When something is available for the user, which user are we talking about?

devblogs.microsoft.com/oldnewthing/20061010-06

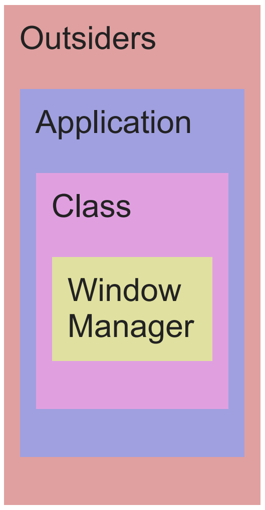
October 10, 2006

Raymond Chen

Some people have taken issue with the term `WM_USER` for the name of the base of the message range that is the province of the window class implementor. (Refresher for those who forget which messages belong to whom.) The complaint is that the user **can't** use them since they belong to the window class. Aha, but the real question is, "Who is the user?" In other words, when we say "user-defined", who is the user that's doing the defining? There are four components involved here, and each one gets its own message range.

- The window manager gets the messages below `WM_USER` .
- The class implementor gets the messages from `WM_USER` to `WM_APP` -1.
- The application (more specifically, the code that creates the window) gets the messages from `WM_APP` to `MAXINTATOM-1` .
- Everybody else gets the messages from `MAXINTATOM` to `MAXWORD` (via `RegisterWindowMessage`).

Who is the user? It depends on your point of view. From the window manager's point of view, the class implementor is the user, since that's the code that is using the window manager. From the class implementor's point of view, it is the application that creates the window that is the user, since that's the code that is using the window class. And from the application's point of view, it's all that outside code that is the user, since that's the code that is using the application. Conversely, you can look "up" the list and observe that from each component's point of view, the messages that belong to the components above it are "reserved". From the class implementor's point of view, the window manager messages are reserved. From the application's point of view, the class implementor's messages are reserved. And from the outsider's point of view, the application's messages are reserved. Let's use one of those generic box diagrams.



Each box considers the messages that belong to its inner boxes as reserved; conversely, each inner box considers the next outer box as its "user". Since the `winuser.h` header file was written by the window manager team, it's not surprising that they look at the world from the window manager's point of view. Therefore, everything outside the window manager is "user-defined" and everything inside the window manager is "reserved".

Of course, if you look at things from the point of view of the class implementor, then the context of the words "reserved" and "user" changes. You can see this, for example, in the dialog box constant `DWLP_USER`, which is the index of window bytes that can be used by the "user" of the dialog box; i.e., by the application's dialog procedure.

Raymond Chen

Follow