Visual Studio Magazine Online
Classic VB Corner

Finding an Associated Executable

After using a given API for a decade or more, you tend to just take it for granted that it works. Karl Peterson shows how he worked around a challenge when it didn't.

October 13, 2009 · by Karl E. Peterson

Not long after I submitted my last column , which showed among other things how to find what application would launch a given document file, I ran across an interesting discussion. An old acquaintance of mine had noticed that FindExecutable wasn't returning results for the .ACCDB file extension.

The discussion evolved and it was discovered that there were numerous 5-letter file extensions FindExecutable didn't like, but others it had no trouble with whatsoever. The pattern was totally random, as were many of the made-up file extensions. All in all, it was a fine demonstration that the ClassicVB community is still alive and thriving in the public newsgroups, as the problem was diagnosed and other methods were found. I'd like to share my own new strategy with you. My thanks to Tony Toews, Mike Williams and "Nobody" for finding the problem and contributing to its solution. It's a good thread that's worth reading.

If you haven't gotten around to moving beyond Office 2003, a little background may be in order. ACCDB is the new extension used by Access 2007 for its data files. Microsoft would like us to view MDBs as relics of the past. Unfortunately, they didn't really test out this concept against their tried and true set of file APIs. If you pass the full path and name of an ACCDB file to FindExecutable, it will return SE_ERR_NOASSOC indicating there is no associated application for that filetype. However, Windows certainly knows the filetype, as it shows "Microsoft Office Access 2007 Database" in Explorer. (Oddly enough, FindExecutable returns correct results on Windows 98. Hmmm.)

Turns out, there's another API function that'll not only tell you what the associated executable is, but much more. AssocQueryString provides numerous association strings, such as the command line to launch a document, the friendly document and application names, DDE topics and commands, and so on. And, best of all, AssocQueryString has no troubles whatsoever with 5-letter file extensions.

That said, to make this solution work on older platforms (NT4, 98, 95) you need to have Internet Explorer 5 (or higher) installed. That means we need to use both solutions together, to assure best possible results. If all you're looking for is the associated executable, the old time FindExecutable will work in nearly all cases.

So, building on our last project , my new FindApplication routine looks like this:

```
Public Function FindApplication(ByVal DocName As String) As String
   Dim sRet As String
   ' Try FindExecutable first...
   sRet = FindApplication1(DocName)
   If Len(sRet) = 0 Then
      ' and AssocQueryString otherwise.
      If Exported("shlwapi", "AssocQueryStringA") Then
         sRet = FindApplication2(DocName)
      End If
   End If
   FindApplication = sRet
End Function
```

The FindApplication1 function uses the previous method of calling FindExecutable, and FindApplication2 uses the new method of calling AssocQueryString. Note that I'm also testing to be sure AssocQueryString is available, to avoid an unhandled error in the unlikely event my code finds itself running on a system with IE4. FindApplication2 looks like this:

```
Private Function FindApplication2(ByVal DocName As String) As String
   ' Minumum OS: Windows 2000, Windows NT 4.0 with IE5,
   '             Windows 98, Windows 95 with IE5
   Dim n As Long
   Dim Buffer As String
   Dim BufLen As Long

   ' Reduce document name to just extension, taking care to
   ' allow cases where just extension itself is passed.
   n = InStrRev(DocName, ".")
   If n Then
      DocName = Mid$(DocName, n)
   Else
      DocName = "." & DocName
   End If

   ' Prepare buffer for results.
   Buffer = Space$(MAX_PATH)
   BufLen = Len(Buffer)

   ' Use the AssocQueryString API to find application
   ' associated with passed document extension.
   n = AssocQueryString(0&, ASSOCSTR_EXECUTABLE, _
         DocName, "open", Buffer, BufLen)
   If n = S_OK Then
      FindApplication2 = Left$(Buffer, BufLen - 1)
   End If
End Function
```

One big difference between these two API functions is that AssocQueryString is just returning results for a given file extension, whereas FindExecutable requires the complete filename of an actually existing file. So in this regard alone, the newer function is far more versatile, in that you no longer need to create a temporary file to determine random associations. I've updated the Which sample on my site with this code, in case you already downloaded the original and would like to update that.

The next logical question is, what are you going to do with that association? Well, odds are, you're planning to launch the document file, right? For nearly all purposes, of course, ShellExecute will work just fine for that. But there are situations where it won't, for whatever reason, and you'd simply like to be able to fire it off yourself using VB's own Shell or the CreateProcess API. Here's another case where FindExecutable doesn't provide quite enough information. For example, if you have the Windows Picture and Fax Viewer associated with GIF or JPG files, FindExecutable will point you straight at C:\WINDOWS\system32\shimgvw.dll. That's not something you would ordinarily be able to execute!

Well, one of the other strings provided by AssocQueryString is the launch command for documents. In the case just mentioned, you'd get back:

```
rundll32.exe C:\WINDOWS\system32\shimgvw.dll,ImageView_Fullscreen %1
```

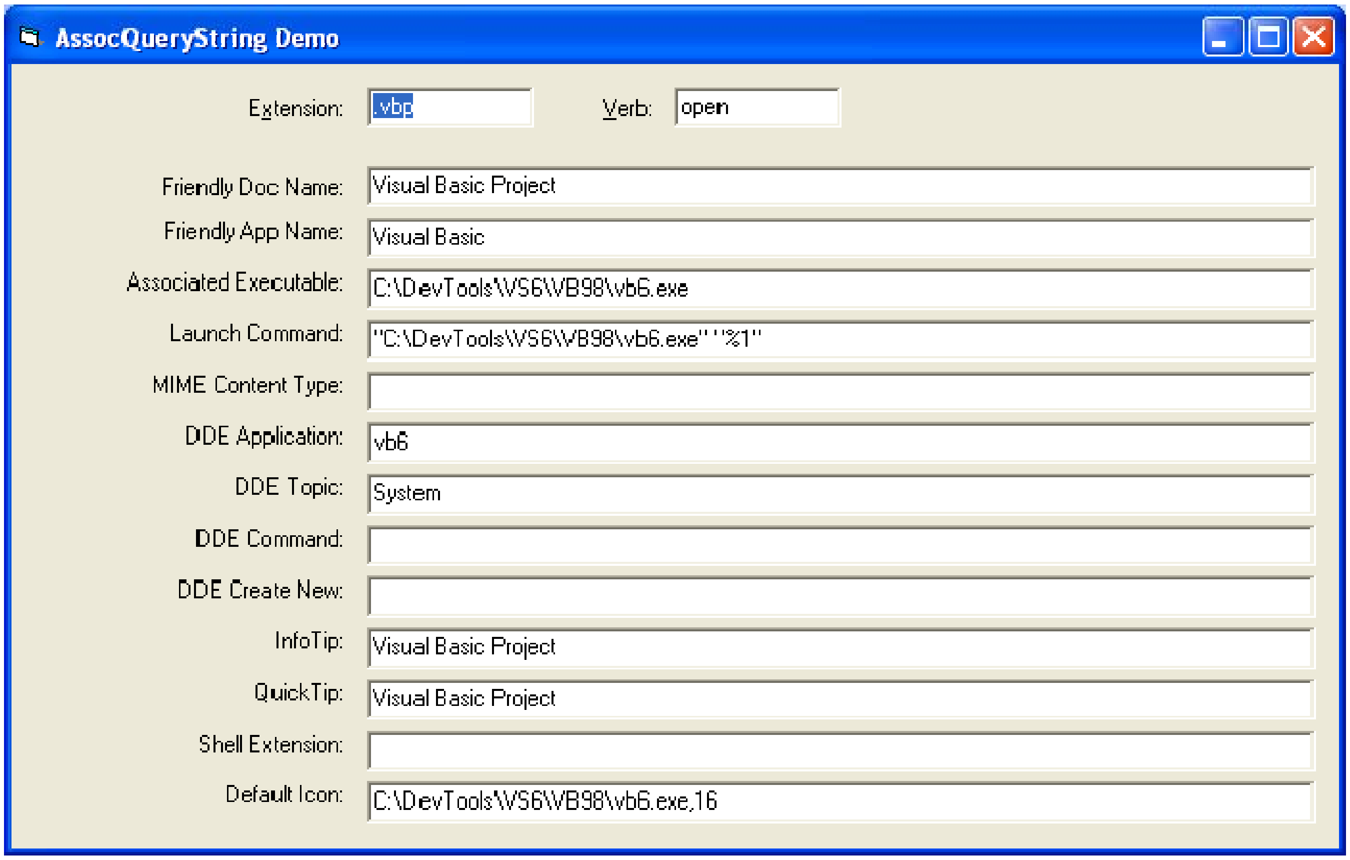
Just about there! As you're probably aware, that %1 parameter represents the filename to be launched. The lack of quotes around it is something that ought to set off a quiet alarm. Most applications that work with long filenames will use "%1" instead to help delineate the actual filename. Reading up on rundll32, we can confirm that it doesn't understand how to parse quoted strings. So, you'll need to chop the filename down to its short version.

Microsoft provides a method to derive a short filename from a long filename in KB175512. Of course, like most KB code, it needs some good scrubbing to really be presentable. Here's my cleaned-up version of that routine:

```
Private Declare Function GetShortPathName Lib "kernel32" _
      Alias "GetShortPathNameA" (ByVal lpszLongPath As String, _
      ByVal lpszShortPath As String, ByVal cchBuffer As Long) As Long

Public Function GetShortName(ByVal LFN As String) As String
    Dim nRet As Long
    Dim Buffer As String

    ' Determine size needed for buffer.
    nRet = GetShortPathName(LFN, vbNullString, 0&)
    Buffer = Space$(nRet)

    ' Call the function, and clean results.
    nRet = GetShortPathName(LFN, Buffer, Len(Buffer))
    GetShortName = Left$(Buffer, nRet)
End Function
```

Given all that, if LFN represents the long filename of a document you'd like to launch, you would build a Shell string by taking the ASSOCSTR_COMMAND string returned by AssocQueryString and replacing "%1" with the document filename. Then, in case there wasn't a "%1" but there is a %1, you'd replace %1 with GetShortName(LFN). In cases where there wasn't a %1 at all, you're kind of left on your own, but can probably assume the program accepts the filename as the first/only parameter on its command line. Safest would be to, again, pass the short filename in these cases.

I've added a new Assoc sample to my site, which demonstrates about a dozen different strings returned by AssocQueryString (see figure below). It lets you enter any extension and verb (open, print, etc.) pair to see what Windows knows about those. The sample provides a drop-in ready class that'll work in any VB6 or VBA application. VB5 users would need to uncomment a replacement function for InstrRev.



Extension:	.vbp
Verb:	open
Friendly Doc Name:	Visual Basic Project
Friendly App Name:	Visual Basic
Associated Executable:	C:\DevTools\VS6\VB98\vb6.exe
Launch Command:	"C:\DevTools\VS6\VB98\vb6.exe" "%1"
MIME Content Type:	
DDE Application:	vb6
DDE Topic:	System
DDE Command:	
DDE Create New:	
InfoTip:	Visual Basic Project
QuickTip:	Visual Basic Project
Shell Extension:	
Default Icon:	C:\DevTools\VS6\VB98\vb6.exe,16

About the Author

Karl E. Peterson wrote Q&A, Programming Techniques, and various other columns for VBPJ and VSM from 1995 onward, until Classic VB columns were dropped entirely in favor of other languages. Similarly, Karl was a Microsoft BASIC MVP from 1994 through 2005, until such community contributions were no longer deemed valuable. He is the author of VisualStudioMagazine.com's new Classic VB Corner column. You can contact him through his Web site if you'd like to suggest future topics for this column.