Creating Good Properties

Deborah Kurata

@deborahkurata | blogs.msmvps.com/deborahk/

Properties

```
private string productName;

public string ProductName
{
  get { return productName; }
  set { productName = value; }
}
```

Getter and setter functions

Guard access to the fields

Encapsulate the fields

Module Overview

Coding Properties

Auto-Implemented Properties

Property Accessibility

Additional Uses of Properties

Expression-Bodied Properties

Benefits of Properties

FAQ

Code in the Getter

```
private string productName;

public string ProductName
{
  get
  {
    var formattedValue =
                productName?.Trim();
    return formattedValue;
  }
  set { productName = value; }
}
```

Check the user's credentials

Check application state

Format the returned value

Log

Lazy loading

Code in the Setter

```
private string productName;
public string ProductName
{
  get { return productName; }
  set
  {
    if (value.Length < 3)
    {
      Message = "Too short";
    }
    else
    {
      productName = value;
    }
  }
}
```

Check the user's credentials

Check application state

Validate the incoming value

Log or change tracking

Format, convert, clean up

Property Best Practices

Do:

Naming

Define a meaningful name

Use PascalCasing

Add code in the getter to protect, format, initialize, …

Add code in the setter to protect, format, validate, …

Avoid:

Naming

Single character name

Abbreviations

Auto-Implemented Properties

```
private string productName;

public string ProductName
{
    get { return productName; }
    set { productName = value; }
}
```

```
public string Category { get; set; }
public int SequenceNumber { get; set; }
```

Concise property declaration

Implicit backing field

Don't allow code in the getter or setter

Best used for simple properties

Initializing Auto-Implemented Properties

```
public string Category { get; set; } = "Tools";
public int SequenceNumber { get; set; } = 1;
public Vendor productVendor { get; set; } = GetDefaultVendor();
```

```
public Vendor productVendor { get; set; }
public Product()
{
   this.ProductVendor = GetVendor();
}
```

Read-Only Auto-Implemented Properties

```
public int InventoryCount { get; }

public Product()
{
   this.InventoryCount = GetInventoryCount();
}
```

```
public int InventoryCount { get; } = InitializeCount();
```

Auto-Property Best Practices

Do:

Naming

Define a meaningful name

Use PascalCasing

Initialize on the declaration when needed

Avoid:

Naming

Single character name

Abbreviations

If property requires code in getter or setter

Property Accessibility

```
public string Category { get; set; }

protected string Category { get; set; }

internal string Category { get; set; }

protected internal string Category { get; set; }

private string Category { get; set; }
```

Property Accessibility (cont)

```
public string Category { internal get; private set; }

internal string Category { get; private set; }

internal string Category { public get; set; }
```

Select the
most restrictive accessibility
that still gets the job done

Additional Uses of Properties

Define concatenated values

Express calculations

Expose related object properties

Concatenated Values

```
public string LastName { get; set; }
public string FirstName { get; set; }


public string FullName
{
   get { return FirstName + " " + LastName; }
}
```

Calculations

```
public int Quantity { get; set; }
public int Price { get; set; }


public int LineItemTotal
{
   get { return Quantity * Price; }
}
```

Related Object Properties

```
public Vendor ProductVendor { get; set; }


public string VendorName
{
   get { return ProductVendor?.CompanyName; }
}
```

Syntax Shortcut

```
public string FullName
{
   get { return FirstName + " " + LastName; }
}


public string FullName => FirstName + " " + LastName;
```

Expression-Bodied Properties

```
public string FullName
{
 get { return FirstName + " " + LastName; }
}
```

```
public string FullName => FirstName + " " + LastName;
```

Syntax Shortcut

Read-only properties

Immediately return a value

Expression-Bodied Properties

```
public string FullName
{
 get { return FirstName + " " + LastName; }
}
```

```
public string FullName => FirstName + " " + LastName;
```

No curly braces

No get keyword

No return statement

Just a =>

Expression-Bodied Properties

```
public string FullName => FirstName + " " + LastName;


public int ItemTotal => Quantity * Price;


public string VendorName => ProductVendor?.CompanyName;
```

Benefits of Properties

```
internal string Category { get; private set; }
public int SequenceNumber { get; internal set; } = 1;
```

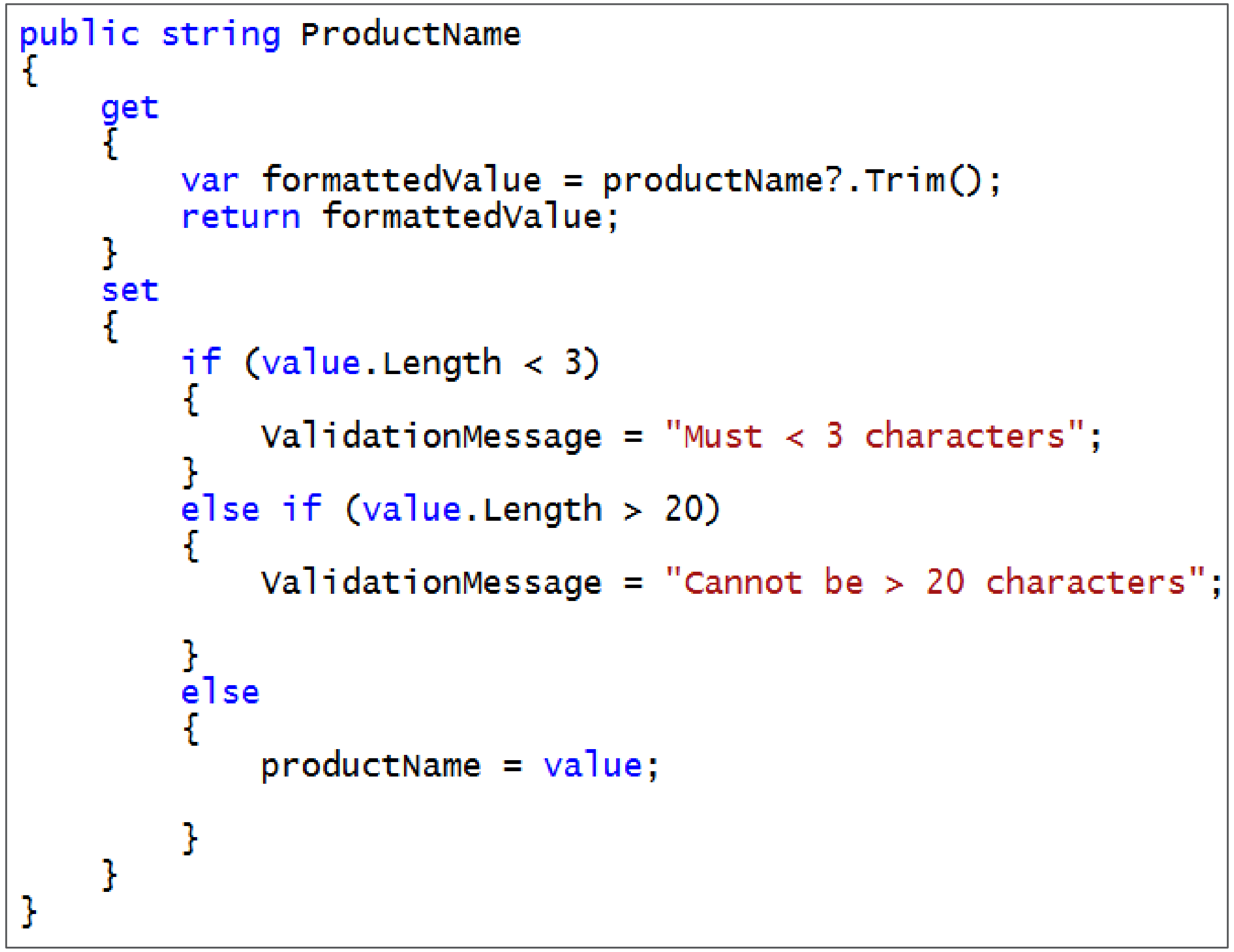
Fine grained access control

Execute code

Set break points or logging

Available for data binding

Benefits of Properties



```
public string ProductName
{
    get
    {
        var formattedValue = productName?.Trim();
        return formattedValue;
    }
    set
    {
        if (value.Length < 3)
        {
            ValidationMessage = "Must < 3 characters";
        }
        else if (value.Length > 20)
        {
            ValidationMessage = "Cannot be > 20 characters";

        }
        else
        {
            productName = value;

        }
    }
}
```

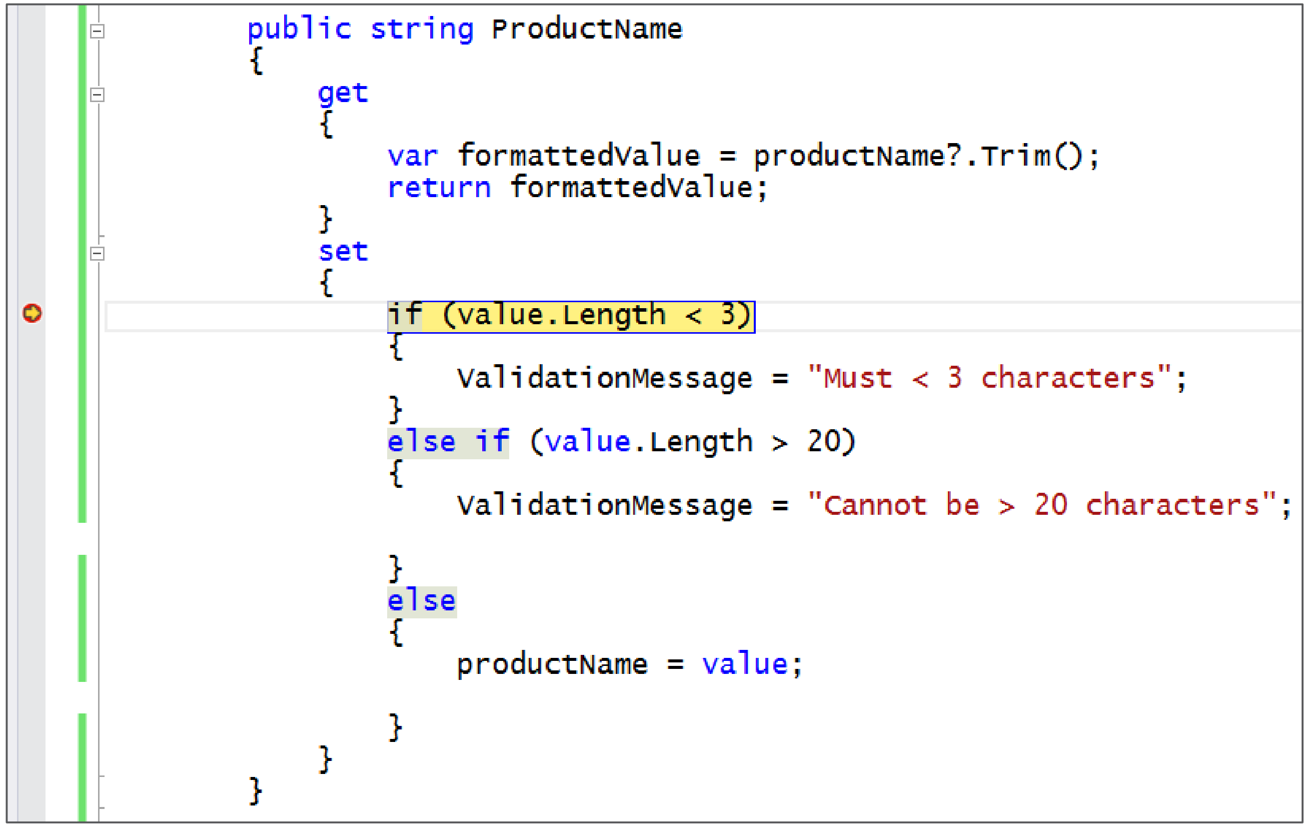
Fine grained access control

Execute code

Set break points or logging

Available for data binding

Benefits of Properties



```
        public string ProductName
        {
            get
            {
                var formattedValue = productName?.Trim();
                return formattedValue;
            }
            set
            {
                if (value.Length < 3)
                {
                    ValidationMessage = "Must < 3 characters";
                }
                else if (value.Length > 20)
                {
                    ValidationMessage = "Cannot be > 20 characters";

                }
                else
                {
                    productName = value;

                }
            }
        }
```

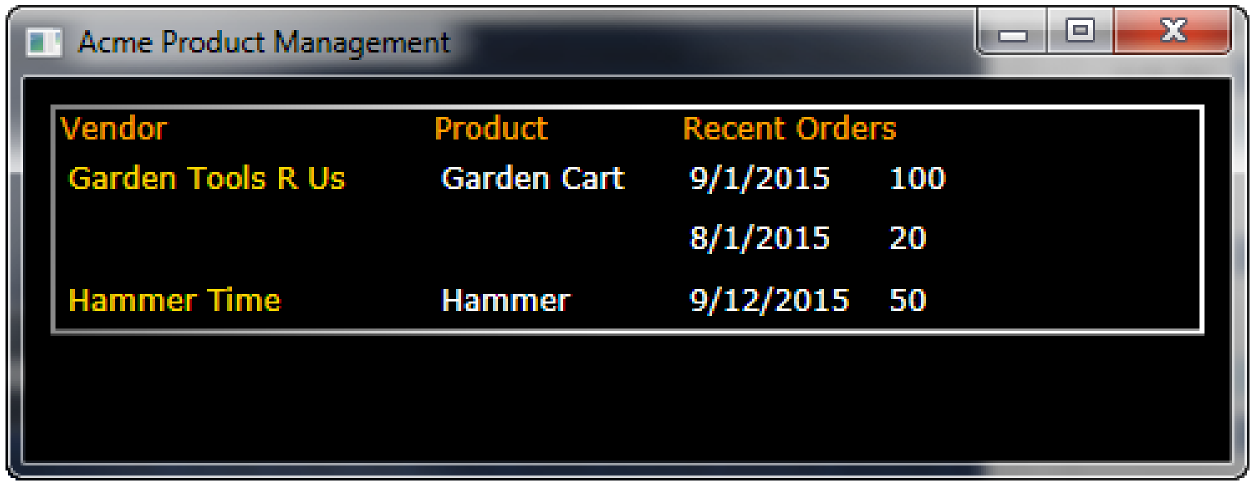
Fine grained access control

Execute code

Set break points or logging

Available for data binding

Benefits of Properties



Vendor	Product	Recent Orders	
Garden Tools R Us	Garden Cart	9/1/2015	100
		8/1/2015	20
Hammer Time	Hammer	9/12/2015	50

Fine grained access control

Execute code

Set break points or logging

Available for data binding

Frequently Asked Questions

- What is the primary purpose of a property?
  - To guard access to the fields of the class
  - And optionally provide a location for logic
- What are auto-implemented properties?
  - Short cut syntax for defining an implicit backing field with its associated property getter and setter

Frequently Asked Questions (cont)

- When should you use an auto-implemented property?
  - When creating simple properties for a class
- When shouldn't you use an auto-implemented property?
  - If the property requires any code in the getter or setter.

This Module Covered

Coding Properties

Auto-Implemented Properties

Property Accessibility

Additional Uses of Properties

Expression-Bodied Properties

Benefits of Properties