

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 برامجنا .. 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 تغنيانا 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 و تفرد لها مع Linq to SQL :

ان 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 الكائناتين 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 أجرائك تعديلات على الكائنات فحتى الكائنات التي كنت تعتقد أنها تحتفظ بالقيم الأصلية قبل التعديل ستكون قد تعرضت للتغير من قبل DataContext .
ما الحل؟؟ خذ هذه 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 بإسناد 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)
'ربط الخيارات السابقة مع DataContext
MyDB.LoadOptions = myopt
```

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

وجوابي هو : افترض أن لديك دالة تكرارية تستدعي كائنات Order و OrderDetails فسيكون الناتج لدينا حوالي ١٠٠ استعلام sql سيرسل لقاعدة البيانات بينما بإمكانك تحديد تحميل OrderDetails مباشرة مع الكائن Orders عندها لن يرسل لقاعدة بياناتك سوى استعلام 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 لا يعمل 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 تنفيذه سيتم'
    'First'
    'نافذة من البيانات لقاعدة إرساله يتم الذي الاستعلام الخطوة في راقب هنا'
    Dim Q1 = (From ord In MyDB1.Categories Select ord).First
    'تري كما اخر استعلام تنفيذه 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
    'حالية النتيجة'
    MsgBox(Q4.Count)

    Dim myOpt As New System.Data.Linq.DataLoadOptions
    'الفرعية الكائنات تحميل ذكرناه كما تعريف'
    'Category'
    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
    'اختلف 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 لا استعلام'
    Dim Q8 = (From prod In Q7.Products Select prod).ToList
    'موجود النتائج ايضا'
    MsgBox(Q8.Count)

End Sub
Function CreateDataContext() As NorthWind.NorthWind
    'إنشاء 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