

# Re: String Reference Type

---

*Source:*

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.framework.aspnet/2008-02/msg00943.html>

---

- *From:* "Lars" <[jon.doe@xxxxxxxxxxxxxxxxxxxxxxxxxxxxx](mailto:jon.doe@xxxxxxxxxxxxxxxxxxxxxxxxxxxxx)>
  - *Date:* Tue, 12 Feb 2008 15:17:35 GMT
- 

Hi

So to the original point; is a string a class or a structure? Ans: It's a class. Therefore its always allocated in the heap and a variable of string type is always a reference to this string object allocated on the heap.

```
Dim x As String = "Hello World"  
Dim y As String  
y = x
```

As with all classes in this case y and x both reference the same String object.

Does the String class implements operators to take care of this.

Can you write you own class for example Complex that defines the assignment operator and comparement operators. Si that if you wrote  $Z1=Z2$  copies the values  $Z2.x$  and  $Z2.y$  to  $Z1$  leavind  $Z2$  unchained. Do you have to create a new object of  $Z$  and return for this?

What makes this less obvious (as Mykola alluded to) is the immutable nature of a String. Once a string object has been allocated and initialised its content cannot be changed, not even changes that don't involve changing the length. This results in code which makes strings look like they are behaving as structures because no where do see an operation on one reference that is visable through another reference.

## Re: String Reference Type

A similar type to a string is a character array but a character array is mutable :-

```
Dim z As Char() = New Char(2) {"A"c, "B"c, "C"c}
Dim x As Integer()
```

```
x = z
```

```
x(1) = "X"c
```

```
Console.WriteLine(z(1))
```

Note that both z and x reference the same Array object. A modification made to the object via one reference is visible using the other. This sort of operation just isn't available on the string object. Its the lack of this sort of thing happening in code to a string which gives the illusion of it being a structure.

However Mykola is in error stating that you can't pass strings as ByRef. You can (in both VB and C#).

```
Public Sub DoSomething(ByRef rs As String)
    rs = "New content"
End Sub
```

```
Dim s as String = "Hello World"
```

```
DoSomething(s)
```

```
Console.WriteLine(s)
```

This confusion arises from the overloaded use of the word Reference (which is why I prefer to think in terms of Structures or Classes). When you think about it all variables are ultimately references to somewhere in memory where a value is stored. For example:-

```
Sub Thing()
    Dim i as Integer
    Dim o as Object
```

The symbol i resolves to a offset pointer from the a fixed point on the stack frame where 4 bytes hold the actual value of the integer. The symbol o resolves to an offset pointer (likely the same as for i + 4) from the same fixed point on the stack frame which will hold a reference to some reference to an object stored on the heap.

## Re: String Reference Type

Passing ByRef has nothing to do with whether the value is a Structure or a Class but everything to do with where relative to a fixed point on a stack frame to find the variable value (be that the actual structure content or the reference to a heap held object). Since at the time of call the size of the current frame is known its relatively simple to supply an offset from a fixed point in the new frame back into the previous.

I Think I got it

```
public void swap( ref String S1, ref String S2 )
{
String tmp1=S1;
S2=S1;
S1=tmp;
}
```

What this function does is swapping the reference values. Compare it to the following C++ and Delphi code

```
public void swap( String&* S1, String&* S2 )
{
String* tmp1=S1;
S2=S1;
S1=tmp;
}
```

```
// Delphi code
procedure swap( var String S1, var String S2 )
Strng tmp
begin
tmp = S2;
S1=S2;
S2=tmp;
end;
```

—  
Anthony Jones – MVP ASP/ASP.NET

Note 1:–

A couple of other reasons (apart from my primary reason that the word reference is already over used) I prefer to think in terms of Structure or Class instead of ValueType or ReferenceType are:–

## Re: String Reference Type

When you take a basic course in Datastructures and Algorithms you learn that a REFERENCE is a reference that refers to another object. Whether it's implemented by pointers, memory on the heap or any thing else doesn't matter. A Value is value treated as a value (object). The OBJECT if you prefer. Every Reference has to refer to an OBJECT. A teacher I had at the University (Basic Datatypes and Algorithms) told us to look at References is a non language dependant matter as below. 14 years later I still recall it.

References (pointers for old C programmers, instances of classes in C#)

```
{Reference} ----> {Object ( 1,2,3,4,5,6,67,78,)}
```

or

```
{Reference} ----> {Object ( 1,2,3,4,5,6,67,78,),Object ( 1,2,3,4,5,6,67,78),Object("EHROIERO") }
```

Objects

```
{Object ( 1,2,3,4,5,6,67,78,)}
```

```
{Object ( 1,2,3,4,5,6,78,)}
```

```
{Object ("DJÖLFD","DODHFO")}
```

1). Its possible to have a reference to a structure. I.e. when its boxed for example referenced via variable typed as Object.

2.) The MustInherit (abstract) type System.ValueType from which all 'ValueTypes' inherit is a class not a structure. How can a structure inherit from a Class? Ans: it can't. Only when Boxed does a Structure inherit from ValueType and at the point its a reference to copy on the heap.

For those of you who have hard understanding heap and stack. Look at the example above. A picture says more than a thousand words as they say. If the object referred to is on the heap, file, on another server doesn't matter. Think abstract and you get it.

Its all quite confusing.

Not really wonce you know that all instances of classes are references its no problem. What I did was to not only mix C++ and C# I also have a lot of Delphi code in memory. In the matter of how to implement instances of classes Delphi is more like C# than C++ is. In a true Obecj Orienter Language as Smalltalk every thing is a reference. As I understand C# is move

## Re: String Reference Type

towards that direction. But C# is not near an truly Object Oriented language. Although it you can use OOP techniques. The build in classes oin C# doesn't support encapsulation fully. For example

```
// Does LabekX know that you change the value of the Text when I do this?  
Does C# allow you to define properties like Delphi.  
LabelX.Text = "Some text";
```

This is important to think of when you wrote your own classes. For every member (variable) in the class write an accessor and modifier method. In general Set Get methods. It makes life easy when you inherit the class in another class.

The terms ValueType and ReferenceType apply globally to an Object (which is always a reference) and indicates whether the type is 'normally' stored directly in the variable declared of that type or whether the variable of that type always holds a reference to the value held on the heap.