

سلسلة

# ASP.NET

www.asp.net

# أساسيات التعامل مع قواعد البيانات

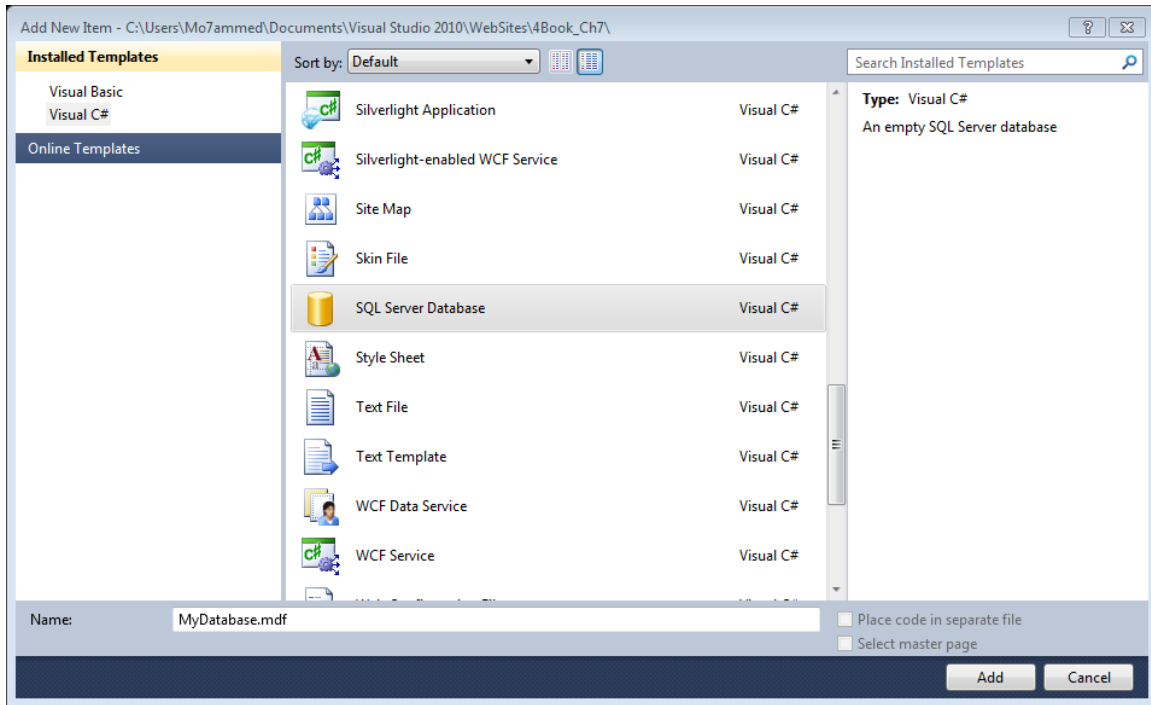
- ✓ إنشاء قاعدة بيانات
- ✓ استخدام الأداة `SqlDataSource`
- ✓ عرض البيانات من الجدول
- ✓ فلترة البيانات
- ✓ التعامل البرمجي مع قواعد البيانات

## أساسيات التعامل مع قواعد البيانات

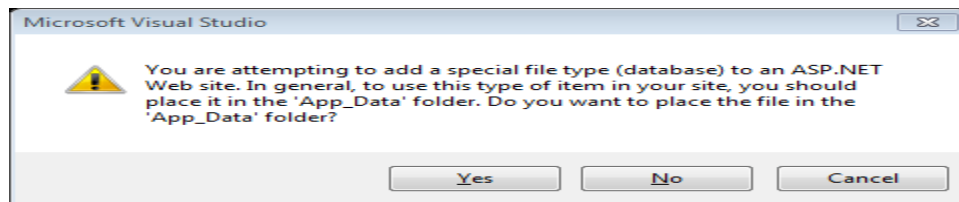
في هذا الفصل سنخطو معاً الخطوة الأولى نحو التعامل مع قواعد البيانات من خلال ASP.NET وذلك بهدف بناء مواقع ديناميكية المحتوى . سنقوم بالبداية بإنشاء قاعدة بيانات بسيطة وملئها ببعض البيانات لتطبيق تمارين هذا الفصل عليها ، حيث سنتعلم كيفية عرض البيانات من الجدول على صفحة الانترنت ، كما سنتعلم كيفية تطبيق عمليات الإضافة والحذف والتعديل على البيانات من خلال الأكواد البرمجية .

### إنشاء قاعدة بيانات

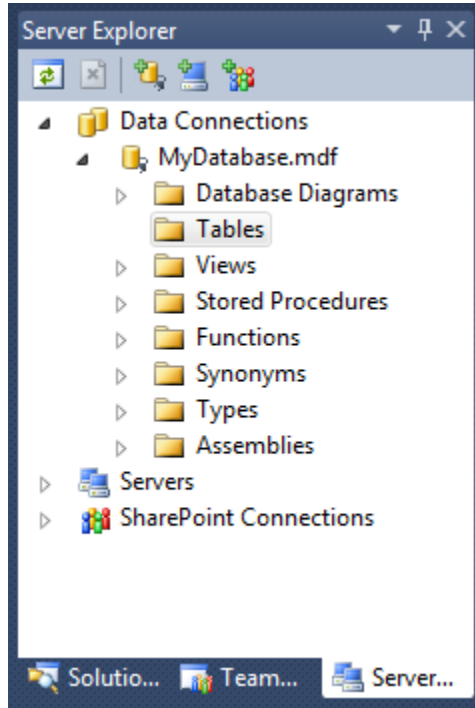
سنقوم بالبداية بإنشاء قاعدة بيانات بسيطة من النوع SQL Server وذلك لنتمكن من تطبيق تمارين هذا الفصل عليها ، ابدأ بإنشاء موقع جديد ، ثم من القائمة File اختر New File ثم اختر SQL Server Database وسمها MyDatabase.mdf ثم اضغط على Add .



ستظهر رسالة تخبرك بأنه سيتم وضع قاعدة البيانات ضمن المجلد App\_Data اضغط على نعم (Yes).



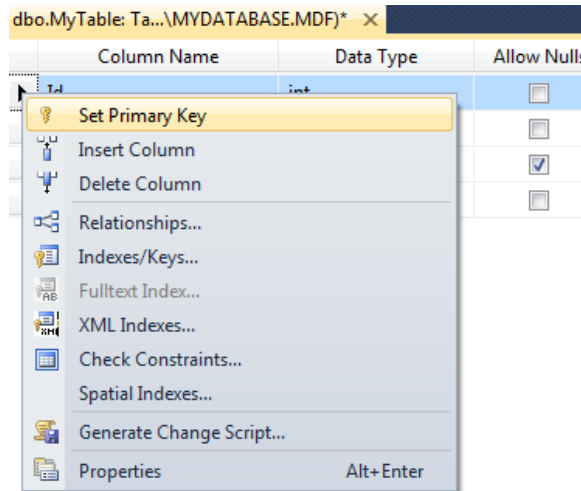
وبالتالي ستجد في القسم Solution Explorer أنه تم إضافة المجلد App\_Data وبداخله قاعدة البيانات MyDatabase . ننقر عليها نقرأ مزدوجاً ليتم الانتقال إلى القسم Server Explorer .



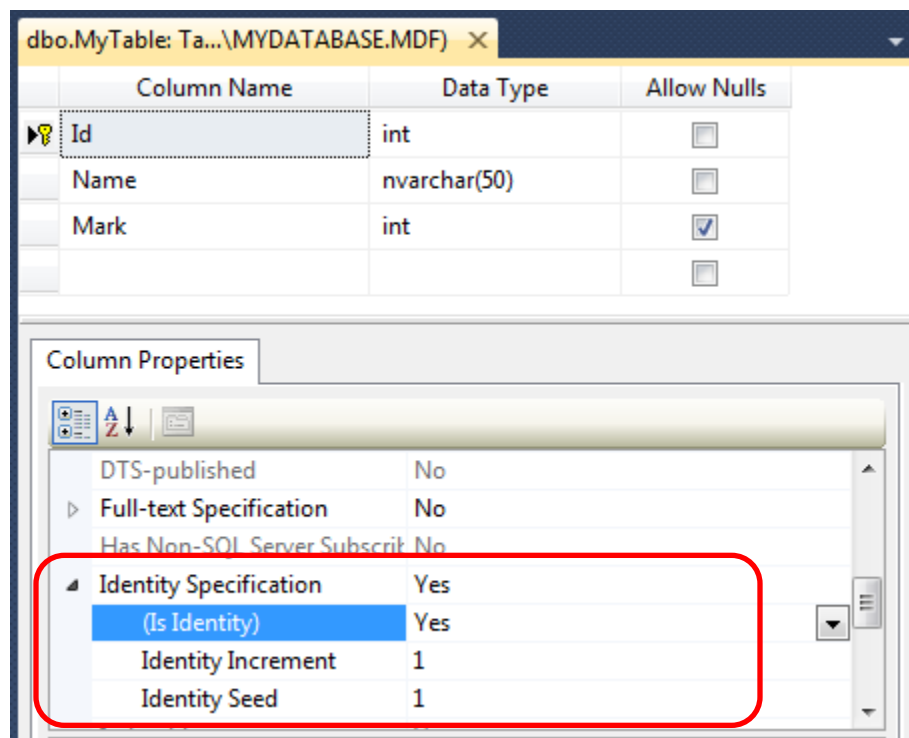
ننقر نقرة يمينية على المجلد Tables ونختار Add New Table ، ونضيف الحقول بالشكل التالي:

| Column Name | Data Type    | Allow Nulls |
|-------------|--------------|-------------|
| Id          | int          | No          |
| Name        | Nvarchar(50) | No          |
| Mark        | int          | Yes         |

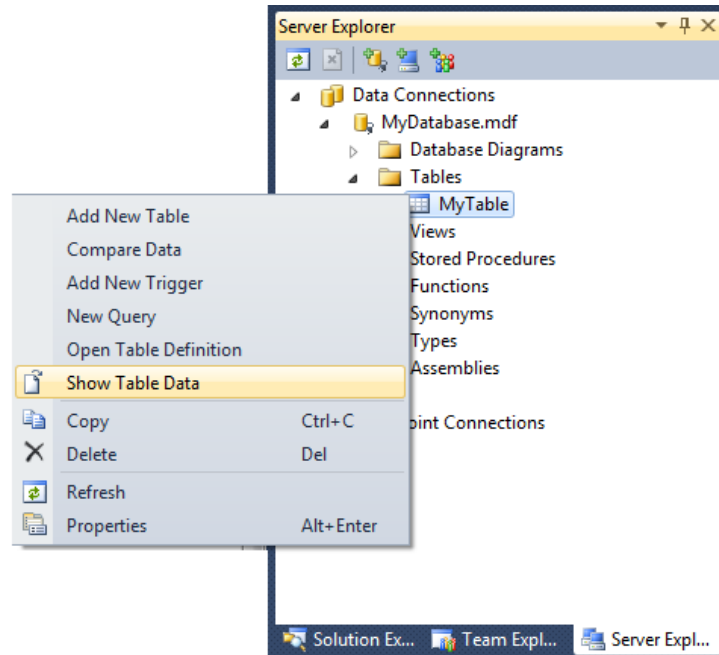
ونحدد الحقل Id كمفتاح أساسي في الجدول ، ويتم ذلك عبر نقرة يمينية على السهم الصغير على يسار الحقل Id واختيار Set Primary Key كما هو واضح في الصورة التالية :



كما نقوم بجعل هذا الحقل يأخذ قيمه بشكل تلقائي ( وذلك لتجنب إدخال قيم مكررة لحقل المفتاح الأساسي ) حيث نقوم بتفعيل الخاصية Is Identity التابعة للحقل Id كالتالي :



نقوم بحفظ الجدول السابق ( من القائمة File نختار Save Table1 ) ونعطيهِ الاسم MyTable .  
نجري نقرة يمينية على هذا الجدول ونختار Show Table Data



نقوم بملىء الجدول ببعض البيانات ثم نغلقه ( ليس بالضرورة كما في الصورة التالية ) .

|   | Id   | Name     | Mark |
|---|------|----------|------|
|   | 1    | Mohammed | 75   |
|   | 2    | Ahmed    | 85   |
|   | 3    | Sandy    | 82   |
| ▶ | 4    | Rama     | 78   |
| * | NULL | NULL     | NULL |

وبهذا نكون قد انتهينا من إعداد قاعدة بيانات بسيطة وأصبحنا جاهزين لتطبيق فقرات هذا الفصل.

### استخدام الأداة SqlDataSource

تمكننا الأداة SqlDataSource من تمثيل قواعد البيانات في صفحات الويب بسهولة ويسر ، ففي أغلب الحالات يتم استخدام هذه الأداة لإدارة قواعد البيانات وبدون كتابة سطر برمجي واحد . يتم استخدام الأداة SqlDataSource لتمثيل الاتصال بقاعدة البيانات و إعداد الأوامر التي ستنفذ على بياناتها ، عموماً الأداة SqlDataSource تستطيع التعامل مع نظم إدارة قواعد البيانات التالية :

- Microsoft SQL Server ✓
- Microsoft SQL Server Express ✓
- Microsoft Access ✓
- Oracle ✓
- DB2 ✓
- MySQL ✓

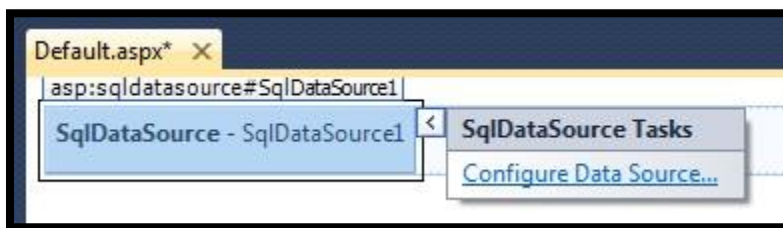
وفضلاً عما سبق فيمكن التعامل مع أي قاعدة بيانات علائقية باستخدام الأداة SqlDataSource .

إذا فإن هذه الأداة هي جسر للوصول للبيانات ويتم استخدامها مع أدوات أخرى لعرض وتحرير البيانات مثل FormView , GridView وغيرهم . تعتبر الأداة SqlDataSource غير مناسبة عند إنشاء تطبيقات متعددة الطبقات multi-tier وذلك لأنها تدمج طبقة البيانات مع طبقة العرض ، وفي هذه الحالة نقوم باستخدام الأداة ObjectDataSource والتي سنناقشها بالتفصيل في فصل لاحق .

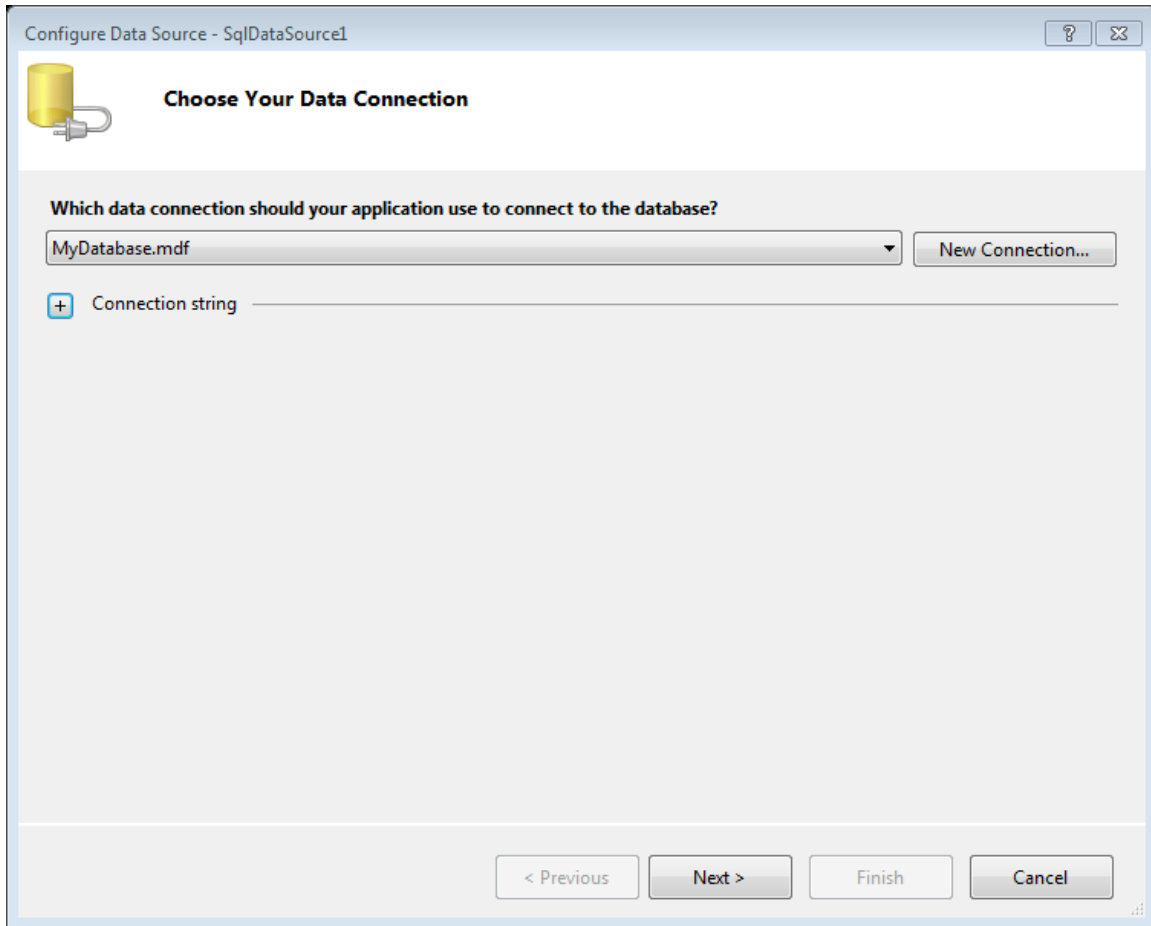
### إنشاء الاتصال بقاعدة البيانات

بشكل افتراضي فإن الأداة SqlDataSource معدة للتعامل مع قواعد البيانات Microsoft SQL Server الإصدار السابع وما بعد ، ويكون البروفايدير ( Provider ) الافتراضي لها ADO.NET .

قم بإضافة صفحة جديدة إلى الموقع ، ثم أضف عليها الأداة SqlDataSource وذلك من القسم Data ضمن الـ Toolbox . نضغط على السهم الصغير للأداة SqlDataSource ( ويدعى Smart Tag ) ، ونضغط على Configure Data Source كما هو واضح في الصورة التالية :



تظهر لنا نافذة لاختيار قاعدة البيانات التي نريد الاتصال بها ، نختار القاعدة MyDatabase ،



وبهذا نكون قد حددنا مصدر البيانات ، إن هذه الخطوة تكافئ كتابة الكود التالي :

### كود C#

```
DataSource=.\SQLEXPRESS;AttachDbFilename=|DataDirectory|\MyDatabase.mdf;Integrated Security=True;User Instance=True
```

والذي سنناقشه لاحقاً . نضغط على Next ، تظهر نافذة لنقرر إن كنا نريد تخزين هذه الاتصال وما الاسم الذي نريد أن نسميه به ، نختار الاسم MyConnectionString ثم نضغط على Next

Configure Data Source - SqlDataSource1

 **Save the Connection String to the Application Configuration File**

Storing connection strings in the application configuration file simplifies maintenance and deployment. To save the connection string in application configuration file, enter a name in the text box and then click Next. If you choose not to do this, the connection string is saved in the page as a property of the data source control.

**Do you want to save the connection in the application configuration file?**

☒ Yes, save this connection as:

< Previous   Next >   Finish   Cancel

تظهر نافذة جديدة وفيها الجداول الموجودة في قاعدة البيانات ، حيث سيتم عرض الجدول الوحيد الموجود في القاعدة التي أنشأناها ، نترك الإعدادات كما هي ونضغط على Next .

Configure Data Source - SqlDataSource1

**Configure the Select Statement**

How would you like to retrieve data from your database?

☐ Specify a custom SQL statement or stored procedure  
☒ Specify columns from a table or view

Name: MyTable

Columns:

|                                     |      |
|-------------------------------------|------|
| <input checked="" type="checkbox"/> | *    |
| <input type="checkbox"/>            | Id   |
| <input type="checkbox"/>            | Name |
| <input type="checkbox"/>            | Mark |

SELECT statement:  
SELECT \* FROM [MyTable]

☐ Return only unique rows  
 WHERE...  
 ORDER BY...  
 Advanced...

< Previous   Next >   Finish   Cancel

يتم عبر النافذة السابقة تحديد البيانات التي سيتم جلبها من قاعدة البيانات ، في القسم الأول نحدد إن كنا سنستقبل البيانات من إجراء مخزن Stored Procedure أو عبارة SQL نقوم ببنائها باستخدام Query Builder ، أما الخيار الثاني وهو الافتراضي ، فيقوم بعرض الجداول والمناظير Views الموجودة في قاعدة البيانات ليتم جلب البيانات منها . في القسم الثاني من النافذة نحدد الجدول أو المنظور المطلوب ثم نختار الأعمدة التي نريد جلبها . في القسم الثالث نلاحظ وجود عدة خيارات ، إن تفعيل الخيار Return only unique rows يؤدي إلى عدم حدوث تكرار في البيانات المسترجعة هذا إن كان مصدر البيانات يحتوي على أسطر مكررة ( عموماً إن وجود مفتاح أساسي يضمن عدم تكرار بيانات الجدول ) ، الزر WHERE لتحديد شرط يتم تطبيقه على البيانات وبالتالي يتم إرجاع الأسطر التي تحقق هذا الشرط . مثلاً لاسترجاع بيانات الطلاب الذين حصلوا على علامة أكبر من ٨٠ ، نضغط على الزر WHERE لتظهر لنا النافذة التالية :

**Add WHERE Clause**

Add one or more conditions to the WHERE clause for the statement. For each condition you can specify either a literal value or a parameterized value. Parameterized values get their values at runtime based on their properties.

Column: Mark

Operator: >=

Source: None

SQL Expression: [Mark] >= @Mark2

Value: 80

Parameter properties

Value: 80

WHERE clause:

| SQL Expression  | Value |
|-----------------|-------|
| [Mark] >= @Mark | 80    |

Buttons: Add, Remove, OK, Cancel

حيث نقوم بالبداية بتحديد العمود الذي سيتم تطبيق الشرط عليه ، ثم نحدد معامل الشرط ( يساوي ، لايساوي ، أكبر ، أصغر ... ) ، ثم نحدد مصدر القيمة التي سيتم استخدامها في الشرط ( مثلا من ملف cookies أو غير ) ، عند اختيار None يتوجب علينا إدخال قيمة ثابتة وبشكل مباشر وذلك ضمن الحقل Value . بعد تحديد جميع أقسام عبارة Where نضغط على الزر Add لتتم إضافة ذلك الشرط . ( إن البيانات المحددة في الصورة السابقة تقوم بإرجاع بيانات الطلاب الذين حصلوا على علامة أكبر أو تساوي ٨٠ ) . نغلق هذه النافذة لنعود إلى النافذة Configure Data Source .

الزر ORDER By وذلك لترتيب البيانات المسترجعة حسب عمود ما، مثلا استرجاع بيانات الطلاب مرتبين أبجدياً حسب الاسم .

Add ORDER BY Clause

Specify the columns you would like to order by.

Sort by

Name ☒ Ascending ☐ Descending

Then by

☒ Ascending ☐ Descending

Then by

☒ Ascending ☐ Descending

SELECT statement:

SELECT \* FROM [MyTable] ORDER BY [Name]

OK Cancel

Ascending تعني ترتيب تصاعدي سواء كان رقمي أو أبجدي ، Descending ترتيب تنازلي.  
 نغلق هذه النافذة لنعود إلى النافذة Configure Data Source .  
 الزر Advanced لتفعيل إمكانية إضافة وتعديل وحذف البيانات من وإلى المصدر المحدد .

Advanced SQL Generation Options

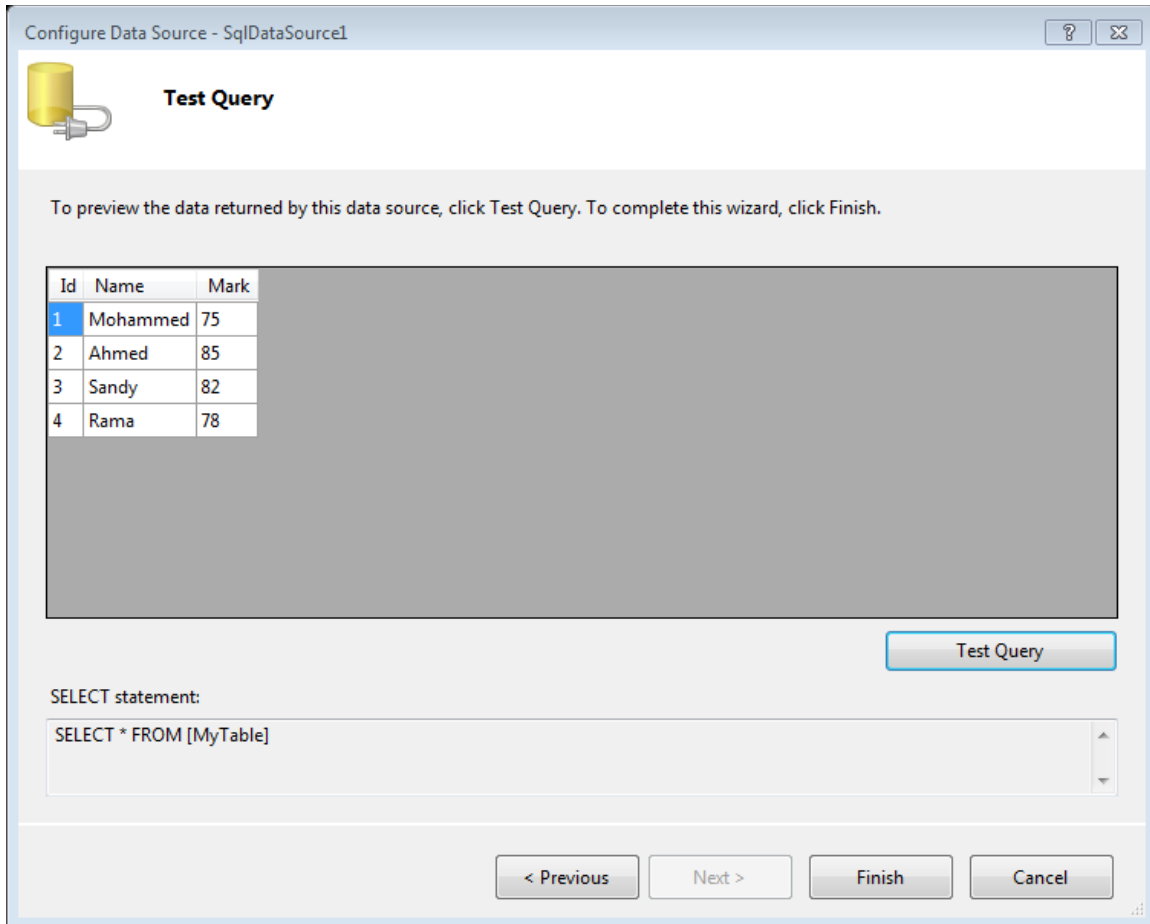
Additional INSERT, UPDATE, and DELETE statements can be generated to update the data source.

☒ **Generate INSERT, UPDATE, and DELETE statements**  
 Generates INSERT, UPDATE, and DELETE statements based on your SELECT statement. You must have all primary key fields selected for this option to be enabled.

☒ **Use optimistic concurrency**  
 Modifies UPDATE and DELETE statements to detect whether the database has changed since the record was loaded into the DataSet. This helps prevent concurrency conflicts.

OK Cancel

لا تتم بتفعيل الخيارين السابقين في هذا المثال ، نغلق هذه النافذة لنعود إلى النافذة Configure Data Source . ونضغط على الزر Next ثم في النافذة التي تليها اضغط على Test Query . للتأكد من نجاح الاتصال مع قاعدة البيانات ثم اضغط على Finish .



وبهذا نكون قد أنهينا إعداد الأداة SqlDataSourec وأصبحت بمثابة وسيط للاتصال مع قاعدة البيانات وجلب المعلومات منها وتنفيذ الأوامر عليها كما سنرى في فقرات لاحقة ، ولكنها لا تقوم بعرض البيانات بمفردها . نفتح الكود الصفحة السابقة ، يجب أن يكون مشابهاً للكود التالي :

## كود ASP.net

```
<div>

    <asp:SqlDataSource ID="SqlDataSource1" runat="server"
        ConnectionString="<%"$ ConnectionStrings:MyConnectionString %">
        SelectCommand="SELECT * FROM [MyTable]"></asp:SqlDataSource>

</div>
```

نلاحظ من الكود السابق أن الخاصية `ConnectionString` قد أخذت القيمة `MyConnectionString` ولكن أين تم تخزين عبارة الاتصال هذه ؟

يتم تخزين إعدادات الاتصال ضمن الملف `web.config` وذلك لعدة أسباب :

- ✓ الأمن و الحماية .
- ✓ سهولة التعامل : بالتأكد فإن إضافة عبارة الاتصال ضمن ملف واحد أسهل بكثير من إضافتها إلى كل صفحة ضمن الموقع ، وبالتالي عندما نريد إجراء تعديل ما عليها فإننا نعدل في مكان واحد فقط .
- ✓ الأداء : حيث يساهم تخزين عبارة الاتصال في الملف `web.config` بأداء أفضل لمجمع الاتصالات على السيرفر ( Connections Pool ) .

نفتح الملف `web.config` لنجد فيه الكود التالي :

#### كود XML

```
<configuration>

  <connectionStrings>
    <add name="MyConnectionString"
      connectionString="Data Source=.\SQLEXPRESS;AttachDbFilename=
        |DataDirectory|\MyDatabase.mdf;Integrated Security=True;User
        Instance=True"
      providerName="System.Data.SqlClient" />
  </connectionStrings>

  <system.web>
    <compilation debug="false" targetFramework="4.0" />
  </system.web>

</configuration>
```

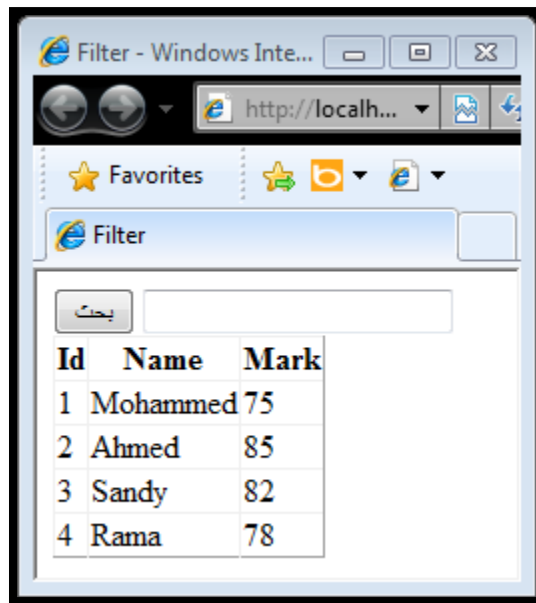
إن عبارة الاتصال السابقة معدة للعمل فقط على الجهاز المحلي `Local` ، ويجب تعديلها عند رفع الموقع على الانترنت ، حيث تقوم شركة الاستضافة بتزويدك بعبارة الاتصال وفقاً لإعداداتهم الخاصة .

قم بإضافة أداة `GridView` إلى الصفحة التي أنشأناها مسبقاً ، واجعل قيمة الخاصية `DataSourceId="sqlDataSource1"` ، قم بتنفيذ الصفحة ليتم عرض بيانات الجدول ضمن الأداة `GridView` ( والتي سنتطرق للتعامل معها بالتفصيل في فصل لاحق ) .

## فلتر البيانات

تدعم الأداة SqlDataSource خاصية الفلتر FilterExpression والتي تمكننا من تحديد شرط ما على البيانات التي نرغب بالحصول عليها من قاعدة البيانات ، وهذا مандعوه بالفلتر ، بالعودة إلى الصفحة السابقة ، وعلى فرض أن عدد الطلاب أصبح كبيراً وأنت بحاجة للبحث عن طالب معين فكيف نقوم بذلك عبر الخاصية FilterExpression ؟

ما سنقوم بعمله هو صفحة مشابهة للصفحة التالية ، حيث يقوم المستخدم بكتابة اسم طالب ما ضمن الأداة TextBox1 ثم الضغط على زر " بحث " ليتم عرض بيانات الطلاب الذين يحملون ذلك الاسم .



أنشئ صفحة جديدة ، قم بإضافة أداة TextBox1 و Button1 ، أضف أداة SqlDataSource واربطها مع الجدول MyTable كما تعلمنا في الفقرة السابقة ، انتقل إلى وضع الكود وقم بالإضافات التالية :

## كود ASP.net

```
<div>

<asp:SqlDataSource ID="SqlDataSource1" runat="server"
    ConnectionString="<%$ ConnectionStrings:MyConnectionString %>"
    SelectCommand="SELECT * FROM [MyTable]"
    FilterExpression="Name LIKE '{0}%'" >
    <FilterParameters>
        <asp:ControlParameter Name="stdName" ControlID="TextBox1"
            PropertyName="TEXT" />
    </FilterParameters>
</asp:SqlDataSource>

<input type="button" value="بحث" />

<table border="1">
| Id | Name | Mark |
| --- | --- | --- |
| 1 | Mohammed | 75 |
| 2 | Ahmed | 85 |
| 3 | Sandy | 82 |
| 4 | Rama | 78 |


</div>
```

```

</FilterParameters>
</asp:SqlDataSource>
<asp:Button ID="Button1" runat="server" Text="بحث" />
<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
<br />
</div>

```

شرح الكود المضاف :

قمنا في البداية بإضافة خاصية الفلتر FilterExpression وأعطيناها القيمة: '0' Name LIKE وهو عبارة عن تعبير SQL ، لاختيار الأسماء المشابهة للبارامتر الممرر (أي عملية فلتر على أساس الاسم) ، أما الرمز '0' فيدل على رقم البارامتر الذي سيتم استخدامه (حيث يبدأ الترقيم ابتداءً من الصفر) . قمنا بعد ذلك بتحديد البارامتر (وهو القيمة المدخلة في TextBox1) حيث صرحنا بالبداية عن قسم بارامترات عبر الوسم <FilterParameters> ثم قمنا بإضافة بارامتر من النوع أداة تحكم وإعطائه الاسم "stdName" ثم قمنا بتحديد الأداة التي سنأخذ منها البارامتر وهي TextBox1 وتحديد الخاصية المعنية وهي Text .

أخيراً ، قم بإضافة الأداة GridView واجعل قيمة الخاصية DataSourceId تساوي القيمة SqlDataSource1 ، قم بتنفيذ الصفحة ، واكتب حرف ما ثم اضغط على الزر بحث ليتم عرض جميع الطلاب الذين تبدأ أسمائهم بذلك الحرف ، جرب كتابة اسم طالب بالكامل وابحث .

#### ملاحظة

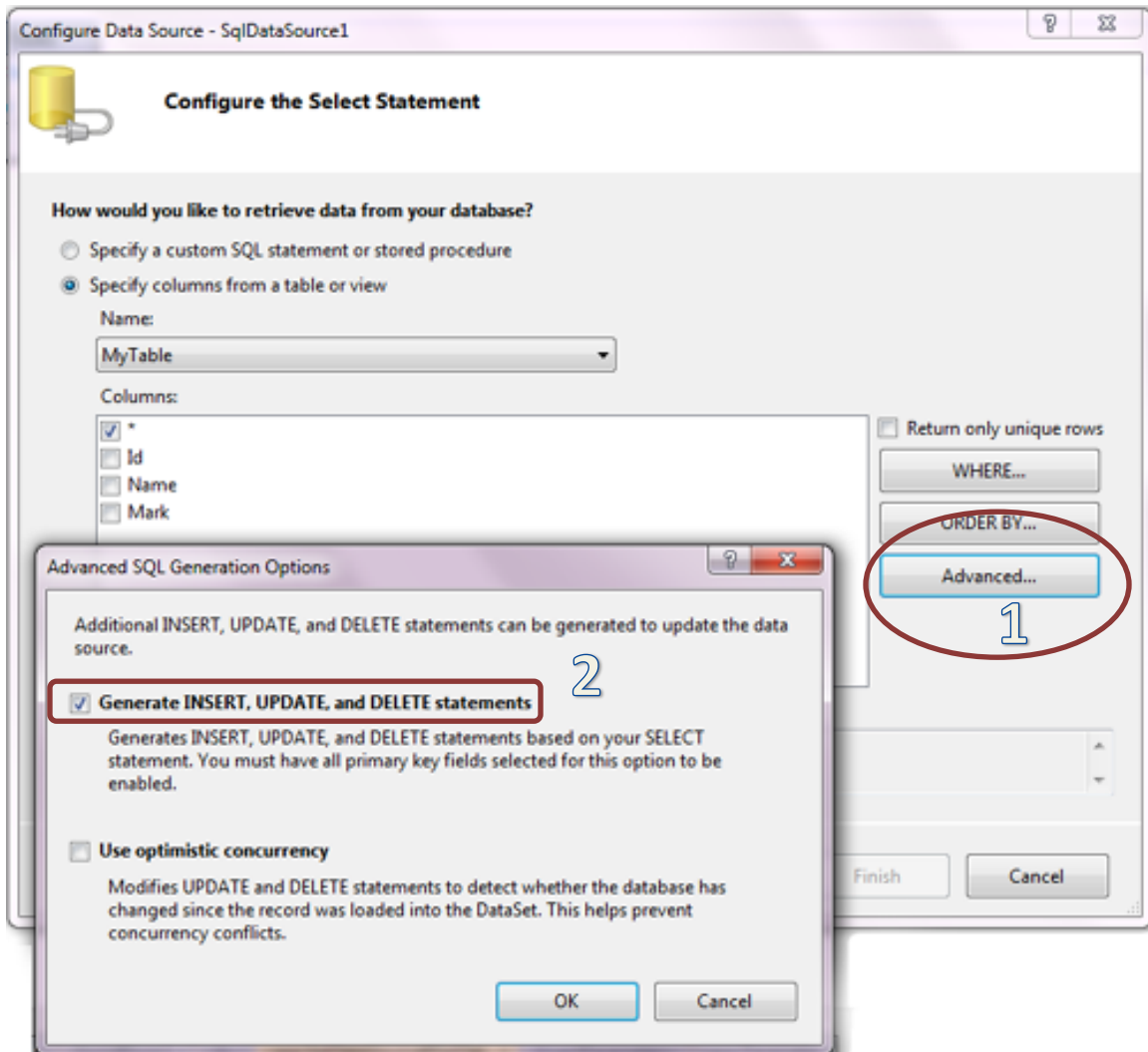
في الصفحة السابقة قمنا بتحديد نوع البارامتر على أنه أداة تحكم <asp:ControlParameter.. وذلك لأننا سنقوم بكتابة البارامتر ضمن أداة تحكم (وقد كانت TextBox) ، يوجد العديد من أنواع البارامترات الأخرى والتي يمكننا من أخذ قيمة البارامتر من ملف Cookies أو من ملف الجلسة Session أو من عنوان الصفحة QueryString وخيارات أخرى وسنتحدث عنها في فصول لاحقة إن شاء الله .

#### العمليات التي تدعمها الأداة SqlDataSource

تدعم الأداة SqlDataSource العمليات الأربع الأساسية التي يتم تنفيذها على البيانات : إضافة ، حذف ، تعديل ، اختيار . وذلك عبر الخصائص التالية :

- InsertCommand ✓
- DeleteCommand ✓
- UpdateCommand ✓
- SelectCommand ✓

سنقوم بتطبيق مثال بسيط للتعرف على الخصائص السابقة ، أضف صفحة جديدة وأضف عليها أداة SqlDataSource وقم بربطها مع الجدول MyTable كما تعلمنا في الفقرة السابقة ، ولكن قم بتفعيل الخيار Generate Insert, Update, and Delete statements كما في الصورة التالية



ثم نكمل باقي الخطوات كالمعتاد ، بعد ذلك يصبح كود الصفحة بالشكل التالي :

كود ASP.net

```
<div>
  <asp:SqlDataSource ID="SqlDataSource1" runat="server"
    ConnectionString="<%"$ ConnectionStrings:MyConnectionString %>"
    DeleteCommand="DELETE FROM [MyTable] WHERE [Id] = @Id"
    InsertCommand="INSERT INTO [MyTable] ([Id], [Name], [Mark]) VALUES
      (@Id, @Name, @Mark)"
    SelectCommand="SELECT * FROM [MyTable]"
```

```

UpdateCommand="UPDATE [MyTable] SET [Name] = @Name, [Mark] = @Mark
WHERE [Id] = @Id">
<DeleteParameters>
  <asp:Parameter Name="Id" Type="Int32" />
</DeleteParameters>
<InsertParameters>
  <asp:Parameter Name="Id" Type="Int32" />

  <asp:Parameter Name="Name" Type="String" />
  <asp:Parameter Name="Mark" Type="Int32" />
</InsertParameters>
<UpdateParameters>
  <asp:Parameter Name="Name" Type="String" />
  <asp:Parameter Name="Mark" Type="Int32" />
  <asp:Parameter Name="Id" Type="Int32" />
</UpdateParameters>
</asp:SqlDataSource>
</div>

```

حيث نلاحظ إضافة عبارات SQL الخاصة بعمليات الإضافة الحذف والتعديل والاستعلام من الجدول المحدد وإسناد كل عبارة للخاصية المقابلة لها في الأداة SqlDataSource ، كما يظهر التصريح عن البارامترات التي يجب تمريرها لعبارات SQL ، فعندما نريد حذف سجل ما من الجدول يكفي تمرير Id ذلك السجل ، ولذلك فإن القسم <DeleteParameters> يحتوي على بارامتر واحد فقط ، أما عندما نريد إضافة سجل جديد فنحن بحاجة لكامل القيم وهي في مثالنا هذا ثلاث قيم ( Id, Name, Mark ) ولذلك تم تمرير ثلاث بارامترات لعبارة الإضافة Insert وذلك ضمن القسم <InsertParameter> . لرؤية كيفية عمل الخصائص السابقة نقوم بإضافة الأداة DetailsView إلى الصفحة ونضبط خصائصها كالتالي :

| الخاصية                  | القيمة         |
|--------------------------|----------------|
| DataSourceID             | SqlDataSource1 |
| AutoGenerateDeleteButton | True           |
| AutoGenerateEditButton   | True           |
| AutoGenerateInsertButton | True           |
| AutoGenerateRows         | False          |
| AllowPaging              | True           |

لا تقلق من الأداة DetailsView فسنتأتي على شرحها وبالتفصيل في فصل قادم إن شاء الله . إنما الهدف من استخدامها هنا فقط لتوضيح عمل خصائص الأداة SqlDataSource ، بعد ضبط قيم الخصائص السابقة كما هو محدد بإمكاننا تنفيذ الصفحة وتجربة إضافة سجلات جديدة على الجدول وحذف والتعديل على سجلات سابقة . أما كود الصفحة فيصبح كاملاً بالشكل التالي :

## كود ASP.net

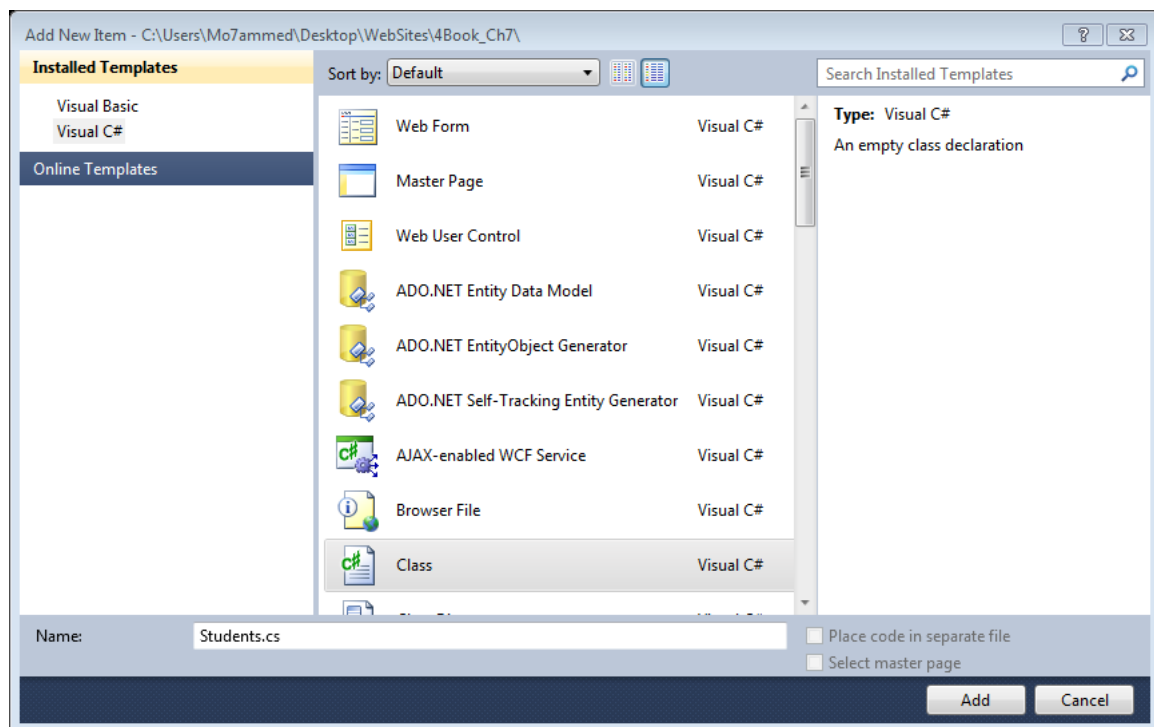
```
<div>
  <asp:SqlDataSource ID="SqlDataSource1" runat="server"
    ConnectionString="<%= $ ConnectionStrings:MyConnectionString %>"
    DeleteCommand="DELETE FROM [MyTable] WHERE [Id] = @Id"
    InsertCommand="INSERT INTO [MyTable] ([Id], [Name], [Mark]) VALUES
      (@Id, @Name, @Mark)"
    SelectCommand="SELECT * FROM [MyTable]"
    UpdateCommand="UPDATE [MyTable] SET [Name] = @Name, [Mark] = @Mark
      WHERE [Id] = @Id">
    <DeleteParameters>
      <asp:Parameter Name="Id" Type="Int32" />
    </DeleteParameters>
    <InsertParameters>
      <asp:Parameter Name="Id" Type="Int32" />
      <asp:Parameter Name="Name" Type="String" />
      <asp:Parameter Name="Mark" Type="Int32" />
    </InsertParameters>
    <UpdateParameters>
      <asp:Parameter Name="Name" Type="String" />
      <asp:Parameter Name="Mark" Type="Int32" />
      <asp:Parameter Name="Id" Type="Int32" />
    </UpdateParameters>
  </asp:SqlDataSource>
  <asp:DetailsView ID="DetailsView1" runat="server"
    DataSourceID="SqlDataSource1"
    AutoGenerateDeleteButton="True"
    AutoGenerateEditButton="True"
    AutoGenerateInsertButton="True"
    AutoGenerateRows="False" AllowPaging="True" DataKeyNames="Id" >
    <Fields>
      <asp:BoundField DataField="Id" HeaderText="Id" ReadOnly="True"
        SortExpression="Id" />
      <asp:BoundField DataField="Name" HeaderText="Name"
        SortExpression="Name" />
      <asp:BoundField DataField="Mark" HeaderText="Mark"
        SortExpression="Mark" />
    </Fields>
  </asp:DetailsView>
</div>
```

قم بتنفيذ الصفحة السابقة ، وجرب إجراء العمليات المختلفة على البيانات ، قم بتعديل المثال السابق واستخدم الأداة GridView عوضاً عن DetailsView .

### التعامل البرمجي مع قواعد البيانات

من النقاط الهامة عند بناء موقع ديناميكي هي معرفة إدارة قواعد البيانات بشكل برمجي ، على العموم فإن الكود البرمجي سيكون مألوف لمبرمجي تطبيقات .NET ، سأكتفي خلال هذه الفصل بعرض مثالين الأول عن كيفية الإضافة لقاعدة البيانات ، والثاني الاستعلام عن البيانات ، وبالتالي سيكون بإمكان المطور أن يكتب توابع التعديل والحذف وغيرها بسهولة كبيرة فالموضوع يتعلق بلغة SQL أكثر من تعلقه بتقنية الـ .NET.

سنقوم بإضافة صف جديد ( Class ) ، من القائمة File اختر New ثم File ( أو ببساطة اضغط على CTRL+N ) ، ثم حدد Class وقم بتسميته Students.cs ( Students.vb لمبرمجي الفيجوال بيسك ) .



ستظهر رسالة تخبرك بأنه سيتم وضع الصف الجديد داخل مجلد يدعى App\_Code فاضغط على نعم ، هذا الصف الجديد سيحتوي على التوابع التي تقوم بالتعامل مع قواعد البيانات ، حيث من الأفضل عدم كتابتها في كود الصفحة .

الخطوة الأولى تكون في استدعاء فضاءات العناوين المسؤولة عن التعامل مع قواعد البيانات .

## كود C#

```
using System.Data;
using System.Data.SqlClient;
using System.Configuration;
```

## كود VB

```
Imports System.Data
Imports System.Data.SqlClient
Imports System.Configuration
```

إن فضاء العناوين System.Configuration ليس للتعامل مع قواعد البيانات ، إنما يستخدم لضبط إعدادات الموقع والتعامل مع ملف الإعدادات web.config . والذي يحتوي على عبارة الاتصال مع قاعدة البيانات .

نقوم بتهيئة باني الصف ليصبح الكود بالشكل التالي :

## كود C#

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.Configuration;

public class Students
{
    SqlConnection con;
    SqlCommand myCommand;
    string connectionString;

    public Students()
    {
        con = new SqlConnection();
        connectionString = ConfigurationManager
            .ConnectionStrings["MyConnectionString"]
            .ConnectionString;
        con.ConnectionString = connectionString;
        myCommand = new SqlCommand();
        myCommand.Connection = con;
    }
}
```

## كود VB

```
Imports System.Data
Imports System.Data.SqlClient
Imports System.Configuration

Public Class Students

    Private con As SqlConnection
    Private myCommand As SqlCommand
    Private connectionString As String

    Public Sub New()
        con = New SqlConnection()
        connectionString = ConfigurationManager
            .ConnectionStrings("MyConnectionString")
            .ConnectionString
        con.ConnectionString = connectionString
        myCommand = New SqlCommand()
        myCommand.Connection = con
    End Sub

End Class
```

حيث إن عبارة الاتصال مع قاعدة البيانات MyConnectionString يجب أن تكون مخزنة في الملف web.config .

## كود XML

```
<configuration>
  <connectionStrings>
    <add name="MyConnectionString"
      connectionString="Data Source=.\SQLEXPRESS;AttachDbFilename=
|DataDirectory|\MyDatabase.mdf;Integrated Security=
True;User Instance=True"
      providerName="System.Data.SqlClient"/>
  </connectionStrings>
  ...
  ...
</configuration>
```

بالإمكان الاستعانة بالأداة SqlDataSource لتقوم بتخزين عبارة الاتصال في الملف السابق دون عناء كتابتها بشكل يدوي كما تعلمنا في فقرات سابقة .

نقوم الآن بكتابة التابع المسؤول عن إضافة البيانات إلى الجدول MyTable بالشكل التالي :

#### كود C#

```
public string addStudents(string stdName, int stdMark)
{
    try
    {
        string mysql = "insert into myTable(Name,Mark)
                        values ( @sname, @smark )";
        myCommand.Parameters.Clear();
        myCommand.Parameters.AddWithValue("@sname", stdName);
        myCommand.Parameters.AddWithValue("@smark", stdMark);
        myCommand.CommandText = mysql;

        con.Open();
        int recs;
        recs = myCommand.ExecuteNonQuery();
        con.Close();
        return recs + " row added successfully.";
    }
    catch (Exception ex)
    {
        if (con.State != ConnectionState.Closed)
            con.Close();
        return ex.Message;
    }
}
```

#### كود VB

```
Public Function addStudents(ByVal stdName As String, ByVal stdMark As Integer) As String
    Try
        Dim mysql As String = "insert into myTable(Name,Mark) values ( @sname,
@smark )"
        myCommand.Parameters.Clear()
        myCommand.Parameters.AddWithValue("@sname", stdName)
        myCommand.Parameters.AddWithValue("@smark", stdMark)
        myCommand.CommandText = mysql

        con.Open()
        Dim recs As Integer
        recs = myCommand.ExecuteNonQuery()
        con.Close()
        Return recs & " row added successfully."
    Catch ex As Exception
```

```

        If con.State <> ConnectionState.Closed Then
            con.Close()
        End If
        Return ex.Message
    End Try
End Function

```

الآن أصبح بإمكاننا إنشاء الصفحة التي نقوم باستدعاء التابع السابق . أضف صفحة جديدة و أضف عليها أداتين من النوع TextBox و زر واحد ( Button ) وأداة Lable واحدة ، وفي حدث النقر على الزر اكتب الكود التالي :

#### كود C#

```

protected void Button1_Click(object sender, EventArgs e)
{
    Students std = new Students();
    string name=TextBox1.Text;
    int mark=int.Parse(TextBox2.Text);
    string result= std.addStudents(name, mark);
    Label1.Text = result;
}

```

#### كود VB

```

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

    Dim std As New Students()
    Dim name As String = TextBox1.Text
    Dim mark As Integer = Integer.Parse(TextBox2.Text)
    Dim result As String = std.addStudents(name, mark)
    Label1.Text = result
End Sub

```

حيث قمنا بإنشاء غرض std من الصف Students ثم قمنا بتنفيذ التابع addStudents وبهذا نبقى كود الصفحة خاليا من الأكواد التي تتعامل بشكل مباشر مع قاعدة البيانات .

كود الصفحة :

```
<div>
    Nmae : <asp:TextBox ID="TextBox1" runat="server" /> <br />
    Mark : <asp:TextBox ID="TextBox2" runat="server" /> <br />
    <asp:Button ID="Button1" runat="server" Text="add"
        onclick="Button1_Click" />

    <hr />
    <asp:Label ID="Label1" runat="server" Text="" />
</div>
```

قم بتنفيذ الصفحة و راقب قاعدة البيانات . يمكن تنفيذ عمليات التعديل و الحذف بشكل مشابه جدا للتابع السابق مع فرق تغيير عبارات الـ SQL .

نعود الآن إلى الصف Students لكتابة تابع يقوم بالاستعلام من قاعدة البيانات ، حيث نمرر له رقم الطالب كبارامتر ويعيد لنا العلامة التي حصل عليها .

```
int getStudentMark(string stdId)
{
    int mark = -1;
    try
    {
        string mysql= "select mark from MyTable WHERE id = @id ";
        myCommand.Parameters.Clear();
        myCommand.Parameters.AddWithValue("@id", stdId);
        myCommand.CommandText = mysql;

        con.Open();
        SqlDataReader dr;
        dr = myCommand.ExecuteReader();
        while (dr.Read())
        {
            mark = int.Parse(dr.GetValue(0).ToString());
        }
        con.Close();
    }
    catch (Exception ex)
    {
        if (con.State != ConnectionState.Closed)
            con.Close();
    }
    return mark;
}
```

