

## Re: ByRef/Ref passing in Web Services

---

*Source:*

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.framework/2005-05/msg00071.html>

---

- *From:* "Sahil Malik [MVP]" <[contactmethrumyblog@xxxxxxxxxxx](mailto:contactmethrumyblog@xxxxxxxxxxx)>
  - *Date:* Tue, 3 May 2005 11:23:08 -0400
- 

Okay here is a more detailed answer – but don't rant at me, I am just trying to discuss with you.

[http://www.w3.org/TR/2000/NOTE-SOAP-20000508/#\\_Toc478383533](http://www.w3.org/TR/2000/NOTE-SOAP-20000508/#_Toc478383533)

<--- 7.1, bullet #5 – says .."… method response is viewed as a single struct containing an accessor for the return value and each [out] or [in/out] parameter. "

Okay what I meant by a dumb schema was – here is an example –

```
public class Customer {  
    public Customer() {}  
    public string FirstName { get {..} set {...} }  
}
```

Basically, XmlSerializer in it's serialization and deserialization – will not loose any nature of the object – because the object itself contains no logic inside, just a dead – data object map.

No I am not thinking in Object terms, if I were, I'd make this guy inherit from a Person, and Person would have a constraint that a person can have only two legs and no more. That would not be extractable via soapsuds/XmlSerialization – thus doesn't fall into the WebService philosophy.

Finally ----

```
> > Why should this work for only Intrinsic data types? :-/ .. or is there  
> > a trick to making it work with serialized object graphs?  
>  
> So, this is a flawed way to think about web services. They're not objects,  
> the'yre XML messages.
```

Agreed. but the above object is serialized to Xml, sent over, and deserialized back to yourproject.webreferencename.object – which has no logic inside, but a dumb object !! So true it is about message passing, but I could pass in a schema. Yes sir, this is still schema headed thinking, and

## Re: ByRef/Ref passing in Web Services

not object headed thinking. I mean, a webservice isn't limited to passing in only Ints/doubles and other scalar types. Even the Customer object I specified is readable through a non .NET web service because (magic words follow) – "The proxy object is created via. the information in the WSDL".

So ... in short, I agree with your arguments,

Now back to the question – How do I make ref type work with non scalar data types? (Not that I'd ever want to, but for academic reasons .. How?)

– Sahil Malik [MVP]

<http://codebetter.com/blogs/sahil.malik/>

"Brock Allen" <ballen@xxxxxxxxxxxxxxxxxxxx> wrote in message

[news:625392632507094095196560@xxxxxxxxxxxxxxxxxxxxxxxxxxxx](mailto:news:625392632507094095196560@xxxxxxxxxxxxxxxxxxxxxxxxxxxx)

> Where is this in the SOAP spec -- and don't say Section 5 encoding rules,  
> since that's long outdated. If it is in the modern version of the SOAP  
spec,

> then I'd be suprised (I can't be bothered to go look). If it's not, then

> I'd rephrase "WebServices can indeed pass ByRef/ref parameters" to say  
"The

> .NET framework will map a ref param onto WebServices semantics". Just a  
subtle,

> yet important distinction.

>

>> So my question is – If I mark an int as "ref" in a WebMethod, it seems  
>> to work. But if I am exposing a dumb schema (non–intelligent business  
>> object), then it doesn't work. (Dumb Schema Non Intelligent Business  
>> object example – a class customer, with private string FirstName,  
>> encapsulated as a property – that's it !! (Default public constructor  
>> present))

>

> I'm not sure what a "Dumb Schema" means, but my sense is that your  
approach

> to web services is from an object mindset which is fatal. My sense is that  
> you also already know this, but at the sake of being redundant, I'll  
mention

> it again. Web Services are about XML message passing. Objects and  
[WebMethods]

> are simply a convenience afforded to you by the framework you choose to  
> use to implement WebServices. That framework is very good at hiding the  
details

> of the underlying messaging framework (SOAP, WSDL, XSD, etc). And in fact  
> it's too good at times. My point is that the framework is so good at  
hiding

> the real thing we're working with (XML) such that it lets you do dumb  
things

> that you really shouldn't be doing. Sure those dumb things work if you  
have

## Re: ByRef/Ref passing in Web Services

- > .NET on both sides, but they don't make sense in a true introp scenario.
- > The reason is that XML Schema is how we represent the structure of the XML
- > we're sending across the wire, and the [WebMethods] framework infers an XSD
- > from your parameters' type definitions. But there are so many things you
- > can do with classes in .NET that don't map correctly to XSD. This is part
- > of the debate over contract first.
- >
- >> Why should this work for only Intrinsic data types? :-/ .. or is there
- >> a trick to making it work with serialized object graphs?
- >
- > So, this is a flawed way to think about web services. They're not objects,
- > they're XML messages.
- >
- > Now, of course, the point of the framework is to make it easier to develop
- > web services, so you can pass objects as long as you know that the
- > inferred
- > XSD is correct. [And finally, the answer I know you're looking for] The
- > ..NET
- > WebService framework uses the XmlSerializer to do this mapping (for the
- > actual
- > object instances passed back and forth, at least). XmlSerializer is a XSD
- > friendly object serilization/deserilization implementation. One part of
- > that
- > is that it only will access what's public because the private keyword is
- > a sementic specific to a .NET class (and web services aren't about passing
- > classes/objects). This is in contrast to the .NET remoting serilization
- > implementation
- > (SoapFormatter, and BinaryFormatter). The purpose of those is to
- > faithfully
- > pass objects across the network because we have .NET on both sides. This
- > will dig into private fields to correctly serialize all of the state so
- > that
- > the object can be fully (and correctly) deserialized on the other side.
- >
- > In my web service implementations I never take .NET classes meant to be
- > used
- > by other .NET code and pass those pas paremeters or return values from my
- > webservice. I instead create "shell" classes that only hold public fields
- > and perhaps references to other "shell" classes. I create them in such a
- > way that I know they'll be serialized out correctly according to the XSD
- > rules. This way I'm able to utilize the .NET framework implementation of
- > web services that wants to map XML messages onto classes and [WebMethods].
- >
- > Sorry for the long rant and aploogies if you already know all of this. I
- > feel like this is such a poorly understood distinction, so if this isn't
- > for you then perhaps for others reading it.
- >
- > -Brock
- > DevelopMentor
- > <http://staff.develop.com/ballen>

>  
>  
>  
>  
>

.

---

- **Follow-Ups:**

- ◆ **Re: ByRef/Ref passing in Web Services**

- ◇ From: Brock Allen

- **References:**

- ◆ **Re: ByRef/Ref passing in Web Services**

- ◇ From: Brock Allen

- Prev by Date: **Re: Problem with Publisher Policy File (Can't install)**

- Next by Date: **Re: Custom Attribute**

- Previous by thread: **Re: ByRef/Ref passing in Web Services**

- Next by thread: **Re: ByRef/Ref passing in Web Services**

- Index(es):

- ◆ **Date**

- ◆ **Thread**