

**Update
inside**

Jürgen KOTZ
Christian WENZ

C# und .NET 6

GRUNDLAGEN, PROFIWISSEN UND REZEPTE



Bonuskapitel und Beispiele unter
plus.hanser-fachbuch.de

HANSER

Kotz/Wenz

C# und .NET 6 Grundlagen, Profiwissen und Rezepte



Ihr Plus – digitale Zusatzinhalte!

Auf unserem Download-Portal finden Sie zu diesem Titel kostenloses Zusatzmaterial.

Geben Sie auf **plus.hanser-fachbuch.de** einfach diesen Code ein:

plus-43qr9-pKvN2



Bleiben Sie auf dem Laufenden!

Unser **Computerbuch-Newsletter** informiert Sie monatlich über neue Bücher und Termine. Profitieren Sie auch von Gewinnspielen und exklusiven Leseproben. Gleich anmelden unter:

www.hanser-fachbuch.de/newsletter



Update inside.

Mit unserem kostenlosen Update-Service zum Buch erhalten Sie aktuelle Infos zu C# und dem .NET Framework.

Und so funktioniert es:

1. Registrieren Sie sich unter:
www.hanser-fachbuch.de/Csharp-update
2. Geben Sie diesen Code ein:

5UKz-TaNroZp-mR5n

Der Update-Service läuft bis März 2024. Als registrierter Nutzer werden Sie in diesem Zeitraum persönlich per E-Mail informiert, sobald ein neues Buch-Update zum Download verfügbar ist.

Wenn Sie Fragen haben, wenden Sie sich gerne an: **Csharp-update@hanser.de**

Jürgen Kotz
Christian Wenz

C# und .NET 6

Grundlagen, Profiwissen und Rezepte

HANSER

Die Autoren:

Jürgen Kotz, München

Christian Wenz, München

Alle in diesem Buch enthaltenen Informationen, Verfahren und Darstellungen wurden nach bestem Wissen zusammengestellt und mit Sorgfalt getestet. Dennoch sind Fehler nicht ganz auszuschließen. Aus diesem Grund sind die im vorliegenden Buch enthaltenen Informationen mit keiner Verpflichtung oder Garantie irgendeiner Art verbunden. Autor und Verlag übernehmen infolgedessen keine juristische Verantwortung und werden keine daraus folgende oder sonstige Haftung übernehmen, die auf irgendeine Art aus der Benutzung dieser Informationen – oder Teilen davon – entsteht.

Ebenso übernehmen Autor und Verlag keine Gewähr dafür, dass beschriebene Verfahren usw. frei von Schutzrechten Dritter sind. Die Wiedergabe von Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. in diesem Buch berechtigt deshalb auch ohne besondere Kennzeichnung nicht zu der Annahme, dass solche Namen im Sinne der Warenzeichen- und Markenschutz-Gesetzgebung als frei zu betrachten wären und daher von jedermann benutzt werden dürften.

Bibliografische Information der Deutschen Nationalbibliothek:

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.d-nb.de> abrufbar.

Dieses Werk ist urheberrechtlich geschützt.

Alle Rechte, auch die der Übersetzung, des Nachdruckes und der Vervielfältigung des Buches, oder Teilen daraus, vorbehalten. Kein Teil des Werkes darf ohne schriftliche Genehmigung des Verlages in irgendeiner Form (Fotokopie, Mikrofilm oder ein anderes Verfahren) – auch nicht für Zwecke der Unterrichtsgestaltung – reproduziert oder unter Verwendung elektronischer Systeme verarbeitet, vervielfältigt oder verbreitet werden.

© 2022 Carl Hanser Verlag München, www.hanser-fachbuch.de

Lektorat: Sylvia Hasselbach

Copy editing: Walter Saumweber, Ratingen

Umschlagdesign: Marc Müller-Bremer, München, www.rebranding.de

Umschlagrealisation: Max Kostopoulos

Titelmotiv: © thinkstockphotos.de/liuzishan

Gesamtherstellung: Kösel, Krugzell

Ausstattung patentrechtlich geschützt. Kösel FD 351, Patent-Nr. 0748702

Printed in Germany

Print-ISBN: 978-3-446-46930-3

E-Book-ISBN: 978-3-446-47349-2

E-Pub-ISBN: 978-3-446-47423-9

Inhalt

Vorwort	XXI
Zusatzmaterial online	XXIII
Teil I: Grundlagen	1
1 .NET 6	3
1.1 Microsofts .NET-Technologie	4
1.1.1 Zur Geschichte von .NET	4
1.1.2 .NET-Features und Begriffe	6
1.2 .NET Core	13
1.2.1 Geschichte von .NET Core	13
1.2.2 LTS - Long Term Support und zukünftige Versionen	15
1.2.3 .NET Standard	15
1.3 Features von .NET 6	16
2 Einstieg in Visual Studio 2022	19
2.1 Die Installation von Visual Studio 2022	19
2.1.1 Überblick über die Produktpalette	19
2.1.2 Anforderungen an Hard- und Software	20
2.2 Unser allererstes C#-Programm	21
2.2.1 Vorbereitungen	21
2.2.2 Quellcode schreiben	24
2.2.3 Programm kompilieren und testen	24
2.2.4 Einige Erläuterungen zum Quellcode	25
2.2.5 Konsolenanwendungen sind out	26
2.3 Die Windows-Philosophie	27
2.3.1 Mensch-Rechner-Dialog	27
2.3.2 Objekt- und ereignisorientierte Programmierung	27
2.3.3 Programmieren mit Visual Studio 2022	29
2.4 Die Entwicklungsumgebung Visual Studio 2022	30
2.4.1 Neues Projekt	30
2.4.2 Die wichtigsten Fenster	33
2.4.3 Projektvorlagen in Visual Studio 2022 - Minimal APIs	36

2.5	Praxisbeispiele	37
2.5.1	Unsere erste Windows-Forms-Anwendung	37
2.5.2	Umrechnung Euro-Dollar	43
3	Grundlagen der Sprache C#	53
3.1	Grundbegriffe	53
3.1.1	Anweisungen	53
3.1.2	Bezeichner	54
3.1.3	Schlüsselwörter	55
3.1.4	Kommentare	56
3.2	Datentypen, Variablen und Konstanten	57
3.2.1	Fundamentale Typen	57
3.2.2	Werttypen versus Verweistypen	58
3.2.3	Benennung von Variablen	59
3.2.4	Deklaration von Variablen	59
3.2.5	Typsuffixe	60
3.2.6	Zeichen und Zeichenketten	61
3.2.7	object-Datentyp	63
3.2.8	Konstanten deklarieren	64
3.2.9	Nullable Types	64
3.2.10	Typinferenz	66
3.2.11	Gültigkeitsbereiche und Sichtbarkeit	66
3.3	Konvertieren von Datentypen	67
3.3.1	Implizite und explizite Konvertierung	67
3.3.2	Welcher Datentyp passt zu welchem?	69
3.3.3	Konvertieren von string	70
3.3.4	Die Convert-Klasse	71
3.3.5	Die Parse-Methode	72
3.3.6	Boxing und Unboxing	73
3.4	Operatoren	74
3.4.1	Arithmetische Operatoren	75
3.4.2	Zuweisungsoperatoren	77
3.4.3	Logische Operatoren	78
3.4.4	Rangfolge der Operatoren	81
3.5	Kontrollstrukturen	82
3.5.1	Verzweigungsbefehle	82
3.5.2	Schleifenanweisungen	87
3.6	Benutzerdefinierte Datentypen	89
3.6.1	Enumerationen	90
3.6.2	Strukturen	91
3.7	Nutzerdefinierte Methoden	93
3.7.1	Methoden mit Rückgabewert	94
3.7.2	Methoden ohne Rückgabewert	95
3.7.3	Parameterübergabe mit ref	97

3.7.4	Parameterübergabe mit out	98
3.7.5	Methodenüberladung	99
3.7.6	Optionale Parameter	100
3.7.7	Benannte Parameter	101
3.8	Praxisbeispiele	102
3.8.1	Vom PAP zur Konsolenanwendung	102
3.8.2	Ein Konsolen- in ein Windows-Programm verwandeln	105
3.8.3	Schleifenanweisungen verstehen	107
3.8.4	Benutzerdefinierte Methoden überladen	110
3.8.5	Anwendungen von Visual Basic nach C# portieren	113
4	OOO-Konzepte	121
4.1	Kleine Einführung in die OOP	121
4.1.1	Historische Entwicklung	122
4.1.2	Grundbegriffe der OOP	123
4.1.3	Sichtbarkeit von Klassen und ihren Mitgliedern	125
4.1.4	Allgemeiner Aufbau einer Klasse	126
4.1.5	Das Erzeugen eines Objekts	128
4.1.6	Einführungsbeispiel	131
4.2	Eigenschaften	137
4.2.1	Eigenschaften mit Zugriffsmethoden kapseln	137
4.2.2	Berechnete Eigenschaften	139
4.2.3	Lese-/Schreibschutz	141
4.2.4	Property-Accessoren	142
4.2.5	Statische Felder/Eigenschaften	143
4.2.6	Einfache Eigenschaften automatisch implementieren	145
4.3	Methoden	147
4.3.1	Öffentliche und private Methoden	147
4.3.2	Überladene Methoden	148
4.3.3	Statische Methoden	149
4.4	Ereignisse	150
4.4.1	Ereignis hinzufügen	151
4.4.2	Ereignis verwenden	154
4.5	Arbeiten mit Konstruktor und Destruktor	157
4.5.1	Konstruktor und Objektinitialisierer	158
4.5.2	Destruktor und Garbage Collector	161
4.5.3	Mit using den Lebenszyklus des Objekts kapseln	163
4.6	Vererbung und Polymorphie	164
4.6.1	Method-Overriding	164
4.6.2	Klassen implementieren	164
4.6.3	Implementieren der Objekte	167
4.6.4	Ausblenden von Mitgliedern durch Vererbung	169
4.6.5	Allgemeine Hinweise und Regeln zur Vererbung	170

4.6.6	Polymorphes Verhalten	172
4.6.7	Die Rolle von System.Object	175
4.7	Spezielle Klassen	176
4.7.1	Abstrakte Klassen	176
4.7.2	Versiegelte Klassen	177
4.7.3	Partielle Klassen	178
4.7.4	Statische Klassen	179
4.8	Schnittstellen (Interfaces)	180
4.8.1	Definition einer Schnittstelle	181
4.8.2	Implementieren einer Schnittstelle	181
4.8.3	Abfragen, ob Schnittstelle vorhanden ist	182
4.8.4	Mehrere Schnittstellen implementieren	183
4.8.5	Schnittstellenprogrammierung ist ein weites Feld ...	183
4.9	Datensatztypen - Records	183
4.9.1	Definition eines Record	184
4.9.2	Mutable Properties	186
4.9.3	Nicht-destruktive Änderung	188
4.9.4	Dekonstruktion	189
4.10	Praxisbeispiele	190
4.10.1	Eigenschaften sinnvoll kapseln	190
4.10.2	Eine statische Klasse anwenden	193
4.10.3	Vom fetten zum schlanken Client	194
4.10.4	Schnittstellenvererbung verstehen	205
4.10.5	Rechner für komplexe Zahlen	210
4.10.6	Sortieren mit IComparable/IComparer	218
4.10.7	Einen Objektbaum in generischen Listen abspeichern	223
4.10.8	OOP beim Kartenspiel erlernen	228
4.10.9	Eine Klasse zur Matrizenrechnung entwickeln	233
4.10.10	Vererbung von Records	239
5	Arrays, Strings, Funktionen	241
5.1	Datenfelder (Arrays)	241
5.1.1	Array deklarieren	242
5.1.2	Array instanziiieren	242
5.1.3	Array initialisieren	243
5.1.4	Zugriff auf Array-Elemente	244
5.1.5	Zugriff mittels Schleife	245
5.1.6	Mehrdimensionale Arrays	246
5.1.7	Zuweisen von Arrays	248
5.1.8	Arrays aus Strukturvariablen	249
5.1.9	Löschen und Umdimensionieren von Arrays	250
5.1.10	Eigenschaften und Methoden von Arrays	252
5.1.11	Übergabe von Arrays	253

5.2	Verarbeiten von Zeichenketten	255
5.2.1	Zuweisen von Strings	255
5.2.2	Eigenschaften und Methoden von String-Variablen	256
5.2.3	Wichtige Methoden der String-Klasse	258
5.2.4	Die StringBuilder-Klasse	260
5.3	Datums- und Zeitberechnungen	263
5.3.1	Die DateTime-Struktur	263
5.3.2	Wichtige Eigenschaften von DateTime-Variablen	264
5.3.3	Wichtige Methoden von DateTime-Variablen	265
5.3.4	Wichtige Mitglieder der DateTime-Struktur	266
5.3.5	Konvertieren von Datumstrings in DateTime-Werte	267
5.3.6	Die TimeSpan-Struktur	267
5.3.7	DateOnly und TimeOnly	269
5.4	Mathematische Funktionen	270
5.4.1	Überblick	270
5.4.2	Zahlen runden	270
5.4.3	Winkel umrechnen	271
5.4.4	Potenz- und Wurzeloperationen	271
5.4.5	Logarithmus und Exponentialfunktionen	271
5.4.6	Zufallszahlen erzeugen	272
5.4.7	Kreisberechnung	273
5.5	Zahlen- und Datumsformatierungen	273
5.5.1	Anwenden der ToString-Methode	274
5.5.2	Anwenden der Format-Methode	275
5.5.3	Stringinterpolation	276
5.6	Praxisbeispiele	277
5.6.1	Zeichenketten verarbeiten	277
5.6.2	Zeichenketten mit StringBuilder addieren	280
5.6.3	Methodenaufrufe mit Array-Parametern	283
6	Weitere Sprachfeatures	289
6.1	Namespaces (Namensräume)	289
6.1.1	Ein kleiner Überblick	289
6.1.2	Einen eigenen Namespace einrichten	290
6.1.3	Die using-Anweisung	292
6.1.4	Namespace Alias	293
6.1.5	Globale using-Anweisungen	293
6.2	Operatorenüberladung	294
6.2.1	Syntaxregeln	294
6.2.2	Praktische Anwendung	295
6.3	Collections (Auflistungen)	296
6.3.1	Die Schnittstelle IEnumerable	296
6.3.2	ArrayList	299

6.3.3	Hashtable	300
6.3.4	Indexer	301
6.4	Generics	303
6.4.1	Generics bieten Typsicherheit	304
6.4.2	Generische Methoden	305
6.4.3	yield - Iteratoren	306
6.5	Generische Collections	306
6.5.1	List-Collection statt ArrayList	306
6.5.2	Vorteile generischer Collections	308
6.5.3	Constraints	308
6.6	Das Prinzip der Delegates	309
6.6.1	Delegates sind Methodenzeiger	309
6.6.2	Einen Delegate-Typ deklarieren	310
6.6.3	Ein Delegate-Objekt erzeugen	310
6.6.4	Anonyme Methoden	312
6.6.5	Lambda-Ausdrücke	313
6.6.6	Lambda-Ausdrücke in der Task Parallel Library	316
6.6.7	Action<> und Func<>	317
6.7	Dynamische Programmierung	319
6.7.1	Wozu dynamische Programmierung?	319
6.7.2	Das Prinzip der dynamischen Programmierung	320
6.7.3	Optionale Parameter sind hilfreich	322
6.7.4	Kovarianz und Kontravarianz	323
6.8	Weitere Datentypen	324
6.8.1	BigInteger	324
6.8.2	Complex	326
6.8.3	Tuple<>	327
6.8.4	SortedSet<>	328
6.9	Praxisbeispiele	329
6.9.1	ArrayList versus generische List	329
6.9.2	Generische IEnumerable-Interfaces implementieren	332
6.9.3	Delegates, Func, anonyme Methoden, Lambda Expressions	336
7	Einführung in LINQ	341
7.1	LINQ-Grundlagen	341
7.1.1	Die LINQ-Architektur	341
7.1.2	Anonyme Typen	343
7.1.3	Erweiterungsmethoden	344
7.2	Abfragen mit LINQ to Objects	345
7.2.1	Grundlegendes zur LINQ-Syntax	346
7.2.2	Zwei alternative Schreibweisen von LINQ-Abfragen	347
7.2.3	Übersicht der wichtigsten Abfrageoperatoren	348

7.3	LINQ-Abfragen im Detail	349
7.3.1	Die Projektionsoperatoren Select und SelectMany	350
7.3.2	Der Restriktionsoperator Where	351
7.3.3	Die Sortierungsoperatoren OrderBy und ThenBy	352
7.3.4	Der Gruppierungsoperator GroupBy	353
7.3.5	Verknüpfen mit Join	356
7.3.6	Aggregat-Operatoren	356
7.3.7	Verzögertes Ausführen von LINQ-Abfragen	358
7.3.8	Konvertierungsmethoden	359
7.3.9	Abfragen mit PLINQ	360
7.4	Praxisbeispiele	363
7.4.1	Die Syntax von LINQ-Abfragen verstehen	363
7.4.2	Aggregat-Abfragen mit LINQ	366
7.4.3	LINQ im Schnelldurchgang erlernen	369
7.4.4	Strings mit LINQ abfragen und filtern	371
7.4.5	Duplikate aus einer Liste entfernen	373
7.4.6	Arrays mit LINQ initialisieren	375
7.4.7	Arrays per LINQ mit Zufallszahlen füllen	377
7.4.8	Einen String mit Wiederholmuster erzeugen	379
7.4.9	Mit LINQ Zahlen und Strings sortieren	380
7.4.10	Mit LINQ Collections von Objekten sortieren	382
7.4.11	Where - Deep Dive	384
8	Neuerungen von C# im Überblick	393
8.1	C# 4.0 - Visual Studio 2010	394
8.1.1	Datentyp dynamic	394
8.1.2	Benannte und optionale Parameter	395
8.1.3	Kovarianz und Kontravarianz	396
8.2	C# 5.0 - Visual Studio 2012	396
8.2.1	Async und Await	397
8.2.2	CallerInfo	398
8.3	Visual Studio 2013	399
8.4	C# 6.0 - Visual Studio 2015	399
8.4.1	String Interpolation	399
8.4.2	Schreibgeschützte AutoProperties	399
8.4.3	Initialisierer für AutoProperties	400
8.4.4	Expression Body Member	400
8.4.5	using static	400
8.4.6	Bedingter Nulloperator	401
8.4.7	Ausnahmenfilter	402
8.4.8	nameof-Ausdrücke	402
8.4.9	await in catch und finally	403
8.4.10	Indexinitialisierer	403

8.5	C# 7.0 – Visual Studio 2017	403
8.5.1	out-Variablen	403
8.5.2	Tupel	404
8.5.3	Mustervergleich	405
8.5.4	Discards	406
8.5.5	Lokale ref-Variablen und Rückgabetypen	407
8.5.6	Lokale Funktionen	407
8.5.7	Mehr Expression Body Member	407
8.5.8	throw-Ausdrücke	408
8.5.9	Verbesserung der numerischen literalen Syntax	408
8.6	C# 7.1 bis 7.3 – Visual Studio 2019	408
8.6.1	C# 7.1	408
8.6.2	C# 7.2	410
8.6.3	C# 7.3	411
8.6.4	Visual Studio 2019 – Live Share	411
8.7	C# 8.0	414
8.7.1	Standardschnittstellenmethoden	414
8.7.2	Vereinfachung von switch-Ausdrücken	416
8.7.3	Eigenschaftenmuster	417
8.7.4	Vereinfachte using-Ressourcenschutzblöcke	417
8.7.5	Null-Coalescing-Zuweisungen	418
8.7.6	Nullable Referenztypen	419
8.7.7	Indizes und Bereiche	420
8.7.8	Weitere Sprachneuerungen	421
8.8	C# 9.0	422
8.8.1	Records	422
8.8.2	Init-only Setter	422
8.8.3	Weitere Verbesserungen in Musterausdrücken	422
8.8.4	Weitere Sprachneuerungen	423
8.9	C# 10	423
8.9.1	Datensatzstrukturen	423
8.9.2	Globale using-Anweisungen	423
8.9.3	Weitere Sprachneuerungen	424

Teil II: Desktop-Anwendungen **425**

9 Einführung in WPF **427**

9.1	Einführung	427
9.1.1	Was kann eine WPF-Anwendung?	428
9.1.2	Die eXtensible Application Markup Language	430
9.1.3	Unsere erste XAML-Anwendung	431
9.1.4	Zielplattformen	437
9.1.5	Applikationstypen	437
9.1.6	Vor- und Nachteile von WPF-Anwendungen	438
9.1.7	Weitere Dateien im Überblick	439

9.2	Alles beginnt mit dem Layout	441
9.2.1	Allgemeines zum Layout	441
9.2.2	Positionieren von Steuerelementen	443
9.2.3	Canvas	446
9.2.4	StackPanel	447
9.2.5	DockPanel	449
9.2.6	WrapPanel	451
9.2.7	UniformGrid	452
9.2.8	Grid	453
9.2.9	ViewBox	458
9.2.10	TextBlock	459
9.3	Das WPF-Programm	463
9.3.1	Die App-Klasse	463
9.3.2	Das Startobjekt festlegen	463
9.3.3	Kommandozeilenparameter verarbeiten	465
9.3.4	Die Anwendung beenden	466
9.3.5	Auswerten von Anwendungsereignissen	466
9.4	Die Window-Klasse	467
9.4.1	Position und Größe festlegen	468
9.4.2	Rahmen und Beschriftung	468
9.4.3	Das Fenster-Icon ändern	468
9.4.4	Anzeige weiterer Fenster	469
9.4.5	Transparenz	469
9.4.6	Abstand zum Inhalt festlegen	470
9.4.7	Fenster ohne Fokus anzeigen	470
9.4.8	Ereignisfolge bei Fenstern	471
9.4.9	Ein paar Worte zur Schriftdarstellung	471
9.4.10	Ein paar Worte zur Darstellung von Controls	474
9.4.11	Wird mein Fenster komplett mit WPF gerendert?	475
10	Übersicht WPF-Controls	477
10.1	Allgemeingültige Eigenschaften	477
10.2	Label	479
10.3	Button, RepeatButton, ToggleButton	480
10.3.1	Schaltflächen für modale Dialoge	480
10.3.2	Schaltflächen mit Grafik	482
10.4	TextBox, PasswordBox	483
10.4.1	TextBox	483
10.4.2	PasswordBox	485
10.5	CheckBox	486
10.6	RadioButton	488
10.7	ListBox, ComboBox	489
10.7.1	ListBox	489

10.7.2	ComboBox	492
10.7.3	Den Content formatieren	494
10.8	Image	496
10.8.1	Grafik per XAML zuweisen	496
10.8.2	Grafik zur Laufzeit zuweisen	496
10.8.3	Bild aus Datei laden	498
10.8.4	Die Grafiskalierung beeinflussen	499
10.9	Slider, ScrollBar	500
10.9.1	Slider	500
10.9.2	ScrollBar	502
10.10	ScrollViewer	502
10.11	Menu, ContextMenu	503
10.11.1	Menu	504
10.11.2	Tastenkürzel	505
10.11.3	Grafiken	506
10.11.4	Weitere Möglichkeiten	507
10.11.5	ContextMenu	508
10.12	ToolBar	509
10.13	StatusBar, ProgressBar	512
10.13.1	StatusBar	512
10.13.2	ProgressBar	514
10.14	Border, GroupBox, BulletDecorator	514
10.14.1	Border	515
10.14.2	GroupBox	516
10.14.3	BulletDecorator	517
10.15	Expander, TabControl	519
10.15.1	Expander	519
10.15.2	TabControl	521
10.16	Popup	522
10.17	TreeView	525
10.18	ListView	529
10.19	DataGrid	529
10.20	Calendar/DatePicker	530
10.21	Ellipse, Rectangle, Line und Co.	534
10.21.1	Ellipse	534
10.21.2	Rectangle	535
10.21.3	Line	536
11	Wichtige WPF-Techniken	537
11.1	Eigenschaften	537
11.1.1	Abhängige Eigenschaften (Dependency Properties)	537
11.1.2	Angehängte Eigenschaften (Attached Properties)	539

11.2	Einsatz von Ressourcen	539
11.2.1	Was sind eigentlich Ressourcen?	539
11.2.2	Wo können Ressourcen gespeichert werden?	540
11.2.3	Wie definiere ich eine Ressource?	541
11.2.4	Statische und dynamische Ressourcen	542
11.2.5	Wie werden Ressourcen adressiert?	544
11.2.6	Systemressourcen einbinden	545
11.3	Das WPF-Ereignismodell	545
11.3.1	Einführung	545
11.3.2	Routed Events	546
11.3.3	Direkte Events	549
11.4	Verwendung von Commands	549
11.4.1	Einführung zu Commands	549
11.4.2	Verwendung vordefinierter Commands	550
11.4.3	Das Ziel des Commands	552
11.4.4	Vordefinierte Commands	553
11.4.5	Commands an Ereignismethoden binden	553
11.4.6	Wie kann ich ein Command per Code auslösen?	554
11.4.7	Command-Ausführung verhindern	555
11.5	Das WPF-Style-System	555
11.5.1	Übersicht	555
11.5.2	Benannte Styles	556
11.5.3	Typ-Styles	558
11.5.4	Styles anpassen und vererben	559
11.6	Verwenden von Triggern	561
11.6.1	Eigenschaften-Trigger (Property Triggers)	562
11.6.2	Ereignis-Trigger	564
11.6.3	Daten-Trigger	565
11.7	Einsatz von Templates	566
11.7.1	Neues Template erstellen	566
11.7.2	Template abrufen und verändern	570
11.8	Transformationen, Animationen, StoryBoards	573
11.8.1	Transformationen	573
11.8.2	Animationen mit dem StoryBoard realisieren	578

Teil III: Technologien 585

12 WPF-Datenbindung 587

12.1	Grundprinzip	587
12.1.1	Bindungsarten	588
12.1.2	Wann eigentlich wird die Quelle aktualisiert?	590
12.1.3	Geht es auch etwas langsamer?	591
12.1.4	Bindung zur Laufzeit realisieren	592

12.2	Binden an Objekte	593
12.2.1	Objekte im XAML-Code instanziiieren	594
12.2.2	Verwenden der Instanz im C#-Quellcode	595
12.2.3	Anforderungen an die Quell-Klasse	596
12.2.4	Instanziiieren von Objekten per C#-Code	597
12.3	Binden von Collections	599
12.3.1	Anforderung an die Collection	599
12.3.2	Einfache Anzeige	600
12.3.3	Navigieren zwischen den Objekten	601
12.3.4	Einfache Anzeige in einer ListBox	603
12.3.5	DataTemplates zur Anzeigeformatierung	604
12.3.6	Mehr zu List- und ComboBox	605
12.3.7	Verwendung der ListView	607
12.4	Noch einmal zurück zu den Details	610
12.4.1	Navigieren in den Daten	610
12.4.2	Sortieren	612
12.4.3	Filtern	612
12.4.4	Live Shaping	613
12.5	Anzeige von Datenbankinhalten	615
12.5.1	Installieren der benötigten NuGet-Pakete	615
12.5.2	Anlegen der Entitätsklassen	616
12.5.3	Die Programmoberfläche	619
12.5.4	Der Zugriff auf die Daten	620
12.6	Formatieren von Werten	623
12.6.1	IValueConverter	623
12.6.2	BindingBase.StringFormat-Eigenschaft	625
12.7	Das DataGrid als Universalwerkzeug	626
12.7.1	Grundlagen der Anzeige	626
12.7.2	UI-Virtualisierung	628
12.7.3	Spalten selbst definieren	628
12.7.4	Zusatzinformationen in den Zeilen anzeigen	630
12.7.5	Vom Betrachten zum Editieren	632
12.8	Praxisbeispiel – Collections in Hintergrundthreads füllen	632
13	.NET MAUI	637
13.1	Einführung	637
13.2	Was kann eine .NET-MAUI-Anwendung?	639
13.3	Die erste .NET MAUI App	640
14	Asynchrone Programmierung	647
14.1	Übersicht	648
14.1.1	Multitasking versus Multithreading	648
14.1.2	Deadlocks	649
14.1.3	Racing	650

14.2	Programmieren mit Threads	651
14.2.1	Einführungsbeispiel	651
14.2.2	Wichtige Thread-Methoden	653
14.2.3	Wichtige Thread-Eigenschaften	655
14.2.4	Einsatz der ThreadPool-Klasse	656
14.3	Sperrmechanismen	657
14.3.1	Threading ohne lock	658
14.3.2	Threading mit lock	659
14.3.3	Die Monitor-Klasse	662
14.3.4	Mutex	666
14.3.5	Methoden für die parallele Ausführung sperren	667
14.3.6	Semaphore	668
14.4	Interaktion mit der Programmoberfläche	670
14.4.1	Die Werkzeuge	671
14.4.2	Einzelne Steuerelemente mit Invoke aktualisieren (Windows Forms)	671
14.4.3	Mehrere Steuerelemente aktualisieren	672
14.4.4	Ist ein Invoke-Aufruf nötig?	673
14.4.5	Und was ist mit WPF?	673
14.5	Timer-Threads	675
14.6	Asynchrone Programmierentwurfsmuster	677
14.6.1	Kurzübersicht	677
14.6.2	Polling	678
14.6.3	Callback verwenden	680
14.6.4	Callback mit Parameterübergabe verwenden	681
14.6.5	Callback mit Zugriff auf die Programmoberfläche	682
14.7	Es geht auch einfacher – async und await	684
14.7.1	Der Weg von synchron zu asynchron	684
14.7.2	Achtung: Fehlerquellen!	686
14.7.3	Eigene asynchrone Methoden entwickeln	689
14.8	Asynchrone Streams	691
14.8.1	Datei erstellen	691
14.8.2	Datei lesen mit <code>IAsyncEnumerable<T></code>	693
14.9	Praxisbeispiele	694
14.9.1	Prozess- und Thread-Informationen gewinnen	694
14.9.2	Ein externes Programm starten	697
15	Die Task Parallel Library	701
15.1	Überblick	701
15.1.1	Parallel-Programmierung	701
15.1.2	Möglichkeiten der TPL	704
15.1.3	Der CLR-Threadpool	705
15.2	Parallele Verarbeitung mit <code>Parallel.Invoke</code>	706
15.2.1	Aufrufvarianten	706
15.2.2	Einschränkungen	708

15.3	Verwendung von Parallel.For	709
15.3.1	Abbrechen der Verarbeitung	710
15.3.2	Auswerten des Bearbeitungsstatus	712
15.3.3	Und was ist mit anderen Iterator-Schrittweiten?	713
15.4	Collections mit Parallel.ForEach verarbeiten	713
15.5	Die Task-Klasse	714
15.5.1	Einen Task erzeugen	714
15.5.2	Den Task starten	715
15.5.3	Datenübergabe an den Task	717
15.5.4	Wie warte ich auf das Ende des Tasks?	718
15.5.5	Tasks mit Rückgabewerten	720
15.5.6	Die Verarbeitung abbrechen	723
15.5.7	Fehlerbehandlung	728
15.5.8	Weitere Eigenschaften	729
15.6	Zugriff auf das User Interface	730
15.6.1	Task-Ende und Zugriff auf die Oberfläche	730
15.6.2	Zugriff auf das UI aus dem Task heraus	732
15.7	Weitere Datenstrukturen im Überblick	734
15.7.1	Threadsichere Collections	734
15.7.2	Primitive für die Threadsynchrisation	734
15.8	Parallel LINQ (PLINQ)	735
15.9	Praxisbeispiele	735
15.9.1	BlockingCollection	735
15.9.2	PLINQ	739
16	Debugging, Fehlersuche und Fehlerbehandlung	741
16.1	Der Debugger	741
16.1.1	Allgemeine Beschreibung	741
16.1.2	Die wichtigsten Fenster	742
16.1.3	Debugging-Optionen	746
16.1.4	Praktisches Debugging am Beispiel	749
16.2	Arbeiten mit Debug und Trace	754
16.2.1	Wichtige Methoden von Debug und Trace	754
16.2.2	Besonderheiten der Trace-Klasse	758
16.2.3	TraceListener-Objekte	758
16.3	Caller Information	760
16.3.1	Attribute	761
16.3.2	Anwendung	761
16.4	Fehlerbehandlung	762
16.4.1	Anweisungen zur Fehlerbehandlung	762
16.4.2	try-catch	763
16.4.3	try-finally	767
16.4.4	Das Standardverhalten bei Ausnahmen festlegen	769

16.4.5	Die Exception-Klasse	770
16.4.6	Fehler/Ausnahmen auslösen	771
16.4.7	Eigene Fehlerklassen	771
16.4.8	Exceptionhandling zur Debugzeit	773
17	Entity Framework Core 6.0	775
17.1	Überblick	775
17.1.1	Objektrelationaler Mapper (ORM)	776
17.1.2	Provider	778
17.1.3	Relationale Beziehungen	779
17.1.4	Benötigte NuGet-Pakete	782
17.2	CodeFirst	783
17.2.1	CodeFirst aus Model	784
17.2.2	CodeFirst mittels ReverseEngineering von bestehender Datenbank ...	793
17.3	Migrationen	798
17.3.1	Initiale Migration bei ReverseEngineering	798
17.3.2	Weitere Migrationen	800
17.4	Lesen und Schreiben von Daten mit EF Core 6	802
17.5	Praxisbeispiele	806
17.5.1	Daten mit EF Core 6 laden und als JSON speichern	806
17.5.2	Eine Datenbank mit EF Core 6 anlegen und Testdaten generieren und anzeigen	815
	Teil IV: Webanwendungen	823
18	Webanwendungen mit ASP.NET Core	825
18.1	Grundlagen	826
18.2	Razor Pages	829
18.2.1	Projektaufbau	829
18.2.2	Razor-Syntax	833
18.2.3	Layout-Vorlagen	838
18.2.4	Modelle für Razor Pages	842
18.2.5	Mit Formularen arbeiten	843
18.3	MVC	850
18.3.1	Projektaufbau	851
18.3.2	Action-Methoden	852
18.3.3	Zustandsmanagement	864
18.4	Praxisbeispiele	869
18.4.1	CRUD mit Entity Framework	869
18.4.2	Authentifizierung und Autorisierung	885
19	ASP.NET Web API	893
19.1	REST	893
19.2	Vorlagen	897

19.3	Daten lesen	903
19.4	Daten schreiben, aktualisieren, löschen	912
19.5	Minimale APIs	921
19.6	Praxisbeispiele	925
19.6.1	Paginierung	925
19.6.2	XML statt JSON	927
19.6.3	CORS	928
20	Blazor	933
20.1	Hosting-Modelle	934
20.2	Projektvorlagen	937
20.3	Blazor-Komponenten	946
20.3.1	Code in/für Komponenten	946
20.3.2	Event-Handling	949
20.3.3	Datenbindung	953
20.4	Services und APIs aufrufen	955
20.5	Weitere Blazor-Features	961
20.5.1	Zustandsmanagement	961
20.5.2	JavaScript-Interoperabilität	963
20.6	Praxisbeispiele	967
20.6.1	Online-Status ermitteln	967
20.6.2	File-Uploads	972
20.6.3	Fehlerbehandlung	974
	Anhang A: Glossar	981
	Anhang B: Wichtige Dateieindungen	985
	Index	987

Vorwort

C# ist die momentan von Microsoft propagierte Sprache, sie bietet die Möglichkeiten und Flexibilität von C++ und erlaubt trotzdem eine schnelle und unkomplizierte Programmierpraxis wie einst bei Visual Basic. C# ist (fast) genauso mächtig wie C++, wurde aber komplett neu auf objektorientierter Basis geschrieben.

Mit .NET 6 hat Microsoft jetzt auch die neueste Version von .NET veröffentlicht, die endlich die bereits für .NET 5 geplante, aber aufgrund der Covid-Pandemie verschobene Vereinheitlichung *One .NET* sowie weitere neue Anwendungsarten (z. B. *.NET MAUI*, auch wenn das jetzt auf das 2. Quartal 2022 verschoben wurde) und weitere Verbesserungen, vor allem im Performancebereich, für die C#-Entwickler enthält.

Damit ist C# das ideale Werkzeug zum Programmieren beliebiger Komponenten für .NET, beginnend bei WPF-, ASP.NET- und mobilen Anwendungen (auch für Android und iOS) bis hin zu systemnahen Applikationen.

Das vorliegende Buch ist ein Angebot für angehende wie auch für fortgeschrittene C#-Programmierer. Seine Philosophie knüpft an die vielen anderen Titel an, die in dieser Reihe in den vergangenen zwanzig Jahren zu verschiedenen Programmiersprachen erschienen sind:

- Programmieren lernt man nicht durch lineares Durcharbeiten eines Lehrbuchs, sondern nur durch unermüdliches Ausprobieren von Beispielen, verbunden mit ständigem Nachschlagen in der Referenz.
- Der Umfang einer modernen Sprache wie C# in Verbindung mit Visual Studio ist so gewaltig, dass ein seriöses Programmierbuch das Prinzip der Vollständigkeit aufgeben muss und nach dem Prinzip „so viel wie nötig“ sich lediglich eine „Initialisierungsfunktion“ auf die Fahnen schreiben kann.

Gegenüber anderen Büchern zur gleichen oder ähnlichen Thematik nimmt dieses Buch für sich in Anspruch, gleichzeitig Lehr- und Übungsbuch zu sein.

Zum Buchinhalt

Dieses Buch wagt den Spagat zwischen einem Grundlagen- und einem Profibuch. Sinn eines solchen Buches kann es nicht sein, eine umfassende Schritt-für-Schritt-Einführung in C# 10.0 zu liefern oder all die Informationen noch einmal zur Verfügung zu stellen, die Sie in der Produktdokumentation ohnehin schon finden und von denen Sie in der Regel nur ein Mausklick oder die F1-Taste trennt.

- Für den *Einsteiger* möchten wir den einzig vernünftigen und gangbaren Weg beschreiten, nämlich nach dem Prinzip „so viel wie nötig“ eine schmale Schneise durch den Urwald der .NET-Programmierung mit C# schlagen, bis er eine Lichtung erreicht hat, die ihm erste Erfolgserlebnisse vermittelt.
- Für den *Profi* möchten wir in diesem Buch eine Vielzahl von Informationen und Know-how bereitstellen, wonach er bisher in den mitgelieferten Dokumentationen, im Internet bzw. in anderen Büchern vergeblich gesucht hat.

Die Kapitel des Buches sind in vier Themenkomplexe gruppiert; online gibt es noch umfangreiches Zusatzmaterial, das auch immer wieder ergänzt wird:

1. Grundlagen der Programmierung mit C# (Buch)

2. Desktop-Anwendungen (Buch)

3. Technologien (Buch)

4. Webtechnologien (Buch)

5. Windows-Forms-Anwendungen (online)

6. Zugriff auf das Dateisystem (online)

7. ADO.NET (online)

8. XML (online)

Die Kapitel innerhalb eines Teils bilden einerseits eine logische Abfolge, können andererseits aber auch quergelesen werden. Im Praxisteil eines jeden Kapitels werden anhand realer Problemstellungen die behandelten Programmier Techniken im Zusammenhang demonstriert.

Updates und Aktualität

Damit Sie möglichst lange mit diesem Buch arbeiten können, haben Sie die Möglichkeit, sich für den kostenlosen Update inside-Service zu registrieren: Geben Sie unter

www.hanserfachbuch.de/Csharp-update

diesen Code ein:

5UKz-TaNroZp-mR5n

Dann erhalten Sie bis März 2024 Aktualisierungen in Form zusätzlicher Kapitel als PDF. Darin stellen wir Ihnen wichtige Neuerungen vor und gehen auf Änderungen ein, die die Inhalte dieses Buches betreffen.

Nobody is perfect

Sie werden in diesem Buch nicht alles finden, was C# bzw. .NET 6 zu bieten haben. Manches ist sicher in einem anderen Spezialtitel ausführlicher beschrieben. Aber Sie halten mit diesem Buch einen überschaubaren Breitband-Mix in den Händen, der sowohl vertikal vom Einsteiger bis zum Profi als auch horizontal von den einfachen Sprachelementen bis hin zu komplexen Anwendungen jedem etwas bietet, ohne dabei den Blick auf das Wesentliche im .NET-Dschungel zu verlieren.

Wenn Sie Vorschläge oder Fragen zum Buch haben, können Sie uns gerne kontaktieren:

juergen.kotz@primetime-software.de

info@christianwenz.de

Wir hoffen, dass wir Ihnen mit diesem Buch einen nützlichen Begleiter bei der .NET-Programmierung zur Seite gestellt haben, der es verdient, seinen Platz nicht im Regal, sondern griffbereit neben dem Computer einzunehmen.

Vielen Dank!

Ein Buch – gerade, wenn es so umfangreich ist – ist stets Teamarbeit. Herzlichen Dank an unsere Lektorin Sylvia Hasselbach, mit der wir seit über 20 Jahren zusammenarbeiten dürfen, sowie an Kristin Rothe und Irene Weilhart. Walter Saumweber hat als Fachlektor seit mehreren Auflagen den Röntgenblick für Ungenauigkeiten und Fehler (sollten wir trotzdem auf weitere Errata aufmerksam gemacht werden, veröffentlichen wir diese auf der Buchseite auf www.hanserfachbuch.de und auf <https://plus.hanserfachbuch.de> bei den Codebeispielen). Lenny Kotz hat uns bei einigen der Grafiken im Buch unterstützt.

Jürgen Kotz und Christian Wenz

München, im Januar 2022

■ Zusatzmaterial online

Der Einfachheit halber werden im ersten Teil die Beispiele weiter mit Windows Forms oder als Konsolenanwendungen erstellt. Der zweite Teil des Buches behandelt dann ausführlich WPF sowie .NET MAUI und im dritten Teil „Technologien“ werden die Beispiele dann mit WPF dargestellt. Der vierte Teil ist dann der Webtechnologie vorbehalten. Alle Beispieldaten dieses Buches und die mittlerweile zahlreichen Bonuskapitel können Sie online herunterladen.

Geben Sie auf

<https://plus.hanserfachbuch.de>

diesen Code ein:

```
plus-43qr9-pKvN2
```

Beim Nachvollziehen der Buchbeispiele beachten Sie bitte Folgendes:

- Kopieren Sie die Buchbeispiele auf die Festplatte. Wenn Sie auf die Projektmappendatei (*.sln) klicken, wird Visual Studio in der Regel automatisch geöffnet und das jeweilige Beispiel wird in die Entwicklungsumgebung geladen, wo Sie es z. B. mittels der F5-Taste kompilieren und starten können.
- Für einige Beispiele ist ein installierter Microsoft SQL Server Express LocalDB oder jegliche andere Instanz eines SQL-Servers erforderlich.
- Beachten Sie die zu einigen Beispielen beigefügten *Liesmich.txt*-Dateien, die Sie auf besondere Probleme hinweisen.

Teil I: Grundlagen



- .NET 6 (Kapitel 1)
- Einstieg in Visual Studio 2022 (Kapitel 2)
- Grundlagen der Sprache C# (Kapitel 3)
- OOP-Konzepte (Kapitel 4)
- Arrays, Strings, Funktionen (Kapitel 5)
- Weitere Sprachfeatures (Kapitel 6)
- Einführung in LINQ (Kapitel 7)
- Neuerungen von C# im Überblick (Kapitel 8)

1

.NET 6

Mit .NET 6 ist im November 2021 die aktuellste Version des Microsoft Frameworks, das mittlerweile auf eine fast 20 jährige Geschichte zurückblickt, erschienen. Und es handelt sich dabei um eine Version, welche das gesamte .NET-Universum wieder vereint, es gibt somit wieder ein „one“ .NET.

Mit dem .NET Framework können völlig unterschiedliche Applikationstypen (Desktop, Web, Mobile) in unterschiedlichen Programmiersprachen (C#, VB, F# etc.) entwickelt werden und egal was man macht, man hat eine einheitliche Funktionsbibliothek, eben das .NET Framework.

Im Jahre 2016 erschien jedoch parallel zum .NET Framework ein sogenanntes .NET Core Framework, das speziell für plattformübergreifende Webentwicklung konzipiert war. Keine Bindung somit mehr an Windows, sondern zeitgemäße plattformübergreifende Möglichkeiten (auch für mobile Plattformen wie iOS und Android), um seine Applikationen zu entwickeln und auch zu deployen. Da es sich bei .NET Core um Open Source handelte, hat Microsoft hier auch bewiesen, dass Open Source für Microsoft nicht nur ein Lippenbekenntnis ist, sondern ein wesentliches Fundament in der weiteren Entwicklung des Softwaregiganten.

Eigentlich war die Wiedervereinigung bereits 2020 mit der Version 5.0 geplant, jedoch musste diese dann aufgrund der Coronapandemie um ein Jahr verschoben werden.

Somit wurden erst mit .NET 6 diese beiden Entwicklungsstränge (.NET und .NET Core) zusammengeführt und man hat wieder ein einheitliches Framework – laut Microsoft auch das performanteste .NET –, um seine Anwendungen zu schreiben. Nur die Entwicklung von Desktopapplikationen mit Windows Forms oder WPF ist noch an Windows gebunden, alle anderen Anwendungsmöglichkeiten sind plattformübergreifend.

■ 1.1 Microsofts .NET-Technologie

Ganz ohne Theorie geht nichts! In diesem leider etwas „trockenen“ Abschnitt werden Einsteiger mit der grundlegenden .NET-Philosophie und den damit verbundenen Konzepten, Begriffen und Features vertraut gemacht. Dazu dürfen Sie Ihrem Rechner ruhig einmal eine Pause gönnen.

1.1.1 Zur Geschichte von .NET

Das Kürzel .NET ist die Bezeichnung für eine gemeinsame Plattform für viele Programmiersprachen. Beim Kompilieren von .NET-Programmen wird der jeweilige Quelltext in MSIL (*Microsoft Intermediate Language*) übersetzt. Es gibt nur ein gemeinsames Laufzeitsystem für alle Sprachen, die sogenannte CLR (Common Language Runtime), die die MSIL-Programme ausführt.

Die im Jahr 2002 eingeführte .NET-Technologie wurde notwendig, weil sich die Anforderungen an moderne Softwareentwicklung dramatisch verändert hatten, wobei das rasant wachsende Internet mit seinen hohen Ansprüchen an die Skalierbarkeit einer Anwendung, die Verteilbarkeit auf mehrere Schichten und ausreichende Sicherheit der hauptsächlichste Motor war, sich nach einer grundlegend neuen Sprachkonzeption umzuschauen.

Mit .NET fand ein radikaler Umbruch in der Geschichte der Softwareentwicklung bei Microsoft statt. Nicht nur dass jetzt „echte“ objektorientierte Programmierung zum obersten Dogma erhoben wird, nein, auch eine langjährig bewährte Sprache wie das alte Visual Basic wurde völlig umgekrempelt und die einst gelobte COM-Technologie wurde zum Auslaufmodell erklärt!



HINWEIS: Dem unerfahrenen Leser mögen einige Ausführungen im Laufe dieses Kapitels noch sehr fremd sein. Machen Sie sich deswegen keine Sorgen und wenn Sie etwas nicht verstehen, kommen Sie einfach am Ende des Buches nochmals hierher zurück.

Warum eine extra Programmiersprache?

C# wurde ausschließlich für das .NET Framework konzipiert, wobei versucht wurde, das Beste aus den etablierten Programmiersprachen Java, JavaScript, Visual Basic und C++ zu kombinieren, ohne aber deren Nachteile zu übernehmen.

Da sich die etablierten Sprachen nicht ohne größere Kompromisse an das .NET Framework anpassen ließen, haben die .NET-Entwickler die Gelegenheit beim Schopf gepackt und eine „maßgeschneiderte“ .NET-Sprache entwickelt. C# setzt auf dem .NET Framework auf und kommt ohne „faule“ Kompatibilitätskompromisse aus, wie sie z. B. teilweise bei Visual Basic erforderlich waren.