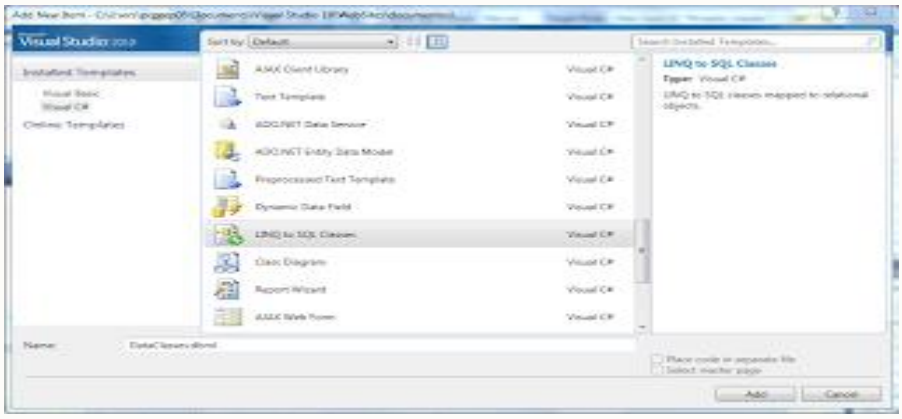
Prof. Ing. Alberto Moreno

Linq la evolución de ADO .net

Una de las grandes novedades de Visual Studio y de .NET Framework 4.5 es el Language Integrated Query (LINQ) que habilita la definición de consultas integradas en el lenguaje de programación y es una de las piedras angulares de las nuevas versiones de C# y VB. Precisamente. No es facil dejar atras nuestros Dataset y nuestros amigos DataAdapters pero es momento de continuar, un consejo para superar la perdida es que imaginen que adonet(dataset,dataadapter,connection) era su carro viejo que funciona y esta perferto sin embargo salio el nuevo de agencia y que esta mejor, corre mas rapido, es mas eficiente, mas bonito y no daña el ambiente.

asi que para ayudarlos a migrarse les doy unos ejemplos con connecion a bases de datos sql server.

Lo primero es la hacer la connecion para lograr esto agreguen un nuevo item en su proyecto y elijan linq.
ejemplo:



ahora ya agregado el dbml tiene que agregar las tablas del server explorer:

bueno ahora ha jugar:
hagamos queries:

Primero la consulta comun:

```
DataClassesDataContext db = new DataClassesDataContext();
var clientes = from c in db.Customers
select new {c.CustomerID, c.CompanyName, c.ContactName, c.Country,c.City };
```

como usar el resultado

usando Bind

```
GridView1.DataSource = clientes;
GridView1.DataBind();
```

o leerlo con un ciclo:

```
foreach (var customer in clientes )
{
Console.WriteLine("Customer {0}: {1}", customer.CustomerId, customer.CompanyName);
}
```

Like

para usar like tiene que hacer un using System.Data.Linq.SqlClient;

```
DataClassesDataContext db = new DataClassesDataContext();
var clientes = from c in db.Customers
where SqlMethods.Like(c.Country, "%" + TextBox1.Text + "%")
select new {c.CustomerID, c.CompanyName, c.ContactName, c.Country,c.City };
```

Concatenar dos campos

```
var query = from c in db.Customers
where c.customerid==txtcodito.text && c.country=="spain"
select new { c.Nombre, nombrecompleto = string.Format("{0} {1}", c.companyname,c.contactname)
};
```

execute scalar ejemplo con function

```
string nombre=(from c in db.customers where p.customerid== txtusuario.text select
c.companyname).Single();

string nombre=(from c in db.customers
 where p.customerid== txtusuario.text select c).Single().companyname;
```

Group by

```
var pais = from p in db.Customers
group p by new { p.Country } into r
select new { r.Key.Country };


var rows = from item in TablaEmpleado.AsEnumerable()
        group item by
              new
                 {
                   empid = item["employeeid"].ToString(),
                   depto = item["DeptoId"].ToString()
                 }
        into g
          select new
                 {
                  EmpresaId = g.Key. empid,
                  depto = g.Key. depto,
                  total = g.Count())
                 };
```

Sum en linq

```
System.Nullable totalFreight =
    (from ord in db.Orders
    select ord.Freight)
    .Sum();


System.Nullable totalUnitsOnOrder =
    (from prod in db.Products
    select (long)prod.UnitsOnOrder)
    .Sum();
```

in en linq

```
string[] countries = new string[] { "Guatemala", "USA", "Mexico" };
var customers =from c in context.Customers
where countries.Contains(c.Country)
select c;
```

comentario:
Eve para que mire que la tomo en cuenta.

Between

```
var query = from p in db.orders
where p.Fecha >= Convert.ToDateTime(TextBox1.Text) && p.Fecha <=
Convert.ToDateTime(TextBox2.Text) select new { p.Fecha };
```

Distinct

```
var query = (from c in db.customers
        orderby c.country
 select c.country).Distinct();


var query = (from c in db.customers
 select c.country).Distinct().OrderByDescending(x=>x.country);
```

Join

```
var groupJoinQuery2 =
from category in categories
join prod in products on category.ID equals prod.CategoryID
orderby category.Name
select new
{
Category = category.Name,
Products = prod.Name
};
```

Prof. Ing. Alberto Moreno

sub consulta

```
var groupJoinQuery2 =
from category in categories
join prod in products on category.ID equals prod.CategoryID into prodGroup
orderby category.Name
select new
{
Category = category.Name,
Products = from prod2 in prodGroup
orderby prod2.Name
select prod2
};
```

Left Outer Joins

```
var query = (from p in dc.GetTable()
join pa in dc.GetTable() on p.Id equals pa.PersonId into tempAddresses
from addresses in tempAddresses.DefaultIfEmpty()
select new { p.FirstName, p.LastName, addresses.State });
```

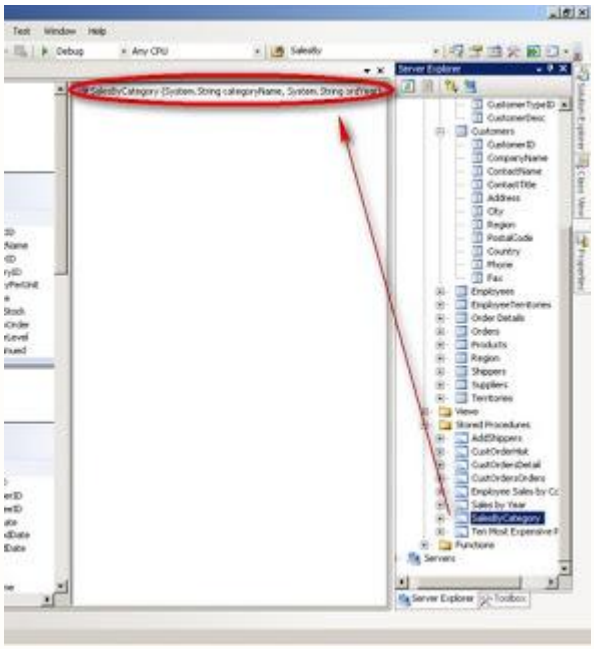
SQL:

```
SELECT [t0].[FirstName], [t0].[LastName], [t1].[State] AS [State]
FROM [dbo].[Person] AS [t0]
LEFT OUTER JOIN [dbo].[PersonAddress] AS [t1] ON [t0].[Id] = [t1].[PersonID]
```

mas ejemplos de join clic aquí

si son de las personas que usan Store Procedure
pues bien es mas facil =)

lo primero que tiene que hacer es bajar el sp de server explorar al contex del linq



codigo:

```
//Obtaining the data source
var dbNorthwind =new NorthwindDataContext() ;

// Create the query
var query = dbNorthwind.SalesByCategory("Beverages","1997");

// Execute the query
foreach (var c in query)
{
Console.WriteLine(c.ProductName + "," + c.TotalPurchase);
}

// si quiere el store procedure solo retorna un valor, esto lo usan para busquedas o totales.

var query = dbNorthwind.ObtenerCategoria(txtcodigo.text).Single();
return query.categoryname;

// otro ejemplo de llamar store procedure con linq
var contactNames =
from result in db.GetCustomersInCity("London")
select result.ContactName;


foreach (string contactName in contactNames)
{
Console.WriteLine(contactName) ;
}
```

Las transacciones tambien son soportadas en linq ejemplo:

```
using(TransactionScope ts = new TransactionScope())
{
db.ExecuteCommand("exec sp_DoSomethingCool");
db.SubmitChanges();
```

```
ts.Complete();
}
```

alternativa:

```
db.LocalTransaction = db.Connection.BeginTransaction();
{
db.SubmitChanges();
db.LocalTransaction.Commit();
db.AcceptChanges();
}
catch {
db.LocalTransaction.Abort();
throw;
}
finally {
db.LocalTransaction = null;
}
```

si quiere manejar la concurecnia en este punto, puede guiarse con el siguiente ejemplo:

```
db.SubmitChanges(ConflictMode.FailOnFirstConflict);
db.SubmitChanges(ConflictMode.ContinueOnConflict);
```

si quieren usar LINQ con Listas

```
var custOrders =
from c in db.Customers
join o in db.Orders on c.CustomerID equals o.CustomerID into orders
where c.CustomerID == "ALFKI"
select new {c.ContactName, orders};

var list = custOrders.ToList() ;

foreach (var listItem in list)
{
Console.WriteLine(listItem.ContactName + " has " + listItem.orders.Count() + " orders, which have
been shipped to:") ;
foreach (Order order in listItem.orders)
{
Console.WriteLine(" Order shipped to - " + order.ShipCountry) ;
}
}
```

Listas genericas

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
```

```
public class MainClass {
public static void Main() {
List customers = GetCustomerList();
var waCustomers =
from c in customers
where c.Region == "R1"
select c;
foreach (var customer in waCustomers) {
Console.WriteLine("Customer {0}: {1}", customer.CustomerId, customer.CompanyName);
foreach (var order in customer.Orders) {
Console.WriteLine(" Order {0}: {1}", order.Id, order.OrderDate);
}
}
}
static List GetCustomerList() {
List empTree = new List();
empTree.Add(new Product { ProductName = "A", Category = "O", UnitPrice = 12, UnitsInStock = 5,
Total = 36, OrderDate = new DateTime(2005, 1, 1), Id = 1 });
empTree.Add(new Product { ProductName = "B", Category = "O", UnitPrice = 2, UnitsInStock = 4,
Total = 35, OrderDate = new DateTime(2005, 1, 1), Id = 1 });
empTree.Add(new Product { ProductName = "C", Category = "O", UnitPrice = 112, UnitsInStock =
3, Total = 34, OrderDate = new DateTime(2005, 1, 1), Id = 1 });
empTree.Add(new Product { ProductName = "D", Category = "O", UnitPrice = 112, UnitsInStock =
0, Total = 33, OrderDate = new DateTime(2005, 1, 1), Id = 1 });
empTree.Add(new Product { ProductName = "E", Category = "O", UnitPrice = 1112, UnitsInStock =
2, Total = 32, OrderDate = new DateTime(2005, 1, 1), Id = 1 });
empTree.Add(new Product { ProductName = "F", Category = "O", UnitPrice = 11112, UnitsInStock
= 0, Total = 31, OrderDate = new DateTime(2005, 1, 1), Id = 1 });

List l = new List();
l.Add(new Customer { CompanyName = "A", Region = "R1", UnitsInStock = 1, Orders = empTree,
CustomerId =0});
l.Add(new Customer { CompanyName = "B", Region = "R2", UnitsInStock = 2, Orders = empTree,
CustomerId = 1 });
l.Add(new Customer { CompanyName = "C", Region = "R3", UnitsInStock = 3, Orders = empTree,
CustomerId = 2 });
l.Add(new Customer { CompanyName = "D", Region = "R4", UnitsInStock = 4, Orders = empTree,
CustomerId = 3 });
l.Add(new Customer { CompanyName = "E", Region = "R5", UnitsInStock = 5, Orders = empTree,
CustomerId = 4 });
return l;
}
}

class Customer : IComparable {
public string CompanyName { get; set; }
public string Region { get; set; }
public List Orders { get; set; }
public int UnitsInStock { get; set; }
public int CustomerId { get; set; }

public override string ToString() {
return String.Format("Id: {0}, Name: {1}, Region: {3}", this.CustomerId, this.CompanyName,
this.Region);
}
int IComparable.CompareTo(Customer other) {
```

```
if (other == null)
return 1;

if (this.CustomerId > other.CustomerId)
return 1;

if (this.CustomerId < other.CustomerId) return -1; return 0; } } class Product : IComparable {
public string ProductName { get; set; }
public string Category { get; set; }
public int UnitPrice { get; set; }
public int UnitsInStock { get; set; }
public int Total { get; set; }
public DateTime OrderDate { get; set; }
public int Id { get; set; }

public override string ToString() {
return String.Format("Id: {0}, Name: {1} , Category: {3}", this.Id, this.ProductName, this.Category);
}
int IComparable.CompareTo(Product other) {
if (other == null)
return 1;
if (this.Id > other.Id)
return 1;

if (this.Id < other.Id) return -1; return 0; }
}http://www.java2s.com/Code/CSharp/LINQ/Useforeachlooptodealwiththeresultfromling.htm
```

Publicado por Anonimo en 11:04 a.m.
Enviar esto por correo electrónicoBlogThis!Compartir en TwitterCompartir en Facebook
Etiquetas: aspnet, linq, linq from sql, linq para sql

2 comentarios:

Luis dijo...

Las operaciones INSERT son mejores en LINQ, pero las operaciones de SELECT son mejores en ADO.

· Reading from a table (ADO vs. LinQ).

http://www.codeproject.com/KB/dotnet/LinQ_Performance_net3_5/Image3.JPG

here is a vas difference between median values of LinQ and ADO as far as reading from table is concerned. ADO wins here , but is because of ADO.net maturity in the market and its tight connection with SQL server or because LinQ (in my opinion) created an overhead by creating the < IEnumerable> interface and an object if each item was draw in the LinQtoSQl dbml designer. For further improvement you should try with loosely typed datasets.

-Filling Dataset using LinQ and ADO and then performing filter operations.

And there is a big difference between mean values. I think that in the LinQ implementation the line where we create an object of DataRow and then add to the table is the place where performance is hitting. ADO implementation wins here.

```
table.LoadDataRow(new Object[] { tempRec.CustomerID, tempRec.TerritoryID,
tempRec.AccountNumber, tempRec.CustomerType, tempRec.rowguid,
tempRec.ModifiedDate}, true);
```

http://www.codeproject.com/KB/dotnet/LinQ_Performance_net3_5/Image6.JPG

En los siguientes los valores más pequeños son los mejores:

ADORead
Mean: 7162452
Median: 7097164

LinQRead
Mean: 14000818
Median: 13825471