Simple Database Programming in VB

The aim of this article is to introduce the basic principles of using Microsoft databases to store information in Visual Basic. It has been written for the Microsoft DAO 3.6 library (dao360.dll), but should be largely compatible with further versions. This library deals with Microsoft Jet databases, ie. .mdb files.

# Setting Up

To use the dao library, go to the menu Project->References... This brings up a list of the libraries that your project is currently referencing and can hence use, along with a list of the other available libraries on your computer. Scroll down to "Microsoft DAO 3.6 Object Library" and click in the check box to reference it. If it does not exist in the list, you may not have this library installed. If there is a similar item, use that. Otherwise try clicking on "Browse..." and going to "C:\Program Files\Common Files\Microsoft Shared\Dao\" This directory should contain the DAO libraries.

Click OK to close the references window and then go to "View->Object Browser" or press F2 to open the Object Browser window. This window can be used to view the current libraries that you are referencing - it is a very handy tool. Select "DAO" from the drop down list at the top left and you will be able to see everything in the DAO library. For instance, if you click on "Database" in the list on the left, this will then list all of the properties, events and methods that are available for the Database object.

# Creating a Database

Creating a database is very easy. In the "globals" list for the DAO library, there is a method "CreateDatabase".

Function CreateDatabase(Name As String, Locale As String, [Option]) As Database

Name is the full path to the database that you want to create.  
Locale is one of the constants defined in LanguageConstants in the Object Browser.  
We are not considering the Option argument.

Place the following code into a Form.

Private Sub CreateDB(Path As String)

' Define a Database variable  
Dim OurDB As Database  
  
' Create our database  
Set OurDB = CreateDatabase(Path, dbLangGeneral)  
  
' OurDB now refers to the database we just created. We don't want to do anything else, so close it.  
OurDB.Close  
  
' Destroy the database object to free memory  
Set OurDB = Nothing

End Sub  
  
Private Sub Form\_Load()

CreateDB "C:\OurDB.mdb"

End Sub

Running this program will create a database at C:\OurDB.mdb and then close it. If you have MS Access installed, you can open this file - some versions of Access will try to persuade you to convert the database to a newer version. Don't do this or you won't be able to access it with your program again.

You will notice that the database is a bit useless as it stands - there is nothing whatsoever in it. The next step is to add a table so that we can store some information. Creating a table is done with the CreateTableDef function which is a member of the Database object.

Function CreateTableDef([Name], [Attributes], [SourceTableName], [Connect]) As TableDef

Add the following code to CreateDB() after the line "Set OurDB = CreateDatabase(...)"

Dim OurTable As TableDef  
  
' Create a table  
Set OurTable = OurDB.CreateTableDef("OurTable")  
  
' Add some fields to the table  
OurTable.Fields.Append OurTable.CreateField("Name", dbText)  
OurTable.Fields.Append OurTable.CreateField("Age", dbInteger)  
  
' Add the table to our database  
OurDB.TableDefs.Append OurTable  
  
Set OurTable = Nothing

This requires some further explanation. One of the properties of a Table is the Fields property. This contains a collection of Field objects that define all of the information for the Table fields. We want to add some fields to our table, so we use the Fields.Append method which takes a Field as an argument. To create this field we have used the Table method CreateField.

Function CreateField([Name], [Type], [Size]) As Field

The Field types are defined in DataTypeEnum which can be found in the Object Browser.

In the same way that a Table has a collection of Fields, a Database has a collection of Tables called TableDefs. Our Table can be added to the collection using the TableDefs.Append method.

Before you try and run this code you will need to delete the file C:\OurDB.mdb.

If you open C:\OurDB.mdb in Access again, you will now be able to see a Table called "OurTable". Opening the table will show the two fields that we added.

# Adding Data

Adding data is a little more complicated. First let us assume that we have already got a database which of course we do. We will now look at opening an existing database.

Add the following code to the form

Private Sub AddRecord(Path As String)

Dim OurDB As Database  
Set OurDB = OpenDatabase(Path)  
  
OurDB.Close  
Set OurDB = Nothing

End Sub

This is the code to open and close an existing database. To add data we need to open a record set. Add the next piece of code before the "OurDB.Close" line.

Dim OurRS As RecordSet  
' Open the table as a recordset  
Set OurRS = OurDB.OpenRecordset("OurTable", dbOpenTable)  
  
' Add a new record  
OurRS.AddNew  
' Set the values of the new record  
OurRS.Fields(0).Value = InputBox$("Input name")  
OurRS.Fields(1).Value = CInt(InputBox$("Input age"))  
OurRS.Update  
OurRS.Close

Add a command button called cmdAdd and add this code to the form:

Private Sub cmdAdd\_Click()

AddRecord "C:\OurDB.mdb"

End Sub

Now you can add a new record to the table by clicking the cmdAddButton. Open the database in Access to view the changes.

# Reading Data

After having added data, reading it back again is not so difficult.

Private Sub ReadAllData(Path As String)

Dim OurDB As Database  
Dim OurRS As Recordset  
Dim i As Integer  
  
Set OurDB = OpenDatabase(Path)  
Set OurRS = OurDB.OpenRecordset("OurTable", dbOpenTable)  
If OurRS.RecordCount > 0 Then

' Go to the first record in the list  
OurRS.MoveFirst  
For i = 0 To OurRS.RecordCount - 1

Debug.Print OurRS.Fields(0).Value & " " & OurRS.Fields(1).Value  
OurRS.MoveNext

Next

End If  
OurRS.Close  
OurDB.Close  
Set OurRS = Nothing  
Set OurDB = Nothing

End Sub

# Deleting Data

To delete a record, we first have to find it. When we have an open recordset the simplest way to do this is in a similar way to our ReadAllData method.

Private Sub DeleteRecord(Path As String)

Dim OurDB As Database  
Dim OurRS As Recordset  
Dim i As Integer  
Dim OurAge As Integer  
Dim OurName As String  
  
OurName = InputBox$("Input name to delete")  
OurAge = CInt(InputBox$("Input age"))  
  
Set OurDB = OpenDatabase(Path)  
Set OurRS = OurDB.OpenRecordset("OurTable", dbOpenTable)  
If OurRS.RecordCount > 0 Then

OurRS.MoveFirst  
For i = 0 To OurRS.RecordCount - 1

If LCase$(OurRS.Fields(0).Value) = LCase$(OurName) And OurRS.Fields(1).Value = OurAge Then

' Delete the currently selected record  
OurRS.Delete  
Exit For

End If  
OurRS.MoveNext

Next

End If  
OurRS.Close  
OurDB.Close  
Set OurRS = Nothing  
Set OurDB = Nothing

End Sub

This assumes that we want to delete the first matching record and that the name is not case sensitive.

# Editing Data

My record editing example is identical to the record deleting example but substituting the "OurRS.Delete" line with:

OurRS.Edit  
OurRS.Fields(1).Value = CInt(InputBox$("Input new age"))  
OurRS.Update

This allows you to change the age of the people in the database.

# Data Bound Controls

A data bound control is a control that connects to a data source - a database. Many controls can be data bound, we are going to look at just two though; the Microsoft ADO Data Control and the Microsoft Hierarchical FlexGrid Control. To add these controls to your project, go to Project->Components... and select them from the list.

Add one each of the two new controls to your form. For simplicity we shall keep the default names of MSHFlexGrid1 and Adodc1.

If you now select MSHFlexGrid1 and look at the DataSource property, you should see that there is an entry for Adodc1 - this means that the Adodc is a compatible data source for the HFlexGrid. Our Adodc1 control will be bound to our database and it will provide the data to MSHFlexGrid1 for displaying.

Edit the Form\_Load event so that it appears as follows:

Private Sub Form\_Load()

' Connect to our database  
Adodc1.ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;Persist Security Info=False;Data Source=C:\OurDB.mdb"  
  
' Select the records that we want to view  
Adodc1.RecordSource = "select Name,Age from OurTable"  
  
' Refresh Adodc1 and any connected controls (ie. MSHFlexGrid1)  
Adodc1.Refresh

End Sub

If you are using a different version of the DAO library, the provider string may need to be altered slightly. For dao350.dll for instance, use

Adodc1.ConnectionString = "Provider=Microsoft.Jet.OLEDB.3.51;Persist Security Info=False;Data Source=C:\OurDB.mdb"

Other than that, the only important part for us is "...Source=C:\OurDB.mdb". This defines which database to use and could of course be specified with a variable:

Adodc1.ConnectionString = "Provider=Microsoft.Jet.OLEDB.3.51;Persist Security Info=False;Data Source=" & Path

The RecordSource statement uses SQL. SQL stands for Structured Query Language and is a standard way of selecting data from a database. This is not meant to be a tutorial on SQL so I shall not be offering many different commands to put in this string. In the form shown, we are selecting the fields Name and Age of all records from the table OurTable. If we only wanted to look at the Name field we could use the following line:

Adodc1.RecordSource = "select Name from OurTable"

Add a command button called cmdAdd to the form along with this code:

Private Sub cmdAdd\_Click()

AddRecord "C:\OurDB.mdb"  
Adodc1.Refresh

End Sub

Then run the program and click on the button to add a new record. After you have finished the HFlexGrid should show your new record.

# Searching for Data

A very important part of databases is the ability to search for data. Add a text box called txtSearch and two command buttons called cmdSubSearch and cmdWordSearch to the form. We are going to use an extended version of the SQL select command to do our searching:

Private Sub cmdSubSearch\_Click()

Adodc1.RecordSource = "select Name,Age from OurTable where instr(Name,'" & txtSearch.Text & "')>0"  
Adodc1.Refresh

End Sub  
  
Private Sub cmdWordSearch\_Click()

If Len(txtSearch) > 0 Then

Adodc1.RecordSource = "select Name,Age from OurTable where Name= '" & txtSearch.Text & "'"

Else

Adodc1.RecordSource = "select Name,Age from OurTable"

End If  
Adodc1.Refresh

End Sub

cmdSubSearch searches Name for the substring specified in txtSearch whereas cmdWordSearch searches for the exact name.

An alternative way of doing this would be to put the code from cmdSubSearch in the txtSearch\_Change event - the search would then occur as you type.

# The End

This concludes my introduction to database programming in VB. Hopefully what I have shown here will allow you to get to grips with this topic. The project I created for this tutorial including all code and a little bit more besides is available here: [Tutorial Files(4KB)](http://atchoo.org/files/vb/SimpleDB.zip)