mcc-
tomsk
.de

Entity Framework Core: tips and tricks

Артур Дробинский
МЦЦ Томск

About Us

MCC Tomsk

Cloud platform

for tele-medicine

Microservices

Message Broker

SOA

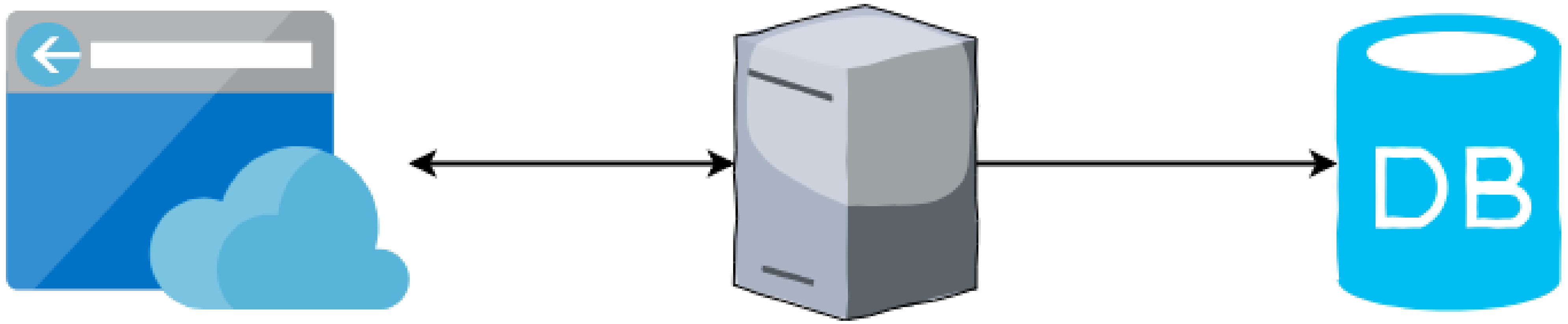
.Net Core

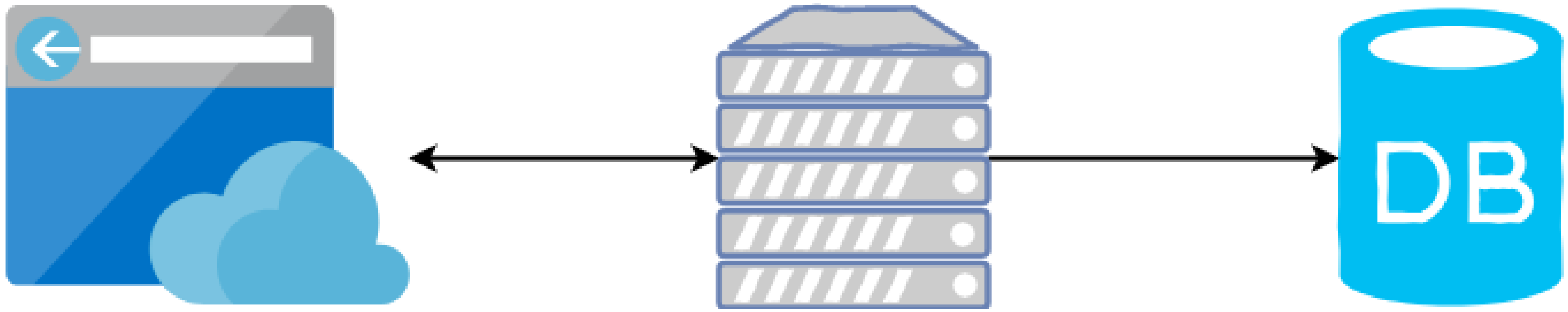
React/Redux

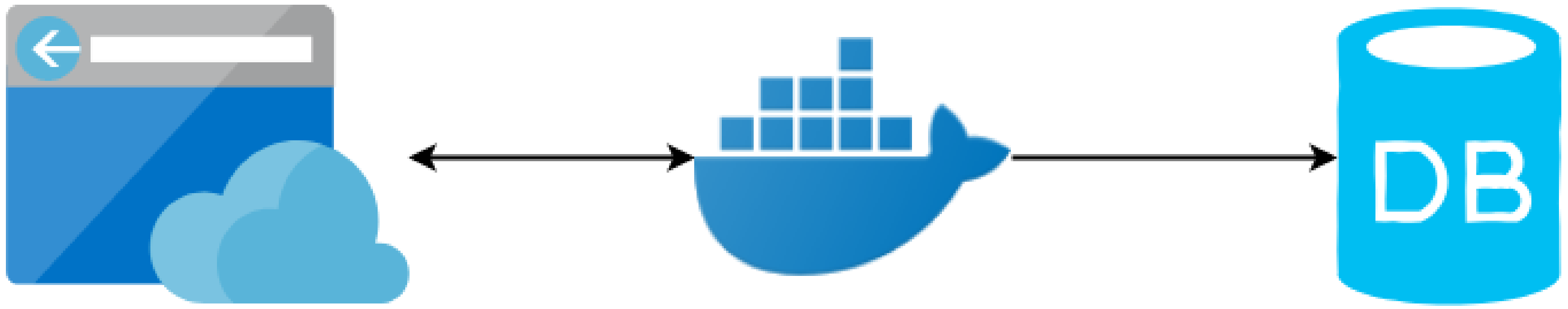
mcc-tomsk.de

Entity Framework

Core


DB


DB


DB

ADO.NET

```
using (SqlConnection connection =
    new SqlConnection(connectionString))
{
    SqlCommand command = new
SqlCommand(queryString, connection);

    connection.Open();
    SqlDataReader reader = command.ExecuteReader();
    reader.Read();
    Console.WriteLine("\t{0}\t{1}\t{2}",
            reader[0], reader[1], reader[2]);
}
```

Entity Framework Core

```
public class Comment
{
    public int Id { get; set; }
    public string Text { get; set; }
    public bool IsDeleted { get; set; }
    public Post Post { get; set; }
}
```

```
var context = new MyContext(options);
var comment = context.Comments.First();

Console.WriteLine("\t{0}\t{1}", comment.Text,
comment.Id);
```

Dapper

Dapper - a simple object mapper for .Net

Entity Framework

Core

```
var sql =
    @"select * from #Posts p
left join #Users u on u.Id = p.OwnerId
Order by p.Id";

var data = connection.Query<Post, User,
Post>(sql, (post, user) => { post.Owner =
user; return post; });
var post = data.First();
```

```
var context = new MyContext(options);
var comment = context.Comments
    .Include(x => x.Post)
    .Where(x => x.Id > 5)
    .Where(x => !x.Post.IsDeleted)
    .OrderBy(x => x.Text)
    .First();

Console.WriteLine("\t{0}\t{1}",
comment.Post.Title, comment.Text);
```

MCC mcc-tomsk.de

Pitfalls

Entities

```
public class Post
{
    public int Id { get; private set; }
    public string Title { get; set; }
    public Blog Blog { get; set; }
    public bool IsDeleted { get; set; }
}

public class Comment
{
    public int Id { get; set; }
    public string Text { get; set; }
    public Post Post { get; set; }
}
```

```
public class Blog
{
    public int Id { get; private set; }
    public string Name { get; set; }
    public ICollection<Post> Posts { get; }

    private Blog() { }
    public Blog(string name)
    {
        Name = name;
    }
}
```

SELECT N+1

```
var posts = context.Posts.ToList();
//SELECT * FROM Posts

foreach (var post in posts)
{
    var commentsCount = context.Comments
        .Count(x => !x.IsDeleted && x.Post.Id == post.Id);
    //SELECT COUNT(1) FROM Comments WHERE IsDeleted = 0 AND PostId = {0}
}
```

SELECT N+1

```
posts = context.Posts.Include(x => x.Comments).ToList();
//SELECT * FROM Posts WHERE IsDeleted = 0
//SELECT * FROM Comments
//INNER JOIN (SELECT Id FROM Posts WHERE IsDeleted = 0) AS T ON PostId = T.Id

foreach (var post in posts)
{
    var commentsCount = post.Comments.Count();
}
```

SELECT N+1

```
var posts3 = context.Posts.Select(x => new
{
    PostTitle = x.Title,
    PostId = x.Id,
    CommentsCount = x.Comments.Count(),
}).ToList();
//SELECT Title, Id, (SELECT COUNT(*)
//FROM Comments AS c WHERE [x].[Id] = [c].[PostId]) AS CommentsCount
//FROM Posts AS x

foreach (var post in posts3)
{
    var commentsCount = post.CommentsCount;
}
```

SELECT N+1

```
var posts4 = context.Posts.Select(x => new
{
    Post = x,
    CommentsCount = x.Comments.Count(),
}).ToList();
//SELECT * FROM Posts
//SELECT Count(*) FROM Comments WHERE PostId = {0}

foreach (var post in posts3)
{
    var commentsCount = post.CommentsCount;
}
```

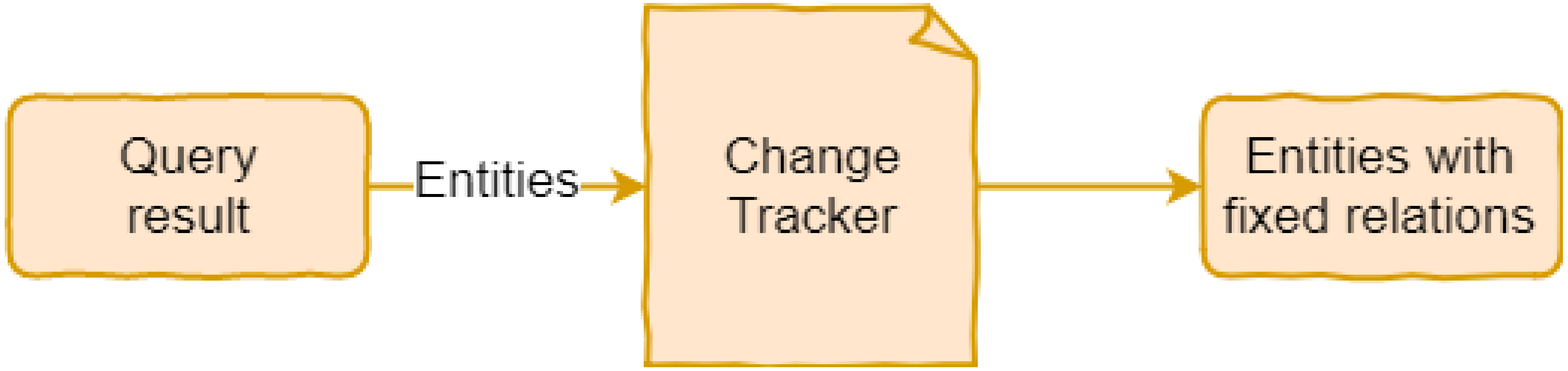
SELECT N+1

```
var posts5 = context.Posts.Select(x => new
{
    PostId = x.Id,
    Blog = x.Blog,
    CommentsCount = x.Comments.Count(),
}).ToList();
//SELECT [Blogs].*, (SELECT COUNT(*) FROM[Comments] AS[c]
//WHERE[x].[Id] = [c].[PostId]  ) AS[CommentsCount]
//FROM[Posts] AS[x]
//LEFT JOIN[Blogs] AS[x.Blog] ON[x].[BlogId] = [x.Blog].[Id]

foreach (var post in posts3)
{
    var commentsCount = post.CommentsCount;
}
```

Can I join you?
comic.browserling.com
An SQL query walks into a bar and sees two tables.
He walks up to them and says "Can I join you?"

Change Tracker

```
context = _createDatabase();
var blog = context.Blogs.FirstOrDefault();
Assert.Null(blog.Posts);

context.Posts.Where(x => x.BlogId == blog.Id).ToList();
Assert.Null(blog.Posts);
```


Query result
Entities
Change Tracker
Entities with fixed relations

ToList()

```
var blog = context.Blogs.Include(x => x.TimeRegion).ToList();

//SELECT * FROM Blogs
//LEFT JOIN TimeRegion ON TimeRegion.Id = Blogs.TimeRegionId
```

```
context.TimeRegions.ToList();
var blog = context.Blogs.ToList();

//SELECT * FROM TimeRegions
//SELECT * FROM Blogs
```

GroupBy

```
 var result = context.Posts
                   .GroupBy(p => p.BlogId)
                   .Select(g => new { BlogId = g.Key, PostCount = g.Count() })
                   .ToList();

//SELECT[p].[BlogId] AS[BID], COUNT(*) AS[cnt]
//FROM[Posts] AS[p]
//GROUP BY[p].[BlogId]
```

GroupBy

```
 var result = context.Posts
                .GroupBy(p => p.BlogId)
                .Select(g => new { BID = g.Key, cnt = g.Count(x => !x.IsDeleted) })
                .ToList();

var result = context.Posts
                .GroupBy(p => p.Blog)
                .Select(g => new { Url = g.Key.Name, Count = g.Count() })
                .ToList();


var optionsBuilder = new DbContextOptionsBuilder<MyContext>()
        .ConfigureWarnings(x => x.Throw(RelationalEventId.QueryClientEvaluationWarning));
```

Aggregates

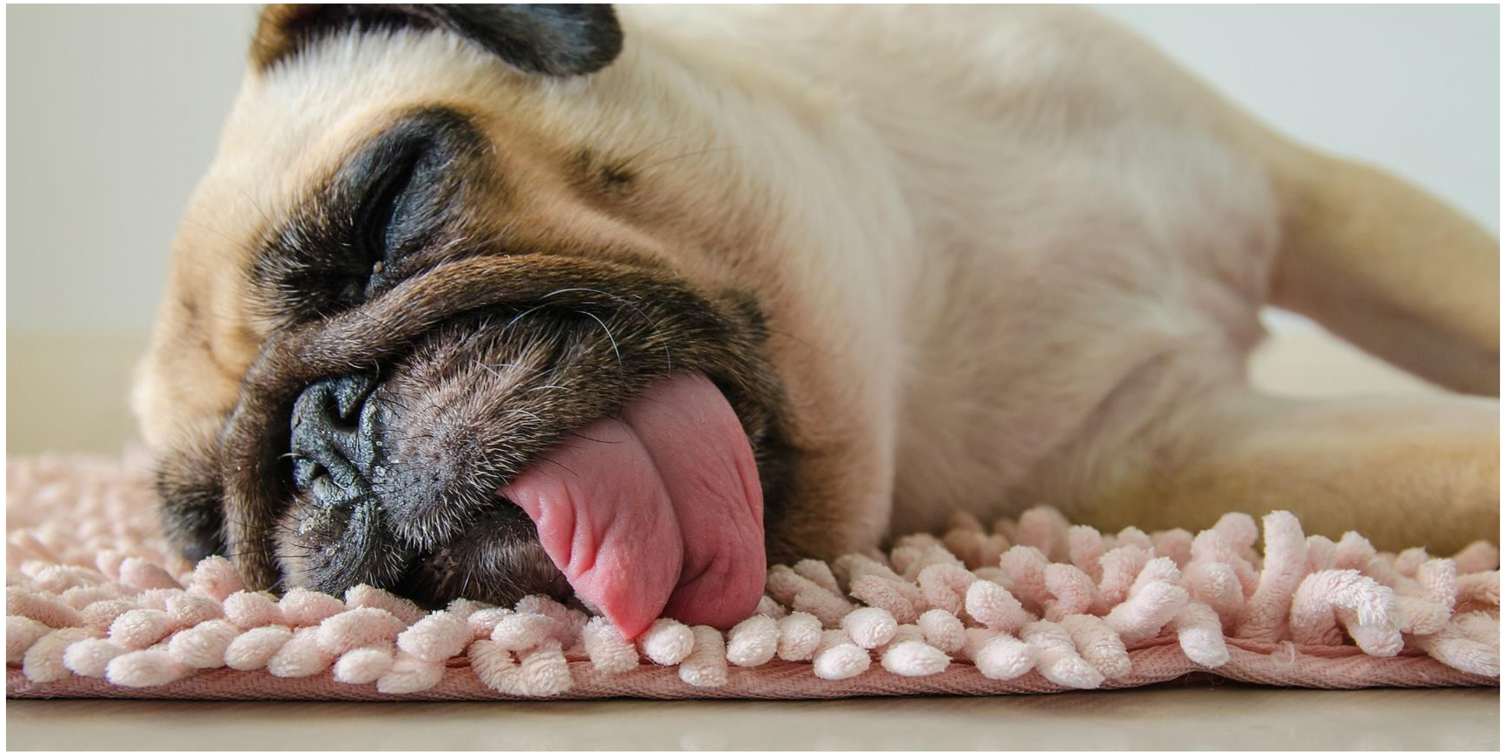
```
var data = context.Blogs.Select(x => new
{
    x.Id,
    MaxPostId = x.Posts.Max(post => post.Id)
}).ToList();
```

```
InvalidOperationException: Sequence contains no elements.
    Microsoft.EntityFrameworkCore.Query.QueryMethodProvider.GetResult<TResult>(IEnumerable
    <ValueBuffer> valueBuffers, bool throwOnNullResult)
```

Min/Max/Average

```
var data = context.Blogs.Select(x => new
{
    x.Id,
    MaxPostId = x.Posts.Max(post => (int?)post.Id)
}).ToList();
```

mcc-
tomsk
.de



Tired? ☺

Private constructors & Collection initializers

```
public class Blog
{
    public int Id { get; private set; }
    public string Name { get; set; }

    private Blog() { }
    public Blog(string name)
    {
        Name = name;
    }
}
```

Private constructors & Collection initializers

```
public class Blog
{
    public int Id { get; private set; }

    public ICollection<Post> Posts { get; }
        = new List<Post>(); //don't do that
}
```

```
var blog = context.Blogs.FirstOrDefault();
var postCount = blog.Posts.Count();
```

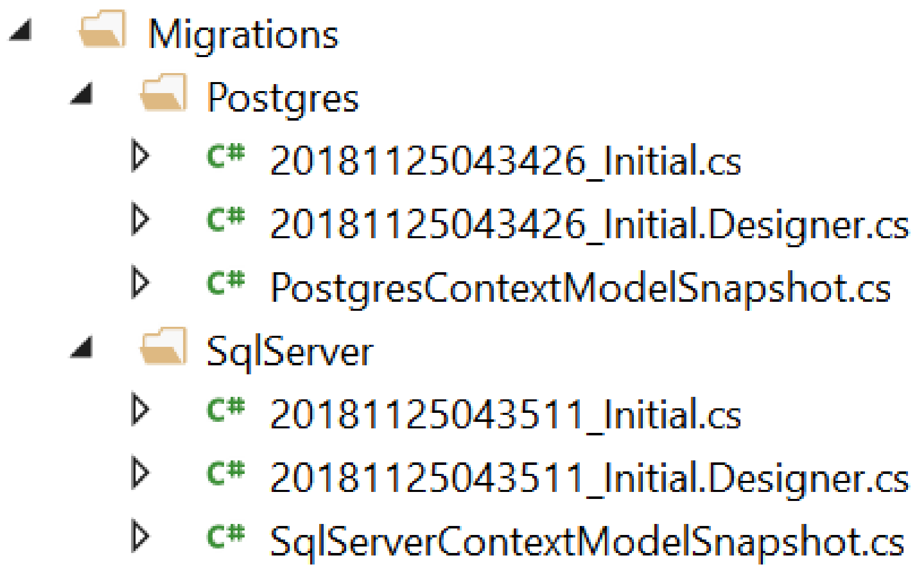
Lazy Loading

```
var optionsBuilder = new DbContextOptionsBuilder<MyContext>()
      .UseLazyLoadingProxies();


var blog = context.Blogs.FirstOrDefault();
//SELECT * FROM Blogs LIMIT 1

var postCount = blog.Posts.Count();
// SELECT * FROM Posts WHERE BlogId = 1
```

- Since 2.1
- No private constructors (but could be protected)
- Not recommended

Target Multiple Databases

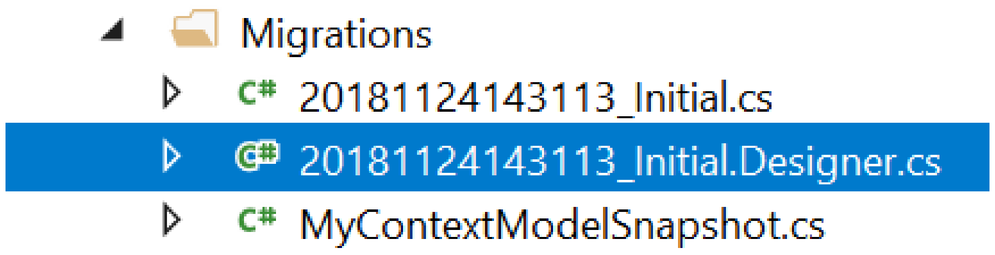
```csharp
public abstract class MainContext : DbContext
{
    public DbSet<Comment> Comments { get; set; }
    protected MainContext(DbContextOptions<MainContext> options) : base(options) { }
}

public class PostgresContext : MainContext
{
    public PostgresContext(DbContextOptions<MainContext> options) : base(options) { }
}

public class SqlServerContext : MainContext
{
    public SqlServerContext(DbContextOptions<MainContext> options) : base(options) { }
}
```



```
Migrations
    Postgres
        20181125043426_Initial.cs
        20181125043426_Initial.Designer.cs
        PostgresContextModelSnapshot.cs
    SqlServer
        20181125043511_Initial.cs
        20181125043511_Initial.Designer.cs
        SqlServerContextModelSnapshot.cs
```

Migrations.Designer.cs (could be deleted)

```
[Migration("20181124143113_Initial")]
partial class Initial
{
    protected override void BuildTargetModel(ModelBuilder modelBuilder)
    {
#pragma warning disable 612, 618
        modelBuilder
            .HasAnnotation("ProductVersion", "2.2.0-preview3-35497")
            .HasAnnotation("Relational:MaxIdentifierLength", 128)
            .HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

        modelBuilder.Entity("XUnitTestProject1.Blog", b =>
            {
                b.Property<int>("Id")
                    .ValueGeneratedOnAdd()
                    .HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);
                b.Property<string>("Name");
                b.HasKey("Id");
                b.ToTable("Blogs");
            });

        modelBuilder.Entity("XUnitTestProject1.Comment", b =>
            {
                b.Property<int>("Id")
                    .ValueGeneratedOnAdd()
                    .HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);
                b.Property<bool>("IsDeleted");
                b.Property<int?>("PostId");
                b.Property<string>("Text");
                b.HasKey("Id");
                b.HasIndex("PostId");
                b.ToTable("Comments");
            });
```

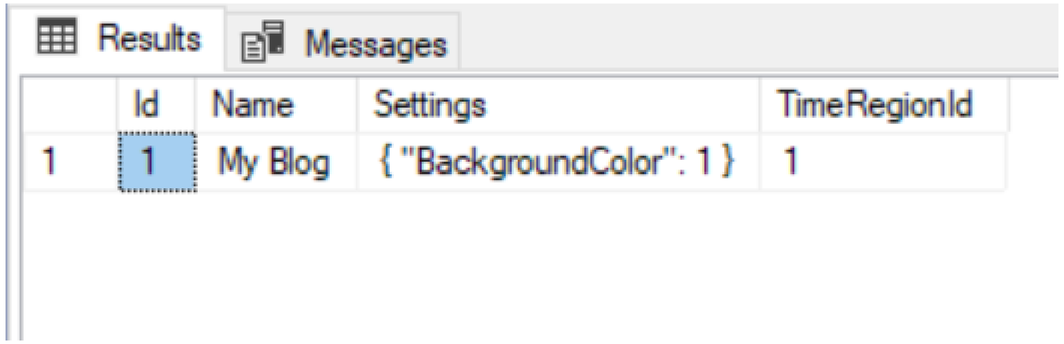

Migrations
20181124143113_Initial.cs
20181124143113_Initial.Designer.cs
MyContextModelSnapshot.cs

Value Convertors

- EnumToStringConverter
- DateTimeToTicksConverter
- Objects as JSON serialized string
  (not built-in)

Value Convertors (JSON serialized-objects)

```
public class Blog {
    public int Id { get; private set; }
    …
    public BlogSettings Settings { get; set; }
}
public class BlogSettings {
    public int BackgroundColor { get; set; }
}
```



	Id	Name	Settings	TimeRegionId
1	1	My Blog	{ "BackgroundColor": 1 }	1

```
protected override void OnModelCreating(ModelBuilder modelBuilder) {
    modelBuilder.Entity<Blog>().Property(e => e.Settings)
        .HasConversion(
            v => JsonConvert.SerializeObject(v),
            v => JsonConvert.DeserializeObject<BlogSettings>(v));
}
```

Query Filter (Soft Delete)

```
public class Post
{
    public int Id { get; private set; }
    public string Title { get; set; }
    public Blog Blog { get; set; }
    public bool IsDeleted { get; set; }
}
```

```
var blogs = context.Posts.ToList();
//SELECT * FROM Posts WHERE IsDeleted = 0
```

```
protected override void OnModelCreating(ModelBuilder modelBuilder) {
  modelBuilder.Entity<Post>().HasQueryFilter(p => !p.IsDeleted);
}
```

mcc-tomsk.de

Testing

In Memory

```
var options = new DbContextOptionsBuilder<MyContext>()
        .UseInMemoryDatabase(Guid.NewGuid().ToString())
        .Options;

_createDatabase = () => new MyContext(options);
```

In Memory & transactions

```
 var options = new DbContextOptionsBuilder<MyContext>()
                .UseInMemoryDatabase(Guid.NewGuid().ToString())
                .ConfigureWarnings(x =>
x.Ignore(InMemoryEventId.TransactionIgnoredWarning))
                .Options;
```

In Memory & constraints??? SQLite!

```
var connection = new SqliteConnection("DataSource=:memory:");
connection.Open(); //an open connection is required for database to exist
var optionsBuilder = new DbContextOptionsBuilder<MyContext>()
        .ConfigureWarnings(x =>
x.Throw(RelationalEventId.QueryClientEvaluationWarning))
        .UseLazyLoadingProxies()
    ;
var options = optionsBuilder.UseSqlite(connection).Options;
```

Migrations

```
[Fact]
public void Migrations_RunAll_MigrationsSuccessfullyApplied()
{
    using (var connection = new SqliteConnection("DataSource=:memory:"))
    {
        connection.Open(); //an open connection is required for database to exist

        var optionsBuilder = new DbContextOptionsBuilder<MyContext>();
        var options = optionsBuilder.UseSqlite(connection).Options;

        var context = new MyContext(options);
        context.Database.Migrate();
    }
}
```

Migrations

```
var context = new MyContext(options);
context.Database.Migrate();

foreach (var entityType in context.Model.GetEntityTypes())
{
    GetDbSet(context, entityType.ClrType).First();
}
```

```
System.InvalidOperationException : Error querying entity 'Post': SQLite Error 1: 'no
such column: p.Date'.
```

mcc-
tomsk
.de

Integration

Z.EntityFramework.Plus.EFCore – Bulk Operations

```
_dbContext.Comments.Where(x => !x.IsDeleted)
    .Update(x => new Comment()
    {
        IsDeleted = true,
    });
```

```
UPDATE A
SET A.[IsDeleted] = @zzz_BatchUpdate_0
FROM[Comments] AS A
INNER JOIN(SELECT[x].[Id], [x].[IsDeleted],
[x].[PostId], [x].[Text]
FROM [Comments] AS [x]
WHERE [x].[IsDeleted] = 0
    ) AS B ON A.[Id] = B.[Id]
```

BulkUpdate does not work with InMemoryDatabase

Z.EntityFramework.Plus.EFCore – FromCache

```
var timeRegions = _dbContext.TimeRegions.FromCache();
_dbContext.AttachRange(timeRegions);
var blogs = _dbContext.Blogs.ToList();

Console.WriteLine(string.Join(", ", blogs.Select(blog => $"{blog.Name}:
{blog.TimeRegion.Offset}")));
```

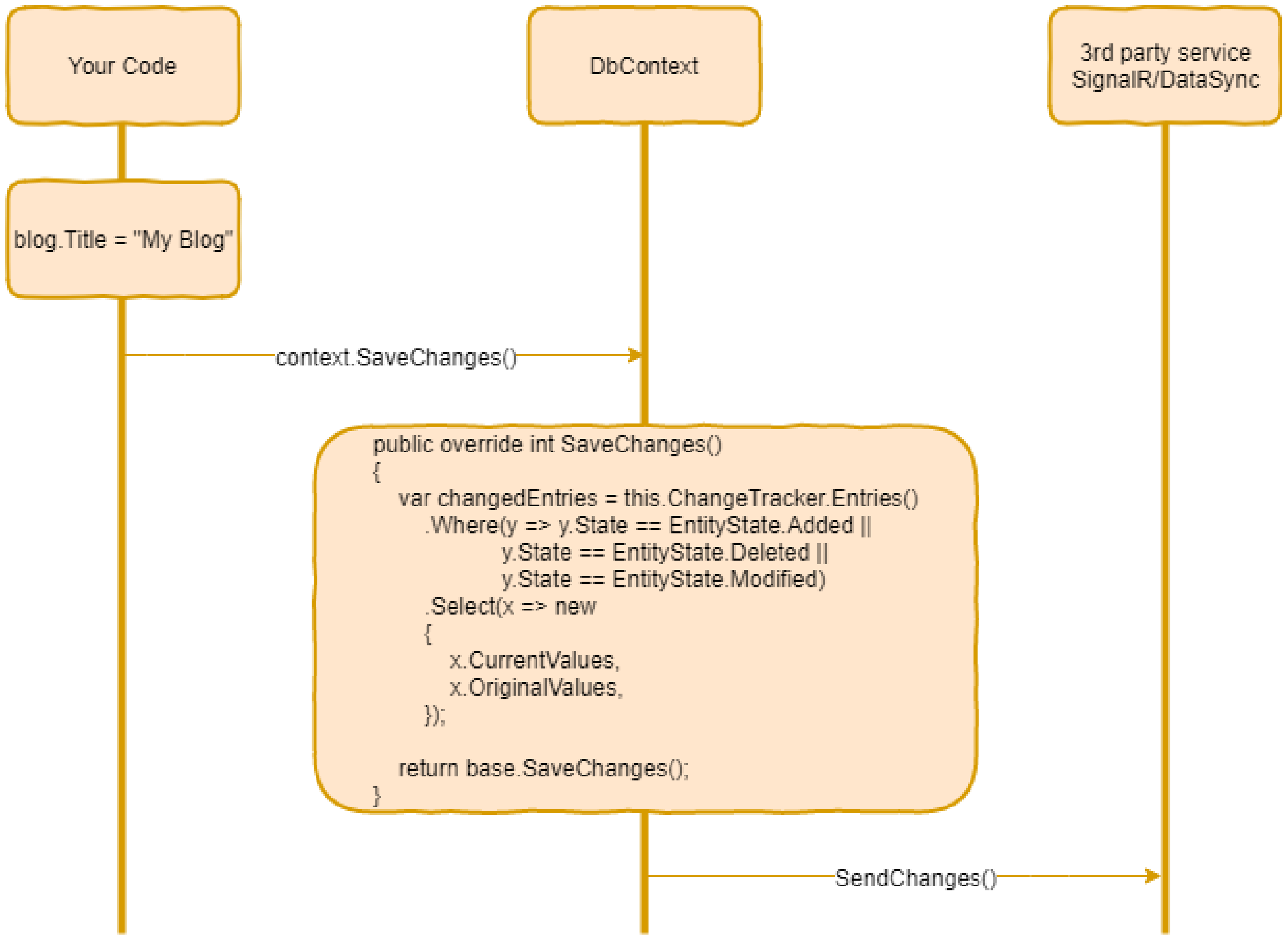
Dapper (EFSqlTranslator)

```
var query = context.Posts
    .Select(p => new
    {
        BlogId = p.Blog.Id,
        Title = p.Title
    })
    .GroupBy(x => x.BlogId)
    .Select(x => new { Id = x.Key, Cnt = x.Count() });

var queryResult = context.Query(query,
    new EFModelInfoProvider(context),
    new SqliteObjectFactory(),
    out var sql);
```

Override SaveChanges

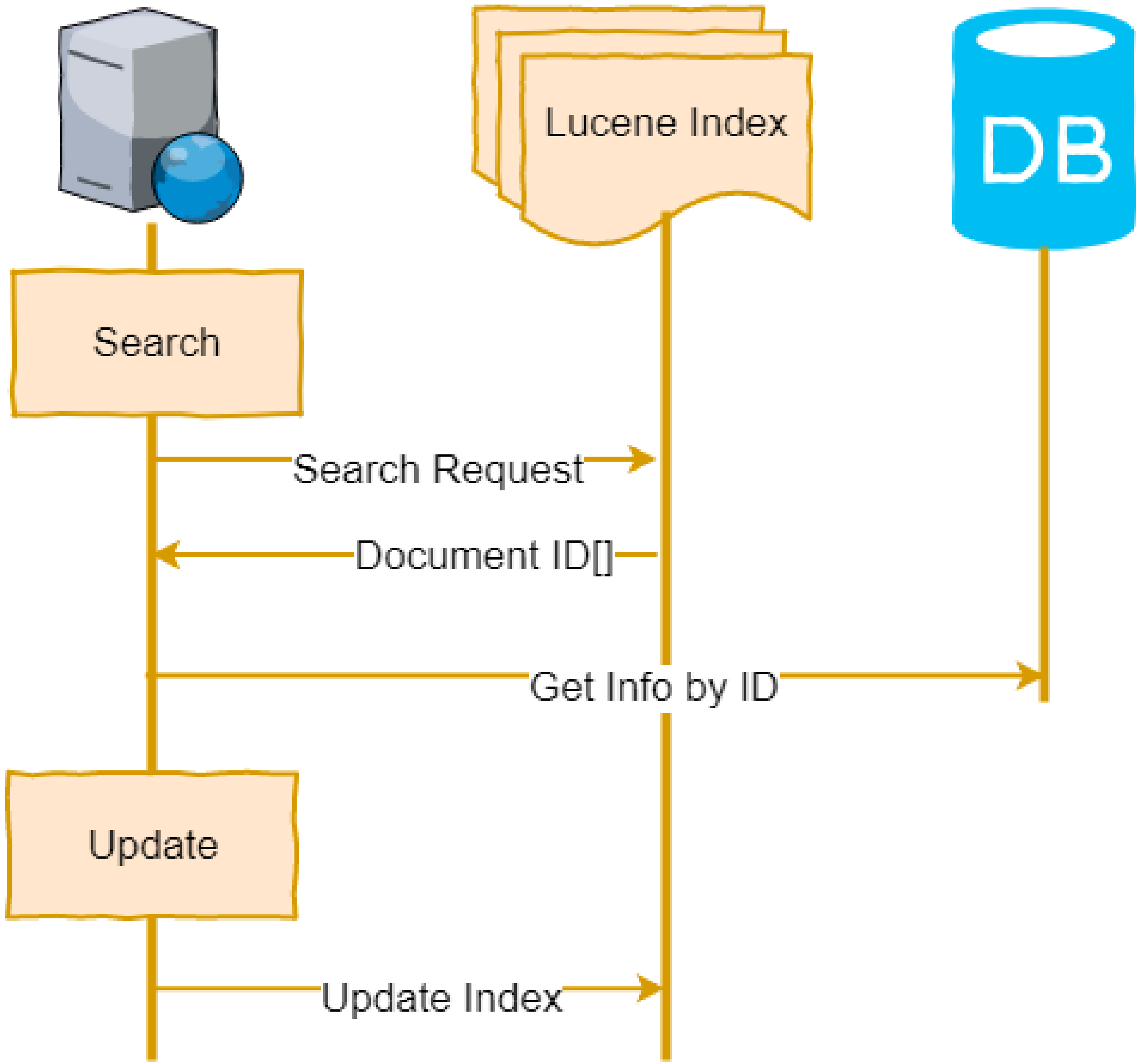
- Watch changes (e.g. cache invalidation)
- Log changes (e.g. audit)
- Inform about changes (send to 3rd party)

Override SaveChanges & SignalR



```
public override int SaveChanges()
{
    var changedEntries = this.ChangeTracker.Entries()
        .Where(y => y.State == EntityState.Added ||
                    y.State == EntityState.Deleted ||
                    y.State == EntityState.Modified)
        .Select(x => new
        {
            x.CurrentValues,
            x.OriginalValues,
        });

    return base.SaveChanges();
}
```

FullText Search



Summary

- SELECT N+1
- Group By & Aggregate
- ChangeTracker
- SaveChanges() override
- Testing & Migrations

mcc-tomsk.de

Спасибо. Вопросы?

Артур Дробинский

artur.drobinskiy@mcc-tomsk.de

http://arturdr.ru

mcc-
tomsk
.de

До встречи 31 января!
TomskDotNet #2!

Точка Кипения

31 января, 18:30