C**Sharp**

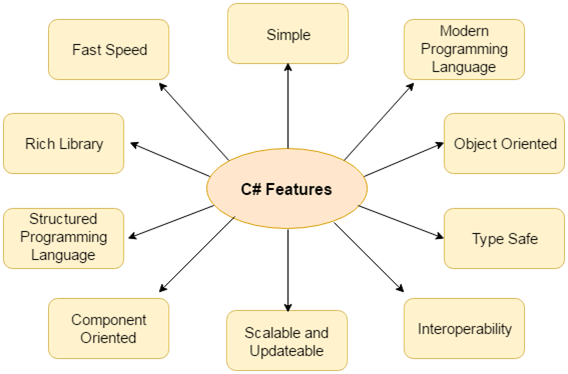
1.1 Difference between Java and C#

|  |  |  |
| --- | --- | --- |
| **No.** | **Java** | **C#** |
| 1) | Java is a **high level, robust, secured and object-oriented programming** language developed by Oracle. | C# is an **object-oriented programming** language developed by Microsoft that runs on .Net Framework. |
| 2) | Java programming language is designed to be run on a Java platform, by the help of Java Runtime Environment (JRE). | C# programming language is designed to be run on the Common Language Runtime (CLR). |
| 3) | Java type safety is safe. | C# type safety is unsafe. |
| 4) | In java, built-in data types that are passed by value are called **primitive types.** | In C#, built-in data types that are passed by value are called **simple types.** |
| 5) | Arrays in Java are direct specialization of **Object.** | Arrays in C# are specialization of **System.** |
| 6) | Java does not support **conditional compilation.** | C# supports conditional compilation using preprocessor directives. |
| 7) | Java doesn't support goto statement. | C# supports goto statement. |
| 8) | Java doesn't support **structures and unions.** | C# supports structures and unions. |
| 9) | Java supports checked exception and unchecked exception. | C# supports unchecked exception. |

1.2 C# Version and freamwork

| Version | .NET Framework | Visual Studio | Important Features |
| --- | --- | --- | --- |
| C# 1.0 | .NET Framework 1.0/1.1 | Visual Studio .NET 2002 | * Basic features |
| C# 2.0 | .NET Framework 2.0 | Visual Studio 2005 | * Generics * Partial types * Anonymous methods * Iterators * Nullable types * Private setters (properties) * Method group conversions (delegates) * Covariance and Contra-variance * Static classes |
| C# 3.0 | .NET Framework 3.0\3.5 | Visual Studio 2008 | * Implicitly typed local variables * Object and collection initializers * Auto-Implemented properties * Anonymous types * Extension methods * Query expressions * Lambda expressions * Expression trees * Partial Methods |
| C# 4.0 | .NET Framework 4.0 | Visual Studio 2010 | * Dynamic binding (late binding) * Named and optional arguments * Generic co- and contravariance * Embedded interop types |
| C# 5.0 | .NET Framework 4.5 | Visual Studio 2012/2013 | * Async features * Caller information |
| C# 6.0 | .NET Framework 4.6 | Visual Studio 2013/2015 | * Expression Bodied Methods * Auto-property initializer * nameof Expression * Primary constructor * Await in catch block * Exception Filter * String Interpolation |
| C# 7.0 | .NET Core | Visual Studio 2017 | * out variables * Tuples * Discards * Pattern Matching * Local functions * Generalized async return types * throw Expressions |

1.3 C# Features



### Type Safe

C# type safe code can only access the memory location that it has permission to execute. Therefore it improves a security of the program.

### Interoperability

Interoperability process enables the C# programs to do almost anything that a native C++ application can do.

### Scalable and Updateable

C# is automatic scalable and updateable programming language. For updating our application we delete the old files and update them with new ones.

### Component Oriented

C# is component oriented programming language. It is the predominant software development methodology used to develop more robust and highly scalable applications.

### Structured Programming Language

C# is a structured programming language in the sense that we can break the program into parts using functions. So, it is easy to understand and modify.

### Rich Library

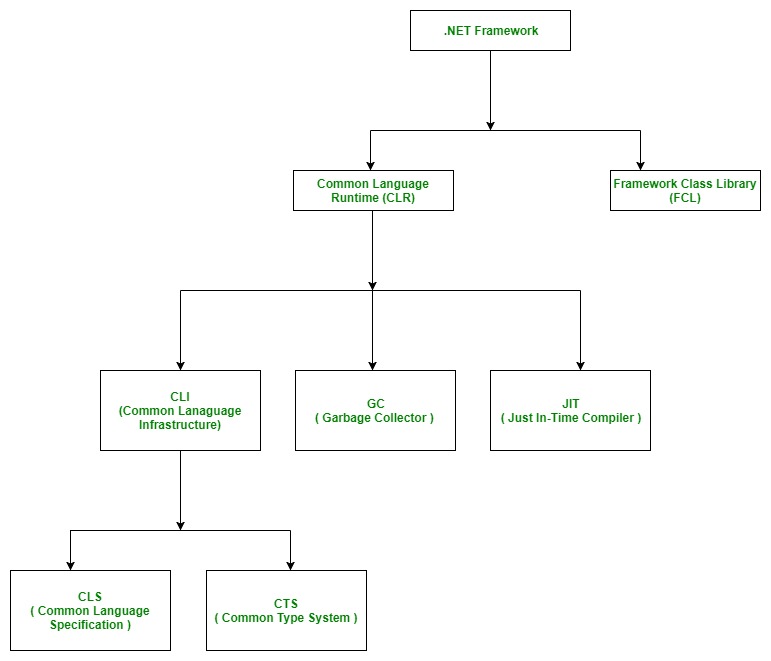
C# provides a lot of inbuilt functions that makes the development fast.

### Fast Speed

The compilation and execution time of C# language is fast.

**Programming Languages which are designed and developed by Microsoft are:**

* C#.NET
* VB.NET
* C++.NET
* J#.NET
* F#.NET
* JSCRIPT.NET
* WINDOWS POWERSHELL



# 1.4 NET Framework Class Library (FCL)

* The Framework Class Library or FCL provides the system functionality in the [**.NET Framework**](https://www.geeksforgeeks.org/introduction-to-net-framework/) as it has various classes, data types, interfaces, etc. to perform multiple functions and build different types of applications such as desktop applications, web applications, mobile applications, etc.
* **The Framework Class Library is integrated with the**[**Common Language Runtime (CLR)**](https://www.geeksforgeeks.org/common-language-runtime-clr-in-c-sharp/)**of the .NET framework** and is used by all the .NET languages such as [**C#**](https://www.geeksforgeeks.org/csharp-programming-language/), F#, Visual Basic .NET, etc.

| **NAMESPACE** | **DESCRIPTION** |
| --- | --- |
| Accessibility | The Accessibility namespace is a part of the managed wrapper for the COM accessibility interface. |
| Microsoft.Activities | The Microsoft.Activities namespace provides support for Windows Workflow Foundation applications. |
| Microsoft.CSharp | The Microsoft.CSharp namespace has support for compilation and code generation for the C# source code. |
| Microsoft.JScript | The Microsoft.JScript namespace has support for compilation and code generation for the JScript source code. |
| Microsoft.VisualBasic | The Microsoft.VisualBasic namespace has support for compilation and code generation for the VisualBasic source code. |
| System | The System namespace has base classes for definition of interfaces, data types, events, event handlers, attributes, processing exceptions etc. |
| System.Activities | The System.Activities namespace handles the creation and working with activities in the Window Workflow Foundation using various classes. |
| System.Collections | The System.Collections namespace has multiple standard, specialized, and generic collection objects that are defined using various types. |
| System.Configuration | The System.Configuration namespace handles configuration data using various types. This may include data in machine or application configuration files. |
| System.Data | The System.Data namespace accesses and manages data from various sources using different classes. |
| System.Drawing | The System.Drawing namespace handles GDI+ basic graphics functionality. Various child namespaces also handle vector graphics functionality, advanced imaging functionality, etc. |
| System.Globalization | The System.Globalization namespace handles language, country, calendars used, format patterns for dates, etc. using various classes. |
| System.IO | The System.IO namespaces support IO like data read/write into streams, data compression, communicate using named pipes etc. using various types. |
| System.Linq | The System.Linq namespace supports Language-Integrated Query (LINQ) using various types. |
| System.Media | The System.Media namespace handles sound files and accessing the sounds provided by the system using various classes. |
| System.Net | The System.Net namespace provides an interface for network protocols, cache policies for web resources, composing and sending e-mail etc. using various classes. |
| System.Reflection | The System.Reflection namespace gives a managed view of loaded methods, types, fields, etc. It can also create and invoke types dynamically. |
| System.Security | The System.Security namespace has the .NET security system and permissions. Child namespaces provide authentication, crytographic services etc. |
| System.Threading | The System.Threading namespace allows multithreaded programming using various types. |
| XamlGeneratedNamespace | The XamlGeneratedNamespace has compiler-generated types that are not used directly from the code. |

# 

# 1.5 Garbage Collection in C#

Automatic memory management is made possible by **Garbage Collection in .NET Framework**.

* When a class object is created at runtime, certain memory space is allocated to it in the heap memory.
* However, after all the actions related to the object are completed in the program, the memory space allocated to it is a waste as it cannot be used.
* In this case, garbage collection is very useful as it automatically releases the memory space after it is no longer required.
* Garbage collection will always work on **Managed Heap** and internally it has an Engine which is known as the **Optimization Engine**.

// example: explicit destruction/finalize  
using System;  
class MyClass : IDisposable  
{  
    public MyClass() //default ctor  
    {  
        this.iNumber = 0;  
        System.Console.WriteLine("ctor:MyClass {0}", iNumber);  
    }  
    public MyClass(Int32 iNumber) // specialized ctor  
    {  
        this.iNumber = iNumber;  
        System.Console.WriteLine("ctor:MyClass {0}", iNumber);  
    }  
    ~MyClass() // dtor or finalize  
    {  
        System.Console.WriteLine("dtor:~MyClass {0}", iNumber);  
    }  
    public void Dispose() // helper finalize function  
    {  
        // here you can free the resources you allocated explicitly  
        System.GC.SuppressFinalize(this);  
    }  
}  
class main  
{  
    static void Main()  
    {  
      MyClass myClass1 = new MyClass();  
      MyClass myClass2 = new MyClass(19);  
      myClass1.Dispose(); // myClass1 is explicitly exposed.  
      System.GC.Collect ();  
      System.GC.WaitForPendingFinalizers (); // myClass2 is implicitly exposed by GC.  
      Console.ReadLine ();  
    }  
}

***1.6 C# Data Type:***

The variables in C# are categorized into the following types −

* Value types
* Reference types
* Pointer types

Value type variables can be assigned a value directly. They are derived from the class **System.ValueType**.

| **Type** | **Represents** | **Range** | **Default Value** |
| --- | --- | --- | --- |
| Dbool | Boolean value | True or False | False |
| byte | 8-bit unsigned integer | 0 to 255 | 0 |
| char | 16-bit Unicode character | U +0000 to U +ffff | '\0' |
| decimal | 128-bit precise decimal values with 28-29 significant digits | (-7.9 x 1028 to 7.9 x 1028) / 100to 28 | 0.0M |
| double | 64-bit double-precision floating point type | (+/-)5.0 x 10-324 to (+/-)1.7 x 10308 | 0.0D |
| float | 32-bit single-precision floating point type | -3.4 x 1038 to + 3.4 x 1038 | 0.0F |
| int | 32-bit signed integer type | -2,147,483,648 to 2,147,483,647 | 0 |

*When a value type is converted to object type, it is called****boxing***

*When an object type is converted to a value type, it is called****unboxing****.*

**1.6.1 Reference Type**

**Built-in** reference types are: **object**, **dynamic,** and **string**.

# C# Implicitly-Typed Local Variable - var

C# 3.0 introduced the implicit typed local variable "var". Var can only be defined in a method as a local variable.

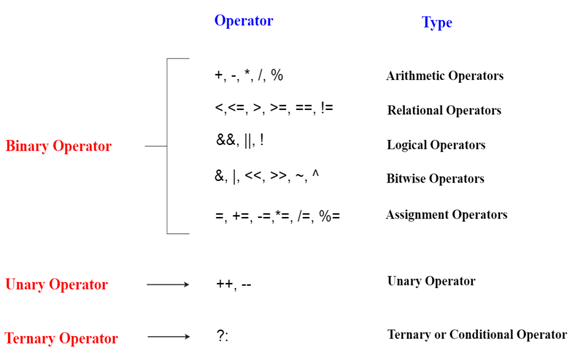
Example: Explicitly Typed Variable

int i = 100;// explicitly typed

var j = 100; // implicitly typed

var j; // Compile-time error: Implicitly-typed variables must be initialized

* **Implicit type conversion** − These conversions are performed by C# in a type-safe manner. For example, are conversions from smaller to larger integral types and conversions from derived classes to base classes.
* **Explicit type conversion** − These conversions are done explicitly by users using the pre-defined functions. Explicit conversions require a cast operator.



### *1.7 Array*

### 1.7.1 C# Single Dimensional Array

**int**[] arr = **new** **int**[5];//creating array

# 1.7.2 C# Multidimensional Arrays

**int**[,] arr=**new** **int**[3,3];//declaration of 2D array

arr[0,1]=10;//initialization

arr[1,2]=20;

arr[2,0]=30;

0 10 0

0 0 20

30 0 0

**int**[,,] arr=**new** **int**[3,3,3];//declaration of 3D array

### 1.7.3 Declaration of Jagged array

**int**[][] arr = **new** **int**[2][];

arr[0] = **new** **int**[4];

arr[1] = **new** **int**[6];

arr[0] = **new** **int**[4] { 11, 21, 56, 78 };

arr[1] = **new** **int**[6] { 42, 61, 37, 41, 59, 63 };

# 

# *1.8 C# Params*

In C#, **params** is a keyword which is used to specify a parameter that takes variable number of arguments. It is useful when we don't know the number of arguments prior. Only one params keyword is allowed and no additional parameter is permitted after params keyword in a function declaration.

**class** Program

    {

        // User defined function

**public** **void** Show(**params** **int**[] val) // Params Paramater

        {

**for** (**int** i=0; i<val.Length; i++)

            {

                Console.WriteLine(val[i]);

            }

        }

        // Main function, execution entry point of the program

**static** **void** Main(**string**[] args)

        {

            Program program = **new** Program(); // Creating Object

            program.Show(2,4,6,8,10,12,14); // Passing arguments of variable length

        }

    }

# *1.9 C# Destructor*

A destructor works opposite to constructor; it destructs the objects of classes. It can be defined only once in a class. Like constructors, it is invoked automatically.

Note: C# destructor cannot have parameters. Moreover, modifiers can't be applied on destructors.

***1.10 C# File Stream***

Using System.IO;

### *1.10.1 Writing single byte into file*

FileStream f = **new** FileStream("e:\\b.txt", FileMode.OpenOrCreate);//creating file stream

f.WriteByte (65); //writing byte into stream

f.Close (); //closing stream

### *1.10.2 reading all bytes from file*

FileStream f = **new** FileStream("e:\\b.txt", FileMode.OpenOrCreate);

**int** i = 0;

**while** ((i = f.ReadByte()) != -1)

{

    Console.Write ((**char**) i);

}

f.Close ();

*1.10.3 C# StreamWriter*

FileStream f = new FileStream("e:\\output.txt", FileMode.Create);

StreamWriter s = new StreamWriter(f);

s.WriteLine ("hello c#");

s.Close ();

f.Close ();

# *1.10.4 C# StreamReader*

FileStream f = new FileStream("e:\\output.txt", FileMode.OpenOrCreate);

StreamReader s = new StreamReader(f);

string line=s.ReadLine();

Console.WriteLine(line);

s.Close();

f.Close();

### *1.10.5 C# TextWriter Example*

using (TextWriter writer = File.CreateText("e:\\f.txt"))

{

writer.WriteLine("Hello C#");

writer.WriteLine("C# File Handling by JavaTpoint");

}

using (TextReader tr = File.OpenText("e:\\f.txt"))

{

Console.WriteLine(tr.ReadToEnd());

}

### *1.10.6 C# BinaryWriter Example*

using (BinaryWriter writer = new BinaryWriter(File.Open(fileName, FileMode.Create)))

{

writer.Write(2.5);

writer.Write("this is string data");

writer.Write(true);

}

### *1.10.7 C# BinaryReader Example*

using (BinaryReader reader = new BinaryReader(File.Open("e:\\binaryfile.dat", FileMode.Open)))

{

Console.WriteLine("Double Value : " + reader.ReadDouble());

Console.WriteLine("String Value : " + reader.ReadString());

Console.WriteLine("Boolean Value : " + reader.ReadBoolean());

}

### *1.10.8 C# FileInfo Properties*

|  |  |
| --- | --- |
| **Properties** | **Description** |
| Attributes | It is used to get or set the attributes for the current file or directory. |
| CreationTime | It is used to get or set the creation time of the current file or directory. |
| Directory | It is used to get an instance of the parent directory. |
| DirectoryName | It is used to get a string representing the directory's full path. |
| Exists | It is used to get a value indicating whether a file exists. |
| FullName | It is used to get the full path of the directory or file. |
| Length | It is used to get the size in bytes of the current file. |
| Name | It is used to get the name of the file. |

### *1.10.9 C# FileInfo Methods*

|  |  |
| --- | --- |
| **Method** | **Description** |
| AppendText() | It is used to create a StreamWriter that appends text to the file represented by this instance of the FileInfo. |
| CopyTo(String) | It is used to copy an existing file to a new file. |
| Create() | It is used to create a file. |
| CreateText() | It is used to create a StreamWriter that writes a new text file. |
| Decrypt() | It is used to decrypt a file that was encrypted by the current account using the Encrypt method. |
| Delete() | It is used to permanently delete a file. |
| Encrypt() | It is used to encrypt a file so that only the account used to encrypt the file can decrypt it. |
| MoveTo(String) | It is used to move a specified file to a new specified location. |
| Open(FileMode) | It is used to open a file in the specified mode. |
| OpenRead() | It is used to create a read-only FileStream. |
| OpenText() | It is used to create a StreamReader with UTF8 encoding that reads from an existing text file. |
| OpenWrite() | It is used to create a write-only FileStream. |
| Refresh() | It is used to refresh the state of the object. |
| Replace(String,String) | It is used to replace the contents of a specified file with the file described by the current FileInfo object. |
| ToString() | It is used to return the path as a string |

### *1.10.10 C# DirectoryInfo Example:*

// Provide directory name with complete location.

DirectoryInfo directory = new DirectoryInfo(@"F:\javatpoint");

try

{

// Check, directory exist or not.

if (directory.Exists)

{

Console.WriteLine("Directory already exist.");

return;

}

// Creating a new directory.

directory.Create();

Console.WriteLine("The directory is created successfully.");

}

### *1.10.11 C# DirectoryInfo Example: Deleting Directory*

DirectoryInfo directory = new DirectoryInfo(@"F:\javatpoint");

try

{

// Deleting directory

directory.Delete();

Console.WriteLine("The directory is deleted successfully.");

}

## **What is Serialization?**

Serialization is the process of bringing an object into a form that it can be written on stream. It's the process of converting the object into a form so that it can be stored on a file, database, or memory; or, it can be transferred across the network. Its main purpose is to save the state of the object so that it can be recreated when needed.

## **What is Deserialization?**

As the name suggests, deserialization is the reverse process of serialization. It is the process of getting back the serialized object so that it can be loaded into memory. It resurrects the state of the object by setting properties, fields etc.

First, we go and import the name space for serialization.

***Using System.Runtime.Serialization;***

Then Import the formatter

***using System.Runtime.Serialization.Formatters.Binary;***

**Types**

* Binary Serialization
* XML Serialization
* JSON Serialization

Advantages of serialization

1. Passing an object from one application to another.
2. Passing an object from one domain to another.

Binary Serialization :

1. Employee emp = **new** Employee {
2. name = textBoxName.Text,
3. phone = textBoxPhone.Text,
4. dob = dateTimePickerDoB.Value,
5. department = textBoxDepartment.Text,
6. salary = Convert.ToInt32(textBoxSalary.Text),
7. additionalInfo = "We don't want it to serialize"
8. };
9. BinaryFormatter bf = **new** BinaryFormatter();
11. FileStream fsout = **new** FileStream("employee.binary", FileMode.Create, FileAccess.Write, FileShare.None);
12. **try**
13. {
14. **using** (fsout)
15. {
16. bf.Serialize(fsout, emp);
17. label6.Text = "Object Serialized";
18. }
19. }

Binary Deserialization:

1. Employee emp = **new** Employee();
3. BinaryFormatter bf = **new** BinaryFormatter();
5. FileStream fsin = **new** FileStream("employee.binary", FileMode.Open, FileAccess.Read, FileShare.None);
6. **try**
7. {
8. **using** (fsin)
9. {
10. emp = (Employee) bf.Deserialize(fsin);
11. label6.Text = "Object Deserialized";
13. textBoxName.Text = emp.name;
14. textBoxPhone.Text = emp.phone;
15. dateTimePickerDoB.Value = emp.dob;
16. textBoxDepartment.Text = emp.department;
17. textBoxSalary.Text = emp.salary.ToString();
18. }
19. }

## ***1.10.12 C# Binary Serialization***

using System.Runtime.Serialization.Formatters.Binary

FileStream stream = new FileStream("e:\\sss.txt", FileMode.OpenOrCreate);

BinaryFormatter formatter=new BinaryFormatter();

Student s = new Student(101, "sonoo");

formatter.Serialize(stream, s);

stream.Close();

***1.10.13 C# Deserialization***

FileStream stream = new FileStream("e:\\sss.txt", FileMode.OpenOrCreate);

BinaryFormatter formatter=new BinaryFormatter();

Student s=(Student)formatter.Deserialize(stream);

Console.WriteLine("Rollno: " + s.rollno);

Console.WriteLine("Name: " + s.name);

stream.Close();

## 

## **1.11 C# Exception Classes**

System.DivideByZeroException

System.NullReferenceException

System.InvalidCastException

System.IO.IOException

System.FieldAccessException

# *1.11.1 C# Exception Filters*

* It is introduced in version C# 6.0. It allows us to specify condition along with a catch block.
* C# provides when keyword to apply a condition (or filter) along with catch block.

### 1.11.2 C# Exception Filter Syntax

catch (ArgumentException e) when (e.ParamName == "?"){ }

try

{

int[] a = new int[5];

a[10] = 12;

}

catch(Exception e) when(e.GetType().ToString() == "System.IndexOutOfRangeException")

{

// Executing some other task

SomeOtherTask();

}

# 1.11.3 C# User-Defined Exceptions

C# allows us to create user-defined or custom exception. It is used to make the meaningful exception. To do this, we need to inherit Exception class.

public class InvalidAgeException : Exception

{

public InvalidAgeException(String message)

: base(message)

{

}

}

try

{

// validate(12);

}

catch (InvalidAgeException e)

{

Console.WriteLine(e);

}

## **1.12 Collections in C#**

## **Types of Collections in C#**

* **System.Collections.Generic** classes
* **System.Collections** classes (Now deprecated)

| Non-generic Collections | Usage |
| --- | --- |
| [ArrayList](https://www.tutorialsteacher.com/csharp/csharp-arraylist) | ArrayList stores objects of any type like an array. However, there is no need to specify the size of the ArrayList like with an array as it grows automatically. |
| [SortedList](https://www.tutorialsteacher.com/csharp/csharp-sortedlist) | SortedList stores key and value pairs. It automatically arranges elements in ascending order of key by default. C# includes both, generic and non-generic SortedList collection. |
| [Stack](https://www.tutorialsteacher.com/csharp/csharp-stack) | Stack stores the values in LIFO style (Last In First Out). It provides a Push() method to add a value and Pop() & Peek() methods to retrieve values. C# includes both, generic and non-generic Stack. |
| [Queue](https://www.tutorialsteacher.com/csharp/csharp-queue) | Queue stores the values in FIFO style (First In First Out). It keeps the order in which the values were added. It provides an Enqueue() method to add values and a Dequeue() method to retrieve values from the collection. C# includes generic and non-generic Queue. |
| [Hashtable](https://www.tutorialsteacher.com/csharp/csharp-hashtable) | Hashtable stores key and value pairs. It retrieves the values by comparing the hash value of the keys. |
| BitArray | BitArray manages a compact array of bit values, which are represented as Booleans, where true indicates that the bit is on (1) and false indicates the bit is off (0). |

## ***1.13 C# Anonymous Methods***

public delegate void AnonymousFun();

static void Main(string[] args)

{

AnonymousFun fun = delegate () {

Console.WriteLine("This is anonymous function");

};

fun();

}

***1.14 Interface***

* Default access specifier of the Interface is public.
* Internal access specifier support for the Interface.
* Private and protected access specifier not support for the Interface.

***1.14.1 How to create Interface:***

Example 1:

interface Test

{

void Test();

}

class ClsTest : Test

{

public void Test()

{

Console.WriteLine("Interface Testing");

}

}

class Program : ClsTest

{

static void Main(string[] args)

{

Program objp = new Program();

objp.Test();

Console.ReadLine();

}

}

Output:

Interface Testing

Example 2:

interface Test

{

void Test();

}

class ClsTest : Test

{

public void Test()

{

Console.WriteLine("Interface Testing");

}

}

class Program : ClsTest

{

public void Test()

{

Console.WriteLine("Interface Testing inside another");

}

static void Main(string[] args)

{

Program objp = new Program();

objp.Test();

Console.ReadLine();

}

}

Output:

Interface Testing inside another

***1.14.2 How to implement private interface in c#***

public class ClsName

{

private interface Test

{

void Test();

}

public class ClsTest : Test

{

public void Test()

{

Console.WriteLine("Interface Testing");

}

}

}

class Program : ConsoleApplication2.ClsName.ClsTest

{

static void Main(string[] args)

{

ConsoleApplication2.ClsName.ClsTest obj = new ConsoleApplication2.ClsName.ClsTest();

obj.Test(); //Output :- Interface Testing

Program objp = new Program();

objp.Test();//Output :- Interface Testing

Console.ReadLine();

}

}

***1.14.3 How to create variable in interface:***

* Variable Inside interface always call in the class

public class ClsInterfaceWithVariable

{

interface IVariable

{

int Vari { get; set; }

}

public class ClsVariable : IVariable

{

int IVariable.Vari { get; set; }

}

public void CallVariable()

{

ClsVariable objVariable = new ClsVariable();

((IVariable)objVariable).Vari=100;

Console.WriteLine("Return Interface Variable : " + ((IVariable)objVariable).Vari);

}

}

class Program

{

static void Main(string[] args)

{

ClsInterfaceWithVariable obj1 = new ClsInterfaceWithVariable();

obj1.CallVariable();//Output :- Return Interface Variable : 100

Console.ReadLine();

}

}

***1.14.4 How to declare same method name in two interfaces***

interface G1

{

void mymethod();

}

interface G2

{

void mymethod();

}

class Geeks : G1, G2

{

void G1.mymethod()

{

Console.WriteLine("Interface 1 Method");

}

void G2.mymethod()

{

Console.WriteLine("Interface 2 Method");

}

}

class Program

{

static void Main(string[] args)

{

G1 obj = new Geeks();

obj.mymethod(); //Output :- Interface 1 Method

G2 ob = new Geeks();

ob.mymethod(); //Output :- Interface 2 Method

Console.ReadLine();

}

}

***1.15 Constructor:***

* Type of constructor:
* Default constructor
* Parameterize constructor
* Copy constructor
* Static constructor
* Private constructor

1. Default constructor

Case 1)

public class ClsPublicConstructor

{

public ClsPublicConstructor()

{

Console.WriteLine("Default Constructor ");

}

}

class Program

{

static void Main(string[] args)

{

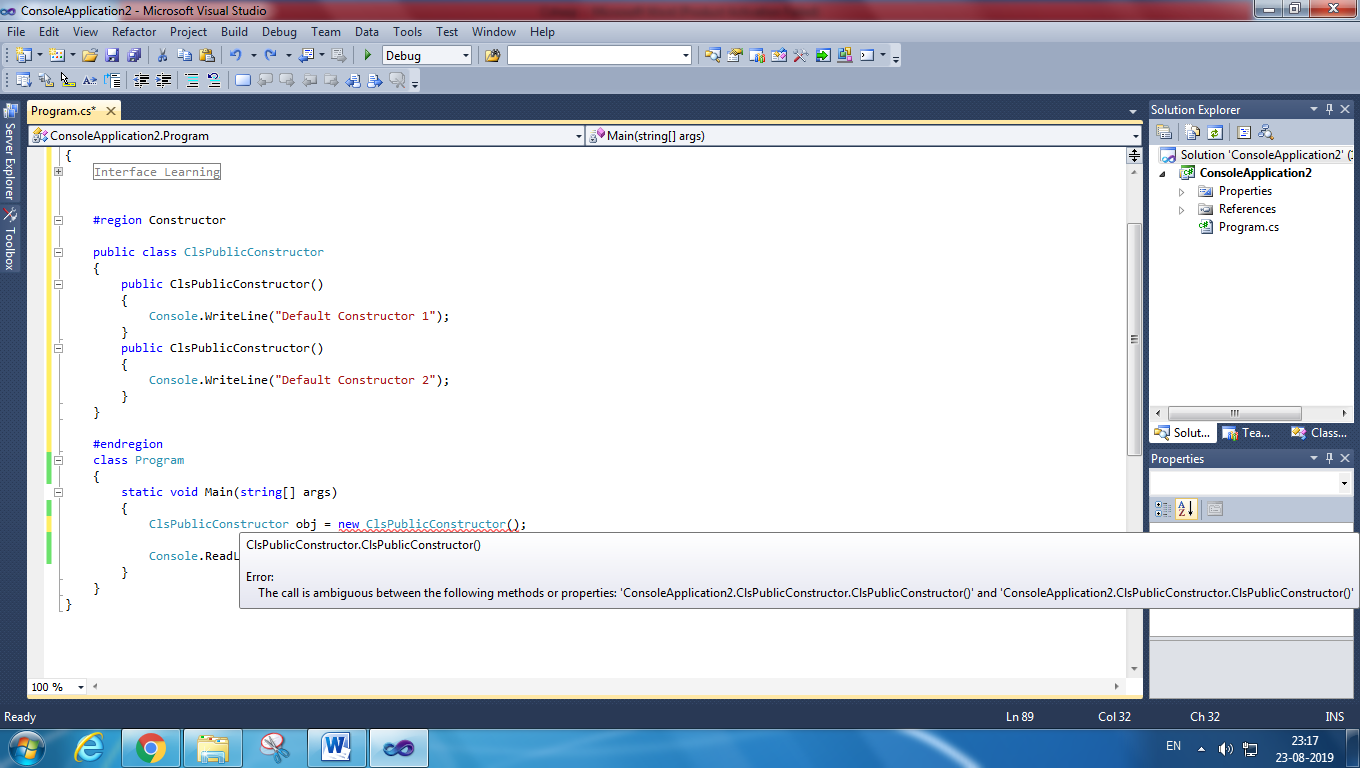
ClsPublicConstructor obj = new ClsPublicConstructor();//output :- Default Constructor

Console.ReadLine();

}

}

Case 2) More than one default, Parameterize constructor not possible to create. If create then will get compile time error



1. *Parameterize constructor*

public class ClsPublicConstructor

{

public ClsPublicConstructor(string name)

{

Console.WriteLine(name);

}

}

class Program

{

static void Main(string[] args)

{

ClsPublicConstructor obj = new ClsPublicConstructor("Default Constructor"); //output :- Default Constructor

Console.ReadLine();

}

}

1. *Copy constructor*

public class ClsCopyConstructor

{

string Name;

public ClsCopyConstructor(string name)

{

this.Name = name;

}

public ClsCopyConstructor(ClsCopyConstructor obj)

{

Console.WriteLine("Copy constructor : " + obj.Name);

}

}

class Program

{

static void Main(string[] args)

{

ClsCopyConstructor obj = new ClsCopyConstructor("Copy Constructor"); //output :- Copy Constructor

ClsCopyConstructor copyobj = new ClsCopyConstructor(obj);

ClsCopyConstructor obj1 = new ClsCopyConstructor("Copy Constructor 1"); //output :- Copy Constructor 1

ClsCopyConstructor copyobj1 = new ClsCopyConstructor(obj1);

Console.ReadLine();

}

}

***Static, Default and Parametrize Constructor:***

public class Clsconstructor

{

public Clsconstructor()

{

Console.WriteLine("Default constructor");

}

public Clsconstructor(string name)

{

Console.WriteLine(name);

}

static Clsconstructor()

{

Console.WriteLine("Static constructor");

}

}

class Program

{

static void Main(string[] args)

{

Clsconstructor obj = new Clsconstructor();

Console.ReadLine();

}

}

//output :-

Default Constructor

Static Constructor

public class Clsconstructor

{

public Clsconstructor()

{

Console.WriteLine("Default constructor");

}

public Clsconstructor(string name)

{

Console.WriteLine(name);

}

static Clsconstructor()

{

Console.WriteLine("Static constructor");

}

}

class Program

{

static void Main(string[] args)

{

Clsconstructor obj = new Clsconstructor("Paramertize constructor");

Console.ReadLine();

}

//output :-

Static Constructor

Paramertize Constructor

***Variable Define in default constructor***

public class Clsconstructorwithvariable

{

public string publicvariable;

private string privatevariable;

static string staticvariable;

internal string internalvariable;

protected string protectedvariable;

public Clsconstructorwithvariable()

{

publicvariable = "public constructor with public variable";

privatevariable = "public constructor with private variable";

staticvariable = "public constructor with static variable";

internalvariable = "public constructor with internal variable";

protectedvariable = "public constructor with protected variable";

}

public void Display()

{

Console.WriteLine(publicvariable);

Console.WriteLine(privatevariable);

Console.WriteLine(staticvariable);

Console.WriteLine(internalvariable);

Console.WriteLine(protectedvariable);

}

}

class Program

{

static void Main(string[] args)

{

Clsconstructorwithvariable obj = new Clsconstructorwithvariable();

obj.Display();

Console.ReadLine();

}

}

//output :-

Public Constructor with public variable

Public Constructor with private variable

Public Constructor with static variable

Public Constructor with internal variable

Public Constructor with protected variable

***Note: Static constructor can only allow static variable only.***

public class Clsconstructorwithvariable

{

public string publicvariable;

private string privatevariable;

static string staticvariable;

internal string internalvariable;

protected string protectedvariable;

public Clsconstructorwithvariable(string \_publicvariable, string \_privatevariable, string \_staticvariable, string \_internalvariable, string \_protectedvariable)

{

this.publicvariable = \_publicvariable;

this.privatevariable = \_privatevariable;

staticvariable = \_staticvariable;

this.internalvariable = \_internalvariable;

this.protectedvariable = \_protectedvariable;

}

public void Display()

{

Console.WriteLine(publicvariable);

Console.WriteLine(privatevariable);

Console.WriteLine(staticvariable);

Console.WriteLine(internalvariable);

Console.WriteLine(protectedvariable);

}

}

class Program

{

static void Main(string[] args)

{

Clsconstructorwithvariable obj = new Clsconstructorwithvariable("public constructor with public variable",

"public constructor with private variable",

"public constructor with static variable",

"public constructor with internal variable",

"public constructor with protected variable");

obj.Display();

Console.ReadLine();

}

}

//output :-

Public Constructor with public variable

Public Constructor with private variable

Public Constructor with static variable

Public Constructor with internal variable

Public Constructor with protected variable

1. ***Static constructor***

A static constructor is used to initialize any static data, or to perform a particular action that needs performed once only. It is called automatically before the first instance is created or any static members are referenced.

public class ClsStaticConstructor

{

static ClsStaticConstructor()

{

Console.WriteLine("Static constructor ");

}

}

class Program

{

static void Main(string[] args)

{

ClsStaticConstructor obj = new ClsStaticConstructor();//Output : Static constructor

Console.ReadLine();

}

}

public class ClsStaticConstructor

{

static ClsStaticConstructor()

{

Console.WriteLine("Static constructor ");

}

public ClsStaticConstructor(string name)

{

Console.WriteLine(name);

}

}

class Program

{

static void Main(string[] args)

{

ClsStaticConstructor obj = new ClsStaticConstructor("Paratamertize constructor");//Output : Static constructor Paratamertize constructor

Console.ReadLine();

}

}

1. ***Private Constructor :***

* If create private constructor not possible to create object of the class
* In same class we can create private as well as parameterize constructor then that object is execute.

Case 1:

public class ClsPrivateConstructor

{

private ClsPrivateConstructor()

{

Console.WriteLine("Private constructor");

}

public ClsPrivateConstructor(string name)

{

Console.WriteLine("constructor : " + name);

}

}

class Program

{

static void Main(string[] args)

{

ClsPrivateConstructor obj = new ClsPrivateConstructor("Paramterzie constructor");//Output : constructor : Paramterzie constructor

Console.ReadLine();

}

}

Case 2:

public class ClsPrivateConstructor

{

private ClsPrivateConstructor()

{

Console.WriteLine("Private constructor");

}

public ClsPrivateConstructor(string name)

{

Console.WriteLine("constructor : " + name);

}

static ClsPrivateConstructor()

{

Console.WriteLine("Static constructor ");

}

}

class Program

{

static void Main(string[] args)

{

ClsPrivateConstructor obj = new ClsPrivateConstructor("Paramterzie constructor");

Console.ReadLine();

}

}

//Output :

Static Constructor

constructor : Parameterzie constructor

public class ClsPrivateConstructor

{

private ClsPrivateConstructor()

{

Console.WriteLine("Private constructor");

}

public static double rate;

public static void Increment()

{

rate++;

}

}

class Program

{

static void Main(string[] args)

{

ClsPrivateConstructor.rate = 10;

ClsPrivateConstructor.Increment();

Console.WriteLine("Increment value : " + ClsPrivateConstructor.rate);//output Increment value : 11

Console.ReadLine();

}

}

1. If class with private constructor and sealed class cannot be inherited, then what is the use of class which cannot be inherited?

That means the developer wants to create a strict singleton class which can not be inherited further. And as said previously private constructor can be excused as we have static classes now. So private constructor + sealed means pure static class

**How to call Private Constructor in c#**

public class ClsPrivateConstructor

{

public int value;

private ClsPrivateConstructor()

{

value = 1;

Console.WriteLine("Private constructor");

}

}

class Program

{

static void Main(string[] args)

{

var constructor = typeof(ClsPrivateConstructor).GetConstructor(BindingFlags.NonPublic | BindingFlags.Instance, null, new Type[0], null);

var instance = (ClsPrivateConstructor)constructor.Invoke(null); //output :value 1

Console.ReadLine();

}

}

***1.16 Access specifier :***

1. Public modifier : we can access to anywhere
2. Private modifier : access with in class
3. Protected modifier : access to child class
4. Internal modifier : access to current assembly
5. Protected internal modifier: combination of protected and internal.

**How to Access Private Method**

class PrivateMethodClass

{

private void PrivateMethod()

{

Console.WriteLine("This is a Private Method");

}

}

class Program

{

static void Main(string[] args)

{

typeof(PrivateMethodClass).GetMethod("PrivateMethod", BindingFlags.NonPublic | BindingFlags.Instance).Invoke(new PrivateMethodClass(), null);

Console.ReadLine();

}

}

output :This is a Private Method.

public class PrivateMethodAccess

{

public void AddNumber()

{

Add();

}

private void Add()

{

int a, b;

a = b = 5;

int c = a + b;

Console.WriteLine(c.ToString());

}

}

class Program

{

static void Main(string[] args)

{

PrivateMethodAccess obj = new PrivateMethodAccess();

obj.AddNumber();

Console.ReadLine();

}

}

output :10

**How to Access Private Variable**: **By using Public Method**

class PrivateVariable

{

private int i = 10;

public void DisplayVariable()

{

Console.Write("The Value of Private Variable=" + i);

}

}

class Program

{

static void Main(string[] args)

{

PrivateVariable objPrivateVariable = new PrivateVariable();

objPrivateVariable.DisplayVariable();

Console.ReadLine();

}

}

output :The value of private Variable=10

**How to Access Private Variable** : **By Using Inner class**

1. **class** Outerclass
2. {
3. **private** **int** i = 10;
4. **private** **int** j = 20;
5. **class** Innerclass
6. {
7. **static** **void** Main()
8. {
9. Outerclass objouter = **new** Outerclass();
10. **int** Result = objouter.i + objouter.j;
11. Console.Write("Sum=" + Result);
12. Console.ReadLine();
13. }
14. }
15. }

**How to Access Private Variable** : **By Using Properties**

class Employee

{

private int \_EmpID = 1001;

public int EmpID

{

get

{

return \_EmpID;

}

}

}

class Program

{

static void Main(string[] args)

{

Employee objEmployee = new Employee();

Console.WriteLine("Employee ID: " + objEmployee.EmpID);

Console.ReadLine();

}

}

* **The internal access specifier hides its member variables and methods from other classes and objects, that is resides in other namespace.**
* **The variable or classes that are declared with internal can be access by any member within application.**
* **It is the default access specifiers for a class in C# programming**
* **enum:**The default and only access modifier supported is public.
* **class:**The default access for a class is private. It may be explicitly defined using any of the access modifiers.
* **interface:**The default and only access modifier supported is public.
* **struct:**The default access is private with public and internal supported as well.

***1.17 Static Class***

static class Class\_Name

{

// static data members

// static method

}

Example

static class Author

{

public static string A\_name = "Ankita";

public static string L\_name = "CSharp";

public static int T\_no = 84;

public static void details()

{

Console.WriteLine("The details of Author is:");

}

}

public class GFG

{

static public void Main()

{

Author.details();

Console.WriteLine("Author name : {0} ", Author.A\_name);

Console.WriteLine("Language : {0} ", Author.L\_name);

Console.WriteLine("Total number of articles : {0} ",

Author.T\_no);

}

}

Output:

The details of Author is:

Author name : Ankita

Language : CSharp

Total number of articles : 84

***Cannot Inherit Static class in c#***

// declaring a static class

public static class GFG

{

static void display()

{

Console.WriteLine("Static Method of class GFG");

}

}

// trying to inherit the class GFG

// it will give error as static

// class can't be inherited

class GFG2 : GFG

{

public static void Main(String[] args)

{

}

}

**Difference between static and non-static class**

|  |  |
| --- | --- |
| **STATIC CLASS** | **NON-STATIC CLASS** |
| Static class is defined using static keyword. | Non-Static class is not defined by using static keyword. |
| In static class, you are not allowed to create objects. | In non-static class, you are allowed to create objects using new keyword. |
| The data members of static class can be directly accessed by its class name. | The data member of non-static class is not directly accessed by its class name. |
| Static class always contains static members. | Non-static class may contain both static and non-static methods. |
| Static class does not contain an instance constructor. | Non-static class contains an instance constructor. |
| Static class cannot inherit from another class. | Non-static class can be inherited from another class. |

***1.18 Abstract***

public abstract class AbstractCls

{

public AbstractCls()

{

Console.WriteLine("public constructor contain in abstract class");

}

public abstract void AbstractMethod();

public void nonAbstractMethod()

{

Console.WriteLine("Non abstract method");

}

}

public class child : AbstractCls

{

public child()

{

Console.WriteLine("public constructor contain in non abstract class");

}

public override void AbstractMethod()

{

Console.WriteLine("Abstract method");

}

}

class Program

{

static void Main(string[] args)

{

child obj = new child();

obj.AbstractMethod();

obj.nonAbstractMethod();

Console.ReadLine();

}

}

Output:

public constructor contain in abstract class

public constructor contain in non abstract class

Abstract method

Non abstract method

public abstract class AbstractCls

{

public AbstractCls(string name)

{

Console.WriteLine(name + "abstract constructor");

}

}

public class child : AbstractCls

{

public child(string name) : base(name)

{

Console.WriteLine(name + "Non abstract constructor");

}

}

class Program

{

static void Main(string[] args)

{

child obj = new child("parametrize constructor : ");

Console.ReadLine();

}

}

Output :

parametrize constructor : - abstract constructor

parametrize constructor : - Non abstract constructor

abstract class A

{

protected A()

{ Console.WriteLine("Abstract class constructor"); }

}

//Derived class

class B : A

{

public B() { Console.WriteLine("Derived class constructor"); }

}

class Program

{

static void Main(string[] args)

{

B obj = new B();

}

}

Output :

Abstract class constructor

Derived class constructor

Example: 2

// abstract class 'GeeksForGeeks'

public abstract class GeeksForGeeks

{

// abstract method 'gfg()'

public abstract void gfg();

}

// class 'GeeksForGeeks' inherit

// in child class 'Geek1'

public class Geek1 : GeeksForGeeks

{

// abstract method 'gfg()'

// declare here with

// 'override' keyword

public override void gfg()

{

Console.WriteLine("class Geek1");

}

}

// class 'GeeksForGeeks' inherit in

// another child class 'Geek2'

public class Geek2 : GeeksForGeeks

{

// same as the previous class

public override void gfg()

{

Console.WriteLine("class Geek2");

}

}

public static void Main()

{

// 'g' is object of class

// 'GeeksForGeeks' class '

// GeeksForGeeks' cannot

// be instantiate

GeeksForGeeks g;

// instantiate class 'Geek1'

g = new Geek1();

// call 'gfg()' of class 'Geek1'

g.gfg();

// instantiate class 'Geek2'

g = new Geek2();

// call 'gfg()' of class 'Geek2'

g.gfg();

}

OUTPUT :

class Geek1

class Geek2

Example: 3

abstract class AreaClass

{

abstract public int Area();

}

class Square : AreaClass

{

int side = 0;

public Square(int n)

{

side = n;

}

public override int Area()

{

return side \* side;

}

}

class gfg

{

public static void Main()

{

Square s = new Square(6);

Console.WriteLine("Area = " + s.Area());

}

}

Example 3 :

abstract class AbstractClass

{

public int AddTwoNumbers(int Num1, int Num2)

{

return Num1 + Num2;

}

public abstract int MultiplyTwoNumbers(int Num1, int Num2);

}

class Derived : AbstractClass

{

public override int MultiplyTwoNumbers(int Num1, int Num2)

{

return Num1 \* Num2;

}

}

// Driver Class

class geek

{

// Main Method

public static void Main()

{

// Instance of the derived class

Derived d = new Derived();

Console.WriteLine("Addition : {0}\nMultiplication :{1}",

d.AddTwoNumbers(4, 6),

d.MultiplyTwoNumbers(6, 4));

}

}

***1.19 Singleton class vs. Static methods***

The following compares Singleton class vs. Static methods:

1. A Static Class cannot be extended whereas a singleton class can be extended.
2. A Static Class can still have instances (unwanted instances) whereas a singleton class prevents it.
3. A Static Class cannot be initialized with a STATE (parameter), whereas a singleton class can be.
4. A Static class is loaded automatically by the CLR when the program or namespace containing the class is loaded.

***1.19.1 Singleton Class***

public class SingletonExample

{

private static SingletonExample instance;

private SingletonExample () {}

public static SingletonExample Instance

{

get

{

if (instance == null)

{

instance = new SingletonExample ();

}

return instance;

}

}

}

Example

public sealed class SingletonExample  
{

private static SingletonExample instance=null;  
private static readonly object objectlockCheck = new object();  
private SingletonExample ()  
{  
}  
public static SingletonExample Instance()  
{

lock (objectlockCheck)  
{

if (instance==null)

instance = new SingletonExample ();

}

return instance;

}

}

***1.20 Partial Class:***

**Advantages:**

* With the help of partial class multiple developers can work simultaneously on the same class in different files.
* With the help of partial class concept you can split the UI of design code and the business logic code to read and understand the code.
* When you were working with automatically generated code, the code can be added to the class without having to recreate the source file like in Visual studio.
* You can also maintain your application in an efficient manner by compressing large classes into small ones.

class Program

{

static void Main(string[] args)

{

Geeks objGeek = new Geeks("Avadhut",1);

objGeek.Display();

}

}

public partial class Geeks

{

private string Author\_name;

private int Total\_articles;

public Geeks(string a, int t)

{

this.Author\_name = a;

this.Total\_articles = t;

}

}

public partial class Geeks

{

public void Display()

{

Console.WriteLine("Author's name is : " + Author\_name);

Console.WriteLine("Total number articles is : " + Total\_articles);

Console.ReadLine();

}

}

***1.21 Structure:***

## **Difference between structs and classes**

|  |  |
| --- | --- |
| **structs** | **classes** |
| structs are value type | classes are reference type |
| structs are stored in stack or a inline | classes are stored on managed heap |
| structs doesn't support inheritance | classes support inheritance |
| But handing of constructor is different in structs. The complier supplies a default no-parameter constructor, which your are not permitted to replace | Constructors are fully supported in classes |

public interface aa

{

// no access specifier is given in interface methods (by defualt they are public)

double Increment();

void DisplayValues();

}

public struct Student : aa

{

int id;

int zipcode;

double salary;

public Student(int id, int zipcode, double salary)

{

this.id = id;

this.zipcode = zipcode;

this.salary = salary;

}

public void DisplayValues()

{

Console.WriteLine("ID: " + this.id.ToString());

Console.WriteLine("Zipcode : " + this.zipcode.ToString());

Console.WriteLine("Salary : " + this.salary.ToString());

}

public double Increment()

{

return (this.salary += 1000.00);

}

}

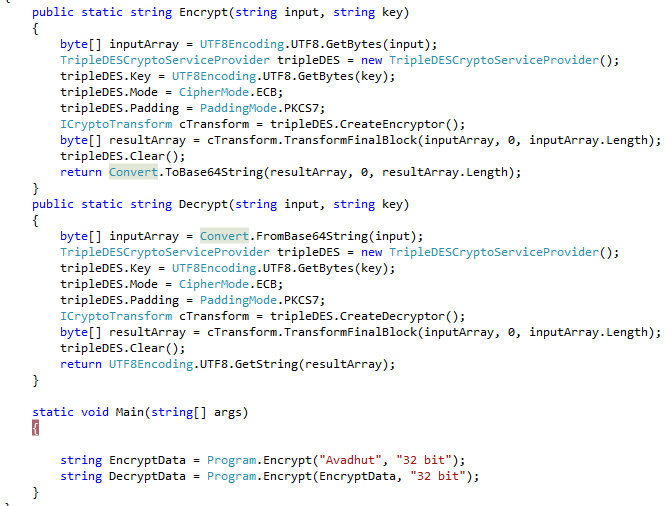
***1.22 Difference between REF and OUT Keyword***

|  |  |
| --- | --- |
| **REF KEYWORD** | **OUT KEYWORD** |
| It is necessary the parameters should initialize before it pass to ref. | It is not necessary to initialize parameters before it pass to out. |
| It is not necessary to initialize the value of a parameter before returning to the calling method. | It is necessary to initialize the value of a parameter before returning to the calling method. |
| The passing of value through ref parameter is useful when the called method also need to change the value of passed parameter. | The declaring of parameter through out parameter is useful when a method return multiple values. |
| When ref keyword is used the data may pass in bi-directional. | When out keyword is used the data only passed in unidirectional. |

***1.23 Cryptography***

How to Encrypt Data and Decrypt Data : Using UTF8Encoding we can possible to convert data to encrypt to decrypt format.

using System.Security.Cryptography;



***1.24 Indexer :***

**Important points to remember on indexers:**

* Indexers are always created with **this** keyword.
* Parameterized property are called indexer.
* Indexers are implemented through get and set accessors for the [ ] operator.
* ref and out parameter modifiers are not permitted in indexer.
* The formal parameter list of an indexer corresponds to that of a method and at least one parameter should be specified.
* Indexer is an instance member so can't be static but property can be static.
* Indexers are used on group of elements.
* Indexer is identified by its signature where as a property is identified it's name.
* Indexers are accessed using indexes where as properties are accessed by names.
* Indexer can be overloaded.

1. **class** Program
2. {
3. **class** IndexerClass
4. {
5. **private** **string**[] names = **new** **string**[10];
6. **public** **string** **this**[**int** i]
7. {
8. **get**
9. {
10. **return** names[i];
11. }
12. **set**
13. {
14. names[i] = value;
15. }
16. }
17. }
18. **static** **void** Main(**string**[] args)
19. {
20. IndexerClass Team = **new** IndexerClass();
21. Team[0] = "Rocky";
22. Team[1] = "Teena";
23. Team[2] = "Ana";
24. Team[3] = "Victoria";
25. Team[4] = "Yani";
26. Team[5] = "Mary";
27. Team[6] = "Gomes";
28. Team[7] = "Arnold";
29. Team[8] = "Mike";
30. Team[9] = "Peter";
31. **for** (**int** i = 0; i < 10; i++)
32. {
33. Console.WriteLine(Team[i]);
34. }
35. Console.ReadKey();
36. }
37. }

## **1.24.1 Difference between Indexers and Properties**

|  |  |
| --- | --- |
| Indexers | Properties |
| Indexers are created with this keyword. | Properties don't require this keyword. |
| Indexers are identified by signature. | Properties are identified by their names. |
| Indexers are accessed using indexes. | Properties are accessed by their names. |
| Indexer are instance member, so can't be static. | Properties can be static as well as instance members. |
| A get accessor of an indexer has the same formal parameter list as the indexer. | A get accessor of a property has no parameters. |
| A set accessor of an indexer has the same formal parameter list as the indexer, in addition to the value parameter. | A set accessor of a property contains the implicit value parameter. |

***1.25 Attribute:***

[obsolete] : Method need to use or not.

1. **class** MyClass
2. {
3. [Obsolete("Use method SaySomething1")]
4. **public** **void** SaySomething(){ Console.WriteLine("Execute saysomething()");}
5. **public** **void** SaySomething1(){ Console.WriteLine("Execute saysomething1()");}
6. }

***[serialization]***

***[NonSerialization]***

***[Web Method]***

***1.26 Delegate***

**What is a Delegate?**

Delegate is similar to a function pointer in C & C++ but the delegates are user defined types in C#. Make a note that delegates are not a member of class, but similar to a class. These are the backbone for events.

**When to use delegates?**

I believe that a lot of people can answer the "what is a delegate?" question in interviews but are not able to explain when to use it. No worries! Let me tell you a few important points about delegates.

1. These are used to represent or refer to one or more functions.
2. These can only be used to define call-back methods.
3. In order to consume a delegate, we need to create an object to delegate

**How many types of delegates are present?**

There are two types of delegates available.

1. Single Cast Delegate
2. Multi Cast Delegate

class Program

{

static void Display(string S)

{

Console.WriteLine("My Name is :" + S);

}

delegate void X(string a);

static void Main(string[] args)

{

X objD = new X(Display);

objD("Rathrola Prem Kumar");

Console.Read();

}

}

1. **class** Program {
2. **public** **void** Display(string S) {
3. Console.WriteLine("My Designation is :" + S);
4. }
5. **public** delegate **void** Delegate(string a);
6. **class** DelegateDemo {
7. **static** **void** Main() {
8. Program obj1 = **new** Program();
9. Delegate objD = **new** Delegate(obj1.Display);
10. objD("Technical Specialist");
11. Console.Read();
12. }
13. }
14. }

***1.26.1 MulitcasteDelegate***

1. **class** Program {
2. **public** **void** Add(**int** x, **int** y) {
3. Console.WriteLine("Sum is:" + (x + y));
4. }
5. **public** **void** Subtract(**int** x, **int** y) {
6. Console.WriteLine("Difference is:" + (x - y));
7. }
8. **public** **void** Multiply(**int** x, **int** y) {
9. Console.WriteLine("Product is:" + (x \* y));
10. }
11. **public** **void** Divide(**int** x, **int** y) {
12. Console.WriteLine("Quotient is:" + (x / y));
13. }
14. }
15. **public** delegate **void** MultiCastDelegate(**int** a, **int** b);
16. **class** ClsDelegate {
17. **static** **void** Main() {
18. Program obj1 = **new** Program();
19. MultiCastDelegate objD = **new** MultiCastDelegate(obj1.Multiply);
20. objD += obj1.Add;
21. objD += obj1.Substract;
22. objD += obj1.Divide;
23. objD(40, 10);
24. objD -= obj1.Add;
25. objD -= obj1.Divide;
26. objD(50, 10);
27. Console.ReadLine();
28. }
29. }

## ***1.26.2 Array of Delegates***

1. **namespace** Delegates
2. {
3. **public** **class** Operation
4. {
5. **public** **static** **void** Add(**int** a, **int** b)
6. {
7. Console.WriteLine("Addition={0}",a + b);
8. }
10. **public** **static** **void** Multiple(**int** a, **int** b)
11. {
12. Console.WriteLine("Multiply={0}", a \* b);
13. }
14. }
16. **class** Program
17. {
18. **delegate** **void** DelOp(**int** x, **int** y);
20. **static** **void** Main(**string**[] args)
21. {
22. // Delegate instantiation
23. DelOp[] obj =
24. {
25. **new** DelOp(Operation.Add),
26. **new** DelOp(Operation.Multiple)
27. };
29. **for** (**int** i = 0; i < obj.Length; i++)
30. {
31. obj[i](2, 5);
32. obj[i](8, 5);
33. obj[i](4, 6);
34. }
35. Console.ReadLine();
36. }
37. }
38. }

***1.26.3 Anonymous Methods***

1. **namespace** Delegates
2. {
3. **class** Program
4. {
5. // Delegate Definition
6. **delegate** **void** operation();
8. **static** **void** Main(**string**[] args)
9. {
10. // Delegate instantiation
11. operation obj = **delegate**
12. {
13. Console.WriteLine("Anonymous method");
14. };
15. obj();
17. Console.ReadLine();
18. }
19. }
20. }

## **1.26.4 Events**

1. **namespace** Delegates
2. {
3. **public** **delegate** **void** DelEventHandler();
5. **class** Program
6. {
7. **public** **static** **event** DelEventHandler add;
9. **static** **void** Main(**string**[] args)
10. {
11. add += **new** DelEventHandler(USA);
12. add += **new** DelEventHandler(India);
13. add += **new** DelEventHandler(England);
14. add.Invoke();
16. Console.ReadLine();
17. }
18. **static** **void** USA()
19. {
20. Console.WriteLine("USA");
21. }
23. **static** **void** India()
24. {
25. Console.WriteLine("India");
26. }
28. **static** **void** England()
29. {
30. Console.WriteLine("England");
31. }
32. }
33. }

***1.27 Thread***

using System.Threading.Tasks;

public static void StartupA(object parameters)

{

string Name = Convert.ToString(parameters);

Console.WriteLine(Name);

}

public static void StartupB(string Thread , string Expression)

{

Console.WriteLine(Thread + ":" + Expression);

}

public static void ThreadStart()

{

Console.WriteLine("Thread Start");

}

static void Main(string[] args)

{

Thread t = new Thread(new ParameterizedThreadStart(StartupA));

t.Name = "Paramatertize Thread";

t.Start(t.Name);

// You can also use an anonymous delegate to do this.

Thread t2 = new Thread(delegate()

{

ThreadStart();

});

t2.Start();

//// Or lambda expressions if you are using C# 3.0

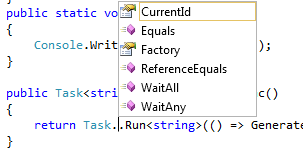
Thread t3 = new Thread(() => StartupB("Thread", "Lambda Expression"));

t3.Start();

Console.ReadLine();

}

***1.27.1 Task Property :***



async / await Tutorial in C#

public static void DoSynchronousWork()

{

// You can do whatever work is needed here

Console.WriteLine("1. Doing some work synchronously");

}

static async Task DoSomethingAsync() //A Task return type will eventually yield a void

{

await GetStringAsync(); // we are awaiting the Async Method GetStringAsync

Console.WriteLine("3. Async task ended...");

}

static async Task GetStringAsync()

{

Console.WriteLine("2. Async task has started...");

}

static void DoSynchronousWorkAfterAwait()

{

//This is the work we can do while waiting for the awaited Async Task to complete

Console.WriteLine("7. While waiting for the async task to finish, we can do some unrelated work...");

for (var i = 0; i <= 5; i++)

{

for (var j = i; j <= 5; j++)

{

Console.Write("\*");

}

Console.WriteLine();

}

}

static void Main(string[] args)

{

DoSynchronousWork();

var someTask = DoSomethingAsync();

DoSynchronousWorkAfterAwait();

someTask.Wait(); //this is a blocking call, use it only on Main method

Console.ReadLine();

}

Thread.Sleep(200);

The Sleep() method suspends the current thread for the specified milliseconds

The Abort() method is used to terminate the thread.

Thread t1 = **new** Thread(**new** ThreadStart(mt.Thread1));

t1.Start();

t1.Abort();

# C# Threading Example: Join () method

It causes all the calling threads to wait until the current thread (joined thread) is terminated or completes its task.

MyThread mt = new MyThread();

Thread t1 = new Thread(new ThreadStart(mt.Thread1));

Thread t2 = new Thread(new ThreadStart(mt.Thread1));

Thread t3 = new Thread(new ThreadStart(mt.Thread1));

t1.Start();

t1.Join();

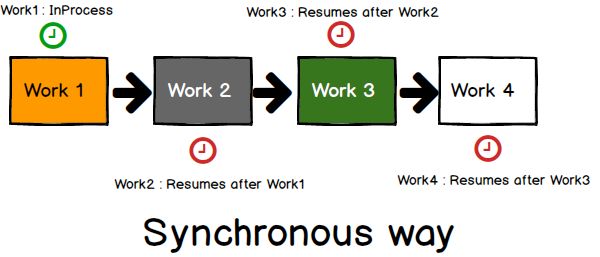
t2.Start();

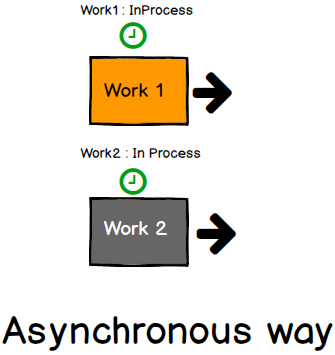
t3.Start();

static void Main(string[] args)  
{   
     ThreadStart ts = new ThreadStart(run);  
     // create new thread  
     Thread thrd = new Thread(ts);  
     // start thread  
     thrd.Start();  
     Thread.Sleep(1000);

for (int t = 10; t > 0; t--)  
     {  
         Console.WriteLine("Main Thread value is :" + t);  
         Thread.Sleep(1000);  
     }  
  
     Console.WriteLine("Good Bye!!!I'm main Thread");  
     Console.ReadLine();

}   
static void run()  
{  
      for (int i = 0; i < 10; i++)  
      {  
          Console.WriteLine("Sub Thread value is : " + i);  
          Thread.Sleep(1000);  
      }  
      Console.WriteLine("Good Bye!!!I'm Sub Thread");  
}





This keyword is used in threading.

It restricts code from being executed by more than one thread at the same time.

private static readonly Object myObj = new Object();

lock (myObj)

{

// critical section

}

## Monitor.Enter and Monitor.Exit

The lock statement is in fact a syntactic shortcut for a call to the methods Monitor.Enter and Monitor.Exit, with a try/finally block.

bool locked = false;

var tempObj = obj;

try

{

Monitor.Enter(tempObj, ref locked);

// body

}

finally

{

if (locked)

{

Monitor.Exit(tempObj);

}

}

* A lock statement, in C#, is a statement that contains the "lock" keyword and is used in multithreaded applications to ensure that the current thread executes a block of code to completion without interruption by other threads.
* The lock statement obtains a mutual exclusion lock for a given object so that one thread executes the code block at a time and exits the code block after releasing the lock.
* The lock keyword calls Enter at the start of the block and Exit at the end of the block.
* Lock keyword actually handles Monitor class at back end.

How to call store procedure in c# with input and output parameter

1. SqlCommand cmd = **new** SqlCommand("spuserdetail", con);
2. cmd.CommandType = CommandType.StoredProcedure;
3. cmd.Parameters.AddWithValue("@UserName", UserName);
4. cmd.Parameters.AddWithValue("@Password", Password);
5. cmd.Parameters.AddWithValue("@Email", Email);
6. cmd.Parameters.AddWithValue("@Country", Country);
7. cmd.Parameters.Add("@ERROR", SqlDbType.Char, 500);
8. cmd.Parameters["@ERROR"].Direction = ParameterDirection.Output;
9. cmd.ExecuteNonQuery();
10. message = (string) cmd.Parameters["@ERROR"].Value;

How to Load and Read XML in Csharp

using System.Xml.Linq;

<object>

<name>Sphere</name>

<material>Steel</material>

<device Id="01">

<model>Model 1</model>

<color>Red</color>

</device>

<device Id="02">

<model>Model 2</model>

<color>Blue</color>

</device>

</object>

XDocument doc = XDocument.Load(@"C:\Users\ABC\Documents\Visual Studio 2012\ConsoleApplication2\ConsoleApplication2\XMLFile1.xml");

foreach (var itemElement in doc.Elements("object"))

{

Console.WriteLine(itemElement.Element("name").Value);

Console.WriteLine(itemElement.Element("material").Value);

foreach (var device in itemElement.Elements("device"))

{

Console.WriteLine(device.Element("model").Value);

Console.WriteLine(device.Element("color").Value);

}

}

How to retrieve connection string from web config file

ConfigurationManger.ConnectionString[“DBMS”].connectionstring

How to read excel file in c#

var fileName = @"C:\ExcelFile.xlsx";

var connectionString = "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=" + fileName + ";Extended Properties=\"Excel 12.0;IMEX=1;HDR=NO;TypeGuessRows=0;ImportMixedTypes=Text\""; ;

using (var conn = new OleDbConnection(connectionString))

{

conn.Open();

var sheets = conn.GetOleDbSchemaTable(System.Data.OleDb.OleDbSchemaGuid.Tables, new object[] { null, null, null, "TABLE" });

using (var cmd = conn.CreateCommand())

{

cmd.CommandText = "SELECT \* FROM [" + sheets.Rows[0]["TABLE\_NAME"].ToString() + "] ";

var adapter = new OleDbDataAdapter(cmd);

var ds = new DataSet();

adapter.Fill(ds);

}

}

How to create Excel file

public static void exportToExcel(DataSet source, string fileName)

{

const string endExcelXML = "</Workbook>";

const string startExcelXML = "<xml version>\r\n<Workbook " +

"xmlns=\"urn:schemas-microsoft-com:office:spreadsheet\"\r\n" +

" xmlns:o=\"urn:schemas-microsoft-com:office:office\"\r\n " +

"xmlns:x=\"urn:schemas- microsoft-com:office:" +

"excel\"\r\n xmlns:ss=\"urn:schemas-microsoft-com:" +

"office:spreadsheet\">\r\n <Styles>\r\n " +

"<Style ss:ID=\"Default\" ss:Name=\"Normal\">\r\n " +

"<Alignment ss:Vertical=\"Bottom\"/>\r\n <Borders/>" +

"\r\n <Font/>\r\n <Interior/>\r\n <NumberFormat/>" +

"\r\n <Protection/>\r\n </Style>\r\n " +

"<Style ss:ID=\"BoldColumn\">\r\n <Font " +

"x:Family=\"Swiss\" ss:Bold=\"1\"/>\r\n </Style>\r\n " +

"<Style ss:ID=\"StringLiteral\">\r\n <NumberFormat" +

" ss:Format=\"@\"/>\r\n </Style>\r\n <Style " +

"ss:ID=\"Decimal\">\r\n <NumberFormat " +

"ss:Format=\"0.0000\"/>\r\n </Style>\r\n " +

"<Style ss:ID=\"Integer\">\r\n <NumberFormat " +

"ss:Format=\"0\"/>\r\n </Style>\r\n <Style " +

"ss:ID=\"DateLiteral\">\r\n <NumberFormat " +

"ss:Format=\"mm/dd/yyyy;@\"/>\r\n </Style>\r\n " +

"</Styles>\r\n ";

System.IO.StreamWriter excelDoc = null;

excelDoc = new System.IO.StreamWriter(fileName);

int sheetCount = 1;

excelDoc.Write(startExcelXML);

foreach (DataTable table in source.Tables)

{

int rowCount = 0;

excelDoc.Write("<Worksheet ss:Name=\"" + table.TableName + "\">");

excelDoc.Write("<Table>");

excelDoc.Write("<Row>");

for (int x = 0; x < table.Columns.Count; x++)

{

excelDoc.Write("<Cell ss:StyleID=\"BoldColumn\"><Data ss:Type=\"String\">");

excelDoc.Write(table.Columns[x].ColumnName);

excelDoc.Write("</Data></Cell>");

}

excelDoc.Write("</Row>");

foreach (DataRow x in table.Rows)

{

rowCount++;

//if the number of rows is > 64000 create a new page to continue output

if (rowCount == 64000)

{

rowCount = 0;

sheetCount++;

excelDoc.Write("</Table>");

excelDoc.Write(" </Worksheet>");

excelDoc.Write("<Worksheet ss:Name=\"" + table.TableName + "\">");

excelDoc.Write("<Table>");

}

excelDoc.Write("<Row>"); //ID=" + rowCount + "

for (int y = 0; y < table.Columns.Count; y++)

{

System.Type rowType;

rowType = x[y].GetType();

switch (rowType.ToString())

{

case "System.String":

string XMLstring = x[y].ToString();

XMLstring = XMLstring.Trim();

XMLstring = XMLstring.Replace("&", "&");

XMLstring = XMLstring.Replace(">", ">");

XMLstring = XMLstring.Replace("<", "<");

excelDoc.Write("<Cell ss:StyleID=\"StringLiteral\">" +

"<Data ss:Type=\"String\">");

excelDoc.Write(XMLstring);

excelDoc.Write("</Data></Cell>");

break;

case "System.DateTime":

//Excel has a specific Date Format of YYYY-MM-DD followed by

//the letter 'T' then hh:mm:sss.lll Example 2005-01-31T24:01:21.000

//The Following Code puts the date stored in XMLDate

//to the format above

DateTime XMLDate = (DateTime)x[y];

string XMLDatetoString = ""; //Excel Converted Date

XMLDatetoString = XMLDate.Year.ToString() +

"-" +

(XMLDate.Month < 10 ? "0" +

XMLDate.Month.ToString() : XMLDate.Month.ToString()) +

"-" +

(XMLDate.Day < 10 ? "0" +

XMLDate.Day.ToString() : XMLDate.Day.ToString()) +

"T" +

(XMLDate.Hour < 10 ? "0" +

XMLDate.Hour.ToString() : XMLDate.Hour.ToString()) +

":" +

(XMLDate.Minute < 10 ? "0" +

XMLDate.Minute.ToString() : XMLDate.Minute.ToString()) +

":" +

(XMLDate.Second < 10 ? "0" +

XMLDate.Second.ToString() : XMLDate.Second.ToString()) +

".000";

excelDoc.Write("<Cell ss:StyleID=\"DateLiteral\">" +

"<Data ss:Type=\"DateTime\">");

excelDoc.Write(XMLDatetoString);

excelDoc.Write("</Data></Cell>");

break;

case "System.Boolean":

excelDoc.Write("<Cell ss:StyleID=\"StringLiteral\">" +

"<Data ss:Type=\"String\">");

excelDoc.Write(x[y].ToString());

excelDoc.Write("</Data></Cell>");

break;

case "System.Int16":

case "System.Int32":

case "System.Int64":

case "System.Byte":

excelDoc.Write("<Cell ss:StyleID=\"Integer\">" +

"<Data ss:Type=\"Number\">");

excelDoc.Write(x[y].ToString());

excelDoc.Write("</Data></Cell>");

break;

case "System.Decimal":

case "System.Double":

excelDoc.Write("<Cell ss:StyleID=\"Decimal\">" +

"<Data ss:Type=\"Number\">");

excelDoc.Write(x[y].ToString());

excelDoc.Write("</Data></Cell>");

break;

case "System.DBNull":

excelDoc.Write("<Cell ss:StyleID=\"StringLiteral\">" +

"<Data ss:Type=\"String\">");

excelDoc.Write("");

excelDoc.Write("</Data></Cell>");

break;

default:

throw (new Exception(rowType.ToString() + " not handled."));

}

}

excelDoc.Write("</Row>");

}

excelDoc.Write("</Table>");

excelDoc.Write(" </Worksheet>");

sheetCount++;

}

excelDoc.Write(endExcelXML);

excelDoc.Close();

}

* Sealed classes are used to restrict the users from inheriting the class.
* A class can be sealed by using the **sealed** keyword.
* The keyword tells the compiler that the class is sealed, and therefore, cannot be extended.
* We can create object of sealed class

## **Finalize**

* Finalize method also called destructor to the class.
* Finalize method can not be called explicitly in the code.
* Only Garbage collector can call the the Finalize when object become inaccessible.
* Finalize method cannot be implemented directly it can only be implement via declaring destructor.

public class MyClass : IDisposable

{

//Construcotr

public MyClass()

{

//Initialization:

}

//Destrucor also called Finalize

~MyClass()

{

this.Dispose();

}

public void Dispose()

{

//write code to release unmanaged resource.

}

}

## **Dispose**

* Garbage collector (GC) plays the main and important role in .NET for memory management so programmer can focus on the application functionality.
* Garbage collector is responsible for releasing the memory (objects) that is not being used by the application.

## **Close Vs Dispose**

Some objects expose Close and Dispose two methods. For Stream classes both serve the same purpose. Dispose method calls Close method inside.

void Dispose()

{

this.Close();

}

Having Dispose method will enable you to write below code and implicitly call dispose method and ultimately will call Close method.

1. **using**(FileStream file = **new** FileStream("path", FileMode.Open, FileAccess.Read))
2. {
3. //Do something with file
4. }

Jagged Array

Example 1 :

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]); // 1

Console.WriteLine(intJaggedArray[0][2]); // 3

Console.WriteLine(intJaggedArray[1][1]); // 5

Example 2 :

int[][,] intJaggedArray = new int[3][,];

intJaggedArray[0] = new int[3, 2] { { 1, 2 }, { 3, 4 }, { 5, 6 } };

intJaggedArray[1] = new int[2, 2] { { 3, 4 }, { 5, 6 } };

intJaggedArray[2] = new int[2, 2];

Console.WriteLine(intJaggedArray[0][1,1]); // 4

Console.WriteLine(intJaggedArray[1][1,0]); // 5

Console.WriteLine(intJaggedArray[1][1,1]); // 6

## **Create and Access Tuples**

1. // Create a 3-tuple
2. var author = **new** Tuple<**string**, **string**, **int**>("Mahesh Chand", "ADO.NET Programming", 2003);
4. // Display author info
5. System.Console.WriteLine("Author {0} wrote his first book titled {1} in {2}.", author.Item1, author.Item2, author.Item3);

## **Tuples in methods**

1. ***public******void****SetTupleMethod(Tuple<****string****,****string****,****int****> tupleAuthor)*
2. *{*
3. *var author2 = tupleAuthor;*
4. *Console.WriteLine("Author:{0}, Title:{1}, Year:{2}.",*
5. *author2.Item1, author2.Item2, author2.Item3);*
6. *}*
7. ts.SetTupleMethod(**new** Tuple<**string**, **string**, **int**>(
8. "Mike Gold", "Code UML", 2005));

## **Return Tuples**

1. **public** **static** Tuple<**string**, **string**, **int**> GetTupleMethod()
2. {
3. // Create a 3-tuple and return it
4. var author = **new** Tuple<**string**, **string**, **int**>(
5. "Mahesh Chand", "Programming C#", 2002);
6. **return** author;
7. }
8. var author2 = TupleSamples.GetTupleMethod();
9. Console.WriteLine("Author:{0}, Title:{1}, Year:{2}.", author2.Item1, author2.Item2, author2.Item3);

You’re now ready to use C# 7.0 tuples. (Value Tuple )

1. // tuple return type
2. **public** (**string**, **string**, **long**) TupleReturnLiteral(**long** id)
3. {
5. **string** name = **string**.Empty;
6. **string** title = **string**.Empty;
7. **long** year = 0;
9. **if** (id == 1000)
10. {
11. name = "Mahesh Chand";
12. title = "ADO.NET Programming";
13. year = 2003;
14. }
16. // tuple literal
17. **return** (name, title, year);
19. }

<https://www.c-sharpcorner.com/article/tuples-in-c-sharp/>

IEnumerable is interface contain to below method

public interface IEnumerable

{

[DispId(-4)]

IEnumerator GetEnumerator();

}

IEnumerator IS Inherit to is interface contain to below method

public interface IEnumerator

{

object Current { get; }

bool MoveNext();

void Reset();

}

## **Important Methods of Static File Class**

| Method | Usage |
| --- | --- |
| AppendAllLines | Appends lines to a file, and then closes the file. If the specified file does not exist, this method creates a file, writes the specified lines to the file, and then closes the file. |
| AppendAllText | Opens a file, appends the specified string to the file, and then closes the file. If the file does not exist, this method creates a file, writes the specified string to the file, then closes the file. |
| AppendText | Creates a StreamWriter that appends UTF-8 encoded text to an existing file, or to a new file if the specified file does not exist. |
| Copy | Copies an existing file to a new file. Overwriting a file of the same name is not allowed. |
| Create | Creates or overwrites a file in the specified path. |
| CreateText | Creates or opens a file for writing UTF-8 encoded text. |
| Decrypt | Decrypts a file that was encrypted by the current account using the Encrypt method. |
| Delete | Deletes the specified file. |
| Encrypt | Encrypts a file so that only the account used to encrypt the file can decrypt it. |
| Exists | Determines whether the specified file exists. |
| GetAccessControl | Gets a FileSecurity object that encapsulates the access control list (ACL) entries for a specified file. |
| Move | Moves a specified file to a new location, providing the option to specify a new file name. |
| Open | Opens a FileStream on the specified path with read/write access. |
| ReadAllBytes | Opens a binary file, reads the contents of the file into a byte array, and then closes the file. |
| ReadAllLines | Opens a text file, reads all lines of the file, and then closes the file. |
| ReadAllText | Opens a text file, reads all lines of the file, and then closes the file. |
| Replace | Replaces the contents of a specified file with the contents of another file, deleting the original file, and creating a backup of the replaced file. |
| WriteAllBytes | Creates a new file, writes the specified byte array to the file, and then closes the file. If the target file already exists, it is overwritten. |
| WriteAllLines | Creates a new file, writes a collection of strings to the file, and then closes the file. |
| WriteAllText | Creates a new file, writes the specified string to the file, and then closes the file. If the target file already exists, it is overwritten. |

## **Modifier Keywords**

| Modifier keywords |
| --- |
| abstract |
| Async |
| Const |
| Event |
| extern |
| New |
| override |
| Partial |
| Readonly |
| Sealed |
| Static |
| Unsafe |
| Virtual |
| Volatile |

## 

## **Access Modifier Keywords:**

| Access Modifiers | Usage |
| --- | --- |
| public | The Public modifier allows any part of the program in the same assembly or another assembly to access the type and its members. |
| private | The Private modifier restricts other parts of the program from accessing the type and its members. Only code in the same class or struct can access it. |
| internal | The Internal modifier allows other program code in the same assembly to access the type or its members. This is default access modifiers if no modifier is specified. |
| protected | The Protected modifier allows codes in the same class or a class that derives from that class to access the type or its members. |

Example: Explicitly Typed Variable

int i = 100;// explicitly typed

var j = 100; // implicitly typed

## **Const**

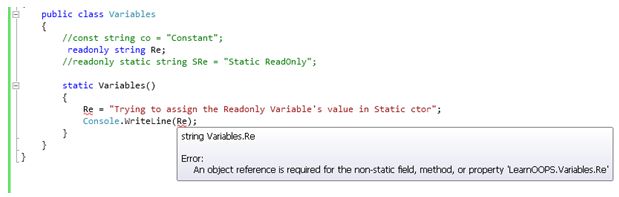
1. Const is nothing but "constant", a variable of which the value is constant but at compile time.
2. it's mandatory to assign a value to it.
3. By default a const is static and we cannot change the value of a const variable throughout the entire program.

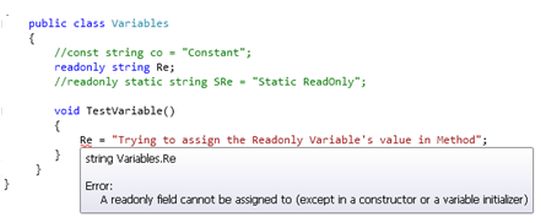


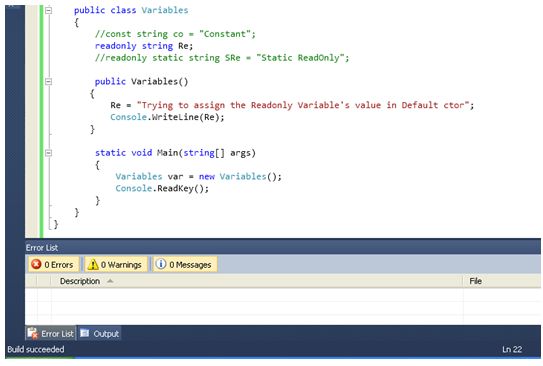


## **Readonly**

Readonly is the keyword whose value we can change during runtime or we can assign it at run time but only through the non-static constructor. Not even a method. Let's see:







**Window Scheduler and Window Service**

**Scheduled Task** -

1. When activity to be carried out on some fixed/predefined schedule.
2. It take less memory and resources of OS.
3. Not required installation.
4. It can have UI (eg. Send reminder mail to defaulters)

**Windows Service**-

1. When a continue monitoring is required.
2. It makes OS busy by consuming more.
3. Require install/uninstallation while changing version.
4. No UI at all (eg. Process a mail as soon as it arrives)

Windows Service Step :

1. Select Window Service Project
2. On Program.cs below code is available

static void Main()

{

ServiceBase[] ServicesToRun;

ServicesToRun = new ServiceBase[]

{

new Service1()

};

ServiceBase.Run(ServicesToRun);

}

1. Open ProjectFileName.cs (Desiger)

Below code

1. **protected** **override** **void** OnStart(**string**[] args)
2. {
3. LogService("Service is Started");
4. timeDelay.Enabled = **true**;
5. }
6. **protected** **override** **void** OnStop()
7. {
8. LogService("Service Stoped");
9. timeDelay.Enabled = **false**;
10. }

**Install Process**

1. Now return to the Scheduler.cs [Design] and right-click on the editor window then click "Add Installer".
2. Then you can see that there will be a new file called "ProjectInstaller.cs" as shown in the following.
3. Right-click on the "serviceInstaller1" and click "Properties".
4. Change the ServiceName to "Test Windows Service" (or your own name) and StartType to "Manual" (or you can choose "Automatic" if you need this service to be automatic).
5. Right-click the serviceProcessInstaller1, go to the properties window and change "Account" to "LocalSystem".
6. Right-click the serviceProcessInstaller1, go to the properties window and change "Account" to "LocalSystem".

## **Installing the Windows Service**

Go to "Start" >> "All Programs" >> "Microsoft Visual Studio 2012" >> "Visual Studio Tools" then click "Developer Command Prompt for VS2012".

Type the following command:

cd <physical location of your TestWindowService.exe file>

in my case it is :

cd C:\Sandbox\WindowServices\TestWindowService\TestWindowService\bin\Debug

Next type the following command:

InstallUtil.exe “TestWindowService.exe”

<https://www.c-sharpcorner.com/UploadFile/naresh.avari/develop-and-install-a-windows-service-in-C-Sharp/>

**LINQ**

**What are the types of LINQ?**

* LINQ to Objects
* LINQ to XML
* LINQ to Dataset
* LINQ to SQL
* LINQ to Entities

**Explain what is the difference between Skip() and SkipWhile() extension method?**

* **Skip()** **:** It will take an integer argument and from the given IEnumerable it skips the top n numbers
* **SkipWhile ():** It will continue to skip the elements as far as the input condition is true. It will return all remaining elements if the condition is dddddsfalse
* **First()** returns First Element Of Sequence.
* **First()** throws Exception when There IS No element Presnt In Table.
* **FirstOrDefault ()** returns First Element Of Sequence.
* **FirstOrDefault ()** does not throws Exception when There IS No element Present in Table.

**Inner join**

var query =

from post in database.Posts

join meta in database.Post\_Metas on post.ID equals meta.Post\_ID

where post.ID == id

select new { Post = post, Meta = meta };

var id = 1;

var query = database.Posts // your starting point - table in the "from" statement

.Join(database.Post\_Metas, // the source table of the inner join

post => post.ID, // Select the primary key (the first part of the "on" clause in an sql "join" statement)

meta => meta.Post\_ID, // Select the foreign key (the second part of the "on" clause)

(post, meta) => new { Post = post, Meta = meta }) // selection

.Where(postAndMeta => postAndMeta.Post.ID == id); // where statement

**Left join**

from events in this.\_context.Context.Events

join rsvps in this.\_context.Context.RSVPs

on events.EventIDequals equals rsvps.EventID into re

from c in re.DefaultIfEmpty()

where c.UserName == userName

select new {events,rsvps}

**Web Service**

**What is a Web Service?**

A web service is a web-based functionality accessed using the protocols of the web to be used by the web applications and uses standard XML Messaging for communication.

XML is used to encode all communications to a Web service. For example, a client invokes a Web service by sending an XML message, then waits for a corresponding XML response. Because all communication is in XML, Web services are not tied to any one operating system or programming.

**What is the Serialization used for WebService?**

Web Services uses XML Serialization.

**What is WSDL?**

WSDL is an acronym for Web Services Description Language. It is a format to describe what a Web Service is going to offer, generally Operations, Definition and Service bindings.

**What is the Web service protocol stack?**

The Web service protocol stack is an evolving set of protocols used to define, discover, and implement Web services. The core protocol stack consists of four layers:

**Service Transport:** This layer is responsible for transporting messages between applications. Currently, this includes HTTP, SMTP, FTP, and newer protocols, such as Blocks Extensible Exchange Protocol (BEEP).

**What is the extension for Web Service?**

.asmx

**What is the namespace for Web Service?**

System.Web.Services

How to Consume web service in web application

* 1. Add service reference in project
  2. On button click event

ServiceName.ServiceNameSoapClient client=new ServiceName.ServiceNameSoapClient()

Client.FunctionName(Parameter);

How Proxy class generate :Visual Studio generate proxy class using WSDL(web service description language)document of web service.

WSDL contain following point.

* 1. All method expose by web service
  2. The parameter and type
  3. The return type of this method

Proxy class will serialize the parameter,prepare a SOPA request message and sent it to the web service.

Web service execute method and retruns a SOAP response message to the proxy.Proxy class deSerialize the SOAP response message and hand to client application

What attribute are use in web service

[WebMethod(EnableSession=true)]

Description : use to specify a description for the web service method

[WebMethod(EnableSession=true,Description=”Add two method”)]

[WebMethod(EnableSession=true,Description=”Add two method”,CacheDuration=20)]

If function overload in web service

[WebMethod(MessageName=”Add2Number”)]

And set below as None

[WebServiceBinding(ConformsTo = WsiProfiles.None)]

## SOAP vs. REST

Let' have a quick overview of SOAP and REST before we do a deep dive into the key differences between them.

**SOAP** – SOAP is a protocol which was designed before REST and came into the picture. The main idea behind designing SOAP was to ensure that programs built on different platforms and programming languages could exchange data in an easy manner.

**REST** – This was designed specifically for working with components such as media components, files, or even objects on a particular hardware device. Any web service that is defined on the principles of REST can be called a RestFul web service. A Restful service would use the normal HTTP verbs of GET, POST, PUT and DELETE for working with the required components.

**Web API**

Web API :

Note : If package manger is not working then we have execute script on power shell

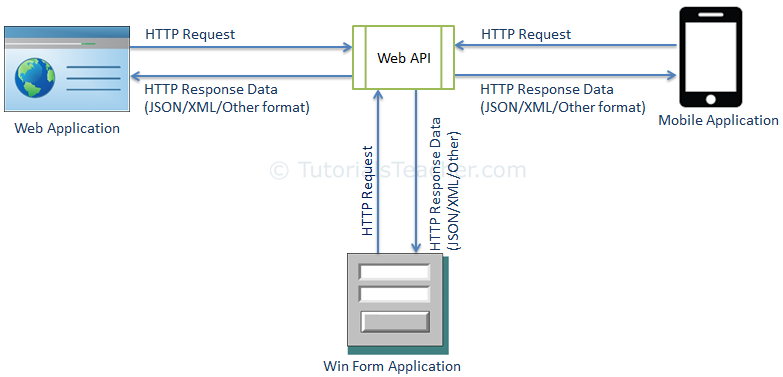
Setp 1 : open windows PowerShell in administrator

Setp 2 : execute comment SET-ExecutionPolicy AllSigned

Setp 3 : Open package manger console

## **ASP.NET Web API**

The ASP.NET Web API is an extensible framework for building HTTP based services that can be accessed in different applications on different platforms such as web, windows, mobile etc.



## **ASP.NET Web API Characteristics**

1. ASP.NET Web API is an ideal platform for building RESTful services.
2. ASP.NET Web API is built on top of ASP.NET and supports ASP.NET request/response pipeline
3. ASP.NET Web API maps HTTP verbs to method names.
4. ASP.NET Web API supports different formats of response data. Built-in support for JSON, XML, BSON format.

| **Web API Version** | **Supported .NET Framework** | **Coincides with** | **Supported in** |
| --- | --- | --- | --- |
| **Web API 1.0** | **.NET Framework 4.0** | **ASP.NET MVC 4** | **VS 2010** |
| **Web API 2 - Current** | **.NET Framework 4.5** | **ASP.NET MVC 5** | **VS 2012, 2013** |

| **Web API** | **WCF** |
| --- | --- |
| **Open source and ships with .NET framework.** | **Ships with .NET framework** |
| **Supports only HTTP protocol.** | **Supports HTTP, TCP, UDP and custom transport protocol.** |
| **Maps http verbs to methods** | **Uses attributes based programming model.** |
| **Uses routing and controller concept similar to ASP.NET MVC.** | **Uses Service, Operation and Data contracts.** |

**When to choose WCF?**

* Choose WCF if you use .NET Framework 3.5. Web API does not support .NET 3.5 or below.
* Choose WCF if your service needs to support multiple protocols such as HTTP, TCP, Named pipe.

**When to choose ASP.NET Web API?**

* Choose Web API if you are using .NET framework 4.0 or above.
* Choose Web API if you want to build a service that supports only HTTP protocol.
* Choose Web API to build RESTful HTTP based services.

**REST Constraint**

* GET
* POST
* PUT
* DELETE

Media Type formatter

config.Formatters.JsonFormatter.SerializerSettings.Formatting = Newtonsoft.Json.Formatting.Indented;

config.Formatters.JsonFormatter.SerializerSettings.ContractResolver = new CamelCasePropertyNamesContractResolver();

public JsonMediaTypeFormatter JsonFormatter { get; }

public class JsonMediaTypeFormatter : MediaTypeFormatter

{

public JsonSerializerSettings SerializerSettings { get; set; }

}

**How to return only JSON from ASP.Net WEB Api service.**

config.Formatters.Remove(config.Formatters.XMLFormatter);

**How to return only XML from ASP.Net WEB Api service.**

config.Formatters.Remove(config.Formatters.JSONFormatter);

* **Use [FormBody] attribute to force web api to get simple types from the request body.**
* **Use [FormBody] attribute to force web api to get complex types from URI.**

WEB API

Web API configuration in **WebApiConfig.cs** file and WebApiConfig registration in Application\_Start (Global.aspx.cs)

protected void Application\_Start()

{

AreaRegistration.RegisterAllAreas();

GlobalConfiguration.Configure(WebApiConfig.Register);

FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);

RouteConfig.RegisterRoutes(RouteTable.Routes);

BundleConfig.RegisterBundles(BundleTable.Bundles);

}

Below code in WebApiConfig.cs

public static void Register(HttpConfiguration config)

{

// Web API configuration and services

// Configure Web API to use only bearer token authentication.

config.SuppressDefaultHostAuthentication();

config.Filters.Add(new HostAuthenticationFilter(OAuthDefaults.AuthenticationType));

// Web API routes

config.MapHttpAttributeRoutes();

config.Routes.MapHttpRoute(

name: "DefaultApi",

routeTemplate: "api/{controller}/{id}",

defaults: new { id = RouteParameter.Optional }

);

//config.Formatters.Add(new CustomJsonFormatter());

//Start Retrun Json

config.Formatters.Remove(config.Formatters.XmlFormatter);

config.Formatters.JsonFormatter.SupportedMediaTypes.Add(new MediaTypeHeaderValue("text/html"));

//End

//Start Retrun Xml

// config.Formatters.Remove(config.Formatters.JsonFormatter);

//End

//config.Formatters.Remove(config.Formatters.JsonFormatter);

//config.Formatters.XmlFormatter.UseXmlSerializer = true;

// var jsonpFormatter = new JsonMediaTypeFormatter(config.Formatters.JsonFormatter);

// config.Formatters.Insert(0, jsonpFormatter);

EnableCorsAttribute cor = new EnableCorsAttribute("\*", "\*", "\*");

config.EnableCors(cor);

}

[FormBody] - Simple Type

[FormUri] - Complex Type

public HttpResponseMessage Put([FromBody] int id, [FromUri] Employee employee)

{

}

HTP Status Code :

## 204 No Content

* The windows Authentication provider lets you authenticates users based on their windows accounts. This provider uses IIS to perform the authentication and then passes the authenticated identity to your code. This is the default provided for ASP.net.
* The passport authentication provider uses Microsoft's passport service to authenticate users.
* The forms authentication provider uses custom HTML forms to collect authentication information and lets you use your own logic to authenticate users. The user's credentials are stored in a cookie for use during the session
* <authentication mode="windows">
* authentication mode="passport">
* <authentication mode="forms">

HttpResponse Message:

Var Message = Request.CreateResponse(HttpStatusCode.Created,object);

Message.Header.Location=new uri(Request.RequestUri + Object.entity.tostring());

Retrun Message

ASP.Net

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm2.aspx.cs" Inherits="WebApplication3.WebForm2" %>

Application Level Event :

void Application\_Start(object sender, EventArgs e)

{

// Code that runs on application startup

}

void Application\_End(object sender, EventArgs e)

{

// Code that runs on application shutdown

}

void Application\_Error(object sender, EventArgs e)

{

// Code that runs when an unhandled error occurs

}

void Session\_Start(object sender, EventArgs e)

{

// Code that runs when a new session is started

}

void Session\_End(object sender, EventArgs e)

{

// Code that runs when a session ends.

// Note: The Session\_End event is raised only when the sessionstate mode

// is set to InProc in the Web.config file. If session mode is set to StateServer

// or SQLServer, the event is not raised.

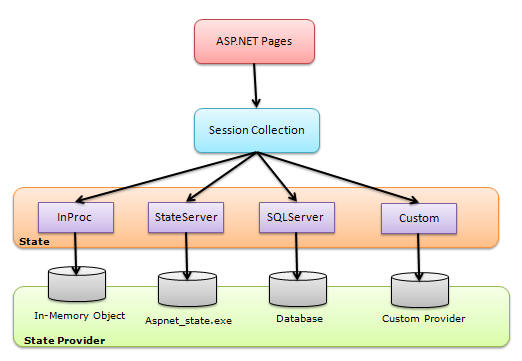
}

View State :

* View state of webform is available only with in that webform.
* ViewState is stored on the page in hidden field called \_ViewState.
* ViewState will be lost if you navigate away from the page or if the browser is closed.

Session State:

* Session state variable are available across all page, but only for a given single session.
* Session state variable store on web server.



1. <sessionstate mode="StateServer" **time** **out**="30" stateConnectionstring="tcpip=127.0.0.1"
2. "tcpip=localhost:42424"  </sessionstate>

Application State:

* Application state variable are available across all page and across all session.
* Application state variable store on web server.

State Management technique

1. Client Side :-

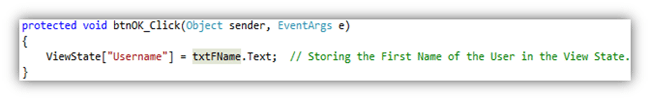
* View State
* Query string
* Cookie
* Hidden Field

1. Server Side

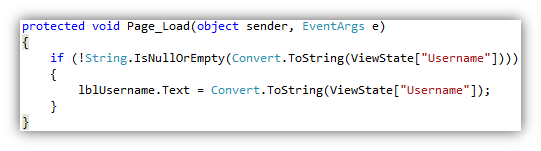
* Application State
* Session State

**View State**  
View State can be used to maintain the State at a page level. The term "Page Level" means that the information is being stored for a specific page and until that specific page is active (i.e. the page which is being currently viewed by the user). Once the user is re-directed or goes to some other page, the information stored in the View State gets lost.

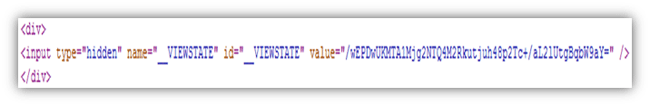
Storing Value in a View State



Retrieving Value from a View State



View State Information is stored in a Hashed Format



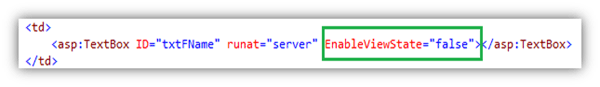
Setting View State at Application Level



Setting View State at Page Level

View-State-Settings2.gif

Setting View State at Control Level



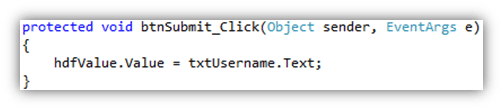
**Advantages of using a View State**

1. It is very simple to use.
2. Data is stored in hashed format and hence a layman won't be able to understand the value of the View State (It still can be hacked by Hackers, so to make it more secure we should try to store the value in an encrypted format.). Check out this link *http://msdn.microsoft.com/en-us/library/ms178199(v=vs.85).aspx* to understand, how to secure View State.
3. It is customizable, as shown above.

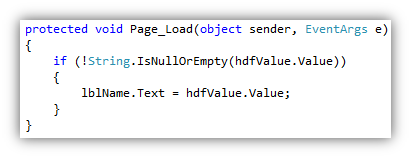
**Disadvantages of using a View State**

1. Information is not encrypted, so it can be easy for a Hacker to get its value.
2. Cannot be used to store sensitive data (eg: Passwords, Credit Card Pins, etc).
3. Might make a page heavy if lots of data is stored in View State.

**Hidden Fields**

Setting Value to a Hidden Field  


Retrieving Value from a Hidden Field



A Hidden Field stores a value at a Page Level

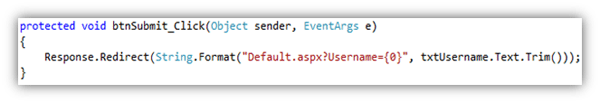
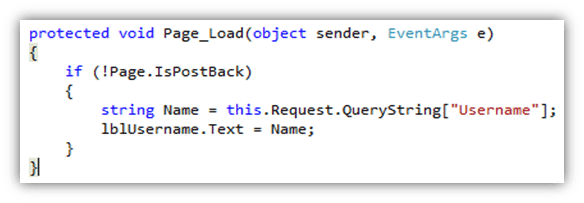
**Advantages**

1. Very simple to use.
2. Hidden Fields store the value in the page itself, hence do not use server resources.

**Disadvantages**

1. Will make a page heavy, if too many Hidden Fields are used to store data.
2. Cannot store sensitive data, as the value that is stored is neither hashed, nor encrypted.

**Query String**  
A Query String is a string which is appended to the end of the Page URL.

Way to pass a value using Query StringWay to read Query String value  
**Cookies**  
  
Cookies are one of the best ways of storing information. It is nothing but a text file which is stored on the client's machine.

How to check cookie are enable:

if (Request.Browser.Cookies)

{

//Check cookie are enable

}

1. HttpCookie userInfo = **new** HttpCookie("userInfo");
2. userInfo["UserName"] = "Annathurai";
3. userInfo["UserColor"] = "Black";
4. userInfo.Expires.Add(**new** TimeSpan(0, 1, 0));
5. Response.Cookies.Add(userInfo);
6. **string** User\_name = **string**.Empty;
7. **string** User\_color = **string**.Empty;
8. HttpCookie reqCookies = Request.Cookies["userInfo"];
9. **if** (reqCookies != **null**)
10. {
11. User\_name = reqCookies["UserName"].ToString();
12. User\_color = reqCookies["UserColor"].ToString();
13. }

## **Type of Cookies**

1. Persist Cookie - A cookie has not have expired time Which is called as Persist Cookie
2. Non-Persist Cookie - A cookie has expired time Which is called as Non-Persist Cookie

**Advantages**

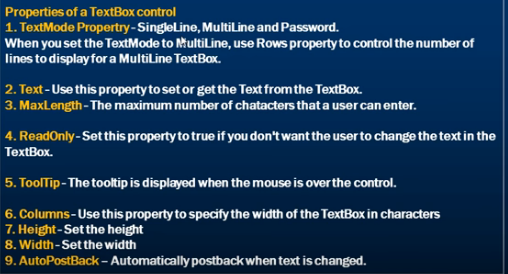
1. Very easy to use.
2. Stored on the client's machine, hence no server resources are utilized.

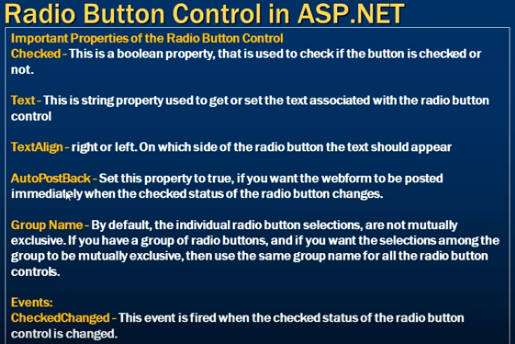
**Disadvantages**

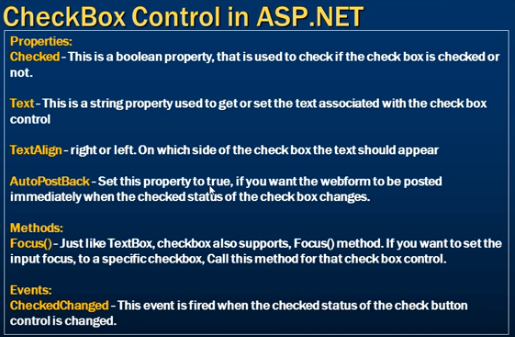
1. A user can disable cookies using browser settings.
2. Since the cookies are stored on the client's machine, there are chances that if the client's machine is hacked then the hacker can view these cookie values. Hence we should not store sensitive information in cookies.

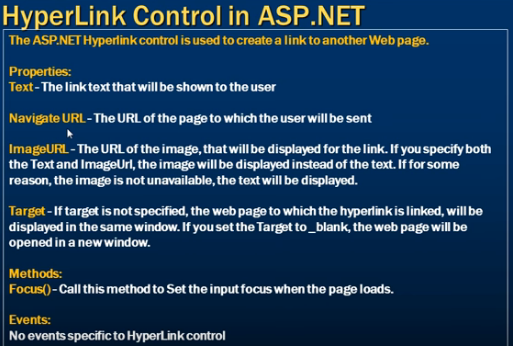
**Limitation of the Cookies**

1. The size of cookies is limited to 4096 bytes.
2. A total of 20 cookies can be used in a single website.









**HyperLink** is used to navigate, it renders to an anchor tag

**LinkButton** renders also to an anchor tag but with postback to server behavior. It's used to have a button with link look & feel.

**ImageButton** renders to an input control of type image and can be user to post to server or perform some client side action.

Command Event of an asp.net button control

When button\_Click event and button\_Comment event are together at that time.

Button\_click event call first then

Button\_Comment event call second.

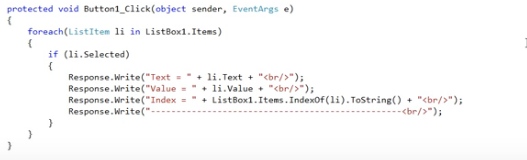
How to fill drop down base on database



How to read xml and fill dropdown in asp.net



ListBox :



AdRotator Control:

Adrotator used to display random ads.Ads information can be stored in XML or database table.



Validation in asp.net

* Range Validator
* Require field validator
* Compare validator
* Regular Expression
* Custom validator
* Validation summary.

Required Field Validator:

<asp:TextBox ID="txtName" runat="server"></asp:TextBox>

<asp:RequiredFieldValidator ID="RequiredName" runat="server" ErrorMessage="Please enter name" ForeColor="Red" ControlToValidate="txtName" Text="\*"></asp:RequiredFieldValidator>

How to check validation pass

if (Page.IsValid)

{

lblStatus.Text = "Data Save Successfully";

lblStatus.ForeColor = System.Drawing.Color.Black;

}

else

{

lblStatus.Text = "Validation Failed";

lblStatus.ForeColor = System.Drawing.Color.Red;

}

Range Validator :

<asp:RangeValidator ID="RangeAge" runat="server" ControlToValidate="txtAge"

ForeColor="Red" ErrorMessage="Please select range between 1 to 100"

MinimumValue="1" MaximumValue="100" Type="Integer">

</asp:RangeValidator>

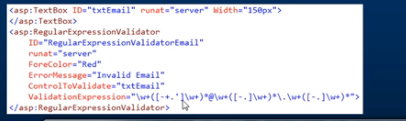
Compare Validator :

<asp:CompareValidator ID="CompareValidator1" runat="server"

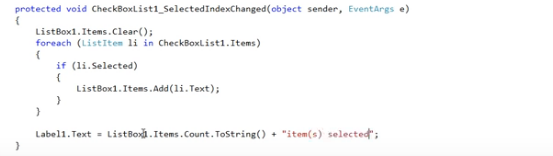
ErrorMessage="Please enter corect password" ControlToCompare="txtpwd" ControlToValidate="txtRePwd"

Operator="Equal" Type="String"></asp:CompareValidator>

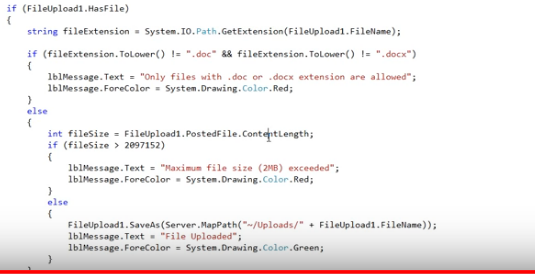
Regular Expression



How to add checkbox check item added to listbox



How to upload file



Page Navigation technique

* Response Redirect
* Server Transfer
* Server Execute
* Cross-page post back
* Window-open

Cross-Page post back

Step 1 : Create form 1 inside that create button

Step 2: <asp:Button ID="Button1" runat="server" Text="Cross Page PostBack to WebForm2"

Width="234px" PostBackUrl="~/WebForm2.aspx" onclick="Button1\_Click" />

Step 3: Create form 2

Step 4: on load

Page prePage = Page.PreviousPage;

if (prePage != null && prePage.IsCrossPagePostBack)

{

//lblTakeName.Text = ((TextBox)prePage.FindControl("txtName1")).Text;

lblTakeName.Text = ((TextBox)prePage.FindControl("txtName")).Text;

lblTakeEmail.Text = ((TextBox)prePage.FindControl("txtEmail")).Text;

lblStatus.Text = "PostBack Using Cross Page Post Back";

}

else

{

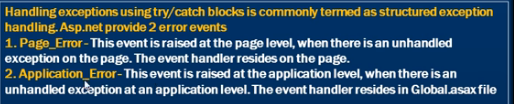
lblTakeName.Text = ((TextBox)prePage.FindControl("txtName")).Text;

lblTakeEmail.Text = ((TextBox)prePage.FindControl("txtEmail")).Text;

lblStatus.Text = "PostBack without Using Cross Page Post Back";

}

Error event :



protected void Page\_Error(object sender, EventArgs e)

{

Exception ex = Server.GetLastError();

Server.ClearError();

Response.Redirect("~/Error.aspx");

}

Custom Error Page:

<customErrors mode="On" defaultRedirect="DefaultErrorPage.aspx">

<error statusCode="404" redirect="PageNotFoundErrorPage"/>

</customErrors>

On page also we can mention error of the page

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs" ErrorPage="~/Error.aspx" Inherits="WebApplication3.WebForm1" %>

Under Window Log:

* Application: Logs information from application like Ms Office,sql server and visual studio
* Security: Log information related to user sign in and access check
* System: Log information related to driver, system.

if (EventLog.Exists("LogerName"))

{

EventLog log = new EventLog("Avadhut");

log.Source = "Avadht.com";

log.WriteEntry("Excepation", EventLogEntryType.Error);

}

How to enable Trace in asp.net

<system.web>

<trace enabled="true"/>

</system.web>

Localhost:/ApplicationName/Trace.axd

After output fetch trace is display

<trace enabled="true" pageOutput="true"/>

Authorization: Who is the user?

Authencation: Check user has right access or not.

Anonymous Authencation allow user to access public area of the web site, without prompting user for username and password.

In IIS how many type of authencation are available.

* Anonymous Authencation
* Basic Authencation
* Digest Authencation
* Form Authencation
* Window authencation: Even we try to access the application at that time every user requires to enter user name and password.
* If you select anonymous authentication, IIS doesn't perform any authentication, Any one is allowed to access the ASP.NET application.
* If you select basic authentication, users must provide a windows username and password to connect. How ever this information is sent over the network in clear text, which makes basic authentication very much insecure over the internet.
* If you select digest authentication, users must still provide a windows user name and password to connect. However the password is hashed before it is sent across the network.
* If you select windows integrated authentication, passwords never cross the network. Users must still have a username and password,

## **Passport authentication**

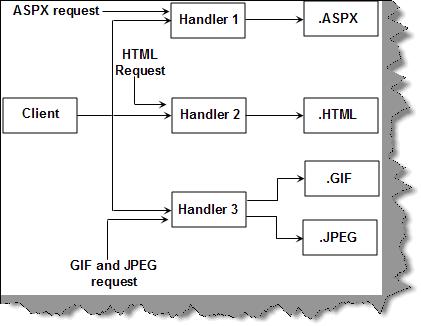
Passport authentication lets you to use Microsoft's passport service to authenticate users of your application. If your users have signed up with passport, and you configure the authentication mode of the application to the passport authentication, all authentication duties are offloaded to the passport servers.

To use passport authentication you have to download the Passport Software Development Kit (SDK) and install it on your server. The SDK can be found at <http://msdn.microdoft.com/library/default.asp?url=/downloads/list/websrvpass.aps>.

## **Forms authentication**

1. When a user requests a page for the application, ASP.NET checks for the presence of a special session cookie. If the cookie is present, ASP.NET assumes the user is authenticated and processes the request.
2. If the cookie isn't present, ASP.NET redirects the user to a web form you provide
3. You can carry out whatever authentication, checks you like in your form. When the user is authenticated, you indicate this to ASP.NET by setting a property, which creates the special cookie to handle subsequent requests.

HttpHandler



<system.web>

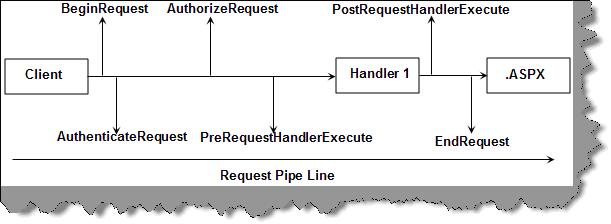
<httpHandlers>

<add verb="\*" path="\*.Shiv,\*.Koirala" type="MyPipeLine.clsMyHandler, MyPipeLine"/>

</httpHandlers>

</system.web>

HttpModule



<httpModules>

<add name="clsMyModule" type="MyPipeLine.clsMyModule, MyPipeLine"/>

</httpModules>

**PLSql**

PL/SQL is an extension of Structured Query Language (SQL) that is used in Oracle

Oracle 11g Express edition

Verify your installation - sqlplus "/ as sysdba"

# Functions in PL/SQL

Syntax to create a function:

1. **CREATE** [OR REPLACE] **FUNCTION** function\_name [parameters]
2. [(parameter\_name [IN | **OUT** | IN **OUT**] type [, ...])]
3. **RETURN** return\_datatype
4. {**IS** | **AS**}
5. **BEGIN**
6. < function\_body >
7. **END** [function\_name];

Example :

1. **create** or replace **function** adder(n1 in number, n2 in number)
2. **return** number
3. **is**
4. n3 number(8);
5. **begin**
6. n3 :=n1+n2;
7. **return** n3;
8. **end**;

Example :

1. **CREATE** OR REPLACE **FUNCTION** totalCustomers
2. **RETURN** number **IS**
3. total number(2) := 0;
4. **BEGIN**
5. **SELECT** count(\*) **into** total
6. **FROM** customers;
7. **RETURN** total;
8. **END**;

Example:

1. **DECLARE**
2. a number;
3. b number;
4. c number;
5. **FUNCTION** findMax(x IN number, y IN number)
6. **RETURN** number
7. **IS**
8. z number;
9. **BEGIN**
10. IF x > y **THEN**
11. z:= x;
12. **ELSE**
13. Z:= y;
14. **END** IF;
16. **RETURN** z;
17. **END**;
18. **BEGIN**
19. a:= 23;
20. b:= 45;
22. c := findMax(a, b);
23. dbms\_output.put\_line(' Maximum of (23,45): ' || c);
24. **END**;

## **Drop Function**

1. **DROP** **FUNCTION** function\_name;

## Calling a PL/SQL function

DECLARE

    sales\_2017 NUMBER := 0;

BEGIN

    sales\_2017 := get\_total\_sales (2017);

    DBMS\_OUTPUT.PUT\_LINE('Sales 2017: ' || sales\_2017);

END;

1. Function it must return a values whereas procedure may or may not return a values.
2. Function mainly used for to computes values. Whereas procedure mainly used for executive business logic.
3. Function to retrieve a value whereas procedure to manipulate a values.
4. Function it must return only one values but it accept many return types.

|  |  |
| --- | --- |
| **SQL** | **PL/SQL** |
| * SQL is a single query that is used to perform DML and DDL operations. | * PL/SQL is a block of codes that used to write the entire program blocks/ procedure/ function, etc. |
| * It is declarative, that defines what need to be done, rather than how things need to be done. | * PL/SQL is procedural that defines how the things needs to be done. |
| * Execute as a single statement. | * Execute as a whole block. |
| * Mainly used to manipulate data. | * Mainly used to create an application. |
| * Interaction with a Database server. | * No interaction with the database server |

## **Variable Declaration in PL/SQL**

sales number(10, 2);

pi CONSTANT double precision := 3.1415;

name varchar2(25);

DECLARE

**a integer := 10;**

**b integer := 20;**

**c integer;**

**f real;**

BEGIN

c := a + b;

dbms\_output.put\_line('Value of c: ' || c);

f := 70.0/3.0;

dbms\_output.put\_line('Value of f: ' || f);

END;

* **Local variables** − Variables declared in an inner block and not accessible to outer blocks.
* **Global variables** − Variables declared in the outermost block or a package.

DECLARE

-- Global variables

num1 number := 95;

num2 number := 85;

BEGIN

dbms\_output.put\_line('Outer Variable num1: ' || num1);

dbms\_output.put\_line('Outer Variable num2: ' || num2);

DECLARE

-- Local variables

num1 number := 195;

num2 number := 185;

BEGIN

dbms\_output.put\_line('Inner Variable num1: ' || num1);

dbms\_output.put\_line('Inner Variable num2: ' || num2);

END;

END;

IF condition THEN

S1;

ELSE

S2;

END IF;

IF(boolean\_expression 1)THEN

S1; -- Executes when the boolean expression 1 is true

ELSIF( boolean\_expression 2) THEN

S2; -- Executes when the boolean expression 2 is true

ELSIF( boolean\_expression 3) THEN

S3; -- Executes when the boolean expression 3 is true

ELSE

S4; -- executes when the none of the above condition is true

END IF;

CASE selector

WHEN 'value1' THEN S1;

WHEN 'value2' THEN S2;

WHEN 'value3' THEN S3;

...

ELSE Sn; -- default case

END CASE;

WHILE condition LOOP

sequence\_of\_statements

END LOOP;

DECLARE

  n\_times NUMBER: = 10;

BEGIN

FOR i IN 1..n\_times LOOP

    DBMS\_OUTPUT.PUT\_LINE (i);

END LOOP;

END

DECLARE

  n\_times NUMBER: = 10;

BEGIN

FOR i IN REVERSE 1..n\_times LOOP

    DBMS\_OUTPUT.PUT\_LINE (i);

END LOOP;

END

## **CHAR vs VARCHAR**

* It is a fixed length data type
* Used to store non-Unicode characters
* Occupiers 1 byte of space for each character

## **NCHAR vs NVARCHAR**

## Is a fixed length data type

* Used to store Unicode characters (for example the languages Arabic, German and so on)
* Occupies 2 bytes of space for each character

Array

DECLARE

type namesarray IS VARRAY(5) OF VARCHAR2(10);

names namesarray;

total integer;

BEGIN

names := namesarray('Kavita', 'Pritam', 'Ayan', 'Rishav', 'Aziz');

total := names.count;

dbms\_output.put\_line('Total '|| total || ' Students');

FOR i in 1 .. total LOOP

dbms\_output.put\_line('Student: ' || names(i));

END LOOP;

END;

CREATE OR REPLACE Procedure SP\_Name

(Name\_in IN varchar2)

IS

BEGIN

.

.

.

END;

**CURSOR**

**Explicit cursor**

DECLARE

cursor c is select \* from emp\_information

where emp\_name='bhavesh';

tmp emp\_information%rowtype;

BEGIN

OPEN c;

Loop exit when c%NOTFOUND;

FETCH c into tmp;

update emp\_information set tmp.emp\_dept='Web Developer'

where tmp.emp\_name='Saulin';

END Loop;

IF c%ROWCOUNT>0 THEN

dbms\_output.put\_line(SQL%ROWCOUNT||' Rows Updated');

ELSE

dbms\_output.put\_line('NO Rows Updated Found');

END IF;

CLOSE c;

END;

## Parameterize Cursor

DECLARE

cursor c(no number) is select \* from emp\_information

where emp\_no = no;

tmp emp\_information%rowtype;

BEGIN

OPEN c(4);

FOR tmp IN c(4) LOOP

dbms\_output.put\_line('EMP\_No: '||tmp.emp\_no);

dbms\_output.put\_line('EMP\_Name: '||tmp.emp\_name);

dbms\_output.put\_line('EMP\_Dept: '||tmp.emp\_dept);

dbms\_output.put\_line('EMP\_Salary:'||tmp.emp\_salary);

END Loop;

CLOSE c;

END;

Implicit cursor as the **SQL cursor**, which always has attributes such as **%FOUND, %ISOPEN, %NOTFOUND**, and **%ROWCOUNT**.

DECLARE

c\_id customers.id%type;

c\_name customerS.No.ame%type;

c\_addr customers.address%type;

CURSOR c\_customers is

SELECT id, name, address FROM customers;

BEGIN

OPEN c\_customers;

LOOP

FETCH c\_customers into c\_id, c\_name, c\_addr;

EXIT WHEN c\_customers%notfound;

dbms\_output.put\_line(c\_id || ' ' || c\_name || ' ' || c\_addr);

END LOOP;

CLOSE c\_customers;

END;

Exception

* System-defined exceptions
* User-defined exceptions

DECLARE

c\_id customers.id%type := 8;

c\_name customerS.Name%type;

c\_addr customers.address%type;

BEGIN

SELECT name, address INTO c\_name, c\_addr

FROM customers

WHERE id = c\_id;

DBMS\_OUTPUT.PUT\_LINE ('Name: '|| c\_name);

DBMS\_OUTPUT.PUT\_LINE ('Address: ' || c\_addr);

EXCEPTION

WHEN no\_data\_found THEN

dbms\_output.put\_line('No such customer!');

WHEN others THEN

dbms\_output.put\_line('Error!');

END;

DECLARE

-- user defined exception

ex\_invalid\_id EXCEPTION;

BEGIN

IF c\_id <= 0 THEN

RAISE ex\_invalid\_id;

EXCEPTION

WHEN ex\_invalid\_id THEN

dbms\_output.put\_line('ID must be greater than zero!');

WHEN no\_data\_found THEN

dbms\_output.put\_line('No such customer!');

WHEN others THEN

dbms\_output.put\_line('Error!');

END;

### PL/SQL Package Advantages

1. You can create package to store all related functions and procedures are grouped together into single unit called packages.
2. Package are reliable to granting a privileges.
3. All function and procedure within a package can share variable among them.
4. Package are support overloading to overload functions and procedures.
5. Package are improve the performance to loading the multiple object into memory at once, therefore, subsequent calls to related program doesn't required to calling physically I/O.

CREATE [OR REPLACE] PACKAGE BODY <package\_name>

IS

<Private element definition>

<sub\_program and public element definition>

.

BEGINE

<Package Initialization>

END <package\_name>

|  |  |  |
| --- | --- | --- |
| **Method** | **Description** | **SYNTAX** |
| EXISTS (n) | This method will return Boolean results. It will return 'TRUE' if the nthelement exists in that collection, else it will return FALSE. Only EXISTS functions can be used in uninitialized collection | <collection\_name>.EXISTS(element\_position) |
| COUNT | Gives the total count of the elements present in a collection | <collection\_name>.COUNT |
| LIMIT | It returns the maximum size of the collection. For Varray, it will return the fixed size that has been defined. For Nested table and Index-by-table, it gives NULL | <collection\_name>.LIMIT |
| FIRST | Returns the value of the first index variable(subscript) of the collections | <collection\_name>.FIRST |
| LAST | Returns the value of the last index variable(subscript) of the collections | <collection\_name>.LAST |
| PRIOR (n) | Returns precedes index variable in a collection of the nth element. If there is no precedes index value NULL is returned | <collection\_name>.PRIOR(n) |
| NEXT (n) | Returns succeeds index variable in a collection of the nth element. If there is no succeeds index value NULL is returned | <collection\_name>.NEXT(n) |
| EXTEND | Extends one element in a collection at the end | <collection\_name>.EXTEND |
| EXTEND (n) | Extends n elements at the end of a collection | <collection\_name>.EXTEND(n) |
| EXTEND (n,i) | Extends n copies of the ith element at the end of the collection | <collection\_name>.EXTEND(n,i) |
| TRIM | Removes one element from the end of the collection | <collection\_name>.TRIM |
| TRIM (n) | Removes n elements from the end of collection | <collection\_name>.TRIM (n) |
| DELETE | Deletes all the elements from the collection. Makes the collection empty | <collection\_name>.DELETE |
| DELETE (n) | Deletes the nth element from the collection. If the nth element is NULL, then this will do nothing | <collection\_name>.DELETE(n) |
| DELETE (m,n) | Deletes the element in the range mthto nth in the collection | <collection\_name>.DELETE(m,n) |

DECLARE

TYPE names\_table IS TABLE OF VARCHAR2(10);

TYPE grades IS TABLE OF INTEGER;

names names\_table;

marks grades;

total integer;

BEGIN

names := names\_table('Kavita', 'Pritam', 'Ayan', 'Rishav', 'Aziz');

marks:= grades(98, 97, 78, 87, 92);

total := names.count;

dbms\_output.put\_line('Total '|| total || ' Students');

FOR i IN 1 .. total LOOP

dbms\_output.put\_line('Student:'||names(i)||', Marks:' || marks(i));

end loop;

END;

SELECT SYSDATE FROM DUAL; // 08/31/2012 5:25:34 PM

SELECT TO\_CHAR(CURRENT\_DATE, 'DD-MM-YYYY HH:MI:SS') FROM DUAL;// 31-08-2012 05:26:14

SELECT ADD\_MONTHS(SYSDATE, 5) FROM DUAL;// 01/31/2013 5:26:31 PM

* 1. What are the three basic sections of a PL/SQL block?
* Declaration section
* Execution section
* Exception section
  1. What is wrong in the following assignment statement?

balance = balance + 2000;

Use of wrong assignment operator. The correct syntax is: balance := balance + 2000;

* 1. What is the purpose of %type data type? Explain with example.

It assigns a variable the same data type used by the column, for which the variable is created. For example,

dcode := dept.detpno%type

4) What is the purpose of %rowtype data type? Explain with example.

It declares a composed variable that is equivalent to the row of a table

emptype := emp%rowtype;

name := emptype.empname;

How to create job in plsql

begin

dbms\_scheduler.create\_job(job\_name => 'TRAFFIC\_DETAILS\_JOB',

job\_type => 'STORED\_PROCEDURE',

job\_action => 'traffic\_details\_temp\_send\_mail',

start\_date => systimestamp,

end\_date => null,

repeat\_interval => 'freq=daily; byhour=10; byminute=0; bysecond=0;',

enabled => true,

auto\_drop => false,

comments => 'your description here.');

end;

SQL> Create public database link MSTEST connect to MSTEST identified by mstest124 using 'dg4msql';

Database link created.

SQL>  select count(\*) from "INFORMATION\_SCHEMA"."TABLES"@MSTEST;

Html :

* <!DOCTYPE> declarations in HTML 4.01.
* In HTML5 there is only one: <!DOCTYPE html>
* <!DOCTYPE html> - It is a **DECLARATION** . It tells browsers what version of HTML are you using, which helps them to properly render the elements on webpage.

|  |  |
| --- | --- |
| **Event** | **Description** |
| onchange | An HTML element has been changed |
| onclick | The user clicks an HTML element |
| onmouseover | The user moves the mouse over an HTML element |
| onmouseout | The user moves the mouse away from an HTML element |
| onkeydown | The user pushes a keyboard key |
| onload | The browser has finished loading the page |

$(document).ready(function() {

// executes when HTML-Document is loaded and DOM is ready

});

$(window).load(function() {

// executes when complete page is fully loaded, including all frames, objects and images

});

The "#" is used to select by id:

$('#myID')

The '.' is used to select elements by class:

$('.myClass')

jQuery.min.js is a compressed version of jQuery.js

Array

var cars = ["Saab", "Volvo", "BMW"];

var name = cars[0];

document.getElementById("demo").innerHTML = cars[0];

### Object

var person = {firstName:"John", lastName:"Doe", age:46};

$.each(data.programs[0], function(key,val) {

alert(key+val);

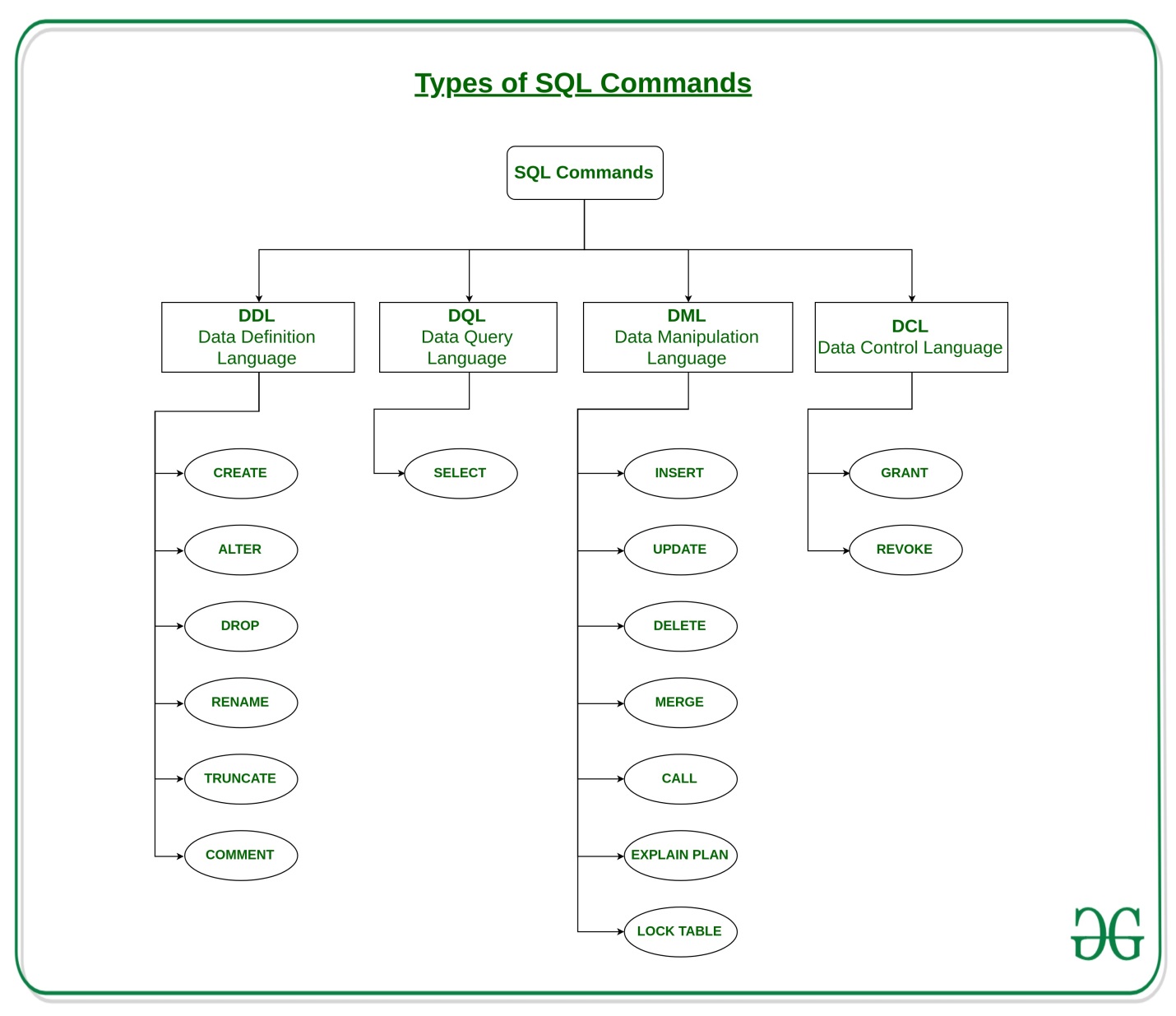
});

1. **var** arr = ['a', 'b', 'c'];
2. arr.forEach(**function**(element) {
3. console.log(element);
4. });
5. **var** arr = ['a', 'b', 'c'];
6. $.each(arr , **function** (index, value){
7. console.log(arr);
8. });

**2**=="2" // true, auto type coercion, string converted into number

**2**==="2" // false, since both operands are not of same type

SQL



## **Single row functions**

Single row functions can be character functions, numeric functions, date functions, and conversion functions. Note that these functions are used to manipulate data items. These functions require one or more input arguments and operate on each row, thereby returning one output value for each row. Argument can be a column, literal or an expression. Single row functions can be used in SELECT statement, WHERE and ORDER BY clause. Single row functions can be -

* **General functions** - Usually contains NULL handling functions. The functions under the category are NVL, NVL2, NULLIF, COALESCE, CASE, DECODE.
* **Case Conversion functions** - Accepts character input and returns a character value. Functions under the category are UPPER, LOWER and INITCAP.
  + UPPER function converts a string to upper case.
  + LOWER function converts a string to lower case.
  + INITCAP function converts only the initial alphabets of a string to upper case.
* **Character functions** - Accepts character input and returns number or character value. Functions under the category are CONCAT, LENGTH, SUBSTR, INSTR, LPAD, RPAD, TRIM and REPLACE.
  + CONCAT function concatenates two string values.
  + LENGTH function returns the length of the input string.
  + SUBSTR function returns a portion of a string from a given start point to an end point.
  + INSTR function returns numeric position of a character or a string in a given string.
  + LPAD and RPAD functions pad the given string upto a specific length with a given character.
  + TRIM function trims the string input from the start or end.
  + REPLACE function replaces characters from the input string with a given character.
* **Date functions** - Date arithmetic operations return date or numeric values. Functions under the category are MONTHS\_BETWEEN, ADD\_MONTHS, NEXT\_DAY, LAST\_DAY, ROUND and TRUNC.
  + MONTHS\_BETWEEN function returns the count of months between the two dates.
  + ADD\_MONTHS function add 'n' number of months to an input date.
  + NEXT\_DAY function returns the next day of the date specified.
  + LAST\_DAY function returns last day of the month of the input date.
  + ROUND and TRUNC functions are used to round and truncates the date value.
* **Number functions** - Accepts numeric input and returns numeric values. Functions under the category are ROUND, TRUNC, and MOD.
  + ROUND and TRUNC functions are used to round and truncate the number value.
  + MOD is used to return the remainder of the division operation between two numbers.

**Character Functions**

1. Case-Manipulative Functions (LOWER, UPPER and INITCAP)

2. Character-Manipulative Functions (CONCAT, LENGTH, SUBSTR, INSTR, LPAD, RPAD, TRIM and REPLACE)

**Input1:** SELECT LOWER('GEEKSFORGEEKS') FROM DUAL;

**Output1:** geeksforgeeks

**Input1:** SELECT UPPER('geeksforgeeks') FROM DUAL;

**Output1:** GEEKSFORGEEKS

**Input1:** SELECT INITCAP('geeksforgeeks is a computer science portal for geeks') FROM DUAL;

**Output1:** Geeksforgeeks Is A Computer Science Portal For Geeks

**Input1:** SELECT CONCAT('computer' ,'science') FROM DUAL;

**Output1:** computerscience

**Input1:** SELECT LENGTH('Learning Is Fun') FROM DUAL;

**Output1:** 15

**Input1:** SELECT SUBSTR('Database Management System', 9) FROM DUAL;

**Output1:** Management System

**Input:** SELECT INSTR('Google apps are great applications','app',1,2) FROM DUAL;

**Output:** 23

**LPAD Input1:** SELECT LPAD('100',5,'\*') FROM DUAL;

**LPAD Output1:** \*\*100

**RPAD Input1:** SELECT RPAD('5000',7,'\*') FROM DUAL;

**RPAD Output1:** 5000\*\*\*

**Input1:** SELECT TRIM('G' FROM 'GEEKS') FROM DUAL;

**Output1:** EEKS

**Input1:** SELECT REPLACE('DATA MANAGEMENT', 'DATA','DATABASE') FROM DUAL;

**Output1:** DATABASE MANAGEMENT

**NOW():** Returns the current date and time.

Example:

SELECT NOW();

Output:

2017-01-13 08:03:52

**CURDATE()**: Returns the current date.

Example:

SELECT CURDATE();

Output:

2017-01-13

**CURTIME():**Returns the current time.

Example:

SELECT CURTIME();

Output:

08:05:15

**DATE()**: Extracts the date part of a date or date/time expression.

Example:  
For the below table named ‘Test’

| **Id** | **Name** | **BirthTime** |
| --- | --- | --- |
| 4120 | Pratik | 1996-09-26 16:44:15.581 |

SELECT Name, DATE(BirthTime) AS BirthDate FROM Test;

Output:

| **Name** | **BirthDate** |
| --- | --- |
| Pratik | 1996-09-26 |

**DATE\_ADD() :** Adds a specified time interval to a date

Example:  
For the below table named ‘Test’

|  |  |  |
| --- | --- | --- |
| Id | Name | BirthTime |
| 4120 | Pratik | 1996-09-26 16:44:15.581 |

SELECT Name, DATE\_ADD(BirthTime, INTERVAL 1 YEAR) AS BirthTimeModified FROM Test;

Output:

|  |  |
| --- | --- |
| **Name** | **BirthTimeModified** |
| Pratik | 1997-09-26 16:44:15.581 |

**DATEDIFF():**Returns the number of days between two dates.Syntax:

SELECT DATE\_DIFF('2017-01-13','2017-01-03') AS DateDiff;

Output:

| **DateDiff** |
| --- |
| 10 |

1. **DATE\_ADD():** It returns a date after a certain time/date interval has been added.

**Syntax:** SELECT DATE\_ADD("2018-07-16", INTERVAL 10 DAY);

**Output:**2018-07-16

1. **DATE\_FORMAT():** It formats a date as specified by a format mask.

**Syntax:** SELECT DATE\_FORMAT("2018-06-15", "%Y");

**Output:**2018

1. **DATE\_SUB():** It returns a date after a certain time/date interval has been subtracted.

**Syntax:** SELECT DATE\_SUB("2017-06-15", INTERVAL 10 DAY);

**Output:**2018-07-16

1. **DAY():** It returns the day portion of a date value.

**Syntax:** SELECT DAY("2018-07-16");

**Output:**16

1. **DAYNAME():** It returns the weekday name for a date.

**Syntax:** SELECT DAYNAME('2008-05-15');

**Output:**Thursday

1. **DAYOFMONTH():** It returns the day portion of a date value.

**Syntax:** SELECT DAYOFMONTH('2018-07-16');

**Output:**16

**NVL(expr1, expr2)** : In SQL, NVL() converts a null value to an actual value. Data types that can be used are date, character and number. Data type must match with each other i.e. expr1 and expr2 must of same data type.  
**Syntax –**

NVL (expr1, expr2)

**expr1** is the source value or expression that may contain a null.  
**expr2** is the target value for converting the null.

**NVL2(expr1, expr2, expr3)** : The NVL2 function examines the first expression. If the first expression is not null, then the NVL2 function returns the second expression. If the first expression is null, then the third expression is returned i.e. If expr1 is not null, NVL2 returns expr2. If expr1 is null, NVL2 returns expr3. The argument expr1 can have any data type.

**Syntax –**

NVL2 (expr1, expr2, expr3)

**expr1** is the source value or expression that may contain null  
**expr2** is the value returned if expr1 is not null  
**expr3** is the value returned if expr2 is null

**NVLIF (expr1, expr2):** If not equal to expr1 and expr2, then expr1 is return

**DECODE()** : Facilitates conditional inquiries by doing the work of a CASE or IF-THEN-ELSE statement.  
The DECODE function decodes an expression in a way similar to the IF-THEN-ELSE logic used in various languages. The DECODE function decodes expression after comparing it to each search value. If the expression is the same as search, result is returned.  
If the default value is omitted, a null value is returned where a search value does not match any of the result values.

**Syntax –**

DECODE(col|expression, search1, result1

[, search2, result2,...,][, default])

Example –

SELECT last\_name, job\_id, salary,

DECODE(job\_id, ’IT\_PROG’, 1.10\*salary,

’ST\_CLERK’, 1.15\*salary,

’SA\_REP’, 1.20\*salary,salary)

REVISED\_SALARY FROM employees;

**COALESCE()** :

Expr\_1 is retrun if it is not null

Expr\_2 is retrun when expr\_1 is null and expr\_1 is not null

Syntax :-

COALESCE (expr\_1, expr\_2, ... expr\_n)

**CASE:**

CASE case\_value

WHEN when\_value THEN statement\_list

[WHEN when\_value THEN statement\_list] ...

[ELSE statement\_list]

END CASE

Example:

CASE department\_name

WHEN 'CS'

THEN UPDATE Faculty SET

department='Computer Science';

WHEN 'EC'

THEN UPDATE Faculty SET

department='Electronics and Communication';

ELSE UPDATE Faculty SET

department='Humanities and Social Sciences';

END CASE

**Various Aggregate Functions**

1) Count()

2) Sum()

3) Avg()

4) Min()

5) Max()

SELECT categoryid, AVG(unitsinstock)

FROM products

GROUP BY categoryid;

**Syntax**

SELECT [Column\_1,…] Aggregate\_Function(Column) from Table\_Name

[WHERE CONDITION]

[GROUP BY Column]

[ORDER BY Column]

[HAVING Aggregate\_FUNCTION]

## **Cross JOIN or Cartesian Product**

This type of JOIN returns the cartesian product of rows from the tables in Join. It will return a table which consists of records which combines each row from the first table with each row of the second table.

Cross JOIN Syntax is,

SELECT column-name-list

FROM

table-name1 CROSS JOIN table-name2;

#### Example of Cross JOIN

Following is the **class** table,

|  |  |
| --- | --- |
| **ID** | **NAME** |
| 1 | abhi |
| 2 | adam |
| 4 | alex |

and the **class\_info** table,

|  |  |
| --- | --- |
| **ID** | **Address** |
| 1 | DELHI |
| 2 | MUMBAI |

Cross JOIN query will be,

SELECT \* FROM

class CROSS JOIN class\_info;

The resultset table will look like,

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **NAME** | **ID** | **Address** |
| 1 | abhi | 1 | DELHI |
| 2 | adam | 1 | DELHI |
| 4 | alex | 1 | DELHI |
| 1 | abhi | 2 | MUMBAI |
| 2 | adam | 2 | MUMBAI |
| 4 | alex | 2 | MUMBAI |

## **INNER Join or EQUI Join**

This is a simple JOIN in which the result is based on matched data as per the equality condition specified in the SQL query.

Inner Join Syntax is,

SELECT column-name-list FROM

table-name1 INNER JOIN table-name2

WHERE table-name1.column-name = table-name2.column-name;

#### Example of INNER JOIN

Consider a **class** table,

|  |  |
| --- | --- |
| **ID** | **NAME** |
| 1 | abhi |
| 2 | adam |
| 3 | alex |
| 4 | anu |

and the **class\_info** table,

|  |  |
| --- | --- |
| **ID** | **Address** |
| 1 | DELHI |
| 2 | MUMBAI |
| 3 | CHENNAI |

**Inner** JOIN query will be,

SELECT \* from class INNER JOIN class\_info where class.id = class\_info.id;

The resultset table will look like,

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **NAME** | **ID** | **Address** |
| 1 | abhi | 1 | DELHI |
| 2 | adam | 2 | MUMBAI |
| 3 | alex | 3 | CHENNAI |

### Natural JOIN

Natural Join is a type of Inner join which is based on column having same name and same datatype present in both the tables to be joined.

The syntax for Natural Join is,

SELECT \* FROM

table-name1 NATURAL JOIN table-name2;

#### Example of Natural JOIN

Here is the **class** table,

|  |  |
| --- | --- |
| **ID** | **NAME** |
| 1 | abhi |
| 2 | adam |
| 3 | alex |
| 4 | anu |

and the **class\_info** table,

|  |  |
| --- | --- |
| **ID** | **Address** |
| 1 | DELHI |
| 2 | MUMBAI |
| 3 | CHENNAI |

**Natural join query will be,**

SELECT \* from class NATURAL JOIN class\_info;

The resultset table will look like,

|  |  |  |
| --- | --- | --- |
| **ID** | **NAME** | **Address** |
| 1 | abhi | DELHI |
| 2 | adam | MUMBAI |
| 3 | alex | CHENNAI |

### LEFT Outer Join

The left outer join returns a resultset table with the **matched data** from the two tables and then the remaining rows of the **left** table and null from the **right** table's columns.

Syntax for Left Outer Join is,

SELECT column-name-list FROM

table-name1 LEFT OUTER JOIN table-name2

ON table-name1.column-name = table-name2.column-name;

To specify a condition, we use the ON keyword with Outer Join.

Left outer Join Syntax for **Oracle** is,

SELECT column-name-list FROM

table-name1, table-name2 on table-name1.column-name = table-name2.column-name(+);

#### Example of Left Outer Join

Here is the **class** table,

|  |  |
| --- | --- |
| **ID** | **NAME** |
| 1 | abhi |
| 2 | adam |
| 3 | alex |
| 4 | anu |
| 5 | ashish |

and the **class\_info** table,

|  |  |
| --- | --- |
| **ID** | **Address** |
| 1 | DELHI |
| 2 | MUMBAI |
| 3 | CHENNAI |
| 7 | NOIDA |
| 8 | PANIPAT |

**Left Outer Join** query will be,

SELECT \* FROM class LEFT OUTER JOIN class\_info ON (class.id = class\_info.id);

The resultset table will look like,

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **NAME** | **ID** | **Address** |
| 1 | abhi | 1 | DELHI |
| 2 | adam | 2 | MUMBAI |
| 3 | alex | 3 | CHENNAI |
| 4 | anu | null | null |
| 5 | ashish | null | null |

### RIGHT Outer Join

The right outer join returns a resultset table with the **matched data** from the two tables being joined, then the remaining rows of the **right** table and null for the remaining **left** table's columns.

Syntax for Right Outer Join is,

SELECT column-name-list FROM

table-name1 RIGHT OUTER JOIN table-name2

ON table-name1.column-name = table-name2.column-name;

Right outer Join Syntax for **Oracle** is,

SELECT column-name-list FROM

table-name1, table-name2

ON table-name1.column-name(+) = table-name2.column-name;

#### Example of Right Outer Join

Once again the **class** table,

|  |  |
| --- | --- |
| **ID** | **NAME** |
| 1 | abhi |
| 2 | adam |
| 3 | alex |
| 4 | anu |
| 5 | ashish |

and the **class\_info** table,

|  |  |
| --- | --- |
| **ID** | **Address** |
| 1 | DELHI |
| 2 | MUMBAI |
| 3 | CHENNAI |
| 7 | NOIDA |
| 8 | PANIPAT |

**Right Outer Join** query will be,

SELECT \* FROM class RIGHT OUTER JOIN class\_info ON (class.id = class\_info.id);

The resultant table will look like,

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **NAME** | **ID** | **Address** |
| 1 | abhi | 1 | DELHI |
| 2 | adam | 2 | MUMBAI |
| 3 | alex | 3 | CHENNAI |
| null | null | 7 | NOIDA |
| null | null | 8 | PANIPAT |

### Full Outer Join

The full outer join returns a resultset table with the **matched data** of two table then remaining rows of both **left** table and then the **right** table.

Syntax of Full Outer Join is,

SELECT column-name-list FROM

table-name1 FULL OUTER JOIN table-name2

ON table-name1.column-name = table-name2.column-name;

#### Example of Full outer join is,

The **class** table,

|  |  |
| --- | --- |
| **ID** | **NAME** |
| 1 | abhi |
| 2 | adam |
| 3 | alex |
| 4 | anu |
| 5 | ashish |

and the **class\_info** table,

|  |  |
| --- | --- |
| **ID** | **Address** |
| 1 | DELHI |
| 2 | MUMBAI |
| 3 | CHENNAI |
| 7 | NOIDA |
| 8 | PANIPAT |

**Full Outer Join** query will be like,

SELECT \* FROM class FULL OUTER JOIN class\_info ON (class.id = class\_info.id);

The resultset table will look like,

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **NAME** | **ID** | **Address** |
| 1 | abhi | 1 | DELHI |
| 2 | adam | 2 | MUMBAI |
| 3 | alex | 3 | CHENNAI |
| 4 | anu | null | null |
| 5 | ashish | null | null |
| null | null | 7 | NOIDA |
| null | null | 8 | PANIPAT |

Set Operator :

#### Example of UNION

The **First** table,

|  |  |
| --- | --- |
| **ID** | **Name** |
| 1 | abhi |
| 2 | adam |

The **Second** table,

|  |  |
| --- | --- |
| **ID** | **Name** |
| 2 | adam |
| 3 | Chester |

Union SQL query will be,

SELECT \* FROM First

UNION

SELECT \* FROM Second;

The resultset table will look like,

|  |  |
| --- | --- |
| **ID** | **NAME** |
| 1 | abhi |
| 2 | adam |
| 3 | Chester |

#### Union All

The **First** table,

|  |  |
| --- | --- |
| **ID** | **NAME** |
| 1 | abhi |
| 2 | adam |

The **Second** table,

|  |  |
| --- | --- |
| **ID** | **NAME** |
| 2 | adam |
| 3 | Chester |

Union All query will be like,

SELECT \* FROM First

UNION ALL

SELECT \* FROM Second;

The resultset table will look like,

|  |  |
| --- | --- |
| **ID** | **NAME** |
| 1 | abhi |
| 2 | adam |
| 2 | adam |
| 3 | Chester |

#### Example of Intersect

The **First** table,

|  |  |
| --- | --- |
| **ID** | **NAME** |
| 1 | abhi |
| 2 | adam |

The **Second** table,

|  |  |
| --- | --- |
| **ID** | **NAME** |
| 2 | adam |
| 3 | Chester |

Intersect query will be,

SELECT \* FROM First

INTERSECT

SELECT \* FROM Second;

The resultset table will look like

|  |  |
| --- | --- |
| **ID** | **NAME** |
| 2 | adam |

#### Example of Minus

The **First** table,

|  |  |
| --- | --- |
| **ID** | **NAME** |
| 1 | abhi |
| 2 | adam |

The **Second** table,

|  |  |
| --- | --- |
| **ID** | **NAME** |
| 2 | adam |
| 3 | Chester |

Minus query will be,

SELECT \* FROM First

MINUS

SELECT \* FROM Second;

The resultset table will look like,

|  |  |
| --- | --- |
| **ID** | **NAME** |
| 1 | abhi |

How to Modify column

**ALTER TABLE table\_name**

**ALTER COLUMN column\_name column\_type;**

How to drop column

**ALTER TABLE table\_name**

**DROP COLUMN column\_name;**

How to add column

**ALTER TABLE Student ADD (AGE number(3),COURSE varchar(40));**

How to add Constraint

ALTER TABLE Persons

ADD CONSTRAINT PK\_Person PRIMARY KEY (ID, LastName);

How to drop constraint

ALTER TABLE Persons  
DROP CONSTRAINT PK\_Person;

How to add foreign key Constraint

ALTER TABLE Persons  
ADD CONSTRAINT PK\_Person

FOREGIN KEY (COLUMN\_NAME)

REFERENCE TABLE\_NAME (COLUMN\_NAME)

The available constraints in SQL are:

* **NOT NULL**: This constraint tells that we cannot store a null value in a column. That is, if a column is specified as NOT NULL then we will not be able to store null in this particular column any more.
* **UNIQUE**: This constraint when specified with a column, tells that all the values in the column must be unique. That is, the values in any row of a column must not be repeated.
* **PRIMARY KEY**: A primary key is a field which can uniquely identify each row in a table. And this constraint is used to specify a field in a table as primary key.
* **FOREIGN KEY**: A Foreign key is a field which can uniquely identify each row in a another table. And this constraint is used to specify a field as Foreign key.
* **CHECK**: This constraint helps to validate the values of a column to meet a particular condition. That is, it helps to ensure that the value stored in a column meets a specific condition.
* **DEFAULT**: This constraint specifies a default value for the column when no value is specified by the user.

**Candidate key:** A column or a set of columns can be called as candidate key if they identify each row of a table uniquely. A table can have multiple candidate keys. One of them is specified as Primary key and rest of them can be called as alternate key.  
  
**Alternate key:** There can be more than one keys which can identify each row of the table uniquely. One of them is defined as primary key and rest of them is called alternate keys of the table.

**Composite Key:**A key formed by combining at least two or more columns.

|  |  |
| --- | --- |
| **Temp Table** | **Table Variable** |
| Temporary table (#temp) is created in the tempdb database. | Table variable (@table) is created in the memory |
| Temporary tables are allowed CREATE Indexes whereas | Table variables aren’t allowed CREATE INDEX instead they can have index by using Primary Key or Unique Constraint |
| Cannot be done with Temporary tables. | Table variable can be passed as a parameter to functions and stored procedures |
| 1. CREATE TABLE# TEMP\_SAVE 2. ( 3. Emp\_IdINT, 4. Emp\_NameVARCHAR(30), 5. Project\_NameVARCHAR(30) 6. ) | 1. **Declare** @My\_var2TABLE 2. ( 3. IIDint, 4. NameNvarchar(50), 5. SalaryInt, 6. City\_NameNvarchar(50) 7. ) |

**When to use Temp Variables:**

* When your data set is smaller, approximately less than 1000 records, then use temp variables , if your result set is larger then use temp table instead of temp variables.

**Points to Remember**

* Temporary Tables are physically created in the tempdb database. These tables act as the normal table and also can have constraints, index like normal tables.
* Table Variable acts like a variable and exists for a particular batch of query execution. It gets dropped once it comes out of batch. It is created in the memory database but may be pushed out to tempdb.

Use Table variable, if you have less than 1000 rows otherwise go for Temporary tables.

**Table variables** (DECLARE @t TABLE) are visible only to the connection that creates it, and are deleted when the batch or stored procedure ends.

**Local temporary tables** (CREATE TABLE #t) are visible only to the connection that creates it, and are deleted when the connection is closed.

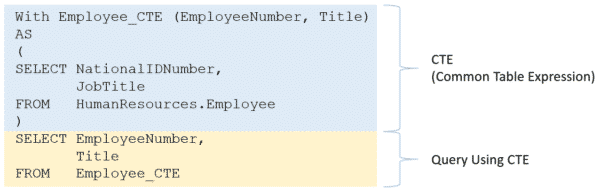
**Global temporary tables** (CREATE TABLE ##t) are visible to everyone, and are deleted when all connections that have referenced them have closed.

**CTE:**

* **Cannot be indexed**
* **Cannot have constraints**
* **Persist only until the next query is run**

**#Temp Table:**

* **Can be indexed**
* **Can have constraints**
* **Persist for the life of the current CONNECTION**
* **Can be referenced by other queries or sub procedures**



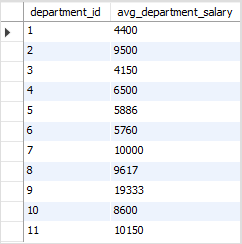
Partition By and Group By clause

SELECT department\_id, ROUND (AVG (salary)) avg\_department\_salary

FROM employees

GROUP BY department\_id

ORDER BYdepartment\_id;



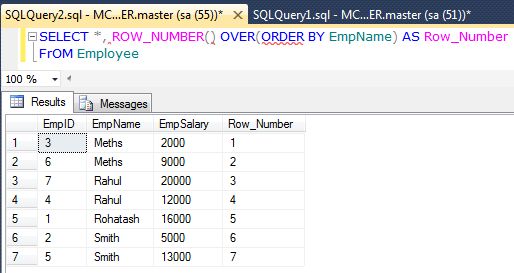
SELECT first\_name, last\_name, department\_id,

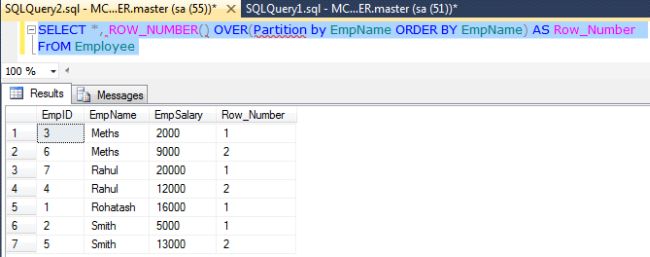
ROUND (AVG (salary)) OVER (PARTITION BY department\_id) avg\_department\_salary

FROM employees;



**Row Number:**

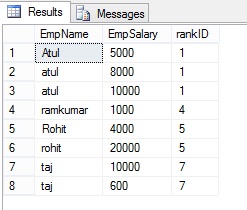




**RANK:** This function will assign a unique value **to each distinct** Row, but it leaves a group between the groups**.**

**SELECT** EmpName, EmpSalary, rank () over (**order** **by** EmpName) **as** rankID

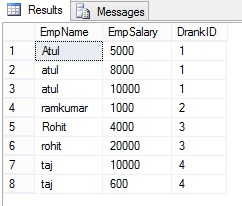
**From** Employee



**DENSE RANK:** Dense Rank () Function is similar to Rank with only difference; this will not leave gaps between groups.

**SELECT** EmpName, EmpSalary, DENSE\_RANK () over (**order** **by** EmpName) **as** DrankID

**From** Employe



# How to read data from XML String and insert in to table in SQL Server

Create PROCEDURE [dbo].[SP\_Insert\_MultipleRows] (  
 @xmlData XML ,  
 @retValue varchar(20) OUTPUT  
)

AS  
BEGIN  
SET @retValue='Failed';

INSERT INTO  [Employee](  
[id],  
[firstName],  
[lastName],  
[company]  
)

SELECT  
COALESCE([Table].[Column].value('ID[1]', 'int'),0) as 'ID',  
[Table].[Column].value('FirstName [1]', 'varchar(20)') as ' FirstName ',  
[Table].[Column].value(' LastName[1]', 'varchar(20)') as ' LastName',  
[Table].[Column].value(' Company [1]', 'varchar(50)') as ' Company'

 FROM @xmlData.nodes('/ Customers / customer') as [Table]([Column])  
IF(@@ROWCOUNT > 0 )  
  SET @retValue='SUCCESS';  
  
  
END

Execute

Declare @retValue1 varchar(50);  
Declare @XmlStr XML;  
SET @XmlStr='<Customers>  
 <customer>  
    <ID>111589</ID>  
    <FirstName>name1</FirstName>  
    <LastName>Lname1</LastName>  
    <Company>ABC</Company>  
  </customer>  
  <customer>  
    <ID>12345</ID>  
    <FirstName>name2</FirstName>  
    <LastName>Lname2</LastName>  
    <Company>ABC</Company>  
  </customer>  
  <customer>  
    <ID>14567</ID>  
    <FirstName>name3</FirstName>  
    <LastName>Lname3</LastName>  
    <Company>DEF</Company>  
  </customer>  
</Customers>';

EXEC [SP\_Insert\_MultipleRows] @xmlData=@XmlStr,@retValue=@retValue1 OUTPUT  
print @retValue1



**Stored Procedure for reading data from XML String.  
  
XML String**  
  
<DataSet>  
  <tblEmp>  
    <name>Vishal</name>  
    <designation>Developer</designation>l  
  </tblEmp>  
  <tblEmp>  
    <name>Jibin</name>  
    <designation>System Analyst</designation>l  
  </tblEmp>  
</DataSet>  
  
**Stored Procedure to read XML string**

CREATE PROC [dbo].[USP\_READXMLString]

(

   @XMLDOC2    XML

)

AS

   BEGIN

         SET NOCOUNT ON

         DECLARE @HANDLE INT

         EXEC SP\_XML\_PREPAREDOCUMENT  @HANDLE OUTPUT,@XMLDOC2

         SELECT \* FROM OPENXML(@HANDLE, '/DataSet/tblEmp', 2)

         WITH (name VARCHAR(50),designation VARCHAR(50))

END

## **Types of DML Triggers**

Insert Trigger

1. CREATE TRIGGER trgAfterInsert on Employee\_Demo
2. FOR INSERT
3. AS
4. declare @empid int,
   1. @empname varchar(55),
   2. @empsal decimal(10,2),
   3. @audit\_action varchar(100);
5. select @empid=i.Emp\_ID from inserted i;
6. select @empname=i.Emp\_Name from inserted i;
7. select @empsal=i.Emp\_Sal from inserted i;
8. set @audit\_action='Inserted Record -- After Insert Trigger.';

insert into Employee\_Demo\_Audit(Emp\_ID,Emp\_Name,Emp\_Sal,Audit\_Action,Audit\_Timestamp)

1. values (@empid,@empname,@empsal,@audit\_action,getdate());
2. PRINT 'AFTER INSERT trigger fired.'

Update Trigger

1. CREATE TRIGGER trgAfterUpdate ON dbo.Employee\_Demo
2. FOR UPDATE
3. AS
4. declare @empid int, @empname varchar(55), @empsal decimal(10,2), @audit\_action varchar(100);
5. select @empid=i.Emp\_ID from inserted i;
6. select @empname=i.Emp\_Name from inserted i;
7. select @empsal=i.Emp\_Sal from inserted i; if update(Emp\_Name)
8. set @audit\_action='Update Record --- After Update Trigger.';
9. if update (Emp\_Sal)
10. set @audit\_action='Update Record --- After Update Trigger.';
11. insert intoEmployee\_Demo\_Audit(Emp\_ID,Emp\_Name,Emp\_Sal,Audit\_Action,Audit\_Timestamp)
12. values (@empid,@empname,@empsal,@audit\_action,getdate());
13. PRINT 'AFTER UPDATE trigger fired.'
14. --Output will be

Delete Trigger

1. -- Create trigger on table Employee\_Demo for Delete statement
2. CREATE TRIGGER trgAfterDelete ON dbo.Employee\_Demo
3. FOR DELETE
4. AS
5. declare @empid int, @empname varchar(55), @empsal decimal(10,2), @audit\_action varchar(100); select @empid=d.Emp\_ID FROM deleted d;
6. select @empname=d.Emp\_Name from deleted d;
7. select @empsal=d.Emp\_Sal from deleted d;
8. select @audit\_action='Deleted -- After Delete Trigger.';
9. insert into Employee\_Demo\_Audit (Emp\_ID,Emp\_Name,Emp\_Sal,Audit\_Action,Audit\_Timestamp)
10. values (@empid,@empname,@empsal,@audit\_action,getdate());
11. PRINT 'AFTER DELETE TRIGGER fired.'

--Output will be

two types of Indexes in SQL Server:

1. Clustered Index
2. Non-Clustered Index

## **Clustered Index**

A clustered index defines the order in which data is physically stored in a table. Table data can be sorted in only way, therefore, there can be only one clustered index per table. In SQL Server, the primary key constraint automatically creates a clustered index on that particular column.

## **Non-Clustered Indexes**

A non-clustered index doesn’t sort the physical data inside the table. In fact, a non-clustered index is stored at one place and table data is stored in another place. This is similar to a textbook where the book content is located in one place and the index is located in another. This allows for more than one non-clustered index per table.

| **Parameters** | **Clustered** | **Non-clustered** |
| --- | --- | --- |
| Size | The size of the clustered index is quite large. | The size of the non-clustered index is small compared to the clustered index. |
| Data accessing | Faster | Slower compared to the clustered index |
| Type of key | By Default Primary Keys Of The Table is a Clustered Index. | It can be used with unique constraint on the table which acts as a composite key. |
| Main feature | A clustered index can improve the performance of data retrieval. | It should be created on columns which are used in joins. |

**Syntax of Normal Index:**

Create index Index\_name on Table\_Name(Column\_Name);

**Syntax for Unique Index:**

Create Unique index  Index\_name on Table\_name(Unique column name);

**Composite Index:**

When 2 or more columns in single table are related which each other and used in where condition of select statement then user should create composite index on the columns which are created

Create index CI\_ENO\_DEPTNO on Employee(Empno,Deptno);

**Syntax Function Based Indexes::**

Create index indexname on tablename(Function\_name(column\_name));

## **Inline Table-Valued Functions**

USE schooldb

GO

CREATE FUNCTION BornBefore

(

@DOB AS DATETIME

)

RETURNS TABLE

AS

RETURN

SELECT \* FROM student

WHERE DOB < @DOB

## **Multi-Statement Table-Valued Function (MSTVF)**

USE schooldb

GO

CREATE FUNCTION GetBornBetween

(

@YearAfter AS DATETIME,

@YearBefore AS DATETIME

)

RETURNS @People TABLE

(

Name VARCHAR (MAX),

Gender VARCHAR(MAX),

DOB DATETIME,

Job VARCHAR(10)

)

AS

BEGIN

INSERT INTO @People

SELECT name, gender, DOB, 'student'

FROM student

WHERE DOB BETWEEN @YearAfter AND @YearBefore

INSERT INTO @People

SELECT name, gender, DOB, 'teacher'

FROM teacher

WHERE DOB BETWEEN @YearAfter AND @YearBefore

RETURN

END

**HTML**

|  |  |
| --- | --- |
| **HTML** | **HTML5** |
| It didn’t support audio and video without the use of flash player support. | It supports audio and video controls with the use of <audio> and <video> tags. |
| It uses cookies to store temporary data. | It uses SQL databases and application cache to store offline data. |
| Does not allow JavaScript to run in browser. | Allows JavaScript to run in background. This is possible due to JS Web worker API in HTML5. |
| It does not allow drag and drop effects. | It allows drag and drop effects. |
| It works with all old browsers. | It supported by all new browser like Firefox, Mozilla, Chrome, Safari, etc. |
| Doctype declaration is too long and complicated. | Doctype declaration is quite simple and easy. |
| Attributes like charset, async and ping are absent in HTML. | Attributes of charset, async and ping are a part of HTML 5. |

* The <!DOCTYPE> declaration must be the very first thing in your HTML document, before the <html> tag.
* The <!DOCTYPE> declaration is not an HTML tag; it is an instruction to the web browser about what version of HTML the page is written in.
* In HTML 4.01, the <!DOCTYPE> declaration refers to a DTD, because HTML 4.01 was based on SGML. The DTD specifies the rules for the markup language, so that the browsers render the content correctly.

$.ajax({

type: "POST",

url: "../Webservices/EmployeeService.asmx/GetEmployeeOrders",

data: "{empid: empid}",

contentType: "application/json; charset=utf-8",

dataType: "json",

success: function(result) {

alert(result.d);

}

$(document).ready(function() {

// executes when HTML-Document is loaded and DOM is ready

console.log("document is ready");

});

$(window).load(function() {

// executes when complete page is fully loaded, including all frames, objects and images

console.log("window is loaded");

});

How to compare two arrays

var array1 = ["GESU687543", "TCNU315504", "TGHU394463"],

array2 = ["TCNU315504", "TRIU805499", "CMAU029901", "GESU687543", "TGHU394463", "NEUL0325B"],

result = array2.filter(function (a) {

return array1.indexOf(a) === -1;

});