Understanding Web Services

If you ask someone if they are familiar with Web services, they usually pause before answering. They know what the Web is and they certainly know what services are, but they are not sure whether you are asking a generic question or asking about a specific technology. Apart from the generic-sounding name, Web services represent a new architecture for creating applications that can be accessed from a different computer. The purpose of this hour is to improve your understanding of the topic at an executive summary level.

In this hour, you will learn

- The definition of Web services
- The promise of Web services
- The key specifications that comprise Web services
- The tools that are commonly used to create Web services

- The challenges that face Web services developers
- The organizations that manage the specifications that Web services are built upon

Understanding What Web Services Are

NEW TERM A Web service is a software application that can be accessed remotely using different *XML*-based languages. Normally, a Web service is identified by a *URL*, just like any other Web site. What makes Web services different from ordinary Web sites is the type of interaction that they can provide.

Most Web sites are designed to provide a response to a request from a person. The person either types in the URL of the site or clicks on a hyperlink to create the request. This request takes the form of a text document that contains some fairly simple instructions for the server. These instructions are limited to the name of a document to be returned or a call to a server-side program, along with a few parameters. Figure 1.1 shows this process.

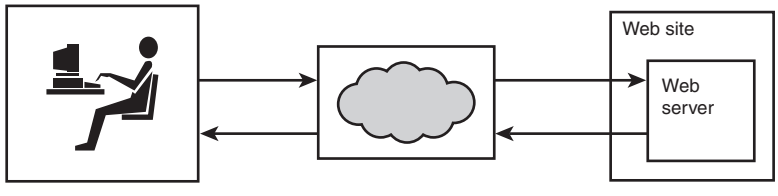

FIGURE 1.1
A browser interacts with a Web server to make requests.

NEW TERM A Web service is similar in that it is accessed via a URL. The difference lies in the content of what is sent in the request from the client to the service. Web service clients send an XML document formatted in a special way in accordance with the rules of the *SOAP specification*. This specification is the topic of Hour 9, "Exchanging Messages with SOAP."

A SOAP message can contain a call to a method along with any parameters that might be needed. In addition, the message can contain a number of header items that further specify the intent of the client. These header items might designate what Web services will get this method call after the current service finishes its work, or they might contain security information. In any case, the complexity of the SOAP message far exceeds the complexity that is possible using only a browser. Figure 1.2 shows this process graphically.

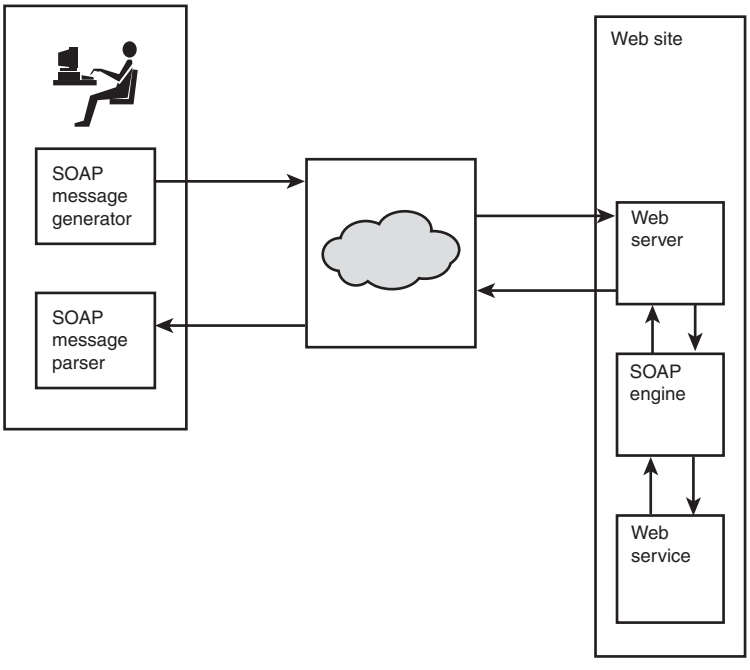

FIGURE 1.2
A client interacts with a Web service via a Web server such as Apache Tomcat or MS Internet Information Server.

The Great Promise of Web Services

NEW TERM Most of the enthusiasm surrounding Web services is based on the promise of interoperability. The Web services architecture is based on sending XML messages in a specific SOAP format. XML can be represented as plain *ASCII* characters, which can be transferred easily from computer to computer. The implications of this are significant:

- It doesn't matter what kind of computer sends the SOAP message or on what operating system it is running.
- It doesn't matter where in the world the client is sending the message from.
- It doesn't matter what language the software that the client is running on was written in.
- There is no need for the client to know what type of SOAP processor is running on the server.

In short, Web services are the Holy Grail of computing. Every software application in the world can potentially talk to every other software application in the world. This communication can take place across all the old boundaries of location, operating system, language, protocol, and so on.

If we take off the rose-colored glasses for a minute, however, we will see that there is much work to be done before this promise is realized. Although the current specifications provide us with a solid beginning, no one believes that they are adequate for every situation. Areas such as security, transaction support, and business process execution are being addressed, but they are not yet incorporated into the SOAP specification.

The Key Components

Web services transactions take place between components. You can either program these components yourself, download them from open source software foundations such as Apache, or purchase them from commercial vendors such as Microsoft or IBM. There is no requirement that you obtain all the components that you use from a single vendor; you can write some, download others, and purchase still more.

For a Web service transaction to complete successfully, all of the components involved in processing the transaction must behave in ways that the other components expect them to. Given the differing vendors involved in the creation of the components, you might expect that they would have a lot of problems interacting. Although Web services interoperability is difficult to attain, it can be remedied by the creation of high-quality standards and a religious adherence to these standards by every programmer and vendor involved. At this writing, the following are considered the core Web services standards:

- **SOAP**—SOAP originally stood for Simple Object Access Protocol. But SOAP is now considered a specification name and not an acronym. SOAP is a specification that defines an XML grammar for both sending messages and responding to messages that you receive from other parties. The goal of SOAP is to describe a message format that is not bound to any hardware or software architecture, but one that carries a message from any platform to any other platform in an unambiguous fashion.

  NEW TERM

  The SOAP standard contains two parts: the header that carries processing instructions and the body that contains the *payload*. The payload contains the information that you want to send. The two types of SOAP messages are documents and *Remote Procedure Calls (RPCs)*. The payload of a document message is any XML document that you want moved from one computer to another. An RPC is a method call that is intended to be executed on the Web service's computer. The RPC message performs the same function as an ordinary method call in an ordinary programming language. The difference is that this call can take place over the Internet. SOAP is the subject of Hour 9.

- **Extensible Markup Language (XML)**—Extensible Markup Language (XML) is the language that all the Web services Languages are built on. XML is a tool for constructing self-describing documents. In fact, XML is more of a meta-language than a language in that it is used to create grammars. These grammars are described in XML schemas that specify the tags that are allowed and the relationships between the elements defined by these tags. SOAP, WSDL, and UDDI are all XML-based grammars. XML is the subject of Hour 7, "Understanding XML."
- **Hypertext Transport Protocol (HTTP)**—Hypertext Transport Protocol (HTTP) is a standard that precedes the advent of Web services. It was developed to facilitate the transfer of requests from a browser to a Web server. Web services takes advantage of the existence of this mature protocol to move SOAP messages and WSDL documents from one computer to another.

  Newer versions of SOAP describe how other transport mechanisms such as FTP, SMTP, and JMS can be used to perform this same function. At the time of this writing, however, the vast majority of Web services are built on HTTP.
- **Web Services Description Language (WSDL)**—Web services Description Language (WSDL) is a specification that tells us how to describe a piece of software in terms of the method calls that it responds to. These methods are described in an abstract way that is independent of what programming language the actual service is written in or what computer and operating system it runs on. In fact, you can port an application from a personal computer to a mainframe, and the abstract portion of the WSDL description will not change (assuming that the port and protocol don't change when you port them).

  The WSDL also contains a concrete section in which the various details of how to actually make a connection to the service are stored. If a Web service could be accessed using HTTP, FTP, or SMTP, you would find three entries in the concrete section—one for each service.
- **Universal Discovery Description Integration (UDDI)**—The Universal Discovery, Description, and Integration (UDDI) specification describes how a potential customer of a Web service could learn about its capabilities and obtain the basic information needed to make the initial contact with the site. Normally, this contact includes a download of the WSDL.

  UDDI registries can be public, private, or semiprivate. A public directory allows everyone on the planet to examine the information that you post in the registry. A private registry exists behind the firewall of your organization and is only accessible by members of your organization. A semiprivate registry is open only to a limited number of outsiders such as you best trading partners.

Tools and Vendors

Another aspect of Web services that is sometimes hard to grasp is what tools you need to create them. Often, the software tools that you have worked with have come from a single vendor. Microsoft sells Visual Basic and Visual C++, along with a framework and an Integrated Development Environment (IDE). Even though Sun Microsystems allows you to download Java at no cost, specific vendors such as Borland and IBM sell you an IDE.

Web services can be developed using any programming language that supports sockets. You could write a client that generated its own SOAP messages. You could then open a socket and send the message to a Web service listening on that socket. That Web service could do its own SOAP parsing, make its own method calls, write to its own logs, and prepare its own SOAP response message. Finally, it could open a response socket and send the return message to the client, who could then display the results.

Although you could do this, it would be a bit like forging your own shovels and rakes at home before going out into the garden to work. A quick trip to the local hardware store could save you a lot of time and shorten your schedule quite a bit.

Web services tools are numerous because so many software vendors have bought into its promise. In addition, a few startups have also written Web services development tools. The result is a wide variety of choices. You not only get to choose between vendors and IDEs, but you also get to determine what level of tool you want to employ.

Tools range from special Java classes that know how to create SOAP messages to full-blown development environments that remind you of a high quality Fourth generation language tool. The following is a list of the tools that we will cover in this book:

- **Apache Axis**—Apache Software Foundation coordinates the creation of open source projects. One of its projects is a SOAP engine that is normally used with its Tomcat server. Axis is the subject of Hour 13, “Creating Web Services with Apache Axis.”
- **Java**—Sun Microsystems has created a set of optional packages that can access UDDI registries, generate WSDLs, and so on. These packages are the subject of Hour 14, “Creating Web Services with Java.”
- **Visual Studio .NET**—Microsoft’s new way to create Web services is to use this product in conjunction with any one of the Visual Studio languages such as Visual Basic, Visual C++, or C#. Hour 15, “Creating Web Services with .NET,” shows an example that creates a Web service using Visual Basic .NET.

- **Web Services .NET Clients**—Clients created using .NET that can interact with any Web service. Hour 16, "Creating .NET Web Service Clients," covers this.
- **BEA WebLogic Workshop**—BEA is a leading *J2EE* vendor that has created a user-friendly way to create Web services by using an elaborate IDE. Hour 17, "Creating Web Services with BEA WebLogic Workshop," covers this tool.
- **IBM WebSphere Studio Application Developer (WSAD)**—IBM has made the creation of Web services a part of its comprehensive package called WSAD. We cover the creation of Web services with WSAD in Hour 18, "Creating Web Services with IBM WebSphere."
- **Other Important Products**—Many lesser-known companies have quality entries in the Web services development tool market. Hour 19, "Creating Web Services with Other Toolkits," covers Iona XMLbus, The Mind Electric GLUE, PocketSOAP, and SOAP::Lite.

In Hour 20, "Comparing the Different Web Services Tools," we compare the different features of these packages and talk about their strengths and weaknesses.

Who Manages the Web Services Specifications

Earlier in this hour, we covered each of the specifications that form the foundation that Web services are built on. In each instance, we mentioned what governing body was responsible for managing the decision-making process that leads to the publication of a new version of each specification. This seems like quite a confusing arrangement at first because no one authority has the final word.

This lovely confusion is a direct result of the distrust that we have for software vendors and their distrust of each other. Many of us have felt like hostages to one vendor or another. Vendors, by their very nature, are beholden to their stockholders who expect them to earn a profit. This otherwise noble goal can lead to bad behavior; however, when a vendor begins to monopolize a technology to lock in its users, decisions sometimes seem to be made on how best to restrict the user's freedom to defect to another vendor's product.

Vendors also tend to play roughly with each other. If they control one piece of the software suite, they tend to use it to expand their influence into other areas. This can perceived as "fighting dirty" by the competition. As a result, vendors often cooperate or fail to cooperate with each other based not on what is best for the industry, but what will irritate the other the most.

Having fought dozens of wars over the past two decades, users and vendors alike have been looking for an alternative to the madness of cutthroat competition. In the abstract, they all realize that a set of specifications written to be of the greatest benefit to the user would level the playing field. As a result, they have begun to cooperate with a group of nonprofit consortiums whose sole purpose is to draft and publish specifications.

The World Wide Web Consortium

The most important of these organizations is the World Wide Web Consortium (W3C). On its Web site, `www.w3.org`, the W3C states the following:

> The World Wide Web Consortium (W3C) develops interoperable technologies (specifications, guidelines, software, and tools) to lead the Web to its full potential. W3C is a forum for information, commerce, communication, and collective understanding.

The W3C manages the SOAP, WSDL, XML, XML Schema, and HTTP specifications, among others. In addition, the W3C manages the WS-Architecture document, which is currently in a draft status. This document is attempting to establish a formal definition of Web services.

OASIS

Another important organization in the Web services world is the Organization for the Advancement of Structured Information Standards (OASIS). The OASIS Web site, `www.oasis-open.org`, states the following:

> OASIS is a not-for-profit, global consortium that drives the development, convergence, and adoption of e-business standards.

OASIS manages UDDI, WS-Security, and SAML specifications, among others.

WS-I

The Web Services Interoperability Organization (WS-I) is a fairly new organization that has finally succeeded in getting the last big holdout, Sun Microsystems, to join. The WS-I Web site, `www.ws-1.org`, states that the mission of the WS-I is this:

> WS-I is an open industry organization chartered to promote Web services interoperability across platforms, operating systems, and programming languages. The organization works across the industry and standards organizations to respond to customer needs by providing guidance, best practices, and resources for developing Web services solutions.

The WS-I published profiles, testing tools, and sample applications that guarantee, as nearly as possible, that the different standards work together. The profiles are a versioned set of specifications that have been shown to be able to work together successfully.

The Specification Process

The process of creating a new specification is a bit like visiting the proverbial sausage factory—sausage is easier to eat if you don't watch it being made. Normally, a single company discovers that it needs a certain specification to go forward. It normally gets together with a couple of friendly companies and creates a draft. The industry is generally skeptical of vendor-created standards, so the creators of the draft normally try to enlist one or more of their traditional enemies to join with them in publishing the specification. They do this because enemies won't join in publishing a worthless or self-serving specification.

With an enemy on board, the growing group of supporters tries to get an established organization such as W3C or OASIS to take over the management of it. If they succeed, they get a credibility boost. If they are unsuccessful, they normally form a single specification organization to manage the feedback and publish the specification. UDDI.org was formed for this purpose. This new organization tries to get other organizations, especially more traditional enemies, to join while simultaneously working to improve the specification.

Victory is achieved when enough organizations have joined on that there is universal buy-in. Often an organization such as W3C or OASIS will agree to take over a specification after it has gained more acceptance. OASIS took over UDDI after it had proven its viability by signing up supporters. After a specification is moved into a standards organization, it is assigned to a committee.

Commercial organizations often assign internal staff to donate some of their time to these committees. Representatives of large organizations are often voted to chair the committees in order to maintain their support for the specification. Other organizations, who are interested in making sure that the resulting drafts address issues important to them, assign staffers to work on the creation of the drafts. When the committee reaches a consensus that the draft is ready for publication, the draft is published and set aside for a period of time for comment. Anyone can comment on or take issue with the published draft. The committee reviews this feedback, and changes are made if the concern is deemed valid. After the time for comments expires, the committee publishes the specification and calls it a standard (OASIS) or a recommendation (W3C).

After a standard or recommendation is published, vendors can choose to implement the changes and additions in their products. Failure to do so, however, might lead to a loss of prestige in the marketplace, which could, in turn, cause a product to be rejected by potential buyers.

Summary

The promise of Web services is great, but the obstacles are formidable. In our previous scenario, we were exchanging information about an address. If this information had been sensitive, we would have had to use SSL to encrypt the entire transaction because, at present, no security information appears in the SOAP specification.

If there had been any incompatibilities between the versions of SOAP, UDDI, HTTP, and WSDL that we were using, the transaction might not have completed successfully. Web services is also lacking a transaction model to allow for a rollback in case of failure.

Although there is much to be excited about, there is much work to be done. The good news is that we don't have to wait on any one vendor to decide the direction for us. Anyone who wants to become involved in the specification process is welcomed to join in and push.

In this hour, we looked at what a Web service is. Following that, we covered the various standards that form the foundation of Web services. We listed briefly the commercial products that have achieved some popularity in the marketplace.

Next, we looked at how all the different specifications support one another in completing a typical transaction. Finally, we looked at who controls the specifications that govern Web services and how these specifications came to be.

Q&A

Q **What are the key specifications that compose Web services?**

A They are XML, HTTP, SOAP, WSDL, and UDDI.

Q **Who creates new or revised Web services standards?**

A Anyone who wants to can create and submit draft proposals. In practice, this is normally done by groups of companies with a vested interest in the progress of Web services standards.

Workshop

The Workshop is designed to help you review what you've learned, and begin learning how to put your knowledge into practice.

Quiz

1. What is the promise of Web services?
2. What are the core standards that compose Web services?
3. What are some of the challenges that are holding back Web services?
4. Who controls the Web services specifications?

Quiz Answers

1. The promise of Web services is interoperability across hardware, operating system, geographic, and programming language boundaries.
2. The core standards are HTTP, XML, XML schema, SOAP, WSDL, and UDDI.
3. The current standards don't address issues such as security, transaction management, or business process execution. In addition, different versions of the same standard can cause transactions to fail.
4. The specifications that govern Web services are controlled by the user community through standards organizations such as W3C and OASIS.

Activities

1. Visit the W3C Web site at `www.w3.org` and read through the recommendations that are published such as SOAP 1.1 and drafts such as WSDL 1.2.
2. Go to the OASIS Web site at `www.oasis-open.org` and review the UDDI 2.4 specification.
3. Visit `www.apache.org` and read about Apache Axis. Download it if you plan on working the examples later in the book.