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

www.IT-Visions.de®

Dr. Holger Schwichtenberg

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

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 thematisisiert.

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 Losungswort 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*

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	Grundversion mit folgenden Kapiteln: ▪ 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	▪ 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	▪ 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	▪ 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	▪ 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

				▪ 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	▪ 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	▪ 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	▪ Erweiterung des Kapitels: "Aktualisierung auf eine neue Version"

				▪ 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	▪ 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"

				▪ Neues Kapitel "Weitere Tipps und Tricks zum Mapping/Abbildung von Datenbansichten (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	▪ 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	▪ 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	▪ 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	▪ 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	▪ Erweiterung des Kapitels: "Datenbankschemamigrationen" ▪ Neues Kapitel "Praxislösungen/ Continous Integration und Continous 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	▪ Zahlreiche Stellen aktualisiert auf Version 2.1 Preview 1 ▪ Kapitel "LINQ/Gruppierungen" ergänt ▪ 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/TransactionScope" 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	▪ Aktualisert 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	▪ Aktualisert 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	▪ Kapitel "Future Queries" erweitert ▪ Kapitel "Massenoperationen" erweitert ▪ Kapitel "Sequenzen" erweitert ▪ Kapitel "Table Splitting" erweitert ▪ Kapitel "Data Seeding" hinzugefügt

				▪ 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	▪ 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	▪ 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	▪ Aktualisert auf 2.2 RTM ▪ NeuesUnterkapitel "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	▪ 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	▪ Aktualisiert auf Änderungen und Neuerungen in EF Core 3.0 Preview 8 ▪ Kapitel "Kritik an der Entwicklungsgeschwindigkeit von Entity Framework Core" ergänzt

				▪ 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	▪ 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 wiederzugeben.

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-1308852/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 optional 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.

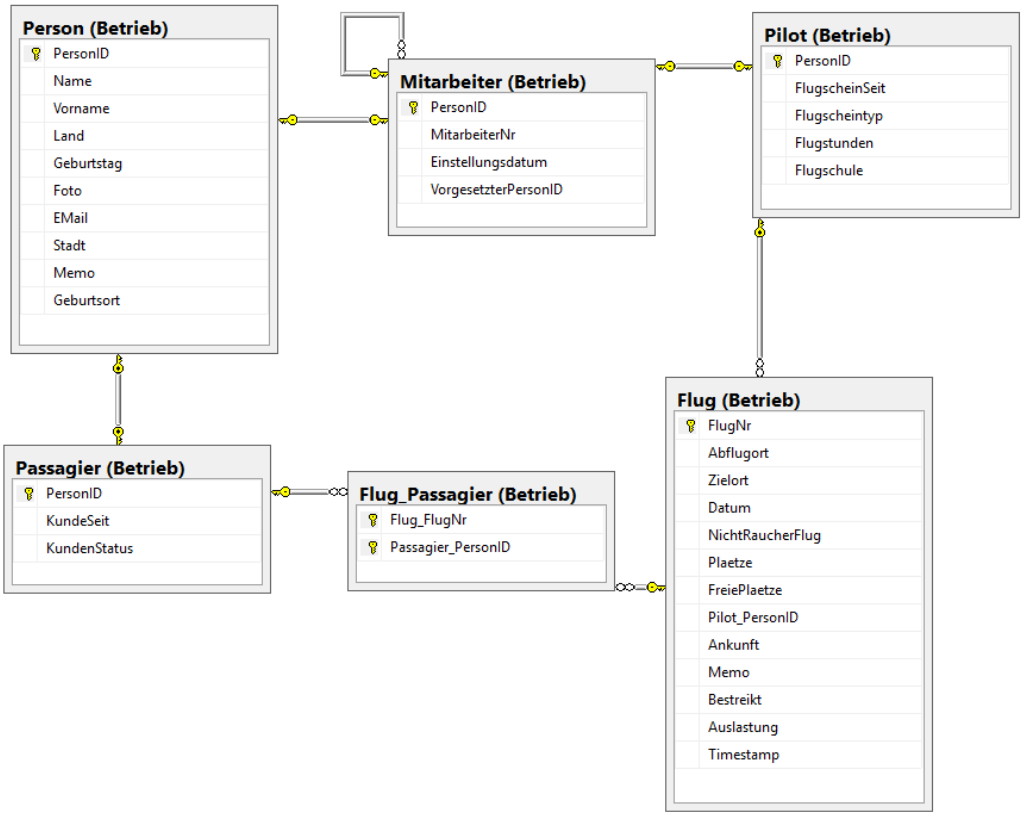

Abbildung 1: World Wide Wings-Datenmodell in der einfacheren Version 1

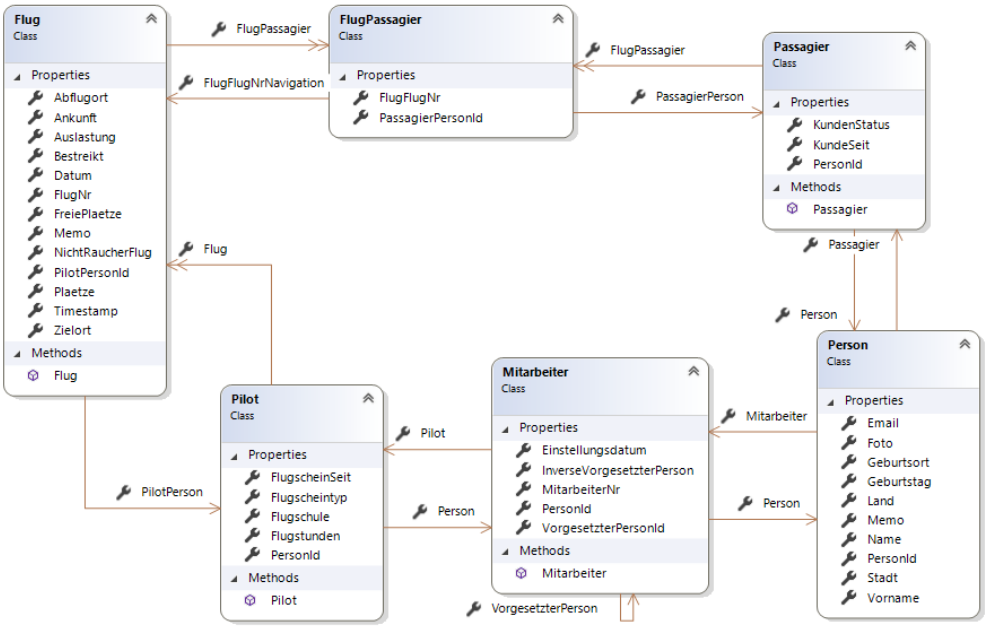

Abbildung 2: Objektmodell zum World Wide Wings-Datenmodell in der einfacheren Version 2

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 **Flugzeutyp** zugeordnet haben. Ein Flugzeugtyp hat eine Beziehung zu **Flugzeugtypdetail**.
- Jede Person und damit auch jeder Pilot und Passagier muss ein **Persondetail**-Objekt besitzen.

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

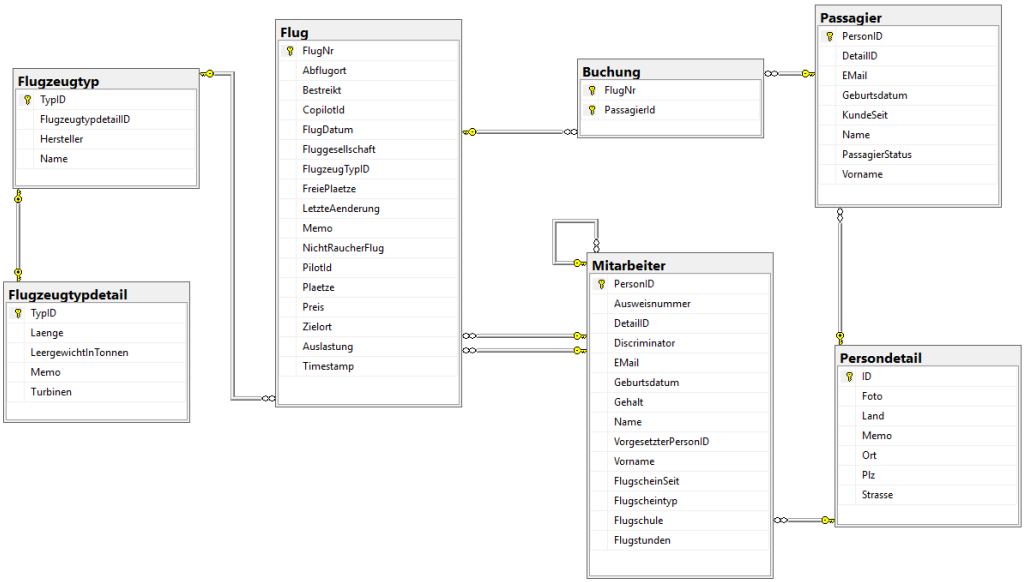

Abbildung 3: World Wide Wings-Datenmodell in der komplexeren Version 2

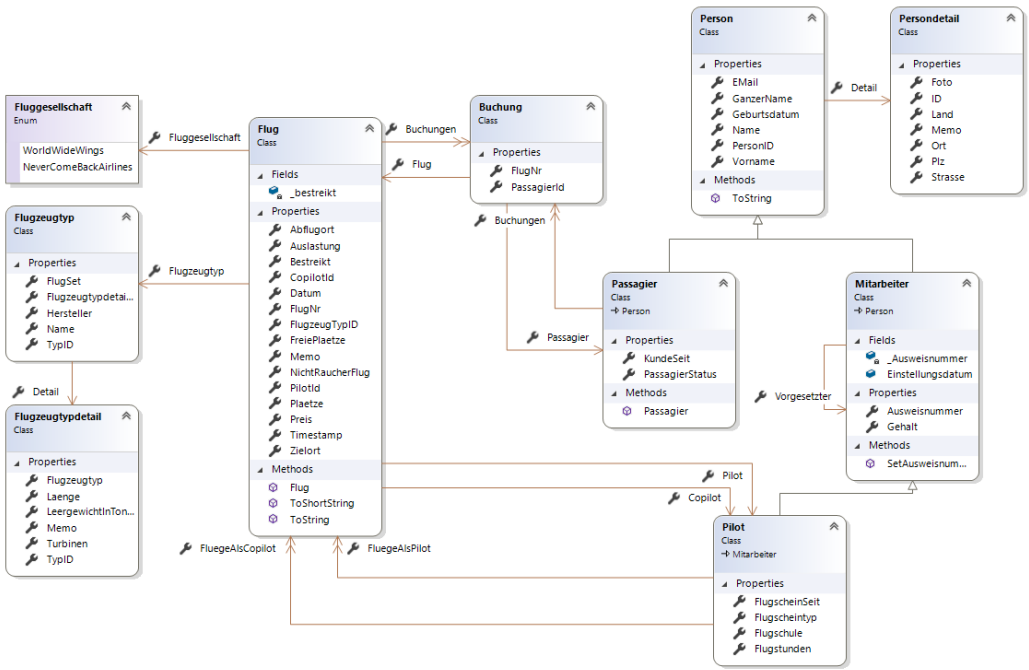

Abbildung 4: Objektmodell zum World Wide Wings-Datenmodell in der komplexeren Version 2

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.

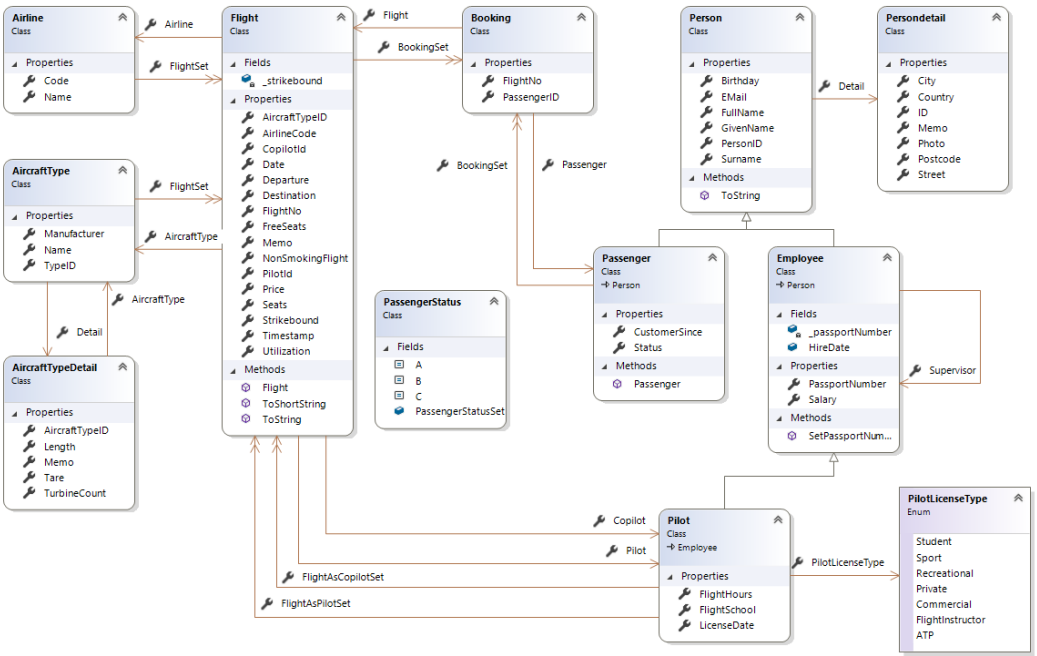

Abbildung: Englische Fassung des Objektmodells zum World Wide Wings-Datenmodell in der komplexeren Version 2

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 ITVisions.UI;
using System.Diagnostics;

namespace ITVisions
{
 /// <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.ToLongTimeString(), 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 + "'>");
    }
    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
  }
}
```