**Visual Basic Properties (GET, SET)**

In visual basic, Property is an extension of the class variable and it provides a mechanism to read, write or change the value of the class variable without affecting the external way of accessing it in our applications.

In visual basic, properties can contain one or two code blocks called accessors and those are called a Get accessor and Set accessor. By using Get and Set accessors, we can change the internal implementation of class variables and expose it without effecting the external way of accessing it based on our requirements.

Generally, in object-oriented programming languages like visual basic we need to define fields as Private, and use the properties to access their values in Public way with Get and Set accessors.

Following is the syntax of defining a property with Get and Set accessor using Property keyword in a visual basic programming language.

<access\_modifier> Property <property\_name> As <return\_type>

{

Get

// return property value

End Get

Set

// Set a new value

End Set

}

If you observe the above syntax, we used an access modifier, Property keyword, and return type to define a property along with Get and Set accessors to make the required modifications to the class variables based on our requirements.

Here, the Get accessor code block will be executed whenever the property is read and the code block of Set accessor will be executed whenever the property is assigned to a new value.

In visual basic, the properties are categorized as three types, they are

**Type** **Description**

Read-Write A property which contains a both Get and Set accessors with Property keyword, we will call it as read-write property.

Read-Only A property which contains only Get accessor with ReadOnly property, we will call it as a read-only property.

Write-Only A property which contains only Set accessor with WriteOnly property, we will call it as write-only property.

In visual basic, Properties won’t accept any parameters and we should not pass property as a reference or out parameter in our application.

Following is the simple example of defining the private variable and property in a visual basic programming language.

Class User

Private name As String

Public Property Uname As String

Get

Return name

End Get

Set(ByVal value As String)

name = value

End Set

End Property

End Class

If you observe the above example, we defined a property called “Uname” and we used Get accessor to return the property value and Set accessors to set the new value. Here, the value keyword in Set accessor is used to define the value that is being assigned by Set accessor.

In visual basic, the Get accessor needs to be use only to return the field value or to compute it and return it but we should not use it for changing the state of an object.

As discussed, we can extend the behaviour of class variables using properties Get and Set accessors. Following is the example of extending the behaviours of Private variable in property using Get and Set accessors in a visual basic programming language.

Class User

Private name As String = "sangram "

Public Property Uname As String

Get

Return name.ToUpper()

End Get

Set(ByVal value As String)

If value = "sangram" Then name = value

End Set

End Property

End Class

If you observe above example, we are extending the behaviour of Private variable name using property called Uname with Get and Set accessors by performing some validations like to make sure Uname value is equals to “sangram” using Set accessor and converting the property text to uppercase with Get accessor.

Here the field “name” is marked as Private. So, if you want to make any changes to this field, you can do it only by calling the property (UName).

In visual basic properties, the Get accessor will be invoked while reading the value of property and when we assign a new value to the property, then the Set accessor will be invoked by using an argument that provides a new value.

Following is the example of invoking the Get and Set accessors of properties in a visual basic programming language.

Dim u As User = New User()

u.Uname = "Rutesh" ' Set accessor will invoke

Console.WriteLine(u.Uname) ' Get accessor will invoke

In the above example, when we assign a new value to the property, the Set accessor will be invoked and the Get accessor will invoked when we try to the read the value from property.

**Visual Basic Properties (Get, Set) Example**

Following is the example of defining the properties with Property keyword, Get and Set accessors to implement required validations without effecting the external way of using it in visual basic programming language.

Module Module1

Class User

Private location As String

Private name As String = "Sucharita Kumari"

Public Property Ulocation As String

Get

Return location

End Get

Set(ByVal value As String)

location = value

End Set

End Property

Public Property Uname As String

Get

Return name.ToUpper()

End Get

Set(ByVal value As String)

If value = "Sucharita" Then name = value

End Set

End Property

End Class

Sub Main()

Dim u As User = New User()

u.Uname = "Saswati"

u.Ulocation = "Visakhapatnam"

Console.WriteLine("Name: " & u.Uname)

Console.WriteLine("Location: " & u.Ulocation)

Console.WriteLine("Press Enter Key to Exit..")

Console.ReadLine()

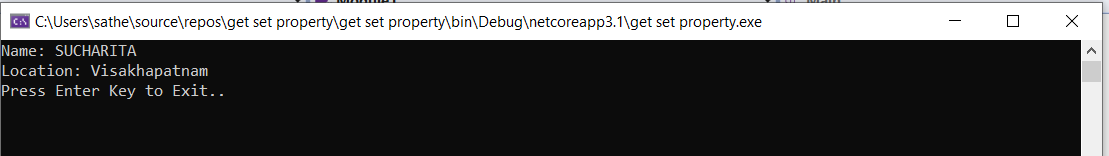
End Sub

End Module

If you observe the above example, we are extending the behaviour of private variables (name, location) using properties (Uname, Ulocation) with Property keyword, Get and Set accessors by performing some validations like to make sure Uname value is equals to only “Sucharita” using Set accessor and converting property text to uppercase with Get accessor.

When we execute the above visual basic program, we will get the result as shown below.

Visual Basic Properties (Get, Set) Example Result



If you observe the above example, our variable text converted to upper case and even after we set the variable text as “Saswati”, it displayed text as “Sucharita” because of the Set accessor validation fails in property.

**Visual Basic Create Read-Only Properties**

As discussed, if property contains only Get accessor with ReadOnly property, we will call it as read-only property. Following is the example of creating read-only properties in a visual basic programming language.

Module Module1

Class User

Private name As String

Private location As String

Public Sub New(ByVal a As String, ByVal b As String)

name = a

location = b

End Sub

Public ReadOnly Property Uname As String

Get

Return name

End Get

End Property

Public ReadOnly Property Ulocation As String

Get

Return location

End Get

End Property

End Class

Sub Main()

Dim u As User = New User("Rutesh", "Hyderabad")

' Compile-time error

' u.Name = "Abhijit";

' Get accessor will invoke

Console.WriteLine("Name: " & u.Uname)

' Get accessor will invoke

Console.WriteLine("Location: " & u.Ulocation)

Console.WriteLine("Press Enter Key to Exit..")

Console.ReadLine()

End Sub

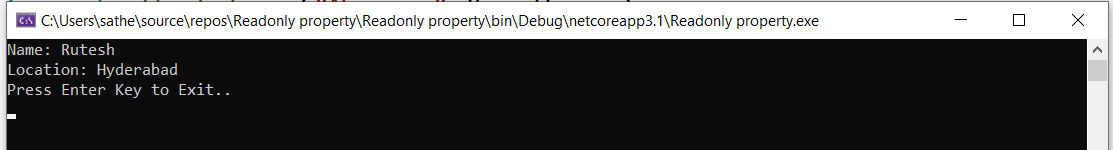
End Module

If you observe the above example, we created properties using only Get accessor with ReadOnly property to make the properties are read-only based on our requirements.

In case, if we uncomment the commented code, we will get a compile-time error because our Uname property doesn’t contain any Set accessor to set the new value. It’s just read-only property.

When we execute the above visual basic program, we will get a result like as shown below.

Visual Basic Create Readonly Properties Example Result



This is how we can create the read-only properties in visual basic applications based on our requirements.

**Visual Basic Create Write Only Properties**

As discussed, if property contains the only Set accessor with WriteOnly property, we will call it as write-only property. Following is the example of creating write-only properties in a visual basic programming language.

Module Module1

Class User

Private name As String

Public WriteOnly Property Uname As String

Set(ByVal value As String)

name = value

End Set

End Property

Private location As String

Public WriteOnly Property Ulocation As String

Set(ByVal value As String)

location = value

End Set

End Property

Public Sub GetUserDetails()

Console.WriteLine("Name: " & name)

Console.WriteLine("Location: " & location)

End Sub

End Class

Sub Main()

Dim u As User = New User()

u.Uname = "Subham"

u.Ulocation = "Koraput"

' Compile-time error

'Console.WriteLine(u.Uname);

u.GetUserDetails()

Console.WriteLine("Press Enter Key to Exit..")

Console.ReadLine()

End Sub

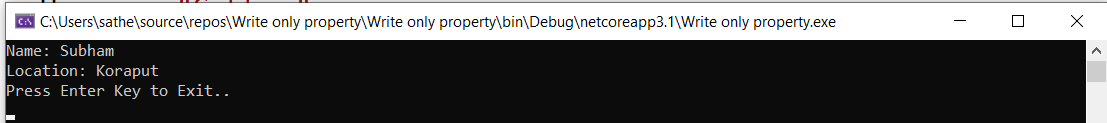
End Module

If you observe the above example, we created properties using only Set accessor with WriteOnly property to make the properties are write-only based on our requirements.

In case, if we uncomment the commented code, we will get the compile-time error because our Uname property doesn’t contain any Get accessor to return a value. It’s a just write-only property.

When we execute above visual basic program, we will get a result like as shown below.

Visual Basic Create WriteOnly Properties Example Result



This is how we can create write-only properties in visual basic applications based on our requirements.

**Visual Basic Auto Implemented Properties**

In visual basic, a property is called as an auto-implemented property when creating it with only Property keyword.

Generally, the auto-implemented properties are useful whenever there is no logic implementation required in property accessors.

Following is the example of creating the auto-implemented properties using Property keyword in a visual basic programming language.

Module Module1

Class User

Public Property Name As String

Public Property Location As String

End Class

Sub Main()

Dim u As User = New User()

u.Name = "Siddhant"

u.Location = "Koraput"

Console.WriteLine("Name: " & u.Name)

Console.WriteLine("Location: " & u.Location)

Console.WriteLine("Press Enter Key to Exit..")

Console.ReadLine()

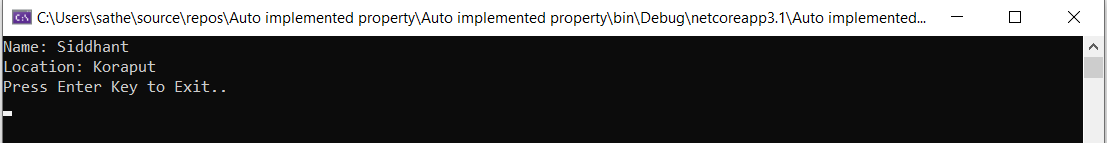
End Sub

End Module

If you observe the above example, we created auto-implemented properties using only Property keyword without having any logic implementation.

When we execute the above visual basic program, we will get the result as shown below.

Visual basic Auto Implemented Properties Example Result



This is how we can create auto-implemented properties in visual basic applications based on our requirements.

**Visual Basic Properties Overview**

The following are the important points which we need to remember about properties in a visual basic programming language.

In visual basic, properties will enable the class variables to expose in public way using Get and Set accessors with Property keyword by hiding implementation details.

In properties, the Get accessor is useful to return the property value and the Set accessor is useful to assign a new value.

The value keyword in Set accessor is useful to define the value which is going to be assigned by the Set accessor.

In visual basic, properties are categorized as read-write, read-only or write-only.