Debugging Techniques- Error handling:

Testing Program

Once coding is completed, we compile our program into machine code. Compilation means converting source code into object code compilation result may be in certain kind of error which are indicated by the computer after the process is over. This error has to be corrected and then the program recompiled. The process continue until the program is error free the three types of errors that are generally occur or encountered are,

1. Syntax error
2. Execution error
3. Logical error

1) Syntax Error

Syntax error occurs when a computer language compiler can not understand a command entered by the programmer. When such an error is occur, the computer rejects the program and print out an error message. These types are error are occurring, during the compilation error and are very easy to correct in may case. These errors arise due to spelling mistakes, missing comma etc. and by incorrect syntax also. Syntax error occurs when write wrong code, for incorrect spelled variables, keywords etc., missing punctuation and for incorrect number of parameters for functions. A message gets displayed that informs about the syntax error when constructing the code or when are executing the application. That displays a message when a syntax error occurred.

2) Execution Error

This error occurs after error free compilation at the time of execution of the program. The execution error is also called runtime error. For example, infinite row, correct output only for selected data, data incorrect or in wrong order. When the user attempts to do an operation that is not possible a run time error occurs. For example you have written code that will access a file and you have not specified the path correctly or that file does not exists in the specified path then run time error occurs.

3) Logical Error

Logical error is due to mistake by programmer while coding the program. The computer does not detect logical error. To correct logical error, we have do go to algorithm of the program. The written a code and in execution are not getting the expected output. That means logical error exists in the code. To trace the error, have to debug the code of the application. Visual Basics provides with different options that help to debug the error.

Debugging in Visual Basic

Visual Basic provide programmer with a set of tools that is hard to build. We can debug our program interactively working through our code line as the program run. This powerful technique work behind the sense in a way that is in valuable to finding our what's going wrong we can also specify where to beginning debugging a program with break point which are lines of code that you tag to make the program halt and debugging process start.

1) Debugging Break Point

Break points are the foundation of Visual Basic debugging. When we set break point in a program and run that program. Program execution continues until one of the break points is in counter and program execution stop, making Visual Basic entered the debug start. We place break point in code by moving the text insertion create to that line and either selecting toggle break point in the debug menu.

2) Debugging Watch Window

In debugging watch window is used to display the value of selecting variables or impression in that window continuously as we execute your program line by line. To add watch window, select variable of expression that you want to watch with the mouse and click add watch item in the debug menu. This open the add watch dialog box.

3) Quick Watch

To examine the value in a variable or expression in the code window when stop at a break point. Select the variable expression we want to examine with the mouse and select quick watch option in the debug menu.

4) Edit Watch

It is used to display value of particular variable or expression when debugging process start, we can see value of variable or expression more than one. After adding expression in watch window, we can correct or make changes as per our requirement.

5) **Step Into:** It execute the next executable line of the code in the application and steps into procedure.

6) **Step Over:** It executes the next executable line of code in the application without stepping into procedure.

7) **Step Out:** It executes the remainder of the current procedure and breaks at the next line in the calling procedure.

8) **Local Window:** It displays the current value of local variable.

9) Immediate Window: It allows you to execute the code or query values while the applications in break mode.

10) Call Stack: While the break mode, present a dialog box that shows procedure that have been called but not yet run to completion.

11) Add Watch: In add watch window we can give expression for calculating or evaluating the result. That means it is used to add expression into watch window.

Tools for Debugging:

- Now to debug the application have to invoke the debug toolbar, so that can access its various options.
- Right click on the menubar or on the standard toolbar and click on Debug option from the popup menu.

The debug toolbar gets invoked.

Another way to invoke the toolbar is

Click on View menu →Toolbars →Debug.

- **Start-** Start button is used to execute the application from its Startup object.
- **Break:** Break buttons halts the execution of the application temporily and enter the break mode.
- **End-** End button ends the execution of the application and returns to the design mode.
- **Breakpoint:** - Breakpoint is used to set or remove the breakpoint. It can set breakpoint on any line of the code and Visual basic suspends execution on that specified line of code. To set the breakpoint on any line of the code in the code editor, select it or place the cursor in the beginning of the line.

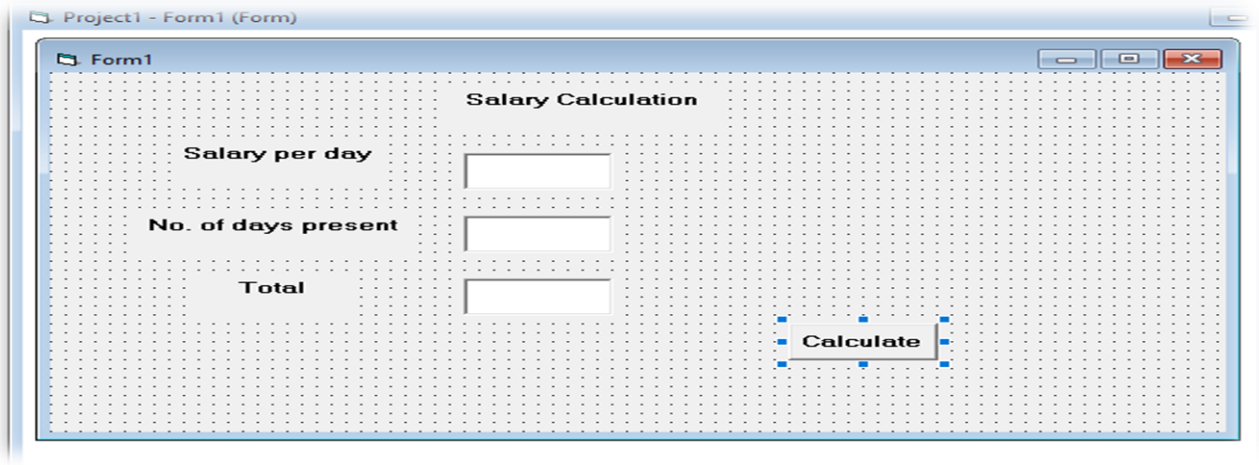
  You can also enter into break mode by,
- Pressing Ctrl+Break key
- Selecting Break option from the Run menu.
- Clicking on Break option from the Standard toolbar.

Trapping Error

To trap the error have to use the On Error statement. To set or enable an error trap in the procedure or function where expect a run time error might occur. The syntax of On Error statement is,

On Error GoTo Error-Handler

To specify the name of the error handler using the Error-handler argument to which the execution must branch off. In case the error occurs the execution is transferred to the error handler ErrHandle.



Code :

```
Dim a As Integer
Dim b As Integer
Private Sub Command1_Click()
On Error GoTo Errhandle
a = Text1.Text
b = Text2.Text
Text3.Text = a * b
Exit Sub
Errhandle:
ans = MsgBox("You have to enter integer value", vbOKOnly, "Error")
End Sub
```

Handling runtime Errors:

In the On Error statement the name of the handler is specified and according Error handling routine is identified. So when the error occurs the On Error statement redirects execution of the program into procedure error handler. The code of the previous example where the On Error statement redirects the execution to the ErrHandle i.e. the error handler of that procedure.

Handler :< error handling Code>

When are creating error handler in the program, try to trap all the possible error that might occur. Moreover, it can call other procedure also. So create a single error handler for similar type of procedure. This action will save time and make code easier to maintain.

Error handling code frequently uses the Err object to display information about the error occurred in the program.

Use of Exit Sub

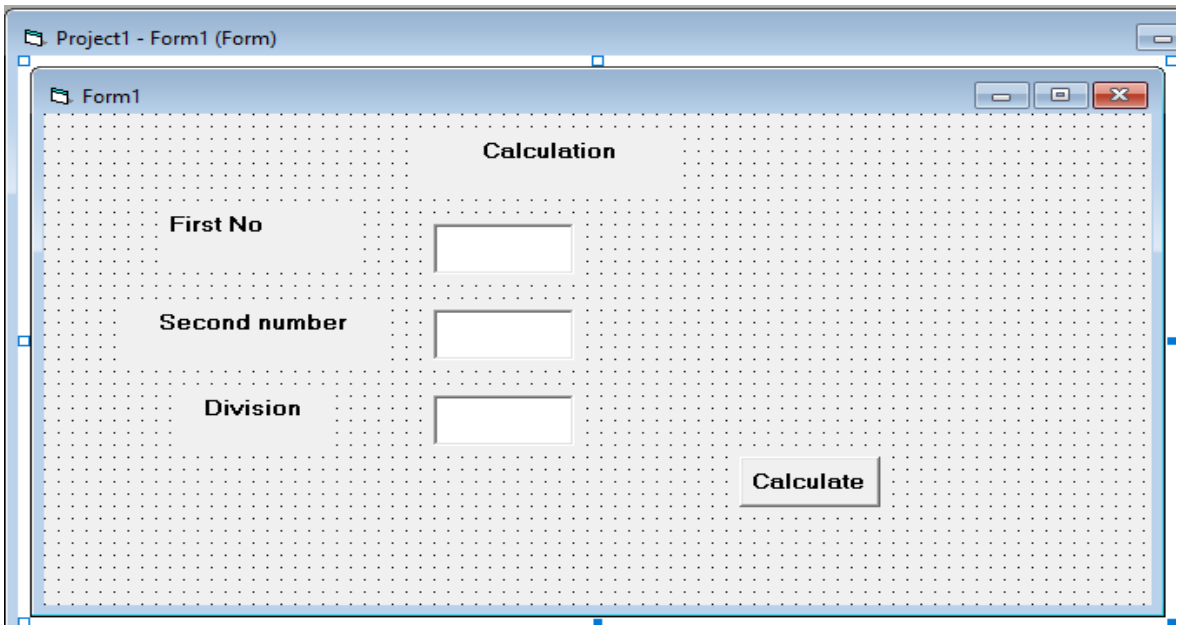
When writing the code for the error handler, need to write Exit Sub or Exit Function statement before the error handler label. In case don't include the exit, sub statement error handler will be executed even though no run time error has occurred in the program.

Resuming program execution:

This can be done with the Resume or Resume Next statement. It should continue from the same position where the branch off occurred to handle the error. This is possible when the execution control exists from the error handler and for that reason Resume or Resume next statement is used and it returns to the statement from where the error occurred.

Resume:

The Resume statement when want to execute the same statement from where the error had occurred. So after correcting the condition for which the error occurred. It can specify the line of code at which the execution should resume after the error handler executes. To do so have to use the Resume line number statement.

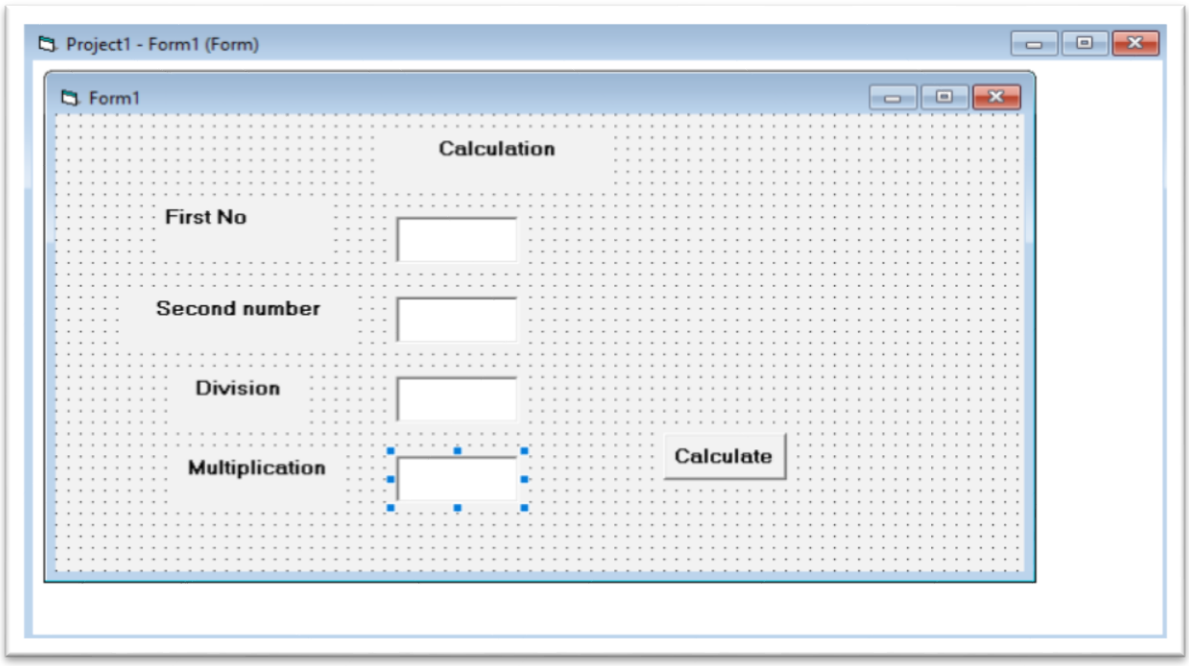



```
Private Sub Command1_Click()
On Error GoTo errdiv
Text3.Text = Val(Text1.Text) / Val(Text2.Text)
Exit Sub
errdiv:
Dim ans As Integer
ans = MsgBox("You have to enter integer value", vbOKOnly, "Error")
If ans = vbYes Then
Dim num As Variant
num = InputBox("Enter a number ", "Number")
```

```
Text2.Text = CInt(num)
Resume
Else
If ans = vbNo Then
End
End If
End If
End Sub
```

Resume Next: -

It can use the Resume next statement when want to execute the immediate statement that follows after the statement in which the error had occurred. When want to repeat the execution of the same statement.

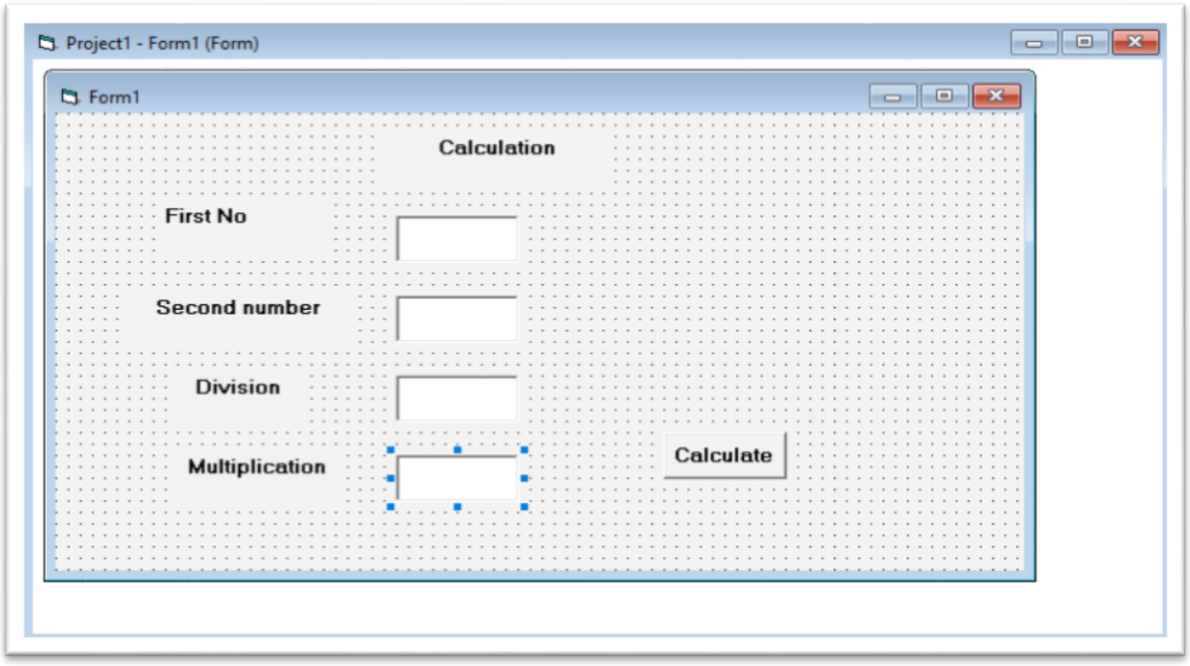



```
Private Sub Command1_Click()
On Error GoTo errdiv
Text3.Text = Val(Text1.Text) / Val(Text2.Text)
Text4.Text = Val(Text1.Text) * Val(Text2.Text)
Exit Sub
errdiv:
Dim ans As Integer
ans = MsgBox("You have to enter integer value", vbOKOnly, "Error")
If ans = vbYes Then
Dim num As Variant
num = InputBox("Enter a number ", "Number")
Text2.Text = CInt(num)
Resume
Else
If ans = vbNo Then
Dim num1 As Variant
num1 = InputBox("Enter a number", "Number")
Text2.Text = CInt(num1)
```

```
Resume Next
End If
End If
End Sub
```

Using Resume Label- Beside these two statements can send the flow of execution into an error handler using a label and from the error handler can return the flow back into the procedure using the label.this is done with resume Label statement. Actually, label works as an identifier in the Resume label statement and it executes when the condition is satisfied.

Resume Label: when this statement executes the control flows to the statement that is identified by the Label.




```
Private Sub Command1_Click()
On Error GoTo errdiv
Calculation:
Text3.Text = Val(Text1.Text) / Val(Text2.Text)
Text4.Text = Val(Text1.Text) * Val(Text2.Text)
Exit Sub
errdiv:
Dim ans As Integer
ans = MsgBox("You have to enter integer value", vbOKOnly, "Error")
If ans = vbYes Then
Dim num As Variant
num = InputBox("Enter a number ", "Number")
Text2.Text = CInt(num)
Resume calculation
Else
If ans = vbNo Then
```

```
Dim num1 As Variant
num1 = InputBox("Enter a number", "Number")
Text2.Text = CInt(num1)
Resume Next
End If
End If
End Sub
```

Error Object

Error object contains information about run time error, when an error occur Visual Basic set the various property of the error object Error such as an error number, a description in an error handling routine, so that your application can respond intelligently to an error situation. There might be any reason of run-time error like Type mismatch, File not available, Disk not ready etc. So, to know the exact reason Err object is used. Moreover, it is used to identify the source of error. This enables to display contex sensitive messages in the error handler. The err object is a special object designed to deal with errors and it contains information about the most recently error occurred. It has many properties and methods.

Error Object Properties

1. **Number:** It contains the number associated with the run time error that occurs.
2. **Description:** It contains the short description of the error.
3. **Source:** It contains the name of the object application that generated the error.
4. **Raise:** Raise method use to generate user define error in this we can give error number, error description and error source.

```
Private Sub Command1_Click()
On Error GoTo errdiv
Calculation:
Text3.Text = Val(Text1.Text) / Val(Text2.Text)
Text4.Text = Val(Text1.Text) * Val(Text2.Text)
Exit Sub
errdiv:
Dim ans As Integer
ans = MsgBox(" The error is” &Err. Number &Err.Description, , vbYesNoCancel, "Error")
If ans = vbYes Then
Dim num As Variant
num = InputBox("Enter a number ", "Number")
Text2.Text = CInt(num)
Resume calculation
Else
If ans = vbNo Then
Dim num1 As Variant
num1 = InputBox("Enter a number", "Number")
Text2.Text = CInt(num1)
Resume Next
End If
End If
End Sub
```

Error object's

Raise method

It is also called user defined errors. The user can raised his own errors using the raise method of error object. The argument passed to the raised method are error number, error source and the error description.

Clear:

To reset the properties of the Err object, Clear method is used. Actually, this method is called automatically in many situations like-Any Resume statement, when the flow of execution exits from the procedure that had generated the error, when on error statement is used.

Working with Data Environment

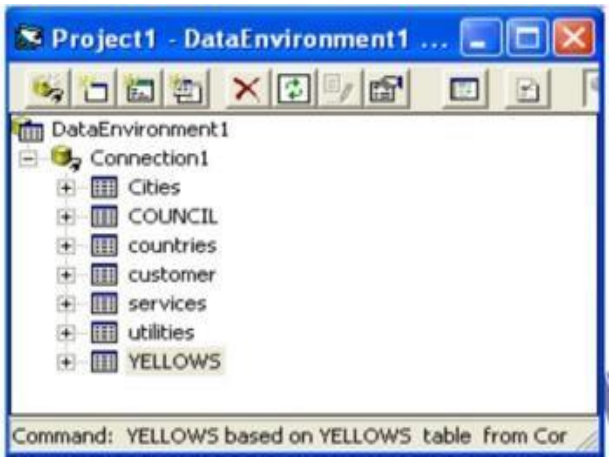
Data Environment

Data environment designer is an interactive, powerful interface introduce in Visual Basic 6.0. The data environment object can be used, much like a data control by binding other controls such as text box, label, check box, to the designer. This designer can be controlled at designing time by changing its properties or programmatically at run time. It provides an interactive, design time environment for creating programmed runtime access to data. Data environment designer can be included by following steps.

- Add data Environment designer through project menu.
- Then select references option.
- From the references dialog box select data environment 1.0 and then click on „Ok" button.

Then we can accomplish following task with data environment designer.

1. Create connection object.
2. Create command object based on stored procedure, tables, SQL statement.
3. Create hierarchies of command object-based grouping of command object.



Connection Object

At design time we can use the data environment designer to create data environment object. To access data using data environment we must create a connection object. Therefore, every data environment must have at least one connection. A connection object represent connection to a remote database that is used as a data source after including data environment in project it automatically create a new connection called connection1.

Creating Connection object

Click Add connection on the data environment designer tool box or right clicks your data Environment designer and select add connection through short cur menu. After adding connection data environment is updated.

Use the following procedure to specify connection object

Property To set connection name and data source.

1. Click right mouse button on Connection object and choose property to access the data link property dialog box.
2. In the provider tab, select the driver between your application and the database.
3. Now select connection tab.
4. Click the button with the ellipsis and locate the database
5. Click the test connection button to make sure the connection works.
6. If the database requires a username and password, supply the same information in the appropriate boxes on the connection tab.
7. Click Ok to return to the data environment window.

The connection object has four tabs.

a) Provider **:** Used to select the OLEDB provider required.

b) Connection: Used to select the database, provide the user id and password appropriate to the requirements of the user

c) Advanced : Used to set timeout parameter, locking level etc.

d) All : Used to view a summery of the connection object‟s setting.

Command Object

Command object define specific detail information about what data is retrieve from a database connection data object can be based on either a database object such as Table, View, Store procedure or a SQL query we can create relationship between command object to retrieve a set of related data.

Create Command Object

To add command function is a available at all time and is independent of the existence of other objects. If a connection object can be identify from the current focus during the data access process the active connection property of the command object is set to that connection object. To add command object click on add command tool button in the data environment designer tool box or right click on connection object then select add command option from short cut menu.

To specify command object property

1. Right click the command item, and from the shortcut menu select Add command. The command1 object will be added under the commands folder.
2. Right click the command object and select properties to open the command1 properties window.
3. In the general tab set the command object‟s name and its connection to connection 1.
4. In the database object box select table and in the object name box, you will see the names of all the table in the database.
5. Select appropriate table name and click apply.
6. Click Ok to return the data environment.
7. Click the plus sign in front of the command1 object to display the table’s field.

The command object has six tabs

a) General : To rename the object or change the connection object.

b) Parameter : To display all the parameter associated with.

c) Relation : Allows the developer to associate command object with each other in a hierarchical relationship

d) Grouping : Allows to specify one or more columns on which the result is to be grouped

e) Aggregates : To Aggregate the data based on one on or more of several function types such as sum and standard deviation.

f) Advanced : It has different options available depending on the data source.

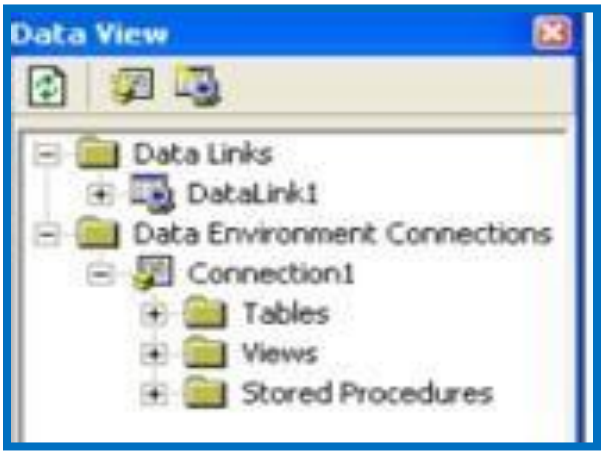
Data View Window

The data view window present a view on to one or more databases connection providing access to the entire structure of the database on a particular connection. The data view window provides the mean to use the Microsoft Visual Basic tools to visually manipulate that structure of a database.

Data view provides a graphical means of organizing our database object including database diagram, Table and Store procedure. We can open, create and Edit our database object.

To open data view window click data view option from view menu or click the data view button on the standard tool bar. The data view window displays editable data from any database to which we have made a data connection. It displays the database object for that database under the database connection „Node" perform following steps to create data link in data view.

1. Select data link in data view window and click right mouse button and click on Add a data link property window.
2. Select a database type which you want to connect and click next to view connection property.
3. Set different option as per your requirement and click on „Ok" command button. Then you get table, views etc under data link1 folder.
4. To display the data select table in data view window and click right mouse button.



Data Report

Data report utility, introduced for the first time in Visual Basic 6.0 is used to design reports. Data report designer is a flexible report generator that can create banded hierarchical reports. In addition to creating printable reports, the report can be exported to HTML or text files. This utility uses a graphic design environment, which lets the developer easily design and generate database reports. The fields can be dragged from the data environment designer to the data report designer.

Data Report Designer Feature

1) Drag and Drop

Functionality for field drag field from the Microsoft data environment designer to the data report designer. When we do this Visual Basic automatically create a text box control on the data report and set the data member and data field property of the drop field.

2) Tool box control

The data report designer is own set of control when a data report designer is added to a project the control automatically creates on a new toolbox tab name Data Report.

3) Print preview

Print preview is used to preview report using show method. It display report in separate window.

4) Print Report

It prints a report programmatically by calling the print report method, when one data report in preview mode user can print report.

5) Exporting Report

The report can be export in the form of HTML or text file when the data report in preview mode, just click on export button it display the other window which help to export the report.

Part of the Data Report

The data report designer consists of following objects.

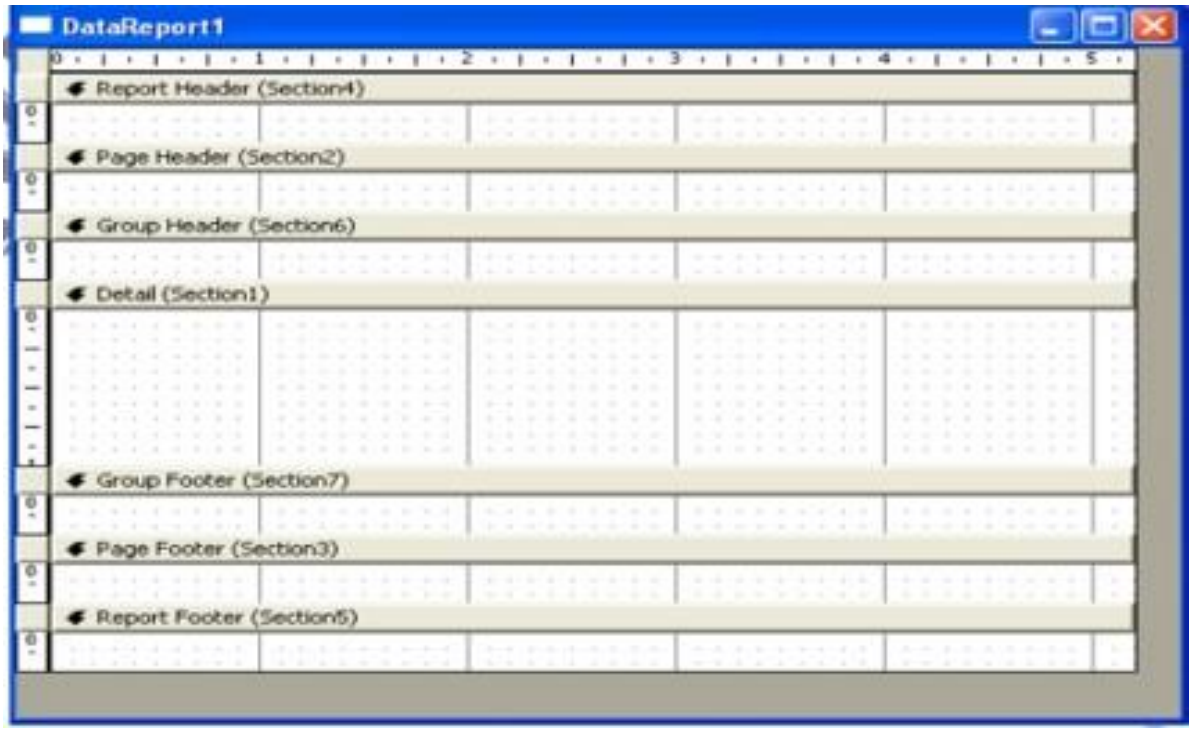
A) Data Report object

Similar to Visual Basic form the data report object has both the visual designer to create the layout of report and code module to programmatically format controls.

B) Section Object

Each section object of the data report designer is represented b a section object in a section collection. At design time each section is represented by a header. The default data report designer consist of following section.

1. Report header
2. Page header
3. Group header / Footer
4. Detail
5. Page Footer
6. Report Footer



1) Report Header

Contain the text that appear at the very beginning of a report Such as the report Title, Author or database name. If you want the report header to be the first page in the report, set its source property to rptpagebreakafter.

2) Page Header

It contains information that goes at the top of every page such as report Title.

3) Group Header / Footer

It contains a repetitive section of the data report. Each group header is match with the group footer. The header and footer pair is associated with a single command object in the data environment designer.

4) Details

It contains the inner most part of the report the detail section is associated with the lowest level command object in a data environment hierarchy.

5) Page Footer

Page footer consist the information that goes at the bottom of every page such as the page number.

6) Report Footer

It consists the text that appear at the very end of the report such as summery information and address and contact number. The last page header and page footer.

C) Data Report Control

Special controls that only work on the data Report designer are included with it. these controls are found in the Visual Basic toolbox but they are placed on separate tab name "DataReport"

Data Report tool box control

1) Textbox control (rpttextbox)

It allows us to format text or assign data format of a database.

2) Label control (rptlabel)

It allows us to place label on the report to identify field or section. We can format label as per our Requirement.

3) Image Control (rptimage)

It enable us to place graphics on report. This control cannot be bound to data field.

4) Line control (rptline)

It use to draw line or rules on the report to further distinguish section.

5) Shape control (rptshape)

It uses to draw rectangle, square, triangle and circle on a report.

6) Function (rptfunction)

It is special type of text box that calculate values as a report is generated.

Method of the data report object

1) Show method

The data report is displayed using the show method in the same way as a form is displayed. The following code displays the data Report1.

Datareprot1.Show vbmodel

2) Print Report Method

dataReport1.PrintReport ([Show Dialog], [Range], [Page Form], [Page To])

show dialog is a Boolean variable, which when set to true. causes the print dialog to be displayed. Range can taken the following values.

rdPrintAllPages= > Prints entire report

rdRangeFromTo = > Specifies that only part of the report will be printed.

3) Export Report method

It allows us to save the report as a file

Creating Report in Visual Basic

In Visual Basic 6.0 it has included tools called data report designer to build the report. In data report designer data can be retrieved through data environment designer. To create report perform following steps

1. Start new project
2. Select data project option from the new project dialog box.
3. In project explorer window, double click on the option data environment.
4. Select and right click on connection 1.
5. Select property option.
6. Select the connection tab.
7. Select and click on the option use connection string.
8. Click on the „Build‟ button.
9. Select machine data source tab.
10. Select Microsoft Access Database option.
11. Click on „Ok‟ command button.
12. Click on test connection button.
13. If the connection is “Ok‟ it displays message box.
14. Click on “Ok‟ button.
15. Click on the “Add‟ command tool from the tool bar.
16. The command1 will get added to the window
17. Select and right click on command1.
18. Select property option and set connection1 in the toolbox.
19. Set source of data as table object and give table name.
20. Double click on data report1 in the project explorer window.

Printing the Report

The easiest way to print a report is to let the user start the operation interactively by clicking on the left most button in the DataReport preview window. Users can pick a printer from the list of installed ones and select a page range and the number of copies they want. They can even print to a file so that

they can do the actual printing later. When we enable interactive printing, all we need to do is to display the Data Report Window, which we can do by using the show method or by designating the Data report designer as the start up object of the current project. We can use several properties to modify the default appearance of the preview window.

Eg. To display a data report in modal maximized window.

DataReport1.WindowState = vbMaximized

DataReprot1.Show vbModal

To start printing process through code we need the print report method of the Data Report designer, which accepts several arguments and returns a long value.

Cookie = Print Report([Show Dialog], [Range], [Page from], [Page to])