

Re: Generic deserialization

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.framework.remoting/2004-07/0020.html>

From: Lee Gillie (ANTISPAMIFICATION_lee_at_odp.com)

Date: 06/30/04

Date: Wed, 30 Jun 2004 08:31:52 -0700

VERY helpful Ken!!! I greatly appreciate your taking time to teach this old dog some new tricks.

Best regards – Lee Gillie – Spokane, WA

"Ken Kolda" <ken.kolda@elliemae-nospamplease.com> wrote in message news:uysBvpjXEHA.3044@TK2MSFTNGP09.phx.gbl...

> *The error you're getting is, as you suspected, due to the fact that your*
> *class derives from MarshalByRefObject. When the consumer gets the reference*
> *to this object and attempts to invoke one of its methods, it must open a*
> *remoting channel back to the source, since that's where the object actually*
> *resides. In this way, your "source" application acts as both client and*
> *server. That's not a problem, but it sounds like that's not the model you're*
> *trying to create.*
>
> *The issue is that you indicated that you don't your message server service*
> *to know about the client's classes. If you mark your classes as Serializable*
> *but not MBR, then the server must generally have the implementation available for the class, since the class will be deserialized on the server.*
> *But, assuming your server service has no need to ever know what the underlying object is (i.e. it just passes it to a consumer which DOES know*
> *what type of object this is), you could do something like this:*
>
> *Create two wrapper classes in a separate assembly than from your remoted*
> *server service classes, call them SourceWrapper and ConsumerWrapper.*
The

> *SourceWrapper* would look something like:
>
> *class SourceWrapper*
> {
> *public SourceWrapper(string serverUri)*
> { *// Save off Uri to instance variable* }
>
> *public void Send(object o)*
> {
> *// Serialize object using BinaryFormatter to get byte array*
> *// Call Activator.GetObject() to get the remoted Source*
object from
> *server*
> *// Invoke Source object's Send(), passing the bytes.*
> }
> }
>
> *The ConsumerWrapper* would, conversely look like:
>
> *class ConsumerWrapper : MarshalByRefObject*
> {
> *public event MessageArrivedEventHandler MessageArrived;*
>
> *public ConsumerWrapper(string serverUri)*
> {
> *// Get the Consumer object from server and add an event*
handler or
> *callback for this instance*
> }
>
> *private onMessageArrived(byte[] data)*
> {
> *// Deserialize object using Binary formatter*
> *// Raise MessageArrived event for client to be notified*
> }
> }
>
> *In other words, the SourceWrapper handles the serialization and the*
> *ConsumerWrapper handles the deserialization so the client app*
doesn't have
> *to worry itself about how this works. The server service in between*
knows
> *nothing except how to pass byte arrays around, so it's completely*
generic.
>
> *Hope that helps –*
> *Ken*
>
>
> *"Lee Gillie" <ANTISPAMIFICATION_lee@odp.com> wrote in message*
> *news:esoOd3iXEHA.1356@TK2MSFTNGP09.phx.gbl...*

> > *Trying to create a "generic" notification facility. Basically I have a*
> > *number of services which currently use NT MAILSLOTS to broadcast their*
> > *in-progress status to desktops on the LAN. In the mailslot there is*
> > *usually complex delimited data as a string.*
> >
> > *My .NET approach is to instead use a "message server service".*
Also
> > *attempting to provide a component that will make either SOURCE or*
> > *CONSUMER objects. The SOURCE object has one simple "Send" method, and*
> > *accepts a parameter of type "Object". Am making a "Consumer" object*
> > *which posts an event, and again, a single parameter of type OBJECT. I*
> > *am attempting to hide all the infrastructure and remoting as much as*
> > *possible, from the caller of the component. The component is also*
> > *exposing a COM interface. The component makes either a source or*
> > *consumer conversation with the service via one single remoted*
> > *component, which can "broadcast" events to clients. All of this works*
> > *great when the notify parameter is a string. But falls flat when*
> > *trying to do this with an item of type OBJECT.*
> >
> > *Here is my extreme lack of knowledge about serialization...*
> >
> > *Via the remoting method, I was able to send from a SOURCE to the*
> > *server, and it makes it back to the CONSUMER in a separate process.*
> > *But the consumer is having trouble looking at the "object". I get*
> > *this:*
> >
> > *Run-time exception thrown :*
> > *System.Runtime.Remoting.RemotingException – This remoting proxy has no*
> > *channel sink which means either the server has no registered server*
> > *channels that are listening, or this application has no suitable*
> > *client channel to talk to the server.*
> >
> > *The "object" being sent through this system is a "class" using only*
> > *fundamental data types, such as string, integer, and so on. The class*
> > *is marked serializable, and I found a need to inherit from MarshaByRef*
> > *to get the send side to work.*
> >

> > *My thought is that once I get this class instance serialized, it is completely self contained. Yet the message seems to indicate the CONSUMER is somehow trying to get back to some remoting host, or such.*

> > *Probably due to the MBR.*

> >

> > *Any hints on preparing my generic "object" parameter to be fully self-sufficient would be greatly appreciated. In my component I don't have knowledge of my caller's classes. If I can, I would like to not burden my caller to perform serialization and deserialization, if possible.*

> >

> >

> > *Can what I am trying to do even be done?*

> >

> >

> >

>

>