

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

1. صناعة طبقة الوصول للبيانات بواسطة DataSet

DataSet وعلمها معروف تقريباً لدى أغلب المبرمجين، تعمل بوضع غير متصل بقاعدة البيانات، أي تقوم بإحضار البيانات ووضعها في الذاكرة، وبعد ذلك يصبح برنامج المستخدم يتعامل مع هذه البيانات الموجودة في الذاكرة كأنها بيانات محلية، وفي أي وقت يمكننا بالنزول بهذه البيانات لقاعدة البيانات وحفظها دفعة واحدة، طبعاً تفيد هذه التقنية كثيراً في تلك البرامج التي تتطلب الكثير من العمليات على قاعدة البيانات، لذلك بدل أن نقوم بإنشاء إتصالات كثيرة مع قاعدة البيانات، يقل عدد الإتصالات لحد كبير جداً.

لنبدأ إذاً.

نقوم بإضافة مشروع جديد من نوع Class Library إلى Solution الخاص بنا، ونقوم بتسمية المشروع DataSetDataAccess، للدلالة على أن الوصول للبيانات يتم بواسطة DataSet. ثم نقوم بتسمية Class1.cs المولد تلقائياً إلى NorthwindDataAccess.cs كما فعلنا في المرة السابقة، الآن نقوم بإضافة عنصر جديد للمشروع من نوع DataSet ونقوم بتسميته DataSet.xsd فنلاحظ ظهور صفحة زرقاء اللون تعتبر المصمم Designer الخاص بـ DataSet التي سنتعامل معها بعد قليل. بعد كل هذه الخطوات سيكون شكل المشروع كالتالي:

DataSet وعلمها معروف تقريباً لدى أغلب المبرمجين، تعمل بوضع غير متصل بقاعدة البيانات، أي تقوم بإحضار البيانات ووضعها في الذاكرة، وبعد ذلك يصبح برنامج المستخدم يتعامل مع هذا البيانات الموجودة بالذاكرة كأنها بيانات محلية، وفي أي وقت يمكننا بالنزول بهذه البيانات لقاعدة البيانات وحفظها دفعة واحدة، طبعاً تفيد هذه التقنية كثيراً في تلك البرامج التي تتطلب الكثير من العمليات على قاعدة البيانات، لذلك بدل أن نقوم بإنشاء إتصالات كثيرة مع قاعدة البيانات، يقل عدد الإتصالات لحد كبير جداً. لنبدأ إذاً.

نقوم بإضافة مشروع جديد Class Library إلى Solution الخاص بنا، ونقوم بتسمية المشروع DataSetDataAccess، للدلالة على أن الوصول للبيانات يتم بواسطة DataSet. ثم نقوم بتسمية Class1.cs المولد تلقائياً إلى NorthwindDataAccess.cs كما فعلنا في المرة السابقة، الآن نقوم بإضافة عنصر جديد للمشروع من نوع DataSet ونقوم بتسميته Northwind.xsd فنلاحظ ظهور صفحة زرقاء اللون تعتبر المصمم Designer الخاص بـ DataSet التي سنتعامل معها بعد قليل. بعد كل هذه الخطوات سيكون شكل المشروع كالتالي:

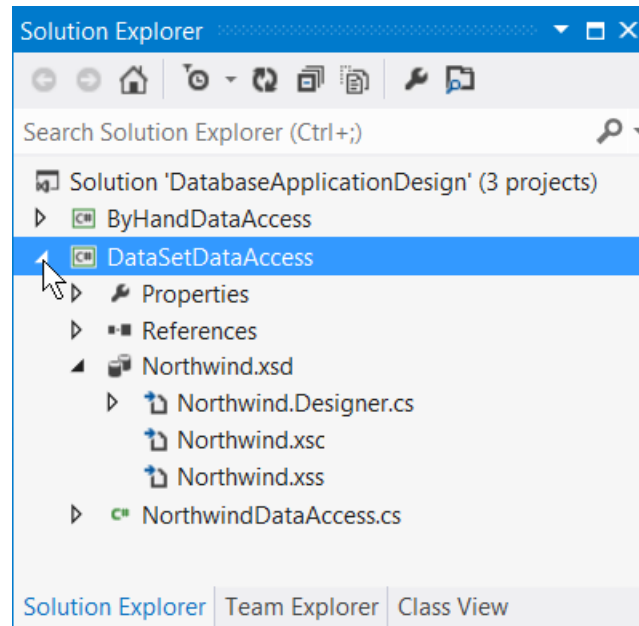


Figure1 : Data Access Layer project with DataSet

الآن نقوم بفتح ملف التصميم Northwind.xsd حيث سنقوم بإضافة الجداول للتعامل معها في عملية التخاطب مع قاعدة البيانات، طريقة إضافة جدول تتم عن طريق عدة خطوات سهلة نقوم من خلالها بتحديد شكل الجدول (الحقول المطلوبة من الجدول) ونقوم أيضاً بتحديد شكل الـ Queries من إضافة وتعديل وحذف وغيرها، سنقوم بعرض خطوات إضافة الجدول Categories ونترك لك إضافة الجدولين الآخرين Products, Suppliers:

1. على واجهة التصميم Northwind.xsd نضغط بالزر الأيمن لتظهر لنا قائمة نختار منها Add ثم TableAdapter.

2. ستظهر لنا نافذة لتحديد الإتصال مع قاعدة البيانات، بما أننا أنشأنا هذا الإتصال مسبقاً فيجب أن يظهر ضمن القائمة ComboBox، او نستطيع إعادة تشكيل الإتصال عن طريق New Connection...، بعد تحديد الإتصال NorthwindConnectionString، نضغط على Next للانتقال إلى الخطوة التالية.

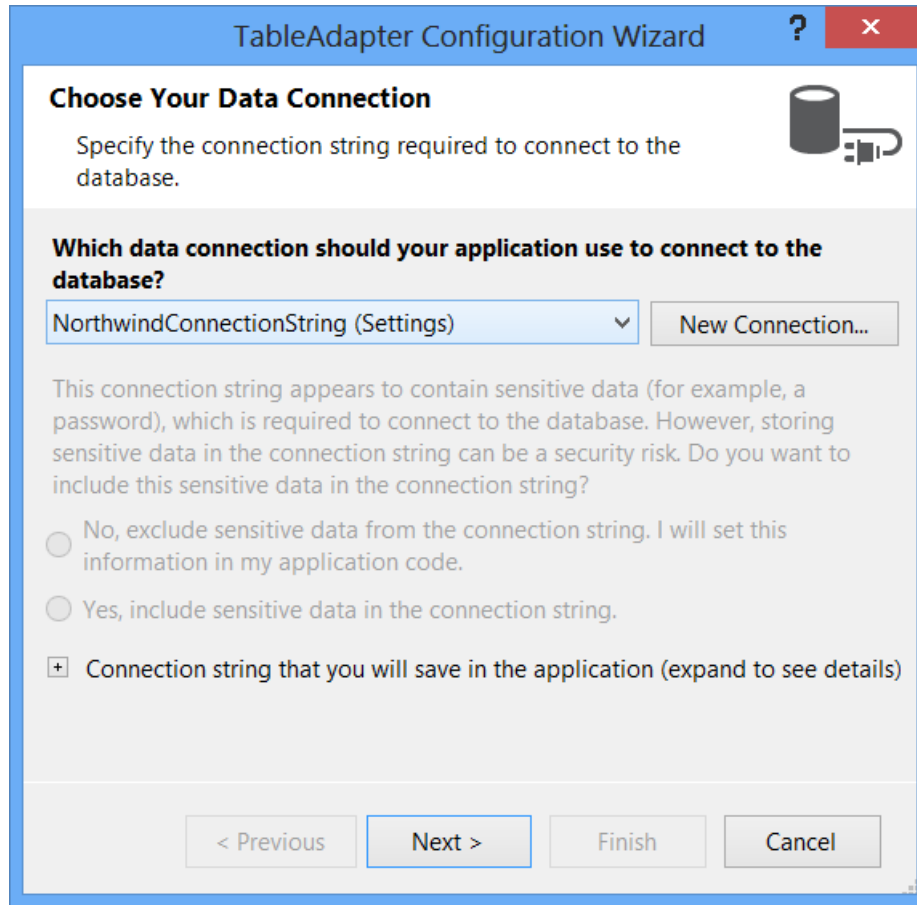


Figure 12: TableAdapter Connection String

3. تظهر لنا قائمة فيها ثلاث خيارات نختار الخيار الأول وهو عبارة عن جملة SQL، وسنلاحظ استخدام الخيارات الأخرى لاحقاً. الآن نضغط على Next

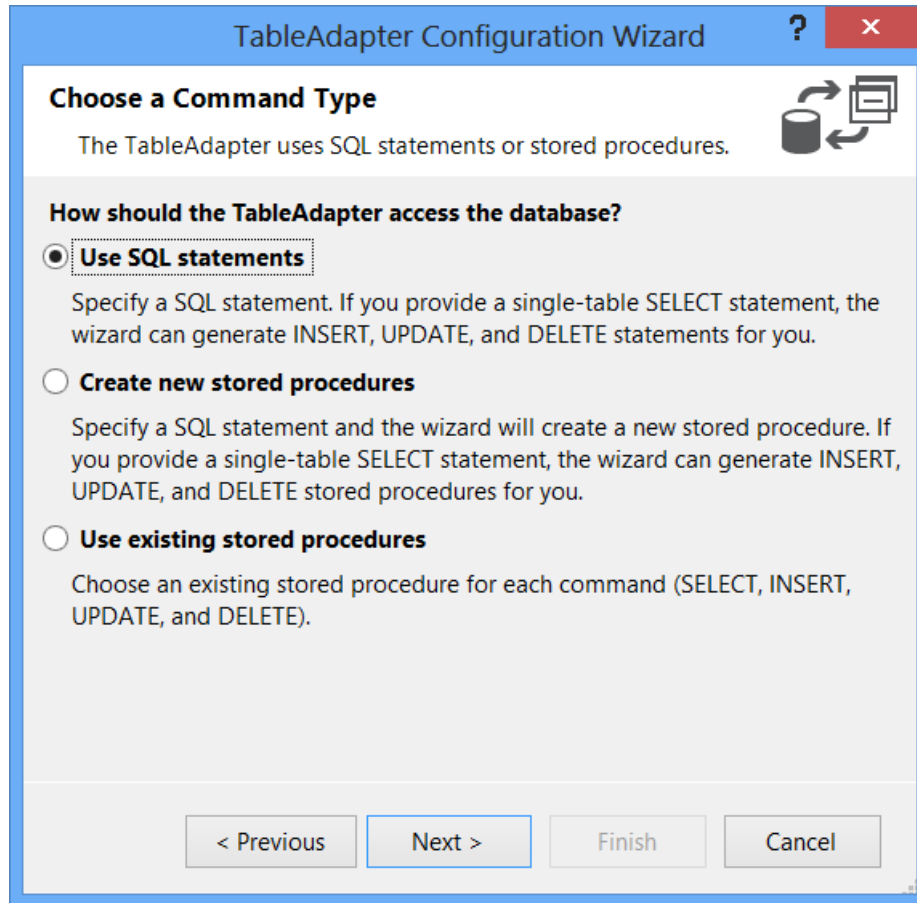


Figure 13: TableAdapter SQL Statements

4. تظهر لنا نافذة جديدة لتحديد شكل الـ Query او SQL Statements بإمكاننا كتابتها يدويا كما نفعل مع جمل SQL عادة. أو بإمكاننا استخدام اداة تدعى Query Builder حيث ستقوم بتوليد الـ Query، لذلك نضغط على الزر Query Builder... فتظهر لنا نافذتان تحوي واحدة على الجداول ضمن قاعدة البيانات، نختار الجدول Categories من النافذة الصغيرة ثم نغلقها، ليظهر لنا تصميم هذا الجدول على النافذة الاخرى، الآن نحن بحاجة فقط لتحديد الحقول المطلوبة من الجدول، يوجد خيار لتحديد كامل الحقول نقوم بتحديد:

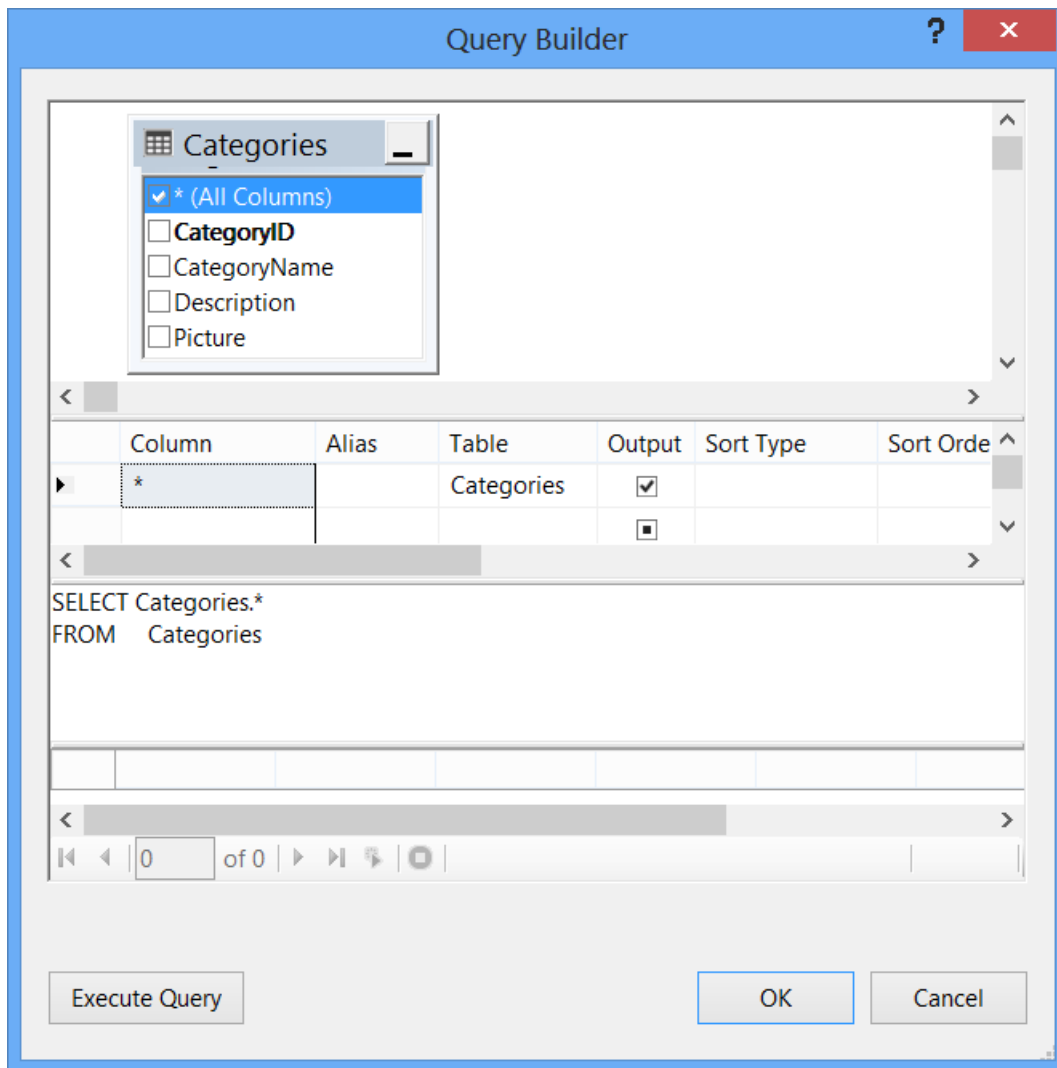


Figure 14: New TableAdapter Query Builder

5. بعد الضغط على OK نقوم بالعودة للنافذة السابقة والتي ستظهر لنا الـ Query المولدة، بالطبع حاليا تعتبر هذه الـ Query بسيطة يمكننا كتابتها بكل سهولة فما هي فائدة الـ Query Builder؟
الفائدة تظهر في بناء جمل مركبة من أكثر من جدول.

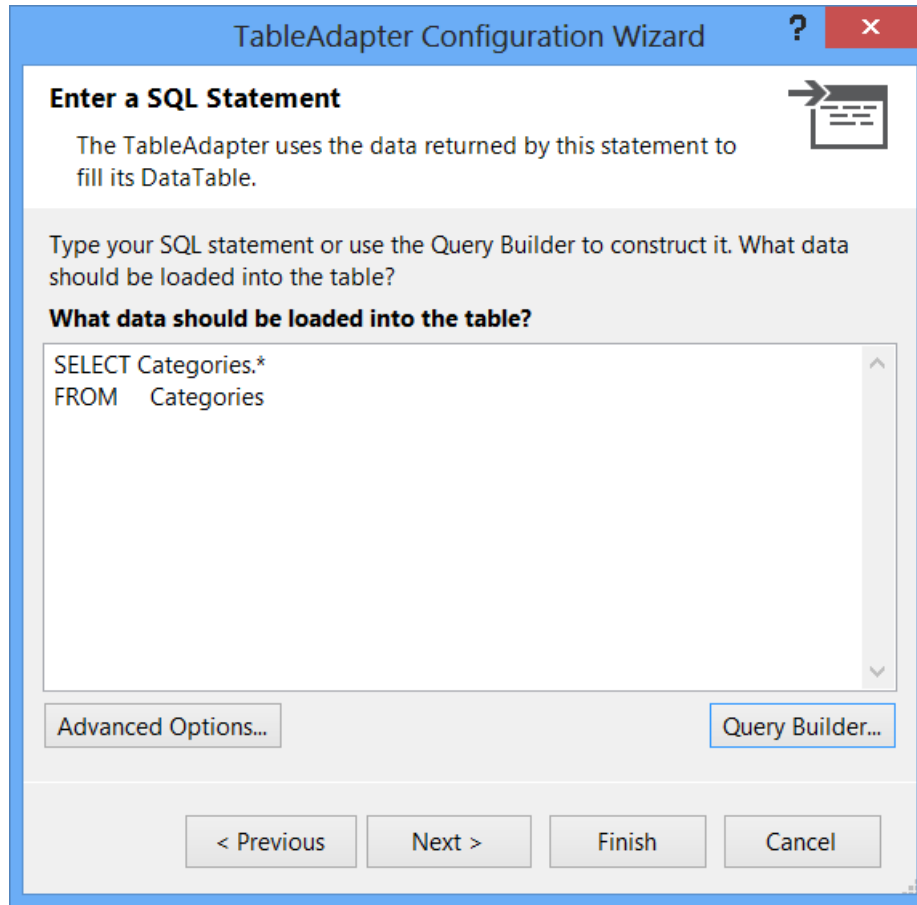


Figure 15: New TableAdapter SQL Statement

6. الآن نضغط على Next لتظهر لنا نافذة أخرى، نستطيع من هذه المرحلة إنها التصميم بالضغط على Finish ولكن الخطوة التالية مهمة نوعاً ما، حيث سنقوم بتسمية الـ Methods المولدة ليتوافق العمل مع التصميم، لذلك سنقوم بتغيير اسم الميثود كالتالي:

TableAdapter Configuration Wizard ? x

Choose Methods to Generate

The TableAdapter methods load and save data between your application and the database.

Which methods do you want to add to the TableAdapter?

☒ **Fill a DataTable**

Creates a method that takes a DataTable or DataSet as a parameter and executes the SQL statement or SELECT stored procedure entered on the previous page.

Method name:

☒ **Return a DataTable**

Creates a method that returns a new DataTable filled with the results of the SQL statement or SELECT stored procedure entered on the previous page.

Method name:

☒ **Create methods to send updates directly to the database**

Creates Insert, Update, and Delete methods that can be called to send individual row changes directly to the database.

Figure 16: New TableAdapter methods names

7. أخيراً نضغط على Finish لنكون قد أنهينا تصميم الجدول بشكل فيكون لدينا الشكل التالي:

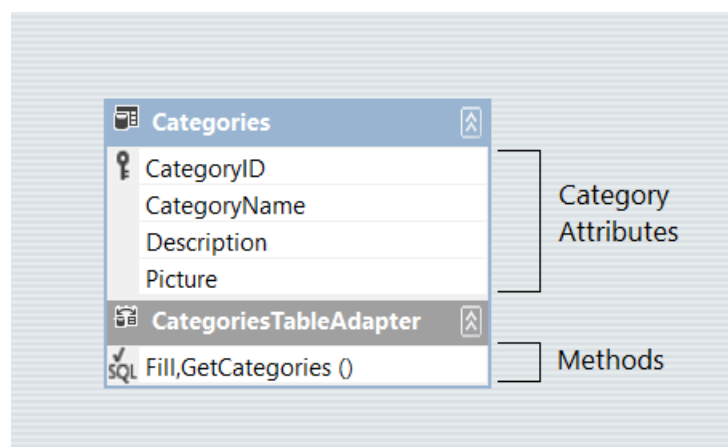


Figure 17: New TableAdapter

الآن نحن بحاجة بعمل الميثود الخاصة بإضافة Category جديد، او تعديل أو حذف Category سابق، لذلك نضغط بالزر الأيمن على أسماء الميثود بالأسفل لتظهر لنا قائمة نختار منها Add Query...

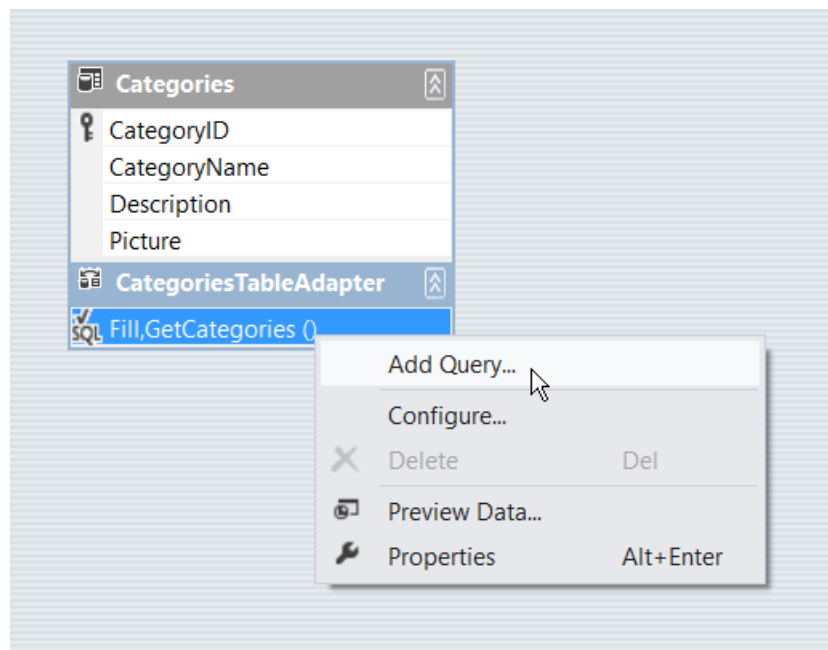


Figure 18: TableAdapter add new query

1. الآن تظهر نافذة شاهدناها سابقا، نختار منها الخيار الأول SQL Statement ثم نضغط على Next
2. الآن تظهر نافذة جديدة نحدد من خلالها نوع الـ Query هل هي إضافة او تعديل او حذف او حتى جلب بيانات من قاعدة البيانات، لذلك سنقوم بإختيار INSERT



Figure 19: Add INSERT query

3. نضغط على Next لتظهر لنا الـ Query المولدة لهذا الغرض بإمكاننا التعديل عليها، لكن نكون حذرين من تغيير حدوث اخطاء بالجملة عند التعديل، ثم نضغط على Next لتظهر لنا نافذة لتسمية هذه الميثود الجديدة، افترضيا تكون InsertQuery لذلك نقوم بتغييرها إلى InsertCategory

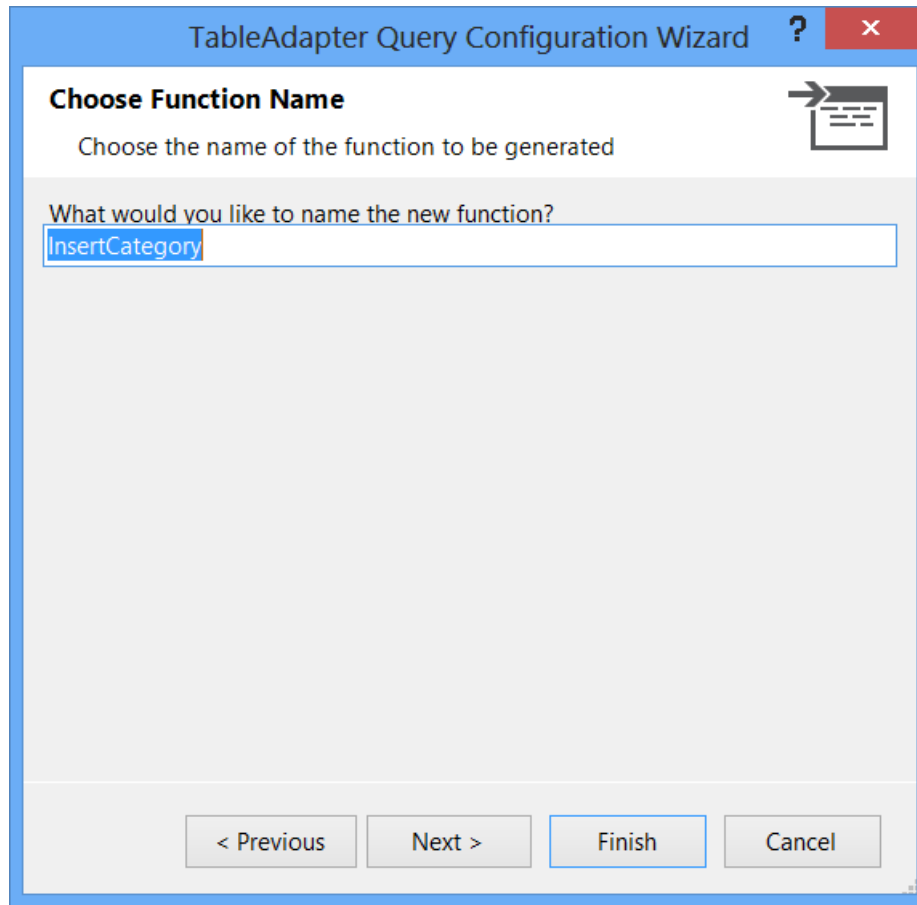


Figure 20: Insert Category method's name

4. نضغط على Finish لننهي التصميم، حيث يمكننا الآن البدء بكتابة كلاس NorthwindDataAccess. ملاحظة: من خلال التصميم التالي فإن الجدول Categories ضمن قاعدة البيانات، أصبح اسمه الآن CategoriesTableAdapter وكل سطر ضمن هذا الجدول يسمى CategoriesRow، لذلك ولكي يكون العمل أكثر سهولة، سنقوم بتعريف الكلاسات Category, Product, Supplier كما فعلنا في المشروع السابق، في الواقع نقوم بالنسخ فقط، الفائدة من ذلك هو فقط لسهولة التعامل مع هذا الطبقة ضمن الطبقات الأخرى ولكي يكون مفهوم البيانات منفصل عن مفهوم الطبقة. بعد نسخ الكلاسات من المشروع السابق ولصقها ضمن مشروعنا الجديد (مع مراعاة تغيير namespace إلى namespace المشروع الحالي) يكون لدينا شكل Solution الحالي بالشكل التالي:

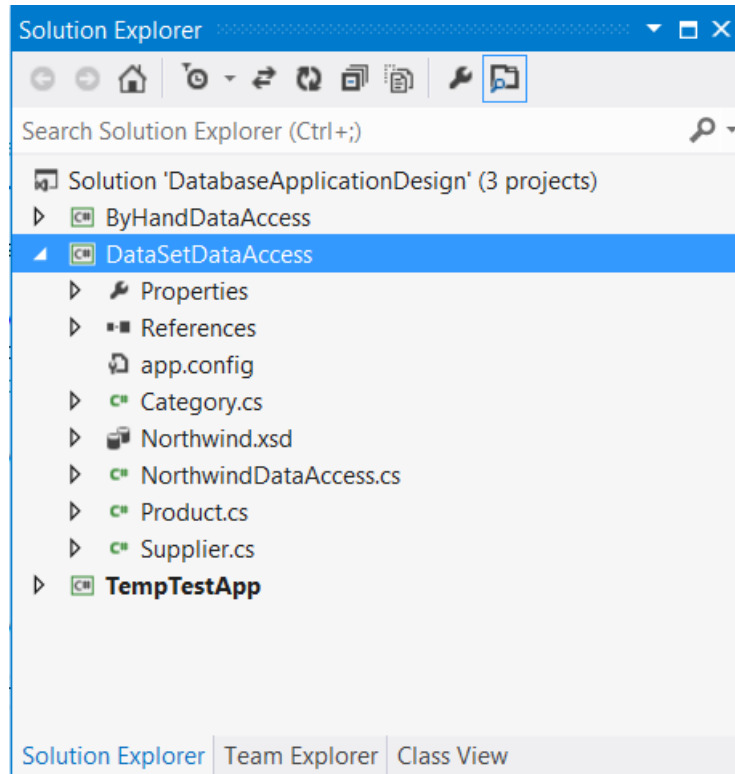


Figure 21: DataSetDataAccess project after adding the classes

نلاحظ بعض العمل اليدوي في هذا الطبقة، والسبب أن هذه التقنية تعتبر قديما نوعا ما، سنلاحظ في الطبقتين التاليتين سهولة التصميم والعمل، حيث لا يوجد للعمل اليدوي نصيب.

الآن نفتح الكلاس NorthwindDataAccess لنقوم بإضافة بعض الميثود التي ستساعدنا في التخاطب مع قاعدة البيانات، وكون أكثر العمل انجز من خلال التصميم، فنحن بحاجة لعمل بعض التنسيقات كما تحدثنا، لذلك نقوم بإنشاء ميثود جديدة تحت إسم GetCategories مهمتها مشابهة لاختها في المشروع السابق حيث ستقوم بقراءة الـ Categories من قاعدة البيانات وارجاعها على شكل List من Categories لذلك سنقوم بإضافة الكود التالي:

```
public class NorthwindDataAccess
{
    NorthwindTableAdapters.CategoriesTableAdapter categoriesAdapter =
        new NorthwindTableAdapters.CategoriesTableAdapter();

    public List<Category> GetCategories()
    {
        List<Category> Categories = new List<Category>();
        Northwind.CategoriesDataTable table = categoriesAdapter.GetCategories();
        foreach (Northwind.CategoriesRow row in table)
        {
            Categories.Add(new Category(row.CategoryID, row.CategoryName,
            row.Description));
        }
    }
}
```

```

    }
    return Categories;
}
}

```

مع هذا الجزء البسيط نكون قد انجزنا المهمة بنجاح، نلاحظ اختلاف الصيغة عن الصيغة في المشروع السابق والسبب هو DataSet التي ساعدتنا في قسم كبير من التصميم والعمل.

لإضافة Category جديد إلى قاعدة البيانات باستخدام الميثود التي قمنا بإضافتها في الخطوات السابقة INSERT نحن بحاجة فقط لاستدعاءها بالشكل التالي:

```

public void InsertCategory(Category category)
{
    categoriesAdapter.InsertCategory(category.Name, category.Descriptipion, null);
}

```

تكرار خطوات العمل السابقة، سيعطينا تعامل كامل مع قاعدة البيانات، مثلاً لتعديل Category معين نقوم بإضافة query الخاصة به كما فعلنا مع INSERT، والأمر نفسه للحذف DELETE، والأمر نفسه بالنسبة لباقي الجداول، أصبحت خطوات العمل واضحة ومفهومة ولا داعي لتكرارها. لذلك اترك لك مهمة تكملة الميثود المتبقية بنفس الأسلوب، وكذلك مهمة اضافة جدول Products إلى التصميم والتعامل معه بنفس الطريقة.

بهذه الخطوات نكون قد انهينا صناعة طبقة الوصول للبيانات بواسطة DataSet، ونصبح جاهزين لصناعة الطبقات الأخرى، لذلك سننتقل لبناء طبقة الوصول للبيانات باستخدام Linq to Sql Classes والتي تتميز بجماليتها وسهولتها في التصميم والعمل.

2. صناعة طبقة الوصول للبيانات بواسطة Linq to Sql Classes

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

1. تصميم طبقة القواعد والضوابط Business Logic Layer

2. تصميم طبقة العرض Presentation Layer