PERSPECTIVES ON THIRD-PARTY INTEGRATION

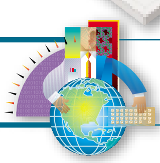
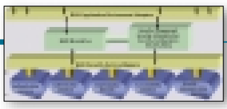
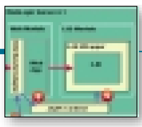
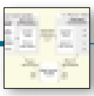
BEA WebLogic DEVELOPER'S JOURNAL

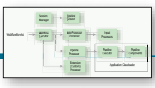
Volume:1 Issue:6

weblogicdevelopersjournal.com

Full Conference Program INSIDE ▼ page 42

Doron Sherman 8



SYS-CON MEDIA

BEA

http://developer.bea.com

BEA

www.developer.bea.com

Wily Technology

www.wilytech.com

Partnering for Success

BY JASON WESTRA
EDITOR-IN-CHIEF

This month ***WLDJ*** focuses on third-party integration. We cover products that integrate at different levels of the BEA e-business platform, and have guest editorials from vendors who have successfully partnered with BEA to provide closely integrated solutions on top of WebLogic Server. Speaking of partnering…I am sorry for using the term "partner," as I think it's overused. It seems like every technology company in existence has at least one press release about a partnership with another firm to "take advantage of synergies," or "to comarket products and services that complement each other."

Dictionary.com defines partnership as:

n. noun 1.) a contract between two or more persons who agree to pool talent and money and share in profits and losses 2.) the members of a business venture created by contract

In other words, a partnership is a relationship of mutual benefit between two entities. What exactly does it mean to successfully partner with BEA? I think you know the benefits you receive – increased exposure, marketability, technical support on BEA products, and discounts on education. However, how does your company, a small software boutique building deployment tools for WebLogic Server, actually provide BEA with some benefit?

In actuality, if you were at BEA's eWorld in February you probably got a reality check. BEA is benefiting greatly from its Star Partner program and its continued interest in it is of great importance to not only its livelihood, but also the livelihood of many software vendors in the J2EE space today. A walk through the exhibit hall where hundreds of vendors showcased their BEA-ready and built-on-BEA products indicated that BEA is as dependent on its partners as its partners are on it.

BEA's Star Partner program focuses on a win-win situation for both parties. The rise in partner numbers is evidence enough of its success. Numerous partners are available for both vertical and horizontal markets to help customers build custom applications more quickly, or allow them to simply install and configure solutions already built on top of WebLogic. Companies interested in becoming partners can apply online, entering company information through a user-friendly Web application. In concept, this puts partner information at the fingertips of needy customers looking for a specific BEA-related skill or product. They can search for a partner on BEA's site at http://bea2.instantsoft.com/sc/advancedsearch.jsp, and, once they have found the right match, contact the Star Partner for business.

The problem I found with this system is that everyone and their brother is a BEA partner for seemingly everything these days! In the Advanced Search section's vertical markets list box, I selected agriculture and found 40 listings. I thought, "Saweeeeeet, I won't have to custom code that grain elevator software after all," but to my chagrin, not a single company in the list had anything to do with the agricultural sciences. To each and every Star Partner reading this editorial, please consider the benefits of the program and your relationship with BEA and don't abuse it. If you abuse the partner program, you're really doing yourself, other partners, BEA, and, most important, potential customers a disservice.

A partnership is a two-way street, so continue to put the forces of innovation and marketing behind your partnership with BEA. The benefits will be countless for your company, your partner (BEA), and every customer you encounter together. Last, check to see if your company is in the program's agriculture vertical market and if so, remove it, please.

AUTHOR BIO...
Jason Westra is the editor-in-chief of *WLDJ* and the CTO of Evolution Hosting, a J2EE Web-hosting firm. Jason has vast experience with the BEA WebLogic Server Platform and was a columnist for *Java Developers Journal* for two years, where he shared his WebLogic experiences with readers.

CONTACT: jason@sys-con.com

EDITORIAL OFFICES
SYS-CON Publications, Inc.
135 Chestnut Ridge Road, Montvale, NJ 07645
Telephone: 201 802-3000 Fax: 201 782-9637
SUBSCRIBE@SYS-CON.COM
BEA WebLogic Developer's Journal (ISSN# 1535-9581)
is published monthly (12 times a year)
Postmaster: Send Address Changes to
BEA WEBLOGIC DEVELOPER'S JOURNAL,
SYS-CON Publications, Inc.
135 Chestnut Ridge Road, Montvale, NJ 07645

SYS-CON MEDIA

Integration via Web Services

THE FIRST CHOICE FOR EXTENDING LEGACY APPLICATIONS

BY **SAM PULLARA**

WebLogic Server 7.0 contains the most advanced, performant, and standards-compliant Web service stack of any application server. With an additional download (until the JAX-RPC specification goes final – it may by the time you see this article – see http://jcp.org/jsr/detail/101.jsp) you get a Java standards–compliant Web service stack that also passes the SOAP interoperability tests. So you might ask how easy is it to use this system to call existing Web services and to build new Web services? The answer is: almost trivial.

Let's look at a concrete example to demonstrate a couple of things: (1) is it straightforward to build Web service clients and (2) the Web service stack interoperates with very complicated and intricate non-WebLogic–hosted Web services. For our Web service, let's look at the TerraServer (http://www.terraserver.com). The TerraServer is hosted by Microsoft and appears to be running the .NET runtime on the server. Our first step will be to generate the client libraries required to access the Web service from the WebLogic Server. To do so, run this Ant script:

```
   <project name="verify" default="all" basedir=".">   <target name="all" >
<clientgen
wsdl="http://terraserver.microsoft.net/TerraService.asmx?WSDL"
packageName="net.terraservice.terraserver"          clientJar="terraserv-
er.jar" />   </target> </project>
```

This Ant script uses the WebLogic Web Service clientgen task to generate a client-side service factory, client-side stubs, and statically typed objects from the WSDL provided by the TerraServer's .NET service. All automatically, without user intervention. The JAR that's generated can then be placed in your EJB or WAR to let you access this service at runtime. Using the documentation that Microsoft provides for the service we can write some simple code that gets an image of the BEA San Francisco office in the lower right-hand corner (my office is on the top floor on the left side of the building):

```
TerraServiceSoap service = new TerraService().getTerraServiceSoap();
LonLatPt point = new LonLatPt(-122.4028244, 37.7921829);   TileMeta meta =
service.GetTileMetaFromLonLatPt(point, Theme.Photo, Scale.Scale1m);
byte[] gif = service.GetTile(meta.getId(););
```

You can easily drop this code into a servlet to display the image or use it from a Swing application by JARring up the generated classes from the Ant script along with the classes from the webserviceclient.jar included in the lib directory of your WebLogic Server distribution. Using external Web services directly from your Java or full J2EE application has never been easier!

On the flip side, we have also added tools for deploying stateless session EJBs and arbitrary Java classes as server-side Web service components using a similar Ant task called <i>servicegen</i>. Not only does the task create a deployable module for your service but it also generates WSDL for your clients. This allows you to take either older EJBs that you were using for your normal J2EE applications or new EJBs/classes that you're writing for an application and automatically make then available to anyone with a proper Web service stack. In the future, not long after 7.0 ships, we'll also offer a tool that takes WSDL and generates a stubbed-out stateless session bean implementation to which you can add your business logic. This will let you take an arbitrary service specification and build an implementation without having to start from scratch. All these tools will increase your productivity, extend interoperability, and help you integrate legacy applications into your J2EE application.

Now when you combine Web services, message-driven beans, and EJBs you can see that building a simple message broker becomes a reality for the skilled J2EE programmer. So for small-scale integration systems it's now possible to build your own with off-the-shelf tools and a little hard work. For those that don't want to be distracted with the intricacies of brokering Web services, there will be a large market for products like WebLogic Integration Server in which you can build whole marketplaces from the ground up without spending your development dollars on infrastructure.

Web services are at the forefront of all new integration initiatives and because of the high interoperability between Web service stacks, they are everyone's first choice for extending legacy applications. As more and more legacy applications expose their functionality via Web services the demand for Web service integration servers will only increase. And with new applications also written to be Web services–compliant, we may not have these legacy integration problems in the future. If you build your systems now with future interoperability in mind, your applications can remain a valuable part of the software infrastructure in your enterprise.

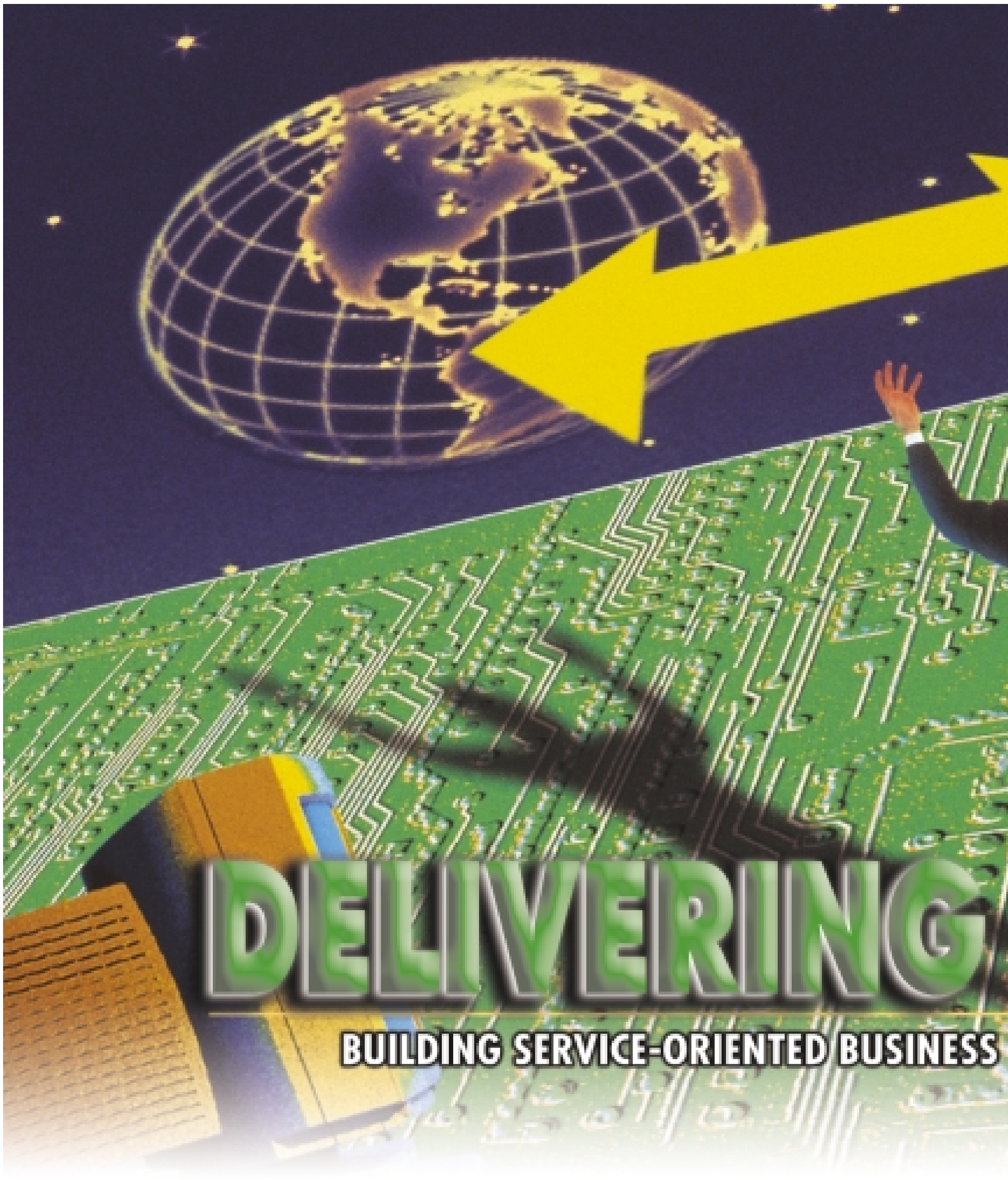
AUTHOR BIO...

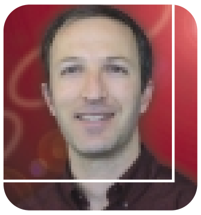
Sam Pullara has been a software engineer at WebLogic since 1996 and has contributed to the architecture, design, and implementation of many aspects of the application server.

CONTACT: sam@sampullara.com

Altaworks

www.altaworks.com





BY
DORON SHERMAN

AUTHOR BIO...
Doron Sherman is CTO of Collaxa, Inc., a Web Service Orchestration Server vendor and a BEA Partner located in Redwood Shores, California. He has been involved with Java since its early days and pioneered application server technology while a founder and chief scientist at NetDynamics.

CONTACT...
doron@collaxa.com

With the rapid adoption of Web services standards and increasing support for asynchronous and XML-based messaging in the J2EE specification (JMS, MDB, JAXM, JAXRPC), it's time to address the challenges involved in building business applications based on a service-oriented architecture.

This evolution impacts current practices for Web application development, which now need to support this new paradigm. It also presents challenges for application deployment and management, commonly addressed today through proprietary enterprise application integration (EAI) solutions. In essence, the notion of a business application is evolving into "orchestration" of services, as the lines between development and integration become blurred.

Service orchestration refers to the assembly of loosely coupled components and the coordination of multiple asynchronous conversations over coarse-grain communication protocols. Implementing orchestration logic in an application produces a consistent set of technical infrastructure requirements, such as managing nonlinear state and long-lived application context, supporting parallel activities and sophisticated join patterns, handling exceptions and events

generated by local and remote components, compensating for failed or cancelled business transactions, and graceful upgrading and management of application functionality.

In this article, I present a sample application demonstrating the orchestration of services implementing a loan procurement business process. I'll describe the technical requirements for orchestration raised by the sample application and how an implementation can leverage the capabilities of the WebLogic Server. Then I'll describe a J2EE-based container for service orchestration that packages the common infrastructure requirements. Finally, we'll show how Java developers can program the orchestration logic of the loan procurement application using a Java-based abstraction.

A New Face for Application Development

It's not an everyday event when multiple technological evolutions, seemingly independent of each other, join together to create sufficient synergy in the software industry to materially change the requirements for business application development as well as for the software platforms that deliver them. The last time something like this happened was in 1993, when a standard protocol (HTTP), a standard presentation markup language (HTML), a Web server, and a Web browser combined to create the ubiquitous publishing medium we know as the World Wide Web. Shortly thereafter, with the introduction and rapid adoption of Java as the language of choice for authoring server-side applications for this medium, the Web application server was born and took the driver seat as the primary mechanism for the delivery of business applications.

Since that time, application development has evolved and matured, in large part due to standardization through the J2EE architecture. This architecture provides an extensible foundation for component-based development of business applications using server-side Java and has facilitated the delivery of self-service applications and user-facing portals into the mainstream. Until recently, these business applications have been characterized by synchronous request/reply interactions with an emphasis on the creation and delivery of dynamic Web pages (presentation), use of encapsulated building blocks (EJB), and connectivity to back-end data stores (e.g., JDBC).

At the same time, organizations have still had to deal with the problem of connecting their information systems to one another, mostly using a variety of middleware technologies and proprietary EAI (enterprise application integration) tools. Even worse, this practice has typically remained separate from the mainstream application development domain and has normally been undertaken by consultants who possess specialized implementation skills. The methodology used for these implementations has continued to steer it away from that used for the delivery of mainstream business applications. However, the continued evolution of the enterprise computing platform and its rapid inroads into technologies such as XML Web services and asynchronous messaging is about to change all this.

ADOPTION OF WEB SERVICES

"Businesses and people that work together need their applications and services to work together." This describes the need for interoperability in the heterogeneous enterprise computing environment. This need is driving the growing move toward Web services, an industry-standard interface and connectivity-technology stack layered on top of standard Internet protocols and a self-describing XML data format. Web services are the new building blocks that will enable application development to extend into and address the challenges of enterprise integration. As functional building blocks, Web services can encapsulate a variety of functionality, such as legacy applications, packaged applications (e.g., ERP, CRM, SCM), and Java components as well as .NET components. WSDL, the interface description language used by Web services, is self-describing, while SOAP, its messaging protocol, provides coarse-grain communication based on human-readable XML data file interchange.

Unlike earlier technologies, such as CORBA and DCOM, Web services are vendor-independent and provide true industry-standard interoperability. Web services can be deployed in various contexts, such as inside or across the firewall, using different transport protocols underlying SOAP (such as HTTP, SMTP, and JMS), and can be simple or composite, synchronous or asynchronous. Regardless of the context, Web services are accessible over SOAP and are introspected through their WSDL interface, making them interoperable. Most importantly, Web services provide for loosely coupled operation by separating the interface from the implementation. Thus, the consumer of a Web service need only rely upon the external contract describing the Web service, without assuming anything about its internal implementation.

THE NEED FOR ASYNCHRONY

Integrating loosely coupled functionality into business applications inherently increases reliability and scalability requirements, because the underlying components are not under the direct control of the application and cannot be guaranteed to always be available. Synchronous access to such components would potentially create performance bottlenecks and severely limit the execution throughput of the application. In any case, application reliability under such circumstances is not possible since the application context resides in memory and is not persisted. Therefore, asynchronous messaging has been introduced in order to address these needs, thus resulting in both improved application throughput and increased overall reliability, by allowing independent execution.

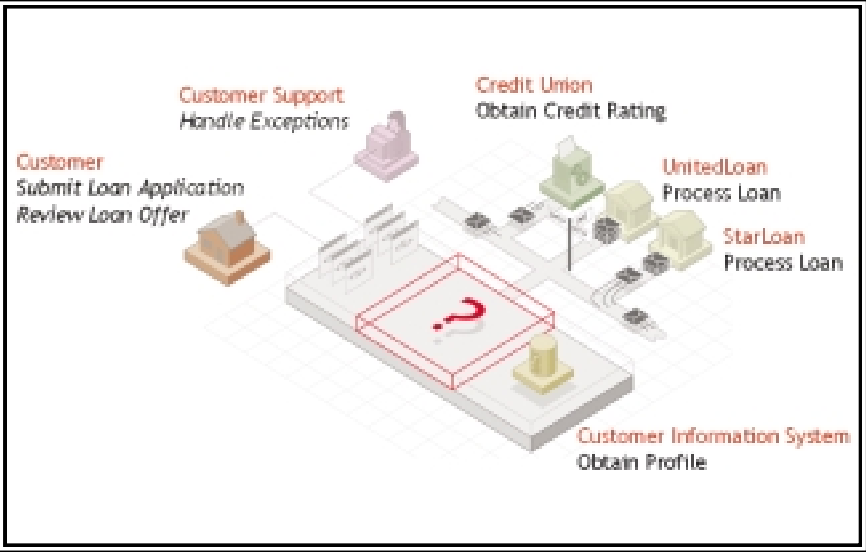
Using asynchronous messaging in a business application opens a range of opportunities for integrating loosely coupled components but also introduces significant new challenges when compared to a synchronous (RPC) programming metaphor. For example, when making an asynchronous call to an external component, the caller doesn't know how long it will take to execute. Therefore, a message correlation mechanism has to be put in place to be able to receive replies and match them to their original invocations. A conversation with a loosely coupled component may fail due to communication or component execution problems (at either end). Therefore, the application has to store its context at the start of the conversation and re-activate it from a persistent store upon completion of the conversation before executing subsequent application logic.

PUBLISHING AND ORCHESTRATING WEB SERVICES

Working with Web services as part of a service-oriented application comprises two steps. The first step involves taking the IT assets that need to be integrated into the application and making them available as Web services – "publishing" them. Publishing Web services is widely supported by the latest versions of the leading J2EE application servers and by a growing number of tools, such as Glue (The Mind Electric), CapeStudio (Cape Clear Software), and several others. With growing support for Web service publishing, EAI tools are quickly losing their traditional competitive edge, i.e., providing proprietary adapters for legacy connectivity.

Orchestration, the second step in working with Web services, entails a more complex set of challenges. Orchestration of Web services involves making the services work together, as well as integrating them into an application. From a development standpoint, orchestration means assembly of loosely coupled services and coordination of the asynchronous conversations in which they are involved into manageable business applications. Experience with orchestration of services as part of a business application reveals a consistent set of infrastructure requirements that repeats itself across such applications. These technical requirements are described in our loan procurement example.

FIGURE 1



Loan Procurement Application

Loan Procurement Example

To better understand the technical infrastructure requirements of service orchestration, let's consider an example in which a loan broker integrates people, internal applications, and external service providers using JMS and Web services. We'll use a simplified version of the loan procurement business process (see Figure 1).

1. A customer submits the loan request through

Mongoose

www.mongoosetech.com

a portal that subsequently starts the execution of the Java orchestration logic.

2. The orchestration logic (shown as a question mark) communicates with a customer information system (through a session bean) and with a credit rating application (through an asynchronous JMS call) to obtain the customer profile and a credit rating.
3. The orchestration logic routes any business exceptions that may be encountered to a customer support rep for manual processing of the application.
4. The orchestration logic then initiates two asynchronous conversations with trading partners UnitedLoan and StarLoan to obtain loan offers that will be presented to the customer as they become available.
5. Once the customer selects a loan offer, the orchestration logic obtains a loan policy from the selected provider and cancels the other loan requests. The loan policy is then issued to the customer.

"The need for interoperability in the heterogeneous enterprise computing environment… is driving the growing move toward Web services"

REQUIREMENTS OF ORCHESTRATION LOGIC

Let's summarize the primary issues involved in developing, deploying, and managing orchestration logic and the technical requirements that need to be addressed in the context of the sample loan procurement application described above.

1. ***Open standards (Java/J2EE, JMS, XML, SOAP, WSDL):*** How do you leverage your existing investment in, and developer knowledge of, Java and J2EE? How is each conversation marshaled into SOAP/XML and JMS messages?
2. ***State and context management:*** How do you coordinate, store, and manage the state of each conversation while utilizing asynchronous messaging? How do you correlate responses to the business transaction that initiated them?
3. ***Loosely coupled services:*** How do you model each conversation to allow for easy adoption of the application as business conditions change? How easily can you add a new loan processor or handle a change in interface?
4. ***Parallel processing:*** How do you design and coordinate parallel conversations (e.g., submit the loan application to UnitedLoan and StarLoan in parallel)? How do you implement sophisticated join patterns ("Cancel the conversation with UnitedLoan if the user selects the offer from StarLoan")?
5. ***Exception management:*** System- and business-level exceptions increase the variability and complexity of orchestration logic tremendously. What happens if UnitedLoan refuses to process the loan application because the information provided is invalid?
6. ***Events and notifications:*** Conversations can span a long period of time and include multiple responses. How do you handle notifications such as loan approval and intermediate progress events? How do you specify and handle timeouts?
7. ***Open nested transactions:*** How do you combine multiple nonlinear conversations into a business transaction? How do you track the history of activities to allow for compensation, if necessary (since you can't roll back someone else's database…)?
8. ***Scalability and performance:*** What happens when the scope of the application increases and you need to support more services, more complex types of conversations, and a higher transaction volume?
9. ***Distributed administration:*** What happens if customer support decides to cancel a submitted loan application request? What happens if a change request occurs in the middle of a conversation?
10. ***Business visibility:*** How do you provide visibility into the state of the conversations? How easily can a business user obtain information on the number of loan requests processed per day or on a particular loan application?
11. ***Version control:*** How do you update the orchestration logic to reflect new business rules? How do you gracefully phase in new versions when 10,000 loan requests are pending?
12. ***Audit trailing:*** Can you trace the history of all conversations related to a specific loan request? Can you provide nonrepudiation for the application?

THE CGI LESSON

Time-to-market and cost pressures, efficient utilization of available developer skills, and management of application delivery risks are the constant elements in the changing world of enterprise software. Similar forces were in effect during the mid-'90s when just about every company was hard pressed to deliver a dynamic Web presence to its customers. To address this need, the static content publishing medium was augmented with a Common Gateway Interface (CGI) that enabled publication of dynamic content on the Web.

The delivery of simple Web applications that quickly grew in complexity introduced a common set of technical requirements. Developers found delivering those requirements in a custom, one-off fashion to be a significant impediment to cost-effective development and management of those applications. Session management, resource pooling (e.g., database connections) and multithreading, to name a few of these requirements, were then embedded in a new piece of infrastructure software that provided a compelling buy-versus-build option for IT decision makers and developers. This new piece of software, which became known as the application server, saved developers from repeatedly implementing (and maintaining) the same application infrastructure. Developers could now focus on authoring the application business logic, saving valuable time and increasing productivity.

In addition, this class of self-service and customer-facing applications became easier to deploy and manage in production with the use of an application server. The CGI lesson, i.e. encapsulating application infrastructure and plumbing and enabling developers to focus on the application business logic, is directly applicable to the problem domain of service orchestration.

Extending J2EE into Orchestration

As we observe the ongoing evolution of the J2EE computing platform, we get a clear indication of how orchestration fits in as J2EE expands to support asynchronous messaging and XML Web services.

FROM JDBC TO EJB

Prior to the introduction of JDBC as a Java API, most application logic was coded using proprietary 4GLs (e.g., Oracle's PL/SQL) and stored procedures. The introduction of JDBC enabled developers to access tabular data sources (most notably SQL databases) in a standard fashion and caused a major shift in business logic programming to the middle-tier.

The widespread adoption of JDBC allowed a layer of data access components to be included in the J2EE application server, eventually standardizing as EJB component technology deployed using an EJB container and providing the underlying database access services and infrastructure. This evolution not only made coding at the JDBC API level unnecessary, but it also elevated the programming abstraction and enabled developers to focus on coding logic and avoid dealing with mundane application infrastructure and plumbing complexities.

THE JSP ANALOGY

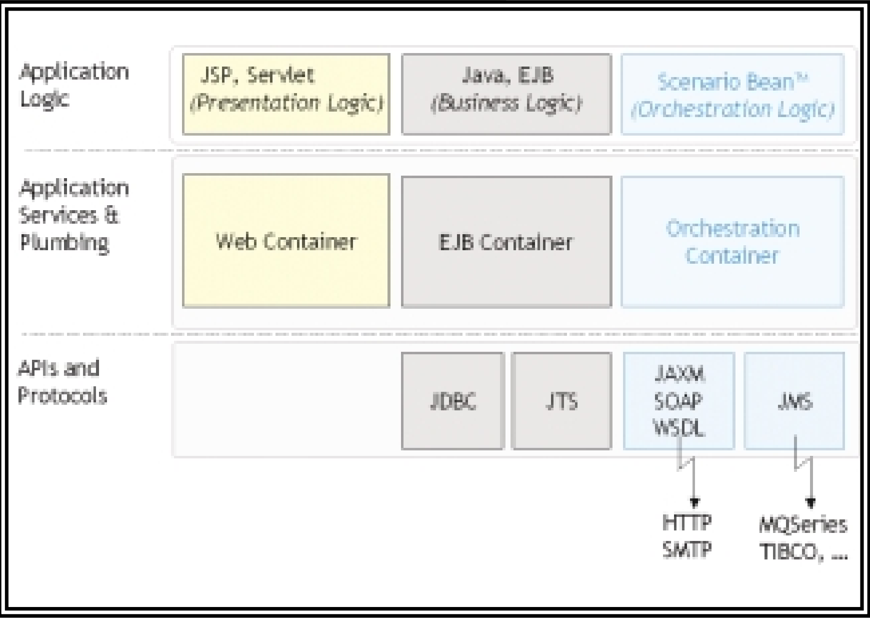
Java is an extremely flexible and powerful programming language and is particularly useful for authoring server-side logic for Web-based business applications. Since many business applications were intended to provide self-service functionality to end users, developers have sought ways to streamline development of the front-end or presentation-logic tier of these applications. The key to streamlining such development has been congruence – finding a straightforward mapping between the problem domain and the solution domain. JavaServer Pages (JSP) was created as part of the J2EE architecture (see Figure 2) to become the Java-based abstraction-enabling congruence for the development of presentation logic.

JSP technology allows Web developers and designers to rapidly develop, and easily maintain, information-rich, dynamic Web pages. JSP separates the user interface from content generation, thus enabling developers to alter page layout without changing the underlying dynamic content. This separation of page logic from UI design and display, and a reusable component-based design is made possible with the use of XML-like tags combined with Java scriptlets. With these features, JSP has simplified and accelerated the process of building Web-based applications.

THE IMPACT OF MESSAGING

Recently, JMS, JAXM, and related APIs and protocols have been introduced into J2EE to provide

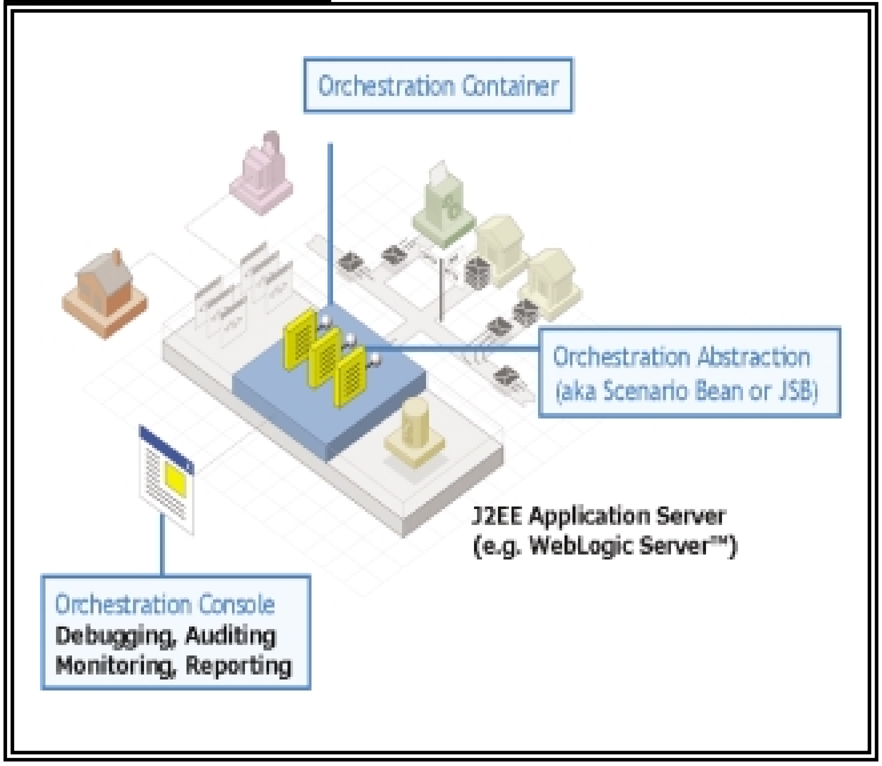
FIGURE 2



Extending the J2EE Architecture

"Using asynchronous messaging in a business application opens a range of opportunities for integrating loosely coupled components" access from Java to asynchronous messaging and XML Web services. This is expected to induce a similar shift in how asynchronous business logic handling of loosely coupled services is programmed. Currently, coding to JMS and JAXM APIs is being hailed as the enabling access layer but it's expected that developers will eventually look for higher-level programming abstractions. Abstracting orchestration logic will leverage JMS and JAXM in much the same way that EJB leveraged JDBC and JTS. This migration of orchestration logic from proprietary EAI solutions (*aka* Businessware, BPM, Workflow, etc.) into the middle tier has already started and is expected to accelerate as effective and easy-to-learn means for programming orchestration logic become available.

FIGURE 3



Web service orchestration server

THE ORCHESTRATION CONTAINER

As is the case with EJB technology, orchestration requires a runtime environment, or container, for deployment. The orchestration container (see Figure 3) implements the complex plumbing and technical requirements (detailed in the earlier example) that don't need to be exposed to developers. A matching component model, called here the JSB, or Java Scenario Bean, enables effective programming for authoring orchestration logic. JSB is a powerful abstraction that assembles loosely coupled services and sequences the conversations among those services. Since these conversations involve asynchronous communication, JSB enables authoring long-lived, multistep orchestration logic across these services. Furthermore, JSB becomes a natural extension of the J2EE architecture since it is fully interoperable with its components, such as JSP, servlets, and EJB. The orchestration container provides a seamless bridge to XML messaging so developers don't need to deal with cumbersome Java-to-XML (and XML-to-Java) datatype mapping.

Leveraging WebLogic Server

As we've demonstrated above, implementing the complex requirements of orchestration can naturally fit as an extension of the J2EE architecture. The orchestration container, based on J2EE, needs to provide a scalable execution environment for the functional components involved in orchestration logic. The orchestration container is then deployed on top of a robust and scalable J2EE application server such as BEA's WebLogic Server. In addition to leveraging the runtime environment of a J2EE application server, orchestration logic also coordinates transactions among loosely coupled components and, through JMS, provides support for asynchronous messaging. Finally, the orchestration container uses the WebLogic Server two-phase commit (XA) transaction capabilities and its high-performance JMS implementation.

The code for this article can be found on the Web at www.sys-con.com/weblogic/sourcec.cfm. Listing 1 shows the Scenario Bean source code for the example loan procurement application. Listing 2 shows the associated XML-based deployment descriptor for the JSB.

Conclusion

As you've seen here, building enterprise applications that integrate functionality represented as loosely coupled services produces the need for orchestration. Orchestration involves the assembly of services and the coordination of asynchronous conversations across loosely coupled communication protocols. There is a need to effectively address the complex technical requirements of orchestration, highlighted through the loan procurement example. With the continued advance of J2EE, which provides API access to asynchronous and XML messaging, we demonstrate a nonintrusive approach that naturally extends the J2EE architecture and provides an open application model.

The illustrated solution is composed of an orchestration container that takes care of the infrastructure and plumbing requirements and a JSP-like abstraction that enables powerful, yet easy, authoring of orchestration logic by Java developers. Finally, the use of an orchestration container allows for versatile debugging, monitoring, reporting, and management of the resulting application.

Sitraka

www.sitraka.com/performance/wldj

Application Integration: Forms, Degrees, and Mechanisms

INTEGRATION OFFERS TREMENDOUS OPPORTUNITY — AND SIGNIFICANT RISK

BY ERICK RIVAS

In a March 2002 survey by Morgan Stanley, 225 CIOs listed extending their current IT investments through application integration as their number one priority. This is not surprising. Given the slow economy, many Fortune 5000 companies are putting off large new packaged-application initiatives, and are looking to make do with what they've got. Why start a multimillion-dollar ERP project when half the company's users can't make use of the applications that are already installed (or sitting on the shelf)?

But when I talk to CIOs about their "application integration" needs in more detail, I don't get a consistent story. One of the reasons for this "impedance mismatch" is that there are really several different forms of application integration. This creates a huge risk because vendors and IT buyers are nodding in agreement about the need for application integration, but each has a different vision of what that means.

In a former life, I encountered the same problem as I managed the integration of software products acquired by a major software vendor. Let me offer you this more concise way to describe application integration, along with examples and the pros and cons of each approach. Using this nomenclature may help you avoid a misunderstanding that results in disappointment or project failure.

Forms, Degrees, and Mechanisms

There are four basic forms of application integration:

- Presentation
- Workflow
- Data
- Component

These integration forms are largely independent of each other and they address the layers in a traditional architecture stack. Each integration *form* can be taken to many *degrees* (based on ROI), which correspond to the complexity or depth of application integration. Integration *mechanisms* refer to the implementation technology used to accomplish the integration.

PRESENTATION INTEGRATION

The presentation form of application integration has to do with having a consistent look and feel (also known as a skin) and common commands within an application and between applications. It also involves the personalization and customization of different application views, depending on which user is logged on.

Example Mechanism: Portlets

Portlets are user-interface components that end users see within a portal. Like a window on a PC or Mac, each portlet owns a portion of the browser or wireless device's screen where it displays views of an application (or content) and provides results. Portlets can be as simple as a message board or as complex as an order entry into an ERP application. Portlets can be designed, assembled, deployed, and managed using a variety of enterprise portal solutions. Some portal products have the ability to rapidly create portlets out of existing content or Web applications, and leverage a variety of technologies (e.g, JSP, servlets, .NET, DHTML).

Pros:

- Portlets are low risk, and give quick, high bang for-the-buck results.
- Portlets allow you to selectively expose different pieces of application functionality to employees, customers, partners, and suppliers without having to install software on each client (or future upgrades).
- Portlets Web-enable existing applications and provides a way to surround them with other enhancing tools and technologies (like collaborative Web services).

Cons:

- Doesn't add new functionality to (or replace) existing applications
- Encapsulating an existing application that is already disliked and putting it within a portal may be like putting lipstick on a pig. It may be pretty, and pretty disappointing.

WORKFLOW INTEGRATION

Workflow integration enables organizations to create and manage Web-based business processes between employees, customers, and business partners.

Example Mechanism: Event/Condition/Action (ECA) Rules Engines

ECA rules engines automate business processes in response to events involving systems or collaborative processes within a group of users. The rules engine receives events that are sent to it as events occur inside a portal or enterprise application. The rules engine applies conditions to the events and if they fall within the condition parameters, the rules engine invokes an action. The rules represent business logic for deciding whether e-mails are sent, alarms are triggered, counters are advanced, or other soft-

Precise

www.precise.com/wldj

ware services are invoked. Examples of products that include business process automation tools are BEA WebLogic Integration, Savvion Business Manager, and Mongoose PortalStudio.

Pros:

- Enable you to automate and streamline business processes that include interaction among people and systems in order to reduce costs and time delays.
- Provide consistency, respond to exceptions, and take corrective action.
- Routes tasks, documents, and information between people and systems.

CONS:

- If not properly designed, workflow integration can be inflexible because it's a tightly coupled way to talk to existing systems.

DATA INTEGRATION

Data integration is the most common form of application integration. Every business needs to aggregate, transform, and integrate data in order to reduce costs and inventories, increase revenue, and improve competitive positioning. There are a wide range of formats (e.g., XML), metadata, infrastructure, databases, and applications with which business objects and transactions interoperate.

Example Mechanism: J2EE CA Connectors

J2EE Connector Architecture (J2EE CA) is a nonproprietary standard for integrating applications with existing enterprise information systems. BEA, Insevo, Resource Adapters, TIBCO, and webMethods currently offer a variety of prepackaged J2EE CA-based connectors for leading ERP, CRM, and other corporate applications.

Pros:

- Allows you to share data across and beyond your enterprise.
- Eliminates the time, effort, and expense of a traditional EAI or B2B integration project.
- Provides a real-time view of your business, allowing you to get up-to-the-minute information from all of your systems and applications.

Cons:

- J2EE CA–based connectors aren't available for every corporate application.
- Some connectors do not support bidirectional data transfer.
- For highly complex data integration requirements (longer and more complex value chains), J2EE CA alone may not be sufficient.

COMPONENT INTEGRATION

Effective sharing and reuse of code is one of the holy grails of software engineering. Component integration has to do with developing fundamental building blocks (components and frameworks), making them readily available, and then leveraging them across enterprise systems in order to build robust and flexible systems with reduced development time and cost.

Example Mechanism: Web Services

Web services are self-contained, self-describing, component-based applications that can be published, located, and invoked across the Web. Web services perform discrete functions that can be simple requests or complex business processes. Standard technologies like XML, UDDI, SOAP, and WSDL are key parts of the Web services architecture. A Web service can be invoked from a portlet and can work with J2EE CA-based connectors (it's not a "one or the other" scenario). Products for developing Web services include AltoWeb, BEA WebLogic Workshop, and Microsoft Visual Studio .NET. Examples of XML-based Web services are Microsoft Passport and Sun Microsystems' Liberty.

Pros:

- Reduces the cost of building new applications and shortens the time it takes to get them into production.
- Increases interoperability between the various development environments so developers don't have to worry about which programming language or operating system they are using. Costs vary widely with each individual Web service. Commoditization will drive these costs down.

Cons:

- Web services are currently at the top of the hype cycle. Some clients talk to me about adopting Web services and at the same time they ask me just what a Web service is. It's important to set the expectations here.

No single integration form is a silver bullet; different companies have different integration needs. In many companies, there are several formal IT initiatives going on in parallel involving multiple integration forms. The specific integration strategies adopted by a CIO depend on many factors. This includes how well existing applications already work together and figuring out what type of integration would provide most bang for the buck (ROI analysis).

One survey said the average Fortune 500 company has 62 major corporate applications. These are normally a combination of commercially available ERP, CRM, or SCM applications, legacy (old, outdated) applications, database systems, and custom-built (proprietary) applications developed in-house using the latest and greatest technologies.

In some cases, the best integration strategy for the short term is to reach for low-hanging fruit. That is, to implement a point-to-point data integration between two applications that requires little time and/or resources. But it is important to keep in mind that point-to-point integration between applications, if not done carefully, can actually exacerbate the overall application integration problem in the long term.

Conclusion

With the current economic slowdown, there is a great demand to do more with current IT investments by integrating existing applications. While there is a tremendous opportunity for those who can deliver application-integration solutions, there is significant risk of failure due to the misunderstanding of the functional requirements, implementation, and business value of a proposed application-integration solution. By defining application integration in terms of forms, degrees, and mechanisms, you can manage expectations, drive requirements, and deliver the most business value.

AUTHOR BIO...

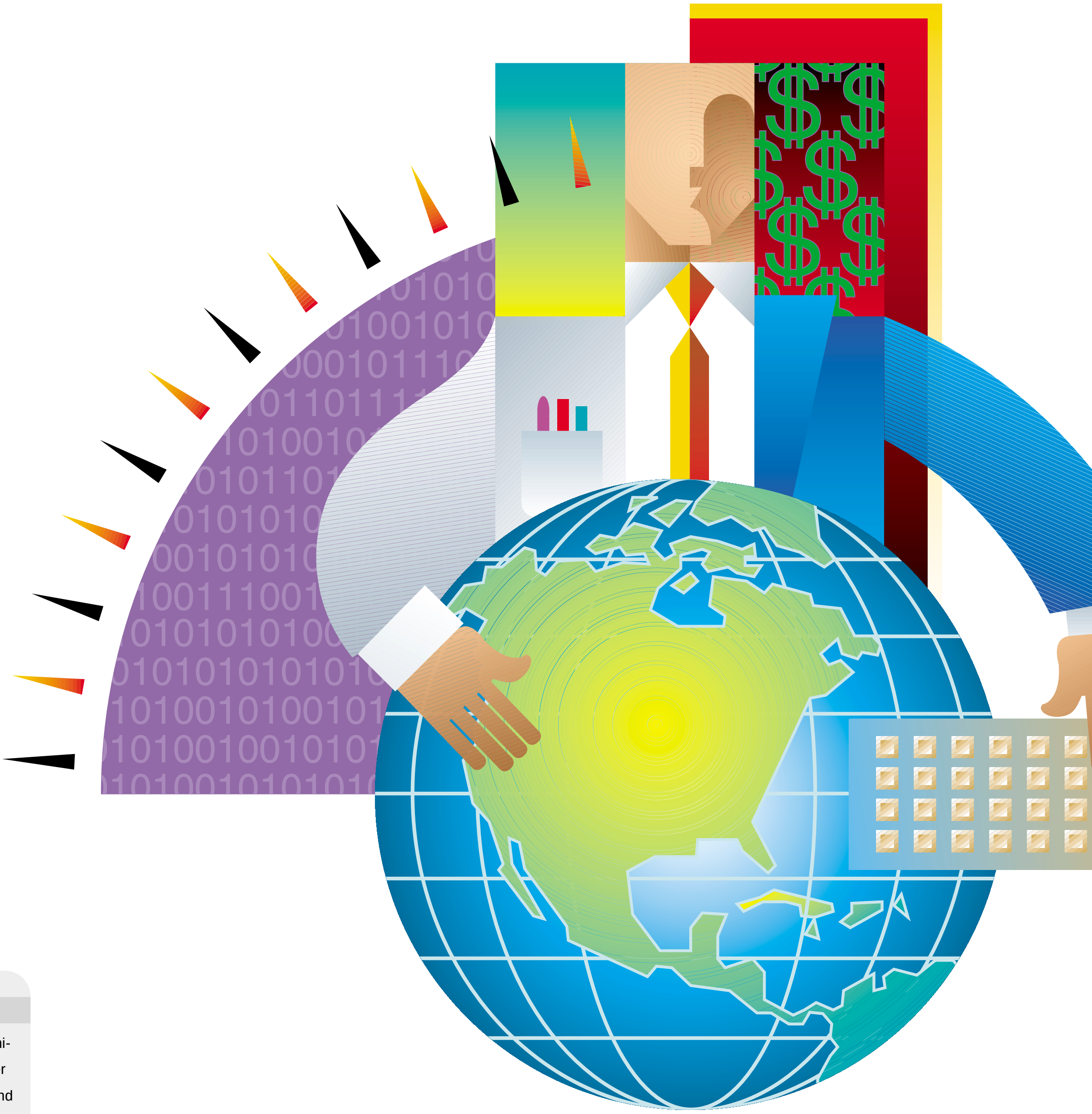
Erick Rivas, president, CEO, and founder of Mongoose Technology, has over 10 years of experience, including everything from software architecture, design, and development to the management of sales, finance, and distribution operations. Prior to founding Mongoose, Erick was cofounder and VP of development at ProtoSoft, a leader in UML-based analysis and design.

CONTACT: erick.rivas@mongoosetech.com

Rational

www.rational.com/ruc

WebLogic &



BY
GUY MOLINARI

AUTHOR BIO...

Guy Molinari is a technical architect and infrastructure developer with application development and data management experience spanning 15 years. For the past three years he has focused on Internet-based application services.

CONTACT...

guy_molinari@hotmail.com

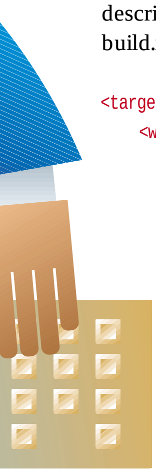
As a developer for a company selling application services to Fortune 500 companies, I was excited about the release of WebLogic 6.1. While its support of J2EE 1.3 features in and of itself is a reason to upgrade or adopt this product, the introduction of SOAP/WSDL-based Web services technology is a significant step forward for an already great application server. What I found interesting about BEA's approach is how relatively simple it is to tap the power of this new infrastructure. For RPC (synchronous-style) services it is a straightforward two-step process.

.NET

MAKING WEB SERVICES THAT WORK

First, create a stateless session EJB that implements the application logic. The only catch here is to design the interface method signatures in such a way that only simple objects are passed. Think of these objects as complex types that expose no behavior other than get/set functionality to manipulate properties (as per the JavaBeans spec). Arrays of complex types and nesting of complex types are also supported. The key design principle is to keep the approach simple and drive behavior into the service itself. If heterogeneity is a design goal, then don't think of Web services as an object serialization technique but rather as a way of exposing business-event functionality.

Second, use the wsgen Ant task to create the enterprise archive .ear file. WebLogic bundles this task as well as Ant version 1.3 with the WLS product distribution. The product documentation provides a detailed description of exactly how to accomplish this. My build.xml scripts include the following snippet:



```
<target name="webservice" depends="init">
    <wsgen destpath="/xylo/dev/${beanname}.ear" con
          text="/${beanname}" protocol="http">
       <rpcservices
          path="${jardest}/ejb_${beanname}.jar">
          <rpcservice bean="${beanname}"
                  uri="/${beanname}"/>
             </rpcservices>
         </wsgen>
     </target>
```

This defines a new target called *webservice*. It depends upon a target called *init*, which by convention defines environment variables and parameters. All of my scripts define the *beanname* variable that's used in various ways to name the necessary files. This target will create and deploy the .ear file within a running WLS instance. That's all there is to it! WLS 6.1 implements a conservative subset of SOAP and WSDL in a very simple way. It leverages the framework already established by EJB and provides the developer with a mechanism for exposing existing EJB components as Web services without any additional Java coding.

Bumps in the Road Are Inevitable in Any Journey

I was working on a project that required that .NET clients be able to call WLS-deployed services. My code worked well when my method signatures used native types (as defined by XML Schema) as a parameter or return value. Unfortunately, complex types or arrays of any type were another matter entirely. This is an unacceptable limitation in the face of real-world requirements. I turned to the WLS documentation looking for insight and saw a note stating that WLS 6.1 supports version 2.0 SP2 of Microsoft's SOAP Toolkit. As a key player on the W3C XML committees, Microsoft decided to adopt the 2001 XML Schema (XSD) specification in its .NET Framework. The SOAP Toolkit is based on the 1999 recommendation, as is WLS 6.1

It's ironic that there's a need for this article, considering that Web services are being touted as the interoperability panacea. In all fairness, the issues addressed arise because newer technologies forming the foundation for emerging protocols such as SOAP and WSDL are still evolving. Understandably, application server vendors must provide infrastructure components that are stable and reliable. These needs must override the desire to support the latest and greatest industry specifications. Undaunted, I decided to dig down deeper into the interoperability issues in the belief that these problems are solvable, given the simplicity and elegance of XML-based Web services technology.

Interoperability Issues

This article outlines the XML-related interoperability issues a developer must face, given the current state of WLS and .NET. They can be categorized into two areas of concern pertinent to WSDL and SOAP, respectively. WSDL (Web Services Description Language) seems like a logical place to begin since before a newly created Web service can be invoked, it must be bound to the calling process.

WSDL STACK INTEROPERABILITY ISSUES

When a WLS Web service is deployed, a public-facing Web page (JSP) is made accessible. This page contains two links: the first generates the WSDL file that describes the service, and the second allows for the downloading of a file called *client.jar* by convention. This .jar file contains the client-side proxy class, SOAP codecs, and some fairly lightweight infrastructure classes provided by BEA to simplify the consumption of services. This is the only .jar file needed in the CLASSPATH, other than client application–related code. For Java-based clients the WSDL is extraneous and is useful only for UDDI registries. For non-Java clients, the opposite is true.

Considering the problem, I knew that the first step in consuming the service was to create the necessary client-side proxy with C# code (although VB.NET would be an equally viable alternative). As a Java programmer with a C/C++ background, I felt comfortable with C# almost immediately, which drove my choice of languages. After downloading and installing the ASP.NET package, I immediately learned that Microsoft's approach to Web services involves the compilation of the WSDL to generate the proxy code. This can be accessed via the command-line utility WSDL.EXE or by using Visual Studio .NET. These tools are HTTP-enabled and can retrieve and compile the WSDL in one step. The WSDL compiler generates a source file with a .cs extension named by the name attribute of the *service* XML element in the WSDL file. This file is then compiled into a .dll. The service is called from the client code by instantiating the generated proxy class and invoking its methods as if they were local to the client process. In my case, the WSDL document created by WLS wouldn't compile correctly with the .NET-provided compiler.

The Web service I created and deployed on WLS is called *PresentationController* and has one method, named *dispatchAction*. This method takes five parameters. The first four are simple string types and the last is an array of *NameValuePair* JavaBean instances specific to my application code. The WSDL file generated by WLS is shown in Listing 1 (the source code for this article can be downloaded at www.sys-con.com/weblogic/sourcec.cfm). Notice the *description* element at the top of the document. The namespace attribute references the 1999 XSD spec. This is the first of three changes that must be made to the WSDL to bring it up to 2001 compatibility. Because my code utilizes the *NameValuePair* JavaBean, the WSDL contains an embedded XSD schema enclosed with a *schema* element. This element contains standard XSD definitions of any complex types that are referenced by the service described. The 1999 XSD specification allowed members of a complex type to be specified using *attribute* elements, while the 2001 spec uses *element* elements (confusing, eh?). Also, the 2001 spec mandates that all members of the complex type be enclosed within a *sequence* element to preserve order occurrence. The modified and MS-compatible WSDL file is shown in Listing 2. I wrote a simple tool that parses the WSDL file as an XML DOM, makes the changes I've previously described, and generates the converted document. The code for this utility is in Listing 3. A slicker approach would be to create a chained servlet that does the conversion on the fly without intermediary files. This exercise is left to the reader.

"Application server vendors must provide infrastructure components that are stable and reliable. These needs must override the desire to support the latest and greatest industry specifications"

SOAP STACK INTEROPERABILITY ISSUES

The next category of interoperability issues revolves around the SOAP messages sent from client to server. As with WSDL, the .NET platform SOAP messages conform to XSD-2001. For the most part these messages are backward-compatible with the WLS FastParser. The showstopper is in the SOAP ArrayType encoding for arrays passed from the client. This is a problem with arrays of native types as well. According to the WLS release notes, there is a known issue, #055596: "WLS Web services do not support SOAP 1.1 multireference compound data types." Again, no workaround is given. XSD-2001 provides a slick optimization technique that allows a node to be referenced in multiple places. For large arrays this can greatly reduce the size of the message, thus conserving bandwidth and shortening parse times. Similar to the WSDL converter, what is needed is a mechanism to filter SOAP interactions.

The Microsoft .NET Framework provides an API for intercepting and altering the SOAP XML payload. It is called SOAP Extensions and can be utilized in client-side proxies as well as .NET-hosted services. Listing 4 contains the C# source code for an extension called *DispatchExtension*, as well as its related *DispatchExtensionAttribute* attribute class. It's beyond the scope of this article to describe in detail how to write .NET SOAP extensions; suffice it to say that the method *fixXML()* does most of the heavy lifting. For the sake of simplicity this is a brute-force approach that completely discards and rebuilds the message body. As an exercise for the reader, this code could be generalized to automatically detect array types and apply intelligent filtering.

When you've written and compiled your SOAP extension, there's one final step. That is to register the attribute class with the appropriate method in your generated proxy class. The following C# snippet from the *PresentationController* proxy class shows how this is done:

```
[DispatchExtension()]
public string dispatchAction(string arg0, string arg1,
string arg2, string arg3, NameValuePair[] arg4) {
```

In the snippet above, the attribute constructor for *DispatchExtension* is placed before the *dispatchAction* method. Be careful! If you change the interface of your service you will need to reapply this modification.

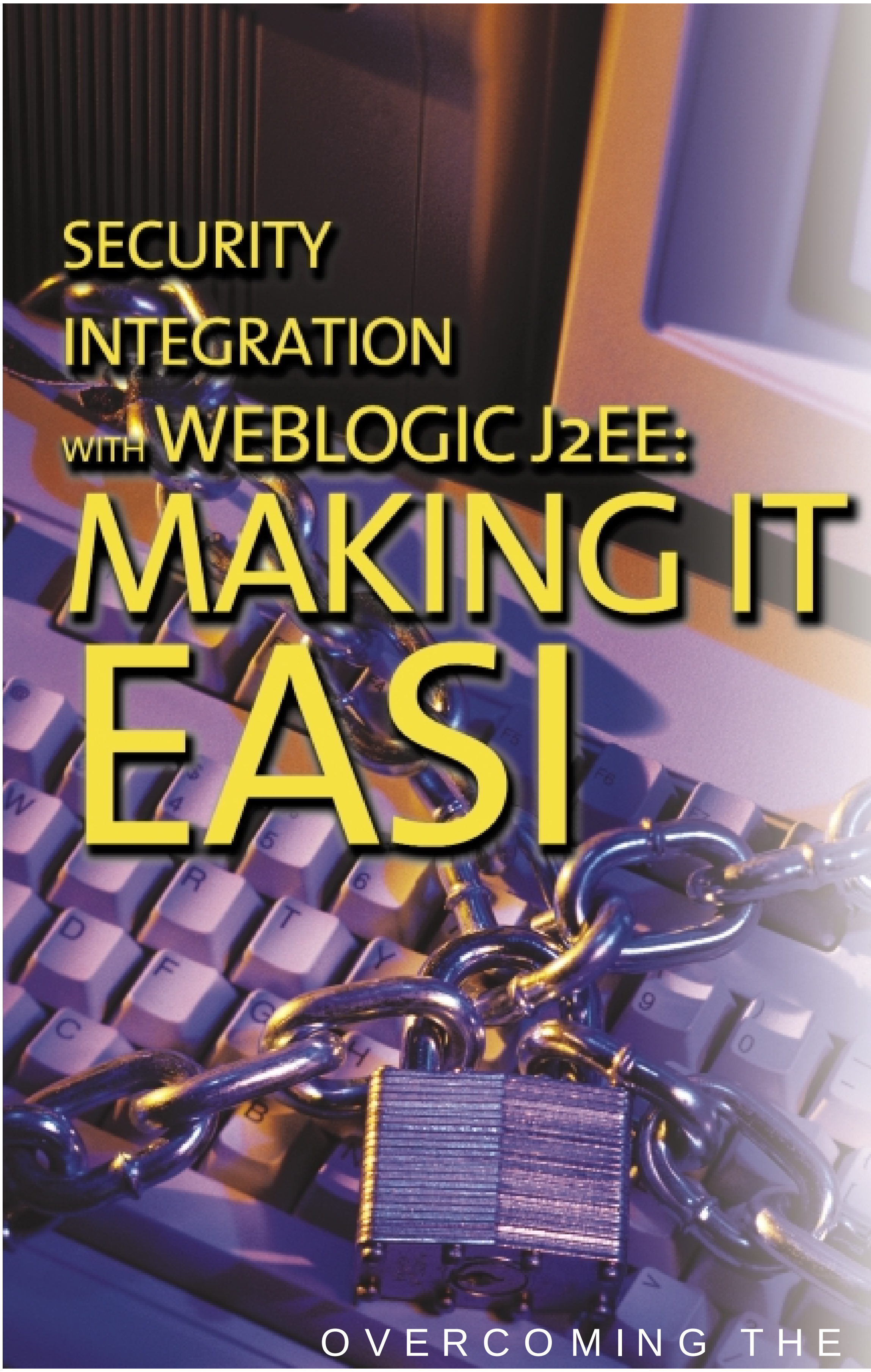
Conclusion

When resolving interoperability challenges, be sure to plan for the eventual removal of adapter code as infrastructure vendors evolve their product offerings and correct software deficiencies.

Hopefully, the next release of WLS will include an update of the XML serialization/deserialization infrastructure and SOAP codecs to resolve these issues. Also, the ability to manipulate the SOAP interactions via a callback interface would be a big plus. Overall, BEA has done a superb job in delivering entry-level Web services capabilities in its currently shipping product release. This platform, combined with the excellent tool offerings of .NET, will provide the developer with a best-of-breed toolbox of architectural options.

WebGain

www.webgain.com/toplink_create3.html



SECURITY INTEGRATION WITH WEBLOGIC J2EE: MAKING IT EASI

OVERCOMING THE

BY
SEAN DOLAN

AUTHOR BIO...

Sean Dolan is a security architect at Quadrasis, a business unit of Hitachi Computer Products (America), Inc. He has 16 years of experience in commercial and scientific systems and application software development, including experience architecting approaches to the integration of J2EE Application Server products into enterprise security frameworks. As a member of the Sun Java Community Process (JCP) program he has participated in the definition of a number of J2EE security–related specifications.

CONTACT...

sean.dolan@quadrasis.com

The rise in popularity of Web services is breaking down barriers and obliterating isolated silos everywhere.

That's good news for any organization looking to make e-business the normal way of doing business with customers, colleagues, and partners. Of course, with every falling barrier comes a new set of challenges around the issue of security. In this article, I'll review the challenges of security integration with WebLogic J2EE applications in a distributed environment – and how they can be addressed with an Enterprise Application Security Integration (EASI) framework based on Web-centric industry standards.

The Challenge of Distributed Security

First, let's agree on the two high-level goals of any security infrastructure in a distributed environment: to limit access to applications, data, and other resources to *only* those with proper authorization; and to make accessing those resources by authorized parties as easy as possible. Meeting both of these goals can be complicated.

In a typical WebLogic implementation, at the front-end, or *perimeter*, tier, we often have Microsoft IIS (Internet Information Server) serving Web pages and providing a first line of security. IIS authenticates users by accessing *user profile* information, including group privilege attributes, residing in Microsoft ADS (Active Directory Service). At the *application* tier, we have WebLogic J2EE applications. These provide user authorization based on *user roles*, residing in WebLogic repositories as EJBs. These WebLogic applications, in turn, interact with mainframe and other enterprise systems in the *back-office* tier. In addition, most organizations have legacy security infrastructures composed of point solutions that provide protection for specific systems or applications.

In addition to the basic challenges of interoperation, such an architecture presents a major maintenance headache. IT must manage two sets of user/group authentication data – one for IIS ADS and the other for WebLogic repositories – to ensure they "map" to each other. This task is multiplied when other domains are involved. And it's further complicated by the fact that IIS attributes and WebLogic roles are arranged according to different hierarchies.

Single Sign-on (SSO), with which a user can log in once and gain access to all authorized resources, is an important feature from the user's point of view. Proper implementation of SSO in distributed environments can be extremely complex. One workaround is to send a user's password from IIS along to WebLogic applications, legacy security solutions, and, in some cases, even back-end systems. This is considered poor security practice, introducing the potential for breaches. And it can diminish system performance, forcing users to wait longer for authentication.

Three Common Options

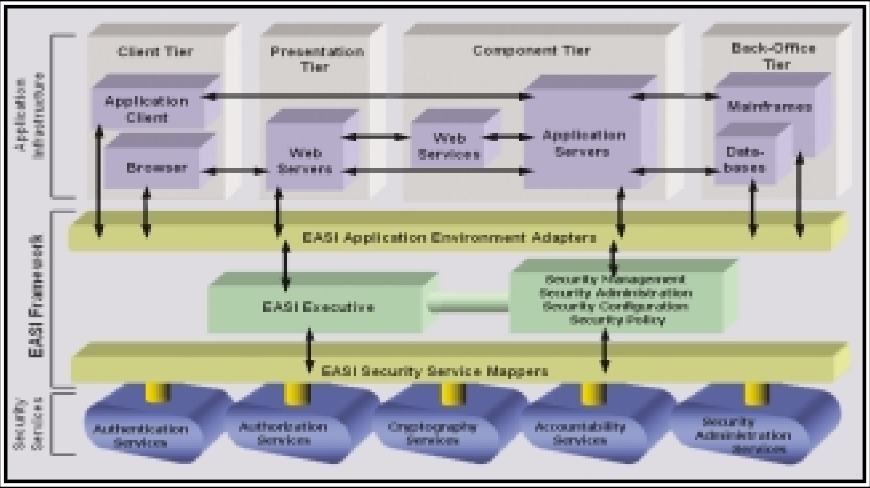
Under pressure to put new applications into production, IT groups facing these challenges have several options.

First, they can disable the security facility built into WebLogic and rely on IIS perimeter security alone. This eliminates the need to map the two user repositories, but it greatly diminishes control over access. IIS does not provide very granular access control. And it isn't known for its robustness – it's certainly not the single line of defense one wants in today's hacker-rich environment.

Another option is to disable IIS perimeter security and rely solely on WebLogic for access control. This approach leaves the front end unprotected, eliminating the critical screening that IIS performs. Web pages become vulnerable to defacement, and sensitive content is no longer properly protected against unauthorized access. Again, not a good option.

The third option is to bite the bullet and try to get IIS and WebLogic to interoperate. As already explained, this raises a host of integration issues associated with getting two dissimilar repositories of user data to communicate. This is typically a costly, proprietary, and very complicated alternative.

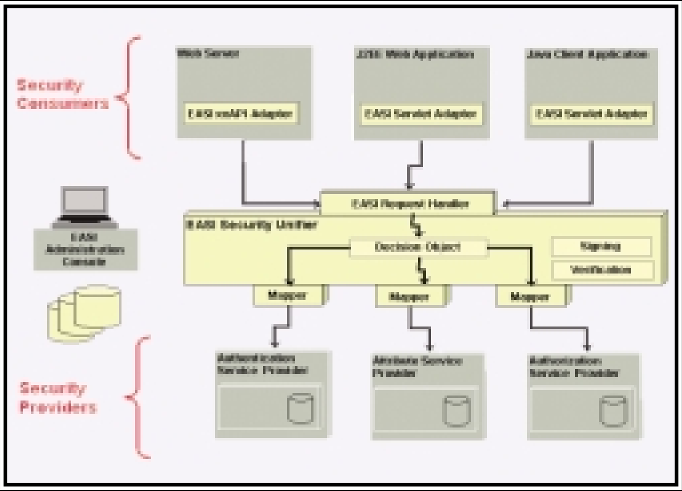
FIGURE 1



EASI framework

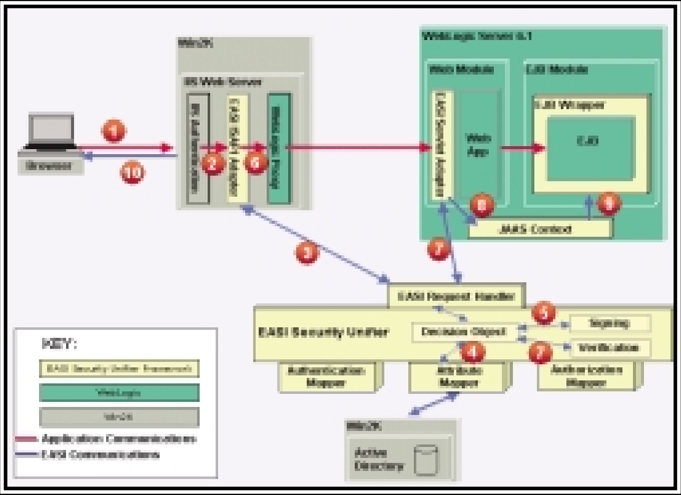
CHALLENGES OF INTEROPERATION

FIGURE 2



Security consumers and security providers

FIGURE 3



EASI framework interactions

A New Option: Unified Security

So how can IT groups bridge the gap between the Microsoft world (ADS) and the WebLogic world (Java/EJB) to achieve seamless security interoperation in such a distributed environment?

The answer lies in a standards-based Enterprise Application Security Integration (EASI) framework (see Figure 1) that can map user/group attribute and authentication information among IIS, WebLogic, back-end systems, and standards-based legacy security services. The result is a unified, "pluggable" security infrastructure that strengthens security, while streamlining secure access and greatly simplifying security administration and system evolution.

This security framework divides the world into security consumers and security providers using a variety of code types (see Figure 2). Security consumers make requests to authenticate a principal, to obtain privilege attributes, or to authorize access to a protected resource. Security providers process authentication, attribute, and authorization requests according to the particular technology upon which the provider is based. The security framework connects security consumers, using adapters, with security providers, via mappers. The framework controls the selection of which mapper to use to satisfy a request from a given adapter. The "pluggability" of this architecture arises from the fact that the framework can be configured to use any mapper to process a request from any adapter. Figure 3 shows the basic elements of this EASI architecture.

While an EASI framework leverages a number of open industry standards (such as XML and SOAP), key to its effectiveness is its use of standards arising from the OASIS SAML (Security Assertion Markup Language) initiative. SAML assertions provide a standardized way to represent user and/or group security statements and security requests. SSL, digital signatures, and other cryptography technologies are used to secure communications within the framework.

With this framework, developers have the flexible tool they need to knit together their entire distributed-security infrastructure – overcoming the differences among IIS, WebLogic applications, back-end systems, and their associated repositories. It can dramatically reduce the cost and time required to implement a WebLogic environment, including multidomain environments, while enhancing security effectiveness. It greatly simplifies the administration and maintenance of that distributed environment, as well as the task of integrating new applications, technologies, and security services, such as the new Security Service Provider Interfaces (SSPI) introduced in WebLogic 7.

THE QUADRASIS EXAMPLE

The Quadrasis EASI Security Unifier is an example of such a framework. It provides developers with a flexible framework and tools for unifying security in today's distributed application environments. Quadrasis provides EASI adapters coded to use the SSPI for WebLogic Server 7 enabling protection for a broad set of WebLogic resources, including JNDI resources, Web resources, JMS resources, and JDBC resources. An Administrative Console enables centralized management of adapters, mappers, and security policies for the entire distributed security infrastructure. Quadrasis is committed to extending the functionality of the EASI framework, expanding its repertoire of ready-made adapters and mappers and encouraging third-party development. For more information about Quadrasis or the EASI Security Unifier, please visit www.quadrasis.com.

ReportMill

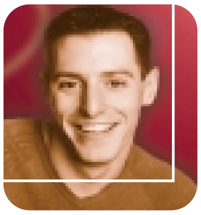
www.reportmill.com

Recently I made a long trip to the East Coast. While there, I was able to meet with a number of developers, customers, and partners. I spoke to a variety of people and heard about a number of interesting community goings-on.

Which Integration Approach is Best?

J2EE CA OR WEB SERVICES? SOMETIMES BOTH!

BY **TYLER JEWELL**



AUTHOR BIO...

Tyler Jewell is BEA's director of technology evangelism. He's the author of *Java Web Services* and the coauthor of *Mastering Enterprise JavaBeans 2.0* and *Professional Java Server Programming (J2EE 1.3)*. He writes widely on the Web on the subject of both J2EE and Web services.

CONTACT...

tyler@bea.com

Lately, I've been on a big Web services kick. I've spent a significant amount of time studying, speaking, and writing about this technology. While speaking to a number of different users groups in the area, I was surprised to see one question bubbling to the surface repeatedly.

People kept asking about the difference between the J2EE Connector Architecture (JCA) and Web services. They were interested in knowing what the criteria should be for selecting one over the other. At first, I was completely perplexed because the answer seemed obvious. But, after giving it some thought, I can easily understand why confusion has started to seep in. Vendors have done such a good job of marketing Web services as standardized integration that companies and individuals have started to voice angst because JCA is also pitched similarly. It naturally leads to the conclusion that these technologies compete and conflict, right? Well, not exactly.

Understanding Positioning

Web services and JCA are both standards that help with integration problems, but they certainly don't compete with one another. It's important that developers and companies understand their differences and what makes the technologies different.

Web services is positioned as a technology that standardizes integration. The JCA specification is designed to standardize enterprise application integration (EAI), which is a subset of the broader integration problem. Therefore, asking the question, "If I have an EAI problem, should I use Web services, JCA, or both?" is completely valid.

The Biggest Difference Is Intrusion

Intrusion is the amount of modification to the legacy system required to support the integration technology.

- ***JCA:*** This technology requires little or no intrusion into the legacy system. JCA adapters gain access to the legacy system by using native APIs, sockets, data access, and a number of other technologies that don't require modification of any existing code bases to be effective. This allows a company to leverage the investment already made in a system in deployment as opposed to investing additional resources into updating the system to support integration.
- ***Web services:*** In order for a legacy system to natively support using Web services as an EAI approach, a high level of intrusion would have to occur. The legacy system would have to be updated and modified to contain a Web services stack. This stack would handle all SOAP message parsing and legacy system invocations – quite a big task. Most enterprises would balk at the idea of modifying an existing COBOL or older mainframe application to introduce an unproven Web services stack.

As a result of this scenario, if a company wants to use Web services for EAI, the most likely strategy would be to purchase a Web services gateway that would intercept Web services messages, do a native invocation of a legacy system, get the response, and convert it to a response Web service message. JCA could easily be used as the technology that does the native legacy system invocation, but many companies might object to having so many layers of message handling to enable EAI.

As a side note, I believe that this EAI problem will ultimately be a thorn in Microsoft's .NET strategy. Since most legacy systems don't run on Microsoft Windows, it's impossible for .NET to do direct legacy system integration. Microsoft has openly declared that Web services are their strat-

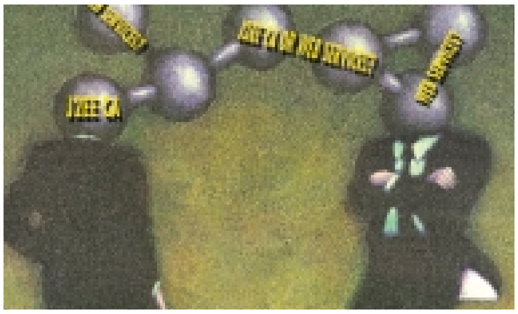
BEA eWorld
Europe
www.bea-eworld.com

egy for EAI in the future. This means that Microsoft is depending upon companies to embed Web services stacks into legacy systems to gain access to them. I would argue that enterprises will balk at the idea and not make this investment. Those enterprises that commit to .NET will be forced to purchase an EAI server (WebLogic Server is a good choice) to intercept Web services requests and then use an EAI-specific technology such as JCA to do the integration.

Context Propagation

A serious infrastructure issue to consider is transaction and security context propagation from clients consuming a legacy system to the legacy system itself.

- ***JCA:*** JCA supports security and transaction context propagation from the JCA server to the legacy system. This is one of the fundamental infrastructure concepts incorporated into the design of JCA and a strong reason it's used in many integration scenarios. This means that if an XA transaction context or a security context identifying a user is created in the application server, when the application server invokes a service on the legacy system, the system is infected with the transaction and security context. In terms of XA transactions, this



  means that if the legacy system is an XA-compliant resource manager, it can participate in an XA transaction. This allows architects to design EAI systems that can ensure ACID transactions with the legacy system and the client consuming it. In terms of security, this means that a single sign-on identifier for an individual can be used between the client and the legacy system. These are very critical characteristics to consider when designing interaction with a legacy system.
- ***Web services***: It's conceivable that a Web service message could propagate a transaction or security context as part of a SOAP header extension, but there is no standard for doing that today. Additionally, even if such a standard existed, it would not be used to propagate XA transactions and internal security user identifiers. Web services decouple clients from servers much the same way MOM systems decouple producers and consumers. Yes, it is possible to propagate an XA transaction context across a Web service message, but the very nature of Web services means that there is no predictability as to when response messages will be sent or arrive at their destination. XA transactions are very short-lived transactions, usually with an automatic transaction rollback occurring after 30 seconds. The unpredictable and disconnected nature of Web services makes them an unrealistic transport for XA transaction contexts. A similar argument can be made for system-level security identifiers. This means that a client using Web services to access a remote system cannot do so using XA transaction semantics.

Don't be discouraged by Web services, however. Web services are intended to expose a business-level interface to a piece of business logic, not a fine-grained method. As such, they do require transactions, security, and a whole slew of other infrastructure technologies, but they must be applied in the context of the environment Web services will operate in (unpredictable response, unpredictable load, applications come and go, XML formats change, etc.). For example, OASIS is working on the Business Transaction Protocol (BTP) to model long-lived transactions for systems in this environment. Additionally, there are a whole slew of security Web services specifications being created. Web services will ultimately change the way developers look at business services and the architecture they use to build them, but you will have to decide if those new approaches are appropriate for EAI.

Code Binding

Code binding is a simple, yet important, aspect of EAI. A comparison of JCA and Web services can be made here, too.

- ***JCA:*** JCA is a Java-based technology. This does not mean that the legacy system has to be authored in Java, but it does mean that the consumer of a legacy application using JCA must be written in Java. This has certain implications for architects. First, any application that uses JCA is bound to its use of JCA. This means that if the application's use of the legacy system needs to change, a developer must physically rewrite the JCA access code to make this change. This may or may not be problematic for some companies. Second, applications can take advantage of strong typing and compile-time checking for clients that are accessing a legacy system. Since the data types and methods used to access the legacy system are exposed as Java types, a Java application that uses JCA can have compile-time checking against semantic usage of the legacy system (it

Sitraka

www.sitraka.com/jclass/wldj

doesn't validate appropriate usage of the system, just that the syntax of using the system is accurate).

- ***Web services:*** A Web service EAI approach would allow a client authored in any language to access the legacy system, whether a Java, C#, Fortran, or COBOL client. Also, since Web services use XML messages to communicate with other systems, a client that dynamically connects to a Web service and obtains a WSDL document to understand message structure doesn't get any benefits associated with compile-time checking. However, the Web service's format can evolve in real time and the application can dynamically evolve with it – i.e., changing the Web service doesn't necessarily mean rewriting the client using the Web service.

Data Binding

Data binding deals with changing the format of the legacy system data to a format that is consumable by the client application.

- ***JCA:*** JCA requires a binding of the data into a Java data type (Java class). This is simple and straightforward for simple data types where there is a clear mapping between the native type and the Java type, but quickly gets sticky when the legacy system has complex relational or binary data that has to be manipulated. JCA 1.0 didn't have a standard way to represent the format of data in the legacy system; dynamic clients couldn't connect to a JCA adapter and "discover" what the data format of the legacy system was. JCA 2.0 will contain a metadata facility that will allow JCA clients to query a JCA adapter to dynamically discover the shape and format of data in the legacy system.
- ***Web services:*** Web services poses an interesting problem for legacy systems, as the data in the legacy system must first be translated into XML for transport and then into the programming language format of the client consuming the Web service. If the EAI system uses the client –> Web service –> server middleware –> native integration adapter –> legacy system approach, the data in the legacy system goes through an ugly native format –> language format of native integration adapter translation –> XML translation –> language format of the client consuming the Web service. Unless data-binding techniques drastically improve, it's possible that a lot of the meaning of the original data could be lost in these translations (not to mention the blatant performance slowdown).

On the other hand, there could be value for many companies to represent legacy data in XML format. XML has semantic business meaning that can be leveraged across applications and systems in a number of different ways. Also, if the legacy system natively exposes its data in XML, Web services might provide a seamless approach that may make using JCA too cumbersome. There are many systems built in the past three years that are now legacy systems using XML as the data format, so this approach may be feasible.

"Vendors have done such a good job of marketing Web services as standardized integration that companies and individuals rightly have started to voice angst because JCA is also pitched similarly.

Conclusion

I hope this article has provided a rapid introduction to the major architectural differences between JCA and Web services for the purpose of doing EAI. We're always interested in hearing about additional positioning arguments and further comparisons. If you have something to add, let us hear about it! Send me an e-mail and let's discuss it.

Missed an issue?

We've got 'em all for you on CD!

JAVA DEVELOPER'S JOURNAL

The most complete library of exclusive *JDJ* articles on one CD!

Check out over 500 articles covering topics such as...
Java Fundamentals, Advanced Java, Object Orientation, Java Applets, AWT, Swing, Threads, JavaBeans, Java & Databases, Security, Client/Server, Java Servlets, Server Side, Enterprise Java, Java Native Interface, CORBA, Libraries, Embedded Java, XML, Wireless, IDEs, and much more!

JDJ The Complete Works
~~Reg $119.99~~
Buy Online
Only $71.99

WebServices JOURNAL · XML JOURNAL

The most complete library of exclusive *WSJ* & *XML-J* articles on one CD!

Check out over 400 articles covering topics such as...
XML in Transit, XML Industry Insider, UML, Integration, WSDL, VoiceXML, XML & E-Commerce, Beginning Web Services, Web Services Tips & Techniques, Ubiquitous Computing, Information Management, XML Script, UDDI, .NET, Objects & XML, XML Middleware, and much more!

Web Services Resource CD
~~Reg $119~~
Buy Online
Only $79

COLDFUSION Developer's Journal

The most complete library of exclusive *CFDJ* articles!

Check out over 250 articles covering topics such as...
Custom Tags, ColdFusion and Java, Finding a Web Host, Conference Reports, Server Stability, Site Performance, SYS-CON Radio, ColdFusion Tips and Techniques, Using XML and XSLT with ColdFusion, Fusebox, Building E-Business Apps, Application Frameworks, Error Handling, and more!

SPECIAL OFFER: Buy CFDJ & CFA The Complete Works For Only $129.99

CFDJ The Complete Works
~~Reg $79.99~~
Buy Online
Only $71.99

CF Advisor

The most complete library of exclusive *CFA* articles!

Check out over 200 articles covering topics such as...
E-Commerce, Interviews, Custom Tags, Fusebox, Editorials, Databases, News, CF & Java, CFBasics, Reviews, Scalability, Enterprise CF, CF Applications, CF Tips & Techniques, Programming Techniques, Forms, Object-Oriented CF, WDDX, Upgrading CF, Programming Tips, Wireless, Verity, Source Code, and more!

CFA The Complete Works
~~Reg $79.99~~
Buy Online
Only $71.99

JDJStore.com

Order Online and Save 10% or More!

www.JDJSTORE.com

OFFER SUBJECT TO CHANGE WITHOUT NOTICE

COLLECT ALL 4 FOR ONLY $229.99
JDJ, WSJ & XML-J, CFDJ, & CFA

WebServices JOURNAL .NET J2EE XML

XML JOURNAL

THE FIRST & ONLY WEB SERVICES RESOURCE CD !

WEB SERVICES RESOURCE CD

THE SECRETS OF THE WEB SERVICES MASTERS

INCLUDES EXCLUSIVE .NET ARTICLES

MORE THAN 400 EXCLUSIVE WEB SERVICES & XML ARTICLES

EDITED BY SEAN RHODY

$119 CD VALUE FROM WEB SERVICES JOURNAL
web services EDGE conference&expo $100 coupon inside

EVERY ISSUE OF WSJ & XML-J EVER PUBLISHED

THE MOST COMPLETE LIBRARY OF EXCLUSIVE WSJ & XML-J ARTICLES ON ONE CD!

"The Secrets of the Web Services Masters"

CD is edited by well-known WSJ Editor-in-Chief Sean Rhody and organized into more than 40 chapters containing more than 400 exclusive WSJ & XML-J articles.

Easy-to-navigate HTML format!

Bonus:

Full .PDF versions of every WSJ & XML-J published since the first issue

XML in Transit
XML B2B
Java & XML
The XML Files
XML & WML
Voice XML
SYS-CON Radio
XML & XSLT
XML & XSL
XML & XHTML
2B or Not 2B
XML Script

XML Industry Insider
<e-BizML>
XML & Business
XML Demystified
XML & E-Commerce
XML Middleware
XML Modeling
CORBA & XML
Data Transition
XML @ Work

XML & Databases
Electronic Data Interchange
Ubiquitous Computing
Information Management
Objects & XML
XML Pros & Cons
Java Servlets
XML Filter

UML Integration
WSDL
Beginning Web Services
Web Services Tips & Techiques
Frameworks
Management
Security
UDDI
.NET

3 YEARS
25 ISSUES
400 ARTICLES
ONE CD

WebServices XML Journal
THE FIRST & ONLY WEB SERVICES RESOURCE CD!
WEB SERVICES RESOURCE CD
THE SECRETS OF THE WEB SERVICES MASTERS
INCLUDES EXCLUSIVE .NET ARTICLES
MORE THAN 400 EXCLUSIVE WEB SERVICES & XML ARTICLES
$119 CD VALUE
EVERY ISSUE OF WSJ & XML-J EVER PUBLISHED

Special Limited Time Price

Now Shipping

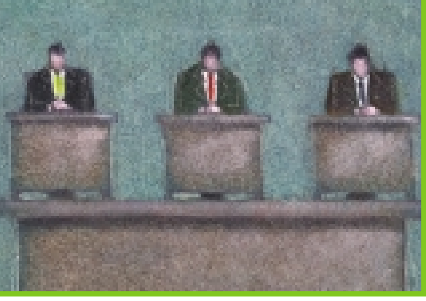
$79 + S&H

ONLINE ORDER AT JDJSTORE.COM

SAVE $40

www.JDJSTORE.com

OFFER EXPIRES JUNE 30, 2002



Last month, I talked about signing up to take the BEA WebLogic Server 6.0 Certification test. At that time, I promised I would go into more detail about what you should study, and that's what I'm including for your use.

JNDI and Web Applications on the WLS 6.0 Certification Test

ONE THAT'S SIMPLE; ONE THAT NEEDS MORE THOUGHT

BY DAVE COOKE

AUTHOR BIO

David Cooke is an experienced software developer currently working for Ness Technologies, Inc. (www.ness-usa.com), a consulting firm located in Dulles, VA. In his current position, he utilizes Java and BEA WebLogic Server 6.0 to build J2EE-compliant e-commerce systems for a variety of clients. Dave maintains Microsoft, Java, and BEA developer certifications.

CONTACT...

dave.cooke@ness-usa.com

This article covers two specific topics on the test: the Java Naming and Directory Interface (JNDI) and Web applications.

But before we start, there's one thing I forgot to mention last month: another benefit of the study material is that there is a strong overlap of these topics with the Sun Web Component Developer for J2EE test. So, by studying for this test, you'll be studying for the Web Component Developer test as well.

Now I'm assuming most of you are fairly experienced J2EE developers, so I'll gloss over most of the material, just pointing out the important aspects of the test. I've also provided a set of review questions at the end of this article that will help readers of all skill levels prepare for the test. Let's get started.

The Java Naming and Directory Interface

There isn't too much direct coverage of JNDI on the test, but using JNDI is important and, therefore, requires a little attention. The primary thing to be concerned about is configuring an InitialContxt and knowing the java:comp naming convention to look up objects.

When you're configuring an InitialContext, an Environment object (or HashTable) is used to initialize the InitialContext. The Environment contains several properties that assist in configuring the InitialContext object. The property names, constants, and default values are shown in Table 1.

In addition to the normal naming convention used by WebLogic Server JNDI, the J2EE specification has defined a standard naming convention for the objects stored in a JNDI tree. You can read more about this naming convention at http://java.sun.com/products/jndi/tutorial/beyond/misc/policy.html, but Table 2 summarizes the important names to know.

That's more or less all you need to know about JNDI for the test. I would recommend having a good general knowledge of JNDI though, since it is used in some of the code in the test questions.

Web Applications

Now that I've covered a minor test topic, it's time to look at something a little more substantial. Web applications is one of the core topics of the BEA WebLogic Server 6.0 test. A good understanding of Web applications is extremely important to passing. Apart from the topic of Enterprise JavaBeans (EJBs), this is the most widely covered topic of the test, so be sure to study this area carefully. To make it easier, I'm going to break the Web applications section down into three areas: servlets, Java Server Pages (JSPs), and deployment.

When you prepare for the HTTP servlets portion, be sure to know the basics of creating an HTTP servlet. You should be familiar with servlet methods, how they are used, and when they are

TABLE 1

CONSTANT	PROPERTY NAME	DEFAULT VALUE
INITIAL_CONTEXT_FACTORY	java.naming.factory.initial	weblogic.jndi.WLInitialContextFactory
PROVIDER_URL	java.naming.provider.url	t3://localhost:7001
SECURITY_PRINCIPAL	java.naming.security.principal	guest
SECURITY_CREDENTIALS	java.naming.security.credentials	guest

Property names, constants, and default values

Special Introductory Offer SAVE $13.89*

WebServices JOURNAL

The world's leading independent Web Services information resource

Get Up to Speed with the Fourth Wave in Software Development

- 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
- and much, much more!

Only $69.99 for 1 year (12 issues)
Newsstand price $83.88 for 1 year

BETTING YOUR FUTURE: .NET OR J2EE?

WebServices JOURNAL

.NET J2EE XML

WSJ2.COM

.NET or J2EE

STREET FIGHTING COMES TO WEB SERVICES

The Week That Was! Web Services Goes Mainstream!

.NET Feature: Leveraging Web Services The Web Services that will change the face of computing — Dave Howard 6

Focus Discovery: UDDI and WSDL: Natural Companions Enabling a software landscape based on Web services — Karsten Januszewski 16

Technology: Exploding Web Services Myths What we can really expect to do with today's technologies — Andy Longshaw 20

Focus Discovery: Second-Generation Portals Poised to deliver the next generation of services-on-demand — Hal Stern 34

Focus Discovery: Web Services & Distributed Objects: Competing or Complimentary Integrating both roles — Alan Conway 40

.NET Feature: Setting the Standards Technology and tools to formalize how business applications communicate — Hitesh Seth 42

Focus Discovery: An Introduction to WSDL A critical part of an automated Web based on services — Mike Richardson

Feature: Web Services & Distributed Component Platforms

www.wsj2.com

SYS-CON MEDIA

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

**Offer subject to change without notice*

TABLE 2

NAME	STORES
java:comp/UserTransaction	UserTransaction object
java:comp/env/ejb/	EJBs
java:comp/env/jdbc/	JDBC DataSources
java:comp/env/jms/	JMS connection factories
java:comp/env/mail/	JavaMail connection factories
java:comp/env/url/	URL connection factories

Naming conventions

TABLE 3

METHOD	DESCRIPTION
init()	Called when the servlet is initialized
destroy()	Called when the servlet is destroyed
service(HttpServletRequest, HttpServletResponse)	Handles all HTTP requests
doGet(HttpServletRequest, HttpServletResponse)	Handles a HTTP GET request
doPost(HttpServletRequest, HttpServletResponse)	Handles a HTTP POST request

Servlet methods

TABLE 4

TAG NAME	DESCRIPTION
<% scriptlet %>	Method-level Java code.
<%! declaration %>	Class-level Java code. This code should be synchronized for thread safety.
<%= expression %>	Returns the result of a single Java expression to the client; expression must evaluate to a string.
<%@ directive %>	An instruction to the JSP compiler, which effects page compilation.
<jsp:action/>	Short-cut functions of the default taglib.

JSP tags

TABLE 5

BUILT-IN OBJECT	JAVA CLASS
request	HttpServletRequest
response	HttpServletResponse
out	JspWriter
session	HttpSession
config	ServletConfig
page	Current servlet class
pageContext	PageContext
application	ServletContext
exception	Exception

Built-in objects

TABLE 6

DIRECTIVE NAME	DESCRIPTION
page	Controls attributes of the JSP page
	<%@ page attribute %>
include	Textually includes a file into the JSP page
	<%@ include file %>
taglib	References a tag library to be used in the JSP page
	<%@ taglib uri prefix %>

Directives

called. Table 3 illustrates the methods that are important to understand for the test.

Be sure to refamiliarize yourself with the HttpServletRequest class and its methods, such as getParameter(), which retrieves parameters from the request. In addition, study the HttpServlet-Response and its methods, such as setContentType(), which is used to inform the client (browser) of which type of data is being sent back.

The HttpSession class represents the user's session. It is retrieved by the getSession() method of HttpServlet. Objects may be stored and retrieved from the HttpSession by using the putValue() and getValue() methods, respectively.

WebLogic Server exposes information about the servlet container through the ServletContext object. You can obtain the Servlet-Context by the getServletContext() method of ServletConfig, which is passed into your servlet's init() method. In addition, the ServletContext can be used to store data on the Web application's scope by using the getAttribute() and setAttribute() methods. It's important to note that, unlike data stored in the HttpSession object, data stored in the ServletContext may be used by all of the users of the Web application.

JAVA SERVLET PAGES

The next subtopic of Web applications is JSPs. Be sure to know the basics of creating a JSP and how WebLogic Server turns your page into a servlet. Also, be very familiar with the JSP tags used in creating a page (see Table 4). Below, I'll discuss each tag and what you should know about it.

SCRIPTLET

Understanding that a scriptlet is just a piece of Java code that will be placed in the service() method of the generated servlet is critical. The code used in a scriplet has access to built-in objects (see Table 5).

DECLARATION

A declaration is very similar to a scriptlet, but its code is placed at the *class* level of the generated servlet. Because of this, you can use declarations for defining methods or class-level variables. Since this code may be accessed by many users it should be synchronized for thread safety. Declarations have access to the same built-in objects that scriptlets do.

EXPRESSION

Not much to know here, but you should be aware that an expression returns the result of a single Java expression to the client. Also, realize that the expression must evaluate to a String.

DIRECTIVE

A directive is an instruction to the JSP compiler that affects page compilation. An important fact to remember about directives is that they affect how the page is compiled, not how the page executes. Table 6 shows a list of directives. I'll break each of them down below.

PAGE DIRECTIVE

Table 7 summarizes some possible values for a page directive.

INCLUDE DIRECTIVE

The include directive textually includes the contents of a file into the JSP. It is important to distinguish between the include directive and the include action (see below).

TAGLIB DIRECTIVE

The taglib directive specifies the location and usage of a tag library. There isn't much coverage of tag libraries on the WebLogic Server 6.0 test, but you may want to take some extra time out to review them if you are also looking to take the Sun Web Component Developer for J2EE test.

ACTION

Generically, an action is a reference from a tag library, but for the purposes of the test, be sure to know the standard set of actions JSP provides (see Table 8).

USEBEAN ACTION

The useBean action accesses a JavaBean object. It is not a valid way to use or create an Entity Java Bean. If the referenced JavaBean doesn't exist, it will be created. The useBean action specifies the scope at which the JavaBean exists (see Table 9).

To retrieve a bean programmatically from a different scope, the getAttribute(id) method of the proper scope object.

INCLUDE ACTION

Remember that the include action is different from the include directive. The include action delegates processing from one page to another by having the initial JSP page accept the request, call an included JSP page to perform some function, and then send the response back to the client.

FORWARD ACTION

Similar to the include action, the forward action also delegates processing from one page to another. However, the forward action works by having the initial JSP page accept the request and forward it to the next page. The last JSP page in the "chain" of pages sends the response back to the client. Multiple forwards may occur, but only the last page is allowed to send a response to the client.

DEPLOYMENTS

We're just about finished with the Web applications section. The last topic to cover here is deployment. Web application deployment is a mixed bag of information.

First, you should be familiar with the Web application directory structure and what each directory is used for (see Table 10).

There is quite a bit on deployment descriptors on the test, but most of the coverage relates to EJB. You should, however, be familiar enough with the descriptors to deploy a servlet and a JSP page. These go in the web.xml file. Examples of servlet and JSP deployment descriptors are shown in Listings 1 and 2.

Along with the deployment information, the web.xml configuration file also defines some security attributes. These security attributes specify how WebLogic Server handles security in a Web application (see Table 11).

And finally, although this really isn't deployment-related, it merits mention: you should know that WebLogic Server provides several servlets to handle incoming requests (see Table 12).

Review

As promised, I've prepared a short study test for you. The answers are provided at the end of the test questions. I'd advise taking the whole test at once and then checking your answers;

TABLE 7

PAGE DIRECTIVE ATTRIBUTE	DESCRIPTION
session	Indicates whether built-in session object is available
errorPage	Specifies an URL to handle all unprocessed exceptions
isErrorPage	Indicates the built-in exception object is available
import	Specifies what packages should be imported
contentType	Sets the MIME type for the response

Values for a page directive

TABLE 8

ACTION	DESCRIPTION
useBean	Uses a JavaBean object. Created if it does not exist.
setProperty	Sets the value of a property of a JavaBean object.
getProperty	Gets the value of a property run time of a JavaBean object.
include	Includes a JSP page at run-time. See the Delegation Models section. Not to be confused with the include directive.
forward	Forwards to a JSP page at runtime.

JSP actions

TABLE 9

SCOPE	STORED IN OBJECT	LIFETIME
page (default)	PageContext	JSP page
request	ServletRequest	browser request
session	HttpSession	user session
application	ServletContext	application server

Scope of the useBean action

TABLE 10

FILE SYSTEM PATH	DESCRIPTION
WebAppName root	Static files (JSP, HTML, Images)
/WEB-INF/web.xml	Deployment descriptors
/WEB-INF/weblogic.xml	WebLogic-specific deployment descriptors
/WEB-INF/classes	Server -side classes such as Servlets/utility
/WEB-INF/lib	.jar files used by Web app

Directory structure

TABLE 11

TAG NAME	DESCRIPTION
<login-config>	Specifies how the user is prompted to login
<security-constraint>	Used to define access privileges to a collection of resources by their URL mapping
<security-role>	Represents a group or principal in the realm
<security-role-ref>	Links a role name to a <security-constraint>

Security-related descriptors

TABLE 12

SERVLET	DESCRIPTION
FileServlet	Handles most client requests, such as HTML pages, images' and other static content. It is WebLogic Server 6.0's default servlet.
JspServlet	Handles processing of JSP pages
CgiServlet	Allows WebLogic to run legacy CGI applications
ServletServlet	Allows WebLogic to run any servlet in the classpath without regard for security. ServletServlet should only be used for development purposes.

Servlets for incoming requests

this way, you'll see how well you'd have done on the real test.

Next time, we'll cover JDBC and EJB. Have fun studying!

1. What property of the InitialContext Environment affects how the InitialContext is created?
   a) weblogic.jndi.WLInitialContextFactory
   b) java.naming.factory.initial
   c) java.naming.provider.url
   d) java.naming.security.principal

2. What method of an InitialContext do you call to retrieve an object from the JNDI tree?
   a) lookup()
   b) find()
   c) bind()
   d) get()

3. Which servlet allows you to execute any other servlet in the classpath?
   a) FileServlet
   b) JspServlet
   c) ServletServlet
   d) CgiServlet

4. What method of the response should you call before you send any content back to the browser?
   a) setMIMEType()
   b) setContentType()
   c) setResponse()
   d) setContent()

5. What class can be used by a Servlet to store attributes at Web application scope?
   a) AppContext
   b) ServerScope
   c) SessionContext
   d) ServletContext

6. Which JSP tag includes a JSP page for executing at run-time?
   a) <%@ include file="somepage.jsp" %>
   b) <jsp:include page="somepage.jsp">
   c) <jsp:forward page="somepage.jsp">
   d) None of the above

7. If a JSP page has the isErrorPage directive set to true, which of the following is true?
   a) An intrinsic exception object is defined
   b) All exceptions raised by any page in the Web application are redirected to this page
   c) This page is displayed when the generateError() method is called
   d) All of the above

8. Select a valid way to call an EJB from a JSP page:
   a) Utilize a standard JSP directive
   b) EJB Creation code in scriptlet
   c) <jsp:useBean> action
   d) All of the above

9. Which deployment descriptor contains the url-pattern used to map a servlet or a JSP to a URL?
   a) <servlet>
   b) <servlet-name>
   c) <servlet-mapping>
   d) <servlet-url-pattern>

10. Which Web application directory houses the deployment descriptor file weblogic.xml?
   a) The Web application root
   b) META-INF
   c) WEB-INF
   d) There is no deployment descriptor file named weblogic.xml.

Answer Key

1. b 2. a 3. c 4. b 5. d 6. b 7. a 8. b 9. c 10. c

Listing 1: Servlet deployment descriptor

```
<servlet>
  <servlet-name>ServletA</servlet-name>
  <servlet-class>Aservlet</ servlet-class>
</servlet>

<servlet-mapping>
  <servlet-name>ServletA<servlet-name>
  <url-pattern>/serv/</url-pattern>
</servlet-mapping>
```

Listing 2: JSP deployment descriptor

```
<servlet>
  <servlet-name>JSPFileA</servlet-name>
  <jsp-file>AJSPFile.jsp</jsp-file>
</servlet>

<servlet-mapping>
  <servlet-name> JSPFileA <servlet-name>
  <url-pattern>/jsp/</url-pattern>
</servlet-mapping>
```

JAVA: INDUSTRY NEWSLETTER
Reply
Reply All
Forward
Flag
Print
Email Newsletters
JAVA DEVELOPERS JOURNAL Industry > Newsletter

Monday, December 10, 2001

BROUGHT TO YOU BY...

aligo Wireless Development Strategies - A Free White Paper. How can you use J2EE/J2ME to successfully deploy Wireless Applications? Aligo, the leader in Java based wireless application servers, offers a free technical paper on Java and Wireless Development. Click Here! for more information.

Newsletter
JDJ News
JDJ List
Subscribe Now!
Special Offer!
Privacy Notice

JAVA DUDES

JDJ NEWS DESK EXCLUSIVE

Oracle Targets Top Spot With New Oracle9iAS and JDeveloper
by Rick Ross - Founder of Javalobby.org

(December 10, 2001) – I returned just yesterday from a fantastic trip to San Francisco for Oracle's OpenWorld conference. It wasn't clear what to expect when I accepted Oracle's invitation to give a presentation, but I was delighted to find the show had a high-energy crowd that was very tuned in to Java, especially J2EE and Web services. The whole show was great, but the most interesting and important items came from Oracle itself - Oracle9i Application Server Release 2 and Oracle9i JDeveloper.

WAKESOFT
Learn more now!
AltoWeb Release 2.5
WEB SERVICES: INDUSTRY NEWSLETTER
WebServices JOURNAL Industry >Newsletter

Monday, December 3, 2001

Stay Tuned for WSJ's December Issue!
Full Editorial Content Coming December 5th

BROUGHT TO YOU BY...

ORACLE Catch the Oracle9i JDeveloper wave and see how built-in profilers and Code Coach makes your Java code tighter and faster than ever before. Based on the latest J2EE and XML standards, JDeveloper combines Java coding, UML modeling, and RAD on NT/Linux/Unix. Click Here! for more information.

Newsletter
WSJ News
Next Issue
Subscribe Now!
Privacy Notice

ONLINE EXCLUSIVE: FULL "KEYNOTE PANEL" TRANSCRIPT

Watch Keynote Panel on SYS-CON.TV

Sean Rhody: Probably the question that is on almost everybody's mind is, what are Web services? I know I have a whole magazine devoted to answering that question, but I think one of the most interesting parts of the magazine is that the answers are different for different people and from different people. So to start things off I'll go down and ask each of the panelists that question. In your mind, Barry, what are Web services?

Barry Morris: I would think we all kind

Tuesday, December 18, 2001

BROUGHT TO YOU BY...

flamenco Secure Web services. Reliable Web services. Manageable Web services. If you need Web services then you need a Web services network. Security, guaranteed delivery, and management services with no code changes. Plug into the Flamenco Network and get going! For a limited time, bring your Web services and test-drive the Flamenco Network for free.
http://www.sys-con.com/email/flamenco1114.cfm

Newsletter
XML News
Subscribe Now!
Special Offer!
Privacy Notice

XML NEWS

Xerces-C++ XML Parser Version 1.6.0 Provides Full Support for W3C XML Schema Recommendation

(Toronto) - The XML Parsers Development division of the IBM Toronto Laboratory has announced the much-anticipated release of the Apache Xerces-C++ XML parser version 1.6.0, including full support for the W3C XML Schema Recommendation version 1.0. Xerces-C++ is a validating XML parser written in a portable subset of C++; Xerces-C++ makes it easy to give applications the ability to read and write XML data. A shared library is provided for parsing, generating, manipulating, and validating XML documents. features: conformance to the XML Specification 1.0; tracking of [Level 1.0], DOM [Level 2.0], SAX/SAX2, namespace, and XML source code, samples, and documentation; and validation of XML; pluggable catalogs, validators performance; and customizable error handling. The latest C++ includes a port to FreeBSD and allows users to through a method call. Source code and binaries Linux, Solaris, and Windows.

high performance, modularity, and scalability. Source are provided with the parser. For to make minimal use of templates, no RTTI, minimal use of #ifdefs.

THE INSIDER INTELLIGENCE YOU NEED...

TO KEEP AHEAD OF THE CURVE

FREE E-Newsletters

SIGN UP TODAY!

Go to www.SYS-CON.com

The most innovative products, new releases, interviews, industry developments, and plenty of solid *i*-technology news can be found in SYS-CON Media's Industry Newsletters. Targeted to meet your professional needs, each e-mail is informative, insightful, and to the point. They're free, and your subscription is just a mouse-click away at www.sys-con.com.

SELECT THE INDUSTRY NEWSLETTERS THAT MATCH YOUR NEEDS!
CHOOSE ONE – OR TRY THEM ALL!

JAVA DEVELOPERS JOURNAL Industry > Newsletter	WebLogic Industry > Newsletter
WebServices Industry >Newsletter	WebSphere Industry > Newsletter
XML JOURNAL Industry > Newsletter	COLDFUSION Developer's Journal Industry > Newsletter
wireless Industry > Newsletter	LINUX BUSINESS WEEK Industry > Newsletter

Don't Delay!
Subscribe for FREE!
at www.sys-con.com

Exclusively from the World's Leading *i*-Technology Publisher SYS-CON MEDIA

I've seen several posts in the public WebLogic server transactions newsgroup in which people have had problems with transactions spread across multiple servers.

Transactions, Commitment and Security

AVOIDING THE HEURISTICMIXEDEXCEPTION

BY PETER HOLDITCH

AUTHOR BIO...

Peter Holditch joined BEA as a consultant in the Northern European Professional Services organization in September 1996. He now works as a presales architect in the UK. Peter has a degree in electronic and computer engineering from the University of Birmingham.

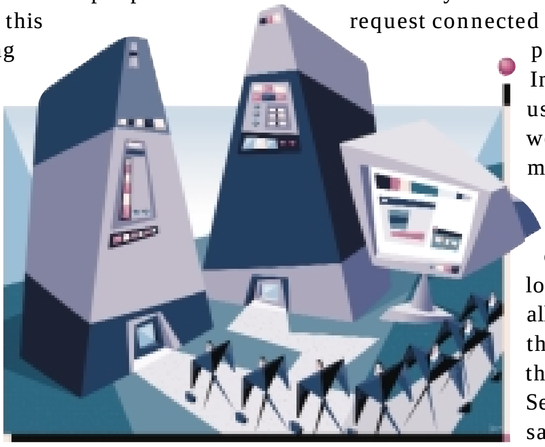
CONTACT...

peter.holditch@bea.com

The gist of these problems is always that they have two EJB components in two different servers. Bean One on Server One executes in the context of a JTA transaction in which it calls Bean Two on Server Two, thus propagating the transaction across to the second server. Finally, the first method ends, and the transaction commits.

When the time to commit comes, the transaction is aborted. Accompanying the abort is a cryptic exception, *javax.transaction.HeuristicMixedException*. Since this comes up repeatedly, I thought I would dedicate this month's column to unravelling what's going on here.

First the euphemism: in the world of transactions, transaction managers don't like to sound too pessimistic, so if they think the world is coming to an end, they don't engage in unnecessary hyperbole, as it might upset the system administrators. Instead, they just quietly say that a *heuristic outcome* has occurred. That means that your carefully orchestrated ACID transaction has just ended with all the data potentially inconsistent, and you'd better go check your databases. Or, to put it another way, the world is at an end!



So with that explanation out of the way, let's think about what's happening here.

The call to Bean 1 has started a transaction that is associated with the thread of control and will propagate through to any database updates and the like that it performs. Then, it calls Bean 2. This will flow a request over to Server 2, where the second bean will be executed. Somehow, the transaction is being flowed over, accompanying the RMI request. Usually, server-to-server communication between WLS instances is done using the T3 protocol, and one of the things that can be achieved using T3 is the attaching of information about transactions to a request. So under the covers, the two WebLogic instances have exchanged notes about the transaction, and now both can participate, and at a later date commit or abort it as a single unit.

Allow Me to Digress...

At this point, a small digression is in order. The latest version of WLS, 7.0, is fully J2EE 1.3–certified, which mandates, among other things, support for EJB 2.0. This in turn mandates application server interoperability using RMI/IIOP. The IIOP protocol, like T3, allows context information such as active transactions to be passed along with method invocations. So, with WLS 7.0, this same scenario could be played out with one of the servers being a WLS instance and the other a different app server (although I don't know why you'd want one – but I'm biased!). Anyway, back to the plot...

Something I haven't mentioned so far in all of this is security. The client that initiated this request connected to Server 1 at some point using an InitialContext with a username and password. When the method on Bean 1 is invoked, the EJB container will check that the logged-in user is allowed to execute the method. Likewise, the EJB container on Server 2 will do the same check before it

Simplex
Knowledge
Company

www.skc.com/J2EE

allows access to the method on Bean 2. This implies that the security ID associated with the request has also flowed over the wire, along with the transaction ID, and indeed it has. This is another capability of the T3 protocol. Incidentally, it's also another requirement of EJB 2.0 container interoperability – a protocol called CSIv2 is used to flow a user identifier over an IIOP connection and this too is supported by WLS 7.0.

The request has flowed from Server 1 to Server 2, along with the transaction and the caller's user ID, and it gets executed there and the result is returned. So far, so good. Now Bean 1 returns to the client and, if the transaction was container-managed, then the container will call commit. It is the job of the transaction co-coordinator (part of the JTA subsystem in Server 1 in this case) to synchronize the prepare and commit phases of this transaction. It will see that there is a participant on Server 2 and send a message to the server instructing it to perform the prepare phase of the two-phase commit protocol. It's at this point that the problems arise. This call isn't being done on behalf of the client, rather it is initiated by the JTA subsystem. What security ID will flow across the wire? In fact, it will be the WebLogic "system" user on whose behalf the prepare phase is executed. It would be a pretty serious security flaw if any Tom, Dick, or Harry could randomly prepare or commit transactions, so the use of the system ID closes this potential security hole. However, we now need to take a look at the security model underpinning WLS 6 (this area is another that has been substantially upgraded in WLS 7). In version 6, two servers will trust each other if they share the same system password. There is no greater degree of trust than trusting the system user, so for the prepare (and subsequent commit) phases to be allowed, the two servers must share a system password.

If they [transaction managers] think the world is coming to an end, they don't engage in unnecessary hyperbole, which might upset the system administrators. Instead, they just quietly say that a *heuristic outcome* has occurred

And Back to the News…

In conclusion, every time this has arisen on the newsgroup (remember, that's where this all started) the solution has been to make sure that all WLS instances that participate in a transaction share a common system password. Also note that transactions can happen that you're not aware of; for instance, if you send a message to a remote JMS queue, a transaction may well be started on your behalf by the JMS subsystem. Remember, next time you see a HeuristicMixedException, check your passwords!

Resources

- *WebLogic server transactions newsgroups:* news://newsgroups.bea.com/weblogic.developer.interest.transaction, or http://newsgroups.bea.com/cgi-bin/dnewsweb?cmd=xover&group=weblogic.developer.interest.transaction&utag=.

web services EDGE conference&expo

JDJ EDGE conference&expo

XML EDGE conference&expo

Jacob K. Javits Convention Center

JACOB K. JAVITS CONVENTION CENTER

New York, NY

CONFERENCE:
June 24-27, 2002

EXPO:
June 25-27, 2002

Gold Sponsor BEA & Web Services Edge 2002 East

PRESENT:

bea®

PARTNER PAVILION

VISIT WITH BEA & THEIR PARTNERS

BEA Corporate: *Booth* **#318**
BEA Partner Pavilion: *Booth* **#312**

MEET AND SPEAK, face-to-face with BEA's Partners.

sitraka | webGAIN | CACHECON

LOOK ON the Web Services Edge 2002 East Special Events page for the entire listing of BEA Partners that will be participating in the BEA Partner Pavilion.

www.sys-con/webservicesedge2002east/specevents.cfm

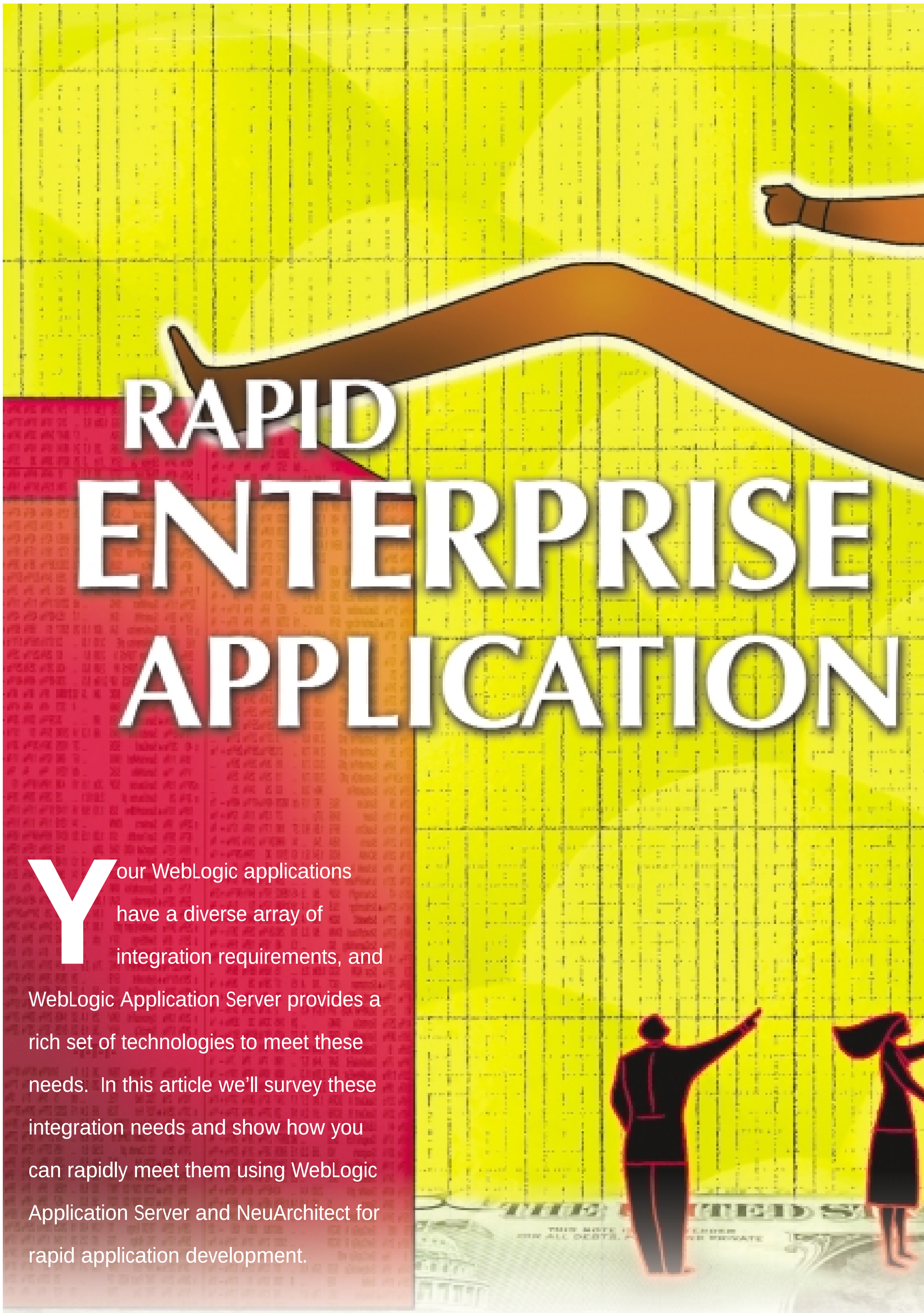
BELOW IS a sampling of participating BEA partners' business focus and expertise.

- Tools/Technology ISV
- Application ISV
- Java/EJB Development Tools
- Application Development
- Application & Integration Frameworks

REGISTER AT www.sys-con.com/webservicesedge2002east/registernew.cfm

CONTACT: Michael Pesick
Regional Sales Manager
SYS-CON Events
1-201-802-3057
michael@sys-con.com

SYS-CON EVENTS
www.sys-con/webservicesedge2002east/



RAPID ENTERPRISE APPLICATION

Your WebLogic applications have a diverse array of integration requirements, and WebLogic Application Server provides a rich set of technologies to meet these needs. In this article we'll survey these integration needs and show how you can rapidly meet them using WebLogic Application Server and NeuArchitect for rapid application development.

BY
ARUN GUPTA & JOSEPH NOONAN

AUTHOR BIOS...

Arun Gupta is CTO and cofounder of NeuVis Software, an innovator in n-tier rapid application development (RAD). He is a 28-year software industry veteran and expert in RAD. His prior company, DataEase Int'l, created the award-winning DataEase RAD product that shipped more than 1 million copies. Arun was also CTO of Home Financial Network, now Sybase Financial Fusion.

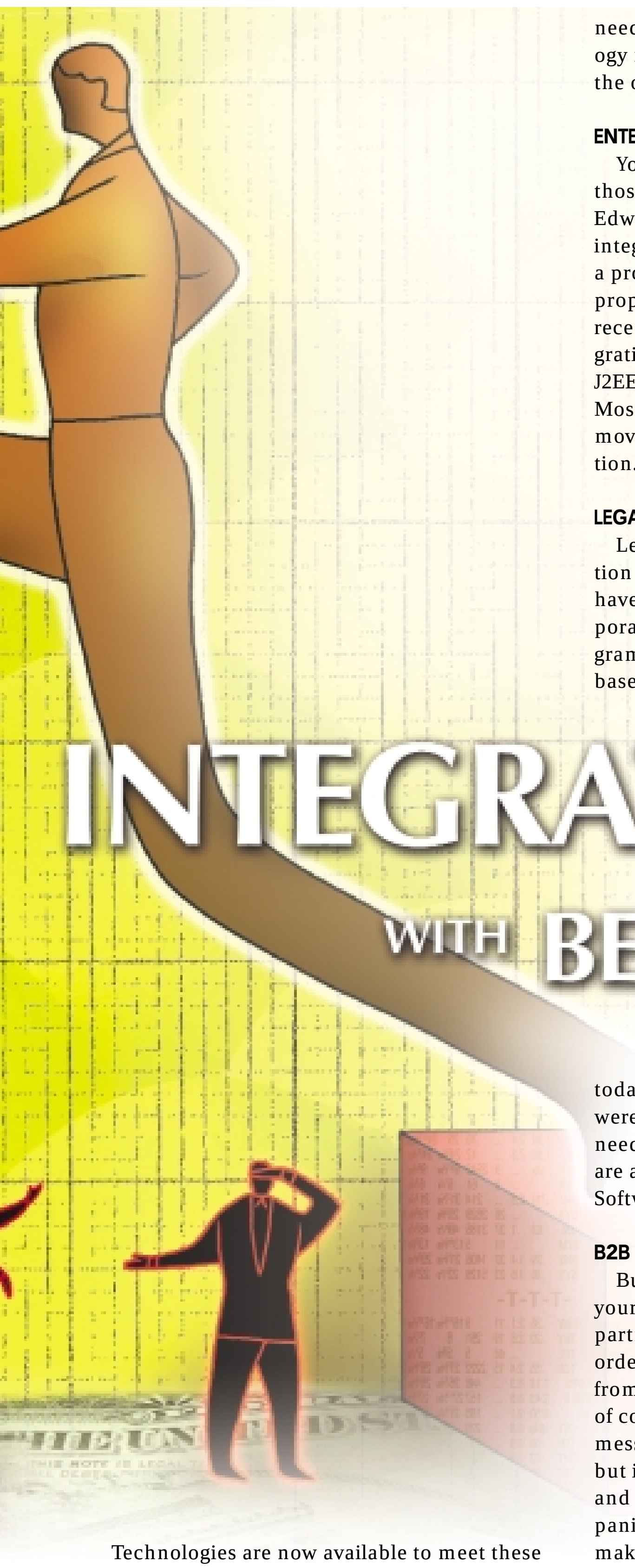
Joseph Noonan has been in software development for 18 years. He is product development manager at NeuVis Software, where he manages the runtime/J2EE construction and system architecture groups focusing on J2EE technologies and integration. Joe has an extensive background in object-oriented analysis and design, UML, and system architectural development. He is coauthor of a Windows 98 API programming book.

CONTACT...

agupta@neuvis.com
jnoonan@neuvis.com

The integration needs of your application may be categorized (see Figure 1) as :

1. Integration with packaged enterprise applications, such as SAP, Siebel, or PeopleSoft
2. Integration with legacy systems, such as logic in a CICS application or data in an IMS database
3. B2B integration with customers and partners
4. Publishing and using components in a component-based development environment



INTEGRATION WITH BEA WEBLOGIC

Technologies are now available to meet these application integration needs. They include messaging using XML and JMS, Web services, many types of connectors – J2EE connectors, legacy connectors, and API connectors. For each need, more than one type of integration technology may be available, and you'll need to select the one most suited to your needs.

ENTERPRISE APPLICATION INTEGRATION

Your packaged enterprise applications, such as those from SAP, PeopleSoft, Siebel, and J. D. Edwards, may provide alternative methods of integration. SAP, for example, originally provided a proprietary API interface called BAPI, and a proprietary messaging system called iDocs. More recently, SAP has offered standards-based integration approaches that include XML messaging, J2EE Connector Architecture, and Web services. Most packaged application vendors are similarly moving toward offering standards-based integration.

LEGACY INTEGRATION

Legacy integration requires that your application be able to interact with older systems that have data and program elements of noncontemporary technologies such as CICS or MVS programs and IMS, VSAM, IDMS, and ADABAS databases. These systems were written well before today's sophisticated integration technologies were available; therefore, legacy connectors are needed to wrap data and logic. These connectors are available from many vendors, including iWay Software and NeuVis.

B2B INTEGRATION

Business-to-business (B2B) integration allows your applications to interact directly with your partners' applications – for example, sending orders directly to your vendors or receiving them from your customers – to provide an efficient way of conducting your business. Originally, EDI messaging was developed to meet these needs, but it proved to be cumbersome and expensive and has been implemented mostly by large companies. Now XML messaging over the Internet is making B2B integration more flexible and less expensive. Web services will also be a promising technology for B2B integration after it has developed appropriate standards and infrastructure for security and reliability.

COMPONENT-BASED DEVELOPMENT

Component-based development allows you to reuse your application components across many applications. Integration at the component level involves developing a common interface that other programs and systems can use to access the features of the component. You have a choice of component technologies: you can publish and use your EJB components as session and entity beans, or you can bridge to COM and CORBA components. Most importantly, Web services promises to be the universal standard for component-based development – that will be implemented on all platforms and technologies.

Integration Technologies and Rapid Integration Development

Application integration needs can be met with a variety of technologies. In the following sections we'll examine key technologies available on WebLogic and see how you can rapidly develop your application integration solution using the WebLogic platform and NeuArchitect, a RAD tool from NeuVis software.

MESSAGING USING XML AND JMS

Messaging is the most flexible and formal means of integrating your application with existing applications. It is suitable only if the applications being integrated have an agreed-upon and supported protocol for messaging. This protocol may be an industry standard, or a custom protocol adopted between the providers of the applications. Messaging is a very popular means of integration within the enterprise and for B2B integration. Messaging involves two technologies – XML as the format of the message and JMS-enabled transports for transporting the messaging between applications (see Figure 2).

XML – LINGUA FRANCA OF INTEGRATION

One of the main problems with early integration efforts was the lack of a standard for sending information between programs and partners. There was EDI, but it required a customized communications infrastructure to support that was also very expensive on a per-transaction basis. What was needed was something that could work (securely) over the Internet and could provide a significantly reduced per-transaction cost. Enter XML.

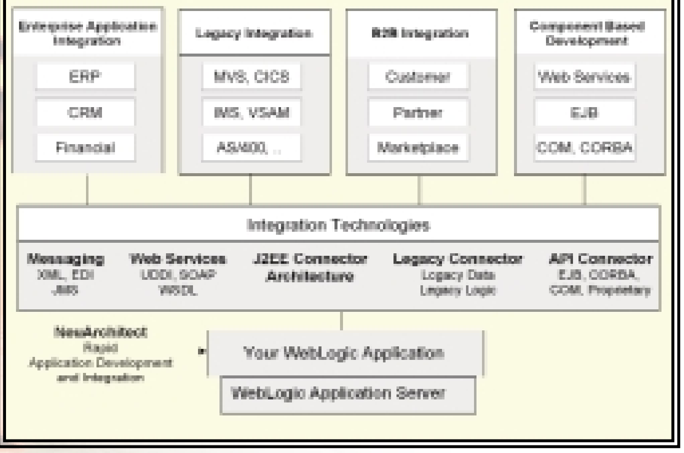
XML has had a profound impact on integration, as well as computing as a whole. It's the basis for other technologies, such as SOAP (Simple Object Access Protocol), the protocol used as part of Web services. XML provides a means of describing both the structure and content of data in a form that is easily exchanged over the Internet. The format is defined either in a DTD document or as an XML schema. These formats are then exchanged between trading partners. Additionally, a variety of industries have developed XML-based standards for describing data, which further eases the task of integration.

JMS MESSAGING

Java Message Service (JMS) is a standard J2EE API for reliable asynchronous messaging between applications. Many vendors, including BEA, have implemented this API. BEA's offering is composed of a transport mechanism and the JMS application layer API for administration and sending and receiving messages. These systems usually support two messaging paradigms: point-to-point (PTP) and publish-subscribe (pub-sub). PTP is a mechanism in which messages are sent from a particular source to a particular destination. The messaging system uses queues to accomplish this. Pub-sub works differently in that messages are sent to possibly multiple destinations. Receivers subscribe to a particular topic and when a message is generated for that topic all the receivers who subscribe to it receive the message.

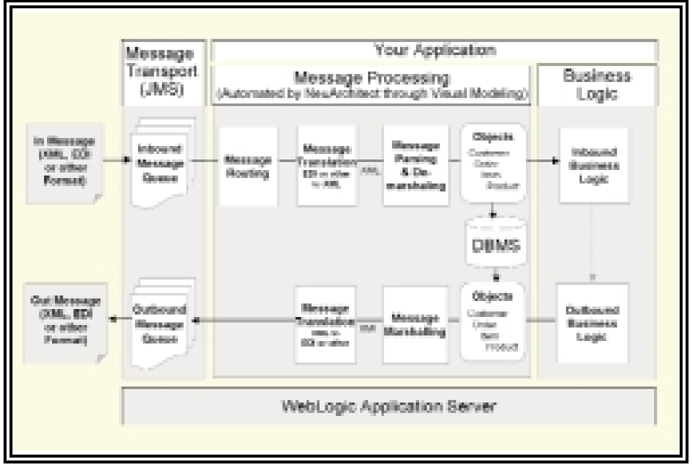
With the release of the EJB 2.0 specification, a new mechanism was added to support JMS messaging: the message-driven bean or MDB. The MDB container provides a standardized listening

FIGURE 1



Application integration: Needs and technologies

FIGURE 2



Rapid messaging integration using XML and JMS

LARGEST GATHERING OF DEVELOPERS, PROGRAMMERS, AND *i*-TECHNOLOGY PROFESSIONALS IN THE WORLD!!

JUNE 24-27, 2002 • JACOB K. JAVITS CONVENTION CENTER • NEW YORK, NY

WIN A $35,000 LUXURY CAR!

ATTENDEES WILL BE INVITED TO TAKE A GOLF SWING TO WIN AND RIDE OFF IN A $35,000 LUXURY CAR!

Sponsored by webMethods

REGISTER BY MAY 31 TO SAVE $200!

GO TO WWW.SYS-CON.COM NOW!

JAVA XML WEB SERVICES

Focus on Java

Java, now mainstream, is the dominant back-end technology upon which next-generation technologies are evolving.

Hear from the leading minds in Java how this essential technology offers robust solutions to *i*-technology professionals and senior IT/IS strategy decision-makers.

JAVA IN JUNE ESPECIALLY IN NEW YORK

JDJ EDGE conference&expo

Java Track: June 25-27

The Java Track, with something for every developer from beginner to advanced, will also look into the role that Java is playing in building up Web Services.

JV1	Java Security Advanced Concepts
JV2	Optimizing Database Performance in J2EE Applications
JV3	Detecting, Diagnosing, and Overcoming the Five Most Common J2EE Application Performance Obstacles
JV4	Building Asynchronous Applications Using Java Messaging
JV5	Building "Smart Client" Applications using J2SE and J2ME
JV6	.NET vs J2EE
JV7	Building Scalable Web Applications and Web Services
JV8	Hot Breaking Session
JV9	Java Tools for Extreme Programming
JV10	Building Truly Portable J2EE Applications
JV11	Security for J2EE Application Servers

Focus on XML

XML is today's essential technology to develop and deploy Web services. Here's your chance to learn from expert practitioners how XML is making the next phase of the Web a reality.

Focus on standards, interoperability, content management, and today's new quest for more efficient and cost-effective Internet and intranet-enabled business process integration.

XML NEXT G OF ENTERPRISE DEPLOYMENT

XML EDGE conference&expo

XML Track: June 25-27

The XML Track will focus on the various facets of XML technologies as they apply to solving business computing problems. This track targets audiences ranging from beginners to system architects and advanced developers.

XM1	Data - a Key Part of Web Services
XM2	OASIS Standards Update
XM3	A Universal Business Language
XM4	Achieving Standards-Based Mobile eBusiness Success with XML & Web Services
XM5	Using XML for Rapid Application Development and Deployment with Web Services
XM6	Open Up Your RDBMS with Open Standard Technologies
XM7	XML Web Services: Standards Update
XM8	Bringing XML to PKI
XM9	Building a Corporate Information Model in XML
XM10	XML in the Enterprise and Inter-Enterprise World
XM11	XML and RDB Relationships

Focus on Web Services

Web Services, the next generation technology that will enable the Internet to work for you and your business, and finally provide that ROI you have been after, will be put under a microscope at Web Services Edge East 2002.

Information-packed sessions, exhibits, and tutorials will examine Web Services from every angle and will provide cutting-edge solutions and a glimpse at current and future implementations. Hear from the innovators and thought leaders in Web Services. Enjoy a highly interactive CEO Keynote panel that will debate and discuss the realities and promise of Web Services.

WEB SERVICES SKILLS, STRATEGY, AND VISION

web services EDGE conference&expo

Web Services Track: June 25-27

The Web Services Track will concentrate on the latest emerging standards. It will include discussions of .NET, Sun ONE, J2EE and App Servers, the role of UDDI, progress of the standards-making bodies, SOAP, and Business Process Management. It is intended for developers, architects, and IT management.

WS1	Starting Out In Web Services: Fundamentals In Web Services
WS2	State of the Web Services Industry
WS3	Web Scripting Languages: Options for Dynamic Web Development
WS4	Building a Web Services Application Infrastructure
WS5	Deploying a Corporate Portal
WS6	The Enterprise Service Bus: (ESB): Leveraging Web Services
WS7	Developments in Web Services Standards
WS8	Unlocking the Value of Enterprise Web Services
WS9	Guarding the Castle: Security and Web Services
WS10	Practical Experiences with Web Services and J2EE
WS11	Designing Web Services Using UML

.NET TRACK: June 25-27

Combining technology, platform, and architecture, the .NET Track provides insight on the latest developments for the Windows platform.

NT1	Configuring .NET for Security, Performance, and Reliability
NT2	Changing Your Environment to .NET
NT3	Going Mobile with .NET
NT4	.NET on Other Platforms (FreeBSD, MONO, Portable .NET)
NT5	Inside the CLR
NT6	Accessing Data from .NET
NT7	Advanced .NET Web Services
NT8	Migrating Legacy Code to .NET
NT9	Advanced User Interfaces with GDI+

IT Strategy Track: June 25-27

The IT Strategy Track will focus on the managerial and design aspects of the various development disciplines. Topics will include Standards, Architecture, Design Patterns and Best Practices, Project Planning and Management, and Extreme Programming.

IT1	Developing, Deploying and Managing Web Services
IT2	Key Trends and Technologies for Building an Enterprise Web Services Architecture
IT3	Selecting a Framework: Toolkit, Platform, or Roll Your Own?
IT4	Getting Started with Web Services
IT6	Overcoming the Web Services Barriers
IT7	The Real Issue: Improving Your Enterprise with Enterprise Web Services
IT8	Minimizing Risks and Maximizing Investments in J2EE Development Through the Use of Reusable Application Architecture and Frameworks
IT9	Application Integration – Building a Flexible Web Services Architecture
IT10	The Economics of Web Services
IT11	Hooking It All Together - Application Integration Case Study

Who Should Attend...

- DEVELOPERS, PROGRAMMERS, ENGINEERS
- *i*-TECHNOLOGY PROFESSIONALS
- SENIOR BUSINESS MANAGEMENT
- SENIOR IT/IS MANAGEMENT/C LEVEL EXECUTIVES
- ANALYSTS, CONSULTANTS

Sun Microsystems at JDJEdge 2002: Java Fast Paths & Java University℠ Program at the Jacob K. Javits Convention Center, NYC

Java Fast Paths

Attend fast-paced, code-level, full-day Java technology developer training designed to provide you with the skills to confidently approach the industry's core Java technology certification exams. Don't just say you know it… prove it! Most developers recognize the expanding importance of gaining Java technology certification. If you're like those who've already taken the exams, the real hurdle to certification is finding time to prepare for the tests.

Monday, June 24, 2002:

- Java™ 2 Platform: Developer Certification Fast Path
- Java™ 2 Platform: Architect Certification Fast Path
- Web Component Developer: Certification Fast Path

Java University℠

The Java University℠ Program complements this year's JDJEdge Conference by offering three aggressive, full-day, code-level training classes for experienced software developers, architects and engineers.

Attend code-level training in sessions designed by industry luminaries, and recognized experts. Sessions cover XML and Web services technology. Whether you're a beginning or a veteran developer, architect, or software engineer, you'll benefit from these value-packed full-day courses. Register now. Seating is limited.

Tuesday, June 25, 2002
Developing Solutions Using Java™ Technology and XML

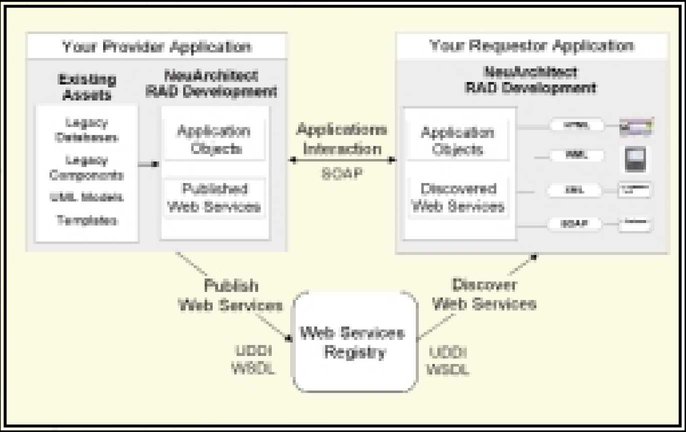
Wednesday, June 26, 2002
Web Services Programming Using Java™ Technology and XML

Thursday, June 27, 2002
Java™ APIs for Enterprise Web Services

FOR EXHIBIT INFORMATION CONTACT: MICHAEL PESICK 135 CHESTNUT RIDGE RD. • MONTVALE, NJ 07645 • 201 802-3057 MICHAEL@SYS-CON.COM

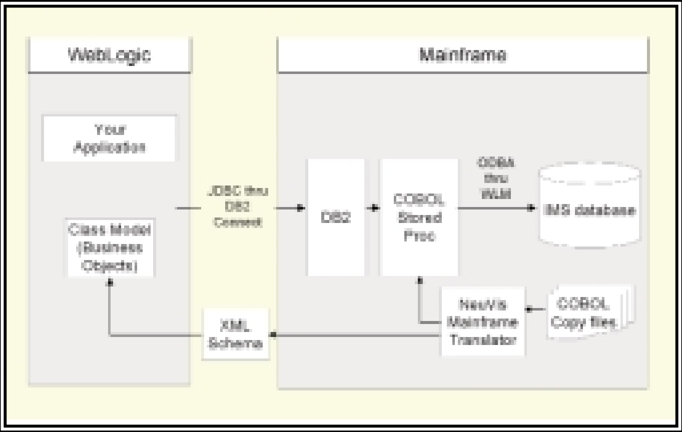
OWNED AND PRODUCED BY SYS-CON EVENTS SYS-CON MEDIA

FIGURE 3



Rapid Web services integration

FIGURE 4



Rapid legacy integration

service that monitors queues or topics for messages. It has a single standard interface method called OnMessage() that is executed once a message is received. This mechanism eliminates the need for the developer to write complicated listeners that would need to provide thread management, load balancing, and transactional controls. The container handles all of it. WebLogic supports the concept of the MDB in versions 6.0 and higher.

Now let's look at how you can create some rapid integration solutions using WebLogic and NeuArchitect.

RAPID INTEGRATION USING XML AND JMS

As a first step, you will need to do some setup in the WebLogic Console. You need to do the following:

1. Enable JMS for the main server instance. Click on the server instance and select the services tab and then the JMS tab.
2. Create a JMS Server instance and assign it to your main server instance. You can do this by clicking on the JMS Servers node in the administration tree.
3. Once a JMS Server is created, you need to create your destinations. Under your new JMS Server node, create your queues.

That's pretty much it as far as configuration. You can then use NeuArchitect to create the business logic to process either inbound or outbound messages. In NeuArchitect, you need to do the following:

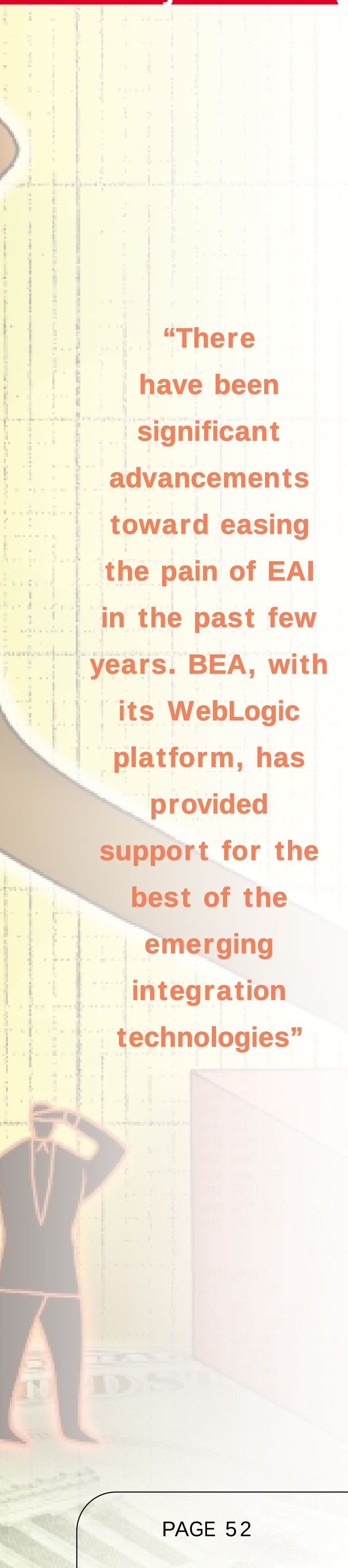
1. Create a project. Set the application server to BEA WebLogic 6.1 (or later). Set the Messaging technology to JMS.
2. Using the class diagram, create your business classes. The classes you define will be generated for the target technology with a host of essential plumbing code, so all you have to do is write your business methods.
3. For a given class, use the Messages tab to define your message. You can identify it as inbound or outbound, and select what queue to use, encryption methods, and so on. For inbound messages, you can choose to use MDBs to process them.
4. For your message, you can use the NeuArchitect Message Mapper to load a DTD for an XML message and then map the elements of the DTD to the business classes you created in the class model. It will generate all the code for reading the message and mapping the elements of the message to the business objects.
5. Write your business method for processing the data that was mapped to your classes. Instead of parsing methods, you simply address objects and attributes. The NeuArchitect API allows you to insert the information into a database or do any further processing you wish.
6. Press the Construct Messages button to generate the Java code, build the beans, and automatically deploy them onto WebLogic.

Web Services

Web services is a newer technology that has the potential to transform integration and computing itself. The technology allows programs to communicate with one another using standard protocols such as HTTP. A service provider will write a piece of software that provides a useful business function. Using special tools, the service provider creates a definition of this service using Web Services Description Language (WSDL). Using this definition, the service provider will then publish the service to a Universal Description, Discovery, and Integration (UDDI) server that acts as a lookup registry similar to the way DNS works for finding hosts on the Internet. Anyone who wishes to use this service will query the UDDI server to discover a list of services from the provider and select the one they wish to use. As part of the discovery process, the user is provided with information on the nature of the service, how to call it, what parameters it accepts, and what information is returned from the service. The mechanism for communicating between a program and a Web

Simplex Knowledge Company

www.skc.com/training



“There have been significant advancements toward easing the pain of EAI in the past few years. BEA, with its WebLogic platform, has provided support for the best of the emerging integration technologies”

service is Simple Object Access Protocol, or SOAP. SOAP is XML-based, and provides a powerful mechanism for making requests and receiving responses (see Figure 3).

RAPID INTEGRATION USING WEB SERVICES

BEA, along with several other vendors, has been an industry leader when it comes to Web services support. WebLogic Version 6.1 supports the deployment of Web services by providing automatic generation of WSDL definitions and the handling of SOAP messages for deployed Web services. NeuArchitect also provides development support to make it easy to create and deploy Web services on WebLogic. Here’s what you need to do:

1. There’s no real setup for the WebLogic Server. The Web services support is supplied by the basic installation of Version 6.1 or higher.
2. Using NeuArchitect, create an application and create your business classes.
3. When you create your business classes, you also create your business rules in the form of methods on the classes. These rules are written in Java. With a click of a mouse, you can declare any of these methods to be Web services. Additionally, NeuArchitect allows you to create and publish components that can combine several classes and methods together. Again, with a click of a mouse, all the methods in the component can be defined as Web services.
4. Once you’ve created your methods and/or components, construct them. NeuArchitect will use WebLogic’s inherent WSDL creation facility to generate the WSDL. WebLogic also provides the location of the WSDL, which is needed for publishing to a UDDI server.
5. The developer can point at a public or private UDDI server, and then provide the WSDL from WebLogic to indicate where the Web services are published. All of this is done with a few clicks of the mouse.

Web services can also be used to reinvigorate existing assets. NeuArchitect can create wrappers that call existing functions. These wrappers would then be published to WebLogic as Web services.

J2EE Connector Architecture

In the 1990s, enterprises began the process of consolidating internal systems by deploying enterprise resource planning (ERP) systems from companies such as SAP, Baan, J.D. Edwards, and PeopleSoft. These systems provided all the basic functionality for an enterprise: human resources, accounting, order processing, and inventory control, to name a few. These systems, once deployed, became the lifeblood of the organization, and a need arose to integrate them with other systems used by the enterprise.

Initial efforts required coding to a complex API provided by the ERP vendor. Sun has attempted to address this problem by providing the J2EE Connector architecture specification. WebLogic has introduced support for this in Version 6.1 and higher, providing a standardized method for accessing these ERP systems. ERP vendors write adapters to their systems that provide connectivity, as well as a discovery mechanism so the calling program can know the structure of the data that the ERP system supports.

RAPID INTEGRATION WITH LEGACY CONNECTORS

In order to integrate with legacy systems, you need to use the appropriate legacy connector. NeuArchitect provides legacy connectors to access data on IMS and VSAM databases and logic on CICS and MVS platforms. It also integrates with more than 140 legacy adapters offered by iWay Software. The following steps illustrate rapid legacy integration with the IMS database connector (see Figure 4).

1. Using the mainframe translator software provided with NeuArchitect, translate your COBOL COPY files into an XML schema file that will be imported into NeuArchitect and COBOL stored procedures that will perform the read/write access on the mainframe.
2. Import the XML schema file into NeuArchitect. It’s converted into a UML class model that remains mapped to your IMS database.
3. You can now develop the rest of your application – e.g., you can integrate the application through messaging and Web services and design Web and wireless transactions. Autoconstruct and deploy the application.
4. When you use the application, it connects to the mainframe using JDBC through DB2 connect and uses the COBOL stored procedures that were generated in Step 1 to access the IMS database for read/write access.

In a similar manner, you can use other legacy connectors to access other types of legacy databases and logic.

Conclusions

There have been significant advancements toward easing the pain of EAI in the past few years. BEA, with its WebLogic platform, has provided support for the best of the emerging integration technologies. Moving forward, BEA will continue to refine and enhance WebLogic’s integration capabilities.

Coupled with RAD tools such as NeuVis’ NeuArchitect, the obstacles to integrating complex systems are being eliminated. This allows developers to use integration to form a comprehensive, cohesive enterprise information network that adds enormous value to the enterprise.

Now in More than 5,000 bookstores worldwide

subscribe Now!

FOR FAST DELIVERY

Go Online and Subscribe Today!

The World's Leading Independent WebLogic Developer Resource
FOR WLS DEVELOPERS BY WLS DEVELOPERS
BEA WebLogic
DEVELOPER'S JOURNAL
An introduction to... WebLogic

SPECIAL INTRODUCTORY OFFER SAVE $31*

HURRY, DON'T DELAY! OFFER EXPIRES JUNE 30, 2002

Helping you enable intercompany collaboration on a global scale

- Product Reviews
- Case Studies
- Tips, Tricks and more!

WebLogicDevelopersJournal.com

SYS-CON Media, the world's leading publisher of *i*-technology magazines for developers, software architects, and e-commerce professionals, brings you the most comprehensive coverage of WebLogic. *Only $149 for 1 year (12 issues) regular price $180.

SYS-CON MEDIA

WLDJ ADVERTISER INDEX

ADVERTISER	URL	PHONE	PAGE
Altaworks	www.altaworks.com	603-598-2582	7
BEA	http://developer.bea.com	408-570-8000	2,3
BEA eWorld Europe	www.bea-eworld.com	404-240-5506	29
BEA Partner Pavilion	www.sys-con.com	201-802-3057	45
EnginData	www.engindata.com		56
JDJEdge	www.sys-con.com	201-802-3069	49
JDJ Store	www.jdjstore.com	201-802-3012	33
Mongoose	www.mongoosetech.com	281-461-0099	11
Precise	www.precise.com/wldj	800-310-4777	17
Rational	www.rational.com/ruc	888-889-0074	19
ReportMill	www.reportmill.com	214-513-1636	27
Simplex Knowledge Company	www.skc.com/J2EE	845-620-3700	43
Simplex Knowledge Company	www.skc.com/training	845-620-3700	51
Sitraka	www.sitraka.com/performance/wldj	800-663-4723	15
Sitraka	www.sitraka.com/jclass/wldj	800-663-4723	31
Sitraka	www.sitraka.com/performasure/wldj	800-663-4723	60
Sun Microsystems	www.sun.com/servers/entry/beapromo3	800-765-9200	37
SYS-CON Media	www.sys-con.com	201-802-3012	44
SYS-CON Industry Newsletters	www.sys-con.com	201-802-3020	41
SYS-CON Reprints	www.sys-con.com	201-802-3026	57
WebGain	www.webgain.com/toplink_create3.html	877-WebGain, ext. 15858	23
WebLogic Developer's Journal	www.sys-con.com/weblogic	888-303-5282	53
Web Services Journal	www.wsj2.com	888-303-5282	37
Web Services Resource CD	www.jdjstore.com	201-802-3012	34, 35
Wily Technology	www.wilytech.com	888-GET-WILY	4
XML Journal	www.sys-con.com	888-303-5282	32

Advertiser is fully responsible for all financial liability and terms of the contract executed by their agents or agencies who are acting on behalf of the advertiser.

NEXT MONTH

Focus on Performance

WebLogic Server Performance Tuning
The many ways your server can be tuned

JRockit
Enhancing your performance

The Grinder – Load Testing for Everyone
An easy-to-use, Java-based, load-generation and performance-measurement tool

Plus:

Make the Most of WebLogic Classloaders
Strategies and packaging designs to effectively work with them

Using WLS EJB-QL Extensions
A closer look at WebLogic's extensions to the EJB-QL

BEA WebLogic DEVELOPER'S JOURNAL

Building Adaptive Applications With Reusable Components

USING THE WEBFLOW AND PIPELINE FRAMEWORKS IN WEBLOGIC PORTAL 4.0

BY DWIGHT MAMANTEO

AUTHOR BIO...

Dwight Mamanteo, a technical manager with BEA's Global Alliances Technical Services organization, has been with BEA since 1999. His responsibilities include providing technical enablement support to BEA's strategic SI and ISV partners. Dwight has been involved with object-oriented programming, design, and architecture since 1993.

CONTACT...

dwight@bea.com

Two daunting tasks face application architects and project managers alike.The first is to architect a solution that will reduce the risks involved with implementing new and changing business requirements during the application development and post-application deployment stages of a project. The second is to architect a solution that will reduce the development time and increase the corporation's Return on Investment (ROI) in past projects by reusing prebuilt visual and business components

BEA WebLogic Portal 4.0 contains frameworks designed to allow the creation of adaptive applications using visual, data-cleansing, and business components reaped from previous application development projects. The Webflow and Pipeline frameworks enable developers to dynamically link together reusable components that follow the Model-View-Controller (MVC) design pattern.

Webflow Framework

The Webflow framework is designed to build Web applications that can adapt to changing Web-business requirements. The main controller behind the Webflow framework is the WebflowExecutor class (see Figure 1). The WebflowExecutor intercepts events emanating from a visual, input processor or business component, consults with a centralized XML-based application scenario file; and invokes the next predefined visual, input processor or business component.

The XML-based application scenario file enables the decoupling of application components, which is key to developing adaptive applications with reusable components. During runtime, each component sends an event type and location identifier to the WebflowExecutor via the WebflowServlet. The WebflowExecutor then determines the next component in the sequence to execute. The runtime determination of sequence flows by the WebflowExecutor allows the developer to simply modify the scenario file in order to alter the business flow of the online application. The alteration of the sequence file may include rearranging existing links between components, removing existing components, or adding new components. The ability to add and remove components in an application that uses the Webflow and Pipeline frameworks without any code modifications is made possible by using the provided JSP Webflow tag libraries (in the case of visual components) and by implementing the appropriate interfaces (in the case of input processors and pipeline components).

VISUAL COMPONENTS

The visual components that can be used in the Webflow framework are JSP and HTML pages. The only supported events that visual components can generate are hyperlinks and forms. In the case of JSP pages, the developer need only inform the Webflow servlet of the event type and origination ID, which would then be forwarded to the WebflowExecutor. JSP helper tag libraries are provided to help the developer in developing JSP pages, as shown below.

```
<a href="<webflow:createWebflowURL
origin="checkout.jsp" event="link.continue"
httpsInd="http"/>"> Check Out </a>
```

The example above is invoked when the user selects that corresponding hyperlink, in which case the origin ID ("checkout.jsp"), the event type ("link.continue"), and the protocol to be used ("http") are sent to the Webflow servlet.

In the case of HTML visual components, the visual component developer would pass the

Webflow servlet the same information as a JSP visual component. However, since JSP tag libraries are not usable in HTML pages, the information has to be explicitly placed in the appropriate format, as shown below:

```
<a href=http://localhost:7501/example/application?name-
space=order&origin=checkout.jsp&event=link.continue>
Check Out </a>
```

The fact that destinations aren't hard-coded in each visual component allows the developer to rearrange the site map without having to modify JSP or HTML code.

INPUT PROCESSOR COMPONENTS

In order to ensure that the information being sent to the business components is valid and well-formatted, a data-cleansing step should be added to the execution flow. InputProcessors are specialized Java classes that can be used to perform the data-cleansing task. To add an InputProcessor to the Web application scenario, a developer would simply implement the InputProcessor interface or extend the InputProcessorSupport class (see Figure 2), and link the InputProcessor to the appropriate components in the XML-based scenario file.

The InputProcessor developer may also use an implementation of the ValidatedValues interface to report back to the visual component the results of the completed data-cleansing task. The visual component developer can then access the status of each form field via the <Webflow:-validatedForm> JSP tag. In addition to cleansing data, InputProcessors can also be used to perform conditional branching within a Web application where, depending upon the branching logic contained in the InputProcessor, the return value will tell the WebflowExecutor which visual or business component to execute next.

After the InputProcessor completes its processing, a return object is sent back to the Webflow-Executor. The return object, coupled with the knowledge of the InputProcessor type, allows the WebflowExecutor to determine the next component to execute by referring to the XML-based scenario file. Similar to the visual components, hard-coded links to other components are not implemented in the InputProcessors, which allow the developer to reuse data-cleansing and conditional-branching components, and provides the ability to change the application's behavior without recoding.

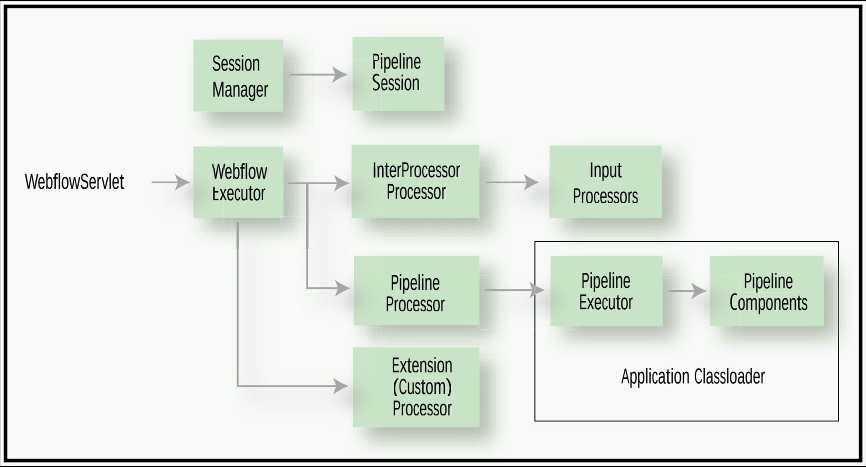
Pipeline Framework

The Pipeline framework is designed to build business applications that can adapt to changing business requirements. The framework follows the Pipe-and-Filter architectural pattern and works in conjunction with the Webflow framework. After the WebflowExecutor has consulted the XML-based application scenario file and determined that a pipeline needs to be executed, control is handed over to the PipelineExecutor to execute the pipeline. After completion of the pipeline invocation, control is handed back to the WebflowExecutor to determine which components in the application sequence to invoke next. It's important to understand that all the components in the Webflow and Pipeline frameworks can be linked together in any sequence and that information may be shared between these components through the PipelineSession object.

PIPELINE

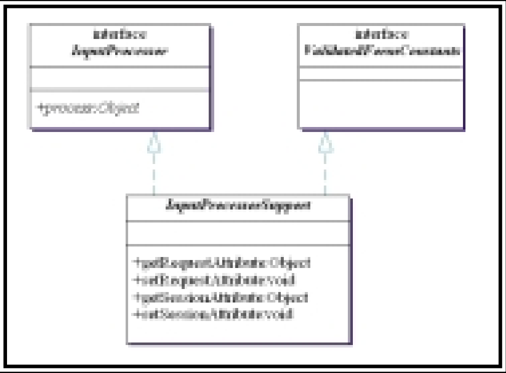
The PipelineExecutor serves as the controlling class in the Pipeline framework, handling the linkage and exceptions between pipeline components. The pipeline components in the Pipeline framework are essentially business components that perform the required business functionality. Each pipeline can be composed of a sequence of one or more pipeline components and exceptions can be generated by any of the pipeline components. These exceptions can either be fatal, which

FIGURE 1



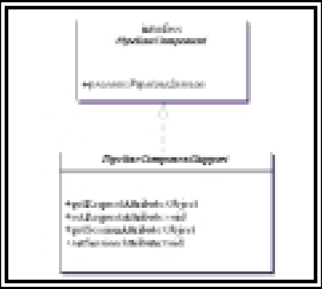
High-level architecture

FIGURE 2


interface
InputProcessor
+process:Object
interface
ValidatedFormConstants
InputProcessorSupport
+getRequestAttribute:Object
+setRequestAttribute:void
+getSessionAttribute:Object
+setSessionAttribute:void

InputProcessor hierarchy

FIGURE 3



Pipeline component hierarchy

stops the processing of the pipeline, or nonfatal, which allows the next component in the sequence to be invoked. A pipeline can also be wrapped in a single transactional context, allowing the components in the pipeline to act as one atomic unit of work. As with the WebflowExecutor in the Webflow framework, the PipelineExecutor references a pipeline scenario script to determine which pipeline component to invoke next, given a specific event and origination.

BUSINESS COMPONENT

The business components that can be linked together in a single pipeline may consist of Java and Stateless Session EJB classes. The business component (i.e, pipeline component) is generally used to execute coarse-grain business logic or integrate to existing back-end systems.

To add a business component to the pipeline sequence, a developer would simply implement the PipelineComponent interface or extend the PipelineComponentSupport class (see Figure 3), and link the business component implementation to the other business components in the XML-based pipeline scenario file.

In order to allow the sharing of information in a pipeline, the PipelineSession object is passed between business components via the required Process method implementation. The PipelineSession object can also be contained within the same transactional context as the components in the pipeline itself.

GUI Editor Tool

Since the XML-based scenario files are integral to the Webflow and Pipeline frameworks, the ability to create and edit these files is incorporated into the WebLogic Portal GUI tool, the E-Business Control Center (EBCC). Each framework has its own specialized drawing area (see Figures 4 and 5) and multiple scenario files are differentiated via a namespacing scheme. The visual editing tool is very easy to use with drag-and-drop, zoom, scenario-linking, and attribute-editing features. The ability to graphically create an application by linking together visual and business components from a corporation's certified IT application component library is a powerful step in building adaptable applications in a substantially reduced time frame.

Conclusion

The Webflow and Pipeline frameworks, which are part of the BEA WebLogic Portal product,

EnginData Presents

2002 Developer Market Survey Reports

Our comprehensive reports offer insight and strategy to guide your most critical business decisions in today's fastest growing technologies...

✓ Establish your product and marketing strategy

✓ Understand your customer's needs

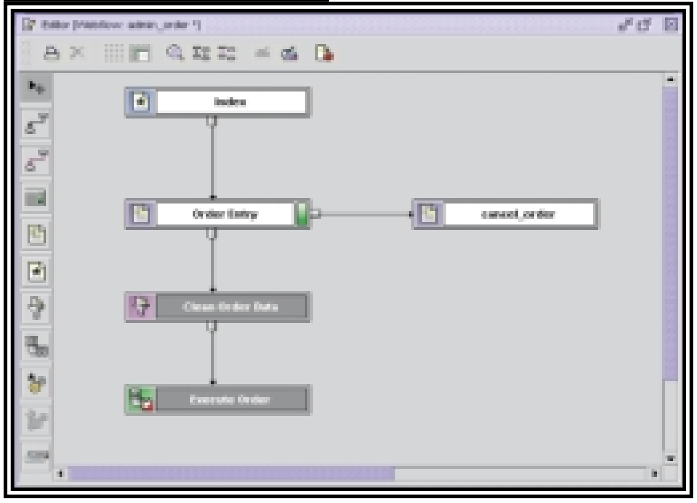
✓ Evaluate Technology & Trends

Preview and order reports at www.engindata.com

engindata.com

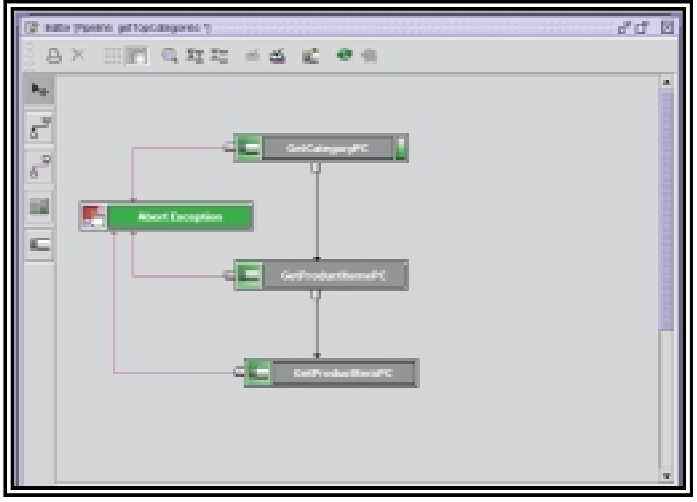
engindata✓ RESEARCH

FIGURE 4



Webflow scenario canvas

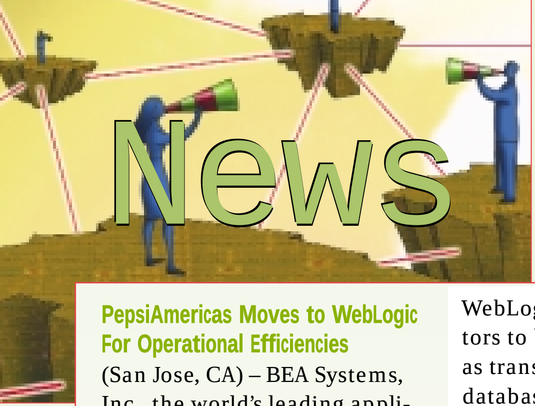
FIGURE 5



Pipeline scenario canvas

address the need for an architecture that is able to adapt to changing business requirements and improve ROI by enabling the reuse of application components.

By simply implementing the required framework interfaces, visual, data-cleansing, and business components can easily be linked together to form a fully functional application. This application can then be adapted to address new business functionality by visually relinking new or existing components from previous projects. Additionally, the implementation of the MVC design pattern provides the separation between the presentation logic and the underlying business processes, thus allowing the ability to modify any one of the tiers without impacting the others.



PepsiAmericas Moves to WebLogic For Operational Efficiencies

(San Jose, CA) – BEA Systems, Inc., the world's leading application infrastructure software company, has announced that PepsiAmericas, Inc., one of the world's largest beverage bottlers, has standardized on the BEA WebLogic Enterprise Platform for the company's back-office systems and Web-based application infrastructure.

PepsiAmericas (PAS) is migrating its Java-based applications from IBM WebSphere to BEA WebLogic because of changing business requirements, which resulted in a reevaluation of its software infrastructure. PAS conducted a thorough review of its platform requirements before choosing BEA for its support of industry standards, scalability, total cost of ownership, and integration capabilities. Meeting these requirements will allow PAS to more easily and cost-effectively integrate with third parties across its value chain and transform its technological capabilities from a business limitation into a strategic asset. www.bea.com, www.pepsiamericas.com

Wily Technology Announces Support for WebLogic Server 7.0

(Burlingame, CA) – Wily Technology, a leader in enterprise Web application management, has announced the integration of Introscope 3.0 with BEA Systems' application server, BEA WebLogic Server 7.0.

Introscope delivers component-level visibility into production Web applications, WebLogic Server, and connectors to back-end systems such as transaction servers and databases, allowing IT personnel to isolate application performance issues with pinpoint accuracy. Direct integration of Introscope's patented Agent technology with WebLogic Server allows customers to automatically monitor WebLogic Server as soon as Introscope is installed. Additionally, Introscope utilizes Java Management Extensions (JMX) for advanced monitoring of WebLogic, as well as better management control of any JMX-enabled application. www.wilytech.com

Sitraka JClass ServerViews and JClass DesktopViews Now Shipping

(Toronto) – Sitraka, a leader in J2EE performance assurance, has announced that Sitraka JClass ServerViews 2.1 and Sitraka JClass DesktopViews 6.0 are now shipping. All JClass products now support Java 2 SDK version 1.4.

Sitraka JClass ServerViews 2.1 is a set of 100% Java-based, server-side components that bring dynamic charts and Adobe PDF documents from a back-end J2EE application server to Web browsers. Sitraka JClass DesktopViews 6.0, formerly known as JClass Enterprise Suite, is a comprehensive collection of integrated Java components and is fully scalable to mission-critical development environments. www.sitraka.com

Research In Motion, BEA Mobilize Web Services

(San Jose, CA and Waterloo, Ontario: – BEA Systems and Research In Motion (RIM) Limited, a leader in the mobile communications market, have announced their strategic relationship to deliver a framework that eases the development and delivery of mobile enterprise Web services applications for the thousands of organizations using BlackBerry.

BEA and RIM plan to jointly deliver a framework for developing and deploying mobile Web services on the BEA WebLogic Enterprise Platform. The goal is to bring BEA WebLogic Workshop and BlackBerry together in a single solution to support an always-on, push-based architecture with end-to-end security and back-end integration to corporate systems. www.rim.net

Macromedia Introduces ColdFusion MX

(San Francisco) – Macromedia, Inc. has announced Macromedia ColdFusion MX, a rapid server scripting environment for creating rich Internet applications. Macromedia ColdFusion MX, previously code-named "Neo," brings the proven productivity of ColdFusion to the highly scalable, standards-based Java technology architecture. ColdFusion MX offers innovations for creating rich Internet applications and working with XML, Web services, and the Microsoft .NET Framework.

ColdFusion MX is built on a new architecture that delivers the scalability, reliability, and power of the Java platform without the complexity. Customers can deploy ColdFusion MX as a standalone server or on top of Java application servers, such as BEA WebLogic. ColdFusion MX for BEA WebLogic Server is expected to ship later this year. www.macromedia.com

Ceon Introduces Ceon Integration and Provisioning Suite 4.0

(Redwood City, CA) – Ceon Corporation, the Intelligent Service Fulfillment Company, has announced its Ceon Integration & Provisioning Suite 4.0 (Ceon IPS 4.0), fourth-generation platform that adds a significant performance and scalability "boost" to its comprehensive, flexible, and feature-rich service fulfillment solution.

Ceon IPS 4.0 provides event-based provisioning solutions that ensure network elements and software systems perform all provisioning tasks accurately and automatically. It enables service providers to drive down operating costs by reducing the number of manual provisioning steps, accelerate revenue growth by enabling service providers to introduce and deploy new products, and empower customers with self-activation so they can begin using any and all services they demand.

Ceon IPS 4.0 is based on open standards with a distributed, highly scalable architecture designed for multiservice solutions and to solve service-fulfillment problems. www.ceon.com

Sun Microsystems

www.sun.com/servers/entry/beapromo3

Sitraka

www.sitraka.com/performasure/wldj