C# Properties

C# Properites doesn't have storage location. C# Properites are extension of fields and accessed like fields.

The Properties have accessors that are used to set, get or compute their values.

Usage of C# Properties

1. C# Properties can be read-only or write-only.
2. We can have logic while setting values in the C# Properties.
3. We make fields of the class private, so that fields can't be accessed from outside the class directly. Now we are forced to use C# properties for setting or getting values.

C# Properties Example

```
1.  using System;
2.    public class Employee
3.     {
4.        private string name;
5.
6.        public string Name
7.        {
8.           get
9.           {
10.             return name;
11.          }
12.          set
13.          {
14.             name = value;
15.          }
16.       }
17.   }
18.   class TestEmployee{
19.      public static void Main(string[] args)
20.       {
21.          Employee e1 = new Employee();
22.          e1.Name = "Sonoo Jaiswal";
23.          Console.WriteLine("Employee Name: " + e1.Name);
24.
25.       }
26.    }
```

Output:

```
Employee Name: Sonoo Jaiswal
```

C# Properties Example 2: having logic while setting value

```
1.  using System;
2.    public class Employee
3.     {
4.        private string name;
5.
6.        public string Name
7.        {
8.           get
9.           {
10.             return name;
11.          }
12.          set
13.          {
14.             name = value+" JavaTpoint";
15.
16.          }
17.       }
18.   }
19.   class TestEmployee{
20.      public static void Main(string[] args)
21.       {
22.          Employee e1 = new Employee();
23.          e1.Name = "Sonoo";
24.          Console.WriteLine("Employee Name: " + e1.Name);
25.       }
26.    }
```

Output:

```
Employee Name: Sonoo JavaTpoint
```

C# Properties Example 3: read-only property

```
1.  using System;
2.    public class Employee
3.     {
4.        private static int counter;
5.
6.        public Employee()
7.        {
8.           counter++;
9.        }
10.       public static int Counter
11.       {
```

```
        get
        {
            return counter;
        }
     }
  }
  class TestEmployee{
     public static void Main(string[] args)
     {
        Employee e1 = new Employee();
        Employee e2 = new Employee();
        Employee e3 = new Employee();
        //e1.Counter = 10;//Compile Time Error: Can't set value

        Console.WriteLine("No. of Employees: " + Employee.Counter);
     }
   }
```

Output:

```
No. of Employees: 3
```