

Re: Why does x.ToString() throw an error if x == null?

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.csharp/2008-08/msg02705.html>

- *From:* "Ben Voigt [C++ MVP]" <rbv@xxxxxxxxxxxxxxx>
 - *Date:* Thu, 28 Aug 2008 14:27:40 -0500
-

Michael C wrote:

"jp2msft" <jp2msft@xxxxxxxxxxxxxxxxxxxxxxxxxxxxx> wrote in message news:8D2A44C2-94FA-428F-A3A2-1F077B844B0D@xxxxxxxxxxxxxxxxxxxxx

Even if you declare an object, it doesn't really exist yet until you have initialized it and the compiler has not set aside any space for it.

The bulk of the object does exist even before you declare the object. Generally the actual code will be greater than the object itself (in size) and all of the code exists and is ready to call before an instance is created.

To say, "Look at the space for this object that I have not created yet and tell me what it says" would naturally throw an error.

There is no reason calling a function on a null object *has* to raise an exception, the designers of C# just designed it that way. They actually needed to put an extra check in to stop it working. Basically when you call, say, object.ToString you end up calling a global function ToString where the pointer to the object is passed in, ie

```
x.ToString();
```

translates to (under the hood of course)

```
public static string ToString(object* pObj)  
{  
    //convert pObj to string  
}
```

I'm sure it's more complicated than that but that's the basic idea.

Re: Why does x.ToString() throw an error if x == null?

No, it's not really even all that close. And the devil is in the details.

The actual call x.ToString(), being virtual, ends up looking like it was generated by something like this:

```
(*g_types[x->typetag].Methods['ToString'])(x);
```

unless the JIT hasn't compiled the MSIL into machine language yet, in which case it does that before calling the function.

The actual function called depends on the actual run-time type of the instance passed in. The run-time type is encoded in the first four bytes (on x86) of each ref-typed object instance. The first four bytes of the object instance referred to by null.... do you see the problem now? No instance implies no type tag which implies no way to select the "right" ToString implementation.

The compiler does the null check before the static function is called but there is no reason that it needs to. It can easily call the static ToString function and let that function decide whether to allow nulls or not.

Michael