CHAPTER 7

Remoting and Web Services

WITH .NET, CHOOSING a mechanism for making remote method calls is finally a positive experience. With .NET remoting and Web services, we have two great options. But with options come choices. One question we hear most often from developers is, “Microsoft keeps talking about Web services, but when should I use remoting?” Depending on whom you ask, you will probably get different answers.

There are at least two different camps when it comes to this topic. The first camp claims that Web services are the future of distributed programming and will be the only way to make remote procedure calls in the future. This camp looks forward to the new Microsoft Global XML Web Services Architecture (GXA) to provide improvements in security, transaction support, routing, and other services required for true distributed programming.

The other camp believes that there is certainly a place for Web services and looks forward to the improvements to come with GXA. But this camp believes that the true value in Web services comes in integrating disparate systems and platforms across networking boundaries, and that they will not be a panacea for all distributed programming needs. Remoting still has a strong place in this camp’s toolbox for application development.

As you can imagine, the answers to the question first posed are shaped by those sentiments. The response of those who favor Web services goes something like this: “You should always default to using Web services and only use remoting when you need some service that is not currently implemented for Web services or you absolutely need the speed improvement.” The response of the other camp is more along the lines of “Web services are great for interoperability and systems integration, but remoting provides the best performance and function for strict remote procedure calls.”

We are sure you are asking, “So which camp is right?” But, before we get to that, we believe that it is always important to understand what factors are involved before making a decision. To that end, we want to lay out the differences and similarities between remoting and Web services and discuss where each is strongest.
Remoting Overview

In the .NET world, remoting gives you the most flexibility in how your remote method calls are carried out in a variety of ways: the options it provides for transport, serialization format, server host, activation, and state management are a few that come to mind immediately. With these choices, your configuration can go from mimicking Web services almost exactly to being completely contradictory to that model.

Transport

Remoting allows you to use either TCP or HTTP as your transport. HTTP, the same protocol used by Web services, is a more verbose protocol than TCP and is therefore slower. This flexibility is nice, however, because depending on the server host type you choose, you may want or need to use one or the other.

Serialization Format

When choosing which format to which your messages should get serialized, you have the option of using the Simple Object Access Protocol (SOAP), which again mimics the Web service world, or you can use a more compact binary serialization that improves performance. You can use these two options regardless of the host, and therefore it comes down to a decision between trying to use interoperability with SOAP or pure performance with binary.

Server Host

Remoting gives you the option of using any kind of Windows executable to host your server objects or Internet Information Server (IIS). Most often this means that you will be using either a Windows service or IIS in a production environment. Having this flexibility can be vitally important because some companies will want to limit the use of IIS on application servers due to its perceived security implications. Using a custom host executable, not IIS, is also the only way to employ the TCP transport to get the best performance.

Activation

Remoting provides for either server or client activation and supplies a rich lease management model for client-activated objects to control their lifetime. Allowing for different types of activation and lifetime management provides a distinct