Understanding Deferred Execution, Streaming, and Non-streaming Operations



Paul D. Sheriff
Business / IT Consultant

psheriff@pdsa.com | www.pdsa.com



Module Goals



LINQ execution methods

- Deferred
- Immediate
- Non-streaming
- Streaming

Under-the-hood of streaming methods

Create custom filter (non -streaming)

Create custom filter using yield keyword (streaming)



LINQ Execution Methods

Types of LINQ Execution

Deferred

Immediate

Streaming

Non-streaming

LINQ Operator Classification https://bit.ly/3koPK3n



Deferred Execution

A LINQ query is a data structure ready to execute

Query is not executed until a value is needed

Execution happens

ForEach(), Count(), ToList(), OrderBy(), etc.



```
IEnumerable<Product> query =
  (from prod in Products
  where prod.Color == "Red"
  select prod);

foreach(var item in query) {
   Console.WriteLine(item.Name);
}
```

◆ Create a query (data structure)

■ Query is not executed until a value is requested

Immediate Execution

Query is executed immediately

An operator/method that requires all items to be processed (ToList, OrderBy, etc.)



```
IEnumerable<Product> query =
  (from prod in Products select prod)
  .ToList();
```

- ◆ Create a query (data structure)
- Because the .ToList() operator is applied, the query is executed immediately



Streaming Operators

Results can be returned prior to the entire collection is read

Examples

Distinct(), DistinctBy() GroupBy(), Join(), Select(), Skip(), Take(), Union(), Where()



```
var results = Products
  .Select(p => p)
  .Where(p => p.Color = "Red");
```

- Both Select() and Where() are deferred and streaming operations
- ◄ If they were not, then the Products collection would have to be looped through two times; once for the Select(), and once for the Where()



Non-Streaming Operators

All data in collection must be read before a result can be returned

Examples

Except(), ExceptBy(), GroupBy(), GroupJoin(), Intersect(), IntersectBy() Join(), OrderBy(), ThenBy()

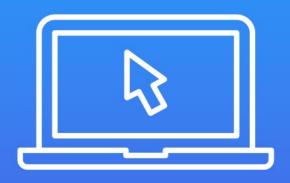


Demos of Deferred Execution





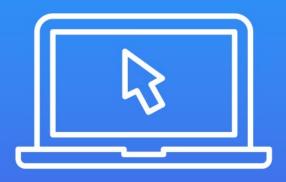
Deferred execution using foreach



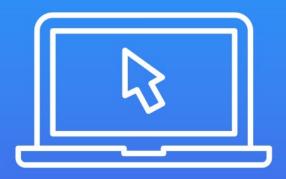
Step-through of deferred execution using foreach



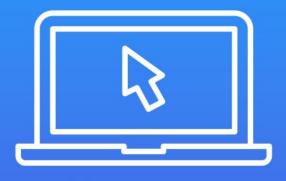
Deferred execution using enumerator



Show streaming nature of Where() and Take()



Simple filtering extension method

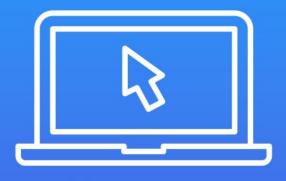


Custom extension method with Take()



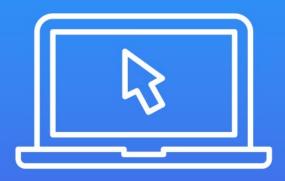
Use the yield keyword to create a stream





Using yield and Take()





Using yield and OrderBy()





Deferred execution can be advantageous

- Better performance
- Less iterations

Take advantage of yield for your extensions



Course Summary



LINQ is a great way to query collections

Very efficient operations

- Filtering
- Sorting
- Extracting
- Joining
- Grouping
- Aggregating

Take advantage of deferred operations



I hope you enjoyed this course!



Paul D. Sheriff

Business / IT Consultant

psheriff@pdsa.com | www.pdsa.com

