

Re: Microsoft Losing Interest in C#?

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.csharp/2004-09/5974.html>

From: Daniel O'Connell [C# MVP] (*onyxkirx_at_--NOSPAM--comcast.net*)

Date: 09/24/04

Date: Thu, 23 Sep 2004 20:00:21 -0500

>>>

>>> *If the translator knows how to map a Java SDK call to a .NET framework call, part of that mapping should be knowing which direction(s) to copy the array in. In this case, it needed to be copied out but not in, and JLCA generated the precise opposite. Sorry, but it's a low-quality product.*

>>

>> *I'm not arguing its quality, I don't really care, I just think its overly arrogant to assume that any application can know every API's purpose and make decisions based on that, especially those which could be in, out, or in/out depending on other parameter values. This case *could* have, I suppose, but I doubt the entire SDK is that clean cut. Your general case scenario is just bad and wrong-minded, IMHO.*

>

> *Gee, and we were getting along so well. :-)*

Sorry, I came across a little gruff, ;). Honestly I wasn't *that* impressed by it, myself, though I do think v3 is considerably better than v1.

>

> *Actually, it wouldn't be that hard. There's a finite list of API calls it tries to convert, and for each of them, determining whether the inputs are modified or not is just a question of reading the documentation. JLCA already has API mappings, or it couldn't have gotten that far. Annotating them with some simple semantic information doesn't add much more work. And it's a perfect opportunity to use unit tests to verify the behavior: write tests in Java, translate them, and ensure that both versions act identically. (More difficult with a random number generator than with most calls, I admit.)*

It might not be terribly hard, I'm just more concerned with the possibility that both choices will be wrong in some cases, can that semantic information express all possibilities or only a most common scenario? Can the analyzer determine which possibility is in use? If not, chances are you are going to get hidden, subtle bugs that upset someone. It just may not happen to be either of us, ;).

- > *Actually, copying byte arrays is a bad idea. In the array, the byte is*
- > *simply an octet; it makes no difference whether it's signed or unsigned.*
- > *It isn't until the byte comes out of the array that it matters, and then*
- > *it can be cast to byte before any arithmetic gets done on it.*
- >

I tend to agree, I think array covariance of primitive types (or at least signed/unsigned alternatives of one type) is something worth having in the language. But that goes well beyond the JICA, it is stuck using what it can. I suppose it could convert to a byte array, but that wouldn't be very literal and may upset people as well, ;).

- >
- >>
- >> *Also, I think translating Java into C# is a pretty bad idea. The two are*
- >> *terribly different, and I wouldn't think that java code would be*
- >> *something I would want in C# (or vice versa, for that matter).*
- >
- > *C# is bigger, since it has events, delegates, unmanaged pointers, etc.*
- > *But for the common subset, 99% of the difference between the two languages*
- > *is syntax. The object models are close to identical, and the common APIs*
- > *are awfully similar too; notice how `Random.nextBytes(0)` translated to*
- > *`Random.NextBytes()`. It's actually not all that difficult to do correctly.*

While most of the api's are similar, personally I think it is the eventing model that changes things. I write very few substantial pieces of code in C# that doesn't use an event here or there, and I doubt there is a lot of extensive java code that doesn't use and/or expose event sinks.

However, the other issue is weird bugs like this one. While this particular bug is a semantic one due to the JICA, I don't think the application is capable of definitively assuring no changes in behavior. Like I said, JJ2k was something like 350 issues. While I certainly couldn't write a jpeg2000 implementation in the period of time it would take the maintainer to fix those issues, the time for me personally to learn enough about jpeg2000 and java to fix those errors without causing issues and keep the java baggage would probably be high enough to consider a native implementation using .NET platform conventions. Thus, for the uninitiated, it's a pretty useless tool

For me, personally, architectural conventions can go a long way towards productivity. While I don't have many particular problems with java's conventions, seeing them all of a sudden within .NET code is a shocker and breaks the mindset somewhat.