Why does my control send its notifications to the wrong window after I reparent it?

devblogs.microsoft.com/oldnewthing/20100316-00

March 16, 2010

Raymond Chen

Commenter MontagFTB noticed that some controls have the problem that if you reparent the control, it still sends notifications to its old parent. We looked at the faulty diagnosis last time. What's the real reason?

The control cached its original parent window.

Most complex controls communicate with the parent window frequently, and in order to avoid calling `GetParent`, the control gets its parent once and caches the value for future use. Under normal conditions, this cache works very well since reparenting a window is extremely rare and is generally considered an unusual condition. Like the adoption of a child, it's the sort of thing you should only be doing with the coordination of all three parties (the old parent, the new parent, and the child).

When you reparent the control, the cached value in the child window is no longer correct. But since you didn't coordinate this with the child window, the control doesn't know this, and it keeps talking to the old parent. Unlike the Post Office, you can't submit a change of address form to the window manager and tell it, "Hey, if somebody tries to send a message to windows X, deliver it to window Y if the return address is window Z." (Actually, the Post Office stops forwarding mail after one year.)

Since window reparenting is considered to be an unusual condition, most controls don't have provisions for telling them, "Hey, I reparented you. Please send future notifications to that window over there." The window manager is fine with all your reparenting games, but the other participants may have made assumptions based on the stability of the window hierarchy.

Where does that leave MontagFTB? (It is at this point where a general topic gradually turns into addressing questions that are applicable only to MontagFTB's situation and aren't all that useful to others. This is something I try to avoid, because this is a blog, not a consulting service.)

First, you can avoid the staging window and just create the controls with the correct parent. I don't know why the staging window was necessary, so this may not be a viable solution. If it was merely to avoid flicker, then you can create the controls as hidden windows, and then do a massive `ShowWindow` when they are ready. Or you can create the controls at negative coordinates (so they don't appear inside the parent's client rectangle), and then when you're ready, perform a big `EndDeferWindowPos` to move them all into position at once.

If you really need to have the staging window, you can have the staging window do the message forwarding. If it receives a `WM_COMMAND` or `WM_NOTIFY` notification message from one of these given-away child windows, it just forwards the message to the new owner. However, this violates the guideline that "When reparenting a window, the old parent, the new parent, and the child all need to be involved in the process if you want the adoption to go smoothly," so I would not recommend it.

If you don't want to make the staging window have to deal with message forwarding (for example, if you intend on destroying the staging window once you have removed all the child windows), then you can insert a level of redirection: Create a container window as a child of the staging window, and create the child windows as children of the container. Then when it's time to reparent the controls, move the container window to the new parent. This adheres to the guideline because the windows involved in the reparenting (the final destination, the staging window, and the container window) are all under your control, and therefore you can make sure all internal state is correct when you change the bookkeeping relationship among them. And since the controls are destined for a dialog box, you should give the container the `WS_EX_CONTROLPARENT` style so that they can participate in dialog box navigation.

Raymond Chen

Follow