Integration, Web Services Style

Michael Carey, Michael Blevins, Pal Takacsi-Nagy
BEA Systems, Inc.
{mcarey, mblevins, pal}@bea.com

Abstract

In mid-2002, BEA released WebLogic Workshop, an integrated development framework that dramatically simplifies the task of building enterprise class web services and deploying them in the J2EE environment. In this article, we describe where BEA is going next in terms of web services. In particular, we provide an overview of the next generation of BEA's WebLogic Integration product, which is currently undergoing a major web services inspired overhaul. We explain how web services and application integration relate in the next release of WebLogic Integration; we describe how business processes (a.k.a. workflows) can be used to construct applications by orchestrating web services and how the emerging XML Query standard provides the XML data manipulation capabilities required for web service-oriented integration. We also briefly describe how BEA's new Liquid Data for WebLogic product complements WebLogic Integration by adding web service-based data integration capabilities to the BEA Platform.

1 Introduction

The integration of heterogeneous applications and data has been a long-standing problem in both the enterprise computing industry and computer science research. As evidence, it has been twenty years since the Multibase project at the Computer Corporate of America first tried to provide a common data model and query interface to a collection of disparate database systems [5], it has been over ten years since the OMG CORBA 1.0 specification was published in an attempt to provide a standard way to interconnect distributed heterogeneous applications [7], and it has been ten years since the founders of Tibco (then Teknekron Information Systems) tried to bring order to application integration by developing an event-oriented (*a.k.a.* publish/subscribe) "information bus" with application adaptors [6].

With the advent of XML and its accompanying technologies and standards, it appears that there is finally real hope on the integration horizon. In particular, web services offer many of the basic ingredients (in what have become widely accepted forms) required to crack the integration problem in a standards-based way [4] - including a simple and flexible physical data format (XML), a logical data description mechanism (XML Schema), a loosely-coupled remote invocation facility (SOAP), a service interface description mechanism (WSDL), and a declarative query language (XQuery). Work is in progress on standards for some of the "holes" in today's web service technology, such as XML web service orchestration protocols, security, and transactions. As a result, it is our belief that web services will finally succeed where various previous architectures and standards have fallen short.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

Earlier this year, BEA released a major new product called WebLogic Workshop [BEA02a]. BEA WebLogic Workshop provides a unified platform consisting of an integrated development environment and a runtime container [BEA02b] that make developing and deploying J2EE-based enterprise applications an activity that all corporate application developers (not just Java/J2EE experts) can undertake successfully. Release 1.0 specifically focuses on the task of implementing, composing, and deploying industrial-strength web services. To make it easy to use enterprise resources such as databases, JMS queues, existing EJBs, and other web services to build a new web service, WebLogic Workshop provides a controls framework - developers can set properties, call methods, and handle events using controls that provide simplified interfaces to the resources, shielding them from the low-level details and complexities required to use such resources directly. Web service invocations can be transported over HTTP or JMS, can be stateless or stateful, and can be either synchronous or asynchronous. WebLogic Workshop generates the low-level J2EE artifacts (stateless session beans, entity beans, message-driven beans, JDBC calls and result set processing, etc.) needed to support the applications' choice of controls and control annotations.

2 Web Services and Application Integration

In the past, application integration technologies have suffered from drawbacks such as insufficiently loose interface bindings (e.g., CORBA [7]) or use of proprietary object models (e.g., first-generation EAI products such as [6]). XML and web services offer dramatic improvements in these respects. The next generation of BEA WebLogic Integration (WLI 8.0) is based solidly on XML and web services standards, particularly in its new business process management capabilities. On the "backend" side, it utilizes the WebLogic Workshop web service control paradigm to make it simple to construct business processes that integrate applications that are exposed through web service interfaces. Many packaged application vendors, such as SAP, Siebel, and PeopleSoft, are moving towards providing web service APIs; others are building tools that export legacy mainframe applications in web service form, while still others are making important functionality for business partners available through web services (e.g., the Amazon.com Web Service). In addition, BEA WLI 8.0 (like WLI 7.0) includes a set of J2EE-CA adaptors that provide XML "application views" (accessible via WebLogic Workshop's application view control) that expose XML interfaces to a wide variety of non-web-service-enabled systems. Within a business process, WLI 8.0 provides built-in support for XML workflow variables, optionally typed by XML Schema types, as well as for Java workflow variables. XML variables are used for storing and manipulating the XML documents and data that flow through integration-oriented applications; their XML content is accessible via Xquery [8] statements. In addition, WLI 8.0 provides web service invocation as one of the primary means for starting or continuing interactions with business processes - yielding a paradigm where business processes *are* web services.

2.1 Business Processes as Web Services

Figure 1 shows a simple example of a business process for handling purchase orders. This process was created using the graphical business process editor of WLI 8.0, which supports the creation and editing of Java Work Flow (JWF) programs. Beneath this graphical view, a JWF is a Java class with annotations that describe the flow (i.e., the business process orchestration) logic; the annotations reference Java or XQuery methods within the class that implement the detailed business logic. This simple purchase order example illustrates how web services relate to business processes in WLI 8.0 - this workflow exposes a SOAP operation that accepts a purchase order asynchronously, places orders for the line items contained therein, and then responds to the requestor with a purchase order reply message by performing a SOAP callback. The process uses a JWF *foreach* loop construct to iterate over the set of line items in the purchase order. In the underlying JWF file for the business process (not shown), the incoming purchase order is stored in an XML workflow variable and an XQuery expression is used to control the looping by enumerating each line item in turn from within the XML purchase order variable. Inside the loop, a web service call is made to send the line item to a backend order management system, and the response comes back from the order management system in the form of a web service callback. JWF includes constructs to specify such flow actions as message sending and receiving, looping, conditional branching, parallel execution, waiting for one of a possible set of messages, Java method invocation, and transaction and exception handling.

It is important to note that JWF flow annotations are not intended to replace Java as a general-purpose programming language. Rather, the flow annotations augment Java's capabilities by adding support for the construction of *long-running, reliable, asynchronous* programs. For example, the line item callbacks in the purchase order example above can take a large (indefinite) amount of time to occur, perhaps hours or even days depending on the nature of the backend system. One of the major benefits of a flow language is that it enables applications like this to be easily constructed by corporate developers. The JWF runtime container uses transactions and queuing to reliably execute and sequence (and recover, if necessary) the individual Java- and/or XQuery-based workflow steps, it handles call/callback correlation, and it enables the application to be deactivated (utilizing entity beans and persistent storage under the covers) during long periods of inactivity - even in the midst of loops in the flow, etc. Note also that the flow description essentially indicates which, and when, various types of messages are expected by the workflow. The workflow container then delivers incoming messages, such as calls and callbacks, to the workflow in the order expected - which may differ from their actual order of receipt. Issues such as these cannot be solved by the Java programming language alone, and handling them manually by utilizing Java and advanced J2EE features such as entity beans and message-driven beans is a difficult task, especially for a corporate application developer.

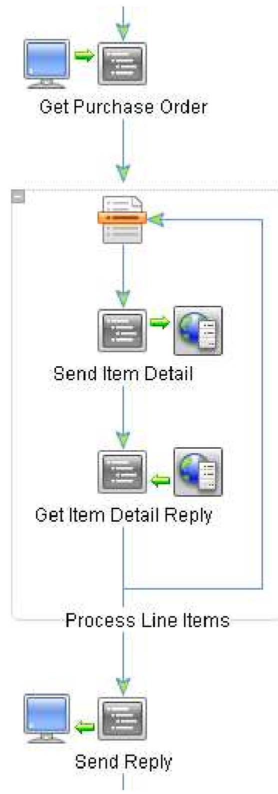

Figure 1: Purchase Order Business Process.

2.2 Slicing and Dicing XML Data

In order to handle the XML data flowing through a business process, WLI 8.0 embeds a high-performance streaming Xquery [8] execution engine. XQuery queries can be found in several roles within business processes - they can be used to transform data on the way into or out of a workflow, and they can also be used to control workflow looping and branching. WLI 8.0 includes graphical XQuery editing tools for these purposes. Figure 2 shows a screen shot from the user interface of the WLI 8.0 XML data transformation tool; a user selects the desired source and target XML Schema descriptions for a given data transformation and then interacts with the tool to describe the transformation in an intuitive graphical manner. Figure 3 shows the XQuery query that is created and (two-way) edited as the result of Figure 2's visual XML mapping specification.

In addition to XML data, the same XML Schema-centric mapping paradigm is used to specify mappings between XML and binary data formats. In the case of binary data, the data transformation developer provides an additional XML file that describes to the system how to parse the binary format of interest to/from XML; this file can be created graphically as well by using the FormatBuilder tool already included as part of WLI 7.0.

Internally, WLI 8.0 includes an XML "document store" layer that provides the required backing store for XML and binary messages and workflow variables. Such data is parsed once, on entry to the system, and then

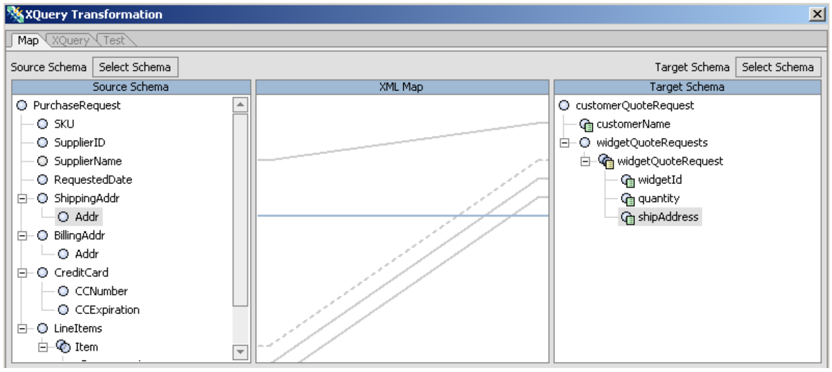

Figure 2: User Interface for XML Data Transformations.

```
<customerQuoteRequest>
    <customerName>{data($src/SupplierName)}</customerName>
    <widgetQuoteRequests>
     {
       for $Item_1 in $src/LineItems/Item return
           <widgetQuoteRequest>
               <widgetId>{data($Item_1/PartNumber)}</widgetId>
               <quantity>{data($Item_1/Quantity)}</quantity>
               <shipAddress>{data($src/ShippingAddr/Addr)}</shipAddress>
           </widgetQuoteRequest>
        }
    </widgetQuoteRequests>
</customerQuoteRequest>
```

Figure 3: Corresponding Generated XQuery.

stored in an efficient tokenized form for further consumption. Also, XML and binary data is passed by reference between workflows running within the same WebLogic Server cluster for added efficiency.

3 Web Services and Data Integration

In addition to taking an XML- and web-services-centric approach to application integration, BEA now has a newly announced mid-tier data integration offering - BEA Liquid Data for WebLogic [3] - which is built on the same standards-based foundation. Liquid Data is designed to complement BEA's WebLogic Integration product by providing a unified, real-time XML view of disparate enterprise data sources. Specifically, Liquid Data views each data source in an enterprise as being an XML data source with a given XML Schema. Sources supported today include relational databases, XML files, web services, WLI adaptor-based application views, and custom data sources (exposed by registering custom Java functions that consume and produce XML). BEA Liquid Data includes an extensive set of administration pages, delivered as an extension to the BEA WebLogic Server application server's console, that enable data sources to be added, configured, and dropped. For creating the unified XML view of the enterprise, Liquid Data includes a graphical Data View Builder tool (roughly similar in function to the WLI 8.0 data transformation tool) that enables a data architect to create XQuery-based views and parameterized stored queries. Note that, like WLI 8.0, the XML, XML Schema, and XQuery web standards lie at the heart of Liquid Data.

Web services play two roles in BEA Liquid Data for WebLogic. On the data source side, web services are an important class of data source that can be included in Liquid Data views and queries. For example, one could form a unified Customer view in Liquid Data by combining customer information from an Oracle database with order information exported by a backend order management system's web service APIs. On the data producer side, as mentioned above, a data architect can use Liquid Data to create a set of parameterized stored queries for use by enterprise application developers. These queries can be exported to the enterprise's internal developer community via web service APIs. (EJB APIs are also supported.) Note that this paradigm provides a very important form of control to the data architect - rather than exposing the enterprise's various operational data sources to ad hoc queries, the architect chooses and publishes a limited set of parameterized queries (e.g., find customer by id, find customer by name, etc.) that have been deemed acceptable to run against the operational data. In this way, Liquid Data uses web services to provide real-time data visibility and aggregation capabilities for "front-office" applications (such as customer care systems, employee self-service portals, and so on).

In terms of their relationship, BEA Liquid Data provides integrated read access to disparate enterprise data, while BEA WebLogic Integration provides the corresponding update capability required to build complete applications. For example, "view employee" is an operation easily provided by Liquid Data, while a corresponding "insert employee" operation requires a potentially complex workflow that may take several days in a large corporation (e.g., to notify the hiring manager, obtain computer and e-mail accounts, arrange office space, notify benefit providers, receive and persist acknowledgements of all these actions, etc.).

4 Conclusion

In this short paper, we have provided an overview of how web services lie at the heart of BEA's approach to application and data integration. In the next generation of the BEA WebLogic Integration product, WLI 8.0, business processes are naturally exposed as web services. In addition, business processes in WLI 8.0 can themselves consume other web services, making it easy to compose new long-running application activities through web service and business process orchestration. Inside, these processes can manipulate XML data, based on XML Schemas, using a combination of high-level JWF actions, XQuery logic, and Java logic. BEA's new Liquid Data for WebLogic product provides integrated real-time access to disparate enterprise data sources. Again, web service APIs play a key role both outside (where the data architect publishes the available query operations as web services) and inside (where web services can themselves be Liquid Data data sources) the Liquid Data system, and inside, the XML Schema and XQuery standards are key. Coupled with WebLogic Server and WebLogic Workshop as part of the overall BEA WebLogic Platform product suite, Liquid Data for WebLogic and WebLogic Integration provide a comprehensive, web-service-based foundation for addressing the integration problems faced by today's enterprises.

References

[1] BEA WebLogic Workshop: The Future of Web Services - Here Today, BEA Systems White Paper, March 2002.

[2] The BEA WebLogic Workshop JWS Container, BEA Systems White Paper, August 2002.

[3] BEA Liquid Data for WebLogic: Increasing Visibility in the Distributed Enterprise, BEA Systems White Paper, October 2002.

[4] Curbera, F., et al, Unraveling the Web Services Web: An Introduction to SOAP, WSDL, and UDDI, IEEE Internet Computing 6(2), March-April 2002.

[5] Landers, T., and Rosenberg, R., An Overview of Multibase, Proc. 2nd Int'l. Symp. on Distributed Data Bases, Berlin, Germany, North-Holland Publishing Co., September 1982.

[6] Oki, B., et al, The Information Bus - An Architecture for Extensible Distributed Systems, Proc. 14th ACM Symp. on Operating Systems Principles, Asheville, NC, December 1993.

[7] Vinoski, S., Distributed Object Computing with CORBA, C++ Report, July-August 1993.

[8] XQuery 1.0: An XML Query Language, http://www.w3.org/TR/xquery/, W3C Working Draft, August 2002.