© 2001 Fawcette Technical Publications
Web Content

Newsletter	
Issue Date	
Section	
Main file name	
Accompanying ZIP file name	
Listing file name	
Sidebar file name	
Table file name	
Screen capture file names	
Infographic/illustration file names	
Photos or book scans	
Special instructions for Art dept.	
Editor	
Status	
Spellchecked (set Language to English U.S.)	
EN review	
Character count	
Newsletter blurb	

Overline:

Byline:
by Juval Lowy

Technology Toolbox:
[[Art: Check box for those highlighted below.]]
VB.NET
C#
SQL Server 2000
ASP.NET
XML
VB6

Other:
Xxxxxxx

Note:
Xxxx xxxx xxxx xxx.

Resources:

- xxxxxxx
- xxxxxxx

Head:

C# Vs VB.NET: Choosing your .NET Tool

Deck:

All .NET components, regardless of the language they were developed in, execute in the same runtime called the *Common Language Runtime* (CLR) environment (hence the name). The CLR is like a warm blanket surrounding your code, providing it with memory management, a secure environment to run in, object location transparency, concurrency management, and access to the underlying operating system services. Existing compilers produce CLR-ignorant code; that is, code that does not comply with the CLR type system and code that is not managed by the CLR. Visual Studio.NET comes with four CLR-compliant languages: C#, Visual Basic.NET, JScript.NET and Managed C++.

JScript.NET and Managed C++ are mostly for migration proposes, and it is safe to assume that the majority of .NET developers will choose to disregard them.

The real question as a .NET developer is should you choose C# or VB.NET.

The official Microsoft party line is that all .NET languages are equal in performance and ease of use, and therefore choosing a language should be based on esthetics and personal preference. According to this philosophy, if you come from a C++ background, you will naturally pick up C#, and if you come from a Visual Basic 6 background, you will select Visual Basic.NET. I believe that basing the decision merely on your current language is wrong, and that you have to look not at where you are coming from, but at where you are heading. Let me explain.

The developers community today is roughly divided into two types of development tools users: the rapid application developers and the Enterprise (or system developers). The two types address different business needs, and use different tools. Rapid development today is easier in VB6 than with C++ tools such as MFC, while at the same time, because of its inherit messy nature, VB6 does not lend itself well into large, maintainable applications. In addition, VB6 has built-in imitations that impede scalability (lack of multithreading or object pooling support), and the VB community at large lacks the matching development skills to do system or Enterprise development.

C++ developers on the other hand have access to unlimited power (as well as liability), but mastering ATL or MFC may take years. Applying objects-oriented analysis and design methodologies in a large application requires great deal of skill and time, and will result in unmaintainable code if done poorly.

The two developer communities are therefore distinct not only because of the different languages they use, but primarily, because of the different types of applications developed. In fact, even today, fresh out of the box, C# offers features that VB.NET does not have, and visa versa. For example: C# has native support for automatic disposing of resources (the using statement) to expedite releasing of resources, easing the development of scalable application. C# has native support for locking an object to protect it from concurrent access by multiple threads, and C# has native compiler support for automatic generation of documentation page and development-time tips. All these features are absent from VB.NET. On the other hand, VB.NET has easier syntax for hooking up event handlers to methods (such as the method handling a button click on a form), and it allows for non structured error handling statements, unlike the rigid C# statements.

I believe that these differences are not accidental. It's not that VB.NET developers are exempt from documentation. Far from it. It's just that the product managers at Microsoft had other features, with higher priorities on their list for the first release of VB.NET, features that cater more for rapid development, and less for long term maintenance of intricate Enterprise applications.

I suggest to you that in the future the languages will continue to evolve on deferment paths, each adding features and capabilities that better fit its target users. C# will evolve to better serve the Enterprise market, and VB.NET will be the ideal tool for rapid development.
It is therefore my opinion that when making your decision on which .NET language you choose, you should base on what you intend to use it for, rather than on what is your background. If today you are C++ developer, and you intend to specialize in rapid development, it would be wise to choose VB.NET. If on the other hand, you are a VB6 developer, and you target the Enterprise market, even with no C++ background, you should bite the bullet today, and choose C#.

Captions:
Figure xx
xxxxxxxxx
xxxxxxxxxxxxxxxxx