.NET Frame Work

Overview of .NET Framework

- A software developing platform.
- It provides tools and libraries.
- It is a set technology to work together to solve problems.
- Support more than 30 languages.
- It is the foundation for designing, developing, and deploying applications.

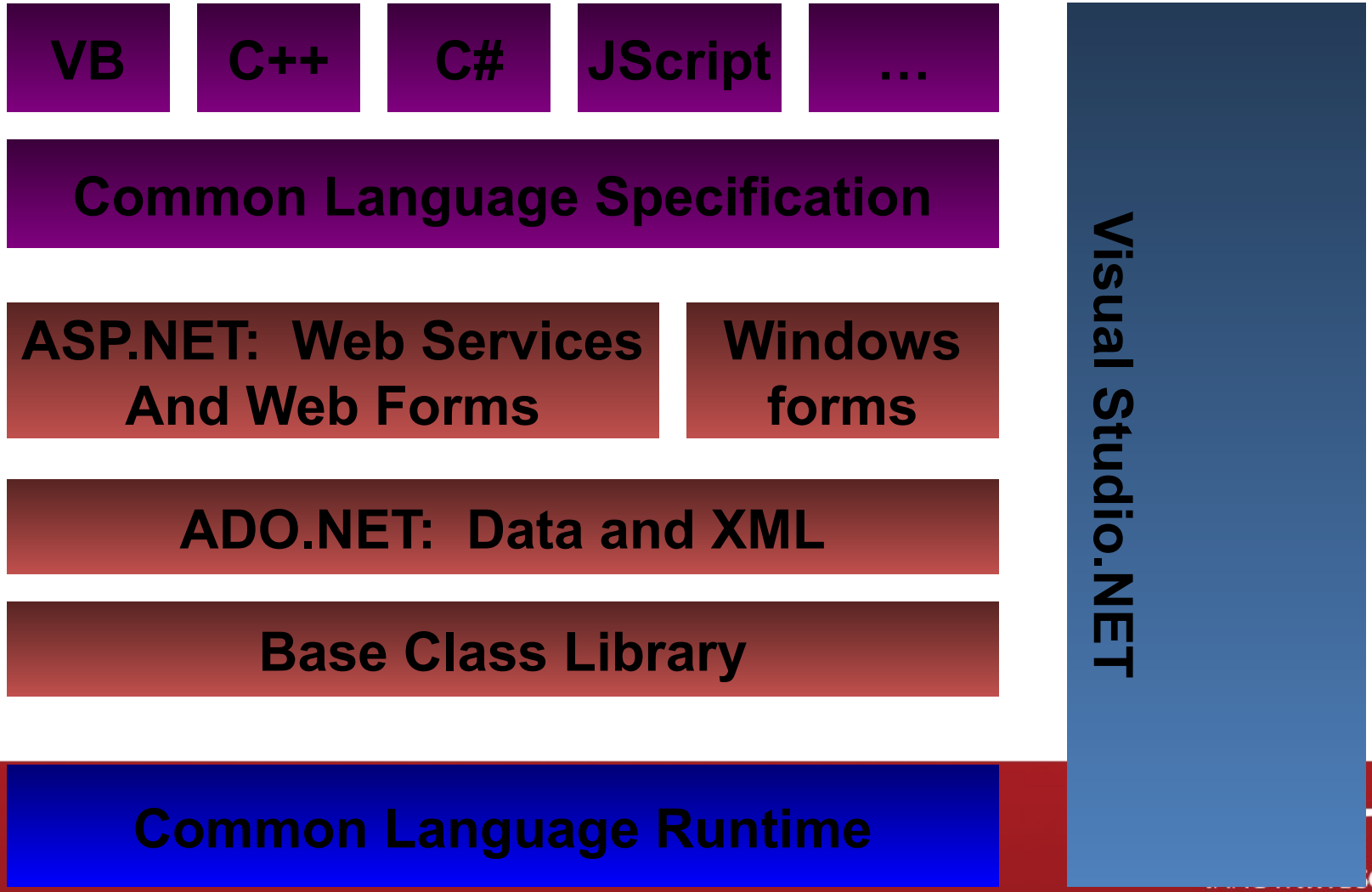
- We can create webforms,window forms and web services
- Web forms- web applications
- Window forms- desktop/window based
- Web services-distributed applications
- The .NET Framework consists of all the technologies that help in creating and running robust, scalable, and distributed applications.

- We can create webforms,window forms and web services
- Web forms- web applications
- Window forms- desktop/window based
- Web services-distributed applications
- The .NET Framework consists of all the technologies that help in creating and running robust, scalable, and distributed applications.

.NET Framework Features

- Easy to develop web applications
- Oops support
- Muti language support
- Automatic memory management
- Strong XML support

The .NET Framework Overview



.NET Frameworks consists of

- XML
- ADO
- CLS – Set of rules that .net language must follow
- CTS – Support different data types
- CLR

Common Language Runtime (CLR):

- CLR is the environment where all programs in .NET are executed.
- CLR provides services such as code compilation, memory allocation, and garbage collection.
- CLR allows the execution of code across different platforms by translating code into Microsoft Intermediate Language (MSIL).
- MSIL is a low level language that the CLR understands.
- MSIL is converted into machine language during execution by the JIT compiler. During JIT compilation, code is also checked for type safety.
- Type safety ensures that objects are always accessed in a compatible way.

- CLR consists of a set of common rules followed by all the languages of the .NET Framework. This set of rules is known as Common Language Specification (CLS).
- CLS enables an object or application to interact with the objects or applications of other languages.

- The classes that follow the rules specified by CLS are termed as CLS-compliant classes. The classes defined in the .NET Framework class library are CLS-compliant.

- One of the specifications defined in CLS is CTS, which provides a type system that is common across all languages. CTS define how data types are declared, used, and managed in the code at run time.
- While executing the program, CLR:
  - Identifies the process of compilation
  - Identifies the process of code execution

The .NET Framework Class Library:

- The .NET Framework class library was designed to make it easier to perform the most common programming tasks.
- The .NET Framework consists more than 13,000 classes you can use when building your application
- The .NET Framework class library works with any .NET language, such as VB.NET, VC++ .NET, and VC#.
- The .NET Framework class library provides classes that can be used in the code to accomplish a range of common programming tasks, such as string management, data collection, database connectivity, and file access.
- The .NET Framework class library comprises of:
  - Namespaces
  - Assemblies

Namespaces

- Namespace is simply a category.
- A namespace provides a fundamental unit of logical code grouping.
- Microsoft divides the classes in the framework into separate namespaces.
- For example, all the classes for working with the file system are located in the System.IO namespace.

Assemblies

- Smallest executable unit in a .net framework.
- An assembly is an actual .dll file on your hard drive in which the classes in .NET Framework are stored.
- An assembly is the primary unit of deployment , security, and version control in .NET Framework.
- All types in the .NET Framework must exist in assemblies; the common language runtime does not support types outside of assemblies.

C# compiler

- It is used to convert program written in one language into a machine level language that the system can understand. Or convert source code into machine code.

Difference between managed and unmanaged code

Managed Code

- Managed code is code written in many high-level programming languages that are available for use with the Microsoft .NET Framework, including VB.NET, C#, J#, JScript.NET etc. .
- The Managed Code running under a Common Language Runtime cannot be accessed outside the runtime environment as well as cannot call directly from outside the runtime environment.
- It offers services like garbage collection, run-time type checking, reference checking etc

Unmanaged Code

- Unmanaged code compiles straight to machine code and directly executed by the Operating System.
- All code compiled by traditional C/C++ compilers are Unmanaged Code.

Visual Studio IDE Components

- Microsoft Visual Studio is an Integrated Development Environment (IDE) to work with Microsoft languages. It is the premier tool that developers can posses to easily work with Microsoft technologies.
- **Toolbox Window**

The Toolbox window contains a list of controls or components that you can drag and drop onto your design surface.

- **Solution Explorer Window**

In Visual Studio .NET, a solution is a set of one or more projects that are part of the same application.

The Solution Explorer window shows you an expandable list of projects, each project's references, and each project's components.

If this window is closed, you can open it by selecting the View, Solution Explorer menu item.

Components may be made up of forms, classes, modules, and any other file types it takes to create your application.

Double-click an item in order to edit that item within the IDE.

- The Solution Explorer window displays a series of buttons across its top, and these buttons dynamically change based on the item you have selected in the Solution Explorer window.

Class View Window

- When you start creating your own classes, you may want to see a list of all the properties and methods available in those classes.

- **Server Explorer Window**

The Server Explorer window (accessed using the View, Server Explorer menu) allows you to view the various services available on a particular server. These services include Crystal Services (for working with Crystal Reports), Event Logs, Message Queues, Performance Counters, and SQL Servers.

- **Properties Window**

The Properties window provides a visual means of investigating and altering the properties of any object within the Visual Studio .NET environment.

You can display the Properties window using the View, Properties Window menu item.

Once this window is visible, you can either view the list alphabetically or categorized by attribute.

Some properties within this window can be selected from a list; others allow you to click a button that brings up a dialog box. Still others require you to supply text by typing into a text box.

- **Object Browser Window**

Similar to the Class View window, the Object Browser window shows you a list of classes and their respective members.

The main difference between these two tools is that the Object Browser allows you to browse all referenced components, not just the components for the current project like the Class View window does.

A nice feature of the Object Browser is that it also shows you the full declaration for the method or property.

- **Task List Window**

The Task List window displays, among other items, any To Do items that you have entered in your code.

In addition, you'll see information about build errors in this window. Bring up the Task List window using the View

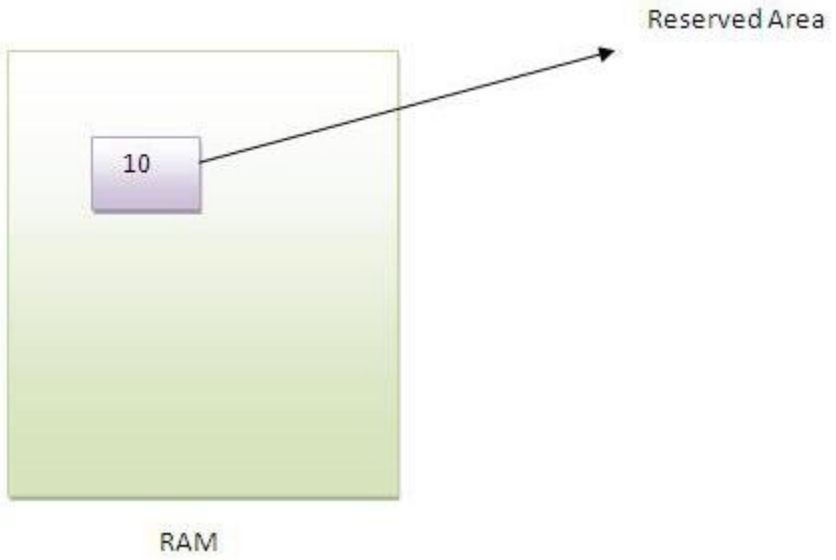
C#

SIMPLE C# PROGRAM

```
Using system;
Namespace  Program
{
 public class simple
{
          static void main(string[]   args)
          {
                   console.WriteLine(“Hai”);
                   Console.Readkey();
           }
}
}
```

VARIABLE

Variable is name of reserved area allocated in memory.



Declaration of a variable

```
<data_type> <variable_name>=<value>;
            Eg:  Int a=10;
```

Data types

- It defines the type or size of a data .
- Different types:
- Int
- Char
- Float
- double

Data Type	Default Value	Default size
boolean	false	1 bit
char	'\u0000'	2 byte
byte	0	1 byte
short	0	2 byte
int	0	4 byte
long	0L	8 byte
float	0.0f	4 byte
double	0.0d	8 byte

Keywords and constants

- Predefined or reserved words
- Cannot use as an identifier

Eg: absract,this,double,byte

Constants

- has a fixed value
- Constants can be three types:

- Interger constants
- Floating point
- Character constants

Identifiers

- Names of various program elements that uniquely identify an element.
- Keyword cannot use

  Eg: name,age,…..

C# - Operators

An operator is a symbol that tells the compiler to perform specific mathematical or logical manipulations.

- Arithmetic Operators
- Relational Operators
- Logical Operators
- Assignment Operators
- Bitwise Operators
- Misc Operators

Arithmetic Operators

- + Adds two operands
- - Subtracts second operand from the first
- * Multiplies both operands
- / Divides numerator by de-numerator
- % Modulus Operator and remainder of after an integer division
- ++ Increment operator increases integer value by one
- -- Decrement operator decreases integer value by one

Relational Operators

- **==** Checks if the values of two operands are equal or not, if yes then condition becomes true.
- **!=** Checks if the values of two operands are equal or not, if values are not equal then condition becomes true.
- **>** Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.
- **<** Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.
- **>=** Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.
- **<=** Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.

Logical Operators

- **&&** Logical AND operator. If both the operands are non zero then condition becomes true.
- **||** Logical OR Operator. If any of the two operands is non zero then condition becomes true.
- **!** Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false.!

Assignment Operators

- = Simple assignment operator
- += Add AND assignment operator
- -= Subtract AND assignment operator
- *= Multiply AND assignment operator
- /= Divide AND assignment operator
- %= Modulus AND assignment operator

Bitwise Operators

Bitwise operator works on bits and perform bit by bit operation.

- **&** Binary AND Operator
- **|** Binary OR Operator
- **^** Binary XOR Operator
- **~** Binary Ones Complement Operator
- **<<** Binary Left Shift Operator.
- **>>** Binary Right Shift Operator.

Misc Operators

- Sizeof - Returns the size of a data type.
- Typeof - Returns the type of a class.
- ? : - Conditional Expression

OPERATORS

Operators	Precedence
postfix	*expr++ expr--*
unary	*++expr --expr +expr -expr* ~ !
multiplicative	* / %
additive	+ -
shift	<< >> >>>
relational	< > <= >= instanceof
equality	== !=
bitwise AND	&
bitwise exclusive OR	^
bitwise inclusive OR	|
logical AND	&&
logical OR	||
ternary	? :
assignment	= += -= *= /= %= &= ^= |= <<= >>= >>>=

BRANCHING STATEMENTS

- **if Statement**

An if statement consists of a Boolean expression followed by one or more statements.

Syntax:-

```
if(Boolean_Expression)
   {
      //statement to be executed
     }
```

The If...Else Statement:-

An if statement can be followed by an optional *else* statement, which executes when the Boolean expression is false.

Syntax:-

```
if(Boolean_expression){
//Executes when the Boolean expression is true
 }
else
{
//Executes when the Boolean expression is false
 }
```

The if...else if...else Statement:-

An if statement can be followed by an optional *else if...else* statement,
which is very useful to test various conditions using single if...else if statement.

Syntax:-

```
if(Boolean_expression 1)
{
//Executes when the Boolean expression 1 is true
 }
else if(Boolean_expression 3){
//Executes when the Boolean expression 3 is true
 }else
 {
 //Executes when the none of the above condition is true.
}
```

Nested if...else Statement:-

It is always legal to nest if-else statements
which means you can use one if or else if statement
inside another if or else if statement.

Syntax:-

```
if(Boolean_expression 1)
{
//Executes when the Boolean expression 1 is true
if(Boolean_expression 2)
{
 //Executes when the Boolean expression 2 is true
 }
 }
```

The switch Statement:-

A switch statement allows a variable to be tested for equality against a list of values.
Each value is called a case,
and the variable being switched on is checked for each case.

Syntax:-

```
switch(expression)
{
 case value : //Statements
 break;
 //optional case value : //Statements
 break;
 //optional //You can have any number of case statements.
default : //Optional
 //Statements
 }
```

LOOPING STATEMENTS

There may be a situation when we need to execute a block of code several number of times, and is often referred to as a loop.

- while Loop
- do...while Loop
- for Loop

The while Loop:

A while loop is a control structure that allows you to repeat a task a certain number of times.

Syntax:-

```
while(Boolean_expression)
{
    //Statements
 }
```

The do...while Loop:-

A do...while loop is similar to a while loop, except that a do...while loop is guaranteed to execute at least one time.

Syntax:

```
do {
   //Statements
      }while(Boolean_expression);
```

The for Loop:

A for loop is a repetition control structure that allows you to efficiently write a loop that needs to execute a specific number of times.

Syntax:-

```
for(initialization; Boolean_expression; update)
{
//Statements
 }
```