crystal decisions™

Seagate Crystal Data Object

Using the different methods and properties of the Crystal Data Object (CDO)

Overview

The objective of this tutorial is to show you how easy it is to use the different methods of the Crystal Data Object (CDO). Once you have completed this tutorial you will have a good understanding of how to use the different properties and methods of the CDO.

Contents

The Crystal Data Object (CDO) Reference

All CDO methods and properties are part of a single Rowset object exposed by the CDO automation server. This is the only object that can be created that is exposed by CDO, and a Rowset object must be created using the CreateObject function (or early-bound using the "New" keyword of Visual Basic) before any subsequent CDO methods or properties are used.

AddField Method

Description

This method adds new fields to the data set.

Usage

object.AddField fieldName, fieldType

For example:

```
CDORowset.AddField "Order Amount"
```

Parameters

Parameter	Description
fieldName	The name of the new field. This name must match the field name used in the field definition file used to create the report.
fieldType	The type of the field (optional).

Remarks

- The field type of the new field is determined by the first value passed to the field using the AddRows method if you do not use the fieldType parameter.
- Fields can only be added to a Rowset object that has not yet been initialized with data using the AddRows method. If data has been added to a Rowset object, calling AddField will produce a runtime error.

AddRows Method

Description

The AddRows method adds rows of data, or records, to a CDO Rowset object. Data can be defined in an array of VARIANT types. In most cases, the array must be two-dimensional. The first dimension indicates the number of rows that

will be added to the Rowset object. The second dimension must be equal to the number of fields added to the Rowset object with the AddField method. If the Rowset object has only one field, then the array passed to AddRows can be one-dimensional, indicating only the number of rows to be added. The field type of each element in the row must be consistent across all rows contained within the data set. Initially, the field types are undefined until either the AddField method is called with a set data type, or a row is added to the data set. The data types occurring within the first row implicitly define the data types for subsequent rows added to the data set. If subsequent rows do not match the data type defined for that column, an exception is raised (error).

Usage

object.AddRows rowData

For example:

```
CDORowset.AddRows Rows
```

Parameters

Parameter	Description
RowData	An array of Variant types containing the data to be added to the Rowset object. The array must be two-dimensional in most cases. The first dimension indicates the number of rows (records), the second-dimension indicates the number of fields. The size of the second dimension must match the number of fields added to the Rowset object using the AddField method.

Remarks

- Once the AddRows method is called, subsequent calls to AddField will fail. No fields may be added to a Rowset object once it has been initialized with data using the AddRows method.
- AddRows can be called multiple times to add data to a Rowset object. Each time the method is called, data is simply added to the end of the existing rows in the Rowset.

DeleteField Method

Description

This method deletes a field from a Rowset object. Fields can only be deleted from a Rowset object if the Rowset has not been initialized with data using the AddRows method. Once the Rowset has been initialized with data, a call to the DeleteField method will produce an error.

Usage

object.DeleteField fieldName

For example:

```
CDORowset.DeleteField "Order Amount"
```

Parameters

Parameter	Description
FieldName	The name of the field to be removed from the Rowset.

Remarks

- For information on adding fields to a Rowset, see the AddField method.
- For information on how to intialize a Rowset object with data, see the AddRows Method.

GetColCount Method

Description

Returns the number of columns (fields) defined for the Rowset object.

Usage

object.GetColCount

For example:

```
CDORowset.GetColCount
```

Remarks

- The value returned by this function is a 16-bit integer indicating the number of fields (columns) in the Rowset object.

GetEOF Method

Description

Determines if the end of the Rowset has been reached.

Usage

object.getEOF

For example:

```
CDORowset.getEOF
```

Return Value

This method returns True (1) if the end of the Rowset has been reached; False (0) if the end has not been reached.

Remarks

- You can move through a Rowset object at runtime using MoveFirst method, MoveNext method, and MoveTo method.

GetFieldData Method

Description

Returns data for a given field (column) of the Rowset, based on the currently selected record (row).

Usage

object.GetFieldData col

For example:

```
CDORowset.GetFieldData 0
```

Parameters

Parameter	Description
Col	A 16-bit integer indicating the column (field) in the Rowset being sought. This value corresponds to the second dimensional value in the array passed to the AddRows Method.

Remarks

- Only data for the currently selected record (row) is returned for the specified field. You move through the records of a Rowset using MoveFirst method, MoveNext method, and MoveTo method.
- Since the field type for the given field is unknown, this method returns a value of type Variant.

GetFieldName Method

Description

Obtains the name of the field corresponding to the given column. Returns the name of a specified field (column) of the Rowset.

Usage

object.GetFieldName col

For example:

```
CDORowset.GetFieldName 0
```

Parameters

Parameter	Description
col	A 16-bit integer indicating the column (field) in the Rowset being sought. This value corresponds to the second dimensional value in the array passed to the AddRows method.

Remarks

- This method returns a string with the name of the field specified for the Rowset.

GetFieldType Method

Description

Obtains the type of data contained in the named field. This field type is determined when the first row is added to the Rowset object using the AddRows method.

Usage

object.GetFieldType fieldName

For example:

```
CDORowset.GetFieldType "Order Amount"
```

Parameters

Parameter	Description
fieldName	The name of the field (column) for which the field type is required. This name must match one of the field names used with the AddField method.

Remarks

- This method returns a 16-bit integer value that indicates the field type. The following table shows possible values and corresponding field types, along with Visual Basic constants where possible.

Field Type	Value	Visual Basic Constant
Short Integer (16-bit)	2	VbInteger
Long Integer (32-bit)	3	VbLong
Single (float)	4	VbSingle
Double	5	VbDouble
Currency	6	VbCurrency
Date	7	VbDate
String	8	VbString
Boolean	11	vbBoolean

MoveFirst Method

Description

Moves to and selects the first record (row) in the Rowset. The Rowset must contain data before this method can be called.

Usage

object.MoveFirst

For example:

```
CDORowset.MoveFirst
```

Remarks

- This method, along with MoveNext method and MoveTo method, allows you to move through the records of the Rowset at runtime.
- Data for a specific field of the currently selected record can be obtained with the GetFieldData method.
- If the Rowset has not been initialized with data using the AddRows method, this method will produce an error.

MoveNext Method

Description

Moves to and selects the next record (row) in the Rowset after the currently selected record. The Rowset must contain data before this method can be called.

Usage

object.MoveNext

For example:

```
CDORowset.MoveNext
```

Remarks

- This method, along with MoveFirst method and MoveTo method, allows you to move through the records of the Rowset at runtime. Data for a specific field of the currently selected record can be obtained with the GetFieldData method.
- If Rowset has not been initialized with data using the AddRows method, this method will produce an error.

MoveTo Method

Description

Moves to and selects the specified record (row) in the Rowset. The Rowset must contain data before this method can be called.

Usage

object.MoveTo recordNum

For example:

```
CDORowset.MoveTo 5
```

Parameters

Parameter	Description
recordNum	A 32-bit integer indicating the record (row) to move to and select. This is a 0 (zero) based index where the first record is number 0, the second is number 1, etc.

Remarks

- This method, along with MoveFirst method, and MoveNext method, allows you to move through the records of the Rowset at runtime. Data for a specific field of the currently selected record can be obtained with the GetFieldData method.
- If Rowset has not been initialized with data using the AddRows method, this method will produce an error.

Reset Method

Description

Removes all data and fields for the Rowset object. Both field names and records are removed.

Usage

object.Reset

For example:

```
CDORowset.Reset
```

Remarks

- The Rowset becomes uninitialized again, and fields and rows must be added using AddField method, and AddRows method.

RowCount

Description

Specifies the number of rows contained within the data set.

Usage

object.RowCount

Remarks

- Use this property to find out how many rows currently exist in a CDO Rowset. The value obtained is a 32-bit integer.
- This property is read-only. You can not assign the number of rows in a Rowset using this property. To add rows, you must use the AddRows method.

Tutorial which uses the methods of the CDO object

This tutorial shows you how to use most of the methods of the CDO object. The methods GetEOF and DeleteField methods are not used in this tutorial. These methods can be added later into the project created by this tutorial.

1. Open the Seagate Crystal Reports application, if it is not already running.
2. Create a new Report by selecting New from the File menu and choosing Standard expert.
3. For report data choose SQL/ODBC and the Active Data(Field Definitions Only) driver.
4. From the Data Definition dialog choose the orders.ttx, which is located in the Seagate Crystal Reports directory, and Add this as the report's data

source. (depending on your version of the active data driver (P2SMON.DLL), the dialog box will look like either Figure 1 or Figure 2.

FIGURE 1

FIGURE 2

5. Select the Fields tab and Add the fields OrderID, Order Amount, Order Date and Shipped Via.

6. Select the Sort tab and sort the report by the Shipped Via field.

7. Select the Total tab and total the Order Amount field.

8. Select the Style tab and choose Shading and in the Report Title text box type “CDO Tutorial”.

9. Click Preview Sample.

10. Ensure the option “Save Data with report” is unchecked in the File menu and then create a new folder and save the report to this folder.

11. Open the Visual Basic 5.0 or 6.0 application, if it is not already running.

12. Create a new Standard EXE project by either selecting one from the Visual Basic start up dialog or navigating to New Project under the File menu.

13. As this tutorial will be using the CDO object and CPEAUT automation server you will need to include the reference in the project for early binding. From the menu bar select Project | References and check Crystal Data Object and Crystal Report Engine x.x Object Library, where x.x is the version. Click OK.

14. In the General Declaration section declare the following variables. These variables are declared at the module level so they will not go out of scope.

```
Dim rsCDO As CrystalDataObject.CrystalComObject
Dim curRecord As Long 'used to track current record
Dim appl As CRPEAuto.Application
Dim rpt As CRPEAuto.Report
```

15. In the Form1_Load event instantiate an instance of the CDO rowset with the following code

```
Private Sub Form_Load()
    Set rsCDO = New CrystalDataObject.CrystalComObject
End Sub
```

16. Save the Project to the same folder as the report that was created previously.

17. From the menu bar select Project | Components and check Microsoft Flex Grid Control 5.0 and OK. This will add the Flex Grid Control to the toolbox.

18. Add the Flex Grid control to the Form1 and make the following changes to the properties, you may also need to adjust Form1 form size. Set the following properties of Form1 accordingly.

```
Top = 0
Left = 0
Width = 7455
Height = 2175
```

19. Add a Data control to the form. Select Properties of the DataControl and set the Database property the Xtreme.mdb sample database, which is located in the Seagate Crystal Report directory, and set the Orders table as the Recordsource. Set the Visible property to False

20. Select the Properties of the MSFlexGrid control and set the datasource property to the DataControl (Data1)

21. Add four label controls (do not add as a control array) and four textboxes (do not add as a control array). These controls will be used to display the data from from the CDO rowset. The labels will contain the CDO field headings and the text boxes will contain the values. Pair the controls together (i.e. label1 with textbox1) and place the pairings underneath each other and the Flexgrid on the Form. (refer to the picture at the end of this tutorial for the end result)

22. In the General Declaration section of Form1 create the following Procedure. This procedure populates the textbox control with data from the current record in the CDO rowset. For this tutorial the first four fields in the rowset are used to populate these controls.

```
Public Sub mGetFieldData()

    Text1.Text = rsCDO.GetFieldData(0)
    Text2.Text = rsCDO.GetFieldData(1)
    Text3.Text = rsCDO.GetFieldData(2)
    Text4.Text = rsCDO.GetFieldData(3)

End Sub
```

23. Add four command buttons as a Control array to the form and set the captions of the buttons as follows:

```
Index 0 = |<
```

```
Index 1 = <<
Index 2 = >>
Index 3 = >|
```

24. In the Command1_Click event enter the following code which is used to navigate through the records in the rowset.

```
Private Sub Command1_Click(Index As Integer)

    Const FIRST_REC = 0
    Const PREV_REC = 1
    Const NEXT_REC = 2
    Const LAST_REC = 3
        Select Case Index
            Case FIRST_REC
                rsCDO.MoveFirst
                 curRecord = 1
            Case PREV_REC
                If curRecord = 1 Then
                    rsCDO.MoveFirst
                Else
                     curRecord = curRecord - 1
                    rsCDO.MoveTo curRecord
                End If
            Case NEXT_REC
                If curRecord = rsCDO.RowCount Then
                   Exit Sub
                Else
                    curRecord = curRecord + 1
                    rsCDO.MoveTo curRecord
                End If
            Case LAST_REC
                rsCDO.MoveTo rsCDO.RowCount
                 curRecord = rsCDO.RowCount
            Case Else
                Exit Sub
        End Select
        mGetFieldData
    End Sub
```

25. Add another command button and change it caption to “Build Rowset”

26. In Command2_Click event add the following code. This code builds the CDO row data using the data from the flexgrid

```
Private Sub Command2_Click()

    Dim vbArray() As Variant
    Dim i as Integer
       Dim icol as Integer
       Dim irow as Integer

   'empty rowset in case it contain field/data
    rsCDO.Reset
    'set grid to first row which is field names.
    MSFlexGrid1.Row = 0
    'adds fields to the CDO dataset and assigns the
    'field name from grid column.
    For i = 1 To MSFlexGrid1.Cols - 1
        MSFlexGrid1.Col = i
        rsCDO.AddField MSFlexGrid1.Text
    Next i

'Dimension array.  2 is subtracted from the count
property
'because the grid is 0-based and the first row and col
in
'grid is not used
    ReDim vbArray(MSFlexGrid1.Rows - 2,
(MSFlexGrid1.Cols - 2))

    'Populate array with data from grid
    For iCol = 1 To (MSFlexGrid1.Cols - 1)
        MSFlexGrid1.Col = iCol 'set column
        For iRow = 1 To MSFlexGrid1.Rows - 1
            MSFlexGrid1.Row = iRow 'set row
            vbArray(iRow - 1, iCol - 1) =
          MSFlexGrid1.Text
        Next iRow
    Next iCol

    'data is passed to the CDO rowset
    rsCDO.AddRows vbArray
```

```
    'gets field name of the first four fields
    'in the rowset and set label captions
    Label1.Caption = rsCDO.GetFieldName(0)
    Label2.Caption = rsCDO.GetFieldName(1)
    Label3.Caption = rsCDO.GetFieldName(2)
    Label4.Caption = rsCDO.GetFieldName(3)

    'move to the first record in rowset
    rsCDO.MoveFirst
    'populate textboxes with first record data
    mGetFieldData
    'set rec counter to first record
    curRecord = 1
End Sub
```

27. Add another command button and change its caption to “RESET”

28. In Command3_Click event add the following code. This code removes all data and fields from the rowset.

```
Private Sub Command3_Click()
        'empties all data and fields from rowset
    rsCDO.Reset
    mGetFieldData

End Sub
```

29. Add one final command button and change it caption to “Preview”

30. In Command4_Click event add the following code. This code is used to preview the report.

```
Private Sub Command4_Click()

    Set appl = New CRPEAuto.Application
    Set rpt = appl.OpenReport(App.Path + "\report1.rpt")
    rpt.Database.Tables(1).SetPrivateData 3, rsCDO
    rpt.Preview

End Sub
```

31. You should now have a Form which looks similar to the following. If so then run the Project. You should be able to dynamically build a rowset, navigate the records, reset the rowset object and preview a report.