #Fic

    ' libération


    Set reg = Nothing
```

```
Set Matches = Nothing
```

```
End Function
```

### Exemple :

```
?countmatches("c:\temp\long.txt", " ")  
7500
```

lien :  [Les Expressions Rationnelles et Access par la pratique](#)



Sommaire > Système > Réseaux

## Comment envoyer un e-mail ?

Auteurs : Romain Puyfoulhoux ,

Cochez Microsoft MAPI Controls 6.0 dans la liste des composants.  
Insérez un contrôle MAPISession et un contrôle MAPIMessages à votre projet

vb

```
MAPISession1.SignOn
MAPIMessages1.MsgIndex = -1
MAPIMessages1.SessionID = MAPISession1.SessionID
MAPIMessages1.RecipDisplayName = "toto@domaine.fr" 'Destinataire
MAPIMessages1.MsgSubject = "Un petit bonjour" 'Objet
MAPIMessages1.MsgNoteText = "Salut." 'Texte
MAPIMessages1.Send
MAPISession1.SignOff
```

## Comment envoyer un mail SMTP ?

Auteurs : hpj ,

NOTE : à partir de Windows 2000 Il existe déjà la méthode d'envoi par MAPI dans la FAQ mais tout le monde n'a pas de serveur Exchange. Ajouter la référence "Microsoft CDO for Windows x Library (cdosys.dll)"

vb

```
Dim config As CDO.Configuration
Dim email As CDO.Message

Set config = New CDO.Configuration
With config.Fields
    .Item("http://schemas.microsoft.com/cdo/configuration/sendusing") = CDO.cdoSendUsingPort
    .Item("http://schemas.microsoft.com/cdo/configuration/smtpserver") = "smtp.monserveur.com"
    .Update
End With

Set email = New CDO.Message
With email
    Set .Configuration = config
    .From = "toto@a.com"
    .To = "tata@a.com"
    .Subject = "Sujet"
    .Textbody = "Blabla"
    .Send
End With
```

## Comment décoder un fichier attaché en base 64 ?

Auteurs : Alexandre Lokchine ,

Le codage base 64 est l'un des formats de base servant à encoder les fichiers attachés dans les e-mails en vue de leur transmission. Ce code pourra vous servir à décoder les fichiers attachés en base64.

Placer ce code dans la section Générale d'un formulaire ou dans un module :

vb

```
Function Base64Decode(ByVal base64String as String) as String

    Const Base64 = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/"
    Dim dataLength, sOut, groupBegin
    Dim numDataBytes, CharCounter, thisChar, thisData, nGroup, pOut

    'Suppression des espaces/entrées/tab, s'il y en a
    base64String = Replace(base64String, vbCrLf, "")
    base64String = Replace(base64String, vbTab, "")
    base64String = Replace(base64String, " ", "")

    'la longueur de la chaîne passée doit être un multiple de 4
    dataLength = Len(base64String)
    If dataLength Mod 4 <> 0 Then
        Err.Raise 1, "Base64Decode", "Bad Base64 string."
        Exit Function
    End If

    ' Decodage de chaque groupe:
    For groupBegin = 1 To dataLength Step 4

        ' Chaque groupe se transforme en 3 octets.
        numDataBytes = 3
        nGroup = 0

        For CharCounter = 0 To 3
            ' On convertit chaque caractère en 6 bits de données, et l'ajouter à un
            ' entier pour assurer le stockage temporaire. Si le caractère est
            ' un '=', il y a un byte de données de moins (il ne peut avoir que 2 '=' au
            ' maximum dans toute la chaîne).

            thisChar = Mid(base64String, groupBegin + CharCounter, 1)

            If thisChar = "=" Then
                numDataBytes = numDataBytes - 1
                thisData = 0
            Else
                thisData = InStr(1, Base64, thisChar, vbBinaryCompare) - 1
            End If
            If thisData = -1 Then
                Err.Raise 2, "Base64Decode", "Bad character In Base64 string."
                Exit Function
            End If
            nGroup = 64 * nGroup + thisData
        Next

        'Hex divise l'entier long en 6 groupes de 4 bits
        nGroup = Hex(nGroup)

        'Ajout des zéros de tête
        nGroup = String(6 - Len(nGroup), "0") & nGroup

        'Conversion de l'entier en hexa en 3 caractères
        pOut = Chr(CByte("&H" & Mid(nGroup, 1, 2))) + _
            Chr(CByte("&H" & Mid(nGroup, 3, 2))) + _
            Chr(CByte("&H" & Mid(nGroup, 5, 2)))

        'concatenation avec la chaîne de sortie
        sOut = sOut & Left(pOut, numDataBytes)
    Next

    Base64Decode = sOut
End Function
```

```
vb
End Function
```

## Comment envoyer un mail avec Lotus Notes ?

Auteurs : Alexandre Lokchine ,

```
vb

'Envoi d'un mail avec Lotus Notes
'Subject : sujet du mail
'Attachment : nom d'une pièce jointe
'Recipient : adresse e-mail du destinataire principal
'ccRecipient : destinataire en copie
'bccRecipient : destinataire en copie invisible
'BodyText : corps du mail
'SaveIt : mettre à True pour que le mail soit sauvegardé
'Password : mot de passe

Public Sub SendNotesMail(ByVal Subject As String, ByVal Attachment As String, _
    ByVal Recipient As String, ByVal ccRecipient As String, _
    ByVal bccRecipient As String, ByVal BodyText As String, _
    ByVal SaveIt As Boolean, ByVal Password As String)

    Dim Maildb As Object      'La base des mails
    Dim UserName As String   'Le nom d'utilisateur
    Dim MailDbName As String 'Le nom de la base des mails
    Dim MailDoc As Object    'Le mail
    Dim AttachME As Object   'L'objet pièce jointe en RTF
    Dim Session As Object    'La session Notes
    Dim EmbedObj As Object   'L'objet incorporé

    'Crée une session notes
    Set Session = CreateObject("Notes.NotesSession")

    '*** Cette ligne est réservée aux versions 5.x et supérieur : ***
    Session.Initialize (Password)

    'Récupère le nom d'utilisateur et crée le nom de la base des mails
    UserName = Session.UserName
    MailDbName = Left$(UserName, 1) & Right$(UserName, (Len(UserName) - InStr(1,
UserName, " "))) & ".nsf"

    'Ouvre la base des mails
    Set Maildb = Session.GETDATABASE("", MailDbName)
    If Not Maildb.ISOPEN Then Maildb.OPENMAIL

    'Paramètre le mail à envoyer
    Set MailDoc = Maildb.CREATEDOCUMENT
    MailDoc.Form = "Memo"
    MailDoc.sendto = Recipient
    MailDoc.CopyTo = ccRecipient
    MailDoc.BlindCopyTo = bccRecipient
    MailDoc.Subject = Subject
    MailDoc.Body = BodyText
    MailDoc.SAVEMESSAGEONSEND = SaveIt

    'Prend en compte les pièces jointes
    If Attachment <> "" Then
        Set AttachME = MailDoc.CREATERICHTEXTITEM("Attachment")
        Set EmbedObj = AttachME.EMBEDOBJECT(1454, "", Attachment, "Attachment")
        MailDoc.CREATERICHTEXTITEM ("Attachment")
    End If

    'Envoie le mail
```

vb

```
MailDoc.PostedDate = Now()  
MailDoc.SEND 0, Recipient  
  
Set Maildb = Nothing  
Set MailDoc = Nothing  
Set AttachME = Nothing  
Set Session = Nothing  
Set EmbedObj = Nothing  
End Sub
```

Il est aussi possible d'indiquer à Lotus Notes plusieurs destinataires en affectant un tableau de type Variant à la propriété sendto :

vb

```
Dim recip(25) as Variant  
  
recip(0) = "emailaddress1"  
recip(1) = "emailaddress2"  
maildoc.sendto = recip
```

## Comment obtenir l'adresse IP de la machine ?

Auteurs : **Alexandre Lokchine** , **Romain Puyfoulhoux** ,

Copiez ce code dans un module standard :

vb

```
Private Const MAX_ADAPTER_NAME_LENGTH As Long = 256  
Private Const MAX_ADAPTER_DESCRIPTION_LENGTH As Long = 128  
Private Const MAX_ADAPTER_ADDRESS_LENGTH As Long = 8  
Private Const ERROR_SUCCESS As Long = 0  
  
Private Type IP_ADDRESS_STRING  
    IpAddr(0 To 15) As Byte  
End Type  
  
Private Type IP_MASK_STRING  
    IpMask(0 To 15) As Byte  
End Type  
  
Private Type IP_ADDR_STRING  
    dwNext As Long  
    IpAddress As IP_ADDRESS_STRING  
    IpMask As IP_MASK_STRING  
    dwContext As Long  
End Type  
  
Private Type IP_ADAPTER_INFO  
    dwNext As Long  
    ComboIndex As Long 'reserved  
    sAdapterName(0 To (MAX_ADAPTER_NAME_LENGTH + 3)) As Byte  
    sDescription(0 To (MAX_ADAPTER_DESCRIPTION_LENGTH + 3)) As Byte  
    dwAddressLength As Long  
    sIPAddress(0 To (MAX_ADAPTER_ADDRESS_LENGTH - 1)) As Byte  
    dwIndex As Long  
    uType As Long  
    uDhcpEnabled As Long  
    CurrentIpAddress As Long  
    IpAddressList As IP_ADDR_STRING
```

**vb**

```

GatewayList As IP_ADDR_STRING
DhcpServer As IP_ADDR_STRING
bHaveWins As Long
PrimaryWinsServer As IP_ADDR_STRING
SecondaryWinsServer As IP_ADDR_STRING
LeaseObtained As Long
LeaseExpires As Long
End Type

Private Declare Function GetAdaptersInfo Lib "iphlpapi.dll" _
    (pTcpTable As Any, pdwSize As Long) As Long

Private Declare Sub CopyMemory Lib "kernel32" Alias "RtlMoveMemory" _
    (dst As Any, src As Any, ByVal bcount As Long)

Private Function TrimNull(item As String)

Dim pos As Integer

pos = InStr(item, Chr$(0))
If pos Then
    TrimNull = Left$(item, pos - 1)
Else
    TrimNull = item
End If

End Function

Public Function LocalIPAddress() As String

Dim cbRequired As Long
Dim buff() As Byte
Dim Adapter As IP_ADAPTER_INFO
Dim AdapterStr As IP_ADDR_STRING
Dim ptr1 As Long
Dim sIPAddr As String
Dim found As Boolean

GetAdaptersInfo ByVal 0&, cbRequired

If cbRequired > 0 Then
    ReDim buff(0 To cbRequired - 1) As Byte
    If GetAdaptersInfo(buff(0), cbRequired) = ERROR_SUCCESS Then
        ptr1 = VarPtr(buff(0))
        Do While (ptr1 <> 0)
            CopyMemory Adapter, ByVal ptr1, LenB(Adapter)
            With Adapter
                sIPAddr = TrimNull(StrConv(.IpAddressList.IpAddress.IpAddr, vbUnicode))
                If Len(sIPAddr) > 0 Then
                    found = True
                    Exit Do
                End If
            End With
            ptr1 = .dwNext
        Loop
    End If
End If

LocalIPAddress = sIPAddr

End Function

```

La fonction LocalIPAddress() vous renvoie l'adresse IP.

## Comment obtenir le nom de la machine ?

Auteurs : [Abelman](#) , [hpj](#) ,

Copiez cette déclaration au début d'un module standard :

```
vb
Public Declare Function GetComputerName Lib "kernel32" Alias "GetComputerNameA" _
    (ByVal lpBuffer As String, nSize As Long) As Long
```

Copiez ensuite cette fonction dans votre module : (pour Win 98)

```
vb
Private Function NomOrdinateur() As String

Dim sComputerName As String
Dim iSize As Long

'Un premier appel pour avoir le nombre de caractères nécessaire pour sComputerName
GetComputerName sComputerName, iSize

'On met sComputerName à la bonne taille
sComputerName = Space(iSize)

'Appel final
GetComputerName sComputerName, iSize
NomOrdinateur = sComputerName

'PS : On aurait aussi pu déclarer sComputerName avec une taille assez grande :
' (dim sComputerName as string*32).
' Un seul appel de GetComputerName aurait alors suffit

End Function
```

Pour Win 2000 et supérieur :

```
vb
Dim NomOrdinateur As String

NomOrdinateur = Environ("COMPUTERNAME")
```

## Comment savoir si l'on est connecté à internet ?

Auteurs : [Romain Puyfoulhoux](#) ,

La fonction ConnexionInternetActive() ci-dessous renvoie Vrai si l'on est connecté à internet.

```
vb
Private Declare Function InternetGetConnectedState Lib "wininet.dll" _
    (ByRef lpdwFlags As Long, _
    ByVal dwReserved As Long) As Long
```

vb

```
Public Function ConnexionInternetActive() As Boolean

ConnexionInternetActive = InternetGetConnectedState(0&, 0&)

End Function
```

## Comment uploader un fichier par FTP ?

Auteurs : **ThierryAIM**, **Romain Puyfoulhoux**,

Cochez "Microsoft Internet Transfer Control" dans les composants du projet. Placez un contrôle de ce type sur une form. Son nom par défaut est "Inet1". Le code ci-dessous montre comment l'utiliser pour uploader un fichier :

vb

```
Private Sub CopieParFtp(ByVal source As String, ByVal destination, ByVal login As String, _
    ByVal motdepasse As String, ByVal url As String)

With Inet1
    .AccessType = icDirect
    .Protocol = icFTP
    .URL = "ftp://" & login & ":" & motdepasse & "@" & url
    .Execute , "SEND " & source & " " & "/" & destination
    While .StillExecuting
        DoEvents
    Wend
    .Cancel
End With

End Sub
```

Voici un exemple d'appel à la procédure :

vb

```
CopieParFtp "c:\lettre.txt", "lettre.txt", "bill", "HgDrk62B", "microsoft.com"
```

## Comment envoyer un fichier via FTP avec des API ?

Auteurs : **shwin**,

vb

```
'-----
'Déclaration des API
'-----

Private Declare Function InternetCloseHandle Lib "wininet.dll" _
    (ByVal hInet As Long) As Integer

Private Declare Function InternetConnect Lib "wininet.dll" Alias "InternetConnectA" _
    (ByVal hInternetSession As Long, ByVal sServerName As String, _
    ByVal nServerPort As Integer, ByVal sUserName As String, _
    ByVal sPassword As String, ByVal lService As Long, _
    ByVal lFlags As Long, ByVal lContext As Long) As Long

Private Declare Function InternetOpen Lib "wininet.dll" Alias "InternetOpenA" _
    (ByVal sAgent As String, ByVal lAccessType As Long, _
    ByVal sProxyName As String, ByVal sProxyBypass As String, _
    ByVal lFlags As Long) As Long
```

vb

```

Private Declare Function FtpSetCurrentDirectory Lib "wininet.dll" Alias "FtpSetCurrentDirectoryA" _
    (ByVal hFtpSession As Long, ByVal lpszDirectory As String) As Boolean

Private Declare Function FtpPutFile Lib "wininet.dll" Alias "FtpPutFileA" _
    (ByVal hConnect As Long, ByVal lpszLocalFile As String, _
    ByVal lpszNewRemoteFile As String, ByVal dwFlags As Long, _
    ByVal dwContext As Long) As Boolean

Private Sub envoi_fichier_Click()
Dim HwndConnect As Long
Dim HwndOpen As Long

HwndOpen = InternetOpen("SiteWeb", 0, vbNullString, vbNullString, 0) 'Ouvre internet
HwndConnect = InternetConnect(HwndOpen, "ftp.mon_site_web.qc.ca", 21, "mon_username", _
    "mon_password", 1, 0, 0) 'Connection au site ftp

'On arrive par défaut dans le répertoire ftp.mon_site_web.qc.ca/root/
'Cependant, je veux envoyer le fichier dans le dossier page_web/documents
'Donc il faut que je me déplace dans le ftp pour me positionner dans le bon répertoire
FtpSetCurrentDirectory HwndConnect, "page_web/documents"

'Le fichier est envoyé ici
'Mon fichier sur le disque dur est le suivant: C:\windows\bureau\test.txt
'Cependant, je veux que le fichier sur le ftp se nomme shwin.txt donc, le fichier test.txt est renommé en shwin.
'On aurait pu laisser le nom test.txt au lieu de shwin.txt mais ainsi, on a un exemple de renommage.
FtpPutFile HwndConnect, "C:\windows\bureau\test.txt", "shwin.txt", &H0, 0

InternetCloseHandle HwndConnect 'Ferme la connection
InternetCloseHandle HwndOpen 'Ferme internet
    
```

## Comment obtenir les noms de toutes les machines sur un domaine Windows ?

**Auteurs : Alexandre Lokchine ,**

Une méthode consiste à passer par les appels internes de NetBIOS. La fonction NetServerEnum() n'étant disponible que sur Windows NT ou supérieur, ce code ne fonctionne pas sur Windows 9x.

**Copiez le code suivant dans un module :**

vb

```

'Définition des constantes
Private Const MAX_PREFERRED_LENGTH As Long = -1
Private Const NERR_SUCCESS As Long = 0&
Private Const ERROR_MORE_DATA As Long = 234&
Private Const SV_TYPE_ALL As Long = &HFFFFFFF
Private Const SV_PLATFORM_ID_OS2 As Long = 400
Private Const SV_PLATFORM_ID_NT As Long = 500

'Masque pour obtenir la version OS Majeure à partir de la variable version globale
Private Const MAJOR_VERSION_MASK As Long = &HF

Private Type SERVER_INFO_100
    sv100_platform_id As Long
    sv100_name As Long
End Type

Private Declare Function NetServerEnum Lib "netapi32" _
    (ServerName As String, Level As Integer, Buffer As Long, MaxBufferSize As Long, _
    ReturnCode As Long) As Long
    
```



vb

```

(ByVal servername As Long, ByVal level As Long, buf As Any, _
ByVal prefmaxlen As Long, entriesread As Long, totalentries As Long, _
ByVal servertype As Long, ByVal domain As Long, resume_handle As Long) As Long

Private Declare Function NetApiBufferFree Lib "netapi32" (ByVal Buffer As Long) As Long

Private Declare Sub CopyMemory Lib "kernel32" Alias "RtlMoveMemory" _
(pTo As Any, uFrom As Any, ByVal lSize As Long)

Private Declare Function lstrlenW Lib "kernel32" _
(ByVal lpString As Long) As Long

Public Function GetServers(sDomain As String) As String

    'liste de tous les serveurs dans un domaine

    Dim bufptr          As Long
    Dim dwEntriesread   As Long
    Dim dwTotalentries  As Long
    Dim dwResumehandle  As Long
    Dim se100           As SERVER_INFO_100
    Dim success          As Long
    Dim nStructSize     As Long
    Dim cnt             As Long
    Dim resultat        As String

    nStructSize = LenB(se100)
    'la liste des noms est obtenue avec la fonction NetServerEnum
    success = NetServerEnum(0&, 100, bufptr, MAX_PREFERRED_LENGTH, dwEntriesread, _
        dwTotalentries, SV_TYPE_ALL, 0&, dwResumehandle)

    If success = NERR_SUCCESS And _
        success <> ERROR_MORE_DATA Then 'si tout se passe bien

        For cnt = 0 To dwEntriesread - 1
            CopyMemory se100, ByVal bufptr + (nStructSize * cnt), nStructSize
            'on scanne le buffer en memoire et pour chaque entrée, conversion en String
            resultat = resultat & GetPointerToByteStringW(se100.sv100_name) & "|"
        Next

    End If

    'nettoyage du buffer que le système a reservé pour la liste des noms
    Call NetApiBufferFree(bufptr)

    'on retourne le string contenant les noms séparés par des "|"
    GetServers = resultat

End Function

Public Function GetPointerToByteStringW(ByVal dwData As Long) As String

    'fonction auxiliaire qui reçoit un pointeur vers une chaîne dans un buffer interne
    'Windows et la convertit en String exploitable en VB

    Dim tmp() As Byte
    Dim tmpLen As Long

    If dwData <> 0 Then
        tmpLen = lstrlenW(dwData) * 2
        If tmpLen <> 0 Then
            ReDim tmp(0 To (tmpLen - 1)) As Byte
            CopyMemory tmp(0), ByVal dwData, tmpLen
            GetPointerToByteStringW = tmp
        End If
    End If

End Function

```

```
vb
End Function
```

Ensuite, la fonction GetServers est utilisée de la manière suivante :

```
vb
Dim maliste as String
maliste=GetServers(vbNullString)
```

Cette fonction nous retourne la liste des noms des machines, séparés par le caractère "|". Il est ensuite recommandé d'appeler la fonction Split() pour copier les noms dans un tableau.

## Comment obtenir l'adresse MAC de la carte réseau ?

Auteurs : Alexandre Lokchine ,

Une adresse MAC est un identifiant stocké dans une interface réseau. Copiez le code ci-dessous dans un module standard. La fonction GetMACAddress() vous renvoie l'adresse MAC.

```
vb
Private Const NCBASTAT As Long = &H33
Private Const NCBNAMSZ As Long = 16
Private Const HEAP_ZERO_MEMORY As Long = &H8
Private Const HEAP_GENERATE_EXCEPTIONS As Long = &H4
Private Const NCBRESET As Long = &H32

Private Type NET_CONTROL_BLOCK
'definition du type net control Block
ncb_command As Byte
ncb_retcode As Byte
ncb_lsn As Byte
ncb_num As Byte
ncb_buffer As Long
ncb_length As Integer
ncb_callname As String * NCBNAMSZ
ncb_name As String * NCBNAMSZ
ncb_rto As Byte
ncb_sto As Byte
ncb_post As Long
ncb_lana_num As Byte
ncb_cmd_cplt As Byte
ncb_reserve(9) As Byte
ncb_event As Long
End Type

Private Type ADAPTER_STATUS
'definition du type pour definir le statut de l'adaptateur réseau
adapter_address(5) As Byte
rev_major As Byte
reserved0 As Byte
adapter_type As Byte
rev_minor As Byte
duration As Integer
frmr_rcv As Integer
frmr_xmit As Integer
iframe_rcv_err As Integer
xmit_aborts As Integer
xmit_success As Long
rcv_success As Long
```

```

vb
    iframe_xmit_err    As Integer
    recv_buff_unavail As Integer
    t1_timeouts       As Integer
    ti_timeouts       As Integer
    Reserved1         As Long
    free_ncbs         As Integer
    max_cfg_ncbs     As Integer
    max_ncbs         As Integer
    xmit_buf_unavail  As Integer
    max_dgram_size   As Integer
    pending_sess     As Integer
    max_cfg_sess     As Integer
    max_sess         As Integer
    max_sess_pkt_size As Integer
    name_count       As Integer
End Type

Private Type NAME_BUFFER
    name      As String * NCBNAMSZ
    name_num  As Integer
    name_flags As Integer
End Type

Private Type ASTAT
    adapt      As ADAPTER_STATUS
    NameBuff(30) As NAME_BUFFER
End Type

Private Declare Function Netbios Lib "netapi32" (pncb As NET_CONTROL_BLOCK) As Byte

Private Declare Sub CopyMemory Lib "kernel32" _
    Alias "RtlMoveMemory" (hpvDest As Any, ByVal hpvSource As Long, ByVal cbCopy As Long)

Private Declare Function GetProcessHeap Lib "kernel32" () As Long

Private Declare Function HeapAlloc Lib "kernel32" _
    (ByVal hHeap As Long, ByVal dwFlags As Long, ByVal dwBytes As Long) As Long

Private Declare Function HeapFree Lib "kernel32" _
    (ByVal hHeap As Long, ByVal dwFlags As Long, lpMem As Any) As Long

Public Function GetMACAddress() As String

    Dim tmp As String
    Dim pASTAT As Long
    Dim NCB As NET_CONTROL_BLOCK
    Dim AST As ASTAT

    NCB.ncb_command = NCBRESET
    Call Netbios(NCB)

    NCB.ncb_callname = ""
    NCB.ncb_command = NCBASTAT

    NCB.ncb_lana_num = 0
    NCB.ncb_length = Len(AST)
    'allocation de la memoire dans le tas du processus
    pASTAT = HeapAlloc(GetProcessHeap(), HEAP_GENERATE_EXCEPTIONS Or _
        HEAP_ZERO_MEMORY, NCB.ncb_length)

    If pASTAT = 0 Then
        Debug.Print "pas assez de mémoire!" 'bon, y a peu de chance que ca arrive :o)
        Exit Function
    End If

    NCB.ncb_buffer = pASTAT
    'appel de la fonction netbios qui va nous donner les stats de la carte

```

vb

```
'(dont l'adresse MAC)
Call Netbios(NCB)

CopyMemory AST, NCB.ncb_buffer, Len(AST)

tmp = Right$("00" & Hex(AST.adapt.adapter_address(0)), 2) & " " & _
      Right$("00" & Hex(AST.adapt.adapter_address(1)), 2) & " " & _
      Right$("00" & Hex(AST.adapt.adapter_address(2)), 2) & " " & _
      Right$("00" & Hex(AST.adapt.adapter_address(3)), 2) & " " & _
      Right$("00" & Hex(AST.adapt.adapter_address(4)), 2) & " " & _
      Right$("00" & Hex(AST.adapt.adapter_address(5)), 2)
'désallocation de la mémoire...
HeapFree GetProcessHeap(), 0, pASTAT

GetMACAddress = tmp

End Function
```

## Comment ouvrir la fenêtre de connexion ou déconnexion à un lecteur réseau ?

Auteurs : Alexandre Lokchine , Romain Puyfoulhoux ,

Placez ces déclarations dans un module standard :

vb

```
Public Declare Function WNetConnectionDialog Lib "mpr.dll" (ByVal hwnd As Long, _
                                                         ByVal dwType As Long) As Long
Public Declare Function WNetDisconnectDialog Lib "mpr.dll" (ByVal hwnd As Long, _
                                                           ByVal dwType As Long) As Long
Public Const RESOURCETYPE_DISK = &H1, RESOURCETYPE_PRINT = &H2
```

Ensuite, utilisez l'appel adéquat dans chacune des situations :

vb

```
Dim x As Long

'Connecter un lecteur réseau
x = WNetConnectionDialog(Me.hwnd, RESOURCETYPE_DISK)

'Déconnecter un lecteur réseau
x = WNetDisconnectDialog(Me.hwnd, RESOURCETYPE_DISK)

'Connecter une imprimante
x = WNetConnectionDialog(Me.hwnd, RESOURCETYPE_PRINT)

'Déconnecter une imprimante
x = WNetDisconnectDialog(Me.hwnd, RESOURCETYPE_PRINT)
```

## Comment obtenir la liste des ports série, parallèle, réseau ouverts ?

Auteurs : hpj , Alexandre Lokchine ,

Copiez ce code dans un module :

vb

```
Private Declare Function EnumPorts Lib "winspool.drv" Alias "EnumPortsA" _
```

vb

```

        (ByVal pName As String, ByVal nLevel As Long, _
        lpbPorts As Any, ByVal cbBuf As Long, _
        pcbNeeded As Long, pcReturned As Long) As Long

Private Declare Function lstrlenA Lib "kernel32" (lpString As Any) As Long
Private Declare Function lstrcpyA Lib "kernel32" (lpString1 As Any, lpString2 As Any) As Long

Private Const SIZEOFPORT_INFO_2 = 20

Private Type PORT_INFO_2
    pPortName As Long
    pMonitorName As Long
    pDescription As Long
    fPortType As Long
    Reserved As Long
End Type

Private Enum PortTypes
    PORT_TYPE_WRITE = &H1
    PORT_TYPE_READ = &H2
    PORT_TYPE_REDIRECTED = &H4
    PORT_TYPE_NET_ATTACHED = &H8
End Enum

Private Function GetStrFromPtrA(lpszA As Long) As String
    GetStrFromPtrA = String$(lstrlenA(ByVal lpszA), 0)
    Call lstrcpyA(ByVal GetStrFromPtrA, ByVal lpszA)
End Function

Public Function GetPorts() As String

    Dim pcbNeeded As Long, pcReturned As Long, Boucle As Integer
    Dim PortI2() As PORT_INFO_2
    Dim StrPortType As String, ret As String

    EnumPorts vbNullString, 2, 0, 0, pcbNeeded, pcReturned
    If pcbNeeded Then
        ReDim PortI2((pcbNeeded / SIZEOFPORT_INFO_2))
        If EnumPorts(vbNullString, 2, PortI2(0), pcbNeeded, pcbNeeded, pcReturned) Then
            For Boucle = 0 To (pcReturned - 1)
                With PortI2(Boucle)
                    StrPortType = ""
                    If (.fPortType And PORT_TYPE_WRITE) Then StrPortType = "write "
                    If (.fPortType And PORT_TYPE_READ) Then StrPortType = StrPortType & "read "
                    If (.fPortType And PORT_TYPE_REDIRECTED) Then StrPortType =
StrPortType & "redirected "
                    If (.fPortType And PORT_TYPE_NET_ATTACHED) Then StrPortType =
StrPortType & "network"
                    ret = ret & GetStrFromPtrA(.pPortName) & " (" & StrPortType & ")" & "|"
                End With
            Next
        End If
    End If

    If Len(ret) > 0 Then ret = Left(ret, Len(ret) - 1)
    GetPorts = ret

End Function

```

La fonction GetPorts renvoie la liste des ports ouverts, séparés par le caractère '|'. Il est ensuite recommandé d'appeler la fonction Split() afin de copier les éléments dans un tableau.

## Comment redémarrer un poste à distance ?

Auteurs : Romain Puyfoulhoux ,

NOTE : possible uniquement sous Windows NT/2000/XP Il faut avoir les droits sur la machine distante. Fonction : InitiateSystemShutdown qui prend pour argument : \* le nom de la machine à redémarrer \* le message à afficher sur l'ordinateur distant \* le temps d'attente avant l'arrêt ou le reboot de la machine en ms \* un booléen indiquant si vous souhaitez forcer ou non la fermeture des applications \* un booléen à vrai si l'on souhaite rebooter après l'arrêt de la machine

vb

```
Public Declare Function InitiateSystemShutdown Lib "advapi32.dll" Alias _
    "InitiateSystemShutdownA" (ByVal lpMachineName As String, _
    ByVal lpMessage As String, ByVal dwTimeout As Long, _
    ByVal bForceAppsClosed As Long, ByVal bRebootAfterShutdown As Long) As Long

Public Sub RedemarrerADistance()
    Call InitiateSystemShutdown("\\COMPUTER1", "Votre ordinateur va être arrêté à distance _
    depuis VB dans 5 secondes", 5000, True, False)
End Sub
```

lien :  [Site source](#)

## Comment effectuer un ping sur une adresse IP ?

Auteurs : Alexandre Lokchine ,

La solution est expliquée et largement commentée dans un tutoriel dont voici le lien :

lien : [Ping sur une adresse IP](#)

## Comment faire un "ping" en VB

Auteurs : Gaël Donat ,

Il faut utiliser la fonction Ping(HostName As String) du code suivant :

```
Const SOCKET_ERROR = 0
Private Type WSADATA
    wVersion As Integer
    wHighVersion As Integer
    szDescription(0 To 255) As Byte
    szSystemStatus(0 To 128) As Byte
    iMaxSockets As Integer
    iMaxUdpDg As Integer
    lpVendorInfo As Long
End Type
Private Type Hostent
    h_name As Long
    h_aliases As Long
    h_addrtype As Integer
    h_length As Integer
    h_addr_list As Long
End Type
```

```

Private Type IP_OPTION_INFORMATION
    TTL As Byte
    Tos As Byte
    Flags As Byte
    OptionsSize As Long
    OptionsData As String * 128
End Type
Private Type IP_ECHO_REPLY
    Address(0 To 3) As Byte
    Status As Long
    RoundTripTime As Long
    DataSize As Integer
    Reserved As Integer
    data As Long
    Options As IP_OPTION_INFORMATION
End Type
Private Declare Function GetHostByName Lib "wsock32.dll" Alias "gethostbyname" (ByVal
    HostName As String) As Long
Private Declare Function WSASStartup Lib "wsock32.dll" (ByVal wVersionRequired&, lpWSAdata As
    WSAdata) As Long
Private Declare Function WSACleanup Lib "wsock32.dll" () As Long
Private Declare Sub CopyMemory Lib "kernel32" Alias "RtlMoveMemory" (hpvDest As Any, hpvSource As
    Any, ByVal cbCopy As Long)
Private Declare Function IcmpCreateFile Lib "icmp.dll" () As Long
Private Declare Function IcmpCloseHandle Lib "icmp.dll" (ByVal HANDLE As Long) As Boolean
Private Declare Function IcmpSendEcho Lib "ICMP" (ByVal IcmpHandle As Long, ByVal DestAddress As
    Long, _
    ByVal RequestData As String, _
    ByVal RequestSize As Integer, RequestOptns As IP_OPTION_INFORMATION, ReplyBuffer As IP_ECHO_REPLY,
    _
    ByVal ReplySize As Long, ByVal TimeOut As Long) As Boolean
Private Function Ping(HostName As String) As Integer
    Dim hFile As Long, lpWSAdata As WSAdata
    Dim hHostent As Hostent, AddrList As Long
    Dim Address As Long, rIP As String
    Dim OptInfo As IP_OPTION_INFORMATION
    Dim EchoReply As IP_ECHO_REPLY
    Call WSASStartup(&H101, lpWSAdata)
    If GetHostByName(HostName + String(64 - Len(HostName), 0)) <> SOCKET_ERROR Then
        CopyMemory hHostent.h_name, ByVal
        GetHostByName(HostName + String(64 - Len(HostName), 0)), Len(hHostent)
        CopyMemory AddrList, ByVal hHostent.h_addr_list, 4
        CopyMemory Address, ByVal AddrList, 4
    End If
    hFile = IcmpCreateFile()
    If hFile = 0 Then
        MsgBox "Unable to Create File Handle"
        Exit Function
    End If
    OptInfo.TTL = 255
    If IcmpSendEcho(hFile, Address, String(32, "A"), 32, OptInfo,
    EchoReply, Len(EchoReply) + 8, 2000) Then

    rIP = CStr(EchoReply.Address(0)) + "." + CStr(EchoReply.Address(1)) + "." + CStr(EchoReply.Address(2)) + "." +
    _
    CStr(EchoReply.Address(3))
    Else
        Ping = -1
    End If
    If EchoReply.Status = 0 Then
        Ping = EchoReply.RoundTripTime
    End If
    Call IcmpCloseHandle(hFile)
    Call WSACleanup
End Function

```

Cette fonction renvoie -1 en timeout, sinon elle renvoie le temps en millisecondes pour établir le ping.

## Comment récupérer le chemin UNC d'un fichier ?

Auteurs : argyronet ,

*Si vous souhaitez créer un lien avec un fichier situé sur un réseau local, utilisez un chemin UNC (universal naming convention) (convention d'affectation de noms (UNC) : convention de dénomination de fichiers qui fournit un moyen de situer un fichier quelle que soit la machine où il se trouve. Plutôt que de spécifier une lettre de lecteur et un chemin d'accès, un nom UNC utilise la syntaxe \\serveur\partage\chemin\nom\_fichier.), au lieu de la lettre d'identification d'une unité réseau mappée dans l'Explorateur Windows de Microsoft.*

Dans un module, placez le code suivant :

vb

```
Private Declare Function WNetGetConnection Lib "mpr.dll" Alias "WNetGetConnectionA" _
    (ByVal lpszLocalName As String, ByVal lpszRemoteName As String, cbRemoteName As Long) As Long

Public Function fnctGetUNCPath(ByVal PathName As String) As String
    Const MAX_UNC_LENGTH As Integer = 512
    Dim strUNCPath As String
    Dim strTempUNCName As String
    Dim lngReturnErrorCode As Long

    strTempUNCName = String(MAX_UNC_LENGTH, 0)
    lngReturnErrorCode = WNetGetConnection(Left(PathName, 2), strTempUNCName, _
    MAX_UNC_LENGTH)

    If lngReturnErrorCode = 0 Then
        strTempUNCName = Trim(Left(strTempUNCName, InStr(strTempUNCName, vbNullChar) - 1))
        strUNCPath = strTempUNCName & Mid(PathName, 3)
    End If

    fnctGetUNCPath = strUNCPath
End Function
```

Exemple d'utilisation :

vb

```
MsgBox fnctGetUNCPath("U:\Argyronet\OneAnyFile.txt")
```

## Comment Récupérer l'adresse MAC d'un PC distant

Auteurs : Cafeine ,

Voici un module pour récupérer une adresse MAC distante :

```
Option Explicit

' Déclarations pour GetRemoteMACAddress
Private Declare Function inet_addr Lib "WSOCK32.DLL" _
    (ByVal s As String) As Long
Private Declare Function SendARP Lib "iphlpapi.dll" _
    (ByVal DestIP As Long, _
```



```

    ByVal SrcIP As Long, _
    pMacAddr As Long, _
    PhyAddrLen As Long) As Long
Private Declare Sub CopyMemory Lib "KERNEL32" _
    Alias "RtlMoveMemory" _
    (dst As Any, _
    src As Any, _
    ByVal bcount As Long)
' Déclarations pour LetterToUNC
Private Const RESOURCETYPE_ANY = &H0
Private Const RESOURCE_CONNECTED = &H1

Private Type NETRESOURCE
    dwScope As Long
    dwType As Long
    dwDisplayType As Long
    dwUsage As Long
    lpLocalName As Long
    lpRemoteName As Long
    lpComment As Long
    lpProvider As Long
End Type
Private Declare Function WNetOpenEnum Lib "mpr.dll" Alias _
    "WNetOpenEnumA" (ByVal dwScope As Long, ByVal dwType As Long, _
    ByVal dwUsage As Long, lpNetResource As Any, lphEnum As Long) _
    As Long
Private Declare Function WNetEnumResource Lib "mpr.dll" Alias _
    "WNetEnumResourceA" (ByVal hEnum As Long, lpcCount As Long, _
    lpBuffer As Any, lpBufferSize As Long) As Long
Private Declare Function WNetCloseEnum Lib "mpr.dll" ( _
    ByVal hEnum As Long) As Long
Private Declare Function lstrlen Lib "KERNEL32" Alias "lstrlenA" _
    (ByVal lpString As Any) As Long
Private Declare Function lstrcpy Lib "KERNEL32" Alias "lstrcpyA" _
    (ByVal lpString1 As Any, ByVal lpString2 As Any) As Long

' Déclarations pour LetterToUNC
Private Const WS_VERSION_REQD = &H101
Private Const WS_VERSION_MAJOR = WS_VERSION_REQD \ &H100 And &HFF&
Private Const WS_VERSION_MINOR = WS_VERSION_REQD And &HFF&
Private Const MIN_SOCKETS_REQD = 1
Private Const SOCKET_ERROR = -1
Private Const WSADescription_Len = 256
Private Const WSASYS_Status_Len = 128

Private Type HOSTENT
    hName As Long
    hAliases As Long
    hAddrType As Integer
    hLength As Integer
    hAddrList As Long
End Type

Private Type WSADATA
    wversion As Integer
    wHighVersion As Integer
    szDescription(0 To WSADescription_Len) As Byte
    szSystemStatus(0 To WSASYS_Status_Len) As Byte
    iMaxSockets As Integer
    iMaxUdpDg As Integer
    lpszVendorInfo As Long
End Type

Private Declare Function WSAGetLastError Lib "WSOCK32.DLL" () As Long
Private Declare Function WSASStartup Lib "WSOCK32.DLL" (ByVal _
    wVersionRequired As Integer, lpWSADATA As WSADATA) As Long
Private Declare Function WSACleanup Lib "WSOCK32.DLL" () As Long

```

```

Private Declare Function gethostname Lib "WSOCK32.DLL" (ByVal hostname$, _
ByVal HostLen As Long) As Long
Private Declare Function gethostbyname Lib "WSOCK32.DLL" (ByVal _
hostname$) As Long
Private Declare Sub RtlMoveMemory Lib "KERNEL32" (hpvDest As Any, ByVal _
hpvSource&, ByVal cbCopy&)

'-----
'-----
Public Function GetRemoteMACAddress(ByVal pIPDistant As String) As String
Dim lAddr As Long
Dim lMacAddr As Long
Dim lMacAddrByte() As Byte
Dim lPhyAddrLen As Long
Dim lCpt As Integer
' Transforme l'adresse IP texte en adresse IP numérique
lAddr = inet_addr(pIPDistant)
If lAddr <> -1 Then
    ' Taille d'une adresse MAC = 6
    lPhyAddrLen = 6
    ' Recherche l'adresse MAC distante
    If SendARP(lAddr, 0&, lMacAddr, lPhyAddrLen) = 0 Then
        If (lMacAddr <> 0) And (lPhyAddrLen <> 0) Then
            ' Tableau de byte qui contiendra l'adresse MAC
            ReDim lMacAddrByte(0 To lPhyAddrLen - 1)
            ' Copy l'adresse MAC dans le tableau (lMacAddr est une adresse mémoire)
            CopyMemory lMacAddrByte(0), lMacAddr, ByVal lPhyAddrLen
            ' Converti l'adresse MAC en texte
            GetRemoteMACAddress = ""
            For lCpt = LBound(lMacAddrByte) To UBound(lMacAddrByte)
                GetRemoteMACAddress =
GetRemoteMACAddress & Right("00" & Hex(lMacAddrByte(lCpt)), 2) &
IIf(lCpt = UBound(lMacAddrByte), "", "-")
            Next
        End If
    End If
End If
End Function
'-----
'-----
'-----
' Source KB Microsoft : http://support.microsoft.com/kb/192689/fr
Public Function LetterToUNC(DriveLetter As String) As String
Dim hEnum As Long
Dim NetInfo(1023) As NETRESOURCE
Dim entries As Long
Dim nStatus As Long
Dim LocalName As String
Dim UNCName As String
Dim i As Long
Dim r As Long

' Begin the enumeration
nStatus = WNetOpenEnum(RESOURCE_CONNECTED, RESOURCETYPE_ANY, _
0&, ByVal 0&, hEnum)

LetterToUNC = "Drive Letter Not Found"

'Check for success from open enum
If ((nStatus = 0) And (hEnum <> 0)) Then
    ' Set number of entries
    entries = 1024

    ' Enumerate the resource
    nStatus = WNetEnumResource(hEnum, entries, NetInfo(0), _
CLng(Len(NetInfo(0))) * 1024)

```

```

' Check for success
If nStatus = 0 Then
  For i = 0 To entries - 1
    ' Get the local name
    LocalName = ""
    If NetInfo(i).lpLocalName <> 0 Then
      LocalName = Space(lstrlen(NetInfo(i).lpLocalName) + 1)
      r = lstrcpy(LocalName, NetInfo(i).lpLocalName)
    End If

    ' Strip null character from end
    If Len(LocalName) <> 0 Then
      LocalName = Left(LocalName, (Len(LocalName) - 1))
    End If

    If UCase$(LocalName) = UCase$(DriveLetter) Then
      ' Get the remote name
      UNCName = ""
      If NetInfo(i).lpRemoteName <> 0 Then
        UNCName = Space(lstrlen(NetInfo(i).lpRemoteName) _
          + 1)
        r = lstrcpy(UNCName, NetInfo(i).lpRemoteName)
      End If

      ' Strip null character from end
      If Len(UNCName) <> 0 Then
        UNCName = Left(UNCName, (Len(UNCName) _
          - 1))
      End If

      ' Return the UNC path to drive
      LetterToUNC = UNCName

      ' Exit the loop
      Exit For
    End If
  Next i
End If
End If

' End enumeration
nStatus = WNetCloseEnum(hEnum)
End Function

'-----
'-----

' Source KB Microsoft : http://support.microsoft.com/kb/160215/fr
Private Function hbyte(ByVal wParam As Integer)
  hbyte = wParam \ &H100 And &HFF&
End Function

Private Function lobyte(ByVal wParam As Integer)
  lobyte = wParam And &HFF&
End Function

Private Sub SocketsInitialize()
  Dim WSAD As WSADATA
  Dim iReturn As Integer
  Dim sLowByte As String, sHighByte As String, sMsg As String
  iReturn = WSASStartup(WS_VERSION_REQD, WSAD)
  If iReturn <> 0 Then
    MsgBox "Winsock.dll is not responding."
  End If
End Sub

If lobyte(WSAD.wversion) < WS_VERSION_MAJOR Or (lobyte(WSAD.wversion) = _

```

```

WS_VERSION_MAJOR And hbyte(WSAD.wversion) < WS_VERSION_MINOR) Then
sHighByte = Trim$(Str$(hbyte(WSAD.wversion)))
sLowByte = Trim$(Str$(lbyte(WSAD.wversion)))
sMsg = "Windows Sockets version " & sLowByte & "." & sHighByte
sMsg = sMsg & " is not supported by winsock.dll "
MsgBox sMsg
End
End If
'IMaxSockets is not used in winsock 2. So the following check is only
'necessary for winsock 1. If winsock 2 is requested,
'the following check can be skipped.
If WSAD.iMaxSockets < MIN_SOCKETS_REQD Then
sMsg = "This application requires a minimum of "
sMsg = sMsg & Trim$(Str$(MIN_SOCKETS_REQD)) & " supported sockets."
MsgBox sMsg
End
End If
End Sub

Private Sub SocketsCleanup()
Dim lReturn As Long
lReturn = WSACleanup()
If lReturn <> 0 Then
MsgBox "Socket error " & Trim$(Str$(lReturn)) & " occurred in Cleanup "
End
End If
End Sub

Public Function GetIpFromHost(ByVal pHostName As String) As Variant
Dim hostname As String * 256
Dim hostent_addr As Long
Dim host As HOSTENT
Dim hostip_addr As Long
Dim temp_ip_address() As Byte
Dim i As Integer
Dim ip_address As String
Dim lCpt As Integer
Dim lResult() As String
On Error GoTo Gestion_Erreurs
SocketsInitialize
' Retire le double \
If Left(pHostName, 2) = "\\ " Then
pHostName = Right(pHostName, Len(pHostName) - 2)
End If
' Retire un éventuel chemin
If InStr(pHostName, "\") > 0 Then
pHostName = Left(pHostName, InStr(pHostName, "\") - 1)
End If
hostname = Trim$(pHostName & vbNullChar)
hostent_addr = gethostbyname(hostname)
If hostent_addr = 0 Then
MsgBox "Winsock.dll is not responding."
Exit Function
End If
RtlMoveMemory host, hostent_addr, LenB(host)
RtlMoveMemory hostip_addr, host.hAddrList, 4
'get all of the IP address if machine is multi-homed
lCpt = 0
Do
ReDim temp_ip_address(1 To host.hLength)
RtlMoveMemory temp_ip_address(1), hostip_addr, host.hLength

For i = 1 To host.hLength
ip_address = ip_address & temp_ip_address(i) & "."
Next
ip_address = Mid$(ip_address, 1, Len(ip_address) - 1)

ReDim lResult(lCpt)

```

```
lResult(lCpt) = ip_address
lCpt = lCpt + 1
ip_address = ""
host.hAddrList = host.hAddrList + LenB(host.hAddrList)
RtlMoveMemory hostip_addr, host.hAddrList, 4
Loop While (hostip_addr <> 0)
Gestion_Erreurs:
SocketsCleanup
GetIpFromHost = lResult
End Function
'-----
'
```

Il y a trois fonctions :

- - LetterToUNC qui transforme une lettre de disque en nom réseau
- - GetIpFromHost qui recherche les adresses IP d'un serveur
- - GetRemoteMACAddress qui recherche l'adresse MAC à partir d'une IP

Y a juste à rechercher la lettre du disque sur lequel est la base distante pour remplacer le "Q:" que j'ai mis en dur dans le code.

lien : [FAQ](#) Comment obtenir l'adresse MAC de la carte réseau ?

Sommaire > Bases de données

## Pourquoi j'ai l'erreur "type de données incompatible dans l'expression du critère" ?

Auteurs : Romain Puyfoulhoux ,

Cette erreur a lieu lorsque la valeur d'un critère dans la clause WHERE d'une requête Sql est mal écrite. Les valeurs spécifiées pour des champs de type texte doivent être entre apostrophes, et pour les champs de type date, entre "#".

```
SELECT * FROM documents WHERE ((titre like 'how-to%') AND (creation>#01/01/2002#) AND (auteur=10))
```

## Pourquoi ma requête, qui a un critère sur une date, ne me renvoie aucun enregistrement ?

Auteurs : Romain Puyfoulhoux ,

Parce que la date est lue au format mois/jour/année. Donc dans vos requêtes, les dates doivent être dans ce format. L'exemple suivant utilise la fonction Format() afin de formater la date correctement :

vb

```
sql= "SELECT * from documents WHERE (DateCreation >= #" & Format(dateSaisie,"mm/dd/yyyy") & "#")"
```

## Pourquoi j'ai une erreur de syntaxe dans ma requête quand un des arguments contient une apostrophe ?

Auteurs : Romain Puyfoulhoux ,

Parce que l'apostrophe doit être doublée sinon elle est considérée comme la fin de la chaîne. Voici une fonction qui double les apostrophes trouvées dans la chaîne passée en paramètre.

vb

```
Private Function DoubleQuote(ByVal chaine As String) As String  
  
DoubleQuote = Replace(chaine, "'", "''")  
  
End Function
```

## Comment spécifier les valeurs des paramètres d'une procédure stockée ?

Auteurs : Romain Puyfoulhoux ,

Les paramètres sont dans la collection Parameters de l'objet Command :

vb

```
Set cmd = New ADODB.Command  
cmd.CommandType = adCmdStoredProc  
cmd.CommandText = "procedure1"  
Set cmd.ActiveConnection = cnn 'cnn est un objet Adodb.Connection  
cmd.Parameters.Item("@auteur").Value = 10  
cmd.Parameters.Item("@type").Value = "how-to"
```


vb

```
cmd.Parameters.Item("@langage").Value = "vb"
```

## Qu'est-ce que MDAC et où puis-je le télécharger ?

**Auteurs :** [Romain Puyfoulhoux](#) ,

MDAC, qui signifie Microsoft Data Access Component, contient les composants d'accès aux bases de données que sont OLE DB, ADO, ODBC et RDS, ainsi que quelques pilotes et providers pour ODBC et OLEDB.

Le contenu des différentes versions et les liens pour les télécharger sont sur la page officielle :  <http://www.microsoft.com/downloads/>

**lien :** [Comment inclure MDAC à mon programme d'installation ?](#)

## Comment lire la structure d'une base de données avec ADO ?

**Auteurs :** [Jean-Marc Rabilloud](#) , [Romain Puyfoulhoux](#) ,

Pour lire la structure des bases on utilise la méthode `OpenSchema` de l'objet `Connection`. Cette méthode permet de lire des informations du schéma, soit globales, soit restreintes selon les paramètres passés.

Pour exécuter l'exemple suivant, vous devez sélectionner la référence "Microsoft ActiveX Data Object 2.X Library" ainsi que le composant "Microsoft Windows Common Controls 6.0". Déposez sur votre feuille un contrôle `TreeView` nommé "TreeView1", un bouton de commande nommé "Command1", et un textbox "Text1". Puis copiez le code ci-dessous dans le module de la form.

vb

```
Private Sub Command1_Click()

Dim MaConn As ADODB.Connection, rstTable As ADODB.Recordset
Dim rstEnfant As ADODB.Recordset, rstCompl As ADODB.Recordset
Dim cmpt1 As Long, cmpt2 As Long, NomTable As String, Indexed As Boolean
Dim cleNoeudTable As String, cleNoeudChamps As String, cleNoeudCles As String
Dim cleNoeudChampCourant As String, cleNoeudCleCourante As String

'création de la connexion
Set MaConn = New ADODB.Connection
MaConn.Provider = "Microsoft.Jet.OLEDB.4.0;"
MaConn.Open Text1.text

'création du recordset contenant la structure des tables
Set rstTable = New ADODB.Recordset
Set rstTable = MaConn.OpenSchema(adSchemaTables)

'Ajout de la racine du treeview
TreeView1.Nodes.Add , , "r", "Structure"
TreeView1.Nodes(1).Expanded = True

'Parcours de la collection des tables
cmpt1 = 1
Do While Not rstTable.EOF

    NomTable = rstTable!Table_Name

    'élimine les tables systèmes et les Vues
    If InStr(1, NomTable, "MSYS", vbTextCompare) <> 1 And rstTable!Table_Type <> "VIEW" Then
```

vb

```

'Ajout de la table au treeview
cleNoeudTable = "t" & cmpt1
TreeView1.Nodes.Add "r", tvwChild, cleNoeudTable, NomTable

'création du recordset des champs de la table
Set rstEnfant = New ADODB.Recordset
Set rstEnfant = MaConn.OpenSchema(adSchemaColumns, Array(Empty, Empty, NomTable, Empty))
cmpt2 = 1
cleNoeudChamps = cleNoeudTable & "ch"
cleNoeudCles = cleNoeudTable & "cl"
TreeView1.Nodes.Add cleNoeudTable, tvwChild, cleNoeudChamps, "champs"
TreeView1.Nodes.Add cleNoeudTable, tvwChild, cleNoeudCles, "clés"

Do While Not rstEnfant.EOF

    'ajoute au treeview le champ et ses caractéristiques
    cleNoeudChampCourant = cleNoeudChamps & cmpt2
    TreeView1.Nodes.Add cleNoeudChamps, tvwChild, cleNoeudChampCourant, rstEnfant!

COLUMN_NAME
    TreeView1.Nodes.Add cleNoeudChampCourant, tvwChild, cleNoeudChampCourant & "n", _
        "peut être NULL -> " & rstEnfant!IS_NULLABLE
    TreeView1.Nodes.Add cleNoeudChampCourant, tvwChild, cleNoeudChampCourant & "t", _
        "Type -> " & Switch(rstEnfant!DATA_TYPE = adInteger, "Long", _
            rstEnfant!DATA_TYPE = adSmallInt, "Entier",
            rstEnfant!DATA_TYPE = adWChar, "String")
    If Not IsNull(rstEnfant!CHARACTER_MAXIMUM_LENGTH) Then _
        TreeView1.Nodes.Add cleNoeudChampCourant, tvwChild, cleNoeudChampCourant & "l", _
            "Max Caractères -> " & rstEnfant!CHARACTER_MAXIMUM_LENGTH
    If Not IsNull(rstEnfant!NUMERIC_PRECISION) Then
        TreeView1.Nodes.Add cleNoeudChampCourant, tvwChild, cleNoeudChampCourant & "p", _
            "Précision -> " & rstEnfant!NUMERIC_PRECISION
        TreeView1.Nodes.Add cleNoeudChampCourant, tvwChild, cleNoeudChampCourant & "e", _
            "Echelle -> " & rstEnfant!NUMERIC_SCALE
    End If

    'vérifie si le champ est indexé
    Set rstCompl = New ADODB.Recordset
    Set rstCompl = MaConn.OpenSchema(adSchemaIndexes, Array(Empty, Empty, Empty, Empty,
NomTable))
    Indexed = False
    Do While Not rstCompl.EOF
        If rstCompl!COLUMN_NAME = rstEnfant!COLUMN_NAME Then
            Indexed = True
            TreeView1.Nodes.Add cleNoeudChampCourant, tvwChild, cleNoeudChampCourant & "i",
            "Indexé -> Oui" & Iif(rstCompl!
Unique, " sans ", " avec ") & "doublons"
            Exit Do
        End If
        rstCompl.MoveNext
    Loop
    If Not Indexed Then TreeView1.Nodes.Add cleNoeudChampCourant, tvwChild, _
        cleNoeudChampCourant & "i", "Indexé -> Non"
    rstCompl.Close
    Set rstCompl = Nothing
    cmpt2 = cmpt2 + 1
    rstEnfant.MoveNext
Loop
rstEnfant.Close
cmpt2 = 2

'recherche les clés de la tables
Set rstEnfant = MaConn.OpenSchema(adSchemaPrimaryKeys, Array(Empty, Empty, NomTable))
If Not rstEnfant.EOF Then
    'ajout de la clé primaire si elle existe
    TreeView1.Nodes.Add cleNoeudCles, tvwChild, cleNoeudTable & "cp", _

```



vb

```

                                "Clé primaire -> " & rstEnfant!COLUMN_NAME

End If
rstEnfant.Close

'Ajout des clés étrangères et de leurs caractéristiques
Set rstEnfant = MaConn.OpenSchema(adSchemaForeignKeys, _
                                Array(Empty, Empty, Empty, Empty, Empty, NomTable))

Do While Not rstEnfant.EOF
    cleNoeudCleCourante = cleNoeudCles & cmpt2
    TreeView1.Nodes.Add cleNoeudCles, tvwChild, cleNoeudCleCourante, _
                        "Clé Etrangère -> " & rstEnfant!FK_COLUMN_NAME
    TreeView1.Nodes.Add cleNoeudCleCourante, tvwChild, cleNoeudCleCourante & "cl", _
                        "Clé de la table -> " & rstEnfant!PK_TABLE_NAME
    TreeView1.Nodes.Add cleNoeudCleCourante, tvwChild, cleNoeudCleCourante & "ur", _
                        "Update Rules -> " & rstEnfant!UPDATE_RULE
    TreeView1.Nodes.Add cleNoeudCleCourante, tvwChild, cleNoeudCleCourante & "dr", _
                        "Delete Rules -> " & rstEnfant!DELETE_RULE

    cmpt2 = cmpt2 + 1
    rstEnfant.MoveNext

Loop
cmpt1 = cmpt1 + 1
rstEnfant.Close
Set rstEnfant = Nothing
End If
rstTable.MoveNext
Loop
TreeView1.Style = tvwTreelinesPlusMinusText
End Sub

```

Lancez le projet. Saisissez dans le textbox le chemin complet d'une base de données Access et cliquez sur le bouton de commande. Toute la structure de la base de données apparaît dans le treeview.

## Comment déterminer si une table existe avec ADO ?

**Auteurs : Romain Puyfoulhoux ,**

La fonction ci-dessous reçoit une connexion ouverte et le nom de la table à tester et renvoie Vrai si la table existe.

vb

```

Public Function TableExiste(cnn As ADODB.Connection, ByVal strTable As String) As Boolean

Dim rstTables As ADODB.Recordset

'création du recordset contenant la structure des tables
Set rstTables = cnn.OpenSchema(adSchemaTables)

'Parcours de la collection des tables
Do While Not rstTables.EOF
    If rstTables.fields("Table_Name").Value = strTable And
    rstTables.fields("Table_Type").Value <> "VIEW" Then
        TableExiste = True
        Exit Do
    End If
    rstTables.MoveNext
Loop
rstTables.Close

```

```
vb
End Function
```

## Pourquoi la propriété RecordCount me renvoie toujours -1 ?

Auteurs : **Romain Puyfoulhoux** ,

Parce qu'avec certains types de curseurs, la propriété Recordcount n'est pas valorisée automatiquement. Vous devez dans ce cas appeler tout d'abord la méthode movelast afin de vous placer sur le dernier enregistrement, puis appeler la méthode movefirst pour vous replacer sur le premier enregistrement, si nécessaire. Après un appel à movelast, recordcount contiendra la bonne valeur.

Pour éviter d'avoir à faire cette opération, privilégiez les curseurs "static" (cursorType = adOpenStatic) ou keyset (cursorType = adOpenKeyset). Enfin, si votre but est uniquement de déterminer si votre recordset contient des enregistrements ou pas, préférez l'utilisation de la propriété eof juste après l'ouverture du recordset.

## Comment connaître le nombre de lignes affectées par une requête UPDATE (ADO) ?

Auteurs : **Delphi-ne** ,

Après votre requête UPDATE, placez ce code :

```
vb
Dim rs as New Adodb.Recordset
rs.CursorLocation = adUseclient
Set rs = ObjetCommand.Execute(nNbRecordAffected)
```

## Comment créer un lien ODBC à partir de VB ?

Auteurs : **ThierryAIM** ,

Placez 3 CommandButton sur une Form et les codes respectifs suivants.  
Info : Me.hwnd est le handle de la Form appelant la configuration du DNS

```
vb
Private Const ODBC_ADD_DSN = 1
Private Const ODBC_CONFIG_DSN = 2
Private Const ODBC_REMOVE_DSN = 3

Private Declare Function SQLConfigDataSource Lib "ODBC32.DLL" (ByVal hwndParent As Long, _
    ByVal fRequest As Long, ByVal lpszDriver As String, _
    ByVal lpszAttributes As String) As Long

Private Sub Command1_Click()
    '---- Créer un DSN -----
    Dim strDriver As String
    Dim strAttributes As String
    Dim intRet As Long

    strDriver = "Microsoft Access Driver (*.mdb)" & Chr$(0)
    strAttributes = "DSN=MS Access Perso" & Chr$(0)
    strAttributes = strAttributes & "DESCRIPTION=Test DSN par VB6" & Chr$(0)
```

vb

```
strAttributes = strAttributes & "DBQ=c:\bd_test.mdb" & Chr$(0)
intRet = SQLConfigDataSource(vbNull, ODBC_ADD_DSN, strDriver, strAttributes)
If intRet Then
    MsgBox "DSN Created"
Else
    MsgBox "Create Failed"
End If

End Sub

Private Sub Command2_Click()
'---- Supprimer un DSN ----
Dim strDriver As String
Dim strAttributes As String
Dim intRet As Long

strDriver = "Microsoft Access Driver (*.mdb)" & Chr$(0)
strAttributes = "DSN=MS Access Perso" & Chr$(0)
intRet = SQLConfigDataSource(vbNull, ODBC_REMOVE_DSN, strDriver, strAttributes)
If intRet Then
    MsgBox "DSN Removed"
Else
    MsgBox "Remove Failed"
End If

End Sub

Private Sub Command3_Click()
'---- Configurer un DSN ----
Dim strDriver As String
Dim strAttributes As String
Dim intRet As Long

strDriver = "Microsoft Access Driver (*.mdb)" & Chr$(0)
strAttributes = "DSN=MS Access Perso" & Chr$(0)
intRet = SQLConfigDataSource(Me.hWnd, ODBC_CONFIG_DSN, strDriver, strAttributes)

End Sub
```

## Comment ouvrir la boîte de dialogue pour créer/éditer une connexion OleDb ?

Auteurs : Jean-Marc Rabilloud ,

Il faut ajouter à votre projet les références : - Microsoft OLE DB Service component 1.0 Type Library (oledb32.dll) - Microsoft ActiveX Data Object 2.x Library

Pour créer une nouvelle connexion :

vb

```
Private Sub Command1_Click()
Dim DataOLE As New MSDASC.DataLinks
Dim MaConn As ADODB.Connection

On Error GoTo Command1_Click_Error

Set MaConn = DataOLE.PromptNew
MsgBox MaConn.ConnectionString
Exit Sub
Command1_Click_Error:
MsgBox "Error " & Err.Number & " (" & Err.Description & ")"
Err.Clear
```

```
vb  
End Sub
```

Pour éditer une connexion existante :

```
vb  
  
Dim ret As Boolean  
ret = DataOLE.PromptEdit(MaConn) '-- MaConn est un objet ADODB.Connection  
MsgBox MaConn.ConnectionString
```

## Comment créer une base mdb, sans Access :

Auteurs : **ThierryAIM**,

En utilisant jet et ADOX (installés par défaut sous XP)

Cocher la référence : *Microsoft ADO Ext 2.x for DDL and Security*

```
Sub CreerBASE()  
'Creer une base ACCESS par ADOX  
Dim NewBase As ADOX.Catalog  
Dim NewTable As ADOX.Table  
Set NewBase = New ADOX.Catalog  
NewBase.Create "Provider=Microsoft.Jet.OLEDB.4.0;" & _  
    "Data Source=c:\base.mdb;" & _  
    "Jet OLEDB:Engine Type=5;"  
  
    Set NewTable = New ADOX.Table  
    With NewTable  
        .Name = "Table1"  
        .Columns.Append "Champ1", adInteger  
        .Columns("Champ1").Attributes = adColNullable  
        .Columns.Append "Champ2", adWChar, 50  
    End With  
    NewBase.Tables.Append NewTable  
    Set NewTable = Nothing  
    Set NewBase = Nothing  
End Sub
```

Attention il conviendra d'intégrer une gestion d'erreur.

## Comment compacter une base de données avec ADO ?

Auteurs : **Tofalu**,

Bien qu'ADO ne fournisse pas de méthode agissant sur la structure du fichier mdb, il est possible d'utiliser JRO (Jet Réplication Object)

Pour cela ajouter une référence Microsoft JRO à votre projet et utiliser la syntaxe suivante :

```
Dim jro As jro.JetEngine  
Set jro = New jro.JetEngine  
jro.CompactDatabase "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=d:  
\MaBase.mdb;Jet OLEDB:Database Password=test", _
```

```
"Provider=Microsoft.Jet.OLEDB.4.0;Data Source=d:\MaBaseCompactee.mdb;Jet OLEDB:Engine Type=4;Jet OLEDB:Database Password=test"
```

## Comment créer une base mdb, en DAO :

Auteurs : bbil ,

En utilisant DAO 3.6

Cocher la référence : *Microsoft DAO 3.6 Object Library*

```
Sub CreerBaseDAO()  
    Dim oDAO36 As New DAO.DBEngine  
    Dim oBase As DAO.Database  
    Dim oTable As DAO.TableDef  
    'Création base  
    Set oBase = oDAO36.CreateDatabase("c:\tmp\Mabase.MDB", ";LANGID=0x0409;CP=1252;COUNTRY=0", 64)  
    'Création d'une table  
    Set oTable = oBase.CreateTableDef("Table1")  
    oTable.Fields.Append oTable.CreateField("Champ1", dbInteger)  
    oTable.Fields.Append oTable.CreateField("Champ2", dbText, 50)  
    oBase.TableDefs.Append oTable  
    oBase.Close  
End Sub
```

lien :  [Définition et manipulation de données avec DAO](#)

## Comment afficher les tables et champs d'une base de données ?

Auteurs : sovo ,

Ce code permet d'afficher toutes les tables et champs d'une base de données, ceci peut être pratique si l'on veut connaître sa base.

Ce code fonctionne 100 % avec des bases ACCESS.

Rajoutez la référence à *référence Microsoft ActiveX Data Object 2.X*

```
'  
' Utilise la référence Microsoft ActiveX Data Object 2.X ..  
'  
'Tout d'abord il faut deux ListBox  
'Tables : qui va avoir toutes les Tables  
'Champs : qui va avoir tous les champs d'une table donne  
'  
Dim CNX As ADODB.Connection  
Dim Schema As ADODB.Recordset  
Dim DBPath As String  
Private Sub Form_Load()  
    DBPath = "c:\tmp\mabase.mdb"  
    'tout d'abord il faut bien évidemment ouvrir une connexion avec la base de donne  
    ACS_Connect  
    'Affiche les tables  
    ViewTables  
End Sub  
'Connexion avec une base de donnee  
Public Sub ACS_Connect()  
    Set CNX = New ADODB.Connection  
    CNX.Provider = "Microsoft.Jet.Oledb.4.0"  
    CNX.ConnectionString = DBPath 'Le de ta base de donnee  
    CNX.Open  
End Sub
```

```
'Recuperation et affichage des tables dans Tables, cette procedure
Public Sub ViewTables()

    Set Schema = CNX.OpenSchema(adSchemaTables)
    Do Until Schema.EOF
        If Schema!TABLE_TYPE = "TABLE" Then
            Tables.AddItem Schema!TABLE_NAME
        End If
        Schema.MoveNext
    Loop
    Schema.Close
End Sub

'Maintenant lorsqu'on clique sur une table (dans le listbox Tables), on affiche les champs
'dans le Listbox Champs
Private Sub Tables_Click()
    Champs.Clear
    Set Schema = CNX.OpenSchema(adSchemaColumns)

    While Not Schema.EOF
        If Schema!TABLE_NAME = Tables.Text Then
            Champs.AddItem Schema!COLUMN_NAME
        End If
        Schema.MoveNext
    Wend
    Schema.Close
End Sub
```

## Sommaire > Documentation et installation

### Comment ouvrir mon fichier d'aide .hlp depuis mon application ?

Auteurs : **Romain Puyfoulhoux** ,

Lorsque l'utilisateur de votre programme appuie sur la touche F1, VB ouvre le fichier d'aide dont le chemin est spécifié dans App.HelpFile.

Si le fichier d'aide se trouve dans le répertoire de l'exécutable, et s'appelle Aide.hlp, il faudra donc placer cette ligne au début du programme :

vb

```
App.HelpFile = App.Path & "\Aide.hlp"
```

Si le fichier d'aide doit aussi s'ouvrir sur le clic d'un bouton ou d'un menu, il suffira de simuler un clic sur la touche F1 avec l'instruction SendKeys :

vb

```
Private Sub btnAide_Click()  
  
'ouvre le fichier d'aide quand on clique sur le bouton btnAide  
SendKeys "{F1}"  
  
End Sub
```

### Comment ouvrir mon fichier d'aide .chm par du code ?

Auteurs : **ThierryAIM** ,

Si votre fichier d'aide est un fichier d'extension .chm, vous pouvez déclencher son ouverture lors de l'appui sur la touche F1 de la même façon que celle décrite Comment ouvrir mon fichier d'aide .hlp depuis mon application ?.

Si vous avez besoin de passer par le code, voici le source d'un module de classe à copier dans votre projet. Créez donc un module de classe et nommez-le CHelp.

vb

```
Private Enum HH_COMMAND  
    HH_DISPLAY_TOPIC = &H0  
    HH_HELP_FINDER = &H0  
    HH_DISPLAY_TOC = &H1  
    HH_DISPLAY_INDEX = &H2  
    HH_DISPLAY_SEARCH = &H3  
    HH_SET_WIN_TYPE = &H4  
    HH_GET_WIN_TYPE = &H5  
    HH_GET_WIN_HANDLE = &H6  
    HH_GET_INFO_TYPES = &H7  
    HH_SET_INFO_TYPES = &H8  
    HH_SYNC = &H9  
    HH_ADD_NAV_UI = &HA  
    HH_ADD_BUTTON = &HB  
    HH_GETBROWSER_APP = &HC  
    HH_KEYWORD_LOOKUP = &HD  
    HH_DISPLAY_TEXT_POPUP = &HE  
    HH_HELP_CONTEXT = &HF  
    HH_TP_HELP_CONTEXTMENU = &H10  
    HH_TP_HELP_WM_HELP = &H11  
    HH_CLOSE_ALL = &H12  
    HH_ALINK_LOOKUP = &H13
```

```

vb
End Enum

Private Type structHH_FTS_QUERY
    cbStruct As Long
    fUnicodeStrings As Long
    pszSearchQuery As String
    iProximity As Long
    fStemmedSearch As Long
    fTitleOnly As Long
    fExecute As Long
    pszWindow As String
End Type

Private Declare Function HtmlHelp Lib "hhctrl.ocx" Alias "HtmlHelpA" _
    (ByVal hwndCaller As Long, _
    ByVal pszFile As String, _
    ByVal uCommand As HH_COMMAND, _
    dwData As Any) As Long

Public Sub Show(NewFile As String, Optional WindowPane As String, Optional ContextID)
    Dim Fichier As String
    Fichier = NewFile

    If Len(WindowPane) Then
        Fichier = Trim(Fichier) & ">" & Trim(WindowPane)
    End If

    If IsMissing(ContextID) Then
        Call HtmlHelp(0, Fichier, HH_DISPLAY_TOC, ByVal 0&)
    Else
        Call HtmlHelp(0, Fichier, HH_HELP_CONTEXT, ByVal CInt(ContextID))
    End If
End Sub

Public Sub ShowIndex(NewFile As String, Optional WindowPane As String)
    Dim Fichier As String
    Fichier = NewFile

    If Len(WindowPane) Then
        Fichier = Trim(Fichier) & ">" & Trim(WindowPane)
    End If

    Call HtmlHelp(0, Fichier, HH_DISPLAY_INDEX, ByVal 0&)
End Sub

Public Sub ShowSearch(NewFile As String, Optional WindowPane As String)

    Dim Fichier As String
    Dim HH_FTS_QUERY As structHH_FTS_QUERY
    With HH_FTS_QUERY
        .cbStruct = Len(HH_FTS_QUERY)
        .fUnicodeStrings = 0&
        .pszSearchQuery = "TEST"
        .iProximity = 0&
        .fStemmedSearch = 0&
        .fTitleOnly = 0&
        .fExecute = 1&
        .pszWindow = ""
    End With
    Fichier = NewFile

    If Len(WindowPane) Then
        Fichier = Trim(Fichier) & ">" & Trim(WindowPane)
    End If
    Call HtmlHelp(0, Fichier, HH_DISPLAY_SEARCH, HH_FTS_QUERY)
End Sub

```



**App.HelpFile** vous renvoie le nom du fichier d'aide que vous avez indiqué dans les propriétés du projet. Vous pouvez aussi modifier cette propriété à l'exécution. **App.HelpFile** doit bien sûr être ici un fichier **.chm** compilé.

Voyons comment afficher l'index de l'aide dans la fenêtre **W1**, celle-ci ayant été auparavant définie dans le projet d'aide. Le paramétrage de la fenêtre d'affichage est optionnel.

vb

```
Dim objHelp As Chelp
Set objHelp = New Chelp
Call objHelp.ShowIndex(App.HelpFile, "W1")
Set objHelp = Nothing
```

Nous pouvons aussi afficher le sommaire :

vb

```
Dim objHelp As Chelp
Set objHelp = New Chelp
Call objHelp.Show(App.HelpFile, "W1")
Set objHelp = Nothing
```

Ainsi que la page de recherche :

vb

```
Dim objHelp As Chelp
Set objHelp = New Chelp
Call objHelp.ShowSearch(App.HelpFile, "W1")
Set objHelp = Nothing
```

Il n'est pas possible de transmettre la chaîne à rechercher directement à l'onglet de recherche, comme l'indique cet article de Microsoft

Enfin voici comment afficher une page à partir de son **ContextID** :

vb

```
Dim objHelp As Chelp
Set objHelp = New Chelp
Call objHelp.Show (App.HelpFile, "W1", 320)
Set objHelp = Nothing
```

## Comment faire un programme d'installation ?

**Auteurs : Romain Puyfoulhoux ,**

Vous pouvez utiliser l'assistant d'emballage et de déploiement. Vous le trouverez dans le gestionnaire des suppléments de Visual Basic, ou dans les programmes du menu Démarrer, dans les outils Microsoft Visual Studio 6.0.

Il existe aussi des logiciels commerciaux complets et de bonne qualité, tels ceux édités par les sociétés **Installshield** et **Wise**.

Et enfin quelques outils gratuits :

- **Inno Setup** : vous permet de créer un programme d'installation personnalisé
- **IsTool** : interface graphique pour Inno Setup, permet aussi de convertir un fichier setup.lst créé par l'Assistant d'Empaquetage en fichier iss pour Inno Setup
- **Visual Studio Installer 1.1** : créateur de programme d'installation, n'existe pas en français

## Quels sont les fichiers nécessaires pour que mon logiciel fonctionne ?

**Auteurs : Romain Puyfoulhoux ,**

**Visual Basic 3.0**

**VBRUN300.DLL**

**Visual Basic 4.0**

**VB40032.DLL, OLEPRO32.DLL**

**Visual Basic 5.0**

**MSVBVM50.DLL, OLEAUT32.DLL, OLEPRO32.DLL, STDOLE2.TLB, ASYCFILT.DLL, COMCAT.DLL**

**Visual Basic 6.0**

**MSVBVM60.DLL, OLEAUT32.DLL, OLEPRO32.DLL, STDOLE2.TLB, ASYCFILT.DLL, COMCAT.DLL**

Les versions des fichiers distribués dans les différents Service Packs de chaque produit sont souvent différentes de celles livrées dans la version d'origine.

Ces fichiers constituent le runtime Visual Basic de chaque version. Mais dans votre programme d'installation il faut ajouter les éventuels ActiveX et références que vous avez inclus dans votre projet. Si vous utilisez ADO, il faut inclure MDAC.

## Comment enregistrer un ActiveX ?

**Auteurs : Romain Puyfoulhoux ,**

Un ActiveX doit figurer dans la base de registre afin qu'il soit pris en compte par le système et qu'il puisse fonctionner. Ceci est pris en charge par le programme regsvr32.exe qui se trouve dans le répertoire système.

Pour enregistrer un ActiveX :

```
vb
```

```
regsvr32.exe <fichier>
```

Pour l'opération inverse :

```
vb
```

```
regsvr32.exe /u <fichier>
```

## Pourquoi le setup de VB m'indique que des fichiers systèmes sont périmés pendant l'installation de mon application ?

Auteurs : **Romain Puyfoulhoux** ,

Ce message a lieu si un ou plusieurs fichiers systèmes à mettre à jour sont chargés en mémoire par Windows au démarrage du système. Le fichier étant chargé en mémoire, Windows doit avoir redémarré pour que la version incluse dans votre setup soit chargée à la place de la version actuelle. Pour éviter ce désagrément, regardez quels sont les fichiers présents dans la section [bootstrap files] du fichier setup.lst. Vous pouvez fournir dans votre setup les versions de ces fichiers qui sont livrées avec VB6, plutôt que les mises à jour qui ont été installées sur votre système. En effet, plus les versions que vous fournirez seront récentes, plus les postes clients devant redémarrer lors de l'installation de votre programme seront nombreux. Les versions incluses dans votre setup sont celles qui sont présentes dans votre répertoire système.

Vérifiez aussi dans le fichier setup.lst que le fichier msvcr7.dll n'est pas dans la section [setup1 files] mais [bootstrap files], et déplacez-le dans cette section le cas échéant.

## Comment inclure MDAC à mon programme d'installation ?

Auteurs : **Romain Puyfoulhoux** ,

Si vous utilisez l'assistant d'emballage et de déploiement, ajoutez le fichier Mdac\_typ.exe à votre paquetage, et choisissez \$(AppPath) pour son répertoire de destination. Lors de l'installation, le setup exécutera Mdac\_typ.exe avant d'installer vos fichiers.

Pour pouvoir installer MDAC sous Windows 95 et Windows 98, vous devez d'abord avoir installé respectivement DCOM95 et DCOM98.

## Comment associer une extension à un programme ?

Auteurs : **Jean-Marc Rabilloud** , **Khorne** ,

Le principe repose sur la création de plusieurs clés dans la base de registres. Il existe deux méthodes : l'utilisation du Windows Script Host Object Model et celle des API Windows.

Avec Wshom.ocx :

```
vb
```

```
Dim MaCle As WshShell 'ou As object

Set MaCle = New WshShell 'ou Set MaCle = CreateObject("WScript.Shell")
'nom de votre type de fichier
MaCle.RegWrite "HKEY_CLASSES_ROOT\Test Ext\", "Test Ext", "REG_SZ"
'commande à exécuter pour ouvrir les fichiers de ce type
MaCle.RegWrite "HKEY_CLASSES_ROOT\Test Ext\shell\open\command\", "C:\jmarc\FAQ
\TestExt.exe %1", "REG_SZ"
'icone à utiliser pour représenter les fichiers de ce type, ici la troisième icone contenue dans TestExt.exe
'(la première icone a l'index 0)
```

vb

```

MaCle.RegWrite "HKEY_CLASSES_ROOT\Test Ext\DefaultIcon\", "C:\jmarc\FAQ\TestExt.exe,2", "REG_SZ"
""
'extension correspondant à ce type de fichier
MaCle.RegWrite "HKEY_CLASSES_ROOT\.jmr\", "Test Ext", "REG_SZ"

```

Avec les anciennes versions de Wshom.ocx, la classe WshShell s'appelle IWshShell\_Class.

Avec les API :

vb

```

Private Declare Function RegCreateKey Lib "advapi32.dll" Alias "RegCreateKeyA" _
    (ByVal hKey As Long, ByVal lpSubKey As String, _
    phkResult As Long) As Long
Private Declare Function RegSetValue Lib "advapi32.dll" Alias "RegSetValueA" _
    (ByVal hKey As Long, ByVal lpSubKey As String, _
    ByVal dwType As Long, ByVal lpData As String, _
    ByVal cbData As Long) As Long

Private Const HKEY_CLASSES_ROOT = &H80000000
Private Const MAX_PATH = 255
Private Const REG_SZ = 1

Private Sub cmdAssocExt_Click()

Dim MaCle As Long

Call RegCreateKey(HKEY_CLASSES_ROOT, "Test Ext", MaCle)
Call RegSetValue&(MaCle, "", REG_SZ, "Test Ext", 0&)
Call RegSetValue&(MaCle, "shell\open\command", REG_SZ, "C:\jmarc\FAQ\TestExt.exe %1", MAX_PATH)
Call RegSetValue&(MaCle, "DefaultIcon", REG_SZ, "C:\jmarc\FAQ\TestExt.exe,2", MAX_PATH)
Call RegCreateKey&(HKEY_CLASSES_ROOT, ".jmr", MaCle)
Call RegSetValue&(MaCle, "", REG_SZ, "Test Ext", 0&)

End Sub

```

Dans les deux cas, tout fichier ayant une extension ".jmr" sera ouvert avec le programme TestExt.exe.

Un autre exemple de programmer ces actions :

vb

```

Private Declare Sub SHChangeNotify Lib "shell32.dll" ( _
ByVal wEventId As Long, _
ByVal uFlags As Long, _
dwItem1 As Any, _
dwItem2 As Any)

Private Const SHCNE_ASSOCCHANGED = &H80000000
Private Const SHCNF_IDLIST = &H0&

Private Function Associer(AdApp As String, AdIcon As String, Extention As Variant,
NomDuFichier As String) As Boolean

On Error GoTo F

Set WshShell = CreateObject("Wscript.Shell")

For v = LBound(Extention, 1) To UBound(Extention, 1)
WshShell.RegWrite "HKEY_CLASSES_ROOT\" & Extention(v) & "\", NomDuFichier, "REG_SZ"
Next v

```

vb

```
AdSp = "HKEY_CLASSES_ROOT\" & NomDuFichier & "\"
WshShell.RegWrite AdSp, "Khorne File Crypted", "REG_SZ"
WshShell.RegWrite AdSp & "DefaultIcon\", AdIcon, "REG_SZ"
WshShell.RegWrite AdSp & "Shell\open\command\", Chr(34) & AdApp & Chr(34) & " %1", "REG_SZ"
SHChangeNotify SHCNE_ASSOCCHANGED, SHCNF_IDLIST, 0, 0
Associer = True

F:

End Function

Private Sub Form_Load()
    Insérer
End Sub

Sub Insérer()
    ExtentionAMettreEnRelation = Array("kh1", "kh2", "kh4")
    CheminDeLAppli$ = "d:\sp\Khorne.exe"
    CheminDeLicone$ = "d:\sp\Khorne.ico"
    NomDuGenreDeFichier$ = "Image Cryptée Spécifique"
    If Associer(CheminDeLAppli$, CheminDeLicone$, ExtentionAMettreEnRelation, NomDuGenreDeFichier
$) Then
        MsgBox "Changement réussi.", vbInformation, "Super ça a marché!!!"
    Else
        If
(MsgBox("Ca n'a pas marché car l'un des paramètres insérés est mauvais.", vbCritical + vbYesNo, "Erreur") = vb
Insérer
        End If
    End Sub
```



Sommaire > Liaison Office > Excel

## Comment utiliser Excel en VB6 ?

Auteurs : bbil ,

Rajoutez, au projet la référence "Microsoft Excel xx.x Object library"  
(menu projet Référence)  
puis utilisez par exemple CreateObject pour créer l'objet Microsoft Excel :

### Ouvrir Excel

```
Private Sub CdOuvrirExcel_Click()  
Dim oExcel As Excel.Application  
Set oExcel = CreateObject("Excel.Application")  
oExcel.Visible = True 'Affiche l'application excel  
MsgBox "Excel est ouvert"  
oExcel.Quit  
Set oExcel = Nothing 'libération mémoire..  
End Sub
```

lien :  [Utiliser Excel à partir de Visual Basic](#)

## Comment créer un classeur Excel en VB6 ?

Auteurs : bbil ,

Rajoutez, au projet la référence "Microsoft Excel xx.x Object library"  
puis utilisez la méthode Add de la collection Workbooks

### Créer classeur Excel

```
Private Sub cdCreerClasseur_Click()  
Dim oExcel As Excel.Application  
Dim oWk As Workbook  
Set oExcel = CreateObject("Excel.Application")  
oExcel.Visible = False 'Masque l'application excel (valeur par défaut)  
Set oWk = oExcel.Workbooks.Add  
'rajoute par exemple la date et heure actuelle en Feuille cellule A1  
oWk.Sheets(1).Range("A1") = Now  
'Sauve le classeur  
oExcel.DisplayAlerts = False ' Pour éviter des questions si classeur déjà existant  
oWk.SaveAs App.Path & "\MonClasseur.xls"  
oWk.Close False 'Ferme le classeur  
oExcel.Quit  
Set oWk = Nothing  
Set oExcel = Nothing 'libération mémoire..  
End Sub
```

lien :  [Utiliser Excel à partir de Visual Basic](#)

## Comment modifier un classeur ?

Auteurs : bbil ,

Rajoutez, au projet la référence "Microsoft Excel xx.x Object library"  
puis utilisez la méthode Open de la collection Workbooks, pour ouvrir le classeur avant de le modifier

### Modifier un classeur existant

```

Private Sub cdModifClasseur_Click()
    Dim oExcel As Excel.Application
    Dim oWk As Workbook
    Set oExcel = CreateObject("Excel.Application")
    oExcel.Visible = False 'Masque l'application excel (valeur par défaut)
    On Error Resume Next 'Pour éviter les erreur si classeur n'existe pas
    Set oWk = oExcel.Workbooks.Open(App.Path & "\MonClasseur.xls")
    On Error GoTo 0
    If oWk Is Nothing Then
        MsgBox "Erreur sur ouverture classeur", vbCritical
        Exit Sub
    End If
    'rajoute par exemple la date et heure actuelle sur la dernière ligne colonne A.
    oWk.Sheets(1).Range("A65535").End(xlUp).Offset(1, 0) = Now
    'Sauve le classeur
    oWk.save
    oWk.Close False 'Ferme le classeur
    oExcel.Quit
    Set oWk = Nothing
    Set oExcel = Nothing 'libération mémoire..
End Sub

```

lien :  [Utiliser Excel à partir de Visual Basic](#)

### Comment ajouter une feuille dans un classeur ?

Auteurs : [bbil](#),

Rajoutez, au projet la référence "Microsoft Excel xx.x Object library"  
 une fois le classeur ouvert ( voir [FAQ](#) Comment modifier un classeur ? ) utilisez le code suivant pour l'insertion de la feuille

#### Ajouter une feuille à un classeur existant

```

...
Set oSh = oWk.Worksheets.Add
oSh.Name = "Mafeuille" 'Renomme la feuille
oSh.Range("C3") = "Agit sur la feuille"
'===== fin ajout feuille
Set oSh = Nothing ' libération mémoire...

```

lien : [FAQ](#) Comment modifier un classeur ?

### Comment exécuter une Macro (sub ..) excel depuis VB6 ?

Auteurs : [bbil](#),

Rajoutez, au projet la référence "Microsoft Excel xx.x Object library"  
 Le classeur doit contenir la procédure publique à exécuter.

```

Private Sub cdLancerMacro_Click()
    Dim oExcel As Excel.Application
    Dim oWk As Workbook
    Set oExcel = CreateObject("Excel.Application")
    oExcel.Visible = True
    Set oWk = oExcel.Workbooks.Open(App.Path & "\MonClasseur.xls")
    On Error GoTo 0

```



```
If oWk Is Nothing Then
    MsgBox "Erreur sur ouverture classeur", vbCritical
    Exit Sub
End If

oExcel.Run "MaMacro" ' lance la macro

Set oWk = Nothing
Set oExcel = Nothing 'libération mémoire..
End Sub
```

Sommaire > Liaison Office > Word

## Comment connaître la version de Word installée ?

Auteurs : **SilkyRoad** ,

**Vous devez ajouter la référence à *Microsoft Word xx.x library* à votre projet (Menu Projet >> Références...).**

vb

```
Private Function InfoWordVersion() As String
    Dim objWord As Word.Application
    Set objWord = CreateObject("Word.Application")    '-- ouvrir une session Word
    InfoWordVersion = "Version: " & objWord.Version & vbCrLf & _
        "Build: " & objWord.Build & vbCrLf & "Product Code: " & objWord.ProductCode()
    objWord.Quit    '-- fermer la session Word
    Set objWord = Nothing    '-- détruire l'objet Word
End Function

Private Sub Form_Load()
    MsgBox InfoWordVersion
End Sub
```

## Comment savoir si Word est déjà ouvert ?

Auteurs : **SilkyRoad** , **ThierryAIM** ,

**Vous devez ajouter la référence à *Microsoft Word xx.x library* à votre projet (Menu Projet >> Références...).**  
**Cette fonction booléenne renvoie "Vrai" si Word est déjà ouvert, sinon, renvoie "Faux"**

vb

```
Private Function IsWordOpen() As Boolean
    Dim objWord As Word.Application
    On Error Resume Next
    Set objWord = GetObject(, "Word.Application")
    IsWordOpen = Not objWord Is Nothing
    Set objWord = Nothing    '-- détruire l'objet Word
End Function
```

## Comment ouvrir Word ou un fichier Word avec OLE ?

Auteurs : **Romain Puyfoulhoux** ,

**Vous devez ajouter la référence à *Microsoft Word xx.x library* à votre projet (Menu Projet >> Références...).**  
**Ce premier exemple ouvre un fichier Word, affiche son texte avec MsgBox, et ferme le fichier.**

vb

```
Dim DocWord As Word.Document

Set DocWord = GetObject("c:\mes documents\Article.doc")
DocWord.Activate
MsgBox DocWord.Range.Text
DocWord.Close False
Set DocWord = Nothing
```

**Ce deuxième exemple ouvre Word, crée un nouveau document, et rend la fenêtre de l'application visible.**

vb

```
Dim objWord As Word.Application
Dim docWord As Word.Document

Set objWord = CreateObject("Word.Application")    '-- ouvrir une session Word

Set docWord = objWord.Documents.Add
objWord.Visible = True    '-- voir l'application Word
docWord.SaveAs sDoc    '-- enregistrer le nouveau document
Set objWord = Nothing    '-- détruire l'objet Word
```

## Comment créer un nouveau document Word ?

**Auteurs : ThierryAIM ,**

Vous devez ajouter la référence à *Microsoft Word xx.x library* à votre projet (Menu Projet >> Références...).

vb

```
Private Sub CreateNewDocWord(sDoc As String)
    Dim objWord As Word.Application
    Dim docWord As Word.Document
    Dim Fichier As String

    Set objWord = CreateObject("Word.Application")    '-- ouvrir une session Word
    Set docWord = objWord.Documents.Add    '-- Ajouter un nouveau document à la collection
    objWord.Visible = True    '-- montrer l'application Word
    docWord.SaveAs FileName:=sDoc

    Set docWord = Nothing    '-- détruire l'objet Document
    Set objWord = Nothing    '-- détruire l'objet Word
End Sub
```

**Exemple :**

vb

```
Private Sub Command1_Click()
    CreateNewDocWord "c:\MonNouveauDocument.doc"
End Sub
```

## Comment imprimer un document Word ?

**Auteurs : SilkyRoad ,**

Vous devez ajouter la référence à *Microsoft Word xx.x library* à votre projet (Menu Projet >> Références...).

vb

```
Private Sub PrintDocWord(sDoc As String)
    Dim objWord As Word.Application
    Dim docWord As Word.Document
    Dim Fichier As String

    Set objWord = CreateObject("Word.Application")    '-- ouvrir une session Word
    objWord.Visible = False    '-- masquer l'application Word
    Set docWord = objWord.Documents.Open(sDoc)    '-- ouvrir le document Word
```

```

vb
docWord.PrintOut    '-- imprimer le document

docWord.Close      '-- fermer le document Word
objWord.Quit       '-- fermer la session Word
Set docWord = Nothing    '-- détruire l'objet Document
Set objWord = Nothing    '-- détruire l'objet Word
End Sub

```

**Exemple :**

```

vb
Private Sub Command1_Click()
    CommonDialog1.Filter = "Fichiers Word (*.doc)|*.doc"
    CommonDialog1.ShowOpen
    PrintDocWord CommonDialog1.FileName
End Sub

```

## Comment lister les propriétés d'un document Word ?

**Auteurs :** SilkyRoad , ThierryAIM ,

**Vous devez ajouter la référence à *Microsoft Word xx.x library* à votre projet (Menu Projet >> Références...).**

```

vb
Private Sub ProprietesDocWord(sDoc As String)
    Dim objWord As Word.Application
    Dim docWord As Word.Document
    Dim Propriete As Object

    On Error GoTo ProprietesDocWord_Error

    Set objWord = CreateObject("Word.Application")    '-- ouvrir une session Word
    objWord.Visible = False
    Set docWord = objWord.Documents.Open(sDoc)    '-- ouvrir le document Word

    '-- liste toutes les propriétés et leur valeur dans la fenêtre de débogage
    For Each Propriete In docWord.builtinDocumentProperties
        On Error Resume Next
        Debug.Print "Nom : " & Propriete.Name & " = " & Propriete.Value
    Next

    docWord.Close    '-- fermer le document Word
    objWord.Quit    '-- fermer la session Word
    '    objWord.Quit SaveChanges:=wdDoNotSaveChanges
    Set docWord = Nothing    '-- détruire l'objet Document
    Set objWord = Nothing    '-- détruire l'objet Word
    Exit Sub

ProprietesDocWord_Error:
    MsgBox "Error " & Err.Number & " (" & Err.Description & ") in procedure ProprietesDocWord"
    Err.Clear
    Set docWord = Nothing    '-- détruire l'objet Document
    Set objWord = Nothing    '-- détruire l'objet Word
End Sub

```

**Exemple :**

vb

```
Private Sub Command1_Click()  
    CommonDialog1.Filter = "Fichiers Word (*.doc)|*.doc"  
    CommonDialog1.ShowOpen  
    compterNombrePagesDocWord CommonDialog1.FileName  
End Sub
```

Vous trouverez un exemple d'utilisation des propriétés d'un document dans le lien ci-dessous

lien : [FAQ](#) Comment connaître le nombre de pages d'un document Word ?

## Comment connaître le nombre de pages d'un document Word ?

Auteurs : SilkyRoad ,

Vous devez ajouter la référence à *Microsoft Word xx.x library* à votre projet (Menu Projet >> Références...).

vb

```
Private Function NombrePagesDocWord(sDoc As String) As Integer  
    Dim objWord As Word.Application  
    Dim docWord As Word.Document  
  
    On Error GoTo compterNombrePagesDocWord_Error  
  
    Set objWord = CreateObject("Word.Application")    '-- ouvrir une session Word  
    Set docWord = objWord.Documents.Open(sDoc)      '-- ouvrir le document Word  
  
    With docWord  
        NombrePagesDocWord = .builtinDocumentProperties("Number of Pages")  
    End With  
  
    docWord.Close    '-- fermer le document Word  
    objWord.Quit     '-- fermer la session Word  
    Set docWord = Nothing    '-- détruire l'objet Document  
    Set objWord = Nothing    '-- détruire l'objet Word  
    Exit Function  
  
compterNombrePagesDocWord_Error:  
  
    MsgBox "Error " & Err.Number & " (" & Err.Description & ") in procedure compterNombrePagesDocWord"  
    Err.Clear  
    Set docWord = Nothing    '-- détruire l'objet Document  
    Set objWord = Nothing    '-- détruire l'objet Word  
End Function
```

Exemple :

vb

```
Private Sub Command1_Click()  
    CommonDialog1.Filter = "Fichiers Word (*.doc)|*.doc"  
    CommonDialog1.ShowOpen  
    DoEvents  
    MsgBox "Il y a " & compterNombrePagesDocWord(CommonDialog1.FileName) & " page(s) _  
dans le document Word : " & CommonDialog1.FileTitle
```

```
vb  
End Sub
```

lien : [FAQ Comment lister les propriétés d'un document Word ?](#)

## Comment désactiver la correction orthographique de Word ?

Auteurs : [argyronet](#) ,

Vous devez ajouter la référence à *Microsoft Word xx.x library* à votre projet (Menu Projet >> Références...).  
Après avoir instancié Word dans votre code, vous pouvez appeler une procédure codée comme suit :

```
vb  
  
Sub DesactiverCorrectionDocument(ByVal oWord As Word.Application)  
Const PERSO_DIC_PATH As String = "C:\Documents and Settings\UserName\Application Data\Microsoft  
\Épreuve\PERSO.DIC"  
  
With oWord  
    .Options.CheckSpellingAsYouType = False  
    .Options.CheckGrammarAsYouType = False  
    .Options.SuggestSpellingCorrections = True  
    .Options.SuggestFromMainDictionaryOnly = False  
    .Options.CheckGrammarWithSpelling = False  
    .Options.ShowReadabilityStatistics = True  
    .Options.IgnoreUppercase = True  
    .Options.IgnoreMixedDigits = True  
    .Options.IgnoreInternetAndFileAddresses = True  
    .Options.AllowCombinedAuxiliaryForms = True  
    .Options.EnableMisusedWordsDictionary = True  
    .Options.AllowCompoundNounProcessing = True  
    .Options.UseGermanSpellingReform = False  
    .ActiveDocument.ShowGrammaticalErrors = True  
    .ActiveDocument.ShowSpellingErrors = True  
    .Languages(wdFrench).SpellingDictionaryType = wdSpelling  
    .Languages(wdFrench).DefaultWritingStyle = "Vérification rapide"  
    .ActiveDocument.ActiveWritingStyle(wdFrench) = "Vérification rapide"  
    .CustomDictionaries.ClearAll  
    .CustomDictionaries.Add(PERSO_DIC_PATH).LanguageSpecific = False  
    .CustomDictionaries.ActiveCustomDictionary = CustomDictionaries.Item(PERSO_DIC_PATH)  
End With  
End Sub
```

## Comment modifier les marges d'un document Word, exprimées en cm ?

Auteurs : [ThierryAIM](#) ,

Vous devez ajouter la référence à *Microsoft Word xx.x library* à votre projet (Menu Projet >> Références...).  
Cette fonction reçoit en paramètres optionnels :

- le nom du document (si ce paramètre est omis, un nouveau document est créé)
- les valeurs des marges à définir (seules les valeurs spécifiées seront modifiées)

```
vb  
  
Private Function MargesDocWord(Optional sDoc As String = "", Optional lMargin, Optional rMargin, _  
Optional tMargin, Optional bMargin) As Boolean  
    Dim objWord As Word.Application  
    Dim docWord As Word.Document  
  
    On Error GoTo MargesDocWord_Error
```

vb

```

Set objWord = CreateObject("Word.Application")      '-- ouvrir une session Word

If sDoc = "" Then
    Set docWord = objWord.Documents.Add          '-- créer un nouveau document Word
Else
    Set docWord = objWord.Documents.Open(sDoc)   '-- ouvrir le document Word
End If
objWord.Visible = True

With docWord.PageSetup
    If Not IsMissing(lMargin) Then .LeftMargin = objWord.CentimetersToPoints(lMargin)
    If Not IsMissing(rMargin) Then .RightMargin = objWord.CentimetersToPoints(rMargin)
    If Not IsMissing(tMargin) Then .TopMargin = objWord.CentimetersToPoints(tMargin)
    If Not IsMissing(bMargin) Then .BottomMargin = objWord.CentimetersToPoints(bMargin)
End With

MargesDocWord = True
Exit Function

MargesDocWord_Error:
MsgBox "Error " & Err.Number & " (" & Err.Description & ") in procedure MargesDocWord"
Err.Clear
MargesDocWord = False
End Function
    
```

**Exemple :**

vb

```

Private Sub Command1_Click()
    CommonDialog1.Filter = "Fichiers Word (*.doc)|*.doc"
    CommonDialog1.ShowOpen
    MargesDocWord CommonDialog1.FileName, 1.2, 1.2, 1.5, 1.5
End Sub
    
```

## Comment intercepter la fermeture de WORD

**Auteurs :** bbil ,

**Lorsqu'on pilote WORD en automation, il est possible d'accéder aux événements de l'objet application Word. Pour cela on crée un module de classe, ClassWord, qui va nous permettre d'intercepter l'événement Quit.**

```

Public WithEvents AppWord As Word.Application
Private Sub AppWord_Quit()
    AppWord.WindowState = wdWindowStateMinimize 'Minimise la fenêtre word
    MsgBox "Fin de word" 'Attente du clic opérateur pour fermer word
End Sub
    
```

**Une initialisation est nécessaire :**

```

Dim MonWord As New Word.Application
Dim X As New ClassWord
Private Sub Form_Load()
    Set X.AppWord = MonWord.Application ' Initialise le gestionnaire d'événements.
    MonWord.Visible = True

    'Pour l'exemple création d'un nouveau document
    Dim MonDoc As Word.Document
    
```

```
Set MonDoc = MonWord.Documents.Add
MonDoc.Range.InsertAfter "Bonjour .. "
End Sub
```

**Pensez à cocher la référence à *Microsoft Word X*.**



Sommaire > Liaison Office > Divers Office

## Comment importer une feuille Excel dans une MSFlexGrid ?

Auteurs : Jean-Marc Rabilloud , ThierryAIM ,

Voici une méthode dont le principe consiste à faire un copier-coller de la feuille Excel vers la MsFlexGrid. Vous devez ajouter la référence "Microsoft Excel x.0 Object Library" à votre projet.

Placez le code suivant dans un module standard. La procédure Excel2Flexgrid importe dans une MSFlexGrid le fichier dont le nom est passé en paramètre.

vb

```
Public Sub Excel2Flexgrid(flexgrid As MSFlexGrid, ByVal fichier As String)

    Dim xlapp As Excel.Application
    Dim classeur As Excel.Workbook, feuille As Excel.Worksheet, Plage As Excel.Range

    Set xlapp = New Excel.Application
    xlapp.DisplayAlerts = False
    Set classeur = xlapp.Workbooks.Open(fichier)
    Set feuille = xlapp.ActiveSheet
    Set Plage = feuille.Range("A1").CurrentRegion

    With flexgrid
        .Cols = Plage.Columns.Count
        .Rows = Plage.Rows.Count
        .Col = 0
        .Row = 0
        .ColSel = .Cols - 1
        .RowSel = .Rows - 1
        Plage.Copy
        .Clip = Replace(Clipboard.GetText, vbCrLf, vbCr)
    End With

    Set Plage = Nothing
    Set feuille = Nothing
    classeur.Close False
    Set classeur = Nothing
    Set xlapp = Nothing

End Sub
```

## Comment récupérer le carnet d'adresses d'Outlook ?

Auteurs : ThierryAIM , Alexandre Lokchine ,

Ce code n'est à utiliser seulement si Outlook est le client de messagerie par défaut, sinon MAPI vous renvoie une erreur.

Cochez la référence à Outlook dans les références du projet. Posez un textbox nommé "résultat" sur une form et placez ce code dans le module de la form :

vb

```
Private Declare Function MAPIDetails Lib "MAPI32.DLL" Alias "BMAPIDetails" _
    (ByVal Session&, ByVal UIParam&, Recipient As MapiRecip, _
    ByVal Flags&, ByVal Reserved&) As Long
Private Declare Function MAPIResolveName Lib "MAPI32.DLL" Alias "BMAPIResolveName" _
    (ByVal Session&, ByVal UIParam&, ByVal UserName$, _
```

```

vb
ByVal Flags&, ByVal Reserved&, Recipient As MapiRecip) As
Long

Private Type MapiRecip
    Reserved As Long
    RecipClass As Long
    Name As String
    Address As String
    EIDSize As Long
    EntryID As String
End Type

Sub listemail()
    Dim X As Long, I As Long
    Dim out As Outlook.Application
    Dim a As Object, mapi As Object
    Dim ctrlists As Integer
    Dim info As MapiRecip

    Set out = New Outlook.Application
    Set mapi = out.GetNameSpace("MAPI")
    For ctrlists = 1 To mapi.AddressLists.Count
        Set a = mapi.AddressLists(ctrlists)
        For X = 1 To a.AddressEntries.Count
            I = MAPIResolveName(0, 0, a.AddressEntries(X), 0, 0, info)
            resultat.Text = resultat.Text & "Nom : " & info.Name & " " & "@" : " & _
                Replace(info.Address, "SMTP:", "") & vbCrLf
            'I = MAPIDetails(0, 0, info, 0, 0) 'pour editer les détails.
            DoEvents
        Next
        DoEvents
    Next
    Set a = Nothing
    Set out = Nothing
    Set mapi = Nothing
End Sub

```

La procédure listemail affiche les informations dans le textbox.

## Comment imprimer, visualiser, modifier un état réalisé sous Access depuis VB ?

**Auteurs :** Khany ,

Ce code permet d'utiliser les états existants sous Access. Cochez les références à Access dans le projet Visual Basic et copiez le code approprié.

**Ouverture de la base de données Access :**

```

vb

Dim MaDbMat As String
Dim MesEtats As Access.Application

MaDbMat = App.Path & "\MaBase.mdb"

Set MesEtats = New Access.Application
MesEtats.OpenCurrentDatabase MaDbMat, False

```

**Pour imprimer un état sans le visualiser :**

vb

```
MesEtats.DoCmd.OpenReport "NomdeMonEtat", acViewNormal
```

Pour prévisualiser un état en mode plein écran :

vb

```
MesEtats.Visible = True  
MesEtats.DoCmd.OpenReport "NomdeMonEtat", acViewPreview  
MesEtats.DoCmd.Maximize
```

Pour le mode Design en plein écran :

vb

```
MesEtats.Visible = True  
MesEtats.DoCmd.OpenReport "NomdeMonEtat", acViewDesign  
MesEtats.DoCmd.Maximize
```

## Comment enregistrer une présentation PowerPoint au format html ?

Auteurs : **ThierryAIM** ,

Vous devez ajouter la référence à *Microsoft PowerPoint xx.x library* à votre projet (Menu Projet >> Références...).

vb

```
Private Sub Command1_Click()  
Dim pPoint As PowerPoint.Application  
Dim dPoint As PowerPoint.Presentation  
  
Set pPoint = CreateObject("PowerPoint.Application")  
pPoint.ShowWindowsInTaskbar = False  
pPoint.Visible = True  
  
Set dPoint = pPoint.Presentations.Open("C:\Test.ppt")  
dPoint.SaveAs "C:\Test.html", ppSaveAsHTML  
dPoint.Close  
pPoint.Quit  
Set pPoint = Nothing  
Set dPoint = Nothing  
End Sub
```



Sommaire > Divers > Routines

## Comment obtenir le temps d'exécution d'une partie de mon code ?

Auteurs : **Romain Puyfoulhoux** ,

Ajoutez cette déclaration au début de votre module :

vb

```
Private Declare Function GetTickCount Lib "kernel32" () As Long
```

GetTickCount renvoie le nombre de millisecondes qui s'est écoulé depuis le démarrage du système. Appelez-la au début de votre code, puis à la fin, et la différence entre les deux résultats vous donnera le nombre de millisecondes qui s'est écoulé entre les deux appels.

vb

```
Dim Debut As Long, Fin As Long
Debut = GetTickCount()

'ici le code à chronométrer

Fin = GetTickCount()
MsgBox "Temps mis en millisecondes : " & Fin - Debut
```

## Comment exécuter un code à la première exécution d'un programme ?

Auteurs : **Jean-Marc Rabilloud** ,

Il y a plusieurs méthodes pour faire cela. Habituellement on utilise un emplacement particulier du registre situé sous cette clé :

**HKEY\_CURRENT\_USER\Software\VB and VBA Program Settings\appname\section\key**

Cette partie du registre est directement manipulable avec les quatre fonctions suivantes :

- **SaveSetting appname, section, key, value** : permet de créer ou de modifier une clé du registre.
- **GetSetting(appname, section, key [, default])** ou **GetAllSettings((appname, section)** : permet de récupérer une ou des clés
- **DeleteSetting appname, section, key** : supprime une clé.

Bien sûr, ces fonctions ne permettent pas une gestion complète du registre mais elles vous permettent de stocker quelques valeurs très simplement. C'est ce que nous allons faire avec le code suivant.

vb

```
Private Sub Form_Load()

'vérifie l'existence de la clé
If Len(GetSetting("MonAppli", "Demar", "DejaEx")) = 0 Then
'si elle n'existe pas création de celle-ci
SaveSetting "MonAppli", "Demar", "DejaEx", "Vrai"
'Le code placé ici ne s'exécutera qu'une fois
MsgBox "Je n'apparaîtrai plus", vbInformation + vbOKOnly
End If
```

```
vb
End Sub
```

## Comment transmettre des données à un document Word ?

Auteurs : **Khany**, **Romain Puyfoulhoux**,

Voici une méthode utilisant les signets. Dans Microsoft Word, un signet est un emplacement nommé que l'on utilise comme référence. Il est ajouté via la commande Signets qui est dans le menu Insertion. Ici les signets vous serviront de conteneurs pour les informations envoyées par votre programme. Vous devez leur octroyer un nom afin de pouvoir les identifier depuis Visual Basic. Attention, vous ne pouvez pas ajouter plusieurs fois un signet du même nom.

Voici un exemple d'envoi de données. Dans votre projet, n'oubliez pas de cocher la référence Microsoft Word x.0 Object Library.

```
vb

Dim MyWord As Word.Application, doc As Word.Document
Dim signet As String, i As Long

Set MyWord = New Word.Application

With MyWord

    Set doc = .Documents.Open("c:\modele.doc")
    'Rs est un recordset Adodb, NomPers et PrenomPers sont des signets
    doc.Bookmarks("NomPers").Range.Text = Rs.Fields("nom").Value
    doc.Bookmarks("PrenomPers").Range.Text = Rs.Fields("prenom").Value

    ' exemple de signets allant de Mat1 à Mat11 remplis par les valeurs d'une table TMat
    For i = 0 To 10
        signet = "Mat" & Trim(Str(i + 1))
        doc.Bookmarks(signet).Range.Text = TMat(i)
    Next i
    doc.SaveAs "c:\etat.doc" 'enregistre sous un autre nom
    .Visible = True 'rend l'application visible
    Set doc = Nothing
End With

DoEvents
Set MyWord = Nothing
```

Dans cet exemple le modèle est enregistré sous un autre nom puis le document est rendu visible. Vous pouvez au contraire imprimer le document sans que l'utilisateur sache que Word est utilisé et sans que les modifications soient sauvegardées :

```
vb

doc.PrintOut
doc.Close wdDoNotSaveChanges
```

## Comment extraire un élément d'une chaîne délimitée qui est à une position donnée ?

Auteurs : **Jean-Marc Rabilloud**,

On utilise la fonction ci-dessous qui attend en paramètres la chaîne, la position de l'élément à extraire et le délimiteur. Cette fonction renvoie une chaîne vide lorsqu'elle ne peut pas procéder à l'extraction.

Dans le cas d'une extraction partielle, cette fonction est beaucoup plus rapide qu'un Split.

vb

```

Function ExtraitElement(ChaineRecherche As String, Position As Long, Delim As String) As String

'Renvoie une chaine vide si l'extraction n'est pas possible
On Error GoTo Err_Function

Dim compteur As Long, LastPos As Long, CurPos As Long

If InStr(ChaineRecherche, Delim) = 0 Or Len(ChaineRecherche) = 0 Then Exit Function
LastPos = 1
Do
    CurPos = InStr(LastPos, ChaineRecherche, Delim)
    If CurPos = 0 Then
        If compteur = Position - 1 Then ExtraitElement = Mid(ChaineRecherche, LastPos)
        Exit Do
    Else
        compteur = compteur + 1
        If compteur = Position Then
            ExtraitElement = Mid(ChaineRecherche, LastPos, CurPos - LastPos)
            Exit Do
        End If
    End If
    LastPos = CurPos + 1
Loop While CurPos > 0
Exit Function

Err_Function:
MsgBox "Error " & Err.Number & ": " & Err.Description
Resume Next

End Function

```

## Comment n'autoriser qu'une seule instance de mon application ?

**Auteurs : Romain Puyfoulhoux ,**

Certaines applications n'acceptent d'être ouverte qu'une seule fois. Si l'on essaie de l'ouvrir une deuxième fois, la fenêtre de la première instance repasse en premier plan et est restaurée si nécessaire.

Pour tester le code ci-dessous, créez un projet et ajoutez une form. Son nom est Form1 par défaut. Ajoutez ensuite le code ci-dessous dans un module standard. Enfin, sélectionnez "Sub Main" comme objet de démarrage dans les propriétés du projet.

vb

```

Private Declare Function FindWindow Lib "user32" Alias "FindWindowA" _
    (ByVal lpClassName As String, _
    ByVal lpWindowName As String) As
    Long
Private Declare Function GetWindow Lib "user32" (ByVal hwnd As Long, ByVal wCmd As Long) As Long
Private Declare Function ShowWindow Lib "user32" (ByVal hwnd As Long, ByVal nCmdShow As Long) As
    Long
Private Declare Function SetForegroundWindow Lib "user32" (ByVal hwnd As Long) As Long

Private Const SW_RESTORE = 9
Private Const GW_HWNDPREV = 3

Private Sub Main()

Dim lngHandle As Long

'Cherche une fenêtre qui serait déjà ouverte

```

```

vb
lngHandle = GetPreviousWindow
If lngHandle > 0 Then
    'fenêtre trouvée, on l'affiche
    DisplayWindow lngHandle
Else
    Form1.Show
End If

End Sub

Private Function GetPreviousWindow() As Long

Dim strTitre As String
Dim lngHwnd As Long

'Sauvegarde le titre de l'application et le modifie
'sinon on trouverait toujours une instance de l'application : celle qui vient d'être lancée
strTitre = App.Title
App.Title = "---" & App.Title
'Récupère le handle de la fenêtre principale (invisible)
lngHwnd = FindWindow("ThunderRT6Main", strTitre)
'Obtient le handle de la fenêtre visible
If lngHwnd > 0 Then GetPreviousWindow = GetWindow(lngHwnd, GW_HWNDPREV)
'Restaure le titre original
App.Title = strTitre

End Function

Private Sub DisplayWindow(ByVal lngHandle As Long)

ShowWindow lngHandle, SW_RESTORE
SetForegroundWindow lngHandle

End Sub

```

La Fonction `GetPreviousWindow()` renvoie le handle de la fenêtre de l'application si celle-ci a déjà été ouverte. La fonction `DisplayWindow()` restaure et met au premier plan la fenêtre dont le handle est passé en paramètre. Dans la procédure `Main`, nous recherchons une instance existante. Si nous en avons trouvé une, nous l'activons, sinon nous affichons `Form1`.

Il existe aussi un autre moyen :

```

vb

Private Sub Form_Load()
    If App.PrevInstance Then
        MsgBox "Désolé, une instance est déjà active" & vbCrLf & "Le programme va s'arrêter"
    End
End If
End Sub

```

## Comment lire un fichier XML ?

Auteurs : [Romain Puyfoulhoux](#) ,

La lecture d'un fichier XML se fait à l'aide d'un parseur. Dans les références du projet, ajoutez Microsoft XML.

Voici un exemple qui affiche dans la fenêtre de débogage la liste des balises contenues dans un document xml.



vb

```
Private Sub BrowseChildNodes(root_node As IXMLDOMNode)

    Dim i As Long

    For i = 0 To root_node.childNodes.length - 1
        If root_node.childNodes.Item(i).nodeType <> 3 Then Debug.Print
            root_node.childNodes.Item(i).baseName
            BrowseChildNodes root_node.childNodes(i)
        Next
    End Sub

Private Sub BrowseXMLDocument(ByVal filename As String)

    Dim xmlDoc As DOMDocument, root As IXMLDOMElement

    Set xmlDoc = New DOMDocument
    xmlDoc.async = False
    xmlDoc.Load filename
    Set root = xmlDoc.documentElement
    If Not root Is Nothing Then
        Debug.Print root.baseName
        BrowseChildNodes root
    End If
End Sub
```

Appelez simplement la procédure **BrowseXMLDocument** en passant en paramètre le chemin du fichier. Cette procédure ouvre le fichier puis appelle la procédure **BrowseChildNodes** qui parcourt l'ensemble des balises de façon récursive.

lien : [Home page de MSXML](#)

## Comment générer aléatoirement un mot de passe ?

Auteurs : **Catbull** ,

Génération de 10 mots de passe de 8 caractères.

vb

```
Private Const CaracteresAutorises As String = "0123456789abcdefghijklmnopqrstuvwxyz"

Public Sub main()
    Dim Index As Integer

    For Index = 1 To 10
        Debug.Print Generer(8)
    Next Index
End Sub

Public Function Generer(Longueur As Integer) As String
    Dim Index As Integer

    Randomize
    For Index = 1 To Longueur
        Generer = Generer & Mid(CaracteresAutorises, Int(Len(CaracteresAutorises) * Rnd()) + 1, 1)
    Next Index
End Function
```

```
vb  
End Function
```

## Comment faire une capture d'écran ?

Auteurs : ridan ,

Dans une Form ajouter un contrôle PictureBox. Ajouter ces déclarations dans un module :

```
vb  
  
Private Declare Function BitBlt Lib "gdi32.dll" ( _  
    ByVal hDestDC As Long, _  
    ByVal x As Long, _  
    ByVal y As Long, _  
    ByVal nWidth As Long, _  
    ByVal nHeight As Long, _  
    ByVal hSrcDC As Long, _  
    ByVal xSrc As Long, _  
    ByVal ySrc As Long, _  
    ByVal dwRop As Long) As Long  
  
Private Declare Function GetDesktopWindow Lib "user32.dll" () As Long  
  
Private Declare Function GetDC Lib "user32.dll" ( _  
    ByVal hwnd As Long) As Long  
  
Private Const SRCCOPY As Long = &HCC0020  
  
Public Sub ScreenShot(Pic As PictureBox)  
    Pic.AutoRedraw = True  
    Pic.Width = Screen.Width  
    Pic.Height = Screen.Height  
    Pic.Visible = False  
    BitBlt Pic.hDC, 0&, 0&, Screen.Width, Screen.Height, GetDC(GetDesktopWindow()), 0&, 0&, SRCCOPY  
    SavePicture Pic.Image, "C:\ScreenShot.bmp"  
End Sub
```

Appel de la procédure :

```
vb  
ScreenShot Picture1
```

[lien : Page sources : capture d'écran](#)

[lien : Page sources : capture d'écran via API](#)

[lien : Page sources : capture d'écran via PrintScreen](#)

## Comment lister les variables d'environnement d'une application ?

Auteurs : ridan ,

Fonctionne sous Win 2000 et supérieur.

```
vb  
  
Dim MyStr As String  
i = 1  
Do
```

```
vb
If Environ(i) = "" Then
    Exit Do
Else
    MyStr = MyStr & Environ(i) & vbCrLf
    i = i + 1
End If
Loop
```

### Comment connaître le type du contenu d'un TextBox ?

Auteurs : Catbull ,

```
vb
Dim D as Double

If IsNumeric(Text1.Text) Then
    D=Cdbl(Text1.Text)

    If D-Int(D) = 0 Then
        'Le nombre est entier
    Else
        'Le nombre est décimal
    End if
Else
    'Ce n'est pas un nombre
End if
```

### Comment savoir si le contenu d'un TextBox est un Integer ?

Auteurs : Catbull ,

```
vb
Private Function isInteger(Expression As Variant) As Boolean
    Dim D As Double

    If IsNumeric(Text1.Text) Then
        D = Cdbl(Text1.Text)
        If D = Int(D) Then isInteger = True
    End If
End Function
```

### Comment effectuer des conversions vers le Décimal, l'Hexadécimal ou le Binaire ?

Auteurs : hpj ,

```
vb
Function Dec2Hex(dec As Long) As String
    Dec2Hex = hex(dec)
End Function

Function Hex2Dec(hex As String) As Long
    Hex2Dec = Val("&h" & hex)
End Function

Function Hex2Bin(hex As String) As String
```

vb

```

Dim i As Byte
Dim resultat As String

For i = 1 To Len(hex)
    Select Case Mid(hex, i, 1)
        Case "0": resultat = resultat & "0000"
        Case "1": resultat = resultat & "0001"
        Case "2": resultat = resultat & "0010"
        Case "3": resultat = resultat & "0011"
        Case "4": resultat = resultat & "0100"
        Case "5": resultat = resultat & "0101"
        Case "6": resultat = resultat & "0110"
        Case "7": resultat = resultat & "0111"
        Case "8": resultat = resultat & "1000"
        Case "9": resultat = resultat & "1001"
        Case "A": resultat = resultat & "1010"
        Case "B": resultat = resultat & "1011"
        Case "C": resultat = resultat & "1100"
        Case "D": resultat = resultat & "1101"
        Case "E": resultat = resultat & "1110"
        Case "F": resultat = resultat & "1111"
    End Select
Next i

Hex2Bin = resultat

End Function

Function Bin2Dec(bin As String) As String

    Dim i As Byte
    Dim resultat As Long

    resultat = 0

    For i = 1 To Len(bin)
        If Mid(bin, Len(bin) - i + 1, 1) = 1 Then
            resultat = resultat + 2 ^ (i - 1)
        End If
    Next i

    Bin2Dec = resultat

End Function

```

Pour les autres conversions il suffit de combiner les fonctions...

## Comment convertir un nombre décimal en binaire ?

Auteurs : fdraven ,

```

Public Function DecimalToBinaire(DecVal As Double) As String
    'Variable temporaire qui sert lors du traitement du nombre à convertir
    Dim NbTmp As Double
    'Variable/Indice de boucle
    Dim IndiceP as Integer
    NbTmp = DecVal
    For IndiceP = 1 To Int(Log(DecVal) / Log(2)) + 1
        DecimalToBinaire = CDb1(NbTmp Mod 2) & DecimalToBinaire
        NbTmp = CDb1(Int(NbTmp / 2))
    Next IndiceP

```

```
End Function
```

Il vous suffit ensuite de l'appeler ainsi pour convertir par exemple 1324 en binaire :

```
DecimalToBinaire(1324)
```

lien : [FAQ](#) Comment effectuer des conversions vers le Décimal, l'Hexadécimal ou le Binaire ?

## Comment Arrondir un nombre à sa valeur supérieure ou inférieure

Auteurs : Tofalu ,

Contrairement à la fonction Round qui arrondi un nombre à sa valeur la plus proche en fonction des décimales choisies, cette fonction propose la fonction myRound qui arrondi un nombre à sa valeur supérieure et inférieur en fonction des décimales choisies.

```
Enum myRoundEnum
    myRoundup = -1
    myRoundDown = 1
End Enum

Public Function myRound(vValeur As Variant, Optional byNbDec As Byte, Optional eSens As myRoundEnum = myRoundup) As Variant
    myRound = eSens * Int(eSens * vValeur * 10 ^ byNbDec) / 10 ^ byNbDec
End Function
```

L'appel de cette Fonction :

```
Sub test()
    MsgBox myRound(4.333, 1, myRoundup)
    MsgBox myRound(4.333, 1, myRoundDown)
End Sub
```

## Comment effectuer un calcul statistique à partir des valeurs contenues dans un tableau ?

Auteurs : ThierryAIM ,

Voici par exemple une fonction pour calculer un écart type sur les valeurs contenues dans un tableau :

```
Public Function EcartTypeP(tbl As Variant) As Double
    Dim Var1, Var2
    For i = 1 To UBound(tbl)
        Var1 = Var1 + (tbl(i) * tbl(i)) ' somme des carrés
        Var2 = Var2 + tbl(i) 'somme des valeurs
    Next
    EcartTypeP = Sqr(((UBound(tbl) * Var1) - (Var2 * Var2)) / (UBound(tbl) * UBound(tbl)))
End Function
```

Voici le code à mettre afin de tester cette fonction. Nous remplissons tout d'abord un tableau pour ensuite en calculer l'écart type dont le résultat s'affichera dans la fenêtre d'exécution.

```
Private Sub Bouton1_Click()
```

```
Dim table(10)
'remplir le tableau
For i = 1 To 10
    table(i) = i
Next
'dans un module faites ctrl+g pour afficher la fenêtre d'exécution
Debug.Print EcartTypeP(table)
End Sub
```

## Comment savoir si un nombre est pair ou non ?

Auteurs : spacefrog ,

vb

```
Function ispair(mavar As Long) As Boolean
    ispair = Not mavar And 1
End Function
```

Exemples : ispair(2) retourne true ispair(1) retourne false

## Comment afficher une image en plein écran ?

Auteurs : méphistopheles , Khany ,

En parlant d'image plein écran, on peut penser à des situations différentes. Soit une image qui doit rester proportionnelle afin de ne pas être déformée, soit un fond de feuille uni qui peut être étiré sans perdre de qualité.

Le premier exemple s'attache à l'image proportionnelle :

Dans l'événement Click de l'image, placez le code suivant :

vb

```
Private Sub Picture1_Click()
    Form2.Image1.Picture = Picture1.Picture
    'intégration de l'image dans image1 de form2
    Form2.Show 'afficher form2
    Form2.Tag = Picture1.Height / Picture1.Width
    'calcul du rapport hauteur sur largeur de l'image à copier et envoi dans le tag de la form2
End Sub
```

Dans le code de la form2, qui doit comporter un image et un timer, entrez ceci :

vb

```
Private Sub Form_Click()
    Unload Me 'décharger form2
End Sub

Private Sub Form_Load()
    Image1.Visible = False
    Timer1.Interval = 1
    Timer1.Enabled = True
    'ces commandes ne sont pas nécessaires mais elles permettent de ne pas avoir à entrer les caractéristiques des objets
    dessus
End Sub

Private Sub Timer1_Timer()
```

```

vb
'le timer est utilisé car il faut un délai minimal avant que l'image se retrouve dans imagel
If Me.Tag > (Screen.Height / Screen.Width) Then
    Timer1.Tag = 1
Else
    Timer1.Tag = 0
End If
'détermine si l'image est plus haute que large
If Timer1.Tag = 1 Then
    Me.PaintPicture Image1.Picture, (Screen.Width - (Screen.Width * (Screen.Height / Screen.Width) /
    -
        Me.Tag)) / 2, 0, Screen.Width * (Screen.Height / Screen.Width) / Me.Tag, Screen.Height
Else
Me.PaintPicture Image1.Picture, 0, (Screen.Height - (Screen.Height * (Screen.Width /
Screen.Height) * _
    Me.Tag)) / 2, Screen.Width, Screen.Height * (Screen.Width / Screen.Height) * Me.Tag
End If
'prendre image sur image1
'si image moins large que l'écran par rapport à sa hauteur, centrer paintpicture
'selon l'axe horizontal hauteur de l'image = hauteur de l'écran et largeur de l'image proportionnelle
'sinon centre l'image selon l'axe vertical,
'largeur de l'image = largeur de l'écran et hauteur de l'image proportionnelle
Timer1.Enabled = False
End Sub

```

Par contre, pour afficher un fond de feuille qui peut être un fond "plein écran", il suffit de définir les propriétés de la feuille de fond comme suit :

- **BorderStyle = 0 - None**
- **WindowState = 2 - Maximized**

Dans la feuille dite "de fond", placez un contrôle Image1 renommé ImgFond et le code suivant :

```

vb
Private Sub Form_Load()
    ImgFond.Picture = LoadPicture(PathGraphiques & "fdbleu.jpg")
End Sub

Private Sub Form_Resize()
    ImageFond Me
End Sub

```

Ensuite, dans un module BAS, placez la routine de redimensionnement :

```

vb
Public Sub ImageFond(Feuille As Form)
    Feuille.ImgFond.Move 0, 0, Feuille.ScaleWidth, Feuille.ScaleHeight
End Sub

```

Le contrôle image ne doit pas spécialement être étendu à toute la feuille, il faut juste spécifier sa propriété *Stretch* à *True*.

Comment savoir si mon programme est exécuté depuis l'IDE de VB6 ou en mode compilé ?

**Auteurs : ThierryAIM ,**

**Une astuce parmi d'autres consiste à utiliser la fonction de l'API Windows *GetModuleFileName* dans un module ou une form de votre application.**

## Déclarations :

vb

```
Private Declare Function GetModuleFileName Lib "kernel32" Alias "GetModuleFileNameA" _  
(ByVal hModule As Long, ByVal lpFileName As String, ByVal nSize As Long) As Long
```

**GetModuleFileName** : cette fonction de l'API Windows retourne *ByRef* le chemin complet et le nom du fichier exécutable qui contient le module dans lequel cette fonction est appelée.

- renvoie {VB6 install path}\vb6.exe, si le programme est exécuté depuis l'environnement de développement de VB6 (l'IDE).

- renvoie {chemin du projet}\NomProjet.exe, dans le cas contraire.

La fonction ci-dessous retourne :

- Vrai si le programme tourne depuis l'IDE de Visual Basic

- Faux si le programme tourne depuis l'exécutable

(Nécessite la fonction [FAQ](#) Comment récupérer le nom d'un fichier à partir d'un chemin complet ?)

vb

```
Public Function IsRunningIDE() As Boolean  
    On Error Resume Next  
    Dim sBuffer As String  
    Dim Ret As Integer  
  
    sBuffer = String$(255, Chr$(0))  
    Ret = GetModuleFileName(0, sBuffer, Len(sBuffer))  
    If Ret > 0 Then  
        sBuffer = Mid(sBuffer, 1, Ret)  
        IsRunningIDE = Lcase(ExtractFileName(sBuffer)) = "vb6.exe"  
    End If  
End Function  
  
Private Sub Form_Load()  
    MsgBox IsRunningIDE  
End Sub
```

**NOTA** : remplacez "vb6.exe" par votre version de Visual Basic, si vous utilisez une version antérieure

## Comment utiliser la fonction split en VB5 ?


Auteurs : Delbeke ,

La fonction split, n'existe pas en vb5 mais on peut créer sa propre fonction :

```
Public Function Split(ByVal MyString As String, Optional ByVal Separator As String = " ") As Variant  
    Dim iPosit As Long  
    Dim Table() As String  
    ReDim Table(0)  
    iPosit = InStr(MyString, Separator)  
    If iPosit = 1 Then  
        MyString = Mid$(MyString, Len(Separator) + 1)  
        iPosit = InStr(MyString, Separator)  
    End If  
    While iPosit > 0  
        Table(UBound(Table)) = Left(MyString, iPosit - 1)  
        MyString = Mid$(MyString, iPosit + Len(Separator))  
        ReDim Preserve Table(UBound(Table) + 1)  
        iPosit = InStr(MyString, Separator)  
    End While
```



```
Wend
Table(UBound(Table)) = MyString
Split = Table
End Function
```

 Cette fonction s'utilise comme la fonction split standard de VB6, vous trouverez plusieurs exemples dans cette faq.

lien : Une autre façon d'écrire la fonction voir FAQ Access 97

## Comment Encoder des url ?

Auteurs : DarkVader ,

En utilisant la fonction JavaScript, Escape :  
cochez la référence à [Microsoft Script Control](#)

```
Dim strBase As String, strEncode As String, SC As New ScriptControl
strBase = "Ceci est une chaîne encodée"
SC.Language = "JavaScript"
strEncode = SC.Eval("escape(" + Chr(34) + strBase + Chr(34) + ")")
```

L'inverse est aussi possible par l'utilisation de unescape

```
SC.Eval("unescape(" + Chr(34) + strEncode + Chr(34) + ")")
```

## Comment récupérer les propriétés d'un PDF

Auteurs : Cafeine ,

Pour obtenir les propriétés d'un PDF (nom, titre, etc.), il faut lire le fichier en mode binary. L'utilisation des RegExp permettra d'accéder aux informations recherchées. C'est ce que fait la fonction suivante :

```
Function GetPDFTitle(ByVal strFic As String, strObj As String) As String

Dim fic As Integer
Dim strExp As String
Dim strBuff As String * 1024
Dim i As Integer

Dim reg As VBScript_RegExp_55.RegExp
Dim Match As VBScript_RegExp_55.Match
Dim Matches As VBScript_RegExp_55.MatchCollection

Set reg = New VBScript_RegExp_55.RegExp

reg.Global = True
reg.Multiline = False
reg.IgnoreCase = True
reg.Pattern = "/" & strObj & " \((.*)\)"

Reset

fic = FreeFile

Open strFic For Binary Access Read As #fic

Do While Not EOF(fic)
```

```
Get #fic, , strBuff
strExp = strExp & strBuff

If reg.Test(strExp) = True Then
    Set Matches = reg.Execute(strExp)
    For Each Match In Matches
        GetPDFTitle = Match.SubMatches(0)
    Next Match
    Exit Function
Else
    strExp = right(strExp, 1024)
End If
Loop
Reset

Set Match = Nothing
Set Matches = Nothing
Set reg = Nothing

End Function
```


#### Exemple d'utilisation :

```
getpdftitle( "d:\temp\20060331164202.pdf", "Title")
```

#### Il est possible de remplacer title par un des champs suivants :

- **CreationDate**
- **ModDate**
- **Title**
- **Creator**
- **Author**

lien :  [Tutoriel : Le PDF gratuit pour Access](#)

lien :  [Les expressions rationnelles / régulières dans Access par la pratique](#)

lien : [FAQ](#) [Comment utiliser les expressions régulières ?](#)

## Comment effectuer un codage/décodage base64 ?

Auteurs : [Delbeke](#) ,

**Ceci est plus une astuce qu'un vrai code**

**On utilise une référence *As Microsoft XML, version 2.0* , qui en sous produit, fait le codage/décodage du base 64 fonction d'encodage :**

```
Public Function Encode_Base64(Text As String) As String

    Dim Xml As New MSXML.DOMDocument
    Dim Conv As MSXML.IXMLDOMElement
    Dim Arr() As Byte
    If Text = "" Then
        Encode_Base64 = ""
        Exit Function
    End If
    Arr = StrConv(Text, vbFromUnicode)
    Set Conv = Xml.createElement("Base64")
    Conv.dataType = "bin.base64"
    Conv.nodeTypeValue = Arr
```

```
Encode_Base64 = Conv.Text
End Function
```

**fonction décode :**

```
Public Function Decode_Base64(Text As String) As String
    Dim Xml As New MSXML.DOMDocument
    Dim Conv As MSXML.IXMLDOMElement
    If Text = "" Then
        Decode_Base64 = ""
        Exit Function
    End If
    Set Conv = Xml.createElement("Base64")
    Conv.dataType = "bin.base64"
    Conv.Text = Text
    Decode_Base64 = StrConv(Conv.nodeTypeValue, vbUnicode)
End Function
```



*Vous pouvez utiliser une version plus récente de "Microsoft XML", dans ce cas modifiez les déclarations de variables en remplaçant MSXML... par MSXML2...*

Comment effectuer un codage/décodage en Quote-Printable ?

**Auteurs : Delbeke ,**

Le codage Quote-Printable est un codage où toutes les lettres non comprises dans l'intervalle [33-60] [62-126] sont remplacés par =XX , où XX est le code ascii du caractère.

Le tout est découpé en ligne de 76 caractères maximum.

**fonction d'encodage :**

```
Public Function EncodeQuotedPrintable(Text As String) As String
    Dim lPntIn As Long           'compteur caractères dans Text
    Dim lPntOut As Long          'position insertion dans buffer
    Dim lLenLign As Long         'Longueur ligne en cours
    Dim Buffer As String          'buffer reception du codage
    Dim Char As String           'le caractère en cours d'analyse
    Dim AsciiCode As Integer     'son code ascii
    If Text = "" Then
        EncodeQuotedPrintable = ""
        Exit Function
    End If
    Buffer = String(Len(Text) * 3, 0) ' au max, 3 caractères en sortie pour chaque caractère en entrée
    lPntOut = 1
    lLenLign = 1
    For lPntIn = 1 To Len(Text)
        Char = Mid(Text, lPntIn, 1)
        AsciiCode = Asc(Char)
        Select Case AsciiCode
            Case 33 To 60, 62 To 126
                'caractère littéral
                'tous ces caractères peuvent être acceptés tels quels
                Mid(Buffer, lPntOut, 1) = Char
                lPntOut = lPntOut + 1
                lLenLign = lLenLign + 1
            Case 9, 32
                '-----
                'version abandonnée
                'Mid(Buffer, lPntOut, 1) = Char
                'lPntOut = lPntOut + 1
                'lLenLign = lLenLign + 1
                '-----
                'le caractère blanc et le caractère tab sont censés être acceptés sans codage
        End Select
    Next
End Function
```

```

'mais pas s'ils terminent une ligne ! auquel cas il doivent être codés.
'Comme ce n'est pas simple à coder, je les code tous
Mid(Buffer, lPntOut, 3) = "=" & Right("00" & Hex(AsciiCode), 2)
lPntOut = lPntOut + 3
lLenLign = lLenLign + 3
Case Else
'on code tous les autres caractères
Mid(Buffer, lPntOut, 3) = "=" & Right("00" & Hex(AsciiCode), 2)
lPntOut = lPntOut + 3
lLenLign = lLenLign + 3
End Select
End Select
If lLenLign > 72 Then
'si on arrive en bout de ligne (qui ne doit pas passer 76 caractères)
'on insère une continuation de ligne ( =CRLF )
Mid(Buffer, lPntOut, 3) = "=" & vbCrLf
lPntOut = lPntOut + 3
lLenLign = 1
End If
Next
EncodeQuotedPrintable = Left(Buffer, lPntOut - 1)
'si on termine par "=" & vbCrLf , on le retire
If Right(EncodeQuotedPrintable, 3) = "=" & vbCrLf Then
EncodeQuotedPrintable = Left(EncodeQuotedPrintable, Len(EncodeQuotedPrintable) - 3)
End If
End Function

```


### fonction décode :

```

Public Function DecodeQuotedPrintable(Text As String) As String
Dim lPntIn As Long      'compteur caractères dans Text
Dim lPntOut As Long    'position insertion dans buffer
Dim Buffer As String    'buffer reception du dé-codage
Dim Char As String     'le caractere en cours d'analyse
Dim AsciiCode As String 'son code ascii en hexadécimal
If Text = "" Then
DecodeQuotedPrintable = ""
Exit Function
End If
Buffer = String(Len(Text), 0) ' au max, 1 caractère en sortie pour chaque caractère en entrée
lPntOut = 1

For lPntIn = 1 To Len(Text)
Char = Mid(Text, lPntIn, 1)
Select Case Char
Case "="
AsciiCode = Mid(Text, lPntIn + 1, 2)
If AsciiCode = vbCrLf Then
'caractère continuation de ligne
lPntIn = lPntIn + 2
Else
'caractère codé
Mid(Buffer, lPntOut, 1) = Chr(Val("&H" & AsciiCode))
lPntOut = lPntOut + 1
lPntIn = lPntIn + 2
End If
Case Else
'caractère littéral
Mid(Buffer, lPntOut, 1) = Char
lPntOut = lPntOut + 1
End Select
Next
DecodeQuotedPrintable = Left(Buffer, lPntOut - 1)
End Function

```

 Vous pouvez utiliser une version plus récente de "Microsoft XML", dans ce cas modifiez les déclarations de variables en remplaçant MSXML... par MSXML2...

## Comment trier un tableau d'entier ?

Auteurs : méphistopheles ,

Une adaptation du QuickSort des sources algo de développez : page sources algo

```
Public Function QUICKSORT(ByRef Tableau() As Integer, Optional ByVal Debut As Integer = -1, _
Optional ByVal Fin As Integer = -1)
'attention, ne pas avoir un tableau avec des indices négatifs.
If Debut = -1 Then Debut = LBound(Tableau, 1)
If Fin = -1 Then Fin = UBound(Tableau, 1)

Dim Pivot As Integer
Dim Gauche As Integer
Dim Droite As Integer
Dim temp As Integer
Pivot = Debut
Gauche = Debut
Droite = Fin
Do
If Tableau(Gauche) >= Tableau(Droite) Then
temp = Tableau(Gauche)
Tableau(Gauche) = Tableau(Droite)
Tableau(Droite) = temp
Pivot = Gauche + Droite - Pivot
End If
If Pivot = Gauche Then
Droite = Droite - 1
Else
Gauche = Gauche + 1
End If
DoEvents
Loop Until Gauche = Droite
If Debut < Gauche - 1 Then QUICKSORT Tableau, Debut, Gauche - 1
If Fin > Droite + 1 Then QUICKSORT Tableau, Droite + 1, Fin
End Function
```

## Comment enlever les accents d'une chaîne

Auteurs : Cafeine ,

Voici une solution en utilisant les API.

Coller ce code dans un nouveau module :

```
Private Declare Function FoldString Lib "kernel32.dll" Alias _
"FoldStringA" (ByVal dwMapFlags As Long, ByVal lpSrcStr As Long, _
ByVal cchSrc As Long, ByVal lpDestStr As Long, ByVal cchdest As Long) As Long

Function OteAccents(ByVal str As String) As String

Dim i As Integer
OteAccents = Space(Len(str))

For i = 0 To (Len(str) - 1) * 2 Step 2
FoldString &H40, StrPtr(str) + i, 1, StrPtr(OteAccents) + i, 1
Next i
```

```
End Function
```

Sommaire > Divers > Manipulation de dates

## Comment obtenir une date à partir des numéros du jour, de la semaine et de l'année ?

Auteurs : Jean-Marc Rabilloud , Khany , Romain Puyfoulhoux ,

vb

```
Public Function InvDatePart(ByVal PosJour As Integer, ByVal NumSemaine As Integer, ByVal Annee As Integer) As Date

    Dim tmpDate As Date

    tmpDate = CDate("1/1/" & Format$(Annee))
    If Weekday(tmpDate, vbMonday) < 6 Then NumSemaine = NumSemaine - 1
    tmpDate = DateAdd("ww", NumSemaine, tmpDate)
    tmpDate = DateAdd("d", PosJour - Weekday(tmpDate, vbMonday), tmpDate)
    InvDatePart = tmpDate

End Function
```

Par exemple, InvDatePart(3, 10, 2004) vous renvoie la date correspondant au troisième jour de la dixième semaine de l'année 2004.

## Comment déterminer le premier jour d'une semaine ?

Auteurs : hpj ,

**NOTE : Ce code considère que la semaine n°1 est celle qui contient au moins quatre jours dans la nouvelle année.**

vb

```
Public Function PremierJourSemaine(ByVal numSemaine As Byte, ByVal annee As Integer) As Date

    Dim d As Date
    Dim numS As Integer

    ' premier jour de l'année
    d = DateSerial(annee, 1, 1)

    ' numéro de la semaine du 1er janvier
    numS = DatePart("ww", d, vbMonday, vbFirstFourDays)

    ' si le 1er janvier fait partie de la dernière semaine de l'année précédente
    ' alors passe à la semaine suivante (la première de la nouvelle année)
    If numS <> 1 Then d = DateAdd("d", 7, d)

    ' calcule le premier jour de la première semaine de l'année
    d = DateAdd("d", 1 - Weekday(d, vbMonday), d)

    ' ajoute (numSemaine-1) semaines
    PremierJourSemaine = DateAdd("ww", numSemaine - 1, d)

End Function
```

```
vb  
End Function
```

## Comment convertir une chaîne de caractères en date ?

Auteurs : Romain Puyfoulhoux ,

Pour convertir une expression en date, vous pouvez utiliser la fonction `CDate(expression)`. Si l'expression à convertir n'est pas une date correcte d'après le format indiqué dans les paramètres régionaux de Windows, `CDate` essaie d'utiliser un autre format. Si la fonction ne réussit pas à faire la conversion, elle génère une erreur de type "Type Mismatch". Dans l'exemple ci-dessous, une date est demandée à l'utilisateur. Sa réponse est alors convertie en date.

```
vb  
  
Dim Rep As String, DateDeNaissance As Date  
Do  
    Rep = InputBox("Quelle est votre date de naissance ?")  
Loop While (Not IsDate(Rep))  
DateDeNaissance = CDate(Rep)
```

## Comment ajouter des heures, des jours ou des mois à une date ?

Auteurs : Romain Puyfoulhoux ,

Vous pouvez utiliser la fonction `DateAdd (intervalle, nombre, date)`

**intervalle** : chaîne de caractères indiquant l'intervalle de temps que vous voulez ajouter ("m" : mois, "d" : jour, "yyyy" : année, etc...)

**nombre** : nombre d'intervalles que vous voulez ajouter

**valeur renvoyée** : une date contenant le résultat (la date passée en argument n'est pas modifiée)

Quelques exemples :

```
vb  
  
today = Now()  
  
DateAdd("m", 3, today)    'renvoie today + 3 mois  
DateAdd("d", 2, today)    'renvoie today + 2 jours  
DateAdd("ww", 1, today)   'renvoie today + 1 semaine  
DateAdd("h", 1, today)    'renvoie today + 1 heure
```

## Comment calculer le temps écoulé entre deux dates ?

Auteurs : Romain Puyfoulhoux ,

Avec la fonction `DateDiff()` :

```
vb  
  
Dim date1 As Date, date2 As Date  
  
date1 = "01/01/2004"
```



vb

```
date2 = "01/01/2005"  
  
MsgBox "Durée en jours : " & DateDiff("d", date1, date2)  
MsgBox "Durée en nombre de mois : " & DateDiff("m", date1, date2)
```

## Comment convertir une date julienne ?

Auteurs : Jean-Marc Rabilloud ,

vb

```
Dim DateJulienne As String  
Dim Reponse As Date  
  
DateJulienne = "2002211"  
Reponse = DateSerial(CInt(Left(JulianVal, 4)), 1, CInt(Mid(JulianVal, 5)))  
  
' La valeur renvoyée est une date  
  
MsgBox Reponse
```

## Comment connaître le dernier jour du mois ?

Auteurs : cafeine ,

Il suffit de passer au mois suivant et de retirer 1 jour. Le code suivant se base sur le mois en cours.

vb

```
Dim Reponse As Date  
  
Reponse = CDate("01/" & Month(Date) + 1 & "/" & Year(Date))  
Reponse = DateAdd("d", -1, Reponse)  
  
MsgBox Reponse
```

## Comment trier ou comparer des dates facilement ?

Auteurs : Jacques Malatier ,

*Extrait de l'aide Microsoft Visual Basic :*

Les dates sont stockées sous la forme d'une partie d'un nombre réel.

Les valeurs situées à gauche du séparateur décimal représentent la date, tandis que celles situées à sa droite représentent l'heure. Les valeurs négatives correspondent à des dates antérieures au 30 décembre 1899.

L'astuce consiste donc à transformer les dates en valeurs décimales, avant de les comparer ou de les trier !

Exemple :

vb

```
If CDec(CDate(MaDate1)) >= CDec(CDate(MaDate2)) And CDec(CDate(MaDate3)) < CDec(CDate(Now)) Then
```

Un autre exemple de  Tri d'une ListView sur une colonne Date par SilkyRoad

## Comment savoir si une année est bisextile ?

Auteurs : random , Theo ,

2 solutions vous sont proposées pour répondre à la question posée :

1) Fun, avec toutes les explications adéquates :

vb

```
Function IsBisextile(maDate As Date) As Boolean
'Fonction de contrôle de la bisextilité d'une année à partir d'une date

'Les conditions pour avoir une année bisextile sont les suivantes:
' - année divisible par 4 : année bisextile
' - exception : année divisible par 100 : année non bisextile
' - exception de l'exception : année divisible par 400 : année bisextile

If Year(maDate) Mod 4 = 0 And (Year(maDate) Mod 100 <> 0 Or Year(maDate) Mod 400 = 0) Then
    IsBisextile = True
Else
    IsBisextile = False
End If

End Function
```

2) Moins fun mais tout aussi efficace, par code direct sur le 29 février :

vb

```
Function IsBisextile(madate As Date) As Boolean
    IsBisextile = Day(DateSerial(Year(madate), 3, 0))=29
End Function
```

A vous de choisir celle que vous préférez !

## Comment créer un timer sans utiliser le composant standard de VB6 ?

Auteurs : ThierryAIM ,

A défaut d'utiliser le contrôle Timer, il existe les API SetTimer et KillTimer qui ont l'avantage : \* de pouvoir créer un timer sans le contrôle donc éventuellement sous VBA \* de pouvoir créer un timer dont la fréquence est codée sur un long, soit une fréquence supérieure à 65535ms, ce qui évite la solution d'appeler une variable publique servant de multiple dans le contrôle timer \* de pouvoir créer un timer sans form en faisant pointer sur une classe

vb

```
Declare Function SetTimer Lib "user32" (ByVal hwnd As Long, ByVal nIDEvent As Long, ByVal uElapse As Long, ByVal lpTimerFunc As Long) As Long
Declare Function KillTimer Lib "user32" (ByVal hwnd As Long, ByVal nIDEvent As Long) As Long

Sub TimerProc(ByVal hwnd As Long, ByVal nIDEvent As Long, ByVal uElapse As Long, ByVal lpTimerFunc As Long)
    '-- ici, le code a exécuter par le Timer
End Sub
```

vb

```
End Sub

Public Sub main()
    KillTimer 0, 0
    SetTimer 0, 0, 5000, AddressOf TimerProc
End Sub
```