Với SQLExpress để truy vấn dữ liệu ta phải dùng đến các câu lệnh - Query khá phức tạp. Hơn nữa để sử dụng trong ứng dụng C# lại thêm 1 tầng phức tạp nữa với các câu lệnh: dùng ConnectionStringkhởi tạo kết nối tới DataBase, tự khai báo các biến để chạy 1 lệnh - command, rồi còn phải tính toán đầu ra của câu lệnh... Thật là quá phức tạp nếu như ta có 1 chương trình "khủng".

Vậy làm sao để giải quyết vấn đề này?

Một đề xuất là sử dụng Linq to SQL.

Không chỉ áp dụng cho truy vấn SQL mà LinQ còn có khả năng hỗ trợ trên nhiều nền tảng khác: XML, SQLite, Excel...

I. LinQ là gì?

Để giảm gánh nặng thao tác trên nhiều ngôn ngữ khác nhau và cải thiện năng suất lập trình, Microsoft đã phát triển giải pháp tích hợp dữ liệu cho .NET Framework có tên gọi là LINQ(Language Integrated Query), đây là thư viện mở rộng cho các ngôn ngữ lập trình C# và Visual Basic.NET (có thể mở rộng cho các ngôn ngữ khác) cung cấp khả năng truy vấn trực tiếp dữ liệu Object, CSDL và XML.

LINQ là một tập hợp các thành phần mở rộng cho phép viết các câu truy vấn dữ liệu ngay trong một ngôn ngữ lập trình, như C# hoặc VB.NET. Khi tạo một đối tượng LINQ thì Visual Studio sẽ tự động sinh ra các lớp có các thành phần tương ứng với CSDL của chúng ta. Khi muốn truy vấn, làm việc với CSDL ta chỉ việc gọi và truy xuất các hàm, thủ tục tương ứng của LINQ mà không cần quan tâm đến các câu lệnh SQL thông thường.

Tóm lại LINQ ra đời để giảm công sức cho những quá trình đơn giản và “chung chung” trước đây.

Điểm mạnh (chưa chắc về độ mạnh, nhưng hay) của LINQ là “viết truy vấn cho rất nhiều các đối tượng dữ liệu”. Từ CSDL, XML Data Object … thậm chí là viết truy vấn cho một biến mảng đã tạo ra trước đó. Vì vậy mới có các khái niệm LinQ to SQL, LinQ to XML, blo bla ….

Tuy nhiên so với mô hình Entity (Entity Framework), LINQ có yếu điểm là chậm và thiếu nhất quán (hiện đại tất phải hại điện).

LINQ có từ bản .NET 3.5, vậy nên tối thiểu chương trình của bạn phải chạy trên nền tảng này.

Visual Studio 2008, hoặc các phiên bản Express của nó là các bộ công cụ phát triển tiêu biểu cho ứng dụng dùng LINQ.

II. Sử dụng LinQ

|  |
| --- |
| [LinQ là gì và tại sao nên dùng LinQ](http://4.bp.blogspot.com/-b4aX5qvEGew/UsfFthNUYnI/AAAAAAAAClQ/MHiEQPoYRf8/s320/LinqToSql.gif) |
| Câu lệnh SELECT trong SQL được thực hiện bởi LINQ |

Trong phần này tôi sẽ sử dụng 1 project demo để các bạn thấy được cách làm việc với LINQ như thế nào?!

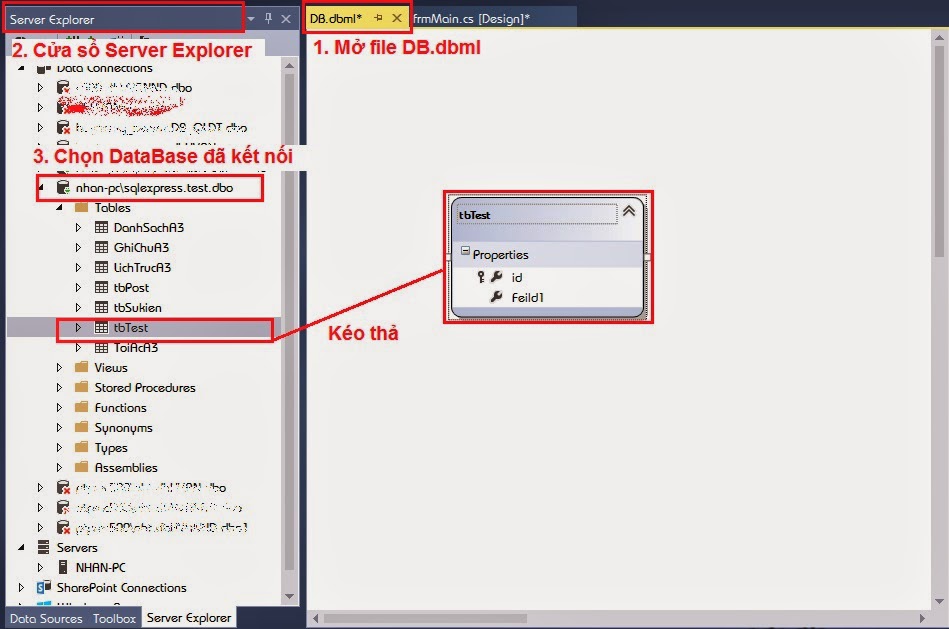
CSDL là 1 bảng tbTest với 2 trường: id (kiểu int, tự động tăng - Identity) và Feild1 (kiểu nvarchar(50)).

1. Khởi tạo đối tượng LinQ to SQL

Trong project chọn Add > Data > LinQ to SQL classes

Ở đây tôi tạo 1 file DB.dbml trong thư mục DB để dễ quản lý.

Sau khi có được file ta tiến hành kéo các table cần thiết vào để tự động sinh các thủ tục LinQ

[](http://4.bp.blogspot.com/-lML9WtOaNc8/U1O4NWB1CzI/AAAAAAAABTI/e3uERAZk76A/s1600/Ket+noi+LINQ+voi+DataBase+table+nvnhan2910.jpg)

Vậy là xong, ta đã tạo được các thủ tục cần thiết để làm việc với LINQ. Easy??!!   :)

2. Truy vấn dữ liệu

Tương tự như Entity ta phải khai báo 1 biến DataContext để tương tác với LinQ

|  |  |
| --- | --- |
| 1 | DB.DBDataContext db = new DB.DBDataContext(); |
| 1 |  |

Có 2 kiểu truy vấn dữ liệu:  
Query Syntax

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | var abc = from p in db.tbTests            where p.id > 10            select p;    // Hoặc chọn một số trường  var abc = from p in db.tbTests            where p.id > 10            select new            {                p.id,                p.Feild1            }; |

Method Syntax

|  |  |  |
| --- | --- | --- |
| 1 | var xyz = db.tbTests.Where(p => p.id > 10).Select(p => new { p.id, p.Feild1 }); | |
| 1 |  |

Câu truy vấn SQL tương ứng

|  |  |
| --- | --- |
| 1  2  3  4 | SELECT [id]        ,[Feild1]    FROM [dbo].[tbTest]    WHERE [id] > 10 |

Chọn từ nhiều bảng theo kiểu Inner Join (lấy những bản ghi có điều kiện thỏa mãn)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29 | // Query Syntax  var result = from p in products               join c in categories on p.CategoryID equals c. CategoryID               select new               {                   ProductName=p.ProductName,                   CategoryName=c.CategoryName               };    // Hoặc chọn từ 2 bảng bằng cách from 2 lần  :)  var result = from p in products               from c in categories               where p.ProductName equals c.CategoryName               select new               {                   ProductName=p.ProductName,                   CategoryName=c.CategoryName               };    // Method Syntax  var result = products.Join(               categories,               p=>p.CategoryID,               c=>c.CategoryID,               (p,c) => new               {                   ProductName=p.ProductName,                   CategoryName=c.CategoryName               }); |

Cú pháp khác tương tự các câu lệnh trong SQL.  
Các biến ở trên trả về có kiểu IEnumrable, 1 loại kiểu dữ liệu giống như List. Các thao tác cơ bản với loại biến này:

xyz.FirstOrDefault(): Chọn bản ghi đầu tiên hoặc mặc định

xyz.Skip(5): Nhảy qua n bản ghi

xyz.Take(5): Lấy n bản ghi đầu tiên

xyz.ToList(): Chuyển sang kiểu List

xyz.Count(): đếm số bản ghi

xyz.Select(...), xyz.Where(...), xyz.Join(...): Các câu lệnh truy vấn theo kiểu Method Syntax

bla bla

Đôi khi bạn không tìm được cú pháp thích hợp hoặc LinQ không hỗ trợ loại truy vấn mà vốn có trong SQL thì bạn có thể thực hiện trực tiếp câu lệnh đó thông qua LinQ:

|  |  |
| --- | --- |
| 1 | var result = db.ExecuteQuery<int>("SELECT NEXT VALUE FOR seq\_tbCanBo") |

3. Thêm, sửa, xóa dữ liệu thông qua LinQ

Thêm dữ liệu

|  |  |  |
| --- | --- | --- |
| 1  2  3  4 | DB.tbTest a = new DB.tbTest();      // Khai báo đối tượng mới  a.Feild1 = txtFeild1.Text;      // id là giá trị tự động tăng > ko cần thay đổi  db.tbTests.InsertOnSubmit(a);   // Thêm đối tượng a vào  db.SubmitChanges();     // Lưu thay đổi | |
| 1 |  |

Sửa dữ liệu

|  |  |
| --- | --- |
| 1  2  3  4  5 | var a = (from p in db.tbTests           where p.id == int.Parse(txtId.Text)           select p).FirstOrDefault();  a.Feild1 = txtFeild1.Text;  db.SubmitChanges(); |
| 1 |  |

Xóa dữ liệu

|  |  |
| --- | --- |
| 1  2  3  4  5 | var a = (from p in db.tbTests           where p.id == int.Parse(txtId.Text)           select p).FirstOrDefault();  db.tbTests.DeleteOnSubmit(a);  db.SubmitChanges(); |

III. Bindding

Bindding là cách thuận tiện để thay đổi, cập nhật, thêm mới bản ghi ở CSDL một cách dễ dàng trực quan, sử dụng LinQ.  
Giả sử ta thực hiện Bindding ở 1 bảng, có 2 chế độ (kiểu hiển thị) là GridView và Detail:

Chế độ GridView: có 1 gridview để hiển thị các bản ghi trong bảng, tương tự như khi chúng ta chạy câu truy vấn SELECT \* trong SQL

Detail: các Control cho phép thay đổi giá trị của bản ghi đang được chọn ở Gridview. Khi click vào 1 hàng ở Gridview thì dữ liệu của hàng đó sẽ được điền tự động vào các Control này.

Cách dùng Bindding:

Trong chế độ Design form, mở cửa sổ Data Sources, ở đây các bảng trong file DB.dbml (được tạo khi khởi tạo LinQ) sẽ được hiển thị, giúp ta có thể kéo vào trong form của mình

Với mỗi bảng có các tùy chọn để chọn chế độ xem: Gridview, Detail,...

|  |
| --- |
| [LinQ là gì và tại sao nên dùng LinQ?](http://4.bp.blogspot.com/-l69cmzq7V_M/U1PcjZWYFzI/AAAAAAAABTg/1OgpNyhvhZU/s1600/Tao+Bindding+nvnhan2910.png) |
| Tạo Bindding bằng cách kéo thả |

Sau khi kéo như vậy thì nó sinh ra 1 đối tượng là tbTestBinddingSource, đây chính là đối tượng tương tác với SQL, LinQ.

Quy định nguồn dữ liệu cho Bindding:

|  |  |
| --- | --- |
| 1  2  3  4  5 | // DataSource là 1 bảng có sẵn  tbTestBindingSource.DataSource = db.tbTests;    // Datasource là một đối tượng IEnumrable lấy từ truy vấn LINQ  tbTestBindingSource.DataSource = result; |
| 1 |  |

Bạn có thể kéo 1 BinddingNavigator để thực hiện các thao tác thêm, xóa dễ dàng

Khi đã có Bindding như thế này rồi thì việc thêm sửa xóa cực kỳ đơn giản, không cần code nhiều, sau khi thay đổi trên gridview, detail để lưu tất cả các thay đổi chỉ cần

|  |  |
| --- | --- |
| 1  2 | tbTestBindingSource.EndEdit();  db.SubmitChanges(); |

Các thao tác có thể làm với BinddingSource

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | tbTestBindingSource.RemoveCurrent();    // Xóa hàng hiện tại    // Di chuyển trên gridview  tbTestBindingSource.MoveFirst();  tbTestBindingSource.MoveLast();  tbTestBindingSource.MoveNext();  tbTestBindingSource.MovePrevious();    tbTestBindingSource.Position = 5;   // Xác định vị trí cho Bindding |

Sau số thao tác thay đổi trên grid view, bạn muốn xem số hàng thêm mới, sửa xóa??

|  |  |
| --- | --- |
| 1  2  3 | int insert = db.GetChangeSet().Inserts.Count;  int update = db.GetChangeSet().Updates.Count;  int delete = db.GetChangeSet().Deletes.Count; |

[Linq] Giới thiệu

[](http://3.bp.blogspot.com/-nNo613uMclM/UsfBLV9GhDI/AAAAAAAAClE/8UMg5og3OUY/s1600/Linq.jpg)

thuvienwinform - Linq(Language Integrated Query - Ngôn ngữ tích hợp truy vấn) là thư viện mở rộng cho các ngôn ngữ lập trình C#, VB,...được Microsoft tích hợp vào .NET Framework (từ phiên bản 3.5).  
Linq cho phép chúng ta làm việc với các tập dữ liệu như CSDL, XML, HTML,...qua đó người ra ta phân loại Linq như sau:

 LINQ to Objects: làm việc với dữ liệu trên bộ nhớ (thường là các danh sách kế thừa interface IEnumerable) sử dụng cú pháp ngôn ngữ LINQ.

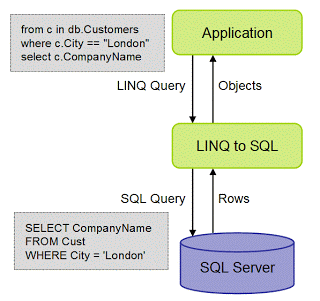
 LINQ to XML: làm việc với dữ liệu XML sử dụng cú pháp ngôn ngữ LINQ.

 LINQ to Entities: làm việc với Entity Framework.

 LINQ to SQL: Làm việc với SQL Server.

 LINQ to Datasets: Một kỹ thuật cho phép sử dụng LINQ trong một ứng dụng dựa trên DataSet mà không cần viết lại sử dụng LINQ to SQL.

 PLINQ: Hỗ trợ sử dụng LINQ trong lập trình song song.

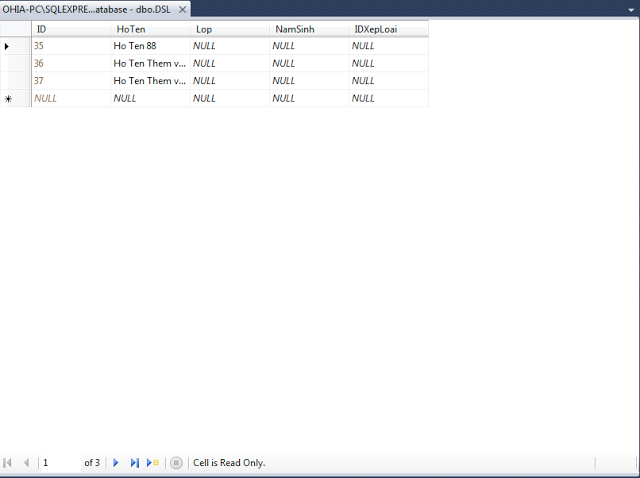
[](http://4.bp.blogspot.com/-b4aX5qvEGew/UsfFthNUYnI/AAAAAAAAClQ/MHiEQPoYRf8/s1600/LinqToSql.gif)

Tuy nhiên, trong các bài tiếp theo trong nhãn này mình chỉ nói đến LINQ to SQL!  
Như chúng ta đã biết để thao tác với CSDL cũng có thể dùng ADO.NET. Vậy chúng khác nhau như nào?  
- Đầu tiên Linq cho phép tạo câu truy vấn ngay trong ngôn ngữ lập trình. Với ADO.NET thì chúng ta phải viết câu truy vấn như một string vậy, điều đó rất khó kiểm lỗi (phải chạy thử mới biết được lỗi)  
- Khi thao tác với CSDL (Thêm, sửa, xóa,...) thì Linq đã tự tối ưu cho chúng ta để đảm bảo tốc độ cao nhất. Đơn cử như việc select, trong ADO.NET thì ta sử dụng "select \*..." sau đó lọc ra những trường cần dùng, còn Linq thì chỉ select những trường cần chọn mà thôi nên nó sẽ tối ưu hơn. Hay nói cách khác thì Linq đã tối ưu sẵn cho ta rồi!  
- Về yêu cầu thì ADO.NET chỉ cần phiên bản .NET 2.0, còn Linq thì yêu cầu phiên bản 3.5  
Vậy Linq có phải được tạo ra để thay thế ADO.NET? Không phải, bạn có thể làm việc với Linq mà không dính đến CSDL. Linq to Sql là một phần mở rộng của Linq.

[Linq to Sql] Load và Thêm dữ liệu

[](http://2.bp.blogspot.com/-nNo613uMclM/UsfBLV9GhDI/AAAAAAAAClI/fkdcyfd0T48/s1600/Linq.jpg)

thuvienwinfom - Để thêm dữ liệu vào CSDL bằng Linq ta cần khai báo thư viện System.Linq (using System.Linq). Thường thì khi ta tạo project mới thì nó đã thêm sẵn dòng này cho ta rồi! OK bây giờ ta sẽ thực hiện việc thêm dữ liệu bằng Linq, 1 nốt nhạc là xong :v  
Tải demo: Bài 4  
Chuẩn bị: (vì mấy cái này cần cho mấy bài sau nên mọi người mới tập sử dụng Linq nên làm theo y hệt sẽ dễ hiểu hơn)  
I. Tạo môi trường luyện tập  
1. CSDL  
+ Tên bảng: DSL (danh sách lớp)

[](http://3.bp.blogspot.com/-r2nQPjpVET0/UsfK3Aj94SI/AAAAAAAAClg/z8e_FnjOxFE/s1600/CSDLLinq-thuvienwinform.png)

+ Form mẫu. Các biến như trong hình

+ Add item LINQ to SQL cho form này:  
B1: project -> add new item (CS + A)

B2: chọn LINQ to SQL và đặt tên cho nó là: LinqToSql -> ok

B3: Kéo bảng DSL đã tạo vào -> Ctrl + S

Tạo 1 biến db như sau:

LinqToSqlDataContext db = new LinqToSqlDataContext();

OK vậy là đã tạo được môi trường luyện tập...

II. Load dữ liệu từ CSDL ra datagridView bằng LINQ to SQL

Để tiện sử dụng lần sau ta tạo sẵn một thủ tục Refresh như sau:

private void Refresh()

{

dgvNoiDung.Rows.Clear();

var list = from p in db.DSLs

where p.HoTen == p.HoTen//Chọn toàn bộ bảng

select p;

dgvNoiDung.DataSource = list;

}

Giải thích:

var list: khai báo 1 biến chưa có kiểu

form p in db.DSLs

where p.HoTen == p.HoTen

select p; :chọn toàn bộ bảng DSL

OK cho Refresh vào form load là xong viêc load dữ liệu:

private void Form1\_Load(object sender, EventArgs e)

{

//dgvNoiDung.DataSource = db.DSLs;//Cách 1

Refresh();//Cách 2

}

III. Thêm dữ liệu vào CSDL

- Nếu ai đã từng sử dụng ADO.NET thì sẽ cảm thấy nó hơi ngược một chút. Trong ADO.NET thì chúng ta select -> frrom -> where còn trong LINQ to SQL thì from -> where -> select.

- Để thêm 1 họ tên vào bảng DSL ta code như sau:

private void btThem\_Click(object sender, EventArgs e)//Sự kiện click btThem

{

//Thêm mới

DSL thanhVien = new DSL();//Tạo 1 thành viên mới

thanhVien.HoTen = txtHoTen.Text;//Gán giá trị HoTen của thanhvien

db.DSLs.InsertOnSubmit(thanhVien);//Thêm thanhvien này vào csdl

db.SubmitChanges();//Xác nhận thay đổi

Refresh();//Làm mới lại datagridView

}

Giải thích:  
db.DSLs.InsertOnSubmit: thêm một thành viên vào bảng DSL  
db.SubmitChanges: xác nhận thay đổi trong bảng. Nếu không có dòng này thì dữ liệu sẽ không được thêm vào CSDL

[Linq to Sql] Sửa dữ liệu

[](http://2.bp.blogspot.com/-nNo613uMclM/UsfBLV9GhDI/AAAAAAAAClI/fkdcyfd0T48/s1600/Linq.jpg)

thuvienwinform - Sau[bài 1: Load và thêm dữ liệu](http://www.thuvienwinform.blogspot.com/2014/01/linq-to-sql-bai-1-load-va-them-du-lieu.html) thì trong bài này mình sẽ giới thiệu code để sửa dữ liệu trong CSDL bằng LINQ to SQL. Các nguyên liệu như CSDL, form vẫn dùng ở bài trước. OK code khá đơn giản thui:

Tải Demo: bài 4

private void btSua\_Click(object sender, EventArgs e)//Sửa

{

//Sửa

//Xác định thanh viên cần sửa

var thanhVien = (from tv in db.DSLs

where tv.HoTen == dgvNoiDung.CurrentRow.Cells["HoTen"].Value.ToString()//Chọn thành viên nào có tên giống tên nhập vào

select tv

).SingleOrDefault();

thanhVien.HoTen = txtHoTen.Text;

db.SubmitChanges();

Refresh();

}

Giải thích:  
SingleOrDefault(): chọn 1 đối tượng mà thôi  
Ở đâu mình truy vấn thành viên có tên giống với tên trong txtHoTen sau đó tiến hành gán cho nó tên mới thông qua txtHoTen luôn

[Linq to Sql] Xóa dữ liệu

[](http://2.bp.blogspot.com/-nNo613uMclM/UsfBLV9GhDI/AAAAAAAAClI/fkdcyfd0T48/s1600/Linq.jpg)

thuvienwinform - Sau bài 2: Sửa dữ liệu hôm nay mình sẽ giới thiệu với mọi người cách xóa một họ tên trong bảng DSL. Cũng tương tự như sửa vậy, đầu tiên cần xác định xem cái nào cần xóa sau đó ta xóa thành viên được chọn là xong. Ở đây việc chọn mình xác định bằng cách là dùng sự kiện CellClick của datagridView để xác định thành viên đang chọn.  
  
Tải demo: bài 4  
  
Code:

private void dgvNoiDung\_CellClick(object sender, DataGridViewCellEventArgs e)

{

txtHoTen.Text = dgvNoiDung.CurrentRow.Cells["HoTen"].Value.ToString();

}

private void btXoa\_Click(object sender, EventArgs e)//Xóa

{

//Xóa

var thanhVien = db.DSLs.SingleOrDefault(tv => tv.HoTen == txtHoTen.Text);//Xác định thành viên cần xóa

db.DSLs.DeleteOnSubmit(thanhVien);//Xóa thanh viên đã chọn

db.SubmitChanges();//Xác nhận thay đổi

Refresh();//làm mới lại datagridView

}

[Linq to Sql] Tim kiếm

[](http://2.bp.blogspot.com/-nNo613uMclM/UsfBLV9GhDI/AAAAAAAAClI/fkdcyfd0T48/s1600/Linq.jpg)

thuvienwinform - Trong bài này mình sẽ hướng dẫn các bạn cách truy vấn để tìm kiếm bằng Linq to Sql. Cũng tương tự như sửa, xóa đầu tiên cần xác định đối tượng tìm và lưu vào một biến, rồi chọn Datasource của datagridView là biến đó là OK  
  
Code:

private void btTimKiem\_Click(object sender, EventArgs e)//Tìm kiếm

{

//Tìm Kiếm

var listThanhVien = from ltv in db.DSLs

where ltv.HoTen.Contains(txtHoTen.Text)//Chọn những thành viên nào có tên gần giống với txtHoTen.Text

select ltv;

dgvNoiDung.DataSource = listThanhVien;

dgvNoiDung.Refresh();

}

Một số hàm hữu ích trong LINQ to SQL

[](http://2.bp.blogspot.com/-nNo613uMclM/UsfBLV9GhDI/AAAAAAAAClI/fkdcyfd0T48/s1600/Linq.jpg)

thuvienwinform - LINQ to SQL có rất nhiều hàm, thủ tục hỗ trợ chúng ta trong việc lọc dữ liệu chọn dữ liệu và tất nhiên là nó đã được tối ưu sẵn, cách sử dụng cũng rất đơn giản, dễ ràng trong việc sử dụng, hôm nay mình xin được giới thiệu với mọi người một số hàm, thủ tục chọn dữ liệu  
  
1. Chọn dữ liệu từ 2 bảng:  
- Để chọn dữ liệu từ 2 bảng ta sẽ dùng lệnh from như sau:

var canBos = from cb in db.tbCanBos

from phong in db.tbPhongs

where (cb.IdPhong == phong.IdPhong) && (phong.TenPhong == "Khoa Ngữ")

select new { hotenthat = cb.Hoten, quequan = cb.QueQuan, cb.HeSoLuong, phong.TenPhong };

- Chú ý: trong select ta phải dùng new, tức là tạo một bảng mới gồm các thành viện được chọn.  
  
2. Sắp xếp tăng dần theo một trường  
- Cách 1: Sắp xếp ngay trong câu lệnh truy vấn:

var canBos = from cb in db.tbCanBos

from phong in db.tbPhongs

where (cb.IdPhong == phong.IdPhong) && (phong.TenPhong == "Khoa Ngữ")

orderby cb.HeSoLuong descending //Sap xep theo hesoluong

select new { hotenthat = cb.Hoten, quequan = cb.QueQuan, cb.HeSoLuong, phong.TenPhong };

Câu truy vấn này sẽ chọn ra những cán bộ thuộc khoa ngoại ngữ và sắp xếp tăng dần theo hesoluong  
  
- Cách 2: Truy vấn xong mới sắp xếp (cách này hay hơn)

var canBos = from cb in db.tbCanBos

from phong in db.tbPhongs

where (cb.IdPhong == phong.IdPhong) && (phong.TenPhong == "Khoa Ngữ")

select new { hotenthat = cb.Hoten, quequan = cb.QueQuan, cb.HeSoLuong, phong.TenPhong };

canBos = canBos.OrderByDescending(p => p.HeSoLuong);

3. Lấy số dữ liệu nhất định  
- Như ví dụ lấy số cán bộ ở trên, ta chỉ cẩn lấy ra 10 đầu tiên trong kết quả thu được thôi, câu lệnh sẽ là:

canBos = canBos.Take(10);

Hoặc 

canBos = canBos.Skip(10);

- Với cách dùng Skip() thì ta sẽ bỏ qua 10 người ở đầu danh sách và lấy những người từ vị trí 11 trong danh sách thu được  
  
4. Đếm số dữ liệu thu được sau khi truy vấn

int soNguoi = canBos.Count(10);

Đăng bởi : [Hoài Nông](https://plus.google.com/118362489316588664368)

Nguồn thuvienưinform

[Posted by: Thủy Trần Thanh](http://expressmagazine.net/development/1798/linq-la-gi-va-tai-sao)

[LINQ, là gì và tại sao ?](http://expressmagazine.net/development/1798/linq-la-gi-va-tai-sao)

LINQ ra đời bởi vì cách viết Entity trước đây tạo ra quá là nhiều code nhưng lại có thể dùng auto coding (xài Code Smith chẳng hạn). Có thể ở background, MS có thuê mấy thèng viết Code Smith làm LINKQ cũng nên ^^.

Tóm lại LINQ ra đời để giảm công sức cho những quá trình đơn giản và “chung chung” trước đây.

Đó là nỗ lực kết xuất truy vấn ra Entity Object. Lập trình viên sẽ không phải care đến việc viết các Entities nữa.

Điểm mạnh (chưa chắc về độ mạnh, nhưng hay) của LINQ là “viết truy vấn cho rất nhiều các đối tượng dữ liệu”. Từ CSDL, XML Data Object … thậm chí là viết truy vấn cho một biến mảng đã tạo ra trước đó. Vì vậy mới có các khái niệm LinQ to SQL, LinQ to XML, blo bla ….

Thích nhất là cái “truy vấn cho một biến mảng” (mình xin không được comments nhiều về biến mảng, vì mình không phải là dân .NET, trong khi hình như là C#, VB.NET có một vài kiểu định nghĩa List, Generics Collection gì gì đó. Trường hợp bạn muốn sắp xếp lại mảng đó, bạn có thể sử dụng LINQ thay vì viết vòng lặp FOR/WHILE như trước.

Tuy nhiên so với mô hình Entity (Entity Framework), LINQ có yếu điểm là chậm và thiếu nhất quán (hiện đại tất phải hại điện).

Vì thế, với các ứng dụng cỡ bự, yêu cầu sự ổn định, khả năng scalable cao, tạm thời không nên sử dụng LINQ, hoặc chỉ sử dụng LINQ cho các thành phần cơ bản, ít liên quan đến Business Logic.

Trong các hệ thống lớn, khi bạn cần thay đổi một công thức nào đó chẳng hạn, bạn nên sử dụng một “công thức ngoài mã nguồn”. (Stored Procedure chẳng hạn). Khi đó bạn sẽ không phải sửa code, re-compile lại hệ thống. Khi sử dụng LINQ, chắc chắn bạn sẽ tốn thời gian cho công việc này (viết lại LINQ query, re-compile, re-deploy… rõ ràng là cực kì phiền phức).