protected members in a sealed class

Asked 15 years, 10 months ago Modified 12 years, 7 months ago

Viewed 7k times

1

I'm writing a WebPart, which means that I inherit from `System.Web.UI.WebControls.WebParts.WebPart` and I override the method `protected override void CreateChildControls()`.

However, I'd like to make the class `sealed` if possible, but that gives two issues: Just sealing the class gives a warning "new protected member declared in sealed class".

Changing the access modifier from `protected` to `private` or `internal` gives a compiler error telling me I can't change the modifier when inheriting.

That leaves me wondering: Is there any problem with sealing it and ignoring the warning? Or could this lead to any negative side effects further down the road? It *seems* to work just fine, but the devil is usually in the details.

Edit: I was just being stupid. The "new protected member" error was for a function that was indeed NOT overridden and just accidentially was declared as `protected`. Thanks for the Pragma-Tip though!

c# .net

Share

Improve this question

Follow

edited May 22, 2012 at 0:49

Paul Bellora
55.2k ● 18 ● 134 ● 183

asked Jan 30, 2009 at 9:58

Michael Stum
181k ● 119 ● 407 ● 540

4 Answers

Sorted by: Highest score (default)

6

A protected member can be seen by sub classes so you're slightly altering the interface of the class; If you declare it 'private' it's only ever seen locally so it doesn't affect the interface.

Declare your method private not protected and the warning should go away.

Share Improve this answer

Follow

answered Apr 16, 2011 at 15:35

Christopher Lightfoot
1,167 ● 1 ● 12 ● 27

2 I ran into this when refactoring from a regular class to a singleton. This is the answer I was looking for. Sort of a no-brainer when you really think about it. Why would a method be "protected" if it is in a class that is "sealed"? There would never be a deriving class ... so the warning tells you that you are writing code that contradicts the class permissions. It will

work, but the warnings are well placed. +1 to widgisoft!
– Zack Jannsen Jun 12, 2013 at 14:32

5

Are you sure you're overriding correctly? Personally I cannot repro this behavior. But if it worries you, you can use

```
#pragma warning disable 0628
// Offending code
#pragma warning restore 0628
```

Share Improve this answer Follow

answered Jan 30, 2009 at 10:05

Anton Gogolev
116k ● 39 ● 203 ● 291

4

The fact that it says there's a *new* protected member declared in the class is slightly worrying.

Hmm... I can't reproduce this in simple test code:

```
using System;

public class Base
{
    protected virtual void Foo()
    {
    }
}

public sealed class Derived : Base
{
    protected override void Foo()
    {
    }
}
```

compiles without warnings with .NET 3.5SP1. Are you *definitely* overriding the base method? Check that you really have got the `override` modifier. (Sorry if that sounds patronising - I'm not trying to accuse you of being lax or anything. I'm just stumped otherwise...)

Share Improve this answer Follow

answered Jan 30, 2009 at 10:02

Jon Skeet
1.5m ● 889 ● 9.3k ● 9.3k

0

Sounds to me like it is being daft. I'd ignore the warning, after all it's only stating that what you are doing is illogical like having a public ctor on an abstract type. The worst case scenario is a bit of confusion.

I think i've had this as well but only in Compact Framework code, is it Full Framework in this case?

Share Improve this answer Follow

answered Jan 30, 2009 at 10:16

Quibblesome
25.4k ● 10 ● 62 ● 104