Recursive Functions

Function that calls itself

Crystal Long (strive4peace)

MsAccessGurus.com

YouTube/LearnAccessByCrystal

Index

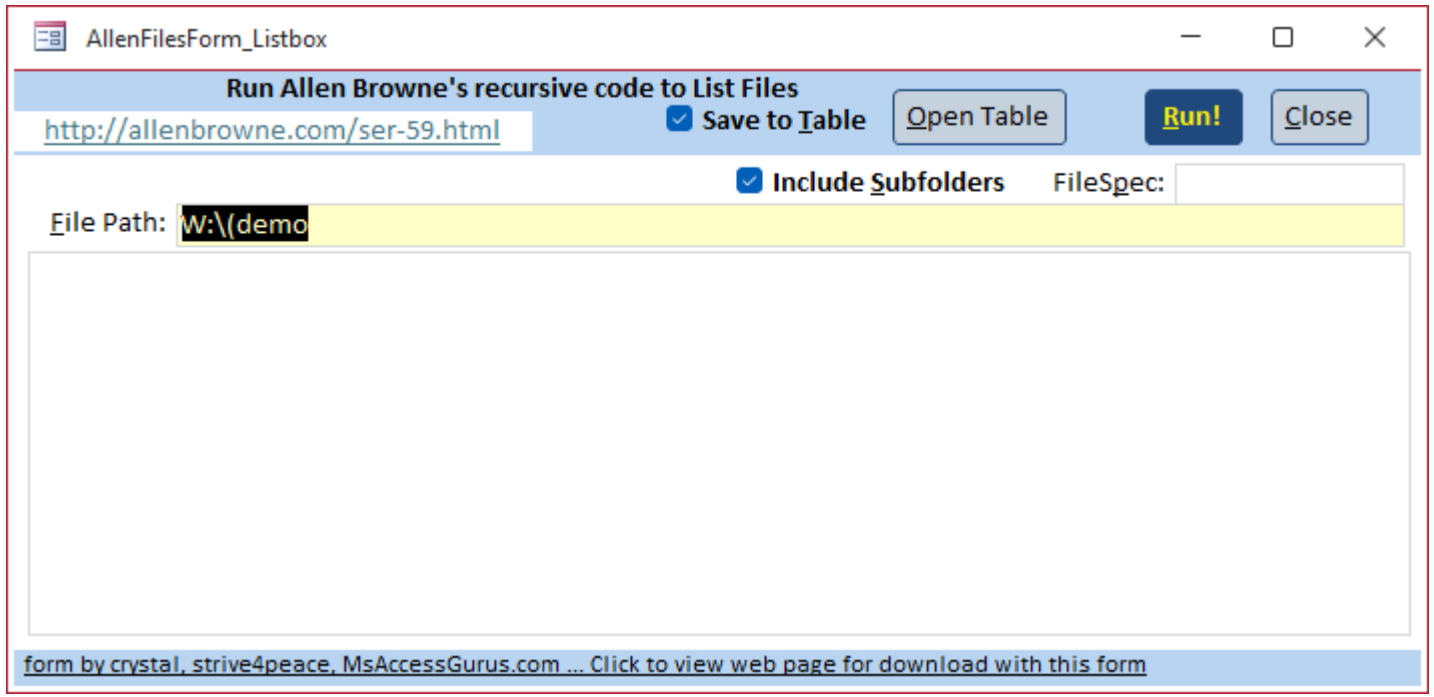
— *Allen Browne's* ListFiles

Recursive part is in the FillDir function called by ListFiles that adds on to a folder path down from another folder to make a complete path to a file

For instance:
there may be files in the specified path AND files in folders below it.

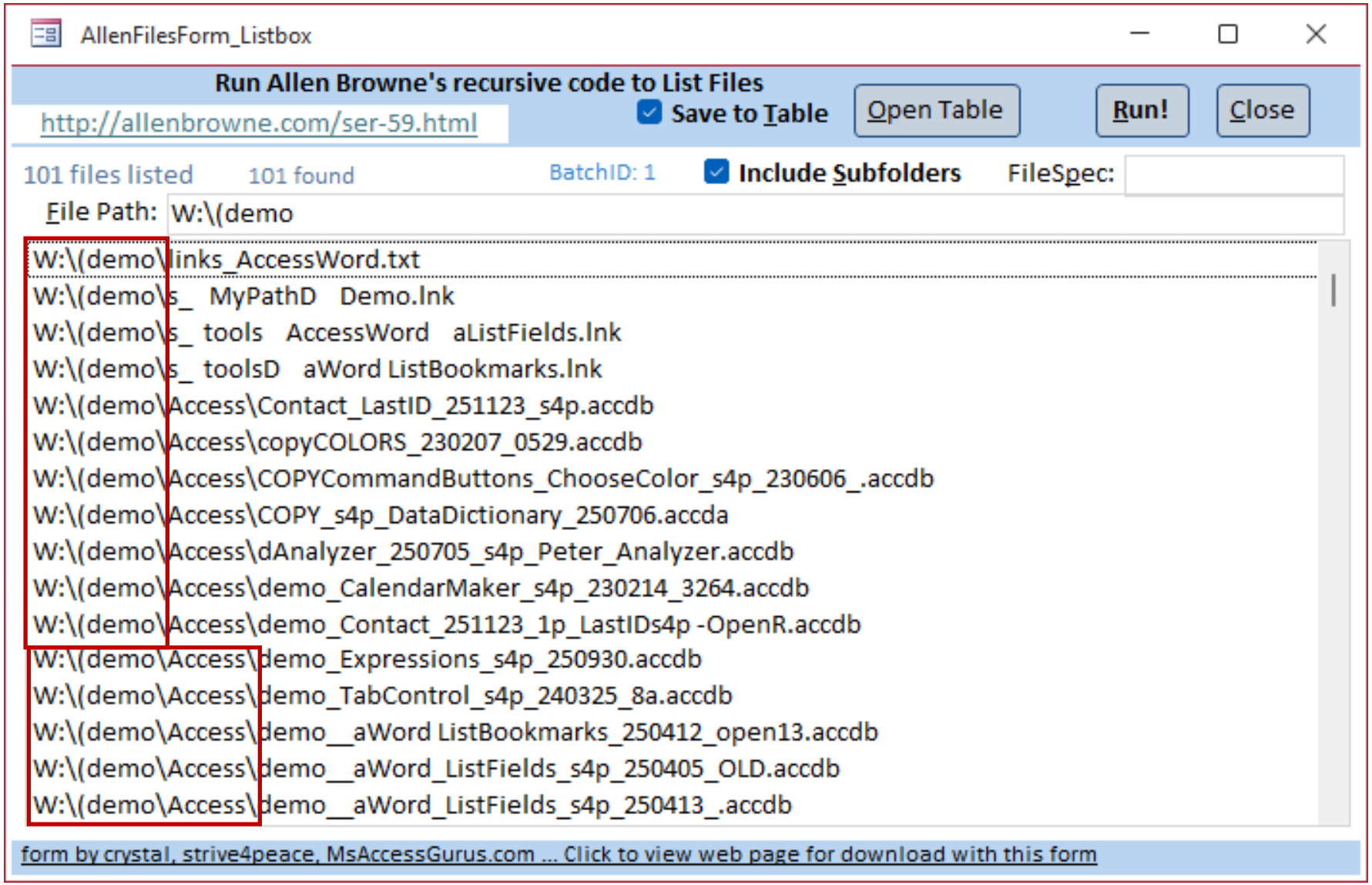
W:\(demo\Access\Contacts\Contacts_BE_SampleData.accdb

Form to Run *Allen's* ListFiles code



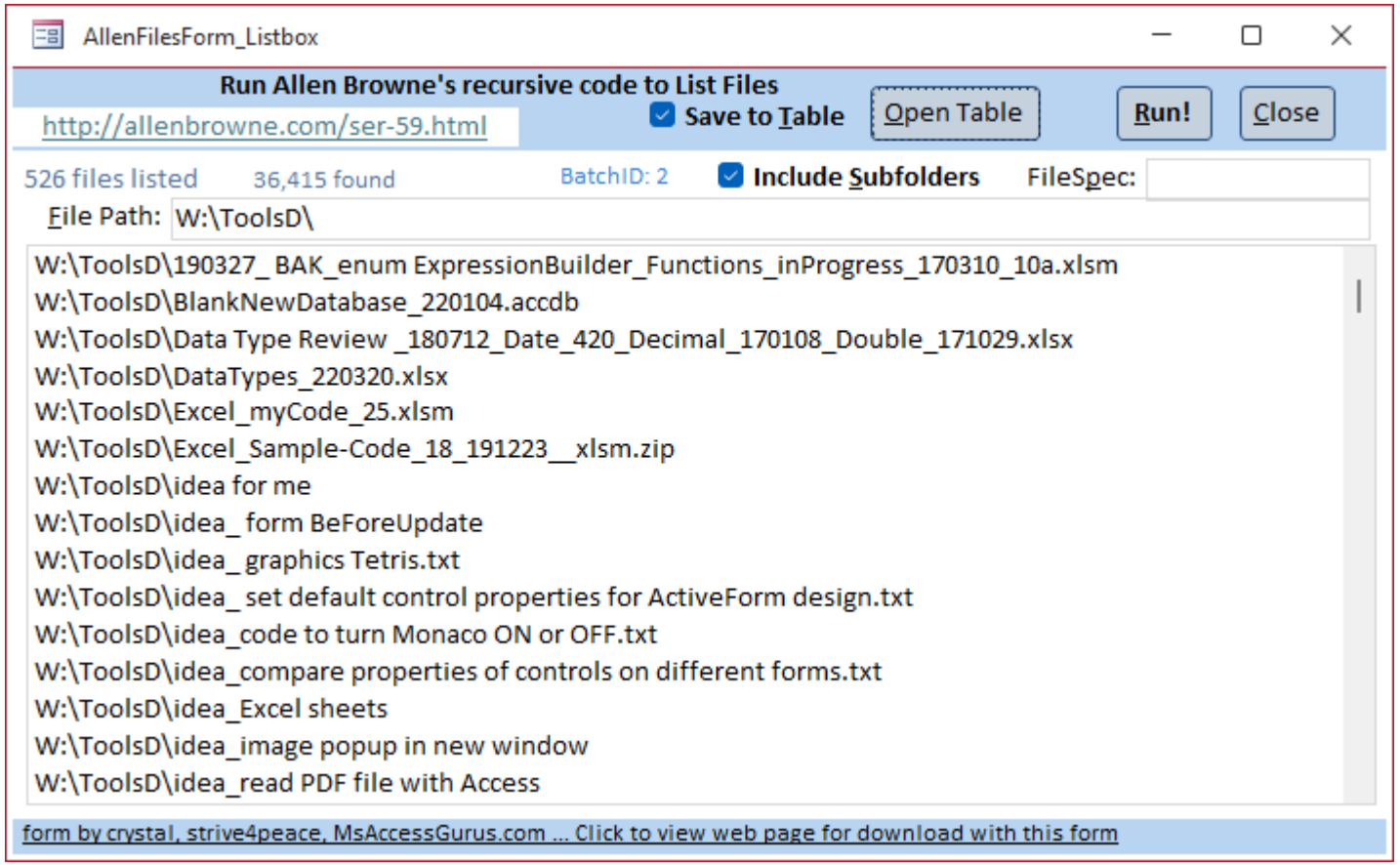
1. Type or Paste File Path
2. Decide Include Subfolders
3. Specify FileSpec if desired
4. Decide Save to Table *(s4p_FileList)*
5. Click Run!

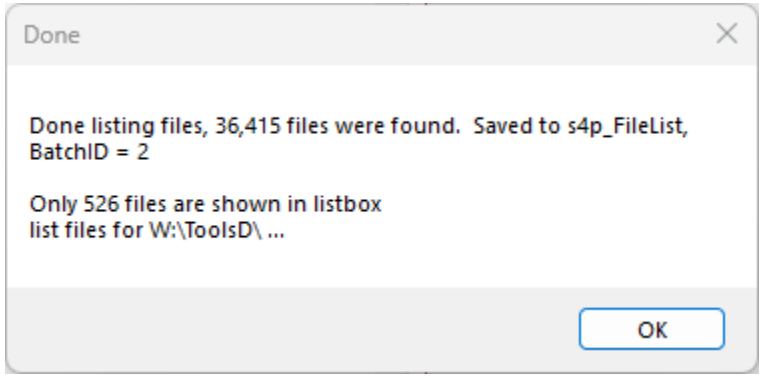
Allen's ListFiles Result



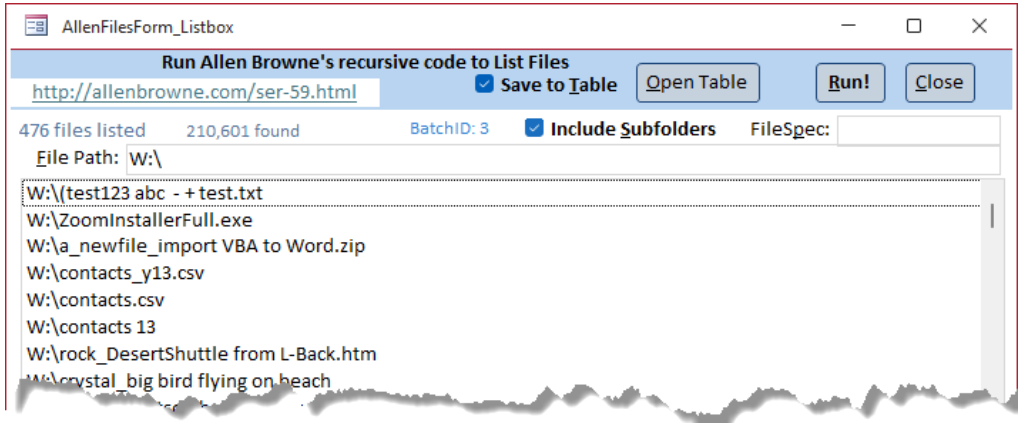
Show file names and paths.
If the "file" is a folder then,
optionally if Include Subfolders is checked,
recursively loop each subfolder and add more files to the collection.

Too many results for listbox





BatchID=3: ran it again on the whole drive and got even more files



RUNNING get list of files for specified path ... W:\ToolsD\CHEMISTRY\Notes\

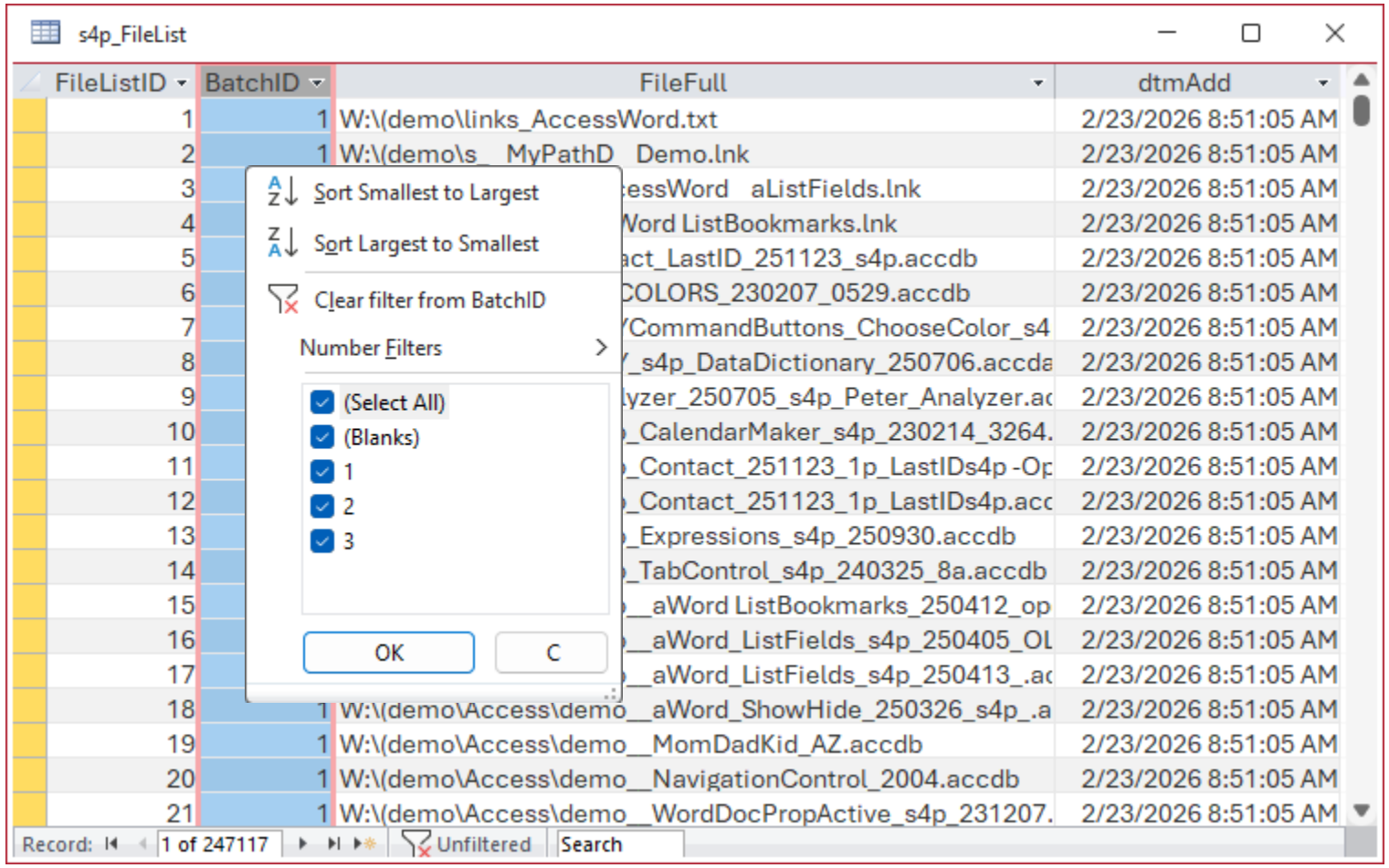
Look at StatusBar in lower left to see it is running

query to show number of files for each batch

s4p_qFileList_Batch_CountNumFiles

BatchID	NbrFiles
1	101
2	36,415
3	210,601

Record: 1 of 3 No Fil



Do whatever you want with the results

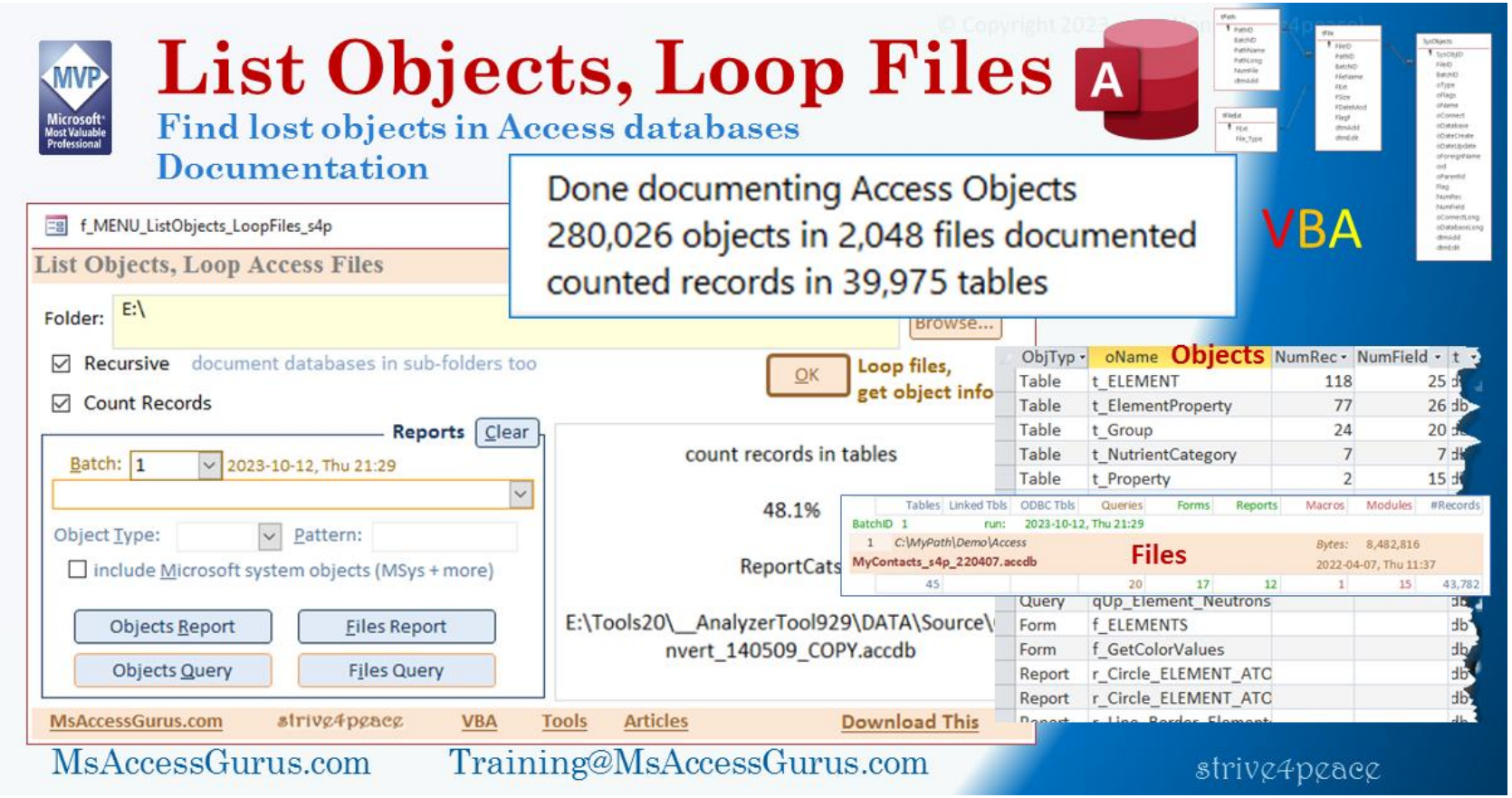
add more fields to the table (s4p_FileList) to parse and get information like:

- Path
- File name
- Extension
- Size
- Date created
- Date modified

and then depending on file type:

- Dimensions
- Length
- whatever you write code to figure out

List objects, loop files



Did you lose track of something important in your many Access databases?

I wrote this to find a module and found it!

https://msaccessgurus.com/tool/ListObjects_LoopFiles.htm

— Concepts

to better understand Allen's code

Help - Dir Function

Learn / VBA /
https://learn.microsoft.com/en-us/office/vba/language/reference/user-interface-help/dir-function

Dir function

Returns a **String** representing the name of a file, directory, or folder that matches a specified pattern or file attribute, or the volume label of a drive.

Syntax

Dir [(*pathname*, [*attributes*])]

The **Dir** function syntax has these parts:

Part	Description
pathname	Optional. String expression that specifies a file name; may include directory or folder, and drive. A zero-length string ("") is returned if *pathname* is not found.
attributes	Optional. Constant or numeric expression, whose sum specifies file attributes. If omitted, returns files that match *pathname* but have no attributes.

Settings

The *attributes* argument settings are:

Constant	Value	Description
vbNormal	0	(Default) Specifies files with no attributes.
vbReadOnly	1	Specifies read-only files in addition to files with no attributes.
vbHidden	2	Specifies hidden files in addition to files with no attributes.
vbSystem	4	Specifies system files in addition to files with no attributes. Not available on the Macintosh.
vbVolume	8	Specifies volume label; if any other attribute is specified, **vbVolume** is ignored. Not available on the Macintosh.
vbDirectory	16	Specifies directories or folders in addition to files with no attributes.
vbAlias	64	Specified file name is an alias. Available only on the Macintosh.

ADDING FILES to a collection

```
' first time, attributes not specified so Normal files
strTemp = Dir( strFolder & strFileSpec )

' loop
 Do While strTemp <> vbNullString
       colDirList.Add strFolder & strTemp
       strTemp = Dir
   Loop
```

Object Browser says that vbNullString = ""
although it can mean more

Help - Collection

Learn / VBA /

Collection object

A **Collection** object is an ordered set of items that can be referred to as a unit.

Remarks

The **Collection** object provides a convenient way to refer to a related group of items as a single object. The items, or members, in a collection need only be related by the fact that they exist in the collection. Members of a collection don't have to share the same data type.

A collection can be created the same way other objects are created. For example:

VB Copy

```
Dim X As New Collection
```

After a collection is created, members can be added by using the Add method and removed by using the Remove method. Specific members can be returned from the collection by using the Item method, while the entire collection can be iterated by using the For Each...Next statement.

Example

This example creates a **Collection** object (`MyClasses`), and then creates a dialog box in which users can add objects to the collection.

To see how this works, choose the **Class Module** command from the Insert menu and declare a public variable called `InstanceName` at the module level of Class1 (type **Public** `InstanceName`) to hold the names of each instance. Leave the default name as Class1. Copy and paste the following code into the General section of another module, and then start it with the statement `ClassNamer` in another procedure.

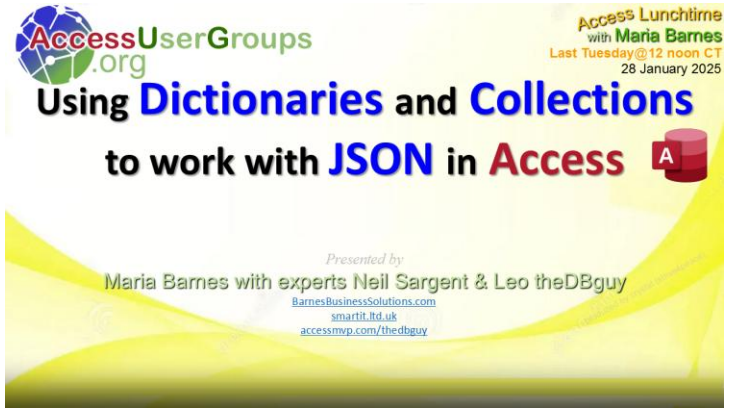
(This example only works with host applications that support classes.)

VB Copy

```
Sub ClassNamer()
    Dim MyClasses As New Collection    ' Create a Collection object.
    Dim Num    ' Counter for individualizing keys.
    Dim Msg As String    ' Variable to hold prompt string.
    Dim TheName, MyObject, NameList    ' Variants to hold information.
```

VBA Collection Object

https://learn.microsoft.com/en-us/office/vba/language/reference/user-interface-help/collection-object



https://youtu.be/oKRtghY9cJ0

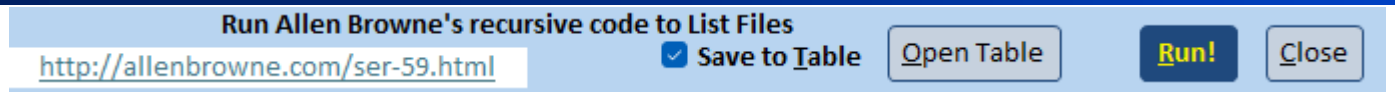
Helper Function: *Allen's* TrailingSlash

```
Public Function TrailingSlash( varIn As Variant ) As String
    If Len( varIn ) > 0& Then                    & is the type declaration
        If Right( varIn, 1& ) = "\" Then         character for Long Integer
            TrailingSlash = varIn          RETURN value passed if it already ends with "\"
        Else
            TrailingSlash = varIn & "\"    RETURN value with "\" on end if not there
        End If
    End If
End Function
```

— ListFiles and FillDir Code

by Allen Browne, modified by Crystal

Call ListFiles



the Run! button calls Allen's code to build a collection of file names and paths.

First parameter is path to look in for files and the only required argument. Second is a file specification with wildcards. Third is whether or not to include subfolders. Last is listbox control for results instead of writing to Immediate window.

```
' recursively list files for path
Call ListFiles( sPath _
   , .FileSpec _
   , .Checkbox_IncludeSubfolders _
   , .lstFileList)
```

```
' List files recursively, by Allen Browne
' http://allenbrowne.com/ser-59.html

Public Function ListFiles(strPath As String, _
   Optional strFileSpec As String, _
   Optional bIncludeSubfolders As Boolean, _
   Optional lst As ListBox)
```

Allen's ListFiles VBA

```
'from http://allenbrowne.com/ser-59.html
Public Function ListFiles(strPath As String,_
   Optional strFileSpec As String,_
   Optional bIncludeSubfolders As Boolean,_
   Optional lst As ListBox)
On Error GoTo Err_Handler
   'Purpose:   List the files in the path.
   'Arguments: strPath = the path to search.
   '          strFileSpec = "*.*" unless you specify differently.
   '          bIncludeSubfolders: If True, results from subdirectories as well.
   '          lst: if you pass in a list box, items are added to it.
   '             If not, files are listed to immediate window.
   '             The list box must have RowSourceType property set to ValueList.
   'Method:    FilDir() adds items to collection, recursively for subfolders.
   Dim colDirList As New Collection
   Dim varItem As Variant

   ' ~ inialize variable value
   gnCount = 0  ' ~~~ added by crystal

   'call function that can be recursive
   Call FillDir(colDirList, strPath, strFileSpec, bIncludeSubfolders)

 ' ~ how many files are in the collection
   gnCount = colDirList.Count   ' ~~~ added by crystal

   'Add the files to a list box if one was passed in.
   ' Otherwise list to the Immediate Window.
   If lst Is Nothing Then
      For Each varItem In colDirList
         Debug.Print varItem
      Next
   Else
      For Each varItem In colDirList
         lst.AddItem varItem
      Next
   End If
   '------------------- 's4p ~~~ added by crystal
   ' write collection to a table
   If gbSAVEtoTABLE <> False Then
     Call SaveCollectionToTable_s4p(colDirList)
   End If

Exit_Handler:
   Exit Function

Err_Handler:
   MsgBox  "Error " & Err.Number &  ": " & Err.Description
   Resume Exit_Handler
   Resume
End Function
```

Allen's FillDir VBA (recursive) *add files*

in calling procedure: Dim colDirList As New Collection 'nothing is in collection until **FillDir** adds file names preceeded by path

```
'from http://allenbrowne.com/ser-59.html
Private Function FillDir( colDirList As Collection, _
  ByVal strFolder As String, _
  strFileSpec As String, _
   bIncludeSubfolders As Boolean)
   'Build up a list of files, and then add add to this list
   ' , any additional folders
   Dim strTemp As String
   Dim colFolders As New Collection
   Dim vFolderName As Variant

   'Add the files to the folder.
   strFolder = TrailingSlash( strFolder )    strFolder is the path
   strTemp = Dir( strFolder & strFileSpec )    get first file name in folder

   Do While strTemp <> vbNullString  do while there is a file name in folder
      colDirList.Add strFolder & strTemp  Add item to collection
      strTemp = Dir  get next file in folder
   Loop

   If bIncludeSubfolders Then
      'Build collection of additional subfolders.
      strTemp = Dir(strFolder,vbDirectory)
      Do While strTemp <> vbNullString
         If (strTemp <>  ".") And (strTemp <>  "..") Then
            If (GetAttr(strFolder & strTemp) And vbDirectory) <> 0& Then
               colFolders.Add strTemp
            End If
         End If
         strTemp = Dir
      Loop
      'Call function recursively for each subfolder.
      For Each vFolderName In colFolders
         Call FillDir(colDirList, strFolder & TrailingSlash(vFolderName) _
                        , strFileSpec, True)
      Next vFolderName
   End If
End Function
```

Allen's FillDir VBA (recursive) *subfolders*

```
'from http://allenbrowne.com/ser-59.html

Private Function FillDir( colDirList As Collection, _
  ByVal strFolder As String, _
  strFileSpec As String, _
   bIncludeSubfolders As Boolean)
   'Build up a list of files, and then add add to this list
   ' , any additional folders
   Dim strTemp As String
   Dim colFolders As New Collection
   Dim vFolderName As Variant

   'Add the files to the folder.
   strFolder = TrailingSlash(strFolder)
   strTemp = Dir(strFolder & strFileSpec)

   Do While strTemp <> vbNullString
      colDirList.Add strFolder & strTemp
      strTemp = Dir
   Loop

   If bIncludeSubfolders Then
      'Build collection of additional subfolders.
      strTemp = Dir( strFolder, vbDirectory) Normal Files + Folders
      Do While strTemp <> vbNullString
         If (strTemp <>  ".") And (strTemp <>  "..") Then
            If (GetAttr(strFolder & strTemp) And vbDirectory) <> 0& Then
               colFolders.Add strTemp   strTemp is now the folder path
            End If
         End If
         strTemp = Dir
      Loop
      'Call function recursively for each subfolder.
      For Each vFolderName In colFolders
         Call FillDir( colDirList, strFolder & TrailingSlash( vFolderName ) _
                  , strFileSpec, True)
      Next vFolderName
   End If
End Function
```

in calling procedure: Dim colDirList As New Collection 'nothing is there until this procedure fills the collection with files

— s4p Save Relationship Paths

RelPath	BaseTable	RelTable	Levl	RelField	NoteRP	ParentTable	ParentField
c_Category—c_Contact—c_Contact	c_Category	c_Contact	2	CID_	self-join	c_Contact	CID
c_Category—c_Contact—c_eAddress	c_Category	c_eAddress	2	CID		c_Contact	CID
c_Category—c_Contact—c_List	c_Category	c_List	2	CID_Li		c_Contact	CID
c_Category—c_Contact—c_List—c_List	c_Category	c_List	3	ListID_	self-join	c_List	ListID
c_Category—c_Contact—c_List—c_Member	c_Category	c_Member	3	ListID		c_List	ListID
c_Category—c_Contact—c_Member	c_Category	c_Member	2	CID		c_Contact	CID
c_Category—c_Contact—c_Phone	c_Category	c_Phone	2	CID		c_Contact	CID
c_Category—c_Contact—c_URL	c_Category	c_URL	2	CID		c_Contact	CID

Import RelPath table and module

to see RelPath in action, you need to put it into a database.

import:

- s4p_RelPath *table*
- s4p_mod_RelPath_Recurse *module*

into your database.

Compile, Save, and run **DoRelPath_All_s4p** OR **DoRelPath_OneTable_s4p**

RelPath Results

RelPath	BaseTable	RelTable	Levl	RelField	NoteRP	ParentTable	ParentField
c_AdrType—c_Address	c_AdrType	c_Address	1	TypIDadr		c_AdrType	TypIDadr
c_Category—c_Contact	c_Category	c_Contact	1	CatID		c_Category	CatID
c_Category—c_Contact—c_Address	c_Category	c_Address	2	CID		c_Contact	CID
c_Category—c_Contact—c_Contact	c_Category	c_Contact	2	CID_	self-join	c_Contact	CID
c_Category—c_Contact—c_eAddress	c_Category	c_eAddress	2	CID		c_Contact	CID
c_Category—c_Contact—c_List	c_Category	c_List	2	CID_Li		c_Contact	CID
c_Category—c_Contact—c_List—c_List	c_Category	c_List	3	ListID_	self-join	c_List	ListID
c_Category—c_Contact—c_List—c_Member	c_Category	c_Member	3	ListID		c_List	ListID
c_Category—c_Contact—c_Member	c_Category	c_Member	2	CID		c_Contact	CID
c_Category—c_Contact—c_Phone	c_Category	c_Phone	2	CID		c_Contact	CID
c_Category—c_Contact—c_URL	c_Category	c_URL	2	CID		c_Contact	CID
c_Contact—c_Address	c_Contact	c_Address	1	CID		c_Contact	CID

— DEMO

demonstrate in Access

— Resources

Links

Download this presentation + ACCDB databases for ListFile and RelPath
https://msaccessgurus.com/presentation/Recursive_Functions.htm

List files recursively by Allen Browne
http://allenbrowne.com/ser-59.html

Dir function
https://learn.microsoft.com/en-us/office/vba/language/reference/user-interface-help/dir-function

Collection object
https://learn.microsoft.com/en-us/office/vba/language/reference/user-interface-help/collection-object

List Objects, Loop files – names of Access objects
https://msaccessgurus.com/tool/ListObjects_LoopFiles.htm

AL: Using Dictionaries and Collections to work with JSON in Access by Maria Barnes, Neil Sargent, Leo theDBguy (1:03:47)
https://youtu.be/c7PwDuXPLeA

Next Meeting: Interfacing with the Outlook calendar by Maria Barnes
https://accessusergroups.org/calendar/lunchtime-2-2026-03-31/

Access Lunchtime Playlist on YouTube:
https://www.youtube.com/playlist?list=PLwIULo1M2EWvapx3zAhd3jhX_ITigxM6M

Next Access Lunchtime Meeting