What are attributes in .NET?

Asked 16 years, 4 months ago Modified 5 years, 10 months ago

Viewed 70k times

211

What are attributes in .NET, what are they good for, and how do I create my own attributes?

c# .net glossary .net-attributes

Share

Improve this question

Follow

edited Apr 18, 2017 at 19:58

Peter Mortensen

31.6k ● 22 ● 109 ● 133

asked Aug 21, 2008 at 15:59

Corey

14.3k ● 7 ● 39 ● 35

11 Answers

Sorted by: Highest score (default)

152

Metadata. Data about your objects/methods/properties.

For example I might declare an Attribute called: DisplayOrder so I can easily control in what order properties should appear in the UI. I could then append it to a class and write some GUI components that extract the attributes and order the UI elements appropriately.

```
public class DisplayWrapper
{
    private UnderlyingClass underlyingObject;

    public DisplayWrapper(UnderlyingClass u)
    {
        underlyingObject = u;
    }

    [DisplayOrder(1)]
    public int SomeInt
    {
        get
        {
            return underlyingObject .SomeInt;
        }
    }

    [DisplayOrder(2)]
    public DateTime SomeDate
    {
        get
        {
            return underlyingObject .SomeDate;
        }
    }
}
```

Thereby ensuring that SomeInt is always displayed before SomeDate when working with my custom GUI components.

However, you'll see them most commonly used outside of the direct coding environment. For example the Windows Designer uses them extensively so it knows how to deal with custom made objects. Using the BrowsableAttribute like so:

```
[Browsable(false)]
public SomeCustomType DontShowThisInTheDesigner
{
    get{/*do something*/}
}
```

Tells the designer not to list this in the available properties in the Properties window at design time for example.

You *could* also use them for code-generation, pre-compile operations (such as Post-Sharp) or run-time operations such as Reflection.Emit. For example, you could write a bit of code for profiling that transparently wrapped every single call your code makes and times it. You could "opt-out" of the timing via an attribute that you place on particular methods.

```
public void SomeProfilingMethod(MethodInfo targetMetho
object[] args)
{
    bool time = true;
    foreach (Attribute a in target.GetCustomAttributes
    {
        if (a.GetType() is NoTimingAttribute)
        {
            time = false;
            break;
        }
    }
    if (time)
    {
        StopWatch stopWatch = new StopWatch();
        stopWatch.Start();
        targetMethod.Invoke(target, args);
        stopWatch.Stop();
        HandleTimingOutput(targetMethod, stopWatch.Dur
    }
    else
    {
        targetMethod.Invoke(target, args);
    }
}
```

Declaring them is easy, just make a class that inherits from Attribute.

```
public class DisplayOrderAttribute : Attribute
{
    private int order;

    public DisplayOrderAttribute(int order)
    {
        this.order = order;
    }

    public int Order
    {
        get { return order; }
    }
}
```

And remember that when you use the attribute you can omit the suffix "attribute" the compiler will add that for you.

NOTE: Attributes don't do anything by themselves - there needs to be some other code that uses them. Sometimes that code has been written for you but sometimes you have to write it yourself. For example, the C# compiler cares about some and certain frameworks frameworks use some (e.g. NUnit looks for [TestFixture] on a class and [Test] on a test method when loading an assembly). So when creating your own custom attribute be aware that it will not impact the behaviour of your code at all. You'll need to write the other part that checks attributes (via reflection) and act on them.

Share Improve this answer Follow

edited Feb 5, 2019 at 19:28

answered Aug 21, 2008 at 16:18

Quibblesome

25.4k ● 10 ● 62 ● 104

32 For what it's worth, this is a list of all (built in) .NET attributes: msdn.microsoft.com/en-us/library/aa311259(VS.71).aspx – wprl Sep 28, 2008 at 0:37

1 How would you use your "SomeProfilingMethod" as an attribute? – RayLoveless Oct 17, 2016 at 22:16

@RayLoveless its not an attribute, SomeProfilingMethod is the instrumentation code that is looking for profiling attributes. Specifically in the example I gave its looking for an "opt-out" attribute (NoTimingAttribute) as opposed to an "opt-in" one. The idea is that it times everything. – Quibblesome Oct 18, 2016 at 7:48

@Quibblesome could you add something like "Attributes don't do anything by themselves - there needs to be some *other* code to use them (compiler cares about couple, different frameworks use some). Just creating attribute will not impact behavior of the code - you need to write the other part that checks attributes (via reflection) and act on them". (or I can do that if you are ok). Many people expect attributes to magically work and none of the answers here clarify that. (or just link to stackoverflow.com/questions/4879521/… which covers it) – Alexei Levenkov Feb 5, 2019 at 18:05

only if you stop using Bing. Nah. j/k I use DuckDuckGo as my primary which mainly uses Bing iirc. :) – Quibblesome Feb 5, 2019 at 19:26

36

Many people have answered but no one has mentioned this so far...

Attributes are used heavily with reflection. Reflection is already pretty slow.

It is *very worthwhile* marking your custom attributes as being `sealed` classes to improve their runtime performance.

It is also a good idea to consider where it would be appropriate to use place such an attribute, and to attribute your attribute (!) to indicate this via `AttributeUsage`. The list of available attribute usages might surprise you:

- Assembly
- Module
- Class
- Struct
- Enum
- Constructor
- Method
- Property

- Field
- Event
- Interface
- Parameter
- Delegate
- ReturnValue
- GenericParameter
- All

It's also cool that the AttributeUsage attribute is part of the AttributeUsage attribute's signature. Whoa for circular dependencies!

```
[AttributeUsageAttribute(AttributeTargets.Class, Inher
public sealed class AttributeUsageAttribute : Attribut
```

Share Improve this answer Follow

answered Feb 16, 2009 at 18:02

Drew Noakes
310k ● 168 ● 696 ● 761

15

Attributes are a kind of meta data for tagging classes. This is often used in WinForms for example to hide controls from the toolbar, but can be implemented in your own application to enable instances of different classes to behave in specific ways.

Start by creating an attribute:

```
[AttributeUsage(AttributeTargets.Class, AllowMultiple=
public class SortOrderAttribute : Attribute
{
    public int SortOrder { get; set; }

    public SortOrderAttribute(int sortOrder)
    {
        this.SortOrder = sortOrder;
    }
}
```

All attribute classes must have the suffix "Attribute" to be valid.
After this is done, create a class that uses the attribute.

```
[SortOrder(23)]
public class MyClass
{
    public MyClass()
    {
    }
}
```

Now you can check a specific class' `SortOrderAttribute` (if it has one) by doing the following:

```
public class MyInvestigatorClass
{
    public void InvestigateTheAttribute()
    {
        // Get the type object for the class that is u
        // the attribute.
        Type type = typeof(MyClass);

        // Get all custom attributes for the type.
        object[] attributes = type.GetCustomAttributes
            typeof(SortOrderAttribute), true);
        // Now let's make sure that we got at least on
        if (attributes != null && attributes.Length > 
        {
            // Get the first attribute in the list of
            // that is of the type "SortOrderAttribute
            // be one since we said "AllowMultiple=fal
            SortOrderAttribute attribute =
                attributes[0] as SortOrderAttribute;

            // Now we can get the sort order for the c
            int sortOrder = attribute.SortOrder;
        }
    }
}
```

If you want to read more about this you can always check out MSDN which has a pretty good description.
I hope this helped you out!

Share Improve this answer

Follow

edited Mar 5, 2018 at 9:25

meJustAndrew
6,583 ● 10 ● 55 ● 86

answered Aug 21, 2008 at 16:33

Patrik Svensson
13.8k ● 8 ● 58 ● 77

5

An attribute is a class that contains some bit of functionality that you can apply to objects in your code. To create one, create a class that inherits from System.Attribute.

As for what they're good for... there are almost limitless uses for them.

http://www.codeproject.com/KB/cs/dotnetattributes.aspx

Share Improve this answer Follow

answered Aug 21, 2008 at 16:00

TheSmurf
15.6k ● 3 ● 41 ● 48

2 "functionality" is the wrong word here; they are metadata, not functionality – Marc Gravell Dec 4, 2014 at 8:19

5

Attributes are like metadata applied to classes, methods or assemblies.

They are good for any number of things (debugger visualization, marking things as obsolete, marking things as serializable, the list is endless).

Creating your own custom ones is easy as pie. Start here:

http://msdn.microsoft.com/en-us/library/sw480ze8(VS.71).aspx

Share Improve this answer Follow

answered Aug 21, 2008 at 16:02

Stu
15.8k ● 4 ● 45 ● 74

5

In the project I'm currently working on, there is a set of UI objects of various flavours and an editor to assembly these objects to create pages for use in the main application, a bit like the form designer in DevStudio.

These objects exist in their own assembly and each object is a class derived from `UserControl` and has a custom attribute. This attribute is defined like this:

```
[AttributeUsage (AttributeTargets::Class)]
public ref class ControlDescriptionAttribute : Attribu
{
public:
  ControlDescriptionAttribute (String ^name, String ^d
    _name (name),
    _description (description)
  {
  }

  property String ^Name
  {
    String ^get () { return _name; }
  }

  property String ^Description
  {
    String ^get () { return _description; }
  }

private:
  String
    ^ _name,
    ^ _description;
};
```

and I apply it to a class like this:

```
[ControlDescription ("Pie Chart", "Displays a pie char
public ref class PieControl sealed : UserControl
{
  // stuff
};
```

which is what the previous posters have said.

To use the attribute, the editor has a `Generic::List <Type>` containing the control types. There is a list box which the user can drag from and drop onto the page to create an instance of the control. To populate the list box, I get the `ControlDescriptionAttribute` for the control and fill out an entry in the list:

```
// done for each control type
array <Object ^>
  // get all the custom attributes
  ^attributes = controltype->GetCustomAttributes (true

Type
  // this is the one we're interested in
  ^attributetype = ECMMainPageDisplay::ControlDescript

// iterate over the custom attributes
for each (Object ^attribute in attributes)
{
  if (attributetype->IsInstanceOfType (attribute))
  {
    ECMMainPageDisplay::ControlDescriptionAttribute
      ^description = safe_cast <ECMMainPageDisplay::Co
^> (attribute);

    // get the name and description and create an entr
    ListViewItem
      ^item = gcnew ListViewItem (description->Name);

    item->Tag = controltype->Name;
    item->SubItems->Add (description->Description);

    mcontrols->Items->Add (item);
    break;
  }
}
```

Note: the above is C++/CLI but it's not difficult to convert to C# (yeah, I know, C++/CLI is an abomination but it's what I have to work with :-( )

You can put attributes on most things and there are whole range of predefined attributes. The editor mentioned above also looks for custom attributes on properties that describe the property and how to edit it.

Once you get the whole idea, you'll wonder how you ever lived without them.

Share Improve this answer Follow

edited Jul 3, 2012 at 13:40
user142162

answered Aug 21, 2008 at 16:54
Skizz
71k ● 10 ● 74 ● 109

4

As said, Attributes are relatively easy to create. The other part of the work is creating code that uses it. In most cases you will use reflection at runtime to alter behavior based on the presence of an attribute or its properties. There are also scenarios where you will inspect attributes on compiled code to do some sort of static analysis. For example, parameters might be marked as non-null and the analysis tool can use this as a hint.

Using the attributes and knowing the appropriate scenarios for their use is the bulk of the work.

Share Improve this answer Follow

answered Aug 21, 2008 at 16:43

denis phillips
12.7k ● 5 ● 34 ● 47

3

Attributes are, essentially, bits of data you want to attach to your **types** (classes, methods, events, enums, etc.)

The idea is that at run time some other type/framework/tool will query **your** type for the information in the attribute and act upon it.

So, for example, Visual Studio can query the attributes on a 3rd party control to figure out which properties of the control should appear in the Properties pane at design time.

Attributes can also be used in Aspect Oriented Programming to inject/manipulate objects at run time based on the attributes that decorate them and add validation, logging, etc. to the objects without affecting the business logic of the object.

Share Improve this answer Follow

answered Aug 21, 2008 at 17:57

urini
33k ● 14 ● 42 ● 37

2

You can use custom attributes as a simple way to define tag values in sub classes without having to write the same code over and over again for each subclass. I came across a nice concise example by John Waters of how to define and use custom attributes in your own code.

There is a tutorial at http://msdn.microsoft.com/en-us/library/aa288454(VS.71).aspx

Share Improve this answer Follow

answered Aug 21, 2008 at 16:02
Chris Miller
4,899 ● 4 ● 34 ● 50

2

To get started creating an attribute, open a C# source file, type `attribute` and hit [TAB]. It will expand to a template for a new attribute.

Share Improve this answer Follow

answered Dec 27, 2008 at 1:55
Jay Bazuzi
46.4k ● 16 ● 115 ● 170

6 How does it answer the question? it should be a comment, not an answer. – gdoron Nov 18, 2014 at 7:40

1

Attributes are also commonly used for Aspect Oriented Programming. For an example of this check out the PostSharp project.

Share Improve this answer Follow

answered Oct 21, 2009 at 20:21
Josh G
14.3k ● 7 ● 64 ● 74

Highly active question. Earn 10 reputation (not counting the association bonus) in order to answer this question. The reputation requirement helps protect this question from spam and non-answer activity.