A Data Access Layer You're Proud of

Without Entity Framework

TRAILHEAD TECHNOLOGY PARTNERS

Jonathan "J." Tower

Hi, I’m J.

Jonathan "J." Tower

Partner & Principal Consultant

Trailhead Technology Partners

TRAILHEAD TECHNOLOGY PARTNERS

trailheadtechnology.com

- Microsoft MVP in ASP.NET
- Telerik/Progress Developer Expert
- Organizer of Beer City Code

- jtower@trailheadtechnology.com
- trailheadtechnology.com/blog
- jtowermi

github.com/jonathantower/dal-without-ef

What We’ll Cover

Pitfalls of EF

What I Need from an ORM

Micro-ORMs

CQS Pattern

If You Give $100, So Will I!

bit.ly/lou-water

"charity:water is a non-profit organization that provides clean and safe drinking water to people in developing nations. The organization was founded in 2006 and has helped fund 22,936 projects in 24 countries, benefiting over 4.6 million people." - Wikipedia

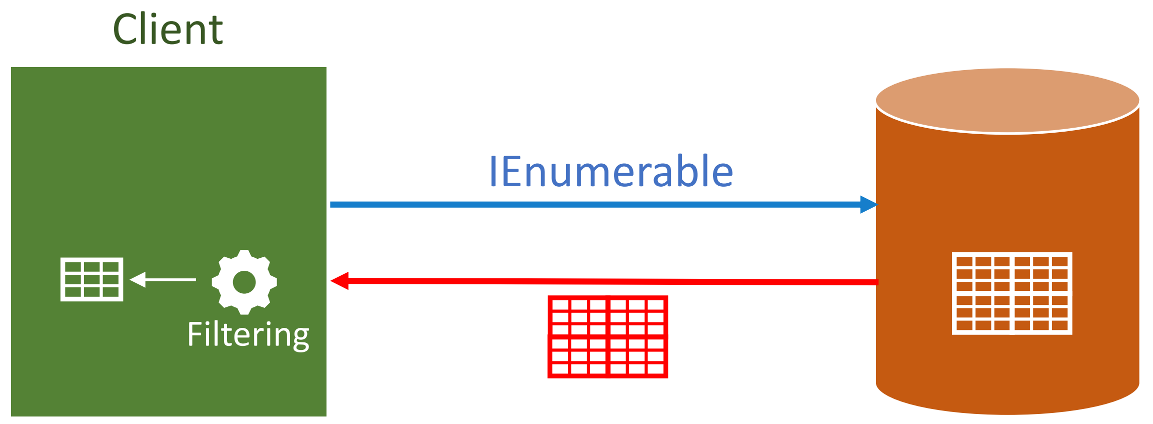
"4/4 Stars"
- CharityNavigator.org

charity: water

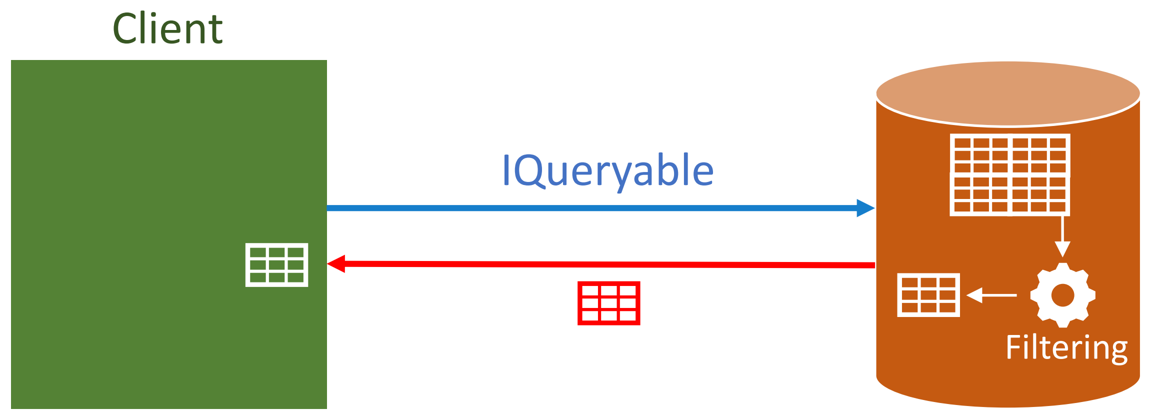
Pitfalls of Entity Framework

LINQ: IEnumerable vs IQueryable

IEnumerable



IQueryable

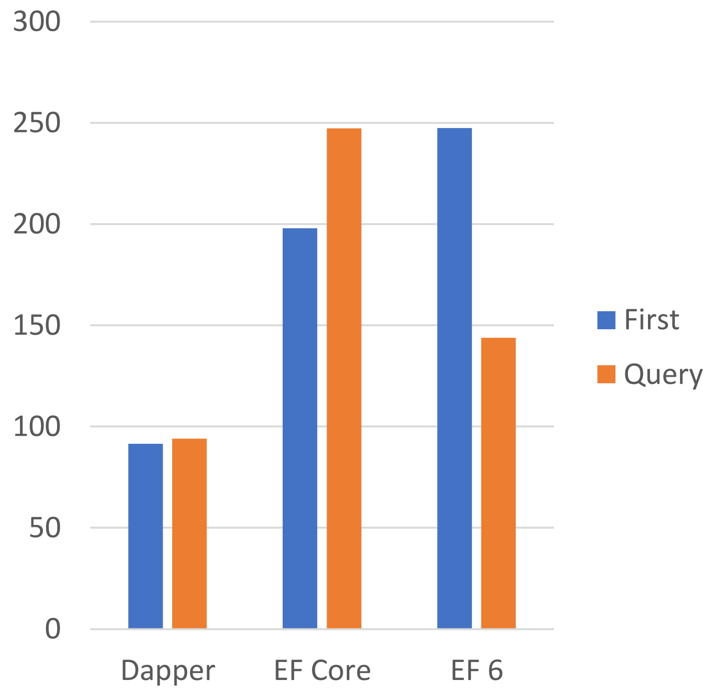


LINQ: IEnumerable vs IQueryable

```
// queryable
var myOrders = dbContext.Orders;

// enumerable
var myOrders = dbContext.Orders.ToList();

// which is happening here?
var order = myOrders
     .FirstOrDefault(o => o.OrderID == 170);
```

Query Syntax and Performance

ORM	Method	Mean	Allocated
First			
Dapper	QueryFirstOrDefault<T>	91.51 us	13.46 KB
EF Core	First	197.91 us	20.25 KB
EF 6	First	247.53 us	48.29 KB
Query			
Dapper	Query<T> (buffered)	94.05 us	13.79 KB
EF 6	SqlQuery	143.86 us	27.86 KB
EF Core	SqlQuery	247.25 us	20.56 KB



Source: https://github.com/StackExchange/Dapper#performance

Change Tracking

```
// changes tracked
var orders = dbContext.Orders
                .Where(p => p.EmployeeId >= 3)
                .ToList();


// changes NOT tracked
var orders2 = dbContext.Orders
                .Where(p => p.EmployeeId >= 3)
                .AsNoTracking()
                .ToList();
```

Lazy Loading is Dangerous

```
var orders = dbContext.Orders
    .Where(o => o.StatusId == 1)
    .ToList();

if (orders.Any(o => o.OrderDetails.Count() == 0)
{
    // delete order
}
```

What I Need from an ORM

Object-Relationship Mapping (ORM)

SQL

```
SELECT Id,
       FirstName,
       LastName,
       DOB
FROM People
WHERE Id = 1
```

Object

```
public class Person
{
    public int Id { get; set; }
    public string FirstName { get; set; }
    public string LastName { get; set; }
    public DateTime DOB { get; set; }
}
```

More Direct Control of SQL Syntax

```
from oa in context.OfficeAddresses
select new { oa.FirstName,  oa.LastName,
             Title = (from c in context.Contacts
                      where c.ContactID == oa.ContactID
                      select c.Title).FirstOrDefault(),
             oa.Street1, oa.City, oa.StateProvince
};
```

WHAT WILL THE GENERATED SQL LOOK LIKE?

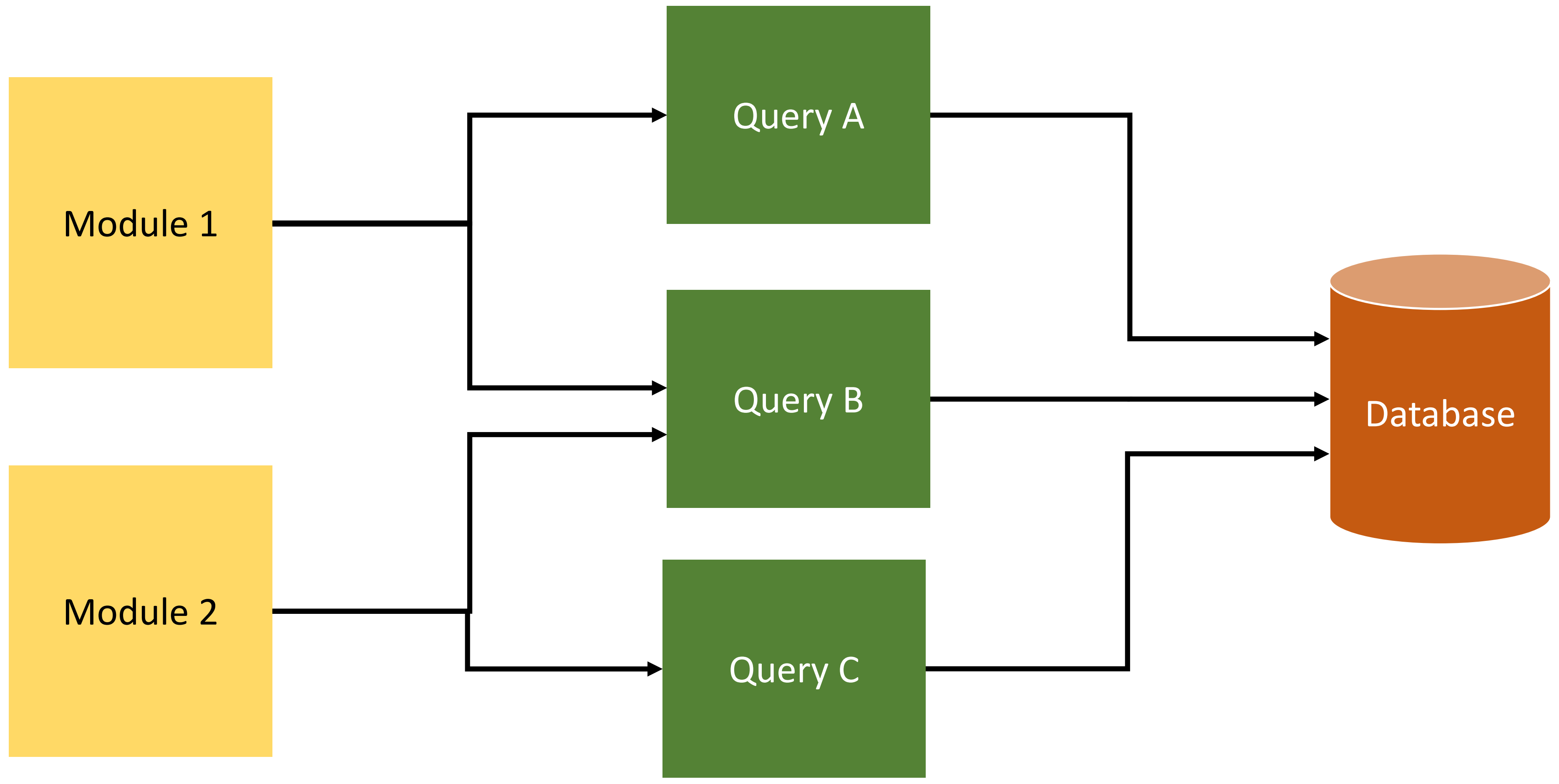
More Direct Control of SQL Syntax

```
var sql = "select ... from orders ...";

var orders = RunThisQuery(sql).IntoListOf<Order>();
```

Clear Delineation: In-Memory & Database

```
// filter on database
var sql = "select * from orders where typeid = 1";
var orders = RunThisQuery(sql).IntoListOf<Order>();

// filter in memory
var orders = orders.Where(o => o.IsComplete == true);
```

Reusability of Data Access Components



Consistency and Security

FILTER QUERIES AUTOMATICALLY

AUDIT FIELDS

Micro ORMs

What Makes a Micro ORM?

Map between objects & database queries

Small, lightweight

A Micro ORM is Unlikely to Have

LAZY LOADING

QUERY CACHING

IDENTITY TRACKING

CHANGE TRACKING

OO QUERY LANGUAGES (LINQ)

CONCEPT OF A UOW (TRANSACTION)

OrmLite (Via ServiceStack)

```
var dbFactory = new OrmLiteConnectionFactory(":memory:", SqliteDialect.Provider);
using (IDbConnection db = dbFactory.Open())
{
    db.DropAndCreateTable<Todo>();

    var todo = new Todo { Content = "Learn OrmLite", Order = 1 };
    db.Save(todo);

    var savedTodo = db.SingleById<Todo>(todo.Id);
    savedTodo.Content = "Updated";
    db.Save(savedTodo);

    db.DeleteById<Todo>(savedTodo.Id);
}
```

PetaPoco

```
var db = new PetaPoco.Database("constr");

var todo = new Todo { Id = 2, Content = "Learn PetaPoco", Order = 2 };
db.Save("Todos", "Id", todo);

db.Query<Todo>("SELECT * FROM Todos"));

db.Delete("Todos", "Id", todo.Id);
```

Dapper

```
using (var db = new SqlConnection(constr))
{
    // select many
    var todos = db.Query<ToDo>("SELECT * FROM ToDos").ToList();

    // execute parameterized SQL
    var param = new { Completed = true, Id = 1 };
    db.Execute(
        "UPDATE ToDos SET Completed = @Completed WHERE Id = @Id",
        param);
}
```

Entity Framework as a Micro-ORM

```
var blogs = dbContext.Blogs
    .FromSql("SELECT * FROM dbo.Blogs")
    .ToList();

var blogs = dbContext.Blogs
    .FromSql("EXECUTE dbo.GetMostPopularBlogs")
    .ToList();

var userId = new SqlParameter("userId", 1234);
var blogs = dbContext.Blogs
    .FromSql("EXECUTE dbo.GetMostPopularBlogsForUserId @userId", userId)
    .ToList();
```

CQS Pattern

Halmarks of CQS

"Command Query Separation"

Command methods should

- not return anything (void)
- only mutate the object state

Query methods should

- return results
- be immutable
- not change the object state

CQRS vs CQS

“Command Query Responsibility Segregation”

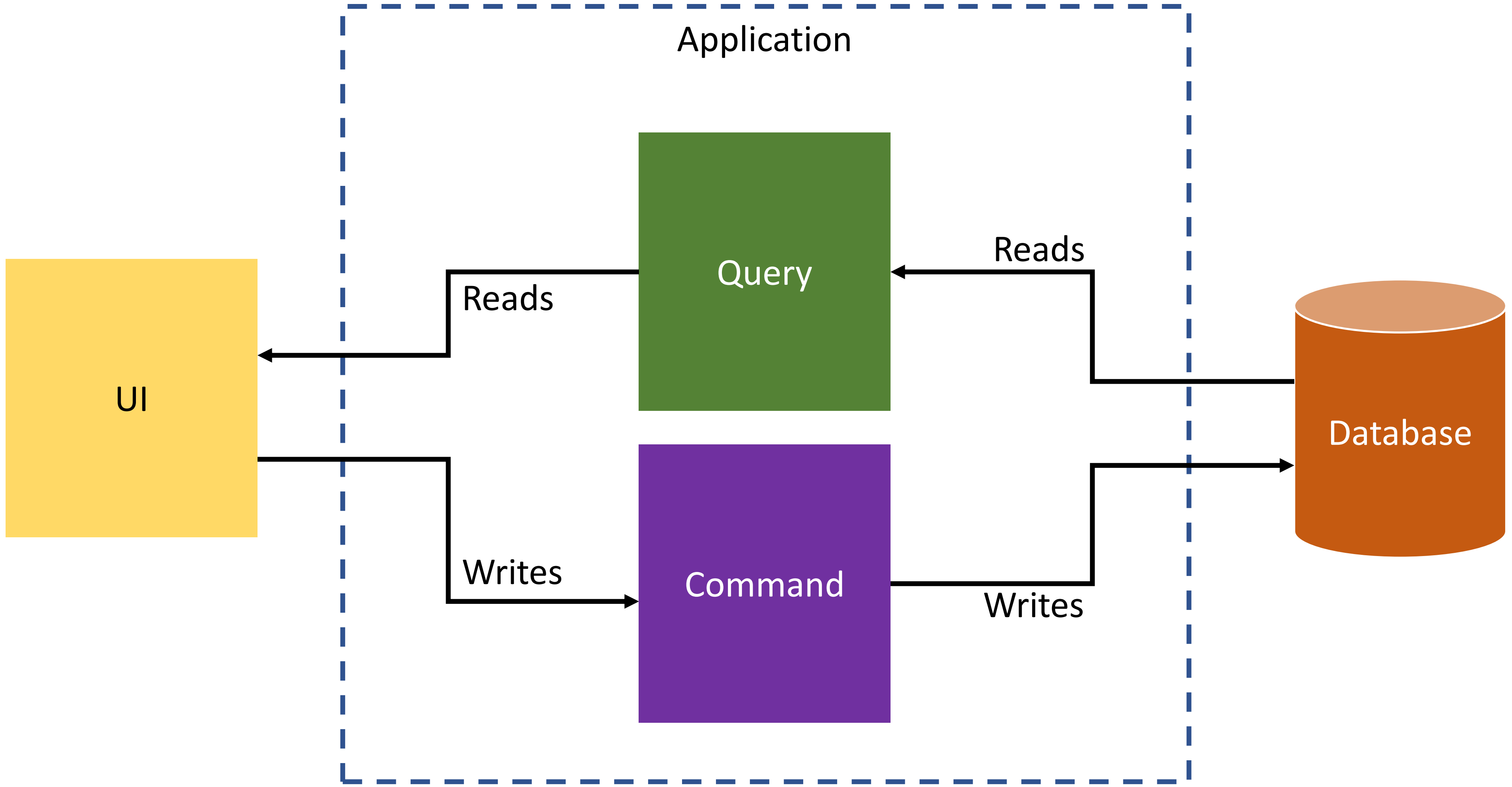
READ AND WRITE MODELS ARE COMPLETELY SEPARATE

STORE YOUR DATA IN THE SAME OR DIFFERENT DATABASES

INVOLVES MESSAGE QUEUES AND EVENTS

RELATED TO DDD (DOMAIN DRIVEN DESIGN)

CQS


Application
UI
Reads
Query
Reads
Database
Writes
Command
Writes

CQS: Queries

```
public class OrderQueries : IOrderQueries
{
    public async Task<IEnumerable<OrderDto>> GetOrdersAsync()
    {
        using (var cn = new SqlConnection(_constr))
        {
            cn.Open();
            return await cn.QueryAsync<OrderDto>(
              @"SELECT ...");
        }
    }
}
```

CQS: Commands

```
public class OrderCommands : IOrderCommands
{
    public async Task UpdateOrderAsync(OrderDto order)
    {
        using (var cn = new SqlConnection(_constr))
        {
            cn.Open();
            var rowsAffected = await cn.ExecuteAsync(
              @"UPDATE Orders SET ... WHERE OrderID = @OrderId",
              new { OrderId = order.OrderId, ... });

            if (rowsAffected > 0) throw new Exception("Update failed");
        }
    }
}
```

CQS: Usage from an API Controller

```
public class OrderController : Controller
{
    public OrderController(IOrderCommans commands, IOrderQueries queries) { ... }

    [HttpPut]
    public async IActionResult UpdateOrder(OrderDto order)
    {
        await _commands.UpdateOrderAsync(order)
        return Ok();
    }

    [HttpGet]
    public async IActionResult GetOrders()
    {
        return Ok(await _queries.GetOrdersAsync());
    }
}
```

Recap

Pitfalls of EF

What I Need
from an ORM

Micro-ORMs

CQS Pattern

If You Give $100, So Will I!

bit.ly/lou-water

"charity:water is a non-profit organization that provides clean and safe drinking water to people in developing nations. The organization was founded in 2006 and has helped fund 22,936 projects in 24 countries, benefiting over 4.6 million people." - Wikipedia

"4/4 Stars"
- CharityNavigator.org

charity: water

Thanks! Questions?

Jonathan "J." Tower
Partner & Principal Consultant
Trailhead Technology Partners

trailheadtechnology.com

Microsoft MVP in ASP.NET
Telerik/Progress Developer Expert
Organizer of Beer City Code

jtower@trailheadtechnology.com
trailheadtechnology.com/blog
jtowermi

github.com/jonathantower/dal-without-ef