

Multilayer Database Access using Sql Server and C#

الوصول للبيانات متعدد الطبقات بواسطة **Sql Server** و **C#**

طارق جهاد الجبواي

2013

tareqjehad@yahoo.com

المحتويات:

1. تصميم طبقة الوصول للبيانات Data Access Layer

1. إنشاء قاعدة البيانات
2. إنشاء مشروع جديد
3. إنشاء اتصال مع قاعدة البيانات
4. صناعة طبقة الوصول للبيانات يدوياً
5. صناعة طبقة الوصول للبيانات بواسطة DataSet
6. صناعة طبقة الوصول للبيانات بواسطة Linq To Sql Classes
7. صناعة طبقة الوصول للبيانات بواسطة ADO.NET Entity Framework Model

2. تصميم طبقة القواعد والضوابط Business Logic Layer

3. تصميم طبقة العرض Presentation Layer

تحدثنا في المقالة الماضية حول منهجية العمل في تصميم تطبيقات قواعد البيانات، وبيئاً الطبقات الثلاث التي يجب علينا العمل عليها لإتمام المشروع المراد بكل سهولة ويسر. وذكرنا أيضاً أهمية الفصل الطبقي هذا في إعادة الاستخدام لاحقاً أو في الاستخدام وهذا ما سنشاهده في هذا الجزء الذي سيتناول الجزء العملي، وطريقة العمل على جميع الطبقات، وسيكون التطبيق على تقنيات ميكروسوفت NET Platform، وسنستخدم قاعدة بيانات Northwind المعروفة عالمياً، لكي لا نضطر لصنع قاعدة بيانات من الصفر، كون المقالة تركز على منهجية التطبيق والاستخدام وليس صناعة قواعد البيانات، وسنحاول تغطية أكبر قدر ممكن من المفاهيم بإذن الله تعالى.

1. تصميم طبقة الوصول للبيانات Data Access Layer

في الواقع هناك الكثير من الطرق أو التقنيات التي نستطيع بواسطتها تعريف طبقة الوصول للبيانات كما تحدثنا في المقالة الماضية، سأقوم هنا بعرض الطرق الممكنة لصناعة هذه الطبقة، سنقوم بوضع كل طبقة في مشروع مستقل لكي لا نخلط المفاهيم مع بعضها. طبعاً سأقوم بشرح بسيط عن كل تقنية لكي نفهم الفرق بينها وبين التقنيات الأخرى، لكي تحصل لدينا فكرة عامة، وبالتالي نستطيع تحديد نوع التقنية التي سنستخدمها في مشاريعنا حسب الحاجة. سنبدأ أولاً بتجهيز قاعدة البيانات Northwind على Sql Server ثم نجهز الاتصال مع قاعدة البيانات، وبالتالي نستطيع الاستفادة منها في تطبيق باقي المشروع.

1. إنشاء قاعدة البيانات

سنقوم باستخدام قاعدة بيانات Northwind المعروفة عالمياً، ولإنشاء هذه القاعدة سنقوم بتوليدها من الـ Script الخاص بها، ستجد ملف في المرفقات اسمه Northwin.sql نحتاج إلى تشغيله على Sql Server للحصول على قاعدة البيانات كاملة. لذلك سنقوم بفتح الـ Sql Server، وننتصل مثلاً مع SQLEXPRESS أو localhost، بعدها نقوم بالضغط بالزر اليمين على إسم السيرفر في Object Explorer ونعمل New Query.

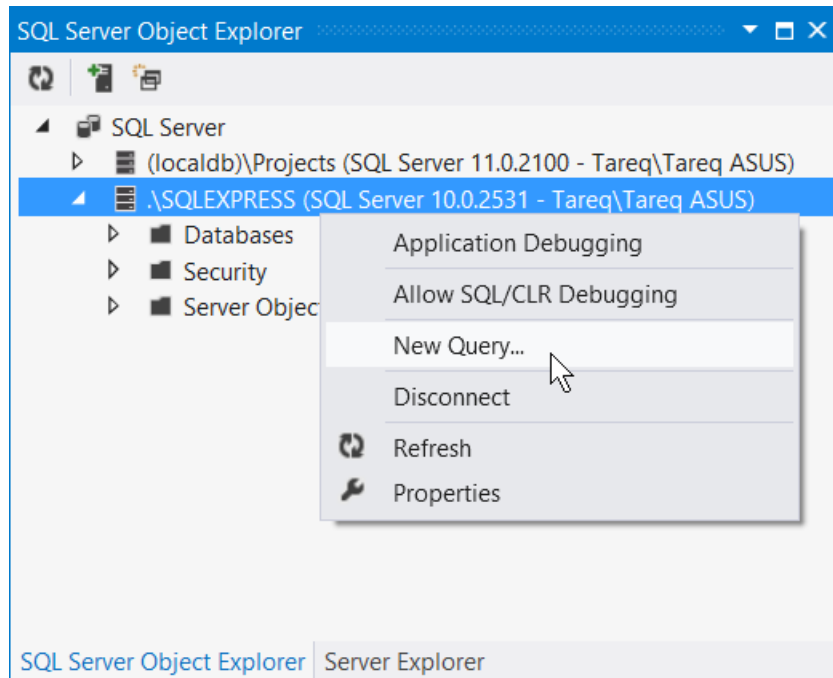


Figure1 : Create a new query to generate the database

الآن نقوم بنسخ محتوى الملف Northwind.sql ونلصقه داخل الـ Query، بعدها نقوم بعمل تنفيذ (F5). حيث سيقوم السيرفر بتوليد قاعدة البيانات Northwind سيخبرنا السيرفر أن العمل تم بوجود بعض الأخطاء، ليست هناك أي مشكلة، كل ما علينا الآن هو تحديث مجلد Databases لنجد أنه تم إضافة قاعدة البيانات Northwind إليها بنجاح، بالطبع تحوي هذه القاعدة على الكثير من Tables, Views, Stored Procedure بالإضافة لوجود العلاقات بين الجداول، وهذا السيناريو مشابه لما نقوم ببنائه في جميع تطبيقاتنا. وتعتبر قاعدة البيانات هذه مناسبة للعمل، خصوصاً في الأهداف التعليمية.

2. إنشاء مشروع جديد

الخطوة التالية هي إنشاء مشروع جديد، بالطبع هنا يكون الخيار للمستخدم في إنشاء مشروعه على البيئة التي تناسبه، لذلك سنقوم بالعمل على Visual Studio، سنقوم أولاً بإنشاء Solution فارغ ثم ننتقل لإضافة المشاريع عليه حسب الحاجة. كوني تحدثت عن إمكانية إعادة الاستخدام لهذه الطبقات، لذلك سأقوم بعمل كل طبقة في مشروعه الخاص، بحيث تكون مفصولة عن الطبقات الأخرى، وهذا الأمر هو من أهم مفاهيم إعادة الاستخدام Reusable، فالقاعدة تقول كلما قلت اعتمادية أو ارتباط عنصر على عنصر آخر، كلما زادت الفرصة في إعادة استخدامه، لذلك نحن هنا، في هذا العمل.

نقوم بفتح الـ Visual Studio، ثم نقوم بإضافة مشروع جديد فارغ، وذلك بالذهاب إلى:

File → New → Project → Other Project Type → Visual Studio Solutions

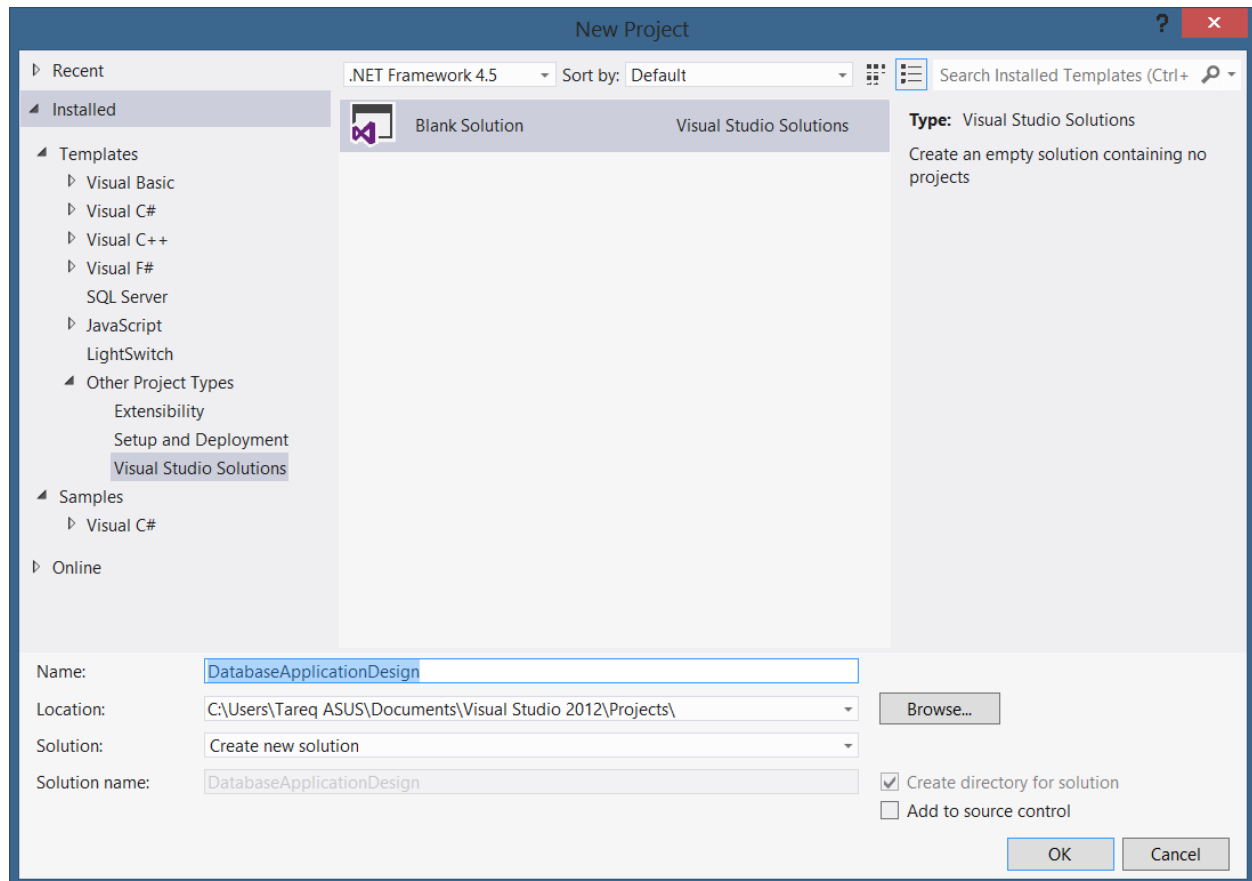


Figure2 : Create a new solution with Visual Studio

بعد تسمية المشروع، ووضعه في مسار مناسب، نضغط على موافق ليتولد لدينا Solution فارغ لا يحوي بداخله أي مشروع، لأننا سنقوم بإضافة المشاريع لاحقاً، لكن ما يهمنا هو توضيها ضمن Solution واحد. لذلك سيكون لدينا شكل الـ Solution Explorer كالتالي:

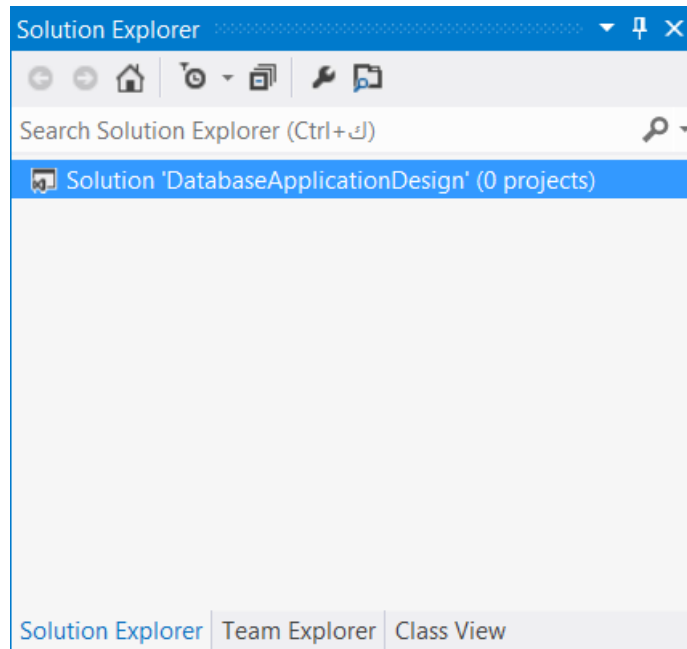


Figure3 : Solution Explorer contains no project

عملياً، الآن أصبح لدينا مسار لمشاريعنا سنقوم بمتابعة المقالة كاملة على أساس هذا Solution، وسوف نفرق بين إسم Solution وبين إسم المشروع، فلكل واحد خواصه كما نعلم.

3. إنشاء الإتصال مع قاعدة البيانات

الآن وقبل الخوض في إنشاء المشاريع الباقية، علينا تعريف الإتصال بقاعدة البيانات، لكي نتمكن من استخدام هذا الإتصال في المراحل القادمة، ولا نضطر بالتالي لكتابته كل مرة، لذلك ما علينا فعله الآن هو الذهاب إلى نافذة Server Explorer، وإذ لم تكن ظاهرة على Visual Studio نعمل التالي:

View→ Server Explorer

بعد الذهاب إلى هذه النافذة نقوم بالضغط على زر Connect To Database.

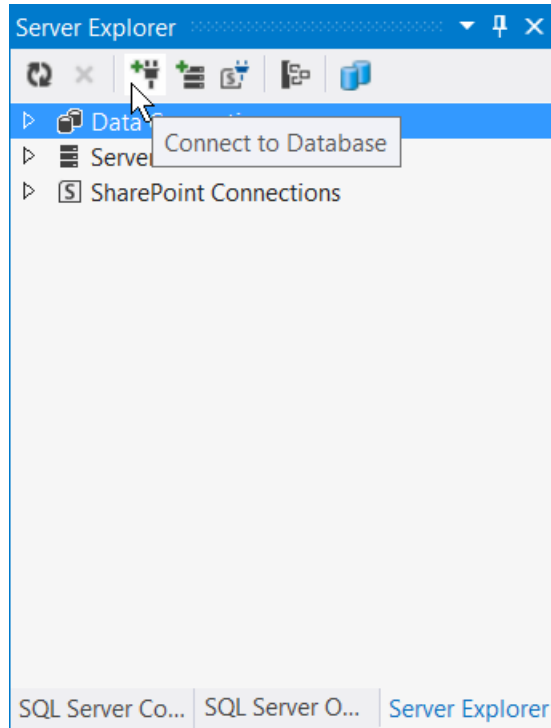


Figure4 : Add new connection from Server Explorer

سيظهر لنا بعدها نافذة إعداد الإتصال الجديد، هذه النافذة مشهورة لكل المطورين على Visual Studio، فكل ما علينا هو تحديد السيرفر الذي نعمل عليه (SQL EXPRESS في مثالنا هذا)، ومن ثم اختيار اسم قاعدة البيانات من ضمن قواعد بيانات هذا السيرفر، وهكذا نحصل على إتصال مع قاعدة البيانات بكل سهولة.

Enter information to connect to the selected data source or click "Change" to choose a different data source and/or provider.

Data source:
Microsoft SQL Server (SqlClient) Change...

Server name:
.\SQLEXPRESS Refresh

Log on to the server

☒ Use Windows Authentication
☐ Use SQL Server Authentication

User name:
Password:
☐ Save my password

Connect to a database

☒ Select or enter a database name:
Northwind

☐ Attach a database file:
 Browse...
Logical name:

Advanced...

Test Connection OK Cancel

Figure5 : Connect to SQL EXPRESS with Northwind database

يمكننا التحقق من نجاح الإتصال بالضغط على Test Connection والذي سيظهر لنا رسالة تحدد نتيجة الإتصال، في أغلب الأحيان تكون النتيجة ايجابية، كونه استطاع الوصول إلى السيرفر واستخراج قواعد البيانات من ضمنه، ولكن ليس من الخطأ التحقق من نجاح الإتصال. طبعاً سيقوم Visual Studio بإضافة هذا الاتصال إلى مجموعة الإتصالات ضمن Data Connections ضمن Server Explorer

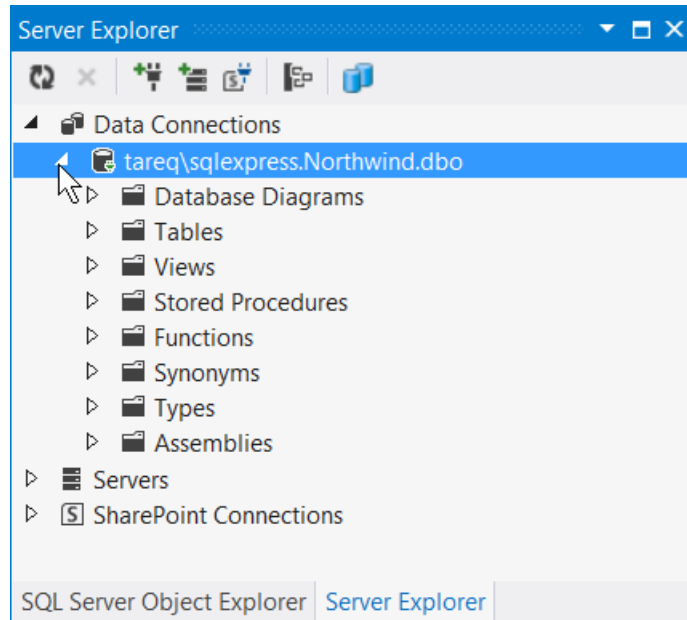


Figure6 : Northwind connection added to the Server Explorer

الآن لو ذهبنا إلى خصائص هذا الاتصال (F4)، سنلاحظ أنه يحتفظ بمعلومات الاتصال الجديد ضمن
الخاصية Connection String

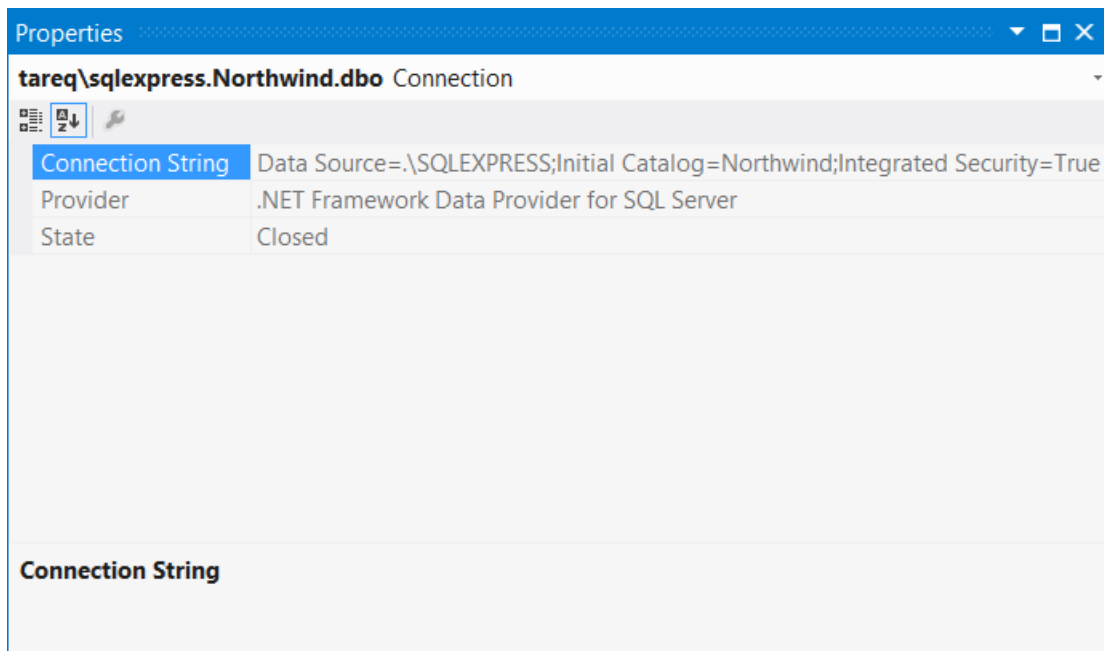


Figure7 : Northwind connection properties

عناصر هذا الإتصال الأساسية هي ثلاث عناصر:

1. Data Source

عبارة عن مزود قواعد البيانات أو السيرفر، في حالتنا (SQLEXPRES.\) المقصود بالنقطة (.) أن السيرفر موجود على نفس الجهاز localhost. والـ SQLEXPRESS هي نسخة هذا السيرفر.

2. Initial Catalog

يستخدم لتحديد اسم قاعدة البيانات، (Northwind) في مثالنا، يكفيننا الاسم فقط.

3. Integrated Security

إعدادات الأمان والحماية، لتحديد إمكانية الوصول لقاعدة بياناتنا، عندما نضعها إلى true فنقصد أن يستخدم إعدادات الأمان الخاصة بويندوز، فمثلاً المستخدمين العاديين للنظام لا يمكنهم الوصول لها، أما Administrator بالطبع يمكنه الوصول بكل سهولة. أما إذا قمنا بوضعها إلى false فنحن بحاجة لتحديد اسم مستخدم وكلمة مرور، لنتمكن من الوصول.

4. صناعة طبقة الوصول للبيانات يدوياً

سنقوم في هذه الفقرة بإنشاء المشروع يدوياً، وسنقوم بتعريف جميع ما يلزمنا يدوياً، كوننا لا نعتمد على أي تقنية أخرى، كل ما نحتاجه هو مجموعة من الكلاسات SQL التي تساعدنا في الإتصال بقاعدة البيانات، لذلك نذهب إلى Solution ونضغط باليمين عليه، ثم نضيف مشروع جديد:

Solution Name → Right Click → Add → New Project → Class Library

نقوم بإختار نوع المشروع Class Library وهو عبارة عن مشروع مكتبة. لا يحوي على أي واجهات، فنتائج هذا المشروع عبارة عن ملف dll وليس ملف تنفيذي. بالطبع من الجيد تحزيم مشروع معين ضمن مكتبة ديناميكية (Dynamic Link Library) dll، لأنه بذلك نستطيع إخفاء الكود الخاص بنا، وإعطاء خدمات فقط، ولاننسى السرعة التي تقدمها هذه المكتبات كونها مترجمة Compiled، أي ليس لها علاقة ب Compile المشروع الخاص بنا، وهي أفضل أنواع Packaging.

نقوم بتسمية هذه الطبقة (ByHandDataAccess) فيتم إنشاء المشروع وإضافته إلى Solution Explorer

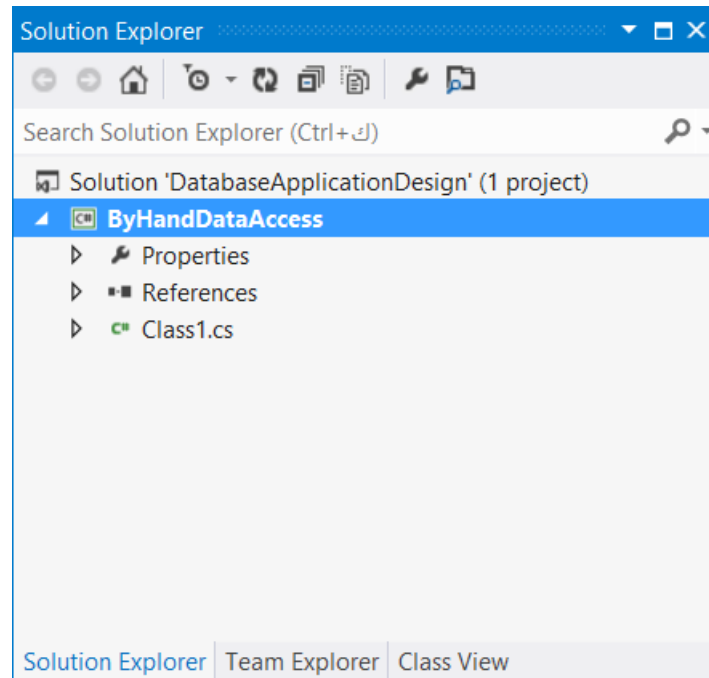


Figure8 : ByHandDataAccess class library project

الآن نقوم بإضافة ملف إعدادات إلى مشروعنا كونه لا يحويه، لذلك نضغط بالزر الأيمن على اسم المشروع ثم نختار إضافة عنصر جديد ثم نضيف ملف Settings.settings

Project Name → Right Click → Add → New Item → Settings.settings

نسمي الملف Setting.settings، الآن نحن بحاجة لإضافة Connection String إلى هذا الملف لنتمكن من الوصول إليه بسرعة، ونتمكن من تغييره لاحقاً. لذلك نضيف حقل إلى هذه الملف ونسميه Connection String ونضع له القيمة النصية التي تولدت معنا عندما أنشأنا الإتصال سابقاً.

	Name	Type	Scope	Value
▶	ConnectionString	string	User	Data Source=.\SQLEXPRESS;Initial Catalog=Northwind;Integrated Security=True
*	Setting	string	User	

Figure9 : Add connection string to settings file

بالطبع نحن بحاجة لإضافته في هذا النوع من المشروع، كوننا ننشأ يدوياً، أي لا توجد آلية لتعريف Connection String تلقائياً ضمن مشروعنا، أما في التقنيات اللاحقة، فلسنا بحاجة لهذه الخطوة، حيث أنه سيضاف تلقائياً إلى ملف App.config.

الآن نقوم بإعادة تسمية Class1.cs إلى NorthwindDataAccess.cs بعدها نقوم بفتحه، ونضيف public لتعريف الكلاس، ونضيف فضاء أسماء جديد هو SqlConnection لنتمكن من التحدث مع قاعدة البيانات، ليصبح لدينا تعريف الكلاس بالشكل التالي:

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

namespace ByHandDataAccess
{
    public class NorthwindDataAccess
    {

    }
}
```

الآن ولكي نتمكن من حفظ واسترجاع البيانات بسهولة، علينا بتعريف بعض الـ Classes التي تمثل الجداول في قاعدة البيانات، ولتسهيل العمل سنعمل على ثلاث جداول فقط، هي Categories, Suppliers, Products لذلك سنقوم بتعريف كلاس لكل جدول من هذه الجداول ضمن طبقة الوصول للبيانات، وبالتالي داخل وخارج هذه الطبقة سيكون تعاملنا مع Object بدل التعامل مع مجموعة من المتغيرات. لنلقي نظرة على محتوى هذه الكلاسات الثلاث:

Category.cs

[illegible]

```
}  
}
```

الآن لنذهب ونرى الكلاس Supplier والذي يحوي على مجموعة من الخواص نكتفي بها طبعاً حيث أن هناك بعض الخواص الأخرى ضمن الجدول، ولكن ما يهمنا هو المعلومات الحالية:

Supplier.cs

```
public class Supplier  
{  
    public int ID { get; set; }  
  
    public string CompanyName { get; set; }  
  
    public string ContactName { get; set; }  
  
    public string ContactTitle { get; set; }  
  
    public string Country { get; set; }  
  
    public string Address { get; set; }  
  
    public string City { get; set; }  
  
    public string Phone { get; set; }  
  
    public Supplier() { }  
  
    public Supplier(int id, string companyName,  
        string contactName, string contactTitle,  
        string country, string address,  
        string city, string phone)  
    {  
        ID = id;  
        CompanyName = companyName;  
        ContactTitle = contactTitle;  
        ContactName = contactName;  
        Country = country;  
        Address = address;  
        City = city;  
        Phone = phone;  
    }  
  
    public override string ToString()  
    {  
        return string.Format("ID: {0}\nCompany Name: {1}\nContactName:  
        {2}\nContactTitle: {3}\nCountry: {4}\nAddress:  
        {5}\nCity: {6}\nPhone: {7}\n\n",  
            ID, CompanyName, ContactName, ContactTitle, Country,  
            Address, City, Phone);  
    }  
}
```

الآن نذهب ونرى كلاس Product، طبعاً هذا الكلاس يحوي على علاقات بداخله، لذلك سنقوم بتعريف بعض Foreign Keys ضمنه لكل من Category, Supplier

Product.cs

```
public class Product
{
    public int ID { get; set; }

    public int CategoryID { get; set; }

    public int SupplierID { get; set; }

    public string Name { get; set; }

    public string QuantityPerUnit { get; set; }

    public decimal UnitPrice { get; set; }

    public bool Disconnected { get; set; }

    public Product() { }

    public Product(int id, int categoryId,
        int supplierId, string name,
        string quantityPerUnit, decimal unitPrice,
        bool disconnected)
    {
        ID = id;
        CategoryID = categoryId;
        SupplierID = supplierId;
        Name = name;
        QuantityPerUnit = quantityPerUnit;
        UnitPrice = unitPrice;
        Disconnected = disconnected;
    }

    public override string ToString()
    {
        return string.Format("ID: {0}\nName: {1}\nCategoryID: {2}\nSupplierID: {3}\nQuantity Per Price: {4}\nDisconnected: {5}",
            ID, Name, CategoryID, SupplierID, QuantityPerUnit, Disconnected);
    }
}
```

جميل. هذا كل ما نحتاجه لتعريف البيانات التي سنتعامل معها. نستطيع تطوير هذه الكلاسات لتقدم خدمات رائعة، ولكننا سنكتفي بهذا القدر هنا، كون التقنيات اللاحقة ستزودنا بتلك الخدمات تلقائياً. سننتقل الآن للعمل والتحدث مع قاعدة البيانات من خلال NorthwindDataAccess والذي سنكتب بداخله أساسيات التحدث مع قاعدة البيانات وجلب البيانات وما إلى ذلك.

نعرف اوبجكت من نوع SqlConnection وهو الذي سيتصل بقاعدة البيانات وتفتحها لنا وتصبح جاهزة للعمل، هذه الخطوة نقوم نحن بتطبيقها يدوياً عندما نقوم بفتح SQL Server ومن ثم تحديد نوع السيرفر SQL EXPRESS مثلاً ونضغط على Connect. ونحن بحاجة أيضاً لاوبجكت أخر من نوع SqlCommand وهو يشبه ما نقوم به عند عمل New Query ومن خلال هذان العنصران يمكننا طلب ما نريد من قاعدة البيانات. لذلك سيكون تعريفها كالتالي:

```
public class NorthwindDataAccess
{
    private SqlConnection connection;
    private SqlCommand query;

    public NorthwindDataAccess()
    {
        connection = new SqlConnection(Settings.Default.ConnectionString);
        query = new SqlCommand("", connection);
    }
}
```

هكذا تصبح قاعدة البيانات جاهزة للتعامل معنا، لذلك سنقوم بطلب مجموعة الـ Categories الموجودة ضمن قاعدة البيانات، أي سنقوم بقراءة حقول جدول Categories من قاعدة البيانات سطر بسطر، وسنخزن كل سطر ضمن اوبجكت من نوع Category الذي قمنا بصنعه سابقاً. ثم نقوم بإرجاع List من هذه العناصر، ما ستحويه هذه الدالة المسماة GetCategories هو تعريف query الخاصة بقراءة الجدول، ثم تعريف Reader لقراءة أسطر الجدول ثم تخزين هذه الأسطر ضمن List، وأخيراً نرجعها للمستخدم، بالطبع لا ننسى فتح وإغلاق الاتصال مع قاعدة البيانات. كالتالي:

```
public List<Category> GetCategories()
{
    List<Category> Categories = new List<Category>();

    query.CommandText = "SELECT * FROM Categories";

    connection.Open();

    SqlDataReader reader = query.ExecuteReader();

    while (reader.Read())
    {
        int id = reader.GetInt32(0);

        string name = reader.GetString(1);

        string description = reader.GetString(2);

        Category category = new Category(id, name, description);
    }
}
```

```

        Categories.Add(category);
    }
    connection.Close();

    return Categories;
}

```

للتوضيح، في الواقع قمت بتجربة الميثود السابقة للتأكد من فعاليتها، فقامت فقط بإستدعاءها وطباعة محتواها على الشاشة فكانت النتائج كالتالي:

```

C:\Windows\system32\cmd.exe
ID: 1
Name: Beverages
Description: Soft drinks, coffees, teas, beers, and ales

ID: 2
Name: Condiments
Description: Sweet and savory sauces, relishes, spreads, and seasonings

ID: 3
Name: Confections
Description: Desserts, candies, and sweet breads

ID: 4
Name: Dairy Products
Description: Cheeses

ID: 5
Name: Grains/Cereals
Description: Breads, crackers, pasta, and cereal

ID: 6
Name: Meat/Poultry
Description: Prepared meats

ID: 7
Name: Produce
Description: Dried fruit and bean curd

ID: 8
Name: Seafood
Description: Seaweed and fish

Press any key to continue . . .

```

Figure10 : Test the GetCategories method

بعد التأكد من نجاعة العمل، سنقوم بتطبيق الميثودتين التاليتين (GetSuppliers, GetProducts)، اللتان ستجلبان بيانات Suppliers, Products وسيكون التطبيق مطابق للسابق، مع ملاحظة إختلاف خصائص وحقول كل جدول.

```
public List<Supplier> GetSuppliers()
{
    List<Supplier> Suppliers = new List<Supplier>();

    query.CommandText = "SELECT * FROM Suppliers";
    connection.Open();

    SqlDataReader reader = query.ExecuteReader();

    while (reader.Read())
    {
        int id = reader.GetInt32(0);

        string companyName = reader.GetString(1);

        string contactName = reader.GetString(2);

        string contactTitle = reader.GetString(3);

        string country = reader.GetString(8);

        string address = reader.GetString(4);

        string city = reader.GetString(5);

        string phone = reader.GetString(9);

        Supplier supplier = new Supplier(id, companyName, contactName,
                                         contactTitle, country, address, city,
                                         phone);

        Suppliers.Add(supplier);
    }
    connection.Close();

    return Suppliers;
}
```

هكذا نكون انتهينا مع عمليات جلب البيانات، لا يزال لدينا عمليات الإضافة والتعديل والحذف، لذلك سنقوم بكتابتها، بدون شرح مفصل، كونها معروفة وتقليدية بالنسبة للكثير، وكما تحدثنا سابقاً هذه الطريقة قليلة الاستخدام كون التقنيات الحديثة حلت الكثير من المشاكل وأعطت الكثير من الحلول والميزات، لذلك سنكتفي بكتابة عمليات الإضافة والتعديل والحذف على جدول Category، وبالطبع العمل مشابه للجداول الأخرى.

أولاً: عملية إضافة Category جديد إلى قاعدة البيانات، كل ما نحتاجه هنا هو كود صرف خاص بإدخال البيانات للقاعدة، بدون التحقق من الشروط والضوابط، حيث أن التحقق في مثل هذه المرحلة والذي يقوم به الكثير هو خلط بين المفاهيم المتبعة بالتصميم، فالآن أنا اتكلم مع قاعدة بيانات، فلا يجب علي وضع شروط لإدخال البيانات. التحدث مع قاعدة البيانات واضح وصريح.

```
public void InsertCategory(Category category)
{
    query.CommandText = "INSERT INTO Categories (CategoryName, Description) VALUES (@Name, @Description)";

    query.Parameters.AddWithValue("@Name", category.Name);
    query.Parameters.AddWithValue("@Description", category.Descriptpion);
    connection.Open();
    query.ExecuteNonQuery();
    connection.Close();
}
```

ثانياً: تعديل Category معين ضمن قاعدة البيانات

```
public void UpdateCategory(Category category)
{
    query.CommandText = "UPDATE Categories SET CategoryName =@Name , Description = @Description WHERE CategoryID = @ID";

    query.Parameters.AddWithValue("@Name",category.Name);
    query.Parameters.AddWithValue("@Description", category.Descriptpion);
    query.Parameters.AddWithValue("@ID", category.ID);
    connection.Open();
    query.ExecuteNonQuery();
    connection.Close();
}
```

ثالثاً: حذف Category من قاعدة البيانات

```
public void DeleteCategory(int categoryId)
{
    query.CommandText = "DELETE FROM Categories WHERE CategoryID = @ID";

    query.Parameters.AddWithValue("@ID", categoryId);
    connection.Open();
    query.ExecuteNonQuery();
    connection.Close();
}
```

الآن بعد أن نكمل على نفس الطريقة باقي الدوال، نكون قد انتهينا من عمل طبقة الوصول للبيانات يدوياً، كل ما علينا فعله هو تسمية الميثود المطلوبة وكتابة Query بداخلها وتنفيذها بدون أي إضافات أخرى. ربما يتساءل البعض ماذا لو حدث خطأ معين أثناء التنفيذ، ماذا سنفعل وقتها؟ سؤال وجيه بالطبع، وهو هدفنا في هذا الكتاب، فقد فصلنا في الطبقات وجعلنا كل طبقة تقوم بعملها بصورة منطقية بدون إضافات ليس لها علاقة بعمل الطبقة، وسيتم معالجة هذه الأخطاء في الطبقة الثانية Business Logic Layer. الآن علينا الذهاب لرؤية كيفية صنع المثال السابق بإستخدام تقنيات ميكروسوفت والتي ستقوم بعمل جزء كبير من العمل عوضاً عنا، فكما لاحظنا في الخطوة السابقة احتجنا للكثير من العمل لإنجاز عمل طبقة الوصول للبيانات.

إلى الدرس القادم بإذن الله.

5. صناعة طبقة الوصول للبيانات بواسطة DataSet

6. صناعة طبقة الوصول للبيانات بواسطة Linq to Sql Classes

7. صناعة طبقة الوصول للبيانات بواسطة ADO.NET Entity Framework Model

2. تصميم طبقة القواعد والضوابط Business Logic Layer

3. تصميم طبقة العرض Presentation Layer