ADO.Net ??

Bridge between Front End and Backend

Front End could be console , web application , windows application

Similarly,, backend could be any data source , Database, XML , Collection , Array

Add namespace : using System.Data.SqlClient;

SqlConnection

SqlCommand Selec \* from Employee

Insert into Employee(id, name, address) values()

Com.ExecuteNonQuery()

How do we use parameterized Queries

How do we call stored procedures

Limitation of ADO.Net :

1. The queries are checked at run time they are not checked at compilation time

2. There is no intellisense

3. For multiple data sources , my queries will change

Which means as a developer we have to learn programming language as well as multiple backend language syntax also

The solution for this is LINQ

Language Integrated Query Language

1. We only use one uniform language with different type of data sources

2. It has intellisense.

3. It is checked at compile time

4. Can be used with any data source which implements either IEnumerable interface or IQueryable Interface

Collections implements IEnumerable interface

Lazy Loading : var oddData = (from temp in num

where temp % 2 == 0

select temp

);

It writes Query inside the variable ,. Until we use foreach, values are not retrieved. Query is fired in foreach loop

Eager Loading : When we write the query, we get actual values

We call it by calling any function on the Query

.ToList()

.ToArray()

.Max()

.Sum()

var oddData = (from temp in num

where temp % 2 == 0

select temp

).ToList();

using System;

using System.Collections;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LINQDemo

{

class Program

{

static void Main(string[] args)

{

int[] num = new int[] { 1, 2, 3, 4, 5, 6, 7, 8,9, 10 };

// Sql Query

// Select \* from num

// LINQ

var data = (from temp in num

select temp);

foreach(var temp in data)

{

Console.WriteLine(temp);

}

//Console.WriteLine("Data more than 5");

//// Sql Query

//// Select \* from num where id > 1

//var data1 = (from temp in num

// where temp > 5

// select temp

// );

//foreach (var temp in data1)

//{

// Console.WriteLine(temp);

//}

//Console.WriteLine("Even numbers");

//// Sql Query

//// Select \* from num where id > 1

//var evenData = (from temp in num

// where temp %2 == 0

// select temp

// );

//foreach (var temp in evenData)

//{

// Console.WriteLine(temp);

//}

//Console.WriteLine("Odd numbers");

//// Sql Query

//// Select \* from num where id > 1

//var oddData = (from temp in num

// where temp % 2 == 0

// select temp

// );

//foreach (var temp in oddData)

//{

// Console.WriteLine(temp);

//}

Console.WriteLine("Eager Loading Case");

//Eager Loading

var eagerLoadingData = (from temp in num

select temp).ToList();

foreach (var temp in eagerLoadingData)

{

Console.WriteLine(temp);

}

var maxNumber = (from temp in num

select temp).Max();

Console.WriteLine("Max no is " + maxNumber);

var sum = (from temp in num

select temp).Sum();

}

}

}

Collection

using System;

using System.Collections;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LINQDemo

{

class Program

{

static void Main(string[] args)

{

List<int> num = new List<int>() { 1, 2, 3, 4, 5, 6, 7, 8,9, 10 };

// Sql Query

// Select \* from num

// LINQ

var data = (from temp in num

select temp);

foreach(var temp in data)

{

Console.WriteLine(temp);

}

Console.WriteLine("Data more than 5");

// Sql Query

// Select \* from num where id > 1

var data1 = (from temp in num

where temp > 5

select temp

);

foreach (var temp in data1)

{

Console.WriteLine(temp);

}

Console.WriteLine("Even numbers");

// Sql Query

// Select \* from num where id > 1

var evenData = (from temp in num

where temp % 2 == 0

select temp

);

foreach (var temp in evenData)

{

Console.WriteLine(temp);

}

Console.WriteLine("Odd numbers");

// Sql Query

// Select \* from num where id > 1

var oddData = (from temp in num

where temp % 2 == 0

select temp

);

foreach (var temp in oddData)

{

Console.WriteLine(temp);

}

Console.WriteLine("Eager Loading Case");

//Eager Loading

var eagerLoadingData = (from temp in num

select temp).ToList();

foreach (var temp in eagerLoadingData)

{

Console.WriteLine(temp);

}

var maxNumber = (from temp in num

select temp).Max();

Console.WriteLine("Max no is " + maxNumber);

var sum = (from temp in num

select temp).Sum();

}

}

}

Expression

X + y

Z+3\*5

X=>y , => Lambda operator

Lambda Expression

(input) => (output)

Lambda Expressions are used to write anonymous methods in a short way

List<String> num = new List<String>() { "1", "2", "3", "4", "5", "6", "7", "8","9", "10" };

// Sql Query

// Select \* from num

// LINQ

var data = (from temp in num

select temp);

foreach(var temp in data)

{

Console.WriteLine(temp);

}

var data = (from temp in num

orderby temp descending

select temp);

foreach(var temp in data)

{

Console.WriteLine(temp);

}

Properties : Wrapper around the private variables of a class

Getter setter

Get() is used to return value

Set() is to store value in a variable

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LINQDemo

{

class Student

{

int rn;

public int Rn

{

get { return rn; }

set { rn = value; }

}

string name;

public string Name

{

get { return name; }

set { name = value; }

}

public void Get() { }

public void Show() { }

static void Main()

{

Student student = new Student();

student.Rn = 9;

Console.WriteLine(student.Rn);

}

}

}

// Read only property

public int Rn { get; }

// Validation

int marks;

public int Marks

{

get { return marks; }

set

{

if (value > 100)

marks = 0;

else

marks = value;

}

}

class Student

{

// Read only property

public int Rn { get; }

// Validation

int marks;

public int Marks

{

get { return marks; }

set

{

if (value > 100)

marks = 0;

else

marks = value;

}

}

public Student(int rn)

{

Rn = 10;

}

public void Get() { }

public void Show() { }

static void Main()

{

Student student = new Student(20);

// student.Rn = 9;

Console.WriteLine(student.Rn);

}

}

A variable is declared directly in a class or in a struct.

A property is a flexible mechanism to read write validate compute value of a private field / variable

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LINQDemo

{

class Emp

{

public int id { get; set; }

public string name { get; set; }

public int salary { get; set; }

static void Main()

{

List<Emp> emplist = new List<Emp>();

Emp e1 = new Emp(){ id=1, name="Ajay", salary=12000};

Emp e2 = new Emp() { id = 2, name = "Ajay", salary = 12000 };

emplist.Add(e1);

emplist.Add(e2);

}

}

// Collection Initializer

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LINQDemo

{

class Emp

{

public int id { get; set; }

public string name { get; set; }

public int salary { get; set; }

static void Main()

{

//List<Emp> emplist = new List<Emp>();

//Emp e1 = new Emp() { id = 1, name = "Ajay", salary = 12000 };

//Emp e2 = new Emp() { id = 2, name = "Ajay", salary = 12000 };

//emplist.Add(e1);

//emplist.Add(e2);

//collection initializer

List<Emp> emplist = new List<Emp>()

{

new Emp() { id=1, name="Deepak", salary=12000},

new Emp() { id=1, name="Deepak", salary=12000},

new Emp() { id=1, name="Deepak", salary=12000},

new Emp() { id=1, name="Deepak", salary=12000},

new Emp() { id=1, name="Deepak", salary=12000},

new Emp() { id=1, name="Deepak", salary=12000},

new Emp() { id=1, name="Deepak", salary=12000},

new Emp() { id=1, name="Deepak", salary=12000},

new Emp() { id=1, name="Deepak", salary=12000}

};

}

}

}