

Re: Delegates and Events confusion

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.csharp/2004-06/3552.html>

From: Rhy Mednick (*rhy_at_rhy.com*)

Date: 06/15/04

Date: Tue, 15 Jun 2004 16:30:10 -0700

Thanks. I figured that I needed to unsubscribe in some way but since I wasn't storing an internal reference to the subscriber. I added the code like the following and it solved the problem but I don't really understand why:

```
item.MyEvent -= new EventHandler(item_MyEvent);
```

This just seems really odd to me. If the += assignment adds a reference to a new event handler it seems that handler would be a different object than the one created with the new operator in the -= assignment. I could see something like this working:

```
EventHandler eh = new EventHandler (item_MyEvent);
item.MyEvent += eh;
```

...then later...

```
item.MyEvent -= eh;
```

I can see that working because it's the same object, but when I use the new keyword twice don't I have two unique references?

– Rhy

"J.Marsch" <jeremy@ctcdeveloper.com> wrote in message
news:%23gmDw9yUEHA.760@TK2MSFTNGP12.phx.gbl...

> *Are you unsubscribing the event when you remove the instances of Class C
> from memory?*

>

> *If you subscribe to an event and do not unsubscribe, the subscriber will
> be*

> *held in memory until the event publisher (class A) is collected.*

>

> *Here's why:*

> *A delegate has a member called Target. Target is a reference to the
> subscriber (Class C). When you add the event, the publisher holds a
> reference to the delegate which in turn holds a reference to the
> subscriber,*

> *so C is held in memory. The graph looks like this:*

>

> *A ----> EventDelegate ----->C*

>
> *So, the reason that you are getting multiple event fires is that you have
> more instances of C in memory than you think that you do.*
>
> *What you probably want to do is to keep track of all of the event
> subscriptions that C makes and have C implement IDispose. When you are
> ready to kill C, call its Dispose method. Inside the Dispose, unsubscribe
> from all of the events that you are subscribed to.*
>
> *"Rhy Mednick" <rhy@rhy.com> wrote in message
> news:uVScA1yUEHA.584@TK2MSFTNGP09.phx.gbl...*
>> *I have a class (let's call it ClassA) that I've written which has events.
>> In another class (let's call it ClassB) I create a collection of ClassA
>> objects. In a third class (ClassC) I create a reference to some of the
>> ClassA objects created in ClassB. In ClassC I hook into the ClassA
>> events
>> with a foreach loop so that I hook each object. The code is something
> like
>> this:
>> class ClassC {
>> void SomeMethod()
>> {
>> foreach (ClassA item in ClassACollection)
>> {
>> item.MyEvent += new EventHandler(item_MyEvent);
>> }
>> }
>> }
>>
>> *Objects of type ClassC keep getting created and deleted, but the objects
> of
>> ClassA that it references stay in memory. Therefore, every time I load a
>> new instance of ClassC the event gets hooked again. What I'm seeing is
> that
>> I fire the event once but I end up getting it raised in the code that
>> responds to it (item_MyEvent method) for every time it was added when I
> only
>> want it to get caught once.*
>>
>> *How can I be sure that I only add the handler to the event once for each
>> item? Since the code is in ClassC it's not allowed to inspect the
>> ClassA.MyEvent to see if something's already hooked to it.*
>>
>> *I hope this makes sense. Creating a repro case wouldn't make sense here
>> because it's my application logic that's causing the problem.*
>>
>> *Thanks.*
>>
>>
>
>*