***LINq***

LINQ is Language Integrated Query .

 LINQ adds a rich, standardized query syntax in .NET programming languages that allows developers to interact with any type of data.

There are various types of LINQ that we have as shown above in the pictorial representation.

* LINQ to Objects
* LINQ to Datasets
* LINQ to SQL
* LINQ to Entities
* LINQ to XML

Advantages of LINQ

Following are the advantages of using linq in our applications.

1. It has full type checking at compile time.
2. It has intellisense so we can avoid silly errors.
3. Its query can be reused.
4. We can debug it using .NET debugger.
5. It supports filtering, sorting, ordering, grouping with much less effort.

Disadvantages of LINQ

Following are the drawbacks of using linq in our applications.

1. Since it is written in the code so we cannot make use of Cache execution plan which is a SQL feature as we do in stored procedure.
2. Writing complex queries in LINQ is a bit tedious as compared to SQL.
3. If the query is not written properly then performance is degraded.
4. We have to build the project and deploy the DLL every time some change is done in the query.

LINQ has its own advantages and disadvantages so based on requirements we can use linq in our applications.

There are two ways in which we can write queries in LINQ

* Using Query Syntax
* Using Method Syntax

## LINQ Query Syntax

In LINQ. we write LINQ query by following certain rules regarding the syntax which is quite different from the SQL. We use below LINQ query syntax hierarchy to write a LINQ query.

from <variable> in <collection>

<where, joining, grouping, operators etc.> <lambda expression>

<select or groupBy operator> <format the results>

This order is to be followed while writing queries in LINQ. “from” keyword will form the starting point of the LINQ query followed by a user defined variable followed by “in” which actually specifies our source collection or Data source followed by a where clause, if there is a certain condition in the query can be used before the select to filter out the records and select is followed by group by and into clause.

The order of the clauses in LINQ query is as below

| **Clause** | **Description** |
| --- | --- |
| From | [Identifier] |
| In | [Source Collection] |
| Let | [Expression] |
| Where | [Boolean Expression] |
| order by | [Expression] |
| Select | [Expression] |
| group by | [Expression] |
| Into | [Expression] |

The following simple code snippets shows the linq query syntax in detail.

**C# Code**

int[] Num = { 1, 2, 3, 4, 5, 6, 7, 8, 9 };

IEnumerable<int> result = from numbers in Num

                                            where numbers >3

                                            select numbers;

## LINQ Query Syntax Example

An example is shown below that uses LINQ. In this example we have an integer array having nine elements. We will print the elements from given array where element value greater than **3** and display them using LINQ.

**C# Code**

using System;

using System.Collections.Generic;

using System.Linq;

namespace Linqtutorials

{

class Program

{

static void Main(string[] args)

{

int[] Num = { 1, 2, 3, 4, 5, 6, 7, 8, 9 };

//LINQ Query Syntax to Print Numbers Greater than 3

IEnumerable<int> result = from numbers in Num

where numbers >3

select numbers;

foreach (var item in result)

{

Console.WriteLine(item);

}

Console.ReadLine();

}

}

}