

Re: soapsuds

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.framework.remoting/2004-06/0144.html>

From: Sahil Malik (contactmethrumblog_at_nospam.com)

Date: 06/09/04

Date: Wed, 9 Jun 2004 14:24:32 -0400

Why is remoting slow? Because as COM used to let you create an ActiveX Exe, or a Windows Service with a COM interface, and hence you had a pre-instantiated exe running in a singleton mode, for remoting, even if you are restricted to one machine only, you still have to use the full fledged remoting framework as you would on multiple machines, including a TCP connect, which makes it slower. Whereas the classfactory implementation I suggested, tries and detects if it is indeed all on one machine, in which case it will get you the instance of the object with a "new", or it will create a masquerading proxy using `Activator.GetObject` in which case it will be no faster/slower but exactly the same as a plain vanilla remoted object.

Revised framework implementations will probably not break your application .. atleast not severly. However, the paradigm might shift from remoting to Indigo .. which is relatively different. remoting however will continue to work just as "dir" works even today on dos prompt.

Ok your requirements seem to be a bit different than mine. While I didn't want older versions talking to newer ones, you seem to have a backward compatibility requirement. Let me come back to that issue afterwards and let me answer your other questions first. The redeployment framework was actually my method of updating clients, since I didn't want V1 clients talking to V2 clients, but in your scenario, you (I guess) want the users to pay for V2, before they can use it, but until then continue using V1.

– Sahil Malik

Independent Consultant

You can reach me thru my blog – <http://dotnetjunkies.com/WebLog/sahilmalik/>

"Richard Bell" <richardb@rogge.co.uk> wrote in message
news:uCr6%23QkTEHA.2464@TK2MSFTNGP10.phx.gbl...

> *Many thanks for that. I am not being funny, I really do want to understand
> your points and this general desire to use interfaces to control
versioning
> rather than using the versioning built into the framework. Forget the
> soapsuds utility. Let's just assume that we have a process which generates
> the proxy code for the remote object that works. Code it by hand if
> necessary. We can then compile the client using this code which allows us*

to

- > *instantiate objects using the new operator. Both client and server are*
- > *oblivious as regards remoting and I do not need to write any bespoke*
- > *instantiation routines. I am, however, not precluded from doing so.*
- >
- > *Regarding your points:–*
- > *#1. Again, forget soapsuds. Code the proxy stubs by hand if you like. None*
- > *of this stops you from implementing your own remoting framework. The whole*
- > *thing is extensible. You are free to serialise objects in any way that you*
- > *want.*
- > *#2. In your framework you are forced to either have an all new v2 client*
- and
- > *server or be sure that the v1 clients will work with the v2 server. The*
- > *former requires you to distribute a new client. What about clients that*
- have
- > *not paid for v2 software? Are you going to have all new v2 interfaces in*
- the
- > *new v2 General assembly? Or are v2 clients going to be serviced by a*
- > *seperate server/port? In my case I can handle all of these scenarios,*
- plus,
- > *I have the option to allow old client installations to use the v2 server,*
- > *without changing the old client installations. I have everything that you*
- > *have plus extra options. I do have a 'common' assembly on client and*
- server,
- > *the same as you, I just use*
- > *IMyRemoteInterface i = (IRemoteMyInterface)new MyRemoteObject();*
- > *All we are talking about is how the remote object is instantiated. I am*
- > *using the framework. You are using a bespoke version of what the framework*
- > *does for you.*
- > *#3 Why is remoting slow? Can you be sure that your ClassFactory object is*
- > *optimised for all potential remoting scenarios? I am making the*
- assumption,
- > *and I do not think that it is entirely unjustified, that revised framework*
- > *implementations will not break the existing one. Therefore, there is a*
- good
- > *chance that I get the updated architecture for free. In any case, the*
- > *changes I need to make are unlikely to exceed those that would be required*
- > *for a bespoke method.*
- > *#4 I do not see what your redeployment framework, very good though it*
- > *sounds, has to do with instantiating objects. What you have implemented is*
- > *perfectly feasible when using a proxy class on the client side. However, I*
- > *reiterate, I can place a new server assembly on my server machine and all*
- my
- > *clients, old and new, will still happily carry on working. Unless I am*
- > *missing something, I do not think that you can say the same.*
- >
- > *"Sahil Malik" <contactmethrumyblog@nospam.com> wrote in message*
- > *news:OgvJcpiTEHA.3988@TK2MSFTNGP10.phx.gbl...*
- > *I don't know what wise men would say, but here is what I'd say.*
- > >
- > > *Thanks for your comments. I have come across this view many times,*

> > *however,*
> > > *I do not understand it. I have had no problems using soapsuds at all*
> *apart*
> > > *from the one that I have mentioned in this post which I have now*
> *resolved.*
> > > *This would appear to be related to namespacing. If the interface in*
the
> > > *'general' assembly is in the same namespace as the implementation*
object
> > *in*
> > > *the 'server' assembly, then the soapsuds extraction falls over.*
Placing
> > *the*
> > > *objects in different namespaces resolved the issue. Is there a*
> *restriction*
> > > *in .Net that namespaces cannot span assemblies? I am not aware of it.*
In
> > *any*
> > > *case, it is easy enough to code the 'proxy' remote class code*
including
> > *the*
> > > *Soap attributes by hand.*
> >
> > *The problem with Soapsuds is, you cannot have one small exe generate*
this
> > *information for every possible case out there. Other such examples in*
.NET
> > *framework, where you are better off implementing stuff yourself are the*
> > *Serializable Attribute versus ISerializable, CommandBuilder versus*
> > *specifying commands etc. etc. (fairly long list really). Anyway without*
> > *wasting time on "Why not to use soapsuds", since there is plenty about*
> *that*
> > *online, just don't use it .. hehe :).*
> >
> > > *I can now use new to instantiate my remote object, access it via an*
> > > *interface defined in a shared assembly (which can be implemented my*
> *other*
> > > *remote objects) which leaves me totally free to redevelop the server*
> > *object.*
> > > *I have complete control over versioning via the framework. I believe*
> *that*
> > *is*
> > > *what versioning was designed to do. However, nothing is stopping me*
from
> > > *adding a V2 interface or allowing old clients to connect to new server*
> > > *assemblies.*
> >
> > *True, and that in my mind is a better approach. You should have a*
seperate
> > *dll which sits both on the server and the client allowing objects to*
talk

> to
>> each other. Yes you could have a seperate V2 interface in the same dll,
> but
>> the question is, why would you want older clients to talk to newer
server
>> assemblies? Ideally the whole product should work as one unified
version.
>> i.e. Server version = 2.x, will work only with Client version 2.x. And
the
>> good news is that you really don't have to do anything to enforce that.
As
>> long as the common dll on both client and server have the same matching
>> version #'s, the Server and Client will work in tandem. Whats better,
you
>> could now have side by side versions, and actually have 2 servers same
>> machine different versions running on different ports.
>>
>>> I am very keen to totally separate the client and server from the
> remoting
>>> process and use new to instantiate remote objects so that I can
enhance
>> and
>>> develop the remoting protocol without having to recode either client
or
>>> server objects. This cannot be achieved via the framework by using
>>> interfaces or abstract classes. I have seen custom implementations
that
>>> allow using instantiation via new by basically walking through the
> config
>>> file to create the remote object, but is that not what the framework
> does
>> on
>>> your behalf? Will I be able to do it better? Furthermore, if I create
a
>>> custom solution, I will most likely not benefit from future
enhancements
>> to
>>> the framework remoting methodology.
>>
>> Thats a good idea. With Indigo on the horizon, a few things will change,
> so
>> your communication methodology should be changeable. What I have done
is,
>> that I have designed my own class factory. That class factory is
> responsible
>> for either doing a "new", or Activator.GetObject depending upon
Appdomain,
>> so my locally configured objects don't get hit by the performance
overhead
>> of remoting (remoting is verrrryyy slow). Another added side benefit
from

> > *this is, that really if I use a class factory, when architecture changes,*
> > *all I have to do is modify the class factory, and not worry about "new"*
> > *doesn't work anymore ore something similar thereof. You get the added*
> > *performance bang too :).*
> >
> > > *However, my major problem with using interfaces only to control*
> > *versioning*
> > > *is that you may well run into (actually in my view most likely to run*
> > *into)*
> > > *deployment problems down the road. In particular, how would you force*
> > > *existing old clients to use old server assemblies? You may want to. I*
> > *would*
> > > *not want to make the assumption at the start of the project that v 1.0*
> > > *clients will always be able to use v N.0 server assemblies.*
> >
> > *I have already elaborated on this in the first paragraph, but let me add*
> > *another good peice of info. I have another common dll whose version*
number
> > *I*
> > *donot change. When the client starts up, it uses that common dll (you*
> > *could*
> > *eve use soapsuds for something this simple), to check the version*
number,
> > *and if the client's version is out of whack, it automatically downloads*
> > *the*
> > *new MSI, fires up an installer.exe, which first kills the client, and*
then
> > *runs the MSI. One important thing to consider here, whenever I recreate*
> > *the*
> > *MSI, I change the product version number, but I DONOT change the product*
> > *package GUID and another GUID there, that way it "thinks" that really*
this
> > *is a re–install of the same product (AHA tricked ya). That way my client*
> > *will*
> >
> > > *a) Always remain updated.*
> > > *b) Never get out of version errors.*
> > > *c) The upgrade process is very very simple for the end user.*
> >
> > *Hope that helped.*
> >
> > – *Sahil Malik*
> > *Independent Consultant*
> > *You can reach me thru my blog at –*
> > *<http://www.dotnetjunkies.com/weblog/sahilmalik/>*
> >
> >
> >
> > *"Richard Bell" <richardb@rogge.co.uk> wrote in message*
> > *news:%23JT397gTEHA.3336@TK2MSFTNGP10.phx.gbl...*

> > > *Thanks for your comments. I have come across this view many times,*
> > *however,*
> > > *I do not understand it. I have had no problems using soapsuds at all*
> *apart*
> > > *from the one that I have mentioned in this post which I have now*
> *resolved.*
> > > *This would appear to be related to namespacing. If the interface in*
> *the*
> > > *'general' assembly is in the same namespace as the implementation*
> *object*
> > *in*
> > > *the 'server' assembly, then the soapsuds extraction falls over.*
Placing
> > *the*
> > > *objects in different namespaces resolved the issue. Is there a*
> *restriction*
> > > *in .Net that namespaces cannot span assemblies? I am not aware of it.*
In
> > *any*
> > > *case, it is easy enough to code the 'proxy' remote class code*
including
> > *the*
> > > *Soap attributes by hand.*
> > >
> > > *I can now use new to instantiate my remote object, access it via an*
> > > *interface defined in a shared assembly (which can be implemented my*
> *other*
> > > *remote objects) which leaves me totally free to redevelop the server*
> > *object.*
> > > *I have complete control over versioning via the framework. I believe*
> *that*
> > *is*
> > > *what versioning was designed to do. However, nothing is stopping me*
from
> > > *adding a V2 interface or allowing old clients to connect to new server*
> > > *assemblies.*
> > >
> > > *I am very keen to totally separate the client and server from the*
> *remoting*
> > > *process and use new to instantiate remote objects so that I can*
enhance
> > *and*
> > > *develop the remoting protocol without having to recode either client*
or
> > > *server objects. This cannot be achieved via the framework by using*
> > > *interfaces or abstract classes. I have seen custom implementations*
that
> > > *allow using instantiation via new by basically walking through the*
> *config*
> > > *file to create the remote object, but is that not what the framework*
> *does*

> > *on*
> > > *your behalf? Will I be able to do it better? Furthermore, if I create*
a
> > > *custom solution, I will most likely not benefit from future*
enhancements
> > > *to*
> > > *the framework remoting methodology.*
> > >
> > > *However, my major problem with using interfaces only to control*
> *versioning*
> > > *is that you may well run into (actually in my view most likely to run*
> > *into)*
> > > *deployment problems down the road. In particular, how would you force*
> > > *existing old clients to use old server assemblies? You may want to. I*
> > > *would*
> > > *not want to make the assumption at the start of the project that v 1.0*
> > > *clients will always be able to use v N.0 server assemblies.*
> > >
> > > *What say the wise men?*
> > >
> > > *"Sahil Malik" <contactmethrummyblog@nospam.com> wrote in message*
> > > *news:%23lsDy\$XTEHA.504@TK2MSFTNGP11.phx.gbl...*
> > > > *Wise men say -> Don't use soapsuds*
> > > >
> > > > --
> > > > -- *Sahil Malik*
> > > > *Independent Consultant*
> > > > *You can reach me thru my blog -*
> > > > *<http://dotnetjunkies.com/WebLog/sahilmalik/>*
> > > >
> > > >
> > > >
> > > > *"Richard Bell" <richardb@rogge.co.uk> wrote in message*
> > > > *news:O2lM2IUTEHA.2716@tk2msftngp13.phx.gbl...*
> > > > > *When I use soapsuds to extract meta data for a class which*
> *implements*
> > *an*
> > > > *interface which is defined in a separate assembly I get a null*
> > *reference*
> > > > > *error. Do all interfaces/abstract classes which are implement by a*
> > *class*
> > > > > *being extracted have to be defined within the same assembly, or,*
am
> *I*
> > > > *doing*
> > > > > *something wrong. Many thanks.*
> > > > >
> > > > >
> > > > >
> > > > >
> > > > >
> > > > >

microsoft.public.dotnet.framework.remoting: Re: soapsuds

> > >
> >
> >
>
>