Kapitel 1: Vorwort

Programming in different languages is like composing pieces in different keys, particularly if you work at the keyboard. If you have learned or written pieces in many keys, each key will have its own special emotional aura. Also, certain kinds of figurations «lie in the hand» in one key but are awkward in another. So you are channeled by your choice of key. In some ways, even enharmonic keys, such as C-sharp and D-flat, are quite distinct in feeling. This shows how a notational system can play a significant role in shaping the final product.

(Douglas R. Hofstadter: Gödel, Escher, Bach: An Eternal Golden Braid. Basic Books, 1979; Kapitel 10; gefunden von Antonio Cisternino von der Universität Pisa, *http://www.di.unipi.it/~cisterni/*)

Was passiert, wenn Microsoft dem Delphi-Architekten Anders Heijlsberg die Windows-Plattform, Java, C++ sowie eine Prise Visual Basic als Zutaten gibt? Es entsteht eine neue Programmiersprache. Aber wie nennt man sie? Da sie in der Tradition von C steht und C sehr stark etabliert ist, sollte der Name auf die Herkunft hinweisen, dabei jedoch auch klar machen, daß es sich um eine neue Generation handelt. So entstand der Name »C#« für die neue Programmiersprache von Microsoft.

Ein Musiker würde sagen, C# ist C mit Erhöhungszeichen, also »Cis«. Eine Verbesserung ist damit natürlich nicht gemeint, vielmehr eine Erhöhung um einen Halbton. Wenn man einen Blick in die Geschichte von C wirft, wird das Wortspiel jedoch klarer: C entstand aus der Programmiersprache B, benannt nach dem Arbeitgeber von Ken Thompson, den Bell Laboratories. Betrachtet man eine Klaviatur, stellt man fest, daß C die auf B (dem im deutschen Sprachraum H genannten Ton) folgende weiße Taste ist. C# wiederum ist die auf C folgende schwarze Taste. C# bezeichnet somit einen weiteren Generationssprung, obwohl der Buchstabe C weiter im Namen enthalten ist. Übrigens sind auf der Klaviatur B# und C dieselbe Taste.

Die Doppelbödigkeit des Namens geht aber noch weiter: Im kommerziellen Bereich ist heutzutage C++ weitaus häufiger im Einsatz als C. Auch der Name C++ soll darauf hinweisen, daß C++ der Nachfolger von C ist. Hier bediente man sich jedoch nicht einer Tonleiter, sondern dem C-Operator zum Inkrementieren. Microsoft verwendet für den Begriff C# somit eine ähnliche Symbolik wie der C++-Erschaffer Bjarne Stroustrup und unterstreicht damit die Verwandschaft zwischen den beiden Programmiersprachen.

Aufgrund des Namens könnte man meinen, C# sei eine Variante von C++. Dieser Eindruck wird allerdings durch die Aussprache des Begriffs C# im Englischen bewußt geschmälert: »see sharp« bedeutet übersetzt so viel wie »scharf sehen«. Das soll heißen: C# ist präziser, exakter, konsequenter. Diese Sichtweise verstärkt sich, wenn man C# mit »schärferes C« übersetzt.

Was ist nun aber C#? Ein besseres C++? Ein konsequenteres Visual Basic? Ein Java-Clone? Ein Delphi mit C-Syntax? Wie muß man C# in die Welt der Programmiersprachen einordnen? Microsoft positioniert C# eindeutig gegen Java, weshalb die Syntax dieser beiden Sprachen zu großen Teilen übereinstimmt und nur in Details voneinander abweicht. Auffällig ist weiterhin, daß C# fast vollständig über den Funktionsumfang von Delphi beziehungsweise Object Pascal verfügt. Und als Grundlage dieser Melange nennen die Väter von C# die Programmiersprache C/C++.

Sind Sie C++-Programmierer? Dann ist für Sie C# eine sichere und rein objektorientierte Variante von C++. Sind Sie Delphi/Object-Pascal-Programmierer? Dann kennen Sie die meisten Konzepte von C# bereits und müssen sich nur auf eine andere Syntax einstellen und darauf achten,

daß C# wie C(++) und Java case-sensitiv ist. Sind Sie Java-Programmierer? Dann ergeben sich für Sie nur minimale Veränderungen, die erweiterten Möglichkeiten von C# werden Sie aber recht bald schätzen lernen. Sind Sie Visual-Basic-Programmierer? Dann wird Sie C# nicht überraschen. Von welcher Programmiersprache Sie auch kommen, Sie werden sich auf C# mit Leichtigkeit umstellen können. Wie ähnlich sich die erwähnten Sprachen sind, zeigt die Tabelle am Beginn von Kapitel 7.

Für Microsoft ist C# eine tragende Säule der Firmen-Strategie .NET. Auch bei diesem Begriff, der englisch »dot net« ausgesprochen wird, wird mit der Wirkung des Wortes gespielt: .NET bezieht sich auf die vernetzte Welt und soll suggerieren, daß jedes Produkt mit diesem Zusatz für das Internet geschaffen wurde.

So klangvoll und hintersinnig die Begriffe C# und .NET auch sind, so ungeschickt wirken sie im Internet: Das Zeichen »#« besitzt in einer HTTP-URL eine Sonderbedeutung. Eine Website mit der URL *http://www.c#.com* kann es somit nicht geben, sie müßte mit *http://www.c%23.com* umschrieben werden. Ganz davon abgesehen sind Sonderzeichen in Domainnamen nicht erlaubt. Auch die weltweit führende Suchmaschine Google hatte anfangs Probleme damit. Für C# mußten die Google-Entwickler Ausnahmeregeln einführen, damit die Internet-Gemeinde Informationen über diese Programmiersprache finden kann. Bis heute hat Google Probleme mit dem Wort ».NET«. Der Punkt wird im Suchbegriff ignoriert. Sucht man beispielsweise nach »Java .NET«, erhält man nicht etwa Seiten über das Verhältnis zwischen Java und .NET, sondern Seiten über Netzwerkprogrammierung in Java. Doch nicht nur die vernetzte Welt hat Probleme mit den Begriffen. Auch das Stöhnen der Bibliothekare kann man sich vorstellen.

Garantiert keine Begriffsverwirrung werden Sie beim Lesen dieses Buches erleben! Das Ziel dieses Buchs ist es, Ihnen C# und den essentiellen Teil des .NET Frameworks praxisnah vorzustellen. Es beginnt deshalb mit einer kurzen Einführung in die *Idee und Begriffswelt* von Microsoft .NET. Danach werden die wichtigsten *Entwicklungsumgebungen* vorgestellt. Da die Wahl der Entwicklungsumgebung in erster Linie eine Frage des Geschmacks ist, haben wir Wert darauf gelegt, neben Visual Studio auch andere beliebte IDE-Alternativen wie den C#-Builder oder SharpDevelop zu berücksichtigen. Das Kapitel 3 widmet sich der *Programmiersprache C#* und ist für diejenigen, die bereits Erfahrungen mit objektorientierten Programmiersprachen gesammelt haben, geschrieben.

Da man mit einer Programmiersprache ohne *Bibliotheken* nicht viel erreichen kann, stellt das nachfolgende Kapitel die wichtigsten Teile des .NET Frameworks vor. Natürlich kann ein Buch über C# und .NET nur Auschnitte aus dem umfassenden Angebot vorstellen. Immerhin füllt die MSDN-Referenz in gedruckter Form tausende Seiten Papier. Bei der Auswahl der Themen orientierten wir uns an den in der Praxis am häufigsten benötigten Anwendungen des .NET Frameworks. Lernen Sie, wie

man Web Services programmiert, Websites mit ASP.NET aufbaut oder Daten aus einer Datenbank mit ADO.NET erhält. Darüber hinaus haben wir einen weiteren Schwerpunkt auf die klassischen Themen der Windows-Programmierung gelegt. Die Erzeugung einer Benutzeroberfläche mit Windows Forms ist deshalb ebenso Thema dieses Kapitels wie die Interaktion von .NET-Anwendungen mit COM-Komponenten.

Kapitel 5 widmet sich der Welt der *mobilen Endgeräte* und stellt vor, wie man Anwendungen mit dem für dieses Anwendungsgebiet geschaffenen Microsoft .NET Compact Framework programmiert. Besonders *häufig auftretenden Problemstellungen* widmet sich das darauf folgende Kapitel. Es zeigt beispielsweise, wie man mit Dateien umgeht oder wie Multithreading in einer C#-Anwendung realisiert werden kann.

C# als Nesthäkchen unter den objektorientierten Programmiersprachen wird für viele Software-Entwickler nicht die erste Sprache sein. Die meisten werden vielmehr von C++ oder Java kommen. Um diesem Umstand Rechnung zu tragen, erleichtert Kapitel 7 Umsteigern durch eine Darstellung der *Unterschiede zwischen C# und C++* beziehungsweise *Java* die *Migration*. Das nächste Kapitel wirft einen Blick hinter die Kulissen des *.NET-Assemblercodes* und zeigt, wie einfach es ist, aus einer kompilierten .NET-Anwendung einen gültigen C#-Quelltext zu generieren und erläutert, wie man das verhindern kann. Zum Abschluß wird demonstriert, wie man auf die Funktionen von Windows selbst zurückgreift.

Kapitel 8 widmet sich der *Integration von .NET-Anwendungen in Windows* und liefert Antworten auf Fragen wie »Was muß die Installationsroutine machen?« oder »Was passiert mit DLL-Dateien?«. Daß aber C# und .NET nicht nur auf die Windows-Plattform angewiesen ist, zeigt das letzte Kapitel, das die *Open-Source-Implementierungen des .NET Frameworks*, die auch auf *Unix-Systemen wie Linux* laufen, vorstellt.

Dieses großartige Buch wurde von mehreren Autoren verfaßt, die Sie vielleicht schon von ihren Veröffentlichungen in diesem Verlag oder der Zeitschrift Toolbox her kennen. Dies sind

Michael Fischer, Alexander Mayer, Michael Starke und Michael Steil

Sie haben diesem Projekt viel Zeit und Hingabe gewidmet, weshalb ich ihnen an dieser Stelle ganz herzlich danke!

Ihnen, lieber Leser, wünsche ich viel Spaß und viel Erfolg beim Arbeiten mit diesem Buch (und mit C#)!

Christian Bleske

Herausgeber