Web Services

1.1 What Are Web Services?

A web service is a piece of business logic, located somewhere on the Internet, that is accessible through standard-based Internet protocols such as HTTP or SMTP. Using a web service could be as simple as logging into a site or as complex as facilitating a multiorganization business negotiation.

Given this definition, several technologies used in recent years could have been classified as web service technology, but were not. These technologies include win32 technologies, J2EE, CORBA, and CGI scripting. The major difference between these technologies and the new breed of technology that are labeled as web services is their standardization. This new breed of technology is based on standardized XML (as opposed to a proprietary binary standard) and supported globally by most major technology firms. XML provides a language-neutral way for representing data, and the global corporate support ensures that every major new software technology will have a web services strategy within the next couple years. When combined, the software integration and interoperability possibilities for software programs leveraging the web services model are staggering. A web service has special behavioral characteristics:

XML-based

By using XML as the data representation layer for all web services protocols and technologies that are created, these technologies can be interoperable at their core level. As a data transport, XML eliminates any networking, operating system, or platform binding that a protocol has.

Loosely coupled

A consumer of a web service is not tied to that web service directly; the web service interface can change over time without compromising the client's ability to interact with the service. A tightly coupled system implies that the client and server logic are closely tied to one another, implying that if one interface changes, the other must also be updated. Adopting a loosely coupled architecture tends to make software systems more manageable and allows simpler integration between different systems.

Coarse-grained

Object-oriented technologies such as Java expose their services through individual methods. An individual method is too fine an operation to provide any useful at a corporate level. Building a Java program from scratch requires the creation of several fine-grained methods that are then composed into a coarse-grained service that is consumed by either a client or another service. Businesses and the interfaces that they expose should be coarse-grained. Web services technology provides a natural way of defining coarse-grained services that access the right amount of business logic.

Ability to be synchronous or asynchronous

Synchronicity refers to the binding of the client to the execution of the service. In synchronous invocations, the client blocks and waits for the service to complete its operation before continuing. Asynchronous operations allow a client to invoke a service and then execute other

functions. Asynchronous clients retrieve their result at a later point in time, while synchronous clients receive their result when the service has completed. Asynchronous capability is a key factor in enabling loosely coupled systems.

Supports Remote Procedure Calls (RPCs)

Web services allow clients to invoke procedures, functions, and methods on remote objects using an XML-based protocol. Remote procedures expose input and output parameters that a web service must support. Component development through Enterprise JavaBeans (EJBs) and .NET Components has increasingly become a part of architectures and enterprise deployments over the past couple of years. Both technologies are distributed and accessible through a variety of RPC mechanisms. A web service supports RPC by providing services of its own, equivalent to those of a traditional component, or by translating incoming invocations into an invocation of an EJB or a .NET component.

Supports document exchange

One of the key advantages of XML is its generic way of representing not only data, but also complex documents. These documents can be simple, such as when representing a current address, or they can be complex, representing an entire book or RFQ. Web services support the transparent exchange of documents to facilitate business integration.

1.1.1 The Major Web Services Technologies

Several technologies have been introduced under the web service rubric and many more will be introduced in coming years. In fact, the web service paradigm has grown so quickly that several competing technologies are attempting to provide the same capability. However, the web service vision of seamless worldwide business integration is not be feasible unless the core technologies are supported by every major software company in the world.

Over the past two years, three primary technologies have emerged as worldwide standards that make up the core of today's web services technology. These technologies are:

Simple Object Access Protocol (SOAP)

SOAP provides a standard packaging structure for transporting XML documents over a variety of standard Internet technologies, including SMTP, HTTP, and FTP. It also defines encoding and binding standards for encoding non-XML RPC invocations in XML for transport. SOAP provides a simple structure for doing RPC: document exchange. By having a standard transport mechanism, heterogeneous clients and servers can suddenly become interoperable. .NET clients can invoke EJBs exposed through SOAP, and Java clients can invoke .NET Components exposed through SOAP.

Web Service Description Language (WSDL)

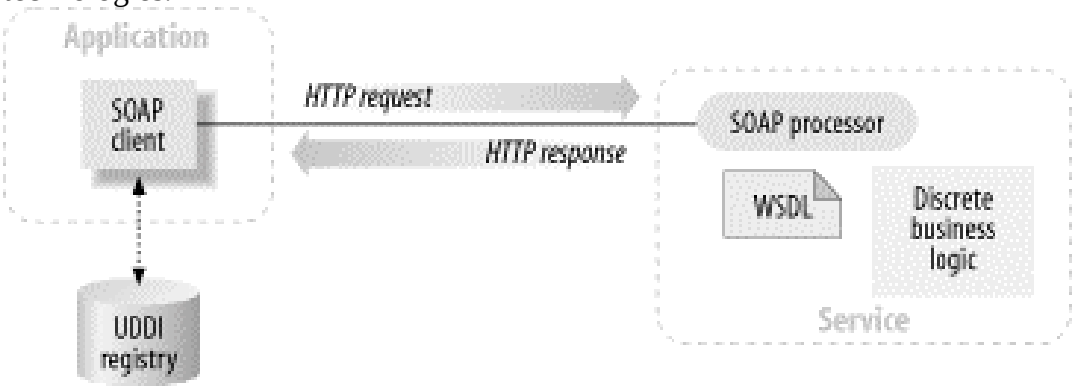
WSDL is an XML technology that describes the interface of a web service in a standardized way. WSDL standardizes how a web service represents the input and output parameters of an invocation externally, the function's structure, the nature of the invocation (in only, in/out, etc.), and the service's protocol binding. WSDL allows disparate clients to automatically understand how to interact with a web service.

Universal Description, Discovery, and Integration (UDDI)

UDDI provides a worldwide registry of web services for advertisement, discovery, and integration purposes. Business analysts and technologists use UDDI to discover available web services by searching for names, identifiers, categories, or the specifications implemented by the web service. UDDI provides a structure for representing businesses, business relationships, web services, specification metadata, and web service access points.

Individually, any one of these technologies is only evolutionary. Each provides a standard for the next step in the advancement of web services, their description, or their discovery. However, one of the big promises of web services is seamless, automatic business integration: a piece of software will discover, access, integrate, and invoke new services from unknown dynamically without the need for human intervention. Dynamic integration of this nature requires the combined involvement of SOAP, WSDL, and UDDI to provide a dynamic, standard infrastructure for enabling the dynamic business of tomorrow. Combined, these technologies are revolutionary because they are the first standard technologies to offer the promise of a dynamic business. In the past, technologies provided features equivalent to SOAP, WSDL, and UDDI in other languages, but they weren't supported by every major corporation and did not have a core language as flexible as XML.

Figure 1-1 provides a diagram that demonstrates the relationship between these three technologies.



The relationship between these pieces (SOAP, WSDL, and UDDI) can be described as follows: an application acting in the role of a web services client needs to locate another application or a piece of business logic located somewhere on the network. The client queries a UDDI registry for the service either by name, category, identifier, or specification supported. Once located, the client obtains information about the location of a WSDL document from the UDDI registry. The WSDL document contains information about how to contact the web service and the format of request messages in XML schema. The client creates a SOAP message in accordance with the XML schema found in the WSDL and sends a request to the host (where the service is).

Practical Applications for Web Services

Because of the cross-platform interoperability promised by SOAP and web services, we can provide practical business solutions to problems that, until now, have only been a dream of distributed-computing proponents.

It's easy to see the use for simple, discrete web services such as a currency conversion service that converts dollars to Euros or a natural language translation service that converts English to French. Today, web sites such as *www.xmethods.com* are dedicated to hosting simple web services

This scenario becomes more exciting when we see real companies using web services to automate and streamline their business processes. Let's use the concept of a Business-to-Consumer (B2C) portal. Web-based portals, such as those used by the travel industry, often combine the offerings of multiple companies' products and services and present them with a unified look and feel to the consumer accessing the portal. It's difficult to integrate the backend systems of each business to provide the advertised portal services reliably and quickly.
Web services technology is already being used in the integration between Dollar Rent A Car Systems, Inc. and Southwest Airlines Co. Dollar uses the Microsoft SOAP Toolkit to integrate its online booking system with Southwest Airlines Co.'s site. Dollar's booking system runs on a Sun Solaris server, and Southwest's site runs on a Compaq OpenVMS server. The net result (no pun intended) is that a person booking a flight on Southwest Airline's web site can reserve a car from Dollar without leaving the airline's site. The resulting savings for Dollar are a lower cost per transaction. If the booking is done online through Southwest and other airline sites, the cost per transaction is about $1.00. When booking through traditional travel agent networks, this cost can be up to $5.00 per transaction.

The healthcare industry provides many more scenerios in which web services can be put to use effectively. A doctor carrying a handheld device can access your records, health history, and your preferred pharmacy using a web service. The doctor can also write you an electronic prescription and send it directly to your preferred pharmacy via another web service. If all pharmacies in the world standardized a communication protocol for accepting prescriptions, the doctor could write you a subscription for any pharmacy that you selected. The pharmacy would be able to fulfill the prescription immediately and have it prepared for you when you arrive or couriered to your residence.

This model can be extended further. If the interfaces used between doctors and pharmacies are standardized using web services, a portal broker could act as an intermediary between doctors and pharmacies providing routing information for requests and better meet the needs of individual consumers. For example, a patient may register with an intermediary and specify that he wants to use generic drugs instead of expensive brand names. An intermediary can intercept the pharmaceutical web service request and transform the request into a similar one for the generic drug equivalent. The intermediary exposes web services to doctors and pharmacies (in both directions) and can handle issues such as security, privacy, and nonrepudiation.