Deliver Composite Applications with Java, WS-BPEL & SOA

Supporting the complete lifecycle

by Kevin Schmidt, Prabhu Balashanmugam, and Gopalan Raj

Java is an outstanding language for building components, services, and many applications that are vendor and platform neutral. The vast adoption of Java technology by the industry in the past decade is a testament to the power of Java. Development of new applications, services, and components using Java is not going away, but many organizations have progressively moved to the next phase in maturing their IT Infrastructure. This phase is driven by many factors including how businesses operate today, having to constantly adjust to market trends, and that IT has moved from being a support organization to being the backbone of business and, hence, needs to keep pace with the organization. Continuous and faster alignment with changing business needs, time-to-market, and cost are the factors that determine success in this phase.

There are some technologies that are starting to play a critical role in this phase, for example, service-oriented architecture (SOA) is a key enabler. Java EE technology is a natural service-enabler of existing applications, thereby forming the foundation of SOA. Service-enabled applications create the opportunity to compose functions from disparate and cross-functional applications to model business processes that transcend application and enterprise boundaries. Web Services Business Process Execution Language (WS-BPEL) provides a faster way to compose and orchestrate services by reuse. Java and WS-BPEL complement each other perfectly and provide a solid foundation for integrating services and delivering composite applications.

This article will briefly explain what these technologies are and how they can work together to improve developer productivity and business agility.

Kevin Schmidt is the director of product management for SOA and Business Integration Software at Sun Microsystems, Inc. He has 17 years of experience in the software industry in roles that include product management, professional services, pre-sales, and development. Most recently he has focused on composite applications using Java, Web services, SOA, BPM, and related technologies.

kevin.schmidt@sun.com

The Technologies – Java, WS-BPEL, and SOA

There are no globally accepted definitions for the technologies that this article will explore. So let's clarify them now as a common understanding is needed before getting into the details.

Service-oriented architecture is a technical pattern for implementing cohesive and loosely coupled business and technical functions with well-defined interfaces. Such services are consumed through the details specified in the interface and without any knowledge about the implementation.

While the SOA-based infrastructure model shown in Figure 1 has a few, but well-defined, layers of services, in reality there may be many more layers, as the services are reused and composed to create coarser-grained services. As illustrated in the diagram, services can consume each other to provide layers of services and such a model can be implemented using any language including Java. The service is not technology-dependent as long as it can be consumed through the well-defined interface. Java EE 5 is a set of coordinated technologies that enable solutions for developing, deploying, and managing server-centric applications. WS-BPEL is an XML-based execution language that can be used to compose the coarse-grained services into broader services or complete applications.

Tough Decisions

Technology without the right set of tools often does more damage than good. Identifying the right set of tools for a technology is as important as choosing the technology.

The science of delivering composite applications becomes more of an art when architects try to understand when to switch from Java to WS-BPEL. This decision often determines the agility of the composite application.

The Right Set of Tools

Sun Microsystems provides a bundle of tools and servers that we will be using for this article to build our composite application, specifically the Java EE 5 SDK Update 2 Tools Bundle. This can be

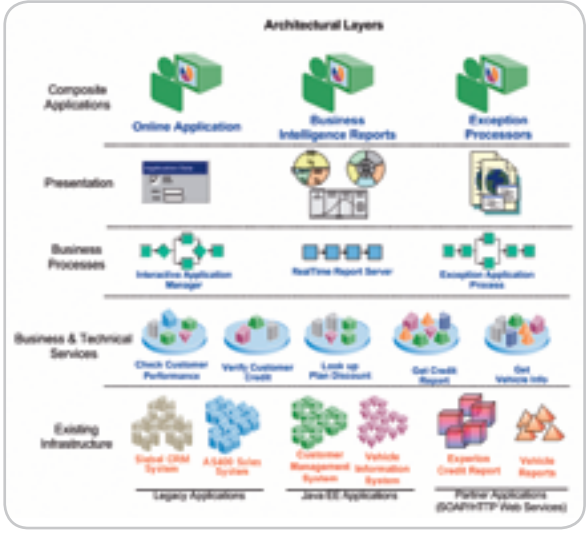

Figure 1 SOA-based infrastructure

found at http://java.sun.com/javaee/downloads/ and includes:

- Java EE 5 Application Server
- NetBeans 5.5 with NetBeans Enterprise Pack 5.5
- Java Business Integration (JBI) runtime and WS-BPEL Service Engine

This bundle provides the integrated tooling and runtime environments needed to build, deploy, and manage composite applications using Java and WS-BPEL. The tooling, in addition to a full Java SE/EE development environment, includes a WS-BPEL editor and debugger, WSDL and XSD editors, and a project system for creating services and composing them into applications. The runtime provides the Java EE 5 application server but also the WS-BPEL engine needed to execute the business processes created in the WS-BPEL editor.

In effect this bundle provides the capability to support the complete life cycle to develop, deploy, and manage applications and business processes, composing services from Java EE applications and Web services.

Putting the Technology into Action

Use Case

To put the technology and the tools in context, let's take a scenario that you must be very familiar with, a scenario in which a new external-facing application is being developed iteratively by reusing functions from existing Java EE applications and external Web services.

In the architecture depicted in Figure 2, there are three existing Java EE Applications: 1) the *Performance Evaluator* – provides evaluations on the fiscal performance of existing customers; 2) the *Vehicle Information Server* – evaluates market conditions and vehicle history and serves a detailed report on the worthiness of the vehicle; and 3) the *Financial Index Server* – provides the current running rates and values that should be used to compute the loan details.

The new application should provide a service to receive loan applications over the Web and respond with a decision. The loan decision should be arrived at by applying predefined processing rules on the following data: 1) applicant's financial worthiness, 2) vehicle's value, 3) loan amount, and 4) current policies and indices for loans.

Design

The new application could be completely implemented in Java technology with just servlets, JSPs, EJBs, and POJOs. However, WS-BPEL is a more suitable language for service composition and business process execution. As mentioned earlier, the decision about when to start composing services using WS-BPEL can be tricky. The general rule is that Java can be used for extending existing services or composing fine-grained Java EE services to create more services that may not yet provide a cohesive and complete business or technical function. Should the new processing unit implement business or technical logic, which is more than invoking services and basic data transformations between invocations, that's an indication that it may be more suitable to implement Java. However, when the application or composition needed is process-centric and involves invoking services and hooking them together with flow logic and simple mappings, WS-BPEL may be the best way to implement this layer of the application.

The new application requires the development of the following services: 1) *Applicant Evaluator* – a service that aggregates an internal report if the applicant is an existing customer with an external credit report; 2) *Loan Approval Processor* – a service that takes an applicant's fiscal performance, vehicle report, and current indicators and comes up with a decision; and 3) the *Vehicle Loan Application Process* that will be exposed to the external world. This process will receive an incoming credit application, invoke the Applicant Evaluator Service, Vehicle Information Service, and Financial Index Service in that order and pass the results from all these services to the Loan Approval Processor service and return the decision to the caller.

Figure 2 Existing applications, functions, and technology

Figure 3 Proposed applications, functions, and technology

Figure 4 Annotated Vehicle Information Server EJB

The Applicant Evaluator service will be implemented as a stateless session bean annotated to be exposed as a Web service. This service will invoke an external Web service to get a credit report and also invoke another stateless session bean – Performance Evaluator – to get an internal credit report, if one is available.

The Vehicle Information Server and Financial Index Server Stateless Session Bean functions will be exposed as Web services. The Loan Approval Processor service will be implemented as a stateless session bean and annotated to be exposed as a Web service.

The Vehicle Loan Application Process will be implemented as a business process that invokes the Applicant Evaluator, Vehicle Information Server, Financial Index Server, and Loan Approval Processor services in that order (see Figure 3).

Implementation

Java EE technology and the tools provided in NetBeans allow users to adopt bottom-up or top-down approaches or a combina-

Prabhu Balashanmugam is a product manager for SOA and Business Integration Software at Sun Microsystems, Inc. He is currently responsible for managing Business Process Management and Data Management products. Prior to joining Sun Microsystems, he has implemented and managed various software solutions for companies across the globe.

prabhu.balashanmugam@sun.com

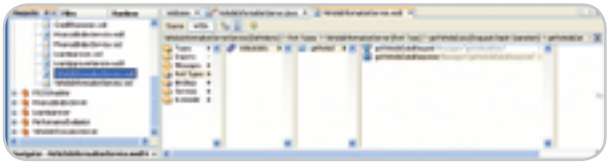

Figure 5 Generated WSDL for the annotated EJB

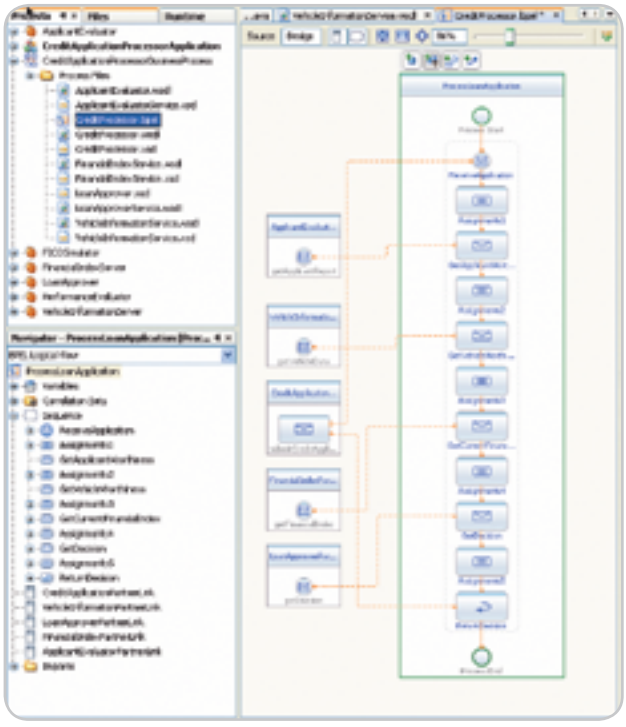

Figure 6 Loan application evaluation process

tion of both to deliver composite applications. In this example we will follow a bottom-up development approach. We'll start by building Java services, followed by invoking the services in a business process. We'll then compose the services in a composite application, deploy the application on Sun Application Server 9.1, and test the composite application.

NetBeans Enterprise Pack 5.5 provides the necessary capability to perform all the above tasks from within the IDE without the need to use any other tool or editor.

The Java Services

Java EE 5 enables functionality in an existing Java EE application to be easily service-enabled by annotating the Java classes. The creation of new Java EE applications that are Web-service enabled can also be done easily using the EJB Module Creation Wizard in NetBeans. In the first phase, the EJB Modules Vehicle Information Server and Financial Index Server are service-enabled by adding the appropriate annotations to the classes.

In the second phase, two new services are created. The Applicant Evaluator service is a stateless session bean also exposed as a Web service. This service aggregates results from an external Web service, FICOSimulator, and the Performance Evaluator stateless session bean. The FICOSimulator Web service simulates an external Web service that returns a credit report (see Figure 4).

The Loan Approval Processor is another service developed as a stateless session bean, also exposed as a Web service. This service combines reports on the applicant's credit worthiness, the vehicle's value, the loan indicators, and returns a decision based on predefined business rules.

Composing the Services

The Business Process Editor in NetBeans Enterprise Pack 5.5 can be used to compose services. The Business Process Editor enables users to model business processes graphically in a visual environment and the WS-BPEL code is automatically generated, corresponding to the visual model. However, the user can choose to make changes directly in the generated WS-BPEL code and the visual model is automatically synchronized to the graphical view. The services invoked by business processes are defined as WSDLs (see Figure 5). Users can create new WSDLs using the WSDL and XSD Editors directly in NetBeans. However, users can also import existing WSDLs and XSDs and edit them using the editors.

A BPEL Project, CreditApplicationProcessorBusinessProcessor, is created and the WSDLs and XSDs for the following services are included: 1) Applicant Evaluator, 2) Vehicle Information Server, 3) Financial Index Server, and 4) Loan Approval Processor Service. Once the WSDLs are imported, Partner Link Types are created for the Port Type that will be invoked from the business process. Please note that the credit application process is going to be exposed as a Web service, so, another WSDL is created, CreditProcessor.wsdl, that represents the interface to the business process .

Partner Link Types specify the role that will be played by the service defined in the Port Type. For example, the Partner Link Type CustomerDataServicesPartners below specifies that the service will play the role of a CustomerReportProvider when the function getCustomerReport is invoked.

```
<partnerLinkType name="CustomerDataServicesPartner">
<role name="CustomerReportProvider"
portType="getCustomerReport"/>
</role>
</partnerLinkType>
```

Once Partner Links Types are created in the WSDLs for the PortTypes, they can now be invoked from a business process. The user can simply drag and drop the WSDL on the Business Process Editor canvas and the Business Process Editor automatically recognizes the available Partner Link Types and shows a wizard for configuring the Partner Links for the business process. The wizard allows the user to determine if the business process will be a consuming partner or a providing partner of the service defined in the Partner Link Type.

The WS-BPEL 2.0 specification provides various types of activities for modeling complex and real-world business processes. The business process can receive and reply to messages or it can just receive messages without responding. Business processes can also receive messages asynchronously from external sources. The key constructs supported in the WS-BPEL specification include the ability to invoke external services, handle exceptions, process compensation, and error condition logic. It also allows for modeling the concurrent and conditional execution of activities. The sample used in this article uses only a few of the constructs, namely, Receive, Reply, Invoke, and Assign activities. The Invoke activity is used to consume the services and the Assign activity is used to set and get values to the input and from the output messages of the invoked service. The Assign activity opens up a mapper that allows users to view all the variables in the business process and graphically get and set values between them. The mapper also provides data processing and transformation functions that

Gopalan Suresh Raj, an architect at Sun Microsystems, Inc., is a member of Sun's research and architecture team. His expertise spans enterprise architectures and distributed computing. He is the author of a number of technical books and articles. You can read and participate in discussions with him at http://blogs.sun.com/gopalan.

gopalan.raj@sun.com

can be used in conjunction with the assignments. The Receive activity is used to receive an incoming message that creates an instance of the business process at runtime. The reply activity is used to send a response message back to the caller and the business process instance is then discarded.

The user starts by creating a template business process by laying out the activities and then configuring them one at a time. The user drags and drops the CreditProcess WSDL and configures the business process as the providing partner of the service. The user configures the Receive and Reply activities to implement the PortType specified in the CreditProcess WSDL. The user then drags and drops all the other WSDLs and configures the business process as a consuming partner of those services. The user uses one invoke activity per service consumed and configures the invoke activity to link it with the appropriate Partner Link. Once configured, the business process looks similar to Figure 6.

Create and Deploy Composite Applications

NetBeans provides the capability to group more than one business process available under different projects into one composite application. This allows you to group multiple logical business processes and manage them as one logical unit during deployment. NetBeans also provides the necessary runtime components for deploying composite applications. The runtime components include Java Application Server 9.1, WS-BPEL Service Engine, and HTTP/SOAP Binding Components. The composite applications can be easily created by selecting and adding components from different projects in the environment. NetBeans also allows users to build and deploy composite applications with a single click.

In the sample used in this article, the user adds the business process module from the project CreditApplicationProcessorBusinessProcess to the Composite Application project. The composite application is then deployed at runtime.

Testing

The need to write a client application just to test a newly deployed composite application can be daunting. NetBeans eliminates this need by providing a powerful test facility. This facility is automatically added under the Composite Application projects. The users can create any number of test cases to test various functions of the deployed composite application and specify the success criteria for these tests. The input and output messages for the test cases are also automatically generated when the user selects the WSDL that corresponds to the business process to be tested. The user can then customize the messages as needed in the editor shown in Figure 7. The test facility also maintains a log of all the test results. In case of system errors, the server and application logs can be viewed from NetBeans. This facility allows users to immediately test just-deployed composite applications.

The user adds a test case for the application discussed in the article. The output message can be customized before the first run or the output message from the first run of the test case can be used as the expected output message.

Okay, What's Next?

Java EE 5 SDK provides the complete set of tools and environments required for composing services using WS-BPEL. Invoking all services as SOAP/HTTP Web services could bring out the obvious concern about performance. WS-BPEL can be used to directly invoke Enterprise JavaBeans to compose functionality from Java EE applications. Java EE 5 SDK can be used to model business processes that can invoke internal and external Web services and Enterprise JavaBeans directly, as shown in Figure 8.

Business processes can invoke the services available in those technologies including REST Style Web services. If you are interested in finding out what other options are available for composing services and processing

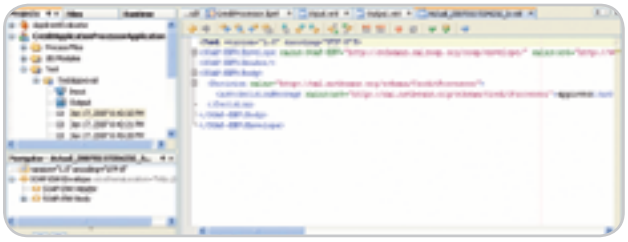

Figure 7 Composite Application Testing Facility

Figure 8 WS-BPEL invoking SOAP/HTTP Web services and Enterprise JavaBeans

business events, check the following open source projects.

- NetBeans: http://netbeans.org/
- GlassFish: http://glassfish.dev.java.net/
- Open ESB: http://open-esb.dev.java.net/

You can get the latest version of the products and updates on the enhancements that are currently being worked on. You can also join and contribute to these projects if you're interested.

The source code for the sample projects used in this article can be downloaded from the online version of this article at http://jdj.sys-con.com.

Summary

This article has discussed and shown the many capabilities in the Java EE 5 SDK Tools Bundle for delivering composite applications using Java, WS-BPEL, and SOA technologies.

First, this article discussed the benefits of these two technologies and how they can work together to enable the development of next-generation applications.

Second, this article demonstrated how NetBeans provides a truly integrated development environment that allows users to extend existing Java services, create new Java services, and compose Java services with a business process. This article has also shown how NetBeans can be used to iteratively develop, build, deploy, and test composite applications seamlessly, thereby reducing the overall turnaround time and effort required to deliver composite applications.

Resources

- Java EE 5 SDK: http://java.sun.com/javaee/downloads/
- NetBeans IDE & Enterprise Pack: http://www.netbeans.org/
- Project Open ESB: http://open-esb.dev.java.net/
- WS-BPEL 2.0 Specification: http://www.oasis-open.org/committees/download.php/14616/wsbpel-specification-draft.htm
- Technical Blog: http://blogs.sun.con/gopalan/
- Technical Blog: http://blogs.sun.com/kevinschmidt/
- Technical Blog: http://blogs.sun.com/theaquarium/