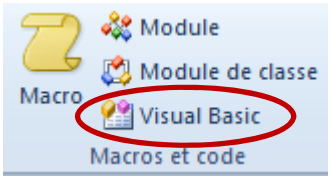
AUTOMATISER SES TÂCHES DANS ACCESS AVEC LE VBA

Le VBA — Visual Basic Application — est un langage permettant d'automatiser ses tâches dans les logiciels de la suite Microsoft Office en regroupant plusieurs commandes (des macros-commandes). Si de nombreuses fonctions sont communes à l'ensemble des logiciels — conditions logiques, boucles, etc. — d'autres sont spécifiques à certains logiciels — gestion des cellules avec Excel, déplacement dans un texte avec Word, accès à des champs avec Access, etc. —, voici, ci-dessous, quelques commandes spécifiques à Access.

ACCÈS AU VBA

Onglet Créer
Visual Basic

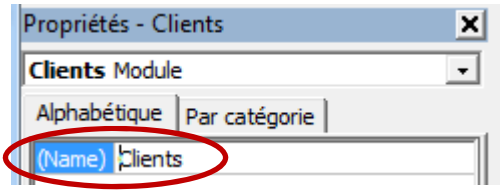
ou **Alt | F11**



CRÉER UNE COMMANDE EN VBA

CRÉATION D'UN MODULE

1. Un module va permettre de regrouper plusieurs macros en VBA
2. Menu Insertion
   Module

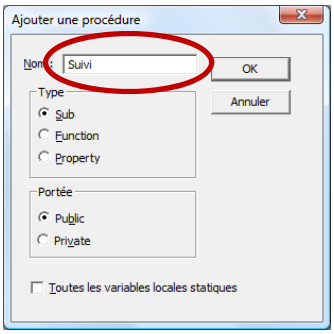


Pour renommer le module actif

1. Dans le volet de gauche, cliquer dans la case *(Name)*
2. Saisir un nom pour le module (exemple : Clients)

CRÉATION D'UNE PROCÉDURE VBA

La proécdure va contenir les différentes commandes de la macro en VBA

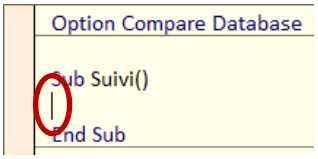


1. Menu Insertion
   Procédure
2. Saisir le nom de la procédure voulue (exemple : Suivi) sans espace
   Laisser *Sub* cocher et **OK**

✓ Il est possible de saisir directement au clavier
le titre de sa macro dans son module,
exemple : *Sub Suivi*, les parenthèses et le *End Sub* sont ajoutés automatiquement.

SASIE DES COMMANDES VBA

Entre le *Sub* et *End Sub*, saisir le code — l'ensemble des commandes — de sa macro :



- *Option Compare Database* est une commande facultative indiquant que les tris et les comparaisons des textes seront faits en fonction des paramètres régionaux de la base de données
- *Private* devant une procédure indique que celle-ci n'est accessible qu'à partir de son module,
  exemple : `Private Sub Suivi()` qui est la valeur par défaut
  Public indique que la procédure est accessible à partir de tous les modules,
  exemple : `Public Sub Suivi()`

QUELQUES EXEMPLES DE CODE VBA SUR LES FORMULAIRES

OUVRIR UN FORMULAIRE, UN ÉTAT, UNE REQUÊTE

FORMULAIRE

Ouverture du formulaire « Formation clients »

```
DoCmd.OpenForm "Formation clients", acNormal
```

Quelques paramètres du mode d'ouverture

- **acNormal** : mode Formulaire pour la saisie (ouverture par défaut)
- **acDesign** : mode Création
- **acLayout** : mode Page
- **acPreview** : mode Aperçu avant impression
- **acFormDS** : mode Feuille de données

Ouverture d'un formulaire avec un filtre sur le champ Ville

Seuls les formulaires et les états peuvent être ouverts avec un filtre

```
DoCmd.OpenForm "Formation clients", acNormal, , "[Ville]='Paris'"
```

✓ La condition est entre guillemets droits et Le critère de filtre est entre apostrophes
Après *acNormal*, il y a deux virgules, car cet emplacement permet d'appeler un filtre par son nom

ÉTAT

```
DoCmd.OpenReport "État Formation", acPreview
```

Quelques paramètres

- **acPreview** : aperçu avant impression
- **acNormal** : lance l'impression
- **acDesign** : mode Création

TABLE

```
DoCmd.OpenTable "Produits", acNormal
```

Quelques paramètres

- **acNormal** : pour pouvoir saisir des données dans la table
- **acReadOnly** : pour imprimer les données de la table
- **acDesign** : mode Création pour modifier les champs

REQUÊTE

```
DoCmd.OpenQuery "Requête Clients", acNormal
```

Quelques paramètres

Ceux-ci sont proches de ceux des tables :

- **acNormal** : pour pouvoir modifier les données dans la requête
- **acReadOnly** : pour imprimer le résultat de la requête
- **acDesign** : mode Création pour modifier les champs, les tris, les critères, etc. de la requête

Fermeture d'un formulaire

Fermeture du formulaire actif

```
DoCmd.Close
```

Fermeture d'un formulaire ouvert appelé *Formulaire Produits*

```
DoCmd.Close acForm, "Formulaire Produits", acSaveNo
```

Pour enregistre ou nom les modifications dans le formulaire

- acSaveYes : sauvegarde les modifications dans le formulaire (et non des données)
- acSaveNo : ne sauvegarde pas les modifications du formulaire

Atteindre un champ dans le formulaire actif

Atteindre un champ du formulaire principal

```
DoCmd.GoToControl "[Ville]"
```

Atteindre un champ dans un sous-formulaire

1. Accéder au formulaire principal avec `DoCmd.OpenForm...`
2. Accéder au sous-formulaire :
   `Forms![Formulaire principal]![Sous-formulaire].SetFocus`
3. Accéder au champ voulu :
   `Forms![Formulaire principal]![Sous-formulaire]![Champ].SetFocus`

Exemple pour atteindre le champ *Cours*, du sous-formulaire *Suivi des formations*, dans le formulaire *Formation clients*

```
' Ouverture du formulaire principale "Formation clients"
DoCmd.OpenForm "Formation clients", acNormal
' Accès au sous-formulaire "Suivi des formations"
Forms![Formation clients]![Suivi des formations].SetFocus
' Accès au champ "Cours"
Forms![Formation clients]![Suivi des formations]![Cours].SetFocus
```

Modifier la valeur d'un champ

Affecte la valeur *Facturation* dans le champ Suivi du sous-formulaire *SF Commande* dans le formulaire *Formulaire Clients* :

```
Forms![Formulaire clients]![SF Commande]![Suivi].Value = "Facturation"
```

Simplification des commandes d'appel de formulaire

Avec Me.

Si c'est le formulaire actif qui est concerné, il est possible de remplacer ses coordonnées par `Me.`

Exemple pour atteindre le sous formulaire *SF Commande*

À la place d'écrire

```
Forms![Formulaire clients]![SF Commande].SetFocus
```

Écrire

```
Me.[SF Commande].SetFocus
```

Exemple pour modifier le contenu du champ *Suivi* dans le sous formulaire *SF Commande*

À la place d'écrire

```
Forms![Formulaire clients]![SF Commande]![Suivi].Value = "Facturation"
```

Écrire

```
Me.[SF Commande].Form.[Suivi].Value = "Facturation"
```

AVEC WITH

Pour simplifier certaines commandes il est possible d'appliquer une série d'instructions à plusieurs objets avec les commandes

```
With...
      Liste des objets
End With
```

Exemple d'utilisation

Au lieu d'écrire plusieurs fois Me.[SF Commande].

```
Me.[SF Commande].SetFocus
Me.[SF Commande].Form.[Suivi].Value = "Facturation"
Me.[SF Commande].Form.[Date suivi].Value = Date + 15
Me.[SF Commande].Form.[SF Commande]![Montant HT].Value = _
    Int(Me.[SF Commande].Form.[SF Commande]![Montant HT].Value * 1.1)
Me.[SF Commande].Form.[SF Commande]![Quantité].SetFocus
```

Il est possible d'écrire

```
With Me.[SF Commande]
    .SetFocus
    .Form.[Suivi].Value = "Facturation"
    .Form.[Date suivi].Value = Date + 15
    .Form.[SF Commande]![Montant HT].Value = Int(.[Montant HT].Value * 1.1)
    .Form.[SF Commande]![Quantité].SetFocus
End With
```

GÉRER LES ENREGISTREMENTS

SE DÉPLACER ENTRE LES ENREGISTREMENTS

```
DoCmd.GoToRecord , , acFirst ' pour le premier enregistrement
```

Paramètres

`DoCmd.GoToRecord , ,`	`acFirst`	Premier enregistrement
	`acLast`	Dernier enregistrement
	`acPrevious`	Enregistrement précédent
	`acNext`	Enregistrement suivant
	`acNewRec`	Nouvel enregistrement

Atteindre un enregistrement précis

```
DoCmd.GoToRecord , , acGoTo, 7 ' atteint l'enregistrement n° 7
```

Connaître le numéro de l'enregistrement courant

```
MsgBox CurrentRecord
```

SE DÉPLACER DANS LES ENREGISTREMENTS D'UN SOUS-FORMULAIRE

```
Me.[SF Commande].Form.Recordset.MoveNext
```

✓ *Me* désigne le formulaire principal

`Me.[SF Commande].Form.Recordset.`	`MoveFirst`	Premier enregistrement
	`MoveLast`	Dernier enregistrement
	`MovePrevious`	Enregistrement précédent
	`MoveNext`	Enregistrement suivant

Accéder à un nouvel enregistrement du sous-formulaire

```
Me.[SF Commande].Form.Recordset.AddNew
```

Se déplacer entre les enregistrements

```
Me.[SF Commande].Form.Recordset.Move 3 ' avance de trois enregistrements
```

✓ Pour reculer, il faut mettre un nombre négatif, exemple :

```
Me.[SF Commande].Form.Recordset.Move -2 ' recule de deux enregistrements
```

Ajouter un enregistrement

```
Me.[SF Commande].Form.Recordset.AddNew
```

Connaître le numéro de l'enregistrement courant d'un sous-formulaire

Affiche le numéro de l'enregistrement actif dans le sous-formulaire « SF Commande » dans le formulaire principal « Formulaire Clients » :

```
MsgBox Forms![Formulaire clients]![SF Commande].Form.CurrentRecord
```

ACTUALISATION DES DONNÉES

AFFICHER OU MASQUER LES MISES À JOUR ET LES QUESTIONS À L'ÉCRAN

DÉSACTIVE OU DÉSACTIVE L'ACTUALISATION DE L'ÉCRAN

Désactiver l'actualisation de l'écran permet de gagner du temps dans son code. Toutefois, il ne faut pas oublier de la réactiver en fin de programme.

```
DoCmd.Echo False
      Commandes VBA
DoCmd.Echo True
```

POUR DÉSACTIVER OU ACTIVER L'AFFICHAGE DES CONFIRMATIONS D'ENREGISTREMENT

Cette commande évite, en cas de changement, d'avoir le message demandant confirmation des sauvegardes. Par contre, il faut penser à la réactiver en fin de programme.

```
DoCmd.SetWarnings False
      Commandes VBA
DoCmd.SetWarnings True
```

L'instruction avec *False* est à placer en début de programme pour désactiver la commande souhaitée et celle avec *True* est à placer en fin de programme pour réactiver la commande.

ACTUALISER DES DONNÉES APRÈS DES MODIFICATIONS

Après la modification de données, il se peut qu'il faille réactualiser tous les calculs qui en dépendent. Pour actualiser les données, placer à la fin du code l'une des instructions suivantes :

- **Refresh** : pour mettre à jour les données de la base de données (si celle-ci ont été modifiée dans le formulaire)
- **Requery** : pour mettre à jour les données d'un formulaire (par exemple, refaire un calcul dans un formulaire suite à la modification d'une donnée)

✓ Il est possible de préciser le formulaire ouvert à actualiser, exemple :

```
Forms![Formulaire Clients].Requery
```

PARCOURIR LES DONNÉES D'UNE TABLE OU D'UNE REQUÊTE

Pour pouvoir parcourir et modifier les objets d'une table ou d'une requête, Access VBA dispose d'instruction permettant d'utiliser des objets d'accès aux données — *Data Access Object* (DAO) en anglais) — dont voici, ci-dessous, un exemple d'utilisation. Cette méthode est plus rapide que de faire défiler les pages d'un formulaire.

Pour cela, il faudra faire appel aux objets *Recordset*. Un objet *Recordset* contient des données regroupées dans un tableau en ligne et en colonne. C'est une sorte de table provisoire permettant d'accéder stockant les données d'une table réelle.

MODIFIER TOUTES LES DONNÉES D'UNE TABLE

Exemple : augmenter de 10% les valeurs contenues dans le champ *Montant HT*

```
Private Sub Bouton_Augmente_Click()
' Défini RsAugmente en tant qu'objet en enregistrement-écriture
Dim RsAugmente As Recordset

    ' RsAugmente correspond aux enregistrements de la table Commande
    Set RsAugmente = CurrentDb.OpenRecordset("SELECT * FROM [Commande] ")

    ' Recommence la boucle tant que l'on est pas au dernier enregistrement
    Do While Not RsAugmente.EOF
        RsAugmente.Edit ' Autorise les modifications des champs
        ' Augmente de 10% le champ MontantHT avec une valeur entière Int():
        RsAugmente("[Montant HT]") = Int(RsAugmente("[Montant HT]") * 1.1)
        RsAugmente.Update ' mise à jour du champ modifier
        RsAugmente.MoveNext ' Passe à l'enregistrement suivant
    Loop ' retour au début de la boucle
    Refresh ' Mise à jour des données
End Sub
```

FONCTIONS DE RECHERCHES DE DOMAINES

Les fonctions de domaine permettent de faire des calculs de statistiques sur un champ contenu dans une table en fonction de certains critères

STRUCTURE EN VBA

Fonction("[Champs à calculer]","[Table]","Critère")

QUELQUES FONCTIONS DE DOMAINE

Les fonctions suivantes font un calcul du champ dans la table choisie en fonction du critère saisi.

- DSum() : somme
- DCount() : nombre
- DAvg() : moyenne
- DMax() : maximum
- DMin() : minimum
- DFirst() : première donnée
- DLast() : dernière donnée
- DLookup() : recherche d'une valeur

EXEMPLE DE FONCTIONS DE DOMAINE

Somme du champ *Chiffre d'affaires* si la ville est Paris

```
DSum("[Chiffre d'affaires]", "[Clients]", "[Ville]='Paris'")
```

Affiche le plus grand chiffre d'affaires dont l'activité correspond au contenu de la variable *Choix*

```
DMax("[Chiffre d'affaires]", "[Clients]", "[Activité]>='" & Choix "'")
```

Compte le nombre d'enregistrement dont le champ *Chiffre d'affaires* est supérieur à 100 000

```
DCount("[Chiffre d'affaires]", "[Clients]", "[Chiffre d'affaires]>=10000")
```

Affiche le nom de la première société dont la date d'ancienneté est avant 2017

```
DLookup("[Société]", "[Clients]", "[Ancienneté]>=#01/01/2017#")
```

IMPRIMER UN ÉTAT AU FORMAT PDF

Il est possible d'imprimer un état — ou un formulaire, une table, etc. — au format PDF.

Voici un exemple de code qui imprime l'état « État Clients » au format PDF dans le dossier courant

```
Private Sub Bouton_PDF_Click()
    Dim CheminPDF As String
    ' chemin de sauvegarde du fichier PDF
    CheminPDF = CurrentProject.Path & "\Suivi des Clients.pdf"
    DoCmd.OutputTo acOutputReport, "État Clients", "PDF", CheminPDF, True
End Sub
```

✓ Le True, à la fin lance l'affichage du fichier PDF, sinon mettre False pour créer le fichier pDF sans l'ouvrir

ACCÈS À D'AUTRES PROGRAMMES

RÉCUPÉRATION DE DONNÉES D'ACCESS VERS EXCEL

En VBA, il est possible de contrôler un classeur Excel à partir de commandes exécutées dans une base de données d'Access, ce qui peut être très pratique, par exemple, pour récupérer des données d'Access vers Excel.

PRÉPARATION

Avant, à partir de l'éditeur de code VBA, il est nécessaire d'activer l'accès aux objets d'Excel :

Menu Outils
Références
Cocher *Microsoft Excel 14.0 Object*

✓ Cette manipulation est à faire pour chaque base de données utilisant Excel

EXEMPLE DE CODE

Voici un exemple de code qui récupère la société, le chiffre d'affaires et la date d'ancienneté dans un formulaire d'Access afin d'envoyer ces données dans un classeur Excel

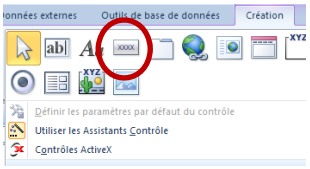
```
Private Sub Bouton_Excel_Click()
    Dim AppExcel As New Excel.Application ' Application Excel
    Dim WClasseur As Excel.Workbook ' Classeur
    Dim WFeuille As Excel.Worksheet ' Feuille de calcul
    Dim CheminFichier As String
    Dim ActuSoc As String, ActuCA As Single, ActuDate As Date

    ' Récupération des données
    ActuSoc = [Société]
    ActuCA = [Chiffre d'affaires]
    ActuDate = [Ancienneté]
    ' Chemin d'accès
    CheminFichier = CurrentProject.Path & "\Access_VBA.xlsx"
    'Lancement d'Excel
    Set AppExcel = CreateObject("Excel.Application")
    Set WClasseur = AppExcel.Workbooks.Open(CheminFichier)
    'Accés à la feuille « Récupération »
    Set WFeuille = AppExcel.Worksheets("Récupération")
    With WFeuille
        .Activate ' Activation de la feuille Récupération
        ' Affichage des données dans Excel
        .Cells(2, 1).Value = ActuSoc
        .Cells(2, 2).Value = ActuCA
        .Cells(2, 3).Value = ActuDate
    End With
    AppExcel.Visible = True ' Affichage d'Excel
    ' Libération de la mémoire pour Excel
    Set WFeuille = Nothing
    Set WClasseur = Nothing
    Set AppExcel = Nothing
End Sub
```

CRÉATION D'UNE MACRO EN VBA À PARTIR D'UN BOUTON DANS UN FORMULAIRE

CRÉER LE BOUTON

DESSINER LE BOUTON

1. Dans le formulaire, en mode Création, cliquer sur l'onglet *Création* et sur le bouton *Bouton*

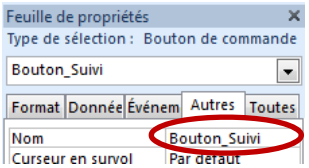


2. Tracer le cadre qui va contenir le bouton
3. Si nécessaire, cliquer sur le bouton **Annuler** pour arrêter l'assistant à la création de bouton
4. Changer le libellé du bouton directement dans le bouton

CHANGER LE NOM DU BOUTON

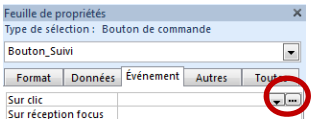
✓ et non le texte qui apparaît dans le bouton

1. Dans la feuille de propriétés — onglet *Création*, *Feuille de propriétés* si elle n'apparaît pas — accéder à l'onglet *Autres*
2. Dans la rubrique *Nom*, saisir le nom voulu, exemple : *Bouton_Suivi* et ↵



Saisir un code VBA qui sera affecté un bouton

1. Cliquer sur le bouton voulu
2. Dans la feuille de propriétés, accéder à l'onglet *Événements*
3. À droite de « Sur clic », cliquer sur le bouton de générateur de code (…)
   ou cliquer sur le bouton de liste déroulante (▾) pour affecter un code VBA existant



4. Cliquer sur « Générateur de code » et **OK**
5. Dans la fenêtre de code VBA qui apparaît, saisir le code voulu

POUR EXÉCUTER LA COMMANDE

1. Se remettre dans le formulaire en mode *Formulaire*
2. Cliquer sur le bouton voulu

✓ La macro s'exécute automatiquement

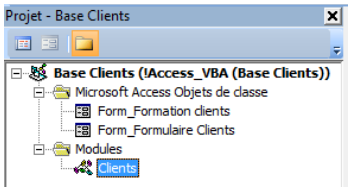
POUR RETROUVER SES COMMANDES VBA

POUR RETROUVER UNE COMMANDE QUI SE LANCE À PARTIR D'UN BOUTON

1. Dans le formulaire en mode Création, cliquer sur le bouton voulu
2. Dans la feuille de propriétés, accéder à l'onglet *Événements*
3. À droite de « Sur clic », cliquer sur le bouton de générateur de code (…)

POUR ACCÉDER À N'IMPORTE QUELLE COMMANDE

1. Onglet Créer
   Visual Basic ou **Alt | F11**
2. Dans l'explorateur de projets, à gauche,
   cliquer deux fois sur le formulaire
   ou le module contenant les macros voulues



✓ Si l'explorateur n'apparaît pas : menu *Affichage*, *Explorateur de projets*

LIENS

Déplacement dans les formulaires

https://access.developpez.com/faq/?page=PositionForm

Manipulation des données en DAO

http://warin.developpez.com/access/dao/?page=partie_2

http://warin.developpez.com/access/dao/?page=partie_4

http://warin.developpez.com/access/dao/?page=partie_5

Fonctions de domaine

http://starec.developpez.com/tuto/fonctionsdomaines