Interacting with Your EF Core Data Model

Julie Lerman
MOST TRUSTED AUTHORITY ON ENTITY FRAMEWORK

@julielerman thedatafarm.com

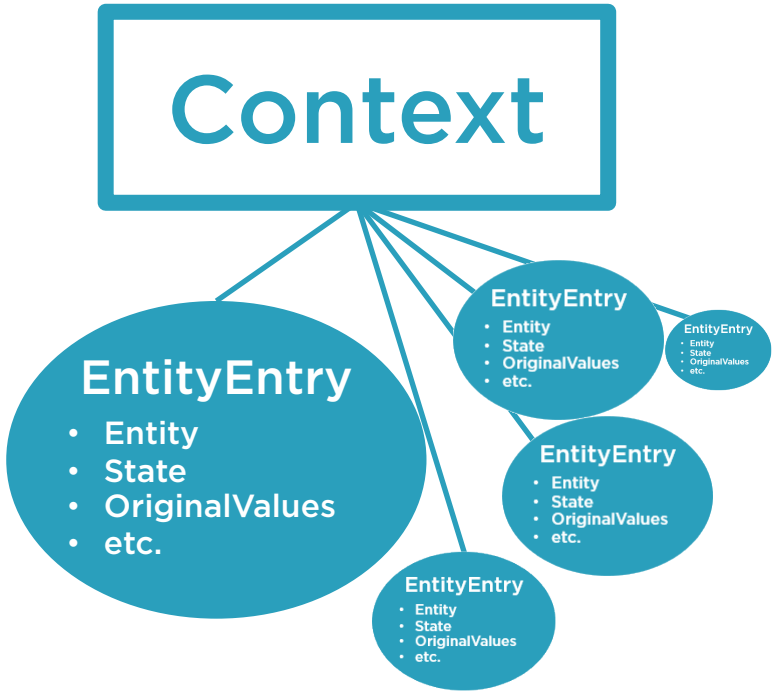
Module Overview

- Exploring SQL generated by EF Core
- Adding EF Core logging to the app
- Bulk operation support
- Query workflow
- Filters and aggregates in queries
- Updating and deleting objects
- Persisting data in disconnected apps
- De-activate tracking in disconnected apps

Looking at SQL Built by EF Core

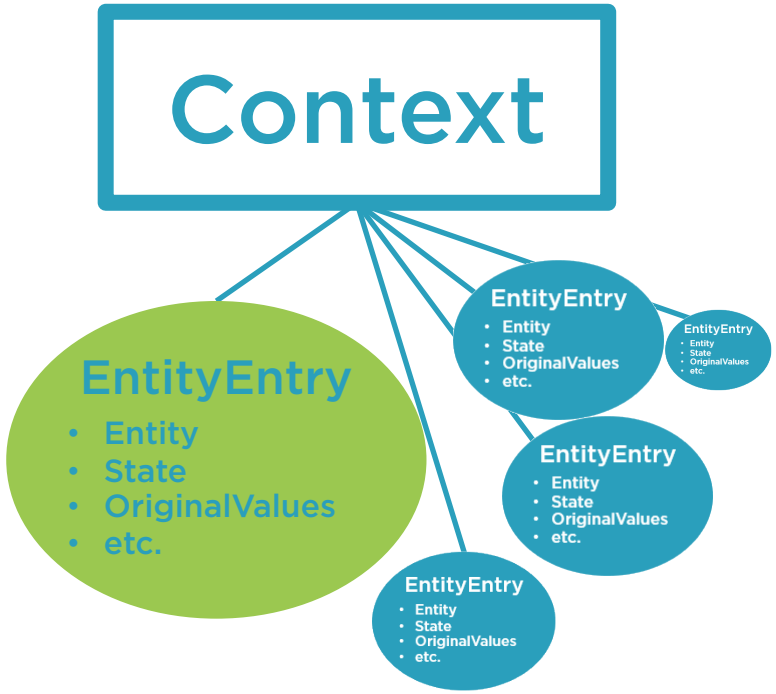
Under the Covers: Tracking Entities

```
context.Samurais.Add(samuraiObject)
```



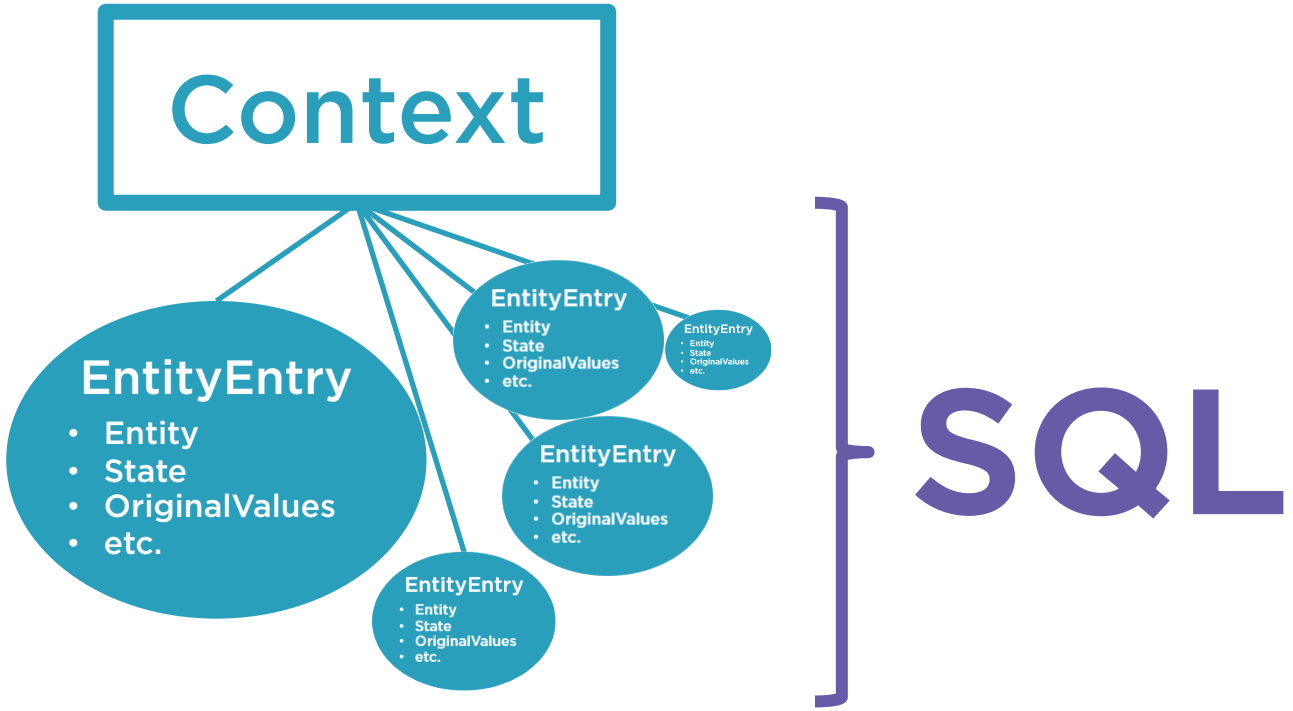
Under the Covers: Tracking Entities

```
context.Samurais.Add(samuraiObject)
```



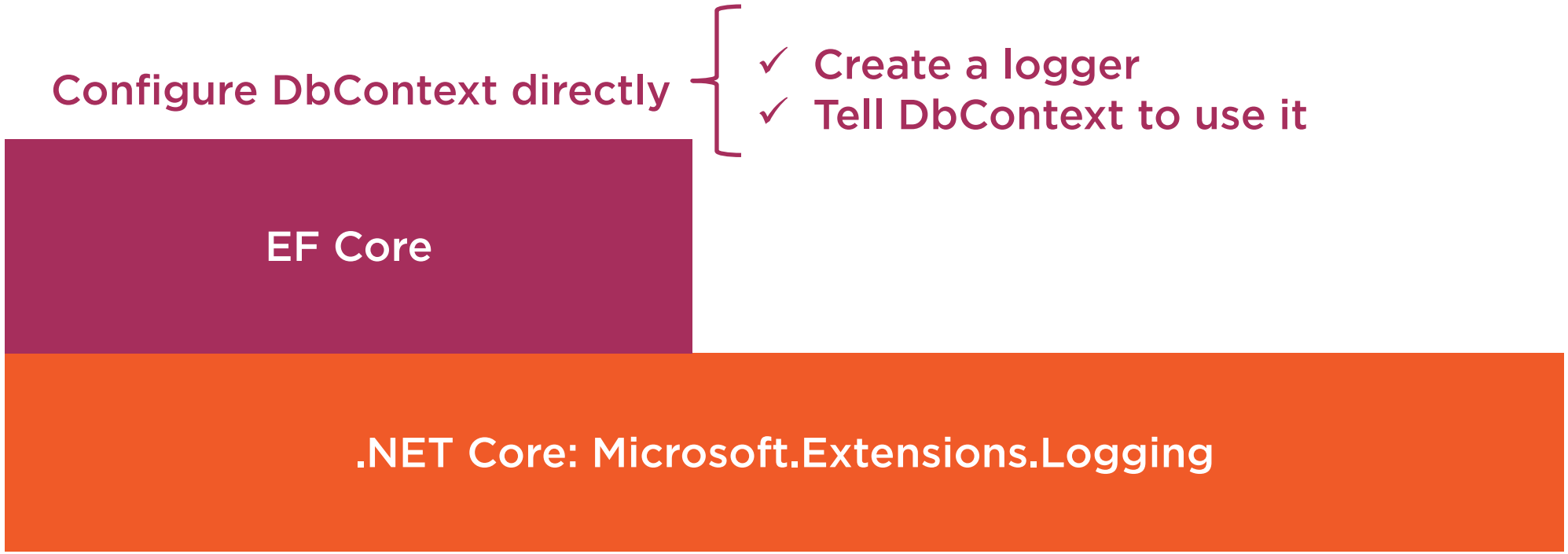
Under the Covers: Tracking Entities

```
context.Samurais.Add(samuraiObject)
```

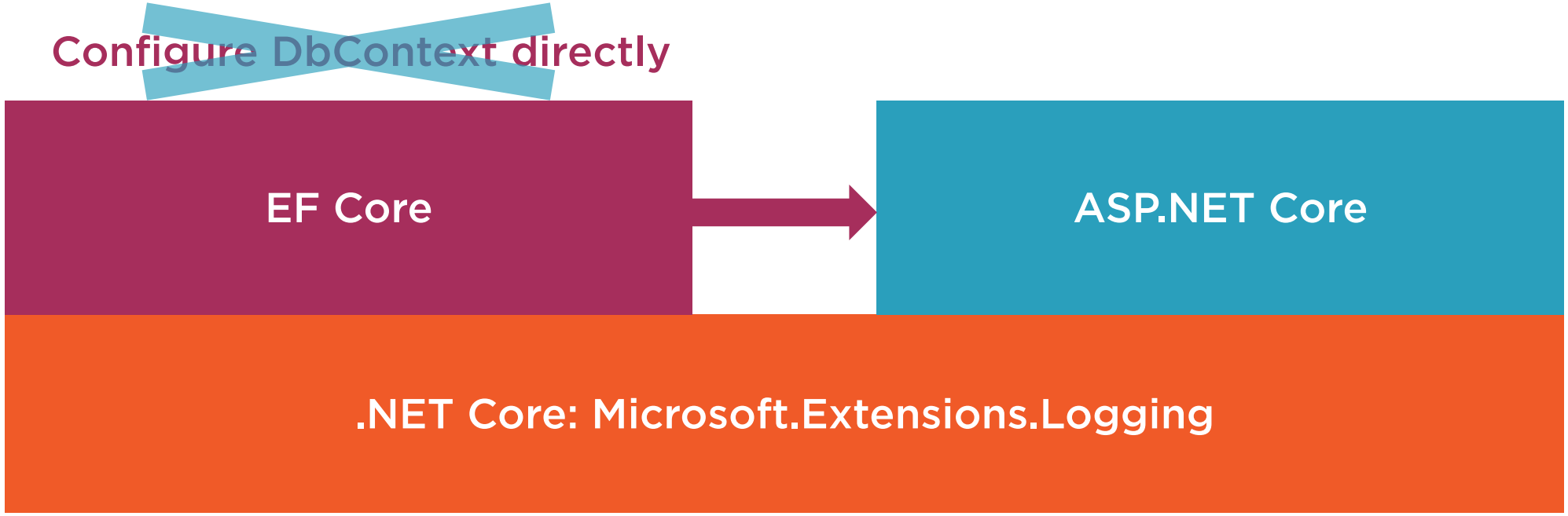


Adding Logging to EF Core's Workload

.NET Core Logging



.NET Core Logging



Benefiting from Bulk Operations Support

At least 4 operations needed for SQL Server provider to batch commands

Tracking Methods on DbSet and DbContext

```
context.Samurais.Add(…)
context.Samurais.AddRange(…)
```

Track via DbSet

DbSet indicates type

```
context.Add(…)
context.AddRange(…)
```

Track via DbContext

Context will discover type(s)

Batch Operation Batch Size

- Default size & more is set by database provider
- Additional commands will be sent in extra batches
- Override batch size in DbContext OnConfiguring

```
protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
{
    optionsBuilder
        .UseLoggerFactory(MyConsoleLoggerFactory)
        .EnableSensitiveDataLogging(true)
        .UseSqlServer(connectionString, options=>options.MaxBatchSize(150));
}
```

Understanding the Query Workflow

Query Workflow

Express & execute query

```
context.Samurais.ToList()
```

EF Core reads model, works with provider to work out SQL

Sends SQL to database

```
SELECT * from Samurais
```

Receives tabular results

3	Ms.	Donnie	F.	Carreras
4	Ms.	Janet	M.	Gates
5	Mr.	Lucy	*NULL*	Harrington
6	Mr.	Joop	X.	Carroll
7	Mr.	Dominic	P.	Gash
10	Ms.	Kathleen	M.	Garza
11	Ms.	Kathleen	*NULL*	Harding
12	Mr.	Johnny	A.	Caprio
16	Mr.	Christopher	R.	Beck
18	Mr.	David	J.	Liu
19	Mr.	John	A.	Beaver

Materializes results as objects

Adds tracking details to DbContext instance

Two Ways to Express LINQ Queries

LINQ Methods

```
context.Samurais.ToList();
```

```
context.Samurais
.Where(s=>s.Name==“Julie”)
.ToList()
```

LINQ Query Syntax

```
(from s in context.Samurais
 select s).ToList()
```

```
(from s in context.Samurais
 where s.Name==“Julie”
 select s).ToList()
```

Database Connection Remains Open During Enumeration

```
foreach (var s in context.Samurais){
   Console.WriteLine(s.Name);
}
```

- Minimal effort on enumeration, ok

```
foreach (var s in context.Samurais){
   RunSomeValidator(s.Name);
   CallSomeService(s.Id);
   GetSomeMoreDataBasedOn(s.Id);
}
```

- Lots of work for each result. Connection stays open until last result is fetched.

```
var samurais=context.Samurais.ToList()
foreach (var s in samurais){
   RunSomeValidator(s.Name);
   CallSomeService(s.Id);
   GetSomeMoreDataBasedOn(s.Id);
}
```

- Smarter to get results first

Filtering in Queries

The **GREAT LINQ OVERHAUL** for EF Core 3 impacts query behavior

Enabling Sensitive Data to Show in Logs

Default: Parameters are hidden

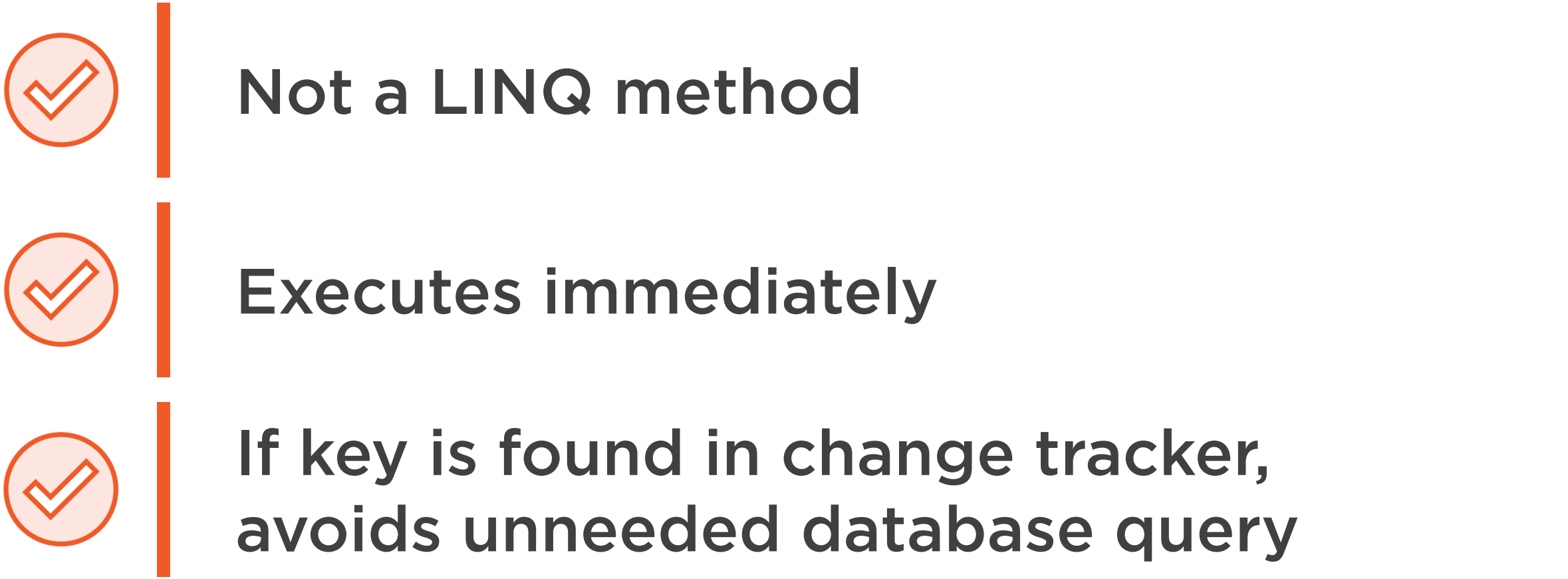
```
[__name_0='?' (Size = 4000)]
```

Configure with OptionsBuilder

```
protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
{
  optionsBuilder
  .UseLoggerFactory(ConsoleLoggerFactory)
  .EnableSensitiveDataLogging()
  .UseSqlServer(connectionString);
}
```

```
[__name_0='Sampson' (Size = 4000)]
```

DbSet.Find(key)



Filtering Partial Text LINQ

Like

EF.Functions.Like(property, %abc%)

```
_context.Samurais.Where(s=>
  EF.Functions.Like(s.Name,"%abc%")
)
```

SQL LIKE(%abc%)

Contains

property.Contains(abc)

```
_context.Samurais.Where(s=>
  s.Name.Contains("abc")
)
```

SQL LIKE(%abc%)

Aggregating in Queries

EF Core Parameter Creation

Search value is directly in query

```
...Where(s=>s.Name==“Sampson”)
```

No parameter is created in SQL

```
SELECT * FROM T
WHERE T.Name=‘Sampson’
```

Search value is in a variable

```
var name=“Sampson”

...Where(s=>s.Name==name)
```

Parameter is created in SQL

```
@parameter=‘Sampson’

SELECT * FROM T
WHERE T.Name=@parameter
```

3.0: LINQ to Entities Execution Methods

ToList()	ToListAsync()
First()	FirstAsync()
FirstOrDefault()	FirstOrDefaultAsync()
Single()	SingleAsync()
SingleOrDefault()	SingleOrDefaultAsync()
Last()*	LastAsync()*
LastOrDefault()*	LastOrDefaultAsync()*
Count()	CountAsync()
LongCount()	LongCountAsync()
Min(), Max()	MinAsync(), MaxAsync()
Average(), Sum()	AverageAsync(), SumAsync()
	AsAsyncEnumerable**

Not a LINQ method, but a DbSet method that will execute:

Find(keyValue)	FindAsync(keyValue)

*Last methods require query to have an OrderBy() method otherwise will return full set then pick last in memory
**New to EF Core 3 with C#8 support

Updating Simple Objects

Skip & Take for Paging

1. Aardvark
2. Abyssinian
3. Adelie Penguin
4. Affenpinscher
5. Afghan Hound
6. African Bush Elephant
7. African Civet
8. African Clawed Frog
9. African Forest Elephant
10. African Palm Civet

Get first 10 animals

Skip(0).Take(10)

11. African Penguin
12. African Tree Toad
13. African Wild Dog
14. Ainu Dog
15. Airedale Terrier
16. Akbash
17. Akita
18. Alaskan Malamute
19. Albatross
20. Aldabra Giant Tortoise

Get next 10 animals

Skip(10).Take(10)

Deleting Simple Objects

Deleting May Seem a Little Weird

```
_context.Samurais.Add(samurai)
_context.Samurais.AddRange(samuraiList)

_context.Add(samurai)
_context.AddRange(samurai, battle)
```

```
_context.Samurais.Update(samurai)
_context.Samurais.UpdateRange(samuraiList)

_context.Update(samurai)
_context.UpdateRange(samurai, battle)
```

```
_context.Samurais.Remove(samurai)
_context.Samurais.RemoveRange(samuraiList)

_context.Remove(samurai)
_context.RemoveRange(samurai, battle)
```

- DbSet Add, AddRange
- DbContext Add, AddRange

- DbSet Update, UpdateRange
- DbContext Update, UpdateRange
- DbSet Remove, RemoveRange
- DbContext Remove, RemoveRange

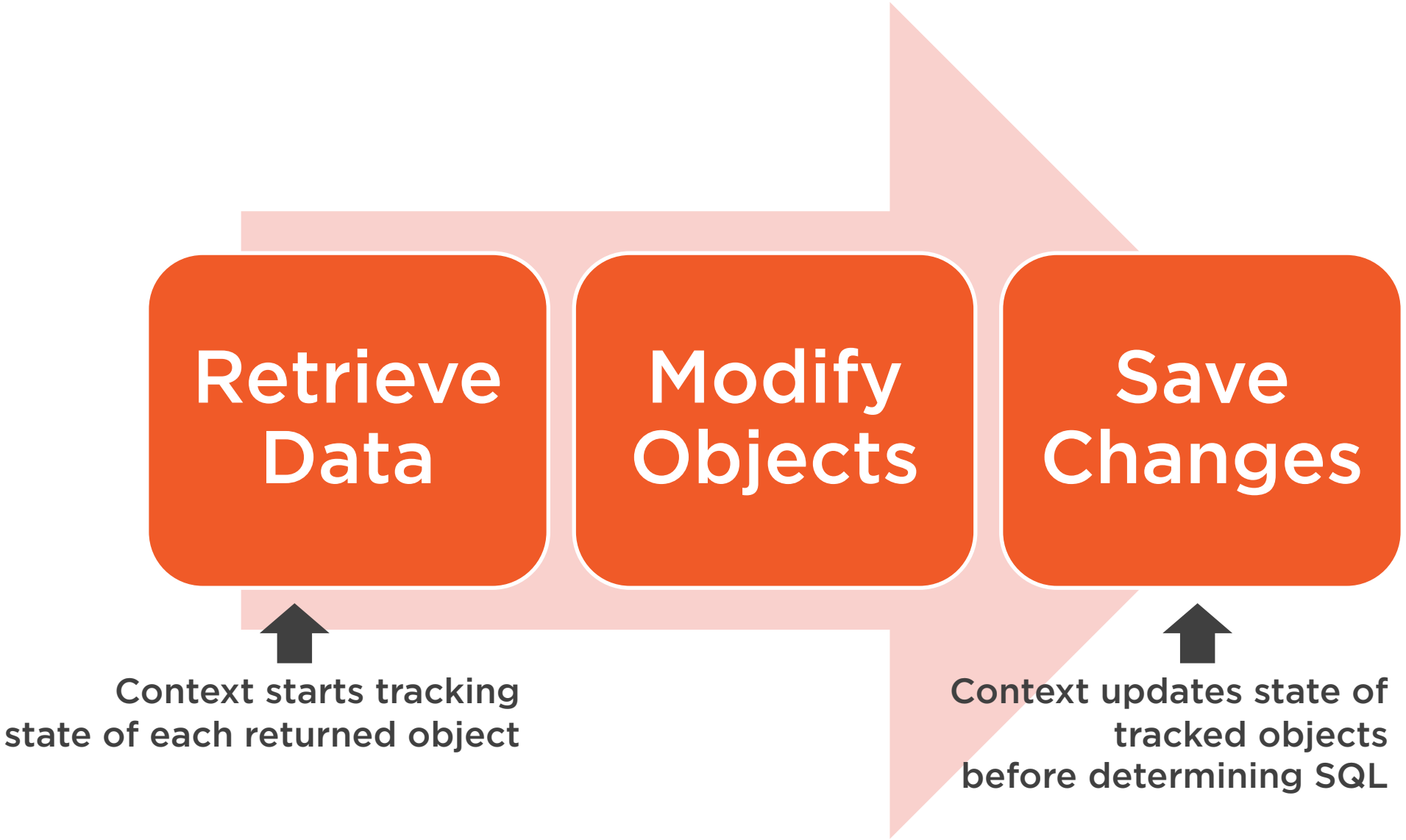
Workarounds for Required Object to Delete

Fake object with key property filled:
watch out for possible side effects

Stored procedure called using
EF Core raw SQL feature
Further on in this course

Persisting Data in Disconnected Scenarios

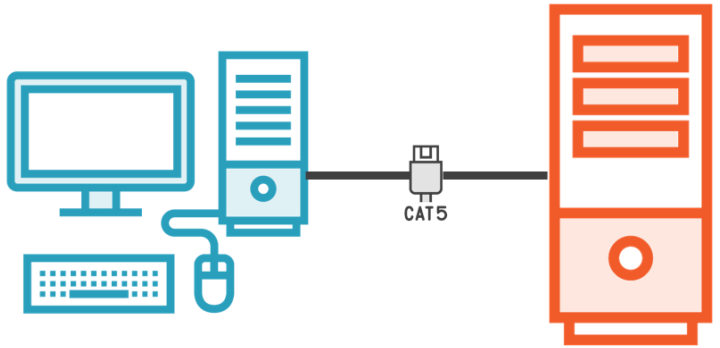
Working in a Single DbContext Instance



Connected Data Access

Client Storing Data Locally

Network Connected Clients



Disconnected Clients

In disconnected scenarios, it’s up to you to inform the context about object state

Enhancing Performance in Disconnected Apps with No-Tracking Settings

Review

- Log EF Core SQL commands
- Inserts, updates and deletes
- Bulk operations
- Comprehend EF Core querying
- Filtering and aggregating in queries
- Persisting in disconnected apps e.g., web site
- Improve performance when tracking is not needed

Resources

Entity Framework Core on GitHub github.com/aspnet/entityframework

EF Core Documentation docs.microsoft.com/ef

The Case of Entity Framework Core's Odd SQL, Richie Rump
brentozar.com/archive/2017/05/case-entity-framework-cores-odd-sql/

Entity Framework Core 3.0: A Foundation for the Future
codemag.com/Article/1911062/Entity-Framework-Core-3.0-A-Foundation-for-the-Future

Entity Framework in the Enterprise, Pluralsight course bit.ly/PS_EFEnt

Logging SQL and Change-Tracking Events in EF Core, MSDN Mag Oct 2019
msdn.microsoft.com/magazine/mt830355

Interacting with Your EF Core Data Model

Julie Lerman
MOST TRUSTED AUTHORITY ON ENTITY FRAMEWORK

@julielerman thedatafarm.com