الفصل الثانى عشر
استخدام كائنات ADO

*إذا كنت ممن استخدموا أحد إصدارات **Visual Basic** التى، فلاشك أنك تعلم الكثير عن كائنات **ADO** المستخدمة فى وصل تطبيقاتك بقواعد البيانات المختلفة. سنقوم فى 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.*

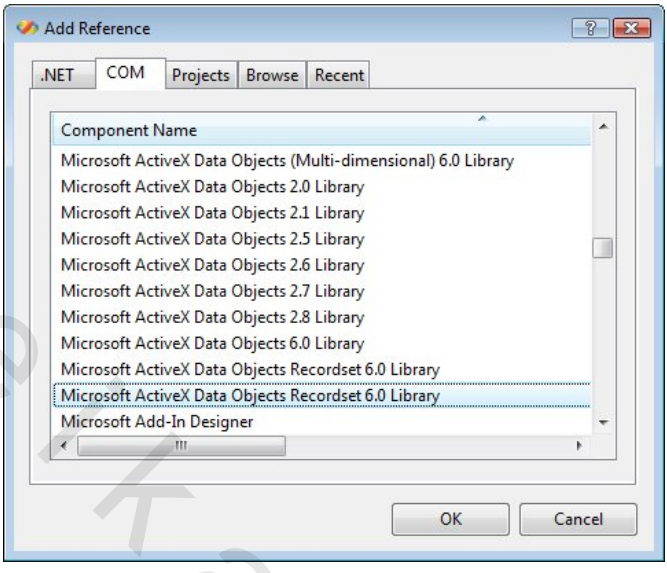
تعرفنا فى الفصل السابق على أساسيات الوصول إلى قواعد البيانات وإنشاء عبارات SQL. وهذا الفصل هو أول الفصول التى سنقوم فيها بالتعرف على كائنات بيانات ActiveX أو ActiveX Data Objects (ADO) وهى الوسيلة التى يتم من خلالها الوصول إلى جميع أنواع قواعد البيانات من خلال كود Visual Basic. وفى الإصدار قبل السابق من Visual Basic (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 شىء الآن بطعم .NET.. ولكن على الرغم من ذلك، فإن ADO هى التقنية الأساسية التى تم منها اشتقاق ADO.NET ومازال هناك العديد من البرامج التى تستخدم هذه التقنية، وهذا هو السبب الأول الذى جعلنا نتحدث عنها رغم استبدالها. أما السبب الثانى فلأن العديد من المبادئ المستخدمة للوصول إلى قاعدة البيانات واستخدامها من خلال ADO هى نفسها المستخدمة فى 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 2008، وهو ما سنتعرف عليه فى 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 أى كائن خارجى، يجب أن تقوم بإضافة مرجع إلى ADO قبل أن تقوم باستخدامه (انظر شكل ١٢-١). وبمجرد إضافتك لهذا المرجع، يمكنك استخدام الكائنات التى يحتوى عليها نموذج ADO داخل تطبيقك كما لو كانت جزءاً منه.

يوضح جدول ١٢-١ التالى الكائنات المختلفة الموجودة داخل نموذج ADO.



شكل ١٢-١ يمكنك تضمين كائنات ADO داخل Visual Studio 2008 من التبويب COM بالمربع الحوارى Add Reference

جدول ١٢-١ 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 مخزَّن
Property	يتيح لك الوصول إلى خصائص كائن ADO

يحتوى المرجع Microsoft ActiveX Data Objects 6.x Library على جميع الكائنات الموجودة بالجدول السابق، بينما يحتوى المرجع Microsoft ActiveX Data Objects Recordset 6.x Library على الكائن Recordset فقط والذى سنتعرف عليه بالتفصيل فى هذا الفصل.

عند إنشاء الكائنات الموجودة بالجدول١٢-١ السابق، يمكنك استخدام البادئة ADODB كما فى الكود التالى:

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

سنقوم فى الأجزاء التالية بالتعرف على بعض خصائص ووظائف وأحداث كائنات ADO. يمكنك التعرف على الوثيقة الكاملة من ADO وكذلك التحديثات والأمثلة المختلفة من العنوان التالى على شبكة الإنترنت:
www.microsoft.com/data/ado

استخدام الكائن Connection

كى تستطيع التعامل مع البيانات الموجودة داخل قاعدة البيانات، يجب أن تقوم أولاً بإنشاء الاتصال مع قاعدة البيانات وذلك من خلال الكائن Connection الذى يساعدك فى الاتصال بقاعدة البيانات أو الانفصال عنها. والخطوة الأولى فى إنشاء الاتصال هى إنشاء حالة جديدة من هذا الكائن كما فى الكود التالى:

```
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 فى الكود التالى:

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

```
con.Open()
```

فى 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 مباشرةً إلى الوظيفة **()Open** كما يلى:

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

تحتوى بعض كائنات **ADO** الأخرى مثل الكائن **Command** على الخاصية **Connection** التى تقبل إما سلسلة بيانات الاتصال أو الكائن **Connection**.

باتباع أيٍ 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 يلى:

```
con.Close()
```

انتبه أثناء تنفيذ الأمثلة الواردة فى 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) تحتوى على العديد من المعلومات والتى تشتمل على معرف المستخدم **User ID** وكلمة المرور **Password** واسم قاعدة البيانات **Database Name** حيث يتم كتابة هذه المعاملات فى صورة أزواج من الأسماء والقيم مفصولةً بعلامة الفاصلة المنقوطة ";". وعند إنشاء معاملات الاتصال تحتاج إلى تعيين المعلومات التالية:

- نوع قاعدة البيانات التى ترغب فى 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) والتى تشتمل على قيم اسم المستخدم User ID (أو UID) وكلمة المرور Password (أو PWD).
- موقع قاعدة البيانات والذى قد يشتمل على قيمة للخادم Server وقيمة لقاعدة البيانات Database وذلك فى حالة SQL Server أو اسم قاعدة البيانات فقط فى حالة قاعدة بيانات Microsoft Access.

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

يوضح الكود التالى 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 يلى:

- فى المثال الأول تم تعيين معاملات الاتصال بقاعدة بيانات Access باستخدام مزود OLEDB.
- فى المثال الثانى تم تعيين معاملات الاتصال بقاعدة بيانات Access باستخدام مشغل OLEDB.

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

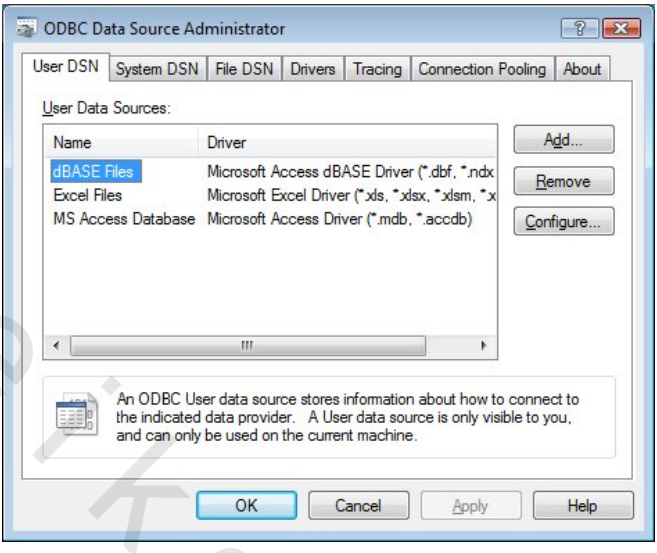
أحد الخيارات الهامة عند إنشائك لمعامل الاتصال هو استخدام مشغل ODBC أو مزود OLE DB وذلك بتعيين قيمة للمعامل Driver أو المعامل Provider وليس لكلاهما معاً، حيث تعتبر ODBC اختصاراً للعبارة Open Database Connectivity وهى تقنية قياسية لمشغلات قواعد البيانات الموجودة منذ سنوات. أما OLE DB فهى أحدث من سابقتها، 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 فى الكود التالى:

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

وكما ترى فإن سلسلة الاتصال فى هذا الكود بسيطة للغاية. فهى تحتوى فقط على DSN واسم المستخدم وكلمة المرور حيث تم تخزين بقية البيانات على مصدر بيانات ODBC المعرف على الحاسب. لإعداد مصدر بيانات ODBC على حاسبك، افتح نافذة لوحة التحكم Control Panel ثم انقر المجموعة Administrative Tools نقراً مزدوجاً ثم انقر رمز Data Sources من النافذة الناتجة نقراً مزدوجاً، تظهر نافذة ODBC Data Source Administrator التى يمكنك استخدامها فى إدخال بيانات الاتصال حيث يتم تخصيص System DSN لجميع المستخدمين بينما يختص User DSN بالمستخدم الحالى فقط (انظر شكل ١٢-٢).



شكل ١٢-٢ 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()

يحتوى الكائن Connection على الوظيفة ()Execute التى تستخدم فى 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 التى يمكنك تنفيذها عن طريق أى من الكائنات Recordset وCommand وConnection حيث يحتوى كل منهم على وظيفة لاسترجاع البيانات من قاعدة البيانات.

لتوضيح كيفية استخدام الوظيفة ()Execute فى 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 هى St_ID وSt_fname وSt_lname (أو انسخ قاعدة البيانات من على القرص المدمج المرفق بالكتاب) ثم تابع معنا الخطوات الآتية:

١. افتح بيئة تطوير Visual Studio 2008 ثم قم بإنشاء تطبيق نوافذى جديد باسم مناسب وليكن Execute.

٢. قم بإضافة مرجع لكائنات ADO كما أوضحنا من قبل (راجع شكل ١٢-١).

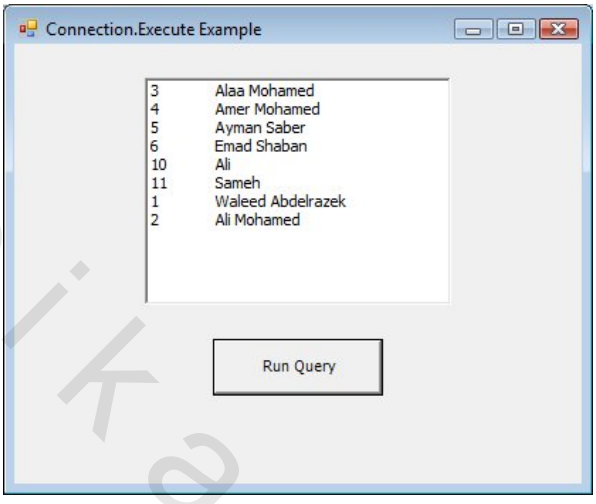
٣. قم بإضافة مربع سرد من مربع الأدوات إلى النموذج وأعد تسميته إلى lstStudents.

٤. قم بإضافة زر أمر من مربع الأدوات إلى النموذج وأعد تسميته إلى btnLoadList وعنوانه إلى Run Query.

٥. قم بإضافة الكود التالى إلى إجراء حدث نقر الزر btnLoadList:

```
Private Sub btnLoadList_Click(ByVal sender As System.Object,
ByVal e As System.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 Not rs.EOF
      sSSN = rs.Fields("St_id").Value.ToString
      sName = rs.Fields("St_fname").Value.ToString.Trim & " "
                        & rs.Fields("St_lname").Value.ToString
      lstStudents.Items.Add(sSSN & vbTab & sName)
      rs.MoveNext()
    End While
    rs.Close()
    con.Close()
  End Sub
```

٦. قم بتشغيل التطبيق ثم انقر الزر الموجود بالنموذج، تلاحظ امتلاء مربع السرد بالبيانات الموجودة بالجدول **Students** (انظر شكل ١٢-٣).



شكل ١٢-٣ نتيجة الاستعلام المنفذ من خلال الوظيفة **Execute**

تم فى الكود السابق استدعاء الوظيفة **()Execute** التى تنتمى للكائن **Connection** حيث قامت بدورها بإرجاع مجموعة سجلات **Recordset** التى 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** التى لا تقوم بإرجاع مجموعة سجلات كحذف السجلات أو إضافتها كما فى الكود التالى الذى يقوم بحذف أحد السجلات:

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

العمليات الأساسية لمجموعة السجلات

تعرفنا فى المثال السابق على كيفية الحصول على البيانات من قاعدة البيانات ووضعها داخل كائن مجموعة السجلات **RecordSet** التى تحتوى 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 يحتوى على أسماء الحقول وقيمها المختلفة، حيث تشكل كل مجموعة من الحقول سجل واحد، كما تتجمع السجلات كلها لتكوين مجموعة السجلات الناتجة.

سنقوم فيما يلى 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 يحتوى على مجموعة من الوظائف والخصائص المستخدمة فى استرجاع البيانات. وكما هو الحال مع جميع الكائنات، يجب أن تقوم أولاً بإنشاء حالة جديدة من الكائن **Recordset** قبل أن تقوم باستخدامه كما فى الكود التالى:

```
Dim rsStudent As ADODB.Recorset
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 يلى:

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

حيث:

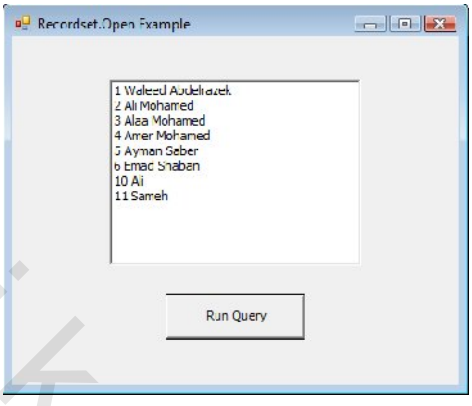
- يفترض السطر الأول من الكود أن **con** تمثل اتصال مفتوح يشير إلى مصدر البيانات كما أوضحنا من قبل.

- فى السطر الثانى تم استخدام سلسلة الاتصال وفى هذه الحالة يتم إنشاء كائن الاتصال ضمنياً.

يوضح الكود التالى 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 الحالى بنفس مواصفات النموذج السابق وليكن باسم **Recordset.vb** ثم قم بإدخال الكود التالى فى إجراء حدث نقر الزر الموجود بالنموذج:

```
Private Sub btnLoadList_Click(ByVal sender As System.Object,
ByVal e As System.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_lname from Students order
                                                by St_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 ١٢-٤).



شكل ١٢-٤ عرض محتويات قاعدة البيانات من خلال الوظيفة Recordset.Open()

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

إظهار محتويات الحقول

استخدمنا فى الكود السابق الأدلة فى الحصول على بيانات الحقول من مجموعة حقول الكائن Recordset حيث قامت عبارة SQL بإرجاع ثلاثة حقول هى 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 فى الكود السابق باستخدام أيٍ من الصيغتين التاليتين:

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

وهناك ملاحظة أخرى فى الكود السابق وهى إرجاع قيمة الحقل ككائن من النوع Field. ففى الإصدارات السابقة من Visual Basic كان يكفى إرجاع قيم الحقول حيث كانت

القيمة **Value** هى القيمة الافتراضية، أما **Visual Basic 2008** فيطلب تعريف الخاصية **Value** صراحةً. يستخدم الكود التالى دوارتين من النوع **For** لطباعة كل من أسماء الحقول وقيمها من داخل مجموعة السجلات على نافذة الخرج **Output**:

```
'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** وتنتهى 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 الحالى وتعديلها كما رأينا فى الأمثلة السابقة. ويمكنك تصور مجموعة السجلات كما لو كانت ملف طويل، وفى أى وقت يكون السجل الحالى 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 الحالى:

- الوظيفة **MoveFirst()**: وتتسبب فى الانتقال إلى السجل الأول بمجموعة السجلات بعد علامة بداية الملف **(BOF)** مباشرةً.

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

تعبر كل من BOF وEOF عن الخصائص التى توضح نقاط بداية ونهاية مجموعة السجلات على الترتيب.

تعيين نوع مؤشر السجل الحالى

قمنا فى 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 فى الكود السابق باستخدام الوظيفة MovePrevious() أو MoveLast()، فسيقوم المترجم بالاعتراض على الفور وإظهار رسالة الخطأ المناسبة وذلك لعدم صحة نوع المحث Cursor type المستخدم والذى يشير بدوره إلى المكان الحالى كما فى نافذة محث الأوامر(Dos) تماماً. ولأننا لم نقم بتعيين قيمة الخاصية Recordset.CursorType، يتم استخدام قيمتها الافتراضية adOpenForwardOnly والتى لا تدعم استخدام الوظيفة MovePrevious() أو MoveLast().

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

١. قم بإضافة نموذج جديد إلى التطبيق الحالى 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 ١٢-٥).
٥. قم بتغيير خصائص الأدوات المضافة بالاستعانة بجدول ١٢-٢ التالى.

جدول ١٢-٢ خصائص الأدوات المضافة للنموذج

الأداة	العنوان	الاسم
مربع السرد	———	lstData
الزر الأيسر	<< Prev	btnPrevious
الزر الأوسط	Next >>	btnNext
الزر الأيمن	Jump by #	btnJymp
مربع النص	———	txtJumb

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

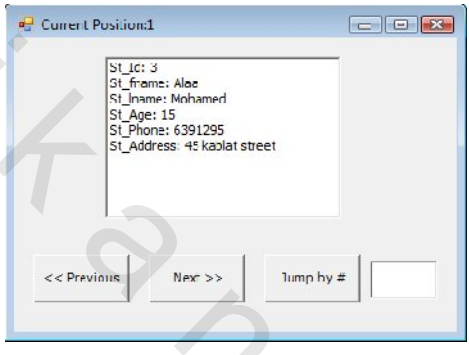
```
Private rs As ADODB.Recordset
Public Sub New()
    MyBase.New()
    'This call is required by the Windows Form Designer.
    InitializeComponent()

    '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(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnNext.Click
    rs.MoveNext()
    DisplayCurrentRecord()
End Sub

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

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

قمنا فى هذا الكود بإنشاء مجموعة بيانات باستخدام محث ساكن **Static Cursor** وذلك بتخصيص القيمة **adOpenStatic** للخاصية **CursorType** والتى تسمح بالتنقل بين السجلات فى كلا الاتجاهين على عكس القيمة **adForwardOnly** التى استخدمناها من قبل وتسمح بالتقدم إلى الأمام فقط. وبعد ذلك قمنا بإنشاء الدالة المخصصة **DisplayCurrentRecord()** التى يتم فيها اختبار الخاصيتين **BOF** و**EOF** أولاً قبل

عرض أسماء وقيم حقول السجل الحالى. وهذه الخطوة ضرورية للغاية وذلك لأن محاولة الوصول إلى حقل غير موجود بالسجل الحالى يؤدى إلى حدوث خطأ فى عملية الاتصال.

فم بتشغيل التطبيق السابق ولا تنسى أن يكون النموذج الجديد هو نموذج البدء وذلك من مربع خصائص التطبيق، تحصل على عرض للسجل الحالى بحيث يمكنك التنقل بين السجلات للأمام أو الخلف أو حتى الذهاب مباشرةً إلى أحد السجلات وذلك من خلال الأزرار الثلاثة ومربع النص الموجود بالنموذج (انظر شكل ١٢-٥).



شكل ١٢-٥ الانتقال بين السجلات فى 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 فى الكود التالى:

```
rsInvoices.Open("spGetInvoicestoProcess")
if rsInvoices.RecordCount = 0 Then
        MessageBox.Show("No Invoices to process")
Else
        PreparePrinter()
        ProcessInvoices(rsInvoices)
End If
```

يتم فى هذا الكود استدعاء الدالة PreparePrinter() فقط فى 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 وSort التى يمكنها أداء نفس المهام السابقة دون التغيير فى استعلام SQL المستخدم، حيث يمكنك استخدام الخاصية Sort لترتيب السجلات بتخصيصها بقيم الحقول المستخدمة فى الترتيب مفصولةً بالعلامة "," كما يلى:

```
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 يلى:

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

عند تعيين أى من الخاصيتين Filter أو Sort يتم تحديث محتويات مجموعة السجلات تلقائياً كى تعكس التغييرات الجديدة.

تغيير البيانات داخل مجموعة البيانات

بعد أن عرفت كيفية الحصول على البيانات من قاعدة البيانات وكيفية عرضها على الشاشة، من الضرورى 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 التى ترغب فى تحديث بياناتها وأخيراً يتم استدعاء الوظيفة **()Update** كما يلى:

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

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

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

يوضح الكود التالى 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_lname").Value = Abdelwahed
rs.Update()

rs.Close()
MessageBox.Show("record added")
```

وكما هو الحال مع عبارة **SQL INSERT**، يجب تخصيص القيم للحقول التى 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 أيٍ من وظائف التنقل بين السجلات التى ذكرناها من قبل.

يمكنك أيضاً حذف السجل الحالى داخل مجموعة السجلات باستخدام الوظيفة **()Delete** كما يلى:

```
rs.Delete()
```

وبمجرد استدعاء هذه الدالة لا يصبح هناك وجوداً للسجل الحالى، 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 فى هذه الحالة يتسبب فى 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 فى الكود التالى:

```
re.LockType = ADODB.LockTypeEnum.adLockOptimistic
```

ولعلك تتساءل الآن عن مدى أهمية هذه الخاصية. والإجابة ببساطة شديدة أنك إذا قمت بتعديل السجلات فى قاعدة بيانات متعددة المستخدمين، يجب أن تكون على دراية كافية بمفهوم تأمين السجلات **Record Locking** والذى يعنى منع المستخدمين الآخرين من محاولة تعديل نفس السجل الموجود فى قاعدة البيانات فى نفس الوقت. لذا يتم التحكم فى تأمين السجلات من خلال الخاصية **LockType** التى يمكن أن تحتوى على إحدى القيم التالية:

- القيمة **adLockReadOnly** وتقوم بتعيين بيانات مجموعة السجلات للقراءة فقط
- القيمة **adLockPessimistic** وتوفر التأمين التشاؤمى للسجلات وهو ما يعنى تأمين السجل أثناء تعديله.
- القيمة **adLockOptimistic** وتوفر التأمين التفائلى للسجلات وهو ما يعنى تأمين السجلات عند استدعاء الوظيفة **Update** فقط.
- القيمة **adLockBatchOptimistic** وتوفر تحديث أكثر من سجل فى 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 فى تحديد نوع المؤشر المستخدم على كيفية ارتباط مجموعة السجلات بالبيانات الأساسية الموجودة بقاعدة البيانات.

يوضح جدول ١٢-٣ التالى القيم المختلفة للخاصية **CursorType** ومدلول كل منها.

جدول ١٢-٣ قيم الخاصية **CursorType**

القيمة	المدلول
adOpenForwardOnly	تستخدم لتسريع استرجاع البيانات ولكن تسمح بالتحرك إلى الأمام فقط داخل مجموعة السجلات
adOpenKeySet	تتيح لبرنامجك مشاهدة بعض تغييرات البيانات التى تتم من قِبَل بقية المستخدمين
adOpenDynamic	تتيح لبرنامجك مشاهدة جميع تغييرات البيانات التى تتم من قِبَل بقية المستخدمين

adOpenStatic	لا يمكن من خلال هذه القيمة مشاهدة أى تغييرات تتم من قِبَل بقية المستخدمين

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

من خصائص المؤشر الهامة المستخدمة عند التعامل مع مجموعة السجلات، الخاصية **CursorLocation** والتى 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 فى 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** مخزَّن داخل كائن قابل للاستخدام أكثر من مرة وبالتالى 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 مخزَّن كى يقوم المستخدم بتحديد الحرف الأول بنفسه وذلك كما يلى:

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

بعد ذلك يمكنك إنشاء كائن Command وإخباره بالإجراء المخزَّن وبيانات معامله كما فى الكود التالى:

```
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** التى شرحناها فى هذا الفصل حتى الآن. كما يحتوى الكائن **Command** على مجموعة من الكائنات **Parameter** التى يمثل كل منها معامل داخل الإجراء المخزّن. ويمكنك بعد إنشاء الكائن **Command** احتواء هذه المعاملات واستدعائها مرةً أخرى من خلال عدد قليل من أسطر الكود كما يلى:

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

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

يحتوى الكائن **Command** أيضاً على الوظيفة **()Execute** التى تعمل بطريقة مشابهة لمثيلتها فى 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 التى تتيح لك العمل مع البيانات بطريقة لم تكن تتخيلها من قبل. فعلى سبيل المثال، فى 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 التى تعمل داخل الذاكرة بمعزلٍ 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 التى يستخدمها ما يقرب من مائة ألف مستخدم يحاولون عرض قوائم المنتجات فى نفس الوقت وبالتالى 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** التى تحدد مكان المؤشر إذا كان فى قاعدة البيانات (ناحية الخادم) أم فى 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 فى الكود التالى:

```
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 التى ذكرناها من قبل. فمثلاً فى الكود التالى يتم إنشاء مجموعة سجلات تحتوى على حقلين أحدهما للاسم والآخر لرقم التليفون كما يلى:

```
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 = "Waleed Abdelrazek"
rs.Fields("Phone").Value = "0106521042"
rs.Update()
```

والجديد فى هذا الكود هو استخدام الوظيفة **()Append** التى 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 فى حالة الحقل **Phone**.

يمكنك كذلك حفظ مجموعة السجلات داخل ملف بتنسيق **XML** على القرص الصلب باستخدام الكود التالى:

```
rs.Save("c:\temp.xml",
ADODB.PersistFormatEnum.adPersistXML)
```

كما يمكنك فى أى وقت تحميل هذا الملف مرةً أخرى باستخدام الكود التالى:

```
rs.Open("c:\temp.xml")
```

تحديث قاعدة البيانات

تعرفنا قبل ذلك على كيفية تأمين السجلات لمنع مجموعة المستخدمين من التعامل مع نفس الحقل فى 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 التعييرات على قاعدة البيانات حتى تقوم بتسجيل هذه السجلات. ومن ثمَّ يمكنك تحديث قاعدة البيانات

بطريقتين، الأولى بإرجاء الأمر إلى ADO كى يقوم بعملية التحديث المجمَّعة Batch Update، أما الطريقة الثانية فتتم يدوياً من خلال الكود بمعالجة كل سجل داخل مجموعة السجلات المنفصلة باستدعاء إجراء مخزَّن لتحديث سجل قاعدة البيانات المصاحب. وفيما يلى 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 يلى:

```
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 = "2405330"
rs.Update()
rs.MoveNext()
rs.Delete()
```

٢. إعادة تنشيط عملية الاتصال مع قاعدة البيانات بتخصيص كائن الاتصال المفتوح للخاصية ActiveConnection كما يلى:

```
rs.ActiveConnection = con
```

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

```
rs.MarshalOptions = adMarshalModifiedOnly
rs.UpdateBatch()
```

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

تحديث السجلات يدوياً

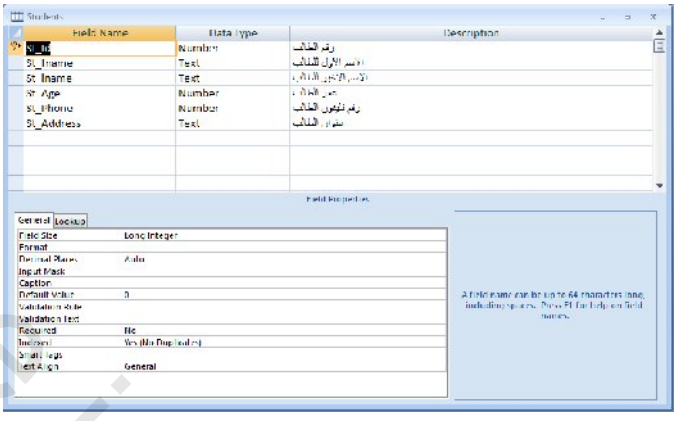
ربما أردت فى بعض الأحيان مزيداً من التحكم فى عملية التحديث، وفى هذه الحالة يمكنك استخدام الخاصية **Status** المصاحبة للكائن **Recordset** لتحديد نوع التحديث الذى يتم على سجل معين، وذلك كما فى الكود التالى:

```
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 تطبيقى

سنقوم فيما يلى بإنشاء مثال تطبيقى على استخدام **ADO** حيث نتعرف على كيفية إظهار مجموعة السجلات وتعديلها من خلال أداة شبكة البيانات **DataGridView** الموجودة داخل **Visual Studio 2008** والتى تدعم استخدام **ADO.NET** التى سنتحدث عنها فى 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 التى تحتوى على نفس الاسم ولتكن الحقول **St_Age** و**St_Phone** و**St_Address** (انظر شكل ١٢-٦).



شكل ١٢-٦ الجدول الذى سنقوم باستخدامه فى التطبيق

٢. افتح بيئة تطوير **Visual Studio 2008** إذا لم تكن مفتوحةً بالفعل ثم قم بإنشاء تطبيق نوافذى 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** إلى مشروعك كما تعلمت من قبل (راجع شكل ١٢-١).

٤. قم بإضافة ثلاثة أزرار أمر من مربع الأدوات إلى أعلى النموذج مع تغيير أسمائهم وعناوينهم كما فى جدول ١٢-٤.

٥. قم بإضافة الأداة **DataGridView** من مربع الأدوات إلى النموذج مع تغيير الخاصية **Name** إلى **grdList**.

٦. قم بإنشاء الكود التالى للأزرار الثلاثة:

جدول ١٢-٤ خصائص أزرار الأمر الموجودة بالنموذج

الزر	العنوان	الاسم
الأيمن	تحميل مجموعة السجلات	**btnLoadRecordset**
الأوسط	كتابة مجموعة البيانات	**btnWriteDataset**
الأيسر	تحميل مجموعة البيانات	**btnLoadDataset**

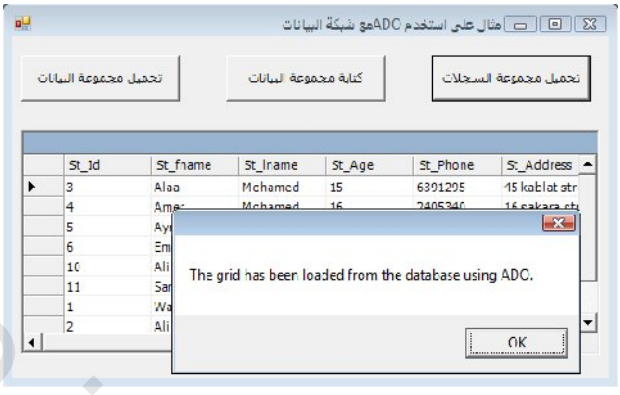
```
Private dsStudent As DataSet

Private Sub btnLoadRecordset_Click(ByVal sender As
System.Object, ByVal e As System.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:\temprs.xml") Then
      System.IO.File.Delete("d:\temprs.xml")
  End If
  rs.Save("d:\temprs.xml",
          ADODB.PersistFormatEnum.adPersistXML)
  dsStudent = New DataSet()
  dsStudent.ReadXml("d:\temprs.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(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnWriteDataset.Click

   If System.IO.File.Exists("d:\tempds.xml") Then
        System.IO.File.Delete("d:\tempds.xml")
    End If
    dsStudent.WriteXml("d:\tempds.xml")
    MessageBox.Show("The contents of the dataset bound to the
                     grid have been written to c:\tempds.xml")
   End Sub
```

```
Private Sub btnLoadDataSet_Click(ByVal sender As
System.Object, ByVal e As System.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 يلى:

- يتم فى الإجراء **btnLoadRecordset()** تحميل مجموعة السجلات من جدول **Students** ثم تخزينها داخل ملف على القرص الصلب باسم **temprs.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 فى الإجراء **btnWriteDataset()** كتابة بيانات مجموعة البيانات إلى الملف **tempds.xml** مع إظهار رسالة تفيد ذلك.
- يتم فى الإجراء **btnLoadDataset()** ملء مجموعة البيانات من الملف **tempds.xml** ثم عرض هذه البيانات داخل شبكة البيانات. أما فى 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 ١٢-٧).



شكل ١٢-٧ 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 مرةً أخرى ثم انقر زر "تحميل مجموعة البيانات"، تلاحظ تحميل البيانات وبها التغييرات التى قمت بإجرائها.

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