

Re: I freaking HATE var!!

Re: I freaking HATE var!!

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.csharp/2007-04/msg02093.html>

- *From:* "Nicholas Paldino [.NET/C# MVP]" <mvp@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx>
 - *Date:* Fri, 13 Apr 2007 19:50:52 -0400
-

Not to marginalize your post, but your last statement pretty much axes all the effort you put into it:

If only it wasn't too late. The tsunami is coming. But our engineering team will NOT take the plunge into C# 3.0.

If you don't like it, don't use it, end of story. Outside of how you use it, and the engineering team that you work with uses it, what do you care if someone else uses it? It's not going to break any of the code you have now, so that's a non-issue. Your coding standards are going to dictate what you can use and how you name your variables, and your ecosystem will be preserved.

The only thing I think that Bill is wrong on is that in the example that selects strings from an array of strings, it should be assigned to `IEnumerable<string>`. I agree, `var` shouldn't be used in cases where you know the type, and in this case, you do know the type that is being returned. However, in order to get projections to work, there is no way around `var`.

Granted, having VS.NET wire up the code for you in known types will be nice (and it should do something like this, converting projections/anonymous types to known-types), but the amount of code you would have to write by hand if you didn't have VS is enormous, and they can't depend on the IDE to pick up where language features fail.

I agree that those that use `var` when they know the type of what is being used (and yes, even in the case of those long generic types, use the `using` directive to alias it instead) should be skewered, but really, it's not that bad.

It's also Friday, I'd save this kind of rhetoric-filled post for Monday. That's what Monday's are for.

--
- Nicholas Paldino [.NET/C# MVP]
- mvp@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Re: I freaking HATE var!!

"Jon Davis" <jon@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx> wrote in message
news:%23vAeTfifHHA.3960@xxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bill Wagner posted something here ..

<http://msdn2.microsoft.com/en-us/vcsharp/default.aspx>

"Local Type Inference, Anonymous Types, and var"

"Of all the features in C# 3.0, local type inference is generating the most questions and misunderstanding. You know, 'var'. The fact is local type inference is not as scary as it seems. In fact, you're not losing strong typing. It's a simple time saver that is actually necessary to support anonymous types."

Great. I appreciate that it's "necessary" to support LINQ (and LINQ is the only major reason why it's necessary to support anonymous types).

Well then I guess my issue is with anonymous types. But I dislike var. It is stinky to my nostrils. Bill can argue till the cows come home that it's still strongly typed and produces MSIL that is no different than C# 2.0 code. I don't care, it's not the MSIL that I have to stare at all day long.

It was only a couple years ago that I was running around singing the praises of Microsoft and celebrating the wonderful, magical LINQ invention (back then it was a Microsoft Research invention, C-Omega). Finally, SQL and C# would be beautifully merged syntactically. So don't get me wrong, I understand and appreciate LINQ's value.

But I also understand and appreciate the hell I went through, and why I am so glad I went through it, to wean myself off of Visual Basic 6.0 (Variant-land) and javascript and vbscript, and into Java and C#. In gross celebration of lessons learned, I hacked together a silly Variant object for C#, just for fun.

<http://www.planetsourcecode.com/vb/scripts/ShowCode.asp?txtCodeId=2854>

It didn't take long that I began really appreciating strongly typed declarations. It had nothing to do with performance. It had little to do with stability. It had mostly to do with the fact that as I'm scrolling up and down in my code, or when I right-click a variable and choose "Go To Definition", I immediately saw what the heck it was. Not what it was named. Not what was assigned to it. But what it was.

So var produces strongly-typed MSIL. Great. But it's still anonymous.

Introducing var forces us to revert back to the old Hungarian notation style naming conventions—that is, variables will have to have their types inferred in the name. Yes, the type is strongly typed at compile-time. And if Microsoft builds Visual Studio right, yes, IntelliSense tooltips will reveal what type a variable is. But what if we don't want to hover over the variable with the mouse? What if we're not using Visual Studio? When

Re: I freaking HATE var!!

Re: I freaking HATE var!!

editing code, if you have to do extra work of any sort to discover the type of a variable, productivity is lost.

The biggest grief of var is that it's really only useful for LINQ+SQL, but it will compel Variant-style VB coders and javascript script kiddies to dilute the somewhat more disciplined structure of C# for no good reason. Normal declarations that do not involve LINQ and could do just as well with a standard type reference will use var instead....

```
var doc1 = myComplexObject.GetDoc();  
// oh, by the way, GetDoc() returns a System.Xml.XmlDocument
```

```
var doc2 = myOtherObject.GetDoc();  
// oh, and um, GetDoc() returns a System.Windows.Forms.HtmlDocument
```

What's wrong with this picture? I'll tell you what's wrong: "var" sucks!! There is nothing here that makes what doc1 and what doc2 are. The method names of each object aren't necessarily wrong. But now they'd have to be renamed to expose the type of data being returned. Where is the advantage?

Bill makes the lousy argument:

```
<quote>  
string[] words = { "cherry", "apple", "blueberry" };
```

```
var sortedWords =  
from w in words  
orderby w  
select w;
```

It's fairly obvious that sortedWords is some sequence of strings. (It's actually a System.Query.OrderedSequence<string, string>). In my opinion, OrderedSequence<string,string> doesn't add any new information for me when I'm trying to read this code. In fact, I think it's clearer with the var keyword.

```
</quote>
```

No, Bill, when you hide the declaration of the "words" variable, it is not obvious that sortedWords is some sequence of strings. What if a Words was an array of a complex type called Word? And why are you limiting yourself to words in these samples? How often do we have the luxury of working with objects that are "obviously" composed of strings or arrays or collections thereof?

Most often, types consist of all kinds of weird values. Even when pounding against a SQL database table, a particular table could have an ID column that could be an integer, it could be a string, it could be a GUID. Obvious? No. Strongly typed? Sure, buried down the MSIL. But the only way we would be able to find out what we're dealing with is if you actually dig into the MSIL or open up the database designer or, more likely, SQL Server Management Studio, and examine the design of the database table. Suddenly LINQ ain't so time-saving.

Re: I freaking HATE var!!

What LINQ should have done is have strongly typed SQL statements rather than infer the crap out of everything based on a remote database schema. Visual Studio should have auto-generated some strongly-typed code based on a select at design-time. You could have been just as lazy, but far more verbose. Terseness getting severely overrated right now; I'm obviously speaking in favor of verbosity.

Try "var" in another language. "Boo" for duck typing--I think Boo is a great language. But don't corrupt C# with this insolent laziness.

"var" is an evil beast that will take an otherwise wonderful language and make horrible programmers out of people. It will cause C# to be exactly what VB.NET already is -- a strongly-typed language that is very powerful but that is used by undisciplined people who are too lazy and undisciplined, or most often just plain too ignorant, to know how to write well-defined, clearly readable, well-designed, easily maintainable code.

If only it wasn't too late. The tsunami is coming. But our engineering team will NOT take the plunge into C# 3.0.

Jon