

Forum:[Hack Forums](#)

Thread: [Easy way to prevent cross thread exceptions with Windows Forms](#)

gigajew - September 9, 2018, 12:57 pm

This is my code to prevent cross thread exceptions when (for example) setting the text of a textbox from another thread

Here's the code. You only need this one snippet instead of all kinds of delegates and crap.

This is some sample usage

SIM-3991524 - September 9, 2018, 7:30 pm

Honestly my head hurts everytime i see async. I much prefer to just initial Thread, set isBackground to True and Start()

0xDEAD10CC - September 9, 2018, 7:42 pm

(Sep 9, 2018 07:30 PM)f411en Wrote: [\[CITE\]](#) Honestly my head hurts everytime i see async. I much prefer to just initial Thread, set isBackground to True and Start()
Except that doesn't do the same thing at all as async/await.

As for the example, it doesn't make sense, but even if you wanted to use something similar to this, I wouldn't ever lock in an async method like that. That's potential for inadvertent deadlocks in my opinion.

BahNahNah - September 9, 2018, 9:11 pm

I normally just do something like this.

SIM-3991524 - September 10, 2018, 8:32 am

(Sep 9, 2018 07:42 PM)0xDEAD10CC Wrote: [\[CITE\]](#) Except that doesn't do the same thing at all as async/await.

As for the example, it doesn't make sense, but even if you wanted to use something similar to this, I wouldn't ever lock in an async method like that. That's potential for inadvertent deadlocks in my opinion.

hey dead, ik your into native winapi alot too. If this were native C/C++ would you use CreateThread? Obviously you will be more efficient with asynchronous call, but I think there's always a multithreaded solution that feels... cleaner

0xDEAD10CC - September 10, 2018, 8:44 am

(Sep 10, 2018 08:32 AM)f411en Wrote: [\[CITE\]](#) hey dead, ik your into native winapi alot too. If this were native C/C++ would you use CreateThread? Obviously you will be more efficient with asynchronous call, but I think there's always a multithreaded solution that feels... cleaner

It depends on what you're doing, sometimes it doesn't make sense to use a thread because you need to join the thread anyways along with potential thread context switching causing the overall flow of execution to be worse for performance than if you were to just do things on the same thread.

Ex: updating a single label on a thread to display 1 .. 100; you would need to halt threads that would display higher numbers until the lower numbers have been displayed on the label, it needs to happen sequentially anyways, and this isn't quantum mechanics, so it can't be 1..100 all at the same time either. If you spawned ~100 threads for each, you would probably see worse performance from having threads wait to be resumed at a later time.

SIM-3991524 - September 10, 2018, 8:57 am

(Sep 10, 2018 08:44 AM)0xDEAD10CC Wrote: [\[CITE\]](#) It depends on what you're doing, sometimes it doesn't make sense to use a thread because you need to join the thread anyways along with potential thread context switching causing the overall flow of execution to be worse for performance than if you were to just do things on the same thread. Ex: updating a single label on a thread to display 1 .. 100; you would need to halt threads that would display higher numbers until the lower numbers have been displayed on the label, it needs to happen sequentially anyways, and this isn't quantum mechanics, so it can't be 1..100 all at the same time either. If you spawned ~100 threads for each, you would probably see worse performance from having threads wait to be resumed at a later time.

Agreed agreed, obviously the convention exist for a reason. Ive also noticed my .net apps having a stupid high number of threads. Have you seen that at all? Ive had times where just the default winForm project had like 4 threads lol. Dont remember exact numbers

(Sep 9, 2018 09:11 PM)BahNahNah Wrote: [\[CITE\]](#) I normally just do something like this.
bahnanawa did you get RFed?

Nick (Name) - September 10, 2018, 10:47 am

I still don't get why people are still using WinForms, why not switch to WPF?

(Sep 10, 2018 08:57 AM)f411en Wrote: [\[CITE\]](#) bahnanawa did you get RFed?
Everyone's reputation got reset.

SIM-3991524 - September 10, 2018, 10:53 am

(Sep 10, 2018 10:47 AM)Nick (Name) Wrote: [\[CITE\]](#) I still don't get why people are still using WinForms, why not switch to WPF?

Everyone's reputation got reset.

I use winForms because its what i know and i can make UI's quickly in it. WPF is newer by all means

And wtf? when did that happen?

gigajew - September 10, 2018, 3:15 pm

(Sep 9, 2018 09:11 PM)BahNahNah Wrote: [\[CITE\]](#) I normally just do something like this.

Then you can also just

and save some rows

0xDEAD10CC - September 10, 2018, 3:19 pm

(Sep 10, 2018 03:15 PM)gigajew Wrote: [\[CITE\]](#) Then you can also just
Code public void InvokeForm(Action a) {
 this.Invoke(a);

}and save some rows
You're missing the point of the InvokeRequired bit there.

gigajew - September 10, 2018, 4:50 pm

(Sep 10, 2018 03:19 PM)0xDEAD10CC Wrote: [\[CITE\]](#) You're missing the point of the InvokeRequired bit there.

Doesn't matter in this case it'll be invoked either way so

0xDEAD10CC - September 10, 2018, 5:19 pm

(Sep 10, 2018 04:50 PM)gigajew Wrote: [\[CITE\]](#) Doesn't matter in this case it'll be invoked either way so

You're assuming the use case where someone is always using threads. That does not mean that the code you suggested as an alternative does the same as what he had, and I wouldn't suggest it just to "save lines of code" either.

gigajew - September 10, 2018, 5:46 pm

(Sep 10, 2018 05:19 PM)0xDEAD10CC Wrote: [\[CITE\]](#) You're assuming the use case where someone is always using threads. That does not mean that the code you suggested as an alternative does the same as what he had, and I wouldn't suggest it just to "save lines of code" either.

No you can use without threads too that's the whole point of the one liner instead of the if checks

0xDEAD10CC - September 10, 2018, 6:49 pm

(Sep 10, 2018 05:46 PM)gigajew Wrote: [\[CITE\]](#) No you can use without threads too that's the whole point of the one liner instead of the if checks

I know you can but what I was saying is that the code is not the same as your one-liner solution, and the fact that you don't really understand what is going on there. You're still one of those people that thinks LINQ is better because it allows you to write lots of one-liner code I presume too? ... ?

I don't know why you're suggesting a one-liner in his case either anyways, because if you applied that same logic to your example, all you'd have left is this:

```
Code await Task.Factory.FromAsync(  
    control.BeginInvoke(modifier, control),  
    control.EndInvoke  
);
```

I'd advise you to read these:

<https://stackoverflow.com/a/22409146>

<https://stackoverflow.com/a/12179408>

<https://docs.microsoft.com/en-us/dotnet/framework/winforms/controls/how-to-make-thread-safe-calls-to-windows-forms-controls>

This is also where people go wrong with this type of setup... They invoke the entire processing back on the UI thread instead of the code that just updates the UI itself which is mainly the reason why you would marshal back to the UI thread in the first place, but then it defeats the point of even doing something on a separate thread. Most people for that reason will look at this and use it incorrectly without understanding what they're doing, and it's not the best generic implementation to begin with either.

Delegate invocation is really the only syntactic sugar that would make such a call identical. (i.e. `modifier(...)` or `modifier.Invoke(...)`)

gigajew - September 10, 2018, 8:16 pm

<https://stackoverflow.com/questions/1423446/thread-control-invoke>

0xDEAD10CC - September 10, 2018, 8:23 pm

(Sep 10, 2018 08:16 PM)gigajew Wrote: [\[CITE\]](#)

<https://stackoverflow.com/questions/1423446/thread-control-invoke>

I can tell what I posted went completely over your head because this thread just regurgitates the overall concept of updating a control from another thread (which we already know how to do at this point) and shows that you don't understand what I was pointing out because this isn't even a proper response to my last post.

I'm talking about thread context switching being expensive and the reason why you wouldn't want to throw away the normal call which doesn't call Invoke from a control entirely under all circumstances just for the purpose of being thread-safe and having a one-liner, and you're just posting about how to marshal back to a UI thread using Invoke() by providing a link and none of your own input lol.

gigajew - September 10, 2018, 8:52 pm

If you really are afraid of the minimal costs of using Invoke every time you can do this and still use a one-liner (note this is an extension method)

BahNahNah - September 10, 2018, 8:57 pm

Here is an example why you might want to check if invoking is required.

shit code ik

If the socket fails, you will be calling form the same thread, so why bother invoking?

(Sep 10, 2018 08:52 PM)gigajew Wrote: [\[CITE\]](#) If you really are afraid of the minimal costs of using Invoke every time you can do this and still use a one-liner (note this is an extension method)

Why does it matter if its on one line?

0xDEAD10CC - September 10, 2018, 9:02 pm

(Sep 10, 2018 08:52 PM)gigajew Wrote: [\[CITE\]](#) If you really are afraid of the minimal costs of using Invoke every time you can do this and still use a one-liner (note this is an extension method)

Why did you choose Form when the base (Control) is what you should be concerned with for the Invoke method ?

gigajew - September 10, 2018, 9:30 pm

It's just an example, you could make it an extension of control as well

`RaZoR - September 11, 2018, 2:40 am

I'm using this, I don't have to invoke again and again coz I'm adding it to upper of the function if invoke required call to function again with invoke :D less memory cost

BahNahNah - September 11, 2018, 4:00 am

(Sep 11, 2018 02:40 AM)`RaZoR Wrote: [\[CITE\]](#)

I'm using this, I don't have to invoke again and again coz I'm adding it to upper of the function if invoke required call to function again with invoke :D less memory cost
How is that less memory cost? You are calling 4 methods to invoke.

CheckStatusForUI -> Invoke -> Lambda -> CheckStatusForUI.

Then you have that call stack for the rest of the method. You could remove the lambda but its still not really any more efficient.

Anyway the memory usage really isnt the concern with invoking, its the thread swapping.

(nitpicking idk im bored)

`RaZoR - September 11, 2018, 4:04 am

(Sep 11, 2018 04:00 AM)BahNahNah Wrote: [\[CITE\]](#) How is that less memory cost? You are calling 4 methods to invoke.

CheckStatusForUI -> Invoke -> Lambda -> CheckStatusForUI.

Then you have that call stack for the rest of the method. You could remove the lambda but its still not really any more efficient.

Anyway the memory usage really isnt the concern with invoking, its the thread swapping.

(nitpicking idk im bored)

as the logic, you are correct but when I use this it makes less ram usage, try it maybe you will realize.

BahNahNah - September 11, 2018, 4:45 am

(Sep 11, 2018 04:04 AM)`RaZoR Wrote: [\[CITE\]](#) as the logic, you are correct but when I use this it makes less ram usage, try it maybe you will realize.

Wew i did some testing and yeah you are correct. Running it 100 times the recursive method saves 10 bytes of memory. Performance wise they are basically identical and changes between runs.

I personally think the InvokeForm method is cleaner and easier to read though,

`RaZoR - September 12, 2018, 4:08 am

(Sep 11, 2018 04:45 AM)BahNahNah Wrote: [\[CITE\]](#) Wew i did some testing and yeah you are correct. Running it 100 times the recursive method saves 10 bytes of memory. Performance wise they are basically identical and changes between runs.

I personally think the InvokeForm method is cleaner and easier to read though, well, depending on the task that you are doing.

I could save a lot of server memory with my Unity game server.

ppaida - September 13, 2018, 3:36 pm

Use tasks and pass synchronization tokens. Use proper locking constructs.