# C# 特性（Attribute）

**特性（Attribute）**是用于在运行时传递程序中各种元素（比如类、方法、结构、枚举、组件等）的行为信息的声明性标签。您可以通过使用特性向程序添加声明性信息。

## 规定特性（Attribute）

规定特性（Attribute）的语法如下：

[attribute(positional\_parameters, name\_parameter = value, ...)]

element

## 预定义特性（Attribute）

.Net 框架提供了三种预定义特性：

* AttributeUsage
* Conditional
* Obsolete

### **AttributeUsage**

预定义特性 **AttributeUsage** 描述了如何使用一个自定义特性类。它规定了特性可应用到的项目的类型。

规定该特性的语法如下：

[AttributeUsage(

validon,

AllowMultiple=allowmultiple,

Inherited=inherited)]

其中：

* 参数 validon 规定特性可被放置的语言元素。它是枚举器 *AttributeTargets* 的值的组合。默认值是 *AttributeTargets.All*。
* 参数 *allowmultiple*（可选的）为该特性的 *AllowMultiple* 属性（property）提供一个布尔值。如果为 true，则该特性是多用的。默认值是 false（单用的）。
* 参数 *inherited*（可选的）为该特性的 *Inherited* 属性（property）提供一个布尔值。如果为 true，则该特性可被派生类继承。默认值是 false（不被继承）。

例如：

[AttributeUsage(AttributeTargets.Class |AttributeTargets.Constructor |AttributeTargets.Field |AttributeTargets.Method |AttributeTargets.Property, AllowMultiple = true)]

### **Conditional**

这个预定义特性标记了一个条件方法，其执行依赖于它顶的预处理标识符。

它会引起方法调用的条件编译，取决于指定的值，比如 **Debug** 或 **Trace**。例如，当调试代码时显示变量的值。

规定该特性的语法如下：

[Conditional(conditionalSymbol)]

### **Obsolete**

这个预定义特性标记了不应被使用的程序实体。它可以让您通知编译器丢弃某个特定的目标元素。

规定该特性的语法如下：

[Obsolete(

message)][Obsolete(

message,

iserror)]

其中：

* 参数 *message*，是一个字符串，描述项目为什么过时的原因以及该替代使用什么。
* 参数 *iserror*，是一个布尔值。如果该值为 true，编译器应把该项目的使用当作一个错误。默认值是 false（编译器生成一个警告）。

## 创建自定义特性（Attribute）

.Net 框架允许创建自定义特性，用于存储声明性的信息，且可在运行时被检索。该信息根据设计标准和应用程序需要，可与任何目标元素相关。

创建并使用自定义特性包含四个步骤：

* 声明自定义特性
* 构建自定义特性
* 在目标程序元素上应用自定义特性
* 通过反射访问特性

最后一个步骤包含编写一个简单的程序来读取元数据以便查找各种符号。元数据是用于描述其他数据的数据和信息。

例：

// 一个自定义特性 BugFix 被赋给类及其成员

[AttributeUsage(AttributeTargets.Class |AttributeTargets.Constructor |AttributeTargets.Field |AttributeTargets.Method |AttributeTargets.Property,AllowMultiple = true)]

public class DeBugInfo : System.Attribute{

private int bugNo;

private string developer;

private string lastReview;

public string message;

public DeBugInfo(int bg, string dev, string d)

{

this.bugNo = bg;

this.developer = dev;

this.lastReview = d;

}

public int BugNo

{

get

{

return bugNo;

}

}

public string Developer

{

get

{

return developer;

}

}

public string LastReview

{

get

{

return lastReview;

}

}

public string Message

{

get

{

return message;

}

set

{

message = value;

}

}}

### **应用自定义特性**

通过把特性放置在紧接着它的目标之前，来应用该特性：

[DeBugInfo(45, "Zara Ali", "12/8/2012", Message = "Return type mismatch")][DeBugInfo(49, "Nuha Ali", "10/10/2012", Message = "Unused variable")]class Rectangle