Sborník vědeckých prací Vysoké školy báňské - Technické univerzity Ostrava
číslo 2, rok 2005, ročník LI, řada strojní
článek č. 1465

David FOJTÍK[*]

INNOVATIONS OF MICROSOFT VISUAL BASIC .NET 2005 LANGUAGE

INOVACE V JAZYCE VISUAL BASIC .NET 2005

Abstract

This contribution describes the latest novelties of the newly prepared Visual Basic .NET 2005 language, whose release is long awaited. The paper introduces both – the new properties, which depends with new generation of Visual Studio 2005 and .NET framework 2.0 and also properties, which are well-known from C# language, but we missed them at Visual Basic so far.

Abstrakt

Tento příspěvek popisuje novinky připravované verze jazyka MS Visual Basic .NET 2005 jehož uvedení se očekává v letošním roce. Článek představuje jednak zcela nové vlastnosti, které souvisejí s novou generací Visual Studia 2005 a .NET farmeworku 2.0, a věnuje se také popisu vlastností, které jsou již známé z jazyka C#, ale které doposud ve Visual Basicu chyběly.

1 INTRODUCTION

This year will, from the point of view of programmes, bring many interesting events with the introduction of the new version of Visual Studio .NET 2005. At the same time, an entirely new version of the .NET framework 2.0 and many other improvements regarding programming languages will appear. Even Microsoft Visual Basic, which was more popular earlier and independently of the .NET platform, will not be speared of these improvements. Surprisingly, this language was left out, when .NET received some very useful features of which the competitive C# was proud of. The new version of Visual Basic .NET 2005 now implements these features, and many more. Just about these innovations this contribution discuss.

2 NEW OPERATOR *ISNOT*

This new operator is simply combination of two operators *Is* and *Not*. Using of this operator makes code execution to be faster and also source code is well arranged (see example of source code below).

```
' This conditions…                 ' Can by write now like this
If Not Object Is Nothing Then      If Object IsNot Nothing Then
' ...                              ' ...
End If                             End If
```

3 NEW CYCLE COMANNDS - *CONTINUE* { *DO* | *FOR* | *WHILE* }

This command (known already from C language) is using in the body of cycles for immediately execution of another loop of cycle and that way leaving out following command of body of cycle. After word *Continue* follows always one of key words (Do, For, While), thereby construction with concrete cycle is specified in case of more nested loops itself.

[*] Ing., Ph.D., Department of Control Systems and Instrumentation, Faculty of Mechanical Engineering, VŠB-TU Ostrava, 17.listopadu 15/2172, Ostrava-Poruba, tel. (+420) 59 732 4223, david.fojtik@vsb.cz

4 VISUAL BASIC DEVELOPMENT WITH MY - KEYWORD MY

This new language element makes an easy orientation at the hierarchical structure of objects, which are accessible during application execution. The new keyword represents an imaginary object on the top of the pyramid of these objects. After typing a dot after word My, IntelliSense offers a reference at least to three primary objects: *My.Application*, *My.Computer*, *My.User*. Through these objects, it is possible to access, for example, the computer file system (*My.Computer.FileSystem*) or to find the name of an actual logged user (*My.User.name*). Depending on the type of application, *IntelliSense* logically offers also other references, for example: *My.Forms*, *My.Webservices* etc.

5 CODE SEPARATION USING PARTIAL TYPES - KEYWORD PARTIAL

An interesting novelty of .NET programming languages is the possibility of separating a class or structure definition into particular parts. It means, that definitions can be composed from several partial fragments, which can be located in different source files. This ability is useful, when more programmers create a class or when it's possible to divide a class into several logical parts.

Individual partial definitions of a class are identified by the keyword *Partial* at Visual Basic. This definition can be then understood as just one part, so that it is possible for another in the same way-declared partial definition to occur at the corresponding namespace. A compiler will put these parts together and will create only one definition of this class. In case of two or more partial definitions, it is possible to leave the keyword *Partial* out of one of them.

FileA.vb

```
Namespace Company
    Public Class CPerson
        Private mFirstName As String
        Private mLastName As String
    End Class
End Namespace
```

FileB.vb

```
Namespace Company
    Partial Class CPerson
        Public Sub New(ByVal FName As String, _
                       ByVal LName As String)
            mFirstName = FirstName
            mLastName = LastName
        End Sub
    End Class
End Namespace
```

6 PROPERTIES WITH MIXED ACCESS LEVELS

In the present version of Visual Basic .NET, it is necessary to define a common access qualifier for both accesses (reading and writing). Now, it is possible to have different access levels. For example, it is possible to define a property, which is fully accessible inside the assembly, but only for reading outside the assembly.

```
Public Property FirstName() As String
    Get:  Return mFirstName: End Get
    Friend Set(ByVal value As String): mFirstName = value: End Set
End Property
```

7 TYPE OF INDEPENDENT DEFINITION (GENERIC TYPES)

The latest novelty of .NET platform is the possibility of creating quite general algorithms with non-predefined data types. This technique can be used for generic classes, structures, interfaces, procedures, and delegates. The principle consists in creating a delegate, which is substituted by a corresponding type during an operation. So that a compiler is already working with a concrete data type and therefore, full data type control is ensured.

As an example, you can see a declaration of class, which contains the only one shared method Change. This method is defined for changing two variables of various data types.

```
Public Class CMyGeneric(Of DelegateType)
    Public Shared Sub Change(ByRef first As DelegateType, ByRef second As DelegateType)
        Dim dump As DelegateType = first
        first = second
        second = dump
    End Sub
End Class
```

From the example in Visual Basic, it can be seen that a new keyword *Of* occurred in this relation, to identify the delegate. This delegate is used instead of the data type during variable declaration. The object can be used as follows:

```
Sub test()
    Dim s1 As String = "First", s2 As String = "Second"
    CMyGeneric(Of String).change(s1, s2)
    MsgBox(String.Format("s1 = {0}, s2 = {1}", s1, s2)) 'S1 = Second, s2 = First
    Dim n1 As Long = 100, n2 As Long = 200
    CMyGeneric(Of Long).change(n1, n2)
    MsgBox(String.Format("n1 = {0}, n2 = {1}", n1, n2)) 'n1 = 200, n2 = 100
End Sub
```

In general, .NET Framework 2.0 offers in the namespace *System.Collections.Generic* several similar-defined classes for working with collections

8 THE USING COMMAND

Visual Basic .NET is, in contrast to its previous versions, already fully object-oriented. However, this positive fact brings some complications due to the concept used. Especially the problem is with object lifetime control, strictly speaking with its destruction. There is Garbage Collector, which cares about object destruction. Its behavior isn't deterministic, so that problem is, at this moment, in the necessity to immediately detach used sources. Of course, this problem is well known, such that a programmer can force Garbage Collector to an object's immediate destruction or it can rather detach sources by the Dispose method.

At Visual Basic.NET 2005 it is now possible to make this action easy by the Using command. The Command Using defines a block of source code, in which the object is valid, and after leaving this block by various ways it is automatically detached. The following source code example describes the principle of this action.

```
Private Sub PrintText(ByVal Text As String, ByVal Grph As Graphics)
    Using F As New Font("Arial", 14), BrushText As New SolidBrush(Color.Blue)
        'Objects F and BrushText only exist in this block
         Grph.DrawString(Text, F, BrushText, 0, 0)
    End Using
    'Objects F and BrushText not exist now
End Sub
```

9 GENERATION OF XML DOCUMENTATION IN VISUAL BASIC

The possibility of creating documentation from special XML comments was until now the privilege of C# language only. Now, this property is available also at the new Visual Basic .NET 2005. The principle consists in creating a special commentary separated by the ''' symbol (three apostrophes) and after them the required XML text, with corresponding elements follows. These commentaries automatically use the tools IntelliSense and Object Browser. Optionally, the compiler can use them for the creation of documentation.

```
''' <summary>
''' Parameterized constructor of The Object
''' </summary>
''' <param name="Numerator">The Numerator of The Fraction</param>
''' <param name="Denominator">
''' The Denominator of The Fraction (a value must be different to zero)
''' </param>
Public Sub New(ByVal Numerator As Long, ByVal Denominator As Long)
    mNumerator = Numerator: mDenominator = Denominator: Normalize(Me)
End Sub
```

10 OPERATORS OVERLOADING

Among the interesting properties, which have been missing at Visual Basic till now, is operator overloading. This property allows he use of standard operators with objects or structures, as well as ordinary variables. For example, it is possible to define a class there, which represents a fraction, and it can be worked with their objects by standard operators +, -, *, / etc.

```
Sub Main()
    Dim f1 As New CFraction(7, 6), f2 As New CFraction(2, 5)
    Dim f As CFraction = -(f1 + f2)
    Console.WriteLine("({0})+({1})=({2})", f1.ToString, f2.ToString, f.ToString)
End Sub
```

It can be seen from the example, that operator overloading is executed by the key word *Operator*, followed by the operator symbol. In addition, this function must be always *Shared*. Of course it's possible to overload both the binary as well as unary operators. In case of using one operator in unary and also in binary form simultaneous definition will be simply distinguished by a number of parameters. Definition of using operators at Visual Basic:

```
'Operator of binary: Fraction = Fraction1 - Fraction2
Public Shared Operator -(ByVal L As CFraction, ByVal R As CFraction) As CFraction
    Dim Result As New CFraction
    If L.Denominator = R.Denominator Then
        Result.Denominator = L.Denominator
        Result.Numerator = L.Numerator - L.Numerator
    Else
        Result.Denominator = L.Denominator * R.Denominator
        Result.Numerator = L.Numerator * R.Denominator - R.Numerator * L.Denominator
    End If
    Normalize(Result): Return Result
End Operator
'Operator of Unary: Fraction = -Fraction
Public Shared Operator -(ByVal Fraction As CFraction) As CFraction
    Return New CFraction(-Fraction.Numerator, Fraction.Denominator)
End Operator
```

11 CONCLUSIONS

As results from text, the new version of language brings many improvements, which makes language more effective. Another amount of novelties and many times more important brings itself .NET Framework 2.0 together with Visual Studio .NET 2005. Most of them are focused on the support of desktop applications. It can be concluded then that Microsoft would like to promote the.NET platform at this level as well. Only the time will show if the Visual Basic .NET 2005 will replace the previous version of Visual Basic 6.0.

REFERENCES

[1] MICROSOFT. *MSDN Library for Visual Studio 2005 Beta 2*. Redmond : Microsoft Corporation Redmond USA, April 2005.

[2] BABIUCH, M. What Can We Expect In ASP.NET 2.0? Ostrava. In *Proceedings of XXX. Seminary ASR '05 "Instruments and Control"*. Ostrava : Katedra ATŘ, VŠB-TU Ostrava, 29. 4. 2005, s. 31-38. ISBN 80-248-0774-2.

[3] KULHÁNEK, J. The Speed of Component Based Application in .NET Platform. In *5th International Carpathian Control Conference*. Zakopane, Poland : AGH-UST Krakow, 25. - 28. 5. 2004, pp. 843-848. ISBN 83-89772-00-0.

[4] FARANA, R., SMUTNÝ, L. & BABIUCH, M. Experimental and Virtual Laboratories in Distance Learning, In *Proceedings of the 2004 WSEAS International Conference on APPLIED MATHEMATICS*. Corfu, Greece : WSEAS, August 17-19, 2004, Paper 488-223, 4 pp. ISBN 960-8457-01-7.

Reviewer: prof. Dr. RNDr. Lubomír Smutný, VŠB-Technical University of Ostrava