

RE: Remoting and serialization

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.csharp/2004-08/4762.html>

From: javatopia (javatopia_at_discussions.microsoft.com)

Date: 08/21/04

Date: Fri, 20 Aug 2004 20:27:02 -0700

Hi Uttam,

What you have on the Java side is RMI, which is EXACTLY the same as a MarshalByRefObject in .NET. Hmm, only easier because you don't have to do any post-processing steps to make it work (except to create a remoting server that escrows the marshalled objects).

In fact, if you use MarshalByRefObject for all of your "shared" objects (not classes, mind you, because you STILL have to leave a copy of the interface definition on the client), then you can make use of an interface for all of your interactions. The Microsoft PetShop is an excellent example of making use of interfaces and serialization. To extend it to reference marshalling, just envision "MarshalByRefObject" as the base class for all of their DAL classes.

The decision to use Java versus .NET isn't so much about technology as it is about your target platform, in this case. Both technologies will work the same, and you will find that .NET is *much* easier to program than RMI and Java (I have 8 years of Java under me and almost 3 years of .NET). The *best* part of .NET is the container – you get it with ever installation of Windows "Professional" and server, whereas Java will require you to go and find a container.

If you need further help or advice, please don't hesitate to contact me direct via email (jwa@javatopia.com). I am more than happy to help out a fellow Java/.NET programmer.

One more thing – in the Java world, remember that any time you de-serialize an object, you need the ENTIRE class graph for that object on your client. Serialization is a nasty beast no matter how you shake it. .NET remoting using object references, on the other hand, allows you to proxy interfaces on the client, which means you get to leverage a shallow representation of the server object and not sacrifice functionality.

— Jake

jwa@javatopia.com

"Uttam" wrote:

> *Hello,*
> *We are at a very crucial decision making stage to select between .Net*
> *and Java. Our requirement is to download a class at runtime on the*
> *client computer and execute it using remoting or rmi.*
> *Just to keep my question short I am posting trimmed version of my*
> *code.*
>
>
> *//file: Serializable.cs*
>
> *[Serializable]*
> *public class Serializable:ISerializable*
> *{*
> *public string Execute()*
> *{*
> *string str="original text";*
> *return str;*
> *}/public string Execute()*
> *}/public class Serializable:ISerializable*
>
>
> *//file: Client.cs*
>
> *public class Client*
> *{*
> *public static void Main()*
> *{*
> *try*
> *{*
> *ISerializable iSerializable =*
> *(ISerializable)iRemoteObj.GetSerializableObj();*
> *response = iSerializable.Execute();*
>
> *} catch (Exception e)*
> *{*
> *}*
> *}/public static void Main()*
> *}*
>
>
> *I want to keep ONLY interfaces on the client and Implementation on the*
> *server. During runtime I want to get the remote object and execute*
> *them locally.*
>
> *I am able to do the same with the JAVA but in .NET I am forced to*
> *have implementation files of these interfaces on the client computer.*
> *which does not make any sense to me. that means I will have to update*
> *the implementation on all the client computers in case I will have to*
> *update some code in that class.*
> *Then I don't even need remoting I can very well write a standalond*
> *component;*

>
> *This is what I get if i do not copy implementation class on the*
> *client*
>

> *System.Runtime.Serialization.SerializationException: Cannot find the*
> *assembly Serializable, Version=0.0.0.0, Culture=neutral,*
> *PublicKeyToken=null.*
>
> *Server stack trace:*
> *at*
> *System.Runtime.Serialization.Formatters.Binary.BinaryAssemblyInfo.GetAssembly()*
> *at*
> *System.Runtime.Serialization.Formatters.Binary.ObjectReader.GetType(BinaryAssemblyInfo*
> *assemblyInfo, String name)*
> *at*
> *System.Runtime.Serialization.Formatters.Binary.ObjectMap..ctor(String*
> *objectName, String[] memberNames, BinaryTypeEnum[] binaryTypeEnumA,*
> *Object[] typeInformationA, Int32[] memberAssemIds, ObjectReader*
> *objectReader, Int32 objectId, BinaryAssemblyInfo*
> *assemblyInfo, SizedArray assemIdToAssemblyTable)*
> *at*
> *System.Runtime.Serialization.Formatters.Binary.ObjectMap.Create(String*
> *name, String[] memberNames, BinaryTypeEnum[] binaryTypeEnumA, Object[]*
> *typeInformationA, Int32[] memberAssemIds, ObjectReader objectReader,*
> *Int32 objectId, BinaryAssemblyInfo asse*
> *mbleInfo, SizedArray assemIdToAssemblyTable)*
> *at*
>
> *System.Runtime.Serialization.Formatters.Binary.__BinaryParser.ReadObjectWithMapTyped(BinaryObjectWithMapTy*
> *record)*
> *at*
>
> *System.Runtime.Serialization.Formatters.Binary.__BinaryParser.ReadObjectWithMapTyped(BinaryHeaderEnum*
> *binaryHeaderEnum)*
> *at*
> *System.Runtime.Serialization.Formatters.Binary.__BinaryParser.Run()*
> *at*
> *System.Runtime.Serialization.Formatters.Binary.ObjectReader.Deserialize(HeaderHandler*
> *handler, __BinaryParser serParser, Boolean fCheck, IMethodCallMessage*
> *methodCallMessage)*
> *at*
> *System.Runtime.Serialization.Formatters.Binary.BinaryFormatter.Deserialize(Stream*
> *serializationStream, HeaderHandler handler, Boolean fCheck,*
> *IMethodCallMessage methodCallMessage)*
> *at*
> *System.Runtime.Remoting.Channels.CoreChannel.DeserializeBinaryResponseMessage(Stream*
> *inputStream, IMethodCallMessage reqMsg, Boolean bStrictBinding)*
> *at*
> *System.Runtime.Remoting.Channels.BinaryClientFormatterSink.SyncProcessMessage(IMessage*
> *msg)*

```
>  
> Exception rethrown at [0]:  
> at  
> System.Runtime.Remoting.Proxies.RealProxy.HandleReturnMessage(IMessage  
> reqMsg, IMessage retMsg)  
> at  
> System.Runtime.Remoting.Proxies.RealProxy.PrivateInvoke(MessageData&  
> msgData, Int32 type)  
> at NsServer.IRemoteObject.GetSerializableObj()  
> at Client.Client.Main()  
>  
>
```