

C# und .NET 8

Grundlagen, Profiwissen und Rezepte

DAS INHALTS- VERZEICHNIS

» Hier geht's
direkt
zum Buch

Inhalt

Vorwort	XXIII
Zusatzmaterial online	XXV
Teil I: Grundlagen	1
1 .NET 8	3
1.1 Microsofts .NET-Technologie	4
1.1.1 Zur Geschichte von .NET	4
1.1.2 .NET-Features und Begriffe	7
1.2 .NET Core	14
1.2.1 Geschichte von .NET Core	14
1.2.2 LTS - Long Term Support und zukünftige Versionen	16
1.2.3 .NET Standard	16
1.3 Features von .NET 6	17
1.4 Features von .NET 7	18
1.5 Features von .NET 8	19
2 Einstieg in Visual Studio 2022	21
2.1 Die Installation von Visual Studio 2022	21
2.1.1 Überblick über die Produktpalette	22
2.1.2 Anforderungen an Hard- und Software	23
2.2 Unser allererstes C#-Programm	23
2.2.1 Vorbereitungen	23
2.2.2 Quellcode schreiben	27
2.2.3 Programm kompilieren und testen	27
2.2.4 Einige Erläuterungen zum Quellcode	28
2.2.5 Konsolenanwendungen sind out	29
2.3 Die Windows-Philosophie	29
2.3.1 Mensch-Rechner-Dialog	30
2.3.2 Objekt- und ereignisorientierte Programmierung	30
2.3.3 Programmieren mit Visual Studio 2022	31

2.4	Die Entwicklungsumgebung Visual Studio 2022	33
2.4.1	Neues Projekt	33
2.4.2	Die wichtigsten Fenster	36
2.4.3	Projektvorlagen in Visual Studio 2022 – Minimal APIs	39
2.5	Praxisbeispiele	41
2.5.1	Unsere erste Windows-Forms-Anwendung	41
2.5.2	Umrechnung Euro-Dollar	46
2.6	Neuerungen in Visual Studio 2022 Version 17.8	55
3	Grundlagen der Sprache C#	57
3.1	Grundbegriffe	57
3.1.1	Anweisungen	57
3.1.2	Bezeichner	58
3.1.3	Schlüsselwörter	59
3.1.4	Kommentare	60
3.2	Datentypen, Variablen und Konstanten	61
3.2.1	Fundamentale Typen	61
3.2.2	Werttypen versus Verweistypen	62
3.2.3	Benennung von Variablen	63
3.2.4	Deklaration von Variablen	63
3.2.5	Typsuffixe	64
3.2.6	Zeichen und Zeichenketten	65
3.2.7	object-Datentyp	67
3.2.8	Konstanten deklarieren	68
3.2.9	Nullable Types	68
3.2.10	Typinferenz	70
3.2.11	Gültigkeitsbereiche und Sichtbarkeit	70
3.3	Konvertieren von Datentypen	71
3.3.1	Implizite und explizite Konvertierung	71
3.3.2	Welcher Datentyp passt zu welchem?	73
3.3.3	Konvertieren von string	74
3.3.4	Die Convert-Klasse	75
3.3.5	Die Parse-Methode	76
3.3.6	Boxing und Unboxing	77
3.4	Operatoren	78
3.4.1	Arithmetische Operatoren	79
3.4.2	Zuweisungsoperatoren	81
3.4.3	Logische Operatoren	82
3.4.4	Rangfolge der Operatoren	85
3.5	Kontrollstrukturen	86
3.5.1	Verzweigungsbefehle	86
3.5.2	Schleifenanweisungen	91
3.6	Benutzerdefinierte Datentypen	93
3.6.1	Enumerationen	94
3.6.2	Strukturen	95

3.7	Nutzerdefinierte Methoden	97
3.7.1	Methoden mit Rückgabewert	98
3.7.2	Methoden ohne Rückgabewert	99
3.7.3	Parameterübergabe mit ref	101
3.7.4	Parameterübergabe mit out	102
3.7.5	Methodenüberladung	103
3.7.6	Optionale Parameter	104
3.7.7	Benannte Parameter	105
3.8	Praxisbeispiele	106
3.8.1	Vom PAP zur Konsolenanwendung	106
3.8.2	Ein Konsolen- in ein Windows-Programm verwandeln	109
3.8.3	Schleifenanweisungen verstehen	111
3.8.4	Benutzerdefinierte Methoden überladen	114
3.8.5	Anwendungen von Visual Basic nach C# portieren	117
4	OOP-Konzepte	125
4.1	Kleine Einführung in die OOP	125
4.1.1	Historische Entwicklung	126
4.1.2	Grundbegriffe der OOP	127
4.1.3	Sichtbarkeit von Klassen und ihren Mitgliedern	129
4.1.4	Allgemeiner Aufbau einer Klasse	130
4.1.5	Das Erzeugen eines Objekts	132
4.1.6	Einführungsbeispiel	135
4.2	Eigenschaften	141
4.2.1	Eigenschaften mit Zugriffsmethoden kapseln	141
4.2.2	Berechnete Eigenschaften	143
4.2.3	Lese-/Schreibschutz	145
4.2.4	Property-Accessoren	146
4.2.5	Statische Felder/Eigenschaften	147
4.2.6	Einfache Eigenschaften automatisch implementieren	149
4.3	Methoden	151
4.3.1	Öffentliche und private Methoden	151
4.3.2	Überladene Methoden	152
4.3.3	Statische Methoden	153
4.4	Ereignisse	154
4.4.1	Ereignis hinzufügen	155
4.4.2	Ereignis verwenden	158
4.5	Arbeiten mit Konstruktor und Destruktor	161
4.5.1	Konstruktor und Objektinitialisierer	162
4.5.2	Destruktor und Garbage Collector	165
4.5.3	Mit using den Lebenszyklus des Objekts kapseln	167
4.6	Vererbung und Polymorphie	168
4.6.1	Method-Overriding	168
4.6.2	Klassen implementieren	168

4.6.3	Implementieren der Objekte	171
4.6.4	Ausblenden von Mitgliedern durch Vererbung	173
4.6.5	Allgemeine Hinweise und Regeln zur Vererbung	174
4.6.6	Polymorphes Verhalten	176
4.6.7	Die Rolle von System.Object	179
4.7	Spezielle Klassen	180
4.7.1	Abstrakte Klassen	180
4.7.2	Versiegelte Klassen	181
4.7.3	Partielle Klassen	182
4.7.4	Statische Klassen	183
4.8	Schnittstellen (Interfaces)	184
4.8.1	Definition einer Schnittstelle	185
4.8.2	Implementieren einer Schnittstelle	185
4.8.3	Abfragen, ob Schnittstelle vorhanden ist	186
4.8.4	Mehrere Schnittstellen implementieren	187
4.8.5	Schnittstellenprogrammierung ist ein weites Feld ...	187
4.9	Datensatztypen – Records	187
4.9.1	Definition eines Record	188
4.9.2	Mutable Properties	190
4.9.3	Nicht-destruktive Änderung	192
4.9.4	Dekonstruktion	193
4.10	Praxisbeispiele	194
4.10.1	Eigenschaften sinnvoll kapseln	194
4.10.2	Eine statische Klasse anwenden	197
4.10.3	Vom fetten zum schlanken Client	198
4.10.4	Schnittstellenvererbung verstehen	209
4.10.5	Rechner für komplexe Zahlen	214
4.10.6	Sortieren mit IComparable/IComparer	222
4.10.7	Einen Objektbaum in generischen Listen abspeichern	227
4.10.8	OOP beim Kartenspiel erlernen	232
4.10.9	Eine Klasse zur Matrizenrechnung entwickeln	237
4.10.10	Vererbung von Records	243
5	Arrays, Strings, Funktionen	245
5.1	Datenfelder (Arrays)	245
5.1.1	Array deklarieren	246
5.1.2	Array instanziiieren	246
5.1.3	Array initialisieren	247
5.1.4	Zugriff auf Array-Elemente	248
5.1.5	Zugriff mittels Schleife	249
5.1.6	Mehrdimensionale Arrays	250
5.1.7	Zuweisen von Arrays	252
5.1.8	Arrays aus Strukturvariablen	253
5.1.9	Löschen und Umdimensionieren von Arrays	254

5.1.10	Eigenschaften und Methoden von Arrays	256
5.1.11	Übergabe von Arrays	257
5.2	Verarbeiten von Zeichenketten	259
5.2.1	Zuweisen von Strings	259
5.2.2	Eigenschaften und Methoden von String-Variablen	260
5.2.3	Wichtige Methoden der String-Klasse	262
5.2.4	Die StringBuilder-Klasse	264
5.3	Datums- und Zeitberechnungen	267
5.3.1	Die DateTime-Struktur	267
5.3.2	Wichtige Eigenschaften von DateTime-Variablen	268
5.3.3	Wichtige Methoden von DateTime-Variablen	269
5.3.4	Wichtige Mitglieder der DateTime-Struktur	270
5.3.5	Konvertieren von Datumstrings in DateTime-Werte	271
5.3.6	Die TimeSpan-Struktur	271
5.3.7	DateOnly und TimeOnly	273
5.4	Mathematische Funktionen	274
5.4.1	Überblick	274
5.4.2	Zahlen runden	274
5.4.3	Winkel umrechnen	275
5.4.4	Potenz- und Wurzeloperationen	275
5.4.5	Logarithmus und Exponentialfunktionen	275
5.4.6	Zufallszahlen erzeugen	276
5.4.7	Kreisberechnung	277
5.5	Zahlen- und Datumsformatierungen	277
5.5.1	Anwenden der ToString-Methode	278
5.5.2	Anwenden der Format-Methode	279
5.5.3	Stringinterpolation	280
5.6	Praxisbeispiele	281
5.6.1	Zeichenketten verarbeiten	281
5.6.2	Zeichenketten mit StringBuilder addieren	284
5.6.3	Methodenaufrufe mit Array-Parametern	287
6	Weitere Sprachfeatures	293
6.1	Namespaces (Namensräume)	293
6.1.1	Ein kleiner Überblick	293
6.1.2	Einen eigenen Namespace einrichten	294
6.1.3	Die using-Anweisung	296
6.1.4	Namespace Alias	297
6.1.5	Globale using-Anweisungen	297
6.2	Operatorenüberladung	298
6.2.1	Syntaxregeln	298
6.2.2	Praktische Anwendung	299
6.3	Collections (Auflistungen)	300
6.3.1	Die Schnittstelle IEnumerable	300

6.3.2	ArrayList	303
6.3.3	Hashtable	304
6.3.4	Indexer	305
6.4	Generics	307
6.4.1	Generics bieten Typsicherheit	308
6.4.2	Generische Methoden	309
6.4.3	yield - Iteratoren	310
6.5	Generische Collections	310
6.5.1	List-Collection statt ArrayList	310
6.5.2	Vorteile generischer Collections	312
6.5.3	Constraints	312
6.6	Das Prinzip der Delegates	313
6.6.1	Delegates sind Methodenzeiger	313
6.6.2	Einen Delegate-Typ deklarieren	314
6.6.3	Ein Delegate-Objekt erzeugen	314
6.6.4	Anonyme Methoden	316
6.6.5	Lambda-Ausdrücke	317
6.6.6	Lambda-Ausdrücke in der Task Parallel Library	320
6.6.7	Action<> und Func<>	321
6.7	Dynamische Programmierung	323
6.7.1	Wozu dynamische Programmierung?	323
6.7.2	Das Prinzip der dynamischen Programmierung	324
6.7.3	Optionale Parameter sind hilfreich	326
6.7.4	Kovarianz und Kontravarianz	327
6.8	Weitere Datentypen	328
6.8.1	BigInteger	328
6.8.2	Complex	330
6.8.3	Tuple<>	331
6.8.4	SortedSet<>	332
6.9	Praxisbeispiele	333
6.9.1	ArrayList versus generische List	333
6.9.2	Generische IEnumerable-Interfaces implementieren	336
6.9.3	Delegates, Func, anonyme Methoden, Lambda Expressions	340
7	Einführung in LINQ	345
7.1	LINQ-Grundlagen	345
7.1.1	Die LINQ-Architektur	345
7.1.2	Anonyme Typen	347
7.1.3	Erweiterungsmethoden	348
7.2	Abfragen mit LINQ to Objects	349
7.2.1	Grundlegendes zur LINQ-Syntax	350
7.2.2	Zwei alternative Schreibweisen von LINQ-Abfragen	351
7.2.3	Übersicht der wichtigsten Abfrageoperatoren	352

7.3	LINQ-Abfragen im Detail	353
7.3.1	Die Projektionsoperatoren Select und SelectMany	354
7.3.2	Der Restriktionsoperator Where	355
7.3.3	Die Sortierungsoperatoren OrderBy und ThenBy	356
7.3.4	Der Gruppierungsoperator GroupBy	357
7.3.5	Verknüpfen mit Join	360
7.3.6	Aggregat-Operatoren	360
7.3.7	Verzögertes Ausführen von LINQ-Abfragen	362
7.3.8	Konvertierungsmethoden	363
7.3.9	Abfragen mit PLINQ	364
7.4	Praxisbeispiele	367
7.4.1	Die Syntax von LINQ-Abfragen verstehen	367
7.4.2	Aggregat-Abfragen mit LINQ	370
7.4.3	LINQ im Schnelldurchgang erlernen	373
7.4.4	Strings mit LINQ abfragen und filtern	375
7.4.5	Duplikate aus einer Liste entfernen	377
7.4.6	Arrays mit LINQ initialisieren	379
7.4.7	Arrays per LINQ mit Zufallszahlen füllen	381
7.4.8	Einen String mit Wiederholmuster erzeugen	383
7.4.9	Mit LINQ Zahlen und Strings sortieren	384
7.4.10	Mit LINQ Collections von Objekten sortieren	386
7.4.11	Where - Deep Dive	388
8	Neuerungen von C# im Überblick	397
8.1	C# 4.0 - Visual Studio 2010	398
8.1.1	Datentyp dynamic	398
8.1.2	Benannte und optionale Parameter	399
8.1.3	Kovarianz und Kontravarianz	400
8.2	C# 5.0 - Visual Studio 2012	400
8.2.1	Async und Await	401
8.2.2	CallerInfo	402
8.3	Visual Studio 2013	403
8.4	C# 6.0 - Visual Studio 2015	403
8.4.1	String Interpolation	403
8.4.2	Schreibgeschützte AutoProperties	403
8.4.3	Initialisierer für AutoProperties	404
8.4.4	Expression Body Member	404
8.4.5	using static	404
8.4.6	Bedingter Nulloperator	405
8.4.7	Ausnahmenfilter	405
8.4.8	nameof-Ausdrücke	406
8.4.9	await in catch und finally	406
8.4.10	Indexinitialisierer	406
8.5	C# 7.0 - Visual Studio 2017	407

8.5.1	out-Variablen	407
8.5.2	Tupel	407
8.5.3	Mustervergleich	408
8.5.4	Discards	410
8.5.5	Lokale ref-Variablen und Rückgabetypen	411
8.5.6	Lokale Funktionen	411
8.5.7	Mehr Expression Body Member	411
8.5.8	throw-Ausdrücke	412
8.5.9	Verbesserung der numerischen literalen Syntax	412
8.6	C# 7.1 bis 7.3 – Visual Studio 2019	412
8.6.1	C# 7.1	412
8.6.2	C# 7.2	414
8.6.3	C# 7.3	415
8.6.4	Visual Studio 2019 – Live Share	415
8.7	C# 8.0	418
8.7.1	Standardschnittstellenmethoden	418
8.7.2	Vereinfachung von switch-Ausdrücken	420
8.7.3	Eigenschaftenmuster	421
8.7.4	Vereinfachte using-Ressourcenschutzblöcke	421
8.7.5	Null-Coalescing-Zuweisungen	422
8.7.6	Nullable Referenztypen	423
8.7.7	Indizes und Bereiche	423
8.7.8	Weitere Sprachneuerungen	425
8.8	C# 9.0	425
8.8.1	Records	425
8.8.2	Init-only Setter	426
8.8.3	Weitere Verbesserungen in Musterausdrücken	426
8.8.4	Weitere Sprachneuerungen	426
8.9	C# 10	427
8.9.1	Datensatzstrukturen	427
8.9.2	Globale using-Anweisungen	427
8.9.3	Weitere Sprachneuerungen	427
8.10	C# 11	428
8.10.1	Raw String Literals	428
8.10.2	Generische Mathematik	430
8.10.3	Generische Attribute	431
8.10.4	Listenmuster	431
8.10.5	Lokale Dateitypen	432
8.10.6	Required Member	433
8.10.7	Automatische Standardstrukturen	434
8.11	C# 12	434
8.11.1	Primäre Konstruktoren auch für Klassen und Strukturen	434
8.11.2	Samlungsausdrücke	435
8.11.3	Weitere Sprachneuerungen	435

Teil II: Desktop-Anwendungen	437
9 Einführung in WPF	439
9.1 Einführung	439
9.1.1 Was kann eine WPF-Anwendung?	440
9.1.2 Die eXtensible Application Markup Language	442
9.1.3 Unsere erste XAML-Anwendung	443
9.1.4 Zielplattformen	449
9.1.5 Applikationstypen	449
9.1.6 Vor- und Nachteile von WPF-Anwendungen	450
9.1.7 Weitere Dateien im Überblick	451
9.2 Alles beginnt mit dem Layout	453
9.2.1 Allgemeines zum Layout	453
9.2.2 Positionieren von Steuerelementen	455
9.2.3 Canvas	458
9.2.4 StackPanel	459
9.2.5 DockPanel	461
9.2.6 WrapPanel	463
9.2.7 UniformGrid	464
9.2.8 Grid	465
9.2.9 ViewBox	470
9.2.10 TextBlock	471
9.3 Das WPF-Programm	475
9.3.1 Die App-Klasse	475
9.3.2 Das Startobjekt festlegen	475
9.3.3 Kommandozeilenparameter verarbeiten	477
9.3.4 Die Anwendung beenden	478
9.3.5 Auswerten von Anwendungsereignissen	478
9.4 Die Window-Klasse	479
9.4.1 Position und Größe festlegen	480
9.4.2 Rahmen und Beschriftung	480
9.4.3 Das Fenster-Icon ändern	480
9.4.4 Anzeige weiterer Fenster	481
9.4.5 Transparenz	481
9.4.6 Abstand zum Inhalt festlegen	482
9.4.7 Fenster ohne Fokus anzeigen	482
9.4.8 Ereignisfolge bei Fenstern	483
9.4.9 Ein paar Worte zur Schriftdarstellung	483
9.4.10 Ein paar Worte zur Darstellung von Controls	486
9.4.11 Wird mein Fenster komplett mit WPF gerendert?	487

10	Übersicht WPF-Controls	489
10.1	Allgemeingültige Eigenschaften	489
10.2	Label	491
10.3	Button, RepeatButton, ToggleButton	492
10.3.1	Schaltflächen für modale Dialoge	493
10.3.2	Schaltflächen mit Grafik	494
10.4	TextBox, PasswordBox	495
10.4.1	TextBox	495
10.4.2	PasswordBox	497
10.5	CheckBox	498
10.6	RadioButton	500
10.7	ListBox, ComboBox	501
10.7.1	ListBox	502
10.7.2	ComboBox	505
10.7.3	Den Content formatieren	506
10.8	Image	508
10.8.1	Grafik per XAML zuweisen	508
10.8.2	Grafik zur Laufzeit zuweisen	508
10.8.3	Bild aus Datei laden	510
10.8.4	Die Grafikskalierung beeinflussen	511
10.9	Slider, ScrollBar	512
10.9.1	Slider	512
10.9.2	ScrollBar	514
10.10	ScrollViewer	514
10.11	Menu, ContextMenu	515
10.11.1	Menu	516
10.11.2	Tastenkürzel	517
10.11.3	Grafiken	518
10.11.4	Weitere Möglichkeiten	519
10.11.5	ContextMenu	520
10.12	ToolBar	521
10.13	StatusBar, ProgressBar	524
10.13.1	StatusBar	524
10.13.2	ProgressBar	526
10.14	Border, GroupBox, BulletDecorator	526
10.14.1	Border	527
10.14.2	GroupBox	528
10.14.3	BulletDecorator	529
10.15	Expander, TabControl	531
10.15.1	Expander	531
10.15.2	TabControl	533
10.16	Popup	534

10.17	TreeView	537
10.18	ListView	541
10.19	DataGrid	541
10.20	Calendar/DatePicker	542
10.21	Ellipse, Rectangle, Line und Co.	546
10.21.1	Ellipse	547
10.21.2	Rectangle	547
10.21.3	Line	548
11	Wichtige WPF-Techniken	549
11.1	Eigenschaften	549
11.1.1	Abhängige Eigenschaften (Dependency Properties)	549
11.1.2	Angehängte Eigenschaften (Attached Properties)	551
11.2	Einsatz von Ressourcen	551
11.2.1	Was sind eigentlich Ressourcen?	551
11.2.2	Wo können Ressourcen gespeichert werden?	552
11.2.3	Wie definiere ich eine Ressource?	553
11.2.4	Statische und dynamische Ressourcen	554
11.2.5	Wie werden Ressourcen adressiert?	556
11.2.6	Systemressourcen einbinden	557
11.3	Das WPF-Ereignismodell	557
11.3.1	Einführung	557
11.3.2	Routed Events	558
11.3.3	Direkte Events	561
11.4	Verwendung von Commands	561
11.4.1	Einführung zu Commands	561
11.4.2	Verwendung vordefinierter Commands	562
11.4.3	Das Ziel des Commands	564
11.4.4	Vordefinierte Commands	565
11.4.5	Commands an Ereignismethoden binden	565
11.4.6	Wie kann ich ein Command per Code auslösen?	566
11.4.7	Command-Ausführung verhindern	567
11.5	Das WPF-Style-System	567
11.5.1	Übersicht	567
11.5.2	Benannte Styles	568
11.5.3	Typ-Styles	570
11.5.4	Styles anpassen und vererben	571
11.6	Verwenden von Triggern	573
11.6.1	Eigenschaften-Trigger (Property Triggers)	574
11.6.2	Ereignis-Trigger	576
11.6.3	Daten-Trigger	577
11.7	Einsatz von Templates	578
11.7.1	Neues Template erstellen	578
11.7.2	Template abrufen und verändern	582

11.8	Transformationen, Animationen, StoryBoards	585
11.8.1	Transformationen	585
11.8.2	Animationen mit dem StoryBoard realisieren	590
12	WPF-Datenbindung	597
12.1	Grundprinzip	597
12.1.1	Bindungsarten	598
12.1.2	Wann eigentlich wird die Quelle aktualisiert?	600
12.1.3	Geht es auch etwas langsamer?	601
12.1.4	Bindung zur Laufzeit realisieren	602
12.2	Binden an Objekte	603
12.2.1	Objekte im XAML-Code instanziiieren	604
12.2.2	Verwenden der Instanz im C#-Quellcode	605
12.2.3	Anforderungen an die Quell-Klasse	606
12.2.4	Instanziiieren von Objekten per C#-Code	607
12.3	Binden von Collections	609
12.3.1	Anforderung an die Collection	609
12.3.2	Einfache Anzeige	610
12.3.3	Navigieren zwischen den Objekten	611
12.3.4	Einfache Anzeige in einer ListBox	613
12.3.5	DataTemplates zur Anzeigeformatierung	614
12.3.6	Mehr zu List- und ComboBox	615
12.3.7	Verwendung der ListView	617
12.4	Noch einmal zurück zu den Details	620
12.4.1	Navigieren in den Daten	620
12.4.2	Sortieren	622
12.4.3	Filtern	622
12.4.4	Live Shaping	623
12.5	Anzeige von Datenbankinhalten	625
12.5.1	Installieren der benötigten NuGet-Pakete	625
12.5.2	Anlegen der Entitätsklassen	627
12.5.3	Die Programmoberfläche	630
12.5.4	Der Zugriff auf die Daten	632
12.6	Formatieren von Werten	634
12.6.1	IValueConverter	634
12.6.2	BindingBase.StringFormat-Eigenschaft	636
12.7	Das DataGrid als Universalwerkzeug	637
12.7.1	Grundlagen der Anzeige	637
12.7.2	UI-Virtualisierung	639
12.7.3	Spalten selbst definieren	639
12.7.4	Zusatzinformationen in den Zeilen anzeigen	641
12.7.5	Vom Betrachten zum Editieren	643
12.8	Praxisbeispiel - Collections in Hintergrundthreads füllen	643

13	.NET MAUI	649
13.1	Einführung	650
13.2	Was kann eine .NET MAUI-Anwendung?	651
13.3	Die erste .NET MAUI-App	652
13.4	Definieren einer eigenen View	661
13.5	Daten in MAUI anzeigen	663
13.6	Styles in MAUI	667
13.6.1	Styles	667
13.6.2	Themes	668
13.7	Navigation unter MAUI	670
Teil III: Technologien		673
14	Asynchrone Programmierung	675
14.1	Übersicht	676
14.1.1	Multitasking versus Multithreading	676
14.1.2	Deadlocks	677
14.1.3	Racing	678
14.2	Programmieren mit Threads	679
14.2.1	Einführungsbeispiel	679
14.2.2	Wichtige Thread-Methoden	681
14.2.3	Wichtige Thread-Eigenschaften	683
14.2.4	Einsatz der ThreadPool-Klasse	684
14.3	Sperrmechanismen	685
14.3.1	Threading ohne lock	686
14.3.2	Threading mit lock	687
14.3.3	Die Monitor-Klasse	690
14.3.4	Mutex	694
14.3.5	Methoden für die parallele Ausführung sperren	695
14.3.6	Semaphore	696
14.4	Interaktion mit der Programmoberfläche	698
14.4.1	Die Werkzeuge	699
14.4.2	Einzelne Steuerelemente mit Invoke aktualisieren (Windows Forms)	699
14.4.3	Mehrere Steuerelemente aktualisieren	701
14.4.4	Ist ein Invoke-Aufruf nötig?	701
14.4.5	Und was ist mit WPF?	702
14.5	Timer-Threads	704
14.6	Asynchrone Programmierentwurfsmuster	705
14.6.1	Kurzübersicht	705
14.6.2	Polling	706
14.6.3	Callback verwenden	708
14.6.4	Callback mit Parameterübergabe verwenden	709
14.6.5	Callback mit Zugriff auf die Programmoberfläche	710

14.7	Es geht auch einfacher – async und await	712
14.7.1	Der Weg von synchron zu asynchron	712
14.7.2	Achtung: Fehlerquellen!	714
14.7.3	Eigene asynchrone Methoden entwickeln	717
14.8	Asynchrone Streams	719
14.8.1	Datei erstellen	720
14.8.2	Datei lesen mit <code>IAsyncEnumerable<T></code>	721
14.9	Praxisbeispiele	722
14.9.1	Prozess- und Thread-Informationen gewinnen	722
14.9.2	Ein externes Programm starten	725
15	Die Task Parallel Library	729
15.1	Überblick	729
15.1.1	Parallel-Programmierung	729
15.1.2	Möglichkeiten der TPL	732
15.1.3	Der CLR-Threadpool	733
15.2	Parallele Verarbeitung mit <code>Parallel.Invoke</code>	734
15.2.1	Aufrufvarianten	734
15.2.2	Einschränkungen	736
15.3	Verwendung von <code>Parallel.For</code>	737
15.3.1	Abbrechen der Verarbeitung	738
15.3.2	Auswerten des Verarbeitungsstatus	740
15.3.3	Und was ist mit anderen Iterator-Schrittweiten?	741
15.4	<code>Collections</code> mit <code>Parallel.ForEach</code> verarbeiten	741
15.5	Die Task-Klasse	742
15.5.1	Einen Task erzeugen	742
15.5.2	Den Task starten	743
15.5.3	Datenübergabe an den Task	745
15.5.4	Wie warte ich auf das Ende des Tasks?	746
15.5.5	Tasks mit Rückgabewerten	748
15.5.6	Die Verarbeitung abbrechen	751
15.5.7	Fehlerbehandlung	756
15.5.8	Weitere Eigenschaften	757
15.6	Zugriff auf das User Interface	758
15.6.1	Task-Ende und Zugriff auf die Oberfläche	758
15.6.2	Zugriff auf das UI aus dem Task heraus	760
15.7	Weitere Datenstrukturen im Überblick	762
15.7.1	Threadsichere <code>Collections</code>	762
15.7.2	Primitive für die Threadsynchronisation	762
15.8	Parallel LINQ (PLINQ)	763
15.9	Praxisbeispiele	763
15.9.1	<code>BlockingCollection</code>	763
15.9.2	PLINQ	767

16	Debugging, Fehlersuche und Fehlerbehandlung	769
16.1	Der Debugger	769
16.1.1	Allgemeine Beschreibung	769
16.1.2	Die wichtigsten Fenster	770
16.1.3	Debugging-Optionen	774
16.1.4	Praktisches Debugging am Beispiel	777
16.2	Arbeiten mit Debug und Trace	782
16.2.1	Wichtige Methoden von Debug und Trace	782
16.2.2	Besonderheiten der Trace-Klasse	786
16.2.3	TraceListener-Objekte	786
16.3	Caller Information	788
16.3.1	Attribute	789
16.3.2	Anwendung	789
16.4	Fehlerbehandlung	790
16.4.1	Anweisungen zur Fehlerbehandlung	790
16.4.2	try-catch	791
16.4.3	try-finally	795
16.4.4	Das Standardverhalten bei Ausnahmen festlegen	797
16.4.5	Die Exception-Klasse	798
16.4.6	Fehler/Ausnahmen auslösen	799
16.4.7	Eigene Fehlerklassen	799
16.4.8	Exceptionhandling zur Debugzeit	801
17	Entity Framework Core	803
17.1	Überblick	803
17.1.1	Objektrelationaler Mapper (ORM)	804
17.1.2	Provider	806
17.1.3	Relationale Beziehungen	807
17.1.4	Benötigte NuGet-Pakete	810
17.2	CodeFirst	812
17.2.1	CodeFirst aus Model	812
17.2.2	CodeFirst mittels ReverseEngineering von bestehender Datenbank	822
17.3	Migrationen	827
17.3.1	Initiale Migration bei ReverseEngineering	827
17.3.2	Weitere Migrationen	828
17.4	Lesen und Schreiben von Daten mit EF Core	831
17.5	Neuerungen in Entity Framework Core 7 und 8	835
17.5.1	JSON-Spalten	835
17.5.2	Bulk Updates	840
17.5.3	SaveChanges()	841
17.5.4	Mapping von Stored Procedures	841
17.5.5	Optimiertes SQL für Contains-Abfragen	842
17.5.6	Enums werden innerhalb von JSON-Spalten als ints gespeichert	842
17.5.7	Primitive Collections	842

17.6	Serialisierung mit System.Text.Json	845
17.7	Praxisbeispiele	850
17.7.1	Daten mit EF Core laden und als JSON speichern	850
17.7.2	Eine Datenbank mit EF Core anlegen und Testdaten generieren und anzeigen	857
Teil IV: Webanwendungen		865
18 Webanwendungen mit ASP.NET Core		867
18.1	Grundlagen	868
18.2	Razor Pages	871
18.2.1	Projektaufbau	872
18.2.2	Razor-Syntax	875
18.2.3	Layout-Vorlagen	881
18.2.4	Modelle für Razor Pages	884
18.2.5	Mit Formularen arbeiten	886
18.3	MVC	893
18.3.1	Projektaufbau	894
18.3.2	Action-Methoden	895
18.3.3	Zustandsmanagement	906
18.4	Praxisbeispiele	911
18.4.1	CRUD mit Entity Framework	911
18.4.2	Authentifizierung und Autorisierung	928
18.4.3	Zugriffsbeschränkung	935
19 Web APIs mit ASP.NET Core		939
19.1	REST	939
19.2	Vorlagen	943
19.3	Daten lesen	949
19.4	Daten schreiben, aktualisieren, löschen	958
19.5	Minimale APIs	968
19.6	Praxisbeispiele	973
19.6.1	Paginierung	973
19.6.2	XML statt JSON	975
19.6.3	CORS	976
20 Blazor		981
20.1	Hosting-Modelle	982
20.2	Projektvorlagen	985
20.3	Blazor-Komponenten	993
20.3.1	Code in/für Komponenten	993
20.3.2	Event-Handling	997
20.3.3	Datenbindung	1000

20.4	Services und APIs aufrufen	1003
20.5	Weitere Blazor-Features	1008
20.5.1	Zustandsmanagement	1008
20.5.2	JavaScript-Interoperabilität	1011
20.5.3	Render-Modi und Streaming	1015
20.6	Praxisbeispiele	1017
20.6.1	Online-Status ermitteln	1017
20.6.2	File-Uploads	1026
20.6.3	Fehlerbehandlung	1028
20.6.4	Datengrid	1033
Anhang A: Glossar		1035
Anhang B: Wichtige Dateiendungen		1039
Index		1041