

.NET Remoting – client development/design–time and run–time requirements

Source:

<http://www.tech–archive.net/Archive/DotNet/microsoft.public.dotnet.framework.remoting/2007–01/msg00037.html>

- *From:* "Zoe Hart" <zoehart@xxxxxxxxxxxxxxxxxxx>
 - *Date:* Tue, 16 Jan 2007 11:04:41 –0500
-

I have a .NET Remoting solution. My remote objects are hosted in a Windows Service on my remote server. My remote objects are configured for SingleCall Server Activation, so instantiation on the client results in a proxy object and the remote objects aren't instantiated on the server until a method is called on the proxy object. My client is a hybrid ASP/ASP.NET web application. The ASP pages access the .NET Remoting objects via a custom ..NET Factory class I built that handles the COM Interop and makes the Remoting call to instantiate the remote objects. Once the ASP page has the proxy remote object, it no longer needs to go through the Factory class but makes its calls directly to the remote object through the proxy object. The ASP.NET pages can instantiate the remote objects directly with the New keyword.

It seems that at development/compile time, the ASP.NET portion of the client web application needs to have a copy of the DLL that defines the remote objects in its bin folder or it won't compile. Is that accurate?

Does the client web application (ASP and/or ASP.NET) need to have the DLL that defines the remote objects anywhere on the client machine at runtime? When an ASP.NET page (or my Factory class) instantiates a new remote object via the New keyword (remoteObject = new MyRemoteObject()) where does the information that defines the proxy class come from? Does it need to be on the client machine or does the client machine get it from the remote server?

The reason I'm asking is that I recently updated several methods in my remote object to take an additional, optional parameter. Since it was optional, I had thought I could update my remote server first and then update my client machines and that old clients would work correctly while they waited to be updated. But they didn't. They reported error number –2146233077, "8 arguments were passed to 'MyRemoteObject::MyRemoteMethod'. 9 arguments were expected by this method. If I drop an updated DLL (one that defines the remote classes with the new optional parameter) on the client machines without changing the client pages that are making the actual calls (so they're still passing the same number of arguments), the error goes away.

This will necessarily change my deployment strategy and suggests I need to

.NET Remoting – client development/design–time and run–time requirements

somehow simultaneously update my remote service and all of my client web servers, even for the addition of an optional parameter. Is that correct?

Thanks,
Zoe