Having an owner window from another process is tricky, but it's sometimes the right thing to do

devblogs.microsoft.com/oldnewthing/20110331-00

March 31, 2011

Raymond Chen

A customer had a main program (let's call it A) and a helper program (let's call it B), and the customer wanted and wanted B to act like a modal dialog relative to A.

> When B is launched, we disable A's window and then call `SetForegroundWindow(hwndB)` to simulate a modal dialog. How do we make sure that focus goes to B's window and not A's? We've found that if the user clicks on the (now-disabled) window from the process A, then window B loses focus. This is not the behavior from regular modal windows however: For normal modal windows, clicking on the disabled owner activates the modal popup.
>
> One idea is to watch for `WM_ACTIVATE(FALSE)` notifications on `hwndSecondProcess`, and if the window that took focus from us is the one from the first process, then take it back with SetForegroundWindow(hwndSecondProcess).
>
> But then we wondered, since we disabled window A, will it even get the normal activation message?

Since the window is disabled, it will not receive activation messages because disabled windows cannot be activated. So no, this solution won't work. The subject line of the question, however, gave the answer without even realizing it. The subject was *Out-of-proc pseudo-parent/child window relationship*. (Well, okay, the subject line confused parent/child with owner/owned, but that's a common source of sloppiness when talking about the relationship among windows.) Instead of having a pseudo-owner/owned window relationship, just have a real one. Why fake it when you can get the real thing?

When you call `DialogBox` in process B, pass `hwndA` as the owner window. Now the two windows have a genuine owner/owned relationship, along with the standard behaviors that come with it. It's legal to have an owner/owned relationship that crosses process boundaries. Note that when you do this, it attaches the two threads' input queues so you have to be careful if both windows process input at the same time. Fortunately, in the modal dialog case, only one of the windows accepts input at a time, so the scariest part of attached input queues doesn't apply.

Raymond Chen

Follow