Visual Basic .NET

<u>MINISTERIAL EXPECTATIONS</u>

1. Introduction to visual basic and .NET
2. Visual studio development environment
3. Syntax of VB.NET
    - Form and control elements
    - Control structures
    - Procedures
    - Identification and elimination of errors

Outline

- Overview of VB.Net
- Environment Setup
- Program Structure
- Basic Syntax
- Data Types
- Variables
- Constants
- Modifiers
- Statements
- Directives
- Operations
- Decision Making
- Loops
- Strings
- Date & Time
- Arrays
- Collections
- Subs
- Classes & Objects
- Exception Handling
- File Handling
- Basic Controls
- Dialog Boxes
- Advanced Forms
- Event Handling

OVERVIEW OF VISUAL BASICS

Visual Basic .NET (VB.NET) is an object-oriented computer programming language implemented on the .NET Framework. Although it is an evolution of classic Visual Basic language, it is not backwards-compatible with VB6, and any code written in the old version does not compile under VB.NET.

Like all other .NET languages, VB.NET has complete support for object-oriented concepts. Everything in VB.NET is an object, including all of the primitive types (Short, Integer, Long, String, Boolean, etc.) and user-defined types, events, and even assemblies. All objects inherits from the base class Object.

VB.NET is implemented by Microsoft's .NET framework. Therefore, it has full access to all the libraries in the .Net Framework. It's also possible to run VB.NET programs on Mono, the open-source alternative to .NET, not only under Windows, but even Linux or Mac OSX.

Setup Environment

The .Net Framework

Before getting to the tools involved in VB.NET it’s important we first of all understand what is .Net Framework. The .Net framework is a revolutionary platform that helps you to write the following types of applications:

- Windows applications
- Web applications
- Web services

The .Net framework applications are multi-platform applications. The framework has been designed in such a way that it can be used from any of the following languages: Visual Basic, C#, C++, Jscript, and COBOL, etc.

All these languages can access the framework as well as communicate with each other.

The .Net framework consists of an enormous library of codes used by the client languages like VB.Net. These languages use object-oriented methodology.

Following are some of the components of the .Net framework

- Common Language Runtime (CLR)
- The .Net Framework Class Library
- Common Language Specification
- Common Type System
- Metadata and Assemblies
- Windows Forms
- ASP.Net and ASP.Net AJAX
- ADO.Net

- Windows Workflow Foundation (WF)
- Windows Presentation Foundation
- Windows Communication Foundation (WCF)
- LINQ

Integrated Development Environment (IDE) For VB.Net

Microsoft provides the following development tools for VB.Net programming

- Visual Studio 2010 (VS)
- Visual Basic 2010 Express (VBE)
- Visual Web Developer

The last two are free. Using these tools, you can write all kinds of VB.Net programs from simple command-line applications to more complex applications. Visual Basic Express and Visual Web Developer Express edition are trimmed down versions of Visual Studio and has the same look and feel. They retain most features of Visual Studio.

Writing VB.Net Programs on Linux or Mac OS

Although the.NET Framework runs on the Windows operating system, there are some alternative versions that work on other operating systems. Mono is an open-source version of the .NET Framework which includes a Visual Basic compiler and runs on several operating systems, including various flavors of Linux and Mac OS. The most recent version is VB 2012.

The stated purpose of Mono is not only to be able to run Microsoft .NET applications cross-platform, but also to bring better development tools to Linux developers. Mono can be run on many operating systems including Android, BSD, iOS, Linux, OS X, Windows, Solaris and UNIX.

Program Structure

Before we study basic building blocks of the VB.Net programming language, let us look a bare minimum VB.Net program structure so that we can take it as a reference in upcoming sections.

VB.Net Hello World Example

A VB.Net program basically consists of the following parts:

- Namespace declaration
- A class or module
- One or more procedures
- Variables
- The Main procedure
- Statements & Expressions
- Comments

Hello World example

```
Imports System
Module Module1
  'This program will display Hello World
  Sub Main()
    Console.WriteLine("Hello World")
    Console.ReadKey()
  End Sub
End Module
```

Program Explain

- The first line of the program **Imports System** is used to include the System namespace in the program.
- The next line has a **Module** declaration, the module *Module1*. VB.Net is completely object oriented, so every program must contain a module of a class that contains the data and procedures that your program uses.
- Classes or Modules generally would contain more than one procedure. Procedures contain the executable code, or in other words, they define the behavior of the class. A procedure could be any of the following:
    - Function
    - Sub
    - Operator
    - Get
    - Set
    - AddHandler
    - RemoveHandler
    - RaiseEvent

- The next line('This program) will be ignored by the compiler and it has been put to add additional comments in the program.
- The next line defines the Main procedure, which is the entry point for all VB.Net programs. The Main procedure states what the module or class will do when executed.
- The Main procedure specifies its behavior with the statement
- **Console.WriteLine("Hello World")** *WriteLine* is a method of the *Console* class defined in the *System* namespace. This statement causes the message "Hello, World!" to be displayed on the screen.
- The last line **Console.ReadKey()** is for the VS.NET Users. This will prevent the screen from running and closing quickly when the program is launched from Visual Studio .NET.

Compile & Execute VB.Net Program

If you are using Visual Studio.Net IDE, take the following steps:

- Start Visual Studio.
- On the menu bar, choose File → New → Project.
- Choose Visual Basic from templates
- Choose Console Application.
- Specify a name and location for your project using the Browse button, and then choose the OK button.
- The new project appears in Solution Explorer.
- Write code in the Code Editor.
- Click the Run button or the F5 key to run the project. A Command Prompt window appears that contains the line Hello World.

You can compile a VB.Net program by using the command line instead of the Visual Studio IDE:

- Open a text editor and add the above mentioned code.
- Save the file as **helloworld.vb**
- Open the command prompt tool and go to the directory where you saved the file.
- Type **vbc helloworld.vb** and press enter to compile your code.
- If there are no errors in your code the command prompt will take you to the next line and would generate **helloworld.exe** executable file.
- Next, type **helloworld** to execute your program.
- You will be able to see "Hello World" printed on the screen.

Basic Syntax.

VB.Net is an object-oriented programming language. In Object-Oriented Programming methodology, a program consists of various objects that interact with each other by means of actions. The actions that an object may take are called methods. Objects of the same kind are said to have the same type or, more often, are said to be in the same class.

When we consider a VB.Net program, it can be defined as a collection of objects that communicate via invoking each other's methods. Let us now briefly look into what do class, object, methods and instance variables mean.

- **Object**: Objects have states and behaviours. Example: A dog has states colour, name, breed as well as behaviour wagging, barking, eating, etc. An object is an instance of a class.
- **Class**: A class can be defined as a template/blueprint that describes the behaviours/states that objects of its type support.

- **Methods**: A method is basically a behaviour. A class can contain many methods. It is in methods where the logics are written, data is manipulated and all the actions are executed.
- **Instance Variables**: Each object has its unique set of instance variables. An object's state is created by the values assigned to these instance variables.

A Rectangle Class in VB.Net

For example, let us consider a Rectangle object. It has attributes like length and width. Depending upon the design, it may need ways for accepting the values of these attributes, calculating area and displaying details.

Let us look at an implementation of a Rectangle class and discuss VB.Net basic syntax on the basis of our observations in it

```
Imports System
Public Class Rectangle
  Private length As Double
  Private width As Double

  'Public methods
  Public Sub AcceptDetails()
    length = 4.5
    width = 3.5
  End Sub
  Public Function GetArea() As Double
    GetArea = length * width
  End Function
  Public Sub Display()
    Console.WriteLine("Length: {0}", length)
    Console.WriteLine("Width: {0}", width)
    Console.WriteLine("Area: {0}", GetArea())
  End Sub
  Shared Sub Main()
    Dim r As New Rectangle()
    r.Acceptdetails()
    r.Display()
    Console.ReadLine()
```

```
   End Sub
End Class
```

Output

```
Length: 4.5
Width: 3.5
Area: 15.75
```

In previous examples, we created a Visual Basic module that held the code. Sub Main indicates the entry point of VB.Net program. Here, we are using Class that contains both code and data. You use classes to create objects. For example, in the code, r is a Rectangle object.

An object is an instance of a class

Dim r As New Rectangle()

A class may have members that can be accessible from outside class, if so specified. Data members are called fields and procedure members are called methods.

Shared methods or **static** methods can be invoked without creating an object of the class. Instance methods are invoked through an object of the class.

```
Shared Sub Main()
   Dim r As New Rectangle()
   r.Acceptdetails()
   r.Display()
   Console.ReadLine()
End Sub
```

Identifiers

An identifier is a name used to identify a class, variable, function, or any other user-defined item. The basic rules for naming classes in VB.Net are as follows:

- A name must begin with a letter that could be followed by a sequence of letters, digits (0 - 9) or underscore. The first character in an identifier cannot be a digit.
- It must not contain any embedded space or symbol like ? - +! @ # % ^ & * ( ) [ ] { } . ; : " ' / and \. However, an underscore (_) can be used.
- It should not be a reserved keyword.

VB.Net Keywords

The following table lists the VB.Net reserved keywords −

AddHandler	AddressOf	Alias	And	AndAlso	As	Boolean
ByRef	Byte	ByVal	Call	Case	Catch	CBool
CByte	CChar	CDate	CDec	CDbl	Char	CInt
Class	CLng	CObj	Const	Continue	CSByte	CShort

CSng	CStr	CType	CUInt	CULng	CUShort	Date
Decimal	Declare	Default	Delegate	Dim	DirectCast	Do
Double	Each	Else	ElseIf	End	End If	Enum
Erase	Error	Event	Exit	False	Finally	For
Friend	Function	Get	GetType	GetXML Namespace	Global	GoTo
Handles	If	Implements	Imports	In	Inherits	Integer
Interface	Is	IsNot	Let	Lib	Like	Long
Loop	Me	Mod	Module	MustInherit	MustOverride	MyBase
MyClass	Namespace	Narrowing	New	Next	Not	Nothing
Not Inheritable	Not Overridable	Object	Of	On	Operator	Option
Optional	Or	OrElse	Overloads	Overridable	Overrides	ParamArray
Partial	Private	Property	Protected	Public	RaiseEvent	ReadOnly
ReDim	REM	Remove Handler	Resume	Return	SByte	Select
Set	Shadows	Shared	Short	Single	Static	Step
Stop	String	Structure	Sub	SyncLock	Then	Throw
To	True	Try	TryCast	TypeOf	UInteger	While
Widening	With	WithEvents	WriteOnly	Xor		

Data Types

Data types refer to an extensive system used for declaring variables or functions of different types. The type of a variable determines how much space it occupies in storage and how the bit pattern stored is interpreted.

Data Types Available in VB.Net

VB.Net provides a wide range of data types. The following table shows all the data types available