Utiliser les contrôles dans un UserForm, en VBA Excel

par SilkyRoad (silkyroad.developpez.com)

Date de publication : 15/08/2006

Dernière mise à jour : 23/06/2007

Ce document décrit comment ajouter et utiliser des contrôles dans un UserForm.
Vous y trouverez une description des objets de base, et aussi quelques informations sur d'autres objets plus spécifiques.
Les exemples présentés ne sont qu'une partie infime des options offertes par tous les contrôles. Les possiblités d'utilisation sont très étendues et simples à mettre en oeuvre.
Toutes les procédures de ce document ont été testées en utilisant Excel2002.

I - Introduction

I-A - Comment Insérer un contrôle

Vous devez tout d'abord créer un **UserForm** dans votre classeur afin d'y ajouter des contrôles.
Les contrôles sont accessibles depuis l'éditeur de macro, dans la fenêtre "boîte à outils".
Normalement la boîte à outils s'affiche lorsque que vous créez un UserForm. Dans le cas contraire, Sélectionnez le Menu Affichage / Boîte à outils.

Lorsque vous passez le curseur de la souris sur les objets, leur nom s'affiche dans une infobulle. Sélectionnez un des contrôles disponibles et positionnez le dans l'UserForm.
Si l'objet que vous souhaitez utiliser n'est pas visible, faites un clic droit dans la boîte à outils, sélectionnez l'option "Contrôles supplémentaires". Dans la nouvelle fenêtre qui s'affiche, cochez la ligne qui vous intéresse puis cliquez sur OK pour valider.
Utilisez la fenêtre de propriétés pour personnaliser le contrôle (Changer le nom, une fonctionnalité, l'apparence de l'objet...etc...). Si cette fenêtre n'est pas affichée par défaut: Sélectionnez le contrôle et appuyez sur la touche raccourci "F4".
Si les macros servant à lire ou modifier une propriété sont placées dans l'UserForm, vous pouvez écrire:

Vba
```
Label1.Caption = "Test"
```

Si la procédure est externe (placée dans un module standard, un autre UserForm...etc...) vous devez spécifier le nom de la Forme contenant l'objet:

Vba
```
UserForm1.Label1.Caption = "Test"
```

I-B - Boucler sur les contrôles contenus dans la Forme

Si votre projet contient de nombreux objets, il peut être intéressant de créer des boucles pour optimiser les procédures.
Cet exemple boucle sur les contrôles afin de récupérer leur nom:

Vba
```
Dim Ctrl As Control

'Boucle sur la collection de contrôles
For Each Ctrl In Me.Controls
    MsgBox Ctrl.Name
Next Ctrl
```

Vous pouvez aussi filtrer sur un type de contrôle spécifique.
Cette procédure permet de boucler sur l'ensemble des contrôles et affiche leur contenu s'il s'agit de TextBox.

Vba
```
Dim Ctrl As Control

For Each Ctrl In Me.Controls
    If TypeOf Ctrl Is MSForms.TextBox Then MsgBox Ctrl.Object.Value
```

```vba
Next Ctrl
```

Une autre possiblité.

```vba
Dim Ctrl As Control

For Each Ctrl In Controls
    If TypeName(Ctrl) = "TextBox" Then MsgBox Ctrl.Object.Value
Next Ctrl
```

Encore un autre exemple pour boucler sur les TextBox: Transférer le contenu de 10 TextBox dans la plage de cellules A1:A10

```vba
Private Sub CommandButton1_Click()
    Dim i As Integer

    For i = 1 To 10
    Cells(i, 1) = Me.Controls("TextBox" & i)
    Next i
End Sub
```

Cette dernière méthode nécessite que les TextBox soient nommés TextBox1 à Textbox10 et ordonnés pour correspondre aux cellules A1:A10. Cela demande de la rigueur lors de la création mais vous fera gagner un temps précieux, une économie dans le nombre de lignes de macro et une meilleure lisibilité de vos procédures.
Lorsque vous devez ajouter un grand nombre de contrôles qui seront utilisés de la même manière (par exemple un groupe de TextBox), il devient vite fastidieux de réécrire plusieurs fois la même procédure. Dans ce cas, vous pouvez utiliser des modules de classe. Les classes définissent le fonctionnement des objets. Un objet peut être défini par une propriété, une méthode ou un évènement. (Vous n'écrivez qu'une seule procédure qui sera appliquable à tous les contrôles que vous aurez défini dans une collection).

II - Les contrôles standards

II-A - Label

Les contrôles Labels (intitulés) sont principalement utilisés pour afficher des messages d'information. Lorsque l'USF est affiché, Vous ne pouvez modifier le contenu que par macro.
Cet exemple modifie le texte d'un Label:

Vba
```
Label1.Caption = "le nouveau texte"
'Il est aussi possible d'écrire
'Label1 = "le nouveau texte"
```

Pour lire le contenu d'un Label:

Vba
```
MsgBox Label1.Caption
'Il est aussi possible d'écrire
'MsgBox Label1
```

Spécifier des polices type Symbole par macro:
En général vous pouvez utiliser directement la syntaxe Label1.Font.Name ="Arial" pour définir une police. Pour paramétrer des polices type Symbole (Wingdings, Webdings ...), utilisez:

Vba
```
Label1.Font.Name ="Wingdings"
Label1.Font.Charset = 2
```

Nota:
Utilisez le même principe pour les TextBox.

Tout comme Les **UserForm**, les contrôles possèdent des évènements.
Exemple:
Le Label nommé Label1 suit le curseur de la souris après le 1er clic sur lui. Le 2eme clic permet de désactiver cette action.

Vba
```
Option Explicit

Dim Cible As Boolean

Private Sub UserForm_Initialize()
    Cible = False
End Sub

Private Sub Label1_Click()
    If Cible = True Then
        Cible = False
        Else
        Cible = True
    End If
End Sub


Private Sub Label1_MouseMove(ByVal Button As Integer, ByVal Shift As Integer, _
ByVal X As Single, ByVal Y As Single)
    If Cible = True Then
```

```vba
        Label1.Left = Label1.Left + X
        Label1.Top = Label1.Top + Y
    End If
End Sub


Private Sub UserForm_MouseMove(ByVal Button As Integer, _
ByVal Shift As Integer, ByVal X As Single, ByVal Y As Single)
    If Cible = True Then
        Label1.Left = X
        Label1.Top = Y
    End If
End Sub
```

II-B - CheckBox

Le contrôle CheckBox (Case à cocher) permet de renvoyer les valeurs:
- Vrai (Lorsque la case est cochée)
- Faux (Lorsque la case est décochée)
- Un CheckBox peut aussi prendre la valeur Null si la propriété TripleState = True

Cet exemple affiche la valeur du CheckBox1 lorsque l'on clique dessus.

```vba
Private Sub CheckBox1_Change()
    Select Case CheckBox1.Value
        Case True: CheckBox1.Caption = "Vrai"
        Case False: CheckBox1.Caption = "Faux"
        Case Else: CheckBox1.Caption = "Null"
    End Select
End Sub
```

Modifier la valeur du CheckBox1.

```vba
Private Sub CommandButton1_Click()
    CheckBox1 = True
End Sub
```

II-C - OptionButton

Les contrôles Optionbutton (Boutons d'option) permettent de faire **un** choix parmi plusieurs options. Lorsqu'une des options est sélectionnée, les autres sont toutes désactivées. Il existe deux possibilités pour gérer un groupe d'OptionButton.

1. Utiliser la propriété GroupName en Attribuant la même chaine (par exemple "GR1") à tous les contrôles que vous souhaitez regrouper.

Cet exemple permet de retrouver l'Optionbutton sélectionné.

```vba
Private Sub CommandButton1_Click()
    Dim Ctrl As Control

    'Boucle sur tous les contrôles
    For Each Ctrl In Me.Controls
        'Vérifie qu'il s'agit d'un OptionButton
        If TypeOf Ctrl Is MSForms.OptionButton Then
```

```vba
            'Vérifie si l'OptionButton fait partie d'un groupe nommé "GR1"
             If Ctrl.GroupName = "GR1" Then
                'Affiche le Caption de l'optionButton qui a la valeur True
                If Ctrl.Value = True Then
                    MsgBox Ctrl.Caption
                    'Sort de la boucle (Il ne peut y a voir qu'une
                    'réponse à True)
                    Exit For
                End If
            End If
        End If
    Next
End Sub
```

2. La deuxième solution consiste à regrouper les OptionButton dans un Frame. Cet exemple permet de retrouver l'Optionbutton qui est sélectionné dans le Frame (Le Frame est supposé ne pas contenir d'autres types de contrôles).

```vba
Private Sub CommandButton1_Click()
    Dim Ctrl As Control

    For Each Ctrl In Frame1.Controls
        If Ctrl.Object.Value = True Then
            MsgBox Ctrl.Object.Caption
            Exit For
        End If
    Next Ctrl
End Sub
```

II-D - CommandButton

Les contrôles CommandButton (Boutons de commande) servent principalement à lancer des procédures en utilisant l'évènement Click.

```vba
Private Sub CommandButton1_Click()
    MsgBox "Coucou"
End Sub
```

Rendre un bouton actif ou inactif.

```vba
CommandButton1.Enabled=True 'pour activer
'CommandButton1.Enabled=False 'pour désactiver
```

Rendre un bouton visible ou invisible.

```vba
CommandButton1.Visible = True
'CommandButton1.Visible = False
```

II-E - TextBox

Les contrôles TextBox (Zones de texte) sont généralement utilisés comme champs de saisie dans les boîtes de dialogue.
Cet exemple transfére le contenu d'un TextBox dans la cellule A1:

Vba
```
Range("A1") = TextBox1
```

Transférer le contenu de la cellule A2 dans le TextBox:

Vba
```
TextBox1 = Range("A2")
```

Forcer un format date type jj/mm/aaaa dans un TextBox

Vba
```
Private Sub TextBox1_Change()
 Dim Valeur As Byte
 TextBox1.MaxLength = 10 'nb caractères maxi autorisé dans le textbox
 Valeur = Len(TextBox1)
 If Valeur = 2 Or Valeur = 5 Then TextBox1 = TextBox1 & "/"
End Sub


'Ensuite pour vérifier que c'est bien une date qui a été saisie
Private Sub CommandButton1_Click()
    If Not IsDate(TextBox1) Then
        MsgBox "Format incorrect"
        TextBox1 = ""
        Exit Sub
        Else
        MsgBox "Format correct"
        '...la suite de la procédure
    End If
End Sub
```

Déclencher une tabulation automatique lorsque le nombre de caractères maxi autorisé (4) est atteint.

Vba
```
Private Sub UserForm_Initialize()
    'Définit le nombre de caractères maxi dans le textbox
    TextBox1.MaxLength = 4
    'Définit la tabultation automatique
    TextBox1.AutoTab = True
End Sub
```

Remplacer les caractères par des astérisques, lors de la saisie dans un textbox.

Vba
```
Private Sub userForm_Initialize()
    Me.TextBox1.PasswordChar = "*"
End Sub
```

Aller à la ligne dans un TextBox en utilisant la touche clavier "Entrée".
Nota:

Par défaut, il faut utiliser les combinaisons de touche Ctrl+Entrée ou Shift+Entrée pour aller à la ligne dans un TextBox et avoir préalablement paramétré la propriété MultiLine à true.

Vba

```
Private Sub userForm_Initialize()
    With TextBox1
        'Autorise les mutilignes dans le TextBox
        'Attention: cette propriété est toujours à False par défaut
        .MultiLine = True

        'Spécifie que la touche ENTRÉE ajoutera une nouvelle ligne.
        .EnterKeyBehavior = True
    End With
End Sub
```

Forcer les majuscules lors de la saisie dans un TextBox.

Vba

```
Private Sub textBox1_keyPress(ByVal keyAscii As MSForms.ReturnInteger)
    keyAscii = Asc(UCase(Chr(keyAscii)))
End Sub
```

Appliquer le Focus dans un Textbox et sélectionner tout le texte qu'il contient.

Vba

```
With TextBox1
    .SetFocus
    .SelStart = 0
    .SelLength = Len(TextBox1.Text)
End With
```

Garder le focus dans un TextBox tant qu'il est vide.

Vba

```
Private Sub TextBox1_Exit(ByVal Cancel As MSForms.ReturnBoolean)
 If TextBox1.Value = "" Then Cancel = True
End Sub
```

Autoriser uniquement la saisie de valeurs numériques dans un TextBox.

Vba

```
'avec la virgule non valide (entier)
Private Sub textBox1_Change()
    On Error Resume Next

    If Not IsNumeric(Right(TextBox1, 1)) Then
        MsgBox "Le caractere saisi n'est pas valide"
        TextBox1 = Left(TextBox1, Len(TextBox1) - 1)
    End If
End Sub
```

Vba

```
'avec la virgule valide(décimale)
Private Sub textBox1_Change()
    On Error Resume Next

    If Not IsNumeric(Right(TextBox1, 1)) And Right(TextBox1, 1) <> "," Then
```

```vba
        MsgBox "Le caractere saisi n'est pas valide"
        TextBox1 = Left(TextBox1, Len(TextBox1) - 1)
    End If
End Sub
```

Un autre exemple pour autoriser uniquement la saisie de valeurs numériques dans un TextBox

```vba
'Auteur Didier MdF
Private Sub TextBox1_KeyPress(ByVal KeyAscii As MSForms.ReturnInteger)
    Select Case KeyAscii
    Case Is < 48, Is > 57
        KeyAscii = 0
    End Select
End Sub
```

Supprimer les symboles du saut de ligne (carré blanc) lors du transfert d'un TextBox vers une cellule.

```vba
Range("A1") = _
    Application.WorksheetFunction.Substitute(TextBox1, vbCrLf, Chr(10))
```

II-F - ComboBox

Un contrôle ComboBox (Zone de liste modifiable) permet de créer une liste de choix dans un menu déroulant.
Remarque:
Ayez toujours une réflexion sur ce que vous allez y afficher et le nombre de données que vous allez insérer...Un ComboBox de 500 items est difficilement lisible.
Les méthodes et propriétés sont identiques pour les Combobox et les ListBox. La ListBox posséde seulement en plus une option permettant les multisélections dans la liste de choix. Vous pouvez réutiliser les exemples des 2 chapitres en fonction de vos besoins.
La propriété Style spécifie de quelle façon l'utilisateur va utiliser le contrôle.

Constante	Valeur	Description
fmStyleDropDownCombo	0	Le contrôle ComboBox se comporte comme une liste modifiable déroulante. L'utilisateur peut saisir une valeur dans la zone d'édition ou en sélectionner une dans la liste déroulante.
fmStyleDropDownList	2	Le contrôle ComboBox se comporte comme une zone de liste. L'utilisateur doit choisir une valeur dans la liste.

Sans macro, vous pouvez indiquer manuellement la plage source en indiquant par exemple Feuil1!A1:A100 dans le champ **RowSource** de la fenêtre Propriétés.
Si le nom de la feuille contient un espace, il faut encadrer le nom par des quotes: 'Nom feuille'!A1:A100
Si le nom de la feuille n'est pas spécifié, c'est la feuille active qui sera prise en compte.
Utiliser La méthode AddItem pour Remplir un ComboBox.

Vba
```
Private Sub UserForm_Initialize()
    Dim i As Byte

    For i = 1 To 5
        ComboBox1.AddItem "Ligne" & i
    Next i
End Sub
```

Affecter une valeur par défaut lors de l'affichage du ComboBox.

Vba
```
ComboBox1.ListIndex = 0
'L'index 0 correspond à la première donnée contenue dans le ComboBox
```

Remplir un ComboBox sans doublon.

Vba
```
Private Sub UserForm_Initialize()
    Dim j As Integer

    'Récupère les données de la colonne A...
    For j = 1 To Range("A65536").End(xlUp).Row
        ComboBox1 = Range("A" & j)
        '...et filtre les doublons
        If ComboBox1.ListIndex = -1 Then ComboBox1.AddItem Range("A" & j)
    Next j
End Sub
```

Supprimer tous les items contenus dans une ComboBox.

Vba
```
ComboBox1.Clear
```

Récupérer le contenu de l'item sélectionné.
La procédure renvoie une chaîne vide si rien n'est affiché dans le ComboBox.

Vba
```
Private Sub CommandButton1_Click()
    MsgBox ComboBox1.Value
End Sub
```

II-G - ListBox

Le contrôle ListBox (Zone de liste) permet de choisir un ou plusieurs éléments dans une liste de choix.
Utiliser la propriété RowSource pour remplir une Listbox.

Vba
```
Private Sub UserForm_Initialize()
    'Remarque:
    'ColumnHeads = True spécifie que la première cellule précédent
    'la plage source est utilisée comme titre dans la ListBox.
    'ColumnHeads ne fonctionne pas pour la propriété List() et la
    'méthode AddItem.
```

```vba
    ListBox1.ColumnHeads = True

    ListBox1.RowSource = "Feuil1!A2:A10"
End Sub
```

Utiliser La méthode AddItem pour Remplir une ListBox.

```vba
Private Sub UserForm_Initialize()
    Dim i As Byte

    For i = 1 To 5
        ListBox1.AddItem "Ligne" & i
    Next i
End Sub
```

Remarque:
La méthode AddItem possède un 2eme argument falcutatif qui indique à quelle ligne doit être placée la nouvelle donnée. Si l'argument n'est pas spécifié (comme dans l'exemple précédent), chaque nouvelle ajout vient s'insérer en dernière ligne, à la suite des valeurs existantes.
L'exemple suivant place les données dans la 1ere ligne de la ListBox.

```vba
ListBox1.AddItem "donnée test", 0
```

Utiliser la propriété List pour alimenter une ListBox.

```vba
Private Sub UserForm_Initialize()
 ListBox1.List() = Range("A1:A10").Value
End Sub
```

Récupérer le contenu de la ligne sélectionnée.
Si aucune ligne n'est sélectionnée, la macro renvoie une erreur. Pour y remédier il est possible de tester préalablement la valeur listIndex:
Si aucune ligne n'est sélectionnée, ListIndex = -1. On peut donc écrire : If listBox1.listIndex = -1 Then Exit Sub
La valeur ListIndex est égale à 0 pour la 1ere ligne , 1 pour la 2eme ligne ...etc...

```vba
If listBox1.listIndex = -1 Then Exit Sub
MsgBox ListBox1.List(ListBox1.ListIndex)
```

Compter le nombre de données contenues dans la ListBox.

```vba
MsgBox ListBox1.listCount
```

Sélectionner la 3eme ligne dans une ListBox.

```vba
ListBox1.ListIndex = 2
```

Afficher le 3eme item de la Listbox en haut dans la zone visible

Vba
```
ListBox1.TopIndex = 2
```

Boucler sur toutes les données d'une ListBox.

Vba
```
Private Sub CommandButton1_Click()
    Dim i As Integer

    'Les index des Listbox commencent par zéro
    For i = 0 To ListBox1.ListCount - 1
        Debug.Print ListBox1.List(i)
    Next i
End Sub
```

Transférer les données d'une ListBox dans la Feuille de calcul.

Vba
```
Private Sub CommandButton1_Click()
    With ListBox1
        Sheets("Feuil1").Range(Cells(1, 1), Cells(.ListCount, 1)) = .List
    End With
End Sub
```

Supprimer un élément dans une ListBox.
L'exemple ci-dessous enlève un Item lors du double clic sur la ligne.

Vba
```
Private Sub Listbox1_Dblclick(ByVal Cancel As MSForms.ReturnBoolean)
    'Remarque:
    'La propriété RowSource n'accepte pas cette méthode
    ListBox1.RemoveItem (ListBox1.ListIndex)
End Sub
```

Déplacer les Item de la Listbox d'un index vers le haut lors d'un double clic sur la ligne.

Vba
```
Private Sub Listbox1_dblClick(ByVal Cancel As MSForms.ReturnBoolean)
    Dim Cible As Integer

    On Error Resume Next

    With ListBox1
        If .ListIndex < 0 Then Exit Sub
        Cible = .ListIndex
        If Cible = 0 Then Exit Sub

        .AddItem .Text, Cible - 1
        .RemoveItem Cible + 1
        .Selected(Cible - 1) = True
    End With
End Sub
```

Déplacer un Item de la Listbox n'importe où dans la liste.

Vba
```
Dim Cible As Boolean
```

Vba
```
Dim Valeur As String

'Le premier double clic enregistre l'item sélectionné dans une variable,
'puis supprime la ligne.
'Le second Double Clic insère la variable en mémoire sous la ligne sélectionnée.

Private Sub Listbox1_Dblclick(ByVal Cancel As MSForms.ReturnBoolean)
    If Cible = False Then
            Cible = True
            Valeur = ListBox1
            ListBox1.RemoveItem ListBox1.ListIndex
        Else
            Cible = False
            ListBox1.AddItem Valeur, ListBox1.ListIndex + 1
    End If
End Sub
```

Pour autoriser la mutisélection dans une Listbox, vous devez préalablement indiquer 1_fmMultiSelectMulti dans la propriété "Multiselect" de la Listbox.
Un exemple pour boucler sur les lignes sélectionnées dans la Listbox.

Vba
```
Private Sub commandButton1_Click()
    Dim i As Byte

    'boucle sur les éléments de la listbox
    For i = 0 To ListBox1.ListCount - 1
        If ListBox1.Selected(i) = True Then MsgBox ListBox1.List(i)
    Next i
End Sub
```

Créer une ListBox multicolonnes.

Vba
```
Private Sub UserForm_Initialize()
    Dim i As Byte, j As Byte

    'Définit le nombre de colonnes dans la ListBox
    ListBox1.ColumnCount = 7

    '---
    'Définit la largeur des colonnes d'une ListBox:
    'Par défaut, la largeur des colonnes est de 72 points
    '(72 points = 1 pouce)
    ListBox1.ColumnWidths = "50;80;50;60;50;70;50"
        '---
        'Il est aussi possible de définir la dimension des colonnes en centimètres
        'ListBox1.ColumnWidths = "2 cm; 1,5 cm ......"
        '---
    '---

    For i = 1 To 20
        'Ajoute une ligne et insère une donnée dans la colonne de gauche
        ListBox1.AddItem "Ligne" & i

        'Ajoute des données dans les colonnes de droite
        For j = 1 To 7
        ListBox1.List(ListBox1.ListCount - 1, j) = i & j
        Next j

    Next i
End Sub
```

Extraire la valeur de la 3eme colonne d'une Listbox, dans la ligne sélectionnée.

Vba
```
MsgBox ListBox1.List(ListBox1.ListIndex, 2)
```

Créer des séparateurs de colonne dans une ListBox multicolonnes.
Il n'existe pas de propriété pour cela, mais vous pouvez tester ce subterfuge: (Une autre solution consiste à utiliser les ListView)

Vba
```
Private Sub UserForm_Initialize()
Dim i As Byte, j As Byte

ListBox1.ColumnCount = 7
ListBox1.ColumnWidths = "50;15;50;15;50;15;50"

For i = 1 To 20
    ListBox1.AddItem "Ligne" & i

    For j = 2 To 7 Step 2
    ListBox1.List(ListBox1.ListCount - 1, j) = i & j
    Next j

    For j = 1 To 6 Step 2 'boucle pour créer les "séparateurs" de colonnes
    ListBox1.List(ListBox1.ListCount - 1, j) = Chr(124)
    Next j
Next i
End Sub
```

Synchroniser l'affichage entre 2 ListBox. Quand un item est sélectionné dans une des 2 listes, le même item est activé dans l'autre.

Vba
```
Private Sub ListBox2_Click()
    With ListBox1
        .TopIndex = ListBox2.TopIndex
        .ListIndex = ListBox2.ListIndex
    End With
End Sub


Private Sub ListBox1_Click()
    With ListBox2
        .TopIndex = ListBox1.TopIndex
        .ListIndex = ListBox1.ListIndex
    End With
End Sub
```

Afficher la fin de la ListBox

Vba
```
ListBox1.TopIndex = ListBox1.ListCount
```

Remplir une ListBox sans doublons

Vba
```
Private Sub UserForm_Initialize()
    Dim Cell As Range
    Dim Unique As New Collection
    Dim Valeur As Range
    Dim i As Integer
```

Vba

```
    'Récupère la derniere ligne non vide dans la colonne A
    i = Range("A65536").End(xlUp).Row

    On Error Resume Next
    'boucle sur les cellules de la colonne A
    For Each Cell In Range("A1:A" & i)
        'Stocke les données dans une collection
        '(La collection n'accepte que des données uniques et permet donc
        ' de filtrer facilement les doublons).
        Unique.Add Cell, CStr(Cell)
    Next Cell
    On Error GoTo 0

    'Boucle sur le contenu de la collection pour alimenter la ListBox
    For Each Valeur In Unique
        Me.ListBox1.AddItem Valeur
    Next Valeur
End Sub
```

II-H - ScrollBar

Le contrôle ScrollBar (Barre de défilement) permet d'incrémenter ou de décrémenter des valeurs en fonction de spécifications (valeur mini, valeur maxi, pas).
Cet exemple spécifie les paramètres du ScrollBar lors de l'initialisation du UserForm et l'évènement ScrollBar1_Change permet d'afficher les modifications dans un Label.

Vba

```
Private Sub UserForm_Initialize()
    With ScrollBar1
        .Min = 0 'Valeur mini
        .Max = 100 'Valeur maxi

        'Spécifie la distance de déplacement intervenant lorsque l'utilisateur
        'clique entre le curseur de défilement et la flèche de défilement.
        .LargeChange = 10

        'Spécifie le déplacement se produisant lorsque l'utilisateur clique sur
        'les flèches de défilement dans le contrôle ScrollBar
        '(La valeur par défaut = 1)
        .SmallChange = 5
    End With
End Sub


Private Sub ScrollBar1_Change()
    Label1 = ScrollBar1.Value
End Sub
```

II-I - SpinButton

Le contrôle SpinButton (Toupie) permet d'incrémenter ou de décrémenter des valeurs en fonction de spécifications (valeur mini, valeur maxi, pas).
Cet objet fonctionne sur le même principe que le ScrollBar, mais ne possède pas de barre de défilement. Il ne possède pas de propriété LargeChange.
Cet exemple spécifie les paramètres du SpinButton lors de l'initialisation du UserForm et l'évènement SpinButton1_Change permet d'afficher les modifications dans un Label.

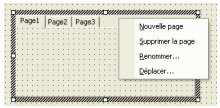
Vba
```
Private Sub UserForm_Initialize()
    With SpinButton1
        .Min = 0 'Valeur mini
        .Max = 100 'Valeur maxi

        'Spécifie le déplacement se produisant lorsque l'utilisateur clique sur
        'les flèches de défilement dans le contrôle SpinButton
        '(La valeur par défaut = 1)
        .SmallChange = 5
    End With
End Sub


Private Sub SpinButton1_Change()
    Label1 = SpinButton1.Value
End Sub
```

II-J - MultiPage

Un MultiPage, comme son nom l'indique, est constitué de plusieurs pages. Ce contrôle est particulièrement intéressant lorsque vous devez gérer un grand nombre d'objets dans la Forme.
Pour ajouter une page, faites un clic droit dans la barre d'onglets puis Sélectionnez l'option "Nouvelle page".



L'indexation des pages:
- 0 = Première page
- 1 = Deuxième page
- 2 = Troisième page

Par exemple, se positionner sur la Page3 lors de l'ouverture de l'USF.

Vba
```
Private Sub userForm_Initialize()
    Me.MultiPage1.Value = 2
End Sub
```

Empêcher l'accès à la Page2.

Vba
```
Me.MultiPage1.Pages(1).Enabled = False
```

Masquer la Page2.

Vba
```
Me.MultiPage1.Pages(1).Visible = False
```

Vérifier si la Page2 est active.

Vba
```
If Me.MultiPage1.SelectedItem.Index = 1 Then _
            MsgBox "La page 2 est active"
```

Afficher le nom de la page sélectionnée.

Vba

```
Private Sub MultiPage1_Change()
    MsgBox MultiPage1.SelectedItem.Name
End Sub
```

Ajouter une page par macro et compter le nombre de pages.

Vba

```
Private Sub CommandButton1_Click()
    Dim Pge As Page

    'Ajoute une page
    Set Pge = MultiPage1.Pages.Add
    Pge.Caption = "Nouvelle page"

    'compte le nombre de pages
    MsgBox MultiPage1.Pages.Count
End Sub
```

Afficher le nom du contrôle qui a le focus dans la page active.

Vba

```
MsgBox MultiPage1.SelectedItem.ActiveControl.Name
```

II-K - Image

Ce contrôle permet de visualiser des images dans l'UserForm.
Remarque:
Il est toujours préférable de charger une image externe. Evitez (si possible) de sauvegarder les images dans le classeur pour ne pas en augmenter le poids.
La propriété PictureSizeMode définit le mode d'affichage:

Constante	Valeur	Description
fmPictureSizeModeClip	0	(valeur par défaut). L'image n'est pas redimensionnée.
fmPictureSizeModeStretch	1	Étire l'image pour l'adapter aux dimensions du contrôle . Cette définition déforme l'image dans le sens horizontal ou vertical.
fmPictureSizeModeZoom	3	Agrandit l'image en fonction des dimensions du contrôle mais ne la déforme ni dans le sens horizontal, ni dans le sens vertical.

Attention: Certains formats de fichiers ne sont pas reconnus: PNG, TIF...
Vous pouvez utiliser les images GIF, mais elles ne seront pas animées. Pour visualiser une image GIF animée, utilisez par exemple l'objet WebBrowser.
Charger une image:

Vba

```
Private Sub CommandButton1_Click()
    Dim Fichier As String
```

```vba
    Fichier = "C:\fourmiz.jpg"

    'Vérifie si le fichier existe.
    If Dir(Fichier) <> "" Then
        'si le fichier existe, il est chargé pour visualisation.
        Image1.Picture = LoadPicture(Fichier)
        Else
        'Sinon, affiche aucune image.
        Image1.Picture = LoadPicture("")
    End If
End Sub
```

Décharger une image de son contrôle Image.

```vba
Set Image1.Picture = Nothing
```

Utiliser des Scrollbars pour se déplacer dans une image dont la taille est supérieure à celle du contrôle:
Lorsque vous utilisez l'objet Image, vous pouvez uniquement visualiser le résultat en mode Stretch, Zoom ou Clip. Pour visualiser un fichier à sa taille réelle et avoir la possibilité de s'y déplacer même si sa dimension est superieure à celle de l'objet, insérez le contrôle Image dans un Frame puis utilisez la macro ci dessous.

```vba
Private Sub UserForm_Initialize()
    Image1.AutoSize = True
    Image1.Picture = LoadPicture("C:\fourmiz.jpg")

    With Me.Frame1
        .ScrollBars = fmScrollBarsBoth
        .ScrollHeight = Image1.Height
        .ScrollWidth = Image1.Width
    End With
End Sub
```

II-L - Frame

Les Frames (Cadres) servent à regrouper les contrôles de façon logique. Par exemple, si vous avez besoin de masquer un groupe de contrôles, placez les dans un Frame et masquez ce dernier: Les objets seront aussi cachés. Les Frames sont souvent utilisés pour gérer les groupes d'OptionButton.
Boucler sur tous les objets d'un Frame.

```vba
Private Sub CommandButton1_Click()
    Dim i As Integer

    For i = 0 To Frame1.Controls.Count - 1
        MsgBox Frame1.Controls.Item(i).Name
    Next i
End Sub
```

II-M - RefEdit

Le contrôle RefEdit permet de récupérer la référence à une plage de cellules.

En cliquant sur l'objet vous obtenez une fenêtre invitant à sélectionner une cellule ou une plage de cellules. Après avoir refermé cette fenêtre, vous pouvez manipuler la plage de cellules cible. Cet exemple insère une croix "x" dans la plage sélectionnée par le RefEdit.

Vba

```
Private Sub CommandButton1_Click()
    Dim Plage As String

    Plage = RefEdit1.Value
    'Vérifie s'il y a eu une sélection
    If Plage = "" Then
        MsgBox "Opération annulée"
        Exit Sub
    End If

    'Insère une croix dans toutes les cellules de la plage
    Range(Plage) = "x"
    Unload Me
End Sub
```

II-N - ToggleButton

Le contrôle ToggleButton (Bouton bascule) permet de renvoyer les valeurs:
Vrai (Lorsque le bouton est activé)
Faux (Lorsque le bouton est désactivé)
Dans cet exemple, la couleur et le texte du contrôle sont modifiés à chaque changement de statut.

Vba

```
Private Sub ToggleButton1_Click()
    With ToggleButton1
        If .Value = True Then
            .BackColor = RGB(0, 255, 0) 'Vert
            .Caption = .Value
        ElseIf .Value = False Then
            .BackColor = RGB(255, 0, 0) 'Rouge
            .Caption = .Value
        End If
    End With
End Sub
```

III - Les contrôles spécifiques

Certains des contrôles décrits dans ce chapitre nécessitent une licence et il vous appartient de vérifier ce point avant de distribuer votre projet.

III-A - WebBrowser

La documentation MSDN
Tutoriel en cours de rédaction.

III-B - Office Web Components (OWC)

Le complément Microsoft Office Web Components (Composants Web) est une collection de contrôles (Component Object Model ou COM) permettant de publier sur le Web des feuilles de calcul, des graphiques et des tableaux croisés dynamiques. Ces objets sont aussi utilisables en VBA.
Chaque version d'OWC correspond à une version d'Office:

- Office2000 : OWC9
- OfficeXP : OWC10
- Office2003 : OWC11

Le complément d'Office 2002: Composants Web Office
Le complément d'Office 2003: composants Web Office

III-B-1 - ChartSpace

L'objet ChartSpace permet de visualiser des graphiques dans un UserForm.
Téléchargez un classeur démo.

III-B-2 - SpreadSheet

L'objet SpreadSheet permet d'utiliser des tableurs Excel dans un UserForm.
Tutoriel en cours de rédaction.

III-B-3 - PivotTable

L'objet PivotTable permet d'utiliser des tableaux croisés dynamiques dans un UserForm.
Consultez le tutoriel.

III-C - WindowMediaPlayer

Utiliser Windows Media Player en VBA

III-D - ShockwaveFlash

Piloter une animation Flash en VBA

III-E - Windows Image acquisition

Utiliser la librairie Windows Image Acquisition en VBA

III-F - TreeView

Le contrôle TreeView permet d'afficher des informations hiérarchisées sous forme d'arborescence. Le TreeView est constitué de noeuds (parents et enfants).
Consultez le praticiel de Jacma: Maîtriser le composant TreeView.

III-G - ListView

Le contrôle ListView permet d'afficher des informations (dans le style du volet de droite de l'explorateur Windows).
Utiliser le contrôle ListView en VBA

III-H - ImageList

Le contrôle ImageList permet de stocker et gérer des images dans un classeur.
Remarque:
Il est toujours préférable de charger une image externe. Evitez (si possible) de sauvegarder les images dans le classeur pour ne pas en augmenter le poids.
Utiliser le contrôle ImageList en VBA

III-I - Les calendriers

III-I-1 - Calendar

Le contrôle Calendar permet d'afficher un calendrier dans un UserForm.
Vous pouvez utiliser cet objet en sélectionnant "Contrôle Calendrier xx.x" dans la liste des contrôles supplémentaires (xx.x dépend de votre version d'Office).
Cet exemple permet d'initialiser la date du jour dans le calendrier et insère la date sélectionnée dans un TextBox.

Vba

```
Private Sub UserForm_Initialize()
 'spécifie la date du jour lors de l'affichage de l'USF
 Calendar1.Value = Now
End Sub

Private Sub Calendar1_Click()
    TextBox1.Value = Calendar1.Value
End Sub
```

III-I-2 - Monthview

Le contrôle MonthView permet aussi d'afficher un calendrier dans un UserForm.
Vous pouvez utiliser cet objet en sélectionnant "Microsoft MonthView Control 6.0" dans la liste des contrôles supplémentaires.
Cet exemple permet d'initialiser la date du jour dans le calendrier et insère la date sélectionnée dans un TextBox.

Vba

```
Private Sub UserForm_Initialize()
 'spécifie la date du jour lors de l'affichage de l'USF
 MonthView1.Value = Now
End Sub

Private Sub MonthView1_DateClick(ByVal DateClicked As Date)
```

```vba
    TextBox1.Value = DateClicked
End Sub
```

III-I-3 - DatePicker

Le contrôle DatePicker permet d'afficher un calendrier déroulant dans un UserForm.
Vous pouvez utiliser cet objet en sélectionnant "Microsoft Date and Time Picker Control 6.0" dans la liste des contrôles supplémentaires.
Cet exemple permet d'initialiser la date du jour dans le calendrier et affiche la date sélectionnée.

```vba
Private Sub UserForm_Initialize()
 'spécifie la date du jour lors de l'affichage de l'USF
 DTPicker1.Value = Now
End Sub

Private Sub DTPicker1_Change()
        MsgBox DTPicker1.Value
End Sub
```

III-J - StatusBar

Le contrôle StatusBar permet (en autre) d'afficher l'heure au format hh:mm, dans un UserForm.
Sélectionnez la ligne "Microsoft StatusBar control ,Version 6.0" dans la liste des contrôles supplémentaires.
Placez l'objet dans l'UserForm.
Cliquez sur "(Personnalisé)" dans la fenêtre Propriétés.
Sélectionnez l'onglet "Zones" (ou Panels) dans la boîte de dialogue.
Choisissez l'option "5-sbrTime" dans le champ Style.
Cliquez sur "Appliquer", puis sur le bouton OK.
L'heure est affichée dans l'UserForm.

III-K - ProgressBar

Téléchargez un classeur démo

IV - Liens

La FAQ

La page Sources

V - Téléchargement