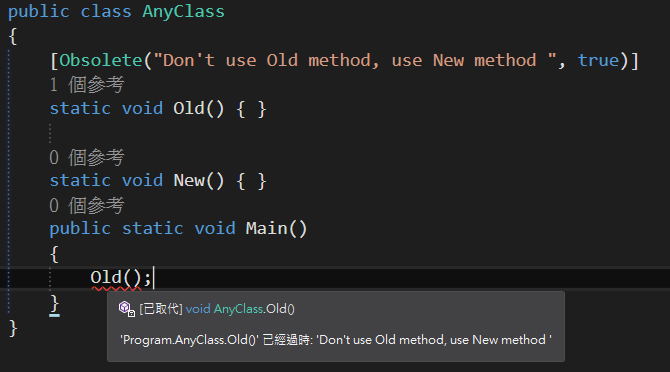
Attributes是一种新的描述信息，我们既可以使用attributes来定义设计期信息（例如帮助文件，文档的URL），还可以用attributes定义运行时信息（例如，使xml中的元素与类的成员字段关联起来）。我们也可以用attributes来创建一个“自描述”的组件。在这篇指南中我们将明白怎么创建属性并将其绑定至各种语言元素上，另外我们怎么样在运行时环境下获取到attributes的一些信息。

定义：

MSDN中做如下定义：An attribute is a piece of additional declarative information that is specified of a declaration.

使用预定义attributes:

在C#中已有一小组预定义的attributes,在我们学习怎么创建自定义attributes前,先来了解下在我们的代码中使用哪些预定义的attributes.



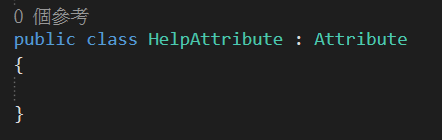
仔細看下该实例，在该实例中我们用到了“Obsolete”attribute，它标记了一个不该再被使用的语言元素（译者注：这里的元素为方法），该属性的第一个参数是string类型，它解释为什么该元素被荒弃，以及我们该使用什么元素来代替它。实际中，我们可以书写任何其他文本来替代这段文本。第二个参数是告诉编译器把依然使用这被标识的元素视为一种错误，这就意味这编译器会因此而产生一个警告。

当我们试图编译上面的程序就会看到该报错。

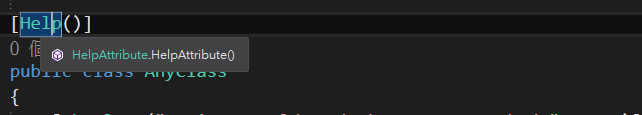
开发自定义Attributes：

现在我们即将了解怎么开发自定义的Attributes。这儿有个小小处方，有它我们就可以学会创建自定义的Attributes。

在C#中，我们的Attributes类都派生于System.Attribute类（A class that derives from the abstract class System.Attribute, whether directly or indirectly, is an attribute class. The declaration of an attribute class defines a new kind of attribute that can ba placed on a declaration.）

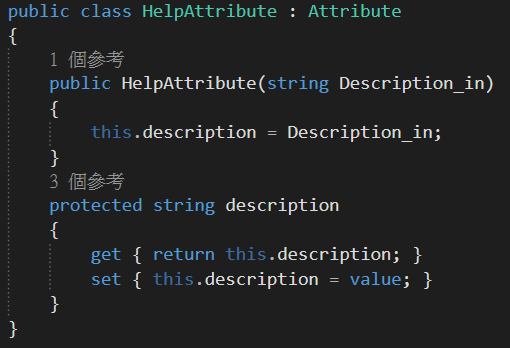


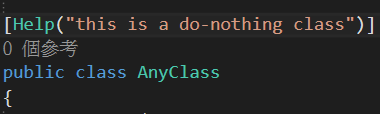
由上可知，我们已经创建了一个自定义attribute。现在就可以去装饰我们的类了。



注意：按惯例我们是用“Attribute”作为attribute类名的后缀，然而，当我们把attribute绑定到某语言元素时，是不包含Attribute后缀的。编译器首先在System.Attribute的继承类中查找该attribute,如果没有找到,编译器会把Attribute追加到该Attribute的名字后面，然后查找它。

但是迄今为止，该attribute没有任何用处。为了是它有点用处，让我们在它里面加点东西。





在上面的例子中，我们在attribute类中添加了一个属性，在最后一节中，我们将在运行时查询该属性。

定义或控制自定义Attribute的用法：

AttributeUsage类是一另预定义类（译者注：attribute类本身用这个attribute System.AttributeUsage来标记）,它将帮助我们控制我们自定义attribute的用法,这就是,我们能为自定义的attribute类定义attributes.

它描述了一个自定义attribute类能被怎么样使用.

AttributeUsage提供三个属性.我们将他们放置到我们的自定义attribute类上,第一个特性是:

ValidOn

通过这个属性,我们能指定我们的自定义attribute可以放置在哪些语言元素上.这组我们能把自定义attribute类放置其上的语言元素 被放在枚举器AttributeTarget中。我们可以使用bitwise（译者注：这个词不知道怎么翻译好，但它的意思是可以这么用：[AttributeUsage( (AttributeTarget)4, AllowMultiple = false, Inherited = false)], 4代表就是class元素,其他诸如1代表assembly, 16383代表all等）或者”.”操作符绑定几个AttributeTargets值.(译者注:默认值为AttributeTarget.All )

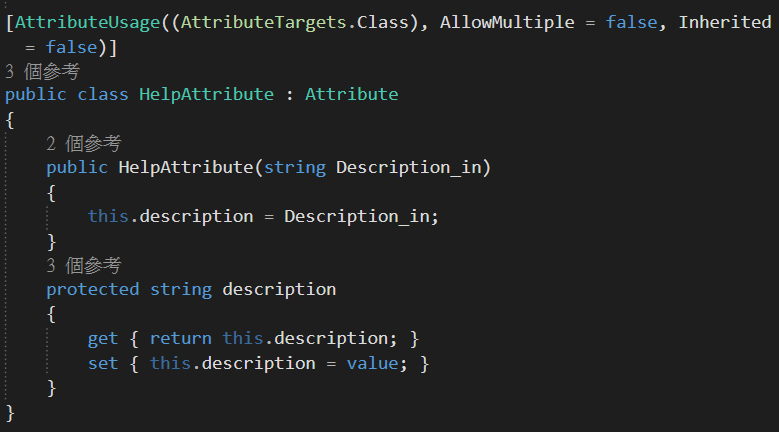
AllowMultiple

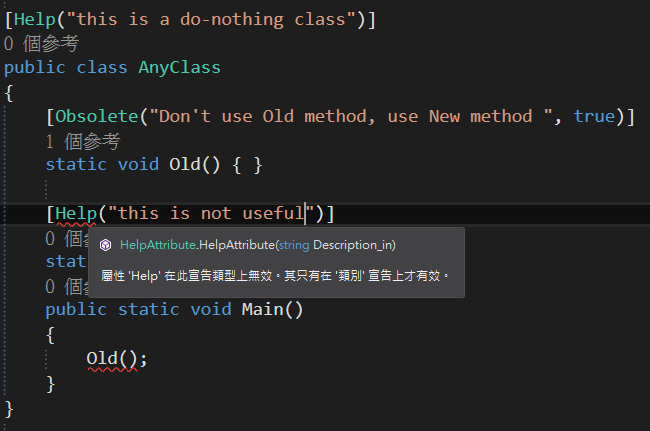
该属性标识我们的自定义attribute能在同一语言元素上使用多次.(该属性为bool类型,默认值为false,意思就是该自定义attribute在同一语言元素上只能使用一次.)

Inherited

我们可以使用该属性来控制我们的自定义attribute类的继承规则.该属性标识我们的自定义attribute是否可以由派生类继承.

下面我们就来实际运用一下.



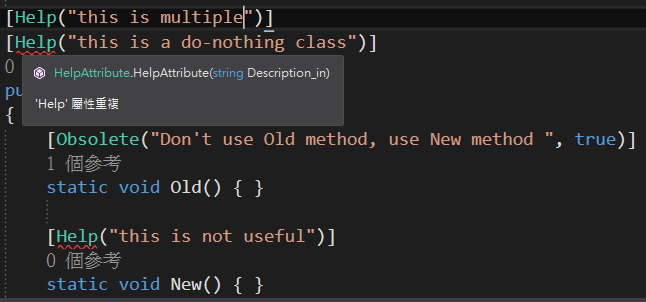


首先我们注意AttributeTarget.Class,它规定这个help attribute只能放置在语言元素class之上.这就意味着,会出现如上图所示的错误.

我们可以使用AttributeTarget.All来允许Help Attribute可以放置在任何预定义的语言元素上:



现在考虑下AllowMultiple = false，该参数规定attribute不能在同一语言元素上放置多次。

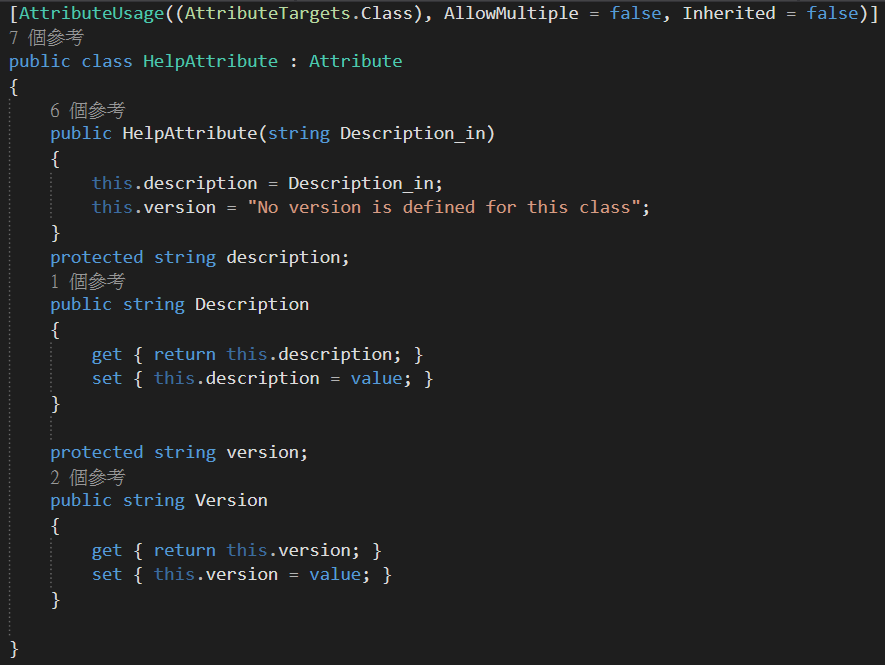


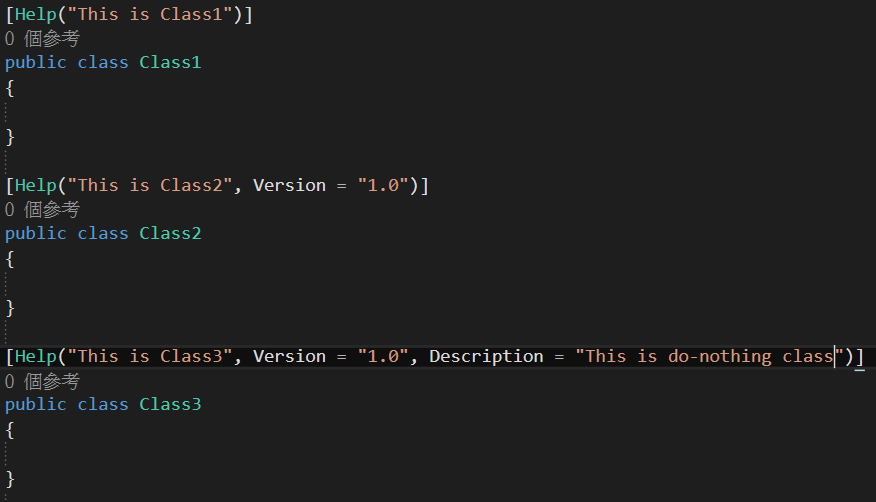
现在我们该讨论下最后的属性了，“Inherited”，指出当把该attribute放置与一个基类之后，是否派生类也继承了该attribute。

如果绑定至某个attribute类的“Inherited”被设为true，那么该attribute就会被继承，反之如果设为false或没有定义，那么该attribute就不会被继承。

可选参数和命名参数：

可选参数是attribute类构造函数的参数。它是强制性的，必须在每次attribute绑定至某语言元素时提供一个值。而另一方面，命名参数倒是真正的可选参数，不是在attribute构造函数的参数。





当我们在class1中查询help attribute的属性，我们将得到：

Help.Description:This is Class1

Help.Version:No Version is defined for this class

因为我们没有为Version这个属性定义任何值,所以在構造函數中设定的值被我们查询出来了。

现在查询class2 的结果是：

Help.Description:This is class2

Help.Version:1.0