BCA Semester-5
US05CBCA01-Visual Programming through VB.NET

UNIT-1 Introduction to .NET Framework and VB.NET

Microsoft .NET

Microsoft .NET (pronounced "dot net") is a software component that runs on the Windows operating system. .NET provides tools and libraries that enable developers to create Windows software much faster and easier. .NET benefits end-users by providing applications of higher capability, quality and security. The .NET Framework must be installed on a user's PC to run .NET applications.

.NET technology provides the ability to quickly build, deploy, manage, and use connected, security-enhanced solutions with Web services

VB.NET

- Visual Basic .NET (VB.NET), is an object-oriented computer programming language that can be viewed as an evolution of the classic Visual Basic (VB), which is implemented on the .NET Framework
- It is the next generation of the visual Basic language.
- It supports OOP concepts such as abstraction, inheritance, polymorphism, and aggregation.

The .NET Framework Architecture

A programming infrastructure created by Microsoft for building, deploying, and running applications and services that use .NET technologies, such as desktop applications and Web services

- It is a platform for application developers.
- It is tiered, modular, and hierarchal.
- It is a service or platform for building, deploying and running applications.
- It consists of 2 main parts: Common language runtime and class libraries.

The common language runtime is the bottom tier, the least abstracted. The .NET Framework is partitioned into modules, each with its own distinct responsibility. The architectural layout of the .NET Framework is illustrated in following figure:

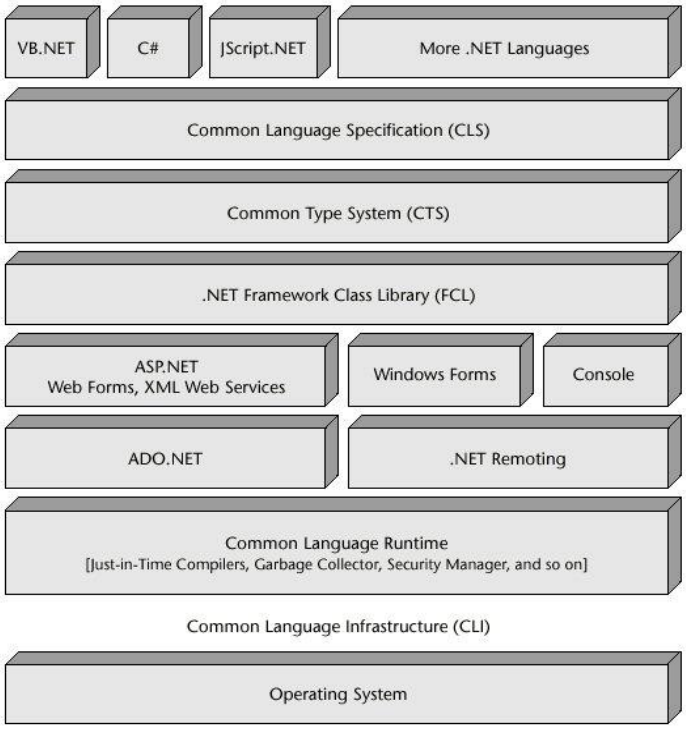


Figure 1 An overview of the .NET architecture.

Here we examine the following key components of the .NET Framework:

CLR (Common Language Runtime)

The .NET Framework provides a runtime environment called the Common Language Runtime or CLR (similar to the Java Virtual Machine or JVM in Java), which handles the execution of code and provides useful services for the implementation of the program.

The CLR is the execution engine for .NET applications and serves as the interface between .NET applications and the operating system. The CLR provides many services such as:

- Loads and executes code
- Converts intermediate language to native machine code
- Manages memory and objects
- Enforces code and access security
- Handles exceptions
- Interfaces between managed code, COM objects, and DLLs
- Provides type-checking
- Provides code meta data (Reflection)

- Provides profiling, debugging, etc.
- Separates processes and memory

FCL (Framework Class Library)

It is also known as a base class library. The FCL is a collection of over 7000 reusable classes , interfaces, and value types that enable .NET applications to

- read and write files,
- access databases,
- process XML,
- display a graphical user interface,
- draw graphics,
- use Web services, etc.

The .Net Framework class library (FCL) organized in a hierarchical tree structure and it is divided into Namespaces. Namespaces is a logical grouping of types for the purpose of identification. Framework class library (FCL) provides the consistent base types that are used across all .NET enabled languages. The Classes are accessed by namespaces, which reside within Assemblies.

CTS (Common Type System)

- The CLS is a common platform that integrates code and components from multiple .NET programming languages.
- CTS allow programs written in different programming languages to easily share information.
- CLS forms a subset of CTS. This implies that all the rules that apply to CTS also apply to CLS also.
- It defines rules that a programming language must follow to ensure that objects written in different programming languages can interact with each other.
- CTS provide cross language integration.
- The common type system supports two general categories of **types**:

1. Value Type
2. Reference Type

Value Type: Stores directly data on stack. In built data type . For ex. Dim a as integer.

Reference Type: Store a reference to the value's memory address, and are allocated on the heap. For ex: dim obj as new oledbconnection.

The Common Language Runtime (CLR) can load and execute the source code written in any .Net language, only if the type is described in the Common Type System (CTS)

Common Language Specification(CLS)

- CLS includes basic language features needed by almost all applications.
- CLS id a subset of the Common Type System.
- It serves as a guide for library writes and compiler writers.
- The CLS is also important to application developers who are writing code that will be used by other developers.

Microsoft Intermediate Language

- When you compile your Visual Basic .NET source code, it is changed to an intermediate language (IL) that the CLR and all other .NET development environments understand.
- All .NET languages compile code to this IL, which is known as Microsoft Intermediate Language, MSIL, or IL.
- MSIL is a common language in the sense that the same programming tasks written with different .NET languages produce the same IL code.

- At the IL level, all .NET code is the same regardless of whether it came from C++ or Visual Basic.

- When a compiler produces Microsoft Intermediate Language (MSIL), it also produces Metadata.
- The Microsoft Intermediate Language (MSIL) and Metadata are contained in a portable executable (PE) file .
- Microsoft Intermediate Language (MSIL) includes instructions for loading, storing, initializing, and calling methods on objects, as well as instructions for arithmetic and logical operations, control flow, direct memory access, exception handling, and other operations. Figure 2 shows what happens to your code from its inception in Visual Studio to execution.

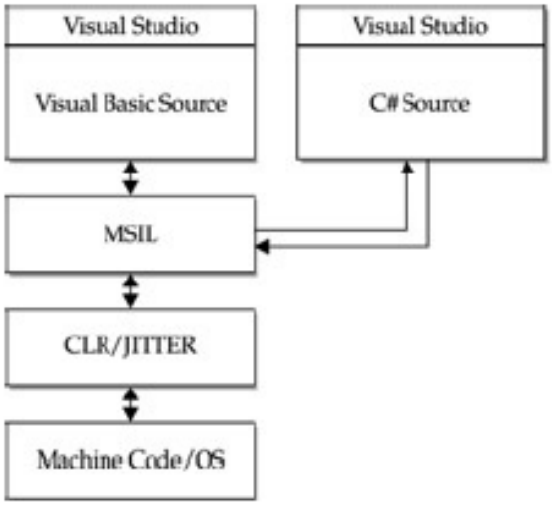


Figure 2: Following the IL

JIT (Just-in-Time)

- The Common Language Runtime (CLR) provides various Just In Time compilers (JIT) and each works on a different architecture depending on Operating System. That is why the same Microsoft Intermediate Language (MSIL) can be executed on different Operating Systems without rewrite the source code.
- Just In Time (JIT) compilation preserves memory and save time during application initialization.
- Just In Time (JIT) compilation is used to run at high speed, after an initial phase of slow interpretation.
- Just In Time Compiler (JIT) code generally offers far better performance than interpreters.

What is a .Net Assembly? (For the purpose of reference)

- Whatever .NET language you create applications with, compilers generate an *assembly*, which is a file containing .NET executable code and is composed essentially by two kinds of elements: MSIL code and metadata.
- The .NET assembly is the standard for components developed with the Microsoft.NET. Dot NET assemblies may or may not be executable, i.e., they might exist as the executable (.exe) file or dynamic link library (DLL) file.
- All the .NET assemblies contain the definition of types, versioning information for the type, meta-data, and manifest. The designers of .NET have worked a lot on the component (assembly) resolution.

The structure of an assembly
Assemblies contain code that is executed by the Common Language Runtime.

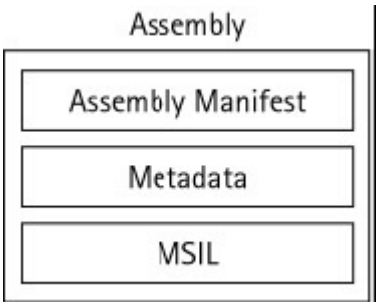


Figure: A diagram of assembly

Assemblies are made up of the following parts:

- The assembly manifest
- Type metadata
- Microsoft Intermediate Language (MSIL) code

The assembly manifest is where the details of the assembly are stored. The assembly is stored within the DLL or EXE itself. Assemblies can either be single or multifile assemblies and, therefore, assembly manifests can either be stored in the assembly or as a separate file. The assembly manifest also stores the version number of the assembly to ensure that the application always uses the correct version.
The metadata contains information on the types that are exposed by the assembly such as security permission information, class and interface information, and other assembly information.

Contents of an Assembly
• Assembly Manifest
• Assembly Name
• Version Information
• Types
• Locale
• Cryptographic Hash
• Security Permissions

An assembly does the following functions:
• It contains the code that the runtime executes.
• It forms a security boundary. An assembly is the unit at which permissions are requested and granted.
• It forms a type boundary. Every type's identity includes the name of the assembly at which it resides.

• It forms a reference scope boundary. The assembly's manifest contains assembly metadata that is used for resolving types and satisfying resource requests. It specifies the types and resources that are exposed outside the assembly.
• It forms a version boundary. The assembly is the smallest version able unit in the common language runtime; all types and resources in the same assembly are versioned as a unit.
• It forms a deployment unit. When an application starts, only the assemblies the application initially calls must be present. Other assemblies, such as localization resources or assemblies containing utility classes, can be retrieved on demand. This allows applications to be kept simple and thin when first downloaded.
• It is a unit where side-by-side execution is supported
Assembly Manifest
Every assembly, whether static or dynamic, contains a collection of data that describes how the elements in the assembly relate to each other. The assembly manifest contains this assembly metadata. An assembly manifest contains the following details:
• Identity. An assembly's identity consists of three parts: a name, a version number, and an optional culture.
• File list. A manifest includes a list of all files that make up the assembly.
• Referenced assemblies. Dependencies between assemblies are stored in the calling assembly's manifest. The dependency information includes a version number, which is used at run time to ensure that the correct version of the dependency is loaded.
• Exported types and resources. The visibility options available to types and resources include "visible only within my assembly" and "visible to callers outside my assembly."
• Permission requests. The permission requests for an assembly are grouped into three sets:
1) those required for the assembly to run,

2) those that are desired but the assembly will still have some functionality even if they aren't granted, and 3) those that the author never wants the assembly to be granted.

There are two kinds of assemblies in .NET;

- private
- shared

Private assemblies Is the assembly which is used by application only, normally it resides in your application folder directory.

Shared assemblies - It resides in GAC, so that anyone can use this assembly. Public assemblies are always share the common functionalities with other applications.

An assembly can be a single file or it may consist of the multiple files. In case of multi-file, there is one master module containing the manifest while other assemblies

exist as non-manifest modules. A module in .NET is a sub part of a multi-file .NET assembly. Assembly is one of the most interesting and extremely useful areas of .NET architecture along with reflections and attributes, but unfortunately very few people take interest in learning such theoretical looking topics.

Visual Studio .NET IDE

Visual Studio .NET IDE (Integrated Development Environment) is the Development Environment for all .NET based applications which comes with rich features. VS .NET IDE provides many options and is packed with many features that simplify application development by handling the complexities. Visual Studio .NET IDE is an enhancement to all previous IDE's by Microsoft.

Important Features

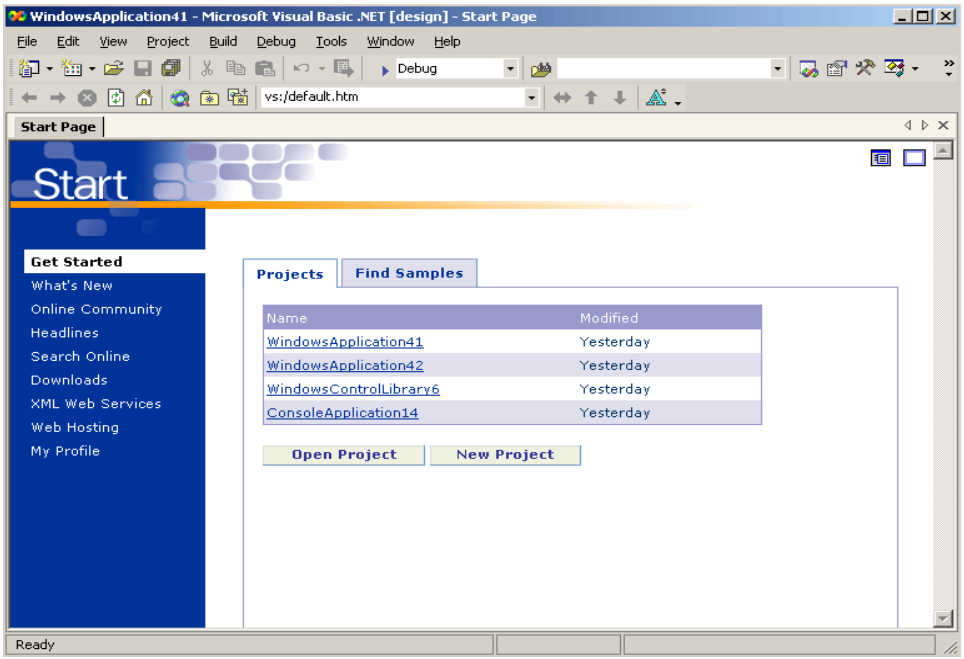
One IDE for all .NET Projects: Visual Studio .NET IDE provides a single environment for developing all types of .NET applications.

Option to choose from Multiple Programming Languages: You can choose the programming language of your choice to develop applications based on your expertise in that language.

IDE is Customizable: You can customize the IDE based on your preferences.

Built-in Browser: The IDE comes with a built-in browser that helps you browse the Internet without launching another application.

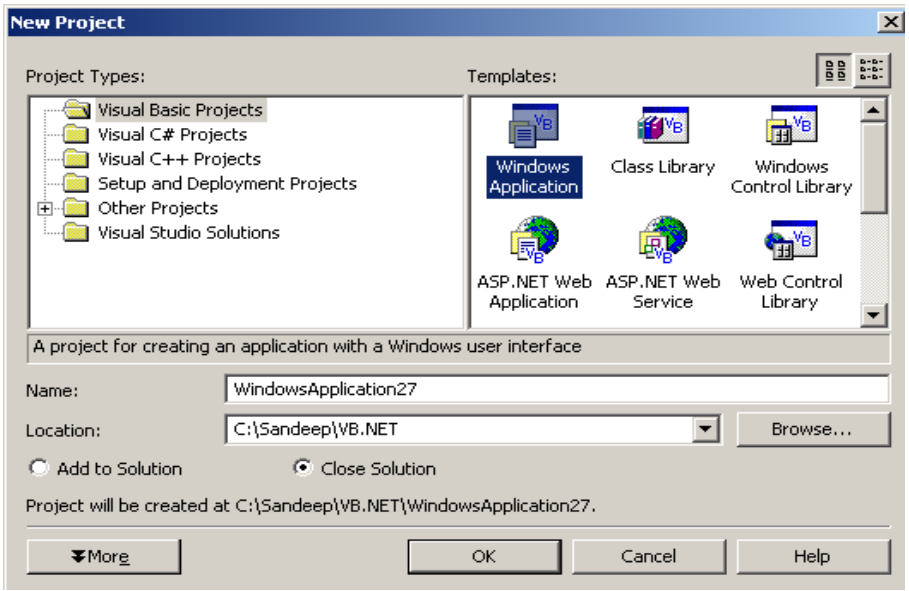
When we open VS .NET from Start->Programs->Microsoft Visual Studio .NET->Microsoft Visual Studio .NET the window that is displayed first is the Start Page which is shown below. The start Page allows us to select from the most recent projects (last four projects) with which we worked or it can be customized based on your preferences.



New Project Dialogue Box

The New Project dialogue box like the one in the image below is used to create a new project specifying it's type allowing us to name the project and also specify it's location on the disk where it is saved.

The default location on the hard disk where all the projects are saved is C:\DocumentsandSettings\Administrator\MyDocuments\VisualStudioProjects.



Project Types

Following are different templates under Project Types and their use.

Windows Application: This template allows to create standard windows based applications.

Class Library: Class libraries are those that provide functionality similar to Active X and DLL by creating classes that access other applications.

Windows Control Library: This allows to create our own windows controls. Also called as User Controls, where you group some controls, add it to the toolbox and make it available to other projects.

ASP .NET Web Application: This allows to create web-based applications using IIS. We can create web pages, rich web applications and web services.

ASP .NET Web Service: Allows to create XML Web Services.

Web Control Library: Allows to create User-defined controls for the Web. Similar to user defined windows controls but these are used for Web.

Console Application: A new kind of application in Visual Studio .NET. They are command line based applications.

Windows Service: These run continuously regardless of the user interaction. They are designed for special purpose and once written, will keep running and come to an end only when the system is shut down.

Other: This template is to develop other kinds of applications like enterprise applications, database applications etc.

The Integrated Development Environment (IDE) shown in the image below is what we actually work with. This IDE is shared by all programming languages in Visual Studio.

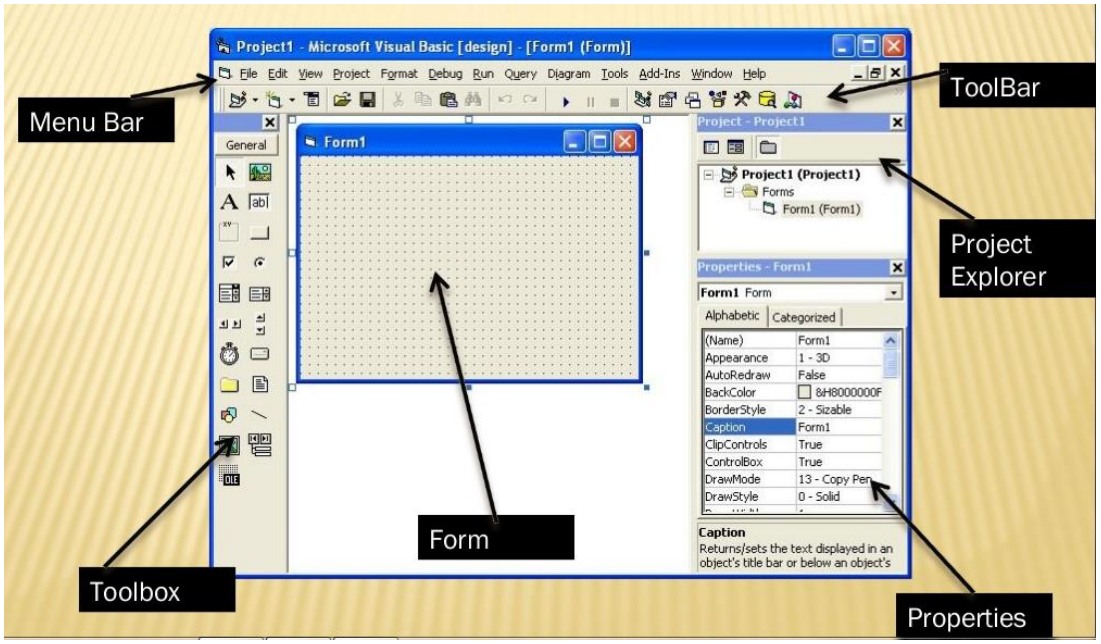


Figure : Visual Studio IDE

Visual Studio .NET's menu is dynamic, meaning that items will be added or removed depending on what you are trying to do. The menu bar consist of the following options:

File: With this menu you can create a new project, open existing one, save the current project ,exit form the vb.net etc

Edit: The Edit menu provides access to the items you would expect: Undo, Redo, Cut, Copy, Paste, and Delete.

View: The View menu provides quick access to the windows that make up the IDE, such as the Solution Explorer, Properties window, Output window, Toolbox, etc.

Project: The Project menu allows you to add various extra files to your application.

Build: The Build menu becomes important when you have completed your application and want to be able to run it without the use of the Visual Basic .NET environment.

Debug: The Debug menu allows you to start and stop running your application within the Visual Basic .NET IDE. It also gives you access to the Visual Studio .NET **debugger**.

Data: The Data menu helps you use information that comes from a database. It only appears when you are working with the visual part of your application ,not when you are writing code.

Format: The Format menu also only appears when you are working with the visual part of your application. Items on the Format menu allow you to manipulate how the windows you create will appear to the users of your application.

Tools : The Tools menu has commands to configure the Visual Studio .NET IDE, as well as links to other external tools that may have been installed.

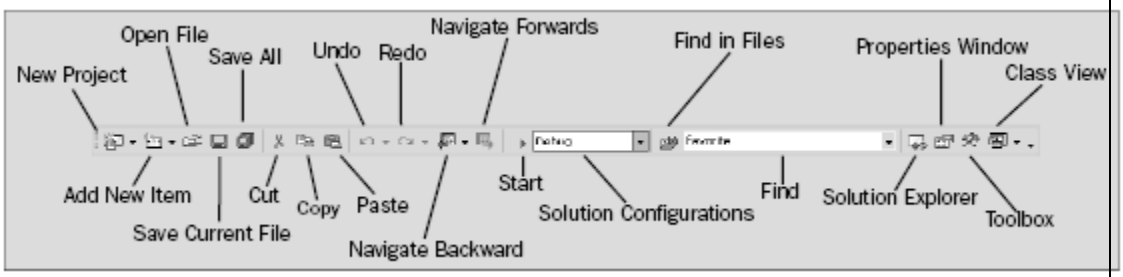
Window: The commands on this menu allow you to change the physical layout of the windows in the IDE.

Help: The Help menu provides access to the Visual Studio .NET documentation.

The Toolbars

There are many toolbars available within the IDE, including Formatting, Image Editor, and Text Editor, which you can add to and remove from the IDE via the View | Toolbars menu option. Each one provides quick access to often-used commands, preventing you from having to navigate through a series of menu options. For example, the leftmost icon on the toolbar shown below (New Project) is available from the menu by navigating to File | New | Project.

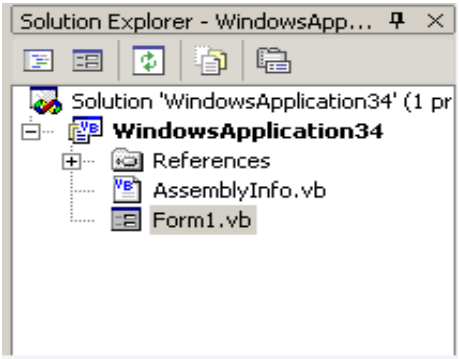
The default toolbar (called Standard) appears at the top of the IDE as:



The toolbar is segmented into groups of related options, which are separated by a vertical bar.

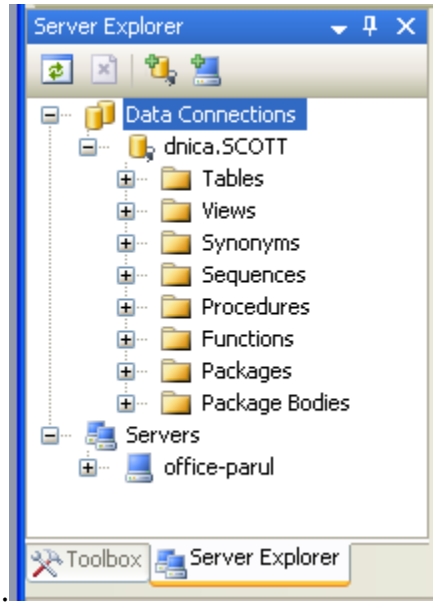
Solution Explorer Window

The Solution Explorer window gives an overview of the solution we are working with and **lists all the files in the project.** An image of the Solution Explorer window is shown below.



Server Explorer Window

The Server Explorer window is a great tool that provides "drag and drop" feature and helps us work with databases in an easy graphical environment. For example, if we drag and drop a database table onto a form, VB .NET automatically creates connection and command objects that are needed to access that table. The image below displays Server Explorer window



Code Designer Window

Code Designers like the image below allows us to edit and write code. This is the window that opens when we double-click on a form or any control. This is the place where we write all the code for the application.

The two drop-down list boxes at the top of the code window in the image below.

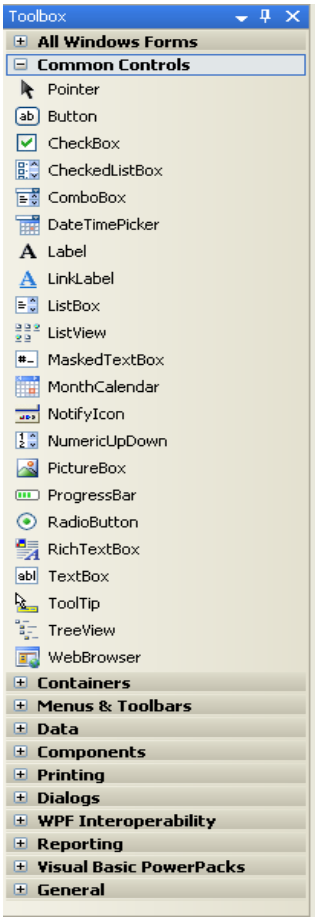
- The left box allows us to select the object's code we are working with and
- The right box allows us to select the part of code that we want to work.

Also notice the "+" and "-" boxes in the code designer. You can use those boxes to display code Visual Basic .NET already created, like, Windows Forms Designer generated code, etc.

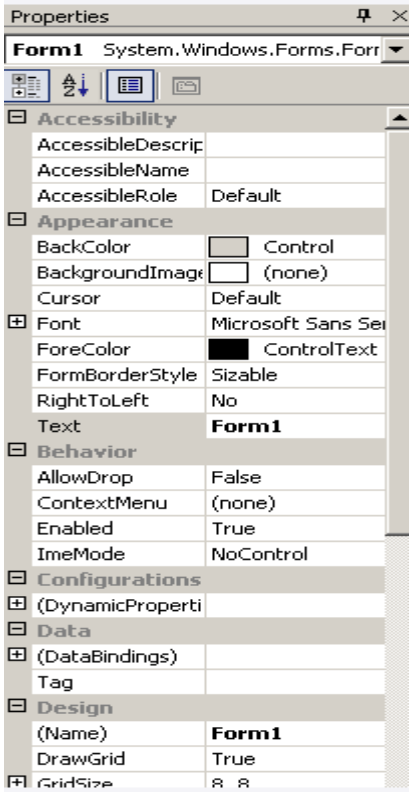
```
Form1                                   DataGrid1_Navigate
Public Class Form1
    Inherits System.Windows.Forms.Form

    Windows Form Designer generated code

    Private Sub CheckBox1_CheckedChanged(ByVal sender As System.Obje

    End Sub

    Private Sub DataGrid1_Navigate(ByVal sender As System.Object, By

    End Sub
End Class
```

Toolbox Window

The toolbox window is the window that gives us access to all controls, components, etc. As you can see from the image below, the toolbox uses tabs to divide it's contents into categories (Data, Components, Windows Forms ,Container, General etc.). The Data tab displays tools for creating datasets and making data connections, the Windows Forms tab displays tools for adding controls to forms, the General tab is left empty by default.



Properties Window



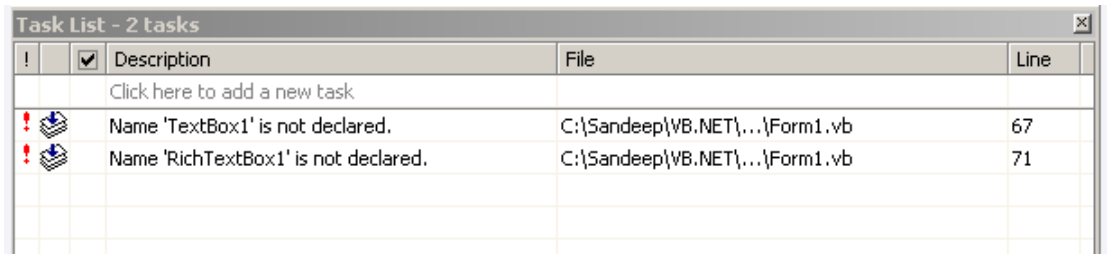
The properties window allows us to set properties for various objects at design time.

For example, if you want to change the font, font size, back color, name, text that appears on a button, textbox etc, you can do that in this window.

You can view the properties window by selecting View->Properties Window from the main menu or by pressing F4 on the keyboard. You can display properties of the selected object in sorting manner or in categoried manner.

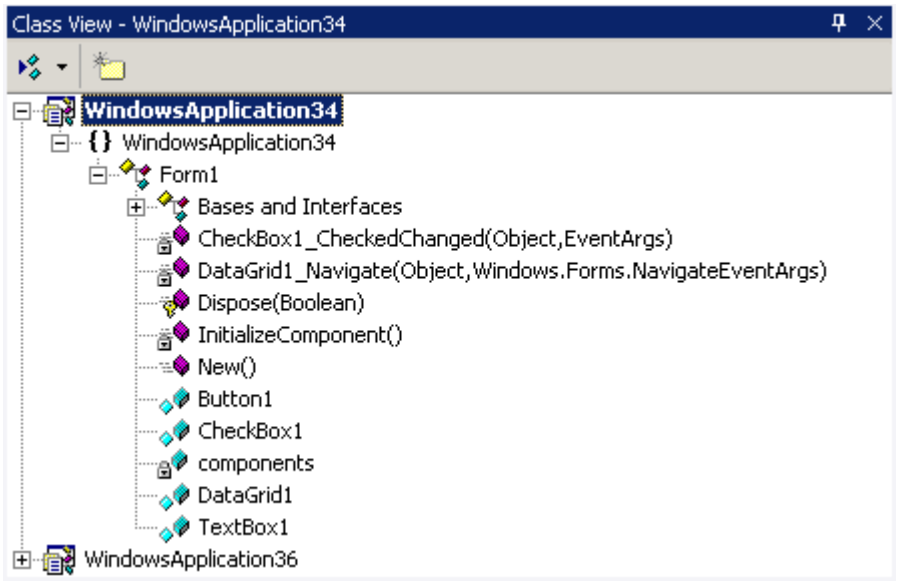
Task List Window

The task list window displays all the tasks that VB .NET assumes we still have to finish. You can view the task list window by selecting View->Show tasks->All or View->Other Windows->Task List from the main menu. As you can see from the image, the task list displayed "TextBox1 not declared", "RichTextBox1 not declared". The reason for that message is, there were no controls on the form and attempts where made to write code for a textbox and a richtextbox. Task list also displays syntax errors and other errors you normally encounter during coding.



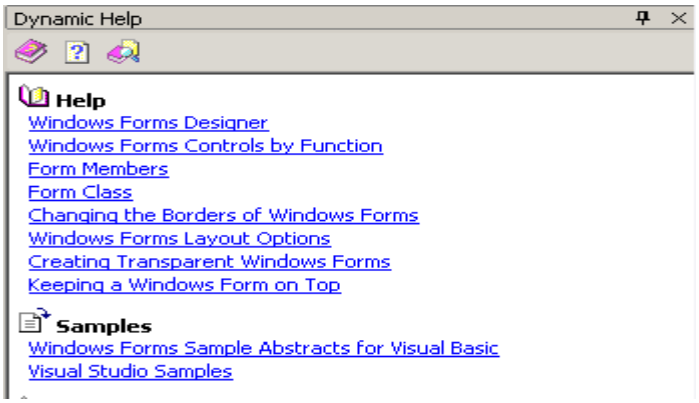
Class View Window

The class view window like the image below is the window that presents solutions and projects in terms of the classes they contain and the members of these classes. Using the class view window also helps us to find a member of a class that we want to work with.



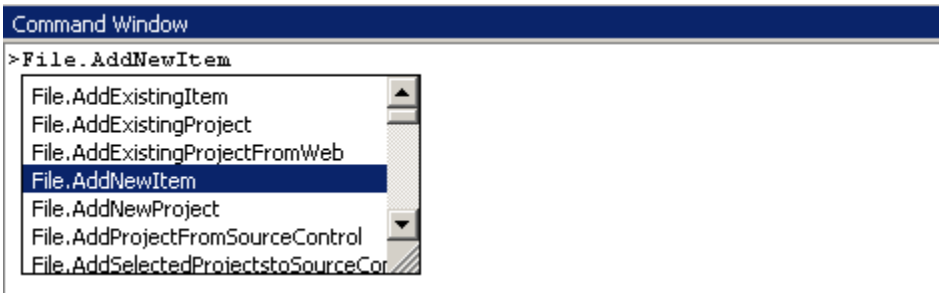
Dynamic Help Window

The dynamic help window displays help which looks up for things automatically. For example, if you want to get help with a form, select the form and select Help->Dynamic Help from the main menu. Doing that displays all the information relating to forms. You can get help relating to anything with this feature.



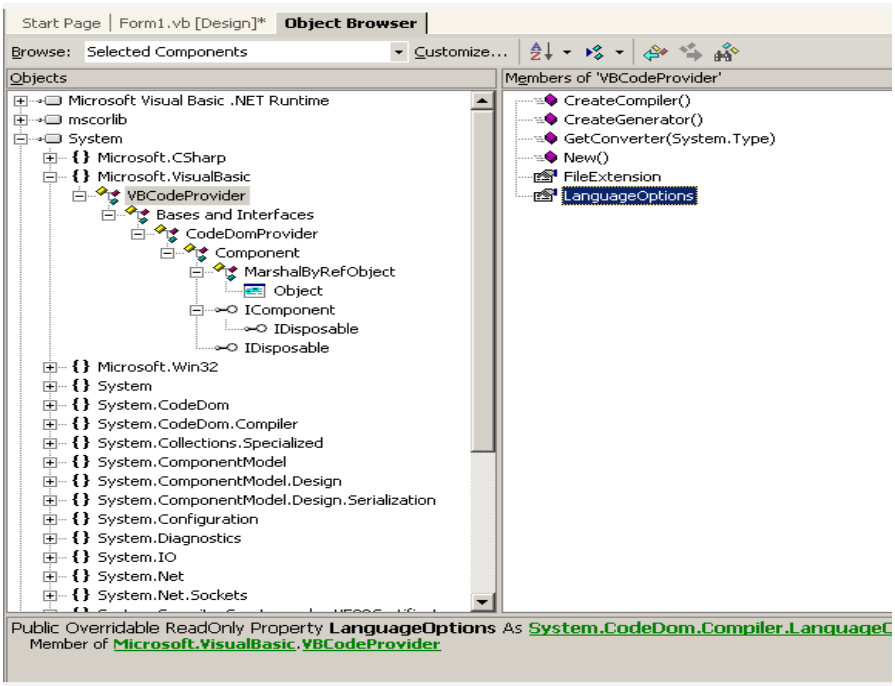
Command Window

The command window in the image below is a useful window. Using this window we can add new item to the project, add new project and so on. You can view the command window by selecting View->Other Windows->Command Window from the main menu.



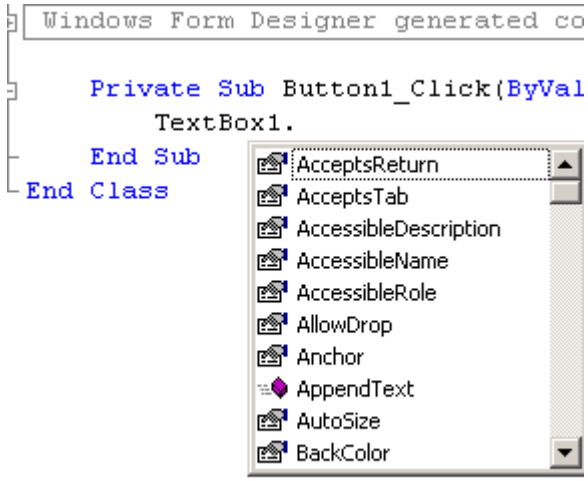
Object Explorer Window

The object explorer window allows us to view all the members of an object at once. It lists all the objects in our code and gives us access to them. The image below displays an object explorer window. You can view the object explorer window by selecting View->Other Windows-> Object Browser from the main menu.



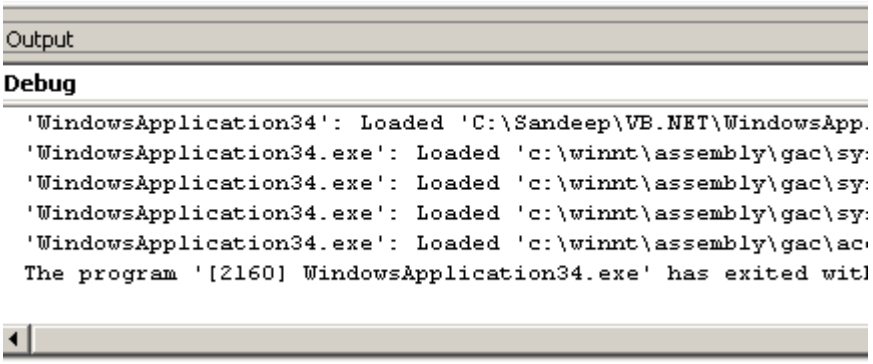
Intellisense

Intellisense is what that is responsible for the boxes that open as we type the code. IntelliSense provides a list of options that make language references easily accessible and helps us to find the information we need. They also complete the typing for us. The image below displays that.



Output Window

The output window as you can see in the image below displays the results of building and running applications.



Variables:

A variable is something that is used in a program to store data in memory

A variable has a name and a data type which determine the kind of data the variable can store.

Variable declaration:

Dim statement is used to declare a variable.

Variable name should follow the following rules:

- Begin with a letter or _.
- Must contain atleast one numeric digit or alphabetic character.
- Maximun 1023 characters are allowed.
- It must be unique in its scope.

Syntax

For example:

```
Dim Rollno As Integer

Dim Rollno As Integer = 10
```

The following are the data type supported by VB.Net .

Numeric Data Type(Short, Integer, Long, Single, Double, Decimal)

Character Data Type (Char, String)

Miscellaneous Data Type(Boolean , Byte, Date, Object)

Data Type	Sample value	Range of values
String	"hello there"	Alphanumeric characters, digits and other characters
Char	'a'C	Single characters the C stands for character
Integer	-5	Larger numbers ranging from -2147483648 to 2147483647
Long	18459038	Very Large whole numbers
Boolean	True	**True** or **False**
Byte	10	small numbers range 0 to 255
Short	1234	medium numbers ranging from -32,768 to 32,767
Single	5.678	Single-precision floating point numbers with six digits of accuracy.
Double	-2.4585E30	Double-precision floating point numbers with 14 digits of accuracy
Decimal	10.50	Decimal fractions, such as dollars and cents.
Date	#5/23/2002#	month, day, year, various date formats available
Object	frm as Form	Objects represent a group of many values

Constants and Operators

CONSTANTS

The variables are called **constants those values can not change during execution** and are also known as **"read only".** This means you can read the value again but you can never write a new value to it.

You use the keyword **const** to specify that a variable will be a constant and initialize it with a value.

Const Max = 10

Once you initialize the constant variable with a value it can never change.

Operators

Visual Basic comes with many built-in operators that allow us to manipulate data. An operator performs a function on one or more operands. For example, we add two variables with the "+" addition operator and store the result in a third variable with the "=" assignment operator like this: int x + int y = int z. The two variables (x ,y) are called operands. There are different types of operators in Visual Basic and they are described below in the order of their precedence.

Operators may me Unary or Binary

Unary used with a single operand for example Ans= -10

Binary used with two operands for example Ans= 10 / 5

Operators also categorized in following categories:

Arithmetic Operators

Arithmetic operators are used to perform arithmetic operations that involve calculation of numeric values. The table below summarizes them:

Operator	Use
^	Exponentiation
-	Negation (used to reverse the sign of the given value, exp -intValue)
*	Multiplication
/	Division for ex a=11/5 then answer is 5.5
\	Integer Division for ex a=11\5 then answer is 5
Mod	Modulus Arithmetic
+	Addition
-	Subtraction

Concatenation Operators

Concatenation operators join multiple strings into a single string. There are two concatenation operators, + and & as summarized below:

Operator	Use
+	**String Concatenation**
&	**String Concatenation**

Comparison Operators

A comparison operator compares operands and returns a logical value based on whether the comparison is true or not. The table below summarizes them:

Operator	Use
=	Equality
<>	Inequality
<	Less than
>	Greater than
>=	Greater than or equal to
<=	Less than or equal to

Logical / Bitwise Operators

The logical operators compare Boolean expressions and return a Boolean result. In short, logical operators are expressions which return a true or false result over a conditional expression. The table below summarizes them:

Operator	Use
Not	Negation
And	Conjunction
AndAlso	Conjunction
Or	Disjunction
OrElse	Disjunction
Xor	Disjunction

Types of Variable:

Value type (A variable of value type stores its associated data directly within itself.)

Reference type (A variable of reference type always contains a reference to a value of that type or a null reference.)

For eg:

Dim pi As Double = 3.14159 (Value type)

Dim obj1 As Object (Reference type)
obj1 = New myobj()

BOXING:

Converting Value Types to Reference Types known as **boxing.**
For eg:

```
'Implicit Conversion
Dim x As Int32 = 10
Dim o As Object = x

'Explicit Conversion
Dim x As Int32 = 10
Dim o As Object = CObj(x)
```

UNBOXING:

Converting reference type back to the value type is known as **Unboxing**.
For eg:

```
Dim x As Int32 = 5
```

```
Dim o As Object = x                ' Implicit Boxing
x = o                              ' Implicit UnBoxing

Dim x As Int32 = 5
Dim o As Object = x                ' Implicit Boxing
x = CInt(o)                        ‘ Explicit UnBoxing
```

Type Casting:

A type cast is an explicit conversion of an expression to a given type. Explicit boxing and explicit unboxing are examples of TypeCasting.