

Chapter 1

Introducing Web Services

In This Chapter

- ▶ Discovering Web services
 - ▶ Demystifying the Microsoft .NET platform
 - ▶ Unleashing the technologies surrounding Web services
-

Web services promise to be the next major frontier in computing. Up until the advent of Web services, interoperability and integration (the exchange of data among computer systems) were extremely limited or cumbersome. Prior to Web services, limited integration took place with numerous technologies, vendors, obstacles, and formats that prevented the sharing of data. But then Web service technology came along and changed all that.

Web services is a promising new technology that solves virtually all the problems that have existed in traditional distributed computing. Web services are programmable and platform-independent. All you need is network connectivity that understands how to transmit HTTP requests and you're in business.

Microsoft has generated an exceptionally comprehensive platform for designing, developing, testing, and deploying Web services-based applications, but has taken this concept one step further and made its tools language-independent as well. In this chapter, I give you an overview of Web services: why they are needed, what they do, how they are used, the players involved, and the technologies involved.

At this time, I advise you to grab a cup of coffee, sit back, take off your shoes, prop up your feet, and dive right in!

What in the World Are Web Services?

No doubt you have heard of Web services. You can't exactly avoid them if you have any job function in information technology (IT) today. If you are a CEO, CTO, project manager, software architect, or developer, you have at least listened to some of the hype about Web services.

10

Part I: Web Services Overview

If I had to give you an oversimplified definition of the term *Web services*, I would say that Web services encompass the technology that's used in allowing data to be transmitted across the Internet by using a familiar programming methodology. To avoid any confusion about what Web services transmit over the Internet, I want to emphasize the fact that *only data* is transmitted using Web services technology. Web services do not have a visual interface, such as text boxes, radio buttons, and the like.

For example, a Web service may be offered in a B2B (business to business) scenario whereby Company A provides a currency conversion Web service and Company B, in turn, uses this Web service to provide the currency conversion functionality to its customers. The Web service offered by Company A can also be used by Company C in a different capacity. For example, Company C may combine Company A's Web service with other functionality and offer it as a Web service to other companies. In both scenarios, specific functionality is developed and made available as a programmable Web service that can be accessed by other companies over the Internet.

Of course, Web services are technically a lot more complicated than my oversimplified definition states. Try asking ten of your co-workers what a Web service is; you're likely to get ten different answers. So, who's right? Most likely, all of them. Everybody has their own definition because they describe how it affects their own personal lives.

Really, you need to decide for yourself (with the help of this book) what Web services are and how they affect your business. The term *Web services* means something different to each person, depending on his or her interest. Executives are likely to not understand exactly how the Web services technology can simplify development, but they are going to understand the high return on investment (*ROI*). Conversely, developers and architects will likely understand that the Web services technology can help shorten development time and drastically make connectivity easier, but they won't grasp the immediate benefits of high ROI. Typically, developers and architects simply want to implement the coolest solution and use the latest and greatest technologies. With Web services, everyone can have his or her wish!



Web services is a technology for transmitting data over the Internet and allowing programmatic access to that data using standard Internet protocols. The term *Web service* is not used to represent a company that simply offers services on the Web, such as a banking Web site. Although this company that offers a service over the Web, it doesn't necessarily make its service available by using a programmatic interface that allows two applications to be integrated. In fact, a Web service allows a developer to include functionality into a program without needing to "reinvent the wheel" and without needing to know anything about the business or complexity of the Web service that he or she is using.



Web services as a technology is strictly related to the transmission of data over the Internet or your company's intranet (local network). ROI comes into play when you start analyzing what Web services can do for your organization or how effective they are to implement or create. I discuss how Web services affect your business throughout this book, but specifically in this chapter and in Chapter 2. I show you how to create Web services in Chapter 6.

Microsoft supports Web services with something called the *.NET Framework*. I cover the .NET Framework in the “Parts is parts: Putting the .NET pieces together” section, located later in this chapter. I discuss how the .NET Framework impacts your organization in Chapter 2. Together, all tools and technologies that support Web services are collectively referred to as the *Microsoft .NET platform*.

What Web Services Do for You

Web services is a broad term that represents all the technologies used to transmit data across a network by using standard Internet protocols, typically HyperText Transfer Protocol (HTTP). An eXtensible Markup Language (XML) format is used to represent the data, which is why Web services are sometimes known as *XML Web services*.

You can think of an individual Web service as a piece of software that performs a specific task (also known as a *function*), and makes that task available by exposing a set of operations that can be performed (known as *methods* or *Web methods*) with the task. Additionally, each of the methods exposes a set of variables that can accept data passed into the method. These variables are known as *parameters* or *properties*. Together, the properties and methods refer to a Web service's *interface*. For example, Company A creates a Web service that provides currency rate functionality, which may expose a method called `GetRate`. Company B is then able to pass a parameter called `CountryCode` into the `GetRate` method. The `GetRate` method takes the `CountryCode` parameter, looks up the appropriate currency rate in a database, and returns the rate back to the program that requested it.

In this example, which database did Company A use to access the currency rate information? What was the name of the database server? What communication mechanisms and security mechanisms were used to access the database server? The answer to all of these questions is “I don't care.” The beauty of a Web service is the concept of *encapsulation*. Encapsulation allows the complexity of retrieving the actual currency rate to be completely self-contained within the company that created the Web service (Company A). The only thing that Company B knows is that they called a Web service to get a currency rate and it was given to them.

12 Part I: Web Services Overview



A little something about ASP

If you're using the Microsoft platform, the Web server used to host Web services is, of course, Internet Information Server (IIS). If you've done any Web development at all on the Microsoft platform, you know IIS intimately. IIS is used to host Active Server Pages (also known as *ASP*). Active Server Pages were developed to allow dynamic programming and scripting within the Web execution environment. However, ASP presented some technical issues and, just like all software, grew into

something else: ASP.NET. ASP.NET has countless advantages over ASP; these advantages are covered in Chapter 2.

Where does ASP.NET come from? How does it get installed? The answer to both questions is the .NET Framework. I briefly discuss the .NET Framework later in this chapter (in the section "Parts is parts: Putting the .NET pieces together"), and I discuss it in even more detail in Chapter 2.

Web services are made possible by placing the programs, or applications, on a Web server, such as Microsoft Internet Information Server (IIS). Because the application resides on a Web server, it can be called, or *invoked*, from any other computer on the network by using HTTP. The Web service provides seamless distributed computing across the entire network, as long as both sides know how to use a Web service. If you've been in business for a while, you probably know that providing distributed services has always been a challenge in the past.



One major advantage of invoking or creating Web services over HTTP is that if the Web server is on the Internet, the network administrators on both ends of the data transmission don't have to open any additional ports in their firewalls. All transmission of data is sent across port 80 (typically) by using HTTP. Port 80 is always open in a firewall because it is the same port used to browse the Internet. The fact that the network administrators don't need to open additional ports means that you face virtually no additional security risk in using Web services.

Another major advantage in Web services is that (because Web services conform to open standards) a Web service written on one platform (such as the Microsoft platform) can call another Web service written on another platform (such as Linux).

Because of their innate flexibility, Web services make the notion of *software as a service* a real possibility. And because Web services provide integration between two systems, *software as a service* refers to the possibility of not

having to install software on workstations or servers, but rather, being able to use it from across the Internet. In fact, Web services can change the way you use all your computing resources by doing the following:

- ✓ **Save hassle:** Imagine you need to install Microsoft Office. If Microsoft decides that they want to make Office available as a Web service, you don't have to go out, purchase the software, and install it out of the box to all the computers in your network. Instead, you can get the full functionality of that piece of software across a Web interface without any installation at all.
- ✓ **Save money:** Imagine renting or leasing software instead of buying it. You can "break your lease" when and if the applications aren't working for your organization, which can save you lots of money.
- ✓ **Stay ahead of the game without even trying:** Imagine not having to keep up-to-date with the latest version. The latest version is always available from the vendor who provides the software as a service.

All of these things are possible when using software as a service. What you're really doing when you use software as a service this way is *outsourcing* functions that you used to perform within your organization. Many businesses already outsource software products as a service, including my company, Transport:80. For more information, visit the Web site at www.transport80.com. Microsoft (and other companies) may, in the future, make their products available as a service.

Using Web Services on the Microsoft .NET Platform

To understand the benefits of Web services on the Microsoft platform, you have to examine the problems of distributed computing in the past. *Distributed computing* is the notion of having multiple computers in different locations individually provide computing power for the purpose of processing information. To help illustrate distributed computing in action, consider the following scenario. You develop a Web-based application that enables your customers to do the following:

- ✓ Create and manage user accounts
- ✓ Pay bills (this includes credit card processing)
- ✓ Look up prior bills for the past three years
- ✓ Support hundreds of simultaneous users

14 Part I: Web Services Overview

All these processes involve complex Web activity, real-time database activity, and historical database activity. In a typical networking environment, multiple complex applications can't be hosted on a single server. Therefore, the applications must be distributed across multiple servers. If all of the servers that you use for these applications are physically placed next to each other and are on the same platform (that is, Microsoft), you will have few challenges communicating among servers.

On the other hand, if some of the servers are located in other buildings, across the Internet, or on different platforms, you face major challenges trying to get the machines to talk to each other. Web services solves all the problems that occur with traditional distributed computing. You'll see this throughout the rest of this book.

Parts is parts: Putting the .NET pieces together

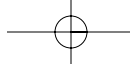
Many parts and terms make up or describe the Microsoft environment that support Web services. This section outlines those parts.



Having some familiarity with these parts and terms is helpful before you move on to subsequent chapters in this book.

Microsoft uses the following parts in the support of Web services:

- ✓ **.NET:** This is Microsoft's vision and surrounding technologies for making applications available any time, any place, on any device. For .NET to be realized, Microsoft put a tremendous amount of thought into the architecture of .NET. Microsoft also thoroughly planned all of the .NET tools, technologies, and subsystems that comprise the .NET platform. Saying that Microsoft is *betting the farm* on .NET is not an understatement. Because of Microsoft's power and standing in the industry, .NET is assured of being a great success.
- ✓ **.NET platform:** This is Microsoft's entire suite of tools, technologies, and services that support Microsoft's vision of connected applications being made available any time, any place, on any device. Web services is a large part of that vision. The .NET platform encompasses the .NET Framework and other components that are outlined in Chapter 2.
- ✓ **.NET Framework:** This is Microsoft's set of services that is used to support Web services. The specific services that comprise the .NET Framework, such as the Common Language Runtime (CLR), Base classes, and so forth, are discussed in Chapter 2. Web services would not work without the .NET Framework. You'll see when you get to Chapter 2, so hold your horses!



- ✓ **Visual Studio .NET:** This is a development tool that is used to create applications for the .NET platform, including Web services. Using Visual Studio .NET is easier than you may think. I show you how to use Visual Studio .NET (also known as *VS.NET*) in Chapter 6.

Understanding the standards that make Web services happen on the .NET platform

Web services are limited only by the creativity of the developers who create them and the software architects who design them. With the rise in popularity and flexibility of markup languages and other technologies, along with the fact that these technologies aren't restricted by platforms, many standards are used in making Web services function on the .NET platform.

The terminology surrounding those standards can be confusing, so I want to make sure that you have a brief understanding before you get to the later chapters that dive deeper into those terms. Because these terms are so interrelated, getting a handle on them now is important. Here are the major terms:

- ✓ **eXtensible Markup Language (XML):** *XML* is a format that is used to describe and format data. XML is formatted into a series of hierarchical tags that structures data based on the way it makes sense to your company. The XML data, known as an *XML Document*, is typically stored in a text file with an `.xml` extension. This XML data can be transmitted over the Internet using Web service technology. As you can see, XML is unbelievably flexible and capable of crossing platforms *and* even applications. XML is discussed further in Chapter 3.
- ✓ **XML Stylesheet Language (XSL):** Because XML is used to describe data only, it contains no instructions for how that data is to be presented on the screen (font size, font color, and so on). That's where XSL comes in. *XSL* is a format that describes how XML should be translated into HTML for display on the Web.



XML only describes data, not how the data is to be displayed. For example, suppose that you have an XML file that describes a banking transaction. The file contains the data related to the transaction itself, such as the account number, the amount of the transaction, and so on. The XML file does not contain any information about fonts, placement of the data on the screen, or anything else; that's where XSL comes in. XSL enables you to indicate how to format the XML so that it can be displayed on the Web.

16

Part I: Web Services Overview

A brief history of data transfer before Web services

Prior to the advent of Web services, many technologies promised to overcome the hurdles of transmitting data between distributed systems. You are probably familiar with many of these technologies:

- ✓ **File Transmission Protocol (FTP):** People use FTP to send and receive files, but FTP is very difficult to automate. FTP is also not geared towards being programmable; it simply transfers files.
- ✓ **Electronic Data Interchange (EDI):** This has proven to be quite useful in the transmission of data but does have a couple of problems. First, it is quite rigid. Documents must conform to a specific format. For example, all purchase orders are transmitted with the same format, which may sound like a good thing, but what if you wanted to add custom fields to your purchase orders? That's where EDI starts to fall apart. Second, EDI is notoriously expensive to implement. EDI files are text-based, but to process those text files requires expensive software and hardware.
- ✓ **Distributed Component Object Model (DCOM):** This one is based on the Microsoft standard, Component Object Model (COM). COM is a model for compiling code into programmable objects. COM is a great technology, but it works only on a single computer. When you start to get into distributed computing, DCOM is supposed to fill the gap. DCOM is programmable and distributed but has two major problems: DCOM is difficult to implement, and it is dependent on the Microsoft platform.
- ✓ **CORBA:** Even though this sounds like a new dance step, it actually stands for Common Object Request Broker Architecture. CORBA is similar to DCOM, but works on a UNIX platform only.
- ✓ **Floppy disk/CD-ROM/e-mail:** Sending files by floppy disk, CD-ROM, or e-mail are all still popular ways of sending files because these methods don't require network authentication or permissions. However, these methods of transmitting data are not programmable. They simply replace FTP.

- ✓ **XML Schema Description (XSD):** *XSD* is a document (file) that describes the *schema* (also known as the *format*) of an XML file. XSD files are used to test that an XML file conforms to the format specified in the XSD file. XSD is discussed in detail in Chapter 3.
- ✓ **Document Object Model (DOM):** *DOM* is a programmable object model that represents the contents of an HTML browser. For example, if a browser contains two text boxes, three radio buttons, and a button, the DOM allows you to access in your code the properties associated with these graphical elements.
- ✓ **XML Document Object Model (XMLDOM):** *XMLDOM* is a programmable object model for an XML document. XMLDOM allows for the querying, updating, inserting, and deleting of elements or nodes within an XML document. I tell you more about elements and nodes in Chapter 3.

- ✔ **Simple Object Access Protocol (SOAP):** *SOAP* is a standard protocol used to transmit XML data over a network, such as the Internet. Web services define the interface of a programmatic object, but SOAP allows a call to a Web service to be routed to the proper location over the Internet. SOAP is discussed further in Chapter 4.
- ✔ **Universal Description, Discovery, and Integration (UDDI):** *UDDI* is a standard for registering Web services that are available over the Internet. It is the *yellow pages* or *white pages* of Web services. Just because you have a Web service available doesn't mean that anyone will be able to find it. That's where UDDI comes in. UDDI allows someone to search a database and find your company or your services. UDDI is discussed further in Chapter 9.
- ✔ **Web Services Description Language (WSDL):** *WSDL* is a standard format for describing your Web service interface that you expose to applications that want to use, or *consume*, your Web service. WSDL is covered in Chapter 9.

We Are Not Alone

Most of the discussion and advertising (some call it *hype*) around Web services is related to the .NET platform (Microsoft's platform). However, just because Microsoft is the *loudest* doesn't mean that it is the only company that supports Web services. Plenty of other companies have jumped on the bandwagon.

You may wonder how other companies can support Web services. The answer is simple. Creating the concept of Web services and related technologies was not an effort by Microsoft alone. It was a joint specification developed by Microsoft and other companies and submitted as a standard for passing data over the Internet to the World Wide Web Consortium (known as the *W3C*). The specification was adopted by the *W3C*, which makes it an open, publicly available standard. Although I don't discuss the details of the standard per se, everything I discuss in this book is Microsoft's interpretation of this standard and how it chose to implement those standards. That's what the whole buzz around .NET is about.



Except for this section, this book is dedicated entirely to Web services on the Microsoft .NET platform. However, discussing what the competition is up to is only fair, even though I don't provide a review or an opinion on any of the other vendors' products. I've listed the other vendors in alphabetical order, so as to not show favoritism.

18

Part I: Web Services Overview

HP

HP has a Web service platform, called *HP Web Services Platform* (Version 2.0 at the time of this writing), that can be used to generate Web services. It was formerly known as *e-Speak*. Its platform is built around exposing Java and J2EE (Java 2 platform, Enterprise Edition) objects as Web services.

The HP Web Services Platform consists of the following tools that enable you to build Web services:

- ✓ **HP-SOAP:** Application server that processes SOAP messages
- ✓ **HP Registry Composer:** Enables the registration of Web services to public and private registries
- ✓ **HP Service Composer:** Development tool for graphically creating Web services

You can obtain more information about HP Web services at www.hp.com/go/webservices.

IBM

IBM fully embraces the concept of Web services and is integrating it into many of its products. In fact, IBM has worked with Microsoft to create applications that prove interoperability between platforms. IBM's answer to Web services is to provide a platform consisting of the following:

- ✓ **WebSphere Application Server:** A server that supports and hosts Web services on the IBM platform
- ✓ **WebSphere Studio:** A graphical environment for creating Web services, servlets, Java Server Pages (JSP), and JavaBeans
- ✓ **WebSphere SDK for Web Services (WSDK):** A toolkit that provides everything a developer needs to create Web services on the IBM platform, including samples, examples, architectural documentation, scripts, Java SDK, and even a scaled-down version of its WebSphere product
- ✓ **Web Services Toolkit (WSTK):** A toolkit that provides documentation and prototypes of how to develop Web services on the IBM platform

You can obtain more information about IBM Web services at www3.ibm.com/software/solutions/webservices.

Novell

Novell's strategy for implementing Web services is called *one Net*. one Net is a vision for allowing all networks to communicate with each other. This includes intranets, extranets, wireless networks, and more. To support this vision, Novell acquired SilverStream, apparently for their exteNd Web Services Platform.

The eXtend Web Services Platform consists of the following:

- ✓ **exteNd Composer:** A visual integration server, allowing connection to legacy systems
- ✓ **exteNd Director:** This enables developers to define personalization, workflow, business rules, and more
- ✓ **exteNd Workbench:** This enables the rapid development of Java/J2EE Web services applications
- ✓ **exteNd Application Server:** This provides runtime environment for Novell's Web services

You can obtain more information about Novell's Web services at www.novell.com/webservices.

Oracle

Oracle was a straggler in announcing a Web services strategy. However, it is supporting Web services on its Oracle9iAS Web Services Platform. The platform consists of:

- ✓ **OC4J:** The runtime environment for Oracle's Web services
- ✓ **Oracle9i Jdeveloper:** A graphical development tool for creating Java/J2EE-based objects and Web services
- ✓ **Oracle Enterprise Manager:** A management console for administering Web services
- ✓ **Oracle9iAS UDDI Registry:** This enables the publishing of Web services into UDDI and allows searching with UDDI

You can obtain more information about Oracle's Web services at <http://otn.oracle.com/tech/webservices/content.html>.

20 Part I: Web Services Overview

Sun Microsystems

Sun has supported the Web services standards by embracing them into its ONE (*Open Net Environment*) architecture. Sun also provides some tools to help with the development and support of Web services on the Sun platform:

- ✓ **Sun ONE Studio:** A graphical environment for developing Web services on the Sun platform
- ✓ **Sun ONE Web Server:** A Web server for hosting Web services
- ✓ **Sun ONE Application Server:** A server for serving J2EE-compliant applications and hosting Web services

You can obtain more information about Sun Microsystems Web services at www.sun.com/webservices.