Who decides what can be done with an object or a control?

devblogs.microsoft.com/oldnewthing/20080205-00

February 5, 2008

Raymond Chen

This is one of those things that is obvious to me, but perhaps is not obvious to everyone. An object establishes what can be done with it. Any rights granted by the object go to the creator. The creator can in turn grant rights to others. But if you're a third party to the object/creator relationship, you can't just step in and start messing around without the permission of both the object and the creator. For example, unless you have permission of the creator of a list view control, you can't go around adding, removing, and changing items in the list view. The creator of the list view decides what goes in it, and at most you can look at it to see what the creator decided to put in it. I say "at most" because you often can't even do that: If the fact that the item is a list view control is not public, then the program that created the list view might decide to use some other control in a future version, and then your code that tries to look at the list view will stop working. Naturally, any private data associated with a control (such as the `LPARAM` associated with a list item) is entirely under the purview of the control's creator. The creator of the control decides what the item data means, and that can change from version to version if not explicitly documented. (For example, the meaning of the item data associated with the main list view in Explorer windows has changed at pretty much every major release of Windows.) Generally speaking, without the permission of the creator of a control and the control itself, you can't do anything to it. You can't hide it, show it, move it, change its scroll bars, mess with its menus (assuming it even uses menus at all), change its text, or destroy it. The code that created the control and the control itself presumably maintain their own state (the control maintains state about itself, and the creator maintains state about what it has done to the control). If you start messing with the control yourself, you may find that your changes seem to work for a while and then are suddenly lost when the control decides to re-synchronize its internal state with its external state. Or worse, things will just be permanently out of sync, and the program will start acting strange. If the control and the control's creator have not provided a way to coordinate your actions with them, then you can't mess with the control.

If you're tempted to go mess with somebody else's controls, think about how you would like it if the tables were turned. How would you feel if somebody wrote a program that took the controls in your program and started messing with them, say adding items to your list box

and using the item data to extract information out of your program? What if they started filing bugs against you for changing the way your program operates internally?

Raymond Chen

Follow