

Re: Safely Raising Events in Multithreaded app.

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.csharp/2007-03/msg00360.html>

- *From:* "Calum" <calum.ritchie.groups@xxxxxxxxxx>
 - *Date:* 2 Mar 2007 09:21:05 -0800
-

Very true, lock blocks access to a block of code in an object from multiple threads. Putting a lock around the copy of the thread only stops more than one thread copying the event at the same time if they are invoking the method in the same object (i.e. all have a reference to the object that contains the method).

On 2 Mar, 17:13, "Laura T." <L...@xxxxxxxxxxxxx> wrote:

lock(this) wouldn't help anything if another thread wanted to add subscription to the event.

It does not protect that the event would get changed (lock(this) is not freeze every write in this class, and generally should not be used).

The meaning of taking a temp reference is that if the last subscriber of the event unsubscribes, the TestEvent would be null and the TestEvent() call would fail for null reference. Instead, you take the reference to a temp var. If the var was not null, it won't be null thereafter, even if the last subscriber unsubscribes. The temp var is still not null.

Hence the temp() call will not crash even if there are no subscribers left. It will at most be a nop.

"smkramer072fdasfasdf" <smkramer072fdasfa...@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx> ha scritto nel
messaggionews:7338697F-71F9-4055-A225-9E854FAD9D66@xxxxxxxxxxxxxxxxxxxxx

This questions is in reference to the article on Using Events in the C# Programmers Reference
(<http://msdn2.microsoft.com/en-us/library/ms173168.aspx>).

Under the Rasing Events section it says in order to avoid a race condition copy the event to a temporary variable before invoking it. That makes sense.

My question is is this: if we really want this to be thread safe why doesn't

this copy have to be within a locked block. For example, wouldn't this be

Re: Safely Raising Events in Multithreaded app.

a
better solution:

```
lock (this)
{
// Safely invoke an event:
TestEventDelegate temp = TestEvent;
}
```

```
if (temp != null)
{
temp(this, new System.EventArgs());
}
```

Or is there something I'm missing here?

Thanks,