

Rubans Microsoft personnalisés

Ceci est l'icône de l'Application qui apparaît dans la barre des tâches quand l'Application est ouverte



Dans l'Application la Création d'un Ruban Personnalisé se fait en **QUATRE ÉTAPES**

- 1) **Première Étape :**
Saisie de tous les éléments du Ruban Personnalisé dans les formulaires et sous-formulaires de l'Application
- 2) **Deuxième Étape :**
[Saisie du Code dans les Modules](#) pour le fonctionnement des contrôles
- 3) **Troisième Étape :**
La Création du Code XML stocké dans la table USysRibbons avec la Commande « Créer / Éditer le Ruban »
- 4) **Quatrième Étape :**
L'Activation du Ruban Personnalisé par la Commande « **Compacter et Réparer la Base de données** » ou par la **Fermeture et la Réouverture de l'application**

Paramètres Access importants pour le bon fonctionnement de l'Application

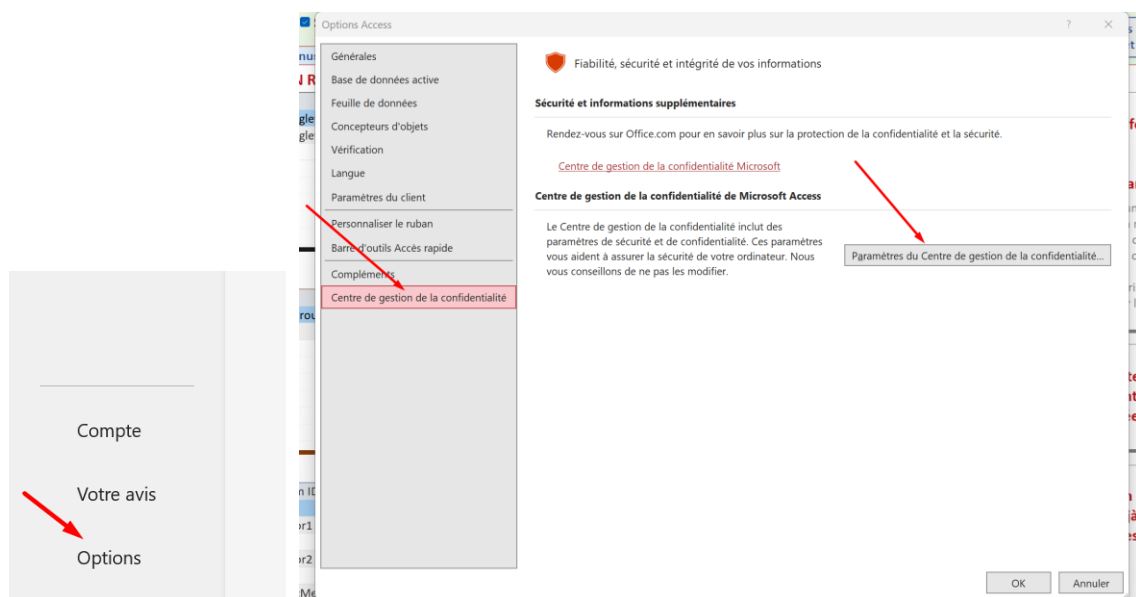
Activation des contrôles ActiveX et des Macros

Menu – FICHIER

Cliquer sur **Options**

Cliquer sur **Centre de gestion de la confidentialité**

Cliquer sur **Paramètres du Centre de gestion de la confidentialité**

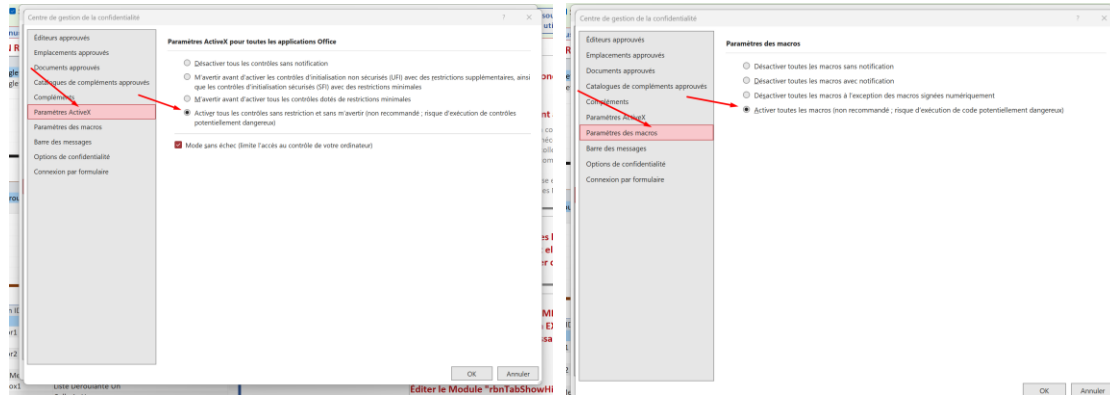


Cliquer sur **Paramètres ActiveX**

Sélectionner la case **Activer tous les contrôles sans restrictions...**

Cliquer sur **Paramètres de macros**

Sélectionner la case **Activer toutes les macros...**



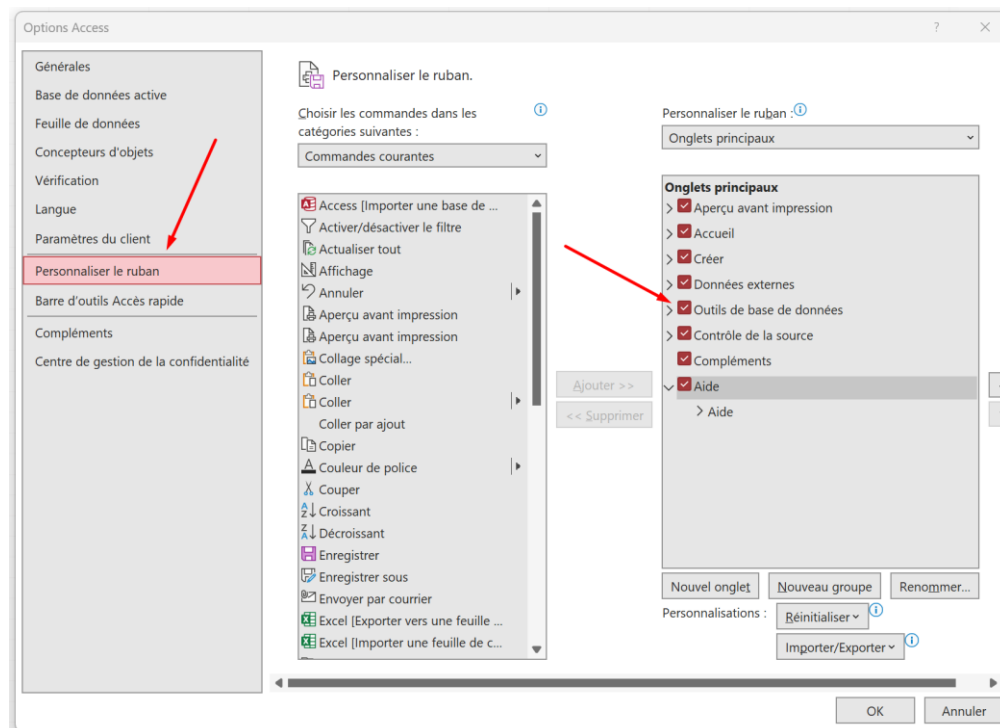
Paramètres importants à vérifier dans l'éditeur Visual Basic pour le bon fonctionnement de l'Application

**Accès à l'éditeur Visual Basic et Insertion de toutes les références nécessaires
(Le raccourci **Alt + F11** peut aussi ouvrir l'éditeur)**

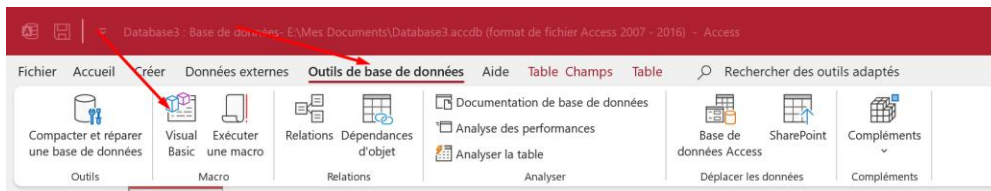
Menu Fichier – Options

Cliquer sur **Personnaliser le ruban**

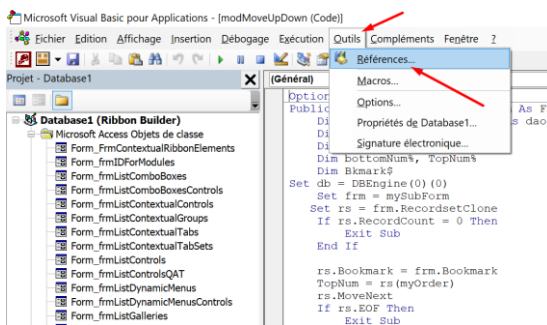
Cocher la case « **Outils de base de données** »



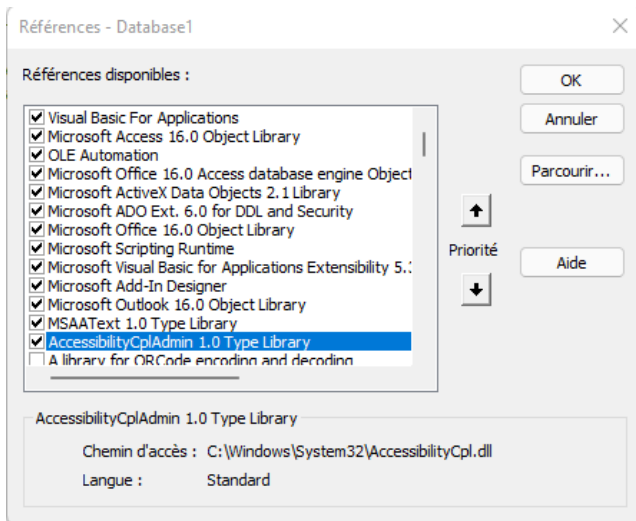
L'Onglet **Outils de base de données** sera affiché dans le ruban
Cliquer sur **Visual Basic** pour avoir accès à l'éditeur



Dans l'éditeur Visual Basic cliquer sur **Outils**
Puis cliquer sur **Références...**



S'assurer que **toutes les références** ci-dessous soient bien cochées
Essentiel pour le bon fonctionnement de l'Application



Informations générales importantes

Après avoir éditer le code dans l'éditeur Visual Basic

Si la fenêtre « **Microsoft Visual Basic pour Applications** » a été ouverte (fenêtre permettant d'entrer ou d'éditer du code VBA), il est nécessaire de faire un « **Compacter et Réparer Une Base de Données** » (File Compact and Repair Database).

Sinon le Ruban pourrait ne pas fonctionner comme prévu.

Menu Fichier – Informations – Compacter et réparer la base de données

CHANGER le NOM d'un élément déjà sauvegardé et contenant des Relations :

Changer le NOM de l'élément dans le formulaire de saisie

Après CHAQUE changement Cliquer sur « **Créer / Éditer le Ruban** » pour **actualiser** le changement

On peut changer d'autres NOMS si désiré

Pour terminer : « **Compacter et Réparer une Base de Données** »

N.B. IMPORTANT

Après des changements de nom d'Identifiants ou de nom d'Item dans les formulaires on doit faire les mêmes changements dans les Modules des Contrôles qui contiennent ces noms d'Identifiants ou d'Item.

Sinon certains contrôles ne fonctionneront plus.

CHANGER DE RUBAN

Utiliser les **LISTES DÉROULANTES** de Recherche contenant les noms des Rubans pour changer d'un Ruban à l'Autre.

Le Ruban Sélectionné devient actif dans le formulaire « **FrmRibbonElements** » et « **frmUSysRibbons** » qu'on se serve de la liste déroulante de n'importe lequel des deux formulaires. Cf : IMPORTANT

IMPORTANT

Pour que le changement du Ruban soit effectif, il est nécessaire de « **Compacter et Réparer une Base de Donnée** » (File Compact and Repair Database); cette action actualise le Ruban Sélectionné et le rend accessible pour la base de données.

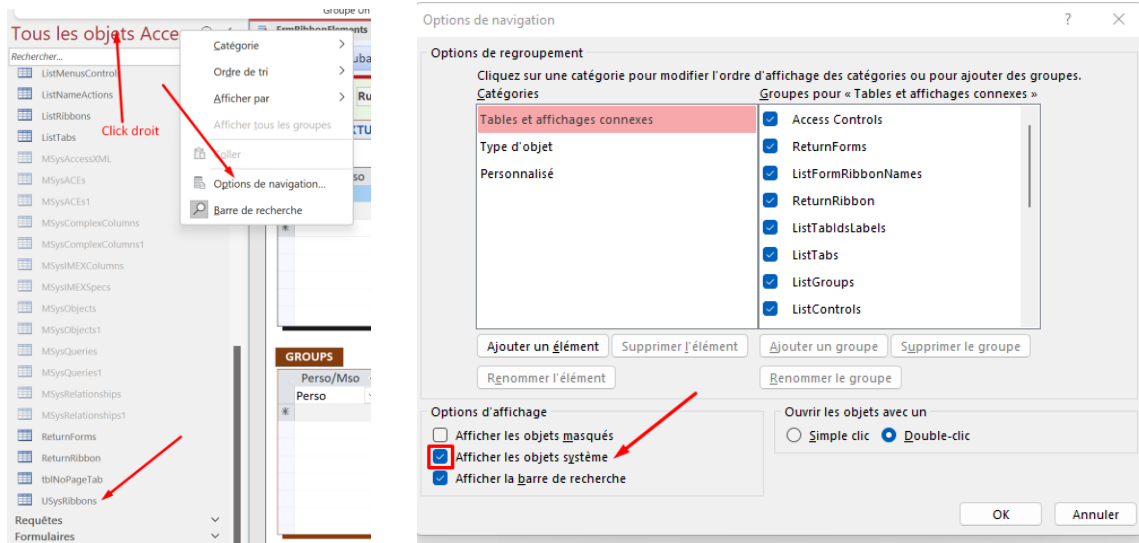
FICHIERS XML POUR LES RUBANS

Les fichiers XML créés sont stockés dans la table **USysRibbons**

Pour voir cette table on doit activer « **Afficher les objets systèmes** »

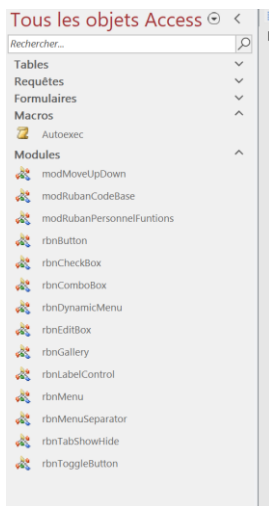
Click droit sur Tous les objets Access – Options de navigation

Cocher la case Afficher les objets système



SAISIE DU CODE DANS LES MODULES POUR LE FONCTIONNEMENT DES CONTRÔLES

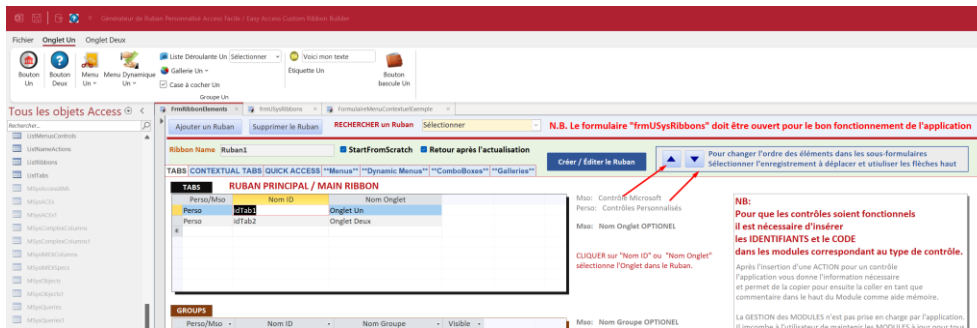
Saisir les codes dans les différents modules



Modules de Base du Ruban

Les trois premiers Modules commençant par « **mod** » sont pour le fonctionnement de l'Application et du Ruban ainsi que pour les Fonctions personnalisées.

Module **modMoveUpDown** est utilisé dans le formulaire FrmRibbonElements
Sert à changer l'ordre des différents contrôles



Module **modRubanCodeBase**

- Pour charger le ruban à l'ouverture
- Pour activer les différents rubans
- Pour activer les différents Tabs
- Pour charger les icônes
- AutoExecStart()

Module **modRubanPersonnelFunctions**

Fonctions nécessaires pour le bon fonctionnement de l'Application

Modules pour les contrôles du Ruban

Tous les autres Modules commençant par « **rbn** » sont pour le Code des Contrôles du Ruban

À la création d'un Contrôle nécessitant du code l'application vous donne la possibilité de copier l'information nécessaire et de la coller au haut du Module comme aide-mémoire.

L'application ouvre le module correspondant au Contrôle pour l'édition des Identifiants et du Code.

Les fonctions de base utilisées dans ces Modules sont appelés des CALLBACKS.

Les Callbacks sont spécifique au type de contrôle utilisé; par exemple pour un Button on aura "**ButtonCallbackOnAction**"

Ces CALLBACKS sont appelés par l'élément « onAction » dans le code XML des Rubans.

Exemple : onAction="**ButtonCallbackOnAction**"

Il est nécessaire d'insérer les Identifiants de chaque contrôle et le code à exécuter dans ces Callbacks.

Dans l'exemple suivant qui se trouve dans « rbnButton »

Le Code de base de la procédure est en **ORANGE**

Les Select Case et Case des Rubans sont en **VERT**

Les Select Case et Case des Contrôles sont en **BLEU**

Tous les Identifiants de contrôles nécessaires sont en **ROUGE**

Les Code à exécuter sont en **MAUVE**

Le texte en **NOIR** est une suggestion de commentaires pour bien classer les Rubans et leurs Contrôles; il n'a aucun effet sur la procédure ou la fonction.

```
Public Sub ButtonCallbackOnAction(control As IRibbonControl)
myRibbonName = Forms!FrmRibbonElements!NameRb.Value
'NORMAL TABS //////////////////////////////////////
  Select Case myRibbonName
    'RIBBON - Ruban1
    Case "Ruban1"
      'TAB - Onglet 1
      'GROUP - Groupe 1
      'CONTROLS
      'Insérer l'IDENTIFIANT du Button
      Select Case control.ID
        Case "idButton1"
          MsgBox "Bouton 1"
        Case "idButton2"
          MsgBox "Bouton 2"
      'GROUP - Name
      'Insérer l'IDENTIFIANT du Button
      'CONTROLS
      Case "IdentifiantButton"
        MsgBox "Text"

    'TAB - Onglet 2
    'GROUP - Groupe 2
    'Insérer l'IDENTIFIANT du Button
    'CONTROLS
    Case "IdentifiantButton"
      MsgBox "Text"
  End Select

  'RIBBON - Ruban2
  Case "Ruban2"
    'TAB - Onglet 1
    'GROUP - Groupe 1
    'Insérer l'IDENTIFIANT du Button
    'CONTROLS
```

```

        Select Case control.ID
        Case "IdentifiantButton"
            MsgBox "Text"
        Case "IDButton"
            MsgBox "Text"
        End Select
    End Select
'CONTEXTUAL TABS //////////////////////////////////////
    Select Case myRibbonName
    'RIBBON - Ruban1
    Case "Ruban1"
        'TAB - Onglet Contextuel
        'GROUP - Groupe Contextuel
        'CONTROLS
        Select Case control.ID
        'Insérer l'IDENTIFIANT du Button
        Case "idButtonCtx1"
            MsgBox "Bouton Contextuel 1"
        Case "Nom"
            MsgBox "Bouton Contextuel 2"
        End Select
    'RIBBON - Autre
    Case "IdentifiantRuban"
        'TAB - Onglet Contextuel
        'GROUP - Groupe Contextuel
        'CONTROLS
        Select Case control.ID
        'Insérer l'IDENTIFIANT du Button
        Case "IdentifiantControl"
            MsgBox "Texte"
        Case "Nom"
            MsgBox "Texte"
        End Select
    End Select
'QUICK ACCESS //////////////////////////////////////
    Select Case myRibbonName
    'RIBBON - Ruban1
    Case "Ruban1"
        'CONTROLS
        Select Case control.ID
        'Insérer l'IDENTIFIANT du Button
        Case "idInstructions"
            DoCmd.SetWarnings False
            DoCmd.Echo Fals

```

```

Application.FollowHyperlink "E:\Mes Documents\Access Help Code and Ribbon\Ribbon
Creation\Notes sur l'utilisation des Rubans\Note App Ribbon.pdf"
DoCmd.SetWarnings True
DoCmd.Echo True
End Select
'RIBBON - NomRuban
Case "NomRuban"
'CONTROLS
'Insérer l'IDENTIFIANT du Button
Select Case control.ID
Case "IdentifiantControl"
MsgBox "text"
End Select
End Select
End Sub

```

COMMANDES POUVANT ÊTRE ATTRIBUÉES À UN CONTRÔLE

- 1) Les CallBacks
- 2) Les Macros
- 3) Les Les Fonctions

Ces Commandes sont attribuées au Contrôle pendant la création du Ruban dans la fenêtre des Propriétés des contrôles du Ruban dans le Champ « Fonction/Macro ».

Les **Callbacks** sont définis dans les Modules comme on vient de l'expliquer.

On les insère en se servant de la **Liste Déroulante** et en choisissant le CallBack correspondant au **type de contrôle**; le nom du contrôle en question se trouve en haut à gauche de la fenêtre.

The screenshot shows the 'Ribbon Control Properties' window. The 'Function/Macro' dropdown is open, displaying a list of possible actions. Red arrows point to the 'CALLBACKS' section, which includes 'ButtonCallbackOnAction'. The 'Type' dropdown is set to 'Visible'. The 'Enabled' dropdown is set to 'Oui'.

En insérant un **CallBack** qui se trouve dans la liste déroulante l'application ouvre une fenêtre permettant de copier l'information nécessaire pour ensuite la coller dans le Module comme aide-mémoire.

En cliquant sur Copy ID le programme ouvre le Module correspondant au type de contrôle sélectionné.

Les **Fonctions Personnalisées** doivent être créées par l'utilisateur.

Pour les associer aux contrôles désirés il suffit d'entrer le Nom de la Fonction dans le Champ « Fonction/Macro » précédée du signe = et suivie des ().

Il faut s'assurer que les Fonctions soient accessibles en les déclarant **Public** dans le Module ou dans le Formulaire.

Exemple Public FunctionTest()

Exemple de saisie dans le Champ Fonction/Macro : **=FunctionTest()**

Format Icône large *

Fonction/Macro =FunctionTest() *

SAISIES POSSIBLES

- CALLBACKS: CHOISIR dans la Liste Déroulante - EXEMPLE: ButtonCallbackOnAction
- MACROS: SAISIR le nom de la Macro - EXEMPLE: Macro1
- FUNCTIONS: SAISIR le nom de la Fonction avec = et () - EXEMPLE: =FunctionTest()

Visible Oui *

Enabled Oui *

Les **Macros** doivent être créées par l'utilisateur.

Pour les Macros il suffit d'entrer le Nom de la Macro.

Exemple : Macro1

Format Icône large *

Fonction/Macro Macro1 *

SAISIES POSSIBLES

- CALLBACKS: CHOISIR dans la Liste Déroulante - EXEMPLE: ButtonCallbackOnAction
- MACROS: SAISIR le nom de la Macro - EXEMPLE: Macro1
- FUNCTIONS: SAISIR le nom de la Fonction avec = et () - EXEMPLE: =FunctionTest()

Visible Oui *

Enabled Oui *

CONTEXTUAL TABS

```
<contextualTabs>
  <tabSet idMso="TabSetFormReportExtensibility">
    <tab id="tabContextual" label="My Contextual Tab">
      <group id="MyContextualGroup" label="My Group in
        Contextual Tab">
        <labelControl id="myLabel1a" label="Sample:"/>
        <button id="btnDemo" size="large" label="Demo"/>
      </group>
    </tab>
  </tabSet>
</contextualTabs>
```

Le tabSet « **TabSetFormReportExtensibility** » est nécessaire pour l’affichage des contextual tabs.

S’assurer que les champs pour « **image** », « **action** », « **visible** », « **enabled** » soit bien remplis

CODE À INSÉRER DANS UN FORMULAIRE UTILISANT UN ONGLET CONTEXTUEL

Option Compare Database

```
*****
'* Les trois procédures ci-dessous sont nécessaires au bon fonctionnement du menu contextuel
*
*****
```

Dim myRibbonName As String

Private Sub Form_Activate()

On Error GoTo erreur

myRibbonName = Forms!FrmRibbonElements!NameRb.Value

Me.RibbonName = myRibbonName

myRibbon.Invalidate

erreur:

If Err.Number = 91 Then Exit Sub

End Sub

Private Sub Détail_Paint()

On Error GoTo erreur

'Nécessaire à l'ouverture et à l'activation du formulaire

myRibbon.Invalidate 'Pour actualiser le Ruban

myRibbon.ActivateTab (myIDContextualTab) 'Pour sélectionner l'Onglet contextuel

'Call ChangeTabRibbon(myRibbon, myIDContextualTab, "Perso") 'Une seule étape pour actualiser le Ruban et sélectionner l'onglet; peut remplacer les deux lignes précédentes

erreur:

If Err.Number = 91 Then Exit Sub

End Sub

Private Sub Form_Deactivate()

On Error GoTo erreur

'Nécessaire à la désactivation du formulaire

myRibbon.Invalidate

'Pour actualiser le ruban'

erreur:

If Err.Number = 91 Then Exit Sub

End Sub

Désactiver des Menus Contextuels Microsoft

Exemple avec « TabSetFormDatasheet » le menu contextuel Microsoft du menu Fichier



MSo/Perso :« Mso »

Nom ID: « TabSetFormDatasheet »

Visible : « Non »

QAT Quick Access Toolbar / Barre d'outils accès rapide

La Barre d'outils accès rapide personnalisée fonctionne seulement quand la propriété du Ruban est « **StartFromScratch** » dans le fichier XML du Ruban.

```
<customUI xmlns="http://schemas.microsoft.com/office/2009/07/customui"
loadImage="CallBackLoadImage" onLoad="OnRibbonLoad">
<ribbon startFromScratch="true">
```

Structure du code XML pour le Quick Access Toolbar

```
<qat>
  <documentControls>
    <control idMso="FileCompactAndRepairDatabase"
      visible="true"
      enabled="true"/>
    <button id="idInstructions"
      label="Instructions"
      image="Button-Help-icon.bmp"
      onAction="ButtonCallbackOnAction"
      visible="true"
      enabled="true"
      screentip="Instructions sur l'application"/>
  </documentControls>
</qat>
```

Pour utiliser les Rubans dans une autre base de données

Méthode 1 avec LIAISON à Ribbon Builder.accdb

Cette méthode permet de faire des changements dans Ribbon Builder.accdb qui vont être automatiquement mis à jour dans la nouvelle base de données.

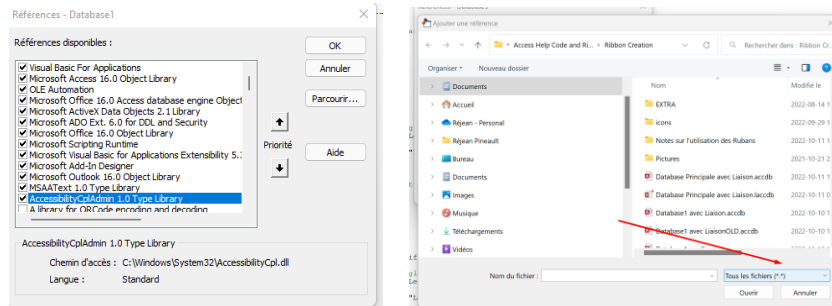
Créer la nouvelle base de données

La renommer et la sauvegarder dans le même dossier que celui qui contient les dossiers icons et Pictures

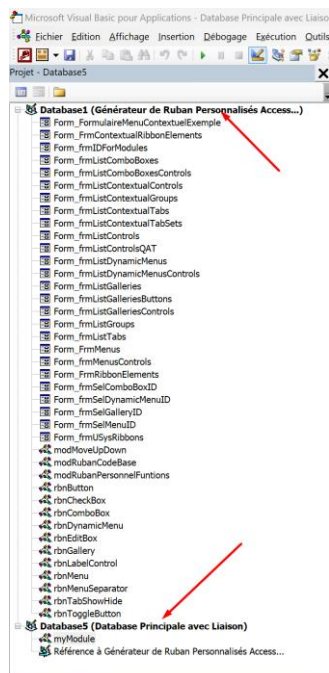
Activer toutes les références nécessaires dans la fenêtre du code Visual Basic sous outils références

Créer une référence vers Ribbon Builder.accdb en utilisant le bouton Parcourir dans la fenêtre des références.

Pour que les fichiers Access apparaissent
s'assurer de choisir le type « **Tous les fichiers (*.*)** »

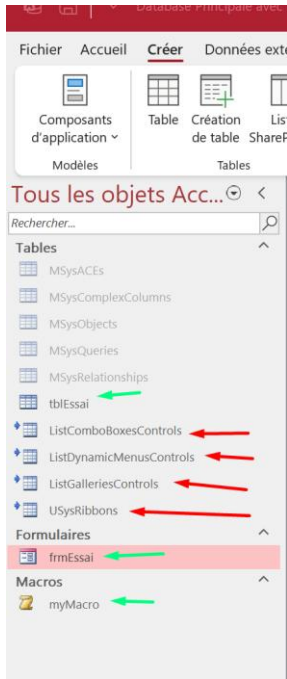


On se retrouve avec l'arborescence suivante dans l'Éditeur Visual Basic



Créer une **liaison** avec les Tables suivantes :

USysRibbons
ListComboBoxesControls
ListDynamicMenusControls
ListGalleriesControls



Les tables Liées sont en Rouge

La table, le formulaire et la Macro de la Base de données Personnelle Liée sont en Vert

Les **Macros** et les **Public Functions** présentes dans la Base de Données Personnelle Liée peuvent être insérées directement dans le Champ **Fonction/Macro** dans la fenêtre des propriétés des Buttons.

N.B. Si les fonctions se trouvent dans un **Formulaire** ou un **Rapport** Ceux-ci doivent être **ouverts** dans la Base de données Personnelle Liée pour pouvoir exécuter les Fonctions
(Ne s'applique pas aux Fonctions dans les Modules)

Code à insérer dans un Callback pour appeler les Public Procedures ou Fonctions et les Macros

Bien noter que les Procédures et les Fonctions doivent être déclarée « **Public** »

LOCAL : Fonctions présentes dans la base de données de l'Application du Ruban

EXTERNE : Fonctions présentes dans la base de données liée

LOCAL : FONCTIONS ET MACROS DANS LA BASE DE DONNÉES DE L'APPLICATION DU RUBAN

FONCTION dans un FORMULAIRE de l'Application du ruban

```
DoCmd.OpenForm "frmTestFunctionFormulaireLocal", acNormal  
Call Forms("frmTestFunctionFormulaireLocal").fctTestFunctionFormulaireLocal
```

FONCTION dans un RAPPORT de l'Application du ruban

```
DoCmd.OpenReport ("rptTestFunctionRapportLocal"), acViewReport  
Call Reports("rptTestFunctionRapportLocal").fctTestFunctionRapportLocal
```

MACRO dans l'Application du ruban

```
DoCmd.RunMacro "mcrTestMacroLocal"
```

FONCTION dans un MODULE de l'Application du Ruban

```
Call fctTestFunctionModuleLocal
```

EXTERNE : FONCTIONS ET MACROS DANS LA BASE DE DONNÉES PERSONNELLE LIÉE

FONCTION dans un FORMULAIRE de la base de données Personnelle liée

```
DoCmd.OpenForm "rptTestFunctionFormulaireExterne", acNormal  
Call Forms("frmTestFunctionFormulaireExterne").fctTestFunctionFormulaireExterne
```

FONCTION dans un RAPPORT de la base de données Personnelle liée

```
DoCmd.OpenReport ("rptTestFunctionRapportExterne"), acViewReport  
Call Reports("rptTestFunctionRapportExterne").fctTestFunctionRapportExterne
```

MACRO dans une base de données Personnelle liée

```
Application.Run "fctRunMacro", "mcrTestMacroExterne"
```

« fctRunMacro » est une fonction insérée dans un Module de la Base de données liée

```
Public Function fctRunMacro(strNomMacro)  
DoCmd.RunMacro strNomMacro  
End Function
```

FONCTION dans un MODULE de la base de données Personnelle liée

```
Application.Run "fctTestFunctionModuleExterne"
```

N.B.

Les Fonctions et les Macros EXTERNES ne fonctionnent que lorsque la Base de données Principale LIÉE est ouverte.

Sinon les commandes saisies dans le code de la base de données de l'Application ne fonctionneront pas.

Méthode 2 avec IMPORTATION des élément à partir de Ribbon Builder.accdb

Cette méthode ne mettra pas à jour les changements apportés dans Ribbon Builder.accdb
Les changements désirés doivent être faits au niveau de la table USysRibbons.

Créer la nouvelle base de données

La sauvegarder dans le même dossier que celui qui contient les dossiers icons et Pictures

Activer toutes les références nécessaires dans la fenêtre du code Visual Basic sous outils références comme pour la première méthode

Importer :

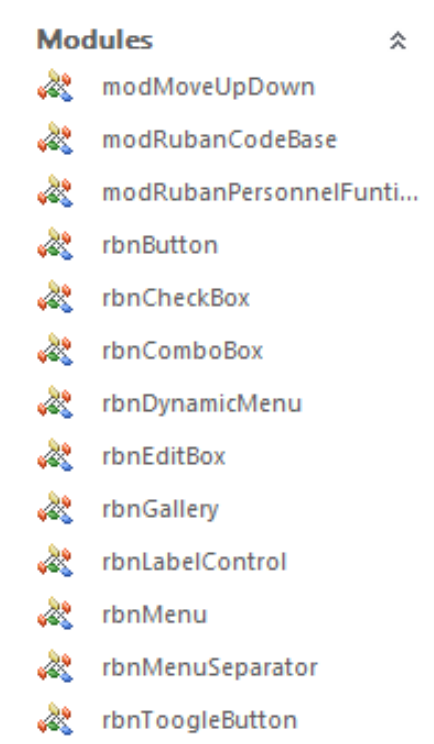
USysRibbons

ListComboBoxesControls (peuvent être édités dans la nouvelle base de données)

ListDynamicMenusControls (peuvent être édités dans la nouvelle base de données)

ListGalleriesControls (peuvent être édités dans la nouvelle base de données)

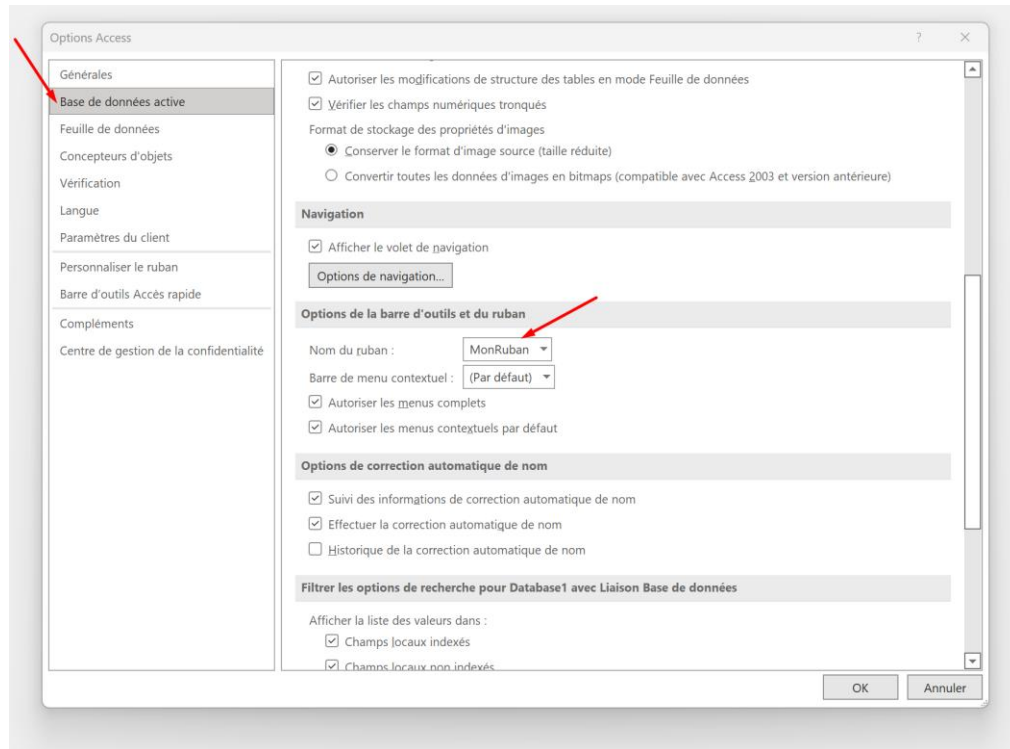
Tous les modules



Pour assigner le Ruban à la base de données complète

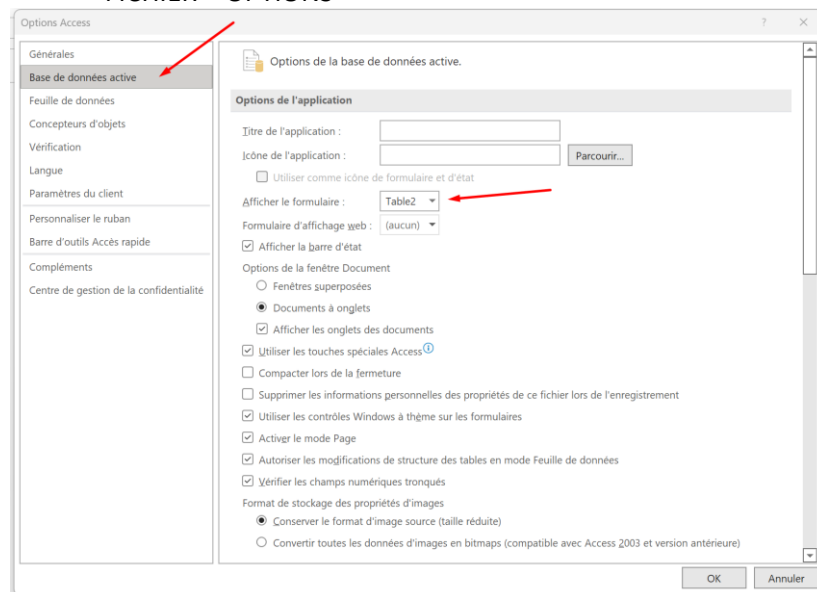
Menu **FICHIER**

Menu **OPTIONS**



Pour ouvrir avec un FORMULAIRE particulier

FICHIER – OPTIONS



Toutes les commandes intégrées dans le ruban de Microsoft

Access

On peut exécuter ces commandes en utilisant :

Le **nom** de la constante ou la **valeur** de la constante

'Affiche la boîte de dialogue A propos de Microsoft Access en utilisant le **nom** de la constante

Application.RunCommand acCmdAboutMicrosoftAccess

DoCmd.RunCommand acCmdAboutMicrosoftAccess

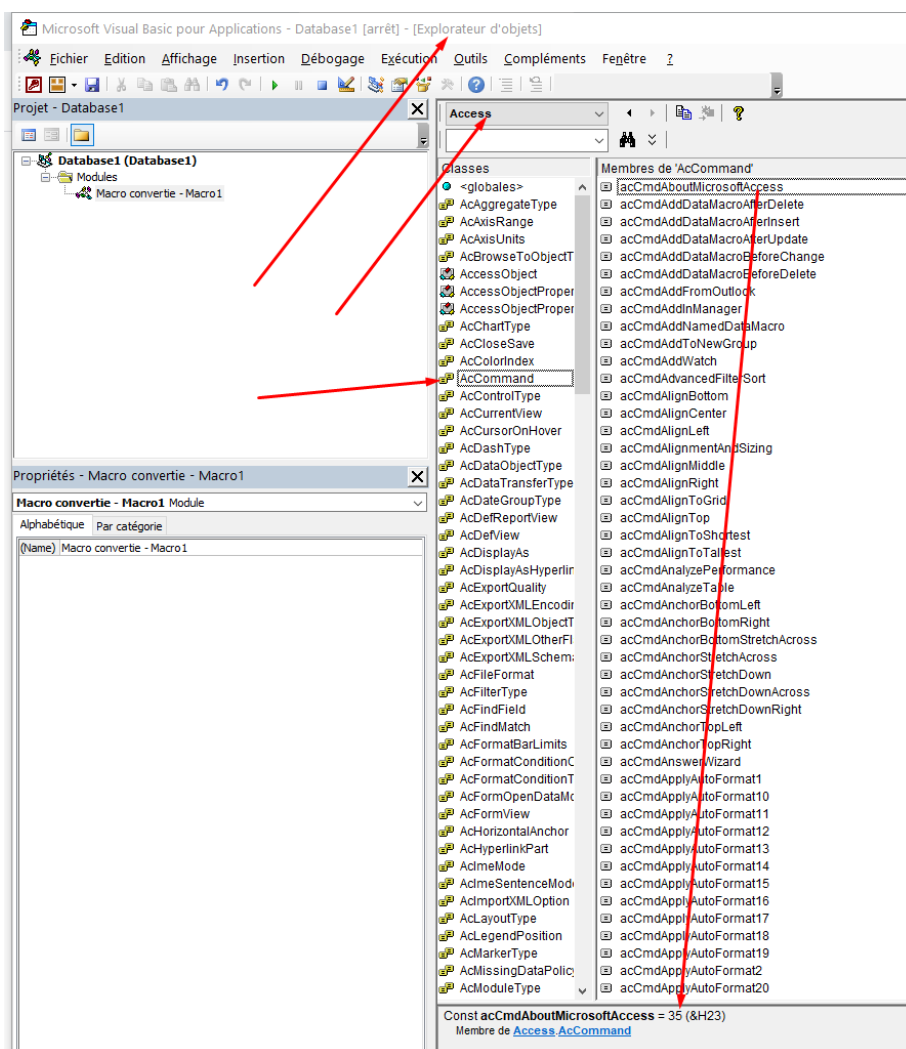
RunCommand acCmdAboutMicrosoftAccess

'Affiche la boîte de dialogue A propos de Microsoft Access en utilisant la **valeur** de la constante

Application.RunCommand 35

DoCmd.RunCommand 35

RunCommand 35



Nom et images des commandes Microsoft

<https://bert-toolkit.com/imagenso-list.html>

Code pour ouvrir une page HTML

```
Case "idImageMso"  
Dim myURL as String  
myURL = "https://bert-toolkit.com/imagenso-list.html"  
Application.FollowHyperlink myURL, ,True
```

Code pour ouvrir un fichier

```
Case "idImageMso"  
DoCmd.SetWarnings False  
DoCmd.Echo False  
Application.FollowHyperlink CurrentProject.Path & "\\docs\\imageMso Gallery.pdf"  
DoCmd.SetWarnings True  
DoCmd.Echo True
```

Cette dernière commande est utilisée dans le module rbnButton sous la section QuickAccess.