**LINQ SEMINAR Manual**

Prerequisite:

Language:Basic knowledge of .Net Framework3.5/4.5, C#,

Database: SQL

Software: Visual Studio 2008 onward

SQL Server 2012

|  |  |  |
| --- | --- | --- |
| **TABLE OF CONTENT** | | |
| **Sr.no** | **Topic** | **Duration(MIN)** |
| 1 | [Introduction](#Introduction) | 5 |
| 2 | [What is LINQ & Advantages](#WhatisLINQ) | 10 |
|  | Tea Break | 10 |
| 3 | [LINQ API](#LINQAPI) | 10 |
| 4 | [Lambda Expression](#LambdaExpressions) | 10 |
| 5 | [LINQ Method Syntax](#LINQMethodSyntax) | 5 |
| 6 | [LINQ Query Syntax](#QuerySyntax) | 5 |
| 7 | [LINQ TO SQL](#linqtosql) | 15 |
| 8 | [Restrict Operator Example](#RestrictOperator) | 15 |
| 9 | [Project operator Example](#ProjectOperator) | 10 |
| 10 | [Join Operator Example](#JoinOperator) | 10 |
| 11 | [Group by/Having(Example)](#GroupbyOperator) | 10 |
| 12 | [Questions](#Question) | 5 |
|  | Total Hours | 120 |

1. **[Introduction](#TableofContent)**

Language-Integrated Query (LINQ) is a powerful query language introduced with .NET 3.5 & Visual Studio 2008. LINQ can be used with C# or Visual Basic to query different data sources.

LINQ Seminar are packed with easy to understand explanations, real-world examples, useful tips, informative notes and points to remember.

**2. [What is LINQ&Advantages:](#TableofContent)**

LINQ (Language Integrated Query) is uniform query syntax in C# and VB.NET used to save and retrieve data from different sources. It is integrated in C# or VB, thereby eliminating the mismatch between programming languages and databases, as well as providing a single querying interface for different types of data sources.

For example, SQL is a Structured Query Language used to save and retrieve data from a database. In the same way, LINQ is a structured query syntax built in C# and VB.NET used to save and retrieve data from different types of data sources like an Object Collection, SQL server database, XML, web service etc.

LINQ always works with objects so you can use the same basic coding patterns to query and transform data in XML documents, SQL databases, ADO.NET Datasets, .NET collections, and any other format for which a LINQ provider is available.

[](http://www.tutorialsteacher.com/Content/images/linq/linq-usage.PNG)

## **Advantages of LINQ**

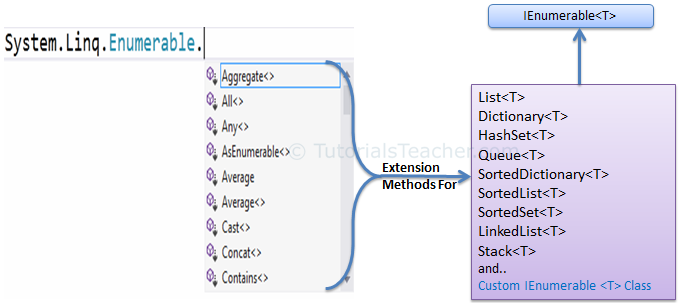
* **Familiar language**: Developers don’t have to learn a new query language for each type of data source or data format.
* **Less coding**: It reduces the amount of code to be written as compared with a more traditional approach.
* **Readable code**: LINQ makes the code more readable so other developers can easily understand and maintain it.
* **Standardized way of querying multiple data sources**: The same LINQ syntax can be used to query multiple data sources.
* **Compile time safety of queries**: It provides type checking of objects at compile time.
* **IntelliSense Support**: LINQ provides IntelliSense for generic collections.
* **Shaping data:** You can retrieve data in different shapes.

**3. [LINQ API](#_top)**

# **Enumerable**

**E**numerable class includes extension methods for the classes that implement [IEnumerable<T>](http://msdn.microsoft.com/en-us/library/9eekhta0(v=vs.110).aspx" \o "IEnumerable<T> class members" \t "http://www.tutorialsteacher.com/linq/_blank) interface, this include all the collection types in [System.Collections.Generic](http://www.tutorialsteacher.com/csharp/csharp-generic-collections" \o "C# Generic collections" \t "http://www.tutorialsteacher.com/linq/_blank) namespaces such as List<T>, Dictionary<T>, SortedList<T>, Queue<T>, HashSet<T>, LinkedList<T> etc.

The following figure illustrates that the extension methods included in Enumerable class can be used with generic collection in C# or VB.Net.

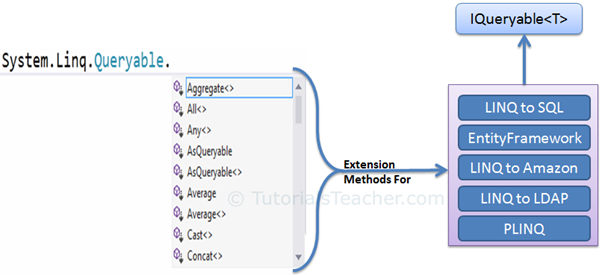
[](http://www.tutorialsteacher.com/Content/images/linq/Enumerable-extension-methods.png)IEnumerable<T> extension methods in Enumerable class

## Queryable

The **Queryable** class includes extension methods for classes that implement [IQueryable<t>](http://msdn.microsoft.com/en-us/library/vstudio/bb351562(v=vs.100).aspx" \o "IQueryable<T> members" \t "http://www.tutorialsteacher.com/linq/_blank) interface. IQueryable<T> is used to provide querying capabilities against a specific data source where the type of the data is known. For example, Entity Framework api implements IQueryable<T> interface to support LINQ queries with underlaying database like SQL Server.

Also, there are APIs available to access third party data; for example, LINQ to Amazon provides the ability to use LINQ with Amazon web services to search for books and other items by implementing IQueryable interface.

The following figure illustrates that the extension methods included in Queryable class can be used with various native or third party data providers.

[](http://www.tutorialsteacher.com/Content/images/linq/Queryable-extension-methods.png)IQueryable extension methods in Queryable class

Visit MSDN to know all the extension methods of [Enumerable](https://msdn.microsoft.com/en-us/library/system.linq.enumerable(v=vs.110).aspx" \t "http://www.tutorialsteacher.com/linq/_blank) and [Queryable](https://msdn.microsoft.com/en-us/library/system.linq.queryable(v=vs.110).aspx" \t "http://www.tutorialsteacher.com/linq/_blank) class.

IMG_259Points to Remember :

1. Use **System.Linq** namespace to use LINQ.
2. LINQ api includes two main static class Enumerable & Queryable.
3. The static **Enumerable** class includes extension methods for classes that implements IEnumerable<T> interface.
4. IEnumerable<T> type of collections are in-memory collection like List, Dictionary, SortedList, Queue, HashSet, LinkedList
5. The static **Queryable** class includes extension methods for classes that implements IQueryable<T> interface
6. Remote query provider implements IQueryable<T>. eg. Linq-to-SQL, LINQ-to-Amazon etc.

# 4. [Lambda Expression](#TableofContent)

C# 3.0(.NET 3.5) introduced the lambda expression along with LINQ. The lambda expression is a shorter way of representing [anonymous method](http://www.tutorialsteacher.com/csharp/csharp-anonymous-method) using some special syntax.

To create a lambda expression, you specify input parameters (if any) on the left side of the lambda operator [=>](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/operators/lambda-operator), and you put the expression or statement block on the other side. For example, the lambda expression x => x \* x specifies a parameter that’s named x and returns the value of xsquared.

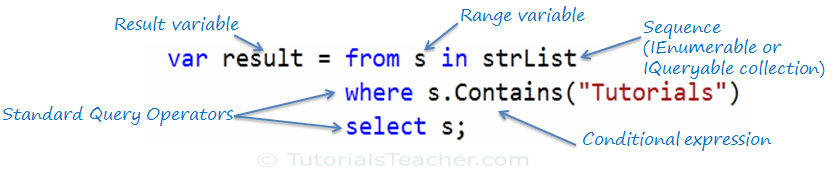
* (input-parameters) => expression
* (x, y) => x == y
* (int x, string s) => s.Length > x
* () => SomeMethod()
* (input-parameters) => { statement; }

Fore more details refers the following links

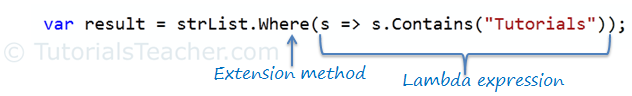
https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/statements-expressions-operators/lambda-expressions

## 5. [Query Syntax](#TableofContent)

Query syntax is similar to SQL (Structured Query Language) for the database. It is defined within the C# or VB code.

[](http://www.tutorialsteacher.com/Content/images/linq/linq-query-syntax.png)

# 6. [LINQ Method Syntax](#TableofContent)

[](http://www.tutorialsteacher.com/Content/images/linq/linq-method-syntax.png)

**[7.LINQ TO SQL](#linqtosql)**

1. Open the sql server and write the following queries

create database linqpractice

Use database linqpractice

CREATE TABLE Employee (

employee\_id BIGINT IDENTITY(1, 1) NOT NULL,

first\_name VARCHAR(50) NOT NULL,

last\_name VARCHAR(50) NOT NULL,

salary DECIMAL(10,0) NOT NULL,

joining\_date DATETIME NOT NULL,

department VARCHAR(20) NOT NULL

);

INSERT INTO Employee

VALUES

('John', 'Abraham', 1000000, '2013-01-01 12:00:00 AM', 'Banking'),

('Michael', 'Clarke', 800000, '2013-01-01 12:00:00 AM', 'Insurance'),

('oy', 'Thomas', 700000, '2013-02-01 12:00:00 AM', 'Banking'),

('Tom', 'Jose', 600000, '2013-02-01 12:00:00 AM', 'Insurance'),

('Jerry', 'Pinto', 650000, '2013-02-01 12:00:00 AM', 'Insurance'),

('Philip', 'Mathew', 750000, '2013-01-01 12:00:00 AM', 'Services'),

('TestName1', '123', 650000, '2013-01-01 12:00:00 AM', 'Services'),

('TestName2', 'Lname%', 600000, '2013-02-01 12:00:00 AM', 'Insurance');

CREATE TABLE Incentives (

employee\_ref\_id BIGINT NOT NULL,

incentive\_date DATE NOT NULL,

incentive\_amount DECIMAL(10,0) NOT NULL

);

INSERT INTO Incentives

VALUES

(1, '2013-02-01', 5000),

(2, '2013-02-01', 3000),

(3, '2013-02-01', 4000),

(1, '2013-01-01', 4500),

(2, '2013-01-01', 3500);

* 1. Open Visual Studio and create new Project Console Application. And give the name as Linqpractice.
  2. Right Click the solution explorer->Add->New Item->Linq To SQL Classes and give the name as DataClasses1.dbml and add it in the solution.
  3. From the server explorer drag the Employee and Incentives table so that Database is mapped to Object
  4. Now open the DataClasses1.designer.cs file and see how the database is mapped to Object
  5. Copy the DataClasses1DataContext class name from the DataClasses1.designer.cs because all the object are there in this.

**[8. Restrict Operator Example](#TableofContent)**

* 1. Open the program.cs file

And write the following following functions.

public void whereexample()

{

DataClasses1DataContext d = new DataClasses1DataContext();

/\*where clause(restict operator)

Get employee details from employee table whose employee name is “John”

Ans: \*/

//method syntax

var res= d.Employees.Where(k => k.first\_name == "John");

foreach (var item in res)

{

Console.WriteLine(item.first\_name+" "+item.last\_name+" "+item.joining\_date+" "+item.salary);

}

//query syntax

var res1= from emp in d.Employees

select emp;

foreach (var item in res)

{

Console.WriteLine(item.first\_name + " " + item.last\_name + " " + item.joining\_date + " " + item.salary);

}

}

**[9. Project Operator Example(selected Field)](#TableofContent)**

Public void projectexample()

{

DataClasses1DataContext d = new DataClasses1DataContext();

/\*.Get FIRST\_NAME,LAST\_NAMe from employee table

Ans :\*/

//Query Syntax

var res = from emp in d.Employees

select new { emp.first\_name,emp.last\_name};

foreach (var item in res)

{

Console.WriteLine(item.first\_name+" "+item.last\_name);

}

}

**[10. Join Operator](#TableofContent)**

public void joinexample()

{

DataClasses1DataContext d = new DataClasses1DataContext();

/\* Select first\_name, incentive amount from employee and

\* incentives table for those employees who have incentives

Ans :\*/

var res = from emp in d.Employees

join inc in d.Incentives on emp.employee\_id equals inc.employee\_ref\_id

into prods

from p in prods

select new { p.incentive\_amount, emp.first\_name };

foreach (var item in res)

{

Console.WriteLine(item.first\_name.TrimEnd() + " " + item.incentive\_amount);

}

}

**[11.Group by/Having(Example)](#TableofContent)**

/\*45.Get department wise maximum salary from employee table order by salary ascending

Ans :\*/

public void querygroupby()

{

DataClasses1DataContext d = new DataClasses1DataContext();

var res = from emp in d.Employees

group emp by emp.department into emp1

select new

{

Department = emp1.First().department,

MaxSalary = emp1.Max(k => k.salary)

};

foreach (var item in res)

{

Console.WriteLine(item.Department + " " + item.MaxSalary);

}

}

//group by having

/\*.Select department,total salary with respect to a department from employee table where total salary greater than 800000 order by Total\_Salary descending

Ans :\*/

public void querygroupbyhaving()

{

var res = (from emp in d.Employees

group emp by emp.department into g

orderby g.Sum(k => k.salary) descending

select new

{

department = g.First().department,

totalsalary = g.Sum(p => p.salary)

}

).Where(k => k.totalsalary >= 800000);

foreach (var item in res)

{

Console.WriteLine(item.department + " " + item.totalsalary);

}

I hope you get the overview of how LINQ works

**[12. Solve the below Questions](#TableofContent)**

1. The full form of LINQ is \_\_\_\_\_\_\_.
2. A class must implement \_\_\_\_\_\_\_\_\_\_\_\_ interface in order to provide querying facility using LINQ
3. Output of below queries

var teenAgerStudent = from s in studentList

where s.Age > 12 && s.Age < 20

select s;

1. var teenAgerStudents = studentList.Where(s => s.Age > 12 && s.Age < 20)

.ToList<Student>();

The above query is an example of \_\_\_\_ syntax.

5. s => s.Age > 12 && s.Age < 20; is an example of which expression