eBook Gratuit

APPRENEZ outlook-vba

eBook gratuit non affilié créé à partir des **contributeurs de Stack Overflow.**

#outlook-vba

Table des matières

À propos

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: outlook-vba

It is an unofficial and free outlook-vba ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official outlook-vba.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapitre 1: Démarrer avec outlook-vba

Remarques

Cette section fournit une vue d'ensemble de ce qu'est outlook-vba et pourquoi un développeur peut vouloir l'utiliser.

Il devrait également mentionner tous les grands sujets dans outlook-vba, et établir un lien avec les sujets connexes. La documentation de outlook-vba étant nouvelle, vous devrez peut-être créer des versions initiales de ces rubriques connexes.

Examples

introduction

Il existe actuellement trois sujets présentant Outlook VBA et au moins trois autres sont prévus.

La partie 1 explique comment accéder à Visual Basic Editor.

Si vous êtes un utilisateur d'Outlook 2003 et un utilisateur d'Excel VBA, vous apprendrez peu pour cette partie car l'accès à Outlook Visual Basic Editor est identique à celui d'Excel Visual Basic Editor.

Avec Outlook 2007 et versions ultérieures, l'onglet `Développeur` . qui donne accès à Visual Basic Editor, ne s'affiche pas pour une nouvelle installation. Pour afficher l'onglet `Développeur` , vous devez effectuer un certain nombre d'étapes décrites dans cette partie. Il n'y a pas de code dans cette partie.

Les parties 2 et 3 décrivent les magasins et les dossiers dans lesquels Outlook stocke les données. Vous pouvez les considérer comme l'équivalent des classeurs et feuilles de calcul d'Excel. La division entre les parties 2 et 3 est quelque peu arbitraire. La partie 2 décrit les magasins et les dossiers et inclut des macros pour afficher les noms de tous les magasins accessibles et des dossiers de niveau supérieur dans ces magasins. La partie 3 comprend une macro pour accéder aux dossiers de niveau inférieur. Une paire de macros utilise la récursivité qu'un nouveau programmeur peut trouver difficile à comprendre. Le lecteur devrait chercher à comprendre tout le code de la partie 2. Il serait cependant légitime de comprendre ce que fait cette paire de macros sans comprendre comment ils atteignent leur objectif.

La partie 4, la prochaine partie à écrire, présentera les `MailItems` qui contiennent les emails. La partie 3 comprend une macro pour déplacer un dossier d'un parent à un autre, mais la plupart des macros opèrent sur les objets contenus dans les dossiers, pas sur les dossiers eux-mêmes. A en juger par les questions sur le débordement de la pile, `MailItems` intéresse le plus les programmeurs.

La partie 5 présentera les `CalendarItems` qui contiennent des rendez-vous. La partie 6 présentera la création de nouveaux classeurs Excel à partir d'Outlook et la lecture et la mise à jour de

classeurs existants. La septième partie présentera les événements à moins qu'un sujet plus important soit immédiatement identifié.

Il est important de comprendre qu'il s'agit d'une introduction à Outlook VBA et non d'une introduction à VBA. La deuxième partie donne des indications sur la manière d'obtenir des informations sur VBA, mais comme la langue est la même sur tous les produits Office, une description de celle-ci appartient à cette introduction à Outlook VBA.

Outlook Visual Basic pour Applications

Visual Basic pour Applications (VBA) est le langage macro derrière tous les produits Microsoft Office et est essentiellement identique pour tous les produits Office. Ce qui diffère d'un produit à l'autre est le modèle d'objet. Excel contient des classeurs, des feuilles de calcul et des cellules. Access a des tables et des attributs. Outlook a des dossiers, des e-mails et des rendez-vous. C'est le modèle d'objet qui différencie Excel VBA de Outlook VBA.

Sujets avancés

Les différentes parties de l'introduction visent à fournir les informations dont tout programmeur nouveau pour Outlook VBA aurait besoin. Une grande partie du code a été à l'origine développée avec Outlook 2003 et a été testée avec Outlook 2016. Elle devrait rester inchangée avec toute version intermédiaire.

De nouvelles fonctionnalités ont été introduites depuis Outlook 2003, et les programmeurs souhaitent ou doivent y accéder. Il est prévu que des "sujets avancés" seront écrits pour décrire cette fonctionnalité.

Lire Démarrer avec outlook-vba en ligne: https://riptutorial.com/fr/outlook-vba/topic/8111/demarrer-avec-outlook-vba

Chapitre 2: Introduction Partie 1: Accès à Visual Basic Editor d'Outlook

Introduction

Accéder à Visual Basic Editor d'Outlook, insérer votre premier module et renommer ce module.

Connaissances préalables attendues : vous êtes un utilisateur Outlook.

Avec Outlook 2003, vous pouvez immédiatement sélectionner Visual Basic Editor. Avec les versions ultérieures, vous devez ajouter l'onglet Développeur avant de pouvoir sélectionner Visual Basic Editor.

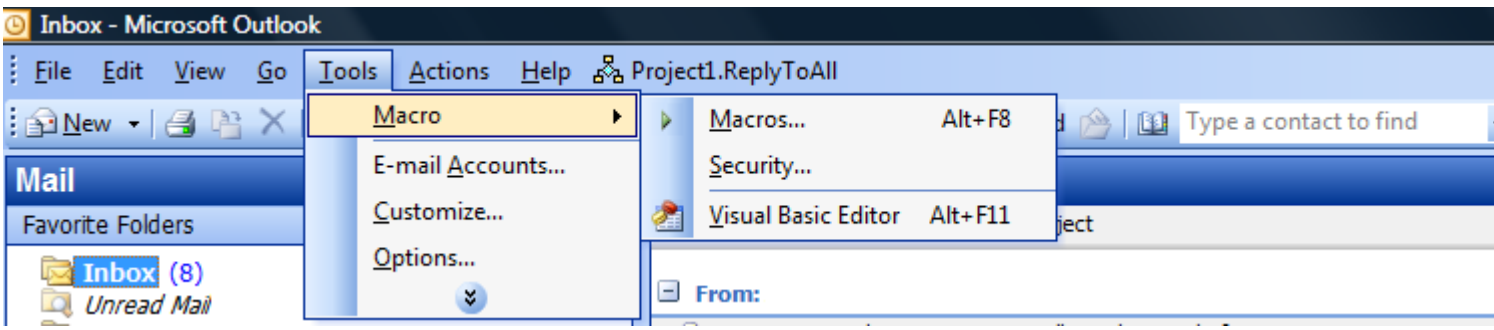
Examples

1.1 Accéder à Visual Basic Editor d'Outlook 2003

Toutes les images proviennent de versions britanniques d'Outlook. Je sais que certains noms sont traduits dans la langue locale pour d'autres versions et je suppose que la plupart des noms des onglets sont traduits. La séquence d'onglets est probablement inchangée dans les versions non anglaises. Alternativement, vous devrez regarder vos onglets et décider lequel serait équivalent, par exemple, à «Outils»

Avec Outlook 2003 ouvert, le haut de la fenêtre peut ressembler à:

Cliquez sur Outils et déplacez le curseur sur Macros pour voir:



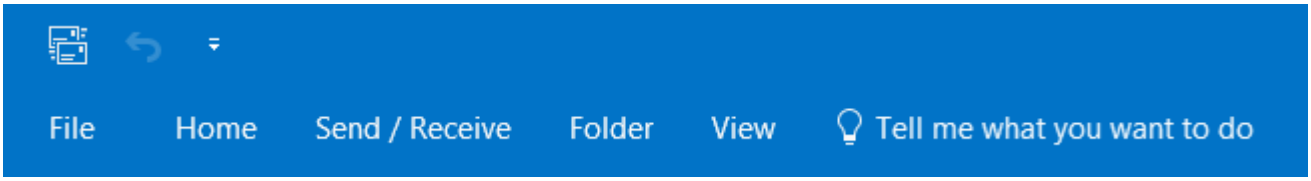
Déplacez le curseur vers le bas puis cliquez sur `Visual Basic Editor` . Sinon, quittez les sélections et cliquez sur `Alt` + `F11` .

1.2 Accéder à Visual Basic Editor dans Outlook 2007 et versions ultérieures

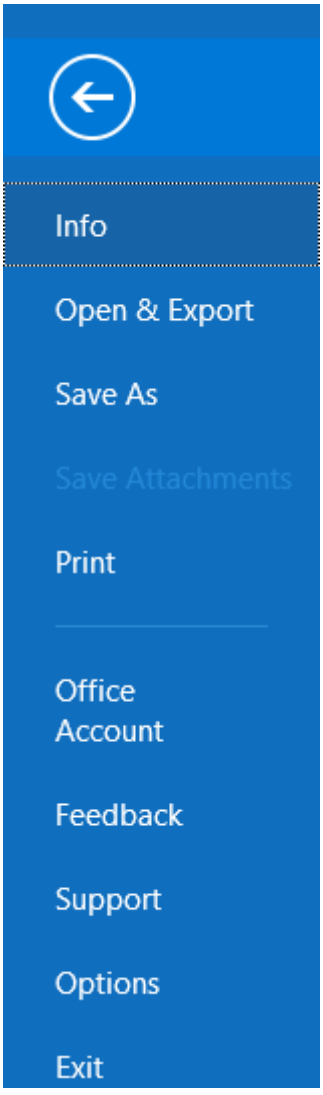
Toutes les images de cette section proviennent de la version britannique d'Outlook 2016. Je sais que certains noms sont traduits dans la langue locale pour d'autres versions et je suppose que la plupart des noms des onglets sont traduits. La séquence d'onglets est probablement inchangée dans les versions non anglaises. Alternativement, vous devrez regarder vos onglets et décider lequel serait équivalent, par exemple, à «Outils»

Les fenêtres Outlook 2010 sont formatées différemment mais sont essentiellement identiques. Je comprends que les autres versions sont également essentiellement identiques à Outlook 2016.

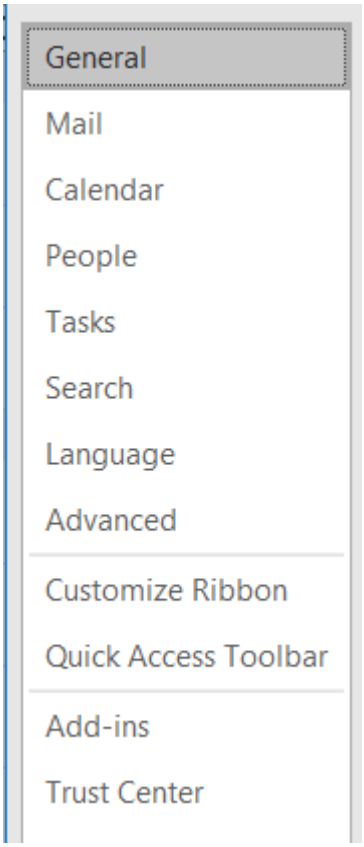
Le haut de la fenêtre principale peut ressembler à:



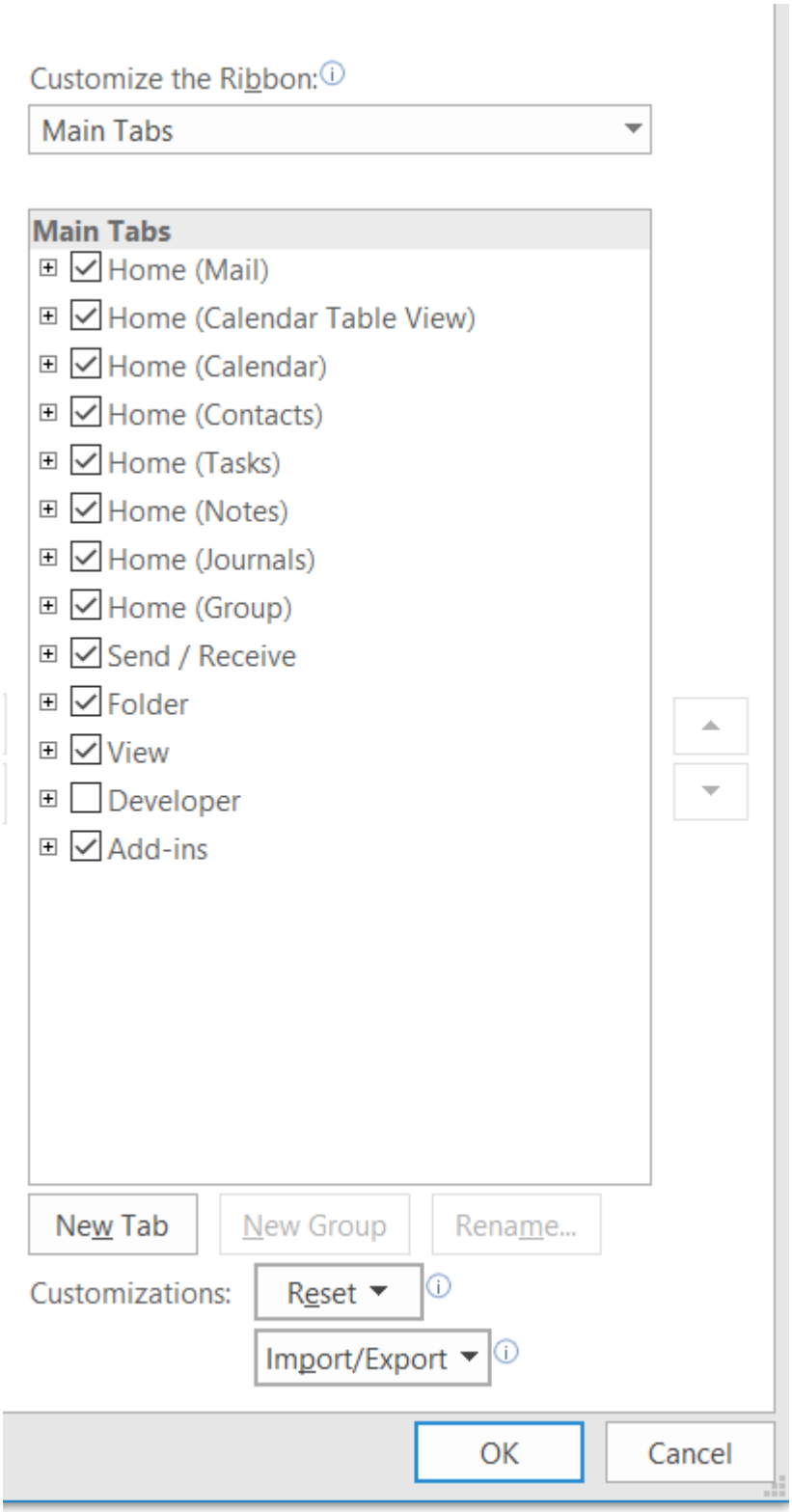
Cliquez sur `Fichier` , à gauche, pour obtenir les éléments suivants à gauche de la fenêtre:



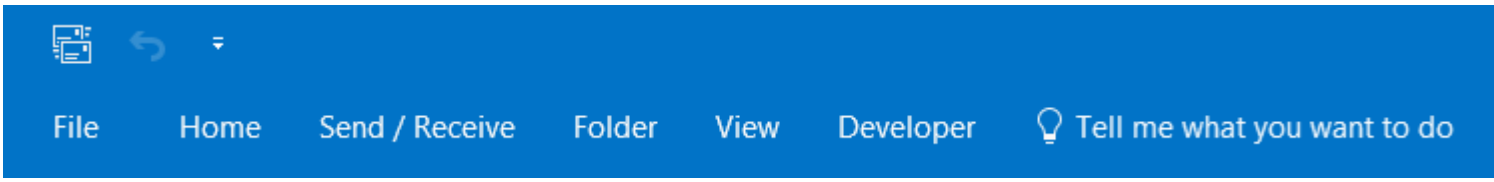
Cliquez sur `Options` , près du bas, pour obtenir les éléments suivants à gauche de la fenêtre:



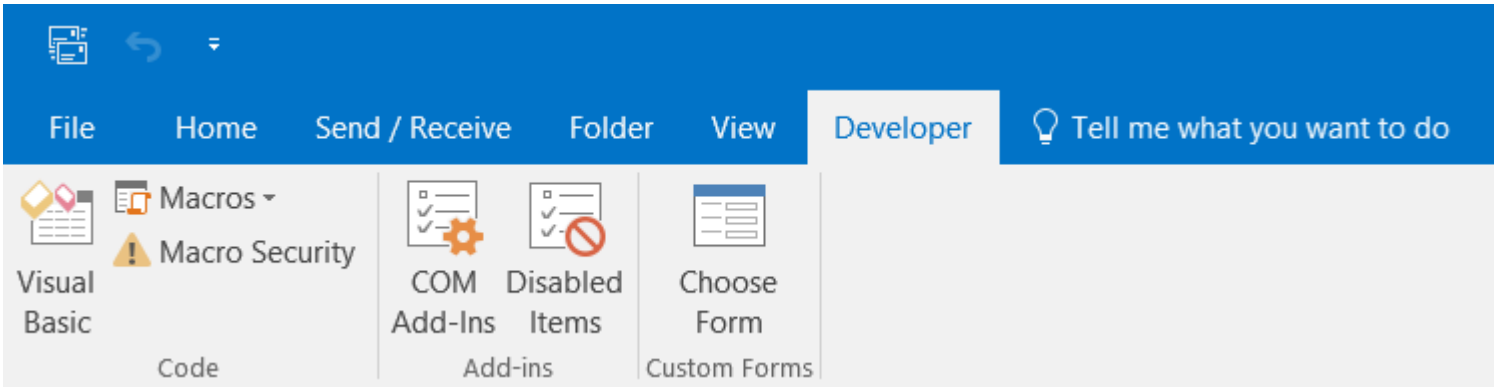
Cliquez sur `Personnaliser le ruban` , à mi-chemin. pour obtenir ce qui suit à droite de la fenêtre:



Cliquez sur la case à côté de «Developer», en bas, pour cocher la case, puis cliquez sur `OK` , en bas. La fenêtre principale réapparaîtra mais aura changé pour:



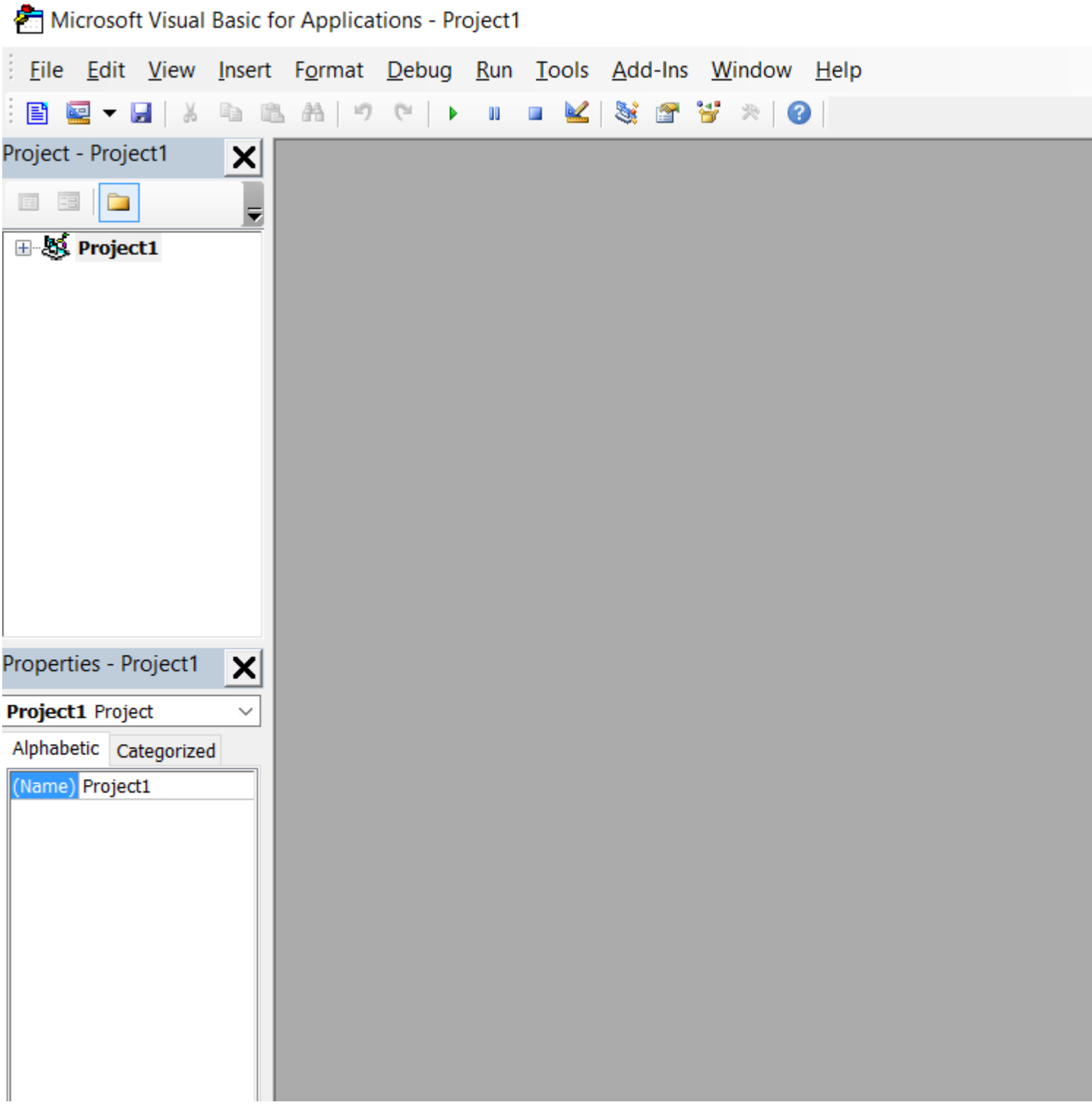
Cliquez sur le nouvel onglet `Developer` pour obtenir:



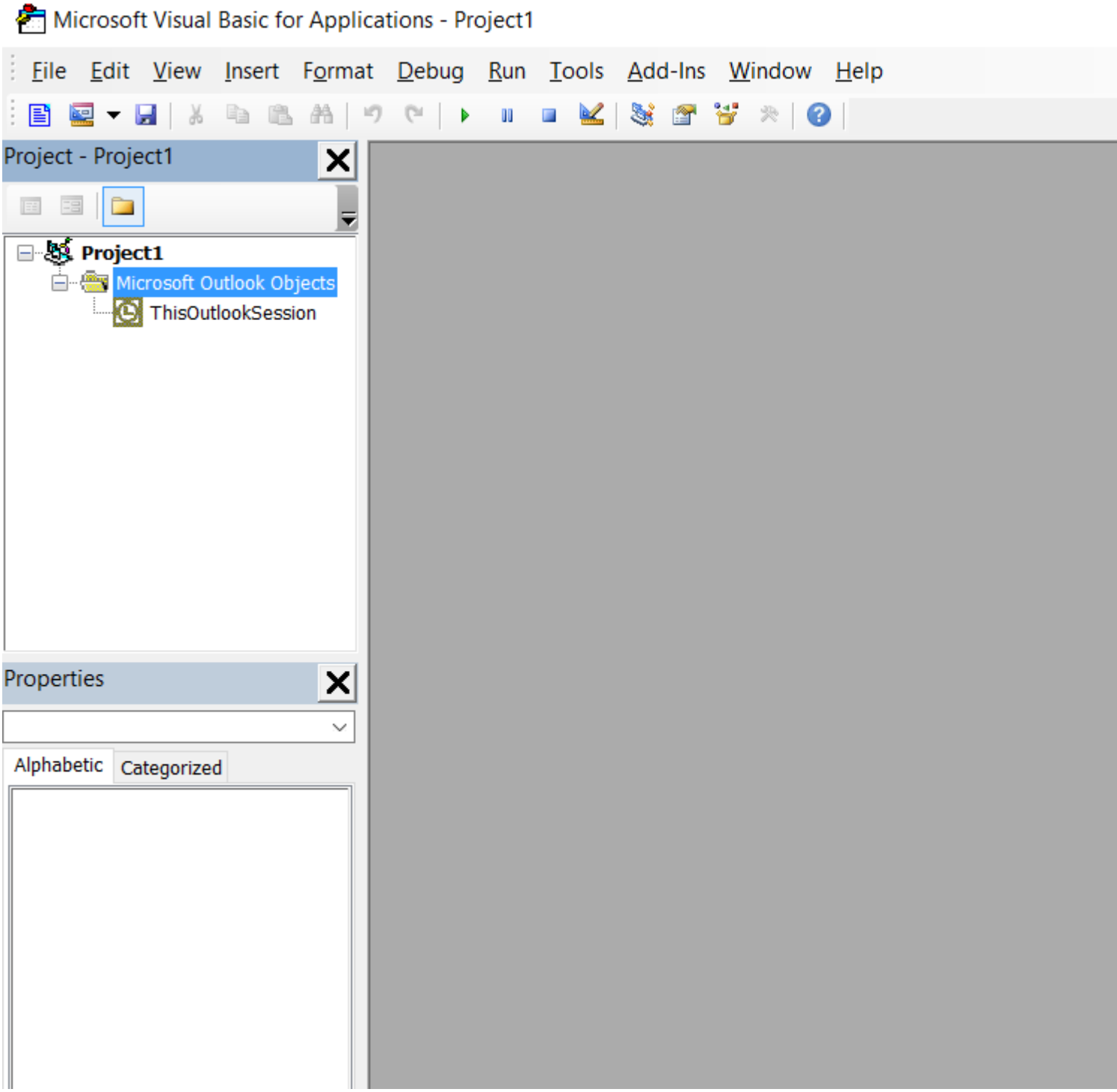
Cliquez sur `Visual Basic` , sur la gauche, pour sélectionner Visual Basic Editor.

1.3 Premiers pas avec Visual Basic Editor

Les images de cette section proviennent toutes d'Outlook 2016, mais elles pourraient provenir d'Outlook 2003. Outlook VBA a peut-être changé au fil des ans, mais à mes yeux, l'éditeur VBA ne l'a pas été. Quelle que soit votre version, vous verrez quelque chose comme:

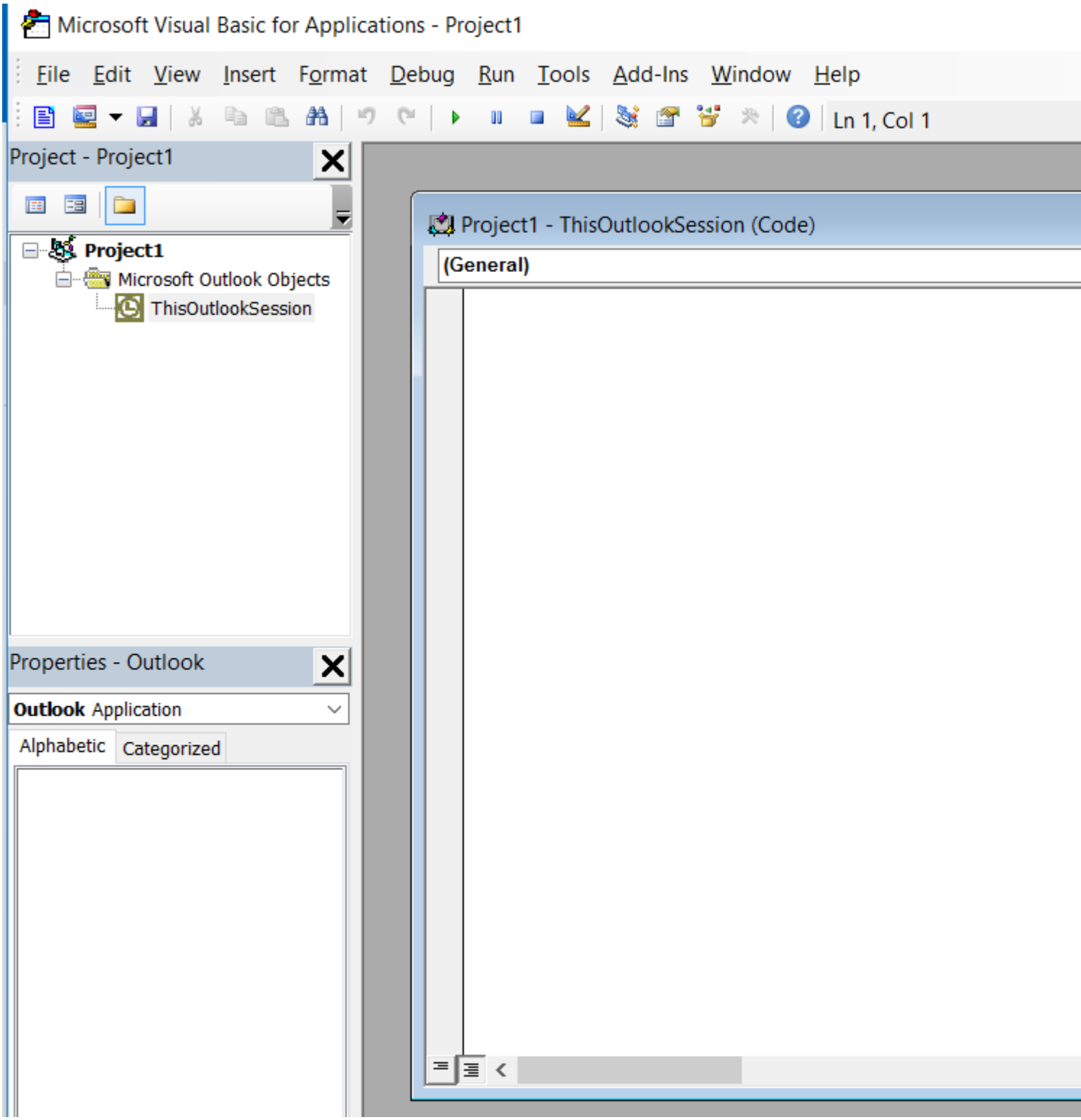


Au-dessus, il y a un «+» contre « **Projet1** ». Si vous avez un "+" cliquez dessus et ensuite le "+" contre "Microsoft Outlook Objects" pour obtenir:



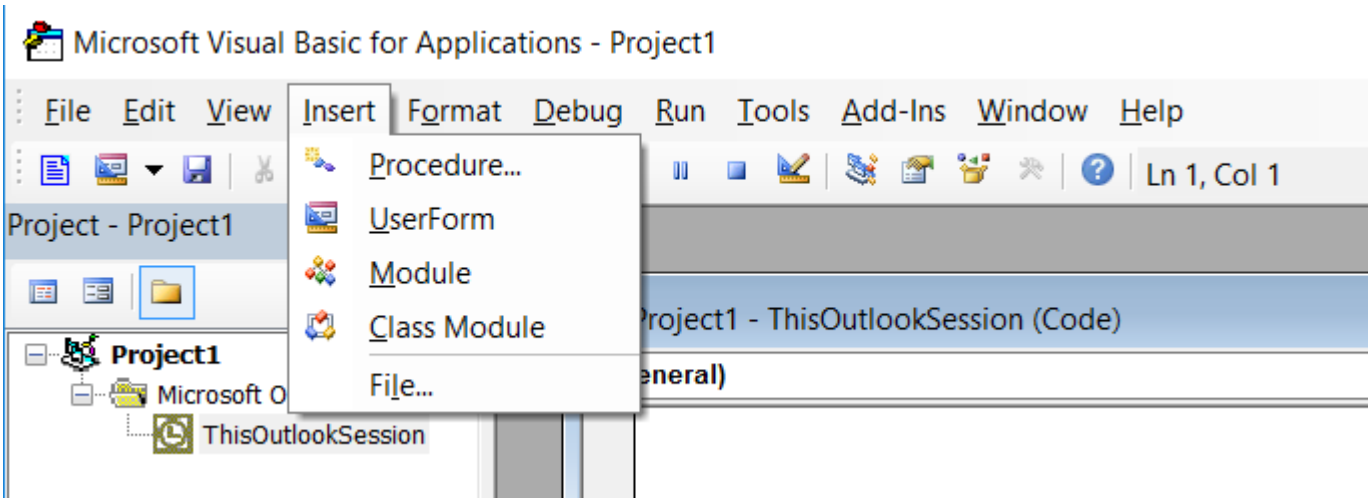
La fenêtre Propriétés peut ne pas être présente ou, si elle est présente, peut être positionnée ailleurs dans la fenêtre de l'éditeur VB. Nous n'en avons pas besoin pour le moment. Vous pouvez le fermer en cliquant sur la croix et utiliser `F4` pour le rendre visible à tout moment. Normalement, je ne l'ai pas vu car je n'ai pas besoin d'accéder aux propriétés la plupart du temps et ma liste Explorateur de projets occupe la plus grande partie du côté gauche. Je vous suggère de le garder visible jusqu'à ce qu'il devienne une nuisance.

Si vous cliquez sur `ThisOutlookSession` , la zone grise deviendra blanche ou, comme dans l'image ci-dessous, une fenêtre de code apparaîtra dans la zone grise:

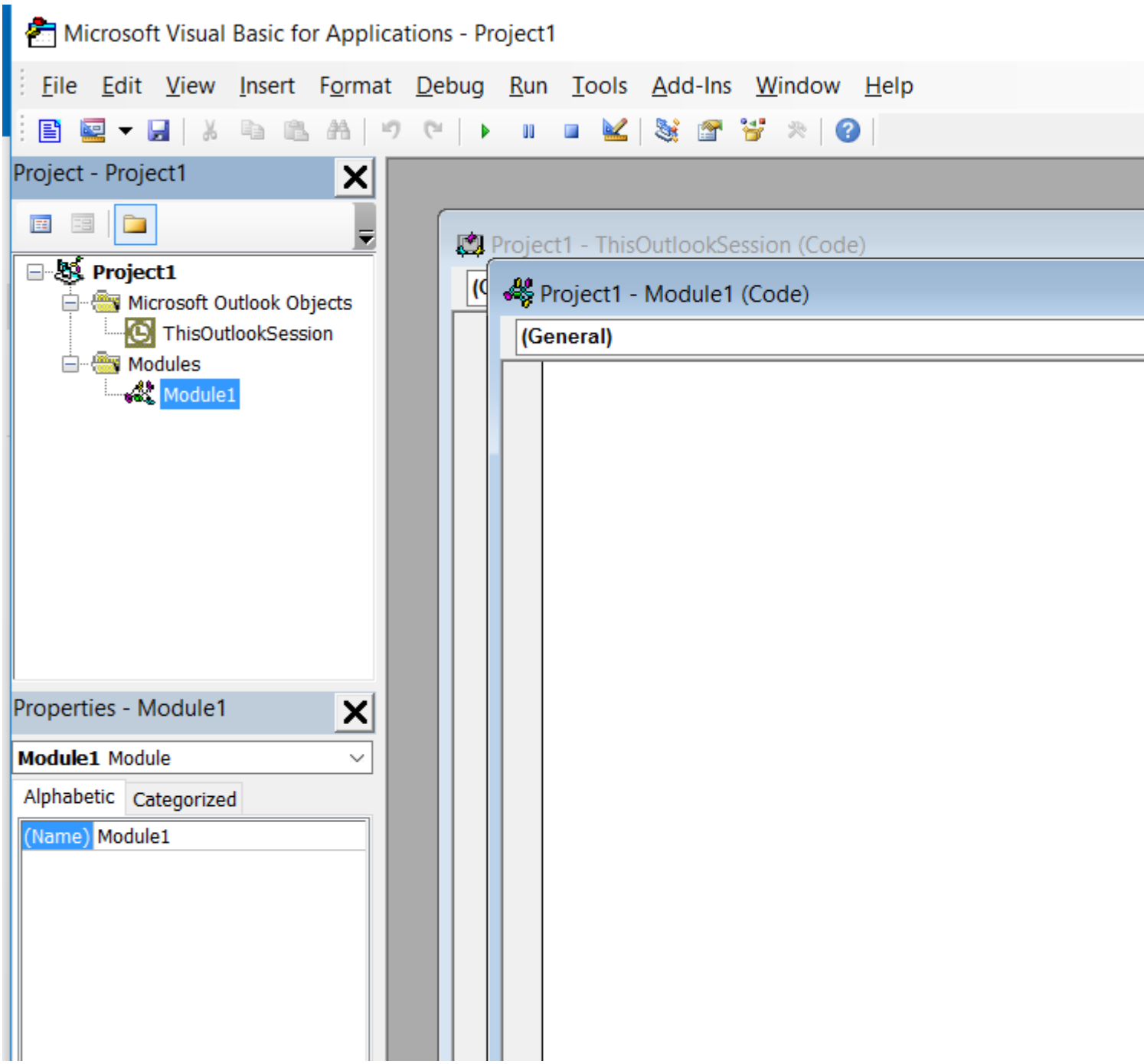


Vous pouvez taper n'importe quel code dans cette fenêtre de code. Cependant, les routines d'événement (qui sont discutées à la fin de ce tutoriel) doivent être saisies dans cette fenêtre de code. Je vous recommande de réserver la zone de code ThisOutlookSession pour les routines d'événement.

Au lieu de cela, cliquez sur `Insérer` pour obtenir:



Cliquez sur `Module` pour ajouter un module:



Mon nouveau module s'appelle "Module1". Si votre version d'Outlook est une version non anglaise, votre module aura un nom équivalent dans votre langue. Vous pouvez ajouter d'autres modules qui seront nommés "Module2", "Module3", etc.

Si je crée un classeur Excel, pour lequel je n'ai besoin que d'un seul module, je pourrais laisser le nom «Module1». Mais avec Outlook, toutes mes macros doivent aller ici, donc j'ai beaucoup de modules. Au fil des ans, j'ai écrit de nombreuses routines que je réutilise de manière répétée. J'ai un module pour les routines VBA générales, un autre pour les routines d'accès à Excel, un autre pour les routines Outlook VBA et un module par tâche Outlook que j'effectue avec des macros. Si vous regardez la fenêtre Propriétés, vous verrez que la seule propriété d'un module est son nom. Cliquez sur le "Module1" contre "Nom" et vous pouvez le changer pour un nom valide (commence par une lettre, contient des lettres et des chiffres uniquement, etc.). Vous obtenez des erreurs étranges si un module et une procédure ont le même nom, alors je commence tous les noms de modules avec “Mod” et je n'utilise pas ce préfixe pour mes procédures. Pourquoi ne pas renommer ce module «ModIntro» ou similaire prêt pour la prochaine partie de ce tutoriel?

Ces zones de code et les zones de saisie de données de n'importe quel éditeur. Cliquez sur la zone de code pour la sélectionner et tapez votre code ou collez-le dans le code copié ailleurs, tel que la section suivante de ce didacticiel.

1.4 Ce que vous devez retenir de cette partie du tutoriel

- Votre version d'Outlook a-t-elle besoin de vous pour ajouter l'onglet Développement? Si tel est le cas, vous n'aurez pas besoin de répéter ce processus avant d'avoir une nouvelle installation Outlook. Reviens ici quand cela arrive.
- Rappelez-vous comment entrer dans Visual Basic Editor.
- Rappelez-vous comment créer et renommer un module.

Lire Introduction Partie 1: Accès à Visual Basic Editor d'Outlook en ligne: https://riptutorial.com/fr/outlook-vba/topic/8877/introduction-partie-1--acces-a-visual-basic-editor-d-outlook

Chapitre 3: Introduction Partie 2: Stockages et dossiers de niveau supérieur

Introduction

Première partie d'une introduction aux magasins et aux dossiers qu'ils contiennent. Contient des macros pour afficher (1) les noms des magasins accessibles et (2) les noms des magasins accessibles et les dossiers de premier niveau qu'ils contiennent.

Examples

2.1 Connaissances préalables attendues

- Vous êtes un utilisateur Outlook et comprenez des termes tels que «courrier électronique», «heure de réception», «sujet» et «volet de dossiers».
- Vous savez comment accéder à Visual Basic Editor d'Outlook et créer un module. Voir Introduction Partie 1 si nécessaire.
- Vous avez au moins une connaissance de base de VBA. Je déclare les sous-programmes et les variables sans explication. J'utilise Withs, Ifs et Loops sans explication. Je vous dis que quelque chose est une collection. Je vous dis de copier du code sur un module et de l'exécuter. Il existe de nombreux didacticiels en ligne, mais la plupart sont destinés à Excel VBA et se concentrent davantage sur l'utilisation du langage avec Excel que sur le langage. La recherche de “tutoriel VBA” en amène certains qui se concentrent sur la langue plus que l'application qui semble satisfaisante.
- Vous n'êtes pas obligé de connaître le modèle d'objet Outlook; Ce tutoriel vous en présente une petite partie.

2,2 magasins

Outlook stocke les e-mails, les éléments de calendrier, les notes, les tâches, etc. dans les fichiers appelés **magasins** . Si vous regardez votre fenêtre de dossiers, vous verrez quelque chose comme:

```
Aaaaaaaaaa
  Inbox
  Drafts
  Deleted Items
  :  :

Bbbbbbbbbb
  Inbox
  Drafts
  Deleted Items
  :  :

Ccccccccc
```

```
:  :
```

"Aaaaaaaaaa", "Bbbbbbbbbb" et "Ccccccccc" sont les noms d'utilisateur ou d'affichage des magasins. J'ai toujours accepté les valeurs par défaut d'Outlook pour ces noms qui ont changé au fil des ans. Une fois le nom par défaut, mon nom est maintenant mon adresse e-mail. Le nom de fichier pour ces magasins peut être le même mais avec une extension telle que PST ou OST ou peut être quelque chose de complètement différent. Une macro VBA a besoin du nom d'utilisateur pour accéder à un magasin et ne concerne pas les noms de fichiers ou l'extension.

Vous pouvez avoir autant de magasins que vous le souhaitez. J'ai «fichier de données Outlook» qui a été créé pour moi lorsque j'ai installé Outlook. Lorsque j'ai ajouté des comptes pour mes adresses e-mail, Outlook a créé de nouveaux magasins nommés pour l'adresse e-mail, tels que «JohnDoe@hotmail.com» et «DoeJohn@gmail.com». Pour réduire la taille de mon magasin principal, je sauvegarde les anciens emails dans les magasins avec des noms tels que «Archive 2015».

Si vous êtes un utilisateur professionnel, vous pouvez avoir accès à des magasins partagés ou à des magasins de collègues.

Les macros ci-dessous montrent trois manières différentes de répertorier les magasins auxquels vous pouvez accéder. Je vous suggère de créer un nouveau module pour contenir le code ci-dessous et d'utiliser `F4` pour accéder aux propriétés du module afin que vous puissiez le nommer «ModIntro» ou un autre nom de votre choix. Si vous avez terminé la première partie de cette série, vous aurez déjà un tel module.

Copiez ces macros dans un module et testez que chacune donne la même sortie.

```
Sub ListStores1()

  Dim InxStoreCrnt As Integer
  Dim NS As NameSpace
  Dim StoresColl As Folders

  Set NS = CreateObject("Outlook.Application").GetNamespace("MAPI")
  Set StoresColl = NS.Folders

  For InxStoreCrnt = 1 To StoresColl.Count
    Debug.Print StoresColl(InxStoreCrnt).Name
  Next

End Sub
Sub ListStores2()

 Dim StoresColl As Stores
 Dim StoreCrnt As Store

 Set StoresColl = Session.Stores

 For Each StoreCrnt In StoresColl
   Debug.Print StoreCrnt.DisplayName
 Next

End Sub
Sub ListStores3()
```

```
  Dim InxStoreCrnt As Long

  With Application.Session
    For InxStoreCrnt = 1 To .Folders.Count
      Debug.Print .Folders(InxStoreCrnt).Name
    Next
  End With

End Sub
```

Vous constaterez avec VBA qu'il existe souvent plusieurs méthodes pour obtenir le même effet. J'ai montré ci-dessus trois méthodes pour accéder aux magasins. Vous n'avez pas besoin de vous souvenir de tous - choisissez votre propre favori - mais vous devez savoir qu'il existe plusieurs méthodes car d'autres personnes, dont vous pourriez avoir besoin d'étudier le code, auront des favoris différents.

Les variables `StoresColl` dans les macros `ListStores1()` et `ListStores2()` sont toutes deux des collections mais contiennent différents types d'objet: `Store` et `Folder` . Un objet `Store` ne peut référencer qu'un fichier sur votre disque. Un `Folder` peut référencer un fichier sur le disque, mais peut également référencer des dossiers dans un magasin, tels que «Boîte de réception» et «Éléments envoyés». `Stores` , les `Folders` , le `Store` et le `Folder` font tous partie du modèle d'objet Outlook. Cette série de didacticiels vous présente le modèle mais il ne s'agit pas d'une définition formelle. Si vous voulez une définition formelle, tapez «Outlook VBA Object Model» dans votre moteur de recherche préféré. Assurez-vous de regarder la version VBA du modèle.

2.3 Dossiers de haut niveau

Dans mon exemple de volet de dossiers ci-dessus, je ne répertorie que trois dossiers standard: «Boîte de réception», «Brouillons» et «Éléments supprimés». Il existe d'autres dossiers standard et vous pouvez créer autant de dossiers que vous le souhaitez. Certaines personnes créent des dossiers sous Boîte de réception, mais je préfère créer de nouveaux dossiers au même niveau que la boîte de réception. Vos dossiers peuvent avoir des sous-dossiers qui peuvent avoir leurs propres sous-dossiers à n'importe quelle profondeur.

La macro suivante produira une liste du formulaire:

```
A
   A1
   A2
   A3
B
   B1
   B2
C
   C1
   C2
   C3
   C4
```

où A, B et C sont des magasins et A1, B1, C1 et ainsi de suite sont des dossiers dans A, B et C. Si A1, B1, C1 et ainsi de suite ont des sous-dossiers, ils ne seront pas répertoriés par cette

macro. L'accès aux dossiers les plus imbriqués sera abordé dans la prochaine partie de ce tutoriel.

```
Sub ListStoresAndTopLevelFolders()

  Dim FldrCrnt As Folder
  Dim InxFldrCrnt As Long
  Dim InxStoreCrnt As Long
  Dim StoreCrnt As Folder

  With Application.Session
    For InxStoreCrnt = 1 To .Folders.Count
      Set StoreCrnt = .Folders(InxStoreCrnt)
      With StoreCrnt
        Debug.Print .Name
        For InxFldrCrnt = .Folders.Count To 1 Step -1
          Set FldrCrnt = .Folders(InxFldrCrnt)
          With FldrCrnt
            Debug.Print "   " & .Name
          End With
        Next
      End With
    Next
  End With

End Sub
```

2.4 Ce que vous devez retenir de ce tutoriel

- Un magasin est un fichier dans lequel Outlook stocke des e-mails, des éléments de calendrier, des notes, des tâches, etc.
- Un magasin peut contenir des dossiers standard Outlook tels que «Boîte de réception» et «Éléments envoyés».
- Un magasin peut également contenir des dossiers créés par l'utilisateur.
- Les dossiers standard Outlook et les dossiers créés par l'utilisateur peuvent contenir des sous-dossiers, des sous-sous-dossiers, etc. créés par l'utilisateur.
- Comment lister les magasins
- Comment lister les magasins et les dossiers de premier niveau dans ces magasins.

Confession: je ne me souviens d'aucun des «Hows». J'ai des sous-programmes et des fonctions qui me rappellent.

Lire Introduction Partie 2: Stockages et dossiers de niveau supérieur en ligne: https://riptutorial.com/fr/outlook-vba/topic/8876/introduction-partie-2--stockages-et-dossiers-de-niveau-superieur

Chapitre 4: Introduction Partie 3: Magasins et tous leurs dossiers

Introduction

Termine la présentation des magasins et des dossiers lancés dans la partie 2 de ce didacticiel

Connaissances préalables attendues : Vous avez étudié la partie 2 de ce tutoriel ou connaissez déjà son contenu.

Examples

3. 0 Contenu

- Comment référencer un dossier accessible.
- Comment obtenir le nom complet d'un dossier référencé.
- Une paire de routines qui répertorieront tous les dossiers de chaque magasin accessible.
- Une routine pour déplacer un dossier d'un dossier parent vers un autre.

3.1 Fonction GetFldrNames () nécessaire pour plusieurs macros de démonstration

Un certain nombre de macros de démonstration dans cette partie nécessitent une fonction que je vais expliquer plus tard. Pour le moment, veuillez simplement copier `GetFldrNames()` sur un module approprié. J'utilise fréquemment cette fonction et la garde, ainsi que d'autres fonctions similaires, que j'utilise dans de nombreuses macros différentes, dans un module nommé «ModGlobalOutlook». Vous pourriez aimer faire la même chose. Vous pouvez également préférer conserver la macro avec toutes les autres macros de cette série de didacticiels. vous pouvez le déplacer plus tard si vous changez d'avis.

```
Public Function GetFldrNames(ByRef Fldr As Folder) As String()

  ' * Fldr is a folder. It could be a store, the child of a store,
  '   the grandchild of a store or more deeply nested.
  ' * Return the name of that folder as a string array in the sequence:
  '    (0)=StoreName (1)=Level1FolderName (2)=Level2FolderName  ...

  ' 12Oct16  Coded
  ' 20Oct16  Renamed from GetFldrNameStr and amended to return a string array
  '          rather than a string

  Dim FldrCrnt As Folder
  Dim FldrNameCrnt As String
  Dim FldrNames() As String
  Dim FldrNamesRev() As String
  Dim FldrPrnt As Folder
  Dim InxFN As Long
```

```
  Dim InxFnR As Long

  Set FldrCrnt = Fldr
  FldrNameCrnt = FldrCrnt.Name
  ReDim FldrNamesRev(0 To 0)
  FldrNamesRev(0) = Fldr.Name
  ' Loop getting parents until FldrCrnt has no parent.
  ' Add names of Fldr and all its parents to FldrName as they are found
  Do While True
    Set FldrPrnt = Nothing
    On Error Resume Next
    Set FldrPrnt = Nothing   ' Ensure value is Nothing if following statement fails
    Set FldrPrnt = FldrCrnt.Parent
    On Error GoTo 0
    If FldrPrnt Is Nothing Then
      ' FldrCrnt has no parent
      Exit Do
    End If
    ReDim Preserve FldrNamesRev(0 To UBound(FldrNamesRev) + 1)
    FldrNamesRev(UBound(FldrNamesRev)) = FldrPrnt.Name
    Set FldrCrnt = FldrPrnt
  Loop

  ' Copy names to FldrNames in reverse sequence so they end up in the correct sequence
  ReDim FldrNames(0 To UBound(FldrNamesRev))
  InxFN = 0
  For InxFnR = UBound(FldrNamesRev) To 0 Step -1
    FldrNames(InxFN) = FldrNamesRev(InxFnR)
    InxFN = InxFN + 1
  Next

  GetFldrNames = FldrNames

End Function
```

3.2 Référencement d'un dossier par défaut

Dans `TestDefaultFldr()` je définis `Fldr` sur la boîte de réception par défaut. La constante `olFolderInbox` peut être remplacée par d'autres valeurs donnant accès à l'un des dossiers par défaut. Si vous tapez `Set Fldr = Session.GetDefaultFolder(` l'éditeur VB affichera une liste déroulante de toutes les valeurs possibles).

```
Sub TestDefaultFldr()

  Dim Fldr As Folder

  Set Fldr = Session.GetDefaultFolder(olFolderInbox)

  Debug.Print Join(GetFldrNames(Fldr), "|")

End Sub
```

Sur mon ordinateur portable, `TestDefaultFldr()` affiche `Outlook data file|Inbox` qui a été une surprise. J'ai écrit `GetFldrNames(Fldr)` pour m'assurer que le dossier que j'avais référencé était celui que je voulais. J'avais accédé à la boîte de réception par défaut et j'ai trouvé que c'était vide! Store "Fichier de données de sortie" est venu avec l'installation par défaut et je l'avais ignoré car

Outlook avait créé un magasin pour chacun de mes comptes de messagerie. Ce n'est qu'après avoir découvert ma boîte de réception vide par défaut que je pensais à la manière dont Outlook connaîtrait le compte dont je souhaiterais avoir le compte par défaut. Parmi les dossiers Outlook standard, il n'y a pas de valeur par défaut ou la valeur par défaut se trouve dans «Fichier de données de sortie». Il est peut-être possible de modifier la boîte de réception par défaut, mais je n'ai pas enquêté, car je ne suis pas sûr de celui de mes comptes de messagerie que je modifierais par défaut. Rappelez-vous simplement que tous vos éléments de calendrier, vos tâches, etc., se trouvent dans le «fichier de données Outlook» et assurez-vous d'inclure «Outlook.pst» dans votre liste d'archives.

La plupart des objets Outlook ont la propriété `Parent` . `GetFldrNames(Fldr)` enregistre le nom du dossier dans un tableau avant d'essayer d'accéder à son parent. Il boucle des noms en ajoutant des noms à la fin du tableau jusqu'à ce qu'il atteigne le magasin. Le magasin n'a pas de parent et la tentative d'accès à celui-ci échoue. La séquence de noms dans le tableau est inversée, puis renvoyée à l'appelant. J'ai utilisé `Join` pour transformer le tableau de noms en une chaîne affichable.

3.3 Référencement d'un dossier dans un magasin accessible

`TestFldrChain()` montre comment référencer un dossier dans un magasin accessible:

```
Sub TestFldrChain()

  Dim Fldr As Folder

  Set Fldr = Session.Folders("A").Folders("A2"). _
                           Folders("A21").Folders("A213")

  Debug.Print Join(GetFldrNames(Fldr), "|")

End Sub
```

Dans `TestFldrChain()` : A est le nom d'un magasin; A2 est le nom d'un dossier dans A; A21 est le nom d'un dossier dans A2 et A213 est le nom d'un dossier dans A21.

Que se passe-t-il ici?

`Session` a une propriété `Folders` qui est une liste de tous les magasins accessibles.

`Session.Folders(integer)` , que j'ai utilisé dans la partie 2 de ce tutoriel, me permet de parcourir les magasins en séquence lorsque je ne connais pas leurs noms. `Session.Folders("A")` me permet d'accéder à un dossier lorsque je connais son nom.

`Session.Folders("A")` est un dossier et possède également une propriété `Folders` .

`Session.Folders("A").Folders("A2")` me donnent accès au dossier "A2" dans le magasin "A".

Je peux enchaîner autant de `Folders("x")` s que nécessaire pour atteindre un dossier. Si la chaîne est trop longue pour une ligne, vous pouvez diviser la déclaration en plusieurs lignes.

Recherchez le dossier le plus imbriqué dans votre installation et remplacez A, A2, A21 et A213

par les noms de votre magasin et de vos dossiers. Augmentez ou diminuez le nombre de dossiers dans la chaîne si nécessaire.

Si vous mettez à jour et exécutez `TestFldrChain()` , les résultats suivants seront `TestFldrChain()` , sauf que A, A2 et ainsi de suite auront été remplacés par les noms de vos dossiers:

```
A|A2|A21|A213
```

3.4 Liste des noms de chaque dossier dans chaque magasin accessible

Dans la deuxième partie, il vous a été montré comment répertorier chaque magasin accessible et les dossiers de niveau supérieur dans chaque magasin. Cela impliquait une boucle dans les magasins, puis une boucle pour chaque magasin via ses dossiers. Ci-dessus, vous avez vu comment référencer un dossier connu à n'importe quelle profondeur dans la hiérarchie des dossiers. Cela impliquait de chaîner autant de `Folders("x")` s que nécessaire pour atteindre le dossier.

Je souhaite maintenant répertorier chaque dossier, à n'importe quelle profondeur, dans chaque magasin. La technique de codage la plus simple pour résoudre ce type de problème lorsque vous devez abaisser des chaînes de différentes longueurs est la **récursivité** . Si vous êtes un programmeur sérieux dans un autre langage ou outil, vous connaissez peut-être déjà la récursivité. Si vous avez l'ambition d'être un programmeur sérieux, vous devrez comprendre la récursivité à terme, mais pas nécessairement aujourd'hui. La «récursivité» est l'un de ces concepts que beaucoup trouvent difficile à saisir au début. Vous pouvez taper «Récursivité» dans votre moteur de recherche préféré et lire les différentes tentatives d'explication de ce concept. Alternativement, vous pouvez accepter ces travaux macro mais ne vous inquiétez pas comment ils fonctionnent.

Notez le commentaire dans `ListStoresAndAllFolders()` : ces macros nécessitent une référence à «Microsoft Scripting Runtime». Cliquez sur `Outils` dans la barre d'onglets située en haut de la fenêtre de l'éditeur VB, puis cliquez sur `Références` . Vous obtiendrez une liste de toutes les références disponibles (bibliothèques). Certains au sommet seront déjà cochés. Les autres sont en ordre alphabétique. Faites défiler la liste et cliquez sur la case à gauche de «Microsoft Scripting Runtime» pour obtenir une coche. Puis cliquez sur `OK`

```
Sub ListStoresAndAllFolders()

 ' Displays the name of every accessible store
 ' Under each store, displays an indented list of all its folders

 ' Technique for locating desktop from answer by Kyle:
 ' http://stackoverflow.com/a/17551579/973283

 ' Needs reference to "Microsoft Scripting Runtime" if "TextStream"
 ' and "FileSystemObject" are to be recognised

 Dim FileOut As TextStream
 Dim FldrCrnt As Folder
 Dim Fso As FileSystemObject
 Dim InxFldrCrnt As Long
 Dim InxStoreCrnt As Long
```

```
  Dim Path As String
  Dim StoreCrnt As Folder

  Path = CreateObject("WScript.Shell").SpecialFolders("Desktop")

  Set Fso = CreateObject("Scripting.FileSystemObject")
  Set FileOut = Fso.CreateTextFile(Path & "\ListStoresAndAllFolders.txt", True)

  With Application.Session
    For InxStoreCrnt = 1 To .Folders.Count
      Set StoreCrnt = .Folders(InxStoreCrnt)
      With StoreCrnt
        FileOut.WriteLine .Name
        For InxFldrCrnt = .Folders.Count To 1 Step -1
          Set FldrCrnt = .Folders(InxFldrCrnt)
          Call ListAllFolders(FldrCrnt, 1, FileOut)
        Next
      End With
    Next
  End With

  FileOut.Close

End Sub
Sub ListAllFolders(ByRef Fldr As Folder, ByVal Level As Long, ByRef FileOut As TextStream)

  ' This routine:
  '  1. Output name of Fldr
  '  2. Calls itself for each child of Fldr
  ' It is designed to be called by ListStoresAndAllFolders()

  Dim InxFldrCrnt As Long

  With Fldr
    FileOut.WriteLine Space(Level * 2) & .Name
    For InxFldrCrnt = .Folders.Count To 1 Step -1
      Call ListAllFolders(.Folders(InxFldrCrnt), Level + 1, FileOut)
    Next
  End With

End Sub
```

Après avoir exécuté `ListStoresAndAllFolders` , il y aura un nouveau fichier sur votre DeskTop nommé «ListStoresAndAllFolders.txt» qui contiendra la liste promise des magasins et des dossiers.

3.5 Déplacement d'un dossier d'un dossier parent vers un autre

Pourquoi est-ce que je veux référencer un dossier? Dans la partie suivante, je vais vous montrer comment accéder aux e-mails dans un dossier référencé. Ici, je vais vous montrer comment déplacer un dossier. J'ai créé un dossier nommé «Test» dans ma boîte de réception. Dans `TestMoveFolder()` , j'ai remplacé «A» par le nom du magasin contenant ma boîte de réception. L'exécution de `TestMoveFolder()` déplacé «Test» vers «Éléments supprimés».

```
Sub TestMoveFolder()

  Dim FldrDest As Folder
```

```
  Dim FldrToMove As Folder

  Set FldrToMove = Session.Folders("A").Folders("Inbox").Folders("Test")
  Set FldrDest = Session.Folders("A").Folders("Deleted Items")

  FldrToMove.MoveTo FldrDest

End Sub
```

3.6 Ce que vous devez retenir de cette partie du tutoriel

- Comment référencer un dossier par défaut et les limitations possibles de cette technique.
- Comment référencer un dossier unique à n'importe quelle profondeur dans un magasin accessible.
- Comment afficher le nom complet d'un dossier référencé.
- Comment référencer l'une des nombreuses bibliothèques disponibles qui fournissent des fonctionnalités au-delà de l'ensemble de sous-programmes et de fonctions par défaut.
- Comment afficher le nom de chaque dossier dans chaque magasin accessible.
- Comment déplacer un dossier d'un dossier parent vers un autre.

Lire Introduction Partie 3: Magasins et tous leurs dossiers en ligne: https://riptutorial.com/fr/outlook-vba/topic/8874/introduction-partie-3--magasins-et-tous-leurs-dossiers

Crédits

S. No	Chapitres	Contributeurs
1	Démarrer avec outlook-vba	Community, Tony Dallimore
2	Introduction Partie 1: Accès à Visual Basic Editor d'Outlook	Tony Dallimore
3	Introduction Partie 2: Stockages et dossiers de niveau supérieur	Tony Dallimore
4	Introduction Partie 3: Magasins et tous leurs dossiers	Tony Dallimore