

# Web Services JOURNAL

Preview Issue | JUNE 2001

WSJ2.COM

**PREMIERING  
at Web  
Services  
Edge**

Sept. 24-25, 2001 New York, NY

EAST

web services **EDGE**  
conference & expo

WEST Oct. 24-25, 2001 Santa Clara, CA

Oct. 22-25, 2001 Santa Clara, CA

**XML** **EDGE**  
conference & expo

Sept. 23-26, 2001 New York, NY

**JDJ** **EDGE**  
conference & expo

From the Editor  
**The Rise of Web Services**  
by Sean Rhody pg. 3

From The Industry  
**Web Services: The Power to  
Change the World**  
by Steve Benfield pg. 22

**SYS-CON  
MEDIA**



**Feature:** Writing Your First Web Service:  
A Tutorial *Bottom-up design and a Web service wrapper*

Andrew McCright

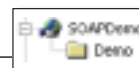
4



**Feature:** Creating Web Services with Microsoft  
Soap Toolkit *A beta of version 2.0 that really delivers*

Wei Meng Lee

10



**Another View:** Where are Web Services Going?  
*And can they deliver on their promise?*

JP Morgenthal

6



**ebXML:** Electronic Business XML: Making Web  
Services Work for Business *Defining a new standard*

David Russell

18



## COMING IN THE PREMIER ISSUE

**Feature:** Deriving SOAP:  
*The eXtensibility of XML and Simplicity of SOAP*



Ali Solehdin

22

**Feature:** Web Services Using  
Pluggable Providers

Mark A. Richman

34



**Feature:** Sending Out-of-Band Messages  
to SOAP-Based Web Services



Mark Moore

46

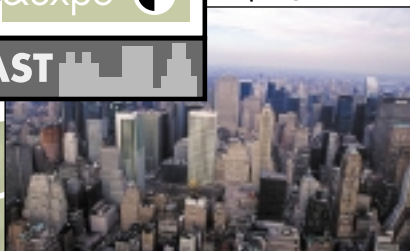
# WEB SERVICES 2001 INTERNATIONAL

Hilton New York, NYC

Santa Clara Convention Center, CA



Sept 23-26, 2001



Co-located with JDJEdge 2001 in NY



Oct 22-25, 2001



Co-located with XMLEdge2001 in CA

Offering over **100** information-packed sessions exploring the implications of Web Services and their impact on business processes.

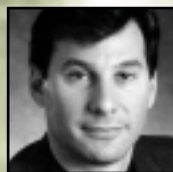
Register by  
June 30, 2001  
**SAVE  
\$600**

## KEYNOTE PANEL: The Web Services Paradigm



**GOSLING**

Creator of Java  
VP & Fellow, Sun Microsystems



**BARATZ**

CEO, Zaplet, Inc.  
Former President, JavaSoft



**ROSS**

Founder, JavaLobby  
Panel Moderator



**LYNCH**

President  
Macromedia



**SCOTT**

CEO, PointBase, Inc.  
Cofounder, Oracle

## A Preview of Conference Tracks

Real-World Web Services:  
Is It Really XML's Killer App?

Demystifying EBXML:  
A Plain English Introduction

Authentication, Authorization and Auditing:  
Securing Web Services

Wireless: Enable Your  
WAP Projects for Web Services

The Web Services Marketplace:  
An Overview of Tools, Engines and Servers

Building Wireless Applications  
with Web Services

How to Develop and Market  
Your Web Services

Integrating XML in a  
Web Services Environment

Real-World UDDI

WSDL: Definitions  
and Network Endpoints

Implementing SOAP-Compliant APPs

Deploying EJBs in a  
Web Services Environment

Swing-Compliant Web Services



## Programmer's Paradise

**BEA**  
**www.bea.com**

# WebServices JOURNAL

## EDITORIAL ADVISORY BOARD:

STEVE BENFIELD, RON BEN-NATAN, GRAHAM GLASS,  
WEI MENG LEE, NORBERT MIKULA, JIM MILBERY  
JP MORGENTHAU, AJIL SAGAR, SIMEON SIMEONOV,  
JAMES TAUBER

## EDITOR-IN-CHIEF: SEAN RHODY

EDITORIAL DIRECTOR: JEREMY GEELAN  
EXECUTIVE EDITOR: GAIL SCHULTZ  
MANAGING EDITOR: CHERYL VAN SISE  
SENIOR EDITOR: M'LOU PINKHAM  
EDITOR: NANCY VALENTINE  
ASSOCIATE EDITOR: JAMIE MATUSOW  
ASSISTANT EDITOR: GREGORY LUDWIG  
EDITORIAL INTERN: NIKI PANAGOPOULOS

## INDUSTRY EDITOR: NORBERT MIKULA

APPLICATIONS EDITOR: RON BEN-NATAN  
PRODUCT REVIEW EDITOR: JIM MILBERY  
XML EDITOR: AJIL SAGAR  
STANDARDS EDITOR: JAMES TAUBER

## WRITERS IN THIS ISSUE

STEVE BENFIELD, WEI MENG LEE, ANDREW MCCRIGHT,  
J.P. MORGENTHAU, SEAN RHODY, DAVID RUSSELL

## SUBSCRIPTIONS

FOR SUBSCRIPTIONS AND REQUESTS FOR BULK ORDERS,  
PLEASE SEND YOUR LETTERS TO SUBSCRIPTION DEPARTMENT.  
SUBSCRIPTION HOTLINE: [SUBSCRIBE@SYS-CON](mailto:SUBSCRIBE@SYS-CON)  
COVER PRICE: \$5.99/ISSUE  
DOMESTIC: \$89.00/YR. (12 ISSUES)  
CANADA/MEXICO: \$99.00/YR. OVERSEAS: \$129.00/YR.  
(U.S. BANKS OR MONEY ORDERS). BACK ISSUES: \$10 EA.,  
INTERNATIONAL \$15. EA.

## PUBLISHER, PRESIDENT, AND CEO: FUAT A. KIRCAALI

VICE PRESIDENT, PRODUCTION & DESIGN: JIM MORGAN  
SENIOR VICE PRESIDENT, SALES & MARKETING:  
CARMEN GONZALEZ

## VICE PRESIDENT, SALES & MARKETING: MILES SILVERMAN

VICE PRESIDENT, SYS-CON EVENTS: CATHY WALTERS

## ADVERTISING ACCOUNT MANAGERS:

ROBYN FORMA  
MEGAN RING

## ASSOCIATE SALES MANAGERS:

CHRISTINE RUSSELL  
CARRIE GEBERT  
ALISA CATALANO

## VICE PRESIDENT, CIRCULATION: AGNES VANEK

NEWSSTAND SALES MANAGER: CHERIE JOHNSON

## ART DIRECTOR: ALEX BOTERO

## ASSOCIATE ART DIRECTORS:

CATHRYN BURAK  
RICHARD SILVERBERG

## ASSISTANT ART DIRECTORS:

ABRAHAM ADDO  
AARATHI VENKATARAMAN

## WEBMASTER: ROBERT DIAMOND

## WEB DESIGNERS: STEPHEN KILMURRAY

GINA ALAYAN  
CAROL AUSLANDER

## WEB DESIGN INTERN: PURVA DAVE

## ASSISTANT CONTROLLER: JUDITH CALNAN

## ACCOUNTS PAYABLE: JOANN LAROSE

## EDITORIAL OFFICES

SYS-CON MEDIA 135 CHESTNUT RIDGE RD., MONTVALE, NJ 07645  
TELEPHONE: 201 802-3000 FAX: 201 782-9600  
[SUBSCRIBE@SYS-CON.COM](mailto:SUBSCRIBE@SYS-CON.COM)

WEB SERVICES JOURNAL is published monthly (12 times a year)  
for \$89 by SYS-CON Publications, Inc., 135 Chestnut Ridge Rd.,  
Montvale, NJ 07645

Periodicals postage rates are paid at Montvale, NJ 07645  
and additional mailing offices.

POSTMASTER: Send address changes to:

WEB SERVICES JOURNAL, SYS-CON Publications, Inc.,  
135 Chestnut Ridge Rd., Montvale, NJ 07645

© PU copyright and Worldwide Distribution sections from JDJ masthead

# The Rise of Web Services

WRITTEN BY Sean Rhody



## Author Bio:

Sean Rhody is the editor-in-chief of Web Services Journal. He is a respected industry expert and a consultant with a leading Internet service company.

Without a doubt, 2001 is a rebuilding year. The market is down, especially the tech stocks. The dot-coms that were leading the charge are now the dot bombs that we're all trying to distance ourselves from. So it might be easy to conclude that the Internet revolution is over, and the bad guys won. But that would be naïve, and more than a little premature.

While we're definitely experiencing a market correction, and feeling the pain of all those NetMarkets that had more chutzpah than marketing savvy, there's still a strong drive within business to move towards a collaborative environment. As the market moves slowly but inexorably towards a more cooperative model, the fits and starts that go along with that move are nothing more than the growing pains of an economy that needs to become global in order to achieve additional growth.

In one sense, Web Services is in its infancy. It's an infant because the Internet revolution and the dot-com craze showed us that there is value to be had in cooperation. The problem is, it's extremely difficult to determine an operating model that accommodates the idea of cooperation and competition while still providing strong leadership.

And leadership is critical for this next stage of business growth. Technology provides the underpinnings, the support structure if you will, for business growth. But as the Internet has laid bare the true nature of industry economies, it has become imperative that businesses adapt their operating models in order to grow.

## THE STAGNATION EFFECT

Part of the dot-com fallout has been a reevaluation of the market. It's become clear to the majority that exchanges, while useful, are not the be-all and end-all of the actual market. Nor is the ability to market product online (in a business-to-business context) so compelling that companies are willing to completely change the way they do business.

Many of these businesses have good reason to continue their current patterns. While compelling from a marketing standpoint, neutral exchanges suffered from several fatal flaws. Neutrality also led to a lack of cooperation from the major industry players. While neutrality is a laudable goal, the liquidity of the market actually drives much more value than any neutral exchange can.

Commoditization is another myth within the exchange industry. The reality of the market is that any true commodity would have already achieved a neutral exchange years ago. To the outsider, steel is steel, and benzene is benzene. But to industry insiders, those who truly know the workings of the industry, these are not commodities but distinct, individual products, with specifications and tolerances that make it very difficult to compare them as apples to apples.

It's precisely the collapse of the dot coms, coupled with the failures of exchanges, that has led to stagnation in the industry. And yet now is precisely the time when a new paradigm is rising from the ashes of dot carnage – witness the arrival of Web Services.

Companies can't go back to doing business in the "traditional" ways. For better or worse the dot com revolution has forever altered the landscape and playing field of the market as we know it. We cooperate with people we compete with. We're willing to participate in online markets, as long as there's a perceived gain. The business rules of today aren't carved in stone – we're lucky if they actually get written to disk before they change. The corporations that can handle this environment will thrive; the others will wish they had adapted.

The driving force behind handling that kind of rapid change will be Web Services. We're at the forefront of the revolution now – and in the coming months we'll be seeing how different companies, some vendors, some start-ups, some Fortune 500 are creating the landscape of the Web Services world.

Web Services will be a critical component in the business plans of all corporations, because it provides a means of addressing some of the basic problems of the industry – communication and cooperation, at a level business people can understand. Assuredly, there will be technology involved, but the real value that Web Services will bring is the ability to unite companies in commerce transactions at the business process level. The work that's been done over the past few years within the open standards community to build systems capable of interoperating will now be leveraged to build ecosystems uniting supply chains across industries. The dot-com is dead. Welcome to the Web Services Era. ☺





# Writing Your

Web service







# First A Tutorial

*Just the first of many fun tools for the enterprising Web developer*

**B**y now, you've read the hype. You know that Web Services is the "next big thing." You know it utilizes the hottest technologies such as SOAP and UDDI, but if you are like me, you are tired of reading white papers, high- level architecture papers, and magazine articles telling you what it is. You want to start writing your own Web service. Your wait is at an end.

## BACKGROUND

In case you have not read all the white papers and architecture documents, it is important to understand Web Services from a high level. Readers familiar with the high-level concepts may wish to skip this section.

Definition: A Web service is any program that is callable by another program across the Web in a platform/language/object model neutral fashion (using standardized Web Services protocols). In short, a Web service is to an application what a Web page is to a person.

The Web Services protocols mentioned above include Simple Object Access Protocol (SOAP); Universal Description, Discovery, and Integration (UDDI); and Web Services Description Language (WSDL).

Written By  
Andy McCright



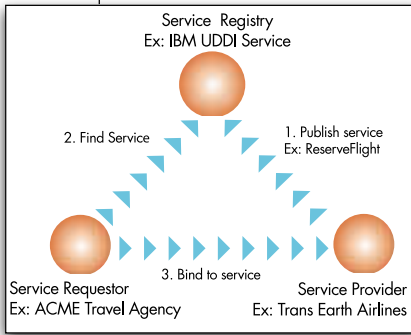
BIO

Andy McCright is a software engineer for IBM software group. His responsibilities include development of the WebSphere Application Server and customer engagement. Recently, he has been focusing on Web Services technologies and how they will play into the next generation of application servers.

 andymc@us.ibm.com



FIGURE 1 Typical Web Service Execution



The process for a typical Web service is described in Figure 1. In this example, the service provider publishes its service interface (a WSDL document) to a service registry (a UDDI provider). A service requester searches the service registry and finds a Web service that will suit its needs. It then binds to the service provider in order to execute the service (using SOAP).

The Web service itself is really nothing more than a Java servlet, bean, EJB, etc. The Web Services architecture is simply a wrapper for accessing this preexisting code in a platform and language-independent manner.

One of the main benefits that Web Services brings to the table is ease of integration. Consider a B2B scenario in which Company A, built upon Microsoft technology, wishes to connect to Company B, built on Java servlets. The process of integrating these two polar technologies requires significant developer time and resources. Things become even more interesting when Company A wishes to interface with a third company, Company C, built on CORBA/C++. Company A must now begin another long integration project. Enter Web Services. Now integration efforts that used to take 12 months may take as little as 12 days.

The key player for reducing integration time is SOAP. SOAP is an XML technology that describes remote procedure calls (RPC). It is similar to Java Remote Method Invocation (RMI) but is platform- and language-independent. Thus, a C++ program can execute methods on a Java program, running on the other side of the planet.

As you can see, the power of Web Services presents countless possibilities. Now let's write our own Web service.

## WHAT DO I NEED?

The examples in this article were written in the IBM XML and Web Services Development Environment (WSDE) and executed in the IBM WebSphere Application Server Advanced Edition 3.5.3 run-time environment. WSDE is available for free download from [www.alphaWorks.ibm.com/tech/wsde](http://www.alphaWorks.ibm.com/tech/wsde).

The WebSphere Application Server 60-day evaluation version is available at [www.ibm.com/software/Webservers/appserv/download.html](http://www.ibm.com/software/Webservers/appserv/download.html).

At the time of this writing, the WSDE was only supported on Windows NT/2000. The evaluation version of WebSphere Application Server will run on the Red Hat Linux and NT/2000 platforms. The WSDE also supports the Apache Tomcat as a runtime environment (<http://jakarta.apache.org/tomcat/index.html>).

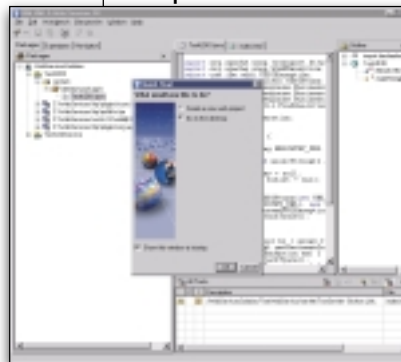
After downloading and installing WSDE and WebSphere, you are ready to begin development.

## FROM THE BOTTOM UP

Our example will focus on bottom-up design strategy. This means that we will develop the application first, and then fit a Web service wrapper on top of it. At least initially, most Web Services development will follow this strategy.

The Web service we will create is a stock quote generator. When given a stock symbol, this service will return the current price for one share. Users of this Web service will probably be portal companies that provide stock advice to their customers.

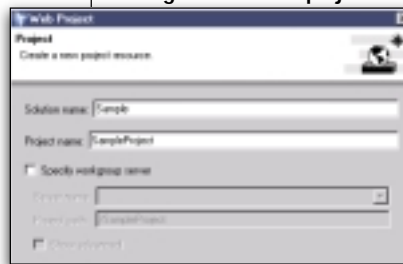
FIGURE 2 The Web Services Development Environment



Open the WSDE environment (see Figure 2). Select "Create a new Web project," and then click OK. Enter solution and project names (see Figure 3). Continue through the wizard with the default values. When you have finished you will have created the Web project in which you will develop your Web service.

Next, change the perspective to Java (from

FIGURE 3 Creating a solution and project



the menu bar select Perspective/Switch To/Java Perspective). This provides a more suitable development environment for coding. In the left pane, switch to the "Packages" tab, and then expand to the "servlets" directory. Right click on servlets and select "New/Class..." to create a Java bean that will return the stock price. Name the class StockQuoteService.

Enter the code for StockQuoteService (see Listing 1; all code listings can be found at [www.wsj2.com](http://www.wsj2.com)). This class will read in a properties file containing a number of stock symbols and their corresponding prices. Save the file when finished. You may have compiler errors because you are missing a required class, StockNotFoundException. The code for this exception is found in Listing 2. Once you have entered the code for both of these classes, you are ready to test it.

In order to test the new application, you need to create a file called stock\_quote.properties that contains the stock symbols and prices you wish to display. An example of this file is in Listing 3. This file must be located in the directory specified by QUOTE\_FILENAME in StockQuoteService.java found on line 10.

Now, from a command prompt, you can test the code within the <WSDE DIRECTORY>/workbench/Sample/SampleProject/servlets by executing "java StockQuoteService <stock\_symbol>," where <stock\_symbol> is a symbol in your stock\_quote.properties file. You should see the corresponding price for that stock.

## NEXT STOP: THE WEB

At this point we are ready to add some spice. Now that we have a working application, it is time to turn this ordinary command-line program into a Web service that is accessible to the world.

The first step is to start the WebSphere admin server. You start the admin server from the Windows Services (on NT, Start/Settings/Control Panel/Services; on 2000, Start/Settings/Control Panel/Administrative Tools/Component Services/Services). Find and start the service labeled, "IBM WS AdminServer."

The next step is to create a new WSDL Web service in WSDE (File/New/WSDL Web Service). This will bring up a wizard that looks like Figure 4. You can accept the defaults here, but make sure that you select "Create a Web service (WSDL) from an existing Java bean." You may also need to select a folder; select /Sample/SampleProjects/servlets. Click Next.

On the next screen, select the StockQuoteService.class file and proceed. The third screen allows you to select which method to publish. Select getQuote(String).



FIGURE 4 Create a Web service wizard



The fourth screen has many fields, as shown in Figure 5. First of all, you will need to select IBM WebSphere Standard Edition v3.5.2 (even if you are using 3.5.3). You can leave the defaults for the rest. This information will be stored in the SOAP deployment descriptor.

The next screen is where you will actually deploy the SOAP Server and SOAP deployment descriptor from the previous screen. Change

FIGURE 5 Web service deployment options

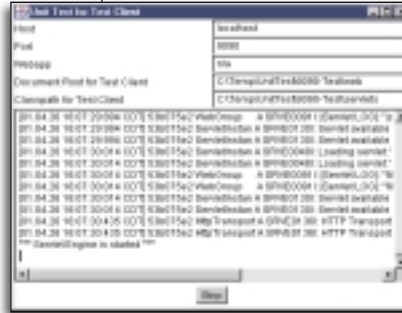


the directory of the WebSphere Install Path if necessary. Then press Deploy. Once you have successfully deployed the app to the SOAP Server, press next.

This is where you will generate a client proxy. A client proxy is a Java class that allows a client application to use your Web service remotely. It contains the methods you declared in the third screen plus `getEndPoint()` and `setEndPoint(java.net.URL)`. These two methods allow the client application to set the location of the SOAP Server. The methods you declared create a SOAP request to the specified SOAP Server with the parameters that the client specified, all behind the scenes! We will discuss this more later.

The next screen generates the test client for the Web service. It is a JavaServer Page (JSP) that runs on the specified port (default 9090) and will invoke the methods on the `StockQuoteServiceProxy` class. When you click the Launch button, it starts a test servlet engine from which the test client is executed. It

FIGURE 6 The unit test servlet engine



also launches your Web browser to the `testclient.jsp` page. The servlet engine is shown in Figure 6; the test client page in Figure 7.

Click on the Web service creation wizard. This screen simply displays what has happened (see Figure 8). Click Finish and return to the Web browser.

In the top left frame is a list of methods from the services `StockQuoteServiceProxy` class. Clicking on a method here changes the top right frame to allow you to enter any

FIGURE 7 The test client page

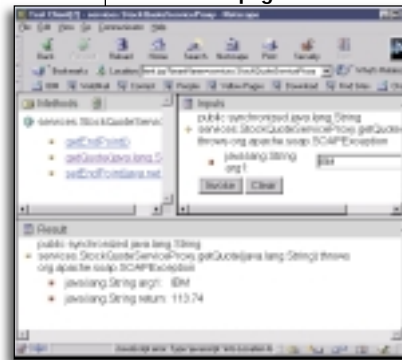
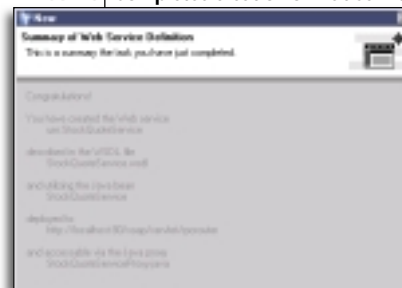


FIGURE 8 Completed creation of Web service



parameters, and invoke the service's method. Experiment with this to work out any bugs in your service's implementation. For example, rename the properties file and try to invoke `getQuote(String)`. You should see a `SOAPException` in the bottom frame. You can also use stock symbols that you have not defined.

If everything has gone correctly, you have just created your first Web service! Of course, the test client application is the only client so far, and it is not terribly useful. If all you wanted

to do was display stock prices in a Web page, you have many less complex alternatives (servlet/JSP, CGI, etc.). So, to take better advantage of your new Web service, let's write a client application with logic that depends on this Web service.

## THE SERVICE REQUESTER

If you are a stock day trader, you know the importance of selling and buying at just the right moment (hopefully buying low and selling high). If you are like me, you cannot afford to hire a stockbroker to constantly monitor the market. But if you could write an application that monitored a stock (or list of stocks), you could build in logic to buy or sell once the stock reaches a set price. This idea is the basis for our client application.

Listing 4 shows the code for the Stock Broker (service requester) client. To run the `StockBroker`, you will need to create a `broker.properties` file such as that in Listing 5. This will allow you to set up the stock you wish to purchase (or sell), the range the stock price must be within in order to buy or sell it, and the quantity of shares to buy or sell. The operation attribute must be either "BUY" or "SELL". Also, when executing the `StockBroker` you will need to ensure that the client proxy and `soap.jar` and the `<WSDE DIRECTORY>/workbench/Sample/Sample Project/servlets` directory is in the classpath (e.g., `java-classpath .;c:\itp\plugins\org.apache.soap\soap.jar` `StockBroker`).

Once you have started the `StockBroker`, it will sit and wait until the stock price falls (or rises) into the specified range. Open the `stock_quote.properties` file in a text editor such as Notepad, and change the specified stock price so that it fits within your range. Within a few seconds, `StockBroker` will alert you that it has purchased (or sold) a number of shares.

The key to writing Web service requester applications is in using the generated proxy class methods (see Listing 6). If you have access to a networked environment, you can write a more advanced client application that will make a SOAP request to another server [using the `setEndPoint(URL)` method in the proxy class], possibly on another platform.

## SUMMARY

If you have made it this far, you have successfully created and tested your own Web service and client. If this were a real enterprise quality client, you could distribute it via CDROMs, FTP, etc., to your paying customers. Furthermore, you could publish your service in a UDDI registry, on the Internet.

While this article has shown you how to write a very simple Web service, I hope that it will also encourage you to explore the various Web Services tools and offerings available.



# Where Are Web Services Going?

*They might become popular, but are they worth it?*

**W**eb Services is a term that is being used to define a set of technologies that exposes business functionality over the Web as a set of automated interfaces. These automated interfaces allow businesses to discover and bind to interfaces at run-time, supposedly minimizing the amount of static preparation that is needed by other integration technologies. The question is, "Do Web Services solve the business problem they propose to solve?"

The business problem in the case of Web Services is the integration of extranet services into larger applications. My belief is that Web Services do not offer any great advancement forward in solving extranet integration, which I hope to illustrate throughout this article. Let's start by further identifying the business problem and the components of Web Services that serve as a solution.

The first part of the business problem is discovering the available services that have been exposed by a particular organization. These services may be transactional, informational, or visual. A transactional service allows the user of the service to invoke a transaction, such as ordering a new computer system. An informational service allows users to query information, such as the status of an already placed order or to request product documentation. A visual service will respond by provid-

ing data that is designed for visual display, such as in a Web browser.

The business problem is defined as two-parts: (1) how can we locate and identify a service for use, and (2) once found, how can we incorporate that service as part of a larger extranet application. It is important to recognize that this is about extranet applications here. Web Services is a powerful, but potentially heavyweight, service that may fill the needs of an extremely large population, but might be excessive for intranet integration purposes.

Allow me to further clarify this point. There are those who confuse the entire Web Services architecture with the portion of Web Services that satisfies the second part of the business problem (described by #2 above). Indeed, there is a need to leverage and use services in a loosely coupled manner for intranet application integration as well as for extranet applications, but there are more lightweight technologies that are available to assist in these endeavors. We will cover this further in the section on "Leveraging Web Services."

For now, we can identify the problem as attempting to incorporate existing extranet services into internal business processes for the purpose of automation and increased communications.

### HOW TO DISCOVER A SERVICE

To discover these types of services, a group of companies, led by Microsoft, Ariba, and IBM, created the Universal Description, Discovery, and Integration (UDDI) specification. This specification forms the basis of a registry that is designed to store service profiles by organization, and contains all the necessary informa-

tion required to understand the intent and invoke a particular service. Note that not all of this information is intended for machine-to-machine communications. A service interface might be represented by a telephone number as well as by a TCP/IP port.

The UDDI specification focuses on the storage, organization, and architecture of the registry service, and not on the usage of the information that is contained within. UDDI leverages the Simple Object Access Protocol (SOAP) to define interfaces for publishing and inquiring about the information in the UDDI repository.

### LEVERAGING WEB SERVICES

Initially, some human intervention will probably be required to locate and identify a Web service. This will probably occur during the design phase of a new or modified business process. This Web service will then be incorporated into an application that extends across the firewall during one or more steps of the business process to retrieve information associated with that process.

For example, an automobile manufacturer may want to automate their supply chain by extending across the Internet and retrieving order and inventory status from a supplier. This will probably take place as part of a larger planning process that also leverages information from internal ERP and other operational systems.

The next set of technologies that assists with this type of initiative will help companies with binding and executing remote services in an automated fashion. Again, this is where some confusion occurs regarding the purpose of the binding technology. In the case of Web Services, there is an accepted specification called Web Service Description Language (WSDL) that is an XML document that defines the inputs and outputs of a Web service, including the XML Schemas that should be used to create the input and output documents.

Each WSDL document contains both an abstract definition of the service and how that service binds to a particular network implementation and data format bindings. The following example is from the WSDL specifi-



#### Author Bio

JP Morgenthal is CTO of iKimbo, a leading provider of enterprise instant communications solutions. He is also author of *Enterprise Application Integration with XML and Java* and a world-renowned expert on distributed computing.



jp@ikimbo.com



cation and defines two messages that will be used to create a virtual port and how that port binds to an HTTP/SOAP implementation.

```
<message name="GetLastTradePriceInput">
  <part name="body"
    element="xsd1:TradePrice" />
</message>

<message
  name="GetLastTradePriceOutput">
  <part name="body"
    element="xsd1:TradePriceResult" />
</message>
```

GetLastTradePriceInput and GetLastTradePriceOutput each describe a message whose body consists of either TradePrice or TradePriceResult document types.

```
<portType name="StockQuotePortType">
  <operation name="GetLastTradePrice">
    <input
      message="tns:GetLastTradePriceInput" />
    <output
      message="tns:GetLastTradePriceOutput" />
  </operation>
</portType>
```

In the above section, portType defines a virtual port that creates an interface called StockQuotePortType with an operation called GetLastTradePrice. GetLastTradePrice is analogous to a method on the StockQuotePortType interface for those familiar with this type of terminology. It also defines that the input to this method is a GetLastTradePriceInput and the output is returned as GetLastTradePriceOutput.

The binding section in Listing 1 physically links the interface and the method described earlier to a specific implementation. In this case, the binding states that the service is implemented on the URL specified by the soapAction attribute of the soap:operation element and that it returns a SOAP message with a body conforming to the stockquote XML Schema.

This section of the WSDL document is critical for users of the service to understand as it tells them how to create the SOAP message for delivery. Other such bindings exist for HTTP GET/POST and MIME.

```
<service name="StockQuoteService">
  <documentation>My first service</docu-
    mentation>
  <port name="StockQuotePort"
    binding="tns:StockQuoteBinding">
    <soap:address location="http://exam-
      ple.com/stockquote" />
  </port>
</service>
```

Finally, we name the service StockQuoteService, which links it to the SOAP binding and the abstract definition of the StockQuotePort. Users of this service will send their SOAP messages to the URL defined by the soap:address element. By definition then we can tell that there are multiple URLs used to implement this service. The external service definition is identified by the service element and the internal service implementation is identified by the binding element.

In the above example, a significant portion of the specification is dedicated to late binding efforts that allow a service definition to be connected to many different implementations. While this could be used for the purposes of intranet application integration, there are many more assumptions that can be made about the environment in which that type of integration is being performed. For example, it may prove more advantageous to use IBM's MQ Series to connect two disparate applications and to use a SOAP message between them to provide the loose coupling desired without inheriting all of the overhead of the multiple layers discussed here, which are necessary for extranet binding to occur.

## WHERE IS THIS LEADING?

It is my belief that XML is so flexible and can be so easily molded to fit any problem space that it has led to people using XML in scenarios where it is actually debilitating to the application instead of an enhancement. I call this illogical need of engineers to use XML without solid reasoning the "Not Invented For XML" (NIFX) syndrome—a derivation of the well-known Not Invented Here (NIH) syndrome. Web Services in total is one example of NIFX.

Web Services, like many other extranet integration technologies, still suffers from many of the same hurdles as its predecessors, namely CORBA, DCOM, Java RMI, or DCE. XML does provide a more flexible data serialization mechanism to use in tandem with an extranet integration, but adds a new dimension of complexity to the overall application. For example, there are now multiple layers of parsing and unmarshaling that must occur before the actual implementation is invoked. Also, it is as strongly tied to the need for standardization of data formats as any of the preexisting extranet integration technologies.

Additionally, Web Services does not provide solutions for any unanswered questions business executives might have with regard to exposing functionality to its trading partners. Indeed, many of the functions that businesses

have chosen to expose they already do or could have with existing Web application technologies, such as HTTP, HTML, and servlets/Active Server Pages. Also, the security model surrounding Web Services needs time to mature as we continue to find holes in Internet security implementations on a monthly basis. And, WSDL is simply a new incarnation of the Interface Definition Language (IDL) that was used by Microsoft's DCOM and CORBA.

Web Services does offer an alternative to invoking publicly exposed functionality. That is, users are no longer tied to "scraping" ever-changing HTML forms in order to integrate over the Web, but still gain the benefit of a document metaphor for integration over a pure application programming interface. However, the downside risk is that users are now susceptible to XML Schema versioning the development of XML document handlers for the return values.

Web Services will become prevalent because anything XML is perceived by business to be simpler to use and to write applications to process — not because it is a major improvement over other extranet integration technologies. It is my belief that the industry would have been better served by focusing on using XML as a serialization format between disparate existing extranet integration technologies, thus leveraging the maturity of those solutions while minimizing the disparities between them. ☎

### Listing 1: Linking the interface to a specific implementation

```
<binding name="StockQuoteSoapBinding"
  type="tns:StockQuotePortType">
  <soap:binding style="document"
    transport="http://schemas.xmlsoap.org/
      soap/http" />
  <operation name="GetLastTradePrice">
    <soap:operation
      soapAction="http://example.com/GetLas-
        tTradePrice" />
    <input>
      <soap:body use="literal"
        namespace="http://example.com/stock-
          quote.xsd"
        encodingStyle="http://schemas.xml-
          soap.org/soap/encoding/" />
    </input>
    <output>
      <soap:body use="literal"
        namespace="http://example.com/stock-
          quote.xsd"
        encodingStyle="http://schemas.xml-
          soap.org/soap/encoding/" />
    </output>
  </operation>
</binding>
```

# **IBM**

**[www.ibm.com/websphere/speed](http://www.ibm.com/websphere/speed)**



# **IBM**

**[www.ibm.com/websphere/speed](http://www.ibm.com/websphere/speed)**

*Major improvements  
to a technology for  
creating and deploying  
Web Services*

Creating Web Services with the

# Microsoft SOAP Toolkit

Version 2 beta




Written By  
Wei Meng Lee

BIO: .....

Wei Meng Lee is a writer, developer, and trainer specializing in XML and Web technologies. He is also a co-author for Wrox's *Beginning WAP, WML and WMLScript* and Manning's *Dynamic WAP Application Development*. Wei Meng spoke at the Wrox's Professional Wireless Developer conference as well as WirelessDevCon 2000.

Wei Meng currently holds a full time job as a lecturer at Ngee Ann Polytechnic, Singapore.

 [wei\\_meng\\_lee@hotmail.com](mailto:wei_meng_lee@hotmail.com)

Microsoft recently released the beta 1 of the SOAP Toolkit for Visual Studio version 2.0. While the new release still has some rough spots, the toolkit has provided developers with an easy way to deploy Web Services, especially for those running on the Windows platform. Version 2 of the SOAP toolkit contains some drastic changes from its earlier version. Developers familiar with version 1 of the toolkit will know that it uses the Remote Object Proxy Engine (ROPE) for consuming Web Services and the Web Service Description Language (WSDL) to define the contract for the Web service.

Specifically, the SOAP Toolkit 2 supports the following:

- A SOAP client for developing a client-side application that consumes Web Services
- A SOAP Server for exposing the functionality of COM objects on the server
- Complete support for the Web Services Description Language (WSDL) 1.0 Specification
- Universal Description, Discovery, and Integration (UDDI)
- SOAP Specification version 1.1.

In this article, I illustrate the use of the MS SOAP Toolkit to solve a real world problem. The example used here involves a book distributor that maintains a database of titles containing such information as pricing and publisher. At the moment, bookstores would make routine calls to the customer support of the distributor to check for the pricing of new titles. This method is time consuming and places a burden on the distributor, that could deploy its limited manpower for more productive tasks. Using Web Services, the distributor can expose the functionality that allows anyone (in particular bookstore owners) to check for the pricing of books through the Internet.

We will build three components:

- A COM object that accesses the database for querying the prices of books
- A Web service that exposes the functionality of the COM object
- A client application that consumes the Web service

For this article, I used Windows 2000 Advanced Server and IIS 5.0 on the server side and IE 5.5 on the client side.

## OBTAINING AND INSTALLING THE MS SOAP TOOLKIT

The MS SOAP Toolkit for Visual Studio is available for download at: <http://msdn.microsoft.com/xml>

The installation of the SOAP Toolkit is straightforward. After installation, the toolkit can be found in the directory `c:\program files\mssoapSDK`. Three sub-folders are in this directory:

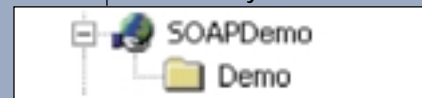
- Binaries
- Documentation
- Samples

The *Binaries* folder contains the relevant DLLs as well as the WSDL generator. You will need the WSDL generator to simplify the process of creating the WSDL file, which is not a very pleasant thing to do by hand. The *documentation* folder contains the help file for the toolkit. The help file gives you an overview of how the toolkit works. The *samples* folder contains some VB and ASP samples. Note that there is no documentation for these samples and you are pretty much on your own. Read the HTML document that comes with them carefully for instructions on trying them out.

## CONFIGURING IIS FOR WEB SERVICES

The first thing to do is to configure the Web server, IIS 5 in this case. For this article, I have created a virtual directory (see Figure 1).

FIGURE 1 Virtual directory



The *Demo* virtual directory is mapped to `c:\inetpub\soaptoolkit\demo`.

## CREATING OUR COM COMPONENTS USING VB

Before we create our Web service, we must first create a COM component using Microsoft Visual Basic 6.0. The component that we are creating in this article is pretty simple. It basically contains a function that performs a search of books in a database. For simplicity, I use Access 2000 so that you can try out the examples yourself without the need for a database server.

Listing 1 shows the code for our COM component (in Visual Basic). Name the project *BookStoreSystem* and the class module *Query* (see Figures 2 and 3).

FIGURE 2 Virtual Directory

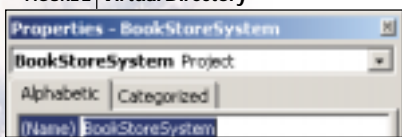
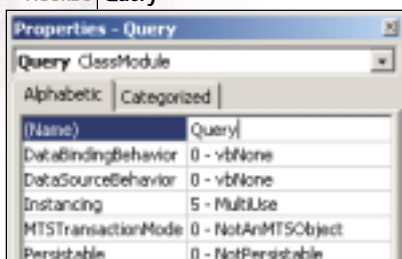


FIGURE 3 Query



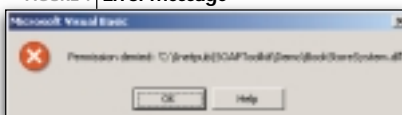
The function `searchBook()` takes in a single argument and outputs a string. Notice that the string returned is actually in XML format containing information about a book's title, ISBN, price, and publisher.

Once the code is created, compile the COM components (DLL) and be sure to note the following:

*Make sure that the DLL is compiled with the "Retained in Memory" and "Unattended execution" options.*

Also, once a COM component is compiled and used, any attempt to modify the code and re-create the DLL will encounter an error message (see Figure 4).

FIGURE 4 Error message



In this case, apart from rebooting your machine in order to release the DLL, you can do the following:

- At the command prompt, type:  
C:\>net stop iisadmin /y  
to stop IIS
  - Next, recompile the COM component. Now the DLL can be overwritten.
  - Restart IIS by typing:  
C:\>net start w3svc
- Our Access table looks like Figure 5.

## USING THE WSDL GENERATOR TO GENERATE THE WSDL AND WSMML FILES

Once the COM component is generated, we are now ready to generate the WSDL and WSMML files needed by the SOAP toolkit. To generate the WSDL and WSMML files, invoke the *wsdlgen* application located in the *Binaries* folder.

FIGURE 5 Access table

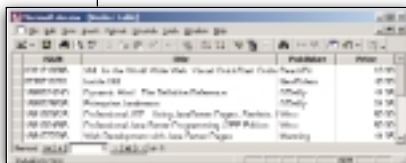
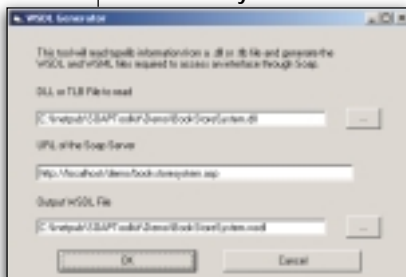


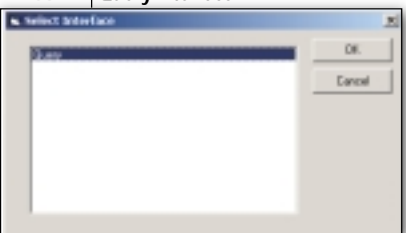
FIGURE 6 Virtual Directory



When you click the *OK* button, the *wsdlgen* application will prompt you to select the interface of your COM object to expose as Web Services. In our example, we have the *Query* interface (see Figure 7).

The *wsdlgen* application will generate two files: *BookStoreSystem.wsdl* and *BookStoreSystem.wsmml*. We will take a more detailed look at the two files a little later.

FIGURE 7 Query interface



## BUILDING THE SOAP LISTENER

Notice that the *wsdlgen* utility requires the URL of the SOAP Server: `http://localhost/demo/BookStoreSystem.asp`

Where is this ASP file? We have to build it manually:

```
<% Response.ContentType="text/xml" %>
<%
    set soapserver =
    Server.CreateObject("MSSOAP.SoapServer")
    wsdl =
    Server.MapPath("BookStoreSystem.wsdl")
    wsmml =
    Server.MapPath("BookStoreSystem.wsmml")

    call soapserver.init(wsdl, wsmml)
    call soapserver.SoapInvoke(request,
    response)
%>
```

Save this ASP file in:

```
c:\inetpub\wwwroot\demo.
```

That's all you need to provide a Web service!

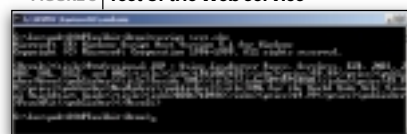
## CREATING THE CONSUMER FOR THE WEB SERVICE

A Web service provider without a consumer is useless. Let's now build our Web service consumer to test our newly built Web service. I use VBScript to build the consumer:

```
set soapclient =
CreateObject("MSSOAP.SoapClient")
Call soapclient.mssoapinit("http://localhost/Demo/BookStoreSystem.wsdl",
"BookStoreSystem", "BookStoreSystemPortType"
)
```

```
On Error Resume Next
wscript.echo soapclient.searchBook
("XML")
if soapclient.faultcode<>" " then
    wscript.echo "soapclient.faultcode - " & soapclient.faultcode
    wscript.echo "soapclient.faultstring - " & soapclient.faultstring
    wscript.echo "soapclient.faultfactor - " & soapclient.faultfactor
    wscript.echo "soapclient.detail - " & soapclient.detail
end if
```

FIGURE 8 Test of the Web service



Save the file as *test.vbs*. To test our Web service, simply type the following at the command prompt:

```
C:\inetpub\wwwroot\demo>cscript
test.vbs
```

If everything goes well, you should see something like Figure 8.

Note that in this example both the client and the server are on the same machine.

## UNDERSTANDING THE PARTS INVOLVED

Now that our Web service is built and consumed, let's look back and see how the various parts work.

## WEB SERVER DESCRIPTION LANGUAGE (WSDL)

The Web Server Description Language (WSDL) is an XML document that describes the services offered by a server. You can think of WSDL as a contract that specifies the services offered and how the client should package its request so that the services can be delivered properly (see Listing 2).

As you can see in Listing 2, the WSDL file contains the following five primary elements:

- <types>
- <messages>

- <portType>
- <binding>
- <service>

For more information on the meaning of all these elements, refer to the SOAP documentation provided with the SOAP toolkit.

When a client wants to make use of a Web service, it first loads the WSDL file and uses the information contained in it to format a SOAP request. The WSDL and WSMML (see below) are both generated by the wsdlgen utility, something that you won't want to do by hand!

You can choose to restrict the number of methods exposed by the COM object by modifying the relevant elements in the WSDL and WSMML files.

## WEB SERVICE META LANGUAGE (WSMML)

The Web Service Meta Language (WSMML) is specific to the current implementation of SOAP by Microsoft. It is used to map services described in the WSDL file to methods available in a COM object.

```
<?xml version='1.0' encoding='UTF-16' ?>
<!-- Generated 02/11/01 by Microsoft SOAP
SDK WSDL File Generator, Version 1.0 -->
<servicemapping name='BookStoreSystem'>
  <using PROGID='BookStoreSystem.Query'
cachable='0' ID='BookStoreSystemObject' />
  <port name='BookStoreSystemPortType'>
    <operation name='searchBook'>

      <execute uses='BookStoreSystemObject'
method='searchBook'

dispID='1610809344'>
        <parameter callIndex='1'

name='searchStr' elementName='searchStr' />
        <parameter callIndex='-1'

name='retval' elementName='Result' />
      </execute>
    </operation>
  </port>
</service>
</servicemapping>
```

The WSMML file contains the <servicemapping> element that contains the <service> child element. The <service> element contains information on the COM component to be used (as indicated by the PROGID) as well as the method(s) to invoke (as indicated by the <operation> element).

## THE SOAP LISTENER

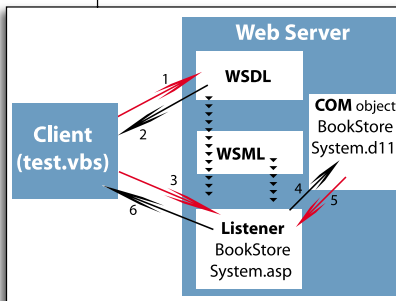
Since the Web server is providing a service, there must be a file that sits on the Web server to "listen" for an incoming request and to

send back the results. This is the function of the BookStoreSystem.asp file we created earlier:

```
<% Response.ContentType="text/xml" %>
<%
  set soapserver =
Server.CreateObject("MSSOAP.SoapServer")
  wsdl =
Server.MapPath("BookStoreSystem.wsdl")
  wsml =
Server.MapPath("BookStoreSystem.wsml")

  call soapserver.init(wsdl, wsml)
  call soapserver.SoapInvoke(request,
response)
%>
```

FIGURE 9 Processing incoming requests



The MS SOAP Toolkit provides the SOAP Server, which processes the incoming requests and responding with the results (see Figure 9).

Let's summarize the various steps that are involved:

1. Client instantiates the SOAPClient object and requests the WSDL file
2. Web server sends back the WSDL file to client
3. Client uses the information in the WSDL file to build SOAP messages and sends the request to the ASP listener
4. The listener instantiates the SOAPServer object and loads the WSDL and WSMML files. The listener then executes the COM object whose methods are mapped to the WSDL file.
5. The COM object returns the result to the listener
6. The listener packages the result as a SOAP message and sends the SOAP message as a response back to the client

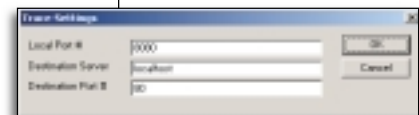
## TRACING SOAP PACKETS

Throughout the entire process, we have not coded a single SOAP message because of the detail that the SOAP Toolkit encapsulates. While this is good when both the client and the server are running on Windows, it is not workable if the client is running on a non-Windows platform. In this case, you need to create the SOAP message on the client-side so that it can be sent to the server, and when the server

responds with a SOAP message, you have to interpret it for the results.

When I was toying with the toolkit, I was able to see the structure of the SOAP messages that were created, sent, and received. To do so, I use the tcpTrace utility, downloadable from <http://www.pocketsoap.com/tcptrace/default.asp>. The tcpTrace utility allows you to monitor the TCP traffic between your client and your server. In my test environment, I am running both my client and my server on the same machine; my trace setting is shown in Figure 10.

FIGURE 10 Trace Setting Values



The setting indicates that all the traffic going into port 8080 will be redirected to port 80 and their contents will be monitored. To trace the traffic generated by our SOAP client, we need to modify the following two files:

1. Test.vbs:

```
Call
soapclient.mssoapinit("http://local-
host:8080/Demo/BookStoreSystem.wsdl",
"BookStoreSystem","BookStoreSystemPortTy
pe")
2. BookStoreSystem.wsdl
```

```
<soap:operation
soapAction='http://localhost:8080/demo/b
ookstoresystem.asp' />
...
<soap:address
location='http://localhost:8080/demo/boo
kstoresystem.asp' />
...
```

Once the two files are modified, I run the script again:

```
C:\inetpub\soap\toolkit\demo>cscript
test.vbs
```

Immediately, two entries are displayed in the tcpTrace window (see Figure 11).

The first entry is when the SOAP client requests the WSDL file. This is shown by the GET request shown in the right-top window. The right-bottom window shows the WSDL file returned. Since the WSDL file is saved as a Unicode file, the tcpTrace utility is not able to display it.

The second entry shows the soap client sending a SOAP request to the server (shown in the right-top window). The SOAP request is:

```
<?xml version="1.0" encoding="UTF-8"
standalone="no"?>
```



```
<SOAP-ENV:Envelope SOAP-
ENV:encodingStyle="http://schemas.xml-
soap.org/soap/encoding/" xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/enve-
lope/">
  <SOAP-ENV:Body>
    <m:searchBook
xmlns:m="http://localhost/demo/BookStoreSys-
tem.xsd">
      <searchStr>XML</searchStr>
    </m:searchBook>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

The server also returns a SOAP message containing the results (shown in the right bottom window). This message is shown in Listing 3.

Since I have indicated in my COM object that my method searchBook() is going to return a string, the SOAP server will replace all XML reserved characters (like "<" and ">") with their respective entities.

FIGURE 11 First entry: Soap client requests WSDL file

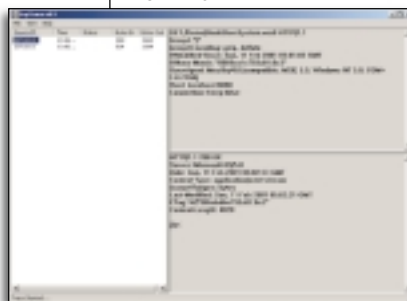
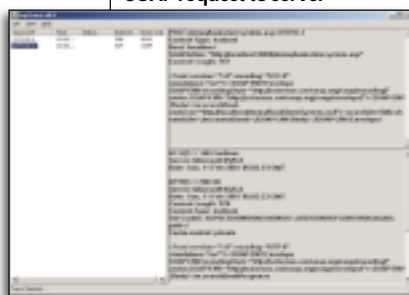


FIGURE 12 Second entry: Soap client sends SOAP request to server



## USING THE SOAP CLIENT ON THE WEB BROWSER

Having tested our Web service, let's now see how we can make use of this service in real life. Let's create a Web page that allows a bookstore owner to enter a search string so that the results can be obtained from the Web service (see Listing 4).

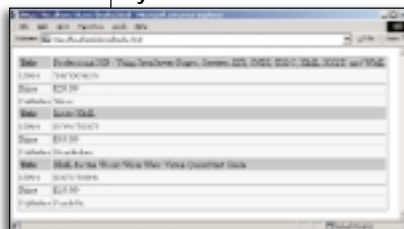
When a user keys in a search string, say "xml," the Web service is contacted and the following result will be shown:

Let's look at the listing in more detail:

```
'===Initialise the SOAP client===
```

```
set soapclient =
CreateObject("MSSOAP.SoapClient")
call
soapclient.mssoapinit("http://localhost/Demo
/BookStoreSystem.wsdl",
"BookStoreSystem", "BookStoreSystemPortType")
```

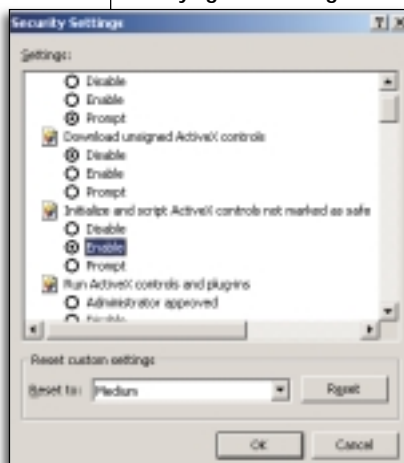
FIGURE 13 Displaying the results returned by the Web service



Since the SOAP toolkit objects aren't marked safe for scripting, you need to modify your IE settings to allow ActiveX objects not marked as safe to be created and used.

As our search result is returned as an XML string, I need to be able to extract the relevant information. The XML string returned is:

FIGURE 14 Modifying the IE Settings



```
<Result>
  <title>Professional JSP : Using
JavaServer Pages, Servlets, EJB, JNDI,
JDBC, XML, XSLT, and WML</title>
  <isbn>1861003625</isbn>
  <price>59.99</price>
  <publisher>Wrox</publisher>
  <title>Inside XML</title>
  <isbn>0735710201</isbn>
  <price>49.99</price>
  <publisher>NewRiders</publisher>
  <title>XML for the World Wide Web:
Visual QuickStart Guide</title>
  <isbn>0201710986</isbn>
  <price>19.99</price>
  <publisher>PeachPit</publisher>
</Result>
```

The easiest way to extract the relevant information is to use the XML DOM:

```
set xmldoc =
CreateObject("MSXML2.DOMDocument")
'---load into a DOM---
xmldoc.loadxml
(soapclient.searchBook(str1))
```

We then print out the books information using:

```
xmldoc.documentElement.childNodes(i).text
```

where documentElement refers to the root element (<Result>) and the childNodes(i) refers to the child elements (<title><isbn><price><publisher>).

## CONCLUSION

This article has touched on creating Web Services using the Microsoft SOAP toolkit version 2 beta 1. Web Services offer a lot of excitement for distributed computing developers, and for the next couple of months we are definitely going to see many more Web Services created and deployed. Until the full release of the MS SOAP Toolkit is launched, we have time to start exploring this cool technology!

## AUTHOR'S NOTE:

At the time this article went to press, the Microsoft SOAP Toolkit version 2.0 Gold Release had just been released. There are minor changes to the beta release but overall things should look similar. Here are some of the changes:

- Inclusion of a trace utility
- Support for WSDL 1.1 standard

For more information and to download a copy of the Microsoft SOAP Toolkit 2.0, go to the following URL: <http://msdn.microsoft.com/xml/default.asp>



# **Cape Clear**

**[www.Capeclear.com](http://www.Capeclear.com)**

**Listing 1**

```

Public Function searchBook(searchStr As String) As String
    Dim conn As New Connection
    Dim rs As New Recordset
    Dim connStr, sql, xmlStr As String
    Dim discount As Integer

    connStr = "Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=C:\InetPub\SOAPToolkit\Demo\Distributor.mdb;Persist
Security Info=False"
    conn.Open connStr
    sql = "SELECT * FROM Books WHERE Title LIKE '%" &
searchStr & "%'"

    Set rs = conn.Execute(sql)
    xmlStr = "<Result>"
    While Not rs.EOF
        xmlStr = xmlStr & "<title>" & Trim(rs("Title")) &
"
"
        xmlStr = xmlStr & "<isbn>" & Trim(rs("ISBN")) &
"
"
        xmlStr = xmlStr & "<price>" & Trim(rs("Price")) &
"
"
        xmlStr = xmlStr & "<publisher>" &
Trim(rs("Publisher")) & "</publisher>"
        rs.MoveNext
    Wend
    xmlStr = xmlStr & "</Result>"
    searchBook = xmlStr

    rs.Close
    conn.Close
    Set rs = Nothing
    Set conn = Nothing
End Function

```

**Listing 2**

```

<?xml version='1.0' encoding='UTF-16' ?>
<!-- Generated 02/11/01 by Microsoft SOAP SDK WSDL File
Generator, Version 1.0 -->
<definitions name='BookStoreSystem' targetNamespace =
'http://localhost/demo/BookStoreSystem.wsdl'
xmlns:tns='http://localhost/demo/BookStoreSystem.wsdl'
xmlns:xsd='http://localhost/demo/BookStoreSystem.xsd'
xmlns:soap='http://schemas.xmlsoap.org/wsdl/soap/'
xmlns='http://schemas.xmlsoap.org/wsdl/'>
<types>
<schema
targetNamespace='http://localhost/demo/BookStoreSystem.xsd'
xmlns='http://www.w3.org/1999/XMLSchema'>
</schema>
</types>
<message name='searchBook'>
<part name='searchStr' type='string' />
</message>
<message name='searchBookResponse'>
<part name='searchStr' type='string' />
<part name='Result' type='string' />
</message>
<portType name='BookStoreSystemPortType'>
<operation name='searchBook'
parameterOrder='searchBookInOut1'>
<input message='tns:searchBook' />
<output message='tns:searchBookResponse' />
</operation>
</portType>
<binding name='BookStoreSystemBinding'
type='tns:BookStoreSystemPortType'>
<soap:binding style='document'
transport='http://schemas.xmlsoap.org/soap/http' />
<operation name='searchBook'>
<soap:operation soapAction='http://localhost/demo/book-
storesystem.asp' />
<input>
<soap:body use='encoded'
namespace='http://localhost/demo/BookStoreSystem.xsd'
encodingStyle='http://schemas.xmlsoap.org/soap/encoding/' />
</input>
<output>
<soap:body use='encoded'
namespace='http://localhost/demo/BookStoreSystem.xsd'
encodingStyle='http://schemas.xmlsoap.org/soap/encoding/' />
</output>
</operation>
</binding>
<service name='BookStoreSystem'>
<port name='BookStoreSystemPortType'
binding='tns:BookStoreSystemBinding'>
<soap:address location='http://localhost/demo/book-
storesystem.asp' />
</port>
</service>
</definitions>

```

**Listing 3**

```

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.xml-
soap.org/soap/encoding/" xmlns:SOAP-ENV="http://schemas.xml-
soap.org/soap/envelope/">
<SOAP-ENV:Body>
<m:searchBookResponse
xmlns:m="http://localhost/demo/BookStoreSystem.xsd">
<searchStr>XML</searchStr>
<Result><Result><title>Professional JSP :
Using JavaServer Pages, Servlets, EJB, JNDI, JDBC, XML, XSLT,
and
WML</title><isbn>1861003625</isbn><price>
59.99</price><publisher>Wrox</publisher><tit
le>Inside
XML</title><isbn>0735710201</isbn><price>
49.99</price><publisher>NewRiders</publisher><t
title>XML for the World Wide Web: Visual QuickStart
Guide</title><isbn>0201710986</isbn><price>
19.99</price><publisher>PeachPit</publisher><
lt;/Result>
</Result>
</m:searchBookResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

**Listing 4**

```

<html>
<script language="vbscript">

Sub SearchTitles
'=====

'===DOM for the results returned===
dim xmldoc
dim soapclient

on Error resume next
str1 = trim(document.form1.searchStr.value)

'===Initialise the SOAP client===
set soapclient = CreateObject("MSSOAP.SoapClient")
call
soapclient.mssoapinit("http://localhost/Demo/BookStoreSystem.wsdl
", "BookStoreSystem", "BookStoreSystemPortType")

set xmldoc = CreateObject("MSXML2.DOMDocument")
'---load into a DOM---
xmldoc.loadxml (soapclient.searchBook(str1))

if soapclient.faultcode<>" then
document.write "An error has occurred : " &
soapclient.detail
exit sub
end if

document.write "<table border=1>"
if xmldoc.documentElement.childNodes.length=0 then
document.write "No books found."

for i = 0 to xmldoc.documentElement.childNodes.length - 1
step 4
document.write "<tr bgcolor='silver'><td><b>Title
</b></td><td>" & xmldoc.documentElement.childNodes(i).text &
"</td></tr>"
document.write "<tr
bgcolor='whitesmoke'><td>ISBN</td><td>" &
xmldoc.documentElement.childNodes(i+1).text & "</td></tr>"
document.write "<tr
bgcolor='whitesmoke'><td>Price</td><td>$" &
xmldoc.documentElement.childNodes(i+2).text & "</td></tr>"
document.write "<tr
bgcolor='whitesmoke'><td>Publisher</td><td>" &
xmldoc.documentElement.childNodes(i+3).text & "</td></tr>"
next
document.write "</table>"
'=====

End Sub

</script>

<form method="post" name="form1">
<b>Search by title</b><br/>
<input type="text" name="searchStr">
<button type="submit" onclick="SearchTitles">Search</button>
</form>

</html>

```



**AltoWeb**  
**[www.altoweb.com](http://www.altoweb.com)**

# Electronic Business XML (ebXML): Making Web Services Work for Business

*Setting the foundations today for the promises of tomorrow*

**C**urrent excitement surrounding the Web Services computing model is predicated on the understanding that this model will deliver a mechanism solving key problems faced in the computing and business worlds today.

These include:

- **Open interoperability:** The requirement that heterogeneous systems communicate without predefined tight infrastructure coupling.
- **Dynamic computing:** The requirement that once a technology is deployed new opportunities that arise can be seamlessly engaged without reengineering.

As Web Services can be produced and consumed by any programming language on any platform, this enables loose coupling across heterogeneous systems. Dynamic computing is achieved through support for dynamic publication and discovery of Web Services being offered over the Internet.

The Web Services architecture comprises a number of key building blocks that combine to enable the Web Services computing model. Fundamentally, there are three distinct building block technologies:

- **Service description:** For a Web service to be useful, at a minimum it must be published

in a manner that allows consumers of the service to understand its function and the type of inputs and outputs the service expects, and determine the transport protocols that can be used to "talk" to the service and discern where on the Internet the service can be engaged.

- **Service Publication and Discovery:** For a Web service to be useful it must first publish its availability in a manner that permits discovery by interested parties.
- **Service usage:** Once a service description has been obtained via the discovery mechanism it must be possible to use that service through standard communication mechanisms.

Based on these core building blocks Web services are being made available today. The types of service being offered vary greatly: from simple Weather Report services to Microsoft's Passport user authentication service. However, these types of Web services exhibit simplistic interaction semantics, if you will, an "ask and you shall receive" style of behavior. Using a Web Services infrastructure to enable electronic business places more stringent demands on Web service interaction.

In the remainder of this article I will explain how Electronic Business XML, or ebXML, takes the Web Services computing model and defines a standard for applying these technologies to meet real world business demands. To provide a clear frame of reference for you, I will consider more familiar Web

Services technologies in parallel with our ebXML discussion while considering each of the core building blocks of the Web Services architecture in turn.

## SERVICE DESCRIPTION

This component of the Web Services architecture requires a mechanism that allows a meaningful description of a service to be defined. Web Services Description Language (WSDL) is a common XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. A service name combined with knowledge of the service provider may well give a WSDL-defined service some business context, but no formal business semantics associated with these types of services can be specified.

The ebXML architecture defines the Business Process Specification Schema (BP Schema). This XML Schema supports the specification of business documents and transactions and the required choreography of these transactions that comprise a complete business collaboration.

As service interactions are modeled as collaborations, either binary (two-party) or multi-party, it follows that a participating party must agree to play one or more roles in a collaboration. When publishing a service a party selects the roles they are willing to engage in.

In essence, the BP Schema lends sufficient functionality for businesspeople to model interactions with a service applying real business context. This context is formalized through the BP Schema description and all interested parties can explicitly reference this business context at service publication.



Author Bio

David Russell is CTO and co-founder of Bind Systems ([www.bindsys.com](http://www.bindsys.com)), a software vendor developing Web Services technologies to enable collaborative electronic business. Bind Systems develops the BindPartner Business Collaboration Platform. Russell lives and works in Dublin, Ireland.



[dr@bindsys.com](mailto:dr@bindsys.com)

## SERVICE PUBLICATION AND DISCOVERY

As stated earlier, for a Web service to be useful it must first publish its availability in a manner that permits discovery by interested parties. The notion of a registry is used to aid in this goal. A company submits a service description to

# JDJEDGE 2001 INTERNATIONAL JAVA DEVELOPER CONFERENCE & EXPO

LEADING EDGE JAVA TECHNOLOGIES FOR  
THE NETWORK ECONOMY



THE LARGEST JAVA CONFERENCE AND EXPO EVER ON THE EAST COAST!

TOP 100 JAVA VENDORS EXHIBITING BY INVITATION ONLY!



Co-located with Web Services Edge  
September 24-25, 2001

**Expo**

September 24-25, 2001

Conference September 23-26, 2001

Hilton New York, New York City

CERTIFICATION FAST TRACKS

## Track 1

Macromedia  
ColdFusion



## Track 2

Java  
University



## Track 3

IBM  
WebSphere



## Track 4

BEA  
WebLogic



## Track 5

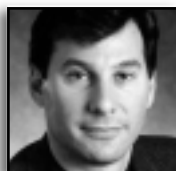
Oracle  
9i



## KEYNOTE PANEL: WEB SERVICES PARADIGM



**GOSLING**  
Creator of Java  
VP & Fellow, Sun Microsystems



**BARATZ**  
CEO, Zaplet, Inc.  
Former President, JavaSoft



**ROSS**  
Founder, JavaLobby  
Panel Moderator



**LYNCH**  
President  
Macromedia



**SCOTT**  
CEO, PointBase, Inc.  
Cofounder, Oracle

## THE DEPTH AND BREADTH OF EDUCATION TO ADVANCE YOUR PROFESSIONAL SKILLS

JDJEdge Keynotes and  
Unmatched Faculty to be  
Announced at JavaOne!

Over 150 Intensive Sessions,  
Tutorials, Night School and  
Introducing Certification Fast Tracks

The Largest Java Expo  
Floor on the East Coast

The World's Leading Java  
Companies Exhibiting by  
Invitation Only...



ONLINE REGISTRATION OPENS TODAY

LIMITED SPACE AVAILABLE!

MANY TRACKS WILL SELL OUT BEFORE CONFERENCE DATE

[WWW.SYS-CON.COM/JDJEDGE](http://WWW.SYS-CON.COM/JDJEDGE)

Owned and Produced by SYS-CON Events, Inc.

Media Sponsors





an Internet-accessible registry, which acts as both a store for service descriptions and a means for service discovery through browsing and intelligent search utilities. Universal Description, Discovery and Integration (UDDI) is an example of a registry mechanism that can be used for Web service publication and discovery. UDDI can therefore be used, for example, to publish WSDL-defined services, making them available for discovery and usage.

To publish a "business-ready" service the service description must consider more business-critical parameters before publication. These can include the quality and level of service that can impact the execution of the service, the communication between service provider and consumer, and the ability, if any, to handle failure conditions or recovery procedures, in addition to the types and granularity of security expected among others.

The ebXML architecture defines a Collaboration Protocol Profile (CPP). A CPP is an XML document that allows a party to express both the business collaborations in which they can engage and the quality and levels of service they can support in delivering that service. A CPP therefore defines the comprehensive set of capabilities of a single party, from both a technological and a business context. A formal CPP description can be published and then universally understood by interested parties.

Publishing a CPP is achieved by posting the document to an ebXML global registry. An ebXML registry provides a set of services that enable the sharing of information between trading partners. An ebXML registry supports more than just service description publication and discovery. Catalogs of reusable business collaboration definitions and reusable business documents are also made available through the registry. A CPP can simply reference a business collaboration or document defined within a registry if so desired.

I should point out here that UDDI and ebXML registry services can complement one another. In real world deployment environments it is probable that UDDI will be used to discover services published in UDDI registries that link to ebXML registries for ebXML-enabled services.

## SERVICE USAGE

A simple Web service can be engaged as soon as a description for that service has been found. In a business environment where two or more parties plan to engage in electronic

business, the terms and conditions of engagement need to be formally agreed upon and then adhered to.

As noted in the preceding section, a CPP defines the capabilities of a single party. Agreement must therefore be negotiated between parties. The ebXML architecture defines a Collaboration Protocol Agreement (CPA). A CPA is an XML document that represents the intersection of two CPPs and is mutually agreed upon by both parties who wish to conduct electronic business.

Negotiation patterns exist that can be executed over the ebXML transport that enable CPA negotiation between business parties. A CPA is negotiated after the service discovery phase. A CPA then governs the interaction behaviors of a party by constraining the runtime environment to a set of parameters agreed to by all parties who will execute such an agreement.

Once a CPA has been agreed upon, electronic business interaction may commence. While WSDL-described services permit transport bindings to protocols such as SOAP and HTTP, it is the ebXML messaging service that provides a standard way to exchange business messages among ebXML trading partners.

The ebXML messaging service provides a secure, consistent, and reliable mechanism to

exchange messages between users of the ebXML infrastructure. Maximizing the benefits of SOAP 1.1 messaging supporting MIME envelopes that permit direct attachments, the ebXML transport extends and embraces the value of what has come before. It is its ability to transport payload of any type, with sequencing of payloads in complex communications, supporting comprehensive security mechanisms and non-repudiation all while enforcing the "rules of engagement" as defined by the active CPA that make the usage of ebXML transport a further key component to "business-ready" Web Services.

## CONCLUSION

I promised to explain how ebXML takes the Web Services computing model and defines a standard for applying these technologies to meet real world business demands. This article should have identified for you the business-critical demands and the technologies required to meet those demands at each level in the Web Services architecture. While there is real value in the simple Web Services of today, if Web Services are to deliver on current expectations in the business domain, I firmly believe it will be comprehensive initiatives such as Electronic Business XML that will make that happen. ©



# XMLEdge2001 INTERNATIONAL CONFERENCE & EXPO MEETING BUSINESS CHALLENGES WITH XML



The ONLY XML Event  
Backed by the Power of



## Plan to Attend...This 4-Day Cutting-Edge Conference

Register  
Online Today!  
**SAVE  
\$600**

*XML...Dramatically Changing the Substance of Corporate Systems*

- XML will fundamentally improve the speed, cost and flexibility of business applications.
- IT Managers expect XML Technology to enable performance of new operations and shorten application development time by 30% to 50%
- E-commerce transactions using XML are predicted to rise from .5% in 2000 to more than 40% by the end of 2003.
- XML products now under development will hit the market in the second half of 2001. *See them first at XMLEdge!*



**www.XMLEdge2001.com**

### Conference Tracks

#### Track 1: PRESENTING XML

The latest tips, tricks and strategies for presenting XML in various front-end presentation environments

XSL / XSLT / CSS / XLINKS / XPOINTERS  
XHTML  
BROWSER SUPPORT FOR XML APPLICATIONS  
WML AND WAP  
VOICEXML  
SCRIPTING AND XML (PERL / PHP / PYTHON / JAVASCRIPT)  
WEB SERVICES  
WIRELESS XML: WAP / WML, XML AND MOBILE COMMUNICATIONS

#### Track 2: REAL-WORLD XML

Actual case studies using XML and related technologies in various industry sectors

XML TOOLS / UTILITIES APPLIED TO ACTUAL APPLICATIONS  
DOCUMENT AND CONTENT MANAGEMENT (EDITORS / PARSERS / SERVERS / TRANSLATORS)  
REAL-WORLD APPLICATIONS  
XML ISSUES IN THE REAL WORLD (PERFORMANCE / SECURITY / COMPRESSION / SERIALIZATION)  
USE OF XML IN INDUSTRY SECTORS (MANUFACTURING / HIGH TECH, ETC.)

#### Track 3: XML & JAVA

The latest developments in XML and Java environments as well as insights into developing solutions using XML, Java and related technologies

DOM/SAX / JDOM  
JAVA AND XML PROGRAMMING (JSP, JMS, ETC.)  
PARSING XML IN JAVA  
XML AND DATABASE PROGRAMMING (XQL)

#### Track 4: EAI with XML

Comprehensive coverage on developing Enterprise Application Integration solutions with XML

XML AND APPLICATION SERVERS  
B2B SERVERS  
PROCESS AND WORKFLOW INTEGRATION  
XML-BASED INTEGRATION USING MESSAGING SYSTEMS  
ERP INTEGRATION  
DATABASE INTEGRATION

#### Track 5: XML STANDARDS

The latest news and updates on emerging XML standards from the industry gurus

W3C STANDARDS  
XML DESIGN PATTERNS  
DOM/SAX / JDOM  
ROSETTANET  
SOAP / BIZTALK  
DTDS AND XML SCHEMAS  
XML REPOSITORIES (BIZTALK / OASIS)

THE 2001XMLEdge INTERNATIONAL CONFERENCE & EXPO IS IN NO WAY RELATED TO XMLEdge BY MOBILEQ, A PLATFORM FOR A COMPLETE WIRELESS INTERNET INFRASTRUCTURE FOR DEVELOPING, INTEGRATING, MANAGING AND EVOLVING APPLICATIONS FOR THE BROADENING SPECTRUM OF WIRELESS DEVICES.



# Web Services: ?

## The Power to Change the World



Steve  
Benfield

is Chief Technology Officer  
for SilverStream Software.  
SilverStream eXtend is the  
first complete environment  
for delivering service-  
oriented applications.  
SilverStream eXtend enables  
businesses to create,  
assemble, consume and  
deploy Web Services through  
J2EE or pervasive legacy  
integration.



Steve can be contacted at  
sbenfield@silverstream.com

So, people keep asking me, "Steve, how come you are such a wild and crazy guy?"

OK, sorry, a late 70s flashback there.

No, they ask me, "Steve, what's your take on Web Services?" To that, I have a standard opening:

1. There is nothing special about Web Services.
2. Web Services will change the world.

The reason I use this opening set of remarks is because before I did, I'd get one of the following reactions when getting into a conversation about Web Services:

1. Web Services? No thanks, I already have enough contractors.
2. Web Services is revolutionary. Are you ready to unleash a force of unstoppable economic power from your tiny little laptop?
3. Oh, Web Services, it's yet another RPC mechanism. It's middleware. Big Deal.

OK...so responses 2 and 3 seem to be at odds with each other. And at some level they are. But on another, they aren't.

### THERE IS NOTHING SPECIAL ABOUT WEB SERVICES

Technically speaking, Web Services and the SOAP, WSDL, and UDDI specifications aren't really earth shattering. Haven't we visited this all before? Wasn't it called CORBA and IDL? Doesn't CORBA have location independence? So now it's just using XML...big woo. Oh yeah, CORBA had security and transactions as well and today Web Services doesn't.

And on a purely technical level, this is correct. Web Services, as a technology, is just an RPC mechanism that uses XML over HTTP using a UDDI registry for service lookup. (OK you purists, I

know it doesn't have to be HTTP and it doesn't have to be UDDI.) I don't know if you remember CORBA in the early 90s? Well, CORBA was supposed to open up this gateway for everything to be able to talk to everything. And you know what? On a technical level, it would work. Except for the fact that at the time we were pre-Internet, pre-HTTP, pre-firewall (ok...kind of), pre-XML, and pre-eBusiness. Oh, it also didn't help that only seven people on the entire planet knew how to fully implement this stuff.

This leads us to the next statement.

### WEB SERVICES WILL CHANGE THE WORLD

Today we're wired; we've got protocols that we all agree upon; we're used to communicating to servers inside other organizations; everyone on the planet has a Web server or an app server; we're using XML for more and more; and we're in love with eBusiness. So the combination of these technologies and the forces driving eBusiness leads us to this second conclusion.

We weren't ready for the promise of CORBA because it required too much to be in place. So Web Services really delivers on the promise of CORBA.

Because of the successes in both competitive advantage and profits that eBusiness has brought, the pressures are on to do more and more. And the focus isn't on the first-mover advantage that had corporations running scared because of the dot-coms; now it's on what we refer to as the right-mover advantage. That is, doing the right things in IT for the business—not just the things that get the fastest payback.

Gartner Group says 80 percent of core mission-critical systems inside organizations are isolated from the net. They also say that in the next few years

these legacy systems will be extended to the Web and that the primary mechanism to do so will be Web Services.

So the reason that Web Services will change the world is that it opens up new ways to get core business processes out of their proprietary shells and available for calling by a variety of users, the first being developers inside the same organization. If Web Services did nothing more than provide a standard way to extend the life and increase the availability of existing functionality it would do wonders. What Web Services says to an organization is that regardless of which platform a department chooses, they can be assured that the functions are available and shareable with other departments. (Those of you on IT Standards Committees should disregard that little statement about different departments using different platforms; we know that would never happen.)

Today when two companies integrate their systems for some purpose, it tends to be a one-off solution—a custom integration. So if you want to integrate with multiple business partners you'll have to write a lot of code. Web Services helps get by that problem and calls on developers and business people to think about generalizing those services and turning them into repeatable (and resalable?) business processes. Web Services enables IT to reduce costs through componentized business processes, increased interoperability, and reuse of existing legacy systems. It also enables business to seek out new ways of working with business partners to reduce cost or increase revenue. It may take a while for IT to embrace all of this, but saving and making more money is something businesses always embrace. ©



# WebServices JOURNAL

Special  
Introductory  
Offer  
**SAVE 20%**

## The World's Leading Independent Web Services Developer Source

Helping you deliver the power of the network to business  
Cutting-Edge Technologies ■ Products ■ News

**Only \$69 for 1 year (12 issues)**  
Regular price \$89 for 1 year

**SUBSCRIBE  
NOW AND  
SAVE  
\$20**



**www.wsj2.com**

**SYS-CON  
MEDIA**

SYS-CON Media, the world's leading publisher of technology magazines for developers, software architects, and e-commerce professionals, brings you the most comprehensive coverage of Web Services.

# WEB SERVICES EDGE 2001 CO-LOCATED WITH JDJEDGE 2001 CONFERENCE REGISTRATION OPTIONS

Call 201 802-3069 if you have any questions  
regarding registration

**Best  
Value!**  
Register  
before  
June 30, 2001

**Save  
\$600**  
with Gold  
Passport  
Registration

## Includes:

2 FastTrack  
Certification Classes  
**plus**

Web Services Edge 2001  
and JDJEdge 2001  
Conferences

	Early Bird before 6/30/01	Pre-Edge before 7/31/01	On-Site and after 7/31/01
<b>GP. Gold Passport</b> .....	<b>\$1,395</b>	<b>\$1,695</b>	<b>\$1,995</b>
<i>Includes all Web Services Edge Conference Sessions (9/24-25) and JDJEdge Conference Sessions (9/23-26) plus 2 FastTrack Certification Classes</i>			
<b>3DP. Three Day Conference Plus</b> .....	<b>\$1,295</b>	<b>\$1,595</b>	<b>\$1,895</b>
<i>Includes all Web Services Edge Conference Sessions (9/24-25), and JDJEdge Conference Sessions (9/23-26) plus 1 FastTrack Certification Class</i>			
<b>3D. Three Day Conference</b> .....	<b>\$995</b>	<b>\$1,295</b>	<b>\$1,595</b>
<i>Includes Web Services Edge Conference Sessions (9/24-25) and JDJEdge Conference Sessions (9/23-26)</i>			
<b>2FT. Two Certification FastTracks</b> .....	<b>\$1,095</b>	<b>\$1,395</b>	<b>\$1,695</b>
<i>(select any two)</i>			
<input type="checkbox"/> (CF) ColdFusion FastTrack			
<input type="checkbox"/> (JC) Sun Java FastTrack			
<input type="checkbox"/> (WS) IBM WebSphere FastTrack			
<input type="checkbox"/> (WL) BEA WebLogic FastTrack			
<input type="checkbox"/> (9i) Oracle 9i FastTrack			
<b>1FT. One Certification FastTrack</b> .....	<b>\$695</b>	<b>\$895</b>	<b>\$1,195</b>
<i>(select one)</i>			
<input type="checkbox"/> (CF) ColdFusion FastTrack			
<input type="checkbox"/> (JC) Sun Java FastTrack			
<input type="checkbox"/> (WS) IBM WebSphere FastTrack			
<input type="checkbox"/> (WL) BEA WebLogic FastTrack			
<input type="checkbox"/> (9i) Oracle 9i FastTrack			
<b>2D. Any Two Days</b> .....	<b>\$745</b>	<b>\$895</b>	<b>\$1,050</b>
<i>Includes Web Services Edge (9/24-25) and JDJEdge (9/23-26) Sessions (select any two days)</i>			
<input type="checkbox"/> Monday, September 24			
<input type="checkbox"/> Tuesday, September 25			
<input type="checkbox"/> Wednesday, September 26 (JDJEdge Sessions only)			
<b>1D. Any One Day</b> .....	<b>\$475</b>	<b>\$595</b>	<b>\$725</b>
<i>Includes Web Services Edge (9/24-25) and JDJEdge (9/23-26) Sessions (select any one day)</i>			
<input type="checkbox"/> Monday, September 24			
<input type="checkbox"/> Tuesday, September 25			
<input type="checkbox"/> Wednesday, September 26 (JDJEdge Sessions only)			
<b>All above registration fees include admission to the Expo on September 24 and 25.</b>			
<b>EO. Expo Only</b> .....	<b>Free</b>	<b>Free</b>	<b>\$50.00</b>
<i>Includes Keynotes and Super Sessions</i>			



American Airlines has been designated the Official Airline for Web Services Edge 2001/JDJEdge 2001. American Airlines is providing special rates at substantially reduced savings for all Web Services Edge 2001/JDJEdge 2001 attendees. For details on how you can take advantage of this special offer, please call the Airlines office at 1-800-433-1790. Refer to A1091AM.

## Professional Ground Airport Transportation

1877jetlimo 1-877-538-5466 NY Limo Cars 1-888-888-6569

## Hotel



Hilton New York  
1335 Avenue of the Americas  
New York, NY 10019



Special reduced rates are available for Web Services Edge 2001/JDJEdge 2001 delegates. All reservations must be received on or before August 31, 2001. Reservations received after this date will be accepted on a space available basis only, at the special rate, if available. Reservations may be made by calling 1-800-HILTONS. Please request the group rate for the Web Services Edge 2001/JDJEdge 2001.

[www.sys-con.com/WebServicesEdge/](http://www.sys-con.com/WebServicesEdge/)

# REGISTRATION FORM

4

## A. YOUR JOB TITLE (check one only):

- Information Technology Management  
01 ☐ Internet VP/Director/General Manager  
02 ☐ Internet Technical Manager  
03 ☐ Internet WebMaster  
04 ☐ Internet Web Applications Dev. Manager  
05 ☐ Internet Content/Design Dev. Manager  
06 ☐ IS VP/Director/General Manager  
07 ☐ IS Senior Manager  
08 ☐ IS Networking/Communications Manager  
09 ☐ IS Software Development Manager

## CORPORATE MANAGEMENT

- 10 ☐ President, CEO, COO, CFO  
11 ☐ EVP, VP, Director  
12 ☐ Marketing Manager

## OTHER

- 15 ☐ Consultant (corporate or independent)  
16 ☐ Engineer/Scientist  
17 ☐ Financial Analyst

## B. BUSINESS/INDUSTRY (check one only):

- 19 ☐ Internet and Web Industries  
20 ☐ Computer & Networking Industries  
21 ☐ Manufacturing  
22 ☐ Finance  
23 ☐ Insurance/Legal/Real Estate  
24 ☐ Government  
25 ☐ Education  
26 ☐ Health Care  
27 ☐ Wholesale/Retail  
28 ☐ Transportation  
29 ☐ Utilities  
30 ☐ Architecture/Construction  
31 ☐ Hospitality/Travel  
32 ☐ Agriculture  
33 ☐ Nonprofit/Religious  
34 ☐ Consultant  
35 ☐ Other [Please specify]

## C. TOTAL NUMBER OF EMPLOYEES AT YOUR LOCATION

### AND ENTIRE ORGANIZATION (check all that apply):

- |                | Location                    | Company                     |
|----------------|-----------------------------|-----------------------------|
| 10,000 or more | 48 <input type="checkbox"/> | 49 <input type="checkbox"/> |
| 5,000 - 9,999  | 50 <input type="checkbox"/> | 51 <input type="checkbox"/> |
| 1,000 - 4,999  | 52 <input type="checkbox"/> | 53 <input type="checkbox"/> |
| 500 - 999      | 54 <input type="checkbox"/> | 55 <input type="checkbox"/> |
| 100 - 499      | 56 <input type="checkbox"/> | 57 <input type="checkbox"/> |
| 100 or less    | 58 <input type="checkbox"/> | 59 <input type="checkbox"/> |

## D. WHICH OF THE FOLLOWING DOES YOUR COMPANY USE OR PLAN TO PURCHASE? (check all that apply)

- |  |  |
|--|--|
| 60 <input type="checkbox"/> BEA WebLogic       | 69 <input type="checkbox"/> XML                    |
| 61 <input type="checkbox"/> IBM WebSphere      | 70 <input type="checkbox"/> WebGain VisualCafe     |
| 62 <input type="checkbox"/> Oracle 9i          | 71 <input type="checkbox"/> JBuilder               |
| 63 <input type="checkbox"/> SilverStream       | 72 <input type="checkbox"/> VisualAge              |
| 64 <input type="checkbox"/> Sybase App Server  | 73 <input type="checkbox"/> TogetherJ              |
| 65 <input type="checkbox"/> Forte for Java     | 74 <input type="checkbox"/> Rational Rose          |
| 66 <input type="checkbox"/> Apache JServ       | 75 <input type="checkbox"/> SonicMQ                |
| 67 <input type="checkbox"/> JRun for Java      | 76 <input type="checkbox"/> Other [Please specify] |
| 68 <input type="checkbox"/> Borland App Server | <input type="checkbox"/> None                      |

## E. WHICH OPERATING SYSTEMS DOES YOUR COMPANY USE OR PLAN TO USE? (check all that apply)

- 77 ☐ OS/2  
78 ☐ UNIX – Sun OS/Solaris  
79 ☐ UNIX – HP/UX  
80 ☐ UNIX – AIX  
81 ☐ MS DOS  
82 ☐ Windows 3.X  
83 ☐ Windows 95  
84 ☐ Windows 98  
85 ☐ Windows NT  
86 ☐ Netware  
87 ☐ Macintosh OS  
88 ☐ Mainframe System  
89 ☐ Minicomputer System  
90 ☐ Other [please specify]

☐ None

## F. DO YOU EVALUATE, SPECIFY, RECOMMEND, AUTHORIZE OR APPROVE THE PURCHASE OF COMPUTER PRODUCTS (SW/HW)?

- 91 ☐ Yes  
92 ☐ No

## G. ARE YOU INVOLVED IN INTERNET/INTRANET DEVELOPMENT?

- 93 ☐ Yes  
94 ☐ No  
95 ☐ Yes, within the next 12 months  
96 ☐ Will not be within the next 12 months

## H. IS YOUR COMPANY CURRENTLY INVOLVED IN JAVA-BASED APPLICATION DEVELOPMENT?

- 97 ☐ Yes  
98 ☐ No  
99 ☐ Yes, within the next 12 months  
00 ☐ Will not be within the next 12 months



**CONFERENCE & EXPO: September 24–25, 2001**

co-located with JDJEdge Conference & Expo September 23–26, 2001

*Hilton New York • New York, New York*

## THREE WAYS TO REGISTER

- 1) **ON THE WEB:** Credit Cards only - [www.sys-con.com/webservicesedge/](http://www.sys-con.com/webservicesedge/)  
2) **BY FAX:** Credit Cards only - Fax completed form to: 201-782-9651  
3) **BY MAIL:** 135 Chestnut Ridge Road, Montvale, New Jersey 07645 Attn.: Web Services Edge Registration

**DEADLINE: July 31, 2001:** After this date, bring this form to the show to register for the conference on-site.

Please complete sections 1, 2, 3 and 4

## 1 YOUR INFORMATION (Please Print)

☐ Mr. ☐ Ms.

First Name \_\_\_\_\_ Last Name \_\_\_\_\_

Title \_\_\_\_\_

Company \_\_\_\_\_

Street \_\_\_\_\_

Mail Stop \_\_\_\_\_

City \_\_\_\_\_

State \_\_\_\_\_ Zip \_\_\_\_\_ Country \_\_\_\_\_

Phone \_\_\_\_\_

Fax \_\_\_\_\_

E-Mail \_\_\_\_\_

## 2 PAYMENT METHOD: (Payment in full due with registration)

☐ Check or Money Order Enclosed (Make check payable to SYS-CON Events, Inc.)

Check # \_\_\_\_\_ For \$ \_\_\_\_\_

Charge my ☐ Visa ☐ MasterCard ☐ American Express ☐ Discover

Name on card \_\_\_\_\_

Card # \_\_\_\_\_ Exp. Date \_\_\_\_\_

Signature \_\_\_\_\_

Billing Address (if different from mailing address)

\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_

## 3 PLEASE INDICATE YOUR CONFERENCE CHOICE:

Total Registration fee \$ \_\_\_\_\_

☐ GP Gold Passport

☐ 3DP Three Day Plus

☐ 3D Three Day

☐ 2FT Two Fast Tracks (Select Two)

☐ CU ☐ JC ☐ WS ☐ WL ☐ 9i

☐ 1FT One Fast Track (Select One)

☐ CU ☐ JC ☐ WS ☐ WL ☐ 9i

☐ 2D Any Two Days (select Two)

☐ Monday, 9/24

☐ Tuesday, 9/25

☐ Wednesday, 9/26 (JDJEdge Sessions only)

☐ 1D Any One Day (select Two)

☐ Monday, 9/24

☐ Tuesday, 9/25

☐ Wednesday, 9/26 (JDJEdge Sessions only)

☐ EO Expo Only



If you require special assistance covered under the Americans with Disabilities Act, please call 201-802-3069 by September 7, 2001.

## CANCELLATIONS, SUBSTITUTIONS, REFUNDS

Fax written refund requests to SYS-CON Registration 201-782-9651. Requests for refunds received prior to July 31, 2001 will be honored, less a 10% handling charge; requests received after July 31, 2001, and before August 23, 2001, will be honored less a 20% handling charge. No requests for refunds will be honored after August 23, 2001. Requests for substitutions must be made in writing prior to August 23, 2001. Fax to 201 782-9651

No one under 16 is permitted to attend the Expo.





# **Shinka Technologies**

**[www.shinkatech.com](http://www.shinkatech.com)**