**Part 52 - C# Tutorial - Attributes in C#**

**In this part we will learn**  
1. The purpose of attributes  
2. Using an attribute  
3. Customizing attribute using parameters  
  
  
**Purpose:**Attributes allow you to add declarative information to your programs. This information can then be queried at runtime using reflection.

There are several Pre-defined Attributes provided by .NET. It is also possible to create your own Custom Attributes. Creating custom attributes is beyond the scope of this article.  
  
  
**A few pre-defined attributes with in the .NET framework.**  
Obsolete - Marks types and type members outdated  
WebMethod - To expose a method as an XML Web service method  
Serializable - Indicates that a class can be serialized  
  
  
**Example program using pre defined Obsolete attribute:**  
Obsolete attribute can be used with types or type members that are **obsolete (Outdated).** If a developer uses a type or a type member that is decorated with obsolete attribute, the compiler issues a warning or an error depending on how the attribute is configured.  
  
  
In this sample program, **Add(int FirstNumber, int SecondNumber)** method is decorated with [Obsolete] attribute. If you compile this program, in the output window you will see a warning message (Compile complete -- 0 errors, 1 warnings). Also, visual studio, shows a green squiggly line under the  **Add(int FirstNumber, int SecondNumber)** method. If you hover the mouse over the squiggly line, you should see the warning message.  
  
  
**Note**: If you don't see the warning message (Compile complete -- 0 errors, 1 warnings), rebuild the soultion.  
  
  
using System;  
using System.Collections.Generic;  
public class MainClass  
{  
    private static void Main()  
    {  
        Calculator.Add(10, 15);  
    }  
}  
  
  
public class Calculator  
{  
    [Obsolete]  
    public static int Add(int FirstNumber, int SecondNumber)  
    {  
        return FirstNumber + SecondNumber;  
    }  
    public static int Add(List<int> Numbers)  
    {  
        int Sum = 0;  
        foreach (int Number in Numbers)  
        {  
            Sum = Sum + Number;  
        }  
        return Sum;  
    }  
}  
  
  
The warning message says **'Calculator.Add(int, int)' is obsolete**. However, this message is not completely useful, because it says**'Calculator.Add(int, int)' is obsolete**, but not tell us which other method **should we be using instead**. So this is when we can customize, the warning message**using attribute parameters**.  
  
  
The intention of the developer of Calculator class is that, he wanted us to use **Add(List<int> Numbers)**, instead of int **Add(int FirstNumber, int SecondNumber)**. To communicate this message we can customize the warning message using attribute parameters as shown below. With this customization we are not only communicating that **Add(int FirstNumber, int SecondNumber)** method is obsolete, we are also telling to use the alternative method that is available.

[Obsolete("Use Add(List<int> Numbers) instead")]  
public static int Add(int FirstNumber, int SecondNumber)  
  
  
If you want to generate a compiler error instead of warning, pass true for the bool error parameter of the Obsolete attribute as shown below. Now, we can't even compile the program.  
[Obsolete("Use Add(List<int> Numbers) instead", true)]  
public static int Add(int FirstNumber, int SecondNumber)  
  
  
Finally, If you right click on Obsolete attribute and select Go To Definition, you will see that, an attribute is nothing but a class that inherits from System.Attribute base class.