Foreword

Pat Helland, formerly of Microsoft, has a great acronym he likes to use when talking about interoperability: HST, or "Hooking Stuff Together." (Actually, he uses an altogether different word there in the middle, but I'm told this is a family book, so I paraphrased.) No matter how much you dress it up in fancy words and complex flowcharts, interoperability simply means "Hooking Stuff Together"—something Web Services are all about.

Ever since the second computer came online, True Interoperability remains the goal that still eludes us. IT environments are home to a wide array of different technologies, all of which serve some useful purpose (or so I'm told). Despite various vendors' attempts to establish their tool of choice as the sole/dominant tool for building (and porting) applications, the IT world has only become more—not less—diverse. Numerous solutions have been posited as "the answer" to the thorny problem of getting program "A" to be able to talk to program "B," regardless of what language, platform, operating system, or hardware the two programs are written in or running on. None had proven to be entirely successful, either requiring an "all-or-nothing" mentality, or offering only solutions to handle the simplest situations and nothing more.

In 1998, Don Box and Dave Winer, along with a couple of guys from Microsoft, IBM, and Lotus, sat down and wrote a short document describing an idea for replicating a remote procedure call stack into an XML message. The idea was simple: If all of the various distributed object toolkits available at the time—DCOM, Java RMI, and CORBA being the principal concerns—shared a common wire format, it would be a simple matter to achieve the Holy Grail of enterprise IT programming: True Interoperability.

In the beginning, SOAP held out the prospect of a simpler, better way to Hook Stuff Together: XML, the lingua franca of data, passed over HTTP, the Dark Horse candidate in the Distributed Object wars, with all the semantics of the traditional distributed object programming model surrounding it. It seemed an easy prospect; just slip the XML in where nobody would see it, way down deep in the distributed object's generated code. What we didn't realize at the time, unfortunately, was that this vision was all

too simplistic, and horribly naïve. It might work for environments that were remarkably similar to one another (à la Java and .NET), but even there, problems would arise, owing to differences XML simply couldn't wash away. Coupled with the fact that none of the so-called standards was, in fact, a standard from any kind of legitimate standards body, and that vendors were putting out "WS-Foo" specifications every time you turned around, the intended simplicity of the solution was, in a word, absent. In fact, to put it bluntly, for a long time, the whole Web Services story was more "mess" than "message."

In Chapter 1 of this book, Mark Hansen writes, "Web Services are not easy." Whatever happened to the "Simple" in "SOAP?"

Ironically, even as Web Services start to take on "dirty word" status, alongside EJB and COBOL, the message is becoming increasingly clear, and the chances of "getting it right" have never been higher. Distractions such as the SOAP versus REST debate aside (which really isn't a debate, as anyone who's really read the SOAP 1.2 spec and the REST dissertation can tell), the various vendors and industry groups are finally coming to a point where they can actually Hook Stuff Together in more meaningful ways than just "I'll pass you a string, which you'll parse…."

As an instructor with DevelopMentor—where I taught Java, .NET, and XML—I had the privilege of learning about SOAP, WSDL, and Web Services from the very guys who were writing the specs, including Don Box and Martin Gudgin, our representative to the W3C, who helped coauthor the SOAP and Schema specs, among others. As an industry consultant focused on interoperability between Java, .NET, and other platforms, I get a unique first-person view of real-world interoperability problems. And as an independent speaker and mentor, I get to study the various interoperability toolkits and see how well they work.

Not everybody gets that chance, however, and unless you're a real low-level "plumbing" wonk like I am, and find a twisted joy in reading through the myriad WS-*-related specifications, things like SOAP and WSDL remain arcane, high-bar topics that seemingly nobody in his or her right mind would attempt to learn, just to get your Java code to be able to talk to other platforms. That's okay; quite honestly, you shouldn't have to. If you have to absorb every level of detail in a given programming environment in order to use it, well, something is wrong.

The JAX-WS and JAXB standards were created to help you avoid having to know all those low-level, byzantine details of the Web Services plumbing, unless and until you want to. Mark's book will help you navigate through the twisty parts of JAX-WS and JAXB because he's been there. He had to fight his way through the mess to get to the message, and now he's

going to turn around and act as your guide—Virgil to your Dante, if you will—through the rocky parts.

Because in the end, all of this is supposed to be about Hooking Stuff Together.

—*Ted Neward*
Java, .NET, XML Services
Consulting, Teaching, Speaking, Writing
www.tedneward.com