C# 特性（Attribute）
C# 属性（Property）

C# 反射（Reflection）

反射指程序可以访问、检测和修改它本身状态或行为的一种能力。

程序集包含模块，而模块包含类型，类型又包含成员。反射则提供了封装程序集、模块和类型的对象。

您可以使用反射动态地创建类型的实例，将类型绑定到现有对象，或从现有对象中获取类型。然后，可以调用类型的方法或访问其字段和属性。

优缺点

优点：

- 1、反射提高了程序的灵活性和扩展性。
- 2、降低耦合性，提高自适应能力。
- 3、它允许程序创建和控制任何类的对象，无需提前硬编码目标类。

缺点：

- 1、性能问题：使用反射基本上是一种解释操作，用于字段和方法接入时要远慢于直接代码。因此反射机制主要应用在对灵活性和拓展性要求很高的系统框架上，普通程序不建议使用。
- 2、使用反射会模糊程序内部逻辑；程序员希望在源代码中看到程序的逻辑，反射却绕过了源代码的技术，因而会带来维护的问题，反射代码比相应的直接代码更复杂。

反射（Reflection）的用途

反射（Reflection）有下列用途：

- 它允许在运行时查看特性（attribute）信息。
- 它允许审查集合中的各种类型，以及实例化这些类型。
- 它允许延迟绑定的方法和属性（property）。
- 它允许在运行时创建新类型，然后使用这些类型执行一些任务。

查看元数据

我们已经在上面的章节中提到过，使用反射（Reflection）可以查看特性（attribute）信息。

System.Reflection 类的 **MemberInfo** 对象需要被初始化，用于发现与类相关的特性（attribute）。为了做到这点，您可以定义目标类的一个对象，如下：

```
System.Reflection.MemberInfo info = typeof(MyClass);
```

下面的程序演示了这点：

实例

```
using System;

[AttributeUsage(AttributeTargets.All)]
public class HelpAttribute : System.Attribute
{
   public readonly string Url;

   public string Topic  // Topic 是一个命名（named）参数
   {
      get
      {
         return topic;
      }
      set
      {

         topic = value;
      }
   }

   public HelpAttribute(string url)  // url 是一个定位（positional）参数
   {
      this.Url = url;
   }

   private string topic;
}
[HelpAttribute("Information on the class MyClass")]
class MyClass
{
}

namespace AttributeAppl
{
   class Program
   {
      static void Main(string[] args)
      {
         System.Reflection.MemberInfo info = typeof(MyClass);
         object[] attributes = info.GetCustomAttributes(true);
         for (int i = 0; i < attributes.Length; i++)
         {
            System.Console.WriteLine(attributes[i]);
         }
         Console.ReadKey();

      }
   }
}
```

当上面的代码被编译和执行时，它会显示附加到类 *MyClass* 上的自定义特性：

```
 HelpAttribute
```

实例

在本实例中，我们将使用在上一章中创建的 *DeBugInfo* 特性，并使用反射（Reflection）来读取 *Rectangle* 类中的元数据。

实例

```
using System;
using System.Reflection;
namespace BugFixApplication
{
   // 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;
         }
      }
   }
   [DeBugInfo(45, "Zara Ali", "12/8/2012",
        Message = "Return type mismatch")]
   [DeBugInfo(49, "Nuha Ali", "10/10/2012",
        Message = "Unused variable")]
   class Rectangle
   {
      // 成员变量
      protected double length;
      protected double width;
      public Rectangle(double l, double w)
      {
         length = l;
         width = w;
      }
      [DeBugInfo(55, "Zara Ali", "19/10/2012",
           Message = "Return type mismatch")]
      public double GetArea()
      {
         return length * width;
      }
      [DeBugInfo(56, "Zara Ali", "19/10/2012")]
      public void Display()
      {
         Console.WriteLine("Length: {0}", length);
         Console.WriteLine("Width: {0}", width);
         Console.WriteLine("Area: {0}", GetArea());
      }
   }//end class Rectangle

   class ExecuteRectangle
   {
      static void Main(string[] args)
      {
         Rectangle r = new Rectangle(4.5, 7.5);
         r.Display();
         Type type = typeof(Rectangle);
         // 遍历 Rectangle 类的特性
         foreach (Object attributes in type.GetCustomAttributes(false))
         {
            DeBugInfo dbi = (DeBugInfo)attributes;
            if (null != dbi)
            {
```

```
                Console.WriteLine("Bug no: {0}", dbi.BugNo);
                Console.WriteLine("Developer: {0}", dbi.Developer);
                Console.WriteLine("Last Reviewed: {0}",
                                  dbi.LastReview);
                Console.WriteLine("Remarks: {0}", dbi.Message);
            }
         }

         // 遍历方法特性
         foreach (MethodInfo m in type.GetMethods())
         {
            foreach (Attribute a in m.GetCustomAttributes(true))
            {
               DeBugInfo dbi = (DeBugInfo)a;
               if (null != dbi)
               {
                  Console.WriteLine("Bug no: {0}, for Method: {1}",
                                    dbi.BugNo, m.Name);
                  Console.WriteLine("Developer: {0}", dbi.Developer);
                  Console.WriteLine("Last Reviewed: {0}",
                                    dbi.LastReview);
                  Console.WriteLine("Remarks: {0}", dbi.Message);
               }
            }
         }
         Console.ReadLine();
      }
   }
}
```

当上面的代码被编译和执行时，它会产生下列结果：

```
Length: 4.5
Width: 7.5
Area: 33.75
Bug No: 49
Developer: Nuha Ali
Last Reviewed: 10/10/2012
Remarks: Unused variable
Bug No: 45
Developer: Zara Ali
Last Reviewed: 12/8/2012
Remarks: Return type mismatch
Bug No: 55, for Method: GetArea
Developer: Zara Ali
Last Reviewed: 19/10/2012
Remarks: Return type mismatch
Bug No: 56, for Method: Display
Developer: Zara Ali
Last Reviewed: 19/10/2012
Remarks:
```

C# 特性（Attribute）

C# 属性（Property）

点我分享笔记