

Re: Blocked threads vs Thread.Interrupt

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.csharp/2004-03/8243.html>

From: TT \(\Tom Tempelaere\) (_N_OSPA/\|titi____at_hotmail.com/\|APS0_N_)

Date: 03/30/04

Date: Tue, 30 Mar 2004 06:55:45 GMT

Hi Dave,

"Dave" <noSpamdlevineNNTP2@wi.rr.com> wrote in message
news:eo20NEfFEHA.2576@TK2MSFTNGP11.phx.gbl...

[...]

> *That being said, neither mechanism is the one I currently prefer. When I
> launch a thread I usually have it run in a loop that blocks waiting on
> multiple events; when one occurs it wakes up, does it work, and then goes
> back to sleep. One of those events is a ManualResetEvent which I use as a
> signal to exit the thread. This ensures that everything is synchronized
and
> nothing vital will get interrupted.*

This is what I do now.

But if I read correctly, Thread.Interrupt is actually ideal for waiting
situations. If I would make a thread that waits until this or that, then
Thread.Interrupt seems the only way to unblock if you don't have an explicit
shutdown event.

> *There are other issues with using an abort that are not obvious. One is
that
> if the thread is currently executing unmanaged then the abort will not
> actually be raised until it transitions back into managed code – this can
be
> an arbitrarily long time.*

Thread.Interrupt has the same problem I suppose.

> *Make sure that you set all worker threads to background threads to ensure
> your app can exit even if the thread is still running. Or use a threadpool
> thread.
>
> Dave*

Thanks,

microsoft.public.dotnet.languages.csharp: Re: Blocked threads vs Thread.Interrupt

Tom Tempelaere