

Inhalt

Vorwort zur 7. Auflage	32
1 Allgemeine Einführung in .NET	35
1.1 Warum .NET?	35
1.1.1 Ein paar Worte zu diesem Buch	37
1.1.2 Die Beispielprogramme	39
1.2 .NET unter die Lupe genommen	40
1.2.1 Das Entwicklerdilemma	40
1.2.2 .NET – ein paar allgemeine Eigenschaften	41
1.2.3 Das Sprachenkonzept	42
1.2.4 Die Common Language Specification (CLS)	44
1.2.5 Das Common Type System (CTS)	45
1.2.6 Das .NET Framework	46
1.2.7 Die Common Language Runtime (CLR)	47
1.2.8 Die .NET-Klassenbibliothek	47
1.2.9 Das Konzept der Namespaces	48
1.3 Assemblies	50
1.3.1 Die Metadaten	51
1.3.2 Das Manifest	52
1.4 Die Entwicklungsumgebung	52
1.4.1 Editionen von Visual Studio 2015	52
1.4.2 Hard- und Softwareanforderungen	53
1.4.3 Die Installation	53
1.4.4 Die Entwicklungsumgebung von Visual Studio 2015	54
2 Grundlagen der Sprache C#	59
2.1 Konsolenanwendungen	59
2.1.1 Allgemeine Anmerkungen	59
2.1.2 Ein erstes Konsolenprogramm	59
2.2 Grundlagen der C#-Syntax	62
2.2.1 Kennzeichnen, dass eine Anweisung abgeschlossen ist	62

2.2.2	Anweisungs- und Gliederungsblöcke	63
2.2.3	Kommentare	64
2.2.4	Die Groß- und Kleinschreibung	65
2.2.5	Die Struktur einer Konsolenanwendung	65
2.3	Variablen und Datentypen	67
2.3.1	Variablendeklaration	67
2.3.2	Der Variablenbezeichner	69
2.3.3	Der Zugriff auf eine Variable	69
2.3.4	Ein- und Ausgabemethoden der Klasse »Console«	70
2.3.5	Die elementaren Datentypen von .NET	76
2.3.6	Typkonvertierung	82
2.4	Operatoren	89
2.4.1	Arithmetische Operatoren	90
2.4.2	Vergleichsoperatoren	93
2.4.3	Logische Operatoren	94
2.4.4	Bitweise Operatoren	97
2.4.5	Zuweisungsoperatoren	100
2.4.6	Stringverkettung	101
2.4.7	Sonstige Operatoren	101
2.4.8	Operator-Vorrangregeln	102
2.5	Datenfelder (Arrays)	103
2.5.1	Die Deklaration und Initialisierung eines Arrays	103
2.5.2	Der Zugriff auf die Array-Elemente	104
2.5.3	Mehrdimensionale Arrays	106
2.5.4	Festlegen der Array-Größe zur Laufzeit	107
2.5.5	Bestimmung der Array-Obergrenze	108
2.5.6	Die Gesamtanzahl der Array-Elemente	109
2.5.7	Verzweigte Arrays	109
2.6	Kontrollstrukturen	111
2.6.1	Die »if«-Anweisung	111
2.6.2	Das »switch«-Statement	117
2.7	Programmschleifen	121
2.7.1	Die »for«-Schleife	122
2.7.2	Die »foreach«-Schleife	132
2.7.3	Die »do«- und die »while«-Schleife	133

3	Das Klassendesign	137
3.1	Einführung in die Objektorientierung	137
3.2	Die Klassendefinition	140
3.2.1	Klassen in Visual Studio anlegen	140
3.2.2	Das Projekt »GeometricObjectsSolution«	141
3.2.3	Die Deklaration von Objektvariablen	143
3.2.4	Zugriffsmodifizierer einer Klasse	144
3.2.5	Splitten einer Klassendefinition mit »partial«	145
3.2.6	Arbeiten mit Objektreferenzen	146
3.3	Referenz- und Wertetypen	148
3.3.1	Werte- und Referenztypen nutzen	149
3.4	Die Eigenschaften eines Objekts	150
3.4.1	Öffentliche Felder	150
3.4.2	Datenkapselung	151
3.4.3	Die Ergänzung der Klasse »Circle«	153
3.4.4	Lese- und schreibgeschützte Eigenschaften	154
3.4.5	Sichtbarkeit der Accessoren »get« und »set«	155
3.4.6	Unterstützung von Visual Studio	156
3.4.7	Auto-Property (automatisch implementierte Property)	157
3.5	Methoden eines Objekts	158
3.5.1	Methoden mit Rückgabewert	159
3.5.2	Methoden ohne Rückgabewert	162
3.5.3	Methoden mit Parameterliste	163
3.5.4	Methodenüberladung	165
3.5.5	Variablen innerhalb einer Methode (lokale Variablen)	168
3.5.6	Referenz- und Wertparameter	169
3.5.7	Besondere Aspekte einer Parameterliste	175
3.5.8	Zugriff auf private Daten	180
3.5.9	Die Trennung von Daten und Code	181
3.5.10	Namenskonflikte mit »this« lösen	182
3.5.11	Methode oder Eigenschaft?	183
3.5.12	Umbenennen von Methoden und Eigenschaften	184
3.6	Konstruktoren	185
3.6.1	Konstruktoren bereitstellen	186
3.6.2	Die Konstruktoraufufe	187
3.6.3	Definition von Konstruktoren	187
3.6.4	»public«- und »internal«-Konstruktoren	188

3.6.5	»private«-Konstruktoren	188
3.6.6	Konstruktoraufrufe umleiten	189
3.6.7	Vereinfachte Objektinitialisierung	190
3.7	Der Destruktor	191
3.8	Konstanten in einer Klasse	192
3.8.1	Konstanten mit dem Schlüsselwort »const«	192
3.8.2	Schreibgeschützte Felder mit »readonly«	193
3.8.3	Eine Auto-Property als Konstante	193
3.9	Statische Klassenkomponenten	194
3.9.1	Statische Eigenschaften	194
3.9.2	Statische Methoden	196
3.9.3	Statische Klasseninitialisierer	198
3.9.4	Statische Klassen	198
3.9.5	Statische Klasse oder Singleton-Pattern?	199
3.10	Namensräume (Namespaces)	201
3.10.1	Zugriff auf Namespaces	202
3.10.2	Die »using«-Direktive	204
3.10.3	Globaler Namespace	205
3.10.4	Vermeiden von Mehrdeutigkeiten	205
3.10.5	Namespaces festlegen	206
3.10.6	Der »::«-Operator	207
3.10.7	Unterstützung von Visual Studio bei den Namespaces	209
3.10.8	Die Direktive »using static«	211
3.11	Stand der Klasse »Circle«	211
4	Vererbung, Polymorphie und Interfaces	214
<hr/>		
4.1	Die Vererbung	214
4.1.1	Die Ableitung einer Klasse	215
4.1.2	Klassen, die nicht abgeleitet werden können	217
4.1.3	Konstruktoren in abgeleiteten Klassen	217
4.1.4	Der Zugriffsmodifizierer »protected«	219
4.1.5	Die Konstruktorverkettung in der Vererbung	219
4.2	Der Problemfall geerbter Methoden	223
4.2.1	Geerbte Methoden mit »new« verdecken	225
4.2.2	Abstrakte Methoden	227
4.2.3	Virtuelle Methoden	229

4.3	Typumwandlung und Typuntersuchung von Objektvariablen	230
4.3.1	Die implizite Typumwandlung von Objektreferenzen	230
4.3.2	Die explizite Typumwandlung von Objektreferenzen	232
4.3.3	Typuntersuchung mit dem »is«-Operator	233
4.3.4	Typumwandlung mit dem »as«-Operator	234
4.4	Polymorphie	234
4.4.1	Die »klassische« Methodenimplementierung	235
4.4.2	Abstrakte Methoden	236
4.4.3	Virtuelle Methoden	237
4.5	Weitere Gesichtspunkte der Vererbung	241
4.5.1	Versiegelte Methoden	241
4.5.2	Überladen einer Basisklassenmethode	242
4.5.3	Statische Member und Vererbung	242
4.5.4	Geerbte Methoden ausblenden?	243
4.6	Das Projekt »GeometricObjectsSolution« ergänzen	244
4.6.1	Die Klasse »GeometricObject«	244
4.7	Eingebettete Klassen	247
4.8	Interfaces (Schnittstellen)	248
4.8.1	Einführung in die Schnittstellen	248
4.8.2	Die Schnittstellendefinition	249
4.8.3	Die Schnittstellenimplementierung	250
4.8.4	Die Interpretation der Schnittstellen	255
4.8.5	Änderungen am Projekt »GeometricObjects«	261
4.9	Das Zerstören von Objekten – der Garbage Collector	262
4.9.1	Die Arbeitsweise des Garbage Collectors	262
4.9.2	Expliziter Aufruf des Garbage Collectors	264
4.9.3	Der Destruktor	264
4.9.4	Die »IDisposable«-Schnittstelle	266
4.9.5	Die »using«-Anweisung zum Zerstörung von Objekten	268
4.10	Die Ergänzungen in den Klassen »Circle« und »Rectangle«	269
5	Delegaten und Ereignisse	270
<hr/>		
5.1	Delegaten	270
5.1.1	Einführung in das Prinzip der Delegaten	270
5.1.2	Verwendung von Delegaten	275
5.1.3	Vereinfachter Delegatenaufruf	275

5.1.4	Multicast-Delegaten	275
5.1.5	Anonyme Methoden	277
5.1.6	Kovarianz und Kontravarianz mit Delegaten	280
5.2	Ereignisse eines Objekts	282
5.2.1	Ereignisse bereitstellen	283
5.2.2	Die Reaktion auf ein ausgelöstes Ereignis	286
5.2.3	Allgemeine Betrachtungen der Ereignishandler-Registrierung	288
5.2.4	Wenn der Ereignisempfänger ein Ereignis nicht behandelt	289
5.2.5	Ereignisse mit Übergabeparameter	290
5.2.6	Ereignisse in der Vererbung	294
5.2.7	Ein Blick hinter die Kulissen des Schlüsselworts »event«	295
5.2.8	Die Schnittstelle »INotifyPropertyChanged«	297
5.3	Änderungen im Projekt »GeometricObjects«	298
5.3.1	Überarbeitung des Events »InvalidMeasure«	298
5.3.2	Weitere Ereignisse	299

6 Strukturen und Enumerationen 303

6.1	Strukturen – eine Sonderform der Klassen	303
6.1.1	Die Definition einer Struktur	303
6.1.2	Initialisieren einer Strukturvariablen	304
6.1.3	Konstruktoren in Strukturen	305
6.1.4	Änderung im Projekt »GeometricObjects«	306
6.2	Enumerationen (Aufzählungen)	310
6.2.1	Wertzuweisung an »enum«-Mitglieder	311
6.2.2	Alle Mitglieder einer Aufzählung durchlaufen	311
6.3	Boxing und Unboxing	312

7 Fehlerbehandlung und Debugging 314

7.1	Laufzeitfehler erkennen	315
7.1.1	Die »try ... catch«-Anweisung	317
7.1.2	Behandlung mehrerer Exceptions	319
7.1.3	Die Reihenfolge der »catch«-Zweige	322
7.1.4	Ausnahmen in einer Methodenaufkette	322
7.1.5	Ausnahmen werfen oder weiterleiten	323
7.1.6	Die »finally«-Anweisung	323

7.1.7	Ausnahmefilter	325
7.1.8	Die Klasse »Exception«	326
7.1.9	Benutzerdefinierte Ausnahmen	332
7.2	Debuggen mit Programmcode	337
7.2.1	Einführung	337
7.2.2	Die Klasse »Debug«	338
7.2.3	Die Klasse »Trace«	341
7.2.4	Bedingte Kompilierung	342
7.3	Fehlersuche mit Visual Studio	345
7.3.1	Debuggen im Haltemodus	345
7.3.2	Weitere Alternativen, Variableninhalte zu prüfen	349
8	Auflistungsklassen (Collections)	352
8.1	Collections im Namespace »System.Collections«	352
8.1.1	Die elementaren Schnittstellen der Auflistungsklassen	354
8.2	Die Klasse »ArrayList«	356
8.2.1	Einträge hinzufügen	356
8.2.2	Datenaustausch zwischen einem Array und einer »ArrayList«	359
8.2.3	Die Elemente einer »ArrayList« sortieren	360
8.2.4	Sortieren von Arrays mit »ArrayList.Adapter«	366
8.3	Die Klasse »Hashtable«	367
8.3.1	Methoden und Eigenschaften der Schnittstelle »IDictionary«	368
8.3.2	Beispielprogramm zur Klasse »Hashtable«	369
8.4	Die Klassen »Queue« und »Stack«	374
8.4.1	Die Klasse »Stack«	374
8.4.2	Die Klasse »Queue«	375
8.5	Eigene Auflistungen mit »yield« durchlaufen	376
9	Generics – generische Datentypen	380
9.1	Bereitstellen einer generischen Klasse	382
9.1.1	Mehrere generische Typparameter	384
9.1.2	Vorteile der Generics	384
9.2	Bedingungen (Constraints) festlegen	385
9.2.1	Constraints mit der »where«-Klausel formulieren	385

9.2.2	Typparameter auf Klassen oder Strukturen beschränken	386
9.2.3	Mehrere Constraints definieren	387
9.2.4	Der Konstruktor-Constraint »new()«	387
9.2.5	Das Schlüsselwort »default«	388
9.3	Generische Methoden	389
9.3.1	Methoden und Constraints	390
9.4	Generics und Vererbung	390
9.4.1	Virtuelle generische Methoden	391
9.5	Typkonvertierung von Generics	392
9.6	Generische Delegaten	393
9.6.1	Generische Delegaten und Constraints	394
9.6.2	Anpassung des Beispiels »GeometricObjects«	394
9.7	»Nullable«-Typen	395
9.7.1	Konvertierungen mit »Nullable«-Typen	396
9.7.2	Der »??«-Operator	397
9.8	Generische Collections	397
9.8.1	Die Interfaces der generischen Auflistungsklassen	398
9.8.2	Die generische Auflistungsklasse »List<T>«	398
9.8.3	Vergleiche mit Hilfe des Delegaten »Comparison<T>«	400
9.9	Kovarianz und Kontravarianz generischer Typen	402
9.9.1	Kovarianz mit Interfaces	402
9.9.2	Kontravarianz mit Interfaces	404
9.9.3	Zusammenfassung	405
9.9.4	Generische Delegaten mit varianten Typparametern	406
10	Weitere C#-Sprachfeatures	407
<hr/>		
10.1	Implizit typisierte Variablen	407
10.2	Anonyme Typen	408
10.3	Lambda-Ausdrücke	409
10.3.1	Projektion und Prädikat	411
10.3.2	Expression-bodied Methoden	412
10.4	Erweiterungsmethoden	413
10.4.1	Die Prioritätsregeln	414
10.4.2	Generische Erweiterungsmethoden	416
10.4.3	Richtlinien für Erweiterungsmethoden	416

10.5 Partielle Methoden	417
10.5.1 Wo partielle Methoden eingesetzt werden	419
10.6 Operatorüberladung	420
10.6.1 Einführung	420
10.6.2 Die Syntax der Operatorüberladung	421
10.6.3 Die Operatorüberladungen im Projekt »GeometricObjectsSolution«	422
10.6.4 Die Operatoren »true« und »false« überladen	426
10.6.5 Benutzerdefinierte Konvertierungen	428
10.7 Indexer	432
10.7.1 Überladen von Indexern	434
10.7.2 Parameterbehaftete Eigenschaften	436
10.8 Attribute	439
10.8.1 Das »Flags«-Attribut	441
10.8.2 Benutzerdefinierte Attribute	443
10.8.3 Attribute auswerten	447
10.8.4 Festlegen der Assembly-Eigenschaften in »AssemblyInfo.cs«	450
10.9 Der bedingte NULL-Operator	451
10.10 Der »nameof«-Operator	453
10.10.1 Einsatz in der Anwendung »GeometricObjects«	453
10.11 Dynamisches Binden	454
10.11.1 Eine kurze Analyse	455
10.11.2 Dynamische Objekte	456
10.12 Unsicherer (unsafe) Programmcode – Zeigertechnik in C#	457
10.12.1 Einführung	457
10.12.2 Das Schlüsselwort »unsafe«	458
10.12.3 Die Deklaration von Zeigern	458
10.12.4 Die »fixed«-Anweisung	459
10.12.5 Zeigerarithmetik	460
10.12.6 Der Operator »->«	461
10.13 Das Beispielprogramm »GeometricObjects«	462
11 LINQ	463
<hr/>	
11.1 Einstieg in LINQ?	463
11.1.1 Verzögerte Ausführung	465
11.1.2 LINQ-Erweiterungsmethoden an einem Beispiel	465

11.2 LINQ to Objects	469
11.2.1 Musterdaten	469
11.2.2 Die allgemeine LINQ-Syntax	471
11.3 Die Abfrageoperatoren	472
11.3.1 Übersicht der Abfrageoperatoren	472
11.3.2 Die »from«-Klausel	473
11.3.3 Mit »where« filtern	475
11.3.4 Die Projektionsoperatoren	478
11.3.5 Die Sortieroperatoren	479
11.3.6 Gruppieren mit »GroupBy«	480
11.3.7 Verknüpfungen mit »Join«	482
11.3.8 Die Set-Operatoren-Familie	485
11.3.9 Die Familie der Aggregatoperatoren	487
11.3.10 Quantifizierungsoperatoren	490
11.3.11 Aufteilungsoperatoren	491
11.3.12 Die Elementoperatoren	493
11.3.13 Die Konvertierungsoperatoren	497
12 Arbeiten mit Dateien und Streams	498
<hr/>	
12.1 Einführung	498
12.2 Namespaces der Ein- bzw. Ausgabe	499
12.2.1 Das Behandeln von Ausnahmen bei E/A-Operationen	500
12.3 Laufwerke, Verzeichnisse und Dateien	500
12.3.1 Die Klasse »File«	500
12.3.2 Die Klasse »FileInfo«	506
12.3.3 Die Klassen »Directory« und »DirectoryInfo«	509
12.3.4 Die Klasse »Path«	514
12.3.5 Die Klasse »DriveInfo«	515
12.4 Die »Stream«-Klassen	516
12.4.1 Die abstrakte Klasse »Stream«	517
12.4.2 Die von »Stream« abgeleiteten Klassen im Überblick	520
12.4.3 Die Klasse »FileStream«	521
12.5 Die Klassen »TextReader« und »TextWriter«	527
12.5.1 Die Klasse »StreamWriter«	528
12.5.2 Die Klasse »StreamReader«	532
12.6 Die Klassen »BinaryReader« und »BinaryWriter«	534
12.6.1 Komplexe binäre Dateien	536

13 Binäre Serialisierung	543
13.1 Serialisierungsverfahren	544
13.2 Binäre Serialisierung mit »BinaryFormatter«	545
13.2.1 Die Deserialisierung	547
13.2.2 Serialisierung mehrerer Objekte	548
13.3 Serialisierung mit »XmlSerializer«	550
13.3.1 XML-Serialisierung mit Attributen steuern	553
14 Multithreading	556
14.1 Einführung in das Multithreading	557
14.2 Threads – allgemein betrachtet	558
14.3 Threads erzeugen	560
14.3.1 Die Entwicklung einer einfachen Multithreading-Anwendung	560
14.3.2 Der Delegat »ParameterizedThreadStart«	562
14.3.3 Zugriff eines Threads auf sich selbst	563
14.3.4 Einen Thread für eine bestimmte Zeitdauer anhalten	563
14.3.5 Beenden eines Threads	564
14.3.6 Abhängige Threads – die Methode »Join«	567
14.3.7 Threadprioritäten festlegen	569
14.3.8 Vorder- und Hintergrundthreads	572
14.4 Der Threadpool	572
14.4.1 Ein einfaches Beispielprogramm	573
14.5 Synchronisation von Threads	574
14.5.1 Möglichkeiten der Synchronisation	576
14.5.2 Die Klasse »WaitHandle«	577
14.5.3 Sperren mit »Monitor«	581
14.5.4 Die Klasse »Mutex«	588
14.5.5 Die Klasse »Semaphore«	591
14.5.6 Das Attribut »MethodImpl«	595
14.5.7 Die Klasse »Interlocked«	596
14.5.8 Synchronisation von Threadpool-Threads	596
14.6 Grundlagen asynchroner Methodenaufrufe	597
14.6.1 Asynchroner Methodenaufruf	599
14.6.2 Asynchroner Aufruf mit Rückgabewerten	603
14.6.3 Eine Klasse mit asynchronen Methodenaufrufen	606

15 Die Task Parallel Library (TPL)	610
15.1 Die wichtigsten Klassen der TPL	611
15.2 Die Klasse »Task«	611
15.2.1 Die Konstruktoren eines Tasks	613
15.2.2 Das Erzeugen eines Tasks	614
15.2.3 Daten an einen Task übergeben	616
15.2.4 Auf das Beenden eines Tasks warten	617
15.2.5 Abbruch einer parallelen Operation von außen	619
15.2.6 Fehlerbehandlung	622
15.3 Die Klasse »Parallel«	625
15.3.1 Die Methode »Parallel.Invoke«	625
15.3.2 Schleifen mit »Parallel.For«	625
15.3.3 Den Grad der Parallelität beeinflussen	629
15.3.4 Auflistungen mit »Parallel.ForEach« durchlaufen	630
15.4 Asynchrone Programmierung mit »async« und »await«	631
15.4.1 Die Arbeitsweise von »async« und »await« verstehen	631
15.4.2 Asynchrone Operationen mit Rückgabewert	635
15.4.3 Beispielprogramm	636
15.4.4 Allgemeine und zusammenfassende Betrachtung	638
16 Einige wichtige .NET-Klassen	640
16.1 Die Klasse »Object«	640
16.1.1 Referenzvergleiche mit »Equals« und »ReferenceEquals«	641
16.1.2 »ToString« und »GetType«	641
16.1.3 Die Methode »MemberwiseClone« und das Problem des Klonens	642
16.2 Die Klasse »String«	646
16.2.1 Das Erzeugen eines Strings	647
16.2.2 Die Eigenschaften von »String«	648
16.2.3 Die Methoden der Klasse »String«	648
16.2.4 Zusammenfassung der Klasse »String«	658
16.3 Die Klasse »StringBuilder«	660
16.3.1 Die Kapazität eines »StringBuilder«-Objekts	661
16.3.2 Die Konstruktoren der Klasse »StringBuilder«	662
16.3.3 Die Eigenschaften der Klasse »StringBuilder«	662
16.3.4 Die Methoden der Klasse »StringBuilder«	663
16.3.5 Allgemeine Anmerkungen	665

16.4	Der Typ »DateTime«	666
16.4.1	Die Zeitspanne »Tick«	666
16.4.2	Die Konstruktoren von »DateTime«	667
16.4.3	Die Eigenschaften von »DateTime«	668
16.4.4	Die Methoden der Klasse »DateTime«	669
16.5	Die Klasse »TimeSpan«	670
16.6	Ausgabeformatierung	673
16.6.1	Formatierung mit der Methode »String.Format«	673
16.6.2	Formatierung mit der Methode »ToString«	677
16.6.3	Benutzerdefinierte Formatierung	678
17	Projektmanagement und Visual Studio 2015	681
<hr/>		
17.1	Der Projekttyp »Klassenbibliothek«	681
17.1.1	Mehrere Projekte in einer Projektmappe verwalten	682
17.1.2	Die Zugriffsmodifizierer »public« und »internal«	683
17.1.3	Friend Assemblies	684
17.1.4	Einbinden einer Klassenbibliothek	684
17.2	Assemblies	685
17.2.1	Ein Überblick über das Konzept der Assemblies	685
17.2.2	Allgemeine Beschreibung privater und globaler Assemblies	687
17.2.3	Die Struktur einer Assembly	687
17.2.4	Globale Assemblies	692
17.3	Konfigurationsdateien	698
17.3.1	Die verschiedenen Konfigurationsdateien	698
17.3.2	Die Struktur einer Anwendungskonfigurationsdatei	700
17.3.3	Spezifische Einträge in der Anwendungskonfigurationsdatei	703
17.3.4	Einträge der Anwendungskonfigurationsdatei auswerten	704
17.3.5	Editierbare, anwendungsbezogene Einträge mit <appSettings>	709
17.4	Versionierung einer Assembly	711
17.4.1	Die Herausgeberichtliniendatei	713
17.5	XML-Dokumentation	714
17.5.1	Das Prinzip der XML-Dokumentation	714
17.5.2	Die XML-Kommentar-Tags	716
17.5.3	Generieren der XML-Dokumentationsdatei	718
17.6	Der Klassendesigner (Class Designer)	719
17.6.1	Ein typisches Klassendiagramm	719

17.6.2	Hinzufügen von Klassendiagrammen	721
17.6.3	Die Toolbox des Klassendesigners	721
17.6.4	Das Fenster »Klassendetails«	722
17.6.5	Klassendiagramme als Bilder exportieren	724
17.7	Refactoring	724
17.7.1	Methode extrahieren	725
17.7.2	Bezeichner umbenennen	727
17.7.3	Felder inkapseln	727
17.8	Code-Snippets (Codeausschnitte)	728
17.8.1	Codeausschnitte einfügen	729
17.8.2	Die Anatomie eines Codeausschnitts	729
18	Einführung in die WPF und XAML	732
<hr/>		
18.1	Die Merkmale einer WPF-Anwendung	733
18.1.1	Anwendungstypen	734
18.1.2	Eine WPF-Anwendung und ihre Dateien	735
18.1.3	Ein erstes WPF-Beispiel	738
18.1.4	Wichtige WPF-Features	741
18.1.5	Der logische und der visuelle Elementbaum	744
18.2	XAML (Extended Application Markup Language)	746
18.2.1	Die Struktur einer XAML-Datei	746
18.2.2	Eigenschaften eines XAML-Elements in Attribut-Schreibweise festlegen	748
18.2.3	Eigenschaften im Eigenschaftsfenster festlegen	749
18.2.4	Die Eigenschaft-Element-Syntax	749
18.2.5	Inhaltseigenschaften	750
18.2.6	Typkonvertierung	754
18.2.7	Markup-Erweiterungen (Markup Extensions)	755
18.2.8	XML-Namespaces	758
18.2.9	XAML-Spracherweiterungen	761
19	Die WPF-Layoutcontainer	763
<hr/>		
19.1	Allgemeiner Überblick	763
19.2	Gemeinsame Eigenschaften der Layoutcontainer	764
19.3	Die Layoutcontainer im Detail	765

19.3.1	Das »Canvas«	765
19.3.2	Das »StackPanel«	766
19.3.3	Das »WrapPanel«	769
19.3.4	Das »DockPanel«	770
19.3.5	Das »Grid«-Steuerelement	772
19.3.6	Das »UniformGrid«	778
19.4	Verschachteln von Layoutcontainern	779

20 Fenster in der WPF 782

20.1	Hosts der WPF	782
20.2	Fenster vom Typ »Window«	783
20.2.1	Mehrere Fenster in einer Anwendung	785
20.3	Fenster vom Typ »NavigationWindow«	787
20.3.1	Das »Page«-Element	789
20.3.2	Navigation zwischen den Seiten	791
20.3.3	Der Verlauf der Navigation – das Journal	794
20.3.4	Datenübergabe zwischen den Seiten mit einem Konstruktor	796
20.3.5	Datenübergabe mit der Methode »Navigate«	797
20.3.6	Navigation im Internet	799
20.3.7	Navigieren mit dem Ereignis »RequestNavigate« des »HyperLink«-Elements	799
20.4	Hosts vom Typ »Frame«	800
20.5	Nachrichtenfenster mit »MessageBox«	801
20.5.1	Die Methode »MessageBox.Show«	802
20.6	Standarddialoge in der WPF	805
20.6.1	Der Dialog »OpenFileDialog«	805
20.6.2	Der Dialog »SaveFileDialog«	808

21 WPF-Steuerelemente 809

21.1	Die Hierarchie der WPF-Komponenten	809
21.2	Allgemeine Eigenschaften der WPF-Steuerelemente	810
21.2.1	Den Außenrand mit der Eigenschaft »Margin« festlegen	811
21.2.2	Den Innenrand mit der Eigenschaft »Padding« festlegen	811

21.2.3	Die Eigenschaft »Content«	812
21.2.4	Die Größe einer Komponente	813
21.2.5	Die Ausrichtung einer Komponente	815
21.2.6	Die Sichtbarkeit eines Steuerelements	816
21.2.7	Die Schriften	817
21.3	Die Gruppe der Schaltflächen	818
21.3.1	Die Basisklasse »ButtonBase«	818
21.3.2	Das Steuerelement »Button«	819
21.3.3	Das Steuerelement »ToggleButton«	820
21.3.4	Das Steuerelement »RepeatButton«	821
21.3.5	Das Steuerelement »CheckBox«	822
21.3.6	Das Steuerelement »RadioButton«	822
21.4	Einfache Eingabesteuerelemente	823
21.4.1	Das Steuerelement »Label«	823
21.4.2	Das Steuerelement »TextBox«	825
21.4.3	Das Steuerelement »PasswordBox«	828
21.4.4	Das Steuerelement »TextBlock«	828
21.5	WPF-Listenelemente	832
21.5.1	Das Steuerelement »ListBox«	833
21.5.2	Die »ComboBox«	835
21.5.3	Das Steuerelement »ListView«	836
21.5.4	Das Steuerelement »TreeView«	839
21.5.5	Das Steuerelement »TabControl«	845
21.5.6	Die Menüleiste	846
21.5.7	Das Kontextmenü	849
21.5.8	Symbolleisten	851
21.5.9	Die Statusleiste	854
21.6	Weitere Steuerelemente	855
21.6.1	Das Steuerelement »Tooltip«	855
21.6.2	Die »ProgressBar«	857
21.6.3	Das Steuerelement »Slider«	857
21.6.4	Das »GroupBox«-Steuerelement	858
21.6.5	Das Steuerelement »ScrollViewer«	859
21.6.6	Das Steuerelement »Expander«	861
21.6.7	Das Steuerelement »Border«	862
21.6.8	Die »Image«-Komponente	863
21.6.9	»Calendar« und »DatePicker« zur Datumsangabe	864
21.6.10	Das Steuerelement »InkCanvas«	866

21.7	Das »Ribbon«-Steuerelement	869
21.7.1	Voraussetzungen für den Zugriff auf das »Ribbon«-Control	869
21.7.2	Ein kurzer Überblick	870
21.7.3	Der XAML-Code	870
21.8	Komponenten vom Typ »FlowDocument«	875
21.8.1	Eigenschaften von »FlowDocument«	876
21.8.2	Die Böcke eines »FlowDocument«-Objekts	877
21.8.3	»Inline«-Elemente	881
21.8.4	»FlowDocument«-Objekte mit Code erzeugen	884
21.8.5	Speichern und Laden eines »FlowDocument«-Objekts	887
21.9	Das Element »FlowDocumentViewer«	887
21.9.1	Das Anzeigeelement »FlowDocumentScrollViewer«	888
21.9.2	Das Anzeigeelement »FlowDocumentPageViewer«	888
21.9.3	Das Anzeigeelement »FlowDocumentReader«	889
21.9.4	Allgemeines zum XPS-Format	889
21.9.5	Beispielprogramm	890
21.10	Das Steuerelement »RichTextBox«	891
21.10.1	Formatieren des Inhalts	892
21.10.2	Laden und Speichern	893

22 Dependency Properties 897

22.1	Die Charakteristik von Abhängigkeitseigenschaften	897
22.2	Den Wert einer Abhängigkeitseigenschaft bilden	898
22.3	Definition einer Dependency Property	899
22.3.1	Registrieren einer Abhängigkeitseigenschaft	900
22.3.2	Der Eigenschaftswrapper	901
22.3.3	Die Eigenschaftsmetadaten	902
22.3.4	Freigabe des spezifischen Eigenschaftswertes	906
22.3.5	Vererbung von Abhängigkeitseigenschaften	907
22.4	Validieren einer Abhängigkeitseigenschaft	907
22.4.1	Validieren mit »ValidateValueCallback«	908
22.4.2	Validieren mit »CoerceValueCallback«	908
22.5	Angehängte Eigenschaften (Attached Property)	909
22.5.1	Angehängte Eigenschaften zur Laufzeit ändern	911

23	Ereignisse in der WPF	912
23.1	Ereignishandler bereitstellen	912
23.2	Routing-Strategien	913
23.2.1	Routed Events und der Elementbaum	915
23.2.2	Beispielanwendung	916
23.2.3	Sonderfall: Ereignisse mit der linken Maustaste	917
23.3	Der Ereignishandler	919
23.3.1	Die Klasse »RoutedEventArgs«	919
23.3.2	Die Quelle des Routing-Prozesses	920
23.3.3	Die Eigenschaft »Handled«	921
23.3.4	Registrieren und Deregistrieren eines Ereignishandlers mit Code	922
23.4	Benutzerdefinierte Routed Events	922
23.4.1	Ereignisauslösung	924
23.4.2	Das gebubbelte Ereignis im Elementbaum verwenden	925
23.5	Mausereignisse in der WPF	926
23.5.1	Ziehen der Maus	926
23.5.2	Auswerten der Mausklicks	927
23.5.3	Capturing	928
24	WPF-Datenbindung	931
24.1	Einführungsbeispiel	931
24.2	Bindungsalternativen	934
24.2.1	Die Eigenschaft »Source«	934
24.2.2	Binden an relative Datenquellen	935
24.2.3	Bindungen mit »DataContext«	936
24.3	Die Klasse »Binding«	937
24.3.1	Die Eigenschaft »Path« des »Binding«-Objekts	938
24.3.2	Die Bindungsrichtung festlegen	940
24.3.3	Aktualisieren der Bindung	942
24.3.4	Die Eigenschaft »Converter« einer Bindung	945
24.3.5	Beenden einer Bindung	953
24.4	Validieren von Bindungen	953
24.4.1	Die Validierung im Datenobjekt	954
24.4.2	Eine benutzerdefinierte »ValidationRule«	956
24.4.3	Fehlerhinweise individuell gestalten	958

24.4.4	Ereignisauslösung bei einem Validierungsfehler	960
24.4.5	Mehrere Controls mit »BindingGroup« gleichzeitig validieren	961
24.4.6	Validierung mit der Schnittstelle »IDataErrorInfo«	964
24.5	Binden und Aktualisieren von CLR-Objekten	966
24.5.1	Ein Objekt mit XAML-Code erzeugen und binden	967
24.5.2	Ein Objekt mit C#-Code erzeugen und binden	968
24.5.3	Aktualisieren benutzerdefinierter Objekte	970
24.6	Alternative Datenbindungen	973
24.6.1	Die Klasse »ObjectDataProvider«	973
25	Ressourcen und Styles	977
25.1	Binäre Ressourcen	977
25.1.1	Zugriff auf binäre Ressourcen	978
25.1.2	Zugriff auf binäre Ressourcen mit C#	980
25.2	Logische Ressourcen	980
25.2.1	Die Suche nach einer Ressource	981
25.2.2	Definieren einer Ressource	982
25.2.3	Zugriff auf eine logische Ressource mit C#-Code	983
25.2.4	»StaticResource« vs. »DynamicResource«	984
25.2.5	Anbinden einer dynamischen Ressource mit C#-Code	986
25.2.6	Sonderfall: WPF-Elemente als Ressourcen	986
25.2.7	Anwendungsübergreifende Ressourcen	988
25.2.8	Abrufen von Systemressourcen	990
25.3	Styles	992
25.3.1	Einfache Styles (explizite Styles)	992
25.3.2	Typisierte Styles (implizite Styles)	995
25.3.3	Erweitern eines Styles mit »BasedOn«	997
25.3.4	Ereignisse mit »EventSetter« zentral abonnieren	999
25.4	Trigger	1001
25.4.1	Einfache Trigger (Eigenschaftstrigger)	1002
25.4.2	Mehrere Bedingungen mit »MultiTrigger«	1004
25.4.3	»DataTrigger«	1006
25.4.4	»MultiDataTrigger«	1007
25.4.5	»EventTrigger«	1009
25.5	Templates	1011
25.5.1	Allgemeines zu »ControlTemplate«-Elementen	1012
25.5.2	Definition innerhalb eines Styles	1017

25.6	Ermitteln des visuellen Elementbaums	1018
25.6.1	Das Tool »Expression Blend«	1018
26	Weitere Möglichkeiten der Datenbindung	1021
26.1	»ItemsControl«-Steuerelemente	1022
26.2	Binden an ein »ListBox«-Element	1023
26.2.1	Die Klasse »ObservableCollection<T>«	1025
26.2.2	Die Darstellung eines »ListBoxItem«-Elements anpassen	1027
26.2.3	Datendarstellung mit »DataTemplate«-Objekten festlegen	1029
26.2.4	»DataTemplate« mit Trigger	1031
26.3	Datenbindung an ADO.NET- und LINQ-Datenquellen	1034
26.3.1	Das Binden an ADO.NET-Objekte	1034
26.3.2	Das Binden an LINQ-Ausdrücke	1035
26.4	Navigieren, Filtern, Sortieren und Gruppieren	1036
26.4.1	Navigieren in einer Datenmenge	1037
26.4.2	Sortieren von Datenmengen	1041
26.4.3	Filtern von Daten	1042
26.4.4	Gruppieren von Daten	1046
26.5	Das Steuerelement »DataGrid«	1050
26.5.1	Elementare Eigenschaften des »DataGrid«-Objekts	1052
26.5.2	Spalten definieren	1053
26.5.3	Details einer Zeile anzeigen	1059
27	2D-Grafik	1061
27.1	Shapes	1061
27.1.1	Allgemeine Beschreibung	1061
27.1.2	»Line«-Elemente	1062
27.1.3	»Ellipse«- und »Rectangle«-Elemente	1063
27.1.4	»Polygon« - und »Polyline« -Elemente	1063
27.1.5	Darstellung der Linien	1063
27.2	Path-Elemente	1065
27.2.1	Das Element »GeometryGroup«	1066
27.2.2	Das Element »CombinedGeometry«	1067
27.2.3	Geometrische Figuren mit »PathGeometry«	1068

27.3 »Brush«-Objekte	1069
27.3.1 »SolidColorBrush«	1070
27.3.2 »LinearGradientBrush«	1071
27.3.3 »RadialGradientBrush«	1073
27.3.4 Muster mit »TileBrush«	1074
27.3.5 Bilder mit »ImageBrush«	1077
27.3.6 Effekte mit »VisualBrush«	1078
27.3.7 Das Element »DrawingBrush«	1079

28 WPF – weitergehende Techniken 1081

28.1 WPF und Multithreading	1081
28.1.1 Nachrichtenschleife und »Dispatcher«-Klasse	1082
28.1.2 »BeginInvoke« und »Invoke«	1084
28.1.3 Die »DispatcherObject«-Klasse	1085
28.2 Globalisierung und Lokalisierung	1086
28.2.1 Globalisierung	1086
28.2.2 Lokalisierung	1087

29 WPF-Commands 1102

29.1 Allgemeine Beschreibung	1102
29.2 Ein erstes Programmbeispiel	1103
29.3 Die Befehlsquelle	1106
29.3.1 Das Befehlsziel mit »CommandTarget« angeben	1107
29.3.2 Einen Fokusbereich mit der Klasse »FocusManager« definieren	1108
29.3.3 Zusätzliche Daten mit »CommandParameter« bereitstellen	1109
29.4 WPF-Commands	1109
29.4.1 Die Arbeitsweise eines Befehls	1110
29.4.2 Die Klasse »CommandManager«	1111
29.5 »RoutedCommand«-Objekte und »CommandBindings«	1115
29.5.1 Vordefinierte WPF-Commands	1115
29.5.2 Die Klasse »RoutedCommand«	1116
29.5.3 Befehlsbindungen mit »CommandBinding« einrichten	1117
29.5.4 Befehlsbindung mit Programmcode	1119
29.5.5 Befehle mit Maus oder Tastatur aufrufen	1120
29.5.6 Benutzerdefinierter »RoutedCommand«	1122

30 Das MVVM-Pattern	1124
30.1 Die Theorie hinter dem Model-View-ViewModel-Pattern	1124
30.2 Allgemeine Beschreibung des Beispielprogramms	1125
30.3 Der Ausgangspunkt im Beispiel »MVVM_Origin«	1127
30.4 Das Bereitstellen des Modells	1127
30.5 Bereitstellen des ViewModels	1129
30.5.1 Abrufen und Bereitstellen der Daten	1130
30.5.2 Die Anbindung der Daten an die »ListView« der View	1132
30.5.3 Die Anbindung der Textboxen	1133
30.6 WPF-Commands und Eigenschaften im ViewModel	1133
30.6.1 Die Umsetzung von Commands im Model-View-ViewModel	1134
30.6.2 Die allgemeine Beschreibung eines Commands mit »RelayCommand«	1135
30.6.3 Ergänzen der Klasse »MainViewModel«	1136
30.6.4 Die aktuelle Position des Datensatzzeigers	1138
30.7 »RoutedCommand«-Objekte im MVVM	1139
30.7.1 Änderungen im »MainWindow«	1140
30.7.2 Ergänzungen im ViewModel	1141
30.7.3 Die Ereignishandler der »CommandBinding«-Objekte	1143
30.8 Beliebige Ereignisse mit »EventTrigger«-Objekten behandeln	1144
30.8.1 Mausereignisse Triggern	1144
30.8.2 Ergänzung des ViewModels	1147
30.9 Die Klasse »Person« durch ein ViewModel kapseln	1148
30.9.1 Die Model-spezifische Klasse »PString«	1149
30.9.2 Die Model-spezifische Klasse »PDateTime«	1150
30.9.3 Die Klasse »PersonViewModel«	1152
30.9.4 Notwendige Anpassungen in »MainViewModel«	1155
30.9.5 Anpassungen im XAML-Code	1157
30.10 Die Schaltflächen »Rückgängig« und »Speichern«	1159
30.10.1 Eine Änderung zurücknehmen	1159
30.10.2 Die Änderungen der Listenobjekte speichern	1161
30.11 Ein Control in der View fokussieren	1165
30.11.1 Erste Überlegungen	1165
30.11.2 Definition der angehängten Eigenschaft	1166
30.11.3 Die angehängte Eigenschaft im XAML-Code	1167
30.11.4 Das ViewModel ergänzen	1168

30.12 Die Listenelemente sortieren	1169
30.12.1 Ergänzungen im XAML-Code	1169
30.12.2 Ergänzungen im ViewModel	1169
30.12.3 Die Klassen »PString« und »PDateTime« anpassen	1170
30.13 Ereignisse im ViewModel auslösen	1171
30.13.1 Die Löschestätigung	1172
30.13.2 Das Schließen des Fensters	1173

31 Benutzerdefinierte Controls 1175

31.1 Erstellen eines benutzerdefinierten Steuerelements	1175
31.2 Der XAML-Code des »UserControl«-Elements	1177
31.3 Die Programmlogik des Steuerelements	1178
31.3.1 Die Eigenschaften	1178
31.3.2 Die Methoden von »ColorMixer«	1180
31.3.3 Ein Ereignis bereitstellen	1181
31.3.4 Das Steuerelement um ein »Command« ergänzen	1182
31.4 Das Steuerelement »ColorMixer« testen	1183

32 Datenbankzugriff mit ADO.NET 1186

32.1 Vorbereitung	1186
32.2 Einleitung in ADO.NET	1187
32.2.1 Die Datenprovider	1188
32.3 Die Verbindung zu einer Datenbank herstellen	1189
32.3.1 Die Verbindungszeichenfolge	1190
32.3.2 Öffnen und Schließen einer Verbindung	1195
32.3.3 Das Verbindungspooling	1200
32.3.4 Die Ereignisse eines »Connection«-Objekts	1203
32.3.5 Verbindungszeichenfolgen aus einer Konfigurationsdatei abrufen	1205
32.3.6 Die Klasse »SqlConnection« im Überblick	1207
32.3.7 Verbindungen mit dem OLE-DB-Datenprovider	1209
32.4 Die Datenbankabfrage	1212
32.4.1 Das »SqlCommand«-Objekt	1212
32.4.2 Abfragen, die genau ein Ergebnis liefern	1217

32.4.3	Das »SqlDataReader«-Objekt	1218
32.4.4	Datensätze einlesen	1219
32.4.5	Schließen des »SqlDataReader«-Objekts	1221
32.4.6	MARS (Multiple Active Result Sets)	1222
32.4.7	Batch-Abfragen mit »NextResult« durchlaufen	1223
32.4.8	Das Schema eines »SqlDataReader«-Objekts untersuchen	1224
32.5	Parametrisierte Abfragen	1227
32.5.1	Parametrisierte Abfragen mit dem SqlClient-Datenprovider	1227
32.5.2	Die Klasse »SqlParameter«	1229
32.5.3	Asynchrone Abfragen (klassisch)	1230
32.5.4	Gespeicherte Prozeduren (Stored Procedures)	1235
32.6	Der »SqlDataAdapter«	1242
32.6.1	Ein Programmbeispiel	1243
32.6.2	Die Konstruktoren der Klasse »DataAdapter«	1244
32.6.3	Die Eigenschaft »SelectCommand«	1244
32.6.4	Den lokalen Datenspeicher mit »Fill« füllen	1245
32.6.5	Mehrere »DataAdapter«-Objekte aufrufen	1247
32.6.6	Tabellenzuordnung mit »TableMappings«	1249
32.6.7	Das Ereignis »FillError« des »SqlDataAdapter«-Objekts	1253
32.7	Daten im lokalen Speicher – das »DataSet«	1254
32.7.1	Verwenden des »DataSet«-Objekts	1255
32.7.2	Dateninformationen in eine XML-Datei schreiben	1259
32.7.3	Dem »DataSet« Schemainformationen übergeben	1260
32.7.4	Schemainformationen bereitstellen	1262
32.7.5	Eigenschaften einer »DataColumn«, die der Gültigkeitsprüfung dienen	1262
32.7.6	Die Constraints-Klassen einer »DataTable«	1263
32.7.7	Das Schema mit Programmcode erzeugen	1264
32.7.8	Schemainformationen mit »SqlDataAdapter« abrufen	1266
32.7.9	Änderungen in einer »DataTable« vornehmen	1269
32.7.10	Was bei einer Änderung einer Datenzeile passiert	1274
32.7.11	Manuelles Steuern der Eigenschaft »DataRowState«	1278
32.8	Mit mehreren Tabellen arbeiten	1279
32.8.1	Der Weg über JOIN-Abfragen	1279
32.8.2	Mehrere Tabellen in einem »DataSet«	1281
32.8.3	Eine »DataRelation« erzeugen	1281
32.8.4	»DataRelation«-Objekte und Einschränkungen	1282
32.8.5	In Beziehung stehende Daten suchen	1284
32.8.6	Ergänzung zum Speichern von Schemainformationen in einer XML-Schema-Datei	1286

32.9 Aktualisieren der Datenbank	1287
32.9.1 Aktualisieren mit dem »CommandBuilder«-Objekt	1287
32.9.2 Manuell gesteuerte Aktualisierungen	1290
32.9.3 Aktualisieren mit »ExecuteNonQuery«	1291
32.9.4 Manuelles Aktualisieren mit dem »DataAdapter«	1298
32.9.5 Den zu aktualisierenden Datensatz in der Datenbank suchen	1302
32.9.6 Den Benutzer über fehlgeschlagene Aktualisierungen informieren	1306
32.9.7 Konfliktverursachende Datenzeilen bei der Datenbank abfragen	1309
32.10 Objekte vom Typ »DataView«	1315
32.10.1 Eine »DataView« erzeugen	1315
32.10.2 Auf die Datenzeilen in einer »DataView« zugreifen	1316
32.10.3 Die Eigenschaft »Sort« und die Methode »Find«	1316
32.10.4 Die Methode »FindRows«	1317
32.10.5 Die Eigenschaft »RowFilter«	1317
32.10.6 Die Eigenschaft »RowStateFilter«	1318
32.10.7 Änderungen an einem »DataView«-Objekt	1318
32.10.8 Aus einer »DataView« eine »DataTable« erzeugen	1319
32.11 Stark typisierte »DataSet«-Objekte	1321
32.11.1 Ein stark typisiertes »DataSet« erzeugen	1321
32.11.2 Die Anatomie eines typisierten »DataSet«-Objekts	1324
32.11.3 Typisierte »DataSet«-Objekte manuell im Designer erzeugen	1331
32.11.4 Weitergehende Betrachtungen	1333
32.11.5 Der »TableAdapter«	1333
32.11.6 Einen »TableAdapter« mit Visual Studio erzeugen	1334
32.11.7 »TableAdapter« im Code verwenden	1339
32.12 Fazit: Typisierte oder nicht typisierte »DataSet«-Objekte?	1345
33 Das Entity Framework (EF)	1346
<hr/>	
33.1 Das Entity Framework im Überblick	1346
33.1.1 Die Organisation der Daten im Entity Framework	1348
33.1.2 Ein erstes Entity Data Model (EDM) erstellen	1349
33.1.3 Das Entity Data Model im Designer	1353
33.1.4 Assoziationen im Entity Data Model	1357
33.1.5 Der Kontext der Entitäten	1358
33.1.6 Der Aufbau des Entity Data Models	1359
33.1.7 Die Klassen des Entity Data Models (EDM)	1363
33.1.8 Die Architektur des Entity Frameworks	1370

33.2	Abfragen im Kontext des EDM	1372
33.2.1	Abfragen mit LINQ	1373
33.2.2	In Beziehung stehende Daten laden	1390
33.2.3	Abfragen mit Entity SQL	1396
33.2.4	Der EntityClient-Provider	1401
33.2.5	Abfrage-Generator-Methoden (QueryBuilder-Methoden)	1404
33.2.6	SQL-Direktabfragen	1405
33.3	Aktualisieren von Entitäten	1406
33.3.1	Entitäten ändern	1406
33.3.2	Hinzufügen neuer Entitäten	1408
33.3.3	Löschen einer Entität	1413
33.4	Der Lebenszyklus einer Entität	1415
33.4.1	Das Team der Objekte, die den Zustand verwalten	1416
33.4.2	Neue Entitäten im Objektkontext	1417
33.4.3	Die Zustände einer Entität	1419
33.4.4	Zusätzliche Entitäten in den Datencache laden	1420
33.4.5	Das »ObjectStateEntry«-Objekt	1423
33.4.6	Die Klasse »EntityKey«	1428
33.4.7	Komplexere Szenarien	1430
33.5	Konflikte behandeln	1433
33.5.1	Allgemeine Betrachtungen	1434
33.5.2	Konkurrierende Zugriffe mit dem Entity Framework	1435
34	Die DbContext-API	1442
<hr/>		
34.1	Datenabfragen mit »DbContext«	1442
34.1.1	Einfache Datenabfragen	1443
34.1.2	Eine Entität mit »DbSet<>.Find« suchen	1444
34.1.3	Lokale Daten mit »Load« und »Local«	1445
34.1.4	In Beziehung stehende Daten laden	1447
34.2	Ändern von Entitäten	1452
34.2.1	Entitäten ändern	1452
34.2.2	Hinzufügen einer neuen Entität	1453
34.2.3	Löschen einer Entität	1454
34.3	Change Tracking (Änderungsnachverfolgung)	1457
34.3.1	Snapshot Change Tracking	1458
34.3.2	Change-Tracking Proxies	1461

34.4 Kontextlose Entitäten ändern	1463
34.4.1 Entitätszustände	1464
34.4.2 »DbContext« eine neue Entität hinzufügen	1465
34.4.3 »DbContext« eine geänderte Entität hinzufügen	1466
34.4.4 »DbContext« eine zu löschende Entität angeben	1467
Index	1469