39

LINQ

Introducing LINQ

- LINQ is a 'uniform query syntax' that allows you to retrieve data from various data sources such as arrays, collections, databases, XML files.

LINQ Query - Example

```
var result = Customers.Where(temp => temp.Location == "Hyderabad").ToList();
//returns a list of customers from Hyderabad location.
```

Advantages of LINQ

- **Single Syntax - To Query Multiple Data Sources**
  - Developer uses the same LINQ syntax to retrieve information from various data sources such as collections, SQL Server database, Entity Framework DbSet's, ADO.NET DataSet etc.
- **Compile-Time Checking of Query Errors**

- Errors in the LINQ query will be identified while compilation time / while writing the code in Visual Studio.

- **IntelliSence Support**
  - The list of properties of types are shown in VS IntelliSence while writing the LINQ queries.

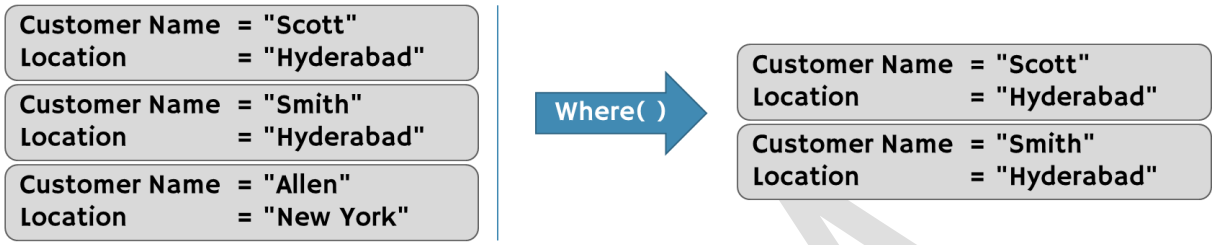
LINQ Extension Methods

All the LINQ extension methods are defined in the static class called IEnumerable in System.Linq namespace.

Classification	LINQ Extension Methods / LINQ Operators
Filtering	Where, OfType
Sorting	OrderBy, OrerByDescending, ThenBy, ThenByDescending, Reverse
Grouping	GroupBy
Join	Join
Project	Select, SelectMany
Aggregation	Average, Count, Max, Min, Sum
Quantifiers	All, Any, Contains
Elements	ElementAt, ElementAtOrDefault, First, FirstOrDefault, Last, LastOrDefault, Single, SingleOrDefault
Set Operations	Distinct, Except, Intersect, Union
Partitioning	Skip, SkipWhile, Take, TakeWhile
Concatenation	Concat
Equality	SequenceEqual
Generation	DefaultEmpty, Empty, Range, Repeat
Conversion	AsEnumerable, AsQueryable, Cast, ToArray, ToDictionary, ToList

Where( )

› Where() method filters collection based on given lambda expression and returns a new collection with matching element.



Where Extension Method - Declaration

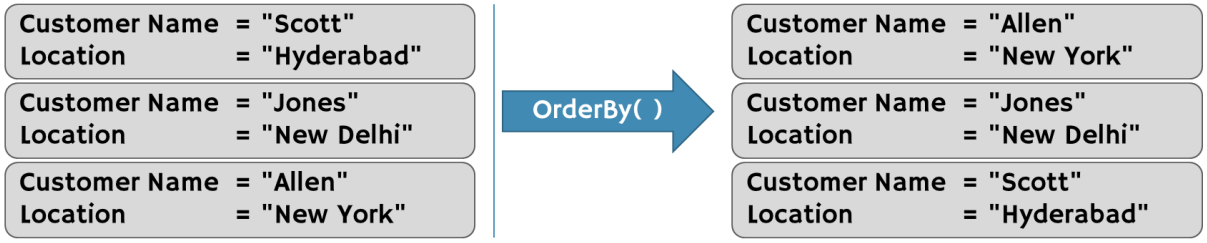
```
Where(Func<TSource, bool>  predicate)
```

Where Extension Method - Usage

```
var  result = Customers.Where(temp => temp.Location == "Hyderabad").ToList( );
//returns a list of customers from Hyderabad location.
```

OrderBy( )

› OrderBy() method sorts collection based on given lambda expression (property) and returns a new collection with sorted elements.



OrderBy Extension Method - Declaration

```
OrderBy(Func<TSource, TKey>  keySelector)
```

OrderBy Extension Method - Usage

```
var  result = Customers.OrderBy(temp => temp.CustomerName).ToList( );
//returns a list of customers sorted based on customer name.
```

OrderByDescending Extension Method - Declaration

```
OrderByDescending(Func<TSource, TKey>  keySelector)
```

OrderByDescending Extension Method - Usage

```
var  result = Customers.OrderByDescending(temp => temp.CustomerName).ToList( );
//returns a list of customers sorted based on customer name in descending order.
```

ThenBy Extension Method - Declaration

```
ThenBy(Func<TSource, TKey>  keySelector)
```

ThenBy Extension Method - Usage

```
var  result = Customers.OrderBy(temp => temp.Location)
              .ThenBy(temp => temp.CustomerName).ToList( );
//returns a list of customers sorted based on location and customer name.
```

ThenByDescending Extension Method - Declaration

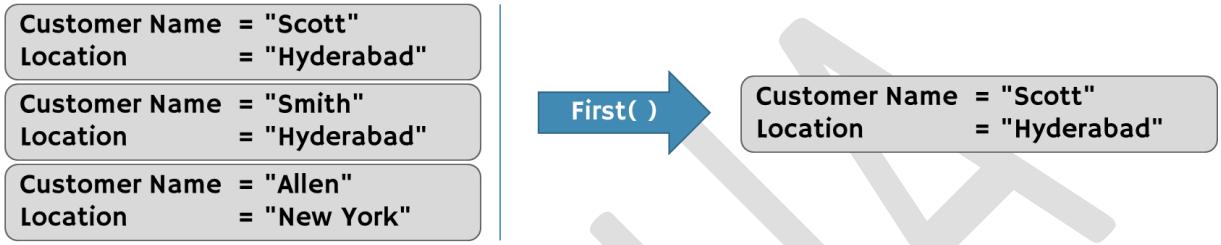
```
ThenByDescending(Func<TSource, TKey>  keySelector)
```

ThenByDescending Extension Method - Usage

```
var  result = Customers.OrderBy(temp => temp.Location)
              .ThenByDescending(temp => temp.CustomerName).ToList( );
//returns a list of customers sorted based on location (ascending) and customer name (descending).
```

First( )

- First() method returns first element in the collection that matches with the collection.
- It throws exception if no element matches with the condition.



First Extension Method - Declaration

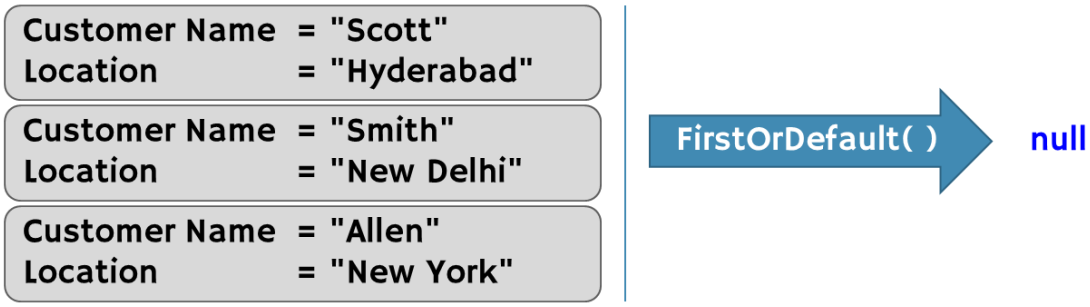
```
First(Func<TSource, bool>  predicate)
```

First Extension Method - Usage

```
var  result = Customers.First(temp => temp.Location == "Hyderabad");
//returns the first customer from Hyderabad location.
```

FirstOrDefault( )

- FirstOrDefault() method returns first element that matches with the condition.
- It returns null if no element matches with the condition.



FirstOrDefault Extension Method - Declaration

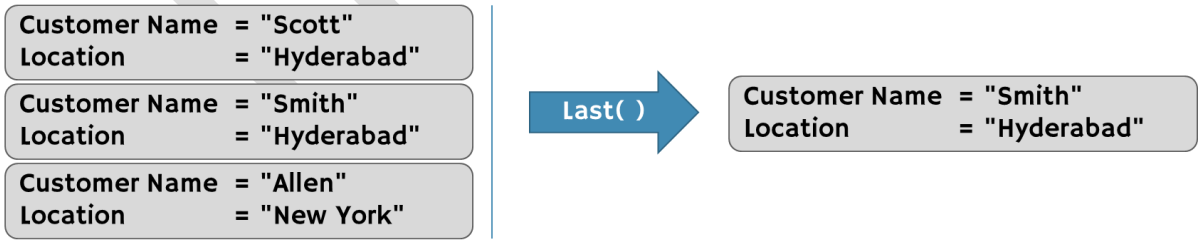
```
FirstOrDefault(Func<TSource, bool>  predicate)
```

FirstOrDefault Extension Method - Usage

```
var  result = Customers.FirstOrDefault(temp => temp.Location == "London");
//returns the first customer from London location (or) returns null if not exists.
```

Last( )

- Last() method returns last element in the collection that matches with the collection.
- It throws exception if no element matches with the condition.



Last Extension Method - Declaration

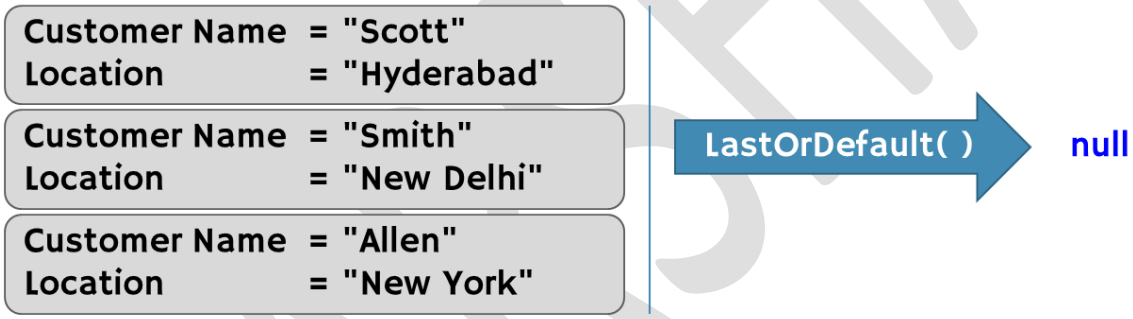
```
Last(Func<TSource, bool>  predicate)
```

Last Extension Method - Usage

```
var result = Customers.Last(temp => temp.Location == "Hyderabad");
//returns the last customer from Hyderabad location.
```

LastOrDefault( )

- LastOrDefault() method returns last element that matches with the condition.
- It returns null if no element matches with the condition.



LastOrDefault Extension Method - Declaration

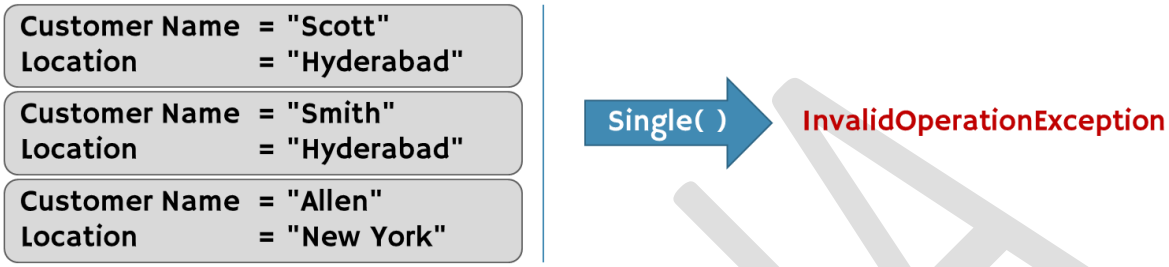
```
LastOrDefault(Func<TSource, bool> predicate)
```

LastOrDefault Extension Method - Usage

```
var result = Customers.LastOrDefault(temp => temp.Location == "London");
//returns the last customer from London location (or) returns null if not exists.
```

Single( )

- It returns first element (only one element) that matches with the collection.
- It throws exception if no element or multiple elements match with the condition.



Single Extension Method - Declaration

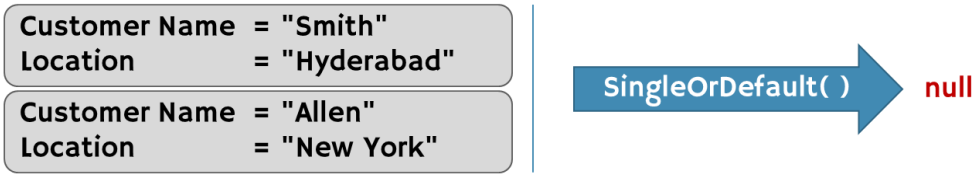
```
Single(Func<TSource, bool>  predicate)
```

Single Extension Method - Usage

```
var  result = Customers.Single(temp => temp.Location == "Hyderabad");
//returns the first (only one customer) from Hyderabad location.
but it throws exception if none / multiple elements matches with the condition.
```

SingleOrDefault( )

- It returns first element (only one element) that matches with the collection.
- It returns null if no element matches with the condition; but it throws exception if multiple elements match with the condition.



SingleOrDefault Extension Method - Declaration

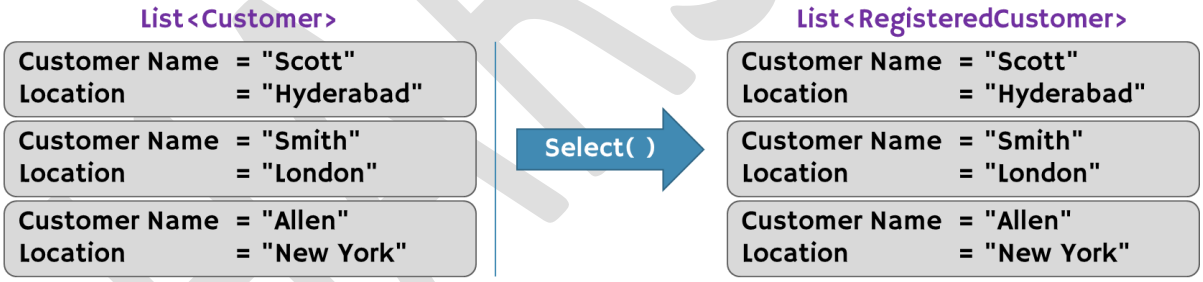
```
SingleOrDefault(Func<TSource, bool> predicate)
```

SingleOrDefault Extension Method - Usage

```
var result = Customers.SingleOrDefault(temp => temp.Location == "London");
//returns the first (only one customer) from London location.
it throws exception if multiple elements matches with the condition; but null in case of no match.
```

Select( )

- It returns collection by converting each element into another type, based on the conversion expression.



Select Extension Method - Declaration

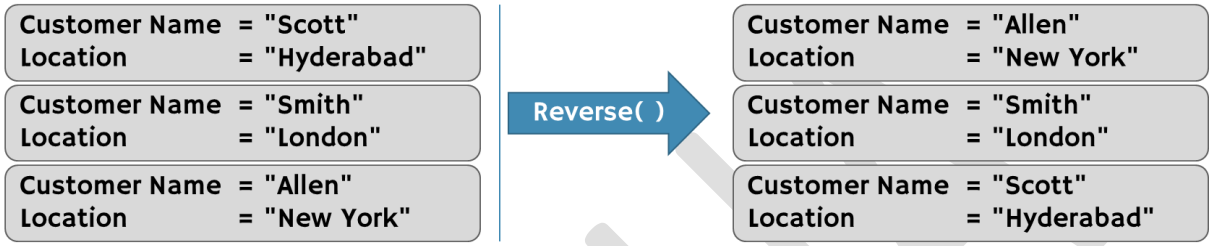
```
Select(Func<TSource, TResult> selector)
```

Select Extension Method - Usage

```
var result = Customers.Select(temp => new RegisteredCustomer()
        { CustomerName = temp.CustomerName, Location = temp.Location } );
//converts all customers into a collection of RegisteredCustomer class.
```

Reverse( )

› It reverses the collection.



Reverse Extension Method - Declaration

```
Reverse( )
```

Reverse Extension Method - Usage

```
var result = Customers.Reverse( );    //reverses the customers collection
```

Min, Max, Count, Sum, Average( )

- It performs aggregate operations such as finding minimum value of specific property of all elements of a collection.

Min, Max, Count, Sum, Average

```
var result1 = Students.Min( temp => temp.Marks );     //minimum value of Marks property
var result2 = Students.Max( temp => temp.Marks );     //maximum value of Marks property
var result3 = Students.Count( );                      //count of elements
var result4 = Students.Sum( temp => temp.Marks);      //sum value of Marks property
var result5 = Students.Average( temp => temp.Marks);  //average value of Marks property
```