2

Web Services Basics

If you ask five people to define Web services, you'll probably get at least six answers. Some people use the term "Web services" to describe applications that communicate with Simple Object Access Protocol (**SOAP**). (SOAP is an XML messaging protocol. We'll discuss it in detail in Chapter 3.) Other folks use the term to describe only the SOAP interface. Still other people vehemently object to the idea of constraining the definition to a specific technology such as SOAP. Some people use the term to describe any application that communicates over the Internet. Other people use the term to describe any Web-based application. Some people view Web services as anything that's accessible over the Web. And some people use the term to describe the **software-as-a-service** business model.

There's no one official definition of "Web services"

Given that there is no official consensus within the industry, I am establishing my own set of names and definitions. I want to give you a basic grounding to help you understand this technology, so my goal is to make things as simple and straightforward as possible.

For the purposes of this book I am defining my own terminology

What Is a Web Service?

The simplest and most basic definition that I can give you is that a Web service is an application that provides a **Web API**. As mentioned in Chapter 1, an API supports application-to-application communication. A Web API is an API that lets the applications communicate using XML and the Web.

A Web service is an application with a Web API

So here's the basic concept: Web services use the Web to perform application-to-application integration. A lot of the hype around Web services talks about dynamic assembly of Web-based software

Web services use the Web for application-to-application integration

services. It talks about the software-as-a-service business model. It talks about spontaneous discovery of new business partners. My advice is to ignore this hype. It's possible that at some point in the future some of these glossy images will become reality, but please don't let the science fiction stories distract you from reality or dissuade you from using this technology today to solve real business issues.

Why Web Services?

Web services help you integrate applications

Rather than "what?" I think the more important question is "why?" Why should you care about Web services? The answer is that Web services mitigate the application integration crisis. They help you integrate applications, and they do so at a significantly lower price point than any other integration technology.

XML and the Web solve the "Traditional Middleware Blues"

Web services represent a new form of middleware based on XML and the Web. XML and the Web help solve the challenges associated with traditional application-to-application integration, which I identified in Chapter 1 as the Traditional Middleware Blues. To summarize:

- ❏ Traditional middleware doesn't support heterogeneity.
- ❏ Traditional middleware doesn't work across the Internet.
- ❏ Traditional middleware isn't pervasive.
- ❏ Traditional middleware is hard to use.
- ❏ Traditional middleware is expensive.
- ❏ Traditional middleware maintenance costs are outrageous.
- ❏ Traditional middleware connections are hard to reuse.
- ❏ Traditional middleware connections are fragile.

Web services support heterogeneous interoperability

Web services address these issues. Web services are platform- and language-independent. You can develop a Web service using any language, and you can deploy it on any platform, from the tiniest device to the largest supercomputer. More to the point, any Web

service can be accessed by any other application, regardless of either's language or platform. Web services communicate using XML and Web protocols, which are pervasive, work both internally and across the Internet, and support heterogeneous interoperability.

Web services are inexpensive

Web services simplify the process of making applications talk to each other. Simplification results in lower development cost, faster time to market, easier maintenance, and reduced total cost of ownership. The bottom line is this: Web services allow you to integrate your applications at a fraction of the cost of traditional middleware.

Web services are flexible and adaptable

Traditional RPC-style middleware, such as RPC, CORBA, RMI, and DCOM, relies on tightly coupled connections. A tightly coupled connection is very brittle, and it can break if you make any modification to the application. Tightly coupled connections are the source of many a maintenance nightmare. In contrast, Web services support loosely coupled connections. Loose coupling minimizes the impact of changes to your applications. A Web service interface provides a layer of abstraction between the client and server. A change in one doesn't necessarily force a change in the other. The abstract interface also makes it easier to reuse a service in another application. Loose coupling reduces the cost of maintenance and increases reusability.

Defining "Web" and "Service"

A Web service is a service that lives on the Web

So let's dig a little deeper into our definition. Just what is a Web service? If we dissect the name, we can infer that a Web service has something to do with the Web and something to do with services. I like to say that a Web service is a service that lives on the Web. This definition doesn't help us very much, though, unless we know the meaning of the terms "Web" and "service." So let's start there.

The Web is a huge information space filled with interconnected resources

The Web is an immensely scalable information space filled with interconnected resources. The architecture for the Web has been developed and standardized by the World Wide Web Consortium (**W3C**). A **Web resource** is any type of named information object—such as a word processing document, a digital picture, a Web page, an e-mail account, or an application—that's accessible through the Web. All resources on the Web are connected via the Internet, and you access Web resources using standard Internet protocols. Any network-enabled application or device can access any resource in the Web. Right off the bat, you can see that the Web solves one of your integration challenges: The Web is pervasive and provides universal connectivity.

A service is an application that can be consumed by software

A **service** is an application that exposes its functionality through an application programming interface (API). In other words, a service is a resource that is designed to be consumed by software rather than by humans.

"Service" refers to the service-oriented architecture

The term "service" implies something special about the application design. It refers to something known as the **service-oriented architecture** (**SOA**). The SOA is the basic architecture used by most RPC-style middleware systems. Chapter 3 talks about the SOA in detail.

An interface hides the complexities of the internal system

One of the most important features of the SOA is the separation of interface from implementation. A service exposes its functionality through an interface, and that interface hides the inner workings of the application. A client application doesn't need to understand how the service actually performs its work. All it needs to understand is how to use the interface. To give you an analogy, let's look at a car. A car is a complicated machine, but the car provides a set of interfaces that's relatively simple to use. To start a car, you don't need to know how an internal combustion engine

works, or even how the starter motor works. You only need to know how to use the interface that the car supplies to start it: Turn the key.

A Web service is both a Web resource and a service

A Web service possesses the characteristics of both a Web resource and a service. It is an application that exposes its functionality through an API, and it is a Web resource that is designed to be consumed by software rather than by a human sitting at a browser.

A service is software that does work for other software

Understanding the concept of a service is key to understanding Web services. A service is a piece of software that does work for other software. In most circumstances, a service runs on a server, waiting for an application to call it and ask it to do some work. In many cases services don't provide any type of human interface, and the only way to access the service is through its API.

A service can perform system or business functions

A service can perform system functions or business application functions. For example, a file service can create, find, save, or delete a file. A stock quote service can retrieve the current ask and bid prices of an equity.

A service plays the role of server in a client/server relationship

All client/server technologies rely on this basic concept of a service. A service is the business or system application that plays the part of the server in a client/server relationship. Print servers, file servers, database servers, Web servers, and **application servers** are all examples of service-oriented systems.

Application services often run in an application server

Any business application that exposes its capabilities through an API is a service. Business application services often run in an application server. An application server manages and coordinates the utilization of all resources available in a shared, multiprocessing environment, enabling optimized performance, scalability, reliability, and availability.

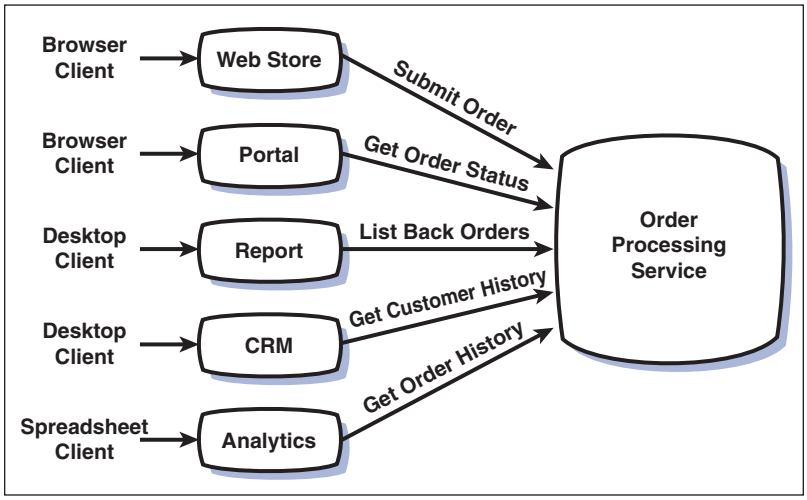

Figure 2-1: A service can be shared by many different applications.

A service is a shared resource

You often need this type of scalability because many different users can share a single service. A service is a shared resource. One reason you might want to design a business application as a service is to consolidate your efforts and reduce duplicated work. If there is a particular piece of functionality that many of your applications need to perform, you should build this functionality as a service rather than reimplement the functionality in each application.

One service can support many applications

For example, as shown in Figure 2-1, it's much simpler and easier to manage and maintain your order processing system if you have only one application service that actually processes orders. This one service can support all the different ways that you offer to place orders, inquire about order status, and generate reports about orders.

Building Services

You use middleware to create your network API

To let clients access a service over the network, you must build a network API for the service. You generally use some type of communication middleware to create a network API. You can use a traditional middleware technology, such as RPC, DCOM, CORBA, RMI, or MOM, but all these technologies suffer from the Traditional

Middleware Blues. If you want to make your services available to heterogeneous users across multiple systems (including the Internet) at a reasonable cost, you should use middleware technology that supports these requirements.

Web services are Web-based middleware

Web services represent a new type of middleware that relies on the Web. The Web resolves the pervasive aspects of the Traditional Middleware Blues.[1] The Web is pervasive. The Web is free. The Web is completely vendor-, platform-, and language-independent. The Web uses the Internet as its native communication protocol. Web services support easy integration, flexibility, and service reusability.

Web Evolution

The Web was designed for interactive communication

The Web was originally created to support interactive communication. People use the Web to communicate with other people and to access information. You use e-mail and instant messaging to converse with friends and colleagues. You use a browser to access information.

A dynamic Web site allows people to talk to applications

In the early days of the Web, a **Web site** was simply a set of static pages that were stored in files. You could view only the text and pictures contained in these files. To change what users saw, a Web site operator had to edit the files. Soon we realized that we could also use the Web to access dynamic information. When you link to a dynamic page, the **Web server** doesn't merely display a file. Instead it calls an application that dynamically generates and renders the requested information. The introduction of this technique marked the point when the Web evolved to allow people to talk to applications.

[1]The other Traditional Middleware Blues tend to be a function of tightly coupled connections. Web services solve these issues using XML. Chapter 3 discusses XML.

Web services allow applications to talk to applications

Web services represent the next step in the Web's evolution because they allow applications to talk to applications. Web-based application-to-application integration allows us to exploit the universal connectivity and immense scalability of the Web, and it supports a much richer set of usage models than do human-oriented applications.

Web sites support humans; Web services support software

Figure 2-2 summarizes the differences between a Web *site* and a Web *service*. A Web site represents a group of Web resources that are designed to be accessed by humans, and a Web service represents a group of Web resources that are designed to be accessed by applications.

A service interface must be structured and unambiguous

The interfaces to these two types of applications are fundamentally different. A Web site supports human clients who have a tremendous capacity to interpret the meaning of information. The site returns information as a Hypertext Markup Language (**HTML**) page—a string of text containing formatting information, often including graphics, clickable buttons, and links. A human interprets this information based on visual layout and physical association. In contrast, an application can't interpret information this way. An

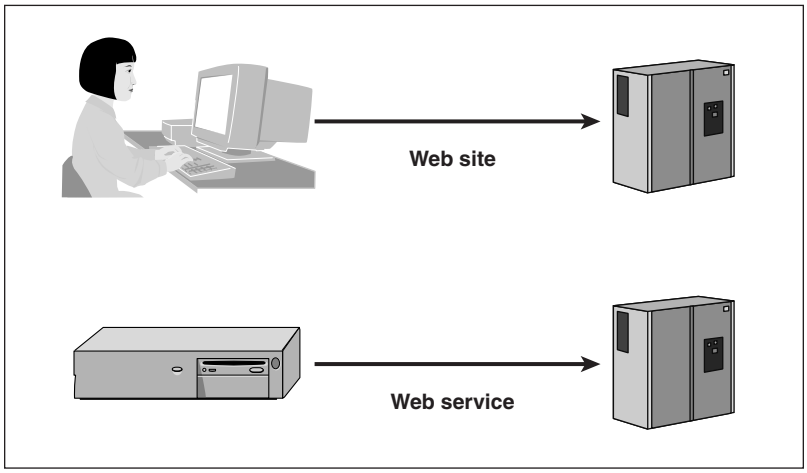

Figure 2-2: A Web site is designed to be accessed by humans. A Web service is designed to be accessed by applications.

application needs unambiguous information. It needs to know what programmatic functions are available, and it needs to know how to structure and interpret the data being exchanged. A Web API defines these programmatic functions and data structures in a completely unambiguous way.

Defining Characteristics of Web Services

A Web service is a Web resource that provides an API

A Web service exhibits the following defining characteristics:

- A Web service is a Web resource. You access a Web service using platform-independent and language-neutral Web protocols, such as HTTP. These Web protocols ensure easy integration of heterogeneous environments.
- A Web service provides an interface—a Web API—that can be called from another program. This application-to-application programming interface can be invoked from any type of application. The Web API provides access to the application logic that implements the service.
- A Web service is typically registered and can be located through a Web service **registry**. A registry enables service consumers to find services that match their needs. These service consumers may be humans or other applications.
- Web services support loosely coupled connections between systems. Web services communicate by passing XML messages to each other via a Web API, which adds a layer of abstraction to the environment that makes the connections flexible and adaptable.

Understanding the Scope of Web Services

Web services concepts can be divided into four layers

So now that we have the basic definition down, let's go back to the big picture. How do you build Web services? What do you need to run Web services? How do you use Web services? Obviously this

topic covers a lot of territory. Figure 2-3 divides the scope of our discussion into four basic concepts: XML and **Web services technologies**, **Web services infrastructure**, **Web services**, and **Web services application templates**. Each layer builds on the layers below it.

XML and Web services technologies provide the foundation for Web services

The bottom layer in Figure 2-3 comprises XML and Web services technologies. These technologies provide the foundation for Web services. Don't worry about all the acronyms used in this illustration. We'll take a closer look at these technologies in Chapters 3–5. (If you can't wait, you can find definitions for the acronyms in the Glossary.)

Infrastructure refers to products that implement Web services technologies

The next layer in Figure 2-3 represents Web services infrastructure: products that implement the XML and Web services technologies. You use these products to build, deploy, manage, and use Web services. Chapters 8 and 9 will take a closer look at Web services infrastructure.

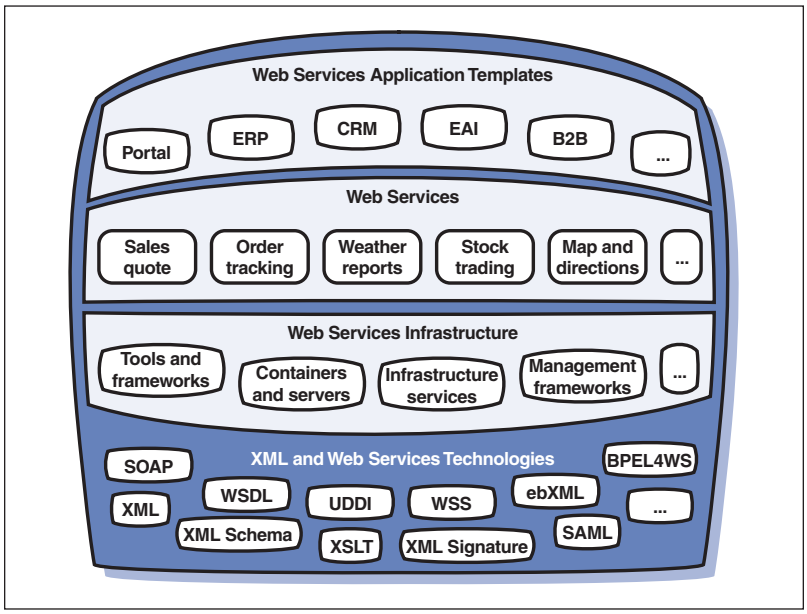

Figure 2-3: Web services concepts can be divided into four logical layers: XML and Web services technologies, Web services infrastructure, Web services, and Web services application templates.

Web services are applications that communicate using Web services infrastructure

A Web service represents an information resource or business process that you have made available to other applications through a Web API. In particular, it is a resource that supports application-to-application communication using Web services infrastructure. You can turn any piece of application code into a Web service. A Web service can do whatever you've programmed it to do. Figure 2-3 lists five sample Web services: sales quote, order tracking, weather reports, stock trading, and map and directions.

Application templates represent systems that benefit from Web services

Web services application templates represent the kinds of applications and initiatives for which Web services technology offers substantial benefits. The list of templates in Figure 2-3 is by no means exhaustive, but it identifies some of the more popular uses of Web services, such as portals, enterprise resource planning, customer relationship management, enterprise application integration initiatives, and business-to-business integration. We'll discuss a number of real-life Web services applications in this chapter and in Chapter 7.

Web Services Business Models

Web services is not a business model

You may have noticed that I didn't list software-as-a-service as a Web services application template. That's because software-as-a-service isn't an application. It's a business model in which you license subscription rights to access hosted software rather than license the rights to deploy the software in your own organization. For example, Salesforce.com uses the software-as-a-service business model. Salesforce.com hosts a CRM system, and users pay a monthly subscription fee to use the software.

Many people equate Web services with software-as-a-service

A lot of the early hype about Web services led many people to equate Web services with the software-as-a-service business model. The hype projects a blue-sky vision of being able to dynamically discover, assemble, and consume Internet-based software services.

But IDC predicts that the realization of this vision is at least 10 years away. I view that prediction as optimistic.

Web services should support your existing business models

My point is that, except in a few rare circumstances, you don't sell a Web service. Instead you sell some other type of product or service, and you use Web services to help you do that. Only in very rare circumstances are Web services the focus of their own business model. Without a viable business model, it's hard to create a business case for Web services. For example, let's look at Google.

Google

Google's business model is based on advertising

Google is the world's leading Web search company. Google provides a public search engine that contains an index of more than three billion Web pages. The normal interface to this search engine is a human-oriented browser interface. The business model that supports this public service is advertising. Users can access the service for free in exchange for viewing a few ads. Google collects revenues from the businesses that place the ads.

The Google Web service provides an API to the search engine

Google also provides a Web service interface to this public search engine. It calls this Web service the Google Web APIs. You can use these Web APIs to query the Google search engine from an application rather than from a browser. The results of the search are returned as structured data so that the requesting application can process the information.

Google is encouraging users to create innovative applications using these Web APIs

As of the time of this writing, this Web service is still in an experimental stage. Google is encouraging developers to use their imagination to create new and interesting applications using the Google Web APIs. Here are three examples:

- Subject monitoring: issue regularly scheduled Web searches to find any new information on a particular subject

- Market research: issue regularly scheduled Web searches and analyze the difference in the amount of information available on the subject over time
- Plagiarism search: search for phrases from a piece of writing to ensure that it is original material

The Google Web APIs undermine Google's normal business model

Researchers and developers may be excited about the Google Web APIs, but it's hard to figure out what benefit Google will gain from this Web service other than goodwill. The Google Web APIs undermine Google's normal business model. The Google Web APIs don't constitute a new service. Instead they simply provide a programmatic interface to Google's public Web search engine. The Web APIs are free. Users are required to register, and they are limited to 1,000 queries per day per user, but users of the Google Web APIs don't receive the Google advertisements.

Web services should support your existing business models

The cost of an individual Google search is minuscule. Google views it as a reasonable investment to give away a few million searches in exchange for the generation of goodwill. But in general, I wouldn't recommend that you follow Google's example. Web services should be designed to support your existing business model. They should provide a new or improved mechanism to sell or use an existing product or service.

Kinko's

"File, Print . . . Kinko's" will allow you to send a print job to Kinko's directly from your Office application

For example, let's look at Kinko's, the world's leading provider of document solutions and business services. Kinko's has offered a browser-based utility for quite a while that allows you to send documents from your PC directly to Kinko's for printing. Now Kinko's wants to use Web services to make the process even more seamless. Kinko's plans to roll out a "File, Print . . . Kinko's" Web service in mid-2003. This Web service allows you to send a print job to Kinko's over the Internet directly from any Microsoft Office

application. The service will require you to install a small add-in to Office, which will supply the client interface to the Kinko's Web service. After you've installed this add-in, "Kinko's" will appear in your list of printers when you select File and Print . . . from the Office menu. When you select the Kinko's print service, Office will launch Kinko's client interface, which then presents you with an easy-to-use dialog box to guide you through the process of submitting a print job. The dialog box will help you find a convenient Kinko's location, select options such as stapling and binding, and specify payment, notification, and delivery methods.

You can send the print job to any Kinko's anywhere in the world

Suppose you're sitting in your hotel room writing a proposal in Microsoft Word. When you're finished, you select File, Print . . . Kinko's. The hotel's high-speed Internet connection sends the print job to a Kinko's in another city, and the proposal is delivered directly to your client. Kinko's will even send you a notification when the job is complete.

Kinko's Web service supports the company's core business model

The "File, Print . . . Kinko's" Web service doesn't compete with the company's core business model. It enhances it by providing another way for users to submit print jobs. And it provides a level of convenience that many users will certainly appreciate.

Amazon

Amazon provides a Web API to support its marketing affiliates

Amazon also uses Web services to enhance its core business model. Amazon's business model is based on online retail sales. Amazon is renowned for the features of its online catalog, which provides the primary consumer sales interface. The catalog is designed to be viewed by a human sitting at a browser. Amazon also wants to make this catalog available to applications so that its 800,000 marketing affiliates can more easily sell products for Amazon. So Amazon created a Web API for its catalog. Before it offered this Web API, it was quite difficult to access the Amazon

catalog from an application. You needed to build a screen scraping application that simulated a human sitting at a browser.

Amazon hopes its Web service will boost book sales

The new Amazon Web API allows Amazon's marketing affiliates to easily incorporate Amazon content and features into their Web sites. Many of Amazon's most popular search facilities—such as keyword search, ISBN search, and even "Listmania!"—are available through the Web service. Now consumers can buy products from Amazon transparently through the affiliate sites. The affiliate Web site uses the Amazon Web service to search Amazon's catalog and display the results on its own site, including features such as Amazon reviews and book ratings. This free Web service is a win-win situation for both the affiliates and Amazon. Each time a consumer makes an Amazon purchase through the affiliate site, the affiliate earns a 15% referral fee. Meanwhile Amazon expects to see a boost in product sales.

UPS

UPS OnLine Tools streamline the logistics process

UPS also uses Web services to promote sales. UPS provides a set of Web APIs called UPS OnLine Tools. Businesses can use these APIs to connect their applications directly to the UPS logistics system to add integrated shipping, tracking, and related functionality. UPS OnLine Tools are available at no charge, and UPS provides free e-mail support. As with Amazon, this Web service offers a win-win situation. Customers appreciate the way this Web service can streamline their logistics process; UPS can expect to see an increase in UPS shipments.

T-Mobile

T-Mobile uses Web services to enable a new business model

Sometimes Web services can help enable a new business model. T-Mobile International, a division of Deutsche Telekom, is one of the world's leading international mobile communication providers. One of its service offerings, T-Mobile Online, provides a wireless Web portal for more than three million T-Mobile customers in

Austria, the Czech Republic, Germany, and the United Kingdom. As with most wireless plans, the business model is based on consumer usage.

T-Mobile needs interesting content to attract users

When first planning T-Mobile Online, T-Mobile realized that to promote consumer usage it needed to provide interesting content on the portal. Recruiting content providers was critical to the success of this new venture. T-Mobile needed to make sure that it was as easy as possible for content providers to join the network.

The content providers need consumer info and billing services

One of the biggest challenges T-Mobile faced was figuring out a way to give the content providers access to information about individual consumers. Providers need this information to furnish customized, localized, useful content. Another challenge was devising an affordable micro-payment system to ensure that the content providers got paid for their services.

Web services ensure easy content integration

Given that each content provider might have a completely different IT infrastructure, T-Mobile elected to use Web services. All consumer information and billing services are made available to the content providers as Web services, as shown in Figure 2-4. The Web services ensure that content providers can quickly, easily, and inexpensively integrate their content into the T-Mobile portal.

Web services enable this m-commerce business model

This venture has been very successful. T-Mobile Online has enlisted more than 200 content providers to make the wireless Web interesting and appealing to T-Mobile consumers. Through T-Mobile Online, these content providers provide services such as e-mail, Short Message Service (SMS) messaging, news, sports scores, restaurant recommendations, directions, stock trading, banking, ticket purchases, gambling, and more. T-Mobile doesn't charge either its consumers or the content providers for these Web services. Instead T-Mobile makes money from the increased airtime

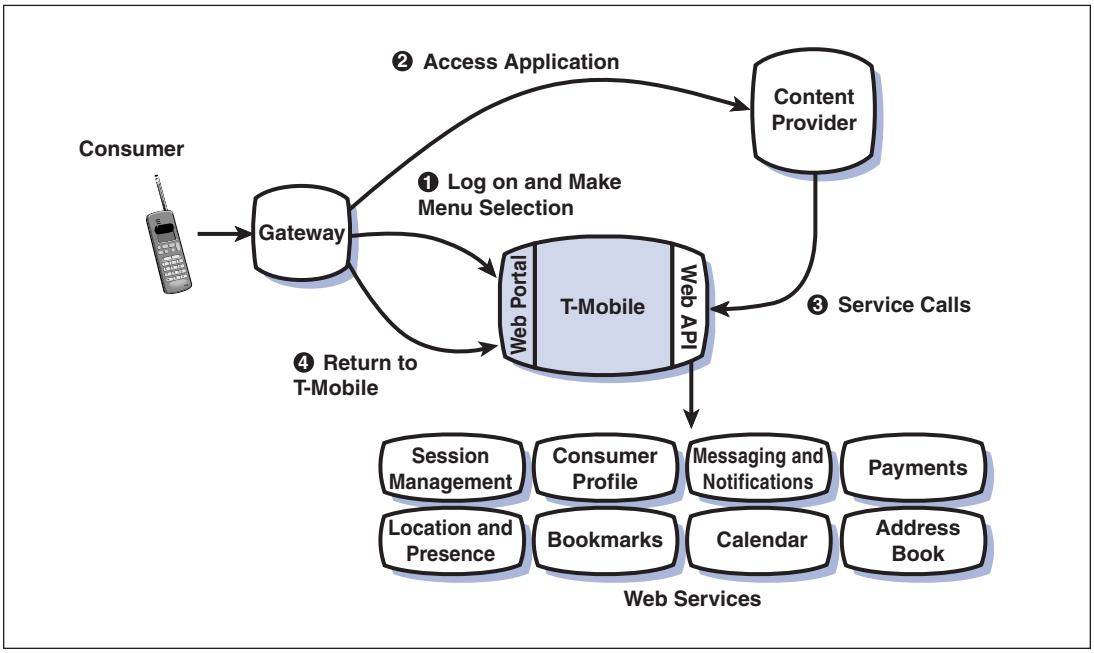

Figure 2-4: T-Mobile Web services maintain user session information, automatically capture and manage billing and payment services, and allow content providers to obtain information about consumers.

the consumers use to access these third-party offerings. The Web services aren't the focus of the business model, but it wouldn't work without them.

Internal Integration

Web services can make it easier for your customers and partners to do business with you

In the examples I've cited so far, I've talked only about external integration applications. One key theme that permeates all these examples is that Web services can make it easier for your customers or partners to do business with you. Anything that simplifies business integration is a valuable commodity. Another recurring theme is that Web services do not themselves define a business model. Instead, they support existing business models, and in some circumstances they enable a new business model.

Web services lower the cost of application integration

Although the external applications are interesting, most production applications based on Web services are internal application projects. As with external Web services, internal Web services should support your core business model. You can use them to improve and optimize your internal application systems to make your business processes work better. The first and foremost reason you should be exploring Web services is that they can dramatically lower the cost of application integration.

Merrill Lynch saved more than 96% on a project with Web services

Merrill Lynch completed an internal application integration project in 2002. The idea was to build an integration bus to provide access to mainframe-based Customer Information Control System (CICS) applications. An integration bus is a common pathway that multiple applications can use to communicate. The original estimated cost for the project based on message-oriented middleware was $800,000. Then the company switched to Web services technology. Rather than purchase software licenses for the MOM technology on a host of different platforms and then build a bunch of adapters to allow the various client applications to use the MOM middleware, Merrill Lynch developed a small SOAP gateway for the CICS environment for only $30,000. Now any client environment can access the CICS environment using SOAP, and Merrill Lynch doesn't need any special software or any special adapters on any of its systems.

Executive Summary

A Web service is an application that provides a Web API

The simplest definition of a Web service is an application that provides a Web API. The Web API exposes the functionality of the application to other applications. The Web API relies on Web services technology to manage communications. Web services technology is pervasive, vendor-independent, language-neutral, and very low-cost.

A Web API enables application integration

The purpose of a Web API is to enable application integration. More specifically, a Web API lets you integrate heterogeneous applications. You can use Web services to achieve many different goals. You can use them to implement internal point-to-point application integration projects. You can use them to consolidate your development efforts and reduce redundant applications. You can use them to implement a general-purpose integration bus for your internal application systems. And you can use them to make it easier for your partners and your customers to do business with you.

Web services should support your core business model

Web services do not represent a new business model. Instead Web services are a technology that you can use to build systems to support a business model.

Web services let you do more with less

IT departments are being asked to do more with less. There's less money in the budget to buy software, and there are fewer people to do the work. Nearly every application development project involves some level of application integration. It just makes sense to reduce the cost and simplify the process of doing integration. Web services are an obvious choice.