

RE: Advanced Windows Form message priority / threading.

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.csharp/2004-11/6320.html>

From: Brian Keating EI9FXB (*csharp*)

Date: 11/29/04

Date: Mon, 29 Nov 2004 09:07:03 -0800

I've been thinking about this a little more,
an call to Control.Invoke is basically a function all onto my Form (control)
from the ThreadPool,

If i want my Invoke calls in my message queue I prob have to use events? or
maybe PInvoke on SendMessage etc?
Hopefully these will arrive on my message que and I can introduce a
prioritised system?

thanks
Brian

"Brian Keating EI9FXB" wrote:

```
> Hello again,  
> I've already placed a few posts on this topic.  
> This time i've a simple application that exhibits my problem,  
> I've placed sample solution 8k on my website should anyone be interested in  
> having a look. http://briankeating.net/transfer/test.zip  
>  
> To recap the problem I expected (and found).  
> I've a main GUI thead (main form), this GUI thread has an UpdateTextBox  
> function that appends a string in a textbox and It also has a button that  
> stops my worker thread(s) form calling Invoke on main thead.  
>  
> private void UpdateTextBox(string strText)  
> {  
> System.Diagnostics.Debug.Assert(InvokeRequired != true, "Invoke was  
> required", "calling begin invoke not on  
> the gui thread");  
> richTextBox1.Focus();  
> richTextBox1.AppendText(strText);  
> }  
>  
> When my form loads I start two worker threads and these threads just loop
```

microsoft.public.dotnet.languages.csharp: RE: Advanced Windows Form message priority / threading.

```
> around and call Invoke on my main Form with a string to display.
>
> private void ThreadEntryPoint()
> {
> while (!bStop)
> {
> string str = "Thread Id: " + GetHashCode() + ",\t" + nCount.ToString() + "
> This is a passing string\r\n";
>
> Invoke(new UpdateTextboxDelegate(UpdateTextBox), new object[] {str});
> Interlocked.Increment(ref nCount);
> //Thread.Sleep(1000); //slow sending
> Thread.Sleep(10); //fast sending
> } //end while
>
> MessageBox.Show(Thread.CurrentThread.ToString() + " thread closing");
> }
>
> Now if my worker threads are slow sending i.e. I call Thread.Sleep(1000);
> i.e. wait a seond between each invoke call on the gui thread, then I have no
> problem I can click my button and the event handler is called
>
> private void button1_Click(object sender, System.EventArgs e)
> {
> bStop = true;
> }
>
> However if my two threads are sending fast (only waiting 10 miliseconds)
> then my GUI thread is so busy processing these update events that if i click
> the stop button nothing happens.
>
> ... Hope that kinda clears things up about a GUI thread becoming unresponsive.
>
> So what are my options?
> I'm thinking that Inovke is basically like SendMessage so maybe somehow i
> can replace the windows message loop to do the following, (basically
> prioritised message queue)
> loop
> peek all windows messages and process them, (get/dispatch)
> then peek for one of my Inoke calls (get/dispatch)
> endloop
>
> Now people that have helped me already may be asking what has as this to do
> with windows running in different threads, well the reason I want to do this
> is, say for example i have 2 windows in my application, one window is busy
> receiving messages from a worker thread (so this is gonna become
> unresponsive), i wouldn't like all other windows to become unresponsive too),
> switching between windows will require my busy thread to handle some window
> messages (loose focus maybe etc) so a prioritised message queue will allow
> this, hey maybe this is enough I'll keep at it and see what happens.
>
```

RE: Advanced Windows Form message priority / threading.

microsoft.public.dotnet.languages.csharp: RE: Advanced Windows Form message priority / threading.

- > *But I guess this post comes down to one question,*
- > *Is there anyway I can plug into a windows forms message queue and check for*
- > *pending invoke calls as described above?*
- >
- > *Thanks for any help*
- > *Brian.*
- >
- > *p.s.*
- > *This is only a sample test app, so it's not perfect for many a reason,*
- > *I realise the two worker threads calling Invoke is pretty much the same as*
- > *One worker thread calling invoke.*
- > *clicking close on the forms gets executed straight away for some reason*
- > *(maybe it's handled differently?)*
- > *Clicking close will throw an exception cause i havn't stopped the worker*
- > *thread(s) from sending.*