

Re: [OT] My First C# (warning – long post)

Re: [OT] My First C# (warning – long post)

Source: <http://coding.derkeiler.com/Archive/Cobol/comp.lang.cobol/2007-02/msg00016.html>

- *From:* "Pete Dashwood" <dashwood@xxxxxxxxxxxxxxxxxxxxxxxxxxxxx>
 - *Date:* Fri, 2 Feb 2007 12:32:47 +1300
-

"LX-i" <lx0007@xxxxxxxxxxxxx> wrote in message
[news:19a17\\$45c1fe8b\\$454920f8\\$23261@xxxxxxxxxxxxxxxxx](news:19a17$45c1fe8b$454920f8$23261@xxxxxxxxxxxxxxxxx)

Pete Dashwood wrote:

"LX-i" <lx0007@xxxxxxxxxxxxx> wrote in message
[news:d8eb1\\$45c17062\\$454920f8\\$31633@xxxxxxxxxxxxxxxxx](news:d8eb1$45c17062$454920f8$31633@xxxxxxxxxxxxxxxxx)

Pete Dashwood wrote:

Hi Daniel,

Unfortunately, much as I'd like to, I'm just too busy at the moment to analyse this properly.

Fortunately, Andrew is doing so (I had a quick look at his comments and endorse them 100%) and one set of criticisms is probably all you need on a first attempt :-)

I'm a big guy (well, metaphorically). I can take it.

I formed some first impressions from a quick look (before I saw Andrew's comments.... :-)) and they were as follows:

1. It is a procedural translation into an OO environment. Even down to the "Working-storage" at the top of the Class... :-)

Aw man, cut me some slack – show me a COBOL program with less than 15 working-storage variables! :)

Re: [OT] My First C# (warning – long post)

Re: [OT] My First C# (warning – long post)

I did cut you some slack... Have a look at properties if you really want global variables

I thought that's what they were... Isn't that how you define properties?
"public [type] [name]"?

No. Those are global variables. Click on Help in VS2005 and then click "index". Enter "properties"... Or start with this direct local link:

ms-help://MS.VSExpressCC.v80/MS.NETFramework.v20.en/dv_csref/html/f7f67b05-0983-4cdb-96af-1855d24c96

(And, I know you were cutting me some slack – I was just being silly...)

(The reason I did this that way,

especially for the block of private ones, was just so that the memory didn't have to be loaded every time. Can you imagine loading that array of COBOL keywords every time `isCobolKeyword` is called?)

No, and I wouldn't do it... :-)

Along those lines – is there a better way to search through an array other than "for (int i = 0; i < array.length; i++)"?

Yes, "foreach". I think Frank has covered it.

You'd think that you could cast a substring of one character TO a character, wouldn't you? Yes, the `Trim()` is probably extra now, but that was my attempt to get it to quit griping at me that I had given it a "String", when it wanted a "char". (I know in Java you can't do a switch on a string – I'm guessing that it's that way in C# as well. If not, then I spent 20 minutes wasting my time... :-> Which wouldn't

Re: [OT] My First C# (warning – long post)

Re: [OT] My First C# (warning – long post)

surprise me to find out...)

Don't make Java assumptions about C#. You can certainly switch on a char or string, (as long as they are constant). When it "gripes" at you, there is a reason. I consider these gripes as a voice saying: "Pete, you are about to learn something..." :-). Fortunately, as the learning process proceeds, the voice becomes less insistent. But you have to listen to it, do the homework, and not just look for a way to dismiss it as an irritation... :-)

I did look at it as a learning experience. I was just frustrated at the time because I had a 700-line+ file of C# that wouldn't execute because of this one little problem.

Yeah, I know... been there :-). Nevertheless, patience is an essential part of the learning process...

Here's a sample from live code that covers Casting string to Char, setting up a Global property, and managing a COBOL data block in C#...

```
}  
  
private string _IBreturn;  
  
// A read-write instance property:  
  
public string IBreturn  
{  
  
    get  
  
    {  
  
        return _IBreturn;  
  
    }  
  
    set  
  
    {  
  
        _IBreturn = value;  
  
        _IBreturn = _IBreturn.PadRight(5, char.Parse("0"));  
  
    }  
  
}
```

Re: [OT] My First C# (warning – long post)

```
_IntBlock = _IBreturn +  
  
_IntBlock.Substring  
  
(_IBreturn.Length, _IntBlock.Length –  
_IBreturn.Length);  
  
}  
  
}
```

So you put braces after the variable declaration? That's the first time I've seen that structure. :)

(As you can see, I do not share your aversion to braces surrounding code blocks being on a new line, and I actually find this helpful...)

Yeah, and you code in SECTIONS too! ;)

Touche! (Nevertheless, my code works... :-))

Of course I'm kidding – that just goes to show that different folks have different ideas of what "readable" is. My biggest complaint in how much north-south real estate it takes up on my screen. I guess if my methods were more of a proper size, I wouldn't have to scroll to see them. Maybe that's a corollary to the 10-line rule – if you can't see it at one time in the editor, it's too big!

Absolutely. Now you're getting it. Small is beautiful.

(Is it double-spaced in the IDE, or was that just what happened when you pasted it into the newsreader?)

No, it's the bloody newsreader. I was tempted to post in HTML but I realise most people don't have their newsreaders configured for it.

The "set" method inserts a Pic x(5) return code into the start of an interface block, defined as a COBOL structure of 8192 bytes (this is for compatibility with COM BSTRING type.)

Re: [OT] My First C# (warning – long post)

Re: [OT] My First C# (warning – long post)

Why did you use `char.Parse("0")` instead of just `'0'`?

Because `'0'` wouldn't work...? :-)

The return code must be padded with zeroes if it does not occupy the full 5 bytes. Although `_IBreturn` is a string, it must be padded with `Chars`. You may consider this is "cheating" so here's a direct cast to `char`, also taken from live code :-)...

```
string temp = ", [";  
  
char[] trimChars = temp.ToCharArray();
```

No, no cheating comment from me... :) I wonder if

```
dr["element_type"].ToString().ToCharArray()[0]  
  
would do what I was trying to do there?
```

Why not try it and see? The empirical approach always trumps the theoretical one... :-)

The final action of the "set" method above is to update the COBOL interface block with the new return code. Note that the return code is a private field that cannot be accessed other than by means of the get and set methods; the public copy of the field automatically uses these methods when it is referenced. You could argue that this is unnecessary; why not just slice a substring of `_intBlock` when you want the return code? It is unwieldy to keep referencing substrings of an 8K string simply to get 5 bytes. Keeping each field in the block separate as a private property and updating the block when they get changed, by means of their set methods, seems to work pretty well. I can also "audit" the integrity of the block because I have each field of it as a separate entity. The entire block can be easily reconstructed from these fields and that might not be so easy if it is being sliced left right and centre. Finally, this approach lends itself to automation and can be easily generated from COBOL definitions.

That makes sense to me. It's a similar concept to a few tables we have in our database at work. They hold "interface parameters", and since the

Re: [OT] My First C# (warning – long post)

Re: [OT] My First C# (warning – long post)

parameters vary for each interface, we have it broken down for each format. It certainly beats reference modification... especially when the next guy comes along and doesn't know that the number of days to keep transaction history for this interface is in columns 4–6... :)

3. The embedded SQL is just a hangover from procedural code; it is inefficient and ugly. Think in tiers; separate data access from business logic. Treat db connections as precious; don't hold them any longer than you need to. I posted a link previously which outlines the C# approach and it is quite different from the procedural embedded SQL approach. (Since my own stuff is now working, I can also confirm that it is VERY fast...)

See my post to Andrew on this topic – how?

Did you even look at the article I referenced? :-) It not only explains How, it shows WHY :-)

I looked at it, and I sort of understand how they did it... I'm going to play with some things over the next few days. I'll post what I find, with hopefully enough COBOL references that the other folks won't toss me out on my head. :)

I strongly recommend some experimentation.

Setting a connection and issuing SQL is pretty unavoidable, but after that there is a big difference between using a SQL reader and using a DataAdapter/DataSet. The DataSet has methods and properties that are really useful (and cool :-)). You simply don't get them if you just write embedded SQL or open a cursor. The DataAdapter allows you to process everything and only apply the updates at the end when you are satisfied – the DataSet automatically notifies errors if you change something incorrectly.

Have a look at the Data facilities in VS 2005. If you know what you're going to be accessing, it will build the code for you in minutes. My stuff is completely dynamic and designed to work with any database of a given family, so I need to know what the structure is at runtime. DataSet methods and properties give me that. It was really problematic for me NOT

Re: [OT] My First C# (warning – long post)

Re: [OT] My First C# (warning – long post)

to be able to use the facilities in VS 2005 because of the unknown nature of what I would be accessing. Time and again I realised that the IDE would generate the code if I could simply tell it what I wanted. I spent many hours combing the support and help to understand the new (to me) concepts and finally get what I needed. I still can't believe it actually works, but it does :-)

In looking at things back and forth in writing this reply, it's starting to click.

Excellent! It will take a little time. It is a journey of many small steps... Each method you write will be "better" than the last one.

For myself, I can see the difference in what I write today and what I wrote even 5 weeks ago. In a year, I'll probably be embarrassed by what I have and want to rewrite the lot... :-) At this stage, as I mentioned earlier, elegance is only of passing interest. The prime criterion is that it works. When it works, it not only builds confidence, but it also buys time to do further study and make the next round even better.

<remainder snipped>

Pete.

.