Performancetuning für LINQ und Entity Framework

Speaker: Olaf Lischke

Speaker: Olaf Lischke

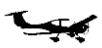
macht .NET, seit es .NET gibt

versucht, Projekte und Seminare zu kombinieren

singt Tenor in zwei Chören

zockte schon auf dem ZX 81, heute ausschließlich auf PC

fotografiert, seit er eine Kamera halten kann



fliegt, wenn Wetter und Zeit es zulassen (TMG/SEP)

Überblick

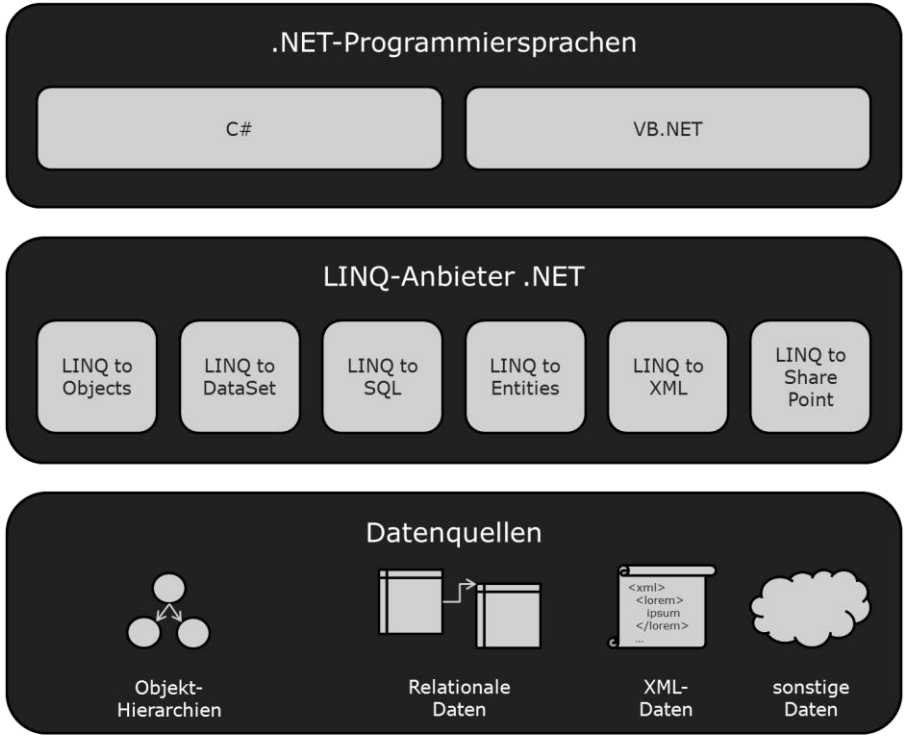
- Language Integrated Query (LINQ) ⇨ Sprachfeature
- „Black Patterns“
- Paralleles LINQ
- Entity Framework

LINQ ist ein Sprachfeature

(und kann auch Datenzugriffe, aber eben nicht nur…)

LINQ als Sprachfeature

- Mengenoperationen in prozeduralen Sprachen
- Auch in TypeScript, JavaScript, Java, Python, F#,…
- Providerkonzept über `IEnumerable<T>` `IQueryable<T>`



Quelle: Wikipedia

„Black Patterns“

LINQ „Black Patterns“

- Hidden Allocation
  - „Lambdas are Closures“
  - Anonymous Typing
- Deferred Execution

Paralleles LINQ („PLINQ“)

Parallel LINQ (PLINQ)

- Bestandteil der Task Parallel Library (TPL)
  - daher gilt alles aus der TPL auch hier:
    - Partitionierung durch Taskplaner
    - Cancellation, DegreeOfParallelism…
- `ParallelIEnumerable` bringt u.a.
  - `.AsParallel()`
  - `.AsOrdered()`
  - `.ForAll()`

Paralleles LINQ

- AsParallel()

Aggregate	**First**	**Positional Select**	**SkipWhile**
All	**FirstOrDefault**	**Positional SelectMany**	Sum
Any	GroupBy	**Positional SkipWhile**	**Take**
Average	GroupJoin	**Positional TakeWhile**	**TakeWhile**
Cast	Intersect	**Positional Where**	ThenBy
Concat	Join	Range	ThenByDescending
Contains	**Last**	Repeat	ToArray
Count	**LastOrDefault**	Reverse	ToDictionary
DefaultIfEmpty	LongCount	Select	ToList
Distinct	Max	SelectMany	ToLookup
ElementAt	Min	SequenceEqual	ToSequence
ElementAtOrDefault	OfType	Single	Union
Empty	OrderBy	SingleOrDefault	Where
Except	OrderByDescending	**Skip**	**Zip**

Quelle: [blogs.microsoft.com]

Erzwingt sequentielle Ausführung in .NET:

4 + 4.5

4

Entity Framework

IEnumerable vs. IQueryable

IEnumerable

- Forward-only
- Deferred Execution
- „In-Memory“
  - Verarbeitung ausschl. im Speicher
  - Kein Lazy-Loading
- Collections, XML

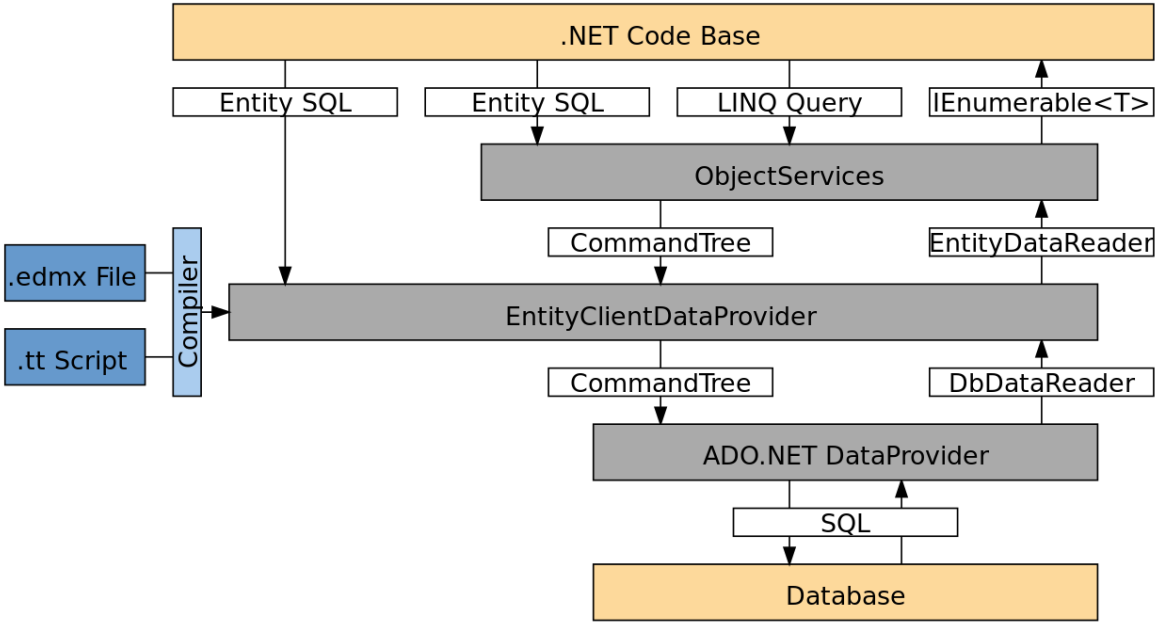
LINQ2Objects

IQueryable

- Forward-only
- Deferred Execution
- „Remote“
  - Verarbeitung durch den Provider
  - Lazy-Loading, Filtern, Paging providerseitig
- Datenbankzugriffe

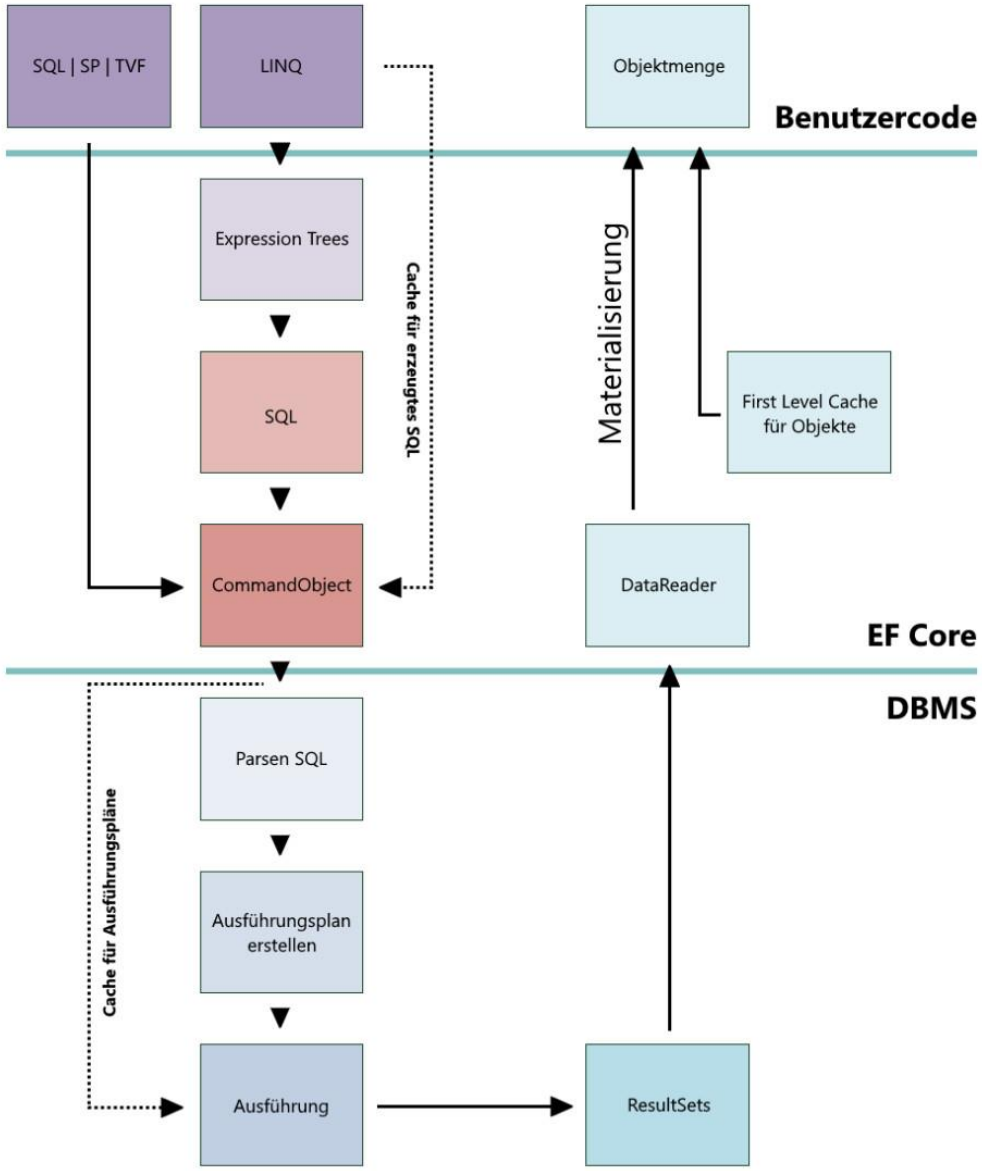
LINQ2Entities

Entity Framework 6.x



Quelle: Wikipedia

EF Core



EF Core – LINQ Queries

- Nicht alle Queries werden vom Server evaluiert
- „Usual Suspects“:
  - GroupBy
  - Union
  - Select/Where mit Lamdas und Projektion
- Exception werfen, falls Query clientseitig evaluiert wird:

```
protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder
{
  optionsBuilder
    .UseSqlServer(@"Server=(localdb)\mssqllocaldb;Database=EFQuerying;Truste
    .ConfigureWarnings(warnings =>
        warnings.Throw(RelationalEventId.QueryClientEvaluationWarning));
```

Entity Framework „Stellschrauben“

- ChangeTracker
  - .AsNoTracking()
- Anzahl generierter Queries
  - .Include() statt Schleifen
  - JOINs präferieren
- Datenmenge
  - Projektion
  - In-Memory vs. Roundtrips zur DB
    (siehe IEnumerable vs. IQueryable)
  - Wieviele Kontexte…?

Fazit

Fazit

- „To LINQ, or not to LINQ“ ist hier keine Frage
- „It‘s just a tool, dude!“
- Bewertung:
  Developer-Performance <> Runtime-Performance

Vielen Dank!

Slides und Samples auf
https://github.com/olaflischke/bastaspring2019

Für spätere Fragen:
olaf.lischke@lischke-edv.de