

Re: Representing Biz Entity in XML: Use XMLSerializer or custom using XSD?

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.framework.webservices/2004-04/0322.html>

From: David (*anonymous_at_discussions.microsoft.com*)

Date: 04/27/04

Date: Tue, 27 Apr 2004 14:16:03 -0700

Dino,

Thanks for your response.

Yes, I took a look at XSD.exe but didn't think it would work with our BusinessEntity since currently our BusinessEntity is a custom .NET class object (and not a dataset or XML document) with some private variables (some being classes themselves) and/or read-only properties that don't make the translation when using XMLSerializer.

Representing our business entity in an XSD was difficult because of these internal constructs that we don't necessarily want to publish access to yet need to be around when the BusinessEntity is being used in our own .NET WinForm client (which uses Binary serialization to rehydrate the object). For example an ArrayList of possible values for a variable/property that come from a database that we don't want to give users of this object the ability to modify (or have the client access the database again).

Our web service was only meant for those applications other than our own (and not necessarily .NET aware) that just want to get a snapshot of the data in an XML format (and don't care about trying to reconstitute into a .NET object). Using XMLSerializer works great except for the fact that some read-only properties didn't come across that we wanted too.

I guess you could say the root (ha, get it?) of our problems is trying to marrying up between a complex .NET object that implements OOP and an XML document to be used by a Web Service.

So I was thinking that we just define an XSD outside of the BusinessEntity that represents that BusinessEntity in an XML format (sort of like BusinessEntity 'lite'). And also include things that our primary application doesn't care about but downstream applications might (for example, the terminology of certain properties for a report).

Our Web Service 'facade' would then just create the regular old BusinessEntity object and then the complex BusinessEntity would have to know that particular parts of itself go into particular elements defined by the XSD. The Web Service would then just call a function on the BusinessEntity, giving it the XSD maybe, and would return an XML document for the Web Service to give back to the caller.

Am I totally reinventing the wheel here or making things way harder than they have to be? Should I go back to the drawing board on how to represent our BusinessEntity to get it into something a little more XML

friendly? How else could I represent portions of my BusinessEntity that are read-only in the .NET programming world but yet have them show up by using XMLSerializer? I guess I could just leave the 'Set' part of the Property declaration blank.

Again, thanks for replying.

-David

----- Dino Chiesa [Microsoft] wrote: -----

Have you looked at the xsd.exe tool included in the .NET SDK ?

What it allows you to do is define an XSD, then generate .NET classes (C#, VB) that correspond to that XSD. Those classes can then serialize to an XML document that validates to that given XSD.

So: Produce your entity definitions in XSD, and generate the classes via xsd.exe. you can also import this XSD into a WSDL document, and generate your server skeletons and client proxies from it (using wsdl.exe).

A skeleton method for the webservice generated in this way might look like:

```
BusinessEntity Method1(Request a) { ... }
```

And you could just return a BusinessEntity on the wire from that method. If you sniffed the wire you would see an XML document that validates to the given XSD. The method you described, (a method that would return an XmlDocument built against the XSD.) is implicitly provided by the XmlSerialization in .NET.

If for some reason you needed to, you could also generate the Xml manually, via calls to the XmlSerializer classes.

xsd.exe has some limitations. It does not support the full type features in XSD. But in many cases it is enough to meet your needs.

To answer some of your specific q's:

- > *Of course the business entity would have to do the creation and field*
- > *mapping to the XML document. I am not sure if this is a good idea or*
- > *not.*
- >

Not. XmlSerializer does this for you, why do it manually? Unless XmlSerializer presents a blocking limitation, I would just use it. It's simple and easy and free (low labor cost).

- > *Also, consumers of the WS could query just for the latest XSD file for*
- > *whatever purpose outside of actually getting a business entity*
- > *manifestation.*

This is easily done via the foo.asmx?WSDL (autogen'd WSDL) feature. The XSD will be included in the returned doc. However for control, you may wish to

microsoft.public.dotnet.framework.webservices: Re: Representing Biz Entity in XML: Use XMLSerializer or custom using

disable autogen of WSDL and just expose the XSD as you described.

I guess the main point is that there is no inherent conflict in your desire to use XSD to define types, and the desire to use BusinessEntities in code. ..NET lets you have it all!

-Dino

--

Dino Chiesa
Microsoft Developer Division
d i n o c h @ E l i d e T h i s . m i c r o s o f t . c o m