Developing Internet Computing Applications using Web Services

Srihari Muthyala and Jiang B. Liu
Computer Science & Information Systems Department
Bradley University
Peoria, IL 61615
U.S.A.

Abstract: Service-oriented computing has many advantages over the proprietary distributed object-oriented computing on the Internet. Business can provide the functionalities on different platforms in terms of web services to the clients, where the web services model is modular, self-describing, self-contained applications that are accessible over the Internet. In this paper, we will discuss our experience of implementing the Internet computing using web services on IBM WebSphere and Microsoft .Net platforms.

Key-Words: Internet computing, service-oriented computing, web services

1. Introduction

Internet computing enables large-scale aggregation and sharing of computational, data and other resources across institutional boundaries. It seeks to exploit PCs and workstations to create powerful distributed computing systems with global applications using the ever-expanding resources of the Internet. Today many companies have already made most of their information systems available to all of their divisions and departments, or even their customers or partners on the Web. Thus, there is an increasing demand for technologies that support the connecting or sharing of resources and data in a very flexible and standardized manner.

Over the last few years, the software industry has changed dramatically. Rather than purchasing or licensing software to install on a workstation or a PC, users now prefer to access services over the Internet. The trend moves software away from monolithic, fat-client-hosted and maintained applications toward a new development model that is often called "Web services." Before Web services, Internet computing and e-commerce were based on client/server computing model in which many clients talk to many servers through the net [1-3]. Communicating partners have to go through certain prearrangement in terms of what common object model they have to use or what common communication protocol they have to agree upon in order to share resources. The exchanges of information are through enterprise application integration (EAI). Developers need to create one-time, proprietary solutions for system integration [4]. A new, often makeshift solution had to be developed each time two companies wanted to share resources. Moreover the two companies need know each other's platform in order to exchange information.

There are many technologies developed since then for supporting the Internet computing. Java J2EE and .Net are two such dominant ones [5-6]. In both the platforms, Web services have emerged as the promising next generation of Web-based technology for computing and exchanging information. Web Services is "a revolutionizing enterprise computing" [7-8]. They allow any piece of software to communicate with a standardized XML messaging system on any different platforms. The introduction of Extensible Markup Language (XML) was an important step to simplifying the application integration process. XML enables developers to separate the content of data exposed over the Web from its presentation. Furthermore, the web services model is modular, self-describing, self-contained applications that are accessible over the Internet. This shift towards a service-oriented approach will not only standardize interaction, but also allows for more flexibility in the distributed process. Moreover it increases Internet resources and various remote methods to invoke available

resources. A service-oriented architecture thus has to focus on how services are described and organized to support their dynamic, automated discovery and use. In this paper, we will discuss our experience of implementing the Internet computing using web services.

2. Web Services Essentials

Web services are the various resources available on Internet, which can be identified by searching in Internet registries. Any software application methods can be converted into web services with minor changes. There exists lot of tools to convert the existing EJBs, JAVA objects, and C++ objects into various web services. Web Services makes it easier to call objects or applications in different environments. Web Services is an adaptive evolution of distributed and Internet computing. The advantages of Web Services are the simplicity and flexibility involved as compared with the technologies like EJB, DCOM, and CORBA component-base computing model.

Web Services allow any piece of software to communicate with a standardized XML messaging system. The combination of Web services security, Web services management, and the standards for transactional integrity will allow the construction of long running, secure, shared Web services.

2.1 XML

The introduction of Extensible Markup Language (XML) was an important step to simplifying the application integration process. XML enables developers to separate the content of data exposed over the Web from its presentation. A predefined markup language like HTML defines a way to describe information in one specific class of documents. XML, on the other hand, lets you define your own customized markup languages for different kinds of documents. This means that data can be easily exchanged, not only among humans through Internet browsers, but also among computers. XML lies at the core of web services, and provides a common language for describing Remote Procedure Calls, web services, and web service directories.

The XML main focus is on building up business vocabulary language, defining documents, and gaining industry consensus for B2B applications. The big push was to bring the Web into an enterprise and make data accessible via a browser. All of those transactions require trust and security, making it mission-critical to devise common XML mechanisms for authenticating merchants, buyers, and suppliers to each other, and for digitally signing and encrypting XML documents like contracts and payment transactions. XML opens the door to interoperable information.

2.2 SOAP

Simple Object Access Protocol (SOAP), a key component of Web Services, is the latest of the protocols that allow different applications to interoperate. Although remote objects can give a program almost unlimited power over the Internet, but most firewalls block non-HTTP requests. SOAP, an XML-based protocol, gets around this limitation to provide intraprocess communication across machines. Moreover it uses XML and is therefore text-based protocol, easy to parse and read. Thus improves the information exchange over web. It is rapidly becoming the standard protocol for accessing a Web Service, and accessing the service is key.

2.3 WSDL

Web Services Description Language (WSDL) is an XML vocabulary for describing a Web Service. A WSDL document describes what functionality a Web Service offers, how it communicates, and where it is accessible. WSDL provides a structured mechanism to describe the operations a Web Service that it can perform, the formats of the messages that it can process, the protocols that it supports, and the access point of an instance of the Web Service that it can define.

SOAP development tools can use a WSDL description to automatically generate a SOAP interface. A WSDL description defines a service as a collection of network endpoints or ports. Each port is defined abstractly as a port type, which supports a collection of operations. Each operation processes a particular set of messages. A binding maps a port type to a specific protocol and data format. A port instantiates a port type and binding at a specific network address. A

WSDL description is an XML document that contains a set of definitions.

2.4 UDDI

The Universal Description, Discovery and Integration) (UDDI) provides a mechanism to register and categorize Web Services that you offer and to locate Web Services that you would like to consume. UDDI is itself a Web Service. Just as businesses list their products and services in a telephone directory, Web service brokers use this specification to register services that service requesters can then discover and invoke. Web-based applications interact with a UDDI registry using SOAP messages.

Before Web services can really take root, the UDDI registry, an important catalyst in the evolution of Web services, must be set up. Without UDDI, or other types of Web services directories, Web services could be hard to find.

2.5 Benefits of web services

Web services have been accepted by all organizations and businesses. It is playing a very important role in enterprise computing and information exchanging.

1. Web services are self-contained

On the client side, no additional software is required. A programming language with XML and HTTP client support is enough to get you started. On the server side, merely a Web server and a SOAP server are required. It is possible to Web services enable an existing application without writing a single line of code.

2. Web services are self-describing

Neither the client nor the server knows or cares about anything besides the format and content of request and response messages (loosely coupled application integration). The definitions of the message format travels with the message; no external metadata repositories or code generation tool are required.

3. Web services can be published, located, and invoked across the Web

This technology uses established lightweight Internet standards such as HTTP. It leverages the existing infrastructure. Some additional standards that are required to do so include SOAP, WSDL, and UDDI.

4. Web services are easily accessible

Web services are distributed over the Internet. Web services make use of existing, ubiquitous transport protocols like HTTP, leveraging existing infrastructure and allowing information to be requested and received in real time. Current IT infrastructure for addressing, security and performance can be applied to Web services applications as well.

5. Web services are language-independent and interoperable

Client and server can be implemented in different environments. Existing code does not have to be changed in order to be Web service enabled. Web services permit the use of a vast array of clients—Java, C++, .NET, JavaScript, Perl, and so on.

6. Web services are inherently open and standard-based

XML and HTTP are the major technical foundation for Web services. A large part of the Web service technology has been built using open-source projects. Therefore, vendor independence and interoperability are realistic goals this time.

7. Web services are dynamic

Dynamic e-business can become reality using Web services because, with UDDI and WSDL, the Web service description and discovery can be automated.

8. Web services build on proven mature technology

There are a lot of commonalities, as well as a few fundamental differences to other distributed computing frameworks. For example, the transport protocol is text based and not binary.

9. Web services are loosely coupled

Traditionally, application design has depended on tight interconnections at both ends. Web services require a simpler level of coordination that allows a more flexible re-configuration for an integration of the services in question.

10. Web services provide the ability to wrap existing applications

Already existing stand-alone applications can easily be integrated into the service-oriented architecture by implementing a Web service as an interface.

3. Develop a ShoppingCart Web Service Application

We have implemented a web service of a common e-commerce application Shopping Cart to explore the service-oriented computing. This implementation is based on a Java EJB built using IBM WebSphere studio. Various methods in ShoppingCartManger object are exposed as web service. The WebSphere application developer generates the following WSDL files when the ShoppingCartManager class was converted into a web service:

- ShoppingCartManager.wsdl

This file describes all the parameters used in this service and each methods input and output responses.

- ShoppingCartManagerBinding.wsdl

This file contains service interface; the service interface describes the abstract type interface and its protocol binding.

- ShoppingCartManagerService.wsdl

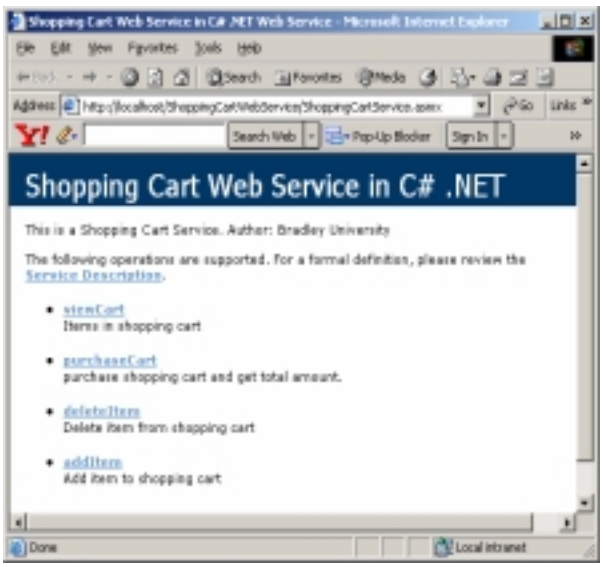
This file contains service implementation, which describes services access information.

ShoppingCartManagerManager.wsdl

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions
name="ShoppingCartManagerService"
targetNamespace="http://myservices.com.wsdl/S
hoppingCartManagerService/"
xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:binding="http://myservices.com.wsdl/Sho
ppingCartManagerBinding/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/s
oap/"
xmlns:tns="http://myservices.com.wsdl/Shoppin
gCartManagerService/">
<import
location="ShoppingCartManagerBinding.wsdl"
namespace="http://myservices.com.wsdl/Shoppi
ngCartManagerBinding/"/>
 <service
name="ShoppingCartManagerService">
 <port
binding="binding:ShoppingCartManagerBinding
" name="ShoppingCartManagerPort">
 <soap:address
location="http://localhost:9081/ShoppingCart/ser
vlet/rpcrouter"/>
</port></service></definitions>
```

After publishing the above web service, users can access the above wsdl file and generated ShoppingCart client to access the shopping cart methods. There are many tools which convert the wsdl file into proxy file, client GUI and even provide testing environment such as WebSphere studio and Virtual Studio .NET).

.NET Test Client



Web Sphere Studio Test Client

4. Web Services across Different Platforms

Interoperability is one of the main promises of Web services. Web services are designed to be

independent of the underlying operating system and programming language. One of the most pressing challenges of Internet computing is the effective integration of heterogeneous information sources. Web service interoperability goals are to provide seamless and automatic connections from one software application to another. SOAP, WSDL, and UDDI protocols define a self-describing way to discover and call a method in a software application regardless of location or platform. Data is marshaled into XML request and response documents and moved between software packages using HTTP or message-based protocols. Thus it improves the data exchanging over web (heterogeneous).

Web services represent an evolving set of standards that will enable diverse and occasionally obstreperous applications to more easily discover each other and seamlessly exchange data via the Internet. For instance, programs written in Java and running on Solaris can find and call code written in C# that run on Windows XP, or programs written in Perl that run on Linux, without any concern about the details of how that service is implemented.

4.1 Example depicts the exchanging information between .NET web service and J2EE web service Client

In this example, the Java J2EE application uses a Microsoft .NET web service to extract database table information using DB2 database where .NET web service exposes a method which returns the list of usernames and their encrypted passwords as a DataSet data structure (Figure 1).

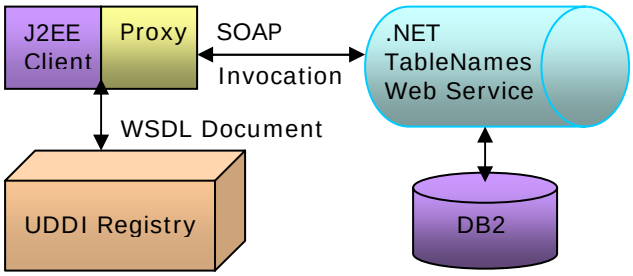


Figure 1: .NET web service that returns a DataSet to J2EE client

The WebServiceClient (java proxy) is generated using an exact location of .NET web service wsdl file. e.g.
http://12.221.232.23/DBWebService1/Service1.asmx?WSDL
The above will return XML formatted information regarding users and encrypted passwords, later it is easy to parse XML message as per requirement.

4.2 Example depicts a heterogeneous system using web services

There are many applications that can help save money and keep employees and customers connected using web services. An example below depicts how various businesses interact with each other for information exchange using web services (Figure 2). If a client requested a data from server and data is scattered over Internet among various business entities then there must be a provision to contact all business and gather required information. This is very tedious job and consumes lot of time and resources. Moreover it requires knowledge about all business platforms. But with the web services it can be very easy to utilize the resources over Internet. As shown in Figure 2 Business 1 can utilize both .NET and J2EE web services without knowing about both BUSINESS 2 and BUSINESS 3 implementations. Thus Businesses can mix and match Web services with any devices in any network with minimal programming.

5. Conclusion

Web Services implement a set of technologies (XML, SOAP, WSDL, and UDDI) that allow users to develop, catalog, and publish business services for delivery and use on the Web. The universal agreement of the technical community on the SOAP specification, based on XML and remote procedure calls (PRCs), provides the backbone of Web Services technology; as now users can access business functionality through system communication regardless of platform, object model, or programming language. The UDDI registry serves as Internet business yellow pages, providing access to published Web Services by potentially anyone with a browser. WSDL serves as an XML vocabulary for describing the Web Services interface, defining the published service operations, and defining the service location and binding details. Lastly, Web

Services use the ubiquitous protocols of the Web, mainly HTTP and e-mail (SMTP), to ensure universal access and delivery. Our implementation of the web services indicated that they are the most powerful technologies for delivering web-based computing on the Internet across different platforms. It will have a great impact on the enterprise application integration towards service-oriented computing paradigm.

References

[1] S. Asbury and S. R. Weiner, Developing java Enterprise Applications, Wiley, 2001.

[2] Jiang B. Liu, "Web based Enterprise Computing Development using J2EE," Industrial Information Technology Handbook, CRC Press, to be published in March 2004.

[3] Jiang B. Liu, "Multi-tiered Internet Computing using Java Technologies," Proceedings of IECON 27th Annual Conference of the IEEE Industrial Electronics Society, pp 1789-1793, Denver, Colorado, December 2001.

[4] I. Charlesworth and T. Jones, "The EAI and Web Services Report," EAI Journal, Vol. 5, No. 3, pp 11-18, March 2003.

[5] H. M. Deitel et al., Java Web Services for Experienced Programmers, Deitel Developer Series, Prentice Hall, 2003.

[6] Microsoft .Net Developer Training Set, MSDN, 2001.

[7] M.P. Papazoglou and D. Georgakopoulos, "Service-Oriented Computing," Communications of ACM, Vol. 46, No. 10, pp 24-29, Oct. 2003.

[8] Jian Yang, "Web Service Componentization," Communications of ACM, Vol. 46, No. 10, pp 35-40, Oct. 2003

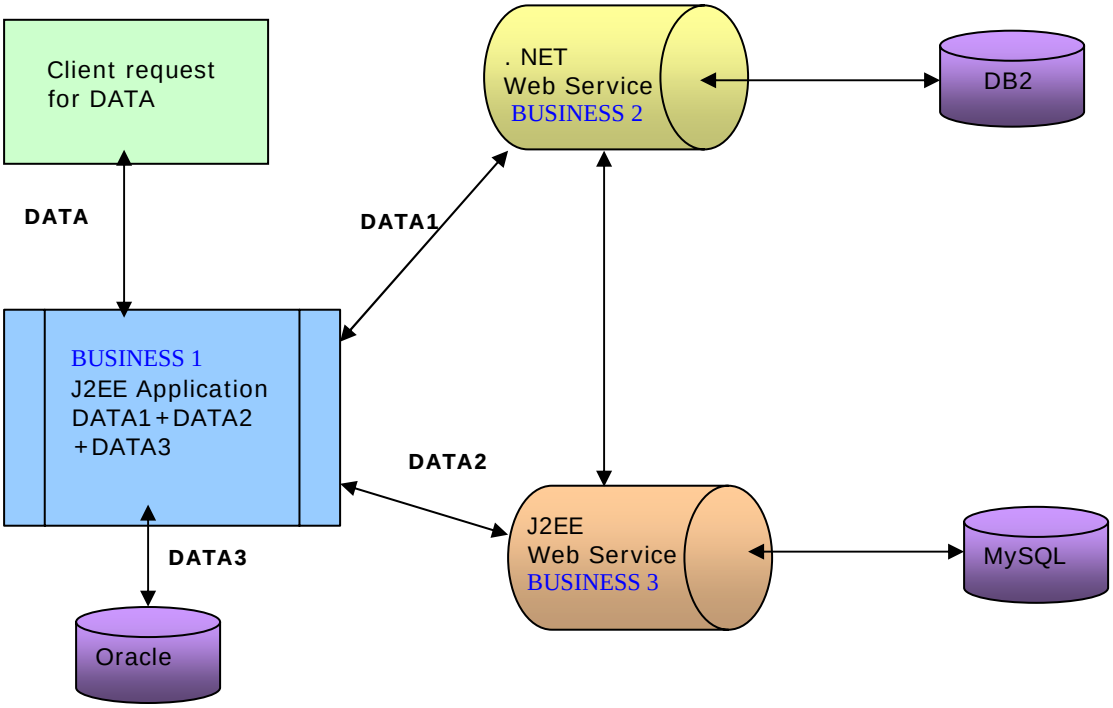


Figure 2: A business application across different platforms using web services.