

Interacting with Your EF Core Model



Julie Lerman

MOST TRUSTED AUTHORITY ON ENTITY FRAMEWORK

@julielerman thedatafarm.com



Module Overview



Inserting, updating and deleting simple objects

EF Core's batch command support

Querying against the data model

Filtering queries

EF Core behavior in disconnected apps

Logging database commands and more



Getting EF Core to Output SQL Logs



Inserting Simple Objects



```
var samurai = new Samurai { Name = "Julie" };  
using (var context = new SamuraiContext())  
{  
    context.Samurais.Add(samurai);  
}
```



The context is now *tracking* the samurai object

SaveChanges Internal Workflow

```
context  
.Samurais  
.Add(samurai);
```

```
context  
.SaveChanges();
```

Examine each tracked object

Read state of object

Work out SQL commands

Execute each SQL command in the database

Capture any returned data



Batching Commands When Saving



Batch Operation Batch Size

Default size 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));
    }
]
```



Querying Simple Objects



Two Ways to Express LINQ Queries

LINQ Methods

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

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

LINQ Query Syntax

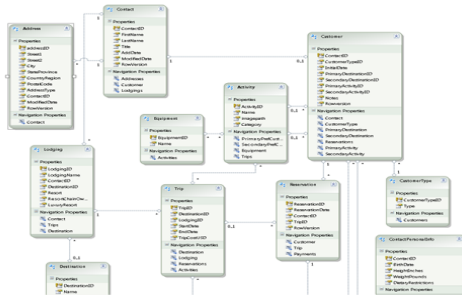
```
(FROM s IN context.Samurais)  
.ToList()
```

```
(FROM s  
  IN context.Samurais  
  WHERE s.Name=="Julie")  
.ToList()
```



Basic Workflow

Define model



Express & execute query

(from p in people select p)
.ToList()

EF determines
& executes SQL

SELECT * from people

EF transforms results
into your types

User modifies data



Code triggers save

DbContext.SaveChanges

EF determines & executes SQL

**UPDATE people
SET Firstname='Julie'
WHERE id=3**

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



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);  
}
```

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

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

◀ Minimal effort on enumeration, ok

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

◀ Smarter to get results first



Filtering Partial Text LINQ

Like (new to EF Core 2!)

`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%)`



Query performance has
improved greatly
in EF Core,
although on the surface,
not much has changed



A DbContext instance used for multiple operations will keep track of all of the entities used in each of those operations.



Filtering Data in Queries



LINQ to Entities Execution Methods

ToList()
First()
FirstOrDefault()
Single()
SingleOrDefault()
Last()*
LastOrDefault()*
Count()
LongCount()
Min()
Max()
Average()

ToListAsync()
FirstAsync()
FirstOrDefaultAsync()
SingleAsync()
SingleOrDefaultAsync()
LastAsync()*
LastOrDefaultAsync()*
CountAsync()
LongCountAsync()
MinAsync()
MaxAsync()
AverageAsync()

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



Updating Simple Objects



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
```



Disconnected Updates

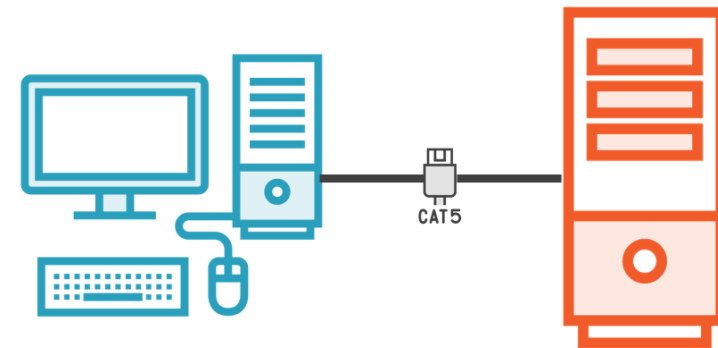


Connected Data Access

Client Storing Data Locally



Network Connected Clients



Disconnected Clients



Disconnected Data Access Methods

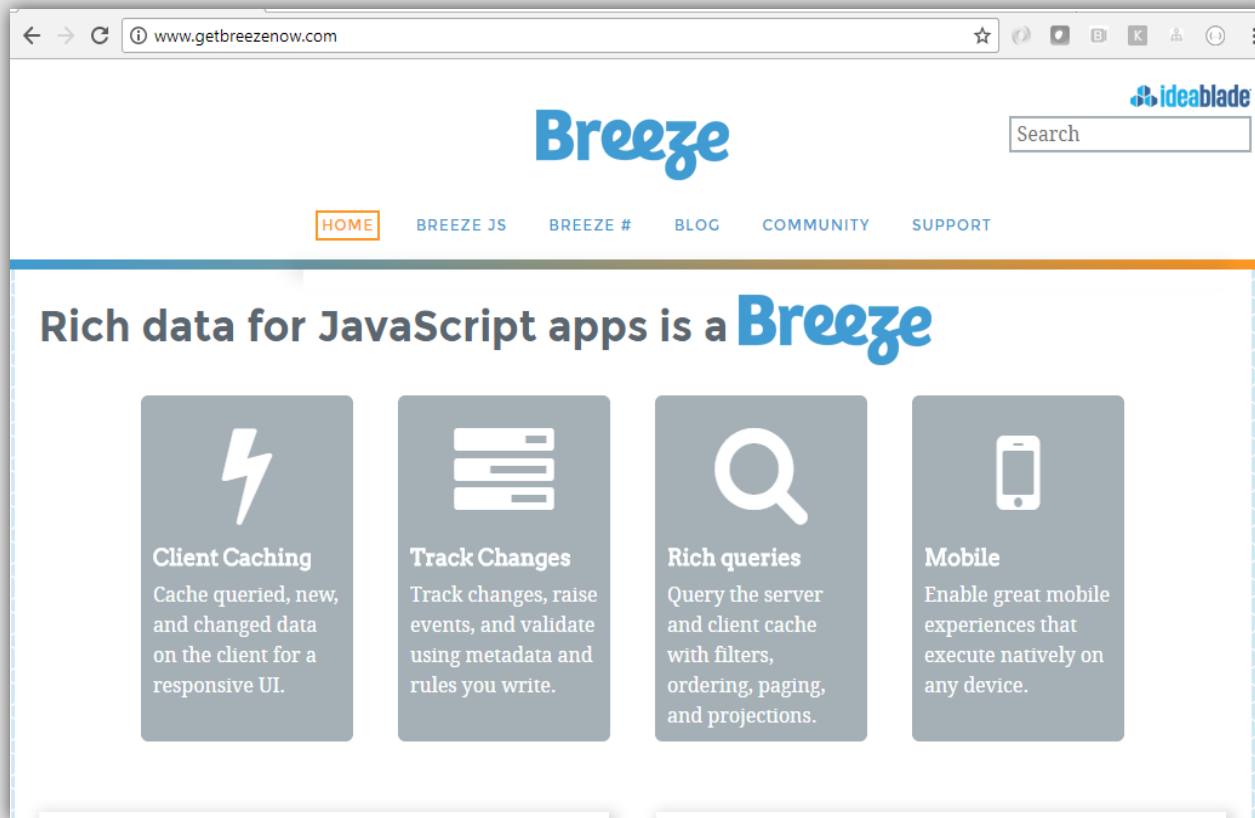
```
public Samurai GetSamuraiById(int id)
{
    using (var context=new SamuraiContext())
    {    //query & return a samurai
    }
}

public void UpdateSamurai(Samurai samurai)
{
    using (var context=new SamuraiContext())
    {    //update samurai in database
    }
}

public void InsertSamurai(Samurai samurai)
{
    using (var context=new SamuraiContext())
    {    //insert samurai into database
    }
}
```



Tools for Tracking Between Server and Client



The screenshot shows the Breeze website at www.getbreezenow.com. The page features the Breeze logo, a search bar, and a navigation menu with links to HOME, BREEZE JS, BREEZE #, BLOG, COMMUNITY, and SUPPORT. The main content area highlights the tagline "Rich data for JavaScript apps is a Breeze" and lists four key features: Client Caching, Track Changes, Rich queries, and Mobile.

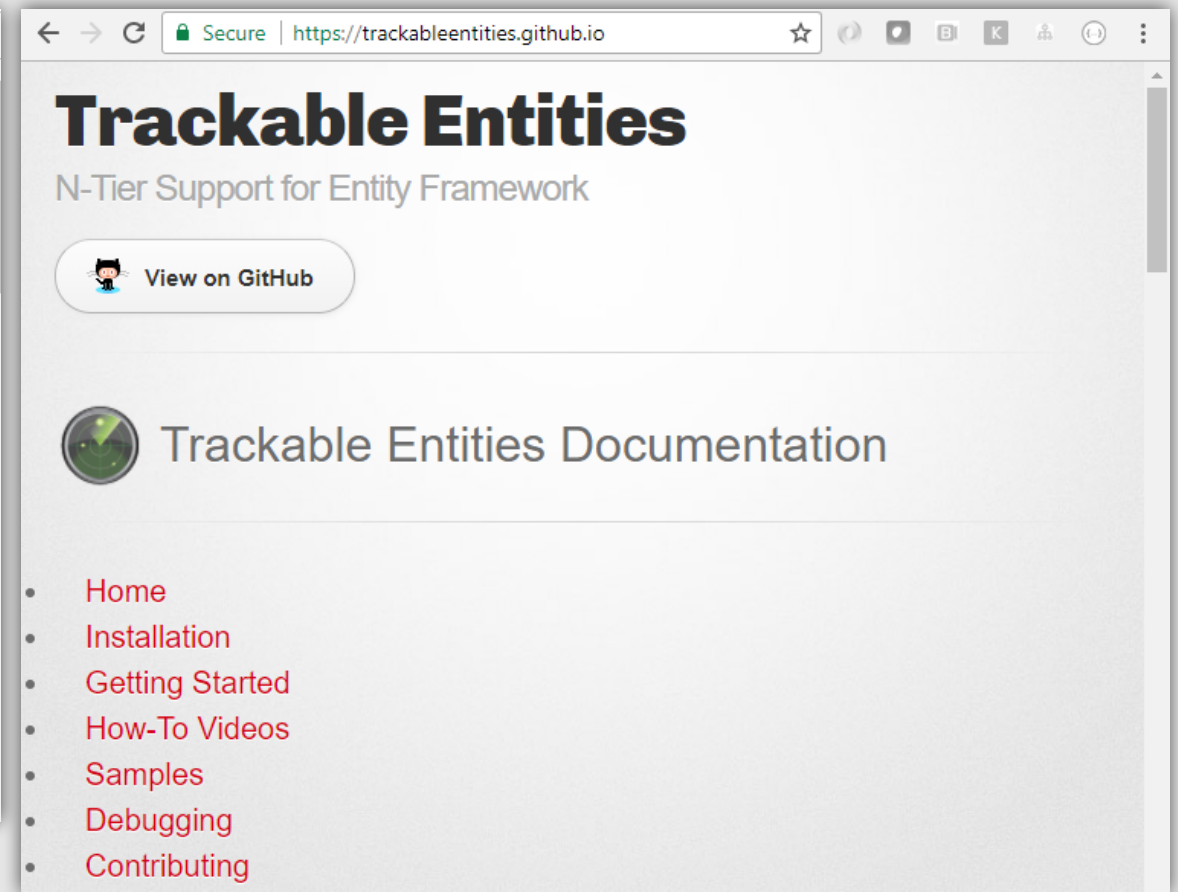
Breeze

Search

HOME BREEZE JS BREEZE # BLOG COMMUNITY SUPPORT

Rich data for JavaScript apps is a **Breeze**

- Client Caching**
Cache queried, new, and changed data on the client for a responsive UI.
- Track Changes**
Track changes, raise events, and validate using metadata and rules you write.
- Rich queries**
Query the server and client cache with filters, ordering, paging, and projections.
- Mobile**
Enable great mobile experiences that execute natively on any device.



The screenshot shows the Trackable Entities website at <https://trackableentities.github.io>. The page features the title "Trackable Entities" with the subtitle "N-Tier Support for Entity Framework". It includes a "View on GitHub" button and a link to "Trackable Entities Documentation". A sidebar menu lists various resources: Home, Installation, Getting Started, How-To Videos, Samples, Debugging, and Contributing.

Secure | <https://trackableentities.github.io>

Trackable Entities

N-Tier Support for Entity Framework

[View on GitHub](#)

Trackable Entities Documentation

- [Home](#)
- [Installation](#)
- [Getting Started](#)
- [How-To Videos](#)
- [Samples](#)
- [Debugging](#)
- [Contributing](#)



```
_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



Deleting Objects with EF Core



DbContext can only delete
objects it is aware of,
i.e., already tracking



Review

Add, update & delete through DbSet or DbContext

Logs can capture SQL and other info

Commands are batched by default

Delete requires a full object in memory

Use LINQ execute methods to trigger a query

EF Core can process queries with server side and local logic

EF Core on server can't magically track what happens on client

Resources

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

EF Core Roadmap bit.ly/efcoreroadmap

EF Core Documentation docs.efproject.net

Entity Framework Core: Getting Started (for EF Core 1.1) bit.ly/PS_EFCoreStart

Bulk Operations Commands Explanation by Richie Rump:
<https://www.brentozar.com/archive/2017/05/case-entity-framework-cores-odd-sql/>

BreezeJS and Breeze.NET: www.getbreezenow.com

Trackable Entities trackableentities.github.io



Interacting with Your EF Core Model



Julie Lerman

MOST TRUSTED AUTHORITY ON ENTITY FRAMEWORK

@julielerman thedatafarm.com

