EF Deep Dive

using C#, SQLite, SQL Server & MySQL

Donaueschingen, Januar 2024

Intro

Who am I?

Florian Schick

// Independent Software Developer //

SCHICK

SOFTWARE ENTWICKLUNG

Focus on

Full-Stack with .NET/Core, C#, Angular, Vue.js

Clean code // easy to read, easy to maintain //

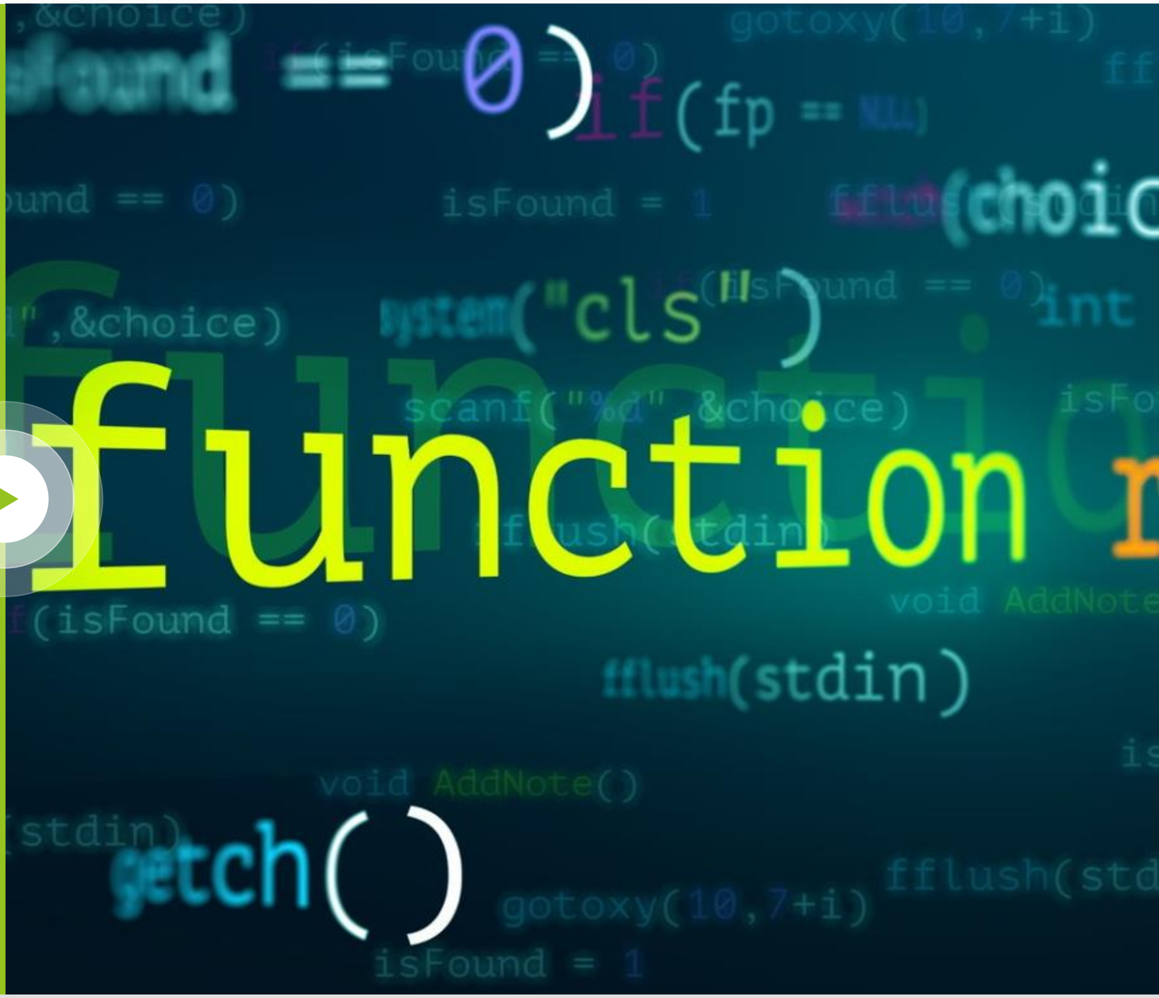
Contact

florian.schick@schick-software.de

+49 771 8979378

Intro

Agenda



Overview

Entity Framework

Entity Framework Core

Object-Database-Mapper for .NET. It supports LINQ queries, change tracking, updates, and schema migrations.

Entity Framework 6

Stable and supported product, but is no longer being actively developed.

Supported Databases

Build-In Support

- Azure SQL
- SQL Server (2012 and higher)
- SQLite (3.7 and higher)
- In-Memory (no referential integrity)
- Azure Cosmos DB SQL-API

Using 3rd-Party Libraries

- MySQL, MariaDB
- PostgreSQL
- Oracle DB 11.2 and higher
- MongoDB (Preview)

- SQL Server Compact
- Apache Kafka
- InterBase
- Firebird (3.0 and higher)
- DB2, Informix
- Microsoft Access
- ...

Overview

Workflows

SCHICK
SOFTWARE ENTWICKLUNG

Code Only (EF Core & EF)

Model First

`Models are created in code`

`DDL SQL is generated`

DB First

`Database is created externally`

`Models are generated`

Visual Designer (EF Only)

Model First

`Models are created in the designer`

`DDL SQL and models are generated`

DB First

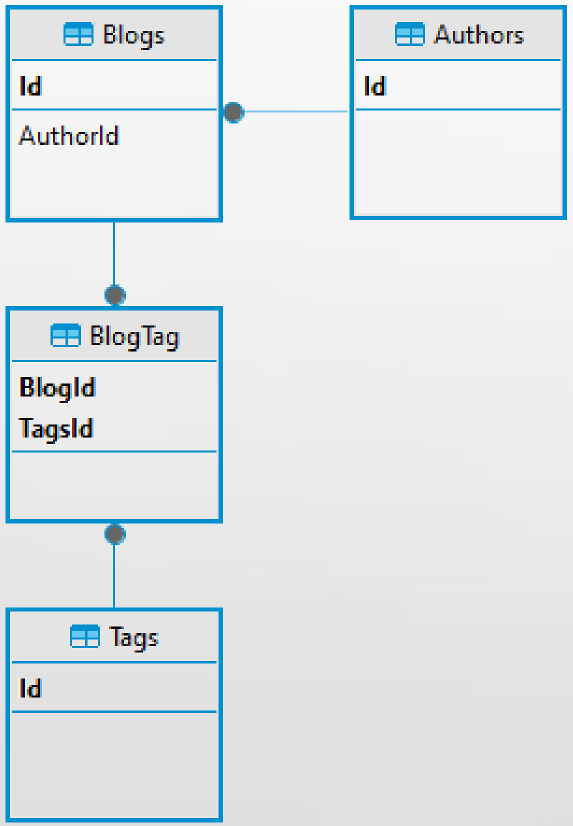
`Database is created in the designer`

`DDL SQL and models are generated`

DDL = Data Definition Language, CREATE/ALTER TABLE, …

Configuration

Model



```
public class Blog
{
    public Guid Id { get; set; }

    public string Title { get; set; }

    public Guid AuthorId { get; set; }
    public Author Author { get; set; }

    public List<Tag> Tags { get; set; }
}


public class Author
{
    public Guid Id { get; set; }
    public string Name { get; set; }
}


public class Tag
{
    public Guid Id { get; set; }
    public string Name { get; set; }
}
```

Configuration

DB Context

```csharp
public sealed class DeepDiveDbContext : MultiDbContext
{
    public DbSet<Blog> Blogs { get; set; }

    public DbSet<Author> Authors { get; set; }

    public DbSet<Tag> Tags { get; set; }

    public DeepDiveDbContext(DatabaseType databaseType, string connectionString)
        : base(databaseType, connectionString) { }

    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        base.OnModelCreating(modelBuilder);

        modelBuilder.Entity<Blog>()
            .HasOne(blog => blog.Author)
            .WithMany()
            .OnDelete(DeleteBehavior.Restrict);

        modelBuilder.Entity<Blog>()
            .HasMany(blog => blog.Tags)
            .WithMany();
    }
}
```

Configuration

Migration

SCHICK
SOFTWARE ENTWICKLUNG

```
public class Initial : Migration
{
    protected override void Up(MigrationBuilder migrationBuilder)
    {
        migrationBuilder.CreateTable(
            name: "Blogs",
            columns: table => new
            {
                Id = table.Column<Guid>(type: "uniqueidentifier", nullable: false),
                ...
            },
            constraints: table =>
            {
                table.PrimaryKey("PK_Blogs", x => x.Id);
                table.ForeignKey(
                    name: "FK_Blogs_Authors_AuthorId",
                    column: x => x.AuthorId,
                    principalTable: "Authors",
                    principalColumn: "Id",
                    onDelete: ReferentialAction.Cascade);
            });
    }

    ...
}
```

Package Manager Console

```
PM> Add-Migration Initial
```

Configuration

Migration SQL

```
BEGIN TRANSACTION;
GO

CREATE TABLE [Blogs] (
    [Id] uniqueidentifier NOT NULL,
    [Title] nvarchar(max) NOT NULL,
    [AuthorId] uniqueidentifier NOT NULL,
    [Created] datetime2 NOT NULL,
    [Published] datetime2 NULL,
    CONSTRAINT [PK_Blogs] PRIMARY KEY ([Id]),
    CONSTRAINT [FK_Blogs_Authors_AuthorId] FOREIGN KEY ([AuthorId])
        REFERENCES [Authors] ([Id]) ON DELETE CASCADE
);
GO

...

CREATE INDEX [IX_Blogs_AuthorId] ON [Blogs] ([AuthorId]);
GO

COMMIT;
GO
```

Package Manager Console

```
PM> Update-Database
or
PM> Script-Migration
or
dbContext.Database.Migrate();
```

CRUD Operations

Showtime

Change Tracking

Snapshot

// Standard //

By default, EF Core creates a snapshot of every entity's property values when it is first tracked by a DbContext instance. The values stored in this snapshot are then compared against the current values of the entity in order to determine which property values have changed.

INotifyPropertyChange & INotifyPropertyChanging

```
public partial class Blog : ObservableValidator
{
    [Required, ObservableProperty]
    private required string _title;
}

public sealed class DeepDiveDbContext : MultiDbContext
{
    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        modelBuilder.HasChangeTrackingStrategy(
           ChangeTrackingStrategy.ChangingAndChangedNotifications
        );
    }
}
```

Package Manager Console

```
PM> Install-Package CommunityToolkit.Mvvm
```

Transactions

SaveChanges()

// Standard //

By default, if the database provider supports transactions, all changes in a single call to SaveChanges are applied in a transaction.

Explicit Transaction

```
using var transaction = await dbContext.Database.BeginTransaction();
```

External Transaction

```
using var connection = new SqlConnection();
connection.Open();
using var transaction = connection.BeginTransaction();
dbContext.Database.UseTransaction(transaction);
```

Column Configuration

Showtime

SCHICK

SOFTWARE ENTWICKLUNG

Column Configuration

Value Conversions

```
public sealed class DeepDiveDbContext : MultiDbContext
{
    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        base.OnModelCreating(modelBuilder);

        var enumToStringConverter = new EnumToStringConverter<BlogStatus>();
        modelBuilder.Entity<Blog>()
            .Property(e => e.Status)
            .HasConversion(enumToStringConverter);

        // Conversion can also be done with lambdas:
        modelBuilder.Entity<Blog>()
           .Property(e => e.Status)
           .HasConversion(
               blogStatus => blogStatus.ToString(),
               valueString => Enum.Parse<BlogStatus>(valueString)
           );

        // Or using a built-in converter:
        modelBuilder.Entity<Blog>()
           .Property(x => x.Status)
           .HasConversion<string>();
    }
}
```

JSON Columns

Showtime

JSON Columns

Configure JSON-Mapping

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
{
    base.OnModelCreating(modelBuilder);

    //...

    modelBuilder
        .Entity<Author>()
        .OwnsOne(
            author => author.Address,
            navigationBuilder =>
            {
                if (DatabaseType != DatabaseType.MySql)
                    navigationBuilder.ToJson();
            });
}
```

SQL Queries and Stored Procedures

```
-- SQL Server
CREATE PROCEDURE GetPublishedBlogs
AS
    SET NOCOUNT ON;
    SELECT * FROM [Blogs] WHERE [Status] = 'Published'
    RETURN;
GO


-- MySQL
CREATE PROCEDURE `GetPublishedBlogs`()
BEGIN
    SELECT * FROM `Blogs` WHERE `Status` = 'Published';
END
```

SQL Queries

Showtime

SCHICK
SOFTWARE ENTWICKLUNG

High-Level Interception

```
public sealed class DeepDiveDbContext : MultiDbContext
{
    public override Task<int> SaveChangesAsync(bool acceptAllChangesOnSuccess, CancellationToken cancellationToken = default)
    {
        HandleBlogCreated();
        return base.SaveChangesAsync(acceptAllChangesOnSuccess, cancellationToken);
    }

    private void HandleBlogCreated()
    {
        ChangeTracker.DetectChanges();
        var trackedBlogs = ChangeTracker.Entries<Blog>().ToList();
        foreach (var blog in trackedBlogs)
        {
            switch (blog.State)
            {
                case EntityState.Added:
                    blog.Entity.Created = DateTime.UtcNow;
                    break;
                default:
                    blog.Property(x => x.Created).IsModified = false;
                    break;
            }
        }
    }
}
```

Low-Level Interception

```
public class DeepDiveDbContext : MultiDbContext
{
    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
    {
        base.OnConfiguring(optionsBuilder);
        optionsBuilder.AddInterceptors(new TableLockInterceptor());
    }
}

public class TableLockInterceptor : DbCommandInterceptor
{
    public const string USE_TABLE_LOCK = "Use table lock";

    public override InterceptionResult<DbDataReader> ReaderExecuting(IDbCommand command, ...)
    {
        AddTableLockIfRequested(command);
        return result;
    }

    private static void AddTableLockIfRequested(IDbCommand command)
    {
        if (command.CommandText.StartsWith($"-- {USE_TABLE_LOCK}"))
            command.CommandText += " WITH (TABLOCKX, HOLDLOCK)";
    }
}
```

Interception

Showtime

SCHICK

SOFTWARE ENTWICKLUNG

Custom Functions

Database Functions

```
-- SQL Server
-- Transforms „Hello World“ into „H**** W*****“
CREATE FUNCTION dbo.Obfuscate(@input NVARCHAR(MAX)) RETURNS NVARCHAR(MAX) AS
BEGIN
    DECLARE @result NVARCHAR(MAX);

    SELECT @result = STRING_AGG(LEFT(WordList.Word, 1) + REPLACE(SPACE(LEN(WordList.Word) - 1), ' ', '*'), ' ')
    FROM (SELECT value AS Word FROM STRING_SPLIT(@input, ' ')) AS WordList;

    RETURN @result;
END



-- MySQL
-- Transforms „Hello World“ into „H**** W*****“
CREATE FUNCTION `Obfuscate`(`input` LONGTEXT)
    RETURNS LONGTEXT
            DETERMINISTIC
BEGIN
    RETURN REGEXP_REPLACE(`input`, '(?<!^)(?<!\\s)\\w', '*');
END;
```

Custom Functions

Database Functions

SCHICK
SOFTWARE ENTWICKLUNG

```
public static class StringExtensions
{
    private static readonly MethodInfo _obfuscate = typeof(StringExtensions).GetMethod(nameof(Obfuscate), new[] { typeof(string) })!;

    public static string Obfuscate(this string input)
        => Regex.Replace(input, @"(?<!(^|\s))\w", "*");

    public static void RegisterObfuscateFunction(this ModelBuilder modelBuilder)
    {
        if (databaseType == DatabaseType.Sqlite)
            throw new NotSupportedException("SQLite does not support user defined functions");

        modelBuilder
           .HasDbFunction(_obfuscate)
           .HasSchema("dbo")
           .HasName(nameof(Obfuscate))
           .IsNullable();
    }
}
```

Custom Functions

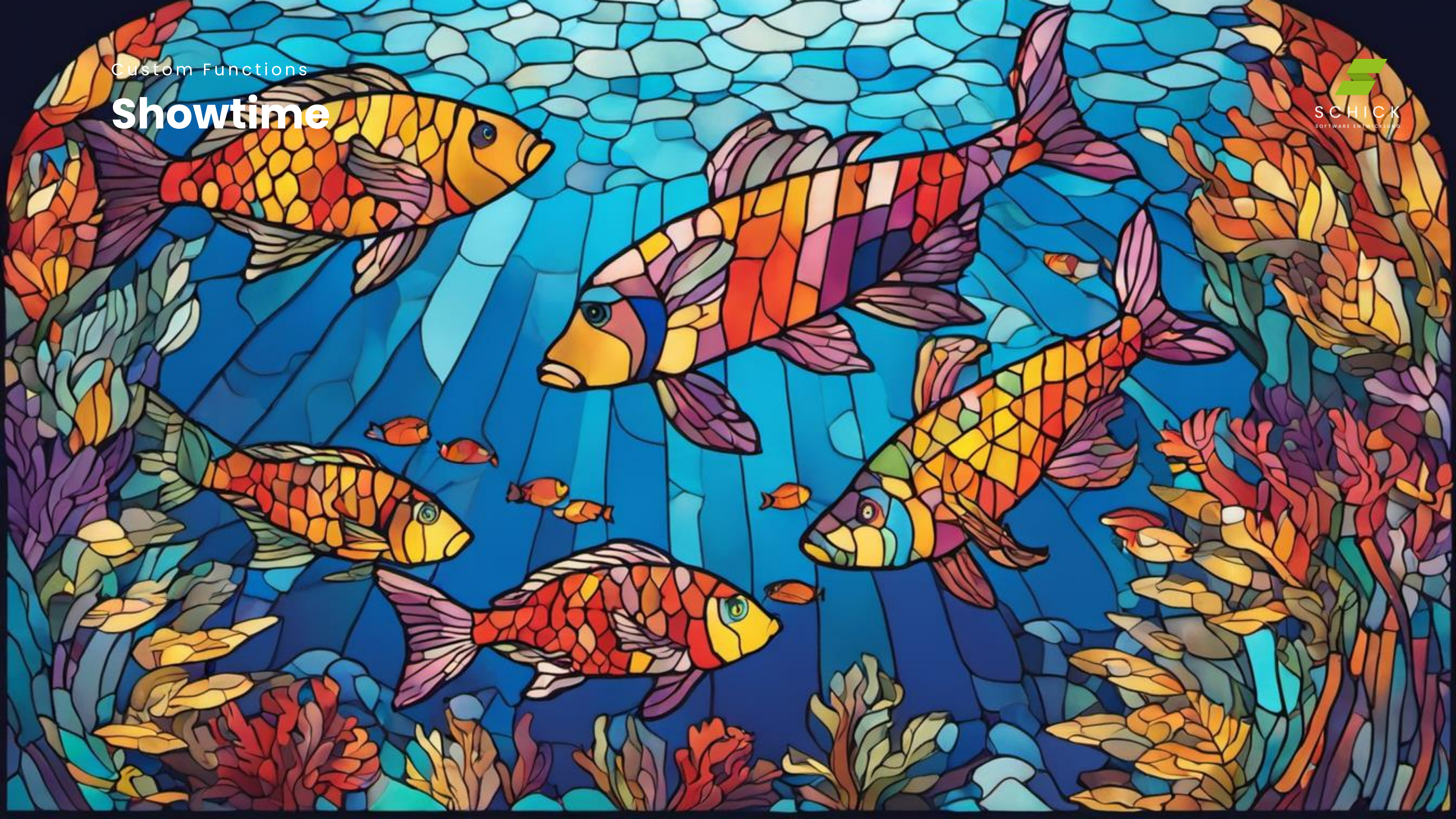
In-Memory Functions (SQLite)

```
public static class StringExtensions
{
    public static string Obfuscate(this string input)
        => Regex.Replace(input, @"(?<!(^|\s))\w", "*");

    public static void RegisterObfuscateFunction(this DbContextOptionsBuilder optionsBuilder)
        => optionsBuilder.AddInterceptors(new ObfuscateFunctionsInterceptor());

    private class ObfuscateFunctionsInterceptor : DbConnectionInterceptor
    {
        public override void ConnectionOpened(DbConnection connection, ConnectionEndEventData eventData)
        {
            base.ConnectionOpened(connection, eventData);
            CreateFunctionObfuscate((SqliteConnection)connection);
        }

        private static void CreateFunctionObfuscate(SqliteConnection connection)
        {
            if (databaseType != DatabaseType.Sqlite)
                throw new NotSupportedException("Only SQLite supports in-memory functions");

            connection.CreateFunction(nameof(Obfuscate), (Func<string, string>)Obfuscate, isDeterministic: true);
        }
    }
}
```

Custom Functions

Showtime

SCHICK
SOFTWARE ENTWICKLUNG



Generate SQL Statements

```
public static class MyDbFunctionsExtensions
{
    private static readonly MethodInfo _guidLike = typeof(MyDbFunctionsExtensions).GetMethod(nameof(Like))!;

    public static bool Like(this Guid guid, string? pattern)
    {
        if (string.IsNullOrEmpty(pattern))
            return false;

        return Regex.IsMatch(guid.ToString(), pattern.ToRegexPattern());
    }

    public static void RegisterGuidLikeFunction(this ModelBuilder modelBuilder)
    {
        modelBuilder
            .HasDbFunction(_guidLike)
            .HasTranslation(CreateLikeExpression);
    }

    private static SqlExpression CreateLikeExpression(IReadOnlyList<SqlExpression> parameters)
        => new LikeExpression(parameters[0], parameters[1], null, null);
}
```

SQL Expressions

Showtime

SCHICK
SOFTWARE ENTWICKLUNG

Thank you

www.schick-software.de