

Re: Parsing between a character and sysmbol

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.vb/2006-08/msg03145.html>

- *From:* "Branco Medeiros" <branco.medeiros@xxxxxxxxx>
 - *Date:* 19 Aug 2006 12:21:34 -0700
-

I wrote:

Also, accessing array items inside a loop should be avoided whenever possible.

And then Scott M. wrote:

That's ridiculous! Where did you get that from? One of the benefits of arrays and collections is the ease of iterating through them via loops. Your code is perfectly fine, but please don't make up best practices just to bolster your code over others.

I'm not trying to bolster anything over anyone, just givin' away advise that I gathered along the road. You're free to accept it or not.

Considering the tone of you message, and knowing that I can really propose ridiculous things without noticing, I took the time to call ILDasm on your code. This is what I got inside its main loop (I'm not even talking about using Split here, that required the conversion of the "-" char into an array of chars. The things you learn with ILDasm...):

```
For i = 0 To y(0).Length - 1
```

```
    If IsNumeric(y(0)(i)) Then z &= y(0)(i)
```

```
; Locates the element Y(0) and pushes it on the stack
```

```
IL_002d: ldloc.2
```

```
IL_002e: ldc.i4.0
```

```
IL_002f: ldelem.ref
```

```
; Gets the i-th char from the element
```

```
; on the stack (y(0)(i))
```

Re: Parsing between a character and symbol

```
IL_0030: ldloc.0 ;
IL_0031: callvirt char String::get_Chars(int32)

; Boxes the char from the previous step and
; calls IsNumeric(Object) (oops!)
IL_0036: box System.Char ;
IL_003b: call bool IsNumeric(object)

;Skips the code if the result of the
;previous step was false
IL_0040: brfalse.s IL_0057

; Pushes z on the stack
IL_0042: ldloc.3

; locates (again!), the element y(0)(i):
; pushes y(0) onto the stack
IL_0043: ldloc.2
IL_0044: ldc.i4.0
IL_0045: ldelem.ref

;gets the i-th char
IL_0046: ldloc.0
IL_0047: callvirt char String::get_Chars(int32)

;Converts it to String (!!!)
IL_004c: call string ToString(char)

; concatenates z and the previous string (creating another string)
; and points z to it
IL_0051: call string String::Concat(string, string)
IL_0056: stloc.3
```

Next

As you can see, accessing `y(0)(i)` is somewhat inefficient, because the code will have to locate the base element by index twice, and the char by index twice, at every loop. This means locating this element ten times, for the current example. I guess you'd agree that I can locate the element only once *outside* the loop, and the given char only once per cycle (inside the loop). That's exactly what happens when you use an enumeration on a string, so I really prefer this approach than accessing an indexed element inside a loop, anytime (unless, of course, I need the index for something else).

As for the string concatenation, two new strings are created at each loop cycle, one from the char and one as the result of the concatenation. To produce your five-chars string you created at least 10 temporary strings. I don't know about you but I don't think this is

Re: Parsing between a character and sysmbol

efficient at all... Now, I can only guess how the StringBuilder works, but I'm positive it doesn't use temporary strings, but more likely an efficient array of chars that will be resized fewer times than you created strings (if resized at all).

Finally, knowing that IsNumeric() boxes its parameter will really make me stay far away from it (unless, of course, I need it's VB6 semantics, which is hardly the case, nowadays).

Hope this clarifies things a bit.

Of course, you can allways argue that the string you're manipulating is so small that the number of resources and the time taken to process it is negligible, and maybe I'll agree with you on this. But, who knows, the OP's example may or may not reflect his actual string. Besides, there's no information on how many other strings he has to handle.

Best regards,

Branco.

.