Day-9

09 December 2020 08:37

Kiss : Keep it simple stupid.

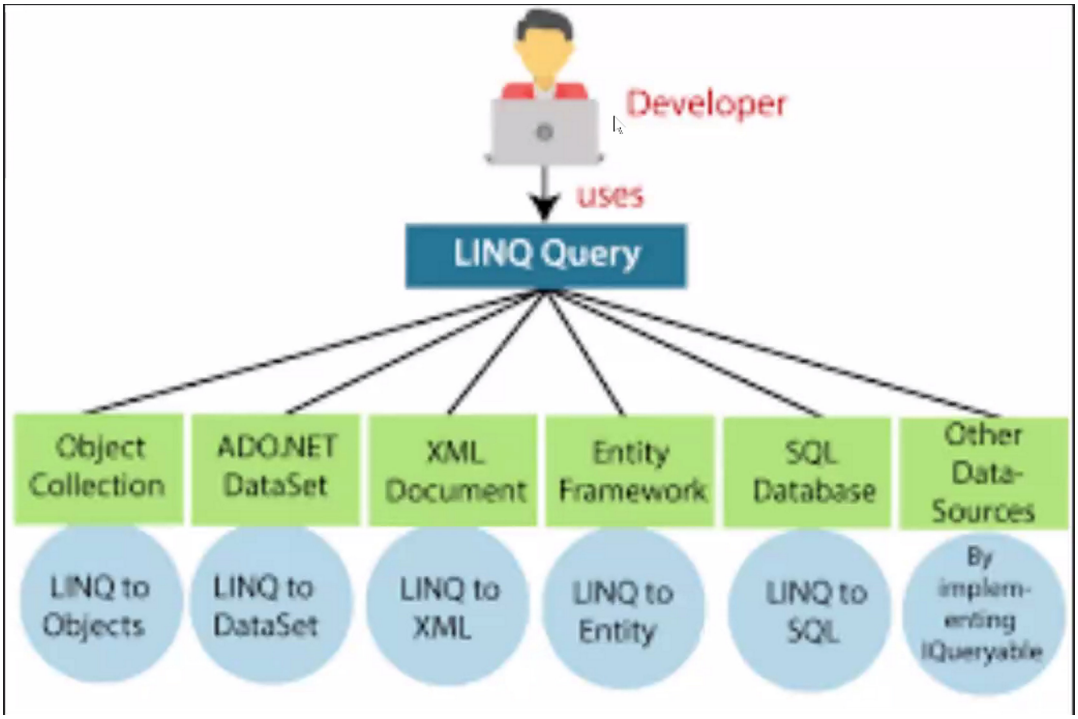
File Edit Format View Help

```
LINQ
Language INtegrated Query

Collections, Database, XML, DataSet
```

LINQ: it give common data set,
Common unified way

LINQ is known as **Language Integrated Query** and it is introduced in *.NET 3.5* and Visual Studio 2008. The beauty of LINQ is it provides the ability to *.NET* languages(like C#, VB.NET, etc.) to generate queries to retrieve data from the data source. For example, a program may get information from the student records or accessing employee records, etc. In, past years, such type of data is stored in a separate database from the application, and you need to learn different types of query language to access such type of data like SQL, XML, etc. And also you cannot create a query using C# language or any other *.NET* language.



Entity Framework => same as Hibernet

Data As a Table

```
public static void AddRecs()
{
    lstDept.Add(new Department { DeptNo = 10, DeptName = "SALES" });
    lstDept.Add(new Department { DeptNo = 20, DeptName = "MKTG" });
    lstDept.Add(new Department { DeptNo = 30, DeptName = "IT" });
    lstDept.Add(new Department { DeptNo = 40, DeptName = "HR" });

    lstEmp.Add(new Employee { EmpNo = 1, Name = "Vikram", Basic = 10000, DeptNo = 10, Gender = "M" });
    lstEmp.Add(new Employee { EmpNo = 2, Name = "Vikas", Basic = 11000, DeptNo = 10, Gender = "M" });
    lstEmp.Add(new Employee { EmpNo = 3, Name = "Abhijit", Basic = 12000, DeptNo = 20, Gender = "M" });
    lstEmp.Add(new Employee { EmpNo = 4, Name = "Mona", Basic = 11000, DeptNo = 20, Gender = "F" });
    lstEmp.Add(new Employee { EmpNo = 5, Name = "Shweta", Basic = 12000, DeptNo = 20, Gender = "F" });
    lstEmp.Add(new Employee { EmpNo = 6, Name = "Sanjay", Basic = 11000, DeptNo = 30, Gender = "M" });
    lstEmp.Add(new Employee { EmpNo = 7, Name = "Arpan", Basic = 10000, DeptNo = 30, Gender = "M" });
    lstEmp.Add(new Employee { EmpNo = 8, Name = "Shraddha", Basic = 11000, DeptNo = 40, Gender = "F" });
}
```

OPERATIONs: Select All

```
static List<Employee> lstEmp = new List<Employee>();
static List<Department> lstDept = new List<Department>();
static void Main1()
{
    AddRecs();
    //from SINGLE_OBJECT in COLLECTION select SOMETHING
    var emps = from emp in lstEmp select emp;
    //IEnumerable<Employee> emps = from emp in lstEmp select emp;
    //List<Employee> emps =(List<Employee>) from emp in lstEmp select emp;

    foreach (Employee emp  in emps)
    {
        Console.WriteLine( emp.Name );
    }

    Console.ReadLine();

}
```

Select Name and Basic:

```
var emps = from emp in lstEmp select new { emp.Name, emp.Basic };
```

Where Condition:

```
var emps = from emp in lstEmp
           where emp.Basic > 10000
           select emp;
```

OrderBy:

```
var emps = from emp in lstEmp
           orderby emp.DeptNo, emp.Name descending
           select emp;
```

Join Two Table

```
var emps = from emp in lstEmp
           join dept in lstDept
                on emp.DeptNo equals dept.DeptNo
           select new { dept.DeptName, emp.Name };
```

Passing Function as a parameter to Select:

Create one Method:

```
static Employee GetEmployees(Employee obj)
{
    return obj;
}
```

Query:

```
var emps = lstEmp.Select(GetEmployees);
```

Passing anonmous method as a parameter to Select:

```
var emps = lstEmp.Select(delegate (Employee obj)
{
    return obj;
});
```

Using Lambda function

```
var emps = lstEmp.Select(emp=>emp);
```

```
//using a lambda instead of anon method
//var emps = lstEmp.Select(emp => emp);
var emps1 = lstEmp.Where(emp => emp.Basic > 11000);
var emps2 = lstEmp.Where(emp => emp.Basic > 11000).Select(emp => emp);
var emps3 = lstEmp.Where(emp => emp.Basic > 11000).Select(emp => emp.Name);
var emps = lstEmp.Where(emp => emp.Basic > 11000).Select(emp => new { emp.Name, emp.Basic });
```

```
var emps4a = lstEmp.Select(emp => emp).Where(emp => emp.Basic > 11000);
var emps4b = lstEmp.Where(emp => emp.Basic > 11000).Select(emp => emp);

var emps5a = lstEmp.Where(emp => emp.Basic > 11000).Select(emp => emp.Name);
//var emps5b = lstEmp.Select(emp => emp.Name).Where(emp => emp.Basic > 11000);
```

OrderBy :

```
//using a lambda instead of anon method
var emps = lstEmp.OrderBy(emp => emp.Name);
var emps2 = lstEmp.OrderByDescending(emp => emp.Name);
```

Join :

```
var emps2a = lstEmp.Join(lstDept, emp => emp.DeptNo, dept => dept.DeptNo, (emp, dept) => emp);
var emps2b = lstEmp.Join(lstDept, emp => emp.DeptNo, dept => dept.DeptNo, (emp, dept) => dept);
var emps2c = lstEmp.Join(lstDept, emp => emp.DeptNo, dept => dept.DeptNo, (emp, dept) => emp.Basic);
var emps2d = lstEmp.Join(lstDept, emp => emp.DeptNo, dept => dept.DeptNo, (emp, dept) => dept.DeptName);
var emps = lstEmp.Join(lstDept, emp => emp.DeptNo, dept => dept.DeptNo, (emp, dept) => new { dept.DeptName, emp.Name });
```

Deferred Execution:

An output will be change according to adding data

```
static void Main11()
{
    AddRecs();
    //deferred execution
    var emps = from emp in lstEmp select emp;
    Console.WriteLine();
    lstEmp.RemoveAt(0);
    foreach (var emp in emps)
    {
        Console.WriteLine(emp.Name + "," + emp.EmpNo);
    }
    Console.WriteLine();
    lstEmp.Add(new Employee { EmpNo = 9, Name = "New", Basic = 11000, DeptNo = 40, Gender = "F" });
    foreach (var emp in emps)
    {
        Console.WriteLine(emp.Name + "," + emp.EmpNo);
    }
    Console.ReadLine();
}
```

Immediate Execution:

An output will not change as per adding data becauase in this case the we are assigning to new object..

```
static void Main()
{
    AddRecs();
    var emps = from emp in lstEmp select emp;
    //immediate execution
    emps = emps.ToList();

    Console.WriteLine();
    lstEmp.RemoveAt(0);
    foreach (var emp in emps)
    {
        Console.WriteLine(emp.Name + "," + emp.EmpNo);
    }
    Console.WriteLine();
    lstEmp.Add(new Employee { EmpNo = 9, Name = "New", Basic = 11000, DeptNo = 40, Gender = "F" });
    foreach (var emp in emps)
    {
        Console.WriteLine(emp.Name + "," + emp.EmpNo);
    }
    Console.ReadLine();
}
```

```
var emps = from emp in lstEmp select emp;  // linq query

var emps = lstEmp.Select(....); // linq method/function

Select is an extension method written for IEnumerable<T>
```

Threading:
Speeding task
Multiple task run at a time
Async Code
- Delegates
- System.Threading
- System.Threading.Task

Asynchronously :

```
//calling a method asynchronously using delObj.BeginInvoke(....
namespace AsyncCodeWithDelegates1
{
    class Program
    {
        static void Main1()
        {
            Console.WriteLine("Before");
            Action o = Display;
            o.BeginInvoke(null, null);
            Console.WriteLine("After");
            Console.ReadLine();
        }
        static void Display()
        {
            System.Threading.Thread.Sleep(5000);
            Console.WriteLine("Display");
        }
    }
}
```

```
//calling a method with parameters asynchronously using delObj.BeginInvoke(....
namespace AsyncCodeWithDelegates2
{
    class Program
    {
        static void Main2()
        {
            Console.WriteLine("Before");
            Action<string> o = Display;
            o.BeginInvoke("aaa", null, null);
            Console.WriteLine("After");
            Console.ReadLine();
        }
        static void Display(string s)
        {
            System.Threading.Thread.Sleep(5000);
            Console.WriteLine("Display" + s);
        }
    }
}
```

```
//calling a method with parameters asynchronously using delObj.BeginInvoke(....
//also using a callback func
namespace AsyncCodeWithDelegates3
{
    class Program
    {
        static void Main1()
        {
            Console.WriteLine("Before");
            Func<string, string> o = Display;
            o.BeginInvoke("aaa", CallBackFunc, null) ;
            Console.WriteLine("After");
            Console.ReadLine();
        }
        static string Display(string s)
        {
            System.Threading.Thread.Sleep(5000);
            Console.WriteLine("Display" + s);
            return s.ToUpper();
        }

        static void CallBackFunc(IAsyncResult ar)
        {
            Console.WriteLine("callback func called");
        }
    }
}
```

```
//calling a method with parameters asynchronously using delObj.BeginInvoke(....
//also using a callback func ( as an anon method - to enable us to access objDel in the callback func)
//get the return value with objDel.EndInvoke
namespace AsyncCodeWithDelegates4
{
    class Program
    {
        static void Main()
        {
            Console.WriteLine("Before");
            Func<string, string> o = Display;
            o.BeginInvoke("aaa", delegate(IAsyncResult ar)
            {
                Console.WriteLine("callback func called");
                string retval = o.EndInvoke(ar);
                Console.WriteLine("retval = " + retval);
            }, null);

            Console.WriteLine("After");
            Console.ReadLine();
        }
```

```
        static string Display(string s)
        {
            System.Threading.Thread.Sleep(5000);
            Console.WriteLine("Display" + s);
            return s.ToUpper();
        }

        static void CallBackFunc(IAsyncResult ar)
        {
            Console.WriteLine("callback func called");

        }
    }
```