WINDOWS AND VISUAL BASICS
CHAPTER ONE

1. Fundamentals of Visual Basic

Introduction:

A basic is the fastest and easiest way to create applications for Microsoft Windows. Visual Basic provides complete set of tools to simplify rapid application development both for the experienced professional and new window programmers.

In the name Visual basic- the "Visual" part refers to the method used to create the graphical user interface (GUI). Unlike many languages which requires numerous lines of coding to describe the appearance and location of interface elements, Visual Basic provides pre-built provides per-built objects that can be used to from the Graphical User interface (GUI).

The "Basic" Part refers to the BASIC language as its basic syntax of statements is retained by Visual Basic. But visual basic not contains several hundred statements, functions, and keywords, many of which relate directly to the windows GUI.

The Visual Basic programming language is not unique to Visual Basic. The Visual Basic programming system, applications edition included in Microsoft excel. Microsoft access and many other window applications uses the same language. The Visual Basic scripting Edition (VBScript) is a widely used scripting language and a subject of the Visual Basic language. So mastering Visual Basic also helps to master these other areas.

New features of Visual Basic 6.0

Whenever a product's version number increases by one, it means that several enhancements have been made over the previous version. Before looking at the completely new additions to Visual Basic 6.0, this section presents general Visual Basic features briefly.

General Features

The compiler is Visual Basic gives many different options for optimizing the compiled code, such as Optimization for fat code, optimization for small code and favor of Pentium pro etc.

The Visual Basic is a very open environment that supports the Client/Server architecture, ActiveX, Component Object Model (COM), Distributed Component Object Model (DCOM). It also supports Open Database Connectivity (ODBC).

Intelli Sense

A new feature that Microsoft calls intelliSense enables the system to react in real time during coding. There are five basic intelliSense features:

- Quick Info
- Complete word
- Data tips
- List members
- List constants

Quick info is a feature that presents the syntax of the procedure, which is being typed, in a tool tip like window. It even goes one step further in that it will bold the specific parameter which is being coded in real time.

Complete word is a feature that automatically completes the word which is typed in real time, after enough characters that make the word distinct within the list of available words, has been typed.

Data tips is a very useful feature that simply shows the value of the variable in a yellow too tip format at run time.

List members is a feature that is used to simply list all of the properties or methods available constant values for a given property are listed.

Other features

Multiple projects can be loaded into the IDE – integrated Development Environment (Explained shortly) and treated as one. For example, a standard project and an active X project can be loaded at the same time. This saves tremendous amount of time for coding and debugging. Here's a list of some other enhancements:

- ❖ Break points will be toggled simply with a mouse click,
- ❖ Bookmarks can placed for quick location.
- ❖ Properties in the property window can be manipulated alphabetically or even by category.
- ❖ A nice new feature is block comment and uncomment. With the click of a mouse button, all highlighted code will be commented or uncommented
- ❖ Aligning and formatting can easily be applied to one or more controls.

Visual Basic 6.0 features

This section briefs what are newly added in Visual basic 6.0 in various categories.

ADO (ActiveX Data Objects)
Data Report

It allows to use drag and drop to quickly create reports from any recordset, including hierarchical recordset.

Hierarchical Flex Grid control

It is an updated version of the Flex grid control that, in additing to supporting all the functionality of the Flex grid control, can display a hierarchy of ADO recordset. Each recordset returned is displayed as a separate band within the grid and can be formatted independently.

Data List Control, Data Combo Controls

These controls are OLE DB version of the DBList and DB combo controls. They also support the new ADO Data control.

New in Internet Features
IIS Applications

This feature enables to write server side internet applications that use Visual Basic code to respond to user requests from a browser.

DHTML Applications

DHTML Application allows to write Visual Basic code to respond to actions on an HTML page, without transferring processing to the sever.

Getting in to Visual Basic
To start Visual Basic

- Click the start button in windows Taskbar, choose Microsoft Visual Basic 6.0 group and click the Microsoft Visual Basic 6.0 item.

Simply double click the Microsoft Visual Basic 6.0 icon on the Desktop.

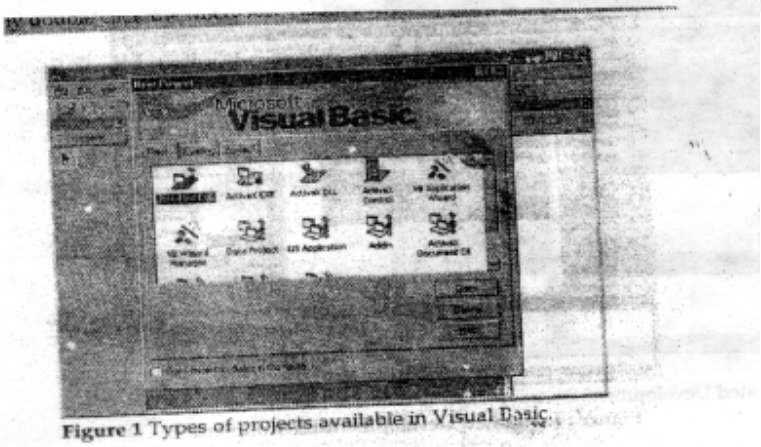


Figure 1 Types of projects available in Visual Basic

Standard EXE it is a typical project type and used to create standard executable files.

ActiveX EXE, ActiveX DLL these types of projects are used to create ActiveX code. Components which are OLE automation serves. These two types of projects are identical I functionality, but are packaged differently (as executable files or Dynamic Link libraries)

ActiveX Control This type of projects used to create new ActiveX controls.

Visual Basic Application wizard It generates a new, fully functional application from which more complex application can be built.

Data project It creates a project that automatically includes data report and data environment...

Active X Document EXE, Active X document DLL These project types are used to active X document. Which are in essence Visual Basic applications that can run in the web browser such as Internet Explorer.

IIS Application, DHTML application. These projects are used to create internet applications for both server-side and client side.

The new project dialog box as figure has three tabs;

- New Existing Recent

The new tab allows to select the types of a new project, as explained already. Switching to the existing tab will enable to select an existing project and open it. Switching to the existing tab will enable to select an existing project and open it. Switching to the recent will enable to select and open the most recently worked projects during the last few days.

Select the standard EXE icon in the New project window, and click the OK button. This will open the integrated Development Environment (IDE) as in figure.

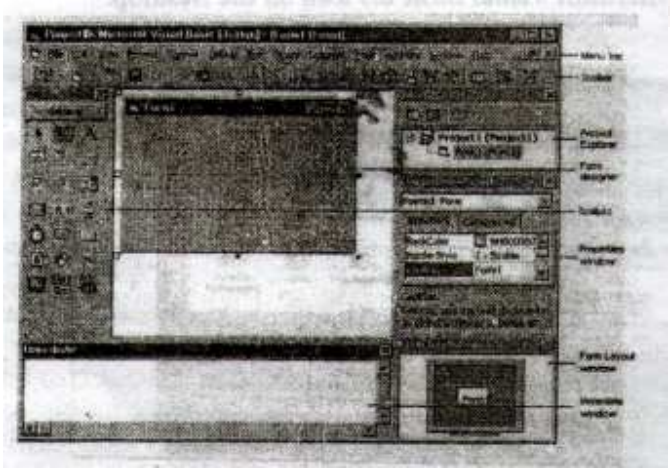


Figure: Integrated Development Environment

The following sections briefly explain the parts of the IDE.

The menu bar

The menu bar contains the commands needed to work with the Visual Basic. The Basic menus are:

- **File:** Contains the commands for opening and saving projects and creating executable files and list of recent projects.
- **Edit:** Contains editing commands (Undo, copy, paste, and so on) plus a number of commands for formatting and editing code (find replace)
- **View:** Contains commands for showings or hiding components of the IDE.
- **Project:** Contains commands that add components to the current project reference to windows objects, and new tools to the Toolbox.
- **Format:** Contains commands for aligning the control on the form.
- **Debug:** Contain usual debugging commands.
- **Run:** Contain the commands that start, break, and end execution of the current application.
- **Tools:** Contains tools needed in building active X components and ActiveX controls;
- Contains the command to start the menu editor and the options command, which lets to customize the environment.
- **Add-Ins:** Contains add- ins that can be added and removed as needed. By default, only the Visual data manager Add in is installed in this menu. Add n manager command
- **Window:** Contains the commands to arrange windows on the screen; the standard window menu of a windows application.
- **Help:** Contains information to help user as her works.

The tool bars

Provided quick access to commonly used commands in the programming environment. By default, the standard toolbar is displayed when Visual Basic starts. Additional toolbars for

editing, form design, and debugging can be toggled on or off from the tool bars command on the view menu. For example, to view the debug toolbar, choose view toolbar and check the debug option. Like wise to hide the debug follow the same procedure, but this time uncheck the debug option.

Toolbar can be docked beneath the menu bar or can 'float' if the vertical bar on the left edge is selected and dragged away from the menu bar. Each toolbar and its brief description are given below.

◊ Standard toolbar is just below the menu bar and is displayed by default.

- The Edit toolbar contains the commands of the Edit menu.

- The Debug toolbar contains the commands of the Debug menu.

- The Form Editor menu contains the commands of the Format menu.

The Project Explorer

Lists the forms and modules incurrent project. A project is the collation of files that used to build an application.

The project components are organized in folders, and the project window is called the application's user interface. By default, the toolbox contains the pointer icon and the icons of 20 intrinsis controls.

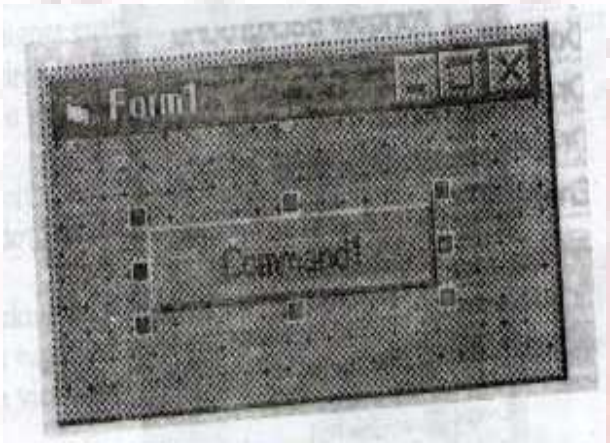
To place a control, say command button, on the form

- ◊ Select the command button with the mouse.
- ◊ Move the mouse over the form. When the mouse pointer is over the form it will turn to cross.
- ◊ Draw the control on the form, just as drawing a rectangle.

The Project Explorer

The properties window contains the property setting for the selected control. Properties are attributes of an object, such as its size, font, background color and so on. By setting these properties the appearance of the selected control can be adjusted easily. For example to change the caption and appearance of a command button follow the steps given below.

- ➢ Select the command button icon on the tool bar and draw the button on the form. The button will appear with its default name as its caption, as in



- ➢ Make sure that the command button is selected and press F4 or click the properties window to make it as active.

- ➢ In the properties window (Figure 2.4) locate the caption property and changes its value as Click & me

- Placing ampersand before the letter M will make the caption of the command button to appear as click me. It meaning is that this button can pressed by using keyboard characters Alt and M (Alt + M).

- To change the background color of the command button, locate the back color property and click the down arrow button next to the current value of the color. This will display the color selection box as in figure 2.5
- Select the Inactive Title Bar color. Now the command button will appear as in figure 2.6

Note: when the color selection box is displayed, its palette tab might be active by default. This tab will show the colors alone. To view each color's description switch to the system tab.

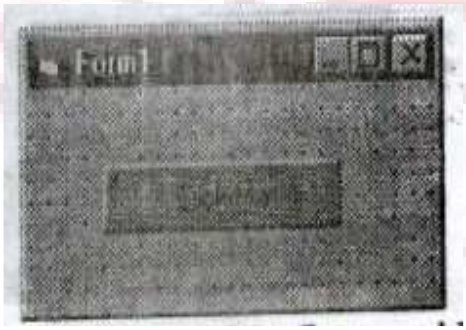


Figure: New appearance of the Command Button

The Form Designer

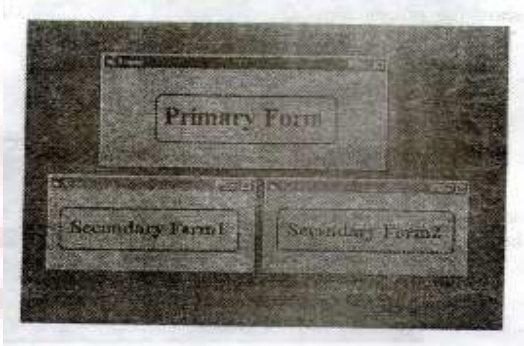
The form designer enables to add controls, graphics, and pictures to a form to define the interfaces in the desired look. Each form in an application has it own form designer. The form designer displays two windows for each form.

- The form itself (the enables of the user interface)
- A code window (the code behind the elements of the form)

Double clicking on a form will bring its code window front. Double clicking on a from name in the project explorer window will bring the form to the front. Or to switch between these window easily, the view code and view form at he top of the project explorer window (Figure 2.7) an be used.

The Form Layout

The form layout window helps to position the forms of an application using a small graphical representation of the screen. This window is very useful in the application that uses multiple forms, because each form's position can easily be specified with respect to the main form. For example, assume that an application contains three forms. The form layout windows will show the layouts of the all three forms. Mouse can be used to drag these layouts to the desired location as in figure.



When the application shown all the forms, they will appears in the screen as in figure.

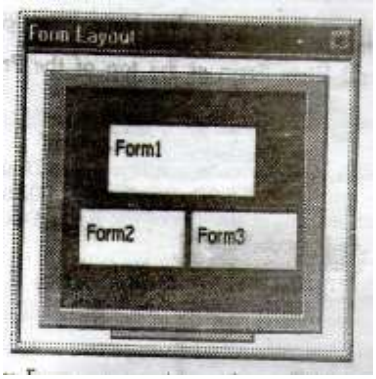


Figure: Forms – appearing on the Desktop as positioned in he Form Layout window

The Immediate Window

The immediate window is a debugging aid. While an application is running it can be paused and the immediate window can be used to examine or change the values of the application's variable. It can be used to execute simple basic commands even when the application is not running. For example, type print 30+40 in the immediate window and hit enter. The result 70 will appear in the next line.

Context Menus

It contains shortcuts to frequently performed action. When right mouse button click, context menu pops up with list of shortcuts specific to part of the environment where the mouse is right click, for example context menu displays items name components...., Add tab…., dock able, and hide, when the mouse is right clicked on the toolbox (figure 2.10)

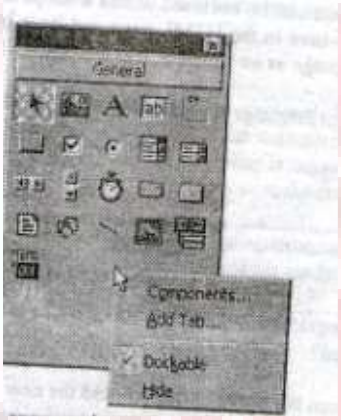


Figure: Context menu that appears when the right mouse Button is clicked on the Toolbox.

Basic Scripting In Visual Basic

Microsoft Visual Basic scripting edition, the newest member of the Visual Basic family of programming languages, bring active scripting to a wide variety of environments, including web client scripting in Microsoft internet explorer and web server scripting in Microsoft internet information server. No scripting language including VBscripts has disc accessing features and so it is accepted to be a safe language to deploy in the web.

Easy To Use And Learn

If you already known visual Basic or Visual Basic for applications, VBScript will be very familiar already known Visual Basic, once you learn VBScripts on your way to programming with the whole family of Visual Basic languages.

It is a free language and so it can be used anywhere. We can write our VBSCRIPT program in notepad and execute it using internet explorer. You can use the SCRIPT element to and VBScript code to an HTML page.

The <SCRIPT> Tag

Any scripting language can be enclosed within a script tag in a ordinary HTML file. The script tag can be placed anywhere in the HTML page and its is better to place it in the head tag. We should mention the language as an attribute. It is a container tag.

```
<script language = vbscript>
</script>
```

CHAPTER TWO
VISUAL BASIC PROGRAM CONCEPTS

Variable

In Visual Basic, as in any other programming language, variables store values during a program's execution. A variable is a named storage location that can contain a value, which can contain a value, which can be modified during the program execution or if the value of a specified memory location changes at run time then it is defined to be a variable. The name with which we assign or manipulate the memory location at runtime is called a variable name. each variable has a Name (which uniquely identifies it), value and type. For example, the variable Name can have the value "Ram", and the variable Age can have the value 24. here name and age are variable name an "ram" and 24 are their values, and string and integer are types of variables name and age respectively.

Declaring variables

In most programming languages, variable must be declared. In other words, compiler must be told in advance, about the variable to be used. If the compiler knows the variables and their types, it can produce optimized code. When the compiler is informed that the variable age will hold a number, then it will set required bytes in memory to make ready to use it.

Note: the complier is a translator that translates the programming code into the Machine's understandable form in order to make it ready to execution

A variable name:

- Must begin with a letter
- Can't contain an embedded period or embedded type declaration character
- Must not exceed 225 characters.
- Must be unique within the same scope.

Explicit declaration

To declare variables and to allocate storage space the dim statement is used as in the following:

```
Dim meters as Integer
Dim greetings as String
```

The first variable, meters will store as integer value such as 10 or 388, and the second variable, greetings, will store a text such as "Happy Birthday". When the compiler finds as Dim statement, it creatures place holder by reserving some space in memory and assigning a

name to it. Each time this name is used its subsequent statement, Visual Basic uses this area in memory to read or set it value. For instance, when the following statement is used.

Meters = 143

Visual Basic places the value 143 in the place holder reserved for the variable meters. When the program asks for the value of this variable, Visual Basic get it from the same area of memory.

Types of Variables

The type of a variable determines how to allocate storage space in the computer's memory to their values. All variables have that determines what kind of data they can store.

Visual Basic recognizes the following types of variables:

- Numeric
- String
- Variant
- Date
- Object

The numeric Data Types

Visual Basic supplies several numeric data types- integer, long (long Integer) single (single –precision floating point), double (double-precision floating point), and currency. Using a numeric data type generally uses less storage space than a variant. If it is known that a variable. Will store whole numbers (such as 12,388,60 etc) rather than numbers with a fractional amount (such as 3.57, 0.7876, etc) then it is better to declare it as an integer or long type variable. Operations are faster with integers, and these types consume less memory than other data types.

Different type of variables can be declared in a single Dim statement. For example, the following declaration statements.

Dim var1as single
Dim var2 as Double
Dim var3 as Currency
Can be minimized as in the following
Dim var1 as single, var2 as double, var3 as currency.

Different types of numbers are represented internally in different formats. All numeric values are truncated to a certain extent. The result of the operation 1/3 is 0.333333… (an infinite number of digits 3). It will fill the total capacity of RM with the digit3, and the result will still be truncated. Here's simple but illuminating example.

Dim a as single, b as double

Single and Double are two basic data types for storing floating point number (numbers that have a fractional part), and the double type can represent these numbers more accurately than the single type. The following statements.

A=1/3
Debug. print a
Will produce the result in the immediate window.
0.3333333
And if the following statements are executed subsequently:

A= a*100000
The output will be
33333.34

The above result is not accurate, as it is not rounded properly. If this result is divided by 10000, its result will be 0.3333334, which is different from the initial result (0.33333333). This is an import at point in numeric calculations, and it is called error propagation. In long sequences of numeric calculations, errors propagate. If the above same operation is performed with the variable b, which is declared as double as in the following:

B=1/3
Debut. Print b

The result will be
0.333333333333333333
33333.33333333333
Which is more accurate.

The choice of data types of variable can make a difference in the result of the calculations. The proper variable types are determined by the nature of the values they present. The choice of data types is frequently a trade off between precision and speed of execution (less precise data type are manipulated faster). The following table shows rage and storage space allocated for each type of numeric variable.

Data type	Storage size	Range
Integer		-32,768 to 32,767
Long (long integer)		-2,147,483,648 to 2,147,483,647
Single (single – precision floating - point)		-3402823E38 to -1401298E-45 for negative values; 1,4019298R-45
Double (double – precision floating - point)		-1.7976931348623E308 to –4.940656445841247E for negative
Currency (scaled integer)		-922,337,203,685,477.5808 to 922

The byte data type

The byte type holds basically an integer in the range 0 to 225 and they are declared with the following statement.

Dim varname as Byte

As the name implies bytes allocated in the memory for byte data type is one. Byte variables are frequently used to access files, image and second files, and so on. All operators that works on integers work with the byte data type except unary minus. Since byte is an unsigned type with the range 0-255, it cannot represent a negative number.

String variables

The string data type stores only text, and string variable are declared with the following statements;

Dim varname as string

By default, a string variable is variable length string: the string grows or shrinks as new data is assigned to it. String variables can also be declared with fixed length as in below:

Syntax

Dim string * size

For example, to declare a string that is always 50 characters long, the following statement is used.

Dim Emp Name As string *50

If a string of fewer than 50 characters is assigned, the Emp Name is padded with enough trailing spaces to total 50 characters. If a string that is to long for the fixed-length string is assigned, Visual Basic simply truncates the characters.

If a string variable's size will change drastically during the course of an application, it is better to declare it as fixed-length to prevent Visual Basic form having to resize it constantly, which slows down the execution speed.

The Boolean Data Type

The Boolean data type stores true/false values. Although a single bit would be adequate, for efficiency reasons Visual Basic allocates two bytes t this data type. Boolean variables are declared as

Dim Running as Boolean

These variables are combined with the logical operators AND, OR and Not. The Not operator toggle the value of a Boolean variable. The following statement.

Running = true then

```
        Running = false
Else
        Running = true
End if
```

The Data Variable

Data and time values are stored internally as in a special way. A variable declared as Data:

```
Dim expiration as date
```

Can store both date and time values. The following are all valid assignments.

```
Expiration = "01/01/2003"
Expiration = "01/23/2003"
Expiration = "13:03:05: AM"
EXPIRATION = "02/23/2003 13:03:05:AM"
```

Two date variables can be subtracted to know the difference as in the following:

```
Dim Date 1 as date, Date2 as date;
Dim days as integer;
Date1-"01/02/2003"
Date2 = "01/08/2003"
Days = Date2-Date1
```

In the above example the integer variable days will hold he value 6 which is the difference between the dates date1 and date2 . The value are stored in the date variables in mm/dd/yyyy (mm-month, dd-day, year-year) format

Integer values can be added to a date variable to add days. For example, if date holds the date "01/01/2003" the following line

```
Debut. print date 1+10
Will print the value "01/11/2003"
```

Note: to add an hour to a date variable, add 1/24 of a day; to addaminute add 1(24* a day

Object variables

Object variables are stored as 32-bit (4byte) addresses that refer to object within an application or within some application. A variable declared as object is one that can subsequently be assigned (using the set statement) to refer to any actual object recognized by the application.

For example, it has been assumed that a form in a Visual Basic application has two command buttons namely command 1 and command 2. and two object variables are declared as follows:

Dim a as command button, b as command button;

Each of these two object variables can be set to one of the two command button with the following statements:

Set a= command1
Set b= command2

From now on the command buttons properties can be manipulated through the variables a and b. to change the caption property of the first command button, the following statement is used:

a.Caption = “Click me”

To turn on the bold attribute of the second command button (so that is caption appears in bold), the following statement is used;

b.font Bold = true

This section just briefed about object variables. Objects and object variables are explained in detail in chapter “Object Programming in Visual Basic

The variant variable

A variant variable is capable of storing all system-defined types of data. Variant variables are declared without specifying a type as follows:

Dim var

Or for cleaner code it cab be declared as Dim var as variant. Variant variable an be used in both numeric and string calculations. There is no need to perform any type of conversions. Visual Basic automatically performs any necessary conversion. For

Example

```
Dim some value                    ‘variant by default.
E value =”17”                      ‘some value contains “17”
Some value = some value-15         ‘some value now contains the
                                    “numeric value2.
Some value = “U” & Some value      ‘some value now contains “U2” (a
                                    tow character string)
```

Constants

some variable don't change value during the execution of a program. These are constants that appears many times in the program code. Math calculations, the value of pi (3.14159) may appear many times in the program code.

These values are best represented by constants. Instead of typing the value 3.14159 over and over again, a constant will be defined with a constant name say pi, and it can be used in the program code as follows;

Area = pi* radius ^2

Creating Your Own Constants

The syntax for declaring a constant is:

[public| private] Const constantname [As type] = expression

The argument constant name is a valid symbolic name (the rules are the same as those for creating variable names), and expression is composed of numeric or string constants and operators; however, you can't use function call in expression.

A const statement can represent a mathematical or date/time quantity:

Const con Pi = .141592265358979
Public Const con max plants as Integer =9
Const con release date =#1/1/95#
The const statement can also be used to define string constants:
Public const con version "07.10A"
Const con code name = "Enigma'

You can place more than one constant declaration of a single line if you separate them with commas:

Public const con Pi = 3.14, con Max Planets =9,_
Con Worldpop =3E+09

The expression on the right side of the equal sign (=) is often a number or literal sting, but it can also be an expression that results in a number or string (although that expression can't contain calls to functions). You can even define constants in terms of previously defined constants:

Const con Pi2 = con pi*2
Once you define constants, you can place them in your code to make it more
Readable. For example:
Static SolarSystme 91 to con Max planets)
If num people >con World Pop then Exit sub

Operators

Operators can be classified into three main categories

1. Arithmetic Operators
2. Relational or Comparison Operators
3. Logical Operators

The table lists the operators available in vbscript and most of them are self explanatory.

Arithmetic		Comparison		Logical	
Description	Symbol	Description	Symbol	Description	Symbol
Exponentiation	^	Equality	=	Logical Negation	Not
Unary Negation	-	Inequality	<>	Logical Conjunction	AND
Division	/	Greater than	>	Logical Exclusion	Xor
Multiplication	*	Less than	<	Logical Disjunction	Or
Integer	\	Less than or equal to	<=	Logical Equivalence	Eqv
Modulus Arithmetic	mod	Grater than or equal to	>=	Logical Implication	Imp
Addition	+	Object equivalence	Is		
Subtraction	-				
String Concatenation	&				

ARRAYS

A standard structure for storing data in any programming language is an array. Whereas individual variables can hold single entities, such as a number, a date, or a string, arrays can hold sets or related data. An array has a name, as does a variable, and the values stored in it cab be accessed by an index.

Suppose if salary for 10 employees is to be stored, then variables Salay1, salary2 …Salay10 could be created to hold all 10 employee's salary. But if number of employee's goes to very large extent then it is impossible to create a variable for each employee. This way it not elegant too. Arrays are useful here, as they allow to refer to a series of variables by the same name and to use a number (an index0 to tell them apart. Arrays are declared as in the following;

Dim salary (99) as Integer

This declaration create an array named salary with 100 elements, with index numbers running from 0 to 99 and holds 100 values; salaries of 100 employees. Salary (0) is the first person's salary, salary (1) the second person's salary and so no. values are assigned to the elements of arrays as follows:

Salary (0) = 1000

All elements in an array have the same data type, of course, when the date type is variant, the individual elements can contain different kinds of data. By default, the first element of an array has index 0 (lower-bound) and the number that appears in parentheses in the Dim statement is the array's upper limit (upper-bound). However, the array's first element need not be zero. Lower limit (lower-bound) can explicitly be specified as in the following:

Dim Salary (1 to100) as integer

The lower bound can have any other value, provided it is smaller than the upper bound. The following declarations are valid:

Dim Greetings (10 to 20) as String
Dim Amounts (100 to 101) as Double

Suppose if the array Salary contains salaries of all the employees, one thing that is to be remembered is which person corresponds to each salary. This can be avoided by declaring another array names as in the following:

Dim Names (1 to 100) as string

Now values to the elements of both arrays can be assigned as in the following:
Names (0) =' Ekambaram"
Salary (0) = 2000
Names (1) = "prema"
Salary (1) = 15000
Name (2) = "Gayathri"
Salary (2) =50000

Multi – Dimensional Arrays

Sometimes when storing values in an arrays, its related information has also to be stored. For example, to keep track of each pixel on a computer screen, both X and Y coordinates are to be referred. This can be doe using a multidimensional array to store the values.

Arrays of multiple dimensions can be declared as follows:

```
Dim Matrix (9, 9) as double
```

The above statements declares a two dimensional 10 by 10 array. But arrays can be extended for more dimensions as in the following:

```
Dim Multi D (3, 1 To 10, 1 to 15)
```

This declaration creates an array that has three dimensions with sizes 4 by 10 by 15. The total number of elements is the product of these three dimensions, or 600.

The benefit of using multidimensional arrays is that they are conceptually easier to manage.

Dynamic Arrays

Sometimes when declaring arrays its required size may not be known. Instead of making it large enough to hold the maximum number of data (which means the most of the array may be empty), arrays can be declared dynamically.

A dynamic arrays can be resized at time. Dynamic arrays are among the most flexible and convenient features in Visual Basic and they help to manage memory efficiently. For example, a large array can be used for short time and then can be feed up memory to the system when that array is no longer used.

Dynamic arrays are created in the same way as normal arrays, but without specifying it dimensions as in the following:

```
Dim Dyn Array ()
```

Later in the program, when the number of elements to be stored in the array is known, the Redim statement is used to redimension the array with the required size as follows.

```
Redim Dyn Array (count)          'Count is probably as user-
'entered value.
```

The Redim statement can appear only in a procedure. Unlike the Dim statement Redim is executable and so, can appear only inside a procedure. An array can redimensioned to multiple dimensions as follows:

```
Dim Matrix () as Double
Redim Matrix (9, 9, 9)
```

Note: the Redim statement can't change the type of the array. Moreover, subsequent Redim statements can change the bounds of an array and number of its dimensions

REM Statement

Rem comments

Used to include explanatory remarks in a program. Allows comments to be added to a program. Everything on the line after the Rem statement is ignored by Visual Basic. A apostrophe (‘) can also be used in lieu of Rem statement.

Syntax

Rem comment
Or
‘comment

The comment argument is the text of any comment you want to include. After the Rem keyword, a space is required before comment.

The following examples illustrates the use of Rem statement:

```
Dim MyStr1, MyStr2
MyStr1 = “Hello”: Rem comment after a statement separated by a colon.
MyStr2 = “Goodbye” ‘This is also a comment; no colon is need.
Rem Comment on a line with no code; no colon is needed.
```

End statement

Ends a procedure or block.

Syntax
End
End function
End if
End property
End select
End sub
End type
End with
The End statement syntax has these forms.

Statement	Description
End	terminates execution immediately. Never required by itself but may be placed any where in a procedure to end code execution, close files opened with the open statement and to clear variables.

End Function	Required to end a Function statement.
End If	Required to end a block if .. then … Else statement.
End property	Required to end a property Let, property Get, or property set procedure.
End select	Required to end a select case statement.
End Sub	Required to end sub statement.
End Type	Required to end a user-defined type definition (type statement).
End with	Required to end a with statement.

CHAPTER THREE
Functions & control Flow statements

A function is similar to a subroutine, but a function returns a result. Subroutines perform a task and don't report anything to the calling program; functions commonly carry out calculations and report the result. The statements making up a function are placed in a pair of function/ End function statements. Moreover, because a function reports a result, it must have a type, as in the following:

```
Function Next Day () As Date
Nest Day =date() + 1
End function
```

The next day () function returns tomorrow's date by adding 1 day to the current date. Because it must report the result to the calling program, the Next Day() function has a type, as do variables, and the result is assigned to its name (which can't be done with subroutines.)

The Exit function statement causes an immediate exit from a function procedure, as the Exit Sub statement does in the subroutine. Program execution continues with the statement following the statement that called the function.

Input Box Function

Display a prompt in a dialog box, waits for the user to input text or click a button and returns the content entered by the user.

Syntax

Input Box (prompt [title] [default] [xpos] [yops])

The arguments are described in the following table 1.3

Argument	Description
prompt	Required string expression display as the message in the dialog Box.
Title	Optional. String expression displayed in the title bar of the dialog Box. If omitted, the application name is placed in the title bar.
Default	Optional. String expression displayed in the text box as the default Response if no other input is provided. If omitted, the text box is Displayed empty.
Xpos	Optional. Numeric expression that specifies, in twips, the hori distance of the left edge of the dialog box from the left edge of the Screen. If omitted, the dialog box is horizontally centered.

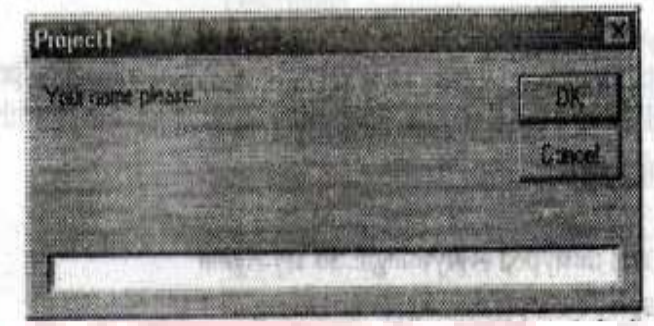
Yops	Optional. Numeric expression that specifies, in twips, the vertical distance of the upper edge of the dialog box from the top of the screen. If is omitted, the dialog box is vertically positioned approximately one-third of the way down the screen.

Table: Arguments of Input Box function

Example 1

```
Dim Name as String
‘specifying prompt alone
Name = inbox (“Your name please…”)
```

The above statement will display the input box as in figure 1.13



When the user enters his name and clicks OK, the Input Box return his name to the variable Name.

MsgBox Function

Displays a message in a dialog box, waits for the user to click a button and returns and integer indicating which button the user clicked.

Syntax

MsgBox (prompt [buttons][title])

The prompt and title arguments are same as in Input box function. The buttons argument is a numeric expression that is the sum of values specifying the number and type of buttons to display, the icon style to use, the identify of the default button, and the modality of the message box. If omitted, the default value for buttons is (). It settings are given in the following table 1.4

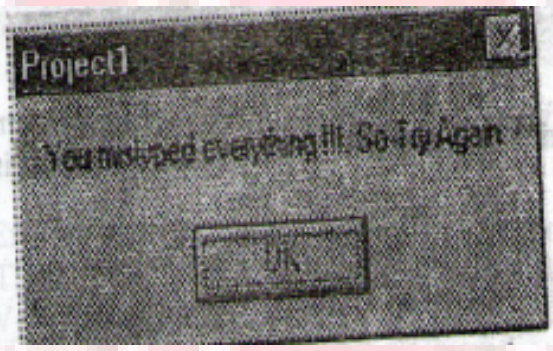
Vb OK only	0	Display OK button only
Vb OK cancel	1	Display OK and cancel buttons
Vb Abort Retry Ignore	2	Display Abort, retry and Ignore buttons.

Vb Yes No Cancel	3	Display Yes, No and cancel buttons.
Vb Yes No	4	Display Yes, and No buttons.
Vb Retry Cancel	5	Display Retry and cancel buttons.
Vb Critical	16	Display Critical Massage icon.
Vb Question	32	Display warning Query icon.
Vb exclamation	48	Display Warning Message icon.
Vb Information	64	Display Information Message icon.
Vb Default button 1	0	First button is default
Vb Default button 2	256	Second button is default
Vb Default button 3	512	Third button is default
Vb Default button 4	768	Fourth button is default.

Table: Values of the buttons argument.

Example 1

MsgBox "you mistyped everything!!! So try again"
This statement display a simple message box as in figure 1.15



Control Flow Statements

Programs are not monolithic sets of commands that carry out the same calculations every time they are executed. Instead, they adjust their behavior depending on the data supplied or based on the result of a test condition. Visual Basic provides three – control flow, or decision, structures to take a course of action depending on the outcome of test. They are

- If… Then
- If… Then…Else
- Select case

If … Then … End If

The if structure test the condition specified and, if it it's true, executes the statement (s) that follow. The if structure can have a single line or multiple line syntax.

If condition Then Statement

In the above statement, Visual Basic evaluates the condition and, if it's true, executes the statement that follows. If the condition is not True, it conditions with the statement following the If structure. Multiple statements can also supplied, provide they must be separated by colon as in the following:

If condition Then Statement1: Statement2: Statement3

Here is an example of single line in statement:

If Salary > 3000 Then DA percent =100

This statement can be broken into multiple lines, as it will be easier to read as follows:

```
If Salary > 3000 then
        DA percent =100
End if
```

When a statement that is to be executed, are placed next to the if statement's line, to mark the end of if block End If statement is used as in the above example.

If … Then … Else … End … If

A variation of the If … Then is the if … Then … Else Statement, which executes one block of statements if the condition is true and another if the condition is false .

Syntax

```
If condition Then
        Statement block – 1
Else
        Statement block – 2
End if
```

Visual Basic evaluates the condition. If it's true, it executes the first block of statements and then jumps to the statement following the End If statement. If the condition is False, Visual Basic ignores the first block of statements and executes the block following the Else Keyword.

Example

```
If salary >5000 Then

        DaPercent =100
        Hra Percent = 80
Else
        Dapercetn = 80
        Hrapecent = 60
```

End if

Another variation of the If… Then … Else statement uses several conditions, with the Elseif keyword.

Syntax

```
If condition 1 Then
        Statement block -1
Elseif condition2 Then
        Statement block - 2
Elseif condition3 Then
        Statement block – 3
Else
        Statement block – 4
End if
```

A If statement can have any number of Else if clauses. The conditions are evaluates from the top, and in one of them is True, the corresponding block of statements is executed. The Else clause will be executed if none of the previous expressions is True.

Example

```
If score > 90 Then
        Result = “Excellent”
Elseif Score >75 Then
        Result = “Very Good”
Elseif Score > 50 Then
        Result = “Good”
Else
        Result = “Failed”
End If
```

This statement is easier to read, but not as efficient in terms of execution time because Visual Basic evaluates all conditions, even if the first one in True. This inefficiency is overcome with Select case statement.

Select Case

The select case structure compares the same expression to a different value. The advantage of the select case statement over multiple if. Then. Else statement is that it makes the code easer to read and maintain.

Syntax

```
Select case expression
```

```
        Case Value 1
                Statement block – 1
        Case Value 2
                Statement block – 2
        ---------------  ---------------
        --------------------------------
        Case Else
                Statement block -3
End Select
```

Some case statements can be followed by multiple values, separated by commas or can have a range or a single condition as given in the example below.

```
Dim Number as Integer

Number = {Some integer value}

Select Case Number                      'Evaluate Number'
Case 1
Print "Number-1"

Case 2, 3,4,5,6                         'Number between 2 and 6'
Print "Number – between 2 and 6"

Case 7 to 10                            'Number between 7 and 10'
Print "Number –between 7 and 10"

Case Is <=20
Print "Number –between 11 and 20"

Case Else
Print "Number Not between 1 and 20"

End select
```

Loop statements

Loop statements allow executing one or more lines of code repetitively. Many task consists of trivial operations that must be repeated over and over again, can be done using loop statement. Visual Basic supports the following loop statements:

Do… Loop

The Do…Loop statement is used to execute a block of statements for an indefinite number of times. There are several variations of the Do… Loop statement, but each evaluates a

Boolean Condition to determine whether a continue execution. As with if… then, the condition must be a value or expression that evaluates to False 9zero) or to True (nonzero).

The following Do … Loop executes the statements as long as the condition is True:

```
Do While Condition
        Statement
Loop
```

When Visual Basic executes this Do Loop, it first tests condition. If condition is False 9zero0, it skips past all the statement. If it's True (nonzero), Visual Basic executes the statements and then goes back to the Do while statement and tests the condition again.

Consequently, the loop can execute any number of times. As long as condition is nonzero or True. The statements never execute if condition is initially False.

Example

In this example the loop block is executed while the variable 1 is equal to 10

```
Dim I as Integer
I = 10
Do while I = 10
I= {some Expression} 'Some expression may return 10 or other value.
Loop
```

When the condition I = 10 is tested, the result will be true as I holes the value 10. so the execution of the loop block will begin. If some expression returns 10, the loop block will be executed again. Otherwise the execution will jump on the next statement of the loop block.

Another variation of the Do… Loop statement executes the statements first and then tests condition after each execution. This variation guarantees at least one execution of statements:

Do
Statements
Loop while condition

For … Next

Do … loop is useful when a block of statements are to be executed for unknown number of times. But if a block of statements are to be executed for specific number of times the a For … Next loop is a better choice. Unlike a do loop, a for loop uses a variable called a another that increases or decrease in valued during each repetition of the loop.

Syntax

For counter = start To End [Step Increment]

Statements
Next [counter]
For Each … next

A for each … next loop is similar to a for … Next loop, but it repeats a group of statements for each element in a collection of objects or in an array instead of repeating the statements a specified number of times. This is especially helpful when the number of elements of a collection is not know.

Syntax

For Each element **In** group
Statement
Next element

Example

This example prints the contents of array **Names.**

```
Dim Age (10) as integer
Dim Item ‘Must be variant when used with arrays
----------------------
----------------------
For each item in Age ()
        Print item

    Next
```

With … Endwith

WITH … ENDWITH provides a convenient way to specify a number of properties for a single object. Note that you can also execute methods from within a WITH … ENDWITH structure, specifies multiple properties for an object.

Syntax

WITH object name
Statement
ENDWITH

CHAPTER FOUR
CONTROLS

In the previous chapter we have discussed about variables, constants, loops and control statements. This chapter will give the ways of creating and implementing the controls. It also helps us to understand the concept of control Arrays. Controls are used to receive user input and display output and have its own set of properties, method and events. Let us discuss few of those controls in this chapter.

Controls

Customs controls are the building blocks of a Visual Basic application. Controls are used to add significant functionality to our programs, and they are easy to work. In Visual Basic, forms are the foundations and are generally used to build programs. A form is where you put all the things that people interact with as they use your program. Those things you put on the form are controls, which enable the people who use your program to do things such as enter text and click buttons. That is, controls are also interfaces between the user and the application. Every control has got its own properties, events and methods. You can set the properties and write the code to make the control active.

For example if you build a house, you start with a foundation – think or this as the form. On top of the foundation, you add all the things that allow you use the house; a floor, walls and doors. These things are the controls.

The toolbox is the containing window that holds the custom controls for your applications (see figure 4.1). These are also several advanced controls that come with Visual Basic. Some controls work with multimedia, and others utilize the Internet. Best of all, you can now cerate your own ActiveX custom controls and add them to you Toolbox.

Adding and removing controls

You can add controls to a form in tow ways: by double clicking and by drawing. Whenever you double click an icon on the toolbar, the associated controls appears on you form. When you do this, though, you control where the control goes; you're at the mercy of Visual Basic. When you draw a control on your form, you can put it wherever you want it.

Draw a control on a form

1. Click the control's toolbox icon
2. Move the mouse pointer over your form. Notice that your pointer is now shaped as a crosshair instead of an arrow.
3. Click (and hold) the mouse button where you want the control to go.
4. Drag the mouse down slightly and to the left. As you move the mouse, notice that a box stars to appear.
5. When the box is the proper size, let go of the mouse button. The control you selected now appears on the form.

Remove a control from a form

1. Select the control you want to delete by clicking it. The control you selected will appear with a box at each corner and side.
2. Press the Delete key.

You can also remove a control by right clicking it. From the context menu that appears and selecting Delete.

How to Size and position a control

When you're drawing controls on a form, you don't have to be exact. It's very easy to make them bigger or smaller, and to put them in a different spot on the form.

Size controls with the mouse

In the toolbox, select the pointer tool (if it isn't already selected0. on your form, select the control you want to resize. Grab a sizing handle with the mouse by moving the pointer over it and then holding down the left mouse button. You know when you're over the sizing handle because the mouse pointer turns into a double – sided arrow. While holding down the mouse button, notice that a box appears. The box shows you what the size of the control will be. When it's the right size, release the mouse button. Changing the position of a control is also easy. Just click it to select it, and drag it to its new position.

Intrinsic controls

Intrinsic controls are automatically displayed in the tool box when a form is loaded. When you program in Visual Basic you'll use a relatively small set of controls. However, these controls are very powerful. With them, you can add buttons, check boxes, labels and text boxes to your programs. You can use them to see files on you hard drive right from your program. You can even read database! These basic controls are intrinsic controls.

From object

The most basic object you will be working with in Visual Basic is the from object, which is the visual foundation for building an application. It is basically a window that you can add different elements to in order to create a complete application. Every application you can see on the screen is based on some type of form. Before going into the details of application development, let's take a look at the most basic form object that you will probably use most often in your projects: the single form.

As you learn Visual Basic, most of your applications will only have one interface, which is the single form. When you become more experienced and start writing letter, editor styled applications, you may want to use multiple forms, called documents.

The appearance of Forms

The main characteristic of a form is the title bar, on which the form's caption displayed. On the left end of the title bar is the control menu icon. Clicking on this icon opens the control menu. On the right side of the title bar are three buttons minimize, maximize, and close. Clicking on these buttons minimizes, maximizes and closed the form. When a form is maximized, the maximize button is replace by normal button, which restores the form to its size and position before it was maximized.

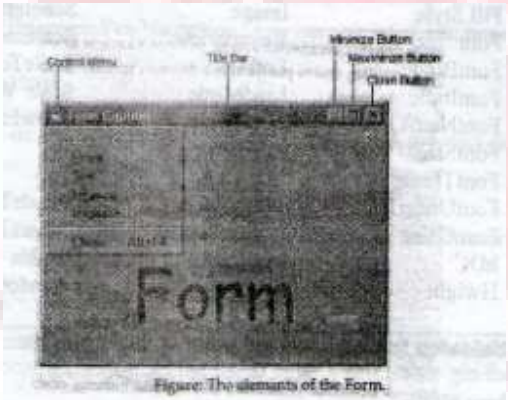


Figure: The elements of the Form.

Control Menu

Control menu is a simple menu that allows you to restore, move, resize, minimize, maximize and close the form. The enable this button on your from, set the form's control box property to true in the form's properties window or determines whether the control-menu box is displayed on the form at run time. That is, if it is set to false the control box won't appear at left corner of the title bar of the form.

The control menu controls the following commands.

- Restore — Restore a maximized form to its size before it was maximized. Enabled only if the form is maximized.
- Move — let the user move the form around with the keyboard.
- Size — Lets the use resize the control with the keyboard.

- Minimize Minimizes the form.
- Maximize Maximizes the form.
- Close Close the form.

Form properties

The following shows the properties for a form object

ActiveControl	Drew width	Help Context ID	Negotiate Menus
Active From	Enabled	h Wnd	Picture
Appearance	Fill Color	Icon	Scale Height
AutoRedraw	Fill Style	Image	Scale left
Backcolor	Font	Keypreview	Scale mode
BordeStyle	Font Bold	Left	Scale Top
Caption	Font Italic	Lind Mode	Scale Width
ClipControls	Font Name	Link Topic	Shown in Taskbar
Control Box	Font Size	Max Button	Tag
Controls	Font Transparent	MDIChild	Top
CurrentX	Font Underline	Mouse Icon	What's This Help
CurrentY	Fore Color	Mouse Pointer	What's This button
Draw Mode	HDC	Moveable	Width
Draw Style	Height	Name	Window State

The load and Unload to load and unload the forms. The load statement has the following syntax:

Load form Name

And the Unload statement has this syntax:

Unload form Name

The form Name is the name of the form to be loaded or unloaded. When Visual Basic starts, it loads the default Form, say Form1, and displays it. To load other Forms, say form2, the load statement is to be issued as in below:

Load Form2

This statement will load the Form2 into the memory. But the Form2 will not be visible immediately. To make it visible its show method (explained short) must be called.

This method removes a form from the screen, but doesn't unload it

Form Name. Hid

When a form is hidden, the user can't interact with it.

Common Intrinsic Controls

Controls are elements that provide the user interface. The user interface is what appears in the application's window when it runs. It enables the user to work with the application such as entering text and clicking buttons. The elements of the User Interface are common to all

windows applications and they are called as Intrinsic controls. They are all shown as icons in the Visual Basic toolbox.

The text box control

It provides an area to enter or display text. It behaves like a min text editor.

The label control
This control displays text on a form that the user can't edits. Labels commonly identify other controls.

The command button control
Carries out a command or action when a user chooses it.

The option button control
Option buttons, or radio buttons, appear in a group, and the user can choose only one of them

The list box control
It contains a list of options from which the user can chose one or more. Ti can contain many lines, and the user can scroll the list to locate an item.

The combo box control
The combo box control is similar to the List box control, button it contains a text edit field. The user can either choose an item form the list or enter a new string gin the edit filed.

The picture control box
The picture box control is used to display bitmaps, icons or windows metafiles.

The image control
The image control is similar to the picture box control and it can also display images, but it supports only a few features of the picture box.

The shape control
The shape control is used to draw graphical elements, such as boxes and circles on the surface of a form.

The line control
The line control is used t draw lines on the surface of a form.

The frame control
This control is used to draw boxes on the form and to group other elements.

The drive list box control

It displays the drives on the system in a drop down list from which the user can select a valid drive.

The directory list box control
It displays a list of all folders in the current drive and lets the user move up or down in the hierarchy of the folders.

The file list box control
It displays a list of all files in the current folder.

The timer control
It is used to perform tasks at regular intervals.

The horizontal and vertical scroll bars
These controls can be used as inputs devices or indicators of speed or quantity for example to control the volume of an computer game or to view the time elapsed in a timed process.

Common methods

- Move Changes an object's position in response to a code request.
- Drag Handles the execution of a drag and drop operation by the user.
- Set Focus Gives focus to the object specified in the method call.
- ZOrder Determines the order in which multiple objects appear on screen.
- Refresh Forces a complete repaint of a form or object
- TabIndex Determines the tab order of most objects within their parent form.

The following is the list of common events:

- Click The user click primary (left) mouse button on an object.
- Dbl Click The user double clicks the primary (left) mouse button on an object.
- Drag drop The user drags an object to another location.
- Drag Over The user drags an object another over control.
- Got focus An object receives focus.
- Key Down The user presses and releases a keyboard key while an object has focus.
- Key up The user releases a keyboard key while an object has focus.
- Lost Focus An object loses focus.
- Mouse Down The user presses any mouse button while the mouse pointer is over an object.
- Mouse Move The user moves the mouse pointer over an object.
- Mouse Up The releases any mouse button while the mouse.
- Change Indicates that the contents of the control have changed.

Option Button

Option button controls (shown here) are used to allow the user to select one and only one, option from a group of option. Usually option button, if there is to be only one group together within a frame control, but they can also be grouped on a plain from, if there is to be only one group of option buttons. Thus, if you had a frame specifying a delivery method, you might have one button for UPS (united Parcel Service) and another for courier delivery. Products can only be shipped by one of these methods (not both and not one). In contrast, option buttons representing, say bold and italic settings for text would not make sense. Text can be both bold and italic, or neither (none).

Check Box control

A Check Box control is rather similar to an option button. Both often partake in groups, and the value property is tested to see if a check box is on or off. But there are two fundamental differences between check boxes and option buttons: check boxes are valid as single controls- a single option button is probably counter intuitive. Check boxes (even when a group) are not mutually exclusive. Finally, check boxes have three possible settings for the value property.

The picture box control

As you might expect, picture boxes often graphics (for example, bitmaps, icons, JPEGs and GIF s). In this role. Picture boxes are similar to image controls. However, picture boxes and images have slightly different properties and therefore behave differently. If you just want to show a picture, then a image control is usually a better choice than a picture box, images take up less memory and are a light weight version of picture boxes. However, if you want to move the graphic around the form, a picture box produces a smoother display. In addition, you can create text and use graphics methods in a picture box at run time. The graphics methods enable you to draw lines, circles and rectangles at run time. But most importantly for this application, picture boxes can act as containers for other controls. Thus you can place a command button within a picture box. In this respect, picture boxes function as “forms within forms”

Picture Box properties

Picture sets the graphic to be displayed in the control. At design time a graphic file name can easily be specified through the properties window. But to assign a picture at run time the load picture statement is used as the following:

Picture1. Picture = Load Picture (“C: /Windows/ Clouds.bmp”)
The load picture statement loads the clouds. Bmp file from the disk to memory and assigns it to the picture property of the picture 1 control.

Auto size (Boolean) Determines whether a control is automatically resized to display its entire contents.

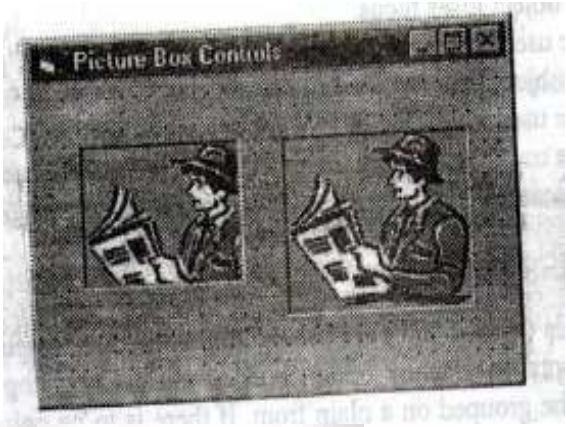


Figure: Different appearance of a same picture due to different values of the

Figure: Different appearance of a same picture due to different values of the Auto size property of the picture box control.

In the above figure 1.28 the first picture box control has the value false in its Auto size property. So only portion of a picture that can fit into the current coordinates of the first picture box is displayed. The second picture box has the value true in its Auto size property. So , it is automatically resized to accommodate the entire picture.

The image Box control

The image control (prefix-img) is first picture box control has the value false in its Auto size property. So only portion of a picture that can fit into the current coordinates of the picture box is displayed. The second picture box has the value true in its Auto size property. So, it is automatically resized to accommodate the entire picture.

The image control that comes with Visual Basic can now display bitmap (>BMP), icon (ICO), metafile (>WMF), JPEG (>JPG) and (GIF) (>GIF) files. This makes it easier to display graphics from the World Wide Web, as well as graphics from other popular graphics program.

Image Box Properties

Picture same as in picture box control.

Stretch (Boolean) Indicates whether the picture has to be resized to fit the size of the control. This property is totally different from the picture box control's Auto Size. When the auto size is set to false, only portion of the picture that fits into the current coordinates of the picture box is displayed. But when stretch is set to false the image control is resized to the coordinates of the picture. That is it behaves in the similar manner of the picture box control when its auto size is set to true. But when the stretch is set to true the picture will be enlarged or compressed to the current coordinates of the image control.

The Timer Control

The timer control is one of the few controls always hidden at run time. This means you don't have to find room for it on a form-it can go anywhere, even on top of existing controls. The timer basically Do just one thing: it checks the system clock and acts accordingly.

The List Box Control

A list box is an ideal way of presenting users with all list of data. Users can browse the data in the list box or select one or more items as the basis for further processing. The user can't edit the data in a list box directly-one way around that is to have a combo box instead; combo boxes are discussed next. When the list of data is too long for the list box then VB will add a vertical scroll bar.

List Box Methods

Add item adds an item to a list box at run time. For example, if the text in a textbox entered by the user is to be added to a list box this method can be used to do the same.

Syntax

List. AddItem item, index

The item argument specifies the text to be added to the list box. The index argument (optional) specifies the position whether the new item is to be place. For example, if the new item is to be inserted at the top of the list the index argument will have the value 0. if this argument is omitted, the new item will be placed are the bottom of the list.

The Month View Control

The month view control makes it easy for user to view and set data information via a calendar like interface. Users can select a single date or a range of dates.

The control can be navigated using either the keyboard or mouse. Buttons at the top to the control are used to scroll months in and out of view.

In addition, the control has the ability to display up to 12, months at a time. This can be helpful to give users the ability to view data information around the date of interest.

Control Arrays

A control array is an array of controls, in the same way that an integer is an array of integers. A group of controls that share common names, types and event procedures. Each control has a unique index. When a control in the array recognizes an event, it calls the event procedure for the group and passes the index as an argument, allowing your code to determine which control recognized the event. When you add a second control of the same type of the form, Visual Basic presents you a message asking 'Do you want to create a control array'. Choose yes to create a control array.

By grouping similar controls into an array (much likes arrays that hold strings or some other data type), you can down on the amount of code in our program. Because all of the controls in the array share event handlers (for example), there is only one click event for an array of command button controls no matter how many controls are in the array) you do not have to duplicate event code for each individual control.

Combining dynamic controls with control arrays offers the best of both worlds' dynamic controls enable your program to do a little jig in step with your end user's needs the control) array enables you to easily mange the pool of dynamically crated control.

There are many benefits to using control arrays instead of several individual controls Fist, control arrays use fewer resources then individual controls. Second, controls arrays share common events. For example, if you have an array of command buttons. The same click event procedure is called no matter which button is pressed.

If you start with a text box name Text1 and select to create a control array. Visual Basic assigns Text1 an index number of 0 and creates a new member of the Text1 array. With an index number of 0 and creates a new number of the text1 array, with an index number of 1. if you make additional copies of Text1 or Text2, their index properties are set to 3,4 and on.

Example showing how to handle a click event for control arrays

```
Private Sub cmd button Array_Click (Index As Integer)
    Select case Index
        Case 0 textindex. Text = 1
        Case1: textindex. Text = 2
            Case2: textindex. Text = 3
            Case3: textindex. Text = 4
            Case4: textindex. Text = 5
            Case5: textindex. Text = 6
              End Select
            End Sub.
```

CHAPTER FIVE

PROCEDURE

When we write a lengthy program some times we may have to write a set of statements repeatedly. It will be a time consuming one. So to avoid that we can write the code and deep in separate file and when ever we need to execute the statements we can call it. The advantages of using procedures in programming are (a) It is easier to debug a program with a procedures, which breaks a program into discrete logical limits. (b) Procedures used in one program can act as building blocks for other programs with slight modifications.

General Procedures

A general procedure tells the application how to perform a specific task. Once a general procedure is defined, it must be specifically invoked by the application. By contrast, an event procedure remains idle unit called upon to respond to events caused by the user or triggered by the system.

Why create general procedures? One reason is that several different event procedures might need the same actions performed. A good programming strategy is to put common statements in a separate procedure (a general procedure) and have your event procedures call it. This eliminates the need to duplicate code and also makes the application easier to maintain. For example, the VCR sample application uses a general procedure called by the click events for several different scroll buttons. Figure2.3 illustrates the use of a general procedure. Code in the click events calls the button manager sub procedure, which runs its own code, and then returns control to the click event procedure.

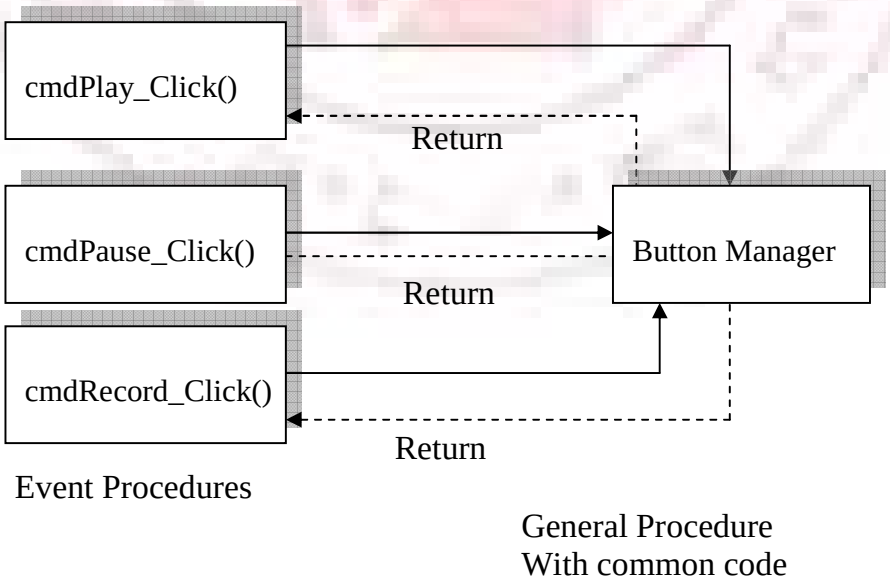


Figure: General procedures are called by event procedures

Event procedures

When an object in Visual Basic recognizes that an event has occurred, it automatically invokes the event procedure using the name corresponding to the event. Because the name establishes an association between the object and the code, event procedures are said to be attached to forms and controls.

- An event procedure for a control combines the control's actual name (specified in the Name property), an underscore (_) and the event name. for instance, if you want a command button named cmd play to invoke an event procedure when it is clicked, use the procedure cmdplay – click

- An event procedure for a from combines the word" Form", an underscore, and the event name. If you want a form to invoke an event procedure when it is clicked, use the procedure From _ click. (Like controls, forms do have unique name, but they are not used in the names of event procedures). If you are using the MDI form, the event procedure combines the word "MDI form", an underscore, and the event name, as in MDIF orm_Load.

All event procedures use the same general syntax.
Syntax for a control event
Syntax for a from event

```
Private Sub controlname _ eventname (arguments)
        Statements

End sub
Private Sub form_ eventname (arguments)
        Statements
End Sub.
```

Although you can write event procedures from scratch, it's easier to use the code procedures provide by Visual Basic, which automatically include the correct procedure names. You can select a template in the Code Editor window by selecting an object from the object box and then selecting a procedure from the procedure box.

It's also a good idea to set the name property of your controls before you start writing event procedures for them. If you change the name of a control after attaching procedure to it, you must also change the name of the procedure to match the new name of the control. Otherwise, Visual Basic won't be able to match the control to procedure. When a procedure name does not match a control name, it becomes a general procedure.

Creating And Calling Function

You have now met two types of sub procedures. The firs type (Private by default) is the event dub procedure behind forms and controls. These event procedures. Are already defined for you.The second type the ones you add yourself are called general sub procedures. There are also

functions procedures.These you must create yourself, and they can be private of public. Function procedures return a value to the procedures that call them. The best way to understand function procedures is to try one out. Here's one that calculates the cube root of a number.

1. Open a form ' code window by double –clicking it in the For Designer
2. Crete the template by typing the following on a blank line in a form's code window and pressing enter.

```
Public Function Cube Root
```

This Creates the following Coe:

```
Public Function Cube Root ()
End Function
```

3. Now alter the template as follows

```
Public function Cube root (x as Double) as Double
If x=0 then
    Cube Root =0
    Exit function
End if
Cube Root = 10 ((Log (abs(x))/ Log 910)/3)
If x <0 then
    Cube Root = Cube Root
End if
End function
```

The math involved here isn't important. But a few other things are:

- The function ends with an End Function rather than an End Sub
- You can jump out of a function with exit function.
- The function must return a value the line Cube Root = 0 is one of the three possible lines that do that.
- You can specify the typ0e of value returned here it's a Double (a numeric variable that can handle very large and small values as well as decimals) by adding the statements as followed by the data type of the return value.

4. To call this function, you must assign its return value to another variable. Try the following code in a click event for a form.

```
Dim Y As Double
    Y=Cube Root (27)
Print Y
```

5. Now run the application and click the command button.
   This code prints the return value straight onto the form

   Normally, you may want to assign the return value to a control on the form, such as a text box control.

```
txtText1.Text = Y
```

   the result is

Alternatively, you can have just the line

TxtText1.Text=CubeRoot (27)

Here the return value is being assigned directly to a control.

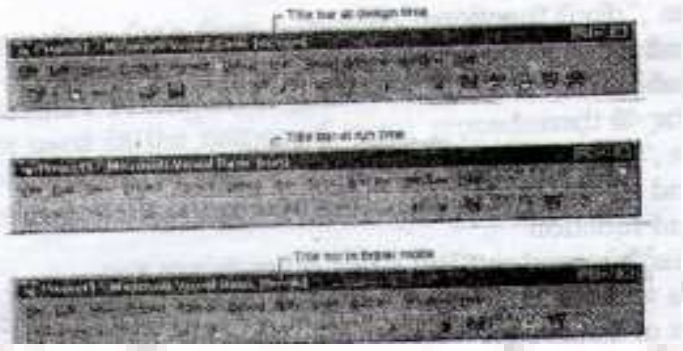
Hence you call a function procedure, you must enclose the parameter (s) in parentheses. This contrast with calling a sub procedure.

Design – Time, Run – Time, And Break Modes

Understanding the three modes of operations of the Visual Basic will ease the debugging process. Applications are built in Design-time mode and run in run time mode. This chapter introduces break mode, which suspends the execution of the program so that data can be examined and alerted.

Identifying the current mode

The Visual Basic title bar always shows the current mode. The following Figures show the title bar for design time, run time, and break mode.



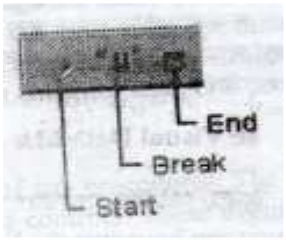
While an application is running, a situation may occur to switch to the suspend mode (break mode). For example, the execution of the application may be imprisoned in an indefinites loop. In this case the running application mighty be switched to the creak mode this is done:

- By choosing the break from Run menu.
- By clicking the break button.
- By pressing the CTRL +BREAK.

In the suspend mode, it is possible to view and edit code (choosing code from the view menu. Or pressing F7), example or modify data, restart the application, end execution, or continue execution from the same point.

Using the tool bar to change modes

The toolbar provides three buttons that lets to change quickly from one mode to another. These buttons appear as in Figure 7.2



Whether any of these buttons is available depends on whether Visual Basic is in run time mode, design time mode, or break mode. The following table lists the buttons available for different modes.

Mode	Toolbar button available
Design time	Start
Run Time	Break, End
Break	Continue, End (in break mode, the start button becomes the continue button)

CHAPTER SIX
Data base features

Computers are mainly used for keep track of information. In this unit we will go on creating files by using Visual Basic data manager and adding deleting the records and so on…

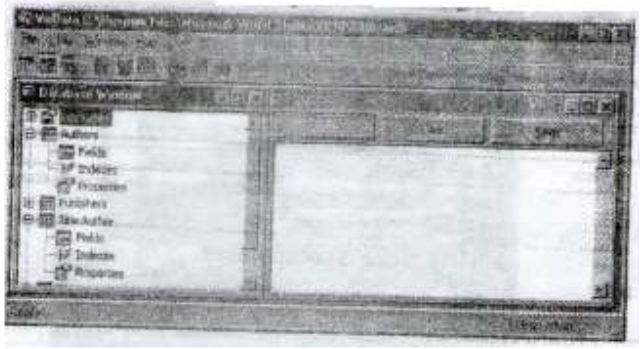
Visual data manager

Visual Data is an application that was created using Visual Basic itself. A built version of Vis Data is used as an add-in, accessible from the add-Ins menu in the Visual Basic figure

Figure: Visual Data Manager Option in Visual Basic Ad-Ins menu, which is used to activate the Visual Data Manager.

It is used to create database, tables, queries, and indexes and modify them. When an existing database is opened the visual data manager displays the contents and properties manager window with the BIBLIO database open in design mode.

Figure Vis Data Manager



Creating a database using a Visual Data Manager

To create a new database, choose file| New| Microsoft Access| Version 7.0 MDB. A dialog box will appear asking path and filename to store the database being created. Give a file name and choose desired location to save the database.

There is one more option in the Microsoft Access Submenu, Version 2.0 MDB, which is used to create database Inversion 2.0 format. Still man concerns use applications, which are built

using earlier version of Visual Basic. They may require database in older format. In this case along, the required database can be created using version 2.0 MDB option. Otherwise database can be created in latest version format (why to look back the stone Age world).

Note: to open an already existing access database choose File |Open database Microsoft Access |.And select the database name in the open Microsoft access database dialog box.

Creating a simple database

This section shows how to create a simple database students with a table students Dt1 (student details), which contain fields student, student name and Total Marks. It will be useful for rest of this chapter.

Choose File |New | Microsoft Access| version 7.0 MDB.
Type students in the file name text box and click OK.
Right click on te database window and choose New table.
In the table Structure box type student Dtls in the table Name Text box.
Click Add field button in the Table structure box to ad fields.
Type the value as given below in the corresponding text boxes.

Text box	**Value**
Name	Student Id
Type	Text
Size	10
Validation Text	Student Id is not specified
Validation Rule	<>””

- Click Ok to add the field name. The field Dialog box will still remain to accept another field. Proceed for the field student name with the value given below.

Text box	**Value**
Name	Student Id
Type	Text
Size	30
Validation Text	Student Id is not specified
Validation Rule	<>””

Now click Ok and proceed for the field Total Marks with the values given in the following:

Text box	**Value**
Name	Total marks
Type	Integer
Validation Text	Total marks is not specified or negative

Validation Rule >=0

Click OK and close to close the Add field dialog box.

Click build Table button in the Table structure dialog box to complete the creation of table students.

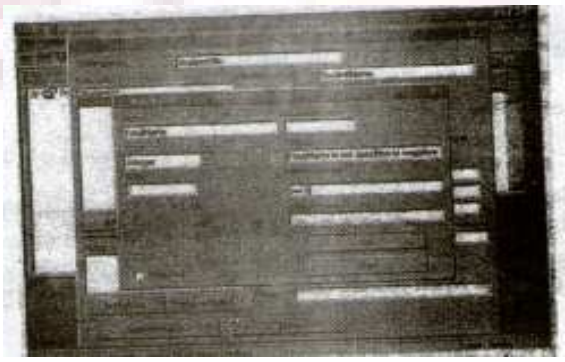
Creating a table

Once a database is created, it is ready to give birth to tables. To create a table, Right clicks in the database window and choose New table from the context menu Figure 9.3

Choosing the above option will bring a dialog box called Table structure dialog box Enter the name of the table being created in the table name textbox. And to add fields to the table follow the steps given in the next section.

Adding Fields

In the table structure dialog box click the Add field button to display the Add field dialog box. It appears as in figure 9.4



Add Field Dialog Box Options

The options of the Add field dialog box are given below.

Name

Allows to type the name of the field being added.

Ordinal position

Allows to determine the relative position of the field.

Type

Lets to determine the operational or data type of the field such integer, text etc.

Validation text

Lets to add a text of the message that the application displays if a user tries to enter an **invalid value for a field.**

Size

Lets to determine the maximum size, in bytes, of the field.

- Fixed field – If selected creates a field with a fixed size.
- Variable field – If selected, allows the user to modify the size of the field by dragging its borders.

Validation rule

Lets to determine what data is valid in a field as it is added. It is a string that describes a comparison in the form of an SQL WHERE clause, but without the WHERE reserved word. For example, if the value of the customer Age must be positive, then its validation Rule string will be ">0"

Default value

Lets to determine the default value for the field

Autoincrfield

Automatically gets value that is incremented by one from the last record when a new **record is added.**

Allowzerolength

Allows having a zero-length string as a valid setting.

Required

Indicates if the field requires a non NULL value.

Ok

Appends the current field definition to the current table.

Close

Closes the form when you are finished adding fields.

Opening the tables

To open a table select the table in the database window, right clicks on it, and chooses open, - or – Simply double clicks

This will open the table in a separate form (can be termed as table form) as shown in the following figure 9.5

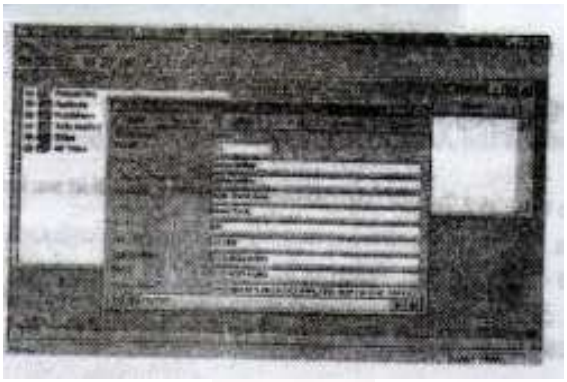
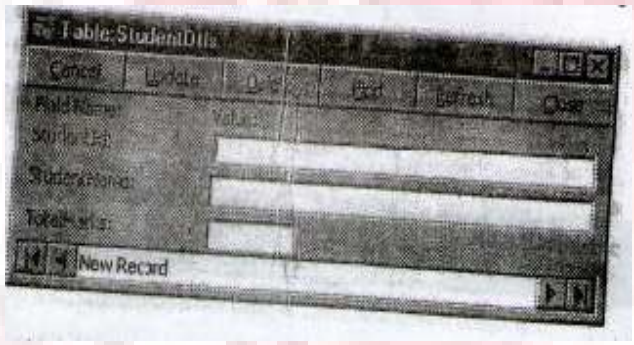


Figure: Opened Publishers table in VisData.

In this form records can be added, deleted and updated. The following steps how to add, delete, and Edit the records to the student Dtls table.

Manipulating recodes using Visual Data Master

Double click on the table name student Dtls in the database window to open it



As no record is added to the table, the student Dtl table form will appear in ready state to accept values for new record as given below figure 9.6

Figure 9.6 Student Dtls table form in ready state to accept data. Enter the following values in the corresponding text boxes.

Field name	**Value**
Student Id	1005
Student name	Malarvizhi
Total marks	450

Click update button to commit new record addition.

Now the cancel button will be replaced by add button.

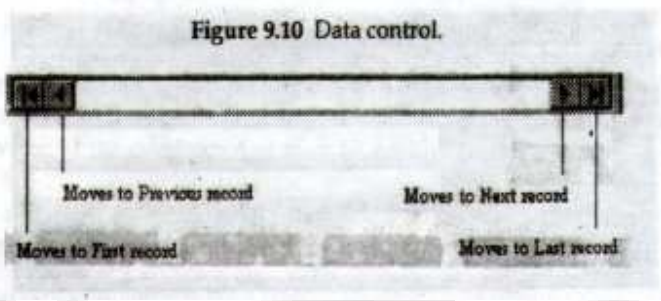
From now onwards to add recodes, Add button is to be clicked first, values to be entered next and finally update button is to be clicked. By following these steps enter the data given below to add five more records.

Student Id	Student Name	Total Marks
1001	Kumar	475
1002	Raju	480
1003	Shanmugam	466
1004	Vanisaraswathy	450
1005	Selva sudari	455

Note: while entering records instead of checking update button, cancel button can be clicked to cancel the record addition process.

At the bottom to the student Dtls, these are a control called Data control. It contains four buttons that enables to navigate the records such as moving to first record, moving to previous record, moving to next record and moving to last record. The following figure shows the data controls with its button's description.

Figure 9.10 Data control



To delete a record, use the data control to move to the record, which is to be deleted and click delete button. When VisData manager ask confirmation, click yes deletion. Clicking no will abort the deletion.

To edit a record, use data control to move to the required record, modify the record and either click update or any one of navigation button, VisData will ask whether to commit changes as shown below.

Figure: VisData asking confirmation to commit changes.

Click yes to commit changes or No to cancel the changes.

Introduction to data control

Many modern applications are concerned with the storage, organization and retrieval of data, to address the needs of the software developers creating these products, visual basic provides a rich set of data access features. Visual Basic provides many standard controls, such as text boxes and labels that can be bound to the Data control, in addition to the DBCombo, DBGrid and DBList. These controls are provided specifically for binding to a data control.

A data control binds data aware controls to Microsoft Access or to other ODBC data sources, enabling to move from record to record and to display and manipulate data from the records in bound controls. A bound control is a control that is assigned to a field in a database.

Most data access operations can be performed with a data control without writing any code. If a database and a record source a table, a view a SQL statement is set to a data control. Then it is automatically populated with data from that database when the form that has the data control, is activated.

The following figure shows the data control and DAO with bound controls.

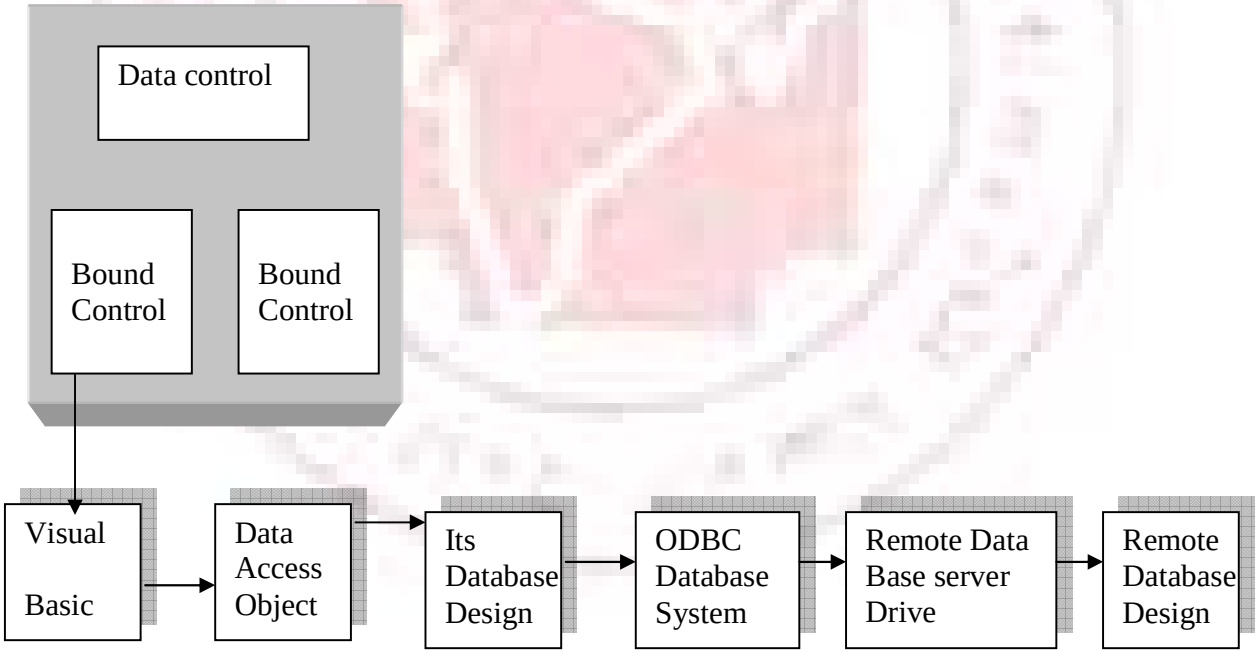


Figure: The data control, DAO, and bound controls

The intrinsic Data control on the Visual Basic toolbox, implements data access by using the Microsoft jet Database engine the same database engine that powers Microsoft Access. This technology gives seamless access to many standard database formats and allows to create data

aware applications without writing any code. The intrinsic Data control is best suited to smaller (desktop) database, such as Access and ISAM databases.

Data control's Icon

The intrinsic Data control can be used to create applications that display, edit, and update information from many types of existing databases, including Microsoft access, Btrieve, dbase, Microsoft FoxPro, and paradox.

Data control properties

Align Set or return a value that determines how the control is placed on the form.

Appearance Which determines whether or not to use 3D appearance.

Back color A numeric values that determines the control's background color

BOF Action Which determines how the data control reacts to EOF and BOF conditions.

Caption Set or return a sting representing the object's caption

Connect Set or return the type of database being used.

Database Which is read only property that exposes the data controls underlying database object.

Data field Which sets or return a value that binds the control to a field in the current record.

Drag Icon Set or return the icon to be used for drag operation

Drag Mode Set or return the type of drag mode available.

EOF Action Which determines action when the record set's EOF is reached

Exclusive This reads or set whether a database is opened for single user or multi-user access. Read and write at both design time and runtime.

Options Set or return one or more characteristics of the record set object

Recordset The data (set of recorded) available to the data control

Recordset type Sets or return the type of Recodest object of the data control

Record source Which specifies the part of the database seen by the control.

CHAPTER SEVEN
Data Access Object

Data Access objects (DAO) communicate with Microsoft Access and other ODBC complaint data sources through the JET engine. They provide properties and methods that allow to perform all the operation necessary to manage such a system, including the ability to do the following.

1. Create databases
2. Define tables, fields, and indexes
3. Establish relations between tables.
4. Navigate and query the database, and so on.

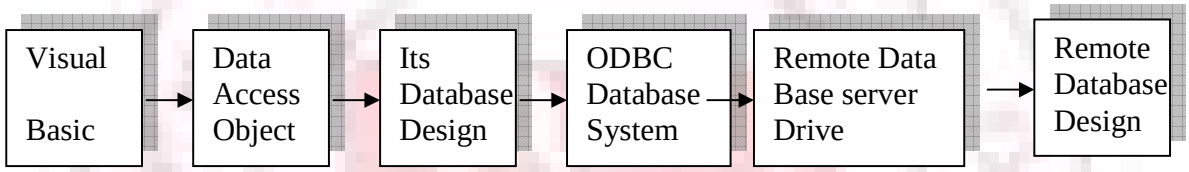


Figure: A typical remote data access using DAO and the JET engine.

The data Access objects (DAO) model is a collection of object classes that provide properties and methods for database programming. This model furnishes facilities for creating databases, defining tables, fields and indexes, establishing relations between tables navigation and querying the database, and so on.

The Microsoft jet database engine translates operations on data access objects into physical operations on he database files, handling all those mechanics of interfacing with the different supported databases. This approach simplifies access to the database and insulates from the underlying mechanics of retrieving and updating data. It affords great flexibility because the same objects, properties and methods can be used with a wide variety of supported database formats.

Through DAO and the JEF engine, Visual Basic recognizes three categories of database, and these are summarized below:

Category	Description
Visual Basic Database	Also called native databases, these databases Files use the same format as Microsoft Access. These databases are created and manipulated directly

	by the jet engine and provide maximum flexibility and speed.
External Databases	These are indexed sequential access method (ISAM) database in several formats, including Btrieve, dBase III, dBase IV, Microsoft Fox Pro version 2 dand 2.5 and paradox version s 3.x and 4
ODBC Databases	These include client/server databases that conform to the ODBC standard, such as Microsoft SQL server and Oracle.

The DAO programming model provides the following features:

1. Advanced result set management.
2. A universal programming model that can access any ODBC database regardless
3. Key, static, and forward only scrolling snapshot cursor implementation.
4. The ability to create updateable cursors against complex result sets created as a join product.
5. Universal error management

When to use DAO

1. DAO is the only data access technology that supports 16 bit operations. If an application must run within a 16 bit environment, then DAO is the only choice.
2. If the application must access both native Microsoft Jet and ODBC resources, DAO provides a consistent programming model.

Accessing a database using DAO

A DAO based application generally follows the logic given below while accessing a data source:

- **Create the work space** Define the user session, including user identification, password, and database type (such as Microsoft jet or ODBC)
- **Open the database** Specifies a connection string for a particular workspace object, with information such as data source name and database table
- **Open the recordset** Runs and SQL query (with or without parameters) and populates the recordset.
- **Use the recordset** The query result set is now available to the application. Depending on the cursor type, the row data can be browsed and changed.
- **Close the recordset** Drops the query results and closes the recordset.
- **Close the database** Closes the database and releases the connection.

The following code shows how to open the database students and to close it.

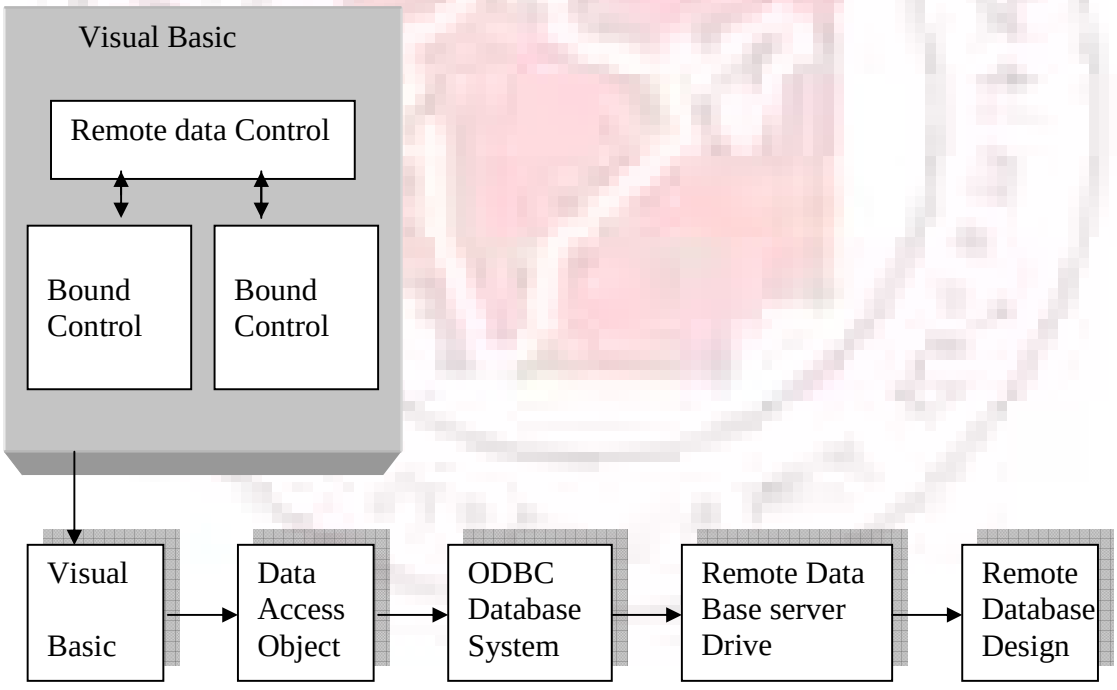
```
Dim ws as workspace
Dim db as database
‘Setting ws to default workspace
Set we= DBEngine. Workspace
```

```
'Setting db to students database using workspace's open database' method
Set db = ws Open database ("Students. Mdb")

'Some work with students database
'Closing students database

db close
'closing workspace
Ws.Close
```

About Remote Data Control (RDC)

The Remote Data Control (RDC) binds controls to an ODBC remote database. The remote Data Control is similar to the data control, except that it creates and manipulates RDO objects. RDO and RDC helps to access ODBC data sources through data aware or bound controls without going through the JET engine, resulting significantly higher performance and greater flexibility when accessing remote data sources.

Figure: The Remote Data control, RDO and bound controls.



The remote data control performs all operation on the current row. The remote Data Control automatically handles a number of contingencies, including empty result sets. Adding new rows, editing and updating existing rows; converting and displaying complex data types, and handling some types of errors.

The Remote Data control provides an interface between Remote Data Object (RDO) and data aware bound controls. With the remote Data it is possible to

- Establish a connection to a data source based on its properties
- Pas the current row's data to corresponding bound controls
- Permit the user to position the current row pointer.
- Pass any changes made to the bound controls back to the data source
- Remote Data Object (RDO) and collections provide a frame work for using code to create flagship interface to relational ODBC data sources.

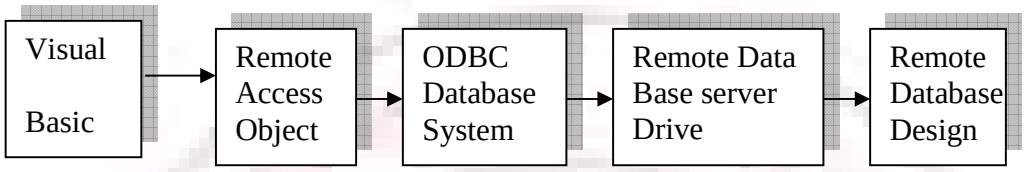


Figure: A typical remote data access using RDO.

RDO is specifically designed to deal with remote intelligent data sources such as SQL server and Oracle. RDO can execute ordinary table based queries, but it is especially good at building and executing queries against stored procedures.

The following advantages are associated with the RDO programming model:

- It is an universal programming model that can access any 32 bit level ODBC data source
- All remote databases such as SQL server and Oracle can virtually be accessed.
- Key set, static, dynamic and forward only cursors are implemented on the sever side.
- It gives the ability to create standalone rdo query and rdo connection objects and to associate queries with connections at design time
- It provides advanced result set management, including the ability to limit the number of retuned rows, and to handle the input, output and return value arguments of stored procedures.

Active X Data Objects (ADO)

It is emerging as another data access alternative, which may replace the need for other interfaces. ADO enables to write a Client application to access and manipulate data in a database server through a provider. ADO's primary benefits are ease of use, high speed, low memory overhead, and small disk footprint.

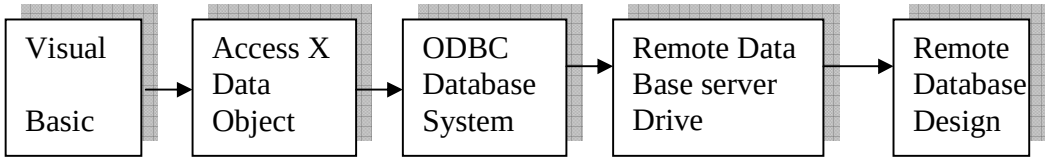


Figure: An ADO connection to a remote data source.

In ADO, the object hierarchy is de emphasized… unlike data access objects (DAO) and remote data objects (RDO), ADO allows creating objects independently; no longer having to navigate through a hierarchy to create objects. The ADO model also results in fewer objects, which allows for a smaller working set.

ADO is designed to eventually negate the need for all other interfaces. ADO is to specifically designed for ISAM or relational database access, but as an object interface to any data source. ADO is built around a set of core functions that all data sources are expected to implement.

The ADO programming model provides the following features:
Advanced record set cache management.
Different cursor types, including the potential for support of back end specific cursors.
Independently created objects.
Support for stored procedures with in/out parameter and return values.
Support for limits on the number of returned rows and other query goals.

Visual Basic also includes ADO Data control (ADODC), which works in the similar manner to the remote Data control, except that is uses OLE DB provider instead of ODBC driver.

Advantage of Active X Data Objects

Flat object models: ADO does not impose hierarchical object creation. That is to create a recordset object in DAO, a workspace object and database objects are to be created first. Then alone recordset object can be created. In ADO either this kind of hierarchical creation can be followed or its objects can be crated independently.

Less number of objects: ADO does not have large amount of objects like DABO and RDO, which lead difficulty in coding. ADO objects set is less than DAO and RDO, but more powerful.

Direct assignment Data bound controls: To fill data bound controls its data sources property can be set to a recordset object instead of setting to a data control (explained later in this chapter)

Enhanced performance by recordset object: Indexed fields greatly enhance the performance of the Recordset object using find methods, and sort and filter properties.

Recordset persistence: Recordset data can be saved in a file. Later this persisted file can be used to recreate the recordset object.

The ADO Object model

The ADO object model's hierarchy is given below.

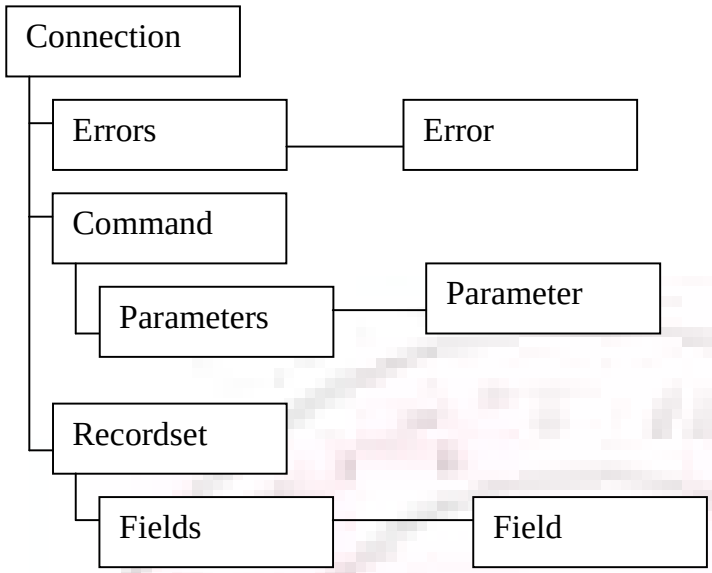
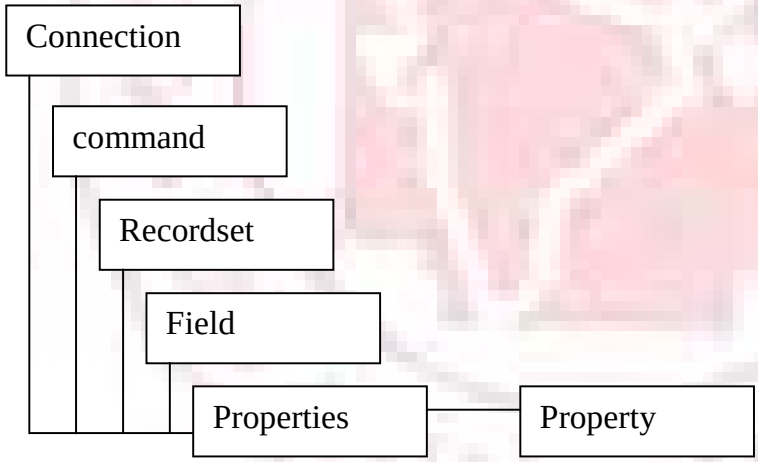


Figure An ADO object model.

Each of the connection, command, recordset and field objects also has a properties collection.



ADO Data Control

The ADO data control user Microsoft ActiveX Data Object (ADO) to quickly create connections between data bound controls and data providers (any controls that feature a Data Source property). Data providers can be any source written to the OLE DB specifications.

Creating a sample data base application with ADO data control

This section describes steps involved in creating a sample data base application with ADO data control. The behavior of this application will be similar to those applications, which are created using data control and RDC in the previous chapters. But the different lies in the design process. The following steps explain to design a simple application with the ADO data control.

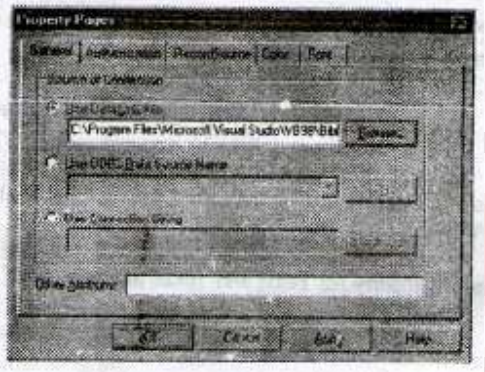
To create a simple database application that uses ADO data control

Draw an ADO Data Control on a form (The Icon's tooltip is "ADODC")

If the control is not available in the toolbox, press CTRL + T to display the components dialog box. In the components dialog, click Microsoft ADO data control.

On the toolbox, click the ADO data control to select it. Then press F4 to display the properties window.

In the properties window, click connection string to display the connection string dialog box.



Creating an ADO database with a Front End

Using the ADO data control in your applications is no more difficult than using the standard DAO Data control. In the following example, you will create a sample front end for an ADO database. Although this application is very simplistic, it does illustrate how easy it is to create an application using ADO. To create the front end for the Database, perform the following steps.

1. confirm that you have a valid. MDL files it you will be using an OLE DB Data source.
2. start a new project. On the form, place an ADO control. If the ADO control. Is not available in your toolbox, you can add it form the components dialog box and select the Microsoft ADO data control option shown in figure.

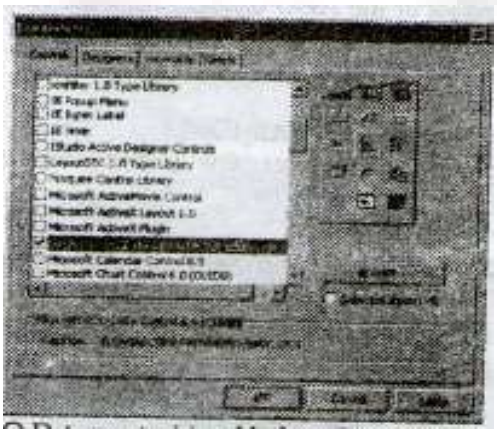


Figure: The ADO Data control is added to the toolbox after the Components Dialog box is closed

3. In the properties window of the ADO control, click the ellipsis button on the connection string property to display the connection string dialog box. This is shown in figure 14.2

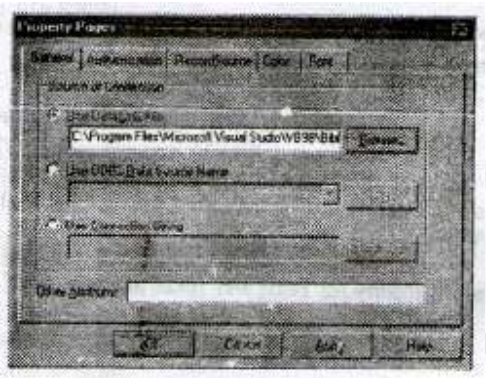


Figure: Property pages allow you to enter the ODBC source and connecting string

4. Select the General tab. If you have created a Data link file (MDL), select the use data link file check box and click the browse button to locate the MDL file. Alternatively, you could select either the use ODBC Data source name or use connection sting check boxes to create a connection to your database.
5. In the properties window of the ADO control, click the ellipsis button to display the record source dialog box. In the command Text (SQL) text box, enter the following SQL statement.

SELECT *FROM Authors WHERE Au_ID=72

This set the record source property to a SQL statement. This is shown in

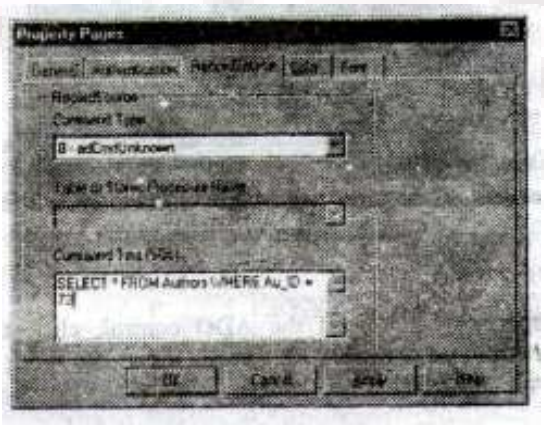


Figure- SQL statements can also be enter via the Record Source tab on the Property Pages dialog box

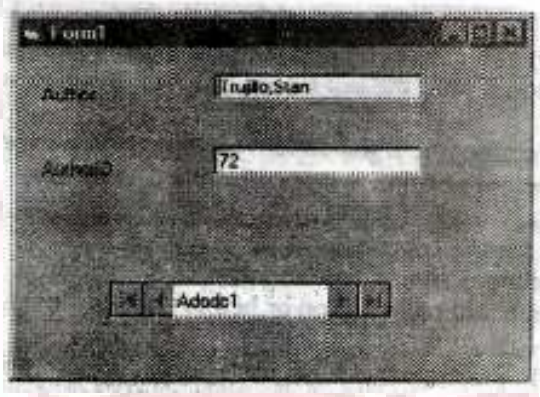
6. place two labels and two text box on the form. The captions of the text boxes should be Author and Author ID

7. Set the Data source property for Text1 and Text2 to the name of the ADO control on the form.

8. In the property window of Text1, set the data field property to author.

9. Set the data field property of Text2 to Au_ID

10. Press F5 to run the application. The completed form should resemble the one in figure



The ADO data control can be major resource hog. The ADO control will automatically use at least two connections for the first ADO control on a form. Additional ADO controls on the same form require one connection each.

An alternative to using the ADO data control is to set the properties programmatically. The sample code segment is listed below.

Example

```
Private Sub Form_Load ()
    With ADODC1
        Connecting string = “driver = {SQL server};” &_
        “Sever=orion; uid=sa;pwd;database=pubs”
        Record source – “Select *From Authors where Au_ID = 72”
    End with
    Set Text1.Datasource = ADODC1
    Text1.Dtafield = “Author”
End sub
```

Connecting to a database

ADO can be used to connect with a OLE DB data source and non data base types also.
A sample code is given to connect with SQL sever database

```
    Dim con as ADODB. CONNECTION
```

```
Dim cmd as string
Cmd = “provider = SQLOLEDB.1;”
```

BOF and EOF properties

- **BOF (Beginning Of File):** This property returns a Boolean value that indicates whether the current record position is before the first record in a Recordset object.

- **EOF (End Of File):** This property returns a Boolean value that indicates whether the current record position is after the last record in a Recordset object.

- The BOF property returns true if the current record position is before the first record and False if the current record position is on or before the last record.

- The EOF property returns true if the current record position is after the last record and False if the current record position is on or before the last record.

The BOF and EOF properties can be used to determine whether a recordset object contains records or whether the cursor has gone beyond the limits of a recordset object when one of a move method is used to move the record pointer from record to record.

If both the BOF and EOF property is True, it means there is no current record. If an attempt is made to move beyond the limits of a recordset object then an error will occur. So these properties can be used in conjunction with the move methods. For example, to prevent he move operation beyond the last record the following code may be used.

```
rs.MoveNext
If rs. EOF then
        MsgBox “End of the file!, ponter being moved to
        last_record.”
Rs.MoveLast
/End if
```

At the end of the record set rs, its EOF property is set to True. So, as the if condition satisfies the move next method will be roll backed. Similarly the BOF property can be used with the move previous method.

CHAPTER EIGHT
MULTIPLE DOCUMENT INTERFACE

A multiple document interface is used for opening many windows at the same time. All the document windows are contained in a parent window, which provides a workspace in the application. Visual Basic applications can have only one MDI form, which contains all the child forms. A child form is an ordinary form that has its child property set to True. Child forms are displayed within the internal area of a MDI form at run time.

Multiple Document Interface

Mainly there are two styles of interfaces in the windows based application: the single-document interface (SDI) and the multiple – document interface (MDI). An example of the SDI interface is the WordPad applications include with Microsoft windows. In WordPad, only a single document may be open and in order to open another, the currently opened. Document must be closed. The figure dhows the WordPad application, which is a SDI application.

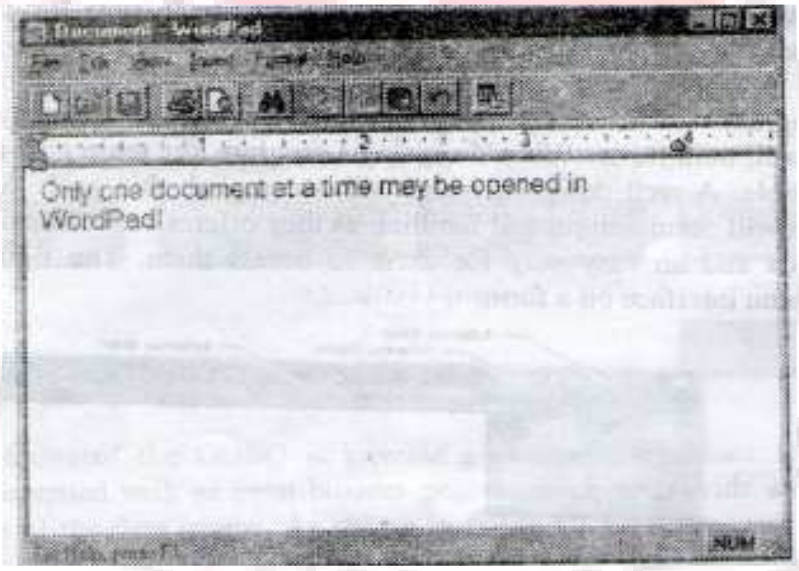


Figure: WordPad , a single –document interface (SDI) application.

Applications such as Microsoft excel and Microsoft world for windows are MDI interfaces; they allow displaying multiple documents at the same time, with each documents displayed in its own window. A MDI application can be recognized by the inclusion of a window menu item with submenus for switching between windows or documents. The figure shows the excel, which is a MDI application.

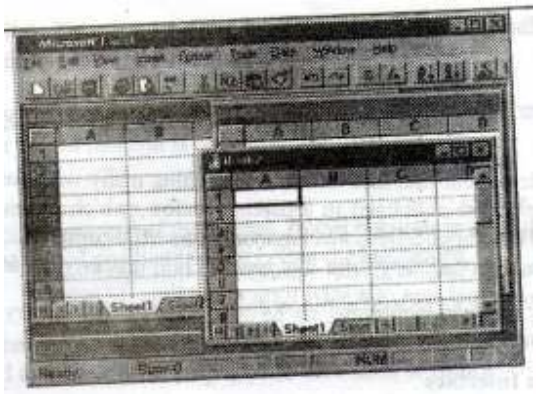


Figure: Microsoft Excel, a multiple-document interface (MDI) application

In multiple document interfaces, documents or child windows are contained in a parent window, which provides a workspace for all the child windows in the application. For example, Microsoft excel allows to create and display multiple document windows of different types. Each individual window is confined to the area of the excel parent window. When the excel is minimized, all the document windows are minimized as well' only the parent window's icon appears in the task bar.

Menu

Any program with more than a few simple functions or features can benefit from the additional of a well built menu. When designing a program, the main goal is to make its features as easy as possible. A well designed menu will accomplish this goal. With the help of good menu, progr4mms will seem natural and familiar, as they offer as convenient and consistent way to group commands and an easy way for users to access them. The figure 5.3 illustrates the elements of a menu interface on a form.

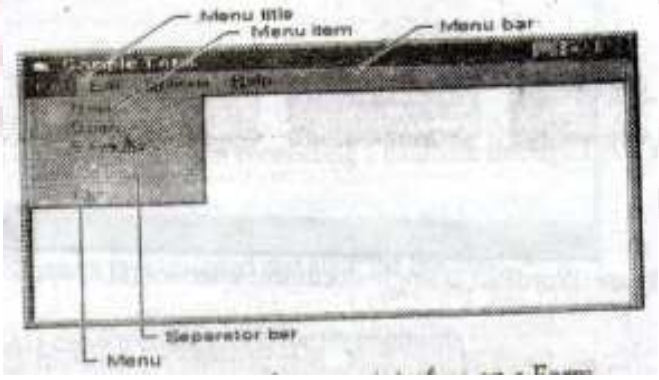


Figure: the elements of a menu interface on a Form.

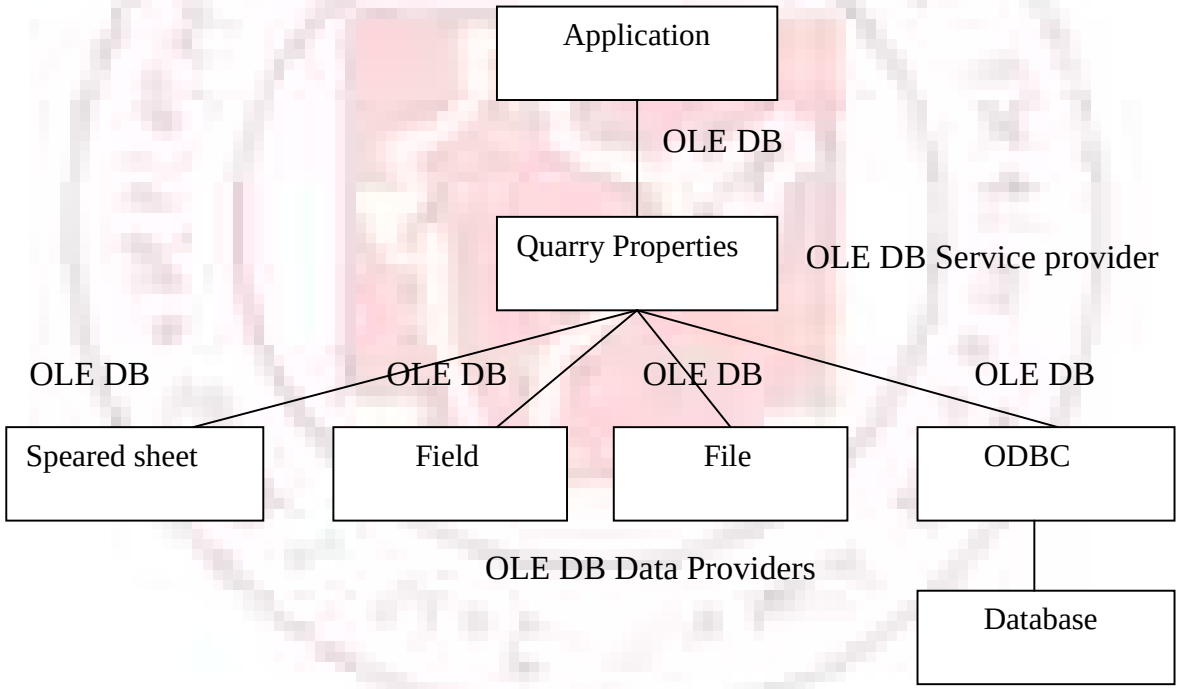
The menu bar appears immediately below the title bar on the form and contains one or more menu titles. When a menu title is clicked (such as File), a menu containing a list of menu items drops down. Menu items can included commands (such as new and exit), separator bars, and submenu titles. To make application easier to use, menu items must be grouped according to

their function. In the above, for example, the file related commands new, open and save as are all found on the file menu.

Dynamic Data Exchange (DDE)

DDE is the basic foundation for interposes communication among Microsoft windows applications. In a DDE conversation, the application creates the link is known as the destination application, and the application that responds in known as a source application. (these used to be referred to as client and sever respectively). Any application that supports DDE can serve as either a source or a destination. One application can serve as both destination and a source with several other applications at the same time. Only one active DDE link should be established between a Visual Basic control or form and another application. For example, a link between a Microsoft excel spreadsheet cell and a text control is acceptable as long the parent form does not already have link with the same cell. You might cause an infinite loop of updates if there were more than one link between the same two applications.

Object Linking And Embedding (OLE)

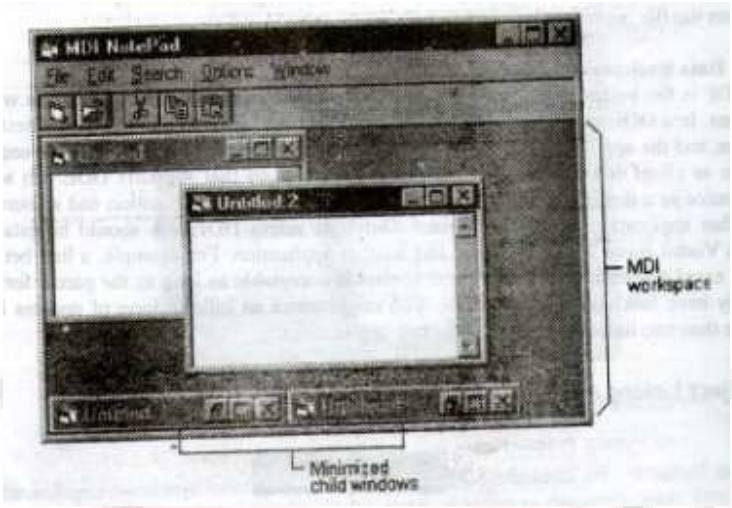


The primary focus of the ODBC is providing a consistent interface to database data sources. OLEDB is designed with an even broader goal in mind; to provide a methodology to access data regardless of the data source. As shown in figure OLEDB becomes the data access bridge for documents, e-mail systems, file systems, spreadsheets, and database sources using IDBC drivers.

Introduction to MDI form

In Visual Basic a child form is an ordinary form that has it MDI child property set to true. An application can include many MDI child forms of similar or different types. At run time,

child forms are displayed within the workspace of the MDI parent form (the area inside he form's borders and below the title and menu bars). When a child form is minimized, its icon appears within the workspace of the MDI form instead of on the task bar as in figure.



An MDI application can also include standard, now MDI forms that are not contained in the MDI form. A typical use of a standard form in a MDI application is to display a modal dialog box.

An MDI form is similar to an ordinary form with one restriction. Intrinsic controls can't be placed on a MDI form unless they have he Align property (such as a picture box control) or invisible interface (such as a timer control). More over an application can have only one MDI form. The MDI form's main property and method are given below:

Property	**Description**
Active form	it specifies the active MDI child form.

Method	**Description**
Arrange	arrange the windows or icons within a MDI form object

Syntax

MDI form. Arrange arrangement

A value or constant that specifies how to arrange windows or icons on a MDI form object, as described in Table.

Constant	Value	Description
Vb Cascade	2	Cascades all non minimized MDI child form
Vb title	1	Tiles all non minimized MDI child forms horizontally

Horizontal		
Vb title Vertical	2	Tiles all non minimized MDI child forms vertically
Vb arrange Icon	3	Arrange icons for minimized MDI child forms

Example

To cascade the child forms or the MDI form 1 the following statement is used. MDI form1. Arrange Vb cascade

MDI Child Forms properties

MDI form and all of its child forms take on special characteristics:

- All child forms are displayed within the MDI form's workspace. The user can move and size child forms like any other form; however, they are restricted to this workspace
- When a child form is minimized, its icon appears on ht MDI form instead of the taskbar. When the MDI form is minimized, the MDI form and all of its child forms are represented by a single icon. When the MDI form is restored, the MDI form and all the child forms are displayed in the same state they were in before being minimized.
- When a child form is maximized, its captions combined with the caption of the MDI form and is displayed in the MDI form's title bar (see figure 5.5)
- The active child form's menus (if any) are displayed on the MDI form's menu bar, not on the child form.

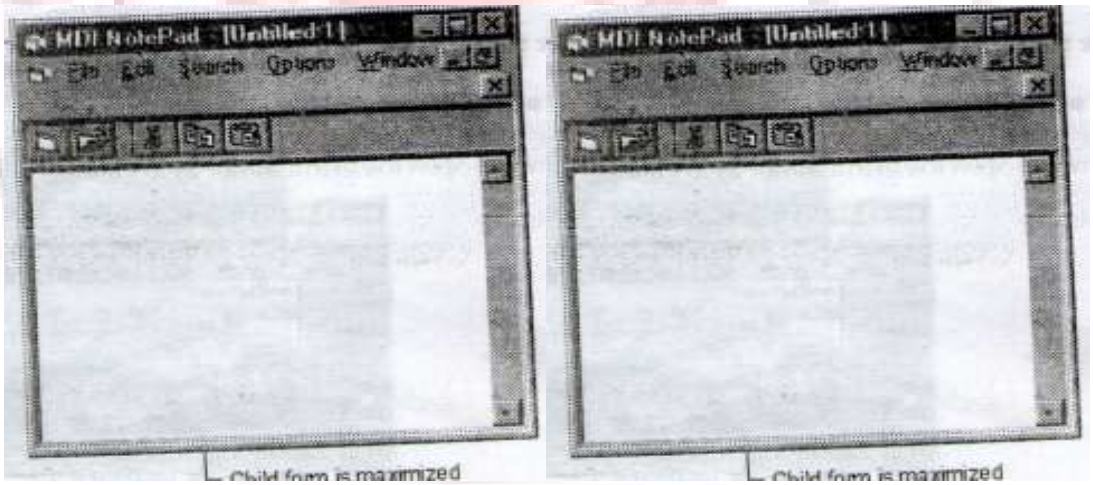


Figure: A child form caption combined with the caption of an MDI form.

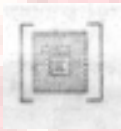
CHAPTER NINE
MENUS

Menus are one of the most common and characteristic elements of the windows user interface. Even in the old days of character based displays, menus were used to display methodically organized choices and guide the user through an application. Despite the visually rich interfaces window application and the many alternatives, menus are still the most popular menus of organizing a large number of options. Manu applications duplicate some or all of their menus in the form of icons on a toolbar, but the menu is a standard fixture of a form. You cane turn the toolbar on and off, but not the menus.

Menu Editor

Visual Basic provides menu editor, which helps to create new menus, add new commands to existing menus, and change and delete existing menus. Menus can be attached only to forms, and you design them with the menu editor. To see how the menu editor works, start a new standard EXE project, and when form 1 appears in the design window, choose Tools> menu editor to open the menu editor. Alternatively you can click them menu editor button on the toolbar

- Form the Tools menu chooses Menu Editor.



Click the Menu Editor icon on the tool bar

This will open the menu Editor as in figure.

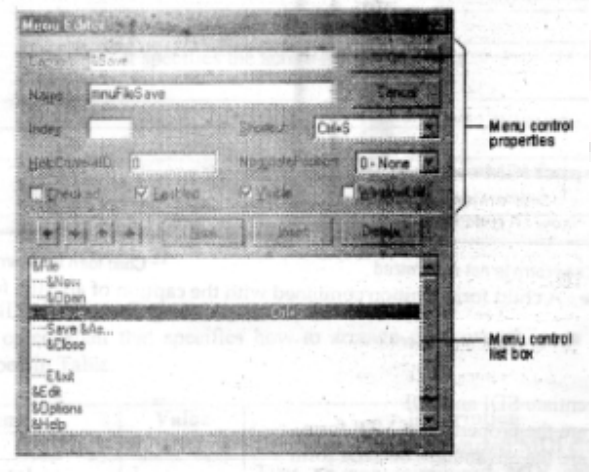


Figure: The Menu Editor.

While most menu control properties can be set using the menu editor, all menu properties are available in the properties window. The two most important properties for menu controls are.

- **Name** - This is the name used to refer the menu control form code
- **Caption** - This is the text that appears on the control
- **Index** - This is to create an array of menu commands
- **Checked** - Checked to indicate that they are on or unchecked to Indicate that they are off.
- **Enabled** - This is to set the initial status of a command by checking Or clearing the enable box.
- **Visible** -This is tom make a command invisible.
- **Window** -This is used to maintain a list of all open windows.

Menu control arrays

In the menu editor window add a menu option and set its index property to 0. You can then add commands with the same name and consecutive index values. At design time, you don't have to add more than one option. One command with its index property set to 0 is adequate to create the menu control array. You can use this array's name and an index value to add new options at runtime.

Figure show the RT menu application, which demonstrates how to add to and remove items from a menu at runtime.

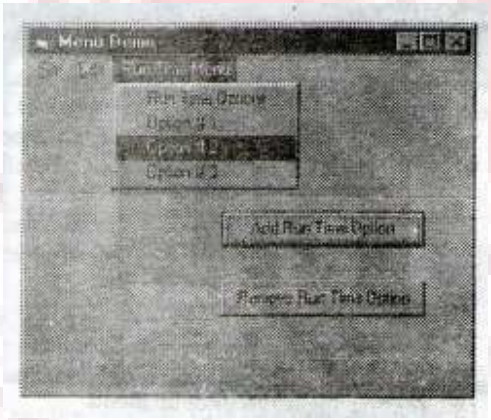


Figure: The RT Menu Application

Initially, the form's menu contains the following items:

File

Open
Save
Exit

Edit

Copy
Cut
Paste

Run time menu

Commands grouped in submenus are intended form the left. The last command's name is Run time options and its index is 0. Once a command option is specified as a control array, you can easily add commands to the runtime options array with the load method. After all, menu commands are very similar to controls.

The two buttons at the bottom of the form add commands to and remove commands from the runtime menu. Each menu command is appended at the end of the menu, and the commands are removed from the bottom of the menu (the most recently added commands)

Pop up menu

A pop up menu is a floating menu that is displayed over a form. Independent of the menu bar. The items displayed on the pop-up menu depend on where the pointer is located when the right mouse button is pressed; therefore, pop-up menus are also called context menus. Pop-up menus provide an efficient method for accessing common, contextual commands. For example, if the right mouse button is clicked in a text box, a contextual menu would appear, as in figure. 16.4

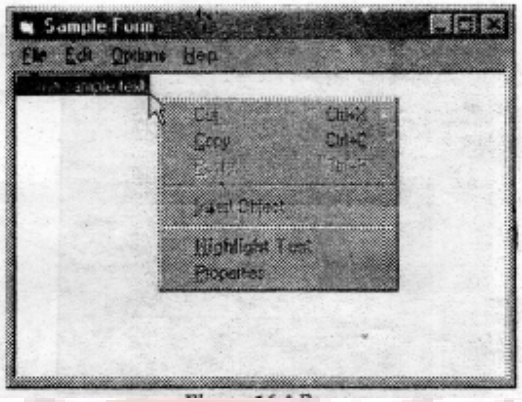


Figure: 16.4 Popup menu.

Any menu that has at least one menu item can be displayed at run time as a pop-up menu. Popup menu method is used to displayed pop-up menus, and its syntax is as follows:

Popup menu menuname [flags [x[y[bold command]]]

Only the first argument is required, ad it is the menu's name, as specified in the menu editor window. The other arguments are optional. The x and y arguments are the coordinates of a point on the form (or control) where the menu will be displayed. The flags argument defines the location behavior of a pop-up menu and can have one of the values. Shown below.

Location constants	**Description**
Vb popup menu left align	Default. The specified x location defines the left edge of the pop-up menu
Vb popup menu center align	The pop up menu is centered around the specified x location

Vb popup menu right align	The specified x location defines the right edge of the popup menu

The following code displays a popup menu named menu File with its top border centered on the form when the user clicks the form with the right mouse button.

```
Private Sub Form_Mouse Up (bottom as integer, shift As_
            integer, X as single, Y as single)

If button = 2 then ‘check if right mouse button was clicked
 popup menu mnuEdit, vbpopup menu center align x, y

End if
End sub
```

Note: To create a menu that will not display on the menu bar, its top-level menu item’s visible property is to unchecked design time. When Visual Basic displays it as a pop-up menu, its Visible property is ignored.

The bold command argument

The bold command argument is used to specify the name of a menu control in the displayed pop-up menu, to appear in bold to emphasize the most frequently used menu item. For example, assume that there is a menu named as mnu menu and one of its items is explore name as item explore. The following popup menu statement.

```
Popupmenu mnu menu,,,, itm explore
```

Will display the popup menu as in Figure

Figure: a popup menu with a bold item

Dynamic Data Exchange And Object Linking And Embedding

An important feature of Microsoft windows operating system is its ability for applications to share information. OLE is a means of communication, which gives application the power to directly use and manipulate other windows applications. OLE is an important windows topic available in Visual Basic. Any object that supports OLE can be linked. OLE specification permits the user to link and embed objects and also edit the object with in the container application.

Programmers are not historians. They don't like to read lengthy histories about a specific technology; they just want to learn how to use it. But when talking about a specific technology as all encompassing as Active X, knowing a little bout its previous history is a must. Before you can understand where Active X is going, you first need to know where it came form.

Active X can trace it roots all the way back to Dynamic Data Exchange (DDE)a way of passing data and commands between applications. Thought DDE is still supported by Visual Basic to maintain backward compatibility, it is seldom used.

Out of DDE evolved OLE1.0 object linking embedding. True to its name OLE brought the ability to link objects to one another or to embed one object inside another. The classic example of using OLE is the work processor. Spreadsheet combination. A reference to a reference, the spreadsheet program that is used to edit the spreadsheet would be automatically loaded and changes could be made. In this way, the spreadsheet is linked to the word processing document. With OLE, the spreadsheet can also be embedded into the word processing document. A window that shows the contents of the spreadsheet would be contained within the document . change to the spreadsheet can then be made inside that window. The spreadsheet program is still used to make the changes, but this is transparent to the user. He or she never has to leave the world processor to make changes to the spreadsheet.

DDE and OLE 1.00 are technologies that were designed to change the way we use computers. They bridged the gab between application-centric use and data centric use. In application centric computing, to work on certain kind of file (such as word Processing document, a graphic image or a spreadsheet), you would call up whichever program was used to edit that kind of file. To work on a word processing document, you need only click on the document itself. The operating system then determines which application is associated with that file type and loads it, along with the document.

Although the concept of data centric computing sounds like it does nothing more than makes tings easier for the user, it goes far beyond that. It also facilitates the embedding of different object types within other objects. A web browser, for example, can display an HTML document that contains many different object types: images, sounds, videos, animation, and others. If the web browser does not natively support the object type, the external program (typically, a "plug-in" for Netscape browsers or an ActiveX control in internet explorer) that is associated with the object type can be used to display the object within the page. Again, this is transparent to the user.

Object Embedding

With this technique, you can insert an object from one application (the server application) into another application (the container application). The inserted object is a copy of the original and can be manipulated and stored separately and apart from the original object. For example, you can embed a range of cells from an excel worksheet in a worked document. To edit the cells you switch to excel by double clicking the embedded excel object if the container application supports in place editing, you will see the menus of the server application right in the container application.

Inserting a OLE object

An OLE object is an item that is exposed, or made available, by an OLE server application. OLE server applications expose different types (or classes) or objects. OLE objects are used in container applications. As figure illustrates, excel (the OLE server) can expose a worksheet (the OLE object) that can be inserted as it in a word document (the container application)

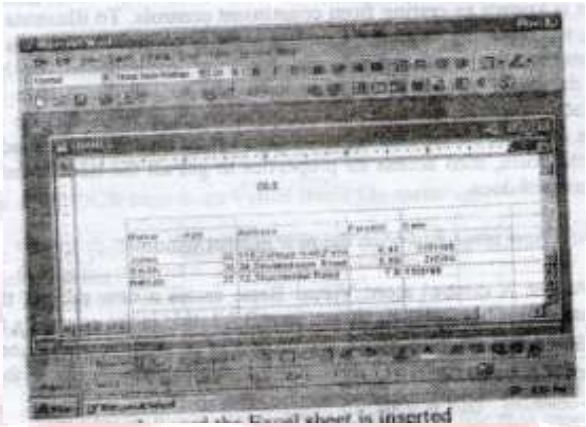


Figure: in word the Excel sheet is inserted

OLE Automation

The method allows you to programmatically manipulate objects exposed by another application form within your Visual Basic applications. It's also a standard that defines how code is shard between applications and how applications can be controlled form within other is applications. For example, when you copy a range of cells from an excel spreadsheet onto a word document, you embed the range of cells. With OLE automation, you application can request that excel perform some calculations and return the result to word. For example, you can pass a table to excel and request that excel manipulate the numeric data in ways that are not possible with work's tools and then return the processed data to word.

CHAPTER TEN
ACTIVE X Controls

Creating an active X control

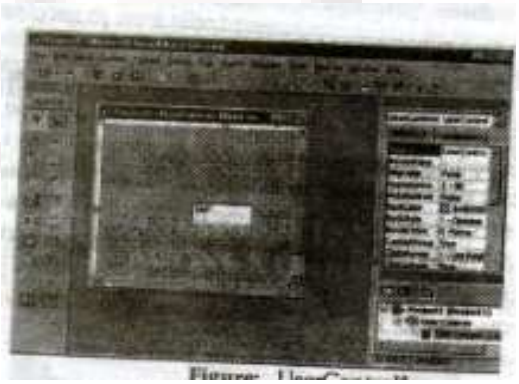
The simplest way to create the user interface of an Active X control is to use controls that already exist. This is known as creating from constituent controls. To illustrate the techniques for creating a control, we are going to create a simple security control that allows the user to enter a user ID and password. If you handle logins for your applications, or have to pass user information to a database, you at some time have had to ask for a user ID and password. Typically, you place a couple of labels and text boxes on a form and do a little bit of text processing. This is a simple enough task, but wouldn't it be nice if all you had to do was draw a single control on the form, then access its properties to get all the needed information? This is what the security control does.

1. Choose File> new project to open the new project window

2. Click the Active X control icon. Visual Basic creates a new project that contain a user control, named user control 1 as shown in gif17.2 this is the control's form on which the visible interface will be built. (once the project is created, you will need to set the name property of the control otherwise your control will go through life with the generic name user control (or 2 and so on)

3. Click the control from the toolbox and place it on the User Controll.

4. Set the basic properties of the control (name or the caption property for a label or command button)



Using the Active X control

After you have finished developing and resting your control, you can use the control in many programs. You can easily use the control in any other VBprogrma that you write. You can also use the control in any application that supports ActiveX controls. These applications

include Microsoft office, other development tools such as FoxPro, and even internet applications or web pages that support ActiveX. With this type of usage available, you can see why the capability to create ActiveX control is such a powerful feature in Visual Basic.

Compiling the control

The first step to being able to use your control in other application is to complete it. When you compile a control, you are creating an ICX file that can be accessed by other programs. Compiling your control involves the following steps:

1. Select the project in your project group that contains the control
2. select he make OCX item form Visual Basic file menu
3. From the make project dialog box, click the options button to set any necessary compilation options.
4. IN the make project dialog box set the name and path for the OCX file.
5. Click the OK button to start the compilation.
6. When the process is finished, your will have an OCX file that others can use to access your control.

Working with the control in Visual Basic

To use your ActiveX control in other Visual Basic program, you have tow options:

- Your can use the OCX that contains the control.
- You can ad the CTL control definition file to the project.

Using the OCX for the control is the preferred method of accession the control unless you are going to modify the control. Even then, you are better off modifying the original control project and recompiling the control than making modifications in a new project. To add your OCX control to your Visual Basic toolbox, open the components dialog box by pressing Ctrl +T. Then click the browse button in the dialog to bring up the open dialog that allows you specify the name and path of the control. When you select the control, it will appear in the components dialog and will be market to be added to your toolbox. After closing the components dialog, your control will appear in the toolbox and will be accessible to the current project, like any other control.

Using the control in a web page

Internet explores 3.0 and 4.0, as well as some other internet programs are capable of displaying and using Active X controls. The key to using the control is the use of the object tag

in the HTML code for the web page. The <object> tag specifies the following information about the control:

- The control's ID
- A name for the instance of the control
- The class ID of the control

This information identifies the control to your web browser and allows the control to be displayed if the browser supports Active X. The following code shows one instance of the HTML code needed to embed an ActiveX control in a page.

<object id = "Test1" name "Test1"

classid = "clsid: 21F43727 – A8FD- 11D1 – BCCD- 0000C051F659"

border = "0"

width = "188" height = "54"></object>

web page designer make it easy to add Active X controls to your pages. These designers handle all the object information for you so you don't have to remember all the details about class IDs and the other information.

Testing the active X control

Once you have created the ActiveX control, you need to test it make sure that all the procedures works as you think they should. You can use all the standard debugging techniques that are available in Visual Basic to debug your control. There are only a few different steps involved in setting up a test for your Active X control.

1. Add a standard project to your development environment to create a project group, which is the mechanism used for testing Active X control and servers

2. Close the user control window of your control project. If the user control (form) window is open, Visual Basic assumes that you are still working on the design of the control and will not allow you to create an instance of the control in another form

3. Create an instance of the control on you test form. Then you can test the properties of the control and write code to test its methods.

Active X documents

Active X documents are designed the same way as the Visual Basic forms are designed. That is, they can have intrinsic controls and Active XC controls as the normal Visual Basic forma have. They can also show message boxes and secondary forms. But, in contrast to he

independent appearance of the Visual Basic forms, Active X documents appear in containers, i.e internet browser windows such a Internet Explorer, Netscape navigator, etc. with Internet explorer 33.0 or later, active X document can be saved and ready but writing to and reading from the document's data file.

The base object of an Active X document is the user Document object that resembles a standard Visual Basic form object with some exceptions.

The user document object has most but not all, of the events that are found on a form object. The events present on a form that are not found on the user document include.

- Activate
- Deactivate
- Load
- Query Unload
- Unload

Event present on the user document, but not found on a form object include.

- Enter focus
- Exit focus
- Hide
- Init properties
- Read properties
- Scroll
- Show
- Write properties

Active X documents can be packaged in either in process or out or process components

Hyperlink objects

Hyperlink object behaves as a property of user documents. Using the properties and methods of the hyperlink object, the active X document can request a hyperlink aware container, such as Microsoft internet explorer, to jump to an another active X document.

This object has three methods:

- Navigate to
- Go back
- Go forward

Designing and compiling ActiveX Document

This section creates an Active X Document project called AcXDoc by demonstrating how to.

- Add user documents
- Ad forms
- Use hyperlink object
- Add menus to the user document

To create Act X Doc project

- Choose file | New project to open the new project dialog box.

- Double click the Active X document DLL icon. This will create an Active X document with default user document 1.

Active X Document DLL icon

The Active X document DLL project types will package the Active X documents as in process component. To package it as out of process component use Active X document EXE project type

- Choose project | Add user document to add a second user document
- Repeat the above step to add third use document.
- Name them ActX Doc 1, ActDoc2 ActDoc3 respectively.
- Choose project | Add form to add a form and name it as form address.

Compiling your documents

After you have finished testing and debugging you document, you are ready to compile the document for distribution. To start the compilation process, select the make register. Ext menu option from the file menu of Visual Basic. This will then open the make project dialog box, prompting you the supply a name and location for the EXE or DLL file. The name of the .vdb files is based on the name property of the user document object. This file is placed in the same folder that you specified for the EXE file. After you have compiled the ActiveX document, you can view it in any container program that handles ActiveX document, you can view it in any container program that handles ActiveX document. Internet Explorer and office binder are two such programs. At the time this book is being written, Netscape navigator 4.0 does not support ActiveX documents. It is already mentioned that ActiveX documents can be built as either in process or out of process component. In both cases, when the project is compiled, in addition to creating an exe. Or a .dll file. Visual Basic creates a Visual Basic document file (Figure 18.5), which has the extension. Vbd and it is placed in the dame directory as the compiled component (.exe or .dll)

The .vbd file is actually OLE structured storage-this basically means that data in the file can be accessed and manipulated via standard OLE interfaces. The internet explorer uses this interface. So only instead of specifying. dob files are mentioned while calling navigate to method to hyperlink object to open another document.

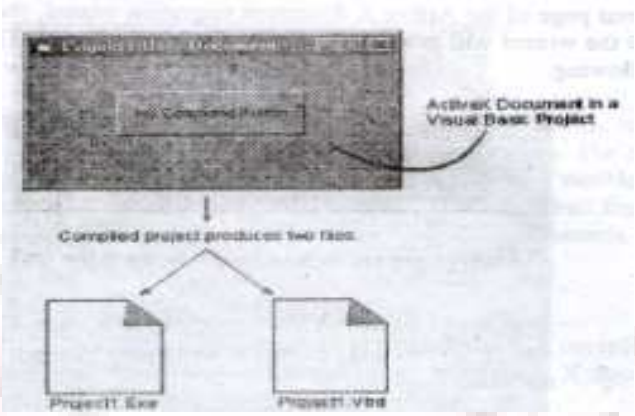


Figure: compiling user documents produces.exe (or.dll) and .vbd files

Active X document migration wizard

The active X document migration wizard must be available to the Visual Basic IDE. If you have not already done so, you can make the wizard available by selecting if from the add in manager dialog box, which is accessible by choosing the add in manager item from the add in menu. After you have added the wizard to your desktop, you can urn it by choosing the ActiveX document migration wizard item from the add in a menu.

To begin the conversion process, the project whose4 forms you want to convert must be open. After you have opened the project, launch the ActiveX document migration wizard. At this point you will be presented with a series of screens that will guide you through the conversion process. The first of these screens in introductory only, so click the next button to continue.

The second screen of the wizard, shown in fig 18.6 includes a list box form which you can select the forms that you would like to convert. Any forms that have been defined for this project will be displayed in this list box. To select forms, simply click in the checkbox next to the name of the form. When you have made all your selections, click the next button to continue.

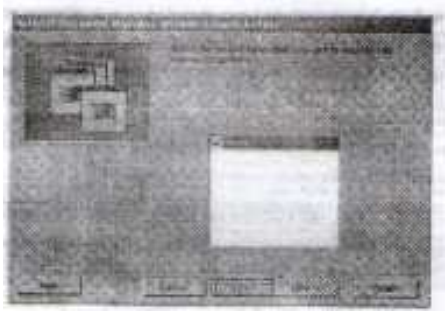


Figure: You can select multiple forms in the list box

The options page of the ActiveX document migration wizard, shown in Figure 18.7 lets you control how the wizard will process the forms you have selected. The three options enable you to do the following.

Figure: select the options that are appropriate to your needs

- Choose to comment out invalid code. This option will comment out statements such as load, unload, or end that are not supported by active X document.
- Remove original forms after conversion. This option will remove the forms form the current project after the conversion is made. Typically, you will not want to check this option because you will want your original project intact.
- Choose whether to convert your project to an ActiveX EXE or Active DLL project. The option defaults to Active EXE (Active XDLL are used for creating shared components rather than applications)

After making your choices, click the next button to continue. The last page of the wizard allows you to view a summary report that describes the wizard's actions. After making your selection, click the finish button to begin the conversion. The summary report is a valuable development tool because it provides you with information regarding the steps you need to complete. Figure 18.8 shown the summary report.

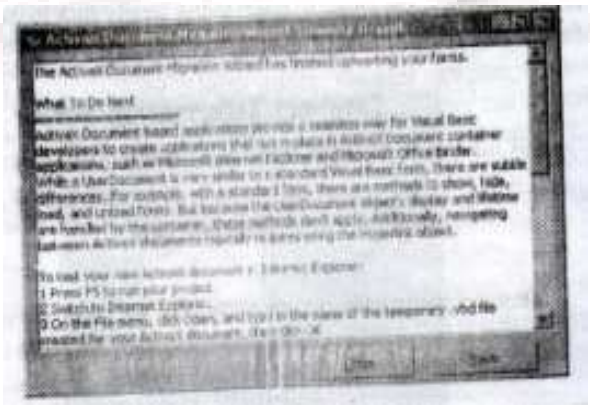


Figure: The Active X Document migration wizard uses a summary report.

Viewing the wizard's work

When the active X document migration wizard has finished its work, it place the newly created user document objects in the same project as the original forms. The document source files are stored in the same folder as the original form files and are given similar names, with the appropriate extension. For example, a form stored in the file frm Test .frm creates a user document stored in the file docTezt1. doc. As previously stated, the controls of the form are copied to the use document and their relative positions are preserved.

Also as stated previously, most of the code from your original form is copied over to the user document. "Invalid code is commented out and identified by the Active X document migration wizard.

CHAPTER ELEVEN
Internet information server

Microsoft internet information server is a network file and application server and is included with windows NT server. IIS supports Hypertext transfer protocol (HTTP), files transfer protocol (FTP) and gopher information protocols.

An ISS (Internet Information Server) application is a Visual Basic application that lives on a web server and responds to request from the browser. An IIS application uses HTML to present its user interface and uses compiled Visual Basic to process requests and response to events in the browser.

To the user, and IIS application appears to be mode up of a series of HTML pages. To the developer, an IIS application is mode up of a special type of object called a web class, that in turn contains a series of resources called web items. The web class acts as the central functional unit of the application, processing data from the browser and sending information to the users. You define a series of procedures that determine how the web class responds to these request. The web times are the HTML pages and other data the web class can send to the browser in response to a request.

Introduction to IIS & ASP

As developing DHTML documents are eased in Visual Basic through DHTML application project type, developing ASP documents are also eased in Visual Basic through IIS application project type. This section explain about IIS application project type and shows how to use it.

Web classes

A web class a Visual Basic component that resides on a web server and responds t input from the browse. When an IIS application project type is opened, Visual Basic opens a web designer that creates the web class. Web classes typically contain web item (HTML files) and code that deliver those webitems to a client.

A web class in an IIS application has an associated .asp (Active server pages) file that Visual Basic generates automatically during the compile process. The .asp file hosts the web class on the web server and launches its scripts for execution.

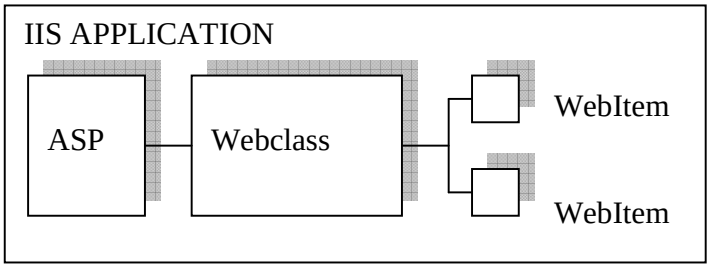


Figure: Relationship between Webclass and ASP file.

As shown in the figure, each webclass has its own SP. In turn, a webclass can have many webitems (HRML files) associated with it.

Advantage of IIS application

- Reduced cost of deployment per user. End users of an IIS application can run the application using only a browser; no special software needs to be installed on their computers for the application to work.
- A familiar development environment and model for Visual Basic programmer
- Access to a broad audience. IIS applications work with a wide variety of browsers and operating systems, so it can easily reach a side audience.
- It can send HTML pages either from templates or from code to the browser.

Note: templates are nothing but webitems that holds designed HTML pages. A temple can have only one HTML page.

In the chapter the discussion will go on application of IIS, ASP and DHTML then Debugging the IIS application and development process.

IIS applications VS. ASP Application

IIS applications bear a superficial resemblance to Active server page4s applications. Both types of application present dynamic websites and perform their processing on the server rather than the client. However, each ahs its unique advantages. Active Server pages are for script developers interested in authoring web pages, and offer the unique capability of intermingling script with HTML IIS applications are for Visual Basic developers building web based applications, rather than web pages. IIS application allow for complicated business processing and easy access from almost any browser or platform.

IIS application VS > DHTML applications

IIS applications are similar to DHTML applications in one way. Both arrange the contents of web document dynamically. However, there are some key differences between the two types of applications:

- Dependency – DHTML applications are dependent on internet explorer 4.0 or later. But end users of an IIS application do not need a specific operating system or browser.

- Location of processing – IIS applications are designed to perform most of their processing on the web server, but DHTML applications perform most of their processing on the browser machine.

Webclass designer

When a project of types IIS application is opened, Visual Basic presents webclass designer as in Figure:

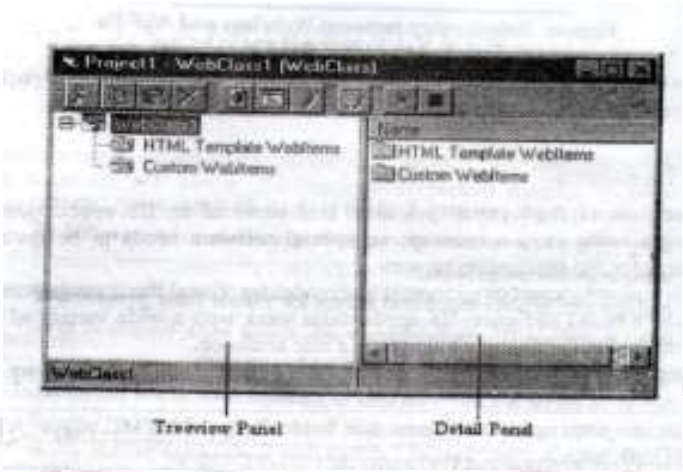


Figure: Webclass designer.

The treeview panel displays each webitem and connected event in the webclass. There are two kinds or webitems:

- HTML template webitems – HTML files that can be sent to the browser in response to a request
- Custom webitems – programmatic resources that creates HTML files on the fly.

The detail panel displayed information about the currently selected item in the treeview panel.

IIS application Development process

The process of creating an IIS application is similar to creating any other project in Visual Basic. The overall processes presented below.

The overall process for creating IIS application is:

1. Start a new project and select IIS application as the project type.
2. Save the project

Note: Unlike forms based Visual Basic applications, you must save an IIS application before you add HTML template web items to it

3. Add as many HTML temple webitems and custom webitems to the webclass needed.
4. Add any custom events to the project

5. Write code for all standard, template, and custom events in the project.

Note: make sure you write code for the start event, or your application will not run unless the user specifies a webitem in the base URL for more information see “Setting The start Event”

6. Add other code modules or web class objects to the project.

7. Test and debug the application by running the project and viewing the application in the browser. It is recommended that you test all browsers you plan to support before releasing the application to end users.

8. Complete the project

9. Deploy the application

Examples

This section shows how to create an ASP file using Visual Basic IIS application project type. Before creating the IIS application create an HTML file secret.htm whose code listing is given below.

```
<HTML
<BODY>
<CENTER>
<H2><U> now you are in the secreat site </U><H2>
</CENTER>
</H2>
</BODY>
</HTML>
```

This file will be used as a webitem in the IIS application project.

Creating a new IIS project

- Choose File |new project to open the new project dialog box
- Double click the IIS application icon to create an IIS application project

- Name the project a SIISPro

- In the project window double click the webclass1 to bring its designer front

- Set the properties of the webclass1 as given below:

Property	Value	Description
Name	wclsSecret	specifies the name of the webclass
Name in URL	key secret	specifies the name of the web class as it appears in the address that invoke the webclass. That is, its corresponding asp page will be created as key secret. Asp (key to secret) which is used in the address string.

- Save the project with the default names. (it is very important to store the project in a spedific folder. Then alone the HTML files that are address as webitems can be stored in the project folder).
- Right click on the HTML template webitems in the Treeview panel and choose add HTML template from the pop up menu that appears.
- In the dialog box that appears browse and select secret.Htm and click open. This will add a template under the HTML template webitems in the treeview panel as templae1. rename it as secret htm.
- Open the code window of the web class designer
- Replace the code of the web class start event procedure [webclass_start ()] with the following code.

```
Private sub Class_start ()
      Dim param
Set param = Request.Quary string

If param (“name”) = “Indian” then
      ‘rendering the secret site
      secretHtm. Write template

Else

      ‘Writing a reply to the user
      With Response
      .Write “<html>”
      .Write “<body>”
      .Write “You are not an authorized user to go to_ the secret site”
      .Write”</body>”
      .write”</html>”
End with
End if
End sub
```

This event will occur when the key Secret .ASP invoke this web class. This procedure checks whether visitor is an authorized user or not. It he is an authorized user, it sends the contents of the template file to the browser using the write Template method. Otherwise it creates an HTML page, which indicates that he is not an authorized user, and sends it to the browser.

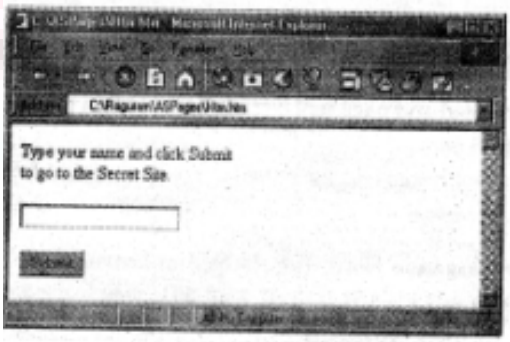
- Choose File | Make IIS pro.dll to compile the project. Visual Basic will create key secret.ASP file.

- Create an HTML HTM.htm file whose code listing is given below. This file will call the key secret.ASP file.

  ```
  <HTML>
  <BODY>
  Type your name and click submit <BR>
  To go to the secret site.
  <BR><BR>
  <FORM method =”Get”
  Action = http://127.0.0.1/ASP/keysecret.ASP>
  <INPUT type = text Name =”Name”><BR><BR>
  <INPUT type = submit = “submit” value=Submit>
  </FROM>
  </BODY>
  </HTML>
  ```

  The string /ASP specifies the virtual directory which is mapped to the project directory in which the key secret .ASP file is stored.

- Open the Htm.htm in the internet Explorer, which will resemble the figure:



- Type Indian in the text field and click the submit button. This will call the key secret. ASP file, which invokes the web class. The web class verifies the entered name. as it is Indian (the valid name), it will supply the secret.htm file, which is stored in the secret Htm template. The browser receives this file and renders it as in Figure:

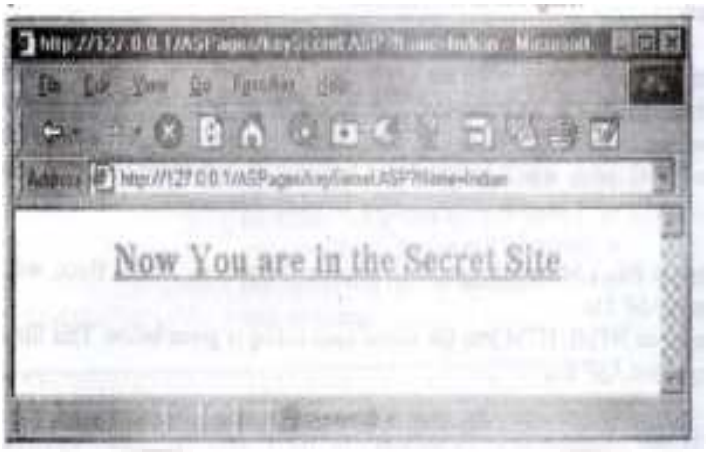


Figure: HTML pages sent by the ASP file when the user is an authorized person.

Is some other name is typed and the submit button is clicked, the web class crates HTML page on the fly indicating that the user is not an authorized user and sends it to the browser. The browser renders it an in Figure.

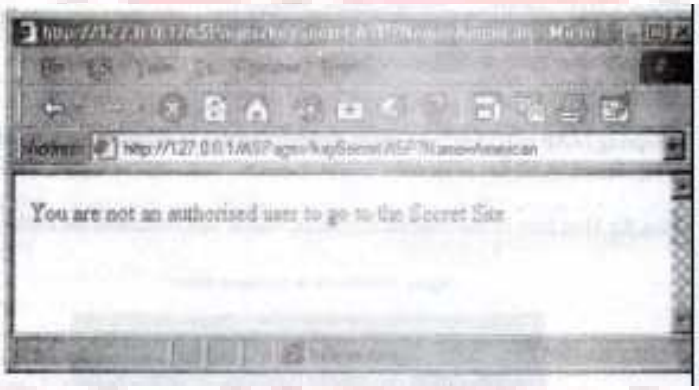


Figure – HTML created and page sent by the ASP file when the user is not an authorized person.

Debugging your IIS application

You debug an IIS application in the same way you do any other Visual Basic application- by entering run mode from Visual Basic. Visual Basic loads the webclass run time, creates the virtual root form which to run the .asp file for the application, if necessary, and launches the system's default browser with an HTTP reference to the .asp file, the .asp file, in turn, launches the webclass.

Note although you can view the hen files associated with your application in the browser by opening them from the browser's file menu, this is not debugging your application. You must use the start option from Visual Basic to enter debugging mode.

When the debug, you have the full Visual Basic development environment at your disposal. You can use all the tools available in Visual Basic break points, watch variables, debug statement, and so on to debug your project.

Visual Basic prompts you when you debug that it is going to create a virtual directory for your project. A virtual directory is a directory outside you web server's home directory that appears to browsers as a subdirectory of the home directory. Allows you to publish contents to the web from directories outside the home directory structure. You cannot change the location of the Virtual directory Visual Basic creates for the webclass.

The project properties dialog box's debugging panel settings determine whether the system waits for you to tell it what to do when you go into run mode or automatically starts the webclass you specify. When you chose to automatically start the webclass, Visual Basic launches Internet Explorer, navigates to the URL for you application, and fires the webclass Begin Request event.

Visual Basic deletes all temporary files when it comes out of run mode. In addition, it destroys the instance of the webclass designer and restarts the designer in design mode.

Error in webclasses

You can use Visual Basic error-handling features in you ISS applications to trap errors and take corrective action. When an error occurs, Visual Basic sets the various properties of the error object, Err, such as an error number or a description. You can use the Err object and its prop0erties in an error-handling routine so that you application can respond intelligently to an error situation.

In addition to standard error handling, IIS applications allow you to use two special features to handle errors:

- You can use the Trace method to debug your application of the production computer.
- You can use the Fatal Error Response event to respond to serious run time errors.

Using the trace method

You can use the trace method to help identify errors during the debug process and to track performance and statistical data. The trace method sends a specified string to the win 32 output Debug String API. The string can then be captured to a suitable debugging tool such as DBMON.

Using the trace method can allow you to debug on your production server computer and record useful information such as information you need.

Handling fatal errors

A fatal error on a webclass is one from which the application cannot recover or restore the appropriate webitem. For example, a fatal error might be an unhandled error within a webclass event, a structural error, or an unexpected error within the runtime DLL. Following such an error, the webclass run time fires the fatal response event. The application is terminated and the instance of the webclass is destroyed.

When a fatal error occurs, the application can write a message to the response object in the handler for the Fatal Error Response event. This message can be one that you write, or it can be the default message for the .asp file associated with the webclass. To write your own message, use the response object, then set the send default argument of the fatal error response event to false. To use the default error message, leave the send default argument set to true.

Note: the webclass run time provides an error property that is only available from within the fatal error response event. This property returns an object that describes the error that caused the webclass to terminate.

The webclass run time also logs fatal errors to the NT event log. On Windows 95 systems, the run time DLL creates a log file in the windows directory and logs the error there.

CHAPTER TWELVE
Internet concepts

If there is one technology that caught up literally overnight and has affected more users than any others, it is the web. The world wide web (WWW) is the set of all web sites and the documents they can provide to clients (Users). Visual Basic 6 has evolved to help the programmers to build web applications. There are two different types of web applications – Dynamic HTML (DHTML) applications and Microsoft Internet Information server (IIS) applications.

Internet concepts

This section briefs essential internet concepts as follows

Internet

The internet is a global network of computers that communicate using a common language. It is similar to the international telephone system – no one owns or controls the whole thing, but it is connected in a way that makes it work like one big network. The internet uses a common protocol to communicate TCP/IP (Transmission Control Protocol/ Internet protocol). TCP/IP is a simple protocol because it had to be implemented consistently on all computers and operating systems. Indeed, TCP/IP is a truly universal protocol, it's there when you need it and allows your computer to connect to any other computer on the Internet.

Each computer on the Internet has a unique address, for example, 193.25.84.100. Each number is a value in the range 0 through 255, which means the internet have more than 256*256*256*256, or approximately 4,000,000,000 computers. To accommodate a large number of users, internet service provider use a pool of addresses (since not all users connect at once, 256 addresses may accommodate 100users or more. It would be nice if we all had a unique IP address, like an e-mail address, but this is not possible, if we did, we could build wide area networks that span the globe easily. However, every time you connect to your internet service provider you get a different IP address.

Intranet

Unlike internet, which is a global network, it is a private network, but it uses the internet communication standards and tools to provide information to the restricted users. For example, a company may setup a web site that is accessible only to its employees who are geographically separated. On an intranet. You can exploit the web model to simple operations, without the security issues you face on the Internet or the limitation imposed by connection computers with modems.

An intranet is also a network of computers operating on the TCP/IP protocol, but it is not global. Intranets are restricted to the users of a corporation. A university, or some other

organization, and they are not accessible by the outside world. Unlike the world wide web, intranets don't have more than one server. This machine supplies all the documents required by the client. Many corporation use intranets to provide information to their employee, and they run another web site for external users. The reason for building corporate intranets to exploit the technology. They made the web so popular. So the main characteristic of a corporate intranet is not that it use TCP/IP, as much as HTTP (Hyper Text Transfer Protocol)

World Wide Web

The world wide web (WWW or simply the web) give a graphical, easy to navigate interface for looking at documents on the internet. These documents as well as the links between them, comprise a web of information.

Files or pages on the web are interconnected. This connection is made by means of special text or graphics called hyperlinks.

Pages can contain text, image, moves, and sounds just about anything. These pages can be located on computers anywhere in the world. When a connection to the internet is made, it means equal access is available to information world side.

The computers that host web sites are called server; their service is to provide the documents that the clients request. Clients are the seemingly endless numbers of personal computers connected to the internet. Web browser, such as Interne Explorer (explained in the next section), helps to exploit the web.

Internet Explorer

Microsoft Internet Explorer is a web browser. Just as Microsoft word is a tool to create and format documents, or Microsoft excel is a tool to create spreadsheets and perform calculations, internet explorer is a tool to navigate and access, or "browse", information on the web.

The internet Explorer toolbar provides a range of detailed functions and commands for managing the browser. The address bar below the toolbar displays the address of the current web page. Typing an address and hitting enter in it will enable to visit to the specified page. Clicking a hyperlink will also enable to jump to the new page. With Netscape dominating the web browser and server market, Microsoft, in the early 1996 introduced a new web browser and server to the internet community. Microsoft internet explorer is destined to be one of the most popular web browsers for several reasons. Not only is the browser easy to use and is supported by the world's largest software company, it also supports several other specific HTML extensions. These include the following.

- Background sounds played automatically when the web page is loaded
- In line animations AVI files instead of graphic images
- Marquees that scroll across the browser window.

Working to the Internet

For a computer to be a server on the internet, it must have two things: an address by which other computers can locate it and the capability to understand and process the various protocols. A server is assigned a unique numeric ID called an IP (Internet Protocol) address. As the numeric address is very difficult to remember their corresponding domain name is used, which are in friendlier format such as http://www.Microsoft .Com.

The software used on the physical computer to make it a server that can speak the protocols of the internet and respond accordingly is called internet server software or web server such as Microsoft Internet information server.

For a client computer to be able to communicate with a server on the internet, it must have a connection to the internet. They, when connected, it must have a way to contact and receive data from internet servers through the various protocols. The connection is accomplished via an internet service provider (ISP), such as VSNL, satyam online, etc. The tool to communicate to the server and receive the data retuned by the internet server is handled by the internet browser, such as Internet Explore. The Figure 5.1 depicts the working of the internet.

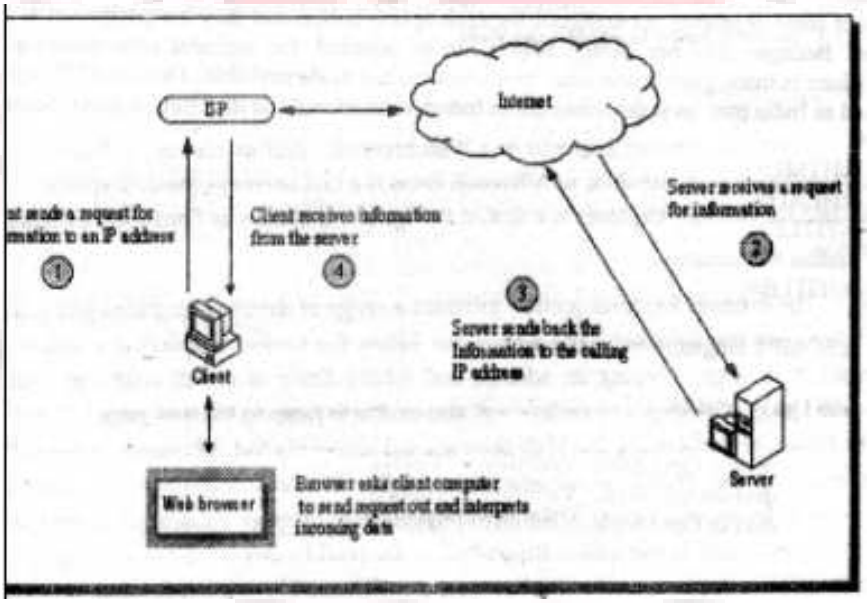


Figure: Working of Internet.

CHAPTER THIRTEEN
DHTML

This chapter discuss on HTML developing, its structure and its applications.

Developing an Dynamic HTML Document

This section develops a simple DHTML document, which has 5 major segments (designed using DIV tags). The first segment has one password box and a command button. This segment alone is visible when the document is loaded in browser and all other segments are invisible. When a user enters a password and clicks the command button, the script written for the command button's click event, verifies the validity of the entered password. It is valid the first segment disappears and second segment appears, which has three option 9radio) buttons. The other three segments are designed for each option button and they are positioned at a same. Location. Because only visible and other two segments are made invisible. This DHTML is named as India. Htm, as it describes about Indian monuments and its listing is given below:

```
<HTML>
<HEAD>
<TITLE>
Indian Monuments
</TITLE>

<SCRIPT language = "VBScript">

Sub Opt1_ onclick ()

        divFor Opt1.Style. Visibility = "Visible"
        divFor Opt2.Style. Visibility = "Hidden"
        divFor Opt3.Style. Visibility = "Hidden"

End Sub

Sub Opt2 _ Onclick ()

        divFor Opt1.Style. Visibility = "Visible"
        divFor Opt2.Style. Visibility = "Hidden"
        divFor Opt3.Style. Visibility = "Hidden"

End Sub

Sub Opt3 _ Onclick ()

        divFor Opt1.Style. Visibility = "Visible"
```

```
            divFor Opt2.Style. Visibility = “Hidden”
            divFor Opt3.Style. Visibility = “Hidden”

End Sub

    Sub cmdLogin_OnClick ()

If frm Login.txtPword.Value = “Indian” then

            Msgbox “You are logged in”
            divLogin.Style. Visibility = “Hidden”
            divOptions.Style. Visibility = “Visible”
            divFor Opt1.Style. Visibility = “Visible”

Else
            Msgbox “password incorrect”
End if
End Sub

</SCRITP>

</HEAD>

<BODY background = “bacjgrd.fig”>

<CENTRE>
<DIV id = Login>
<FORM id = frm login>
<LABEL> password Id = txt pword>

</LABEL>
<INPUT type = button id = cmd login value = “Login”>
</FORM>
</DIV>

<DIV id = div options Style = “Position: Absolute;

        Left;20 Top:90; Visibility: Hidden”>

Choose any one of monuments MBR>

To view its description
<DIV Style = “Position : Relative; Top:10; width:130; height:80;
        background- color: Yellow”>
<FORM id =frm Options>
```

```
<INPUT type = Radio name = Monument Id = Opt1 checked > taj
mahal <BR>
< INPUT type = Radio name = Monument Id = Opt2 checked > qutub minar <BR>
<INPUT type = Radio name = Monument Id = Opt3 checked > Fatehpur siki <BR>
</DIV>
</FORM>
</DIV>

<DIV Id = divFor Opt1 style = “Visibility : Hidden”>

<IMG Scr = “C\Ragruam\Images\Tajmahal.GIF”

Style = “Position : Absolute; Left 275; top: 85;

<DIV Style = “Position: Absolute; left 20 top: 240: width 450; background-color:yellow”>
one among the 7 world wonders. Built by Shajahan at the river bank of Yamuna.
</DIV>
</DIV>

<DIV Id = divFor Opt2 Style = “Visibility : Hidden”>
<IMG Scr = “C\Ragruam\Images\Qminar.Gid”
Style = “Position : Absolute; left 275; Top: 85”>
<DIV Style = “Position: Absolute; Left 20 top 240: width 450;
background color:yellow”>

Astonishing tower with height 242 feet. Started by Quthuputheen and successfully completed by
Eldhuthmish.
</DIV>
</DIV>

<DIV Id = divFor Opt3.Style = “Visibility : Hidden”>
<IMG Scr = “C\Ragruam\Images\FSikri.Gif”
Style = “Position : Absolute; left 275; Top: 85”>

<DIV Style = “Position: Absolute; Left 20 top 240: width 450;
background color:yellow”>
Instance for sculpting art. Nick named as love designed in stone. Built by great emperor Akbar.

</DIV>
</DIV>
</BODY>
</HTML>
```

When this HTML page is opened in the internet explorer it will prompt to enter the password as in figure.

Figure: India.htm – prompting the visitor to enter the password.

Enter the password Indian and click login button. Now the first segment in this document that contains the textfield button, will disappear and the other segments that describes about Indian monuments will appear as in Figure.

DHTML application in Visual Basic

So far, web pages are created in non visual environment (Notepad). Creating web pages in this method consumes much time and also positioning the controls and elements of web page at proper location will cause the developer's eye turn to red. As so much of tags are used even to create simple web pages, there, is a chance that few tags may be misspelled. And it is difficult to trace out the error caused due to this misspelling.

Visual Basic 6.0 includes new project type called DHTML application, which provides visual environment to rapidly develop the DHTML applications.

Web pages VS. Forms

DHTML applications are structured differently than forms based Visual Basic applications. In a DHTML application, the user interface consists of a series of HTML pages rather than forms. An HTML pages is like a form in that it contains all the Visual elements that makes up the application's user interface. Textbox, Image, radio button and check boxes can easily be placed on an HTML at desired locations.

An HTML page is stored in a htm file that is analogous to a frm file. The following table sums up the differences between forms based applications and web-based application.

	Form – based application	Web – based application
User interface	Visual Basic forms	HTML pages
File format	.frm files	.html or html files. Or generated from Visual Basic code
Creator	Developer	Web designer or developer
Run time	Visual Basic runtime DLL, msvbvm60.dll	Web browser with msvbvm60.dll

Table: Difference between forms based application and web based application.

24.4 Structure of DHTML application

DHTML applications are mode up of the following pieces:

- One or more HTML pages
- Visual Basic code that handles the events generated from the HTML pages.
- A project DLL that contains Visual Basic code is accessed by the run time components garneted automatically when the application is compiled.

When a DHTML project is compiled, all HTML pages are stored in separate .htm file. The code that handles the events generated from the HTML pages are compiled in to a .dll files. The id of the .dll file is automatically included in the .htm files. So whenever and event rises from a .htm file, the code for that event in the .dll files is executed by the browser automatically.

Advantages of Visual Basic DHTML applications

Visual Environment HTML pages can be created in the familiar Visual Basic IDE. As it is a visual environment, control can be placed in the HTML page at the desired locations easily. Debugging tools can also be used to fix the bugs in VBscripts.

Code security When scripts are embedded with an HTML page, anyone can access the page, read the script, and make changes to it. S Visual Basic compiles the code into .dll the code is not part of the HTML file itself, and so it can’t be tampered.

The DHTML page designer

When a project of type DHTML is opened, Visual Basic Presents DHTML page designer as in Figure 5.4 which enables to:

- Create a new HTML page, or edit the contents of an existing page.
- Determine which elements on the page re programmable and explore all dynamic elements on the page.

- Access the code editor window to write code for each programmable element of the page.

The following sections brief the different areas of the DHTML page designer.

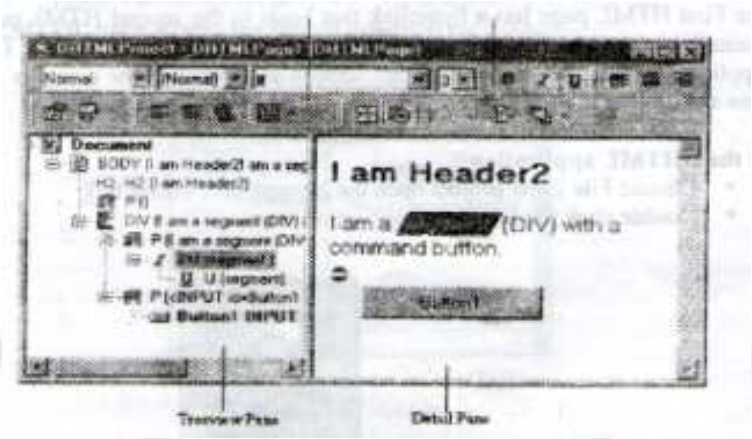


Figure: HTML Page Designer.

Tree view pane

Display a hierarchical representation of all of the elements. Within an HTML page. For each element, the page designer lists the ID (if one exists), the type of control and in some cases the beginning for the content for that element, Elements with IDs are indicated in bold. All elements in the tree view are children of the body element and the document object.

Detail pane

Present a drawing surface that enables to create a new page or edit the contents of an existing page. It provides a visual environment to add HTML to the page, position them, and set properties that control their physical appearance.

Format tool bar band

Contains buttons that are frequently used when formatting an HTML page in the designer. The items on this toolbar are available whenever a DIV element, or a hyperlink, or a piece for text, or the BODY or DOCUMENT object in the treeview or detail panes of the designer is selected.

When a toolbar button is clicked to carry out the action represented by the button, Visual Basic inserts the appropriate opening and closing HTML tags or tag attributes to the selection. These tags are not visible within the designer, but the result of formatting are rendered in the detail pane.

Element Toolbar band

Contains buttons that are fr4equently used when grouping the elements on the page such as by using DIV tag or working with tables. With the exception of the table operations icon, the

options on this toolbar band are available only when some elements are selected. The table operations icon is always available.

Note: The HTML files created using DHTML page designer are stored in the files the extension .dsr. the actual .htm files are created only when the application is compiled.

Creating an DHTML application using Visual Basic

This section shows how to create a simple DHTML application, which contains three HTML pages.

The First HTML page has a hyperlink that leads to the second HTML page. The second page contains a command whose clicking leads to the third page using code. This is not a full fledged application. But the purpose of this section is to show how creation of the DHTML applications is simpler than by using HTML tags.

To create the DHTML application

- Choose File |New project open the New project dialog box
- Double click the DHTML application icon.

DHTML application icon

This will create a new DHTML application with default DHTML page (DHTML page)

- Choose project | Add DHTML page to add the second page.
- Repeat the above step to add the third page.
- Name the project as seas, DHTML page1 as Arabic sea, DHTML page2 as Bay of Bengal, and DHTML page3 as Indian ocean using properties window.
- In the project explorer window double click as Arabic sea to bring its design window front.
- In the format band select header 1 for the HTML block elements combo box and enter “Arabic sea” in the details pane.
- Hit enter and type the following:
  Click me to go to bay of Bengal
- Select the word “me” and click the make selection into link icon in the element toolbar band.
- Right click on the word “me” in the popup menu that appears, choose properties. This will display the properties dialog box.
- Enter Bay of Bengal .htm in the link text box and leads to bay of Bengal in the popup text box as in figure 5.5 this text will appear in the browser window when the mouse pointer
- Moves over the hypertext “me” when it is clicked 4 Bay of Bengal .htm page will appear in the browser.

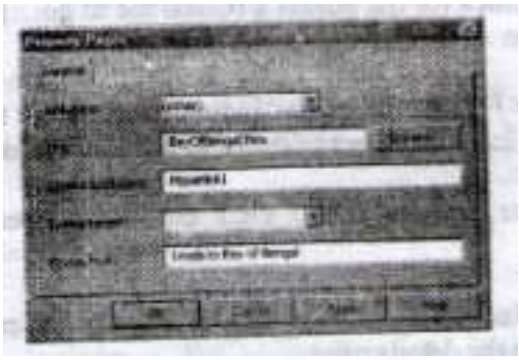


Figure: destination (through Link) and Popup text for the hypertext "*Me*"

The designed DHTML page Arabic sea should resemble the figure

Figure: Designed DHTML Page Arabic Sea.

- In the project explorer window double click the bay of Bengal to bring its design window front.
- In the Format band select Hander 1 from the HTML block elements combo box and enter "Bay of Bengal" in the details pane.
- Draw a button and set its caption as "Click me to go to Indian ocean"

The designed DHTML page Bay of Bengal should resemble the figure.

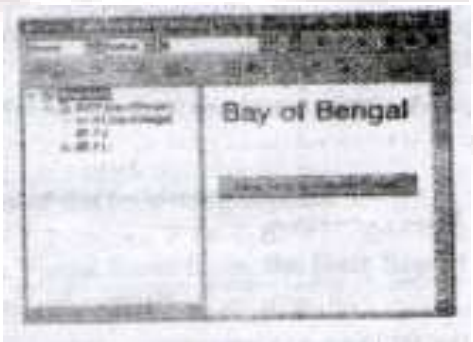


Figure: Designed DHTML Page Bay of Bengal.

- Write the following lines for the button's (Click me to go…) click event procedure base window. Location .href = "Indian ocean .htm"

The above line assign the HTML file name “Indian ocean .htm” to the href (Hyper line reference) property of the location object of the base windows to jump to that file.

- In the project explore window double click the Indian ocean to bring its design window front.

- In the format band select header 1 from the HTML block elements combo box and enter “Indian ocean’ in the details pane

- Save the project and choose file |make seas.dll

l

- When the Visual Basic prompts the .dll file name accept the default file name seas. Dell

- When the Visual Basic prompts file name for each DHTML files, enter them as Arabic sea. Htm, Bay of Bengal. Htm, and Indian ocean respectively

- Open the Internet explorer and in its address text box type Arabic sea, htm with full path and hit enter. This will open the Arabic sea. Htm file in the internet Explorer.

- Test the hyperlink and button to jump to their destinations.

CHAPTER FOURTEEN
Data report

Introduction to data report

Helps you design a report that displays fields and records from the underlying table or query.

A report is an effective way to present data in a printed format. You can display the information the way you want to see it

You create the report using graphical object called Data report designer controls. Data report designer controls. Include: a data-bound Textbox control, a function control which displays calculated figures, and Image control for inserting graphics, labels that display captions, and a line and a shape control that graphically organizes the data.

Note: Although the data report designer controls are similar to Visual Basic intrinsic controls, data repot designer controls have a limited subset of features. When the Data report designer is added to a project, the designer's controls are placed in the Visual Basic toolbox on a new tab named data report and can be used only in the Microsoft data report.

Creating repots is a main function of any good business application. A system might have useful, data, but without a coherent way to present it, the numbers are meaningless. So, Microsoft provides a Data Report designer. Which is used to generate eye catching reports. It is used in conjunction with a data source such as the data environment designer.

The Data Report generates reports using records from database. To use it.

1. Configure a data source, such as the Microsoft data environment, to access a database.
2. set the Data member property of the data report object to a data member.
3. set the data member property of the Data Report.
4. Right click the designer and click retrieve structure,
5. Add appropriate controls to the appropriate sections.
6. Set the data member and Data field properties for each control.
7. At run time, use the show method to display the data report.

Use the data report object to programmatically change the appearance and behavior of the data report by changing the layout of each section object.

The data report designer also features the ability to export reports using the export report method. This method allows you specify an export format object, from the export formats collection, to use as a template \for the report.

Part of the Data Report

The Data Reporter designer consists of the following objects:

Data Report Object - Similar to a Visual Basic form, the Data Report object has a visual designer. It is used to create the layout of a report.

Section object – The Data report designer contains sections which helps to configure the report more elegantly. The sections of the Data report designer is explained in the next section.

Data report controls – 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 a separate tab named "Data Report'

Report Designer – User the report Designer to create and modify reports. When the report designer window is active, Visual Basic display report controls toolbar.

Data report designer features – the data report designer has several features:

Drag and drop functionality for fields – When fields are dragged from the Microsoft data environment designer to the data report designer, Visual Basic automatically create a text box control on the data report and sets the Data member and Data Field Properties of the dropped field. A command object can also be dragged from the Data environment designer to the data report designer. In that case, for each of the fields contained by the command object, a text box control with be created on the date report; the Data member and Data field property for each text box will be set to the appropriate values

Toolbox controls – The data report designer features its own set of controls. When a Data report designer is added to a project, the controls are automatically crated on a new toolbox tab named Data report. Most of the controls are functionally identical to Visual Basic intrinsic controls, and include a label, shape, image, textbox, and line control. The sixth control, the function control, automatically generates one of four kinds of information: sum, average, minimum, of maximum.

Print Report – When a report is generated, it can be printed by clicking the printer icon on the toolbar.

Note: A printer must be installed on the computer to print a report

File export – in addition to creating printable reports, it can be exported to text files, by clicking the export icon and specifying text file name.

Section of the Data Report Designer

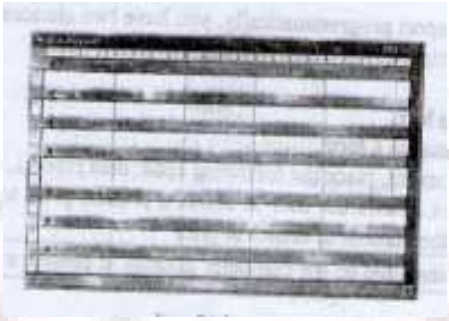
The Data report designer contains the following sections:

Report Header – Contains the text that appears at the very beginning of a report, such as the report title, author, or database name.

Page Header – Contains information that goes at the top of every page, such as the report's title.

Group Header/Footer – contains a "repeating" section of the data report. Each group header is matched with group footer. The header and footer pair are associated with a single command object in the data environment designer.

Details- Contains the innermost "repeating" part (the records) of the report. The details section is associate with the lowest level command object in a data environment hierarchy.



Page Footer – Contains the information that goes at the bottom of every page, such as the page number.

Report Footer – Contains the text that appears at the very end of the report, such as summary information, or an address of contact name.

Data report controls – When a new data report designer is added to a project, the following controls are automatically placed in the toolbox tab named data report.

- Text Box control (Rpt text Box) – Holds the data that is supplied at runtime.
- Label control (RptLabel) – used to identify fields or sections.

Figure: Data controls for Data report on the Toolbox.

- **Image control (RptImage) –** enables to place graphics on a report.

Note: This control cannot be bund to a data field.

- **Line Control (Rpt Line)-** lets to draw rules on the report to further distinguish
- **Shape Control (Rpt Shape) –** Enables to place rectangles, triangles, or circles 9and ovals) on a report.

- **Function control (Rpt Function) -** A special text box that calculates values as the report is generated.

Printing a Data Report

Printing a data report can be accomplished in one of two ways. The user can click the print button that appears on the data report in print preview mode (using the show method), or you can programmatically enable printing using the print report method. If an error occurs during printing, trap it in the error event.

Choosing to display a print dialog box

When printing a report programmatically, you have two choices: to print by displaying the print dialog box or by printing without displaying the dialog box.

To display the print dialog box.

1. Add a command button to a form.
2. In the button's click event, place the following code: data report1. print report true.

The print dialog box allows the user to select a printer, print to file, select a range of pages to print, and specify the number of copies to print.

Note: Printers must be installed on the computer in order to present a choice of printers

Printing without a Dialog Box

In some cased, you may wish to print the report without user intervention. The print report method also gives you the option of selecting a range of pages to print, either all, or a specified range.

To print without displaying the dialog box

1. Add a command button to a from
2. In the button's click event, place the following code:

   Data report1. Print report false

Or, to specify a range of pages to print, use the code below:

0 Data report1 print report false, rpt created with the data report designer.

Print Report Method

At run time, prints the data report created with the data report designer.

Syntax

Object. Print report (Show Dialog, Range, and Page From, Page To)

The print report method syntax has these parts:

Past description object required. And object expression that evaluates to an object in the applies to list. Show dialog Optional. A Boolean expression that determines if the print dialog box is shown. Range optional. Sets an integer that determines if all the pages in the report will be executed or a range of page, as shown in setting. Page from optional. An integer that sets the page from which to start printing. Page to optional. An integer that sets the page at which to stop printing.