There can be more than one (or zero): Converting a process to a window

devblogs.microsoft.com/oldnewthing/20080220-00

February 20, 2008

Raymond Chen

A common question I see is "How do I find the window that corresponds to an `HINSTANCE`?" This question comes pre-loaded with the assumption that there is only one window that corresponds to an `HINSTANCE`, which is true for only the most rudimentary of Win32 programs. Even a simple program like Notepad has more than one window with the same `HINSTANCE`, as Spy quickly reveals. But when I hear this question, I smell something suspicious. First, instance handles are meaningful only when you specify which process you're referring to, since an instance handle in Win32 is the base address of a module, and different processes can have different DLLs loaded at the same base address (thanks to separate address spaces). Second, I can't think of any normal scenarios where you'd even care about finding all the windows that have a particular instance handle, especially since the instance handle is ignored if the window belongs to a global class, and global classes are typically the only classes you are interested in anyway when you go snooping into another process. (After all, if it's not your process, you don't really know much of anything about its private classes.) But you probably already know that the person asking the question is asking the wrong question. They almost certainly launched a program with the `ShellExecute` function and now want to locate its window so they can do, um, *something* with it. But you already know that you can't do anything meaningful with the `HINSTANCE` returned by the `ShellExecute` function aside from compare it against the number 32, since it's not really an instance handle on Win32. That the return value is of type `HINSTANCE` is just a carry-over from 16-bit Windows. But the person who asked the question never got that far in reading the documentation and just said, "Well, I've got an `HINSTANCE` and I need a window, so can somebody help me convert this `HINSTANCE` to a window?" without understanding what's really going on. They need Y, but have X (a fake X, but still), so obviously the thing they need next is a way to convert your X to a Y. Even if it doesn't make sense. It's like asking for double-strength placebos. Okay, so you get past the initial disconnect of asking for the wrong thing. Say you have a process ID or a thread ID and you want to find the window for that process or thread. Again, there can be more than one, and there might be zero. If it's a process you're after, you can enumerate the windows on the desktop and use the `GetWindowThreadProcessId` function to determine what thread and process they belong to. If you are interested in the windows that belong to a thread, you can use the

`EnumThreadWindows` function. Your next task is deciding which of those windows is the one you want. When you try to explain this to people, you sometimes get stuck in the *Why won't you answer my question?* cycle. "I have a thread ID. How do I get the corresponding window?" *You can use the* `EnumThreadWindows` *function to get all the windows on the thread.* "Yes, I know about `EnumThreadWindows`, but how do I get the window that I want?" *Well, you haven't said what you wanted yet.* "I want the window that corresponds to the thread." *But which one? How will you decide among all the windows?* "That's what I'm asking you!" *But you haven't yet described what you want.* "I want the window corresponding to the thread. Why won't you answer my question?" Note that saying, "I am looking for the top-level unowned window" is a step forward, but it still doesn't uniquely identify a window. There can be multiple top-level unowned windows in a process. For example, Explorer typically has lots of top-level unowned windows. There's the desktop, the taskbar, your open folder windows, and property sheets. If you ask for "the" top-level unowned window of Explorer, which one do you want?

Perhaps people are getting the idea that there is a way to uniquely specify "the" window for a process because the `System.Diagnostics.Process` object has a property called `MainWindowHandle`. The documentation for that property doesn't do anything to dispel the notion, either. I have no idea how that property decides among multiple top-level unowned windows.

Raymond Chen

Follow