[1320] LINQ Operators

Note: This page is primarily for reference only.

There are many different operators that you can use in your LINQ queries. Here is a list of each of the operators grouped by function. The groups fall into 3 broad categories: Sequence->Sequence, Sequence->Element or Value, and Void->Sequence.

Sequence→Sequence

Most query operators fall into this category—accepting one or more sequences as input and emitting a single output sequence.

Filtering
IEnumerable<TSource> →IEnumerable<TSource>
Returns a subset of the original elements.
Where, Take, TakeWhile, Skip, SkipWhile, Distinct

Projecting
IEnumerable<TSource> →IEnumerable<TResult>
Transforms each element with a lambda function. SelectMany flattens nested sequences; Select and SelectMany perform inner joins, left outer joins, cross joins, and non-equi joins with LINQ to SQL and EF.
Select, SelectMany

Joining
IEnumerable<TOuter>, IEnumerable<TInner>→ IEnumerable<TResult>
Meshes elements of one sequence with another. Join and GroupJoin operators are designed to be efficient with local queries and support inner and left outer joins.
The Zip operator enumerates two sequences in step, applying a function over each element pair. Rather than naming the type arguments TOuter and TInner, the Zip operator names them TFirst and TSecond:
IEnumerable<TFirst>, IEnumerable<TSecond>→ IEnumerable<TResult>
Join, GroupJoin, Zip

Ordering
IEnumerable<TSource> →IOrderedEnumerable<TSource>
Returns a reordering of a sequence.
OrderBy, ThenBy, Reverse

Grouping
IEnumerable<TSource> →IEnumerable<IGrouping<TKey,TElement>>

Groups a sequence into subsequences.

GroupBy

Set operators

IEnumerable<TSource>, IEnumerable<TSource>→ IEnumerable<TSource>

Takes two same-typed sequences and returns their commonality, sum, or difference.

Concat, Union, Intersect, Except

Conversion methods: Import

IEnumerable→IEnumerable<TResult>

OfType, Cast

Conversion methods: Export

IEnumerable<TSource> →An array, list, dictionary, lookup, or sequence

ToArray, ToList, ToDictionary, ToLookup, AsEnumerable, AsQueryable

Sequence→Element or Value

The following query operators accept an input sequence and emit a single element or value.

Element operators

IEnumerable<TSource> →TSource

Picks a single element from a sequence.

First, FirstOrDefault, Last, LastOrDefault, Single, SingleOrDefault, ElementAt, ElementAtOrDefault, DefaultIfEmpty

Aggregation methods

IEnumerable<TSource> →scalar

Performs a computation across a sequence, returning a scalar value (typically a number).

Aggregate, Average, Count, LongCount, Sum, Max, Min

Quantifiers

IEnumerable<TSource> →bool

An aggregation returning true or false.

All, Any, Contains, SequenceEqual

Void→Sequence

In the third and final category are query operators that produce an output sequence from scratch.

Generation methods

void→IEnumerable<TResult>

Manufactures a simple sequence.

Empty, Range, Repeat