

Auf einen Blick

1	Allgemeine Einführung in .NET	37
2	Grundlagen der Sprache C#	65
3	Das Klassendesign	145
4	Vererbung, Polymorphie und Interfaces	221
5	Delegaten, Ereignisse und Lambda-Ausdrücke	283
6	Strukturen und Enumerationen	319
7	Fehlerbehandlung und Debugging	331
8	Auflistungsklassen (Collections)	371
9	Generics – generische Datentypen	401
10	Weitere C#-Sprachfeatures	429
11	LINQ – Language Integrated Query	505
12	Arbeiten mit Dateien und Streams	539
13	Serialisierung	585
14	Multithreading	599
15	Die Task Parallel Library (TPL)	651
16	Grundlegende .NET-Klassen	679
17	Projektmanagement und Visual Studio 2019	721
18	Die Zukunft: .NET Core und .NET Standard	773
19	Einführung in das Entity Framework	789
20	Database First mit dem EDM-Designer	815
21	Entity Framework – Code First	907
22	Einführung in die WPF und XAML	957
23	Die WPF-Layoutcontainer	989
24	Fenster in der WPF	1009
25	WPF-Steuerelemente	1037
26	Dependency Properties	1099
27	Ereignisse in der WPF	1115
28	Ressourcen, Styles, Trigger und Templates	1135
29	WPF-Datenbindung	1177
30	WPF – weitergehende Techniken	1279
31	WPF-Commands	1307
32	Das MVVM-Pattern	1331
33	2D-Grafik	1385
34	Komponententests (Unit-Tests)	1405

Inhalt

Materialien zum Buch	32
Vorwort zur 8. Auflage	34

1 Allgemeine Einführung in .NET 37

1.1 Warum .NET?	37
1.1.1 Die Zukunft von .NET	39
1.1.2 Ein paar Worte zu diesem Buch	40
1.1.3 Die Beispielprogramme	42
1.2 .NET unter die Lupe genommen	43
1.2.1 Das Entwicklerdilemma	43
1.2.2 .NET – ein paar allgemeine Eigenschaften	44
1.2.3 Das Sprachenkonzept	45
1.2.4 Die Common Language Specification (CLS)	47
1.2.5 Das Common Type System (CTS)	48
1.2.6 Das .NET Framework	49
1.2.7 Die Common Language Runtime (CLR)	50
1.2.8 Die .NET-Klassenbibliothek	51
1.2.9 Das Konzept der Namespaces	52
1.3 Assemblies	53
1.3.1 Die Metadaten	54
1.3.2 Das Manifest	55
1.4 Die Entwicklungsumgebung	55
1.4.1 Editionen von Visual Studio 2019	56
1.4.2 Hard- und Softwareanforderungen	56
1.4.3 Die Installation	57
1.4.4 Die Entwicklungsumgebung von Visual Studio 2019	59

2 Grundlagen der Sprache C# 65

2.1 Konsolenanwendungen	65
2.1.1 Allgemeine Anmerkungen	65
2.1.2 Ein erstes Konsolenprogramm	65

2.2 Grundlagen der C#-Syntax	70
2.2.1 Kennzeichnen, dass eine Anweisung abgeschlossen ist	70
2.2.2 Anweisungs- und Gliederungsblöcke	71
2.2.3 Kommentare	72
2.2.4 Die Groß- und Kleinschreibung	73
2.2.5 Die Struktur einer Konsolenanwendung	73
2.3 Variablen und Datentypen	75
2.3.1 Variablendeklaration	75
2.3.2 Der Variablenbezeichner	76
2.3.3 Der Zugriff auf eine Variable	77
2.3.4 Ein- und Ausgabemethoden der Klasse »Console«	77
2.3.5 Die elementaren Datentypen von .NET	83
2.3.6 Ausgabe ganzzahliger Datentypen	89
2.3.7 Elementare Typkonvertierung	90
2.4 Operatoren	97
2.4.1 Arithmetische Operatoren	98
2.4.2 Vergleichsoperatoren	101
2.4.3 Logische Operatoren	102
2.4.4 Bitweise Operatoren	105
2.4.5 Zuweisungsoperatoren	108
2.4.6 Stringverkettung	108
2.4.7 Sonstige Operatoren	109
2.4.8 Operator-Vorrangregeln	109
2.5 Datenfelder (Arrays)	110
2.5.1 Die Deklaration und Initialisierung eines Arrays	110
2.5.2 Der Zugriff auf die Array-Elemente	112
2.5.3 Mehrdimensionale Arrays	113
2.5.4 Festlegen der Array-Größe zur Laufzeit	114
2.5.5 Bestimmung der Array-Obergrenze	116
2.5.6 Die Gesamtanzahl der Array-Elemente	116
2.5.7 Verzweigte Arrays	117
2.6 Kontrollstrukturen	118
2.6.1 Die »if«-Anweisung	119
2.6.2 Das »switch«-Statement	124
2.7 Programmschleifen	129
2.7.1 Die »for«-Schleife	129
2.7.2 Die »foreach«-Schleife	140
2.7.3 Die »do«- und die »while«-Schleife	141

3 Das Klassendesign	145
3.1 Einführung in die Objektorientierung	145
3.2 Die Klassendefinition	148
3.2.1 Klassen in Visual Studio anlegen	148
3.2.2 Das Projekt »GeometricObjectsSolution«	149
3.2.3 Die Deklaration von Objektvariablen	151
3.2.4 Zugriffsmodifizierer einer Klasse	152
3.2.5 Splitten einer Klassendefinition mit »partial«	152
3.2.6 Arbeiten mit Objektreferenzen	153
3.3 Referenz- und Wertetypen	155
3.3.1 Werte- und Referenztypen nutzen	156
3.4 Die Eigenschaften eines Objekts	157
3.4.1 Öffentliche Felder	157
3.4.2 Datenkapselung mit Eigenschaftsmethoden (Properties)	158
3.4.3 Auto-Properties (automatisch implementierte Properties)	162
3.4.4 Unterstützung von Visual Studio	164
3.5 Methoden eines Objekts	164
3.5.1 Methoden mit Rückgabewert	164
3.5.2 Methoden ohne Rückgabewert	168
3.5.3 Methoden mit Parameterliste	169
3.5.4 Methodenüberladung	171
3.5.5 Variablen innerhalb einer Methode (lokale Variablen)	173
3.5.6 Modifizierer eines Parameters	175
3.5.7 Besondere Aspekte einer Parameterliste	181
3.5.8 Zugriff auf private Daten	186
3.5.9 Die Trennung von Daten und Code	187
3.5.10 Namenskonflikte mit »this« lösen	188
3.5.11 Methode oder Eigenschaft?	189
3.5.12 Umbenennen von Methoden und Eigenschaften	191
3.6 Konstruktoren	191
3.6.1 Konstruktoren bereitstellen	192
3.6.2 Die Konstruktoraufrufe	193
3.6.3 Definition von Konstruktoren	194
3.6.4 »public«- und »internal«-Konstruktoren	194
3.6.5 »private«-Konstruktoren	195
3.6.6 Konstruktoraufrufe umleiten	195
3.6.7 Vereinfachte Objektinitialisierung	197
3.7 Der Destruktor	198

3.8	Konstanten in einer Klasse	199
3.8.1	Konstanten mit dem Schlüsselwort »const«	199
3.8.2	Schreibgeschützte Felder mit »readonly«	199
3.9	Statische Klassenkomponenten	200
3.9.1	Statische Eigenschaften	200
3.9.2	Statische Methoden	202
3.9.3	Statische Klasseninitialisierer	204
3.9.4	Statische Klassen	204
3.9.5	Statische Klasse oder Singleton-Pattern?	205
3.10	Namensräume (Namespaces)	207
3.10.1	Zugriff auf Namespaces	208
3.10.2	Die »using«-Direktive	210
3.10.3	Globaler Namespace	210
3.10.4	Vermeiden von Mehrdeutigkeiten	210
3.10.5	Namespaces festlegen	211
3.10.6	Der »::«-Operator	213
3.10.7	Unterstützung von Visual Studio bei den Namespaces	215
3.10.8	Die Direktive »using static«	216
3.11	Aktueller Stand der Klasse »Circle«	217
4	Vererbung, Polymorphie und Interfaces	221
4.1	Die Vererbung	221
4.1.1	Die Ableitung einer Klasse	222
4.1.2	Klassen, die nicht abgeleitet werden können	224
4.1.3	Konstruktoren in abgeleiteten Klassen	225
4.1.4	Der Zugriffsmodifizierer »protected«	226
4.1.5	Die Konstruktorverkettung in der Vererbung	227
4.2	Der Problemfall geerbter Methoden	230
4.2.1	Geerbte Methoden mit »new« verdecken	232
4.2.2	Abstrakte Methoden	234
4.2.3	Virtuelle Methoden	236
4.3	Typkonvertierung und Typuntersuchung von Objektvariablen	237
4.3.1	Die implizite Typkonvertierung von Objektreferenzen	237
4.3.2	Die explizite Typkonvertierung von Objektreferenzen	239
4.3.3	Typuntersuchung mit dem »is«-Operator	240
4.3.4	Typkonvertierung mit dem »as«-Operator	241
4.3.5	Pattern Matching (Musterabgleich) mit dem »is«-Operator	242

4.4	Polymorphie	243
4.4.1	Die »klassische« Methodenimplementierung	244
4.4.2	Abstrakte Methoden	245
4.4.3	Virtuelle Methoden	246
4.5	Weitere Gesichtspunkte der Vererbung	250
4.5.1	Versiegelte Methoden	250
4.5.2	Überladen einer Basisklassenmethode	251
4.5.3	Statische Member und Vererbung	252
4.5.4	Geerbte Methoden ausblenden?	252
4.6	Das Projekt »GeometricObjectsSolution« ergänzen	253
4.6.1	Die Klasse »GeometricObject«	254
4.7	Eingebettete Klassen	257
4.8	Interfaces (Schnittstellen)	258
4.8.1	Einführung in die Schnittstellen	258
4.8.2	Die Schnittstellendefinition	258
4.8.3	Die Schnittstellenimplementierung	259
4.8.4	Die Interpretation der Schnittstellen	265
4.8.5	Sortieren im Beispiel »GeometricObjectsSolution«	270
4.8.6	Weitere Anpassungen am Projekt »GeometricObjectsSolution«	272
4.9	Das Zerstören von Objekten – der Garbage Collector	273
4.9.1	Die Arbeitsweise des Garbage Collectors	273
4.9.2	Expliziter Aufruf des Garbage Collectors	275
4.9.3	Der Destruktor	276
4.9.4	Die »IDisposable«-Schnittstelle	277
4.9.5	Die »using«-Anweisung zum Zerstörung von Objekten	279
4.10	Die Ergänzungen in den Klassen »Circle« und »Rectangle«	280
5	Delegaten, Ereignisse und Lambda-Ausdrücke	283
5.1	Delegaten	283
5.1.1	Einführung in das Prinzip der Delegaten	283
5.1.2	Verwendung von Delegaten	288
5.1.3	Vereinfachter Delegatenaufruf	288
5.1.4	Multicast-Delegaten	288
5.1.5	Kovarianz und Kontravarianz mit Delegaten	290
5.2	Ereignisse eines Objekts	293
5.2.1	Ereignisse bereitstellen	294

5.2.2	Die Reaktion auf ein ausgelöstes Ereignis	297
5.2.3	Allgemeine Betrachtungen der Ereignishandler-Registrierung	299
5.2.4	Wenn der Ereignisempfänger ein Ereignis nicht behandelt	300
5.2.5	Ereignisse mit Übergabeparameter	302
5.2.6	Ereignisse in der Vererbung	306
5.2.7	Ein Blick hinter die Kulissen des Schlüsselworts »event«	307
5.2.8	Die Schnittstelle »INotifyPropertyChanged«	308
5.3	Lambda-Ausdrücke	310
5.3.1	Anonyme Methoden	310
5.3.2	Lambda-Ausdrücke	311
5.3.3	Expression-bodied Member	313
5.4	Änderungen im Projekt »GeometricObjectsSolution«	314
5.4.1	Überarbeitung des Events »InvalidMeasure«	314
5.4.2	Weitere Ereignisse im Projekt »GeometricObjects«	315

6 Strukturen und Enumerationen 319

6.1	Strukturen – eine Sonderform der Klassen	319
6.1.1	Die Definition einer Struktur	319
6.1.2	Initialisieren einer Strukturvariablen	320
6.1.3	Konstruktoren in Strukturen	321
6.1.4	Änderungen im Projekt »GeometricObjectsSolution«	322
6.2	Enumerationen (Aufzählungen)	325
6.2.1	Wertzuweisung an »enum«-Variablen	327
6.2.2	Alle Mitglieder einer Aufzählung durchlaufen	327
6.3	Boxing und Unboxing	328

7 Fehlerbehandlung und Debugging 331

7.1	Laufzeitfehler erkennen	332
7.1.1	Die »try ... catch«-Anweisung	334
7.1.2	Behandlung mehrerer Exceptions	336
7.1.3	Die Reihenfolge der »catch«-Zweige	339
7.1.4	Ausnahmen in einer Methodenaufkette	339
7.1.5	Ausnahmen werfen oder weiterleiten	340
7.1.6	Die »finally«-Anweisung	340
7.1.7	Ausnahmefilter	342

7.1.8	Die Klasse »Exception«	343
7.1.9	Benutzerdefinierte Ausnahmen	349
7.2	Debuggen mit Programmcode	355
7.2.1	Einführung	355
7.2.2	Die Klasse »Debug«	356
7.2.3	Die Klasse »Trace«	359
7.2.4	Bedingte Kompilierung	360
7.3	Fehlersuche mit Visual Studio	363
7.3.1	Debuggen im Haltemodus	363
7.3.2	Weitere Alternativen, Variableninhalte zu prüfen	368

8 Auflistungsklassen (Collections) 371

8.1	Collections im Namespace »System.Collections«	371
8.1.1	Die elementaren Schnittstellen der Auflistungsklassen	373
8.2	Die Klasse »ArrayList«	375
8.2.1	Einträge hinzufügen	375
8.2.2	Datenaustausch zwischen einem Array und einer »ArrayList«	378
8.2.3	Die Elemente einer »ArrayList« sortieren	379
8.2.4	Sortieren von Arrays mit »ArrayList.Adapter«	385
8.3	Die Klasse »Hashtable«	387
8.3.1	Methoden und Eigenschaften der Schnittstelle »IDictionary«	387
8.3.2	Beispielprogramm zur Klasse »Hashtable«	388
8.4	Die Klassen »Queue« und »Stack«	393
8.4.1	Die Klasse »Stack«	394
8.4.2	Die Klasse »Queue«	395
8.5	Eigene Auflistungen mit »yield« durchlaufen	396

9 Generics – generische Datentypen 401

9.1	Bereitstellen einer generischen Klasse	403
9.1.1	Mehrere generische Typparameter	405
9.1.2	Vorteile der Generics	405
9.2	Bedingungen (Constraints) festlegen	406
9.2.1	Constraints mit der »where«-Klausel formulieren	406
9.2.2	Typparameter auf Klassen oder Strukturen beschränken	407

9.2.3	Mehrere Constraints definieren	408
9.2.4	Der Konstruktor-Constraint »new()«	408
9.2.5	Das Schlüsselwort »default«	409
9.3	Generische Methoden	410
9.3.1	Methoden und Constraints	411
9.4	Generics und Vererbung	411
9.4.1	Virtuelle generische Methoden	412
9.5	Typkonvertierung von Generics	413
9.6	Generische Delegaten	414
9.6.1	Generische Delegaten und Constraints	415
9.7	»Nullable«-Typen	415
9.7.1	Konvertierungen mit »Nullable«-Typen mit dem »??«-Operator	416
9.8	Generische Collections	417
9.8.1	Die Interfaces der generischen Auflistungsklassen	417
9.8.2	Die generische Auflistungsklasse »List<T>«	418
9.8.3	Vergleiche mit Hilfe des Delegaten »Comparison<T>«	420
9.9	Kovarianz und Kontravarianz generischer Typen	421
9.9.1	Kovarianz mit Interfaces	422
9.9.2	Kontravarianz mit Interfaces	424
9.9.3	Zusammenfassung	425
9.9.4	Generische Delegaten mit varianten Typparametern	426
9.10	Ergänzungen im Beispielprojekt »GeometricObjectsSolution«	426
10	Weitere C#-Sprachfeatures	429
10.1	Implizit typisierte Variablen	429
10.2	Anonyme Typen	430
10.3	Erweiterungsmethoden	431
10.3.1	Die Prioritätsregeln	432
10.3.2	Generische Erweiterungsmethoden	434
10.3.3	Richtlinien für Erweiterungsmethoden	435
10.4	Spezielle Methoden	435
10.4.1	Partielle Methoden	436
10.4.2	Lokale Funktionen	439

10.5	Operatorüberladung	442
10.5.1	Einführung	442
10.5.2	Die Syntax der Operatorüberladung	442
10.5.3	Die Operatorüberladungen im Projekt »GeometricObjectsSolution«	444
10.5.4	Benutzerdefinierte Typkonvertierung	449
10.6	»Nullable«-Referenztypen	452
10.6.1	»Nullable«-Referenztypen im Beispiel »GeometricObjects«	455
10.7	Indexer	459
10.7.1	Überladen von Indexern	461
10.7.2	Parameterbehaftete Eigenschaften	463
10.8	Attribute	466
10.8.1	Das »Flags«-Attribut	468
10.8.2	Benutzerdefinierte Attribute	471
10.8.3	Attribute auswerten	475
10.8.4	Festlegen der Assembly-Eigenschaften in »AssemblyInfo.cs«	477
10.9	Der bedingte NULL-Operator	479
10.10	Der »nameof«-Operator	480
10.10.1	Einsatz in der Anwendung »GeometricObjects«	481
10.11	Dynamisches Binden	482
10.11.1	Eine kurze Analyse	483
10.11.2	Dynamische Objekte	484
10.12	Tupel	485
10.12.1	Benannte und unbenannte Tupel	487
10.12.2	Zuweisungsoperationen	488
10.12.3	Projektionsinitialisierer	488
10.12.4	Tupelvergleiche mit »==« und »!=«	489
10.12.5	Tupel als Methodenrückgabewerte	489
10.12.6	Dekonstruktion von Tupeln	490
10.12.7	Einzelne Tupelelemente ausschließen	491
10.13	Pattern Matching (Musterabgleich)	492
10.13.1	»switch« ohne »case«	494
10.14	Rückgabewerte mit »ref«	496
10.15	Unsicherer (unsafe) Programmcode – Zeigertechnik in C#	498
10.15.1	Einführung	498
10.15.2	Das Schlüsselwort »unsafe«	499
10.15.3	Die Deklaration von Zeigern	499
10.15.4	Die »fixed«-Anweisung	500

10.15.5	Zeigerarithmetik	502
10.15.6	Der Operator »->«	503
11	LINQ – Language Integrated Query	505
11.1	Einstieg in LINQ?	505
11.1.1	Verzögerte Ausführung	507
11.1.2	LINQ-Erweiterungsmethoden an einem Beispiel	507
11.2	LINQ to Objects	511
11.2.1	Musterdaten	511
11.2.2	Die allgemeine LINQ-Syntax	513
11.3	Die Abfrageoperatoren	515
11.3.1	Übersicht der Abfrageoperatoren	515
11.3.2	Die »from«-Klausel	515
11.3.3	Mit »where« filtern	517
11.3.4	Die Projektionsoperatoren	520
11.3.5	Die Sortieroperatoren	521
11.3.6	Gruppieren mit »GroupBy«	522
11.3.7	Verknüpfungen mit »Join«	524
11.3.8	Die Set-Operatoren-Familie	527
11.3.9	Die Familie der Aggregatoperatoren	529
11.3.10	Quantifizierungsoperatoren	532
11.3.11	Aufteilungsoperatoren	533
11.3.12	Die Elementoperatoren	535
11.3.13	Die Konvertierungsoperatoren	538
12	Arbeiten mit Dateien und Streams	539
12.1	Einführung	539
12.2	Namespaces der Ein- bzw. Ausgabe	540
12.2.1	Das Behandeln von Ausnahmen bei E/A-Operationen	541
12.3	Laufwerke, Verzeichnisse und Dateien	541
12.3.1	Die Klasse »File«	542
12.3.2	Die Klasse »FileInfo«	548
12.3.3	Die Klassen »Directory« und »DirectoryInfo«	550
12.3.4	Die Klasse »Path«	555
12.3.5	Die Klasse »DriveInfo«	556

12.4	Die »Stream«-Klassen	558
12.4.1	Die abstrakte Klasse »Stream«	559
12.4.2	Die von »Stream« abgeleiteten Klassen im Überblick	561
12.4.3	Die Klasse »FileStream«	562
12.5	Die Klassen »TextReader« und »TextWriter«	569
12.5.1	Die Klasse »StreamWriter«	570
12.5.2	Die Klasse »StreamReader«	573
12.6	Die Klassen »BinaryReader« und »BinaryWriter«	575
12.6.1	Komplexe binäre Dateien	578
13	Serialisierung	585
13.1	Serialisierungsverfahren	586
13.2	Binäre Serialisierung mit »BinaryFormatter«	587
13.2.1	Die Deserialisierung	589
13.2.2	Serialisierung mehrerer Objekte	591
13.3	Serialisierung mit »XmlSerializer«	593
13.3.1	XML-Serialisierung mit Attributen steuern	596
14	Multithreading	599
14.1	Einführung in das Multithreading	600
14.2	Threads – allgemein betrachtet	601
14.3	Mit der Klasse »Thread« arbeiten	603
14.3.1	Die Entwicklung einer einfachen Multithreading-Anwendung	603
14.3.2	Der Delegat »ParameterizedThreadStart«	605
14.3.3	Zugriff eines Threads auf sich selbst	606
14.3.4	Einen Thread für eine bestimmte Zeitdauer anhalten	606
14.3.5	Beenden eines Threads	607
14.3.6	Abhängige Threads – die Methode »Join«	609
14.3.7	Threadprioritäten festlegen	611
14.3.8	Vorder- und Hintergrundthreads	614
14.4	Der Threadpool	614
14.4.1	Ein einfaches Beispielprogramm	615

14.5 Synchronisation von Threads	616
14.5.1 Möglichkeiten der Synchronisation	618
14.5.2 Die Klasse »WaitHandle«	618
14.5.3 Sperren mit »Monitor«	622
14.5.4 Die Klasse »Mutex«	630
14.5.5 Die Klasse »Semaphore«	632
14.5.6 Das Attribut »MethodImpl«	637
14.5.7 Die Klasse »Interlocked«	637
14.5.8 Synchronisation von Threadpool-Threads	638
14.6 Grundlagen asynchroner Methodenaufrufe	639
14.6.1 Asynchroner Methodenaufruf	640
14.6.2 Asynchroner Aufruf mit Rückgabewerten	644
14.6.3 Eine Klasse mit asynchronen Methodenaufrufen	647
15 Die Task Parallel Library (TPL)	651
15.1 Die wichtigsten Klassen der TPL	652
15.2 Die Klasse »Task«	652
15.2.1 Die Konstruktoren eines Tasks	653
15.2.2 Das Erzeugen eines Tasks	655
15.2.3 Tasks mit Rückgabewert	656
15.2.4 Daten an einen Task übergeben	656
15.2.5 Auf das Beenden eines Tasks warten	658
15.2.6 Abbruch eines Tasks von außen	659
15.2.7 Bei Abbruch eine Callback-Methode aufrufen	661
15.2.8 Fehlerbehandlung	662
15.3 Die Klasse »Parallel«	665
15.3.1 Die Methode »Parallel.Invoke«	665
15.3.2 Schleifen mit »Parallel.For«	666
15.3.3 Den Grad der Parallelität beeinflussen	670
15.3.4 Auflistungen mit »Parallel.ForEach« durchlaufen	671
15.4 Asynchrone Programmierung mit »async« und »await«	671
15.4.1 Die Arbeitsweise von »async« und »await« verstehen	672
15.4.2 Asynchrone Operationen mit Rückgabewert	676

16 Grundlegende .NET-Klassen	679
16.1 Die Klasse »Object«	679
16.1.1 Referenzvergleiche mit »Equals« und »ReferenceEquals«	680
16.1.2 »ToString« und »GetType«	681
16.1.3 Die Methode »MemberwiseClone« und das Problem des Klonens	681
16.2 Die Klasse »String«	685
16.2.1 Das Erzeugen eines Strings	686
16.2.2 Die Eigenschaften von »String«	687
16.2.3 Die Methoden der Klasse »String«	687
16.2.4 Zusammenfassung der Klasse »String«	697
16.3 Die Klasse »StringBuilder«	699
16.3.1 Die Kapazität eines »StringBuilder«-Objekts	700
16.3.2 Die Konstruktoren der Klasse »StringBuilder«	701
16.3.3 Die Eigenschaften der Klasse »StringBuilder«	701
16.3.4 Die Methoden der Klasse »StringBuilder«	702
16.3.5 Allgemeine Anmerkungen	704
16.4 Der Typ »DateTime«	705
16.4.1 Die Zeitspanne »Tick«	705
16.4.2 Die Konstruktoren von »DateTime«	706
16.4.3 Die Eigenschaften von »DateTime«	707
16.4.4 Die Methoden der Klasse »DateTime«	708
16.5 Die Klasse »TimeSpan«	709
16.6 Ausgabeformatierung	712
16.6.1 Formatierung mit der Methode »String.Format«	712
16.6.2 Formatierung mit der Methode »ToString«	716
16.6.3 Benutzerdefinierte Formatierung	717
17 Projektmanagement und Visual Studio 2019	721
17.1 Der Projekttyp »Klassenbibliothek«	721
17.1.1 Mehrere Projekte in einer Projektmappe verwalten	723
17.1.2 Die Zugriffsmodifizierer »public« und »internal«	724
17.1.3 Friend Assemblies	724
17.1.4 Einbinden einer Klassenbibliothek	725
17.2 Assemblies	726
17.2.1 Ein Überblick über das Konzept der Assemblies	726

17.2.2	Allgemeine Beschreibung privater und globaler Assemblies	728
17.2.3	Die Struktur einer Assembly	728
17.2.4	Globale Assemblies	733
17.2.5	Die Installation einer Assembly im GAC mit dem Tool »gacutil.exe«	737
17.3	Konfigurationsdateien	738
17.3.1	Die verschiedenen Konfigurationsdateien	739
17.3.2	Die Struktur einer Anwendungskonfigurationsdatei	741
17.3.3	Spezifische Einträge in der Anwendungskonfigurationsdatei	744
17.3.4	Einträge der Anwendungskonfigurationsdatei auswerten	745
17.3.5	Editierbare, anwendungsbezogene Einträge mit <appSettings>	750
17.4	Versionierung einer Assembly	752
17.4.1	Die Herausgeberrichtliniendatei	754
17.5	XML-Dokumentation	755
17.5.1	Das Prinzip der XML-Dokumentation	755
17.5.2	Die XML-Kommentar-Tags	757
17.5.3	Generieren der XML-Dokumentationsdatei	758
17.6	Der Klassendesigner (Class Designer)	759
17.6.1	Ein typisches Klassendiagramm	760
17.6.2	Hinzufügen von Klassendiagrammen	762
17.6.3	Die Toolbox des Klassendesigners	763
17.6.4	Das Fenster »Klassendetails«	763
17.6.5	Klassendiagramme als Bilder exportieren	765
17.7	Refactoring	766
17.7.1	Methode extrahieren	766
17.7.2	Bezeichner umbenennen	768
17.8	Code-Snippets (Codeausschnitte)	769
17.8.1	Codeausschnitte einfügen	769
17.8.2	Die Anatomie eines Codeausschnitts	770
18	Die Zukunft: .NET Core und .NET Standard	773
18.1	Allgemeines	774
18.2	Die drei Säulen von .NET	775
18.2.1	.NET Framework	775
18.2.2	.NET Core	775
18.2.3	Xamarin	777

18.3	.NET Standard	777
18.3.1	Beispiel mit .NET Standard	780
18.4	Portieren von .NET Framework nach .NET Standard	783
18.5	C#-Sprachergänzungen für .NET Core	784
18.5.1	Die Struktur »System.Range«	785
18.5.2	Die Struktur »System.Index«	787
19	Einführung in das Entity Framework	789
19.1	Das Entity Framework im Überblick	789
19.1.1	Die Organisation der Daten im Entity Framework	791
19.2	Erstellen eines Entity Data Models (EDM)	792
19.2.1	Kurzbeschreibung des EDM	796
19.2.2	Allgemeine Eigenschaften einer Entität	796
19.2.3	Eigenschaftstypen eines Entitätsobjekts	798
19.2.4	Assoziationen im Entity Data Model	801
19.2.5	Der Kontext der Entitäten	802
19.2.6	Der Aufbau des Entity Data Models	803
19.2.7	ConnectionStrings	807
19.2.8	Strategien, mit Entity Framework zu arbeiten	808
19.3	Die automatisch erzeugten Klassen im EDM	809
19.3.1	Die Klasse »DbContext«	810
19.3.2	Die Entitätsklassen	811
20	Database First mit dem EDM-Designer	815
20.1	Einfache Datenabfragen mit LINQ-to-Entities	815
20.1.1	Allgemeine Abfragen	816
20.1.2	Der Unterschied der beiden Schnittstellen »IEnumerable« und »IQueryable«	817
20.1.3	Sofortiges Ausführen einer Abfrage	818
20.1.4	Eine Entität mit der Methode »Find« suchen	820
20.1.5	Lokale Daten mit »Load« laden	821
20.1.6	Abfragen des lokalen Datencaches mit »Local«	822
20.1.7	»Local« und die »ObservableCollection<TEntity>«	824
20.2	In Beziehung stehende Daten laden	825
20.2.1	Das Lazy Loading	825

20.2.2	Das Eager Loading	828
20.2.3	Das explizite Laden mit »Load«	830
20.2.4	Mit »Query« vor dem Laden filtern	832
20.3	Ändern von Entitäten	834
20.3.1	Entitäten ändern	834
20.3.2	Hinzufügen einer neuen Entität	835
20.3.3	Löschen einer Entität	838
20.3.4	SQL mit Entity Framework	840
20.4	Das Verfolgen der Änderungen	841
20.4.1	Das »DbEntityEntry«-Objekt	841
20.4.2	Die Klasse »DbPropertyValues«	844
20.4.3	Die Klasse »DbChangeTracker«	846
20.4.4	Entitätszustände	847
20.4.5	Die automatische Änderungsnachverfolgung mit »DetectChanges«	848
20.4.6	Änderungsnachverfolgung mit dynamischen Proxies	851
20.5	Die Change Tracker API	854
20.5.1	Die »Current-«, »Original-« und »DatabaseValues« ermitteln	855
20.5.2	Mit individuellen Eigenschaften arbeiten	859
20.5.3	Ändern mit der »Reference«-Navigationseigenschaft	861
20.5.4	Die Methode »Entries« der Klasse »ChangeTracker«	863
20.5.5	Eine Entität aus der Datenbank aktualisieren	864
20.6	Parallelitätskonflikte behandeln	865
20.6.1	Allgemeine Betrachtungen	866
20.6.2	Das Standardverhalten bei konkurrierenden Zugriffen	868
20.6.3	Das Aktualisierungsverhalten mit »Fixed« beeinflussen	869
20.6.4	Aktualisierung mit einer Timestamp-Spalte	870
20.6.5	Auf die Ausnahme »DbUpdateConcurrencyException« reagieren	871
20.7	Asynchrone Abfrage- und Speicheroperationen	879
20.8	Transaktionen	881
20.9	Kontextlose Entitäten ändern	883
20.9.1	Self-Tracking-Entities (STE)	884
20.10	Validieren mit dem Entity Framework	887
20.10.1	Mit Data Annotations Eigenschaften validieren	890
20.10.2	Validieren einer kompletten Entität	898
20.10.3	Validieren mehrerer Entitäten im Kontext	901
20.10.4	Das Überschreiben der Methode »DbContext.ValidateEntity«	903

21	Entity Framework – Code First	907
21.1	Erste Schritte	907
21.2	Entity Framework 6 Power Tools	909
21.3	Das erste Code-First-Modell	909
21.3.1	Die Datenbank erzeugen	912
21.3.2	Erzeugen der Datenbank in einer beliebigen Datenbankinstanz	914
21.3.3	Das Initialisierungsverhalten der Datenbank steuern	915
21.4	Einführung in die Konfiguration von Code First	917
21.4.1	Konfigurieren mit Data Annotations	918
21.4.2	Konfigurieren mit der Fluent API	919
21.4.3	Separate Konfigurationsklassen für die Fluent API	921
21.5	Konventionen und Konfiguration im Detail	922
21.5.1	Primärschlüssel	923
21.5.2	Elementare Data Annotations	927
21.5.3	Entitäten auf das optimistische Sperren vorbereiten	929
21.5.4	Entitäten ohne Timestamp-Spalten	932
21.6	Komplexe Typen	933
21.6.1	Komplexe Typen mit Data Annotations	936
21.7	Konventionen und Konfiguration von Beziehungen	937
21.7.1	Konfiguration mit Data Annotations	939
21.7.2	Konfiguration einer 1:n-Beziehung mit der Fluent API	940
21.7.3	Entitäten und Fremdschlüssel	943
21.7.4	Die Annotation »ForeignKey«	944
21.7.5	Kaskadierende Löscherweitergabe	946
21.7.6	Das Attribut »InverseProperty«	948
21.7.7	1:1-Beziehungen mit Code First	951
21.7.8	n:n-Beziehungen mit Code First	955
22	Einführung in die WPF und XAML	957
22.1	Die Merkmale einer WPF-Anwendung	958
22.1.1	Anwendungstypen	959
22.1.2	Eine WPF-Anwendung und ihre Dateien	960
22.1.3	Ein erstes WPF-Beispiel	963
22.1.4	Wichtige WPF-Features	966
22.1.5	Der logische und der visuelle Elementbaum	968

22.2 XAML (Extended Application Markup Language)	970
22.2.1 Die Struktur einer XAML-Datei	970
22.2.2 Eigenschaften eines XAML-Elements in Attribut-Schreibweise festlegen	973
22.2.3 Eigenschaften im Eigenschaftsfenster festlegen	973
22.2.4 Die Eigenschaft-Element-Syntax	974
22.2.5 Inhaltseigenschaften	975
22.2.6 Typkonvertierung	978
22.2.7 Markup-Erweiterungen (Markup Extensions)	979
22.2.8 XML-Namespaces	982
22.2.9 XAML-Spracherweiterungen	985
22.2.10 Die Direktive »#region« nutzen	987

23 Die WPF-Layoutcontainer

23.1 Allgemeiner Überblick	989
23.2 Gemeinsame Eigenschaften der Layoutcontainer	990
23.2.1 Das »Canvas«	991
23.2.2 Das »StackPanel«	992
23.2.3 Das »WrapPanel«	995
23.2.4 Das »DockPanel«	996
23.2.5 Das »Grid«-Steuerelement	998
23.2.6 Das »UniformGrid«	1004
23.3 Verschachteln von Layoutcontainern	1005

24 Fenster in der WPF

24.1 Hosts der WPF	1009
24.2 Fenster vom Typ »Window«	1010
24.2.1 Mehrere Fenster in einer Anwendung	1012
24.3 Fenster vom Typ »NavigationWindow«	1014
24.3.1 Das »Page«-Element	1016
24.3.2 Navigation zwischen den Seiten	1018
24.3.3 Der Verlauf der Navigation – das Journal	1021
24.3.4 Datenübergabe zwischen den Seiten mit einem Konstruktor	1023
24.3.5 Datenübergabe mit der Methode »Navigate«	1024
24.3.6 Navigation im Internet	1026

24.3.7 Navigieren mit dem Ereignis »RequestNavigate« des »HyperLink«-Elements	1026
24.4 Hosts vom Typ »Frame«	1027
24.5 Nachrichtenfenster mit »MessageBox«	1029
24.5.1 Die Methode »MessageBox.Show«	1029
24.6 Standarddialoge in der WPF	1032
24.6.1 Der Dialog »OpenFileDialog«	1032
24.6.2 Der Dialog »SaveFileDialog«	1035

25 WPF-Steuerelemente

25.1 Die Hierarchie der WPF-Komponenten	1037
25.2 Allgemeine Eigenschaften der WPF-Steuerelemente	1039
25.2.1 Den Außenrand mit der Eigenschaft »Margin« festlegen	1039
25.2.2 Den Innenrand mit der Eigenschaft »Padding« festlegen	1040
25.2.3 Die Eigenschaft »Content«	1040
25.2.4 Die Größe einer Komponente	1042
25.2.5 Die Ausrichtung einer Komponente	1043
25.2.6 Die Sichtbarkeit eines Steuerelements	1043
25.2.7 Die farbliche Darstellung	1044
25.2.8 Die Schriften	1044
25.3 Die Gruppe der Schaltflächen	1045
25.3.1 Die Basisklasse »ButtonBase«	1045
25.3.2 Das Steuerelement »Button«	1046
25.3.3 Das Steuerelement »ToggleButton«	1047
25.3.4 Das Steuerelement »RepeatButton«	1047
25.3.5 Das Steuerelement »CheckBox«	1049
25.3.6 Das Steuerelement »RadioButton«	1049
25.4 Einfache Eingabesteuerelemente	1050
25.4.1 Das Steuerelement »Label«	1050
25.4.2 Das Steuerelement »TextBox«	1051
25.4.3 Das Steuerelement »PasswordBox«	1055
25.4.4 Das Steuerelement »TextBlock«	1055
25.5 WPF-Listenelemente	1058
25.5.1 Das Steuerelement »ListBox«	1058
25.5.2 Die »ComboBox«	1061
25.5.3 Das Steuerelement »ListView«	1061

25.5.4	Das Steuerelement »TreeView«	1065
25.5.5	Das Steuerelement »TabControl«	1071
25.5.6	Die Menüleiste	1072
25.5.7	Das Kontextmenü	1075
25.5.8	Symbolleisten	1076
25.5.9	Die Statusleiste	1079
25.6	Weitere Steuerelemente	1080
25.6.1	Das Steuerelement »Tooltip«	1080
25.6.2	Die »ProgressBar«	1082
25.6.3	Das Steuerelement »Slider«	1082
25.6.4	Das »GroupBox«-Steuerelement	1082
25.6.5	Das Steuerelement »ScrollViewer«	1083
25.6.6	Das Steuerelement »Expander«	1085
25.6.7	Das Steuerelement »Border«	1086
25.6.8	Die »Image«-Komponente	1087
25.6.9	»Calendar« und »DatePicker« zur Datumsangabe	1089
25.7	Das »Ribbon«-Steuerelement	1091
25.7.1	Voraussetzungen für den Zugriff auf das »Ribbon«-Control	1091
25.7.2	Ein kurzer Überblick	1092
25.7.3	Der XAML-Code	1092

26 Dependency Properties 1099

26.1	Die Charakteristik von Abhängigkeitseigenschaften	1099
26.2	Den Wert einer Abhängigkeitseigenschaft bilden	1100
26.3	Definition einer Dependency Property	1101
26.3.1	Registrieren einer Abhängigkeitseigenschaft	1102
26.3.2	Der Eigenschaftswrapper	1103
26.3.3	Die Eigenschaftsmetadaten	1105
26.3.4	Zurückstellen auf den Standardwert	1108
26.3.5	Vererbung von Abhängigkeitseigenschaften	1109
26.4	Validieren einer Abhängigkeitseigenschaft	1109
26.4.1	Validieren mit »ValidateValueCallback«	1110
26.4.2	Validieren mit »CoerceValueCallback«	1110
26.5	Angehängte Eigenschaften (Attached Properties)	1111
26.5.1	Angehängte Eigenschaften zur Laufzeit ändern	1113

27 Ereignisse in der WPF 1115

27.1	Ereignishandler bereitstellen	1115
27.2	Routing-Strategien	1116
27.2.1	Routed Events und der Elementbaum	1118
27.2.2	Beispielanwendung	1119
27.2.3	Sonderfall: Ereignisse mit der linken Maustaste	1120
27.3	Der Ereignishandler	1122
27.3.1	Die Klasse »RoutedEventArgs«	1122
27.3.2	Die Quelle des Routing-Prozesses	1123
27.3.3	Die Eigenschaft »Handled«	1124
27.3.4	Registrieren und Deregistrieren eines Ereignishandlers mit Code	1125
27.4	Benutzerdefinierte Routed Events	1125
27.4.1	Ereignisauslösung	1127
27.4.2	Das gebubbelte Ereignis im Elementbaum verwenden	1128
27.5	Mausereignisse in der WPF	1129
27.5.1	Ziehen der Maus	1129
27.5.2	Auswerten der Mausclicks	1130
27.5.3	Capturing	1131

28 Ressourcen, Styles, Trigger und Templates 1135

28.1	Binäre Ressourcen	1135
28.1.1	Zugriff auf binäre Ressourcen	1136
28.1.2	Zugriff auf binäre Ressourcen mit C#	1138
28.2	Logische Ressourcen	1138
28.2.1	Die Suche nach einer Ressource	1139
28.2.2	Definieren einer Ressource	1140
28.2.3	Zugriff auf eine logische Ressource mit C#-Code	1141
28.2.4	»StaticResource« vs. »DynamicResource«	1142
28.2.5	Anbinden einer dynamischen Ressource mit C#-Code	1144
28.2.6	WPF-Elemente als Ressourcen	1145
28.2.7	Anwendungsübergreifende Ressourcen	1146
28.2.8	Abrufen von Systemressourcen	1149
28.3	Styles	1150
28.3.1	Untypisierte Styles (explizite Styles)	1151
28.3.2	Typisierte Styles (implizite Styles)	1153

28.3.3	Erweitern eines Styles mit »BasedOn«	1155
28.3.4	Ereignisse mit »EventSetter« zentral abonnieren	1157
28.4	Trigger	1158
28.4.1	Einfache Trigger (Eigenschaftstrigger)	1160
28.4.2	Mehrere Bedingungen mit »MultiTrigger«	1162
28.4.3	»DataTrigger«	1163
28.4.4	»MultiDataTrigger«	1165
28.4.5	»EventTrigger«	1167
28.5	Templates	1169
28.5.1	Allgemeines zu »ControlTemplate«-Elementen	1170
28.5.2	Definition innerhalb eines Styles	1175
29	WPF-Datenbindung	1177
29.1	Die Klasse »Binding«	1180
29.1.1	Binden an eine Datenquelle	1181
29.1.2	Mit »Path« an eine Eigenschaft der Datenquelle binden	1184
29.1.3	Die Bindungsrichtung mit »Mode« festlegen	1186
29.1.4	Das Aktualisieren mit »UpdateSourceTrigger« steuern	1188
29.1.5	Die Ereignisse »SourceUpdated« und »TargetUpdated«	1190
29.1.6	Beenden einer Bindung	1192
29.2	Konverter mit »IValueConverter« und »IMultiValueConverter«	1192
29.2.1	Einfaches Konverter-Beispiel	1193
29.2.2	Ein weiteres Konverter-Beispiel	1196
29.2.3	Konverter und »MultiBinding«	1198
29.3	Validieren von Bindungen	1200
29.3.1	Die Validierung mit »ExceptionValidationRule«	1200
29.3.2	Eine benutzerdefinierte »ValidationRule«	1203
29.3.3	Validieren mit der Schnittstelle »IDataErrorInfo«	1206
29.3.4	Fehlerhinweise individuell gestalten	1207
29.3.5	Ereignisauslösung bei einem Validierungsfehler	1209
29.3.6	Mehrere Controls mit »BindingGroup« gleichzeitig validieren	1209
29.4	Datenbindung mit »ObjectDataProvider«	1213
29.4.1	Aufruf eines parametrisierten Konstruktors	1214
29.4.2	Objektmethode aufrufen	1215
29.4.3	Bindung an eine statische Methode	1215
29.5	Aktualisieren von Datenklassen	1216
29.5.1	Ein Objekt mit XAML-Code erzeugen und binden	1217

29.5.2	Ein Objekt mit C#-Code erzeugen und binden	1218
29.5.3	Aktualisieren benutzerdefinierter Objekte	1219
29.6	Datenbindung von Listen-Steuerelementen	1223
29.6.1	Das Layout eines »ItemsControl«-Steuerelements anpassen	1224
29.6.2	Binden an ein »ListBox«-Element	1225
29.6.3	Die Klasse »ObservableCollection<T>«	1228
29.6.4	Die Darstellung eines »ListBoxItem«-Elements anpassen	1230
29.6.5	»DataTemplate« mit »DataTrigger«	1234
29.7	Datenbindung und das Entity Framework	1237
29.8	Das Steuerelement »DataGrid«	1238
29.8.1	Elementare Eigenschaften des »DataGrid«-Objekts	1238
29.8.2	Beispielprogramm mit dem »DataGrid«	1240
29.8.3	Details einer Zeile anzeigen	1246
29.8.4	DataGrid und 1:n-Beziehungen	1247
29.9	Das »TreeView«-Control	1250
29.9.1	Der Programmcode	1251
29.9.2	Das Window	1255
29.10	Navigieren, Filtern, Sortieren und Gruppieren	1259
29.10.1	Navigieren in einer Datenmenge	1260
29.10.2	Sortieren von Datenmengen	1263
29.10.3	Filtern von Daten	1266
29.10.4	Gruppieren von Daten	1268
29.11	Dynamische Zuweisung von Styles und »DataTemplate«-Objekten	1272
29.11.1	Styles dynamisch ändern	1272
29.11.2	»DataTemplate« per C#-Code auswählen	1275
30	WPF – weitergehende Techniken	1279
30.1	WPF und Multithreading	1279
30.1.1	Nachrichtenschleife und »Dispatcher«-Klasse	1280
30.1.2	Zugriff auf UI-Komponenten aus einem Worker-Thread	1283
30.1.3	»BeginInvoke« und »Invoke«	1284
30.1.4	Die »DispatcherObject«-Klasse	1285
30.2	Globalisierung und Lokalisierung	1286
30.2.1	Globalisierung	1286
30.2.2	Lokalisierung	1287

30.3 Benutzerdefinierte Controls	1296
30.3.1 Erstellen eines benutzerdefinierten Steuerelements	1296
30.3.2 Der XAML-Code des »UserControl«-Elements	1298
30.3.3 Die Programmlogik des Steuerelements	1299
30.3.4 Das Steuerelement »ColorMixer« testen	1304
31 WPF-Commands	1307
31.1 Allgemeine Beschreibung	1308
31.2 Ein erstes Programmbeispiel	1308
31.2.1 Integrierte Befehle in den UI-Komponenten	1310
31.3 Die Befehlsquelle	1311
31.3.1 Das Befehlsziel mit »CommandTarget« angeben	1312
31.3.2 Einen Fokusbereich mit der Klasse »FocusManager« definieren	1313
31.3.3 Zusätzliche Daten mit »CommandParameter« bereitstellen	1314
31.4 WPF-Commands	1315
31.4.1 Die Arbeitsweise eines Befehls	1316
31.4.2 Die Klasse »CommandManager«	1317
31.5 »RoutedCommand«-Objekte und »CommandBindings«	1320
31.5.1 Vordefinierte WPF-Commands	1321
31.5.2 Die Klasse »RoutedCommand«	1322
31.5.3 Befehlsbindungen mit »CommandBinding« einrichten	1323
31.5.4 Befehlsbindung mit Programmcode	1325
31.5.5 Befehle mit Maus oder Tastatur aufrufen	1325
31.5.6 Benutzerdefinierter »RoutedCommand«	1327
32 Das MVVM-Pattern	1331
32.1 Die Theorie hinter dem Model-View-ViewModel-Pattern	1331
32.2 Allgemeine Beschreibung des Beispielprogramms	1332
32.3 Der Ausgangspunkt im Beispiel »MVVM_Origin«	1334
32.4 Das Bereitstellen des Modells	1334
32.5 Bereitstellen des ViewModels	1336
32.5.1 Abrufen und Bereitstellen der Daten	1337

32.5.2 Die Anbindung der Daten an die »ListView« der View	1339
32.5.3 Die Anbindung der Textboxen	1340
32.6 WPF-Commands und Eigenschaften im ViewModel	1341
32.6.1 Die Umsetzung von Commands im Model-View-ViewModel	1341
32.6.2 Die allgemeine Beschreibung eines Commands mit »RelayCommand«	1342
32.6.3 Ergänzen der Klasse »MainViewModel«	1344
32.6.4 Die aktuelle Position des Datensatzzeigers	1346
32.7 »RoutedCommand«-Objekte im MVVM	1347
32.7.1 Änderungen im »MainWindow«	1348
32.7.2 Ergänzungen im ViewModel	1349
32.7.3 Die Ereignishandler der »CommandBinding«-Objekte	1351
32.8 Beliebige Ereignisse mit »EventTrigger«-Objekten behandeln	1352
32.8.1 Mausereignisse triggern	1352
32.8.2 Ergänzung des ViewModels	1355
32.9 Die Klasse »Person« durch ein ViewModel kapseln	1356
32.9.1 Die Model-spezifische Klasse »PString«	1357
32.9.2 Die Model-spezifische Klasse »PDateTime«	1358
32.9.3 Die Klasse »PersonViewModel«	1360
32.9.4 Notwendige Anpassungen in »MainViewModel«	1363
32.9.5 Anpassungen im XAML-Code	1365
32.10 Die Schaltflächen »Rückgängig« und »Speichern«	1368
32.10.1 Eine Änderung zurücknehmen	1368
32.10.2 Die Änderungen der Listenobjekte speichern	1370
32.11 Ein Control in der View fokussieren	1374
32.11.1 Erste Überlegungen	1374
32.11.2 Definition der angehängten Eigenschaft	1375
32.11.3 Die angehängte Eigenschaft im XAML-Code	1376
32.11.4 Das ViewModel ergänzen	1377
32.12 Die Listenelemente sortieren	1378
32.12.1 Ergänzungen im XAML-Code	1378
32.12.2 Ergänzungen im ViewModel	1378
32.12.3 Die Klassen »PString« und »PDateTime« anpassen	1379
32.13 Ereignisse im ViewModel auslösen	1380
32.13.1 Die Löschbestätigung	1381
32.13.2 Das Schließen des Fensters	1382

33 2D-Grafik	1385
33.1 Shapes	1385
33.1.1 Allgemeine Beschreibung	1385
33.1.2 »Line«-Elemente	1386
33.1.3 »Ellipse«- und »Rectangle«-Elemente	1387
33.1.4 »Polygon« - und »Polyline« -Elemente	1387
33.1.5 Darstellung der Linien	1387
33.2 Path-Elemente	1389
33.2.1 Das Element »GeometryGroup«	1390
33.2.2 Das Element »CombinedGeometry«	1391
33.2.3 Geometrische Figuren mit »PathGeometry«	1392
33.3 »Brush«-Objekte	1393
33.3.1 »SolidColorBrush«	1394
33.3.2 »LinearGradientBrush«	1395
33.3.3 »RadialGradientBrush«	1397
33.3.4 Muster mit »TileBrush«	1399
33.3.5 Bilder mit »ImageBrush«	1401
33.3.6 Effekte mit »VisualBrush«	1402
33.3.7 Das Element »DrawingBrush«	1404

34 Komponententests (Unit-Tests) 1405

34.1 Was ist ein Unit-Test?	1405
34.2 Ein erster Komponententest	1408
34.2.1 Das »AAA«-Prinzip	1410
34.2.2 Automatisches Generieren eines Testprojekts	1412
34.2.3 Ausführen eines Unit-Tests	1413
34.3 Komponententest schreiben und ausführen	1415
34.3.1 Unit-Tests gruppieren	1415
34.3.2 Merkmale festlegen	1417
34.3.3 Wiedergabelisten	1423
34.3.4 Exceptions testen	1424
34.3.5 Codeabdeckung (Code Coverage)	1426
34.3.6 Testen von privaten Methoden	1429
34.3.7 Die Attribute »TimeOut« und »Ignore«	1431
34.3.8 Komponententests debuggen	1431

34.4 Die Klasse »TestContext«	1432
34.5 Data-Driven Unit Tests (datengetriebene Tests)	1436
34.5.1 Eine Datenbank als Datenquelle	1437
34.5.2 Data-Driven Test mit XML	1439
34.5.3 Data-Driven Test mit einer CSV-Datei	1441
34.5.4 Die Datenquelle in der »App.config«-Datei konfigurieren	1443
34.5.5 Testdaten mit dem Attribut »DataRow«	1446
34.5.6 Testdaten mit dem Attribut »DynamicData«	1447
34.6 Lebenszyklus- Attribute	1449
34.6.1 Die Attribute »TestInitialize« und »TestCleanup«	1450
34.6.2 Die Attribute »ClassInitialize« und »ClassCleanup«	1452
34.6.3 Die Attribute »AssemblyInitialize« und »AssemblyCleanup«	1453
34.7 Testen mit »Assert«	1453
34.7.1 Die Klasse »Assert«	1454
34.7.2 Die Klasse »StringAssert«	1458
34.7.3 Die Klasse »CollectionAssert«	1459
34.8 Test-Driven Development – TDD	1462
Index	1465