Dynamic e-Business with DB2® and Web Services

IBM Data Management

The Internet infrastructure is ready to support a new generation of e-business applications, called Web services. Web services represent the next level of function and efficiency in e-business. Specifically, Web services are enhanced e-business applications that are easier to advertise and easier to discover — by other businesses — because they are described in a more uniform way on the Internet. These new enhancements allow e-business applications to be connected more easily both inside and outside the enterprise.

The Web services infrastructure is based on the eXtensible Markup Language (XML). Messages and data flow between a service requester and a service provider using XML. This paper briefly describes Web services and then describes how DB2 data can be dynamically transformed to XML and the important role that DB2 plays in a Web services world.

Trademarks: The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

DB2
DB2 Universal Database
IBM
MQSeries
WebSphere

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States and/or other countries.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Dynamic e-Business with DB2® and Web Services

Web Services: The Next Generation of Application Integration

*The Web's audience expands to include programs as well as humans.*

It is a bold goal: Create an architecture that makes it possible for software to do what humans do with the Web—access documents and run applications in a general way without requiring application specific knowledge and client software. An architecture that supports Web services provides the groundwork to realize that goal.

Before describing the architecture, let's first understand what is meant in this paper by "Web service". A Web service is a set of application functions that perform some useful service on the behalf of a requester such as informational or transactional functions. A Web service can be described and published to the network for use by other programs across the network. Examples of publicly available Web services today include a stock quote service, a service to retrieve news from Web news sources, a service to return maps of historic weather events by zip code, currency conversion services, and a service to return highway conditions in California. Because Web services are modular, related Web services can be aggregated into a larger Web service. For example, one can envision a wireless application composed of separate Web services that do such things as obtaining stock quotes, subscribing to news services, converting currency, and managing calendars.

One particularly nice aspect of Web services is that the level of abstraction they provide makes it relatively simple to wrap an existing enterprise application and turn it into a Web service. Web services are based on the XML standard data format and data exchange mechanisms, which provide both flexibility and platform independence. With Web services, requesters typically do not know or care about the underlying implementation of Web services, making it easy to integrate heterogeneous business processes. And Web services provide you with a way to make your key business processes accessible to your customers, partners, and suppliers. For example, an airline could provide its airline reservation systems as a Web service to make it easier for its large corporate customers to integrate the service into their travel planning applications. A supplier can make its inventory levels and pricing accessible to its key buyers.

In one possible example, a buyer matches up incoming purchase orders with transportation services.

1) The buyer access the local database to select a list of purchase orders. 2) While viewing the detail for a particular purchase order, the buyer selects a transportation service provider from an approved list, a list that is kept in a private registry. 3) For each provider, the buyer receives quotes dynamically using Web services capabilities. Each request is bound and sent to the location specified in the registry and processed by the supplier's Web service. The supplier's Web service takes input about the request, accesses its database and returns the quote to the requester. 4) The buyer then chooses one service provider based on the prices quoted and the selection is then added back to the purchase order page to reflect the selection of a shipping service provider.

1
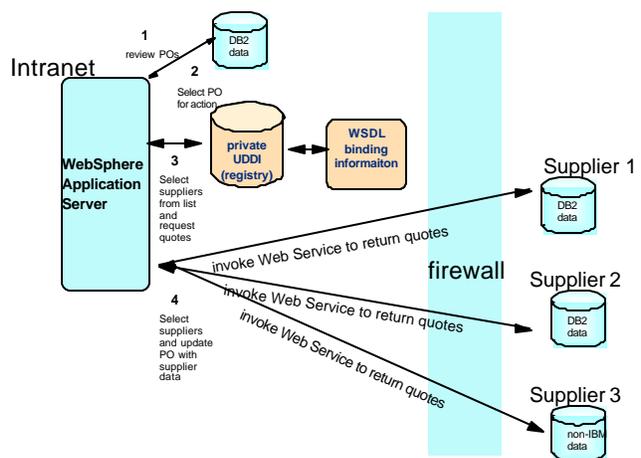
Web services are likely to become ubiquitous where existing technologies have not. Web services leverage XML for data representation and exchange and do not require complex language-dependent mappings and compile time bindings. Web services offer both ease of development and ease of modification. Further, Web services do not mandate tight synchronous relationships between requesters and service providers. This further simplifies the implementation of Web services in an Internet environment where it is impossible to tightly control network behavior. The reliance on XML for data exchange, and the abundance of existing and emerging tools for Web service technology, make it relatively easy to get up and running with your first Web service.

Web Services Fundamentals

The nature of Web services make them natural components in a service-oriented architecture. In a typical service-oriented architecture, service providers host a network accessible software module (for the purposes of this paper, a Web service). A service provider defines a service description for a Web service and publishes it to a service registry. A service requester uses a find operation to retrieve the service description from the registry and uses the service description to bind with the service provider and invoke or interact with the Web service implementation. Let's see how this model is realized using Web services.
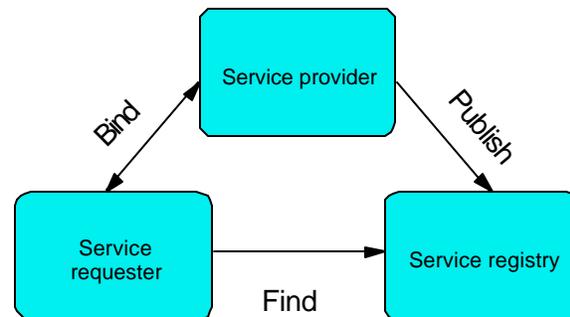
Service-oriented architecture

In simple terms, a Web service is created by wrapping an application in such a way that it can be accessed using standard XML messages which are themselves wrapped in such a way that masks the underlying transport protocol. The service can be publicized by being registered in a standard-format registry. This registry makes it possible for other people or applications to find and use the service.

The pieces of the Web services architecture include:
- A Web service (a general term used to describe software that can be invoked over the Web)

- Application-specific messages that are sent in standard XML document formats conforming to the corresponding service description.

-

A SOAP request consists of the envelope itself, which contains the namespaces used by the rest of the SOAP message, an optional header, and the body, which may be a remote procedure call (RPC) or an XML document.

SOAP builds on existing Internet standards such as HTTP and XML, but can be used with any network protocol, programming language, or data encoding model. For example, it is possible to send SOAP messages over IBM MQSeries®, FTP or even as mail messages.

- The logical interface and the service implementation are described by the Web Services Description Language (WSDL). WSDL is an XML vocabulary used to automate the details involved in communicating between Web services applications. There are three pieces to WSDL: a data type description (XML Schema), an interface description, and binding information. The interface description is typically used at development time and the binding information may be used at either development or execution time to actually invoke a particular service at the specified location. The service description is key to making the Web services architecture loosely coupled and reducing the amount of *required* shared understanding and custom programming between service providers and service requesters.

- To enable service requesters to find your Web service, you can publish descriptive information, such as taxonomy, ownership, business name, business type and so on, via a registry that adheres to the Uniform Description, Discovery and Integration (UDDI) specification or into some other XML registry. The UDDI information can include a pointer to WSDL interfaces, the binding information, as well as the actual business name (the name that makes the purpose of the Web service understandable to humans). A UDDI registry is searchable by programs, enabling a service requester to bind to a UDDI provider to find out more information about a service before actually using it.

- The ability to compose Web services together is provided by Web Services Flow Language (WSFL), another specification for which IBM is taking the lead. WSFL can be used to describe a business process (that is, an execution flow from beginning to end), or a description of overall interactions between varying Web services with no specified sequence.

If we look at how all of these specifications work together, a Web service can be defined as a modular application that can be:
- *Described* using WSDL
- *Published* using UDDI
- *Found* using UDDI
- *Bound* using SOAP (or HTTP GET /POST)
- *Invoked* using SOAP (or HTTP GET/POST)
- *Composed* with other services into new services using WSFL

You can restrict access to Web services much as you would restrict access to Web sites that are not available to everyone. WebSphere® provides many options for controlling access and for authentication. The standards for this are still emerging. Microsoft and IBM have proposed a SOAP security extension to the W3C as the mechanism for XML digital signatures. The SOAP security extension included with WebSphere Application Server 4.0 is intended to be a security architecture based on the SOAP Security specification, and on widely-accepted security technologies such as secure socket layer (SSL). When using HTTP as the transport mechanism, there are different ways to combine HTTP basic authentication, SSL, and SOAP signatures to handle varying needs of security and authentication.

Accessing DB2 Data through Web Services

IBM is enabling its key programming models and application servers with Web services and is providing and developing tools to automatically generate Web services from existing artifacts such as Java Beans, EJBs, and stored procedures. This paper describes another way that you can use to submit SQL statements and, if you choose, control the format of the returned data. This support is known as the Web services Object Runtime Framework (WORF), and it is available in beta from the DB2 XML Extenders web site at http://www.ibm.com/software/data/db2/extenders/xmlext/ . The following types of Web service operations are supported:

- XML-based query or storage. In other words, an XML document is stored in DB2 relational tables and composed again on retrieval. This method of operation requires the presence of DB2 XML Extender.
- SQL-based operations, such as calling stored procedures, or inserting, updating, deleting DB2 data.

XML-based query: XML-based query allows you to compose XML documents from relational data. You can also break an XML document down into its component parts and store it into relational tables. Part of the underlying support for this functionality is provided by DB2 XML Extender. The store and retrieve operations are handled by special stored procedures that are shipped with DB2 XML Extender.

One of the inputs into both storage and retrieval is the user-specified mapping file that creates the association between relational data and XML document structure. This mapping file is called a *document access definition* (DAD) and provides a way that you can create an XML document with the XML elements and attributes named as you please and with the shape that you want. The focus of this approach is in moving and manipulating XML documents.

XML document composed from DB2 data

SQL-based query: SQL-based querying is simply the ability to send SQL statements, including stored procedure calls, to DB2 and to return results with a default tagging.

The focus of this approach is actually the data in and out of the database, not on shaping the results in a particular way.

SQL-based query does not require the use of DB2 XML Extender because there is no user-defined mapping of SQL data to XML elements and attributes. Instead, the data is returned using only a simple mapping of SQL data types, using column names as elements.

Rows and columns returned with default XML tagging

However, if you are using DB2 XML Extender to store XML documents within a single column of a table, you can use SQL-based query to retrieve those documents intact as a character large object (CLOB), or to invoke the user-defined functions that extract parts of the document. Another feature of DB2 XML Extender is the ability to store frequently-accessed data in side tables, thereby enabling speedy searches on XML documents that are stored in columns.

Another useful thing you can do with SQL-based query is to invoke DB2 stored procedures. Stored procedures are natural for conversion to Web services since they are themselves an encapsulation of programming logic and database access. A Web service invocation of a stored procedure makes it possible to dynamically provide input parameters and to retrieve results.

Making it Work

Both the XML-based and SQL-based forms of querying are controlled by a file called a *document access definition extension* (DADX). The DADX defines the operations that can be performed by the Web service. For example, you might have a DADX that specifies the operations to find all orders for parts, find all orders for parts with a particular color, and orders for parts that are above a certain specified price. (The color or price can be specified at runtime as input parameters by using host-variable style notation in the query.)

There can be multiple DADX files for a single database.

DADX

```
<?xml version="1.0"?>
<DADX xmlns="urn:ibm:.com:dadx"....>
.
 <SQL_call>CALL Myproc(:query1, :query2,
:query3)</SQL_call>

 <SQL_query>select * from order_tab
...
 </SQL_query>

 <SQL_update>insert into order_tab
...
 </SQL_update>

<retrieveXML>
     <DAD_ref>getstart_xcollection.dad</DAD_ref>
     <SQL_override>
     select o.order_key, customer_name,
customer_email...

     </SQL_override>
  </retrieveXML>
```

Tools are planned to be available to include support for creating DAD files, DADX files, and to also include wizards for client proxy generation and for configuring database connections. This tool support is planned to be included with the beta of WebSphere Studio Application Developer (planned to be available with VisualAge for Java 4.0 EE).

You can also write DADX files using any text editor. After you create a DADX file, the DB2 Web services download provides support to automatically generate WSDL files, including support for UDDI Best Practices. In addition, support is provided to generate the deployment descriptor needed to deploy the Web service into WebSphere, and to generate a documentation and test page, which you can use for testing and as a basis for building the client part of your Web application.
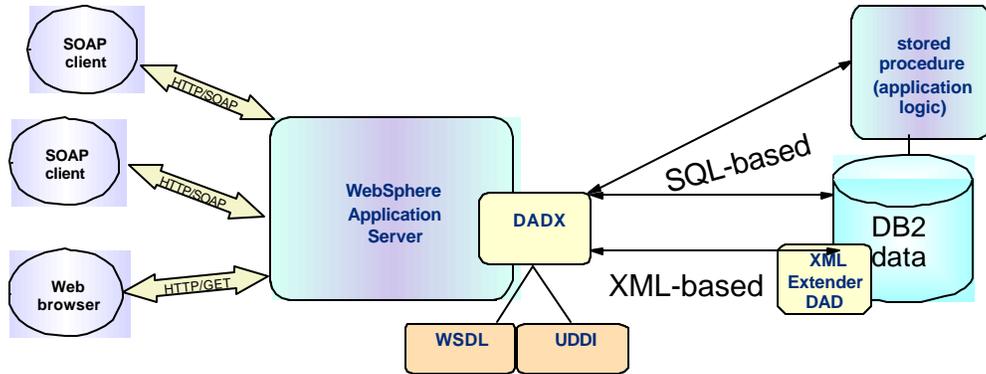
Web Services Architecture

IBM Video Central for e-Business Scenario

A common need among our three fictional video rental chains, MegaVideo, SuperVideo, and IndependentVideo, is the need to ensure that the customers they rent to have a good credit history. Solutions that are bounded by a particular video chain lose any benefit of understanding a customer's history with other chains. And this, unfortunately, has been the way our video chains have been forced to operate.

One of the companies, SuperVideo, has already developed a tightly-coupled intranet application that accesses DB2 data directly. SuperVideo does not have any information about a particular customer's record at other chains. MegaVideo contracted a third-party vendor to develop a centralized customer infraction repository. This allowed other companies to register and query customer credentials. However, the solution used a proprietary interface and it was difficult to integrate with existing applications, meaning that the solution had few registrants. Small chains, like IndependentVideo, cannot afford to create a solution of their own or to do the work to integrate the proprietary registry into their operations.

Thus, IBM Video Central was conceived. IBM Video Central is a hypothetical Web service provider for video rental applications. The purpose of this service provider is to provide a central data repository that can be accessed by registered Web-based applications. The IBM Video Central application provides a suite of Web services to address the business needs of the client in two areas: business services, serving the administrative needs of the client, and customer services, that enable the client both to serve and to manage

their customer base.

IBM Video Central Web Services
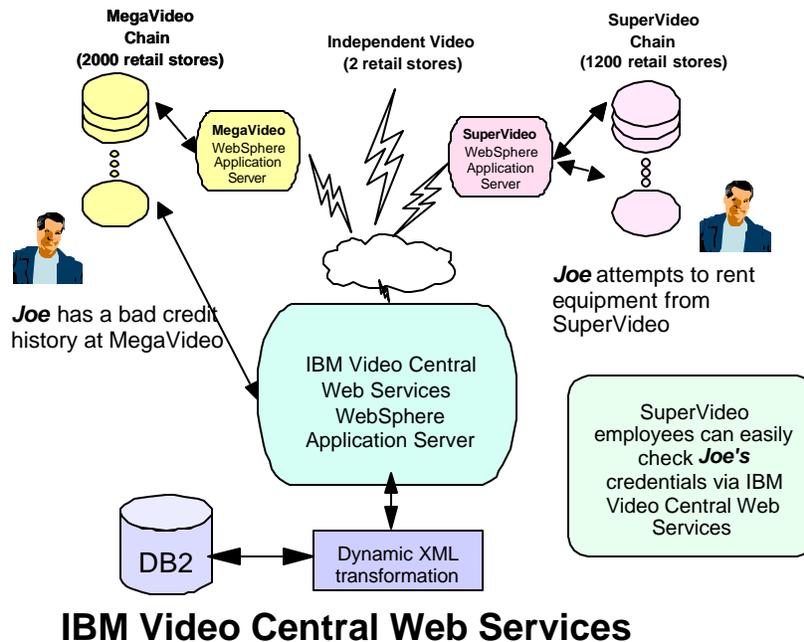
IBM Video Central reduces business risk and operational costs for our three video chains, because its service for validating customer credential is universally accessible. By adding additional consumer-based services, such as rental history, wish-list registry, and a recommended video list, it is possible for our IBM Video Central businesses to increase their revenue per customer.

IBM Video Central is a sample that currently implemented using Java Beans and is available for you to download at http://www.ibm.com/software/data/developer/samples/video/index.html. A future version of IBM Video Central is planned to be implemented using the DADX method described in this paper.

Information Integration using Web Services

IBM recognizes that Web services provides not just an architecture for integrating applications, but also for integrating data. And DB2 provides the capability to both manage data and provide intelligent, optimized access to data. This initial support of Web services support for DB2 is only the first step toward deeper integration of Web services in DB2. In the near future, it may be possible to invoke Web services directly from DB2 itself, through a stored procedure or user-defined function. And the federated database capabilities in DB2 would enable quick and easy integration of diverse back-end data using Web services as a near real-time data source.

Relevant Resources

Information about Web services and data management can be found at:
http://www.ibm.com/software/data/webservices

General information about Web services can be found at http://www.ibm.com/software/webservices and at http://www.ibm.com/developerworks/webservices

Information about XML Protocol can be found at http://www.w3.org/2000/xp/

Information about DB2 XML Extender can be found at
http://www.ibm.com/software/data/db2/extenders/xmlext/index.html

Brown, Allen, Barbara Fox, Hada Satoshi, Brian LaMacchia, and Hiroshi Maruyama. *SOAP Security Extensions: Digital Signature*. W3C Note. IBM Corporation and Microsoft, 2001.
http://www.w3.org/TR/SOAP-dsig/

Christensen, Erik, Francisco Curbera, Greg Meredith, Sanjiva Weerawarana. *Web Services Description Language (WSDL) 1.1*. W3C Note. Ariba, IBM Corporation, and Microsoft, 2001.
http://www.w3.org/TR/wsdl

Ferguson, Donald F. *Web Services Architecture: Direction and Position Paper*. IBM Corporation. Paper for W3C Web Services Workshop, April 11-12, 2001. http://www.w3.org/2001/03/WSWS-popa/paper44