EBook Gratis

APRENDIZAJE outlook-vba

Free unaffiliated eBook created from
Stack Overflow contributors.

#outlook-vba

Tabla de contenido

Acerca de

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

Capítulo 1: Empezando con Outlook-VBA

Observaciones

Esta sección proporciona una descripción general de qué es Outlook-vba y por qué un desarrollador puede querer usarlo.

También debe mencionar cualquier tema importante dentro de outlook-vba y vincular a los temas relacionados. Dado que la Documentación para outlook-vba es nueva, es posible que deba crear versiones iniciales de esos temas relacionados.

Examples

Introducción

Actualmente hay tres temas que introducen Outlook VBA y se planean al menos tres más.

La Parte 1 describe cómo obtener acceso al Editor de Visual Basic.

Si es usuario de Outlook 2003 y usuario de Excel VBA, aprenderá poco sobre esta parte, ya que acceder al Editor de Visual Basic de Outlook es lo mismo que acceder al Editor de Visual Basic de Excel.

Con Outlook 2007 y versiones posteriores, la pestaña `Desarrollador` . que da acceso al Editor de Visual Basic, no se muestra para una nueva instalación. Para mostrar la pestaña `Desarrollador` , debe realizar una serie de pasos que se describen en esta parte. No hay código en esta parte.

Las partes 2 y 3 describen tiendas y carpetas que son donde Outlook almacena datos. Puede pensar en ellos como el equivalente de los libros y las hojas de trabajo de Excel. La división entre parte 2 y 3 es algo arbitraria. La Parte 2 describe las tiendas y carpetas e incluye macros para mostrar los nombres de todas las tiendas accesibles y las carpetas de nivel superior dentro de esas tiendas. La parte 3 incluye una macro para acceder a las carpetas de nivel inferior. Un par de macros utiliza la recursión que un programador nuevo puede encontrar difícil de entender. El lector debe tratar de comprender todo el código en la Parte 2. Sin embargo, sería legítimo entender qué hace ese par de macros pero no entender cómo logran su objetivo.

La parte 4, la siguiente parte que se escribirá, presentará `MailItems` que contienen correos electrónicos. La Parte 3 incluye una macro para mover una carpeta de un padre a otro, pero la mayoría de las macros operan en los objetos contenidos dentro de las carpetas y no en las carpetas. A juzgar por las preguntas sobre el desbordamiento de pila, `MailItems` es de mayor interés para los programadores.

La parte 5 presentará `CalendarItems` que tienen citas. La Parte 6 introducirá la creación de nuevos libros de Excel desde Outlook y la lectura y actualización de los libros existentes. La Parte 7 introducirá eventos a menos que se identifique algún tema más importante de inmediato.

Es importante entender que esto es una introducción a Outlook VBA, no una introducción a VBA. La Parte 2 brinda orientación sobre dónde obtener información sobre VBA, pero dado que el lenguaje es el mismo en todos los productos de Office, una descripción de la misma pertenece a esta introducción a Outlook VBA.

Outlook Visual Basic para Aplicaciones

Visual Basic para aplicaciones (VBA) es el lenguaje de macros detrás de todos los productos de Microsoft Office y es esencialmente idéntico en todos los productos de Office. Lo que difiere de un producto a otro es el modelo de Objeto. Excel tiene libros de trabajo, hojas de cálculo y celdas. El acceso tiene tablas y atributos. Outlook tiene carpetas, correos electrónicos y citas. Es el Modelo de objetos el que hace que Excel VBA sea diferente de Outlook VBA.

Temas avanzados

Las diversas partes de la introducción pretenden proporcionar la información que necesitaría cualquier programador nuevo en Outlook VBA. Gran parte del código se desarrolló originalmente con Outlook 2003 y se probó con Outlook 2016. Debería funcionar sin cambios con cualquier versión intermedia.

Se ha introducido una nueva funcionalidad desde Outlook 2003 a la que los programadores desearán o necesitarán acceder. Se prevé que se escribirán "temas avanzados" para describir esta funcionalidad.

Lea Empezando con Outlook-VBA en línea: https://riptutorial.com/es/outlook-vba/topic/8111/empezando-con-outlook-vba

Capítulo 2: Introducción Parte 1: Obtener acceso al Editor de Visual Basic de Outlook

Introducción

Obtener acceso al Editor de Visual Basic de Outlook, insertar su primer módulo y cambiar el nombre de ese módulo.

Conocimiento previo esperado : usted es un usuario de Outlook.

Con Outlook 2003, puede seleccionar inmediatamente el Editor de Visual Basic. Con versiones posteriores, debe agregar la pestaña Desarrollador antes de poder seleccionar el Editor de Visual Basic.

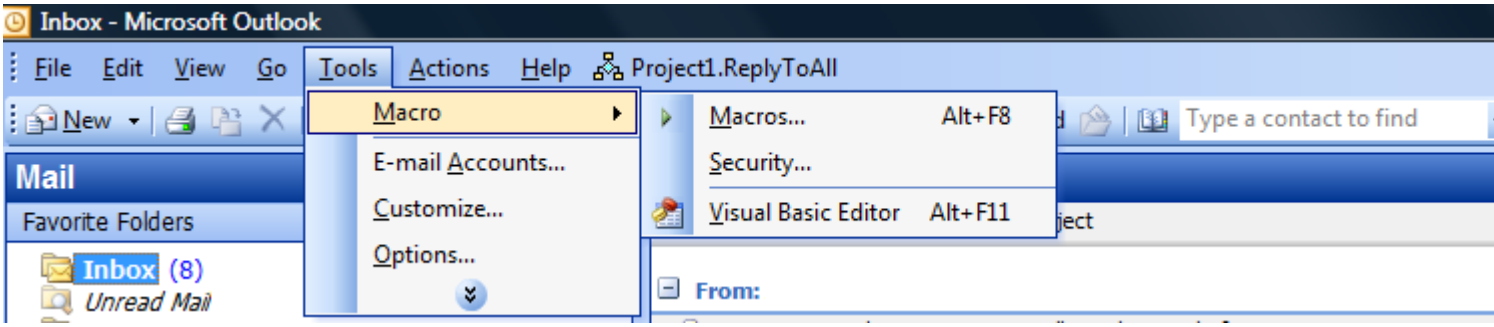
Examples

1.1 Obtener acceso al Editor de Visual Basic de Outlook 2003

Todas las imágenes son de las versiones del Reino Unido de Outlook. Sé que algunos nombres se traducen al idioma local para otras versiones y asumo que la mayoría de los nombres de las pestañas están traducidos. Probablemente la secuencia de pestañas no se modifique en las versiones que no están en inglés. Alternativamente, tendrá que mirar sus pestañas y decidir cuál sería equivalente a, por ejemplo, "Herramientas"

Con Outlook 2003 abierto, la parte superior de la ventana podría verse así:

Haga clic en Herramientas y mueva el cursor a Macros para ver:



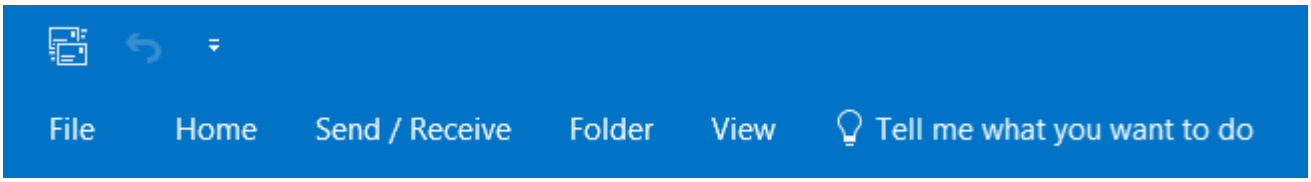
Mueva el cursor hacia la derecha y luego hacia abajo y haga clic en `Editor de Visual Basic` . Como alternativa, salga de las selecciones y haga clic en `Alt` + `F11` .

1.2 Obtención de acceso al Editor de Visual Basic en Outlook 2007 y versiones posteriores

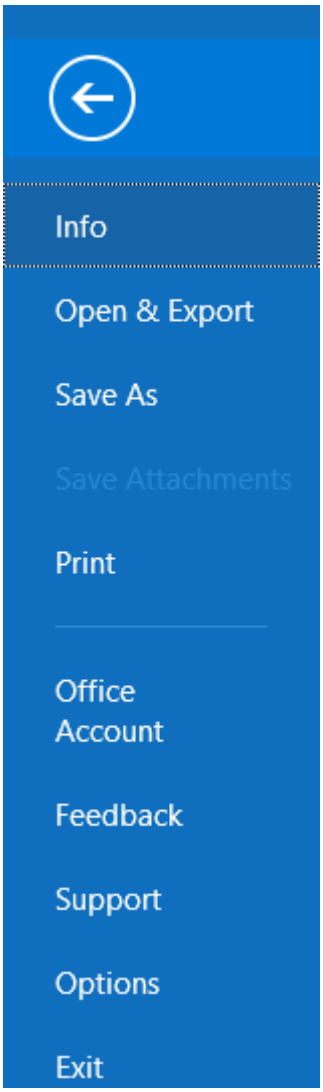
Todas las imágenes en esta sección son de la versión de Outlook 2016 para el Reino Unido. Sé que algunos nombres se traducen al idioma local para otras versiones y asumo que la mayoría de los nombres de las pestañas están traducidos. Probablemente la secuencia de pestañas no se modifique en las versiones que no están en inglés. Alternativamente, tendrá que mirar sus pestañas y decidir cuál sería equivalente a, por ejemplo, "Herramientas"

Las ventanas de Outlook 2010 tienen un formato diferente pero son esencialmente idénticas. Entiendo que otras versiones también son esencialmente idénticas a Outlook 2016.

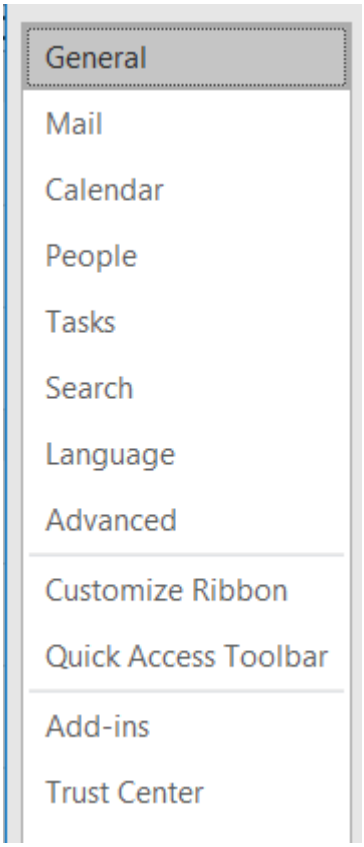
La parte superior de la ventana principal podría verse así:



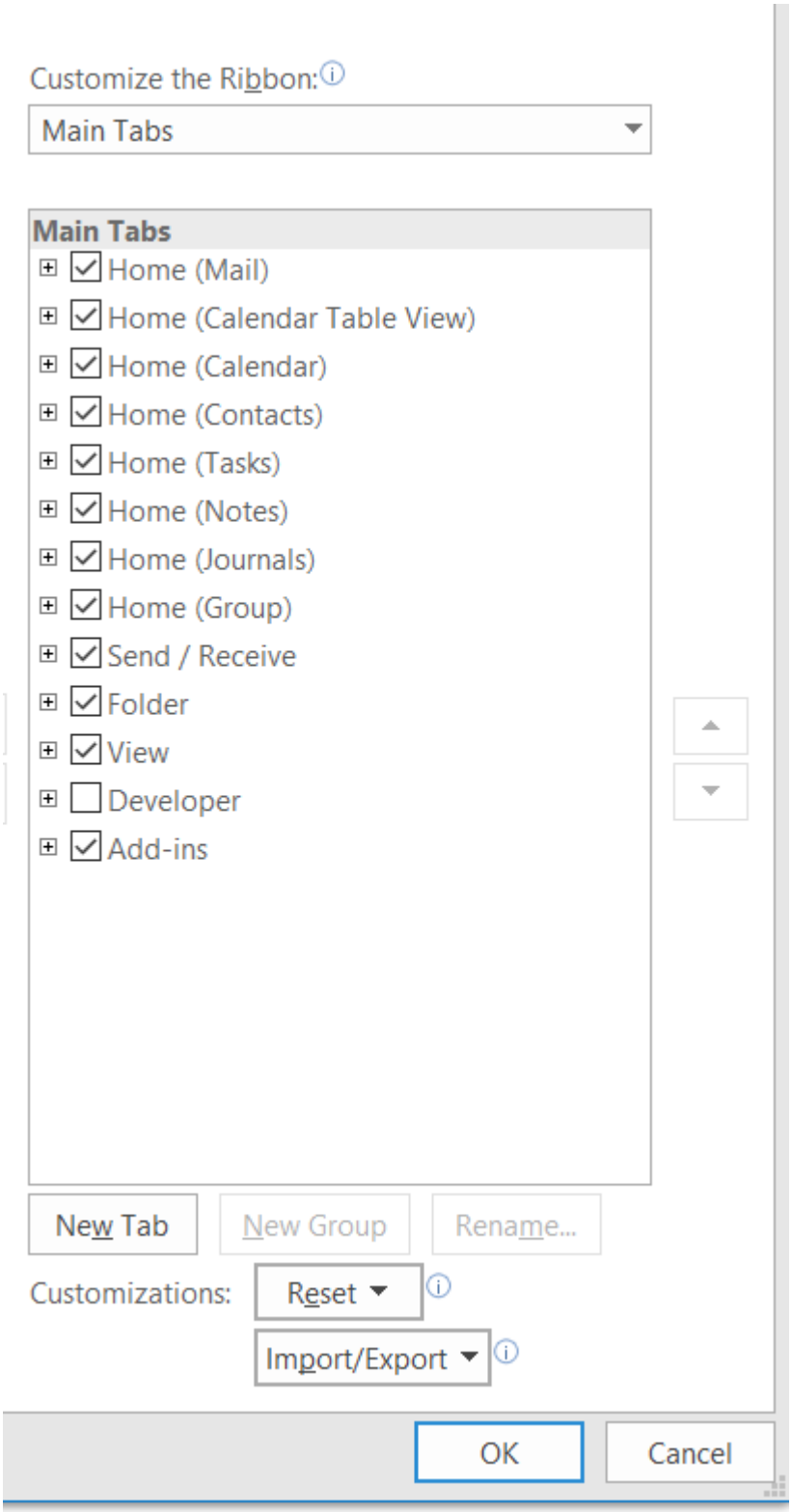
Haga clic en `Archivo` , a la izquierda, para obtener lo siguiente a la izquierda de la ventana:



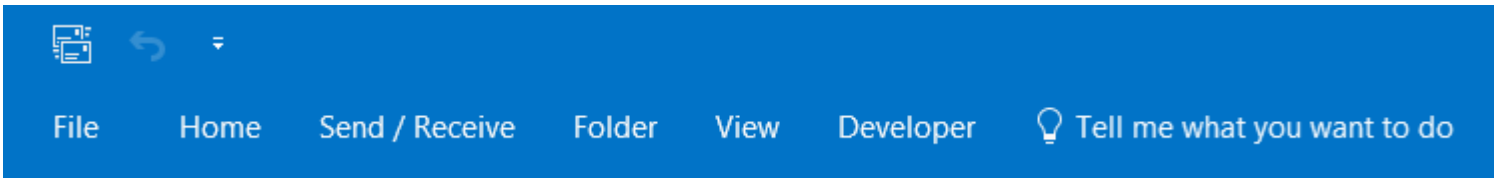
Haga clic en `Opciones` , cerca de la parte inferior, para obtener lo siguiente a la izquierda de la ventana:



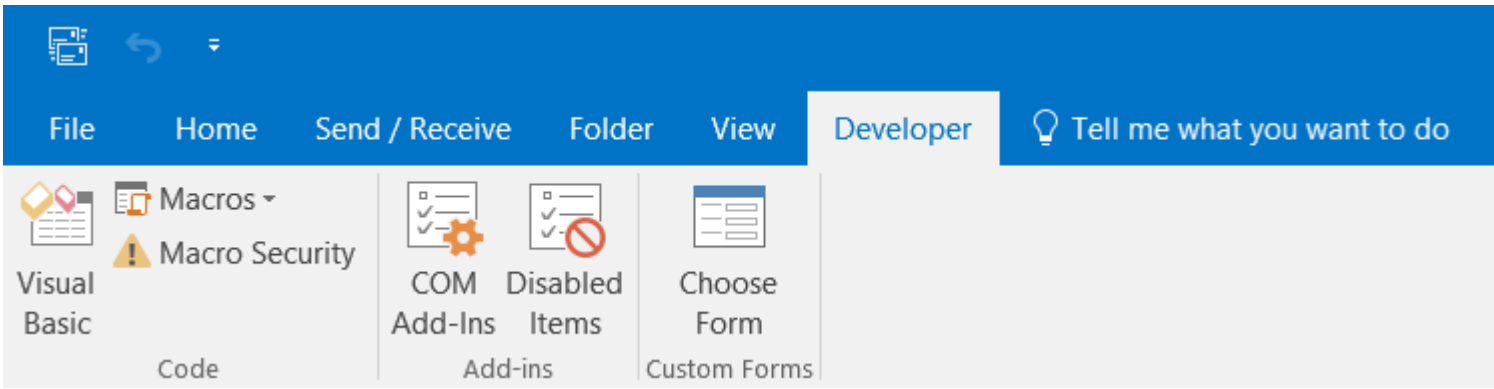
Haga clic en `Personalizar la cinta` , hasta la mitad. para obtener lo siguiente a la derecha de la ventana:



Haga clic en la casilla junto a "Desarrollador", cerca de la parte inferior, para obtener una marca y luego haga `clic` en `Aceptar` , en la parte inferior. La ventana principal volverá a aparecer pero se habrá cambiado a:



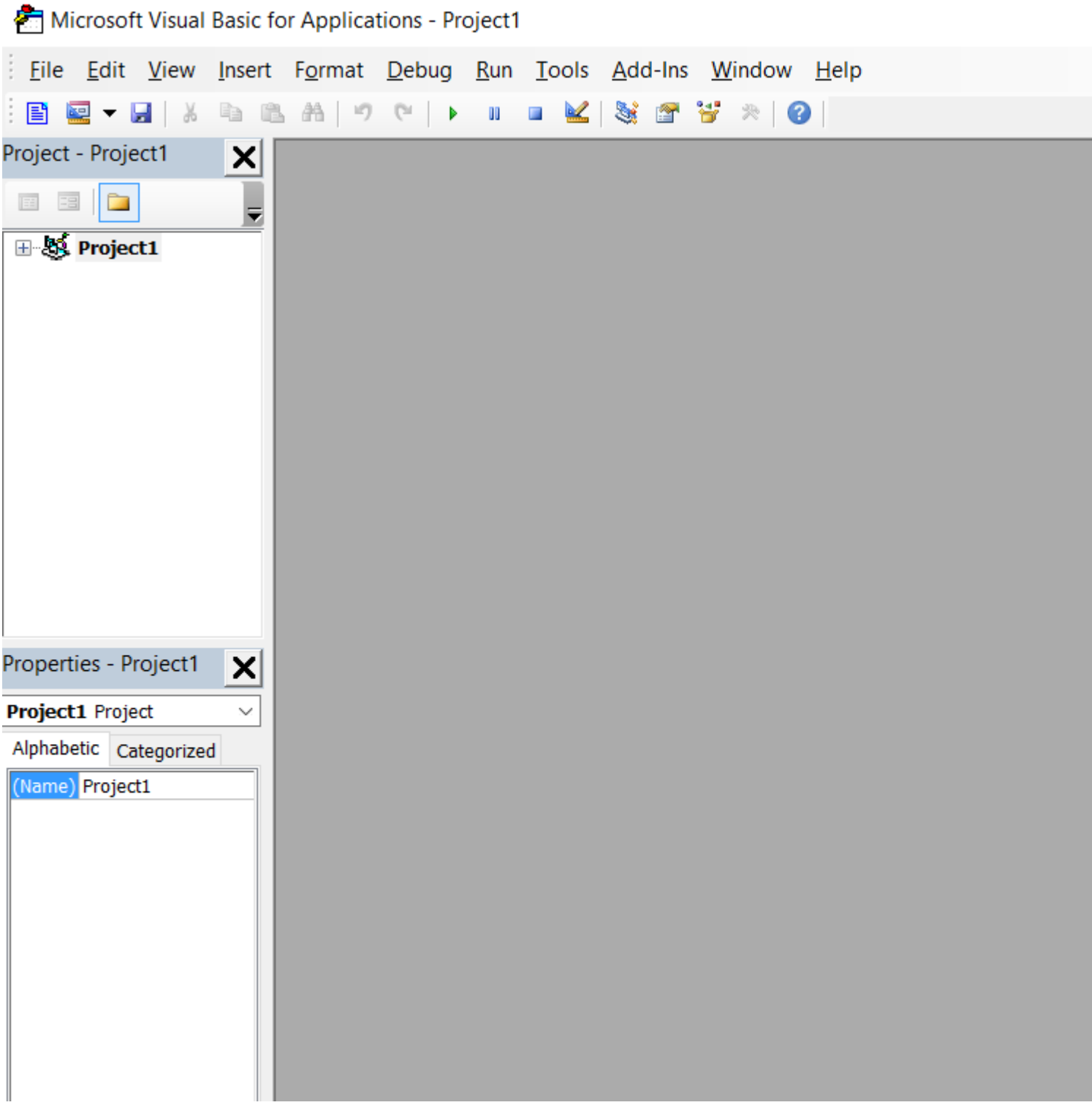
Haga clic en la nueva pestaña `Desarrollador` para obtener:



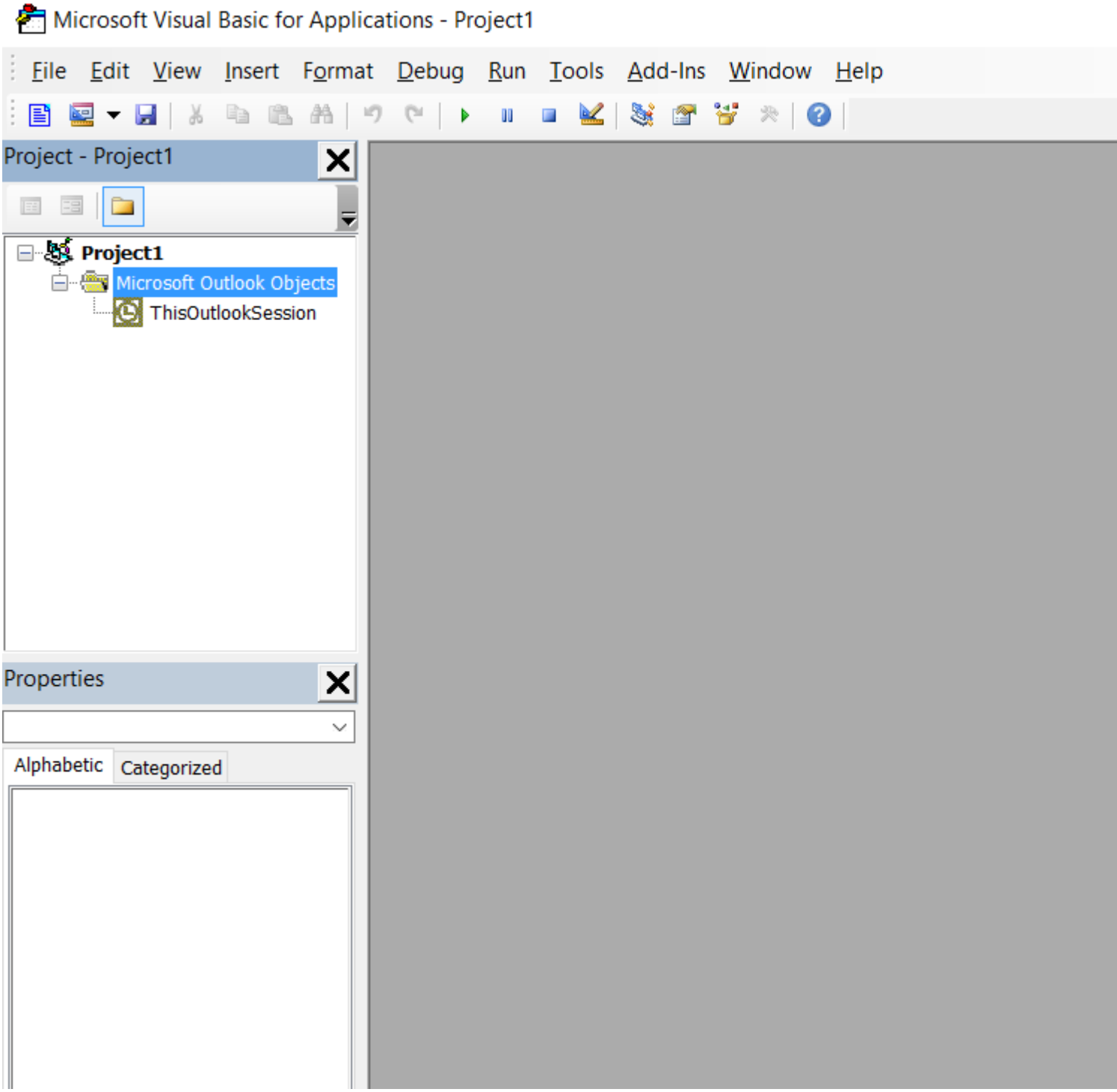
Haga clic en `Visual Basic` , a la izquierda, para seleccionar el Editor de Visual Basic.

1.3 Comenzando con el Editor de Visual Basic

Las imágenes en esta sección son todas de Outlook 2016, pero podrían haber venido de Outlook 2003. El VBA de Outlook puede haber cambiado con los años, pero a mi parecer el Editor de VBA no lo ha hecho. Cualquiera que sea la versión que tengas, verás algo como:

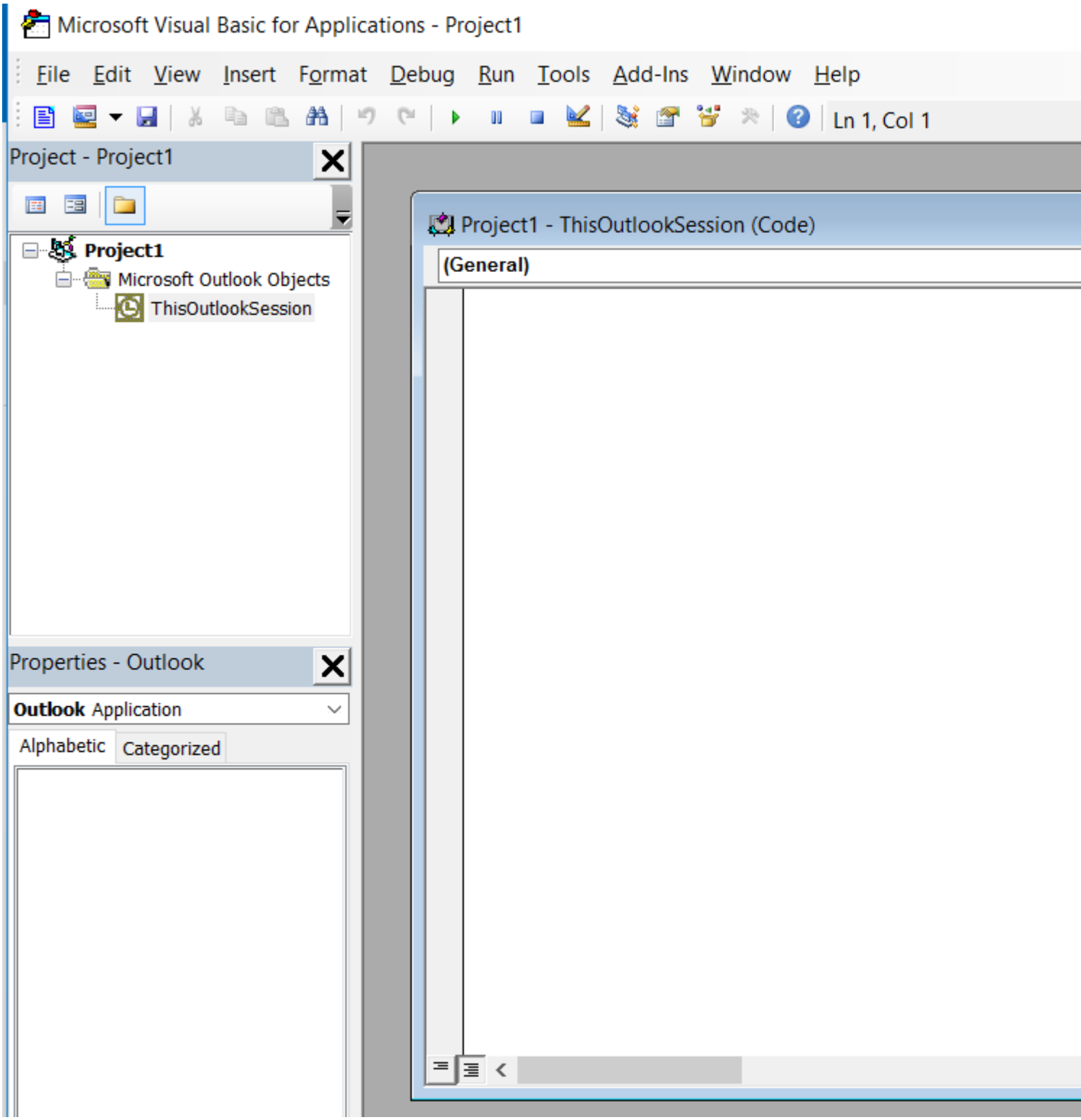


Arriba hay un "+" contra " **Project1** ". Si tiene un “+”, haga clic en él y luego el “+” en "Objetos de Microsoft Outlook" para obtener:



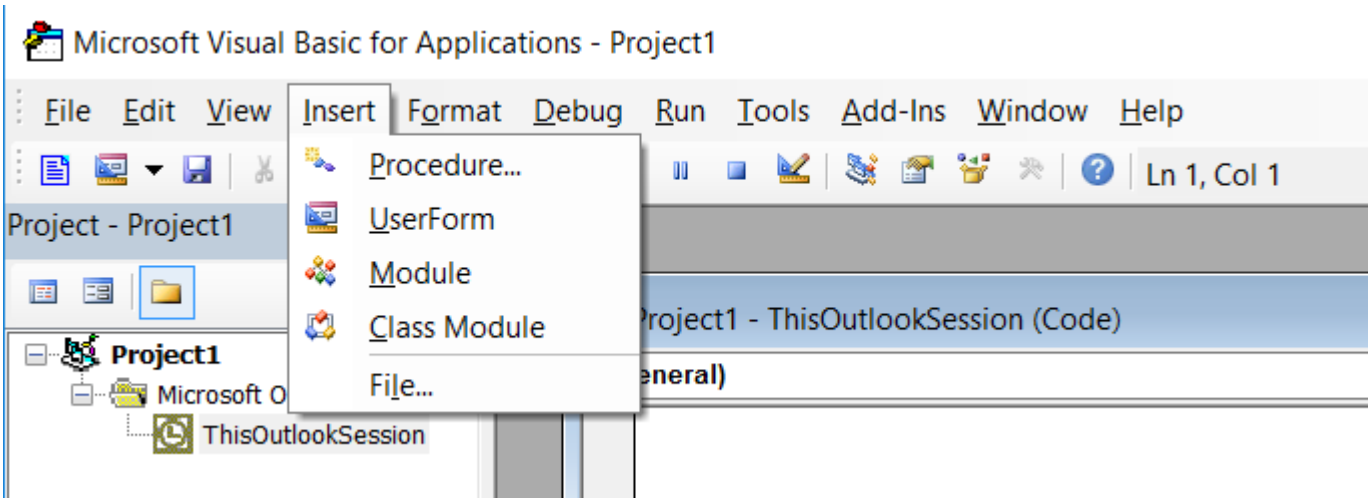
Es posible que la ventana Propiedades no esté presente o, si está presente, puede ubicarse en otro lugar dentro de la ventana del Editor VB. No lo necesitamos por el momento. Puede cerrarla haciendo clic en la cruz y puede usar `F4` para hacerla visible nuevamente en cualquier momento. Normalmente no lo tengo visible porque no necesito acceder a las propiedades la mayor parte del tiempo y mi lista del Explorador de proyectos ocupa la mayor parte del lado izquierdo. Te sugiero que lo mantengas visible hasta que se convierta en una molestia.

Si hace clic en `ThisOutlookSession` , el área gris se volverá blanca o, como en la imagen de abajo, aparecerá una ventana de código dentro del área gris:

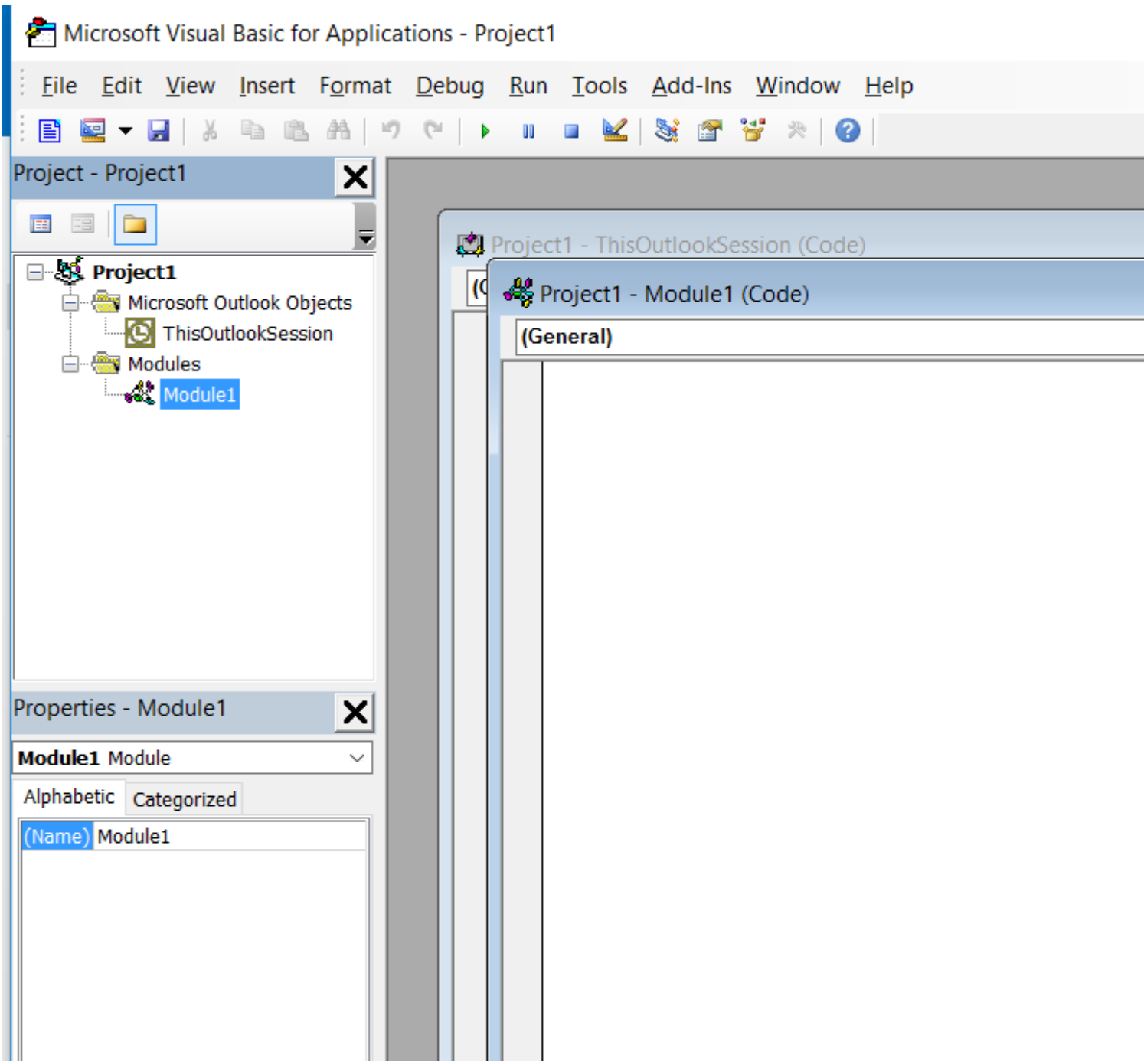


Puede escribir cualquier código en esta ventana de código. Sin embargo, las rutinas de eventos (que se discuten hacia el final de este tutorial) deben escribirse en esta ventana de código. Le recomiendo que reserve el área de código de ThisOutlookSession para las rutinas de eventos.

En su lugar, haga clic en `Insertar` para obtener:



Haga clic en `Módulo` para agregar un módulo:



Mi nuevo módulo se llama "Module1". Si su versión de Outlook es una versión no inglesa, su módulo tendrá un nombre equivalente en su idioma. Puede agregar más módulos que se llamarán "Module2", "Module3" y así sucesivamente.

Si estoy creando un libro de Excel, para el cual solo necesito un módulo, podría dejar el nombre como "Módulo1". Pero con Outlook, todas mis macros tienen que ir aquí, así que tengo muchos módulos. A lo largo de los años he escrito muchas rutinas que reutilizo repetidamente. Tengo un módulo para rutinas generales de VBA, otro para rutinas para acceder a Excel, otro para rutinas de Outlook VBA y luego un módulo por tarea de Outlook que realizo con macros. Si observa la ventana Propiedades, verá que la única propiedad de un módulo es su nombre. Haga clic en el "Módulo 1" en contra de "Nombre" y puede cambiarlo por cualquier nombre válido (comienza con una letra, contiene letras y números, etc.). Obtiene errores extraños si un módulo y un procedimiento tienen el mismo nombre, así que comienzo todos los nombres de mis módulos con "Mod" y no uso este prefijo para mis procedimientos. ¿Por qué no renombrar este módulo "ModIntro" o similar listo para la siguiente parte de este tutorial?

Estas áreas de código y como las áreas de entrada de datos de cualquier editor. Haga clic en el área de código para seleccionarlo y escriba su código o pegue el código copiado de otro lugar, como la siguiente sección de este tutorial.

1.4 Lo que debes recordar de esta parte del tutorial

- ¿Tu versión de Outlook necesitaba que agregaras la pestaña Desarrollo? Si es así, no necesitará repetir este proceso hasta que tenga una nueva instalación de Outlook. Vuelve aquí cuando eso suceda.
- Recuerda cómo entrar al editor de Visual Basic.
- Recuerda cómo crear y renombrar un módulo.

Lea Introducción Parte 1: Obtener acceso al Editor de Visual Basic de Outlook en línea: https://riptutorial.com/es/outlook-vba/topic/8877/introduccion-parte-1--obtener-acceso-al-editor-de-visual-basic-de-outlook

Capítulo 3: Introducción Parte 2: Tiendas y carpetas de nivel superior.

Introducción

Primera parte de una introducción a las tiendas y las carpetas que contienen. Contiene macros para mostrar (1) los nombres de las tiendas accesibles y (2) los nombres de las tiendas accesibles y las carpetas de nivel superior dentro de ellas.

Examples

2.1 Conocimiento previo esperado

- Usted es un usuario de Outlook y comprende términos como "correo electrónico", "hora recibida", "asunto" y "Panel de carpetas".
- Sabes cómo acceder al Editor de Visual Basic de Outlook y crear un módulo. Ver Introducción Parte 1 si es necesario.
- Tienes al menos un conocimiento básico de VBA. Declaro las subrutinas y las variables sin explicación. Uso Withs, Ifs y Loops sin explicación. Te digo que algo es una colección. Le digo que copie el código a un módulo y lo ejecute. Hay muchos tutoriales en línea, aunque la mayoría son para Excel VBA y se concentran más en usar el lenguaje con Excel que en el lenguaje. Al buscar el "tutorial de VBA" se muestran algunos que se concentran en el idioma más que en la aplicación que parece satisfactoria.
- No es necesario que conozca el modelo de objetos de Outlook; Este tutorial te presenta una pequeña parte de él.

2.2 tiendas

Outlook almacena correos electrónicos, elementos de calendario, notas, tareas, etc. en archivos conocidos como **Tiendas** . Si miras el panel de carpetas, verás algo como:

```
Aaaaaaaaaa
  Inbox
  Drafts
  Deleted Items
  :  :

Bbbbbbbbbb
  Inbox
  Drafts
  Deleted Items
  :  :

Ccccccccc
:  :
```

"Aaaaaaaaaa", "Bbbbbbbbbb" y "Ccccccccccc" son los nombres de usuario o de visualización de las tiendas. Siempre he aceptado los valores predeterminados de Outlook para estos nombres que han cambiado con los años. Una vez que el valor predeterminado era mi nombre ahora es mi dirección de correo electrónico. El nombre de archivo para estas tiendas puede ser el mismo pero con una extensión como PST u OST o puede ser algo completamente diferente. Una macro de VBA necesita el nombre de usuario para acceder a una tienda y no se preocupa por los nombres de archivo o la extensión.

Puedes tener tantas tiendas como desees. Tengo el "archivo de datos de Outlook" que se creó cuando instalé Outlook. Cuando agregué cuentas para mis direcciones de correo electrónico, Outlook creó nuevas tiendas con el nombre de la dirección de correo electrónico como "JohnDoe@hotmail.com" y "DoeJohn@gmail.com". Para reducir el tamaño de mi tienda principal, guardo correos electrónicos antiguos en tiendas con nombres como "Archivo 2015".

Si es un usuario comercial, puede tener acceso a tiendas compartidas o a tiendas de colegas.

Las macros a continuación muestran tres formas diferentes de enumerar las tiendas a las que puede acceder. Le sugiero que cree un nuevo módulo para mantener el código a continuación y que use `F4` para acceder a las propiedades del módulo, de modo que pueda llamarlo "ModIntro" o algún otro nombre de su elección. Si completó la Parte 1 de esta serie, ya tendrá dicho módulo.

Copie estas macros en un módulo y pruebe que cada una de ellas dé el mismo resultado.

```
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
```

Encontrará con VBA que a menudo hay varios métodos para lograr el mismo efecto. Arriba he mostrado tres métodos de acceso a las tiendas. No es necesario que los recuerde todos, elija su propio favorito, pero debe tener en cuenta que existen varios métodos porque otras personas, cuyo código debe estudiar, tendrán diferentes favoritos.

Las variables `StoresColl` en las macros `ListStores1()` y `ListStores2()` son colecciones pero tienen diferentes tipos de objetos: `Store` y `Folder` . Un objeto de `Store` solo puede hacer referencia a un archivo en su disco. Una `Folder` puede hacer referencia a un archivo en el disco, pero también puede hacer referencia a carpetas dentro de una tienda como "Bandeja de entrada" y "Elementos enviados". `Stores` , las `Folders` , la `Store` y la `Folder` forman parte del modelo de objetos de Outlook. Esta serie de tutoriales le presenta el modelo, pero no es una definición formal. Si desea una definición formal, escriba "outlook vba object model" en su motor de búsqueda favorito. Asegúrese de mirar la versión VBA del modelo.

2.3 Carpetas de nivel superior

En el ejemplo de mi Panel de carpetas anterior, solo enumero tres carpetas estándar: "Bandeja de entrada", "Borradores" y "Elementos eliminados". Hay otras carpetas estándar y puede crear tantas carpetas como desee. Algunas personas crean carpetas en la Bandeja de entrada, pero prefiero crear nuevas carpetas al mismo nivel que la Bandeja de entrada. Sus carpetas pueden tener subcarpetas que pueden tener sus propias subcarpetas a cualquier profundidad.

La siguiente macro producirá una lista del formulario:

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

donde A, B y C son tiendas y A1, B1, C1, etc., son carpetas dentro de A, B y C. Si A1, B1, C1 y demás tienen subcarpetas, esta macro no las incluirá en la lista. El acceso a carpetas más profundamente anidadas se tratará en la siguiente parte de este tutorial.

```
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 Lo que debes recordar de este tutorial

- Una tienda es un archivo en el que Outlook almacena correos electrónicos, elementos del calendario, notas, tareas, etc.
- Una tienda puede contener carpetas estándar de Outlook como "Bandeja de entrada" y "Elementos enviados".
- Una tienda también puede contener carpetas creadas por el usuario.
- Tanto las carpetas estándar de Outlook como las carpetas creadas por el usuario pueden contener subcarpetas creadas por el usuario, subcarpetas, etc., a cualquier profundidad.
- Cómo listar tiendas.
- Cómo enumerar las tiendas y las carpetas de nivel superior dentro de esas tiendas.

Confesión: No recuerdo ninguno de los "Hows". Tengo subrutinas y funciones que recuerdo para mi.

Lea Introducción Parte 2: Tiendas y carpetas de nivel superior. en línea: https://riptutorial.com/es/outlook-vba/topic/8876/introduccion-parte-2--tiendas-y-carpetas-de-nivel-superior-

Capítulo 4: Introducción Parte 3: Tiendas y todas sus carpetas.

Introducción

Completa la introducción a las tiendas y carpetas iniciada en la parte 2 de este tutorial.

Conocimiento previo esperado : usted ha estudiado la parte 2 de este tutorial o ya está familiarizado con su contenido.

Examples

3. 0 contenidos

- Cómo hacer referencia a cualquier carpeta accesible.
- Cómo obtener el nombre completo de una carpeta de referencia.
- Un par de rutinas que juntas enumerarán cada carpeta dentro de cada tienda accesible.
- Una rutina para mover una carpeta de una carpeta principal a otra.

3.1 Función GetFldrNames () que se necesita para varias de las macros de demostración

Algunas de las macros de demostración dentro de esta parte requieren una función que explicaré más adelante. Por el momento, simplemente copie `GetFldrNames()` en un módulo adecuado. Utilizo esta función con frecuencia y la guardo, y otra similar, que uso en muchas macros diferentes, en un módulo llamado "ModGlobalOutlook". Puede que quieras hacer lo mismo. Alternativamente, puede preferir mantener la macro con todas las demás macros dentro de esta serie de tutoriales; Puedes moverlo más tarde si cambias de opinión.

```
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 Referencia a una carpeta por defecto

En `TestDefaultFldr()` configuré `Fldr` en la Bandeja de entrada predeterminada. La constante `olFolderInbox` se puede reemplazar por otros valores que dan acceso a cualquiera de las carpetas predeterminadas. Si escribe `Set Fldr = Session.GetDefaultFolder(` , el editor de VB mostrará una lista desplegable de todos los valores posibles.

```
Sub TestDefaultFldr()

  Dim Fldr As Folder

  Set Fldr = Session.GetDefaultFolder(olFolderInbox)

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

End Sub
```

En mi computadora portátil, `TestDefaultFldr()` muestra `Outlook data file|Inbox` que fue una sorpresa. Escribí `GetFldrNames(Fldr)` para asegurarme de que la carpeta a la que hacía referencia era la que quería. ¡Había accedido a la bandeja de entrada predeterminada y encontré que estaba vacía! El "archivo de datos de salida" de la tienda vino con la instalación predeterminada y lo había ignorado desde que Outlook había creado una tienda para cada una de mis cuentas de correo electrónico. Solo después de descubrir mi Bandeja de entrada predeterminada vacía,

pensé en cómo Outlook sabría cuál de mis cuentas de correo electrónico era la cuenta que querría como predeterminada. De las carpetas estándar de Outlook, no hay un valor predeterminado o el valor predeterminado está dentro de "Archivo de datos de salida". Puede ser posible cambiar qué Bandeja de entrada es la Bandeja de entrada predeterminada, pero no he investigado porque no estoy seguro de cuál de mis cuentas de correo electrónico haría la predeterminada si cambiara. Solo recuerde que todos los elementos de su calendario, tareas, etc. están dentro del "archivo de datos de Outlook" y asegúrese de incluir "Outlook.pst" en su lista de archivos.

La mayoría de los objetos de Outlook tienen la propiedad `Parent` . `GetFldrNames(Fldr)` registra el nombre de la carpeta en una matriz antes de intentar acceder a su padre. Hace un bucle para agregar nombres al final de la matriz hasta que llega a la tienda. La tienda no tiene un padre, por lo que el intento de acceder falla. La secuencia de nombres en la matriz se invierte y luego se devuelve a la persona que llama. He usado `Join` para convertir el conjunto de nombres en una cadena visualizable.

3.3 Referencia a cualquier carpeta dentro de cualquier tienda accesible

`TestFldrChain()` muestra cómo hacer referencia a cualquier carpeta dentro de cualquier tienda accesible:

```
Sub TestFldrChain()

  Dim Fldr As Folder

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

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

End Sub
```

En `TestFldrChain()` : A es el nombre de una tienda; A2 es el nombre de una carpeta dentro de A; A21 es el nombre de una carpeta dentro de A2 y A213 es el nombre de una carpeta dentro de A21.

¿Que está sucediendo aquí?

`Session` tiene una propiedad `Folders` que es una lista de todas las tiendas accesibles.

`Session.Folders(integer)` , que usé en la Parte 2 de este tutorial, me permite recorrer las tiendas en secuencia cuando no sé sus nombres. `Session.Folders("A")` me permite acceder a una carpeta cuando sé su nombre.

`Session.Folders("A")` es una carpeta y también tiene una propiedad `Folders` .

`Session.Folders("A").Folders("A2")` me dan acceso a la carpeta "A2" dentro de la tienda "A".

Puedo encadenar tantas `Folders("x")` como sea necesario para llegar a cualquier carpeta. Si la cadena es demasiado larga para una línea, puede dividir la declaración en varias líneas como lo

he hecho yo.

Busque la carpeta más anidada dentro de su instalación y reemplace A, A2, A21 y A213 por los nombres de su tienda y carpetas. Aumente o disminuya el número de carpetas en la cadena según sea necesario.

Si actualiza y ejecuta `TestFldrChain()` , se mostrará lo siguiente, excepto que A, A2, etc., serán reemplazados por los nombres de sus carpetas:

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

3.4 Listado de los nombres de cada carpeta dentro de cada tienda accesible

En la Parte 2, se le mostró cómo enumerar cada tienda accesible y las carpetas de nivel superior dentro de cada tienda. Esto implicó un bucle a través de las tiendas y luego un bucle para cada tienda a través de sus carpetas. Anteriormente, ha visto cómo hacer referencia a una carpeta conocida a cualquier profundidad dentro de la jerarquía de carpetas. Esto implicó encadenar tantas `Folders("x")` como sea necesario para llegar a la carpeta.

Ahora quiero enumerar cada carpeta, a cualquier profundidad, dentro de cada tienda. La técnica de codificación más fácil para resolver este tipo de problema en el que debe bajar cadenas de longitudes variables es la **recursión** . Si usted es un programador serio en otro lenguaje o herramienta, es posible que ya sepa sobre la recursión. Si tiene ambiciones de ser un programador serio, necesitará comprender la recursión eventualmente, pero no necesariamente hoy. La "recursión" es uno de esos conceptos que muchos encuentran difíciles de entender al principio. Puede escribir "Recursión" en su motor de búsqueda favorito y leer los distintos intentos de explicar este concepto. Alternativamente, puede aceptar estos trabajos de macro, pero no preocuparse de cómo funcionan.

Tenga en cuenta el comentario en `ListStoresAndAllFolders()` : estas macros necesitan una referencia a "Microsoft Scripting Runtime". Haga clic en `Herramientas` en la barra de pestañas en la parte superior de la ventana Editor VB y luego haga clic en `Referencias` . Obtendrá una lista de todas las referencias disponibles (bibliotecas). Algunos en la parte superior ya estarán marcados. El resto está en orden alfabético. Desplácese hacia abajo en la lista y haga clic en el cuadro a la izquierda de "Microsoft Scripting Runtime" para obtener una marca. A continuación, haga `clic` en `Aceptar`

```
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
```

Después de ejecutar `ListStoresAndAllFolders` , habrá un nuevo archivo en su DeskTop llamado "ListStoresAndAllFolders.txt" que contendrá la lista prometida de tiendas y carpetas.

3.5 Mover una carpeta de una carpeta principal a otra

¿Por qué quiero hacer referencia a una carpeta? En la siguiente parte, le mostraré cómo acceder a los correos electrónicos dentro de una carpeta referenciada. Aquí te mostraré cómo mover una carpeta. Creé una carpeta llamada "Prueba" dentro de mi bandeja de entrada. En `TestMoveFolder()` , reemplacé "A" con el nombre de la tienda que contiene mi Bandeja de entrada. La ejecución de `TestMoveFolder()` movió "Prueba" a "Elementos eliminados".

```
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 Lo que debes recordar de esta parte del tutorial

- Cómo hacer referencia a una carpeta predeterminada y las posibles limitaciones de esta técnica.
- Cómo hacer referencia a cualquier carpeta individual a cualquier profundidad dentro de cualquier tienda accesible.
- Cómo mostrar el nombre completo de una carpeta referenciada.
- Cómo hacer referencia a una de las muchas, muchas bibliotecas disponibles que proporcionan funcionalidad más allá del conjunto predeterminado de subrutinas y funciones.
- Cómo mostrar el nombre de cada carpeta dentro de cada tienda accesible.
- Cómo mover una carpeta de una carpeta padre a otra.

Lea Introducción Parte 3: Tiendas y todas sus carpetas. en línea: https://riptutorial.com/es/outlook-vba/topic/8874/introduccion-parte-3--tiendas-y-todas-sus-carpetas-

Creditos

S. No	Capítulos	Contributors
1	Empezando con Outlook-VBA	Community, Tony Dallimore
2	Introducción Parte 1: Obtener acceso al Editor de Visual Basic de Outlook	Tony Dallimore
3	Introducción Parte 2: Tiendas y carpetas de nivel superior.	Tony Dallimore
4	Introducción Parte 3: Tiendas y todas sus carpetas.	Tony Dallimore