

jürgen KOTZ

Visual C# 2019

GRUNDLAGEN
PROFIWISSEN
REZEPTE

// C#-Grundlagen
// LINQ, OOP, ADO.NET, EF und WPF
// Über 150 Praxisbeispiele



Im Internet: Bonuskapitel und
Code-Beispiele

HANSER

Kotz

Visual C# 2019
Grundlagen, Profiwissen und Rezepte



Bleiben Sie auf dem Laufenden!

Unser **Computerbuch-Newsletter** informiert Sie monatlich über neue Bücher und Termine. Profitieren Sie auch von Gewinnspielen und exklusiven Leseproben. Gleich anmelden unter:

www.hanser-fachbuch.de/newsletter



Jürgen Kotz

Visual C# 2019

Grundlagen, Profiwissen und Rezepte

HANSER

Der Autor:
Jürgen Kotz, München

Alle in diesem Buch enthaltenen Informationen, Verfahren und Darstellungen wurden nach bestem Wissen zusammengestellt und mit Sorgfalt getestet. Dennoch sind Fehler nicht ganz auszuschließen. Aus diesem Grund sind die im vorliegenden Buch enthaltenen Informationen mit keiner Verpflichtung oder Garantie irgendeiner Art verbunden. Autor und Verlag übernehmen infolgedessen keine juristische Verantwortung und werden keine daraus folgende oder sonstige Haftung übernehmen, die auf irgendeine Art aus der Benutzung dieser Informationen – oder Teilen davon – entsteht.

Ebenso übernehmen Autor und Verlag keine Gewähr dafür, dass beschriebene Verfahren usw. frei von Schutzrechten Dritter sind. Die Wiedergabe von Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. in diesem Buch berechtigt deshalb auch ohne besondere Kennzeichnung nicht zu der Annahme, dass solche Namen im Sinne der Warenzeichen- und Markenschutz-Gesetzgebung als frei zu betrachten wären und daher von jedermann benutzt werden dürften.

Bibliografische Information der Deutschen Nationalbibliothek:

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.d-nb.de> abrufbar.

Dieses Werk ist urheberrechtlich geschützt.

Alle Rechte, auch die der Übersetzung, des Nachdruckes und der Vervielfältigung des Buches, oder Teilen daraus, vorbehalten. Kein Teil des Werkes darf ohne schriftliche Genehmigung des Verlages in irgendeiner Form (Fotokopie, Mikrofilm oder ein anderes Verfahren) – auch nicht für Zwecke der Unterrichtsgestaltung – reproduziert oder unter Verwendung elektronischer Systeme verarbeitet, vervielfältigt oder verbreitet werden.

© 2019 Carl Hanser Verlag München, www.hanser-fachbuch.de

Lektorat: Sylvia Hasselbach

Copy editing: Walter Saumweber, Ratingen

Umschlagdesign: Marc Müller-Bremer, München, www.rebranding.de

Umschlagrealisation: Max Kostopoulos

Titelmotiv: © thinkstockphotos.de/liuzishan

Gesamtherstellung: Kösel, Krugzell

Ausstattung patentrechtlich geschützt. Kösel FD 351, Patent-Nr. 0748702

Printed in Germany

Print-ISBN: 978-3-446-45802-4

E-Book-ISBN: 978-3-446-46099-7

E-Pub-ISBN: 978-3-446-46253-3

Inhalt

| | |
|--|--------------|
| Vorwort | XXIII |
| Teil I: Grundlagen | 1 |
| 1 Einstieg in Visual Studio 2019 | 3 |
| 1.1 Die Installation von Visual Studio 2019 | 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 | 8 |
| 1.2.4 Einige Erläuterungen zum Quellcode | 9 |
| 1.2.5 Konsolenanwendungen sind out | 10 |
| 1.3 Die Windows-Philosophie | 10 |
| 1.3.1 Mensch-Rechner-Dialog | 10 |
| 1.3.2 Objekt- und ereignisorientierte Programmierung | 11 |
| 1.3.3 Programmieren mit Visual Studio 2019 | 12 |
| 1.4 Die Entwicklungsumgebung Visual Studio 2019 | 13 |
| 1.4.1 Neues Projekt | 14 |
| 1.4.2 Die wichtigsten Fenster | 16 |
| 1.5 Microsofts .NET-Technologie | 20 |
| 1.5.1 Zur Geschichte von .NET | 20 |
| 1.5.2 .NET-Features und Begriffe | 22 |
| 1.6 Praxisbeispiele | 29 |
| 1.6.1 Unsere erste Windows-Forms-Anwendung | 30 |
| 1.6.2 Umrechnung Euro-Dollar | 35 |
| 2 Grundlagen der Sprache C# | 45 |
| 2.1 Grundbegriffe | 45 |
| 2.1.1 Anweisungen | 45 |
| 2.1.2 Bezeichner | 46 |

| | | |
|--------|--|----|
| 2.1.3 | Schlüsselwörter | 47 |
| 2.1.4 | Kommentare | 48 |
| 2.2 | Datentypen, Variablen und Konstanten | 49 |
| 2.2.1 | Fundamentale Typen | 49 |
| 2.2.2 | Werttypen versus Verweistypen | 50 |
| 2.2.3 | Benennung von Variablen | 51 |
| 2.2.4 | Deklaration von Variablen | 51 |
| 2.2.5 | Typsuffixe | 52 |
| 2.2.6 | Zeichen und Zeichenketten | 53 |
| 2.2.7 | object-Datentyp | 55 |
| 2.2.8 | Konstanten deklarieren | 56 |
| 2.2.9 | Nullable Types | 56 |
| 2.2.10 | Typinferenz | 58 |
| 2.2.11 | Gültigkeitsbereiche und Sichtbarkeit | 59 |
| 2.3 | Konvertieren von Datentypen | 59 |
| 2.3.1 | Implizite und explizite Konvertierung | 59 |
| 2.3.2 | Welcher Datentyp passt zu welchem? | 62 |
| 2.3.3 | Konvertieren von string | 62 |
| 2.3.4 | Die Convert-Klasse | 64 |
| 2.3.5 | Die Parse-Methode | 65 |
| 2.3.6 | Boxing und Unboxing | 65 |
| 2.4 | Operatoren | 67 |
| 2.4.1 | Arithmetische Operatoren | 68 |
| 2.4.2 | Zuweisungsoperatoren | 70 |
| 2.4.3 | Logische Operatoren | 71 |
| 2.4.4 | Rangfolge der Operatoren | 73 |
| 2.5 | Kontrollstrukturen | 74 |
| 2.5.1 | Verzweigungsbefehle | 74 |
| 2.5.2 | Schleifenanweisungen | 79 |
| 2.6 | Benutzerdefinierte Datentypen | 82 |
| 2.6.1 | Enumerationen | 82 |
| 2.6.2 | Strukturen | 83 |
| 2.7 | Nutzerdefinierte Methoden | 86 |
| 2.7.1 | Methoden mit Rückgabewert | 86 |
| 2.7.2 | Methoden ohne Rückgabewert | 88 |
| 2.7.3 | Parameterübergabe mit ref | 89 |
| 2.7.4 | Parameterübergabe mit out | 90 |
| 2.7.5 | Methodenüberladung | 91 |
| 2.7.6 | Optionale Parameter | 92 |
| 2.7.7 | Benannte Parameter | 93 |
| 2.8 | Praxisbeispiele | 94 |
| 2.8.1 | Vom PAP zur Konsolenanwendung | 94 |
| 2.8.2 | Ein Konsolen- in ein Windows-Programm verwandeln | 97 |
| 2.8.3 | Schleifenanweisungen verstehen | 99 |

| | | |
|----------|---|------------|
| 2.8.4 | Benutzerdefinierte Methoden überladen | 101 |
| 2.8.5 | Anwendungen von Visual Basic nach C# portieren | 104 |
| 3 | OOP-Konzepte | 113 |
| 3.1 | Kleine Einführung in die OOP | 113 |
| 3.1.1 | Historische Entwicklung | 114 |
| 3.1.2 | Grundbegriffe der OOP | 115 |
| 3.1.3 | Sichtbarkeit von Klassen und ihren Mitgliedern | 117 |
| 3.1.4 | Allgemeiner Aufbau einer Klasse | 118 |
| 3.1.5 | Das Erzeugen eines Objekts | 120 |
| 3.1.6 | Einführungsbeispiel | 123 |
| 3.2 | Eigenschaften | 128 |
| 3.2.1 | Eigenschaften mit Zugriffsmethoden kapseln | 128 |
| 3.2.2 | Berechnete Eigenschaften | 130 |
| 3.2.3 | Lese-/Schreibschutz | 132 |
| 3.2.4 | Property-Accessoren | 133 |
| 3.2.5 | Statische Felder/Eigenschaften | 134 |
| 3.2.6 | Einfache Eigenschaften automatisch implementieren | 136 |
| 3.3 | Methoden | 138 |
| 3.3.1 | Öffentliche und private Methoden | 138 |
| 3.3.2 | Überladene Methoden | 139 |
| 3.3.3 | Statische Methoden | 140 |
| 3.4 | Ereignisse | 142 |
| 3.4.1 | Ereignis hinzufügen | 142 |
| 3.4.2 | Ereignis verwenden | 145 |
| 3.5 | Arbeiten mit Konstruktor und Destruktor | 148 |
| 3.5.1 | Konstruktor und Objektinitialisierer | 149 |
| 3.5.2 | Destruktor und Garbage Collector | 152 |
| 3.5.3 | Mit using den Lebenszyklus des Objekts kapseln | 154 |
| 3.6 | Vererbung und Polymorphie | 155 |
| 3.6.1 | Method-Overriding | 155 |
| 3.6.2 | Klassen implementieren | 156 |
| 3.6.3 | Implementieren der Objekte | 159 |
| 3.6.4 | Ausblenden von Mitgliedern durch Vererbung | 160 |
| 3.6.5 | Allgemeine Hinweise und Regeln zur Vererbung | 162 |
| 3.6.6 | Polymorphes Verhalten | 163 |
| 3.6.7 | Die Rolle von System.Object | 166 |
| 3.7 | Spezielle Klassen | 167 |
| 3.7.1 | Abstrakte Klassen | 167 |
| 3.7.2 | Versiegelte Klassen | 168 |
| 3.7.3 | Partielle Klassen | 169 |
| 3.7.4 | Statische Klassen | 170 |
| 3.8 | Schnittstellen (Interfaces) | 171 |
| 3.8.1 | Definition einer Schnittstelle | 171 |

| | | |
|----------|--|------------|
| 3.8.2 | Implementieren einer Schnittstelle | 172 |
| 3.8.3 | Abfragen, ob Schnittstelle vorhanden ist | 173 |
| 3.8.4 | Mehrere Schnittstellen implementieren | 173 |
| 3.8.5 | Schnittstellenprogrammierung ist ein weites Feld | 174 |
| 3.9 | Praxisbeispiele | 174 |
| 3.9.1 | Eigenschaften sinnvoll kapseln | 174 |
| 3.9.2 | Eine statische Klasse anwenden | 177 |
| 3.9.3 | Vom fetten zum schlanken Client | 179 |
| 3.9.4 | Schnittstellenvererbung verstehen | 190 |
| 3.9.5 | Rechner für komplexe Zahlen | 194 |
| 3.9.6 | Sortieren mit Comparable/Comparer | 203 |
| 3.9.7 | Einen Objektbaum in generischen Listen abspeichern | 208 |
| 3.9.8 | OOP beim Kartenspiel erlernen | 213 |
| 3.9.9 | Eine Klasse zur Matrizenrechnung entwickeln | 218 |
| 4 | Arrays, Strings, Funktionen | 225 |
| 4.1 | Datenfelder (Arrays) | 225 |
| 4.1.1 | Array deklarieren | 226 |
| 4.1.2 | Array instanzieren | 226 |
| 4.1.3 | Array initialisieren | 227 |
| 4.1.4 | Zugriff auf Array-Elemente | 228 |
| 4.1.5 | Zugriff mittels Schleife | 229 |
| 4.1.6 | Mehrdimensionale Arrays | 230 |
| 4.1.7 | Zuweisen von Arrays | 232 |
| 4.1.8 | Arrays aus Strukturvariablen | 233 |
| 4.1.9 | Löschen und Umdimensionieren von Arrays | 234 |
| 4.1.10 | Eigenschaften und Methoden von Arrays | 236 |
| 4.1.11 | Übergabe von Arrays | 237 |
| 4.2 | Verarbeiten von Zeichenketten | 239 |
| 4.2.1 | Zuweisen von Strings | 239 |
| 4.2.2 | Eigenschaften und Methoden von String-Variablen | 240 |
| 4.2.3 | Wichtige Methoden der String-Klasse | 242 |
| 4.2.4 | Die StringBuilder-Klasse | 244 |
| 4.3 | Reguläre Ausdrücke | 247 |
| 4.3.1 | Wozu werden reguläre Ausdrücke verwendet? | 247 |
| 4.3.2 | Eine kleine Einführung | 248 |
| 4.3.3 | Wichtige Methoden/Eigenschaften der Klasse Regexp | 248 |
| 4.3.4 | Kompilierte reguläre Ausdrücke | 250 |
| 4.3.5 | RegexOptions-Enumeration | 251 |
| 4.3.6 | Metazeichen (Escape-Zeichen) | 252 |
| 4.3.7 | Zeichenmengen (Character Sets) | 253 |
| 4.3.8 | Quantifizierer | 254 |
| 4.3.9 | Zero-Width Assertions | 256 |
| 4.3.10 | Gruppen | 259 |

| | | |
|----------|---|------------|
| 4.3.11 | Text ersetzen | 260 |
| 4.3.12 | Text splitten | 261 |
| 4.4 | Datums- und Zeitberechnungen | 262 |
| 4.4.1 | Die DateTime-Struktur | 262 |
| 4.4.2 | Wichtige Eigenschaften von DateTime-Variablen | 263 |
| 4.4.3 | Wichtige Methoden von DateTime-Variablen | 264 |
| 4.4.4 | Wichtige Mitglieder der DateTime-Struktur | 265 |
| 4.4.5 | Konvertieren von Datumstrings in DateTime-Werte | 265 |
| 4.4.6 | Die TimeSpan-Struktur | 266 |
| 4.5 | Mathematische Funktionen | 268 |
| 4.5.1 | Überblick | 268 |
| 4.5.2 | Zahlen runden | 268 |
| 4.5.3 | Winkel umrechnen | 269 |
| 4.5.4 | Potenz- und Wurzeloperationen | 269 |
| 4.5.5 | Logarithmus und Exponentialfunktionen | 269 |
| 4.5.6 | Zufallszahlen erzeugen | 270 |
| 4.5.7 | Kreisberechnung | 271 |
| 4.6 | Zahlen- und Datumsformatierungen | 271 |
| 4.6.1 | Anwenden der ToString-Methode | 272 |
| 4.6.2 | Anwenden der Format-Methode | 273 |
| 4.6.3 | Stringinterpolation | 274 |
| 4.7 | Praxisbeispiele | 275 |
| 4.7.1 | Zeichenketten verarbeiten | 275 |
| 4.7.2 | Zeichenketten mit StringBuilder addieren | 278 |
| 4.7.3 | Reguläre Ausdrücke testen | 281 |
| 4.7.4 | Methodenaufrufe mit Array-Parametern | 283 |
| 5 | Weitere Sprachfeatures | 287 |
| 5.1 | Namespaces (Namensräume) | 287 |
| 5.1.1 | Ein kleiner Überblick | 287 |
| 5.1.2 | Einen eigenen Namespace einrichten | 288 |
| 5.1.3 | Die using-Anweisung | 289 |
| 5.1.4 | Namespace Alias | 290 |
| 5.2 | Operatorenüberladung | 291 |
| 5.2.1 | Syntaxregeln | 291 |
| 5.2.2 | Praktische Anwendung | 291 |
| 5.3 | Collections (Auflistungen) | 293 |
| 5.3.1 | Die Schnittstelle IEnumerable | 293 |
| 5.3.2 | ArrayList | 295 |
| 5.3.3 | Hashtable | 297 |
| 5.3.4 | Indexer | 297 |
| 5.4 | Generics | 300 |
| 5.4.1 | Generics bieten Typsicherheit | 300 |

| | | |
|----------|--|------------|
| 5.4.2 | Generische Methoden | 301 |
| 5.4.3 | Iteratoren | 302 |
| 5.5 | Generische Collections | 303 |
| 5.5.1 | List-Collection statt ArrayList | 303 |
| 5.5.2 | Vorteile generischer Collections | 304 |
| 5.5.3 | Constraints | 304 |
| 5.6 | Das Prinzip der Delegates | 305 |
| 5.6.1 | Delegates sind Methodenzeiger | 305 |
| 5.6.2 | Einen Delegate-Typ deklarieren | 305 |
| 5.6.3 | Ein Delegate-Objekt erzeugen | 306 |
| 5.6.4 | Delegates vereinfacht instanziiieren | 308 |
| 5.6.5 | Anonyme Methoden | 308 |
| 5.6.6 | Lambda-Ausdrücke | 310 |
| 5.6.7 | Lambda-Ausdrücke in der Task Parallel Library | 312 |
| 5.7 | Dynamische Programmierung | 313 |
| 5.7.1 | Wozu dynamische Programmierung? | 314 |
| 5.7.2 | Das Prinzip der dynamischen Programmierung | 314 |
| 5.7.3 | Optionale Parameter sind hilfreich | 317 |
| 5.7.4 | Kovarianz und Kontravarianz | 317 |
| 5.8 | Weitere Datentypen | 318 |
| 5.8.1 | BigInteger | 318 |
| 5.8.2 | Complex | 321 |
| 5.8.3 | Tuple<> | 321 |
| 5.8.4 | SortedSet<> | 322 |
| 5.9 | Praxisbeispiele | 324 |
| 5.9.1 | ArrayList versus generische List | 324 |
| 5.9.2 | Generische IEnumerable-Interfaces implementieren | 327 |
| 5.9.3 | Delegates, anonyme Methoden, Lambda Expressions | 330 |
| 5.9.4 | Dynamischer Zugriff auf COM Interop | 334 |
| 6 | Einführung in LINQ | 339 |
| 6.1 | LINQ-Grundlagen | 339 |
| 6.1.1 | Die LINQ-Architektur | 339 |
| 6.1.2 | Anonyme Typen | 341 |
| 6.1.3 | Erweiterungsmethoden | 342 |
| 6.2 | Abfragen mit LINQ to Objects | 343 |
| 6.2.1 | Grundlegendes zur LINQ-Syntax | 344 |
| 6.2.2 | Zwei alternative Schreibweisen von LINQ-Abfragen | 345 |
| 6.2.3 | Übersicht der wichtigsten Abfrageoperatoren | 346 |
| 6.3 | LINQ-Abfragen im Detail | 347 |
| 6.3.1 | Die Projektionsoperatoren Select und SelectMany | 348 |
| 6.3.2 | Der Restriktionsoperator Where | 350 |
| 6.3.3 | Die Sortierungsoperatoren OrderBy und ThenBy | 350 |
| 6.3.4 | Der Gruppierungsoperator GroupBy | 352 |

| | | |
|----------|--|------------|
| 6.3.5 | Verknüpfen mit Join | 354 |
| 6.3.6 | Aggregat-Operatoren | 355 |
| 6.3.7 | Verzögertes Ausführen von LINQ-Abfragen | 357 |
| 6.3.8 | Konvertierungsmethoden | 358 |
| 6.3.9 | Abfragen mit PLINQ | 359 |
| 6.4 | Praxisbeispiele | 362 |
| 6.4.1 | Die Syntax von LINQ-Abfragen verstehen | 362 |
| 6.4.2 | Aggregat-Abfragen mit LINQ | 365 |
| 6.4.3 | LINQ im Schnelldurchgang erlernen | 367 |
| 6.4.4 | Strings mit LINQ abfragen und filtern | 370 |
| 6.4.5 | Duplikate aus einer Liste oder einem Array entfernen | 371 |
| 6.4.6 | Arrays mit LINQ initialisieren | 374 |
| 6.4.7 | Arrays per LINQ mit Zufallszahlen füllen | 376 |
| 6.4.8 | Einen String mit Wiederholmuster erzeugen | 378 |
| 6.4.9 | Mit LINQ Zahlen und Strings sortieren | 379 |
| 6.4.10 | Mit LINQ Collections von Objekten sortieren | 380 |
| 6.4.11 | Ergebnisse von LINQ-Abfragen in ein Array kopieren | 383 |
| 7 | Neuerungen von C# im Überblick | 385 |
| 7.1 | C# 4.0 – Visual Studio 2010 | 385 |
| 7.1.1 | Datentyp dynamic | 385 |
| 7.1.2 | Benannte und optionale Parameter | 386 |
| 7.1.3 | Kovarianz und Kontravarianz | 387 |
| 7.2 | C# 5.0 – Visual Studio 2012 | 388 |
| 7.2.1 | Async und Await | 388 |
| 7.2.2 | CallerInfo | 389 |
| 7.3 | Visual Studio 2013 | 390 |
| 7.4 | C# 6.0 – Visual Studio 2015 | 390 |
| 7.4.1 | String Interpolation | 390 |
| 7.4.2 | Schreibgeschützte AutoProperties | 390 |
| 7.4.3 | Initialisierer für AutoProperties | 391 |
| 7.4.4 | Expression Body Funktionsmember | 391 |
| 7.4.5 | using static | 392 |
| 7.4.6 | Bedingter Nulloperator | 392 |
| 7.4.7 | Ausnahmenfilter | 393 |
| 7.4.8 | nameof-Ausdrücke | 393 |
| 7.4.9 | await in catch und finally | 394 |
| 7.4.10 | Indexinitialisierer | 394 |
| 7.5 | C# 7.0 – Visual Studio 2017 | 394 |
| 7.5.1 | out-Variablen | 394 |
| 7.5.2 | Tupel | 395 |
| 7.5.3 | Mustervergleich | 396 |
| 7.5.4 | Discards | 398 |
| 7.5.5 | Lokale ref-Variablen und Rückgabetypen | 398 |

| | | |
|---------------------------------|---|------------|
| 7.5.6 | Lokale Funktionen | 398 |
| 7.5.7 | Mehr Expression-Bodied Member | 399 |
| 7.5.8 | throw-Ausdrücke | 399 |
| 7.5.9 | Verbesserung der numerischen literalen Syntax | 399 |
| 7.6 | C# 7.1 bis 7.3 – Visual Studio 2019 | 400 |
| 7.6.1 | C# 7.1 | 400 |
| 7.6.2 | C# 7.2 | 401 |
| 7.6.3 | C# 7.3 | 402 |
| 7.6.4 | Visual Studio 2019 – Live Share | 403 |
| Teil II: WPF-Anwendungen | | 407 |
| 8 | Einführung in WPF | 409 |
| 8.1 | Einführung | 409 |
| 8.1.1 | Was kann eine WPF-Anwendung? | 410 |
| 8.1.2 | Die eXtensible Application Markup Language | 411 |
| 8.1.3 | Verbinden von XAML und C#-Code | 416 |
| 8.1.4 | Zielpattformen | 421 |
| 8.1.5 | Applikationstypen | 422 |
| 8.1.6 | Vor- und Nachteile von WPF-Anwendungen | 423 |
| 8.1.7 | Weitere Dateien im Überblick | 424 |
| 8.2 | Alles beginnt mit dem Layout | 426 |
| 8.2.1 | Allgemeines zum Layout | 426 |
| 8.2.2 | Positionieren von Steuerelementen | 428 |
| 8.2.3 | Canvas | 432 |
| 8.2.4 | StackPanel | 433 |
| 8.2.5 | DockPanel | 435 |
| 8.2.6 | WrapPanel | 437 |
| 8.2.7 | UniformGrid | 438 |
| 8.2.8 | Grid | 439 |
| 8.2.9 | ViewBox | 444 |
| 8.2.10 | TextBlock | 446 |
| 8.3 | Das WPF-Programm | 449 |
| 8.3.1 | Die App-Klasse | 450 |
| 8.3.2 | Das Startobjekt festlegen | 450 |
| 8.3.3 | Kommandozeilenparameter verarbeiten | 452 |
| 8.3.4 | Die Anwendung beenden | 453 |
| 8.3.5 | Auswerten von Anwendungsereignissen | 453 |
| 8.4 | Die Window-Klasse | 454 |
| 8.4.1 | Position und Größe festlegen | 454 |
| 8.4.2 | Rahmen und Beschriftung | 455 |
| 8.4.3 | Das Fenster-Icon ändern | 456 |
| 8.4.4 | Anzeige weiterer Fenster | 456 |
| 8.4.5 | Transparenz | 456 |

| | | |
|----------|---|------------|
| 8.4.6 | Abstand zum Inhalt festlegen | 457 |
| 8.4.7 | Fenster ohne Fokus anzeigen | 458 |
| 8.4.8 | Ereignisfolge bei Fenstern | 458 |
| 8.4.9 | Ein paar Worte zur Schriftdarstellung | 459 |
| 8.4.10 | Ein paar Worte zur Darstellung von Controls | 462 |
| 8.4.11 | Wird mein Fenster komplett mit WPF gerendert? | 463 |
| 9 | Übersicht WPF-Controls | 465 |
| 9.1 | Allgemeingültige Eigenschaften | 465 |
| 9.2 | Label | 467 |
| 9.3 | Button, RepeatButton, ToggleButton | 468 |
| 9.3.1 | Schaltflächen für modale Dialoge | 468 |
| 9.3.2 | Schaltflächen mit Grafik | 470 |
| 9.4 | TextBox, PasswordBox | 471 |
| 9.4.1 | TextBox | 471 |
| 9.4.2 | PasswordBox | 473 |
| 9.5 | CheckBox | 474 |
| 9.6 | RadioButton | 476 |
| 9.7 | ListBox, ComboBox | 477 |
| 9.7.1 | ListBox | 477 |
| 9.7.2 | ComboBox | 480 |
| 9.7.3 | Den Content formatieren | 482 |
| 9.8 | Image | 484 |
| 9.8.1 | Grafik per XAML zuweisen | 484 |
| 9.8.2 | Grafik zur Laufzeit zuweisen | 484 |
| 9.8.3 | Bild aus Datei laden | 486 |
| 9.8.4 | Die Grafikskalierung beeinflussen | 487 |
| 9.9 | MediaElement | 488 |
| 9.10 | Slider, ScrollBar | 490 |
| 9.10.1 | Slider | 490 |
| 9.10.2 | ScrollBar | 492 |
| 9.11 | ScrollViewer | 492 |
| 9.12 | Menu, ContextMenu | 493 |
| 9.12.1 | Menu | 494 |
| 9.12.2 | Tastenkürzel | 495 |
| 9.12.3 | Grafiken | 496 |
| 9.12.4 | Weitere Möglichkeiten | 498 |
| 9.12.5 | ContextMenu | 498 |
| 9.13 | ToolBar | 499 |
| 9.14 | StatusBar, ProgressBar | 502 |
| 9.14.1 | StatusBar | 502 |
| 9.14.2 | ProgressBar | 504 |

| | | |
|-----------|---|------------|
| 9.15 | Border, GroupBox, BulletDecorator | 505 |
| 9.15.1 | Border | 505 |
| 9.15.2 | GroupBox | 506 |
| 9.15.3 | BulletDecorator | 507 |
| 9.16 | RichTextBox | 509 |
| 9.16.1 | Verwendung und Anzeige von vordefiniertem Text | 510 |
| 9.16.2 | Neues Dokument zur Laufzeit erzeugen | 511 |
| 9.16.3 | Sichern von Dokumenten | 512 |
| 9.16.4 | Laden von Dokumenten | 513 |
| 9.16.5 | Texte per Code einfügen/modifizieren | 514 |
| 9.16.6 | Texte formatieren | 515 |
| 9.16.7 | EditingCommands | 517 |
| 9.16.8 | Grafiken/Objekte einfügen | 518 |
| 9.16.9 | Rechtschreibkontrolle | 519 |
| 9.17 | FlowDocumentPageViewer & Co. | 520 |
| 9.17.1 | FlowDocumentPageViewer | 520 |
| 9.17.2 | FlowDocumentReader | 520 |
| 9.17.3 | FlowDocumentScrollViewer | 521 |
| 9.18 | FlowDocument | 521 |
| 9.18.1 | FlowDocument per XAML beschreiben | 522 |
| 9.18.2 | FlowDocument per Code erstellen | 524 |
| 9.19 | Expander, TabControl | 526 |
| 9.19.1 | Expander | 526 |
| 9.19.2 | TabControl | 527 |
| 9.20 | Popup | 529 |
| 9.21 | TreeView | 531 |
| 9.22 | ListView | 535 |
| 9.23 | DataGrid | 535 |
| 9.24 | Calendar/DatePicker | 536 |
| 9.25 | Ellipse, Rectangle, Line und Co. | 540 |
| 9.25.1 | Ellipse | 541 |
| 9.25.2 | Rectangle | 541 |
| 9.25.3 | Line | 542 |
| 10 | Wichtige WPF-Techniken | 543 |
| 10.1 | Eigenschaften | 543 |
| 10.1.1 | Abhängige Eigenschaften (Dependency Properties) | 543 |
| 10.1.2 | Angehängte Eigenschaften (Attached Properties) | 545 |
| 10.2 | Einsatz von Ressourcen | 545 |
| 10.2.1 | Was sind eigentlich Ressourcen? | 545 |
| 10.2.2 | Wo können Ressourcen gespeichert werden? | 546 |
| 10.2.3 | Wie definiere ich eine Ressource? | 547 |
| 10.2.4 | Statische und dynamische Ressourcen | 548 |

| | | |
|-----------|---|------------|
| 10.2.5 | Wie werden Ressourcen adressiert? | 550 |
| 10.2.6 | Systemressourcen einbinden | 550 |
| 10.3 | Das WPF-Ereignismodell | 551 |
| 10.3.1 | Einführung | 551 |
| 10.3.2 | Routed Events | 552 |
| 10.3.3 | Direkte Events | 554 |
| 10.4 | Verwendung von Commands | 554 |
| 10.4.1 | Einführung zu Commands | 555 |
| 10.4.2 | Verwendung vordefinierter Commands | 555 |
| 10.4.3 | Das Ziel des Commands | 557 |
| 10.4.4 | Vordefinierte Commands | 558 |
| 10.4.5 | Commands an Ereignismethoden binden | 558 |
| 10.4.6 | Wie kann ich ein Command per Code auslösen? | 560 |
| 10.4.7 | Command-Ausführung verhindern | 561 |
| 10.5 | Das WPF-Style-System | 561 |
| 10.5.1 | Übersicht | 561 |
| 10.5.2 | Benannte Styles | 562 |
| 10.5.3 | Typ-Styles | 564 |
| 10.5.4 | Styles anpassen und vererben | 565 |
| 10.6 | Verwenden von Triggern | 568 |
| 10.6.1 | Eigenschaften-Trigger (Property Triggers) | 568 |
| 10.6.2 | Ereignis-Trigger | 570 |
| 10.6.3 | Daten-Trigger | 571 |
| 10.7 | Einsatz von Templates | 572 |
| 10.7.1 | Neues Template erstellen | 572 |
| 10.7.2 | Template abrufen und verändern | 576 |
| 10.8 | Transformationen, Animationen, StoryBoards | 579 |
| 10.8.1 | Transformationen | 579 |
| 10.8.2 | Animationen mit dem StoryBoard realisieren | 584 |
| 11 | WPF-Datenbindung | 589 |
| 11.1 | Grundprinzip | 589 |
| 11.1.1 | Bindungsarten | 590 |
| 11.1.2 | Wann eigentlich wird die Quelle aktualisiert? | 592 |
| 11.1.3 | Geht es auch etwas langsamer? | 593 |
| 11.1.4 | Bindung zur Laufzeit realisieren | 594 |
| 11.2 | Binden an Objekte | 595 |
| 11.2.1 | Objekte im XAML-Code instanziiieren | 596 |
| 11.2.2 | Verwenden der Instanz im C#-Quellcode | 597 |
| 11.2.3 | Anforderungen an die Quell-Klasse | 598 |
| 11.2.4 | Instanziiieren von Objekten per C#-Code | 599 |
| 11.3 | Binden von Collections | 601 |
| 11.3.1 | Anforderung an die Collection | 601 |
| 11.3.2 | Einfache Anzeige | 602 |

| | | |
|-------------------------------------|---|------------|
| 11.3.3 | Navigieren zwischen den Objekten | 603 |
| 11.3.4 | Einfache Anzeige in einer ListBox | 605 |
| 11.3.5 | DataTemplates zur Anzeigeformatierung | 606 |
| 11.3.6 | Mehr zu List- und ComboBox | 607 |
| 11.3.7 | Verwendung der ListView | 609 |
| 11.4 | Noch einmal zurück zu den Details | 612 |
| 11.4.1 | Navigieren in den Daten | 612 |
| 11.4.2 | Sortieren | 614 |
| 11.4.3 | Filtern | 614 |
| 11.4.4 | Live Shaping | 615 |
| 11.5 | Anzeige von Datenbankinhalten | 617 |
| 11.5.1 | Datenmodell per EF-Designer erzeugen | 617 |
| 11.5.2 | Die Programmoberfläche | 621 |
| 11.5.3 | Der Zugriff auf die Daten | 622 |
| 11.6 | Formatieren von Werten | 624 |
| 11.6.1 | IValueConverter | 625 |
| 11.6.2 | BindingBase.StringFormat-Eigenschaft | 627 |
| 11.7 | Das DataGrid als Universalwerkzeug | 627 |
| 11.7.1 | Grundlagen der Anzeige | 627 |
| 11.7.2 | UI-Virtualisierung | 629 |
| 11.7.3 | Spalten selbst definieren | 629 |
| 11.7.4 | Zusatzinformationen in den Zeilen anzeigen | 631 |
| 11.7.5 | Vom Betrachten zum Editieren | 632 |
| 11.8 | Praxisbeispiel – Collections in Hintergrundthreads füllen | 633 |
| Teil III: Technologien | | 637 |
| 12 | Zugriff auf das Dateisystem | 639 |
| 12.1 | Grundlagen | 639 |
| 12.1.1 | Klassen für den Zugriff auf das Dateisystem | 640 |
| 12.1.2 | Statische versus Instanzen-Klasse | 640 |
| 12.2 | Übersichten | 641 |
| 12.2.1 | Methoden der Directory-Klasse | 642 |
| 12.2.2 | Methoden eines DirectoryInfo-Objekts | 642 |
| 12.2.3 | Eigenschaften eines DirectoryInfo-Objekts | 642 |
| 12.2.4 | Methoden der File-Klasse | 643 |
| 12.2.5 | Methoden eines FileInfo-Objekts | 644 |
| 12.2.6 | Eigenschaften eines FileInfo-Objekts | 644 |
| 12.3 | Operationen auf Verzeichnisebene | 645 |
| 12.3.1 | Existenz eines Verzeichnisses/einer Datei feststellen | 645 |
| 12.3.2 | Verzeichnisse erzeugen und löschen | 646 |
| 12.3.3 | Verzeichnisse verschieben und umbenennen | 646 |
| 12.3.4 | Aktuelles Verzeichnis bestimmen | 647 |
| 12.3.5 | Unterverzeichnisse ermitteln | 647 |

| | | |
|-----------|---|------------|
| 12.3.6 | Alle Laufwerke ermitteln | 648 |
| 12.3.7 | Dateien kopieren und verschieben | 649 |
| 12.3.8 | Dateien umbenennen | 649 |
| 12.3.9 | Dateiattribute feststellen | 650 |
| 12.3.10 | Verzeichnis einer Datei ermitteln | 651 |
| 12.3.11 | Alle im Verzeichnis enthaltenen Dateien ermitteln | 652 |
| 12.3.12 | Dateien und Unterverzeichnisse ermitteln | 652 |
| 12.4 | Weitere wichtige Klassen | 653 |
| 12.4.1 | Die Path-Klasse | 653 |
| 12.4.2 | Die Klasse FileSystemWatcher | 654 |
| 12.4.3 | Die Klasse ZipArchive | 656 |
| 12.5 | Datei- und Verzeichnisdialoge | 657 |
| 12.5.1 | OpenFileDialog | 658 |
| 12.5.2 | SaveFileDialog | 660 |
| 12.6 | Praxisbeispiele | 661 |
| 12.6.1 | Infos über Verzeichnisse und Dateien gewinnen | 661 |
| 12.6.2 | Eine Verzeichnisstruktur in die TreeView einlesen | 666 |
| 13 | Dateien lesen und schreiben | 671 |
| 13.1 | Grundprinzip der Datenpersistenz | 671 |
| 13.1.1 | Dateien und Streams | 671 |
| 13.1.2 | Die wichtigsten Klassen | 672 |
| 13.1.3 | Erzeugen eines Streams | 673 |
| 13.2 | Dateiparameter | 673 |
| 13.2.1 | FileAccess | 673 |
| 13.2.2 | FileMode | 674 |
| 13.2.3 | FileShare | 674 |
| 13.3 | Textdateien | 675 |
| 13.3.1 | Eine Textdatei beschreiben bzw. neu anlegen | 675 |
| 13.3.2 | Eine Textdatei lesen | 676 |
| 13.4 | Binärdateien | 678 |
| 13.4.1 | Lese-/Schreibzugriff | 678 |
| 13.4.2 | Die Methoden ReadAllBytes und WriteAllBytes | 679 |
| 13.4.3 | Erzeugen von BinaryReader/BinaryWriter | 679 |
| 13.5 | Sequenzielle Dateien | 680 |
| 13.5.1 | Lesen und Schreiben von strukturierten Daten | 680 |
| 13.5.2 | Serialisieren von Objekten | 681 |
| 13.6 | Dateien verschlüsseln | 682 |
| 13.6.1 | Das Methodenpärchen Encrypt/Decrypt | 683 |
| 13.6.2 | Verschlüsseln mit der CryptoStream-Klasse | 683 |
| 13.7 | Praxisbeispiele | 684 |
| 13.7.1 | Auf eine Textdatei zugreifen | 684 |
| 13.7.2 | Einen Objektbaum persistent speichern | 688 |

| | | |
|-----------|---|------------|
| 13.7.3 | Eine Datei verschlüsseln | 693 |
| 13.7.4 | PDFs erstellen/exportieren | 698 |
| 13.7.5 | Eine CSV-Datei erstellen | 702 |
| 13.7.6 | Eine CSV-Datei mit LINQ lesen und auswerten | 703 |
| 13.7.7 | Einen korrekten Dateinamen erzeugen | 706 |
| 14 | Asynchrone Programmierung | 707 |
| 14.1 | Übersicht | 708 |
| 14.1.1 | Multitasking versus Multithreading | 708 |
| 14.1.2 | Deadlocks | 709 |
| 14.1.3 | Racing | 710 |
| 14.2 | Programmieren mit Threads | 711 |
| 14.2.1 | Einführungsbeispiel | 711 |
| 14.2.2 | Wichtige Thread-Methoden | 713 |
| 14.2.3 | Wichtige Thread-Eigenschaften | 715 |
| 14.2.4 | Einsatz der ThreadPool-Klasse | 716 |
| 14.3 | Sperrmechanismen | 717 |
| 14.3.1 | Threading ohne lock | 718 |
| 14.3.2 | Threading mit lock | 719 |
| 14.3.3 | Die Monitor-Klasse | 722 |
| 14.3.4 | Mutex | 726 |
| 14.3.5 | Methoden für die parallele Ausführung sperren | 727 |
| 14.3.6 | Semaphore | 728 |
| 14.4 | Interaktion mit der Programmoberfläche | 730 |
| 14.4.1 | Die Werkzeuge | 731 |
| 14.4.2 | Einzelne Steuerelemente mit Invoke aktualisieren | 731 |
| 14.4.3 | Mehrere Steuerelemente aktualisieren | 732 |
| 14.4.4 | Ist ein Invoke-Aufruf nötig? | 733 |
| 14.4.5 | Und was ist mit WPF? | 733 |
| 14.5 | Timer-Threads | 735 |
| 14.6 | Asynchrone Programmierentwurfsmuster | 736 |
| 14.6.1 | Kurzübersicht | 736 |
| 14.6.2 | Polling | 738 |
| 14.6.3 | Callback verwenden | 739 |
| 14.6.4 | Callback mit Parameterübergabe verwenden | 740 |
| 14.6.5 | Callback mit Zugriff auf die Programmoberfläche | 741 |
| 14.7 | Asynchroner Aufruf beliebiger Methoden | 743 |
| 14.7.1 | Die Beispielklasse | 743 |
| 14.7.2 | Asynchroner Aufruf ohne Callback | 744 |
| 14.7.3 | Asynchroner Aufruf mit Callback und Anzeigefunktion | 745 |
| 14.7.4 | Aufruf mit Rückgabewerten (per Eigenschaft) | 746 |
| 14.7.5 | Aufruf mit Rückgabewerten (per EndInvoke) | 747 |
| 14.8 | Es geht auch einfacher - async und await | 748 |
| 14.8.1 | Der Weg von synchron zu asynchron | 748 |

| | | |
|-----------|--|------------|
| 14.8.2 | Achtung: Fehlerquellen! | 751 |
| 14.8.3 | Eigene asynchrone Methoden entwickeln | 753 |
| 14.9 | Praxisbeispiele | 755 |
| 14.9.1 | Prozess- und Thread-Informationen gewinnen | 755 |
| 14.9.2 | Ein externes Programm starten | 758 |
| 15 | Die Task Parallel Library | 761 |
| 15.1 | Überblick | 761 |
| 15.1.1 | Parallel-Programmierung | 761 |
| 15.1.2 | Möglichkeiten der TPL | 764 |
| 15.1.3 | Der CLR-Threadpool | 764 |
| 15.2 | Parallele Verarbeitung mit Parallel.Invoke | 765 |
| 15.2.1 | Aufrufvarianten | 766 |
| 15.2.2 | Einschränkungen | 767 |
| 15.3 | Verwendung von Parallel.For | 767 |
| 15.3.1 | Abbrechen der Verarbeitung | 769 |
| 15.3.2 | Auswerten des Verarbeitungsstatus | 770 |
| 15.3.3 | Und was ist mit anderen Iterator-Schrittweiten? | 771 |
| 15.4 | Collections mit Parallel.ForEach verarbeiten | 772 |
| 15.5 | Die Task-Klasse | 773 |
| 15.5.1 | Einen Task erzeugen | 773 |
| 15.5.2 | Den Task starten | 774 |
| 15.5.3 | Datenübergabe an den Task | 775 |
| 15.5.4 | Wie warte ich auf das Ende des Tasks? | 777 |
| 15.5.5 | Tasks mit Rückgabewerten | 778 |
| 15.5.6 | Die Verarbeitung abbrechen | 781 |
| 15.5.7 | Fehlerbehandlung | 785 |
| 15.5.8 | Weitere Eigenschaften | 786 |
| 15.6 | Zugriff auf das User Interface | 788 |
| 15.6.1 | Task-Ende und Zugriff auf die Oberfläche | 788 |
| 15.6.2 | Zugriff auf das UI aus dem Task heraus | 789 |
| 15.7 | Weitere Datenstrukturen im Überblick | 791 |
| 15.7.1 | Threadsichere Collections | 791 |
| 15.7.2 | Primitive für die Threadsynchroisation | 792 |
| 15.8 | Parallel LINQ (PLINQ) | 792 |
| 15.9 | Praxisbeispiele | 792 |
| 15.9.1 | BlockingCollection | 792 |
| 15.9.2 | PLINQ | 796 |
| 16 | Debugging, Fehlersuche und Fehlerbehandlung | 797 |
| 16.1 | Der Debugger | 797 |
| 16.1.1 | Allgemeine Beschreibung | 797 |
| 16.1.2 | Die wichtigsten Fenster | 798 |

| | | |
|-----------|---|------------|
| 16.1.3 | Debugging-Optionen | 802 |
| 16.1.4 | Praktisches Debugging am Beispiel | 804 |
| 16.2 | Arbeiten mit Debug und Trace | 808 |
| 16.2.1 | Wichtige Methoden von Debug und Trace | 808 |
| 16.2.2 | Besonderheiten der Trace-Klasse | 812 |
| 16.2.3 | TraceListener-Objekte | 813 |
| 16.3 | Caller Information | 816 |
| 16.3.1 | Attribute | 816 |
| 16.3.2 | Anwendung | 816 |
| 16.4 | Fehlerbehandlung | 817 |
| 16.4.1 | Anweisungen zur Fehlerbehandlung | 817 |
| 16.4.2 | try-catch | 818 |
| 16.4.3 | try-finally | 822 |
| 16.4.4 | Das Standardverhalten bei Ausnahmen festlegen | 825 |
| 16.4.5 | Die Exception-Klasse | 826 |
| 16.4.6 | Fehler/Ausnahmen auslösen | 827 |
| 16.4.7 | Eigene Fehlerklassen | 827 |
| 16.4.8 | Exceptionhandling zur Debugzeit | 829 |
| 16.4.9 | Code Contracts | 829 |
| 17 | JSON und XML in Theorie und Praxis | 831 |
| 17.1 | JSON – JavaScriptObjectNotation | 831 |
| 17.1.1 | Grundlagen | 831 |
| 17.1.2 | De-/Serialisierung mit JSON | 832 |
| 17.2 | XML – etwas Theorie | 835 |
| 17.2.1 | Übersicht | 836 |
| 17.2.2 | Der XML-Grundaufbau | 837 |
| 17.2.3 | Wohlgeformte Dokumente | 838 |
| 17.2.4 | Processing Instructions (PI) | 840 |
| 17.2.5 | Elemente und Attribute | 841 |
| 17.3 | XSD-Schemas | 842 |
| 17.3.1 | XML-Schemas in Visual Studio analysieren | 843 |
| 17.3.2 | XML-Datei mit XSD-Schema erzeugen | 846 |
| 17.3.3 | XSD-Schema aus einer XML-Datei erzeugen | 847 |
| 17.4 | Verwendung des DOM unter .NET | 848 |
| 17.4.1 | Übersicht | 848 |
| 17.4.2 | DOM-Integration in C# | 849 |
| 17.4.3 | Laden von Dokumenten | 850 |
| 17.4.4 | Erzeugen von XML-Dokumenten | 851 |
| 17.4.5 | Auslesen von XML-Dateien | 853 |
| 17.4.6 | Direktzugriff auf einzelne Elemente | 854 |
| 17.4.7 | Einfügen von Informationen | 855 |
| 17.4.8 | Suchen in den Baumzweigen | 858 |

| | | |
|-----------|---|------------|
| 17.5 | XML-Verarbeitung mit LINQ to XML | 861 |
| 17.5.1 | Die LINQ to XML-API | 861 |
| 17.5.2 | Neue XML-Dokumente erzeugen | 863 |
| 17.5.3 | Laden und Sichern von XML-Dokumenten | 864 |
| 17.5.4 | Navigieren in XML-Daten | 866 |
| 17.5.5 | Auswählen und Filtern | 868 |
| 17.5.6 | Manipulieren der XML-Daten | 869 |
| 17.5.7 | XML-Dokumente transformieren | 870 |
| 17.6 | Weitere Möglichkeiten der XML-Verarbeitung | 874 |
| 17.6.1 | Schnelles Suchen in XML-Daten mit XPathNavigator | 874 |
| 17.6.2 | Schnelles Auslesen von XML-Daten mit XmlReader | 876 |
| 17.6.3 | Erzeugen von XML-Daten mit XmlWriter | 878 |
| 17.6.4 | XML transformieren mit XSLT | 880 |
| 17.7 | Praxisbeispiele | 882 |
| 17.7.1 | Mit dem DOM in XML-Dokumenten navigieren | 882 |
| 17.7.2 | XML-Daten in eine TreeView einlesen | 887 |
| 17.7.3 | In Dokumenten mit dem XPathNavigator navigieren | 891 |
| 18 | Einführung in ADO.NET und Entity Framework | 899 |
| 18.1 | Eine kleine Übersicht | 899 |
| 18.1.1 | Die ADO.NET-Klassenhierarchie | 899 |
| 18.1.2 | Die Klassen der Datenprovider | 900 |
| 18.1.3 | Das Zusammenspiel der ADO.NET-Klassen | 903 |
| 18.2 | Das Connection-Objekt | 904 |
| 18.2.1 | Allgemeiner Aufbau | 904 |
| 18.2.2 | SqlConnection | 904 |
| 18.2.3 | Schließen einer Verbindung | 905 |
| 18.2.4 | Eigenschaften des Connection-Objekts | 906 |
| 18.2.5 | Methoden des Connection-Objekts | 908 |
| 18.2.6 | Der SqlConnectionStringBuilder | 909 |
| 18.2.7 | ConnectionString in der Konfigurationsdatei | 910 |
| 18.3 | Das Command-Objekt | 911 |
| 18.3.1 | Erzeugen und Anwenden eines Command-Objekts | 911 |
| 18.3.2 | Erzeugen mittels CreateCommand-Methode | 912 |
| 18.3.3 | Eigenschaften des Command-Objekts | 912 |
| 18.3.4 | Methoden des Command-Objekts | 915 |
| 18.3.5 | Freigabe von Connection- und Command-Objekten | 917 |
| 18.4 | Parameter-Objekte | 919 |
| 18.4.1 | Erzeugen und Anwenden eines Parameter-Objekts | 919 |
| 18.4.2 | Eigenschaften des Parameter-Objekts | 920 |
| 18.5 | Das SqlCommandBuilder-Objekt | 921 |
| 18.5.1 | Erzeugen | 921 |
| 18.5.2 | Anwenden | 922 |

| | | |
|---|---|------------|
| 18.6 | Das DataReader-Objekt | 922 |
| 18.6.1 | DataReader erzeugen | 923 |
| 18.6.2 | Daten lesen | 923 |
| 18.6.3 | Eigenschaften des DataReaders | 924 |
| 18.6.4 | Methoden des DataReaders | 925 |
| 18.7 | Das DataAdapter-Objekt | 925 |
| 18.7.1 | DataAdapter erzeugen | 926 |
| 18.7.2 | Command-Eigenschaften | 927 |
| 18.7.3 | Fill-Methode | 928 |
| 18.7.4 | Update-Methode | 929 |
| 18.7.5 | DataSet | 930 |
| 18.8 | Entity Framework | 933 |
| 18.8.1 | Überblick | 933 |
| 18.8.2 | CodeFirst | 934 |
| 18.8.3 | CodeFirst aus Datenbank | 940 |
| 18.9 | Praxisbeispiele | 948 |
| 18.9.1 | Wichtige ADO.NET-Objekte im Einsatz | 948 |
| 18.9.2 | Eine Aktionsabfrage ausführen | 951 |
| 18.9.3 | Eine StoredProcedure aufrufen | 955 |
| 18.9.4 | Daten mit Entity Framework laden und als JSON speichern | 958 |
| 19 | Weitere Techniken | 969 |
| 19.1 | Zugriff auf die Zwischenablage | 969 |
| 19.1.1 | Das Clipboard-Objekt | 969 |
| 19.1.2 | Zwischenablage-Funktionen für Textboxen | 971 |
| 19.2 | .NET-Reflection | 971 |
| 19.2.1 | Übersicht | 972 |
| 19.2.2 | Assembly laden | 972 |
| 19.2.3 | Mittels GetType und Type Informationen sammeln | 973 |
| 19.2.4 | Dynamisches Laden von Assemblies | 975 |
| 19.3 | Praxisbeispiele | 977 |
| 19.3.1 | Nutzer und Gruppen des aktuellen Systems ermitteln | 977 |
| 19.3.2 | Testen, ob Nutzer in einer Gruppe enthalten ist | 980 |
| 19.3.3 | Die IP-Adressen des Computers ermitteln | 981 |
| 19.3.4 | Diverse Systeminformationen ermitteln | 983 |
| Anhang A: Glossar | 991 | |
| Anhang B: Wichtige Dateiextensions | 995 | |
| Index | 997 | |

Vorwort

C# ist die momentan von Microsoft propagierte Sprache, sie bietet die Möglichkeiten und Flexibilität von C++ und erlaubt trotzdem eine schnelle und unkomplizierte Programmierpraxis wie einst bei Visual Basic. C# ist (fast) genauso mächtig wie C++, wurde aber komplett neu auf objektorientierter Basis geschrieben.

Damit ist C# das ideale Werkzeug zum Programmieren beliebiger Komponenten für das Microsoft .NET Framework, beginnend bei Windows Forms- über WPF-, ASP.NET- und mobilen Anwendungen (mittlerweile auch für Android und iOS) bis hin zu systemnahen Applikationen.

Das vorliegende Buch ist ein Angebot für künftige wie auch für fortgeschrittene C#-Programmierer. Seine Philosophie knüpft an die vielen anderen Titel an, die in dieser Reihe in den vergangenen zwanzig Jahren zu verschiedenen Programmiersprachen erschienen sind:

- Programmieren lernt man nicht durch lineares Durcharbeiten eines Lehrbuchs, sondern nur durch unermüdliches Ausprobieren von Beispielen, verbunden mit ständigem Nachschlagen in der Referenz.
- Der Umfang einer modernen Sprache wie C# in Verbindung mit Visual Studio ist so gewaltig, dass ein seriöses Programmierbuch das Prinzip der Vollständigkeit aufgeben muss und nach dem Prinzip „so viel wie nötig“ sich lediglich eine „Initialisierungsfunktion“ auf die Fahnen schreiben kann.

Gegenüber anderen Büchern zur gleichen oder ähnlichen Thematik nimmt dieses Buch für sich in Anspruch, gleichzeitig Lehr- und Übungsbuch zu sein.

Zum Buchinhalt

Wie Sie bereits dem Buchtitel entnehmen können, wagt das vorliegende Werk den Spagat zwischen einem Grundlagen- und einem Profibuch. Sinn eines solchen Buches kann es nicht sein, eine umfassende Schritt-für-Schritt-Einführung in C# 7.3 zu liefern oder all die Informationen noch einmal zur Verfügung zu stellen, die Sie in der Produktdokumentation (MSDN) ohnehin schon finden und von denen Sie in der Regel nur ein Mausklick oder die F1-Taste trennt.

- Für den *Einsteiger* möchte ich den einzig vernünftigen und gangbaren Weg beschreiten, nämlich nach dem Prinzip „so viel wie nötig“ eine schmale Schneise durch den Urwald der .NET-Programmierung mit C# schlagen, bis er eine Lichtung erreicht hat, die ihm erste Erfolgserlebnisse vermittelt.

- Für den *Profi* möchte ich in diesem Buch eine Vielzahl von Informationen und Know-how bereitstellen, wonach er bisher in den mitgelieferten Dokumentationen, im Internet bzw. in anderen Büchern vergeblich gesucht hat.

Die Kapitel des Buches und die Onlinekapitel sind in fünf Themenkomplexe gruppiert:

1. Grundlagen der Programmierung mit C# (Buch)
2. WPF-Anwendungen (Buch)
3. Technologien (Buch)
4. Windows Forms-Anwendungen (online)
5. Bonuskapitel (online)

Die Kapitel innerhalb eines Teils bilden einerseits eine logische Aufeinanderfolge, können andererseits aber auch quergelesen werden. Im Praxisteil eines jeden Kapitels werden anhand realer Problemstellungen die behandelten Programmiertechniken im Zusammenhang demonstriert.

Zu den Codebeispielen

Auf vielfachen Wunsch habe ich mich dazu entschlossen, ab dieser Auflage des Buches die Beispiele auch mit WPF zu erstellen. Der Einfachheit halber werden im ersten Teil die Beispiele weiter mit Windows Forms erstellt. Der zweite Teil des Buches behandelt dann ausführlich WPF und im dritten Teil „Technologien“ werden die Beispiele dann mit WPF dargestellt. Alle Beispieldaten dieses Buches und das Bonuskapitel zu Windows Forms können Sie unter der folgenden Adresse herunterladen:

www.hanser-fachbuch.de

Geben Sie bitte im Suchfeld „Visual C# 2019“ ein und klicken Sie auf der Buchdetailseite auf die Registerkarte *Extras*.

Beim Nachvollziehen der Buchbeispiele beachten Sie bitte Folgendes:

- Kopieren Sie die Buchbeispiele auf die Festplatte. Wenn Sie auf die Projektmappendatei (*.sln) klicken, wird Visual Studio in der Regel automatisch geöffnet und das jeweilige Beispiel wird in die Entwicklungsumgebung geladen, wo Sie es z. B. mittels **F5**-Taste kompilieren und starten können.
- Für einige Beispiele ist ein installierter Microsoft SQL Server Express LocalDB oder jegliche andere Instanz eines SQL-Servers erforderlich.
- Beachten Sie die zu einigen Beispielen beigelegten *Liesmich.txt*-Dateien, die Sie auf besondere Probleme hinweisen.

Nobody is perfect

Sie werden in diesem Buch nicht alles finden, was C# bzw. das .NET Framework 4.7.2 zu bieten haben. Manches ist sicher in einem anderen Spezialtitel noch besser oder ausführlicher beschrieben. Aber Sie halten mit diesem Buch einen überschaubaren Breitband-Mix in den Händen, der sowohl vertikal vom Einsteiger bis zum Profi als auch horizontal von den einfachen Sprachelementen bis hin zu komplexen Anwendungen jedem etwas bietet, ohne dabei den Blick auf das Wesentliche im .NET-Dschungel zu verlieren.

Wenn Sie Vorschläge oder Fragen zum Buch haben, können Sie mich gerne kontaktieren:

juergen.kotz@primetime-software.de

Ich hoffe, dass ich Ihnen mit diesem Buch einen nützlichen Begleiter bei der .NET-Programmierung zur Seite gestellt habe, der es verdient, seinen Platz nicht im Regal, sondern griffbereit neben dem Computer einzunehmen.

Jürgen Kotz

München, im Sommer 2019

Teil I: Grundlagen



- Einstieg in Visual Studio 2019 (Kapitel 1)
- Grundlagen der Sprache C# (Kapitel 2)
- Objektorientiertes Programmieren (Kapitel 3)
- Arrays, Strings und Funktionen (Kapitel 4)
- Weitere wichtige Sprachfeatures (Kapitel 5)
- Einführung in LINQ (Kapitel 6)
- Neuerungen von C# im Überblick (Kapitel 7)

1

Einstieg in Visual Studio 2019

Dieses Kapitel bietet Ihnen einen effektiven Schnelleinstieg in die Arbeit mit Visual Studio 2019. Gleich nachdem Sie die Hürden der Installation gemeistert haben, erstellen Sie mit C# Ihre ersten .NET-Anwendungen, werden dabei en passant mit den grundlegenden Features der Entwicklungsumgebung vertraut gemacht und nach dem Prinzip „so viel wie nötig“ in die .NET-Philosophie eingeweiht. Nach der Lektüre dieses Kapitels und dem Nachvollzug der abschließenden Praxisbeispiele sollte der Einsteiger über eine brauchbare Ausgangsbasis verfügen, um den sich vor ihm gewaltig auftürmenden Berg von Spezialkapiteln in Angriff zu nehmen.

■ 1.1 Die Installation von Visual Studio 2019

Ohne eine angemessen ausgestattete „Werkstatt“ ist die Lektüre dieses Buches nutzlos. Programmieren lernt man nur durch Beispiele, die man unmittelbar selbst am Rechner ausprobiert!



HINWEIS: Voraussetzung für ein erfolgreiches Studium dieses Buches ist ein Rechner mit einer lauffähigen Installation von Visual Studio 2019!

1.1.1 Überblick über die Produktpalette

Alle im Handel angebotenen Visual-Studio-Pakete basieren auf dem .NET Framework 4.7.2. Für welches der im Folgenden aufgeführten Produkte man sich entscheidet, hängt von den eigenen Anforderungen und Wünschen ab und ist nicht zuletzt auch eine Frage des Geldbeutels.