

Re: Locking Desktop and Threading Issue

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.framework/2007-05/msg00023.html>

- *From:* Jenbo <eamonnjennings@xxxxxxxxxxxx>
 - *Date:* 1 May 2007 12:27:09 -0700
-

Thanks for the advice, basically my forms cover the full desktop, they are survey forms that the user takes an action against, HR related stuff. I can get the forms to cover the full desktop in normal processing, I have a class that does this ok, but if for example the user locks the desktop and the application creates a form when the user unlocks the desktop the form does not show on the full desktop and the windows taskbar is shown, I realise this is probably normal behaviour for windows, this is why I wanted to sleep the application, so it can no process the form and then when it wakes up again the application can create the normal form covering all the windows desktop. So if I sleep the thread I can then wake it and process as normal when the session event is unlock. So maybe I need to let it process as normal and go back to my windows handler for the covering of the desktop and check that piece,

Thanks
Eamonn

On 1 May, 18:41, "Peter Duniho" <NpOeStPe...@xxxxxxxxxxxxxxxxxxxx> wrote:

On Tue, 01 May 2007 09:35:44 -0700, Jenbo <eamonnjenni...@xxxxxxxxxxxx> wrote:

Hi, I have an application that displays forms on the users desktop, what I am doing with it is having a SystemEvents SessionSwitchEventHandler to capture the lock of the desktop, I would like to kill the forms when this event is captured, that piece is fine, but I would like to sleep the app until the SessionSwitchReason.SessionUnlock happens then I can wake it again and go the correct part of the code? Can I wake a sleeping thread, or should I not sleep the thread at all, is there another way to get the thread to sleep or hibernate and then wake u

It's not clear to me that there's really any need to do any of the things you want to do. What are your forms doing that you can't just leave them open when the desktop is locked? Windows will take care of hiding them

Re: Locking Desktop and Threading Issue

from the user and preventing user input, and when the desktop is unlocked then they will still be there, ready to go. Windows is also smart enough to not dedicate any attention to the user interface of the application while it's essentially inactivated by the desktop being locked (not that the UI component of an application is generally a significant load on the computer in the first place).

That said, it is possible to explicitly get a thread to simply stop doing anything. Under Windows, a thread cannot "sleep" or "hibernate" per se (not in the sense that the computer can do those things), but you can create a waitable event and call the `WaitOne()` method on that. The thread that calls that method will stop and wait at that call until some other thread sets the event, releasing the thread for further execution.

If you use a `ManualResetEvent`, and you insert a call to the event's `WaitOne()` method at some appropriate spot (generally a point in some code's processing loop that is executed often enough that latency for any pausing of the thread will be low, but not so often that calling `WaitOne()` causes a significant performance problem), then when you want the thread to pause you reset the event and when you want the thread to continue to set the event.

Of course, you still need at least one thread around to process the `SessionSwitchEvent` and manage any other threads that are blocked. But I suppose if you have a thread that is doing something particularly i/o or CPU intensive and you want it to stop taking system resources when the desktop is locked, you could build in this sort of pausing mechanism without affecting the main GUI thread.

If the above doesn't help you, perhaps you could be more explicit about why it is you believe you need to "put your application to sleep" when the desktop is locked. This isn't something that normally needs to be done, so you should probably be more clear about what abnormal situation exists in your scenario that justifies or demands that sort of behavior.

Pete