

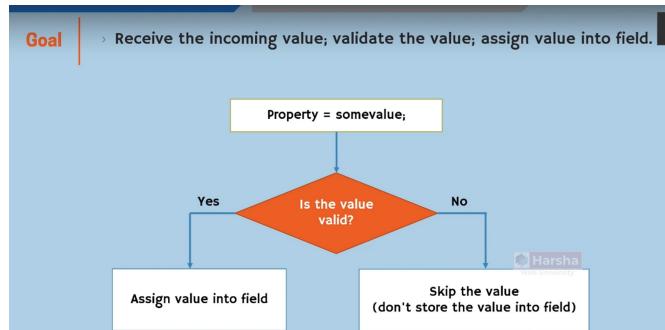
Properties

Harsha Vardhan
UI Expert
.NET Expert
Module Lead
Corporate Instructor
Harsha Web University

85. Creating Properties

- 1 Understanding Properties
What is Property and its features.
- 2 Set & Get Accessors
How 'set' accessor initializes value of field.
How 'get' accessor returns value of field.
- 3 Readonly Properties
Properties with only 'get' accessor.
- 4 Write-only Properties
Properties with only 'set' accessor.

- 5 Auto-Implemented Properties
Properties without definition of 'set' and 'get' accessors.
New feature of C# 3.0.
- 6 Auto-Implemented Property Initializer
Initializing default value into auto-implemented properties.
New feature of C# 6.0.



Introducing Properties Next: Syntax of Property

What

- Property is a collection of two accessors (get-accessor and set-accessor).

```

class Car
{
    private string _carBrand;
    public string CarBrand
    {
        set
        {
            this._carBrand = value;
        }
        get
        {
            return _carBrand;
        }
    }
}
  
```

Declaration of private field.
Validation & Initialization of field
Returning value of field

CarBrand = value;

CarBrand

Syntax of Property → Next: Set Accessor (vs) Get Accessor

Syntax

```
accessModifier modifier type PropertyName
{
    set { field = value; } | Set accessor
    get { return field; } | Get accessor
}
```

1. private
2. protected
3. private protected
4. internal
5. protected internal
6. public

1. static
2. virtual
3. abstract
4. override
5. new
6. sealed

Set Accessor (vs) Get Accessor → Next: Features and Advantages of Properties

Set Accessor	Get Accessor
<code>set { field = value; }</code>	<code>get { return field; }</code>
<ul style="list-style-type: none"> Used to validate the incoming value and assign the same into field. Executes automatically when some value is assigned into the property. 	<ul style="list-style-type: none"> Used to calculate value and return the same (or) return the value of field as-it-is. Executes automatically when the property is retrieved.

Set Accessor (vs) Get Accessor → Next: Features and Advantages of Properties

Set Accessor	Get Accessor
<code>set { field = value; }</code>	<code>get { return field; }</code>
<ul style="list-style-type: none"> Has a default (implicit) parameter called "value", which represents current value i.e. assigned to the property. Can't have any additional parameters. But can't return any value. 	<ul style="list-style-type: none"> Has no implicit parameters. Can't have parameters. Should return value of field.

Properties and Methods are just part of the class, they will not occupy any memory.

5. Creating Properties → Features and Advantages of Properties

- Properties create a protection layer around fields, preventing assignment of invalid values into properties & also do some calculation automatically when someone has invoked the property.
- No memory will be allocated for the property.
- Access modifier is applicable for the property, set accessor and get accessor individually.
- But access modifiers of accessors must be more restrictive than access modifier of property.
- Note: You can set access modifiers on either of 'set' accessor or 'get' accessor; not both at once.

```
internal modifier dataType PropertyName
{
    private set { property = value; }
    get { return property; }
}
```

The Benefit of Property is that, we can add validation logic before assigning the value into the corresponding field.

86. Readonly & Writeonly Properties → Agenda

- 1 Understanding Properties
What is Property and its features.
- 2 Set & Get Accessors
How 'set' accessor initializes value of field.
How 'get' accessor returns value of field.
- 3 Readonly Properties
Properties with only 'get' accessor.
- 4 Write-only Properties
Properties with only 'set' accessor.

Use Readonly Properties when you want to allow the user to read the value of the field; but don't want to assign the value into the field.

For example, in the Employee class, the salary of the employee can be read any no of times. But you don't want to allow of the user to modify the value of the Salary property of the Employee Class.

Properties | Read-only & Write-only Property | Next: Auto-implemented Properties | Harsha

Readonly Property	Write-only Constructor
<pre>accessModifier type propertyName { get { return field; } }</pre>	<pre>accessModifier type propertyName { set { field = value; } }</pre>
<ul style="list-style-type: none">› Contains ONLY 'get' accessor.› Reads & returns the value of field; but not modifies the value of field.	<ul style="list-style-type: none">› Contains ONLY 'set' accessor.› Validates & assign incoming value into field; but return the value.

5 Auto-implemented Properties
Properties without definition of 'set' and 'get' accessors.
New feature of C# 3.0.

6 Auto-implemented Property Initializer
Initializing default value into auto-implemented properties.
New feature of C# 6.0.

Advisor Accessibility

What

- › Property with no definition for set-accessor and get-accessor.
- › Used to create property easily (with shorter syntax).
- › Creates a private field (with name as `_propertyName`) automatically, while compilation-time.
- › Auto-implemented property can be Read-only (only 'get' accessor) property; but it can't be Write-only (only 'set' accessor).

Syntax

```
accessModifier modifier type propertyName
{
    accessModifier set;
    accessModifier get;
}
```

Harsha Web University

- › Useful only when you don't want to write any validation or calculation logic.

6 Auto-implemented Property Initializer
Initializing default value into auto-implemented properties.
New feature of C# 6.0.

Harsha Web University

What

- › New feature in C# 6.0
- › You can initialize value into auto-implemented property.

Syntax

```
accessModifier modifier type propertyName { set; get; } = value;
```

Harsha Web University

Key points to remember



- › It is recommended to use Properties always in real-time projects.
- › You can also use 'Auto-implemented properties' to simplify the code.
- › Properties doesn't occupy any memory (will not be stored).
- › Properties forms a protection layer surrounding the private field that validates the incoming value before assigning into field.
- › Read-only property has only 'get' accessor; Write-only property has only 'set' accessor.
- › Properties can't have additional parameters.

C#



Harsha Vardhan

- › UI Expert
- › .NET Expert
- › Module Lead
- › Corporate Instructor

Indexers

Rarely used in C#

Introducing Indexers > Next: Syntax of Indexer

Goal

- › Receive a number / string. Search for the particular item among a group of items; set or get value into the group of items.
- › It provides shorter syntax to access a group of items.

```

    graph TD
      A[referenceVariable[index or name]] --> B{Find item based on given index / name}
      B --> C[Assign value into group]
      B --> D[Skip the value (don't store the value into field)]
  
```

Introducing Indexers > Next: Syntax of Indexer

What

- › Indexer is a special member of class, which contains set-accessor and get-accessor to access a group of items / elements.

```

class Car
{
    private string _brands = "brand1, brand2, brand3";

    public string this[int index]
    {
        set
        {
            this._brands = value;
        }
        get
        {
            return _brands;
        }
    }
}
  
```

[0] = "other brand"

[0]

Syntax of Indexer Next: Key Points to Remember

Syntax

```
accessModifier modifier type this[parameter]
{
    set { field = value; }      | Set accessor
    get { return field; }       | Get accessor
}
```

1. private
2. protected
3. private protected
4. internal
5. protected internal
6. public

1. virtual
2. abstract
3. override
4. new
5. sealed

Harsha
Web University

Key Points to Remember

Key points to remember

- Indexers are always created with 'this' keyword.
- Indexers are generally used to access group of elements (items).
- Parameterized properties are called indexer.
- Indexers are implemented through get and set accessors along with the [] operator.
- Indexer must have one or more parameters.
- ref and out parameter modifiers are not permitted in indexer.
- Indexer can't be static.
- Indexer is identified by its signature (syntax of calling, where as a property is identified it's name).
- Indexer can be overloaded.