شرح Linq

تم الشرح بلغة #c

للمهندس سعد الضبي

ssyd12@hotmail.com

إهداء إلى روح والدي رحمه الله تعالى

مبتدىء	×
متوسط	✓
خبير	✓

لمشاهدة المزيد من كورساتنا تفضلو الى قناتنا المبرمج العربي على الرابط التالي

https://www.youtube.com/channel/UCinR4wnJdU-zr3yn-9ywZyQ?view_as=subscriber

فهرست	
العنوان	رقم الصفحة
تعريف Linq	٣
هيكلية Linq	٤
المعاملات في Linq	٦
المعامل OfType	٧
المعامل OrderBy	٨
المعامل thanBy	٩
المعامل GroupBy	١٠
المعامل select	١٠
المعامل Average	١١
المعامل count	١٢
المعامل Max	١٢
المعامل SUM	١٢
LINQ TO SQL	١٣
تكوين Linq To Sql	١٣
الاتصال بقاعدة البيانات	١٤
اختبار الاتصال	١٥
تكوين بيئة LINQ TO SQL	١٦
شرح بيئة LINQ TO SQL	١٨
مثال تطبيقي	١٩
تضمين كلاس الإتصال	١٩
الإضافة إلى قواعد البيانات	٢٠
التعديل على البيانات	٢١
الحذف من البيانات	٢٢
جلب البيانات من قواعد البيانات	٢٣
مثال تطبيقي	٢٥
مثال تطبيقي	٢٦
مثال تطبيقي	٢٧
جلب البيانات بواسطة شرط	٢٨
جمع عمود من DataGrideView	٢٩

قبل أن نبدأ في سرد دروس هذا الكتاب ستلاحظ ان الكتاب استخدم كثيراً الكلمات المحجوز التالية

١- Var ويطلق عليه متغير البيانات المجهولة وأنشأته Microsft في Net. لعلاج الكثير من المشاكل التي تنشأ بسبب غموض القيمة الراجعة مختصر القول اذا لم تكن متأكد من نوع القيمة الراجعة هل string او int او Boolean يمكنك استخدام متغير البيانات المجهولة var ثم اعطائه كائن

٢- عائلة Collection : استخدمنا في هذا الكتاب العديد من الأمثلة التي تعتمد على عائلة Collections إذا لم يكن لديك فكرة عن عائلة Collections فقد قمنا بشرحها على قناتنا المبرمج العربي
على الرابط التالي
https://www.youtube.com/playlist?list=PLc27PqKL-_w7p3Yafirvv0J92190vYo9q

تعريف Linq

حزمة Linq وتسمى جمل الاستعلام المتكاملة او جمل الاستعلام المدمجة ويستخدمها المبرمج لتسهيل جمل الاستعلام وعند استخدامها ستجد فيها القوة والسهولة والمرونة وتوفير الوقت والجهد فالكود الذي ربما تكتبه في ١٠ اسطر ربما تختصره بسطر واحد باستخدام Linq فهذه الحزمة تحتوي على العديد من الكلاسات الجاهزة التي تمكن المبرمج من استخدامها بكل يسر وسهولة فهي ليست صعبة انما سهلة وميسرة وتدعمها لغات Net. بقوة
كي تتعرف على ماهي Linq هي كما أسلفنا جمل استعلام شبيهة بلغة SQL ولكن ليس تشابهآ كبيرآ فهي تختلف من ناحية البنية بعض الشيء
الفرق والجوهر بينها وبين ال SQL هوان SQL مخصصة لجلب البيانات من قاعدة البيانات
اما ال Linq فهي مخصصة لجلب البيانات من بيئة العمل مثل البيانات داخل المصفوفات او Datasetاو XML او المصفوفات او حتى DataGrideView الصورة في المنشور توضح الموارد التي يمكن لحزمة LINQ ان تجلب البيانات منها

وهو عبارة عن استعلام يوفره إطار العمل .NET. ويقوم LINQ بجلب البيانات من موارد بيئة العمل كما تحدثنا في الدرس السابق سنتحدث هنا عن كيفية قراءة البيانات من مصفوفة باستخدام Linq كما تلاحظ تم تعريف مصفوفة تحتوي على كلمات نصية
المطلوب هو قراءة او طباعة كل الكلمات التي تحتوي على حرف s
طبعا جملة linq تعيد كائن لذلك يجب تخزين نتيجة استعلام linq داخل كائن في السطر التالي تم تعريف كائن اسمه myLinqQuery لتخزين النتيجة التي ستعيدها جملة الاستعلام linq داخل هذا المتغير
لاحظ جملة الاستعلام بدأت بالكلمة المحجوزة from التي تقرأ من المصفوفة ثم بعدها تم تعريف كائن اسمه nameلتخزين كل الكلمات من المصفوفة التي اسمها names داخل المتغير name ثم تم استخدام الكلمة المحجوزة where كي يشترط انه فقط يتم قراءة كل الكلمات من المصفوفة التي تحتوي على الحرف a
بعد ان تم تركيب جملة الاستعلام linq تم طباعة النتيجة باستخدام loop من نوع foreach

هيكلية Linq

كما اسلفنا سابقاً ان Linq هو شبيه باستعلام Sql مع تغيير بسيط من ناحية الهيكلية ودائماً نتيجة استعلام Linq يتم تخزينها داخل متغير من نوع Var ويسمى متغير البيانات المجهولة وأيضاً دائماً يتم استخدام كائن مشتقة من الكلمة المحجوزة From وهذا يسمى المدى أو Rang او المخزن المؤقت الذي سيتم تخزين البيانات داخله ومعالجتها ثم ارسال البيانات الى المتغير من نوع var كما وضحنا في البداية لاحظ معنا الشكل التالي

```
Var Rs=from s in sourceData
Where condition
Select s
```

Var Rs	المتغير الذي سيتم تخزين نتيجة الاستعلام داخله
from s	From بداية الاستعلام في linq يبدأ بالكلمة المحجوزة from يليها كائن اسمه s طبعاً اسم الكائن اختياري اما form فهي كلمة محجوزة طبعاً الكائن الذي يلي الكلمة المحجوزة from هو الكائن الذي سيتم تخزين البيانات فيه بشكل مؤقت ومعالجتها ومن ثم ارسالها الى المتغير من نوع var
where	كلمة محجوزة في حال إذا أردنا كتابة شرط معين
condition	وتعني جملة الشرط المعطى في استعلام Linq
Select	تعني تخزين البيانات داخل المتغير الذي تم انشاءه بعد الكلمة المحجوزة form كما تلاحظ انه تم كتابته بعد select مباشرة

انظر الشكل التالي

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

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            string[] Names = { "saad", "Jamal", "Mohammed", "Ali" };
            var MyLinq = from s in Names
                         where s.Contains("s")
                         select s;

            foreach (var name in MyLinq)
                Console.WriteLine(name + " ");
            Console.ReadKey();

        }
    }
}
```

الشكل رقم (٢)

المعاملات في Linq

مثال رقم (٢)

في هذا المثال يتم تعريف list يحتوي على مجموعة من الجمل
المطلوب هو قراءة الجمل فقط التي تحتوي على كلمة Tuturials باستخدام جملة استعلام linq
يتم تعريف متغير من نوع var يعني كائن ليتم تخزين جملة الاستعلام داخل هذا الكائن الذي تم تسميته result
تم استخدام جملة الاستعلام linq مبدوءة بالكلمة المحجوزة from طبعا from يتبعها متغير من نوع كائن هذا المتغير الذي يأتي مباشرة بعد الكلمة المحجوزة form هوالكائن الذي سيتم تخزينالبيانات داخله ثم يرسلها الى المتغيرresult هذا المائة اسمه result بعد ذلك يتم كتابة الكلمة المحجوزة in يعني سيتم قراءة البيانات من ال Listثم يتم تخزينها داخل الكائن s ليتم إرسال النتيجة الى المتغير result ثم بعد ذلك تم كتابة الشرط ان يتم تخزينالجمل التي تحتوي على الكلمة tutorial
ثم تأتي كلمة select او الكلمة المحجوزة select في نهاية الجملة

```
static void Main(string[] args)
{
    List<string> L1 = new List<string>
    {
      "En saad", "D.Jamal", "Mohammed", "Ali"
    };

    var MyLinq = from s in L1
                 where s.Contains("En")
                 select s;

    foreach (var name in MyLinq)
        Console.WriteLine(name + " ");
    Console.ReadKey();

}
```

الشكل رقم (٢)

مثال رقم ٣

يمكنك بناء جملة استعلام من نوع linq بدون استخدام الكلمتين المحجوزة from و select حيث يمكنك مباشرة استخدام اسم المصفوفة او الكائن الخاص ب list ثم اتباعه بكلمة Where مباشرة بشرط تعريف كائن الذي سيتم تخزين النتيجة فيه متبوعة بالعلامة الجبرية => ليشير الى المحتوى الشرطي كما هو موضح في الصورة

```
static void Main(string[] args)
{
    List<string> L1 = new List<string>
    {
      "En saad", "D.Jamal", "Mohammed", "Ali"
    };


    var MyLinq = L1.Where(s => s.Contains("En"));

    foreach (var name in MyLinq)
        Console.WriteLine(name + " ");
    Console.ReadKey();

}
```

الشكل رقم (٣)

مثال رقم (٤) المعامل fType

كما قلنا سابقآ بأن linq تستخدم لقوتها واختصار الوقت والكود ومن واقع تجربة فان استخدام مكاتب ال linq يرفع من جودة البرنامج ويقلل من الأخطاء التي تحدث وقت التشغيل

سنتحدث هنا عن معامل توفره حزمة Linq وهو المعامل OfType يقوم هذا المعامل بتصفية البيانات فمثلآ اذا كان لدي بيانات مختلطة تنتمي الى string و int وأردت الحصول على البيانات التي تنتمي الى النوع string فقط هنا استخدم المعامل OfType وكما هو موضح في الصورة لدي list يحتوي على بيانات نصية وبيانات عددية

في البداية أردت الحصول على البيانات النصية تم استخدام المعامل OfType وتم تعريف النوع String للحصول على البيانات النصية فقط كذلك في السطر الذي يليه تم الحصول على البيانات العدلية فقط من داخل list باستخدام المعامل linq وتم إسناد النوع int للحصول على البيانات العددية فقط

```
    static void Main(string[] args)
{
    IList L1 = new ArrayList();
    L1.Add(0);
    L1.Add("saad");
    L1.Add("Gamal");
    L1.Add(9);



    var Mystring = from s in L1.OfType<string>()
                   select s;

    foreach (var name in Mystring)
        Console.WriteLine(name + " ");


    var MyNumber = from s in L1.OfType<int>()
                   select s;

    foreach (var name in MyNumber)
        Console.WriteLine(name + " ");
    Console.ReadKey();

}
```

الشكل رقم (٤)

مثال رقم ٥ المعامل OrderBy

يستخدم المعامل orderBy لترتيب العناصر أبجدياً

يمكنك استخدام orderBy مع استعلام Linq لترتيب العناصر من الاصغر الى الأكبر حسب الرقم أو أبجدياً حسب ترتيب الحرف انظر الشكل التالي

```
IList L1 = new ArrayList();
L1.Add("Khaled");
L1.Add("saad");
L1.Add("Gamal");
L1.Add("salim");

var Mystring = from s in L1.OfType<string>()
               orderby s
               select s;

foreach (var name in Mystring)
    Console.WriteLine(name + " ");
```

الشكل رقم (٥)

مثال رقم (٦) المعامل Sorting يستخدم المعامل ThenBy لترتيب البيانات حسب قيمة محددة او لترتيب السجلات حسب قيمة محددة وهو شبيه بالمعامل orderBy إلا أن Sorting يستخدم مع البيانات المجمعة

وحتى نستطيع استخدام المعامل ThenBy مع بيانات مجمعة اتبع الخطوات التالية

١- قم ببناء class اسمه student

```
public class Student
{
    public int StudentID { get; set; }
    public string StudentName { get; set; }
    public int Age { get; set; }
}
```

٢-

```
IList<Student> studentList = new List<Student>() {
    new Student() { StudentID = 1, StudentName = "Saad", Deg = 60 } ,
    new Student() { StudentID = 2, StudentName = "Khaled",  Deg = 80 } ,
    new Student() { StudentID = 3, StudentName = "Sami",  Deg = 90 } ,
    new Student() { StudentID = 4, StudentName = "Gamal" , Deg = 30 } ,
    new Student() { StudentID = 5, StudentName = "Ali" , Deg = 98 },
    new Student() { StudentID = 6, StudentName = "Moncif" , Deg = 70 }
};
```

الشكل رقم (٧)

٣- الآن سوف نقوم بترتيب الطلاب حسب الدرجة باستخدام المعامل ThenBy

٤- الآن سوف نقوم ببناء جملة إستعلام Linq بالشكل التالي

```
var SortThan = studentList.OrderBy(s => s.StudentName).ThenBy(s => s.Deg);
```

قمنا بإنشاء كائن اسمه SortThanهذا الكائن سوف يتم تخزين نتيجة إستعلام Linq داخله بعد ذلك استخدمنا الكائن الخاص با IList المسمى Student واستخدمنا المعامل OrderBy بعد ذلك انشأنا كائن اسمه s لتخزين كافة القيم فيه ثم استخدمنا المعامل ThanBy لترتيب السجلات حسب الدرجة Deg

٥- سنقوم الآن بطباعة المخرجات بالشكل التالي

```
foreach (var name in SortThan)
    Console.WriteLine(name.StudentID + " " + name.StudentName + " " + name.Deg);
Console.ReadKey();
```

الشكل رقم (٨)

مثال رقم (٦) المعامل GroupBy

يستخدم المعامل GroupBy في جملة استعلام Linq لتجميع البيانات حسب القيم المتشابهة كما هو الحال في SQL

إعتماداً على المثال السابق يمكننا تجميع السجلات حسب الدرجات المتشابهة بالشكل التالي

```
        static void Main(string[] args)
    {
     IList<Student> studentList = new List<Student>() {
     new Student() { StudentID = 1, StudentName = "Saad", Deg = 60 } ,
     new Student() { StudentID = 2, StudentName = "Khaled",  Deg = 80 } ,
     new Student() { StudentID = 3, StudentName = "Sami",  Deg = 90 } ,
     new Student() { StudentID = 4, StudentName = "Gamal" , Deg = 30 } ,
     new Student() { StudentID = 5, StudentName = "Ali" , Deg = 90 },
     new Student() { StudentID = 6, StudentName = "Moncif" , Deg = 70 }
};

     var GroupList = from s in studentList
                     group s by s.Deg;

     foreach (var name in GroupList)
         Console.WriteLine("{0}", name.Key );
     Console.ReadKey();

    }
```

الشكل رقم (٩)

مثال رقم (٧) المعامل select

تحدثنا عن المعامل select من بداية الكتاب وقلنا انه يأتي من ضمن بناء جملة استعلام Linq يمكننا استخدام المعامل select لإختيار قيمة محدد من السجل في حال إذا كان لدينا سجل يحتوي على عدة قيم ونريد فقط اختيار قيمة

إعتماداً على المثال السابق لنفترض أننا نريد فقط طباعة اسم الطالب من سجل الطلاب لتنفيذ ذلك شاهد الشكل التالي

```
var selectResult = from s in studentList
                   select s.StudentName;
```

الشكل رقم (١٠)

لاحظ انه تم استخدام المعامل select ثم حددنا اسم الطالب s.StudentName ليتم طباعة فقط اسم الطالب

المعاملات المنطقية

مثال رقم (٨) المعامل Average

: يستخدم المعامل Average لإيجاد المعدل العام لمجموعة من القيم

إعتماداً على المثال السابق لنفترض اننا نريد ان نقوم بإيجاد المعدل العام للطلاب كل ماعلينا القيام به هو استخدام كلمة average وتحديد العمود الذي نريد ايجاد المعدل منه وهو Deg عمود الدرجات

```
      static void Main(string[] args)
  {
   IList<Student> studentList = new List<Student>() {
   new Student() { StudentID = 1, StudentName = "Saad", Deg = 60 } ,
   new Student() { StudentID = 2, StudentName = "Khaled",  Deg = 80 } ,
   new Student() { StudentID = 3, StudentName = "Sami",  Deg = 90 } ,
   new Student() { StudentID = 4, StudentName = "Gamal" , Deg = 30 } ,
   new Student() { StudentID = 5, StudentName = "Ali" , Deg = 90 },
   new Student() { StudentID = 6, StudentName = "Moncif" , Deg = 70 }
};

   var selectResult = studentList.Average (s => s.Deg  );


   Console.WriteLine(selectResult);
   Console.ReadKey();
```

الشكل رقم (١١)

لاحظ في المثال السابق تم تحديد استخدام المعامل Average لإيجاد المعدل وتم تحديد العمود Deg

اما في حال إذا كان لديك مصفوفة او List يحتوي على اعداد فقط وليس بيانات مجمعة يمكنك استخدام المعامل Average بدون تحديد اي عمود

مثال رقم (٩) المعامل count : يستخدم المعامل count لإيجاد عدد القيم على سبيل المثال تريد ايجاد عدد الطلاب او عدد الموظفين او عدد الطلاب الناجحين او الراسبين يمكنك استخدام المعامل Count

إعتماداً على المثال السابق نريد ايجاد عدد الطلاب كل ماعلينا هو استخدام المعامل count في جملة استعلام Linq كما هو موضح في الشكل التالي

```
var selectResult = studentList.Count ( );
```

الشكل رقم (١٢)

مثال رقم (١٠) لنفترض اننا نريد ايجاد المعدل العام للطلاب الناجحين فقط هنا نحتاج شرط النجاح وهو ان يكون معدل الطالب اكبر او يساوي ٥٠ لذلك سوف نجري تعديلاً في الشريط في جملة استعلام linq كما هو موضح في الشكل التالي

```
var selectResult = studentList.Count (s=>s.Deg>=5 );
```

الشكل رقم (١٣)

مثال رقم (١١) المعامل Max يقوم بإيجاد اكبر قيمة لمجموعة قيم محددة

إعتماداً على المثال السابق نريد ايجاد اكبر درجة في درجات الطلاب طالما واننا حددنا درجة يجب علينا تحديد عمود الدرجات وهو Deg كما هو في المثال السابق

```
var selectResult = studentList.Max(s=>s.Deg );
```

الشكل رقم (١٤)

مثال رقم (١٢) المعامل Sum : يستخدم المعامل Sum لإيجاد المجموع الكلي لمجموعة من القيم

على سبيل المثال نريد ايجاد مجموع كافة درجات الطلاب طالما واننا حددنا درجة يجب علينا تحديد عمود الدرجات وهو Deg كما هو في المثال السابق

```
var selectResult = studentList.Sum(s=>s.Deg);
```

الشكل رقم (١٥)

Linq To SQl

الأمثلة السابقة كانت مقدمة عن Linq حتى يأخذ القاريء فكرة عن linq وعن قوتها وعن الدوال التي تقدمها linq في هذا الفصل سوف ندخل إلى Linq To Sql

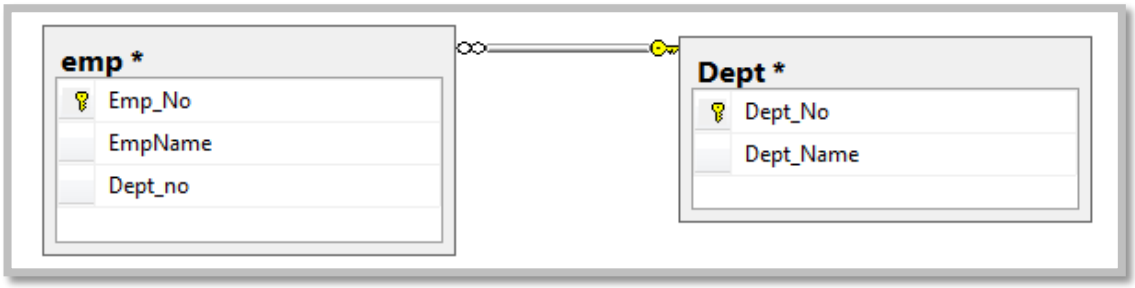
يعد LINQ to SQL أحد مكونات .NET Framework. الإصدار ٣,٥ ويوفر بنية تحتية لوقت التشغيل لإدارة البيانات العلائقية ككائنات. سيسمح لنا LINQ to SQL بالوصول إلى البيانات والحصول عليها من قاعدة بيانات SQL باستخدام استعلامات .LINQ. يسمح لنا بإجراء عمليات تحديد وإدراج وتحديث وحذف في جداول مثل SQL باستخدام استعلامات.LINQ.

في LINQ إلى SQL ، يتم تعيين نموذج بيانات قاعدة البيانات الترابطية إلى نموذج كائن ، وأثناء تنفيذ التطبيق ، يتم تحويل نموذج كائن استعلامات LINQ إلى SQL للحصول على البيانات المطلوبة من قاعدة البيانات ، بينما تقوم بإرجاع البيانات من قاعدة البيانات LINQ إلى SQL بتحويل نتائج SQL إلى نموذج كائنات LINQ.

- ## تكوين Linq To Sql

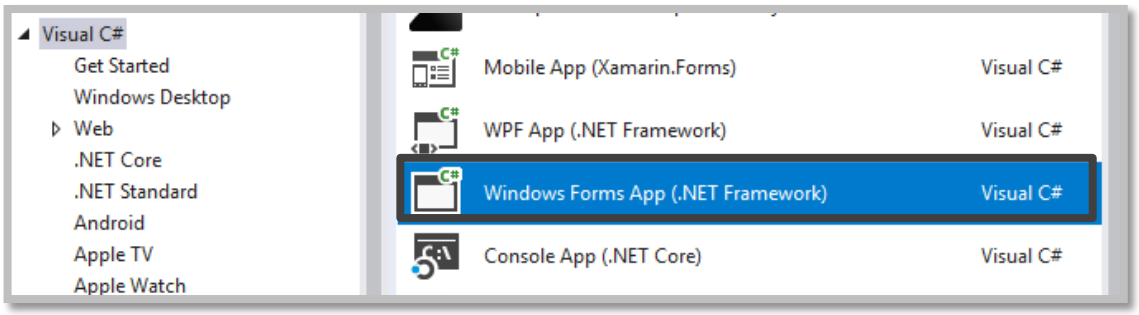
حتى يتضح لنا العمل مع Linq To SQL سنقوم بعمل مثال متسلسل نوضح فيه كيفية ربط البيانات مع Linq To Sql

١- قم بإنشاء جدولين في sql server كالتالي جدول Dept وجدول Emp



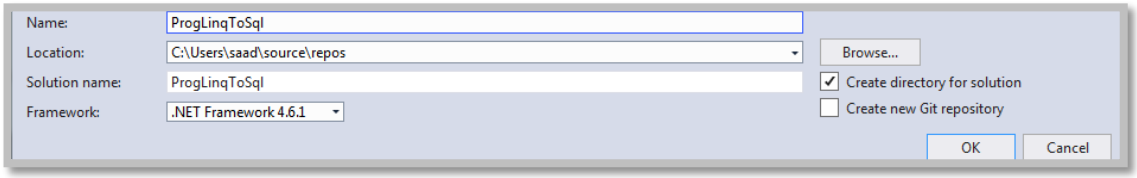
الشكل رقم (١٦)

٢ – افتح مشروع جديد في Visual Studio ثم اختر C# ومن الخيارات اختر windows forms App



الشكل رقم (١٧)

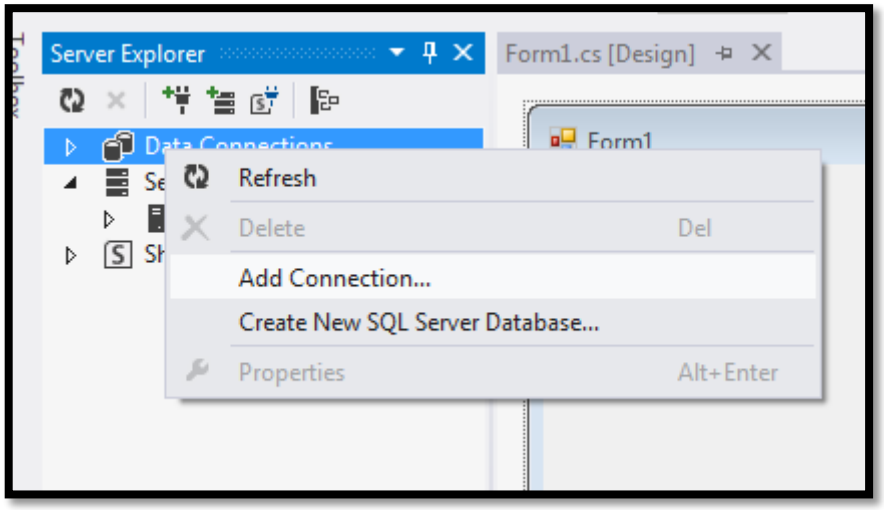
٢- من اسفل الواجهة قم بتسمية المشروع ثم اضغط على الزر ok



الشكل رقم (١٨)

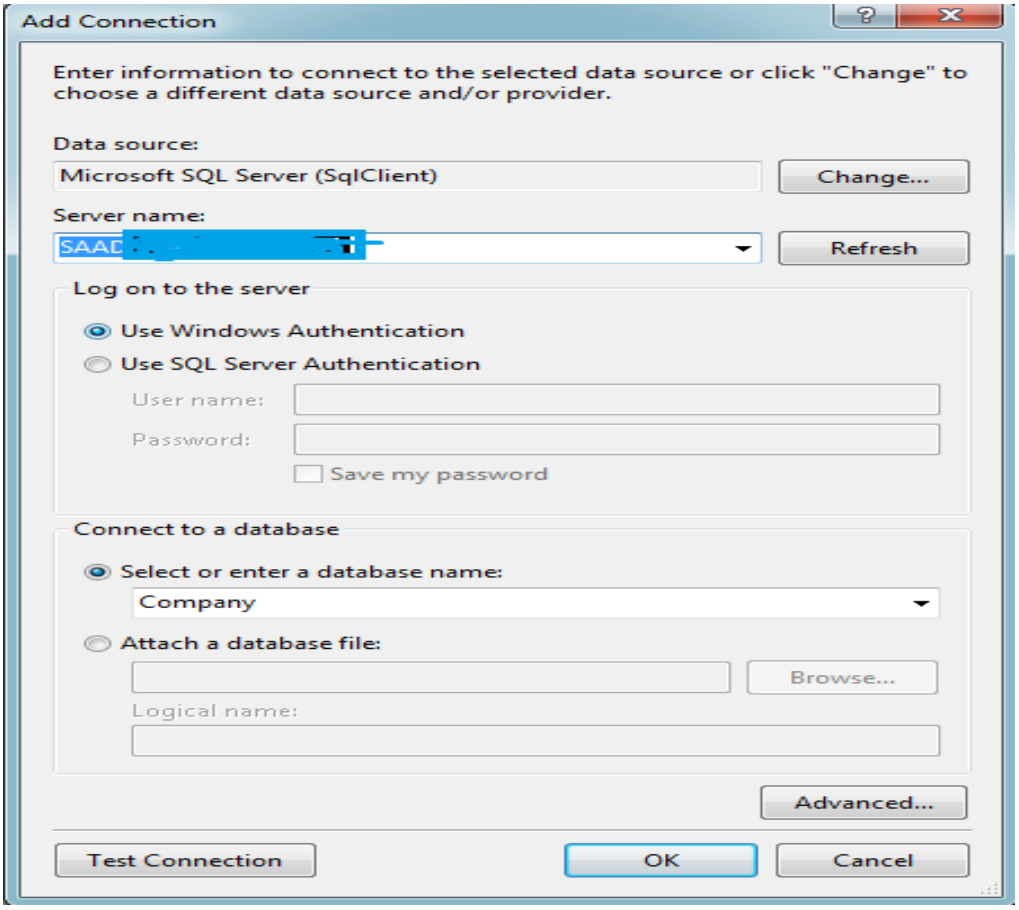
٣- سنقوم الاتصال بقاعدة البيانات للإتصال بالجداول التي قمنا بإنشاءها

٤- من يمين الواجهة اضغط على Data Connection ثم اختر Add Connection…



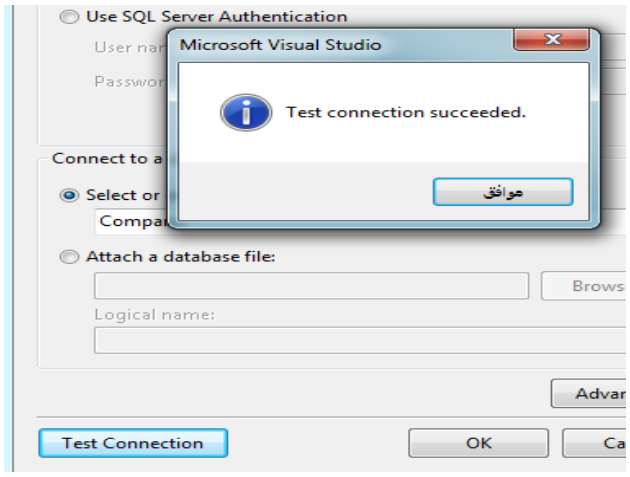
الشكل رقم(١٩)

٥- في الواجهة التالية اكتب اسم السيرفر تحت الخيار Server Name واسم قاعدة البيانات تحت الخيار Select Enter Data Base Name انظر الشكل ٢٠



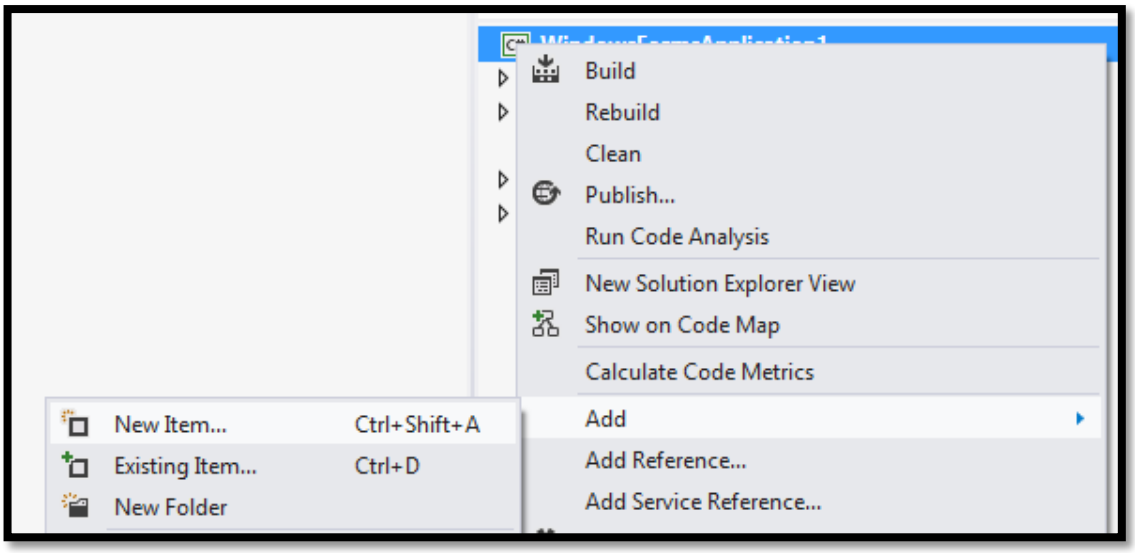
الشكل (٢٠)

٦- قم بالضغط على الزر Test Connectionلإختبار الاتصال ثم اضغط على الزر ok



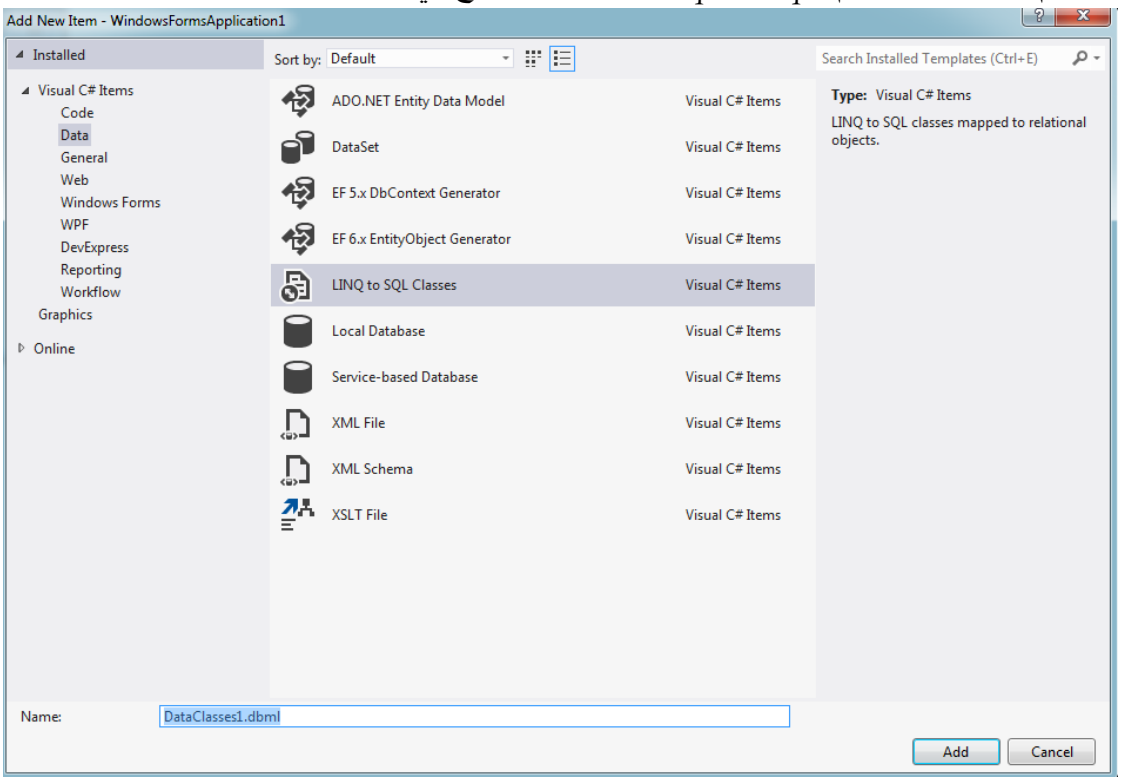
الشكل (٢١)

٧- الآن قم بالضغط على اسم المشروع في يمين الواجهة ثم اختر Add ثم اختر New Item



الشكل (٢٢)

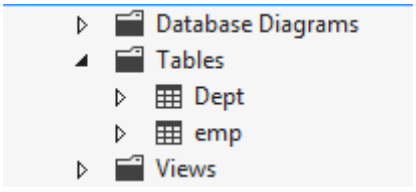
٨- الآن قم باختيار Data ثم Linq To Sql كما هو موضح في الصورة



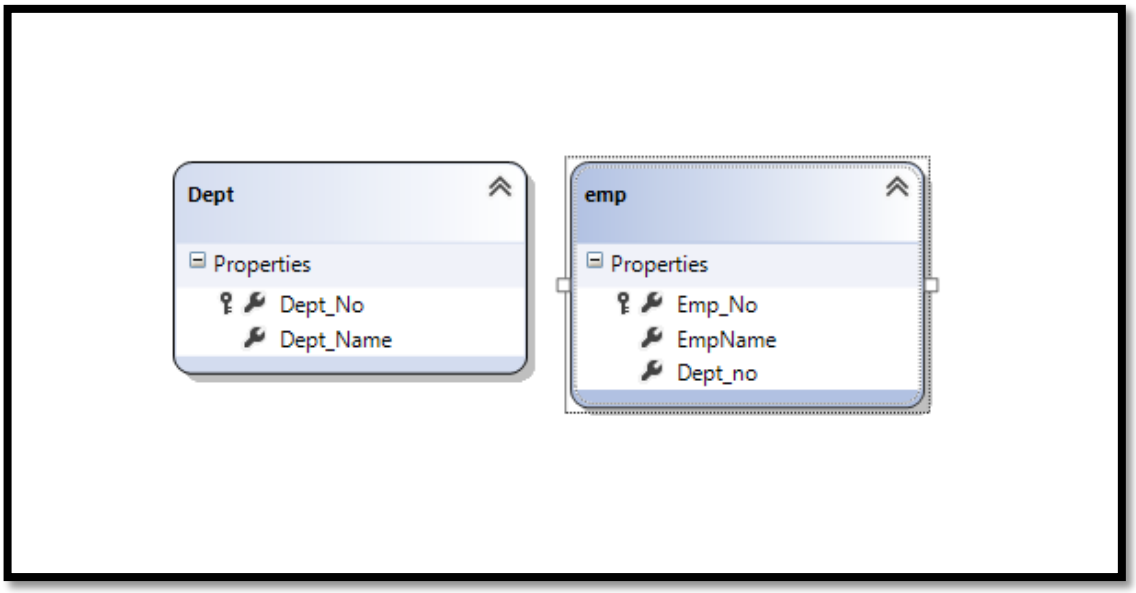
الشكل (٢٣)

٩- ستلاحظ انه اعطاك اسماً للقاعدة DataClasses1.dbml يمكنك تغييره او تركه كما هو

١٠- الآن قم باستعراض الجداول لسحبها الى ساحة العمل

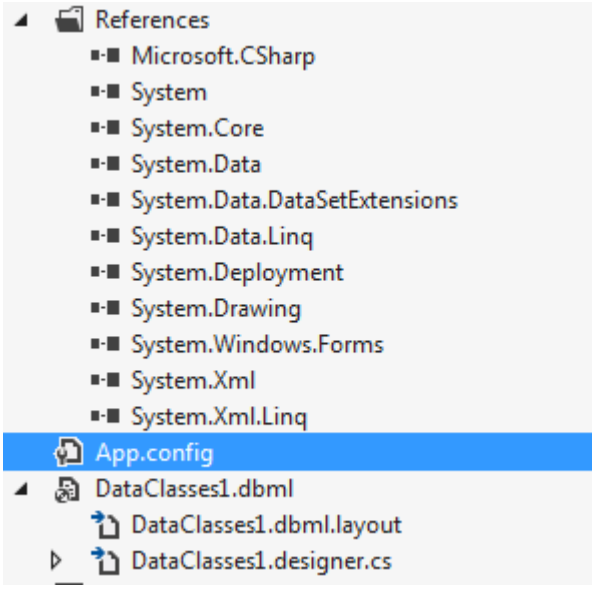


الشكل (٢٤)



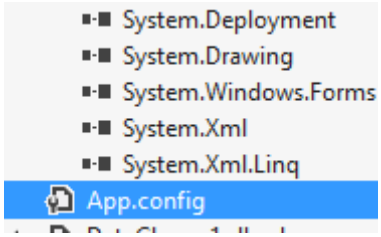
الشكل (٢٥)

١١- ستلاحظ ان visual studio قد قام بتوليد مكتبات وكلاسات جاهزة كي تساعدك على العمل



الشكل (٢٦)

١٢- من ضمن الملفات التي تم توليدها هو الملف App.config وهذا الملف يحتوي على السلسة النصية التي توضح اسم الاتصال واسم السيرفر واسم القاعدة .



الشكل (٢٧)

١٣- الملف Data Class الذي ينتهي بالاسم LayOut يحتوي على الجداول التي قمنا بسحبها إلى ساحة العمل
١٤- الملف DataClass.Designer يحتوي على الكلاسات الجاهزة التي تساعد المبرمج على الاتصال والتخاطب مع قاعدة البيانات بكل سهولة
١٥- بالنزر الى الكلاس DataClass.Designer سنلقي نظرة على اهم الدوال والكلاسات في هذا الملف الذي تم توليده تلقائياً

١- الكلاس `DataClasses1DataContext` وهذا الكلاس الذي يتم استخدامه للإتصال بقاعدة البيانات
٢- ستلاحظ ايضاً انه تم توليد كلاس خاص لكل جدول تم جلبهم الى ساحة العمل كما تلاحظ في الشكل هناك كلاس خاص للجدول Dept اسمه Depts وكلاس خاص بالجدول Emp اسمه Emps

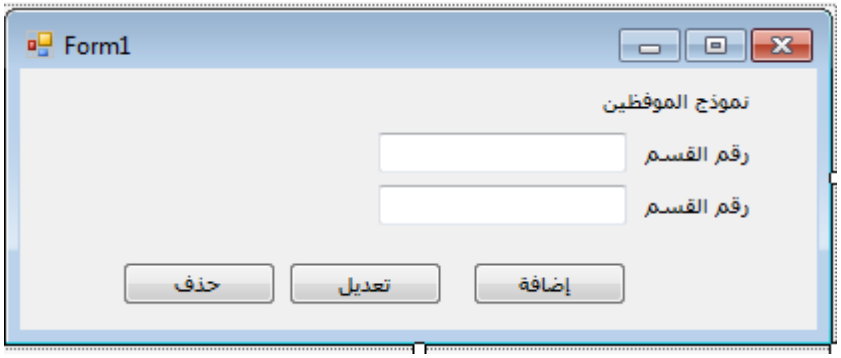
```
public System.Data.Linq.Table<Dept> Depts
{
    get
    {
        return this.GetTable<Dept>();
    }
}

public System.Data.Linq.Table<emp> emps
{
    get
    {
        return this.GetTable<emp>();
    }
}
```

الشكل (٢٨)

سنقوم الآن بتطبيق مثال بسيط لنتعرف على الكلاسات التي يوفرها لنا Linq مثل insert و Update و Delete طبعاً هذه الكلاسات تم بنائها من ADO.Net وأي مبرمج قد جرب البرمجة سابقاً مع ADO.Net سيعرف أن Linq قد وفر عليه الكثير من الجهد والوقت فالأسطر العديدة التي كنا نكتبها من اجل إضافة بيانات مع ADO.Net قد يوفرها Linq لنا في سطر واحد عبر الكلاسات الجاهزة لكن هذا لا يجعلنا أن ننسى أن هذه الكلاسات في الأصل تم بنائها بواسطة ADO.Net وحتى نتعرف على كيفية عمل هذه الكلاسات الجاهزة سنقوم بتطبيق مثال بسيط خطوة بخطوة

مثال (١٣) قم ببناء واجهة في بيئة C# خاصة بإدخال بيانات القسم بالشكل التالي

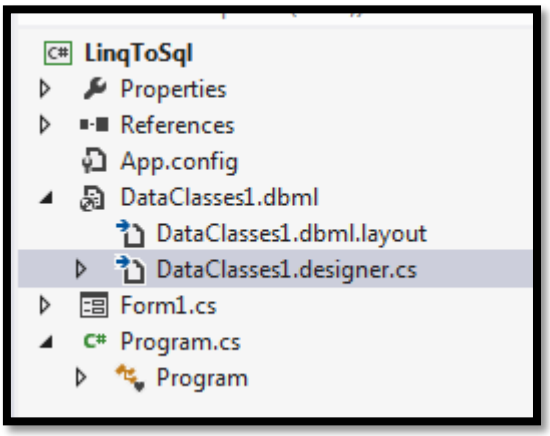


الشكل (٢٩)

من الواجهة السابقة سوف نستطيع ان نتحكم ببيانات جدول القسم من حيث الاضافة والتعديل والحذف

وحتى نتمكن من التحكم بجدول القسم يجب علينا ان نتمكن اولاً من الوصول إلى قاعدة البيانات أولاً وحتى نستطيع الوصول إلى قاعدة البيانات يجب علينا أن نقوم بتضمين كلاس الاتصال بقاعدة البيانات الذي تم توليده تلقائياً في DataClasses.designer.cs والتي أشرنا إليها سابقاً

٣- قم بالقنر على الحزمة DataClasses.designer.cs لفتحها ستجدها موجودة تحت واجهة المشروع



الشكل (٣٠)

٤- قم بنسخ الكلاس `DataClasses1DataContext`

```
[global::System.Data.Linq.Mapping.DatabaseAttribute(Name="Company")]
public partial class DataClasses1DataContext : System.Data.Linq.DataContext
{
```

الشكل (٣١)

٥- الآن إذهب إلى نموذج الأقسام وقم بتضمين الكلاس مع انشاء كائن اسمه db لهذا الكلاس بالشكل التالي

```
public partial class Form1 : Form
{
    DataClasses1DataContext db = new DataClasses1DataContext();
    public Form1()
```

الشكل (٣٢)

٦- الهدف من تضمين الكلاس DataClasses1DataContext هو حتى نستطيع الوصول إلى قاعدة البيانات وهذا يسمى كلاس الإتصال اما الكائن db فهو اسم اختياري يمكنك تسميته باي اسم اخر وقد قمنا بتضمينه في الأعلى حتى يصبح Global أو عام كي نستطيع استخدامه في اكثر من مكان وحتى لا نحتاج إلى تعريفه مرة اخرى في اي اجراء او دالة

٧- الآن سنقوم ببرمجة زر الإضافة insert حتى نستطيع من هذا الزر إضافة البيانات إلى الجدول Dept وحتى نستطيع الوصول الى الجدول Dept لا بد ان نقوم باستخدام الكلاس Dept الذي تم توليده تلقائياً في الملف DataClasses.designer.cs طبعاً كما تلاحظ في الشكل التالي تم تضمين كلاس امسه Dept موجود داخل الملف DataClasses.designer.cs ومن هذا الكلاس سوف نستطيع الوصول الى الجدول Dept والتحكم بالبيانات من حيث الاضافة insert والتعديل update والحذف Delete

```
[global::System.Data.Linq.Mapping.TableAttribute(Name="dbo.Dept")]
public partial class Dept : INotifyPropertyChanging, INotifyPropertyChanged
{

    private static PropertyChangingEventArgs emptyChangingEventArgs = new PropertyChangingEventArgs(String.Empty);

    private int _Dept_No;

    private string _Dept_Name;

    Extensibility Method Definitions

    public Dept()
    {
        OnCreated();
    }
```

الشكل (٣٣)

٨- الأن إذهب إلى حدث الزر اضافة وقم بتضمين الكلاس بالشكل التالي

```
private void button1_Click(object sender, EventArgs e)
{
    Dept dept = new Dept();
}
```

الشكل (٣٤)

كما تلاحظ في الشكل ٣٤ تم تضمين الكلاس وإعطاءه كائن اسمه dept

٩- الأن سوف نقوم ببناء امر ادخال البيانات من مربعات النص الى داخل الجدول باستخدام الكائن dept طبعاً عن طريق الكائن Dept سوف نستطيع الوصول الى حقول الجدول dept ومن خلال ذلك سوف نقوم باضافة البيانات الى الجدول عن طريق امر الإضافة انظر الشكل التالي

```
private void button1_Click(object sender, EventArgs e)
{
    Dept dept = new Dept();
    dept.Dept_No = int.Parse(textBox1.Text);  // اضافة رقم القسم
    dept.Dept_Name = textBox2.Text; // اضافة اسم القسم

}
```

الشكل (٣٥)

كما تلاحظ في الشكل السابق بإسناد القيم من مربعات النصوص Texts Box إلى حقول الجدول

الآن حتى نستطيع تنفيذها وارسال القيم الى قواعد البيانات لا بد ان نستخدم الكلاس InsertOnSubmit طبعاً هذا الكلاس هو متوفر داخل DataClasses1DataContext التي قمنا بتعريفها في بداية التمرين في بشكل Global واعطيناها كائن اسمه db وإعتماداً على الكائن db سوف نستخدم الكلاس InsertOnSubmit ليصبح شكل الامر بالشكل التالي

```
db.Depts.InsertOnSubmit(dept);
```

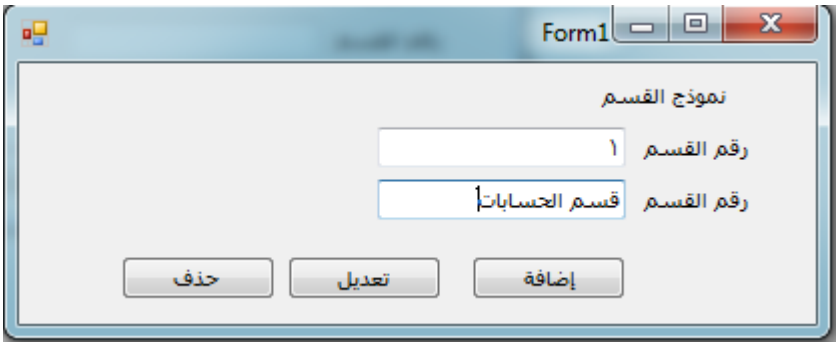
وأخيراً سوف نستخدم امر التنفيذ النهائي SubmitChanges وهذا الامر متوفر داخل الكائن db أيضاً وهذا الامر مهمهمته تنفيذ الاوامر السابقة ليصبح شكل الاوامر النهائي بالشكل التالي انظر الشكل (٣٦)

```
private void button1_Click(object sender, EventArgs e)
{
    Dept dept = new Dept();
    dept.Dept_No = int.Parse(textBox1.Text);  // اضافة رقم القسم
    dept.Dept_Name = textBox2.Text; // اضافة اسم القسم
    db.Depts.InsertOnSubmit(dept);
    db.SubmitChanges();
}
```

الشكل (٣٦)

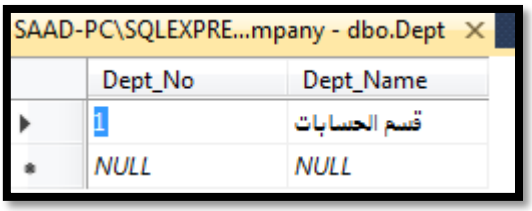
بعد ذلك تأتي الآن مرحلة التجربة

سنقوم بتشغيل البرنامج ثم ندخل بيانات للتجربة بعد ذلك نضغط على الزر **إضافة**



الشكل (٣٧)

بعد الضغط على الزر **اضافة** ستلاحظ ان البيانات قد تم اضافاتها الى الجدول



الشكل (٣٨)

وبهذا نكون قد اخذنا فكرة واضحة عن كيفية الاتصال بقاعدة البيانات عن طريق Linq وكيفية الاتصال بالقاعدة وكيفية الوصول الى الجدول وارسال البيانات اليه

سنقوم الآن بالتعرف على كيفية التعديل في بيانات الجدول وبرمجة زر تعديل

طبعاً في حالة التعديل سنقوم بتعديل بسيط على كود الإضافة وسيتم حذف السطر db.Depts.InsertOnSubmit(dept); لأنه سطر إضافة الى الجدول وسنكتفي ببقية الأسطر كما هي انظر الشكل التالي

```
private void button2_Click(object sender, EventArgs e)
{
    Dept dept = new Dept();
    dept.Dept_No = int.Parse(textBox1.Text);  // اضافة رقم القسم
    dept.Dept_Name = textBox2.Text; // اضافة اسم القسم
    db.SubmitChanges();
}
```

الشكل (٣٩)

يجب التركيز على السطر الأخير للتذكير فقط الكائن db هو الذي تم تعريفه كـ Global في أعلى البرنامج وهو المسؤول عن الاتصال بقاعدة البيانات وهو الذي يتحكم بتنفيذ الاوامر في نهاية كل إجراء

أخيراً سنقوم الأن ببرمجة زر **الحذف**

حتى نستطيع حذف اي سجل من قاعدة البيانات يجب تحديد قيمة من السجل نفسه على سبيل المثال يتم حذف السجل عندما نعطيه رقم السجل او اسم السجل ليتم حذفه كلا الامرين صحيح لكن قد يتكرر الاسم فنقع في خطأ ريما نحذف اكثر من سجل تمحمل نفس الاسم لكن الحذف بطريقة امنة يكون حسب رقم السجل المفتاح الرئيسي لذلك سوف نعطيه رقم السجل او رقم القسم حتى تستطيع الحذف بأمان

الخطوة الأولى للحذف سنقوم باشتقاق كائن للحذف اسمه dept بالطريقة التالية كما نفعل في كل مرة

Dept dept = new Dept();

الخطوة الثانية وهي خطوة مهمة حيث اننا نريد ان نصل إلى السجل من داخل قاعدة البيانات وطالما اننا نريد ان نصل الى قاعدة البيانات يجب علينا ان نستخدم الكائن المسؤول عن الاتصال بقاعدة البيانات وهو db وعن طريق الكائن db سوف نستدعي الكلاس الخاص بالجدول dept وهو Depts كما شرحنا في السابق بعد ذلك سوف سوف نستخدم الدالة single وهذه الدالة تقوم بتحديد سجل واحد باستخدام شرط والشرط سوف يكون هي القيمة الموجودة في مربع النص الخاص برقم القسم سووف يكون الشرط كالتالي

dept = db.Depts.Single(x => x.Dept_No == int.Parse(textBox1.Text));

لاحظ اننا انشأنا كائن اسمه x ليشير الى الجدول dept ثم قمنا بتحديد رقم القسم وحددنا الشرط عندما يساوي القيمة الموجودة في مربع رقم القسم

بعد ان حددنا الشرط سوف نستخدم الأمر أو الكلاس DeleteOnSubmit الذي سيتولى أمر الحذف

واخيراً سوف نقوم بتنفيذ الاوامر باستخدام الكائن db الخاص بالاتصال بقاعدة البيانات ليصبح الشكل النهائي لأمر الحذف كالتالي

```
private void button3_Click(object sender, EventArgs e)
{
    Dept dept = new Dept();
    dept = db.Depts.Single(x => x.Dept_No == int.Parse(textBox1.Text));
    db.Depts.DeleteOnSubmit(dept);
    db.SubmitChanges();
}
```

الشكل (٤٠)

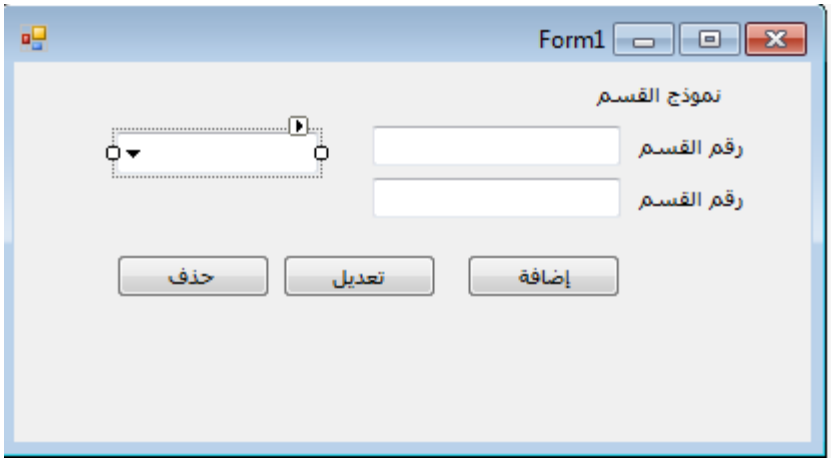
- جلب البيانات من قاعدة البيانات بواسطة linq

يمكن جلب البيانات بكل يسر وسهولة من قاعدة البيانات بواسطة linq وتطبيق استعلام linq ووضع الشروط بكل سهولة وللتعرف على ذلك سنقوم بتطبيق مثال بسيط جداً وهو كيفية قراءة البيانات من جدول Dept من قاعدة البيانات وكيفية تعبئة combobox من البيانات التي تم قراءتها

طبعاً لقراءة البيانات من قاعدة البيانات علينا اولا استخدام كائن الاتصال db المشتق من كلاس الاتصال `DataClasses1DataContext` وبعد ذلك يتم تحديد الجدول الذي نريد القراءة منه

- مثال

نريد تعبئة ComboBox من الجدول Dept علينا في هذه الحالة جلب البيانات ولا نستطيع جلب البيانات إلا باستخدام كائن مشتق من الكلاس `DataClasses1DataContext`
طبعاً نحن اشتققنا كائن اسمه db من الكلاس `DataClasses1DataContext`
ولتوضيح ذلك سوف نقوم بتطبق مثال بسيط
قم بإدراج ComboBox الى نموذج الأقسام كما هو موضح في الشكل (٤١)



الشكل (٤١)

- بعد ذلك اذهب الى الحدث Load وذلك بالنقر المزدوج على انفس النموذج
- في الحدث Load اكتب الكود التالي

```
private void Form1_Load(object sender, EventArgs e)
{

    comboBox1.DisplayMember = "Dept_Name";
     comboBox1.ValueMember = "Dept_No";
     comboBox1.DataSource = db.Depts;

}
```

الشكل (٤٢)

مع ان السطر الأول والثاني لا يدخل ضمن اختصاص Linq لكن ستقوم بشرحها

السطر الأول يستخدم الدالة `combobox1.DisplayMemper` لإظهار اسم القسم في combobox

الشطر الثاني ويستخدم الدالة `combobox.ValueMember` لتكون القيمة الحقيقية التي يتم ارجاعها هي رقم الصف عند اختيار قيمة من ال combobox

السطر الأخير وهو المهم قمنا بجلب البيانات باستخدام linq وذلك باستخدام الكائن db المشتق من الكلاس DataClasses1DataContext وعن طريق الكائن db تم الوصول الى الكلاس Depts الذي يمكننا من الوصول الى الجدول Dept الآن بمجرد تشغيل النموذج ستلاحظ انه قد تم تعبئة النموذج بالبيانات التي تم جلبها من الجدول dept

```
private void Form1_Load(object sender, EventArgs e)
{

    comboBox1.DisplayMember = "Dept_Name";
     comboBox1.ValueMember = "Dept_No";
     comboBox1.DataSource = db.Depts;

}
```

الشكل (٤٣)

- **جلب البيانات من قاعدة البيانات بواسطة linq بواسطة شرط معطى**

يمكننا جلب البيانات من قاعدة البيانات حسب شرط معطى بواسطة استعلام Linq على سبيل المثال نريد جلب الاقسام التي ارقامها اكبر من واحد في هذه الحالة سوف نستخدم استعلام linq لكتابة الشرط ليصبح الكود بالشكل التالي

```
private void Form1_Load(object sender, EventArgs e)
{

    comboBox1.DisplayMember = "Dept_Name";
     comboBox1.ValueMember = "Dept_No";
     comboBox1.DataSource = from dept in db.Depts
                            where dept.Dept_No > 1
                            select dept;
}
```

الشكل (٤٤)

تم استخدام استعلام linq لتحقيق الشرط وهو فقط جلب البيانات من جدول dept حسب الشرط المعطى وهو جلب الاقسام التي ارقامها اكبر من 1 وتم صياغة الشرط بالشكل التالي

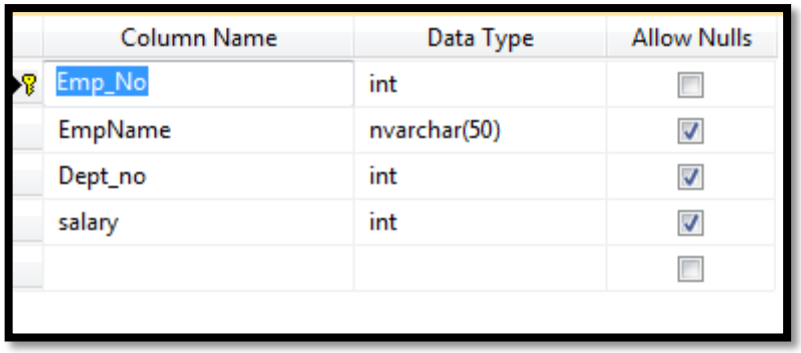
```
where dept.Dept_No > 1
select dept;
```

مثال تطبيقي

سنقوم الآن بتطبيق كل ماتعلمناه على جدول الموظفين لتطبيق المثال اتبع الخطوات التالية

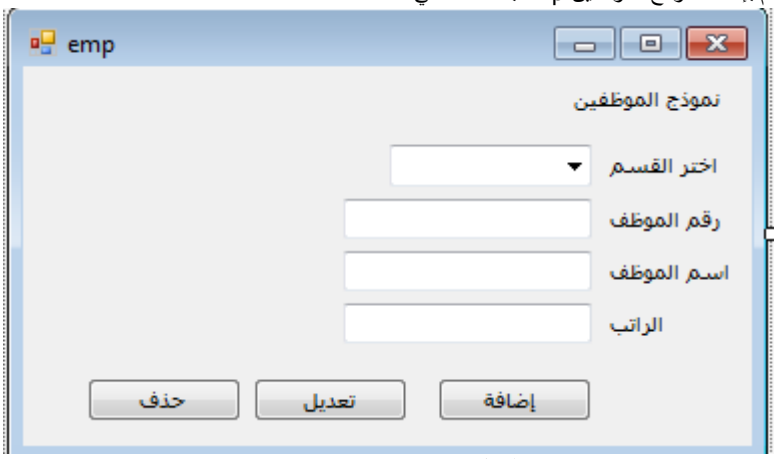
١- إذهب الى جدول الموظفين واضغط على الجدول بالزر الأيمن ثم اختر Design

٢- قم بإضافة حقل اسمه Salary



الشكل (٤٥)

٣- قم بإنشاء نموذج الموظفين Emp بالشكل التالي



الشكل (٤٦)

٤- الآن قم بتضمين كلاس الاتصال بشكل عام تحت اسم الكلاس الرئيسي emp

```
public partial class emp : Form
{
    DataClasses1DataContext db = new DataClasses1DataContext();
```

الشكل (٤٧)

٥- سنقوم الآن بتعبئة comboBox من جدول الأقسام ، انقر نقرتين مزدوجة على نفس النموذج للذهاب الى الحدث Load

ثم اكتب الكود التالي الذي كتبناه مسبقاً في جدول الأقسام وتم شرحه انظر الشكل (٤٧)

الشكل (٤٧)

٦- الأن سنقوم ببرمجة الزر **إضافة** وفي هذ الخطوة سوف نستخدم الكلاس طبعاً نحن قد شرحنا أين يقع هذا الكلاس ان كنت قد نسيت عليك بالعودة لقراءة الكتاب مرة أخرى
سنوقم الأن بتضمين كائن اسمه emp1 من الكلاس emp

```
emp emp1 = new emp();
```

٧- سنقوم الأن بإسناد البينات الى حقول الجدول emp بالشكل التالي مع امر الحفظ والتنفيذ لتصبح كافة الاوامر بالشكل التالي

```
private void button1_Click(object sender, EventArgs e)
{
    employee emp1 = new employee();
    emp1.Dept_no =int.Parse (comboBox1.SelectedValue.ToString());  // رقم القسم
    emp1.Emp_No = int.Parse(txtEmpNo.Text);   // رقم الموظف
    emp1.EmpName = TxtEMpName.Text;  // اسم الموظف
    emp1.salary = TxtEmpSalary.Text; // راتب الموظف
    db.emps.InsertOnSubmit(emp1);  // حفظ البيانات الى الجدول
    db.SubmitChanges();  // تنفيذ عملية الحفظ

}
```

الشكل (٤٨)

٨- بعد ذلك سنقوم ببرمجة زر التعديل الذي لا يختلف كثيراً عن امر الحفظ فقط سنقوم بحذف سطر الإدخال لأننا عند التعديل نقوم بالتعديل على البيانات حسب الشرط ولا نحتاج الى ادخال

```
db.emps.InsertOnSubmit(emp1);  // الجدول الى البيانات حفظ
```

بعد حذف سطر الادخال ستصبح الاوامر بالشكل التالي

```
private void button1_Click(object sender, EventArgs e)
{
    emp emp1 = new emp();
    emp1.Dept_no =int.Parse (comboBox1.SelectedValue.ToString());  // رقم القسم
    emp1.Emp_No = int.Parse(txtEmpNo.Text);   // رقم الموظف
    emp1.EmpName = TxtEMpName.Text;  // اسم الموظف
    emp1.salary =int.Parse(TxtEmpSalary.Text); // راتب الموظف
    db.emps.InsertOnSubmit(emp1);  // حفظ البيانات الى الجدول
    db.SubmitChanges();  // تنفيذ عملية الحفظ

}
```

الشكل (٤٩)

٩- سننتقل الأن إلى برمجة الزر حذف وقد شرحناه في الدرس السابق وهو كالتالي

```
private void button2_Click(object sender, EventArgs e)
{
    emp emp1 = new emp();
    emp1.Dept_no = int.Parse(comboBox1.SelectedValue.ToString());  // رقم القسم
    emp1.Emp_No = int.Parse(txtEmpNo.Text);   // رقم الموظف
    emp1.EmpName = TxtEMpName.Text;  // اسم الموظف
    emp1.salary = int.Parse(TxtEmpSalary.Text); // راتب الموظف

    db.SubmitChanges();  // تنفيذ عملية الحفظ
}
```

الشكل (٥٠)

متابعة مع اوامر الزر نجد اننا قد اشتقنا كائن اسمه emp1 من الكلاس emp

```
emp emp1 = new emp();
```

في السطر الثاني قمنا ببناء الشرط المطلوب للحذف وهو تحديد السجل الذي نريد حذفه ومن البديهي ان يكون السجل هو رقم الموظف لذلك حددنا مربع النص المسمى `txtEmpNo.Text` في نفس الشرط وتم بناء الشرط بالشكال التالي

```
emp1 = db.emps.Single(x => x._Emp_No == int.Parse(txtEmpNo.Text));
```

الجزء الأيسر من السطر `emp1 = db.emps.Single` يتم تضمين الكلاس `emps` من كائن الاتصال `db` ثم تم تضمين الدالة `single` لتشير الى سجل واحد فقط طبعاً الدالة `single` تطلب شرط واحد فقط وهو تحديد قيمة في السجل المحدد

سننتقل الأن الى جلب البيانات بواسطة استعلامات Linq وعرضها في الفورم

١- قم بإضافة DataGrideView إلى النموذج ليصبح بالشكل التالي

الشكل (٥١)

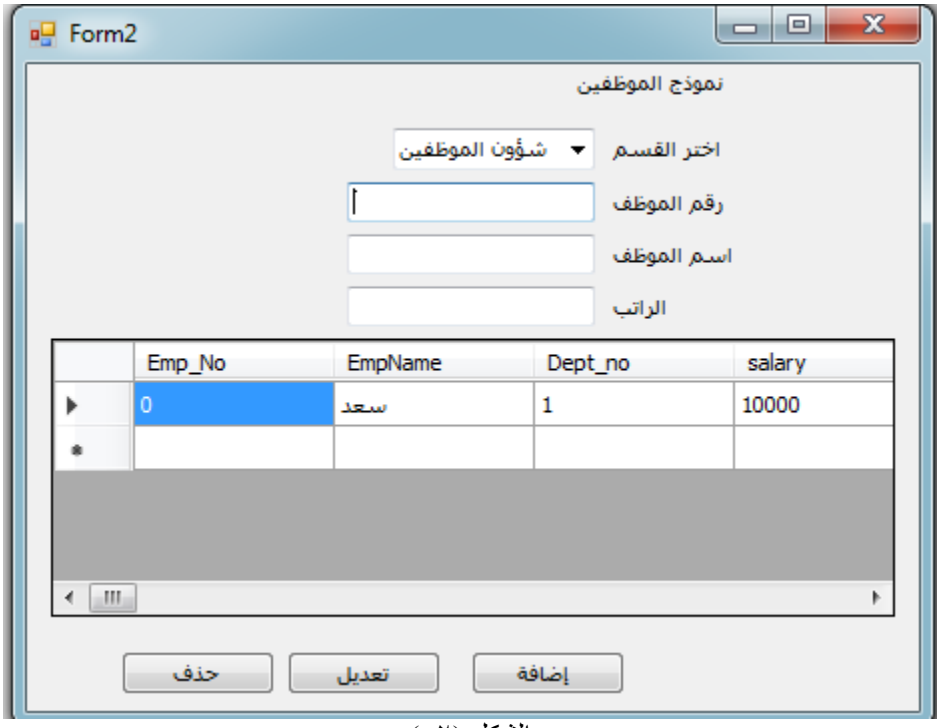
٢- سنقوم الآن بجلب البيانات وعرضها داخل DataGrideView بواسطة Linq

٣- إذهب إلى حدث `Form2_Load` واكتب الكود التالي

```
dataGridView1.DataSource = from emp in db.emps
                           select emp ;
```

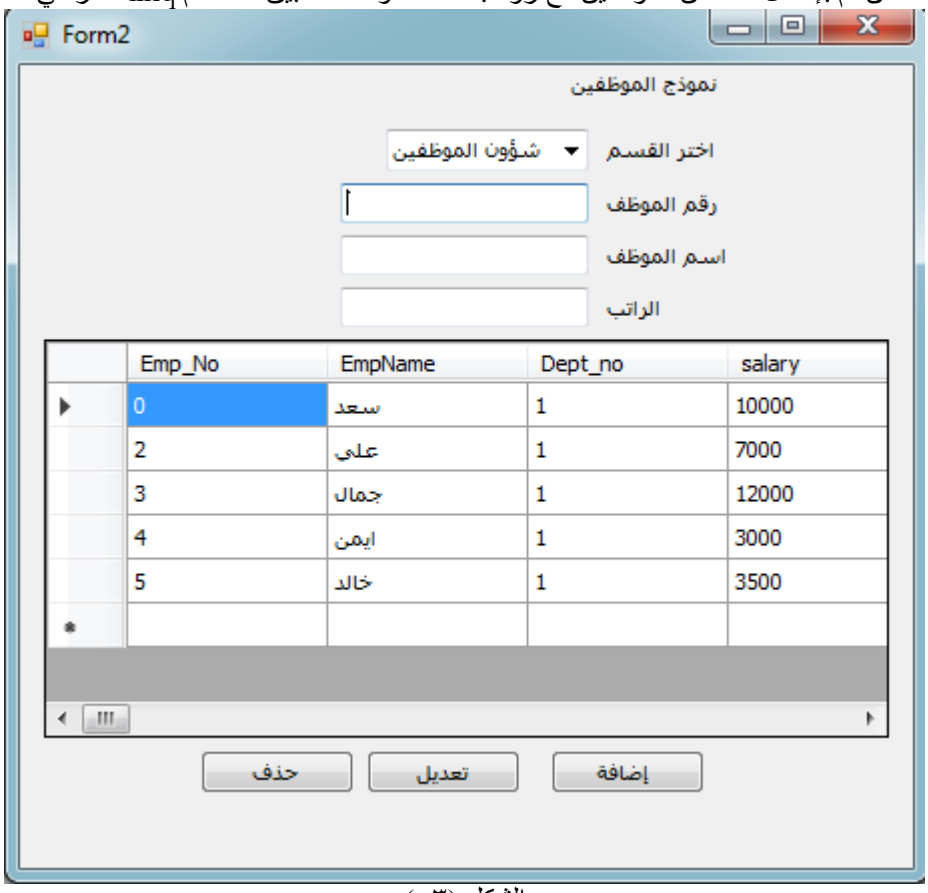
كما تلاحظ فقد قمنا بتعبئة DataGrideView من البيانات التي تم جلبها من جدول الموظفين بواسطة استعلام Linq ثم اسندناها الى DataGrideView بواسطة الخاصية DataSourc

٤- الان قم بعرض النموذج ستلاحظ انه قد تم عرض بيانات الموظفين الذين قمت بإدخالهم



الشكل (٥٢)

٥- الآن قم بإدخال عدد من الموظفين مع رواتب مختلفة وذلك لتطبيق استعلام linq الشرطي



الشكل (٥٣)

٦- سنقوم الان بتطبيق بعض الامثلة على الاستعلام الشرطي بواسطة linq

مثال إظهار بيانات الموظفين فقط الذين رواتبهم اكبر من ٥٠٠٠

لإظهار الموظفين فقط الذين تزيد رواتبهم عن ٥٠٠٠ سنوقم بتعديل بسيط على استعلام linq ليصبح بالشكل التالي

سوف نقوم بتحديد الحقل salary لتحديد الشرط في استعلام linq

```
//------------------------------------------ ن
dataGridView1.DataSource = from emp in db.emps
                           where emp.salary>5000
                           select emp ;
```

الشكل (٥٤)

مثال : إظهار البيانات مرتبة حسب الراتب من الاكبر إلى الأصغر

لتطبيق هذا المثال سوف نقوم بااستخدام المعامل OrderBy التي تناولناها في بداية الكتاب وبما أننا نريد نريد الترتيب حسب حقل الراتب فإننا سوف نقوم بتحديد الحقل salary بعد استخدام المعامل Orderby وسيكون شكل الاستعلام بالشكال التالي

```
dataGridView1.DataSource = from emp in db.emps
                           orderby  emp.salary
                           select emp;
```

الشكل (٥٥)

مثال : جمع عمود في DataGrideView باستخدام Linq

نستطيع باستخدام DataGrideView جمع عمود معين بكل سهولة وبسطر واحد حيث يزودنا Linq بالعديد من **الدوال** التي تسهل على المستخدم العمل مع DataGrideView على سبيل المثال إذا اردنا جمع عمود Salary في DataGrideView واظهار ناتج الجمع في **مربع نص** يمكننا ذلك باستخدام الدالة **sum**

سنقوم بتطبيق هذا المثال بالخطوات التالية

١- قم بإضافة مربع نص إلى النموذج بالشكل التالي

الشكل (56)

٢- قم بتسمية مربع النص بالإسم txtSumSalary

٣- قم بتعريف متغير من نوع Double وليكن اسمه x1
٤- الأن سوف نقوم بتضمين الدالة sum داخل الحدث Load في نموذج الموظفين بالشكل التالي

```
double x1;
x1 = dataGridView1.Rows.Cast<DataGridViewRow>() // Total Invoic
    .Sum(t => Convert.ToDouble(t.Cells[3].Value));
txtSumSalary.Text = x1.ToString();
```

الشكل (٥٧)

٥- تم تحديد الخلية رقم ٣ في DataGrideView لأن عمود Salary ترتيبه ٣
٦- الأن قم بعرض النموذج ستجد انه تم جمع رواتب الموظفين وتم اظهارها في مربع النص