

هديتي المتواضعة لكم هي عن القادم الجديد (نسبيا) LINQ رغم انه توجد مواضيع عديدة غنية عن هذه التقنية فلامانع من التنوع. واحب ان اذكركم بأني مجرد هاو للبرمجة تتواضع معرفتي ومعلوماتي امام الاساتذة والمحترفين و الاعضاء في هذا المنتدى فإن اخطأت أرجو منكم أن تصححوا خطأي وسأكون لكم ممنونا. فلا أريد أن أعرض معلومات خاطئة لأخوتي العرب. ويترتب على كوني هاو ايضا فترات من الانقطاع المفاجئ نتيجة ظروف عملي .

سأضع سلسلة من المقالات بلغة سهلة مبسطة عن تقنية LINQ to SQL وانا انشرها بالتزامن في منتدى اخر صديق من اجل ان يعم النفع .. والمقالات تحتوي ارشادات عملية لبناء برنامج لتطبيق LINQ على ارض الواقع وقد اخترت التعامل مع قواعد بيانات SQL Server Compact لأن ادوات انشائها وتطويرها مضمنة ومتوفرة لكل مستخدم للفيجوال بيزيك ٢٠٠٨ بالإضافة لدعمها لتقنية LINQ مباشرة . وما ينطبق عليها ينطبق على نسختي Enterprise و Express .

نبدأ باسم الله ...

مقدمة :

مقدمة عن LINQ

ما هي LINQ ؟

باختصار LINQ هي مجموعة تعليمات جديدة للتعامل مع البيانات بكافة اشكالها وايضا كانت (List,DataSet,DataBase,Array...etc) وهي شديدة الشبه ب SQL التي نعرفها جيدا ولكن الفرق أن SQL تتعامل مع كائنات قاعدة البيانات كالجداول مثلا مباشرة بينما LINQ تتعامل مع كائنات فيجوال بيزيك (كالمجموعات Collections و المصفوفات Arrays) بعبارة أخرى : عبارات SQL تترجم وبالتالي يتم تدقيقها على مستوى قاعدة البيانات (Engine) ... اما عبارات LINQ فهي تترجم وتنفذ على مستوى Visual Basic . وهذا احد جوانب قوة LINQ .

من DAO إلى LINQ :

وفي مسيرة تطور التعامل مع البيانات فإن LINQ الان في قمة الهرم . فقد بدأنا قديما مع DAO المملة والبطيئة .. ثم بهرنا ADO بكائناتها المتخصصة وقدرتها على التعامل مع معظم مصادر البيانات بنفس الكائنات مستفيدة من OLE DB .. ثم ADO.NET التي تبنت منصة .NET. للتطوير ودعم OOP واخيرا تم تغليف ADO.NET بالمزود LINQ to DataSet و اتاحة LINQ to SQL للتعامل المباشر مع قواعد بيانات SQL Server وتوسيع LINQ لتشمل كافة انواع البيانات (objects و XML و SQL) كما سنرى فهي البديل الأمثل للتعامل مع البيانات.

وكما سبق ان حدث عندما قدمت لنا مايكروسوفت ADO ظل كثير من المبرمجين متمسكين ب DAO و RDO ..سنجد أن كثيرا من المبرمجين لازالوا يحجمون عن LINQ ويفضلون ADO.net ولاضير في هذا فانا رغم حماسي الشديد ل LINQ لا ازال الجأ احيانا لكائنات ADO. ولكن اؤكد لكم LINQ ستفرض نفسها بقوة.

لماذا اترك ما تعلمته وجربته واتجه لما يسمى LINQ ؟

حسنا اذا لم تكن من مستخدمي LINQ فانك بحاجة الى تعلم لغتي برمجة !!! ...نعم الاولى هي visual basic.net والثانية هي SQL ... يجب أن تعرف قواعد الكتابة كل منهما و اخطاء كل منهما وعند حدوث خطأ منطقي (نتيجة غير متوقعة) سترجع اولا برنامجك في الفيجوال ثم ستذهب لتجربة عدة عبارات SQL في قاعدة بياناتك و تعود لتنسخ عبارة SQL المناسبة وتضعها في برنامجك لتكتشف أن هناك خطأ في كتابة اسم حقل فتعود لترى اسم الحقل الصحيح وتصححه ثم تكتشف ان جملة SQL لا تلبي احتياجك وتضطر لمراجعة ملفات المساعدة للبحث عن دالة SQL تقوم بالعمل وتتمنى لو أن كل العمل في نفس الصفحة حيث لاتضطر للمغادرة والتقصي والبحث من اجل اسم جدول او عمود أو دالة ستتمنى لو ان كل شيء امامك كما اعتدت في بيئة الفيجوال ليقوم المحرر بتقديم الكائنات لك لتنفذ منها كما اعتدنا في الفيجوال او ما يسمى بخاصية IntelliSense ولكن الواقع غير ذلك ... يجب ان تحفظ قاعدة بياناتك وجدولها و اعمدتها وانواع بيانات اعمدتها عن ظهر

قلب وعلاقتها ... وأآآآه من العلاقات فانت بحاجة لالف سطر من INNER JOIN في SQL لتمثيل العلاقات .. وإذا كنت تتعامل مع أكثر من نوع قواعد بيانات فستكتب على الكتب والانترنت تتعلم ميزات لغة SQL لهذا النوع ... ماذا بعد .. هل هذه حالك؟؟؟

طبعاً لأ..... فانت من النوع الآخر الذي يستخدم أدوات الفيجوال التي تعلم الكسل والمسماة "السحرة Wizards" - والسحرة دائماً خبياء - فانت لا تتعب كثيراً والمعالج يقوم بكل شيء فانت مستعد لتتحدثني انه يمكنك كتابة برنامج كامل للتعامل مع قاعدة البيانات بدون أي سطر كود وبدون LINQ ووجه راسها ... !!! نعم اصدقك ولكن جرب ان تنقل برنامجك لجهاز آخر .. أو ان تكتب برنامجاً حقيقياً وليس تعليمياً أو تجريبياً .. اكاد اجزم انك ستحتاج للتعامل للبيانات من خلال الكود .. وستصبح من اصحاب النوع الأول .

البك الحل ... استخدم LINQ ... سهلة بسيطة سريعة وقوية ... وستبقى بعد الانتهاء من تصميم بنية قاعدة بياناتك في فيجوال بيزيك ولن تحتاج لمغادرته ولا لاستخدام المشعوذين (أقصد Wizards فهي تأتي بالانكليزية بمعنى ساحر) لأن الفيجوال سيقوم بتلقينك ما تريد كتابته وفوق هذا سيخبرك هو ببنية قاعدة البيانات وايضا سيمثل لك العلاقات خير تمثيل حيث تصل للعلاقة المطلوبة بنقطة فقط () .. تخيلوا اربع اسطر Inner Join ستصبح اربع نقاط !! واما الاخطاء فانتهي عهد ان تعمل فحص اين اضع Double؟؟!! وما هو مقابل int32؟؟ وماذا اعمل ب Null ... فالراعي الرسمي ل LINQ .. السيد IntelliSense سيقوم بالعمل ويوبخك اذا اخطأت هكذا اخطاء

لم تقتنع الظاهر ان اسلوبي لم يعجبك ... ساستخدم اسلوب اكثر تقليدية ... للسيد المحترم القارئ نقدم لكم مزايا LINQ:

- 1- التعامل مع كائنات قواعد البيانات ككائنات برمجية (جدول Customer يصبح مجموعة Customers Collection) أي احضارها للبرنامج وبالتالي تفعيل خاصية IntelliSense الرائعة .
- 2- استخدام دوال Functions فيجوال بيزيك الاعتيادية مع البيانات بدلاً من دوال SQL . مع ان جماعة مايكروسوفت اضافت دوال جديدة شديدة الشبه بدوال SQL .
- 3- واجهة واحدة للتعامل مع البيانات اينما كانت سواء بالذاكرة او بقواعد بيانات Oracle او SQL Server .
- 4- لست بحاجة لاستخدام SQL ومعرفة فروقاتها بين محركات قواعد البيانات المختلفة .
- 5- تصيد الأخطاء في الاستعلامات يتم قبل الترجمة واثناء كتابة الكود .
- 6- تمثيل العلاقات Foreign keys بشكل مجموعات فرعية Sub-Collections كما سترون لاحقاً .
- 7- اذا لم تقتنع بعد هذا اعتبرها شيء جديد يثير الفضول وجربها يا اخي .

هل سنبدأ بالكلام المفيد ؟

حسنًا اطلت الشرح والاقناع سأشرح قليلاً انواع مزودات LINQ لنتتقي ما يناسبك .
LINQ to SQL : مختصة بالتعامل مع قواعد بيانات SQL Server بكافة انواعها وباختصار اداة ترجمة قوية لاستعلاماتك إلى استعلامات SQL ترسل من تحت الطاولة دون علمك لقاعدة البيانات . واهم ما يعجبني فيها أنني استطيع استخدام أي دالة Function في جملة Where حتى دالة من تعريفك . ونحن كما ذكرت لكم سنتعامل مع هذا المزود من خلال قاعدة بيانات SQL Server Compact المضمنة في VS 2009 .
LINQ to Object : مختصة بالتعامل مع كائنات فيجوال بيزيك مثلاً استعلام لمصفوفة Array او لمجموعة Collection أو قائمة List أي شيء يدعم الواجهة IEnumerable.IInterface .
LINQ to XML : لجماعة مبرمجي الويب خاصة وبيانات XML حيث لا أعرف عنها الكثير فاتركها للمختصين فأنا اتعقد من منظر الأقواس <> يذكرني بالدوس وابامه السوداء .
LINQ to DataSet : ويمكن استخدام هذا المزود لقواعد البيانات الأخرى غير SQL Server . مثلاً Access أو Oracle أو MySQL .. ألخ هو طريقة للالتفاف على عدم دعم LINQ المباشر لهذه المحركات . فنحضرها إلى متناول استعلامات LINQ فسراً باستخدام DataSet القادرة على احضار أي قواعد بيانات .

وللعلم فإن **LINQ to ADO.Net** تطلق على كل من (linq to sql و linq to dataset و linq to entities)

ولا تقلقوا فالمبدأ نفسه وإذا تعلمت استعلامات أي مزود ستتعلم استعلامات البقية ولكن التنوع لاختلاف بنية البيانات في كل مزود .

بحمد الله انتهت المقدمة وسأبدأ في المقالات التعليمية حيث في المقالة القادمة كيفية انشاء برنامج يدعم LINQ to SQL .

اليوم الأول

بناء برنامج يدعم LINQ to SQL

السلام عليكم ورحمة الله وبركاته

سأبدأ من الألف إلى الياء وذلك للمبتدئين في استخدام LINQ خطوات عمل البرنامج:

١. إنشاء برنامج جديد فارغ من نوع . Windows Form Application
٢. إضافة و تصميم قاعدة البيانات عن طريق . Server Explorer
٣. إنشاء فئات العمل ل LINQ عن طريق SQLMetal
٤. تطبيق استعلامات LINQ للتعامل مع البيانات (عرض-حذف-إضافة-تعديل)

طبعا الخطوة الأولى سهلة ولا تحتاج لشرح ... وبعد انشاء البرنامج سيظهر لك نموذج وحيد Form1 اتركه كما هو ونبدأ بالخطوة الثانية....

بناء وتصميم قاعدة البيانات

لإنشاء قاعدة بيانات جديدة : نذهب للقائمة Project ونختار Add New Item و ثم نختار Local قاعدة بيانات ونسميها اي اسم. فيظهر معالج الاتصال بقاعدة البيانات اضغط Cancel لأننا لا نريد انشاء . DataSet سنجد ملف الداتا قد اضيف لنا Solution Explorer نضغط عليه فيفتح لنا Server Explorer ثم نضيف الجداول لقاعدة البيانات وننشئ العلاقات اللازمة.

إضافة قاعدة بيانات موجودة : وللسهولة ساستخدم قاعدة بيانات NorthWind.sdf الشهيرة حيث نختار من قائمة Project ثم Add Existing Item ثم نضيف ملف NorthWind.sdf التي تجدها في المجلد :

C:\Program Files\Microsoft SQL Server Compact Edition\v3.5\Samples

وسيطر معالج الاتصال بقاعدة البيانات فنضغط Cancel الان هذه الخطوة مهمة لكل قواعد البيانات المحلية : حيث أن الفيچوال ينشئ قاعدتي بيانات احدهما في مجلد البرنامج والثانية للتنفيذ في مجلد Debug وبالتالي فعندما نقوم باضافة بيانات اثناء تنفيذ البرنامج نتفاجأ انها اختفت عندما نستعرض البيانات في server explorer لذلك نقوم بالتالي : من نافذة المشروع Solution Explorer اختار ملف قاعدة البيانات وفي الخصائص عدل الخاصية Copy to Output Directory إلى. Do not Copy

الان نحفظ المشروع تمهيدا للخطوة الثالثة

إنشاء فئات العمل مع LINQ

ما هي فئات العمل مع LINQ ؟ هي مجموعة فئات تمثل قاعدة البيانات بكافة جداولها واعمدتها على شكل كائنات يمكن التعامل معها بسهولة من داخل

البرنامج فهي باختصار تقوم بتغليف كائنات قاعدة البيانات وتقديمها للمستخدم بشكل واضح وسهل كما اعتدنا مع جميع فئات .Net. وهذا يؤدي لتقليل الاخطاء بالبرنامج وتقليل الكود اللازم للتعامل مع البيانات.

ان البيئة المثالية لإنشاء فئات العمل هي O/R Designer ولكن لا اعلم لماذا لم تضاف مايكروسوفت دعم قواعد بيانات SQL CE له؟! وبالتالي سنستخدم البرنامج البديل وهو SQLMetal.exe وهو يأتي مع الفيجوال:

١. اولاً نغلق البرنامج ونذهب للمجلد C:\Program Files\Microsoft SDKs\Windows\v6.0A\bin

٢. ثم ننسخ الملف SQLMetal.exe إلى مجلد البرنامج الذي فيه ملف Northwind.sdf

٣. نبقى في مجلد برنامجنا وننشئ اختصاراً لملف SQLMetal.exe الذي نسخناه

٤. نفتح خصائص الاختصار الذي انشأناه وفي خانة (Target الهدف) نضع بعد اسم البرنامج الاوامر التالية:

`/namespace:NorthWind /code:NorthWind.vb /Language:vb /pluralize NorthWind.sdf`

للتوضيح أكثر تصبح خانة الهدف كالتالي :

`"C:\Users\Muhannad\Documents\Visual Studio 2008\Projects\SampleLINQ\SampleLINQ\SqlMetal.exe" /namespace:NorthWind /code:NorthWind.vb /Language:vb /pluralize NorthWind.sdf`

٥. والان نقوم بتشغيل ملف الاختصار ستظهر نافذة موجه الاوامر للحظات ثم تختفي وسنجد أنه لدينا ملف جديد في المجلد اسمه NorthWind.vb وهو المطلوب .

٦. نفتح مشروعنا ونختار من قائمة Project اضافة الملف Add existing Item و نذهب لمجلد البرنامج ونضيف ملفنا الجديد NorthWind.VB

٧. بقي خطوة أخيرة حيث يتطلب اضافة refrence للبرنامج ل..System.Data.Linq الطريقة لم لا يعرف من قائمة Project ثم Add refrence ثم تحت علامة .NET. ستجد System.Data.Linq أضفها للمشروع.

ملاحظة هامة : يجب إعادة إنشاء ملف الفئات عند أي تغيير في تصميم قاعدة البيانات مثلاً اضافة جدول جديد أو أعمدة جديدة لجدول موجود وعندها لا تحتاج الا لتشغيل الاختصار السابق وانتبه أن لا تكون قاعدة البيانات مفتوحة في مكان اخر كالفيجوال مثلاً.

وقد أضفت البرنامج الذي تم انشاءه في هذا اليوم بشكل رابط لمن يريد اختصار الوقت والتطبيق مباشرة : [LinqSample](#).

الان اصبحنا جاهزين لاستخدام LINQ To SQL في برنامجنا وسأبدأ قريباً بالخطوة الهامة الرابعة وهي التطبيقات وهي التي ستكشف لكم مدى سهولة وقوة تعبيرات LINQ ان شاء الله .

مع التحية

اليوم الثاني

البداية في استخدام LINQ

في الموضوع السابق انشأنا برنامجا يرتبط بقاعدة بيانات NorthWind.sdf أنشأنا فئات عمل Linq باستخدام SQLMetal ولمن يريد الاستفادة من الدرس يفضل انشاء البرنامج حسب الدرس السابق لأنني سأمزج بين المعلومات النظرية والتطبيق العملي مباشرة في كافة المراحل .

الآن سأطرق للموضوع المهم وهو التعامل مع LINQ بعد انتهينا من بناء الفئات باستخدام SQLMetal . ان الفئات التي تشكلت لدينا هي صلتنا بقاعدة البيانات فيعد الآن أصبح كل سجل بقاعدة البيانات ممثل بكائن وكل مجموعة سجلات ممثلة ب List - أو بنية تشبهها بالاحرى - مثلا الجدول Customers الآن نصل اليه عن طريق الكائن Customers والذي يحتوي على كائنات من نوع Customer التي انشأها لنا SQLMetal وستجد لكل فئة من نوع Customer خصائص هي عبارة عن أعمدة الجدول Customer وعند الشرح العملي سنتضح الصورة أكثر .

الآن في البرنامج ضع اداة DataGridView على كامل الفورم حيث سنستخدمها لعرض نواتج تجاربنا مع استعلامات Linq .

الخطوة الأولى لاستخدام LINQ TO SQL هو انشاء الكائن DataContext والذي مهمته تأمين الاتصال بقاعدة البيانات وطبعاً نحن لن ننشئ هذا الكائن مباشرة انما هناك الفئة Northwind التي انشأها لنا SQLMetal والتي هي اصلاً مشتقة من DataContext وبالتالي سننشئ كائن من هذه الفئة كمتغير محلي Field في الفورم . تحت كلمة Public Class

```
Public Class Form1
Dim MyDB As New NorthWind.NorthWind("Northwind.sdf")
```

طبعاً تلاحظ اننا كتبنا NorthWind.NorthWind وذلك لأننا عند انشاء الفئات sqlmetal طلبنا منه أن ينشئ نطاق للأسماء Namespace باسم NorthWind فلا بد للوصول لفئاتنا استخدام هذا النطاق قبلها . الآن أصبحت كل البيانات في متناول يدك وسنبدأ بتجربة اكواد مختلفة لتتعرف على استعلامات LINQ أكثر ونقول وداعاً لاستعلامات SQL وكود ADO.Net الممل والمكرر .

استعلامات LINQ

بشكل عام استعلامات LINQ اما تبدأ ب From وتعيد مجموعة سجلات أو ب Aggregate وتعيد عندها قيمة واحدة وساقوم بترجمة كل استعلام نستخدمه لاستعلام SQL المقابل له ليسهل فهمه .

اولاً: استعلامات From :

صيغتها العامة المبسطة

From element In collection [Where condition] Select elements[vars]

وفي مثالنا تصبح Collection تصبح MyDB التي تمثل قاعدة البيانات كاملة وطبعاً ما بين اقواس مربعة هو اختياري .

الآن سنضع جملة from بسيطة في الحدث Form_Load لتعرض البيانات في DataGridView .

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load

    MyDB.Log = Console.Out

    Dim mySource = From cust In MyDB.Customers Select cust

    DataGridView1.DataSource = mySource

End Sub
```

نفذ البرنامج وسترى الجدول Customers كاملاً لديك . ولقد أضفت العبارة MyDB.Log = Console.Out حيث ستكتب لك في نافذة Output ما يتم إرساله لقاعدة البيانات وسترى أنها استعلامات SQL المعروفة فهذه هي وظيفة LINQ to SQL ترجمة ما نكتبه لقاعدة البيانات ولا تنسى ان تلقي نظرة على نافذة Output بعد كل استعلام . أعود للاستعلام السابق بإمكانك اعتبار cust سجل ،، فترجمته لعبارة مفهومة هو " لكل سجل نفترض ان اسمه cust في مجموعة السجلات MyDB.Customers اعطني cust " يعني كل السجلات ومقابلته ب SQL :

SELECT * FROM Customers

أما ناتج الاستعلام mySource فهو نوع مجهول Anonymous Type ولا تشغل بالك كثيراً به فالتطبيقات البسيطة ل LINQ لا تحتاج لمعرفة قوية بهذه الأنواع . يمكنك اعتبار الناتج مجموعة تحتوي كائنات من نوع Customer .

وناتج الاستعلام يمكنك الاستفادة منه بعدة طرق مثلاً في المثال السابق استخدمناه ك Data Source . أو يمكنك التنقل خلاله من خلال حلقة For..Each مثال :

```
For Each mycust In mySource

    Console.WriteLine (mycust.ContactName)

Next
```

أو اخذ أول سجل منه فقط واستعراض بياناته مثلاً ...

```
Dim FirstCust = mySource.First

Console.WriteLine (FirstCust.CompanyName, FirstCust.ContactName,
FirstCust.City, FirstCust.Address)
```

وكل ما ينطبق على List(of T) ينطبق عليه .. وفوق ذلك يمكن الاستعلام عنه في جملة From جديدة..مثلا

```
Dim LondonCust = From x In mySource Where x.City = "London" Select x
```

وبإمكانك ان تضع ناتج الاستعلام LondonCust كمصدر DataSource للداتا جريد .

لاحظ اثناء كتابتك للاستعلامات السابقة أن Visual BASIC يقترح عليك العناصر و الحقول ويصحح لك الأخطاء وهذا سبب قولي أن استخدام استعلامات LINQ اسهل واصح من استخدام استعلامات SQL فلامجال مع Linq لخطأ لاتعرف ما هيته ترسله لك قاعدة البيانات وتختار في امره .ويكون سببه بسيط كاختيارك لنوع بيانات خاطئ مثلا String بدلا من Integer فهنا المحرر سيخبرك بالخطا مباشرة.

نعود لبعض التطبيقات وساسرد هنا استعلام LINQ وقم انت باستبداله في حدث Form_Load وشاهد النتيجة وبعده ساذكر استعلام SQL المقابل ولسهولة فهم الاستعلامات ساكتب كل بداية كلمة محجوزة في سطر باستخدام _ .

```
Dim mySource = From cust In MyDB.Customers _
                Select cust.ContactName, cust.City
```

SQL : SELECT ContactName,City FROM Customers

```
Dim mySource = From cust In MyDB.Customers _
                Select cust.ContactName, cust.City
```

SQL : SELECT ContactName,City FROM Customers

```
Dim mySource = From cust In MyDB.Customers _
                Where cust.City = "London"
                Select cust.ContactName, cust.ContactTitle, cust.Address
```

SQL : SELECT ContactName, ContactTitle, Address FROM Customers WHERE City='London'

```
Dim mySource = From ord In MyDB.Orders _
                Where ord.Customer.City = "London"
                Select ord.OrderDate,ord.OrderID,ord.Customer.ContactTitle
```

SQL : SELECT Orders.[Order Date],Orders.[Order ID],
Customers.[Contact Title] FROM Orders
INNER JOIN Customers ON Customers.[Customer ID] = Orders.[Customer ID]
WHERE Customers.City = 'London'

لاحظ الاستعلام الاخير أن العلاقات ممثلة بشكل مجموعة كائنات فكل كائن Order يحتوي على كائن Customer يمثل العلاقة بين Customer ID في الجدولين وهذه طريقة ذكية وسهلة لتمثيل العلاقات خاصة العلاقات المتشابهة بين عدة جداول حيث كنت سابقا اكتب اكثر من جملة Inner Join في الاستعلام . وطبعا يمكنك استخدام الكلمات

المحجوزة التالية مع الاستعلامات Like, Order By, Distinct واقترح عليك ان تجرب ما يقترحه عليك المحرر وتحاول بناء جمل استعلامات LINQ اكثر تعقيدا لتعتاد على استخدامها . الى لقاء قريب .حيث سأتطرق ل Aggregate و استعلامات اكثر تعقيدا .

اليوم الثالث :

اليوم سنكمل مشوارنا مع استعلامات LINQ ولن ادخل بكل تفاصيل الكلمات المحجوزة للاستعلامات

ثانيا : استعلامات Aggregate

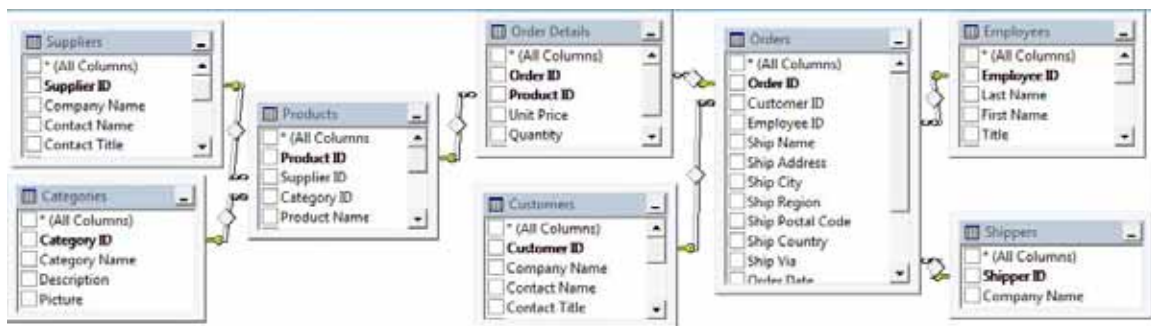
تتميز بانها تعيد قيمة واحدة لأحد دوال التجميع Aggregation
Sum مجموع القيم - **Average** المعدل الحسابي - **Count** عدد القيم - **Min** أصغر قيمة - **Max** أكبر قيمة .
وهي سهلة الاستخدام
مثال أضف الكود التالي لبرنامجنا في أي مكان تريد ...

```
Dim CustomerCount = Aggregate cust In MyDB.Customers Into Count()
'SQL : SELECT Count(*) From Customers
Dim OrderSum = Aggregate ord In MyDB.Orders Into Max(ord.OrderDate)
'SQL : SELECT Max([Order Date]) FROM Orders
Console.WriteLine("Customers Count: {0} Type : {1} ", CustomerCount,
TypeName(CustomerCount))
Console.WriteLine("Max Order Date Count: {0} Type : {1} ", OrderSum,
TypeName(OrderSum))
```

وقد اضفت جملة SQL الموافقة بعد الاستعلام وكما تلاحظ فإن استعلامات Aggregate تعيد قيمة واحدة دائما ونوع متغير ناتج الاستعلام سيكون أحد انواع DataType وليس مجموعة Collection كما اعتدنا . كما يمكن ان يحتوي استعلام From على تجميع باستخدام Aggregate

العلاقات في LINQ TO SQL :

طبعاً جميعنا يعرف العلاقات وفائدتها . ومع استخدام Linq اصبح لدينا تطبيق سهل ووصول جيد لكافة كائنات قاعدة البيانات ... وكما سبق فإننا نستخدم قاعدة بيانات NorthWind ولذلك قبل الخوض في علاقاتها هذه صورة توضح بنية العلاقات فيها



طبعاً كما ذكرت انشاء العلاقات وحفظها في قاعدة البيانات قبل انشاء الفئات باستخدام sqlMetal يسهل بناء استعلاماتنا بشكل كبير حيث يقوم باني الفئات (sqlmetal أو O/R designer) بتضمين العلاقات وتقديمها لنا

بطريقة برمجية ككائنات . مثلاً من الشكل السابق لنفرض أنني أريد معرفة فقط أسماء وأرقام المنتجات التي شحنت الى إيطاليا . سيكون عندها استعمال LINQ بهذه البساطة :

```
From det In MyDB.OrderDetails
Where det.Order.ShipCountry = "Italy"
Select det.Product.ProductID, det.Product.ProductName Distinct
```

هذا في حالة ترجمة العلاقات في الفئات . أما اذا لم يتم انشاء العلاقة في قاعدة البيانات فانت بحاجة ل Join ويصبح الاستعلام نفسه كالتالي :

```
From det In MyDB.OrderDetails
Join ord In MyDB.Orders On ord.OrderID Equals det.OrderID
Join prd In MyDB.Products On prd.ProductID Equals det.ProductID
Where ord.ShipCountry = "Italy"
Select prd.ProductID, prd.ProductName Distinct
```

وطبعاً نظير الاستعلامين السابقين في SQL هو :

```
SELECT DISTINCT Products.[Product Name] FROM [Order Details]
INNER JOIN Orders ON [Order Details].[Order ID] = Orders.[Order ID]
INNER JOIN Products ON [Order Details].[Product ID] = Products.[Product ID] WHERE (Orders.[Ship Country] = 'Italy')
```

والنتيجة أن علاقة واحد إلى كل أو ما تسمى Foreign Key حيث يكون هناك جدول مفتاح اساسي (-Primary Key) مثل جدول Customers و جدول اخر يحتوي المفتاح الفرعي (Foreign-Key) الذي يشير للمفتاح الاساسي تكون ممثلة في فئات Linq to SQL بشكل :

(كل كائن من جدول Primary-Key له مجموعة من كائنات جدول Foreign-Key)

والعكس غير صحيح

(كل كائن من جدول Foreign-Key ممثل بكائن وحيد من جدول Primary-Key)

وهذه الجمل صحيحة مهما تشعبت العلاقات كما شاهدنا بالمثال السابق.

وهذه الميزة لها عيوب عند التعامل مع كمية كبيرة من البيانات حيث سيتم تحميل الجدول المطلوب مع علاقاته والكائنات الفرعية مما يشكل حملاً على الذاكرة والمعالج وهذا يمكن التحكم به من خلال Load Options للكائن DataContext كما سنرى لاحقاً .

الاستخدامات الاساسية للاستعلامات :

بناء مصدر بيانات Data Source :

كما رأينا سابقاً في المقال الثاني يمكن استخدام أي استعلام Linq كمصدر بيانات لأي عنصر تحكم لعرض البيانات وهناك ملاحظة مهمة بالنسبة لاستعلامات Linq وهي أن أي مجموعة استعلامات تعيد نفس القيمة أو السجل أو نفس مجموعة السجلات تؤثر لنفس الكائن حتى لو كانا استعلامين مختلفين . فقد تتفاجأ عند وضع مصدر بيانات ل combobox و DataGridView من نفس الجدول أنه عند تحريك الاختيار في الكومبو تتحرك المؤشرات في datagrid.

التجميع Grouping :

باستخدام Group By حيث يتم فرز السجلات على شكل مجموعات مستقلة وفقاً لمعيار معين . لنفرض أنني لا أريد عرض جدول Products بالكامل وإنما أريده بطريقة منسقة ومجموعة حسب Category عندها نلجأ ل group By

```
From pro In MyDB.Products
Group pro By pro.CategoryID Into CatGroup = Group
```

هنا يصبح ناتج الاستعلام عبارة عن مجموعتين بشكل شجري بالتالي نحن بحاجة لحلقتين For Each للتجوال عبر جميع السجلات الحلقة الأولى الأولى تعيد مجموعة CatGroup والحلقة الثانية تتجول في هذه المجموعة كالتالي :

```
Dim mySource = From pro In MyDB.Products
Group pro By pro.CategoryID Into CatGroup = Group
```

```

For Each grp In mySource
    Console.WriteLine("Category ID = {0}", grp.CategoryID)
    For Each prod In grp.CatGroup
        Console.WriteLine("Product Name: " & prod.ProductName)
    Next
Next

```

والاستخدام الاخر الأكثر شيوعا للتجميع هو مع الدوال التجميعية مثلا اذا اردنا أن نجد عدد الطلبات orders لكل موظف Employee سنستخدم الاستعلام التالي :

```

From ord In MyDB.Orders _
Group ord By ord.EmployeeID Into EmpGroup = Group _
Select EmployeeID, OrdersCount = EmpGroup.Count()

```

الترتيب :

باستخدام Order By ويمكن استخدام Take و Skip معها لتعطينا جزء من المجموعة مثلا للحصول على اخر ١٠ طلبات نستخدم الاستعلام :

```

From ord In MyDB.Orders _
Select ord Order By ord.OrderDate Descending _
Take 10

```

تصفية السجلات :

وهو باستخدام الكلمة Where وبعدها قيمة منطقية و ٩٠% من القيم المنطقية والدوال مقبولة بالنسبة Linq to SQL ولاحقا سأعطي أمثلة للدوال واستخدامها بطريقة سهلة مع Linq .

ونصيحتي لكل متعلم جديد ل LINQ جرب ثم جرب ثم جرب كل صيغ الاستعلامات الممكنة . لكي تأخذ مرونة في التعامل
وإلى اللقاء مع المقال القادم الذي سيكون عن تعديل البيانات وإضافة وحذف البيانات مع Linq . والسلام

اليوم الرابع

تعديل وتحديث البيانات

السلام عليكم

كمدخل لمفاهيم تعديل وتحديث البيانات أحب أن اذكر الجميع بأننا في Linq أصبحنا نتعامل مع كائنات وبالتالي مفهوم تحديث البيانات هو عبارة عن تعديل لخصائص هذه الكائنات والمجموعات أو حذف أو اضافة كائنات ثم يقوم مزود SQL to Linq بتحويل تعديلاتنا إلى جمل SQL يتم ارسالها لقاعدة البيانات .

أولا إضافة البيانات :

١ - إضافة البيانات لجدول واحد :

نقوم أولا بانشاء نسخة من كائن الجدول (مثلا Customer أو Category) ثم نسند القيم للحقول المطلوبة ونستخدم الكائن DataContext لإلحاق البيانات بقاعدة البيانات والميزة الجميلة أن LINQ يقوم لوحده بتحديث بقية الحقول التي لم ندخلها لقيمها الجديدة. مثلا الكود التالي يضيف بيانات لجدول الفئات Categories في NorthWind ثم يعرض قيمة ID للفئة الجديد .

```
Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button2.Click
    Dim MyDB As New NorthWind.NorthWind("Northwind.sdf")
    MyDB.Log = Console.Out
    Dim myCat As New NorthWind.Category
    With myCat
        .CategoryName = "Arabicaa"
        .Description = "Arabs"
    End With
    MyDB.Categories.InsertOnSubmit(myCat)

    MyDB.SubmitChanges()
    MsgBox("new ID : " & myCat.CategoryID)

End Sub
```

لا تنسى أن تلقي نظرة على نافذة Output .

٢ -إضافة البيانات لجدول مترابطة Master/Details:

غالبا ما نحتاج ادخال المستخدم لإضافة البيانات لعدة جداول مرتبطة مثلا عند ادخال العميل لفاتورة ما نحتاج لإدخال البيانات في جدول الفواتير ثم نضيف تفاصيل كل فاتورة في جدول التفاصيل ... وفي قاعدة Northwind لنفرض أننا نريد إدخال منتجات Products جديدة تتبع لفئة جديدة ..ما سنقوم به هو انشاء كائن جديد من نوع Category وسنجد في خصائصه تلقائيا مجموعة Products فنضيف لها ما نحتاجه من منتجات و نرسل التغييرات لقاعدة البيانات .وهذا مثال الكود :

```
Dim myCat As New NorthWind.Category
With myCat
    .CategoryName = "Arabicooo"
    .Description = "Arabians"
    .Products.Add(New NorthWind.Product With _
        {.ProductName = "AAAAAA", .Discontinued = False})
    .Products.Add(New NorthWind.Product With _
        {.ProductName = "BBBBBB", .Discontinued = False})
    .Products.Add(New NorthWind.Product With _
        {.ProductName = "CCCCCC", .Discontinued = False})
End With

MyDB.Categories.InsertOnSubmit(myCat)

MyDB.SubmitChanges()
MsgBox("CategoryID ID : " & myCat.CategoryID & " My Products : " & _
    myCat.Products.Count)
```

ولاحظ أننا لم نضع CategoryID في كودنا لأن Linq to SQL سيتكفل بإحضارها . وإذا القيت نظرة على نافذة Output ستجد أن كودنا السابق تمت ترجمته لخمس جمل SQL الأولى لإضافة فئة جديدة Category والثاني للحصول على CategoryID للفئة الجديدة و الثلاث البقية لإضافة المنتجات Product الجديدة .

ثانيا تعديل البيانات :

إذا فهمت الفقرة السابقة فالتالي أسهل باختصار طريقة التعديل . (أحضر الكائن المطلوب .. عدله ... أرسله) مثلا لنعدل أحد المنتج السابق الذي أضفناه المسمى CCCCCC ليصبح اسمه XXXXXX هذا الكود

```
Dim myProduct = From pro In MyDB.Products _
                  Where pro.ProductName = "CCCCCC" Select pro
If myProduct.Count > 0 Then myProduct.First.ProductName = "XXXXXX"
MyDB.SubmitChanges()
```

ثالثا حذف البيانات :

ايضا بنفس الطريقة السابقة : أحضر الكائن ثم احذفه ثم أخبر القاعدة (قاعدة البيانات طبعا ...) وهذا كود لحذف المنتج XXXXXX :

```
Dim myProduct = From pro In MyDB.Products _
                  Where pro.ProductName = "XXXXXX" Select pro
If myProduct.Count > 0 Then _
    myDB.Products.DeleteOnSubmit(myProduct.First)
MyDB.SubmitChanges()
```

ولحذف كل ما أضفناه في الأكواد السابقة لقاعدة بيانات Northwind حيث سنحذف Category التي تبدأ ب Arab ومنتجاتها بهذا الكود :

```
Dim myCat = From cat In MyDB.Categories _
             Where cat.CategoryName Like "Arab*" Select cat

If myCat.Count > 0 Then
    For Each cat In myCat
        MyDB.Products.DeleteAllOnSubmit(cat.Products)
    Next
End If

MyDB.Categories.DeleteAllOnSubmit(myCat)
MyDB.SubmitChanges()
```

رابعا: نطاق العمليات Transactions :

اعتدنا دائما لإنشاء برنامج يعتمد عليه أن نستخدم عن تغيير البيانات كائنات Transaction خاصة عند حدوث خطأ ما أثناء ادخال بيانات مترابطة فيجب الغاء العملية برمتها .. فليس من المعقول أن نسجل الفاتورة مثلا ناقصة بعض الاغراض .

وبالنسبة ل Linq To SQL فهي تدعم Transaction داخليا فعند أي استدعاء ل SubmitChanges سيقوم مزود Linq تلقائيا بإنشاء Transaction خاص بهذا الاستدعاء حيث ولدى حدوث أي خطأ فإن عملية تحديث البيانات سيتم الغائها تلقائيا .. جرب هذا الكود أن Category المقصودة لازالت موجودة . (حيث تعمدت أن أولد خطأ عن طريق حذف كائن من Category مرتبط بكائنات من Product وهذا لا تسمح به قاعدة البيانات حيث يجب حذف الطرف الآخر من العلاقة).

```
Dim myCat = From cat In MyDB.Categories Where cat.CategoryID = 2
MyDB.Categories.DeleteAllOnSubmit(myCat)
Try
    MyDB.SubmitChanges()
Catch ex As Exception
    MsgBox("Error !! all changes canceled")
End Try
```

ويمكن تعديل الكود السابق بحيث نتتبع الخطأ و نعالجه في جملة Catch ثم نستدعي SubmitChanges مرة أخرى.

وهناك طريقة أخرى عن طريق خاصية Transaction للكائن DataContext حيث بإمكاننا استخدام نطاق Transaction من كائنات ADO.net وستكون كل العمليات التي ستتم ضمن هذا النطاق وإذا عملنا RollBack ستلغى كل التغييرات .

بهذا ينتهي مقال اليوم .. امل أن يكون مفيدا للجميع .. إلى لقاء مع الموضوع القادم الذي سيكون عن ميزة رائعة يمكن الاستفادة منها في Linq وهي Extensions .. وايضا سنلقي نظرة أعمق على الكائن DataContext .

والسلام ...

اليوم الخامس

توسعة الوظائف Extensions

إن كل نوع بيانات أو فئة في VB2008 له مجموعة من الدوال كما اعتدنا مثلا (String,Integer,List) ونحن نستخدم هذه الوظائف بشكل اعتيادي في كل برنامجنا .. ولكن ماذا لو احتجنا لتوسيع هذا النوع ليحتوي وظائف جديدة نحتاجها في البرنامج ؟ .. سابقا كنا ننجز هذا العمل باشتقاق فئة جديدة و اضافة الوظائف لها ولكن الآن يمكننا بسهولة اضافة وظيفة أو أكثر لنوع بيانات باستخدام Extensions .

في Linq هذه الميزة أحيانا تغنينا عن كتابة أكواد كثيرة ومتكررة لتحقيق هدف معين .. لنفترض أننا نريد الحصول على مجموع حقول نصية لعدة سجلات ... لن نجد دالة تجمع في Linq أو في SQL تحقق الهدف المطلوب ولذلك سننشئ دالتنا الخاصة ونلحقها بنوع الحقل الذي نريد تعديله لتصبح كبقية دوال التجميع (Sum,Count...) جزء من نوعنا .

ترجمة للكلام السابق : مهمتنا الحالية هي انشاء دالة (وظيفة) نسميها JoinNames تقوم بعرض مجموعة حقول نصية بشكل حقل نصي واحد مفصول بفواصل . وسيكون مثالي من قاعدة بيانات NorthWind عرض الفئات Categories بحيث يكون بجانب كل سجل من Category مجموعة المنتجات Products التي تخص هذه الفئة مفصولة بفواصل .

فالنوع المطلوب توسعته هنا هو Product وبالأصح مجموعة Product يعني بلغة البرمجة

IEnumerable(of Product)

الكود كالتالي بكل بساطة :

```
Imports System.Runtime.CompilerServices
...
...
Public Module Ext
    <Extension()> Public Function JoinNames(ByVal ProductList As
IEnumerable(Of NorthWind.Product)) As String

        Dim Names = From pro In ProductList Select pro.EnglishName

        Dim NamesString = String.Join(",", Names.ToArray)
        Return NamesString
    End Function
End Module
```

الان لنستخدم هذه الدالة الجديدة في مثالنا و نعرضها في datagrid :

```
Dim mySource = From cat In MyDB.Categories _
                Let ProductsNames = cat.Products.JoinNames _
                Order By cat.CategoryName _
                Select Number = cat.CategoryID, Name =cat.CategoryName, _
                Products = ProductsNames

DataGridView1.DataSource = mySource
```

نظرة أعمق على DataContext

خصوصية الكائنات وتفرداها مع Linq to SQL :

إن الكائن DataContext كما ذكرنا سابقا هو وسيط اتصال Linq To SQL مع قواعد البيانات فهو الذي يترجم ما نعمله بالكائنات إلى جمل SQL . وايضا من مهماته التحكم باحضار البيانات والتحقق منها !!! كيف ذلك !! لكي لا يحدث تضارب بالبيانات ولأسباب أخرى فإن DataContext لا يكرر وجود نفس البيانات ... المعنى أنه لو استدعيت نفس مجموعة السجلات في مكانين مختلفين وبطريقتين مختلفتين فسيكون لديك في الذاكرة نفس المجموعة من السجلات !! وهذا مثال :

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click

    ''Ado.net Objects
    Dim con As New SqlConnection("Data Source =
Northwind.sdf")
    Dim com1 = con.CreateCommand
    com1.CommandText = "Select Top (1) * From Categories Where [Category
ID] = 3"
    con.Open()
    Dim Cat1 = com1.ExecuteResultSet(SqlServerCe.ResultSetOptions.None)
    Dim Cat2 = com1.ExecuteResultSet(SqlServerCe.ResultSetOptions.None)
    con.Close()
    ''LINQ to SQL DataContext
    Dim MyDB As New NorthWind.NorthWind("Northwind.sdf")
    MyDB.Log = Console.Out
    Dim myCat = From catg In MyDB.Categories _
                Where catg.CategoryID = 3 Select catg
    Dim Cat3 = myCat.First
    Dim Cat4=MyDB.Categories.Where(Function(cat) cat.CategoryID = 3).First

    ''التساوي فحص
    Console.WriteLine("Cat1,Cat2 are Equals ? " & Cat1.Equals(Cat2))
    Console.WriteLine("Cat3,Cat4 are Equals ? " & Cat3.Equals(Cat4))

End Sub
```

نحن تعودنا في ADO,Net أن يكون الكائنتين مختلفين رغم أنهما متساويين بالمحتويات جرب الكود السابق وسترى أن كائني resultset متساويين (للتذكرة كائن ResultSet مثل SqlDataReader ولكنه قابل للكتابة) ما أهمية ذلك ؟؟؟ من ذلك نستنتج أن تنتبه عند أجراءك تعديلات على الكائنات حتى الكائنات التي كنت تعتقد أنها تحتفظ بالقيم الأصلية قبل التعديل ستكون قد تعرضت للتغير من قبل DataContext .
ما الحل ؟؟ خذ هذ القاعدة و طبقها دائما في Linq to SQL

ملاحظة هامة :

دائما انشئ كائن جديد DataContext لاستعلاماتك يعني لاتعامله مثل Connection وتصنع كائنا واحدا لكل التطبيق انما اجعله كائنا على مستوى الاجراء أو الوظيفة Procedure Level كما في جميع امثلي السابقة اذا لاحظتم .

التحميل من قاعدة البيانات :

عندما نكتب استعلاما ما من استعلامات Linq to SQL فإننا لانقوم فعليا الا بتعريف الاستعلام وقولبة المتغيرات لاستقبال نتائج الاستعلام . ويتم تنفيذ الاستعلام (أي ترجمته لجملة SQL وارسالها لقاعدة البيانات) لعرض النتائج عند الحاجة لذلك ويمكنك ان تلاحظ ذلك بمشاهدة نتائج نافذة OutPut اذا كنت وضعت السطر

```
MyDB.Log = Console.Out
```

مثلا اذا كتبنا السطر التالي

```
Dim myCat = From catg In MyDB.Categories Select catg
```

فهو عبارة عن تعريف الاستعلام وتنفيذه الفعلي يتم عندما يحتاج البرنامج للبيانات وذلك أما بحلقة For Each أو عند استدعاء الأمر ToList أو عند اسناده ل datasource جرب كتابة كود لذلك وتتبع تنفيذ البرنامج بوضع BreakPoint و سترى متى يتم عرض جملة SQL .

فإذا أردنا أن يتم التنفيذ فورا فقط حول السطر السابق إلى ما يلي

```
Dim myCat = (From catg In MyDB.Categories Select catg).ToList
```

والحالة الأخرى التي يتم فيها ارسال جملة SQL هي عند اعتماد التغييرات باستخدام SubmitChanges .

وأما بالنسبة للكائنات الفرعية (أي المرتبطة بالعلاقات) فهناك عدة خيارات :

تحميل الكائنات الفرعية عند الحاجة لذلك : وهو الخيار الافتراضي وهنا تكون خاصية ObjecttrackingEnabled معينة للقيمة True وهنا يتم تحميل أي كائن فرعي (جدول مرتبط) عند اللزوم .

تحميل الكائنات الأساسية فقط : عندها يتم تحميل الكائنات الأساسية فقط ولا يتم تحميل الكائنات الفرعية وذلك باسناد القيمة False للخاصية ObjecttrackingEnabled. وتصبح البيانات للقراءة فقط !! .

التحكم بتحميل الكائنات : وذلك عن طريق خيارات التحميل **Load Options** فنتحكم نحن بما يتم تحميله من قاعدة البيانات مع كل كائن . مثلا بإمكاننا أن نخبر DataContext أنه عند تحميل مجموعة Orders أن يحمل مع كل كائن Order مجموعة OrdersDetails و هنا فإن DataContext يرجع لخاصية ObjectTrackingEnabled لمعرفة ما إذا كان يجب تحميل بقية الكائنات الفرعية أم لا .

لتوضيح الأمر أكثر سأناقش هذه الحالات لاحقا ثم أعرض مثال توضيحي. وقبل ذلك فإن طريقة استخدام LoadOptions كالتالي :

لنفترض أنني أريد تطبيق المثال في الأسطر السابقة أي تحميل كل كائن Orders مع كائن OrdersDetails :

```
Dim MyDB As New NorthWind.NorthWind("Northwind.sdf")
MyDB.Log = Console.Out
'LoadOptions انشاء كائن
Dim myopt As New System.Data.Linq.DataLoadOptions
'تحديد ما نريد تحميله من العلاقات للكائن Order
myopt.LoadWith(Of NorthWind.Order) (Function(o As NorthWind.Order) o.OrderDetails)
```

```
' ' DataContext مع السابقة مع
MyDB.LoadOptions = myopt
```

قد يتسائل شخص ما ما فائدة ذلك؟؟؟ لنترك كل شي كما هو افتراضيا وندع DataContext يطلب الكائنات التي نحتاجها !!

وجوابي هو : افترض أن لديك دالة تكرارية تستدعي كائنات Order و OrderDetails فسيكون الناتج لدينا حوالي ١٠٠ استدعاء sql سيرسل لقاعدة البيانات بينما بإمكانك تحديد تحميل OrderDetails مباشرة مع الكائن Orders وعندها لن يرسل لقاعدة بياناتك سوى استدعاء واحد جرب ذلك ولاحظ الفرق في الأداء.(كثيرون يشتكون من أداء Linq والسبب هو أن كودهم يواصل ارسال الاستعلامات ويقتل الأداء .

الآن لنوضح كل ما ذكرناه سابقا بمثال عملي

ومثالي هذا الكود وهو لايعمل شيئا سوى تحميل كائنات من قاعدة البيانات ولكن تتبع تنفيذه و راقب نافذة OutPut لترى ما يحدث توضيح الخطوات موجود مع الكود :

```
Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button2.Click

    ' ' ' ' ' تجربة 1
    Dim MyDB1 = CreateDataContext()
    ' ' ' ' ' الطب عند الكائنات تحميل تمكين
    MyDB1.ObjectTrackingEnabled = True
    ' ' ' ' ' لوجود مباشرة الاستعلام تنفيذ سيتم
    ' ' ' ' ' نافذة من البيانات لقاعدة إرساله يتم الذي الاستعلام الخطوة في راقب هنا
    Dim Q1 = (From ord In MyDB1.Categories Select ord).First
    ' ' ' ' ' ترى كما اخر استعلام تنفيذ تم هنا
    Dim Q2 = (From prod In Q1.Products Select prod).ToList
    ' ' ' ' ' كائنات وجود النتيجة
    MsgBox(Q2.Count)

    ' ' ' ' ' تجربة 2
    Dim MyDB2 = CreateDataContext()
    ' ' ' ' ' الكائنات تحميل الغاء
    MyDB2.ObjectTrackingEnabled = False
    ' ' ' ' ' السابقة الاستعلام جملة نفس لاحظ
    Dim Q3 = (From ord In MyDB2.Categories Select ord).First
    ' ' ' ' ' شيء أي ارسال لا يتم هنا لاحظ
    Dim Q4 = (From prod In Q3.Products Select prod).ToList
    ' ' ' ' ' خالية النتيجة
    MsgBox(Q4.Count)

    Dim myOpt As New System.Data.Linq.DataLoadOptions
    ' ' ' ' ' الفرعية الكائنات تحميل ذكرناه كما تعريف
    ' ' ' ' ' للكائن
    myOpt.LoadWith(Of NorthWind.Category) (Function(cat As NorthWind.Category)
cat.Products)

    ' ' ' ' ' تجربة 3
    Dim MyDB3 = CreateDataContext()
    ' ' ' ' ' الطلب عند التحميل تمكين
    MyDB3.ObjectTrackingEnabled = True
    ' ' ' ' ' التحميل خيارات تحديد
    MyDB3.LoadOptions = myOpt
    ' ' ' ' ' يختلف كيف الاستعلام راقب الآن
    Dim Q5 = (From ord In MyDB3.Categories Select ord).First
    ' ' ' ' ' تحميلها سبق لأنه هنا استعلام لا يوجد
    Dim Q6 = (From prod In Q5.Products Select prod).ToList
    ' ' ' ' ' نتائج يوجد
    MsgBox(Q6.Count)

    ' ' ' ' ' تجربة 4
    Dim MyDB4 = CreateDataContext()
    ' ' ' ' ' الطب عند التحميل الغاء
    MyDB4.ObjectTrackingEnabled = False
    ' ' ' ' ' التحميل خيارات
```

```

MyDB4.LoadOptions = myOpt
'' ايضا الاستعلام جملة لاحظ
Dim Q7 = (From ord In MyDB4.Categories Select ord).First
'' ينفذ الاستعلام
Dim Q8 = (From prod In Q7.Products Select prod).ToList
'' موجود النتائج ايضا
MsgBox(Q8.Count)

End Sub
Function CreateDataContext() As NorthWind.NorthWind
'' لإنشاء بسيطة دالة DataContext
Dim myDB As New NorthWind.NorthWind("Northwind.sdf")
myDB.Log = Console.Out
Return myDB
End Function

```

هكذا أعتقد أنني استطعت على عجلة إعطاء دفعة لمن يريد الانتقال لاستخدام LINQ to SQL ليضع قدمه على طريق البداية. أرجو أن أكون قد وقفت في العرض والأمثلة .

مع أنني لاحظت عدم وجود أي أسئلة أو استفسارات وقد كان أحد أساتذتي عند كل نهاية درس يسأل "هل من سؤال؟" وعندما لا أحد يسأل يقول "إما كلكم فاهمين أو ولا واحد كان سامعني "

المهم أن يكون طرحي مفيدا ولا تنسونا من الدعاء . ولم تنتهي بعد حيث لدي أيضا بعض الأفكار و التقنيات لاستخدام Linq to SQL وسأطرحها على شكل مواضيع مستقلة والسلام عليكم