

Teil A

Schnell zurechtfinden

In diesem Teil:

Einführung	3
Das ist neu in der IDE von Visual Studio 2008	13
Das ist neu im Compiler von Visual Basic 2008	31
Klassen und Objekte	47
Arrays und Auflistungen	79
Generics (Generika) und generische Auflistungen	109

Kapitel 1

Einführung

In diesem Kapitel:

Die Geschichte von .NET	4
Wissenswertes zur Installation von Visual Studio 2008	6
Die weiteren Kapitel in diesem Teil	10

Die Geschichte von .NET

Lassen Sie uns eine kurze Bestandsaufnahme machen: Wir hatten die .NET Frameworks 1.0 und 1.1, das waren aus Entwicklersicht die Anfänge von Microsofts .NET-Initiative. Für erste Versionen liefen sie erstaunlich stabil, auch wenn sie natürlich schon zum damaligen Zeitpunkt noch merklich ausbaufähig waren. In erster Linie waren sie für uns Visual Basic-Programmierer aber auf jeden Fall mal ein Schock, denn sie führten in der Version 8.0 das erste Mal echtes OOP und damit mehr als nur objektbasierende Programmierung ein – für viele von uns der eigentliche Grund, zunächst bei VB6 zu bleiben, und das mit der Schutzbehauptung »Wir warten ja mindestens auf die reifere Version 2.0«.

Dann kam die Version 2.0, und wow! Die brachte uns Generics und zu C# kompatible Operatorenüberladung, einen gut optimierten Visual Basic-Compiler, der auch geschwindigkeitsmäßig erwachsen und dem C#-Compiler ebenbürtig geworden war, den My-Namespace mit einer ganzen Bibliothek an simpelst aufrufbaren Miniprogrammen und -routinen für (fast) jeden Zweck. Und auch einen gründlich überarbeiteten Editor samt vieler neuer cooler Funktionen, aber der litt anfangs noch gerade bei Mehrprozessorsystemen an Schluckauf in Form von Racing Conditions zwischen ihm und dem Backgroundcompiler, und zog deswegen oft das ganze System ins Nirwana, was aber dann mit SP1 und dem kurze Zeit später erscheinendem SP1 fürs SP1 (genannt: *Update für Vista*) weitestgehend behoben wurde. Und es kamen neu die so genannten Nullable-Datentypen – allerdings in ihrer finalen Version vergleichsweise spät in der Produktentwicklung, um nicht zu sagen: zu spät, denn sie waren neben anderen Dingen Wegbereiter dafür, dass die Entwicklung von SQL Server 2005 und Visual Studio 2005 begann, quasi »Out of Sync« zu laufen. Der Hintergrund:

Nullable-Datentypen versetzen auf der Basis von Generics normale Datentypen in die Lage, den Wert Null (also nicht 0, sondern Nothing in Visual Basic) anzunehmen. Boxt man nun einen Null-repräsentierenden Datentyp in ein Objekt, sollte es konsequenterweise auch Nothing widerspiegeln, das wäre logisch, und so sieht es auch die finale Implementierung der .NET 2.0-CLR vor – nicht aber die Version, die noch kurz vor der RTM der 2.0 CLR zum Testen unterwegs war: Dort konnte – gemäß Standardimplementierung – natürlich ein definierter und in ein Objekt geboxter Wertetyp *nicht* dazu führen, dass ein Objekt Null (Nothing) wurde. Die gewünschte finale Verhaltensweise konnte man deswegen nur mit einem heftigen Eingriff in der CLR selbst erreichen, und diesen Eingriff kurz vor Schluss vorzunehmen, war nicht nur gewagt, sondern hatte natürlich weitere Konsequenzen für alle darauf basierenden Entwicklungen – wie beispielsweise den SQL Server 2005, der es ja zum ersten Mal ermöglichte, dass .NET-Assemblies in seinem Kontext zur Ausführung kommen konnten, sodass er von dieser Änderung ja nicht gerade nur am Rande betroffen war. Nullables sind ja schließlich sehr essentiell, wenn es um das Speichern von Datentypen geht, die auch in der Datenbank *Null* werden. Schlussendlich ist alles in allem die Version 2.0 des .NET Frameworks eine wirklich runde, sauber laufende und bis Windows 2000 als Plattform herunterreichende Lösung, mit der es – meiner Meinung nach jedenfalls – unglaublich viel Spaß macht, zu entwickeln.

.NET Framework 3.0

Und dann kam Windows Vista mit dem .NET Framework 3.0. Und das mit .NET Framework 3.0 ist, jedenfalls was die Versionsgebung angeht, so eine Sache: Denn während 1.1 und 2.0 die jeweils vorherigen Versionen CLR-technisch verbesserten, Fehler ausmerzten und teilweise leider auch mit neuen Konzepten für so

genannte Breaking Changes¹ sorgten, ist das .NET Framework 3.0, das übrigens das erste .NET Framework war, das werksseitig in einem Betriebssystem (nämlich Vista) vorhanden war, nur eine *Erweiterung*. Die Kernkomponenten wurden in keinsten Weise angefasst, und so gab es auch keine Breaking Changes. Stattdessen gibt es ein paar zusätzliche neue Assemblies, die Bestandteile des .NET Frameworks 3.0 wurden, aber die haben es wirklich in sich:

- Die **Windows Presentation Foundation** (WPF), zu ihrer Entwicklungszeit eher bekannt unter dem Codenamen **Avalon**, stellt Klassenbibliotheken für ein neues Benutzerschnittstellenkonzept dar, deren Struktur und Aufbau prinzipiell durch eine Abwandlung von XML namens XAML² beschrieben werden. Auf diese Weise können – ähnlich wie beim Code-Behind-Verfahren von ASP.NET – Design und Programmlogik erstmals auch in Fat Clients voneinander getrennt werden. Obendrein gibt es zusätzliche API-Funktionen für eine leistungsfähige Animations- und 3D-Engine, die sich direkt der 3D-Computergrafikhardware der Grafikkarten sowie der Direct3D-Technologien bedient, in die sich die Benutzeroberflächen-Klassenbibliothek nahtlos eingliedert. Die Windows Presentation Foundation gibt's ab Kapitel 11 – auch wenn sie »nur« zum .NET Framework 3.0 gehört, haben wir ihr einen ganzen Buchteil (Teil C) gewidmet, da Visual Studio 2008 mit einem eigenen Designer erstmals WPF-Projekte unterstützt.
- Die **Windows Communication Foundation** (WCF), zu ihrer Entwicklungszeit eher bekannt unter dem Codenamen **Indigo**; stellt ein dienstorientiertes Nachrichtensystem dar, mit dem die Entwicklung von lokal oder entfernt miteinander kommunizierenden Komponenten möglich wird. In Ergänzung zu den schon in vorherigen .NET Framework-Versionen bekannten Web Services können auch DTS-kompatible und damit transaktionsfähige Komponenten oder Peer-To-Peer-fähige Szenarien wie Bilddateientausch-Komponenten entwickelt werden.
- Die **Windows Workflow Foundation** (WF) ermöglicht es, mithilfe von Workflows Aufgabenautomatisierung und integrierte Transaktionen zu erstellen.
- **Windows CardSpace**, während der Entwicklungszeit eher bekannt unter dem Codenamen **InfoCard**, stellt eine Softwarekomponente dar, um gesicherte digitale Identitäten beispielsweise für das Anmelden bei Webdiensten speichern zu können.

.NET Framework 3.5

Das .NET Framework 3.5 benutzt prinzipiell ebenfalls die Version 2.0 der CLR, ist aber, anders als die 3.0 Version eben nicht nur ein purer Aufsatz bzw. eine Bibliotheksergänzung, sondern installiert das Service Pack 1 des .NET Framework 2.0, das den Wegbereiter für neue Techniken wie LINQ darstellt. Microsoft

¹ Breaking Changes in diesem Zusammenhang sind Änderungen am Framework, die unter bestimmten Umständen dafür sorgen, dass bereits für eine bestimmte Version konzipierte Programme bei einer Versionsänderung nicht mehr laufen, da sich grundlegende Verhaltensweisen geändert haben. Das hat aber nur auf den ersten Blick wirklich schlimme Auswirkungen, denn das .NET Framework-Konzept sieht es vor, dass mehrere Versionen des Frameworks parallel auf einem Rechner existieren. Im schlimmsten Fall musste also eine ältere Version des Frameworks nachinstalliert werden, um Breaking Changes durch einen Versionswechsel zu umgehen.

² Etwa wie die bayrische »Semmel« mit einem »G« davor ausgesprochen, also etwa »Gsämmel«

verspricht, dass das .NET Framework 2.0 *mit* SP1 bei der Erweiterung so behutsam vorgeht, dass keine Breaking Changes für bereits vorhandene auf .NET Framework 2.0 abzielende Software zu erwarten sind.

Die exakten Spracherweiterungen, die das .NET Framework 3.5 für Visual Basic 2008 ermöglicht, sind zentraler Bestandteil des ersten Teils dieses Buches, und eine Schnellübersicht finden Sie in und ab Kapitel 3.

Welche Softwarevoraussetzungen benötigen Sie?

Um die Beispiele in diesem Buch nachzuvollziehen, benötigen Sie eine aktuelle Version von Visual Studio 2008 in der Professional Edition. Und es wird Sie freuen, zu hören, dass auf der beiliegenden DVD eine 90-Tage-Version genau dieser Version beiliegt. »90-Tage-Testversion« bedeutet, dass Sie diese Version von Visual Studio immerhin ein viertel Jahr kostenlos und *ohne Einschränkungen* verwenden können. Nur: nach Ablauf dieser Probefrist müssen Sie sich entscheiden, wie es weitergeht. Empfehlen kann ich den Lesern unter Ihnen, die auch nach Ablauf dieser Probefrist weiterhin professionell mit Visual Studio 2008 arbeiten möchten, auf jeden Fall den Kauf der Professional Edition zumindest aber der Standardversion von Visual Studio 2008 in Erwägung zu ziehen. Haben Sie sich für die Professional Version entschieden, brauchen Sie noch nicht einmal die Installation zu wiederholen: Ein neuer Installations-Key, den Sie beim wiederholten Ausführen des Setups eingeben können, ist dann ausreichend, um die bereits installierte Version freizuschalten.

Die kostenlose Alternative: die Express Editions

Es gibt auch eine preiswertere Alternative, die wahrscheinlich die bessere für die Hobbyisten oder die Gelegenheitsentwickler unter Ihnen darstellt. Microsoft bietet nämlich mit den so genannten Express Editions abgespeckte Versionen von Visual Studio 2008 an, die jeweils nur eine Programmiersprache beinhalten. So gibt es beispielsweise die C# Express Edition, die C++ Express Edition und natürlich auch die Visual Basic Express Edition.

Und, wie schon gesagt, diese Version können Sie kostenlos von der Microsoft Internetseite herunterladen. Für viele Beispiele dieses Buches, gerade wenn es um LINQ to SQL geht (siehe Kapitel 9), ist diese Version allerdings nicht ausreichend, um alle Funktionen zu nutzen.

Wissenswertes zur Installation von Visual Studio 2008

Eigentlich müssen Sie nichts Außergewöhnliches beachten, wenn Sie Visual Studio auf Ihrem Computer zum Laufen bringen wollen. Die folgende Tabelle zeigt Ihnen die ideale Hardwarevoraussetzung, die ein reibungsloses Arbeiten mit Visual Studio 2008 ermöglicht.

WICHTIG Orientieren Sie sich bei dieser Liste lieber nicht an der angegebenen Mindestkonfiguration nach Microsoft-Spezifikation, die eher humoristischen Anforderungen genügen, sondern besser an der »Wohlfühl-Rundum-Sorglos«-Konfiguration, mit der das Arbeiten Spaß macht, und die nicht wegen langer Wartezeiten zu ständiger Nervosität, Hypertonie, Tachykardie oder Schlaflosigkeit auf Grund zu hohen Kaffeekonsums führt. Ein paar Euro in ein wenig Hardware investiert, können Ihre Turn-Around-Zeiten³ beim Entwickeln drastisch verbessern!

Denken Sie auch daran, dass es sich bei vielen der heutigen Prozessoren, die als Standardkonfiguration von Computern verbaut werden, automatisch um so genannte Dual-Core-Prozessoren (bzw. Quad-Core-Prozessoren) handelt. Solche Prozessoren vereinen zwei (bei Quad-Core-Prozessoren sogar vier) Prozessorkerne unter einem Dach. Ganz einfach ausgedrückt bedeutet das: befindet sich ein Dual-Core-Prozessor in Ihrem Computer, dann verfügt Ihr Computer quasi über *zwei* völlig unabhängig voneinander arbeitende Prozessoren. Und bei Quad-Core-Prozessoren sind das sogar dann vier an der Zahl. Nun gilt es für die Ausstattung von Entwicklungsrechnern folgendes in Erwägung zu ziehen: Bei der Entwicklung von so genannten Multi-Threading-Anwendungen – das sind Anwendungen, bei denen verschiedene Programmteile quasi gleichzeitig ablaufen können – gibt es auf Single-Core-Prozessoren unter Umständen völlig andere Verhaltensweisen dieser parallel laufenden Programmteile als auf Dual- oder Quad-Core-Prozessoren, auf denen Programme ja tatsächlich gleichzeitig laufen können, da sie eben von zwei unabhängigen Prozessorkernen ausgeführt werden. Dem gegenüber stehen Single-Core-Computer, auf denen Multi-Threading-Anwendungen nur scheinbar gleichzeitig laufen – hier wird zwei scheinbar gleichzeitig laufenden Multi-Threading-Programmteilen Prozessorzeit im ständigen Wechsel zur Verfügung gestellt. Sie können auf einem Single-Core-Prozessor also unter Umständen gar nicht alle unterschiedlichen Aspekte einer Multi-Threading-Anwendung erfassen, nachstellen und testen. Deswegen ist es auf jeden Fall empfehlenswert, dass Ihr Entwicklungsrechner ebenfalls über mindestens zwei unabhängig arbeitende Kerne verfügt, damit die Anwendungen, die Sie entwickeln, auch später für Multi-Core-Prozessoren gerüstet sind.

Komponente	Mindestkonfiguration laut Microsoft	Ideal-Konfiguration (erfahrungsgemäß)
Prozessor	1,4 GHz	Dual Core Prozessor mit ausreichend großem Second Level Cache, zum Beispiel Intel Core 2 Duo E6600, Core 2 Quad E6600 oder AMD Athlon 64 X2 6000+. ⁴
Ram	256 MByte (Windows XP SP2) bzw. 512 MByte (Windows Vista)	1 GByte unter Windows XP SP2 bzw. 2 GByte unter Windows Vista.
Festplattengröße (incl. MSDN-Hilfe) – also freier Speicherplatz. HINWEIS: Denken Sie daran, dass gerade bei Notebooks noch ausreichend freier Speicher für Systemdateien, insbesondere für den Suspend-Modus bis zu 6 GBytes freier Speicher verbleiben müssen.	5,4 GByte	5,4 GByte
DVD-Laufwerk	Wird benötigt	Wird benötigt ▶

³ Die Zeit, die man braucht, um beispielsweise nach dem Bemerkten eines Fehlers und nach der entsprechenden Änderung im Code, das Programm neu zu kompilieren und wieder in den »Laufend«-Modus (*Run time mode*) zu versetzen.

⁴ Wir wollen keinen der beiden großen Prozessorhersteller benachteiligen, aber zum Zeitpunkt, zu dem diese Zeilen entstehen, ist von der Firma AMD leider noch kein Quad-Core-Prozessor *wirklich* verfügbar.

Komponente	Mindestkonfiguration laut Microsoft	Ideal-Konfiguration (erfahrungsgemäß)
Grafikkarte	800 x 600 256 Farben	1280 x 1024 True Color; oder mindestens Dual-Monitor-Support mit zwei Mal 1024 x 768 bei True Color. Für den Einsatz unter Windows Vista achten Sie bitte auf eine DirectX 9.0-fähige Grafikkarte mit mindestens 128 MB Grafikkartenspeicher! ⁵

Tabelle 1.1 Die Minimal- und Idealvoraussetzungen an die Hardware für ein reibungsloses Arbeiten mit Visual Studio 2008

Lauffähigkeit von Visual Studio und den erstellten Kompilaten unter den verschiedenen Betriebssystemen

Mit Visual Studio 2008 erstellte Programme laufen *nicht* auf Windows 95 und Windows NT, und unter Windows 98 oder Windows Millennium steht Ihnen nicht der komplette Funktionsumfang des .NET Framework 2.0 zur Verfügung (.NET Framework 2.0-Programme laufen aber grundsätzlich unter diesen beiden älteren Betriebssystemen entgegen vieler Meinungen schon). Sie benötigen allerdings *mindestens* Windows XP Home SP2, Windows XP Professional SP2, die XP-Media-Center-Edition SP2, Windows 2003 Server (natürlich auch R2), Windows 2008 Server oder eine der Windows Vista-Versionen (aber nicht die Starter-Edition! – Home Basic funktioniert hingegen), um die Entwicklungsumgebung (entweder die kostenlose Express Edition oder eine der Visual Studio-Vollversionen) installieren zu können. Die Installation der Entwicklungsumgebung unter Windows 2000 ist nicht mehr möglich.

.NET Framework Version Targeting (Versionsbestimmung des .NET Frameworks für Kompilate)

Wichtig ist es in diesem Zusammenhang zu wissen, dass Sie mit Visual Studio 2008 bestimmen können, mit welcher .NET Framework-Version die von Ihnen entwickelte Anwendung funktioniert. Sie können also bestimmen, dass das Programm, das Sie gerade entwickeln, das .NET Framework 2.0, das .NET Framework 3.0 oder eben das .NET Framework 3.5 verwenden soll (auf Neudeutsch spricht man dabei vom sogenannten ».NET Framework Version Targeting«).

HINWEIS Dabei ist wiederum wichtig zu wissen, dass Programme, die noch unter Windows 2000 SP4 laufen sollen, höchstens auf das .NET Framework 2.0 abzielen können. Die Installation des Microsoft .NET Frameworks 3.0 oder 3.5 ist unter Windows 2000 nicht mehr möglich (und das ist der eigentliche Grund, weswegen Sie für dieses Betriebssystem keine Programme mehr schreiben können, die auf diese .NET Framework-Versionen abzielen).

Unter Windows XP hingegen lassen sich beide neuen .NET Framework-Versionen installieren. Voraussetzung dafür ist allerdings, dass das Service Pack 2 für Windows XP zuvor eingerichtet wurde.

Und jetzt noch eine letzte Anmerkung in diesem Zusammenhang: Windows Vista sowie Windows Server 2008 haben sowohl das .NET Framework 2.0 als auch das .NET Framework 3.0 standardmäßig an Bord. Das .NET Framework in der Version 3.5 muss hingegen auch auf diesen Betriebssystemen individuell installiert werden.

⁵ Kein Budget für einen zweiten Bildschirm, aber Sie haben einen Laptop? Nutzen Sie Ihren Laptop als Zweitbildschirm. Mehr dazu gibt es unter dem IntelliLink A0103.

Parallelinstallation verschiedener Visual Studio-Versionen

Seit Version 6 können alle Visual Studio-Versionen nebeneinander installiert werden. Das gilt allerdings nur für Windows XP SP2 als Betriebssystemplattform, denn sowohl Visual Studio 2002 als auch Visual Studio 2003 lassen sich leider nicht unter Vista zum Laufen bringen. Visual Studio 6 läuft hingegen mit Einschränkungen und auch die mit Visual Basic 6.0 entwickelten Anwendungen sind auf Vista lauffähig.

HINWEIS Die Wahrscheinlichkeit ist vergleichsweise groß, dass Visual Basic 6.0 und Visual Basic 6.0-Anwendungen auf zukünftigen Betriebssystemplattformen nicht mehr lauffähig sein werden! Rechtzeitiges Portieren der Anwendungen von VB6 auf entsprechende .NET-Pendants sollten Sie deswegen nicht auf die lange Bank schieben.

Wieso fehlen noch Features? – Visual Studio 2008 Service Pack 1

Visual Studio 2008 steht noch nicht in den Regalen und schon ist die Rede vom ersten Service Pack zu Visual Studio 2008 – darf das sein?

Es muss. Sollten Sie zu Beginn dieses Kapitels die »Geschichte von .NET« gelesen haben, dann haben Sie vielleicht auch mit Interesse die kleine Anekdote zur *Nullable-Problematik* und dem *Out Of Sync*-Laufen der SQL Server- und der Visual Studio 2005-Produktentwicklung verfolgt. Solche Dinge können beim Entwickeln natürlich passieren, denn: Wenn ein Haupt-Feature eines großen Programms von einer Komponente abhängt, die selbst gerade noch in der Entwicklung ist, sind Zeitprobleme buchstäblich vorprogrammiert. So war es bei SQL Server 2005 und dem .NET Framework 2.0, und so ist es zurzeit (Februar 2008): Momentan gibt es nämlich Probleme beim Verwenden von Visual Studio 2008 mit SQL Server 2008 aus ganz ähnlichen Gründen. Und die können erst mit dem Service Pack 1 von Visual Studio 2008 final behoben werden. Die genauen Probleme finden Sie im Abschnitt »Visual Studio 2008 und SQL Server 2005 bzw. SQL Server 2008« beschrieben.

LINQ to Entities erst mit SP1

Und dann fehlt auch noch ein großes Feature in der aktuellen Version von Visual Studio 2008: Das sogenannte Entity-Framework und damit *LINQ to Entities*. Dazu folgender Hintergrund:

LINQ dient grundsätzlich dazu, aus einer fast beliebigen Datensammlung mit einfachen, in die Sprache Basic integrierten Befehlen, Daten auszuwählen. Bisher, also ohne Service Pack 1, können Sie LINQ dazu verwenden, um ...

- ... Arrays und Auflistungen zu selektieren – das nennt sich dann *LINQ to Objects*. Praktisch jede Auflistung, die die Schnittstellen `IEnumerable` und `IEnumerable(Of Type)` implementieren, können damit selektiert werden.
- ... XML-Dokumente zu selektieren – das nennt sich dann *LINQ to Xml*. *LINQ to Xml* ist prinzipiell ein Programmiermodell für im Speicher befindliche XML-Dokumente, die mit den eingebauten Abfragebefehlen selektiert werden können.
- ... DataSets zu selektieren, deren Daten bereits von einem Datenprovider in ein DataSet geladen wurden – das nennt sich dann *LINQ to DataSets*. Bitte verwechseln Sie das nicht mit *LINQ to SQL* oder *LINQ to Entities*, bei denen der Umweg über ein DataSet eben *nicht* mehr erforderlich ist, und Daten aus einer Datenbank oder einer anderen Datenquelle direkt heraus selektiert werden, bevor sie in ein Businessobjekt – also in den Arbeitsspeicher Ihres Rechners gelangen.

- ... direkt die Daten einer Microsoft SQL-Datenbank abzufragen. Das nennt sich dann *LINQ to SQL*. Hierbei werden die LINQ-Befehle, die in die Visual Basic-Sprache integriert sind, direkt in SQL-Abfragebefehle konvertiert, die dann die gewünschten Objekte in sogenannten Businessobjekten gemaped zurückliefert.
- ... direkt die Daten einer beliebigen Datenquelle abzufragen. Und das nennt sich dann *LINQ to Entities*. Das Problem: Das Entity-Framework, das das erlauben würde, ist schlicht und ergreifend noch nicht fertig, und wird erst mit dem Service Pack 1 verfügbar sein.

Visual Studio 2008 und SQL Server 2005 bzw. SQL Server 2008

Während diese Zeilen entstehen, ist die englische Version von Visual Studio 2008 bereits »RTMt« (sprich: *er-temmt*), was die Abkürzung von »Release to Manufacturing« ist und soviel wie »Für die Produktion freigegeben« bedeutet. Das gilt aber leider nur für Visual Studio 2008 und nicht für SQL Server 2008 – der wird aller Voraussicht nach später im Jahr erscheinen, wahrscheinlich knapp nach dem SP1 für Visual Studio 2008.

Bis dahin gibt es für den SQL Server 2008 so genannte Community Technical Previews, CTPs (etwa: »Technische, öffentliche Vorschauversionen«), mit denen Sie sich schon im Vorfeld einen Eindruck der neuen Features verschaffen können. Das Problem dabei ist: Aufgrund von Framework Versions-Inkompatibilitäten können Sie Verbindungen nämlich nur aus bestimmten Release-Ständen von Visual Studio 2008 heraus zu bestimmten Versionen von SQL Server 2008 herstellen. Je nach Release-Stand können sich Ihre entwickelten Programme mit dem SQL Server 2008 verbinden; es gibt dann aber keine Designer-Unterstützung (was gerade für das Erlernen von *LINQ to SQL* etwas kontraproduktiv ist) – bei einigen Release-Kombinationen ist sogar gar keine Zusammenarbeit möglich.

Visual Studio 2008 funktioniert hingegen mit SQL Server 2005 problemlos, sowohl was Designerunterstützung als auch Ihre entwickelten Anwendungen angeht. In diesem Buch lassen wir uns deswegen gar nicht erst auf irgendwelche Beta-Experimente ein: Alle Beispiele und Beispieldatenbanken aus den entsprechenden *LINQ to SQL*-Beispielen arbeiten grundsätzlich mit der Kombination Visual Studio 2008/SQL Server 2005.

Die weiteren Kapitel in diesem Teil

Möglicherweise fallen Ihnen beim Durchlesen des Inhaltsverzeichnisses oder beim Durchblättern dieses Teils drei Kapitel ins Auge, die Themen behandeln, welche Ihnen nicht gerade brandneu erscheinen werden (was auch zutrifft):

- Klassen (Kapitel 4) hat es sicherlich schon in der ersten .NET-Version gegeben.
- Das Gleiche gilt für Auflistungen (Collections, Kapitel 5).
- Generics – oder Generika, wie es auf deutsch heißt – (ab Kapitel 6) gibt es dann wiederum zwar erst ab der 2005er Version, aber sicherlich kann und will ich Ihnen dieses Thema nicht als neu verkaufen.

Leider sind das die drei großen Themenkomplexe, die Sie beherrschen sollten, um alle fachlichen Voraussetzungen für das Thema LINQ parat zu haben, das im zweiten Teil (B) dieses Buches behandelt wird. Ohne

Klassen, keine Auflistungen. Ohne Auflistungen im Allgemeinen gibt es natürlich auch keine *generischen* Auflistungen, mit denen Ihnen aber ebenfalls die Grundvoraussetzungen für LINQ, das wiederum zum großen Teil auf generischen Auflistungen basiert, fehlen würden.

Aus diesem Grund haben wir (Verlag und Autor) uns nach einigem Hin und Her dazu entschlossen, einige Seiten für die LINQ-Grundlagen zu ergänzen, denn noch heute ist es so, dass gerade Entwicklerteams mit einer langjährigen Visual Basic 6.0-Historie Visual Basic .NET vielfach rein prozedural einsetzen. Und das ist auch nicht notwendigerweise falsch – so antwortete ein Geschäftspartner eines guten Freundes von mir auf die Frage, wie er es geschafft habe, mit seinem Team ein komplettes ERP-System innerhalb nur eines Jahres realisieren zu können: »Wir haben konsequent auf objektorientiertes Programmieren verzichtet!«. Nun gut – hätte es LINQ damals schon gegeben, vielleicht hätten sie es dann in nur 10 Monaten geschafft, aber darum geht es ja gerade gar nicht.

Tatsache ist: Um LINQ richtig verstehen und anwenden zu können, sollten Sie diese drei Themenkomplexe beherrschen. Das Thema Klassen müssen Sie dabei nicht komplett verinnerlicht haben – in diesem Buch wird es deswegen gerade so ausführlich behandelt, wie es zum späteren Verständnis von LINQ notwendig ist.

Wer mehr in Sachen Klassen und OOP lernen möchte, kann sich meines Buches *Visual Basic 2005 – Das Entwicklerbuch* bedienen, das Sie komplett mit Beispielen unter www.activedevelop.de herunterladen können. Dessen Nachfolger, *Visual Basic 2008 – Das Entwicklerbuch*, erscheint im Übrigen erst im 3. Quartal 2008, damit es auch auf die erst mit SP1 kommenden Features eingehen kann.

