Safer subclassing

devblogs.microsoft.com/oldnewthing/20031111-00

November 11, 2003

Raymond Chen

So what was wrong with our little subclassing sketch?

Most people figured this out.

Consider what would happen if somebody else had subclassed the window during the "... do stuff ..." section. When we unsubclassed the window, we would have removed **two** subclasses, the one we installed, and the one that was installed after us. If the other subclass allocated memory (which is very common), then that memory got leaked, in addition to the subclass failing to do whatever it was trying to do.

Do not assume that subclasses are added and removed in a purely stack-like manner. If you want to unsubclass and find that you are not the window procedure at the top of the chain **you cannot safely unsubclass**. You will have to leave your subclass attached until it becomes safe to unsubclass. Until that time, you just have to keep passing the messages through to the previous procedure.

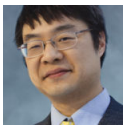
This is quite a cumbersome process, so the shell team wrote some helper functions to do all this for you. The SetWindowSubclass function does all the grunt work of installing a subclass procedure, remembering the previous one, and passing reference data to the subclass procedure you provide. You use the DefSubclassProc function to forward the message to the previous subclass procedure, and when you're done, you use the RemoveWindowSubclass function to remove yourself from the chain. RemoveWindowSubclass does all the work to do the right thing if you are not the window procedure at the top of the chain.

One gotcha that isn't explained clearly in the documentation is that **you must remove your window subclass before the window being subclassed is destroyed**. This is typically done either by removing the subclass once your temporary need has passed, or if you are installing a permanent subclass, by inserting a call to RemoveWindowSubclass inside the subclass procedure itself:

```
...
case WM_NCDESTROY:
  RemoveWindowSubclass(hwnd, thisfunctionname, uIdSubclass);
  return DefSubclassProc(...);
```

One comment expressed concern that a message could be sent between the call to `SubclassWindow` and the store of the previous window procedure into the `OldWndProc` variable. This is actually a safe operation provided that you are doing the work from the thread that owns the window you are subclassing. Remember that message delivery occurs only when the thread is in a receiving state, such as when it calls `GetMessage` or `PeekMessage`. If somebody sends a message when the thread is not in a receiving state, the message merely waits until the thread finally calls `GetMessage` (for example) before being delivered. Since we don't make any message-receiving function calls between the `SubclassWindow` and the store into `OldWndProc`, there is no risk of an untimely message arriving before the store to `OldWndProc` has occurred.

There was another comment that claimed that the `SubclassWindow` macro is undocumented. Actually this macro is so old that the documentation for it has faded almost into obscurity. (You forced my hand; I wasn't going to dig into this header file until tomorrow!)



Raymond Chen

Follow