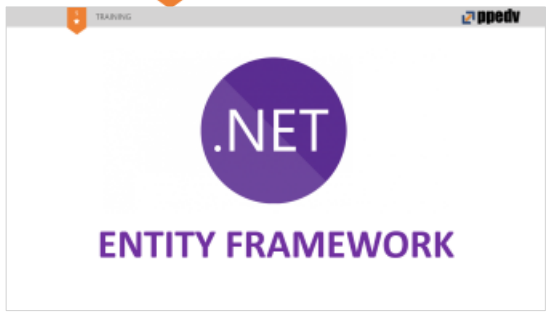


EF > INTRO

EF & ORM

EF > ENTITY DATA MODEL

EF > DBCONTEXT

EF > DBSET

EF > ENTITY TYPES

EF > MODELING CONCEPTS

EF > DB FIRST

EF & LAZY LOADING

EF > MIGRATIONS

EF > QUERING ENTITIES

EF > INHERITENCE

EF > FACTS

ENTITY FRAMEWORK

EF > INTRO

© ppedv AG

EF CORE > INTRO

- WAS IST EF CORE
  - Erweiterung von EF
  - ein ORM-Framework
  - aktuelle Version - 3.1 (03. Dez. 2019) #checkForUpdates

- WOZU IST EF CORE
  - um Datenzugriffscode zu vereinfachen

- ORM - objektrelationales Mapping - ist eine Technik, die das Abfragen und Ändern von Daten einer Datenbank mittels Objekt-Orientierten Paradigmen ermöglicht. Es vereinfacht den Datenzugriff.

© ppedv AG

EFCORE > PACKAGE #OVERVIEW

- Untersuchen Sie die Liste von Elementen im Package EntityFrameworkCore
  - https://docs.microsoft.com/de-de/dotnet/api/microsoft.entityframeworkcore?view=efcore-3.1

© ppedv AG

EF > SOME METHODS #OVERVIEW

- Microsoft.EntityFrameworkCore
  - hat unter anderem die Klasse EntityFrameworkQueryableExtensions
  - diese Klasse hat folgende Methoden:
    - AllAsync()
    - AnyAsync()
    - AsAsyncEnumerable()
    - AsNoTracking()
    - AsTracking()
    - AverageAsync()
    - ContainsAsync()
    - CountAsync()
    - FirstAsync()
    - FirstOrDefaultAsync()
    - ForEachAsync()
    - IgnoreQueryAsync()
    - Include()
    - LastAsync()
    - LastOrDefaultAsync()
    - Load()
    - LoadAsync()
    - LongCountAsync()
    - MaxAsync()
    - MinAsync()
    - SingleAsync()
    - SingleOrDefaultAsync()
    - SumAsync()
    - TagWith()
    - ThenInclude()
    - ToArrayAsync()
    - ToDictionaryAsync()
    - ToListAsync()

© ppedv AG

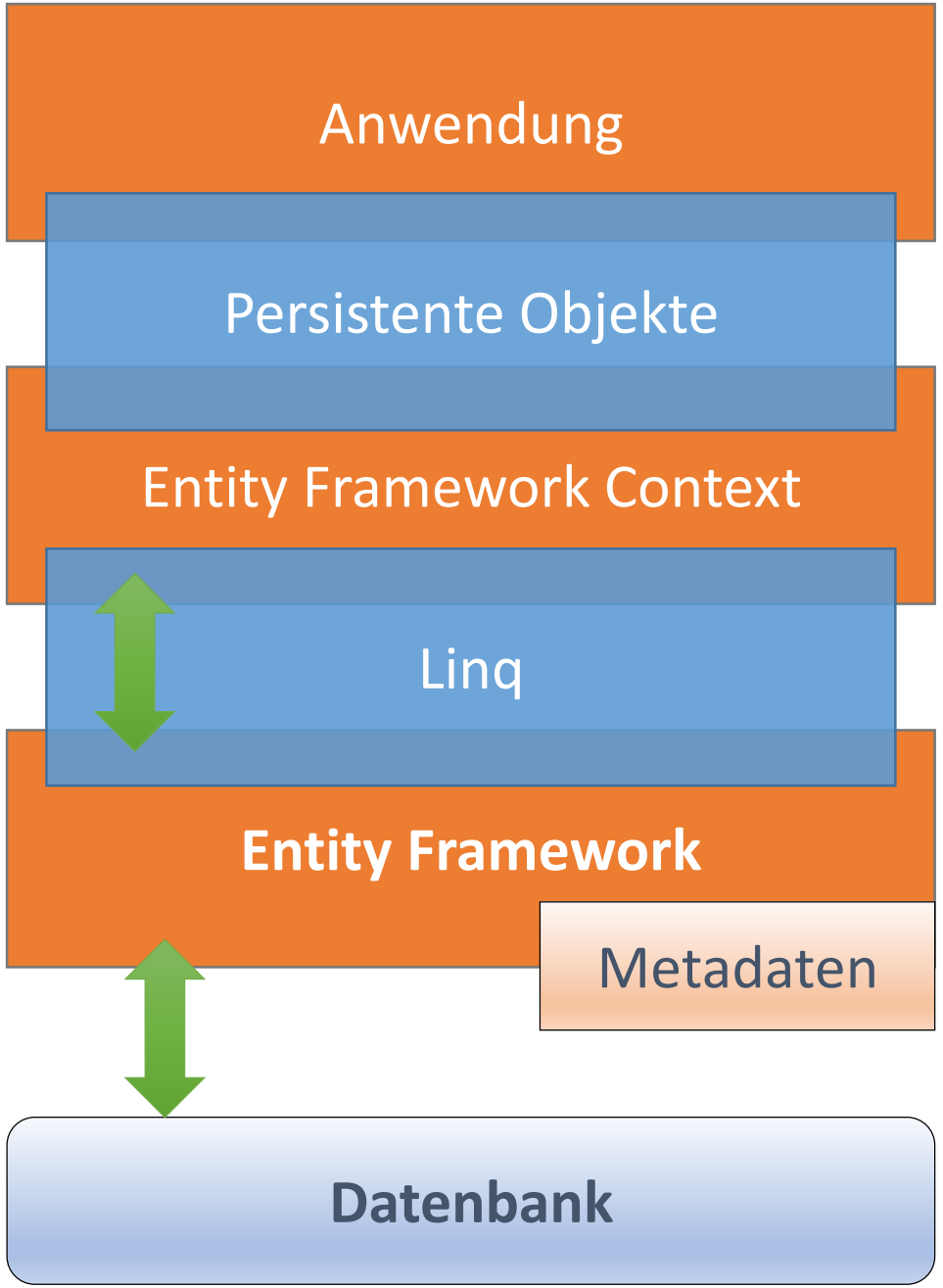
EFCORE > SUBPACKAGES #OVERVIEW

- Microsoft.EntityFrameworkCore. …
  - ChangeTracking
  - Design
  - Diagnostics
  - Infrastructure
  - Internal
  - Metadata
  - Migrations
  - Query
  - Scaffolding
  - SqlServer Microsoft SQL Server database provider for Entity Framework Core.
  - Storage
  - Update
  - ValueGeneration

© ppedv AG

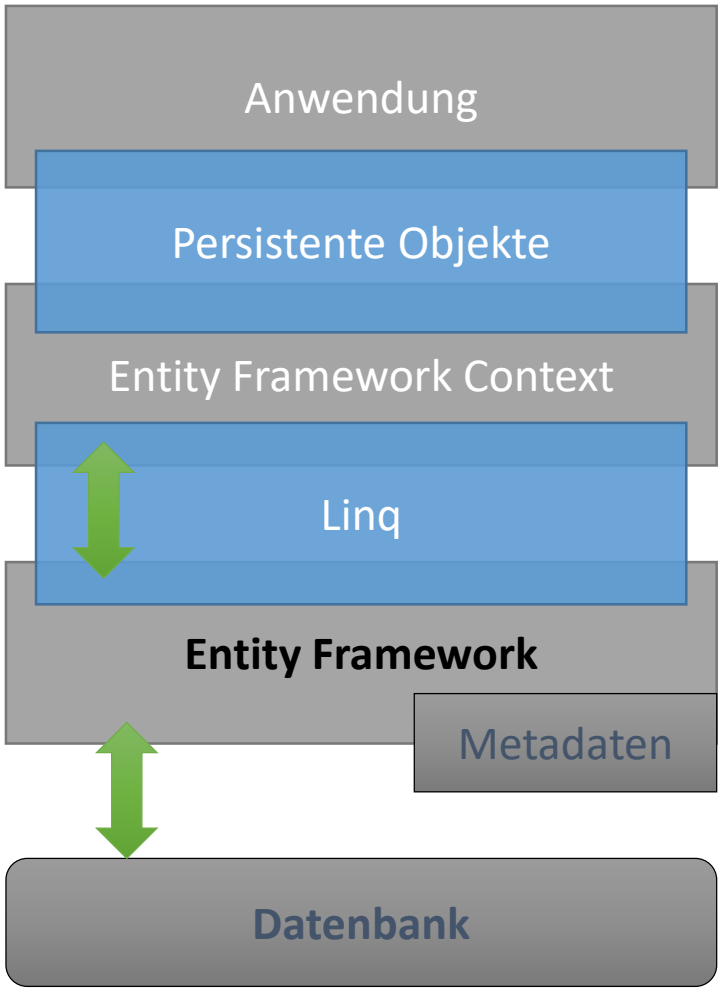
Entity Framework Core

- O/R Mapping Framework
- Leichtgewichtig, Erweiterbar, Cross-Plattform Version von EF6
- Übersetzt LINQ in SQL
- Ermöglicht es, Daten nur bei Bedarf zu laden (lazy loading)
- Unterstützt mehrere relationale Datenbanken (NoSQL noch nicht)
- Code First & Database First Ansatz



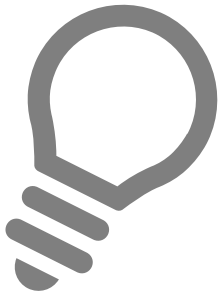
Web API und HTTP

- **Exkurs Entity Framework**
  - O/R Mapping Framework
  - Übersetzt LINQ in SQL
  - Abstrahiert ADO.NET
  - Ermöglicht es, Daten nur bei Bedarf zu laden *(lazy loading)*
  - Der Einsatz von O/R Mappern spart viel Zeit!



Entity Framework Core Schlüsselbegriffe

- **Data Annotations** = definieren Primary und Foreign Keys, Required Fields, … in Entity Klassen
- **DbContext** = eine Session mit einer Datenbank und ermöglicht das Abfragen und Speichern von Instanzen von Entitäten
- **Migrations** = Versionierung des Datenbankcodings
- **Seeding** = DB mit Demodaten füllen
- Environment Variablen für geheime Daten verwenden



© ppedv AG

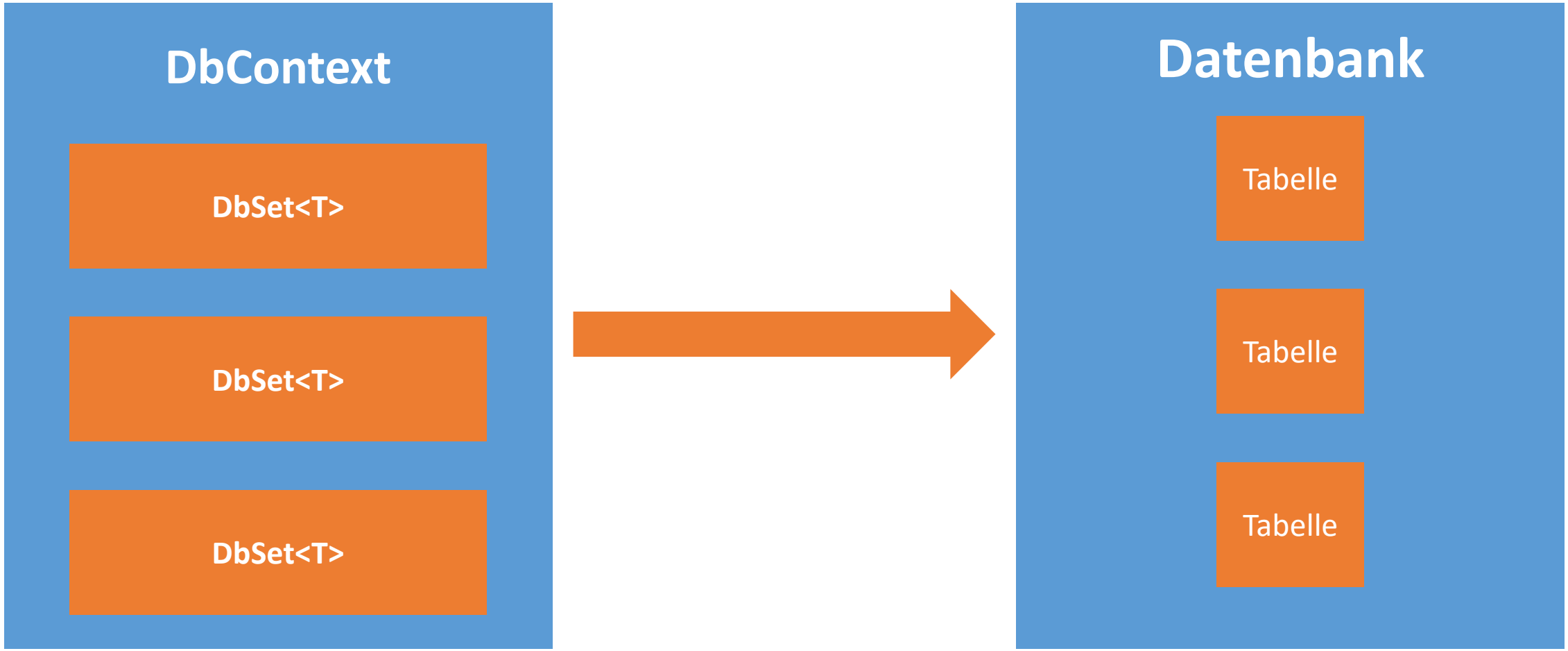
EF > INTRO > TOOLS > CLI

- REFERENCE
- https://docs.microsoft.com/de-de/ef/core/miscellaneous/cli/powershell

© ppedv AG

EF & ORM

Object-Relational Mapping in EF Core



© ppedv AG

Object / Relational Mapping

Objektbasierte Entwicklung
CLR - Klassen

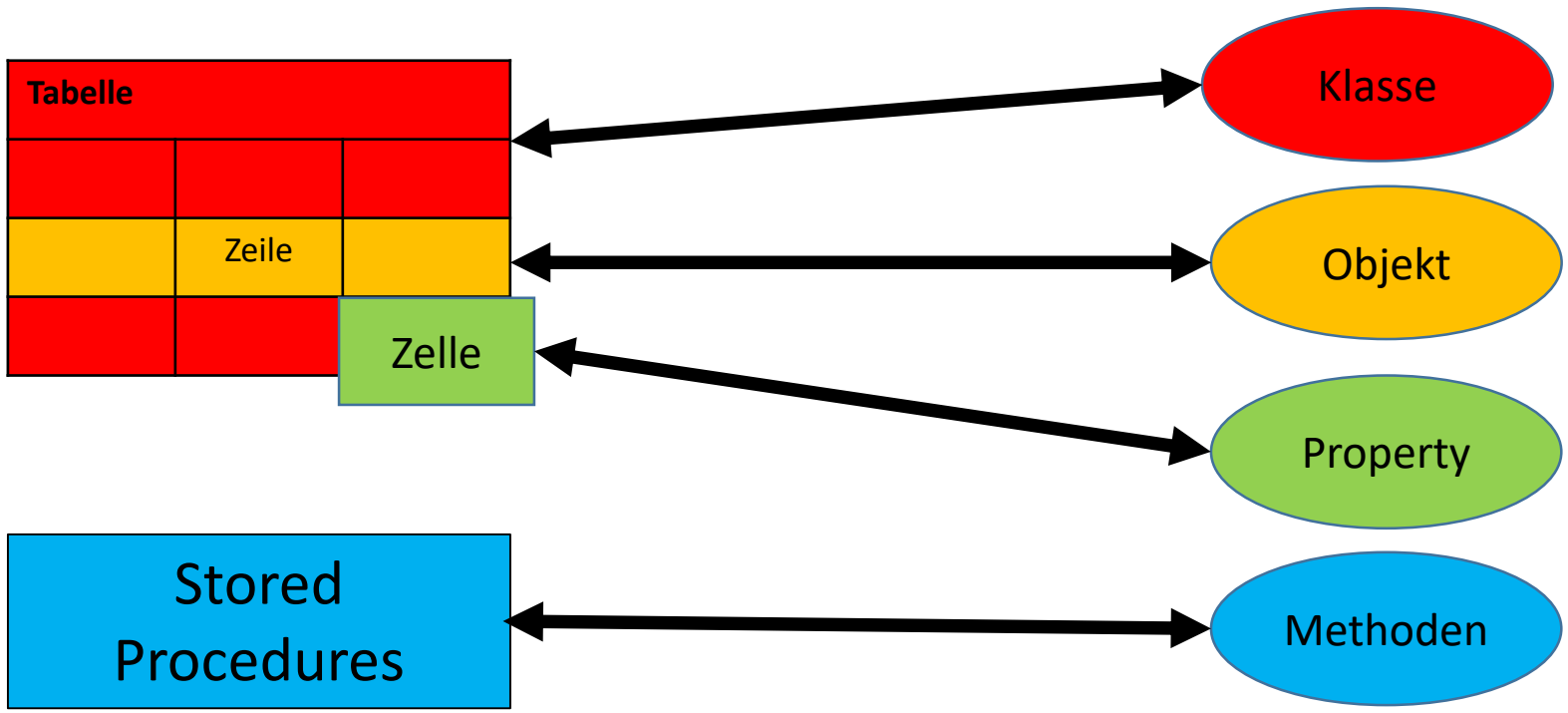
Relationale Datenbank
Microsoft SQL Server

Objektrelationale Abbildung
Entity Framework

Fokus auf Modellierung statt auf Datenbanktechnologie

© ppedv AG

Object / Relational Mapping>



© ppedv AG

Warum Objektrelationales Mapping?

- **Problem:**
  - Relationale Modelle und Objekt Modelle arbeiten schwierig vereinbar
  - Tabellarisches Datenbankformat vs. Objektdarstellung
- **Lösung:**
  - ORM: Bibliothek, die ein Objekt-Relationales Mapping implementiert
  - Kümmert sich um das Mapping zwischen tabellarischem Format und Objektdarstellung
  - ORM spart Entwicklungszeit!

© ppedv AG

EF > ENTITY DATA MODEL

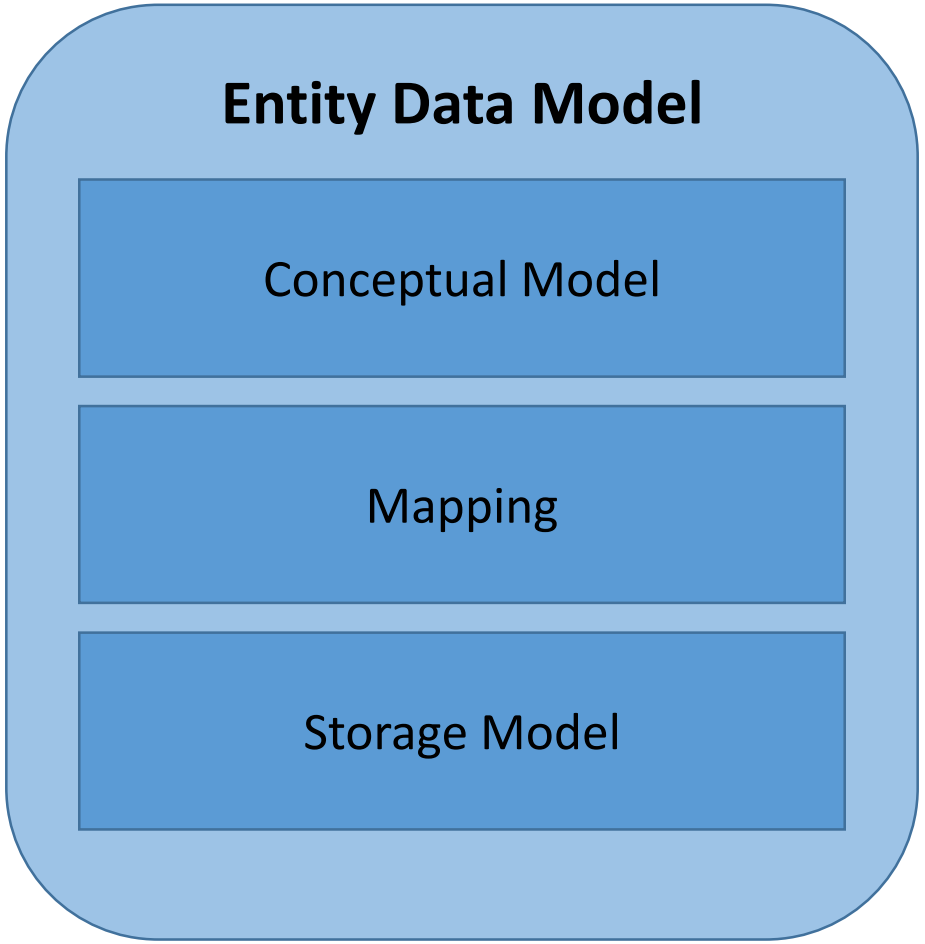
© ppedv AG

Entity Data Model

- Besteht aus drei Hauptkomponenten
  - Conceptual Model
  - Mapping
  - Storage Model
- Komponenten sind XML Dateien
  - Conceptual Schema (CSDL)
  - Mapping Schema (MSL)
  - Storage Schema (SSDL)
- Wird in Version 7 nicht mehr enthalten sein

© ppedv AG

Entity Data Model



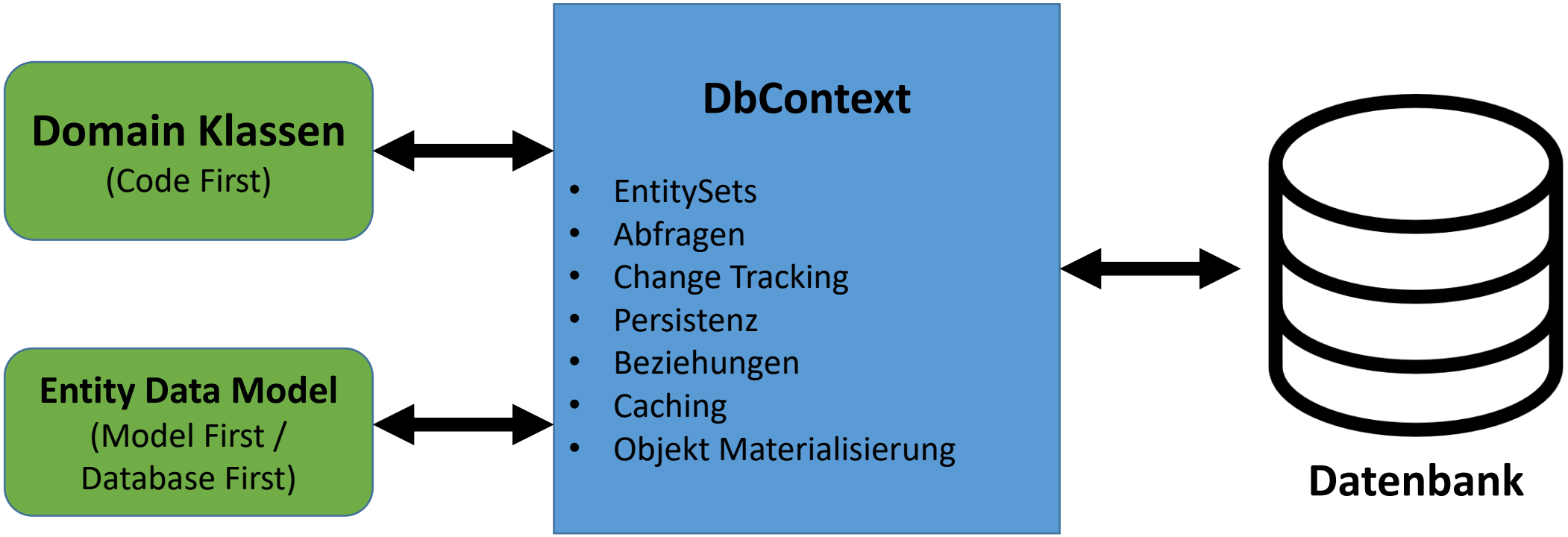
- Conceptual Model
  - Klassen
  - Beziehungen
  - Unabhängig von Datenbank
- Mapping
  - Informationen wie Conceptual Model auf Storage Model abgebildet wird
- Storage Model
  - Tabellen
  - Views
  - Stored Procedures

© ppedv AG

EF > DBCONTEXT

© ppedv AG

DbContext



© ppedv AG

DbContext

- EntitySets
  - DbSet<TEntity> für alle Entitäten, die auf DB Tabellen gemappt werden
- Abfragen
  - DbContext konvertiert LINQ-to-Entities zu SQL und sendet diese an die DB
- Change Tracking
  - Registriert Änderungen, die nach einer Abfrage an Entitäten vorgenommen wurden
- Persistenz
  - DbContext führt Insert, Update und Delete Operationen gegen die DB aus

© ppedv AG

DbContext

- Caching
  - Während der Lebenszeit des Contexts werden die abgefragten Daten zwischengespeichert
- Beziehungen
  - Verwaltet Beziehungen zwischen Entitäten (Fluent API)
- Objekt Materialisierung
  - DbContext erzeugt Objekte aus den Tabelleneinträgen

© ppedv AG

DbContext Konfiguration

- Konfiguration in `ConfigureServices` Methode in Startup Klasse
- `DbContext` benötigt eine Instantz von `DbContextOptions`
- `DbContextOptions`:
  - DB Provider (UseSqlServer, UseSqlite, UseMySql, …)
  - Connection String (appsettings.json)
  - Optionale Behavior und Selector Einstellungen

```
services.AddDbContext<ApplicationDbContext>(options =>
    options.UseSqlite(
        Configuration.GetConnectionString("DefaultConnection")));
```

Connection String für DB Provider

- EF Core Provider Übersicht:
  - https://docs.microsoft.com/en-us/ef/core/providers/
- Connection Strings in appsettings.json:

```
"ConnectionStrings": {
    "MSSQL":"Server=Name;Database=Name;Integrated Security=True;Trusted_Connection=True;",
    "MySQL": "Server=localhost; Database=Name; Uid=user; Pwd=password",
    "SQLite": "Data Source=Name.db“,
    ...
}
```

EF > DBSET

© ppedv AG

DbSet

DbSet repräsentiert eine Menge von Entitäten, mit der CRUD Operationen durchgeführt werden kann

Einige wichtige Methoden:

© ppedv AG

DbSet Methoden

- Add - Fügt eine Entität zum Context im Added Status hinzu

```
myContext.Stars.Add(sun);
```

- Remove - Setzt den Status einer vorhandenen Entität auf Deleted

```
myContext.Stars.Remove(sun);
```

- Attach - Fügt eine Entität zum Context im Unchanged Status hinzu

```
myContext.Planets.Attach(myPlanet);
```

- Find - Durchsucht den Context nach übergebenem Primärschlüssel. Schickt Abfrage gegen DB falls Entität nich im Context gefunden wurde. Gibt null zurück, wenn der Primärschlüssel nicht gefunden wurde

```
myContext.Planets.Find(3);
```

© ppedv AG

DbSet

- `DbSet<TEntity>` wird im `DbContext` hinzugefügt und repräsentiert die Tabellen in objektrelationaler Darstellung
- Zugriff mittels LINQ Extensions

```
public class ApplicationDbContext : DbContext
{
    public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)
        : base(options)
    { }

    public DbSet<Auto> Autos { get; set; }
}
```

DbSet Konfiguration

- Data Annotation Attributes in Model Klassen
- Fluent API => Konfiguration mittels `ModelBuilder` Methoden
  - `Microsoft.EntityFrameworkCore.ModelBuilder`
  - Model Konfiguration, Property Konfiguration, …

```
public class ApplicationDbContext : DbContext
{
    // DbSets

    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        // Konfigurationen
    }
}
```

EF > ENTITY TYPES

Entitätentypen

- POCO
- POCO Proxy

© ppedv AG

POCOs

Beispiel:

```
public class Star
{
    public Star()
    {
        this.Planets = new List<Planet>();
    }

    public int Id { get; set; }

    public string Name { get; set; }

    public float MassRelativeToSun { get; set; }

    public StarSystem System { get; set; }

    public List<Planet> Planets { get; set; }
}
```

© ppedv AG

POCO Proxy

- Dynamisch zur Laufzeit erstellt
- Wrapper Klasse um POCO Entitäten
- Ermöglich Lazy Loading und automatisches Change Tracking

© ppedv AG

POCO Proxy

Damit aus einer POCO Entität zur Laufzeit ein POCO Proxy werden kann, müssen folgende Bediegungen erfüllt sein:

- POCO Klasse muss public sein
- POCO Klasse darf nicht sealed sein
- POCO Klasse darf nicht abstract sein
- Jede Navigationseigenschaft muss public virtual sein
- Jede Collectioneigenschaft muss ICollection<T> sein
- ProxyCreationEnabled muss true sein (default)

© ppedv AG

POCO gültig für Proxy

Beispiel:

```
public class Star
{
    public Star()
    {
        this.Planets = new HashSet<Planet>();
    }

    public int Id { get; set; }

    public string Name { get; set; }

    public float MassRelativeToSun { get; set; }

    public virtual StarSystem System { get; set; }

    public virtual ICollection<Planet> Planets { get; set; }
}
```

© ppedv AG

EF > MODELING CONCEPTS

Modulübersicht

- Modellierung im Designer
  - Database First
  - Model First
- Code-basierte Modellierung
  - Code First
  - Code First from Database

© ppedv AG

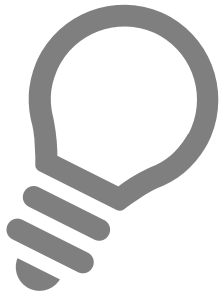
Workflow

	Designer	Code
Bestehende Datenbank	**Database First** - Reverse Engineering - POCOs werden automatisch erzeugt - Kann von Hand geändert werden	**Code First** - Reverse Engineering - Erzeugt POCOs und Mappings im Code
Neue Datenbank	**Model First** - Modell im Designer anlegen - Datenbank wird aus Modell erzeugt - POCOs werden erzeugt	**Code First** - POCOs und Mappings werden im Code von Hand erzeugt - Datenbank wird generiert

© ppedv AG

EF CORE & AUTOMATICALLY CREATE A DB

- **Code First Ansatz:**
  - Code => DB
  - Geeignet für neue Entwicklungen
  - Keine Datenbank Programmierung in SQL notwendig
- **DB First Ansatz:**
  - DB => Code
  - Erstellung der Model Klassen aus der Datenbank
  - Geeignet für bereits bestehende Datenbanken



© ppedv AG

Database First

- Reverse Engineering:
  - Modell und Klassen aus Datenbank
- Graphische Modellierung im Designer

© ppedv AG

Model First

- Modell im Designer entwerfen
- Datenbank wird aus Modell erstellt
- Klassen werden aus Modell generiert

© ppedv AG

Code-basierte Modellierung

- Klassen und Mappings im Code
- Flexibel und übersichtlich (keine XML Dateien)
- Datenbank-Schema Migrationen
- Code-First Konventionen
- Konfiguration mit Annotationen und Fluent API

© ppedv AG

Code First Conventions - Möglichkeiten

- Data Annotations
- Fluent Api

© ppedv AG

Code First / Code First from Database

- Forward Engineering
  - Domain-Klassen und Mappings in Code
  - Datenbank wird aus Code Modell erstellt
  - Änderungen am Modell per Migration auf Datenbankschema anwenden

- Reverse Engineering
  - Klassen und Mappings werden aus vorhandener Datenbank generiert

© ppedv AG

EF > DB FIRST

© ppedv AG

Database First

- Installieren der NuGet Packages:
  - Microsoft.EntityFrameworkCore.SqlServer
  - Microsoft.EntityFrameworkCore.Tools
  - Microsoft.VisualStudio.Web.CodeGeneration.Design

- DbContext generieren mittels PM Console:

```
Scaffold-DbContext "Server= ServerName;Database= DbName;
Trusted_Connection=True;" Microsoft.EntityFrameworkCore.SqlServer
-OutputDir Models –t dbo.Car
```

© ppedv AG

DbContext Konfiguration Startup.cs

- `ConfigureServices` Methode
- DI Model per AddDbContext
- DB Typ per DBContextOptions
  - DB Provider (UseSqlServer, UseSqlite, UseMySql,UseInMemory …)
  - Connection String (appsettings.json)
  - Optionale Behavior und Selector Einstellungen

```
services.AddDbContext<ApplicationDbContext>(options =>
    options.UseSqlite(
        Configuration.GetConnectionString("DefaultConnection")));
```

© ppedv AG

DbContext Konfig

- Per Tabelle/Obkjekt ein DbSet<TEntity>
- Konstruktor zwei Optionen
  - mit DI Options Injection
  - Leer->OnConfiguring (Fluent Api)

```
public class ApplicationDbContext : DbContext
{
    public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)
        : base(options)
    { }

    public DbSet<Auto> Autos { get; set; }
}
```

© ppedv AG

DbSet Konfiguration

- Data Annotation Attributes in Model Klassen
- Fluent API => Konfiguration mittels `ModelBuilder` Methoden
  - `Microsoft.EntityFrameworkCore.ModelBuilder`
  - Model Konfiguration, Property Konfiguration, …

```
public class ApplicationDbContext : DbContext
{
    // DbSets

    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        // Konfigurationen
    }
}
```

© ppedv AG

Connection String für DB Provider

- Dev vs production
- Connection Strings in appsettings.json:

```
"ConnectionStrings": {
    "MSSQL":"Server=Name;Database=Name;Integrated Security=True;Trusted_Connection=True;",
    "MySQL": "Server=localhost; Database=Name; Uid=user; Pwd=password",
    "SQLite": "Data Source=Name.db“,
    ...
}
```

© ppedv AG

Database First Modell Änderungen

- Manuell über die Model Klassen
- Konfiguration mittels *FluentAPI* in `OnModelCreating`

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
{
   modelBuilder.ApplyConfiguration(new CarConfiguration());
}
...
public class CarConfiguration : IEntityTypeConfiguration<Car>
{
    public void Configure(EntityTypeBuilder<Car> builder)
    {
        builder.Property(e => e.Name)
            .IsRequired()
            .HasMaxLength(100);
    }
}
```

Model First & Migrations

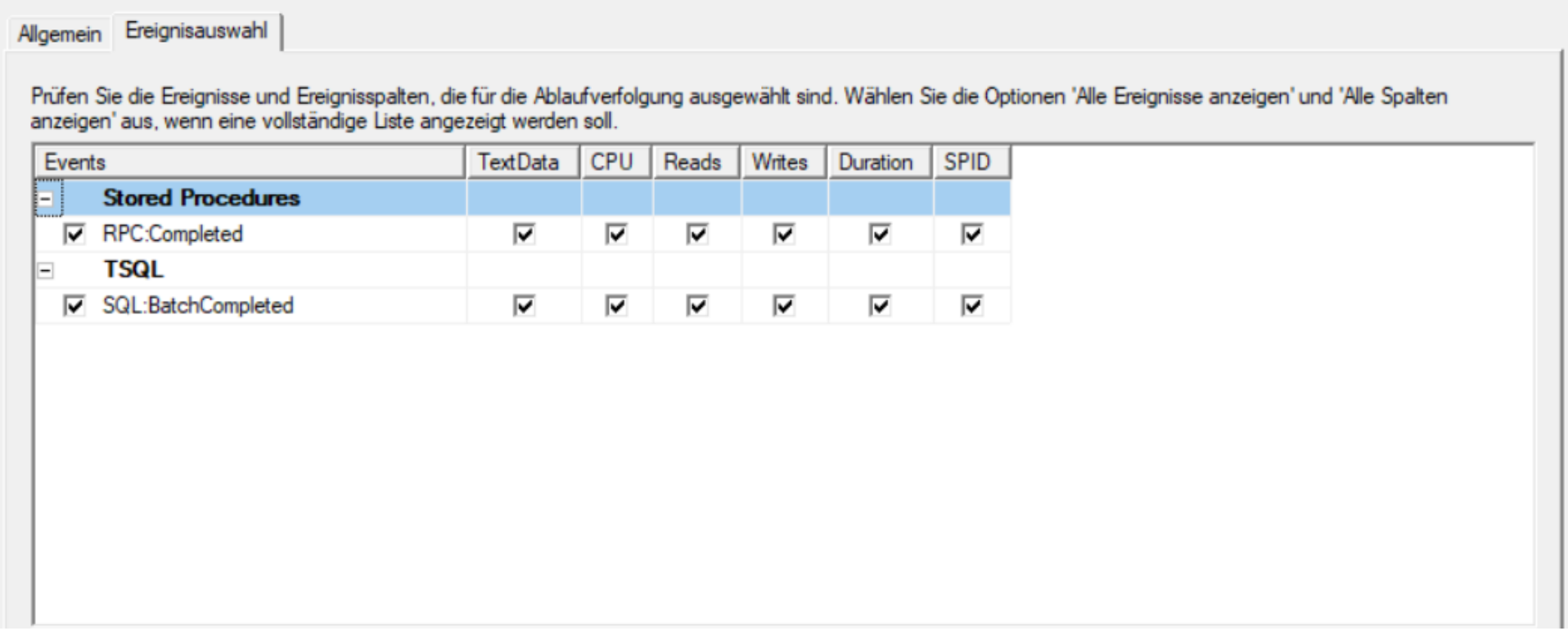
- Datenbank anhand Model Klassen
- Erstellen der Model Klassen für Datenbank Schema
- Verwendung von DB Migrations um Änderungen an DB vorzunehmen:

Package Manager Console Anweisung	Verwendung
add-migration *MigrationName*	Erstellen einer Migration
update-database	Update von Datenbank Schema auf Basis der Migration
remove-migration	Entfernt letzte Migration
script-migration	Erstellung eines SQL Scripts auf Basis der letzten migration
update-database –migration Name	Rücksprung zu spezifischer Migration

© ppedv AG

SQL Server Profiler

- Tool für Debugging, Tracing & Monitoring von SQL Server
- Teil von SQL Server Management Studio: Extras > SQL Server Profiler



Exkurs: Deferred Execution

- `.ToList(), .ToListAsync(), .ToArrayAsync(), .ToDictionary()`
  - Lädt die Daten sofort in den Speicher
- Besser:
  - Query Commands in Variable speichern
  - Ausführung verzögern
  - `IQueryable<T>` erstellt Expression Tree
  - Ausführen mittels `.ToList()` etc.
  - Singleton queries

Übung 5: Entity Framework Core

Konventionen

- ID, TabelleID
- Foreign Key FremdTabelleID
- Virtuelle Propertys
  - List<ClientTabelle>
- Attribute
  - Key
  - DatabaseGenerated(DatabaseGeneratedOption.Identity)

© ppedv AG

Erste Schritte

- Lesen
  - Generische Liste
- Schreiben
  - Entität
  - SaveChanges
- Plain SQL
  - Entity FromSQL
  - Database.ExecuteSqlCommandAsync

EF & LAZY LOADING

© ppedv AG

Lazy Loading

- Verknüpfungen werden nicht geladen
  - Count Children
- Include
  - String
  - Besser Lambda Ausdruck

Globale AbfrageFilter

- OnModelCreating
  - Zb Userrechte
  - HasQueryFilter
- IgnoreQueryFilters

Tracking generiertes SQL

- SQL Profiler
  - AbfrageTags
    - TagWith("This is my spatial query!")
- Logging

CONCURRENCY EXCEPTION

- dt.: Parallelitätsausnahme

```
catch (DbUpdateConcurrencyException)
    {
        if (!MovieExists(Movie.ID))
        {
            return NotFound();
        }
        else
        {
            throw;
        }
    }
```

© ppedv AG

DB AN MODELL ANPASSEN

© ppedv AG

EF > MIGRATIONS

EF > MIGRATIONS > IDEA

- WAS SIND MIGRATIONS
  - eine Reihe von Tools

- WOZU SIND MIGRATIONS
  - mit denen kann man eine Datenbank erstellen und aktualisieren, damit die dem Datenmodell der App entspricht

© ppedv AG

- Add Migration

© ppedv AG

EF > QUERING ENTITIES

Modulübersicht

- LINQ to Entities
- Entity SQL
- Lazy Loading
- Eager Loading
- Explizit Loading
- Change Tracking

© ppedv AG

Entitätsdaten ändern/updaten

Entitätsdaten löschen

EF > INHERITENCE

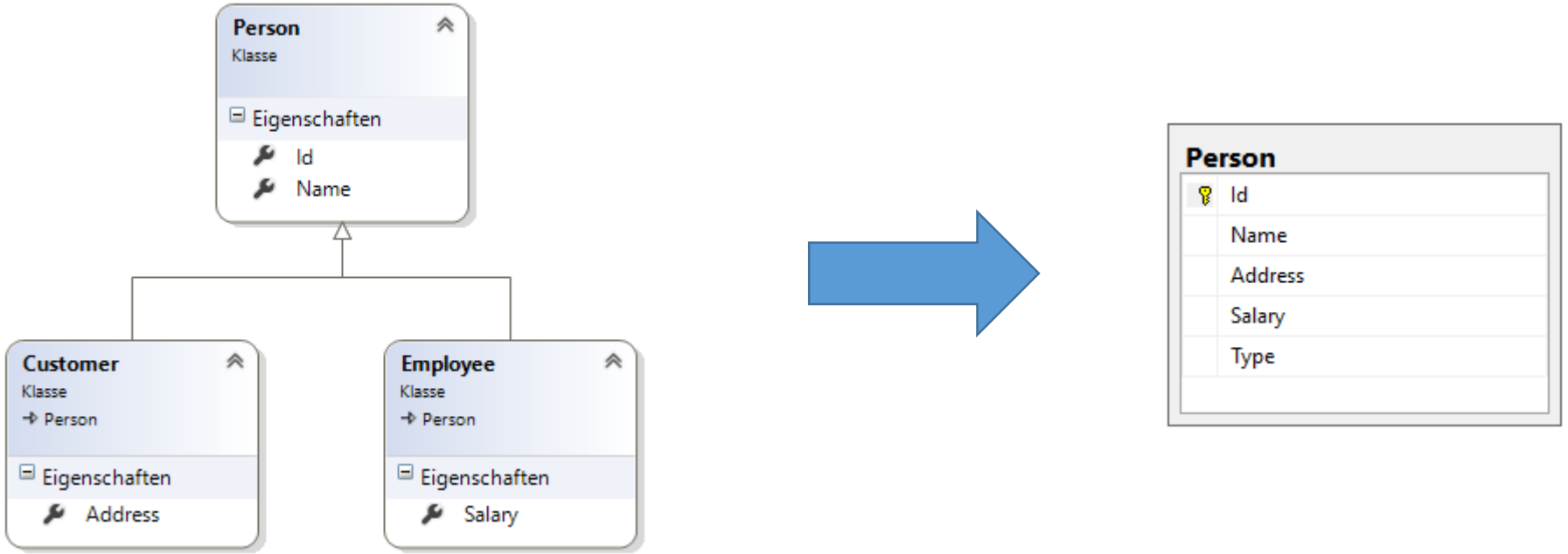
© ppedv AG

Vererbung im Entity Framework

- TPH - Table-per-Hierarchy
- TPT - Table-per-Type
- TPC - Table-per-Concrete-Type

© ppedv AG

Table-per-Hierarchy (TPH)



© ppedv AG

Table-per-Type (TPT)

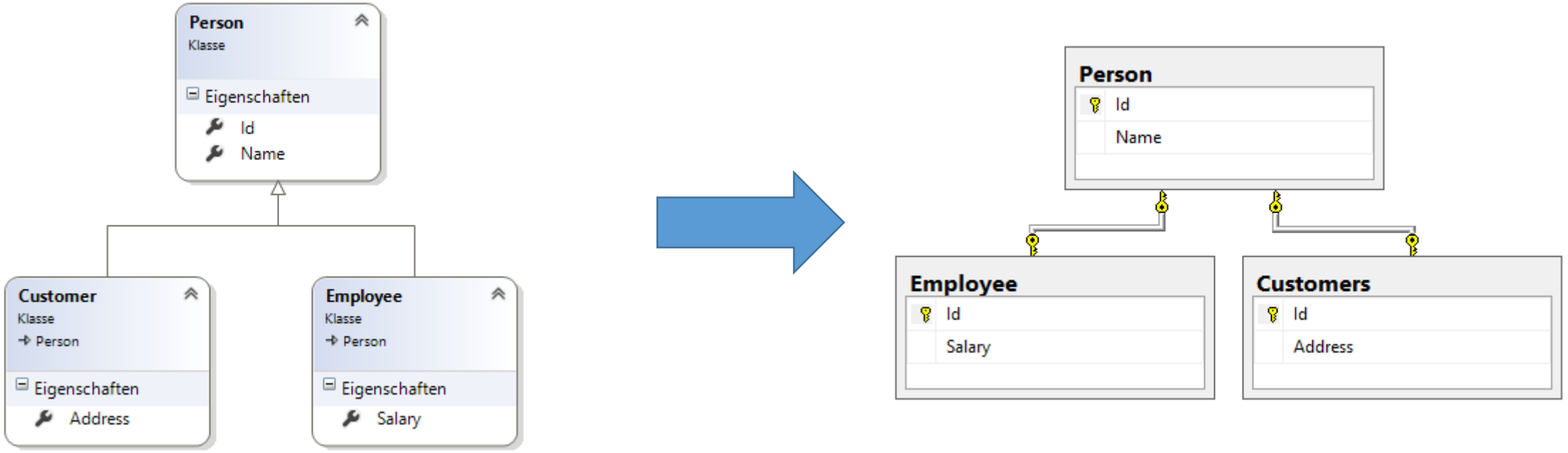
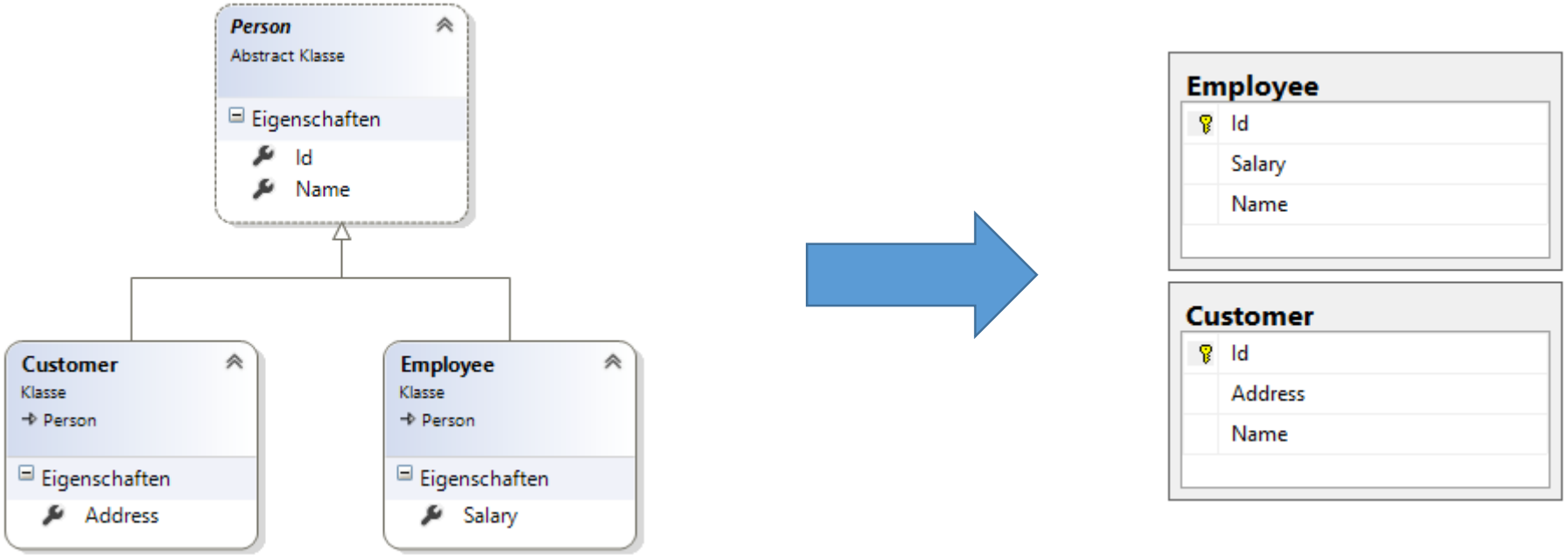


Table-per-Concrete-Type (TPC)



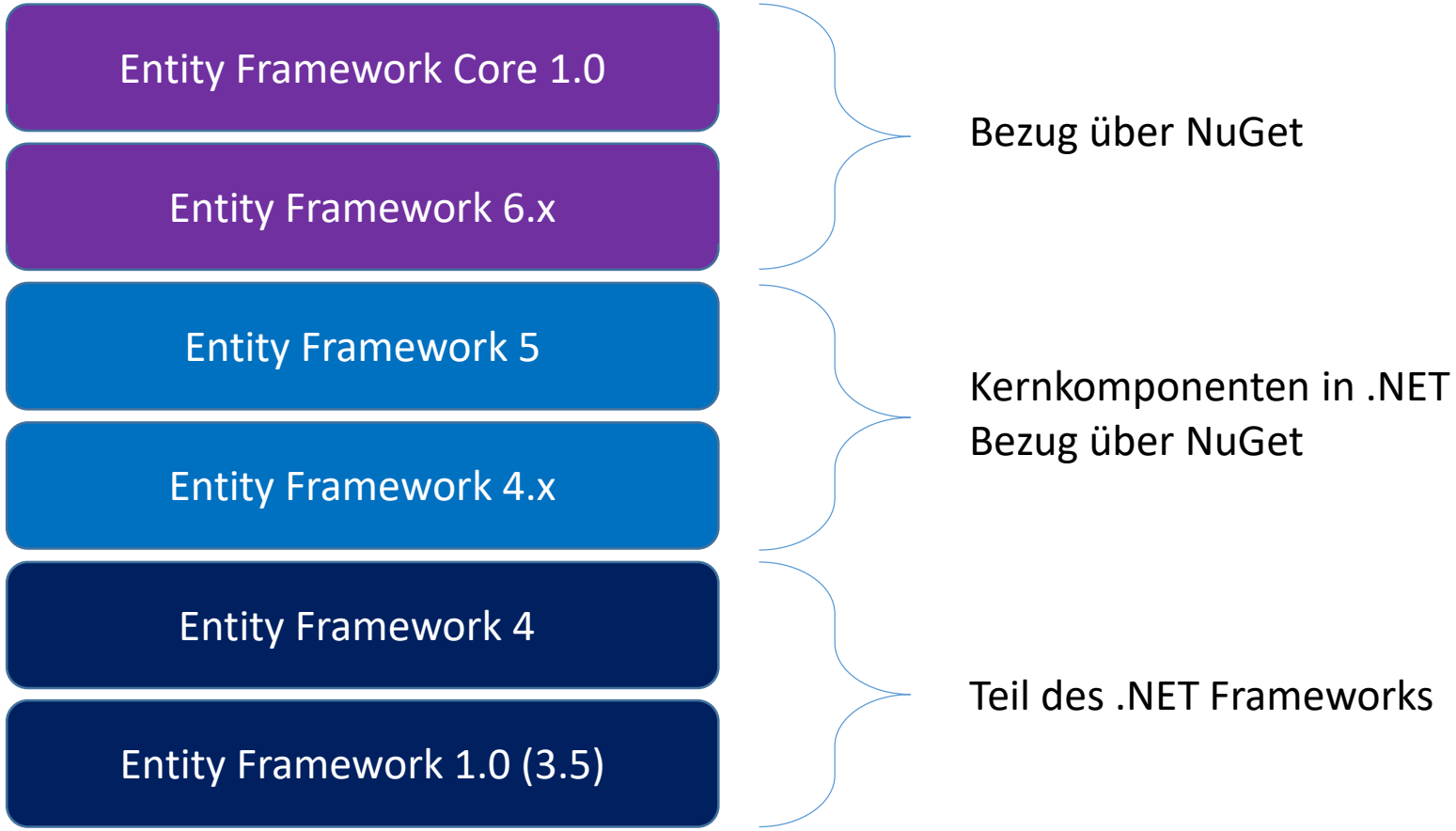
EF > FACTS

© ppedv AG

- EF ist seit Dezember 2019 in Version 6.4 #checkForUpdates
- EF Core ist seit Dezember 2019 in Version 3.1 #checkForUpdates

© ppedv AG

Versionsgeschichte



© ppedv AG

Kompatibilität

EF Version	.Net Version	3.5.1	4.0	4.5	4.5.1
EF 3.5		✓	✗	✗	✗
EF 4.x		✗	✓	✓	✓
EF 5		✗	✗	✓	✓
EF 6		✗	✓	✓	✓

© ppedv AG

Neuerungen in Version 6

- Support für Async
- Connection resilience
- Code-based Configuration
- Interceptors
- Performance Verbesserungen
- Entity Framework ist Open Source

© ppedv AG

Open Source

- Quellcode offen
- Nightly Builds
- Issue Tracking
- Community Contributions
- weiterhin offizieller MS Support (Qualität, Lizenz)

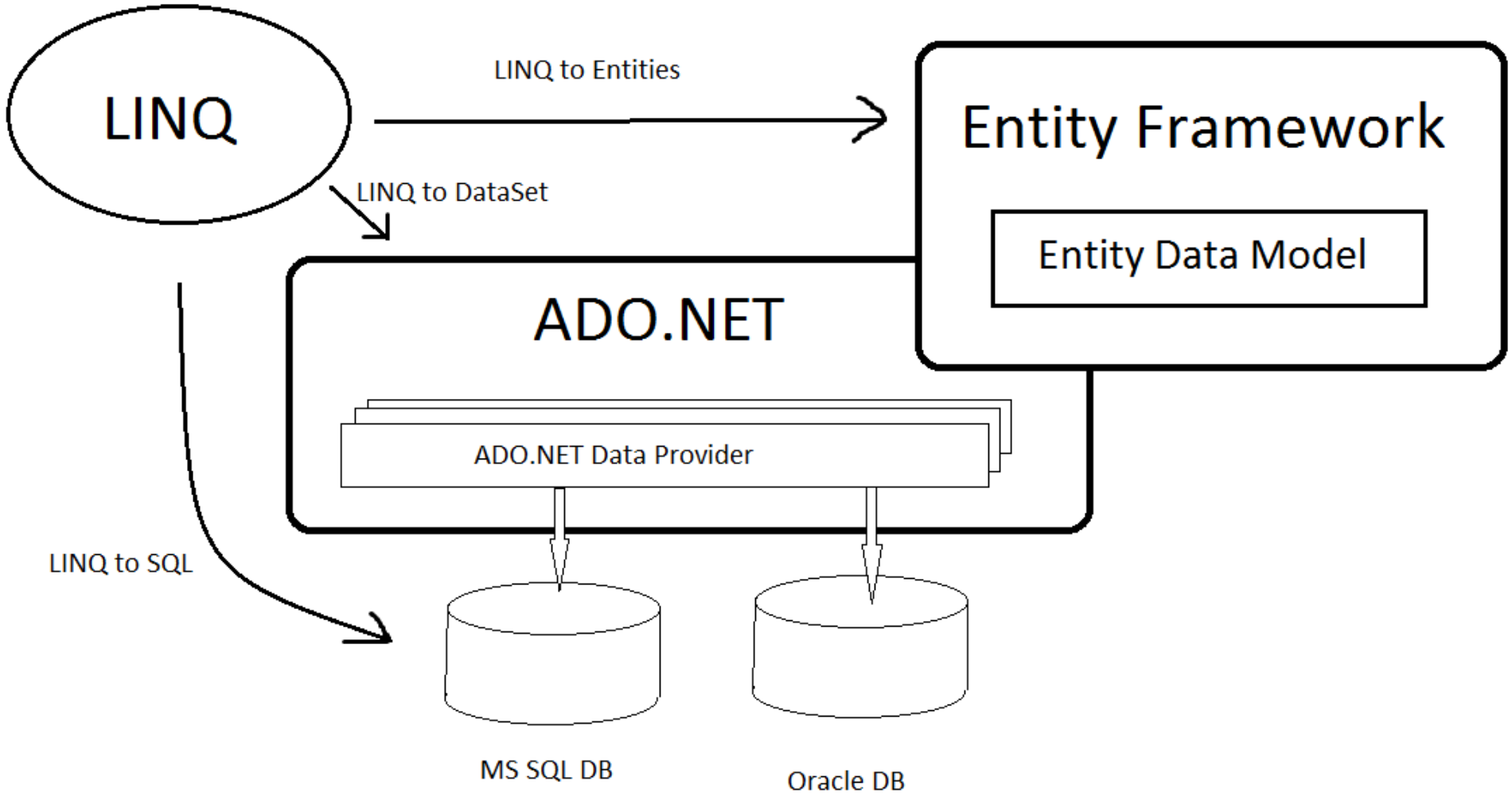
© ppedv AG

Code First only Neuerungen in Version 6

- Custom Conventions
- Support für Stored Procedures

© ppedv AG

EF 6 Architektur



© ppedv AG

EF 6 Architektur

- Baut auf ADO.NET auf
- ADO.NET Data Providers für Datenbankzugriff
- Entity Data Model um Entities/Mappings abzubilden
- LINQ um Queries gegen DB zu schicken

© ppedv AG