

C# und .NET 10

Grundlagen, Profiwissen und Rezepte

DAS INHALTS- VERZEICHNIS

» Hier geht's
direkt
zum Buch

Inhalt

Vorwort	XXVII
Zusatzmaterial online	XXX
Teil I: Grundlagen	1
1 .NET 10	3
1.1 Microsofts .NET-Technologie	4
1.1.1 Zur Geschichte von .NET	4
1.1.2 .NET-Features und Begriffe	8
1.2 .NET Core	16
1.2.1 Geschichte von .NET Core	16
1.2.2 LTS – Long Term Support und zukünftige Versionen	18
1.2.3 .NET Standard	18
1.3 Features von .NET 6	19
1.4 Features von .NET 7	20
1.5 Features von .NET 8	21
1.6 Features von .NET 9	23
1.7 Features von .NET 10	23
2 Einstieg in Visual Studio 2026	25
2.1 Die Installation von Visual Studio 2026	25
2.1.1 Überblick über die Produktpalette	26
2.1.2 Anforderungen an Hard- und Software	27

2.2	Unser allererstes C#-Programm	28
2.2.1	Vorbereitungen	28
2.2.2	Quellcode schreiben	31
2.2.3	Programm kompilieren und testen	32
2.2.4	Einige Erläuterungen zum Quellcode	32
2.2.5	Konsolenanwendungen sind out	33
2.3	Die Windows-Philosophie	34
2.3.1	Mensch-Rechner-Dialog	34
2.3.2	Objekt- und ereignisorientierte Programmierung	35
2.3.3	Programmieren mit Visual Studio 2026	36
2.4	Die Entwicklungsumgebung Visual Studio 2026	37
2.4.1	Neues Projekt	38
2.4.2	Die wichtigsten Fenster	41
2.4.3	Projektvorlagen in Visual Studio 2026 – Minimal APIs	45
2.5	Praxisbeispiele	46
2.5.1	Unsere erste Windows-Forms-Anwendung	46
2.5.2	Umrechnung Euro-Dollar	52
2.6	Neuerungen in Visual Studio 2022 Version 17.8	62
2.7	Die Highlights der neuen Version Visual Studio 2026	62
3	Grundlagen der Sprache C#	65
3.1	Grundbegriffe	65
3.1.1	Anweisungen	65
3.1.2	Bezeichner	66
3.1.3	Schlüsselwörter	67
3.1.4	Kommentare	68
3.2	Datentypen, Variablen und Konstanten	69
3.2.1	Fundamentale Typen	69
3.2.2	Wertetypen versus Verweistypen	71
3.2.3	Benennung von Variablen	71
3.2.4	Deklaration von Variablen	72
3.2.5	Typsuffixe	73
3.2.6	Zeichen und Zeichenketten	74
3.2.7	object-Datentyp	76
3.2.8	Konstanten deklarieren	77

3.2.9	Nullable Types	78
3.2.10	Typinferenz	79
3.2.11	Gültigkeitsbereiche und Sichtbarkeit	80
3.3	Konvertieren von Datentypen	81
3.3.1	Implizite und explizite Konvertierung	81
3.3.2	Welcher Datentyp passt zu welchem?	83
3.3.3	Konvertieren von string	84
3.3.4	Die Convert-Klasse	86
3.3.5	Die Parse-Methode	87
3.3.6	Boxing und Unboxing	87
3.4	Operatoren	89
3.4.1	Arithmetische Operatoren	90
3.4.2	Zuweisungsoperatoren	92
3.4.3	Logische Operatoren	93
3.4.4	Rangfolge der Operatoren	96
3.5	Kontrollstrukturen	98
3.5.1	Verzweigungsbefehle	98
3.5.2	Schleifenanweisungen	103
3.6	Benutzerdefinierte Datentypen	105
3.6.1	Enumerationen	106
3.6.2	Strukturen	107
3.7	Nutzerdefinierte Methoden	110
3.7.1	Methoden mit Rückgabewert	111
3.7.2	Methoden ohne Rückgabewert	112
3.7.3	Parameterübergabe mit ref	113
3.7.4	Parameterübergabe mit out	115
3.7.5	Methodenüberladung	116
3.7.6	Optionale Parameter	117
3.7.7	Benannte Parameter	118
3.8	Praxisbeispiele	119
3.8.1	Vom PAP zur Konsolenanwendung	119
3.8.2	Ein Konsolen- in ein Windows-Programm verwandeln	122
3.8.3	Schleifenanweisungen verstehen	124
3.8.4	Benutzerdefinierte Methoden überladen	127
3.8.5	Anwendungen von Visual Basic nach C# portieren	131

- 4 OOP-Konzepte 139**
 - 4.1 Kleine Einführung in die OOP 139
 - 4.1.1 Historische Entwicklung 140
 - 4.1.2 Grundbegriffe der OOP 142
 - 4.1.3 Sichtbarkeit von Klassen und ihren Mitgliedern 144
 - 4.1.4 Allgemeiner Aufbau einer Klasse 145
 - 4.1.5 Das Erzeugen eines Objekts 147
 - 4.1.6 Einführungsbeispiel 151
 - 4.2 Eigenschaften 157
 - 4.2.1 Eigenschaften mit Zugriffsmethoden kapseln 157
 - 4.2.2 Berechnete Eigenschaften 160
 - 4.2.3 Lese-/Schreibschutz 161
 - 4.2.4 Property-Accessoren 162
 - 4.2.5 Statische Felder/Eigenschaften 163
 - 4.2.6 Einfache Eigenschaften automatisch implementieren 166
 - 4.3 Methoden 167
 - 4.3.1 Öffentliche und private Methoden 168
 - 4.3.2 Überladene Methoden 169
 - 4.3.3 Statische Methoden 169
 - 4.4 Ereignisse 171
 - 4.4.1 Ereignis hinzufügen 172
 - 4.4.2 Ereignis verwenden 175
 - 4.5 Arbeiten mit Konstruktor und Destruktor 179
 - 4.5.1 Konstruktor und Objektinitialisierer 180
 - 4.5.2 Destruktor und Garbage Collector 183
 - 4.5.3 Mit using den Lebenszyklus des Objekts kapseln 186
 - 4.6 Vererbung und Polymorphie 187
 - 4.6.1 Method-Overriding 187
 - 4.6.2 Klassen implementieren 187
 - 4.6.3 Implementieren der Objekte 191
 - 4.6.4 Ausblenden von Mitgliedern durch Vererbung 192
 - 4.6.5 Allgemeine Hinweise und Regeln zur Vererbung 194
 - 4.6.6 Polymorphes Verhalten 196
 - 4.6.7 Die Rolle von System.Object 198

4.7	Spezielle Klassen	200
4.7.1	Abstrakte Klassen	200
4.7.2	Versiegelte Klassen	201
4.7.3	Partielle Klassen	202
4.7.4	Statische Klassen	204
4.8	Schnittstellen (Interfaces)	204
4.8.1	Definition einer Schnittstelle	205
4.8.2	Implementieren einer Schnittstelle	206
4.8.3	Abfragen, ob Schnittstelle vorhanden ist	207
4.8.4	Mehrere Schnittstellen implementieren	207
4.8.5	Schnittstellenprogrammierung ist ein weites Feld	208
4.9	Datensatztypen – Records	208
4.9.1	Definition eines Record	209
4.9.2	Mutable Properties	212
4.9.3	Nicht-destruktive Änderung	214
4.9.4	Dekonstruktion	215
4.10	Praxisbeispiele	216
4.10.1	Eigenschaften sinnvoll kapseln	216
4.10.2	Eine statische Klasse anwenden	219
4.10.3	Vom fetten zum schlanken Client	221
4.10.4	Schnittstellenvererbung verstehen	232
4.10.5	Rechner für komplexe Zahlen	237
4.10.6	Sortieren mit Comparable/Comparer	246
4.10.7	Einen Objektbaum in generischen Listen abspeichern	251
4.10.8	OOP beim Kartenspiel erlernen	257
4.10.9	Eine Klasse zur Matrizenrechnung entwickeln	262
4.10.10	Vererbung von Records	268
5	Arrays, Strings, Methoden	271
5.1	Datenfelder (Arrays)	271
5.1.1	Array deklarieren	272
5.1.2	Array instanziiieren	273
5.1.3	Array initialisieren	273
5.1.4	Zugriff auf Array-Elemente	275
5.1.5	Zugriff mittels Schleife	276

5.1.6	Mehrdimensionale Arrays	277
5.1.7	Zuweisen von Arrays	279
5.1.8	Arrays aus Strukturvariablen	281
5.1.9	Löschen und Umdimensionieren von Arrays	282
5.1.10	Eigenschaften und Methoden von Arrays	284
5.1.11	Übergabe von Arrays	286
5.2	Verarbeiten von Zeichenketten	287
5.2.1	Zuweisen von Strings	288
5.2.2	Eigenschaften und Methoden von String-Variablen	288
5.2.3	Wichtige Methoden der String-Klasse	291
5.2.4	Die StringBuilder-Klasse	294
5.3	Datums- und Zeitberechnungen	297
5.3.1	Die DateTime-Struktur	297
5.3.2	Wichtige Eigenschaften von DateTime-Variablen	299
5.3.3	Wichtige Methoden von DateTime-Variablen	300
5.3.4	Wichtige Mitglieder der DateTime-Struktur	300
5.3.5	Konvertieren von Datumstrings in DateTime-Werte	301
5.3.6	Die TimeSpan-Struktur	302
5.3.7	DateOnly und TimeOnly	304
5.4	Mathematische Methoden	305
5.4.1	Überblick	305
5.4.2	Zahlen runden	306
5.4.3	Winkel umrechnen	306
5.4.4	Potenz- und Wurzeloperationen	306
5.4.5	Logarithmus und Exponentialfunktionen	306
5.4.6	Zufallszahlen erzeugen	307
5.4.7	Kreisberechnung	308
5.5	Zahlen- und Datumsformatierungen	308
5.5.1	Anwenden der ToString-Methode	309
5.5.2	Anwenden der Format-Methode	311
5.5.3	Stringinterpolation	312
5.6	Praxisbeispiele	313
5.6.1	Zeichenketten verarbeiten	313
5.6.2	Zeichenketten mit StringBuilder addieren	316
5.6.3	Methodenaufrufe mit Array-Parametern	320

6	Weitere Sprachfeatures	325
6.1	Namespaces (Namensräume)	325
6.1.1	Ein kleiner Überblick	325
6.1.2	Einen eigenen Namespace einrichten	327
6.1.3	Die using-Anweisung	329
6.1.4	Namespace Alias	330
6.1.5	Globale using-Anweisungen	331
6.2	Operatorenüberladung	332
6.2.1	Syntaxregeln	333
6.2.2	Praktische Anwendung	333
6.3	Collections (Auflistungen)	334
6.3.1	Die Schnittstelle IEnumerable	335
6.3.2	ArrayList	338
6.3.3	Hashtable	339
6.3.4	Indexer	340
6.4	Generics	342
6.4.1	Generics bieten Typsicherheit	343
6.4.2	Generische Methoden	344
6.4.3	yield – Iteratoren	345
6.5	Generische Collections	346
6.5.1	List-Collection statt ArrayList	346
6.5.2	Vorteile generischer Collections	347
6.5.3	Constraints	348
6.6	Das Prinzip der Delegates	348
6.6.1	Delegates sind Methodenzeiger	349
6.6.2	Einen Delegate-Typ deklarieren	349
6.6.3	Ein Delegate-Objekt erzeugen	350
6.6.4	Anonyme Methoden	352
6.6.5	Lambda-Ausdrücke	353
6.6.6	Lambda-Ausdrücke in der Task Parallel Library	356
6.6.7	Action<> und Func<>	358
6.7	Dynamische Programmierung	360
6.7.1	Wozu dynamische Programmierung?	360
6.7.2	Das Prinzip der dynamischen Programmierung	361

- 6.7.3 Optionale Parameter sind hilfreich 363
 - 6.7.4 Kovarianz und Kontravarianz 364
- 6.8 Lokale Funktionen 365
- 6.9 Weitere Datentypen 368
 - 6.9.1 BigInteger 368
 - 6.9.2 Complex 371
 - 6.9.3 Tuple<> 371
 - 6.9.4 SortedSet<> 372
- 6.10 Praxisbeispiele 374
 - 6.10.1 ArrayList versus generische List 374
 - 6.10.2 Generische IEnumerable-Interfaces implementieren 377
 - 6.10.3 Delegates, Func, anonyme Methoden, Lambda Expressions 381
- 7 Einführung in LINQ 387**
 - 7.1 LINQ-Grundlagen 387
 - 7.1.1 Die LINQ-Architektur 387
 - 7.1.2 Anonyme Typen 389
 - 7.1.3 Erweiterungsmethoden 390
 - 7.2 Abfragen mit LINQ to Objects 392
 - 7.2.1 Grundlegendes zur LINQ-Syntax 392
 - 7.2.2 Zwei alternative Schreibweisen von LINQ-Abfragen 394
 - 7.2.3 Übersicht der wichtigsten Abfrageoperatoren 395
 - 7.3 LINQ-Abfragen im Detail 396
 - 7.3.1 Die Projektionsoperatoren Select und SelectMany 397
 - 7.3.2 Der Restriktionsoperator Where 399
 - 7.3.3 Die Sortierungsoperatoren OrderBy und ThenBy 399
 - 7.3.4 Der Gruppierungsoperator GroupBy 401
 - 7.3.5 Verknüpfen mit Join 404
 - 7.3.6 Aggregat-Operatoren 405
 - 7.3.7 Verzögertes Ausführen von LINQ-Abfragen 407
 - 7.3.8 Konvertierungsmethoden 408
 - 7.3.9 Abfragen mit PLINQ 409
 - 7.4 Praxisbeispiele 413
 - 7.4.1 Die Syntax von LINQ-Abfragen verstehen 413
 - 7.4.2 Aggregat-Abfragen mit LINQ 416

7.4.3	LINQ im Schnelldurchgang erlernen	419
7.4.4	Strings mit LINQ abfragen und filtern	421
7.4.5	Duplikate aus einer Liste entfernen	423
7.4.6	Arrays mit LINQ initialisieren	426
7.4.7	Arrays per LINQ mit Zufallszahlen füllen	428
7.4.8	Einen String mit Wiederholmuster erzeugen	429
7.4.9	Mit LINQ Zahlen und Strings sortieren	431
7.4.10	Mit LINQ Collections von Objekten sortieren	432
7.4.11	Where – Deep Dive	435
8	Neuerungen von C# im Überblick	445
8.1	C# 4.0 – Visual Studio 2010	445
8.1.1	Datentyp dynamic	445
8.1.2	Benannte und optionale Parameter	446
8.1.3	Kovarianz und Kontravarianz	448
8.2	C# 5.0 – Visual Studio 2012	448
8.2.1	Async und Await	448
8.2.2	CallerInfo	449
8.3	Visual Studio 2013	450
8.4	C# 6.0 – Visual Studio 2015	451
8.4.1	String Interpolation	451
8.4.2	Schreibgeschützte AutoProperties	451
8.4.3	Initialisierer für AutoProperties	452
8.4.4	Expression Body Member	452
8.4.5	using static	452
8.4.6	Bedingter Nulloperator	453
8.4.7	Ausnahmenfilter	454
8.4.8	nameof-Ausdrücke	454
8.4.9	await in catch und finally	455
8.4.10	Indexinitialisierer	455
8.5	C# 7.0 – Visual Studio 2017	455
8.5.1	out-Variablen	455
8.5.2	Tupel	456
8.5.3	Mustervergleich	457
8.5.4	Discards	458

8.5.5	Lokale ref-Variablen und Rückgabetypen	459
8.5.6	Lokale Funktionen	459
8.5.7	Mehr Expression Body Member	460
8.5.8	throw-Ausdrücke	460
8.5.9	Verbesserung der numerischen literalen Syntax	460
8.6	C# 7.1 bis 7.3 – Visual Studio 2019	461
8.6.1	C# 7.1	461
8.6.1.1	Asynchrone Main-Einstiegsmethode	461
8.6.1.2	Literale Standardausdrücke	462
8.6.1.3	Vereinfachter Aufruf von Tupeln	462
8.6.2	C# 7.2	463
8.6.3	C# 7.3	463
8.7	C# 8.0	464
8.7.1	Standardschnittstellenmethoden	464
8.7.2	Vereinfachung von switch-Ausdrücken	466
8.7.3	Eigenschaftenmuster	468
8.7.4	Vereinfachte using-Ressourcenschutzblöcke	468
8.7.5	Null-Coalescing-Zuweisungen	469
8.7.6	Nullable Referenztypen	470
8.7.7	Indizes und Bereiche	471
8.7.8	Weitere Sprachneuerungen	472
8.8	C# 9.0	473
8.8.1	Records	473
8.8.2	Init-only Setter	473
8.8.3	Weitere Verbesserungen in Musterausdrücken	474
8.8.4	Weitere Sprachneuerungen	474
8.9	C# 10	474
8.9.1	Datensatzstrukturen	475
8.9.2	Globale using-Anweisungen	475
8.9.3	Weitere Sprachneuerungen	475
8.10	C# 11	476
8.10.1	Raw String Literals	476
8.10.2	Generische Mathematik	478
8.10.3	Generische Attribute	479
8.10.4	Listenmuster	480

8.10.5	Lokale Dateitypen	481
8.10.6	Required Member	481
8.10.7	Automatische Standardstrukturen	482
8.11	C# 12	482
8.11.1	Primäre Konstruktoren auch für Klassen und Strukturen	482
8.11.2	Sammlungsausdrücke	483
8.11.3	Weitere Sprachneuerungen	484
8.12	C# 13	484
8.12.1	params auch für Collections möglich	485
8.12.2	Impliziter Indexzugriff	486
8.12.3	Weitere partielle Member	487
8.13	C# 14	488
8.13.1	Erweiterungsmitglieder	488
8.13.2	field	490
8.13.3	Weitere partielle Member	491
8.13.4	Null-bedingte Zuweisung	491
Teil II: Desktop-Anwendungen		493
9 Einführung in WPF		495
9.1	Einführung	496
9.1.1	Was kann eine WPF-Anwendung?	496
9.1.2	Die eXtensible Application Markup Language	498
9.1.3	Unsere erste XAML-Anwendung	500
9.1.4	Zielpattformen	505
9.1.5	Applikationstypen	506
9.1.6	Vor- und Nachteile von WPF-Anwendungen	507
9.1.7	Weitere Dateien im Überblick	508
9.2	Alles beginnt mit dem Layout	510
9.2.1	Allgemeines zum Layout	511
9.2.2	Positionieren von Steuerelementen	513
9.2.3	Canvas	516
9.2.4	StackPanel	517
9.2.5	DockPanel	519
9.2.6	WrapPanel	521
9.2.7	UniformGrid	522

- 9.2.8 Grid 523
 - 9.2.9 ViewBox 529
 - 9.2.10 TextBlock 531
- 9.3 Das WPF-Programm 535
 - 9.3.1 Die App-Klasse 535
 - 9.3.2 Das Startobjekt festlegen 535
 - 9.3.3 Kommandozeilenparameter verarbeiten 537
 - 9.3.4 Die Anwendung beenden 539
 - 9.3.5 Auswerten von Anwendungsereignissen 539
- 9.4 Die Window-Klasse 541
 - 9.4.1 Position und Größe festlegen 541
 - 9.4.2 Rahmen und Beschriftung 541
 - 9.4.3 Das Fenster-Icon ändern 542
 - 9.4.4 Anzeige weiterer Fenster 542
 - 9.4.5 Transparenz 543
 - 9.4.6 Abstand zum Inhalt festlegen 544
 - 9.4.7 Fenster ohne Fokus anzeigen 544
 - 9.4.8 Ereignisfolge bei Fenstern 545
 - 9.4.9 Ein paar Worte zur Schriftdarstellung 545
 - 9.4.10 Ein paar Worte zur Darstellung von Controls 548
 - 9.4.11 Wird mein Fenster komplett mit WPF gerendert? 550
- 10 Übersicht WPF-Controls 551**
 - 10.1 Allgemeingültige Eigenschaften 551
 - 10.2 Label 554
 - 10.3 Button, RepeatButton, ToggleButton 554
 - 10.3.1 Schaltflächen für modale Dialoge 555
 - 10.3.2 Schaltflächen mit Grafik 557
 - 10.4 TextBox, PasswordBox 558
 - 10.4.1 TextBox 558
 - 10.4.2 PasswordBox 560
 - 10.5 CheckBox 561
 - 10.6 RadioButton 563
 - 10.7 ListBox, ComboBox 564
 - 10.7.1 ListBox 565

10.7.2	ComboBox	568
10.7.3	Den Content formatieren	570
10.8	Image	571
10.8.1	Grafik per XAML zuweisen	572
10.8.2	Grafik zur Laufzeit zuweisen	572
10.8.3	Bild aus Datei laden	574
10.8.4	Die Grafikskalierung beeinflussen	575
10.9	Slider, ScrollBar	576
10.9.1	Slider	576
10.9.2	ScrollBar	578
10.10	ScrollViewer	578
10.11	Menu, ContextMenu	580
10.11.1	Menu	580
10.11.2	Tastenkürzel	581
10.11.3	Grafiken	582
10.11.4	Weitere Möglichkeiten	584
10.11.5	ContextMenu	585
10.12	ToolBar	586
10.13	StatusBar, ProgressBar	589
10.13.1	StatusBar	589
10.13.2	ProgressBar	591
10.14	Border, GroupBox, BulletDecorator	592
10.14.1	Border	592
10.14.2	GroupBox	593
10.14.3	BulletDecorator	595
10.15	Expander, TabControl	597
10.15.1	Expander	597
10.15.2	TabControl	598
10.16	Popup	600
10.17	TreeView	603
10.18	ListView	606
10.19	DataGrid	607
10.20	Calendar/DatePicker	608

- 10.21 Ellipse, Rectangle, Line und Co. 612
 - 10.21.1 Ellipse 613
 - 10.21.2 Rectangle 614
 - 10.21.3 Line 614
- 11 Wichtige WPF-Techniken 615**
 - 11.1 Eigenschaften 615
 - 11.1.1 Abhängige Eigenschaften (Dependency Properties) 615
 - 11.1.2 Angehängte Eigenschaften (Attached Properties) 617
 - 11.2 Einsatz von Ressourcen 617
 - 11.2.1 Was sind eigentlich Ressourcen? 618
 - 11.2.2 Wo können Ressourcen gespeichert werden? 618
 - 11.2.3 Wie definiere ich eine Ressource? 619
 - 11.2.4 Statische und dynamische Ressourcen 621
 - 11.2.5 Wie werden Ressourcen adressiert? 622
 - 11.2.6 Systemressourcen einbinden 623
 - 11.3 Das WPF-Ereignismodell 623
 - 11.3.1 Einführung 624
 - 11.3.2 Routed Events 625
 - 11.3.3 Direkte Events 627
 - 11.4 Verwendung von Commands 627
 - 11.4.1 Einführung zu Commands 628
 - 11.4.2 Verwendung vordefinierter Commands 628
 - 11.4.3 Das Ziel des Commands 630
 - 11.4.4 Vordefinierte Commands 631
 - 11.4.5 Commands an Ereignismethoden binden 632
 - 11.4.6 Wie kann ich ein Command per Code auslösen? 633
 - 11.4.7 Command-Ausführung verhindern 634
 - 11.5 Das WPF-Style-System 634
 - 11.5.1 Übersicht 635
 - 11.5.2 Benannte Styles 635
 - 11.5.3 Typ-Styles 637
 - 11.5.4 Styles anpassen und vererben 639
 - 11.6 Verwenden von Triggern 641
 - 11.6.1 Eigenschaften-Trigger (Property Triggers) 642

11.6.2	Ereignis-Trigger	644
11.6.3	Daten-Trigger	645
11.7	Einsatz von Templates	646
11.7.1	Neues Template erstellen	647
11.7.2	Template abrufen und verändern	650
11.8	Transformationen, Animationen, StoryBoards	654
11.8.1	Transformationen	654
11.8.2	Animationen mit dem StoryBoard realisieren	659
12	WPF-Datenbindung	665
12.1	Grundprinzip	665
12.1.1	Bindungsarten	667
12.1.2	Wann eigentlich wird die Quelle aktualisiert?	668
12.1.3	Geht es auch etwas langsamer?	669
12.1.4	Bindung zur Laufzeit realisieren	670
12.2	Binden an Objekte	672
12.2.1	Objekte im XAML-Code instanziiieren	673
12.2.2	Verwenden der Instanz im C#-Quellcode	674
12.2.3	Anforderungen an die Quell-Klasse	675
12.2.4	Instanziiieren von Objekten per C#-Code	677
12.3	Binden von Collections	678
12.3.1	Anforderung an die Collection	678
12.3.2	Einfache Anzeige	679
12.3.3	Navigieren zwischen den Objekten	680
12.3.4	Einfache Anzeige in einer ListBox	682
12.3.5	DataTemplates zur Anzeigeformatierung	684
12.3.6	Mehr zu List- und ComboBox	685
12.3.7	Verwendung der ListView	687
12.4	Noch einmal zurück zu den Details	689
12.4.1	Navigieren in den Daten	690
12.4.2	Sortieren	691
12.4.3	Filtern	692
12.4.4	Live Shaping	693
12.5	Anzeige von Datenbankinhalten	695
12.5.1	Installieren der benötigten NuGet-Pakete	695

- 12.5.2 Anlegen der Entitätsklassen 697
 - 12.5.3 Die Programmoberfläche 700
 - 12.5.4 Der Zugriff auf die Daten 702
- 12.6 Formatieren von Werten 704
 - 12.6.1 IValueConverter 705
 - 12.6.2 BindingBase.StringFormat-Eigenschaft 707
- 12.7 Das DataGrid als Universalwerkzeug 707
 - 12.7.1 Grundlagen der Anzeige 708
 - 12.7.2 UI-Virtualisierung 709
 - 12.7.3 Spalten selbst definieren 710
 - 12.7.4 Zusatzinformationen in den Zeilen anzeigen 712
 - 12.7.5 Vom Betrachten zum Editieren 714
- 12.8 Praxisbeispiel – Collections in Hintergrundthreads füllen 714
- 13 .NET MAUI 719**
 - 13.1 Einführung 720
 - 13.2 Was kann eine .NET MAUI-Anwendung? 722
 - 13.3 Die erste .NET MAUI-App 723
 - 13.4 Definieren einer eigenen View 731
 - 13.5 Daten in MAUI anzeigen 734
 - 13.6 Styles in MAUI 738
 - 13.6.1 Styles 738
 - 13.6.2 Themes 739
 - 13.7 Navigation unter MAUI 742
- Teil III: Technologien 743**
- 14 Asynchrone Programmierung 745**
 - 14.1 Übersicht 746
 - 14.1.1 Multitasking versus Multithreading 746
 - 14.1.2 Deadlocks 747
 - 14.1.3 Racing 748
 - 14.2 Programmieren mit Threads 749
 - 14.2.1 Einführungsbeispiel 750
 - 14.2.2 Wichtige Thread-Methoden 751

14.2.3	Wichtige Thread-Eigenschaften	753
14.2.4	Einsatz der ThreadPool-Klasse	754
14.3	Sperrmechanismen	755
14.3.1	Threading ohne lock	756
14.3.2	Threading mit lock	758
14.3.3	Die Monitor-Klasse	760
14.3.4	Mutex	764
14.3.5	Methoden für die parallele Ausführung sperren	766
14.3.6	Semaphore	767
14.4	Interaktion mit der Programmoberfläche	769
14.4.1	Die Werkzeuge	769
14.4.2	Einzelne Steuerelemente mit Invoke aktualisieren (Windows Forms)	770
14.4.3	Mehrere Steuerelemente aktualisieren	771
14.4.4	Ist ein Invoke-Aufruf nötig?	772
14.4.5	Und was ist mit WPF?	772
14.5	Timer-Threads	774
14.6	Asynchrone Programmierentwurfsmuster	776
14.6.1	Kurzübersicht	776
14.6.2	Polling	777
14.6.3	Callback verwenden	779
14.6.4	Callback mit Parameterübergabe verwenden	780
14.6.5	Callback mit Zugriff auf die Programmoberfläche	782
14.7	Es geht auch einfacher – async und await	783
14.7.1	Der Weg von synchron zu asynchron	784
14.7.2	Achtung: Fehlerquellen!	786
14.7.3	Eigene asynchrone Methoden entwickeln	788
14.8	Asynchrone Streams	791
14.8.1	Datei erstellen	791
14.8.2	Datei lesen mit IEnumerable<T>	793
14.9	Praxisbeispiele	794
14.9.1	Prozess- und Thread-Informationen gewinnen	794
14.9.2	Ein externes Programm starten	797

- 15 Die Task Parallel Library 801**
 - 15.1 Überblick 801
 - 15.1.1 Parallel-Programmierung 801
 - 15.1.2 Möglichkeiten der TPL 804
 - 15.1.3 Der CLR-Threadpool 805
 - 15.2 Parallele Verarbeitung mit Parallel.Invoke 806
 - 15.2.1 Aufrufvarianten 807
 - 15.2.2 Einschränkungen 809
 - 15.3 Verwendung von Parallel.For 809
 - 15.3.1 Abbrechen der Verarbeitung 811
 - 15.3.2 Auswerten des Bearbeitungsstatus 813
 - 15.3.3 Und was ist mit anderen Iterator-Schrittweiten? 814
 - 15.4 Collections mit Parallel.ForEach verarbeiten 814
 - 15.5 Die Task-Klasse 815
 - 15.5.1 Einen Task erzeugen 816
 - 15.5.2 Den Task starten 817
 - 15.5.3 Datenübergabe an den Task 818
 - 15.5.4 Wie warte ich auf das Ende des Tasks? 820
 - 15.5.5 Tasks mit Rückgabewerten 822
 - 15.5.6 Die Verarbeitung abbrechen 826
 - 15.5.7 Fehlerbehandlung 831
 - 15.5.8 Weitere Eigenschaften 832
 - 15.6 Zugriff auf das User Interface 833
 - 15.6.1 Task-Ende und Zugriff auf die Oberfläche 833
 - 15.6.2 Zugriff auf das UI aus dem Task heraus 835
 - 15.7 Weitere Datenstrukturen im Überblick 837
 - 15.7.1 Threadsichere Collections 837
 - 15.7.2 Primitive für die Threadsynchronisation 838
 - 15.8 Parallel LINQ (PLINQ) 838
 - 15.9 Praxisbeispiele 839
 - 15.9.1 BlockingCollection 839
 - 15.9.2 PLINQ 843

16	Debugging, Fehlersuche und Fehlerbehandlung	845
16.1	Der Debugger	845
16.1.1	Allgemeine Beschreibung	846
16.1.2	Die wichtigsten Fenster	846
16.1.3	Debugging-Optionen	851
16.1.4	Praktisches Debugging am Beispiel	854
16.2	Arbeiten mit Debug und Trace	859
16.2.1	Wichtige Methoden von Debug und Trace	859
16.2.2	Besonderheiten der Trace-Klasse	864
16.2.3	TraceListener-Objekte	865
16.3	Caller Information	867
16.3.1	Attribute	867
16.3.2	Anwendung	868
16.4	Fehlerbehandlung	869
16.4.1	Anweisungen zur Fehlerbehandlung	869
16.4.2	try-catch	869
16.4.3	try-finally	875
16.4.4	Das Standardverhalten bei Ausnahmen festlegen	877
16.4.5	Die Exception-Klasse	878
16.4.6	Fehler/Ausnahmen auslösen	879
16.4.7	Eigene Fehlerklassen	880
16.4.8	Exceptionhandling zur Debugzeit	882
16.4.9	Hinweise, welche Exceptions in einer Methode auftreten können	883
17	Entity Framework Core	885
17.1	Überblick	885
17.1.1	Objektrelationaler Mapper (ORM)	886
17.1.2	Provider	888
17.1.3	Relationale Beziehungen	890
17.1.4	Benötigte NuGet-Pakete	893
17.2	Code First	895
17.2.1	Code First aus Model	895
17.2.2	Code First mittels Reverse Engineering von bestehender Datenbank	908

- 17.3 Migrationen 912
 - 17.3.1 Initiale Migration bei Reverse Engineering 912
 - 17.3.2 Weitere Migrationen 914
- 17.4 Lesen und Schreiben von Daten mit EF Core 917
- 17.5 Neuerungen im Entity Framework Core 922
 - 17.5.1 JSON-Spalten 922
 - 17.5.1.1 Erstellung des Modells mit einer JSON-Spalte 923
 - 17.5.1.2 Anlegen und Auslesen von Datensätzen mit JSON-Spalten 926
 - 17.5.2 Bulk Updates 928
 - 17.5.3 SaveChanges() 929
 - 17.5.4 Mapping von Stored Procedures 929
 - 17.5.5 Optimiertes SQL für Contains-Abfragen 930
 - 17.5.6 Enums werden innerhalb von JSON-Spalten als Integers gespeichert 930
 - 17.5.7 Primitive Collections 930
- 17.6 Serialisierung mit System.Text.Json 934
- 17.7 Praxisbeispiele 939
 - 17.7.1 Daten mit EF Core laden und als JSON speichern 939
 - 17.7.2 Eine Datenbank mit EF Core anlegen und Testdaten generieren und anzeigen 947
- 18 Künstliche Intelligenz mit .NET 955**
 - 18.1 Überblick 955
 - 18.1.1 Neuronale Netze 956
 - 18.1.2 Machine Learning 957
 - 18.1.3 Deep Learning 957
 - 18.1.4 Large Language Models – LLMs 958
 - 18.1.5 Generative AI 959
 - 18.1.6 ChatGPT als Beispiel für Generative AI 959
 - 18.2 Machine Learning mit ML.NET 961
 - 18.2.1 ML.NET Workflow 962
 - 18.2.2 Testdatensätze 963
 - 18.3 Praxisbeispiele 963
 - 18.3.1 Klassifizierungsbeispiel 964
 - 18.3.1.1 Datenaufbereitung 965
 - 18.3.1.2 Trainingsdaten einlesen 966

18.3.1.3 Modellanalyse	969
18.3.1.4 Modell nach Training speichern	974
18.3.1.5 Modell wieder laden und mit Testdaten evaluieren/ benutzen	975
18.3.2 Anomalieerkennung	984
18.3.2.1 Datenaufbereitung	984
18.3.3 Trainingsdaten einlesen	986
18.3.4 Training	987
18.3.5 Modellevaluierung	988
Teil IV: Webanwendungen	993
19 Webanwendungen mit ASP.NET Core	995
19.1 Grundlagen	996
19.2 Razor Pages	1000
19.2.1 Projektaufbau	1000
19.2.2 Razor-Syntax	1004
19.2.3 Layout-Vorlagen	1010
19.2.4 Modelle für Razor Pages	1015
19.2.5 Mit Formularen arbeiten	1016
19.3 MVC	1024
19.3.1 Projektaufbau	1025
19.3.2 Action-Methoden	1026
19.3.3 Zustandsmanagement	1040
19.4 Praxisbeispiele	1045
19.4.1 CRUD mit Entity Framework	1045
19.4.2 Authentifizierung und Autorisierung	1064
19.4.3 Zugriffsbeschränkung	1072
20 Web APIs mit ASP.NET Core	1075
20.1 REST	1075
20.2 Vorlagen	1080
20.3 Daten lesen	1088
20.4 Daten schreiben, aktualisieren, löschen	1097
20.5 Minimale APIs	1108
20.6 Praxisbeispiele	1113
20.6.1 Paginierung	1113

- 20.6.2 XML statt JSON 1115
 - 20.6.3 CORS 1117
- 21 Single-Page Applications mit Blazor 1123**
 - 21.1 Hosting-Modelle 1124
 - 21.2 Projektvorlagen 1128
 - 21.3 Blazor-Komponenten 1137
 - 21.3.1 Code in/für Komponenten 1137
 - 21.3.2 Event-Handling 1141
 - 21.3.3 Datenbindung 1145
 - 21.4 Services und APIs aufrufen 1148
 - 21.5 Weitere Blazor-Features 1154
 - 21.5.1 Zustandsmanagement 1155
 - 21.5.2 JavaScript-Interoperabilität 1157
 - 21.5.3 Render-Modi und Streaming 1162
 - 21.6 Praxisbeispiele 1165
 - 21.6.1 Online-Status ermitteln 1165
 - 21.6.2 File-Uploads 1174
 - 21.6.3 Fehlerbehandlung 1177
 - 21.6.4 Datengrid 1182
- 22 Deployment lokal und in der Cloud 1185**
 - 22.1 Deployment mit Visual Studio 1186
 - 22.1.1 Deployment in einen Ordner 1187
 - 22.1.2 Deployment in den IIS 1191
 - 22.2 Deployment in Docker 1195
 - 22.3 Konfiguration anpassen 1200
 - 22.4 Deployment in Azure 1203
 - 22.4.1 Hosting-Optionen 1204
 - 22.4.2 Deployment in Azure App Service 1205
- Anhang A: Glossar 1211**
- Anhang B: Wichtige Dateieindungen 1217**
- Index 1219**