The dangers of filtering window messages

devblogs.microsoft.com/oldnewthing/20050209-00

February 9, 2005

Raymond Chen

The `GetMessage` and `PeekMessage` functions allow you to pass a filter, restricting the window handle or range of messages that the function will retrieve from the message queue. While it's okay to use these filters, make sure you eventually get around to making an unfiltered call so that any straggling messages can come through.

A common mistake is to use a window-filtered GetMessage in your message loop. In our scratch program, a window-filtered `GetMessage` would look like this:

```
while (GetMessage(&msg, hwnd, 0, 0)) { // Wrong!
    TranslateMessage(&msg);
    DispatchMessage(&msg);
}
```

Even though the program creates but one window, this program is nevertheless incorrect.

"How can this be?" you ask. "My program has only one window. Why would there possibly be any messages for any other windows? The filter, while redundant, isn't harmful, is it?"

Many system services create windows on your behalf. For example, if input method editing is enabled, the method editor may create helper windows to assist in character input. If you initialize COM, then COM may decide to create a helper window to assist in inter-thread marshalling. If you use only a filtered GetMessage, then messages destined for these helper windows will never be retrieved, and you will be left scratching your head wondering why your program occasionally hangs when it tries to perform a drag/drop operation, for example.

Moral of the story: Make sure your message loop eventually performs an unfiltered message retrieval so that these services can operate properly.

Raymond Chen

Follow