

VBA EXCEL Création d'une barre de menus personnalisée

par Fred (fring) ([autres articles](#))

Date de publication : 31/05/2008

Dernière mise à jour : 10/06/2008

Au lieu d'insérer des boutons de commande directement sur la feuille Excel, nous allons voir dans cet article comment créer une barre de menus et y insérer des contrôles.
(Réalisé et testé sur Excel 2000)

1 - Création de la barre de menus.....	4
1.1 - Syntaxe générale.....	4
1.2 - Les arguments.....	4
1.2.1 - Name.....	4
1.2.2 - Position.....	4
1.2.3 - MenuBar.....	5
1.2.4 - Temporary.....	6
1.3 - Les propriétés.....	6
1.3.1 - Visible.....	6
1.3.2 - Enabled.....	7
1.3.3 - Protection.....	7
1.3.4 - FindControl.....	8
1.3.5 - ActionControl.....	8
1.3.6 - Builtin.....	8
1.3.7 - RowIndex.....	9
1.4 - Exemples de code.....	9
2 - Les contrôles.....	11
2.1 - Syntaxe générale.....	11
2.2 - Le bouton de commande et ses propriétés.....	11
2.2.1 - Style.....	12
2.2.2 - FaceID.....	14
2.2.3 - Icône personnalisée.....	14
2.2.4 - ID (argument).....	15
2.2.5 - Caption.....	15
2.2.6 - OnAction.....	15
2.2.7 - TooltipText.....	16
2.2.8 - HyperlinkType.....	16
2.2.9 - BeginGroup.....	16
2.2.10 - Copy et CopyFace.....	17
2.2.11 - Move.....	17
2.2.12 - Builtin et BuiltinFace.....	18
2.2.13 - Enabled.....	18
2.2.14 - Visible.....	18
2.2.15 - Before (argument).....	18
2.2.16 - Priority.....	19
2.2.17 - Tag.....	19
2.2.18 - DescriptionText.....	19
2.2.19 - State.....	19
2.3 - La zone de texte et ses propriétés.....	19
2.3.1 - Style.....	20
2.3.2 - Text.....	20
2.3.3 - Clear.....	21
2.3.4 - Width.....	21
2.4 - La zone de liste et ses propriétés.....	21
2.4.1 - Style.....	21
2.4.2 - AddItem.....	21
2.4.3 - Text.....	22
2.4.4 - ListIndex.....	22
2.4.5 - ListCount.....	23
2.4.6 - List(index).....	23
2.4.7 - RemoveItem(index).....	23
2.4.8 - Clear.....	24
2.4.9 - ListHeaderCount.....	24
2.4.10 - DropDownLines.....	24
2.4.11 - DropDownWidth.....	25
2.5 - Les sous-menus.....	25
2.6 - Exemple de code.....	25
3 - Menu Popup.....	28

4 - Les barres de menus prédéfinies.....	31
5 - Remerciements.....	32
VI - Téléchargements.....	33

1 - Création de la barre de menus

La première étape consiste à créer et ajouter à l'application, dans notre cas Excel, une nouvelle barre de menus qui comportera les contrôles personnalisés.

Les contrôles, c'est à dire les boutons, zones de texte, zones de liste et sous-menus sont détaillés au chapitre 2.

1.1 - Syntaxe générale

Syntaxe : **CommandBars.Add**(Name, [Position], [MenuBar], [Temporary])

les arguments entre [] sont facultatifs

Deux méthodes d'utilisation de cette syntaxe sont possibles, avec ou sans initialisation d'une variable :

Code VBA sans variable

```
Sub Ajout_BarreMenu()  
  
CommandBars.Add(Name, [Position], [MenuBar], [Temporary]).Propriété = ValeurPropriété  
  
End Sub
```

Code VBA avec initialisation d'une variable

```
Sub Ajout_BarreMenu()  
Dim Cbar As CommandBar  
  
Set Cbar = CommandBars.Add(Name, [Position], [MenuBar], [Temporary])  
Cbar.Propriété = ValeurPropriété  
  
End Sub
```

Table des matières

1.2 - Les arguments

1.2.1 - Name

L'argument **Name** attribue un nom à votre nouvelle barre de menus.

Dans la mesure du possible, optez pour un nom explicite et concis, ce qui par la suite, facilitera son utilisation dans le code.

Exemple d'utilisation : **Name:="MaBarre"**

Top

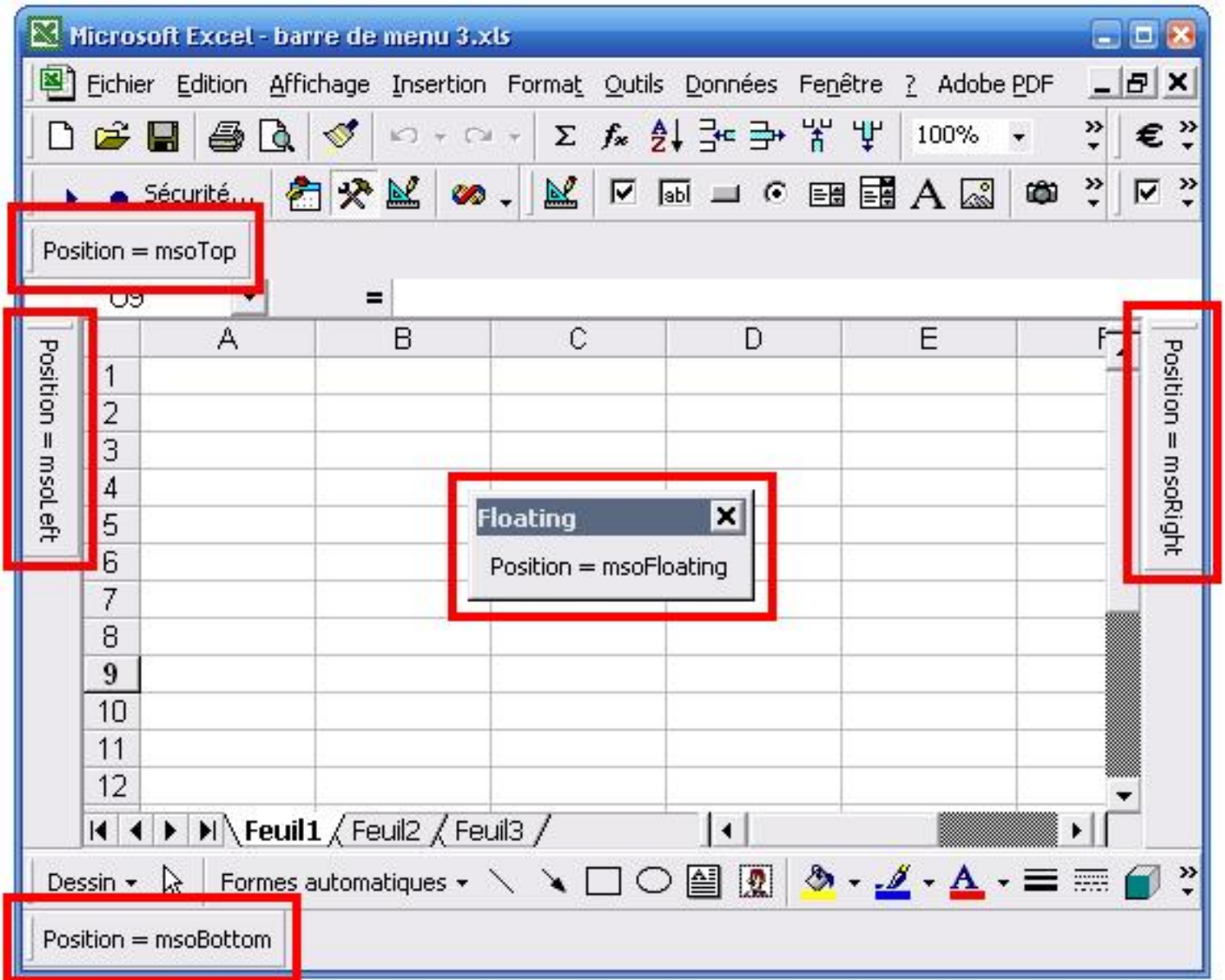
1.2.2 - Position

L'argument **Position** va définir, comme son nom l'indique..., la position de cette nouvelle barre de menus.

Six **constantes** sont possibles :

- **msoBarTop** = en haut (sous les barres de menus déjà en place)
- **msoBarBottom** = en bas
- **msoBarRight** = à droite
- **msoBarLeft** = à gauche
- **msoBarFloating** = position flottante sur l'écran (*argument par défaut*)

- **msoBarPopup** = position particulière, détaillée en fin d'article au chapitre 3.



Cet argument est facultatif, sans le préciser c'est la valeur par défaut (msoBarFloating) qui sera prise en compte.
Exemple d'utilisation : **Position:=msoBarTop**

Top

1.2.3 - MenuBar

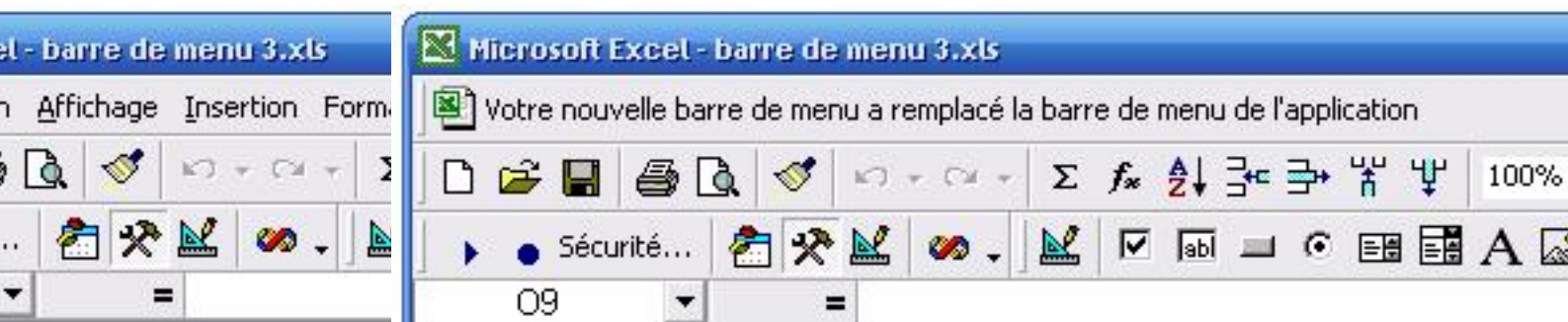
Argument de type booléen permettant, si **MenuBar:=True**, de remplacer le menu de l'application par votre barre de menus personnalisée.

Cet argument est facultatif, valeur par défaut = False.

Aperçu :

Menu original de l'application

Barre de menu personnalisée remplaçant le menu original



Top

1.2.4 - Temporary

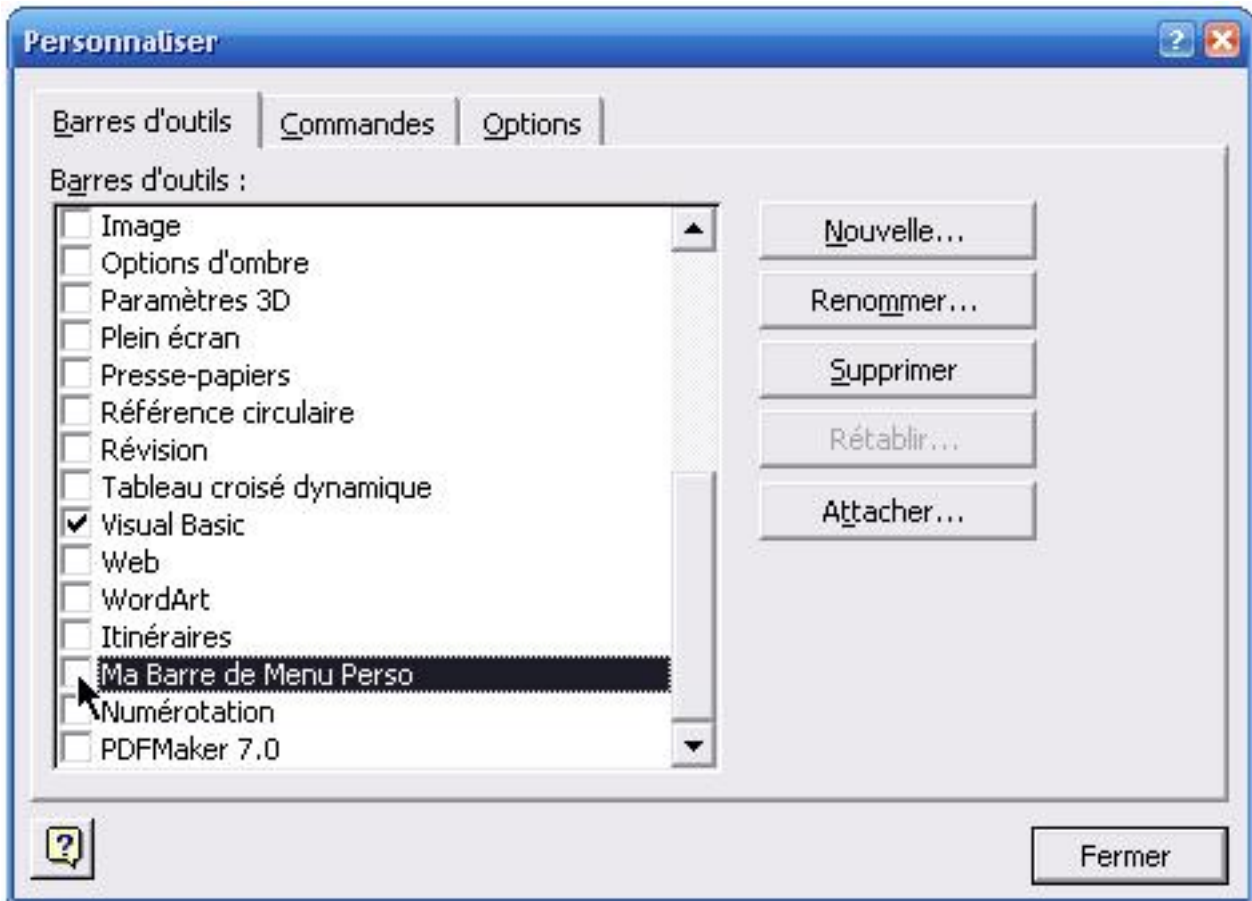
Argument de type booléen permettant de préciser, si **Temporary=True**, que la barre de menus est temporaire. Lors de la fermeture de l'application, la barre de menus sera alors automatiquement supprimée. Cet argument est facultatif, valeur par défaut = False.

Top

1.3 - Les propriétés

1.3.1 - Visible

La propriété **Visible**, de type booléen, permet tout simplement de rendre votre barre de menus visible ou non. Si, lors de la création de la barre de menus, vous omettez de préciser cette propriété, son état par défaut étant "False", la barre de menus sera belle et bien créée mais ne sera pas visible. Dans ce cas, vous pouvez toujours la rendre visible manuellement via le menu **Outils >> Personnaliser** en cochant la barre de menus en question.



Exemple d'utilisation : **CommandBars.Add(Name, [Position], [MenuBar], [Temporary]).Visible = True**

Top

1.3.2 - Enabled

La propriété **Enabled**, de type booléen, a le même effet visuel que la propriété **Visible**.

La différence entre les deux propriétés réside dans le fait que si **Enabled = False**, non seulement la barre de menus ne sera pas visible,

mais en plus elle n'apparaîtra pas dans le menu Outils >> Personnaliser.

Par défaut, l'état de la propriété "Enabled" est "True".

Top

1.3.3 - Protection


La propriété **Protection** va empêcher l'utilisateur d'effectuer des modifications.

Cinq types de protections sont intéressants, les **constantes** sont :

- **msoBarNoCustomize** : empêchera l'utilisateur d'ajouter manuellement un bouton à la barre de menus.
- **msoBarNoMove** : empêchera l'utilisateur de déplacer la barre de menus.
- **msoBarNoChangeDock** : permettra à l'utilisateur de déplacer la barre de menus uniquement sur son axe.
- **msoBarNoVerticalDock** : n'autorisera pas de positionner la barre de menus, verticalement, à gauche ou à droite de la feuille.

- **msoNoHorizontalDock** : n'autorisera pas de positionner la barre de menus, horizontalement, en haut ou en bas de la feuille.

Exemple d'utilisation : **CommandBars.Add(Name, [Position], [MenuBar], [Temporary]).Protection = msoBarNoMove**

 Vous pouvez cumuler plusieurs types de protections, par exemple :
CommandBars.Add(Name, [Position], [MenuBar], [Temporary]).Protection = msoBarNoMove + msoBarNoCustomize

Top

1.3.4 - FindControl

La propriété **FindControl** permettra de trouver un contrôle spécifique afin par exemple de le supprimer.
 La syntaxe est la suivante : **CommandBars("NomDeLaBarre").FindControl(Type, Id, Tag, Visible, Recursive)**
 Chaque argument est facultatif mais minimum un argument doit être utilisé.
 Explication des arguments :

- **Type:=** recherche d'un type de contrôle (*voir chapitre 2*)
- **Id:=** recherche sur un numéro d'index de bouton prédéfini (*voir chapitre 2.2.4*)
- **Tag:=** recherche d'une étiquette attribuée au contrôle (*voir chapitre 2.2.17*)
- **Visible:=** recherche d'un contrôle visible ou non, par défaut l'état de cet argument est True (*voir chapitre 2.2.14*)
- **Recursive:=** True permet de rechercher également dans les sous-menus, par défaut l'état de cet argument est False

Exemple d'utilisation : **CommandBars("MaBarre").FindControl(Type:=msoControlButton, Tag:="bout1", Recursive:=True).Delete**

Top

1.3.5 - ActionControl

La propriété **ActionControl** renvoie, dans la procédure, la propriété désignée du contrôle qui vient d'être actionné pour exécuter cette procédure.
 Syntaxe : **CommandBars.ActionControl.propriété** (*voir chapitre 2 pour plus de détails sur les différentes propriétés*)
 Exemple : *MaVariable = CommandBars.ActionControl.Caption*

Top

1.3.6 - Builtin

La propriété **Builtin**, de type booléen, renvoie la valeur **True** si la barre de menus est une des barres prédéfinies de l'application.
 A quoi peut bien servir cette propriété ?
 Si l'on souhaite supprimer toutes les barres de menus créées lors de la fermeture du classeur, nous pouvons le réaliser via cette propriété,
 ce qui permettra de supprimer toutes les barres de menus persos sans devoir se soucier du nombre ni de leurs noms.
 Exemple d'utilisation :

Code VBA à insérer dans un module standard

Code VBA à insérer dans un module standard

```
Sub Sup_Cbar()
Dim Cbar As CommandBar

For Each Cbar In CommandBars
If Cbar.BuiltIn = False Then Cbar.Delete
Next

End Sub
```

Code VBA à insérer dans l'objet ThisWorkbook

```
Private Sub Workbook_BeforeClose(Cancel As Boolean)
Sup_Cbar
End Sub
```

[Top](#)

1.3.7 - RowIndex

En cas d'insertion de plusieurs barres de menus sur une ligne, cette propriété permet de les positionner.
 Syntaxe : `mabarre.RowIndex = 2` (pour positionner la barre de menus en 2ème position en partant de la gauche).

[Top](#)

1.4 - Exemples de code

Exemple de création d'une barre de menus, positionnement de celle-ci dans la partie supérieure et ajout de protections.

Code de création :

Code VBA à insérer dans un module standard

```
Sub Creation_Cbar()
Dim Cbar As CommandBar

Set Cbar = CommandBars.Add(Name:="MaBarre", Position:=msoBarTop, Temporary:=True)

With Cbar
.Visible = True
.Protection = msoBarNoMove + msoBarNoCustomize
End With

End Sub
```

Code de suppression :

Code VBA à insérer dans un module standard

```
Sub Supp_Cbar()

On Error Resume Next ' <-- gestion d'erreur au cas où la barre n'existe pas
CommandBars("Mabarre").Delete

End Sub
```

Code permettant de créer automatiquement la barre de menus à l'ouverture du classeur :

Code VBA à insérer dans l'objet ThisWorkbook

```
Private Sub Workbook_Open()
```

Code VBA à insérer dans l'objet ThisWorkbook

```
Creation_Cbar  
End Sub
```

L'argument "Temporary" étant fixé à "True", la barre sera automatiquement supprimée lors de la fermeture de l'application.

Vous pouvez malgré tout décider de supprimer cette barre de menus lors de la fermeture du classeur.

Code permettant de supprimer la barre de menus à la fermeture du classeur :

Code VBA à insérer dans l'objet ThisWorkbook

```
Private Sub Workbook_BeforeClose(Cancel As Boolean)  
    Supp_Cbar  
End Sub
```

Top

2 - Les contrôles

La barre de menus ne serait rien sans contrôles.

Cinq Types de contrôles sont possibles, les **constantes** sont :

- **msoControlButton** = bouton de commande
- **msoControlEdit** = zone de texte
- **msoControlDropDown** = zone de liste
- **msoControlComboBox** = zone de liste modifiable
- **msoControlPopup** = sous-menu

Les **Propriétés** pour chacun de ces contrôles sont détaillées ci-dessous.

2.1 - Syntaxe générale

Syntaxe : **CommandBars("MaBarre").Controls.Add(Type)**

voir ci-dessus les constantes (types) possibles

Deux méthodes d'utilisation de cette syntaxe sont possibles, avec ou sans initialisation d'une variable :

Code VBA sans variable

```
Sub Ajout_Controle()

With CommandBars("MaBarre").Controls.Add(Type)
    .Propriété1 = ValeurPropriété1
    .Propriété2 = ValeurPropriété2
    'etc...
End With

End Sub
```

Code VBA avec initialisation d'une variable

```
Sub Ajout_Controle()
Dim Ctrl1 As CommandBarButton ' <-- déclaration de variable pour un contrôle de type bouton de commande
Dim Ctrl2 As
CommandBarComboBox ' <-- déclaration de variable pour un contrôle de type zone de texte ou zone de liste
Dim Ctrl3 As CommandBarPopup ' <-- déclaration de variable pour un contrôle de type sous-menu

Set Ctrl2 = CommandBars("MaBarre").Controls.Add(Type)

With Ctrl2
    .Propriété1 = ValeurPropriété1
    .Propriété2 = ValeurPropriété2
    'etc...
End With

End Sub
```

[Top](#)

2.2 - Le bouton de commande et ses propriétés

Pour rappel, la constante définissant le contrôle type "bouton de commande" est : **msoControlButton**

Syntaxe : **CommandBars("MaBarre").Controls.Add(msoControlButton)**












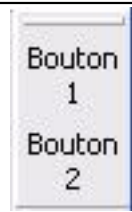


2.2.1 - Style



La propriété **Style** définit le mode d'affichage du bouton de commande.

Syntaxe : *monbouton*.**Style** = **constante**

Exemple : *Style* = *msoButtonIconAndCaption*

8 **constantes** sont possibles :

Constante	Définition	Aperçu barre horizontale	Aperçu barre verticale
msoButtonAutomatic	bouton avec icône <i>constante par défaut</i>		
msoButtonIcon	bouton avec icône		
msoButtonCaption	bouton avec texte		
msoButtonIconAndCaption	bouton avec icône + texte <i>le texte se positionne à droite de l'icône</i>		
msoButtonIconAndCaptionBelow	bouton avec icône + texte <i>le texte se positionne sous l'icône en position verticale, le texte reste horizontal</i>		
msoButtonWrapCaption	bouton avec texte <i>2 lignes de texte si plusieurs mots en position verticale, le texte reste horizontal</i>		
msoButtonIconAndWrapCaption	bouton avec icône + texte <i>2 lignes de texte si plusieurs mots en position verticale, le texte reste horizontal</i>		
msoButtonIconAndWrapCaptionBelow			

	<p align="center"> bouton avec icône + texte <i>le texte se positionne sous l'icône 2 lignes de texte si plusieurs mots en position verticale, le texte reste horizontal</i> </p>		
--	---	--	---

Top

2.2.2 - FaceID

Les **FaceID** sont les icônes prédéfinies, incluses dans MS Office.

Chaque icône est identifiée par un numéro d'index. Il y en a environ...euh...je vous laisse le soin de les compter.

Syntaxe : *monbouton.FaceID = index*

Exemple : *FaceID = 343 (icône représentant une ampoule, utilisée dans le tableau des styles ci-dessus)*

Vous avez un aperçu de toutes les icônes disponibles là --> [Les FaceID](#)

Top

2.2.3 - Icône personnalisée

En plus des icônes prédéfinies ci-dessus, vous pouvez créer votre propre icône à l'aide d'une image personnalisée. La procédure est la suivante :

- insertion de l'image sur la feuille et copie de celle-ci dans le presse-papier
- collage de l'image sur le bouton
- suppression de l'image précédemment insérée sur la feuille

Exemple d'utilisation :

Code VBA à insérer dans un module standard

```

Sub test_image_perso()
Application.ScreenUpdating = False '<-- désactivation du rafraîchissement de l'écran


With ActiveSheet.Pictures.Insert("D:\Images\MonImage.jpg") '<-- insertion de l'image
.Name = "Pic1" '<-- attribution d'un nom à l'image
.Copy '<-- copie de l'image dans le presse-papier
End With

With CommandBars("MaBarre").Controls.Add(msoControlButton) '<-- création du bouton
.Style = msoButtonIconAndCaption '<-- bouton avec icône + texte
.Caption = "Bouton 1" '<-- texte du bouton
.PasteFace '<-- collage de l'image issue du presse-papier
End With

ActiveSheet.Pictures("Pic1").Delete '<-- suppression de l'image insérée sur la feuille

Application.ScreenUpdating = True '<-- réactivation du rafraîchissement de l'écran
End Sub

```

 *Vous ne devez pas vous soucier de la taille de l'image, celle-ci sera automatiquement redimensionnée à la taille du bouton.*

Top

2.2.4 - ID (argument)

L'argument **ID** permet, via un numéro d'identification, d'insérer un bouton prédéfini de l'application avec toutes ses propriétés.

Pour connaître le numéro d'identification d'un bouton en particulier, vous pouvez vous référer aux numéros des **FaceID** ci-dessus, les numéros **Id** et **FaceID** sont identiques.

Syntaxe : `mabarre.Controls.Add(Type:=msoControlButton, Id:=n°)`

Exemple d'utilisation : création d'une barre de menus et insertion des boutons permettant d'ouvrir les applications Word, Acces et PowerPoint.

Code VBA à insérer dans un module standard

```

Sub test_menuID()
Dim Cbar As CommandBar, Cbut As CommandBarButton

Set Cbar = CommandBars.Add(Name:="MaBarre", Position:=msoBarTop, Temporary:=True)
Cbar.Visible = True

Set Cbut = Cbar.Controls.Add(Type:=msoControlButton, ID:=42) '<-- bouton Word
Set Cbut = Cbar.Controls.Add(Type:=msoControlButton, ID:=264) '<-- bouton Access
Set Cbut = Cbar.Controls.Add(Type:=msoControlButton, ID:=267) '<-- bouton PowerPoint

End Sub

```

Aperçu :



i Comme vous pouvez le constater, lorsque l'on insère un bouton prédéfini via son numéro d'identification, il n'est pas nécessaire de préciser d'autres paramètres, tous les paramètres du bouton seront automatiquement ajoutés.

Top

2.2.5 - Caption

La propriété **Caption** permet d'ajouter un texte sur le bouton selon l'emplacement défini par la propriété **Style** (voir chapitre 2.2.1).

Si vous utilisez un style **msoButtonIcon** (bouton avec icône et sans texte), vous pouvez malgré tout utiliser la propriété **Caption**,

le texte apparaîtra sous forme d'une info-bulle lorsque la souris survolera ce bouton.

Syntaxe : `monbouton.Caption = "texte"`

Exemple : `Caption = "Bouton 1"`

Top

2.2.6 - OnAction

La propriété **OnAction** va définir la procédure à exécuter lors du clic sur le bouton.

Syntaxe : `monbouton.OnAction = "Nom de la macro"`

Exemple : `OnAction = "Macro1"`

[Top](#)

2.2.7 - TooltipText

La propriété **TooltipText** permet d'afficher une info-bulle lorsque la souris survole le bouton.

Si cette propriété n'est pas spécifiée, par défaut ça sera la valeur de la propriété **Caption** qui s'affichera dans l'info-bulle.

Syntaxe : `monbouton.TooltipText = "texte de l'info-bulle"`

[Top](#)

2.2.8 - HyperlinkType

La propriété **HyperlinkType** permet d'insérer un lien hypertexte au bouton.

Cette propriété accepte trois **constantes** comme valeur :

- **msoCommandBarButtonHyperlinkNone** = bouton sans lien hypertexte (*valeur par défaut*)
- **msoCommandBarButtonHyperlinkOpen** = bouton avec lien hypertexte
- **msoCommandBarButtonHyperlinkInsertPicture** = bouton permettant d'insérer une image sur la feuille

C'est via la propriété **TooltipText** que l'on va renseigner l'adresse URL de la page web à ouvrir ou de l'image à insérer. L'image peut également provenir de votre PC. TooltipText contiendra alors le chemin complet (`TooltipText = "D:\Images\MonImage.jpg"`).

Exemple d'utilisation : insertion, sur votre barre de menus, d'un bouton permettant d'ouvrir la page de votre site web favori

Code VBA à insérer dans un module standard

```
Sub bouton_DVP()
Dim cbut As CommandBarButton

Set cbut = CommandBars("MaBarre").Controls.Add(Type:=msoControlButton)

With cbut
.Style = msoButtonIcon
.FaceId = 487
.HyperlinkType = msoCommandBarButtonHyperlinkOpen
.TooltipText = "http://www.developpez.net/forums/"
End With

End Sub
```

[Top](#)

2.2.9 - BeginGroup

La propriété **BeginGroup** permet de placer un séparateur juste avant le bouton, ceci afin de créer des groupes ou de mieux visualiser chaque bouton.

Cette propriété est de type booléen, elle accepte donc deux états :

- **True** = avec séparateur
- **False** = sans séparateur (*état par défaut*)

Syntaxe : `monbouton.BeginGroup = True`

Aperçus :

BeginGroup = False



BeginGroup = True



Top

2.2.10 - Copy et CopyFace

La méthode **Copy** permet de copier un bouton d'une barre à l'autre. Toutes les propriétés liées à ce bouton seront également copiées.

Exemple d'utilisation : copie du bouton situé en 3ème position sur la barre de menus "MaBarre1", vers la barre de menus "MaBarre2", et positionnement de celui-ci en première position.

Code VBA à insérer dans un module standard

```

Sub test_copy()
Dim Cbut As CommandBarButton, Cbar As CommandBar

Set Cbar = CommandBars("MaBarre2") '<-- barre de destination
Set Cbut = CommandBars("MaBarre1").Controls(3) '<-- bouton à copier

Cbut.Copy bar:=Cbar,
  before:=1 '<-- si l'argument before n'est pas spécifié, le bouton se placera par défaut en dernière position
End Sub

```

La méthode **CopyFace** permet de copier l'image d'un bouton sur un autre bouton, en combinaison avec la méthode **PasteFace**.

Exemple d'utilisation : recherche du bouton permettant d'imprimer (bouton prédéfini de l'application), dont l'index d'identification est le 4, et copie de son icône sur le deuxième bouton de notre barre personnalisée (*Voir le chapitre 2.2.4 pour plus de détail concernant les index d'identification des boutons prédéfinis*).

Code VBA à insérer dans un module standard


```

Sub test_copyface()

CommandBars.FindControl(Type:=msoControlButton, ID:=4).CopyFace
CommandBars("MaBarre").Controls(2).PasteFace

End Sub

```

 **La copie de l'icône ne fonctionnera pas si le bouton de destination est du style *msoButtonCaption***
 Voir le chapitre 2.2.1 pour plus de détails sur les styles.

Top

2.2.11 - Move

La méthode **Move** s'utilise de la même manière que la méthode **Copy** à la différence que celle-ci va déplacer le bouton d'une barre à l'autre au lieu de le copier.

[Top](#)

2.2.12 - Builtin et BuiltinFace

La propriété **Builtin**, de type booléen, va renvoyer la valeur True s'il s'agit d'un bouton prédéfini de l'application. Voir le chapitre 2.2.4 pour plus de détails sur les boutons prédéfinis.

Exemple d'utilisation : suppression de tous les boutons prédéfinis de notre barre de menus personnalisée

Code VBA à insérer dans un module standard

```
Sub test_del_bouton()
Dim Cbar As CommandBar

Set Cbar = CommandBars("MaBarre")

For i = Cbar.Controls.Count To 1 Step -1
If Cbar.Controls(i).Builtin = True Then Cbar.Controls(i).Delete
Next
End Sub
```

La propriété **BuiltinFace**, de type booléen, va renvoyer la valeur True si l'icône du bouton est une des icônes prédéfinies de l'application.

Voir le chapitre 2.2.2 pour plus de détail sur les icônes prédéfinies.

Son utilisation est identique à la propriété Builtin ci-dessus.

[Top](#)

2.2.13 - Enabled

La propriété **Enabled** permet de "griser" un bouton et de le rendre inactif.

De type booléen, cette propriété accepte 2 états :

- **True** = bouton actif (état par défaut)
- **False** = bouton inactif

Syntaxe : *monbouton.Enabled = False*

[Top](#)

2.2.14 - Visible

La propriété **Visible** s'utilise de la même manière que la propriété **Enabled** à la différence que le bouton devient totalement invisible pour l'utilisateur.

[Top](#)

2.2.15 - Before (argument)

L'argument **Before** permet, lors de l'ajout d'un bouton à votre barre de menus, de le positionner par rapport aux boutons déjà en place.

Syntaxe : *mabarre.Controls.Add(Type:=msoControlButton, Before:=3)* (le bouton sera en 3ème position)

[Top](#)

2.2.16 - Priority

Lorsque l'on insère de nombreux boutons sur la barre de menus, ceux-ci peuvent ne pas être tous visibles par manque de place.

Auquel cas une petite flèche (sur le côté droit de la barre de menus) permet d'ajouter un bouton ce qui aura pour effet de permuter un des boutons visibles par celui choisi.

La propriété **Priority** permet d'affecter aux boutons un degré d'importance.

Le bouton ayant le degré d'importance le plus faible sera permuté en premier. Les valeurs pouvant être attribuées vont de 1 à 7, 7 étant le degré d'importance le plus faible.

Syntaxe : `monbouton.Priority = 1`

[Top](#)

2.2.17 - Tag

Un bouton ne possédant pas de propriété **Nom**, la propriété **Tag** joue un peu le même rôle en attribuant une étiquette au bouton.

Syntaxe : `monbouton.Tag = "étiquette"`

[Top](#)

2.2.18 - DescriptionText

La propriété **DescriptionText**, de type String, permet de lier un texte au bouton de commande.

Ce texte n'est pas visible par l'utilisateur mais peut être récupéré dans une procédure via par exemple une MsgBox.

Syntaxe : `monbouton.DescriptionText = "Message d'explication concernant" & chr(10) & "l'utilisation de ce bouton"`

[Top](#)

2.2.19 - State

La propriété **State** permet de changer l'aspect "levé" ou "enfoncé" du bouton.

Trois **constantes** sont possibles :

- **msuButtonUp** = aspect levé (*valeur par défaut*)
- **msoButtonDown** = aspect enfoncé
- **msoButtonMixed** = aspect enfoncé (*je n'ai pas constaté de différence avec la constante msoButtonDown*)

Syntaxe : `monbouton.State = constante`

[Top](#)

2.3 - La zone de texte et ses propriétés

Pour rappel, la constante définissant le contrôle type "zone de texte" est : **msoControlEdit**

Syntaxe : `CommandBars("MaBarre").Controls.Add(msoControlEdit)`

La zone de texte permet d'introduire un mot ou une phrase et de récupérer cette chaîne de caractères dans la procédure.

Dès lors que la zone de texte possède le focus, la procédure, liée à la propriété **OnAction**, sera exécutée lors de l'appui sur la touche **Tab** ou **Enter**.

Les propriétés suivantes s'utilisent de la même manière que ci-dessus (*voir les propriétés du bouton de commande*) : **Caption - OnAction - TooltipText - BeginGroup - Copy - Move - Enabled - Visible - Before - Priority - Tag - DescriptionText**

2.3.1 - Style

Deux **Styles** d'affichage sont possibles, les **constantes** sont :

- **msoComboNormal** = affichage uniquement de la zone de texte (*valeur par défaut*)
- **msoComboLabel** = affichage de la zone de texte et d'un **Label** à la gauche de celle-ci. Ce style s'utilise avec la propriété **Caption** qui contiendra le texte s'affichant dans le **Label**.

Exemple d'utilisation : insertion sur notre barre de menus de deux zones de texte, une simple et la seconde avec Label

Code VBA à insérer dans un module standard

```

Sub test_zone_txt()

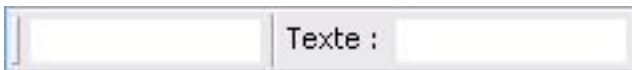
With CommandBars("MaBarre").Controls.Add(Type:=msoControlEdit)
.TooltipText = "info-bulle zone txt 1"
.Tag = "txt1"
.OnAction = "MaMacro1"
End With

With CommandBars("MaBarre").Controls.Add(Type:=msoControlEdit)
.Style = msoComboLabel
.Caption = "Texte :"
.TooltipText = "info-bulle zone txt 2"
.Tag = "txt2"
.OnAction = "MaMacro2"
.BeginGroup = True
End With

End Sub

```

Aperçu :



Top

2.3.2 - Text

La propriété **Text** s'utilise de deux façons différentes.

Soit lors de la création de la zone de texte, ce qui permet d'afficher un texte par défaut dans la zone.

Syntaxe : `mazonetxt.Text = "Texte à afficher"`

Soit dans la procédure exécutée via la propriété **OnAction**, ce qui permet de récupérer le texte introduit dans la zone.
 Syntaxe : `MsgBox CommandBars("MaBarre").Controls(1).Text` (*Controls(1) étant la position de la zone de texte sur la barre de menus*)

Si vous ne connaissez pas la position de la zone de texte sur la barre de menus ou si cette position peut changer au fil du temps,

vous pouvez utiliser la fonction **FindControl** de la barre de menus en cherchant par exemple à identifier le **Tag** de la zone de texte (*Tag inséré lors de la création de la zone de texte*).

Syntaxe : `MsgBox CommandBars("MaBarre").FindControl(Tag:="txt1").Text`

Une troisième possibilité, la plus simple, consiste à utiliser la propriété **ActionControl** que nous avons vue ci-dessus au chapitre 1.3.5 Syntaxe : `MsgBox CommandBars.ActionControl.Text`

Top

2.3.3 - Clear

Après avoir introduit du texte dans la zone, celui-ci reste affiché.

La propriété **Clear** permet d'effacer le contenu de la zone de texte après qu'il ait été récupéré dans la procédure.

Syntaxe : **CommandBars("Mabarre").FindControl(Tag="txt1").Clear**

[Top](#)

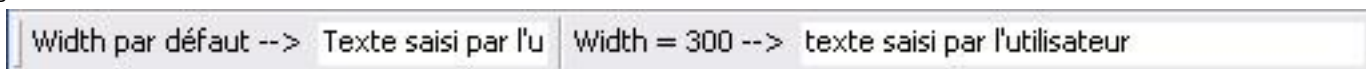
2.3.4 - Width

Si l'utilisateur doit saisir une phrase relativement longue, la propriété **Width** permet d'allonger la zone afin que le texte saisi apparaisse en entier.

Par défaut, la longueur de la zone de texte est de **90 pixels**.

Syntaxe : *mazonetxt.Width = valeur en pixel*

Aperçu :



[Top](#)

2.4 - La zone de liste et ses propriétés

La zone de liste permet d'afficher plusieurs choix possibles sous forme d'une liste déroulante.

Deux types de liste sont possibles, les **constantes** sont :

- **msoControlDropdown** = zone de liste classique
- **msoControlComboBox** = zone de liste dite "modifiable"

Une seule différence entre ces deux types de liste : la zone de liste dite "modifiable" permet à l'utilisateur d'ajouter manuellement une ligne de texte à la liste pré-établie.

Syntaxe : **CommandBars("MaBarre").Controls.Add(msoControlDropdown)** ou *msoControlComboBox*

Les propriétés suivantes s'utilisent de la même manière que ci-dessus (*voir les propriétés du bouton de commande*) :

Caption - OnAction - TooltipText - BeginGroup - Copy - Move - Enabled - Visible - Before - Priority - Tag - DescriptionText

2.4.1 - Style

La propriété **Style** de la zone de liste est identique à la propriété Style de la zone de texte ci-dessus, affichage de la zone de liste avec ou sans Label.

[Top](#)

2.4.2 - AddItem

C'est via la propriété **AddItem** que l'on ajoute les éléments qui vont composer la liste.

Syntaxe : *maliste.AddItem(Text)*

Exemple d'utilisation : ajout sur notre barre de menus d'une zone de liste comportant 10 choix possibles

Code VBA à insérer dans un module standard

```
Sub test_zone_liste()  
Dim x As Byte
```

Code VBA à insérer dans un module standard

```

With CommandBars("MaBarre").Controls.Add(Type:=msoControlDropdown)
    .Style = msoComboLabel
    .Caption = "Liste de choix :"
    .TooltipText = "info-bulle"
    .OnAction = "MaMacrol"

    For x = 1 To 10
        .AddItem ("Choix " & x)
    Next x

End With

End Sub
    
```

Aperçus :



Top

2.4.3 - Text

La propriété **Text** va permettre de récupérer dans la procédure exécutée via la propriété **OnAction**, le texte sélectionné dans la zone de liste.

La procédure sera exécutée à chaque fois que l'utilisateur fera un choix dans la liste.

Syntaxe : **CommandBars("Mabarre").Controls(1).Text** ou **CommandBars.ActionControl.Text**

Top

2.4.4 - ListIndex

La propriété **ListIndex** renvoie le numéro de la ligne sélectionnée par l'utilisateur dans la zone de liste.

Syntaxe : **CommandBars("MaBarre").Controls(1).ListIndex**

Exemple d'utilisation : action différente exécutée dans la procédure selon la sélection faite par l'utilisateur dans la zone de liste comportant 5 choix possibles

Code VBA à insérer dans un module standard

```
Sub MaMacrol()
Dim Choix As Byte

Choix =
CommandBars("MaBarre").Controls(1).ListIndex '<-- Controls(index) : index = position de la liste sur la barre de menus

Select Case Choix
Case 1: MsgBox "action si choix 1"
Case 2: MsgBox "action si choix 2"
Case 3: MsgBox "action si choix 3"
Case 4: MsgBox "action si choix 4"
Case 5: MsgBox "action si choix 5"
End Select

End Sub
```

[Top](#)

2.4.5 - ListCount

La propriété **ListCount** renvoie le nombre de lignes que contient la zone de liste.
 Syntaxe : **CommandBars("MaBarre").Controls(1).ListCount**

[Top](#)

2.4.6 - List(index)

La propriété **List(index)** renvoie le contenu de la ligne mentionnée par l'index.
 Syntaxe : **CommandBars("MaBarre").Controls(1).List(index)** (*index étant le numéro de ligne, de 1 à nombre de lignes*)
 Exemple d'utilisation : boucle permettant de parcourir toutes les lignes de notre zone de liste

Code VBA à insérer dans un module standard

```
Sub test_list_index()
Dim x As Byte, Result As String

With
CommandBars("MaBarre").Controls(1) '<-- Controls(index) : index = position de la liste sur la barre de menus
For x = 1 To .ListCount
Result = Result & Chr(10) & .List(x)
Next x
End With

MsgBox "La liste contient les choix suivants :" & Chr(10) & Result

End Sub
```

[Top](#)

2.4.7 - RemoveItem(index)

La propriété **RemoveItem(index)** permet de supprimer la ligne mentionnée par l'index.
 Syntaxe : **CommandBars("MaBarre").Controls(1).RemoveItem(index)** (*index étant le numéro de ligne, de 1 à nombre de lignes*)

Top

2.4.8 - Clear

La propriété **Clear** permet d'effacer la totalité du contenu de la zone de liste.

Syntaxe : **CommandBars("MaBarre").Controls(1).Clear**

Top

2.4.9 - ListHeaderCount

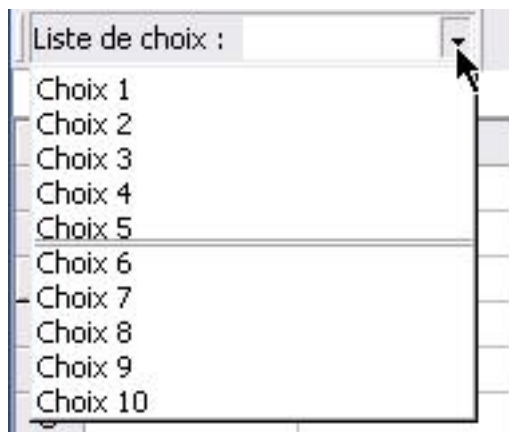
La propriété **ListHeaderCount** permet de tracer une ligne continue entre deux lignes de texte.

Syntaxe : **CommandBars("MaBarre").Controls(1).ListHeaderCount = index**

l'index étant le numéro de la ligne sous laquelle la ligne sera tracée

Aperçu :

ListHeaderCount = 5



Top

2.4.10 - DropDownLines

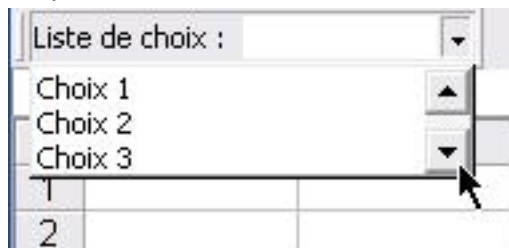
La propriété **DropDownLines** va définir le nombre de lignes affichées simultanément.

Si la zone de liste contient un nombre de lignes supérieur au nombre défini pour l'affichage, une barre de défilement (scrollbar) sera automatiquement ajoutée.

Syntaxe : *maliste*.**DropDownLines = nombre**

Aperçu :

DropDownLines = 3



Top

2.4.11 - DropDownWidth

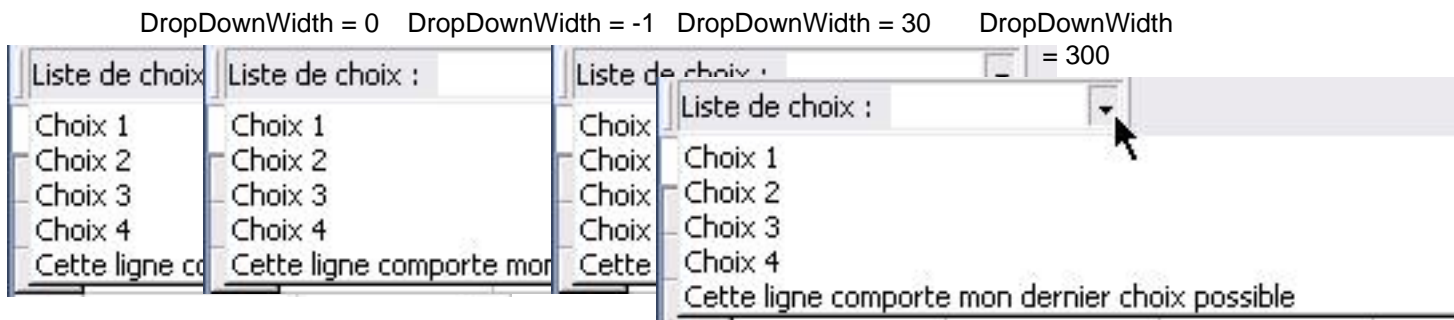
La propriété **DropDownWidth** permet d'ajuster la largeur de la zone de liste.

Syntaxe : *maliste*.**DropDownWidth** = valeur

Trois possibilités de définir la valeur :

- = 0 --> la largeur de la zone de liste est fixe et égale à la largeur du contrôle (*valeur par défaut*)
- = -1 --> la largeur de la zone de liste s'adapte en fonction de la plus longue ligne de texte
- = valeur en pixels --> la largeur de la zone de liste est fixe selon la valeur définie

Aperçus :



i Vous pouvez constater dans les aperçus ci-dessus que seule la largeur de la zone de liste change, la largeur du contrôle sur la barre de menus reste fixe.

[Top](#)

2.5 - Les sous-menus

Les sous-menus vont se révéler utiles lorsque vous créez une barre de menus comportant de nombreux contrôles (boutons, zones de texte et zones de liste).

Un peu à la manière de l'explorateur Windows, vous allez pouvoir réaliser une arborescence de plusieurs sous-menus imbriqués et y insérer les contrôles.

Pour rappel, la constante définissant le contrôle type "sous-menu" est : **msoControlPopup**

Syntaxe : **CommandBars("MaBarre").Controls.Add(msoControlPopup)**

Les propriétés suivantes s'utilisent de la même manière que ci-dessus (*voir les propriétés du bouton de commande*) :

Caption - TooltipText - BeginGroup - Copy - Move - Enabled - Visible - Before - Tag - DescriptionText

Pour mieux comprendre l'utilisation des sous-menus, examinez l'exemple de code ci-dessous et observez les captures d'écran.

[Top](#)

2.6 - Exemple de code

Afin de résumer un peu toutes les procédures que nous venons de parcourir, nous allons mettre tout ça en musique via un exemple de création.

Code VBA à insérer dans un module standard

```

Sub barre_menus_perso()
'déclaration des variables
Dim Cbar As CommandBar, Cbut As CommandBarButton
    
```

Code VBA à insérer dans un module standard

```

Dim Ctxt As CommandBarComboBox, Cpop1 As CommandBarPopup, Cpop2 As CommandBarPopup
Dim x As Byte

'création de la barre de menus
Set Cbar = CommandBars.Add(Name:="MaBarre", Position:=msoBarTop, Temporary:=True)
Cbar.Protection = msoBarNoMove + msoBarNoCustomize '<-- protection de la barre de menus

'insertion sur la barre de menus d'un bouton de commande
Set Cbut = Cbar.Controls.Add(msoControlButton)
With Cbut
    .FaceId = 358 '<-- icône
    .OnAction = "Macro1" '<-- procédure à exécuter
    .TooltipText = "Suppression barre de menus" '<-- info-bulle
    .Tag = "cbut1" '<-- étiquette
End With

'insertion sur la barre de menus d'une zone de texte
Set Ctxt = Cbar.Controls.Add(msoControlEdit)
With Ctxt
    .Style = msoComboLabel '<-- zone de texte avec label
    .Caption = "Date :" '<-- texte du label
    .TooltipText = "Veuillez introduire une date" '<-- info-bulle
    .OnAction = "Macro2" '<-- procédure à exécuter
    .BeginGroup = True '<-- barre de séparation
    .Tag = "ctxt1" '<-- étiquette
End With

'insertion sur la barre de menus d'une zone de liste
Set Ctxt = Cbar.Controls.Add(msoControlDropdown)
With Ctxt
    .Style = msoComboLabel '<-- zone de liste avec label
    .Caption = "Liste :" '<-- texte du label
    .TooltipText = "Faites votre choix" '<-- info-bulle
    .OnAction = "Macro3" '<-- procédure à exécuter
    .BeginGroup = True '<-- barre de séparation
    .Tag = "clist1" '<-- étiquette
    For x = 1 To 5
        .AddItem ("Choix " & x) '<-- ajout de 5 Item
    Next
End With

'insertion sur la barre de menus d'un sous-menu 1
Set Cpop1 = Cbar.Controls.Add(msoControlPopup)
With Cpop1
    .Caption = "Sous-menu 1" '<-- label du sous-menu
    .Tag = "sml" '<-- étiquette
End With

'insertion dans le sous-menu 1 d'un premier bouton de commande
Set Cbut = Cpop1.Controls.Add(msoControlButton)
With Cbut
    .Style = msoButtonCaption '<-- bouton avec texte uniquement
    .Caption = "Bouton 1" '<-- label du bouton
    .OnAction = "Macro4" '<-- procédure à exécuter
    .Tag = "smlcbut1" '<-- étiquette
End With

'insertion dans le sous-menu 1 d'un second bouton de commande
Set Cbut = Cpop1.Controls.Add(msoControlButton)
With Cbut
    .Style = msoButtonCaption '<-- bouton avec texte uniquement
    .Caption = "Bouton 2" '<-- label du bouton
    .OnAction = "Macro4" '<-- procédure à exécuter
    .Tag = "smlcbut2" '<-- étiquette
End With

'insertion dans le sous-menu 1 d'un sous-menu 2
Set Cpop2 = Cpop1.Controls.Add(msoControlPopup)
With Cpop2
    .Caption = "Sous-menu 2" '<-- label du sous-menu
    .Tag = "sm2" '<-- étiquette

```

Code VBA à insérer dans un module standard

```

End With

'insertion dans le sous-menu 2 d'un bouton prédéfini ouvrant l'application Word
Set Cbut = Cpop2.Controls.Add(Type:=msoControlButton, ID:=42)
With Cbut
    .Style = msoButtonIconAndCaption '<-- bouton avec icône + texte
    .Caption = "Word" '<-- label du bouton
End With

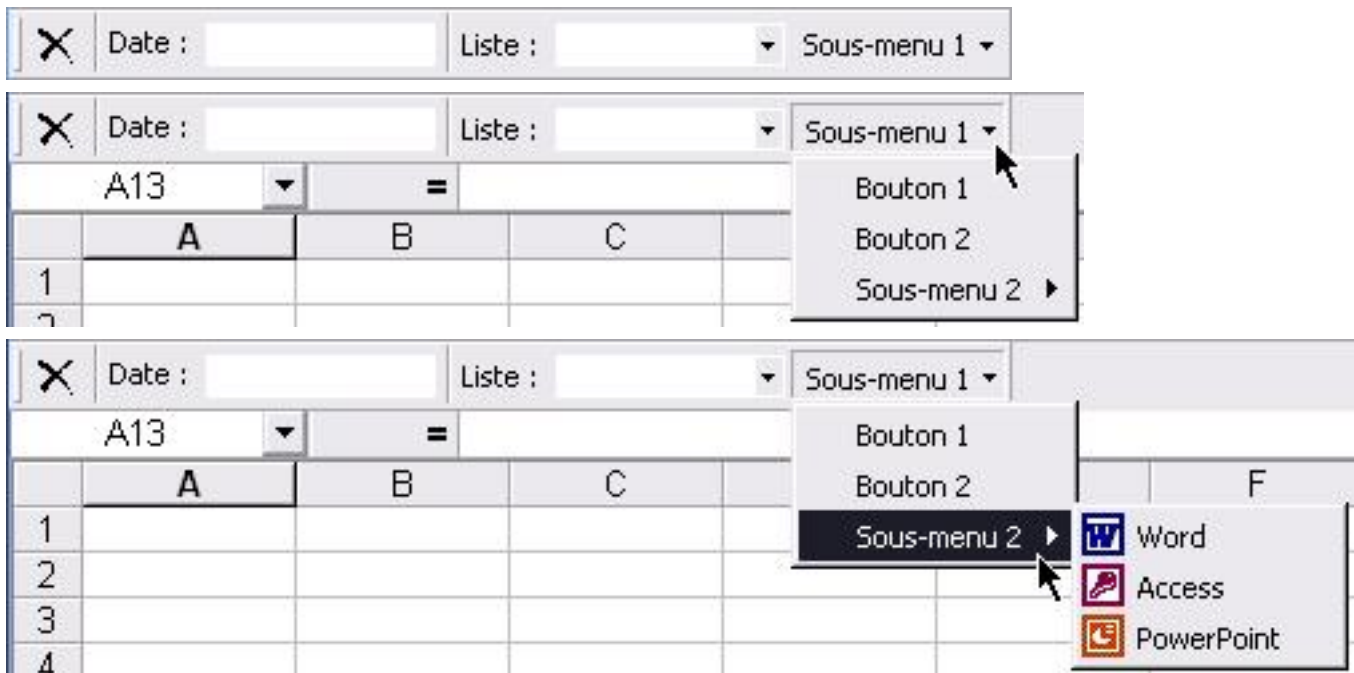
'insertion dans le sous-menu 2 d'un bouton prédéfini ouvrant l'application Access
Set Cbut = Cpop2.Controls.Add(Type:=msoControlButton, ID:=264)
With Cbut
    .Style = msoButtonIconAndCaption '<-- bouton avec icône + texte
    .Caption = "Access" '<-- label du bouton
End With

'insertion dans le sous-menu 2 d'un bouton prédéfini ouvrant l'application PowerPoint
Set Cbut = Cpop2.Controls.Add(Type:=msoControlButton, ID:=267)
With Cbut
    .Style = msoButtonIconAndCaption '<-- bouton avec icône + texte
    .Caption = "PowerPoint" '<-- label du bouton
End With

Cbar.Visible = True '<-- affichage de la barre de menus

End Sub
    
```

Aperçus de la barre de menus perso :



i Vous trouverez, dans le chapitre Téléchargements, un classeur démo comportant la procédure de création de cette barre de menus ainsi que quelques petites procédures exécutées depuis les différents contrôles. La barre de menus est automatiquement créée à l'ouverture et automatiquement supprimée à la fermeture du classeur démo.

Top

3 - Menu Popup

La barre de menus de type Popup est en quelque sorte une barre de menus classique sauf que sa position est particulière.

Celle-ci va s'afficher, tel un popup, sous le curseur de la souris.

Tous les paramètres, propriétés et contrôles sont également valables hormis les deux points suivants qui sont particuliers à la barre de menu de type Popup :

- position = **msoBarPopup**
- affichage = **CommandBars("MaBarrePopup").ShowPopup**

Maintenant que nous avons détaillé, dans les différents chapitres ci-dessus, comment créer une barre de menus et y insérer des contrôles, nous allons directement aller à l'essentiel par un exemple de création et d'utilisation via les codes ci-dessous (*exemple dans le fichier démo*).

- Code de création de la barre de menus avec insertion d'une zone de liste contenant 20 valeurs

Code VBA à insérer dans un module standard

```
Sub Creation_barre_popup()
'déclaration des variables
Dim Cbar As CommandBar, Clist As CommandBarComboBox, x As Byte

'création de la barre de menu de type Popup
Set Cbar = CommandBars.Add(Name:="MaBarrePopup", Position:=msoBarPopup, temporary:=True)

'insertion d'une zone de liste
Set Clist = Cbar.Controls.Add(msoControlDropDown)
With Clist
.TooltipText = "Saisissez une valeur" '<-- info-bulle
.DropDownLines = 10 '<-- limitation de l'affichage à 10 lignes
.OnAction = "Macrol" '<-- procédure à exécuter
For x = 1 To 20
.AddItem ("Valeur " & x) '<-- ajout de 20 Items
Next x
End With
End Sub
```

- Code de suppression de la barre de menus

Code VBA à insérer dans un module standard

```
Sub Supp_barre_popup()
On Error Resume Next
CommandBars("MaBarrePopup").Delete
End Sub
```

- Code de création et de suppression automatique lors de l'ouverture et de la fermeture du classeur

Code VBA à placer dans l'objet ThisWorkbook

```
Private Sub Workbook_BeforeClose(Cancel As Boolean)
Supp_barre_popup '<-- suppression de la barre de menu à l'ouverture du classeur
End Sub

Private Sub Workbook_Open()
Creation_barre_popup '<-- création de la barre de menu à l'ouverture du classeur
'Remarque : la barre est créée mais non visible
End Sub
```

- Code permettant d'afficher la barre de menu

Code VBA à insérer dans un module standard

```
Sub Affiche_barre_popup()
CommandBars( "MaBarrePopup" ).ShowPopup
End Sub
```

- Code d'exécution de la procédure d'affichage de la barre de menus via le clic droit de la souris

Code VBA à placer dans l'objet ThisWorkbook

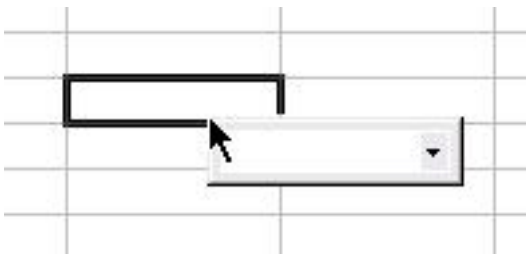
```
Private Sub Workbook_SheetBeforeRightClick(ByVal Sh As Object, ByVal Target As Range, Cancel As Boolean)
Affiche_barre_popup
'désactivation du menu contextuel de l'application
'qui s'affiche en temps normal lors du clic droit de la souris
Cancel = True
End Sub
```

- Procédure qui sera exécutée lors du choix d'une des valeurs dans la zone de liste
 Dans cet exemple, la cellule dans laquelle a été effectué le clic droit de la souris recevra la valeur sélectionnée dans la zone de liste

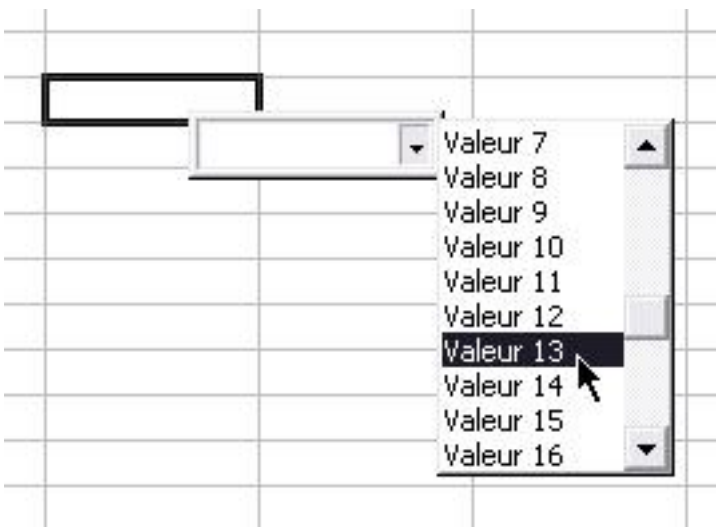
Code VBA à insérer dans un module standard

```
Sub Macrol()
ActiveCell.Value = CommandBars.ActionControl.Text
End Sub
```

Aperçus des résultats :
 Clic droit dans une cellule



Sélection de la valeur dans la zone de liste



Résultat



Top

4 - Les barres de menus prédéfinies

Cet article est axé sur la création d'une barre de menus personnalisée mais vous pouvez également adapter les barres de menus prédéfinies.

Pour de plus amples explications concernant la modification d'une barre de menu prédéfinie, je vous invite à consulter le tutoriel de **Philippe JOCHMANS** (Starec).

Personnaliser vos barres de commandes dans Access

5 - Remerciements

Remerciements :

A toute l'**équipe Office** pour leur dévouement qui fait en sorte que ce **forum** est une véritable mine d'or pour ceux qui cherchent une info, de la plus banale à la plus tordue.

A **Pierre Fauconnier** (mon parrain) pour m'avoir embarqué et guidé dans cette joyeuse galère.

Aux ténors **SilkyRoad**, **Starec**, **Ouskel'n'or**, **Antoun** et **Tirex28** pour leurs ajustements techniques et orthographiques.

VI - Téléchargements

Création d'une barre de menu : **Fichier Demo**