

Re: Implementing a common SOAP Header across multiple Web Service Pages

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.framework.webservices/2007-02/msg00083.htm>

- *From:* "John Saunders" <john.saunders at trizetto.com>
 - *Date:* Wed, 14 Feb 2007 17:55:02 -0500
-

"Joseph Geretz" <jgeretz@xxxxxxxxxx> wrote in message
<news:%23t%23G9SIUHHA.1000@xxxxxxxxxxxxxxxxxxxxxxxxxxxx>

I have the following class which I am serializing and passing back and forth between my Web Service application and the client.

```
public class Token : SoapHeader
{
    public string SID;
    public string UID;
    public string PWD;
}
```

This is working nicely. However, I'd like to elaborate on my WS application by splitting my application up into several different pages. For example, one page will handle Login / Logout (and other session lifecycle) methods, one page will handle document retrieval methods, one page will handle prescription methods, etc, etc.

Joseph,

I think you'll find it easier to not consider these as pages. They're not, really, and they don't need to correspond to pages in a web site.

I think that I have to go this route so that I can have multiple developers working concurrently on different areas of the application without contending for the same ASMX page. Hmm, now that I'm thinking about this, perhaps this can be achieved via partial classes? This might be a better solution so that ultimately, the client would only need to bind to a single Web Service page, rather than to a dozen or so separate Web Service pages representing the various functional areas of my application. So here's an

Re: Implementing a common SOAP Header across multiple Web Service Pages

ancillary question; What is your recommendation for partitioning a Web Service application into areas of functionality? Should these be implemented

as separate pages or as separate partial classes? Are there any performance

or other implications? The difference to these approaches as far as the client would be concerned would be the instantiation of one large class for every transaction, as opposed to the instantiation of different smaller classes for different areas of the application, however in the latter case, the client would need to bind to multiple Web Service pages. Would any of this make any difference?

First of all, you shouldn't be thinking about implementation at this stage of your design. Don't worry about partial classes, versus whatever. Worry about the interface that the clients will use.

Now, it's not going to matter in terms of performance whether you have a single .asmx file (that is, a single [WebService]) with a lot of [WebMethods], or multiple [WebService] classes. The proxy classes are very lightweight and store little, if any, instance data. The differences will have to do with how the client wants to use the service. For instance, is the client likely to use many different parts of your system within a given session? If so, then that suggests the client would be better off instantiating a single class and passing the reference throughout the application (possibly as an instance member), rather than having to instantiate a class when they're in one part of the application, then having to instantiate another in a different part of the application.

Frankly, if this is the first web service you're implementing, I'd keep it simple – a single .asmx file.

As to multiple developers, keep in mind that the .asmx files don't need to contain your implementation. In fact, they probably should not contain your implementation. Instead, each web method should instantiate one or more implementation classes and should delegate the operation to them. I personally like to treat the web service as a Facade in front of my application. My web methods each instantiate a class specific to that method, then delegate to that instance.

Of course, related "service layer" classes share an inheritance hierarchy, so common code is shared in that way. For instance, all of my service layer classes which require that the user be logged in all inherit from a base SessionServiceLayer class, which knows how to validate an authentication token.

In this way, you can easily change how functions are arranged at some later date. For instance, you may find that you have clients who should never even know about some part of your application. You can easily create a .ASMX file which only contains those methods. This will still be very little duplicated code, since the web methods will be little more than:

Re: Implementing a common SOAP Header across multiple Web Service Pages

```
[WebMethod]
public ReturnType OperationOne(int a, string b, ...)
{
    OperationOneServiceLayer service = new OperationOneServiceLayer(a, b,
    ....);
    return service.Execute();
}
```

As to your original question, simply put the soap header class in a separate file. As a public class, it can be used by any class that needs it. Each ..ASMX file will have to have its own public instance of that header, but so what?

John

Anyway, let's say I were to proceed to develop separate ASMX pages. How and where would I define this class so that it would be available to all pages on the server and to the client on the workstation as well? Would I need to define this in a separate Assembly (i.e. DLL Utility Library) which would then need to be deployed to both client and server tiers?

Thanks for your advice.

– Joseph Geretz –