Accessors and Indexers in C#

Contents

- ✓ Properties.
- ✓ Accessors.
- ✓ Example of property.
- ✓ Indexers.
- ✓ Use of indexers.
- ✓ Example of indexer.
- ✓ Difference of property & indexer.

PROPERTIES

- Properties are named members of classes, structures, and interfaces. Member variables or methods in a class or structures are called Fields. Properties are an extension of fields and are accessed using the same syntax. They use accessors through which the values of the private fields can be read, written, or manipulated.
- Properties do not name the storage locations. Instead, they have accessors that read, write, or compute their values.
- For example, let us have a class named Student, with private fields for age, name, and code. We cannot directly access these fields from outside the class scope, but we can have properties for accessing these private fields.

ACCESSORS

The accessor of a property contains the executable statements that helps in getting (reading or computing) or setting (writing) the property. In short getter and setter methods are known as accessors.

Property : Simple Example

```
public class Student
{
   private string Name;
   public string name
      {
            get
            {
                  return name;
            }
            set
            {
                  name = value;
            }
      }
}
```

```
Student s = new Student();
s.Name = "Hemant";
Console.WriteLine(s);
```

INDEXERS

An indexer allows an object to be indexed such as an array. When you define an indexer for a class, this class behaves similar to a virtual array. You can then access the instance of this class using the array access operator ([ ]).

USE OF INDEXERS

- Declaration of behavior of an indexer is to some extent similar to a property. Similar to the properties, you use get and set accessors for defining an indexer. However, properties return or set a specific data member, whereas indexers returns or sets a particular value from the object instance. In other words, it breaks the instance data into smaller parts and indexes each part, gets or sets each part.
- Defining a property involves providing a property name. Indexers are not defined with names, but with the this keyword, which refers to the object instance. The following example demonstrates the concept:

Indexer: Simple Example

```
public class ListBox
{
   private string[] items;
   public string this[int index]
      {
            get
            {
                  return items[index];
            }
            set
            {
                  items[index] = value;
            }
      }
}
```

```
ListBox listBox = new ListBox();
listBox[0] = "hello";
Console.WriteLine(listBox[0]);
```

Difference between Property & Indexer

Property	Indexer
Allows methods to be called as if they were public data members.	Allows elements of an internal collection of an object to be accessed by using array notation on the object itself.
Accessed through a simple name.	Accessed through an index.
Can be a static or an instance member.	Must be an instance member.
A get accessor of a property has no parameters.	A **get** accessor of an indexer has the same formal parameter list as the indexer.
A set accessor of a property contains the implicit **value** parameter.	A **set** accessor of an indexer has the same formal parameter list as the indexer, and also to the value parameter.