

Re: Preventing an application from staying in memory if closed improperly

Source:

<http://www.tech-archive.net/Archive/VB/microsoft.public.vb.general.discussion/2007-05/msg02264.html>

- *From:* "Scott M." <quanker@xxxxxxxx>
 - *Date:* 28 May 2007 18:40:24 -0700
-

On May 28, 3:32 pm, "Mike Williams" <m...@xxxxxxxxxxxxxxxxxxxx> wrote:

"Scott M." <quan...@xxxxxxxx> wrote in message

news:1180375637.628071.323850@xx

Hi Bob, I open database queries, grab my data and quickly close them again. It is possible for the user to hit the X during that time. I think disabling the X might be a good option. If the user has to hit my "End Program" button, then at least I can disable it during critical operations like reading from a database

Actually I never thought of that when I posted my last response, Scott. I never "do" database stuff and so I didn't realise it could cause a problem under such circumstances. So perhaps preventing the user from closing the Form with the X button might be a good idea in your own case. There's just one point though that I'd like to mention about the "disable the Command Button" thing you suggested that you may not know about and that will trip you up if you don't . . .

. . . if you have a block of code that sets a Command Button's Enabled property to False and then *in the same code block* carries out some tasks or calls a Sub or a Function to carry out some tasks, the button appears to the user to fail to respond to his clicks while those tasks are being carried out, but actually all the user's clicks on that button are "queued" and as soon as your code block sets the Command Button's Enabled property back to True then all of those queued click events will be actioned, one after the other, and they will be actioned at a time when the Command Button's Enabled property is set to True. This can cause problems because it will cause the code in the Click event to be executed as many times as the user clicked the button while it was disabled. The way to overcome the problem is to use a DoEvents line immediately before you set the Enabled property back to True. This will cause VB to "action" all of the queued click events on that button, and it will of course do so at a time when the Enabled property is still set to False (because the DoEvents line is

Re: Preventing an application from staying in memory if closed improperly

immediately before the Enabled = True line). This will effectively mean that all of the "queued clicks" are dispensed with without causing the Command button's click event code to actually execute (which is what you want).

Sometimes it will be a problem in your code and sometimes it will not, depending on exactly what it is you are doing and exactly where you are using the "False" and "True" settings for the Command Button, but generally after setting a Command Button's Enabled property to False it is always a good idea to set it back to true in the following manner:

```
DoEvents  
Command1.Enabled = True
```

Mike

It's a simple solution to add the DoEvents line. Thanks for the great idea.

Scott