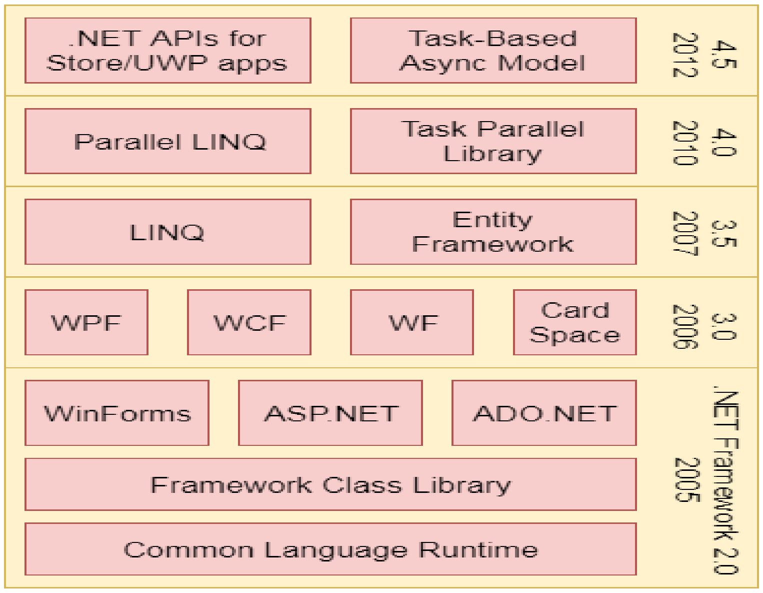
Dot Net

- Dot net is not a language.
- It is a platform or collection of different programming support.
- That helps us to develop rich websites, applications and other computer devices and also to make the activities more of a web browser-oriented.

.NET Framework

- .NET is a framework which is used to develop software applications.
- It is designed and developed by Microsoft and first beta version released on 2000.
- It is used to build applications for web, Windows, phone and provides broad range of functionalities and support for industry standards.
- This framework contains large number of class libraries known as Framework Class Library (FCL).
- The software programs written in .NET are execute in execution environment that is called CLR (Common Language Runtime).
- These both are core and essential parts of the .NET Framework.
- This Framework provides various services like: memory management, networking, security and memory safety. It also supports numerous programing languages like: C#, F#, VB etc.

Following is the .NET framework Stack that shows the modules and components of the Framework.



CLR (Common Language Runtime)

- It is a program execution engine that loads and execute program.
- It acts as a interface between framework and operating system.

FCL (Framework Class Library)

- It is a standard library that is collection of thousands of classes and used to build application.
- The BCL (Base Class Library) is the core of the FCL and provides fundamental functionalities.

WinForms

- Windows Forms is a smart client technology for the .NET Framework, a set of managed libraries that simplify common application tasks such as reading and writing to the file system.

ASP .NET

- ASP .NET is a web framework designed and developed by Microsoft.
- It is used to develop websites, web applications and web services.
- It provides fantastic integration of HTML, CSS and JavaScript.
- It was first released in January 2002.

ADO .NET

- ADO .NET is a module of .Net Framework which is used to establish connection between application and data sources.
- Data sources can be such as SQL Server and XML. ADO .NET consists of classes that can be used to connect, retrieve, insert and delete data.

WPF (Windows Presentation Foundation)

- Windows Presentation Foundation (WPF) is a graphical subsystem by Microsoft for rendering user interfaces in Windows-based applications.

WCF (Windows Communication Foundation)

- It is a framework for building service-oriented applications. Using WCF, you can send data as asynchronous messages from one service endpoint to another.

WF (WorkFlow Foundation)

- Windows Workflow Foundation (WF) is a Microsoft technology that provides an API, an in-process workflow engine, and a rehostable designer to implement long-running processes as workflows within .NET applications.

LINQ (Language Integrated Query)

- It is a query language, introduced in .NET 3.5 framework.
- It is used to make query for data sources with C# or Visual Basics programming languages.

Entity Framework

- It is an ORM based open source framework that is used to work with a database using .NET objects.
- It eliminates lots of developer effort to handle the database.
- It is Microsoft's recommended technology to deal with database.

Parallel LINQ

- Parallel LINQ or PLINQ is a parallel implementation of LINQ to objects.
- It combines the simplicity and readability of LINQ and provide power of parallel programing.
- It can improve and provide fast speed to execute the LINQ query by using all available computer capabilities.

.Benefits of .NET

- It allows the use of multiple languages
- It has horizontal scalability
- .NET creates a unified environment that allows developers to create programs in C++, Java or Virtual Basic
- Interfaces easily with Windows or Microsoft
- All tools and IDEs have been pre-tested and are easily available in the Microsoft Developer Network.
- UI best practices are more consistent
- Language integration is seamless, as you can call methods from C# to VB.NET

.NET Common Language Runtime (CLR)

- .NET CLR is a run-time environment.
- That manages and executes the code written in any .NET programming language.
- It converts code into native code which further can be executed by the CPU.

.NET CLR Functions

Following are the functions of the CLR.

- It converts program into native code.
- Handles Exceptions
- Provides type safety
- Memory management
- Provides security
- Improved performance
- Language independent
- Platform independent
- Garbage collection
- Provides language features such as inheritance, interfaces, and overloading for object-oriented programming.

Features of CLR

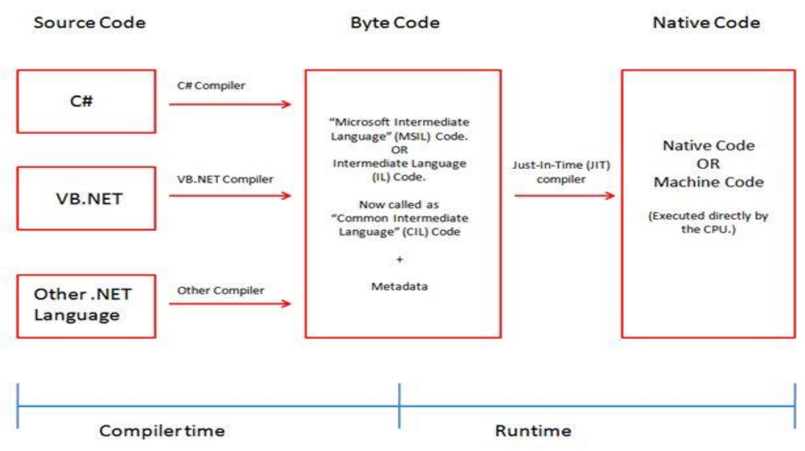
Following is the component features of Common language Runtime.

1. **Base Class Library Support**
   It is a class library that provides support of classes to the .NET application.
2. **Thread Support**
   It manages parallel execution of the multi-threaded application.
3. **COM Marshaler**
   It provides the communication between the COM objects and the application.
4. **Type Checker**
   It checks types used in the application and verify that they match to the standard provided by the CLR.
5. **Code Manager**
   It manages code at execution run-time.
6. **Garbage Collector**
   It releases unused memory and allocate that to a new application.
7. **Exception Handler**
   It handles exception at runtime to avoid application failure.
8. **Class Loader**
   It is used to load all classes at run time.

Compilation and MSIL

The Code Execution Process involves the following two stages:

1. Compiler time process.
2. Runtime process.


Source Code
Byte Code
Native Code
C#
C# Compiler
VB.NET
VB.NET Compiler
Other .NET Language
Other Compiler
"Microsoft Intermediate Language" (MSIL) Code. OR Intermediate Language (IL) Code. Now called as "Common Intermediate Language" (CIL) Code + Metadata
Just-In-Time (JIT) compiler
Native Code OR Machine Code (Executed directly by the CPU.)
Compilertime
Runtime

1. Compiler time process

1. The .Net framework has one or more language compliers, such as Visual Basic, C#, Visual C++, JScript, or one of many third-party compilers.
2. Any one of the compilers translate your source code into Microsoft Intermediate Language (MSIL) code.
3. For example, if you are using the C# programming language to develop an application, when you compile the application, the C# language compiler will convert your source code into Microsoft Intermediate Language (MSIL) code.
4. In short, VB.NET, C# and other language compilers generate MSIL code.
5. Currently "Microsoft Intermediate Language" (MSIL) code is also known as "Intermediate Language" (IL) Code**or** "Common Intermediate Language" (CIL) Code.

   SOURCE CODE -----.NET COMLIPER------> BYTE CODE (MSIL + META DATA)

2. Runtime process.

1. The Common Language Runtime (CLR) includes a JIT compiler for converting MSIL to native code.
2. The JIT Compiler in CLR converts the MSIL code into native machine code that is then executed by the OS.
3. During the runtime of a program the "Just in Time" (JIT) compiler of the Common Language Runtime (CLR) uses the Metadata and converts Microsoft Intermediate Language (MSIL) into native code.

   BYTE CODE (MSIL + META DATA) ----- Just-In-Time (JIT) compiler------> NATIVE CODE

Microsoft Intermediate Language (MSIL)

- A .NET programming language (C#, VB.NET etc.) does not compile into executable code; instead it compiles into an intermediate code called Microsoft Intermediate Language (MSIL).

- As a programmer one need not worry about the syntax of MSIL - since our source code in automatically converted to MSIL.

- The MSIL code is then send to the CLR (Common Language Runtime) that converts the code to machine language which is then run on the host machine.

.NET Framework Class Library

.NET Framework Class Library is the collection of classes, namespaces, interfaces and value types that are used for .NET applications.
It contains thousands of classes that supports the following functions.

- Base and user-defined data types
- Support for exceptions handling
- input/output and stream operations
- Communications with the underlying system
- Access to data
- Ability to create Windows-based GUI applications
- Ability to create web-client and server applications
- Support for creating web services

.NET Framework Class Library Namespaces

Following are the commonly used namespaces that contains useful classes and interfaces and defined in Framework Class Library.

Namespaces	Description
System	It includes all common datatypes, string values, arrays and methods for data conversion.
System.Data, System.Data.Common, System.Data.OleDb, System.Data.SqlClient, System.Data.SqlTypes	These are used to access a database, perform commands on a database and retrieve database.
System.IO, System.DirectoryServices, System.IO.IsolatedStorage	These are used to access, read and write files.
System.Diagnostics	It is used to debug and trace the execution of an application.
System.Net, System.Net.Sockets	These are used to communicate over the Internet when creating peer-to-peer applications.
System.Windows.Forms, System.Windows.Forms.Design	These namespaces are used to create Windows-based applications using Windows user interface components.
System.Web, System.WebCaching, System.Web.UI, System.Web.UI.Design, System.Web.UI.WebControls, System.Web.UI.HtmlControls, System.Web.Configuration,	These are used to create ASP. NET Web applications that run over

System.Web.Hosting, System.Web.Mail, System.Web.SessionState	the web.
System.Web.Services, System.Web.Services.Description, System.Web.Services.Configuration, System.Web.Services.Discovery, System.Web.Services.Protocols	These are used to create XML Web services and components that can be published over the web.
System.Security, System.Security.Permissions, System.Security.Policy, System.WebSecurity, System.Security.Cryptography	These are used for authentication, authorization, and encryption purpose.
System.Xml, System.Xml.Schema, System.Xml.Serialization, System.Xml.XPath, System.Xml.Xsl	These namespaces are used to create and access XML files.

Visual Studio IDE

- ➢ An integrated development environment (IDE), also known as integrated design environment and integrated debugging environment.
- ➢ It is a type of computer software that assists computer programmers to develop software. In the case of Visual Basic .NET, that IDE is Visual Studio.

IDE Contents

- ➢ The Visual Studio IDE consists of several sections or tools.
- ➢ That the developer uses while programming. As you view the IDE for a new project you generally have three sections in view:

1. The Toolbox on the left
2.The Solution Explorer on the right
3.The Code / Design view in the middle

Toolbox

- ➢ The Toolbox is a palette of developer objects, or controls.
- ➢ That are placed on forms or web pages, and then code is added to allow the user to interact with them.
- ➢ An example would be Textbox, Button and ListBox controls. With these three controls added to a Windows Form object the developer could write code that would take text, input by the application user, and added to the ListBox after the button was clicked.

Solution Explorer

- ➢ This is a section that is used to view and modify the contents of the project.
- ➢ A Visual Studio Windows Application Project will generally have a Form object with a code page, references to System components and possibly other modules with special code that is used by the application.

Properties Windows

- ➢ The properties windows shows all the control (like textbox) properties to be change at design time.
- ➢ The Most of this property can be change at run time with some code, but basically most of this properties change the way the control is display on your application.

Code / Design view

- ➢ This is where the magic takes place. Forms are designed graphically.
- ➢ In other words, the developer has a form on the screen that can be sized and modified to look the way it will be displayed to the application users.
- ➢ Controls are added to the form from the Toolbox, the color and caption can be changed along with many other items.
- ➢ This center window of the IDE is also where developers write the code that makes everything in the application work.
- ➢ The code is written in modules, or files, that are either connected to an object (Forms) or called specifically when needed.

Basic elements of C#

- ➢ C# is pronounced as "C-Sharp".
- ➢ **Anders Hejlsberg** is known as the **founder of C# language.**
- ➢ It is based on **C++ and Java**, but it has many additional extensions used to perform component oriented programming approach.
- ➢ C# has evolved much since their first release in the year **2002**.
- ➢ It is an object-oriented programming language provided by Microsoft.
- ➢ That runs on .Net Framework.

The C# programming develop different types of secured and robust applications:

- o Window applications
- o Web applications
- o Web service applications
- o Database applications etc.

C# Variable

- ➢ A variable is a name of memory location.
- ➢ It is used to store data. Its value can be changed and it can be reused many times.
- ➢ It is a way to represent memory location through symbol so that it can be easily identified.
- ➢ The basic variable type available in C# can be categorized as:

Variable Type	Example
Decimal types	decimal
Boolean types	True or false value, as assigned
Integral types	int, char, byte, short, long
Floating point types	float and double
Nullable types	Nullable data types

Rules for defining variables

- A variable can have alphabets, digits and underscore.
- A variable name can start with alphabet and underscore only. It can't start with digit.
- No white space is allowed within variable name.
- A variable name must not be any reserved word or keyword e.g. chars, float etc.

Valid variable names:

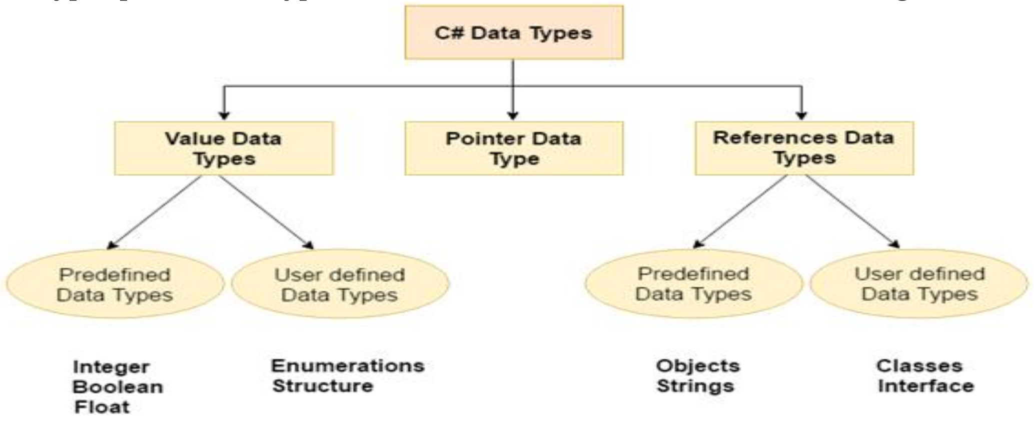
1. int x;
2. int _x;
3. int k20;

Invalid variable names:

1. int 4;
2. int x y;
3. int double;

C# Data Types

A data type specifies the type of data that a variable can store such as integer, floating, character etc.



There are 3 types of data types in C# language.

Types	Data Types
Value Data Type	short, int, char, float, double etc
Reference Data Type	String, Class, Object and Interface
Pointer Data Type	Pointers

Value Data Type

- The value data types are integer-based and floating-point based.
- C# language supports both signed and unsigned literals.

There are 2 types of value data type in C# language.

1) Predefined Data Types - such as Integer, Boolean, Float, etc.

2) User defined Data Types - such as Structure, Enumerations, etc.

Data Types	Memory Size	Range
Char	1 byte	-128 to 127
signed char	1 byte	-128 to 127

unsigned char	1 byte	0 to 127
short	2 byte	-32,768 to 32,767
signed short	2 byte	-32,768 to 32,767
unsigned short	2 byte	0 to 65,535
int	4 byte	-2,147,483,648 to -2,147,483,647
signed int	4 byte	-2,147,483,648 to -2,147,483,647
unsigned int	4 byte	0 to 4,294,967,295
long	8 byte	?9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
signed long	8 byte	?9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
unsigned long	8 byte	0 - 18,446,744,073,709,551,615
float	4 byte	1.5 * 10^{-45} - 3.4 * 10^{38}, 7-digit precision
double	8 byte	5.0 * 10^{-324} - 1.7 * 10^{308}, 15-digit precision
decimal	16 byte	at least -7.9 * $10^{?28}$ - 7.9 * 10^{28}, with at least 28-digit precision

Reference Data Type

- ➢ The reference data types do not contain the actual data stored in a variable, but they contain reference to the variables.
- ➢ If the data is changed by one of the variables, the other variable automatically reflects this change in value.

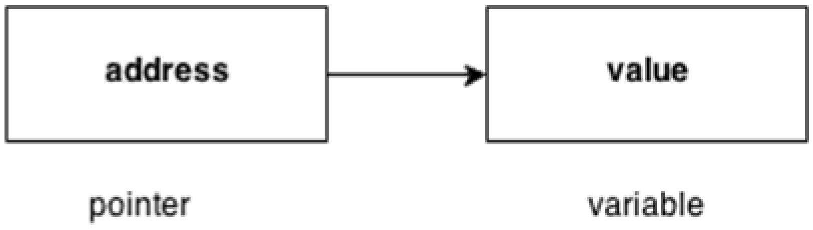
There are 2 types of reference data type in C# language.

1) Predefined Types - such as Objects, String.

2) User defined Types - such as Classes, Interface.

Pointer Data Type

The pointer in C# language is a variable, it is also known as locator or indicator that points to an address of a value.



Symbols used in pointer

Symbol	Name	Description
& (ampersand sign)	Address operator	Determine the address of a variable.
* (asterisk sign)	Indirection operator	Access the value of an address.

Declaring a pointer

The pointer in C# language can be declared using * (asterisk symbol).

```
1. int * a;  //pointer to int
2. char * c; //pointer to char
```

C# Keywords

- A keyword is a reserved word.
- It cannot use it as a variable name, constant name etc.
- In C# keywords cannot be used as identifiers. However, if we want to use the keywords as identifiers, we may prefix the keyword with @ character.

abstract	Base	As	Bool	break	Catch	case
Byte	Char	Checked	class	const	continue	decimal
private	protected	Public	return	readonly	Ref	sbyte
explicit	extern	False	finally	fixed	Float	for
foreach	Goto	If	implicit	in	in (generic modifier)	int
ulong	ushort	unchecked	using	unsafe	Virtual	void
Null	object	Operator	out	out (generic modifier)	override	params
default	delegate	Do	double	else	Enum	event
sealed	short	Sizeof	stackalloc	static	String	struct
switch	This	Throw	true	try	Typeof	uint
abstract	Base	As	bool	break	Catch	case
volatile	while					

Program structures sample input and output operations

C# Example: Hello World

In C# programming language, a simple "hello world" program can be written by multiple ways. Let's see the top 4 ways to create a simple C# example:

- Simple Example
- Using System
- Using public modifier
- Using namespace

C# Simple Example

```
class Program
  {
    static void Main(string[] args)
    {
      System.Console.WriteLine("Hello World!");
    }
  }
```

Output:
Hello World!

Description

class: is a keyword which is used to define class.
Program: is the class name. A class is a blueprint or template from which objects are created. It can have data members and methods. Here, it has only Main method.
static: is a keyword which means object is not required to access static members. So it saves memory.
void: is the return type of the method. It does't return any value. In such case, return statement is not required.
Main: is the method name. It is the entry point for any C# program. Whenever we run the C# program, Main() method is invoked first before any other method. It represents start up of the program.
string[] args: is used for command line arguments in C#. While running the C# program, we can pass values. These values are known as arguments which we can use in the program.
System.Console.WriteLine("Hello World!"): Here, System is the namespace. Console is the class defined in System namespace. The WriteLine() is the static method of Console class which is used to write the text on the console.

C# - Operators and Expression

- An operator is a symbol.
- That tells the compiler to perform specific mathematical or logical manipulations.
- C# has rich set of built-in operators and provides the following type of operators
  1. Arithmetic Operators
  2. Relational Operators
  3. Logical Operators
  4. Bitwise and bit shift operators
  5. Assignment Operators

Arithmetic Operators:

Arithmetic operators are used to perform arithmetic operations such as addition, subtraction, multiplication, division, etc.

Operator	Operator Name	Example
+	Addition Operator	6 + 3 evaluates to 9

Operator	Operator Name	Example
-	Subtraction Operator	10 - 6 evaluates to 4
*	Multiplication Operator	4 * 2 evaluates to 8
/	Division Operator	10 / 5 evaluates to 2
%	Modulo Operator (Remainder)	16 % 3 evaluates to 1

Example :

```
using System;
namespace Operator  {
class ArithmeticOperator  {
            public static void Main(string[] args) {
            double firstNumber = 14.40, secondNumber = 4.60, result;
            int num1 = 26, num2 = 4, rem;
// Addition operator
        result = firstNumber + secondNumber;
        Console.WriteLine("{0} + {1} = {2}", firstNumber, secondNumber, result);
// Subtraction operator
          result = firstNumber - secondNumber;
         Console.WriteLine("{0} - {1} = {2}", firstNumber, secondNumber, result);
// Multiplication operator
           result = firstNumber * secondNumber;
          Console.WriteLine("{0} * {1} = {2}", firstNumber, secondNumber, result);

// Division operator
            result = firstNumber / secondNumber;
            Console.WriteLine("{0} / {1} = {2}", firstNumber, secondNumber, result);
// Modulo operator
            rem = num1 % num2;
            Console.WriteLine("{0} % {1} = {2}", num1, num2, rem);
                        }
}
}
```

Output:

```
14.4 + 4.6 = 19
14.4 - 4.6 = 9.8
14.4 * 4.6 = 66.24
14.4 / 4.6 = 3.1304347826087
26 % 4 = 2
```

Relational Operators

- Relational operators are used to check the relationship between two operands.
- If the relationship is true the result will be true, otherwise it will result in false.
- Relational operators are used in decision making and loops.

Operator	Operator Name	Example
==	Equal to	6 == 4 evaluates to false
>	Greater than	3 > -1 evaluates to true
<	Less than	5 < 3 evaluates to false
>=	Greater than or equal to	4 >= 4 evaluates to true
<=	Less than or equal to	5 <= 3 evaluates to false
!=	Not equal to	10 != 2 evaluates to true

Example :

```
using System;
 namespace Operator {
   class RelationalOperator {
          public static void Main(string[] args){
                         bool result;
                         int firstNumber = 10, secondNumber = 20;

result = (firstNumber==secondNumber);
Console.WriteLine("{0} == {1} returns {2}",firstNumber, secondNumber, result);

result = (firstNumber > secondNumber);
Console.WriteLine("{0} > {1} returns {2}",firstNumber, secondNumber, result);

result = (firstNumber < secondNumber);
Console.WriteLine("{0} < {1} returns {2}",firstNumber, secondNumber, result);

result = (firstNumber >= secondNumber);
Console.WriteLine("{0} >= {1} returns {2}",firstNumber, secondNumber, result);

result = (firstNumber <= secondNumber);
Console.WriteLine("{0} <= {1} returns {2}",firstNumber, secondNumber, result);

result = (firstNumber != secondNumber);
Console.WriteLine("{0} != {1} returns {2}",firstNumber, secondNumber, result);
                }
      }   }
```

Output:

10 == 20 returns False
10 > 20 returns False
10 < 20 returns True
10 >= 20 returns False
10 <= 20 returns True
10 != 20 returns True

Logical Operators:

- Logical operators are used to perform logical operation such as and, or.
- Logical operators operates on boolean expressions (true and false) and returns boolean values.
- Logical operators are used in decision making and loops.

Operand 1	Operand 2	OR (||)	AND (&&)
True	True	true	True
True	False	true	false
False	True	true	false
False	False	false	false

Example :

```
using System;
namespace Operator {
  class LogicalOperator {
          public static void Main(string[] args){
               bool result;
               int firstNumber = 10, secondNumber = 20;

// OR operator
          result = (firstNumber == secondNumber) || (firstNumber > 5);
          Console.WriteLine(result);
// AND operator
          result = (firstNumber == secondNumber) && (firstNumber > 5);
          Console.WriteLine(result);
                  }
          }
}
```

Output:

```
True
False
```

Bitwise and Bit Shift Operator:

Bitwise and bit shift operators are used to perform bit manipulation operations.

Operator	Operator Name
~	Bitwise Complement
&	Bitwise AND
|	Bitwise OR
^	Bitwise Exclusive OR
<<	Bitwise Left Shift
>>	Bitwise Right Shift

Example :

```
using System;
 namespace Operator {
   class BitOperator {
        public static void Main(string[] args){
                        int firstNumber = 10;
                        int secondNumber = 20;
                        int result;

        result = ~firstNumber;
        Console.WriteLine("~{0} = {1}", firstNumber, result);

         result = firstNumber & secondNumber;
        Console.WriteLine("{0} & {1} = {2}", firstNumber,secondNumber, result);

        result = firstNumber | secondNumber;
        Console.WriteLine("{0} | {1} = {2}", firstNumber,secondNumber, result);

        result = firstNumber ^ secondNumber;
        Console.WriteLine("{0} ^ {1} = {2}", firstNumber,secondNumber, result);

        result = firstNumber << 2;
        Console.WriteLine("{0} << 2 = {1}", firstNumber, result);

        result = firstNumber >> 2;
        Console.WriteLine("{0} >> 2 = {1}", firstNumber, result);
                }
        }
}
```

Output :

```
~10 = -11
10 & 20 = 0
10 | 20 = 30
10 ^ 20 = 30
10 << 2 = 40
10 >> 2 = 2
```

Assignment Operators

There are following assignment operators supported by C#

Operator	Operator Name	Example	Equivalent To
+=	Addition Assignment	x += 5	x = x + 5
-=	Subtraction Assignment	x -= 5	x = x - 5
*=	Multiplication Assignment	x *= 5	x = x * 5
/=	Division Assignment	x /= 5	x = x / 5
%=	Modulo Assignment	x %= 5	x = x % 5
&=	Bitwise AND Assignment	x &= 5	x = x & 5
|=	Bitwise OR Assignment	x |= 5	x = x | 5
^=	Bitwise XOR Assignment	x ^= 5	x = x ^ 5
<<=	Left Shift Assignment	x <<= 5	x = x << 5
>>=	Right Shift Assignment	x >>= 5	x = x >> 5
=>	Lambda Operator	x => x*x	Returns x*x

Example :

```
using System;

namespace Operator
{
	class BitOperator
	{
		public static void Main(string[] args)
```

```
                    {
                              int number = 10;

                              number += 5;
                              Console.WriteLine(number);

                              number -= 3;
                              Console.WriteLine(number);

                              number *= 2;
                              Console.WriteLine(number);

                              number /= 3;
                              Console.WriteLine(number);

                              number %= 3;
                              Console.WriteLine(number);

                              number &= 10;
                              Console.WriteLine(number);

                              number |= 14;
                              Console.WriteLine(number);

                              number ^= 12;
                              Console.WriteLine(number);

                              number <<= 2;
                              Console.WriteLine(number);
number >>= 3;
                              Console.WriteLine(number);
                    }
          }
}
```

Output:

```
15
12
24
8
2
2
14
2
8
1
```

Control Statements

C# offers three types of control statements:

1. Selection Statements.
2. Iteration Statements.
3. Jump Statements

1. Selection statements

C# if-else statement:

In C# programming, the *if statement* is used to test the condition.
There are various types of if statements in C#.

if statement
if-else statement
nested if statement
if-else-if ladder

IF Statement

- The if statement tests the condition.
- It is executed if condition is true.

Syntax:

```
if(condition){
//code to be executed
}
```

Example:

```
using System;
public class IfExample
   {
      public static void Main(string[] args)
       {
          int num = 10;
          if (num % 2 == 0)
          {
             Console.WriteLine("It is even number");
          }

       }
   }
```

Output:

```
It is even number
```

IF-else Statement

- The C# if-else statement also tests the condition.
- It executes the *if block* if condition is true otherwise *else block* is executed.

Syntax:

```
if(condition){
//code if condition is true
}else{
//code if condition is false
}
```

Example:

```
using System;
public class IfExample
    {
        public static void Main(string[] args)
        {
            int num = 11;
            if (num % 2 == 0)
            {
                Console.WriteLine("It is even number");
            }
            else
            {
                Console.WriteLine("It is odd number");
            }

        }
    }
```

Output:

```
It is odd number
```

IF-else-if ladder Statement

- ➢ The C# if-else-if ladder statement executes one condition from multiple statements.

Syntax:

```
if(condition1){
//code to be executed if condition1 is true
}else if(condition2){
//code to be executed if condition2 is true
}
else if(condition3){
//code to be executed if condition3 is true
}
...
else{
//code to be executed if all the conditions are false
}
```

Example:

```
using System;
public class IfExample
   {
      public static void Main(string[] args)
      {
         Console.WriteLine("Enter a number to check grade:");
         int num = Convert.ToInt32(Console.ReadLine());

         if (num <0 || num >100)
         {
            Console.WriteLine("wrong number");
         }
         else if(num >= 0 && num < 50){
            Console.WriteLine("Fail");
         }
         else if (num >= 50 && num < 60)
         {
            Console.WriteLine("D Grade");
         }
         else if (num >= 60 && num < 70)
         {
            Console.WriteLine("C Grade");
         }
         else if (num >= 70 && num < 80)
         {
            Console.WriteLine("B Grade");
         }
         else if (num >= 80 && num < 90)
         {
            Console.WriteLine("A Grade");
         }
         else if (num >= 90 && num <= 100)
         {
            Console.WriteLine("A+ Grade");
         }
      }
   }
```

Output:

```
Enter a number to check grade:66
C Grade
```

Output:

```
Enter a number to check grade:-2
wrong number
```

C# switch

- The *switch statement* executes one statement from multiple conditions.
- It is like if-else-if ladder statement in C#.

Syntax:

```
switch(expression){
case value1:
 //code to be executed;
 break;
case value2:
 //code to be executed;
 break;
......

default:
 //code to be executed if all cases are not matched;
 break;
}
```

Example:

```
using System;
  public class SwitchExample
    {
     public static void Main(string[] args)
     {
         Console.WriteLine("Enter a number:");
         int num = Convert.ToInt32(Console.ReadLine());

         switch (num)
         {
             case 10: Console.WriteLine("It is 10")
             break;
             case 20: Console.WriteLine("It is 20");
             break;
             case 30: Console.WriteLine("It is 30");
             break;
             default: Console.WriteLine("Not 10, 20 or 30");
              break;
         }
     }
    }
```

Output:

```
Enter a number:
10
It is 10
```

Output:

```
Enter a number:
55
Not 10, 20 or 30
```

2. Iteration Statements

For Loop

- The *for loop* is used to iterate a part of the program several times.
- If the number of iteration is fixed, it is recommended to use for loop than while or do-while loops.
- The for loop is same as C/C++
- . We can initialize variable, check condition and increment/decrement value.

Syntax:

```
for(initialization; condition; incr/decr){
//code to be executed
}
```

Example:

```
using System;
public class ForExample
  {
    public static void Main(string[] args)
    {
        for(int i=1;i<=5;i++){
          Console.WriteLine(i);
        }
    }
  }
```

Output:

```
1
2
3
4
5
```

Nested For Loop

In C#, we can use for loop inside another for loop, it is known as nested for loop.

Example:

```
using System;
public class ForExample
  {
    public static void Main(string[] args)
    {
      for(int i=1;i<=3;i++){
          for(int j=1;j<=3;j++){
            Console.WriteLine(i+" "+j);
          }
      }
    }
  }
```

Output:

```
1 1
1 2
```

```
1 3
2 1
2 2
2 3
3 1
3 2
3 3
```

While Loop

- The while loop is a entry control loop.
- In while *loop* is used to iterate a part of the program several times.
- If the number of iteration is not fixed, it is recommended to use while loop than for loop.

Syntax:

```
while(condition){
//code to be executed
}
```

Example:

```
using System;
public class WhileExample
  {
    public static void Main(string[] args)
    {
       int i=1;
       while(i<=5)
       {
          Console.WriteLine(i);
          i++;
       }
   }
 }
```

Output:

```
1
2
3
4
5
```

Do-While Loop

- The do-while loop is a exit control loop.
- The do-*while loop* is used to iterate a part of the program several times.
- If the number of iteration is not fixed and you must have to execute the loop at least once, it is recommended to use do-while loop.
- The *do-while loop* is executed at least once because condition is checked after loop body.

Syntax:

```
do{
//code to be executed
}while(condition);
```

Example:

```
using System;
public class DoWhileExample
  {
    public static void Main(string[] args)
    {
      int i = 1;

      do{
        Console.WriteLine(i);
        i++;
      } while (i <= 5) ;

    }
  }
```

Output:

```
1
2
3
4
5
```

3.Jump Statements:

Break Statement

The *break* is used to break loop or switch statement.
It breaks the current flow of the program at the given condition.
In case of inner loop, it breaks only inner loop.

Syntax:

```
jump-statement;
break;
```

Example:

```
using System;
public class BreakExample
  {
    public static void Main(string[] args)
    {
      for (int i = 1; i <= 10; i++)
      {
        if (i == 5)
        {
          break;
        }
        Console.WriteLine(i);
      }
    }
  }
```

Output:

```
1
2
3
4
```

Continue Statement

- The *continue statement* is used to continue loop.
- It continues the current flow of the program and skips the remaining code at specified condition.
- In case of inner loop, it continues only inner loop.

Syntax:

```
jump-statement;
continue;
```

Example:

```
using System;
public class ContinueExample
   {
     public static void Main(string[] args)
     {
        for(int i=1;i<=6;i++){
          if(i==5){
              continue;
          }
          Console.WriteLine(i);
        }
     }
  }
```

Output:

```
1
2
3
4
6
```

Goto Statement

The goto statement is also known jump statement.
It is used to transfer control to the other part of the program.
It unconditionally jumps to the specified label.
It can be used to transfer control from deeply nested loop or switch case label.
Currently, it is avoided to use goto statement in C# because it makes the program complex.

Example

```
using System;
public class GotoExample
   {
     public static void Main(string[] args)
     {
     ineligible:
         Console.WriteLine("You are not eligible to vote!");
```

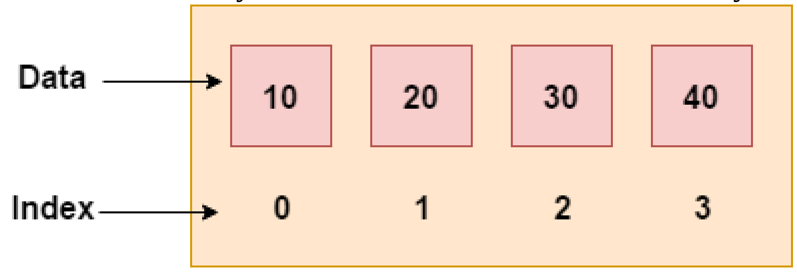
```

        Console.WriteLine("Enter your age:\n");
        int age = Convert.ToInt32(Console.ReadLine());
        if (age < 18){
                goto ineligible;
        }
        else
        {
                Console.WriteLine("You are eligible to vote!");
        }
        }
   }
```

Output:

```
You are not eligible to vote!
Enter your age:
11
You are not eligible to vote!
Enter your age:
5
You are not eligible to vote!
Enter your age:
26
You are eligible to vote!
```

C# Arrays

- The array is a group of similar types of elements.
- That have a contiguous set of memory location.
- Array is an *object* of base type **System.Array**.
- In C#, array index starts from 0. We can store only fixed set of elements in C# array.



Advantages

- Code Optimization (less code)
- Random Access
- Easy to traverse data
- Easy to manipulate data
- Easy to sort data etc.

Disadvantages

- Fixed size

Types

There are 3 types of arrays in C# programming:

1. Single Dimensional Array

2. Multidimensional Array
3. Jagged Array

1. Single Dimensional Array

To create single dimensional array, you need to use square brackets [] after the type.

Syntax:

```
Datatype[] array_name = new Datatype[Size];
```

Example:

```
using System;
public class ArrayExample
{
   public static void Main(string[] args)
   {
      int[] arr = new int[5];//creating array
      arr[0] = 10;//initializing array
      arr[2] = 20;
      arr[4] = 30;

      //traversing array
      for (int i = 0; i < arr.Length; i++)
      {
         Console.WriteLine(arr[i]);
      }
   }
}
```

Output:

```
10
0
20
0
30
```

2. Multidimensional Array

- C# also supports multi-dimensional arrays.
- A multi-dimensional array is a two dimensional series like rows and columns.

Syntax:

```
Datatype[ , ] array_name = new Datatype[row, column];
```

Example:

```
using System;
public class Program
{
        public static void Main(){
                int[,] intArray = new int[3,2]{  {1, 2}, {3, 4}, {5, 6} };
                Console.WriteLine(intArray[0, 0]);

                Console.WriteLine(intArray[0, 1]);

                Console.WriteLine(intArray[1, 0]);

                Console.WriteLine(intArray[1, 1]);

                Console.WriteLine(intArray[2, 0]);

                Console.WriteLine(intArray[2, 1]);
        }
}
```

Output:

```
1
2
3
4
5
6
```

3. Jagged Array

- A jagged array is an array of an array.
- Jagged arrays store arrays instead of any other data type value directly.
- A jagged array is initialized with two square brackets [][].
- The first bracket specifies the size of an array and the second bracket specifies the dimension of the array which is going to be stored as values. (Remember, jagged array always store an array.)

The following jagged array stores a two single dimensional array as a value:

Syntax:

```
Datatype[][,] Array_name = new Datatype[Size][,];
```

Example:

```
using System;
public class Program  {
        public static void Main() {
                int[][] intJaggedArray = new int[2][];

                intJaggedArray[0] = new int[3]{1,2,3};

                intJaggedArray[1] = new int[2]{4,5};
                Console.WriteLine(intJaggedArray[0][0]);
                Console.WriteLine(intJaggedArray[0][2]);
                Console.WriteLine(intJaggedArray[1][1]);
        }
}
```

Output:

```
1
3
5
```

Traversal using foreach loop

- the array elements using foreach loop.
- It returns array element one by one.

Example:

```
using System;
public class ArrayExample
{
    public static void Main(string[] args)
    {
        int[] arr = { 10, 20, 30, 40, 50 };//creating and initializing array

        //traversing array
        foreach (int i in arr)
        {
            Console.WriteLine(i);
        }
    }
}
```

Output:

```
10
20
30
40
50
```

Structures:

- a structure is a value type data type.
- It make a single variable hold related data of various data types.
- The struct keyword is used for creating a structure.
- Structures are used to represent a record.

Example:

```
using System;

struct Books {
  public string title;
  public string author;
  public string subject;
  public int book_id;
};

public class testStructure {
  public static void Main(string[] args) {
    Books Book1;   /* Declare Book1 of type Book */
    Books Book2;   /* Declare Book2 of type Book */

    /* book 1 specification */
    Book1.title = "C Programming";
    Book1.author = "Nuha Ali";
    Book1.subject = "C Programming Tutorial";
    Book1.book_id = 6495407;

    /* book 2 specification */
    Book2.title = "Telecom Billing";
    Book2.author = "Zara Ali";
    Book2.subject =  "Telecom Billing Tutorial";
    Book2.book_id = 6495700;

    /* print Book1 info */
    Console.WriteLine( "Book 1 title : {0}", Book1.title);
    Console.WriteLine("Book 1 author : {0}", Book1.author);
    Console.WriteLine("Book 1 subject : {0}", Book1.subject);
    Console.WriteLine("Book 1 book_id :{0}", Book1.book_id);

    /* print Book2 info */
    Console.WriteLine("Book 2 title : {0}", Book2.title);
    Console.WriteLine("Book 2 author : {0}", Book2.author);
    Console.WriteLine("Book 2 subject : {0}", Book2.subject);
    Console.WriteLine("Book 2 book_id : {0}", Book2.book_id);

    Console.ReadKey();
  }
}
```

Output:

```
Book 1 title : C Programming
Book 1 author : Nuha Ali
Book 1 subject : C Programming Tutorial
Book 1 book_id : 6495407

Book 2 title : Telecom Billing
Book 2 author : Zara Ali
Book 2 subject : Telecom Billing Tutorial
```

Unit – II VB.NET

Introduction to VB.Net

- Visual Basic is a third-generation event-driven programming language first released by Microsoft in 1991.
- Visual Basic .NET (VB.NET) is an object-oriented programming language implemented on the .NET Framework
- VB.NET is an object, including all of the primitive types (Short, Integer, Long, String, Boolean, etc.) and user-defined types, events, and even assemblies. All objects inherits from the base class Object.
- It has full access to all the libraries in the .Net Framework.

The following reasons make VB.Net a widely used professional language −

- ✓ Modern, general purpose.
- ✓ Object oriented.
- ✓ Component oriented.
- ✓ Easy to learn.
- ✓ Structured language.
- ✓ It produces efficient programs.
- ✓ It can be compiled on a variety of computer platforms.
- ✓ Part of .Net Framework.

VB.NET fundamentals:

Variables:

- A variable is a memory location that is used to store data.
- A variable is referred by a name and can store data of a particular data type.
- Data stored in a variable can change at any point in a program.

Data type:

It is a kind of data it may be either number or text.

Example: Integer, bit, string, Date …

Constants:

- It is similar to variable used to store values.
- Values stored in a constant always in same.

Example: pi=3.14

Array

- The array is a group of similar types of elements.
- That have a contiguous set of memory location.
- Array is an *object* of base type **System.Array**.
- In VB.NET array index starts from 0. We can store only fixed set of elements in VB.NET array.

Example:

```
Module arrayApl
  Sub Main()
    Dim n(10) As Integer ' n is an array of 11 integers '
    Dim i, j As Integer
    ' initialize elements of array n '

    For i = 0 To 5
       n(i) = i + 100 ' set element at location i to i + 100
    Next i
    ' output each array element's value '
    For j = 0 To 5
       Console.WriteLine("Element({0}) = {1}", j, n(j))
    Next j
    Console.ReadKey()
  End Sub
End Module
```

Output;

```
Element(0) = 100     Element(1) = 101    Element(2) = 102    Element(3) = 103
Element(4) = 104    Element(5) = 105
```

Object:

- Objects are the basic run time entity they may represent a person a place.
- **Objects** have states and behaviors.

Example:
A dog has states - color, name, breed as well as behaviors – wagging the tail, barking, eating. An object is an instance of a class.

Class Definition

- A class can be defined as a template/blueprint that describes the behavior/state that the object of its type support.
- A class definition starts with the keyword **Class** followed by the class name
- the class body, ended by the End Class statement.

Syntax:

```
Class name [ ( Of typelist ) ]
  [ Inherits classname ]
  [ Implements interfacenames ]
  [ statements ]
End Class
```

Control Structure in VB.NET

- The program have to perform some action or take decision based on some condition at that time you need to use control structure.

- Control structures are very useful for taking decisions based on the condition.
- Control structures structure first test for the condition and based on the outcome of that condition they perform the specified tasks.

  Following are the list of Control structures :

  (1) If … Then … EndIf

  (2) If … Then … Else … EndIf

  (3) If … Then … ElseIf … EndIf

  (4) Select Case….End Select

(1) If …Then … End If Statement

The general syntax of the If … Then …. End If structure is given below:

Syntax:

```
If condition Then
Statement Block
End if
```

Example:

```
If txtName.text = “” then
MsgBox ("Please Enter Name")
txtName.Focus ()
End If
```

(2) If … Then … Else … End If

The General syntax of If … Then … Else … End If structure is given below:

Syntax:

```
If condition then
Statement block 1
Else
Statement block 2
End if
```

Example:

```
If val (txtNumber1.text) > val (txtNumber2.text) then
MsgBox (txtNumber1.text & “Is Maximum”)
Else
MsgBox (txtNumber2.text & “Is Maximum”)
End If
```

(3) If … Then … ElseIf … End If

The general syntax of If … Then … ElseIf … End If structure is given below:

Syntax:

```
If Condition1 Then
Statement Block 1
ElseIf Condition2 Then
Statement Block 2
ElseIf Condition3 Then
Statement Block 3
……..
ElseIf ConditionN Then
Statement Block N
Else
Default Statement Block
End if
```

Example:

```
If txtName.text = “” then
MsgBox (“Please Enter Name”)
txtName. Focus ()
ElseIf txtAge.text = “” then
MsgBox (“Please Enter Age”)
txtAge. Focus ()
ElseIf txtBasic.text = “” then
MsgBox (“Please Enter Basic”)
txtBasic. Focus ()
Else
Basic = Val (txtBasic.text)
DA = Basic * 0.80
HRA = (Basic + DA) * 0.12
Gross = Basic + DA + HRA
txtGross.text = Gross
End If
```

(4) Select Case … End Select

It is also known as multiple choice decision statement. It allows you to select one option from the list of available options. It is the alternative of If …Then ... ElseIf structure. The general syntax for Select Case …. End Select structure is given below:

```
Select Case expression
Case Value1
Statement Block 1
Case Value 2
Statement Block 2
```

```
…………………………
Case Value N
Statement Block N
Case Else
Default statement Block
End Select

Dim A as integer, B as integer, C as Integer
Dim op as string
Select Case op
Case "+"
C = A + B
Case "-"
C = A - B
Case "*"
C = A * B
Case "/"
C = A / B
Case Else
MsgBox ("Wrong Option")
End Select
```

Looping Control Structure

it is required to perform some action or task repeatedly for specified number of times or until some condition is satisfied at that time you need to use Looping control structure.

Following are the List of Looping Control Structure:

(1) Do While … Loop

(2) For … Next

(1) Do While …. Loop

Do while … Loop control structure is used to repeat statements between Do While and Loop statements until the condition specified with the Do While statements evaluates to FALSE.

The general syntax of Do While … Loop structure is given below:

```
Do while Condition
Statement Block
Loop
Dim a As Integer
a = 1
Do While a <= 10
If a Mod 2 <> 0 Then
lblOdd.Text = lblOdd.Text & " " & a
```

```
End If
a = a + 1
Loop
```

(5) For…Next

For…Next control structure is used to repeat Statement Block for specified number of times.

The general syntax of For … Next statement is given below:

```
For counter=Start-Value To End-Value [step Step-Value]
Statement Block
Next
```

Example:

```
Dim i As Integer
Dim n As Integer = 4
Dim fact As Integer = 1
For i = 1 To n Step 1
fact = fact * i
Next
Label1.Text = fact
```

Constructor:

- A Constructor is a special kinds of member function that used to initialize the object.
- A constructor is like a method in that it contain executable code and may be defined with parameter.
- This is first method that is run when an instance of type is created.

Constructor is two types in VB.NET

- Instance constructor
- Shared constructor

Instance constructor:-

"An Instance constructor runs whenever the CLR creates an object from a class"

coding for instance constructor:-

```
Module Module1
   Sub Main()
      Dim con As New Constructor("Hello world")
      Console.WriteLine(con.display())
      'display method
   End Sub
End Module

Public Class Constructor
```

```
    Public x As String
    Public Sub New(ByVal value As String)
        'constructor
        x = value
        'storing the value of x in constructor
    End Sub
    Public Function display() As String
        Return x
        'returning the stored value
    End Function
End Class
```

Output:- Hellow world

Shared Constructor:-

"Shared constructor are most often used to initialize class level data such as shared fields"

coding for shared constructor:-

```
Module Testcons
    Sub Main()
        Console.WriteLine("100")
        B.G()
        Console.WriteLine("200")
    End Sub
End Module
Class A
    Shared Sub New()
        Console.WriteLine("Init A")
    End Sub
End Class
Class B
    Inherits A
    Shared Sub New()
        Console.WriteLine("Init B")
    End Sub
    Public Shared Sub G()
        Console.WriteLine("Hello world")
    End Sub
End Class
```

Output:
```
100
200
Hello world
```

Overriding

A method in a subclass has the same name, same parameters or signature and same return type (or sub-type) as a method in its super-class, then the method in the subclass is said to *override* the method in the super-class.

Method overriding is one of the way by Run Time Polymorphism. That same name function with same parameters in deferent deferent classes called Run time Polymorphism.

Example:

```
Public Class poly

Public Sub show()
WriteLine("hellow")

End Sub

End Class

Public Class poly2
Inherits poly
Public Sub show()
WriteLine("world")
End Sub

End Class

Class output
Public Shared Sub Main()
Dim a As poly2 = New poly2()
a.show()
End Sub

End Class
```

Output:

```
        World
```

Inheritance:

- Inheritance is a mechanism where in a new class is derived from an existing class.
- In VB.NET, classes may inherit the properties and methods of other classes.
- A class derived from another class is called a subclass.
- the class from which a subclass is derived is called a superclass.
- A subclass can have only one superclass, whereas a superclass may have one or more subclasses.

Syntax:

```
Public Class Base
   ----
       ----
End Class
```

```
Public Class Derived
    Inherits Base
    'Derived class inherits the Base class
    ----
        ----
End Class
```

Example:

```
Imports System. Console
Module Module1

    Sub Main()
        Dim Obj As New Subclass()
        WriteLine(Obj.sum())
        Read()
    End Sub

End Module

Public Class Superclass
    'base class
    Public X As Integer = 5
    Public Y As Integer = 15
End Class

Public Class Subclass
    Inherits Base
    'derived class. Class Subclass inherited from class Superclass
    Public Z As Integer = 25

    Public Function sum() As Integer
        'using the variables, function from superclass and adding more functionality
        Return X + Y + Z
    End Function
End Class
```

Output:

```
45
```

Polymorphism

- "Manipulated the object of various classes and invoke method on one object without knowing the object type".

- 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.
- the use of same thing for different purposes. It used to create as many functions 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.

  Example:-polymorphism real word examples are **Student**, **Manager**, **Animal** and etc.

Polymorphism can be divide in to two parts,

- Compile time polymorphism
- Run time polymorphism

Compile time polymorphism:-

- compile time polymorphism achieved by "Method Overloading",
- That same name function with deferent parameters in same class called compile time polymorphism.

Example:

```
Module Module1

Sub Main()
Dim two As New One()
WriteLine(two.add(10))
'calls the function with one argument
WriteLine(two.add(10, 20))
'calls the function with two arguments
WriteLine(two.add(10, 20, 30))
'calls the function with three arguments
End Sub
End Module

Public Class One
Public i, j, k As Integer

Public Function add(ByVal i As Integer) As Integer
'function with one argument
Return i
End Function

Public Function add(ByVal i As Integer, ByVal j As Integer) As Integer
'function with two arguments
Return i + j
End Function
```

```
Public Function add(ByVal i As Integer, ByVal j As Integer, ByVal k As Integer) As Integer
'function with three arguments
Return i + j + k
End Function

End Class
```

Output:

```
        10
        20
        30
```

Run time Polymorphism:-

- ➢ Run time polymorphism achieved by "Method Overriding or Operator Overloading"that same name function with same parameters in deferent deferent classes called Run time Polymorphism.

Example:

```
Public Class poly
Public Sub show()
WriteLine("hellow")
End Sub
End Class

Public Class poly2
Inherits poly
Public Sub show()
WriteLine("world")
End Sub
End Class

Class output
Public Shared Sub Main()
Dim a As poly2 = New poly2()
a.show()
End Sub
End Class
```

Output:

```
        World
```

Interfaces:-

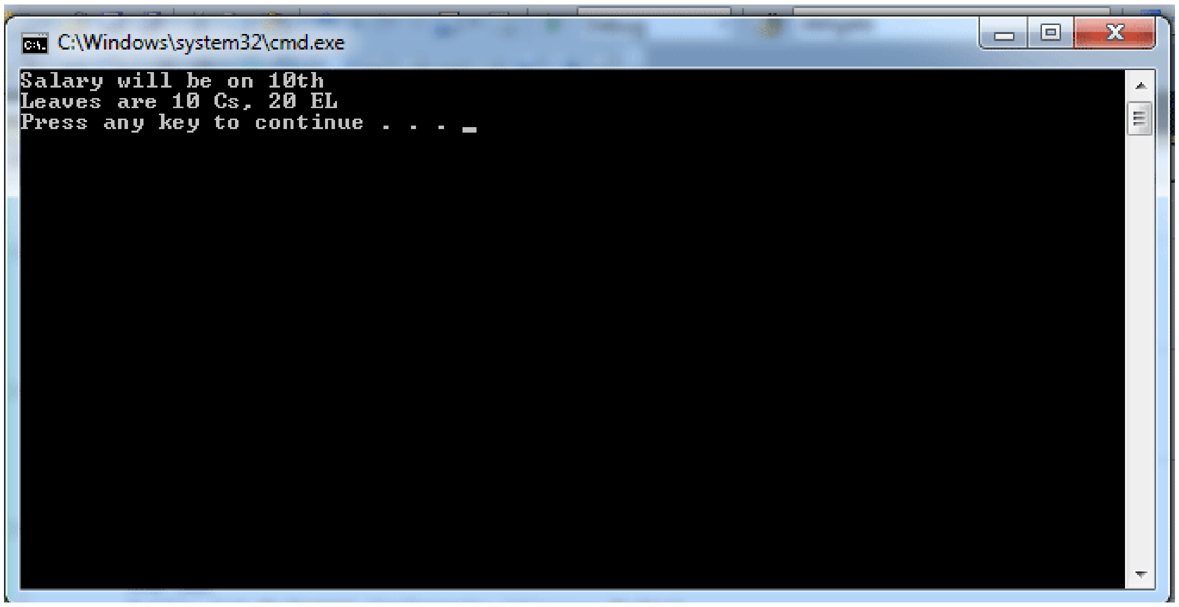
- ➢ A user defined data type similar to class but contains all **abstract** methods.
- ➢ All methods are abstract and public by default.
- ➢ All such methods are overridden in child class.
- ➢ Allows implementing the multiple inheritance.
- ➢ A class can inherit only one other class but any number of interfaces.
- ➢ All interfaces in .NET starts with I.

- **Implements** keyword to implement the interface.
- Use the **Interface** keyword to create an interface.

Example:

```
Module Module1
    Interface Common
        Sub Leaves()
    End Interface
    Interface IHr
        Inherits Common
        Sub ShowSalary()
    End Interface
    Interface IFinance
        Inherits Common
        Sub Budget()
    End Interface
    Class ERP
        Implements IHr
        Implements IFinance
        Public Sub ShowSalary() Implements IHr.ShowSalary
            System.Console.WriteLine("Salary will be on 10th")
        End Sub
        Public Sub Budget() Implements IFinance.Budget
            System.Console.WriteLine("Budget is 10 L")
        End Sub
        Public Sub Leaves() Implements Common.Leaves
            System.Console.WriteLine("Leaves are 10 Cs, 20 EL")
        End Sub
    End Class

    Class ITC
        Public Shared Sub Main()
            Dim h As IHr = New ERP()
            h.ShowSalary()
            h.Leaves()
        End Sub
    End Class
End Module
```

OUTPUT:



```
Salary will be on 10th
Leaves are 10 Cs, 20 EL
Press any key to continue . . .
```

Exception:

- ➢ An exception is a problem that arises during the execution of a program.
- ➢ An exception is a response to an exceptional circumstance that arises while a program is running, such as an attempt to divide by zero.
- ➢ Exceptions provide a way to transfer control from one part of a program to another.
- ➢ Exception handling is built upon four keywords:

Try, **Catch**, **Finally** and **Throw**.

- **Try** − A Try block identifies a block of code for which particular exceptions will be activated. It's followed by one or more Catch blocks.
- **Catch** − A program catches an exception with an exception handler at the place in a program where you want to handle the problem. The Catch keyword indicates the catching of an exception.
- **Finally** − The Finally block is used to execute a given set of statements, whether an exception is thrown or not thrown. For example, if you open a file, it must be closed whether an exception is raised or not.
- **Throw** − A program throws an exception when a problem shows up. This is done using a Throw keyword.

Syntax:

```
Try
   [ tryStatements ]
   [ Exit Try ]
[ Catch [ exception [ As type ] ] [ When expression ]
   [ catchStatements ]
   [ Exit Try ] ]
[ Catch ... ]
[ Finally
  [ finallyStatements ] ]
End Try
```

Example:

```
Module Module1

   Sub Main()
       Try
           ' Try to divide by zero.
           Dim value As Integer = 1 / Integer.Parse("0")
           ' This statement is sadly not reached.
           Console.WriteLine("Hi")
       Catch ex As Exception
           ' Display the message.
           Console.WriteLine(ex.Message)
       End Try
   End Sub

End Module
```

Output

```
Arithmetic operation resulted in an overflow.
```

Delegates and Events

- Delegates are objects that refer to methods.
- They are sometimes described as *type-safe function pointers* because they are similar to function pointers used in other programming languages. But unlike function pointers,
- Visual Basic delegates are a reference type based on the class System.Delegate.
- Delegates can reference both shared methods — methods that can be called without a specific instance of a class — and instance methods.
- Events in VB.NET are handled by delegates, which serve as a mechanism that defines one or more callback functions to process events.
- An event is a message sent by an object to signal the occurrence of an action.
- The action could arise from user interaction, such as a mouse click, or could be triggered by some other program logic.
- The object that triggers the event is called the event sender.
- The object that captures the event and responds to it is called the event receiver.
- In event communication, the event sender class does not know which object or method will handle the events it raises.
- It merely functions as an intermediary or pointer-like mechanism between the source and the receiver.
- The .NET framework defines a special type delegate that serves as a function pointer.

Example

```
Public Class MyEvt
   Public Delegate Sub t(ByVal sender As [Object], ByVal e As MyArgs)
   ' declare a delegate
   Public Event tEvt As t
   'declares an event for the delegate
   Public Sub mm()
      'function that will raise the callback
      Dim r As New MyArgs()
      RaiseEvent tEvt(Me, r)
      'calling the client code
   End Sub
   Public Sub New()
   End Sub
End Class
'arguments for the callback
Public Class MyArgs
   Inherits EventArgs
   Public Sub New()
   End Sub
End Class
Public Class MyEvtClient
   Private oo As MyEvt
   Public Sub New()
      Me.oo = New MyEvt()
      AddHandler Me.oo.tEvt, New MyEvt.t(AddressOf oo_tt)
   End Sub
   Public Shared Sub Main(ByVal args As [String]())
      Dim cc As New MyEvtClient()
      cc.oo.mm()
   End Sub
   'this code will be called from the server
   Public Sub oo_tt(ByVal sender As Object, ByVal e As MyArgs)
      Console.WriteLine("yes")
      Console.ReadLine()
   End Sub
End Class
```

OUTPUT:

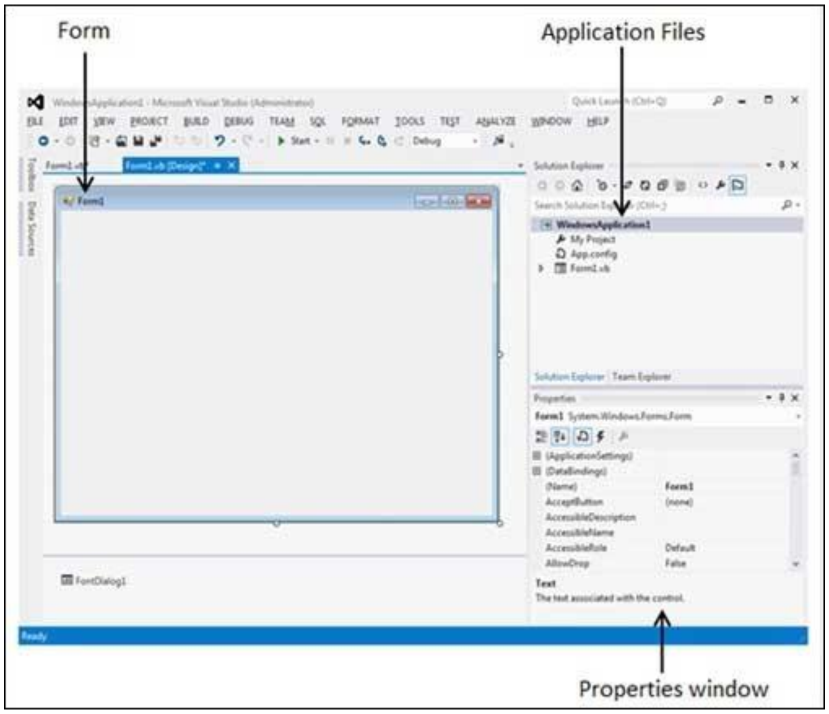
```
yes
```

Unit – III controls

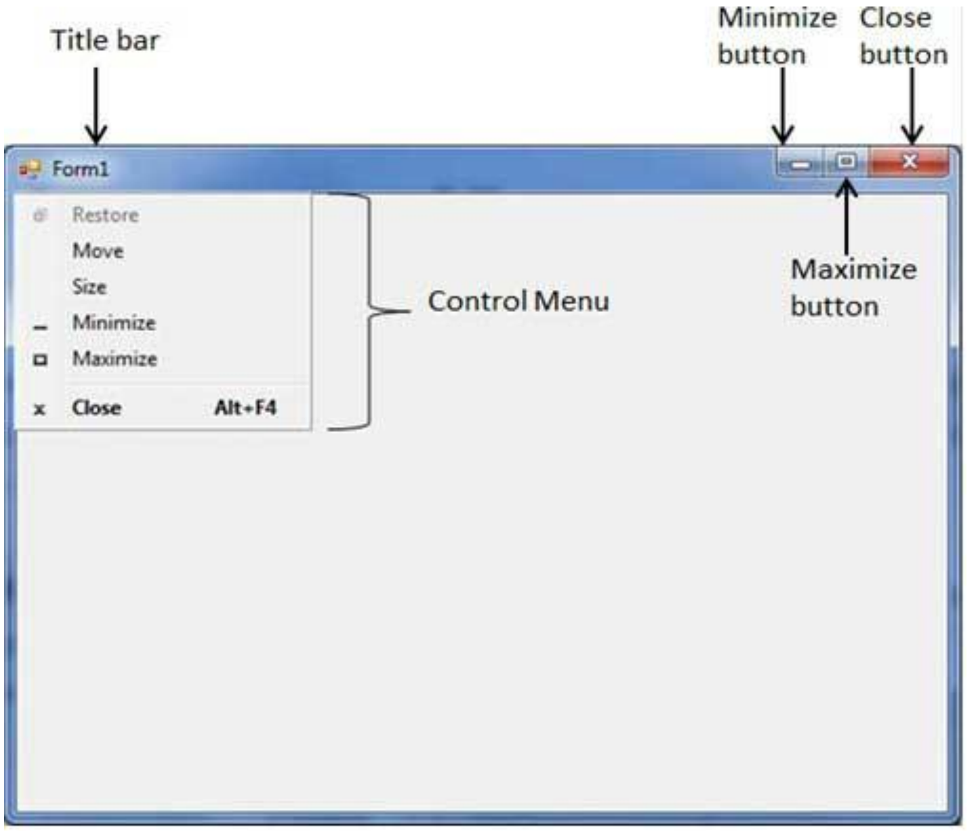
Windows - Forms

The creating a Window Forms Application by following steps in Microsoft Visual Studio - **File → New Project → Windows Forms Applications** Finally, select OK,

Microsoft Visual Studio creates your project and displays following window Form with a name **Form1**.



Visual Basic Form is the container for all the controls that make up the user interface.



If you click the icon on the top left corner, it opens the control menu, which contains the various commands to control the form like to move control from one place to another place, to maximize or minimize the form or to close the form.

Form Properties

S.N	Properties	Description
1	**AcceptButton**	The button that's automatically activated when you press Enter, no matter which control has the focus at the time. Usually the OK button on a form is set as AcceptButton for a form.
2	**CancelButton**	The button that's automatically activated when you hit the Esc key. Usually, the Cancel button on a form is set as CancelButton for a form.

3	**AutoScale**	This Boolean property determines whether the controls you place on the form are automatically scaled to the height of the current font. The default value of this property is True. This is a property of the form, but it affects the controls on the form.
4	**AutoScroll**	This Boolean property indicates whether scroll bars will be automatically attached to the form if it is resized to a point that not all its controls are visible.
5	**AutoScrollMinSize**	This property lets you specify the minimum size of the form, before the scroll bars are attached.
6	**AutoScrollPosition**	The AutoScrollPosition is the number of pixels by which the two scroll bars were displaced from their initial locations.
7	**BackColor**	Sets the form background color.
8	**BorderStyle**	The BorderStyle property determines the style of the form's border and the appearance of the form − • **None** − Borderless window that can't be resized. • **Sizable** − This is default value and will be used for resizable window that's used for displaying regular forms. • **Fixed3D** − Window with a visible border, "raised" relative to the main area. In this case, windows can't be resized. • **FixedDialog** − A fixed window, used to create dialog boxes. • **FixedSingle** − A fixed window with a single line border. • **FixedToolWindow** − A fixed window with a Close button only. It looks like the toolbar displayed by the drawing and imaging applications. • **SizableToolWindow** − Same as the FixedToolWindow but resizable. In addition, its caption font is smaller than the usual.
9	**ControlBox**	By default, this property is True and you can set it to False to hide the icon and disable the Control menu.

10	**Enabled**	If True, allows the form to respond to mouse and keyboard events; if False, disables form.
11	**Font**	This property specify font type, style, size
12	**HelpButton**	Determines whether a Help button should be displayed in the caption box of the form.
13	**Height**	This is the height of the Form in pixels.
14	**MinimizeBox**	By default, this property is True and you can set it to False to hide the Minimize button on the title bar.
15	**MaximizeBox**	By default, this property is True and you can set it to False to hide the Maximize button on the title bar.
16	**MinimumSize**	This specifies the minimum height and width of the window you can minimize.
17	**MaximumSize**	This specifies the maximum height and width of the window you maximize.
18	**Name**	This is the actual name of the form.
19	**StartPosition**	This property determines the initial position of the form when it's first displayed. It will have any of the following values − • **CenterParent** − The form is centered in the area of its parent form. • **CenterScreen** − The form is centered on the monitor. • **Manual** − The location and size of the form will determine its starting position. • **WindowsDefaultBounds** − The form is positioned at the default location and size determined by Windows.

		• **WindowsDefaultLocation** − The form is positioned at the Windows default location and has the dimensions you've set at design time.
20	**Text**	The text, which will appear at the title bar of the form.
21	**Top, Left**	These two properties set or return the coordinates of the form's top-left corner in pixels.
22	**TopMost**	This property is a True/False value that lets you specify whether the form will remain on top of all other forms in your application. Its default property is False.
23	**Width**	This is the width of the form in pixel.

Form Methods

Sr.No.	Method Name & Description
1	**Activate** Activates the form and gives it focus.
2	**ActivateMdiChild** Activates the MDI child of a form.
3	**AddOwnedForm** Adds an owned form to this form.
4	**BringToFront** Brings the control to the front of the z-order.
5	**CenterToParent** Centers the position of the form within the bounds of the parent form.
6	**CenterToScreen** Centers the form on the current screen.
7	**Close** Closes the form.
8	**Contains**

	Retrieves a value indicating whether the specified control is a child of the control.
9	**Focus** Sets input focus to the control.
10	**Hide** Conceals the control from the user.
11	**Refresh** Forces the control to invalidate its client area and immediately redraw itself and any child controls.
12	**Scale(SizeF)** Scales the control and all child controls by the specified scaling factor.
13	**ScaleControl** Scales the location, size, padding, and margin of a control.
14	**ScaleCore** Performs scaling of the form.
15	**Select** Activates the control.
16	**SendToBack** Sends the control to the back of the z-order.
17	**SetAutoScrollMargin** Sets the size of the auto-scroll margins.
18	**SetDesktopBounds** Sets the bounds of the form in desktop coordinates.
19	**SetDesktopLocation** Sets the location of the form in desktop coordinates.
20	**SetDisplayRectLocation** Positions the display window to the specified value.
21	**Show** Displays the control to the user.
22	**ShowDialog** Shows the form as a modal dialog box.

Form Events

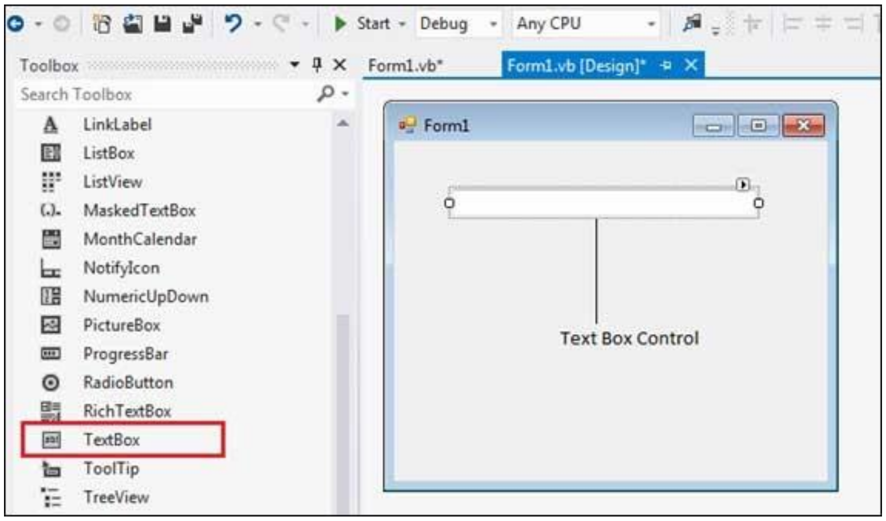
Sr.No.	Event	Description
1	**Activated**	Occurs when the form is activated in code or by the user.
2	**Click**	Occurs when the form is clicked.
3	**Closed**	Occurs before the form is closed.
4	**Closing**	Occurs when the form is closing.
5	**DoubleClick**	Occurs when the form control is double-clicked.
6	**DragDrop**	Occurs when a drag-and-drop operation is completed.
7	**Enter**	Occurs when the form is entered.
8	**GotFocus**	Occurs when the form control receives focus.
9	**HelpButtonClicked**	Occurs when the **Help** button is clicked.
10	**KeyDown**	Occurs when a key is pressed while the form has focus.
11	**KeyPress**	Occurs when a key is pressed while the form has focus.
12	**KeyUp**	Occurs when a key is released while the form has focus.
13	**Load**	Occurs before a form is displayed for the first time.
14	**LostFocus**	Occurs when the form loses focus.
15	**MouseDown**	Occurs when the mouse pointer is over the form and a mouse button is pressed.
16	**MouseEnter**	Occurs when the mouse pointer enters the form.
17	**MouseHover**	Occurs when the mouse pointer rests on the form.
18	**MouseLeave**	Occurs when the mouse pointer leaves the form.
19	**MouseMove**	Occurs when the mouse pointer is moved over the form.
20	**MouseUp**	Occurs when the mouse pointer is over the form and a mouse button is released.
21	**MouseWheel**	Occurs when the mouse wheel moves while the control has focus.
22	**Move**	Occurs when the form is moved.

23	**Resize**	Occurs when the control is resized.
24	**Scroll**	Occurs when the user or code scrolls through the client area.
25	**Shown**	Occurs whenever the form is first displayed.
26	**VisibleChanged**	Occurs when the Visible property value changes.

Text box

controls allow entering text on a form at runtime. By default, it takes a single line of text, however, you can make it accept multiple texts and even add scroll bars to it.

Let's create a text box by dragging a Text Box control from the Toolbox and dropping it on the form.



The Properties of the TextBox Control

The following are some of the commonly used properties of the TextBox control −

s.no	Property & Description
1)	**Font** Gets or sets the font of the text displayed by the control.
2)	**FontHeight** Gets or sets the height of the font of the control.

3)	**ForeColor** Gets or sets the foreground color of the control.
4)	**Lines** Gets or sets the lines of text in a text box control.
5)	**Multiline** Gets or sets a value indicating whether this is a multiline TextBox control.
6)	**PasswordChar** Gets or sets the character used to mask characters of a password in a single-line TextBox control.
7)	**ReadOnly** Gets or sets a value indicating whether text in the text box is read-only.
8)	**ScrollBars** Gets or sets which scroll bars should appear in a multiline TextBox control. This property has values − • None • Horizontal • Vertical • Both
9)	**TabIndex** Gets or sets the tab order of the control within its container.
10)	**Text** Gets or sets the current text in the TextBox.
11)	**TextAlign** Gets or sets how text is aligned in a TextBox control. This property has values − • Left • Right • Center

12)	**TextLength** Gets the length of text in the control.
13)	**WordWrap** Indicates whether a multiline text box control automatically wraps words to the beginning of the next line when necessary.

Methods

s.no	Method Name & Description
1)	**Clear** Clears all text from the text box control.
2)	**Copy** Copies the current selection in the text box to the **Clipboard**.
3)	**Cut** Moves the current selection in the text box to the **Clipboard**.
4)	**Paste** Replaces the current selection in the text box with the contents of the **Clipboard**.
5)	**Paste(String)** Sets the selected text to the specified text without clearing the undo buffer.
6)	**ResetText** Resets the Text property to its default value.
7)	**ToString** Returns a string that represents the TextBoxBase control.
8)	**Undo** Undoes the last edit operation in the text box.

Events:

Sr.No.	Event & Description
1	**Click** Occurs when the control is clicked.

2	**DoubleClick** Occurs when the control is double-clicked.
3	**TextAlignChanged** Occurs when the TextAlign property value changes.

RichTextBox Control in VB.NET

- ➢ RichTextBox Control allows user to display, input, edit and format text information.
- ➢ RichTextBox Control supports advance formatting features as compared to TextBox. Using RichTextBox user can format only selected portion of the text. User can also format paragraph using RichTextBox Control.

Properties

Property	Purpose
AutoWordSelection	It is used to specify weather automatic word selection is ON or OFF. It has Boolean value. Default value is false.
BackColor	It is used to get or set background color of the RichTextBox.
ContextMenuStrip	It is used to specify the name of shortcut menu that is displayed when user right clicks on the RichTextBox.
DetectUrls	It is used to specify weather URL that is entered in RichTextBox is automatically displayed as hyperlink or not. It has Boolean value. Default value is true.
Enabled	It is used to specify weather RichTextBox Control is enabled or not at run time. It has Boolean value. Default value is true.
Font	It is used to set Font Face, Font Style, Font Size and Effects of the text associated with RichTextBox Control.
ForeColor	It is used to get or set Fore color of the text associated with RichTextBox Control.
HideSelection	It is used to specify weather text selection should be hidden or not when RichTextBox lose its focus. It has Boolean value. Default value is true.

MaxLength	It is used to get or set maximum number of characters that can be entered in RichTextBox. Default value is 2147483647.
Multiline	It is used to specify weather RichTextBox can be expanded to enter more than one line of text or not. It has Boolean value. Default value is true.
ReadOnly	It is used to specify weather text associated with RichTextBox is ReadOnly or not. It has Boolean value. Default value is false.
RightMargin	It is used to get or set right margin of the text in RichTextBox.
ScrollBars	It is used to get or set type of scrollbars to be added with RichTextBox control. It has following 4 options: (1) None (2) Horizontal (2) Vertical (3) Both Default value is Both.
Size	It is used to get or set height and width of RichTextBox control in pixel.
Text	It is used to get or set text associated with RichTextBox Control.
TabIndex	It is used to get or set Tab order of the RichTextBox.
TabStop	It is used to specify weather user can use TAB key to set focus on RichTextBox Control or not. It has Boolean value. Default value is true.
Visible	It is used to specify weather RichTextBox Control is visible or not at run time. It has Boolean value. Default value is true.
WordWrap	It is used to specify weather line will be automatically word wrapped while entering multiple line of text in RichTextBox control or not. It has Boolean value. Default value is true.
ZoomFactor	It is used to get or set current zoom level of RichTextBox.
SelectedRtf	It is used to get or set currently selected RTF formatted text in RichTextBox.
SelectedText	It is used to get or set currently selected text in RichTextBox.
SelectionAlignment	It is used to get or set horizontal alignment of the text selected in RichTextBox.
SelectionBackColor	It is used to get or set BackColor of the text selected in RichTextBox.

SelectionBullet	It is used to get or set value which determines weather bullet style should be applied to selected text or not.
SelectionColor	It is used to get or set Fore Color of the text selected in RichTextBox.
SelectionFont	It is used to get or set font face, font style, and font size of the text selected in RichTextBox.
SelectionLength	It is used to get or set number of characters selected in the RichTextBox.
SelectionStart	It is used to get or set starting point of the text selected in the RichTextBox.

Methods

Method	Purpose
Cut	It is used to move current selection of RichTextBox into clipboard.
Copy	It is used to copies selected text of RichTextBox in clipboard.
Paste	It is used to replace current selection of TextBox by contents of clipboard. It is also used to move contents of Clipboard to RichTextBox control where cursor is currently located.
Select	It is used to select specific text from RichTextBox.
SelectAll	It is used to select all text of RichTextBox.
DeselectAll	It is used to deselect all text selected in RichTextBox.
Clear	It is used to clear all text from RichTextBox Control.
AppendText	It is used to append text at the end of current text in RichTextBox Control.
ClearUndo	It is used to clear information from the Undo buffer of the RichTextBox.
Find	It is used to find starting position of first occurrence of a string in the RichTextBox control. If a string is not found then it returns -1.
SaveFile	It is used to save contents of RichTextBox in to Rich Text Format (RTF) file.
LoadFile	It is used to loads a Rich Text Format (RTF) file or standard ASCII text file into RichTextBox Control.
Undo	It is used to undo last edit operation of RichTextBox.
Redo	It is used to redo the last operation that is undo using undo method.

Events

Event	Purpose
TextChanged	It is the default event of RichTextBox Control. It fires each time a text in the RichTextBox control changed.

Label

- It is used to display some text on the form which user cannot edit.
- The user can edit contents of the label control at run time using text property.
- Labels are widely used with form to display text before various controls. So user can easily understand the purpose of controls for which they are placed on the form.

Properties

Property	Purpose
BackColor	It is used to get or set background color of the label.
Font	It is used to set Font Face, Font Style, Font Size and Effects of the text associated with Label Control.
ForeColor	It is used to get or set Fore color of the text associated with Label Control.
Enabled	It is used to specify weather label control is enabled or not at run time. It has Boolean value. Default value is true.
FlatStyle	It is used to get or set appearance of the Label Control when user moves mouse on it or click on it. It has following 4 options: System, Popup, Standard, Flat
Image	It is used to specify an image that is displayed in Label Control.
ImageAlign	It is used to get or set alignment of the image that is displayed in the Label control.
Text	It is used to get or set text associated with the Label control.
TextAlign	It is used to get or set alignment of the text associated with the Label control.
Visible	It is used to specify weather label control is visible or not at run time. It has Boolean value. Default value is true.

Methods

Method	Purpose
Show	It is used to show label control at run time.
Hide	It is used to hide label control at run time.

Events

Event	Purpose
Click	It is the default event of Label Control. It fires each time user clicks on Label Control.
DoubleClick	It fires each time user double clicks on Label Control.
TextChanged	It fires each a text associated with Label Control is changed.

Linklabel

- LinkLabel Control is designed such that it provides the functionality of Hyperlink in window application.
- It is derived from label Control so it also provides all the functionality of Label control.

Properties

Property	Purpose
LinkColor	It is used to get or set Fore color of the Hyperlink in its default state.
ActiveLinkColor	It is used to get or set Fore color of the Hyperlink when user clicks it.
DisabledLinkColor	It is used to get or set Fore color of the Hyperlink when LinkLabel is disabled.
VisitedLinkColor	It is used to get or set Fore color of the Hyperlink when LinkVisited property of LinkLabel is set to true.

LinkVisited	It is used to specify weather Hyperlink is already visited or not. It has Boolean value. Default value is false.
Text	It is used to get or set text associated with LinkLabel Control.
TextAlign	It is used to get or set alignment of the text associated with LinkLabel Control.
ForeColor	It is used to get or set Fore Color of the text associated with LinkLabel Control.
BackColor	It is used to get or set Background color of the LinkLabel Control.
Enabled	It is used to specify weather LinkLabel control is enabled or not at runtime. It has Boolean value true or false. Default value is true.
Visible	It is used to specify weather LinkLabel control is visible or not at runtime. It has Boolean value true or false. Default value is true.

Methods

Method	Purpose
Show	It is used to Show LinkLabel Control at runtime.
Hide	It is used to Hide LinkLabel Control at runtime.
Focus	It is used to set input focus on LinkLabel Control.

Events

Event	Purpose
Link Clicked	It is the default event of LinkLabel Control. It fires each time a user click on a hyperlink of LinkLabel Control.

Button Control

- Button is a widely used control in window application.
- It is used to perform an action.
- The user clicks on a Button the click event associated with the Button is fired and the action associated with the event is executed.

Properties

Property	Purpose

BackColor	It is used to get or set background color of the Button.
Font	It is used to set Font Face, Font Style, Font Size and Effects of the text associated with Button Control.
ForeColor	It is used to get or set Fore color of the text associated with Button Control.
Enabled	It is used to specify weather Button Control is enabled or not at run time. It has Boolean value. Default value is true.
FlatStyle	It is used to get or set appearance of the Button Control when user moves mouse on it or click on it. It has following 4 options: System, Popup, Standard, Flat
Image	It is used to specify an image that is displayed in Button Control.
ImageAlign	It is used to get or set alignment of the image that is displayed in the Button control.
Text	It is used to get or set text associated with the Button Control.
TextAlign	It is used to get or set alignment of the text associated with the Button control.
Visible	It is used to specify weather Button Control is visible or not at run time. It has Boolean value. Default value is true.
TextImageRelation	It is used to get or set position of text in relation with image. It has following 5 options: (1) Overlay (2) ImageAboveText (3) TextAboveImage (4) ImageBeforeText (5) TextBeforeImage It is used when user wants to display both text and image on Button Control.
TabStop	It is used to specify weather user can use TAB key to set focus on Button Control or not. It has Boolean value. Default value is true.

Methods

Method	Purpose
Show	It is used to show Button control at run time.

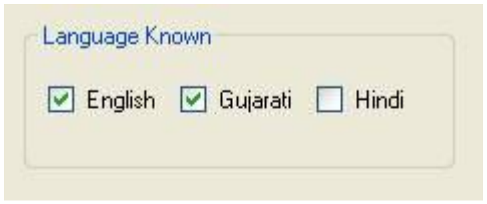
Hide	It is used to hide Button control at run time.
Focus	It is used to set input focus on Button Control at run time.

Event

Event	Purpose
Click	It is the default event of Button Control. It fires each time user clicks on Button Control.

CheckBox

- Check box control is used to represents list of options to the user from which user can select none or any number of options.
- Check Box control allow user to select or deselect particular option.
- Each time user clicks on the Check Box its status get changed. When check box is selected it has a checked status and when it is not selected it has Unchecked status.



Properties

Property	Purpose
AutoCheck	It is used to specify weather CheckBox can automatically change its state or not when it is clicked. It has Boolean value. Default value is true.
CheckAlign	It is used to get or set alignment of the ☑ within CheckBox Control.
Checked	It is used to get or set Boolean value, which indicates weather CheckBox Control is currently selected or not. It has Boolean value. True means selected. Default value is false.
CheckState	It is used to get or set current state of the CheckBox Control. It can have one of the following three values: Checked, Unchecked, Indeterminate

ThreeState	It is used to specify weather CheckBox Control allows three check states instead of two check states or not. It has a Boolean value. Default value is false.
Text	It is used to get or set text associated with CheckBox Control.
TextAlign	It is used to get or set alignment of the text that is associated with CheckBox Control.
Enable	It is used to specify weather CheckBox Control is enabled or not. It has Boolean value. Default value is true.
Visible	It is used to specify weather CheckBox Control is visible or not at run time. It has Boolean value. Default value is true.
BackColor	It is used to get or set background color of the CheckBox.
ForeColor	It is used to get or set color of the text associated with CheckBox.
Font	It is used to get or set font Style, Font Size, Font Face of the text associated with CheckBox.
TabStop	It is used to specify weather user can use TAB key to set focus on CheckBox Control or not. It has Boolean value. Default value is true.

Methods

Method	Purpose
Show	It is used to show CheckBox Control at run time.
Hide	It is used to Hide CheckBox Control at run time.
Focus	It is used to set cursor or focus on CheckBox Control.

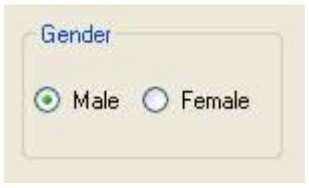
Events

Event	Purpose
CheckedChanged	It is the default event of CheckBox Control. It fires each time the value of checked property is changed.
CheckStateChanged	It fires each time the value of CheckState property is changed.

Radio Button

- RadioButton Control is used to represent list of options to the user from which user can select only one option.

- Every RadioButton that are placed on the form are treated as a single group, from which user can select only one option.



Properties

Property	Purpose
AutoCheck	It is used to specify weather RadioButton can automatically change its state or not when it is clicked. It has Boolean value. Default value is true.
CheckAlign	It is used to get or set alignment of the ◉ within RadioButton Control.
Checked	It is used to get or set current state of the RadioButton Control. It means weather RadioButton Control is currently selected or not. It has Boolean value. True means selected. Default value is false.
Text	It is used to get or set text associated with RadioButton Control.
TextAlign	It is used to get or set alignment of the text that is associated with RadioButton Control.
Enable	It is used to specify weather RadioButton Control is enabled or not. It has Boolean value. Default value is true.
Visible	It is used to specify weather RadioButton Control is visible or not at run time. It has Boolean value. Default value is true.
BackColor	It is used to get or set background color of the RadioButton.
ForeColor	It is used to get or set color of the RadioButton's text.
Font	It is used to get or set font Style, Font Size, Font Face of the RadioButton's text.
TabStop	It is used to specify weather user can use TAB key to set focus on RadioButton Control or not. It has Boolean value. Default value is true.

Methods

Method	Purpose
PerformClick	It is used to fire Click event of RadioButton Control.
Show	It is used to show RadioButton Control.
Hide	It is used to Hide RadioButton Control at run time.

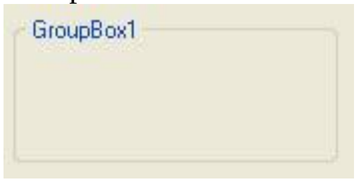
Focus	It is used to set cursor or focus on RadioButton Control.

Events

Event	Purpose
CheckedChanged	It is the default event of RadioButton Control. It fires each time the value of checked property is changed.

GroupBox

- GroupBox control is used to group other controls of VB.NET.
- GroupBox control having a frame to indicate boundary and a text to indicate header or title.
- Generally GroupBox control is used as a container for Radio Button. When Radio Buttons are grouped using GroupBox, user can select one RadioButton from each GroupBox.



Properties of GroupBox Control in VB.NET

Property	Purpose
BackColor	It is used to get or set background color of the GroupBox.
BackgroundImage	It is used to get or set background Image of the GroupBox.
BackgroundImageLayout	It is used to get or set background Image layout of the GroupBox. It has one of the following values: None, Tile, Centre, Stretch, Zoom
Font	It is used to get or set font Style, Font Size, Font Face of the text contained in GroupBox Control.
ForeColor	It is used to get or set color of the text contained in GroupBox Control.
Enabled	It is used to specify weather GroupBox Control is enabled or not. It has Boolean value. Default value is true.

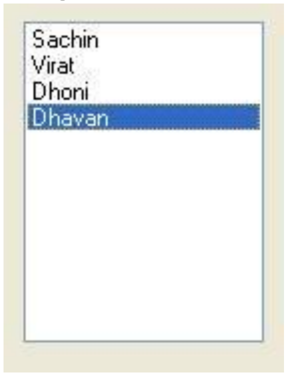
Visible	It is used to specify weather GroupBox Control is visible or not at run time. It has Boolean value. Default value is true.
Text	It is used to get or set Title or Header Text of the GroupBox Control.

Methods

Method	Purpose
Show	It is used to show GroupBox Control.
Hide	It is used to Hide GroupBox Control at run time.
Focus	It is used to set cursor or focus on GroupBox Control.

Listbox

- ListBox Control is a rectangular box type structure which allows user to display list of items from which user can select one or more items at a time.
- It also allows user to sort items of the list in particular order.
- The User can add new items, remove selected or all items from ListBox at run time or design time.



Properties

Property	Purpose
MultiColumn	It is used to specify weather ListBox supports multiple columns or not. It has Boolean value. Default value is false.
ColumnWidth	It is used to specify width of each column in MultiColumn ListBox.
Items	It represents collection of items contained in ListBox control.

Sorted	It is used to specify weather items of ListBox are sorted in alphabetical order or not. It has Boolean value. Default value is false.
SelectionMode	It is used to get or set SelectionMode of ListBox. It determines how user can select the Items of ListBox. It can have one of the following four options: **(1) None:** No Selection is allowed **(2) One:** User can select only one item at a time. **(3) MultiSimple:** User can select or deselect item just by mouse click or pressing spacebar. **(4) MultiExtended:** User can select or deselect items by holding Ctrl key and mouse click. User can also select or deselect items by pressing Shift key and mouse click or arrow key. Default value is One.
ScrollAlwaysVisible	It is used to specify weather Scroll Bars is always associated with ListBox or Not regardless of number of items present in ListBox. It has Boolean value. Default value is false.
HorizontalExtent	It is used to get or set width in pixel, by which a ListBox can scrolled horizontally. It works only when Horizontal Scrollbar Property is set to true.
HorizontalScrollbar	It is used to specify weather ListBox can have Horizontal Scroll Bar or Not, If Number of Items in ListBox are not accommodate in specified width. It has Boolean value. Default value is False.
SelectedIndex	It is used to get or set zero based index of the item currently selected in ListBox.
SelectedItem	It is used to get or set item currently selected in ListBox.
SelectedItems	It is used to get collection of multiple items currently selected in ListBox.
SelectedIndices	It is used to get collection of zero based indexes of all items currently selected in ListBox.
Enable	It is used to specify weather ListBox Control is enabled or not at runtime. It has Boolean value. Default value is true.

Visible	It is used to specify weather ListBox Control is visible or not at runtime. It has Boolean value. Default value is true.
TabStop	It is used to specify weather user can use TAB key to set focus on ListBox Control or not. It has Boolean value. Default value is true.

Methods

Method	Purpose
ClearSelected	It is used to unselect all the items that are currently selected in ListBox.
FindString	It is used to find first occurrences of an item in the ListBox that partially match with string specified as an argument. If an item is found than it returns zero based index of that item, otherwise it returns -1. The search performed by this method is case insensitive.
FindStringExact	It is used to find first occurrences of an item in the ListBox that exactly match with string specified as an argument. If an item is found than it returns zero based index of that item, otherwise it returns -1. The search performed by this method is case insensitive.
GetSelected	It is used to determine weather an item whose index is passed as an argument is selected or not. It returns Boolean value.
SetSelected	It is used to select or deselect an item whose index is passed as an argument. **Example:** ListBox1.SetSelected (1, true) will select second item of ListBox.

Events

Event	Purpose
SelectedIndexChanged	It is the default event of ListBox Control. It fires each time a selected Item in the ListBox is changed.

Methods

Method	Purpose
Add	It is used to add an item at the end of ListBox collection. **Example:** ListBox1.Items.Add(Item)

Insert	It is used to insert an item at specific index position. **Example:** ListBox1.Items.Insert(Index, Item)
Clear	It is used to remove all items from ListBox collection. **Example:** ListBox1.Items.Clear()
CopyTo	It is used to copy all the Items of ListBox into an array that is passed as an argument. It also accepts an index position as an argument to specify from which index position within array copy will start. **Example:** ListBox1.Items.CopyTo(ArrayName, StartIndex)
Remove	It is used to remove first occurrence of an item that matches with value passed as an argument. **Example:** ListBox1.Items.Remove(Value)
RemoveAt	It is used to remove an item from specific index position in ListBox. **Example:** ListBox1.RemoveAt (1) will remove second Item of ListBox.
Contains	It is used to determine weather a ListBox's Item Collection contains an item that is passed as an argument. It returns Boolean value. If an item is found then it returns true otherwise false. **Example:** If ListBox1.Items.Contains (Value) Then MsgBox("Item is Found") Else MsgBox("Item is Not Found") End If

Properties

Property	Purpose
Count	It is used to get number of items available in ListBox's Item Collection.

CheckedListBox

- CheckedListBox is a ListBox with Checkbox to the left side of each item in the list.
- It is derived from ListBox so it provides all the functionality of ListBox Control.

Properties

Property	Purpose
CheckOnClick	It is used to specify weather CheckBox should be toggled (change state) or not when an item is selected in the CheckedListBox. It has Boolean value. Default value is False.
MultiColumn	It is used to specify weather CheckedListBox supports multiple columns or not. It has Boolean value. Default value is false.
ColumnWidth	It is used to specify width of each column in MultiColumn CheckedListBox.
Items	It represents collection of items contained in CheckedListBox control.
Sorted	It is used to specify weather items of CheckedListBox are sorted in alphabetical order or not. It has Boolean value. Default value is false.
SelectionMode	It is used to get or set SelectionMode of CheckedListBox. It determines how user can select the Items of CheckedListBox. It can have one of the following four options: **(1) None:** No Selection is allowed **(2) One:** User can select only one item at a time. **(3) MultiSimple:** User can select or deselect item just by mouse click or pressing spacebar. **(4) MultiExtended:** User can select or deselect items by holding Ctrl key and mouse click. User can also select or deselect items by pressing Shift key and mouse click or arrow key. Default value is One.
ScrollAlwaysVisible	It is used to specify weather Scroll Bars is always associated with CheckedListBox or Not regardless of number of items present in CheckedListBox. It has Boolean value. Default value is false.
HorizontalExtent	It is used to get or set width in pixel, by which a CheckedListBox can scrolled horizontally. It works only when Horizontal Scrollbar Property is set to true.

HorizontalScrollbar	It is used to specify weather CheckedListBox can have Horizontal Scroll Bar or Not, If Number of Items in CheckedListBox are not accommodate in specified width. It has Boolean value. Default value is False.
ThreeDCheckBoxes	It is used to get or set value which determines CheckBox has Flat or Normal Button State. It has Boolean value. Default value is false. When it is true check box has flat button state.
SelectedIndex	It is used to get or set zero based index of the item currently selected in CheckedListBox.
SelectedItem	It is used to get or set item currently selected in CheckedListBox.
SelectedItems	It is used to get collection of multiple items currently selected in CheckedListBox.
CheckedItems	It is used to get collection of multiple items currently checked in CheckedListBox.
SelectedIndices	It is used to get collection of zero based indexes of all items currently selected in CheckedListBox.
CheckedIndices	It is used to get collection of zero based indexes of all items currently checked in CheckedListBox.
Enable	It is used to specify weather CheckedListBox Control is enabled or not at runtime. It has Boolean value. Default value is true.
Visible	It is used to specify weather CheckedListBox Control is visible or not at runtime. It has Boolean value. Default value is true.
TabStop	It is used to specify weather user can use TAB key to set focus on CheckedListBox Control or not. It has Boolean value. Default value is true.

Methods

Method	Purpose
ClearSelected	It is used to unselect all the items that are currently selected in ListBox.
FindString	It is used to find first occurrences of an item in the ListBox that partially match with string specified as an argument. If an item is found than it returns zero based index of that item, otherwise it returns -1. The search performed by this method is case insensitive.

FindStringExact	It is used to find first occurrences of an item in the ListBox that exactly match with string specified as an argument. If an item is found than it returns zero based index of that item, otherwise it returns -1. The search performed by this method is case insensitive.
GetSelected	It is used to determine weather an item whose index is passed as an argument is selected or not. It returns Boolean value.
SetSelected	It is used to select or deselect an item whose index is passed as an argument. **Example:** ListBox1.SetSelected (1, true) will select second item of ListBox.
ClearSelected	It is used to unselect all items in CheckedListBox.
GetItemChecked	It is used to check weather an item whose index is passed as an argument is checked or not. It returns Boolean value.
GetItemCheckState	It is used to get check state of an item whose index is passed as an argument. It returns 1 if item is checked otherwise false.
SetItemCheckState	It is used to set the check state of an item whose index is passed as an argument. **Example:** CheckedListBox1.SetItemChecked (1, CheckState.Checked) will check the second item of CheckedListBox.

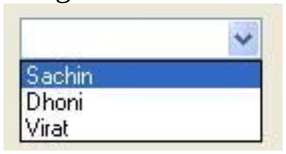
Events

Event	Purpose
SelectedIndexChanged	It is the default event of ListBox Control. It fires each time a selected Item in the ListBox is changed.
ItemCheck	It fires each time an item is checked or unchecked.

Combo Box

- ➢ ComboBox control represents list of items in a drop down list type structure from which user can select only one item at a time.
- ➢ The User can drop down the list to display list of items in a ComboBox.
- ➢ ComboBox control comes with built in TextBox so user can enter new item if it is not available in the list of ComboBox.

- ComboBox is similar to a ListBox but it does not allow user to select more then one items.
- It occupies less space on the form because of its drop down structure.
- The User can add new items, remove selected or all items from ComboBox at runtime or design time.



Properties

Property	Purpose
AutoComplete Mode	It is used to get or set value which determines how AutoComplete option works for ComboBox. It has four options: Suggest, Append, SuggestAppend, None Default value is None.
AutoCompleteSource	It is used to get or set Auto Complete Source for ComboBox.
AutoCompleteCustomSource	It is used to specify Custom Source by defining list of items in it. It works when AutoCompleteSource is set to CustomSource.
DropDownStyle	It is used to get or set value which determines appearance and behavior style of ComboBox. It has three options: **(1)DropDown:** ComboBox with built in TextBox and Drop Down List. **(2)DropDownList:** ComboBox with only DropDown List. **(3) Simple:** ComboBox with expanded DropDown List and built in TextBox.
DropDownWidth	It is used to get or set width of DropDown list in pixel.
DropDownHeight	It is used to get or set height of DropDown list in pixel.
MaxDropDownItems	It is used to get or set value which determines how many items should be display in dropdown list. For remaining items to display you need to scroll the dropdown list. Default value is 8.
MaxLength	It is used to get or set value which determines maximum number of characters that can be entered in built in TextBox of ComboBox.

Text	It is used to get or set text associated with built in TextBox of ComboBox.
Items	It represents collection of items contained in ComboBox control.
Sorted	It is used to specify weather items of ComboBox are sorted in alphabetical order or not. It has Boolean value. Default value is false.
SelectedIndex	It is used to get or set zero based index of the item currently selected in ComboBox.
SelectedItem	It is used to get or set item currently selected in ComboBox.
SelectionStart	It is used to get or set starting index of the text that is entered in the built in TextBox of ComboBox.
Selectionlength	It is used to get or set number of characters that is selected in the built in TextBox of ComboBox.
SelectedText	It is used to get or set text that is selected in built in TextBox of ComboBox.
Enable	It is used to specify weather ComboBox Control is enabled or not at runtime. It has Boolean value. Default value is true.
Visible	It is used to specify weather ComboBox Control is visible or not at runtime. It has Boolean value. Default value is true.
TabStop	It is used to specify weather user can use TAB key to set focus on ComboBox Control or not. It has Boolean value. Default value is true.

Methods

Method	Purpose
FindString	It is used to find first occurrences of an item in the ComboBox that partially match with string specified as an argument. If an item is found than it returns zero based index of that item, otherwise it returns -1. The search performed by this method is case insensitive.
FindStringExact	It is used to find first occurrences of an item in the ComboBox that exactly match with string specified as an argument. If an item is found than it returns zero based index of that item, otherwise it returns -1. The search performed by this method is case insensitive.

SelectAll	It is used to select all the text in the built in TextBox of ComboBox.
Select	It is used to select all or specific range of the text in the built in TextBox of ComboBox. **Syntax:** ComboBox1.Select(Start, Length) **Example:** ComboBox1.Select (1,4) It will select 4 characters starting from second character in the built in textbox of ComboBox.

Events

Event	Purpose
SelectedIndexChanged	It is the default event of ComboBox Control. It fires each time a selected Item in the ListBox is changed.
TextChanged	It fires each time a text is changed in the built in TextBox of ComboBox Control.
DropDown	This event fires each time a DropDown list is displayed by clicking on DropDown arrow.
DropDownClosed	This event fires each time a DropDown list is disappeared after selecting particular Item.

Picturebox

- ➢ PictureBox control allows user to display an image on the form.
- ➢ The User can display image either design time or run time using Picture Box Control.

Properties

Property	Purpose
Image	It is used to get or set an image to be display in the PictureBox.
ErrorImage	It is used to get or set an Error Image that display when loading of Image specified in Image property is failed.

Methods

InitialImage	It is used to get or set an initial Image that display when an Image specified in Image property is not loaded.
ImageLocation	It is used to get or set path or url of image to be display in PictureBox Control.
SizeMode	It is used to get or set the SizeMode. SizeMode determines how an image will display in PictureBox. It has five options: Normal, Stretch Image, Auto Size, Center Image, Zoom
WaitOnLoad	It is used to specify weather processing of an image is stop until it is loaded or not. It has Boolean value. Default value is False.
Enabled	It is used to specify weather PictureBox Control is enabled or not at runtime. It has Boolean value. Default value is true.
Visible	It is used to specify weather PictureBox Control is visible or not at runtime. It has Boolean value. Default value is true.

Method	Purpose
Load	It is used to display an image whose path or url is specified in ImageLocation Property of PictureBox.
LoadAsync	It is used to load an image Asynchronously.
CancelAsync	It is used to cancel Asynchronous loading of an image.

Events

Event	Purpose
Click	It is the default event of PictureBox. It fires each time user clicks on PictureBox.
SizeModeChanged	It fires each time a SizeMode property of the PictureBox is changed.

Timer Control in VB.NET

- Timer Control is used when user wants to perform some task or action continuously at regular interval of time.

Properties

Property Name	Description
Name	It is used to specify name of the Timer Control.
Enabled	It is used to determine weather Timer Control will be enabled or not. It has boolean value true or false. Default value is false.
Interval	It is used to specify interval in millisecond. Tick event of Timer Control generates after the time which is specified in Interval Property.

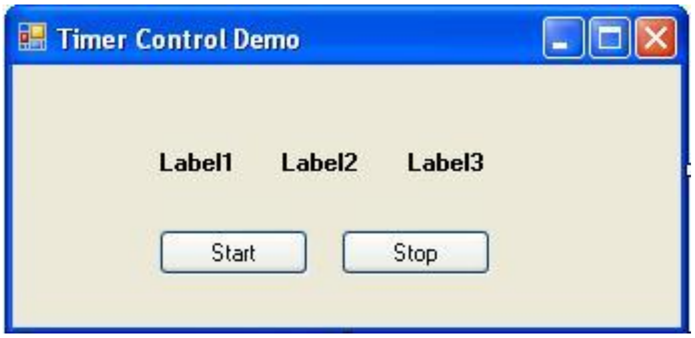
Methods

Method Name	Description
Start	This method is used to start the Timer Control.
Stop	This method is used to stop the Timer Control.

Events of Timer Control

Event Name	Description
Tick	Tick event of the Timer Control fires continuously after the time which is specified in the Inteval property of Timer Control.

Example



Now set Properties of various control as below:

Control Name	Property Name	Value
Form1	Text	Timer Control Demo
Label1	Name	lblHour

Label2	Name	lblMinute
Label3	Name	lblSecond
Button1	Name	btnStart
	Text	Start
Button2	Name	btnStop
	Text	Stop
Timer1	Name	Timer1
	Enabled	true
	Interval	1000

Now double click on the Timer Control and write following code in the **Tick event** of Timer Contro.

```
lblHour.Text = Now.Hour
lblMinute.Text = Now.Minute
lblSecond.Text = Now.Second
```

Now double click on the Start Button and write following code in the **Click event** of Button.

```
Timer1.Start()
```

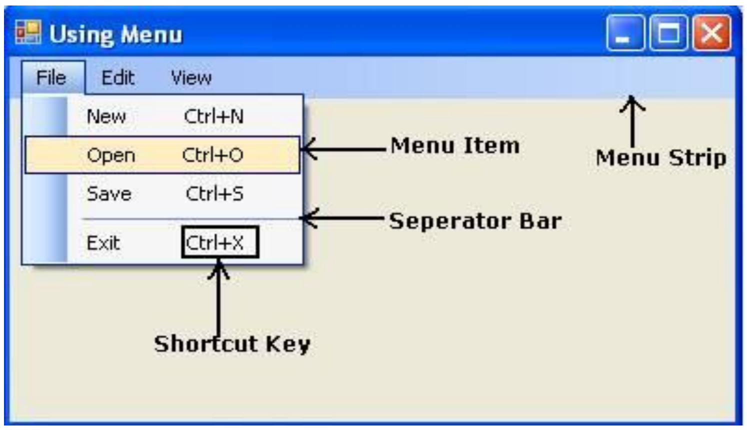
Now double click on the Stop Button and write following code in the **Clcik event** of Button.

```
Timer1.Stop()
```

Menu in VB.NET

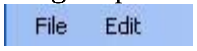
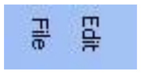
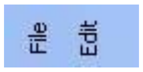
- Menu is one of the most common elements of Graphical User Interface.
- Menu is a one type of control that represents a group of choices to the user and allows user to select any of them according to their requirement.
- Using menu user can organizes various options or commands as per their functionality
- It helps programmer to organize large number of options in a short and easy way.

- It can be attached only with form either SDI or MDI.
  It is displayed immediately under the title bar of the form as shown below:



Properties

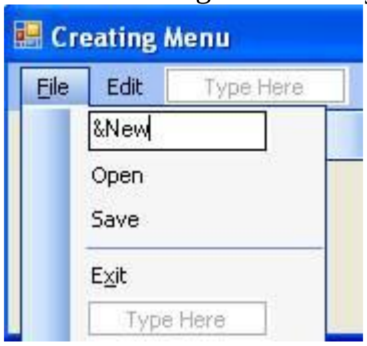
Property	Purpose
AllowItemReorder	It is used to specify weather user can reorder menu items by holding Alt key or not. It has Boolean value. Default value is false.
BackColor	It is used to get or set back color of the MenuStrip.
Enabled	It is used to specify weather MenuStrip is enabled or not at run time. It has Boolean value. Default value is true.
Font	It is used to set Font Face, Font Style, Font Size and Effects of the text associated with Menu Items of MenuStrip Control.
Items	It represents collection of Menu Items contained in Menu Strip control.
LayoutStyle	It is used to get or set Layout Style of Menu Strip Control. It has following 5 options: (1) Stack with Overflow (2) Horizontal Stack With Overflow (3) Vertical Stack With Overflow (4) Flow (5) Table
ShowItemToolTips	It is used to specify weather Tool Tip text will be displayed for each menu item or not when mouse is over that menu item. It has Boolean value. Default value is false.

TextDirection	It is used to get or set value which determines direction of text in each menu Item. It has following 3 options: (1) Horizontal:  (2) Vertical90:  (3) Vertical270: 
Visible	It is used to specify weather MenuStrip is visible or not at run time. It has Boolean value. Default value is true.

Assigning Access key to Menu Items

Access key allows user to select menu item from the keyboard using Alt key.

Access key is combinations of Alt key and other key (Alt + Other key).
In order to select particular Menu Item User has to press Alt key and then press other key which is defined as access key.
In order to assign Access key to Menu Item just precede the character by & as shown below:



In above figure character N is defined as Access key for Menu Item New.

The User can access Menu Item New by pressing Alt + N key.
A character which is defined as an access key for menu item is displayed with underline.
Hence & is used to assign Access Key to particular character in Menu Item, we cannot display & in Menu Item directly.

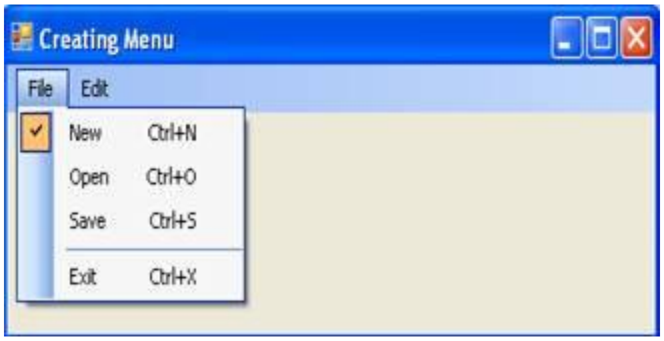
In order to display & in Menu Item we have to precede it with another &. For Example to display Save & Close in Menu Item we have to write Save && Close in text property of Menu Item.

Assigning Shortcut Key to Menu Item

Shortcut Key allows user to perform action associated with particular Menu Item using keyboard. Thus using concept of Shortcut Key user can fire action associated with particular Menu Item using keyboard.

Shortcut Key allows user to perform action with a single keystroke.

Shortcut Key is a combination of (Alt, Shift, Ctrl) key and other key as shown in the figure below:

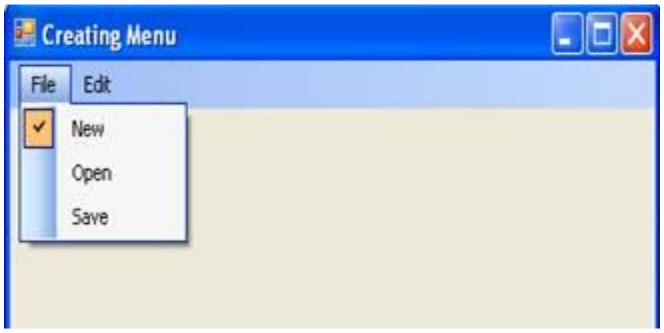


In order to assign Shortcut Key to particular Menu Item following properties are used:

Property	Purpose
ShortcutKeys	It is used to assign Shortcut Key to Menu Item.
ShowShortcutKeys	It is used to specify weather Shortcut Key is displayed beside Menu Item or not. It has Boolean value. Default value is False.
ShortcutKeyDisplayString	It is used to get or set string that is display instead of Shortcut Key.

Adding checkmarks to Menu Item

User can add checkmark to the left side of Menu Item to indicate that the Menu Item is selected or not.



In order to add checkmark to particular menu item following properties are used:

Property	Purpose
Checked	It is used to specify weather checkmark will be displayed to the left side of Menu item or Not. It has Boolean value. Default is False.
CheckOnClick	It is used to specify weather Menu Item will toggle (change) its state or not when it is clicked. It has Boolean value. Default value is false.
CheckState	It is used to get or set state of menu item. It can have one of the following 3 values: (1) Checked (2) Unchecked (3) Indeterminate Default value is Unchecked.

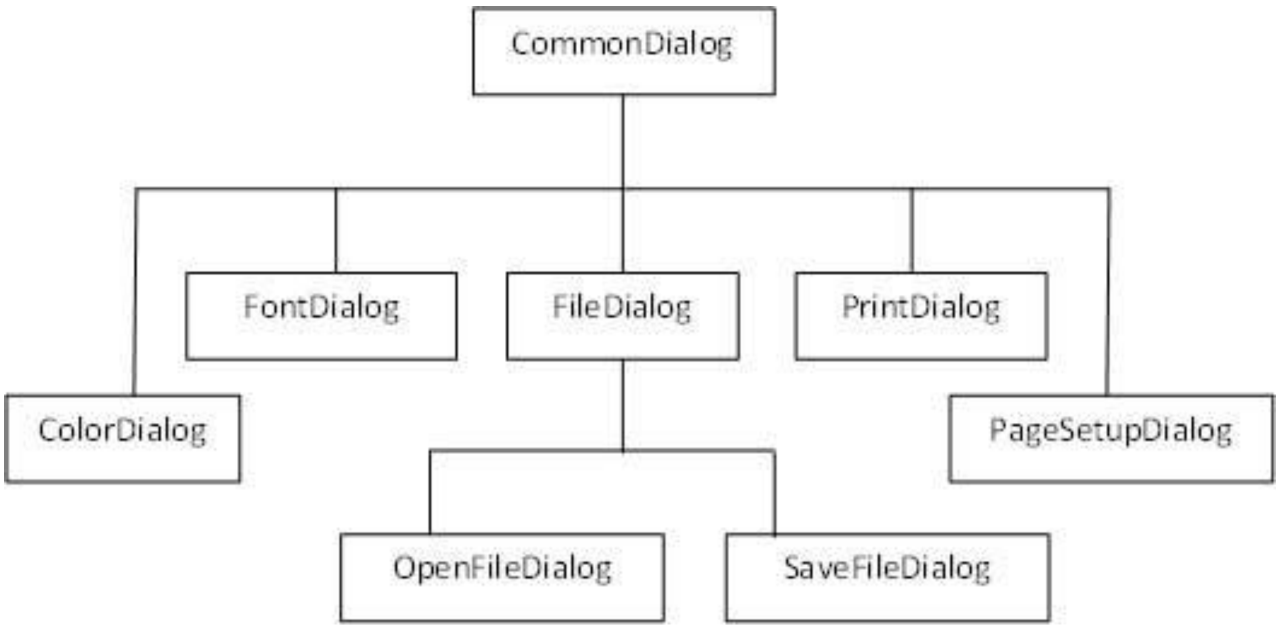
Dialog Boxes

- ➢ There are many built-in dialog boxes to be used in Windows forms for various tasks like opening and saving files, printing a page, providing choices for colors, fonts, page setup, etc., to the user of an application. These built-in dialog boxes reduce the developer's time and workload.
- ➢ All of these dialog box control classes inherit from the **CommonDialog** class and override the *RunDialog()* function of the base class to create the specific dialog box.
- ➢ The RunDialog() function is automatically invoked when a user of a dialog box calls its *ShowDialog()* function.

The **ShowDialog** method is used to display all the dialog box controls at run-time. It returns a value of the type of **DialogResult** enumeration. The values of DialogResult enumeration are −

- **Abort** − returns DialogResult.Abort value, when user clicks an Abort button.
- **Cancel** − returns DialogResult.Cancel, when user clicks a Cancel button.
- **Ignore** − returns DialogResult.Ignore, when user clicks an Ignore button.
- **No** − returns DialogResult.No, when user clicks a No button.
- **None** − returns nothing and the dialog box continues running.
- **OK** − returns DialogResult.OK, when user clicks an OK button
- **Retry** − returns DialogResult.Retry , when user clicks an Retry button
- **Yes** − returns DialogResult.Yes, when user clicks an Yes button

The following diagram shows the common dialog class inheritance −



Open File Dialog Control

Open File Dialog Control allows user to select a file for open.

Methods of Open File Dialog Control in VB.NET

Method	Purpose
ShowDialog	It is used to Show or run OpenFileDialog Control.
Reset	It is used to reset all the properties of OpenFileDialog to its default values.
OpenFile	It is used to open the file which is selected by user in read only mode.

Events of Open File Dialog Control in VB.NET

Event	Purpose
FileOk	It is the default event of OpenFileDialog Control. It fires each time user clicks on Open button of OpenFileDialog Control. It is used to perform specific task when user click on Open button.

Save File Dialog Control in VB.NET

SaveFileDialog Control allows user to:
(1) Specify Location where to save the file.
(2) Specify Name of File by which it is saved.

Properties of Save File Dialog Control in VB.NET

Property	Purpose
AddExtension	It is used to specify weather default extension will be automatically added at the end of file name or not. It has Boolean value. Default value is true.
CheckFileExists	It is used to specify weather warning message will display or not if user select a file that does not exist. It has Boolean value. Default value is true.
CheckPathExists	It is used to specify weather warning message will display or not if user specified path does not exist. It has Boolean value. Default value is true.
CreatePrompt	It is used to specify weather SaveFileDialog Control will display warning message or not when user is about to create a new file that does not exist. It has Boolean value. Default value is false.
DefaultExt	It is used to specify default extension for file name. Default extension is appended at the end of file name if user selects file with no extension.
FileName	It represents full path of the file that is selected by user in the SaveFileDialog Control.
Filter	It is used to specify which type of files will be display in the SaveFileDialog Control. If user wants to display only executable files than user can set Filter Property to **Executable File | *.exe** If user wants to display executable files and Image Files than user can set Filter Property to **Executable File | *.exe | Image Files| *.jpeg**
FilterIndex	It is used to specify which File Type to be displayed in SaveFileDialog Control by default. It is used when user specify more than one File Types in Filter Property. In such situation the first File Type in Filter property becomes default File Type and it has FilterIndex 1, the second File Type has FilterIndex 2 and So on.

InitialDirectory	It is used to specify initial directory for SaveFileDialog Control. Initial Directory means the folder whose files are displayed by the SaveFileDialog Control when it opens.
OverwritePrompt	It is used to specify weather SaveFileDialog Control will display warning message or not when user is about to overwrite already existing file. It has Boolean value. Default value is true.
Title	It is used to specify the text to be display in the title bar of the SaveFileDialog Control.

Methods of Save File Dialog Control in VB.NET

Method	Purpose
ShowDialog	It is used to Show or run SaveFileDialog Control.
Reset	It is used to reset all the properties of SaveFileDialog to its default values.
OpenFile	It is used to open the file which is selected by user in read/write mode.

Events of Save File Dialog Control in VB.NET

Event	Purpose
FileOk	It is the default event of SaveFileDialog Control. It fires each time user clicks on Save button of SaveFileDialog Control. It is used to perform specific task when user click on Save button.

Color Dialog Control in VB.NET

Color Dialog Control allows user to select color from the list of available colors. User can also define custom colors using Color Dialog control.

Properties of Color Dialog Control in VB.NET

Property	Purpose
Color	It is used to get or set the color selected by the user in Color Dialog Control. It is also used set specific color in the Color Dialog Control.
FullOpen	It is used to specify weather Custom Color Section of the Color Dialog Control is by default displayed or not. It has Boolean value. Its default value is false.

AllowFullOpen	It is used to enable or disable **Define Custom Color** button In Color Dialog Control. It has Boolean value. Its default value is true. User can see effect of this property only when FullOpen property is set to false.
AnyColor	It is used to specify weather Color Dialog will display all the available colors in the set of basic colors or not. It has Boolean value. Default value is False.
SolidColorOnly	It is used to specify weather Color Dialog will restrict user to select only solid colors or not. It has Boolean value. Default value is False.

Methods

Method	Purpose
ShowDialog	It is used to Show or run Color Dialog Control.
Reset	It is used to reset all the properties of ColorDialog to its default values.

Font Dialog

Font Dialog Control allows user to:

(1) Set Font Face

(2) Set Font Style

(3) Set Font Size

(4) Set Font Color

(5) Set Font Effects

Properties of Font Dialog Control in VB.NET

Property	Purpose
AllowScriptChange	It is used to specify weather character set (script) can be changed in Font Dialog Control or not. It has Boolean value. Default value is true.
AllowVectorFonts	It is used to specify weather vector font can be selected in Font Dialog Control or not. It has Boolean value. Default value is true.
AllowVerticalFont	It is used to specify weather Vertical font can be selected in Font Dialog Control or not. It has Boolean value. Default value is true.

MaxSize	It is sued to specify Maximum Font Size that user can select from Font Dialog Control.
MinSize	It is sued to specify Minimum Font Size that user can select from Font Dialog Control.
Color	It is used to get color selected by user in the Font Dialog Control. User can also set the color in Font Dialog control using this property.
FixedPitchOnly	It is used to specify weather only Fixed Pitch font can be selected in the Font Dialog Control or not. It has Boolean value. Default value is false.
FontMustExist	It is used to specify weather an error will occur or not, if user selects the font that does not exist in the selection box. It has Boolean value. Default value is false.
Font	It is used to get the font selected in the Font Dialog Control. User can also set the font style in the Font Dialog Control.
ShowApply	It is used to specify weather Apply button will be shown in the Font Dialog Control or not. It has Boolean value. Default value is false.
ShowColor	It is used to specify weather color selection combo box will be shown in the Font Dialog Control or not. It has Boolean value. Default value is false.
ShowEffects	It is used to specify weather font effect options such as Underline, Strikeout and color selection will be shown in the Font Dialog Control or not. It has Boolean value. Default value is true.

Methods of Font Dialog Control in VB.NET

Method	Purpose
ShowDialog	It is used to Show or run Font Dialog Control.
Reset	It is used to reset all the options of FontDialog to its default values.

Events

Event	Purpose
Apply	It is the default event of Font Dialog Control. It fires each time user clicks on Apply button of Font Dialog Control. It is used to apply font settings on selected text without closing Font Dialog Control.

Print Dialog

PrintDialog Control allows user to print document. It allows following facilities to user:

(1) Print Document

(2) Select printer

(3) Specify Page Range

(4) Specify Number of copies

(5) Find Printer on Network

Properties

Property	Purpose
AllowCurrentPage	It is used to specify weather Current Page radio button is enabled or disabled in PrintDialog Control. It has Boolean value. Default value is false. Current Page option allows user to print only current page in which cursor is set.
AllowSelection	It is used to specify weather Selection radio button is enabled or disabled in PrintDialog Control. It has Boolean value. Default value is false. Selection option allows user to print only selected portion of the document.
AllowSomePages	It is used to specify weather Pages radio button is enabled or disabled in PrintDialog Control. It has Boolean value. Default value is false. Pages option allows user to specify range of pages to be print from document. For Example: 3-7 will print page number 3 to 7 from entire document. 1,3,5,8 will print Page Number 1, 3, 5 and 8.
AllowPrintToFile	It is used to specify weather **Print to File** Check Box is enabled or disabled in PrintDialog Control. It has Boolean value. Default value is true.
PrintToFile	It is used to specify weather **Print to File** Check Box is by default selected or not in PrintDialog Control. It has Boolean value. Default value is false.
ShowNetwork	It is used to specify weather Show Network button is displayed in PrintDialog Control or not. It has Boolean value. Default value is true.

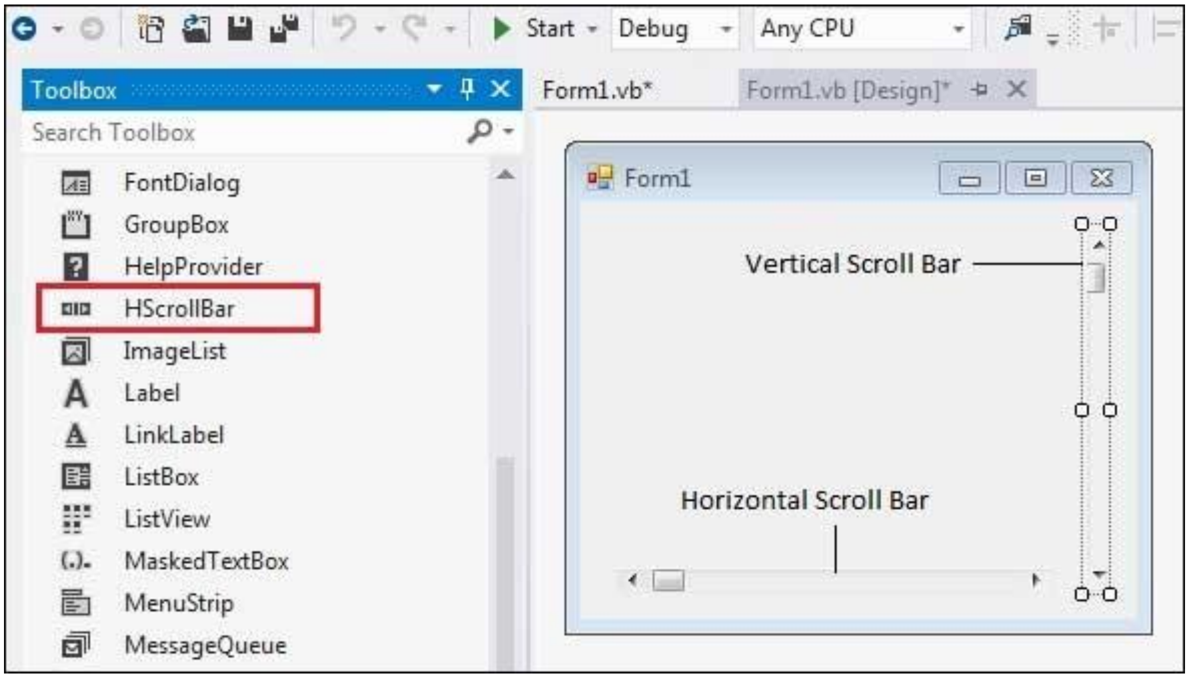
Methods

Method	Purpose
ShowDialog	It is used to Show or run Print Dialog Control.
Reset	It is used to reset all the options of PrintDialog to its default values.

ScrollBar controls

- The ScrollBar controls display vertical and horizontal scroll bars on the form.
- This is used for navigating through large amount of information.
- There are two types of scroll bar controls:
- **HScrollBar** for horizontal scroll bars and **VScrollBar** for vertical scroll bars.
- These are used independently from each other.

Let's click on HScrollBar control and VScrollBar control from the Toolbox and place them on the form.



Properties

Sr.No.	Property & Description
1	**AutoSize** Gets or sets a value indicating whether the ScrollBar is automatically resized to fit its contents.

2	**BackColor** Gets or sets the background color for the control.
3	**ForeColor** Gets or sets the foreground color of the scroll bar control.
4	**ImeMode** Gets or sets the Input Method Editor (IME) mode supported by this control.
5	**LargeChange** Gets or sets a value to be added to or subtracted from the Value property when the scroll box is moved a large distance.
6	**Maximum** Gets or sets the upper limit of values of the scrollable range.
7	**Minimum** Gets or sets the lower limit of values of the scrollable range.
8	**SmallChange** Gets or sets the value to be added to or subtracted from the Value property when the scroll box is moved a small distance.
9	**Value** Gets or sets a numeric value that represents the current position of the scroll box on the scroll bar control.

Methods

Sr.No.	Method Name & Description
1	**OnClick** Generates the Click event.
2	**Select** Activates the control.

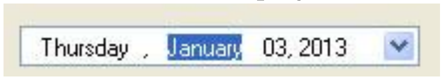
Events

Sr.No.	Event & Description
1	**Click** Occurs when the control is clicked.
2	**DoubleClick** Occurs when the user double-clicks the control.

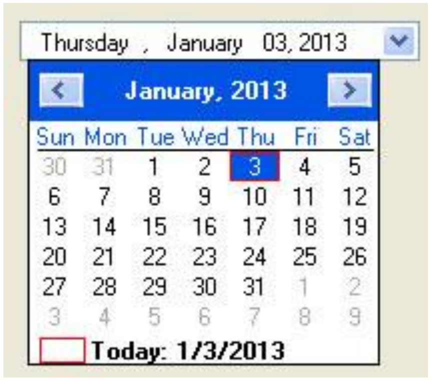
3	**Scroll** Occurs when the control is moved.
4	**ValueChanged** Occurs when the Value property changes, either by handling the Scroll event or programmatically.

Datetimepicker Control 0r calendar control

- DateTimePicker Control allows user to select a date and a time.
- It is also used to display selected date and time in specific format.



By default it displays current date in Long Date format. When user clicks on arrow button it displays full calendar for the current month as shown below:



Properties

Property	Purpose
Format	It is used to get or set format for displaying date and time in DateTimePicker Control. It has four options: Long, Short, Time, And Custom. Default format is long.
CustomFormat	It is used to get or set custom format for displaying date and time in DateTimePicker Control. **For Example:** dd/MM/yyyy will display 01/01/2014 d/M/yyyy will display 1/1/2014 d/M/yy will display 1/1/14 ddd, MMM, yyyy will display Wed, Jan, 2013 dddd, MMMM, yyyy will display

	Wednesday, January, 2014 hh:mm:ss will display 02:25:02 HH:mm:ss will display 14:25:02 hh:mm:ss tt will display 02:25:02 PM
Value	It is used to get date and time selected in DateTimePicker Control.
ShowCheckBox	It is used to specify weather CheckBox is displayed in DateTimePicker or not. It has Boolean value. Default value is false.
Checked	It is used to specify weather CheckBox in DateTimePicker is checked or not. It has Boolean value. Default value is true. It works only when ShowCheckBox property is set to true.
ShowUpDown	It is used to specify weather UpDown arrow is displayed in the DateTimePicker instead of DropDown Calendar or Not. It has Boolean value. Default value is false.
MinDate	It is used to get or set Minimum Date that can be selected using DateTimePicker Control. Default value is 01/01/1753.
MaxDate	It is used to get or set Maximum Date that can be selected using DateTimePicker Control. Default value is 31/12/9998.
Enabled	It is used to specify weather DateTimePicker Control is enabled or not at runtime. It has Boolean value. Default value is true.
Visible	It is used to specify weather DateTimePicker Control is visible or not at runtime. It has Boolean value. Default value is true.
CalendarFont	It is used to get or set Font of the Calendar.
CalendarForeColor	It is used to get or set ForeColor of the calendar font.
CalendarMonthBackground	It is used to get or set Background Color of the Calendar.
CalendarTitleBackColor	It is used to get or set Background Color of Calendar’s Title.
CalendarTitleForeColor	It is used to get or set ForeColor of the font that is displayed in Calendar’s Title.
CalendarTrailingForeColor	It is used to get or set Fore Color of the previous and next month’s date that is displayed in Current Month’s Calendar.

Methods

Method	Purpose
Show	It is used to show DateTimePicker Control at run time.
Hide	It is used to Hide DateTimePicker Control at run time.
Focus	It is used to set cursor or focus on DateTimePicker Control.

Events

Event	Purpose
ValueChanged	It is the default event of DateTimePicker Control. This event fires each time a value in the DateTimePicker Control is changed.
TextChanged	This event fires each time a text that is displayed in the DateTimePicker Control is changed.
CloseUp	This event fires each time a Popup Calendar is disappeared after selecting particular date.
DropDown	This event fires each time a Popup Calendar is displayed by clicking on DropDown arrow.

Unit – IV ASP .NET

Intro in ASP.NET

- ASP.NET is a web framework designed and developed by Microsoft.
- It was first released in January 2002.
- It is used to develop websites, web applications and web services.
- It provides fantastic integration of HTML, CSS and JavaScript.
- It is built on the Common Language Runtime (CLR) and allows programmers to write code using any supported .NET language.

ASP.NET Features

ASP.NET is full of features and provides a platform to create and develop web application. The following features of Web Forms.

1. Cross-platform & container support
2. High performance
3. Asynchronous via async/await
4. Unified MVC & Web API frameworks
5. Multiple environments and development mode
6. Dependency Injection
7. WebSockets & SignalR
8. Cross-Site Request Forgery (CSRF) Protection
9. “Self hosted” Web Applications
10. Action Filters
11. Extensible Output Caching
12. Globalization and Localization
13. Swagger OpenAPI

Page Directives in Asp.Net

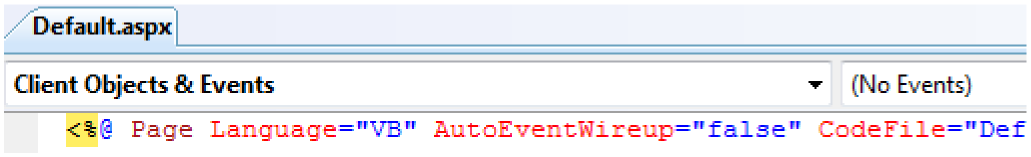
Directives

The directives are instructions that specify optional settings in Asp.Net, but they are not rendered as part of the HTML page return to the client browser.

These instructions include registering a custom control, page language etc.

It describes how the .aspx pages (web forms) or .ascx pages (user controls) are processed by the .Net framework.

Page directive



```
<%@ Page Language="VB" AutoEventWireup="false" CodeFile="Def
```

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:

It 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.

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.
- This directive supports the two attributes Name and src.
- The Name attribute defines the assembly name and the src attribute defines the source of the assembly.

@ Control

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

@ Implements

The @Implements Directive gets the ASP.NET pages to implement .Net framework interfaces. This directive only supports a single attribute interface.

@ 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.

@ Master

Identifies a page as a master page and defines attributes used by the ASP.NET page parser and compiler and can be included only in .master files.

@ MasterType

Defines the class or virtual path used to type the Master property of a page.

@ OutputCache

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

@ PreviousPageType

Creates a strongly typed reference to the source page from the target of a cross-page posting.

@ Reference

Links a page, user control, or COM control to the current page or user control declaratively.

@ Register

Associates aliases with namespaces and classes, which allow user controls and custom server controls to be rendered when included in a requested page or user control.

Introduction to server controls:

Server Controls are the tags that are understood by the server.

There are basically three types of server controls.

- **HTML Server Controls** - Traditional HTML tags
- **Web Server Controls** - New ASP. NET tags
- **Validation Server Controls -** For input validation

ASP.NET Web Forms Server Controls

- ASP.NET provides web forms controls that are used to create HTML components.
- These controls are categories as server and client based.
- The following table contains the server controls for the web forms.
- Web server controls are special ASP. NET tags understood by the server.
- Like HTML server controls, Web server controls are also created on the server and they require a runat="server" attribute to work.
- The Web server controls do not necessarily map to any existing HTML elements and represent more complex elements.
- Mostly all Web Server controls inherit from a common base class, namely the **WebControl** class defined in the **System.Web.UI.WebControls** namespace.

Control Name	**Applicable Events**	**Description**

Label	None	It is used to display text on the HTML page.
TextBox	TextChanged	It is used to create a text input in the form.
Button	Click, Command	It is used to create a button.
LinkButton	Click, Command	It is used to create a button that looks similar to the hyperlink.
ImageButton	Click	It is used to create an imagesButton. Here, an image works as a Button.
Hyperlink	None	It is used to create a hyperlink control that responds to a click event.
DropDownList	SelectedIndexChanged	It is used to create a dropdown list control.
ListBox	SelectedIndexCnhaged	It is used to create a ListBox control like the HTML control.
DataGrid	CancelCommand, EditCommand, DeleteCommand, ItemCommand, SelectedIndexChanged, PageIndexChanged, SortCommand, UpdateCommand, ItemCreated, ItemDataBound	It used to create a frid that is used to show data. We can also perform paging, sorting, and formatting very easily with this control.
DataList	CancelCommand, EditCommand, DeleteCommand, ItemCommand, SelectedIndexChanged, UpdateCommand, ItemCreated, ItemDataBound	It is used to create datalist that is non-tabular and used to show data.

CheckBox	CheckChanged	It is used to create checkbox.
CheckBoxList	SelectedIndexChanged	It is used to create a group of check boxes that all work together.
RadioButton	CheckChanged	It is used to create radio button.
RadioButtonList	SelectedIndexChanged	It is used to create a group of radio button controls that all work together.
Image	None	It is used to show image within the page.
Panel	None	It is used to create a panel that works as a container.
PlaceHolder	None	It is used to set placeholder for the control.
Calendar	SelectionChanged, VisibleMonthChanged, DayRender	It is used to create a calendar. We can set the default date, move forward and backward etc.
AdRotator	AdCreated	It allows us to specify a list of ads to display. Each time the user re-displays the page.
Table	None	It is used to create table.
XML	None	It is used to display XML documents within the HTML.
Literal	None	It is like a label in that it displays a literal, but allows us to create new literals at runtime and place them into this control.

ASP.NET Validation

Validation is important part of any web application. User's input must always be validated before sending across different layers of the application.

Validation controls are used to,

- Implement presentation logic.
- To validate user input data.
- Data format, data type and data range is used for validation.

Validation is of two types

1. Client Side
2. Serve Side

- ➢ Client side validation is good but we have to be dependent on browser and scripting language support.
- ➢ Client side validation is considered convenient for users as they get instant feedback. The main advantage is that it prevents a page from being postback to the server until the client validation is executed successfully.
- ➢ For developer point of view serve side is preferable because it will not fail, it is not dependent on browser and scripting language.

Validation controls

Following are the validation controls

Validator	**Description**
CompareValidator	It is used to compare the value of an input control against a value of another input control.
RangeValidator	It evaluates the value of an input control to check the specified range.
RegularExpressionValidator	It evaluates the value of an input control to determine whether it matches a pattern defined by a regular expression.
RequiredFieldValidator	It is used to make a control required.
ValidationSummary	It displays a list of all validation errors on the Web page.

Rich Controls

- 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 control's category.

1. FileUpload control
2. Calendar control
3. AdRotator control
4. MultiView control
5. Wizard control

FileUpload control

- FileUpload control is used to browse and upload files.
- the file is uploaded and store the file on any drive or database.
- The FileUpload control is the combination of a browse button and a text box for entering the filename.

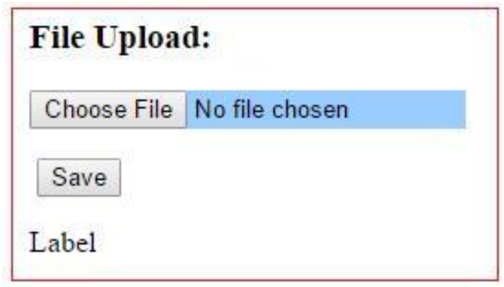
<u>properties.</u>

- **FileBytes:** It returns the contents of uploaded file as a byte array
- **FileContent:** You can get the uploaded file contents as a stream.
- **FileName:** Provides the name of uploaded file.
- **HasFile:** It is a Boolean property that checks whether particular file is available or not.
- **PostedFile:** Gets the uploaded file wrapped in the HttpPostedFile object.

Example

```
using System;
using System.Text;
public partial class RichControl : System.Web.UI.Page
{

   protected void btnSave_Click(object sender, EventArgs e)
   {
      StringBuilder sb = new StringBuilder();
      if (FileUpload1.HasFile)
      {
         try
```

```
            {
               sb.AppendFormat(" Uploaded file: {0}", FileUpload1.FileName);
               //save the file
               FileUpload1.SaveAs(@"C:\" + FileUpload1.FileName);
               //Showing the file information
               sb.Append("<br/> File Name: {0}" + FileUpload1.PostedFile.FileName);
               sb.Append("<br/> File type: {0}"+ FileUpload1.PostedFile.ContentType);
               sb.Append("<br/> File length: {0}" + FileUpload1.FileBytes.Length);
               Label1.Text = sb.ToString();
            }
            catch (Exception ex)
            {
               sb.Append("<br/> Error <br/>");
               sb.Append(ex.Message);
               Label1.Text = sb.ToString();
            }
         }
         else
         {
            Label1.Text = sb.ToString();
         }
      }
   }
```



Calendar control

- ➢ Calendar control provides you lots of property and events.
- ➢ It using these properties and events you can perform the following task with calendar control.

1. Select date.
2. Selecting a day, a week or a month.
3. Customize the calendar's appearance.

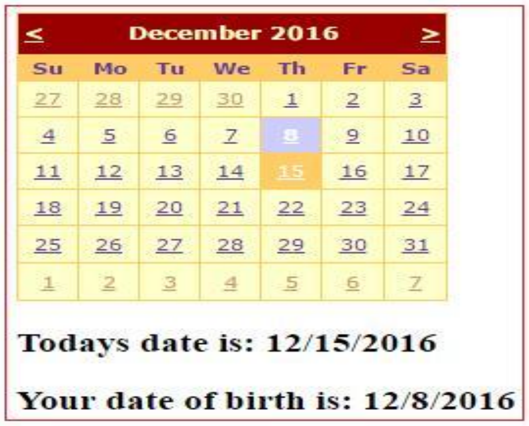
The Calendar control supports three important events:

Event	Description
SelectionChanged	This event is fired when you select a day, a week or an entire month.
DayRender	This event is fired when each data cell of the calendar control is rendered.
VisibleMonthChanged	It is raised when user changes a month.

Example

```
using System;
using System.Text;
public partial class RichControl : System.Web.UI.Page
{
   protected void Calendar1_SelectionChanged(object sender, EventArgs e)
   {
        Label1.Text ="Todays date is: "+ Calendar1.TodaysDate.ToShortDateString();
        Label2.Text = "Your date of birth is: " + Calendar1.SelectedDate.ToShortDateString();
   }
}
```

Output:



Todays date is: 12/15/2016

Your date of birth is: 12/8/2016

AdRotator control

- AdRotator control is used to display different advertisements randomly in a page.
- The list of advertisements is stored in either an XML file or in a database table.
- Lots of websites uses AdRotator control to display the advertisements on the web page.

  To create an advertisement list, first add an XML file to your project.

Code for XML file

```
<?xml version="1.0" encoding="utf-8" ?>
<Advertisements>
   <Ad>
      <ImageUrl>~ /Images/logo1.png</ImageUrl>
      <NavigateUrl>http://www.TutorialRide.com</NavigateUrl>
      <AlternateText>Advertisement</AlternateText>
      <Impressions>100</Impressions>
      <Keyword>banner</Keyword>
  </Ad>
  <Ad>
      <ImageUrl>~ /Images/logo2.png</ImageUrl>
      <NavigateUrl>http://www.TutorialRide.com</NavigateUrl>
      <AlternateText>Advertisement</AlternateText>
      <Impressions>100</Impressions>
      <Keyword>banner</Keyword>
  </Ad>
  <Ad>
      <ImageUrl>~ /Images/logo3.png</ImageUrl>
      <NavigateUrl>http://www.CareerRide.com</NavigateUrl>
      <AlternateText>Advertisement</AlternateText>
      <Impressions>100</Impressions>
      <Keyword>banner</Keyword>
  </Ad>
```

Important properties of AdRotator control.

- **ImageUrl:** The URL of the image that will be displayed through AdRotator control.
- **NavigateUrl:** If the user clicks the banner or ad then the new page is opened according to given URL.
- **AlternateText:** It is used for displaying text instead of the picture if picture is not displayed. It is also used as a tooltip.
- **Impressions:** It is a number that sets how frequently an advertisement will appear.

- **Keyword:** It is used to filter ads or identifies a group of advertisement.

MultiView control

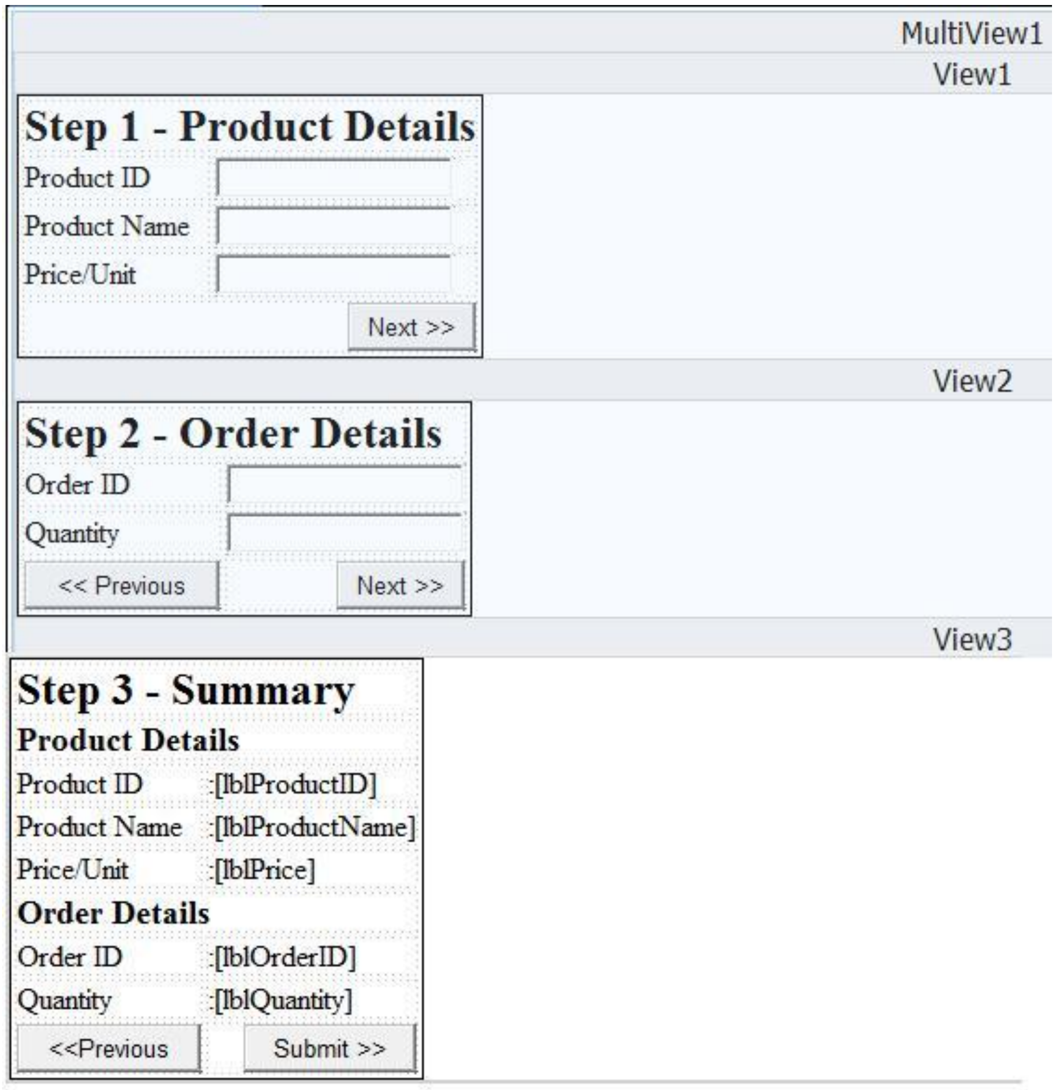
- ➢ MultiView control can be used to create a tabbed page.
- ➢ In many situations, a web form may be very long, and then you can divide a long form into multiple sub forms. MultiView control is made up of multiple view controls.
- ➢ The multiple ASP.NET controls inside view controls. One View control is displayed at a time and it is called as the active view. View control does not work separately. It is always used with a Multiview control.
- ➢ If working with Visual Studio 2010 or later, you can drag and drop a MultiView control onto the form.

Properties

- **ActiveViewIndex:** It is used to determine which view will be active or visible.
- **Views:** It provides the collection of View controls contained in the MultiView control.

For understand the Multiview control, first we will create a user interface as given below.
In the given example, in Multiview control, we have taken three separate View control.

1. In First step we will design to capture Product details
2. In Second step we will design to capture Order details
3. Next we will show summary for confirmation. saved.



MultiViewControlDemo.aspx file

Wizard Control

- This control is same as MultiView control but the main difference is that, it has inbuilt navigation buttons.
- The wizard control enables you to design a long form in such a way that you can work in multiple sub form. It perform the task in a step by step process.
- It reduces the work of developers to design multiple forms.
- It enables to create multi step user interface.
- Wizard control provides with built-in previous/next functionality.

- The Wizard control can contains one or more WizardStep as child controls. Only one WizardStep is displayed at a time. WizardStep control has an important property called as StepType. The StepType property determines the type of navigation buttons that will be displayed for that step.

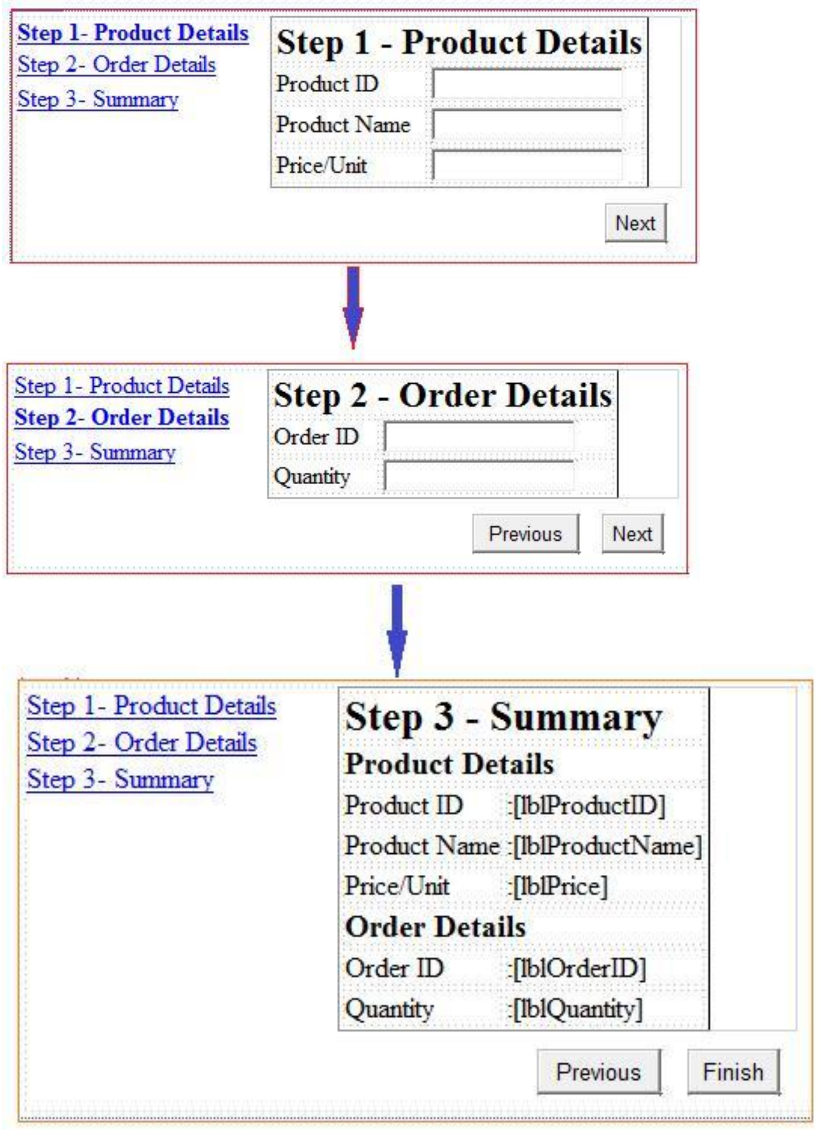
The possible values are:

The StepType associated with each WizardStep determines the type of navigation buttons that will be displayed for that step.

The StepTypes are:

1. Start:
2. Step:
3. Finish:
4. Complete:
5. Auto:

Example:



Custom controls

- ➢ Custom controls are compiled code components.
- ➢ That execute on the server, expose the object model, and render markup text, such as HTML or XML, as a normal Web Form or user control does.

How to choose the base class for a custom control

To write a custom control, directly or indirectly derive the new class from the **System.Web.UI.Control** class or the **System.Web.UI.WebControls**.WebControl class.

- **System.Web.UI.Control** -> the control to render nonvisual elements. For example, <meta> and <head> are examples of nonvisual rendering.

- **System.Web.UI.WebControls.WebControl** -> the control to render HTML that generates a visual interface on the client computer.

It used to change the functionality of existing controls, such as a button or label.
It can directly derive the new class with these existing classes and can change their default behavior.

The Control class provides the basic functionality by which you can place it in the control tree for a Page class.

The Web Control class adds the functionality to the base Control class for displaying visual content on the client computer.

Example, It can use the WebControl class to control the look and styles through properties like font, color, and height.

How to create and use a simple custom control that extends from System.Web.UI.Control using Visual Studio

1. Start Visual Studio.
2. Create a class library project, and give it a name, for example, **CustomServerControlsLib**.
3. Add a source file to the project, for example, **SimpleServerControl.cs**.
4. Include the reference of the **System.Web** namespace in the references section.
5. Check whether the following namespaces are included in the **SimpleServerControl.cs file**.
   System
   System.Collections
   System.ComponentModel
   System.Data
   System.Web
   System.Web.SessionState
   System.Web.UI
   System.Web.UI.WebControls
6. Inherit the **SimpleServerControls** class with the Control base class.
   ```
   public class SimpleServerControl : Control
   ```
7. Override the Render method to write the output to the output stream.
   ```
   protected override void Render(HtmlTextWriter writer)
   {
    writer.Write("Hello World from custom control");
   }
   ```

8. Compile the class library project. It will generate the DLL output.
9. Open an existing or create a new ASP.NET Web application project.
10. Add a Web Forms page where the custom control can be used.
11. Add a reference to the class library in the references section of the ASP.NET project.
12. Register the custom control on the Web Forms page.

```
<%@ Register TagPrefix="CC " Namespace=" CustomServerControlsLib "
Assembly="CustomServerControlsLib " %>
```

13. To instantiate or use the custom control on the Web Forms page, add the following line of code in the <form> tags.

```
<form id="Form1" method="post" runat="server">
   <CC:SimpleServerControl id="ctlSimpleControl" runat="server">
   </CC:SimpleServerControl >
</form>
```

Note In this code, SimpleServerControl is the control class name inside the class library.

14. Run the Web Forms page, and you will see the output from the custom control.

Introduction to collection

- A collection is a set of similar type of objects that are grouped together.
- *System.Collections* namespace contains specialized classes for storing and accessing the data.
- Each collection class defined in .NET has a unique feature.

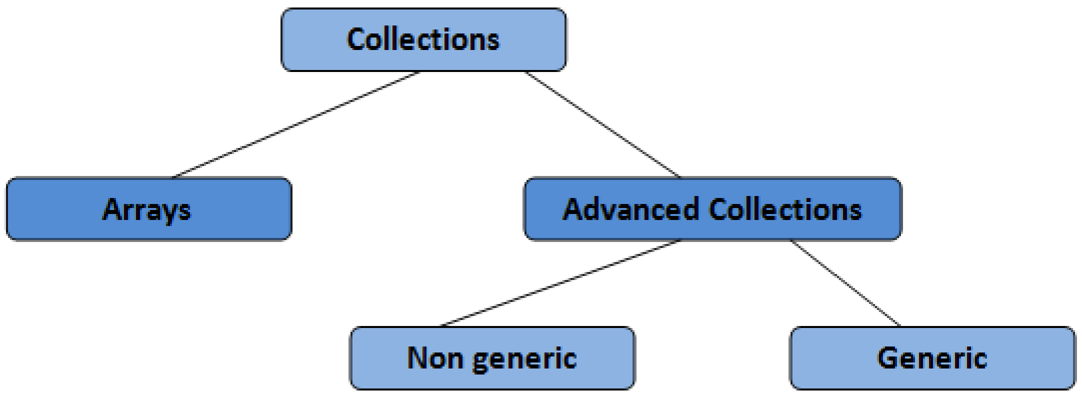
Collection Interfaces

There are some basic operations that are possible on any collection.

1. Search specific object in the collection.
2. Adding or removing objects dynamically in the collection.
3. List the objects in the collection by iterating through it.

- These basic operations are defined in the form of interfaces. All the collection classes implement these interfaces to get the required functionality.
- The implementation can be different for different classes. For example, adding objects in an *Array* is different than adding objects in *ArrayList* collection.
- *ArrayList* grows and shrinks dynamically. Both the collections need the functionality of iteration using *foreach* loop to display the list of objects contained in them.

Types of collections



Arrays

- Array class is defined in *System* namespace.
- Arrays can store any type of data but only one type at a time.
- The size of the array has to be specified at compilation time. Insertion and deletion reduce the performance.

Advanced Collections

- To perform different operations like inserting, deleting, sorting, searching, comparing, and so on.
- To perform these operations efficiently, the data needs to be organized in a specific way. This gives rise to advanced collections. Advanced collections are found in *System.Collections* namespace.

Advanced collections are again divided into two types-

1. **Generic collections**
2. **Non-generic collections**

Non-generic collections:

Every element in non-generic collection is stored as *System.Object* type.

Examples

ArrayList, *Stack*, *Queue*, *HashTable*, and so on.

- ***Boxing***

  Conversion of value type to a reference type is known as boxing. When value is boxed, CLR allocates new object on the heap and copies the value of the value type into that instance. CLR returns a reference of newly created object. This is essentially an up cast as all types are derived from *System.Object* class. Developer need not use wrapper classes or structures for value types to perform the conversion.

Example

```
int speed =80
Object obj= speed;
```

- ***Unboxing***

  It is an opposite operation of boxing, that is copies from reference type to value type on the stack. Explicit casting is required as it is a downcast. It is conversion of a derived type to a base type.

Example

```
int speed =80
Object obj= speed // boxing
int speed=(int) obj // unboxing
```

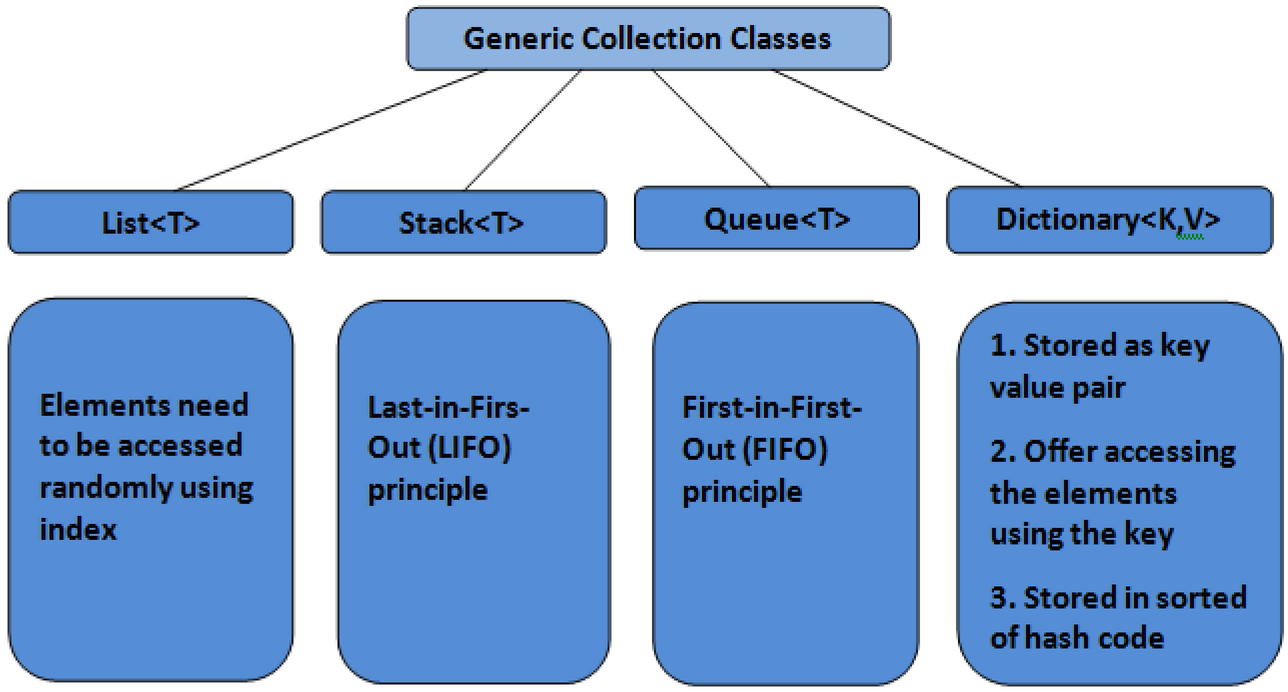
Generic collections

These are defined in *System.Collections.Generic* namespace.

Examples

Generic list, generic queue, and so on.

Some of the types of classes in generic collection are,


Generic Collection Classes
List<T>
Stack<T>
Queue<T>
Dictionary<K,V>
Elements need to be accessed randomly using index
Last-in-Firs-Out (LIFO) principle
First-in-First-Out (FIFO) principle
1. Stored as key value pair
2. Offer accessing the elements using the key
3. Stored in sorted of hash code

XML in ASP.NET

- XML is a ***cross-platform, hardware and software independent,*** text based markup language.
- It enables to store data in a structured format by using meaningful tags.
- XML stores structured data in XML documents that are similar to databases.
- That ***unlike Databases***, XML documents store data in the form of plain text, it can be used across platforms.
- The XML document to specify the structure of the data by creating a DTD or an XML schema.

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>
```

.NET Support for XML

The .NET Framework has extensive support for working with XML documents. IN the .NET framework, the support for XML documents includes:

- ***XML namespace***

- ***XML designer***
- ***XML Web Server control***
- ***XML DOM support***

XML Namespace

- The System.Xml namespace provides a rich set of classes for processing XML data.
- The commonly used classes for working with XML data are:

XmlTextReader:

Provides forward only access to a stream of XML data and checks whether or not an XML document is well formed.

```
XmlTextReader reader = new XmlTextReader("XML1.xml");
```

XmlTextWriter:

Provides forward only way of generating streams or files containing XML data that conforms to W3C XML 1.0. If you want to declare an object of the XmlTextWriter class.

```
XmlTextWriter writer = new XmlTextWriter(Response.Output);
```

.

XmlDocument:

Provides navigating and edinting features of the nodes in an XML document tree.

```
XmlDocument doc = new XmlDocument();
```

XmlDataDocument:

Provides support for XML and relational data in W3C XML DOM. You can use this class with a dataset to provide relational and non-relational views of the same set of data.

```
DataSet ds=new DataSet();
XmlDataDocument doc=new XmlDocument(ds);
```

There are a number of reasons to use XmlDataDocument:

- It gives you the freedom to work with any data by using the DOM.
- There is synchronization between an XmlDatadocument and a DataSet, and any changes in one will be reflected in the other.
- The XML document is loaded into an XmlDatadocument, the schema is preserved.

XmlPathDocument:

Provides a read-only cache for XML document processing by using XSLT. This class is optimizied for XSLT processing and does not check for the conformity of W3C DOM.

```
XmlPathDocument doc=new XmlPathDocument("XML1.xml");
```

XmlNodeReader:

Provides forward-only access to the data represented by the XmlDocument or XmlDataDocument class.

```
XmlDocument doc=new XmlPathDocument();
XmlNodeReader reader=new XmlNodeReader(doc);
```

XslTransform:

Provides support for a XSLT 1.0 style sheet syntax that enables you to transform an XML document by using XSL style sheets

```
Xsltransform xslt = new XslTransform ();
```

Web services:

Introduction

The use of ASP.NET to create Web Services based on industrial standards including Extensible Mark-up Language (**XML**), SOAP (Simple Object Access Protocol), and WSDL (web service definition language).

A Web Service is a software program that uses XML to exchange information with other software via common internet protocols.

The Web Services are a way of interacting with objects over the Internet.
A web service is

- Language Independent.
- Protocol Independent.
- Platform Independent.
- It assumes a stateless service architecture.
- Scalable (e.g. multiplying two numbers together to an entire customer-relationship management system).
- Programmable (encapsulates a task).
- Based on XML (open, text-based standard).
- Self-describing (metadata for access and use).
- Discoverable (search and locate in registries

Key Web Service Technologies

- **XML**- Describes only data. So, any application that understands XML-regardless of the application's programming language or platform has the ability to format XML in a variety of ways (well-formed or valid).
- **SOAP**- Provides a communication mechanism between services and applications.
- **WSDL**- Offers a uniform method of describing web services to other programs.

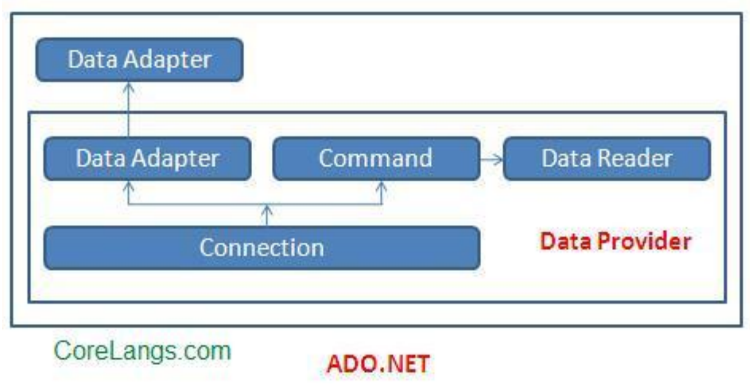
- **UDDI**- Enables the creation of searchable Web services registries.
- When these technologies are deployed together, they allow developers to package applications as services and publish those services on a network.

Web services advantages

- Use open, text-based standards, which enable components written in various languages and for different platforms to communicate.
- Promote a modular approach to programming, so multiple organizations can communicate with the same Web service.
- Comparatively easy and inexpensive to implement, because they employ an existing infrastructure and because most applications can be repackaged as Web services.
- Significantly reduce the costs of enterprise application (EAI) integration and B2B communications.
- Implemented incrementally, rather than all at once which lessens the cost and reduces the organizational disruption from an abrupt switch in technologies.
- The Web Services Interoperability Organization (WS-I) consisting of over 100 vendors promotes interoperability.

Unit – v ADO .NET

Data management in ADO.net



The two key components of ADO.NET are Data Providers and DataSet .

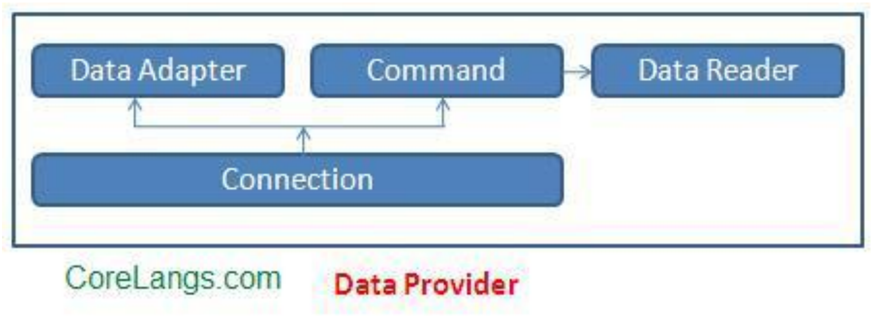
1. The Data Provider classes are meant to work with different kinds of data sources. They are used to perform all data-management operations on specific databases.
2. DataSet class provides mechanisms for managing data when it is disconnected from the data source.

Data Providers

The .Net Framework includes mainly three Data Providers for ADO.NET.

1. Microsoft SQL Server Data Provider ,
2. OLEDB Data Provider and
3. ODBC Data Provider.

SQL Server uses the SqlConnection object,



A data provider contains

1. Connection,

2. Command,
3. DataAdapter,
4. 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.
- It Defines the data source used, server name, database, ect

Command

- The Command Object uses to perform SQL statement or stored procedure to be executed at the Data Source.
- SQL code to be executed within a database connection. Used to interact with data between client and server.

DataReader

- Read Only instance to retrieve data sequencally.
- 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.

DataAdapter

- DataAdapter Object populate a Dataset Object with results from a Data Source .
- Used to manage a DataSet Object. Allows modifications to data.

ADO.NET Introduction

- It is a module of .Net Framework which is used to establish connection between application and data sources.
- Data sources can be such as SQL Server and XML. ADO.NET consists of classes that can be used to connect, retrieve, insert and delete data.
- All the ADO.NET classes are located into **System.Data.dll** and integrated with XML classes located into **System.Xml.dll.**
- ADO.NET has two main components that are used for accessing and manipulating data are the .NET Framework data provider and the DataSet.

Features of ADO.NET:

The new and updated features of ADO.NET.

1. Bulk Copy Operation

Bulk copying of data from a data source to another data source is a new feature added to ADO.NET 2.0.

Bulk copy classes provides the fastest way to transfer set of data from once source to the other. Each ADO.NET data provider provides bulk copy classes.

For example, in SQL .NET data provider, the bulk copy operation is handled by SqlBulkCopy class, which can read a DataSet, DataTable, DataReader, or XML objects. Read more about Bulk Copy here.

2. Batch Update

Batch update can provide a huge improvement in the performance by making just one round trip to the server for multiple batch updates, instead of several trips if the database server supports the batch update feature.

The UpdateBatchSize property provides the number of rows to be updated in a batch. This value can be set up to the limit of decimal.

3. Data Paging

The command object has a new execute method called ExecutePageReader. This method takes three parameters - CommandBehavior, startIndex, and pageSize. So if you want to get rows from 101 - 200, you can simply call this method with start index as 101 and page size as 100.

4. Connection Details

It get more details about a connection by setting Connection's StatisticsEnabled property to True. The Connection object provides two new methods - RetrieveStatistics and ResetStatistics.

The RetrieveStatistics method returns a HashTable object filled with the information about the connection such as data transferred, user details, curser details, buffer information and transactions.

5. DataSet.RemotingFormat Property

The DataSet.RemotingFormat is set to binary, the DataSet is serialized in binary format instead of XML tagged format, which improves the performance of serialization and deserialization operations significantly.

6. DataTable's Load and Save Methods

In previous version of ADO.NET, only DataSet had Load and Save methods.

The Load method can load data from objects such as XML into a DataSet object and Save method saves the data to a persistent media. Now DataTable also supports these two methods.

It can also load a DataReader object into a DataTable by using the Load method.

7. New Data Controls

In Toolbox, you will see these new controls - DataGridView, DataConnector, and DataNavigator. See Figure 1. Now using these controls, you can provide navigation (paging) support to the data in data bound controls.

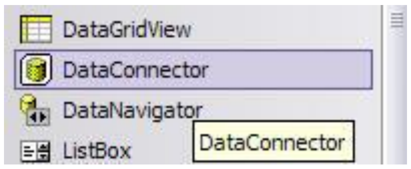


Fig 1. Data bound controls.

8. DbProvidersFactories Class

This class provides a list of available data providers on a machine. It can use this class and its members to find out the best suited data provider for your database when writing a database independent applications.

9. Customized Data Provider

It providing the factory classes now ADO.NET extends its support to custom data provider.

10. DataReader's New Execute Methods

Now command object supports more execute methods. Besides old ExecuteNonQuery, ExecuteReader, ExecuteScaler, and ExecuteXmlReader, the new execute methods are ExecutePageReader, ExecuteResultSet, and ExecuteRow.

ASP.net using SQL Server:

To connect the SQL database into ASP .net using C# language.

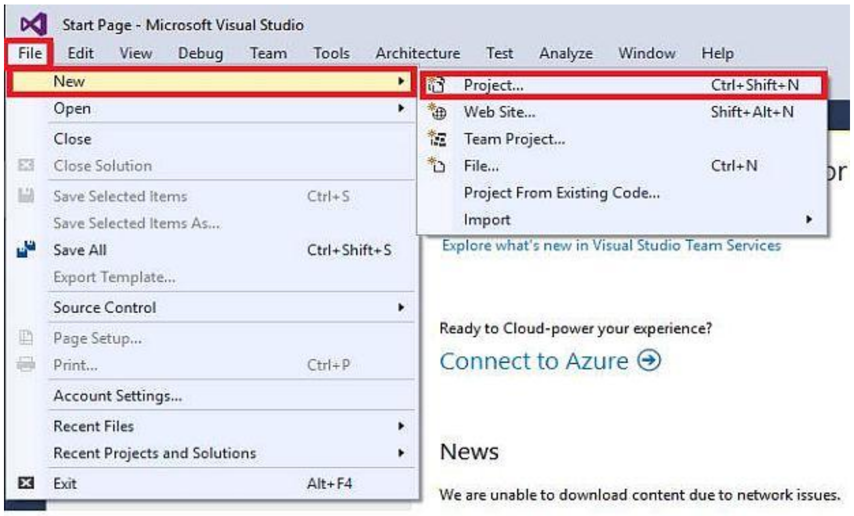
Requirements

- Visual Studio 2008
- ASP.NET 4.5.2
- SQL Server 2008

The following step for connect to the SQL database into ASP.NET, using C#, given below.

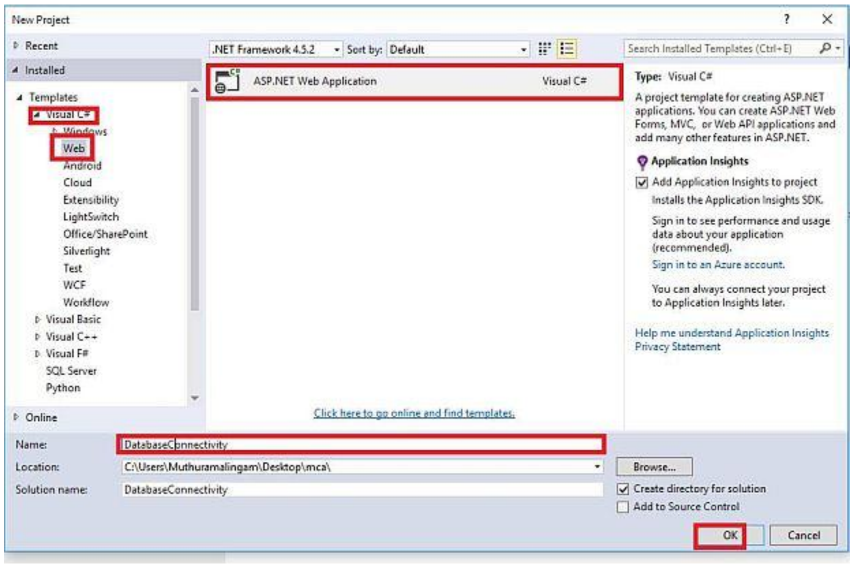
Step 1

Open Visual Studio 2008 , go to the File >> New >> Project (or) the shortcut key "Ctrl+Shift +N".



Step 2

select Visual C# >> Web >> ASP.NET Web Application. Finally, click "OK" button.



.NET Framework 4.5.2 Sort by: Default

Type: Visual C#

A project template for creating ASP.NET applications. You can create ASP.NET Web Forms, MVC, or Web API applications and add many other features in ASP.NET.

☑ Add Application Insights to project

Installs the Application Insights SDK.

Sign in to see performance and usage data about your application (recommended).

You can always connect your project to Application Insights later.

Name: DatabaseConnectivity

Location: C:\Users\Muthuramalingam\Desktop\mca\

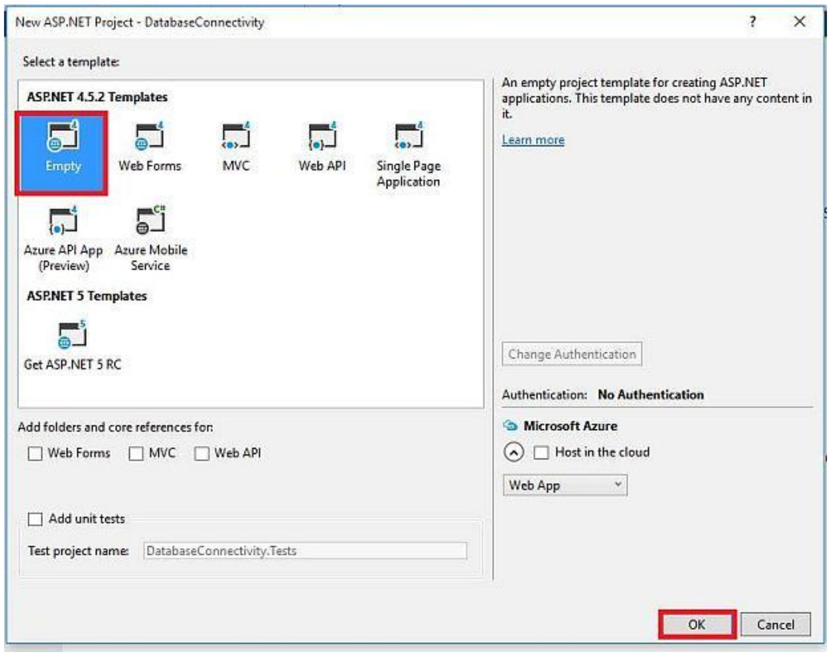
Solution name: DatabaseConnectivity

☑ Create directory for solution

☐ Add to Source Control

Step 3

Here, you can select the template for your ASP.NET Application. We are choosing "Empty" here. Now, click OK button.



An empty project template for creating ASP.NET applications. This template does not have any content in it.

Authentication: No Authentication

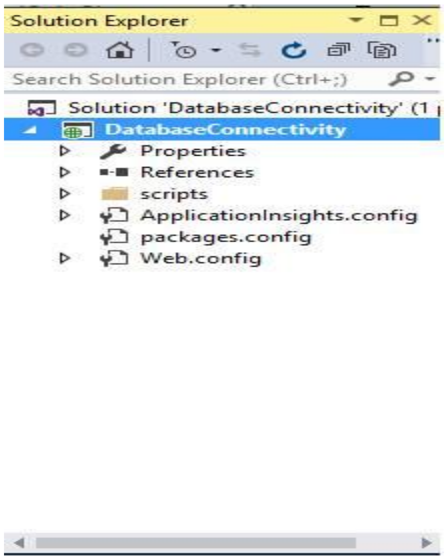
Add folders and core references for:
☐ Web Forms ☐ MVC ☐ Web API

☐ Add unit tests

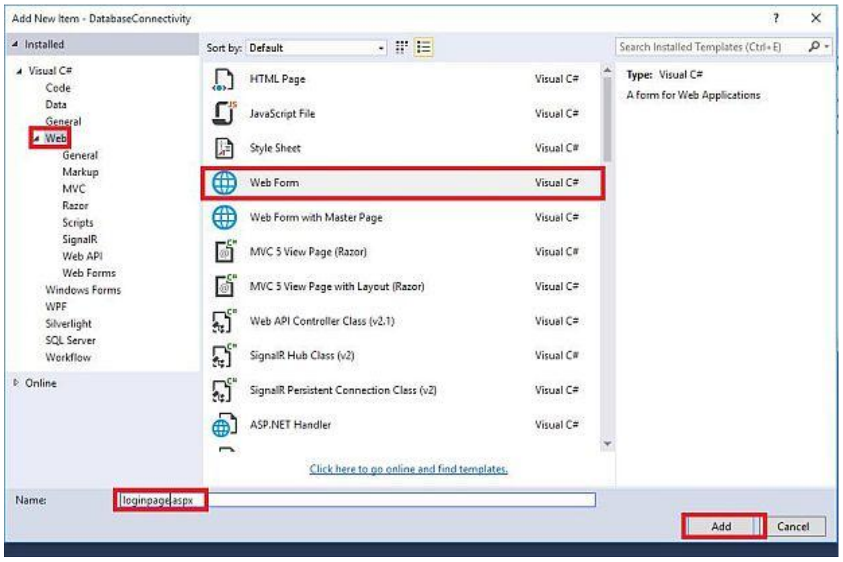
Test project name: DatabaseConnectivity.Tests

Step 4

Now, open the project and look for the Solution Explorer.


Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'DatabaseConnectivity' (1
DatabaseConnectivity
Properties
References
scripts
ApplicationInsights.config
packages.config
Web.config

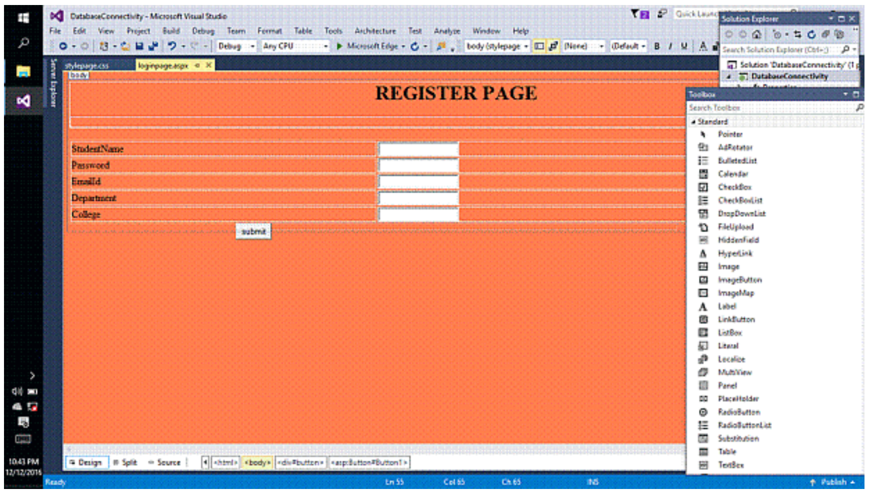
Here, open the default .aspx. If you want a Webform, you can add the page (Web Form). Add+New item (Ctrl+Shift+A).



Type: Visual C#
A form for Web Applications

Name: loginpage.aspx

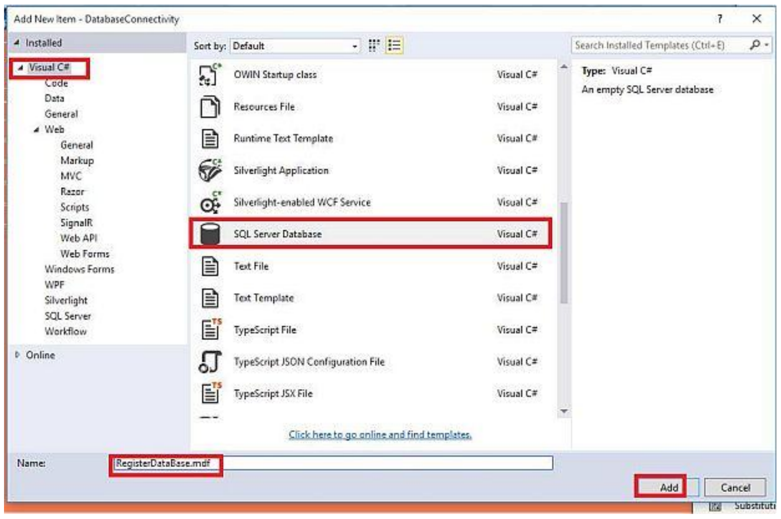
Now, we can create a login page, using ASP.NET code. You need to follow the drag and drop method. Here, we already created the login page.



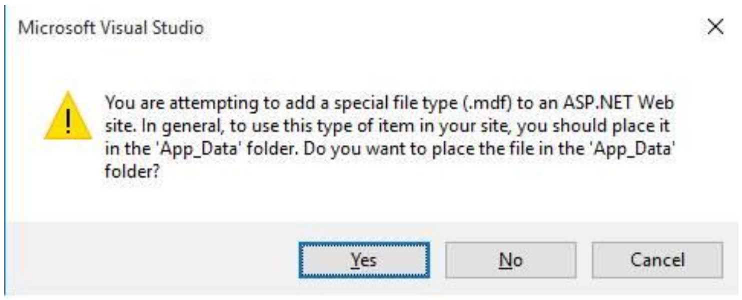
Step 5

Now, open the project. If you want a SQL Server database, you can add the page (SQL Server database). Add+New item (Ctrl+Shift+A).

Here, you can select Visual C# and choose SQL Server database. Afterwards, click "OK" button.

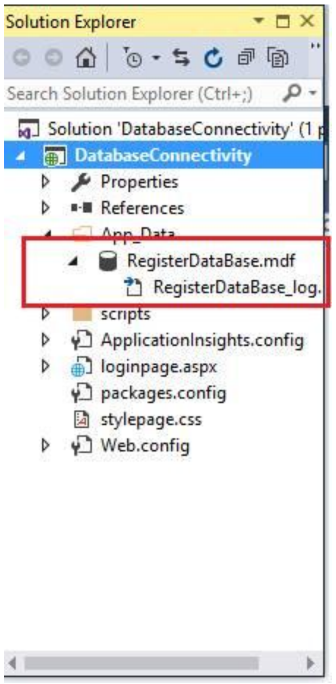

Add New Item - DatabaseConnectivity
Installed
Visual C#
Code
Data
General
Web
General
Markup
MVC
Razor
Scripts
SignalR
Web API
Web Forms
Windows Forms
WPF
Silverlight
SQL Server
Workflow
Online
Sort by: Default
Search Installed Templates (Ctrl+E)
OWIN Startup class Visual C#
Resources File Visual C#
Runtime Text Template Visual C#
Silverlight Application Visual C#
Silverlight-enabled WCF Service Visual C#
SQL Server Database Visual C#
Text File Visual C#
Text Template Visual C#
TypeScript File Visual C#
TypeScript JSON Configuration File Visual C#
TypeScript JSX File Visual C#
Type: Visual C#
An empty SQL Server database
Click here to go online and find templates.
Name: RegisterDataBase.mdf
Add
Cancel

Here, open the new Window and click YES button.


Microsoft Visual Studio

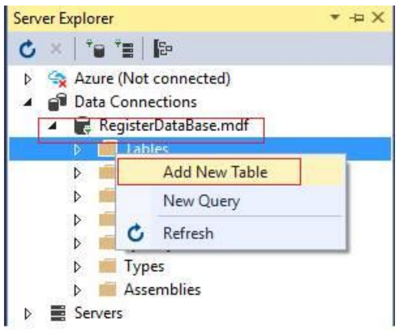
You are attempting to add a special file type (.mdf) to an ASP.NET Web site. In general, to use this type of item in your site, you should place it in the 'App_Data' folder. Do you want to place the file in the 'App_Data' folder?

Now, add this to the database in our project.

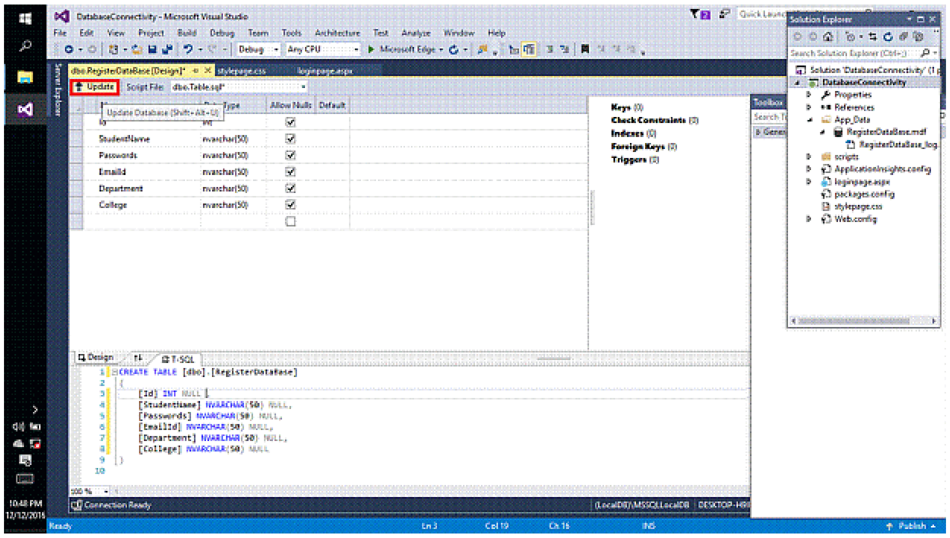


Step 6

Now, we can go to the Server Explorer and add your database. You can click the Tables and afterwards, click Add New Table.



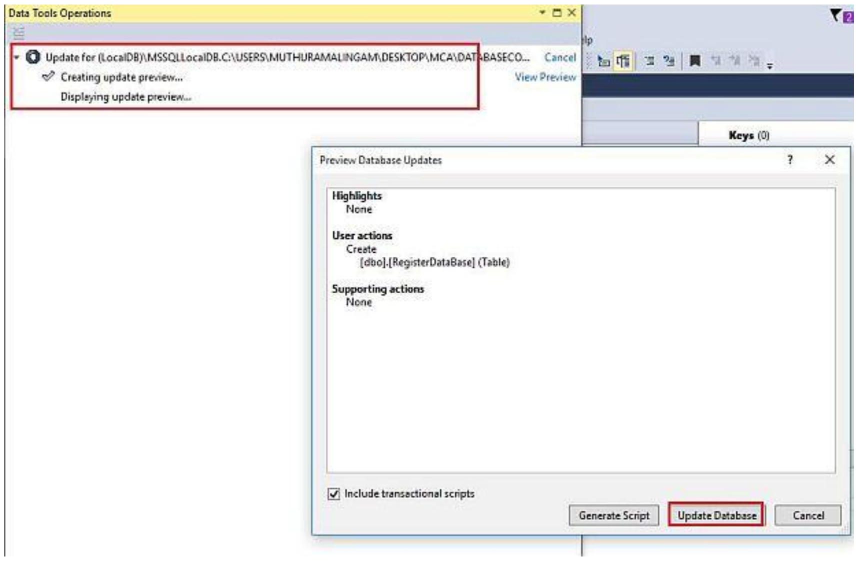
Now, open the new table and you can fill the data, which is like (studentname, password) and afterwards, you click the Update .



Name	Data Type	Allow Nulls	Default
Id	int	✓	
StudentName	nvarchar(50)	✓	
Passwords	nvarchar(50)	✓	
EmailId	nvarchar(50)	✓	
Department	nvarchar(50)	✓	
College	nvarchar(50)	✓	

```
CREATE TABLE [dbo].[RegisterDataBase]
(
    [Id] INT NULL ,
    [StudentName] NVARCHAR(50) NULL,
    [Passwords] NVARCHAR(50) NULL,
    [EmailId] NVARCHAR(50) NULL,
    [Department] NVARCHAR(50) NULL,
    [College] NVARCHAR(50) NULL
)
```

Here, click database in update and subsequently click update the database.



Data Tools Operations

Update for (LocalDB)\MSSQLLocalDB.C:\USERS\MUTHURAMALINGAM\DESKTOP\MCA\DATABASECO... Cancel
Creating update preview... View Preview
Displaying update preview...

Preview Database Updates

Highlights
None

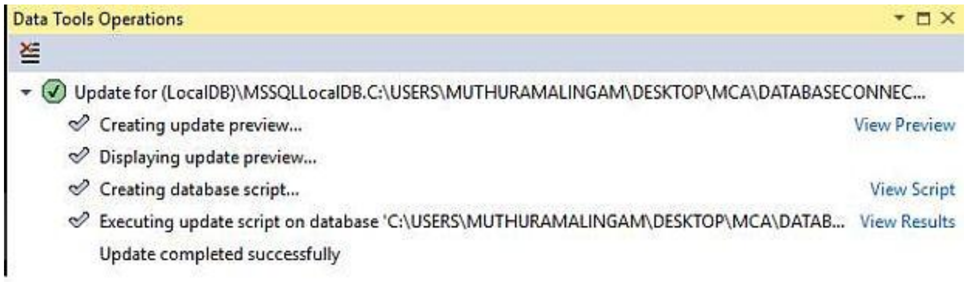
User actions
Create
[dbo].[RegisterDataBase] (Table)

Supporting actions
None

Include transactional scripts

Generate Script | Update Database | Cancel

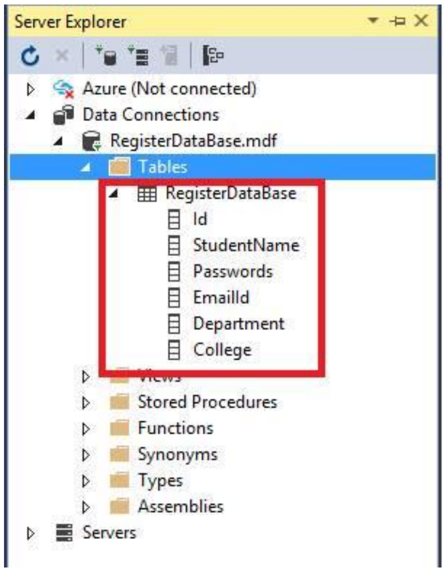
Here, the database is updated.



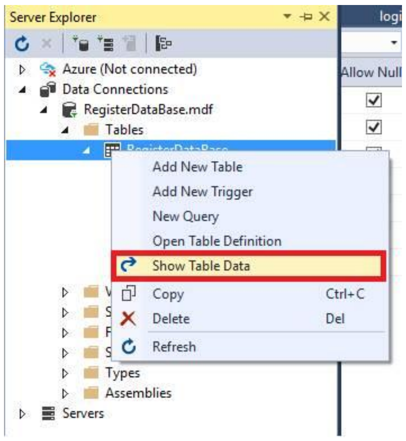
Data Tools Operations

Update for (LocalDB)\MSSQLLocalDB.C:\USERS\MUTHURAMALINGAM\DESKTOP\MCA\DATABASECONNEC...
Creating update preview... View Preview
Displaying update preview...
Creating database script... View Script
Executing update script on database 'C:\USERS\MUTHURAMALINGAM\DESKTOP\MCA\DATAB... View Results
Update completed successfully

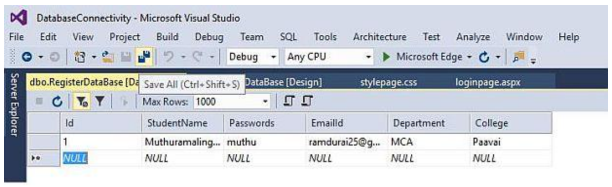
Here, the database and data are added.



Now, we can click on the right click and click Show Table Data.



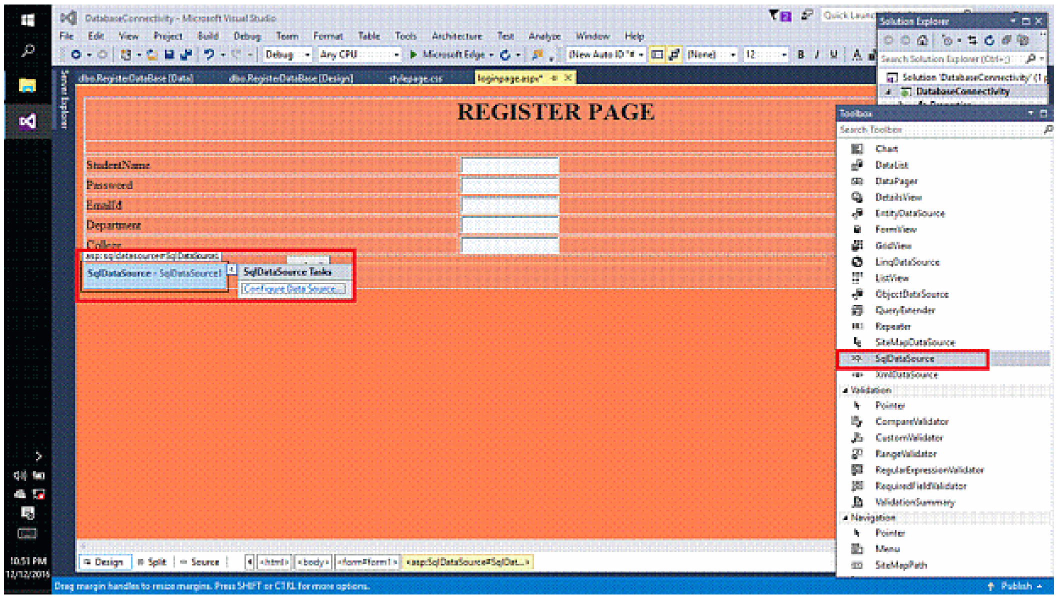
Now, the data is stored.



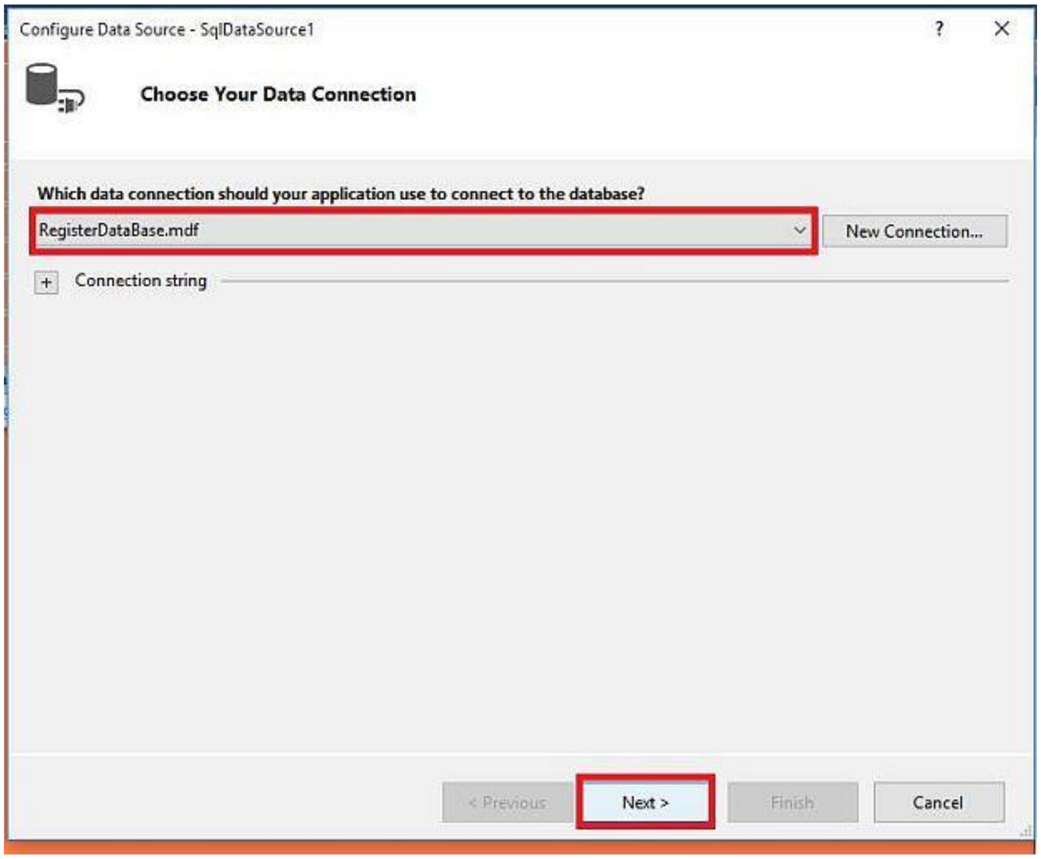
Id	StudentName	Passwords	EmailId	Department	College
1	Muthuramaling...	muthu	ramdurai25@g...	MCA	Paavai
NULL	NULL	NULL	NULL	NULL	NULL

Step 7

Now, you can add SQL Data Source. Drag and drop method. Here, click Configure Data Source



Now, we can choose your database and click NEXT button.

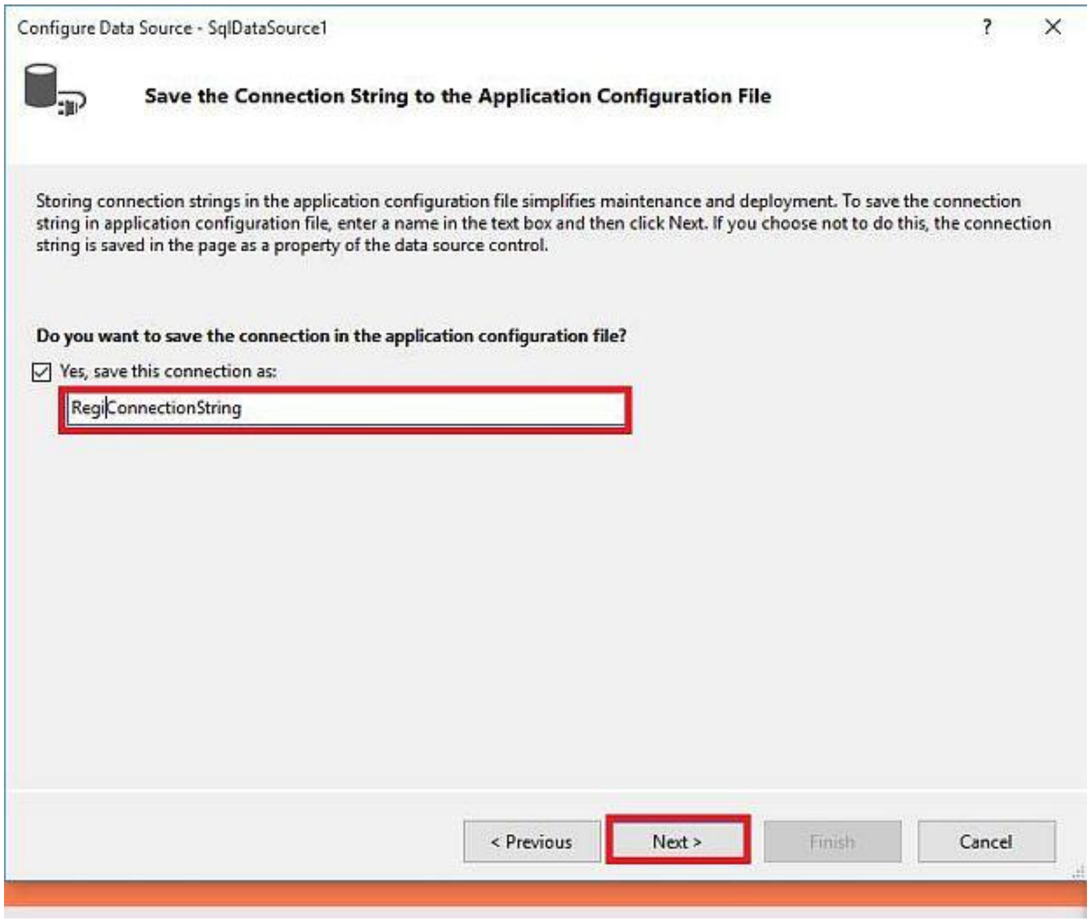

Configure Data Source - SqlDataSource1

Choose Your Data Connection

Which data connection should your application use to connect to the database?

RegisterDataBase.mdf

Now, you can select the ConnectionString and Click NEXT button


Configure Data Source - SqlDataSource1

Save the Connection String to the Application Configuration File

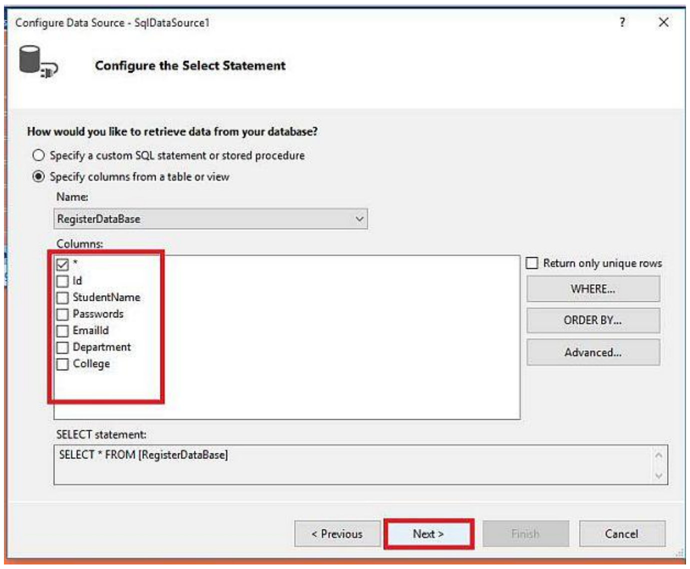
Storing connection strings in the application configuration file simplifies maintenance and deployment. To save the connection string in application configuration file, enter a name in the text box and then click Next. If you choose not to do this, the connection string is saved in the page as a property of the data source control.

Do you want to save the connection in the application configuration file?

☑ Yes, save this connection as:

RegiConnectionString

Now, we can choose Specify columns from a table or view and afterwards, click Next button.


Configure Data Source - SqlDataSource1

How would you like to retrieve data from your database?

○ Specify a custom SQL statement or stored procedure
● Specify columns from a table or view

Name: RegisterDataBase

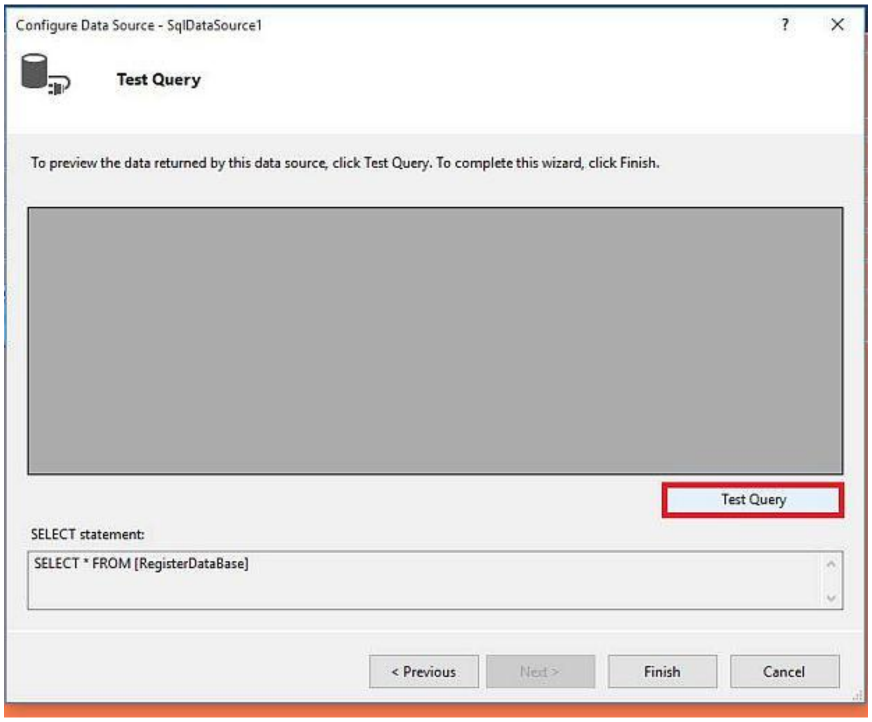
Columns:
☑ *
☐ Id
☐ StudentName
☐ Passwords
☐ EmailId
☐ Department
☐ College

☐ Return only unique rows

SELECT statement:

```
SELECT * FROM [RegisterDataBase]
```

Now, click Test Query.



Configure Data Source - SqlDataSource1

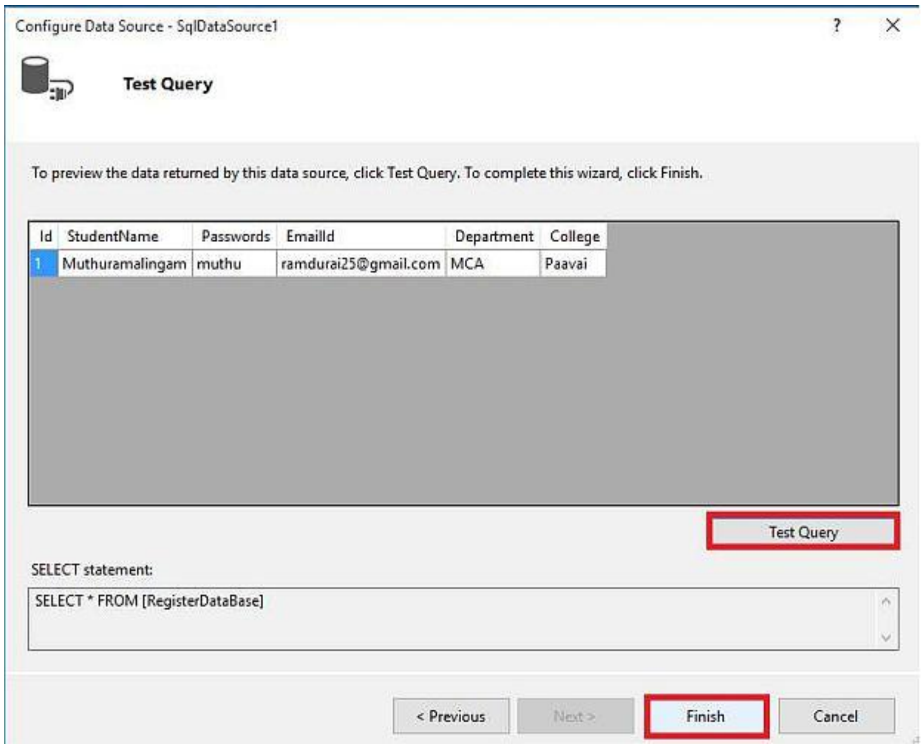
Test Query

To preview the data returned by this data source, click Test Query. To complete this wizard, click Finish.

SELECT statement:

```
SELECT * FROM [RegisterDataBase]
```

Here, add the data and click Finish button.



Configure Data Source - SqlDataSource1

Test Query

To preview the data returned by this data source, click Test Query. To complete this wizard, click Finish.

Id	StudentName	Passwords	EmailId	Department	College
1	Muthuramalingam	muthu	ramdurai25@gmail.com	MCA	Paavai

SELECT statement:

```
SELECT * FROM [RegisterDataBase]
```

Step 8

Now, you can go to CS(C# code) page and you will write the C# code.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data.SqlClient;
using System.Configuration;

namespace DatabaseConnectivity
{
    public partial class loginpage  System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            if(IsPostBack)
            {
                SqlConnection conn = new SqlConnection(ConfigurationManager.ConnectionStrings["RegiConnectionString"].ConnectionString);
                conn.Open();
                string checkuser = "select count(*) from RegisterDataBase where StudentName='"+TextBox1.Text+"'";
                SqlCommand cmd = new SqlCommand(checkuser, conn);
                int temp = Convert.ToInt32(cmd.ExecuteScalar().ToString());

                if (temp == 1)
                {
                    Response.Write("Student Already Exist");
                }

                conn.Close();
            }

            }

        protected void Button1_Click(object sender, EventArgs e)
        {
            try
            {

                SqlConnection conn = new SqlConnection(ConfigurationManager.ConnectionStrings["RegiConnectionString"].ConnectionString);
                conn.Open();
                string insertQuery = "insert into RegisterDataBase(StudentName,Passwords,EmailId,Department,College)values (@studentname,@passwords,@emailid,@department,@college)";
                SqlCommand cmd = new SqlCommand(insertQuery, conn);
                cmd.Parameters.AddWithValue("@studentname", TextBox1.Text);
                cmd.Parameters.AddWithValue("@passwords", TextBox2.Text);
                cmd.Parameters.AddWithValue("@emailid", TextBox3.Text);
                cmd.Parameters.AddWithValue("@department", TextBox4.Text);
                cmd.Parameters.AddWithValue("@college", TextBox5.Text);
                cmd.ExecuteNonQuery();

                Response.Write("Student registeration Successfully!!!thank you");

                conn.Close();

            }
            catch (Exception ex)
            {
                Response.Write("error" + ex.ToString());
            }
        }
    }
}
```

Now, you can see the Loginpage.aspx code.

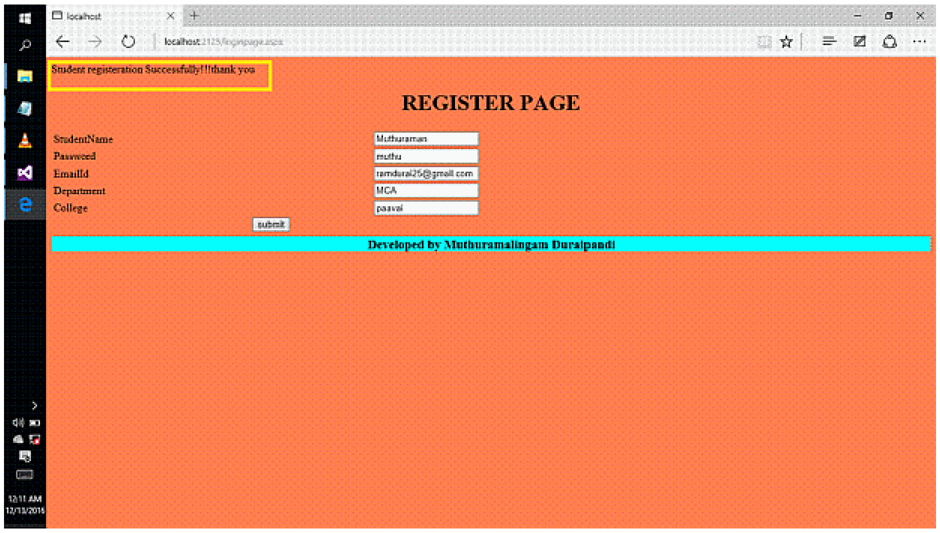
```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="loginpage.aspx.cs" Inherits="DatabaseConnectivity.loginpage" %>

<!DOCTYPE html>

<html xmlns="http//www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    <link href="stylepage.css" type="text/css" rel="stylesheet" />
    <style type="text/css">
        .auto-style1 {
            width 100%;
        }
    </style>
</head>
<body>
    <form id="form1" runat="server">
    <div id="title">
    <h1>REGISTER PAGE</h1>
```

```
</div>
    <div id ="teble"></div>
<table class="auto-style1">
    <tr>
        <td>
            <aspLabel ID="Label1" runat="server" Text="StudentName"></aspLabel></td
>
        <td>
            <aspTextBox ID="TextBox1" runat="server"></aspTextBox></td>
    </tr>
    <tr>
        <td>
            <aspLabel ID="Label2" runat="server" Text="Password"></aspLabel></td>
        <td>
            <aspTextBox ID="TextBox2" runat="server"></aspTextBox></td>
    </tr>
    <tr>
        <td>
            <aspLabel ID="Label3" runat="server" Text="EmailId"></aspLabel></td>
        <td>
            <aspTextBox ID="TextBox3" runat="server"></aspTextBox></td>
    </tr>
    <tr>
        <td>
            <aspLabel ID="Label4" runat="server" Text="Department"></aspLabel></td>

        <td>
            <aspTextBox ID="TextBox4" runat="server"></aspTextBox></td>
    </tr>
    <tr>
        <td>
            <aspLabel ID="Label5" runat="server" Text="College"></aspLabel></td>
        <td>
            <aspTextBox ID="TextBox5" runat="server"></aspTextBox></td>
    </tr>
</table>
<div id="button">
    <aspButton ID="Button1" runat="server" Text="submit" OnClick="Button1_Click"
BackColor="Yellow" />
</div>
    <div id="sim"></div>
    <aspSqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="<%$
ConnectionStringsRegiConnectionString %>" SelectCommand="SELECT * FROM [Reg
isterDataBase]"></aspSqlDataSource>

    <div id="grid">
```

```
            <aspGridView ID="GridView1" runat="server" AllowPaging="True" AllowSorting="True" AutoGenerateColumns="False" CellPadding="4" DataSourceID="SqlDataSource1" ForeColor="#333333" GridLines="None">
                <AlternatingRowStyle BackColor="White" ForeColor="#284775" />
                <Columns>
                    <aspBoundField DataField="Id" HeaderText="Id" SortExpression="Id" />
                    <aspBoundField DataField="StudentName" HeaderText="StudentName" SortExpression="StudentName" />
                    <aspBoundField DataField="Passwords" HeaderText="Passwords" SortExpression="Passwords" />
                    <aspBoundField DataField="EmailId" HeaderText="EmailId" SortExpression="EmailId" />
                    <aspBoundField DataField="Department" HeaderText="Department" SortExpression="Department" />
                    <aspBoundField DataField="College" HeaderText="College" SortExpression="College" />
                </Columns>
                <EditRowStyle BackColor="#999999" />
                <FooterStyle BackColor="#5D7B9D" -Bold="True" ForeColor="White" />
                <HeaderStyle BackColor="#5D7B9D" -Bold="True" ForeColor="White" />
                <PagerStyle BackColor="#284775" ForeColor="White" HorizontalAlign="Center" />
                <RowStyle BackColor="#F7F6F3" ForeColor="#333333" />
                <SelectedRowStyle BackColor="#E2DED6" -Bold="True" ForeColor="#333333" />
                <SortedAscendingCellStyle BackColor="#E9E7E2" />
                <SortedAscendingHeaderStyle BackColor="#506C8C" />
                <SortedDescendingCellStyle BackColor="#FFFDF8" />
                <SortedDescendingHeaderStyle BackColor="#6F8DAE" />
            </aspGridView>
        </div>

            <div id="last">
            <h3>Developed by
                  Muthuramalingam Duraipandi</h3>
        </div>
    </form>
    </body>
</html>
```

Step 9

Here, you need to run any Browser and after a few minutes, you will get some output. Now, we can insert the data into your database.



Student registeration Successfully!!!thank you

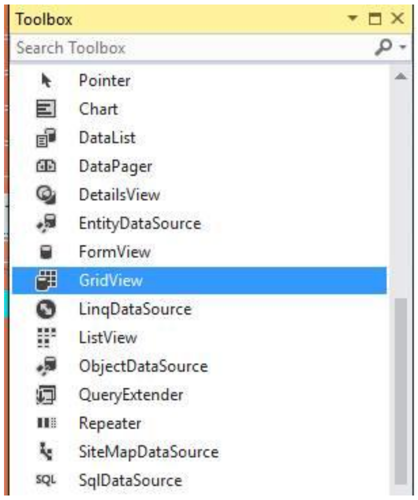
REGISTER PAGE

StudentName	Muthuraman
Password	muthu
EmailId	ramdurai25@gmail.com
Department	MCA
College	paavai

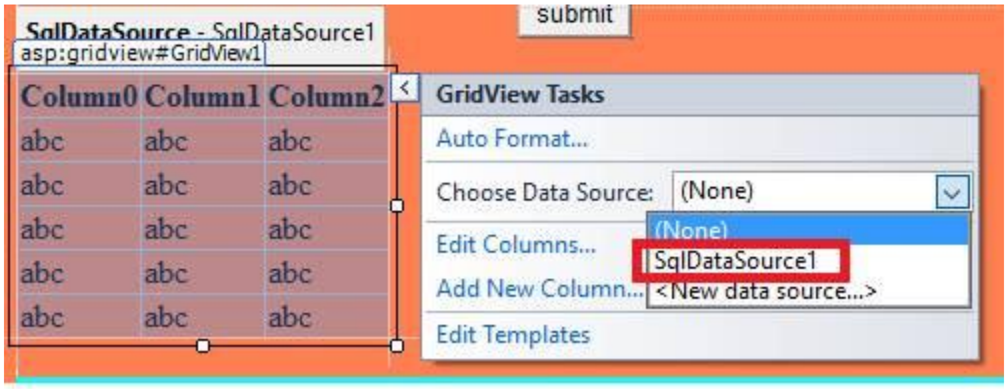
Developed by Muthuramalingam Duraipandi

Step 10

Now, we can added the GridView. Drag and drop method needs to be used.



Now, you can choose the data source, it is sqlDataSource.

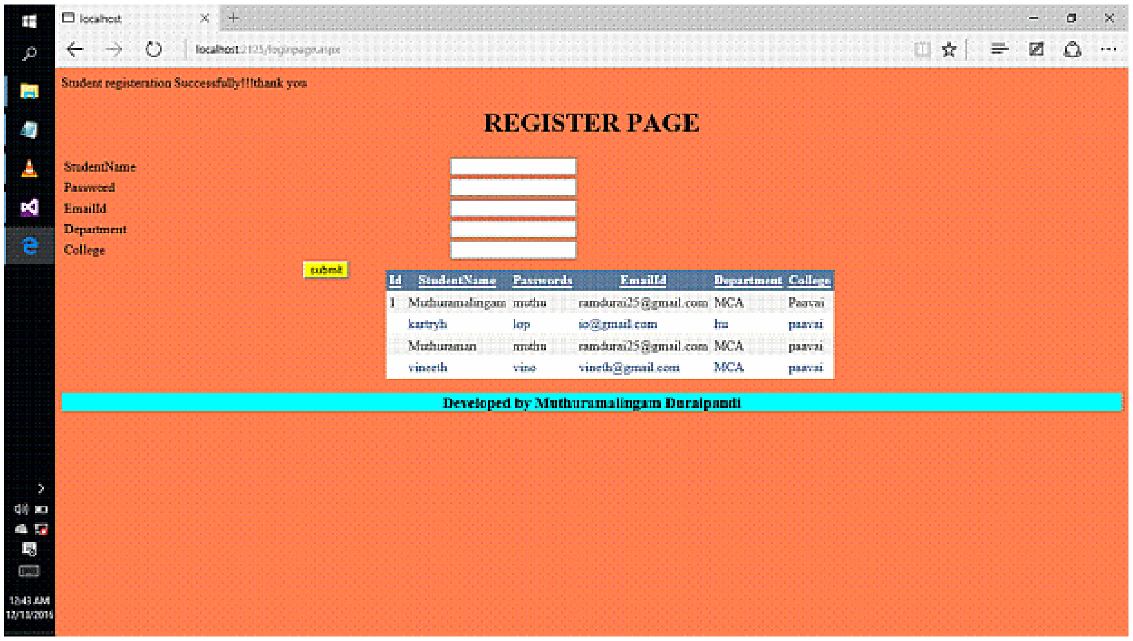


Column0	Column1	Column2
abc	abc	abc
abc	abc	abc
abc	abc	abc
abc	abc	abc
abc	abc	abc

Here, the database data is added to GridView.

Step 11

Here, you need to run any Browser and after a few minutes, you will get an output. Now, we can view the data.



Student registeration Successfully!!!thank you

REGISTER PAGE

StudentName
Password
EmailId
Department
College

Id	StudentName	Passwords	EmailId	Department	College
1	Muthuramalingam	muthu	ramdurai25@gmail.com	MCA	Paavai
	kartryh	lop	io@gmail.com	hu	paavai
	Muthuraman	muthu	ramdurai25@gmail.com	MCA	paavai
	vineeth	vino	vineth@gmail.com	MCA	paavai

Developed by Muthuramalingam Duraipandi