# Changing a window class affects all windows which belong to that class

devblogs.microsoft.com/oldnewthing/20060227-12

February 27, 2006

Raymond Chen

Sometimes you need to state the obvious, and that's fine. You can learn a lot from the obvious. For example, the first topic in my PDC talk consisted of simply stating the obvious. Occasionally, when you state the obvious, you have to follow up by stating the obvious. When I pointed out that the effect of `SetCursor` lasts only until the next `SetCursor`, one commenter proposed as an alternative solution to the problem of how to prevent the cursor from reverting to the arrow,

> What about calling `SetClassLongPtr()` with the `GCLP_HCURSOR` param? Is this an acceptable alternative, or does this have issues of its own?

Being busy is a state of a particular window instance, not a state of the window class. Therefore, the setting of the cursor must be done by the window instance, not by the window class. If you change a window class, then it affects all windows which belong to that class. In particular, changing the class cursor changes the cursor for all windows that belong to the class. In other words, consider the case where you have two windows on the screen that belong to the window class. One of the windows becomes busy, so you change the class cursor to an hourglass. When the user moves the mouse over the window that is busy processing, the cursor turns to an hourglass, as you would expect. But when the user moves the mouse over the window that is not doing any processing, **the cursor will turn to an hourglass there also**, even though that window is perfectly fine.

It is important to understand the scope of what one is changing. Don't make a global change to solve a local problem.

Raymond Chen

**Follow**