Web Services for the Access Developer

Peter Vogel

The hot new technology is Web Services. What is it? Do you care? Does it matter within Access? Peter Vogel answers these questions and more.

YOU'VE probably heard the acronym soup of SOAP, UDDI, WSDL, and (using real words for a change) Web Services. In this article, I'll describe how they all tie together and how they relate to the Access developer.

The first question asked about any new technology is, "Do you care?" The answer for Web Services is "Yes. Definitely." Access developers know more than most developers how fundamental data access is to all applications. Web Services will be just as important to creating applications. Let me repeat that: *just as important*.

XML and SOAP

Web Services refers to a set of technologies built on top of a set of industry standards, principally XML and SOAP. Microsoft has chosen to call the toolsets that implement these standards "XML Web Services," which is sort of like calling Access a "relational data database manager." By their nature, relational databases work with data; by its nature, Web Services use XML.

XML is a set of rules for creating markup languages like HTML. SOAP (Simple Object Access Protocol) is one markup language that follows the rules of XML and describes a format for messages. The SOAP format is particularly well suited for describing method calls (including passing parameters) and returning either the data generated by the function call or an error message describing what went wrong. How important is SOAP? So important that you can skip the rest of this section entirely and go directly to the section on WSDL and UDDI. SOAP is so important that you may never interact with it directly. In the same way that ASCII and other coding schemes are essential if computers are to work with something other than numbers, a messaging format is essential for calling routines. In the same way that you probably don't need to interact with your computer's coding scheme directly, you probably won't need to interact with SOAP directly.

Once a message is formatted using the SOAP tags, you'll need to send the message somewhere. The SOAP standard describes a variety of ways of sending SOAP messages, but the typical method is to use HTTP. This means that you can send a SOAP message to a Web server and receive a response back in the same way that you can request a Web page and get one back. The good thing about this is that HTTP is a very popular protocol implemented everywhere on the Internet. Using HTTP allows you to send SOAP-formatted messages to virtually anyone.

When you request a Web page, you need to specify which page you want. When you send a SOAP message, you need to specify the name of the program that will catch the message, read it, execute the function, get the result back, format the return SOAP message, and send the new message back to you. Summing up, to work with Web Services you must format a message into the SOAP format and send it, via HTTP, to some SOAP processor.

WSDL and UDDI

WSDL (Web Services Description Language) is yet another XML-compliant set of tags. The information in a WSDL document describes the format for the SOAP messages for all the callable methods in a Web Service, along with the URL to send the SOAP messages to. You can think of a WSDL document as one-stop shopping for a Web Service. Your initial reaction may be, "Great! This is all the documentation that I need to work with Web Services," but it's actually much better than that.

Web Services support tools, when fed a WSDL document, take care of formatting your SOAP message, sending the message to the right URL, catching the result, and presenting the data (or error message) back to you. You just need to retrieve the WSDL document, feed it to your SOAP tool, and start calling routines.

For me, the real power of Web Services is at the intersection of WSDL and Web Services support tools. I feed a WSDL document to a tool and can then call methods anywhere on the Internet without knowing where they are. The fact that, currently, SOAP messages are being generated under the hood is irrelevant to me.

So where do you get those WSDL documents? This is where UDDI (Universal Discovery, Description, and

Integration) comes in. UDDI is another standard (one that may end up being a proprietary standard rather than an industry standard). Unlike the previous acronyms, UDDI doesn't describe an XML standard. Instead, UDDI is set of interconnected libraries that hold WSDL documents. While a WSDL document includes the description for messages for a Web Service, a WSDL document isn't limited to that information. The WSDL specification also allows the document's creator to include information about the business that provides the Web Service, including the phone number and e-mail address of who to contact if you want more information.

There are many UDDI-compliant libraries online that support the first version of the specification. Even more libraries are coming online, implementing version 2 of the UDDI specification (still in beta at this writing). You can find a list of libraries at the UDDI home site at www.uddi.org.

Working with Web Services

After all that technology, why do you care? Being able to access any service offered by any vendor anywhere on the Internet opens a whole new world of opportunities. It's hard to imagine that there won't be some service offered in the next year that you'll feel is essential to your applications. My favorite search engine, Google, has already published the WSDL document for accessing Google as a Web Service. Would your users find it worthwhile for some application of yours to be able to generate a list of available resources on the Net for them?

How easy is it to access a Web Service? Download and install the SOAP Toolkit from msdn.microsoft.com. Then check off the Microsoft SOAP Type Library entry in your Tools | Reference list. To access Google, download the API kit from www.google.com/apis/download.html to get the WSDL file. You'll also need to get a Google Account (it's free), which will include a key to be used when accessing the service. You're now ready to write code.

The following sample code searches Google for a list of documents that mention me. The doGoogleSearch method returns an array of XML documents, including resultElement documents that contain information about the pages that match the query. Information for each page includes the page itself, the page's title, and, as shown here, the page's URL:

```
Dim spcl As MSSOAPLib.SoapClient
Dim Results As MSXML2.IXMLDOMNodeList
Dim ResultItem As MSXML2.IXMLDOMNode
Dim ResultPages As MSXML2.IXMLDOMNodeList
Dim ResultPage As MSXML2.IXMLDOMElement

Set spcl = New MSSOAPLib.SoapClient
spcl.mssoapinit "file://c:/GoogleSearch.wsdl"
Set Results = spcl.doGoogleSearch(strGoogleKey, _
    "Peter Vogel", 0, 10, True, "", _
    True, "lang_en", "", "")
For Each ResultItem In Results
  If ResultItem.nodeName = "resultElements" Then
    Set ResultPages = ResultItem.selectNodes("//URL")
    For Each ResultPage In ResultPages
        Me.lstResults.AddItem ResultPage.Text
    Next
  End If
Next
End Sub
```

And there it is: 17 lines of code to access a wealth of information. This is your future (along with Access, of course). ▲

DOWNLOAD **GOOGLE.ZIP at www.smartaccessnewsletter.com**

Peter Vogel (MBA, MCSD) is a principal in PH&V Information Services. PH&V specializes in system design and development for systems that use Microsoft technologies. Peter has designed, built, and installed intranet and component-based systems for Bayer AG, Exxon, Christie Digital, and the Canadian Imperial Bank of Commerce. He's also the editor of Pinnacle's *Smart Access* and *XML Developer* newsletters and wrote *The Visual Basic Object and Component Handbook* (Prentice Hall, currently being revised for .NET). In addition to teaching for Learning Tree International, Peter wrote its Web application development, ASP.NET, and technical writing courses, along with being technical editor of its COM+ course. His articles have appeared in every major magazine devoted to VB-based development, can be found in the Microsoft Developer Network libraries, and will be included in Visual Studio .NET. Peter also presents at conferences around the world. peter.vogel@phvis.com.