2 ways to write a LINQ query:

1. Query Syntax
2. Method Syntax

Query Syntax:

Query Syntax method is a declarative way to call the LINQ operators. It looks similar to SQL and is defined within the language such as C#.

Var query = from List in name select blah

Method Syntax:

The method syntax calls the LINQ operators using methods.

List.Where( x => x.name == “John”);

These methods are part of System.Linq in the .NET framework.

The 2 methods are semantically equivalent.

A query expression operates on one or more data stores using one or more query operators.

A query statement must end in a select clause or group clause.

IEnumerable.Any() – returns if any elements in the collection

In a dictionary an enumerator is someone who collects census data by visiting individual homes to get the data.

LINQ uses deferred execution – deferred execution is when the query is defined but not executed at the point of declaration. When the result is used for the first time, the LINQ query is executed. This means that a few interesting things:

* If you use query.first() for the first element then the whole collection isn’t iterated over, the 1st is returned.
* If the collection is changed, the query result will change without having to restate the query
* Queries can be defined and called on when needed.

LINQ query syntax is built into the language so the queries must be converted by the CLR when the code is compiled. Methods are called in place of the queries. You can call these methods directly and they are known as method query syntax. They are defined as extension methods

Extension methods are methods on a class that extend its functionality. Useful because they can be easily discoverable.

Publis static string AnExtension(this string source)

Extension methods must be a static method within a static class.

In intellisense extension methods appear with a downward icon.

Extension methods make your helper methods more discoverable and easier to call

LINQ qury methods were implemented in .NET 3.5 as extension methods of the Enumerable type. It’s in the System.LINQ namespace.

System.Linq namespace.

Any object that you can iterate over in a foreach loop has access to the LINQ extension methods because they will inherit from IEnumerable.

Method syntax also uses deferred execution.

IEnumerable.First() has an overload that takes a delegate to specify a condition the search must satisfy. We can provide lambda or delegate of type Func that returns a value.

Lambda expressions leverage implicit typing. .NET knows the type you’re iterating through.

=> lambda operator. Separates the params from the expression itself.

Param => body of function

First() will throw an InvalidOperation exception if the searched for entry doesn’t exist. FirstOrDefault is the recommended version because if it doesn’t exist a default value will be returned. Default value is null.

If a lambda has multiple lines, braces and return keyword are required.

Debug.Writeline ?

Can we create a collection of query IEnumerables that we call in at different times?

.Skip(int number) – skips the first number of entries in a collection.

Multiple LINQ expressions used in a single query is known as chaining.

Fluent programming – the output of one method is the input to the next creating a string of operations that together perform a single task.

LINQ provides:

* Traversal operators
* Filter operators
* Projection operators

Traversal operators provide movement through a colletion with LINQ.

Filter operators allows filtering of a list. E.g. Where

Projection operators allows for reshaping of a collection. E.g. Select.

Standard set of query operators allows the extension methods to be applied to any collection that implements IEnumerable of T.

Query syntax is only available for a subset of the query operators.