Comencemos a programar con

VBA - Access

Entrega 17

Trabajar con ficheros

Eduardo Olaz

Trabajar con Ficheros

Desde VBA podemos acceder no sólo a las tablas de nuestras bases de datos, sino también leer y escribir en ficheros de texto e incluso otros tipos de formato, como ficheros de XML, XSL o HTML.

Tenemos a nuestra disposición una serie de procedimientos, e incluso objetos, que nos brindan esa posibilidad.

Además podemos buscar ficheros, borrarlos, crearlos, crear carpetas y otras operaciones estándar.

Trabajar con carpetas

Cuando desarrollamos una aplicación, debemos tener muy claro en qué carpeta estamos haciendo el desarrollo y dónde van a estar los ficheros de datos.

Por ejemplo, si tenemos una aplicación monopuesto, podríamos tener el fichero del programa en la carpeta `C:\MiPrograma`, y los datos en la carpeta `C:\MiPrograma\Datos`.

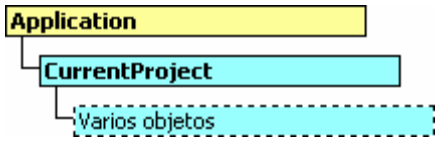
Si distribuimos la aplicación es muy probable que el cliente la instale en otra carpeta.

En ese caso los enlaces que tenemos previstos no funcionarían.

Esto nos lleva a la conclusión de que es clave el que podamos controlar en qué carpeta y unidad de disco está instalada la aplicación.

Este dato lo podemos obtener mediante el objeto **CurrentProject**.

Este objeto pertenece a su vez al objeto **Application** y tiene toda una serie de métodos, colecciones, objetos y métodos, que estudiaremos en una entrega posterior.



Los métodos que ahora nos interesan son **`Path`**, **`Name`** y **`FullName`**.

Veamos el siguiente código:

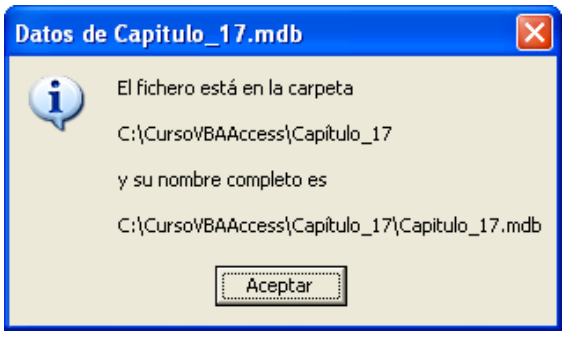
```
Public Function CarpetaActual() As String
    CarpetaActual = CurrentProject.Path
End Function

Public Function NombreBaseDatos() As String
    NombreBaseDatos = CurrentProject.Name
End Function

Public Function NombreCompletoBaseDatos() As String
    NombreCompletoBaseDatos = CurrentProject.FullName
End Function
```

Tras definir estas funciones podemos utilizarlas, por ejemplo para mostrar los datos en un cuadro de mensaje.

```
Public Sub PropiedadesBaseDatos()
```

```
    Dim strMensaje As String
    strMensaje = "El fichero está en la carpeta " _
            & vbCrLf & vbCrLf _
            & CarpetaActual() _
            & vbCrLf & vbCrLf _
            & "y su nombre completo es" _
            & vbCrLf & vbCrLf _
            & NombreCompletoBaseDatos
    MsgBox strMensaje, _
            vbInformation + vbOKOnly, _
            "Datos de " & NombreBaseDatos()
End Sub
```

Tras esto, si ejecuto el procedimiento **PropiedadesBaseDatos** en mi ordenador, me muestra el siguiente mensaje:



De esta manera, si los datos estuvieran en el fichero Datos.mdb ubicado en una carpeta llamada Datos, que cuelgue de la carpeta de la aplicación, podríamos definir una función llamada FicheroDatos que nos devuelva la ruta completa del fichero.

Podría ser algo así como:

```
Public Function FicheroDatos() As String
    Const conBaseDatos As String = "Datos.mdb"
    FicheroDatos = CarpetaActual() _
            & "\Datos\" & conBaseDatos
End Function
```

Si ejecuto en mi ordenador esta función, me devuelve la cadena

```
C:\CursoVBAAccess\Capítulo_17\Datos\Datos.mdb
```

Otro paso interesante sería comprobar si existen la carpeta y el fichero en nuestro disco.

Una de las formas de averiguarlo sería utilizando la función **Dir()**.

Función Dir

La función **Dir()**, nos devuelve un valor de tipo **String** que representa el nombre de un archivo, o carpeta que coincide con el patrón o atributo de archivo que se le ha pasado como parámetro. También nos puede devolver la etiqueta de volumen de una unidad de disco.

Su sintaxis es **Dir**[**(***NombreDeRuta*[, *Atributos*]**)**]

NombreDeRuta es una expresión de cadena que indica la ruta donde queremos buscar y el tipo de fichero que nos interesa.

Además de la ruta podemos especificar características que tenga el nombre del fichero, mediante el uso de los comodines **"*"** y **"?"**. Por ejemplo para obtener los ficheros de cualquier tipo que empezaran por la letra A, en la carpeta "C:\Graficos\" podríamos escribir **Dir**("C:\Graficos**A*.***").

El símbolo **"*"** sustituye a cualquier número de caracteres.

El símbolo sustituye a 1 carácter, o ninguno.

Por ejemplo, **Dir**("C:\Graficos**????.***") nos devolvería nombres de ficheros con hasta cuatro caracteres en el nombre, y con cualquier tipo de extensión.

Dir("C:\Graficos**?a??.***") nos devolvería nombres con hasta cuatro caracteres en el nombre, que tuvieran la letra a como segundo carácter en el nombre , y con cualquier tipo de extensión. Estos nombres de fichero serían válidos para este último caso

g**a**to.bmp
b**a**rt.jpg

Para obtener todos los nombres de fichero de una carpeta que cumplan determinada condición, la primera llamada se efectúa con el nombre de la ruta y sus atributos.

Las siguientes llamadas se hacen únicamente con **Dir()** devolviendo el nombre de los sucesivos ficheros.

Mientras existan ficheros que cumplan la condición, **Dir()** devolverá su nombre. Cuando deje de haberlos, devolverá la cadena vacía **""**.

Atributos es una constante. Los valores que puede tomar son:

vbNormal	(Predeterminado) Especifica archivos sin atributos.
vbReadOnly	Sólo lectura y sin atributos
vbHidden	Archivos ocultos y sin atributos
VbSystem	Del sistema y sin atributos
vbVolume	Etiqueta del volumen
vbDirectory	Carpetas y archivos sin atributos

Si por ejemplo quisiéramos obtener todos los ficheros que fueran del tipo **Gif** y que estuvieran en la carpeta **C:\Graficos\"**, podríamos hacer:

```
Public Sub FicherosGif()
    Dim colFicheros As New Collection
    Dim strCarpeta As String
    Dim strFichero As String
    Dim i As Long

    strCarpeta = "C:\Graficos\"

    strFichero = Dir(strCarpeta & "*.gif")
    If strFichero = "" Then
```

```
            Exit Sub
        Else
            Do Until strFichero = ""
                colFicheros.Add strFichero
                strFichero = Dir()
            Loop
        End If
        For i = colFicheros.Count To 1 Step -1
            Debug.Print colFicheros(i)
            colFicheros.Remove (i)
        Next i
        Set colFicheros = Nothing
    End Sub
```

En este ejemplo busca los ficheros del tipo gif, en la carpeta `"C:\Graficos\"` mediante la expresión

```
Dir(strCarpeta & "*.gif")
```

Si en la primera llamada encuentra alguno, `strFichero` será diferente a la cadena vacía, añade el resultado a la colección `colFicheros`.

Repite el bucle con **`Dir()`** y va añadiendo el resultado a la colección, hasta que devuelva la cadena vacía.

En ese momento hace un bucle que va recorriendo los diferentes elementos de la colección, desde el último hasta el primero, los muestra en la ventana Inmediato y los descarga de la colección.

Al final elimina el objeto **`colFicheros`** asignándole el valor **`Nothing`**.

Si en el primer **`Dir`** no hubiéramos indicado la carpeta donde queremos buscar, por ejemplo escribiendo

```
Dir("*.gif")
```

El procedimiento hubiera mostrado los posibles ficheros **`gif`** que existieran en la ruta de acceso actual. Una de las carpetas habituales suele ser la de **`"Mis documentos"`**.

El valor de esta carpeta la podemos obtener mediante la función **`CurDir`**.

Mediante la función **`Dir`**, podemos también obtener el nombre del volumen, usando la letra de la unidad y la constante **`vbVolume`**.

Por ejemplo:

```
Dir("C:",vbVolume)
Dir("E:",vbVolume)
```

Función CurDir

La función **`CurDir()`**, nos devuelve un valor de tipo **`String`** que representa el nombre de la ruta de acceso actual.

Su sintaxis es **`CurDir`**`[(`*`Unidad`*`)]`

Podemos usar el nombre de la unidad de la que queremos obtener la ruta de acceso.

CurDir("C") **CurDir("D")** **CurDir("F")**

Si no se especifica la unidad de disco o el parámetro ***unidad*** es la cadena vacía (**""**), la función **CurDir** devuelve la ruta de acceso de la unidad de disco actual.

Podemos establecer desde VBA la ruta actual deseada.

Para ello se utiliza la Instrucción **ChDir**.

Instrucción ChDir

La instrucción **ChDir**, establece la carpeta actual. Para ello se le pasa como parámetro la carpeta deseada.

Su sintaxis es **ChDir Ruta**

Si la ruta no existiera, generaría el error 76

"No se ha encontrado la ruta de acceso"

Para cambiar la carpeta actual a "C:\Programa\Datos".

```
ChDir "C:\Programa\Datos"
```

ChDir no cambia la unidad de disco actual, sólo la carpeta del disco especificado

La siguiente instrucción cambia la carpeta de D:

```
ChDir "D:\Comercial\Datos"
```

Si la unidad seleccionada hasta ese momento era la **C**, seguirá siéndolo después de esta última instrucción.

Si en la ruta no se especifica la unidad, se cambia el directorio o carpeta predeterminado de la unidad actual.

Se puede hacer que la carpeta actual sea la de un nivel superior pasándole dos puntos.

```
ChDir "C:\Programa\Datos"
```

Tras esto la carpeta actual es "C:\Programa\Datos\".

```
ChDir ".."
```

Tras esto la carpeta actual pasa a ser "C:\Programa\".

Si quisiéramos cambiar la unidad de disco actual, debemos utilizar la instrucción **ChDrive**.

Instrucción ChDrive

La instrucción **ChDrive**, establece la unidad de disco actual.

Su sintaxis es **ChDrive Unidad**

```
ChDrive "D"
```

Si no existiera, o no estuviera disponible la unidad pasada como parámetro, generaría el error 68

"Dispositivo no disponible"

A veces podemos vernos en la necesidad de crear una carpeta nueva.

Para ello tenemos la instrucción **MkDir**.

Instrucción MkDir

La instrucción **MkDir**, crea una carpeta.

Su sintaxis es **MkDir Ruta**

```
MkDir "C:\MiNuevaCarpeta"
MkDir "C:\MiNuevaCarpeta\Subcarpeta_1"
MkDir "C:\MiNuevaCarpeta\Subcarpeta_2"
```

Tras esto tendremos la carpeta `C:\MiNuevaCarpeta` con dos nuevas carpetas `Subcarpeta_1 y Subcarpeta_2.`

Instrucción RmDir

La instrucción **RmDir**, elimina una carpeta existente.

Su sintaxis es **RmDir Ruta**

Tras crear las carpetas del punto anterior podríamos eliminarlas utilizando

```
RmDir "C:\MiNuevaCarpeta\Subcarpeta_1"
RmDir "C:\MiNuevaCarpeta\Subcarpeta_2"
RmDir "C:\MiNuevaCarpeta"
```

Para eliminar una carpeta con **RmDir** es preciso que ésta esté vacía, es decir: no debe tener ningún archivo ni otra subcarpeta. Caso contrario generaría el error 75

```
"Error de acceso a la ruta o el archivo"
```

Instrucción Kill

La instrucción **Kill**, elimina un archivo existente.

Su sintaxis es **Kill Ruta**

El argumento requerido **Ruta** es una cadena que especifica el nombre o los nombres de los archivos que se van a eliminar.

Puede incluir la ruta completa del fichero.

Al igual que la función **Dir**, también admite utilizar los comodines asterisco "*****" y "**?**".

Por ejemplo, si quisiéramos borrar todos los ficheros **gif** de la carpeta actual, usaríamos

```
Kill "*.gif"
```

Si quisiéramos borrar todos los ficheros de la carpeta actual, usaríamos

```
Kill "*.*"
```

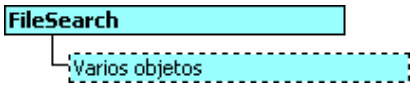
Si quisiéramos borrar todos los archivos que empezaran por **Tmp** :

```
Kill "Tmp*.*"
```

Si quisiéramos borrar todos los ficheros cuyo nombre empiece por **A** y tengan hasta 4 caracteres en el nombre.

```
Kill "A???.*"
```

El objeto FileSearch



Además de los procedimientos vistos con anterioridad, VBA incorpora un objeto que nos puede suministrar una información más completa que los procedimientos vistos en los puntos anteriores. Es el objeto **FileSearch**.

El objeto **FileSearch** es devuelto por el objeto **Application**, a través de su propiedad **FileSearch**.

Propiedades y métodos de FileSearch

Propiedad LookIn

La propiedad **LookIn** nos permite averiguar o establecer la carpeta en donde se va a efectuar la búsqueda de los ficheros. Es por tanto de lectura y escritura.

Propiedad Filename

La propiedad **Filename** devuelve o establece el nombre de los ficheros que se van a buscar. Como en el caso de la función Dir y del procedimiento Kill vistos en los puntos anteriores, se pueden utilizar caracteres comodín **"*"** y **"?"**.

Propiedad SearchSubFolders

La propiedad **SearchSubFolders** (de tipo **Boolean**) devuelve o establece si se va a buscar, además de en la carpeta especificada por la propiedad **LookIn**, en las carpetas que “cuelguen” de ella.

Método Execute

El método **Execute** devuelve el valor **0** si no se ha encontrado ningún fichero con las características buscadas, y un **Long** positivo caso de encontrarse.

Su sintaxis es :

ObjetoFileSearch*.Execute**(SortBy, SortOrder, AlwaysAccurate***)

El método **Execute** admite tres parámetros con los que se puede establecer la forma de ordenar los resultados:

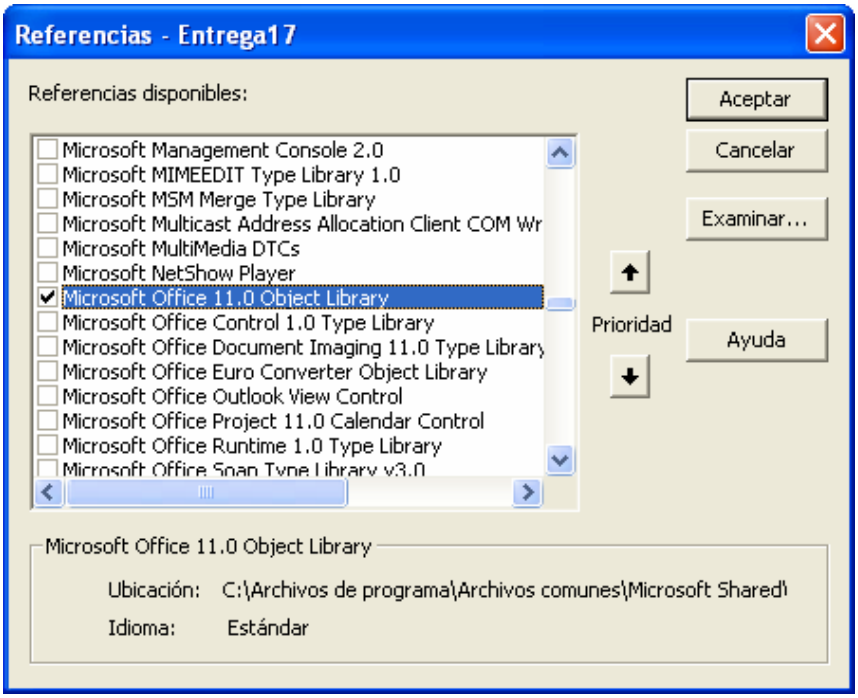
SortBy especifica el método utilizado para la ordenación de los datos obtenidos.

Nombre de la constante	Valor
msoSortByFileName *predeterminado*	**1**
msoSortByFileType	**3**
msoSortByLastModified	**4**
msoSortByNone	**5**
msoSortBySize	**2**

Nota:

Aunque FileSearch puede funcionar sin activar desde Access la referencia a la librería “Microsoft Office NN.N Object Library” si no se hace referencia explícita a ella, no se tiene acceso a las constantes propias de **FileSearch**, así como a la declaración de ciertas variables. Por ello es aconsejable activar la referencia a esa librería.

Para hacerlo, desde la opción de menú [Herramientas] se selecciona la opción [Referencias] y se activa la opción correspondiente a la Biblioteca. En mi ordenador queda así:



Para poder trabajar con **FileSearch**, aunque no se haya efectuado una referencia a la biblioteca mencionada, y que se identifique el valor de las constantes, en esta y sucesivas tablas pongo su nombre y su valor.

SortOrder especifica el orden de los datos obtenidos.

Nombre de la constante	Valor
msoSortOrderAscending *predeterminado*	**1**
msoSortOrderDescending	**2**

AlwaysAccurate Un valor **boolean** que especifica si se van a incluir los nombres de los ficheros que hayan sido agregados, modificados o eliminados desde que se actualizó por última vez el índice. Su valor predeterminado es True.

Este ejemplo busca los ficheros gráficos del tipo **jpg** en la carpeta **"C:\Gráficos"**.

Los almacenará en la colección **FoundFiles** según su nombre y en orden ascendente.

```
Public Sub UsoDeExecute()
    Dim objFileSearch As Object
    Dim i As Long
    Set objFileSearch = Application.FileSearch
    With objFileSearch
        .LookIn = "C:\Gráficos"
        .FileName = "*.jpg"
        ' msoSortbyFileName tiene el valor 1 _
          msoSortOrderAscending tiene el valor 1
        If .Execute(SortBy:=msoSortByFileName, _
```

```
                SortOrder:=msoSortOrderAscending) > 0 Then
                MsgBox "Se han encontrado " & _
                      .FoundFiles.Count & _
                    " ficheros de tipo gráfico jpg"
                For i = 1 To .FoundFiles.Count
                    MsgBox .FoundFiles(i), _
                           vbInformation, _
                           "Fichero N° " & _
                            Format(i, "0000")
                Next i
            Else
                MsgBox "No hay ficheros jpg"
            End If
        End With

    End Sub
```

Propiedad LastModified

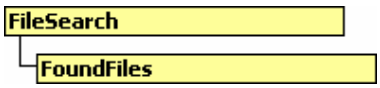
La propiedad **`LastModified`** (de tipo **`Boolean`**) devuelve o establece una constante que indica el tiempo transcurrido desde la última vez que se modificó y guardó el archivo.

El valor predeterminado para esta propiedad es **`msoLastModifiedAnyTime`**, con el valor **7**.

Nombre de la constante	Valor
msoLastModifiedAnyTime *predeterminado*	**7**
msoLastModifiedLastMonth	**5**
msoLastModifiedLastWeek	**3**
msoLastModifiedThisMonth	**6**
msoLastModifiedThisWeek	**4**
msoLastModifiedToday	**2**
msoLastModifiedYesterday	**1**

Este ejemplo, extraído de la ayuda de Access, establece las opciones para la búsqueda de un archivo. Los archivos que devolverá esta búsqueda han sido previamente modificados y están ubicados en la carpeta **`C:\Gráficos`** o en una subcarpeta de ésta.

Objeto FoundFiles



El Objeto **FoundFiles**, es un objeto devuelto por la propiedad **FoundFiles** del objeto **FileSearch**, y que contiene una colección que almacena los datos de los ficheros encontrados mediante el método **Execute**.

Este objeto posee las propiedades de las colecciones **Count** e **Item**, además de las propiedades **Application** y **Creator**.

Count devuelve el número de ficheros encontrados al ejecutar el método **Execute** del objeto **FileSearch**.

Item devuelve los nombres de los ficheros encontrados. Para ello le pasamos el índice correspondiente basado en **cero**.

Application devuelve una referencia a la aplicación contenedora; en nuestro caso **Access**.

Creator devuelve un número **long** asociado a la aplicación contenedora.

Ejemplo

Supongamos que queremos mostrar todos los ficheros que están en la carpeta correspondiente al actual proyecto de Access. Es la carpeta de **CurrentProject.Path**

Para ello utilizaremos el siguiente código:

```
Public Sub UsoDeFileSearch()
    Dim i As Long
    Dim strFicheros As String
    With Application.FileSearch
        .LookIn = CurrentProject.Path
        .FileName = "*.*"
        .SearchSubFolders = True
        If .Execute() > 0 Then
            For i = 1 To .FoundFiles.Count
                strFicheros = strFicheros _
                        & .FoundFiles(i) _
                        & vbCrLf
            Next i
            MsgBox strFicheros, _
                        vbInformation + vbOKOnly, _
                        "Se han encontrado " _
                        & .FoundFiles.Count & _
                        " ficheros"
        Else
            MsgBox "No se han encontrado ficheros", _
                        vbInformation + vbOKOnly, _
                        "Búsqueda en " _
```

```
                                & .LookIn
            End If
        End With
    End Sub
```

Método NewSearch

El método **NewSearch** restablece los valores por defecto de todos los criterios de búsqueda.

Su sintaxis es :

***ObjetoFileSearch*.NewSearch**

Propiedad FileType

La propiedad **FileType** nos permite averiguar o establecer el tipo de archivo que debe buscarse. Es de lectura y escritura

Esta propiedad se debe corresponder a alguna de las constantes **MsoFileType**.

Sus valores posibles son:

Nombre de la constante	Valor
msoFileTypeAllFiles	1
msoFileTypeBinders	6
msoFileTypeCalendarItem	11
msoFileTypeContactItem	12
msoFileTypeCustom	
msoFileTypeDatabases	7
msoFileTypeDataConnectionFiles	17
msoFileTypeDesignerFiles	22
msoFileTypeDocumentImagingFiles	20
msoFileTypeExcelWorkbooks	4
msoFileTypeJournalItem	14
msoFileTypeMailItem	10
msoFileTypeNoteItem	13
msoFileTypeOfficeFiles	2
msoFileTypeOutlookItems	9
msoFileTypePhotoDrawFiles	16
msoFileTypePowerPointPresentations	5
msoFileTypeProjectFiles	19

msoFileTypePublisherFiles	**18**
msoFileTypeTaskItem	**15**
msoFileTypeTemplates	**8**
msoFileTypeVisioFiles	**21**
msoFileTypeWebPages	**23**
msoFileTypeWordDocuments	**3**

Nota:

La constante **msoFileTypeAllFiles**, de valor **1**, hace que se busquen todos los archivos.

La constante **msoFileTypeOfficeFiles**, cuyo valor es **2**, incluye todos los archivos que tienen una de las siguientes extensiones: ***.doc**, ***.xls**, ***.ppt**, ***.pps**, ***.obd**, ***.mdb**, ***.mpd**, ***.dot**, ***.xlt**, ***.pot**, ***.obt**, ***.htm**, o ***.html**.

Ejemplo:

El siguiente código busca en la carpeta **"C:\Gráficos"**, sin tomar en cuenta las subcarpetas, los ficheros tipo **Página Web**.

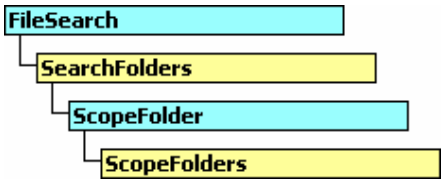
```
Public Sub UsoDeFileType()
    Dim objFileSearch As Object
    Dim i As Long
    Set objFileSearch = Application.FileSearch

    With objFileSearch
        .NewSearch
        .SearchSubFolders = False
        .LookIn = "C:\Gráficos"
        ' msoFileTypeWebPages tiene el valor 23
        .FileType = msoFileTypeWebPages
        If .Execute > 0 Then
                            MsgBox "Se han encontrado " _
            & .FoundFiles.Count & _
            " ficheros de tipo página web"
            For i = 1 To .FoundFiles.Count
            MsgBox .FoundFiles(i), _
                    vbInformation, _
                    "Fichero N° " & Format(i, "0000")
            Next i
        Else
            MsgBox "No hay ficheros web"
        End If
    End With
End Sub
```

Otras propiedades y métodos

En el objeto **FileSearch** podemos encontrar una estructura muy rica de colecciones y objetos.

No pretendo en esta entrega cubrirlos por completo. Para ello remito al lector a la ayuda de Visual Basic para Aplicaciones.

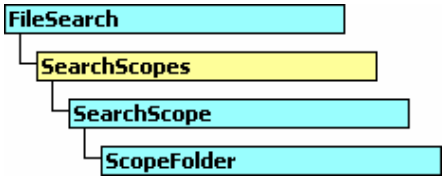
La colección SearchFolders contiene la colección de objetos **ScopeFolder** que determina en qué carpetas se realizará la búsqueda al activar el método **Execute** del objeto **FileSearch**.



El objeto ScopeFolder se corresponde a una carpeta en la que se pueden realizar búsquedas. Los objetos **ScopeFolder** pueden utilizarse con la colección **SearchFolders**.

La colección ScopeFolders contiene la colección de objetos **ScopeFolder** que determina en qué carpetas se realizará la búsqueda al activar el método **Execute** del objeto **FileSearch**.

La colección SearchScopes pertenece al objeto **FileSearch** y contiene los objetos **SearchScope**.



El objeto SearchScope se utiliza para proporcionar acceso a los objetos **ScopeFolder** que pueden agregarse a la colección **SearchFolders**.

Se corresponde a un tipo de árbol de carpetas en las que pueden efectuarse búsquedas utilizando el objeto **FileSearch**.

Como se indica en la ayuda de VBA, las unidades locales de su equipo representan un solo ámbito de búsqueda. Las carpetas de red y las de Microsoft Outlook son también dos ámbitos individuales de búsqueda disponibles. Cada objeto **SearchScope** incluye un solo objeto **ScopeFolder** que corresponde a la carpeta raíz del ámbito de búsqueda.

El ejemplo siguiente está adaptado de la ayuda de VBA y muestra todos los objetos **SearchScope** disponibles actualmente.

```
Public Sub MostrarLosAmbitosDisponibles()

    ' Declara una variable que hace referencia _
     a un objeto SearchScope
    Dim ss As SearchScope

    ' Utiliza un bloque With...End With para referenciar _
    ' el objeto FileSearch
```

```
    With Application.FileSearch

        ' Recorre la colección SearchScopes
        For Each ss In .SearchScopes
            Select Case ss.Type
                Case msoSearchInMyComputer
                    MsgBox "Mi PC" _
               & " es un ámbito de búsqueda disponible."
                Case msoSearchInMyNetworkPlaces
                    MsgBox "Mis sitios de red" _
               & " son un ámbito de búsqueda disponible."
                Case msoSearchInOutlook
                    MsgBox "Outlook" _
               & " es un ámbito de búsqueda disponible."
                Case msoSearchInCustom
                    MsgBox "Hay disponible" _
               & " un ámbito personalizado de búsqueda."
                Case Else
                    MsgBox "No puedo determinar" _
               & " el ámbito de búsqueda."
            End Select
        Next ss
    End With
End Sub
```

El método RefreshScopes actualiza la lista de objetos **ScopeFolder** disponibles actualmente.

Su sintaxis es :

***ObjetoFileSearch*.RefreshScopes**

El siguiente ejemplo, adaptado de la ayuda de VBA, muestra todos los objetos **ScopeFolder** disponibles actualmente en la unidad **C:** del ámbito de **Mi PC** y demuestra la necesidad de utilizar el método **RefreshScopes** cuando se producen cambios en la lista de carpetas.

```
Sub PruebaDelMetodoRefreshScopes()
' Muestra lo que sucede antes o después
' de llamar al método RefreshScopes
' si previamante se ha añadido una nueva carpeta
' a la lista del ámbito de búsqueda.

Dim strCarpeta As String
strCarpeta = "C:\_BorrarDespuésDeSerUsado"
```

```
    ' Si ya existe la carpeta la borramos
    If Len(Dir(strCarpeta)) > 1 Then
         RmDir Path:=strCarpeta
    End If

    ' Refrescamos la lista de carpetas.
    Application.FileSearch.RefreshScopes

    ' Lista antes de crear la carpeta
    Call ListarNombresDeCarpetas

    ' Creamos una nueva carpeta en el disco C:\
    ' Se producirá un error si la carpeta ya existiera

    MkDir Path:=strCarpeta

    ' Lista después de haber creado la carpeta
    ' La carpeta nueva no aparece en la lista.
    Call ListarNombresDeCarpetas

    ' Refrescamos la lista de carpetas.
    Application.FileSearch.RefreshScopes

    ' Ahora la carpeta nueva sí aparece en la lista.
    Call ListarNombresDeCarpetas

    ' Borramos la carpeta
    RmDir Path:=strCarpeta
End Sub

Sub ListarNombresDeCarpetas()
    Dim i As Integer
    Dim strResultados As String

    ' Recorre todas las carpetas en el disco C:\
    ' en Mi Pc e informa de los resultados
    ' .SearchScopes.Item(1) = "Mi Pc"
    ' .ScopeFolders.Item(2) = "C:\"

    With Application.FileSearch.SearchScopes.Item(1). _
        ScopeFolder.ScopeFolders.Item(2)
```

```
            For i = 1 To .ScopeFolders.Count
              strResultados = strResultados & .ScopeFolders. _
                    Item(i).Name & vbCrLf
            Next i

            MsgBox "Nombres de carpetas en C:\...." _
                    & vbCrLf _
                    & vbCrLf _
                    & strResultados
        End With
    End Sub
```

Nota:

*Los apartados anteriores, referentes al objeto **FileSearch**, tienen exclusivamente como objetivo, introducir al lector en el uso de este objeto y sus posibilidades.*

Profundizar en los mismos está fuera del alcance y de los objetivos de este texto, por lo que remito a los posibles interesados a la ayuda de VBA, en donde podrán encontrar una extensa reseña sobre ellos.