

Re: VB.NET advantages

Source: <http://www.tech-archive.net/Archive/VB/microsoft.public.vb.general.discussion/2005-01/1707.html>

From: Michael D. Ober (*mdo._at_.wakeassoc..com*)

Date: 01/14/05

Date: Fri, 14 Jan 2005 11:56:59 -0700

"Ken Halter" <Ken_Halter@Use_Springly_Hotmail.com> wrote in message news:OKtO2gl%23EHA.3924@TK2MSFTNGP15.phx.gbl...

> "Michael D. Ober" <obermd.@.alum.mit.edu.nospam> wrote in message

> news:KpIFd.6317\$pZ4.2768@newsread1.news.pas.earthlink.net...

>>

>> 2 – Object classes are a hell of a lot easier to implement in VB 2005.

>> You

>> don't need a separate .CLS file, although I would still use one for a more

>

> No need? or no support. If no support then, how to share classes between projects?

No need in this case. You can still put object class code in a separate file. In fact, VB 2005 even lets you split a class into different files and the compile will assemble it for you – I'm not sure how useful this would be, but it's there. All VB 2005 source code modules end in ".vb", so the nice differentiation that .FRM, .CLS, and .BAS created is unfortunately gone.

>

>> complex development task than this one. I created an Object class that inherited from the StreamReader and was able to use all the underlying StreamReader methods that I needed, without writing a single bit of code other than "inherit StreamReader" as the first non-comment line of the class.

>

> In VB6, it's "Inherit VB.Runtime.Fullydebugged.FileIO.Functions" ;–)

Yet I've had to go to the Windows API to do something as simple as forcing a file buffer to flush. The only way to do this without the WinAPI is to open your file with "len 1", which is very, very inefficient. Being forced to use the API is simply wrong for something this fundamental.

>

>> 3 – Properties (Property Let/Set/Get) are syntactically much easier to handle and code. In VB 6, it is possible for the Property Let and the

> > *Property Get method code to get separated from each other in the source.*
>
> *That's stuff you'd never notice if you were using 'Single Procedure View'*
> *or, as the setting says, "Full Module View" turned off... which is how I*
> *write code.. since, I'm never working on anything but a single procedure*
at
> *a time and I really like being able to use ctrl-Home/ctrl-End/ctrl-A/etc*

VB 2005 IDE is missing the single procedure view. The collapsible procedure feature provided just isn't as useful.

>
> > *Not so in VB 2005. The overall syntax for this is*
>
> *Coming from VB, that property declaration just looks "wrong" <g>.*

Definitely a syntactical difference that will take some getting used to for anyone doing the conversion.

> >
> > *Note that there is no "SET" as objects no longer have a default method*
or
>
> *I like using Set. Regardless of who writes the code, you can quickly tell*
> *that it's an object they're working on.*

In VB 2005, there is no difference between an object and a standard variable – both may be on the stack or in the heap with a pointer on the stack. The difference in VB 6 is because the Garbage Collector in VB 6 is reference counted and needs help from the programmer. We help it by using "set". It's very easy to write degenerate code that can fool the VB 6 garbage collector. The .Net GC cannot be fooled. Nor is there a difference that I can tell from my limited usage between a record structure and an class structure – both can have methods. Thus the concept of SET someobject = Value doesn't exist. If you type in SET someobject = value, the IDE removes the SET. The problem with default properties is that you can't readily tell the difference between the default property usage and a new object reference. The drawback to removing default properties is that you will always have a property reference, which tends to clutter the code.

> >
> > *4 – The IDE is actually superior in some ways to that of VB 6 (I can't say*
>
> *I'd like to see that <g>*
>
> > *this about the IDE in VB 2002 and VB 2003.). The compiler doesn't stop*
on
> > *the first error and it will even compile in background, putting the*
> > *compile*
>

- > *Hopefully an option. I _want_ the IDE to stop on the first compile error.*
- > *What possible benefit is there to letting it continue? It's just wasting*
- > *time.*
- >

After working in languages that the compiler runs through completely (C and C++ among a dozen or so others) and VB, it is a lot faster to have the compiler report all the errors it can find. Then you start at the top of the list and fix them. Given that the VB 2005 syntax evaluator works in background for you, as you fix one error, if that fixes additional errors they will be removed from the list as well. Although different, this actually leads to faster development than the stop the compile on the first error concept.

- > > *5 – Edit and Continue is back and fully functional. I fixed coding errors*
- > > *on the fly without restarting the debug runs. I copied and pasted in and*
- > > *out of the immediate window. Changed the value of variables in the*
- > > *immediate window. Reexecuted lines of code by copying from the source and*
- > > *pasting into the immediate window.*
- >
- > *Well... *That's* cool (and, about time)*

This is the single item that MS got slammed on by the VB community.

- >
- > > *6 – The locals window is a little harder to use as objects have a lot of*
- >
- > *Not once have I ever used the Locals window in VB5/6*
- >
- > > *7 – The watch window has the same features as the VB 6 watch window, but*
- > > *it*
- > > *also shows the same excessive amount of information as the locals window.*
- >
- > *Side effects of a bloated environment.*

Also a side effect of any true Object Oriented development environment. This is the tradeoff for true OO programming capabilities.

- >
- > > *8 – Intellisense – Way too many options, but the most common ones are*
- > > *first*
- >
- > *Seems like, in this day and age, the IDE could remember which intellisense*
- > *item you selected and show that first next time. Maybe even keep a history*
- > *of the ones you use most, sort them by the number of times they were used*
- > *and show that list first.*
- >

Good idea – send that as a suggestion to MS. Contrary to popular belief, they occasionally listen to their customers (re: Edit & Continue)

>
>> 9 – IDE support for coding – As you close quotes and parenthesis, the IDE
>> highlights the corresponding open mark. I found this really useful when
>> coding lines such as
>
> That is cool and present in VS2003

I think it's also present in VS 2002, but all the other issues with that IDE outweigh it. Excel has it as well and it's really useful when creating complex cell formulas. It's definitely not present in VB 6.

>
>>
>> 10 – IDE support for blocked structures (if then ... end if) When you key
>> in the start of a block the IDE automatically fills in the keyword(s) that
>
> Hopefully an option. CodeSMART 2005 for VB5/6 also does this and it drives
> me nuts. In CodeSMART, it's an option (and turned off here)

Yes – this can be turned off.

>
>> end the block. It also automatically pretty prints your source for you.
>> If
>> the automated pretty printer gets confused, there is a menu option to
>> pretty
>> print the whole file. I discovered that the automatic pretty printer
>> worked
>> about 90% of the time – the rest of the time, failing only when using
>> cut/paste to copy code from one location to another.
>
> Another CodeSMART feature that's just a button click away.
>
>> 12 – Console Applications. These are a royal pain in VB 6 as you must use
>
> Quick (or GW) Basic.
>
>> 13 – A real line/column counter in the editor. I was having to extract
>> data
>> from fixed format, undelimited lines using the Mid\$ function. I opened my
>> source data file in another window and used the line/column information on
>> the status bar to assist in the mid\$(...) statements. What it's missing

> > is
> > a column start/length feature; this could be added to the IDE and I
would
> > add it if I were to find myself having to parse a lot of print files.
In
> > VB
> > 6, I would either have had to manually count my columns or more likely
> > open
> > the file in the VC++ 6 editor were I could use that editors line/column
> > status information.
>
> Maybe missing something here but the VB6 code window **does** show
line/column
> info on the main toolbar (same panel as the compile progress panel)

OK, how do I turn this on in VB 6 SP 5?

>
> > 14 – IDE Code window support for finding errors. Errors are underlined
> > and
> > putting the mouse over the underline will pop up a tooltip explaining
the
> > error.
>
> If the compiler stops on the first error, they're pretty easy to find <g>
>
> > 15 – Debugging unhandled exceptions is a lot easier with the unhandled
> > exception debug window popping up and telling you exactly what the
> > exception
> > is – in English.
>
> Error messages that mean something? Couldn't be from MS... maybe you got a
> 3rd party version of .Net? ;–)
>
> > 16 – The "My" classes. For example get the current desktop, simply code
> >
> > My.Computer.FileSystem.SpecialFolders.Desktop()
>
> That's all cool and stuff but you can do the same thing in VB by placing
the
> code in a class and adding that class to your project.

True, but in VB 2005 it's part of the language – no additional coding
needed. Internally, these call .NET libraries.

>
> > 17 – Sub and Function calls all include a () in the call. If you don't
> > put
> > it there, the IDE will. Non Control–Array property usage is identical
to
> > VB

>
> *That's all we need... a bunch of empty parens in our code <g> Probably not
> an option eh?*

No option – I like the VB 6 syntax better also as it makes for cleaner code.
I also prefer the ability to not use parens on functions unless you are
using the returned value.

Speaking of functions, in VB 6, to return a value, you set the function name
to the value and fall out of the function. In VB 2005, you can still do
this or explicitly code "return <value>". The latter method is more
familiar to someone coming from any other language.

>
> > *6. Control Arrays aren't supported in VB 2005; there is a runtime
> > mechanism
> > to emulate them however. I can't say how easy it is to emulate them as
I
> > haven't done it yet.*
>
> *fwiw, I've found snips that handle "control array" functionality for
vs2003.
> Pretty easy.*
>
> > *18 – The IDE is noticeably slower than that of VB 6. Part of this is
> > simply
>
> No surprise there... so is VS2003's. There's just so much pure junk there
> that it's bound to be slow.*
>
> > *19 – Running a program within the IDE is slower, but the compiled
program
> > ran at about the same speed I would have expected from a compiled VB 6
> > program.*
>
> *Try that with a bunch of controls that need to be refreshed. There are
quite
> a few "why does this take 2–10 times longer to refresh in .Net" type
> questions in the .Net groups.*
>
> > *20 – The compiler generates an intermediate language that can be
> > decompiled.
> > In VB 6, you have an option to generate "native code", but the VB 6
pCode
> > option can also be decompiled. The primary difference is that the .NET
> > intermediate language is documented.*
>
> *That's a deal breaker for alot of people I've talked to. They don't want
to
> go spend \$1500+ for an obfuscator and then another \$XX for software
> protection (trial/shareware) Especially when there are products out there*

- > *that claim to be able to de-obfuscate any .Net code no matter what*
- > *obfuscator they used to hide it.*

How many people compile to pCode in VB 6 without realizing that it can be reverse engineered with realitive ease? Not documenting isn't same thing as security. I've even seen decompilers for C.

- >
- > > *WinSock, WinINet and CDOSYS (used for SMTP mail). You no longer need the*
- > > *much maligned (and rightfully so) FileSystemObject to manipulate*
- > > *directories*
- >
- > *I can't think of a single reason why anyone would "need" the*
- > *FileSystemObject.*

It does make life easier, but you need to watch out for it's rotten performance and the fact that different versions of Windows have different versions of the Scripting Runtime including not at all. If you do any ASP scripting, you must use FSO for file io. VB 6 doesn't have a truely useful "dir" function in that you cannot really nest "dir", so if you're using a library routine or object that uses "dir" inside your own "dir" loop, you're screwed until you realize that there are nested "dir" calls. FSO doesn't have this problem. That said – I hate FSO and have actually recoded my library objects that use it to use the Windows API calls.

- >
- > > *I have documented in the recent thread "OOP" in this NG the improvements*
- > > *in*
- > > *the .NET garbage collector – it's a true mark/sweep/compact GC vs the VB*
- > > *6*
- > > *Reference Count/Compact GC. Basically, the VB 6 GC can be fooled by*
- > > *looped*
- > > *object references. You cannot fool the .NET GC at all, and it handles*
- > > *low*
- > > *memory situations far better than VB 6.*
- >
- > *That's to be seen <g> In VB, if you have a ref count problem, you can*
- > *easily*
- > *fix it (most of the time). Other than that, who cares. VB takes care of*
- > *everything.*
- >
- > > *My recommendation, if you haven't already done so, is to download and*
- > > *install the VB 2005 Express Beta (it's free, even for non MSDN*
- > > *customers)*
- > > *or*
- >
- > *No... sorry... The RTM version of VS2003 is still a beta class product*
- > *imo... I gave up installing MS beta's after having to wipe my drive 3 or 4*
- > *times in the past. Heck, Windows Update caused me to have to wipe my*
- > *drive... and *that's* supposed to be safe! When the final product is*

microsoft.public.vb.general.discussion: Re: VB.NET advantages

- > *released, if I can get my hands on it at that time, I'll probably grab it.*
- > *Meanwhile, let everyone else put up with the headaches <g>*
- >
- > --
- > *Ken Halter – MS–MVP–VB – <http://www.vbsight.com>*
- > *Please keep all discussions in the groups..*
- >

My recommendation stands – anyone interested in VB 2005 should download and install the Beta, knowing that it is Beta and may require a reinstall of VS 6 to remove. Each of us will need to decide what's right regarding VB and it's descendents. Yes VB.Net (7, 7.1, and 2005) are descended from VB 6. Note that the VB 6 mainstream lifecycle support ends on March 31, 2005. Fortunately, these NGs will continue, and if they don't new VB groups can be created in non–MS NG trees.

Mike.