

# Die Zukunft von Visual Basic

## Ein erster Ausblick auf Version 10

Stand: 4/01/2009

Autor: Peter Monadjemi

Der folgende Überblick versteht sich in erster Linie als „Appetithäppchen“ und früher Wegweiser zum geplanten Nachfolger von Visual Basic 9.0. Seit November 2008 gibt es eine allererste Vorabversion vom .NET Framework 4.0, und damit von C# 4.0 und VB 10, die aber noch nicht einmal CTP-Status besitzt (und die sich daher auch niemand „antun“ sollte – mehr dazu in der Einleitung). Das fertige „Visual Basic 10“ dürfte als Teil von .NET 4.0 und „Visual Studio 2010/11“ sicher nicht vor Sommer/Herbst 2010 kommen (was aus meiner Sicht auch vollkommen ausreichend ist).

## Inhalt

- § Ein paar einleitende Worte
- § Alle Neuerungen (sofern bekannt) auf einen Blick
- § Mehrzeilen-Lambda's und Subs als Lambda
- § Einzeilige Properties (Auto-Properties)
- § Der Zeilenfortführungsoperator wird überflüssig
- § Collection-Initialisierer
- § Nullfähige optionale Parameter
- § Covarianz und Contravarianz
- § COM ohne PIA
- § Zusammenspiel mit dynamischen Programmiersprachen
- § Ausblick auf kommende Versionen

Meine Empfehlung gleich vorweg: Der Download der Pre-Alpha von VS 2010, die Microsoft seit der PDC als Download anbietet lohnt sich definitiv nicht. Die Version ist (jedenfalls auf meinem Vista SP1-PC mit 2 GByte RAM) instabil (sie stürzt regelmäßig ab), es fehlen viele wichtige Merkmale und es ist in erster Linie eine Vorschau auf die nächste Version von VSTS (mit dem 3. Januar 2009 wurde die Version zudem „ungültig“ - es gibt inzwischen einen „Patch“, der zu mindestens dieses kleine Problem löst).

Und jetzt zu den etwas erfreulicheren Dingen.

Das VB-Team bei Microsoft hat die wichtigsten Neuerungen in einem kleinen Word-Dokument [zusammengefasst](http://code.msdn.microsoft.com/vbfuture), das es unter <http://code.msdn.microsoft.com/vbfuture> zum Download gibt. Des Weiteren gibt es ein nettes Video auf Channel 9 (<http://channel9.msdn.com/posts/Dan/Lucian-Wischik-and-Lisa-Feigenbaum-Whats-new-in-Visual-Basic-10>), ein paar Blog-Einträge und natürlich den Vortrag auf der PDC 2008 mit dem vielversprechenden Titel „Future directions for Microsoft Visual Basic“, in dem VB10 zum ersten Mal der Öffentlichkeit vorgestellt wurde. Alle PDC-Vorträge gibt es auch zum Anschauen: <https://sessions.microsofttpdc.com/public>.

Wie bereits im Herbst 2005 auf der PDC 2005, damals ging es um das kommende Visual Basic 9.0, das dann erst Anfang 2008 offiziell wurde, gab es auch auf der letzten PDC 2008 einen Vortrag, in dem es um die nächste Visual Basic-Version ging. Trotz des vielversprechenden Titels war der Vortrag auf der PDC thematisch leider ein wenig „durchwachsen“ (und insgesamt alles andere als visionär). Nach einer kurzen Einleitung von Paul Vick, der bis kurzem als Architekt maßgeblich am „Sprachdesign“ von Visual Basic mitgearbeitet hatte, vor kurzem aber das Visual Studio-Team in Richtung Oslo verlassen hat (nicht in die Stadt, sondern in das Team), und der noch ein einmal einen kurzen Rückblick auf die Rolle von Visual Basic als Werkzeug gab, das die jeweils aktuellen Features der Windows-Plattform zugänglich machte, und dabei betonte, dass Visual Basic und C# wieder „in sync“ gebracht werden und die jeweils andere Sprache über kurz oder lang die Neuerungen der anderen Sprache erhalten wird, übernahm mit Lucian Wischik sein Nachfolger als neuer „Spec Lead“ den Vortrag. Lucian begann gleich mit einem absoluten „Killer-Feature“, in dem er zuerst ein Array deklarierte (auch Arrays können ab VB 10 ohne Typenangabe initialisiert werden):

```
Dim Scores() = {10, 20, 30, 40, 50, 60}
```

und anschließend über die ForAll-Methode der Array-Klasse eine Lambda-Funktion aufrief. Dass diese „Funktion“ bei VB 10 auch mehrzeilig sein kann, die einzelnen Zeilen nicht mehr per Zeilenfortführungsoperator verbunden werden müssen und die Function auch eine Sub sein kann war bereits eine kleine Sensation, der „Hammer“ war allerdings als er mit der neuen AsParallel-Methode, die es bei Arrays, Collections und LINQ-Abfragen gibt, auf Parallelverarbeitung „umschaltete“:

```
Scores.AsParallel().ForAll(Sub (o) Console.WriteLine(o))
```

So einfach wird es in Zukunft sein, einen Thread auf mehreren CPUs/Kernen auszuführen, wobei dies weniger das Verdienst von VB bzw. C# ist, sondern der .NET 4.0-Runtime und einer entsprechenden Erweiterung der .NET-Klassenbibliothek. Einfacher kann es sicher nicht mehr werden.

### Alle Neuerungen (sofern bekannt) auf einen Blick

Um es gleich vorweg zu nehmen, allzu viel erwartet Visual Basic-Entwickler nicht mit der nächsten Version (bei C# 4.0 sieht es ähnlich aus). Damit es übersichtlich bleibt, stellt Tabelle 1 die wichtigsten geplanten Neuerungen zusammen, stets verbunden mit dem Hinweis, dass noch das eine oder andere Sprachmerkmal hinzukommen genau wie das eine oder andere Merkmal verschwinden kann. Die Version dürfte in den kommenden Wochen eingefroren werden.

Stichwort	Was steckt dahinter?	Neue Schlüsselwörter	Persönliche Wertung
Array-Literale	Auch (lokale) Arrays ohne Typenbezeichnung können direkt initialisiert werden.	-	Nützlich.
Collection-Initialisierer	Genau wie ein Array kann auch eine (generische) Collection mit Werten initialisiert werden.	From	Sehr nützlich.
Mehrzeilige	Bei VB 9 darf der „Functionscode“, der	-	Sehr

Lamdbas	einer Funktionsvariablen zugewiesen wird, nur aus einem Ausdruck bestehen. Ab VB 10 kann es auch eine Befehlsfolge sein.		nützlich.
Lambdas mit Subs	Bei VB 9 muss ein Befehl/Ausdruck, der einer Funktionsvariable zugewiesen wird, immer einen Wert zurückgeben. Ab VB 10 ist dies nicht mehr erforderlich, so dass es auch eine Sub sein kann.	-	Nicht spektakulär, aber wichtig, um eine sprachliche Parität zu C# herzustellen und manchmal sicher auch praktisch.
Zeilenfortführungsoperator	Der Zeilenfortführungsoperator kann in vielen Situationen entfallen. Davon profitieren vor allem mehrzeilige Lambdas, die sich damit einfach „herunterschreiben“ lassen.	-	Es gibt sicher wichtigeres, in LINQ-Ausdrücken aber mit Sicherheit praktisch.
Property-Abkürzung	Wenn bei einer Property Set- und Get-Accessor lediglich eine private Variable ansprechen, kann diese Definition durch Public/Friend ersetzt werden.	-	Praktisch und überfällig.
Integration mit dynamischen Sprachen	Objekte, die in IronPython oder IronRuby angelegt wurden, können direkt in VB angesprochen werden.	-	Wer's braucht:)
Parallelverarbeitung	Über die AsParallel-Erweiterungsmethode lassen sich Collections und LINQ-Abfragen „parallelisieren“.	AsParallel	Geradezu revolutionär

Tabelle 1: Die für VB 10 geplanten Neuerungen

## Mehrzeilen-Lamdbas und Sub als Lambda

Ein Lambda-Ausdruck (also Funktionscode, der ohne, dass dafür eine Funktion definiert werden muss, als Ausdruck übergeben oder einer Variablen zugewiesen werden kann) wird sich bei VB 10 auch über mehrere Zeilen erstrecken können und es kann auch eine Sub sein, die dann z.B. auch einem Aktionspredikat-Delegaten übergeben werden kann. Dieser Konstrukt entspricht den anonymen Methoden von C# 2.0, wenngleich er bei VB 10 nicht so genannt wird. Damit ist es z.B. möglich, beim Anlegen eines neuen Threads anstelle eines AddressOf <Prozedurname> die Prozedur direkt (inline) einzufügen.

### Beispiel

Das folgende Beispiel fasst gleich mehrere neue Merkmale zusammen. Es weist der Variablen PrimCheck einen mehrzeiligen Funktionscode zu und ruft diese Lambda-Funktion für jede Zahl in einem Array auf, das zuvor mit den neuen Array-Literalen initialisiert wurde, ohne dass dabei ein Typ angegeben wurde. Die Ausgabe erfolgt in einer Sub, die einer Variablen zugewiesen wird.

Für alle „Skeptiker aus Leidenschaft“: Es ist vollkommen klar, dass man für diese trivialen Beispielchen nicht die Neuerungen von VB 10 benötigt. Das Beispiel dient daher lediglich zur Veranschaulichung (ich bin nach wie vor der Meinung, dass Beispiele so einfach wie möglich sein sollten – aber nicht einfacher).

```
Shared Sub Main()
' Beispiel für Multiline-Lambdas und die neue Freiheit am Zeilenende
Dim PrimCheck = Function(n As Integer) As Boolean
    For i As Integer = 2 To Math.Sqrt(n)
        If n Mod i = 0 Then
            Return False
        End If
    Next
    Return True
End Function
' Dank Array Literals muss auch bei einem Array kein Typ mehr angegeben
werden
Dim TestZahlen() = {3, 5, 9, 12, 17, 19, 25}
Dim Prims1 = TestZahlen.Where(PrimCheck)
Console.ForegroundColor = ConsoleColor.Green
For Each P In Prims1
    Console.WriteLine(P)
Next
' Lambdas können jetzt auch Subs sein
Array.ForEach(TestZahlen, Sub (n)
    Console.WriteLine(String.Format("{0} ist eine Primzahl: {1}",
    n, PrimCheck(n)))
End Sub)
```

## Einzeilige Properties (Auto-Properties)

Diese nette „Abkürzung“ war bereits Teil der ersten Vorabversion von Visual Basic 9.0, wurde dann aber wieder herausgenommen: Die Möglichkeit, eine Property in einer Zeile definieren zu können, wenn „Getter“ und „Setter“ lediglich das private Feld zurückgeben bzw. den zugewiesenen Wert zuweisen.

### Beispiel

Das folgende Beispiel definiert eine Klasse mit zwei vollständigen Properties.

```
' Properties werden ein wenig einfacher deklariert
Class Team
    Property Name As String
    Property Gruendungsjahr As Integer
End Class
```

## Der Zeilenfortführungsoperator wird überflüssig

Große Veränderungen kündigen sich an. Der Zeilenfortführungsoperator wird in bestimmten Konstruktionen, vor allem in LINQ-Ausdrücken, überflüssig. Die Umsetzung scheint wohl nicht ganz so trivial gewesen sein wie sie sich anhören könnte. Wie es in dem eingangs erwähnten offiziellen VB 10-Word-Dokument ausführlich beschrieben wird, gibt es aber nach wie vor Situationen, in denen der Zeilenfortführungsoperator erforderlich ist.

## Collection-Initialisierer

Auch dieses Feature war bereits für Visual Basic 9.0 im Gespräch, wurde aber nicht realisiert. Genau wie einfache Objekte werden sich in Zukunft auch

Collections wie z.B. List(Of Typ) oder Dictionary(Of Typ) direkt initialisieren lassen. Das ist wirklich praktisch. Mit From erhält das Schlüsselwort eine erweiterte Bedeutung (intern wird für jedes Element eine Add-Erweiterungsmethode aufgerufen).

#### Beispiel

Das folgende Beispiel initialisiert eine Collection mit Team-Objekten, die anschließend durchlaufen wird.

```
' Collections können direkt initialisiert werden
Dim Teams As New List(Of Team) From {
    new Team With {.Name="VFB Stuttgart", .Gruendungsjahr=1893},
    new Team With {.Name="1 Fc Cannstadt"},
    new Team With {.Name="Stuttgarter Kickers" } }

Console.ForegroundColor = ConsoleColor.Magenta

For Each T In Teams
    If T.Gruendungsjahr = 0 Then
        PrintTeamInfo(T.Name)
    Else
        PrintTeamInfo(T.Name, T.Gruendungsjahr)
    End If
Next
```

## Nullfähige optionale Parameter

Wird an den Namen eines optionalen Parameters ein ? gehängt, wird daraus ein „Nullable-Type“. Er besitzt damit eine HasValue-Property, über die festgestellt werden kann, ob der übergebene Value-Type einen Wert besitzt.

#### Beispiel

Das folgende Beispiel definiert eine Prozedur mit einem optionalen Nullable-Parameter.

```
' Optionale Parameter können Nullable sein
Shared Sub PrintTeamInfo(TeamName As String, _
    Optional Gruendungsjahr As Integer? = 1900)
    Console.WriteLine("Das Gründungsjahr von {0} ist {1}", _
        TeamName, Gruendungsjahr)
End Sub
```

## Covarianz und Contravarianz

Dieses Feature wird es sowohl bei C# 4.0 auch bei Visual Basic 10 geben. Covarianz bedeutet (in etwa), dass einer generischen Collection auch ein Typ zugewiesen werden kann, von dem sich der Typ, zu dem die Collection typenstreng ist, ableitet. Contravarianz bedeutet (vermutlich, weil mir dieser Punkt noch nicht ganz klar ist – ich hätte bei dem Basta!-Vortrag von Oliver Sturm zu „C# - Advanced Features“ besser dabei sein sollen), dass sich z.B. ein String-Array einer Variablen vom Typ Object-Array zugewiesen werden kann, auch wenn sich das String-Array von der Array-Klasse und nicht der Object-Klasse ableitet. In jedem Fall gibt es etwas mehr Flexibilität beim wechselseitigen Zuweisen von Objekten zwischen deren Typen eine „Beziehung“ existiert.

## COM ohne PIA

Dies keine Neuerung der Sprache, sondern ein neues Feature der CLR 4.0. Greift ein .NET-Programm über eine PIA (Primary Interop Assembly) auf eine COM-Bibliothek zu, muss die PIA nicht unbedingt mitausgeliefert werden, da der Compiler die über die PIA zur Entwicklungszeit angesprochenen Typen dynamisch in der Ziel-Assembly nachbilden kann, so dass die PIA nicht mehr benötigt wird. Dies ist vor allem bei .NET-Office-Anwendungen interessant, wo die verschiedenen PIAs noch mit ausgeliefert wurden, um sicherzugehen, dass sie auf dem Zielsystem auch vorhanden sind.

## Zusammenspiel mit dynamischen Programmiersprachen

Die Dynamic Language Runtime (DLR), auf deren Grundlage von Microsoft die dynamischen Sprachen IronPython und IronRuby implementiert wurden, steht auch für VB und C# zur Verfügung. Damit kann ein VB-Programm direkt auf Objekte in einem Ruby-Programm zugreifen, die so dynamisch sind, dass sie jederzeit neue Mitglieder erhalten können.

## Ausblick auf kommende Versionen

Bei Microsoft plant man bekanntlich immer eine Version im Voraus. Wenn die Entwickler jetzt mit Hochdruck an VB 10 arbeiten, ist ein „VB11“ bereits in der Planungsphase. Am Ende des PDC-Vortrags gab Paul Vick einen kurzen Ausblick auf kommende Versionen, ohne allerdings allzu sehr ins Detail zu gehen. Drei wichtige Trends für die Nachfolger von VB10 werden weitere dynamische Spracheigenschaften, weitere Sprachmittel für die parallele Ausführung und noch mehr deklarative Sprachelemente sein (Paul nannte dies den „most important trend“). Die XML-Literale sind ein solcher „deklarativer Trick“, weitere sollen folgen, wenngleich es alleine aus Ressourcengründen nur möglich ist, 1-2 dieser Erweiterungen pro Version unterzubringen. Die „Iteratoren“, die es in C# bereits seit Version 2.0 gibt, und mit der beim Durchlaufen einer Enumeration das aktuelle Objekt außerhalb der Enumerationsschleife verfügbar gemacht werden kann, werden wohl erst mit „VB 11“ kommen. Das gleiche gilt für andere Features, die als Wünsche von außen an das Entwicklungsteam herangetragen werden. Ein wenig von dem, was alles geplant ist, hatte Paul Vick bereits vor der PDC in seinem stets lesenswerten Blog zur Diskussion gestellt (z.B. unter <http://www.panopticoncentral.net/archive/2008/08.aspx>).

Auch die von Microsoft-Chef-Entwickler Anders Hejlsberg für kommende Versionen von C# in Aussicht gestellte Öffnung des Compilers dürfte es wohl erst frühestens mit „VB 12“ geben.

Mein persönliches Fazit ist, dass die wenigen Neuerungen den Sprachkomfort abrunden und den „RAD-Charakter“ von Visual Basic betonen. Microsoft ist es gelungen, den ursprünglichen Charakter von Visual Basic, bei dem sich die Entwickler um die meisten Formalitäten keine Gedanken machen mussten, auf eine neue Grundlage zu stellen, in dem die Freiheit von eins wieder da ist, ohne dass die Entwickler ein schlechtes Gewissen haben müssen, da sich der Compiler z.B. darum kümmert, dass stets typenstrenger Code herauskommt. Es wird aber wohl noch eine Weile dauern, bis Profi-VB-Entwickler wieder z.B. ein Dim A = 10 schreiben, ohne dass sie dabei ein ungutes Gefühl beschleicht.

Wer das alles einmal selber ausprobieren möchte, sollte sich noch ein wenig gedulden und solange warten bis es die erste CTP bzw. Beta der Express Edition von „Visual Basic 2010“ gibt, die keine 7 GByte umfassen dürfte. Das dürfte, nach meiner rein privaten „Schätzung“, zwischen März und Mai der Fall sein. Die aktuell

angebotene Pre-Alpha ist nicht nur völlig aufgebläht (früher gab es dafür den Begriff „Bloatware“, was sich mit „Blähware“ übersetzen ließe), es fehlen auch noch wichtige Neuerungen, wie z.B. Contravarianz bei Schnittstellen und Delegates. Falls es doch jemand nicht abwarten kann, empfehle ich, die VB- und C#-Beispiele in der Kommandozeile zu kompilieren, dann kann zu mindestens nichts abstürzen (so hat es jedenfalls bei mir ganz gut funktioniert).

Peter Monadjemi, Januar 2009