SOA using OpenESB, BPEL, JBI, GlassFish and NetBeans

Sang Shin
Java Technology Architect
Sun Microsystems, Inc.
javapassion.com
Agenda

- Composite Application
- BPEL
- JBI
- JBI and GlassFish
- OpenESB
- Java EE Service Engine
- Intelligent Event Processor
Composite Applications
Traditional Application Development

- Point technologies, products, and APIs
  > For example: EJB, Spring, Hibernate, JSF, Servlets, Struts, etc.

- Lots of glue written by developers
  > Requires a great deal of expertise & time
  > Inflexible
Composite Applications

• A way to compose applications from reusable parts

• Composite applications employ SOA principles
  > Features exposed as Web services
  > Standards-based interaction between services
  > Are themselves compose'able
WSDL Tutorial
(Optional Presentation)
Why WSDL?

• Enables **automation** of communication details between communicating partners
  – Machines can read WSDL
  – Machines can invoke a service defined in WSDL

• Discoverable through registry

• Arbitration
  – 3rd party can verify if communication conforms to WSDL
WSDL Document Example

- Simple service providing stock quotes
- A single operation called `GetLastTradePrice`
- Deployed using SOAP 1.1 over HTTP
- Request takes a ticker symbol of type `string`
- Response returns price as a `float`
WSDL Elements

- Types
- Message
- Operation
- Port Type
- Binding
- Port
- Service
WSDL Elements

• Types
  – Data type definitions
  – Used to describe exchanged messages
  – Uses W3C XML Schema as canonical type system
WSDL Example: Types

<definitions name="StockQuote"
    targetNamespace="http://example.com/stockquote.wsdl"
    xmlns:tns="http://example.com/stockquote.wsdl"
    xmlns:xsd="http://example.com/stockquote.xsd"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/
    xmlns="http://schemas.xmlsoap.org/wsdl/">
    <types>
        <schema targetNamespace="http://example.com/stockquote.xsd"
            xmlns="http://www.w3.org/2000/10/XMLSchema">
            <element name="TradePriceRequest">
                <complexType>
                    <all>
                        <element name="tickerSymbol" type="string"/>
                    </all>
                </complexType>
            </element>
            <element name="TradePrice">
                <complexType>
                    <all>
                        <element name="price" type="float"/>
                    </all>
                </complexType>
            </element>
        </schema>
    </types>
</definitions>
WSDL Elements

- **Messages**
  - Abstract, typed definitions of data being exchanged

- **Operations**
  - Abstract description of an action
  - Refers to an input and/or output messages

- **Port type**
  - Collection of operations
  - Abstract definition of a service
Example: Messages, Operation, Port type

<message name="GetLastTradePriceInput">
    <part name="body" element="xsd1:TradePriceRequest"/>
</message>

<message name="GetLastTradePriceOutput">
    <part name="body" element="xsd1:TradePrice"/>
</message>

<portType name="StockQuotePortType">
    <operation name="GetLastTradePrice">
        <input message="tns:GetLastTradePriceInput"/>
        <output message="tns:GetLastTradePriceOutput"/>
    </operation>
    <!-- More operations -->
</portType>
WSDL Elements

• Binding
  – Concrete protocol and data format (encoding) for a particular Port type
    • Protocol examples: SOAP 1.1 over HTTP or SOAP 1.1 over SMTP
    • Encoding examples: SOAP encoding, RDF encoding

• Port
  – Defines a single communication endpoint
    – Endpoint address for binding
    – URL for HTTP, email address for SMTP

• Service
  – Aggregate set of related ports
Example: Binding, Port, Service

```xml
<binding name="StockQuoteSoapBinding" type="tns:StockQuotePortType">
    <soap:binding style="document">
        transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="GetLastTradePrice">
        <soap:operation
            soapAction="http://example.com/GetLastTradePrice"/>
        <input> <soap:body use="literal" /> </input>
        <output> <soap:body use="literal" /> </output>
    </operation>
</binding>

<service name="StockQuoteService">
    <documentation>My first service</documentation>
    <port name="StockQuotePort" binding="tns:StockQuoteSoapBinding">
        <soap:address location="http://example.com/stockquote"/>
    </port>
</service>
```
BPEL
Need for Business Process

• Developing the web services and exposing the functionality (via WSDL) is not sufficient

• Example Scenario
  > Concert ticket purchase Web service has 3 operations, which need to be performed in the following order
    > Getting a price quote
    > Purchase a ticket
    > Confirmation and cancellation

• We also need a way to orchestrate these functionality in the right order
BPEL Works With WSDL

• Web services are described in WSDL

• We need a way to orchestrate these operations with multiple web services in the right order to perform a Business process
  > Sequencing, conditional behavior etc.

• BPEL provides standard-based orchestration of these operations
BPEL: Relationship to Partners

Orchestrating Process (BPEL)

WSDL

Partner Service

Partner Service

Partner Service

Partner Service
Business Process Needs To...

- Co-ordinate asynchronous communication between services
- Correlate message exchanges between parties
- Implement parallel processing of activities
- Implement compensation logic (Undo operations)
- Manipulate/transform data between partner interactions
- Support for long running business transactions and activities
- Handle exception handling
BPEL Document Structure

<process>
  <!-- Definition and roles of process participants -->
  <partnerLinks> ... </partnerLinks>
  <!-- Data/state used within the process -->
  <variables> ... </variables>
  <!-- Properties that enable conversations -->
  <correlationSets> ... </correlationSets>
  <!-- Exception handling -->
  <faultHandlers> ... </faultHandlers>
  <!-- Error recovery – undoing actions -->
  <compensationHandlers> ... </compensationHandlers>
  <!-- Concurrent events with process itself -->
  <eventHandlers> ... </eventHandlers>
  <!-- Business process flow -->
  (activities)*
</process>
# BPEL Activities

<table>
<thead>
<tr>
<th>Basic Activities</th>
<th>Structured Activities</th>
</tr>
</thead>
<tbody>
<tr>
<td>• &lt;invoke&gt;</td>
<td>• &lt;if&gt;</td>
</tr>
<tr>
<td>• &lt;receive&gt;</td>
<td>• &lt;while&gt;</td>
</tr>
<tr>
<td>• &lt;reply&gt;</td>
<td>• &lt;repeatUntil&gt;</td>
</tr>
<tr>
<td>• &lt;assign&gt;</td>
<td>• &lt;foreach&gt;</td>
</tr>
<tr>
<td>• &lt;throw&gt;</td>
<td>• &lt;pick&gt;</td>
</tr>
<tr>
<td>• &lt;wait&gt;</td>
<td>• &lt;flow&gt;</td>
</tr>
<tr>
<td>• &lt;empty&gt;</td>
<td>• &lt;sequence&gt;</td>
</tr>
<tr>
<td>• &lt;exit&gt;</td>
<td>• &lt;scope&gt;</td>
</tr>
</tbody>
</table>
BPEL: Basic Activities

• *<invoke>*
  > To invoke a one-way or request/response operation on a portType offered by a partner

• *<receive>*
  > To do a blocking wait for a matching message to arrive
  > Can be the instantiator of the business process

• *<reply>*
  > To send a message in reply to a message that was received through a *<receive>*
  > The combination of a *<receive>* and a *<reply>* forms a request-response operation on the WSDL portType for the process
BPEL: Structured Activities

- `<sequence>`
  - Perform activities in sequential order
- `<flow>`
  - Perform activities in parallel
- `<if>`
  - Conditional choice of activities
- `<scope>`
  - Enclose multiple activities in a single scope
Example Business Process

Invoke <InventoryService>

Receive <PO>

Invoke <CreditService>

Reply <Invoice>
Sample Activities in BPEL

```xml
<sequence>
  <receive partnerLink="customer" portType="lns:purchaseOrderPT"
            operation="sendPurchaseOrder" variable="PO"
            createInstance="yes"/>

  <flow>
    <invoke partnerLink="inventoryChecker" portType="lns:inventoryPT"
             operation="checkINV" inputVariable="inventoryRequest"
             outputVariable="inventoryResponse"/>

    <invoke partnerLink="creditChecker" portType="lns:creditPT"
             operation="checkCRED" inputVariable="creditRequest"
             outputVariable="creditResponse"/>
  </flow>

  ... 

  <reply partnerLink="customer" portType="lns:purchaseOrderPT"
          operation="sendPurchaseOrder" variable="invoice"/>
</sequence>
```
BPEL: Relationship to Partners

- Customer Service
- Orchestration Process (BPEL)
- Inventory Checker Service
- Credit Checker Service
- Another Partner Service

WSDL
Why Do You Care on BPEL?

• In SOA-enabled environment, you are more likely to build an application by orchestration various services via BPEL.

• You will probably use BPEL design tool to create a BPEL document.

• The BPEL document is then executed by BPEL engine.
  > Highly likely in JBI enabled platform.
Demo: Building Travel Reservation Composite Application (Demo Scenario Next Slide)

http://www.netbeans.org/kb/60/ep-understand-trs.html
Demo Scenario: Travel Reservation Business Process

- It receives travel reservation request from its client
  - The request contains travel reservation request XML document based on OTA (Open Travel Association)
- It then performs travel reservation business process talking to three partner web services
  - Airline reservation partner web service
  - Hotel reservation partner web service
  - Vehicle reservation partner web service
- The three partner web services are implemented as EJB based web services
Demo Scenario

• See Travel Reservation business process as a BPEL document
• See WSDL documents of partner web services and of the BPEL process web service
• Build and deploy the application over GlassFish and JBI server
• Test the application with test requests
• Perform source-code debugging on BPEL
Services and SOA
What Are Services?

- **Black-box components with well-defined interfaces**
  - Performs some arbitrary function
  - Can be implemented in myriad ways

- **Accessed using XML message exchanges**
  - Using well-known message exchange patterns (MEPs)

- **Metadata in the form of WSDL describes**...
  - Abstract interfaces
  - Concrete endpoints
What Can Services Do?

• Perform business logic
• Transform data
• Route messages
• Query databases
• Apply business policy
• Handle business exceptions
• Prepare information for use by a user interface
• Orchestrate conversations between multiple services
• ...
How Are Services Implemented?

- Enterprise JavaBeans™ (EJB™) technology
- BPEL
- XSLT
- SQL
- Business rules
- Mainframe transaction
- EDI transform
- Humans (yes, really!)
- ...
Example: Purchase Service

- Buyer
  - Bid Request
  - Lowest Bid
  - Accept/Reject
  - Ship Notice

- Purchase Service

- Bid Request
- Bid
- Accept/Reject
- Ship Notice

- Supplier
Purchase Service Functions

- Buyer Endpoint
- Supplier Endpoint
- Buyer Conversation
- Supplier Conversation
- Transaction Fees
- Supplier Selection
- Buyer Credit
- Supplier Routing
- Product Conversion
Purchase Service Functions

- Buyer Endpoint
- Supplier Endpoint
- Buyer Conversation
- Supplier Conversation
- Transaction Fees
- Supplier Selection
- Rule
- EJB
- XQuery
- XSLT
- WSDL/Soap
- BPEL
- Routing Table
- Rule
- Credit

Select Buyer Conversation
Supplier Conversation
Service Oriented Architecture (SOA)

- An architectural principle for structuring systems into coarse-grained services
- Technology-neutral best practice
- Emphasizes the loose coupling of services
- New services are created from existing ones in a synergistic fashion
- Strong service definitions are critical
- Services can be re-composed when business requirements change
Service Implementation over JBI
What Is JBI?

- JBI provides a standard application integration framework
- JBI to Application integration is what Java EE is to Enterprise application
Why JBI?

• Point-to-point integration model is not scalable and hard to maintain

• The traditional EAI model has its problems
  > Proprietary integration server
  > Vendor lock-in
  > High barrier for entry for small, independent, innovative ISV's providing best-of-breed solutions

• There is a need for an open standard framework for application integration
What Is JBI?

- **Standard “meta-container” for integrating “service containers”**
  - Service containers can host any services (service units)
    - Business logic service
    - System services
  - Service can be located locally or remotely
- **Plug-in architecture**
  - Service Engines (SE) – Local service or consumer
  - Binding Components – Remote service or consumer
Service Provider Self-Description

Service containers register the services they provide.
Java Business Integration (JSR 208)
OpenESB
What is Project OpenESB?

• Project OpenESB implements an Enterprise Service Bus (ESB) runtime using Java Business Integration (JBI) as the foundation
  > This allows easy integration of web services to create loosely coupled enterprise class composite applications.

• It also provides various tools for the development, deployment, and management of composite applications
OpenESB Architecture
IDE
- Composite Application Project
- BPEL Editor
- XSLT Editor
- IEP Editor
- Many More Editors

Web Server
- Composite Application Manager
- BPEL Monitor
- XSLT Monitor
- IEP Monitor
- Many More Monitors

Design-Time

Runtime

Management

App Server
- Java EE
- BPEL SE
- XSLT SE
- Many More SEs...
- HTTP BC
- FTP BC
- Many More BCs...

JBI Bus

Open Standard Based Service Bus
- WS-Reliable Messaging
- WS-Security
- WS-FastInfoSet, ...

3rd Party Service Platforms
JBI and GlassFish
JBI Support in GlassFish

- A JBI runtime has been integrated with GlassFish V2
- GlassFish admin console now supports JBI
- Java EE Service Engine act as the bridge between Java EE applications and JBI
- A Java EE application archive (ear/war/jar) can be packaged in a JBI composite application
- JBI runtime has been enhanced to adhere to the appserver clustering architecture
  > Each instance in the app server cluster will also have a JBI runtime in it
JBI in Admin Console

Sun Java System Application Server 9.1 Admin Console - Mozilla Firefox

Getting Started  Latest Headlines

Users: admin  Domains: domain1  Server: localhost

Sun Java™ System Application Server Admin Console

JB1 > Components > sun-http-binding

General  Configuration  Descriptor  Loggers  Monitoring  Libraries

sun-http-binding - Statistics

View and verify message exchange throughput statistics.

Binding Component Statistics (1)

Endpoints  Received Requests  Received Replies  Received Errors  Received Dones  Sent Requests  Sent Replies  Sent Errors  Sent Dones

Providing Endpoints
No items found.

Consuming Endpoints
No items found.

Totals

0 0 0 0 0 0 0 0 0
Usage Scenario
Usage Scenario: Loan Processing

- Loan Requestor Service:
  - LoanRequestProcess
  - WS-I BP
  - BPEL Orchestration
  - LoanProcessor
    - JavaEE
    - TransformReport
      - XSLT
      - LoanReportStore
        - Business Partner thru FTP
        - LoanReportMailer
          - Legacy thru JMS
JBI-based Infrastructure

- BPEL
- JavaEE
- XSLT

NMR

- WS-I BP
- JMS
- File

File server

Client machine
NMR Loan Request Process
Transform Report
JavaEE Loan Processor EJB
XSLT
BPEL
WS-I BP
LoanRS WS
JMS
ReportMail
File
ReportStore
Architecture Refactoring

BPEL Loan Request Service

XSLT Transform Report

JavaEE Loan Processor EJB

NMR

WS-I BP LoanRS WS

JMS ReportMail

File ReportStore

---

BPEL
Loan Request Service

XSLT Transform Report

JavaEE Loan Processor EJB

NMR

WS-I BP LoanRS WS

JMS ReportMail

File ReportStore
Service Engines (SE) & Binding Components (BC)
JBI Components

- **Service Engines**
  - BPEL SE
  - XSLT SE
  - JavaEE SE
  - IEP SE
  - ETL SE
  - SQL SE
  - Workflow SE

- **Binding Comps**
  - MQSeries BC
  - HL7 BC
  - SAP BC
  - SMTP BC
  - HTTP BC
  - JMS BC
  - File BC
  - CICS BC
  - DCOM BC
  - CORBA BC
  - ...

- **Other**
  - Clustering
  - CASA
  - JBI Mock
  - WSIT Tech

- **In Progress**
  - CAM
  - Aspect SE
  - Encoding SE
  - Rules SE
  - Scripting SE
Open ESB Components

The following list of components are currently available at the Open JBI Components project. In this page you can find a component by their function, their solution or by their name. Clicking on a component link will provide details on how they are built, what their current capabilities are and the future plans around these components.

Learn More about Open JBI Components Project
View the JBI Community Wiki's components

By Function

Application Mashup
AOD

Aspects
Aspect SE

Communications
These components provide different methods of communicating with other components of the Enterprise Service Bus.
ADABAS Natural
CICS BC
CORBA BC
DCOM BC
File BC
FTP BC
HTTP BC
SIP BC
XMPP BC

By Solution

Healthcare
These components provide solutions for the healthcare industry.
HL7 BC

Mainframe
These components provide mainframe solutions to integration.
CICS BC
JMS BC

Telecommunications
These components provide solutions for the telecommunications industry.
CORBA BC
SIP BC
XMPP BC

By Name

Alphabetical Listing
If you know the name of the project you are looking for, find it here.

Applications
Mural

Binding Components
CICS BC
CORBA BC
DCOM BC
eMail BC
File BC
FTP BC
HTTP BC
JMS BC
JDBC BC
LDAP BC
MO Series BC
MMS BC
SOAP BC
SIP BC
SMTP BC
SNMP BC
SWIFT BC
TCP/IP BC
UDDI BC
XMPP BC

Data Conversion
Translating different data formats among systems with varying requirements.
ETL SE
XSLT SE
NetBeans Support of OpenESB
Types of SOA “NetBeans” Projects

- When creating a composite application, you typically use the following types of SOA “NetBeans” projects:
  - BPEL Module project
  - XSLT Module project
  - SQL Module project
  - Composite Application project
  - IEP Module project
  - Worklist Module project
  - ETL (Extract, Transform, and Load)
  - EDM (Enterprise Data Mashup)
  - And more
Types of SOA “NetBeans” Projects

New Project

Steps
1. Choose Project
2. ...

Choose Project

Categories:
- JBI Components
- JavaFX
- Java
- Web
- Enterprise
- Mobility
- UML
- Service Oriented Architecture
- Ruby
- NetBeans Plugin Modules
- Samples

Projects:
- Aspect Module
- EDM Module
- ETL Module
- Intelligent Event Processing Module
- Composite Application
- BPEL Module
- SQL Module
- Worklist Module
- XSLT Module

Description:
Creates an empty SCRIPT Module project.
BPEL Module Project

- BPEL Module project is a group of source files which includes:
  - XML Schema (*.xsd) files
  - WSDL files
  - BPEL files

- Within a BPEL Module project, you can author a business process compliant with the WS-BPEL 2.0 language specification.

- Will be added to a Composite application as a JBI module
Composite Application Project

• Composite Application project is a project whose primary purpose is to assemble a deployment unit for the Java Business Integration (JBI) server
  > BPEL Module projects must be added to a Composite Application project in order to be deployed to the BPEL runtime.

• The Composite Application Project can also be used to create and execute test cases that can then be run, in JUnit fashion, against the deployed BPEL processes.
With a Composite Application project, you can:

- Assemble an application that uses multiple project types (BPEL, XSLT, IEP, SQL, etc.)
- Configure external/edge access protocols (SOAP, JMS, SMTP, and others)
- Build JBI deployment packages
- Deploy the application image to the target JBI server
- Monitor the status of JBI server components and applications
NetBeans 6.0 Preview, SOA Documentation

May 2007

This page lists all new and updated documentation available for the SOA functionality in NetBeans 6.0 Preview (M9).
For more information about the runtime parts of SOA, see Open ESB.

For additional information about this Preview, see NetBeans IDE 6.0 Preview Release Information and NetBeans IDE 6.0 Preview Documentation.

Composite Applications
- Creating a "Hello World" Composite Application
- Creating a Loan Processing Composite Application
- Creating a Simple SOA Application With NetBeans

Business Process Execution Language (BPEL)
- Developer Guide to the BPEL Designer
- Understanding the Travel Reservation Service
- Developing a Simple Asynchronous BPEL Process
- Developing a Simple Synchronous BPEL Process
- Using Correlation Sets, Properties, and Property Aliases in BPEL
- XPath Functions and Operations in the BPEL Mapper

WSDL Editor
- Developer Guide to the WSDL Editor
- Introducing the WSDL Editor's Partner View

XML Schema Tools
- Getting Started With XML Schema Tools

XSLT Designer
- XSLT Designer Quick Start Guide
- Working With a Service Bridge XSL

JBI Component Technical Overview

Binding Component and Service Engine Tutorials
- Understanding the HTTP Binding Component
  Illustrates HTTP BC and BPEL SE.
- Using the JMS Binding Component in a
  Stock Alert Purchase Composite Application
  Illustrates JMS BC, File BC, HTTP BC, and
  BPEL SE.
- Using a Manually Created WSDL as a Web Service Client with the Java EE Service Engine
  Illustrates how to use a manually created WSDL as a web service client.
- Using a Manually Created WSDL Derived From Java With the Java EE Service Engine
  Illustrates how to implement a WSDL derived from Java.
- Using a Manually Created WSDL in an EJB
  with the Java EE Service Engine
  Illustrates how to implement a manually created WSDL in an EJB.
- Understanding the File Binding Component
  Illustrates File BC and Java EE SE.
- Understanding the FTP Binding Component
  Illustrates FTP BC, File BC, and BPEL SE.
- Understanding the JDBC Binding Component
  Illustrates JDBC BC, SMTP BC, and BPEL SE.
- Understanding the SMTP Binding Component
  Illustrates SMTP BC and BPEL SE.

Learning Trails
- Basic Java Programming
- Java GUIs and Project Malise
- Web Applications
- Java EE Applications
- Mobile Applications
- SOA Applications
- UML Modelling
- NetBeans Modules and Rich-Client Applications

Docs for Packs
- Mobility Pack
- Visual Web Pack
- Enterprise Pack
- C/C++ Pack
- Ruby Pack
- Profiler
- UML Module

Additional Resources
- FAQs
- Sample Applications
- Support
- Training
- NetBeans 5.0 Docs
- NetBeans 5.5 Docs
- NetBeans 6.0 Docs
- Contribute Documentation

Find: table out
Next Previous Highlight all Match case

Done
Demo: Building "Hello World" Composite Application

www.javapassion.com/handsonlabs/wscompositeapps/
Java EE SE
JavaEE SE

• Ideal place to execute complex business logic
• Bridge between JavaEE container and JBI container
• Provides support for
  > Transactions
  > Resource Pooling
  > Security
• Code re-use – Invoke your EJBs/web applications from OpenESB components (BPEL SE)
• Ability to expose your EJB/Web applications to multiple transports (using BCs) – just add bindings to your WSDL
Scenario 1: Remote through HTTP BC

- BPEL
  - NMR
  - HTTP BC
  - *Marshall DOM to `<xml>`*
  - SOAP/HTTP
  - Network layer

**Diagram:**

- JBI Container
  - HTTP BC
  - HL7 BC
  - SAP BC
  - BPEL SE
    - BPEL process
  - EE SE
    - ejb.jar
    - application.ear

**Flow:**

1. Network layer
2. SOAP/HTTP
3. *Unmarshal `<xml>`*
4. Create DOM
5. JAXB
6. WS.helloWorld(name)
Scenario 2: Local through NMR

- BPEL ➔ NMR ➔ HTTP BC ➔ Marshall DOM to <xml> ➔ SOAP/HTTP ➔ Network layer ➔ SOAP/HTTP ➔ JAXWS ➔ Unmarshal <xml> ➔ Create DOM ➔ JAXB ➔ WS.helloWorld(name)

Advantages:
- Performance
- Transaction propagation
- Security context propagation

Likewise: EJB to BPEL
Java EE Service Engine: Functions as Bridge between App Server and JBI Runtime Env.
Java EE S.E. As a Bridge
Demo: Building “Loan Processing” Composite Application

www.javapassion.com/handsonlabs/wscompositeapps/#Exercise_2
www.javapassion.com/handsonlabs/wscompositeapps/#Exercise_3
Intelligent Event Processor
Real-life Use Cases of IEP

• How many times did Fred login as root for the last 24 hours?
• How many times is “purchase order” business process triggered for the last hour? And what is the average execution time?
• Is a credit card charged for gasoline twice within last one hour?
• Raise an alert when a stock price jump more than 10% relative to its 1 minute moving average price.
• Is the number of JMS messages in the broker increasing over time? What changed?
Real Time Event Collection and Notification

External Service Consumer / Provider

Binding Component

Other Service Engines

Normalized Message Router

IEP Service Engine

JBI Container
Demo: Building IEP Application that Captures Incoming Stock Data Stream
Summary
Summary

• SOA enables flexible and agile enterprise application architecture

• Services can be created and used using Java EE

• BPEL is a service orchestration language for creating composite applications

• Services can be re-implemented using other technologies as long as service interfaces are preserved without changing consumers

• Java Business Integration (JBI) is the enabling infrastructure
SOA using OpenESB, BPEL, JBI, GlassFish and NetBeans

Sang Shin
Java Technology Architect
Sun Microsystems, Inc.
javapassion.com