ExecApp, ExecRegisteredApp, and LookupRegisteredApp - non-MFC functions to execute an application

Hans Dietrich, 9 Jun 2008 CPOL

4.83 (25 votes)

ExecApp is a replacement for WinExec(). ExecRegisteredApp executes the app that is registered for the specified file extension. LookupRegisteredApp retrieves the application file path that is registered for the specified file extension.

Download demo project - 37 Kb

Introduction

I created these functions to handle common cases where you want to run an application and know application file name (the **WinExec()** case), or you don't know application file name, but do know type of file you want to open (the **registered file extension** case).

The functions described below have been tested on Windows 98, 2000, XP, and Vista.

ExecApp

Since we are being told that WinExec is now deprecated, it is necessary to use CreateProcess instead. There are a few things to watch out for with CreateProcess. The first is that CreateProcess creates two handles that you must close or you will have leaks. These are the process and thread handles, which are typically useful only if you are performing timing studies, monitoring critical threads, or something similar. The second thing to be aware of is how CreateProcess deals with arguments passed to the target application - the CreateProcess `lpCommandLine`parameter. It turns out that the simplest, most reliable way to use CreateProcess is to use `NULL` for the first parameter, and pass the application file path and arguments (each enclosed in quotes, separated by a space) in the second parameter.

Here is function header of `ExecApp()`:

```
///////////////////////////////////////////////////////////////////////////////
//
// ExecApp()
//
// Purpose:     Runs the specified application (replacement for WinExec)
//
// Parameters:  lpszCommandLine - [in] command line (including exe file path)
//                               that is passed to CreateProcess()
//              wShowCmd       - [in] Specifies how app window is to be shown.
//                               See ShowWindow() in MSDN for possible values.
//
```

```
// Returns:     BOOL           - TRUE = CreateProcess() succeeded
//
BOOL ExecApp(LPCTSTR lpszCommandLine, WORD wShowCmd /*= SW_SHOWNORMAL*/)
```

LookupRegisteredApp

`LookupRegisteredApp()` retrieves the application file path that is registered for the specified file extension.`LookupRegisteredApp()` does its work by calling the shell function AssocQueryString. This function is exported via the import lib *shlwapi.lib*, which is automatically linked to in *ExecApp.cpp*. (Note: requires **Internet Explorer 5**or later).

Here is function header of `LookupRegisteredApp()`:

```
///////////////////////////////////////////////////////////////////////////////
//
// LookupRegisteredApp()
//
// Purpose:     Retrieves the application registered for the specified file
//              extension
//
// Parameters:  lpszExt        - [in] file extension (e.g., ".txt") used to
//                               look up application file path. Preceding '.'
//                               is necessary.
//              lpszApplication - [out] application path buffer
//              nSize          - [in/out] size of path buffer in TCHARs
//
// Returns:     BOOL           - TRUE = found registered app
//
// Notes:       AssocQueryString() is broken in Vista. If no application is
//              associated with the file extension, in Vista the function returns
//              the "Unknown" application, rather than an error code (as in XP).
//              Adding ASSOCF_IGNOREUNKNOWN to the flags parameter will make the
//              function behave as in XP. ASSOCF_IGNOREUNKNOWN is defined in the
//              latest Platform SDK.
//
BOOL LookupRegisteredApp(LPCTSTR lpszExt,
                         LPTSTR lpszApplication,
                         DWORD *nSize)
```

Aside from the small glitch in using AssocQueryString on Vista (see *Notes* in preceding header), there is another, more serious problem with AssocQueryString: the ANSI version of this function (`AssocQueryStringA`) doesn't work. I don't know what the exact problem is, but the workaround is to convert the parameters to Unicode, call`AssocQueryStringW`, and then convert the result to ANSI. This is why *ExecApp.cpp* contains the private (static) function `_AssocQueryString`.

ExecRegisteredApp

Here is a common scenario: you want to allow user to edit/view a file (such as a log file). But the ".*log*" file extension might not be registered to an editor on the user's system. What to do? Of course you could just throw the file at ShellExecute, but you know what will happen: the user will be presented with a dialog, and then have to go through some clunky browse sequence, and then you will get a support call. Since you know the log file is just plain ASCII, it's better to ask the system to execute the application that is associated with .*txt* files, passing it the name of the log file. That is what `ExecRegisteredApp()` does.

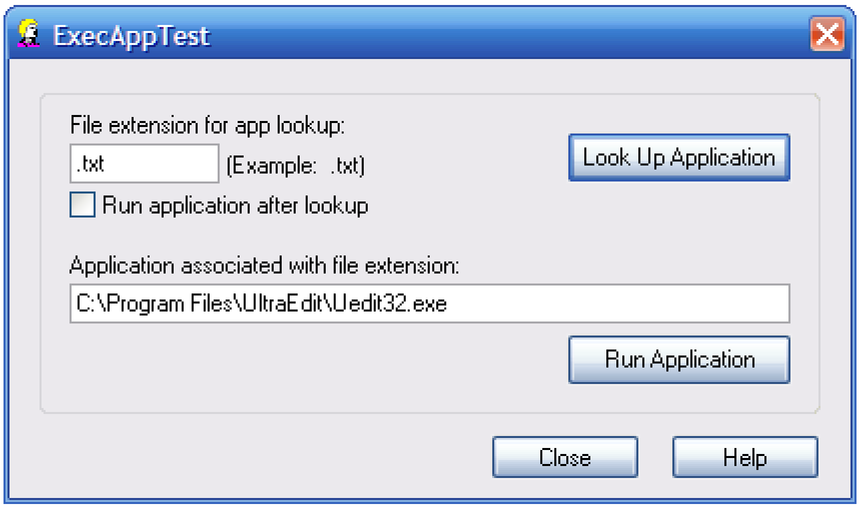
Here is function header of `ExecRegisteredApp()`:

```
///////////////////////////////////////////////////////////////////////////////
//
// ExecRegisteredApp()
//
// Purpose:     Runs the application registered for the specified file extension
//
// Parameters:  lpszArgs - [in] command line arguments that are passed to app
//                         via CreateProcess();  if not already in quotes ("),
//                         they will be enclosed in quotes before CreateProcess()
//                         is called.  May be NULL.
//              lpszExt  - [in] file extension (e.g., ".txt") used to look up
//                         application file path.  Preceding '.' is necessary.
//              wShowCmd - [in] Specifies how the app window is to be shown.
//                         See ShowWindow() in MSDN for possible values.
//
// Returns:     BOOL     - TRUE = found registered app; CreateProcess() succeeded
//
BOOL ExecRegisteredApp(LPCTSTR lpszArgs,    // may be NULL
                       LPCTSTR lpszExt,
                       WORD wShowCmd /*= SW_SHOWNORMAL*/)
```

Is `ExecRegisteredApp()` a replacement for ShellExecute? No, absolutely not. If you're 100% sure that a file type (for example, .*html*) is registered on the target system, then you should certainly use ShellExecute. But there are times when you can't be sure, and it's not possible to register a custom file type. This is where`ExecRegisteredApp()` is useful.

ExecApp Demo

Here is what the **ExecApp** demo looks like:


ExecAppTest
File extension for app lookup:
.txt
(Example: .txt)
Run application after lookup
Look Up Application
Application associated with file extension:
C:\Program Files\UltraEdit\Uedit32.exe
Run Application
Close
Help

How to use

Step 1 - Add Files

To integrate **ExecApp** into your app, you first need to add following files to your project:

- *ExecApp.cpp*
- *ExecApp.h*

The *.cpp* file should be set to **Not using precompiled header** in Visual Studio. Otherwise, you will get error

```
fatal error C1010: unexpected end of file while looking for precompiled header directive
```

Step 2 - Add Header File to Your Source Module

In the module where you want to use **ExecApp**, include header file *ExecApp.h* .

Step 3 - Add Code

See *ExecAppTestDlg.cpp* for examples of how to use.
The file *ExecApp.cpp* contains definitions necessary to use the`AssocQueryString` API. If you have the latest Platform SDK, you can use that instead of the embedded definitions by un-commenting the line **#include <shlwapi.h>** in *ExecApp.cpp*.

Revision History

Version 1.0 - 2008 June 8

- Initial public release

Usage

This software is released into the public domain. You are free to use it in any way you like, except that you may not sell this source code. If you modify it or extend it, please to consider posting new code here for everyone to share. This software is provided "as is" with no expressed or implied warranty. I accept no liability for any damage or loss of business that this software may cause.

License

About the Author

Hans Dietrich

Software Developer (Senior) Hans Dietrich Software

United States

I attended St. Michael's College of the University of Toronto, with the intention of becoming a priest. A friend in the University's Computer Science Department got me interested in programming, and I have been hooked ever since.

Recently, I have moved to Los Angeles where I am doing consulting and development work.

For consulting and custom software development, please see**www.hdsoft.org**.