

## الفصل العاشر استخدام كائنات ADO

إذا كنت ممن استخدموا أحد إصدارات **Visual Basic** القديمة، فلا شك أنك تعلم الكثير عن كائنات **ADO** المستخدمة في وصل تطبيقاتك بقواعد البيانات المختلفة. سنقوم في هذا الفصل بالتعرف على خصائص وعناصر كائنات **ADO** حتى تتمكن من استخدام الكائنات القديمة ولأخذ فكرة عامة عن عمل قواعد البيانات. بانتهاء هذا الفصل، ستتعرف على:

- استخدام كائنات **ADO**.
- استخدام الكائن **Connection**.
- العمليات الأساسية لمجموعة السجلات.
- تغيير البيانات داخل مجموعة البيانات.
- استخدام الكائن **Command**.
- مجموعات السجلات المنفصلة.

تعرفنا في الفصل السابق على أساسيات الوصول إلى قواعد البيانات وإنشاء عبارات SQL. وهذا الفصل هو أول الفصول التي سنقوم فيها بالتعرف على كائنات بيانات **ActiveX** أو **ActiveX Data Objects (ADO)** وهي الوسيلة التي يتم من خلالها الوصول إلى جميع أنواع قواعد البيانات من خلال كود **Visual Basic**. وبدأ من الإصدار **Visual Basic.NET**، قامت مايكروسوفت باستبدال **ADO** بمفهوم جديد يتشابه كثيراً مع المفهوم السابق وأسمته **ADO.NET** مثلما انتقلنا من قبل من **ASP** إلى **ASP.NET** وأصبح كل شيء الآن بطعم **.NET**. ولكن على الرغم من ذلك، فإن **ADO** هي التقنية الأساسية التي تم منها اشتقاق **ADO.NET** ومازال هناك العديد من البرامج التي تستخدم هذه التقنية، وهذا هو السبب الأول الذي جعلنا نتحدث عنها رغم استبدالها. أما السبب الثاني فلأن العديد من المبادئ المستخدمة للوصول إلى قاعدة البيانات واستخدامها من خلال **ADO** هي نفسها المستخدمة في **ADO.NET**. لذا يهدف هذا الفصل أساساً إلى تعلم التفاعل مع قواعد البيانات بغض النظر عن التقنية المستخدمة. كانت هذه خلاصة الفصل، والآن إلى التفاصيل.

عند تطوير التطبيقات الجديدة، يفضل استخدام **ADO.NET** للوصول إلى البيانات داخل **Visual Basic 2012**، وهو ما سنتعرف عليه في الفصل القادم إن شاء الله.

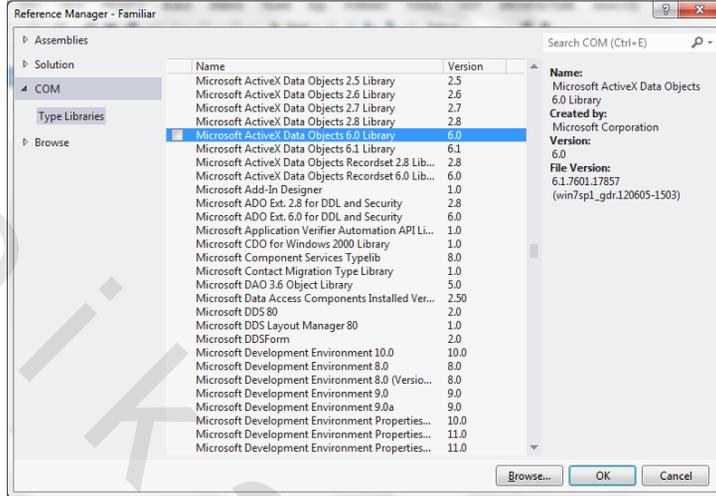


## استخدام كائنات ADO

كما ذكرنا فقد ظهرت **ADO** على الساحة قبل ظهور **.NET**. وأصبحت جزءاً أساسياً من العديد من مواقع الويب وتطبيقات **Visual Basic**. وكما يتم مع أي كائن خارجي، يجب أن تقوم بإضافة مرجع إلى **ADO** قبل أن تقوم باستخدامه (انظر شكل ١٠-١). وبمجرد إضافتك لهذا المرجع، يمكنك استخدام الكائنات التي يحتوى عليها نموذج **ADO** داخل تطبيقك كما لو كانت جزءاً منه.

يوضح جدول ١٠-١ التالي الكائنات المختلفة الموجودة داخل نموذج **ADO**.

## الفصل العاشر : استخدام كائنات ADO



شكل ١٠-١ يمكنك تضمين كائنات ADO داخل VisualStudio 2012 من التبويب COM بالمربع

### الحوارى Reference Manager

جدول ١٠-١ كائنات نموذج ADO

الكائن	الاستخدام
Recordset	يحتوى على السجلات التى منها نحصل على نتيجة الاستعلام
Connection	يتيح التحكم فى الاتصال مع مصدر البيانات
Command	يقوم بتنفيذ أوامر واستعلامات قاعدة البيانات
Error	يقوم باسترجاع الأخطاء من ADO
Field	يمثل جزء من البيانات داخل مجموعة السجلات Recordset
Parameter	يعمل مع الكائن Command لتعيين معامل داخل استعلام أو إجراء مخزن
Property	يتيح لك الوصول إلى خصائص كائن ADO

يحتوي المرجع **Microsoft ActiveX Data Objects 6.x Library** على جميع الكائنات الموجودة بالجدول السابق، بينما يحتوي المرجع **Microsoft ActiveX Data Objects Recordset6.x Library** على الكائن **Recordset** فقط والذي سنتعرف عليه بالتفصيل في هذا الفصل.



عند إنشاء الكائنات الموجودة بالجدول ١٢-١ السابق، يمكنك استخدام البادئة **ADODB** كما في الكود التالي:

```
Dim con AS ADODB.Connection
```

سنقوم في الأجزاء التالية بالتعرف على بعض خصائص ووظائف وأحداث كائنات **ADO**. يمكنك التعرف على الوثيقة الكاملة من **ADO** وكذلك التحديثات والأمثلة المختلفة من العنوان التالي على شبكة الإنترنت:



[www.microsoft.com/data/ado](http://www.microsoft.com/data/ado)

## استخدام الكائن Connection

كي تستطيع التعامل مع البيانات الموجودة داخل قاعدة البيانات، يجب أن تقوم أولاً بإنشاء الاتصال مع قاعدة البيانات وذلك من خلال الكائن **Connection** الذي يساعدك في الاتصال بقاعدة البيانات أو الانفصال عنها. والخطوة الأولى في إنشاء الاتصال هي إنشاء حالة جديدة من هذا الكائن كما في الكود التالي:

```
Dim con AS ADODB.Connection  
Set con = New ADODB.Connection
```

وكل ما تحتاج إليه بعد إنشاء هذه الحالة هو تحديد معاملات الاتصال واستدعاء الوظيفة **Open()**، حيث يمكنك تحديد معاملات الاتصال بطريقتين، الأولى بتخصيص هذه المعاملات للخاصية **ConnectionString** كما في الكود التالي:

```
Dim strInfo AS String  
strInfo = "User ID=sa;Password=;Database=Books;Server=  
bsw2k\NetSDK;Provider=SQLOLEDB"  
con.ConnectionString = strInfo  
con.Open()
```

في هذا الكود قمنا أولاً بتخصيص معاملات الاتصال إلى سلسلة من البيانات ثم تخصيص هذه السلسلة إلى الخاصية `ConnectionString` وأخيراً استدعاء الوظيفة `Open()`. أما الطريقة الثانية لتحديد معاملات الاتصال فتكون بتخصيص هذه المعاملات مباشرة إلى الوظيفة `Open()` كما يلي:

```
con.Open "UID=sa;PWD=;DATABASE=Books; SERVER=
bsw2k;DRIVER={SQL Server}"
```

تحتوي بعض كائنات ADO الأخرى مثل الكائن `Command` على الخاصية `Connection` التي تقبل إما سلسلة بيانات الاتصال أو الكائن `Connection`.



باتباع أي من الطريقتين السابقتين تكون قد حققت الاتصال بقاعدة البيانات المذكورة. وبعد الانتهاء من استخدام الاتصال، يمكنك الانفصال عن قاعدة البيانات من خلال الوظيفة `Close()` كما يلي:

```
con.Close()
```

انتبه أثناء تنفيذ الأمثلة الواردة في هذا الكتاب إلى تغيير اسم الحاسب وكلمة المرور وقاعدة البيانات للتوافق مع الإعدادات الموجودة على حاسبك.



### مفهوم معاملات الاتصال

كما رأيت من المثال السابق فإن معاملات الاتصال (أو سلسلة الاتصال) تحتوي على العديد من المعلومات والتي تشتمل على معرف المستخدم `User ID` وكلمة المرور `Password` واسم قاعدة البيانات `Database Name` حيث يتم كتابة هذه المعاملات في صورة أزواج من الأسماء والقيم مفصولةً بعلامة الفاصلة المنقوطة ";". وعند إنشاء معاملات الاتصال تحتاج إلى تعيين المعلومات التالية:

- نوع قاعدة البيانات التي ترغب في الاتصال بها مثل `SQL Server` و `Access` و `Oracle`. ويمكنك عند إجراء الاتصال بواسطة ADO تعيين اسم مشغل ODBC

من خلال قيمة المعامل **Driver** أو تعيين اسم مزود **OLE DB** من خلال قيمة المعامل **Provider**.

- بيانات عملية الدخول لقاعدة البيانات **Sign-in** (إن وجدت) والتي تشتمل على قيم اسم المستخدم **User ID** (أو **UID**) وكلمة المرور **Password** (أو **PWD**).
- موقع قاعدة البيانات والذي قد يشتمل على قيمة للخادم **Server** وقيمة لقاعدة البيانات **Database** وذلك في حالة **SQL Server** أو اسم قاعدة البيانات فقط في حالة قاعدة بيانات **Microsoft Access**.

تحتوي **ADO.NET** على بعض التصنيفات المصممة خصيصاً للاتصال بخادم **Microsoft SQL** وفي هذه الحالة لست مطالباً بتعيين المزود **Provider** المستخدم.



يوضح الكود التالي بعض الأمثلة على معاملات اتصال **ADO**:

- “**Provider=Microsoft.Jet.OLEDB.3.51;Data Source=e:\Books\Books.mdb**”
- “**Driver=Microsoft Access Driver (\*;/mdb);DBQ=e:\Books\Books.mdb**”
- “**Provider=SQLOLEDB;Password=ledo;User ID=Waleed;Server=SQLSRV1;Database=Books**”
- **User ID=Waleed;PWD=ledo;DATABASE=Books;SERVER=localhost;DRIVER={SQL Server}**

وعن هذا الكود، نوضح ما يلي:

- في المثال الأول تم تعيين معاملات الاتصال بقاعدة بيانات **Access** باستخدام مزود **OLEDB**.
- في المثال الثاني تم تعيين معاملات الاتصال بقاعدة بيانات **Access** باستخدام مشغل **OLEDB**.

- في المثال الثالث تم تعيين معاملات الاتصال بقاعدة بيانات SQL Server باستخدام مزود OLEDB.
- في المثال الأخير تم تعيين معاملات الاتصال بقاعدة بيانات SQL Server باستخدام مشغل SQL ODBC.

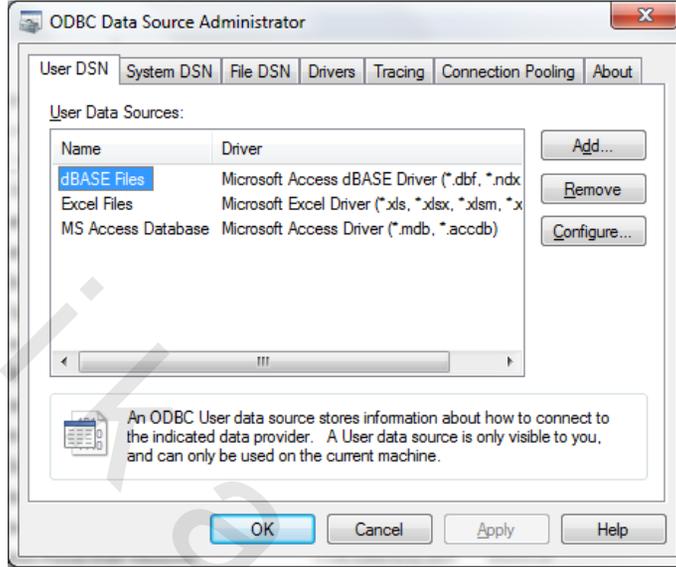
أحد الخيارات الهامة عند إنشائك لمعامل الاتصال هو استخدام مشغل ODBC أو مزود OLE DB وذلك بتعيين قيمة للمعامل Driver أو المعامل Provider وليس لكلاهما معاً، حيث تعتبر ODBC اختصاراً للعبارة Open Database Connectivity وهي تقنية قياسية لمشغلات قواعد البيانات الموجودة منذ سنوات. أما OLE DB فهي أحدث من سابقتها، لذا يوصى باستخدامها دائماً قدر الإمكان.

#### استخدام اسم مصدر البيانات

على الرغم من عدم استخدامها بكثرة كما كانت من قبل، يمكنك استخدام اسم مصدر البيانات (Data Source Name (DSN ضمن سلسلة الاتصال لتعيين مصدر بيانات ODBC كما في الكود التالي:

```
Con.Open"DSN=LocalServer;UID=Waleed;PWD=ledo"
```

وكما ترى فإن سلسلة الاتصال في هذا الكود بسيطة للغاية. فهي تحتوى فقط على DSN واسم المستخدم وكلمة المرور حيث تم تخزين بقية البيانات على مصدر بيانات ODBC المعروف على الحاسب. لإعداد مصدر بيانات ODBC على حاسبك، افتح نافذة لوحة التحكم Control Panel ثم انقر المجموعة Administrative Tools نقرأ مزدوجاً ثم انقر رمز Data Sources من النافذة الناتجة نقرأ مزدوجاً، تظهر نافذة ODBC Data Source Administrator التي يمكنك استخدامها في إدخال بيانات الاتصال حيث يتم تخصيص System DSN لجميع المستخدمين بينما يختص User DSN بالمستخدم الحالي فقط (انظر شكل ١٠-٢).



شكل ٢-١٠ تعيين بيانات الاتصال بمصدر البيانات ODBC

وعلى الرغم من بساطة سلسلة اتصال مصدر بيانات ODBC إلا أنك ملزم بإجراء العديد من الخطوات حتى يعمل برنامجك على أحد الحاسبات.

### استخدام الوظيفة *Execute()*

يحتوي الكائن **Connection** على الوظيفة *Execute()* التي تستخدم في تشغيل عبارة **SQL** على مصدر البيانات. وإذا قامت هذه العبارة باسترجاع سجلات من مصدر البيانات، يمكنك استخدام هذه السجلات ببساطة شديدة عن طريق تخصيص القيمة المرتجعة من

الوظيفة *Execute()* إلى كائن مجموعة سجلات **Recordset Object**.

عند العمل مع **ADO**، يمكنك الوصول إلى نفس الهدف من خلال أكثر من طريقة. ومثال ذلك عملية استرجاع البيانات التي يمكنك تنفيذها عن طريق أي من الكائنات **Recordset** و **Command** و **Connection** حيث يحتوي كل منهم على وظيفة لاسترجاع البيانات من قاعدة البيانات.



لتوضيح كيفية استخدام الوظيفة (**Execute()**) في استرجاع البيانات، قم أولاً بإنشاء قاعدة بيانات من خلال برنامج **Microsoft Access** باسم **Students.mdb** وأضف لها جدول واحد بنفس الاسم يحتوى على ثلاثة حقول هي **St\_ID** و **St\_fname** و **St\_lname** (أو انسخ قاعدة البيانات من على القرص المدمج المرفق بالكتاب) ثم تابع معنا الخطوات الآتية:

١. افتح بيئة تطوير **Visual Studio 2012** ثم قم بإنشاء تطبيق نوافذى جديد باسم مناسب وليكن **Execute**.
٢. قم بإضافة مرجع لكائنات **ADO** كما أوضحنا من قبل (راجع شكل ١٠-١).
٣. قم بتعيين عنوان مناسب للنموذج من خلال الخاصية **Text** وليكن (**Execute()** **Example**).
٤. قم بإضافة مربع سرد من مربع الأدوات إلى النموذج وأعد تسميته إلى **IstStudents**.
٥. قم بإضافة زر أمر من مربع الأدوات إلى النموذج وأعد تسميته إلى **btnLoadList** وعنوانه إلى **Run Query**.
٦. قم بإضافة الكود التالى إلى إجراء حدث نقر الزر **btnLoadList**:

```
Private Sub btnLoadList_Click(sender As Object, e As EventArgs) Handles btnLoadList.Click
```

```
Dim strConnect As String  
Dim sSSN, sName As String  
Dim con As ADODB.Connection  
Dim rs As ADODB.Recordset
```

```
strConnect = "Provider= Microsoft.Jet.OLEDB.4.0; Data Source = d:\Students.mdb"
```

```
con = New ADODB.Connection()
```

```
con.Open(strConnect)
```

```
rs = con.Execute("Select * from Students")
```

```
While Notrs.EOF
```

```
sSSN = rs.Fields("St_id").Value.ToString
```

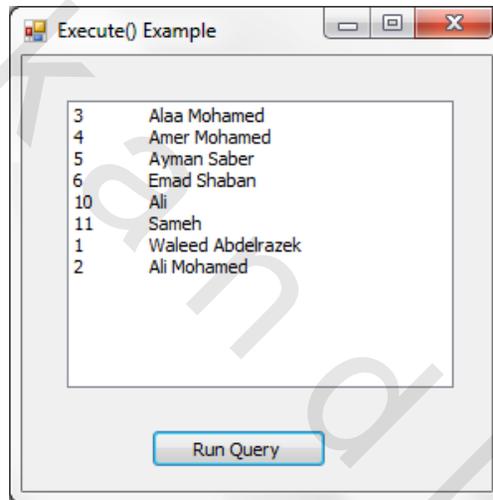
```
sName = rs.Fields("St_fname").Value.ToString.Trim& " "
```

```

&rs.Fields("St_Iname").Value.ToString
lstStudents.Items.Add(sSSN&vbTab&sName)
rs.MoveNext()
End While
rs.Close()
con.Close()
End Sub

```

٧. قم بتشغيل التطبيق ثم انقر الزر الموجود بالنموذج، تلاحظ امتلاء مربع السرد بالبيانات الموجودة بالجدول **Students** (انظر شكل ١٠-٣).



شكل ١٠-٣ نتيجة الاستعلام المنفذ من خلال الوظيفة **Execute**

تم في الكود السابق استدعاء الوظيفة **Execute()** التي تنتمي للكائن **Connection** حيث قامت بدورها بإرجاع مجموعة سجلات **Recordset** التي تم تخزينها بالمتغير **rs**. وبعد ذلك تم استخدام الدوارة **While** للتحرك خلال مجموعة السجلات وإضافة اسم كل طالب ورقمه إلى مربع السرد.

يمكنك أيضاً استخدام الوظيفة **Execute()** لتنفيذ عبارات **SQL** التي لا تقوم بإرجاع مجموعة سجلات كحذف السجلات أو إضافتها كما في الكود التالي الذي يقوم بحذف أحد السجلات:

```
con.Execute "Delete from Students where St_Iname = 'Mohamed' and St_fname = 'Alaa'"
```

### العمليات الأساسية لمجموعة السجلات

تعرفنا في المثال السابق على كيفية الحصول على البيانات من قاعدة البيانات ووضعها داخل كائن مجموعة السجلات RecordSet التي تحتوي بدورها على صفوف من البيانات الموجودة داخل جدول أو أكثر من جداول قاعدة البيانات وذلك من خلال تنفيذ أحد الاستعلامات. وعلى الرغم من أن الاستعلام ربما يشتمل على أكثر من جدول إلا أن مجموعة السجلات الناتجة تبدو بالنسبة للبرنامج وكأنها جدول واحد فقط يحتوي على أسماء الحقول وقيمها المختلفة، حيث تشكل كل مجموعة من الحقول سجل واحد، كما تتجمع السجلات كلها لتكوين مجموعة السجلات الناتجة.

سنقوم فيما يلي بالتعرف على كيفية إنشاء مجموعة سجلات Recordset وعرض سجلاتها من خلال كود Visual Basic.

### إنشاء مجموعة السجلات باستخدام الاستعلامات

تعلمنا منذ قليل كيفية إنشاء وملء الكائن Recordset باستخدام الوظيفة Connection.Execute() إلا أن هذا الكائن يحتوي على مجموعة من الوظائف والخصائص المستخدمة في استرجاع البيانات. وكما هو الحال مع جميع الكائنات، يجب أن تقوم أولاً بإنشاء حالة جديدة من الكائن Recordset قبل أن تقوم باستخدامه كما في الكود التالي:

```
Dim rsStudent As ADODB.Recordset  
rsStudent = New ADODB.Recordset
```

وبعد ذلك يمكنك استخدام خصائص هذا الكائن لتعريف الاتصال ومصدر السجلات ونوع مجموعة السجلات. لتعيين مصدر بيانات الكائن Recordset، قم بتخصيص كائن الاتصال أو سلسلة الاتصال إلى الخاصية ActiveConnection كما يلي:

```
rsStudent.ActiveConnection = con  
rsStudent.ActiveConnection = "DSN=Students"
```

حيث:

- يفترض السطر الأول من الكود أن **con** تمثل اتصال مفتوح يشير إلى مصدر البيانات كما أوضحنا من قبل.
- في السطر الثاني تم استخدام سلسلة الاتصال وفي هذه الحالة يتم إنشاء كائن الاتصال ضمناً.

يوضح الكود التالي استخدام الوظيفة **Recordset.Open()** لملء مجموعة السجلات **Recordset** بالبيانات حيث يتم أولاً إنشاء كائن **Recordset** جديد ثم إضافة بياناته إلى مربع السرد الموجود بالنموذج. قم بإضافة نموذج جديد إلى التطبيق الحالي بنفس مواصفات النموذج السابق وليكن باسم **Recordset.vb** وعنوان **Recordset.Open()** مع جعله نموذج بدء التطبيق ثم قم بإدخال الكود التالي في إجراء حدث نقر الزر الموجود بالنموذج:

```
Private Sub btnLoadList_Click(sender As Object, e As
EventArgs) Handles btnLoadList.Click
    Dim rsStudent As ADODB.Recordset
    Dim con As ADODB.Connection
    Dim strConnect As String
    Dim sSQL As String
    strConnect = "Provider = Microsoft.jet.OLEDB.4.0;Data
Source=d:\students.mdb"
    sSQL = "Select St_ID,St_fname,St_Iname from Students order
bySt_ID"
    con = New ADODB.Connection()
    con.Open(strConnect)
    rsStudent = New ADODB.Recordset()
    rsStudent.ActiveConnection = con
    rsStudent.Open(sSQL)

    While Not rsStudent.EOF
    lstStudents.Items.Add(rsStudent.Fields(0).Value.ToString&
" "&rsStudent.Fields(1).Value.ToString& " " &
rsStudent.Fields(2).Value.ToString)
    rsStudent.MoveNext()
    End While
```

```
rsStudent.Close()  
con.Close()  
End Sub
```

إذا قمت الآن بتشغيل هذا الكود، ستحصل على نفس النتيجة السابقة ولكن مع عرض الأسماء بترتيب أرقام الطلبة (انظر شكل ١٠-٤).



شكل ١٠-٤ عرض محتويات قاعدة البيانات من خلال الوظيفة Recordset.Open()

تأكد من اختيار المسار الصحيح لقاعدة البيانات بما يتوافق مع اسم وموقع قاعدة البيانات على حاسوبك.



### إظهار محتويات الحقول

استخدمنا في الكود السابق الأدلة في الحصول على بيانات الحقول من مجموعة حقول الكائن Recordset حيث قامت عبارة SQL بإرجاع ثلاثة حقول هي رقم الطالب والاسم الأول والاسم الأخير وتأخذ الدليل 0 والدليل 1 والدليل 2 على الترتيب. ولكن يمكنك استخدام أسماء الحقول بدلاً من الأدلة كما فعلنا مع الكائن Connection. وعلى ذلك يمكنك استرجاع قيمة الحقل الأول في الكود السابق باستخدام أي من الصيغتين التاليتين:

```
rsStudent.Fields("St_ID")  
rsStudent.Fields(0)
```

وهناك ملاحظة أخرى في الكود السابق وهي إرجاع قيمة الحقل ككائن من النوع **Field**. ففي الإصدارات القديمة من **Visual Basic** كان يكفي إرجاع قيم الحقول حيث كانت القيمة **Value** هي القيمة الافتراضية، أما **Visual Basic 2012** فيطلب تعريف الخاصية **Value** صراحةً. يستخدم الكود التالي دوارتين من النوع **For** لطباعة كل من أسماء الحقول وقيمها من داخل مجموعة السجلات على النافذة الفورية **Immediate Window**:

```
'Print the field names
For i = 0 To rsStudent.Fields.Count - 1
Debug.Write(rsStudent.Fields(i).Name &vbTab)
Next i
Debug.WriteLine(" ")

'Print values of all fields for each record
rsStudent.MoveFirst()
While Not rsStudent.EOF
For i = 0 To rsStudent.Fields.Count - 1
Debug.Write(rsStudent.Fields(i).Value.ToString&vbTab)
Next i
Debug.WriteLine(" ")
rsStudent.MoveNext()
End While
```

وكما ترى من الكود فإن أدلة الحقول تبدأ من 0 وتنتهي بأقل من عدد حقول مجموعة السجلات بمقدار ١ حيث يتم الوصول إلى عدد السجلات من خلال الخاصية **Count**.

### التنقل داخل مجموعة السجلات

بعد أن تقوم بإرجاع البيانات من مصدر البيانات إلى مجموعة السجلات **Recordset**، يمكنك الوصول إلى قيم الحقول الموجودة بالسجل الحالي وتعديلها كما رأينا في الأمثلة السابقة. ويمكنك تصور مجموعة السجلات كما لو كانت ملف طويل، وفي أي وقت يكون السجل الحالي عبارة عن مؤشر لمكان داخل هذا الملف. وكى تستطيع العمل مع السجلات

المختلفة الموجودة داخل مجموعة السجلات الناتجة، يمكنك استخدام وظائف الانتقال التالية لتغيير السجل الحالي:

- الوظيفة **MoveFirst()**: وتتسبب في الانتقال إلى السجل الأول بمجموعة السجلات بعد علامة بداية الملف (BOF) مباشرةً.
- الوظيفة **MoveLast()**: وتتسبب في الانتقال إلى السجل الأخير بمجموعة السجلات قبل علامة نهاية الملف (EOF) مباشرةً.
- الوظيفة **MoveNext()**: وتتسبب في الانتقال إلى السجل التالي بمجموعة السجلات ناحية علامة نهاية الملف (EOF).
- الوظيفة **MovePrevious()**: وتتسبب في الانتقال إلى السجل السابق بمجموعة السجلات ناحية علامة بداية الملف (BOF).
- الوظيفة **Move()**: وتتسبب في الانتقال إلى الأمام أو إلى الخلف عدداً معيناً من السجلات.

تعتبر كل من EOF و BOF عن الخصائص التي توضح نقاط بداية ونهاية مجموعة السجلات على الترتيب.



تعيين نوع مؤشر السجل الحالي

قمنا في الأمثلة السابقة باستخدام وظيفتين من وظائف التنقل بين السجلات وهما الوظيفة **MoveFirst()** والوظيفة **MoveNext()** كما استخدمنا الدوارة **While** للتحرك إلى الأمام داخل مجموعة سجلات مفتوحة حتى نصل إلى نهاية هذه المجموعة ودلالة ذلك ظهور القيمة **True** داخل الخاصية **EOF**. ولكن إذا قمنا في الكود السابق باستخدام الوظيفة **MovePrevious()** أو **MoveLast()**، فسيقوم المترجم بالاعتراض على الفور وإظهار رسالة الخطأ المناسبة وذلك لعدم صحة نوع المحث **Cursor type** المستخدم والذي يشير بدوره إلى المكان الحالي كما في نافذة محث الأوامر (Dos) تماماً. ولأننا لم نقم بتعيين قيمة الخاصية **Recordset.CursorType**، يتم استخدام قيمتها الافتراضية

adOpenForwardOnly والتي لا تدعم استخدام الوظيفة MovePrevious() أو MoveLast().

سنقوم فيما يلي بتوضيح استخدام وظائف التنقل بين السجلات من خلال مثال عملي بسيط. تابع معنا الخطوات الآتية:

١. قم بإضافة نموذج جديد إلى التطبيق الحالي وليكن باسم Navigation.vb مع جعله نموذج بدء التطبيق.
٢. قم بإضافة مربع سرد من مربع الأدوات إلى النموذج.
٣. قم بإضافة ثلاثة أزرار أمر من مربع الأدوات إلى أسفل النموذج.
٤. قم بإضافة مربع نص من مربع الأدوات إلى النموذج أقصى يمين الأزرار الثلاثة (انظر شكل ١٠-٥).
٥. قم بتغيير خصائص الأدوات المضافة بالاستعانة بجدول ١٠-٢ التالي.

جدول ١٠-٢ خصائص الأدوات المضافة للنموذج

الأداة	العنوان	الاسم
مربع السرد	-	IstData
الزر الأيسر	<<Prev	btnPrevious
الزر الأوسط	Next >>	btnNext
الزر الأيمن	Jump by #	BtnJump
مربع النص	-	TxtJump

٦. قم بإضافة الكود التالي لإجراءات أحداث نقر الأزرار الثلاثة:

```
Private rs As ADODB.Recordset
```

```
Public Sub New()
```

```
MyBase.New()
```

```
'This call is required by the Windows Form Designer.
```

```
InitializeComponent()
```

```
'Add any initialization after the InitializeComponent() call
```

```
rs = New ADODB.Recordset()
```

```
rs.CursorType = ADODB.CursorTypeEnum.adOpenStatic
rs.Open("Select * from Students", "Provider= Microsoft.Jet.
OLEDB.4.0; Data Source = d:\Students.mdb")
DisplayCurrentRecord()
End Sub
```

```
Private Sub DisplayCurrentRecord()
    Dim i As Integer
    Dim s As String
    If rs.BOF Then rs.MoveFirst()
    If rs.EOF Then rs.MoveLast()
    lstData.Items.Clear()
    For i = 0 To rs.Fields.Count - 1
        s = rs.Fields(i).Name & ": " & rs.Fields(i).Value.ToString
    lstData.Items.Add(s)
    Next i
    Me.Text = "Current Position:" & rs.AbsolutePosition
End Sub
```

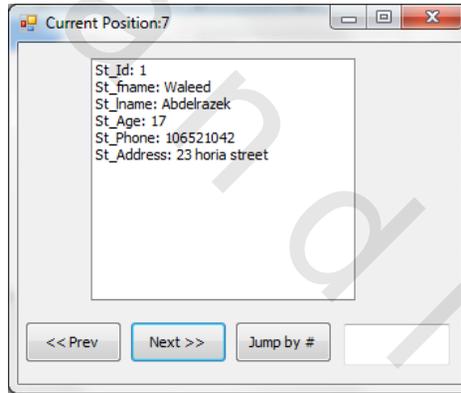
```
Private Sub btnNext_Click(sender As Object, e As EventArgs)
    Handles btnNext.Click
    rs.MoveNext()
    DisplayCurrentRecord()
End Sub
```

```
Private Sub btnPrevious_Click(sender As Object, e As
EventArgs) Handles btnPrevious.Click
    rs.MovePrevious()
    DisplayCurrentRecord()
End Sub
```

```
Private Sub btnJump_Click(sender As Object, e As EventArgs)
    Handles btnJump.Click
    rs.Move(convert.ToInt32(txtJump().Text))
    DisplayCurrentRecord()
End Sub
```

قمنا في هذا الكود بإنشاء مجموعة بيانات باستخدام محث ساكن Static Cursor وذلك بتخصيص القيمة adOpenStatic للخاصية CursorType والتي تسمح بالتنقل بين

السجلات في كلا الاتجاهين على عكس القيمة **adForwardOnly** التي استخدمناها من قبل وتسمح بالتقدم إلى الأمام فقط. وبعد ذلك قمنا بإنشاء الدالة المخصصة **(DisplayCurrentRecord)** التي يتم فيها اختبار الخاصيتين **BOF** و **EOF** أولاً قبل عرض أسماء وقيم حقول السجل الحالي. وهذه الخطوة ضرورية للغاية وذلك لأن محاولة الوصول إلى حقل غير موجود بالسجل الحالي يؤدي إلى حدوث خطأ في عملية الاتصال. فم بتشغيل التطبيق السابق ولا تنسى أن يكون النموذج الجديد هو نموذج البدء وذلك من مربع خصائص التطبيق، تحصل على عرض للسجل الحالي بحيث يمكنك التنقل بين السجلات للأمام أو الخلف أو حتى الذهاب مباشرة إلى أحد السجلات وذلك من خلال الأزرار الثلاثة ومربع النص الموجود بالنموذج (انظر شكل ١٠-٥).



شكل ١٠-٥ الانتقال بين السجلات في كلا الاتجاهين

### الحصول على عدد السجلات

من الخصائص الهامة المصاحبة للكائن **Recordset** الخاصية **RecordCount** المستخدمة لاسترجاع عدد سجلات المجموعة. فعلى الرغم من إمكانية استخدام خاصية **EOF** لحساب عدد السجلات من بداية المجموعة إلى نهايتها، إلا أن استخدام الخاصية **RecordCount** يوفر العديد من خطوات المعالجة كما في الكود التالي:

```
rsInvoices.Open("spGetInvoicestoProcess")
if rsInvoices.RecordCount = 0 Then
    MessageBox.Show("No Invoices to process")
```

```
Else
    PreparePrinter()
    ProcessInvoices(rsInvoices)
End If
```

يتم في هذا الكود استدعاء دالة مخصصة باسم `PreparePrinter()` فقط في حالة احتواء مجموعة السجلات `ReInvoices` على سجل واحد على الأقل. ومن أهم الاستخدامات الهامة للخاصية `RecordCount` إظهار شريط تقدم لمعرفة عدد السجلات المتبقى حتى يتم الانتهاء من معالجة جميع السجلات.

### فرز وترشيح مجموعة السجلات

يمكنك كما تعلم استخدام عبارة `Where` لترشيح مجموعة السجلات الناتجة من عبارة `SQL`، كما يمكنك استخدام عبارة `Order by` لفرز هذه السجلات وترتيبها ترتيباً معيناً تبعاً لحقل أو أكثر من الحقول الموجودة بقاعدة البيانات.

يحتوى الكائن `Recordset` على الخاصيتين `Filter` و `Sort` التى يمكنها أداء نفس المهام السابقة دون التغيير فى استعمال `SQL` المستخدم، حيث يمكنك استخدام الخاصية `Sort` لترتيب السجلات بتخصيصها بقيم الحقول المستخدمة فى الترتيب مفصولة بالعلامة "،" كما يلي:

```
rs.Sort = "State, City, LastName,FirstName"
re.Sort = "Age DESC, FirstName, LastName"
```

وكما فى عبارة `Order by`، يمكنك تعيين إذا ما كان الترتيب تنازلياً أم تصاعدياً من خلال استخدام كلمة `DESC` مع الترتيب التنازلى.

يمكنك أيضاً استخدام الخاصية `Filter` بنفس طريقة استخدام عبارة `Where` كما يلي:

```
rs.Filter = "Age > 15 AND LastName Like 'W%'"
rs.Filter = "State = "Cairo" OR State= "Alexandria"
```

عند تعيين أي من الخاصيتين `Filter` أو `Sort` يتم تحديث محتويات مجموعة السجلات تلقائياً كي تعكس التغييرات الجديدة.



## تغيير البيانات داخل مجموعة البيانات

بعد أن عرفت كيفية الحصول على البيانات من قاعدة البيانات وكيفية عرضها على الشاشة، من الضروري التعرف الآن على كيفية تغيير أو تحديث هذه البيانات. فإذا قمت بإعداد مجموعة السجلات بصورة صحيحة، يمكنك بسهولة شديدة تغيير البيانات التحتية الموجودة بقاعدة البيانات. كل ما تحتاج إليه هو الانتقال إلى السجل المطلوب ثم تخصيص قيمة جديدة لكل حقل من الحقول التي ترغب في تحديث بياناتها وأخيراً يتم استدعاء الوظيفة `Update()` كما يلي:

```
rs.Fields("FirstName").Value = "Waleed"  
rs.Update()
```

يمكنك كذلك إضافة سجلات جديدة إلى الكائن `Recordset` باتباع نفس الخطوات السابقة مع خطوة واحدة إضافية كما يلي:

- استدعاء الوظيفة `AddNew()`
- تخصيص القيم للحقول
- استدعاء الوظيفة `Update()`

يوضح الكود التالي كيفية إضافة سجل جديد إلى جدول `Students`:

```
Dim rs As ADODB.Recordset
```

فتح مجموعة السجلات `RECORDSET`

```
rs = New ADODB.Recordset()  
rs.CursorType = ADODB.CursorTypeEnum.adOpenStatic  
rs.LockType = ADODB.LockTypeEnum.adLockOptimistic  
rs.Open("Select * from Students", "Provider= Microsoft.Jet.  
OLEDB.4.0; Data Source = d:\Students.mdb")
```

إضافة سجل جديد

```
rs.AddNew()  
rs.Fields("St_ID").Value = 20  
rs.Fields("St_Fname").Value = Shreef  
rs.Fields("St_Iname").Value = Abdelwahed  
rs.Update()
```

```
rs.Close()
```

### MessageBox.Show("record added")

وكما هو الحال مع عبارة SQL INSERT، يجب تخصيص القيم للحقول التي لا تستقبل Null وإلا تظهر رسالة خطأ.

إذا لم تقم باستدعاء الوظيفة Update() بعد تعيين قيم حقول السجلات الجديدة، فلن يتم إضافة هذه السجلات إلى قاعدة البيانات وستذهب بياناتها أدراج الرياح بمجرد الانتقال إلى سجل آخر باستخدام أي من وظائف التنقل بين السجلات التي ذكرناها من قبل.



يمكنك أيضاً حذف السجل الحالي داخل مجموعة السجلات باستخدام الوظيفة Delete() كما يلي:

### rs.Delete()

وبمجرد استدعاء هذه الدالة لا يصبح هناك وجوداً للسجل الحالي، لذا يجب أن تعقب عملية الحذف هذه باستدعاء إحدى وظائف التنقل بين السجلات قبل محاولة الوصول إلى قيم الحقول.

### مفهوم تأمين السجلات

تكون مجموعات سجلات ADO افتراضياً للقراءة فقط، لذا فعند محاولة تخصيص قيمة أحد الحقول في هذه الحالة يتسبب في ظهور رسالة خطأ. فإذا أردت إضافة أو تغيير أو حذف السجلات، يجب أن تقوم أولاً بتعيين قيمة أخرى للخاصية LockType المستخدمة لتحديد نوع تأمين السجلات المستخدم كما في الكود التالي:

### re.LockType = ADODB.LockTypeEnum.adLockOptimistic

ولعلك تتساءل الآن عن مدى أهمية هذه الخاصية. والإجابة ببساطة شديدة أنك إذا قمت بتعديل السجلات في قاعدة بيانات متعددة المستخدمين، يجب أن تكون على دراية كافية بمفهوم تأمين السجلات Record Locking والذي يعنى منع المستخدمين الآخرين من محاولة تعديل نفس السجل الموجود في قاعدة البيانات في نفس الوقت. لذا يتم التحكم في

تأمين السجلات من خلال الخاصية **LockType** التي يمكن أن تحتوى على إحدى القيم التالية:

- القيمة **adLockReadOnly** وتقوم بتعيين بيانات مجموعة السجلات للقراءة فقط
- القيمة **adLockPessimistic** وتوفر التأمين التشاؤمي للسجلات وهو ما يعنى تأمين السجل أثناء تعديله.
- القيمة **adLockOptimistic** وتوفر التأمين النفاثلي للسجلات وهو ما يعنى تأمين السجلات عند استدعاء الوظيفة **Update** فقط.
- القيمة **adLockBatchOptimistic** وتوفر تحديث أكثر من سجل في نفس الوقت من خلال الوظيفة **(UpdateBatch)**.

#### مشاهدة تغييرات الآخرين

عند العمل مع قاعدة بيانات متعددة المستخدمين أى يتم الوصول إليها من خلال شبكة من الحاسبات، يجب التأكد من دقة البيانات الموجودة داخل مجموعة سجلاتك. فكما ذكرنا سابقاً أن الخاصية **CursorType** ربما تقيد عملية الانتقال بين السجلات، وعلى ذلك تعتمد النقطة الرئيسية في تحديد نوع المؤشر المستخدم على كيفية ارتباط مجموعة السجلات بالبيانات الأساسية الموجودة بقاعدة البيانات.

يوضح جدول ١٠-٣ التالى القيم المختلفة للخاصية **CursorType** ومدلول كل منها.

جدول ١٠-٣ قيم الخاصية **CursorType**

المدلول	القيمة
تستخدم لتسريع استرجاع البيانات ولكن تسمح بالتحرك إلى الأمام فقط داخل مجموعة السجلات	<b>adOpenForwardOnly</b>
تتيح لبرنامجك مشاهدة بعض تغييرات البيانات التي تتم من قبل بقية المستخدمين	<b>adOpenKeySet</b>

تتيح لبرنامجك مشاهدة جميع تغييرات البيانات التي تتم من قِبَل بقية المستخدمين	adOpenDynamic
لا يمكن من خلال هذه القيمة مشاهدة أى تغييرات تتم من قِبَل بقية المستخدمين	adOpenStatic

يعتمد نوع القيم المتاحة لمؤشر السجلات على نوع قاعدة البيانات المستخدمة، حيث لا تدعم جميع هذه القيم من قِبَل جميع قواعد البيانات. ففي حالة قاعدة بيانات Access على سبيل المثال، تكون القيمة الافتراضية للمؤشر هي adOpenForwardOnly والتي تستخدم للقراءة السريعة من قاعدة البيانات، بينما يفضل استخدام المؤشر AdOpenKeySet في حالة عمليات التحديث والعمليات المركبة الأخرى.

من خصائص المؤشر الهامة المستخدمة عند التعامل مع مجموعة السجلات، الخاصية CursorLocation والتي تحدد إذا ما كانت مجموعة السجلات مرتبطة بمجموعة البيانات أم منفصلة عنها كما سنعرف بعد قليل.



## استخدام الكائن Command

عند العمل مع مجموعة من البيانات، فإنك تقضى معظم الوقت في التعامل مع خصائص ووظائف الكائن Recordset. لكن إذا ما أردت استرجاع هذه البيانات، فستجد أنه لا غنى عن استخدام الكائن ADODB.Command الذى يتيح لك تضمين استعلام أو إجراء SQL مخزن داخل كائن قابل للاستخدام أكثر من مرة وبالتالي يكون هو الحل المثالى إذا أردت تنفيذ عملية من العمليات مرات عديدة. ويتم تخزين معاملات الاستعلام أو الإجراء المخزن داخل التجمع Parameters الخاص بالكائن. وبعد أن تنتهى من إعداد هذا الكائن، يمكنك تغيير معاملاته واستدعاؤه أكثر من مرة.

لتوضيح كيفية استخدام الكائن Command، انظر معى إلى عبارة SQL التالية التى تقوم باسترجاع جميع الطلبة الذين يبدأ اسمهم الأول بالحرف "W":

```
Select * ROM Students WHERE St_fname = 'W%'
```

سنقوم بتحويل هذه العبارة إلى إجراء مخزن كى يقوم المستخدم بتحديد الحرف الأول بنفسه وذلك كما يلي:

```
CREATE PROCEDURE spSearch
@strSearchLetterchar(1)
AS
SELECT *
FROM Students
WHERE St_fname Like @strSearchLetter + '%'
```

بعد ذلك يمكنك إنشاء كائن Command وإخباره بالإجراء المخزن وبيانات معاملة كما في الكود التالي:

```
Dim rs As ADODB.Recordset
Dim cmd As New ADODB.Command()
Dim prm As ADODB._Parameter
Dim con As New ADODB.Connection()
```

'فتح الاتصال بقاعدة البيانات'

```
con.Open("Provider= Microsoft.Jet.
OLEDB.4.0; Data Source = d:\Students.mdb")
```

'تعيين الكائن command'

```
cmd.CommandType =
ADODB.CommandTypeEnum.adCmdStoredProc
cmd.CommandText = "spSearch"
cmd.ActiveConnection = con
```

'تعيين الكائن parameter'

```
prm = cmd.CreateParameter("@strSearchLetter",
_ADODB.DataTypeEnum.adChar, _
ADODB.ParameterDirectionEnum.adParamInput,1, "W")
cmd.Parameters.Append(prm)
```

'تعيين الكائن recordset'

```
rs = New ADODB.Recordset()
rs.CursorType = ADODB.CursorTypeEnum.adOpenStatic
rs.LockType = ADODB.LockTypeEnum.adLockOptimistic
```

تنفيذ الأمر'

rs.Open(cmd)

وكما ترى فإن هذا الكود يحتوى على جميع كائنات ADO التي شرحناها في هذا الفصل حتى الآن. كما يحتوى الكائن Command على مجموعة من الكائنات Parameter التي يمثل كل منها معامل داخل الإجراء المخزن. ويمكنك بعد إنشاء الكائن Command احتواء هذه المعاملات واستدعائها مرةً أخرى من خلال عدد قليل من أسطر الكود كما يلي:

```
cmd.Parameters("@strSearchLetter").Value = "T"
```

```
rs.Close()
```

```
rs.Open(cmd)
```

وفي هذا الكود يتم إعادة فتح مجموعة السجلات مرةً أخرى بعد تغيير معاملات الأمر أو الكائن Command حيث يتم استرجاع جميع الأسماء التي تبدأ بالحرف T.

يحتوى الكائن Command أيضاً على الوظيفة Execute() التي تعمل بطريقة



مشابهة لمثيلتها في الكائن Connection.

## مجموعات السجلات المنفصلة

إذا سبق لك العمل مع تقنيات الوصول إلى البيانات بالإصدارات السابقة من Visual Basic، تجد أن ADO يحتوى على العديد من السمات التي تتيح لك العمل مع البيانات بطريقة لم تكن تتخيلها من قبل. فعلى سبيل المثال، في معظم التطبيقات القديمة، يتم دائماً ربط مجموعة السجلات بقاعدة البيانات المصاحبة طالما كانت مجموعة السجلات محملة داخل الذاكرة. ولكن نظراً لكثرة استخدام الإنترنت وكثرة انقطاع عملية الاتصال بين البرنامج وقاعدة البيانات، تم استحداث مجموعة السجلات المنفصلة أى التي تعمل داخل الذاكرة بمعزل عن قاعدة البيانات المصاحبة، حيث يمكنك تحديث مجموعة البيانات وحفظها على وحدة تخزين إن أحببت وكذلك تحقيق التزامن بينها وبين قاعدة البيانات فيما بعد. وتظهر أهمية السجلات المنفصلة Disconnected Recordset عند العمل مع قواعد

البيانات الكبيرة التي يستخدمها ما يقرب من مائة ألف مستخدم يحاولون عرض قوائم المنتجات في نفس الوقت وبالتالي تقلل هذه التقنية من وقت الاتصال.

### فصل مجموعة السجلات عن قاعدة البيانات

يمكنك فصل مجموعة السجلات Recordset عن قاعدة البيانات عن طريق تخصيص القيمة adUseClient للخاصية CursorLocation التي تحدد مكان المؤشر إذا كان في قاعدة البيانات (ناحية الخادم) أم في الذاكرة (ناحية العميل). وعامةً يمكنك إنشاء مجموعة السجلات المنفصلة باتباع الخطوات الآتية:

١. قم بإنشاء كائن Recordset جديد.
  ٢. قم بتخصيص القيمة adUseClient للخاصية CursorLocation.
  ٣. قم بملء الكائن Recordset بالبيانات من خلال الاتصال بقاعدة البيانات باستخدام الوظيفة Open().
  ٤. قم بتخصيص القيمة Nothing للخاصية ActiveConnection حتى يتم تعطيل عملية الاتصال مع قاعدة البيانات.
- وذلك كما في الكود التالي:

```
Dim con As New ADODB.Connection()
rs = New ADODB.Recordset()
rs.CursorType = ADODB.CursorTypeEnum.adOpenStatic
rs.CursorLocation = ADODB.CursorLocationEnum.adUseClient
rs.Open("Select * from Students", "Provider=
Microsoft.Jet.OLEDB.4.0; Data Source =
e:\books\Students.mdb")
rs.ActiveConnection = Nothing
```

### إنشاء مجموعة السجلات من خلال الكود

على الرغم من سهولة ملء مجموعة السجلات من قاعدة بيانات موجودة بالفعل بسهولة تامة، إلا أنك تستطيع إنشاء مجموعة سجلات من البداية باستخدام الكود عن طريق إنشاء كائن Recordset جديد ثم إنشاء مجموعة الحقول Fields وأخيراً إضافة السجلات

باستخدام نفس الوظائف التي ذكرناها من قبل. فمثلاً في الكود التالي يتم إنشاء مجموعة سجلات تحتوى على حقلين أحدهما للاسم والآخر لرقم التليفون كما يلي:

```
Dim rs As New ADODB.Recordset()
```

إنشاء التركيب'

```
rs.Fields.Append("Name", ADODB.DataTypeEnum.adVarChar,  
30)  
rs.Fields.Append("Phone", ADODB.DataTypeEnum.adVarChar,  
18, ADODB.FieldAttributeEnum.adFldIsNullable)
```

إضافة سجل جديد'

```
rs.Open()  
rs.AddNew()  
rs.Fields("Name").Value = "WaleedAbdelrazek"  
rs.Fields("Phone").Value = "01006521042"  
rs.Update()
```

والجديد في هذا الكود هو استخدام الوظيفة **Append()** التي تقوم بإضافة حقل إلى مجموعة البيانات وتحتوى على أكثر من معامل مثل اسم الحقل ونوع بياناته وطول هذا الحقل وكذلك بعض صفاته مثل عدم استقبال القيمة **Null** كما في حالة الحقل **Phone**. يمكنك كذلك حفظ مجموعة السجلات داخل ملف بتنسيق **XML** على القرص الصلب باستخدام الكود التالي:

```
rs.Save("c:\temp.xml",  
ADODB.PersistFormatEnum.adPersistXML)  
كما يمكنك في أى وقت تحميل هذا الملف مرةً أخرى باستخدام الكود التالي:  
rs.Open("c:\temp.xml")
```

### تحديث قاعدة البيانات

تعرفنا قبل ذلك على كيفية تأمين السجلات لمنع مجموعة المستخدمين من التعامل مع نفس الحقل في نفس الوقت. لكن على الرغم من ذلك، يأخذ مفهوم تأمين السجلات معنىً آخر عند العمل مع مجموعة السجلات المنفصلة. فإذا لم تكن متصلاً بقاعدة البيانات، فأنت

بذلك تقوم بالنظر إلى السجلات كما لو كنت تنظر إلى مجموعة من الكتب داخل المكتبة. فإذا قمت بتحديث أو تعديل بيانات مجموعة السجلات، فلن تنعكس هذه التغييرات على قاعدة البيانات حتى تقوم بتسجيل هذه السجلات. ومن ثمَّ يمكنك تحديث قاعدة البيانات بطريقتين، الأولى بإرجاء الأمر إلى ADO كى يقوم بعملية التحديث المجمعّة Batch Update، أما الطريقة الثانية فتم يدوياً من خلال الكود بمعالجة كل سجل داخل مجموعة السجلات المنفصلة باستدعاء إجراء مخزّن لتحديث سجل قاعدة البيانات المصاحب. وفيما يلي نقوم بإلقاء نظرة خاطفة على كلٍ من الطريقتين.

#### تحديث مجموعة من السجلات

يتيح لك التحديث المجمع للسجلات إجراء العديد من التغييرات على مجموعة السجلات وتطبيقها على قاعدة البيانات المصاحبة مرةً واحدة من خلال الوظيفة UpdateBatch(). ولكى تتمكن من استخدام هذه الوظيفة، يجب تخصيص القيمة adLockBatchOptimistic للخاصية LockType الخاصة بمجموعة السجلات كما يلي:

```
rs.LockType = adLockBatchOptimistic
```

وعلى ذلك تشتمل رحلة التحديث على الخطوات الآتية:

١. إجراء التغييرات على مجموعة السجلات المنفصلة كما فى الكود التالى:

```
rs.Fields("Phone").Value = "22405330"
```

```
rs.Update()
```

```
rs.MoveNext()
```

```
rs.Delete()
```

٢. إعادة تنشيط عملية الاتصال مع قاعدة البيانات بتخصيص كائن الاتصال المفتوح

للخاصية ActiveConnection كما يلي:

```
rs.ActiveConnection = con
```

٣. تحديث قاعدة البيانات من خلال الوظيفة UpdateBatch() كما يلي:

```
rs.MarshalOptions = adMarshalModifiedOnly
```

```
rs.UpdateBatch()
```

وقد استخدمنا الخاصية **MarchalOptions** التي تحدد إذا ما كنت تريد إرجاع مجموعة السجلات بالكامل أم السجلات التي تم تعديلها فقط. وبذلك تتم عملية التحديث بتغيير قيمة الحقل **Phone** في السجل الأول ثم حذف السجل التالي.

### تحديث السجلات يدوياً

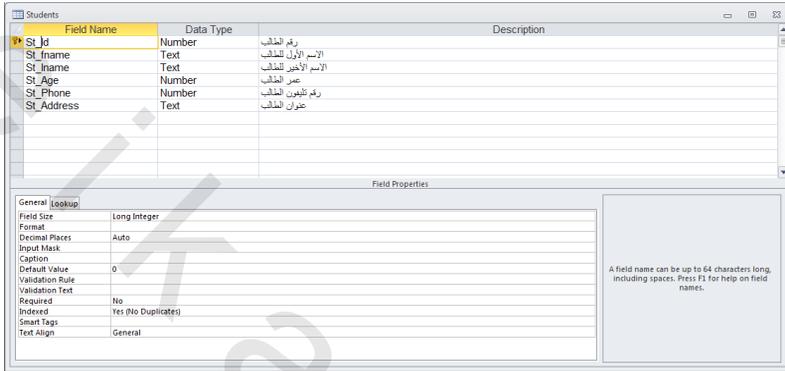
ربما أردت في بعض الأحيان مزيداً من التحكم في عملية التحديث، وفي هذه الحالة يمكنك استخدام الخاصية **Status** المصاحبة للكائن **Recordset** لتحديد نوع التحديث الذي يتم على سجل معين، وذلك كما في الكود التالي:

```
While Not rs.EOF
    Select Case rs.Status
        Case ADODB.RecordStatusEnum.adRecNew
            هذا سجل جديد. قم بتنفيذ العبارة 'INSERT
        Case ADODB.RecordStatusEnum.adRecModified
            حدث تغيير للسجل. قم بتنفيذ العبارة 'UPDATE
    End Select
rs.MoveNext
End While
```

### مثال تطبيقي

سنقوم فيما يلي بإنشاء مثال تطبيقي على استخدام ADO حيث نتعرف على كيفية إظهار مجموعة السجلات وتعديلها من خلال أداة شبكة البيانات **DataGridView** الموجودة داخل **Visual Studio 2012** والتي تدعم استخدام **ADO.NET** التي سنتحدث عنها في الفصل القادم إن شاء الله. ولكن لحسن الحظ أن كلاً من **ADO** و **ADO.NET** يستخدم لغة **XML**. لذا سنقوم بنقل البيانات من كائن مجموعة السجلات **Recordset** إلى كائن مجموعة البيانات **DataSet** من خلال ملف ذى تنسيق **XML**. تابع معنا الخطوات الآتية:

١. قم بإضافة بعض الحقول إلى الجدول **Students** داخل قاعدة البيانات التي تحتوي على نفس الاسم ولتكن الحقول **St\_Age** و **St\_Phone** و **St\_Address** (انظر شكل ١٠-٦).



- شكل ١٠-٦ الجدول الذي سنقوم باستخدامه في التطبيق
٢. افتح بيئة تطوير **Visual Studio 2012** إذا لم تكن مفتوحة بالفعل ثم قم بإنشاء تطبيق نوافذ جديد باسم مناسب وليكن **UsingADO**.
٣. من نافذة **مستكشف الحل**، انقر اسم المشروع بزر الفأرة الأيمن ثم اختر **Add Reference** من القائمة الموضعية وقم بإضافة المرجع **Microsoft ActiveX Data Objects 6.0 library** إلى مشروعك كما تعلمت من قبل (راجع شكل ١٠-١).
٤. قم بإضافة ثلاثة أزرار أمر من مربع الأدوات إلى أعلى النموذج مع تغيير أسمائهم وعناوينهم كما في جدول ١٠-٤.
٥. قم بإضافة الأداة **DataGridView** من مربع الأدوات إلى النموذج مع تغيير الخاصية **Name** إلى **grdList**.
٦. قم بإنشاء الكود التالي للأزرار الثلاثة:
- جدول ١٠-٤ خصائص أزرار الأمر الموجودة بالنموذج

الاسم	العنوان	الزر
-------	---------	------

btnLoadRecordset	تحميل مجموعة السجلات	الأيمن
btnWriteDataset	كتابة مجموعة البيانات	الأوسط
btnLoadDataset	تحميل مجموعة البيانات	الأيسر

```

Private dsStudent As DataSet
Private Sub btnLoadRecordset_Click(sender As Object, e As
EventArgs) Handles btnLoadRecordset.Click
    Dim rs As ADODB.Recordset
    rs = New ADODB.Recordset()
    rs.CursorType = ADODB.CursorTypeEnum.adOpenStatic
    rs.Open("Select * from Students", "Provider=
Microsoft.Jet.OLEDB.4.0; Data Source = d:\Students.mdb")
    If System.IO.File.Exists("d:\temp.rs.xml") Then
        System.IO.File.Delete("d:\temp.rs.xml")
    End If
    rs.Save("d:\temp.rs.xml",
ADODB.PersistFormatEnum.adPersistXML)
    dsStudent = New DataSet()
    dsStudent.ReadXml("d:\temp.rs.xml",
XmlReadMode.InferSchema)
    grdList.DataSource = dsStudent
    grdList.DataMember = "row"
    MessageBox.Show("The grid has been loaded from the
database using ADO.")
End Sub
Private Sub btnWriteDataset_Click(sender As Object, e As
EventArgs) Handles btnWriteDataset.Click
    If System.IO.File.Exists("d:\temp.ds.xml") Then
        System.IO.File.Delete("d:\temp.ds.xml")
    End If
    dsStudent.WriteXml("d:\temp.ds.xml")
    MessageBox.Show("The contents of the dataset bound to
the grid have been written to c:\temp.ds.xml")

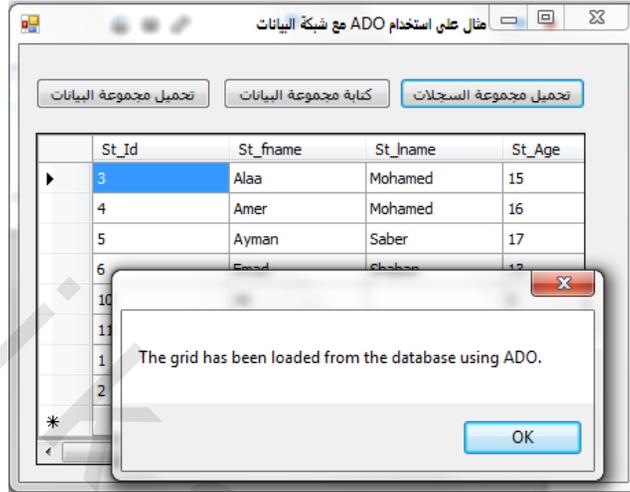
End Sub

```

```
Private Sub btnLoadDataset_Click(sender As Object, e As
EventArgs) Handles btnLoadDataset.Click
    If System.IO.File.Exists("d:\tempds.xml") Then
        dsStudent = New DataSet()
        dsStudent.ReadXml("d:\tempds.xml",
        XmlReadMode.InferSchema)
        grdList.DataSource = dsStudent
        grdList().DataMember = "row"
        MessageBox.Show("The grid has been loaded from the
        XML file c:\tempds.xml.")
    Else
        MessageBox.Show("انقر زر كتابة مجموعة البيانات أولاً")
    End If
End Sub
```

وعن هذا الكود، نوضح ما يلي:

- يتم في الإجراء `btnLoadRecordset()` تحميل مجموعة السجلات من جدول `Students` ثم تخزينها داخل ملف على القرص الصلب باسم `tempds.xml`. وبعد ذلك يتم إنشاء مجموعة بيانات `DataSet` وتحميلها ببيانات هذا الملف ثم تخصيص محتويات المجموعة إلى أداة شبكة البيانات الموجودة بالنموذج وأخيراً إظهار مربع رسالة بتحميل الشبكة من قاعدة البيانات.
- يتم في الإجراء `btnWriteDataset()` كتابة بيانات مجموعة البيانات إلى الملف `tempds.xml` مع إظهار رسالة تفيد ذلك.
- يتم في الإجراء `btnLoadDataset()` ملء مجموعة البيانات من الملف `tempds.xml` ثم عرض هذه البيانات داخل شبكة البيانات. أما في حالة عدم وجود هذا الملف، فيتم إظهار رسالة للمستخدم تحثه على نقر زر "كتابة مجموعة البيانات" أولاً.
- ٧. قم بتشغيل التطبيق ثم انقر زر "تحميل مجموعة السجلات"، تظهر محتويات الجدول `Students` داخل شبكة البيانات (انظر شكل ١٠-٧).



شكل ٧-١٠ استخدام شبكة البيانات مع مجموعة البيانات

٨. قم بتغيير البيانات الموجودة من خلال شبكة البيانات.
٩. انقر زر "كتابة مجموعة البيانات"، يتم كتابة محتويات كائن مجموعة البيانات DataSet إلى ملف XML.
١٠. قم بإغلاق التطبيق وتشغيله مرة أخرى ثم انقر زر "تحميل مجموعة البيانات"، تلاحظ تحميل البيانات وبها التغييرات التي قمت بإجرائها.

سنقوم في الفصول القادمة بالتعرف على كائن مجموعة البيانات DataSet المستخدم مع ADO.NET.

