Why isn't there a SendThreadMessage function?

devblogs.microsoft.com/oldnewthing/20081223-00

Raymond Chen

Here's an interesting customer question:

> Windows has `PostMessage` and `SendMessage`. It also has `PostThreadMessage` but no `SendThreadMessage`. Why isn't there a `SendThreadMessage` function? Am I forced to simulate it with an event?

What would this imaginary `SendThreadMessage` function do? Recall that `SendMessage` delivers the message directly to the window procedure; the message pump never sees it. The imaginary `SendThreadMessage` function would have to deliver the message directly to…. what? There is no "thread window procedure" to deliver it to.

Okay, maybe you still intend to process the thread message in your message pump, but you want the caller of the imaginary `SendThreadMessage` function to wait until you've finished processing the message. But how does it know when you're finished? It can't wait for `DispatchMessage` to return, since `DispatchMessage` can't dispatch thread messages. (Where would it dispatch them *to*?) The processing of the thread message is completely under the control of the message pump. The window manager gives it a thread message, and as far as the window manager is concerned, that's the end of the story.

You might say that the processing of the thread message is complete when somebody next calls `GetMessage` or `PeekMessage`, but there's no guarantee that the next call to a message-retrieval function will come from the message pump. Handling the thread message may result in a call to `MessageBox`, and as a modal function, it will have its own message loop, which will call `GetMessage`, resulting in your imaginary `SendThreadMessage` function deciding that message processing is complete when in fact it's still going on.

What should you do instead? Just create a window and send it a message. The scenarios where you would want to use the `PostThreadMessage` function are very limited and specialized. Under normal circumstances, you should just send a regular window message.

Raymond Chen

Follow