[**net中String是引用类型还是值类型**](http://www.cnblogs.com/yank/archive/2011/10/24/2204145.html)

Posted on 2011-10-24 14:03 [停留的风](http://www.cnblogs.com/yank/) 阅读(12972) 评论(20) [编辑](https://i.cnblogs.com/EditPosts.aspx?postid=2204145) [收藏](http://www.cnblogs.com/yank/archive/2011/10/24/2204145.html)

关于String为值类型还是引用类型的讨论一直没有平息，最近一直在研究性能方面的问题，今天再次将此问题进行一次明确。希望能给大家带来点帮助。 如果有错误请指出。

**来看下面例子：**

[复制代码](javascript:void(0);)

//值类型  
 int a = 1;  
 int b = a;  
 a = 2;  
 Console.WriteLine("a is {0},b is {1}", a, b);  
  
 //字符串  
 string str1 = "ab";  
 string str2 = str1;  
 str1 = "abc";  
 Console.WriteLine("str1 is {0},str2 is {1}", str1, str2);  
 Console.Read();

[复制代码](javascript:void(0);)

根据上面的例子：你觉得输出结果应该是什么？

**输出结果：**

//结果：  
 //a is 2,b is 1  
 //str1 is abc,str2 is ab

str2依然是ab,并没有随str1的改变而改变。

如果string是引用类型，按理Str1和Str指针都指向同一内存地址，如果Str的内容发生改变，Str1应该也会相应变化。

此例子，看着string更像是值类型。

但是MSDN却说String是引用类型，

**引用类型包括：**   
**String**

所有数组，即使其元素是值类型

类类型，如 Form

委托

可参考：<http://msdn.microsoft.com/zh-cn/library/t63sy5hs(VS.80).aspx>

**查看具体引用是否相同**

如果Net能够查看内存地址就容易了，但不允许，只能通过间接方法来实现，看下面：

[复制代码](javascript:void(0);)

static void TestRefAddress()  
 {  
 String str1 = "abc";  
 String str2 = "abc";  
 int a = 1;  
 int b = 1;  
 StringBuilder strb1 = new StringBuilder("abc");  
 StringBuilder strb2 = new StringBuilder("abc");  
 Console.WriteLine("Reference equal for string: " + Object.ReferenceEquals(str1, str2)); //结果true  
 Console.WriteLine("Reference equal for int: " + Object.ReferenceEquals(a, b)); //结果false  
 Console.WriteLine("Reference equal for StringBuilder: " + Object.ReferenceEquals(strb1, strb2)); //结果false  
 Console.WriteLine("Value equal for string: " + str1.Equals(str2)); //结果true，类似于值类型  
 Console.Read();  
 }

[复制代码](javascript:void(0);)

结果为何出现如此情况，分析如下：

[复制代码](javascript:void(0);)

Console.WriteLine("Reference equal for string: " + Object.ReferenceEquals(str1, str2)); //结果true，不同对象，但引用地址相同  
 Console.WriteLine("Reference equal for int: " + Object.ReferenceEquals(a, b)); //结果false,值类型装箱操作造成  
 Console.WriteLine("Reference equal for StringBuilder: " + Object.ReferenceEquals(strb1, strb2)); //结果false，不同对象，引用地址不同  
 Console.WriteLine("Value equal for string: " + str1.Equals(str2)); //结果true，类似于值类型

[复制代码](javascript:void(0);)

由第一条结果，可以判定不同的String的，相同的值，其引用地址相同，再由第四条结果，str1.Equals(str2),两者结合，可得出结论，两个String，如果赋值为同一个值，在内存中只有一个字符串存在，两个引用的地址相同。由此引出String的不变性。

**String的不变性**

string最为显著的一个特点就是它具有恒定不变性：我们一旦创建了一个string，在managed heap 上为他分配了一块连续的内存空间，我们将不能以任何方式对这个string进行修改使之变长、变短、改变格式。所有对这个string进行各项操作（比如调用ToUpper获得大写格式的string）而返回的string，实际上另一个重新创建的string，其本身并不会产生任何变化。   
string   对象称为不可变的（只读），因为一旦创建了该对象，就不能修改该对象的值。有的时候看来似乎修改了，实际是string经过了特殊处理，每次改变值时都会建立一个新的string对象，变量会指向这个新的对象，而原来的还是指向原来的对象，所以不会改变。这也是string效率低下的原因。

String的不变，并非说string不能改变，而是其值不能改变。

在例子中str1="ab",这时在内存中就将“ab”存下来，如果再创建字符串对象，其值也等于“ab”，str2="ab",则并非再重新分配内存空间，而是将之前保存的“ab”的地址赋给str2的引用，这就能印证例子2中的结果。而当str1="abc"其值发生改变时，这时检查内存，发现不存在此字符串，则重新分配内存空间，存储“abc”，并将其地址赋给str1，而str2依然指向“ab”的地址。可以印证例子1中的结果。

**结论：**

**String是引用类型，只是编译器对其做了特殊处理。**

关于讨论：<http://social.msdn.microsoft.com/Forums/zh-CN/visualcshartzhchs/thread/ce580186-86d9-45f7-b5ff-20302caf1324>