An Introduction to Web Services

Tracy Gardner introduces web services, self-describing applications that can be discovered and accessed over the web by other applications.

What are Web Services?

The term web services has gaining visibility in recent months, there has been a W3C workshop on web services [1] and the big industry players such as IBM, Microsoft and Sun have been announcing their web services strategies. So what's it all about and should you be involved? This article aims to be a starting point for answering these questions.

Well, the term web services is fairly self-explanatory, it refers to accessing services over the web. But, there's more to it than that, the current use of the term refers to the architecture, standards, technology and business models that make web services possible. In this article we will use a definition of web services taken from an IBM web services tutorial [2]:

Web services are a new breed of Web application. They are self-contained, self-describing, modular applications that can be published, located, and invoked across the Web. Web services perform functions, which can be anything from simple requests to complicated business processes.

In other words, web services are interoperable building blocks for constructing applications. As an example, we can imagine a distributed digital library infrastructure built on web services providing functionality such as distributed search, authentication, inter-library loan requests, document translation and payment. These web services would be combined by a particular digital library application to offer an environment for reaching information resources that is tailored to its particular user community.

Where the current web enables users to connect to applications, the web services architecture enables applications to connect to other applications. Web services is therefore a key technology in enabling business models to move from B2C (Business to Consumer) to B2B (Business to Business).

Web Services Architecture

There was a reason for choosing a distributed digital library as our example, as well as being a problem domain that is well suited to the web services approach, the architecture of a digital library is very similar to the architecture required to support web services. The digital library community is used to applying verbs such as publish and locate to documents. The web services architecture applies such verbs to applications. In fact, the web services infrastructure can be thought of as a distributed digital library for services rather than information. This means that many of the issues that the digital library community has been looking at, such as metadata for discovery, authentication and authorisation, and business models for accessing intellectual property, are also applicable to web services and must be addressed within a web services context.
The web services architecture has the additional problem of interacting with services once they have been accessed. This is somewhat of an issue for electronic information resources, you need to know how to present the resource to a user, but the infrastructure for this is well established, you can send a MIME type with a document and then use an application that knows how to deal with that document type. The problem with services is that there are potentially many more types of service than types of document and that the interactions with web services are more complex. It's not currently realistic to imagine that an application can discover a completely new kind of service and start interacting with it in a complex manner. The web services architecture therefore requires clients to know in advance the type of service that they will be talking to, but they can dynamically discover a particular implementation of that service. This is like a cross-searching application that combines the results of multiple search-engines that support a particular search protocol, the protocol is known in advance but the particular search-engines could be selected dynamically from a search-engine registry. The cross-searching application is able to use services from applications that didn't even exist when it was written, provided they use the same protocol and can be discovered via a registry.

IBM has published its web services architecture [3] which captures the infrastructure required to support web services in terms of three roles - service provider, service requestor and service registry - and the verbs describing the interactions between them: publish, find, bind (Figure 1). Bind is the step that allows an application to connect to a web service at a particular web location and start interacting with it.

![Figure 1: IBM web services architecture](image)

In this architecture a service is an implementation of a service description and a service description is the metadata describing the service. This metadata must include sufficient information for a service requestor to access the service it describes including its interface and location; resource discovery metadata such as classification may also be included.

A service provider publishes a service description to a service registry. A service requestor then
finds the service description via the service registry, the services description contains sufficient
information for the service requestor to bind to the service provider to use the service. (The
architecture does allow the restricted case where a service requester has access to the service
description via some other means such as hard-coding so that it can go direct to the service
provider.)

The interface part of the service description is known to an application developer so that the
application can be implemented to communicate with services of that type. This is like the developer
of a meta search service knowing the search protocol used by the search services it aggregates. The
protocol would be captured via a web service interface. However, the specific implementations of the
interface, the actual services, are discovered at runtime via a service registry possibly through the use
of resource discovery metadata contained in the service description. For example, a meta search
engine for the medical domain might use a service registry to dynamically find medical search
services that implement the required interface. The matching service descriptions obtained from the
service registry would include sufficient information for the meta search engine, the service
requestor, to bind to each of the service providers in turn and access the search service according to
the interface (Figure 2).

---

**Figure 2: Web services meta search engine example**

**Web Services Standards**

Although the web services architecture can be considered independently of any particular standards,
clearly interoperability is required for largescale adoption of the architecture. A number of key
industry leaders have been working to develop a set of XML-based open standards that enable the
web services architecture to be implemented.

Firstly, a standard way of capturing service descriptions is necessary. The web services description
language (WSDL) [4] has been developed for this purpose. WSDL describes a service as a set of 'ports' which group related interactions that are possible between the application (service requestor) and the web service (service provider). The interactions that are possible though a port are described as 'operations' which may have an input message and optionally a resulting output message. Each operation describes a potential interaction with the web service. This may be a request from the application to the web service. It could also be an interaction that can be initiated by the web service for which the application needs to take action. Interactions in either direction can be one-way or can require a response to be sent.

For example a digital library search service might offer a search port with operations for keyword search (input message contains a simple text string) and advanced search (input message contains a number of field-value pairs), in both cases the output message would contain a list of matching result descriptions.

There are two different kinds of user for WSDL documents. During development of an application that will use a web service, the developer needs to know the interface to the service that the application will bind to. When the application is running it needs details of a specific implementation of that service so that it can bind to it. WSDL can be used to specify both interfaces and their implementations.

We won't look at the format of WSDL here, although if you are familiar with XML you can probably guess roughly what it looks like.

WSDL describes a service in terms of possible interactions with it. A WSDL document provides the potential information content of interactions with a web service but doesn't explain how to communicate that information between an application and a web service. For this purpose, WSDL allows a 'binding' to be specified, in practice this is likely to be another XML-based standard, SOAP [5].

The Simple Object Access Protocol (SOAP) is a standard for XML-based information exchange between distributed applications. Although other transports are possible, SOAP is typically transmitted over HTTP providing a platform for communication with/between web services.

In our digital library example, we know, from the WSDL description, that the search service offers a keyword search operation which takes a text string as its input message. SOAP allows an application to invoke this operation with the specific text string "web services" by sending a SOAP message and to get the result. SOAP specifies the format of the request and response XML documents which are sent over HTTP. There is a standard way of encoding WSDL messages in SOAP to achieve interoperability.

With WSDL and SOAP in our toolkit we can describe web services and use web services from an application. But, how did we know what web services were available in the first place? This is the purpose of Universal Discovery, Description and Integration (UDDI) [6]. UDDI is a specification for distributed registries of web services.

A UDDI web services registry is itself a web service which can be accessed via SOAP from an application that wishes to discover web services. UDDI specifies interfaces for applications to publish web services (as WSDL documents) and to discover web services (via their WSDL documents).

A UDDI entry actually contains more that just a WSDL interface and implementation, it can also include further metadata such as quality of service parameters, payment mechanisms, security and keywords for resource discovery.

With these standards we have the infrastructure to publish (WSDL, UDDI), find (WSDL, UDDI) and bind (WSDL, SOAP) web services in an interoperable manner.

**Where to Find Further Information**
This article has provided an overview of the web services architecture and associated standards. If you want to implement a web services based solution then you will need further information. A good starting point is the IBM Developer Works site which hosts a number of web services articles by various authors [7].

References

1. World Wide Web Consortium Workshop on Web services
2. Web services -- the Web's next revolution
3. IBM Web services Conceptual Architecture
4. Web Services Description Language (WSDL) 1.1
5. Simple Object Access Protocol (SOAP) 1.1
6. uddi.org
7. IBM developerWorks Web Services Zone

Author Details

Tracy Gardner
Advisory Software Engineer
IBM United Kingdom Laboratories (Hursley)

Email: tgardner@uk.ibm.com
Web site: www.hursley.ibm.com

Copyright IBM Corp. 2001