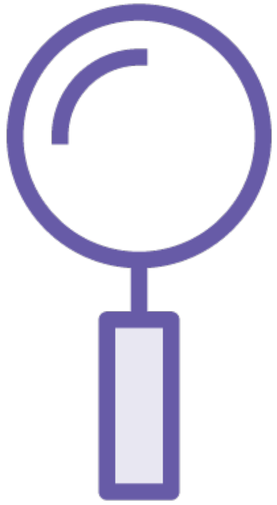
Optimizing Performance

Mark Heath
Software Architect

@mark_heath www.markheath.net

What Are We Aiming For?



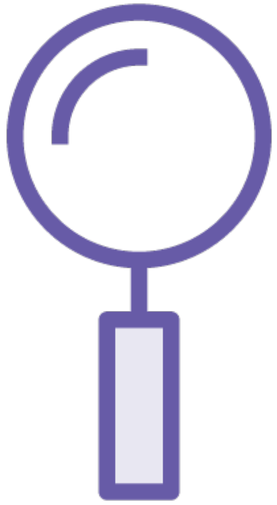
Solution A

Solution B

Solution C

LINQ pipelines can solve complex problems with a single C# expression

What Are We Aiming For?



Most Concise

Solves the problem in the fewest lines of code

Most Readable

More code, but easier to understand what's going on

Fastest

More complicated but produces results quickly

LINQ Performance Pitfalls

Slow:

```
var longest = books.First(b => b.Pages == books.Max(x => x.Pages))
```

Better:

```
var mostPages = books.Max(x => x.Pages);
var longest = books.First(b => b.Pages == mostPages);
```

Best:

```
var longest = books.MaxBy(x => x.Pages);
```

Overview

When and how to performance tune your LINQ Queries

LINQ vs foreach

Tools to optimize LINQ performance

- Parallel LINQ (PLINQ)

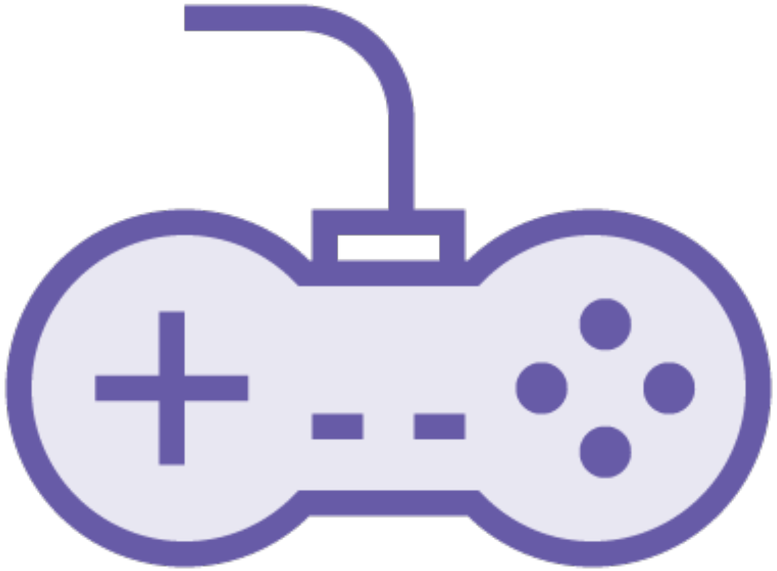
Know what's going on under the hood

Optimizing LINQ to Entities

When Should You Optimize?

Always measure first!

1. **Are there long-running methods in your LINQ pipeline?**
2. **Is this query going to be executed by a database?**
3. **Will this query operate on large in-memory collections of data?**
4. **Does this query need to be run repeatedly many times a second?**



The “game loop”

- Runs many times a second
- Makes complex decisions
- Can benefit from LINQ
- Needs to be fast

What if we abandon LINQ?

- Go faster with foreach?

Why is LINQ slower than for loops?

Small overhead of making additional method calls

- Lambda expressions
- IEnumerator.MoveNext

Usually the difference is minimal

Should you abandon LINQ for performance critical code?

- Not necessarily! ...

Demo

Speeding up LINQ with PLINQ

Don’t forget to write unit tests
before optimizing!

PLINQ is easy to use

- Simply add AsParallel
- Lets you control number of threads

Be careful

- It might make things go slower
- Remember to measure first

Additional LINQ Performance Enhancements

Nick Chapsas: “The fastest way to iterate a List in C# is NOT what you think!

https://www.youtube.com/watch?v=jUZ3VKFyB-A

https://benchmarkdotnet.org

```
foreach (int item in _items)
{
}
```

```
foreach (int item in CollectionsMarshal.AsSpan(_items))
{
}
```

There has been a huge focus on performance in recent .NET versions

Take advantage of .NET performance features (e.g. Span<T>)

Understanding the Implementation

LINQ is an abstraction

```
people.OrderBy(p => p.FirstName)
```

LINQ to objects – quicksort

LINQ to entities – ORDER BY

We don't see the implementation details

Potential for sub-optimal performance

Sub-optimal LINQ Performance

```
orders.Any(o => o.Status == "Refunded")
```



```
bool anyRefunded = false;
foreach (var order in orders)
{
    if (order.Status == "Refunded")
    {
        anyRefunded = true;
        break;
    }
}
```

```
orders.Where(o => o.Status == "Refunded")
      .Count() > 0
```



```
var count = 0;
foreach (var order in orders)
{
    if (order.Status == "Refunded")
    {
        count++;
    }
}
var anyRefunded = count > 0;
```

Sub-optimal LINQ Performance

```
orders.Any(o => IsAwaitingStock(o))
```



```
bool anyRefunded = false;
foreach (var order in orders)
{
    if (IsAwaitingStock(order))
    {
        anyRefunded = true;
        break;
    }
}
```

```
orders.Where(o => IsAwaitingStock(o))
      .Count() > 0
```



```
var count = 0;
foreach (var order in orders)
{
     if (IsAwaitingStock(order))
     {
        count++;
     }
}
var anyRefunded = count > 0;
```

More Performance Pitfalls

Some LINQ to Objects methods cache the entire sequence in memory
e.g. OrderBy, GroupBy, Reverse

```
var last = someSequence.Reverse().First();
```

Caches entire sequence in memory

```
var last = someSequence.ToArray()[someSequence.Count() - 1];
```

Also caches in memory (and enumerates twice!)

```
var last = someSequence.Last();
```

The right way to do it!

Understand what's going on under the hood

This will help you select the most efficient solution

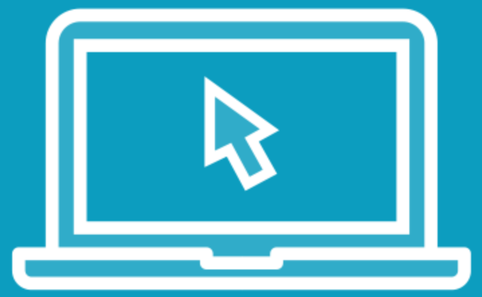
Optimizing LINQ to Entities

Your LINQ pipeline will get turned into SQL

Optimizing SQL performance

- Define appropriate indexes and keys
- Don’t pull down more rows or columns than you need to
- Construct a single query rather than using many

Demo



Avoiding returning too much data

Demo

Avoiding “Select N+1”

Summary

When to optimize LINQ queries

- Large amounts of data
- Running many times a second
- Long-running methods

How to optimize

- Falling back to foreach
- Parallel LINQ (PLINQ)

Before you optimize

- Measure first
- Have a good suite of unit tests

Summary

Know what's going on under the hood

- LINQ to objects – what algorithm will be used?
- LINQ to entities – what SQL will be generated?

Optimizing LINQ to entities

- Avoid returning too much data
- Avoid executing too many SQL queries

Up Next: Debugging and Testing