

Dr. Holger Schwichtenberg

Moderne Datenzugriffslösungen mit Entity Framework Core 3.0

Datenbankprogrammierung mit C# in .NET Core



Version 7.0.3 dieses Buchs

Stand 29.08.2019

Verlag: www.IT-Visions.de, Fahrenberg 40b, D-45257 Essen
ISBN Druckausgabe: 3934279-21-X
ISBN Kindleausgabe: 3934279-22-8
ISBN PDF-Ausgabe: 3934279-23-6
Sprachliche Korrektur: Matthias Bloch, M.A.
Formatierung: Matthias Bloch, M.A.
Bezugsquelle Gedruckt: www.amazon.de/dp/393427921X
Bezugsquelle Kindle: www.amazon.de/dp/B07WPX61SZ
Bezugsquelle PDF: www.leanpub.com/EntityFrameworkCore3



Für Heidi, Felix und Maja

1 Inhaltsverzeichnis

1	Inhaltsverzeichnis	4
2	Vorwort	18
3	Über den Autor	20
4	Über dieses Buch	21
4.1	Versionsgeschichte dieses Buchs	21
4.2	Bezugsquelle für Aktualisierungen	28
4.3	Geplante Kapitel	28
4.4	Programmiersprache in diesem Buch	29
5	Fallbeispiele in diesem Buch	30
5.1	Entitäten	30
5.2	Englische Version des Beispiels	34
5.3	Anwendungsarten in diesem Buch	34
5.4	Hilfsroutinen zur Konsolenausgabe	35
6	Programmcodebeispiel zum Download	41
6.1	Website zum Download	41
6.2	Übersicht über die Beispiele	41
6.3	Technischer Hinweis zu den Beispielen	43
6.4	Qualitätssicherung der Beispiele	43
7	Was ist Entity Framework Core?	45
7.1	Was ist ein Objekt-Relationaler Mapper (ORM)?	45
7.2	ORM in der .NET-Welt	46
7.3	Versionsgeschichte von Entity Framework Core	47
7.4	Unterstützte Betriebssysteme	50
7.5	Unterstützte .NET-Versionen	50
7.6	Unterstützte Visual Studio-Versionen	52
7.7	Unterstützte Datenbanken	52
7.8	Funktionsumfang von Entity Framework Core	54
7.9	Funktionen, die dauerhaft entfallen	55
7.10	Funktionen, die Microsoft noch nachrüsten will	56
7.11	Neue Funktionen in Entity Framework Core	58
7.12	Neuerungen in Entity Framework Core 3.x	59
7.13	Kommende Versionen	59

7.14	Kritik an der Entwicklungsgeschwindigkeit von Entity Framework Core.....	60
7.15	Einsatzszenarien für Entity Framework Core	60
7.16	Migration von ADO.NET Entity Framework zu Entity Framework Core.....	61
8	Installation von Entity Framework Core	63
8.1	NuGet-Pakete.....	63
8.2	Paketinstallation.....	64
8.3	Aktualisierung auf eine neue Version.....	69
9	Konzepte von Entity Framework Core.....	73
9.1	Vorgehensmodelle bei Entity Framework Core	73
9.2	Artefakte bei Entity Framework Core	76
10	Reverse Engineering bestehender Datenbanken	78
10.1	Reverse Engineering-Werkzeuge	78
10.2	Vorbereiten des Reverse Engineering mit PowerShell-Befehlen	78
10.3	Codegenerierung	80
10.4	Generierter Programmcode.....	84
10.5	Beispiel-Client	90
10.6	.NET Core-Tool	91
10.7	Schwächen des Reverse Engineering	92
11	Forward Engineering für neue Datenbanken	94
11.1	Zwei Klassentypen beim Forward Engineering	94
11.2	Beispiele in diesem Kapitel	94
11.3	Regeln für die selbsterstellten Entitätsklassen	95
11.3.1	NuGet-Pakete	95
11.3.2	Properties	96
11.3.3	Datentypen	96
11.3.4	Beziehungen (Master-Detail)	96
11.3.5	Vererbung	99
11.3.6	Primärschlüssel.....	99
11.3.7	Beispiele.....	99
11.4	Regeln für die selbsterstellte Kontextklasse.....	102
11.4.1	NuGet-Pakete	102
11.4.2	Basisklasse	103
11.4.3	Konstruktor	103

11.4.4	Verweise zu den Entitätsklassen.....	103
11.4.5	Provider und Verbindungszeichenfolge	103
11.4.6	Beispiel für eine selbsterstellte Kontextklasse.....	105
11.4.7	Verwendung von DbContextOptions.....	105
11.4.8	Eigenes Datenbankverbindungsmanagement.....	107
11.4.9	Flexible Konfiguration der Kontextklasse.....	107
11.4.10	Thread-Sicherheit	107
11.5	Regeln für die Datenbankschemagenerierung.....	107
11.6	Beispiel-Client	108
11.7	Anpassung per Fluent-API (OnModelCreating()).....	109
11.8	Das erzeugte Datenmodell	111
12	Anpassung des Datenbankschemas.....	114
12.1	Beispiele in diesem Kapitel	114
12.2	Konvention versus Konfiguration	114
12.3	Persistente versus transiente Klassen.....	115
12.4	Namen im Datenbankschema.....	116
12.4.1	Änderung der Tabellen- und Spaltennamen	116
12.4.2	Breaking Change in EF Core 3.0.....	117
12.5	Reihenfolge der Spalten in einer Tabelle.....	117
12.6	Spaltentypen/Datentypen.....	118
12.7	Typkonvertierungen	119
12.8	Pflichtfelder und optionale Felder	120
12.9	Feldlängen	120
12.10	Primärschlüssel	120
12.11	Beziehungen und Fremdschlüssel	121
12.12	Optionale Beziehungen und Pflichtbeziehungen.....	122
12.13	Uni- und Bidirektionale Beziehungen	124
12.14	1:1-Beziehungen	125
12.15	Indexe festlegen	126
12.16	Vererbung.....	127
12.16.1	Vererbungsstrategien	127
12.16.2	TPH und TPCT in Entity Framework Core	129
12.16.3	Mischung von TPCT und TPH	138

12.16.4	Konsistenzprobleme bei TPH	139
12.16.5	Erzwingen von TPH.....	140
12.16.6	Konfiguration der Diskriminatorspalte bei TPH.....	140
12.17	Syntaxoptionen für das Fluent-API	142
12.17.1	Sequentielle Konfiguration.....	142
12.17.2	Strukturierung durch Statement Lambdas	142
12.17.3	Strukturierung durch Unterroutine.....	143
12.17.4	Strukturierung durch Konfigurationsklassen	144
12.18	Massenkonfiguration mit dem Fluent-API.....	145
13	Datenbankschemamigrationen.....	146
13.1	Anlegen der Datenbank zur Laufzeit.....	146
13.2	Schemamigrationen zur Entwicklungszeit.....	147
13.3	Befehle für die Schemamigrationen	147
13.4	ef.exe	148
13.5	Add-Migration	149
13.6	Update-Database	153
13.7	Schemamigrationen bei der Installation.....	155
13.8	Remove-Migration	155
13.9	Script-Migration.....	155
13.10	Schemamigrationen zur Laufzeit.....	156
13.10.1	Verwendung von Migrate()	156
13.10.2	IMigrator-Service	157
13.10.3	Informationen zum Migrationsstand	157
13.10.4	Praxiseinsatz: Ein Kommandozeilenwerkzeug für die Schemamigration	158
13.11	Schemamigrationsszenarien.....	162
13.11.1	Neue Tabellen und Spalten.....	162
13.11.2	Tabellen oder Spalten löschen	164
13.11.3	Tabellen oder Spalten umbenennen	164
13.11.4	Spaltendatentyp ändern	164
13.11.5	NULL-Werte verbieten	165
13.11.6	Kardinalitäten ändern.....	166
13.11.7	Andere Datenbankartefakte anlegen	167
13.11.8	SQL-Skriptdateien ausführen	169

13.11.9	Eigenständige Entitäten bilden	170
13.12	Weitere Möglichkeiten	173
13.13	Probleme bei der Schemamigration in Verbindung mit TFS	174
13.14	Startverhalten von Entity Framework Core	174
14	Daten lesen mit LINQ	175
14.1	Kontextklasse.....	175
14.2	LINQ-Abfragen.....	175
14.3	Schrittweises Zusammensetzung von LINQ-Abfragen	178
14.4	Einsatz von var.....	179
14.5	Repository-Pattern.....	179
14.6	LINQ-Abfragen mit Paging	183
14.7	Projektionen.....	184
14.7.1	Projektion auf einen Entitätstypen.....	185
14.7.2	Projektionen auf einen anonymen Typen	185
14.7.3	Projektionen auf einen beliebigen Typen.....	187
14.8	Abfrage nach Einzelobjekten.....	188
14.9	Laden anhand des Primärschlüssels mit Find()	189
14.10	Gruppierungen	190
14.11	Umgehung für das GroupBy-Problem.....	191
14.11.1	Mapping auf Nicht-Entitätstypen.....	192
14.11.2	Entitätsklasse für die Datenbanksicht anlegen.....	193
14.11.3	Einbinden der Entitätsklasse in die Kontextklasse	193
14.11.4	Verwendung der Pseudo-Entitätsklasse	194
14.11.5	Herausforderung: Migrationen	194
14.11.6	Gruppierungen mit Datenbanksichten.....	196
14.12	LINQ im RAM statt in der Datenbank (Client-Evaluation).....	196
14.13	Falsche Befehlsreihenfolge.....	198
14.14	Eigene Funktionen in LINQ.....	199
14.15	Kurzübersicht über die LINQ-Syntax.....	200
14.15.1	Einfache SELECT-Befehle (Alle Datensätze).....	201
14.15.2	Bedingungen (where).....	201
14.15.3	Bedingungen mit Mengen (in).....	202
14.15.4	Sortierungen (orderby).....	202

14.15.5	Paging (Skip() und Take()).....	202
14.15.6	Projektion.....	203
14.15.7	Aggregatfunktionen (Count(), Min(), Max(), Average(), Sum())	203
14.15.8	Gruppierungen (GroupBy)	204
14.15.9	Einzelobjekte (SingleOrDefault(), FirstOrDefault())	204
14.15.10	Verbundene Objekte (Include()).....	205
14.15.11	Inner Join (Join).....	206
14.15.12	Cross Join (Kartesisches Produkt)	206
14.15.13	Join mit Gruppierung	207
14.15.14	Unter-Abfragen (Sub-Select).....	208
14.16	Lokaler Objektzwischenspeicher in der Kontextklasse	209
14.16.1	Wirkung des Zwischenspeichers.....	209
14.16.2	Neuladen veralterter Objekte (Reload).....	212
14.16.3	Neuladen gelöschter Objekte.....	214
14.16.4	Ein typischer Fehler	215
14.16.5	Den Zwischenspeicher verwalten	217
15	Objektbeziehungen und Ladestrategien.....	218
15.1	Überblick über die Ladestrategien.....	218
15.2	Standardverhalten.....	218
15.3	Lazy Loading	220
15.3.1	Aktivierung des Lazy Loading	220
15.3.2	Gefahren von Lazy Loading	222
15.3.3	Lazy Loading ohne Proxyklassen	223
15.4	Explizites Nachladen (Explicit Loading)	226
15.5	Eager Loading	228
15.6	Relationship Fixup	231
15.6.1	Beispiel für Fall 1	232
15.6.2	Beispiel für Fall 2	233
15.6.3	Beispiel für Fall 3	234
15.7	Preloading mit Relationship Fixup	236
15.8	Objektbeziehungen und lokaler Zwischenspeicher	240
16	Einfügen, Löschen und Ändern	245
16.1	Speichern mit SaveChanges()	245

16.2	Änderungsverfolgung auch für Unterobjekte.....	247
16.3	Zusammenfassen von Befehlen (Batching).....	248
16.4	Das Foreach-Problem	249
16.5	Objekte hinzufügen mit Add()	251
16.6	Verbundene Objekte anlegen	253
16.7	Verbundene Objekte ändern / Relationship Fixup.....	256
16.8	Widersprüchliche Beziehungen.....	258
16.8.1	Objekte löschen mit Remove()	263
16.8.2	Löschen mit einem Attrappen-Objekt.....	265
16.8.3	Massenlöschen	266
16.9	Datenbanktransaktionen	267
16.9.1	Transaktion in einer Kontextinstanz	267
16.9.2	Transaktion über mehrere Kontextinstanzen ohne TransactionScope.....	268
16.9.3	Transaktion über mehrere Kontextinstanzen mit TransactionScope	270
16.10	Change Tracker abfragen.....	272
16.10.1	POCOs	272
16.10.2	Zustand eines Objekts	273
16.10.3	Liste aller geänderten Objekte	275
17	Datenänderungskonflikte (Concurrency).....	278
17.1	Rückblick.....	278
17.2	Im Standard keine Konflikterkennung.....	279
17.3	Optimistisches Sperren / Konflikterkennung	280
17.4	Konflikterkennung für alle Eigenschaften	281
17.5	Konflikteneinstellung per Konvention	282
17.6	Fallweise Konflikteneinstellung	283
17.7	Zeitstempel (Timestamp).....	283
17.8	Konflikte auflösen.....	285
17.9	Pessimistisches Sperren bei Entity Framework Core	289
18	Protokollierung (Logging).....	293
18.1	Verwendung der Erweiterungsmethode Log()	293
18.2	Implementierung der Log()-Erweiterungsmethode	295
18.3	Protokollierungskategorien.....	299
18.4	EnableSensitiveDataLogging.....	300

19	Asynchrone Programmierung.....	301
19.1	Asynchrone Erweiterungsmethoden.....	301
19.2	ToListAsync().....	301
19.3	SaveChangesAsync().....	302
19.4	ForEachAsync().....	303
20	Dynamische LINQ-Abfragen	305
20.1	Schrittweises zusammensetzen von LINQ-Abfragen	305
20.2	Expression Trees.....	306
20.3	Dynamic LINQ	309
21	Daten lesen und ändern mit SQL, Stored Procedures und Table Valued Functions.....	312
21.1	Abfragen mit FromSqlRaw() und FromSqlInterpolated().....	312
21.2	Zusammensetzbarkeit von LINQ und SQL.....	314
21.3	Stored Procedures und Table Valued Functions.....	316
21.4	Nicht-Entitätsklassen als Ergebnismenge	317
21.5	Erweiterungsmethode ExecuteSqlQuery().....	319
21.6	SQL-DML-Befehle ohne Resultset	319
22	Weitere Tipps und Tricks zum Mapping	321
22.1	Shadow Properties.....	321
22.1.1	Automatische Shadow Properties	321
22.1.2	Festlegung eines Shadow Property	322
22.1.3	Ausgabe aller Shadow Properties einer Entitätsklasse	322
22.1.4	Lesen und Ändern eines Shadow Property	323
22.1.5	LINQ-Abfragen mit Shadow Properties.....	324
22.1.6	Praxisbeispiel: Automatisches Setzen bei jedem Speichern	324
22.1.7	Praxisbeispiel: Erweitern der Tabellen zur Betriebszeit der Anwendung	325
22.2	Berechnete Spalten (Computed Columns).....	327
22.2.1	Automatisches SELECT	328
22.2.2	Praxistipp: Spalten mit einer Berechnungsformel anlegen.....	328
22.2.3	Spalten mit einer Berechnungsformel nutzen.....	330
22.2.4	Spalten mit einer Berechnungsformel beim Reverse Engineering.....	331
22.3	Standardwerte (Default Values)	331
22.3.1	Standardwerte beim Forward Engineering festlegen	332
22.3.2	Standardwerte verwenden	332

22.3.3	Praxistipp: Standardwerte schon beim Anlegen des Objekts vergeben.....	334
22.3.4	Standardwerte beim Reverse Engineering.....	335
22.4	Tabellenaufteilung (Table Splitting) mit Owned Types	335
22.4.1	Owned Types	336
22.4.2	Weitere Möglichkeiten mit Owned Types.....	343
22.4.3	Daten schreiben und lesen mit Owned Types.....	344
22.4.4	Einschränkungen bei Owned Types.....	346
22.5	Sequenzobjekte (Sequences).....	347
22.5.1	Was sind Sequenzen?.....	347
22.5.2	Erstellen von Sequenzen mit T-SQL.....	348
22.5.3	Erstellen von Sequenzen beim Forward Engineering	350
22.5.4	Sequenzen im Einsatz	351
22.6	Alternative Schlüssel.....	354
22.6.1	Alternative Schlüssel definieren.....	355
22.6.2	Alternative Schlüssel im Einsatz	357
22.7	Kaskadierendes Löschen (Cascading Delete)	360
22.7.1	Löschoptionen in Entity Framework Core	361
22.7.2	Beispiel	363
22.8	Abbildung von Datenbanksichten (Views)	368
22.8.1	Datenbanksicht anlegen	369
22.8.2	Entitätsklasse für die Datenbanksicht anlegen.....	369
22.8.3	Einbinden der Entitätsklasse in die Kontextklasse.....	370
22.8.4	Verwendung der Datenbanksicht.....	371
22.8.5	Herausforderung: Migrationen	372
22.9	Wertkonvertierungen (Value Converter)	374
22.9.1	Einschränkungen	375
22.9.2	Beispiel 1: Konvertierung zwischen String und Boolean.....	375
22.9.3	Beispiel 2: Konvertierung zwischen Aufzählungstyp und String	378
22.10	Datenbefüllung bei der Schemamigration (Data Seeding).....	382
22.10.1	Herausforderung Shadow Properties.....	385
22.10.2	Bug bei berechneten Spalten	387
23	Weitere Tipps und Tricks zu LINQ und SQL.....	389
23.1	Globale Abfragefilter (ab Version 2.0).....	389

23.1.1	Filter definieren	389
23.1.2	Filter nutzen	389
23.1.3	Praxistipp: Filter ignorieren.....	390
23.1.4	Globale Abfragefilter bei SQL-Abfragen (ab Version 2.0).....	390
23.1.5	Globale Abfragefilter bei Stored Procedures und Table Valued Functions.....	391
23.2	Zukünftige Abfragen (Future Queries).....	391
23.2.1	Konzept der Future Queries	391
23.2.2	Future()	392
23.2.3	FutureValue()	393
23.2.4	Bug in Verbindung mit EF Profiler.....	394
23.3	Befehlsverfolgung mit Query Tags (ab Version 2.2).....	395
23.3.1	TagWith().....	395
23.3.2	Einsatz von TagWith().....	395
23.3.3	Einschränkungen	402
23.4	Benachrichtigungen bei Datenänderungen (Query Notifications).....	402
23.4.1	SqlDependency für Microsoft SQL Server	402
23.4.2	Aufbau des SQL-Befehls	403
23.4.3	Query Notification in einer Konsolenanwendungen.....	404
23.4.4	Diagnosemöglichkeiten.....	407
23.4.5	Query Notification in einer Desktop-Anwendungen.....	408
24	Leistungsoptimierung (Performance Tuning).....	414
24.1	Vorgehensmodell zur Leistungsoptimierung bei Entity Framework Core.....	414
24.2	Best Practices für Ihre eigenen Leistungstests	414
24.3	Leistungsvergleich verschiedener Datenzugriffstechniken in .NET	415
24.4	Objektzuweisung optimieren	416
24.5	Massenoperationen.....	419
24.5.1	Einzellöschen	419
24.5.2	Optimierung durch Batching	419
24.5.3	Löschen ohne Laden mit Pseudo-Objekten.....	421
24.5.4	Einsatz von klassischem SQL anstelle des Entity Framework Core-APIs	422
24.5.5	Lambda-Ausdrücke für Massenlöschen mit EFPlus	424
24.5.6	Massenaktualisierung mit EFPlus.....	426
24.5.7	Optionen für Update() und Delete() bei EFPlus	426

24.6	Leistungsoptimierung durch No-Tracking.....	427
24.6.1	No-Tracking aktivieren.....	427
24.6.2	No-Tracking fast immer möglich	428
24.6.3	No-Tracking im änderbaren Datagrid	431
24.6.4	QueryTrackingBehavior und AsTracking().....	442
24.6.5	Konsequenzen des No-Tracking-Modus	443
24.6.6	Best Practices	443
24.7	Leistungsoptimierung durch Compiled Queries.....	444
24.7.1	Konzept einer Compiled Query.....	444
24.7.2	Compiled Queries in Entity Framework Core	445
24.7.3	Leistungstest.....	445
24.7.4	Einschränkungen	449
24.8	Auswahl der besten Ladestrategie	450
24.9	Zwischenspeicherung (Caching)	450
24.9.1	MemoryCache	451
24.9.2	CacheManager.....	453
24.10	Second-Level-Caching mit EFPlus	460
24.10.1	Einrichten des Second-Level-Cache	461
24.10.2	Verwenden des Second-Level-Cache.....	461
25	Softwarearchitektur mit Entity Framework Core	464
25.1	Monolithisches Modell.....	464
25.2	Entity Framework Core als Datenzugriffsschicht.....	465
25.3	Reine Geschäftslogik.....	466
25.4	Geschäftsobjekt- und ViewModel-Klassen.....	467
25.5	Verteilte Systeme	468
25.6	Fazit	471
26	Zusatzwerkzeuge	472
26.1	Entity Framework Core Power Tools.....	472
26.1.1	Funktionsüberblick	472
26.1.2	Reverse Engineering mit Entity Framework Core Power Tools	473
26.1.3	Schemamigrationen mit Entity Framework Core Power Tools	478
26.1.4	Diagramme mit Entity Framework Core Power Tools.....	480
26.2	LINQPad	481

26.2.1	Aufbau von LINQPad	482
26.2.2	Datenquellen einbinden.....	482
26.2.3	LINQ-Befehle ausführen.....	486
26.2.4	Abspeichern	488
26.2.5	Weitere LINQPad-Treiber.....	488
26.2.6	Interaktive Programmcodeeingabe	489
26.2.7	Fazit zu LINQPad.....	490
26.3	Entity Developer	490
26.3.1	Auswahl der ORM-Technik	491
26.3.2	Reverse Engineering mit Entity Developer	493
26.3.3	Forward Engineering mit Entity Developer	502
26.4	Entity Framework Profiler.....	507
26.4.1	Einbinden des Entity Framework Profilers	509
26.4.2	Befehle überwachen mit Entity Framework Profiler	509
26.4.3	Warnungen vor potenziellen Problemen	512
26.4.4	Analysefunktionen.....	513
26.4.5	Kommandozeilenunterstützung und API	514
26.4.6	Fazit zu Entity Framework Profiler	514
27	Zusatzkomponenten	515
27.1	Oracle-Treiber von DevArt (dotConnect for Oracle)	515
27.1.1	Unterstützte Oracle-Versionen	515
27.1.2	Installation.....	515
27.1.3	Visual Studio-Integration.....	517
27.1.4	Datenbanktreibername	520
27.1.5	Entity Framework Core-Werkzeuge	520
27.1.6	Kontextklasse	521
27.1.7	Entitätsklassen.....	521
27.1.8	Datentypen	521
27.2	Entity Framework Plus (EFPlus).....	523
27.3	Second-Level-Caching mit EFSecondLevelCache.Core	524
27.4	Objekt-Objekt-Mapping mit AutoMapper	524
27.4.1	Objekt-Objekt-Mapping per Reflection	526
27.4.2	AutoMapper	528

27.4.3	Beispielszenario.....	529
27.4.4	Abbildungen konfigurieren	531
27.4.5	Abbildung ausführen mit Map()	531
27.4.6	Nicht-statisches API	532
27.4.7	Abbildungskonventionen	532
27.4.8	Abbildungskonventionen ändern	534
27.4.9	Profilklassen.....	534
27.4.10	Verbundene Objekte	535
27.4.11	Manuelle Abbildungen.....	535
27.4.12	Typkonvertierungen.....	538
27.4.13	Objektmengen.....	539
27.4.14	Vererbung.....	540
27.4.15	Generische Klassen.....	543
27.4.16	Zusatzaktionen vor und nach dem Mapping.....	545
27.4.17	Geschwindigkeit	547
27.4.18	Fazit zu AutoMapper	548
27.5	Andere Erweiterungen.....	548
28	Praxislösungen.....	550
28.1	Entity Framework Core in einer ASP.NET Core-Anwendung.....	550
28.1.1	Das Fallbeispiel "MiracleList"	550
28.1.2	Architektur	554
28.1.3	Entitätsklassen.....	558
28.1.4	Entity Framework Core-Kontextklasse	560
28.1.5	Lebensdauer der Kontextklasse in ASP.NET Core-Anwendungen	563
28.1.6	Geschäftslogik.....	564
28.1.7	WebAPI	573
28.1.8	Verwendung von Entity Framework Core per Dependency Injection	583
28.1.9	Praxistipp: Kontextinstanzpooling (DbContext Pooling).....	586
28.2	DevOps mit Entity Framework (Continous Integration und Continous Delivery)	586
28.2.1	Unit Tests und Integrationstests mit Entity Framework Core.....	587
28.2.2	In-Memory-Treiber.....	587
28.2.3	SQLite- In-Memory-Treiber.....	590
28.2.4	Entity Framework Core beim serverseitigen Build (Continous Integration)	592

28.2.5	Entity Framework Core beim automatischen Release (Continous Delivery)	596
28.3	Entity Framework Core in einer Universal Windows Platform App	597
28.3.1	Das Fallbeispiel "MiracleList Light"	597
28.3.2	Architektur	598
28.3.3	Entitätsklassen.....	599
28.3.4	Entity Framework Core-Kontextklasse.....	601
28.3.5	Startcode	601
28.3.6	Erzeigte Datenbank.....	602
28.3.7	Datenzugriffscode.....	604
28.3.8	Benutzeroberfläche	608
28.4	Entity Framework Core in einer Xamarin-Cross-Platform-App	609
28.4.1	Das Fallbeispiel "MiracleList Light"	609
28.4.2	Architektur	611
28.4.3	Entitätsklassen.....	613
28.4.4	Entity Framework Core-Kontextklasse.....	614
28.4.5	Startcode	615
28.4.6	Erzeigte Datenbank.....	616
28.4.7	Datenzugriffscode.....	616
28.4.8	Benutzeroberfläche	619
28.5	N:M-Beziehungen zu sich selbst.....	621
29	Quellen im Internet	628
30	Stichwortverzeichnis (Index).....	629
31	Werbung in eigener Sache ☺.....	640

2 Vorwort

Liebe Leserinnen und Leser,

ich nutze Entity Framework in echten Softwareentwicklungsprojekten seit der allerersten Version, also seit der Version 1.0 von ADO.NET Entity Framework aus dem Jahr 2008. Zuvor hatte ich einen selbstentwickelten Objekt-Relationalen Mapper in meinen Projekten verwendet. Entity Framework Core ist das Nachfolgeprodukt, das es seit 2016 gibt. Ich setzte seitdem auch (aber nicht ausschließlich) Entity Framework Core in der Praxis ein. Viele Projekte laufen noch mit dem klassischen Entity Framework.

Microsoft entwickelt Entity Framework Core inkrementell, d.h. die ersten Versionen waren noch sehr unvollständig. Mit jeder neuen Version schließt Microsoft weitere Lücken. Seit Version 2.1 hat Entity Framework Core eine gute Reife erreicht, wenn auch es immer noch einzelne Schwächen (wie in jeder Software) gibt, die dieses Buch natürlich thematisiert.

Genau so wie sich Entity Framework Core weiterentwickelt hat, hat sich dieses Buch auch seit der ersten Version im September 2016 kontinuierlich weiterentwickelt. Die vor Ihnen liegende Ausgabe dieses Buchs beschreibt alle Kernaspekte und viele Tipps und Tricks sowie Praxisszenarien zu Entity Framework Core einschließlich der **Version 3.0 Preview 8**. Anfang Oktober 2018 wird eine Version des Buchs zu 3.0 RTM erscheinen. Ich plane, in Zukunft weitere Versionen dieses Buchs zu veröffentlichen, die die kommenden Versionen von Entity Framework Core beschreiben und auch weitere Tipps & Tricks sowie Praxisszenarien ergänzen.

Dieses Buch wird vertrieben auf folgenden Wegen (ich nenne neben dem Verkaufspreis auch, wie viel – bzw wenig – ich als Autor von den jeweiligen Händlern erhalte. Der Rest ist Gewinn der Händler):

- Gedruckt von Amazon für 49,00 Euro (davon erhält der Autor nur 19,19 Euro):
www.amazon.de/dp/393427921X
- Kindle-E-Book von Amazon für 44,00 Euro (davon erhält der Autor nur 12,49 Euro):
www.amazon.de/dp/B07WPX61SZ
- PDF-E-Book von Leanpub für 49,00 Dollar (ca. 44 Euro, davon erhält der Autor ca. 35,30 Euro):
www.leanpub.com/EntityFrameworkCore3

Sie können jederzeit Aktualisierungen der PDF-Buchs zum Preis von 15,00 Dollar beziehen (<https://leanpub.com/EntityFrameworkCore3/c/update>). Leider erlaubt Amazon nicht, dass Sie eine Aktualisierung im Kindle-Format oder in gedruckter Form vergünstigt erhalten. Käufer, die das Buch von Amazon in gedruckter Version bezogen haben, können unter dem o.g. Link zusätzlich das PDF-E-Book ebenfalls so günstig erhalten.

Da diese Preise in Anbetracht der vielen Stunden Arbeit an diesem Buch leider nicht nennenswert dazu beitragen können, den Lebensunterhalt meiner Familie zu bestreiten, ist dieses Projekt ein Hobby. Dementsprechend kann ich nicht garantieren, wann es Updates zu diesem Buch geben wird. Ich werde dann an diesem Buch arbeiten, wenn ich neben meinem Beruf als Softwarearchitekt, Berater und Dozent und meinen sportlichen Betätigungen noch etwas Zeit für das Fachbuchautorenhobby übrig habe.

Falls mir in diesem Buch oder den zugehörigen Downloads menschliche Fehler passiert sind, möchte ich mich dafür schon jetzt in aller Form entschuldigen bei Ihnen. Bitte geben Sie mir einen freundlichen, genau beschriebenen Hinweis auf meine Fehler. Ich freue mich immer über konstruktives Feedback und Verbesserungsvorschläge. Bitte verwenden Sie dazu das Kontaktformular auf www.dotnet-doktor.de.

Ich helfe Ihnen gerne, Ihren eigenen Programmcode zu schreiben, aber ich hoffe, Sie verstehen, dass ich dies nicht ehrenamtlich tun kann. Wenn Sie **technische Hilfe** zu Entity Framework und Entity Framework Core oder anderen Themen rund um .NET, Visual Studio, Windows oder andere Microsoft-Produkte benötigen, stehe ich Ihnen im Rahmen meiner beruflichen Tätigkeit für die Firmen www.IT-Visions.de (Beratung, Schulung, Support) und 5Minds IT-Solutions GmbH & Co KG (Softwareentwicklung, siehe www.5Minds.de) gerne zur Verfügung. Bitte wenden Sie sich für ein Angebot an das jeweilige Kundenteam. Bitte kontaktieren Sie die Firmen aber nicht für Feedback und Verbesserungsvorschläge zu diesem Buch, da dieses Buch reine Privatsache ist.

Die Beispiele zu diesem Buch können Sie herunterladen auf der von mir ehrenamtlich betriebenen **Leser-Website** unter www.IT-Visions.de/Leser. Dort müssen Sie sich registrieren. Bei der Registrierung wird ein Lösungswort abgefragt. Bitte geben Sie dort **Another Life** ein (siehe auch Kapitel "Programmcodebeispiele zum Download").

Herzliche Grüße aus Essen, dem Herzen der Metropole Ruhrgebiet

Holger Schwichtenberg

Hinweise zu Version 7.0.x dieses Buchs:

- Basiert auf Entity Framework Core 7.0 Preview 8
- Enthält noch einzelne TODO

Die Version 7.1 dieses Buchs wird Anfang Oktober 2019 erscheinen:

- Basiert auf Entity Framework Core 3.0 RTM
- ohne TODO
- mit Rechtschreibkorrektur aller Kapitel

3 Über den Autor

- Studienabschluss Diplom-Wirtschaftsinformatik an der Universität Essen
- Promotion an der Universität Essen im Gebiet komponentenbasierter Softwareentwicklung
- Seit 1996 selbstständig als unabhängiger Berater, Dozent, Softwarearchitekt und Fachjournalist
- Fachlicher Leiter des Berater- und Dozententeams bei www.IT-Visions.de
- Leitung der Softwareentwicklung im Bereich Microsoft/.NET bei der 5Minds IT-Solutions GmbH & Co. KG (www.5minds.de)
- Über 65 Fachbücher beim Carl Hanser Verlag, bei O'Reilly, Microsoft Press, APress und Addison-Wesley sowie mehr als 1000 Beiträge in Fachzeitschriften
- Gutachter in den Wettbewerbsverfahren der EU gegen Microsoft (2006-2009)
- Ständiger Mitarbeiter der Zeitschriften iX (seit 1999), dotnetpro (seit 2000) und Windows Developer (seit 2010) sowie beim Online-Portal heise.de (seit 2008)
- Regelmäßiger Sprecher auf nationalen und internationalen Fachkonferenzen (z.B. Microsoft TechEd, Microsoft Summit, Microsoft IT Forum, BASTA, BASTA-on-Tour, .NET Architecture Camp, Advanced Developers Conference, Developer Week, OOP, DOTNET Cologne, MD DevDays, Community in Motion, DOTNET-Konferenz, VS One, NRW.Conf, Net.Object Days, Windows Forum, Container Conf)
- Zertifikate und Auszeichnungen von Microsoft:
 - Microsoft Most Valuable Professional (MVP)
 - Microsoft Certified Solution Developer (MCSD)
- Thematische Schwerpunkte:
 - Softwarearchitektur, mehrschichtige Softwareentwicklung, Softwarekomponenten, SOA
 - Visual Studio, Continous Integration, Continous Delivery, Azure DevOps
 - Microsoft .NET Framework, Visual Studio, C#, Visual Basic
 - .NET-Architektur/Auswahl von .NET-Technologien
 - Einführung von .NET Framework und Visual Studio/Migration auf .NET
 - Webanwendungsentwicklung und Cross-Plattform-Anwendungen mit HTML, ASP.NET, JavaScript/TypeScript und Webframeworks wie Angular
 - Enterprise .NET, verteilte Systeme/Webservices mit .NET, insbesondere Windows Communication Foundation und WebAPI
 - Relationale Datenbanken, XML, Datenzugriffsstrategien
 - Objektrelationales Mapping (ORM), insbesondere ADO.NET Entity Framework und EF Core
 - Windows PowerShell, PowerShell Core und Windows Management Instrumentation (WMI)
- Ehrenamtliche Community-Tätigkeiten:
 - Vortragender für die International .NET Association (INETA)
 - Betrieb diverser Community-Websites: www.dotnet-lexikon.de, www.dotnetframework.de, www.windows-scripting.de, www.aspnetdev.de u. a.
- Firmenwebsites: <http://www.IT-Visions.de> und <http://www.5Minds.de>
- Weblog: <http://www.dotnet-doktor.de>
- Kontakt für geschäftliche Anfragen via Kundenteam:
E-Mail kundenteam@IT-Visions.de sowie Telefon **0201 / 64 95 90 - 0**
- Kontakt für Feedback zu diesem Buch: Kontaktformular auf <http://www.dotnet-doktor.de>



www.IT-Visions.de
Dr. Holger Schwichtenberg

5Minds
IT - S O L U T I O N S

4 Über dieses Buch

4.1 Versionsgeschichte dieses Buchs

Die folgende Tabelle zeigt die Versionen, die von diesem Fachbuch erschienen sind, sowie die darin besprochenen Entity Framework Core-Versionen.

Ergänzungen der Versionsnummer an der dritten Stelle (z.B. 1.2.3) sind kleine Korrekturen im Buch, die nicht explizit in dieser Versionstabelle erscheinen.

Buchversion Datum Umfang	Leanpub. com- Preis für PDF	Amazon.de- Preis für gedruckte Ausgabe und Kindle	Entity Framework Core- Version(en)	Bemerkung
1.0 16.09.2016 101 Seiten	15,00 Dollar	-	1.0.1	<p>Grundversion mit folgenden Kapiteln:</p> <ul style="list-style-type: none"> ▪ Was ist Entity Framework Core? ▪ Reverse Engineering bestehender Datenbanken ▪ Forward Engineering für neue Datenbanken ▪ Anpassung des Datenbankschemas ▪ Schemamigrationen ▪ Daten lesen mit LINQ ▪ Objektbeziehungen und Ladestrategien ▪ Einfügen, Löschen und Ändern
1.1 18.11.2016 122 Seiten	17,50 Dollar	-	1.1	<ul style="list-style-type: none"> ▪ Aktualisiert auf Entity Framework Core Version 1.1 ▪ Neues Unterkapitel: Laden anhand des Primärschlüssels mit Find() ▪ Neues Unterkapitel: Explizites Nachladen ▪ Neues Unterkapitel: Änderungsverfolgung auch für Unterobjekte ▪ Neues Kapitel: Leistungsoptimierung durch No-Tracking ▪ Neues Kapitel: Quellen im Internet

1.2 07.04.2017 145 Seiten	18,50 Dollar	19,99 Euro	1.1.1	<ul style="list-style-type: none"> ▪ Neues Kapitel: Datenänderungskonflikte ▪ Neues Kapitel: Praxislösungen / N:M-Beziehungen zu sich selbst
1.3 14.06.2017 194 Seiten	19,50 Dollar	-	1.1.2 und 2.0- Preview1	<ul style="list-style-type: none"> ▪ Aktualisiert auf Version 2.0 Preview 1 ▪ Erweitert: Fallbeispiel in diesem Buch ▪ Erweitert: LINQ im RAM statt in Datenbank ▪ Erweitert: Regeln für die selbsterstellte Kontextklasse ▪ Neues Kapitel: Artefakte der Entity Framework Core-Programmierung ▪ Neues Kapitel: Daten lesen/Globale Abfragefilter ▪ Neues Kapitel: Einfügen, Ändern und Löschen/Das Foreach-Problem ▪ Neues Kapitel: Einfügen, Ändern und Löschen/Transaktionen ▪ Neues Kapitel: Asynchrone Programmierung ▪ Neues Kapitel: Zusatzwerkzeuge: LINQPad, Entity Developer ▪ Erweitert: Tipps und Best Practices in einigen Kapiteln. ▪ Verbessert: Seitennummernformatierung
1.4 06.07.2017 210 Seiten	19,50 Dollar	19,99 Euro	1.1.2 und 2.0- Preview2	<ul style="list-style-type: none"> ▪ Aktualisiert auf Version 2.0 Preview 2 ▪ Neues Kapitel: Was ist Entity Framework Core/Unterstützte .NET-Versionen ▪ Neues Kapitel: Was ist Entity Framework Core/Unterstützte Visual Studio-Versionen ▪ Neues Kapitel: Installation von Entity Framework Core

				<ul style="list-style-type: none"> ▪ Neues Kapitel: Daten lesen und ändern mit SQL, Stored Procedures und Table Valued Functions
2.0 17.07.2017 296 Seiten	24,50 Dollar	24,99 Euro	1.1.2 und 2.0- Preview2	<ul style="list-style-type: none"> ▪ Neues Kapitel: "Was ist Entity Framework Core?/Was ist ein OR-Mapper?" ▪ Neues Kapitel: "Was ist Entity Framework Core?/ORM in der .NET-Welt" ▪ Neues Kapitel: "Einfügen, Löschen und Ändern/Change Tracker abfragen" ▪ Neues Kapitel: "Praxislösungen/Entity Framework Core in einer Universal Windows App" ▪ Neues Kapitel: "Protokollierung (Logging)" ▪ Neues Kapitel: "Dynamische LINQ-Abfragen" ▪ Kapitel "Daten lesen und ändern mit SQL, Stored Procedures und Table Valued Functions" erweitert um Globale Filter. ▪ Neues Kapitel: "Softwarearchitektur mit Entity Framework Core" ▪ Neues Kapitel: "Zusatzwerkzeuge/Profiling mit Entity Framework Profiler" ▪ Neues Kapitel: "Zusatzwerkzeuge/Objekt-Objekt-Mapping und AutoMapper" ▪ Stichwortverzeichnis (Index) ergänzt
2.1 14.08.2017 296 Seiten	24,50 Dollar	24,99 Euro	1.1.2 und 2.0	<ul style="list-style-type: none"> ▪ Aktualisiert auf die am 14.8. erschienene RTM-Version von Entity Framework Core 2.0
3.0 01.09.2017 395 Seiten	34,50 Dollar	34,99 Euro	1.1.2 und 2.0	<ul style="list-style-type: none"> ▪ Erweiterung des Kapitels: "Aktualisierung auf eine neue Version"

				<ul style="list-style-type: none"> ▪ Neues Kapitel: "Anpassung des Datenbankschemas/Indexe" ▪ Neues Kapitel: "Leistungsoptimierung" ▪ Neues Kapitel: "Tipps&Tricks/Shadow Properties" ▪ Neues Kapitel: "Tipps&Tricks/Table Splitting" ▪ Neues Kapitel: "Tipps&Tricks/Berechnete Spalten" ▪ Neues Kapitel: "Tipps&Tricks/Standardwerte" ▪ Neues Kapitel: "Tipps&Tricks/Sequenzen" ▪ Neues Kapitel: "Tipps&Tricks/Alternative Schlüssel" ▪ Neues Kapitel "Praxislösungen/Entity Framework Core in einer ASP.NET Core-Anwendung"
3.1 19.09.2017 422 Seiten	34,50 Dollar	34,99 Euro	1.1.2 und 2.0	<ul style="list-style-type: none"> ▪ Neues Kapitel: "Konzepte von Entity Framework Core/Vorgehensmodelle" ▪ Neues Kapitel: "Anpassung des Datenbankschemas/Weitere Syntaxoptionen für das Fluent-API" ▪ Neues Kapitel: "Daten lesen mit LINQ/ Umgehung für das GroupBy-Problem" ▪ Neues Kapitel: "Einfügen, Löschen und Ändern/Widersprüchliche Beziehungen" ▪ Aktualisiert: "Leistungsoptimierung/ Leistungsvergleich" ▪ Neues Kapitel: "Weitere Tipps und Tricks zum Mapping/Kaskadierendes Löschen"

				<ul style="list-style-type: none"> ▪ Neues Kapitel "Weitere Tipps und Tricks zum Mapping/Abbildung von Datenbanksichten (Views)" ▪ Neues Kapitel: "Weitere Tipps und Tricks zu LINQ"
4.0 06.10.2017 460 Seiten	39,00 Dollar	39,99 Euro	1.1.3 und 2.0	<ul style="list-style-type: none"> ▪ Neues Kapitel: "Daten lesen mit LINQ/ Kurzübersicht über die LINQ-Syntax" ▪ Neues Kapitel: "Praxislösungen/Entity Framework Core in einer Xamarin-Cross-Platform-App" ▪ Überarbeitetes Kapitel: "Zusatzkomponenten/AutoMapper" ▪ Verbesserung des Layouts
4.1 22.10.2017 474 Seiten	39,00 Dollar	39,99 Euro	1.1.3 und 2.0	<ul style="list-style-type: none"> ▪ Neues Kapitel: "Zusatzwerkzeuge/Entity Framework Core Power Tools"
4.2 20.12.2017 485 Seiten	39,00 Dollar	39,99 Euro	1.1.5 und 2.0.1	<ul style="list-style-type: none"> ▪ Neues Kapitel "Objektbeziehungen und Ladestrategien/Relationship Fixup"
4.3 03.01.2018 473 Seiten	39,00 Dollar	39,99 Euro	1.1.5 und 2.0.1	<ul style="list-style-type: none"> ▪ Einige Kapitel überarbeitet ▪ Neues Kapitel "Zusatzkomponenten/Oracle-Treiber von DevArt" ▪ Erweiterung des Kapitels "Leistungsoptimierung durch No-Tracking" ▪ Erweiterung des Kapitels "Weitere Tipps und Tricks zum Mapping/Shadow Properties" ▪ Formatierung der Listings nun kompakter, daher die verringerte Seitenzahl
4.4 02.03.2018 493 Seiten	39,00 Dollar	nur Kindle für 9,99 Euro	1.1.5 und 2.0.1	<ul style="list-style-type: none"> ▪ Erweiterung des Kapitels: "Datenbankschemamigrationen" ▪ Neues Kapitel "Praxislösungen/ Continuous Integration und Continuous Delivery"

5.0 20.03.2018 521 Seiten	44,00 Dollar	nur Kindle für 9,99 Euro	1.1.5, 2.0.2 und 2.1 Preview 1	<ul style="list-style-type: none"> ▪ Zahlreiche Stellen aktualisiert auf Version 2.1 Preview 1 ▪ Kapitel "LINQ/Gruppierungen" ergänzt ▪ Kapitel "Umgehung für das GroupBy-Problem" aktualisiert auf Entity Framework Core 2.1 Preview 1 ▪ Erweiterung des Kapitels "LINQ/Projektionen" ▪ Erweiterung des Kapitels "LINQ/Repository-Pattern" ▪ Kapitel "SQL/Nicht-Entitätsklassen als Ergebnismenge" aktualisiert auf Entity Framework Core 2.1 Preview 1 ▪ Kapitel "Weitere Tipps zum Mapping/Abbildung von Datenbanksichten" aktualisiert auf Entity Framework Core 2.1 Preview 1 ▪ Kapitel "Weitere Tipps zum Mapping/Wertkonvertierungen" ergänzt ▪ Kapitel "Einfügen, Löschen und Ändern/ Datenbanktransaktionen/TransactionsScope" ergänzt
5.1 15.05.2018 521 Seiten	44,00 Dollar	nur Kindle für 9,99 Euro	1.1.5, 2.0.2 und 2.1 RC 1	<ul style="list-style-type: none"> ▪ Aktualisiert auf 2.1 RC1
5.2 14.06.2018 521 Seiten	44,00 Dollar	nur Kindle für 9,99 Euro	1.1.5, 2.0.3 und 2.1	<ul style="list-style-type: none"> ▪ Aktualisiert auf 2.1 RTM
5.3 21.08.2018 547 Seiten	44,00 Dollar	nur Kindle für 9,99 Euro	1.1.6, 2.0.3 und 2.1.1	<ul style="list-style-type: none"> ▪ Kapitel "Future Queries" erweitert ▪ Kapitel "Massenoperationen" erweitert ▪ Kapitel "Sequenzen" erweitert ▪ Kapitel "Table Splitting" erweitert ▪ Kapitel "Data Seeding" hinzugefügt

				<ul style="list-style-type: none"> ▪ Kapitel "Migration von ADO.NET Entity Framework zu Entity Framework Core" hinzugefügt
6.0 561 Seiten 12.10.2018	44,00 Dollar	nur Kindle für 9,99 Euro	1.1.6, 2.0.3 und 2.1.4 und 2.2 Preview 3	<ul style="list-style-type: none"> ▪ Kapitel "Entity Framework Core Power Tools" erweitert ▪ Kapitel "Oracle-Treiber von DevArt" erweitert ▪ Unterkapitel "Verwendung von DbContextOptions" ergänzt ▪ Kapitel "Query Tags"
6.1 15.11.2018 595 Seiten	49,00 Dollar	nur Kindle für 9,99 Euro	1.1.6, 2.0.3 und 2.1.4 und 2.2 Preview 3	<ul style="list-style-type: none"> ▪ Kapitel "Datenbankschemamigrationen/Migrationsszenarien" erweitert ▪ Kapitel "Forward Engineering für neue Datenbanken" erweitert ▪ Neues Kapitel "Anpassung des Datenbankschemas/Vererbung". ▪ Neues Kapitel "Daten lesen mit LINQ/Lokaler Cache in der Kontextklasse"
6.2 10.12.2018 600 Seiten	49,00 Dollar	nur Kindle für 9,99 Euro	1.1.6, 2.0.3 und 2.1.4 und 2.2.0	<ul style="list-style-type: none"> ▪ Aktualisiert auf 2.2 RTM ▪ Neues Unterkapitel "Benachrichtigungen bei Datenänderungen (Query Notifications)"
6.3 10.02.2018 612 Seiten	49,00 Dollar	nur Kindle für 9,99 Euro	1.1.6, 2.0.3 und 2.1.4 und 2.2.1	<ul style="list-style-type: none"> ▪ Kapitel "Alternative Schlüssel" erweitert ▪ Kapitel "Kaskadierendes Löschen (Cascading Delete)" erweitert ▪ Im Kapitel "Daten lesen mit LINQ" und "Objektbeziehungen und Ladestrategien" die Ausführungen zum lokalen Cache erweitert. ▪ Neues Kapitel "Pläne für Version 3.0"
7.0 27.08.2019 638 Seiten	49,00 Dollar	49,00 Euro Kindle: 44,00 Euro	3.0 Preview 8	<ul style="list-style-type: none"> ▪ Aktualisiert auf Änderungen und Neuerungen in EF Core 3.0 Preview 8 ▪ Kapitel "Kritik an der Entwicklungsgeschwindigkeit von Entity Framework Core" ergänzt

				<ul style="list-style-type: none"> ▪ Kapitel "Query Tags" erweitert ▪ Neues Kapitel "Leistungsoptimierung/Compiled Queries" ▪ Aktualisierung der zentralen Beispiele auf .NET Standard 2.1 und .NET Core 3.0 (ausgenommen UWP und Xamarin, die noch nicht mit Entity Framework Core 3.0 laufen)
7.1 geplant für Oktober 2019	49,00 Dollar	49,00 Euro Kindle: 44,00 Euro	3.0 RTM	<ul style="list-style-type: none"> ▪ Aktualisiert auf Änderungen und Neuerungen in EF Core 3.0 RTM

4.2 Bezugsquelle für Aktualisierungen

Wenn Sie eine ältere Version dieses Buch besitzen, können Sie jederzeit eine aktuelle PDF-Version zum stark vergünstigten Preis von nur 15,00 Dollar (zzgl. 19% Mehrwertsteuer) unter folgendem Link beziehen:

<https://leanpub.com/EntityFrameworkCore3/c/update>

Sie können diesen Link auch verwenden, wenn Sie eine gedruckte Version bei Amazon gekauft haben und nun gerne auch das E-Book zusätzlich hätten (zum Beispiel für die Volltextsuche).

Leider erlaubt Amazon nicht, dass Sie eine Aktualisierung im Kindle-Format oder in gedruckter Form vergünstigt erhalten.

4.3 Geplante Kapitel

Die Reihenfolge der für die folgenden Versionen geplanten Kapitel ist hier zunächst alphabetisch angeordnet und entspricht nicht der Reihenfolge, in der die Kapitel erscheinen werden.

- Anpassung der Code-Generierung beim Reverse Engineering mit Handlebars (Paket EntityFrameworkCore.Scaffolding.Handlebars)
- Auditing mit Entity Framework Plus
- Change Tracking mit INotifyPropertyChanged (ChangeTrackingStrategy)
- Command Interceptors (ab Entity Framework Core 3.0)
- Connection Resiliency / EnableRetryOnFailure (seit Entity Framework Core 1.1)
- Include für abgeleitete Typen (ab Entity Framework Core 2.1)
- Mapping mit (privaten) Feldern (seit Entity Framework Core 1.1)
- Parameter in Konstruktoren von Entitätsklassen (ab Entity Framework Core 2.1)
- Reverse Engineering von Datenbanksichten (ab Entity Framework Core 3.0)
- Scalare Datenbankfunktionen nutzen (seit Entity Framework Core 2.0)

- Spaltensortierung beim Anlegen einer Tabelle (ab Entity Framework Core 2.1)
- SQL Server memory-optimized Tables (seit Entity Framework Core 1.1)
- SQL-Ausführung in Datenbankschemamigrationen mit Sql() (seit Entity Framework Core 1.0)
- Temporale Tabellen (seit SQL Server 2016)
- Trackable Entities (<http://trackableentities.github.io>)
- User Defined Functions (UDF) nutzen
- Visual Studio Analyzer für SQL-Abfragen (ab Entity Framework Core 2.1)
- Zugriff auf Cosmos DB (ab Entity Framework Core 3.0)
- Zusammenarbeit mit SqlDependency
- Zusätzliche Erweiterungen wie z.B. EntityFrameworkCore.Rx, EFDetached.EntityFramework, EntityFrameworkCore.Triggers, EntityFrameworkCore.PrimaryKey und EntityFrameworkCore.TypedOriginalValues
- Zustandsänderungsereignisse im Change Tracker (ab Entity Framework Core 2.1)

4.4 Programmiersprache in diesem Buch

Als Programmiersprache kommt in diesem Buch C# zum Einsatz, weil dies die bei weitem am häufigsten verwendete Programmiersprache in .NET ist. Der Autor dieses Buchs programmiert in einigen Kundenprojekten .NET-Anwendungen zwar auch in Visual Basic .NET, leider bietet dieses Buch jedoch nicht den Raum, alle Listings in beiden Sprachen wiedergeben.

Eine Sprachkonvertierung zwischen C# und Visual Basic .NET ist im WWW kostenfrei verfügbar z.B. auf der Website <http://converter.telerik.com>.

5 Fallbeispiele in diesem Buch

Die meisten Beispielprogrammcodes in diesem Buch drehen sich um das Fallbeispiel der fiktiven Fluggesellschaft "World Wide Wings", abgekürzt "WWWings" oder als dreibuchstabiger Airline Code einfach "WWW". Es gibt auch eine Website zu der Fluggesellschaft (www.world-wide-wings.de) – dort einen Flug zu buchen, möchte der Autor dieses Buchs Ihnen aber nicht empfehlen ☺



Abbildung: Logo der fiktiven Fluggesellschaft "World Wide Wings"

Hinweis: In einzeln Unterkapitel werden andere Fallbeispiele verwendet (z.B. die Aufgabenverwaltung "MiracleList"). Diese Fallbeispiele werden dann in den jeweiligen Kapiteln erläutert.

5.1 Entitäten

Im Anwendungsfall "World Wide Wings" geht es um folgende Entitäten:

- **Flüge** zwischen zwei Orten, bei denen die Orte bewusst nicht als eigene Entität modelliert wurden, sondern Zeichenketten sind (dies vereinfacht das Verständnis vieler Beispiele)
- **Passagiere**, die auf Flügen fliegen
- **Mitarbeiter** der Fluggesellschaft, die wiederum Vorgesetzte haben, die auch Mitarbeiter sind
- **Piloten** als eine Spezialisierung von Mitarbeitern. Ein Flug hat einfacheren Modell nur einen Piloten. Es gibt keinen Copiloten bei World Wide Wings. Den Copiloten abzuschaffen und im Notfall das Flugzeug von der Stewardess landen zu lassen (wie im Film "Turbulence" von 1997), war übrigens ein echter Vorschlag von Michael O'Leary, dem Chef der irischen Fluggesellschaft Ryanair im Jahr 2010 (siehe [<http://www.dailymail.co.uk/news/article-130852/Let-stewardesses-land-plane-crisis-says-Ryanair-boss-Airline-wants-ditch-pilots.html>]).
- **Personen** als Sammlung der gemeinsamen Eigenschaften für alle Menschen in diesem Beispiel. Personen gibt es aber nicht eigenständig, sondern nur in den Ausprägungen/Spezialisierungen Passagier, Mitarbeiter und Pilot. Im objektorientierten Sinne ist Person also eine abstrakte Basisklasse, die keine Instanzen besitzen kann, sondern nur der Vererbung dient.

Es gibt zwei Datenmodelle:

- Das etwas einfachere Modell #1 (alias Modell Version 1, siehe Abbildungen 1 und 2) ist das Ergebnis klassischen relationalen Datenbankdesigns mit Normalisierung. Das Objektmodell daraus entsteht per Reverse Engineering.
- Modell #2 (alias Modell Version 2, siehe Abbildungen 3 und 4) ist das Ergebnis des Forward Engineering mit Entity Framework Core aus einem Objektmodell. Zusätzlich gibt es hier weitere Entitäten (Persondetail, Flugzeugtyp und Flugzeugtypdetail), um weitere Modellierungsaspekte aufzeigen zu können. In diesem Fall gibt es auch für jeden Flug einen optionalen Copiloten.

In Modell #1 gibt es eine jeweils eigene Tabelle für Personen (auch wenn es keine eigenständigen Personen gibt), Mitarbeiter, Piloten und Passagiere. Diese Aufteilung entspricht den Klassen im Objektmodell.

Hinweis: Bitte beachten Sie, dass die Objektmodelle, die in diesem Buch zu den Datenmodellen erstellt werden, nicht das Idealbild eines Objektmodells darstellen können, denn Entity Framework Core unterstützt einige Mapping-Möglichkeiten wie z.B. das N:M-Mapping noch nicht.

Das Objektmodell zum Datenbankschema World Wide Wings Version 1 (Abbildung 2) ist das automatisch von Entity Framework Core aus der Datenbank generierte Objektmodell (Reverse Engineering); es ist bewusst nicht verändert worden, auch wenn einige der generierten Namen unschön sind.

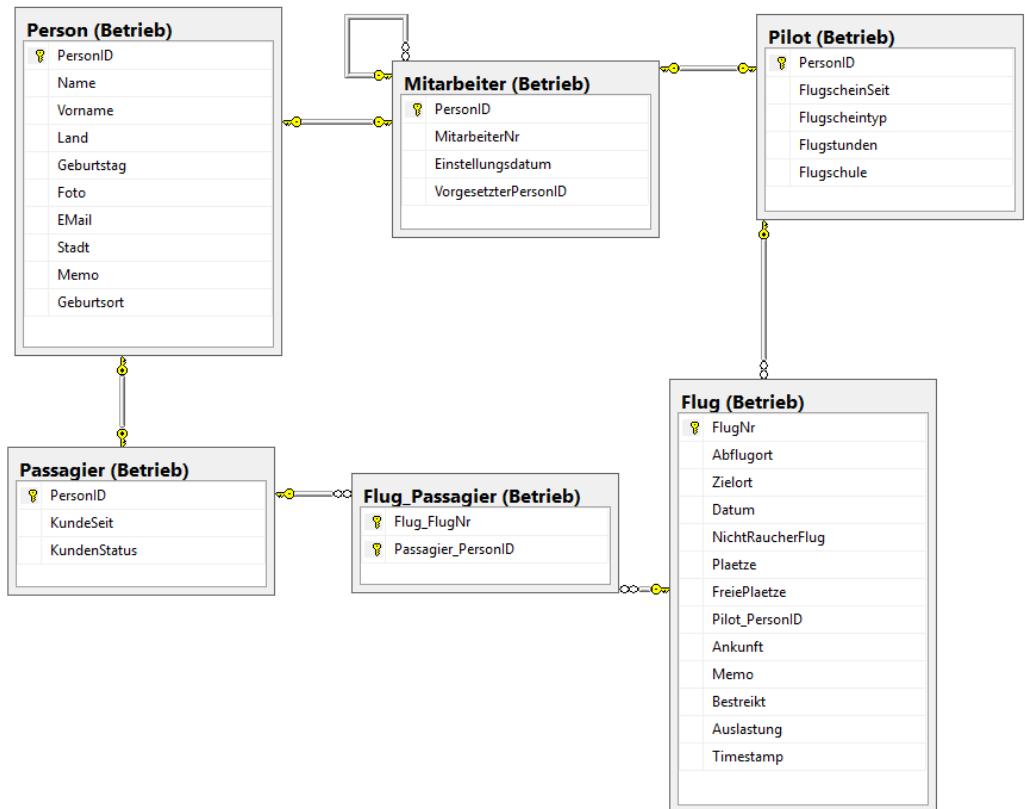


Abbildung 1: World Wide Wings-Datenmodell in der einfacheren Version 1

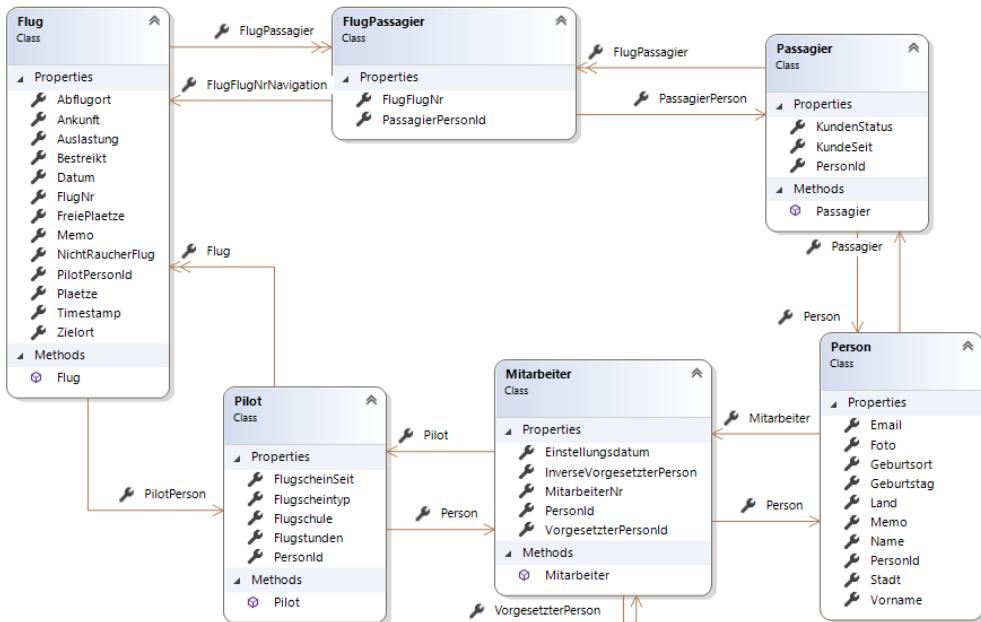


Abbildung 2: Objektmodell zum World Wide Wings-Datenmodell in der einfacheren Version 2

In Modell #2 gibt es lediglich die Tabellen Passagiere und Mitarbeiter für diese vier Entitäten. Entity Framework Core ist derzeit etwas eingeschränkt und unterstützt das "Table per Type"-Mapping (also eine eigenständige Tabelle für jede Klasse) nicht. Daher umfasst die Tabelle Passagiere auch alle Eigenschaften von Person. Die Tabelle Mitarbeiter umfasst neben den Personeneigenschaften die Eigenschaften der Entitäten Mitarbeiter und Pilot. In der Tabelle wird per Diskriminatorspalte unterschieden zwischen Datensätzen, die ein Mitarbeiter sind, und solchen, die ein Pilot sind. Entity Framework Core mischt hier die Konzepte Table per Concrete Type (TPC) und Table per Hierarchy (TPH). Einen dezidierten Einfluss auf diese Abbildung hat man in Entity Framework Core 1.x/2.0 noch nicht. Das klassische Entity Framework bietet hier mehr Optionen.

Die Abhängigkeitsarten in Modell #2 sind:

- Ein **Flug** muss einen Piloten besitzen. Es gibt einen Copilot, aber er ist optional.
- Ein Flug kann optional einen **Flugzeugtyp** zugeordnet haben. Ein Flugzeugtyp hat eine Beziehung zu **Flugzeugtypdetail**.
- Jede Person und damit auch jeder Pilot und Passagier muss ein **Personendetail**-Objekt besitzen.

In diesem Buch kommen beide Datenmodelle vor, teilweise auch in modifizierter Form, um bestimmte Szenarien (z.B. Datenbankschemamigrationen) aufzuzeigen.

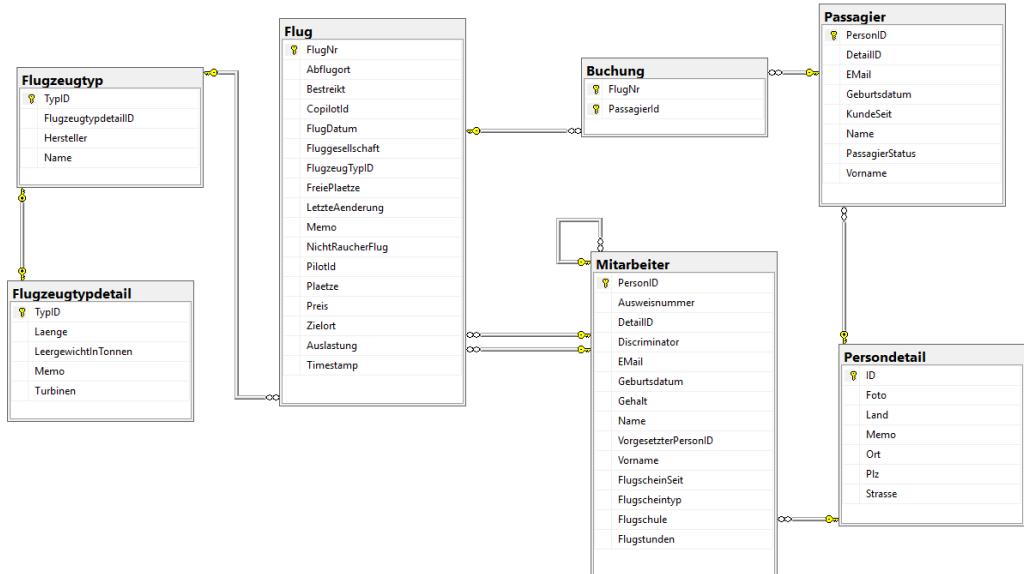


Abbildung 3: World Wide Wings-Datenmodell in der komplexeren Version 2

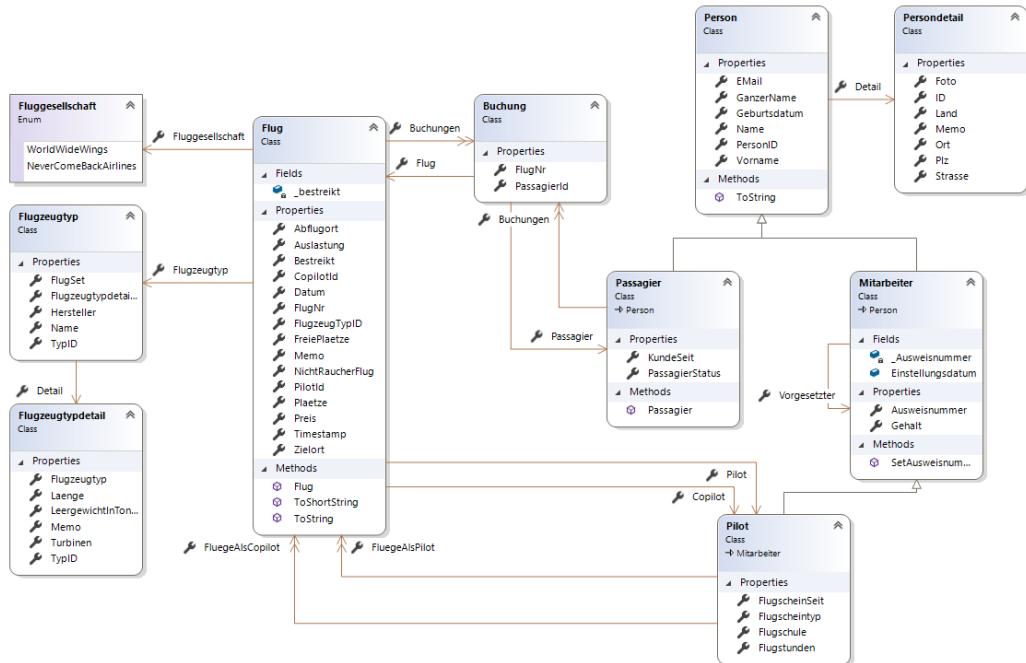


Abbildung 4: Objektmodell zum World Wide Wings-Datenmodell in der komplexeren Version 2

5.2 Englische Version des Beispiels

Da dieses Buch im Jahr 2018 auch in englischer Sprache (beim US-amerikanischen Verlag "APress") erschienen ist, war es notwendig, die Programmcodebeispiele auf Englisch umzustellen, da der Aufwand für die Pflege von Quellcode in zwei Sprachen nicht wirtschaftlich vertretbar war. Daher verwenden einige neuere Beispiele in diesem Buch bereits die englischen Klassennamen, z.B. Flight statt Flug und Passenger statt Passagier sowie Airline statt Fluggesellschaft und Booking statt Buchung usw.

Die folgende Abbildung zeigt das analog Objektmodell in englischer Sprache.

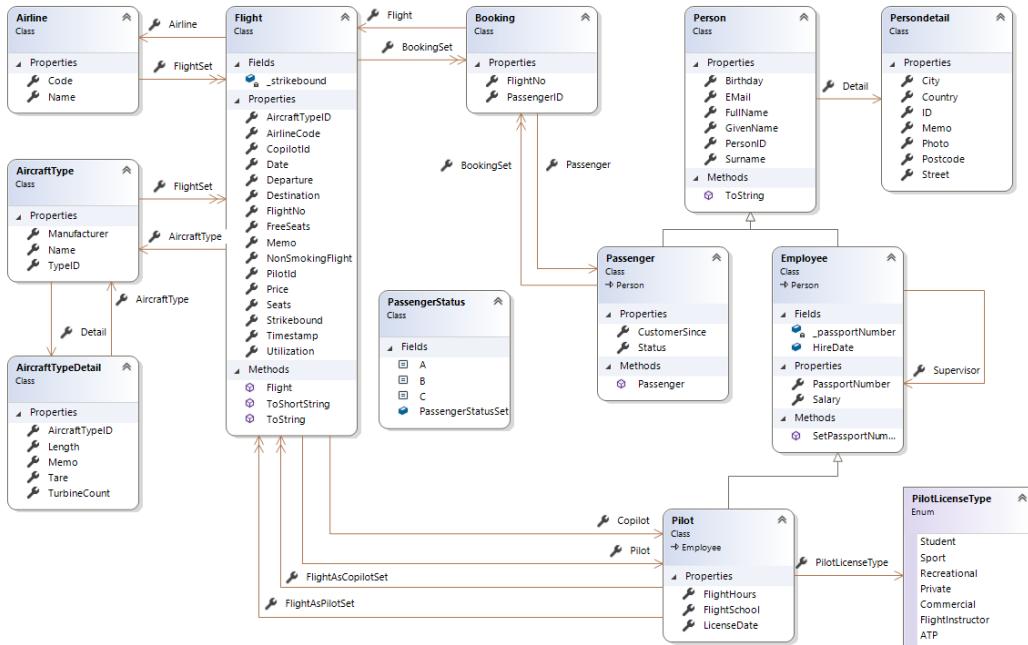


Abbildung: Englische Fassung des Objektmodells zum World Wide Wings-Datenmodell in der komplexeren Version 2

5.3 Anwendungsarten in diesem Buch

In diesem Buch erfolgen Bildschirmausgaben meist an der textbasierten Konsole in Konsolenanwendungen, denn dies ermöglicht die Fokussierung auf den Datenbankzugriff. Beim Einsatz von grafischen Benutzeroberflächen wie Windows Presentation Foundation (WPF), Windows Forms, ASP.NET Webforms oder ASP.NET MVC bzw. ASP.NET Core MVC und ASP.NET Core Razor Pages ist die Darstellung durch Datenbindung entkoppelt, das heißt man würde immer ein zweites Listing brauchen, um zu verstehen, dass die Datenzugriffe überhaupt liefern. Eingaben des Benutzers werden in den Konsolenbeispielen durch Variablen zu Beginn des Programmcodes simuliert.

Der Autor dieses Buchs führt seit vielen Jahren Schulungen und Beratungseinsätze im Bereich Datenzugriff durch und hat dabei die Erfahrung gemacht, dass Konsolenausgaben das didaktisch beste Instrument sind, da die Listings sonst sehr umfangreich und damit schlechter zu verstehen sind.

Natürlich ist die Konsolenausgabe in 99% der Fälle der Softwareentwicklung nicht die gängige Praxis. Grafische Benutzeroberflächen sind Inhalt anderer Bücher, und die Datenbindung hat in der Regel keinen Einfluss auf die Form des Datenzugriffs. Dort, wo der Datenzugriff doch relevant ist, wird dieses Buch auch Datenbindungsbeispiele zeigen.

5.4 Hilfsroutinen zur Konsolenausgabe

Für die Bildschirmausgabe an der Konsole wird an mehreren Stellen nicht nur `Console.WriteLine()` verwendet, sondern auch Hilfsroutinen kommen zur Anwendung, die farbige Bildschirmausgaben erzeugen. Diese Hilfsroutinen in der Klasse `CUI` aus der `ITV_DemoUtil.dll` sind hier zum besseren Verständnis abgedruckt:

Listing: Klasse CUI mit Hilfsroutinen für die Bildschirmausgabe an der Konsole

```
using System;
using System.Runtime.InteropServices;
using System.Web;
using ITVVisions.UI;
using System.Diagnostics;

namespace ITVVisions
{
    /// <summary>
    /// Helper utilities for console UIs
    /// (C) Dr. Holger Schwichtenberg 2002-2018
    /// </summary>
    public static class CUI
    {
        public static bool IsDebug = false;
        public static bool IsVerbose = false;

        #region Print only under certain conditions
        public static void PrintDebug(object s)
        {
            PrintDebug(s, System.Console.ForegroundColor);
        }

        public static void PrintVerbose(object s)
        {
            PrintVerbose(s, System.Console.ForegroundColor);
        }
        #endregion

        #region Issues with predefined colors
        public static void MainHeadline(string s)
        {
            Print(s, ConsoleColor.Black, ConsoleColor.Yellow);
        }

        public static void Headline(string s)
        {
            Print(s, ConsoleColor.Yellow);
        }

        public static void HeaderFooter(string s)
```

```
{  
    Console.ForegroundColor = ConsoleColor.Green;  
    Console.WriteLine(s);  
    Console.ForegroundColor = ConsoleColor.Gray;  
}  
  
public static void SubHeadline(string s)  
{  
    Print(s, ConsoleColor.White);  
}  
  
public static void PrintSuccess(object s)  
{  
    Print(s, ConsoleColor.Green);  
}  
  
public static void H1(string s)  
{  
    MainHeadline(s);  
}  
  
public static void H2(string s)  
{  
    Headline(s);  
}  
  
public static void H3(string s)  
{  
    SubHeadline(s);  
}  
  
public static void PrintGreen(string s)  
{  
    Print(s, ConsoleColor.Green);  
}  
  
public static void PrintYellow(string s)  
{  
    Print(s, ConsoleColor.Yellow);  
}  
  
public static void PrintRed(string s)  
{  
    Print(s, ConsoleColor.Red);  
}  
  
public static void PrintSuccess(object s)  
{  
    Print(s, ConsoleColor.Green);  
}  
  
public static void PrintStep(object s)  
{
```

```
Print(s, ConsoleColor.Cyan);
}

public static void PrintDebugSuccess(object s)
{
    PrintDebug(s, ConsoleColor.Green);
}

public static void PrintVerboseSuccess(object s)
{
    PrintVerbose(s, ConsoleColor.Green);
}

public static void PrintWarning(object s)
{
    Print(s, ConsoleColor.Cyan);
}

public static void PrintDebugWarning(object s)
{
    PrintDebug(s, ConsoleColor.Cyan);
}

public static void PrintVerboseWarning(object s)
{
    PrintVerbose(s, ConsoleColor.Cyan);
}

public static void PrintError(object s)
{
    Print(s, ConsoleColor.White, ConsoleColor.Red);
}

public static void PrintDebugError(object s)
{
    PrintDebug(s, ConsoleColor.White, ConsoleColor.Red);
}

public static void PrintVerboseError(object s)
{
    Print(s, ConsoleColor.White, ConsoleColor.Red);
}

public static void Print(object s)
{
    PrintInternal(s, null);
}
#endregion

#region Print with selectable color

public static void Print(object s, ConsoleColor farbe, ConsoleColor?
hintergrundfarbe = null)
```

```
{  
    PrintInternal(s, farbe, hintergrundfarbe);  
}  
  
public static void PrintDebug(object s, ConsoleColor farbe, ConsoleColor?  
hintergrundfarbe = null)  
{  
    if (IsDebug || IsVerbose) PrintDebugOrVerbose(s, farbe, hintergrundfarbe);  
}  
  
public static void PrintVerbose(object s, ConsoleColor farbe)  
{  
    if (!IsVerbose) return;  
    PrintDebugOrVerbose(s, farbe);  
}  
#endregion  
  
#region Print with additional data  
  
/// <summary>  
/// Print with Thread-ID  
/// </summary>  
public static void PrintWithThreadId(string s, ConsoleColor c =  
ConsoleColor.White)  
{  
    var ausgabe = String.Format("Thread #{0:00} {1:}: {2}",  
System.Threading.Thread.CurrentThread.ManagedThreadId,  
DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss.fff"), s);  
    CUI.Print(ausgabe, c);  
}  
  
/// <summary>  
/// Print with time  
/// </summary>  
public static void PrintWithTime(object s, ConsoleColor c = ConsoleColor.White)  
{  
    CUI.Print(DateTime.Now.Second + ":" + DateTime.Now.Millisecond + ":" + s);  
}  
  
private static long count;  
/// <summary>  
/// Print with counter  
/// </summary>  
private static void PrintWithCounter(object s, ConsoleColor farbe,  
ConsoleColor? hintergrundfarbe = null)  
{  
    count += 1;  
    s = $"{count:0000}: {s}";  
    CUI.Print(s, farbe, hintergrundfarbe);  
}  
  
#endregion  
  
#region internal helper routines
```

```
private static void PrintDebugOrVerbose(object s, ConsoleColor farbe,
ConsoleColor? hintergrundfarbe = null)
{
    count += 1;
    s = $"{{count:0000}}: {s}";
    Print(s, farbe, hintergrundfarbe);
    Debug.WriteLine(s);
    Trace.WriteLine(s);
    Trace.Flush();
}

/// <summary>
/// Output to console, trace and file
/// </summary>
/// <param name="s"></param>
[DebuggerStepThrough()]
private static void PrintInternal(object s, ConsoleColor? farbe = null,
ConsoleColor? hintergrundfarbe = null)
{
    if (s == null) return;

    if (HttpContext.Current != null)
    {
        try
        {
            if (farbe != null)
            {
                HttpContext.Current.Response.Write($"<span style='color:{farbe.Value.DrawingColor().Name}'>{s}<br>{s}</span>");
            }
            if (!HttpContext.Current.Request.Url.ToString().ToLower().Contains(".asmx") && !HttpContext.Current.Request.Url.ToString().ToLower().Contains(".svc") && !HttpContext.Current.Request.Url.ToString().ToLower().Contains("/api/"))
                HttpContext.Current.Response.Write(s.ToString() + "<br>");

            if (farbe != null)
            {
                HttpContext.Current.Response.Write("</span>");
            }
        }
        catch (Exception)
        {
        }
    }
    else
    {
        object x = 1;
        lock (x)
        {
            ConsoleColor alteFarbe = Console.ForegroundColor;
            ConsoleColor alteHFarbe = Console.BackgroundColor;

            if (farbe != null) Console.ForegroundColor = farbe.Value;
        }
    }
}
```

```
    if (hintergrundfarbe != null) Console.BackgroundColor =
hintergrundfarbe.Value;

    //if (farbe.ToString().Contains("Dark")) Console.BackgroundColor =
ConsoleColor.White;
    //else Console.BackgroundColor = ConsoleColor.Black;

    Console.WriteLine(s);
    Console.ForegroundColor = alteFarbe;
    Console.BackgroundColor = alteHFarbe;
}

}

}

#endregion

#region Set the position of the console window
[DllImport("kernel32.dll", ExactSpelling = true)]
private static extern IntPtr GetConsoleWindow();
private static IntPtr MyConsole = GetConsoleWindow();

[DllImport("user32.dll", EntryPoint = "SetWindowPos")]
public static extern IntPtr SetWindowPos(IntPtr hWnd, int hWndInsertAfter, int
x, int Y, int cx, int cy, int wFlags);

// Set the position of the console window without size
public static void SetConsolePos(int xpos, int ypos)
{
    const int SWP_NOSIZE = 0x0001;
    SetWindowPos(MyConsole, 0, xpos, ypos, 0, 0, SWP_NOSIZE);
}

// Set the position of the console window with size
public static void SetConsolePos(int xpos, int ypos, int w, int h)
{
    SetWindowPos(MyConsole, 0, xpos, ypos, w, h, 0);
}

#endregion
}
```