
Abstract

In this paper an attempt has been made to outline some of the new conceptual features of Visual Basic .NET that were not available in the previous version of Visual Basic 6. These are features that are important for the new generation of programming applications and for mastering programming skills. It was noticed that these topics did not receive sufficient attention or were barely covered in many Visual Basic .NET textbooks for tertiary institutions. As a consequence of this, material was often not included in Visual Basic course curriculums. The importance of some of these concepts is discussed. The author concentrates on the advanced features Visual Basic .NET offers, which would have little impact on introductory programming courses but should be considered for advanced papers. The article is supported by online tutorials and resources provided by the author.

Keywords

Visual Basic, .NET, programming, tutorial

1. Introduction

Visual Basic is and has been the programming language of choice for computing papers in many tertiary institutions for many years. Most tertiary providers have recently switched from the previous version Visual Basic (VB) 6 to the newer VB7, known as Visual Basic .NET. Microsoft has introduced many new (for Visual Basic language) concepts to this version. According to Microsoft, "Visual Basic .NET... is not merely Visual Basic 6 with a few new features added on. Instead, Visual Basic has been thoroughly redesigned and restructured. The language has been modernised, with new, richer object models for data, forms, transactions, and almost everything else" (Microsoft, 2001, p.3). With the launch of Visual Studio .NET in 2003, customers gained both the benefits of an enhanced tool and framework functionality. The coming release of Visual Studio 2005 and the .NET Framework will offer innovations and enhancements to the class libraries, common language runtime (CLR), programming languages, and the integrated development environment (IDE).

Unfortunately quite often textbooks or courses just reproduce old content with a new wrapping, without reflecting new language features. The results of a careful review of some of the available textbooks (Bradley & Milspaugh, 2003,2005, Koneman, 2004, Grundgeiger, 2002, Deitel et al., 2004, , Crews & Murphy, 2004, Gantenbein, 2003, Bell & Parr, 2003, Schneider, 2003, Zak, 2002, Kent, 2002 and others) show that most of them (except Gantenbein) do not mention important new features of VB.NET such as threads, delegates and events, XML, and others. This paper addresses these concepts which are barely covered in textbooks but are quite important for the new generation of programming applications and for the mastering programming skills. The topics discussed here are usually taught in advanced programming papers (year three papers of a degree). Some other concepts such as the object-oriented (OO) nature of the new language or data access are broadly discussed and covered in books and therefore they are not considered in this paper.

To illustrate the new concepts better, some on-line tutorials were designed by the author (Gakhovich, 2005). Detailed consideration of VB6 and VB.NET differences is far
beyond the scope of this paper. However those who are interested may refer to Microsoft's 300-page book (Microsoft, 2001).

2. What's New?

On one hand one could say that Visual Basic .NET does not include major changes to the underlying language; there are merely some useful additions to an already solid set of features. But careful analysis of the VB.NET Language Specification (Vick, 2003) and comparison with the older version shows that concepts such as delegates or threads, are new ones and worth more attention. Also it appears that the newly available tools and techniques for events and event handling increase the programmer's power dramatically.

These concepts are not necessarily interesting and important from the "computer science" perspective but rather because they enhance the RAD (Rapid Application Development) programming tool set and may increase programmer's performance dramatically. Below is a discussion of some new concepts seen from various perspectives.

2.1 Threads

One of the many great features of the Windows OS is multiple threads. Multithreading is such a common element of today's programming that it is difficult to find Windows applications that do not use it. Multithreading, which has been available in Java for years, is new to the VB developer. The VB developer has always wanted this feature in the language and it is now available in VB .NET.

Multithreading is useful for a number of reasons. Multiple threads may be used to handle tasks in the background without freezing up the application's interface, making it responsive at any time. When the task is completed, and even in the process of execution, it updates data in the application so the main thread will be able to tell that it is finished and the reflect changes made. One common example is a file search task that may take a long time. Your application may need to respond to external events (e.g. to stop the search, change the search parameters etc.), and probably the only way to provide such a functionality is to develop an application where presentation and processing logic are divided between separate threads.

Another area where threads could be used is when one needs a service that keeps track of, or regularly monitors, several different things at the same time. Threads let you manage a lot of tasks and events simultaneously.

One of the main uses of multi-threading is in multi-user network applications. In most small-scale single-user Windows applications programmers rely on synchronous processing. However, when we need to develop multi-user software we want concurrent access to network resources, and any server application to be able to accommodate multiple users at the same time; this can be achieved with threads. Most importantly, concurrency concerns itself with how to maintain the integrity of the data or process, when different users want to modify the same bit of information. Therefore it is very important to have a thorough understanding of threads.

2.2 Events and Event Handling

Events have been used in Visual Basic for years. One of the most noticeable things about VB.NET is that the parameters passed to event subroutines such as button event handlers are very different. For example, old VB6 syntax is:

Private Sub Command1_Click ()

Whereas in new VB.NET syntax:

Private Sub Button1_Click(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles Button1.Click

These extensions to event handling syntax give much more power to a programmer. Specifically this enables us to get useful information passed through parameters to an
event subroutine. However, in order to use this functionality a deeper understanding of events and event handling in VB .NET is needed. It is worth the attention as the programming we are doing is essentially event driven programming. It should also be pointed out that a good understanding of object-oriented concepts is necessary to comprehend the new event-handling approach in VB.

One more very useful extension in VB.NET is a dynamic hooking of events to event handlers. To tell Visual Basic .NET that a particular event handler should handle a particular event from a particular object, you need to associate the event handler with the event. In VB6 it was only possible to do this at design time and was normally done by Visual Studio IDE. Now we can use the AddHandler statement to set up the association at run time. This makes it much easier to develop a "dynamically generated" graphical user interface (GUI). For example, to associate a newly created button click event with an event handler we need just one line:

```
AddHandler btnNewlyCreated.Click, AddressOf MyHandler
```

where MyHandler is any sub procedure with a signature matching standard click event handlers (see above).

### 2.3 Delegates

In any consideration of events one should not exclude delegates. The connections between events and event handlers, and also the events themselves, are actually implemented by special objects called delegates. However, the reasons for discussing and using delegates are much broader.

It appears that delegates are one of the hardest of the new concepts for students to understand. One reason is that, when explaining delegates, many authors refer to function pointers available in other programming languages. But students (most of them) have never studied C++ or Delphi. On the other hand, understanding and mastering delegates helps students to use events and multithreading more efficiently. When it comes to asynchronous programming (if you delve into such depth) and networking, the notion of delegates is a must.

So, what is a delegate? Delegates are similar to object references but are used to reference methods instead of objects. Really, delegates are special objects themselves. Unlike "normal" objects, a delegate is the signature of a method rather than the class. And a delegate object instance points to an actual method (of any other object).

Delegates are very important for distributed programming. Application software is often designed around the notion of callbacks. The use of callbacks is a programming technique in which one part of an application (or even a separate application) sends out notifications to alert other parts of the application (or other applications) whenever something interesting has occurred. Delegates are the main tool to provide call-back functions in network applications or communication between threads. It should be mentioned that using delegates in .NET is much safer than using C++ function pointers (in other words, delegates are type-safe pointers).

The implementation and usage of delegates is not really complicated because VB.NET does a lot of things behind the scenes.

### 2.4 XML

Since the inception of XML, many developers have wondered why we need it. In what way is it better than plain text or HTML, and what does it do? The power of XML resides in the "X" (which stands for extensible). The author suggests that any debates about the usefulness of XML are unnecessary here.

Why should XML be used and taught in VB.NET? Firstly, unlike VB6, the new version supports XML natively. Understanding and using XML provides us, as developers, with a number of benefits. Many project files in the Visual Studio use XML format. (It is hard to imagine anyone arguing that students do not need to know anything about that). Probably the best example is application configuration files. They are not a default but
are very handy, and sometimes are the only option for providing a flexible configuration process. Additionally developers may provide their custom XML configuration files (as opposed to App.config files managed by Visual Studio).

Secondly, XML is a core component of the entire .NET Framework. Visual Basic and other .NET languages are tightly integrated with XML. XML is used to transfer data to and from databases, to represent database schemas, in object serialization, etc. In addition to supporting the core .NET Framework infrastructure, a rich set of standards compliant XML APIs are available for use from Visual Studio .NET applications (System.Xml Namespace).

Even a basic understanding of XML may provide a lot of benefits to a programmer. For example, .NET provides such extremely powerful and handy tools to deal with XML based databases that even for a beginner it often is easier to work with XML files than with plain text files.

2.5 And More ...

There are some other topics that are less important compared to above ones but might be very helpful.

VB.NET Collection Structures and Algorithms. The VB.NET is not usually associated with the study of data structures and algorithms. The primary reason for this is because most tertiary providers do not consider VB.NET to be a "serious" programming language. The present state of the language, however, aligns it with other, more serious programming languages like Java or C++. Included in the .NET Framework library is a set of collection classes, which range from the ArrayList, the Stack and the Queue classes, to the Hashtable and the SortedList classes, that all may make life much easier for students, to say nothing of professionals. Students can now use a data structure without needing to implement it (which often requires very advanced skills). Working with such collection classes as ArrayList, Hashtable and others is often much easier than dealing with arrays. Experience shows that students prefer to use these .NET collections because of their operational simplicity.

Custom Controls and Forms. This topic may be quite advanced, and, of course, VB programmers have a wide variety of controls to choose from. So why do we need to talk about creating custom controls? Experience shows that some students (especially last year's project students) are quite keen to design their own fancy controls or special controls that are not available from the standard library. With the object-oriented nature of the VB.NET this task is quite an easy one (Kurniawan, 2002). An amazingly simple and effective technique to create custom shaped Windows forms using gif images with a transparent background was demonstrated at the 2005 NACCQ conference (Dawood, 2005) and is presented on-line (Gakhovich, 2005).

Design Patterns. Design pattern in OO programming are solution templates. Previous incarnations of Visual Basic were not truly object-oriented and it was hard to talk about patterns because they rely on true OO language. Now VB is a fully OO language and object-oriented programming suggests "reusability". But reusability does not only apply to the coding through the reuse of classes and objects; when designing an application in an OO system, solutions to OO design problems can also be re-used. Again, this topic might seem too advanced but even basic familiarity with some patterns like observer-observable, model-view-controller, etc., helps to better understand how things work and produce clearer and better solutions. As a matter of fact, some patterns are used quite often without it being noticed, e.g. when binding GUI controls to datasets.

It is the author's point of view these topics are not given sufficient attention in many textbooks. Though it cannot be claimed that the section above presents an exhaustive list of innovations in .NET, or that that the topics listed are extremely vital, and should become compulsory, the tutorials and other resources that the author has prepared to assist him in covering these topics in programming courses were hopefully worth sharing with the reader.

3. Resources for Teaching
Another issue to be addressed in this paper is the availability of resources that can be used to teach the above mentioned new concepts. This problem exists because most of those textbooks that do cover these concepts are written mainly for professionals, and are too complex for students (e.g. Barwell et al, 2003). One of the main sources of information for VB programmers, the Microsoft Developer Network (MSDN), can also be hard to use by inexperienced programmers such as students.

Fortunately, there are a number of very helpful on-line articles and tutorials available. As an example, one of the web sites worth mentioning provides a collection of VB.NET tutorials aimed at programming students (Rasheed, 2005). Some useful resources can be found on Builder-AU web site (http://www.builderau.com.au/), which also provides subscription for newsletters on programming techniques. Unfortunately, the length of this paper does not permit a discussion or even the listing of all those resources that the author has found to be very useful.

The author would also like to make his contribution to these resources, and wants to present some original materials that cover threads, delegates, events and other matters. These tutorials were prepared for his programming courses and can be found at the author's web site (Gakhovich, 2005). Links to other useful resources are also provided there.

4. Conclusion

Visual Basic and Visual Studio are not static. In October 2005, Microsoft released Visual Studio 2005, with new features such as Design Time Configuration of Single Instance Applications, easy Design Time Configuration of Splash Screens, Compiler Feedback Control, some new language extensions etc.

Timely identification and better realization and understanding of new features in the Visual Basic language may help to bring existing courses up to date, and assist in the development of new ones. It will also better satisfy contemporary industry needs, and prepare students for the new requirements of the market.

Acknowledgements

This paper represents a reviewed, revised and extended version of a paper presented at the 18TH anNUAL NACCQ Conference (Gakhovich, 2005a). The author wishes to thank Jeff Wilson and students Noel Hutchings and Shaun Cleland for their constructive discussions, ideas on the paper topics and their help and support. The author would also like to thank Linda Way and Christine Fenton for the thorough editing of the article in terms of style, language and grammar. This greatly improved the overall presentation and readability.

References

Barwell, F., Blair, R., et.al. (2003). Professional VB.NET, 2nd ed.. Wrox,

Copyright © 2005 Gakhovich, D.

The author(s) assign to NACCC and educational non-profit institutions a non-exclusive licence to use this document for personal use and in courses of instruction provided that the article is used in full and this copyright statement is reproduced. The author(s) also grant a non-exclusive licence to NACCC to publish this document in full on the World Wide Web (prime sites and mirrors) and in printed form within the Bulletin of Applied Computing and Information Technology. Authors retain their individual intellectual property rights.

Copyright ©2005 NACCC.
Krassie Petrova, Michael Verhaart & Tim Hunt (Eds.)
An Open Access Journal, DOAJ # 11764120 , ( ✓zotero)