

Wohin entwickeln sich die Sprachen mit Whidbey?

# Keine Frage der Sprache

Auf die beiden Artikel von Marcel Gnoth über die Unterschiede von C# und Visual Basic .NET erhielten wir viele Reaktionen. Es gab Kritik, aber auch jede Menge Hinweise. Aus diesem Grund greifen wir die Thematik noch einmal auf und fügen neue Fakten hinzu.

**O**bwohl es doch eigentlich nur um ein Stück Technologie geht, ist die Wahl der Sprache in vielen Fällen eine emotionale Entscheidung.

Auf die beiden VB.NET-C#-Artikel [1] und [2] bekamen wir überraschend viele Zuschriften. Einige waren eher emotional geprägt, andere mehr sachlich. Einige Punkte waren sehr interessant und auf diese soll hier eingegangen werden.

## Standardisierung

Ein Leser machte darauf aufmerksam, dass in [1] die Standardisierung von C# unterschlagen wurde. Microsoft hat tatsächlich und völlig untypisch die Sprache C# und die Common Language Infrastructure (CLI) bei ECMA und ISO zertifizieren lassen. Einige große Firmen und öffentliche Auftraggeber wünschen eine solche Zertifizierung. Doch was sind die Vorteile einer solchen Zertifizierung? Andere Hersteller können jetzt eigene C#-

und CLI-Implementierungen auch auf Nicht-Microsoft-Plattformen entwickeln (siehe [3]). Für den klassischen Berufsentwickler dürfte das aber nicht sonderlich interessant sein.

Auf der anderen Seite ist Microsoft nicht gerade ein Vorbild, was Standardisierungen angeht. Oft genug hat Microsoft in der Vergangenheit internationale Standards nach eigenem Gusto ausgeweitet oder gebrochen. C# und VB.NET sind beides Microsoft-Kreationen und Microsoft wird diese so weiterentwickeln, wie es Microsoft opportun erscheint.

Folgendes Beispiel, dass dies schon passiert: Die Unterscheidung von Bezeichnern nach Groß- und Kleinschreibung ist sehr wohl konform zur .NET-Laufzeitumgebung (CLR), nicht aber zur Common Language Specification (CLS) – wie fälschlich in [1] behauptet wird. Die CLS soll ermöglichen, dass Assemblies unterschiedlicher Sprachen zusammenarbeiten können. Firmen, die eine Sprache für .NET entwickeln wollen, sollten sich normalerweise an die CLS halten. Was aber macht Microsoft?

In der neuen Version wird Visual Basic genauso wie C# über vorzeichenlose Variablen verfügen, und das entspricht nicht der CLS. Dazu gesellen sich dann noch Property-Prozeduren, deren *Get*- und *Set*-Blöcke einen unterschiedlichen Scope haben können, wie in VB6.

Woran soll sich eine Firma orientieren, wenn sie eine .NET-Sprache entwickeln möchte? An der CLS oder an dem De-facto-Standard, den VB und C# vorgeben?

Theoretisch ist es möglich, den gleichen C#-Code auf einem Windows-Rechner und auf einem Linux-Rechner [3] zu kompilieren. Praktisch ist es eine Frage, welche Komponenten des Frameworks auf dem jeweiligen Rechner zur Verfügung stehen. Die paar Schlüsselwörter spielen nicht mehr die große Rolle. Das Framework und die CLR sind jetzt die Spieler.

## Positionierung durch Microsoft

Auf der PDC und auf der TechEd war etwas über die Positionierung der Sprachen zu hören. Auch in der Road Map [4] in der MSDN werden unterschiedliche Schwerpunkte hervorgehoben. Wenn VB ein Rapid Application Development (RAD) Tool ist, ist C# dann ein SAD Tool? Ein Slow Application Development Tool?

Microsoft hat zurzeit zwei wichtige .NET-Sprachen auf dem Markt, VB.NET und C#. VC++ spielt eine andere Rolle und soll deshalb hier nicht betrachtet werden. Es gibt keine Anzeichen dafür, dass es eine der beiden Sprachen in Zukunft nicht mehr geben wird. In der Beta 1 von Whidbey sind genügend Beispiele in VB.NET wie auch in C# enthalten.

## Unterschiedliche Prioritäten

Vielmehr setzen die beiden Entwicklungsteams unterschiedliche Prioritäten. Für das VB.NET-Team steht einfache Erlernbarkeit und effiziente Anwendungsentwicklung im Vordergrund. Für das C#-Team steht die Entwicklung von Business Frameworks und wiederverwendbaren Komponenten im Vordergrund.

Auf lange Sicht werden immer beide Sprachen über wichtige Features der jeweils anderen verfügen. Ein gutes Beispiel sind Generics. Nachdem Generics ein riesiges Presse- und Expertenecho bekamen, wurden sie flugs in VB.NET eingebaut. Und auch Edit-and-Continue, eine Funktion der VB-Umgebung, wird laut letzten Informationen [7] in C# 2005 enthalten sein. Damit ist es dann auch C#-Programmierern möglich, während des Debuggens Veränderungen am Code vorzunehmen.

## Portierung alter VB6-Projekte

Es sind wesentlich weniger VB6-Entwickler auf .NET umgestiegen, als von Microsoft erhofft war. Somit lautete die Haupt-

## Auf einen Blick

### Autor



**Marcel Gnoth** ist Leiter der Entwicklungsabteilung bei der Berliner NTeam GmbH, die sich als Microsoft Gold Certified Partner auf die Server- und Entwicklerprodukte von Microsoft spezialisiert hat. Neben seiner Arbeit als Programmierer führt er Trainings für Entwickler durch, schreibt für verschiedene Fachmagazine und tritt regelmäßig als Sprecher etwa auf der BASTA auf. Seine Schwerpunkte liegen bei COM, Informationssystemen und der .NET-Programmierung. Sie erreichen ihn unter marcel@gnoth.net und über www.gnoth.net.

frage, wie man die VB6-Programmierer zu .NET bewegen könnte. Schließlich will Microsoft auch mit seinen Entwicklungs-Tools Geld verdienen. Also lag ganz klar eine Priorität von VB.NET 2005 darin, die Portierung von VB6-Anwendungen zu erleichtern. Inwieweit sich die Portierung mit Visual Studio 2005 verbessern wird, muss die Praxis zeigen. Aber das neue *My*-Schlüsselwort und auch die *Default-Instance* einer Form werden dabei helfen.

In VB6 gab es für jedes Formular automatisch eine Instanz, die über den gleichen Namen wie die Form angesprochen werden konnte. Es musste nicht mit *New* eine Instanz der Form erzeugt werden.

### Visual Basic: die einfache Sprache für Einsteiger

Der zweite Schwerpunkt lag darin, mehr Einsteiger zu .NET zu locken. Einige Leser haben die mangelnde Qualifikation vieler VB-Programmierer bemängelt. Aber dass viele Einsteiger sich in der Sprache versuchen, sagt nur aus, dass die Sprache einfach ist, und nichts über die Qualität der Sprache. Und dann frage ich mich immer, was C-Entwickler in den ersten drei Jahren ihrer Programmierpraxis machen? Denn so lange dauert es, bis ein Entwickler wirklich Erfahrung hat und guten Code produziert.

### VB-Funktionen in C# verwenden

In [1] stand, dass etliche Funktionen aus dem Visual-Basic-Namespace auch in C# verwendet werden können, was natürlich von vielen erst einmal für abwegig gehalten werden wird. Aber halt! Die klassische VB-Collection ist zum Beispiel für manche Aufgaben viel besser geeignet. Und auch der Namespace *Microsoft.VisualBasic.Financial* bietet eine Reihe sehr interessanter Funktionen aus der Finanzwelt, beispielsweise so etwas wie Zinsrechnung.

Auch das künftige *My*-Schlüsselwort von VB ist in C# über *Microsoft.VisualBasic.MyServices* verwendbar. Es bietet einige Abkürzungen im Framework und der Entwickler kann dadurch einige Codezeilen sparen, die dann im Hintergrund erledigt werden. Unter Visual Basic .NET 2005 hingegen bietet das Schlüsselwort *My* derzeit Zugriff auf Objekte wie *Application* oder *Computer* (siehe [6]).

Auch der *Microsoft.VisualBasic.Interaction*-Namespace ist einen Blick wert. Er bietet *CreateObject* oder *GetObject* und

anderes. Das sind ganz normale .NET-Assemblies, die von jeder .NET-Sprache aus verwendet werden können.

### Late Binding

Obwohl ich jedem erzähle, wie wichtig *Option Strict* ist, habe ich heute tatsächlich diese Option ausgeschaltet und Late Binding verwendet. Der Zugriff auf das Excel-Objektmodell von .NET aus ist teilweise sehr anstrengend. Zu oft liefert die COM-Interop-Schicht einen Datentyp als *Object* zurück. In solchen Fällen ist Late Binding schon ganz nützlich.

### Die Power von VB6

Eine kleine Anekdote aus Dan Applemans Vortrag [5] setzt VB6 ins rechte Licht. „Was ist Leistungsfähigkeit? VB6 hatte: verwalteten (managed) Speicher, keine Pointer und Garbage Collection. Das waren neue Schlüsselfunktionen in .NET: mehr Sicherheit und Zuverlässigkeit.“ „Also“, schloss Dan Appleman daraus, „haben jetzt alle C++-Entwickler in C# endlich die volle Power von VB6!“

### Ausblick auf VB.NET 2005

Whidbey wird viele neue Funktionen mitbringen. Aber: Ein Rechner sollte mindestens 500 MByte Arbeitsspeicher haben, ab 1GByte macht es erst Spaß (siehe MSDN und [4], [5], [6], [7]). In VB.NET werden folgende Features nachgerüstet:

- using,
- vorzeichenlose Datentypen,
- Operatoren überladen,
- Compiler Warnungen,
- XML Dokumentation,
- Project Options,
- Edit-and-Continue, Debuggen wie bei VB6,

- Code Editor mit Vorschlägen zur Korrektur bei Fehlern,
- Continue-Anweisung kann bei Schleifen auch verschachtelt eingesetzt werden.

### C# rüstet auch

- Refactoring: Damit wird das Umorganisieren von Quellcode einfacher,
- Contravariance und Covariance für Delegates: allgemeinere Delegates deklarieren und diese einer Anzahl von Klassen zur Verfügung stellen,
- anonyme Delegates: Dadurch wird das Eventhandling vereinfacht,
- Friend Assembly: Verwenden von internen Datentypen von einem als Freund deklarierten Assembly,
- die Standard Codeformatierung durch die IDE ist anpassbar,
- Yield-Schlüsselwort für das Erzeugen von IEnumerable Klassen
- Edit-and-Continue. |||||

[1] Marcel Gnoth, In beiden Welten zu Hause, Unterschiede zwischen C# und Visual Basic .NET, dotnetpro, 10/2004, Seite 28 ff.

[2] Marcel Gnoth, Der Kampf der Sprachen, C# oder Visual Basic .NET?, dotnetpro 10/2004, Seite 44 ff.

[3] Das Mono Projekt: [www.mono-project.com/about/index.html](http://www.mono-project.com/about/index.html)

[4] Microsoft Developer Tools Roadmap: [msdn.microsoft.com/vstudio/productinfo/roadmap.aspx](http://msdn.microsoft.com/vstudio/productinfo/roadmap.aspx)

[5] Dan Appleman, VB.Net Programming Secrets, BASTA, September 2004

[6] Patrick A. Lorenz, Die Sprachreform, Neue Merkmale in C# 2.0 und Visual Basic .NET 2.0, dotnetpro 10/04, Seite 38 ff.

[7] Somasegar's WebLog: [blogs.msdn.com/somasegar/archive/2004/10/15/242853.aspx](http://blogs.msdn.com/somasegar/archive/2004/10/15/242853.aspx)

### Meinung: Wenn Sie mich fragen ...

Letztendlich muss jeder Entwickler und jedes Team die einzelnen Features individuell bewerten. Jeder hat andere Vorlieben. Neben vielen Kleinigkeiten, die schön sind, orientiert man sich bei der Wahl der Sprache aber meist an nur wenigen Merkmalen. Für mich sind Edit-and-Continue, Background Compilation und ein Codeeditor, der Korrekturen bei Fehlern vorschlägt, die entscheidenden Kriterien. Edit-and-Continue wird es jetzt in beiden Sprachen geben. Bleibt Refactoring und Background Compilation. Wer weiß, was sich bis zum Erscheinen von Whidbey noch alles ändern wird. Es bleibt eben eine rein emotionale Entscheidung, die auf wenigen winzigen Unterschieden im Detail beruht.