The XML files

WORDS: TIM ANDERSON **ILLUSTRATION:** PAUL PRICE

XML is the buzzword that's on everyone's lips, but pinning down what this technology is really about can be hard. *PCW* explains XML and provides two hands-on workshops that put it to real-world use

XML, XHTML, XSLT, DDF, Schema, SAX, DOM, XLINK, XPATH, SMIL, SOAP, WSDL, UDDI

INFO

STRUCTURE

XML (eXtensible Markup Language) is widely acclaimed as a revolution in computing. Charles Goldfarb, who was involved in its invention, claims it to be 'the holy grail of computing, solving the problem of universal data interchange between dissimilar systems'. It is also a handy format for everything from configuration files to data and documents of almost any type. Through SOAP (Simple Object Access Protocol) or XML RPC, you can also use it to invoke methods on remote objects, to do true distributed computing over the Internet.

So it is significant; but it's also a hard subject to pin down because it describes a whole family of technologies and specifications (see box, XML explained). Most XML is programmatically generated, and you do not need to learn all the specifications in order to benefit from it, any more than you need to learn HTTP (HyperText Transfer Protocol) to use the World Wide Web. What matters is to understand what XML does, so that you can exploit it in your own projects. >

XML EXPLAINED

XML has generated acronyms to well beyond the point of confusion. Here are the key terms, in alphabetical order:

CDATA: A section of an XML document, delimited by CDATA markers, that contains raw character data. Characters that would otherwise be interpreted as tags or other XML metadata can be included in a CDATA section, where they are treated as ordinary characters without any special meaning.

DOM: Document Object Model. This is a way of representing an XML document as a hierarchy of objects that can be manipulated programmatically.

DTD: Document Type Definition. A document that defines elements, attributes and other constraints so that XML documents of the specified type can be validated. An alternative to DTDs are XML Schema.

Namespace: A means of resolving naming conflicts by defining a globally unique name for a particular set of elements. In an XML document, you can tell which namespace an element belongs to by checking its prefix, or by looking for the default namespace. Namespace prefixes are defined with the xmlns attribute.

SAX: Simple API for XML. Originally a Java API for parsing XML. It works by raising events as each significant item in the document is being parsed.

SMIL: Synchronised Multimedia Integration Language. An XML application for multimedia and animation.

SOAP: Simple Object Access Protocol. An XML protocol for doing application messaging and remote procedure calls over the Internet.

SVG: Scalable Vector Graphics. An XML application for defining vector graphics. In theory SVG is the official web standard for vector graphics, but Macromedia's proprietary and non-XML Flash is the current real-world standard.

UDDI: Universal Description, Discovery and Integration. A public registry that enables businesses to publish information about their web services. UDDI offers a programmatic interface as well as a user interface.

Web Service: An application accessible over the Internet through an XML interface.

WSDL: Web Service Description Language. An XML format for describing XML web services, including the type definitions, messages and actions used. The WSDL document should tell applications all they need to know to invoke a particular web service.

XDR: XML Data Reduced. An early version of XML Schema, developed by Microsoft and still used by some Microsoft products. Uses the file extension .xdr.

XHTML: Extensible HTML, a reformulation of HTML as a validated XML application.

XML: eXtensible Markup Language, a specification for defining mark-up languages.

XML-RPC: XML Remote Procedure Call, an older alternative to SOAP with the advantage of being a simpler, settled standard.

XML Schema: An XML format for constraining and validating XML documents and defining data types.

XSD: XML Schema Definition. A document containing an XML Schema, with a .xsd extension.

XSLT: eXtensible StyleSheet Language Transformations. An XML application that defines rules by which an XML document can be transformed into another type of document, which might or might not be XML.

XLINK: XML Linking, or hyperlinks on steroids. An attribute-based language for linking a resource to one or more other resources, with extended information about what type of link it is and how it should be rendered. Links may be bidirectional, and can be defined outside the documents they link.

XPATH: A non-XML language for identifying specific parts of an XML document. This is not just for use in navigation, but also in querying and transforming XML. XPATH expressions are frequently used in XSLT.

XPOINTER: A non-XML language for referencing a point or a range within an XML document. Builds on XPATH.

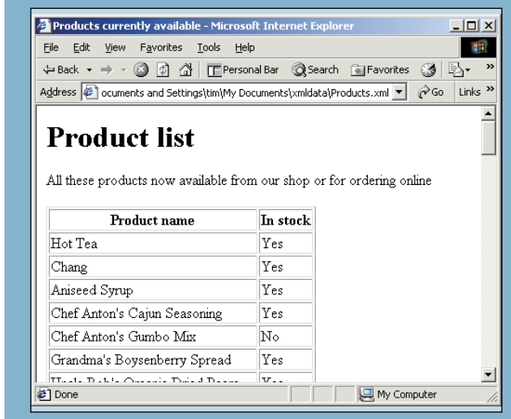
XQUERY: XML Query Language. Work in progress to define a language for querying one or more XML documents, a sort of XML equivalent to SQL (Structured Query Language). XPATH, XPOINTER, XQUERY and XSLT have considerable overlap, a fact that is recognised by the various working groups.

There are excellent online resources for XML:
www.xml.org
http://xml.apache.org
http://w3.org
http://msdn.microsoft.com/xml
http://uddi.org

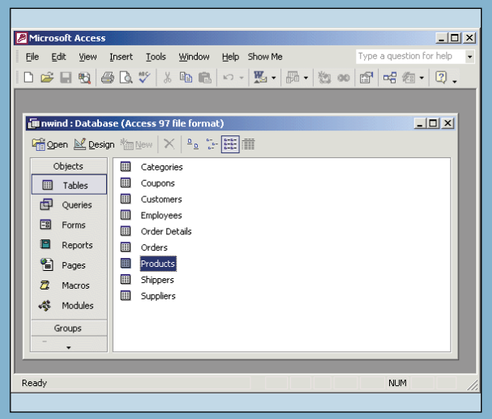
The examples in this workshop are available at www.itwriting.com/pcw/.

For books on XML, see www.wrox.com and www.oreilly.com. O'Reilly's *XML in a Nutshell* (ISBN 0596000588) is a good starting point, while Wrox's *Professional XML 2nd Edition* (ISBN 1861005059) is a comprehensive manual.

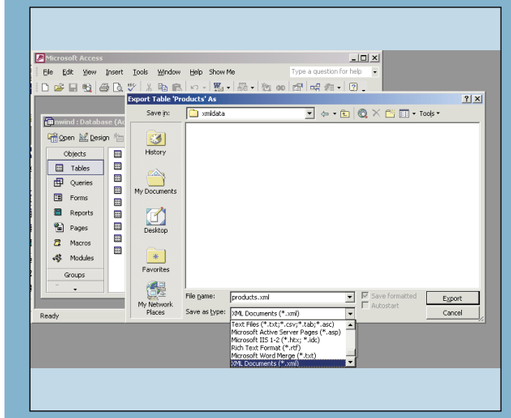
HOW TO PUBLISH DATA TO THE WEB



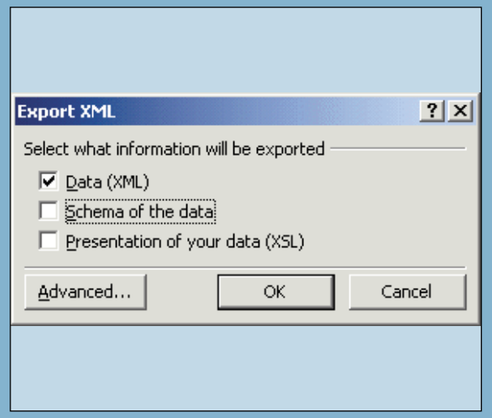
1 This workshop shows how you can use a combination of XML and XSLT to publish data to the web. It uses Access XP, although the same principles apply to any XML data file. In earlier versions of Access, you could easily write a VB routine to export data as XML. The example uses Internet Explorer, although the XSLT itself is in a standard format that will work with other XML parsers.



2 Open the sample Northwind database in Microsoft Access XP, and highlight the Products table. If you want sorted output, you might prefer to create a query sorted by, say, ProductName and use that instead. The strength of the XSLT approach is that you can use a single XML data document and present it in different ways.



3 Create a directory called xmldata (I located mine under My Documents). From the File menu, choose Export, and under Save As Type, choose XML documents. Name the file products.xml, and save it to the xmldata directory.



4 Access XP offers to create Schema and Presentation files. For this example, only the XML is needed. The XSL generated by Access is complex, includes Visual Basic, and is based on an obsolete XSLT standard. Note that to work with XML in IE, you should update the XML parser to at least version 3.0, from Microsoft's website.

Continued overleaf

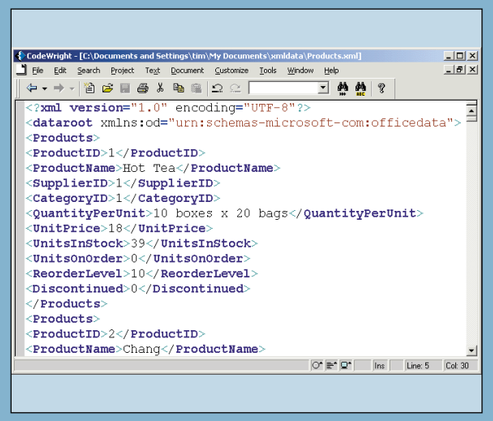
Origins of XML

In the 1970s, three guys at IBM (Charles Goldfarb, Ed Mosher and Ray Lorie) invented GML, a way of marking up technical documents with structural tags. The initials stood for Goldfarb, Mosher and Lorie. According to Goldfarb, he invented the term 'markup language' in order to make better use of the initials, so it became the Standard Generalised Markup Language and was adopted by the ISO in 1986. It is a little confusing, because SGML is not a markup language, but is a specification for defining markup languages. The application of SGML that became well known is HTML (HyperText Markup Language). HTML is an application and defines a specific set of tags suitable for web pages.

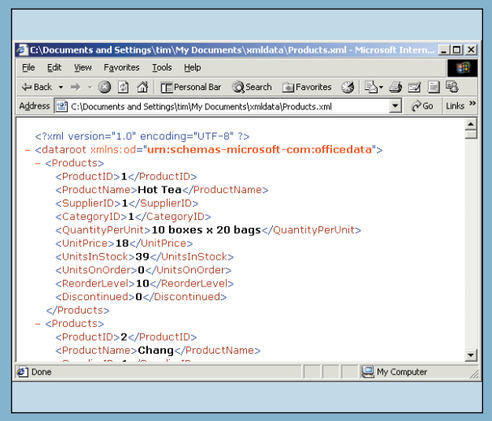
HTML has been widely deployed but as a language it is fundamentally flawed. The original thinking was to separate content from presentation. For example, the tag in a web page means 'emphasise'. It was left up to the user agent how to render that, say as bold text, or in a different colour, or with a different tone of voice in a speech reader. This type of thing does not please page designers, who want to nail down the exact appearance of a page. Therefore HTML got extended with things like <font> tags, which went right against the initial concept. Another problem area was that fierce competition between Netscape and Microsoft led to fragmentation of the standard, which remains a huge problem for web developers. Web pages began to be used for things that went wildly beyond the original concept, including multimedia, animation, online applications, ecommerce and more. Browsers also tried to be tolerant of hastily written web pages that committed crimes such as using an opening tag without a corresponding closing tag. The resulting lack of discipline became a barrier to programmatic interpretation of web content, or the use of HTML for structured data.

In a nutshell, HTML is limited, while SGML is complex. In the late 1990s a group of people, including Jon Bosak, Tim Bray, James Clark, came up with XML, eXtensible Markup

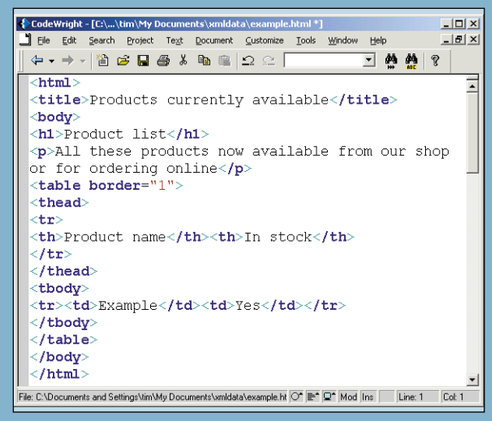
HOW TO PUBLISH DATA TO THE WEB (CONTINUED)



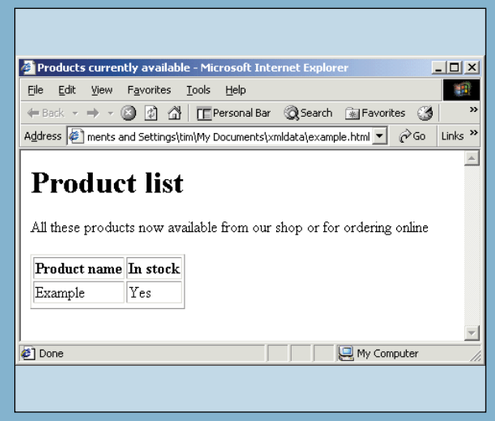
5 The generated XML file is easy to follow. An odd feature is that each record is represented by a <products> element, whereas you would expect the singular <product>. Within each <products> element, each field is represented by its own element, making this element-centric XML.



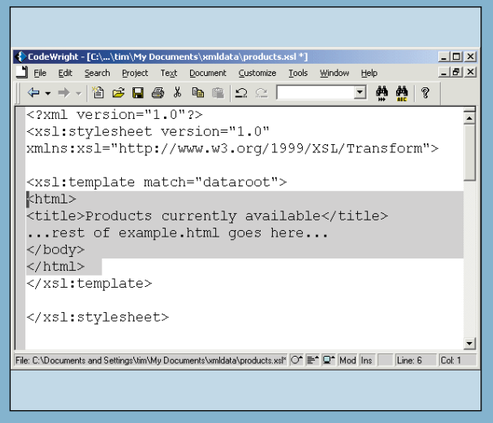
6 The same XML file in Internet Explorer shows its structure, but it is not suitable for immediate display on the web. It includes what to the user will be obscure XML code as well as showing fields that are not required. XML is almost always used as an intermediate format.



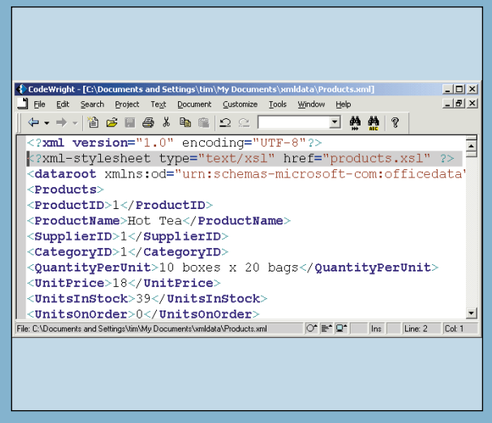
7 The purpose of an XSLT file is to transform XML from one format into another. Before sitting down to write the XSLT, it's a good idea to determine what the final output should be. In this case, I've written a short HTML file which includes a table with just two columns, product name and stock availability. The In Stock field is a yes/no type, whereas the actual data has a stock quantity. I saved the file as example.html.



8 Open example.html in a browser to check the layout. This example HTML is going to be modified to become an XLST file. There is no restriction on what the HTML can contain, although for this example I've kept it simple.



9 Back in the text editor, open up example.html and add the text shown in order to make it a valid XSLT document. To save space, I've omitted some of example.html from the illustration. The entire html ends up in an element called <xsl:template> with a Match attribute that references the dataroot element. Looking back at products.xml, that is the root element. This XSLT replaces all the xml content with the html shown. Save it as products.xsl.

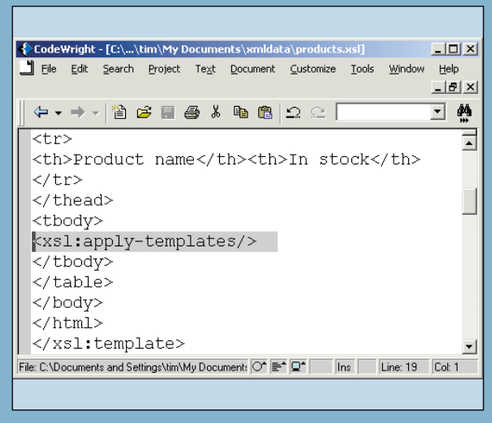


10 Products.xslt is not finished, but it is already valid. Open up products.xml and add the following as the second line:

```
<?xml-stylesheet type="text/xsl" ↙
href="products.xsl" ?>
```

(Key: ↙ code string continues)

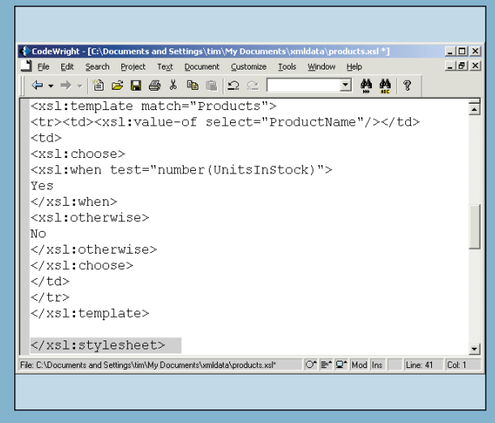
Save the file and open it in Internet Explorer. Instead of appearing as XML, it looks exactly like the HTML file in step 7 above. There are other ways to apply XSLT to an XML file, but this is the simplest.



11 The next step is to edit products.xsl, replacing the dummy <tr> element with an XSL instruction:

```
<xsl:apply-templates/>
```

This tells the processor to look for further template elements that match child nodes of the current element, and apply them.



12 The additional template matches the <products> element. It generates a table row placing the value of the product name in the first cell. Next it uses <xsl:choose> to determine the In Stock value and has an <xsl:when> element that has a test attribute representing an XPath expression. If non-zero, its content is output. If false, the <xsl:otherwise> content is output. The template is complete, and should match the illustration in Step 1.

QUICK TIP

If you are working on Windows, get the MSXML Parser version 3.0 or higher. Older versions support an obsolete version of XSLT

Language. Like SGML, XML is not a markup language, but is a specification for defining markup languages. The W3C (World Wide Web Consortium) then set about reshaping HTML as an XML application, which resulted in XHTML. That is only one small part of what XML is about. The point is that by using XML the industry can specify how to store almost any kind of data, in a form that applications on any platform can import and process.

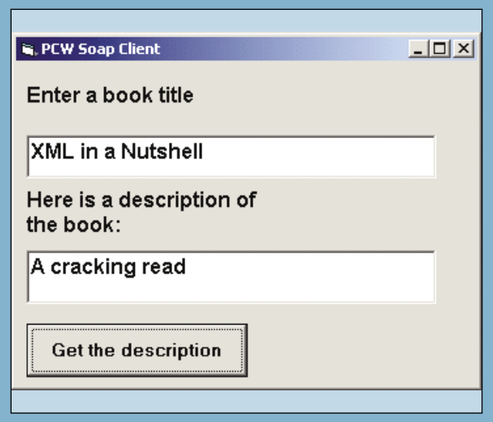
The Microsoft factor

In the mid 1990s Sun Microsystems introduced Java, with the ability to run applications securely on any supported platform. One early use was to create applets, applications designed to run safely in web browsers. But Java's warm adoption across the industry is not much to do with applets, and only a little to do with its strong and productive language features. Rather, Java helped companies such as IBM make sense of their diverse range of operating systems. Having each system running Java greatly simplifies the business of creating interoperable applications. Another example is Oracle, which uses Java-stored procedures as an ideal solution for its cross-platform database. Probably in response to the prospect of a Java-centric computing universe, Microsoft picked XML as an alternative approach to the interoperability puzzle, and became XML's greatest advocate. Unlike Java, which is controlled by Sun, XML is in the hands of the independent W3C, a fact that endeared it to Microsoft. The significance of XML to Microsoft is only now becoming clear, with the company describing its .Net initiative as 'a platform for XML web services'. Through XML, Microsoft's applications can communicate with those running on other platforms. A Java application can employ the services of a COM object (COM

QUICK TIP

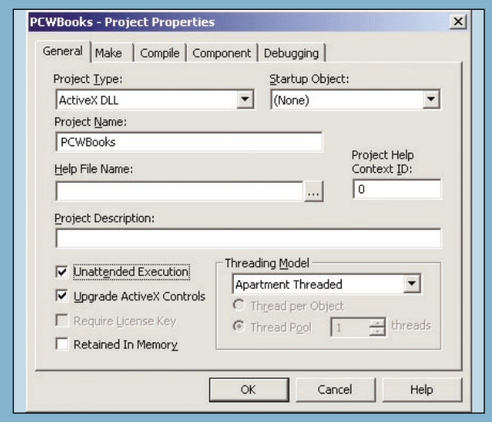
XHTML can be valid HTML. So there is no reason not to author web pages as XHTML now, even though browsers expect HTML.

HOW TO USE SOAP TO CREATE A WEB SERVICE

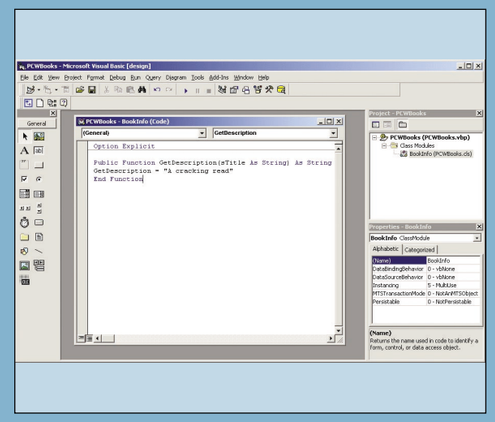


1 This project shows how to build a Visual Basic 6 application and publish it as a web service. Although very simple, the possibilities include having clients on Linux, say, invoke objects in a VB application on Windows. To try this project you need the SOAP Toolkit 2.0, VB 6.0, and Windows NT or 2000 running Internet Information Server. You can find the downloads at http://msdn.microsoft.com.

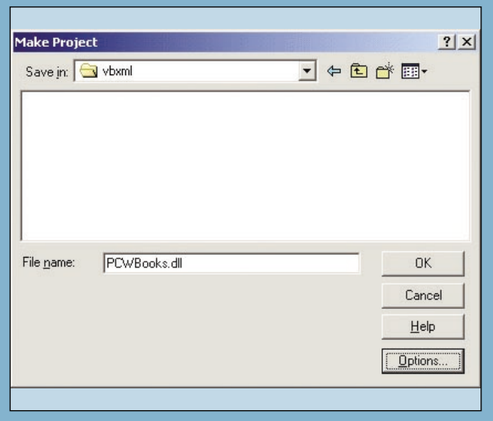
2 The starting point is to create an ActiveX DLL in Visual Basic. I saved this as PCWBooks. ActiveX DLLs can contain most kinds of VB code.



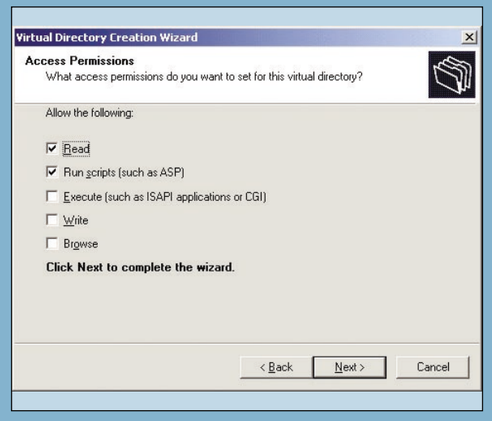
3 Open VB's Project Properties, and rename the project to PCWBooks. It is also worth checking Unattended Execution, which prevents VB from attempting to throw up messageboxes on the web server. Instead, messages get written to the event log. Otherwise, the defaults should be OK.



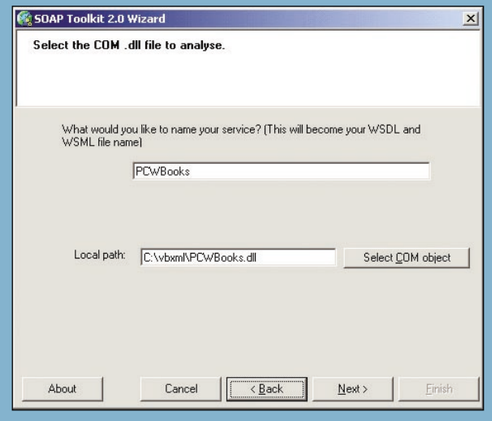
4 Rename the VB class module BookInfo, and add a method GetDescription. This takes a book title argument and returns the description. It might look this up in the SQL Server Pubs database, but for the demo, you don't need a database query. You could simply echo the argument back to the sender, or return a "Hello world" string. In the real world, though, the chances are that your web service will query a database.



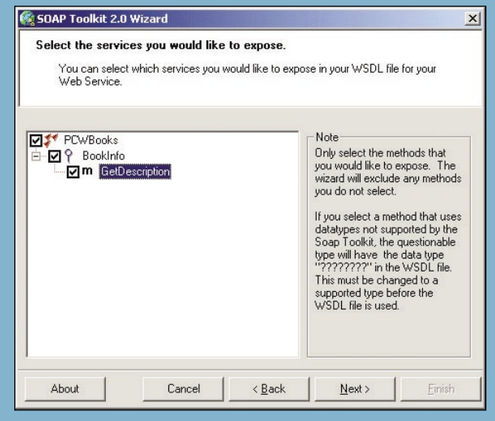
5 Make the project as PCWBooks.dll. It does not matter where you put it, although it should not be directly accessible on the web server. VB should register it for you, or you can run regsvr32 from a command prompt to put the correct details in the Registry.



6 Create a directory called SoapTest, and use the Internet Service Manager to map it to a virtual directory on the web server. The default access permissions of Read and Run scripts are sufficient.



7 Now run the SOAP Toolkit WSDL generator. This is called wsdlgen.exe, or you can find it under Microsoft SOAP Toolkit on the Start menu. Name the service PCWBooks and point it at the DLL you compiled.



8 The wizard detects what methods are published by the COM DLL and invites you to check the ones you want to publish in the web service. For this demo, check all the boxes. Here is where you discover if the wizard has problems with datatypes you have used.

Continued overleaf

QUICK TIP

XML is case-sensitive. If things are not working as expected, check for case mismatches.

being Microsoft's Windows-specific object technology), and vice versa. Hence Microsoft has been busy creating XML interfaces to its server products, such as SQL Server and Exchange.

Microsoft emphatically does not own XML, and the technology has transcended politics by virtue of its sheer usefulness. IBM is a big XML user, while listening to Sun you would think it was a Java technology. The fact that XML is important to all three companies says a lot for its bridge-building potential.

No magic

Now that XML is a buzzword, it is vulnerable to abuse by marketeers who pretend that 'save as xml' is a virtue in itself. Just because a document is in XML is no guarantee of its usefulness. For example, Microsoft Visio can save drawings as XML. These are large documents with hundreds of Visio-specific elements and attributes. Just because it is XML does not mean that AutoCAD or Adobe Illustrator can make sense of it. It might make it easier for other vendors to create an import filter; but the real benefit will come if and when Microsoft and other drawing application vendors sit down to thrash out an agreed XML standard for drawing documents. With XML, standards are everything.

The heart of XML

An XML document is a tree of nested elements, each of which can have none or more attributes. There can only be one root element. Each element has a starting and ending tag, marked by angle brackets, with content in between:

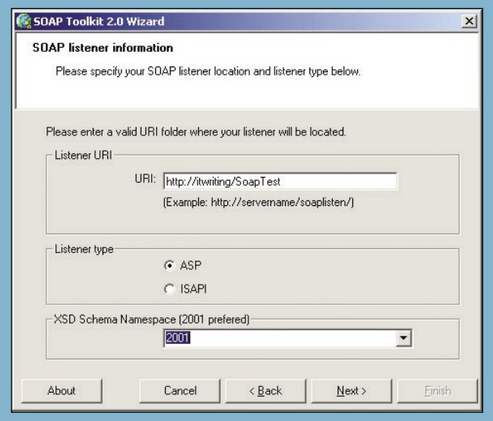
```
<element>...content...</element>
```

The content can contain other elements, or can consist entirely of other elements, or might be empty.

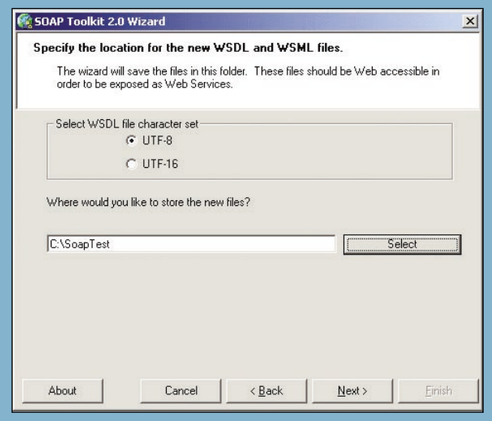
QUICK TIP

Although a namespace URIs (Uniform Resource Identifiers) looks like web addresses, it may not be an address you can view in a browser. Web addresses are convenient as they are unique.

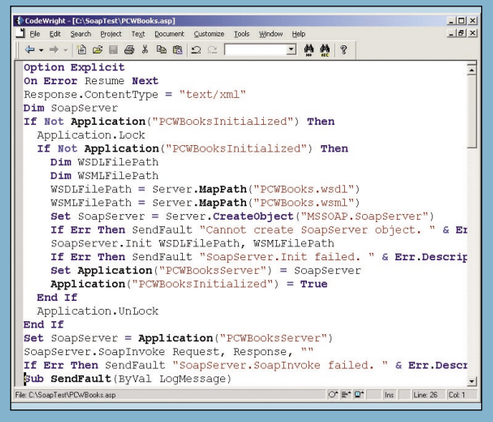
HOW TO USE SOAP TO CREATE A WEB SERVICE (CONTINUED)



9 Next you need to specify the web address of the virtual directory you created, and a listener type. Choose ASP and click Next.

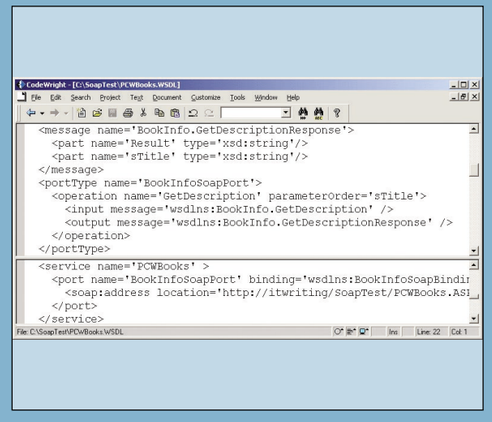


10 The wizard asks you to specify a location for the WSDL and WSML files. Enter the local directory that maps to the virtual directory you have created. WSDL is a SOAP standard, which WSML is specific to the Toolkit. Click Next and Finish.



11 The ASP file specified as a listener has been generated by the wizard. The advantage of using an ASP listener is that you have additional control over incoming requests and error reporting. The code seems long, but most of it is error checking. The key line is:

```
call soapserver.SoapInvoke(request, ↙
response, "")
```



12 Look at the generated WSDL file in an editor. This file tells applications how to call your web service. Make a note of the PortType element and its Name attribute (BookInfoSoapPort), and the Service element and its Name attribute (PCWBooks). Here I've used a split screen to show both those elements. It is worth reading through the whole file, to see how the VB method has been converted into SOAP messages. *Continued overleaf*

QUICK TIP

Check the W3C site for the status of XML specifications. Anything described as a working draft may change substantially. Once it has the status of a Recommendation it is a reliable standard.

Attributes are named values given in the start tag, with the values surrounded by single or double quotations:

```
<element attribute1="value1" ↙
attribute2="value2">
```

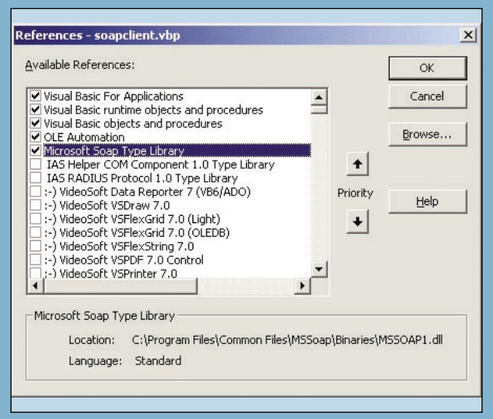
(Key: ↙ code string continues)

This is the essence of XML, and it's nice and simple. Here's an example that will look familiar:

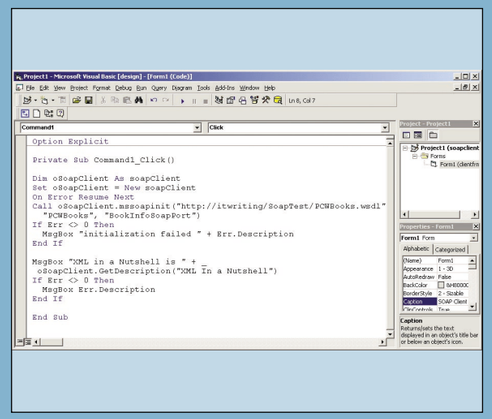
```
<html>
<body background= "mypic.gif">
<h1>HTML or XML?</h1>
</body>
</html>
```

This file can be saved either with an HTML extension, or as XML, and opened in Internet Explorer. It is valid as either and illustrates the close relationship between the two languages.

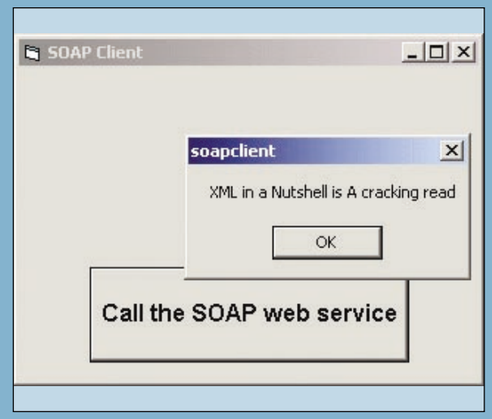
HOW TO USE SOAP TO CREATE A WEB SERVICE (CONTINUED)



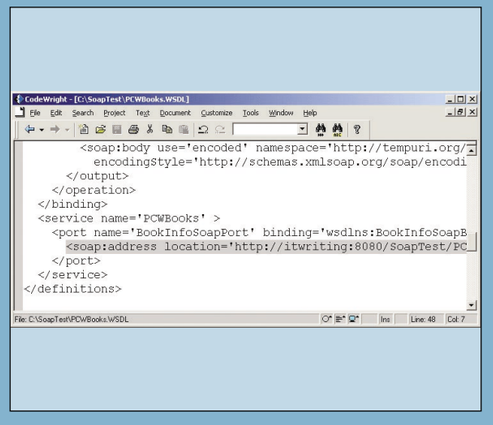
13 To try out the service, start a new VB project. I called it SoapClient. You can do this on another machine on your network, to demonstrate distributed computing via SOAP, or on the same machine. In Project References, set a reference to the Microsoft SOAP Type Library. Where the code reads "itwriting", substitute localhost or the name of your server.



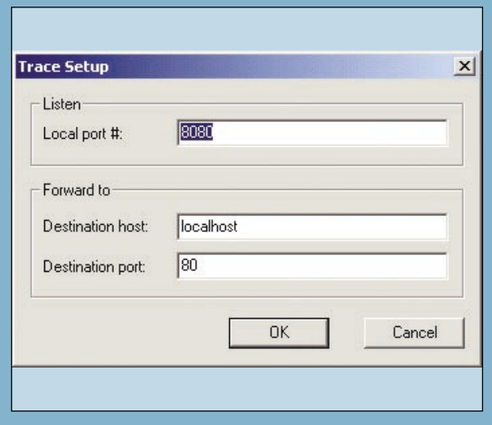
14 Here is the VB code to call the web service. Note that the client code is really very simple. The only line with any challenge is the mssoapinit, which takes the URL of the WSDL file, the service name, and the port name, which you can read from the WSDL file itself. Then you can call the methods of the web service as if they were methods of the SoapClient object.



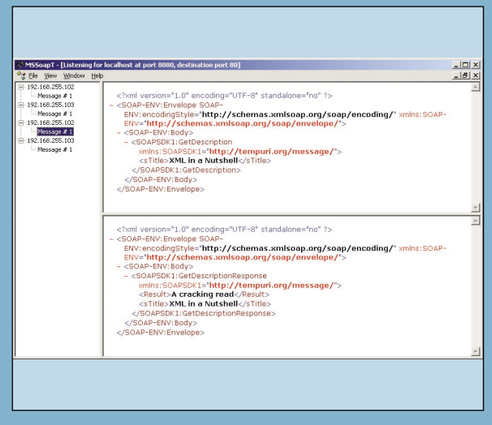
15 The VB SoapClient at work. Note that VB clients must have the MS Soap binaries installed. A module to install these is available on the Microsoft SOAP developer site. You should also be able to create clients in other languages and on other platforms. In practice, expect problems as different vendors iron out incompatibilities.



16 The SOAP Toolkit Trace utility is a great way to troubleshoot. To use it, open up the WSDL document for the service you want to trace. Find the soap:address element and edit its location attribute to use a different port, port 8080, as in the illustration. Save the amended WSDL document.



17 Run the Trace Utility from the SOAP Toolkit. Accept the defaults as shown, for a new formatted Trace. This assumes that the trace utility is running on the server.



18 With the Trace Utility running, try out some calls against your web service. The utility shows both the sent message and the response, which is fantastically useful for troubleshooting. If messages do not show up at all, then the problems may be more fundamental (for example, the client could not find the WSDL file at all).

THE PARTS OF AN XML DOCUMENT

Declaration
This line declares that what follows is an XML document and specifies the version, currently always 1.0, and optionally the encoding or character set, and whether the document stands alone or requires an external schema file. Example:

```
<?xml version="1.0" encoding="utf-8" ?>
```

Another type of declaration indicates the document type, either by reference to an external DTD or by including it inline:

```
<!DOCTYPE pcwdoc SYSTEM
```

Comment
XML comments appear between special delimiters:

```
<!-- a comment -->
```

Processing instruction
Additional information for applications parsing or processing the document. It may even be a script. Uses <? ... ?> delimiters:

```
<?xml-stylesheet href="mystylesheet.css"
```

Namespace
The xmlns attribute either defines a default namespace for everything within that element, or a prefix that resolves name conflicts. The full name of a namespace must be unique, so it often looks like a web address although it is just a name. In the example, everything within the mydoc element belongs to the default pcwexample namespace unless it is prefixed "con:", in which case it belongs to the pcwcontact namespace:

```
<mydoc
xmlns="http://itwriting.com/2001/pcwex

xmlns:con="http://itwriting.com/2001/p
```

Elements and attributes
The basic building blocks of XML. Elements are hierarchical, must have start and end tags, and optionally include attributes in the start tag. Empty elements are those that have no content, and may use a combined start and end tag like <hr/>.

```
<magazine title="Personal Computer
```

Character data
Plain text that forms the content of elements. Elements can also include other elements. Special characters that would otherwise be interpreted as markup or hard to represent are included with either character references, such as — for a long dash, or entity references like < for the "<" character. In the first case the number is Unicode reference, in the second a named entity from the built-in list or defined in a DTD.

CDATA section
Raw character data that will not be interpreted as XML markup. Delimited by:

```
<!CDATA[ ... ]]>
```

There are a few other fundamentals in XML, such as processing instructions and namespaces, but elements and attributes are the heart of it. If you are familiar with object-oriented programming, think of elements as objects and attributes as properties. When designing XML applications, it's hard to decide what should be an element and what should be an attribute. For example, why not have:

```
<body>
<background>mypic.gif</background>
</body>
```

To some extent this is a matter of taste. Some XML is more element-centric, some more attribute-centric.

Validating XML

A well-formed XML document conforms to rules, such as having only one root element, all start tags have matching end tags, elements may not overlap, and so on. You can make up elements and attributes as you go along, and still end up with a well-formed document. It is usually more useful to validate the document according to an agreed schema, of which XHTML is an example. The schema defines what elements may appear, what attributes they may have, and other constraints such as what is optional and what is required. The standard way to do this is with a DTD (Document Type Definition). DTDs have limitations and are not themselves XML documents, so more recently a more powerful alternative called XML Schema has been agreed. A key advantage of XML Schema is support for strong data types, such as string, float, Boolean, decimal and dateTime. XML Schema is the future, but DTDs are still valid and will be around for a long time. Valid XML is both well-formed and validated by conformance to a specified schema. PCW