LINQ – Common Methods

LINQ stands for **L**anguage-**IN**tegrated **Q**ueries. LINQ is a technology that helps in using query capabilities and integrations in C# directly.

LINQ generally operates on the collection types in C# and comes with some great extension methods which serve a variety of purposes in working with collections of types

Common LINQ methods:

`Where()` - returns all the elements from the collection which satisfy a given condition

Syntax: `var result = listName.Where(x => condition);`

Since LINQ methods return a generic type of **IEnumerable**, we use the **IEnumerable** generic data type **`var`** to hold the result.

The **`var`** type should be cast to the type of data in the **`List`** before use.

First() - returns the "first" element in the collection, for an optional condition. If a condition is passed to the function as a predicate, the method returns the first element in the list which satisfies the predicate. **If no elements satisfy the condition, it throws an exception.**

Syntax: `var result = listName.First(x => condition);`

FirstOrDefault() - returns the first element in the collection which satisfy an optional condition.

The difference between **`First()`** is where there are no elements that satisfy the condition or the collection is empty – the method returns the default value for that type: NULL for reference types and respective default value for value types.

Syntax: `var result = listName.FirstOrDefault(x => condition);`

`Single()` - returns the only element in the collection which satisfies a given condition.

There should be only a single element in the collection which satisfies the condition – for example a primary key in a table which is one and unique.

If there are more than one elements which satisfy the condition for the **`Single()`** method, **it throws an exception.**

Syntax: `var result = listName.Single(x => condition);`

SingleOrDefault() - returns the only element which satisfies a given condition. But when this condition returns more than one element or the collection is empty, the method returns a default value – NULL for a reference type and respective default value for a value type.

Syntax: `var result = listName.SingleOrDefault(x => condition);`

LINQ – Common Methods

Any() - returns a boolean value indicating if there exists any element in the collection, which satisfies the condition. **If no condition is provided, the method just returns if the collection is empty or not.**

Syntax : `var result = listName.Any(x => condition);`

OrderBy() - sorts the elements in a given collection in ascending order based on a given condition and returns the sorted collection.

Syntax : `var result = listName.OrderBy(variable);`

To order by multiple values append the subsequent condition using a **ThenBy()** method.

`OrderByDescending()` - Sort the collection is in a descending order.

To sort based on more than one columns, one can use **`ThenByDescending()`** method chained to the **`OrderBy()`** method.

Syntax : `var result = listName.OrderByDescending(variable);`

`ToList()` – Convert an **`IEnumerable`** type a **`List`** type.

This conversion is important in cases where we would want to use a concrete type **`List`** in place of an abstract **`IEnumerable`** or when we want to use the methods available in a "List" type over a collection of an abstract type.