



# Entity Framework Core mit PostgreSQL

Speaker: Olaf Lischke

# Speaker: Olaf Lischke



macht .NET, seit es .NET gibt



versucht, Projekte und Seminare zu kombinieren



singt Tenor in Chören und Musikprojekten



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)

# Was? Postgres? Mal gehört, aber...

- Open Source Project
  - Seit 1986 University of California, Berkley
  - Erste öffentliche Release: 1989
- Einsatz und Distribution kostenlos
- Lange Lebensdauer der Versionen
  - Aktuelle Versionen: 11.21, 12.16, 13.12, 14.9, 15.4, 16
  - Neue Versionen für 11, 12, 13, 14, 15 **und** 16 ca. alle 3 Monate

14th September 2023: PostgreSQL 16 Released!

# Wer benutzt denn sowas...?

- [StackOverflow Survey](#)
- **Unternehmen und Behörden**
  - Netflix
  - Uber
  - Rotes Kreuz (USA)...
- **Cloud-Anbieter**
  - Microsoft Azure
  - Google Cloud
  - Amazon



# Was kann Postgres, was SQL-Server/Oracle nicht kann?

- Database Templating
- Table-Inheritance
- Einfache(?) .conf-Files
- SSL für die Client-Server-Kommunikation
- Vielfältige Replikationstechniken (a/synchron, gestreamt, logisch, physisch...)
- Multi-Version Concurrency Control (MVCC)
- Serializable Snapshot Isolation (SSI)
- Spricht auch Tcl, Perl, Java, Lua, sh, JavaScript
- Läuft auch auf Linux, Unix, Mac, BSD, Solaris...

# Ich kann SQL – reicht das...?

- Ja. PostgreSQL benutzt zwar PL/pgSQL, aber die grundsätzliche Syntax ist sehr ähnlich zu T-SQL
- Teils andere Nomenklatur
- Tabellennamen in "" statt []
- Case Sensitive
- Schema-Angabe!

SQL Server	PostgreSQL
bit	Boolean
[var]char(n)	Text, bpchar, [var]char(n)
uniqueidentifier	ByteA
image	ByteA
rowversion	ByteA
binary	ByteA
geometry	Point, Line, LSeg, Box, Path, Polygon, Circle
?	json, jsonb

# Zeitzone in PostgreSQL

- "UTC everywhere"

- Zeitangabe gilt immer als UTC-Zeitangabe, wenn nichts anderes angegeben.
- **Daher:** DateTime (C#) → timestampz, keine implizite Konvertierungen mehr.
- **Oder:** Selbst konfigurieren, ggf. DateTimeKind verwenden
- **Oder:** Altes Verhalten erzwingen:  
`AppContext.SetSwitch("Npgsql.EnableLegacyTimestampBehavior", true);`

- DateOnly/TimeOnly-Unterstützung

- NodaTime-Unterstützung (<https://nodatime.org/>)

Bisher:

DateTime - Kind	PostgreSQL Typ
UTC	timestampz
Local	timestamp
Unspecified	timestamp

# Gibt's sowas wie ein Management Studio?

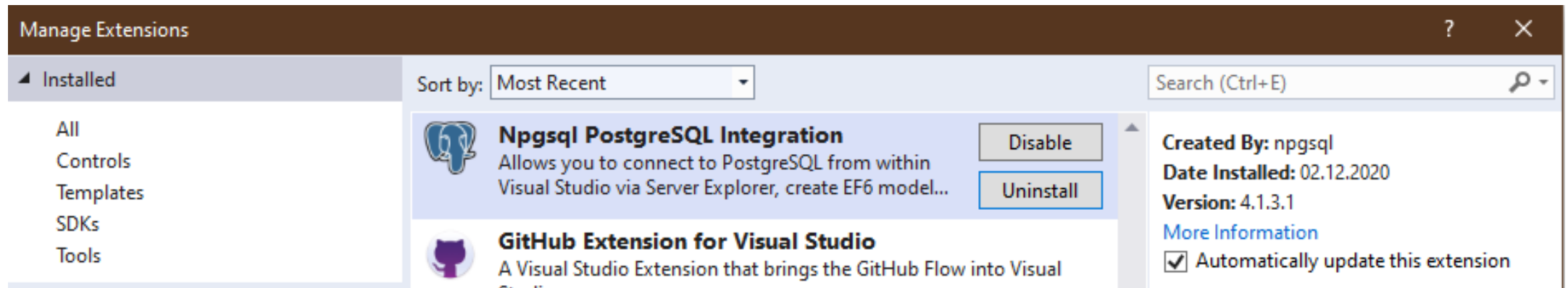
- Im Lieferumfang enthalten ist
  - pgAdmin
    - Browserbasiert
    - benutzt Admin Server (Erststart mit Geduld)
  - SQL Shell (psql)
    - Konsole
- Aber folgende können auch PostgreSQL:
  - Oracle SQL Developer
  - **DBeaver** (<https://dbeaver.io/>)
  - OmniDB (<https://omnidb.org/>)
  - Sqlectron (<https://sqlectron.github.io/>)

Open Source | |kostenlos



# Kann Visual Studio mit Postgres...?

- ~~SQL Server Object Explorer~~ (weil kein SQL Server...)
- Server Explorer
  - Npgsql Postgres Extension <https://marketplace.visualstudio.com>



# Und für EF Core gibt es einen Treiber?

- <https://github.com/npgsql> - Shay Rojansky (<https://github.com/roji>)
- Architektur/Packages:

EF Core

Microsoft.EntityFrameworkCore  
Microsoft.EntityFrameworkCore.Tools  
Microsoft.EntityFrameworkCore.Relational  
Microsoft.EntityFrameworkCore.Design

Provider

**Npgsql.EntityFrameworkCore.PostgreSQL**  
Microsoft.EntityFrameworkCore.SqlServer  
Pomelo.EntityFrameworkCore.MySql  
Oracle.EntityFrameworkCore

Demo:  
Einsatz

# Was kann der Npgsql-Treiber? – Teil 1

- Generell

- **optionsBuilder.UseNpgsql(connectionString,**

- opt =>

- {

- opt.EnableRetryOnFailure();

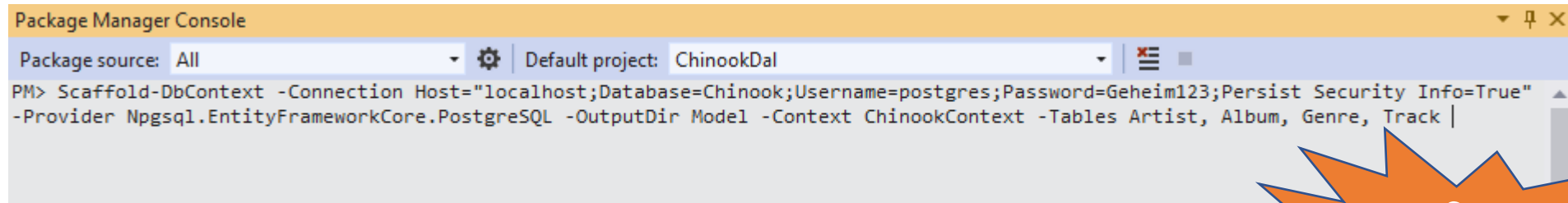
- opt.RemoteCertificateValidationCallback(...);

- opt.ProvideClientCertificatesCallback(...);

- });

# Was kann der Npgsql-Treiber? – Teil 2

- Reverse Engineering
  - Scaffold-DbContext **-Provider Npgsql.EntityFrameworkCore.PostgreSQL**



```
Package Manager Console
Package source: All Default project: ChinookDal
PM> Scaffold-DbContext -Connection Host="localhost;Database=Chinook;Username=postgres;Password=Geheim123;Persist Security Info=True"
-Provider Npgsql.EntityFrameworkCore.PostgreSQL -OutputDir Model -Context ChinookContext -Tables Artist, Album, Genre, Track |
```



- Code First - Bereitstellung  
(aus Microsoft.EntityFrameworkCore.Infrastructure)
  - EnsureCreated() / EnsureDeleted()
  - Migrate()

# Was kann der Npgsql-Treiber? – Teil 3

- Auto-Inkrement
  - entity
    - .HasIdentityOptions(startValue, incrementBy, minValue...)**
- Stored-Procedure-Mapping
  - InsertUsingStoredProcedure()
  - UpdateUsingStoredProcedure()
  - DeleteUsingStoredProcedure()

```
modelBuilder.Entity<Person>()
    .InsertUsingStoredProcedure(
        "People_Insert",
        storedProcedureBuilder =>
        {
            storedProcedureBuilder.HasParameter(a => a.Name);
            storedProcedureBuilder.HasResultColumn(a => a.Id);
        })
    .UpdateUsingStoredProcedure(
        "People_Update",
        storedProcedureBuilder =>
        {
            storedProcedureBuilder.HasOriginalValueParameter(person => pers
            storedProcedureBuilder.HasOriginalValueParameter(person => pers
            storedProcedureBuilder.HasParameter(person => person.Name);
            storedProcedureBuilder.HasRowsAffectedResultColumn();
        })
    .DeleteUsingStoredProcedure(
        "People_Delete",
        storedProcedureBuilder =>
        {
            storedProcedureBuilder.HasOriginalValueParameter(person => pers
```

# Was kann der Npgsql-Treiber? – Teil 4

aus: Microsoft.EntityFrameworkCore

- EntityTypeConfiguration
  - FluentApi Konfiguration (OnModelCreating) je Typ in eigene, übersichtliche Datei auslagern:

1 Verweis | Olaf Lischke, vor 7 Tagen | 1 Autor, 1 Änderung

```
internal class EmployeeConfiguration : IEntityTypeConfiguration<Employee>
{
```

0 Verweise | Olaf Lischke, vor 7 Tagen | 1 Autor, 1 Änderung

```
public void Configure(EntityTypeBuilder<Employee> builder)
{
```

```
    builder.Property(e => e.EmployeeId).ValueGeneratedNever();
```

```
    builder.HasOne(d
```

```
        .WithMany(p =
```

```
        .HasForeignKey
```

```
        .HasConstraint
```

in OnModelCreating:

```
// EntityTypeConfigurations
```

```
modelBuilder.ApplyConfiguration(new EmployeeConfiguration());
```

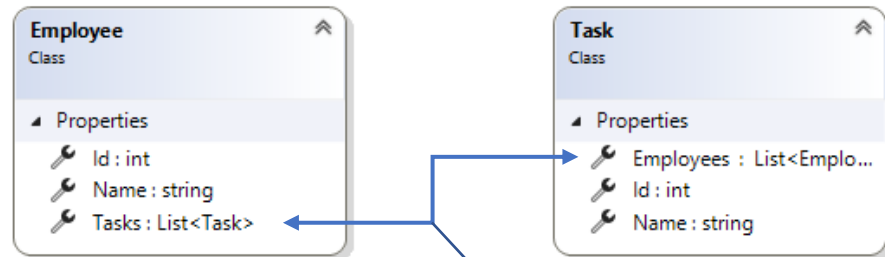
```
modelBuilder.ApplyConfiguration(new EmployeeJsonConfiguration());
```

```
modelBuilder.ApplyConfiguration(new OrderConfiguration());
```

# Was kann der Npgsql-Treiber? - Teil 5

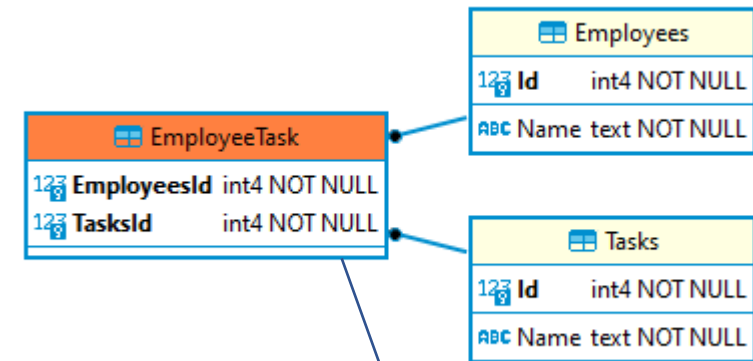
- N:M-Zuordnungen

## Objektmodell



Ein Mitarbeiter hat viele Tasks,  
an einem Task arbeiten viele Mitarbeiter

## Datenbankschema



Enthält das Modell nur Zuordnung,  
wird Zuordnungstabelle angelegt.

Auch Reverse  
Engineered!

ASTA!

# Was kann der Npgsql-Treiber? – Teil 6

- JSON-Datentypen:
  - json – großes Textfeld/BLOB
  - jsonb
    - JSON-Objekte
    - inkl. Indizierung und [Query-Syntax](#)

123 employee_id	JSON data
1	{"LastName":"Davolio","FirstName":"Nancy","Title":"Sales Representati
2	{"LastName":"Fuller","FirstName":"Andrew","Title":"Vice President, Sale
3	{"LastName":"Leverling","FirstName":"Janet","Title":"Sales Representati
4	{"LastName":"Peacock","FirstName":"Michael","Title":"Sales Represent
5	{"LastName":"Buchanan","FirstName":"Margaret","Title":"Sales Manager"
6	{"LastName":"Suyari","FirstName":"Peggy","Title":"Sales Representa
7	{"LastName":"King","FirstName":"Adele","Title":"Sales Representative"
8	{"LastName":"Callahan","FirstName":"Heather","Title":"Sales Coord
9	{"LastName":"Dodds","FirstName":"Isabel","Title":"Sales Represent

**EmployeesJson**  
Class  
  
Properties

- Data : string
- EmployeeId : short

```
EmployeesJson nancy = context.EmployeesJsons
    .FirstOrDefault(
        emp => EF.Functions.JsonContains(emp.Data, "{\"FirstName\":\"Nancy\"}")
    );
```

```
EmployeesJson andrew = context.EmployeesJsons
    .FromSqlRaw("SELECT * FROM employeesjson WHERE data ->> 'FirstName' = 'Andrew'")
    .FirstOrDefault();
```



# Was kann der Npgsql-Treiber? – Teil 7

- Massenoperationen  
ExecuteUpdate(), ExecuteDelete()
- kein ChangeTracking!
- direkte Ausführung ohne SaveChanges()!

```
int delayedOrders = context.Orders
    .Where(od => od.OrderDate.Value.Year == 1998)
    .ExecuteUpdate(
        s => s.SetProperty(od => od.ShippedDate, od => od.ShippedDate.Value.AddYears(10))
    );
```

# Breaking Change zu EF Core 6

- `modelBuilder.Entity<T>.UseXminAsConcurrencyToken();`



Obsolete!

- Stattdessen:

- Standardverfahren wie bei anderen Providern:

```
modelBuilder.Entity<T>  
    .Property(p => p.Version)  
    .IsRowVersion()
```



uint

- Mehr dazu: <https://www.npgsql.org/efcore/modeling/concurrency.html>

# Ausblick auf EF Core 8

- JSON EF Core Style

Unterstützung für JSON analog EF Core Standardverfahren

- HierachyId

Abbildung hierarchischer Daten (auch .NET, nicht nur EF Core)

- „Modell geändert?“

Prüfen, ob Migrationen fehlen

```
dotnet ef migrations has-pending-model-changes  
context.Database.HasPendingModelChanges()
```

- Gleich im Anschluss:

10:30 - Dr. Holger Schwichtenberg: [Neuigkeiten in Entity Framework Core 8.0](#)

# Vielen Dank!

Slides auf

<https://github.com/olaflischke/basta2023>

Für spätere Fragen:

[olaf.lischke@lischke-edv.de](mailto:olaf.lischke@lischke-edv.de)