

Personnalisation du ruban: Les fonctions d'appel CallBacks

par [SilkyRoad \(silkyroad.developpez.com\)](http://silkyroad.developpez.com)

Date de publication : 16 Juillet 2008

Dernière mise à jour : 02 Aout 2008

Ce tutoriel présente les fonctions d'appel CallBacks sous Excel2007.

I - Introduction.....	3
II - Description des arguments contenus dans les fonctions d'appel.....	3
III - Les fonctions d'appel.....	4
III-1 - onLoad.....	4
III-2 - loadImage.....	5
III-3 - getEnabled.....	6
III-4 - onAction.....	8
III-5 - getVisible.....	9
III-6 - getKeytip.....	10
III-7 - getLabel.....	11
III-8 - getImage.....	13
III-9 - getScreentip.....	13
III-10 - getSupertip.....	14
III-11 - getDescription.....	15
III-12 - getShowImage.....	16
III-13 - getShowLabel.....	16
III-14 - getSize.....	16
III-15 - getPressed.....	17
III-16 - getItemCount.....	18
III-17 - getItemID.....	18
III-18 - getItemImage.....	19
III-19 - getItemLabel.....	22
III-20 - getItemScreentip.....	22
III-21 - getItemSupertip.....	22
III-22 - getText.....	22
III-23 - onChange.....	23
III-24 - getSelectedItemID.....	23
III-25 - getSelectedItemIndex.....	24
III-26 - getContent.....	24
III-27 - getItemHeight.....	26
III-28 - getItemWidth.....	26
III-29 - getTitle.....	26
IV - Liens.....	27
V - Remerciements.....	27
VI - Téléchargement.....	27

I - Introduction

Le tutoriel précédent ([La personnalisation du ruban](#)), présentait les attributs et les contrôles utilisables dans vos projets. Ce nouvel article décrit les fonctions d'appel VBA (CallBacks).

Une partie des attributs contenus dans le fichier xml de personnalisation peuvent être associés à des fonctions d'appel VBA. Les paramètres définis dans votre code xml sont alors liés à des procédures placées dans un module standard du classeur. Vous pouvez déclencher une macro depuis le ruban (à l'aide des événements onAction, onChange ...), mais également gérer dynamiquement, paramétrer et modifier les attributs des contrôles par VBA, en leur ajoutant le préfixe **get**.

La caractéristique d'un attribut **getNomAttribut** est de pouvoir mettre à jour dynamiquement sa propriété NomAttribut. Par exemple, au lieu d'indiquer le paramètre `itemHeight="LaValeur"` explicitement en dur dans le fichier xml de personnalisation, vous écrirez `getItemHeight="NomMacroDefinitionHauteurItem"`.

Ensuite, il vous restera à placer la fonction d'appel dans un module standard du classeur:

`Sub NomMacroDefinitionHauteurItem(control As IRibbonControl, ByRef returnedVal)`.

Les chapitres suivants décrivent les fonctions disponibles et proposent quelques exemples d'utilisation.

Remarques :

Bien entendu les CallBacks ne fonctionnent pas si vous désactivez les macros à l'ouverture du classeur.

Attention à bien respecter la casse lorsque vous rédigez vos codes dans le fichier xml de personnalisation.

II - Description des arguments contenus dans les fonctions d'appel

Les fonctions sont constituées d'arguments qui récupèrent des informations dans le ruban ou y renvoient des paramètres.

Voici une description des éléments présents dans les fonctions CallBacks:

L'argument **control**.

Exemple: `Sub ___OnAction(control As IRibbonControl)`

Cet argument identifie le contrôle manipulé par VBA. Il possède 3 propriétés en lecture seule:

* **id** est l'identifiant unique qui permet d'identifier chaque contrôle. Si vous attribuez la même procédure à plusieurs contrôles du fichier xml de personnalisation, l'id vous permet de repérer lequel est en cours d'utilisation.

* **tag** est l'information personnelle complémentaire du contrôle, que vous avez défini dans le fichier xml.

* **context** (Je n'ai pas réussi à le faire fonctionner).

Le mot clé **Byref**.

Exemple : `Sub ___GetEnabled(control As IRibbonControl, ByRef enabled)`

Lorsqu'un argument est précédé du mot clé ByRef, cela signifie que la donnée spécifiée dans votre fonction va être utilisée pour mettre à jour l'attribut du contrôle.

Quand la procédure est appelée par l'application, un objet IRibbonControl représentant le contrôle est défini. Le code vérifie la propriété 'id' de l'objet et en fonction de sa valeur, il assigne une donnée à la variable. Par exemple : `enabled = True`.

L'argument **pressed**.

Exemple : `Sub ___OnAction(control As IRibbonControl, pressed As Boolean)`

Cet argument renvoie la valeur Vrai si le contrôle est coché (pour les checkBox) ou si la touche est enfoncée (pour les toggleButton). La valeur Faux est renvoyée dans le cas contraire.

L'argument **index**.

Exemple : `Sub ___GetItemLabel(control As IRibbonControl, index As Integer, ByRef label)`

Cet argument définit le numéro de l'élément dans les contrôles de type menu déroulant (ComboBox, dropDown , Gallery).

0= le premier élément dans la liste

1= le deuxième élément

... etc ...

L'argument **Id**.

Exemple : `Sub NomProcedure(control As IRibbonControl, Id As String, index As Integer)`

Cet argument renvoie l'identificateur unique de l'item sélectionné (pour les contrôles dropDown et Gallery).

L'argument **text**.

Exemple : `Sub ___OnChange(control As IRibbonControl, text As String)`

Renvoie la donnée contenue dans les contrôles de saisie (editBox, comboBox).

III - Les fonctions d'appel

III-1 - onLoad

La fonction **onLoad** s'applique au contrôle **customUI**.

Elle définit la procédure VBA qui doit être déclenchée lors du chargement du ruban.

Il est possible d'appliquer des actualisations à n'importe quel moment par macro, grâce à la fonction de rappel **onLoad**.

Ce rappel est déclenché une seule fois, après l'ouverture du classeur. Celui-ci va charger une variable objet qui déclarée avec un type **IRibbonUI** et sera placée dans un module standard.

Par exemple : [Public objRuban As IRibbonUI](#).

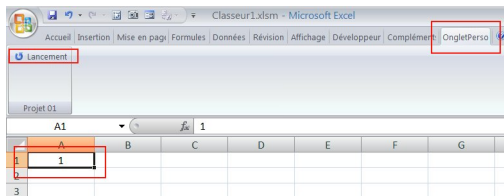
L'objet IRibbonUI possède deux méthodes :

Invalidate() qui actualise en une seule fois tous les contrôles personnalisés du classeur.

InvalidateControl(ControlID As String) qui actualise un contrôle particulier (ControlID correspond à l'identificateur unique du contrôle).

Cet exemple active un bouton de manière conditionnelle.

Le contrôle est utilisable uniquement si le contenu de la cellule A1 contient la valeur 1.



Placez ce code dans le fichier xml de personnalisation :

Xml

```

<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui" onLoad="RubanCharge">
<!-- onLoad="RubanCharge" est déclenché lors du chargement du ruban personnalisé. -->
<ribbon startFromScratch="false">
<tabs>

<tab id="OngletPerso" label="OngletPerso" visible="true">
  <group id="Projet01" label="Projet 01">

    <!-- onAction="ProcLancement" définit la macro déclenchée lorsque vos cliquez sur le bouton. -->
    <!-- getEnabled="Bouton1_Enabled" gère la condition d'activation ou de désactivation. -->
    <button id="Bouton1" label="Lancement" onAction="ProcLancement" size="normal"
      imageMso="Repeat" getEnabled="Bouton1_Enabled"/>

  </group>
</tab>

</tabs>
</ribbon>
</customUI>

```

Ouvrez le classeur Excel. Validez l'éventuel message d'erreur "Impossible d'exécuter la macro 'RubanCharge'...". Ce message est normal car le classeur ne contient pas encore les fonctions de rappel.

Placez ce code dans un module standard :

Vba

```

Option Explicit

Public objRuban As IRibbonUI
Public boolResult As Boolean

'Callback for customUI.onLoad
'Est déclenché lors du chargement du ruban personnalisé.

```

```

Vba
Sub RubanCharge(ribbon As IRibbonUI)
    boolResult = False
    Set objRuban = ribbon
End Sub

'Callback for Bouton1 onAction
'La procédure déclenchée lorsque vous cliquez sur le bouton.
Sub ProcLancement(control As IRibbonControl)
    MsgBox "ma procédure."
End Sub

'Callback for Bouton1 getEnabled
'Active ou désactive le bouton en fonction de la variable boolResult
Sub Bouton1_Enabled(control As IRibbonControl, ByRef returnedVal)
    returnedVal = boolResult
End Sub
    
```

Et placez ce code dans le **module objet Worksheet** de la feuille (où vous allez modifier le contenu de la cellule A1):

```

Vba

Option Explicit

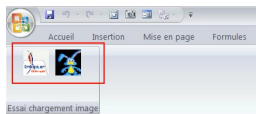
'Evenement Change dans la feuille de calcul.
Private Sub Worksheet_Change(ByVal Target As Range)
    'Vérifie si la cellule A1 est modifiée et si la cellule contient la valeur 1.
    If Target.Address = "$A$1" And Target = 1 Then
        boolResult = True
    Else
        boolResult = False
    End If

    'Rafraichit le bouton personnalisé
    If Not objRuban Is Nothing Then objRuban.InvalidateControl "Bouton1"
End Sub
    
```

Refermez puis ré-ouvrez le classeur (les macros doivent impérativement être activées).
 La variable "objRuban" va être chargée depuis la procédure "RubanCharge" : **Set objRuban = ribbon**
 Si les conditions sont ensuite remplies (en fonction des modifications dans la cellule A1), le contrôle "Bouton1" va être activé ou désactivé.
 Rappel:
 Pour les contrôles de type comboBox, gallery et dynamicMenu, vous pouvez insérer l'attribut **invalidateContentOnDrop** dans le fichier xml de personnalisation afin d'actualiser automatiquement le contenu du contrôle lorsque celui-ci est sélectionné.

III-2 - loadImage

La fonction **loadImage** s'applique au contrôle **customUI**.
 Elle définit la procédure VBA qui va charger des images. De cette manière, toutes les images sont récupérées en une seule fois, au moment où vous activez un élément personnalisé de votre ruban.
 Cet exemple suppose qu'il existe des images nommées "logo16.gif" et "lapin4.gif" dans le répertoire "C:\dossier\". Les noms de fichiers sont spécifiés dans l'attribut image (image="logo16.gif").



Placez ce code dans le fichier xml de personnalisation :

```

Xml
    
```

Xml

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui"
  loadImage="MacroChargementImage">

  <ribbon>
    <tabs>
      <tab id="Tab01" label="Test" >
        <group id="Groupe01" label="Essai chargement image">
          <button id="button01"
            size="large"
            image="logol6.gif"/>

          <button id="button02"
            size="large"
            image="lapin4.gif"/>
        </group>
      </tab>
    </tabs>
  </ribbon>

</customUI>
```

Et la fonction de chargement d'image, à placer dans un module standard :

Vba

```
'Callback for customUI.loadImage
Sub MacroChargementImage(imageID As String, ByRef returnedVal)
  Set returnedVal = LoadPicture("C:\dossier\" & imageID)
End Sub
```

III-3 - getEnabled

La fonction **getEnabled** s'applique aux contrôles **command**, **button**, **checkBox**, **comboBox**, **dropDown**, **dynamicMenu**, **editBox**, **gallery**, **labelControl**, **menu**, **splitButton**, **toggleButton**.

Elle spécifie une valeur booléenne qui indique si le contrôle doit être activé (true ou 1) ou désactivé (false ou 0).

Cet exemple active un bouton personnalisé uniquement si la cellule A1 contient la valeur 1.

Dans le fichier xml de personnalisation :

Xml

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui" onLoad="RubanCharge">
<!-- onLoad="RubanCharge" est déclenché lors du chargement du ruban personnalisé. -->
<ribbon startFromScratch="false">
<tabs>

<tab id="OngletPerso" label="OngletPerso" visible="true">
  <group id="Projet01" label="Projet 01">

    <!-- onAction="ProcLancement" définit la macro déclenchée lorsque vos cliquez sur le bouton. -->
    <!-- getEnabled="Bouton1_Enabled" gère la condition d'activation ou de désactivation. -->
    <button id="Bouton1" label="Lancement" onAction="ProcLancement" size="normal"
      imageMso="Repeat" getEnabled="Bouton1_Enabled"/>

  </group>
</tab>

</tabs>
</ribbon>
</customUI>
```

Dans un module standard :

Vba

```

Option Explicit

Public MonRuban As IRibbonUI
Public boolResult As Boolean

'Callback for customUI.onLoad
'Est déclenché lors du chargement du ruban personnalisé.
Sub RubanCharge(ribbon As IRibbonUI)
    boolResult = False
    Set MonRuban = ribbon
End Sub

'Callback for Bouton1 onAction
'La procédure déclenchée lorsque vous cliquez sur le bouton.
Sub ProcLancement(control As IRibbonControl)
    MsgBox "ma procédure."
End Sub

'Callback for Bouton1 getEnabled
'Active ou désactive le bouton en fonction de la variable boolResult
Sub Bouton1_Enabled(control As IRibbonControl, ByRef returnedVal)
    returnedVal = boolResult
End Sub
    
```

Dans le module objet Worksheet de la feuille (où vous allez modifier le contenu de la cellule A1):

Vba

```

'Evenement Change dans la feuille de calcul.
Private Sub Worksheet_Change(ByVal Target As Range)

    'Vérifie si la cellule A1 est modifiée et si la cellule contient la valeur 1.
    If Target.Address = "$A$1" Then _
        boolResult = Target = 1

    'Rafraichit le bouton personnalisé
    If Not MonRuban Is Nothing Then MonRuban.InvalidateControl "Bouton1"
End Sub
    
```

Les fonctions d'appel peuvent également être appliquées aux commandes prédéfinies.
Un exemple qui désactive la commande "Copier" si le classeur est ouvert en lecture seule :

Xml

```

<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
  <commands>

    <command idMso="Copy"
      getEnabled="macroGetEnabled"/>

  </commands>
</customUI>
    
```

Dans un module standard du classeur :

Vba

```

'Callback for Copy getEnabled
Sub macroGetEnabled(control As IRibbonControl, ByRef returnedVal)
    'Désactive la commande si le classeur est ouvert en lecture seule.
    returnedVal = Not ThisWorkbook.ReadOnly
End Sub
    
```

Vba

```
'Nota:
'Le raccourci clavier (Ctrl+C) n'est pas désactivé.
End Sub
```

III-4 - onAction

La fonction **onAction** s'applique à tous les contrôles.

Elle est déclenchée lorsque vous cliquez sur le contrôle afin de l'utiliser.

Vous trouverez de nombreux exemples dans les différents chapitres de ce tutoriel.

Une deuxième utilisation de la fonction onAction, consiste à désactiver temporairement et réattribuer des procédures personnelles aux commandes prédéfinies du ruban (onAction Repurposed).

Dans ce cas, la procédure contient un deuxième argument : 'cancelDefault'.

Sub NomProcédure(control As IRibbonControl, ByRef cancelDefault).

Pour désactiver la commande prédéfinie, spécifiez **cancelDefault = True**.

Pour réinitialiser la commande, indiquez **cancelDefault = False**.

Cet exemple ajoute un onglet 'test' qui contient un bouton bascule (toggleButton). Si vous cliquez dessus, la commande prédéfinie 'Recherche' (onglet Accueil/groupe Edition) est désactivée. Réutilisez le bouton pour réinitialiser la commande 'Recherche'.

Placez ce code dans le fichier xml de personnalisation du classeur :

Xml

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui"
onLoad="RubanCharge" >
  <commands>
    <command idMso="FindDialogExcel" onAction="MaRecherchePerso" />
  </commands>
  <ribbon startFromScratch="false">
    <tabs>
      <tab id="Tab01" label="Test" >
        <group id="Groupe01" label="essai de réattribution d'un contrôle prédéfini">
          <toggleButton id="ToggleButton01"
            imageMso="SheetProtect"
            size="large"
            getLabel="MacroGetLabel"
            onAction="ModifStatutBoutonRecherche" />
        </group>
      </tab>
    </tabs>
  </ribbon>
</customUI>
```

Dans le module objet "ThisWorkbook" du classeur :

Vba

```
Option Explicit

Private Sub Workbook_Open()
  boolResult = False
End Sub
```

Dans un module standard du classeur :

Vba

```
Option Explicit
Public boolResult As Boolean
Public objRuban As IRibbonUI
```

Vba

```
'Callback for customUI.onLoad
'Est déclenché lors du chargement du ruban personnalisé.
Sub RubanCharge(ribbon As IRibbonUI)
    Set objRuban = ribbon
End Sub

'Callback for ToggleButton01 onAction
'Récupère la valeur du bouton bascule puis désactive ou réinitialise la commande
'prédéfinie 'Recherche'
Sub ModifStatutBoutonRecherche(control As IRibbonControl, pressed As Boolean)
    boolResult = pressed
    objRuban.Invalidate
End Sub

'Callback for FindDialogExcel onAction
'Réattribue temporairement la commande 'Recherche' ou la réinitialise
'en fonction de la valeur de la variable boolResult (utilisation du toggleButton)
Sub MaRecherchePerso(control As IRibbonControl, ByRef cancelDefault)
    If boolResult Then
        MsgBox "Le bouton 'Recherche' est temporairement désactivé"
        'Vous pouvez éventuellement ajouter vos procédures de
        'recherche personnelles.
        '
        cancelDefault = True
    Else
        cancelDefault = False
    End If
End Sub

'Callback for ToggleButton01 getLabel
'Attribue une étiquette au bouton bascule en fonction de sa position
Sub MacroGetLabel(control As IRibbonControl, ByRef returnedVal)
    If boolResult Then
        returnedVal = "Bouton 'Recherche' Désactivié"
    Else
        returnedVal = "Bouton 'Recherche' opérationnel"
    End If
End Sub
```

III-5 - getVisible

La fonction **getVisible** s'applique aux contrôles **tabSet**, **tab**, **group**, **box**, **button**, **buttonGroup**, **checkBox**, **comboBox**, **dropDown**, **dynamicMenu**, **editBox**, **gallery**, **labelControl**, **menu**, **separator**, **splitButton**, **toggleButton**.

Elle renvoie une valeur booléenne qui indique si le contrôle est visible (true) ou masqué (false).

Dans cet exemple, le bouton "Copier" qui placé dans un onglet personnel sera visible uniquement si le nom de l'utilisateur de la session est "mimi".

Placez ce code dans le fichier xml de personnalisation :

Xml

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
  <ribbon startFromScratch="false">
    <tabs>
      <tab id="TB01" label="Test">
        <group id="GR01" label="Essais de bouton">
          <button idMso="Copy" size="normal" getVisible="SiUtilisateur" />
        </group>
      </tab>
    </tabs>
  </ribbon>
</customUI>
```

Xml

```
</ribbon>
</customUI>
```

Dans un module standard du classeur:

Vba

```
'Callback for Copy getVisible
Sub SiUtilisateur(control As IRibbonControl, ByRef returnedVal)
    returnedVal = Environ("username") = "mimi"
End Sub
```

Remarque :

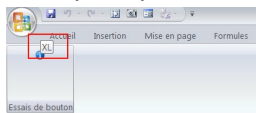
La fonction d'appel getVisible ne marche pas sur les onglets prédéfinis si ceux-ci ont préalablement été masqués par l'attribut **startFromScratch="true"**.

III-6 - getKeytip

La fonction **getKeytip** s'applique aux contrôles **tab**, **group**, **button**, **checkBox**, **comboBox**, **dropDown**, **dynamicMenu**, **editBox**, **gallery**, **menu**, **splitButton**, **toggleButton**.

Elle définit la procédure VBA qui associera un raccourci clavier au contrôle. Les raccourcis clavier sont composés de 1 à 3 caractères maximum et sont accessibles après l'utilisation de la touche ALT.

Exemple à placer dans le fichier de personnalisation, pour associer le raccourci "XL" au bouton personnalisé :



Xml

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
  <ribbon startFromScratch="false">
    <tabs>
      <tab id="TB01" label="Test">
        <group id="GR01" label="Essais de bouton">

          <button id="bouton01"
            imageMso="RefreshStatus"
            onAction="MacroBouton"
            size="normal"
            getKeytip="DefinitRaccourci" />

        </group>
      </tab>
    </tabs>
  </ribbon>
</customUI>
```

Dans un module standard :

Vba

```
Option Explicit

'Callback for bouton01 getKeytip
'Définit le raccourci clavier qui va être associé au bouton.
Sub DefinitRaccourci(control As IRibbonControl, ByRef returnedVal)
    returnedVal = "XL"
End Sub

'Callback for bouton01 onAction
```

Vba

```

Sub MacroBouton(control As IRibbonControl)
    MsgBox "Vous avez cliqué sur le bouton '" & control.id & "'"
End Sub

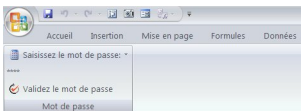
```

III-7 - getLabel

La fonction **getLabel** s'applique aux contrôles **tab**, **group**, **button**, **checkBox**, **comboBox**, **dropDown**, **dynamicMenu**, **editBox**, **gallery**, **labelControl**, **menu**, **toggleButton**.

Elle définit la chaîne de caractères qui va s'afficher dans l'attribut "label" (étiquette) du contrôle.

Le code suivant permet de saisir un mot de passe depuis le ruban et de l'afficher sous forme d'astérisques dans le label (getLabel="ContenuLabel").



Dans le fichier xml de personnalisation :

Xml

```

<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui" onLoad="objRuban">

  <ribbon>
    <tabs>
      <tab id="Essai" label="Essai" >
        <group id="MdP" label="Mot de passe">

          <gallery id="gallery01"
            size="normal"
            imageMso="CalculateNow"
            label="Saisissez le mot de passe:"
            columns="6"
            rows="6"
            getItemCount="NbCaracteres"
            showItemLabel="true"
            getItemLabel="LabelCaractere"
            screentip="Sélectionnez les caractères dans la galerie,&#13;
              puis validez le mot de passe."
            onAction="SelectCaractere" >

            <button id="Bt02" label="Effacer la saisie."
              imageMso="ClearMenu" onAction="EffaceContenuLabel" />

          </gallery>

          <box id="Box01" boxStyle="horizontal">
            <labelControl id="LC01" getLabel="ContenuLabel" />

          </box>

          <button id="Bt01" imageMso="FileStartWorkflow" size="normal"
            label="Validez le mot de passe" visible="true" onAction="ValidationMdP" />

        </group>
      </tab>
    </tabs>
  </ribbon>
</customUI>

```

Dans le **module objet "ThisWorkbook"** du classeur :

Vba

```

Option Explicit

```

Vba

```
'Définit les caractères utilisables pour la saisie du mot de passe
Private Sub Workbook_Open()
    Tableau = Array("A", "B", "C", "D", "E", "F", "G", "H", "I", "J", _
        "K", "L", "M", "N", "O", "P", "Q", "R", "S", "T", "U", "V", _
        "W", "X", "Y", "Z", "0", "1", "2", "3", "4", "5", "6", "7", "8", "9")
End Sub
```

Dans un module standard du classeur :

Vba

```
Option Explicit
Option Base 1

Public Cible As String
Public MonRuban As IRibbonUI
Public Tableau As Variant

'Callback for customUI.onLoad
'Définit l'objet ruban
Sub objRuban(ribbon As IRibbonUI)
    Set MonRuban = ribbon
End Sub

'Callback for gallery01 getItemCount
'Définit le nombre d'élément dans la galerie
Sub NbCaracteres(control As IRibbonControl, ByRef returnedVal)
    returnedVal = UBound(Tableau) + 1
End Sub

'Callback for gallery01 getItemLabel
'Création des éléments dans la galerie
Sub LabelCaractere(control As IRibbonControl, index As Integer, ByRef returnedVal)
    returnedVal = Tableau(index)
End Sub

'Callback for gallery01 onAction
'Met à jour le label après avoir sélectionné un caractère dans la galerie
Sub SelectCaractere(control As IRibbonControl, id As String, index As Integer)
    Cible = Cible & Tableau(index)
    MonRuban.InvalidateControl "LC01"
End Sub

'Callback for LC01 getLabel
'Affiche des asteriques dans le label en lieu et place des caractères
Sub ContenuLabel(control As IRibbonControl, ByRef returnedVal)
    returnedVal = Application.WorksheetFunction.Rept(" ", Len(Cible))
End Sub

'Callback for Bt01 onAction
'Validaton du mot de passe
Sub ValidationMdP(control As IRibbonControl)
    MsgBox "Confirmation du mot de passe : (" & Cible & ")"

    Cible = ""
    MonRuban.InvalidateControl "LC01"
End Sub

'Callback for Bt02 onAction
'réinitialisation: Efface le contenu du label en cas d'erreur
'de saisie.
```

Vba

```
Sub EffaceContenuLabel(control As IRibbonControl)
    Cible = ""
    MonRuban.InvalidateControl "LC01"
End Sub
```

III-8 - getImage

La fonction **getImage** s'applique aux contrôles **group, button, comboBox, dropDown, dynamicMenu, editBox, gallery, menu, toggleButton**.

Elle permet de charger ou de modifier dynamiquement par VBA une image stockée sur le disque dur. L'objet image qui va être passé à l'argument **returnedVal** doit être de type IPictureDisp (Utilisez la fonction LoadPicture).

L'exemple permet de charger une image stockée dans le répertoire "C:\dossier\" et dont le nom est spécifié dans l'attribut "tag" du bouton.

Dans le fichier xml de personnalisation :

Xml

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">

  <ribbon>
    <tabs>
      <tab id="Tab01" label="Test" >
        <group id="Groupe01" label="Essai chargement image">
          <button id="button01"
            getImage="MacroChargementImage"
            tag="logol6.gif"/>
        </group>
      </tab>
    </tabs>
  </ribbon>

</customUI>
```

Dans un module standard de votre classeur :

Vba

```
'Callback for button01 getImage
Sub MacroChargementImage(control As IRibbonControl, ByRef returnedVal)
    Set returnedVal = LoadPicture("C:\dossier\" & control.Tag)
End Sub
```

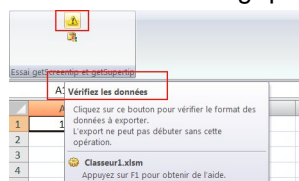
A noter que la fonction d'appel getImage ne fonctionne pas pour les images personnelles stockées directement dans le classeur (rels\customUI\images\NomFichier.png).

III-9 - getScreentip

La fonction **getScreentip** s'applique aux contrôles **group, button, checkBox, comboBox, dropDown, dynamicMenu, editBox, gallery, labelControl, menu, toggleButton**.

Elle définit la procédure VBA qui associe une petite info-bulle d'information pour le contrôle.

Cet exemple crée une info-bulle pour chaque bouton. Les données à afficher sont stockées dans la colonne A de la Feuil2. L'attribut "tag" permet de définir la ligne qui doit être associée à chaque contrôle.



Dans le fichier xml de personnalisation :

Xml

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
  <ribbon>
    <tabs>
      <tab id="Tab01" label="Test" >
        <group id="Groupe01" label="Essai getScreentip et getSupertip">

<button id = "bouton01"
imageMso="MacroSecurity"
getScreentip="AideBouton"
getSupertip="AideComplementaire"
onAction = "MaProcedure"
tag="1" />

<button id = "bouton02"
imageMso="ShowClipboard"
getScreentip="AideBouton"
getSupertip="AideComplementaire"
onAction = "MaProcedure"
tag="2" />
        </group>
      </tab>
    </tabs>
  </ribbon>
</customUI>
```

Dans un module standard du classeur :

Vba

```
'Callback for bouton01 getScreentip
Sub AideBouton(control As IRibbonControl, ByRef returnedVal)
  'L'infobulle est stockée dans la colonne A de la Feuil2
  'l'attribut Tag permet de définir la ligne correspondante
  returnedVal = Worksheets("Feuil2").Cells(CDec(control.Tag), 1)
End Sub

'Callback for bouton01 getSupertip
Sub AideComplementaire(control As IRibbonControl, ByRef returnedVal)
  'L'infobulle est stockée dans la colonne A de la Feuil2
  returnedVal = Worksheets("Feuil2").Cells(CDec(control.Tag), 2)
End Sub

'Callback for bouton01 onAction
'Macro associée aux boutons
Sub MaProcedure(control As IRibbonControl)
  MsgBox "Ma procédure lancée depuis le bouton '" & control.id & "'"
End Sub
```

III-10 - getSupertip

La fonction **getSupertip** s'applique aux contrôles **group**, **button**, **checkBox**, **comboBox**, **dropDown**, **dynamicMenu**, **editBox**, **gallery**, **labelControl**, **menu**, **splitButton**, **toggleButton**.

Elle définit une info-bulle complémentaire pour le contrôle. Celui-ci s'affiche à la suite des informations indiquées dans l'attribut "screentip".

Consultez l'exemple proposé dans le chapitre **getScreenTip**.

Les données à afficher sont stockées dans la colonne B de la Feuil2. L'attribut "tag" permet de définir la ligne qui doit être associée à chaque contrôle.

Vous noterez que les éventuels retours à la ligne que vous appliquez dans les cellules (ALT+ENTREE) apparaissent également dans l'info-bulle.



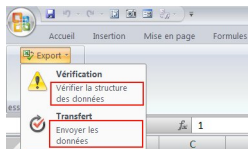
Dans le fichier xml de personnalisation :

III-11 - getDescription

La fonction **getDescription** s'applique aux contrôles **button**, **checkBox**, **dynamicMenu**, **gallery**, **menu**, **toggleButton**.

Elle définit la chaîne de caractères qui va s'afficher dans l'attribut "description" du contrôle.

L'exemple ci-dessous utilise la fonction d'appel getDescription afin de compléter la description des boutons, depuis le code VBA.



Xml

```

<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
  <ribbon startFromScratch="false">
    <tabs>
      <tab id="Tab01" label="Test">
        <group id="Groupe01" label="essai getDescription">

          <menu id="menu01"
            label="Export"
            imageMso="FileCompatibilityChecker"
            itemSize="large" >

            <button id="bouton01"
              label="Vérification"
              imageMso="MacroSecurity"
              getDescription="MacroDescription"
              onAction="MacroVerif"/>

            <button id="bouton02"
              label="Transfert"
              imageMso="FileStartWorkflow"
              getDescription="MacroDescription"
              onAction="MacroTransfert"/>

          </menu>
        </group>
      </tab>
    </tabs>
  </ribbon>
</customUI>
  
```

Dans un module standard du classeur :

Vba

```

'Callback for bouton01 getDescription
'Une description est associée à chaque bouton
Sub MacroDescription(control As IRibbonControl, ByRef returnedVal)
  Select Case control.id
    Case "bouton01": returnedVal = "Vérifier la structure des données"
    Case "bouton02": returnedVal = "Envoyer les données"
  End Select
End Sub
  
```

Vba

```

End Select
End Sub

'Callback for bouton01 onAction
Sub MacroVerif(control As IRibbonControl)
End Sub

'Callback for bouton02 onAction
Sub MacroTransfert(control As IRibbonControl)
End Sub

```

III-12 - getShowImage

La fonction **getShowImage** s'applique aux contrôles **button**, **comboBox**, **dropDown**, **dynamicMenu**, **editBox**, **gallery**, **menu**, **toggleButton**.

Elle renvoie une valeur booléenne qui va indiquer si l'image associée au contrôle doit être affichée ou pas.

Consultez le chapitre consacré à **getItemImage** (getShowImage="MacroShowImage") pour visualiser un exemple.

III-13 - getShowLabel

La fonction **getShowLabel** s'applique aux contrôles **button**, **comboBox**, **dropDown**, **dynamicMenu**, **editBox**, **gallery**, **labelControl**, **menu**, **splitButton**, **toggleButton**.

Elle renvoie une valeur booléenne qui va indiquer si l'étiquette associée au contrôle doit être affichée ou pas.

Consultez le chapitre consacré à **getItemImage** (getShowLabel="MacroShowLabel") pour visualiser un exemple.

III-14 - getSize

La fonction **getSize** s'applique aux contrôles **button**, **dynamicMenu**, **gallery**, **menu**, **splitButton**, **toggleButton**.

Elle attribue une taille au contrôle. Vous pouvez spécifier 2 valeurs :

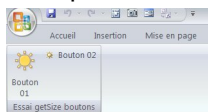
* 0 indique que la taille est 'normal'.

* 1 indique que la taille est 'large'.

Une taille 'normal' tient sur une ligne dans le ruban, et l'étiquette est positionnée sur le côté de l'icône.

Une taille 'large' correspond à 3 lignes 'normal' de hauteur, et l'étiquette est positionnée sous l'icône.

Cet exemple place deux boutons dans le ruban: le premier de taille 'large' et le deuxième de taille 'normal' afin que vous puissiez visualiser la différence de présentation entre les deux options.



Dans le fichier xml de personnalisation :

Xml

```

<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
  <ribbon startFromScratch="false">
    <tabs>
      <tab id="tab1" label="Test" >
        <group id="GR01" label="Essai getSize boutons">
          <button id="bouton01"
imageMso="PictureBrightnessGallery"
          label="Bouton 01"
          getSize="MacroGetSize" />
          <button id="bouton02"
imageMso="PictureBrightnessGallery"
          label="Bouton 02"
          getSize="MacroGetSize" />
        </group>
      </tab>
    </tabs>
  </ribbon>
</customUI>

```

Xml

```

        </tab>
    </tabs>
</ribbon>
</customUI>
    
```

Dans un module standard du classeur :

Vba

```

'Callback for bouton01 getSize
'Attribue une taille aux boutons
Sub MacroGetSize(control As IRibbonControl, ByRef returnedVal)
    Select Case control.id
        'Attribue la taille 'large' au premier bouton
        Case "bouton01": returnedVal = 1
        'Attribue la taille 'normal' au deuxième bouton
        Case "bouton02": returnedVal = 0
    End Select
End Sub
    
```

III-15 - getPressed

La fonction **getPressed** s'applique aux contrôles **checkBox**, **toggleButton**.

Elle permet de définir le statut du contrôle :

- * True (Lorsque le bouton bascule est activé ou que la case est cochée).
- * False (Lorsque le bouton bascule est désactivé ou que la case est décochée).

Cet exemple permet de paramétrer le contrôle 'checkBox' en vue de lui affecter la valeur True (Coché) lors de l'ouverture du classeur.

Xml

```

<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
  <ribbon startFromScratch="false">
    <tabs>
      <tab id="tab1" label="Test" >
        <group id="GR01" label="Essai de checkbox">
          <checkBox id="CheckBox1"
            label="Option"
            onAction="Validation"
            getPressed="MacroGetPressed" />
        </group>
      </tab>
    </tabs>
  </ribbon>
</customUI>
    
```

Dans un module standard du classeur :

Vba

```

Option Explicit

'Callback for CheckBox1 getPressed
'Définit le statut de la case à cocher
Sub MacroGetPressed(control As IRibbonControl, ByRef returnedVal)
    returnedVal = True
End Sub

'Callback for CheckBox1 onAction
'Renvoie la valeur de la case à cocher lorsque vous cliquez dessus
Sub Validation(control As IRibbonControl, pressed As Boolean)
    MsgBox pressed
End Sub
    
```

Vba

End Sub

III-16 - getItemCount

La fonction **getItemCount** s'applique aux contrôles **comboBox**, **dropDown**, **gallery**.

Elle définit le nombre d'éléments qui va être intégré dans le contrôle.

Consultez l'exemple proposé dans le chapitre **getLabel** (getItemCount="NbCaracteres"): la procédure récupère la taille de la **variable tableau** pour déterminer le nombre d'éléments que va contenir le contrôle gallery.

III-17 - getItemID

La fonction **getItemID** s'applique aux contrôles **comboBox**, **dropDown**, **gallery**.

Elle attribue un identificateur unique 'id' à chaque élément du contrôle.

Cet exemple crée 5 éléments dans un contrôle 'dropDown', qui vont être nommés 'strID1' à 'strID5'.

Xml

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
  <ribbon startFromScratch="false">
    <tabs>
      <tab id="TB01" label="Test">
        <group id="GR01" label="Essais getItemID">

          <dropDown id="DropDown01"
            getItemCount="MacroGetItemCount"
            getItemID="MacroGetItemID"
            getItemLabel="MacroGetItemLabel"
            onAction="LancementProcedure" />
        </group>
      </tab>
    </tabs>
  </ribbon>
</customUI>
```

Dans un module standard du classeur :

Vba

Option Explicit

```
'Callback for DropDown01 getItemCount
'Définit le nombre d'éléments dans le menu déroulant
Sub MacroGetItemCount(control As IRibbonControl, ByRef returnedVal)
    returnedVal = 5
End Sub

'Callback for DropDown01 getItemID
'Attribue un id à chaque élément
Sub MacroGetItemID(control As IRibbonControl, index As Integer, ByRef returnedVal)
    returnedVal = "strID" & index + 1
End Sub

'Callback for DropDown01 getItemLabel
'Attribue un label à chaque élément
Sub MacroGetItemLabel(control As IRibbonControl, index As Integer, ByRef returnedVal)
    returnedVal = "LeLabel " & index + 1
End Sub

'Callback for DropDown01 onAction
```

Vba

```
'Est déclenché lorsque vous sélectionné un élément dans le menu déroulant
Sub LancementProcedure(control As IRibbonControl, id As String, index As Integer)
MsgBox "Vous avez sélectionné l'id : " & id & " "
End Sub
```

III-18 - getItemImage

La fonction **getItemImage** s'applique aux contrôles **comboBox, dropDown, gallery**.

Elle définit l'image qui va être associée à un élément particulier du contrôle.

Cet exemple extrait les images .jpg contenues dans le répertoire spécial Windows "Mes images" (dont le chemin est généralement "C:\Documents and Settings\nom_utilisateur\Mes documents\Mes images"). Les images sont affichées dans la galerie sous forme de miniature. Chaque élément de la galerie possède une info-bulle qui contient des informations sur **les propriétés** des images listées.



Dans le fichier xml de personnalisation :

Xml

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui" >
  <ribbon startFromScratch="false">
    <tabs>
      <tab id="Tab01" label="Browser d'images" >
        <group id="Groupe01" label="Mes images">
          <gallery id="Galerie01"
            label="Affichez la prévisualisation d'images"
            imageMso="ObjectPictureFill"
            columns="3"
            onAction="ActionElementGalerie"
            getShowImage="MacroShowImage"
            getShowLabel="MacroShowLabel"
            getItemCount="MacroGetItemCount"
            getItemImage="MacroGetItemImage"
            getItemLabel="MacroGetItemLabel"
            getItemScreentip="MacroGetItemScreenTip"
            getItemSupertip="MacroGetItemSuperTip"
            getItemHeight="MacroGetItemHeight"
            getItemWidth="MacroGetItemWidth"
            getSelectedItemIndex="MacroSelItemIndex" >
          </gallery>
        </group>
      </tab>
    </tabs>
  </ribbon>
</customUI>
```

Dans le module objet "ThisWorkbook" du classeur :

Vba

```
Option Explicit
Option Compare Text
```

Vba

```

Dim x As Integer

Private Sub Workbook_Open()
    Const Cible = &H27 'définit le répertoire "Mes images"
    Dim objShell As Object, objFolder As Object
    Dim strFileName As Object
    Dim Resultat As String
    Dim i As Integer

    Set objShell = CreateObject("Shell.Application")
    Set objFolder = objShell.Namespace(Cible)

    'Récupère le chemin du répertoire Windows "Mes images"
    Chemin = objFolder.Path

    'boucle sur tous les fichiers du repertoire
    For Each strFileName In objFolder.Items
        'pour extraire uniquement les images type .jpg
        If strFileName.IsFolder = False And Right(strFileName, 4) = ".jpg" Then

            x = x + 1
            ReDim Preserve Tableau(1 To 2, 1 To x)
            'Enregistre le nom du fichier dans la premiere dimension du tableau
            Tableau(1, x) = strFileName
            Resultat = ""

            'Boucle sur les propriétés de chaque fichier:
            'ces informations seront affichées dans les info-bulles des éléments de
            'la galerie.
            For i = 0 To 34
                If objFolder.GetDetailsOf(strFileName, i) <> "" Then _
                    Resultat = Resultat & objFolder.GetDetailsOf(strFileName, i) & vbCrLf
            Next

            'Enregistre les propriétés dans la deuxième dimension du tableau
            Tableau(2, x) = Resultat
        End If
    Next

End Sub
    
```

Dans un module standard du classeur :

Vba

```

Option Explicit

Public Chemin As String
Public Tableau() As String

'Callback for Galerie01 onAction
'définit la procédure déclenchée lorsque vous
'cliquez sur un élément de la galerie.
Sub ActionElementGalerie(control As IRibbonControl, id As String, index As Integer)
    MsgBox "Vous avez cliqué sur l'image '" & Tableau(1, index + 1) & "'"
End Sub

'Callback for Galerie01 getShowImage
'Définit si l'image de la galerie doit être affichée
Sub MacroShowImage(control As IRibbonControl, ByRef returnedVal)
    returnedVal = True
End Sub

'Callback for Galerie01 getShowLabel
'Définit si le label de la galerie doit être affichée
    
```

Vba

```

Sub MacroShowLabel(control As IRibbonControl, ByRef returnedVal)
    returnedVal = True
End Sub

'Callback for Galerie01 getItemCount
'Définit le nombre d'éléments de la galerie
'(la taille maxi du tableau = nombre de fichiers .jpg)
Sub MacroGetItemCount(control As IRibbonControl, ByRef returnedVal)
    'La variable VarTab doit impérativement être de type Variant.
    Dim VarTab As Variant

    On Error Resume Next
    'VarTab va prendre la valeur Empty si le tableau est vide.
    VarTab = UBound(Tableau)
    On Error GoTo 0

    If IsEmpty(VarTab) Then
        returnedVal = 0
    Else
        returnedVal = UBound(Tableau, 2)
    End If
End Sub

'Callback for Galerie01 getItemImage
'Charge les images pour chaque élément de la galerie
Sub MacroGetItemImage(control As IRibbonControl, index As Integer, ByRef returnedVal)
    Set returnedVal = LoadPicture(Chemin & "\" & Tableau(1, index + 1))
End Sub

'Callback for Galerie01 getItemLabel
Sub MacroGetItemLabel(control As IRibbonControl, index As Integer, ByRef returnedVal)
    returnedVal = Tableau(1, index + 1)
End Sub

'Callback for Galerie01 getItemScreenTip
Sub MacroGetItemScreenTip(control As IRibbonControl, index As Integer, ByRef returnedVal)
    returnedVal = "Propriétés de '" & Tableau(1, index + 1) & "'"
End Sub

'Callback for Galerie01 getItemSuperTip
'Ajoute des informations sur les propriétés de l'image dans chaque infobulle
Sub MacroGetItemSuperTip(control As IRibbonControl, index As Integer, ByRef returnedVal)
    returnedVal = Tableau(2, index + 1)
End Sub

'Callback for Galerie01 getItemHeight
'Définit la hauteur de l'image qui va s'afficher dans la galerie
Sub MacroGetItemHeight(control As IRibbonControl, ByRef returnedVal)
    returnedVal = 50
End Sub

'Callback for Galerie01 getItemWidth
'Définit la largeur de l'image qui va s'afficher dans la galerie
Sub MacroGetItemWidth(control As IRibbonControl, ByRef returnedVal)
    returnedVal = 50
End Sub

'Callback for Galerie01 getSelectedItemIndex
'Permet de définir un élément de la galerie (à partir de son index) qui sera sélectionné
'par défaut lors de l'ouverture.
Sub MacroSelItemIndex(control As IRibbonControl, ByRef returnedVal)
    '0 = le premier élément de la galerie
    returnedVal = 0

```

Vba

End Sub

Cet exemple présuppose bien entendu que le nombre d'images ne soit pas trop important et que leur taille en ko soit raisonnable ... ;o)

III-19 - getItemLabel

La fonction **getItemLabel** s'applique aux contrôles **comboBox**, **dropDown**, **gallery**. Elle définit la chaîne de caractères qui va s'afficher pour un élément spécifique du contrôle. Consultez les exemples proposés dans les chapitres **getLabel** (getItemLabel="LabelCaractere") et **getItemImage** (getItemLabel="MacroGetItemLabel") : les procédures récupèrent le contenu de la variable tableau afin de créer une étiquette pour chaque élément du contrôle gallery.

III-20 - getItemScreentip

La fonction **getItemScreentip** s'applique aux contrôles **comboBox**, **dropDown**, **gallery**. Elle définit une info-bulle pour un élément particulier du contrôle. Consultez l'exemple proposé dans le chapitre **getItemImage** (getItemScreentip="MacroGetItemScreenTip"). Le code permet d'afficher le nom du fichier associé à l'image.

III-21 - getItemSupertip

La fonction **getItemSupertip** s'applique aux contrôles **comboBox**, **dropDown**, **gallery**. Elle définit une info-bulle complémentaire pour un élément particulier du contrôle. Les données s'affichent à la suite des informations indiquées dans l'attribut "screentip". Consultez l'exemple proposé dans le chapitre **getItemImage** (getItemSupertip="MacroGetItemSuperTip"). Le code permet d'afficher les propriétés des fichiers (Date de création, modification, taille en ko, dimensions en pixels ...etc...) lorsque vous passez le curseur de la souris sur les images.

III-22 - getText

La fonction **getText** s'applique aux contrôles **comboBox**, **editBox**. Elle spécifie une donnée par défaut pour le contrôle. Cet exemple affiche la valeur '1234' dans le contrôle 'editBox' dès l'activation de l'onglet nommé 'OngletPerso'. Dans le fichier xml de personnalisation :

Xml

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
  <ribbon startFromScratch="false">
    <tabs>
      <tab id="OngletPerso" label="Test">
        <group id="GR01" label="Exemple utilisation editBox">

<box id="Box01" boxStyle="horizontal">
  <editBox id="editBox01"
    label="Saisissez le code :"
    onChange="RecupDonnee"
    getText="MacroGetText" />
</box>

<button id="button01"
  onAction="ValidationCode"
  label="Validez le code"
  imageMso="FileServerTasksMenu"
  size="normal" />
```

Xml

```

        </group>
    </tab>
</tabs>
</ribbon>
</customUI>

```

Dans un module standard du classeur :

Vba

```

Option Explicit

Dim Cible As String

'Callback for editBox01 getText
'Spécifie une donnée par défaut pour le contrôle editBox
Sub MacroGetText(control As IRibbonControl, ByRef returnedVal)
    Cible = 1234
    returnedVal = Cible
End Sub

'Callback for editBox01 onChange
'Récupère le contenu du contrôle editBox dans la variable 'Cible'
Sub RecupDonnee(control As IRibbonControl, text As String)
    Cible = text
End Sub

'Callback for button01 onAction
Sub ValidationCode(control As IRibbonControl)
    MsgBox Cible
End Sub

```

III-23 - onChange

La fonction **onChange** s'applique aux contrôles **comboBox**, **editBox**.

Elle est déclenchée lorsqu'une modification est apportée dans la zone de saisie du contrôle et dès que celui perd le focus. A noter que l'événement n'est pas activé si vous remplacez des caractères à l'identique. Consultez l'exemple proposé dans le chapitre **getText** (onChange="RecupDonnee").

III-24 - getSelectedItemID

La fonction **getSelectedItemID** s'applique aux contrôles **dropDown**, **gallery**.

Elle définit un élément de la galerie (à partir de son identificateur unique 'ID') qui sera sélectionné par défaut lors de l'ouverture.

Remarque :

Les fonctions **getSelectedItemIndex** et **getSelectedItemID** ne peuvent pas être utilisées ensemble, pour un même contrôle. Lors de l'ouverture du classeur vous aurez un message d'erreur : "Les attributs s'excluant mutuellement 'getSelectedItemID' et 'getSelectedItemIndex' sont spécifiés."

Ce code permet de définir l'item nommé 'lettre02' comme élément sélectionné par défaut, dans le contrôle 'gallery'. Dans le fichier xml de personnalisation :

Xml

```

<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui" >
    <ribbon startFromScratch="false">
        <tabs>
            <tab id="Tab01" label="Onglet perso" >

```

Xml

```

<group id="Groupe01" label="Essai">

    <gallery id="Galerie01"
label="Test"
columns="3"
imageMso="InkHighlighter"
        showImage="true"
        showLabel="true"
getSelectedItemID="MacroSelItemID" >
    <item id="lettre01" label="A" />
    <item id="lettre02" label="B" />
    <item id="lettre03" label="C" />
    <item id="lettre04" label="D" />
    <item id="lettre05" label="E" />
    <item id="lettre06" label="F" />

</gallery>
</group>
</tab>
</tabs>
</ribbon>
</customUI>

```

Dans un module standard du classeur :

Vba

```

'Callback for Galerie01 getItemID
'Permet de définir un élément de la galerie (à partir de son identificateur unique 'id')
'qui sera sélectionné par défaut lors de l'ouverture.
Sub MacroSelItemID(control As IRibbonControl, ByRef returnedVal)
    returnedVal = "lettre02"
End Sub

```

III-25 - getItemIndex

La fonction **getItemIndex** s'applique aux contrôles **dropDown**, **gallery**.

Elle définit un élément de la galerie (à partir de son index) qui sera sélectionné par défaut lors de l'ouverture.

Remarque :

Les fonctions getItemIndex et getItemID ne peuvent pas être utilisés ensemble, pour un même contrôle. Lors de l'ouverture du classeur vous aurez un message d'erreur : "Les attributs s'excluant mutuellement 'getItemID' et 'getItemIndex' sont spécifiés.

Consultez l'exemple proposé dans le chapitre **getItemImage** (getItemIndex="MacroSelItemIndex").

III-26 - getContent

La fonction **getContent** s'applique au contrôle **dynamicMenu**.

Elle déclenche une procédure VBA qui va définir le code xml de personnalisation à la volée, pour le contrôle dynamicMenu.

Le code suivant, placé dans le fichier xml, crée un menu dynamique qui liste les feuilles de calcul et les feuilles graphiques du classeur actif. Chaque type de feuille est regroupé dans un menu de séparation (contrôle menuSeparator). L'élément sélectionné active la feuille correspondante.



Dans le fichier xml de personnalisation :

Xml

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
<ribbon startFromScratch="false">

  <tabs>
    <tab id="Ongletperso" label="Liste feuilles">

      <group id="GR01" label="Test">

        <dynamicMenu id="ListeDynamique"
          label="Liste feuilles"
          getContent="CreationMenuDynamique"
          invalidateContentOnDrop="true"
          size="normal"
          imageMso="PrintTitles"/>
      </group>

    </tab>
  </tabs>

</ribbon>
</customUI>
```

Ensuite, placez ces procédures dans un module standard du classeur.

Vba

Option Explicit

```
'Callback for ListeDynamique getContent
'Procédure pour construire le menu dynamique
Public Sub CreationMenuDynamique(ctl As IRibbonControl, ByRef content)
  'ouverture de la balise menu
  content = "<menu xmlns="http://schemas.microsoft.com/office/2006/01/customui">"

  'liste les feuilles de calcul du classeur actif
  content = content & ListeFeuilles(ActiveWorkbook)

  'liste les feuilles graphiques du classeur actif
  content = content & ListeCharts(ActiveWorkbook)

  'fermeture de la balise
  content = content & "</menu>"
End Sub
```

```
Private Function ListeFeuilles(Wb As Workbook) As String
  Dim strTemp As String
  Dim Ws As Worksheet

  ' Insertion d'un titre de menu
  strTemp = "<menuSeparator id=""Feuilles"" title=""Feuilles""/>"

  ' ajoute un bouton dans le menu pour chaque feuille du classeur
  For Each Ws In Wb.Worksheets
    strTemp = strTemp & _
      "<button " & _
      CreationAttribut("id", "Bt" & Ws.Name) & " " & _
      CreationAttribut("label", Ws.Name) & " " & _
      CreationAttribut("tag", Ws.Name) & " " & _
      CreationAttribut("onAction", "ActivationFeuille") & ">"

  Next

  ListeFeuilles = strTemp
End Function
```

Vba

```
'Liste les feuilles graphiques contenues dans le classeur
Private Function ListeCharts(Wb As Workbook) As String
    Dim strTemp As String
    Dim Ch As Chart

    If Wb.Charts.Count = 0 Then Exit Function

    strTemp = "<menuSeparator id=""charts"" title=""charts""/>"

    For Each Ch In Wb.Charts
        strTemp = strTemp & _
            "<button " & _
            CreationAttribut("id", "Bt" & Ch.Name) & " " & _
            CreationAttribut("label", Ch.Name) & " " & _
            CreationAttribut("tag", Ch.Name) & " " & _
            CreationAttribut("onAction", "ActivationFeuille") & "/>"
    Next

    ListeCharts = strTemp
End Function

Function CreationAttribut(strAttribut As String, Donnee As String) As String
    CreationAttribut = strAttribut & "=" & Chr(34) & Donnee & Chr(34)
End Function

'Active la feuille sélectionnée lorsque vous cliquez sur un nom
'dans le menu.
Sub ActivationFeuille(control As IRibbonControl)
    Sheets(control.Tag).Activate
End Sub
```

III-27 - getItemHeight

La fonction **getItemHeight** s'applique au contrôle **gallery**.

Elle définit la hauteur (en pixels) de chaque élément qui va s'afficher dans la galerie.

Remarque :

Vous ne pouvez spécifier qu'une seule dimension pour l'ensemble des éléments d'un contrôle.

Consultez l'exemple proposé dans le chapitre **getItemImage** (getItemHeight="MacroGetItemHeight").

III-28 - getItemWidth

La fonction **getItemWidth** s'applique au contrôle **gallery**.

Elle définit la largeur (en pixels) de chaque élément qui va s'afficher dans la galerie.

Remarque :

Vous ne pouvez spécifier qu'une seule dimension pour l'ensemble des éléments d'un contrôle.

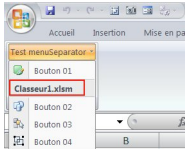
Consultez l'exemple proposé dans le chapitre **getItemImage** (getItemWidth="MacroGetItemWidth").

III-29 - getTitle

La fonction **getTitle** s'applique au contrôle **menuSeparator**.

Elle définit un titre pour le contrôle.

L'exemple suivant insère une séparation entre le premier et le deuxième bouton dans un contrôle menu. Le nom du classeur est récupéré dans la fonction d'appel 'getTitle' et apparaît dans le titre du contrôle 'menuSeparator'.



Placez ce code dans le fichier xml de personnalisation :

Xml

```

<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
<ribbon startFromScratch="false">
  <tabs>
    <tab id="TB01" label="Test">
      <group id="GR01" label="Essais">
        <menu id="Menu01" label="Test menuSeparator" itemSize="normal">
          <button id="bouton01" imageMso="ShapeStylesGallery" label="Bouton 01" />
          <menuSeparator id="Separator01"
            getTitle="MacroTitreMenuSeparator" />
          <button id="bouton02" imageMso="ShapeChangeShapeGallery" label="Bouton 02" />
          <button id="bouton03" imageMso="SelectionPane" label="Bouton 03" />
          <button id="bouton04" imageMso="ObjectsGroup" label="Bouton 04" />
        </menu>
      </group>
    </tab>
  </tabs>
</ribbon>
</customUI>

```

Dans un module standard du classeur :

Vba

```

'Callback for Separator01 getTitle
'indique le titre du menu séparateur
Sub MacroTitreMenuSeparator(control As IRibbonControl, ByRef returnedVal)
  returnedVal = ThisWorkbook.Name
End Sub

```

IV - Liens

Création de rubans personnalisés sous Microsoft Access 2007 , par Christophe Warin.

Customizing the 2007 Office Fluent Ribbon for Developers

Custom UI Editor Tool

La FAQ Access

La FAQ Excel

Le XML dans Microsoft Office (OpenXML), par Olivier Lebeau.

2007 Office System Document: Lists of Control IDs

La personnalisation du ruban sous Excel 2007

V - Remerciements

Je remercie toute l'équipe Office de DVP et particulièrement **Olivier Lebeau**, **Argyronet**, **Ouskelnor**, pour la relecture et la correction du tutoriel.

VI - Téléchargement