

Inhalt

Vorwort	XXI
TEIL I: Grundlagen	1
1 Einstieg in Visual Studio 2017	3
1.1 Die Installation von Visual Studio 2017	3
1.1.1 Überblick über die Produktpalette	3
1.1.2 Anforderungen an Hard- und Software	4
1.2 Unser allererstes C#-Programm	5
1.2.1 Vorbereitungen	5
1.2.2 Quellcode schreiben	7
1.2.3 Programm kompilieren und testen	7
1.2.4 Einige Erläuterungen zum Quellcode	8
1.2.5 Konsolenanwendungen sind out	9
1.3 Die Windows-Philosophie	10
1.3.1 Mensch-Rechner-Dialog	10
1.3.2 Objekt- und ereignisorientierte Programmierung	10
1.3.3 Programmieren mit Visual Studio 2017	12
1.4 Die Entwicklungsumgebung Visual Studio 2017	13
1.4.1 Neues Projekt	13
1.4.2 Die wichtigsten Fenster	14
1.5 Microsofts .NET-Technologie	18
1.5.1 Zur Geschichte von .NET	18
1.5.2 .NET-Features und Begriffe	20
1.6 Praxisbeispiele	28
1.6.1 Unsere erste Windows-Forms-Anwendung	28
1.6.2 Umrechnung Euro-Dollar	32
2 Grundlagen der Sprache C#	41
2.1 Grundbegriffe	41
2.1.1 Anweisungen	41

2.1.2	Bezeichner	42
2.1.3	Schlüsselwörter	43
2.1.4	Kommentare	44
2.2	Datentypen, Variablen und Konstanten	45
2.2.1	Fundamentale Typen	45
2.2.2	Werttypen versus Verweistypen	46
2.2.3	Benennung von Variablen	47
2.2.4	Deklaration von Variablen	47
2.2.5	Typsuffixe	49
2.2.6	Zeichen und Zeichenketten	49
2.2.7	object-Datentyp	52
2.2.8	Konstanten deklarieren	53
2.2.9	Nullable Types	53
2.2.10	Typinferenz	54
2.2.11	Gültigkeitsbereiche und Sichtbarkeit	55
2.3	Konvertieren von Datentypen	56
2.3.1	Implizite und explizite Konvertierung	56
2.3.2	Welcher Datentyp passt zu welchem?	58
2.3.3	Konvertieren von string	58
2.3.4	Die Convert-Klasse	61
2.3.5	Die Parse-Methode	61
2.3.6	Boxing und Unboxing	62
2.4	Operatoren	63
2.4.1	Arithmetische Operatoren	64
2.4.2	Zuweisungsoperatoren	66
2.4.3	Logische Operatoren	67
2.4.4	Rangfolge der Operatoren	69
2.5	Kontrollstrukturen	71
2.5.1	Verzweigungsbefehle	71
2.5.2	Schleifenanweisungen	74
2.6	Benutzerdefinierte Datentypen	77
2.6.1	Enumerationen	77
2.6.2	Strukturen	79
2.7	Nutzerdefinierte Methoden	81
2.7.1	Methoden mit Rückgabewert	82
2.7.2	Methoden ohne Rückgabewert	83
2.7.3	Parameterübergabe mit ref	84
2.7.4	Parameterübergabe mit out	85
2.7.5	Methodenüberladung	86
2.7.6	Optionale Parameter	87
2.7.7	Benannte Parameter	89
2.8	Praxisbeispiele	90
2.8.1	Vom PAP zur Konsolenanwendung	90
2.8.2	Ein Konsolen- in ein Windows-Programm verwandeln	92

2.8.3	Schleifenanweisungen verstehen	94
2.8.4	Benutzerdefinierte Methoden überladen	96
2.8.5	Anwendungen von Visual Basic nach C# portieren	99
3	OOP-Konzepte	107
3.1	Kleine Einführung in die OOP	107
3.1.1	Historische Entwicklung	108
3.1.2	Grundbegriffe der OOP	109
3.1.3	Sichtbarkeit von Klassen und ihren Mitgliedern	111
3.1.4	Allgemeiner Aufbau einer Klasse	112
3.1.5	Das Erzeugen eines Objekts	114
3.1.6	Einführungsbeispiel	117
3.2	Eigenschaften	121
3.2.1	Eigenschaften mit Zugriffsmethoden kapseln	121
3.2.2	Berechnete Eigenschaften	123
3.2.3	Lese-/Schreibschutz	125
3.2.4	Property-Accessoren	126
3.2.5	Statische Felder/Eigenschaften	126
3.2.6	Einfache Eigenschaften automatisch implementieren	129
3.3	Methoden	130
3.3.1	Öffentliche und private Methoden	130
3.3.2	Überladene Methoden	131
3.3.3	Statische Methoden	132
3.4	Ereignisse	134
3.4.1	Ereignis hinzufügen	134
3.4.2	Ereignis verwenden	137
3.5	Arbeiten mit Konstruktor und Destruktor	140
3.5.1	Konstruktor und Objektinitialisierer	141
3.5.2	Destruktor und Garbage Collector	144
3.5.3	Mit using den Lebenszyklus des Objekts kapseln	146
3.5.4	Verzögerte Initialisierung	147
3.6	Vererbung und Polymorphie	148
3.6.1	Klassendiagramm	148
3.6.2	Method-Overriding	149
3.6.3	Klassen implementieren	149
3.6.4	Implementieren der Objekte	153
3.6.5	Ausblenden von Mitgliedern durch Vererbung	154
3.6.6	Allgemeine Hinweise und Regeln zur Vererbung	156
3.6.7	Polymorphes Verhalten	157
3.6.8	Die Rolle von System.Object	160
3.7	Spezielle Klassen	161
3.7.1	Abstrakte Klassen	161
3.7.2	Versiegelte Klassen	163

3.7.3	Partielle Klassen	163
3.7.4	Statische Klassen	165
3.8	Schnittstellen (Interfaces)	165
3.8.1	Definition einer Schnittstelle	166
3.8.2	Implementieren einer Schnittstelle	166
3.8.3	Abfragen, ob Schnittstelle vorhanden ist	167
3.8.4	Mehrere Schnittstellen implementieren	168
3.8.5	Schnittstellenprogrammierung ist ein weites Feld	168
3.9	Praxisbeispiele	169
3.9.1	Eigenschaften sinnvoll kapseln	169
3.9.2	Eine statische Klasse anwenden	172
3.9.3	Vom fetten zum schlanken Client	174
3.9.4	Schnittstellenvererbung verstehen	184
3.9.5	Rechner für komplexe Zahlen	189
3.9.6	Sortieren mit Comparable/Comparer	198
3.9.7	Einen Objektbaum in generischen Listen abspeichern	202
3.9.8	OOP beim Kartenspiel erlernen	208
3.9.9	Eine Klasse zur Matrizenrechnung entwickeln	213
4	Arrays, Strings, Funktionen	219
4.1	Datenfelder (Arrays)	219
4.1.1	Array deklarieren	220
4.1.2	Array instanziiieren	220
4.1.3	Array initialisieren	221
4.1.4	Zugriff auf Array-Elemente	222
4.1.5	Zugriff mittels Schleife	223
4.1.6	Mehrdimensionale Arrays	224
4.1.7	Zuweisen von Arrays	226
4.1.8	Arrays aus Strukturvariablen	227
4.1.9	Löschen und Umdimensionieren von Arrays	228
4.1.10	Eigenschaften und Methoden von Arrays	229
4.1.11	Übergabe von Arrays	231
4.2	Verarbeiten von Zeichenketten	232
4.2.1	Zuweisen von Strings	232
4.2.2	Eigenschaften und Methoden von String-Variablen	233
4.2.3	Wichtige Methoden der String-Klasse	236
4.2.4	Die StringBuilder-Klasse	238
4.3	Reguläre Ausdrücke	241
4.3.1	Wozu werden reguläre Ausdrücke verwendet?	241
4.3.2	Eine kleine Einführung	241
4.3.3	Wichtige Methoden/Eigenschaften der Klasse Regex	242
4.3.4	Kompilierte reguläre Ausdrücke	244
4.3.5	RegexOptions-Enumeration	245
4.3.6	Metazeichen (Escape-Zeichen)	246

4.3.7	Zeichenmengen (Character Sets)	247
4.3.8	Quantifizierer	249
4.3.9	Zero-Width Assertions	250
4.3.10	Gruppen	254
4.3.11	Text ersetzen	254
4.3.12	Text splitten	255
4.4	Datums- und Zeitberechnungen	256
4.4.1	Die DateTime-Struktur	256
4.4.2	Wichtige Eigenschaften von DateTime-Variablen	258
4.4.3	Wichtige Methoden von DateTime-Variablen	258
4.4.4	Wichtige Mitglieder der DateTime-Struktur	259
4.4.5	Konvertieren von Datumstrings in DateTime-Werte	260
4.4.6	Die TimeSpan-Struktur	261
4.5	Mathematische Funktionen	263
4.5.1	Überblick	263
4.5.2	Zahlen runden	263
4.5.3	Winkel umrechnen	264
4.5.4	Potenz- und Wurzeloperationen	264
4.5.5	Logarithmus und Exponentialfunktionen	264
4.5.6	Zufallszahlen erzeugen	265
4.6	Zahlen- und Datumsformatierungen	266
4.6.1	Anwenden der ToString-Methode	266
4.6.2	Anwenden der Format-Methode	268
4.6.3	Stringinterpolation	269
4.7	Praxisbeispiele	270
4.7.1	Zeichenketten verarbeiten	270
4.7.2	Zeichenketten mit StringBuilder addieren	273
4.7.3	Reguläre Ausdrücke testen	277
4.7.4	Methodenaufrufe mit Array-Parametern	278
5	Weitere Sprachfeatures	283
5.1	Namespaces (Namensräume)	283
5.1.1	Ein kleiner Überblick	283
5.1.2	Einen eigenen Namespace einrichten	284
5.1.3	Die using-Anweisung	285
5.1.4	Namespace Alias	286
5.2	Operatorenüberladung	287
5.2.1	Syntaxregeln	287
5.2.2	Praktische Anwendung	287
5.3	Collections (Auflistungen)	288
5.3.1	Die Schnittstelle IEnumerable	289
5.3.2	ArrayList	291
5.3.3	Hashtable	293
5.3.4	Indexer	293

5.4	Generics	296
5.4.1	Klassische Vorgehensweise	296
5.4.2	Generics bieten Typsicherheit	298
5.4.3	Generische Methoden	299
5.4.4	Iteratoren	300
5.5	Generische Collections	301
5.5.1	List-Collection statt ArrayList	301
5.5.2	Vorteile generischer Collections	302
5.5.3	Constraints	302
5.6	Das Prinzip der Delegates	303
5.6.1	Delegates sind Methodenzeiger	303
5.6.2	Einen Delegate-Typ deklarieren	303
5.6.3	Ein Delegate-Objekt erzeugen	304
5.6.4	Delegates vereinfacht instanziiieren	306
5.6.5	Anonyme Methoden	306
5.6.6	Lambda-Ausdrücke	308
5.6.7	Lambda-Ausdrücke in der Task Parallel Library	310
5.7	Dynamische Programmierung	312
5.7.1	Wozu dynamische Programmierung?	312
5.7.2	Das Prinzip der dynamischen Programmierung	312
5.7.3	Optionale Parameter sind hilfreich	315
5.7.4	Kovarianz und Kontravarianz	316
5.8	Weitere Datentypen	317
5.8.1	BigInteger	317
5.8.2	Complex	319
5.8.3	Tuple<>	320
5.8.4	SortedSet<>	321
5.9	Praxisbeispiele	322
5.9.1	ArrayList versus generische List	322
5.9.2	Generische IEnumerable-Interfaces implementieren	325
5.9.3	Delegates, anonyme Methoden, Lambda Expressions	329
5.9.4	Dynamischer Zugriff auf COM Interop	333
6	Einführung in LINQ	337
6.1	LINQ-Grundlagen	337
6.1.1	Die LINQ-Architektur	337
6.1.2	Anonyme Typen	339
6.1.3	Erweiterungsmethoden	340
6.2	Abfragen mit LINQ to Objects	341
6.2.1	Grundlegendes zur LINQ-Syntax	342
6.2.2	Zwei alternative Schreibweisen von LINQ-Abfragen	343
6.2.3	Übersicht der wichtigsten Abfrageoperatoren	344
6.3	LINQ-Abfragen im Detail	345

6.3.1	Die Projektionsoperatoren Select und SelectMany	346
6.3.2	Der Restriktionsoperator Where	348
6.3.3	Die Sortierungsoperatoren OrderBy und ThenBy	348
6.3.4	Der Gruppierungsoperator GroupBy	350
6.3.5	Verknüpfen mit Join	352
6.3.6	Aggregat-Operatoren	353
6.3.7	Verzögertes Ausführen von LINQ-Abfragen	355
6.3.8	Konvertierungsmethoden	356
6.3.9	Abfragen mit PLINQ	357
6.4	Praxisbeispiele	360
6.4.1	Die Syntax von LINQ-Abfragen verstehen	360
6.4.2	Aggregat-Abfragen mit LINQ	363
6.4.3	LINQ im Schnelldurchgang erlernen	365
6.4.4	Strings mit LINQ abfragen und filtern	368
6.4.5	Duplikate aus einer Liste oder einem Array entfernen	369
6.4.6	Arrays mit LINQ initialisieren	372
6.4.7	Arrays per LINQ mit Zufallszahlen füllen	374
6.4.8	Einen String mit Wiederholmuster erzeugen	376
6.4.9	Mit LINQ Zahlen und Strings sortieren	377
6.4.10	Mit LINQ Collections von Objekten sortieren	378
6.4.11	Ergebnisse von LINQ-Abfragen in ein Array kopieren	381
7	C#-Sprachneuerungen im Überblick	383
7.1	C# 4.0 – Visual Studio 2010	383
7.1.1	Datentyp dynamic	383
7.1.2	Benannte und optionale Parameter	384
7.1.3	Covarianz und Contravarianz	386
7.2	C# 5.0 – Visual Studio 2012	386
7.2.1	Async und Await	386
7.2.2	CallerInfo	386
7.3	Visual Studio 2013	387
7.4	C# 6.0 – Visual Studio 2015	387
7.4.1	String Interpolation	387
7.4.2	Schreibgeschützte AutoProperties	387
7.4.3	Initialisierer für AutoProperties	388
7.4.4	Expression Body Funktionsmember	388
7.4.5	using static	388
7.4.6	Bedingter Nulloperator	389
7.4.7	Ausnahmenfilter	389
7.4.8	nameof-Ausdrücke	390
7.4.9	await in catch und finally	390
7.4.10	Indexinitialisierer	390
7.5	C# 7.0 – Visual Studio 2017	391
7.5.1	out-Variablen	391

7.5.2	Tupel	391
7.5.3	Mustervergleich	392
7.5.4	Discards	394
7.5.5	Lokale ref-Variablen und Rückgabetypen	394
7.5.6	Lokale Funktionen	394
7.5.7	Mehr Expression-Bodied Member	394
7.5.8	throw-Ausdrücke	395
7.5.9	Verbesserung der numerischen literalen Syntax	395

TEIL II: Technologien 397

8 Zugriff auf das Dateisystem 399

8.1	Grundlagen	399
8.1.1	Klassen für den Zugriff auf das Dateisystem	400
8.1.2	Statische versus Instanzen-Klasse	400
8.2	Übersichten	401
8.2.1	Methoden der Directory-Klasse	402
8.2.2	Methoden eines DirectoryInfo-Objekts	402
8.2.3	Eigenschaften eines DirectoryInfo-Objekts	402
8.2.4	Methoden der File-Klasse	403
8.2.5	Methoden eines FileInfo-Objekts	404
8.2.6	Eigenschaften eines FileInfo-Objekts	404
8.3	Operationen auf Verzeichnisebene	405
8.3.1	Existenz eines Verzeichnisses/einer Datei feststellen	405
8.3.2	Verzeichnisse erzeugen und löschen	405
8.3.3	Verzeichnisse verschieben und umbenennen	406
8.3.4	Aktuelles Verzeichnis bestimmen	406
8.3.5	Unterverzeichnisse ermitteln	407
8.3.6	Alle Laufwerke ermitteln	407
8.3.7	Dateien kopieren und verschieben	408
8.3.8	Dateien umbenennen	409
8.3.9	Dateiattribute feststellen	409
8.3.10	Verzeichnis einer Datei ermitteln	411
8.3.11	Alle im Verzeichnis enthaltenen Dateien ermitteln	411
8.3.12	Dateien und Unterverzeichnisse ermitteln	412
8.4	Zugriffsberechtigungen	413
8.4.1	ACL und ACE	413
8.4.2	SetAccessControl-Methode	413
8.4.3	Zugriffsrechte anzeigen	414
8.5	Weitere wichtige Klassen	415
8.5.1	Die Path-Klasse	415
8.5.2	Die Klasse FileSystemWatcher	416
8.5.3	Die Klasse ZipArchive	417
8.6	Datei- und Verzeichnisdialoge	419

8.6.1	OpenFileDialog und SaveFileDialog	419
8.6.2	FolderBrowserDialog	421
8.7	Praxisbeispiele	422
8.7.1	Infos über Verzeichnisse und Dateien gewinnen	422
8.7.2	Eine Verzeichnisstruktur in die TreeView einlesen	425
8.7.3	Mit LINQ und RegEx Verzeichnisbäume durchsuchen	428
9	Dateien lesen und schreiben	433
9.1	Grundprinzip der Datenpersistenz	433
9.1.1	Dateien und Streams	433
9.1.2	Die wichtigsten Klassen	434
9.1.3	Erzeugen eines Streams	435
9.2	Dateiparameter	435
9.2.1	FileAccess	435
9.2.2	FileMode	436
9.2.3	FileShare	436
9.3	Textdateien	437
9.3.1	Eine Textdatei beschreiben bzw. neu anlegen	437
9.3.2	Eine Textdatei lesen	438
9.4	Binärdateien	440
9.4.1	Lese-/Schreibzugriff	440
9.4.2	Die Methoden ReadAllBytes und WriteAllBytes	441
9.4.3	Erzeugen von BinaryReader/BinaryWriter	441
9.5	Sequenzielle Dateien	442
9.5.1	Lesen und Schreiben von strukturierten Daten	442
9.5.2	Serialisieren von Objekten	443
9.6	Dateien verschlüsseln und komprimieren	444
9.6.1	Das Methodenpärchen Encrypt/Decrypt	444
9.6.2	Verschlüsseln unter Windows Vista/7/8/10	445
9.6.3	Verschlüsseln mit der CryptoStream-Klasse	446
9.6.4	Dateien komprimieren	447
9.7	Memory Mapped Files	448
9.7.1	Grundprinzip	448
9.7.2	Erzeugen eines MMF	449
9.7.3	Erstellen eines Map View	449
9.8	Praxisbeispiele	450
9.8.1	Auf eine Textdatei zugreifen	450
9.8.2	Einen Objektbaum persistent speichern	454
9.8.3	Ein Memory Mapped File (MMF) verwenden	461
9.8.4	Hex-Dezimal-Bytes-Konverter	463
9.8.5	Eine Datei verschlüsseln	467
9.8.6	Eine Datei komprimieren	470
9.8.7	PDFs erstellen/exportieren	472

9.8.8	Eine CSV-Datei erstellen	475
9.8.9	Eine CSV-Datei mit LINQ lesen und auswerten	478
9.8.10	Einen korrekten Dateinamen erzeugen	480
10	Asynchrone Programmierung	481
10.1	Übersicht	481
10.1.1	Multitasking versus Multithreading	482
10.1.2	Deadlocks	483
10.1.3	Racing	484
10.2	Programmieren mit Threads	485
10.2.1	Einführungsbeispiel	486
10.2.2	Wichtige Thread-Methoden	487
10.2.3	Wichtige Thread-Eigenschaften	489
10.2.4	Einsatz der ThreadPool-Klasse	490
10.3	Sperrmechanismen	492
10.3.1	Threading ohne lock	492
10.3.2	Threading mit lock	494
10.3.3	Die Monitor-Klasse	496
10.3.4	Mutex	500
10.3.5	Methoden für die parallele Ausführung sperren	501
10.3.6	Semaphore	502
10.4	Interaktion mit der Programmoberfläche	503
10.4.1	Die Werkzeuge	504
10.4.2	Einzelne Steuerelemente mit Invoke aktualisieren	504
10.4.3	Mehrere Steuerelemente aktualisieren	506
10.4.4	Ist ein Invoke-Aufruf nötig?	506
10.4.5	Und was ist mit WPF?	507
10.5	Timer-Threads	509
10.6	Die BackgroundWorker-Komponente	510
10.7	Asynchrone Programmierentwurfsmuster	512
10.7.1	Kurzübersicht	513
10.7.2	Polling	514
10.7.3	Callback verwenden	516
10.7.4	Callback mit Parameterübergabe verwenden	516
10.7.5	Callback mit Zugriff auf die Programmoberfläche	518
10.8	Asynchroner Aufruf beliebiger Methoden	519
10.8.1	Die Beispielklasse	519
10.8.2	Asynchroner Aufruf ohne Callback	520
10.8.3	Asynchroner Aufruf mit Callback und Anzeigefunktion	521
10.8.4	Aufruf mit Rückgabewerten (per Eigenschaft)	522
10.8.5	Aufruf mit Rückgabewerten (per EndInvoke)	523
10.9	Es geht auch einfacher – async und await	524
10.9.1	Der Weg von synchron zu asynchron	524

10.9.2	Achtung: Fehlerquellen!	526
10.9.3	Eigene asynchrone Methoden entwickeln	528
10.10	Praxisbeispiele	531
10.10.1	Spieltrieb & Multithreading erleben	531
10.10.2	Prozess- und Thread-Informationen gewinnen	544
10.10.3	Ein externes Programm starten	549
11	Die Task Parallel Library	553
11.1	Überblick	553
11.1.1	Parallel-Programmierung	553
11.1.2	Möglichkeiten der TPL	556
11.1.3	Der CLR-Threadpool	556
11.2	Parallele Verarbeitung mit Parallel.Invoke	557
11.2.1	Aufrufvarianten	558
11.2.2	Einschränkungen	559
11.3	Verwendung von Parallel.For	559
11.3.1	Abbrechen der Verarbeitung	561
11.3.2	Auswerten des Verarbeitungsstatus	562
11.3.3	Und was ist mit anderen Iterator-Schrittweiten?	563
11.4	Collections mit Parallel.ForEach verarbeiten	564
11.5	Die Task-Klasse	565
11.5.1	Einen Task erzeugen	565
11.5.2	Den Task starten	566
11.5.3	Datenübergabe an den Task	568
11.5.4	Wie warte ich auf das Ende des Task?	569
11.5.5	Tasks mit Rückgabewerten	571
11.5.6	Die Verarbeitung abbrechen	574
11.5.7	Fehlerbehandlung	578
11.5.8	Weitere Eigenschaften	579
11.6	Zugriff auf das User-Interface	580
11.6.1	Task-Ende und Zugriff auf die Oberfläche	580
11.6.2	Zugriff auf das UI aus dem Task heraus	582
11.7	Weitere Datenstrukturen im Überblick	584
11.7.1	Threadsichere Collections	584
11.7.2	Primitive für die Threadsynchonisierung	585
11.8	Parallel LINQ (PLINQ)	585
11.9	Praxisbeispiel: Spieltrieb – Version 2	585
11.9.1	Aufgabenstellung	586
11.9.2	Global-Klasse	586
11.9.3	Controller-Klasse	587
11.9.4	LKW-Klasse	589
11.9.5	Schiff-Klasse	590
11.9.6	Oberfläche	593

12	Fehlersuche und Behandlung	595
12.1	Der Debugger	595
12.1.1	Allgemeine Beschreibung	595
12.1.2	Die wichtigsten Fenster	596
12.1.3	Debugging-Optionen	599
12.1.4	Praktisches Debugging am Beispiel	602
12.2	Arbeiten mit Debug und Trace	606
12.2.1	Wichtige Methoden von Debug und Trace	606
12.2.2	Besonderheiten der Trace-Klasse	610
12.2.3	TraceListener-Objekte	611
12.3	Caller Information	613
12.3.1	Attribute	613
12.3.2	Anwendung	614
12.4	Fehlerbehandlung	615
12.4.1	Anweisungen zur Fehlerbehandlung	615
12.4.2	try-catch	615
12.4.3	try-finally	620
12.4.4	Das Standardverhalten bei Ausnahmen festlegen	623
12.4.5	Die Exception-Klasse	624
12.4.6	Fehler/Ausnahmen auslösen	625
12.4.7	Eigene Fehlerklassen	625
12.4.8	Exceptionhandling zur Entwurfszeit	627
12.4.9	Code Contracts	628
13	XML in Theorie und Praxis	629
13.1	XML – etwas Theorie	629
13.1.1	Übersicht	629
13.1.2	Der XML-Grundaufbau	632
13.1.3	Wohlgeformte Dokumente	633
13.1.4	Processing Instructions (PI)	635
13.1.5	Elemente und Attribute	636
13.1.6	Verwendbare Zeichensätze	637
13.2	XSD-Schemas	640
13.2.1	XSD-Schemas und ADO.NET	640
13.2.2	XML-Schemas in Visual Studio analysieren	642
13.2.3	XML-Datei mit XSD-Schema erzeugen	646
13.2.4	XSD-Schema aus einer XML-Datei erzeugen	647
13.3	Verwendung des DOM unter .NET	647
13.3.1	Übersicht	647
13.3.2	DOM-Integration in C#	649
13.3.3	Laden von Dokumenten	649
13.3.4	Erzeugen von XML-Dokumenten	650
13.3.5	Auslesen von XML-Dateien	652

13.3.6	Direktzugriff auf einzelne Elemente	653
13.3.7	Einfügen von Informationen	654
13.3.8	Suchen in den Baumzweigen	657
13.4	XML-Verarbeitung mit LINQ to XML	660
13.4.1	Die LINQ to XML-API	660
13.4.2	Neue XML-Dokumente erzeugen	662
13.4.3	Laden und Sichern von XML-Dokumenten	664
13.4.4	Navigieren in XML-Daten	665
13.4.5	Auswählen und Filtern	667
13.4.6	Manipulieren der XML-Daten	668
13.4.7	XML-Dokumente transformieren	669
13.5	Weitere Möglichkeiten der XML-Verarbeitung	672
13.5.1	XML-Daten aus Objektstrukturen erzeugen	672
13.5.2	Schnelles Suchen in XML-Daten mit XPathNavigator	676
13.5.3	Schnelles Auslesen von XML-Daten mit XmlReader	678
13.5.4	Erzeugen von XML-Daten mit XmlWriter	680
13.5.5	XML transformieren mit XSLT	682
13.6	JSON – JavaScriptObjectNotation	684
13.6.1	Grundlagen	684
13.6.2	De-/Serialisierung mit JSON	684
13.7	Praxisbeispiele	687
13.7.1	Mit dem DOM in XML-Dokumenten navigieren	687
13.7.2	XML-Daten in eine TreeView einlesen	691
13.7.3	In Dokumenten mit dem XPathNavigator navigieren	695
14	Einführung in ADO.NET und Entity Framework	701
14.1	Eine kleine Übersicht	701
14.1.1	Die ADO.NET-Klassenhierarchie	701
14.1.2	Die Klassen der Datenprovider	703
14.1.3	Das Zusammenspiel der ADO.NET-Klassen	705
14.2	Das Connection-Objekt	706
14.2.1	Allgemeiner Aufbau	706
14.2.2	SqlConnection	706
14.2.3	Schließen einer Verbindung	707
14.2.4	Eigenschaften des Connection-Objekts	708
14.2.5	Methoden des Connection-Objekts	710
14.2.6	Der SqlConnectionStringBuilder	711
14.3	Das Command-Objekt	711
14.3.1	Erzeugen und Anwenden eines Command-Objekts	712
14.3.2	Erzeugen mittels CreateCommand-Methode	713
14.3.3	Eigenschaften des Command-Objekts	713
14.3.4	Methoden des Command-Objekts	715
14.3.5	Freigabe von Connection- und Command-Objekten	717

14.4	Parameter-Objekte	718
14.4.1	Erzeugen und Anwenden eines Parameter-Objekts	718
14.4.2	Eigenschaften des Parameter-Objekts	719
14.5	Das CommandBuilder-Objekt	720
14.5.1	Erzeugen	720
14.5.2	Anwenden	720
14.6	Das DataReader-Objekt	721
14.6.1	DataReader erzeugen	721
14.6.2	Daten lesen	722
14.6.3	Eigenschaften des DataReaders	723
14.6.4	Methoden des DataReaders	723
14.7	Das DataAdapter-Objekt	724
14.7.1	DataAdapter erzeugen	724
14.7.2	Command-Eigenschaften	725
14.7.3	Fill-Methode	726
14.7.4	Update-Methode	727
14.8	Entity Framework	728
14.8.1	Überblick	728
14.8.2	DatabaseFirst	730
14.8.3	CodeFirst	738
14.9	Praxisbeispiele	743
14.9.1	Wichtige ADO.NET-Objekte im Einsatz	743
14.9.2	Eine Aktionsabfrage ausführen	745
14.9.3	Eine StoredProcedure aufrufen	747
14.9.4	Die Datenbank aktualisieren	750
15	Das DataSet	755
15.1	Grundlegende Features des DataSets	755
15.1.1	Die Objekthierarchie	756
15.1.2	Die wichtigsten Klassen	756
15.1.3	Erzeugen eines DataSets	757
15.2	Das DataTable-Objekt	759
15.2.1	DataTable erzeugen	759
15.2.2	Spalten hinzufügen	759
15.2.3	Zeilen zur DataTable hinzufügen	760
15.2.4	Auf den Inhalt einer DataTable zugreifen	761
15.3	Die DataView	763
15.3.1	Erzeugen einer DataView	764
15.3.2	Sortieren und Filtern von Datensätzen	764
15.3.3	Suchen von Datensätzen	765
15.4	Typisierte DataSets	765
15.4.1	Ein typisiertes DataSet erzeugen	766
15.4.2	Das Konzept der Datenquellen	767
15.4.3	Typisierte DataSets und TableAdapter	768

15.5	Die Qual der Wahl	769
15.5.1	DataReader – der schnelle Lesezugriff	770
15.5.2	DataSet – die Datenbank im Hauptspeicher	770
15.5.3	Objektrelationales Mapping – die Zukunft?	771
15.6	Praxisbeispiele	772
15.6.1	In der DataView sortieren und filtern	772
15.6.2	Suche nach Datensätzen	774
15.6.3	Ein DataSet in einen XML-String serialisieren	776
15.6.4	Untypisiertes DataSet in ein typisiertes konvertieren	780
16	Verteilen von Anwendungen	787
16.1	ClickOnce-Deployment	788
16.1.1	Übersicht/Einschränkungen	788
16.1.2	Die Vorgehensweise	789
16.1.3	Ort der Veröffentlichung	789
16.1.4	Anwendungsdateien	790
16.1.5	Erforderliche Komponenten	791
16.1.6	Aktualisierungen	792
16.1.7	Veröffentlichungsoptionen	793
16.1.8	Veröffentlichen	794
16.1.9	Verzeichnisstruktur	794
16.1.10	Der Webpublishing-Assistent	796
16.1.11	Neue Versionen erstellen	797
16.2	Setup-Projekte	797
16.2.1	Installation	797
16.2.2	Ein neues Setup-Projekt	799
16.2.3	Dateisystem-Editor	804
16.2.4	Registrierungs-Editor	805
16.2.5	Dateityp-Editor	806
16.2.6	Benutzeroberflächen-Editor	806
16.2.7	Editor für benutzerdefinierte Aktionen	807
16.2.8	Editor für Startbedingungen	808
17	Weitere Techniken	809
17.1	Zugriff auf die Zwischenablage	809
17.1.1	Das Clipboard-Objekt	809
17.1.2	Zwischenablage-Funktionen für Textboxen	811
17.2	Arbeiten mit der Registry	812
17.2.1	Allgemeines	812
17.2.2	Registry-Unterstützung in .NET	814
17.3	.NET-Reflection	815
17.3.1	Übersicht	815
17.3.2	Assembly laden	816
17.3.3	Mittels GetType und Type Informationen sammeln	816
17.3.4	Dynamisches Laden von Assemblies	819

17.4	Praxisbeispiele	822
17.4.1	Zugriff auf die Registry	822
17.4.2	Dateiverknüpfungen erzeugen	824
17.4.3	Betrachter für Manifestressourcen	826
17.4.4	Die Zwischenablage überwachen und anzeigen	829
17.4.5	Die WIA-Library kennenlernen	832
17.4.6	Auf eine Webcam zugreifen	843
17.4.7	Auf den Scanner zugreifen	845
17.4.8	OpenOffice.org Writer per OLE steuern	851
17.4.9	Nutzer und Gruppen des aktuellen Systems ermitteln	858
17.4.10	Testen, ob Nutzer in einer Gruppe enthalten ist	860
17.4.11	Testen, ob der Nutzer ein Administrator ist	862
17.4.12	Die IP-Adressen des Computers bestimmen	863
17.4.13	Die IP-Adresse über den Hostnamen bestimmen	864
17.4.14	Diverse Systeminformationen ermitteln	865
17.4.15	Alles über den Bildschirm erfahren	873
17.4.16	Sound per MCI aufnehmen	875
17.4.17	Mikrofonpegel anzeigen	878
17.4.18	Pegeldiagramm aufzeichnen	880
17.4.19	Sound-und Videodateien per MCI abspielen	883
18	Konsolenanwendungen	893
18.1	Grundaufbau/Konzepte	893
18.1.1	Unser Hauptprogramm – Program.cs	894
18.1.2	Rückgabe eines Fehlerstatus	895
18.1.3	Parameterübergabe	896
18.1.4	Zugriff auf die Umgebungsvariablen	898
18.2	Die Kommandozentrale: System.Console	899
18.2.1	Eigenschaften	899
18.2.2	Methoden/Ereignisse	900
18.2.3	Textausgaben	900
18.2.4	Farbangaben	901
18.2.5	Tastaturabfragen	903
18.2.6	Arbeiten mit Streamdaten	904
18.3	Praxisbeispiel	906
18.3.1	Farbige Konsolenanwendung	906
18.3.2	Weitere Hinweise und Beispiele	908
Anhang A: Glossar		909
Anhang B: Wichtige Dateiextensions		913
Index		915