Wünsche/Themen

Mittwoch, 22. Juni 2022 08:56

Partition Functions

Relations

- Foreign Keys
- n:m

n:m

Context Lebenszyklus

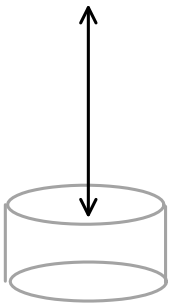
OR-Mapper

Mittwoch, 22. Juni 2022 09:33

GUI

BL

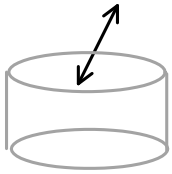
DAL



ADO.NET

Klassen

DbCommand
- Connectionstring
- CommandText : string

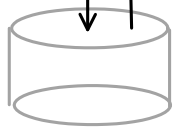


Entity Framework

Klassen

Code First

Reverse Engineering
(Database first)

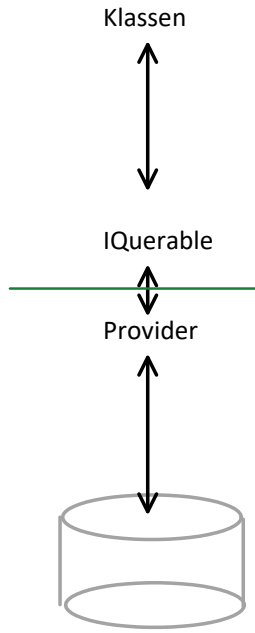


Enity Framework

Mittwoch, 22. Juni 2022 09:42

Voraussetzungen

- relationale Datenbank
- Admin-Rechte auf dem DB-Server



Entity Framework

`Microsoft.EntityFrameworkCore`
`Microsoft.EntityFrameworkCore.Tools`

Provider

SQL Server: `Microsoft.EntityFrameworkCore.SqlServer`
PostgreSQL: `Npgsql.EntityFrameworkCore.PostgreSQL`
SQLite
MySql
Oracle

Language INtegrated Query

Mittwoch, 22. Juni 2022 11:56

Kernelemente

IEnumerable
IQueryable (~ Remote IEnumerable)

IQueryable

- kompiliert die Query in SQL
- führt die SQL-Query per DbCommand (ADO.NET) aus

Deferred Execution

```
var q = context.Customers; // Query-Deklaration, keine Ausführung!

// Zugriff auf die Ergebnismenge der Query führt GetEnumerator() aus
q.ToList();
q.First();

foreach(var element in q)
{
        ...
```

Konfiguration von EntityFramework

Mittwoch, 22. Juni 2022 16:18

FluentApi

```
modelBuilder.Entity<Customer>(entity =>
            {
                entity.ToTable("customers");

                entity.Property(e => e.CustomerId)
                    .HasColumnType("string")
                    .ValueGeneratedNever()
                    .HasColumnName("customer_id");

                entity.Property(e => e.Address)
                    .HasMaxLength(60)
                    .HasColumnName("address");
```

DataAnnotations

```
    [Table("Album")]
    [Index("ArtistId", Name = "IFK_AlbumArtistId")]
    public partial class Album
    {
        public Album()
        {
            Tracks = new HashSet<Track>();
        }

        [Key]
        public int AlbumId { get; set; }
        [StringLength(160)]
        public string Title { get; set; } = null!;
```

Code-First

Donnerstag, 23. Juni 2022 14:50

1. **Nuget-Packages**
    a. Microsoft.EntityFrameworkCore
    b. Microsoft.EntityFrameworkCore.Tools
    c. Provider (zB. Npgsql.EnityFrameworkCore.Postgresql)
2. **DbContext** - Klasse, die von DbContext erbt
    a. DbSets für jede Entität, die in der DB abgebildet werden soll
    b. Ggf. OnConfiguring()
        i. Connectionstring übergeben oder hardcoden
    c. Ggf. OnModelCreating()
3. **Entitäten**
    a. Parameterloser Konstruktor
    b. Primärschlüssel (Tipp: Property namens Id oder <Klasse>Id)
    c. Navigationseigenschaften (1:n, Typ für 1, Collection für n)
4. **Datenbank**
    a. Datenbank aus ConnectionString sollte existieren (Tipp: context.Database.EnsureCreated(), DbCreator-Rechte auf dem DBS)

Migrations

Donnerstag, 23. Juni 2022 16:27

Initiale Migration anlegen!

- sobald Modell steht
- besser nicht auf die Dev-DB ausführen, funktioniert nämlich nicht, falls EnsureCreated bereits verwendet wurde

Weitere Migrations sind inkrementell zur initialen Migration

- vor neuer Migration Build ausführen!

Datenbank auf aktuellen Stand bringen mit Update Database --target Zielmigration

Durchgeführte Migrations werden __EFMigrationHistory dokumentiert

> Tipp:
> Update Database kann auch Rollbacks:
> Als Zielmigration einfach ältere Migration angeben, oder 0 für komplettes Rollback (führt Down-Methode(n) der bereits durchgeführten Migrations aus)

Daten schreiben

Freitag, 24. Juni 2022 08:59

Genau eine Funktion

context.SaveChanges()
context.SaveChangesAsync()

Schreibt alle dem context bekannten Änderungen an den Entitäten in der richtigen Reihenfolge transkationsbasiert mit Exceptionhandling in die Datenbank.

Neue Elemente - Add-Methode für das passende DbSet:

```
context.Customers.Add(abc);
```

context besitzt einen ChangeTracker, Entitäten besitzen EntityState

```
context.Entry(lastOrder).State; // EntityState
context.Entry(lastOrder).CurrentValue; // enthält die aktuellen Werte
context.Entry(lastOrder).OriginalValues; // enthält die Werte, wie sie aus der Datenbank gelesen wurden
```

Optimistic Locking (Optimistisches Sperren)

DbContext-Klasse, OnModelCreating:

```
modelBuilder.Entity<Customer>(entity =>
    {
       // Concurrency Control für Customer aktivieren
       entity.UseXminAsConcurrencyToken();
```

Concurrency Control für Customer aktivieren (Tabellenebene)

```
try
{
    context.SaveChanges();
}
catch (DbUpdateConcurrencyException ex)
{
    // Client wins - trotzdem überschreiben
    // context.Entry(alfki).OriginalValues.SetValues(context.Entry(alfki).GetDatabaseValues());
    // context.SaveChanges();
    // Database wins - aktuelle Werte laden
    context.Entry(alfki).Reload();
}
```

DbContext Lifecycle

Freitag, 24. Juni 2022 11:16

Globaler Context

siehe NorthwindDalTests
Ein großer, langlebiger Context für alle Tests

- Speicherverbrauch

Lokale Contexte (empfohlenes Vorgehen lt. Doku)

siehe Chinook
Context lokal deklariert, wenn benötigt

- EntityStates von Entitäten bei Übergabe an weitere Funktionen

DbContext konfigurieren

Freitag, 24. Juni 2022 11:36

```
private DbContextOptions<ChinookContext> options;

[SetUp]
public void Setup()
{
    options = new DbContextOptionsBuilder<ChinookContext>()
        .UseNpgsql("server=localhost;port=5432;database=chinook;user id=demo;password=Geheim123")
        .LogTo(log => LogIt(log), LogLevel.Information)
        .Options;
}


public void Test1()
{
    ChinookContext context = new ChinookContext(options);
```

OnConfiguring in generierter Context-Klasse wird damit obsolet.

Entity Type Configuration

Freitag, 24. Juni 2022 14:05

Konfiguration einer Entität auslagern in eigene Datei

AlbumConfiguration.cs

```
public class AlbumConfiguration:IEntityTypeConfiguration<Album>
{
    public void Configure(EntityTypeBuilder<Album> builder)
    {
            builder.ToTable("Album");

            builder.HasIndex(e => e.ArtistId, "IFK_AlbumArtistId");
```

DbContext.cs

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
{
    modelBuilder.ApplyConfiguration(new AlbumConfiguration());
```