Entity Framework Core

In diesem Dokument schauen wir uns an, wie wir mit EF core Daten von der Datenbank abfragen können.

Query API

Alle folgenden Abfragen beziehen sich auf die Beispielapplikation.

Für die Query API müssen wir immer folgende Schritte erst umsetzen:

1. DbContext erstellen als Datenbanksession
2. Zugriff auf eine Entität mit DbSet<T> oder Set<T>() (Repräsentiert die Datenbanktabelle)

Mit diesem DbSet<T> können wir dann eine Kombination aus den folgenden zwei Kategorien anwenden (detaillierte Beispiele auf den folgenden Seiten):

1. LINQ Abfragen (IQueryable)
    a. Sequenz als Ergebnis
        i. ToList
        ii. ToArray
        iii. ToDictionary
    b. Projektion mit Select
    c. Filtern und Sortieren
        i. Where
        ii. OrderBy / OrderByDescending
        iii. Skip und Take
    d. Skalares Ergebnis
        i. Any
        ii. Count
        iii. Sum
        iv. Average
2. EntityFramework Ergänzung:
    a. Alle unter 1. gibt es auch als async Version. Diese finden sich aber im Namensraum Microsoft.EntityFrameworkCore
3. Entity Framework spezifisch
    a. Navigationsproperty bezogen
        i. Include
        ii. ThenInclude
    b. Changetracking steuern
        i. AsNoTracking
    c. Insert / Create
        i. Add, AddRange und anschließend SaveChanges
    d. Update
        i. Update, UpdateRange und anschließend SaveChanges
    e. Delete

i. Remove, RemoveRange und anschließend SaveChanges

Zu beachten ist, dass trotz eines AddRange, UpdateRange oder eines RemoveRange teilweise keine echten Bulk operationen ausgeführt werden. Deshalb kann es hier zu performance bottle necks kommen.

Beispiele

Für alle Beispiele ist diese Zeile vorzustellen:
var context = new LeanTrainingDbContext();

1.) Alle Produkte

```
var products = context.Set<Product>().ToList();
// oder in asynchroner Methode
var products = await context.Set<Product>().ToListAsync();
```

2.) **Produkt mit der Id = 42**

```
var product = await context.Products.FindAsync(42);
// oder
var product = await context.Products.SingleOrDefaultAsync(42);
```

3.) **Filtern für Id > 10 und absteigend nach Id sortieren**

```
var products = await context.Products.Where(x => x.Id > 10)
                                     .OrderByDescending(x => x.Id)
                                     .ToListAsync();
```

4.) **Navigationproperty laden, Projektion und Dictionary als Result**

```
var result = await context.Products.Include(x => x.Parts)
                                   .ThenInclude(x => x.PartDefinition)
                                   .Select(product => new
                                   {
                                       Cost = product.Parts.Sum(part =>
                                                  part.PartDefinition.Cost),
                                       Product = product
                                    }
                .ToDictionaryAsync(key => key.Cost, value => value.Product);
```

Siehe zu den Queries ebenfalls das Dokument zur Query Evaluierung.

Außerdem gibt es noch weitere, aber diese sind die am häufigsten verwendeten.
(Join, GroupBy, All usw., Alle LINQ Operatoren die möglich sind)