Créer des boites de dialogue avec VBA

Introduction

Le langage VBA permet d'afficher cinq sortes de boîtes de dialogue que vous pourrez parfois utiliser à la place des boîtes de dialogue personnalisées. Elles sont plus ou moins modifiables, mais sans offrir toutes les possibilités d'une véritable boîte de dialogue personnalisée.

- La fonction MsgBox.
- La fonction InputBox.
- La méthode GetOpenFilename.
- La méthode GetSaveAsFilename.
- La méthode FileDialog.

La fonction MsgBox

Afficher une boîte de message simple

Une fonction MsgBox est utilisable de deux manières :

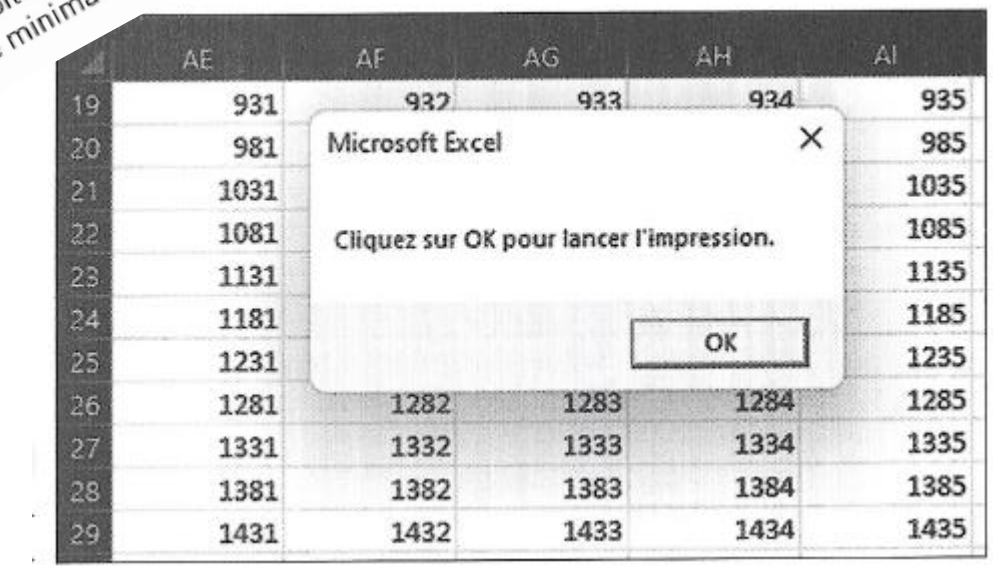
» **Pour afficher un message à destination de l'utilisateur : Dans ce cas, vous ne vous souciez pas du résultat renvoyé par la fonction.**

» **Obtenir une réponse de la part de l'utilisateur : Dans ce cas, vous tenez compte du résultat retourné par la fonction. Il dépend du bouton sur lequel l'utilisateur a cliqué.**

Si vous désirez utiliser cette fonction pour elle-même, ne mettez pas ses arguments entre parenthèses. L'exemple suivant se contente d'afficher un message sans retourner de résultat. Dès que le message est affiché, le code s'arrête jusqu'à ce que l'utilisateur ait cliqué sur OK :

```
Sub DémoMsgBox()
    MsgBox "Cliquez sur OK pour lancer l'impression."
    Sheets("Résultats").PrintOut
End Sub
```

Une boîte de mes-
sage minimaliste.



l'impression commence une fois que l'utilisateur a cliqué sur OK. Mais rien n'est prévu pour annu ler cette opération.

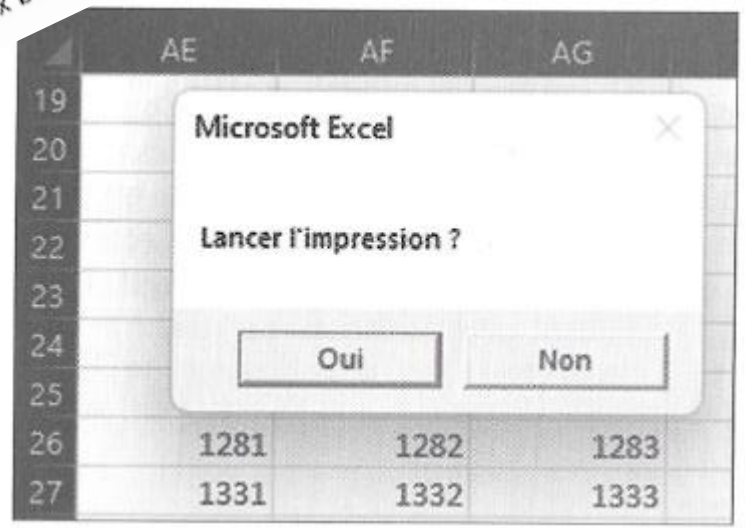
Obtenir une réponse d'une boîte de message

Lorsque le message qui devra être affiché contient plus de boutons qu'un simple bouton OK, le programme VBA doit savoir sur lequel l'utilisateur a cliqué. La fonction MsgBox est heureusement capable de retrouver une valeur associée au bouton qui a été cliqué. Le résultat de la fonction MsgBox peut être affecté à une variable.

Dans le code suivant, quelques constantes prédéfinies facilitent le travail à partir des valeurs retournées par MsgBox :

```
Sub ObtenirUneRéponse()
    Dim Réponse As Integer
    Réponse = MsgBox("Lancer l'impression ?", vbYesNo)
    Select Case Réponse
        Case vbYes
'       ...[code si la réponse est Oui]...
        Case vbNo
'       ...[ code si la réponse est Oui]...
    End Select
End Sub
```

Une boîte de message simple, avec deux boutons.



La Figure montre le résultat. Quand vous exécutez cette procédure, la variable Réponse reçoit la valeur produite, soit vbYes ou vbNo selon le bouton qui a été cliqué. L'instruction Select Case utilise la valeur Réponse pour déterminer l'action que la routine doit accomplir.

Les constantes de la fonction MsgBox.

Constante	Valeur	Description
vbOKOnly	0	N'affiche que le bouton OK.
vbOKCancel	1	Affiche les boutons de commande OK et Annuler.
vbAbortRetryIgnore	2	Affiche les boutons de commande Abandonner, Recommencer et Ignorer.
vbYesNoCancel	3	Affiche les boutons de commande Oui, Non et Annuler.
vbYesNo	4	Affiche les boutons de commande Oui et Non.
vbRetryCancel	5	Affiche les boutons de commande Recommencer et Annuler.
vbCritical	16	Affiche l'icône de message critique (cercle rond avec le X blanc) et émet le son associé.
vbQuestion	32	Affiche l'icône de question (phylactère bleu avec un point d'interrogation) et émet le son associé.
vbExclamation	48	Affiche l'icône d'alerte (triangle jaune avec le point d'exclamation) et émet le son associé.
vbInformation	64	Affiche l'icône d'information (phylactère bleu avec un i) et émet le son associé.
vbDefaultButton1	0	Le premier bouton est le bouton par défaut.
vbDefaultButton2	256	Le deuxième bouton est le bouton par défaut.
vbDefaultButton3	512	Le troisième bouton est le bouton par défaut.
vbDefaultButton4	768	Le quatrième bouton est le bouton par défaut.

Vous pouvez aussi utiliser le résultat de la fonction MsgBox sans passer par une variable, comme le démontre la variante suivante :

```
Sub ObtenirUneRéponse2()
    If MsgBox("Lancer l'impression ?", vbYesNo) = vbYes Then
'       ...[code si clic sur Oui]...
    Else
'       ...[ code si pas de clic sur Oui]...
    End If
End Sub
```

Personnaliser les boîtes de message

La souplesse de l'argument Boutons facilite la personnalisation des boîtes de message. Vous pouvez en effet spécifier quels boutons doivent être affichés, décider si une icône doit apparaître et également quel sera le bouton par défaut (celui qui sera « cliqué » si l'utilisateur appuie sur la touche Entrée).

Pour utiliser plusieurs de ces constantes comme argument, connectez-les simplement avec un opérateur + (plus). Par exemple, pour afficher une boîte de message avec seulement les boutons Oui et Non et l'icône d'exclamation, utilisez l'expression suivante comme deuxième argument MsgBox :

```
vbYesNo + vbExclamation
```

L'exemple suivant utilise une combinaison de constantes pour afficher une boîte de message avec des boutons Oui et Non (vbYesNo) ainsi qu'une icône de questionnement (vbQuestion). La constante vbDefaultButton2 désigne le deuxième bouton (Non) comme bouton par défaut, c'est-à-dire celui qui est pris en compte en appuyant sur la touche Entrée. Par souci de simplicité, j'affecte ces constantes à la variable Config, qui est ensuite utilisée comme second argument de la fonction MsgBox :

```
Sub ObtenirUneRéponse3()
    Dim Config As Long
    Dim Réponse As Integer
                                        ltButton2
    Réponse = MsgBox("Traiter le rapport mensuel ?", Config)
    If Réponse = vbYes Then LancerRapport
End Sub
```

```
Sub ObtenirUneRéponse3()
    Dim Config As Long
    Dim Réponse As Integer
    Config = vbYesNo + vbQuestion + vbDefaultButton2
    Réponse = MsgBox("Traiter le rapport mensuel ?", Config)
    If Réponse = vbYes Then LancerRapport
End Sub
```

L'argument Bouton de la fonction MsgBox définit le bouton par défaut.

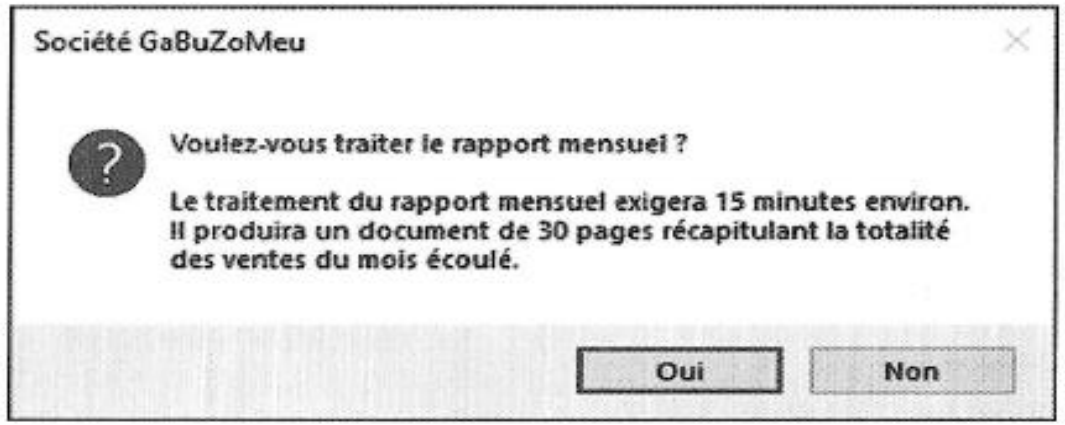
La Figure montre la boîte de message affichée par Excel lorsque la procédure ObtenirUneRéponse3 est exécutée. Si l'utilisateur clique sur le bouton Oui, la routine exécute la procédure nommée LancerRapport (elle n'est pas mentionnée dans le listing). S'il clique sur le bouton Non, ou s'il appuie sur Entrée, la routine s'achève sans qu'aucune action soit entreprise. Comme l'argument Titre a été omis dans la fonction MsgBox, Excel affiche le titre par défaut : Microsoft Excel.

un autre exemple de fonction MsgBox :

```
Sub ObtenirUneRéponse4()
    Dim Msg As String, Title As String
    Dim Config As Integer, Réponse As Integer
    Msg = "Voulez-vous traiter le rapport mensuel ?"
    Msg = Msg & vbNewLine & vbNewLine
    Msg = Msg & "Le traitement du rapport mensuel "
    Msg = Msg & "exigera 15 minutes environ. Il "
    Msg = Msg & "produira un document de 30 pages "
    Msg = Msg & "récapitulant la totalité des "
    Msg = Msg & "ventes du mois écoulé."
    Title = "Société GaBuZoMeu"
    Config = vbYesNo + vbQuestion
    Réponse = MsgBox(Msg, Config, Title)
    If Réponse = vbYes Then LancerRapport
End Sub
```

Cet exemple montre comment placer efficacement un texte long dans une boîte de message. Une variable (Msg) et un opérateur de concaténation (&) ont été utilisés pour construire le message par une succession d'instructions. La constante vbNewLine insère un retour à la ligne (il faut donc l'utiliser deux fois pour produire une ligne vierge). Le nom de la société a été placé dans la barre de titre grâce à l'argument de titre de MsgBox.

Une boîte de dialogue affichée par la fonction MsgBox, avec un titre, une icône et deux boutons.




```
Sub ObtenirUneRéponse4()
    Dim Msg As String, Title As String
    Dim Config As Integer, Réponse As Integer
    Msg = "Voulez-vous traiter le rapport mensuel ?"
    Msg = Msg & vbNewLine & vbNewLine
    Msg = Msg & "Le traitement du rapport mensuel "
    Msg = Msg & "exigera 15 minutes environ. Il "
    Msg = Msg & "produira un document de 30 pages "
    Msg = Msg & "récapitulant la totalité des "
    Msg = Msg & "ventes du mois écoulé."
    Title = "Société GaBuZoMeu"
    Config = vbYesNo + vbQuestion
    Réponse = MsgBox(Msg, Config, Title)
    If Réponse = vbYes Then LancerRapport
End Sub
```

Les constantes utilisées comme valeurs de retour par la fonction MsgBox.

Constante	Valeur	Signification
vbOK	1	L'utilisateur a cliqué sur OK.
vbCancel	2	L'utilisateur a cliqué sur Annuler.
vbAbort	3	L'utilisateur a cliqué sur Abandonner.
vbRetry	4	L'utilisateur a cliqué sur Recommencer.
vbIgnore	5	L'utilisateur a cliqué sur Ignorer.
vbYes	6	L'utilisateur a cliqué sur Oui.
vbNo	7	L'utilisateur a cliqué sur Non.

La fonction InputBox

La fonction InputBox du langage VBA sert à obtenir une valeur unique saisie par l'utilisateur. Il peut s'agir d'un nombre, d'une chaîne de caractères, et même d'une plage d'adresses. C'est une excellente alternative au développement de boîtes de dialogue personnalisées (les objets UserForms) quand le but est de récupérer une seule valeur.

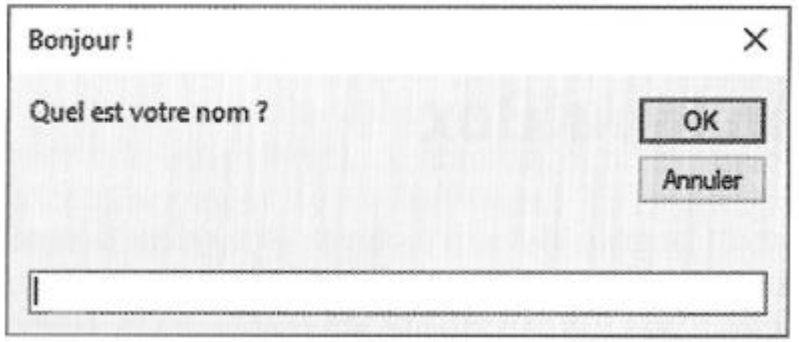
La syntaxe de InputBox

Voici une version simplifiée de la syntaxe de la fonction InputBox :

```
InputBox(invite[, titre][, parDéfaut])
```

Argument	Description
invite	Le texte affiché dans la boîte de saisie.
titre	Spécifie le texte affiché dans la barre de titre de la boîte de saisie (facultatif).
parDéfaut	Définit la valeur par défaut (facultatif).

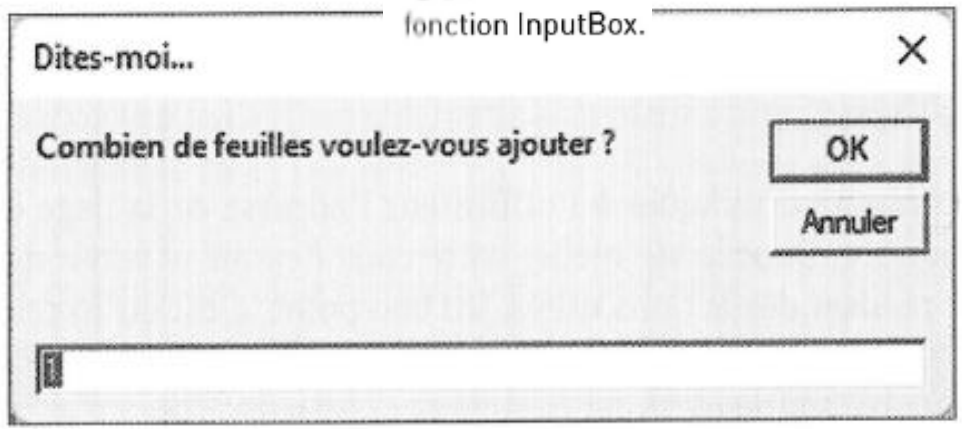
```
LeNom = InputBox("Quel est votre nom ?", "Bonjour !")
```



La fonction VBA InputBox retourne toujours une chaîne. Vous devrez au besoin la convertir en valeur numérique en effectuant les contrôles nécessaires. L'exemple qui suit fait appel à la fonction InputBox pour demander un nombre. Si la chaîne renvoyée contient effectivement une valeur numérique, tout continue normalement. Sinon, le code affiche un message d'erreur.

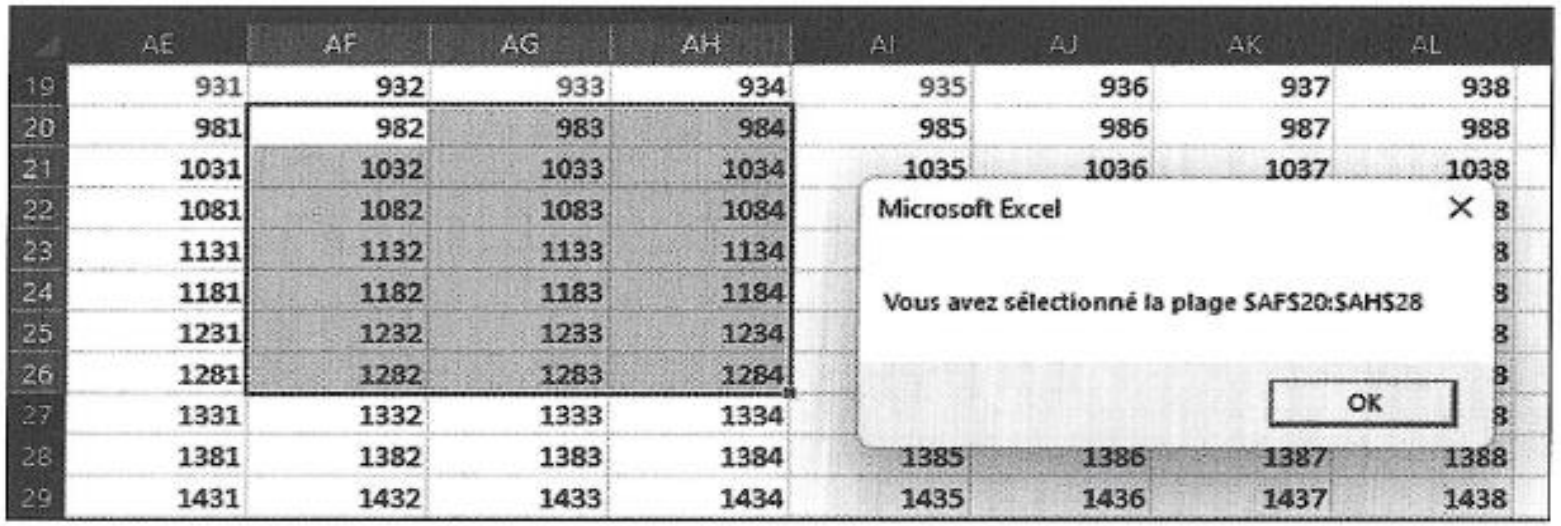
```
Sub AjouteFeuilles()
    Dim Invite As String
    Dim Titre As String
    Dim DefValeur As Integer
    Dim NombFeuilles As String

    Invite = "Combien de feuilles voulez-vous ajouter ?"
    Titre = "Dites-moi..."
    DefValeur = 1
    NombFeuilles = InputBox(Invite, Titre, DefValeur)

    If NombFeuilles = "" Then Exit Sub 'Annulation
    If IsNumeric(NombFeuilles) Then
        If NombFeuilles > 0 Then Sheets.Add Count:=NombFeuilles
    Else
        MsgBox "Entrez un nombre valide !"
    End If
End Sub
```

Un autre exemple d'utilisation de la fonction InputBox.



L'un des grands avantages de la méthode Application InputBox est la possibilité de demander à l'utilisateur de sélectionner une plage au clavier ou avec la souris. Voici un bref exemple :

```
Sub LitPlage()
    Dim Rng As Range
    On Error Resume Next
    Set Rng = Application.InputBox _
      (prompt:="Spécifiez une plage:", Type:=8)
    If Rng Is Nothing Then Exit Sub
    MsgBox "Vous avez sélectionné la plage " & Rng.Address
End Sub
```



Utiliser la méthode InputBox de l'objet Application pour sélectionner une plage de cellules.

La fonction GetOpenFileName

Si une procédure VBA doit demander à l'utilisateur de spécifier un nom de fichier, vous *pourriez* le faire avec la fonction InputBox. Mais ce n'est généralement pas l'outil le plus approprié pour ce genre de tâche, car la plupart des utilisateurs ont des difficultés à se souvenir des chemins et des noms de répertoires, sans compter les risques d'erreurs de saisie.

Pour éviter ces problèmes, utilisez de préférence la méthode GetOpenFilename de l'objet Application. Elle garantit en effet que l'application obtiendra un nom de fichier valide, y compris son chemin complet. Cette méthode affiche la classique boîte de dialogue Ouvrir, c'est-à-dire celle qui apparaît en choisissant la commande Fichier > Ouvrir > Parcourir.

La méthode GetOpenFilename n'ouvre pas véritablement le fichier. Elle ne fait que retourner le nom du fichier sélectionné par l'utilisateur. Vous pourrez ensuite écrire du code qui fera ce que vous désirez à partir de ce nom de fichier.

Exemple d'utilisation de GetOpenFilename

L'argument filtrageFichiers sert à indiquer ce qui doit apparaître dans la liste Types de fichiers, en bas de la boîte de dialogue Ouvrir. Il est constitué d'une paire de chaînes pour le filtrage de fichiers, suivie par la spécification de celui-ci grâce à des caractères de substitution. Les deux éléments sont séparés par une virgule. Si l'argument est omis, l'argument suivant est utilisé par défaut :

```
Tous les fichiers (*.*), *.*
```

Remarquez que la chaîne est constituée de deux parties :

```
Tous les fichiers (*.*)
```

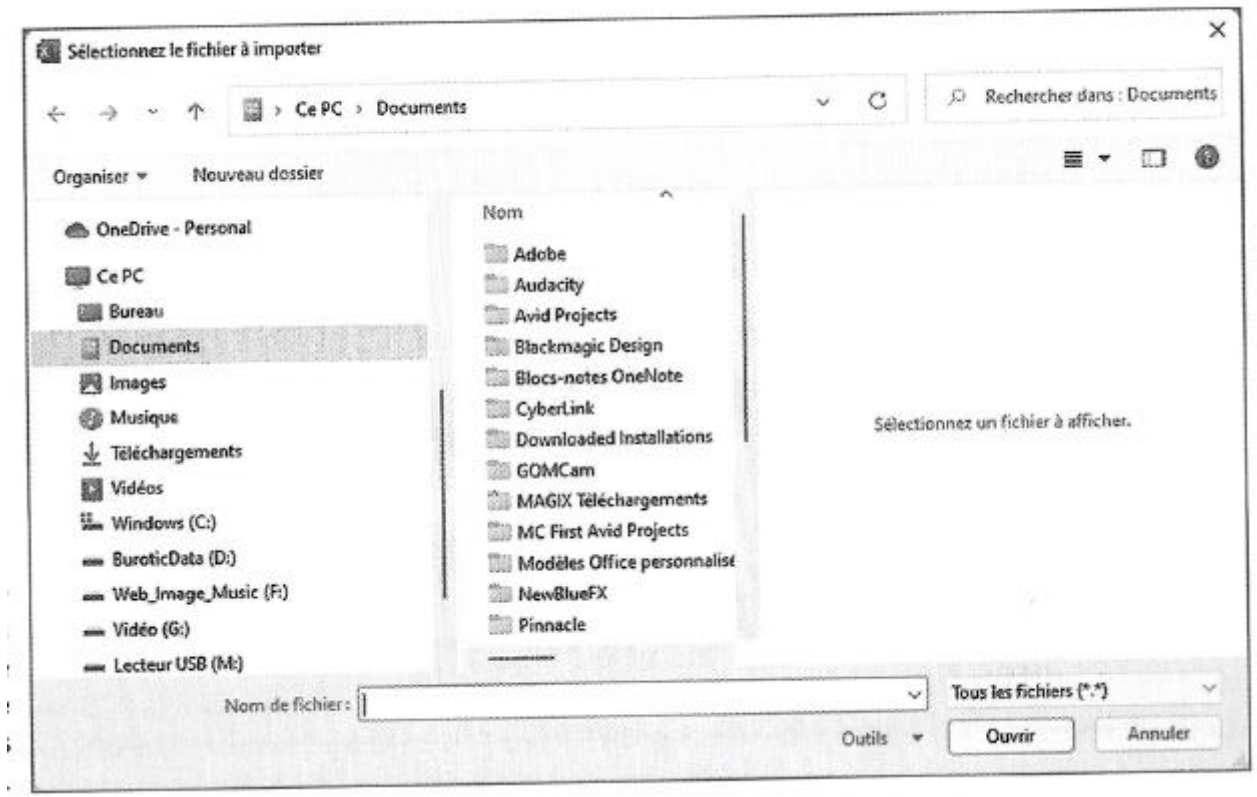
et

```
*.*
```

La première partie de cette chaîne est le texte affiché dans la liste déroulante Types de fichiers. La seconde partie définit quels sont les fichiers affichés dans la boîte de dialogue. Par exemple, *.* signifie *tous les fichiers.*

```
Sub RécupNomFichier()
    Dim typeFichier As String
    Dim filtreIndex As Integer
    Dim Titre As String
    Dim nomFichier As Variant

'   Etablissement de la liste des filtres de fichiers
    typeFichier = "Fichiers texte (*.txt),*.txt," & _
                  "Fichiers Lotus (*.prn),*.prn," & _
                  "Texte (séparateur : point-virgule) (*.csv),*.csv," & _
                  "Fichiers ASCII (*.asc),*.asc," & _
                  "Tous les fichiers (*.*),*.*"

'   Affichage par défaut = *.*
    filtreIndex = 5

'   Définition du message de la barre de titre
    Titre = "Sélectionnez le fichier à importer"

'   Récupération du nom de fichier
    nomFichier = Application.GetOpenFilename(typeFichier, _
       filtreIndex, Titre)

'   Gestion des données provenant de la boîte de dialogue

    If nomFichier = False Then
        MsgBox "Aucun fichier n'a été sélectionné."
    Else
        MsgBox "Vous avez sélectionné le fichier " & nomFichier
    End If
End Sub
```

La méthode GetOpenFilename affiche une boîte de dialogue personnalisée. Elle retourne aussi le nom du fichier sélectionné et son chemin. Elle n'ouvre toutefois pas le fichier.



Dans une application opérationnelle, vous feriez quelque chose de plus utile, comme ouvrir le fichier avec une instruction comme celle-ci :

```
Workbooks.Open nomFichier
```

La fonction GetSaveAsFileName

La méthode GetSaveAsFilename fonctionne comme la méthode GetOpenFile-name, sauf qu'elle affiche la boîte de dialogue Enregistrer sous d'Excel à la place d'Ouvrir. Elle récupère le chemin et le nom de fichier indiqués par l'utilisateur, mais n'en fait rien. Autrement dit, c'est à vous d'écrire le code qui sauvegarde effectivement le fichier.

Voici la syntaxe de cette méthode :

```
object.GetSaveAsFilename([nomFichierInitial], [filtrageFichiers],
        [indexFiltrage], [titre], [texteBouton])
```

Argument	Description
nomFichierInitial	Spécifie le nom par défaut qui apparaît dans le champ Nom de fichier.
filtrageFichiers	Définit les types de fichiers affichés par Excel, comme par exemple *.txt. Plusieurs filtres peuvent être utilisés afin de proposer un choix plus large à l'utilisateur.
indexFiltrage	Définit le filtre de fichier qu'Excel doit proposer par défaut.
titre	Contient le texte affiché dans la barre de titre de la boîte de dialogue.

La procédure qui suit affiche une boîte de dialogue qui permet à l'utilisateur de sélectionner un dossier. Le nom de ce dossier (ou le message « Annulé ») est ensuite affiché à l'aide de la fonction MsgBox :

```
Sub RecupDossier()
    With Application.FileDialog(msoFileDialogFolderPicker)
      .InitialFileName = Application.DefaultFilePath & "\"
      .Title = "Sélectionnez un emplacement pour la sauvegarde"
      .Show
      If .SelectedItems.Count = 0 Then
          MsgBox "Annulé"
      Else
          MsgBox .SelectedItems(1)
      End If
    End With
End Sub
```

L'objet FileDialog permet de spécifier le chemin d'accès initial en fournissant ce-lui-ci dans la propriété InitialFileName. En l'occurrence, ce code se sert de point de départ du chemin par défaut utilisé par Excel.

Afficher les boites pré définies d’Excel

Vous pouvez écrire du code VBA qui exécute des actions comme si elles avaient été sélectionnées dans le ruban ou dans une boîte de dialogue, sans même qu'Excel affiche quoi que ce soit de particulier.

Par exemple, l'instruction ci-dessus est équivalente au fait de choisir, dans le groupe Noms définis de l'onglet Formules, la commande Définir un nom pour afficher la boîte de dialogue Nouveau nom, puis de spécifier dans le champ Nom la valeur NomsMois, et enfin de cliquer sur OK :

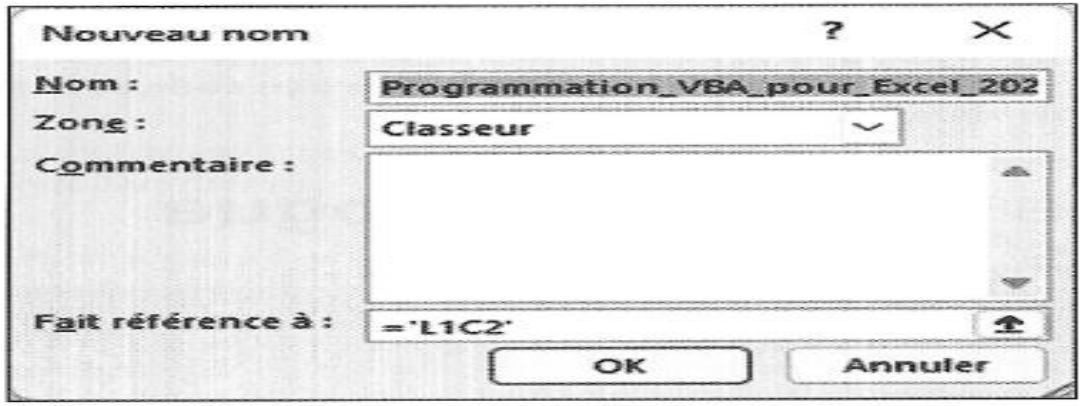
```
Range("A1:A12").Name = "NomsMois"
```

Quand vous exécutez cette instruction, la boîte de dialogue Nouveau nom n'apparaît pas. Et c'est pratiquement toujours ce que vous désirez. Il ne faudrait en effet pas qu'une boîte de dialogue surgisse tout d'un coup sur l'écran lors de l'exécution de la macro.

À la différence du précédent, l'exemple suivant affiche la boîte de dialogue Nouveau nom. La plage qui apparaît dans le champ fait référence à est celle qui était sélectionnée au moment où la commande est exécutée

Le code pour afficher cette boîte de dialogue est le suivant :

```
Sub NouvNom()
    Application.CommandBars.ExecuteMso "NameDefine"
End Sub
```



Afficher une boîte de dialogue d'Excel en utilisant VBA.

ExecuteMso est une méthode de l'objet CommandBars. Elle accepte un seul argument, qui est un paramètre idMso représentant un contrôle du ruban. Malheureusement, ces paramètres ne sont pas listés dans l'aide de VBA. Et comme le ruban est un concept relativement récent, le code qui fait appel à la méthode ExecuteMso n'est pas compatible avec les versions antérieures à Excel 2007.

Boite de dialogue personnalisée: UserForm

Créer une boîte de dialogue personnalisée : vue d'ensemble

Pour créer une boîte de dialogue personnalisée, vous suivrez généralement ces étapes :

1. **Détermination de la manière dont la boîte de dialogue sera utilisée et de l'endroit où elle apparaîtra dans la macro VBA.**

2. **Appui sur Alt+F11 afin d'activer l'éditeur VBE et d'insérer un nouvel objet UserForm.**

   Un objet UserForm ne contient qu'une seule boîte de dialogue personnalisée.

3. **Ajout de contrôles à l'objet UserForm.**

   Les contrôles sont notamment des boîtes de texte, des boutons, des cases à cocher et des zones de liste.

4. **Utilisation de la fenêtre Propriétés pour modifier les propriétés des contrôles ou l'objet UserForm lui-même.**

5. **Écriture de procédures de gestion des événements pour les contrôles (par exemple, une macro est exécutée lorsque l'utilisateur clique sur un bouton dans la boîte de dialogue).**

   Ces procédures sont stockées dans la fenêtre Code de l'objet UserForm.

6. **Écriture d'une procédure – stockée dans un module VBA – qui affiche la boîte de dialogue pour l'utilisateur.**

Travailler avec les objets UserForm

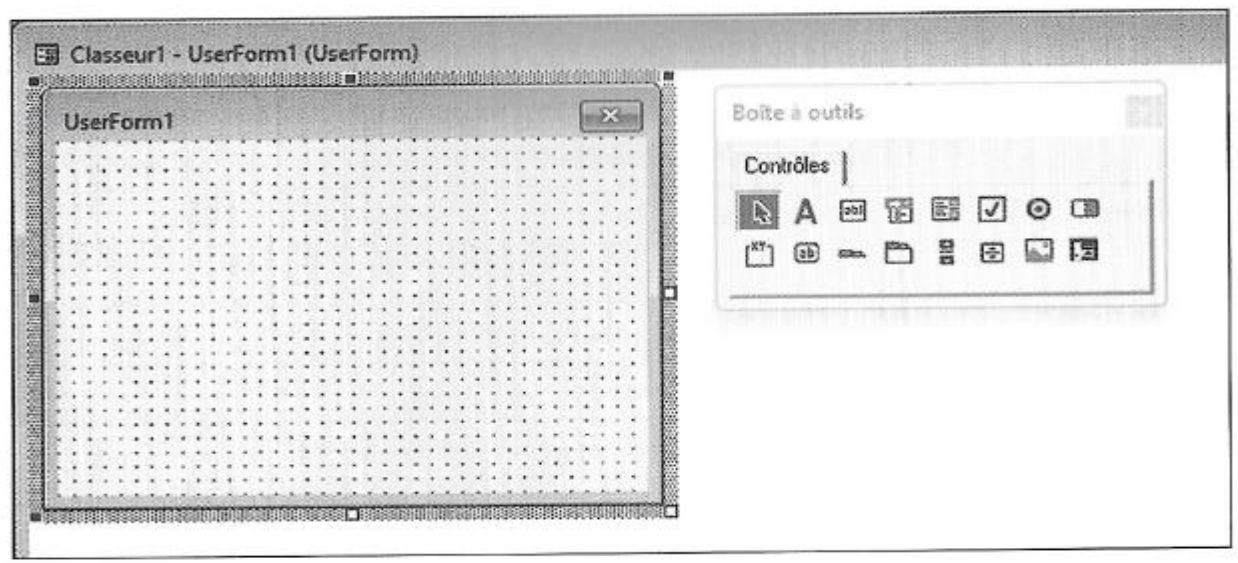
Chacune des boîtes de dialogue personnalisées que vous créez est stockée dans son propre objet UserForm (à raison d'une boîte par objet). Vous créez ces objets UserForm, et vous y accédez dans Visual Basic Editor.

Insérer un nouvel objet UserForm

Procédez comme suit pour insérer un objet UserForm :

1. **Activez l'éditeur VBE en appuyant sur Alt+F11.**
2. **Sélectionnez le classeur dans la fenêtre Projet.**
3. **Choisissez Insertion > UserForm.**

   L'éditeur VBE insère un nouvel objet UserForm contenant une boîte de dialogue vide.



Un nouvel objet UserForm.

outils que vous allez ajouter des contrôles à vos boîtes de dialogue personnalisées. Si vous ne voyez pas cette fenêtre, choisissez la commande Affichage > Boîte à outils.

Pour ajouter un contrôle, cliquez dessus puis faites-le glisser sur la boîte de dialogue pour l'insérer à l'intérieur de celle-ci. Vous pouvez ensuite le déplacer et le redimensionner à l'aide des techniques habituelles.

Les contrôles de la boîte à outils.

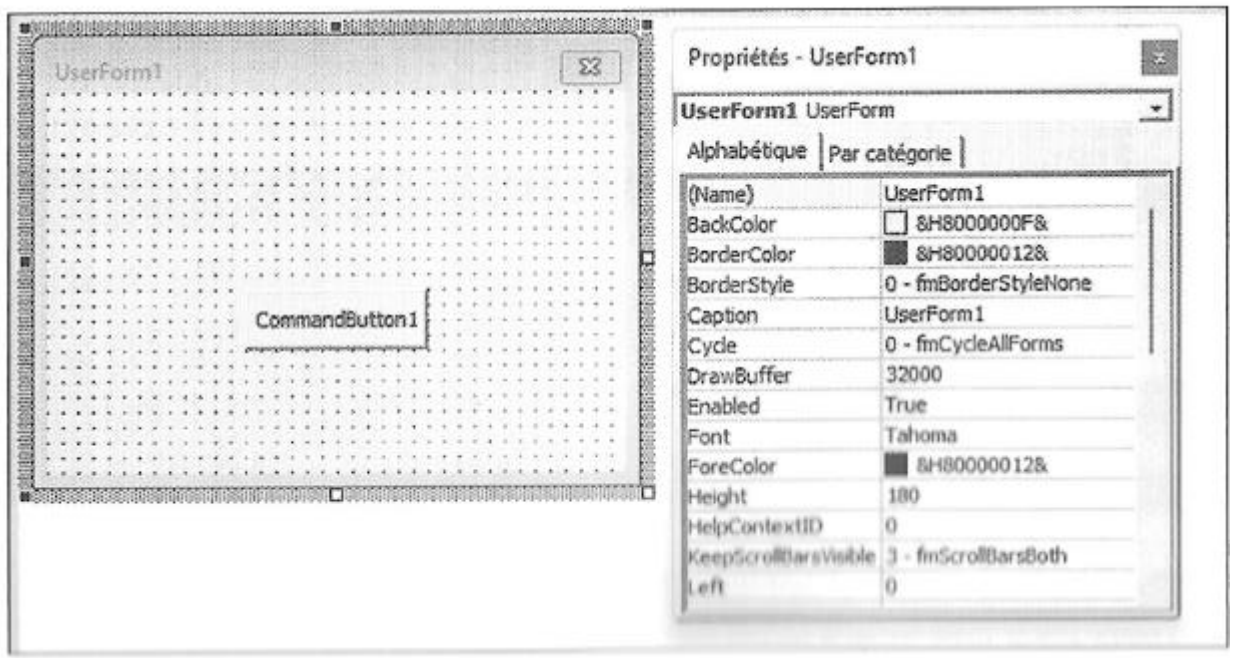
Contrôle	Description
Intitulé	Stocke du texte.
Zone de texte	Permet à l'utilisateur d'entrer du texte.
Zone de liste modifiable	Affiche une liste déroulante.
Zone de liste	Affiche une liste d'éléments.
Case à cocher	Commode pour les options actif/inactif et oui/non.
Bouton d'option	N'autorise le choix que d'une option parmi d'autres du groupe de boutons.

Contrôle	Description
Bouton bascule	Commutateur à deux positions.
Cadre	Contient d'autres contrôles.
Bouton de commande	Bouton cliquable.
Contrôle onglet	Onglets.
Multipage	Conteneur à onglets pour d'autres objets.
Barre de défilement	Barre avec un ascenseur.
Toupie	Bouton cliquable servant généralement à modifier une valeur.
Image	Contient une image.
RefEdit	Permet à l'utilisateur de sélectionner une plage.

Modifier les propriétés d'un contrôle UserForm

Chacun des contrôles d'un objet UserForm contient des propriétés qui déterminent son apparence ou son comportement. Ces propriétés sont modifiables.

Utilisez la fenêtre Propriétés pour modifier les propriétés d'un contrôle UserForm.



La fenêtre Propriétés apparaît en appuyant sur F4. Les propriétés affichées dépendent évidemment de ce qui a été sélectionné et changent par exemple d'un contrôle à un autre. Pour la masquer, cliquez sur sa case de fermeture. La touche F4 la ramènera à la vie lorsque vous en aurez à nouveau besoin.

Voici quelques propriétés de contrôles, suivies de leur traduction :

- **Name (nom)**
- **Width (largeur)**
- **Height (hauteur)**
- **Value (valeur)**
- **Caption (légende)**

Pour modifier une propriété d'un contrôle, procédez comme suit :

1. **Commencez par sélectionner le contrôle voulu dans le cadre de l'objet UserForm.**
2. **Assurez-vous que la fenêtre Propriétés est visible (si besoin est, appuyez sur F4).**
3. **Dans la fenêtre Propriétés, cliquez sur la ligne qui définit la propriété que vous voulez éditer.**
4. **Apportez les changements voulus dans la partie droite de la fenêtre Propriétés.**

Si vous sélectionnez l'objet UserForm lui-même (et non un de ses contrôles), la fenêtre Propriétés affiche les propriétés de la boîte de dialogue proprement dite.

Utiliser les informations fournies par une boîte de dialogue personnalisée

L'éditeur VBE attribue un nom à chacun des contrôles que vous ajoutez à un objet UserForm. Ce nom est celui qui est présent dans la propriété Name. Utilisez-le dans votre code pour faire référence à tel ou tel contrôle. Par exemple, si vous avez placé un contrôle Case à cocher dans un objet UserForm nommé UserForm1, par défaut le contrôle Case à cocher sera nommé CheckBox1. L'instruction ci-dessous fait apparaître le contrôle déjà coché :

```
UserForm1.CheckBox1.Value = True
```

La plupart du temps, votre code sera inséré dans le module de code l'objet UserForm lui-même. Vous pouvez alors omettre son nom et simplifier vos instructions, comme ici :

```
CheckBox1.Value = True
```

La fenêtre Code de l'objet UserForm

Chaque objet UserForm est doté d'une fenêtre Code qui contient le code VBA (les procédures d'événement) exécuté lorsque l'utilisateur interagit avec la boîte de dialogue. Appuyez sur F7 pour afficher la fenêtre Code. Elle reste vide tant que vous n'avez pas ajouté de procédures. Appuyez sur Maj+F7 pour revenir à la boîte de dialogue.

Un autre moyen de passer de la fenêtre Code à la fenêtre UserForm consiste à cliquer, dans la barre de titre de la fenêtre Projet, sur les boutons Afficher le code et Afficher l'objet. Il est également possible de cliquer du bouton droit sur l'objet UserForm et de choisir Code dans le menu qui apparaît. Faites alors un double-clic sur le nom de l'objet UserForm dans la fenêtre Projet pour le réafficher.

Afficher une boîte de dialogue personnalisée

Vous pouvez afficher une boîte de dialogue personnalisée en utilisant la méthode Show de l'objet UserForm, dans une procédure VBA.

La macro qui affiche la boîte de dialogue doit se trouver dans le module VBA, et non dans la fenêtre Code de l'objet UserForm.

La procédure suivante affiche la boîte de dialogue nommée UserForm1 :

```
Sub AfficherBoîteDialogue()
    UserForm1.Show
'   [Placez ici les autres instructions]
End Sub
```

Bibliographie

- Pour la mise en œuvre https://arkham46.developpez.com/articles/office/userformdialog/
- Programmation VBA pour Excel, J. Walkenbach