

Re: Batch file and MFC (Properly Terminating Application)

Source: <http://www.tech-archive.net/Archive/VC/microsoft.public.vc.mfc/2007-02/msg00303.html>

- *From:* "Alexander Grigoriev" <alegr@xxxxxxxxxxxxxx>
 - *Date:* Sun, 4 Feb 2007 10:17:41 -0800
-

Actually, there is no sure safe way to close your application in an arbitrary place even in your own code.

WM_CLOSE can only be safely posted to your main window, if you're not running any secondary modal loop. If you're displaying a dialog, WM_CLOSE will be handled before your DoModal() returns, thus if the dialog was called from any member function of the main frame, the return path may touch deleted window object.

WM_QUIT (explicitly posted or result of PostQuitMessage()) breaks the current modal loop. If it was a secondary loop, the main loop will continue to run. IIRC, MFC reposts WM_QUIT in such a case, to propagate shutdown.

Another tricky case is handling WM_QUERYENDSESSION/WM_ENDSESSION when a modal dialog is displayed. Do you have such a test case for your apps?

"Joseph M. Newcomer" <newcomer@xxxxxxxxxxxxxx> wrote in message news:sd0bs2d8a7pp89or9irsg584j3bpo4gdto@xxxxxxxxxxxx

For the cancel button, you would invoke the OnCancel() handler for the dialog. If you don't write an OnCancel handler, the CDialog::OnCancel handler is invoked (OnCancel is a virtual method invoked by the dialog receiving a WM_COMMAND(IDCANCEL) notification. The CDialog::OnCancel handler will do an EndDialog(IDCANCEL) call, which will, within Windows, ultimately issue a WM_QUIT, but you never see that. Nor should you.

The problem with WM_QUIT is that it stops the message pump dead. So no more messages can be processed. What about that message that says "Prompt the user to ask if all data should be saved"? Sorry. What about the clean shutdown of all threads? Sorry. What about the sequence of messages handling the clean shutdown of the database or network.

Re: Batch file and MFC (Properly Terminating Application)

Sorry. They're all gone. The program exits, leaving all kinds of things broken. There are no protocol messages sent to the server about the network connection closing. There is no chance to write out any buffered data, or properly close files (while the handles will be closed, data which has been buffered in the app isn't flushed because that only happens when you call the high-level close function, such as `fclose`, so the files are left in indeterminate and possibly corrupt state). The server might do a rollback—losing all the changes the user thought had been made—because the app closed without giving it a clean close on its network connections. You bypass all possibility of any recovery mechanism being viable for any purpose. Since you have no idea what is really required by `WM_CLOSE`, this has all the value of doing `ExitProcess()`, which is to say, it is about as clean as writing `(LPBYTE)0 = 0`; except that the null pointer assignment actually generates a useful notification that the program is screwed up, whereas `WM_QUIT` quietly screws up.

Don't confuse the mechanism for exiting the message pump with the policy for exiting the message pump. The framework is responsible for implementing the policy; you just don't know how it does it. The implementation is ultimately a `WM_QUIT` message, but that's about as important as knowing the `ebx` register holds the value of `i` during the for-loop during one particular loop in one particular subroutine.

`WM_QUIT` is handled internally by the MFC framework, AFTER all relevant close operations have been performed, You never issue it in the main GUI thread yourself. If you want to close a non-dialog app, do a `PostMessage(WM_CLOSE)` to the main window. If you want to close a dialog app, call `CDialog::OnOK` or `CDialog::OnCancel` (depending on which kind of termination you want), but only if the user has invoked an operation that is supposed to close the app.

You actually DO use `WM_QUIT` to shut down secondary UI threads, but that's a different issue. And it must be done with care, preferably from within the thread

Re: Batch file and MFC (Properly Terminating Application)

when it
determines that it is in a clean state to shut down.
joe

On 3 Feb 2007 19:34:42 -0800, "one-trick-pony"
<worldofpain.aamir@xxxxxxxxxx> wrote:

ABSOLUTELY POSITIVELY WRONG!!!! NEVER, UNDER ANY
CIRCUMSTANCES, POST
A WM_QUIT MESSAGE
FROM WITHIN THE MAIN GUI THREAD!!!! Just pretend you never
heard of
WM_QUIT.

I am confused about this line. Lets assume, I create the simplest
type of Dialog based application with 2 buttons. One button caption
is Hello and second button is Cancel. When user clicks Hello button a
simple message box pops up and says Hello There! ...easy enough. Now
user clicks on Cancel button and application exits. The two buttons
are created by default in MFC Dialog based application -Ok and
Cancel . Assuming, I deleted Ok button and didnt write any code to
support the Cancel button. Cancel button is created by default and it
always makes application exit without having to code it. So when I
click on it, there was only 1 main thread and I exited out of Main GUI
thread. It seems your comment is wrong but I know there is gap in my
understanding. I thank you for your help.

Joseph M. Newcomer [MVP]
email: newcomer@xxxxxxxxxxxxx
Web: <http://www.flounder.com>
MVP Tips: http://www.flounder.com/mvp_tips.htm