

Re: Another C# marshaling question

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.csharp/2005-11/msg03145.html>

- *From:* "Laurent" <mournblade@xxxxxxx>
 - *Date:* Tue, 29 Nov 2005 14:01:00 +0100
-

Hi Nicholas,

Thank you for your answer, it works !!! I can access my function and the returned value is as expected.

But I still have a little problem... this part of code doesn't work "as it":

```
for (int index = 0; index < outNumber; index++)
{
    // Get the pointer of the location in memory.
    IntPtr elementPointer = new IntPtr(outStructure.ToInt64() + (index *
    Marshal.SizeOf(typeof(FF_Struct))));

    // Now marshal that value.
    outStructs[index] = Marshal.PtrToStructure(elementPointer,
    typeof(FF_Struct));
}
```

In fact, the first line is OK and the others have not been read correctly. When I debug the code, `Marshal.SizeOf(typeof(FF_Struct)) = 32`. I wrote the part of memory I have read into a binary file and examined it with Hex Editor. The size of my structure is 29. When I manually put this value into your code, it works perfectly.

I think something is wrong with my `FF_Struct` class definition, but I don't know why. I'm sure the size of the char arrays is OK. Perhaps the problem comes from the '\0' terminated string ??? Do you have any clue ?

Thanks anyway for your help, you taught me a lot of stuffs today ! :)

Laurent

Re: Another C# marshaling question

"Nicholas Paldino [.NET/C# MVP]" <mvp@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx> a écrit dans le message de news: O7Gnt6D9FHA.4076@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

> Laurent,

>

> You are going to have to manually marshal the structure. Your

> structure declaration needs to be changed:

>

> [StructLayout(LayoutKind.Sequential, CharSet=CharSet.Ansi)]

> public class FF_Struct

> {

> public int index;

> [MarshalAs(UnmanagedType.ByValTStr, SizeConst = FF_ID_LEN)]

> public string numeroCommande;

> [MarshalAs(UnmanagedType.ByValTStr, SizeConst = FF_DATETIME_LEN)]

> public string dateCommande;

> }

>

> Declaring as a class will tell the CLR that it can re-arrange the order

> of the fields as it wishes, which is not what you want in this case.

>

> You will also have to change your function declaration to:

>

> [DllImport("FF_myDLL.dll")]

> public static extern int FF_Function(

> IntPtr inInstance,

> [MarshalAs(UnmanagedType.LPStr)]

> string date,

> ref IntPtr outStructure,

> ref int outNumber

>);

>

> Then, what you have to do on return is marshal the array, like so:

>

> // Allocate the array.

> FF_Struct[] outStructs = new FF_Struct[outNumber];

>

> // Cycle through unmanaged memory, and marshal.

> for (int index = 0; index < outNumber; index++)

> {

> // Get the pointer of the location in memory.

> IntPtr elementPointer = new IntPtr(outStructure.ToInt64() + (index *

> Marshal.SizeOf(typeof(FF_Struct))));

>

> // Now marshal that value.

> outStructs[index] = Marshal.PtrToStructure(elementPointer,

> typeof(FF_Struct));

> }

>

> The only problem here is that you have to figure out how to release the

> memory that was allocated for the structure. The function is allocating

Re: Another C# marshaling question

> it, but you aren't handling that in this situation.
>
> Hope this helps.
>
>
> --
> - Nicholas Paldino [.NET/C# MVP]
> - mvp@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
>
>
> "Laurent" <mournblade@xxxxxxx> wrote in message
> news:u210%23zD9FHA.1484@xxxxxxxxxxxxxxxxxxxxxxxxxxxxx
>> Hi again,
>>
>>
>> I created a thread some days ago, while I was trying to access a C++
>> DLL using my C# program. First of all, I want to thanks all the guys who
>> helped me. But I still have a problem... My C++ DLL has a function which
>> has the following prototype:
>>
>> long FF_Function(void* inInstance, char* inDate, FF_Struct**
>> outStructure, long* outNumber);
>>
>> The structure has the following prototype:
>>
>> struct FF_Struct
>> {
>> long m_Index;
>> char m_ID[FF_ID_LEN];
>> char m_Date[FF_DATETIME_LEN];
>> };
>>
>>
>> I know that the function should return me an array of FF_Struct, and
>> that the parameter outNumber has the number of elements stored in the
>> FF_Struct array. Now, I'm tring to translate it in C#, and here is what I
>> did (which does not work, of course):
>>
>>
>> public class FF_AO_OrderList
>> {
>> public int index;
>> [MarshalAs(UnmanagedType.ByValTStr, SizeConst = FF_ID_LEN)]
>> public string numeroCommande;
>> [MarshalAs(UnmanagedType.ByValTStr, SizeConst = FF_DATETIME_LEN)]
>> public string dateCommande;
>> }
>>
>>
>> [DllImport("FF_myDLL.dll")]

Re: Another C# marshaling question

```
>> public static extern int FF_Function
>> (
>> IntPtr inInstance,
>> [MarshalAs(UnmanagedType.LPStr)]
>> string date,
>> out FF_Struct[] outStructure,
>> out int outNumber
>> );
>>
>>
>> I can access the function, but it returns me an error message ("The
>> parameters specified for the interface are incorrect."). I don't know how
>> to deal with this structure and its double pointer. Even in unsafe mode
>> (then I use a FF_Struct** parameter), I have the same problem.
>>
>>
>> Does anyone knows what I'm missing ? I'd greatly appreciate some help,
>> as I'm completely out of ideas... Thanks !!!!
>>
>>
>> Laurent
>>
>
>
```

• **References:**

- ◆ **[Another C# marshaling question](#)**
 ◇ From: Laurent
- ◆ **[Re: Another C# marshaling question](#)**
 ◇ From: Nicholas Paldino [.NET/C# MVP]

- Prev by Date: **[Re: Difference between type castings?](#)**
- Next by Date: **[Re: DataSet.ReadXml](#)**
- Previous by thread: **[Re: Another C# marshaling question](#)**
- Next by thread: **[MDI Tab controls](#)**
- Index(es):
 - ◆ **[Date](#)**
 - ◆ **[Thread](#)**