

Attributes

Associating declarative information with
C# code



Contents

- About Attributes
- Pre-defined Attributes
- Developing Custom Attributes
- Controlling Attribute Usage
- Parameters
- Querying at Runtime
- Attributes and Windows Forms

About Attributes

- For defining declarative tags (metadata)
- Can be placed on source code entities (Assemblies, Modules, Classes, etc)
- Attribute information retrieved at runtime via reflection
- Can use pre-defined attributes, or define custom ones

Pre-defined Attributes

- The *Obsolete* attribute marks an entity that shouldn't be used
 - *true* indicates error at compile time
- About 200 or so pre-defined attributes
 - <https://msdn.microsoft.com/en-us/library/2e39z096.aspx>

```
using System;
public class AnyClass
{
    [Obsolete("Don't use Old method, use New method", true)]
    static void Old( ) { }

    static void New( ) { }

    public static void Main( )
    {
        Old( );
    }
}
```

AnyClass.Old()' is obsolete: 'Don't use Old method, use New method'

Developing Custom Attributes

- Derive (directly or indirectly) from `System.Attribute`
- The 'Attribute' suffix is a naming convention
- When using the attribute we can drop the suffix

```
public class HelpAttribute : Attribute  
{  
}
```

```
[Help()]  
class ClassWithAttributes  
{  
}
```

Developing Custom Attributes

- Properties can be queried at run time

```
public class HelpAttribute : Attribute
{
    public HelpAttribute(String description) {
        this.description = description;
    }

    protected String description;
    public String Description {
        get {
            return this.description;
        }
    }
}
```

```
[Help("This class does nothing")]
class ClassWithAttributes
{
}
```

Controlling Attribute Usage

- *AttributeUsage* controls where our custom attribute can be placed
- Contains 3 properties
 - ValidOn:
 - which entities can use the attribute
 - AllowMultiple:
 - can the attribute be placed on an entity more than once
 - Inherited:
 - is the attribute inherited by derived classes

```
[AttributeUsage(AttributeTargets.Class,  
    AllowMultiple=false,  
    Inherited=false)]  
public class HelpAttribute : Attribute  
{  
    public HelpAttribute(String description) {  
        this.description = description;  
    }  
  
    protected String description;  
    public String Description {  
        get {  
            return this.description;  
        }  
    }  
}
```

AttributeUsage - ValidOn

- *AttributeTarget.Class* specifies only valid on classes

```
[AttributeUsage(AttributeTargets.Class, AllowMultiple=false, Inherited=false)]  
public class HelpAttribute : Attribute { }
```

- Otherwise produces error

```
ClassWithAttributes.cs: Attribute 'Help' is not valid on this declaration type. It is  
valid on 'class' declarations only.
```

- Possible values for the attribute target:

```
Assembly, Module, Class, Struct, Enum, Constructor, Method, Property, Field, Event,  
Interface, Parameter, Delegate,  
All (Assembly | Module | Class | Struct | Enum | Constructor | Method | Property | Field  
| Event | Interface | Parameter | Delegate),  
ClassMembers (Class | Struct | Enum | Constructor | Method | Property | Field | Event |  
Delegate | Interface)
```


AttributeUsage - AllowMultiple

- Specifies if an attribute can be placed on an entity can more than once

```
using System;

[Help("This class does nothing")]
[Help("it contains a do-nothing method")]
public class ClassWithAttributes
{
}
```



ClassWithAttributes.cs: Duplicate 'Help' attribute

AttributeUsage - Inherited

`[AttributeUsage(AttributeTargets.Class,
AllowMultiple = false, Inherited = false)]`

Querying a derived class will not find the attribute

`[AttributeUsage(AttributeTargets.Class,
AllowMultiple = true, Inherited = false)]`

Querying a derived class will not find the attribute

```
[Help("BaseClass")]  
public class Base  
{  
}  
  
public class Derive : Base  
{  
}
```

AttributeUsage - Inherited

`[AttributeUsage(AttributeTargets.Class,
AllowMultiple = false, Inherited = true)]`

Gets the derived class attribute only, as allow multiples is false. Base class attribute overridden by derived class

`[AttributeUsage(AttributeTargets.Class,
AllowMultiple = true, Inherited = true)]`

Get both attributes

```
[Help("BaseClass")]  
public class Base  
{  
}  
  
[Help("DeriveClass")]  
public class Derive : Base  
{  
}
```

Parameters

- Parameters in the Attribute class constructor are mandatory
- Named parameters are optional and are not parameters of the constructor
- Don't use multiple constructors for optional parameters
 - Mark them as named parameters

```
public class HelpAttribute : Attribute
{
    public HelpAttribute(String description) {
        this.description = description;
        this.version = "undefined";
    }

    protected String description;
    public String Description {
        get {
            return this.description;
        }
        set {
            description = value;
        }
    }

    protected String version;
    public String Version {
        get {
            return this.version;
        }
        set {
            this.version = value;
        }
    }
}
```

```
[Help("This class does nothing", Version = "1.0")]
class ClassWithAttributes
{
}
```

Querying at Runtime

- Query attributes at run time via reflection
- To get the attributes of a class, get the *Type*
 - Use the *typeof()* operator
- Get all attributes by calling *GetCustomAttributes*
 - Pass in *true* to search inheritance chain
- Similar process to get attributes on methods, fields, etc

Querying at Runtime

```
static void Main(string[] args)
{
    Type type = typeof(ClassWithAttributes);
    HelpAttribute HelpAttr;

    //Querying Class Attributes
    foreach (Attribute attr in type.GetCustomAttributes(true)) {
        HelpAttr = attr as HelpAttribute;
        if (null != HelpAttr) {
            Console.WriteLine("Description of ClassWithAttributes - (Version: {0}) :\n {1}",
                HelpAttr.Version, HelpAttr.Description);
        }
    }

    //Querying Class-Method Attributes
    foreach (MethodInfo method in type.GetMethods()) {
        foreach (Attribute attr in method.GetCustomAttributes(true)) {
            HelpAttr = attr as HelpAttribute;
            if (null != HelpAttr) {
                Console.WriteLine("Description of {0} - (Version: {1}) :\n {2}",
                    method.Name, HelpAttr.Version, HelpAttr.Description);
            }
        }
    }
}
```

Attributes and Windows Forms

- .NET Framework provides attributes you can apply to members of your custom controls
- Can affect either run-time or design-time behaviour
- The Windows Forms designer uses attributes extensively

Attributes on Custom Controls

- Using the custom clock control from the Win Forms lesson:
 - *CategoryAttribute*: name of the category to group the property / event
 - *DescriptionAttribute*: Specifies a description for a property/event
 - *DefaultValueAttribute*: Specifies the default value for a property.
- Complete list:
[https://msdn.microsoft.com/en-us/library/vstudio/ms171724\(v=vs.100\).aspx](https://msdn.microsoft.com/en-us/library/vstudio/ms171724(v=vs.100).aspx)

```
public class FirstControl: System.Windows.Forms.Control
{
    private bool showSeconds = true;

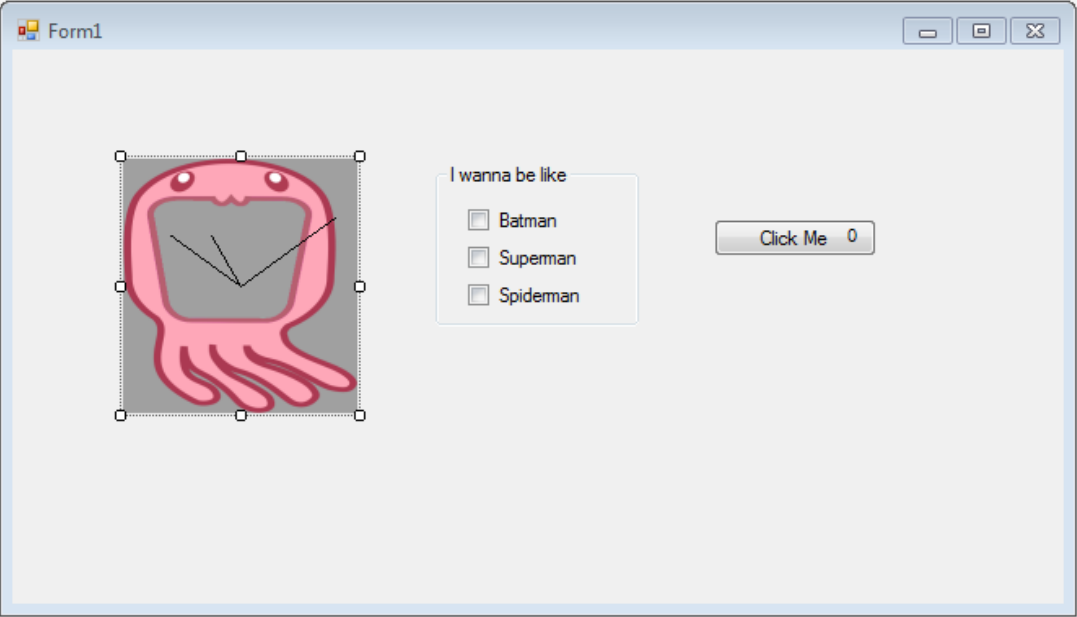
    [
        Category("Show Seconds"),
        Description("Show the second hand."),
        DefaultValue(true)
    ]
    public bool ShowSeconds
    {
        get
        {
            return showSeconds;
        }
        set
        {
            showSeconds = value;
            Invalidate();
        }
    }

    ...
}
```


Attributes on Custom Controls

Form1.cs [Design] X FirstControl.cs

Form1



I wanna be like

- ☐ Batman
- ☐ Superman
- ☐ Spideyman

Click Me 0

Properties

firstControl1 CustomControl.FirstControl

Locked	False
Modifiers	Private
Focus	
CausesValidation	True
Layout	
Anchor	Top, Left
Dock	None
Location	68, 67
Margin	3, 3, 3, 3
MaximumSize	0, 0
MinimumSize	0, 0
Padding	0, 0, 0, 0
Size	144, 156
Show Hours	
ShowHours	True
Show Minutes	
ShowMinutes	True
Show Seconds	
ShowSeconds	True

ShowSeconds

Show the second hand.

Summary

- Attributes proved a mechanism for defining additional information about entities in our program
- A set of pre-defined attributes exist, or we can create our own
- Use reflection to query these attributes at run time
- Add attributes to custom controls to specify run-time or design-time behaviour

References

- Sadaf Alvi. 2002. *Attributes in C#*. [ONLINE] Available at: http://www.codeproject.com/Articles/2933/Attributes-in-C_. [Accessed 09 February 15].
- Microsoft. 2015. *Attribute Hierarchy*. [ONLINE] Available at: <https://msdn.microsoft.com/en-us/library/2e39z096.aspx>. [Accessed 09 February 15].
- Microsoft. 2015. *Attributes in Windows Forms Controls*. [ONLINE] Available at: [https://msdn.microsoft.com/en-us/library/vstudio/ms171724\(v=vs.100\).aspx](https://msdn.microsoft.com/en-us/library/vstudio/ms171724(v=vs.100).aspx). [Accessed 10 February 15].