Web Services

Eric Wohlstadter[1], Stefan Tai[2]
[1]Department of Computer Science, University of British Columbia, Vancouver, BC, Canada
[2]Karlsruhe Service Research Institute, Universität Karlsruhe (TH), Karlsruhe, Germany

Synonyms

e-Services

Definition

Web services provide the distributed computing middleware that enables machine-to-machine communication over standard Web protocols. Web services are defined most precisely by their intended use rather than by the specific technologies used, since different technologies are popular [1]. Web services are useful in a compositional approach to application development; where certain key features of an integrated application are provided externally through one or more remote systems. Additionally, Web service standards are a popular platform for wrapping existing legacy applications in a more convenient format for interoperability between heterogeneous systems. To provide interoperability Web services should follow standards for formatting application messages, describing service interfaces, and processing messages. Two popular technology choices discussed in this article are the SOAP [5] based services and the REST (REpresentational State Transfer) [4] based services.

In contrast to traditional World Wide Web (WWW) resources, Web services decouple the user interface from the underlying programmatic interface, to provide an application programming interface (API) to clients. This API provides the means through which Web services expose remote computational resources or informational content over the Internet. Web services are generally designed to follow a service-oriented architecture which promotes the loose coupling useful for interaction in a wide-area environment. This loose coupling comes at the price of giving up more powerful mechanisms such as stateful objects which introduce tighter coupling [2,7].

Historical Background

Web services address enterprise application integration (EAI) in a wide area environment. Historically, they were designed in response to the need for a simpler base of standards than provided by distributed object computing. The original WWW was also unsatisfactory due to its tight coupling with the user interface.

Distributed object computing provides the abstraction of remote stateful objects whose methods can be invoked transparently by clients. This form of remote method invocation is popular in proprietary distributed systems but has never been widely deployed on the Internet. A standardized technology known as the Common Object Request Broker Architecture (CORBA) is commonly used to develop and deploy distributed object applications. Although the CORBA platform provides powerful abstractions for application programmers, these abstractions tended to introduce a high level of complexity in the underlying infrastructure. Consensus was never reached on several important technical problems introduced by the stateful nature of objects such as distributed garbage collection and object identification [2,7]. Thus, distributed object computing is more appropriate in single enterprise deployment scenarios (e.g., telecommunications and military applications).

With the rise in acceptance of XML, it made sense to leverage this format as a means of machine-to-machine communication. In creating new standards around this format, certain problematic features such as stateful objects were not included in the newly developed standards leading to more lightweight and loosely coupled Web services.

The WWW was created as a collaborative platform for distributing information through web pages: documents with text and images including the capability to link between related content. The standard format for web pages, Hypertext Markup Language, included

mixed elements of both informational content and user interface design. With the tremendous popularity of the WWW, came the opportunity to bring rich up-to-date informational data sources to a large audience. This also required complex business processes to manage requests for data. Since the retrieved data was tangled with user interface code, it was difficult to separate the two concerns for users interested in post-processing the retrieved information. "Wrappers" had to be used; to parse, or "scrape" useful content from delivered pages. To simplify the distribution of information, some web sites began offering remote APIs for information retrieval which has led to the interest in standardized Web services.

Scientific Fundamentals

Since interoperability is a key element for any Web service, some standardized approach must be taken to export service functions. The most formal approach to building Web services is built around two core standards known as SOAP and the Web Services Description Language (WSDL) [8]. The standards are being developed by organizations such as the W3C and OASIS. Other lightweight and dynamic approaches are generally categorized under the moniker of REST services. The REST approach requires few standardized technologies beyond HTTP. For both approaches, the standard way to identify each service is through means of an Internet Uniform Resource Identifier (URI). To locate an existing service, a URI is typically obtained out-of-band through informal means, although other automated approaches to service discovery are being explored.

In their most basic form Web services provide only the basic primitives for exchanging documents between clients and servers. Standards for building robust, transactional, and secure services have been proposed but are not currently widely deployed. The remainder of this article focuses on the core Web services fundamentals and describes how both SOAP/WSDL and REST deal with the details of service implementation: message formatting, interface description, and message processing.

Web services use a human readable message format for transport of application specific data. This practice simplifies the task of debugging and loosens a client's reliance on complex implementations for marshalling data. A popular message format is XML due to its self-described nature and abundance of third-party support for processing and storage. The use of XML is mandated by the SOAP/WSDL approach and is popular in REST services also. Conversion of XML to data structures native for a specific programming language can be done through standardized mappings. These mappings are supported by middleware which automate the mapping process. When REST services are consumed directly from a browser agent through JavaScript, a format known as JSON (JavaScript Object Notation) [3] can be used. Unlike XML, JSON provides a direct human readable serialization of JavaScript objects. This can reduce problematic mismatches between the differing type systems of XML Schema [10] and JavaScript.

As an API, Web services often provide a number of distinct functions each with their own input requirements and output guarantees. This interface can be specified formally or informally. Formal description of Web service interfaces is provided through WSDL. WSDL is an extension of the XML Schema specification and most importantly provides description of the set of functions exposed by a service. WSDL specifications add the possibility of providing statically typed-checked service use for clients. This is done by implementing message format mapping through a code generation and compilation process.

With REST, the inputs to service functions are specified as standard web URIs, making use of URI parameters for data input. Output can be produced using any standard HTTP supported encoding and XML is often used for this purpose. The original description of the REST architecture placed considerable emphasis on the fact that services should be described as a set of resources which could be manipulated using predefined HTTP methods (i.e., GET, POST, DELETE, etc.), rather than an application specific set of functions. However in practice, the term REST more commonly refers to any service accessible with little technology beyond HTTP, whether or not the service is modeled primarily as resources or functions. REST currently provides no formal treatment for description of services, although discussion of a standard REST description language is underway as of this writing.

Although not required, many Web services middleware platforms are built around a common architectural style referred to as the Chain of Responsibility pattern [6]. On a local scale incoming and outgoing Web service messages pass through a series of components, called handlers or interceptors, before being passed to lower network layers. This architecture promotes loose-coupling within the middleware itself.

Handlers addressing separate concerns such as reliability, transactions or security can easily be plugged into the middleware and configured on a per application basis. A large number of standards known as WS-* [9] are being developed to specify the formats and protocols for managing these concerns in middleware implementations.

SOAP provides a standard for encapsulating XML based Web service messages with an envelope that can be used to communicate such extra-functional information such as security tokens, time stamps, etc. Using SOAP, the Chain of Responsibility architecture can be extended to a distributed series of message processing steps. In the distributed scenario each step can be handled at a potentially different intermediary location on the network, giving rise to Web service intermediaries. SOAP also standardizes the fault handling semantics for undeliverable messages and provides a number of message exchange patterns. These patterns capture certain reliability guarantees which can potentially be of use by the underlying middleware for optimizing resource usage.

REST messages are transported according to the semantics of the Hypertext Transport Protocol resource access methods. REST architectural principles mandate the use of stateless protocols wherever possible. REST message processors should consider only point-to-point connections as opposed to an end-to-end connection. This simplifies the interposition of caches or other intermediaries for message processing. REST services should rely only on standardized HTTP message headers for communicating extra-functional processing instructions, to keep implementation infrastructure lightweight.

Key Applications

Web services expose data and application functionality using standard Internet technology for consumption by clients over the Web. For example, Web services are used to provide access to very large product data. Today, the range of Web services offerings spans from very simple services to more advanced offerings in support of specific business requirements. Simple services include format conversion services, transformation services and real-time information services such as news feeds and stock quotes. Business offerings include the full range of e-commerce, CRM, marketing, finance, and communication services, among them verification services such as credit checking, accounting services, or customized notification services. Further, entire business processes such as the shipment of goods are provided as Web services.

The modular nature of Web services supports the de-composition of application functionality and business processes and their (re-)composition into new applications. Services composition applies both within an organization and across different organizations. Compositions can take various forms, including workflow applications that require the use of a process composition language and centralized workflow engine. Web services in this way support EAI using standard Internet technology. Another form of composition are Mashups, re-purposing Web-accessible content and functionality in new Web applications that provide an interactive (human) end-user experience. For example, a Mashup may visualize location information of social event feeds on a map.

Web services are also being used in re-architecting middleware software platforms as service-oriented architectures. Middleware services such as message queuing and data storage are provided as Web services; middleware functionality can thus be externalized and used as a remote service. This model targets small and medium sized enterprises in particular, but applies equally to larger size enterprises as well. More advanced services offerings include application hosting and execution services, which allow clients to have entire software applications hosted and executed externally.

Future Directions

Today, there exists a variety of Web services protocols, APIs, languages, and specifications. Different combinations of Web services technology are being used for different purposes. For example, SOAP and Web services specifications in support of security and reliability are used in enterprise computing, whereas REST services are popular for simple services like data access and applications in the social Web.

From a business viewpoint, Web services establish a new market for trading and contracting electronic services that complement and transform the traditional consulting services and software market. The development of this market today is accompanied by the emergence of intermediaries for buyers and sellers to exchange services. These intermediaries distinguish themselves using different pricing mechanisms, co-marketing efforts, billing and accounting management, services usage monitoring, data integration models,

and infrastructure quality guarantees for services availability. New Web services technology in support of this business development is expected to emerge.

Cross-references

► Service Oriented Architecture, ► Request Broker, ► Interface, ► Discovery, ► Enterprise Application Integration, ► OASIS, ► W3C, ► Coupling and De-Coupling, ► Loose Coupling, ► Tight Coupling, ► SOAP, ► AJAX, ► Web 2.0 (3.0), ► Mashups

Recommended Reading

1. Alonso G., Casati F., Kuno H., and Machiraju V. Web Services: Concepts Architectures and Applications. Springer, Berlin, 2003.
2. Birman K. Like it or not, web services are distributed objects. Commn. ACM, 47(12):60–62, 2004.
3. Crockford D. The application/json media type for JavaScript object notation. Network Working Group, RFC 4627, 2006.
4. Fielding R. Architectural Styles and the Design of Network-based Software Architectures. Ph.D. Dissertation, University of California, 2000.
5. SOAP Version 1.2. W3C Recommendation, 2007.
6. Vinoksi S. Chain of responsibility. IEEE Internet Comput., 6(6):80–83, 2002.
7. Vogels W. Web Services are not Distributed Objects. IEEE Internet Comput., 7(6):59–66, 2003.
8. Web Services Description Language Version 2.0. W3C Recommendation, 2007.
9. Weerawarana S., Curbera F., Leymann F., Storey T., and Ferguson D. Web Services Platform Architecture. Prentice Hall, Upper Saddle River, NJ, 2005.
10. XML Schema. W3C Recommendation, 2004.