EViews Database Objects Library

Overview

The EViews Database Objects (EDO) Library gives you the ability to access data objects held inside EViews databases and workfiles from within an external application.

The library consists of a set of COM objects exported by EViews that can easily be used in a variety of development environments such as Microsoft .NET and Visual Basic for Applications (VBA).

The API provides full read and write access to EViews databases, as well as read-only access to EViews workfiles. The interface provides access to both the observation values and the attributes of an object. Note that the EViews OLE DB interface also provides access to data within EViews file formats but does not support any write operations and does not support object attributes.

The interface is related to the EViews Database Extension (EDX) API because the main objects in the EViews Database Objects (EDO) Library follow interfaces defined by the EDX API. However, the usage scenario for the two features is quite different. The EDX API allows you to connect EViews to an external data source by implementing your own Database Manager and Database classes. The EDO library allows you to work with data currently stored within EViews file formats in an application other than EViews by using Database Manager and Database objects from a library already implemented by EViews.

The EDO library supports reading and writing of the following EViews data objects:

- Numeric series (including series containing dates)
- Alpha series (containing character data)
- Scalars
- Vectors
- Matrices
- Strings
- String Vectors

The library does not currently support access to data within structured EViews objects such as equations or models.

The EViews Database Objects Library consists of three classes:

- EViewsDatabaseManager: The manager class is the entry point for all database activity. The manager should be constructed once during the session. The main task of the manager is to return a new EViewsDatabase object whenever a particular database or workfile is opened.
- EViewsDatabase: The database class represents a currently open EViews database or workfile. The main tasks of the database class are to read objects, write objects, and search through objects within the database.

- Frequency: The frequency class is a utility class that exports functionality available inside of EViews for working with calendar dates and frequencies. Use of the Frequency class is optional – it is not an argument to any function in the database manager or database classes.

The EViewsDatabaseManager and EViewsDatabase classes implement interfaces defined in the EViews Database Extension (EDX) API. Please see the EDX API documentation for a more detailed discussion of the functions within these interfaces and for full documentation of the Frequency class.

Simple examples

In this section we provide some simple code fragments for common operations.

Opening a database or workfile

To open a database or workfile you must first create an EViewsDatabaseManager class and then call the OpenDb() function provided by the database manager to open the database.

```
  Dim dbPath = "C:\mydir\mydb.edb"

  Dim dbMgr = New EViewsEdx.EViewsDatabaseManager
  Dim db = dbMgr.OpenDb(dbPath, EViewsEdx.OpenCreateMode.FileOpen,
EViewsEdx.ReadWriteMode.FileReadOnly, "", "", "")

  'add code for operations on database here…

  db.Close()
  dbMgr.Close()
```

You should construct only one database manager per session. Note that the Database Objects Library is not currently thread safe, even if you construct a separate DatabaseManager within each thread.

Reading an object

To retrieve an EViews data object from a database or workfile simply call the ReadObject() function on the database class.

```
  Dim attr = Nothing
  Dim vals = Nothing
  Dim ids = Nothing
  db.ReadObject("myseries", "", attr, vals, ids)
```

The first argument to the function is the name of the object you would like to retrieve. The second argument is the destination frequency, which is irrelevant for EViews databases and workfiles since only a single frequency of data can be stored with each object name.

Data from the object is returned in the three arguments attr, vals, and ids. Attr will contain a table of attribute names and values, such as ‘freq’ for frequency, ‘start’ for start date and ‘end’ for end date as

well as any custom attributes the user has added to the object. Names will be listed in the first column of the attribute array and values in the second column. Vals will contain an array of observation values. In most cases this will be a one dimensional array, except for matrices which will contain a two-dimensional array. The array may consist of either numeric, string or date values depending on the type of the object.

The ids array is only returned for series objects. It contains the start date of each observation of the series. The ids array will have the same number of elements as the vals array.

Writing an object

To write an object into an EViews database use the WriteObject() function. The WriteObject function supports a variety of different input formats depending on what object you are trying to write.

To write a time series, you can either provide explicit frequency and start date attributes:

```
  'explicit frequency and start date (no date identifiers)
   Dim attr As String = "freq=Q, start=2010Q1"
   Dim vals = New Double(0 To 3) {1.0, 2.0, 3.0, 4.0}
   db.WriteObject("myseries", attr, vals, Nothing,
EViewsEdx.WriteType.WriteOverwrite)
```

Or you can provide an array of observation dates for each data value and let EViews automatically determine the frequency:

```
  'frequency, start and end date automatically determined from date ids
  Dim vals = New Double(0 To 3) {1.0, 2.0, 3.0, 4.0}
  Dim ids = New Date(0 To 3) {#3/31/2010#, #6/30/2010#, #9/30/2010#,
#12/31/2010#}
  db.WriteObject("myseries", Nothing, vals, ids,
EViewsEdx.WriteType.WriteOverwrite)
```

To write a matrix, set the 'type' attribute with value 'matrix' and pass in a two dimensional array of values:

```
  '4 by 2 matrix passed as array
  Dim attr As String = "type=matrix"
  Dim vals = New Double(0 To 3, 0 To 1) {{1, 2}, {3, 4}, {5, 6}, {7, 8}}
db.WriteObject("mymatrix", attr, vals, Nothing,
EViewsEdx.WriteType.WriteOverwrite)
```

Or pass in a one dimensional array of values and specify the dimensions using the 'rows' and 'cols' attributes.

```
  '4 by 2 matrix described using rows/cols
   Dim attr As String = "type=matrix, rows=4, cols=2"
   Dim vals = New Double(0 To 7) {1, 2, 3, 4, 5, 6, 7, 8}
db.WriteObject("newname", attr, vals, Nothing,
EViewsEdx.WriteType.WriteOverwrite)
```

String vectors can be passed in using an array of strings:

```
  'write a string vector
   Dim attr As String = "type=svector"
   Dim vals = New String(0 To 4) {"here", "are", "some", "string", "values"}
   db.WriteObject("newname", attr, vals, Nothing,
EViewsEdx.WriteType.WriteOverwrite)
```

Scalar and string objects can be written using simple in line values:

```
  db.WriteObject("myscalar", "type=scalar", 100.0, Nothing,
EViewsEdx.WriteType.WriteOverwrite)

  db.WriteObject("mystring", "type=string", "hello EViews!", Nothing,
EViewsEdx.WriteType.WriteOverwrite)
```

Note that the only overwrite options supported by EViews databases are WriteOverwrite, which will overwrite any existing object with the same name, and WriteProtect, which will throw a RECORD_NAME_IN_USE exception whenever there is an existing object with the same name.

Searching for objects

To obtain a list of objects within the database or workfile that satisfy specified criteria, first call SearchInitialize() to begin the search, then call SearchNext() repeatedly to retrieve results for each object one at a time.

A typical example might look like this:

```
  'initialize search
  db.SearchByAttributes("name matches abc*", "name, type")

 'retrieve results
 Dim name As String = Nothing
 Dim attr As Object = Nothing
 While (db.SearchNext(name, attr))
     'add code to process each item here...
 End While
```

The searchExpression argument of SearchByAttributes allows you to specify a criterion to select which objects you would like to return based on their attributes, for example:

"freq=m and start<1970"

will select all monthly series that have history available before 1970.

The attrNames arguments selects which attributes you would like to be returned from each object that matches the search criterion. Attribute names should be separated by commas.

Attributes will be returned by SearchNext() in a two dimensional array with attribute names in the first column and attribute values in the second column.

Managing objects

Objects inside EViews databases can be copied, renamed and deleted using the functions CopyObject, RenameObject and DeleteObject. You may use wildcards within the object names ('?' for a single unknown character and '*' for zero or more unknown characters) to copy, rename or delete several objects at once.

Attempts to copy, rename or delete an object that doesn't exist will cause a RECORD_NAME_INVALID exception to be thrown.

Attempts to copy or rename an object to a name that is already in use by another object will cause a RECORD_NAME_IN_USE exception to be thrown. The following code fragment shows an example of copying an object where the user will be asked whether it is OK to overwrite the existing object.

```
Dim errorCode As EViewsEdx.ErrorCode = 0
Try
  'try copy without overwrite
  db.CopyObject(OldName, NewName, False)

Catch errorInfo As COMException
  errorCode = errorInfo.ErrorCode
  If (errorCode <> EViewsEdx.ErrorCode.RECORD_NAME_IN_USE) Then
     ProcessComError(errorInfo)
  End If
  Exit Try
End Try

'if existing object found, prompt for whether to overwrite
If (errorCode = EViewsEdx.ErrorCode.RECORD_NAME_IN_USE) Then
   If (MsgBox("Object " & NewName & " already exists. Overwrite the existing
object?", MsgBoxStyle.OkCancel) = MsgBoxResult.Ok) Then
     Try
        'user has confirmed overwrite, so try copy with overwrite
        db.CopyObject(OldName, NewName, True)
     Catch errorInfo As COMException
        ProcessComError(errorInfo)
        Exit Try
     End Try
   End If
End If
```

Handling errors

Most errors that occur inside the Database Objects Library are handled by throwing exceptions. These exceptions will appear within the .Net environment as System.Runtime.InteropServices.COMException objects.

For robust error handling, your application should wrap each call to a function within the EDO library inside a try/catch block as follows:

```
Try
    'call function in EDO library
     db.DeleteObject("myseries")
Catch errorInfo As COMException
    'handle any errors here…
     Exit Try
End Try
```

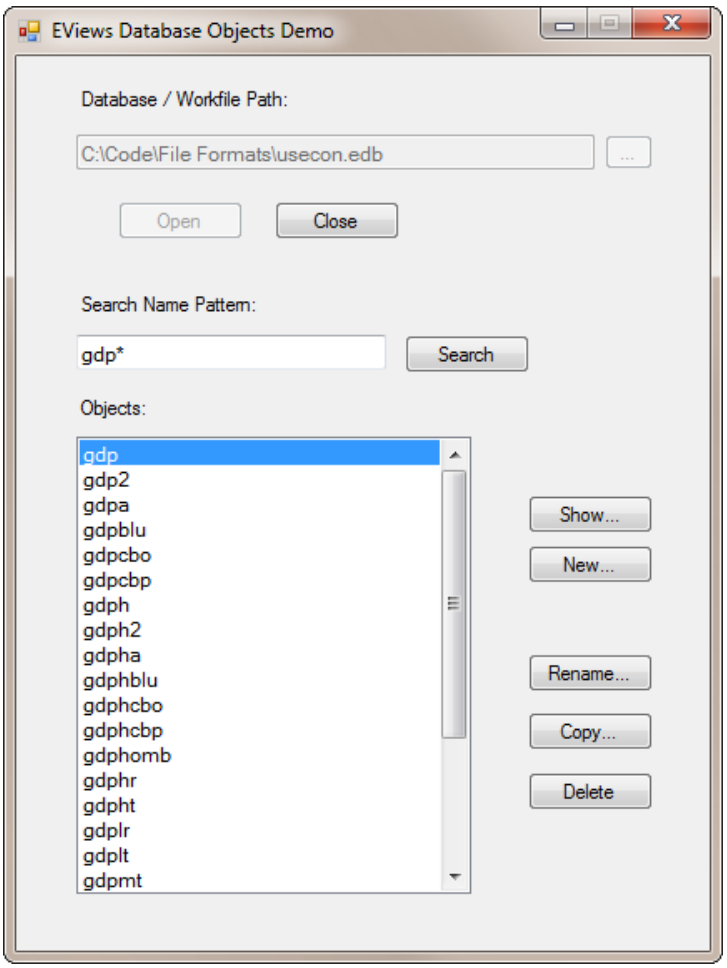
The COMException class contains an ErrorCode member. The value of this member can be compared to values in the EViewsEdx.ErrorCode enumeration to add handling for a specific cause of error within your code.

The COMException class also contains a Message member. This contains a human readable description of the error that occurred. This text should normally be displayed to the user in all cases where you are not specifically handling the error.

Sample Application

The sample project EdoSample available on the EViews web site contains complete source code for a simple application which demonstrates many aspects of the EViews Data Objects library.

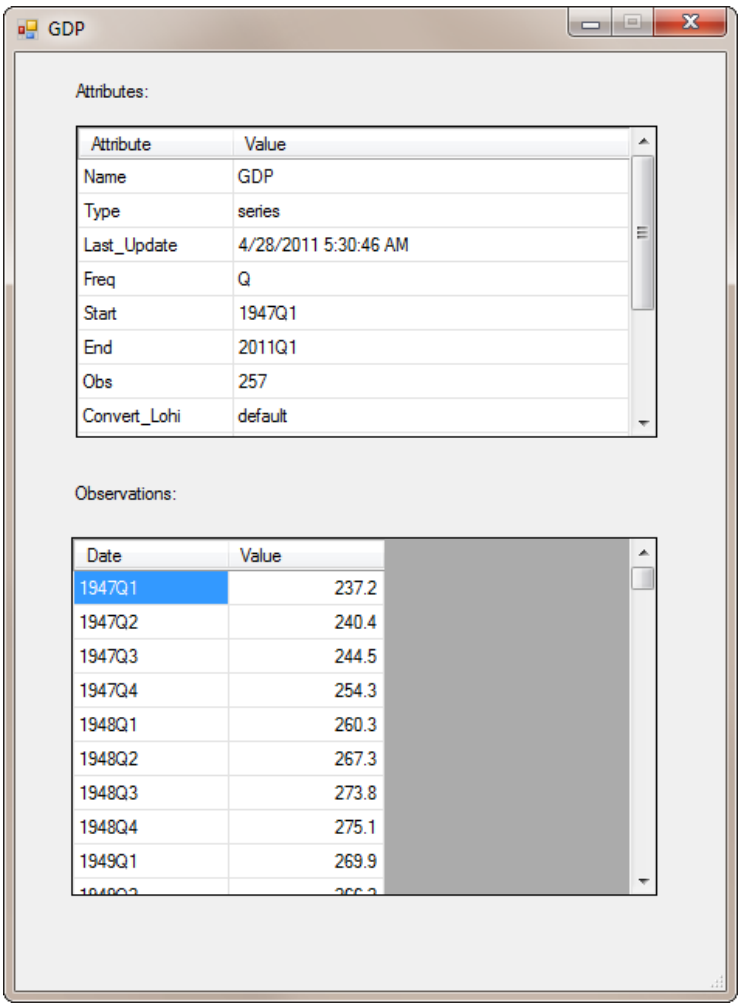
The application opens with a form that lets the user select an EViews database or workfile to open. You may then search for objects within the database or workfile by specifying a name pattern:



If you have chosen to open a database, you can use the Rename, Copy and Delete buttons to rename, copy and delete selected objects within the database. EViews workfiles do not support these functions since the workfile file format does not support partial changes to the workfile content.

This form is implemented in the code file DatabaseForm.vb. The class creates an EViewsDatabaseManager object when the application is launched and holds an EViewsDatabase object for the database that is currently open. All operations on the database are performed within the code of DatabaseForm.

By double clicking on an object you can display the values and attributes of the selected object:



The object display form is implemented in the code file ObjectForm.vb. ObjectForm works entirely with a copy of the attributes and the observations of the object held in memory so it does not directly make use of the EViewsDatabaseManager or EViewsDatabase classes. The code for this class does provide several examples of using the Frequency utility class to move between observation numbers and date values and to format dates for display.

The complete source code for the application is available on our web site. You may find it useful to step through the application to see how various operations have been implemented.