



Fehlersuche und – vermeidung mit Entity Framework

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

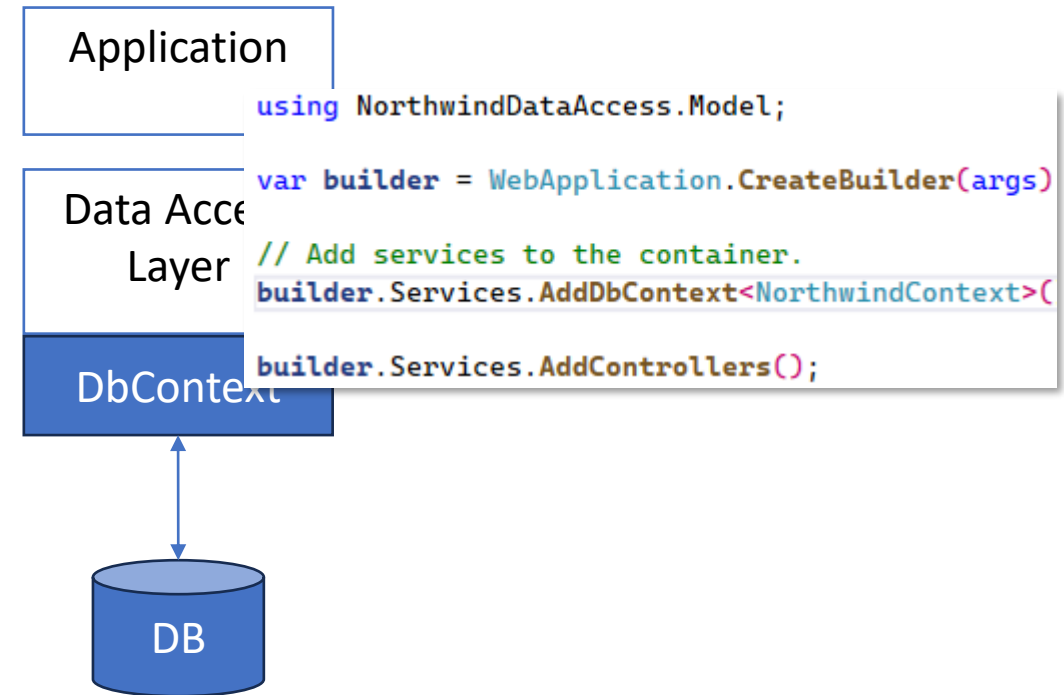
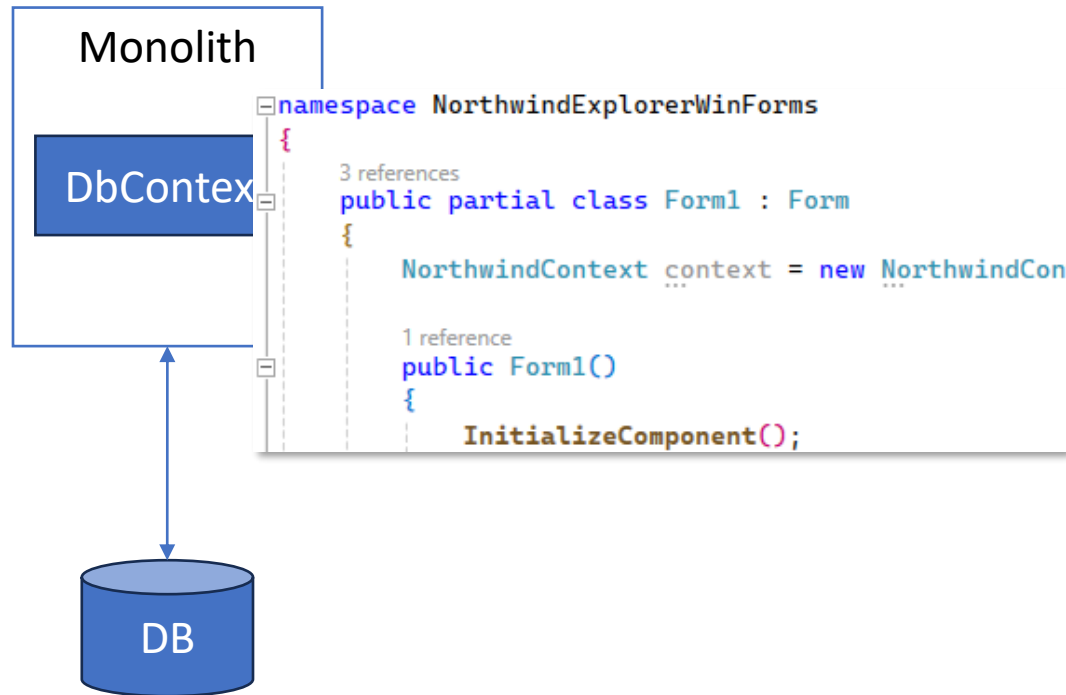


fotografiert, seit er eine Kamera halten kann

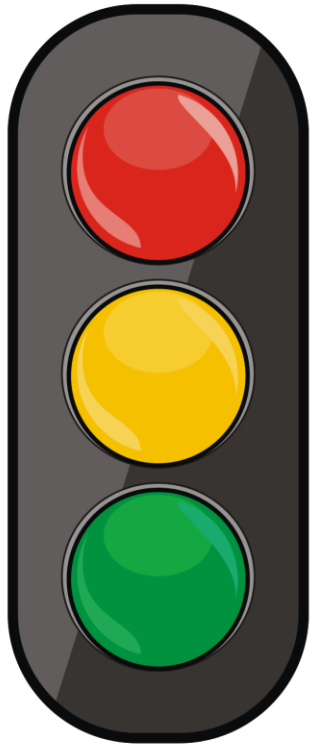


fliegt, wenn Wetter und Zeit es zulassen (TMG/SEP)

Entity Framework – Real-World Architekturen



Datenoperationen mit EF Core



Komplexe Datenstrukturen schreiben

Einfache Daten schreiben

Einfache Daten abfragen

Komplexe Datenstrukturen abfragen

Unverständliche
Exceptions?

Konfiguration?
FluentApi?

LINQ?

Refresher:

DbContext und sein ChangeTracker – Teil I

- Zustand der Entitäten (Tracking)

EntityState-Enum:

- Detached
- Added
- Deleted
- Modified
- Unchanged

```
Customer alfki = context.Customers.Find("ALFKI");  
var state = context.Entry(alfki).State;
```

- Handling von Entitäten

- DbContext/DbSet-Methoden

- .Add(Entity)
 - .Attach(Entity)
 - .Delete(Entity)
 - .Update(Entity)

- schreiben alle **NICHT** in die DB
 - regeln State der gg. Entity und abhängiger Entities!

- Änderungen persistieren (speichern)

- DbContext.SaveChanges()

- DbContext.SaveChangesAsync()

Refresher:

DbContext und sein ChangeTracker – Teil II

- Tracking-Verhalten DbContext

QueryTrackingBehavior

- .TrackAll (Default)

- .NoTracking

- .NoTrackingWithIdentityResolution

konfigurierbar auf Instanz-Ebene

```
context.ChangeTracker.QueryTrackingBehavior  
= QueryTrackingBehavior.NoTracking;
```

```
private NorthwindContext GetContext(string connectionString)  
{  
    DbContextOptions<NorthwindContext> options = (new DbContextOptionsBuilder<NorthwindContext>()  
        .UseSqlServer(connectionString)  
        .UseQueryTrackingBehavior(QueryTrackingBehavior.NoTracking)  
        .Options;  
  
    return new NorthwindContext(options);  
}
```

- Tracking-Verhalten einer Query

IQueryable Extension-Methods

- .AsTracking()

- .AsNoTracking()

- .AsNoTrackingWithIdentityResolution()

verwendbar an jedem IQueryable

Datenoperationen debuggen

- Direktfenster/Immediate Window
 - Zustände abfragen mit ?
 - Methoden direkt aufrufen
- DbContext.ChangeTracker.DebugView
 - Zustand der Entitäten ansehen
 - **Tipp:** DetectChanges()!

```
Immediate Window
? alfki.ContactName
"Maria Anders"
? context.Entry(alfki).State
Unchanged
? context.Entry(alfki).State
Modified
context.ChangeTracker.DetectChanges()
Expression has been evaluated and has no value
```

The screenshot shows the 'Locals' window in Visual Studio. The 'context' variable is expanded, showing 'ChangeTracker' and 'Context'. The 'ChangeTracker' is further expanded, showing 'AutoDetectChangesEnabled' (true), 'CascadeDeleteTiming' (Immediate), and 'DebugView'. The 'DebugView' is expanded, showing 'LongView' and 'ShortView'. The 'LongView' is selected, showing a list of customer entities. The first entity is highlighted: 'Customer {CustomerId: ALFKI} Modified'.

Text Visualizer

Expression: context.ChangeTracker.DebugView.LongView

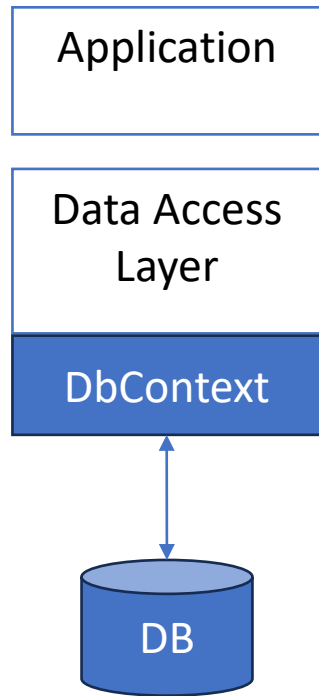
String manipulation: None

Value:

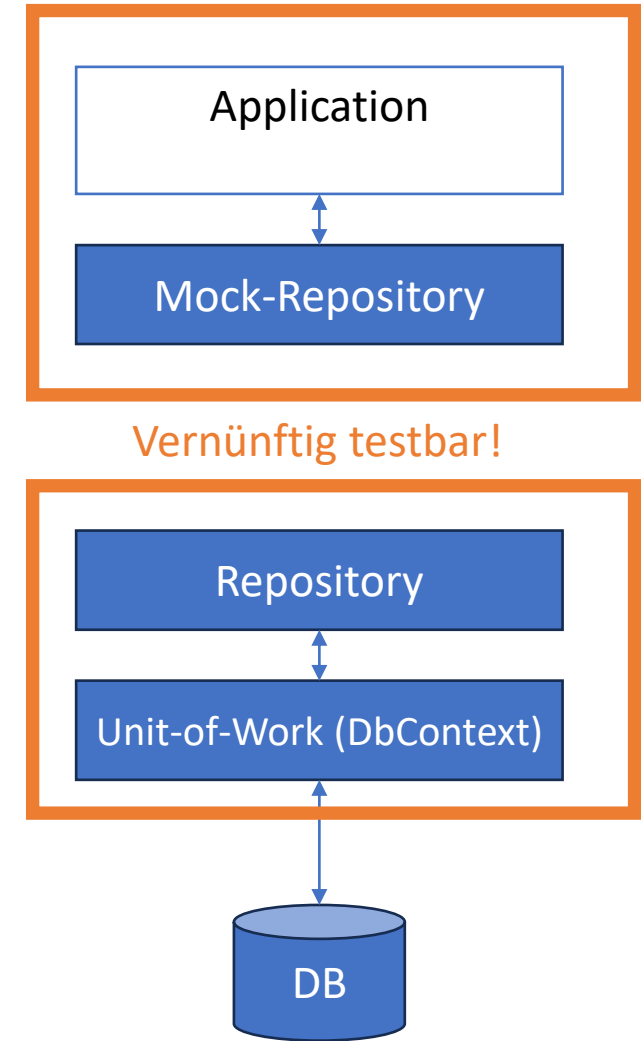
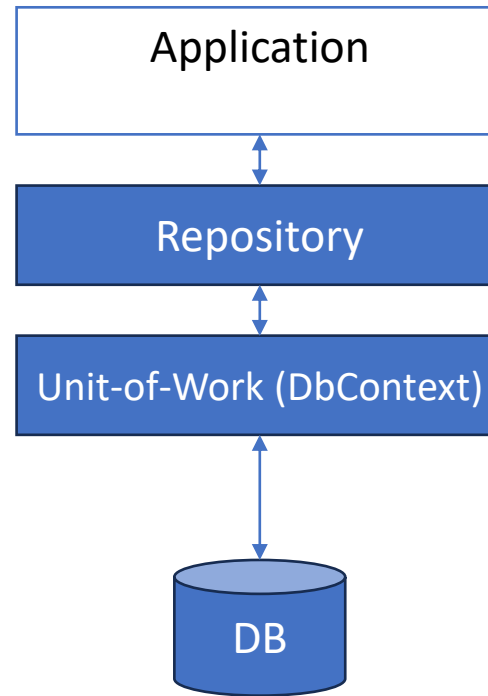
```
Customer {CustomerId: ALFKI} Modified
CustomerId: 'ALFKI' PK
Address: 'Obere Str. 57'
City: 'Berlin'
CompanyName: 'Alfreds Futterkiste'
ContactName: 'Maria Schmitt' Modified Originally 'Maria Anders'
ContactTitle: 'Sales Representative'
```

Repository Pattern Architektur

Ohne Repository



Mit Repository



Repository

- Standardisierung der CRUD-Methoden pro Entität
 - `Get(id)/GetAll()/Find()`
 - `Add()/Update()/Remove()`
- „Single point of responsibility“
 - Genau eine Stelle im Code, die die Funktionalität bereitstellt
- Generisch implementierbar

```
public interface IRepository<T> where T : class
{
    1 reference | Olaf Lischke, 2 hours ago | 1 author, 1 change
    T? GetById(int id);
    1 reference | Olaf Lischke, 2 hours ago | 1 author, 1 change
    IEnumerable<T> GetAll();
    1 reference | Olaf Lischke, 2 hours ago | 1 author, 1 change
    IEnumerable<T> FindAll(Expression<Func<T, bool>> predicate);
    2 references | 1/1 passing | Olaf Lischke, 2 hours ago | 1 author, 1 change
    void Add(T entity);
    1 reference | Olaf Lischke, 2 hours ago | 1 author, 1 change
    void AddRange(IEnumerable<T> entities);
    2 references | 1/1 passing | Olaf Lischke, 2 hours ago | 1 author, 1 change
    void Update(T entity);
    1 reference | Olaf Lischke, 2 hours ago | 1 author, 1 change
    void Remove(T entity);
}
```

- Vorteile:
 - Einheitliche Architektur
 - Testbarkeit
 - Wartbarkeit
 - Erweiterbarkeit
- Nachteil:
 - Zusätzlicher Abstraktionsschicht
 - Einschränkungen bei Nutzung ChangeTracker

Martin Fowler:
“...everything one does during a business transaction which can affect the database.”

Unit of Work

- bündelt die DB-Zugriffe
- „kennt“ alle Entitäten und deren Abhängigkeiten
- verwaltet den DbContext
- löst `SaveChanges()` aus (einzige Stelle im Code!)
- Concurrency-Handling
- Exception-Handling
- Vorteile:
 - Kapselung des DB-Zugriffs
 - Datenkonsistenz
 - Transaktionssicherheit
- Nachteile
 - Weitere Abstraktionsschicht

```
public interface IUnitOfWork : IDisposable
{
    5 references | 2/2 passing | Olaf Lischke, 3 hours ago | 1 author, 1 change
    ICustomerRepository Customers { get; }
    3 references | 1/1 passing | Olaf Lischke, 3 hours ago | 1 author, 1 change
    IOrderRepository Orders { get; }
    2 references | Olaf Lischke, 3 hours ago | 1 author, 1 change
    IOrderDetailRepository OrderDetails { get; }
    3 references | 1/1 passing | Olaf Lischke, 3 hours ago | 1 author, 1 change
    IProductRepository Products { get; }

    3 references | 2/2 passing | Olaf Lischke, 3 hours ago | 1 author, 1 change
    int Complete();
}
```

Vielen Dank!

Slides und Code-Samples auf

<https://github.com/olaflischke/basta-spring-2024>

Für spätere Fragen:

olaf.lischke@lischke-edv.de