lambda

var longWords = words.Where( w ⇒ w.length > 10);

query

var longwords = from w in words where w.length > 10;

ienumerable

int[] scores = new int[] { 97, 92, 81, 60 };

// Define the query expression.

IEnumerable<int> scoreQuery = from score in scores where score > 80 select score;

// Execute the query.

foreach (int i in scoreQuery) {

Console.Write(i + " ");

}

Need for Linq

SqlConnection sqlConnection = new SqlConnection(connectString);

SqlConnection.Open();

System.Data.SqlClient.SqlCommand sqlCommand = new SqlCommand();

sqlCommand.Connection = sqlConnection;

sqlCommand.CommandText = "Select \* from Customer";

return sqlCommand.ExecuteReader (CommandBehavior.CloseConnection)

so,

Northwind db = new Northwind(@"C:\Data\Northwnd.mdf");

var query = from c in db.Customers select c;

Advantages

LINQ offers syntax highlighting that proves helpful to find out mistakes during design time.

LINQ offers IntelliSense which means writing more accurate queries easily.

Writing codes is quite faster in LINQ and thus development time also gets reduced significantly.

LINQ is extensible that means it is possible to use knowledge of LINQ to querying new data source types.

LINQ offers the facility of joining several data sources in a single query as well as breaking complex problems into a set of short queries easy to debug.

Joins

class JoinTables {

class DepartmentClass {

public int DepartmentId { get; set; }

public string Name { get; set; }

}

class EmployeeClass {

public int EmployeeId { get; set; }

public string EmployeeName { get; set; }

public int DepartmentId { get; set; }

}

static void Main(string[] args) {

List <DepartmentClass> departments = new List <DepartmentClass>();

departments.Add(new DepartmentClass { DepartmentId = 1, Name = "Account" });

departments.Add(new DepartmentClass { DepartmentId = 2, Name = "Sales" });

departments.Add(new DepartmentClass { DepartmentId = 3, Name = "Marketing" });

List <EmployeeClass> employees = new List <EmployeeClass>();

employees.Add(new EmployeeClass { DepartmentId = 1, EmployeeId = 1, EmployeeName = "William" });

employees.Add(new EmployeeClass { DepartmentId = 2, EmployeeId = 2, EmployeeName = "Miley" });

employees.Add(new EmployeeClass { DepartmentId = 1, EmployeeId = 3, EmployeeName = "Benjamin" });

var list = (from e in employee join d in departments on e.DepartmentId equals d.DepartmentId select new {

EmployeeName = e.EmployeeName,

DepartmentName = d.Name

});

foreach (var e in list) {

Console.WriteLine("Employee Name = {0} , Department Name = {1}", e.EmployeeName, e.DepartmentName);

}

Substring (select)

List<string> words = new List<string>() { "an", "apple", "a", "day" };

var query = from word in words select word.Substring(0, 1);

Split(select many)

List<string> phrases = new List<string>() { "an apple a day", "the quick brown fox" };

var query = from phrase in phrases

from word in phrase.Split(' ')

select word;

Order by

int[] num = { -20, 12, 6, 10, 0, -3, 1 };

var posNums = from n in num

orderby n

select n;

Order by decending

var posNumsDesc = from n in num

orderby n descending

select n;

Reverse

nav.reverse();

group by :

var groupedList = from employee in employeeList

group employee by employee.DepartmentID into depGroup

orderby depGroup.Key ascending

select depGroup;

foreach (var group in groupedList)

{

Console.WriteLine(string.Format("Dep ID: {0}", group.Key));

foreach (var employee in group)

{

Console.WriteLine(string.Format("\t Employee: {0}", employee.Name));

}

}

Concat

IQueryable<String> custQuery =

(from cust in db.Customers

select cust.Phone)

.Concat

(from cust in db.Customers

select cust.Fax)

.Concat

(from emp in db.Employees

select emp.HomePhone)

;

Delegates with linq

delegate int DelType(int i);

DelType dd = delegate(int value)

{

return (value +2);

};

int i = dd(10);

delegate int DelType(int i);

DelType d = value => value + 2;

int i = d(10);

Example2

public class Program

{

delegate bool IsTeenAger(Student stud);

public static void Main()

{

IsTeenAger isTeenAger = s => s.Age > 12 && s.Age < 20;

Student stud = new Student() { Age = 25 };

Console.WriteLine(isTeenAger(stud));

}

}

public class Student{

public int Id { get; set; }

public string Name { get; set; }

public int Age { get; set; }

}

distinct

products.DistinctBy(x=> x.Code);