

Auf einen Blick

1	Allgemeine Einführung in .NET	33
2	Grundlagen der Sprache C#	57
3	Das Klassendesign	133
4	Vererbung, Polymorphie und Interfaces	205
5	Delegates und Ereignisse	259
6	Strukturen und Enumerationen	289
7	Fehlerbehandlung und Debugging	299
8	Auflistungsklassen (Collections)	333
9	Generics – Generische Datentypen	359
10	Weitere C#-Sprachfeatures	385
11	LINQ	435
12	Arbeiten mit Dateien und Streams	467
13	Binäre Serialisierung	511
14	XML	519
15	Multithreading und die Task Parallel Library (TPL)	613
16	Einige wichtige .NET-Klassen	665
17	Projektmanagement und Visual Studio 2012	705
18	Einführung in die WPF und XAML	759
19	WPF-Layout-Container	789
20	Fenster in der WPF	809
21	WPF-Steuerelemente	833
22	Elementbindungen	921
23	Konzepte von WPF	935
24	Datenbindung	975
25	Weitere Möglichkeiten der Datenbindung	1005
26	Dependency Properties	1041
27	Ereignisse in der WPF	1057
28	WPF-Commands	1077
29	Benutzerdefinierte Controls	1097
30	2D-Grafik	1107
31	ADO.NET – Verbindungsorientierte Objekte	1127
32	ADO.NET – Das Command-Objekt	1149
33	ADO.NET – Der SqlDataAdapter	1177
34	ADO.NET – Daten im lokalen Speicher	1191
35	ADO.NET – Aktualisieren der Datenbank	1231
36	Stark typisierte DataSets	1251
37	Einführung in das ADO.NET Entity Framework	1277
38	Datenabfragen des Entity Data Models (EDM)	1301
39	Entitätsaktualisierung und Zustandsverwaltung	1335
40	Konflikte behandeln	1363
41	Plain Old CLR Objects (POCOs)	1371

Inhalt

Vorwort zur 6. Auflage	31
1 Allgemeine Einführung in .NET	33
1.1 Warum .NET?	33
1.1.1 Ein paar Worte zu diesem Buch	35
1.1.2 Die Beispielprogramme	37
1.2 .NET unter die Lupe genommen	38
1.2.1 Das Entwicklerdilemma	38
1.2.2 .NET – Ein paar allgemeine Eigenschaften	39
1.2.3 Das Sprachenkonzept	40
1.2.4 Die »Common Language Specification« (CLS)	42
1.2.5 Das »Common Type System« (CTS)	43
1.2.6 Das .NET Framework	44
1.2.7 Die »Common Language Runtime« (CLR)	45
1.2.8 Die .NET-Klassenbibliothek	45
1.2.9 Das Konzept der Namespaces	47
1.3 Assemblies	48
1.3.1 Die Metadaten	49
1.3.2 Das Manifest	50
1.4 Die Entwicklungsumgebung	50
1.4.1 Editionen von Visual Studio 2012	50
1.4.2 Hard- und Softwareanforderungen	51
1.4.3 Die Installation	51
1.4.4 Die Entwicklungsumgebung von Visual Studio 2012	52
2 Grundlagen der Sprache C#	57
2.1 Konsolenanwendungen	57
2.1.1 Allgemeine Anmerkungen	57
2.1.2 Ein erstes Konsolenprogramm	57
2.2 Grundlagen der C#-Syntax	60
2.2.1 Kennzeichnen, dass eine Anweisung abgeschlossen ist	60
2.2.2 Anweisungs- und Gliederungsblöcke	61
2.2.3 Kommentare	62
2.2.4 Die Groß- und Kleinschreibung	63
2.2.5 Die Struktur einer Konsolenanwendung	64

2.3 Variablen und Datentypen	66
2.3.1 Variablen Deklaration	66
2.3.2 Der Variablenbezeichner	67
2.3.3 Der Zugriff auf eine Variable	68
2.3.4 Ein- und Ausgabemethoden der Klasse »Console«	68
2.3.5 Die einfachen Datentypen	74
2.3.6 Typkonvertierung	80
2.4 Operatoren	87
2.4.1 Arithmetische Operatoren	88
2.4.2 Vergleichsoperatoren	91
2.4.3 Logische Operatoren	91
2.4.4 Bitweise Operatoren	95
2.4.5 Zuweisungsoperatoren	98
2.4.6 Stringverkettung	98
2.4.7 Sonstige Operatoren	99
2.4.8 Operator-Vorrangregeln	99
2.5 Datenfelder (Arrays)	100
2.5.1 Die Deklaration und Initialisierung eines Arrays	100
2.5.2 Der Zugriff auf die Array-Elemente	102
2.5.3 Mehrdimensionale Arrays	103
2.5.4 Festlegen der Array-Größe zur Laufzeit	104
2.5.5 Bestimmung der Array-Obergrenze	105
2.5.6 Die Gesamtanzahl der Array-Elemente	106
2.5.7 Verzweigte Arrays	106
2.6 Kontrollstrukturen	108
2.6.1 Die »if«-Anweisung	108
2.6.2 Das »switch«-Statement	113
2.7 Programmschleifen	117
2.7.1 Die »for«-Schleife	118
2.7.2 Die »foreach«-Schleife	128
2.7.3 Die »do«- und die »while«-Schleife	129
3 Das Klassendesign	133
3.1 Einführung in die Objektorientierung	133
3.2 Die Klassendefinition	136
3.2.1 Klassen im Visual Studio anlegen	136
3.2.2 Das Projekt »GeometricObjectsSolution«	136
3.2.3 Die Deklaration von Objektvariablen	139
3.2.4 Zugriffsmodifizierer einer Klasse	140

3.2.5	Splitten einer Klassendefinition mit »partial«	140
3.2.6	Arbeiten mit Objektreferenzen	141
3.3	Referenz- und Wertetypen	143
3.3.1	Werte- und Referenztypen nutzen	144
3.4	Die Eigenschaften eines Objekts	145
3.4.1	Öffentliche Felder	145
3.4.2	Datenkapselung mit Eigenschaftsmethoden sicherstellen	147
3.4.3	Die Ergänzung der Klasse »Circle«	149
3.4.4	Lese- und schreibgeschützte Eigenschaften	149
3.4.5	Sichtbarkeit der Accessoren »get« und »set«	150
3.4.6	Unterstützung von Visual Studio 2012	151
3.4.7	Automatisch implementierte Eigenschaften	152
3.5	Methoden eines Objekts	153
3.5.1	Methoden mit Rückgabewert	153
3.5.2	Methoden ohne Rückgabewert	157
3.5.3	Methoden mit Parameterliste	157
3.5.4	Methodenüberladung	159
3.5.5	Variablen innerhalb einer Methode (lokale Variablen)	162
3.5.6	Referenz- und Wertparameter	163
3.5.7	Besondere Aspekte einer Parameterliste	168
3.5.8	Zugriff auf private Daten	173
3.5.9	Die Trennung von Daten und Code	174
3.5.10	Namenskonflikte mit »this« lösen	175
3.5.11	Methode oder Eigenschaft?	176
3.5.12	Umbenennen von Methoden und Eigenschaften	177
3.6	Konstruktoren	178
3.6.1	Konstruktoren bereitstellen	179
3.6.2	Die Konstruktoraufrufe	180
3.6.3	Definition von Konstruktoren	180
3.6.4	»public«- und »internal«-Konstruktoren	181
3.6.5	»private«-Konstruktoren	181
3.6.6	Konstruktoraufrufe umleiten	182
3.6.7	Vereinfachte Objektinitialisierung	183
3.7	Der Destruktor	184
3.8	Konstanten in einer Klasse	185
3.8.1	Konstanten mit dem Schlüsselwort »const«	185
3.8.2	Schreibgeschützte Felder mit »readonly«	186
3.9	Statische Klassenkomponenten	186
3.9.1	Statische Eigenschaften	186
3.9.2	Statische Methoden	189

3.9.3	Statische Klasseninitialisierer	190
3.9.4	Statische Klassen	191
3.9.5	Statische Klasse oder Singleton-Pattern?	192
3.10	Namensräume (Namespaces)	193
3.10.1	Zugriff auf Namespaces	194
3.10.2	Die »using«-Direktive	196
3.10.3	Globaler Namespace	197
3.10.4	Vermeiden von Mehrdeutigkeiten	197
3.10.5	Namespaces festlegen	198
3.10.6	Der »::«-Operator	200
3.10.7	Unterstützung von Visual Studio 2012 bei den Namespaces	201
3.11	Stand der Klasse »Circle«	203
4	Vererbung, Polymorphie und Interfaces	205
4.1	Die Vererbung	205
4.1.1	Basisklassen und abgeleitete Klassen	205
4.1.2	Die Ableitung einer Klasse	206
4.1.3	Klassen, die nicht abgeleitet werden können	208
4.1.4	Konstruktoren in abgeleiteten Klassen	208
4.1.5	Der Zugriffsmodifizierer »protected«	209
4.1.6	Die Konstruktorverkettung in der Vererbung	210
4.2	Der Problemfall geerbter Methoden	214
4.2.1	Geerbte Methoden mit »new« verdecken	216
4.2.2	Abstrakte Methoden	217
4.2.3	Virtuelle Methoden	219
4.3	Typumwandlung und Typuntersuchung von Objektvariablen	220
4.3.1	Die implizite Typumwandlung von Objektreferenzen	220
4.3.2	Die explizite Typumwandlung von Objektreferenzen	222
4.3.3	Typuntersuchung mit dem »is«-Operator	223
4.3.4	Typumwandlung mit dem »as«-Operator	224
4.4	Polymorphie	224
4.4.1	Die »klassische« Methodenimplementierung	225
4.4.2	Abstrakte Methoden	226
4.4.3	Virtuelle Methoden	227
4.5	Weitere Gesichtspunkte der Vererbung	230
4.5.1	Versiegelte Methoden	230
4.5.2	Überladen einer Basisklassenmethode	231
4.5.3	Statische Member und Vererbung	232
4.5.4	Geerbte Methoden ausblenden?	232

4.6	Das Projekt »GeometricObjectsSolution« ergänzen	233
4.6.1	Die Klasse »GeometricObject«	233
4.7	Eingebettete Klassen (Nested Classes)	237
4.8	Interfaces (Schnittstellen)	237
4.8.1	Einführung in die Schnittstellen	237
4.8.2	Die Schnittstellendefinition	238
4.8.3	Die Schnittstellenimplementierung	239
4.8.4	Die Interpretation der Schnittstellen	244
4.8.5	Änderungen am Projekt »GeometricObjects«	249
4.9	Das Zerstören von Objekten – der »Garbage Collector«	251
4.9.1	Die Arbeitsweise des Garbage Collectors	251
4.9.2	Expliziter Aufruf des Garbage Collectors	252
4.9.3	Der Destruktor	253
4.9.4	Die »IDisposable«-Schnittstelle	254
4.9.5	Die Ergänzungen in den Klassen »Circle« und »Rectangle«	257

5 Delegates und Ereignisse

5.1	Delegates	259
5.1.1	Einführung in das Prinzip der Delegates	259
5.1.2	Verwendung von Delegates	263
5.1.3	Vereinfachter Delegatenauftrag	263
5.1.4	Multicast-Delegates	264
5.1.5	Anonyme Methoden	266
5.1.6	Kovarianz und Kontravarianz mit Delegaten	268
5.2	Ereignisse eines Objekts	270
5.2.1	Ereignisse bereitstellen	271
5.2.2	Die Reaktion auf ein ausgelöstes Ereignis	273
5.2.3	Allgemeine Betrachtungen der Ereignishandler-Registrierung	275
5.2.4	Wenn der Ereignisempfänger ein Ereignis nicht behandelt	276
5.2.5	Ereignisse mit Übergabeparameter	277
5.2.6	Ereignisse in der Vererbung	281
5.2.7	Hinter die Kulissen des Schlüsselworts »event« geblickt	282
5.2.8	Die Schnittstelle »INotifyPropertyChanged«	284
5.3	Änderungen im Projekt »GeometricObjects«	285
5.3.1	Überarbeitung des Events »InvalidMeasure«	285
5.3.2	Weitere Ereignisse	286

6	Strukturen und Enumerationen	289
6.1	Strukturen – eine Sonderform der Klassen	289
6.1.1	Die Definition einer Struktur	289
6.1.2	Initialisieren einer Strukturvariablen	290
6.1.3	Konstruktoren in Strukturen	291
6.1.4	Änderung im Projekt »GeometricObjects«	292
6.2	Enumerationen (Aufzählungen)	295
6.2.1	Wertzuweisung an enum-Mitglieder	296
6.2.2	Alle Mitglieder einer Aufzählung durchlaufen	297
6.3	Boxing und Unboxing	298
7	Fehlerbehandlung und Debugging	299
7.1	Laufzeitfehler behandeln	299
7.1.1	Laufzeitfehler erkennen	300
7.1.2	Die »try...catch«-Anweisung	302
7.1.3	Behandlung mehrerer Exceptions	304
7.1.4	Die Reihenfolge der »catch«-Zweige	306
7.1.5	Ausnahmen in einer Methodenaufrufkette	307
7.1.6	Ausnahmen werfen oder weiterleiten	307
7.1.7	Die »finally«-Anweisung	308
7.1.8	Die Klasse »Exception«	309
7.1.9	Benutzerdefinierte Ausnahmen	314
7.2	Debuggen mit Programmcode	319
7.2.1	Einführung	319
7.2.2	Die Klasse »Debug«	320
7.2.3	Die Klasse »Trace«	324
7.2.4	Bedingte Kompilierung	324
7.3	Fehlersuche mit Visual Studio 2012	327
7.3.1	Debuggen im Haltemodus	327
7.3.2	Das »Direktfenster«	330
7.3.3	Weitere Alternativen, um Variableninhalte zu prüfen	331
8	Auflistungsklassen (Collections)	333
8.1	Grundlagen	333
8.2	Collections im Namespace »System.Collections«	333
8.2.1	Die elementaren Schnittstellen der Auflistungsklassen	335

8.3	Die Klasse »ArrayList«	337
8.3.1	Einträge hinzufügen	337
8.3.2	Datenaustausch zwischen einem Array und einer »ArrayList«	340
8.3.3	Die Elemente einer »ArrayList« sortieren	341
8.3.4	Sortieren von Arrays mit »ArrayList.Adapter«	346
8.4	Die Klasse »Hashtable«	348
8.4.1	Methoden und Eigenschaften der Schnittstelle »IDictionary«	348
8.4.2	Beispielprogramm zur Klasse »Hashtable«	349
8.5	Die Klassen »Queue« und »Stack«	353
8.5.1	Die Klasse »Stack«	354
8.5.2	Die Klasse »Queue«	355
8.6	Eigene Auflistungen mit »yield« durchlaufen	356

9 Generics – Generische Datentypen

9.1	Problembeschreibung	359
9.2	Bereitstellen einer generischen Klasse	360
9.2.1	Mehrere generische Typparameter	362
9.2.2	Vorteile der Generics	363
9.3	Bedingungen (Constraints) festlegen	363
9.3.1	Constraints mit der »where«-Klausel	363
9.3.2	Typparameter auf Klassen oder Strukturen beschränken	365
9.3.3	Mehrere Constraints definieren	365
9.3.4	Der Konstruktor-Constraint »new()«	366
9.3.5	Das Schlüsselwort »default«	366
9.4	Generische Methoden	367
9.4.1	Methoden und Constraints	368
9.5	Generics und Vererbung	368
9.5.1	Virtuelle generische Methoden	369
9.6	Konvertierung von Generics	370
9.7	Generische Delegates	371
9.7.1	Generische Delegates und Constraints	372
9.7.2	Anpassung des Beispiels »GeometricObjects«	372
9.8	Nullable-Typen	373
9.8.1	Konvertierungen mit Nullable-Typen	374
9.9	Generische Collections	374
9.9.1	Die Interfaces der generischen Auflistungsklassen	375
9.9.2	Die generische Auflistungsklasse »List<T>«	375
9.9.3	Vergleiche mit Hilfe des Delegaten »Comparison<T>«	378

9.10 Kovarianz und Kontravarianz generischer Typen	379
9.10.1 Kovarianz mit Interfaces	379
9.10.2 Kontravarianz mit Interfaces	381
9.10.3 Zusammenfassung	382
9.10.4 Generische Delegaten mit varianten Typparametern	383
10 Weitere C#-Sprachfeatures	385
10.1 Implizit typisierte Variablen	385
10.2 Anonyme Typen	386
10.3 Lambda-Ausdrücke	387
10.3.1 Projektion und Prädikat	389
10.4 Erweiterungsmethoden	389
10.5 Partielle Methoden	393
10.5.1 Wo partielle Methoden eingesetzt werden	394
10.6 Operatorüberladung	396
10.6.1 Einführung	396
10.6.2 Die Syntax der Operatorüberladung	396
10.6.3 Die Operatorüberladungen im Projekt »GeometricObjectsSolution«	397
10.6.4 Die Operatoren »true« und »false« überladen	402
10.6.5 Benutzerdefinierte Konvertierungen	403
10.7 Indexer	407
10.7.1 Überladen von Indexern	409
10.7.2 Parameterbehaftete Eigenschaften	411
10.8 Attribute	414
10.8.1 Das »Flags«-Attribut	415
10.8.2 Benutzerdefinierte Attribute	418
10.8.3 Attribute auswerten	422
10.8.4 Festlegen der Assembly-Eigenschaften in »Assembly-Info.cs«	424
10.9 Dynamisches Binden	426
10.9.1 Eine kurze Analyse	427
10.9.2 Dynamische Objekte	427
10.10 Unsicherer (unsafe) Programmcode – Zeigertechnik in C#	429
10.10.1 Einführung	429
10.10.2 Das Schlüsselwort »unsafe«	429
10.10.3 Die Deklaration von Zeigern	430
10.10.4 Die »fixed«-Anweisung	431
10.10.5 Zeigerarithmetik	432
10.10.6 Der Operator »->«	433

11 LINQ	435
11.1 Was ist LINQ?	435
11.1.1 Verzögerte Ausführung	436
11.1.2 LINQ-Erweiterungsmethoden an einem Beispiel	437
11.2 LINQ to Objects	440
11.2.1 Musterdaten	440
11.2.2 Die allgemeine LINQ-Syntax	442
11.3 Die Abfrageoperatoren	444
11.3.1 Übersicht der Abfrageoperatoren	444
11.3.2 Die »from«-Klausel	445
11.3.3 Mit »where« filtern	446
11.3.4 Die Projektionsoperatoren	449
11.3.5 Die Sortieroperatoren	450
11.3.6 Gruppieren mit »GroupBy«	451
11.3.7 Verknüpfungen mit »Join«	453
11.3.8 Die Set-Operatoren-Familie	456
11.3.9 Die Familie der Aggregatoperatoren	457
11.3.10 Quantifizierungsoperatoren	460
11.3.11 Aufteilungsoperatoren	461
11.3.12 Die Elementoperatoren	463
11.3.13 Die Konvertierungsoperatoren	466
12 Arbeiten mit Dateien und Streams	467
12.1 Einführung	467
12.2 Namespaces der Ein- bzw. Ausgabe	468
12.2.1 Das Behandeln von Ausnahmen bei E/A-Operationen	469
12.3 Laufwerke, Verzeichnisse und Dateien	469
12.3.1 Die Klasse »File«	469
12.3.2 Die Klasse »FileInfo«	475
12.3.3 Die Klassen »Directory« und » DirectoryInfo«	478
12.3.4 Die Klasse »Path«	482
12.3.5 Die Klasse »DriveInfo«	484
12.4 Die »Stream«-Klassen	485
12.4.1 Die abstrakte Klasse »Stream«	486
12.4.2 Die von »Stream« abgeleiteten Klassen im Überblick	488
12.4.3 Die Klasse »FileStream«	489
12.5 Die Klassen »TextReader« und »TextWriter«	496

12.5.1	Die Klasse »StreamWriter«	496
12.5.2	Die Klasse »StreamReader«	500
12.6	Die Klassen »BinaryReader« und »BinaryWriter«	502
12.6.1	Komplexe binäre Dateien	504
13	Binäre Serialisierung	511
13.1	Einführung in die Serialisierung	511
13.1.1	Serialisierungsverfahren	512
13.2	Serialisierung mit »BinaryFormatter«	513
13.2.1	Die Deserialisierung	515
13.2.2	Serialisierung mehrerer Objekte	516
14	XML	519
14.1	Grundlagen	519
14.2	XML-Dokumente	519
14.2.1	Wohlgeformte und gültige XML-Dokumente	520
14.2.2	Die Regeln eines wohlgeformten XML-Codes	522
14.2.3	Kommentare	525
14.2.4	Verarbeitungsanweisungen	525
14.2.5	Reservierte Zeichen in XML	526
14.2.6	CDATA-Abschnitte	526
14.2.7	Namensräume (Namespaces)	527
14.3	Die Gültigkeit eines XML-Dokuments	534
14.3.1	XML Schema Definition (XSD)	535
14.3.2	Ein XML-Dokument mit einem XML-Schema verknüpfen	536
14.3.3	Die Struktur eines XML-Schemas	539
14.4	Die Klasse »XmlReader«	545
14.4.1	XML-Dokumente mit einem »XmlReader«-Objekt lesen	545
14.4.2	Validieren eines XML-Dokuments	551
14.5	Eigenschaften und Methoden der Klasse »XmlReader«	554
14.6	Die Klasse »XmlWriter«	557
14.6.1	Die Methoden der Klasse »XmlWriter«	561
14.7	Navigation durch XML (XPath)	562
14.7.1	Die Klasse »XPathNavigator«	562
14.7.2	XPath-Ausdrücke	566
14.7.3	Der Kontextknoten	567
14.7.4	Beispiele mit XPath-Ausdrücken	569

14.7.5	Knotenmengen mit der »Select«-Methode	571
14.7.6	Auswerten von XPath-Ausdrücken	575
14.8	Das Document Object Model (DOM)	579
14.8.1	Allgemeines	579
14.8.2	Arbeiten mit » XmlDocument «	581
14.8.3	» XmlDocument « und » XPathNavigator «	582
14.8.4	Die Klasse » XmlNode « (Operationen mit Knoten)	582
14.8.5	Manipulieren einer XML-Struktur	590
14.8.6	Ändern eines Knotens	592
14.8.7	Löschen in einem XML-Dokument	594
14.9	Serialisierung mit » XmlSerializer «	596
14.9.1	XML-Serialisierung mit Attributen steuern	598
14.10	LINQ to XML	601
14.10.1	Allgemeines	601
14.10.2	Die Klassenhierarchie von LINQ to XML	601
14.10.3	Die Klasse » XElement «	602
14.10.4	Die Klasse » XDocument «	605
14.10.5	Navigation im XML-Dokument	605
14.10.6	Änderungen am XML-Dokument vornehmen	611
15	Multithreading und die Task Parallel Library (TPL)	613
15.1	Überblick	613
15.2	Multithreading mit der Klasse » Thread «	614
15.2.1	Einführung in das Multithreading	614
15.2.2	Threadzustände und Prioritäten	614
15.2.3	Zusammenspiel mehrerer Threads	616
15.2.4	Die Entwicklung einer einfachen Multithreading-Anwendung	616
15.2.5	Die Klasse » Thread «	619
15.2.6	Threadpools nutzen	627
15.2.7	Die Synchronisation von Threads	629
15.2.8	Der » Monitor « zur Synchronisation	631
15.2.9	Das Attribut » MethodImpl «	637
15.2.10	Das Synchronisationsobjekt » Mutex «	638
15.2.11	Grundlagen asynchroner Methodenaufrufe	639
15.2.12	Asynchroner Methodenaufruf	640
15.2.13	Asynchroner Aufruf mit Rückgabewerten	644
15.2.14	Eine Klasse mit asynchronen Methodenaufrufen	647
15.3	Die TPL (Task Parallel Library)	650
15.3.1	Allgemeines zur Parallelisierung mit der TPL	651

15.3.2	Die Klasse »Parallel«	651
15.3.3	Die Klasse »Task«	657
15.4	Asynchrone Programmierung mit »async« und »await«	661
16	Einige wichtige .NET-Klassen	665
16.1	Die Klasse »Object«	665
16.1.1	Referenzvergleiche mit »Equals« und »ReferenceEquals«	666
16.1.2	»ToString« und »GetType«	666
16.1.3	Die Methode »MemberwiseClone« und das Problem des Klonens	667
16.2	Die Klasse »String«	670
16.2.1	Das Erzeugen eines Strings	671
16.2.2	Die Eigenschaften von »String«	672
16.2.3	Die Methoden der Klasse »String«	672
16.2.4	Zusammenfassung der Klasse »String«	683
16.3	Die Klasse »StringBuilder«	684
16.3.1	Allgemeines	684
16.3.2	Die Kapazität eines »StringBuilder«-Objekts	685
16.3.3	Die Konstruktoren der Klasse »StringBuilder«	686
16.3.4	Die Eigenschaften der Klasse »StringBuilder«	686
16.3.5	Die Methoden der Klasse »StringBuilder«	687
16.3.6	Allgemeine Anmerkungen	689
16.4	Der Typ »DateTime«	690
16.4.1	Die Zeitspanne »Tick«	690
16.4.2	Die Konstruktoren von »DateTime«	691
16.4.3	Die Eigenschaften von »DateTime«	692
16.4.4	Die Methoden der Klasse »DateTime«	693
16.5	Die Klasse »TimeSpan«	694
16.6	Ausgabeformatierung	697
16.6.1	Formatierung mit der Methode »String.Format«	697
16.6.2	Formatierung mit der Methode »ToString«	701
16.6.3	Benutzerdefinierte Formatierung	701
17	Projektmanagement und Visual Studio 2012	705
17.1	Der Projekttyp »Klassenbibliothek«	705
17.1.1	Mehrere Projekte in einer Projektmappe verwalten	706
17.1.2	Die Zugriffsmodifizierer »public« und »internal«	707
17.1.3	Friend Assemblies	707
17.1.4	Einbinden einer Klassenbibliothek	708

17.2 Assemblies	709
17.2.1 Ein Überblick über das Konzept der Assemblies	709
17.2.2 Allgemeine Beschreibung privater und globaler Assemblies	710
17.2.3 Die Struktur einer Assembly	711
17.2.4 Globale Assemblies	716
17.3 Konfigurationsdateien	721
17.3.1 Die verschiedenen Konfigurationsdateien	721
17.3.2 Die Struktur einer Anwendungskonfigurationsdatei	723
17.3.3 Eine Anwendungskonfigurationsdatei mit Visual Studio 2012 bereitstellen	726
17.3.4 Einträge der Anwendungskonfigurationsdatei auswerten	727
17.3.5 Editierbare, anwendungsbezogene Einträge mit <appSettings>	732
17.4 Versionsumleitung in einer Konfigurationsdatei	734
17.4.1 Die Herausgeberrichtliniendatei	735
17.5 XML-Dokumentation	736
17.5.1 Das Prinzip der XML-Dokumentation	737
17.5.2 Die XML-Kommentartags	739
17.5.3 Generieren der XML-Dokumentationsdatei	740
17.6 Der Klassendesigner (Class Designer)	742
17.6.1 Ein typisches Klassendiagramm	742
17.6.2 Hinzufügen und Ansicht von Klassendiagrammen	743
17.6.3 Die Toolbox des Klassendesigners	744
17.6.4 Das Fenster »Klassendetails«	745
17.6.5 Klassendiagramme als Bilder exportieren	747
17.7 Refactoring	747
17.7.1 Methode extrahieren	748
17.7.2 Bezeichner umbenennen	749
17.7.3 Felder einkapseln	750
17.8 Code-Snippets (Codeausschnitte)	750
17.8.1 Codeausschnitte einfügen	751
17.8.2 Die Anatomie eines Codeausschnitts	752
17.9 »ClickOnce«-Verteilung	753
17.9.1 Allgemeine Beschreibung	753
17.9.2 Erstellen einer ClickOnce-Anwendung	754
17.9.3 Die Installation einer ClickOnce-Anwendung	757
18 Einführung in die WPF und XAML	759
18.1 Die Merkmale einer WPF-Anwendung	759
18.1.1 Anwendungstypen	761

18.1.2	Eine WPF-Anwendung und deren Dateien	762
18.1.3	Ein erstes WPF-Beispiel	765
18.1.4	Wichtige WPF-Features	768
18.1.5	Der logische und der visuelle Elementbaum	770
18.2	XAML (Extended Application Markup Language)	773
18.2.1	Die Struktur einer XAML-Datei	774
18.2.2	Eigenschaften eines XAML-Elements in Attributschreibweise festlegen ...	776
18.2.3	Eigenschaften im Eigenschaftsfenster festlegen	776
18.2.4	Die Eigenschaft-Element-Syntax	777
18.2.5	Inhaltseigenschaften	778
18.2.6	Typkonvertierung	781
18.2.7	Markup-Erweiterungen (Markup Extensions)	782
18.2.8	XML-Namespaces	785
18.2.9	XAML-Spracherweiterungen	787
19	WPF-Layout-Container	789
19.1	Die Container-Steuerelemente	789
19.1.1	Gemeinsame Eigenschaften der Layout-Container	790
19.1.2	Das »Canvas«	791
19.1.3	Das »StackPanel«	792
19.1.4	Das »WrapPanel«	795
19.1.5	Das »DockPanel«	796
19.1.6	Das »Grid«-Steuerelement	798
19.1.7	Das »UniformGrid«	804
19.2	Verschachteln der Layout-Container	805
20	Fenster in der WPF	809
20.1	Hosts der WPF	809
20.2	Fenster vom Typ »Window«	810
20.2.1	Mehrere Fenster in einer Anwendung	812
20.3	Fenster vom Typ »NavigationWindow«	814
20.3.1	Das »Page«-Element	816
20.4	Hosts vom Typ »Frame«	817
20.5	Navigation zwischen den Seiten	818
20.5.1	Navigation mit »HyperLink«	819
20.5.2	Der Verlauf der Navigation – das Journal	820
20.5.3	Navigation mit »NavigationService«	822
20.5.4	Navigation im Internet	824

20.5.5 Navigieren mit dem Ereignis »RequestNavigate« des »HyperLink«-Elements	825
20.6 Datenübergabe zwischen den Seiten	825
20.6.1 Datenübergabe mit der Methode »Navigate«	826
20.7 Nachrichtenfenster mit »MessageBox«	828
20.7.1 Die Methode »MessageBox.Show«	829
21 WPF-Steuerelemente	833
21.1 Die Hierarchie der WPF-Komponenten	833
21.2 Allgemeine Eigenschaften der WPF-Steuerelemente	835
21.2.1 Den Außenrand mit der Eigenschaft »Margin« festlegen	835
21.2.2 Den Innenrand mit der Eigenschaft »Padding« festlegen	835
21.2.3 Die Eigenschaft »Content«	836
21.2.4 Die Größe einer Komponente	838
21.2.5 Die Ausrichtung einer Komponente	839
21.2.6 Die Sichtbarkeit eines Steuerelements	840
21.2.7 Die Farbeinstellungen	841
21.2.8 Die Schriften	842
21.3 Die unterschiedlichen Schaltflächen	842
21.3.1 Die Basisklasse »ButtonBase«	843
21.3.2 Das Steuerelement »Button«	843
21.3.3 Das Steuerelement »ToggleButton«	844
21.3.4 Das Steuerelement »RepeatButton«	845
21.3.5 Das Steuerelement »Checkbox«	847
21.3.6 Das Steuerelement »RadioButton«	847
21.4 Einfache Eingabesteuerelemente	848
21.4.1 Das Steuerelement »Label«	848
21.4.2 Das Steuerelement »TextBox«	849
21.4.3 Das Steuerelement »PasswordBox«	852
21.4.4 Das Steuerelement »TextBlock«	853
21.5 WPF-Listenelemente	856
21.5.1 Das Steuerelement »ListBox«	857
21.5.2 Die »ComboBox«	860
21.5.3 Das Steuerelement »ListView«	861
21.5.4 Das Steuerelement »TreeView«	863
21.5.5 Das Steuerelement »TabControl«	869
21.5.6 Die Menüleiste	870
21.5.7 Das Kontextmenü	873

21.5.8	Symbolleisten	875
21.5.9	Die Statusleiste	878
21.6	Weitere Steuerelemente	879
21.6.1	Das Steuerelement »ToolTip«	879
21.6.2	Die »Progressbar«	881
21.6.3	Das Steuerelement »Slider«	881
21.6.4	Das »GroupBox«-Steuerelement	882
21.6.5	Das Steuerelement »ScrollViewer«	883
21.6.6	Das Steuerelement »Expander«	885
21.6.7	Das Steuerelement »Border«	886
21.6.8	Die »Image«-Komponente	887
21.6.9	»Calendar« und »DatePicker« zur Datumsangabe	889
21.6.10	Das Steuerelement »InkCanvas«	890
21.7	Das »Ribbon«-Steuerelement	893
21.7.1	Voraussetzungen für den Zugriff auf das »Ribbon«-Control	893
21.7.2	Ein kurzer Überblick	893
21.7.3	Der XAML-Code	894
21.8	FlowDocuments	899
21.8.1	Allgemeine Beschreibung eines FlowDocuments	899
21.8.2	Eigenschaften eines »FlowDocuments«	900
21.8.3	Die Blöcke eines »FlowDocuments«	900
21.8.4	Inline-Elemente	905
21.8.5	»FlowDocuments« mit Code erzeugen	907
21.8.6	Speichern und Laden eines »FlowDocuments«	910
21.9	Das Element »FlowDocumentViewer«	911
21.9.1	Das Anzeigeelement »FlowDocumentScrollView«	911
21.9.2	Das Anzeigeelement »FlowDocumentPageViewer«	912
21.9.3	Das Anzeigeelement »FlowDocumentReader«	912
21.10	XPS-Dokumente mit »DocumentViewer«	913
21.10.1	Allgemeines zum XPS-Format	913
21.10.2	Beispielprogramm	914
21.11	Das Steuerelement »RichTextBox«	915
22	Elementbindungen	921
22.1	Einführung in die Bindungstechnik	921
22.1.1	Ein einfaches Bindungsbeispiel	921
22.2	Die Klasse »Binding«	924
22.2.1	Die Bindungsrichtung festlegen	925
22.2.2	Aktualisierung der Bindung	928

22.2.3	Die Ereignisse »SourceUpdated« und »TargetUpdated«	930
22.2.4	Beenden einer Bindung	931
22.3	Bindungsalternativen	931
22.3.1	Die Eigenschaft »Source«	931
22.3.2	Anbindung an relative Datenquellen	932
22.3.3	Die Bindung an »DataContext«	934
23	Konzepte von WPF	935
23.1	Anwendungsspezifische Ressourcen	935
23.2	Anwendungsübergreifende Ressourcen	937
23.2.1	Mehrere Ressourcenwörterbücher	939
23.2.2	Die Suche nach einer Ressource	940
23.3	Logische Ressourcen	940
23.3.1	Statische Ressourcen	941
23.3.2	Dynamische Ressourcen	944
23.3.3	Ressourcen mit C#-Code bearbeiten	945
23.3.4	Abrufen von Systemressourcen	946
23.4	Styles	948
23.4.1	Einfache Styles	948
23.4.2	Typisierte Styles	952
23.4.3	Erweitern von Styles	953
23.4.4	EventSetter	954
23.5	Trigger	956
23.5.1	Eigenschaftstrigger	957
23.5.2	Datentrigger	960
23.5.3	Ereignistrigger	961
23.6	Templates	962
23.6.1	Allgemeines zu »ControlTemplates«	963
23.6.2	Definition innerhalb eines Styles	968
23.7	Ermitteln des visuellen Elementbaums	969
23.7.1	Das Tool »Expression Blend«	969
23.7.2	Standard-Template mit Code abfragen	971
24	Datenbindung	975
24.1	Bindung benutzerdefinierter Objekte	975
24.1.1	Ein Objekt mit XAML-Code erzeugen und binden	976
24.1.2	Ein Objekt mit C#-Code erzeugen und binden	977
24.1.3	Aktualisieren benutzerdefinierter Objekte	979

24.2 Auflistungen binden	981
24.2.1 Allgemeine Gesichtspunkte	981
24.2.2 Anbindung an eine »ListBox«	982
24.2.3 Änderungen der Collection an die bindenden Elemente weiterleiten	984
24.3 Validieren von Bindungen	987
24.3.1 Die Validierung im Datenobjekt	988
24.3.2 Eine benutzerdefinierte »ValidationRule«	990
24.3.3 Validierung mit der Schnittstelle »IDataErrorInfo«	991
24.3.4 Fehlerhinweise individuell anzeigen	993
24.3.5 Ereignisauslösung bei einem Validierungsfehler	995
24.4 Daten konvertieren	995
24.4.1 Mehrfachbindungen und Konverterklassen	999
24.5 Datenbindung an ADO.NET- und LINQ-Datenquellen	1000
24.5.1 Das Binden an ADO.NET-Objekte	1001
24.5.2 Das Binden an LINQ-Ausdrücke	1002
<hr/> 25 Weitere Möglichkeiten der Datenbindung	1005
25.1 »ItemsControl«-Steuerelemente anpassen	1005
25.1.1 Den Style eines »ListBoxItem«-Elements ändern	1006
25.1.2 DataTemplates festlegen	1008
25.1.3 »DataTemplates« mit Trigger	1010
25.2 Alternative Datenbindungen	1014
25.2.1 Die Klasse »ObjectDataProvider«	1014
25.3 Navigieren, Filtern, Sortieren und Gruppieren	1016
25.3.1 Navigieren	1018
25.3.2 Sortieren	1021
25.3.3 Filtern	1022
25.3.4 Gruppieren	1026
25.4 Das Steuerelement »DataGridView«	1030
25.4.1 Elementare Eigenschaften des »DataGridView«	1032
25.4.2 Spalten definieren	1033
25.4.3 Details einer Zeile anzeigen	1039
<hr/> 26 Dependency Properties	1041
26.1 Die Charakteristik von Abhängigkeitseigenschaften	1041
26.2 Den Wert einer Abhängigkeitseigenschaft bilden	1042
26.3 Definition einer Dependency Property	1043
26.3.1 Registrieren einer Abhängigkeitseigenschaft	1044

26.3.2	Der Eigenschaftswrapper	1045
26.3.3	Die Eigenschaftsmetadaten	1046
26.3.4	Freigabe des spezifischen Eigenschaftswertes	1050
26.3.5	Vererbung von Abhangigkeitseigenschaften	1050
26.4	Validieren einer Abhangigkeitseigenschaft	1051
26.4.1	Validieren mit »ValidateValueCallback«	1051
26.4.2	Validieren mit »CoerceValueCallback«	1052
26.5	Angehangte Eigenschaften (Attached Property)	1053
26.5.1	Angehangte Eigenschaften zur Laufzeit ndern	1055
27	Ereignisse in der WPF	1057
27.1	Ereignishandler bereitstellen	1057
27.2	Routing-Strategien	1058
27.2.1	Der durchlaufene Elementbaum	1060
27.2.2	Beispielanwendung	1060
27.2.3	Sonderfall der Mausereignisse	1062
27.3	Der Ereignishandler	1063
27.3.1	Die Klasse »RoutedEventArgs«	1063
27.3.2	Die Quelle des Routing-Prozesses	1064
27.3.3	Die Eigenschaft »Handled«	1065
27.3.4	Registrieren und Deregistrieren eines Ereignishandlers mit Code	1066
27.4	Definition eines Routed Events	1066
27.4.1	Ereignisauslsung	1068
27.4.2	Das Ereignis als Attached Event verwenden	1069
27.4.3	Unterdrckte Ereignisse	1070
27.5	Mausereignisse in der WPF	1071
27.5.1	Ziehen der Maus	1071
27.5.2	Auswerten der Mausklicks	1072
27.5.3	Capturing	1073
28	WPF-Commands	1077
28.1	Allgemeine Beschreibung	1077
28.1.1	Ein einfhrendes Beispiel	1077
28.2	Vordefinierte WPF-Commands	1079
28.3	Commands verwenden	1080
28.3.1	Command-Bindungen einrichten	1081
28.3.2	Lokalitt der Befehlsbindung	1083
28.3.3	Befehlsbindung mit Programmcode	1083

28.3.4	Das Befehlsziel mit »CommandTarget« angeben	1084
28.3.5	Zusätzliche Daten bereitstellen	1085
28.3.6	Befehle mit Maus oder Tastatur aufrufen	1086
28.4	Die Anatomie eines »Command«-Objekts	1087
28.4.1	Das Interface » ICommand «	1088
28.4.2	Die Klassen » RoutedCommand « und » RoutedUICommand «	1088
28.4.3	Das Interface » ICommandSource «	1090
28.5	Das MVVM-Pattern	1091
28.5.1	Ein simples Beispielprogramm	1093
29	Benutzerdefinierte Controls	1097
29.1	Erstellen eines benutzerdefinierten Steuerelements	1097
29.2	Der XAML-Code	1099
29.3	Die Programmlogik des Steuerelements	1100
29.3.1	Die Eigenschaften	1100
29.3.2	Ein Ereignis bereitstellen	1102
29.3.3	Das Steuerelement um einen »Command« ergänzen	1103
29.4	Testanwendung	1104
30	2D-Grafik	1107
30.1	Shapes	1107
30.1.1	Allgemeine Beschreibung	1107
30.1.2	Line-Elemente	1108
30.1.3	Ellipse- und Rectangle-Elemente	1109
30.1.4	Polygon- und Polyline-Elemente	1109
30.1.5	Darstellung der Linien	1109
30.2	Path-Elemente	1111
30.2.1	GeometryGroup	1112
30.2.2	CombinedGeometry	1113
30.2.3	PathGeometry	1114
30.3	Brush-Objekte	1115
30.3.1	SolidColorBrush	1116
30.3.2	LinearGradientBrush	1117
30.3.3	RadialGradientBrush	1119
30.3.4	TileBrush	1120
30.3.5	ImageBrush	1122
30.3.6	VisualBrush	1123
30.3.7	DrawingBrush	1125

31 ADO.NET – Verbindungsorientierte Objekte	1127
31.1 Allgemeines	1127
31.2 Die Datenprovider	1128
31.3 Die Verbindung zu einer Datenbank herstellen	1129
31.3.1 Das Connection-Objekt	1129
31.3.2 Die Verbindungszeichenfolge	1130
31.3.3 Die Verbindung mit einer SQL Server-Instanz aufbauen	1131
31.3.4 Öffnen und Schließen einer Verbindung	1134
31.3.5 Das Verbindingspooling	1138
31.3.6 Die Ereignisse eines »Connection«-Objekts	1142
31.3.7 Verbindungszeichenfolgen aus einer Konfigurationsdatei abrufen	1144
31.3.8 Verbindungen mit dem OleDb-Datenprovider	1146
32 ADO.NET – Das Command-Objekt	1149
32.1 Die Datenbankabfrage	1149
32.2 Das SqlCommand-Objekt	1149
32.2.1 Erzeugen eines SqlCommand-Objekts	1150
32.2.2 Die Methode »CreateCommand« des Connection-Objekts	1151
32.2.3 Ausführen des SqlCommand-Objekts	1151
32.2.4 Die Eigenschaft »CommandTimeout« des SqlCommand-Objekts	1152
32.3 Aktionsabfragen absetzen	1152
32.3.1 Datensätze hinzufügen	1152
32.3.2 Datensätze löschen	1153
32.3.3 Datensätze ändern	1154
32.3.4 Abfragen, die genau ein Ergebnis liefern	1154
32.4 Das SqlDataReader-Objekt	1154
32.4.1 Datensätze einlesen	1155
32.4.2 Schließen des SqlDataReader-Objekts	1157
32.4.3 MARS (Multiple Active Resultsets)	1158
32.4.4 Batchabfragen mit »NextResult« durchlaufen	1159
32.4.5 Das Schema eines SqlDataReader-Objekts untersuchen	1160
32.5 Parametrisierte Abfragen	1162
32.5.1 Parametrisierte Abfragen mit dem SqlClient-Datenprovider	1162
32.5.2 Die Klasse »SqlParameter«	1165
32.5.3 Asynchrone Abfragen	1165
32.5.4 Gespeicherte Prozeduren (Stored Procedures)	1169

33 ADO.NET – Der SqlDataAdapter	1177
33.1 Was ist ein DataAdapter?	1177
33.2 Die Konstruktoren der Klasse DataAdapter	1179
33.3 Arbeiten mit dem SqlDataAdapter	1179
33.3.1 Die Eigenschaft »SelectCommand«	1179
33.3.2 Den lokalen Datenspeicher mit »Fill« füllen	1180
33.3.3 Öffnen und Schließen von Verbindungen	1181
33.3.4 Doppelter Aufruf der Fill-Methode	1182
33.3.5 Mehrere DataAdapter-Objekte aufrufen	1182
33.3.6 Die Spalten- und der Tabellenbezeichner einer DataTable	1183
33.3.7 Paging mit der Fill-Methode	1183
33.4 Tabellenzuordnung mit der Klasse »TableMappings«	1184
33.4.1 Spaltenzuordnungen in einem DataSet	1186
33.4.2 Spaltenzuordnungen einer DataTable	1187
33.4.3 Die Eigenschaft »MissingMappingAction« des DataAdapters	1188
33.5 Das Ereignis »FillError« des SqlDataAdapter	1188
34 ADO.NET – Daten im lokalen Speicher	1191
34.1 Allgemeines	1191
34.2 Verwenden des DataSet-Objekts	1192
34.2.1 Ein DataSet-Objekt erzeugen	1192
34.2.2 Die Anatomie einer DataTable	1192
34.2.3 Der Zugriff auf eine Tabelle im DataSet	1193
34.2.4 Der Zugriff auf die Ergebnisliste	1194
34.2.5 Dateninformationen in eine XML-Datei schreiben	1195
34.3 Gültigkeitsprüfung im DataSet	1196
34.3.1 Dem DataSet Schemainformationen übergeben	1196
34.3.2 Eigenschaften einer DataColumn, die der Gültigkeitsprüfung dienen	1198
34.3.3 Die Constraints-Klassen einer »DataTable«	1199
34.3.4 Das Schema mit Programmcode erzeugen	1200
34.3.5 Schemainformationen mit SqlDataAdapter abrufen	1201
34.4 Änderungen in einer DataTable vornehmen	1204
34.4.1 Editieren einer DataRow	1204
34.4.2 Löschen einer Datenzeile	1206
34.4.3 Eine neue Datenzeile hinzufügen	1206
34.4.4 Der Sonderfall: Autoinkrementspalten	1207
34.4.5 Was bei einer Änderung einer Datenzeile passiert	1209
34.4.6 Manuelles Steuern der Eigenschaft »DataRowState«	1213

34.5 Mit mehreren Tabellen arbeiten	1214
34.5.1 Der Weg über JOIN-Abfragen	1214
34.5.2 Mehrere Tabellen in einem DataSet	1216
34.5.3 Eine DataRelation erzeugen	1216
34.5.4 DataRelations und Einschränkungen	1217
34.5.5 In Beziehung stehende Daten suchen	1219
34.5.6 Ergänzung zum Speichern von Schemainformationen in einer XML-Schemadatei	1221
34.6 Filtern und suchen in einer DataTable	1222
34.6.1 Die Methode »Find«	1222
34.6.2 Die Methode »Select«	1223
34.7 Objekte vom Typ »DataView«	1224
34.7.1 Einen »DataView« erzeugen	1225
34.7.2 Auf die Datenzeilen in einem »DataView« zugreifen	1225
34.7.3 Die Eigenschaft »Sort« und die Methode »Find«	1226
34.7.4 Die Methode »FindRows«	1226
34.7.5 Die Eigenschaft »RowFilter«	1227
34.7.6 Die Eigenschaft »RowStateFilter«	1227
34.7.7 Änderungen an einem »DataView«-Objekt	1227
34.7.8 Aus einem »DataView« eine »DataTable« erzeugen	1229
35 ADO.NET – Aktualisieren der Datenbank	1231
35.1 Aktualisieren mit dem »CommandBuilder«	1231
35.1.1 Die von »SqlCommandBuilder« generierten Aktualisierungsstatements	1233
35.1.2 Konfliktsteuerung in einer Mehrbenutzerumgebung	1233
35.1.3 Die Eigenschaft »ConflictOption« des »SqlCommandBuilders«	1236
35.1.4 Die Eigenschaft »SetAllValues«	1237
35.2 Manuell gesteuerte Aktualisierung	1238
35.2.1 Eigene Aktualisierungslogik	1239
35.2.2 Das Beispielprogramm	1240
35.3 Konfliktanalyse	1242
35.3.1 Den Benutzer über fehlgeschlagene Aktualisierungen informieren	1243
35.3.2 Konfliktverursachende Datenzeilen bei der Datenbank abfragen	1244
35.4 Neue Autoinkrementwerte abrufen	1249
36 Stark typisierte DataSets	1251
36.1 Ein stark typisiertes DataSet erzeugen	1251
36.1.1 Typisierte DataSets mit dem Visual Studio Designer erstellen	1251

36.1.2 Das Kommandozeilentool XSD.exe	1254
36.2 Die Anatomie eines typisierten DataSets	1255
36.2.1 Die Datenzeilen einer Tabelle ausgeben	1255
36.2.2 Datenzeilen hinzufügen	1258
36.2.3 Datenzeilen bearbeiten	1259
36.2.4 Datenzeilen suchen	1259
36.2.5 NULL-Werte im typisierten DataSet	1260
36.2.6 Die Daten in einem hierarchischen DataSet	1260
36.3 Typisierte DataSets manuell im Designer erzeugen	1261
36.3.1 Eine »DataTable« manuell erzeugen	1261
36.3.2 Der »DataTable« Spalten hinzufügen	1262
36.3.3 Beziehungen zwischen den Tabellen erstellen	1262
36.4 Weiter gehende Betrachtungen	1264
36.5 Der »TableAdapter«	1264
36.5.1 Einen »TableAdapter« mit Visual Studio erzeugen	1264
36.5.2 Die Methode »Fill« des »TableAdapters«	1269
36.5.3 Die Methode »GetData«	1270
36.5.4 Die Methode »Update«	1270
36.5.5 Aktualisieren mit den DBDirect-Methoden	1270
36.5.6 TableAdapter mit mehreren Abfragen	1271
36.5.7 Änderungen an einem »TableAdapter« vornehmen	1274
36.6 Fazit: Typisierte oder nicht typisierte DataSets?	1275
37 Einführung in das ADO.NET Entity Framework	1277
37.1 Kritische Betrachtung von ADO.NET	1277
37.2 Ein erstes Entity Data Model (EDM) erstellen	1279
37.3 Das Entity Data Model im Designer	1283
37.3.1 Die übergeordneten Eigenschaften einer Entität	1283
37.3.2 Eigenschaften eines Entitätsobjekts	1284
37.3.3 Assoziationen im Entity Data Model	1287
37.3.4 Der Kontext der Entitäten	1288
37.4 Der Aufbau des Entity Data Models	1289
37.5 Die Klassen des Entity Data Models (EDM)	1292
37.5.1 Die Entitätsklassen	1293
37.5.2 Der ObjectContext	1296
37.6 Die Architektur des Entity Frameworks	1297
37.6.1 Object Services	1298
37.6.2 Die Schichten des Entity Frameworks	1298

38 Datenabfragen des Entity Data Models (EDM)	1301
<hr/>	
38.1 Abfragen mit LINQ to Entities	1302
38.1.1 Allgemeine Begriffe in LINQ	1302
38.1.2 Einfache Abfragen	1302
38.1.3 Navigieren in Abfragen	1309
38.1.4 Aggregatmethoden	1314
38.1.5 Joins in LINQ definieren	1315
38.1.6 In Beziehung stehende Daten laden	1318
38.2 Abfragen mit Entity SQL	1324
38.2.1 Ein erstes Beispiel mit Entity SQL	1324
38.2.2 Die fundamentalen Regeln der Entity-SQL-Syntax	1325
38.2.3 Filtern mit Entity SQL	1326
38.2.4 Parametrisierte Abfragen	1328
38.3 Der EntityClient-Provider	1329
38.3.1 Verbindungen mit »EntityConnection«	1330
38.3.2 Die Klasse »EntityCommand«	1331
38.4 Abfrage-Generator-Methoden (QueryBuilder-Methoden)	1332
38.5 SQL-Direktabfragen	1333
39 Entitätsaktualisierung und Zustandsverwaltung	1335
<hr/>	
39.1 Aktualisieren von Entitäten	1335
39.1.1 Entitäten ändern	1335
39.1.2 Hinzufügen neuer Entitäten	1337
39.1.3 Löschen einer Entität	1341
39.2 Der Lebenszyklus einer Entität im Objektkontext	1344
39.2.1 Der Zustand einer Entität	1344
39.2.2 Das Team der Objekte im Überblick	1344
39.2.3 Neue Entitäten im Objektkontext	1345
39.2.4 Die Zustände einer Entität	1347
39.2.5 Zusätzliche Entitäten in den Datencache laden	1349
39.2.6 Die Zustandsverfolgung mit »MergeOption« steuern	1349
39.3 Das »ObjectStateEntry«-Objekt	1352
39.3.1 Die Current- und Originalwerte abrufen	1354
39.3.2 Die Methode »TryGetObjectStateEntry«	1355
39.3.3 Abrufen bestimmter Gruppen	1355
39.3.4 Die Methode »GetModifiedProperties«	1356
39.4 Die Klasse »EntityKey«	1357
39.4.1 Die Methoden »GetObjectByKey« und »TryGetObjectByKey«	1357

39.5 Komplexere Szenarien	1358
39.5.1 Die Methode »ChangeState«	1359
39.5.2 Die Methoden »ApplyCurrentChanges« und »ApplyOriginalChanges«	1360
40 Konflikte behandeln	1363
40.1 Allgemeine Betrachtungen	1363
40.1.1 Das pessimistische Sperren	1364
40.1.2 Das optimistische Sperren	1364
40.2 Konkurrierende Zugriffe mit dem Entity Framework	1365
40.2.1 Das Standardverhalten des Entity Frameworks	1365
40.2.2 Das Aktualisierungsverhalten mit »Fixed« beeinflussen	1366
40.2.3 Auf die Ausnahme »OptimisticConcurrencyException« reagieren	1367
40.2.4 Das »ClientWins«-Szenario	1368
40.2.5 Das »StoreWins«-Szenario	1370
41 Plain Old CLR Objects (POCOs)	1371
41.1 Ein erstes Projekt mit POCO-Klassen	1371
41.1.1 Erstellen einfacher POCO-Klassen	1371
41.1.2 Erstellen des Objektkontextes	1373
41.2 Datenabfrage mit Hilfe der POCOs	1375
41.2.1 In Beziehung stehende Daten laden	1375
41.3 Änderungen verfolgen	1377
41.3.1 Die Methode »DetectChanges«	1377
41.3.2 In Beziehung stehende POCOs aktualisieren	1379
Index	1385