

Re: Microsoft Losing Interest in C#?

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.csharp/2004-09/5357.html>

From: Mike Schilling (*msscottschilling_at_hotmail.com*)

Date: 09/21/04

Date: Tue, 21 Sep 2004 14:02:23 -0700

"Daniel O'Connell [C# MVP]" <onyxkirx@---NOSPAM---comcast.net> wrote in message news:%23lZKDMBoEHA.3252@TK2MSFTNGP11.phx.gbl...

>> *Does that mean 350 messages about code it can't convert? That's not bad.*
>> *What I recall is generated code that either didn't compile or (much*
>> *worse) compiled but was incorrect. .*
>
> *Yes. I certainly didn't have time to try to fix the project at the time,*
> *still don't really. It was jj2k, a open java implementation of jpeg2000.*
>
> *There were quite a few issues with Writer&TextWriter compatibility, etc.*
>
> *The code doesn't compile right now because, IIRC, the JCLA generated a few*
> *javaisms in the sections where a direct conversion wasn't possible.*

Here, for whatever historical or amusement value it might have, is an evaluation of JLCA I did in May of 2002:

The first file I tried to create was UUID.java. It should be a good candidate for conversion in that:

- It's simple java, and doesn't use any advanced features.
- It has no dependencies on java classes outside the JDK.
- It can be tested standalone. When run, it produces UUIDs.

The conversion produced no errors. It took 7 seconds to convert 500 lines. Attempting to compile the resulting csharp file results in:

```
UUID.cs(99,21): error CS0246: The type or namespace name 'Comparable' could not be found (are you missing a using directive or an assembly reference?)
```

The JLCA doesn't understand the Comparable interface, which was added after 1.1.4 (in 1.2). An odd thing to leave out of a tool which claims to support Collections. In fact, .NET does contain an IComparable interface almost

identical to the Java Comparable interface, but JLCA is apparently unable to make this conversion.

Take 2, remove Comparable:

UUID.cs(99,14): error CS0535: 'UUID' does not implement interface member 'System.Runtime.Serialization.ISerializable.GetObjectData(System.Runtime.Serialization.SerializationInfo, System.Runtime.Serialization.StreamingContext)'
UUID is Serializable, and the .NET implementation of serialization is unsurprisingly quite different from the Java one.

Take 3, remove Serializable:

UUID.cs(144,35): error CS0246: The type or namespace name 'StringTokenizer' could not be found (are you missing a using directive or an assembly reference?)
StringTokenizer was present in 1.0, and the JLCA create some code to emulate it, but apparently doesn't understand the 3-argument constructor form (which may have been added after 1.1.4). This code is used to parse UUIDs, which isn't part of the test, so we'll remove it.

UUID.cs(195,30): error CS0246: The type or namespace name 'MessageFormat' could not be found (are you missing a using directive or an assembly reference?)

UUID.cs(200,10): error CS0246: The type or namespace name 'MessageFormat' could not be found (are you missing a using directive or an assembly reference?)

MessageFormat isn't documented as post-1.1, so I don't know why it isn't supported. We'll re-implement the code that uses it with `StringBuffer.append()`;

UUID.cs(387,4): error CS1547: Keyword 'void' cannot be used in this context
UUID.cs(404,7): error CS1547: Keyword 'void' cannot be used in this context
UUID.cs(412,7): error CS1547: Keyword 'void' cannot be used in this context
UUID.cs(413,7): error CS1547: Keyword 'void' cannot be used in this context

This is cute. There are void functions called `getXXX`, and the JLCA assumes they're accessors for bean properties of type void. We'll rename them.

UUID.cs(479,55): error CS0030: Cannot convert type 'sbyte[]' to 'byte[]'
UUID.cs(483,55): error CS0030: Cannot convert type 'sbyte[]' to 'byte[]'

This is simply bad code generation. C#, like C++, has both signed and unsigned types. "byte" in Java is converted to signed byte (which is correct.) The java code is using `java.util.Random` to fill a byte array with random numbers. JLCA produces a routine to convert the signed byte array to unsigned bytes, which its `Random` number class expects, but there are two

problems with the produced code:

1. JLCA generates spurious, incorrect casts when calling it. These are the errors shown above.

2. The resulting code:

Copies the signed byte array to an unsigned byte array

Fills the unsigned byte array with random numbers

Proceeds to use the signed byte array, which still has zeroes in it

We'll rewrite the C# code to be correct.

UUID.cs(487,17): error CS0221: Constant value '170' cannot be converted to a 'sbyte' (use 'unchecked' syntax to override)

This comes from the line:

```
nodeId[4] = (byte)0xaa;
```

In C#, the cast apparently isn't enough to allow setting a signed byte to a value which doesn't fit. We'll change it to

```
nodeId[4] = -86;
```

take 4:

Conversion works. It takes less than a second to compile and link the C# code. It produces UUIDs which look correct.