Dr. Holger Schwichtenberg

Moderne Datenzugriffslösungen mit Entity Framework Core 2.0/2.1

Datenbankprogrammierung mit C# in .NET/.NET Core/Xamarin

Version 5.1 dieses Buchs

Stand 15.05.2018

www.IT-Visions.de®

Dr. Holger Schwichtenberg

Verlag:	www.IT-Visions.de, Fahrenberg 40b, D-45257 Essen
ISBN:	3934279-19-8
Sprachliche Korrektur:	Matthias Bloch, M.A.
Formatierung:	Matthias Bloch, M.A.
Bezugsquelle PDF:	https://leanpub.com/EntityFrameworkCore
Bezugsquelle Kindle:	https://www.amazon.de/dp/B07BLP6NBH

www.IT-Visions.de
Dr. Holger Schwichtenberg

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 im 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 Versionen 1.x und 2.x stellen zunächst eine in vielen Punkten noch unvollständige Grundversion dar, die in den Folgeversionen dann komplettiert wird. In der in Entwicklung befindlichen Version 2.1 schließt Microsoft einige Lücken. Dieses Buch behandelt bereits die Neuerungen der Version 2.1 soweit diese in der aktuellen ersten Preview-Version enthalten sind.

Dieses **inkrementelle Konzept** habe ich auch mit diesem Buch umgesetzt. Das Buch ist seit September 2016 in mehreren Versionen erschienen. Die vor Ihnen liegende Version 5.0 dieses Buchs beschreibt alle Kernaspekte und viele Tipps und Tricks sowie Praxisszenarien zu Entity Framework Core. 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:

- Kindle-E-Book von Amazon für 9,99 Euro:
  https://www.amazon.de/exec/obidos/ASIN/393427918X/itvisions-21
- PDF-E-Book von Leanpub für 39,00 Dollar:
  https://leanpub.com/EntityFrameworkCore2
- Gedruckt im Carl Hanser Verlag unter dem Titel "Effizienter Datenzugriff mit Entity Framework Core" für 42,00 Euro: *https://www.amazon.de/exec/obidos/ASIN/3446448985/itvisions-21*

Die vorliegende Version dieses Buchs kostet nur rund 40 Euro (Die Plattform *leanpub.com* weist die Preise in Dollar ohne Mehrwertsteuer aus. Der von mir gesetzte Preis von 39,00 Dollar zzgl. 19% Mehrwertsteuer ergibt aktuell 40,47 Euro). Käufer der Grundversion können **Updates** jeweils für 10,00 Dollar (zzgl. Mehrwertsteuer) erwerben (*https://leanpub.com/EntityFrameworkCore/c/update*). Sie erhalten eine Update-Nachricht über Leanpub (sofern Sie nicht gegenüber Leanpub erklären, dass Sie keine Benachrichtigungen wünschen). Später einsteigende Käufer zahlen entsprechend mehr für die dann aktuelle Version. 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. **Bitte beachten Sie, dass über den Verlagsstand des gedruckten Buchs der Carl Hanser-Verlag entscheidet. Um zu erfahren, welchen Stand das Buch hat, fragen Sie bitte bei dem Verlag an.**

Da solch niedrige 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.

Zudem möchte ich darauf hinweisen, dass ich natürlich keinen kostenfreien technischen Support zu den Inhalten dieses Buchs geben kann. Ich freue mich aber immer über konstruktives Feedback und Verbesserungsvorschläge. Bitte verwenden Sie dazu das Kontaktformular auf *www.dotnet-doktor.de*.

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.

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 **12Monkeys** ein.

Herzliche Grüße aus Essen, dem Herzen der Metropole Ruhrgebiet

Holger Schwichtenberg

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
- 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
  - 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.entwickler-lexikon.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: E-Mail **buero@IT-Visions.de** sowie Telefon **0201-64 95 90-0**

4 Über dieses Buch

4.1 Versionsgeschichte dieses Buchs

Die folgende Tabelle zeigt die Versionen, die von diesem Buch 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**	**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

Buchversion Datum Umfang	Leanpub.com-Preis für PDF	Amazon.de-Preis für gedruckte Ausgabe	Entity Framework Core-Version(en)	Bemerkung
				▪ 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

Buchversion Datum Umfang	Leanpub.com-Preis für PDF	Amazon.de-Preis für gedruckte Ausgabe	Entity Framework Core-Version(en)	Bemerkung
				▪ 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

Buchversion Datum Umfang	Leanpub.com-Preis für PDF	Amazon.de-Preis für gedruckte Ausgabe	Entity Framework Core-Version(en)	Bemerkung
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"

Buchversion Datum Umfang	Leanpub.com-Preis für PDF	Amazon.de-Preis für gedruckte Ausgabe	Entity Framework Core-Version(en)	Bemerkung
				▪ 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"

Buchversion Datum Umfang	Leanpub.com-Preis für PDF	Amazon.de-Preis für gedruckte Ausgabe	Entity Framework Core-Version(en)	Bemerkung
				▪ Formatierung der Listings nun kompakter, daher die verringerte Seitenzahl
4.4 02.03.2018 493 Seiten	39,00 Dollar	Nicht verfügbar	1.1.5 und 2.0.1	▪ Erweiterung des Kapitels: "Datenbankschemamigrationen" ▪ Neues Kapitel "Praxislösungen/ Continous Integration und Continous Delivery
5.0 20.3.2018 522 Seiten	44,00 Dollar	Nicht verfügbar	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
5.1 522 Seiten	44,00 Dollar	Nicht verfügbar	1.1.5, 2.0.2 und 2.1 RC 1	▪ Aktualisert auf 2.1 RC1

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/EntityFrameworkCore/c/update2

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 als Kindle 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.

- Auditing mit Entity Framework Plus
- Connection Resiliency / EnableRetryOnFailure (seit Entity Framework Core 1.1)
- Data Seeding und Spaltensortierung beim Anlegen einer Tabelle (ab Entity Framework Core 2.1)
- Include für abgeleitete Typen (ab Entity Framework Core 2.1)
- Mapping mit (privaten) Feldern (seit Entity Framework Core 1.1)
- Migration von Entity Framework 6.x
- Parameter in Konstruktoren von Entitätsklassen (ab Entity Framework Core 2.1)
- Owned Types (ab Entity Framework Core 2.0)
- Scalare Datenbankfunktionen nutzen (seit Entity Framework Core 2.0)
- SQL Server memory-optimized Tables (seit Entity Framework Core 1.1)
- Temporale Tabellen (seit SQL Server 2016)
- Zusätzliche Erweiterungen wie z.B. EntityFrameworkCore.Rx, EFDetached.EntityFramework, EntityFrameworkCore.Triggers, EntityFrameworkCore.PrimaryKey und EntityFrameworkCore.TypedOriginalValues

Folgende Themen in diesem Buch sollen in Zukunft erweitert werden:

- Entity Framework Core in verteilten Systemen / Einsatz mit Webservices
- Reload

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 6.6, 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 7.0, 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 6.6 (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.

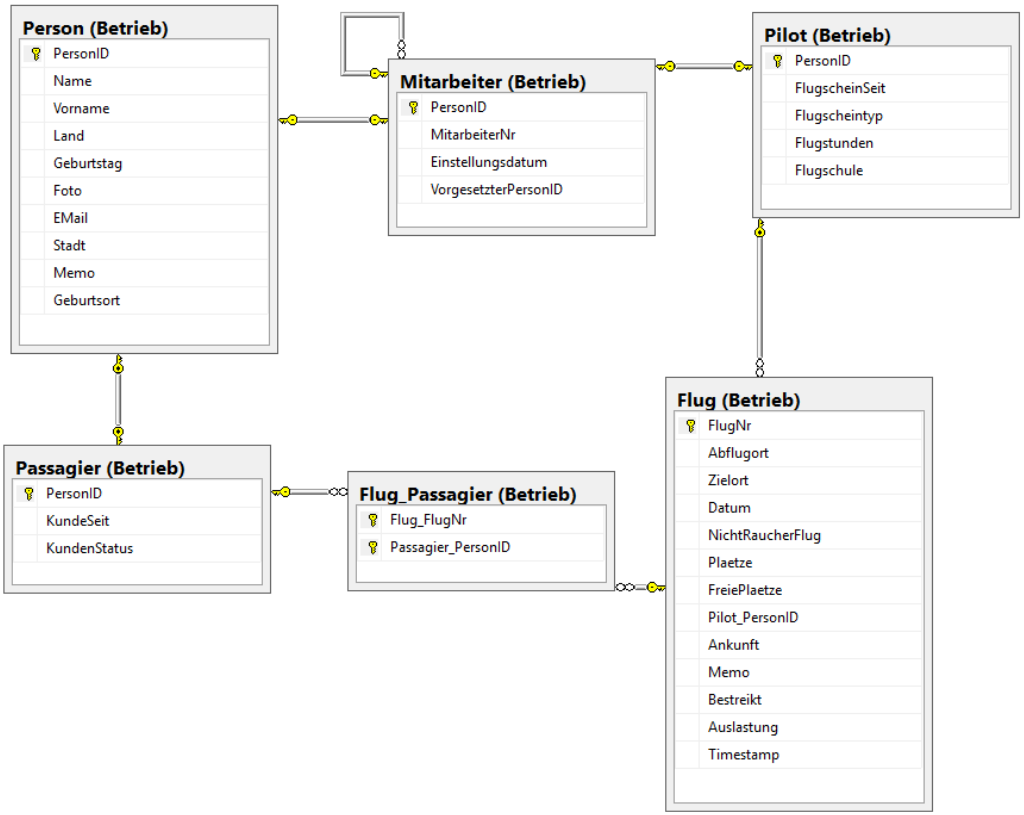

Abbildung 1: World Wide Wings-Datenmodell in der einfacheren Version 6.6

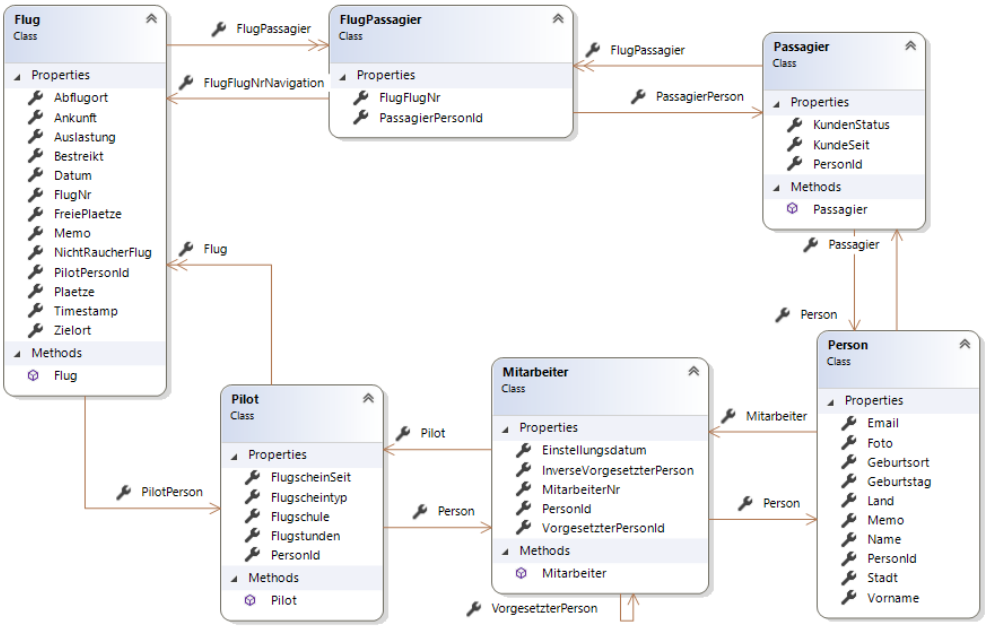

Abbildung 2: Objektmodell zum World Wide Wings-Datenmodell in der einfacheren Version 6.6

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.

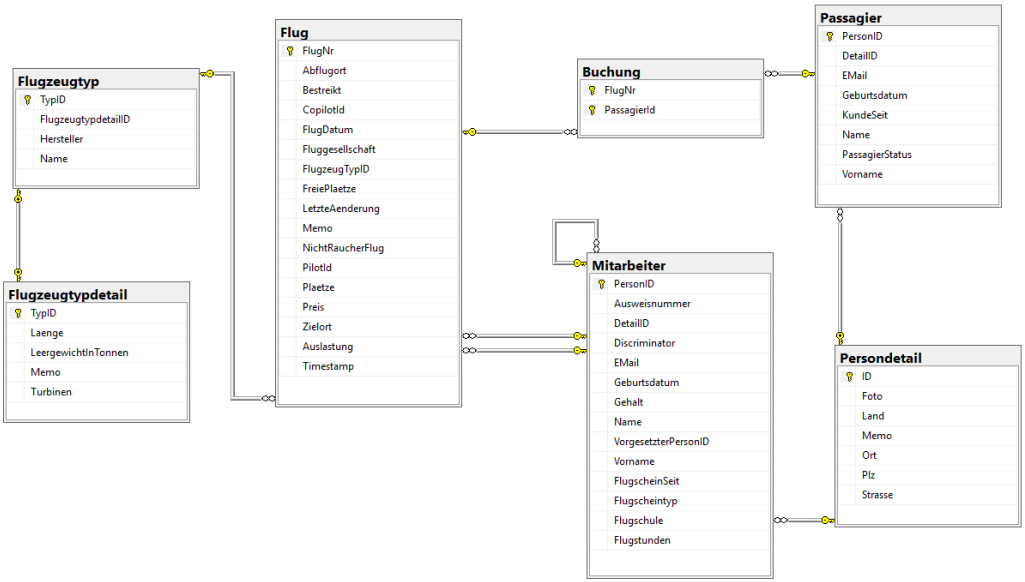

Abbildung 3: World Wide Wings-Datenmodell in der komplexeren Version 7.0

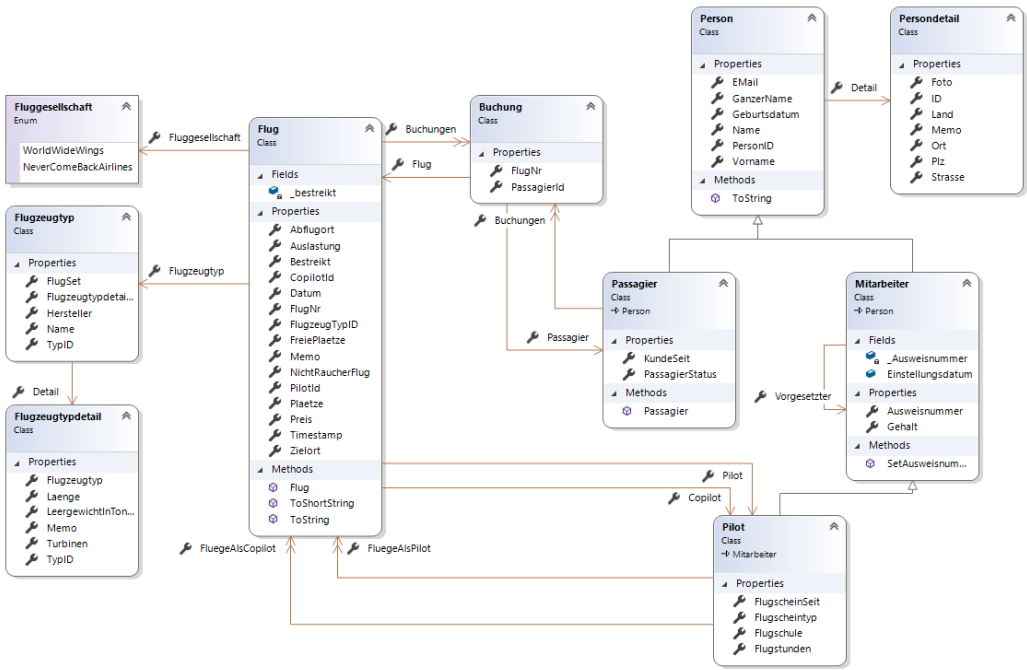

Abbildung 4: Objektmodell zum World Wide Wings-Datenmodell in der komplexeren Version 7.0

5.2 Englische Version des Beispiels

Da dieses Buch bald auch in Englisch (bei APress) erscheinen wird, 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.

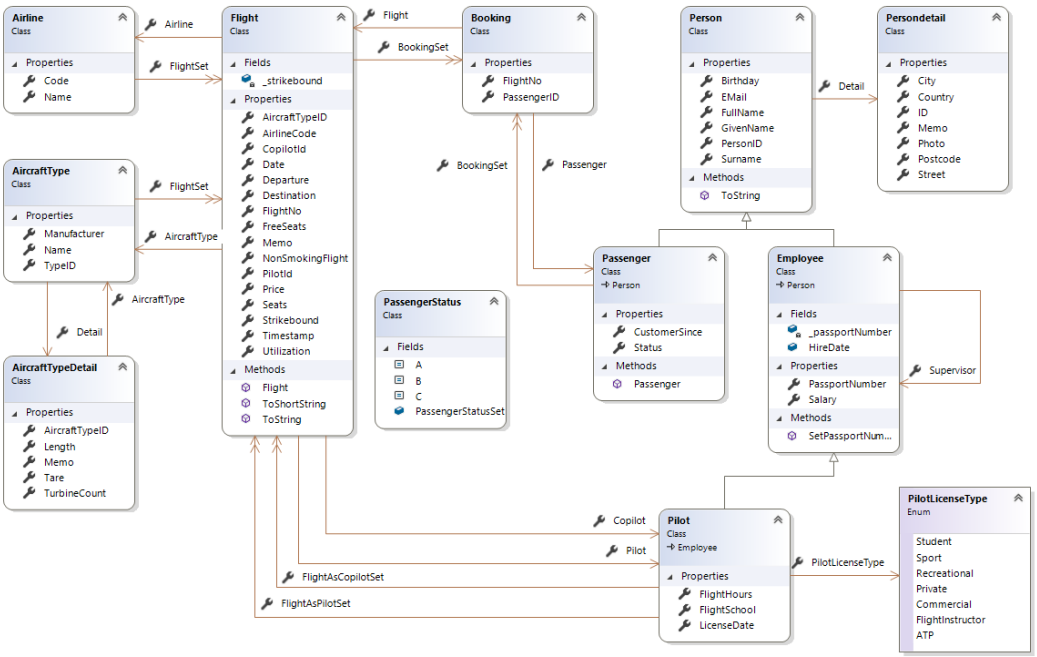

Abbildung: Englische Fassung des Objektmodells zum World Wide Wings-Datenmodell in der komplexeren Version 7.0

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 WPF, Windows Forms, ASP.NET Webforms oder ASP.NET MVC 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
  }
}
```

6 Programmcodebeispiel zum Download

Die Beispiele zu diesem Buch können Sie als Visual Studio-Projekte herunterladen auf der Leser-Website unter *www.IT-Visions.de/Leser*. Dort müssen Sie sich einmalig registrieren. Bei der Registrierung wird ein **Losungswort** abgefragt, das Sie als Käufer dieses Buchs ausweist. Bitte geben Sie dort **12Monkeys** ein. Durch die Registrierung erhalten Sie ein persönliches Kennwort per E-Mail zugesendet, das Sie dann für die Anmeldung nutzen können.

Bitte beachten Sie, dass nicht jede einzelne Zeile Programmcode, die Sie in diesem Buch finden, in den herunterladbaren Projekten enthalten sein kann. Die Projekte bilden funktionierende Lösungen. In diesem Buch warden auch alternative Lösungen für Einzelfälle diskutiert, die nicht unbedingt zu einer Gesamtlösung passen.

Dateiname	**Inhalt**
EFC_Reverse.rar	Beispiel aus dem Kapitel "Reverse Engineering". Das Paket enthält auch die SQL-Skripte für das einfachere "World Wide Wings"-Datenmodell inkl. Testdaten.
EFC_Forward.rar	"World Wide Wings"-Beispiel aus dem Kapitel "Forward Engineering"
EFC_Hauptbeispielsammlung.rar	Beispiele aller anderen Kapitel, die auf einer erweiterten Variante des "World Wide Wings"-Beispiels aus dem Kapitel "Forward Engineering" basieren. Hierin enthalten ist das Objektmodell, aus dem das komplexere Datenmodell zur Entwicklungs- oder Laufzeit angelegt werden kann. Testdaten lassen sich durch einen Datengenerator generieren, der im Quellcode enthalten ist.
EFC_UWP_SQLite.rar und EFC_Xamarin_SQLite.rar	Beispielanwendung: Einfacher Merkzettel "MiracleList Light" als Universal App für Windows 10 und Cross-Platform-App für iOS, Android und Windows 10. Die App speichert Daten mit Hilfe von Entity Framework Core in SQLite. 
EFC_Countries_NMSelf.rar	Ländergrenzen-Beispiel aus dem Kapitel "Praxislösungen"

7 Was ist Entity Framework Core?

Entity Framework Core ist ein Objekt-Relationaler Mapper (ORM) für .NET (.NET Framework, .NET Core, Mono und Xamarin). Entity Framework Core ist eine Neuimplementierung des "ADO.NET Entity Framework".

Zusammen mit .NET Core Version 1.0 und ASP.NET Core Version 1.0 ist auch Entity Framework Core Version 1.0 am 27. Juni 2016 erstmals erschienen. Die Version 2.0 ist am 14. August 2017 erschienen. Version 2.1 ist in Arbeit.

7.1 Was ist ein Objekt-Relationaler Mapper (ORM)?

In der Datenbankwelt sind relationale Datenbanken vorherschend, in der Programmierwelt sind es Objekte. Zwischen den beiden Welten gibt es erhebliche semantische und syntaktische Unterschiede, die man unter dem Begriff "Impedance Mismatch" (zu deutsch: Unverträglichkeit, vgl. [https://dict.leo.org/englisch-deutsch/impedance%20mismatch]) oder "Semantic Gap" (zu deutsch: semantische Lücke) zusammenfasst.

Kern des objektorientierten Programmierens (OOP) ist die Arbeit mit Objekten als Instanzen von Klassen im Hauptspeicher. Die meisten Anwendungen beinhalten dabei auch die Anforderung, in Objekten gespeicherte Daten dauerhaft zu speichern, insbesondere in Datenbanken. Grundsätzlich existieren objektorientierte Datenbanken (OODB), die direkt in der Lage sind, Objekte zu speichern. Allerdings haben objektorientierte Datenbanken bisher nur eine sehr geringe Verbreitung. Der vorherrschende Typus von Datenbanken sind relationale Datenbanken, die Datenstrukturen jedoch anders abbilden als Objektmodelle.

Um die Handhabung von relationalen Datenbanken in objektorientierten Systemen natürlicher zu gestalten, setzt die Software-Industrie seit Jahren auf O/R-Mapper (auch: OR-Mapper oder ORM geschrieben). O steht dabei für objektorientiert und R für relational. Diese Werkzeuge bilden demnach Konzepte aus der objektorientierten Welt, wie Klassen, Attribute oder Beziehungen zwischen Klassen, auf entsprechende Konstrukte der relationalen Welt, wie zum Beispiel Tabellen, Spalten und Fremdschlüssel, ab. Der Entwickler kann somit in der objektorientierten Welt verbleiben und den O/R-Mapper anweisen, bestimmte Objekte, welche in Form von Datensätzen in den Tabellen der relationalen Datenbank vorliegen, zu laden bzw. zu speichern. Wenig interessante und fehleranfällige Aufgaben wie das manuelle Erstellen von INSERT-, UPDATE- oder DELETE-Anweisungen übernimmt der O/R-Mapper hierbei ebenfalls, was zu einer weiteren Entlastung des Entwicklers führt.

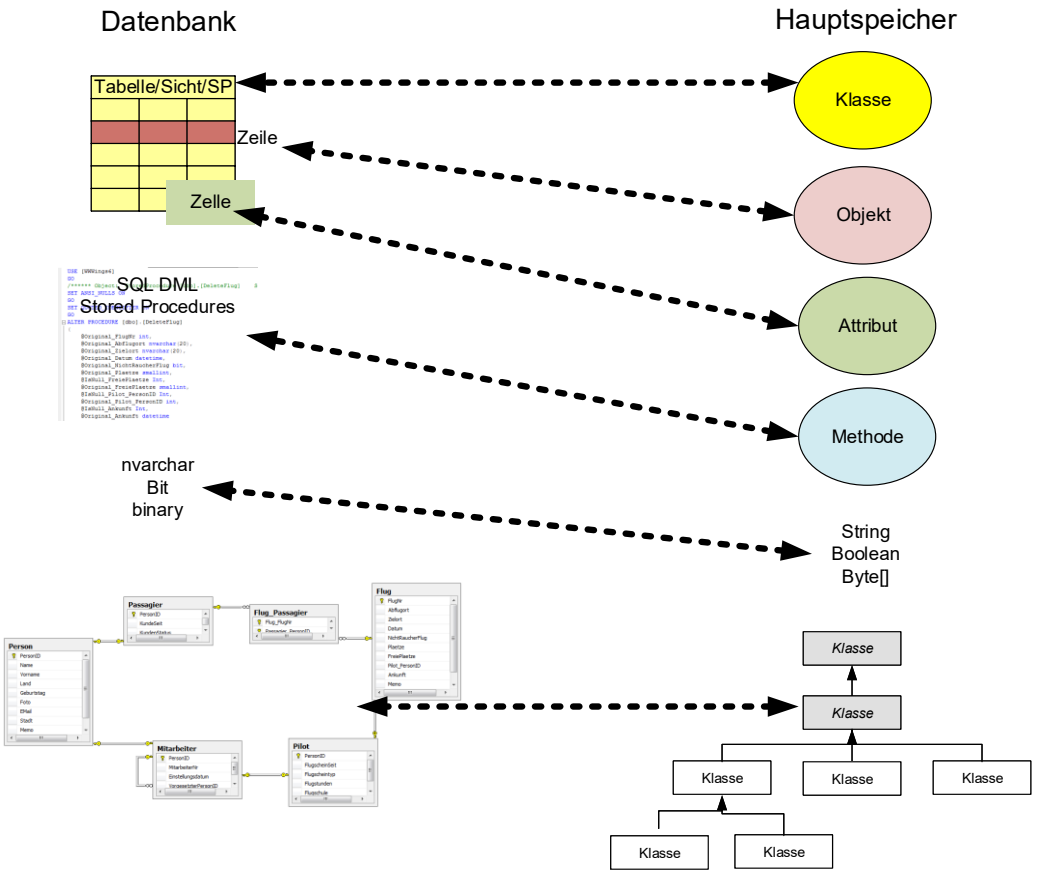

Abbildung: Beim ORM bildet man Konstrukte der OOP-Welt auf die relationale Welt ab.

Zwei besonders hervorstechende Unterschiede zwischen Objektmodell und Relationenmodell sind N:M-Beziehungen und Vererbung. Während man in einem Objektmodell eine N:M-Beziehung zwischen Objekten durch eine wechselseitige Objektmenge abbilden kann, benötigt man in der relationalen Datenbank eine Zwischentabelle. Vererbung kennen relationale Datenbanken gar nicht. Hier gibt es verschiedene Möglichkeiten der Nachbildung, doch dazu später mehr.

7.2 ORM in der .NET-Welt

Wenn ein .NET-Entwickler aus einer Datenbank mit einem DataReader oder DataSet Daten einliest, dann betreibt er noch kein OR Mapping. DataReader und DataSet sind zwar .NET-Objekte, aber diese verwalten nur Tabellenstrukturen. DataReader und DataSet sind aus der Sicht eines Objektmodells untypisierte, unspezifische Container. Erst wenn ein Entwickler spezifische Klassen für die in den Tabellen gespeicherten Strukturen definiert und die Inhalte aus DataSet oder DataReader in diese spezifischen Datenstrukturen umkopiert, betreibt er OR Mapping. Solch ein "händisches OR Mapping" ist für den Lesezugriff (gerade bei sehr breiten Tabellen) eine sehr aufwändige, mühselige und eintönige Programmierarbeit. Will man dann Änderungen in den Objekten auch noch wieder speichern, wird die Arbeit allerdings zur intellektuellen Herausforderung. Denn man muss erkennen können, welche Objekte verändert wurden, da man sonst ständig alle Daten aufs Neue speichert, was in Mehrbenutzerumgebungen ein Unding ist.

Während in der Java-Welt das ORM-Werkzeug schon sehr lange zu den etablierten Techniken gehört, hat Microsoft diesen Trend lange verschlafen bzw. es nicht vermocht, ein geeignetes Produkt zur Marktreife zu führen. ADO.NET in .NET 1.0 bis 3.5 enthielt keinen ORM, sondern beschränkte sich auf den direkten Datenzugriff und die Abbildung zwischen XML-Dokumenten und dem relationalen Modell.

Viele .NET-Entwickler haben sich daher daran gesetzt, diese Arbeit mit Hilfsbibliotheken und Werkzeugen zu vereinfachen. Dies war die Geburtsstunde einer großen Vielfalt von ORM-Werkzeugen für .NET. Dabei scheint es so, dass viele .NET-Entwickler das geflügelte Wort, dass ein Mann in seinem Leben einen Baum gepflanzt, ein Kind gezeugt und ein Haus gebaut haben sollte, um den Punkt "einen OR-Mapper geschrieben" ergänzt haben (wobei der Autor dieses Buchs sich davon auch nicht freisprechen kann, weil er ebenfalls einen OR-Mapper geschrieben hat). Anders ist die Vielfalt der ähnlichen Lösungen kaum erklärbar. Neben den öffentlich bekannten ORM-Werkzeugen für .NET findet man in den Unternehmen zahlreiche hauseigene Lösungen.

Bekannte öffentliche ORM für .NET von Drittanbietern (z.T. Open Source) sind:

- nHibernate
- Telerik Data Access (alias Open Access)
- Genome
- LLBLGen Pro
- Wilson
- Subsonic
- OBJ.NET
- .NET Data Objects (NDO)
- Dapper
- PetaPoco
- Massive
- Developer Express XPO

Neben den aktiven Entwicklern von ORM-Werkzeugen für .NET und den passiven Nutzern gibt eine noch größere Fraktion von Entwicklern, die ORM bisher nicht einsetzen. Meist herrscht Unwissenheit, die auch nicht aufgearbeitet wird, denn es herrscht das Motto "Wenn Microsoft es nicht macht, ist es auch nicht wichtig!"

Mit LINQ-to-SQL und dem ADO.NET Entity Framework sowie Entity Framework bietet Microsoft selbst jedoch inzwischen sogar drei verschiedene Produkte an. Der Softwarekonzern hat aber inzwischen verkündet, dass sich die Weiterentwicklungsbemühungen allein auf das Entity Framework Core konzentieren.

7.3 Versionsgeschichte von Entity Framework Core

Die folgende Abildung zeigt die Versionsgeschichte von Entity Framework Core.

Version	Downloads	Last updated
2.1.0-preview1-final	10,394	21 days ago
2.0.2 (current version)	27,043	6 days ago
2.0.1	427,223	4 months ago
2.0.0	1,576,040	7 months ago
2.0.0-preview2-final	28,130	9 months ago
2.0.0-preview1-final	63,738	5/10/2017
1.1.5	60,698	3 months ago
1.1.4	40,223	4 months ago
1.1.3	107,034	6 months ago
1.1.2	652,728	5/9/2017
1.1.1	514,952	3/6/2017
1.1.0	952,528	11/16/2016
1.1.0-preview1-final	20,205	10/24/2016
1.0.6	2,210	4 months ago
1.0.5	2,312	6 months ago
1.0.4	10,822	5/9/2017
1.0.3	130,072	3/6/2017
1.0.2	48,326	12/12/2016
1.0.1	325,351	9/13/2016
1.0.0	458,491	6/27/2016
1.0.0-rc2-final	74,838	5/16/2016

Abbildung: Entity Framework Core-Versionsgeschichte
[Quelle: https://www.nuget.org/packages/Microsoft.EntityFrameworkCore]

Versionsnummernänderungen an der dritten Stelle (z.B. 1.0.1 und 1.0.2) enthalten nur Fehlerbehebungen. Bei Versionsnummernänderungen an der zweiten Stelle sind auch neue Funktionen enthalten. In diesem Buch wird darauf hingewiesen, wenn eine Funktion besprochen wird, die eine bestimmte Versionsnummer voraussetzt.

HINWEIS: Die endgültige Version der Entity Framework Core-Werkzeuge für Entity Framework Core 1.x ist erst am 6.3.2017 im Rahmen von Entity Framework Core 1.1.1 und Visual Studio 2017 erschienen. Zuvor gab es nur "Preview"-Versionen. Seit Entity Framework Core 2.0 werden die Werkzeuge immer mit den neuen Produktreleases ausgeliefert.

7.4 Unterstützte Betriebssysteme

Genau wie die anderen Produkte der Core-Produktfamilie ist das Entity Framework Core (früherer Name: Entity Framework 7.0) ebenfalls plattformunabhängig. Die Core-Variante des etablierten Objekt-Relationalen Mappers läuft nicht nur auf dem .NET „Full“ Framework, sondern auch auf .NET Core und Mono inklusive Xamarin. Damit kann man Entity Framework Core auf Windows, Windows Phone/Mobile, Linux, MacOS, iOS und Android nutzen.

7.5 Unterstützte .NET-Versionen

Entity Framework Core 1.x läuft auf .NET Core 1.x, .NET Framework ab Version 4.5.1, Mono ab Version 4.6, Xamarin.iOS ab Version 10, Xamarin Android ab Version 7.0 und der Windows Univeral Platform (UWP).

Entity Framework Core 2.0 und 2.1 basieren auf .NET Standard 2.0 und setzen daher eine der folgenden .NET-Implementierungen voraus:

- .NET Core 2.0 (oder höher)
- .NET Framework 4.6.1 (oder höher)
- Mono 5.4 (oder höher)
- Xamarin.iOS 10.14 (oder höher)
- Xamarin.Mac 3.8 (oder höher)
- Xamarin.Android 7.5 (oder höher)
- Universal Windows Platform (UWP) 10.0.16299 (oder höher)

.NET Standard	1.0	1.1	1.2	1.3	1.4	1.5	1.6	2.0
.NET Core	1.0	1.0	1.0	1.0	1.0	1.0	1.0	2.0
.NET Framework (with .NET Core 1.x SDK)	4.5	4.5	4.5.1	4.6	4.6.1	4.6.2		
.NET Framework (with .NET Core 2.0 SDK)	4.5	4.5	4.5.1	4.6	4.6.1	4.6.1	4.6.1	4.6.1
Mono	4.6	4.6	4.6	4.6	4.6	4.6	4.6	5.4
Xamarin.iOS	10.0	10.0	10.0	10.0	10.0	10.0	10.0	10.14
Xamarin.Mac	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.8
Xamarin.Android	7.0	7.0	7.0	7.0	7.0	7.0	7.0	8.0
Universal Windows Platform	10.0	10.0	10.0	10.0	10.0	10.0.16299	10.0.16299	10.0.16299
Windows	8.0	8.0	8.1					
Windows Phone	8.1	8.1	8.1					
Windows Phone Silverlight	8.0							

Abbildung: Implementierungen von .NET Standard
[Quelle: https://docs.microsoft.com/de-de/dotnet/standard/library]

HINWEIS: Microsoft begründet die Beschränkung auf .NET Standard in Entity Framework Core 2.0/2.1 in [https://github.com/aspnet/Announcements/issues/246]. Unter anderem kann dadurch die Größe der Nuget-Pakete deutlich reduziert werden.

7.6 Unterstützte Visual Studio-Versionen

Für die Nutzung von Entity Framework Core 2.0/2.1 benötigt man zwingend Visual Studio 2017 Update 3 oder höher, auch wenn man mit dem klassischen .NET Framework programmiert, da Visual Studio nur mit diesem Update .NET Standard 2.0 kennt und versteht, dass .NET Framework 4.6.1 und höher Implementierungen von .NET Standard 2.0 sind.

Wenn man für .NET Core programmiert, benötigt man für Entity Framework Core 1.x Visual Studio 2017, (die Werkzeuge für Visual Studio 2015 sind veraltert und werden von Microsoft nicht mehr aktualisiert). Für Entity Framework Core 1.x in Verbindung mit dem klassischen .NET Framework reicht auch eine ältere Visual Studio-Version.

7.7 Unterstützte Datenbanken

Die folgende Tabelle zeigt die von Entity Framework Core durch Microsoft (SQL Server, SQL Compact und SQLite von Microsoft) und Drittanbieter (PostgreSQL, DB2, Oracle, MySQL u.a.) unterstützten Datenbankmanagementsysteme.

Auf Mobilgeräten mit Xamarin bzw. im Rahmen von Windows 10 Universal Platform Apps konnte Entity Framework Core 1.x nur lokale Datenbanken (SQLite) ansprechen. Mit der Einführung von

.NET Standard 2.0 steht nun der Microsoft SQL Server-Client auch auf Xamarin und der Windows 10 Universal Platform (ab dem Herbst 2017 Creators Update) zur Verfügung.

Die geplante Unterstützung für NoSQL-Datenbanken wie Redis und Azure Table Storage ist in Version 1.x/2.x von Entity Framework Core noch nicht enthalten. Es gibt aber für MongoDB ein Entwicklungsprojekt auf Github [https://github.com/crhairr/EntityFrameworkCore.MongoDb].

Datenbank	**Anbieter / Preis**	**URL**
Microsoft SQL Server	Microsoft / kostenfrei	*www.nuget.org/packages/Microsoft.EntityFrameworkCore.SqlServer*
Microsoft SQL Server Compact 3.5	Microsoft / kostenfrei	*www.nuget.org/packages/EntityFrameworkCore.SqlServerCompact35*
Microsoft SQL Server Compact 4.0	Microsoft / kostenfrei	*www.nuget.org/packages/EntityFrameworkCore.SqlServerCompact40*
SQLite	Microsoft / kostenfrei	*www.nuget.org/packages/Microsoft.EntityFrameworkCore.Sqlite*
In-Memory	Microsoft / kostenfrei	*www.nuget.org/packages/Microsoft.EntityFrameworkCore.InMemory*
MySQL	Oracle / kostenfrei	*www.nuget.org/packages/MySQL.Data.EntityFrameworkCore*
PostgreSQL	Open Source-Team npgsql.org / kostenfrei	*www.nuget.org/packages/Npgsql.EntityFrameworkCore.PostgreSQL*
DB2	IBM / kostenfrei	*www.nuget.org/packages/EntityFramework.IBMDataServer*
MySQL, Oracle, PostgreSQL, SQLite, DB2, Salesforce, Dynamics CRM, SugarCRM, Zoho CRM, QuickBooks, FreshBooks, MailChimp, ExactTarget, Bigcommerce, Magento	Devart / kostenpflichtig (99 bis 299 Dollar pro Treiberart)	*www.devart.com/purchase.html#dotConnect*

Tabelle: Verfügbare Datenbanktreiber für Entity Framework Core

ACHTUNG: Aufgrund von "Breaking Changes" in den Provider-Schnittstellen, sind die Provider für Entity Framework Core 1.x nicht kompatibel zu Entity Framework Core 2.0/2.1. Man benötigt also für die Version 2.0 neue Provider! Die Treiber für Entity Framework Core

2.0 laufen auch in Entity Framework Core 2.1, einzelne neue Features (z.B. Value Converter) brauchen aber eine neue Version des Treibers.

7.8 Funktionsumfang von Entity Framework Core

Die Abbildung visualisiert, dass Entity Framework Core (gelb) gegenüber dem bisherigen Entity Framework (blau, aktuelle Version 6.x) einige neue Funktionen enthält (Bereich, der nur gelb, aber nicht blau ist). Es gibt aber auch einige Bereiche, die nur blau und nicht gelb sind: Das sind die Funktionen, die in Entity Framework 6.x enthalten sind, aber nicht in Entity Framework Core 1.x/2.0. Microsoft wird einige Funktionen davon in den kommenden Versionen von Entity Framework Core nachrüsten, andere Funktionen werden für immer entfallen.

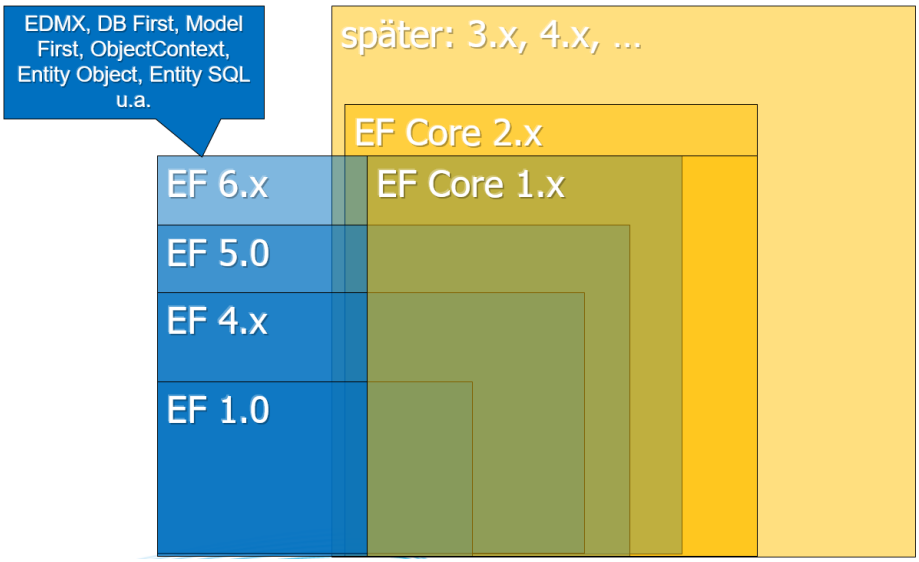

Abbildung: Funktionsumfang des bisherigen Entity Framework im Vergleich zu Entity Framework Core. Links zeigt eine Sprechblase einige Features, die dauerhaft entfallen sind.

7.9 Funktionen, die dauerhaft entfallen

Folgende Funktionen aus dem klassischen Entity Framework hat Microsoft grundsätzlich in Entity Framework Core gestrichen:

- Die Vorgehensweise Database First und Model First. in Entity Framework Core gibt es nur noch das Code-based Modelling (früher Code First), mit dem man sowohl Programmcode aus Datenbanken erzeugen kann (Reverse Engineering) als auch Datenbanken aus Programmcode (Forward Engineering).
- Das Entity Data Model (EDM) und die XML-Repräsentation davon (EDMX) entfallen. Bisher wurde auch beim Code First intern ein EDM im RAM erzeugt. Der Overhead entfällt.
- Die Basisklasse ObjectContext für den Entity Framework-Kontext entfällt. Es gibt nur noch die Basisklasse DbContext. DbContext ist jetzt in Entity Framework Core kein Wrapper um ObjectContext mehr, sondern eine komplett neue, eigenständige Implementierung.
- Die Basisklasse EntityObject für Entitätsklassen entfällt. Die Entitätsklassen sind nun immer Plain Old CLR Objects (POCOs).

- Auch die Abfragesprache Entity SQL (ESQL) entfällt. Es gibt nur noch Unterstützung für LINQ, SQL und Stored Procedures (SPs) sowie Table Valued Functions (TVFs).
- Automatische Schemamigrationen werden nicht mehr angeboten. Schemamigrationen inklusive der Ersterstellung eines Datenbankschemas sind nun zur Entwicklungszeit immer manuell auszuführen. Zur Laufzeit kann eine Migration weiterhin beim ersten Zugriff auf die Datenbank erfolgen.
- Einige Szenarien des komplexeren Mappings zwischen Tabellen und Typen entfallen. Dazu gehört das Multiple Entity Sets per Type (MEST, verschiedene Tabellen auf dieselbe Entität abbilden) und das Kombinieren der Strategien Table per Hierarchy (TPH), Table per Type (TPT) und Table per Concrete Type (TPC) in einer Vererbungshierarchie.

7.10 Funktionen, die Microsoft bald nachrüsten will

In der Roadmap für Entity Framework-Core [*https://github.com/aspnet/EntityFramework/wiki/Roadmap*] dokumentiert Microsoft-Entwickler Rowan Miller, welche Features in Entity Framework-Core fehlen, die man „bald“ nachrüsten will. Dabei ist dies nicht mit einem konkreten Zeitplan hinterlegt. Bemerkenswert ist, dass Microsoft einige dieser Funktionen selbst als „kritisch“ bezeichnet. Zu diesen "kritischen" fehlenden Funktionen gehören:

- Entity Framework Core unterstützt nur den Zugriff auf Tabellen, nicht aber auf Views (Sichten) in der Datenbank. Man kann Views nur nutzen, wenn man den View sowie den Programmcode manuell erstellt und den View wie eine Tabelle behandelt. → **In Entity Framework Core 2.1 ist dies eleganter möglich.**
- Stored Procedures können bisher nur zum Abfragen von Daten (SELECT), nicht aber zum Einfügen (INSERT), Aktualisieren (UPDATE) und Löschen (DELETE) verwendet werden.
- Einige LINQ-Befehle werden derzeit nicht in der Datenbank, sondern im RAM ausgeführt. Dazu gehört auch der GroupBy-Operator, d.h. bei allen Gruppierungen werden alle Datensätze aus der Datenbank ins RAM gelesen und dort gruppiert, was bei allen Tabellen (außer sehr kleinen) zu einer katastrophalen Performance führt. → **In Entity Framework Core 2.1 wird GroupBy() in vielen Fällen in der Datenbank ausgeführt.**
- Es gibt weder ein automatisches Lazy Loading noch ein explizites Nachladen im Entity Framework Core-API. Aktuell kann der Entwickler verbundene Datensätze nur direkt mitladen (Eager Loading) oder mit separaten Befehlen nachladen. → **In Entity Framework Core 2.1 wird Lazy Loading unterstützt.**
- Direktes SQL und Stored Procedures können nur genutzt werden, wenn sie Entitätstypen zurückliefern. Andere Typen werden bisher nicht unterstützt. → **In Entity Framework Core 2.1 wird dies unterstützt.**
- Reverse Engineering bestehender Datenbanken kann man bisher nur von der Kommandozeile bzw. der Nuget-Konsole in Visual Studio starten. Den GUI-basierten Assistenten gibt es nicht mehr.
- Es gibt auch kein „Update Model from Database“ für bestehende Datenbanken, d.h. nach einem Reverse Engineering einer Datenbank muss der Entwickler Datenbankschemaänderungen im Objektmodell manuell nachtragen oder das ganze Objektmodell neu generieren. Diese Funktion gab es aber auch bisher schon bei Code First nicht, sondern nur bei Database First.

- Komplexe Typen (Complex Types), also Klassen, die keine eigene Entität, sondern Teil einer anderen Entität darstellen, gibt es nicht.

7.11 Hohe Priorität, aber nicht kritisch

In einer zweiten Liste nennt Microsoft weitere Funktionen, die sie nicht als kritisch ansehen, die aber dennoch „hohe Priorität“ haben:

- Es gibt bisher keine grafische Visualisierung eines Objektmodells, wie das bislang bei EDMX möglich war.
- Einige der bisher vorhandenen Typkonvertierungen, z.B. zwischen XML und String, gibt es noch nicht. → **Entity Framework Core 2.1 unterstützt Typkonvertierungen.**
- Die Geo-Datentypen Geography und Geometry von Microsoft SQL Server werden bisher nicht unterstützt.
- Entity Framework Core unterstützt keine N:M-Abbildungen: Bisher muss der Entwickler dies mit zwei 1:N-Abbildung und einer Zwischenentität analog zur Zwischentabelle in der Datenbank nachbilden.
- Table per Type wird bislang nicht als Vererbungsstrategie unterstützt. Entity Framework Core verwendet TPH, wenn es für die Basisklasse ein DBSet<T> gibt, sonst TPC. TPC kann man nicht explizit konfigurieren.
- Das Befüllen der Datenbank mit Daten im Rahmen der Migration (Seed()-Funktion) ist nicht möglich. → **Entity Framework Core 2.1 unterstützt dies.**
- Die mit Entity Framework 6.0 eingeführten Command Interceptors, mit denen ein Softwareentwickler von Entity Framework zur Datenbank gesendete Befehle vor und nach der Ausführung in der Datenbank beeinflussen kann, gibt es noch nicht.

Einige Punkte auf dieser High Priority-Liste von Microsoft sind zudem auch neue Features, die Entity Framework 6.x selbst (noch) gar nicht beherrscht:

- Festlegung von Bedingungen für mitzuladene Datensätze beim Eager Loading (Eager Loading Rules)
- Unterstützung für E-Tags.
- Unterstützung für Nicht-Relationale Datenspeicher ("NoSQL") wie Azure Table Storage und Redis. → **CosmosDB ist für Entity Framework Core 2.1 in Arbeit.**

Diese Priorisierung stammt aus der Sicht von Microsoft. Der Autor dieses Buchs würde auf Basis seiner Praxiserfahrung einige Punkte anders priorisieren, zum Beispiel die N:M-Abbildung als „kritisch“ hochstufen: Eine Nachbildung von N:M durch zwei 1:N-Beziehungen im Objektmodell ist zwar möglich, macht aber den Programmcode komplexer. Die Migration von bestehenden Entity Framework-Lösungen zu Entity Framework Core wird damit sehr erschwert.

Das gilt auch für die fehlende Unterstützung von Table per Type-Vererbung: Auch hier muss bestehender Programmcode umfangreich geändert werden. Und auch für neue Anwendungen mit einem neuen Datenbankschema und Forward Engineering gibt es ein Problem: Wenn die Vererbung erst mal mit TPH oder TPC realisiert ist, muss man aufwändig die Daten im Datenbankschema umschichten, wenn man später doch auf TPH setzen will.

Außerdem fehlen in Microsofts Listen auch Features wie etwa die Validierung von Entitäten, die unnötige Roundtrips zur Datenbank ersparen kann, wenn schon im RAM klar ist, dass die Entität die erforderlichen Bedingungen nicht erfüllt.

7.12 Neue Funktionen in Entity Framework Core

Entity Framework Core kann insbesondere mit folgenden Vorteilen gegenüber dem Vorgänger auftrumpfen:

- Entity Framework Core läuft nicht nur in Windows, Linux und MacOS, sondern auch auf Mobilgeräten mit Windows 10, iOS und Android. Auf den Mobilgeräten ist freilich lediglich ein Zugriff auf lokale Datenbanken (z.B. SQLite) vorgesehen. In Windows 10 Universal Apps kann man seit Windows 10 Version 1709 aber auch lokale und entferne Microsoft SQL Server ansprechen mit Entity Framework Core.
- Entity Framework Core bietet eine höhere Ausführungsgeschwindigkeit – insbesondere beim Datenlesen (dabei wird fast die Leistung wie beim handgeschriebenen Umkopieren von Daten aus einem DataReader-Objekt in ein typisiertes .NET-Objekt erreicht).
- Projektionen mit Select() können nun direkt auf Entitätsklassen abgebildet werden. Der Umweg über anonyme .NET-Objekte ist nicht mehr notwendig.
- Per "Batching" fasst Entity Framework Core nun INSERT-, DELETE- und UPDATE-Operationen zu einem Rundgang zum Datenbankmanagementsystem zusammen, statt jeden Befehl einzeln zu senden.
- Standardwerte für Spalten in der Datenbank werden nun sowohl beim Reverse Engineering als auch beim Forward Engineering unterstützt.
- Zur Schlüsselgenerierung sind neben den klassischen Autowerten nun auch neuere Verfahren wie Sequenzen erlaubt.
- Als "Shadow Properties" bezeichnet Entity Framework Core den jetzt möglichen Zugriff auf Spalten der Datenbanktabelle, für die es kein Attribut in der Klasse gibt.
- Mit globalen Filtern können Entwickler Bedingungen festlegen, die automatisch bei jeder Abfrage angewendet werden.
- Mit Value Convertern kann man Werte aus dem Objekt beim Speichern in die Datenbank bzw. beim Laden aus der Datenbank in einen anderen Datentyp konvertieren.

7.13 Einsatzszenarien für Entity Framework Core

Angesichts dieser langen Liste von fehlenden Funktionen stellt sich die Frage, ob und wofür Entity Framework Core in der Version 1.x/2.0 überhaupt zu gebrauchen ist.

Das Haupteinsatzgebiet liegt auf den Plattformen, wo Entity Framework bisher gar nicht lief: Windows Phone/Mobile, Android, iOS, Linux und MacOS.

- Universal Windows Platform (UWP) Apps und Xamarin Apps können nur Entity Framework Core verwenden, nicht das klassische Entity Framework.
- Wenn man eine neue ASP.NET Core-Webnwendung oer WebAPI entwickeln will und diese nicht auf .NET „Full“ Framework, sondern .NET Core basieren soll, führt kein Weg an Entity Framework Core vorbei, denn das bisherige Entity Framework 6.x läuft nicht auf .NET Core.

Allerdings gibtes für ASP.NET Core auch den Weg, als Basis das klassische .NET Framework 4.6.x/4.7.x zu verwenden, sodass man dann auch Entity Framework 6.x nutzen kann.

Ein Szenario, in dem der Einsatz von Entity Framework Core auf dem Webserver empfohlen werden kann, ist das Offline-Szenario, bei dem es auf dem Mobilgerät eine lokale Kopie der Serverdatenbank geben soll. In diesem Fall kann man auf dem Client und dem Server mit demselben Datenzugriffscode arbeiten: Der Client verwendet Entity Framework Core für den Zugriff auf SQLite und der Webserver denselben Entity Framework Core-Programmcode für den Zugriff auf einen Microsoft SQL Server (siehe folgende Abbildung).

Teilen der Datenzugriffsschicht zwischen Mobilgerät und Server mit Entity Framework Core

© Dr. Holger Schwichtenberg, www.IT-Visions.de 2016

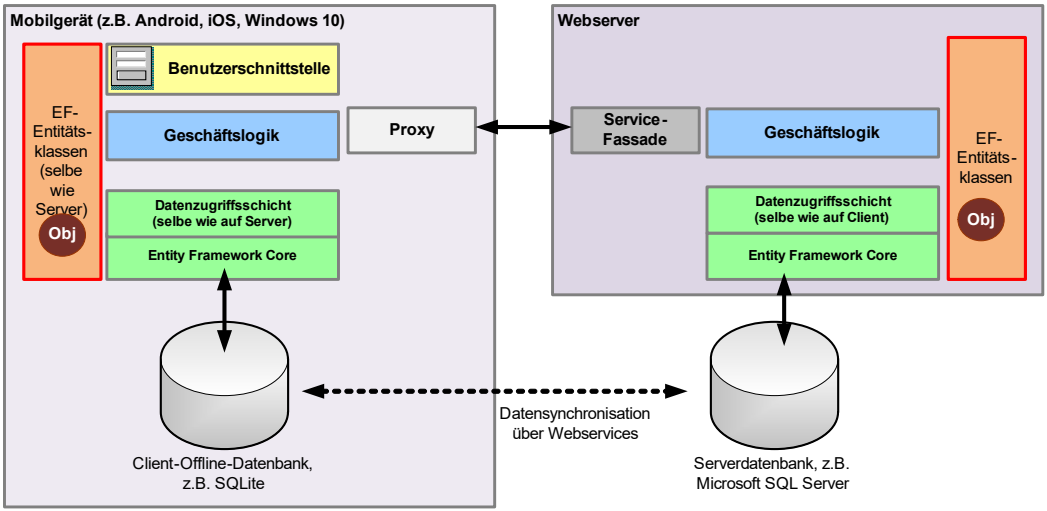

Abbildung: Teilen der Datenzugriffsschicht zwischen Mobilgerät und Webserver mit Entity Framework Core

Für Projekte auf anderen Plattformen gilt:

- Die Migration bestehenden Programmcodes von Entity Framework 6.x auf Entity Framework Core ist aufwändig. Man muss sich gut überlegen, ob die verbesserten Features und die höhere Leistung von Entity Framework Core den Aufwand rechtfertigen.
- Aber in neuen Projekten können Entwickler schon jetzt Entity Framework Core als performante Zukunftstechnik einsetzen und ggf. als Zwischenlösung für existierende Lücken parallel dort noch das bisherige Entity Framework nutzen.