Geleitwort

C# und dessen Entwicklung sind untrennbar mit der darunterliegenden Laufzeitumgebung – dem .NET-Framework – verbunden. Denn obwohl es unter .NET eine Vielzahl von Programmiersprachen gibt, nimmt C# als die Implementierungssprache von .NET eine Sonderstellung ein. Das .NET-Framework verfolgt das Ziel, die Entwicklung diverser Anwendungen auf unterschiedlichen Plattformen wie Windows, Linux, macOS, iOS oder Android zu vereinfachen und den Entwickler dabei im Hinblick auf geforderte Qualitätskriterien wie Sicherheit, Performance und Ressourcenbedarf zu unterstützen.

Heute findet man .NET – und damit auch C# – nicht nur auf Desktop- und Serversystemen, sondern auch auf mobilen Geräten, auf Mikrocontrollern, im IoT-Umfeld sowie in Cloud- und Container-basierten Lösungen. Dank des relativ neuen Konzepts der Web Assemblies gibt es sogar bereits erste Ansätze, C# auch für Web-Frontends einzusetzen.

Um dieser Diversität gerecht zu werden, müssen C# und .NET immer wieder an neue Anforderungen angepasst werden. Beispiele dafür sind moderne CPUs, die Entwickler vor die Aufgabe stellen, sich mit parallelen Anwendungen zu beschäftigen, oder SIMD-Technologien für die Beschleunigung spezieller Berechnungen. Aber auch Spezifika bestimmter Geräteklassen (z.B. kleine Geräte mit limitierten Ressourcen versus große Cloud-Anwendungen) oder die zunehmende Verbreitung von Container-Technologien sind zu berücksichtigen. Um diese Weiterentwicklung auf eine breitere Basis zu stellen, werden zahlreiche .NET-Technologien (z.B. der C#-Compiler) im Rahmen der .NET Foundation als Open-Source-Projekte weiterentwickelt. Neben den etablierten Implementierungen von .NET (z.B. das »klassische« .NET-Framework oder Mono), kam es in den letzten Jahren mit *.NET Core* zu einer vollständigen Neuentwicklung des .NET-Frameworks auf Open-Source-Basis.

Was hat das alles aber mit C# zu tun? Nun, C# wurde so entworfen, dass die Eigenschaften von .NET in dieser Sprache optimal genutzt und Teile von .NET selbst, wie etwa ein Großteil der Klassenbibliothek, in dieser Sprache möglichst einfach implementiert werden können. So ermöglichen zum Beispiel partielle Klassen die effiziente Erweiterung generierter Klassen, Language Integrated Queries (LINQ) erlauben die einfachere parallelisierte Verarbeitung von Daten, und asynchrone Methoden ermöglichen eine bessere Nutzung verfügbarer Ressourcen bzw. die Erstellung benutzerfreundlicherer Anwendungen.

Die Mächtigkeit von C# sowie seine enge Verflechtung mit .NET sind jedoch für manchen Einsteiger ein wenig verwirrend. Genau hier setzt das vorliegende Buch an. Der Autor gibt darin – basierend auf seiner langjährigen Erfahrung mit Programmiersprachen – einen kompakten Überblick über C# für Praktiker. Die Querverweise zu Java sowie zahlreiche Beispiele und Übungsaufgaben mit Musterlösungen ermöglichen ein rasches Einarbeiten in die Materie. Aber auch der Blick hinter die Kulissen der Sprache kommt nicht zu kurz, sodass Leser auch die komplexeren Fähigkeiten der Sprache verstehen und somit sinnvoll in ihren Programmen einsetzen können.

Andreas Schabus
Premier Field Engineer
Microsoft Österreich GmbH

Vorwort

C# ist heute eine der beliebtesten Programmiersprachen, sowohl in der Industrie als auch an Schulen und Universitäten. Als moderne Programmiersprache für Praktiker bietet sie alle relevanten Konzepte der Softwaretechnik wie Typsicherheit, Objektorientierung, Generizität, Parallelität, Ausnahmebehandlung, Versionierung und vieles mehr. Sie enthält auch Elemente funktionaler Sprachen wie Funktionstypen (*Delegates*), Lambda-Ausdrücke, Tupel, Pattern Matching oder Typinferenz und eignet sich hervorragend zur Implementierung komplexer Softwaresysteme auf Desktops, Servern und mobilen Geräten.

Die aktuelle Version C# 7 brachte wieder einige interessante Neuerungen. Zum Beispiel wurden *Tupel* eingeführt, die es erlauben, mehrere Datenwerte zu gruppieren, ohne dafür einen eigenen Datentyp deklarieren zu müssen. Damit lassen sich nun zum Beispiel Funktionen mit mehreren Rückgabewerten realisieren. Wie in vielen funktionalen Sprachen gibt es nun auch in C# das Konzept des *Pattern Matching*, das Fallunterscheidungen bei Typtests und in switch-Anweisungen erleichtert. Ferner dürfen Methoden nun wie in Pascal geschachtelt werden, was eine bessere Strukturierung von Programmen erlaubt und die Lokalität von Variablen fördert. Und schließlich wurde analog zur Parameterübergabe »*by reference*« auch die Rückgabe von Funktionswerten »*by reference*« eingeführt, was die Rückgabe großer Objekte effizienter macht.

C# ist eng mit dem .NET-Framework von Microsoft verknüpft, einer Softwareplattform, die mit vielen Problemen der herkömmlichen Windows-Programmierung aufräumt. Obwohl man unter .NET in vielen verschiedenen Sprachen programmieren kann (unter anderem in Visual Basic .NET, in C++, in IronPython oder in F#), ist C# die Hauptsprache, die das .NET-Framework am besten unterstützt und die von .NET am besten unterstützt wird. Wer also unter .NET programmieren will, kommt früher oder später nicht darum herum, C# zu erlernen.

Inhalt

Dieses Buch beschreibt den gesamten Sprachumfang von C# 7 in kompakter Form. Angefangen von den einfachen Datentypen und Anweisungen über die objektorientierte und komponentenbasierte Programmierung bis hin zu Threads, Generizität, Ausnahmebehandlung, Lambda-Ausdrücken, anonymen Typen oder

der Verarbeitung von Datenströmen mit SQL-artigen Query-Ausdrücken wird der Leser mit allen Eigenschaften von C# vertraut gemacht. Die einzelnen Sprachelemente werden vorwiegend anhand von Beispielen erklärt. Im Anhang findet sich dann die (etwas vereinfachte) Grammatik von C#, die für jedes Sprachelement seine genaue Syntax angibt.

Keine Programmiersprache kommt heute ohne Klassenbibliothek aus. Deshalb gibt dieses Buch auch einen Überblick über die reichhaltige Klassenbibliothek von .NET. Ferner wird das Buch durch ein Kapitel über Fallstudien abgerundet, das auf die Erstellung grafischer Benutzeroberflächen mittels *Windows Forms*, auf die Implementierung von *Web-Services* (Business-to-Business-Dienste im Internet) und die Erstellung dynamischer Webseiten mittels *ASP.NET* eingeht.

Zielpublikum

Dieses Buch richtet sich an Praktiker, die bereits eine Programmiersprache wie Java oder C++ beherrschen und C# kennenlernen wollen. Es richtet sich aber auch an Studenten, die C# in fortgeschrittenen Lehrveranstaltungen zum Thema Objektorientierung, Komponentenorientierung, funktionale Sprachelemente oder .NET-Programmierung einsetzen.

Am Ende jedes Kapitels findet man zahlreiche Übungsaufgaben, zu denen es unter *http://dotnet.jku.at* Musterlösungen gibt. Dadurch ist das Buch auch zum Selbststudium geeignet.

Die Webseite *http://dotnet.jku.at* enthält auch einen umfangreichen Powerpoint-Foliensatz, den ich für eine Vorlesung über C# erstellt habe und der laufend aktualisiert wird. Der Foliensatz soll anderen Vortragenden helfen, C#-Inhalte in ihren eigenen Unterricht einzubauen oder sogar eine eigene Vorlesung über C# zu halten.

Mein Dank gilt meinen Mitarbeitern an der Johannes Kepler Universität Linz für die zahlreichen Kommentare zu diesem Buch sowie für das Korrekturlesen des Manuskripts. Auch Leserinnen und Leser haben immer wieder mit nützlichen Hinweisen zur Verbesserung früherer Versionen dieses Buchs beigetragen. Danken möchte ich auch den Mitarbeitern von Microsoft in Redmond und in Österreich insbesondere für Vorabinformationen zu neuen Sprachversionen von C#. Ganz besonders möchte ich mich aber für die hervorragende Zusammenarbeit mit dem *dpunkt.verlag* bedanken. Angefangen vom Lektorat über die Herstellung und den Vertrieb bis zur professionellen Betreuung der Autoren bietet dieser Verlag einfach alles, was sich ein Autor wünschen kann.

Hanspeter Mössenböck
September 2018