PRELIMINARY

DBEngine (version 3.0) Custom Control Programmer's Guide - Visual Basic Edition

Introduction

This manual is intended to show the Visual Basic programmer how to use the DBEngine 3.0 custom control. This document contains a description of the example programs shipped with the custom control and discusses DBEngine concepts.

This document is a preliminary copy of the DBEngine 3.0 Custom Control Programmer's Guide - Visual Basic Edition. As such, it is contains only a subset of the original document. Specifically, only the example programs are presented here. Registered users of the DBEngine 3.0 Custom Control will receive a complete release copy of the document.

The DBEngine 3.0 Custom Control Reference Guide (distributed in the DBENG3.ZIP pre-registration evaluation file) should be used as a reference document when working with the DBEngine 3.0 Custom Control (DBENG3.VBX).

DBEngine Example Programs

DEMO1

The example program DEMO1 is a simple program which shows how to use the DBEngine in a Visual Basic program. There are no sophisticated algorithms or code present in the DEMO1 program, just simple routines that show you how to get started programming the DBEngine. The DEMO1 program attatches to a single database table C:\CUSTOMER. Notice that all of the DEMO programs expect the CUSTOMER table to be present in the root directory on drive C. If you desire to change the location of this database (**CUSTOMER.DB** and **CUSTOMER.PX**) you must make a corresponding change to the source code to reflect the new location of the CUSTOMER database.

The example database files are not included in this package. However, there is a menu selection called **Create** in each of the example programs which creates the necessary database files. The database files are created on drive C, in the root directory. Make sure that **SHARE** is loaded before using the DBEngine Custom Control. Since the database files are not included with the package, you will receive an error message the first time you run the example program. After the error message go ahead and select the **Create** database option off the main menu to create the required database table.

The CUSTOMER database is used in the DEMO1, DEMO2, and DEMO3 example programs. The CUSTOMER database has the following structure:

Field Name	**FieldType**
Account Number	N*
Last Name	A30
First Name	A20
Middle Initial	A1
Address	A50
City	A30
State	A2
Zip	A10
Phone	A14
Notes	M200

Note: Fields marked with an * represent a key field.

The DEMO1 form (Customer Information Form) contains ten label controls, ten text controls, eight push button controls and one DBEngine control. The ten label controls are present simply to label the ten text controls and are provided for the user interface only. The ten text controls are our windows into the CUSTOMER database. There is a text control available for each field present in the CUSTOMER database. The Visual Basic form's label controls and corresponding text controls provide us with a visual representation of the CUSTOMER database's record structure.

Seven of the eight pushbutton controls are used for database operations. The **Clear** pushbutton simply clears the text controls on the form, providing a simple way to get a blank Customer Information form:

```
Sub ClearButton_Click ()
'Here we simply clear all the fields on the form.
'There are no database specific routines involved here.

AccountNumber.Text = ""
LastName.Text = ""
FirstName.Text = ""
MiddleInitial.Text = ""
Address.Text = ""
City.Text = ""
State.Text = ""
Zip.Text = ""
Phone.Text = ""
Notes.Text = ""
AccountNumber.SetFocus

End Sub
```

The DEMO1 program is well commented so you can see exactly what is happening in the code. As we progress through the description of the internal workings of this program, we will look at the actual DEMO1 source code.

Every line of source code present in the DEMO1 program will be covered in this section. We will begin with the Form_Load subroutine code. Here is a listing of the Form_Load subroutine source code of the DEMO1 program:

```
Sub Form_Load ()

'Here we will set up the DBEngine's table parameters as follows:
'1.specify the database table to be associated with this DBEngine object.
'2.specify what indexes to open the table on.
'3.specify whether or not to buffer changes to disk.

Customer.TableName = "C:\CUSTOMER"  'Specify the database table name
Customer.IndexID = 0                'Open table with all associated indexes
Customer.SaveEveryChange = False    'Buffer changes to disk.

Customer.Action = OpenTable     'Open the database table.
FillForm                        'Fill the form with the info from the first record.

End Sub
```

As can be seen above the DBEngine object needs to be associated with a database table file. The **Customer.TableName = "C:\CUSTOMER"** statement specifies what database table file will be used in subsequent DBEngine operations. The next line **Customer.IndexID = 0** specifies that the database table will be opened with all associated indexes. The **Customer.SaveEveryChange = False** statement will buffer all table changes to disk.

The next statement **Customer.Action = OpenTable** actually opens the database table for processing. Should something go wrong during the *OpenTable* action the *Customer.Reaction* property will have a non-zero value. When the DBEngine opens a database table the current record is always the first record present in the table image (with *IndexID* = 0 the table will be opened on the primary index if the table has a primary index.)

Next we call a subroutine that is responsible for placing the record information into the appropriate fields in our Visual Basic form. The **FillForm** subroutine does just that.

Here is the FillForm subroutine in its entirety:

```
Sub FillForm ()
'Here we wish to develop a subroutine that fills in our
'Customer Information form with the information present in the
'current record of our database
'
'Notice that this Subroutine is not position dependent
'It always fills in the form with the information present
'in the current record.

'We will use other routines to move around in the database
'table. We can then call this routine to fill in our form.
'This routine is reusable!

Customer.Action = GetRecord       'Get the current record

If (Customer.Reaction <> 0) Then  'Check for error.
  DBENG_Error
```

```
End If

Customer.FieldName = "Account Number"           'Database field of
                                                        'interest

Customer.Action = GetField                      'Get the field value
AccountNumber.Text = Customer.FieldValue'Place the field '
                                                'value into 'the
                                                'appropriate text box.
'
'Repeat this three-step process for each and every field on the form.
'
Customer.FieldName = "Last Name"
Customer.Action = GetField
LastName.Text = Customer.FieldValue

Customer.FieldName = "First Name"
Customer.Action = GetField
FirstName.Text = Customer.FieldValue

Customer.FieldName = "Middle Initial"
Customer.Action = GetField
MiddleInitial.Text = Customer.FieldValue

Customer.FieldName = "Address"
Customer.Action = GetField
Address.Text = Customer.FieldValue

Customer.FieldName = "City"
Customer.Action = GetField
City.Text = Customer.FieldValue

Customer.FieldName = "State"
Customer.Action = GetField
State.Text = Customer.FieldValue

Customer.FieldName = "Zip"
Customer.Action = GetField
Zip.Text = Customer.FieldValue

Customer.FieldName = "Phone"
Customer.Action = GetField
Phone.Text = Customer.FieldValue

Customer.FieldName = "Notes"
Customer.Action = GetField
```

```
Notes.Text = Customer.FieldValue

End Sub
```

As can be seen from the above code, this FillForm subroutine is not position dependent. The FillForm subroutine can be called at any time to fill the Visual Basic form with information present in the current database record. Other subroutines or functions will be responsible for moving the DBEngine record pointer (moving from one record to another). Once positioned, a call to FillForm will get the information from the DBEngine control and fill in the Visual Basic form with the current record's information. It is good practice to construct code modules such as the FillForm subroutine which are reusable. Building reusable code modules will reduce an applications overall design time and makes modifications much easier to perform.

It is good DBEngine programming practice to make at least two code modules for each visual form in an application, one that "fills" a form with information from the current record, and one that transfers information from the form to the current record. These two modules should be general purpose, position independent modules that can be used and reused throughout an application.

Looking at these two subroutines (**Form_Load** and **FillForm)** we have all the code required to specify a database table, open the table, and fill a Visual Basic form with a records worth of information.

Let's now look at the **FormToRecord** subroutine. This is the last large subroutine present in the DEMO1 program. The FormToRecord subroutine transfers information from the Customer Information form to the current database record. This subroutine is general purpose and is in no way position dependent. Here it is in it's entirety:

```
Sub FormToRecord ()
'This subroutine transfers the information present in the
'Customer Information form to the database's current record.

Customer.FieldName = "Account Number"           'Target database field
Customer.FieldValue = AccountNumber.Text'Target field value
Customer.Action = PutField                      'Place it in the Account
                                                        'Number field in
the                                                     'current record.

'Do the same three-step process for each remaining field in the Customer
'Information form.
```

```
Customer.FieldName = "Last Name"
Customer.FieldValue = LastName.Text
Customer.Action = PutField

Customer.FieldName = "First Name"
Customer.FieldValue = FirstName.Text
Customer.Action = PutField

Customer.FieldName = "Middle Initial"
Customer.FieldValue = MiddleInitial.Text
Customer.Action = PutField

Customer.FieldName = "Address"
Customer.FieldValue = Address.Text
Customer.Action = PutField

Customer.FieldName = "City"
Customer.FieldValue = City.Text
Customer.Action = PutField

Customer.FieldName = "State"
Customer.FieldValue = State.Text
Customer.Action = PutField

Customer.FieldName = "Zip"
Customer.FieldValue = Zip.Text
Customer.Action = PutField

Customer.FieldName = "Phone"
Customer.FieldValue = Phone.Text
Customer.Action = PutField

Customer.FieldName = "Notes"
Customer.FieldValue = Notes.Text
Customer.Action = PutField

End Sub
```

We will now look at two of the push button controls (**Insert** and **Update**) present on the Customer Information form.

The Insert push button makes a call to our FormToRecord subroutine which transfers information from the Customer Information form to the current record. It then inserts the current record into the database table.

The Update push button also makes a call to FormToRecord to transfer information from the form to the current record. It then however, updates the current record in the database table.

Let's look at the code under these two buttons:

```
Sub InsertButton_Click ()
'This subroutine inserts a record into our database.

FormToRecord      'Transfer info from form to current record.

Customer.Action = InsertRecord   'Insert the record

If (Customer.Reaction <> 0) Then     'If an error
  DBENG_Error
End If

End Sub


Sub UpdateButton_Click ()
'This subroutine updates a record in our database.

FormToRecord     'Transfer info from form to record

Customer.Action = UpdateRecord  'Update record

If (Customer.Reaction <> 0) Then    'if error
  DBENG_Error
End If

End Sub
```

There is not a lot of code here. We just call our FormToRecord subroutine to transfer the information present in the Customer Information form to the current database record. Then we simply perform the appropriate DBEngine Action to insert, or update the current record (there is an error handling subroutine **DBENG_Error** which we will leave to the end of our DEMO1 program discussion.)

The otther push button controls also have very simple code attatched to them:

```
Sub TopButton_Click ()

'This code will make the current record the first record
'in the table.
'It will then fill our form with the data present in the
'first record.

Customer.Action = FirstRecord
If (Customer.Reaction <> 0) Then
  DBENG_Error
End If
FillForm

End Sub


Sub BottomButton_Click ()
'This subroutine moves to the last record in the database
'table and makes that record the current record.
'We then call the FillForm subroutine to place the record
'information on our form.

Customer.Action = LastRecord
If (Customer.Reaction <> 0) Then
  DBENG_Error
End If
FillForm

End Sub



Sub NextButton_Click ()
'This subroutine moves to the next record in the database
'table and makes that record the current record.
'We then call the FillForm subroutine to place the current
'record information into our Visual Basic form.

Customer.Action = NextRecord
If (Customer.Reaction <> 0) Then
  DBENG_Error
End If
FillForm

End Sub
```

```
Sub PreviousButton_Click ()
'This subroutine moves to the previous record in the database
'table and makes that record the current record.
'We then call the FillForm subroutine to place the current
'record information into our Visual Basic form.

Customer.Action = PreviousRecord
If (Customer.Reaction <> 0) Then
  DBENG_Error
End If
FillForm

End Sub


Sub DeleteButton_Click ()
'This subroutine deletes the current record in the database
'table. The FillForm subroutine is then called to fill the
'Visual Basic form with the new current record's information.

Customer.Action = DeleteRecord
If (Customer.Reaction <> 0) Then
  DBENG_Error
End If
End Sub
```

The code attatched to the above push button controls consist of database related procedures which are performed in a single line of Visual Basic source code. All of the DBEngine Actions are performed with a single line of code. Of course many require that specific information be present in other DBEngine properties (for example the DeleteTable Action expects the name of the database table to be present in the DBEngine.TableName property.)

All DBEngine Actions, if performed successfully, will place a zero in the DBEngine.Reaction property. If the DBEngine Action could not be performed successfully, a non-zero value (error code) is placed in the DBEngine.Reaction property. It is good programming practice to construct error handling routines to detect and handle DBEngine errors. A skeletal DBEngine error handling routine **DBENG_Error** is present in the DEMO1 example program:

```
Sub DBENG_Error ()

'This is a skeletal DBEngine error handling routine.
'To construct a complete error handling routine, a separate
'Case statement for every known error (see Error codes in the
'DBEngine Custom Control Reference Manual) should be present
'in the error handling routine.

Select Case Customer.Reaction

Case 101
  MsgBox "Error code:" + Str$(Customer.Reaction) + "-End of table!"
Case 102
  MsgBox "Error code:" + Str$(Customer.Reaction) + "-Start of table!"

Case Else
  MsgBox "Error code:" + Str$(Customer.Reaction) + "-Unknown error!"
End Select
End Sub
```

The above code is the entire Visual Basic source code for the DEMO1 example program. We will now move on to discuss the DEMO2 example program.

DEMO2

The DEMO2 example program is a continuation of the DEMO1 program. The DEMO2 program contains all of the Visual Basic source code present in the DEMO1 program and has one extra subroutine which demonstrates how to search for a value in a database field.

The code to search on a database field is present in the **AccountNumber_LostFocus** subroutine. The AccountNumber Text control is set up to receive a user supplied Account Number. As soon as the user moves out of the AccountNumber Text box (LostFocus), a search is performed on the CUSTOMER database on the Account Number field. If the user-supplied value is located in the CUSTOMER database, the information for that record is displayed in the Customer Information form. If the search was not successfull, the Customer Information form is cleared, the user-supplied Account Number value is placed in the AccountNumber Text control and the user is then expected to enter a new account. Here is the **AccountNumber_LostFocus** subroutine in its entirety:

```
Sub AccountNumber_LostFocus ()

'This LostFocus routine is used to search on the
'Account Number field for a specific user supplied
'Account Number. If the Account Number the user supplies
```

```
'is not found in the database table, the form is cleared,
'the user supplied Account Number is placed back into the
'Account Number field, and we will assume that we are entering
'a new account.

Dim Temp As String      'Temporary variable to hold user typed Account
                                    'Number

Temp = AccountNumber.Text   'Save user supplied Account Number in
                                             'Temp var.

Customer.FieldName = "Account Number"       'We will search on the
                                                        'Account Number field.

Customer.FieldValue = AccountNumber.Text    'We will search for the value
                                                                'the user entered

Customer.Action = PutField              'Submit the search criteria by placing
                                             'the value to be searched for into the
                                                      'DBEngine custom control's
record                                                        'buffer.

Customer.SearchMode = SearchFirst          'We will search for the first
match
                                                     'starting at the first record in the
                                                              'table.

Customer.Action = SearchField             'Begin search

If (Customer.Reaction <> 0) Then
  ClearButton_Click                'Search failed to find a match, clear the
form
  AccountNumber.Text = Temp          'Restore user supplied Account
Number
  LastName.SetFocus                'Begin to process new account info
Else
  Customer.Action = GetRecord        'Search was successfull, get the
                                                        'record

  FillForm                         'Fill in the form with the information
End If

End Sub
```

The ***SearchField*** Action is described in detail in the DBEngine Custom Control Reference Guide. *SearchField* is used in conjunction with the ***SearchMode*** property. In the above example *SearchMode = SearchFirst* which begins with the first record and continues until it finds the value which was placed into the DBEngine by the *PutField* Action. If *SearchField* fails to locate the value, a non-zero value is placed in the *Reaction* property. If *SearchField* is successfull, zero (0), is placed in the *Reaction* property. Detailed information concerning the DBEngine's available search modes can be found in the DBEngine Custom Control Reference Guide under the **SearchMode** property description.

DEMO3

The DEMO3 program (like the DEMO2 program) consists of the base DEMO1 program with one additional subroutine. Like DEMO2, DEMO3 has a **AccountNumber_LostFocus** subroutine. The DEMO2 program used this subroutine to search a database table on a field using the *SearchField* Action. The DEMO3 program uses *SearchKey* in place of *SearchField*. Using *SearchKey* you are able to search a database on multiple key fields.

The DBEngine 3.0 Custom Control supports cas-sensitive as well as case-insensitive database indexes. It also supports composite secondary indexes. For more information concerning index capabilities consult the DBEngine 3.0 Custom Control Reference Guide.

Here is the DEMO3 **AccountNumber_LostFocus** subroutine in its entirety:

```
Sub AccountNumber_LostFocus ()

'This LostFocus routine is used to search the
'database table's primary key for a specific user supplied
'Account Number. If the Account Number the user supplies
'is not found in the database table, the form is cleared,
'the user supplied Account Number is placed back into the
'Account Number field, and we will assume that we are entering
'a new account.

Dim Temp As String     'Temporary variable to hold user typed Account
                       'Number

Temp = AccountNumber.Text   'Save user supplied Account Number in
                            'Temp var.

Customer.FieldName = "Account Number"      'We will search for a value in
```

```
                                                            'Account Number
field.

Customer.FieldValue = AccountNumber.Text    'We will search for the value
                                                            'the user entered

Customer.Action = PutField      'Submit the search criteria by placing
                                            'the value to be searched for into the
                                                    'DBEngine custom control's
record                                                  'buffer.


Customer.KeySearch = 1          'We will search the primary index which
                                            'consists of one field (Account
                                                    'Number) the 1st field in the
database                                                'table.

Customer.SearchMode = SearchFirst      'We will search for the first match
                                                'starting at the first record in the
                                                        'table.

Customer.Action = SearchKey              'Begin search

If (Customer.Reaction <> 0) Then
   ClearButton_Click               'Search failed to find a match, clear the form
   AccountNumber.Text = Temp          'Restore user supplied Account
Number
   LastName.SetFocus                'Begin to process new account info
Else
   Customer.Action = GetRecord        'Search was successfull, get the
record
   FillForm                         'Fill in the form with the information
End If

End Sub
```

The ***SearchKey*** Action is described in detail in the DBEngine Custom Control Reference Guide. *SearchField* is used in conjunction with the ***SearchMode*** and ***KeySearch*** properties. In the above example *SearchMode* = *SearchFirst* which begins with the first record and continues until it finds the value(s) which was placed into the DBEngine by the *PutField* Action(s). If *SearchKey* fails to locate the value, a non-zero value is placed in the *Reaction* property. If *SearchKey* is successfull, zero (0), is placed in the *Reaction* property. Detailed information concerning the DBEngine's available search modes can be found in the DBEngine Custom Control Reference Guide under the **SearchMode** property description.

COMBO

The final example program COMBO is a simple little demo showing how to fill a combo box with information from a database table. The COMBO example program is filled with comments which describe program operation. We will not detail its internals here, see the program source code for a detailed description.

End of Document