

# Using EF Core 6 to Query a Database

---



**Julie Lerman**

Most Trusted Authority on Entity Framework Core

@julielerman thedatafarm.com



# Module Overview



**Understand EF Core's query workflow**

**Basics of querying using EF Core and LINQ**

**Filtering, sorting and aggregating in queries**

**Explore the SQL queries that EF Core is building for you**

**Improve query performance by deactivating tracking when not needed**

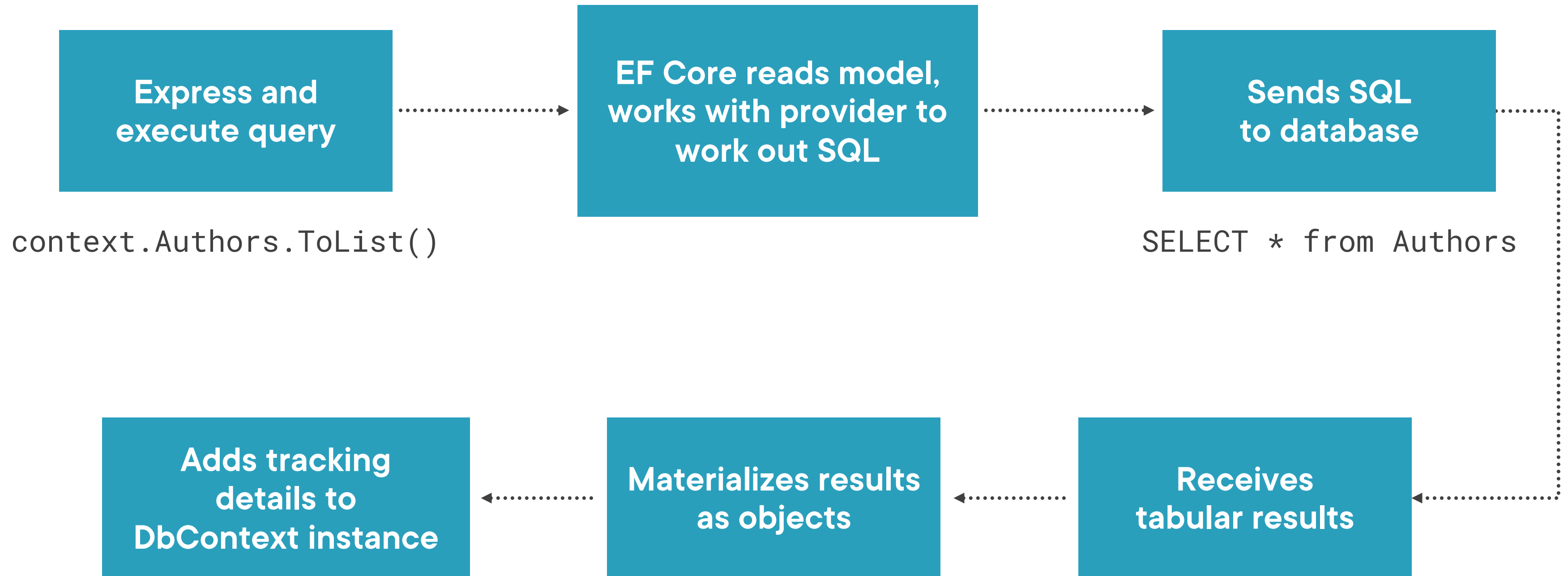


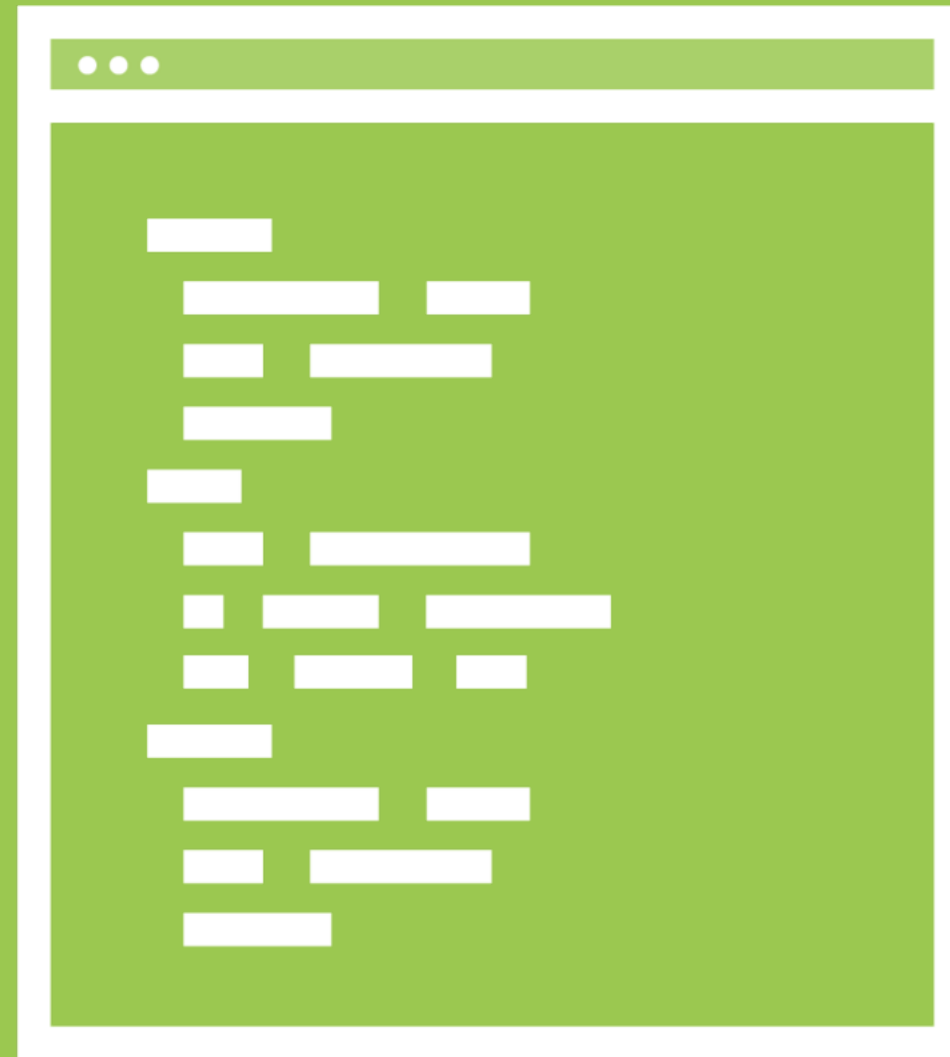
# Querying Basics

---



# Query Workflow





# The Simplest Query

```
_context.Authors.ToList()
```



A LINQ execution method,  
e.g., ToList(),  
triggers the query to execute  
on the database.



# Two Ways to Express LINQ Queries

## LINQ Methods

```
context.Authors.ToList();
```

```
context.Authors  
  .Where(a => a.FirstName == "Julie")  
  .ToList()
```

## LINQ Operators

```
(from a in context.Authors  
 select a)  
.ToList()
```

```
(from a in context.Authors  
 where a.FirstName == "Julie"  
 select a)  
.ToList()
```





# More about LINQ Operators

## Querying Data in EF Core 6

Torben Boeck Jensen





```
var authors = context.Authors.ToList();
```

```
var query = context.Authors;  
var authors = query.ToList();
```

Queries are Composable

**You don't need to convey them in a single expression**

# Triggering Queries via Enumeration

```
var query=_context.Authors;  
foreach (var a in query)  
{  
    Console.WriteLine(a.LastName);  
}
```

```
var query=_context.Authors;  
foreach (var a in  
        context.Authors)  
{  
    Console.WriteLine(a.LastName);  
}
```



Database connection remains  
open during query  
enumeration



# Enumerate vs. Execute

```
foreach (var a in context.Authors){  
    Console.WriteLine(a.FirstName);  
}
```

```
foreach (var a in context.Authors){  
    RunSomeValidator(a.FirstName);  
    CallSomeService(a.Id);  
    GetSomeMoreDataBasedOn(a.Id);  
}
```

```
var samurais=context.Authors.ToList()  
foreach (var a in authors){  
    RunSomeValidator(a.FirstName);  
    CallSomeService(a.Id);  
    GetSomeMoreDataBasedOn(a.Id);  
}
```

◀ **Good enumeration: Minimal effort on enumeration**

◀ **Bad enumeration: Lots of work for each result. Connection stays open until last result is fetched.**

◀ **Execution: Smarter to get results first**

# Filtering Queries Securely by Default

---



Parameterized queries protect  
your database from SQL  
Injection attacks





# More about SQL Injection Attack Security

## Ethical Hacking: SQL Injection

Troy Hunt



# EF Core Parameter Creation

## Search value is directly in query

```
.Where(a=>a.FirstName=="Josie")
```



**No parameter is created in SQL**

```
SELECT * FROM Authors  
WHERE Authors.FirstName='Josie'
```

## Search value in a variable

```
var name="Josie"  
.Where(a=>a.FirstName==name)
```



**Parameter is created in SQL**

```
@P1='Josie'  
SELECT * FROM Authors  
WHERE Authors.FirstName=@P1
```





# Benefiting From Additional Filtering Features

---



# Filtering Partial Text in Queries

## EF.Functions.Like

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

```
_context.Authors.Where(a=>
```

```
    EF.Functions.Like(a.Name, "%abc%")
```

```
)
```



```
SQL LIKE(%abc%)
```

## LINQ Contains

```
property.Contains(abc)
```

```
_context.Authors.Where(a=>
```

```
    a.Name.Contains("abc")
```

```
)
```



```
SQL LIKE(%abc%)
```



# Finding an Entity Using its Key Value



**DbSet.Find(keyvalue)**



**This is the only task that Find can be used for**



**Not a LINQ method**



**Executes immediately**



**If key is found in change tracker, avoids unneeded database query**



# Skip and Take for Paging

- |                            |                            |
|----------------------------|----------------------------|
| 1. Aardvark                | 11. African Penguin        |
| 2. Abyssinian              | 12. African Tree Toad      |
| 3. Adelie Penguin          | 13. African Wild Dog       |
| 4. Affenpinscher           | 14. Ainu Dog               |
| 5. Afghan Hound            | 15. Airedale Terrier       |
| 6. African Bush Elephant   | 16. Akbash                 |
| 7. African Civet           | 17. Akita                  |
| 8. African Clawed Frog     | 18. Alaskan Malamute       |
| 9. African Forest Elephant | 19. Albatross              |
| 10. African Palm Civet     | 20. Aldabra Giant Tortoise |

Get first 10 animals

**Skip(0).Take(10)**

Get next 10 animals

**Skip(10).Take(10)**



# Sorting Data in Queries

---



# Sorting with LINQ

**OrderBy(o=>o.Property)**

**ThenBy(o=>o.Property)**

**OrderByDescending  
(o=>o.Property)**

**ThenByDescending  
(o=>o.Property)**



# LINQ Methods Can Be Combined

```
_context.Authors  
    .Where(a=>a.LastName=="Lerman")  
    .OrderByDescending(a=>a.FirstName).ToList();
```



# Aggregating Results in Queries

---





# LINQ to Entities Aggregate Execution Methods

First()

FirstOrDefault()

Single()

SingleOrDefault()

Last()

LastOrDefault()

Count()

LongCount()

Min(), Max()

Average(), Sum()

FirstAsync()

FirstOrDefaultAsync()

SingleAsync()

SingleOrDefaultAsync()

LastAsync()

LastOrDefaultAsync()

CountAsync()

LongCountAsync()

MinAsync(), MaxAsync()

AverageAsync(), SumAsync()

## No Aggregation

ToList()

AsEnumerable()

ToListAsync()

AsAsyncEnumerable()



# Aggregate Method Pointers



**First methods return the first of any matches**



**First/Single/Last will throw if no results are returned**



**Single methods expect only one match and will throw if there are none or more than one**



**FirstOrDefault/SingleOrDefault/LastOrDefault will return a null if no results are returned**



**Last methods require query to have an `OrderBy()` method otherwise will throw an exception**

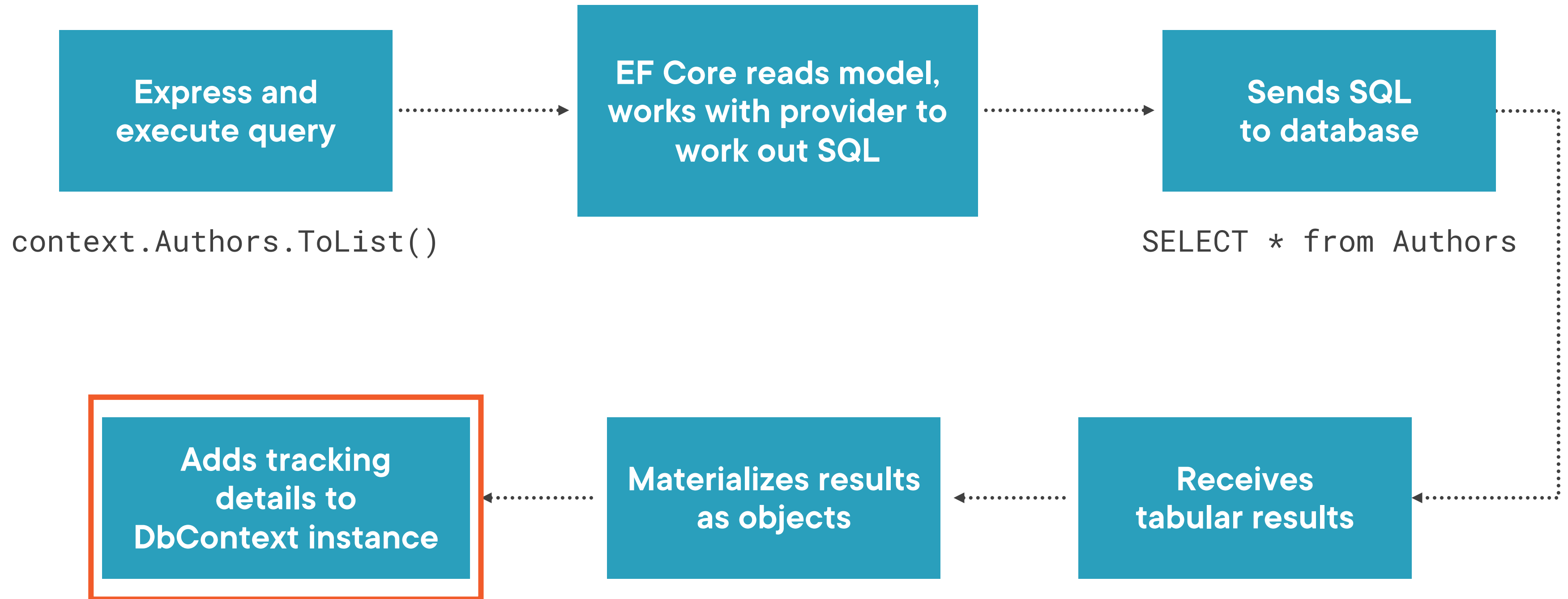


# Enhancing Query Performance When Tracking Isn't Needed

---



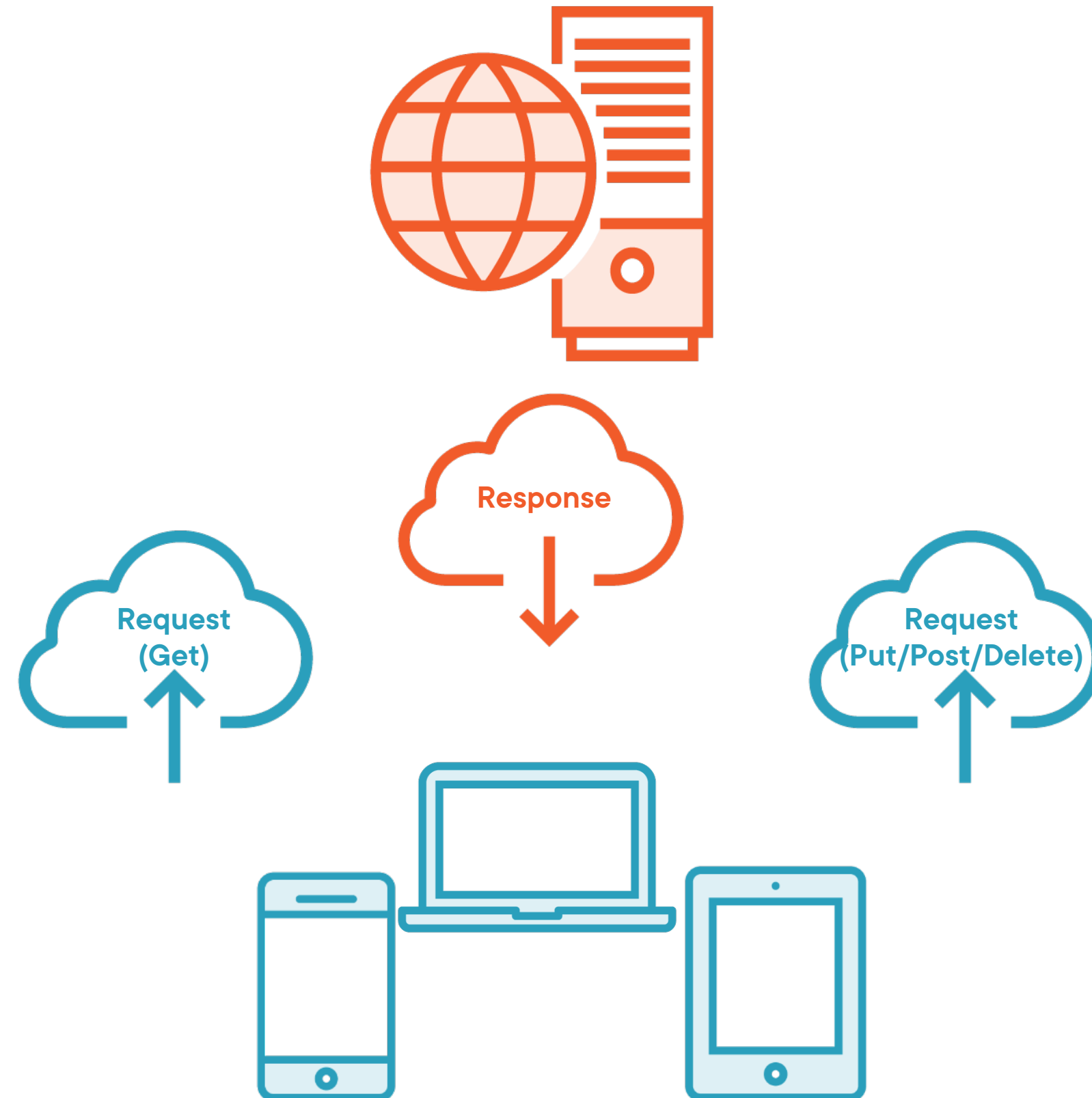
# Query Workflow



Change tracking is expensive.



# Most Common Scenario for No Tracking: Web and Mobile Apps



# No Track Queries and DbContexts

```
var author =  
context.Authors.AsNoTracking()  
.FirstOrDefault();
```

```
protected override void OnConfiguring  
(DbContextOptionsBuilder optionsBuilder)  
{  
    optionsBuilder  
        .UseSqlServer(myconnectionString)  
        .UseQueryTrackingBehavior  
            (QueryTrackingBehavior.NoTracking);  
}
```

◀ **AsNoTracking() returns a query, not a DbSet**

◀ **All queries for this DbContext will  
default to no tracking**

*Use DbSet.AsTracking() for special queries that  
need to be tracked*

## Review



**EF Core, with provider's help, transforms LINQ queries into SQL**

**Query execution is triggered with specific methods**

**Workflow includes sending SQL to database**

**EF Core materializes database results into objects**

**LINQ filter, sorting and aggregating methods are also translated into SQL**

**You can improve performance by disabling the default tracking when not needed**





Up Next:

Tracking and Saving Data with EF Core 6

---



# Resources



**Entity Framework Core on GitHub:** [github.com/dotnet/efcore](https://github.com/dotnet/efcore)



**EF Core Documentation:** [docs.microsoft.com/ef](https://docs.microsoft.com/ef)



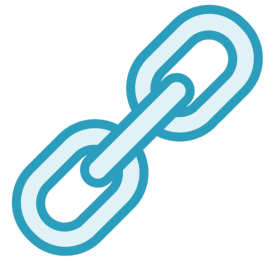
**Ethical Hacking: SQL Injection (Pluralsight course by Troy Hunt):**  
[app.pluralsight.com/library/courses/ethical-hacking-sql-injection](https://app.pluralsight.com/library/courses/ethical-hacking-sql-injection)



**EF Core 6: Fulfilling the Bucket List:** [codemag.com/Article/2111072](https://codemag.com/Article/2111072)



# Resources Cont.



**.NET Docs: Enumerable.FirstOrDefault Method**

[docs.microsoft.com/en-us/dotnet/api/system.linq.enumerable.firstordefault](https://docs.microsoft.com/en-us/dotnet/api/system.linq.enumerable.firstordefault)

