The easy way out is to just answer the question: What is the current Explorer window looking at?

devblogs.microsoft.com/oldnewthing/20101126-00

November 26, 2010

Raymond Chen

A customer had the following question:

> We have an application which copies and pastes files. Our problem is that we want to paste the files into the folder which corresponds to the currently active Windows Explorer window. Right now, we're using `SendKeys.SendWait("^(v)")`, but we find this method unsatisfactory because we want to replace Explorer's default file copy engine with our own custom one. Can you provide assistance?

(And commenter wtroost had no clue why somebody would copy a file by sending window messages to Explorer. Well here you have it.) The easy way out is to answer the question: You can enumerate the active Explorer windows and ask each one what folder it is viewing. There's even a script interface for it. The hard way out is to understand the customer's problem and see if there's a better solution. The question as phrased suggests that the customer hasn't thought through the entire problem. What if the current window is not an Explorer window, or if it's a window on a virtual folder instead of a file system folder (for example, an FTP site)? Simulating keyboard input (in this case, fake-pressing Ctrl+V) is rarely a good solution to a problem; after all, what if the hotkey for Paste changes based on the user's preferred language? Or what if the Explorer window happens to be in a state where Ctrl+V doesn't paste files into the current folder? (For example, focus might be on the Address Bar.) And the fact that they put contents onto the clipboard means that they are overwriting the previous contents of the clipboard. I asked for a little more information about what their application is trying to do.

> This is a file transfer application for computers which are not directly connected to each other, but which are both connected to a common third computer. From the first computer, you run the file transfer application, select some files from the transfer application's interface, and hit Copy. This transfers the files to the common third computer. Then from the second computer, you run the file transfer application and hit Paste, and the program retrieves the files from the common third computer and places them in the folder that you are currently viewing in Windows Explorer.

Oh, the whole "get the path to the folder that Windows Explorer is viewing" is just a strange way of telling the program where to copy the files. In other words, they were using Windows Explorer as a very expensive cross-process replacement for the `SHBrowseForFolder` function.

The recommendation therefore came in two parts:

1. Instead of hijacking Explorer as a directory-picker, just call `SHBrowseForFolder`. You can pass the `BIF_RETURNONLYFSDIRS` flag, and `SHBrowseForFolder` will automatically filter out anything that is not a file system folder, thereby saving you the trouble of filtering them out yourself.
2. If you really want to hijack Explorer as a directory-picker, then add a context menu command to `Directory` or `Directory\Background` called *Paste from Transfer Shelf* (or whatever your application calls that intermediate computer).

Raymond Chen

Follow