Microsoft .NET Framework
Interview mit Benjamin Voigt, Microsoft Student Consultant

Philip Iezzi, 7.8.2003

Benjamin Voigt ist seit 2001 aktiver Microsoft Student Consultant an der Universität Zürich. Als ehemaliger Linux-Crack und Anhänger der OpenSource-Gemeinde besitzt er ein äusserst breit gefächertes Wissen. Gewiss kann er nicht als engstirniger Microsoftianer bezeichnet werden.

Momentan beschäftigt er sich mit XML, SOAP und zahlreichen neuen Entwicklungen innerhalb Microsoft's .NET Framework. Er bereitet sich auf einen .NET Workshop im Oktober 2003 vor, für den er auch sehr viel seiner Freizeit aufwendet. Das Meiste macht Benjamin aus Eigeninitiative und Freude an der Technologie selbst und hat weniger mit seiner Anstellung bei Microsoft selbst zu tun. Dies zeichnet ihn zu einem äusserst interessanten Gesprächspartner aus, und ich bin ihm äusserst dankbar, dass er sich Zeit für dieses Interview genommen hat.

Iezzi: *.NET bietet multi-language Support während J2EE auf Java allein beruht. Fördert .NET nicht das „codemässige Chaos" indem es die Entwickler nicht auf eine einheitliche Sprache zwingt? Macht multi-language Support überhaupt Sinn oder wird sich in .NET so oder so C# (sprich: C-sharp) durchsetzen?*

Voigt: Ich denke nicht, dass man da von einem „Chaos" sprechen kann. Bestehende Applikationen werden ins .NET Framework integriert. Ein kompletter Rewrite bestehender Applikationen ist in vielen Unternehmungen gar nicht denkbar. Wer heute mithalten will, muss flexibel sein und dazu fähig sein, mit der aktuellen Technologie mitzuhalten. Dies ist bestimmt nicht möglich mit einem mehrjährigen Redesign der kompletten Applikationsumgebung – ein Projekt das wohl in den meisten Fällen scheitern würde.
Die Idee hinter dem multi-language Support von .NET ist eigentlich genial. Es basiert auf dem Type Marshalling – die Datentypen der verschiedenen Programmiersprachen werden auf eine gemeinsame Ebene gebracht in der Intermediate Language (IL).
Du sprichst von C# als DIE .NET Sprache. Gewiss, C# wird sich durchsetzen und jede Neuentwicklung unter .NET wird wohl auf C# beruhen. Dies impliziert aber keinesfalls, dass multi-language Support „unnötig" oder gar „störend" ist. Die Schwierigkeit einer Programmiersprache liegt ja nicht an deren Syntax. Vielmehr sind es die Klassenlibraries, welche die Mächtigkeit einer Programmiersprache ausmachen. Und genau dort liegt ja auch das Know-How der professionellen Programmierer. Dieses Know-How soll nicht vom einen auf den anderen Tag nichtig gemacht werden, indem sich die Programmierer an eine neue Sprache gewöhnen müssen. Bestehende Klassenlibraries bisheriger Programmiersprachen können in .NET weiterverwendet werden während die Klassenbibliothek von .NET natürlich äusserst mächtig ist. Nochmals auf Deutsch: C# wird sich durchsetzen, die .NET Klassenbibliothek wird genutzt werden, aber ein Umstieg vom einen auf den nächsten Tag ist nicht möglich und ist im .NET Framework auch gar nicht erforderlich. Genau dies zeichnet .NET aus.

Iezzi: *Wie sieht es mit der Performance aus, wenn man die Common Language Runtime (CLR) von .NET mit der Java Virtual Machine (JVM) von Java vergleicht?*

Voigt: Dazu will und kann ich keine klare Antwort geben. Es gibt zahlreiche Performance Tests von beiden Seiten. Microsoft und Sun behaupten jeweils von sich, dass sie die „schnelleren" seien und beweisen dies auch mit ausführlichen Benchmarks. Performance ist ein sehr umstrittenes Thema.

Es gibt bestimmt Bereiche, wo die CLR von .NET schlechter performiert als die JVM. Es gibt jedoch auch Bereiche, wo es genau umgekehrt ist.
Ein entscheidender Vorteil der CLR darf jedoch nicht ausser Betracht gelassen werden: Das Caching von Binärcode ist ziemlich ausgeklügelt. Wir sprechen hier nicht von J-I-T compilation. Die CLR ist dazu konzipiert, nur dann zu kompilieren, wenn es zwingend nötig ist, d.h. also nur, wenn etwas geändert wurde. Ansonsten läuft das Ganze so schnell wie jeder herkömmliche maschinenspezifische Binärcode. In JAVA mangelt es da noch ein bisschen an Optimierungen. Es existiert bereits eine Art Caching, jedoch ist dieses noch nicht ganz ausgereift, denke ich jetzt mal behaupten zu dürfen. Klar ist, dass Performance ein ewiges Battlefield ist, d.h. Java und .NET werden sich gegenseitig immer versuchen aufzuholen.
Interessant ist das Thread-Handling. In JAVA sind Threads eine ziemlich komplizierte Angelegenheit, wobei Threading für den Programmierer ein ziemlich grosser Aufwand ist. In C# ist dies einiges besser gelöst. Es gibt da einen ThreadPool, bei dem der Programmierer beliebig viele Threads anmelden kann und diese dann automatisch verwaltet werden. Ich habe da Tests mit Windows Media Player SDK gemacht und war ziemlich begeistert von ThreadPool.
Vor allem bei Web Services, die ggf. eine gewisse Roundtrip-Zeit beanspruchen, ist es wichtig Threads zu verwenden.

Iezzi: *Wo liegen die Vorteile an C# gegenüber Java?*

Voigt: Wie ich sehe, hast du bereits den Artikel von John Carrol „Ten reasons to choose C#“[1] (ZDNet) gelesen.
C# ist keine komplette „Revolution“ aller bisherigen Programmiersprachen. So darf man C# nicht sehen. Dennoch hat C# ein paar ziemlich interessante Sprachkonstrukte, von denen ich ein paar aufzählen möchte:

- *Attribute*
  Attribute in C# sind eine Art deklarative Tags, die es ermöglichen, im Code zusätzliche Informationen einzubringen. Verwendet werden kann dies z.B. für Conditional Compilation. Attribute sind relativ mächtig und vielseitig einsetzbar.
- *Eventsystem*
  C# besitzt ein gutes Eventsystem. Während das Ganze in JAVA über die action listeners realisiert wird, sind wir bei C# weniger eingeschränkt. Es gibt das so genannte Delegate-Pattern als Zwischenklasse. Dieses unterstützt Multicast Delegates, also mehr als eine Action auf einen Event.
- *Collections (foreach)*
  In JAVA existiert kein foreach. In C# können wir Collections definieren mit expliziter Angabe wie wir z.B. mittels eines foreach darüber loopen.
- *Properties*
  In JAVA müssen Properties mühsam von Hand mit den berüchtigten get/set Methoden implementiert werden. In C# können wir Properties nutzen, was uns eine saubere Trennung des Field-/Methodenspace ermöglicht.
- *Vererbung*
  Die Vererbung geschieht in C# syntaktisch mit dem Operator „:“. Dabei wird nicht unterschieden, ob von einer Klasse oder einem Interface vererbt wird. Dies vereinfacht das Ganze zwar ein wenig, ist jedoch nicht sonderlich intuitiv. In JAVA wird die Vererbungsart explizit mit extends oder implements definiert.

[1] Ten Reasons to Choose C#, by John Carroll
http://zdnet.com.com/2100-1107_2-1015729.html

Iezzi: *Unterstützt .NET Rapid Application Development (RAD)? Wie steht es mit der schnellen Produktivität im Vergleich zu herkömmlichen Scriptsprachen?*

Voigt: Bei RAD stellt sich die Frage, ob wir dies überhaupt wollen. Ich bin RAD gegenüber äusserst kritisch eingestellt. Ich sehe diesen Begriff eher von der negativen Seite.
RAD könnte man wie folgt definieren: Die Programmierer wollen etwas so schnell wie möglich implementieren, wobei sie dem Design keineswegs Beachtung schenken. Sind wir ehrlich – Wird eine Scriptsprache zur Hilfe genommen für ein grösseres Projekt, einzig um die Einarbeitungszeit gering zu halten, geht es nur um das „get the job done!" ohne auch nur einen Schritt weiterzudenken.
Gewiss, es gibt mächtige Scriptsprachen, die einfach zu erlernen sind. Diese sind jedoch für viele grössere Projekte unbrauchbar, da man dann doch wieder von externen Komponenten abhängig ist. Dies endet oft in einem Code-Salat und das Design ist von Grund auf kurzlebig.
Ich behaupte, dass man RAD nie sauber machen kann. RAD dient bestenfalls für einen Wegwerf-Prototyp, nicht mehr als das!
Dennoch wird RAD natürlich klar „gepushed" durch .NET. Das ist genau, was die Leute wollen. Ich denke aber, .NET kann man deswegen aber noch lange nicht mit der Umgebung einer Scriptsprache vergleichen. .NET ist da doch eine Stufe höher, bedient sich ausgereiften umfangreichen Klassenlibraries und zwingt einen auf ein gutes Design.
VB.NET und die VB Integration ist jedoch speziell auf RAD ausgelegt. Der relativ hohe Verbreitungsgrad von VB erklärt sich teilweise durch die besonderen RAD Merkmale, welche in Zukunft gewiss auch noch mehr ausgebaut werden.

Iezzi: *Wie steht es mit der Einarbeitungszeit in C# im Vergleich zu Java? Wie lange braucht ein MA, der eine Ahnung von Objektorientierter Programmierung hat?*

Voigt: Wie dir vielleicht bereits aufgefallen ist, ist die Syntax von C# sehr nahe an der von JAVA. Nicht schlecht geschaut habe ich, als sich mein erstes C# Programm sogar unter der JVM (Java Virtual Machine) kompilieren liess!
Bezüglich Syntax gibt's also keinen grossen Umschulungsaufwand für jemanden, der JAVA beherrscht. Bezüglich Klassenbibliothek natürlich schon. Wie bereits vorhin erwähnt, ist es genau dort, wo das Know-How des Programmierers liegt.
Theoretisch reicht ein 2-3 Tage Intensivkurs bereits für einen JAVA-Entwickler, um sich mit C# vertraut zu machen und bereits produktiv zu arbeiten. Jemand, der von Objektorientierten Programmiersprachen noch keine allzu grosse Ahnung hat, bräuchte mindestens 1 Woche intensiv, kann dann aber noch kaum produktiv arbeiten. Eben, die Klassenlibraries lernt man erst nach mehreren Monaten richtig kennen.
In .NET hat XML einen der grössten Stellenwerte. Ein bestimmtes Grundwissen sollte also bereits vorhanden sein und der Programmierer sollte die saubere Trennung von Field-/Methodenspace beherrschen.
Seit Kurzem gibt es sogar die Java SWING Implementation für die .NET Sprache J#, um den Umstieg Java Entwicklern noch leichter zu machen. J# ist jedoch aus meiner Sicht klar als eine Zwischenlösung auf dem Weg zu C# positioniert.

Iezzi: *In .NET, im Speziellen in ASP.NET, interessiert mich die Trennung GUI-Design und Programmierung. Könntest du mir veranschaulichen, wie dies gelöst wurde und wie klar die Trennung effektiv ist? Inwiefern wird dies von Visual Studio .NET unterstützt?*
Kann ich mir das Ganze wie ein herkömmliches Template-System vorstellen oder ist es eine Spur intelligenter?

Voigt: Mach dir da keine Sorge! Dies ist genau eine der Hauptstrategien von .NET. Ganz klar richtet sich .NET auf die Trennung von GUI-Design und Programmierung aus. Ich denke auch, dass dies ziemlich optimal gelöst wurde.
Sprechen wir von einer herkömmlichen Applikation mit GUI, nennt sich das Ganze „Visual Inheritance". Der eine Programmierer oder Designer kann sein Objekt für bestimmte Bereiche freigeben, wo dann der andere seinen Teil hinzuliefert. Der Designer kann also z.B. an seiner eigenen Design-Komponente arbeiten und sieht den Gesamtkontext, ohne in die Komponenten eines anderen Designers/Programmierers hineinzufunken. Am besten zeige ich dir dies gleich mal in Visual Studio .NET. [Demonstration Visual Inheritance]
Dich interessiert nun v.a. ASP.NET. Dort wird das Ganze „User Controls" genannt. Diese user controls basieren auf demselben Prinzip. [Demonstration User Controls]
Diese klare Trennung in .NET ist wohl einiges ausgereifter als das Pendant in JSP (Java Server Pages). In JSP wird dies durch XML/XSLT, resp. Cocoon, realisiert.

Iezzi: *Was sind die Neuigkeiten von COM+? Was ist DCOM?*

Voigt: DCOM ist dasselbe wie COM+. Die einen nennen es DCOM, was für distributed COM steht, den anderen gefällt der Name COM+. Die grosse Neuerung ist, dass COM+ nun verteilte Objekte unterstützt. Man kann dies mit Corba vergleichen.

Iezzi: *Wie sieht es mit der Plattformunabhängigkeit aus in Zukunft? Wird sich .NET auch in der OpenSource-Welt durchsetzen? Das Mono Projekt[2] ist ja anscheinend bereits ziemlich weit fortgeschritten. Ist die CLR (Common Language Runtime) offen gelegt? Und wie sieht es mit der Loslösung von IIS (Internet Information Server) als Webserver aus?*

Voigt: Momentan ist dies ein ziemlich interessantes Thema. Soeben, am 4. August, wurde Ximian von Novell aufgekauft[3]. Ximian leitet das Mono Projekt und GNOME. Mono erlaubt, .NET Applikationen auf Linux, UNIX, FreeBSD, Windows und anderen Plattformen laufen zu lassen. Es gibt sogar bereits ein Apache Modul für ASP.NET. Das Ganze ist in einem ziemlich ausgereiften Stadium.
Wie reagiert Microsoft nun auf die ganze Entwicklung? Gewiss, es gibt da bestimmte Patentverletzungen, dies brauchen wir nicht in Frage zu stellen. Aber hat Microsoft wirklich den Anreiz, Ximian resp. nun Novell zu verklagen? Ich würde mal sagen, nein. Novell spielt da einen entscheidenden Faktor und stärkt somit das Projekt. Microsoft wird nicht Novell verklagen.
Die CLR ist übrigens offen gelegt. .NET beruht auf der so genannten CLI (Common Language Infrastructure). Dies ist eine standardisierte Spezifikation die das Typensystem, IL und Komponentenmodell beinhaltet. Standardisiert ist diese nach ECMA[4] und ISO. Microsoft's Implementation der CLI hat den Codenamen „Rotor". Diese Implementation wird als SSCLI (Shared Source CLI) bezeichnet. Mono ist eine weitere Implementation dieser SSCLI.
Das Ganze ist auch relativ neu. Die Standardisierung ging am 2. April 2003 über die Bühne.

---

[2] http://www.go-mono.com/
[3] Press release: http://www.ximian.com/about_us/press_center/press_releases/index.html?pr=novell
[4] http://www.ecma-international.org/
CLI, Standard ECMA-335, http://www.ecma-international.org/publications/standards/ecma-335.htm

Wie du siehst ist also .NET tatsächlich offen gelegt.
Weiter fragst du mich bezüglich Unabhängigkeit von IIS. Gewiss, es gibt bereits ein mod_asp für Apache. Aber wer wird schon auf einem Windows-Rechner .NET über Apache fahren lassen? Eine komplette OpenSource Implementation kann ich mir hingegen durchaus vorstellen. Aber Microsoft wird bestimmt nicht Apache aktiv unterstützen, dafür fehlt nun mal der Anreiz und würde IIS selbst ins schlechte Licht stellen. Ja, mit .NET bist du also relativ festgebunden an IIS und das wird sich nicht so schnell ändern.

Iezzi: *Wie steht es mit dem DB-Zugriff per ADO.NET im Vergleich zu JDBC?*

Voigt: ADO.NET kannst du dir als oberste Abstraktionsstufe vorstellen. Der Datenzugriff wird über ein Dataset abgewickelt, was sich bei JDBC Ressource Set nennt. ADO kann darauf entweder per Stored Procedure, SQL oder OLE auf die Datenbank zugreifen. Mit OLE hätten wir natürlich nochmals eine Indirektionsstufe mehr. Die effektive Implementation hängt sehr stark von dem Einsatzgebiet ab.
JDBC ist ADO konzeptionell sehr ähnlich. ADO.NET arbeitet primär nach dem Prinzip der „Data Adapters“, welche für eine Datenquelle instanziert und konfiguriert werden. Man kann auch eigene „Data Adapters“ implementieren basierend auf der ADO.NET Infrastruktur. Es gibt dabei drei wesentliche Zugriffsmodi: Direktes Kommando (SQL oder der Datenquelle inhärente Query-Sprache), Transaktionen oder Data Adapter.
Der höchste Grad an Abstraktion bietet der Zugriff via Dataset.

Iezzi: *Kennst du Visual Source Safe? Welche Funktionalität bietet es im Vergleich zu anderen externen Produkten? Ich hab da einiges Negatives über VSS gehört. Konfigurationsmanagement ist bei jedem grösseren Projekt ziemlich entscheidend, denke ich.*

Voigt: Zugegeben, ich kenne Visual Source Safe zuwenig. Es gibt einige üble Gerüchte über dieses Tool, dennoch ist es eines der fähigeren.
Willst du ein externes Tool verwenden unter Visual Studio .NET, bietet dir VS .NET dazu eine ziemlich gute Schnittstelle an. Selbst wenn es kein Plug-In oder ähnliches geben sollte, ist eine solche Einbindung innerhalb von Stunden geschrieben. Die API von VS .NET ist nun mal ziemlich komfortabel. Deshalb gibt es auch ca. 175 Unternehmen, die Plug-In’s für VS.NET entwickelt haben, neben zahlreichen nicht kommerziellen Erweiterungen.

Iezzi: *Vielen Dank für dieses aufschlussreiche Interview. Es haben sich doch einige Dinge geklärt, bei denen ich noch unsicher war. Ich komme gerne später wieder mal auf dich zurück.*
Viel Spass an deinem .NET Workshop im Oktober.

Voigt: You’re welcome!