

Re: List<> of struct with property. Cannot change value of property. why?

Re: List<> of struct with property. Cannot change value of property. why?

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.csharp/2007-05/msg02315.html>

- *From:* "Peter Duniho" <NpOeStPeAdM@xxxxxxxxxxxxxxxxxxxx>
 - *Date:* Tue, 15 May 2007 14:45:55 -0700
-

On Tue, 15 May 2007 14:20:51 -0700, Zytan <zytanlithium@xxxxxxxxxx> wrote:

If C# **did** have the "const" keyword, I would expect it to work more like that, than in the way that Zytan would like it to work.

I think I am saying the same thing.

Sorry...I guess I misunderstood what you wanted the language to do.

Once you have this boolean value that states the "const"-ness of a method, then in the case of calling a non-const method on a List<> element that is a temporary (the compiler already knows when it's a temporary), the compiler could complain and say: "Hey, you might be modifying a temporary! You should only call const-methods on temporaries!"

Well, IMHO the compiler could easily provide the same warning today, without the "const" keyword.

After all, in C++ there's a LOT of code that doesn't modify data that isn't marked "const". Relying on the "const" keyword to enable a warning wouldn't have been a good idea in C++, because you'd get a lot of false positives due to the large amount of code that is "const" without using "const".

Conversely, if such a warning is a good idea (in C# or C++), it seems to me that the compiler ought to just warn and forget about trying to determine whether the method is actually a "const" method or not.

Personall, because of the false positive issue, I think such a warning isn't a good idea. You'd see it far too often when it wasn't legitimate for it to actually be useful.

Now, I suppose the language could add a feature in which it requires the compiler to analyze every method and apply its own "const" attribute based on that analysis. But that opens a whole new can of worms, including what is essentially the same problem that the C++ "const" keyword had: until every single function in the call chain supports that attribute, you wind up with a lot of functions that can't be marked as "const"

Re: List<> of struct with property. Cannot change value of property. why?

Re: List<> of struct with property. Cannot change value of property. why?

even though they really are (or conversely, you have to do casting a bunch of stuff to glue the const/not-const stuff together).

My feeling is that while it's possible theoretically to address the issue, I think it's simpler to just make the language simple and consistent, and require developers to understand that the List<> "[" operator is providing a copy of the element in the list. There's too many situations where working directly with the copied value is useful for the compiler to go around warning you every time you do it, and introducing a "const" attribute (explicit or implicit) creates far too many new hassles to make it worth the trouble.

Opinions will vary, of course. :)

Pete

.