DOT NET TECHNOLOGIES

UNIT -1

1.WHAT IS .NET?

- NET is a software framework which is designed and developed by Microsoft.
- The first version of the .Net framework was 1.0 which came in the year 2002.
- In easy words, it is a virtual machine for compiling and executing programs written in different languages like C#, VB.Net etc.
- It is used to develop Form-based applications, Web-based applications, and Web services.
- There is a variety of programming languages available on the .Net platform, VB.Net and C# being the most common ones.
- It is used to build applications for Windows, phone, web, etc.

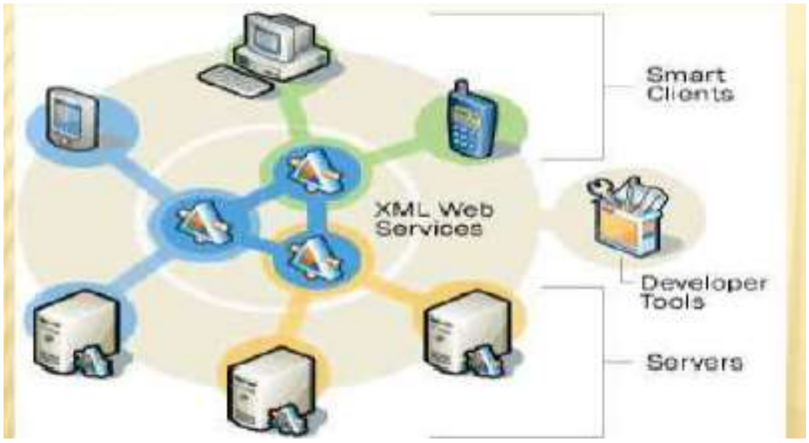
What is a Web Service?

Client to Client - Clients can use XML Web Services to communicate data

Client to Server - Clients can send data to and receive data from servers.

Server to Server - Servers can share data with each other.

Service to Service - web services can work together.



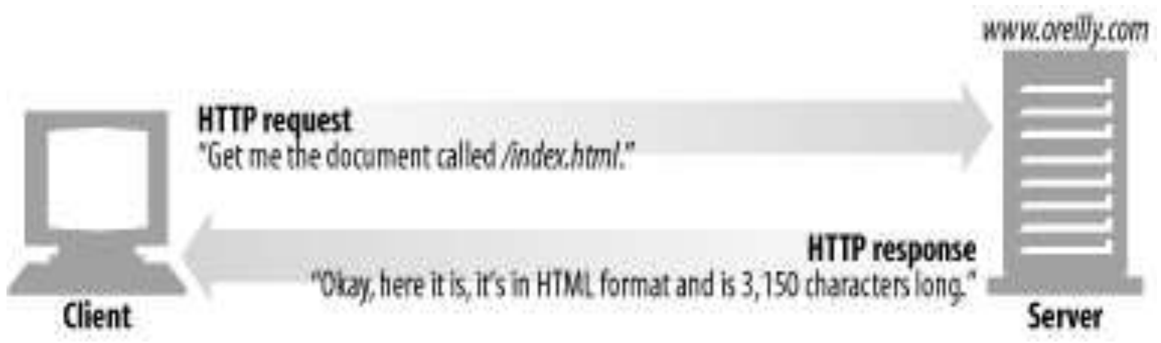
1.1 THE WEB CLIENT/ SERVER MODEL

<u>The Web Clients and Web Servers:</u>

- Web Based environment takes place between Two entities:

  1.A Web Client , which is the software that requests files, data and services

  2 A Web Server,which fulfills the client requests for contents


www.oreilly.com
HTTP request
"Get me the document called /index.html."
HTTP response
"Okay, here it is, it's in HTML format and is 3,150 characters long."
Client
Server

<u>Protocols for Web Client/Server Communication:</u>

The Web Client and Server send information using TCP/IP

- **TCP -** Transmission Control Protocol

  TCP is responsible for breaking data down into small packets before they can be sent over a network, and for assembling the packets again when they arrive.

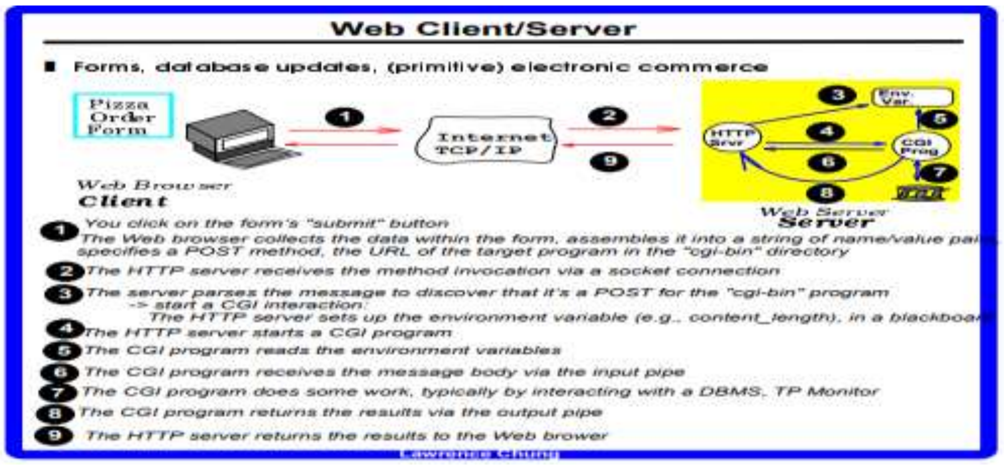
- **IP -** Internet Protocol

  IP takes care of the communication between computers. It is responsible for addressing, sending and receiving the data packets over the Internet.

- **FTP -** File Transfer Protocol

  FTP takes care of transmission of files between computers.

- A socket is line of plumbing established either client or server software that is used to move packets between client and server.
- A Packets is a fragments of data includes information about its origination and destination.
- Web Application communicates over Sockets using higher level protocol the HTTP (HyperText Transfer Protocol).
- HTTP is a protocol which allows the fetching of resources, such as HTML documents.
- It is the foundation of any data exchange on the Web and it is a client-server protocol, which means requests are initiated by the recipient, usually the Web browser.



Web Client/Server

- Forms, database updates, (primitive) electronic commerce

1. You click on the form's "submit" button
   The Web browser collects the data within the form, assembles it into a string of name/value pai
   specifies a POST method, the URL of the target program in the "cgi-bin" directory
2. The HTTP server receives the method invocation via a socket connection
3. The server parses the message to discover that it's a POST for the "cgi-bin" program
   -> start a CGI interaction:
   The HTTP server sets up the environment variable (e.g., content_length), in a blackboa
4. The HTTP server starts a CGI program
5. The CGI program reads the environment variables
6. The CGI program receives the message body via the input pipe
7. The CGI program does some work, typically by interacting with a DBMS, TP Monitor
8. The CGI program returns the results via the output pipe
9. The HTTP server returns the results to the Web brower

Lawrence Chung

<u>SERVER SIDE PROCESSING WITH CGI</u>

- ➢ CGI works by processing the data sent by the web browser to the web server.
- ➢ The data contained in the form fields are encoded by the browser using X-URL-encoding system before transmission.
- ➢ The data is sent by one of 2 methods:
  - Encoding the data an HTTP POST (Post request)
  - Passing the data to script file in URL(Get request).
- ➢ The Web Server receives this data and passes it to CGI application of processing.

1.2 COMPONENTS OF ASP.NET AND THE .NET FRAMEWORK

- ➢ The following are the components of asp.net and the .net framework:

- **Internet Information Server →** server software that deliver web content to the user browser.

- **Asp.net page →** HTML page that host static content and execute code. It act as the glue between the presentation layer and application services.
- **Common Language Runtime(CLR) and the ,Net framework Library →** CLR provide runtime environment for manage code and library provide components for developing applications.
- **Managed Components →** allow sharing of modules among many different applications.
- **Web Services →** provide components based functionality that access over network by any machine using variety of protocols.

1.3 Overview of Internet Information Server(IIS)

- IIS is Microsoft software for serving content on web.
  - IIS use **HTTP →** deliver web content
  - **FTP and SMTP →** send e-mail
- The functionality of IIS is extendable through the ISAPI(Internet Server Application Programmers Interface).
- ISAPI program come in two flavors:
  1. ISAPI Extension
  2. ISPAI Filters

1. ISAPI Extension

- Resemblance to traditional CGI program.

- They receive encoded data from HTTP post request as well as from Get request.
- It exits as window **Dynamic Link Libraries** (DLL).
- DLL are compiled modules whose functions are called by another program during runtime.
- ISAPI extension are also **multithreaded**, which enables them to run many times concurrently.

2. ISAPI Filters

- jobs is to **intercept requests** from a web browser and send back some kind of alternative response.
- It **modify's the default behavior** of web server and it return data□hence name is filter.

When web server has no ISAPI Filters the following sequence occur:

- User requests a **URL-**□browser send HTTP command to the web server.
- Web server **receives the command** and requests the file in URL.
- The **Server sends the content** of the file to the **client** and the connection is closed.

When web server has an ISAPI Filters the following sequence occur:

- User requests a **URL --** >browser send HTTP command to the web server.
- Web server **receives the request** and passes that request data as parameter to the **ISAPI filter code**.

- The ISAPI filter performs some **server side processing** and then send **customized data** through **IIS server** to the client and the connection is closed.

ISAPI Filters have many applications, include the following

- Customized logging of user activity on the server.
- Advance security system that examine the origin of each HTTP request
- On fly decryption of files on the web server based on password.
- Modification of output stream of the web server.

1.4 OVERVIEW OF ASP.NET

- ASP.NET is unique among web application development platforms in 2 main ways,

1.ASP.NET Application use programming language such as VB.NET,C#,JScript.NET, which increase flexibility during development different language can be used based on programming skills and experience.

2. Many reusable controls that make common programming task and basic user interface controls like buttons, text fields and drop down lists included plus advanced controls like calender controls and editable data grid are used in declarative manner such by click and drag the control in your applications.

1. WEB FORM :

- Web forms allow programmers to build form based web applications designed around event driven application model.

- ASP.Net web forms are very powerful for a number of reasons:
    - Web forms can run on any browser. Web forms are controlled server side and rendered using HTML code.
    - ASP.NET determines what content to serve back to user based on automatic browser detection.
    - .NET compatible language can be used to code
    - Web forms execute inside the CLR which brings all advantages

1.5 THE .NET COMMON LANGUAGE RUNTIME AND CLASS LIBRARY

- The .Net CLR controls execution of programs within this environment.
- Code thar runs in CLR known as Managed code
- The CLR enables language code to interact with code written in other language at object level.
- The .Net framework class library is an collection of data types and objects to give programmers start in developing their applications.
- The data type and objects are designed and categorized to make accessing system functionality such as input/output, database access, other services easy.
- The . Net Framework class library also supports objects like basic data types as well as abstract data types.

1.6 MANAGED COMPONENTS IN .NET

- A components is an independent unit of code that encapsulates a task or critical business rules and logic.
- The basic programmatic steps that order entry system performs to fulfill a customer order:
    1. A clerk fills out shipping information, billing information and order details in computerized form then clerk sent details to the server.
    2. The server receives the data. The system compares the order details with the list of current product inventory.
        a) If all item requested are in stock,The system logs the order to a database.
        b) If any item requested are out of stock,The system informs the customer that the order can't fully filled at that time.
    3. The system then calculates total cost and shipping charges and sends the order to the company warehouse via E-Mail.
    4. The order e-mail is received at the warehouse and the product needed are picked then product is shipped to the customer.
- It is a good example for order processing system can be wrapped into an order entry components.
- The .Net Framework also make deployment of the object easy with the use of embedded metadata that describe the details of the component to clients.

1.7 WEB SERVICES

- A Web service, is a component that resides on a Web server and provides information and services to other network applications using standard Web protocols such as HTTP and Simple Object Access Protocol (SOAP).
- .NET Web services provide communications for XML applications that operate over a .NET communications framework.
- Users on the Internet can use applications that are not dependent on their local operating system or hardware and are generally browser-based.
- The main advantage of a Web service is that its consumers can use the service without knowing about the details of its implementation, such as the hardware platform, programming language, object model, etc.
- Web services are designed to provide the messaging infrastructure necessary for communication across platforms using industry standards.
- A Web Service is a software program that uses XML to exchange information with other software via common internet protocols.
- In a simple sense, Web Services are a way of interacting with objects over the Internet.

1.8 LANGUAGE INDEPENDENCE IN THE .NET FRAMEWORK

- One great advantage of .Net Framework platform is the ability to use language to write application.
- The Components Object Model(COM) allows for COM Components in any language such as VB and C++.
- .NET framework language interoperability features shine.
- Every components has to follow rules of communication established for sending information to complied components.
- In team environment,developers can be free to use whatever language they choose to develop objects.

1.9 COM+ COMPONENTS SERVICES AND .NET

- COM+ Components Services provide transaction processing,object pooling,queued components and other features.
- COM+ is an integral part of windows and distributed computing.
- COM+ provides a runtime environment for components,which include COM+ components and .NET assemblies.
- Within this runtime environment, you can do several things:
    1. **Manage automatic transaction:** *you can commit or rollback changes made to data in an automatic fashion when certain event happens in program code.*

2. **Use declarative role-based security*:*** *you can assign execute permissions to certain components based on logical groups to which active user has been assigned.*
3. **Make Component use Object pooling:** *precreate object and keep them in awaiting use by client program.*
4. **Create queued Components:** *you can immediately execute components over a network connection.*

1.10 DIRECTION AND PLANS FOR .NET

- Web Services open standard for communication and encoding of data exchanges, this open door for any computer system to interface with any device the user has.
- Exciting for web services,developed by Microsoft is called .NET My Services.
- Is often refer as “Digital Safe Deposit Box” for the user.
- Inside this personal information space the user can store many things.

.NET MY SERVICES THE FOLLOWING FEATURES WILL BE INCLUDED:

- .NET Profile – name, nickname, special dates, picture and address.
- .NET Contacts – electronic relationships/address book.
- .NET Locations – electronic and geographical location.
- .NET Inbox – Inbox item like email and voice mail, including existing mail system.
- .NET Calendar – time and task management.

1.11 WHAT IS VB.NET?

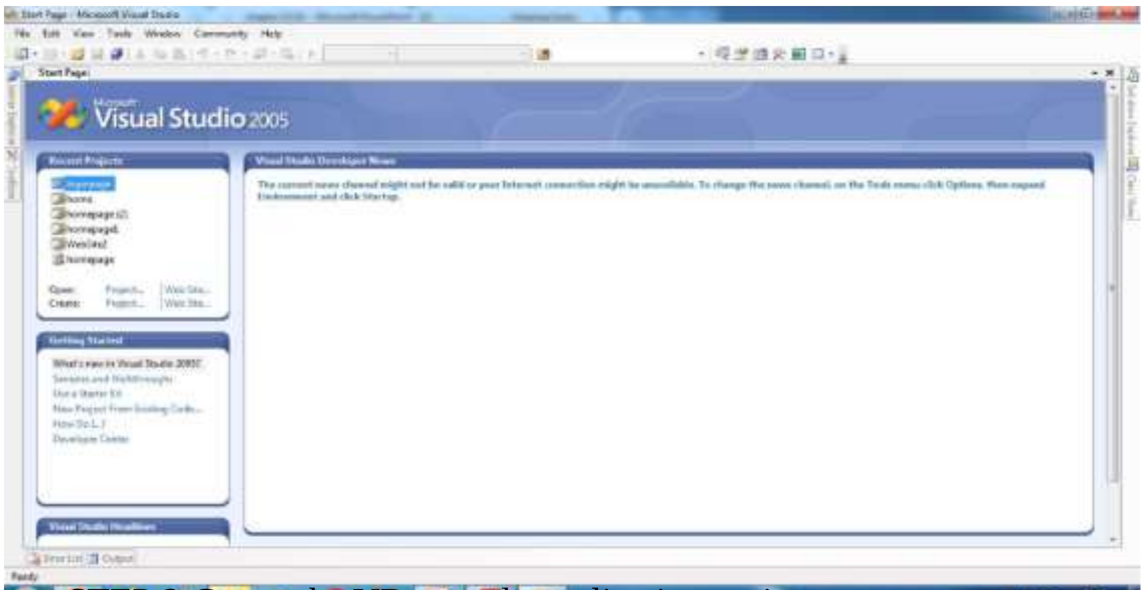
- Visual Basic .NET is an **Object-Oriented Programming** (OOP)language designed by Microsoft.
- It is a simple, modern, object-oriented computer programming language to combine the power of .NET Framework and the common language runtime .
- VB.Net was designed to take advantage of the .NET framework-based classes and run-time environment.
- VB.Net programming is very much based on **BASIC** and Visual Basic programming languages, so if you have basic understanding on these programming languages, then it will be a fun for you to learn VB.Net **programming language** .
- The **syntax** is easy and you will not find yourself writing hundreds of lines of code as there are many shortcuts that make coding so much easier in this language.
- VB.NET is the first fully object-oriented programming (OOP) version of Visual Basic, and as such, supports OOP concepts such as abstraction, inheritance, **polymorphism** , and aggregation.

1.12 YOUR FIRST VB APPLICATION

Step 1: Launch the VS.NET IDE.

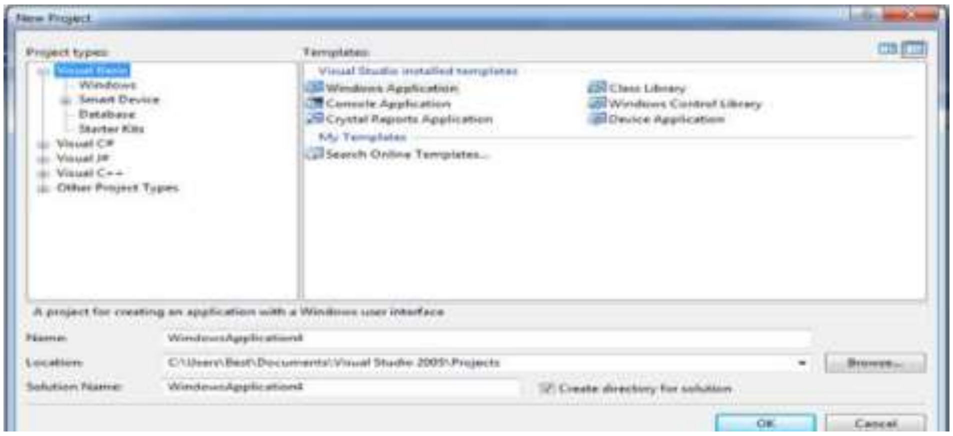
a. From the windows start button, select PROGRAM->MICROSOFT VISUAL STUDIO.NET

7.0 -> MICROSOFT VISUAL STUDIO.NET 7.0.The VS.NET home page appears as


Start Page - Microsoft Visual Studio
File Edit View Tools Window Community Help
Start Page
Microsoft Visual Studio 2005
Recent Projects
Visual Studio Developer News
Getting Started
Visual Studio Headlines
Open: Project... Web Site...
Create: Project... Web Site...
Ready

STEP 2:Create the VB console application project.

a. To begin new project, select File→New → Project from the menu bar. The New Project dialog appears with options for new Project.



A project for creating an application with a Windows user interface

Name: WindowsApplication4

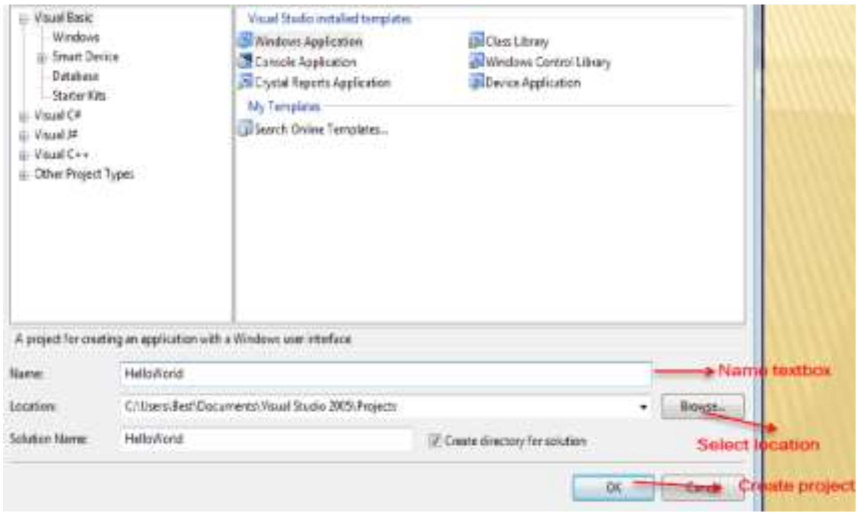
Location: C:\Users\Best\Documents\Visual Studio 2005\Projects

Solution Name: WindowsApplication4

[✓] Create directory for solution

b. Once you locate and highlight the option, then you need to give project name by entering the name in Name textbox as “HelloWorld”. Then you can select the location for project files by clicking the Browse Button or you can use default location.

c. Now click OK Button to create new project.



A project for creating an application with a Windows user interface

Name: HelloWorld

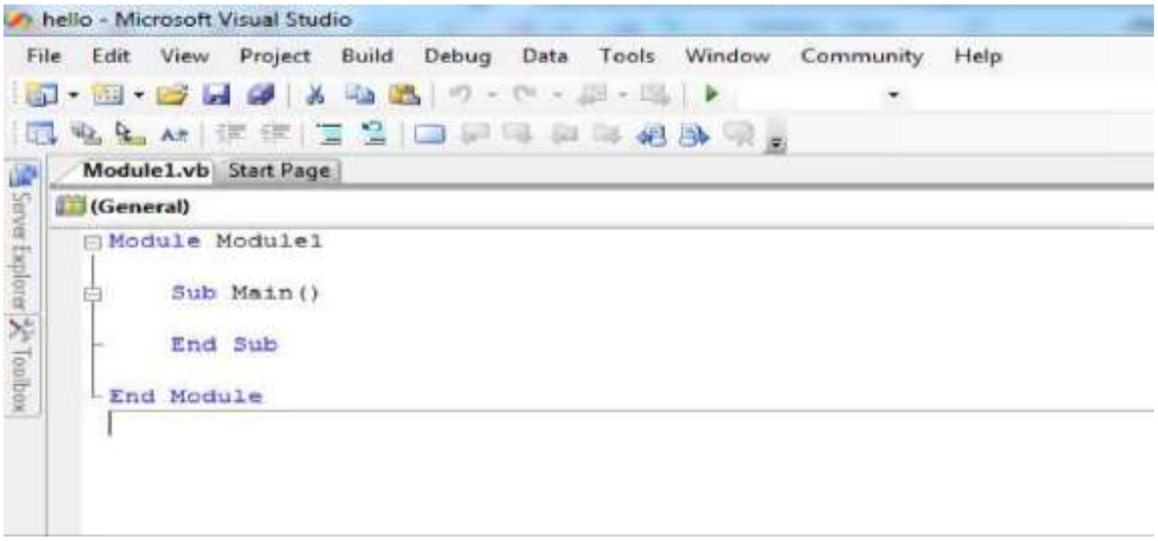
Location: C:\Users\Best\Documents\Visual Studio 2005\Projects

Solution Name: HelloWorld

☑ Create directory for solution

STEP 3: Enter the Program Code.

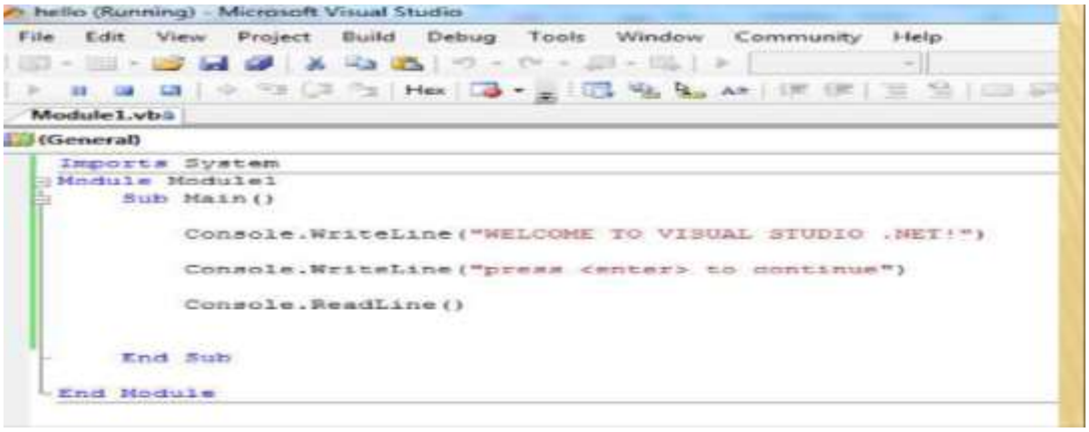
a. you should now see the code window f or Module 1, a default code Module 1 create by VS.NET. You can modify the default code in this window to create your first application.



```
Module Module1

    Sub Main()

    End Sub

End Module
```

b. Now you are ready to type in some code to make “Helloworld “ project say hello to its user.

Example Coding:


hello (Running) - Microsoft Visual Studio
File Edit View Project Build Debug Tools Window Community Help
Module1.vb
(General)

```
Imports System
Module Module1
    Sub Main()

        Console.WriteLine("WELCOME TO VISUAL STUDIO .NET!")

        Console.WriteLine("press <enter> to continue")

        Console.ReadLine()

    End Sub
End Module
```

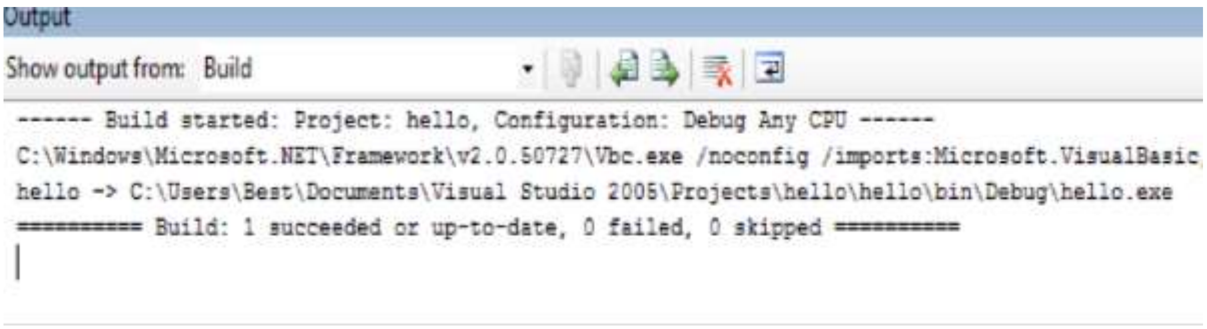
c. Click Save All button on the tool bar. This save your code module and all other modified files.

STEP 4: Build and Run Project.

a. To build the project, Select **Build → Build Solution** from menu. If the build process is successful, then in status bar appear as “Solution update succeeded”.

b. To run the Program,Click Start button in the toolbar or you can press F5 key.Then console window will apprear with the running program

c. When program terminated (after you press <enter> to continue),the system returns to VS.NET.



```
------ Build started: Project: hello, Configuration: Debug Any CPU ------
C:\Windows\Microsoft.NET\Framework\v2.0.50727\Vbc.exe /noconfig /imports:Microsoft.VisualBasic,
hello -> C:\Users\Best\Documents\Visual Studio 2005\Projects\hello\hello\bin\Debug\hello.exe
========== Build: 1 succeeded or up-to-date, 0 failed, 0 skipped ==========
```

1.13 VARIABLES

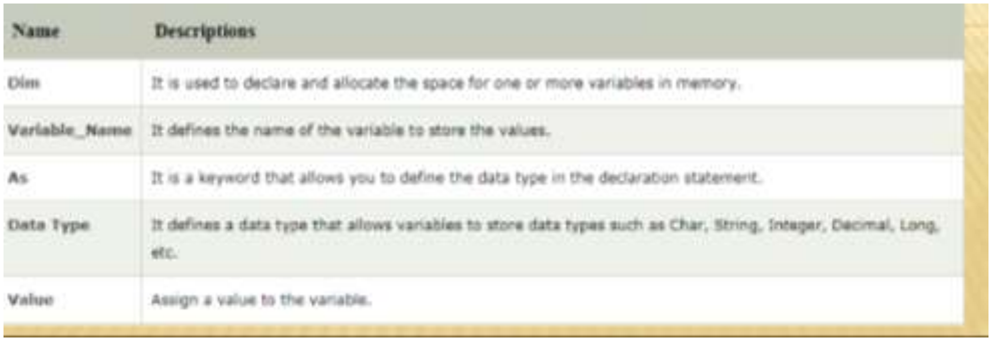
- A variable is a simple name used to store the value of a specific data type in computer memory.
- In VB.NET, each variable has a particular data type that determines the size, range, and fixed space in computer memory.
- With the help of variable, we can perform several operations and manipulate data values in any programming language.

1. VARIABLES DECLARATION

- The declaration of a variable is simple that requires a variable name and data type followed by a Dim.
- A Dim is used in Class, Module, structure, Sub, procedure.

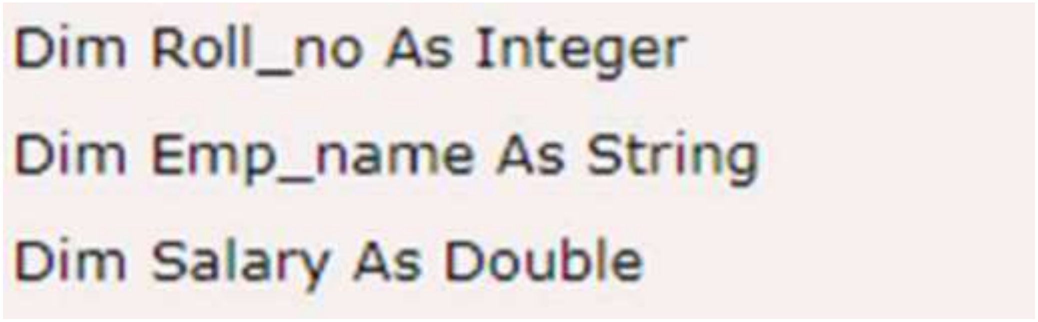
Syntax:

```
Dim [Variable_Name] As [Defined Data Type]
```



Name	Descriptions
Dim	It is used to declare and allocate the space for one or more variables in memory.
Variable_Name	It defines the name of the variable to store the values.
As	It is a keyword that allows you to define the data type in the declaration statement.
Data Type	It defines a data type that allows variables to store data types such as Char, String, Integer, Decimal, Long, etc.
Value	Assign a value to the variable.

- There are some valid declarations of variables along with their data type definition, as shown below:

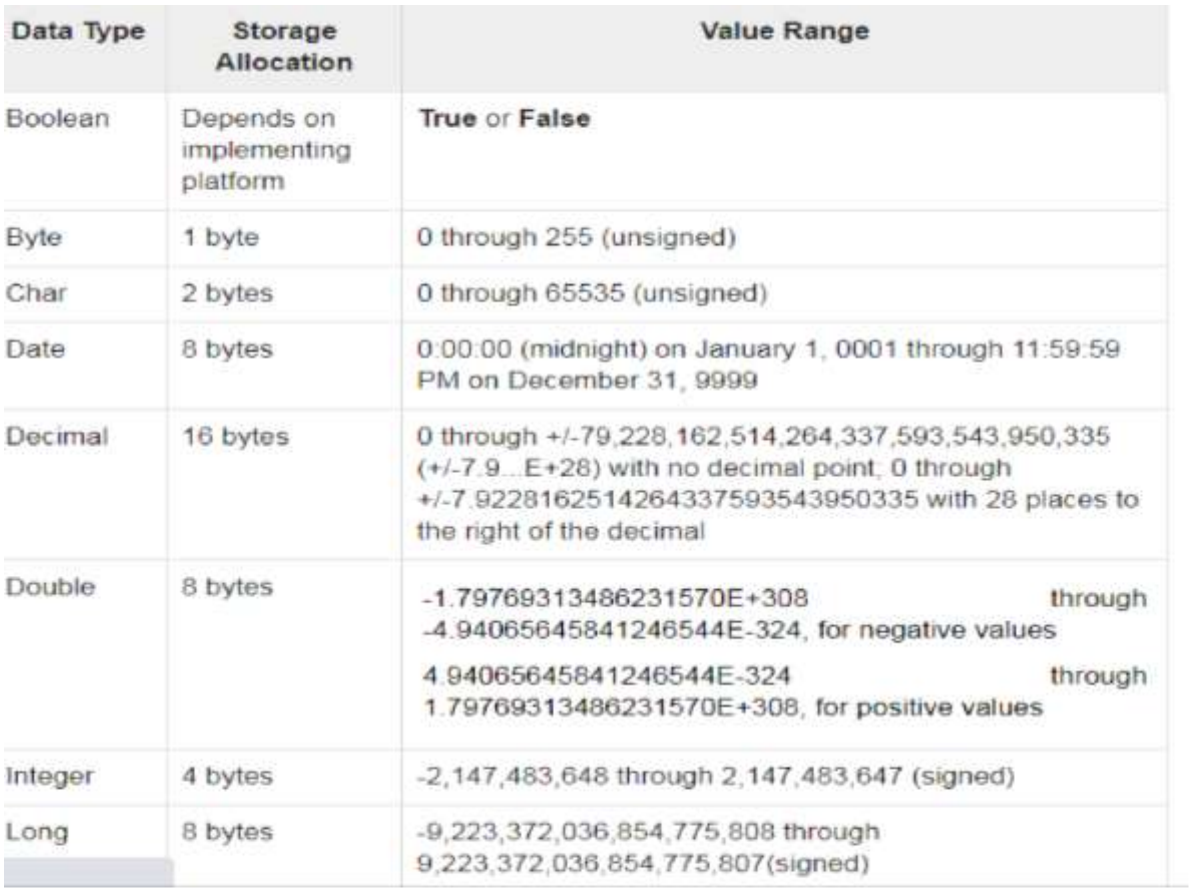


```
Dim Roll_no As Integer

Dim Emp_name As String

Dim Salary As Double
```

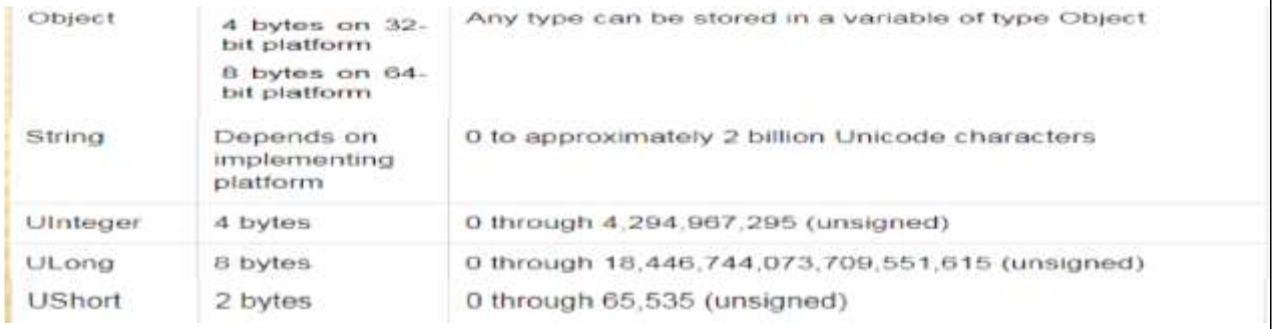
- Further, if we want to declare more than one variable in the same line, we must separate each variable with a comma.

```
Dim
Variable_name1 As DataType1,
variable_name2 As DataType2,
Variable_name3 As DataType3
 Dim Empid,Sudid as Integer
```

2. Data Types Available in VB.Net



Data Type	Storage Allocation	Value Range
Boolean	Depends on implementing platform	**True** or **False**
Byte	1 byte	0 through 255 (unsigned)
Char	2 bytes	0 through 65535 (unsigned)
Date	8 bytes	0:00:00 (midnight) on January 1, 0001 through 11:59:59 PM on December 31, 9999
Decimal	16 bytes	0 through +/-79,228,162,514,264,337,593,543,950,335 (+/-7.9...E+28) with no decimal point; 0 through +/-7.9228162514264337593543950335 with 28 places to the right of the decimal
Double	8 bytes	-1.79769313486231570E+308 through -4.94065645841246544E-324, for negative values 4.94065645841246544E-324 through 1.79769313486231570E+308, for positive values
Integer	4 bytes	-2,147,483,648 through 2,147,483,647 (signed)
Long	8 bytes	-9,223,372,036,854,775,808 through 9,223,372,036,854,775,807(signed)



Object	4 bytes on 32-bit platform 8 bytes on 64-bit platform	Any type can be stored in a variable of type Object
String	Depends on implementing platform	0 to approximately 2 billion Unicode characters
UInteger	4 bytes	0 through 4,294,967,295 (unsigned)
ULong	8 bytes	0 through 18,446,744,073,709,551,615 (unsigned)
UShort	2 bytes	0 through 65,535 (unsigned)

3. VARIABLE INITIALIZATION

- After the declaration of a variable, we must assign a value to the variable. The following syntax describes the initialization of a variable:
- **Syntax:**

  Variable_name = value

  - **Example:**

    Dim Roll_no As Integer 'declaration of Roll_no

    Roll_no = 101 'initialization of Roll_no

  - We can also initialize a variable at the time of declaration as:

    Dim Roll_no As Integer = 101

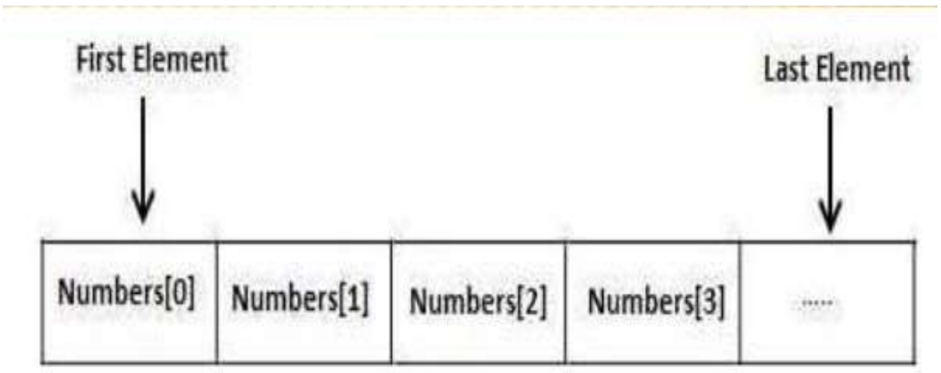
1.14 SCOPE AND LIFETIME OF VARIABLES

- When declare a variables ,the value that it holds has limitations on where and when it can accessed is the scope and lifetime variables.
- variables declared in the declarations section of a module using the Private keyword can be accessed by all the procedures within the module.

- Variables declared in the declaration section of a code module using the Public keyword can be accessed by the whole project.
- And variables declared using the Dim statement within a subroutine or function can only be accessed in the procedure in which they've been declared.

ARRAYS

- Arrays are a collection of data elements that all have the same data type.
- Arrays are useful for processing large set of data without having to declare a separate variable for each data element.
- Arrays have dimensionality, which refer to the number of subscripts used to access each element in the array.
- All arrays have atleast one demension and can support up to sixty dimensions.
- All arrays consist of contiguous memory locations.The lowest address corresponds to the first element and the highest address to the last element.
- The below figure shows a representation of a one dimensional arrays. It consists of five elements, each element a string.



Creating Arrays in VB.Net

- To declare an array in VB.Net, you use the Dim statement.
- For example,

```
Dim Data(30) ' an array of 31 elements
Dim strData(20) As String ' an array of 21 strings
Dim twoDarray(10, 20) As Integer 'a two dimensional array of integers arranged in 10 rows and 20 columns
```

DYNAMIC ARRAYS

- Dynamic arrays are arrays that can be dimensioned and re-dimensioned as par the need of the program. You can declare a dynamic array using the **ReDim** statement.
- Syntax for ReDim statement −

```
ReDim [Preserve] arrayname(subscripts)
```

Where,

- The **Preserve** keyword helps to preserve the data in an existing array, when you resize it.
- **arrayname** is the name of the array to re-dimension.
- **subscripts** specifies the new dimension.

EXAMPLE :

```
Dim marks() As Integer
ReDim Preserve marks(2)
marks(0) = 85
marks(1) = 75
 marks(2) = 90
```

CONVERTING DATA TYPES

- Variables can be converted from one data type to another.
- There are two types on conversion are

1**. Implicit conversion:**

- An implict conversion process makes assumptions on behalf of the programmer about

how the data should be treated during the conversion.

Ex: Dim intA as
interger
= 3
Dim intB as
Long
intB = intA

2**. Explicit Conversion:**

- An Explicit Conversion may also used to convert data types.
- It involves the use of special vb.net functions to convert one data type to another datatype.

Ex:

```
Dim intA as interger = 3
 Dim intB as string
  intB = intA
```

1.14 CONSTANTS

- The **Constants** refer to fixed values that the program may not alter during its execution.

- Constants can be of any of the basic data types like an integer constant, a floating constant, a character constant, or a string.
- In VB.Net, constants are declared using the **Const** statement.

Syntax:

```
const variablename as datatype = value
```

Ex:

```
const strname as string = “ram”
```

1.12 ARITHMETIC AND COMPARISON OPERATORS

What is an operator?

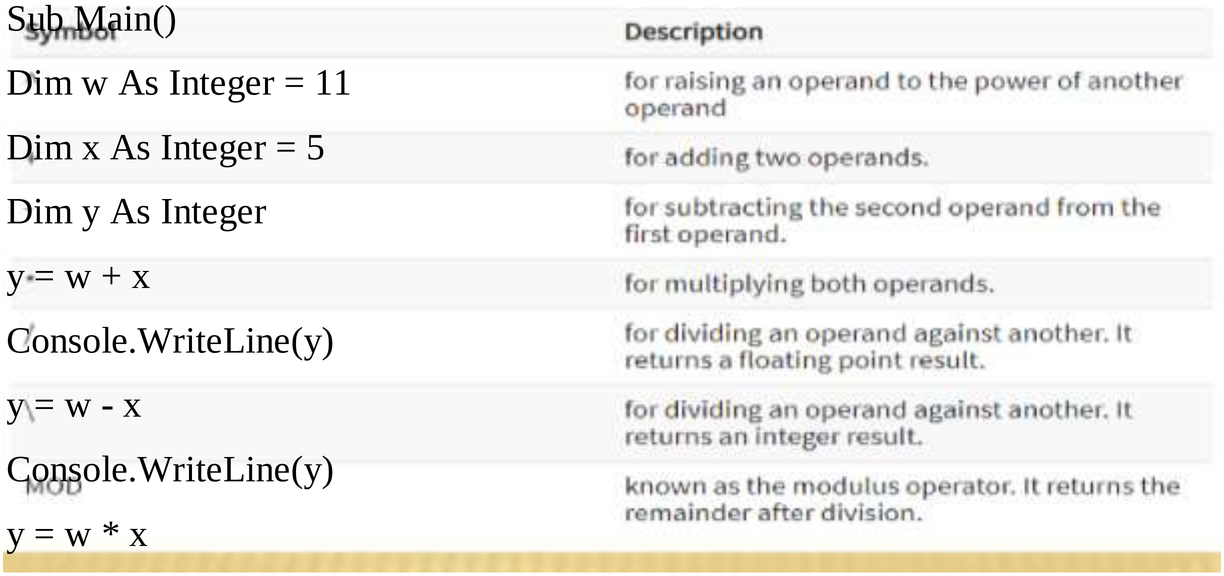
- An operator refers to a symbol that instructs the compiler to perform a specific logical or mathematical manipulation.
- The operator performs the operation on the provided operands.
- Microsoft VB.Net comes with various types of operators.

LIST OF OPERATORS

1. Arithmetic Operators
2. Comparison Operators
3. Logical/Bitwise Operators
4. Bit Shift Operators
5. Assignment Operators
6. Miscellaneous Operators

ARITHMETIC OPERATORS

- You can use arithmetic operators to perform various mathematical operations in VB.NET. They include:



Symbol	Description
^	for raising an operand to the power of another operand
+	for adding two operands.
-	for subtracting the second operand from the first operand.
*	for multiplying both operands.
/	for dividing an operand against another. It returns a floating point result.
\	for dividing an operand against another. It returns an integer result.
MOD	known as the modulus operator. It returns the remainder after division.

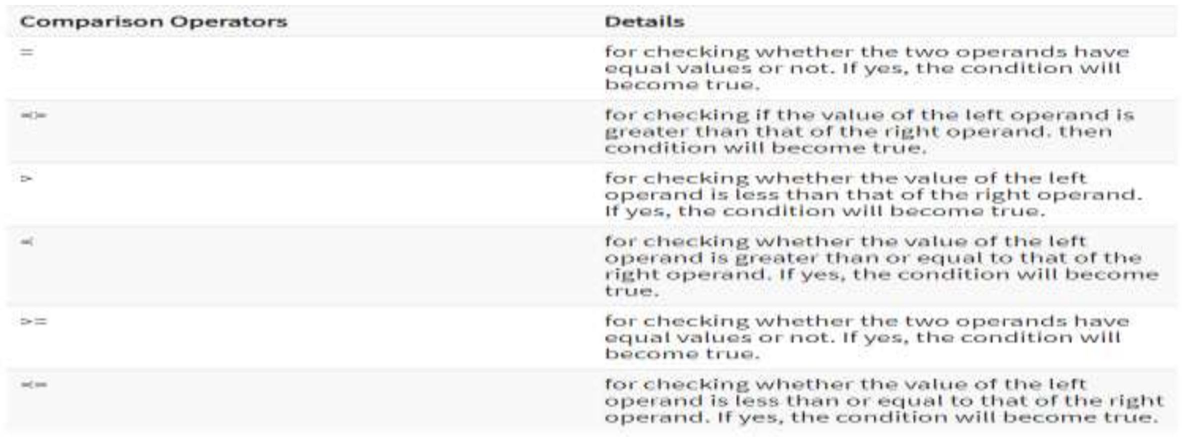
EXAMPLE

```
Sub Main()
Dim w As Integer = 11
Dim x As Integer = 5
Dim y As Integer
y = w + x
Console.WriteLine(y)
y = w - x
Console.WriteLine(y)
y = w * x
Console.WriteLine(y)
```

```
y = w / x
Console.WriteLine(y)
End Sub
```

COMPARISON OPERATORS

- These operators are used for making comparisons between variables. They include the following:



Comparison Operators	Details
=	for checking whether the two operands have equal values or not. If yes, the condition will become true.
<>	for checking if the value of the left operand is greater than that of the right operand. then condition will become true.
>	for checking whether the value of the left operand is less than that of the right operand. If yes, the condition will become true.
<	for checking whether the value of the left operand is greater than or equal to that of the right operand. If yes, the condition will become true.
>=	for checking whether the two operands have equal values or not. If yes, the condition will become true.
<=	for checking whether the value of the left operand is less than or equal to that of the right operand. If yes, the condition will become true.

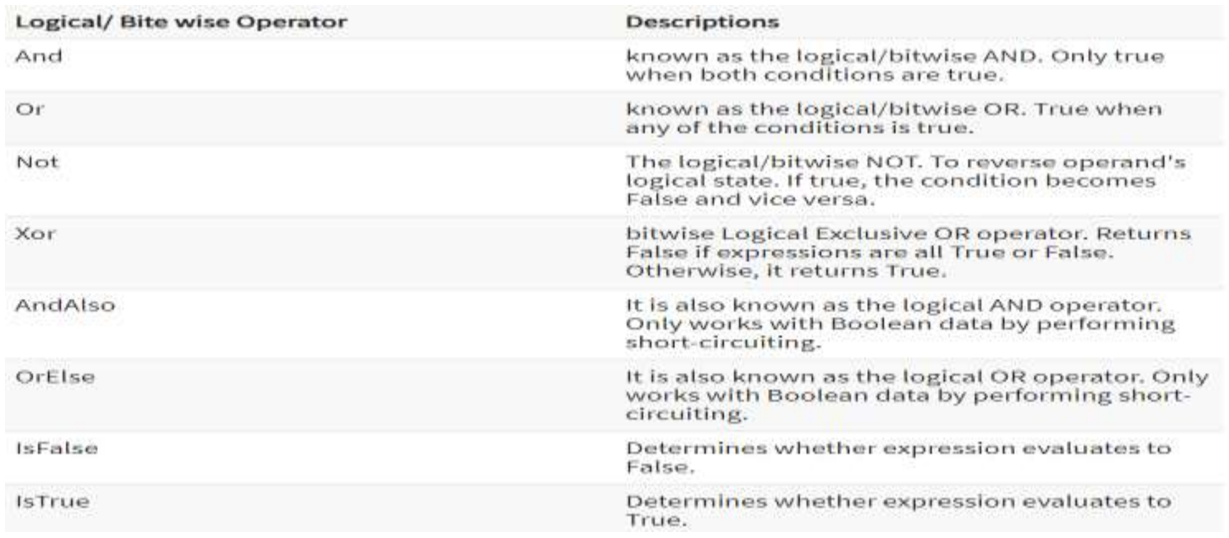
EXAMPLE

```
Sub Main()
Dim x As Integer = 11
Dim y As Integer = 5
If (x = y) Then
Console.WriteLine("11=5 is True")
Else
 Console.WriteLine(" 11=5 is False")
End If
 If (x < y) Then
 Console.WriteLine(" 11<5 is True")
```

```
  Else
Console.WriteLine(" 11<5 is False")
End If
```

LOGICAL/BITWISE OPERATORS

- These operators help us in making logical decisions. They include:



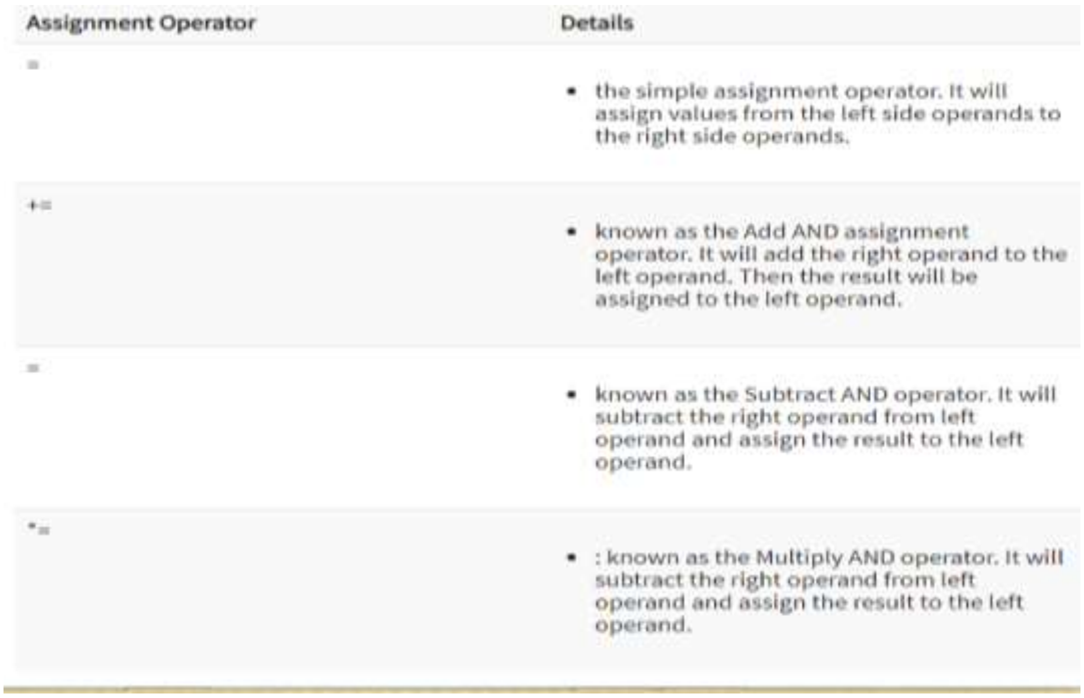
Logical/ Bite wise Operator	Descriptions
And	known as the logical/bitwise AND. Only true when both conditions are true.
Or	known as the logical/bitwise OR. True when any of the conditions is true.
Not	The logical/bitwise NOT. To reverse operand's logical state. If true, the condition becomes False and vice versa.
Xor	bitwise Logical Exclusive OR operator. Returns False if expressions are all True or False. Otherwise, it returns True.
AndAlso	It is also known as the logical AND operator. Only works with Boolean data by performing short-circuiting.
OrElse	It is also known as the logical OR operator. Only works with Boolean data by performing short-circuiting.
IsFalse	Determines whether expression evaluates to False.
IsTrue	Determines whether expression evaluates to True.

EXAMPLE

```
Sub Main()
Dim w As Boolean = True Dim x As Boolean = True Dim y As Integer = 5
Dim z As Integer = 20
  If (w And x) Then Console.WriteLine("w And x- is true")
End If
If (y Or z ) Then
Console.WriteLine("y Or z-is true")
End If
```

```
  If (w Xor x) Then
Console.WriteLine("w Xor x - is true")
End If
```

ASSIGNMENT OPERATORS



Assignment Operator	Details
=	• the simple assignment operator. It will assign values from the left side operands to the right side operands.
+=	• known as the Add AND assignment operator. It will add the right operand to the left operand. Then the result will be assigned to the left operand.
=	• known as the Subtract AND operator. It will subtract the right operand from left operand and assign the result to the left operand.
*=	• : known as the Multiply AND operator. It will subtract the right operand from left operand and assign the result to the left operand.

EXAMPLE

```
Sub Main()
Dim x As Integer = 5
Dim y As Integer
  y = x
  Console.WriteLine(" y = x ", y)
y += x
Console.WriteLine(" y += x ", y)
  y -= x
Console.WriteLine(" y -= x ", y)
```

End Sub

1.13 MODULARIZING YOUR CODE – FUNCTIONS AND SUBROUTINES

- Function and subroutines allow logical organization of your programs code.
- Function and subroutines are used to group together code statements that make up a task.

1.Function

- A function is a group of code statements that executed and give a result back to the code that invoked the function.
- To define a function in vb.net place any code statement between Function and End Function keywords.
- The syntax for the Function statement is −

```
Function FunctionName (ParameterList) As ReturnType
        [Statements]
End Function
```

FunctionName − indicates the name of the function

ParameterList − specifies the list of the parameters

ReturnType − specifies the data type of the variable the function returns

EXAMPLE

```
Function FindMax(ByVal num1 As Integer, ByVal num2 As Integer) As
Integer
Dim result As Integer
        If (num1 > num2) Then
```

```
            result = num1
        Else
          result = num2
          End If
End Function
```

2. SUBROUTINES

- Subroutines usually perform actions and they don't return any result.
- Functions, on the other hand, perform some calculations and return a value.
- This is the only difference between subroutines and functions. Both subroutines and functions can accept arguments, which are values you pass to the procedure when you call it.
- A subroutine is a block of statements that carries out a well-defined task.
- The block of statements is placed within a set of Sub. . .End Sub statements and can be invoked by name.
- The following subroutine displays the current date in a message box and can be called by its name, ShowDate():

Example:

```
    Sub ShowDate()
        MsgBox(Now().ToShortDateString)
     End Sub
```

1.14 CONTROLLING PROGRAM FLOW

- VB.NET provides a complete set of control structures for controlling program flow:
  1. Conditional processing
  2. Flow control statement and
  3. Loops

1. CONDITIONAL PROCESSING

- Conditional Processing tests an expression and executes code based on the value of that expression.

If…. Then …. Else Statements

- The If… Then .. Else statement, which executes one block of statements if the condition is True and, another if the condition is False.
- The syntax of the If… Then … Else statement is as follows:

SYNTAX:

If *expression* **Then**

.. Code to be Executed is expression is True ..

ElseIf *expression* **Then**

.. Code to be executed is expression is True

ElseIf *expression* **Then**

.. Code to be executed is expression is True Else

.. Code to be executed is expression is false

Endif

EXAMPLE

Dim myName As String = “ram"

If myName = "Fred" Then

```
                MessageBox.Show("Hello
          Fred")
           ElseIf myName = "John" Then
                MessageBox.Show("Hello
          John")
                ElseIf myName = "Paul" Then
                          MessageBox.Show("Hello Paul")
                Else
                          MessageBox.Show("unnamed")
End If
```

Select Case Statement :

- *Select Case* construct is extremely useful if you need to evaluate a value or expression against multiple possible outcomes.
- For example you might want to compare a string against a variety of other strings in order to decide what action to take.
- The syntax for the *Select Case* construct is as follows:

Select Case *expression*

    Case *value1, value2, ...*

        ...

    Case *value3, value3, ...*

        ...

    Case *value1, value2, ...*

        ...

End Select

EXAMPLE:

- The following code excerpt shows a *Select Case* construct which takes specific action depended on the value of a String variable:

```
Dim strCity As String = "London“
Dim strCountry As String
Select Case strCity
Case "Paris"

      strCountry = "France“
  Case "London"
      strCountry = "England“
  Case "Berlin"
      strCountry = "Germany"
End Select
      MessageBox.Show("Welcome to " & strCountry)
```

2. FLOW CONTROL STATEMENTS

- VB.NET flow control statements can enable you to branch your code unconditionally.
- There are two statements: Exit and Goto.

Exit statement:

- Use exit when you want to stop code execution within these code structures based on some circumstances.
- Exit takes following keywords: Do, For, Sub, Function, property and Try
- The syntax for the Exit statement is −

  Exit { Do | For | Function | Property | Select | Sub | Try | While }

Example:

```
Sub Main()
```

```
Dim a As Integer = 10
While (a < 20)
Console.WriteLine(a)
a = a + 1
If (a > 15) Then
'terminate   the   loop   using
exit statement Exit While
   End If
End While
Console.ReadLine()
End Sub
```

Goto Statement:

- *GoTo* statement provides a way for program execution to jump to a labeled location in the code.

Example:

```
Private Sub ValidateValue(ByVal intValue As Integer) If intValue > 10 Then
      GoTo EXIT_SUB
else ProcessValue(intValue)
      MessageBox.Show("Valid Number Entered")
Endif
EXIT_SUB:
Exit Sub
End Sub
```

3. Loops

- Loops execute a block of code a predetermined number of times or until a particular test condition has been met.

Do….. Loop Loops.

- A Do...Loop Until loop is a loop that runs until the loop's condition is true, the condition being checked after each iteration of the loop.

Syntax:

Do

Loop code here **Loop** Until condition

EXAMPLE

```
Module loops


    Sub Main()
        Dim a As Integer = 10 Do
        Console.WriteLine("value of a:{0}", a)
        a = a + 1
        Loop While (a < 20)
        Console.ReadLine()
    End Sub
End Module
```

```
Output:  value of a: 10
value of a: 11
 value of a: 12
value of a: 13
value of a: 14
value of a: 15
 value of a: 16
value of a: 17
value of a: 18
value of a: 19
```

For …. Next Loops

- When the number of cycles is know before the loop is initiated, we can use the For

  Next statements.
- In this construct we declare a counter variable, which is automatically increased or decreased in value during each repetition of the loop.

Syntax:

For

Loop code here

Next

EXAMPLE

```
Module Module1 Sub Main()
Dim I as interger =1
  For i As Integer = 0 To 2
     Console.WriteLine("I: {0}", i)
  Next
  End Module
```

For Each statement

- The For Each construct simplifies traversing over collections of data.
- It has no explicit counter.
- The For Each statement goes through the array or collection one by one and the current value is copied to a variable defined in the construct.

```
           Module Example
Sub Main()
```

```
Dim planets() As String = { "Mercury", "Venus“, "Earth", "Mars", "Jupiter",
        "Saturn", "Uranus", "Neptune" }
For Each planet As String In planets
        Console.WriteLine(planet)
Next
End Sub
End Module
```

- The usage of the For Each statement is straightforward.
- The planets is the array that we iterate through.
- The planet is a temporary variable that has the current value from the array.
- The For Each statement goes through all the planets and prints them to the console.

While ….. End While Loops

- The While statement is a control flow statement that allows code to be executed repeatedly based on a given boolean condition.
- This is the general form of the While loop:

```
While (expression):
 statement
End While
```

  - The While keyword executes the statements inside the block enclosed by the While, End While keywords.
  - The statements are executed each time the expression is evaluated to true.

EXAMPLE

```
Module Example
Sub Main()
```

```
Dim i As Integer = 0

Dim sum As Integer = 0
     While i < 10
  i = i + 1
sum += I
  End While
Console.WriteLine(sum)
End Sub
End Module
```

1.16 HANDLING ERRORS AND EXCEPTIONS

- Catching errors and other extraordinary conditions in code has always been a problem for programmers.
- Often errors occur and go unnoticed during program execution.

1. UNSTRUCTURED ERROR HANDLING:

- This is a simple and sometimes effective way to handle errors. It works like this:
- You can call a function and check the return value.
- If the return value corresponds to a defined error result code, you can display an error message or perform whatever action is appropriate for the type of error that occurred.

EXAMPLE

```
If totalline() = success then
          success=writetotal()
  If succes then
```

```
                    if sendnotice() then
                    'some
                          else
                            success = notify()
                         end if
                    end if
               Else
               'error
               End if
```

- To turn off unstructured error handling, you can use the On Error GoTo 0 or On Error GoTo -1 statements (they do the same thing). Here's an example:

```
Sub Main()
On Error GoTo Handler
Dim intItem1 As Integer = 0
Dim intItem2 As Integer = 128
Dim intResult As Integer
intResult = intItem2 / intItem1
On Error GoTo 0
     Console.WriteLine("Press Enter to continue...")
     Console.ReadLine()
     Exit Sub
   Handler:
          Console.WriteLine("An overflow error occurred.")
Resume Next
 End Sub
```

2. STRUCTURED EXCEPTION HANDLING

- VB.NET utilizes the .NET Framework's standard mechanism for error reporting, called Structured Exception Handling.
- It relies on exceptions to report errors that arise in applications.
- Exceptions are classes that trap the error information.
  - To utilize .NET's Structured Exception Handling mechanisms properly, developers need to write smart code that watches out for exceptions and implement code to deal with these exceptions.
  - Structured exception handling provides the following components in the code:

    Try section: The block of code that may result in an exception and always gets executed

    Catch section: The block of code that attempts to act on an exception and is only executed when an exception takes place

    Finally section: The block of code intended to perform any kind of clean up operation and always gets executed

THE TRY...CATCH BLOCK

- The purpose of the *Try...Catch* block is to allow catching errors and specifying a resolution for them. The sample code looks like this:

```
Try
    'Code to be executed
Catch
    ‘Error resolution code
 End Catch
```

THE TRY...CATCH...FINALLY BLOCK

- The purpose of the *Try...Catch...Finally* block is to allow executing the protected code under the Try section, acting on any errors that may arise in the *Catch* block, and following up with the cleanup code in the *Finally* block.

```
Try
    'Code to be executed
 Catch
    'Error resolution code
 Finally
    'Cleanup code
End Catch
```

EXAMPLE

```
Try
    Dim i As Integer = 0
    Dim iresult As Integer
      iresult = 1 / i
       Catch          ex          As          Exception
    MessageBox.Show(ex.ToString())
 Finally
    MessageBox.Show("finally block executed")
    End Try
```

1.16 OBJECT ORIENTED PROGRAMMING

- .NET is fully object oriented platform that allow languages to take full advantage of these OOP features. The features include: Namespace
  - ✓ Classes
  - ✓ Class Properties
  - ✓ Constructors and destructors
  - ✓ Inheritance
  - ✓ Overriding

- ✓ Overloading
- ✓ Polymorphism
- ✓ Interfaces

CLASSES

- ➢ Class is nothing but a template or blue-print for an entity.
- ➢ For example you may have a class that represents real life entity - Employee.
- ➢ The class will provide properties (Name, Age...) as well as actions (CalculateSalary, GoOnLeave...) of the entity.
- ➢ Creating classes is similar to creating namespaces.

```
[VB.NET]
Public Class Class1
      ...
End Class
```

CLASS PROPERTIES

- ➢ Properties encapsulate data members of your class. Properties can be read-write, read only or write only.
- ➢ Here is how you create read-write properties:

```
Public Class Employee
 private strName As String
Public  Property  Name  As  String
Get
        return strName;
End
  Get
```

```
        Set(value  As  String)  strName=value;
End Set
End Property
End Class
```

VB.NET uses Property keyword to declare properties.

- Property definition consists of two parts Get and Set. The get part returns the property value and set part sets some private variable.
- The value in Set routine is received via implicit variable called value in C#. VB.NET allows you to change this.

CONSTRUCTORS AND DESTRUCTORS

- ***Constructors*** used in a class are member functions to initialize or set the objects of a class in VB.net.
- They don’t return any value and are defined in a **Sub** with a keyword **New**.
- Multiple constructors can be created in class with any access specifiers, by default constructors are of **Public** access type.

EXAMPLE

```
  Public Sub New(ByVal setval As Integer)
  a = setval
End Sub
      Public Function disp()
                Return a
      End Function
```

- You can think of destructors as the opposite of constructors: Constructors execute when objects are created, and Destructors execute when the objects are destroyed.

- ***Destructors or finalizer*** is a method used to deallocate the resources of an object that are no longer used, its invoked automatically by the VB.net environment. The keyword **Overrides** is used with the Finalizer method.

Example:

```
 Protected Overrides Sub Finalize()
Console.WriteLine("Calling Destructor")
End Sub
```

INHERITANCE

- One of the most important concepts in object-oriented programming is that of inheritance.
- Inheritance allows us to define a class in terms of another class which makes it easier to create and maintain an application.
- When creating a class, instead of writing completely new data members and member functions, the programmer can designate that the new class should inherit the members of an existing class.
- This existing class is called the **base** class, and the new class is referred to as the **derived** class.

EXAMPLE

```
‘Base class
    Class Shape
            Dim width , height As Integer
                  Public Sub setWidheig(ByVal w As Integer, ByVal h As
            Integer)
                        width = w
                         height = h
                  End Sub
    End Class
‘derived class
```

```
Class Rectangle : Inherits Shape
        Public Function getArea() As Integer
                Return (width * height)
        End Function
End Class
```

OVERRIDING

- ***Overriding*** in VB.net is method by which a inherited property or a method is overidden to perform a different functionality in a derived class.
- The base class function is declared using a keyword Overridable and the derived class function where the functionality is changed contains an keyword Overrides.

EXAMPLE

```
       Public Overridable Function add(x As Integer, y As Integer)
          Console.WriteLine('Function Inside Base Class')
          Return (x + y)
     End Function
    Inherits Over
        Public Overrides Function add(x As Integer,y As Integer)
                Console.WriteLine(MyBase.add(120, 100))
                Console.WriteLine('Function Inside Derived Class')
                Return (x + y)
        End Function
```

OVERLOADING

- Method Overloading allows us to write different versions of a method (i.e. a single class can contain more than one methods (Sub / Functions) of same name but of different implementation).

- And compiler will automatically select the appropriate method based on parameters passed.

Example:

```
Public Overloads
Function add(a As Long, b As Long)
                WriteLine("You are in function add(long, long)") Return a + b
  End Function
```

POLYMORPHISM

- Polymorphism is one of the crucial features of VB.NET, It means "The ability to take on different form", It is also called as Overloading which means the use of same thing for different purposes.
- Using Polymorphism we can create as many functions we want with one function name but with different argument list.
- The function performs different operations based on the argument list in the function call.
- The exact function to be invoked will be determined by checking the type and number of arguments in the function.

INTERFACE

- ***Interfaces in VB.net*** are used to define the class members using a keyword **Interface**, without actually specifying how it should be implemented in a Class.
- Intefaces are examples for multiple Inheritance and are implemented in the classes using the keyword **Implements** that is used before any Dim statement in a class.

Example

```
 Public Interface Interface1
 Function Add(ByVal x As Integer) As Integer
End Interface
```

1.17 MULTITHREADING

- A thread is basically a path of execution through a program. It is also the smallest unit of execution that Win32 schedules.
- A thread consists of a stack, the state of the CPU registers and an entry in the execution list of the system scheduler.
- Each thread shares all of the process's resources.
- A "process" is an executing instance of an application.
- For example, when you double-click the Notepad icon, you start a process that runs Notepad.
- A "thread" is a path of execution within a process.
- When you start Notepad, the operating system creates a process and begins executing the primary thread of that process.
- When this thread terminates, so does the process.
- This primary thread is supplied to the operating system by the startup code in the form of a function address.
- All threads in . NET applications are represented by **System.Threading** namespace.
- **System.Threading** provides classes and interfaces that enable multithreaded programming.
- This namespace includes a **ThreadPool** class that manages groups of threads, a **Timer** class that enables a delegate to be called after a specified amount of time and a **Mutex** class for synchronizing mutually exclusive threads.

- **System.Threading** also provides classes for thread scheduling and wait notification.
- Each thread in a process operates independently.
- Unless you make them visible to each other, the threads execute individually and are unaware of the other threads in a process.

EXAMPLE

```
    Dim thread As System.Threading.Thread
    Private Sub countup()
            Do Until i = 10000
          i = i + 1
        Label1.Text = i Me.Refresh()
    Loop
   End Sub
```

Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click

```
thread = New System.Threading.Thread(AddressOf countup)
thread.Start()
```

End Sub

Unit -2

2.1 The features of .NET

- The Microsoft .NET framework provides a lot of features.
- Microsoft has designed the features of the .NET framework by using the technologies that are required by software developers to develop applications for modern as well as future business needs.
- The key features of .NET are:

1. Common Executive Environment:-

- All .NET applications run under a common execution environment, called the Common Language Runtime.
- The CLR facilitates the interoperability between different .NET languages such as C#, Visual Basic, Visual C++, etc. by providing a common environment for the execution of code written in any of these languages.

2. Common Type System:-

- CTS ensures that objects of the programs that are written in different programming languages can communicate with each other to share data.
- The common type system CTS defines a set of types and rules that are common to all languages targeted at the CLR.
- It supports both value and reference types.

3. Multi-language support:-

- .NET provides multi-language support by managing the compilers that are used to convert the source to intermediate language (IL) and from IL to native code, and it enforces program safety and security.
- The basis for multiple language support is the common type system and metadata.
- The basic data types used by the CLR are common to all languages. There are therefore no conversion issues with the basic integer, floating-point and string types.

4. tool Support:-

- The CLR works hand-in-hand with tools like visual studio, compilers, debuggers, and profilers to make the developer's job much simpler.

5. Security:-

- The CLR manages system security through user and code identity coupled with permission checks.
- The identity of the code can be known and permission for the use of resources granted accordingly.
- This type of security is a major feature of .NET.

6. Automatic Resource Management:-

- The .NET CLR provides efficient and automatic resource management such as memory, screen space, network connections, database, etc.
- CLR invokes various built-in functions of .NET framework to allocate and de-allocate the memory of .NET objects.
- Therefore, programmers need not write the code to explicitly allocate and de-allocate memory to the program.

7. Easy and rich debugging support:-

- The .NET IDE (integrated development environment) provides an easy and rich debugging support.
- Once an exception occurs at run time, the program stops and the IDE marks the line which contains the error along with the details of that error and possible solutions.
- The runtime also provides built-in stack walking facilities making it much easier to locate bugs and error.

8. Simplified development:-

- With .NET installing or uninstalling, a window-based application is a matter of copying or deleting files.
- This possible because .NET components are not referenced in the registry.

9. Framework class library:-

- The framework class library (FCL) of the .NET framework contains a rich collection of classes that are available for developers to use these classes in code Microsoft has developed these classes to fulfill various tasks of applications.

- The classes in the FCL are logically grouped under various namespaces such as system, System.collections, system.Globalization, system.IO, system.text etc.

10. Portability:-

- The application developed in the .NET environment is portable. When the source code of a program written in a CLR compliant language complies, it generates a machine-independent and intermediate code.
- This was originally called the Microsoft Intermediate Language (MSIL) and has now been renamed as the common Intermediate Language (CIL).
- CIL is the key to portability in .NET.

2.2 The anatomy of asp.net pages

1. **The Code Structure of ASP.NET**

- ASP.NET pages begin with a plain text file that has an .aspx extension.
- IIS uses this extension to identify particular files that contain executable code and should be processed as ASP.NET pages.
- The contents of the .aspx file are called the page.
- The page consists of visual markup(HTML) and application logic (executable code).
- ASP.NET also has the ability to store application code in separate files rather than grouping it with HTML. This Concepts is called **Code Behind**.
- **Example**

```
<%@ Page Language="VB" AutoEventWireup="false"
CodeFile="Default.aspx.vb" Inherits="_Default" %>  ------ 1
<html xmlns="http://www.w3.org/1999/xhtml" >
<body>
   <form id="form1" runat="server"> ------- 2
   <div>
 <asp:Label ID="Label2" runat="server" Text="Roll no">
</asp:Label> -------- 3
```

- This .aspx file contains Three new items of interest:

i. ASP.NET page directives (shown in line 1):

- Page directives provide ASP.NET with the information about the code contained in the file.
- They can located anywhere in the file,but they are generally placed at the top.

ii. The runat="Server" attributes (shown in line 2):

- Runat="server" attributes instructs the page compiler to preprocess these HTML elements on the server.

iii. Web Controls(Shown in line 3) :

- Web Controls are server side code pieces that perform a variety of functions.
- Web controls are user interface elements that interact with the server.

2. Execution Stages and State Management

- Web application use a request – response model of communication:
    - ✓ The Web browser request a document from the web server and
    - ✓ the web server responds with the data from that document.
- Web application also contain HTML forms, which allow for user input.
- The program is executed using the form data as input and the results are returned to the browser as HTML. This Sequence of steps is referred to as **Round-Trip**
- Many state Management tasks in ASP.NET are done for you and other state management tasks can be accomplished in a number of ways.
- The ASP.NET web controls and HTML Controls can maintain their associated values between round trip with any code is called view state.
- The view state hidden form field, which is automatically generated by ASP.NET for every .aspx file that contain server side form.

3.Events model for the page class

- The page class contains events that fire when

1. The page loads into the browser and
2. When server code execution ends and page rendering has finished.

- These events are page_load and page_unload.
- You can specify event handling code for these events in your .aspx file.

Example

```
Sub page_load(sender as object,E as EventArgs)
  Label.text = “welcome”
End sub
Sub page_unload(sender as object,E as EventArgs)
  Label.text = “ ”
End sub
```

2.3 Introducing Web Forms

- Web Forms are pages that your users request using their browser.
- These pages can be written using a combination of HTML, client-script, server controls, and server code.
- When users request a page, it is compiled and executed on the server by the framework, and then the framework generates the HTML markup that the browser can render.
- An ASP.NET Web Forms page presents information to the user in any browser or client device.
- Using Visual Studio, you can create ASP.NET Web Forms.
- The Visual Studio Integrated Development Environment (IDE) lets you drag and drop server controls to lay out your Web Forms page. You can then easily set properties, methods, and events for controls on the page or for the page itself.
- These properties, methods, and events are used to define the web page's behavior, look and feel, and so on.

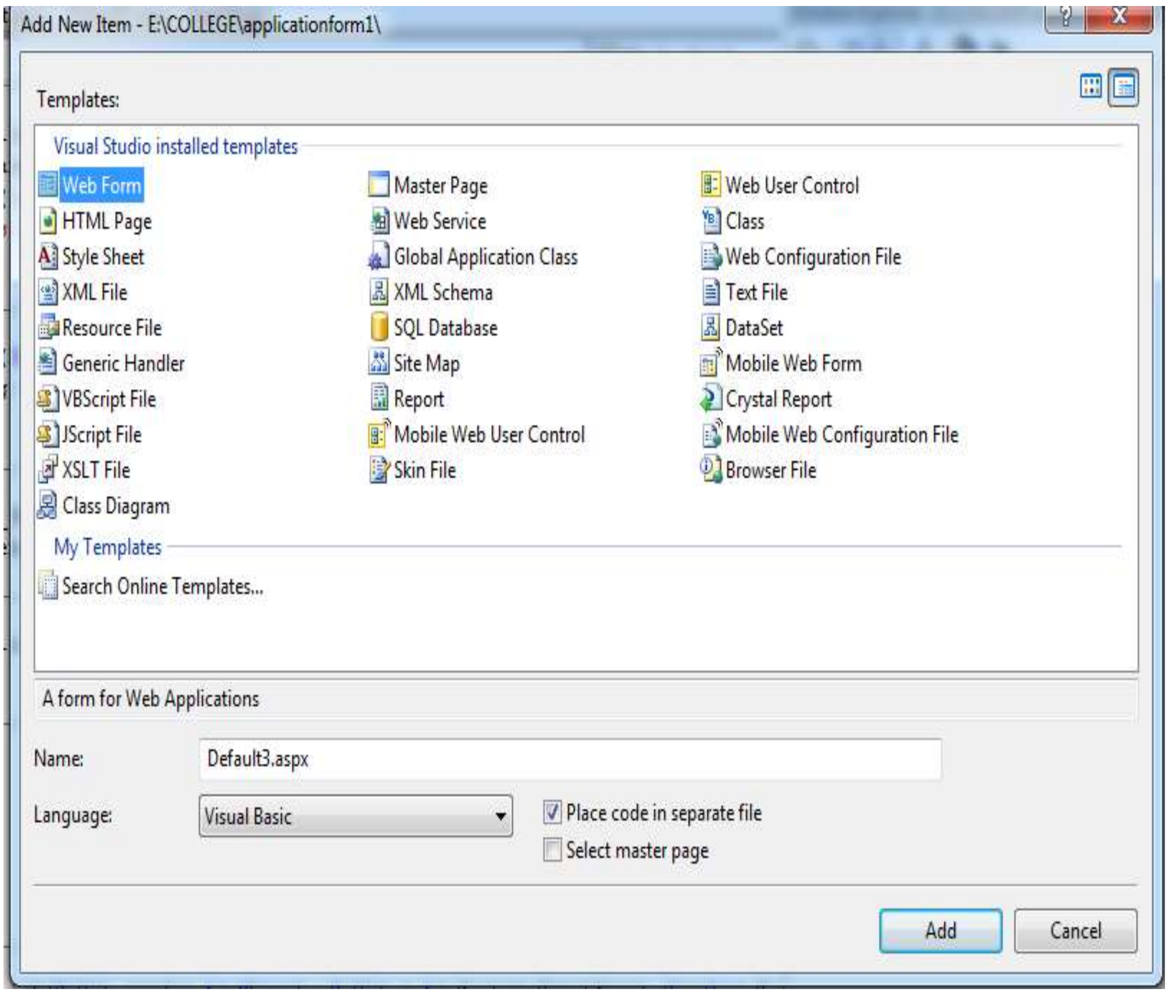
ASP.NET Web Forms are:

1. Based on Microsoft ASP.NET technology, in which code that runs on the server dynamically generates Web page output to the browser or client device.
2. Compatible with any browser or mobile device. An ASP.NET Web page automatically renders the correct browser-compliant HTML for features such as styles, layout, and so on.
3. Compatible with any language supported by the .NET common language runtime, such as Microsoft Visual Basic and Microsoft Visual C#.

4. Built on the Microsoft .NET Framework. This provides all the benefits of the framework, including a managed environment, type safety, and inheritance.
5. Flexible because you can add user-created and third party controls to them.

2.4 VS.NET Web Application and other IDE Basics

- VS.NET offers an easy way to create new ASP.NET web application projects.
- You can create a new web application project in VS.NET by selecting FILE→New→Project from the menu bar.
- Adding new web forms to the application is simple.
- In the Solution Explorer window right click on the icon in web application and select Add→Add new item, the dialog box appear.
- In that select Add Web Form as shown below


Add New Item - E:\COLLEGE\applicationform1\

Templates:

Visual Studio installed templates

Web Form	Master Page	Web User Control
HTML Page	Web Service	Class
Style Sheet	Global Application Class	Web Configuration File
XML File	XML Schema	Text File
Resource File	SQL Database	DataSet
Generic Handler	Site Map	Mobile Web Form
VBScript File	Report	Crystal Report
JScript File	Mobile Web User Control	Mobile Web Configuration File
XSLT File	Skin File	Browser File
Class Diagram		

My Templates

Search Online Templates...

A form for Web Applications

Name: Default3.aspx

Language: Visual Basic

[✓] Place code in separate file

[ ] Select master page

- After select click ok button, then the form is add to the Solution Explorer

- Whenever you want to view the code or when you place a web control on the webform for the first time,a new code module is added to the solution that serves as the code behind module for the web form.
- Example

```
Partial Class _Default------1
   Inherits System.Web.UI.Page------2
Protected Sub Page_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Load------3
        Clear()
End Sub
```

- Assign the class the same name given the web form as shown in line 1
- This class inherits from the page class as shown in line 2,which predefined ASP.NET class.
- Inherited page event handlers defined for page_load in line 3.

2.5 Separating Content And Code – The Code Behind Feature

- VS.NET also offers many features to keep your code organized such as the code behind features.
- Many programmers combined the two elements within the same file.
- The result was code base that are difficult to maintain and mixture of HTML and script.
- The two files are linked together using a special page directive, code behind,which defined in the .aspx file.
- Code behind specifies the pathname to a code module file.
- **Example**:

```
<%@ Page Language="VB" AutoEventWireup="false"
Codebehind="samplewebbutton.vb"
Inherits="webapplication1.samplewebbutton" %>
```

- You have other options when specifying a cade behind class.you may also specify an src attribute,which tells ASP.NET where your class file is located.
- If omitted ASP.NET assumes that the class is located in /bin directory of web application.

```
<%@ Page Language="VB" AutoEventWireup="false"
Src="samplewebbutton.vb"
           Inherits="webapplication1.samplewebbutton"
 %>
```

2.6 Application configuration

- Application in ASP.NET are comprised of a virtual web directory and pages ,file and assemblies contained within the directory and subdirectory.
- You can define setting for this application plus write event handlers for application wide events.
- A special file called Global.asax file control this setting events.

Structure and configuration of the global.asax file

- The Global.asax file contain code that may compromise the security of the web server.
- ASP.NET place restriction on the global.asax file,so that any direct HTTP requests for it is automatically denied.
- When VS.NET creates a new web application,a Global.asax file is created automatically.
- The Global.asax file contains a code behind file name Global.asax.vb
- Example:

  ```
  <% Application codebehind = "Global.asax.vb" Inherits = "lab3_1.Global"
  %>
  ```

  The code behind file shown as

  ```
            Import System.Web
            Import System.Web.SessionState
             \\Public class Global
            Inherits system.web.Httpapplicaion
  ```

2.7 Using HTml Controls

- The HTML server controls are HTML elements that include a runat=server attribute. The HTML server controls have the same HTML output and the same properties as their corresponding HTML tags. In addition, HTML server controls provide automatic state management and server-side events. HTML server controls offer the following advantages:

- The HTML server controls map one to one with their corresponding HTML tags.
- When the ASP.NET application is compiled, the HTML server controls with the runat=server attribute are compiled into the assembly.
- Most controls include an OnServerEvent for the most commonly used event for the control. For example, the <input type=button> control has an OnServerClick event.
- The HTML tags that are not implemented as specific HTML server controls can still be used on the server side; however, they are added to the assembly as HtmlGenericControl.
- When the ASP.NET page is reposted, the HTML server controls keep their values.
- The System.Web.UI.HtmlControls.HtmlControl base class contains all of the common properties. HTML server controls derive from this class.

1.HtmlAnchor Control

- Use the **HtmlAnchor** control to programmatically control an **<a>** HTML element.
- The **<a>** HTML element allows you to create a hyperlink that allows you to move to another location on the page or to another Web page.
- The **HtmlAnchor** control must be well formed with an opening and closing tag. You can specify the caption for the control by placing text between the opening and closing tags.
- **Example**:

```
<a id="programmaticID" href="linkurl" name="bookmarkname“
runat="server" > linktext </a>
```

2. HtmlButton Control

- Use the **HtmlButton** control to program against the HTML **<button>** element. You can provide custom code for the **ServerClick** event of the **HtmlButton** control to specify the action performed when the control is clicked.

- The HTML 4.0 **<button>** element enables you to create buttons composed of embedded HTML elements
- Creates a server-side control that maps to the **<button>** HTML element and allows you create push buttons.
- Example: <button runat="server"> </button>

3. HtmlForm control

- Use the **HtmlForm** control to program against the HTML **<form>** element.
- To take advantage of the postback services, all Web Forms controls, whether HTML, Web, pagelet, or custom, must be nested between well-formed opening and closing tags of the **HtmlForm** control.
- If the tags are not closed properly, ASP.NET will not recognize the element.
- By default, the **HtmlForm** control's **method** attribute is set to **POST**.
- Example
  <form id="programmaticID" method=POST | GET action="srcpageURL" runat="server" > Other controls, input forms, and so on.
  </form>
- You can customize the **method** attribute to suit your needs, but setting the **method** attribute to a value other than **GET** or **POST** can break the built-in view state and post back services provided by the Web Forms.

4. HtmlImage control

- Use the **HtmlImage** control to program against the HTML **<img>** element.
- This control allows you to dynamically set and retrieve the image's source, width, height, border width, alternate text, and alignment by using the **Src**, **Width**, **Height**, **Border**, **Alt**, and **Align** properties, respectively.
- Creates a server-side control that maps to the **<img>** HTML element and allows you to display an image.
- Example
  <img id="programmaticID"
  alt="alttext“

align= top | middle | bottom | left | right border="borderwidth"
height="imageheight"
src="imageURL"
width="imagewidth"
runat="server" >

2.7 Using HTml Controls

- The HTML server controls are HTML elements that include a runat=server attribute. The HTML server controls have the same HTML output and the same properties as their corresponding HTML tags. In addition, HTML server controls provide automatic state management and server-side events. HTML server controls offer the following advantages:

- ✓ The HTML server controls map one to one with their corresponding HTML tags.
- ✓ When the ASP.NET application is compiled, the HTML server controls with the runat=server attribute are compiled into the assembly.
- ✓ Most controls include an OnServerEvent for the most commonly used event for the control. For example, the <input type=button> control has an OnServerClick event.
- ✓ The HTML tags that are not implemented as specific HTML server controls can still be used on the server side; however, they are added to the assembly as HtmlGenericControl.
- ✓ When the ASP.NET page is reposted, the HTML server controls keep their values.
- ✓ The System.Web.UI.HtmlControls.HtmlControl base class contains all of the common properties. HTML server controls derive from this class.

1.HtmlAnchor Control

- Use the **HtmlAnchor** control to programmatically control an **<a>** HTML element.
- The **<a>** HTML element allows you to create a hyperlink that allows you to move to another location on the page or to another Web page.

- The **HtmlAnchor** control must be well formed with an opening and closing tag. You can specify the caption for the control by placing text between the opening and closing tags.
- **Example**:
  <a id="programmaticID" href="linkurl" name="bookmarkname“ runat="server" > linktext </a>

2. HtmlButton Control

- Use the **HtmlButton** control to program against the HTML **<button>** element. You can provide custom code for the **ServerClick** event of the **HtmlButton** control to specify the action performed when the control is clicked.
- The HTML 4.0 **<button>** element enables you to create buttons composed of embedded HTML elements
- Creates a server-side control that maps to the **<button>** HTML element and allows you create push buttons.
- Example: <button runat="server"> </button>

3. HtmlForm control

- Use the **HtmlForm** control to program against the HTML **<form>** element.
- To take advantage of the postback services, all Web Forms controls, whether HTML, Web, pagelet, or custom, must be nested between well-formed opening and closing tags of the **HtmlForm** control.
- If the tags are not closed properly, ASP.NET will not recognize the element.
- By default, the **HtmlForm** control's **method** attribute is set to **POST**.

  Example

  <form id="programmaticID" method=POST | GET action="srcpageURL" runat="server" > Other controls, input forms, and so on.

  </form>
- You can customize the **method** attribute to suit your needs, but setting the **method** attribute to a value other than **GET** or **POST** can break the built-in view state and post back services provided by the Web Forms.

4. HtmlImage control

- Use the **HtmlImage** control to program against the HTML **<img>** element.
- This control allows you to dynamically set and retrieve the image's source, width, height, border width, alternate text, and alignment by using the **Src**, **Width**, **Height**, **Border**, **Alt**, and **Align** properties, respectively.
- Creates a server-side control that maps to the **<img>** HTML element and allows you to display an image.

  Example

```
<img id="programmaticID"
        alt="alttext“
         align= top | middle | bottom | left | right border="borderwidth"
        height="imageheight"
        src="imageURL"
         width="imagewidth"
        runat="server" >
```

5. HtmlGenericControl

- Created on the server in response to tags that include the **runat="server"** attribute/value pair in elements that do not map directly to a specific HTML control.
- These elements include the **<span>, <body>**, **<div>**, and **<font>** elements, among others.
- Creates a server-side control that maps to an HTML element not represented by a specific .NET Framework class, such as **<body>** and **<div>**.
- Syntax:

```
        <span | body | div | font | others id="programmaticID" runat="server" >
Contentbetweentags
  </span | body | div | font | others>
```

- **Example :**

```
  <span id="MySpan" runat="server" />
  Hai
</span>
```

6. HtmlInputButton Control

- Use the **HtmlInputButton** control to program against the **<input type=button>**, **<input type=submit>**, and **<input type=reset>** HTML elements.
- When a user clicks an **HtmlInputButton** control, input from the form that the control is embedded on is posted to the server and processed.
- **Syntax**:

  <input type=button | submit | reset id="programmaticID" OnServerClick="onserverclickhandler" runat="server" >
- **Example**:

  <input Type="Submit" Name="AddButton Value="Add" OnServerClick="AddButton_Click" runat="server"/>

7. **HtmlInputCheckBox Control**

- Use the **HtmlInputCheckBox** control to program against the **<input type=checkbox>** HTML element.
- The **HtmlInputCheckBox** control does not post back to the server when it is clicked.
- The state of the check box is sent to the server for processing when you use a control that posts back the server, such as the **HtmlInputButton** control.
- To determine whether the check box is selected, test the **Checked** property of the control.
- **Syntax**:

  <input type=checkbox id="programmaticID" checked runat="server" >
- The following example demonstrates how to create an **HtmlInputCheckBox** control that allows the user to select a **true** or **false** state.

  <input id="Check1" type=checkbox runat="server" checked/> CheckBox1

8. HtmlInputFile Control

- Use the **HtmlInputFile** control to program against the HTML **<input type=file>** element. You can use the **HtmlInputFile** control to easily design a page that allows users to upload binary or text files from a browser to a directory that you designate on your Web server.

- Syntax:
- <input type=file id="programmaticID“ maxlength="maxfilepathlength" size="widthoffilepathtextbox" postedfile="uploadedfile" runat="server" >
- Example:

  <h1>ASP.NET File Upload Example</h1>

  Select File To Upload to Server:

  <input id="MyFile" type="file" runat="server">

9. HtmlInputHidden Control

- Use the **HtmlInputHidden** control to program against the **<input type=hidden>** HTML element.
- Although this control is part of the form, it is never displayed on the form.
- Creates a server-side control that maps to the **<input type=hidden>** HTML element and allows you to store information in a nonviewable control on the form.
- **Syntax**:

  <input type="hidden" id="programmaticID" value="contentsofhiddenfield" runat="server" >
- **Example:**

  <input id="HiddenValue" type=hidden value="Initial Value" runat="server">

10. HtmlInputImage Control

- Use the **HtmlInputImage** control to program against the HTML **<input type=image>** element. You can use this control in conjunction with the **HtmlInputText**, **HtmlTextArea**, and other controls to construct user input forms.
- This control offers an alternative for browsers that do not support DHTML and the **HtmlButton** control.
- Creates a server-side control that maps to the **<input type=image>** HTML element and allows you to create an button that displays an image.

- **Syntax**:
  <input type=image id="programmaticID" src="imagepath" align="imagealignment" alt="alttext" OnServerClick="onserverclickhandler" width="borderwidthinpixels" runat="server" >
- Example:
  <input type=image id="InputImage2" src="/images/mango.jpg" onmouseover="this.src='/images/banana.jpg';" onmouseout="this.src='/images/mango.jpg';" OnServerClick="Button2_Click" runat="server">

11. HtmlInputRadioButton Control

- Use the **HtmlInputRadioButton** control to program against the HTML **<input type=radio>** element.
- You can group multiple **HtmlInputRadioButton** controls together by setting the **Name** property to a value that is common to all **<input type=radio>** elements within the group.
- Radio buttons in the same group are mutually exclusive; only one radio button in the group can be selected at a time.
- **Syntax**:
  <input type=radio id="programmaticID" checked name="radiobuttongroup" runat="server" >
- **Example**:
  <input type="radio" id="Radio1" name="Mode" checked runat="server"/> Option 1<br>
  <input type="radio1" id="Radio2" name="Mode1" checked runat="server"/> Option 2<br>

12. HtmlInputText Control

- Use the **HtmlInputText** control to run server code against the **<input type=text>** and **<input type=password>** HTML elements.
- As with standard HTML, these controls can be used to enter user names and passwords in HTML forms.

- You can use this control in conjunction with the **HtmlInputButton**, **HtmlInputImage**, or **HtmlButton** control to process user input on the server.
- Creates a server-side control that maps to the **<input type=text>** and **<input type=password>** HTML elements and allows you to create a single line text box to receive user input.
- **Syntax**:

  <input type=text | password id="programmaticID" maxlength="max#ofcharacters" size="widthoftextbox" value="defaulttextboxcontents" runat="server" >
- Example:

  <input ID="Value1" Type="Text" Size="2" MaxLength="3" Value="1" runat="server"/>

  <input ID="Value1" Type=“password" Size=“4" MaxLength=“8“ runat="server"/>

13. HtmlSelect Control

- Use the **HtmlSelect** control to program against the HTML **<select>** element. By default, this control renders as a drop-down list box.
- To determine the selected item from a single selection **HtmlSelect** control, first use the **SelectedItem** property to obtain the index of the selected item. You can then use this index to retrieve the selected item from the **Items** collection.
- Creates a server-side control that maps to the **<select>** HTML element and allows you to create a list control.
- Syntax:

  <select id="programmaticID“ Value="currentitemvalue" runat="server" > <option>value1</option> <option>value2</option> </select>
- Example:

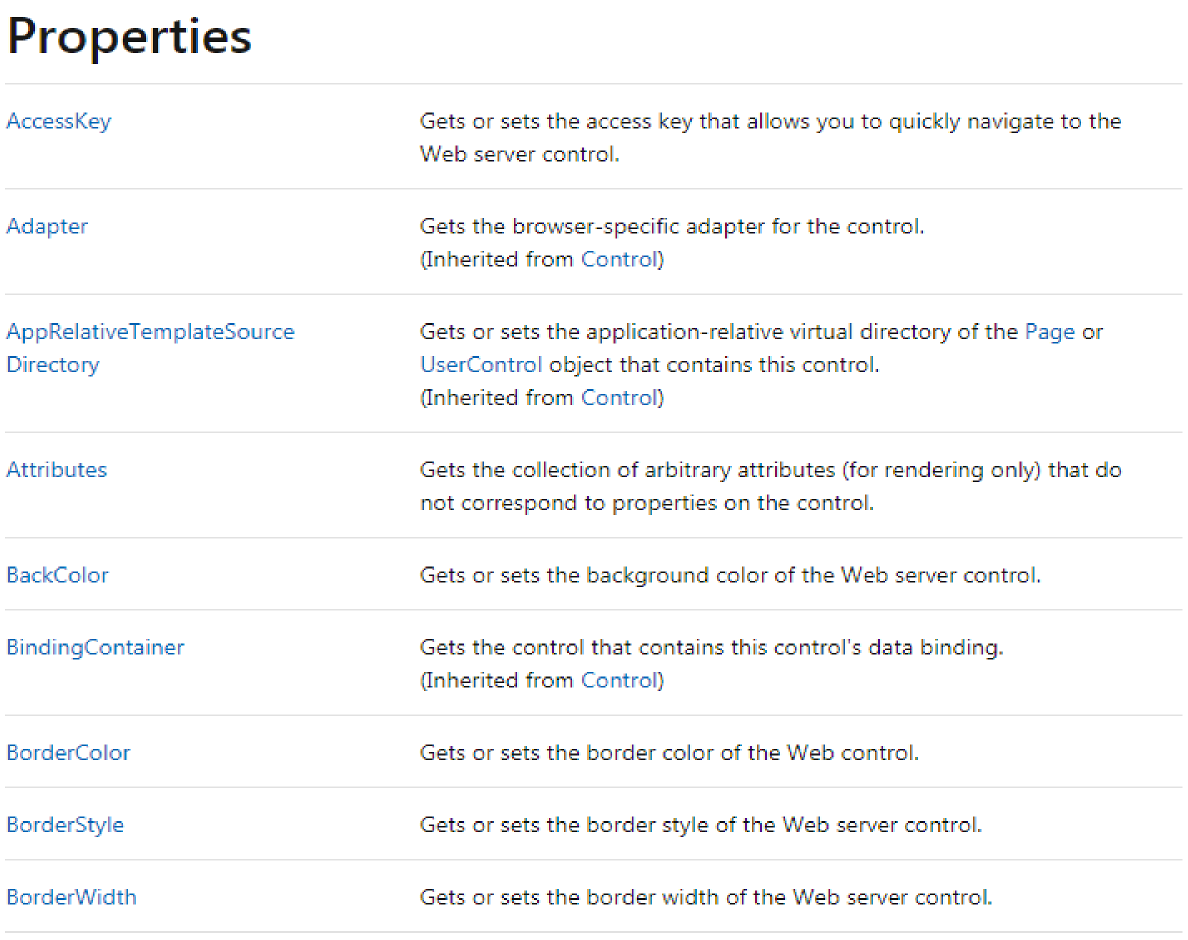
  <select id="ColorSelect" runat="server"> <option>SkyBlue</option> <option>LightGreen</option> <option>Gainsboro</option> <option>LemonChiffon</option> </select>

2.8 Using web Controls

- Web Controls provide richer functionality than their HTML control counter parts.
- By using Web controls instead of HTML controlsin functionality and visual appearance defintion are gained.

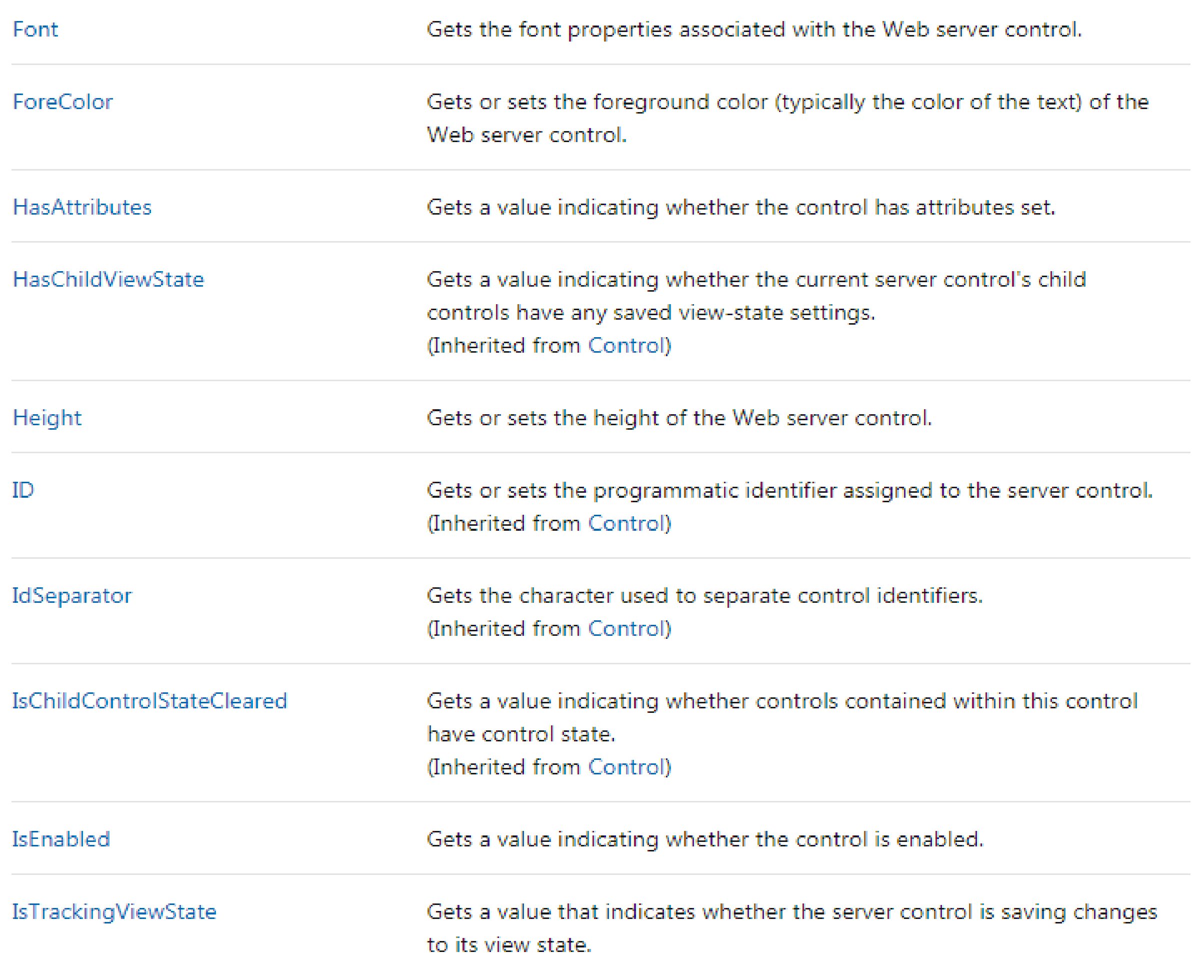
1. **Shared web controls:**

- All the ASP.NET web controls share rich formatting options.Some of characteristics you can programmatically adjust include color,border styles,fonts,text styles and many more shared web control properties are listed below:

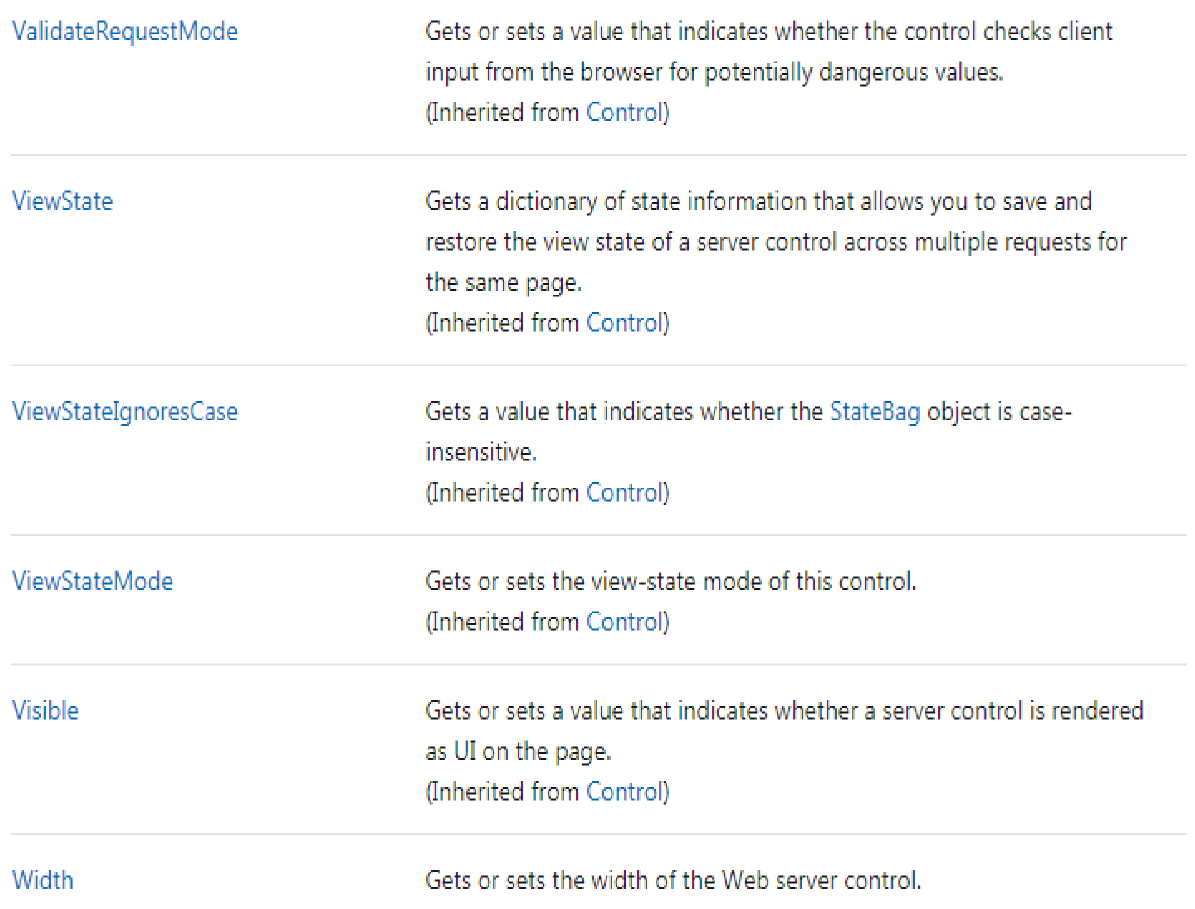


Properties

AccessKey	Gets or sets the access key that allows you to quickly navigate to the Web server control.
Adapter	Gets the browser-specific adapter for the control. (Inherited from Control)
AppRelativeTemplateSource Directory	Gets or sets the application-relative virtual directory of the Page or UserControl object that contains this control. (Inherited from Control)
Attributes	Gets the collection of arbitrary attributes (for rendering only) that do not correspond to properties on the control.
BackColor	Gets or sets the background color of the Web server control.
BindingContainer	Gets the control that contains this control's data binding. (Inherited from Control)
BorderColor	Gets or sets the border color of the Web control.
BorderStyle	Gets or sets the border style of the Web server control.
BorderWidth	Gets or sets the border width of the Web server control.



Font	Gets the font properties associated with the Web server control.
ForeColor	Gets or sets the foreground color (typically the color of the text) of the Web server control.
HasAttributes	Gets a value indicating whether the control has attributes set.
HasChildViewState	Gets a value indicating whether the current server control's child controls have any saved view-state settings. (Inherited from Control)
Height	Gets or sets the height of the Web server control.
ID	Gets or sets the programmatic identifier assigned to the server control. (Inherited from Control)
IdSeparator	Gets the character used to separate control identifiers. (Inherited from Control)
IsChildControlStateCleared	Gets a value indicating whether controls contained within this control have control state. (Inherited from Control)
IsEnabled	Gets a value indicating whether the control is enabled.
IsTrackingViewState	Gets a value that indicates whether the server control is saving changes to its view state.



ValidateRequestMode	Gets or sets a value that indicates whether the control checks client input from the browser for potentially dangerous values. (Inherited from Control)
ViewState	Gets a dictionary of state information that allows you to save and restore the view state of a server control across multiple requests for the same page. (Inherited from Control)
ViewStateIgnoresCase	Gets a value that indicates whether the StateBag object is case-insensitive. (Inherited from Control)
ViewStateMode	Gets or sets the view-state mode of this control. (Inherited from Control)
Visible	Gets or sets a value that indicates whether a server control is rendered as UI on the page. (Inherited from Control)
Width	Gets or sets the width of the Web server control.

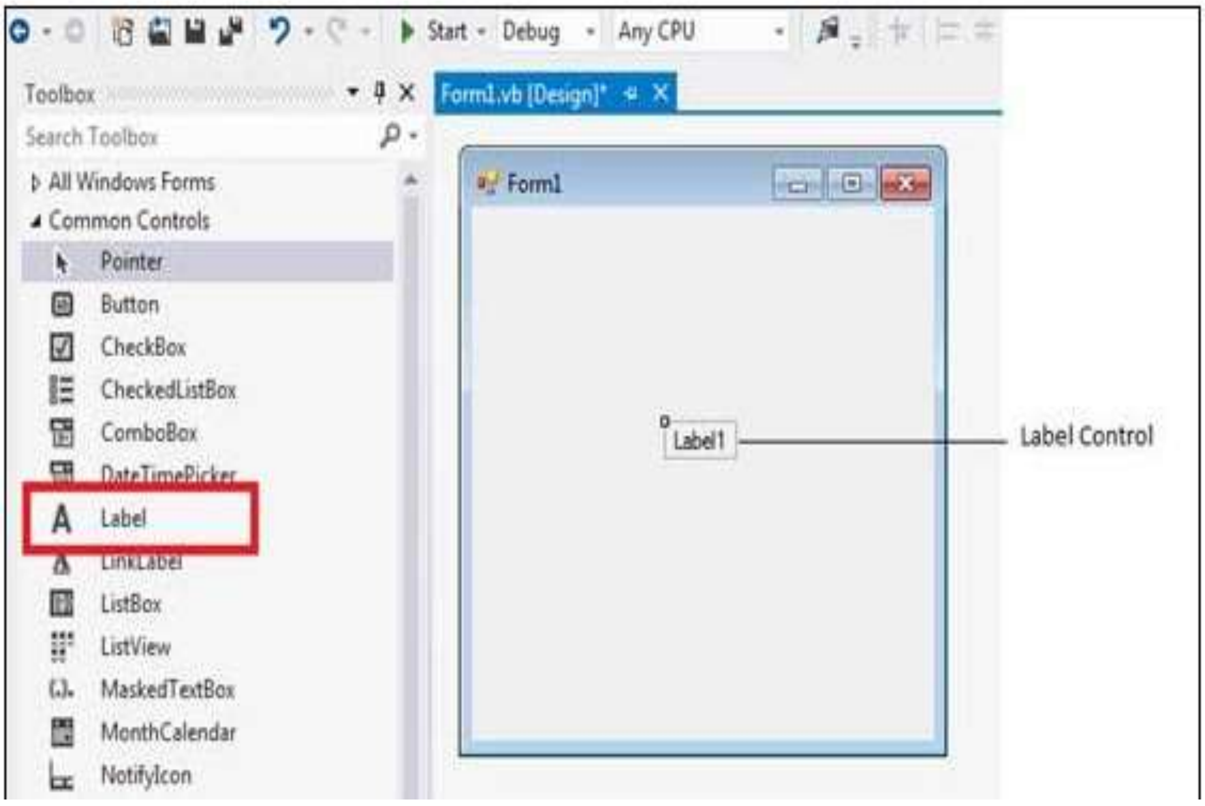
Web Controls for displaying and formatting data

- Display controls simply render text or images to the browser.
- The following are control used to display the data:

1. Label → Displays the text specified in the control's Text property.
2. Panel → Groups a set of controls (like a Frame control in Windows).
3. Table → Displays a table of information. This control has two collections: TableRows, which contains the rows, and TableCells, which contains the columns in a row.
4. TableCell → Represents a cell in a row of a Table control.
5. TableRow →Represents a row inside a Table control.

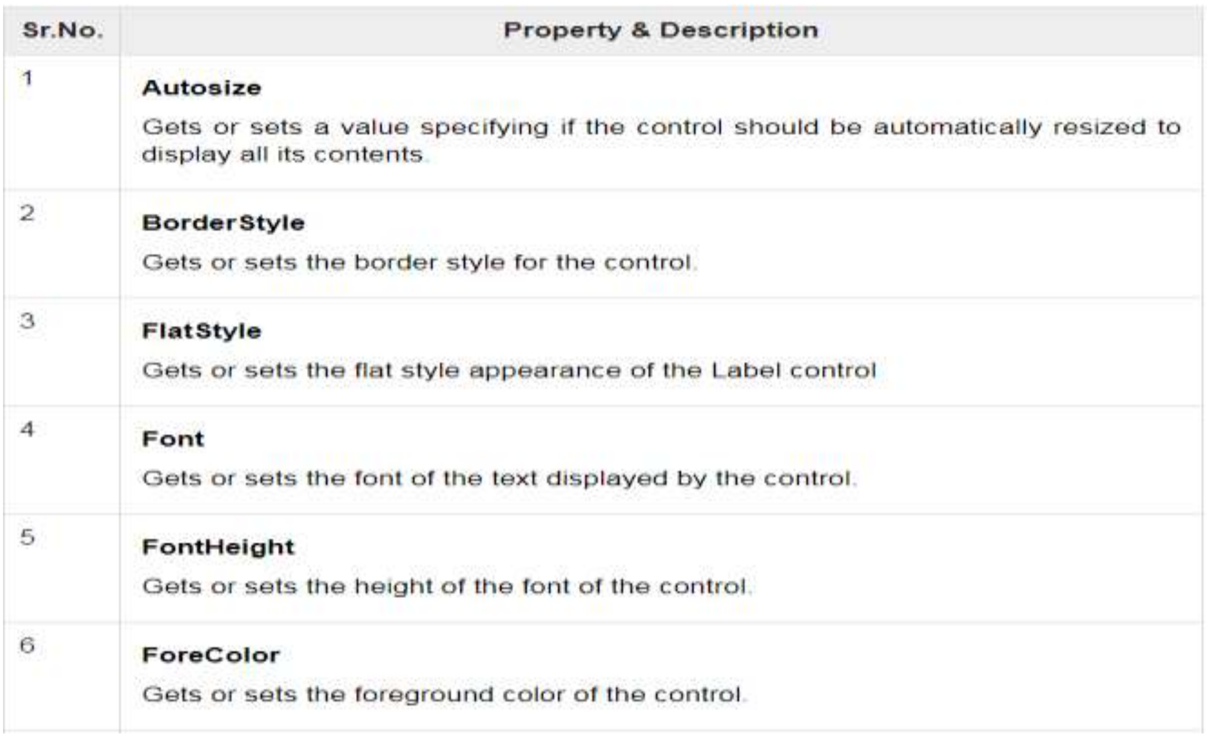
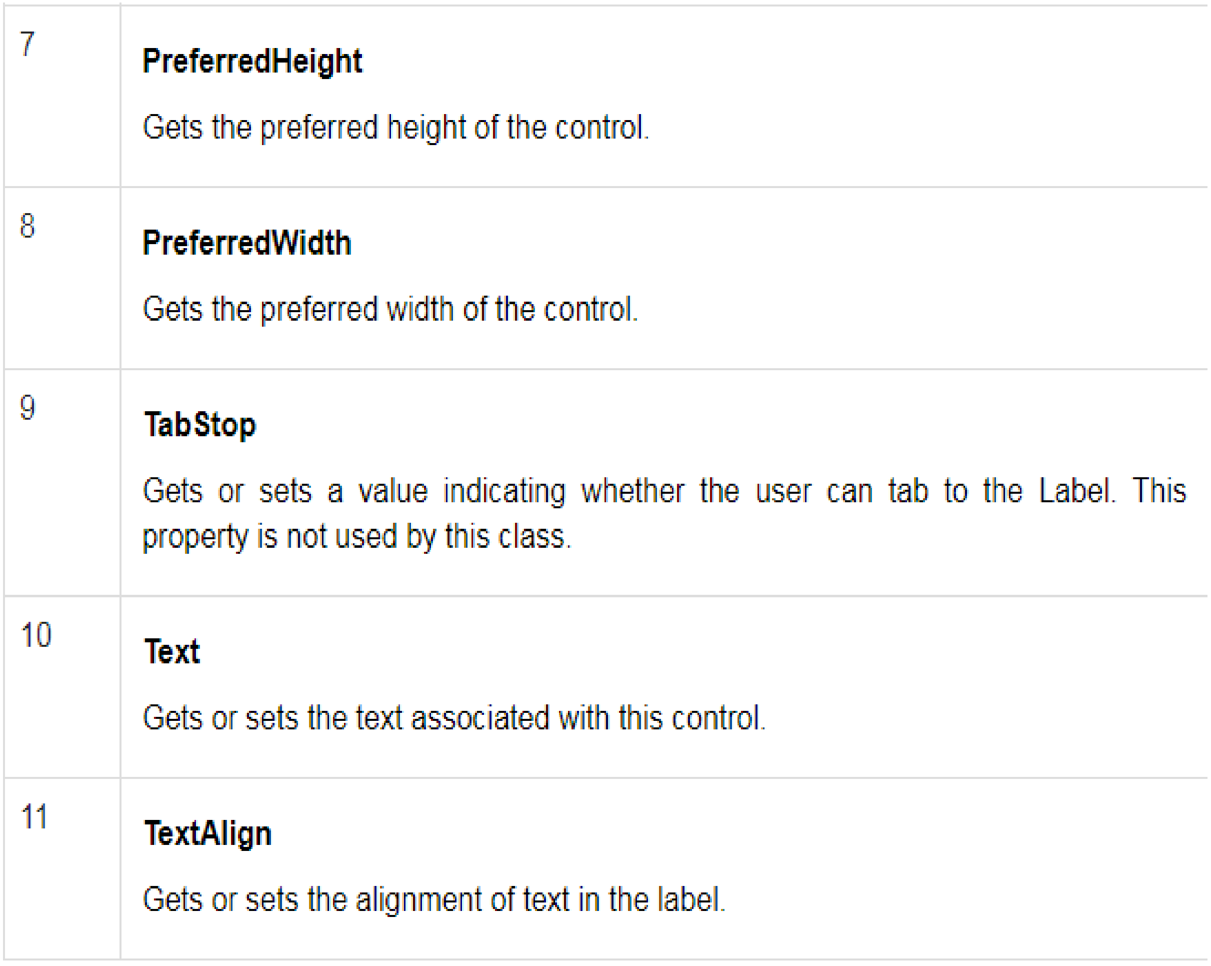
1. Label control

- The Label control represents a standard Windows label.
- It is generally used to display some informative text on the GUI which is not changed during runtime.
- Let's create a label by dragging a Label control from the Toolbox and dropping it on the form.



- **Properties of the Label Control:**

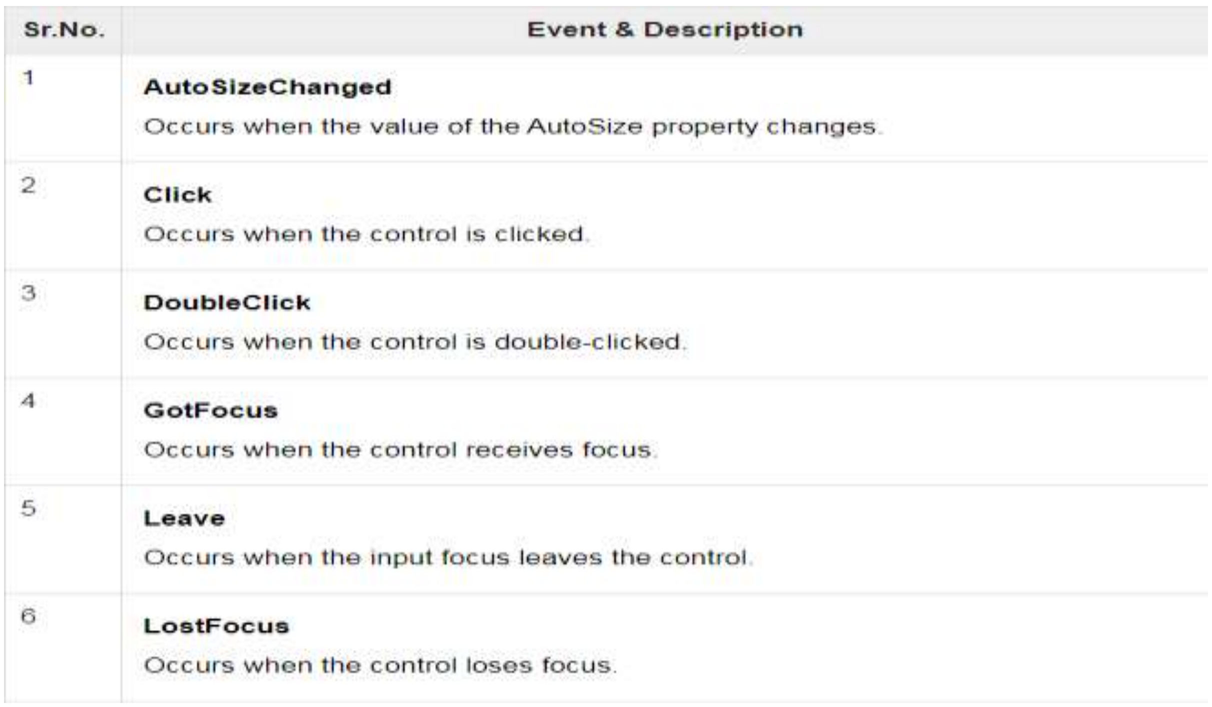
  The following are some of the commonly used properties of the Label control –

Sr.No.	Property & Description
1	**Autosize** Gets or sets a value specifying if the control should be automatically resized to display all its contents.
2	**BorderStyle** Gets or sets the border style for the control.
3	**FlatStyle** Gets or sets the flat style appearance of the Label control
4	**Font** Gets or sets the font of the text displayed by the control.
5	**FontHeight** Gets or sets the height of the font of the control.
6	**ForeColor** Gets or sets the foreground color of the control.
7	**PreferredHeight** Gets the preferred height of the control.
8	**PreferredWidth** Gets the preferred width of the control.
9	**TabStop** Gets or sets a value indicating whether the user can tab to the Label. This property is not used by this class.
10	**Text** Gets or sets the text associated with this control.
11	**TextAlign** Gets or sets the alignment of text in the label.

➢ **Events of the Label Control:**

The following are some of the commonly used events of the Label control



Sr.No.	Event & Description
1	**AutoSizeChanged** Occurs when the value of the AutoSize property changes.
2	**Click** Occurs when the control is clicked.
3	**DoubleClick** Occurs when the control is double-clicked.
4	**GotFocus** Occurs when the control receives focus.
5	**Leave** Occurs when the input focus leaves the control.
6	**LostFocus** Occurs when the control loses focus.

Example:

<asp:label id="Label1“ borderstyle="solid" bordercolor="Green" runat="Server" />

Private Sub Label1_Click(sender As Object, e As EventArgs) _ Handles Label1.Click

Label1.Text = "You have just moved the label" End Sub

2.The Panel Control

- The Panel control works as a container for other controls on the page. It controls the appearance and visibility of the controls it contains. It also allows generating controls programmatically.
- The basic syntax of panel control is as follows:
- <asp:Panel ID= "Panel1" runat = "server"> </asp:Panel>
- The Panel control is derived from the WebControl class.
- Hence it inherits all the properties, methods and events of the same.
- It does not have any method or event of its own.
- However it has the following properties of its own:

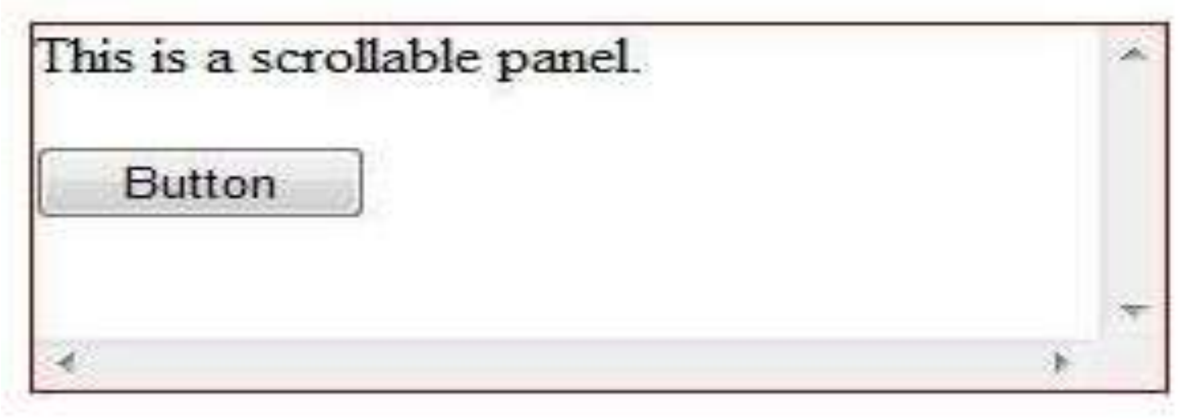
Properties	Description
BackImageUrl	URL of the background image of the panel.
DefaultButton	Gets or sets the identifier for the default button that is contained in the Panel control.
Direction	Text direction in the panel.
GroupingText	Allows grouping of text as a field.
HorizontalAlign	Horizontal alignment of the content in the panel.
ScrollBars	Specifies visibility and location of scrollbars within the panel.
Wrap	Allows text wrapping.

Working with the Panel Control:

- Let us start with a simple scrollable panel of specific height and width and a border style. The ScrollBars property is set to both the scrollbars, hence both the scrollbars are rendered.
- The source file has the following code for the panel tag:

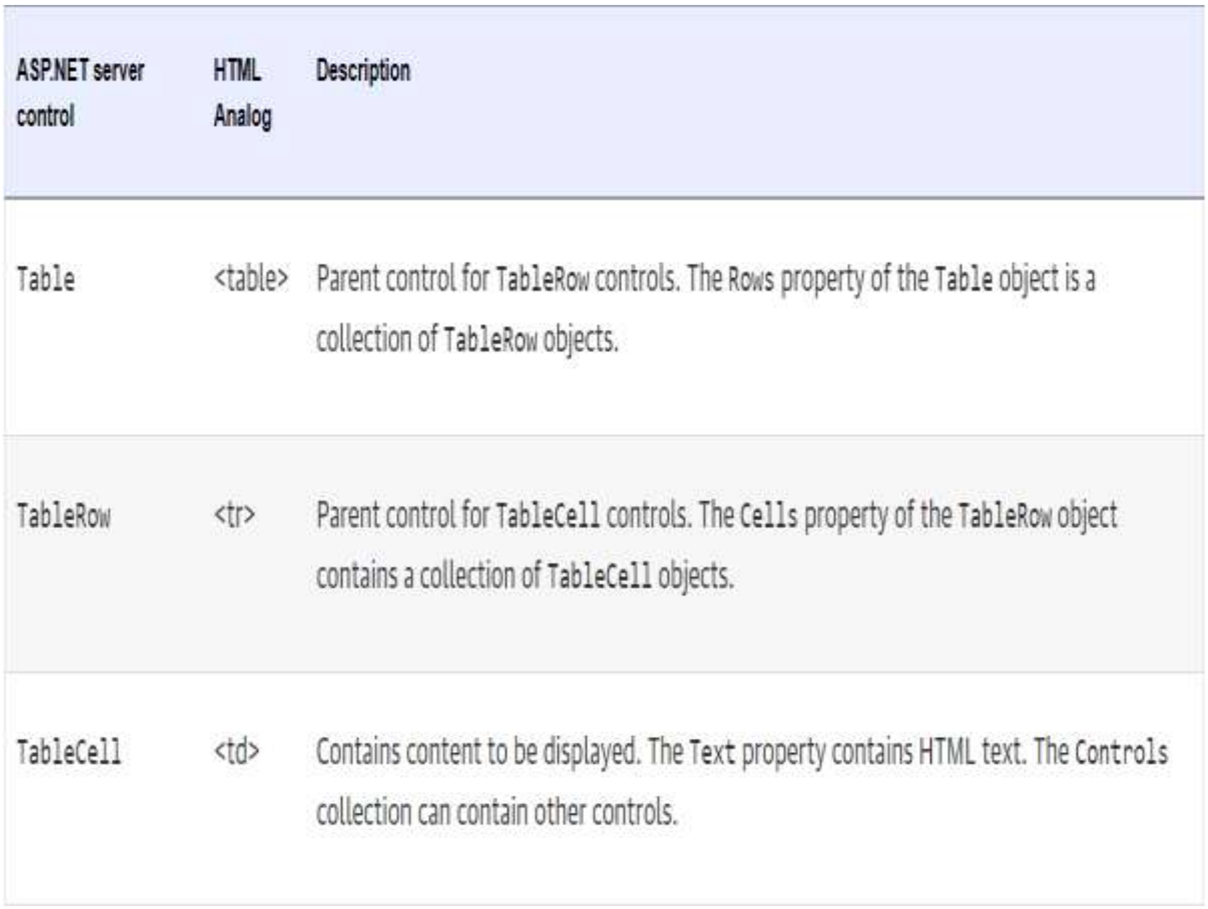
```
  <asp:Panel ID="Panel1" runat="server" BorderColor="#990000"
BorderStyle="Solid" Borderstyle="width:1px" Height="116px"
ScrollBars="Both" style="width:278px">
 This is a scrollable panel. <br /> <br />
 <asp:Button ID="btnpanel" runat="server" Text="Button"
style="width:82px" /> </asp:Panel>
```

- The panel is rendered as follows:

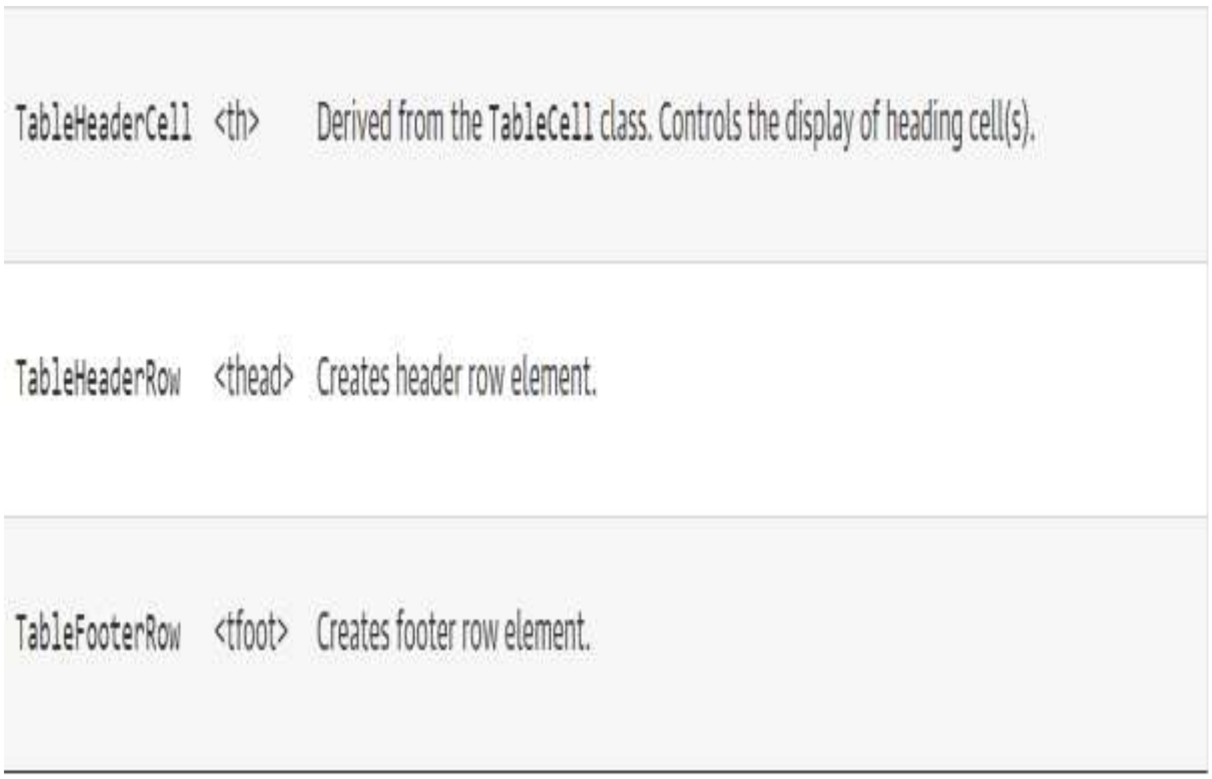


3.The table, Tablerow and table cell controls

- The Table Web server control enables you to create tables on ASP.NET pages that you can program in server code.
- The TableRow and TableCell Web server controls provide a way to display the content for the Table control.
- The Table control acts as a parent control for TableRow controls.
- The table supports a property named Rows that is a collection of TableRow objects.
- By adding or deleting items in this collection, you specify the rows for the table.
- The TableRow control in turn supports a collection named Cells that contains TableCell objects.
- The content to be displayed in the table is added to the TableCell control. The cell has a Text property that you set to any HTML text. Alternatively, you can display controls in the cell by adding controls to the cell's Controls collection.



ASP.NET server control	HTML Analog	Description
Table	`<table>`	Parent control for TableRow controls. The Rows property of the Table object is a collection of TableRow objects.
TableRow	`<tr>`	Parent control for TableCell controls. The Cells property of the TableRow object contains a collection of TableCell objects.
TableCell	`<td>`	Contains content to be displayed. The Text property contains HTML text. The Controls collection can contain other controls.

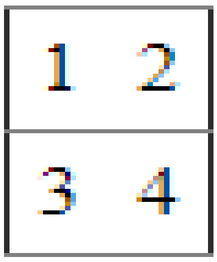


TableHeaderCell	<th>	Derived from the TableCell class. Controls the display of heading cell(s).
TableHeaderRow	<thead>	Creates header row element.
TableFooterRow	<tfoot>	Creates footer row element.

- **Example**

```
<asp:Table runat="server" CellPadding="5"
GridLines="horizontal" HorizontalAlign="Center">
  <asp:TableRow>
    <asp:TableCell>1</asp:TableCell>
    <asp:TableCell>2</asp:TableCell>
  </asp:TableRow>
  <asp:TableRow>
    <asp:TableCell>3</asp:TableCell>
    <asp:TableCell>4</asp:TableCell>
  </asp:TableRow>
</asp:Table>
```

Output:

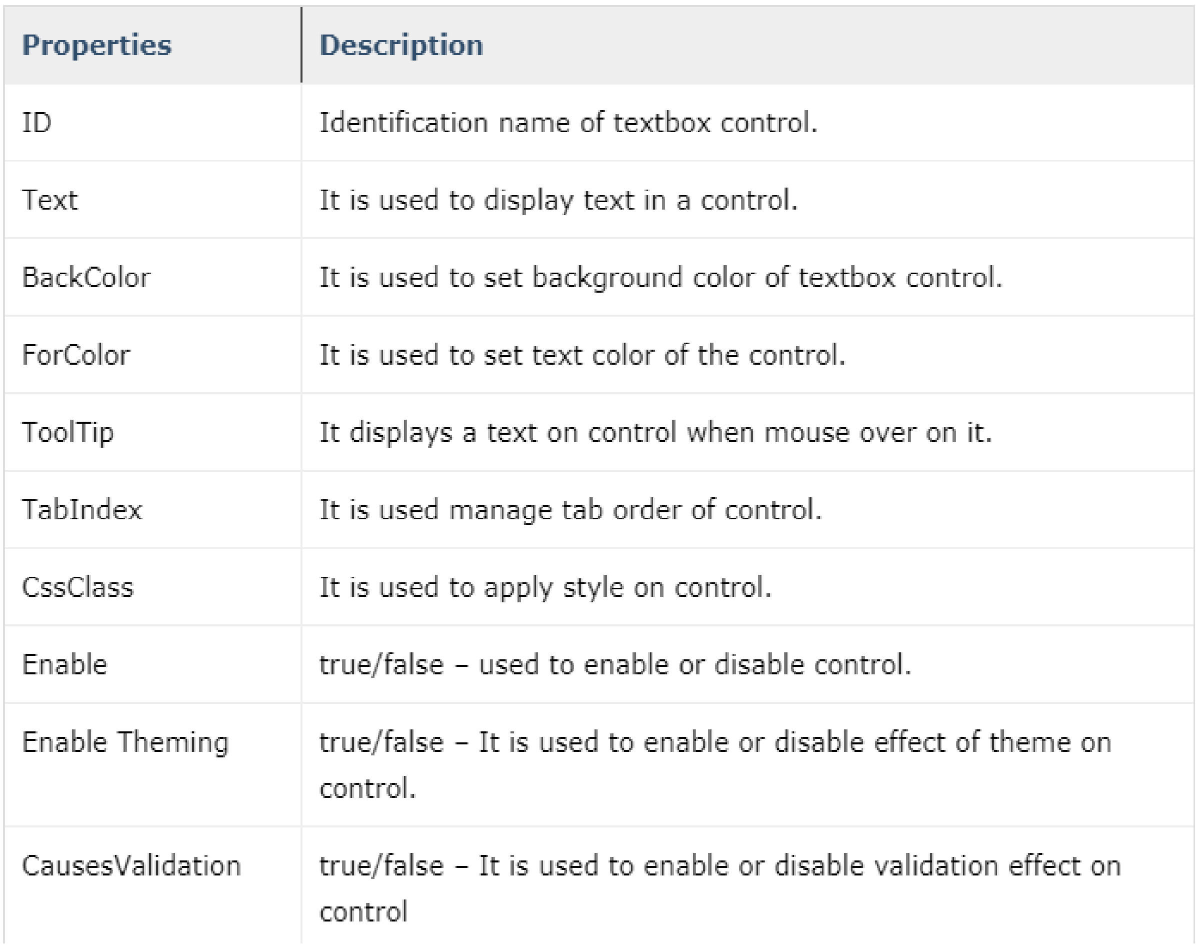


1	2
3	4

2.10 web Controls for creating buttons

- Whenever you need the server to respond to a user click,use one of the button web controls.
- Which button you use depends on the design requirements of your site.
- Buttons can be graphics image,a hyperlink or a regular push button.
- The following are list of buttons used in ASP.NET:

1. The Button Control
2. The Image Button Control
3. The Link Button Control



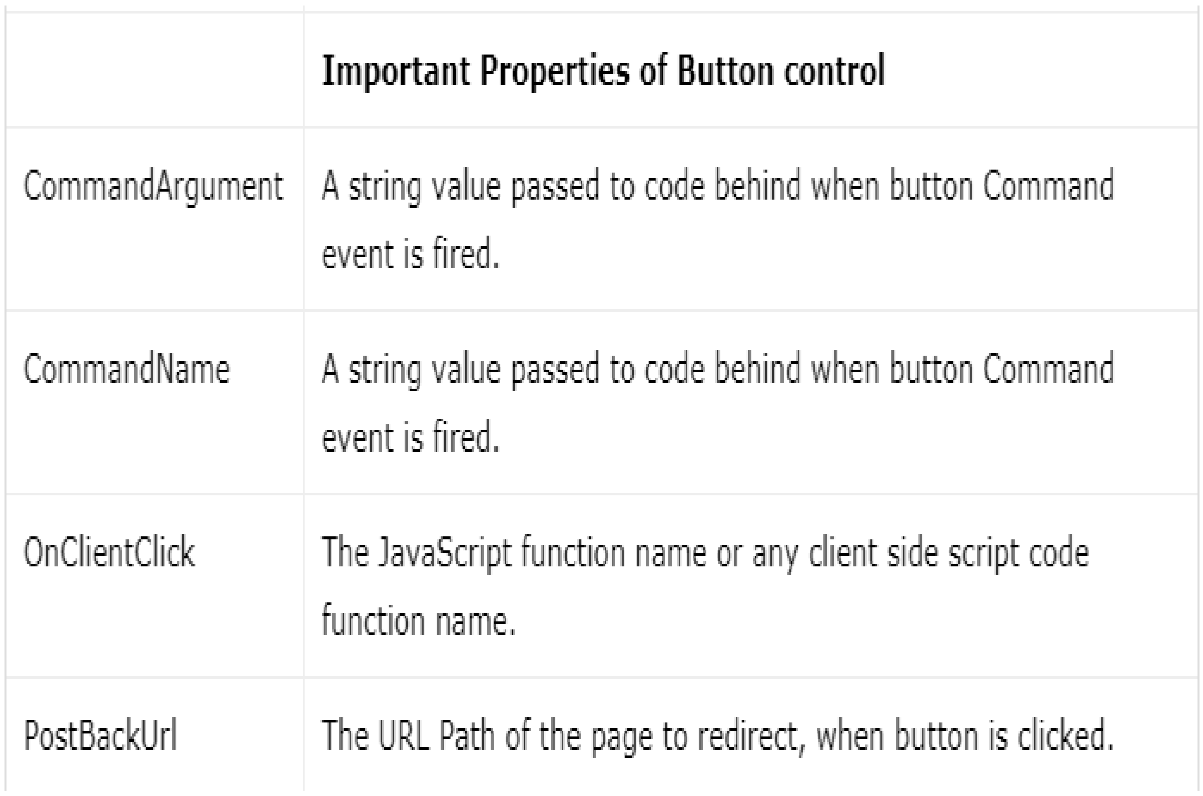
Properties	Description
ID	Identification name of textbox control.
Text	It is used to display text in a control.
BackColor	It is used to set background color of textbox control.
ForColor	It is used to set text color of the control.
ToolTip	It displays a text on control when mouse over on it.
TabIndex	It is used manage tab order of control.
CssClass	It is used to apply style on control.
Enable	true/false – used to enable or disable control.
Enable Theming	true/false – It is used to enable or disable effect of theme on control.
CausesValidation	true/false – It is used to enable or disable validation effect on control

1.The Button Control

- The Button control is used to display a push button. The push button may be a submit button or a command button. By default, this control is a submit button.
- A submit button does not have a command name and it posts the page back to the server when it is clicked. It is possible to write an event handler to control the actions performed when the submit button is clicked.

- A command button has a command name and allows you to create multiple Button controls on a page. It is possible to write an event handler to control the actions performed when the command button is clicked.
- The example below demonstrates a simple Button control:
  <asp:Button id=“Btnsubmit" Text="Submit" runat="server“ />
- The Button control have the following Event handling:

1. Click
2. Command
3. Data Binding
4. Load
5. Unload



	Important Properties of Button control
CommandArgument	A string value passed to code behind when button Command event is fired.
CommandName	A string value passed to code behind when button Command event is fired.
OnClientClick	The JavaScript function name or any client side script code function name.
PostBackUrl	The URL Path of the page to redirect, when button is clicked.

2.The Image Button Control

- ImageButton control in ASP.Net is used in button formation by which we can use the images.
- It is like a button with an image on it. Generally, we have seen the images on the website and after clicking on it, certain activities performed.
- So, in this case, we need to use Image Button control.

- It is used to fire an event after clicking on the Image Button either on the client or server-side.
- We can set the image, which we want on the button.
- This image button will respond to the mouse click as soon as we click.
- When we click the image button control, it raises both the events that click and command events.
- The example source code to design the image button is:

```
        <asp:ImageButton ID="ImageButton1" runat="server"
ImageUrl="~\Educba.png" Height="64px" Width="158px" Imagealign="Left" />
```

3.The link button

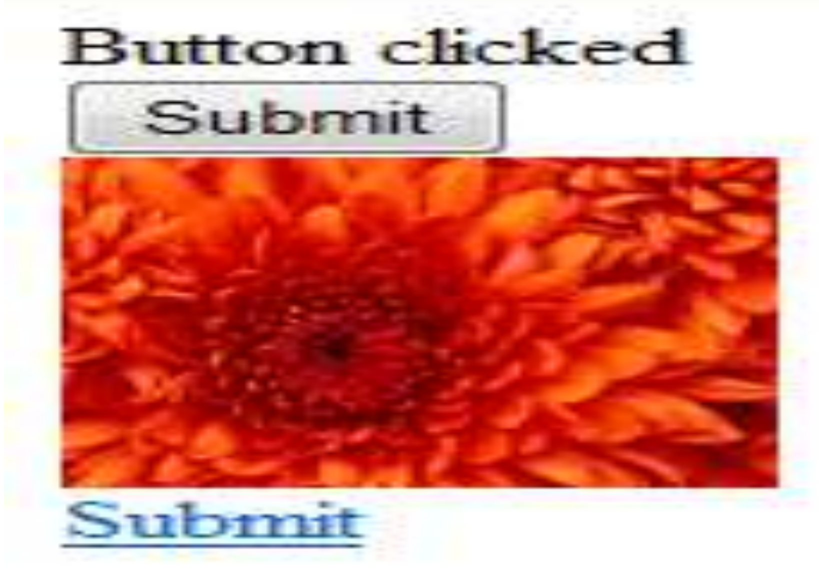
- LinkButton server control displays a hyperlink styled button on the web page.
- the LinkButton control which is run at the server and the generated HTML code is returned as a response to the browser.
- The LinkButton control generates the HTML anchor (hyperlink) element. It lets the users navigate to a section within the page or to another page.
- The LinkButton control looks like a hyperlink on the web page and functions as a button.
- This makes it a very useful control.
- The example source code to design the Link button is:

```
 <asp:LinkButton ID="linkButtonId" runat="server">
      DisplayText
</asp:LinkButton>
```

- Example program using button controls

```
<body>
   <form id="form1" runat="server">
   <div>
     <asp:Label ID="lbldisplay" runat="server" ></asp:Label><br />
   <asp:Button ID="button" runat="server" Text="Submit" /><br />
      <asp:ImageButton ID="ImageButton1" runat="server" Height="100px"
ImageUrl="~/button.jpg"
```

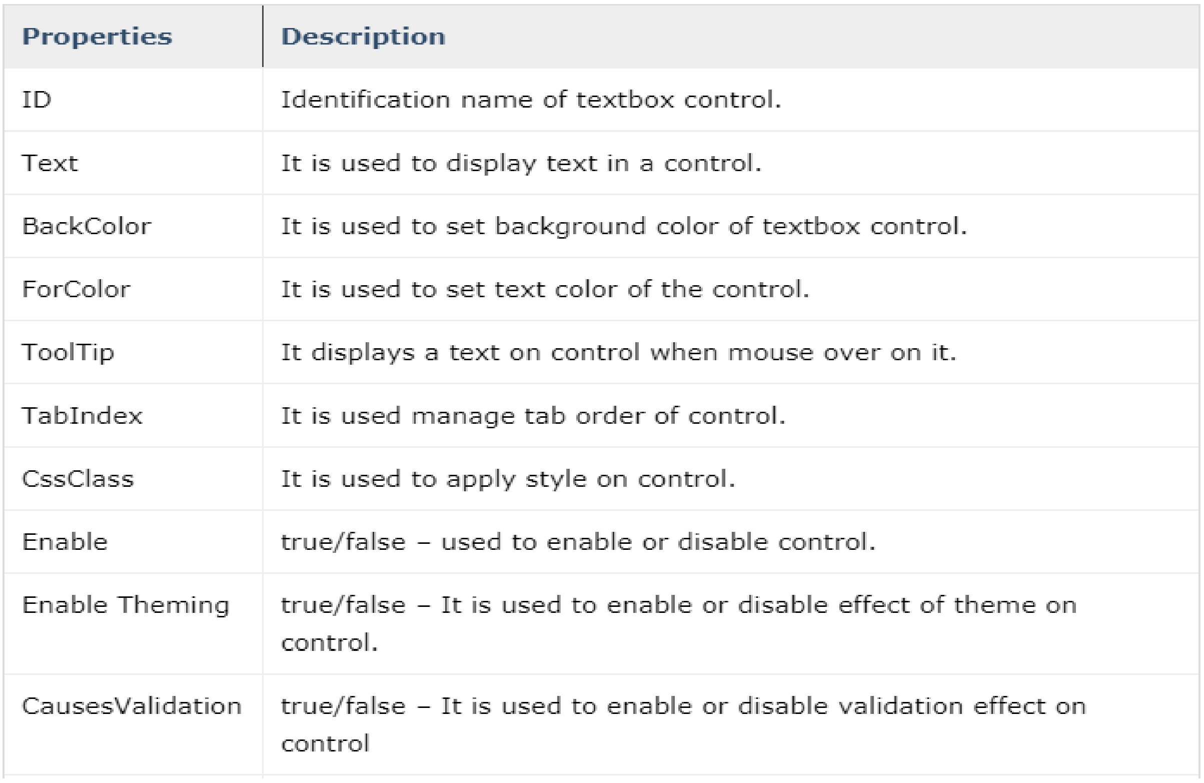
```
        Width="100px" /><br />
    <asp:LinkButton ID="lbbutton" runat="server">Submit</asp:LinkButton>
    </div>
    </form>
</body>
Protected Sub button_Click
        lbldisplay.Text = "Button clicked"
    End Sub
      Protected Sub lbbutton_Click
        lbldisplay.Text = "Link Button clicked"
    End Sub
    Protected Sub ImageButton1_Click
        lbldisplay.Text = "Image Button clicked"
    End Sub
```



2.11 Web control for Inputting text

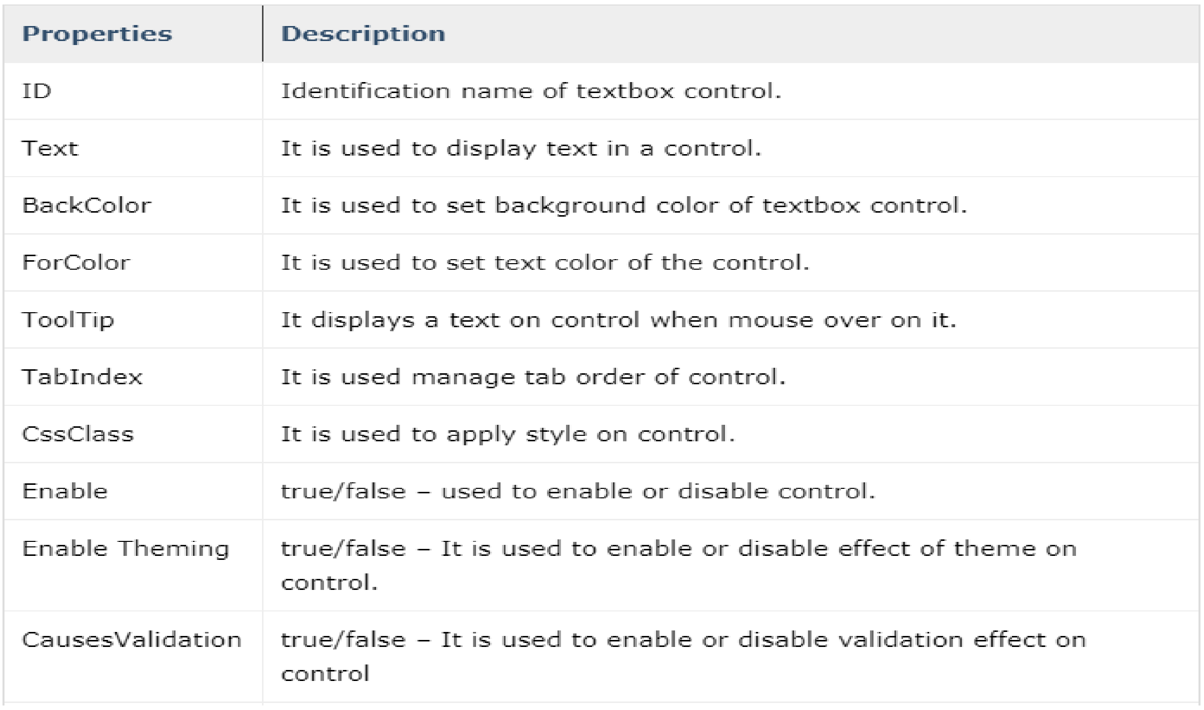
- Textbox control is most usable web server control in asp.net.
- TextBox control is a rectangular box which is used to take user to input.
- In simple word the TextBox is a place where user can input some text on asp.net web form.

- To use TextBox on page we can write code or just drag and drop from toolbox.
- Example: <asp:TextBox ID="txtusername" runat="server"></asp:TextBox>
- All server side control has its own properties and events. below list shows textbox properties.

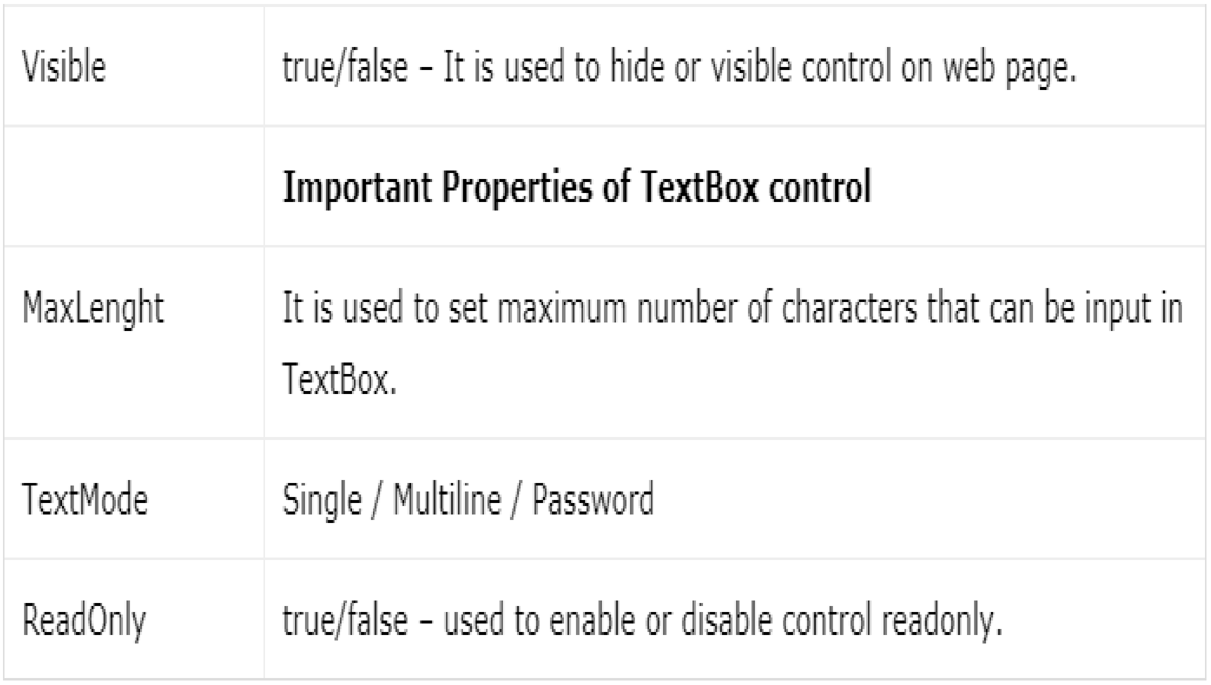


Properties	Description
ID	Identification name of textbox control.
Text	It is used to display text in a control.
BackColor	It is used to set background color of textbox control.
ForColor	It is used to set text color of the control.
ToolTip	It displays a text on control when mouse over on it.
TabIndex	It is used manage tab order of control.
CssClass	It is used to apply style on control.
Enable	true/false – used to enable or disable control.
Enable Theming	true/false – It is used to enable or disable effect of theme on control.
CausesValidation	true/false – It is used to enable or disable validation effect on control

- All server side control has its own properties and events. below list shows textbox properties.



Properties	Description
ID	Identification name of textbox control.
Text	It is used to display text in a control.
BackColor	It is used to set background color of textbox control.
ForColor	It is used to set text color of the control.
ToolTip	It displays a text on control when mouse over on it.
TabIndex	It is used manage tab order of control.
CssClass	It is used to apply style on control.
Enable	true/false – used to enable or disable control.
Enable Theming	true/false – It is used to enable or disable effect of theme on control.
CausesValidation	true/false – It is used to enable or disable validation effect on control



Visible	true/false – It is used to hide or visible control on web page.
	Important Properties of TextBox control
MaxLenght	It is used to set maximum number of characters that can be input in TextBox.
TextMode	Single / Multiline / Password
ReadOnly	true/false – used to enable or disable control readonly.

The example source code to design the Text box is:

```
 <table>
      <tr>
   <td    <asp:Label ID="lblusername" runat="server"
Text="Username"></asp:Label>
   </td>
   <td>
   <asp:TextBox ID="txtusername" runat="server"></asp:TextBox>
   </td>
   </tr>
   <tr>
   <td>
   <asp:Label ID="lblpassword" runat="server"
Text="Password"></asp:Label>
   </td>
   <td>
   <asp:TextBox ID="txtpassword" runat="server" ></asp:TextBox>
   </td>
</tr>
</table>
```

2.15 ASP.NET Page Directives

- Page directives control setting for the code compiler like page settings and namespaces to import into page.
- Each of the page directives can have several different attributes associated with it.
- The attributes usually set what language to use for the page and what class to use for the code behind file.
- For example

  <%@ Page Language="VB“ CodeFile="Default.aspx.vb" Inherits="_Default" %>

1.The page and control directives

- The most commonly used directive is the @ Page directive and it can be used only in Web Forms.
- Page directive allows you to specify many configuration options for the page. By default, Visual Studio creates a page directive as shown below:
- <%@ Page Language="VB“ CodeFile="Default.aspx.vb" Inherits="_Default" %>
- You can include only one @ Page directive in your .aspx file.
- Also you should specify one language in the Language attribute.
- This can be any .NET Framework-supported language, including VB.Net, C#, or JScript.
- AutoEventWireup controlled the automatic binding of page events based on the method naming convention.
- CodeFile specifies a path to the referenced code-behind file for the page.
- Inherits defines the name of the class from which to inherit. This can be any class derived from the Page class

2. Different types of directives in Asp.Net

- **@ Assembly**

  The @Assembly Directive attaches assemblies to the page or an ASP.NET user control thereby all the assembly classes and interfaces are available to the class.

- **@ Control**

  Defines control-specific attributes used by the ASP.NET page parser and compiler and can be included only in .ascx files (user controls).

- **@ Import**

  The Import directive imports a namespace into a web page, user control page of application. If the Import directive is specified in the global.asax file, then it is applied to the entire application.

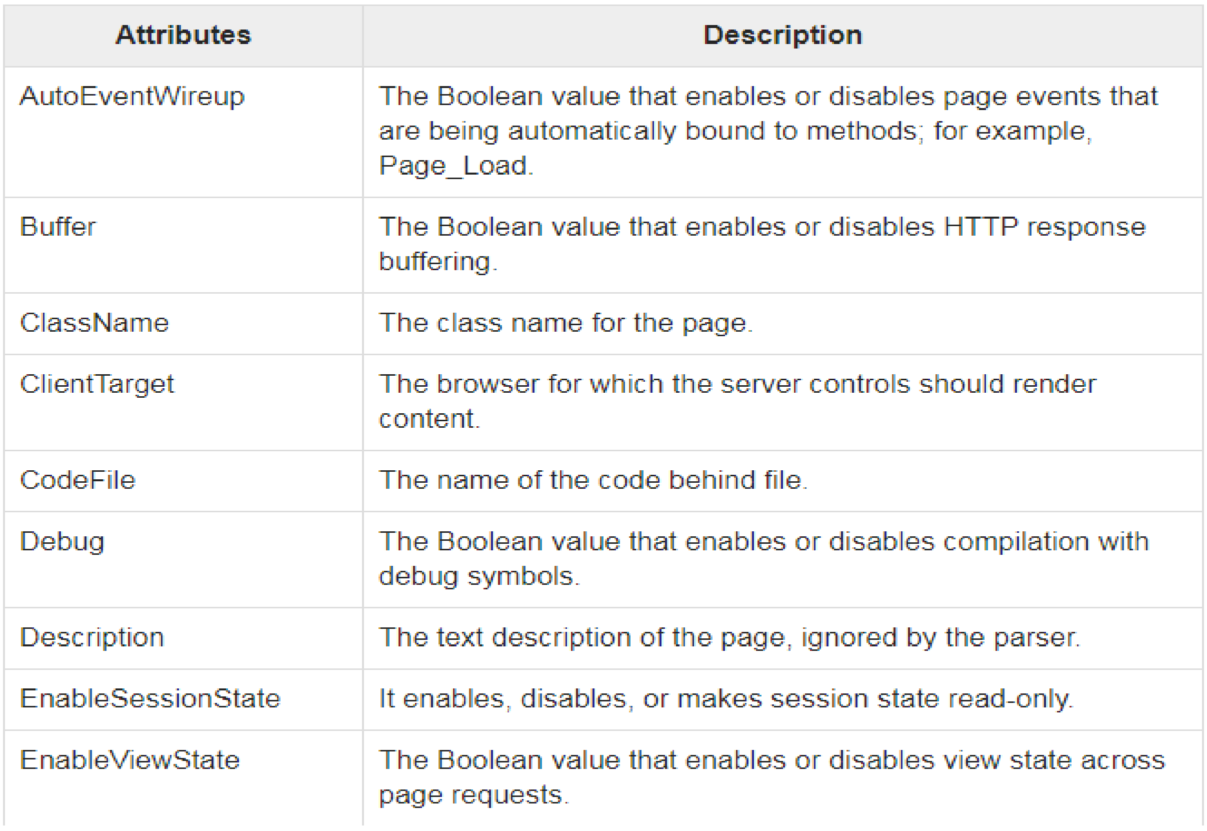
- **@ OutputCache**

  The OutputCache directive controls the output caching policies of a web page or a user control.

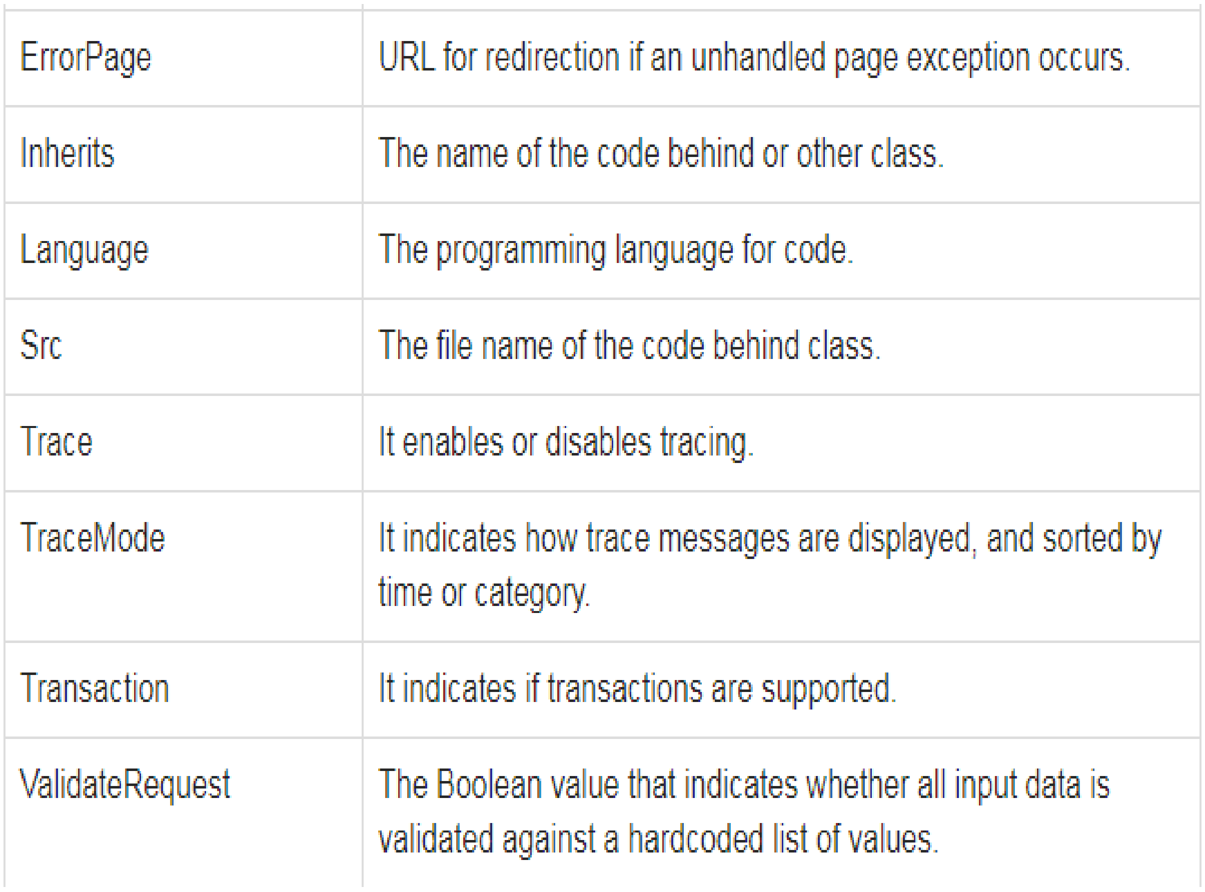
- **@ Page**

  The @Page directive enables you to specify attributes and values for an Asp.Net Page to be used when the page is parsed and compiled

Attributes used for page and control directives



Attributes	Description
AutoEventWireup	The Boolean value that enables or disables page events that are being automatically bound to methods; for example, Page_Load.
Buffer	The Boolean value that enables or disables HTTP response buffering.
ClassName	The class name for the page.
ClientTarget	The browser for which the server controls should render content.
CodeFile	The name of the code behind file.
Debug	The Boolean value that enables or disables compilation with debug symbols.
Description	The text description of the page, ignored by the parser.
EnableSessionState	It enables, disables, or makes session state read-only.
EnableViewState	The Boolean value that enables or disables view state across page requests.



ErrorPage	URL for redirection if an unhandled page exception occurs.
Inherits	The name of the code behind or other class.
Language	The programming language for code.
Src	The file name of the code behind class.
Trace	It enables or disables tracing.
TraceMode	It indicates how trace messages are displayed, and sorted by time or category.
Transaction	It indicates if transactions are supported.
ValidateRequest	The Boolean value that indicates whether all input data is validated against a hardcoded list of values.

3.The @import Directives

- The Import directive imports a namespace into a web page, user control page of application.
- If the Import directive is specified in the global.asax file, then it is applied to the entire application.
- If it is in a page of user control page, then it is applied to that page or control.
- The basic syntax for import directive is:

<%@ Import namespace="System.Drawing" %>

<%@import namespace="YourNamespace" %>

<%@import namespace="System.Net.Sockets" %>

4.**The Register Directive**

- The Register derivative is used for registering the custom server controls and user controls.
- The basic syntax of Register directive is:

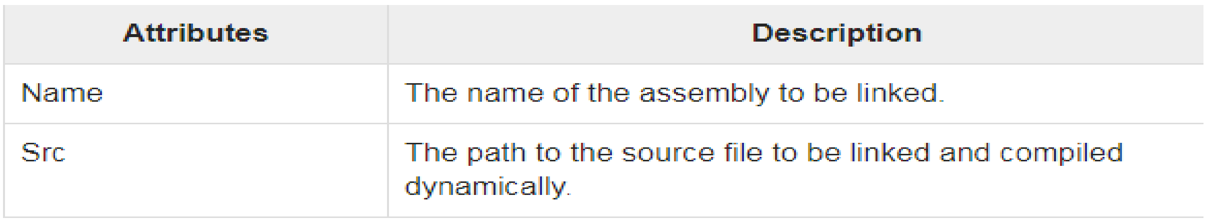
  <%@ Register Src="~/footer.ascx" TagName="footer" TagPrefix="Tfooter" %>

- The TagPrefix attributes sets an alies for a namespace.

- The Tag name attributes specifies an alies to use for a class
- Namespace is the namespace to associate with the specified tag prefix.

5. **The Assembly Directive**

- The Assembly directive links an assembly to the page or the application at parse time.
- This could appear either in the global.asax file for application-wide linking, in the page file, a user control file for linking to a page or user control.
- The basic syntax of Assembly directive is:
- <%@ Assembly Name ="myassembly" %>
- The attributes of the Assembly directive are:



Attributes	Description
Name	The name of the assembly to be linked.
Src	The path to the source file to be linked and compiled dynamically.

6. **The OutputCache Directive**

- The OutputCache directive controls the output caching policies of a web page or a user control.
- The basic syntax of OutputCache directive is:

  <%@ OutputCache Duration="15" VaryByParam="None" %>
- Duration is measured in seconds.
- This is the length of time during which the output cache for the page is kept active

  Vary is a comma separated list of HTTP headers used to vary output cache.

2.15 ASP.NET Rich Controls

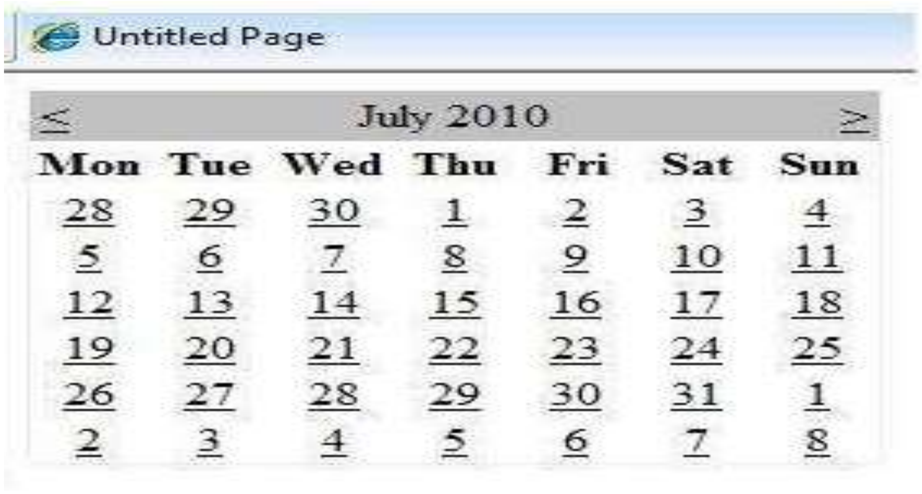
- These web controls, which are sometimes referred to as rich controls, use dynamic HTML and client side script when appropriate to provide better user interaction.
- ASP.NET provides large set of controls. These controls are divided into different categories, depends upon their functionalities.
- The followings control comes under the rich controls category.

  1. Calendar control

2. AdRotator Control

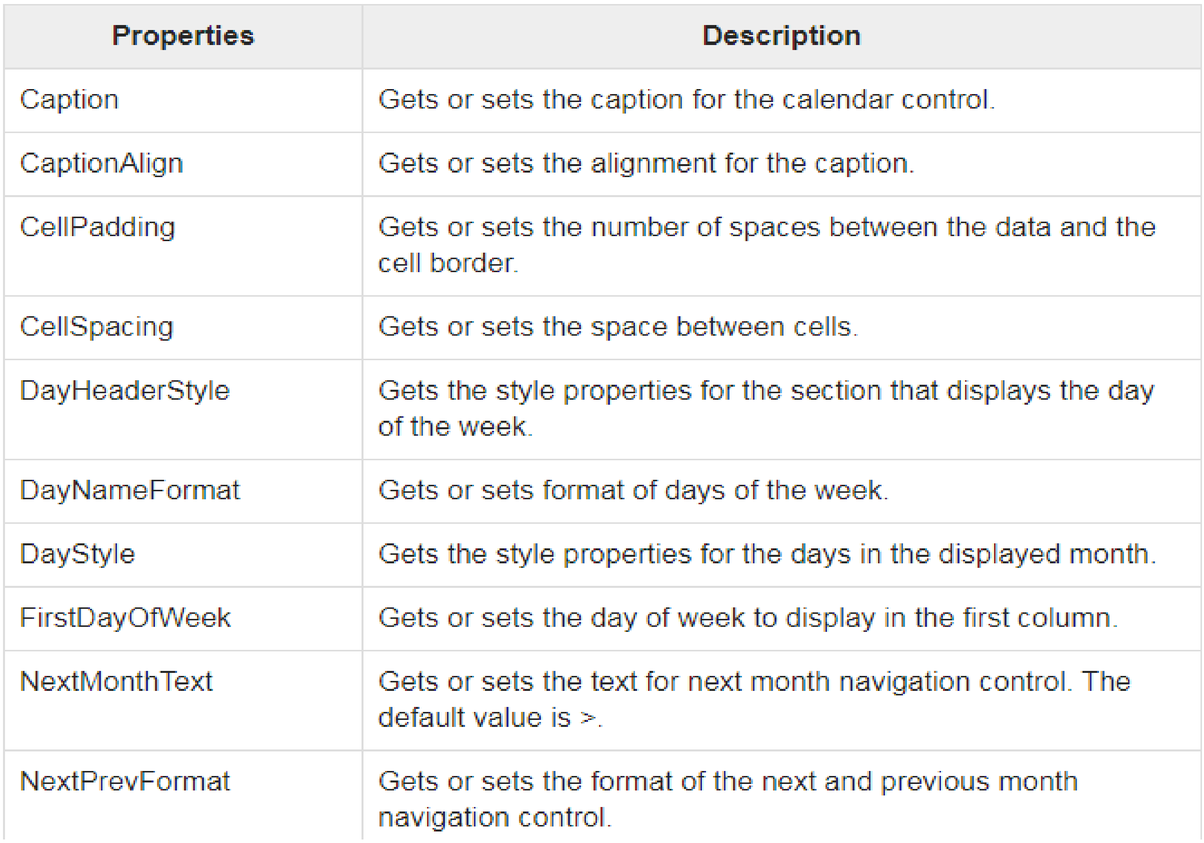
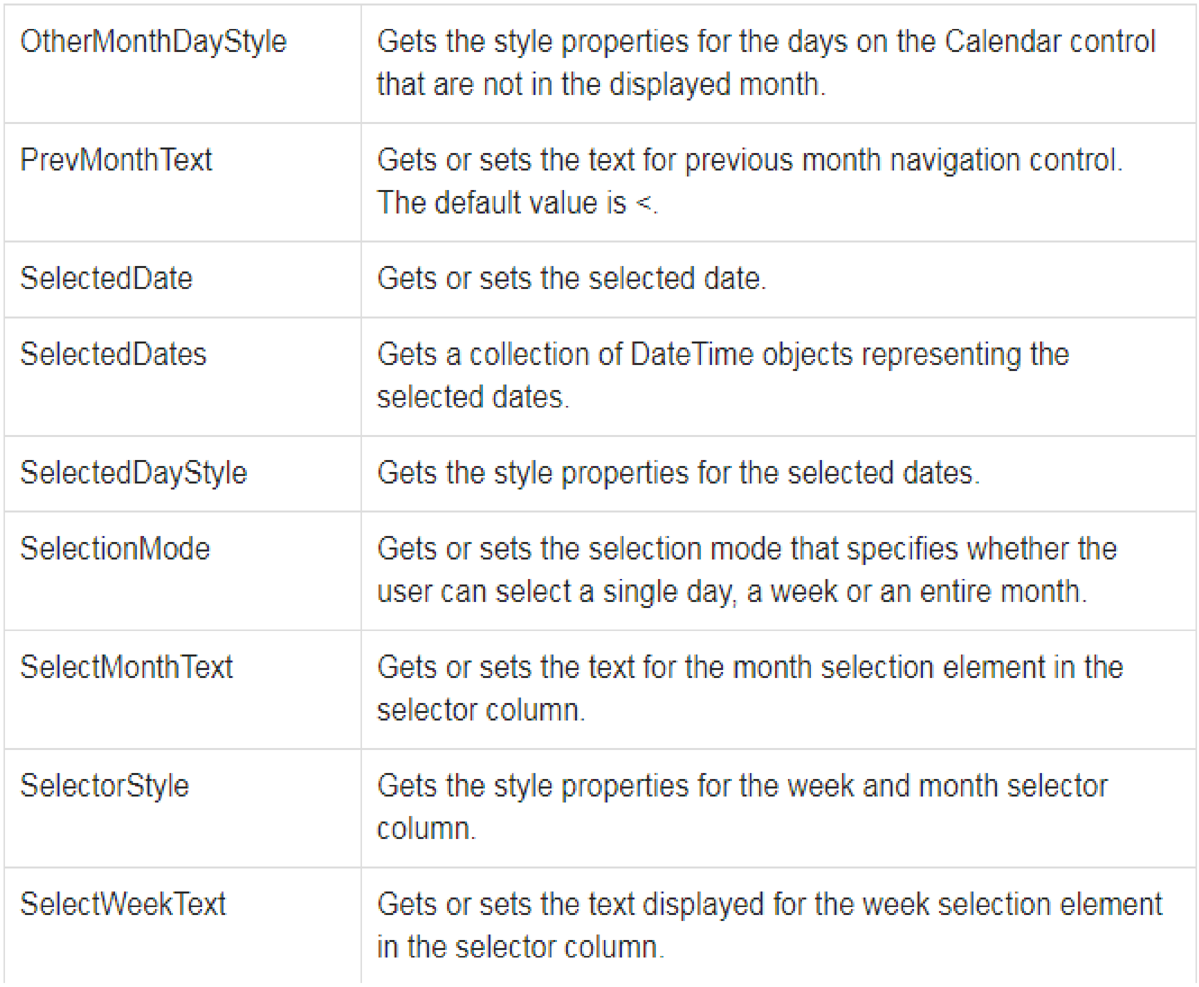
1. **Calendar control**

- ➢ By far the most complex server control in ASP.NET is the Calendar control. This creates a fully interactive *one-month-at-a-time* calendar as an HTML table.
- ➢ the default output from the Calendar control, and you can see that it really is a *rich* control in that it saves an incredible amount of effort on behalf of the developer.
- ➢ The code used for inserting the control into the source of the page is:
- ➢ <ASP:Calendar id="MyControl" runat="server" />



<			July 2010			>
Mon	Tue	Wed	Thu	Fri	Sat	Sun
28	29	30	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	1
2	3	4	5	6	7	8

- ➢ Notice that it automatically defaults to the current date ,if you don't specify a date.
- ➢ If you scroll to the bottom of this page, you can see some of the properties that can be set to change the appearance.
  You can write an event handler for the SelectionChanged event, and obtain the date that the user selected by querying the SelectedDate property of the control within that event handler.

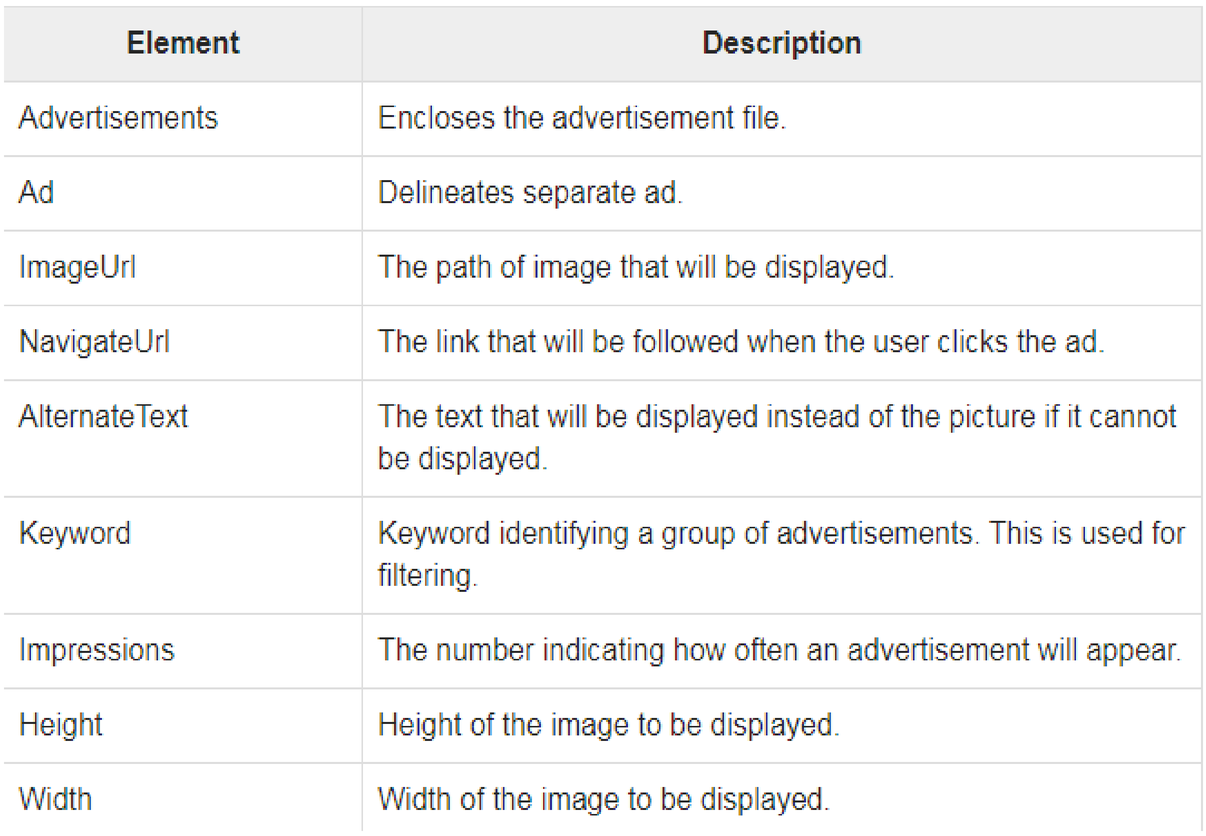
Properties	Description
Caption	Gets or sets the caption for the calendar control.
CaptionAlign	Gets or sets the alignment for the caption.
CellPadding	Gets or sets the number of spaces between the data and the cell border.
CellSpacing	Gets or sets the space between cells.
DayHeaderStyle	Gets the style properties for the section that displays the day of the week.
DayNameFormat	Gets or sets format of days of the week.
DayStyle	Gets the style properties for the days in the displayed month.
FirstDayOfWeek	Gets or sets the day of week to display in the first column.
NextMonthText	Gets or sets the text for next month navigation control. The default value is >.
NextPrevFormat	Gets or sets the format of the next and previous month navigation control.
OtherMonthDayStyle	Gets the style properties for the days on the Calendar control that are not in the displayed month.
PrevMonthText	Gets or sets the text for previous month navigation control. The default value is <.
SelectedDate	Gets or sets the selected date.
SelectedDates	Gets a collection of DateTime objects representing the selected dates.
SelectedDayStyle	Gets the style properties for the selected dates.
SelectionMode	Gets or sets the selection mode that specifies whether the user can select a single day, a week or an entire month.
SelectMonthText	Gets or sets the text for the month selection element in the selector column.
SelectorStyle	Gets the style properties for the week and month selector column.
SelectWeekText	Gets or sets the text displayed for the week selection element in the selector column.

2. AdRotator control

- The AdRotator control randomly selects banner graphics from a list, which is specified in an external XML schedule file. This external XML schedule file is called the advertisement file.
- The AdRotator control allows you to specify the advertisement file and the type of window that the link should follow in the AdvertisementFile and the Target property respectively.
- The basic syntax of adding an AdRotator is as follows:

  ```
  <asp:AdRotator runat = "server" AdvertisementFile = "adfile.xml" Target =
  "_blank" />
  ```
- The advertisement file is an XML file, which contains the information about the advertisements to be displayed.
- XML is not a language in itself, like HTML, but a set of rules for creating new markup languages. It is a meta-markup language.
- It allows developers to create custom tag sets for special uses. It structures, stores, and transports the information.
- Following is an example of XML file:

  ```
          <BOOK>
    <NAME> Learn XML </NAME>
   <AUTHOR> Samuel Peterson </AUTHOR> <PUBLISHER> NSS
  Publications </PUBLISHER> <PRICE> $30.00</PRICE>
    </BOOK>
  ```
- The following are properties of adrotator control:



Element	Description
Advertisements	Encloses the advertisement file.
Ad	Delineates separate ad.
ImageUrl	The path of image that will be displayed.
NavigateUrl	The link that will be followed when the user clicks the ad.
AlternateText	The text that will be displayed instead of the picture if it cannot be displayed.
Keyword	Keyword identifying a group of advertisements. This is used for filtering.
Impressions	The number indicating how often an advertisement will appear.
Height	Height of the image to be displayed.
Width	Width of the image to be displayed.

➢ **Example**:

```
     <Ad> <ImageUrl>rose2.jpg</ImageUrl> <NavigateUrl>
   http://www.babybouquets.com.au
</NavigateUrl>
 <AlternateText>
   Order roses and flowers
</AlternateText> <Impressions>20</Impressions>
<Keyword>gifts</Keyword>
</Ad>
```

2.16 Validation Controls

- ➢ ASP.Net provides various validation controls that validate the user data to ensure that the data entered by the user are satisfied with the condition.
- ➢ The following are the list of validations:

1. **The BaseValidator Class**
2. **RequiredFieldValidator**
3. **RangeValidator**
4. **RegularExpressionValidator**

5. **CompareValidator**
6. **CustomValidator**

1. **The BaseValidator Class**

- All the validation controls in ASP.Net inherit the properties and methods of the BaseValidator class.
- It helps in making a generic suite of validation controls. The important properties and methods of the BaseValidator class are:

❖ **ControlToValidate –** It indicates the input control to validate. It must be a unique value throughout the form.

❖ **Enabled –** It enables or disables the validator.

❖ **Text –** It holds the message to be displayed in the event of a validation failure.

❖ **ErrorMessage –** The value in this attribute is displayed either when ValidationSummary control is used or when Text property is missing.

❖ **IsValid –** A Boolean attribute which indicates whether the control is valid or not.

❖ **Validate() –** This method revalidates the control and updates the IsValid

2. **RequiredFieldValidator**

- This is an elementary validation control. Almost all the forms have some fields that are mandatory to be filled up by the user before proceeding forward. The Required Field Validator ensures that such fields are not left empty.
- Syntax:

```
        <asp:RequiredFieldValidator
   ID="someUniqueId"
runat="server“
   ControlToValidate ="someUniqueControlId"
ErrorMessage="ErrorToDisplayOnValidationFailure“>
</asp:RequiredFieldValidator>
```

3. **RangeValidator**

- The RangeValidator control simply specifies the permitted range within which the input value should fall. This is most helpful for numeral input values such as age or for Date input values.
- Syntax:

```
  <asp:RangeValidator ID="some unique id"
runat="server"
   ControlToValidate ="someUniqueControlId"
ErrorMessage="ErrorToDisplayOnValidationFailure"
Type="Integer" MinimumValue=”0” MaximumValue=”100”>
</asp:RangeValidator>
```

4. **RegularExpressionValidator**

- RegularExpressions, or simply Regex, are patterns that define the format of the text. If the text is in the same format, Regex returns true, else false.
- It is recommended to read about Regex if you are not familiar with it. This will also give you an idea about how Regex patterns are formed and how to decipher a Regex pattern.
- Thus, a RegularExpressionValidator is a very versatile validation control. It matches the input text against the pattern specified in the ValidationExpression property.
- Syntax:

```
          <asp:RegularExpressionValidator ID="someUniqueId"
runat="server"
    ControlToValidate ="someUniqueControlId“
ErrorMessage="ErrorToDisplayOnValidationFailure"
ValidationExpression=”aRegexPattern”>
</asp:RegularExpressionValidator>
```

- **Regex pattern** can be ^[a-z][0-9]. This indicates that a text must start with an alphabet and follow by a numeral.

5. CompareValidator

- The CompareValidator control compares the value of one control with either a fixed value or a value in another control.

- **Syntax**

```
          <asp:CompareValidator ID="someUniqueId"
runat="server"
  ControlToValidate ="someUniqueControlId"
ErrorMessage="ErrorToDisplayOnValidationFailure"
Type="string" ControlToCompare=”ControlToValidateId”>
</asp:CompareValidator>
```

- **ControlToCompare** – It holds the ControlToValidate Id of another form of control. The value of both the form fields is then compared.

6. **CustomValidator**

- ASP.Net also allows the freedom of writing your own validator.
- It also allows putting more complex validations in place.
- Validations that are business or application-specific can be written using custom validators.
- The custom validation code is written in a function in the code-behind page and the function name is passed as an attribute to the CustomValidator class.
- Custom validation can be done either at the client-side or the server-side.
- **Syntax**

```
          <asp:CustomValidator ID="someUniqueId"
runat="server“
    ControlToValidate ="someUniqueControlId"
ErrorMessage="ErrorToDisplayOnValidationFailure"
ClientValidationFunction=”functionName”>
</asp:CustomValidator>
```

2.17 SAVING STATE WITH STATEBAG OBJECTS

- View State Object Special object to manage information for current web page between different calls of same web page.
- A hidden variable called _VIEWSTATE is used to hold the values of ViewState object and resubmit on every click.
- ViewState is enabled by default so if you view a web form page in your browser you will see a line similar to the following near the form definition in your rendered HTML:

```
<input type="hidden" name="_____VIEWSTATE"
value="dDwxNDg5OTk5MzM7Oz7DblWpxMjE3ATl4Jx621QnCmJ2VQ==" />
```

For example:

- Create a web form having a login and password field and a login button.
- If login and password goes wrong for three times then disable button is given in the login message "Your Account is Blocked"

VB code:

```
Protected Sub Button1_Click(ByVal sender As Object, ByVal e As EventArgs)
Handles Button1.Click
    If Not (TextBox1.Text = "abc" AndAlso
    TextBox2.Text = "xyz") Then If ViewState("ec") IsNot
    Nothing Then
    count = Convert.ToInt32(ViewState("ec"))
    End If
count += 1
ViewState("ec") = count
Label1.Text = "count is "
& count
   If count = 3 Then
```

```
    Button1.Enabled
    = False
Label1.Text = "Your Account is blocked”
End If
    End If
    End Sub
```

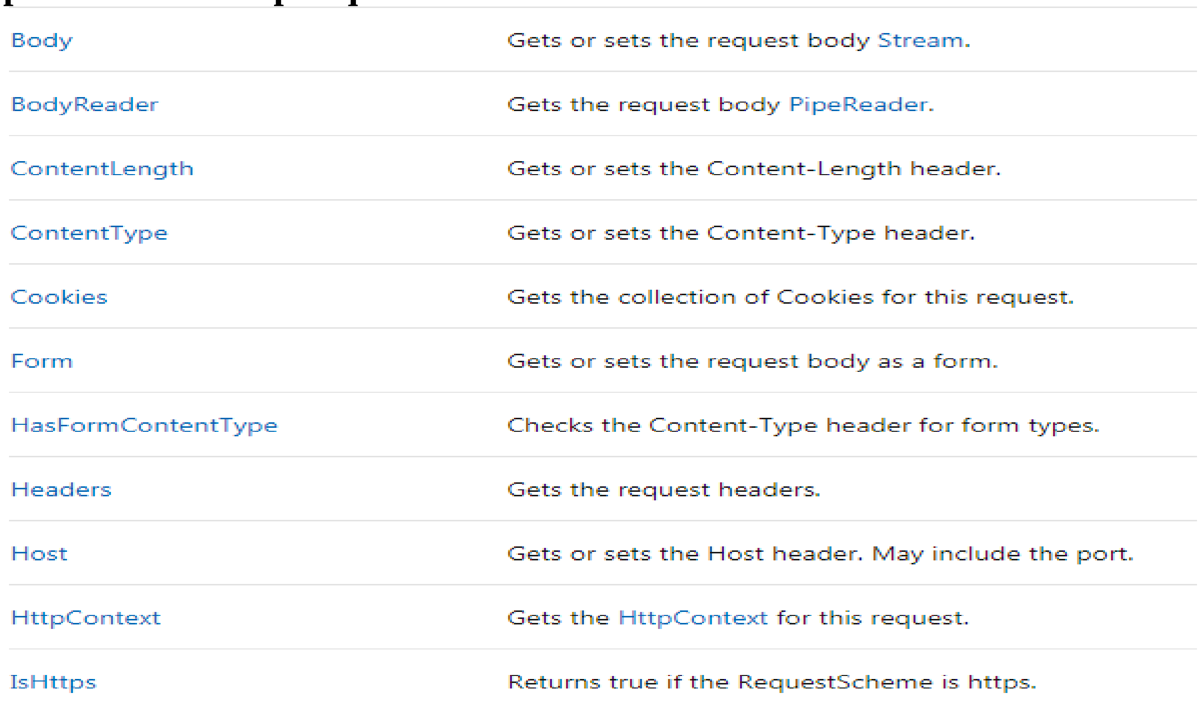
2.18 ASP.NET Intrinsic Objects

- ASP.NET provides intrinsic objects to enable low-level access to the Web application framework. With the help of these intrinsic objects, you can work directly with the underlying HTTP streams, server, session, and application objects. The intrinsic objects can be accessed in a Web form through the properties of the Page class.

.Intrinsic Objects and Their Mappings to the Page Class Properties

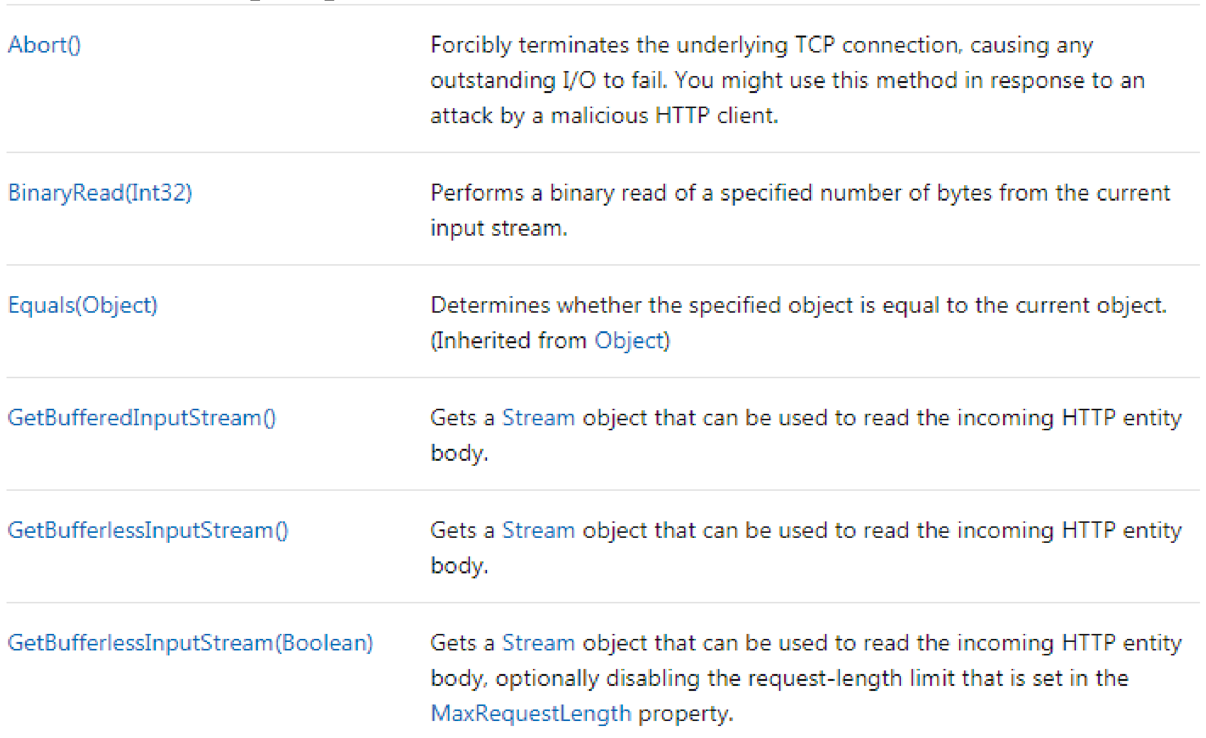
Intrinsic Object	**Property of the Page Class**
Http Request	Request
Http Response	Response
Http Server Utility	Server
Http Application State	Application
Http Session State	Session

Properties of the HttpRequest Class



Body	Gets or sets the request body Stream.
BodyReader	Gets the request body PipeReader.
ContentLength	Gets or sets the Content-Length header.
ContentType	Gets or sets the Content-Type header.
Cookies	Gets the collection of Cookies for this request.
Form	Gets or sets the request body as a form.
HasFormContentType	Checks the Content-Type header for form types.
Headers	Gets the request headers.
Host	Gets or sets the Host header. May include the port.
HttpContext	Gets the HttpContext for this request.
IsHttps	Returns true if the RequestScheme is https.

Methods of the HttpRequest Class



Abort()	Forcibly terminates the underlying TCP connection, causing any outstanding I/O to fail. You might use this method in response to an attack by a malicious HTTP client.
BinaryRead(Int32)	Performs a binary read of a specified number of bytes from the current input stream.
Equals(Object)	Determines whether the specified object is equal to the current object. (Inherited from Object)
GetBufferedInputStream()	Gets a Stream object that can be used to read the incoming HTTP entity body.
GetBufferlessInputStream()	Gets a Stream object that can be used to read the incoming HTTP entity body.
GetBufferlessInputStream(Boolean)	Gets a Stream object that can be used to read the incoming HTTP entity body, optionally disabling the request-length limit that is set in the MaxRequestLength property.

The HttpServerUtility Object

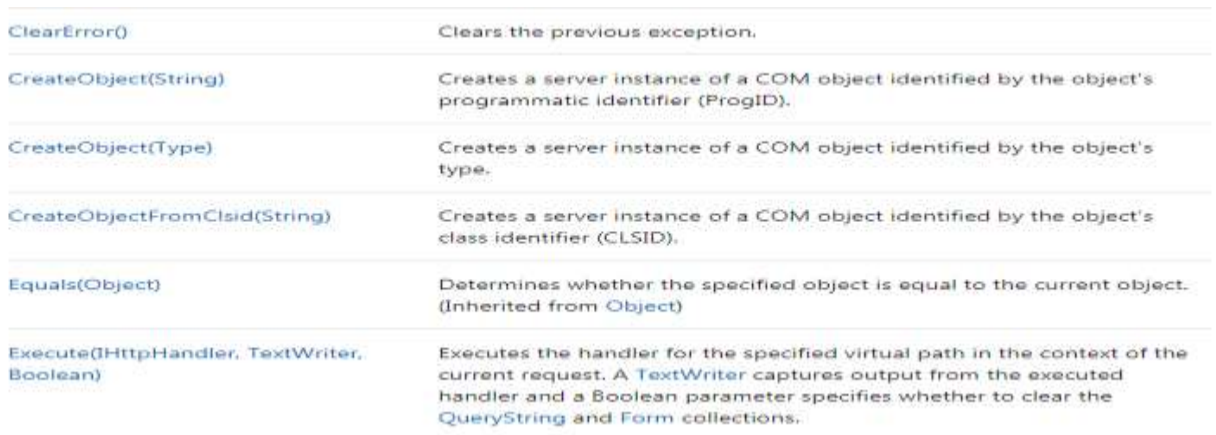
The HttpServerUtility object contains utility methods and properties to work with the Web server. It contains methods to enable HTML/URL encoding and decoding, execute or transfer to an ASPX page, create COM components, and so on.

➢ The Server property of the Page class provides access to the HttpServerUtility object.

Properties of the HttpServerUtility Class

Property	Description
Machine Name	Returns the name of the server that hosts the Web application.
ScriptTime out	Indicates the number of seconds that are allowed to elapse when processing the request before a timeout error is sent to the client.

Methods of the HttpServerUtility Class



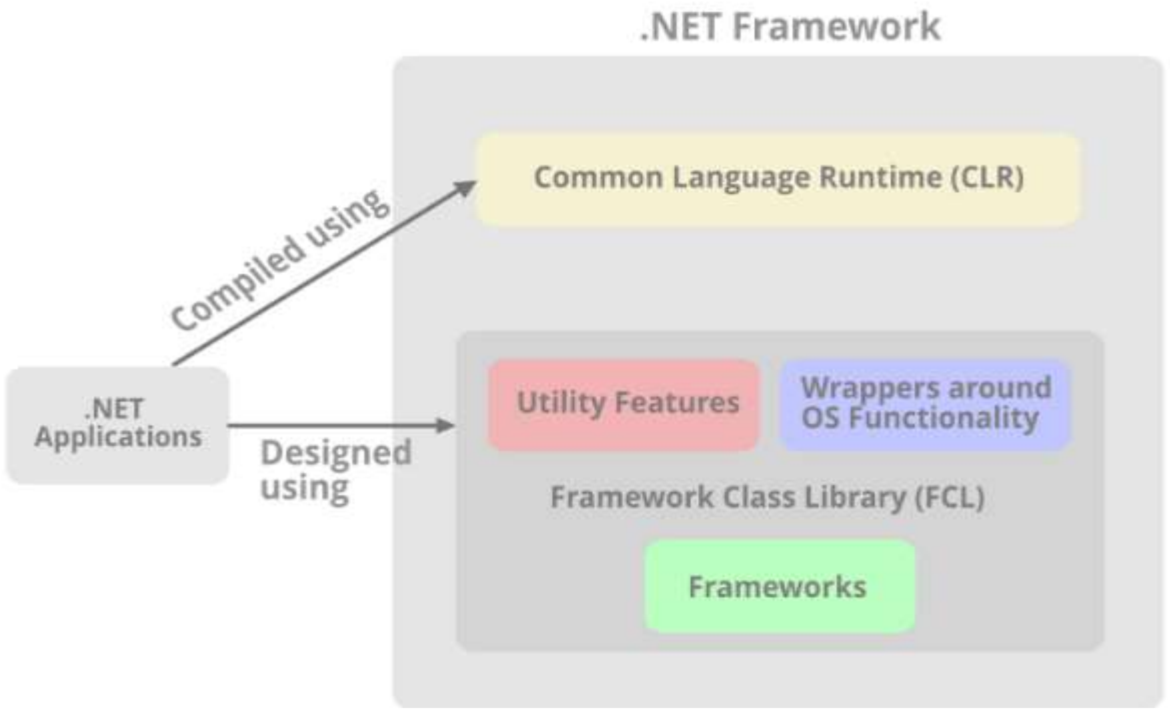
ClearError()	Clears the previous exception.
CreateObject(String)	Creates a server instance of a COM object identified by the object's programmatic identifier (ProgID).
CreateObject(Type)	Creates a server instance of a COM object identified by the object's type.
CreateObjectFromClsid(String)	Creates a server instance of a COM object identified by the object's class identifier (CLSID).
Equals(Object)	Determines whether the specified object is equal to the current object. (Inherited from Object)
Execute(IHttpHandler, TextWriter, Boolean)	Executes the handler for the specified virtual path in the context of the current request. A TextWriter captures output from the executed handler and a Boolean parameter specifies whether to clear the QueryString and Form collections.

UNIT- 3

Using the .NET Framework Class Library

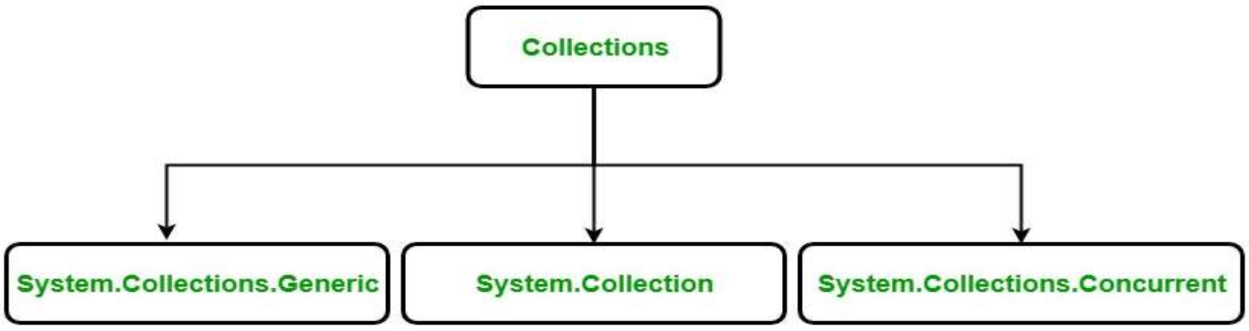
3.1 Common Features of the .NET Framework Class Library

- The Framework Class Library or FCL provides the system functionality in the **.NET Framework** as it has various classes, data types, interfaces, etc. to perform multiple functions and build different types of applications such as desktop applications, web applications, mobile applications, etc.
- The Framework Class Library is integrated with the Common Language Runtime (CLR) of the .NET framework and is used by all the .NET languages such as **C#**, F#, Visual Basic .NET, etc
- The functionality of the Framework Class Library can be broadly divided into **three** categories i.e *utility features written in .NET, wrappers around the OS functionality* and *frameworks.*
- These categories are not rigidly defined and there are many classes that may fit into more than one category.


.NET Framework
Common Language Runtime (CLR)
Compiled using
.NET Applications
Designed using
Utility Features
Wrappers around OS Functionality
Framework Class Library (FCL)
Frameworks

3.2 Using Data Collections (System.collections)

- Collections standardize the way of which the objects are handled by your program. In other words, it contains a set of classes to contain elements in a generalized manner.
- With the help of collections, the user can perform several operations on objects like the store, update, delete, retrieve, search, sort etc.
- C# divide collection in several classes, some of the common classes are shown below:



Below table contains the frequently used classes of the System.Collections namespace:

CLASS NAME	**DESCRIPTION**
ArrayList	It is a dynamic array means the size of the array is not fixed, it can increase and decrease at runtime.
Hashtable	It represents a collection of key-and-value pairs that are organized based on the hash code of the key.
Queue	It represents a first-in, first out collection of objects. It is used when you need a first-in, first-out access of items.
Stack	It is a linear data structure. It follows LIFO(Last In, First Out) pattern for Input/output.

3.3 Handling File Input/output and Directories (System.IO)

- The File class encapsulates various operations that can be performed on a single file, such as creating,copying,deleting,moving and opening files.

- Many of the methods of the file class return special classes that are used to read and write files.
- These are collectively known as stream classes.
- StreamReader is a class file which is designed for reading a line of information from a text file and it is inherited from the abstract base class TextReader and this base class represents a reader, which can read a sequential series of characters.
- The StreamWriter class is used to write data or information to the text file. It is a class file which is inherited from the abstract class TextWriter which can write a sequential series of characters.
- Here are some commonly used reader and writer classes:

  - ✓ BinaryReader and BinaryWriter – for reading and writing primitive data types as binary values.
  - ✓ StreamReader and StreamWriter – for reading and writing characters by using an encoding value to convert the characters to and from bytes.
  - ✓ StringReader and StringWriter – for reading and writing characters to and from strings.
  - ✓ TextReader and TextWriter – serve as the abstract base classes for other readers and writers that read and write characters and strings, but not binary data.

1. **Reading Text Files**
   - StreamReader type derives from a base class named TextReader.
   - Let us see some of the main members of the abstract class TextReader.
     - ✓ Read() -- Reads data from an input stream.
     - ✓ ReadLine() -- Reads a line of characters from the current stream and returns the data as a string.
     - ✓ ReadToEnd() -- Reads all characters to the end of the TextReader and returns them as one string.
2. **Writing Text Files**

- StreamWriter type derives from a base class named TextWriter. This class defines members that allow derived types to write textual data to a given character stream.
- Let us see some of the main members of the abstract class TextWriter.

  - ✓ close() -- Closes the Writer and frees any associated resources.
  - ✓ Write() -- Writes a line to the text stream, with out a newline.
  - ✓ WriteLine() -- Writes a line to the text stream, with a newline.
  - ✓ Flush() -- Clears all buffers.

Example:

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;

using System.IO;

public partial class ReadWriteFile : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }
    protected void btnReadFile_Click(object sender, EventArgs e)
    {
        //Reading from File1

        TextReader tr = new StreamReader(Server.MapPath("File-1.txt"));
        Label1.Text = tr.ReadLine();
        Label1.Text += tr.ReadToEnd();
        tr.Close();
    }
    protected void btnWriteFile_Click(object sender, EventArgs e)
    {
        //Writing into File2

        TextWriter tw = new StreamWriter(Server.MapPath("File-2.txt"));
        tw.WriteLine("Cherukuri Venkateswarlu");
        tw.Close();
    }
}
```

3.4 Watching the File System for Changes (System.IO.FileSystemWatcher)

- The FileSystemWatcher class in the System.IO namespace can be used to monitor changes to the file system.
- It watches a file or a directory in your system for changes and triggers events when changes occur.
- In order for the FileSystemWatcher to work, you should specify a directory that needs to be monitored.
- The FileSystemWatcher raises the following events when changes occur to a directory that it is monitoring.

- ✓ Changed: This event is triggered when a file or a directory in the path being monitored is changed
- ✓ Created: This event is triggered when a file or a directory in the path being monitored is created
- ✓ Deleted: This event is triggered when a file or a directory in the path being monitored is deleted
- ✓ Error: This event is triggered there is an error due to changes made in the path being monitored
- ✓ Renamed: This event is triggered when a file or a directory in the path being monitored is renamed

Creating a simple file system watcher in C#

- ➢ You can build a Windows Service that uses the FileSystemWatcher class and sends out notifications as and when changes occur to the path being watched.
- ➢ The following code snippet shows how the MonitorDirectory method would look like.
- ➢ This method would be used to monitor a particular directory and raise events whenever a change occurs. The directory path is passed as an argument to the method.
- ➢ Example:

```
private static void MonitorDirectory(string path)
  {
      FileSystemWatcher fileSystemWatcher = new FileSystemWatcher();
      fileSystemWatcher.Path = path;
      fileSystemWatcher.Created += FileSystemWatcher_Created;
      fileSystemWatcher.Renamed += FileSystemWatcher_Renamed;
      fileSystemWatcher.Deleted += FileSystemWatcher_Deleted;
      fileSystemWatcher.EnableRaisingEvents = true;
  }
```

3.5 Using the Windows Event Log

- ➢ It's common for server software to persistently record information about what it's
doing. This activity is called logging.
- ➢ It's common for programmers to spend time creating logging features in the

applications they write— both as a debugging tool and as a way for users and system.

- Programmers typically don't create their own logging facilities in the applications they create, because the operating system provides one for them: the Windows event log.
- The Windows event log is a central, system-managed place where any application, service, or device can log information.
- It is available on Windows NT and Windows 2000.
- Logged events usually contain status information or failure messages.
- You are free to use the event log from any of your applications, including ASP.NET applications, as a way of persistently storing information pertaining to your application's behavior in a way that's easy for system administrators to access.

Using the EventLog Class

- Any application, including ASP.NET applications, can access the event log.
- Your applications will write information to the application log (rather than to the security or system logs).
- The .NET framework provides a class for handling the event log. This is the EventLog class, found in System.Diagnostics.
- To send information to the application log, you use the EventLog class.
- This class is created for you automatically, so you do not need to instantiate it (although you can if you need to).
- Event sources must be registered before you write information to the log.
- You do this by calling the CreateEventSource method of the EventLog object writing an event to the Event Log Void

```
Button1_Click(Object Sender, EventArgs e)
{
 if(!EventLog.SourceExists("MyApp"))
{
    EventLog.CreateEventSource("MyApp", "Application");

}
EventLog.WriteEntry("MyApp", "This is just a test.",
   EventLogEntryType.Information);
}
```

- The WriteEntry method is overloaded; the code example shows the most commonly used form.
- The first parameter is a string representing the event source.
- The second parameter is the message to insert in the log.
- The third parameter is an event type; this is a member of the enumeration System.Diagnostics.EventLogEntryType.
- You can view the output of this code by launching the Event Viewer (found in Programs, Administrative Tools).
- After running the code, click Application Log.
- You should be able to see an Information entry for the MyApp event source.
- Double-clicking the event displays a property sheet that shows you the detail for the event (the description "This is just a test").

```
if(!EventLog.SourceExists("MyApp"))
{
 EventLog.CreateEventSource("MyApp", "Application");
}
EventLog.WriteEntry("MyApp", "This is just a test.",
   EventLogEntryType.Information);
}
```

3.6 Working with Active Directory Services

- The System.DirectoryServices.ActiveDirectory namespace provides a high level abstraction object model that builds around Microsoft Active Directory services tasks.
- The Active Directory service concepts such as forest, domain, site, subnet, partition, and schema are part of the object model.
- The System.DirectoryServices.ActiveDirectory namespace is used to automate Active Directory management tasks. System.DirectoryServices.ActiveDirectory is not used to access data that resides within Active Directory or any other directory service.
- The System.DirectoryServices namespace should be used for this purpose.
- *The System.DirectoryServices.ActiveDirectory namespace is intended for use by application developers* who are **familiar with .NET Framework programming using Visual Basic .NET or C#**.
- The services that ADS provides as the core functionality required by a centralized user management system.
  - ✓ **Domain Services**: Stores data and manages communications between the users and the DC. This is the primary functionality of AD DS.
  - ✓ **Certificate Services**: Allows your DC to serve digital certificates, signatures, and public key cryptography.
  - ✓ **Lightweight Directory Services**: Supports LDAP for cross platform domain services, like any Linux computers in your network.
  - ✓ **Directory Federation Services**: Provides SSO authentication for multiple applications in the same session, so users don't have to keep providing the same credentials.
  - ✓ **Rights Management**: Controls information rights and data access policies. For example, Rights Management determines if you can access a folder or send an email.

3.7 Using Message Queues (System.Messaging)

- **Message Queuing** is a message infrastructure and a development platform for creating distributed messaging applications for the Microsoft Windows

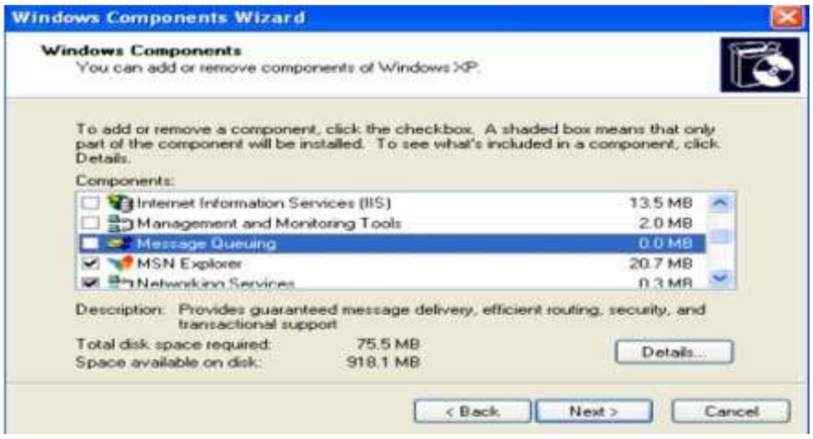
Operating System.

- Message Queuing applications can use the Message Queuing infrastructure to communicate heterogeneous networks and with computers that may be offline.
- Message Queuing provides guaranteed message delivery, efficient routing, security, transaction support and priority based messaging.
- Administrative Privileges are required to create a queue.
- Message Queuing is useful when the client application is often disconnected from the network.
- For example, salespersons can enter order data directory at the customer's site. The application sends a message for each order to the message queue that is located on the client's system.
- Message Queuing can also be useful in a connected environment. For example, the server of a website is fully loaded with order transactions at some specific time periods, say evening times or morning times, but the load is low at night time.
- The solution to this problem would be to buy a faster server or add an additional server to the system.
- There are different types of Message Queues:

1. Normal Message
2. Acknowledgement Message
3. Respond Message
4. Report Message

MSMQ HTTP Support:

This support allows you to send and receive messages using the HTTP protocol.



Windows Components

You can add or remove components of Windows XP.

To add or remove a component, click the checkbox. A shaded box means that only part of the component will be installed. To see what's included in a component, click Details.

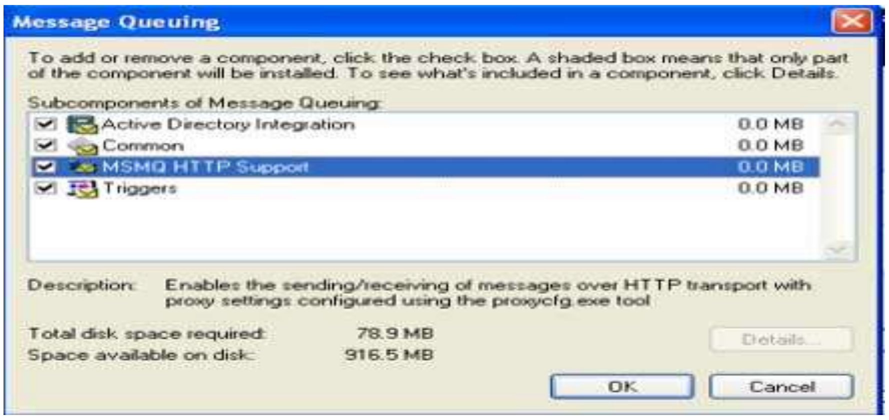
Components:

☐	Internet Information Services (IIS)	13.5 MB
☐	Management and Monitoring Tools	2.0 MB
☐	Message Queuing	0.0 MB
✓	MSN Explorer	20.7 MB
✓	Networking Services	0.3 MB

Description: Provides guaranteed message delivery, efficient routing, security, and transactional support.

Total disk space required: 75.5 MB

Space available on disk: 918.1 MB



To add or remove a component, click the check box. A shaded box means that only part of the component will be installed. To see what's included in a component, click Details.

Subcomponents of Message Queuing:

✓	Active Directory Integration	0.0 MB
✓	Common	0.0 MB
✓	MSMQ HTTP Support	0.0 MB
✓	Triggers	0.0 MB

Description: Enables the sending/receiving of messages over HTTP transport with proxy settings configured using the proxycfg.exe tool

Total disk space required: 78.9 MB

Space available on disk: 916.5 MB

Create a message Queues

- In C#, we can also create Message Queues programmatically using the Create() method of MessageQueue class. With Create() method, the path of the new queue must be passed.
- Once the create() method is invoked, properties of the queue can be changed. Using label property, the label of the queue is set to "First Queue".

Example

```
using System;
using System.Collections.Generic;
using System.Messaging;
using System.Text;

namespace FirstQueue
{
    class Program
    {
        static void Main(string[] args)
        {
            using (MessageQueue queue = MessageQueue.Create(@". \myqueue"))
            {
                queue.Label = "First Queue";
                Console.WriteLine("Queue Created:");
                Console.WriteLine("Path: {0}, queue.Path");
                Console.Writeline("FormatName: {0}, queue.FormatName");
            }
        }
    }
}
```

3.8 Manipulating XML Data

- XML is a ***cross-platform, hardware and software independent,*** text based markup language, which enables you to store data in a structured format by using meaningful tags.
- XML stores structured data in XML documents that are similar to databases. Notice that ***unlike Databases***, XML documents store data in the form of plain text, which can be used across platforms.
- In an XML document, you specify the structure of the data by creating a DTD or an XML schema.
- When you include a DTD in an XML document, the software checks the structure of the XML document against the DTD.
- This process of checking the structure of the XML document is called ***validating***.
- The software performing the task of validating is called a validating parser.
- The following code defines the structure of an XML document that will store data related to books:

-
  ```
  <?xml version="1.0"?>
  <Books>
   <Book bid="B001">
    <Title> Understanding XML </Title>
    <Price> $30 </Price>
  ```

```
    <author>
      <FirstName> Lily </FirstName>
      <LastName>
        Hicks <LastName>
        </author>
  </Book>
  <Book bid="B002">
    <Title> .NET Framework </Title>
    <Price> $45 </Price>
    <author>
      <FirstName> Jasmine </FirstName>
      <LastName>
        Williams <LastName>
        </author>
  </Book>
</Books>
```

XML Web Server Control

- An XML Web Server control is used to display the contents of an XML document without formatting or using XSL Transformations.
- You can optionally specify a XSLT style sheet that formats the XML document before it is displayed in an XML server control.
- The XML Web Server control belongs to the System.Web.UI.WebControls namespace.
- You can add an XML Web Server control to a Web form by dragging the control from the Web forms tab of the toolbox.

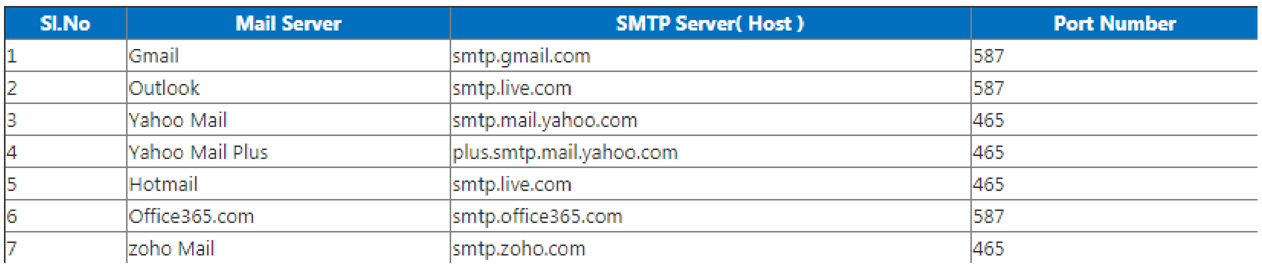
The XML Web Server control has the following properties:

- ***DocumentSource***: Allows you to specify the URL or the path of the XML document to be displayed in the Web form.

- ***TransformSource:*** Allows you to specify the URL of the XSLT file, which transforms the XML document into the required format before it is displayed in the Web form.
- ***Document:*** Allows you to specify a reference to an object of the XMLDocument class. This property is available only at runtime.
- ***Transform:*** Allows you to specify a reference to an object of the XMLTransform class. This property is available only at runtime.

3.9 Sending Internet E-mail

- Simple Mail Transfer Protocol (SMTP) is a TCP/IP protocol used in sending and receiving e-mail.
- Most e-mail systems that send mail over the Internet use SMTP to send messages from one server to another.
- The messages can then be retrieved with an e-mail client using either POP or IMAP.
- The following is a list of SMTP Server and Port Numbers:



Sl.No	Mail Server	SMTP Server(Host)	Port Number
1	Gmail	smtp.gmail.com	587
2	Outlook	smtp.live.com	587
3	Yahoo Mail	smtp.mail.yahoo.com	465
4	Yahoo Mail Plus	plus.smtp.mail.yahoo.com	465
5	Hotmail	smtp.live.com	465
6	Office365.com	smtp.office365.com	587
7	zoho Mail	smtp.zoho.com	465

Important SMTP Class Properties:

- **Host:** Server URL for SMTP (check the preceding table).
- **EnableSsl:** Check your host accepts SSL Connections (True or False).
- **Port:** Port Number of the SMTP server (check the preceding table).
- **Credentials:** Valid login credentials for the SMTP server (the email address and password).

- **UseDefaultCredentials:** When we set to True in UseDefaultCredentials then that specifies to allow authentication based on the credentials of the account used to send emails.

Unit – 4

Building .NET Managed Components for COM+

- Modern software systems are component-based. Components make large, enterprise-wide software systems easier to manage since system functionality is divided among several components.
- VB.NET and all other .NET-supported languages have the ability to create these components, which can be used and reused in a variety of projects, including (but not limited to) ASP.NET projects, Windows applications, and unmanaged code.

4.1 The concept of Managed Code Execution

- Managed code is any source code you compile that targets the .NET Framework.
- All the code we've examined so far in this book (ASP.NET, console applications, and so on) has been managed code.
- The .NET Framework is a runtime environment in which code executes. Internal tasks, such as allocating and freeing memory, and software services (like the kinds discussed in Chapter 4) are handled by the .NET Framework as well.
- As a result, many Windows applications could be distributed with as little as one file, the .exe file that contained the main program code. Since no other files were required, many developers referred to these types of applications as native code applications since the preinstalled runtime code was quite small.
- VB developers using version 6.0 or earlier might be aware of special support files that must be installed on the deployment computer in order for VB applications to run.
- These .dll files make up the VB runtime environment. Like the C/C++ runtime code mentioned above, it provides a code wrapper around operating system internals and services.
- Programs targeted to such a runtime don't compile to machine object code.
- Instead, they compile to another language, referred to as an intermediate language.
- The runtime then executes this intermediate language using an engine built for a particular operating system.

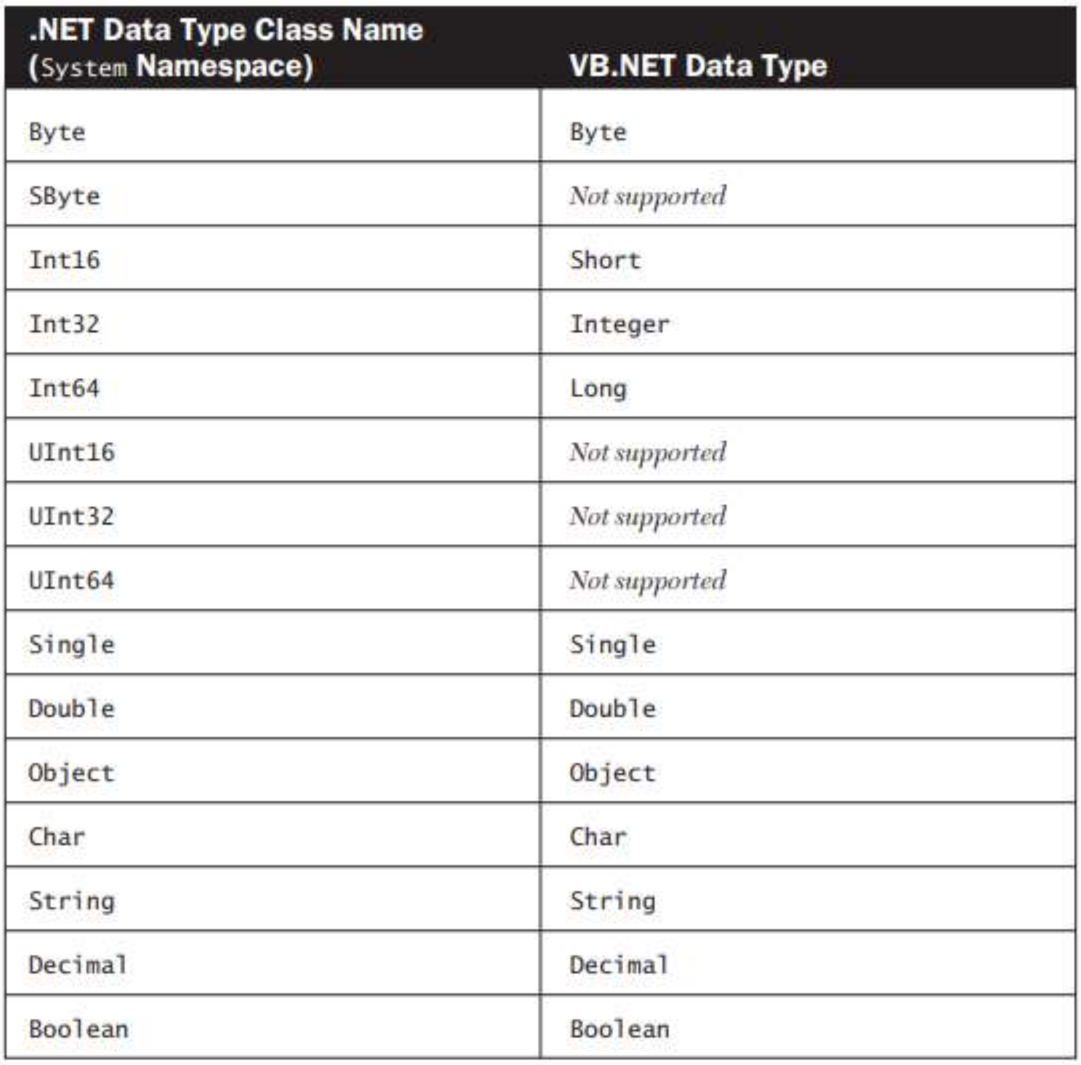
- The most popular type of intermediate language system is Java. The intermediate language for Java is referred to as bytecode.
- The .NET Framework works in a similar manner.
- Compiled Microsoft .NET code is referred to as Microsoft Intermediate Language (MSIL).

4.2 The Common Language Runtime

- The runtime environment for the .NET Framework is called the Common Language Runtime (CLR).
- Managed code execution happens inside the CLR space. The goal of the CLR is to provide an environment that includes language integration, exception handling, security, versioning, deployment, debugging, profiling, and component interaction.
- Metadata makes cross-language integration possible. When you compile .NET managed code, the metadata gets stored along with the object code.
- Metadata describes to the CLR various types of information (for example, data types, members, and references) used in the code.
- This data is used by the CLR to "manage" the code execution by providing such services as memory allocation, method invocation, and security enforcement.

  The Common Type System

  - CLR implements a series of data types that are cross-language compatible. That system of data types is known as the Common Type System (CTS).
  - CTS data types include simple value types, classes, enumerated value types, interfaces, and delegates.
  - Simple value data types include primitive types as well as user-defined types.
  - Primitive types include integers, Boolean values, and strings. These types are included in the System namespace.
  - The data types used thus far in the VB.NET programs in this book are also included in this namespace.
  - Primitive Data Types and VB.NET Equivalents



.NET Data Type Class Name (System Namespace)	VB.NET Data Type
Byte	Byte
SByte	*Not supported*
Int16	Short
Int32	Integer
Int64	Long
UInt16	*Not supported*
UInt32	*Not supported*
UInt64	*Not supported*
Single	Single
Double	Double
Object	Object
Char	Char
String	String
Decimal	Decimal
Boolean	Boolean

Example:

```
Structure Student  -------------   1
        Dim FirstName As String
        Dim LastName As String
        Dim SSN As String
        Dim ClassRank As Integer
End Structure
 With udtStudent  ---------------- 2
                .FirstName = "Matt"
                .LastName = "Crouch"
                .SSN = "888-88-1234"
                 .ClassRank = 2
End With
```

- The example shows a typical structure definition. The members are enclosed in the Structure . . . End Structure block that begins in line 1.

- The VB.NET With . . . End With statement in line 2 is used to save some typing. It allows you to refer to the individual members of the structure without fully qualifying the names of the structure members.

4.3 COM+ Component Services

- A particular technology that deserves attention is COM+ Component Services.
- These technologies were the predecessors to the .NET Framework.

Overview of COM

- The Component Object Model (COM) was designed to address the shortcomings of conventional object-oriented languages like C++ and traditional binary software distribution of code.
- COM is about not a particular type of software but rather a philosophy of programming.
- COM objects are roughly equivalent to normal classes, but COM defines how these objects interact with other programs at the binary level.
- This "binary encapsulation" allows you to treat each COM object as a "black box."
- In the Windows environment, these binary objects (COM objects) are packaged as either DLLs or executable programs.
- COM is also backed by a series of utility functions that provide routines for instantiating COM objects, process communication, and so on.
- COM was the first methodology to address object-oriented software reuse.
- COM has enjoyed great commercial success; many third-party software vendors provide COM objects to perform a wide range of tasks, from e-mail to image processing.
- COM is also highly useful for creating components called business objects.

Overview of Transactions

- Several smaller steps are involved in a transaction.
- The success or failure of the transaction depends on whether or not all of the smaller steps are completed successfully.

- If a failure occurs at any point during a transaction, you don’t want any data changes made by previous steps to remain.
- A transaction is committed when all steps have succeeded. A failed transaction causes a rollback to occur.
- Well-designed transactions conform to ACID principles. ACID is an acronym for Atomicity, Consistency, Isolation, and Durability.

■ **Atomicity** means that either the operation that the component performs is completely successful or the data that the component operates on does not change at all. If a failure occurs at any step that could compromise the integrity of the system, the changes are undone.

■ **Consistency** deals with preserving the system state in the case of a transaction failure.

■ **Isolation** means that a transaction acts as though it has complete control of the system. In effect, this means that transactions are executed one at a time.

■ **Durability** is the ability of a system to return to any state that was present before the execution of a transaction.

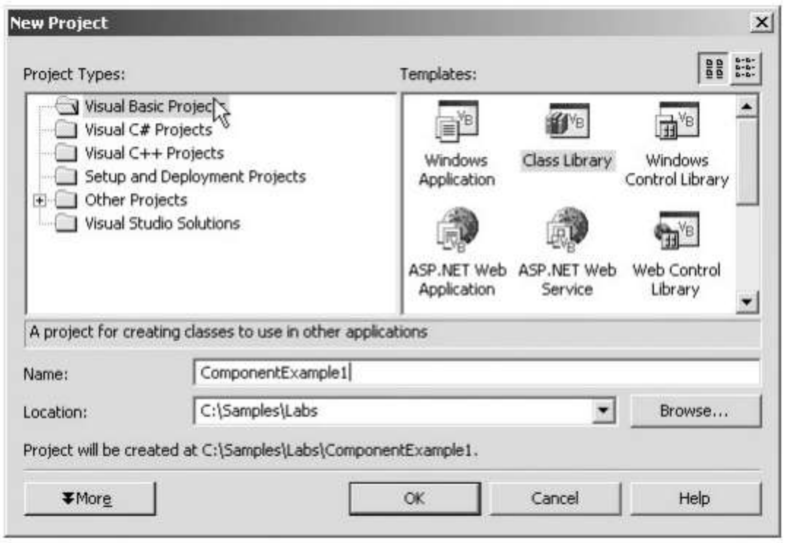
4.4 **Using VB.NET to Develop Managed Components**

- The concepts you need to understand to build managed components using VB.NET.
  Defining Namespaces
- Namespaces are a way to hierarchically organize class names. Namespaces become especially useful when the number of classes available to you is quite large.
- You’ve been exposed to namespaces quite a bit already in this book. Whenever you write VB.NET code, you use .NET classes in the System namespace.
- For example, when you use the Console.WriteLine() command, you are using an assembly called Console that exists in the System namespace.

```
System.Console.WriteLine( . . . )
```

Using the Class Library

- Managed components start with a VS.NET project called a class library.
- The class library puts the infrastructure in place for creating an assembly for packing a component.



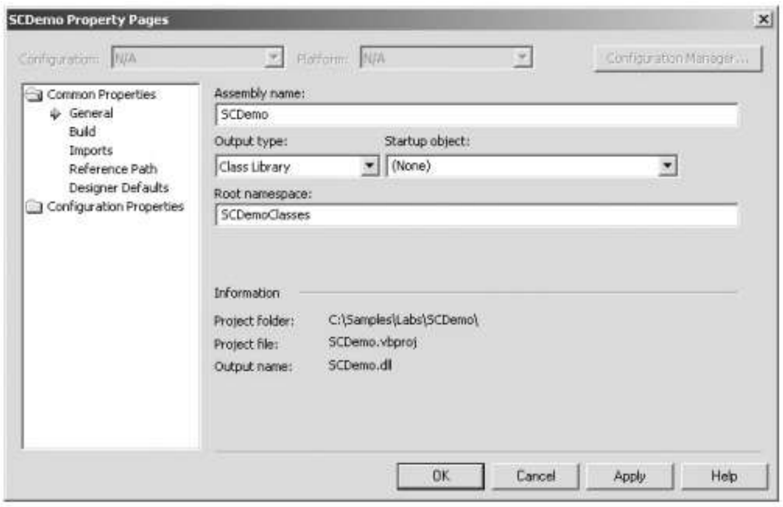
A project for creating classes to use in other applications

Name: ComponentExample1

Location: C:\Samples\Labs

Project will be created at C:\Samples\Labs\ComponentExample1.

- You can view and modify them by selecting the Project➔Properties menu. This opens the Property Pages dialogshown below



Assembly name: SCDemo

Output type: Class Library

Startup object: (None)

Root namespace: SCDemoClasses

Information

Project folder: C:\Samples\Labs\SCDemo\

Project file: SCDemo.vbproj

Output name: SCDemo.dll

Using Component “Plumbing” Code

- When class library projects are started in VB.NET, some stub code is generated for you.
- A new, “empty” component’s “plumbing” code looks something like this: Public Class Class1 End Class
- The VS.NET IDE adds this code for you for an empty class called Class1.

- A .vb file is also generated that contains the class's code (which VS.NET should be displaying in the front-most window).
- Give the class another name (in the example below, we'll name it CTaxComponent) by changing it in the editor and then renaming the class file in the Solution Explorer to CTaxComponent.
- The code should now say:

```
Public Class CTaxComponent
.
.
.
 End Class
```

Adding Initialization Code

- Whenever your component is instantiated, the New() subroutine is called.
- Any initialization code you require goes here.
- As an illustration, suppose your class contains a private member variable that you wish to have an initial default value when the component is instantiated.
- Such initialization code could look like the following.

```
 Private m_dblTaxRate As Double

 Public Sub New()

        MyBase.New ' Specify default percent tax rate

        m_dblTaxRate = 5

End Sub
```

Creating Methods

- When creating class libraries and components, you'll notice that the same principles apply to components. Methods are simply member functions and subroutines.

- In order for other .NET applications to access methods you define, you need to expose them by marking them for public access.
- Here’s a public method definition and implementation to illustrate. Public Function CalculatePriceAfterTax(_ ByVal Amount As Double) As Double CalculatePriceAfterTax = Amount * (1 + (m_dblTaxRate / 100)) End Function

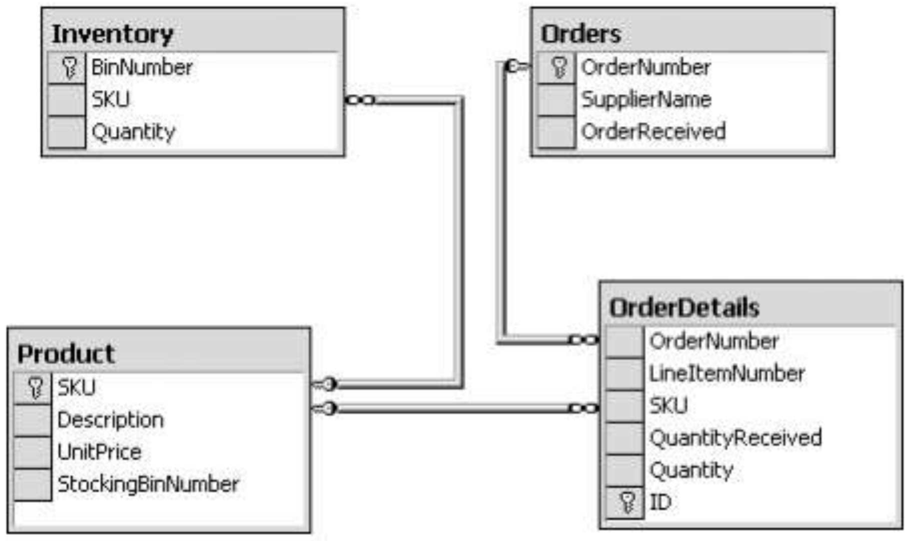
Creating Properties

  - Component properties are created in the same way as properties for regular classes. Like methods, they should be marked for public access.
  - The property in the sample code below sets or gets the value of the private class member variable used in the previous example.

```
Public Property TaxRate() As Double
Get
        TaxRate = m_dblTaxRate
End Get
Set(ByVal Value As Double)
        m_dblTaxRate = Value
End Set
End Property
```

4.5 Serviced Components

- With COM+ Component Services, your components can take advantage of automatic transaction processing, object pooling, and role-based security.
- Components that use COM+ Component Services are called serviced components.
- Setting up a component to take advantage of these features is not difficult.
- It requires creating a class library project as shown earlier plus creating several component classes to handle the business logic.
- Example: Three different types of users will work with this system: Suppliers, Supervisors, and Receiving Clerks.



- The above shows the database schema used for the ordering and inventory system, showing tables, columns, and foreign key relationships among columns.

5.6 Building VB.NET Serviced Components

AN ORDERING AND INVENTORY SYSTEM MADE WITH SERVICED COMPONENTS

STEP 1. Create a new VB.NET class library project.

a. Open VS.NET and select File➔New➔Project.

b. Highlight Class Library and give the new project the name “SupermarketSystem”.

STEP 2. Design the components.

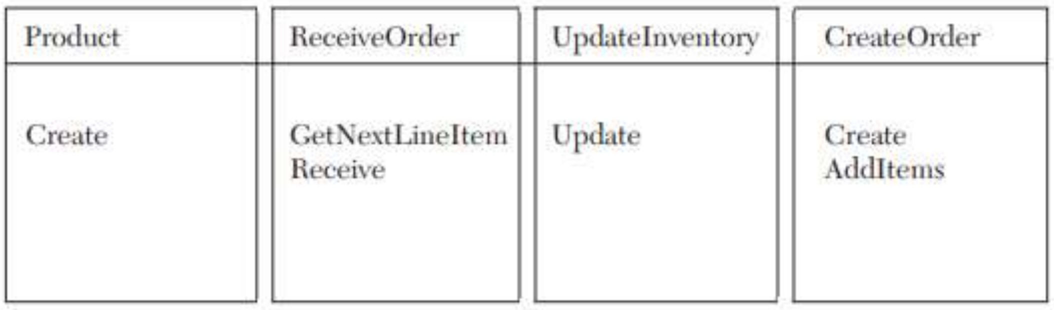
a. Consider the design of your system. First, establish the users of the system and their roles:

• Supervisors (adding new products, receiving orders, updating inventory)

• Receiving Clerks (receiving orders)

• Suppliers (shipping goods to the supermarket and supplying order information)

b. Organize these functions into a series of classes for your class library. Four classes will handle the business logic of the different duties that each user will perform: Product, ReceiveOrder, UpdateInventory, and CreateOrder. Lab Figure shows below have the classes and the methods of each class.



STEP 3. Write code for component functionality. In the process of implementing the components, we'll discuss these topics:

• How to use the System.EnterpriseServices namespace and one particular class: the ServicedComponent class

• How to use class attributes to specify the type of COM+ support you want your components to have

• How to specify an interface from which to call the components

• How to control the transaction inside the components

a. Create the classes outlined in the specifications in Step 2: Product, ReceiveOrder, UpdateInventory, and CreateOrder. For each of these classes, right-click the project in the Solution Explorer and select Add

- Building serviced components requires support from COM+ Component Services.
- The .NET Framework implements this support through classes in the System.EnterpriseServices namespace.
- VS.NET doesn't include this reference by default, so you'll need to add it yourself. Select Project➔Add Reference from the menu and select System.EnterpriseServices from the list.
- Click the Select button and then click OK.

c. Now you're ready to add component functionality. Start by adding the following code for the Product class.

STEP 4. Create the ASP.NET application.

a. Create a new VB ASP.NET project called “SupermarketWeb”.

b. Add a reference to System.EnterpriseServices to the project by rightclicking the project icon and selecting Add Reference . . . from the pop-up menu

STEP 5. Run the application.

STEP 6. Add role-based security

STEP 7. Test the role-based security.

4.6 The need for Web Services

- ➢ A web service is a unit of managed code that can be remotely invoked using HTTP.
- ➢ That is, it can be activated using HTTP requests. Web services allow you to expose the functionality of your existing code over the network.
- ➢ Once it is exposed on the network, other applications can use the functionality of your program.
- ➢ Different books and different organizations provide different definitions to Web Services. Some of them are listed here.
- ➢ A web service is any piece of software that makes itself available over the internet and uses a standardized XML messaging system.
- ➢ XML is used to encode all communications to a web service. For example, a client invokes a web service by sending an XML message, then waits for a corresponding XML response.
- ➢ As all communication is in XML, web services are not tied to any one operating system or programming language—Java can talk with Perl; Windows applications can talk with Unix applications.
- ➢ Web services are self-contained, modular, distributed, dynamic applications that can be described, published, located, or invoked over the network to create products, processes, and supply chains.

- These applications can be local, distributed, or web-based. Web services are built on top of open standards such as TCP/IP, HTTP, Java, HTML, and XML.
- Web services are XML-based information exchange systems that use the Internet for direct application-to-application interaction. These systems can include programs, objects, messages, or documents.
- A web service is a collection of open protocols and standards used for exchanging data between applications or systems.
- Software applications written in various programming languages and running on various platforms can use web services to exchange data over computer networks like the Internet in a manner similar to inter-process communication on a single computer.
- This interoperability (e.g., between Java and Python, or Windows and Linux applications) is due to the use of open standards.
- To summarize, a complete web service is, therefore, any service that −
  - ✓ Is available over the Internet or private (intranet) networks
  - ✓ Uses a standardized XML messaging system
  - ✓ Is not tied to any one operating system or programming language
  - ✓ Is self-describing via a common XML grammar
  - ✓ Is discoverable via a simple find mechanism

4.7 Overview of Web Services

- Web services are compiled classes just like the classes we have worked with up to this point.
- These classes are contained within a .asmx file.
- This is how ASP.NET identifies a web service.

The .asmx file

- *Active Server Method File*, a file with the ASMX file extension is an ASP.NET Web Service Source file.
- Unlike ASP.NET web pages that use the .ASPX file extension, ASMX files function as a service that doesn't have a graphical user interface and instead is used to move data and perform other actions behind the scenes.
- ASMX are XML based files and always start with the "@ WebService" directive which is used to correlate the URL (Uniform Resource Locator) address of the Web Service with its implementation.
- This is followed by the "Language" parameter which is needed to set the programming language to JScript, C# or Visual Basic.

```
<%@ Webservice Language=”VB” class=”mywebserviceclass” %>

        Imports System.Web.Services
```

  - The second line denotes that you wish to import the required class libraries that enable web services to work.

Web Service Classes And Web Methods

- A Web services begins with a class definition.
- Web services classes are just like classes that we have created before but with a couple of distinguishing elements.
- Simple class definition

```
Public class firstclass

Inherits System.web,services,webservice

<webMethod>Public Function Helloworld() as string

        Helloworld = “Hello World”

End Function
```

4.8 Web Services Description Language

- WSDL stands for Web Services Description Language. It is the standard format for describing a web service.
- WSDL was developed jointly by Microsoft and IBM.

Features of WSDL

- WSDL is an XML-based protocol for information exchange in decentralized and distributed environments.
- WSDL definitions describe how to access a web service and what operations it will perform.
- WSDL is a language for describing how to interface with XML-based services.
- WSDL is an integral part of Universal Description, Discovery, and Integration (UDDI), an XML-based worldwide business registry.
- WSDL is the language that UDDI uses.
- WSDL is pronounced as 'wiz-dull' and spelled out as 'W-S-D-L'.

WSDL Usage

- WSDL is often used in combination with SOAP and XML Schema to provide web services over the Internet.
- A client program connecting to a web service can read the WSDL to determine what functions are available on the server.
- Any special datatypes used are embedded in the WSDL file in the form of XML Schema. The client can then use SOAP to actually call one of the functions listed in the WSDL.

The WSDL Document Structure

- The main structure of a WSDL document looks like this −

```
<definitions>
  <types>
    definition of types........
  </types>

  <message>
    definition of a message....
```

```
   </message>

   <portType>
      <operation>
         definition of a operation.......
      </operation>
   </portType>

   <binding>
      definition of a binding....
   </binding>

   <service>
      definition of a service....
   </service>
</definitions>
```

- A WSDL document can also contain other elements, like extension elements and a service element that makes it possible to group together the definitions of several web services in one single WSDL document.

4.9 Web Service Wire Formats

- The final piece of the ASP.NET Web Services infrastructure consists of the Web service wire formats.
- Wire formats define the method by which Web service request and response messages are encoded and transported between the Web service and any consumer.
- To maximize the reach of Web services on the Internet, standard Internet protocols are used.
- ASP.NET Web Services support three wire formats:

1. **HTTP-GET:**

- The HTTP-GET protocol encodes Web service operation requests and arguments in the URL to the Web service. The operation is coded as part of the URL string and any arguments are coded as query string parameters appended to the base URL.
- For example:

  http://localhost/ctemp/ctemp.asmx/ctemp?Temperature= 32&FromUnits=F&ToUnits=C

- This URL specifies the Web-addressable entry point for the CTemp Web service (Ctemp.asxm), including the method to be called (also named ctemp).
- The arguments to the ctemp method are passed as query string arguments to the method request.
- Similar to the way in which WSDL documents are generated and returned to requests for such information via a URL to the Web service entry point file (the ASMX file), the HTTP-GET method of calling Web service methods is handled by theWebServiceHandlerFactory class.
- This class takes the uRl and query string parameters as input and translates this into a method call on the appropriate Web service class implementation.
- .NET Proxy classes use the HttpGetClientProtocol class in the System.Web.Services. Protocols namespace to invoke Web services that support the HTTP-GET protocol.

2. HTTP-POST :

- The HTTP-POST protocol encodes Web service operation requests and arguments within the payload area of the HTTP-POST request as name/value pairs.
- This technique of invoking a Web service method is identical in operation to the HTTP-GET method (from an ASP.NET perspective), except in the way in which the Web service call arguments are passed to the server.
- Once again, the .NET Framework's WebServiceHandlerFactory class is responsible for extracting the method name and arguments from the request and calling the appropriate Web service method found in the Web service class implementation.
- .NET Proxy classes use the HttpPostClientProtocol class in the System.Web.Services. Protocols namespace to invoke Web services that support the HTTP-POST protocol.

3. HTTP-SOAP.

- HTTP-SOAP is the default ASP.NET Web Service wire format.
- It is based on the SOAP specification and supports the widest range of simple and complex data types (including document-oriented operations).
- Web service request and response messages are encoded into SOAP messages that are included in the payload area of an HTTP-POST message.

- SOAP messages are encoded in XML using the SOAP vocabulary defined in the specification.
- .NET Proxy classes use the HttpSimpleClientServiceProtocol class in the System. Web.Services.Protocols namespace to invoke Web services that support the HTTP-SOAP protocol.

4.10 Web Services Discovery

- Web Services Discovery provides access to software systems over the Internet using standard protocols.
- In the most basic scenario there is a Web Service Provider that publishes a service and a Web Service Consumer that uses this service. Web Service Discovery is the process of finding suitable web services for a given task.
- Publishing a web service involves creating a software artifact and making it accessible to potential consumers.
- Web service providers augment a service endpoint interface with an interface description using the Web Services Description Language (WSDL) so that a consumer can use the service.
- The Web services framework provides two methods for discovering web services:Static Discovery and Dynamic discovery

Static Discovery:

- **Static discovery allows you to explicitly define the location of individual web services.**
- **It also allows you to group together the location of web services into logical unit.**
- **A Discovery (.disco) file is, not surprisingly,an XML file with a defined schema.**

```
<disco:discovery
.
.
</disco:discovery>
```

Dynamic Discovery:

- Dynamic discovery provides an automatic way to discover multiple web services.
- It allow user to discover every available web services underneath a given URL.
- A dynamic discovery file looks like the following code:

```
<dynamicdiscovery>
```

```
<exclude path="">
.
</dynamicdiscovery>
```

- Automatically enables the discovery of web services in the directory in which the file resides.Notice the <exclude> tags in the file.

4.11 COMPONENTS OF WEB SERVICE XML-RPC

This is the simplest XML-based protocol for exchanging information between computers.

1. XML-RPC is a simple protocol that uses XML messages to perform RPCs.
2. Requests are encoded in XML and sent via HTTP POST.
3. XML responses are embedded in the body of the HTTP response.
4. XML-RPC is platform-independent.
5. XML-RPC allows diverse applications to communicate.
6. A Java client can speak XML-RPC to a Perl server.
7. XML-RPC is the easiest way to get started with web services.
8. SOAP
9. SOAP is an XML-based protocol for exchanging information between computers.
10. SOAP is a communication protocol.
11. SOAP is for communication between applications.
12. SOAP is a format for sending messages.
13. SOAP is designed to communicate via Internet.
14. SOAP is platform independent.
15. SOAP is language independent.
16. SOAP is simple and extensible.
17. SOAP allows you to get around firewalls.
18. SOAP will be developed as a W3C standard.
19. WSDL
20. WSDL is an XML-based language for describing web services and how to access them.
21. WSDL stands for Web Services Description Language.
22. WSDL was developed jointly by Microsoft and IBM.
23. WSDL is an XML based protocol for information exchange in decentralized and distributed environments.
24. WSDL is the standard format for describing a web service.

25. WSDL definition describes how to access a web service and what operations it will perform.
26. WSDL is a language for describing how to interface with XML-based services.
27. WSDL is an integral part of UDDI, an XML-based worldwide business registry.
28. WSDL is the language that UDDI uses.
29. WSDL is pronounced as 'wiz-dull' and spelled out as 'W-S-D-L'
30. UDDI
31. UDDI is an XML-based standard for describing, publishing, and finding web services.
32. UDDI stands for Universal Description, Discovery, and Integration.
33. UDDI is a specification for a distributed registry of web services.
34. UDDI is platform independent, open framework.
35. UDDI can communicate via SOAP, CORBA, and Java RMI Protocol.
36. UDDI uses WSDL to describe interfaces to web services.
37. UDDI is seen with SOAP and WSDL as one of the three foundation standards of web services.
38. UDDI is an open industry initiative enabling businesses to discover each other and define how they interact over the Internet.

UNIT-5

Accessing Data with ADO.NET

- **ActiveX Data Object.NET** (ADO.NET) is a software **library** in the .NET framework consisting of software components **providing data access services**.
- ADO.NET is designed to enable developers to write managed code for obtaining **disconnected access to data sources**, which **can be relational or non-relational** (such as XML or application data).
- This feature of ADO.NET helps to create data-sharing, distributed applications.

5.1 Overview of Data Access on the Web

- Data access technologies have a long and colourful history,from earliest file based data stores to ADO.NET.
- The Evolutionary path of data access

1.Flat Files: A **flat-file database** is a database stored in a file called a **flat file**. Records follow a uniform format, and there are no structures for indexing or recognizing relationships between records. The file is simple. A flat file can be a plain text file, or a binary file. Relationships can be inferred from the data in the database, but the database format itself does not make those relationships explicit.

2. Legacy or Maniframe Data: The program controlling the terminals session,in turn,passes the “Scraped” data back to the browser.It a slow process,and the ways you can retrieve,update and assiminlate data are limited by established interface of the mainframes character based user program.

3. Proprietary Database APIs: An **API** is a set of programming code that enables data transmission between one software product and another. It also contains the terms of this data exchange. APIs are sets of requirements that govern how one application can communicate and interact with another. APIs can also allow developers to access certain internal functions of a program, although this is not typically the case for web APIs. In the simplest terms, an API allows one piece of software to interact with another piece of software, whether within a single computer via a mechanism provided by the operating system or over an internal or external TCP/IP-based or non-TCP/IP-based network

4. Standard APIs : ODBC is a single C language API used to access relational databases on the wide variety of distributed system.ODBC has two main drawbacks:

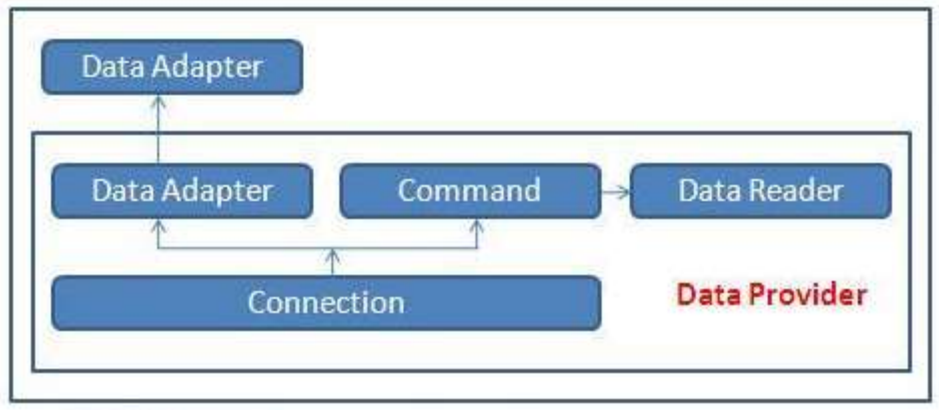
1. The API Functions have only a call level interface compatible with this language.

2.ODBC requires many calls to the ODBC API to perform the simplest of operations.Many of the low level details of database access with ODBC are left to the programmer.

5.2 ADO.NET programming objects and architecture

- ADO.NET consist of a set of Objects that expose data access services to the .NET environment.
- It is a data access technology from Microsoft .Net Framework , which provides communication between relational and non relational systems through a common set of components .
- System.Data namespace is the core of ADO.NET and it contains classes used by all data providers.
- ADO.NET is designed to be easy to use, and Visual Studio provides several wizards and other features that you can use to generate ADO.NET data access code.

Data Providers and DataSet



- The two key components of ADO.NET are Data Providers and DataSet . 1. The Data Provider classes are meant to work with different kinds of data sources. They are used to perform all data-management operations on specific databases.

  2. DataSet class provides mechanisms for managing data when it is disconnected from the data source.

Data Providers

- The .Net Framework includes mainly three Data Providers for ADO.NET. They are the Microsoft SQL Server Data Provider , OLEDB Data Provider and ODBC Data Provider .
- SQL Server uses the SqlConnection object , OLEDB uses the OleDbConnection Object and ODBC uses OdbcConnection Object respectively.
- A data provider contains Connection, Command, DataAdapter, and DataReader objects. These four objects provides the functionality of Data Providers in the ADO.NET.

Connection

- The Connection Object provides physical connection to the Data Source. Connection object needs the necessary information to recognize the data source and to log on to it properly, this information is provided through a connection string.

Command

- The Command Object uses to perform SQL statement or stored procedure to be executed at the Data Source. The command object provides a number of Execute methods that can be used to perform the SQL queries in a variety of fashions.

DataReader

- The DataReader Object is a stream-based , forward-only, read-only retrieval of query results from the Data Source, which do not update the data. DataReader requires a live connection with the databse and provides a very intelligent way of consuming all or part of the result set.

DataAdapter

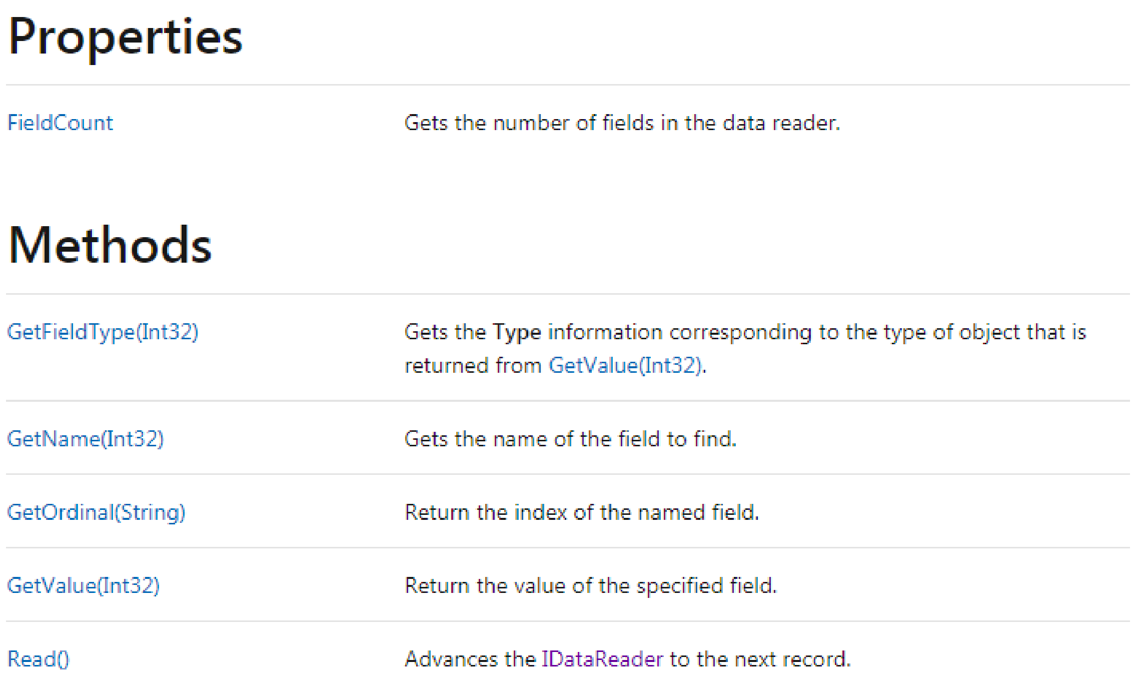
- DataAdapter Object populate a Dataset Object with results from a Data Source . It is a special class whose purpose is to bridge the gap between the disconnected Dataset objects and the physical data source.

5.3 **Displaying Database Data**

- The Simplest of ADO.NET code using the IDataReader interface to read data from a single database.

IDataReader interface (System.Data.IDataReader)

- The IDataReader interface is implemented by all data readers. A data reader provides a way to read a command-only stream of data from a data source.
- You can't create a data reader directly.
- Instead, an open IDataReader instance is returned from the IDbCommand.ExecuteReader( ) method.
- You can then use the Read( ) method to move from one row to the next. The Read( ) method returns true if it can successfully read a row or false if there are no more rows remaining, and it has moved beyond the bounds of the result set.
- There is no way to move backward. When it is first created, the IDataReader doesn't yet point to a row.
- You must call Read( ) to move to the first row before you attempt to read any information.
- You can use NextResult( ) to move to the next result set if you are using a command or stored procedure that returns multiple result sets.



Properties

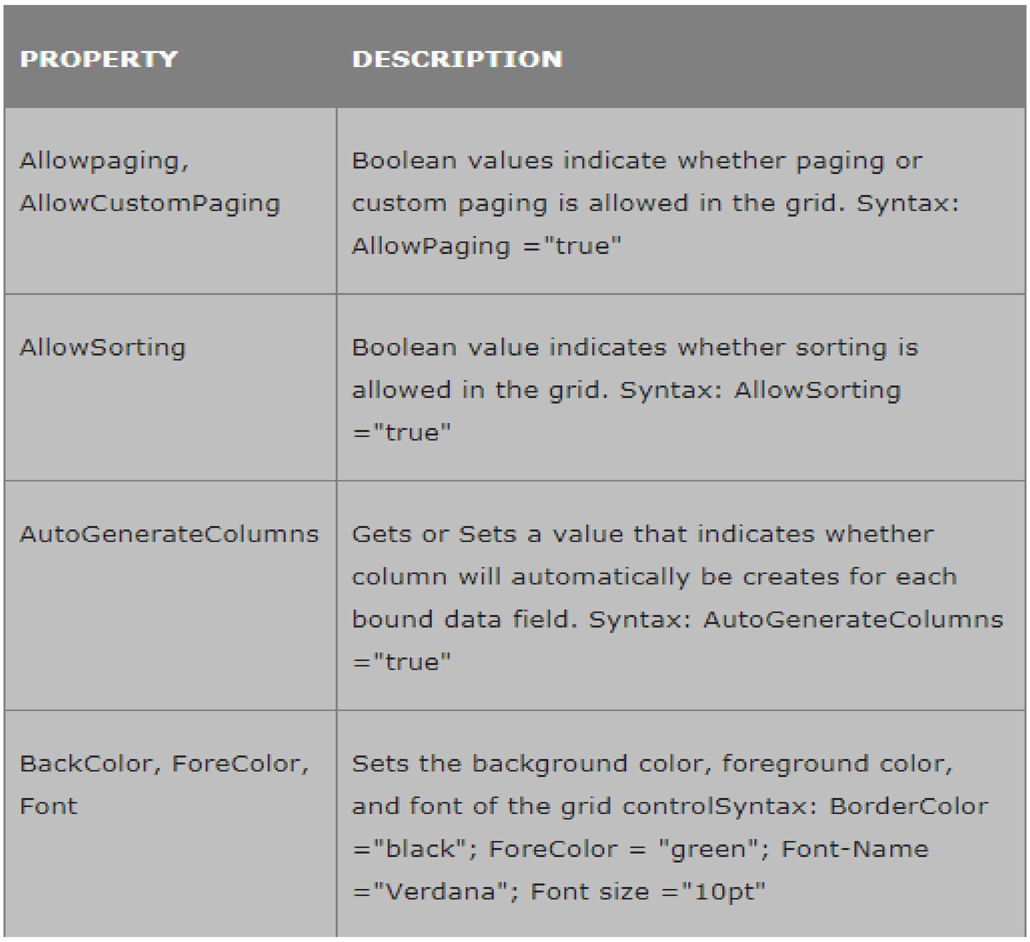
FieldCount	Gets the number of fields in the data reader.

Methods

GetFieldType(Int32)	Gets the **Type** information corresponding to the type of object that is returned from GetValue(Int32).
GetName(Int32)	Gets the name of the field to find.
GetOrdinal(String)	Return the index of the named field.
GetValue(Int32)	Return the value of the specified field.
Read()	Advances the IDataReader to the next record.

5.4 Programming with the DataList and DataGrid Controls

- DataGrid control is one of the most powerful Web controls. By using just a DataGrid control, you can write full-fledged Web applications for your Web site in this section, you'll learn all about the Data Grid methods and properties.
- The following examples are heavily based on this control.
- You can use a DataGrid control to display tabular data.
- It also provides the ability to insert, update, sort, and scroll the data. Using and binding a Data Grid is ability to insert, update, sort, and scroll the data.
- Using and binding a Data Grid is easy with the help of the ADO.NET datasets.
- This grid is capable of auto generating the columns and rows depending on which data you're connecting.
- Another control worth mentioning is the DataList control. A DataList control provides a list view of the data from the data source. These controls work similarly.
- DataGrid and DataList controls have similar properties. Table describes some of the common DataGrid properties.

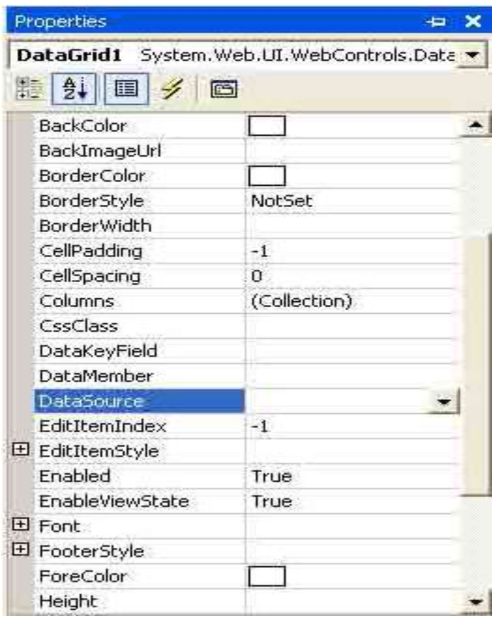


PROPERTY	DESCRIPTION
Allowpaging, AllowCustomPaging	Boolean values indicate whether paging or custom paging is allowed in the grid. Syntax: AllowPaging ="true"
AllowSorting	Boolean value indicates whether sorting is allowed in the grid. Syntax: AllowSorting ="true"
AutoGenerateColumns	Gets or Sets a value that indicates whether column will automatically be creates for each bound data field. Syntax: AutoGenerateColumns ="true"
BackColor, ForeColor, Font	Sets the background color, foreground color, and font of the grid controlSyntax: BorderColor ="black"; ForeColor = "green"; Font-Name ="Verdana"; Font size ="10pt"

CurrentPageIndex	Index of the currently displayed page.
DataKeyField	Primary Key field in the data source.
DataSource	Fills the grid with the data. Syntax: DataGrid1.DataSource = ds.Tables["Student"].DefaultView;
EditorItemIndex	Index of the item to be edited.
EditItemStyle	Style of the item to be edited.
HeaderStyle, FooterStyle	Header and footer styles. Syntax: HeaderStyleBackColor ="#00aaaa"; FooterStyleBackColor="#00aaaa"
GridLines	Gets or sets the grid line style. Syntax: Grid Lines ="vertical"

Setting DataGrid Control Properties at Design-Time

➢ In VS .NET you can set the DataGrid control's properties at design-time by right-clicking on the control and selecting the properties menu item. This displays the Properties window



DataGrid1 System.Web.UI.WebControls.Date

BackColor	☐
BackImageUrl	
BorderColor	☐
BorderStyle	NotSet
BorderWidth	
CellPadding	-1
CellSpacing	0
Columns	(Collection)
CssClass	
DataKeyField	
DataMember	
DataSource	
EditItemIndex	-1
⊞ EditItemStyle	
Enabled	True
EnableViewState	True
⊞ Font	
⊞ FooterStyle	
ForeColor	☐
Height	

➢ Most of these properties should look familiar to you by now.

- You can set the DataSource property to a data view, dataset, data view manager, or a data reader.
- The DataMember property represents a column of a data table if you want to display only one column in the control.

5.5 Working with the DataSet and DataTable Objects

- Creating a DataTable and adding it to a DataSet is pretty much the same as adding a DataColumn to a DataTable.
- You don’t have to create another project to show this feature, just comment what you have written inside your main method and paste the following code.

```
// Create a DataSet object
DataSet ds = new DataSet("MyDataSet");


// Add a DataTable named Table-1
DataTable table1 = ds.Tables.Add("Table-1");
// Here you can configure it, that is add some columns
// in the way we showed previously
DataTable table2 = new DataTable("Table-2");
ds.Tables.Add(table2);
DataTable table3 = new DataTable("Table-3");
DataTable table4 = new DataTable("Table-4");
// Configure those tables and then add them
// together
ds.Tables.AddRange(new DataTable[] { table3, table4 });
Console.WriteLine("DataSet {0} has the following {1} DataTables",
ds.DataSetName, ds.Tables.Count);
```

```
16.foreach(DataTable dt in ds.Tables)
17.Console.WriteLine("\t\t\t{0}", dt.TableName);
18.Console.ReadKey();
```

- When you create a DataSet instance, you can pass a string value as a parameter in order to name that DataSet (line 2).
- You can create and add a DataTable to a DataSet at the same time, using the **DataSet.Tables.Add** method (line 5).
- The returned table, of course can be configured later.
- You can add multiple tables to a DataSet using **DataSet.Tables.AddRange** fuction (line 16).
- Build and run your solution.

5.6 SQL transaction class

- Database transaction takes a database from one consistent state to another.
- At the end of the transaction the system must be in the prior state if the transaction fails or the status of the system should reflect the successful completion if the transaction goes through.
- It is a single unit of work.
- If a transaction is successful, all of the data modifications made during the transaction are committed and become a permanent part of the database.
- If a transaction encounters errors or gets rolled back, then all of the data modifications are erased.

Transaction commands

- The transaction commands are only used in SQL DML language like INSERT, UPDATE and DELETE, you cannot use it with DDL or DCL language as these DDL and DCL languages are used to in creating structure and SQL security.
- The transaction commands are given below,

- *COMMIT*

  This command is used to save the changes invoked by the transaction.

- *ROLLBACK*

  This command is used to undo the changes made by transaction.

- *SAVEPOINT*

  With the help of this command you can roll the transaction back to a certain point without rolling back the entire transaction.

- *SET TRANSACTION*

  This command is used to specify characteristics for the transaction. For example, you can specify a transaction to be read only, or read write it. Also helps set the name of transaction.

  - Example:

```
DECLARE @TranName VARCHAR(20);
SELECT @TranName = 'MyTransaction';


BEGIN TRANSACTION @TranName;
USE AdventureWorks2012;
DELETE FROM AdventureWorks2012.HumanResources.JobCandidate
   WHERE JobCandidateID = 13;


COMMIT TRANSACTION @TranName;
GO
```

Properties of SQL Transactions

There are 4 properties in SQL Server Transactions called as ACID.

- **Atomicity −** This property is used to ensure that all operations within the work unit are completed successfully. Otherwise, the dealing is aborted at the purpose of failure and the previous operations square measure rolled back to their former state.
- **Consistency –** This property is used to ensure that the database properly changes states upon a successfully committed transaction.
- **Isolation −** Permits transactions to operate independently of and transparent to each other.

- **Durability** − This property is used to ensure that the result or impact of a committed transaction persists in case of a system failure.

5.7 Working with Typed DataSet Objects.

- we can specify the Data type when we create a DataColumn for a DataTable.
- This is to enforce the runtime Type-safety for the column so that only data of specified data type can be stored in the column.
- In the same way, in most of the cases we prefer to make a DataSet itself as Type-safe so as to protect it from runtime mismatch. Hence Typed DataSets generate classes that expose each object the in the DataSet in Type-safe manner.
- These classes inherits directly from DataSet class.
- A DataColumn defines the column name and datatype. We can create a new DataColumn using the DataColumn constructor or by invoking the Add method of the DataTable.
- Columns collection property. The DataColumn is the fundamental building block for creating the schema of a DataTable. We build the schema by adding one or more DataColumn objects to the DataColumnCollection.
- For more information, see Adding Columns to a Table.
- Each DataColumn has a DataType property that determines the kind of data the DataColumn contains.

5.8 Security in asp.net application Authentication:

- It is the process of ensuring the user's identity and authenticity.
  ASP.NET allows four types of authentications:

1. Windows Authentication
2. Forms Authentication
3. Passport Authentication
4. Custom Authentication

Authorization:

It is the process of defining and allotting specific roles to specific users.

Confidentiality:

It involves encrypting the channel between the client browser and the web server.

Integrity:

It involves the integrity of data. For example,implementing digital
maintaining signature.

Forms-Based Authentication

Traditionally, forms-based authentication involves editing the web.config file and adding a login page with appropriate authentication code.

The web.config file could be edited and the following codes written on it:

```
<configuration>
<system.web>
<authentication mode="Forms">
<forms loginUrl ="login.aspx"/>
</authentication>

<authorization>
<deny users="?"/>
</authorization>
</system.web>
...
...
</configuration>
```

The login.aspx page mentioned in the above code snippet could have the following code behind file with the usernames and passwords for authentication hard coded into it.

```
protected bool authenticate(String uname, String pass)
{
  if(uname == "Tom")
  {
    if(pass == "tom123") return true;
  }

  if(uname == "Dick")
  {
    if(pass == "dick123")
      return true;
  }

  if(uname == "Harry")

  {
    if(pass == "har123") return true;
  }

  return false;
}

public void OnLogin(Object src, EventArgs e)
{
  if (authenticate(txtuser.Text, txtpwd.Text))
  {
    FormsAuthentication.RedirectFromLoginPage(txtuser.Text,
    chkrem.Checked);
  }
  else
  {
    Response.Write("Invalid user name or password");
  }
}
```

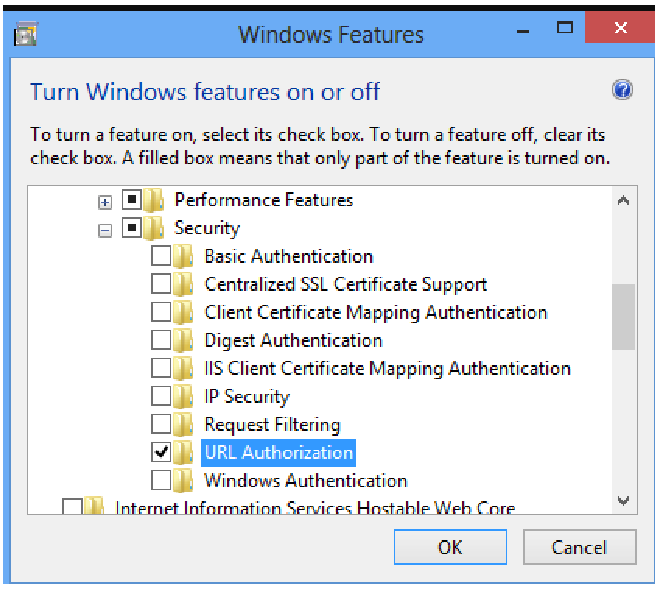
5.9 IIS Authentication and Authorization Security

- The <authorization> element allows you to configure the user accounts that can access your site or application.
- Use authorization in combination with authentication to secure access to content on your server.
- Authentication confirms the identity of a user, while authorization determines what resources users can or cannot access.
- IIS defines two types of authorization rules, Allow rules and Deny rules:

- Allow rules let you define the user accounts or user groups that can access a site, an application, or all the sites on a server.
- Deny rules let you define the user accounts or user groups that cannot access a site, an application, or all the sites on a server.

Windows 8 or Windows 8.1

1. On the **Start** screen, move the pointer all the way to the lower left corner, right-click the **Start** button, and then click **Control Panel**. - In **Control Panel**, click **Programs and Features**, and then click **Turn Windows features on or off**. - Expand **Internet Information Services**, expand **World Wide Web Services**, expand **Security**, and then select **URL Authorization**.



Windows Features

Turn Windows features on or off

To turn a feature on, select its check box. To turn a feature off, clear its check box. A filled box means that only part of the feature is turned on.

⊞ ■ Performance Features
⊟ ■ Security
☐ Basic Authentication
☐ Centralized SSL Certificate Support
☐ Client Certificate Mapping Authentication
☐ Digest Authentication
☐ IIS Client Certificate Mapping Authentication
☐ IP Security
☐ Request Filtering
☑ URL Authorization
☐ Windows Authentication
☐ Internet Information Services Hostable Web Core
OK
Cancel

2. Click Ok and the Close

To add an authorization rule

1. Open **Internet Information Services (IIS) Manager**:
   - If you are using Windows Server 2012 or Windows Server 2012 R2:
     - On the taskbar, click **Server Manager**, click **Tools**, and then click **Internet Information Services (IIS) Manager**.
   - If you are using Windows 8 or Windows 8.1:

        - Hold down the **Windows** key, press the letter **X**, and then click **Control Panel**.
        - Click **Administrative Tools**, and then double-click **Internet Information Services (IIS) Manager**.
    - If you are using Windows Server 2008 or Windows Server 2008 R2:
        - On the taskbar, click **Start**, point to **Administrative Tools**, and then click **Internet Information Services (IIS) Manager**.
    - If you are using Windows Vista or Windows 7:
        - On the taskbar, click **Start**, and then click **Control Panel**.
        - Double-click **Administrative Tools**, and then double-click **Internet Information Services (IIS) Manager**.
2. In the **Connections** pane, expand the server name, expand **Sites**, and then navigate to the site or application on which you want to configure authorization.
3. In the **Home** pane, double-click **Authorization Rules**.
4. To add a new authorization rule, in the **Actions** pane click **Add Allow Rule...** or **Add Deny Rule...**
5. Apply the authorization settings needed for your site or application, and then click **OK**.

5.10 ASP.NET Authentications Security

- ➢ In this series of articles, we'll look at some options for securing a web API from unauthorized users.
- ➢ This series will cover both authentication and authorization.

- *Authentication* is knowing the identity of the user. For example, Alice logs in with her username and password, and the server uses the password to authenticate Alice.

- *Authorization* is deciding whether a user is allowed to perform an action. For example, Alice has permission to get a resource but not create a resource.

➢ The first article in the series gives a general overview of authentication and authorization in ASP.NET Web API. Other topics describe common authentication scenarios for Web API.

Authentication

➢ Web API assumes that authentication happens in the host. For web-hosting, the host is IIS, which uses HTTP modules for authentication. You can configure your project to use any of the authentication modules built in to IIS or ASP.NET, or write your own HTTP module to perform custom authentication.

➢ When the host authenticates the user, it creates a *principal*, which is an IPrincipal object that represents the security context under which code is running. The host attaches the principal to the current thread by setting **Thread.CurrentPrincipal**.

HTTP Message Handlers for Authentication

➢ Instead of using the host for authentication, you can put authentication logic into an HTTP message handler. In that case, the message handler examines the HTTP request and sets the principal.

➢ When should you use message handlers for authentication? Here are some tradeoffs:

- An HTTP module sees all requests that go through the ASP.NET pipeline. A message handler only sees requests that are routed to Web API.

- You can set per-route message handlers, which lets you apply an authentication scheme to a specific route.
- HTTP modules are specific to IIS. Message handlers are host-agnostic, so they can be used with both web-hosting and self-hosting.
- HTTP modules participate in IIS logging, auditing, and so on.
- HTTP modules run earlier in the pipeline. If you handle authentication in a message handler, the principal does not get set until the handler runs. Moreover, the principal reverts back to the previous principal when the response leaves the message handler.

Authorization

- ➢ Authorization happens later in the pipeline, closer to the controller. That lets you make more granular choices when you grant access to resources.

- *Authorization filters* run before the controller action. If the request is not authorized, the filter returns an error response, and the action is not invoked.
- Within a controller action, you can get the current principal from the **ApiController.User** property. For example, you might filter a list of resources based on the user name, returning only those resources that belong to that user.
  - This filter checks whether the user is authenticated. If not, it returns HTTP status code 401 (Unauthorized), without invoking the action.
  - You can apply the filter globally, at the controller level, or at the level of individual actions.