ADVISOR EXPERT ON
LOTUS NOTES & DOMINO R5

December, 2001
Volume 3, Issue 7

www.Advisor.com

Lotus Domino Rnext/5.x

WEB SERVICES

Domino's Standards Support Continues: UDDI and WSDL

By John Duggan

In my article "Web Services: Using Apache SOAP for Domino" (ADVISOR EXPERT ON R5, November 2001), I outlined how to write and implement a Web service using Apache SOAP and Domino. That article didn't cover how to publicize and find other available Web services on the Internet. As you can see in figure 1, Web services operate within a collection of open standards. This article discusses two of those open standards: Universal Description, Discovery, and Implementation (UDDI) and Web Services Description Language (WSDL).

UDDI

UDDI is an online registry of businesses, services, and service providers. The purpose of UDDI is to provide a global Web services registry, but it isn't restricted to Web services. Any company can register, regardless of whether they supply Web services. The registry lets a business:

- Describe itself with its own name, address, telephone number, and other contact details. (This section of UDDI is known as White Pages.)
- Discover other businesses that offer services based on industry code, geographical information, services, or products. This makes it possible for a business to quickly find other relevant businesses online (known as Yellow Pages).
- Integrate with other businesses using rules, service descriptions, data binding, or other technical details. This provides a quick process of conducting business online quickly and easily (known as Green Pages).

You can access the UDDI registry via a Web browser or through an application using SOAP. You can use one of the following addresses to register or search the registry: http://www.ibm.com/services/uddi/ or http://uddi.microsoft.com/search.aspx.

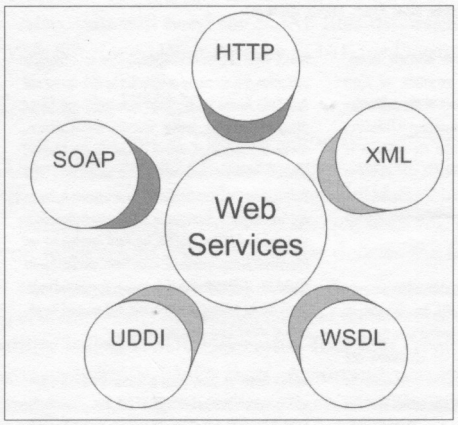

***Figure 1:* Web services**—A collection of open standards.

Subscriber Download

A sample database and the code used in this article.
http://Advisor.com/Article/DUGGJ04

UDDI is intended to be a single registry that is replicated among many nodes, so it shouldn't matter which UDDI registry you use. You can refer to the UDDI Replication Specification at http://uddi.org/specification.html for further information.

The UDDI specification is currently at version 2, with version 3 to be released soon (http://uddi.org). The specification is a joint initiative of more than 200 companies, listed at www.uddi.org/community, and will eventually be released to the World Wide Web Consortium (W3C) as an open standard.

From a developer's point of view, UDDI consists of four objects:

businessEntity—Business contact information.

businessService—List of business services offered by the business entity.

bindingTemplate—The technical details necessary to invoke a Web service.

tModel—Specification information for a particular Web service. Business communities will most likely define common protocols for interoperability among their Web services. These protocols are called fingerprints and you implement them within the UDDI registry as tModels.

Figure 2 shows the relationships between these objects. If you do any development with the UDDI registry, you should have a good understanding of these objects.

WSDL

WSDL version 1.1 is an XML language that describes the interface of a Web service. It contains the semantics and administration-related information of a Web service.

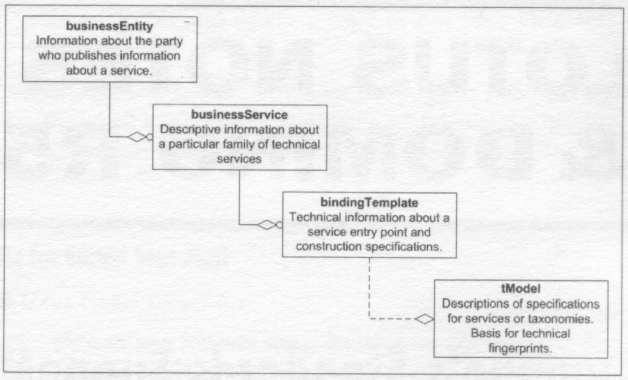

***Figure 2:* How UDDI objects work together**—The relationships between the UDDI objects.

Currently, the UDDI specification doesn't require WSDL to describe a Web service interface because it can be described using text or a formal description language. The advantage of using WSDL is that it can be programmatically processed.

In a nutshell, according to the WSDL specification, the XML file contains the following information:

Types—A container for data type definitions.

Message—An abstract, typed definition of the data being communicated.

Operation—An abstract description of an action supported by the service.

Port Type—An abstract set of operations supported by one or more endpoints.

Binding—A concrete protocol and data format specification for a particular port type.

ADVISOR EXPERT ON LOTUS NOTES & DOMINO R5

Editorial Director John L. Hawkins
Executive Managing Editor Jeanne Banfield Hawkins
Senior Managing Editor Christa Coleman
Managing Editor Elizabeth Olsen
Technical Editors Steve Caudill, Terrance A. Crow
Production Services Manager Jose S. Martinez
Production Manager Tina L. Bennett
Graphic Designer Tina L. Bennett
Circulation Manager Kristen Murphy

ADVISOR MEDIA, Inc.
Founded 1983
Chairman & Chief Executive Officer John L. Hawkins
President Jeanne Banfield Hawkins
Vice President Geri Beaty
Group Publisher Michael Callan

Contact ADVISOR Via the Web: The fastest way to reach editors, customer service, conferences and seminars, and other departments, is by going to www.Advisor.com/ContactAdvisor.

858-278-5600
Fax: 858-278-0300
5675 Ruffin Road, Suite 200
San Diego, CA 92123

Advertising Sales To speak to a sales representative about advertising opportunities, call 800-336-6060 or 858-278-5600, e-mail AdSales@Advisor.com.

Reprints and Bulk Sales: To purchase reprints or bulk magazines, contact the customer service department.

Back Issues: If a back issue is still in stock it can be purchased for US$20, and US$25 for Canada and other countries, includes postage. Contact the customer service department.

Permissions: To quote from an article, contact our Permissions Editor. Include the publication date, the title of the article, the portion you want to reprint, and the purpose of your request.

Photocopy rights: Permission to photocopy articles for use in educational institutions is granted by the Copyright Clearance Center. Contact Academic Permissions at: 222 Rosewood Dr.
Danvers, MA 01923
978-750-8400, fax 978-750-4470

ADVISOR.com
Get comprehensive technical and business resources at ADVISOR.COM. You'll find news, articles, tips, jobs, user groups, product directories, events, forums, training, and more.

ADVISOR MEDIA
Printed in the USA

ADVISOR EXPERT ON LOTUS NOTES & DOMINO R5 (ISSN 1524-6396) is published by ADVISOR MEDIA, Inc., 5675 Ruffin Road, Suite 200, San Diego, CA 92123, 858-278-5600. POSTMASTER: Send address changes to ADVISOR EXPERT ON LOTUS NOTES & DOMINO R5, 5675 Ruffin Road, Suite 200, San Diego, CA 92123. Subscription: 12 issues US$199, Canada US$219 (includes GST R127384618), other countries US$239.

The generator creates two files:

1. DominoSharePrice.wsdl, containing the service details.
2. DominoSharePrice-interface.wsdl, containing the message, port type, and binding details.

The first file imports the second using an imports XML tag. You create two files to let a single interface have many implementations. For instance, you may have a single Web service that provides hotel room and travel bookings. The single Web service can be listed in the registry as two separate entries as a room booking service and a travel booking service. Both services would refer to the single Web service.

You should take a look at the generated files to get an idea of the information stored in them. In particular, look at the contents in the <binding> tags and how that contains information to connect to the Web service.

Registering with the UDDI

The next step is to register the Web service with a UDDI registry. I choose to use the IBM test registry (http://www-3.ibm.com/services/uddi/). You can host your own private registry by downloading a UDDI registry from http://www.alphaworks.ibm.com/tech/UDDIreg, but be aware that this version requires WebSphere and DB2.

Before providing any service details, you must register and create a new business. After that is completed, create a new service type called DominoSharePrice with the overview URL pointing to the file DominoSharePrice.wsdl (e.g., http://localhost/webservices.nsf/DominoSharePrice.wsdl?openpage).

Figure 5 shows the completed service type details.

Assuming you've created everything successfully, you should be able to retrieve the details by searching for a service type called DominoSharePrice and see the results as in figure 6.

It is possible to programmatically update the UDDI registry. For further details, see "UDDI4j: Matchmaking with Web Services" (http://www-106.ibm.com/developerworks/library/ws-uddi4j.html). This is mostly used to automatically update details when the Web service changes, or provide an alternative UI to the registry.

Writing the agent to search the UDDI

After you've registered the Web service, how can you access the UDDI programmatically to find and call your Web service? The sample database WEBSERVICES.NSF contains a Java agent "Call Web Service" that calls the UDDI registry and retrieves all the information you require to call the Web service. This section outlines the code contained in that agent.

The first step is to import all the relevant packages, which include WSDL, UDDI, and SOAP:

```
import com.ibm.uddi.*;
......
import com.ibm.wsdl. *;
......
import org.apache.soap.* ;
.......
```

The next step is to create a connection to a UDDI registry node of your choice:

```
UDDIProxy proxy = new UDDIProxy();
```

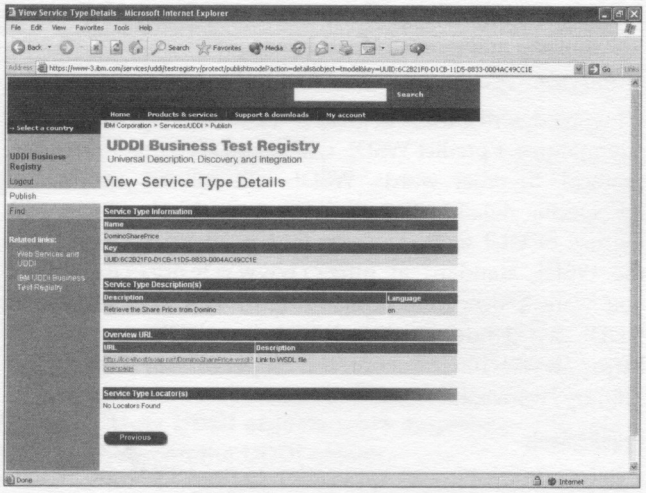

***Figure 5:* Completed service type details**—Setting up the service type DominoSharePrice within the UDDI registry.

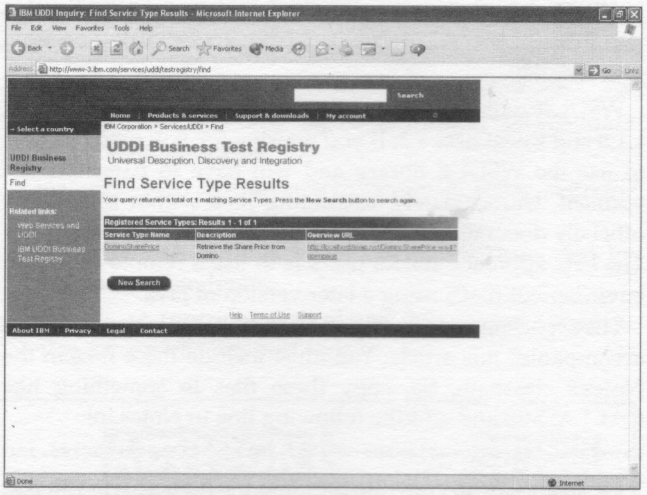

***Figure 6:* Get the details**—Search for DominoSharePrice within the IBM test registry.

```
proxy.setInquiryURL("http://www-
3.ibm.com/services/uddi/testregistry/inquiryapi");
```

At this stage, perform a search on the registry. You can perform four different searches: searching for a business, service, binding, or a tModel. For this example, search for a tModel called Domino Share Price:

```
tModelList tmodellist = proxy.find_tModel("Domino Share
Price", null, 10);
```

The call returns many tModels, if there is more than one. For this example, you only want the first tModel. You can find in-depth information about the objects used in the agent in the UDDI4j and WSDL4j documentation.

```
Vector tModelInfoVector =
tml.getTModelInfos().getTModelInfoVector();
TModelInfo tmi = (TModelInfo)
tModelInfoVector.elementAt(0);
TModelDetail tmd =
proxy.get_tModelDetail(tmi.getTModelKey());
Vector tModelDetails = tmd.getTModelVector();
TModel tm = (TModel) tModelDetails.elementAt(0);
```

Port—A single endpoint defined as a combination of a binding and a network address.

Service—A collection of related endpoints.

I won't describe these in great detail because, as the technology matures, I predict WSDL will disappear as an underlying protocol. In other words, WSDL will still be present, but the specific details will be hidden from the developer with the use of GUI tools. You can find more information about the WSDL standard at http://www.w3c.org/TR/wsdl. If you want a deeper understanding of WSDL in relation to UDDI, see "Understanding WSDL in a UDDI Registry" at http://www-106.ibm.com/developerworks/webservices/library/ws-wsdl/#7.

Requirements

Before developing an application to use UDDI and WSDL, you must have two Java packages:

UDDI4J.jar—Allows publishing, binding, and finding information within a UDDI registry from http://www-124.ibm.com/developerworks/projects/uddi4j/.

WSDL4.jar—A utility to read and parse WSDL files (http://www-124.ibm.com/developerworks/projects/wsdl4j).

WSDL4j relies upon Java 1.2, whereas Notes R5.0.8 uses 1.1.8 and Rnext uses 1.3. If you're using R5, you won't be able to use the WSDL4j library and will have to parse the file yourself. Unfortunately, upgrading Java within Notes isn't officially documented and parsing the file isn't a simple task. The best approach for R5 users is to use a remote call to an environment that's using a later version of Java.

These packages are included with the sample database that accompanies this article. You must include these files in the Notes's classpath. So, copy these files to something like C:\CLASSES and add the following line to Notes.ini:

```
JavaUserClasses=c:\classes\uddi4j.jar;c:\classes\wsdl4j.jar
```

Or, using Rnext, copy the files into the Notes \jvm\lib\ext directory.

Also, you will need the SOAP files discussed in my article "Web Services: Using Apache SOAP for Domino."

Creating the WSDL file

For this example, I use the same Web service defined in the "Using Apache SOAP for Domino" article, which retrieves a share price for a company located within a Notes database.

Before registering this Web service with the UDDI registry, you must create a WSDL file that defines the interface to it.

Fortunately, the IBM Web Services toolkit (http://www.ibm.com/developerworks/webservices) provides a utility called WSDL Generator Tool that creates a WSDL file from a compiled Java class. Assuming you have the toolkit installed, you can launch wsdlgen from the command line.

The first question the utility asks is what type of service you're using. Select Java Class and you'll see a screen similar to figure 3.

Enter the details as shown in figure 3. The value used for the target namespace is the URL location of the to-be-generated WSDL files. When you generate the files, store them in a Notes database as Notes pages. So, for this example, the target namespace is: http://localhost/webservices.nsf/DominoSharePrice.wsdl?OpenPage.

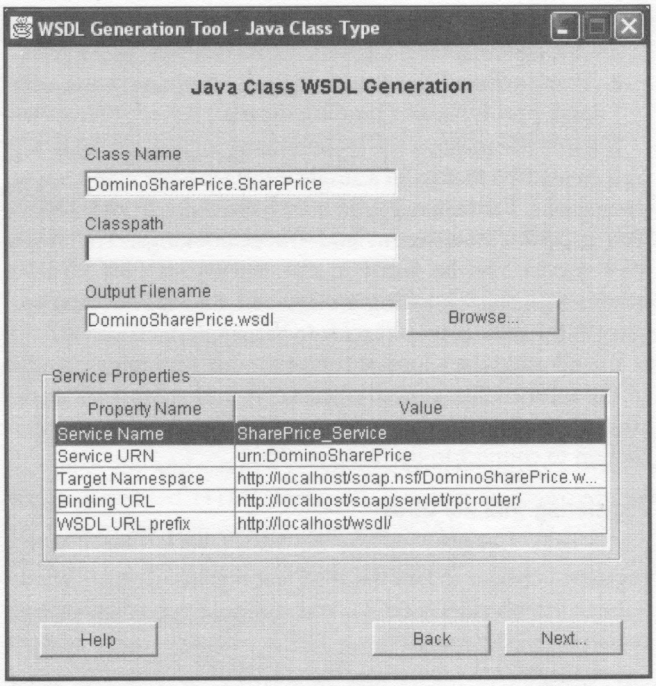

***Figure 3*: The details**—Creating the WSDL file using 'wsdlgen'.

Click on Next and the screen in figure 4 shows all the available public methods and asks which ones you would like to be made available for your Web service. Your Web service has only one method, so select getSharePrice(String).

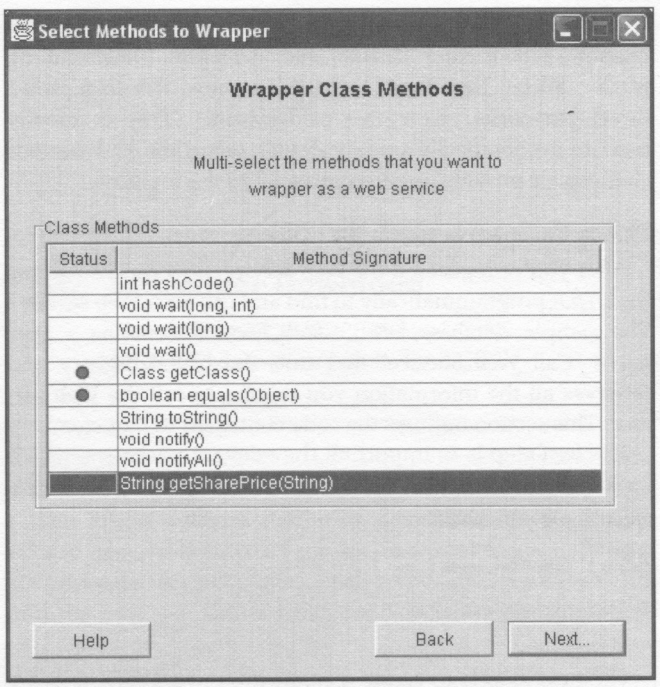

***Figure 4:* List of public methods**—Select your method, getSharePrice(String).

When you've retrieved the tModel details, you can extract the overview URL and parse the resulting WSDL file using the WSDLReader class in the WSDL4j package:

```
OverviewURL ou = tm.getOverviewDoc().getOverviewURL();
Definition definition_details = WSDLReader.readWSDL(null,
ou.getText());
```

You've now parsed the WSDL file and can retrieve the URL to call for your Web service:

```
Service svc = (Service)
definition_details.getServices().values().iterator().next
();
Port prt = (Port)
svc.getPorts().values().iterator().next();
SOAPAddress sa = (SOAPAddress)
prt.getExtensibilityElements().get(0);
String SOAPURL = sa.getLocationURI();
```

As I mentioned earlier, two WSDL files were generated, so you have to retrieve the details for the implementation WSDL and parse that file:

```
Vector impWSDL = (Vector)
definition_details.getImports().values().iterator().next(
);
ImportImpl importdoc = (ImportImpl) impWSDL.get(0);
Definition importdef = WSDLReader.readWSDL(null, import-
doc.getLocationURI());
```

Within this implementation of the WSDL document, you're only interested in how to bind to the Web service. In other words, you only want the method to call the URI of the Web service.

First, get the binding details:

```
Binding bind = (Binding) importdef.getBindings().
  values().iterator().next();
BindingOperation soapop = (BindingOperation)
bind.getBindingOperations().get(0);
```

From this, you can retrieve the method name and the URI:

```
String MethodName = soapop.getName();
String URI = (SOAPBody)
soapop.getBindingInput().getExtensibilityElements().get(0
). getNamespaceURI();
```

You now have all the information you need! You can call the Web service using SOAP calls as outlined in the "Using Apache SOAP" article.

If you have problems with Web services development, try the UDDI Test Bed. This useful tool lets you make a call to the UDDI registry within a Web browser. You can find it at http://www.ecerami.com/uddi/index.php4.

Another useful tool is the Test SOAP Client where you can enter the address of a WSDL file, which returns a dynamically generated HTML form you can use to send requests directly to the Web service. The Test SOAP Client is available at http://www.soapclient.com/soaptest.html.

Putting the pieces together

You should now have a basic understanding of Web services and how SOAP, WSDL, and UDDI fit together. This technology is still evolving, and I predict many new standards will eventually appear to simplify the process of using Web services.

Fortunately, Rnext is compliant with these new technologies, and you can adapt it to work with them. Check http://www.ibm.com/developerworks/webservices and the LOTUS ADVISOR publications to keep up-to-date on this changing technology. ■