

## Re: Array of value types

**Source:**

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.csharp/2004-10/1206.html>

---

**From:** Richard Blewett [DevelopMentor] ([richardb\\_at\\_develop.com](mailto:richardb_at_develop.com))

**Date:** 10/06/04

To: microsoft.public.dotnet.languages.csharp

Date: Wed, 06 Oct 2004 04:31:56 -0700

Yes

Regards

Richard Blewett – DevelopMentor

<http://staff.develop.com/richardb/weblog>

[nntp://news.microsoft.com/microsoft.public.dotnet.languages.csharp/](mailto:nntp://news.microsoft.com/microsoft.public.dotnet.languages.csharp/)>

Just one more question regarding value types;

consider this snippet:

```
foreach (IFDENTRY e in ifd.entries)
{
...
}
```

Given that IFDENTRY is a structure, and ifd.entries is a field of type System.Array

of type IFDENTRY, is this causing a copy as well for each item in the array?

---

Regards,

Dennis JD Myr?n

Oslo Kodebureau

"Jon Skeet [C# MVP]" <[skeet@pobox.com](mailto:skeet@pobox.com)> wrote in message news:MPG.1bcc948f778b2a9898b5b9@msnews.microsoft.com...

Dennis Myr?n <[dennis@oslokb.no](mailto:dennis@oslokb.no)> wrote:

> >Does it really need to be a structure then?

> Well, i thought so...

> The structure directly represents a chunk of a binary file.

> I use:

> [ StructLayout( LayoutKind.Sequential, Pack = 0x1 ) ]

> to explicitly specify that the order is essential.

> I use System.Runtime.InteropServices.Marshal class

> to create the structure instance from a byte array.

> Quite embarrassing, I was sure the StructLayout attribute could only

microsoft.public.dotnet.languages.csharp: Re: Array of value types

- > *be applied to structures, but as i tested making it a class now, it*
- > *still compiles without errors.*

I wouldn't like to say whether you can still do it, to be honest.  
Interop is far from a specialty of mine :(

However, if you have an array of references, you \*probably\* won't be able to pass that as an array of that type by interop, because the data won't be in the right place. Ask in the interop group for more information.

On the other hand, if you're going to pass all of this via interop anyway, then it's probably not going to be that much of an issue doing the in-memory copy to start with.

- > >> *My option might be to, rather than assigning the elements,*
- > >> *to use the array directly with indexing(a lot of indexing then):*
- > >>
- > >> *STRUCT [] a = new STRUCT [0xA];*
- > >> *a [0x0].PROPERTY = 0x0;*
- >
- > > *Yes, you could do that – but if you're going to set any significant*
- > > *number of properties, wouldn't you be better off with a copy anyway?*
  
- > *Just out of interest, where would the breakpoint(array index accesses)*
- > *be where to rather do a copy than working with array indices*
- > *(i can test this by myself though, just if you already know a good*
- > *answer)?*

I don't know of anything like that – it's just a case of balancing the convenience of copying the whole structure in one go with the increasingly marginal performance benefits of using indexing.

--

Jon Skeet – <skeet@pobox.com>

<http://www.pobox.com/~skeet>

If replying to the group, please do not mail me too

----

Incoming mail is certified Virus Free.

Checked by AVG anti-virus system (<http://www.grisoft.com>).

Version: 6.0.771 / Virus Database: 518 – Release Date: 28/09/2004

[microsoft.public.dotnet.languages.csharp]