UDC 681.3

A. S. BORANBAYEV

OPTIMAL METHODS FOR JAVA WEB SERVICES

This article introduces the major components of the Web services architecture. It talks about how Web services standards are used to solve problems in business situations, and gives a brief overview of the major Web services concepts.

1. Overview.

This article is for IT personnel involved with Web Services as well as anyone interested in understanding the web service environment used today. Web Services as a software system are designed to support interoperable machine to machine interaction over a network in real time [1].

Web Services allow applications at different locations to communicate with one another. Applications that use Web Services for interaction can run on different platforms and be programmed using languages that are non-compatible to each other. An example would be a Java application sharing information with a .NET application via Web Service. This makes Web Services extremely portable. Web Services use XML (Extensible Markup Language) to code and decode the data and SOAP (Simple Object Access Protocol) to transport it using open protocols.

Web services help solve the interoperability problem by giving different applications a way to share business logic in real time. A real world example of the interoperability with Web Services would be an accounting department that uses Win2000 Server billing system connecting with the IT Suppliers that use UNIX Server.

Interoperability is not the only reason to use Web Services. They also provide reusable application components. Reusable methods that perform business functionality, which are exposed to consumers, are referred to as operations. A group of these operations is called a service. There are certain things that different applications need very often. That is why there is no reason to make pieces of an application over and over again, when a service for that functionality is available.

This document will talk about how Web services standards are used to solve problems in business situations, and will give a brief overview of the major Web services concepts and standards.

2. How are Web Services Used?

Web Services are widely used in various companies to help solve real business problems, and provide an array of services open to consuming applications. A commonly used Web Service would be Lookup Employee Information by RNN (Tax payee identification number). This returns the Employee details for a given Employee RNN. We can also lookup Employee by Employee ID, in this case it returns the Employee details for a given Employee ID.

Web Services are not limited to look up type functionality. There are Web Services consumed by Rate and Status that will create users in Authenticator when a new employee is hired.

The principles behind web services are simple [2]:

1. The web service provider defines a format for requests for its service and the response the service will generate.

2. A computer makes a request for the web services across the network.

3. The web service performs some action, and sends the response back.

In a real world example this action might be validating a credit card number, retrieving a stock quote, finding the best price for a particular product on the internet, or translating some text to another language. Regardless of the service all communication between are done through SOAP.

3. What is SOAP?

The Simple Object Access Protocol (SOAP) is a protocol for exchanging XML-based messages over computer networks, normally using HTTP/HTTPS [3]. This is the common language of Web Services. It is used as an extensible message envelope format, with «bindings» to underlying protocols.

SOAP messages are sent back and forth between the service provider and service user in SOAP envelopes, containing a request for some action and the result of that action. SOAP envelopes are XML formatted, and are easy enough to decode.

The following examples show a SOAP request and response generated when invoking a web service operation that calculates the cost to mail a package based on the specified zip code, and the weight of a package.

SOAP request:

The SOAP envelope body (soap:Body) includes the operation name (calculate), and the operation parameters (zipcode and weight).

```
<?xml version=»1.0" encoding=»utf-8" ?>
<soap:Envelope>
 . . .
 <soap:Body>
 <calculate>
 <zipcode xsi:type=»xsd:int»>53187</zipcode>
  <weight xsi:type=»xsd:float»>0.75</weight>
 </calculate>
</soap:Body>
</soap:Envelope>
```

SOAP response:

In the SOAP envelope body, the SOAP response includes the result value.

```
<?xml version=»1.0" encoding=»utf-8" ?>
 <SOAP-ENV:Envelope
   <!— namespace declarations —> >
 <SOAP-ENV:Body>
   <calculateResponse>
     <calculateResult xsi:type=»xsd:float»>1.08</
calculateResult>
    </calculateResponse>
  </SOAP-ENV:Body>
 </SOAP-ENV:Envelope>
```

Thus, through the help of SOAP we can construct a request to get some work done, and have a consistently formatted XML response returned across the network.

4. What is a WSDL?

WSDL (Web Services Description Language) is a specification defining how to describe web services in a common XML grammar [4]. It is an XML based service description binding that describes how the service is bound to a messaging protocol. It uses the SOAP messaging protocol. The WSDL defines services as collections of network endpoints, or ports.

WSDL describes four critical pieces of data:

– Interface information describing all publicly available functions

– Data type information for all message requests and message responses

– Binding information about the transport protocol to be used

– Address information for locating the specified service

WSDL represents a contract between the service requestor and the service provider, in much the same way that a Java interface represents a contract between client code and the actual Java object. The crucial difference is that WSDL is platform (and language) independent and is used to describe SOAP services.

Using WSDL, a client can locate a web service and invoke any of its publicly available functions. With the help of WSDL tools, we can also automate this process, enabling applications to easily integrate new services with no manual code. WSDL therefore represents a cornerstone of the web service architecture, because it provides a common language for describing services and a platform for automatically integrating those services.

4.1 Style and Use.

A WSDL SOAP binding can be either a Remote Procedure Call (RPC) style binding or a Document Style binding. A SOAP binding can have an encoded use or a literal use. This gives us four style/use models:

1. RPC/encoded
2. RPC/literal
3. Document/encoded
4. Document/literal

The terms used to name the style/use does not imply that the RPC style should be used for RPC programming models and that the Document Style should be used for document or messaging programming models. The style has nothing to do with a programming model. It merely states how to translate a WSDL binding to a SOAP message. We can use either style with any programming model. The style and use that we focus on in this document - are document/literal.

5. What is JAX-RPC (JSR-101)?

JAX-RPC stands for "Java API for XML-based Remote Procedure Calls". JAX-RPC allows invoking from a Java application a Java based Web Service with a known description while still being consistent with its WSDL description.

JAX-RPC uses SOAP and HTTP to do RPCs over the network. RPC stands for remote procedure calls, and is used for making procedure or function calls and receiving the responses. The SOAP specification defines the necessary structure, a convention for doing RPCs, and its corresponding responses. The RPCs and responses are transmitted over the network using HTTP as the primary transport mechanism.

From an application developer's point of view, an RPC-based system has two aspects: the server side (the Web service) and the client side. The Web service exposes the procedures that can be executed, and the client does the actual RPC over the network.

As discussed above, a Web service environment is based on open standards such as SOAP, HTTP, and WSDL. It is therefore possible that a Web service or a client wasn't developed using the Java platform. However, JAXR-RPC provides the mechanism that enables a non-Java client to connect to a Web service developed using Java platform, and vice versa [9].

The runtime protocol works as follows:

1. A Java program invokes a method on a stub.
2. The stub invokes routines in the JAX-RPC runtime system.
3. The runtime system converts the remote method invocation into a SOAP message.
4. The runtime system transmits the message as an HTTP request.

The consumer becomes aware of the provider services through the machine readable WSDL file. Construction of the consumer relies on the WSDL from the provider.

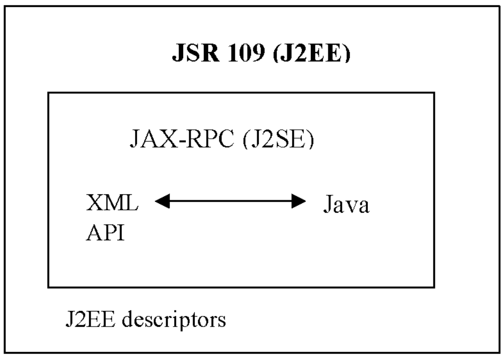
JAX-RPC is WS-I compliant and is widely accepted among the industry. WS-I stands for "The Web Services Interoperability Organization". WS-I is an industry consortium created to promote interoperability among the stack of web services specifications [13].

6. JSR-109 (Web Services in the J2EE Environment).

There are many standards in the areas of Web Services' lifecycles, security measures, and transactions. JSRs have been created to begin the process of defining the different aspects of how Web services might be supported in a J2EE-compliant application server [10].

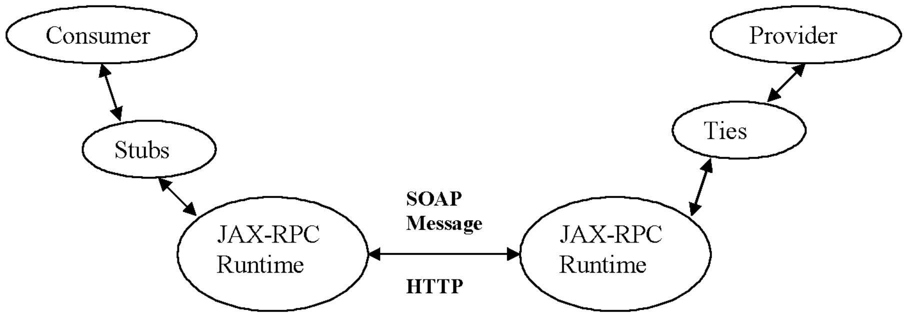
JSR-109 specifies the web services programming model and architecture for J2EE (Java 2 Enterprise Edition). JSR-109 builds on SOAP 1.1 and WSDL 1.1 to cover the use of JAX-RPC in a J2EE environment. It also defines a deployment model to J2EE application servers. JSR 109 specifically discusses local client access to Web services, server lifecycle, and deployment of Web services [10].

JSR 109 and JAX-RPC:



JAX-RPC 1.1 and JSR 109 are part of J2EE 1.4.

7. A Deeper Look at How JAX-RPC Works.
Application Model:



– Stubs are local objects that represent the remote procedures.

– Ties are classes that reside on the provider and enable communication with the consumer.

It is assumed that the consumer is aware of the web service and the remote procedure that it can execute on the web service [9].

1. The consumer calls the method on the stub that represents the remote procedure.

2. The stub executes the necessary routines on the JAX-RPC runtime system.

3. The runtime system converts this method call into a SOAP message and transmits the message to the provider as an HTTP request.

4. The provider, upon receipt of the SOAP message, invokes the methods on the JAX-RPC runtime.

5. The JAX-RPC runtime converts the SOAP request into a method call.

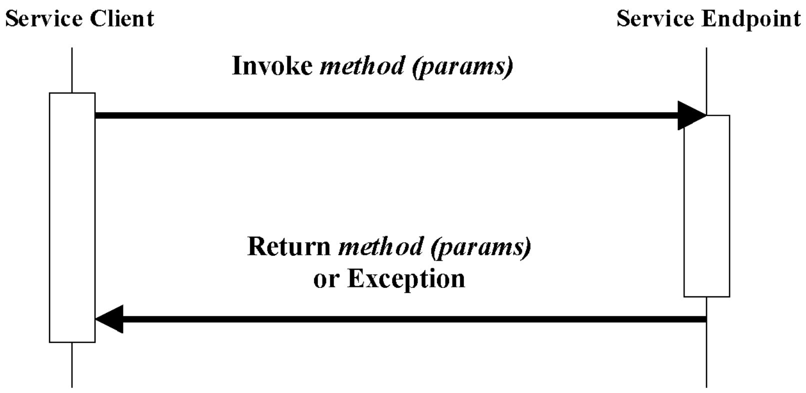
6. The JAX-RPC runtime calls the method on the tie object.

7. The tie object calls the method on the implementation of the web service.

8. The response to the RPC call is sent in a SOAP response message as an HTTP response.

The mode used for a JAX-RPC service is synchronous. Synchronous Request-Response: The client invokes a remote procedure and blocks until it receives a return or an exception.

Synchronous
Request-response Mode



8. Security.

Security restricts the consumers allowed to access the service through authentication. Consumers connect to a service but are restricted from executing specific services. Security needs to be implemented on the Provider and Consumer.

Web Services should incorporate a framework that handles security. The purpose of web services security framework is to provide a means to secure web services. The framework provides authentication and authorization capabilities. When the security framework is used authentication is always performed. All consumers accessing web services must provide valid ID and password. Authorization is optional - meaning an operation can be setup so that it is accessible by all consumers and other operations can be setup so that only few consumers can access it [12].

The framework uses Username Token Profile 1.0 standards from OASIS [11]. This standard specifies the format in which user credentials are communicated between the consumer and the provider.

The security framework is based on JAX-RPC handler architecture. The server/provider can be configured via a configuration file to control access to services that it provides. The Web Services Security Framework utilizes a “keystore.xml” file. The “keystore.xml” has the password and login for the particular service. Consult Web Services Security Framework Guide for Provider (Server) using WebSphere 6.1 if the data transferred service needs to be secured as well.

9. Monitoring.

Web Services are complex and can be difficult to manage and monitor. Web service integration involves multiple systems and people; it introduces some complexities of managing and monitoring. Service Consumers need to know how their systems would behave when using web services, and to be able to design and provide their business performance efficiently. Service Providers need to know the usage patterns and usage load of their services so that they can predict the loads, allocate appropriate resources, and plan for expansion if and when needed. To enable the above mentioned features, there are various monitoring tools used for service management and reporting, which provide more visibility into Web Services.

We always want to be alerted and informed about the abnormal usage or response times so that we can take appropriate action in time. We want to be able to compile historical data to measure how well Web Services are performing over time. This information is important for designing network, designing applications, capacity planning, adhering to Service Level Agreements (SLA), estimating future loads and trends of usage, and presenting insight to the business users on how the business services are being used so that they can make their business more effective and available.

10. Approach for Web Service Design.

Many companies use a bottom up approach for the provider. Bottom up is a strategy for building the service from a Java bean in the project. We usually start by developing the class that will be exposed as a Web Service. Once the class is written, we use the wizard found in the IDE (Integrated Development Environment) to automate the transition from class to service [8]. This will take the written code and generate a WSDL file from it.

When designing the classes that will be exposed as a service or use a service we usually keep in mind a few rules:

– The service name should match the business functionality, with a clear and concise method name.

– We should make our Web Service reusable across all applications. Web Service should not cater to a specific application.

– A consumer calling a service should be autonomous, no other functionality should be executed then forced to wait for the service to return.

– Consumers should always implement a timeout mechanism when calling a service.

What not to do with Web Services:

– Web Services are not a large data transport technology. We should not use a Web Service like an FTP server.

– Web Services hold no state. State should be preserved on the client application side. Web Services perform a specific operation and return a value. They should not retain the information sent to them.

– Application functionality should not tie up resources while waiting for a Web Service to return. Applications should be safeguarded with “timeout” ability in case the Web Service hangs.

11. Summary.

In this paper, we have discussed both the benefits of Web services for business applications and the state of standards development for these services.

At different organizations the term Web Service usually refers to consumers and providers that communicate using XML messages that follow the SOAP standard. There is also a machine readable description of the operations supported by the server, a description in the Web Services Description Language (WSDL). Many organizations prefer using the JAX-RPC with a "document/literal style and use" to allow more flexibility in the WSDL and overall service itself.

Web Services are Web APIs that can be accessed over a network in real time, such as the Intranet via HTTP or HTTPS, and executed on a remote system hosting the requested services. For the most part Web Services are synchronous but by nature are not limited to it.

It is always a good idea to test and monitor all Web Services, even if they are used internally. There are monitoring tools that will monitor services and help the provider develop their informal and formal SLA.

REFERENCES

1. Web service, from Wikipedia, the free encyclopedia: http://en.wikipedia.org/wiki/Web_service

2. Web Services Activity - http://www.w3.org/2002/ws/

3. Specification SOAP - http://www.w3.org/tr/soap

4. Specification WSDL - http://www.w3.org/tr/wsdl

5. IBM Developers Work zone (articles/examples/ about J2EE technologies, XML technologies, and web services) - http://www.ibm.com/developerWorks

6. Various XML projects Apache Software Foundation - http://xml.apache.org/

7. Website Sun Microsystems, about Java - http://java.sun.com/

8. http://en.wikipedia.org/wiki/Integrated_development_environment

9. Chapter 11: Working with JAX-RPC from the book «Java APIs for XML Kick Start» by Aoyon Chowdhury and Parag Choudhary, published by Sams Publishing.

10. JSR 109: Web Services Inside of J2EE Apps, written by Al Saganich, 08/07/2002. Published on ONJava.com: http://www.onjava.com/pub/a/onjava/2002/08/07/j2eewebsvs.html

11. Web Services Security, UsernameToken Profile 1.0, OASIS Standard 200401, March 2004: http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0.pdf

12. Systinet Server Web Services Security framework: http://www.systinet.com/doc/ssj-60/ssj/secguide.html

13. Web Services Interoperability, from Wikipedia, the free encyclopedia: http://en.wikipedia.org/wiki/WS-I

Резюме

Мақала веб-сервистердің архитектурасының негізгі компоненттеріне арналған. Практикалық есептерді шешу үшін веб-сервистердің стандарттарының қалай қолданылатындылығы көрсетілген, веб-сервистердің негізгі концепциясының сипаттамасы берілген.

Резюме

Статья посвящена главным компонентам архитектуры веб-сервисов. Описывается, как можно использовать стандарты веб-сервисов для решения практических задач, дается описание основных концепции веб-сервисов.

L.N. Gumilyov Eurasian National University

Поступила 2.07.06г.