# How can I clear event subscriptions in C#?

Asked 16 years, 2 months ago  Modified 3 years, 2 months ago  Viewed 135k times

152

Take the following C# class:

```
c1 {
 event EventHandler someEvent;
}
```

If there are a lot of subscriptions to `c1`'s `someEvent` event and I want to clear them all, what is the best way to achieve this? **Also consider that subscriptions to this event could be/are lambdas/anonymous delegates.**

Currently my solution is to add a `ResetSubscriptions()` method to `c1` that sets `someEvent` to null. I don't know if this has any unseen consequences.

c# .net events delegates

Share
Improve this question
Follow

edited Nov 1, 2011 at 14:43

Jason Plank
2,336 ● 5 ● 32 ● 40

asked Sep 30, 2008 at 15:32
programmer
4,392 ● 4 ● 26 ● 21

I described a working answer using Reflection here: stackoverflow.com/questions/91778/… – Vassili Apr 5, 2021 at 17:23

## 10 Answers

Sorted by: Highest score (default)

195

From within the class, you can set the (hidden) variable to null. A null reference is the canonical way of representing an empty invocation list, effectively.

From outside the class, you can't do this - events basically expose "subscribe" and "unsubscribe" and that's it.

It's worth being aware of what field-like events are actually doing - they're creating a variable *and* an event at the same time. Within the class, you end up referencing the variable. From outside, you reference the event.

See my article on events and delegates for more information.

Share

Improve this answer

Follow

edited Oct 9, 2012 at 22:44

David Dowdle

423 ● 6 ● 10

answered Sep 30, 2008 at 16:05

Jon Skeet

1.5m ● 889 ● 9.3k ● 9.3k

3 If you're stubborn, you can force it clear via reflection. See stackoverflow.com/questions/91778/… . – Brian Oct 29, 2010 at 21:36

1 @Brian: It depends on the implementation. If it's *just* a field-like event or an `EventHandlerList`, you may be able to. You'd have to recognise those two cases though - and there could be any number of other implementations. – Jon Skeet Oct 30, 2010 at 6:49

@Joshua: No, it will set the variable to have a value of null. I agree that the variable won't be called `hidden`. – Jon Skeet May 11, 2016 at 17:37

@JonSkeet That's what I (thought) I said. The way it was written confused me for 5 minutes. – user1881400 May 11, 2016 at 18:01

@JoshuaLamusga: Well you said it would clear an invocation list, which sounds like modifying an existing object. – Jon Skeet May 11, 2016 at 18:01

40

Add a method to c1 that will set 'someEvent' to null.

```
public class c1
{
    event EventHandler someEvent;
    public ResetSubscriptions() => someEvent = null;
}
```

Share

Improve this answer

Follow

edited Jun 29, 2020 at 5:30



AustinWBryan

3,328 ● 3 ● 27 ● 44

answered Sep 30, 2008 at 15:33

programmer

4,392 ● 4 ● 26 ● 21

1 That is the behavior I am seeing. As I said in my question, I don't know if I'm overlooking something. – programmer Sep 30, 2008 at 15:42

11

```
class c1
{
    event EventHandler someEvent;
    ResetSubscriptions() => someEvent = delegate { };
}
```

It is better to use `delegate { }` than `null` to avoid the null ref exception.

Share

Improve this answer

edited Jun 29, 2020 at 5:51



AustinWBryan

3,328 ● 3 ● 27 ● 44

answered Apr 30, 2014 at 16:33

Feng

119 ● 1 ● 2

Follow

2 Why? Could you please expand on this answer? – S. Buda Oct 31, 2018 at 22:06

1 @S.Buda Because if it's null then you'll get a null ref. It's like using a `List.Clear()` vs `myList = null`. – AustinWBryan Jun 29, 2020 at 5:51

2 Given that you can do `someEvent?.Invoke(...)` in modern C# I'm not sure if this is important, or if the slight lack of clarity of using a noop delegate is worth it. – StayOnTarget Apr 22, 2022 at 14:16

8

The best practice to clear all subscribers is to set the someEvent to null by adding another public method if you want to expose this functionality to outside. This has no unseen consequences. The precondition is to remember to declare SomeEvent with the keyword 'event'.

Please see the book - C# 4.0 in the nutshell, page 125.

Some one here proposed to use `Delegate.RemoveAll` method. If you use it, the sample code could follow the below form. But it is really stupid. Why not just `SomeEvent=null` inside the `ClearSubscribers()` function?

```
public void ClearSubscribers ()
{
   SomeEvent = (EventHandler) Delegate.RemoveAll(SomeEvent, SomeEvent);
   // Then you will find SomeEvent is set to null.
}
```

Share

Improve this answer

Follow

edited Jun 29, 2020 at 5:31



AustinWBryan

3,328 ● 3 ● 27 ● 44

answered Oct 16, 2012 at 12:00



Cary

404 ● 1 ● 7 ● 15

Delegate.RemoveAll valid for MulticastDelegate: `public delegate string TableNameMapperDelegate(Type type);public static TableNameMapperDelegate TableNameMapper;` ? – Kiquenet Aug 5, 2020 at 9:29

6

Setting the event to null inside the class works. When you dispose a class you should always set the event to null, the GC has problems with events and may not clean up the disposed class if it has dangling events.

Share Improve this answer Follow

answered Oct 1, 2008 at 11:17

Jonathan C Dickinson

7,269 ● 4 ● 38 ● 46

5

You can achieve this by using the Delegate.Remove or Delegate.RemoveAll methods.

Share Improve this answer Follow

answered Sep 30, 2008 at 15:36

Micah

116k ● 87 ● 237 ● 331

7 I don't believe this will work with lambda expressions or anonymous delegates.
– programmer Sep 30, 2008 at 15:45

3

**Conceptual extended boring comment.**

I rather use the word "event handler" instead of "event" or "delegate". And used the word "event" for other stuff. In some programming languages (VB.NET, Object Pascal, Objective-C), "event" is called a "message" or "signal", and even have a "message" keyword, and specific sugar syntax.

```
const
  WM_Paint = 998;  // <-- "question" can be done by several talkers
  WM_Clear = 546;

type
  MyWindowClass = class(Window)
    procedure NotEventHandlerMethod_1;
    procedure NotEventHandlerMethod_17;

    procedure DoPaintEventHandler; message WM_Paint; // <-- "answer" by this
listener
    procedure DoClearEventHandler; message WM_Clear;
  end;
```

And, in order to respond to that "message", a "event handler" respond, whether is a single delegate or multiple delegates.

Summary: "Event" is the "question", "event handler (s)" are the answer (s).

Share
Improve this answer
Follow

edited Mar 24, 2011 at 19:07

answered Mar 24, 2011 at 18:39

umlcat
4,133 ● 3 ● 21 ● 29

2

# Remove all events, assume the event is an "Action" type:

```
Delegate[] dary = TermCheckScore.GetInvocationList();

if ( dary != null )
{
    foreach ( Delegate del in dary )
    {
        TermCheckScore -= ( Action ) del;
    }
}
```

Share Improve this answer Follow

answered Dec 20, 2013 at 20:00

Googol
47 ● 1

1 If you're inside of the type that declared the event you don't need to do this, you can just set it to null, if you're outside of the type then you can't get the invocation list of the delegate. Also, your code throws an exception if the event is null, when calling `GetInvocationList`.
– Servy Dec 20, 2013 at 20:03

0

This is my solution:

```
public class Foo : IDisposable
{
    private event EventHandler _statusChanged;
    public event EventHandler StatusChanged
    {
        add
        {
            _statusChanged += value;
        }
        remove
        {
            _statusChanged -= value;
        }
    }

    public void Dispose()
    {
```

```
            _statusChanged = null;
        }
    }
```

You need to call `Dispose()` or use `using(new Foo()){/*...*/}` pattern to unsubscribe all members of invocation list.

Share Improve this answer Follow

answered Mar 5, 2018 at 7:14

Jalal

6,834 ● 9 ● 65 ● 100

---

0

Instead of adding and removing callbacks manually and having a bunch of delegate types declared everywhere:

```
// The hard way
public delegate void ObjectCallback(ObjectType broadcaster);

public class Object
{
    public event ObjectCallback m_ObjectCallback;

    void SetupListener()
    {
        ObjectCallback callback = null;
        callback = (ObjectType broadcaster) =>
        {
            // one time logic here
            broadcaster.m_ObjectCallback -= callback;
        };
        m_ObjectCallback += callback;

    }

    void BroadcastEvent()
    {
        m_ObjectCallback?.Invoke(this);
    }
}
```

You could try this generic approach:

```
public class Object
{
    public Broadcast<Object> m_EventToBroadcast = new Broadcast<Object>();

    void SetupListener()
    {
        m_EventToBroadcast.SubscribeOnce((ObjectType broadcaster) => {
            // one time logic here
        });
    }

    ~Object()
```

```
    {
        m_EventToBroadcast.Dispose();
        m_EventToBroadcast = null;
    }

    void BroadcastEvent()
    {
        m_EventToBroadcast.Broadcast(this);
    }
}


public delegate void ObjectDelegate<T>(T broadcaster);
public class Broadcast<T> : IDisposable
{
    private event ObjectDelegate<T> m_Event;
    private List<ObjectDelegate<T>> m_SingleSubscribers = new
List<ObjectDelegate<T>>();

    ~Broadcast()
    {
        Dispose();
    }

    public void Dispose()
    {
        Clear();
        System.GC.SuppressFinalize(this);
    }

    public void Clear()
    {
        m_SingleSubscribers.Clear();
        m_Event = delegate { };
    }

    // add a one shot to this delegate that is removed after first broadcast
    public void SubscribeOnce(ObjectDelegate<T> del)
    {
        m_Event += del;
        m_SingleSubscribers.Add(del);
    }

    // add a recurring delegate that gets called each time
    public void Subscribe(ObjectDelegate<T> del)
    {
        m_Event += del;
    }

    public void Unsubscribe(ObjectDelegate<T> del)
    {
        m_Event -= del;
    }

    public void Broadcast(T broadcaster)
    {
        m_Event?.Invoke(broadcaster);
        for (int i = 0; i < m_SingleSubscribers.Count; ++i)
        {
            Unsubscribe(m_SingleSubscribers[i]);
        }
        m_SingleSubscribers.Clear();
```

```
    }
}
```

Share

Improve this answer

Follow

edited Jun 29, 2020 at 12:09

answered Jan 29, 2020 at 15:37

barthdamon

**49** ● 4

Can you please format your question and remove all the white space on the left? When you copy and paste from an IDE this can happen – AustinWBryan Jun 29, 2020 at 5:32

Just got rid of that white space, my bad – barthdamon Jun 29, 2020 at 12:11