

GURUS: VB .NET alternatives to C# type conversion operators and operator overloading?

Source: <http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.framework/2004-04/0433.html>

From: Krunoslav (*ktrs66_at_hotmail.com*)

Date: 04/06/04

Date: Tue, 6 Apr 2004 21:46:39 +0200

Hi ,

I have implemented a struct in C# which acts as a sort of a variant type, i.e its constructors accept numeric types, dates and strings and the struct itself is convertible to those types but no other types and can perform arithmetic operations with those types too.

But arithmetic operations are not as important to me as the seamless conversion to and fro.

For the conversion purpose I had to implement static conversion operators, implicit and explicit.

I also implemented IConvertible interface in the struct.

NOW THE PROBLEM.

After testing the class in VB .NET I realized that those static converters as well as any static operators are unavailable. In C# everything works fine, but in VB I could not say something like

```
MyStruct = 100
```

```
or MyStruct = CType(100, MyStruct)
```

even though MyStruct has a constructor with an int value (and others) as well as the conversion operators.

Examination of my objects with ildasm proves that all of the conversion operators are there.

As I say in C# everything works perfectly as it should.

So I considered implementing a custom TypeConverter for my struct to see if this will help.

Then I performed the test in VB .NET to see what I can do with just the VB .NET itself not C#.

I wrote the following in VB as a test, just the constructors and the TypeConverter:

I implemented a type converter SmartTypeConverter for the structure too.

namespace test

```
<TypeConverter(GetType(SmartTypeConverter))> _
Public Structure VBSmartVar
    Private m_object As Object

    Public Sub New(ByVal v As String)
        m_object = New Object()
        m_object = v
    End Sub

    Public Sub New(ByVal v As Integer)
        m_object = New Integer()
        m_object = v
    End Sub

    Public Sub New(ByVal v As Double)
        m_object = New Double()
        m_object = v
    End Sub

    Public Overrides Function ToString() As String
        Return CStr(m_object)
    End Function

    Public Overloads Function GetTypeCode() As TypeCode
        If TypeOf m_object Is String Then
            Return CStr(m_object).GetTypeCode()
        End If
        If TypeOf m_object Is Integer Then
            Return CInt(m_object).GetTypeCode()
        End If
        If TypeOf m_object Is Double Then
            Return CDbI(m_object).GetTypeCode()
        End If
    End Function

    Public Overloads Function GetHashCode() As Integer
        If TypeOf m_object Is String Then
            Return CStr(m_object).GetHashCode()
        End If
        If TypeOf m_object Is Integer Then
            Return CInt(m_object).GetHashCode()
        End If
        If TypeOf m_object Is Double Then
            Return CDbI(m_object).GetHashCode()
        End If
    End Function
```

```
End Structure  
End Namespace
```

The test proved unsuccessful, I still can not say either
MyStruct = 100
or CType(100, MyStruct).

What I would like is something like this
MyStruct1 = 100
MyStruct2 = 100.45

```
MyInteger = CInt(MyStruct)  
MyDouble = CDbI(MyStruct)
```

etc

I do not include here a type converter I used for brevity, but it implements
CanConvertTo, CanConvertFrom, ConvertTo, ConvertFrom and IsValid.

Is it POSSIBLE to do something like this in VB like it is in C# and if so
how please?

This issue may severely affect my entire system design construct from ground
up.

As you can see, I am more comfortable with C# due to a C++ background though
I used to program simple apps in VB 6. So if anyone knows a simple way to do
this in both C# and
VB please let me know.
Thanks to anyone that gives me any meaningful tips.

my email: krtrs66@hotmail.com.