Lab 10: Write your own Database code in VB .NET

In this section, we'll take a look at the objects that you can use to open and read data from a Database. We'll recreate what the Wizard has done. That way, you'll see for yourself just what is going on behind the scenes.

Basically speaking, ADO.NET provides a set of classes to support you to develop database applications and enable you to connect to a data source to retrieve, manipulate, and update data with your database. The classes provided by ADO.NET are core in developing a professional data - driven application, and they can be divided into the following three major components:

- Data Provider
- DataSet
- DataTable

Data Provider contains four classes: Connection, Command, DataAdapter, and DataReader. These four classes can be used to perform the different functionalities to help you to:

1. Set a connection between your project and the data source using the Connection object.
2. Execute data queries to retrieve, manipulate, and update data using the Command object.
3. Move the data between your DataSet and your database using the DataAdapter object.
4. Perform data queries from the database (read - only) using the DataReader object.

The DataSet class can be considered as a database container, and it can contain multiple data tables. These data tables are only a mapping to those real data tables in your database. But these data tables can also be used separately without connecting to the DataSet. In this case, each data table can be considered as a DataTable object.

The Connection Object

The Connection Object is what you need if you want to connect to a database. There are a number of different connection objects, and the one you use depends largely on the type of database you're connecting to. Because we're connecting to an Access database, we'll need something called the OLE DB connection object.

OLE stands for Object Linking and Embedding, and it’s basically a lot of objects (COM objects) bundled together that allow you to connect to data sources in general, and not just databases. There are a number of different OLE DB objects (called data providers), but the one we'll use is called "ACE". Others are SQL Server and Oracle.

So place a button on your form. Double click your button to open up the code window. Add the following line:

```
Dim con As New OleDb.OleDbConnection
```

The variable con will now hold the Connection Object. Notice that there is a full stop after the OleDB part. You'll then get a pop up box from where you can select OleDbConnection. We're also creating a New object on this line. This is the object that you use to connect to an Access database.

Setting a Connection String

We need to pass two things to our new Connection Object: the technology we want to use to do the connecting to our database; and where the database is. (If your database was password and user name protected, you would add these two parameters as well. Ours isn't, so we only need the two.)

The technology is called the Provider; and you use Data Source to specify where your database is. So add this to your code:

```
Dim dbProvider As String
Dim dbSource As String

dbProvider = "PROVIDER= Microsoft.ACE.OLEDB.12.0;"
dbSource = "Data Source = C:\AddressBook.accdb"

con.ConnectionString = dbProvider & dbSource
```

The first part specifies which provider technology we want to use to do the connecting (ACE). The second part, typed after a semi-colon, points to where the database is. In the above code, the database is on the C drive, in the root folder. The name of the Access file we want to connect to is called AddressBook.accdb. (Note that "Data Source" is two words, and not one.)

If you prefer, you can have the provider and source on one line, as below (it's on two here because it won't all fit on one line):

```
con.ConnectionString = "PROVIDER=Microsoft.ACE.OLEDB.12.0;Data Source =
C:\AddressBook.accdb"
```

But your coding window should now look like this:

```
Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
    Dim con As New OleDb.OleDbConnection

    Dim dbProvider As String
    Dim dbSource As String

    dbProvider = "PROVIDER= Microsoft.ACE.OLEDB.12.0;"
    dbSource = "Data Source = C:\AddressBook.accdb"

    con.ConnectionString = dbProvider & dbSource
End Sub
```

This assumes that you have copied the AddressBook database over to the root folder of your C Drive. If you've copied it to another folder, change the "Data Source" part to match. For example, if you copied it to a folder called "databases" you'd put this:

Data Source = C:\databases\AddressBook.accdb

But back to our connection code. ConnectionString is a property of the con variable. The con variable holds our Connection Object. We're passing the Connection String the name of a data provider, and a path to the database.

Opening the Connection

Now that we have a ConnectionString, we can go ahead and open the database. This is quite easy - just use the Open method of the Connection Object:

con.Open()

Once open, the connection has to be closed again. This time, just use the Close method:

con.Close()

Add the following four lines to your code:

```
con.Open()

MsgBox("Database is now open.")

con.Close()

MsgBox("Database is now closed.")
```

Your coding window will then look like this (use the file path below, if you have moved the database to your Documents folder):

```
Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
    Dim con As New OleDb.OleDbConnection

    Dim dbProvider As String
    Dim dbSource As String

    dbProvider = "PROVIDER= Microsoft.ACE.OLEDB.12.0;"
    dbSource = "Data Source = C:\Users\USER\Documents\AddressBook.accdb"

    con.ConnectionString = dbProvider & dbSource

    con.Open()
    MsgBox("Database is now open.")

    con.Close()
    MsgBox("Database is now closed.")
End Sub
```

Test out your new code by running your program. Click your button and the two message boxes should display. If they don't, make sure your Data Source path is correct. If it isn't, you might see the error message saying that it can't find the path to the database. The line con.Open in your code will then be highlighted in green. You need to specify the correct path to your database.

Now that we've opened a connection to the database, we need to read the information from it. This is where the DataSet and the DataAdapter come in. ADO.NET uses something called a DataSet to hold all of your information from the database. The DataSet is not something you can draw on your form, like a Button or a Textbox. The DataSet is something that is hidden from you, and just stored in memory.

The Connection Object and the DataSet can't see each other. They need a go-between so that they can communicate. This go-between is called a DataAdapter. The Data Adapter contacts your Connection Object, and then executes a query that you set up. The results of that query are then stored in the DataSet.

The Data Adapter and DataSet are objects. You set them up like this:

```
Dim ds As New DataSet
Dim da As OleDb.OleDbDataAdapter

da = New OleDb.OleDbDataAdapter( sql, con )
```

The Data Adapter

The Data Adapter is a property of the OLEDB object, hence the full stop between the two:

OleDb.OleDbDataAdapter

We're passing this object to the variable called da. This variable will then hold a reference to the Data Adapter.

While the second line in the code above sets up a reference to the Data Adapter, the third line creates a new Data Adapter object. You need to put two things in the round brackets of the Object declaration: Your SQL string (which we'll get to shortly), and your connection object. Our Connection Object is stored in the variable which we've called con. You then pass the New Data Adapter to your variable da:

da = New OleDb.OleDbDataAdapter(sql, con)

We need something else, though. The sql in between the round brackets is the name of a variable. We haven't yet set this up. We'll have a look at SQL in a moment. But bear in mind what the Data Adaptor is doing: Acting as a go-between for the Connection Object and the Data Set

Structured Query Language

SQL (pronounced SeeKwel), is short for Structured Query Language, and is a way to query and write to databases (not just Access). The basics are quite easy to learn. If you want to grab all of the records from a table in a database, you use the SELECT word. Like this:

```
SELECT * FROM Table_Name
```

SQL is not case sensitive, so the above line could be written:

```
Select * from Table_Name
```

But your SQL statements are easier to read if you type the keywords in uppercase letters. The keywords in the lines above are SELECT and FROM. The asterisk means "All Records". Table_Name is the name of a table in your database. So the whole line reads:

```
"SELECT all the records FROM the table called Table_Name"
```

You don't need to select all (*) the records from your database. You can just select the columns that you need. The name of the table in our database is tblContacts. If we wanted to select just the first name and surname columns from this table, we can specify that in our SQL String:

```
SELECT tblContacts.FirstName, tblContacts.Surname FROM tblContacts
```

When this SQL statement is executed, only the FirstName and Surname columns from the database will be returned. There are a lot more SQL commands, but for our purposes this is enough.

Because we want to SELECT all (*) the records from the table called tblContacts, we pass this string to the string variable we have called sql:

```
sql = "SELECT * FROM tblContacts"
```

Your code window should now look like this (though the file path to your database might be different):

```
Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
    Dim con As New OleDb.OleDbConnection

    Dim dbProvider As String
    Dim dbSource As String

    Dim ds As New DataSet
    Dim da As OleDb.OleDbDataAdapter
    Dim sql As String

    dbProvider = "PROVIDER= Microsoft.ACE.OLEDB.12.0;"
    dbSource = "Data Source = C:\Users\USER\Documents\AddressBook.accdb"

    con.ConnectionString = dbProvider & dbSource

    con.Open()

    sql = "SELECT * FROM tblContacts"
    da = New OleDb.OleDbDataAdapter(sql, con)

    MsgBox("Database is now open.")

    con.Close()
    MsgBox("Database is now closed.")
End Sub
```

Now that the Data Adapter has selected all of the records from the table in our database, we need somewhere to put those records - in the DataSet.

Filling the DataSet

The Data Adapter can fill a DataSet with records from a Table. You only need a single line of code to do this:

```
da.Fill(ds, "AddressBook")
```

As soon as you type the name of your Data Adapter (da for us), you'll get a pop up box of properties and methods. Select Fill from the list, then type a pair of round brackets. In between the round brackets, you need two things: the Name of your DataSet (ds, in our case), and an identifying name. This identifying name can be anything you like. But it is just used to identify this particular Data Adapter Fill.

Add the new line after the creation of the Data Adaptor:

```
da = New OleDb.OleDbDataAdapter(sql, con)
da.Fill(ds, "AddressBook")
```

And that's it. The DataSet (ds) will now be filled with the records we selected from the table called tblContact. There's only one slight problem - nobody can see the data yet! What we want to do now is to display the records on a Form, so that people can see them. So do this:

- Add two textboxes to your form
- Change the Name properties of your textboxes to txtFirstName and txtSurname
- Go back to your code window
- Add the following two lines:

```
txtFirstName.Text = ds.Tables("AddressBook").Rows(0).Item(1)
txtSurname.Text = ds.Tables("AddressBook").Rows(0).Item(2)
```

You can add them after the line that closes the connection to the database. Once the DataSet has been filled, a connection to a database can be closed.

Your code should now look like this:

```
Dim con As New OleDb.OleDbConnection

Dim dbProvider As String
Dim dbSource As String

Dim ds As New DataSet
Dim da As OleDb.OleDbDataAdapter
Dim sql As String

dbProvider = "PROVIDER= Microsoft.ACE.OLEDB.12.0;"
dbSource = "Data Source = C:\Users\USER\Documents\AddressBook.accdb"

con.ConnectionString = dbProvider & dbSource

con.Open()
Sql = "SELECT * FROM tblContacts"
da = New OleDb.OleDbDataAdapter(sql, con)
da.Fill(ds, "AddressBook")
MsgBox("Database is now open.")

con.Close()
MsgBox("Database is now closed.")

txtFirstName.Text = ds.Tables("AddressBook").Rows(0).Item(1)
txtSurname.Text = ds.Tables("AddressBook").Rows(0).Item(2)
```

Before the code is explained, run your program and click the button. You should see "John Smith" displayed in your two textboxes. So let's examine the code that assigns the data from the DataSet to the textboxes. The first line was this:

```
txtFirstName.Text = ds.Tables("AddressBook").Rows(0).Item(1)
```

It's rather a long line! But after the equals sign, you type the name of your DataSet (ds for us). After a full stop, select Tables from the popup list. The Tables property needs something in between round brackets. This is NOT the name of your database table! It's that identifier you used with the Data Adapter Fill. We used the identifier "AddressBook".

Type a full stop and you'll see another list popping up at you. Select Rows from the list. In between round brackets, you need a number. This is a Row number from the DataSet. We want the first row, which is row zero in the DataSet:

```
ds.Tables("AddressBook").Rows( 0 )
```

Type full stop after Rows(0) and the popup list appears again. To identify a Column from the DataSet, you use Item. In between round brackets, you type which column you want:

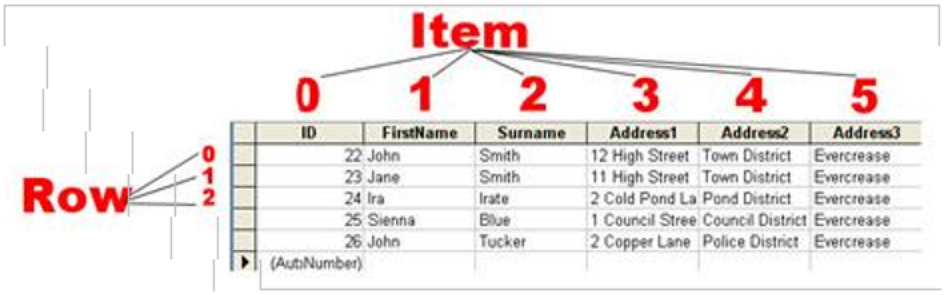
```
ds.Tables("AddressBook").Rows(0).Item( 1 )
```

In our Access database, column zero is used for an ID field. The FirstName column is the second column in our Access database. Because the Item collection is zero based, this is item 1 in the DataSet.

You can also refer to the column name itself for the Item property, rather than a number. So you can do this:

```
ds.Tables("AddressBook").Rows(0).Item("FirstName")
ds.Tables("AddressBook").Rows(0).Item("Surname")
```

If you get the name of the column wrong, then VB throws up an error. But an image might clear things up. The image below shows what the items and rows are in the database.



The image shows which are the Rows and which are the Items in the Access database Table. So the Items go down and the Rows go across.

However, we want to be able to scroll through the table. We want to be able to click a button and see the next record. Or click another button and see the previous record. You can do this by incrementing the Row number. To see the next record, we'd want this:

```
txtFirstName.Text = ds.Tables("AddressBook").Rows(1).Item(1)
txtSurname.Text = ds.Tables("AddressBook").Rows(1).Item(2)
```

The record after that would then be:

```
txtFirstName.Text = ds.Tables("AddressBook").Rows(2).Item(1)
txtSurname.Text = ds.Tables("AddressBook").Rows(2).Item(2)
```

So by incrementing and decrementing the Row number, you can navigate through the records. Let's see how that's done.