SOA using OpenESB, BPEL, JBI, GlassFish and NetBeans

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

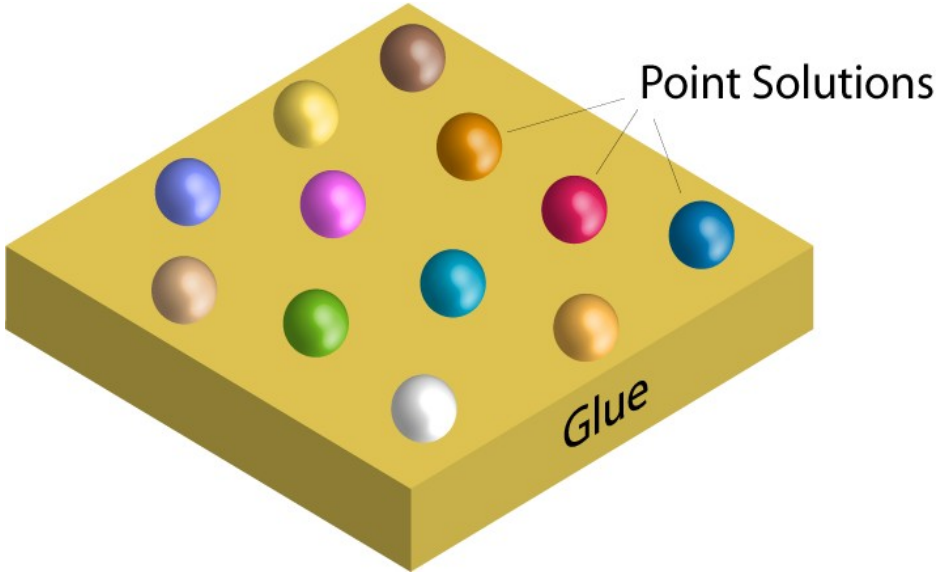
Agenda

- Composite Applications
- BPEL
- JBI
- JBI and GlassFish
- Java EE Service Engine
- Open ESB
- Open ESB runtime, tools, and sample apps

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:xsd1="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>
```

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

```
<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

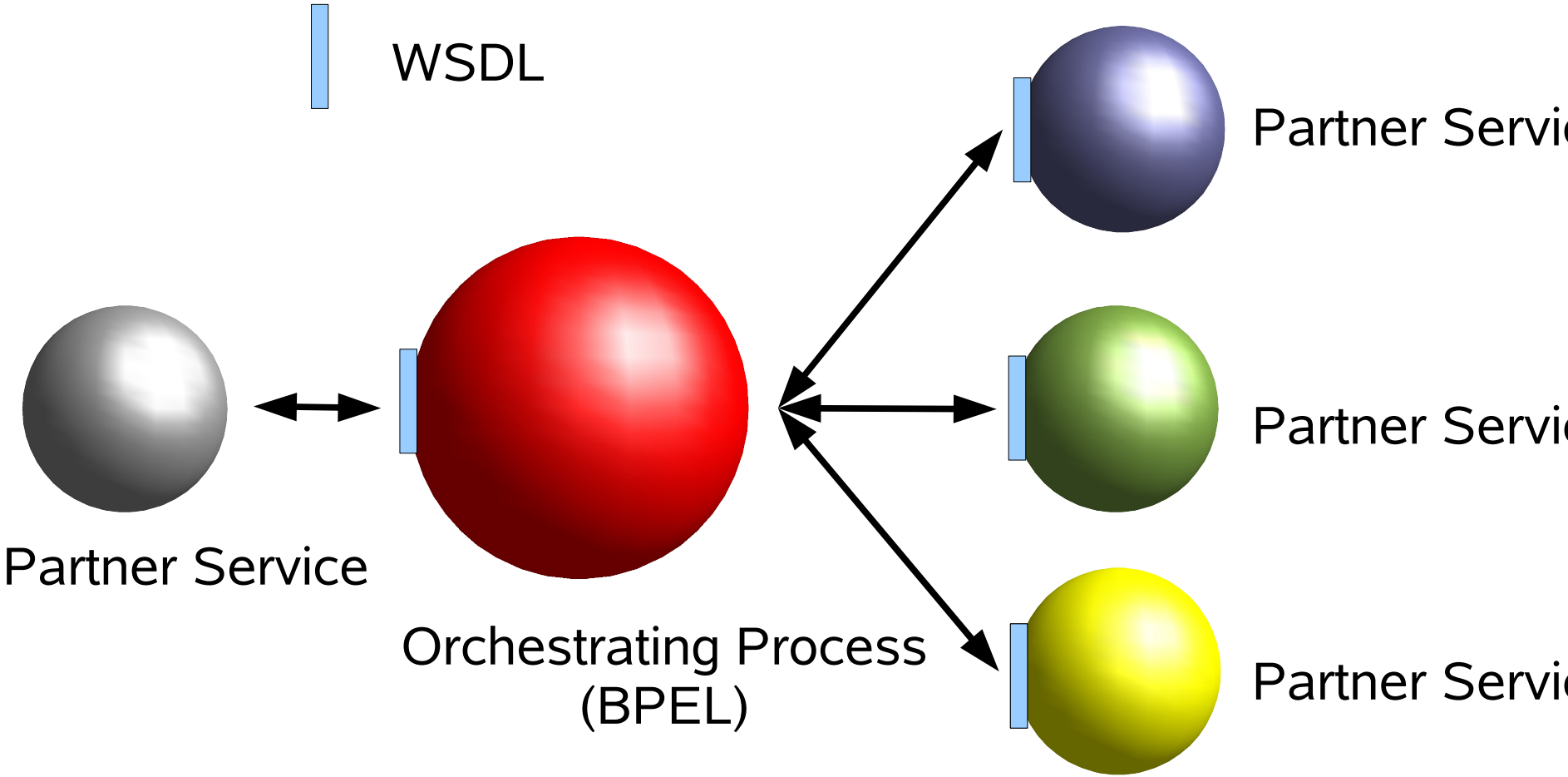
Need for Business Process

- 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



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
- Need for universal data model for message exchange

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

Basic Activities

- `<invoke>`
- `<receive>`
- `<reply>`
- `<assign>`
- `<throw>`
- `<wait>`
- `<empty>`
- `<exit>`

Structured Activities

- `<if>`
- `<while>`
- `<repeatUntil>`
- `<foreach>`
- `<pick>`
- `<flow>`
- `<sequence>`
- `<scope>`

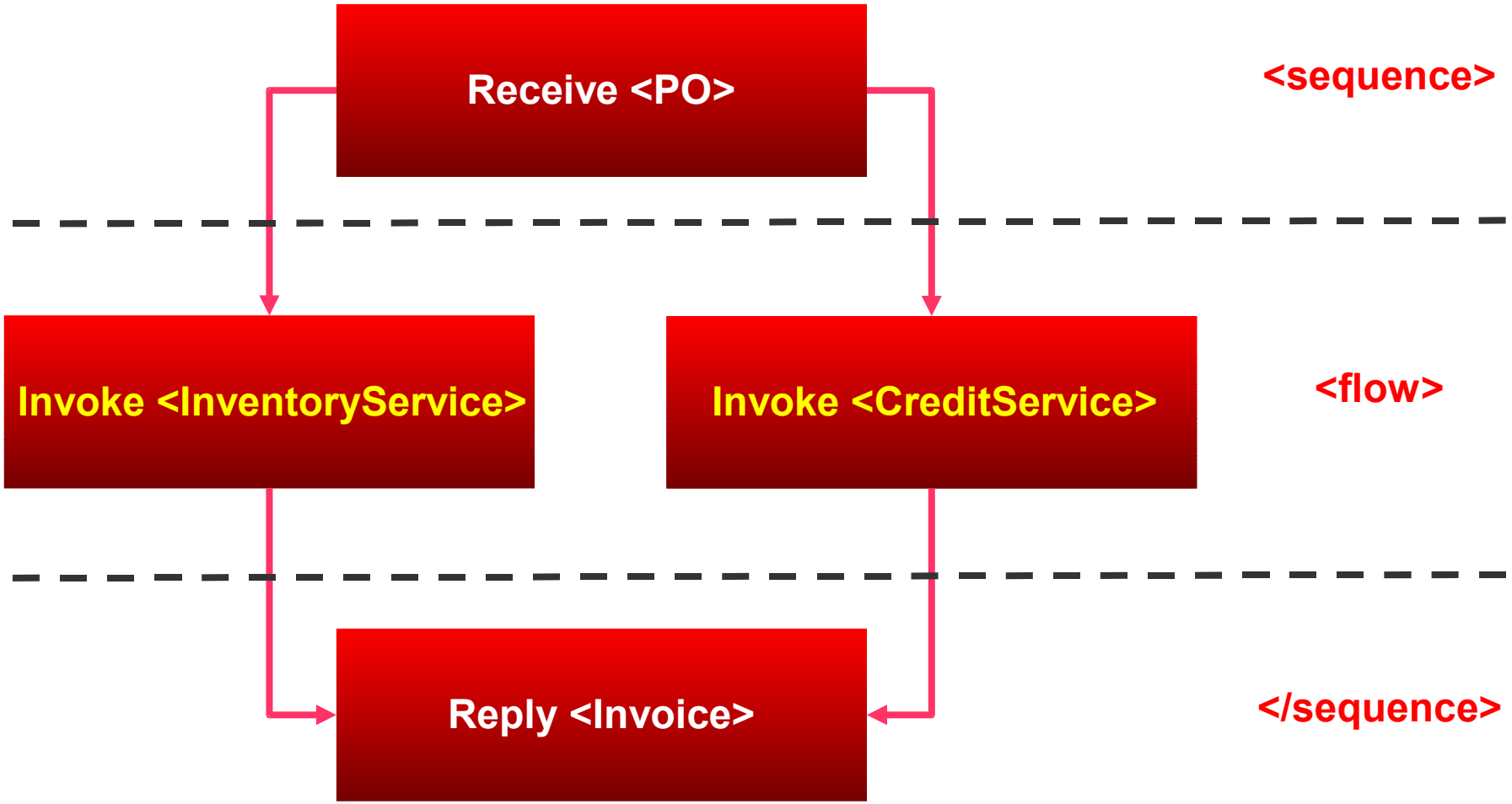
BPEL: Basic Activities

- <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

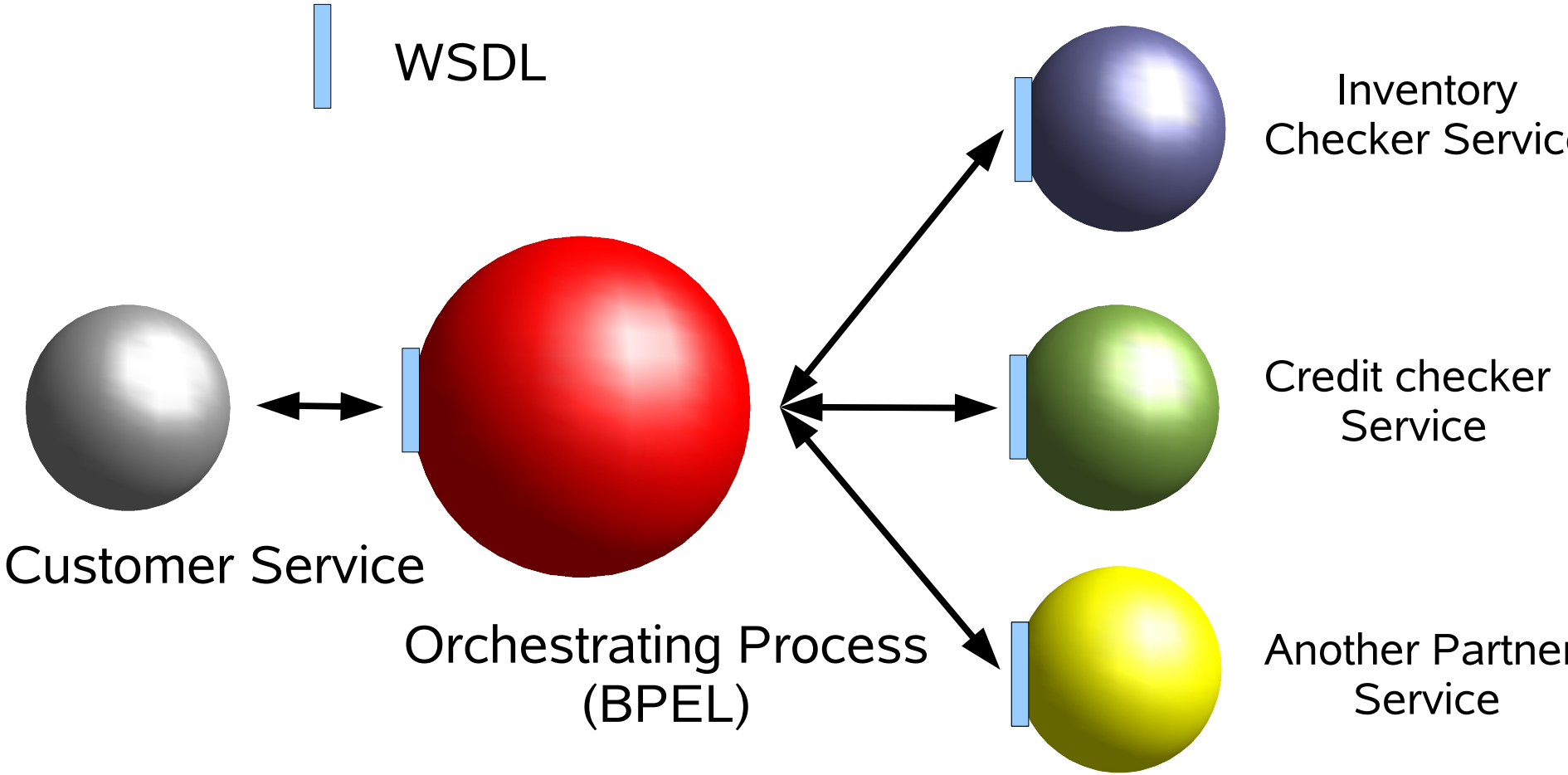


Sample Activities in BPEL

```
<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



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

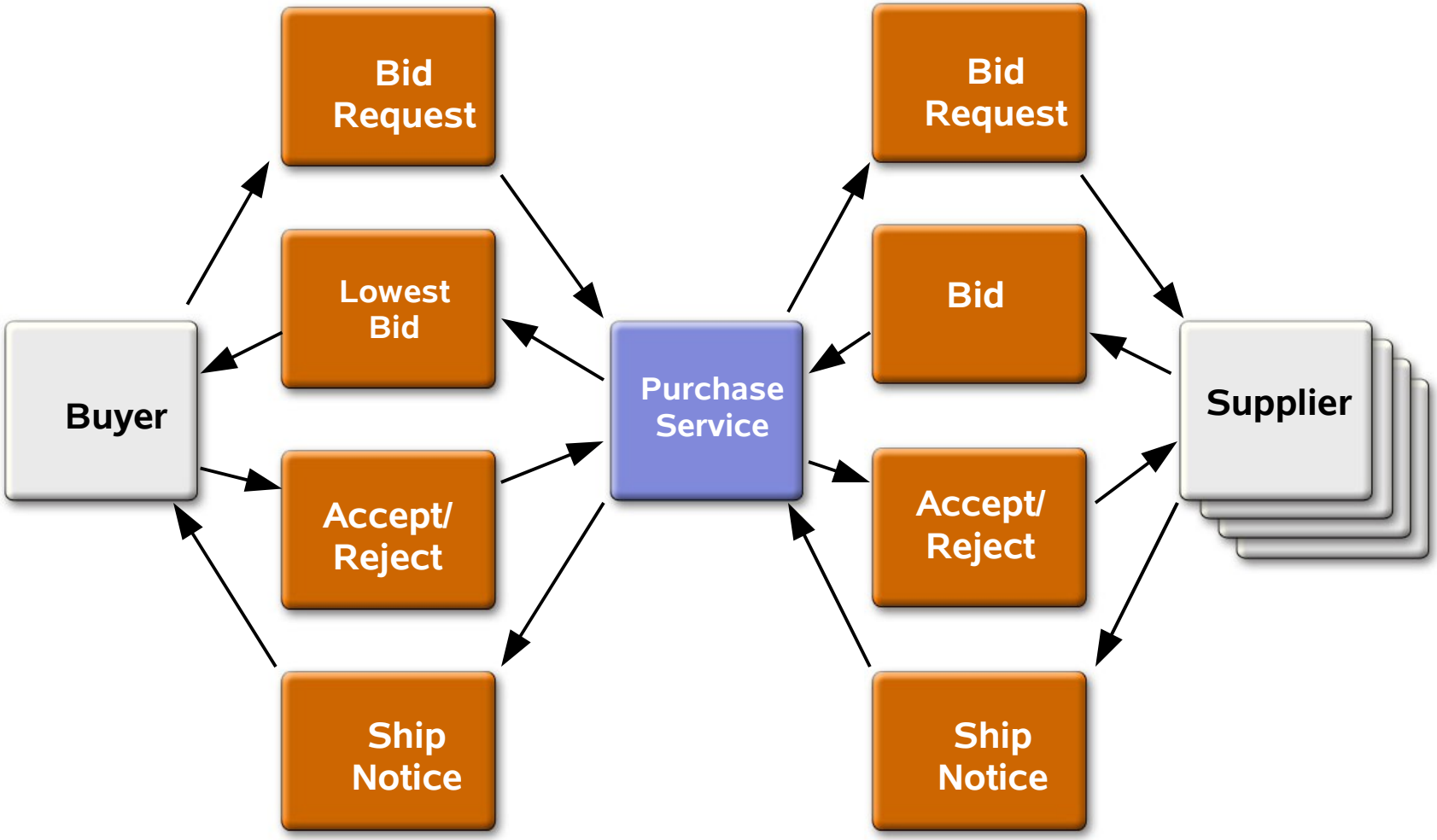
What Can Services Do?

- 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
- …

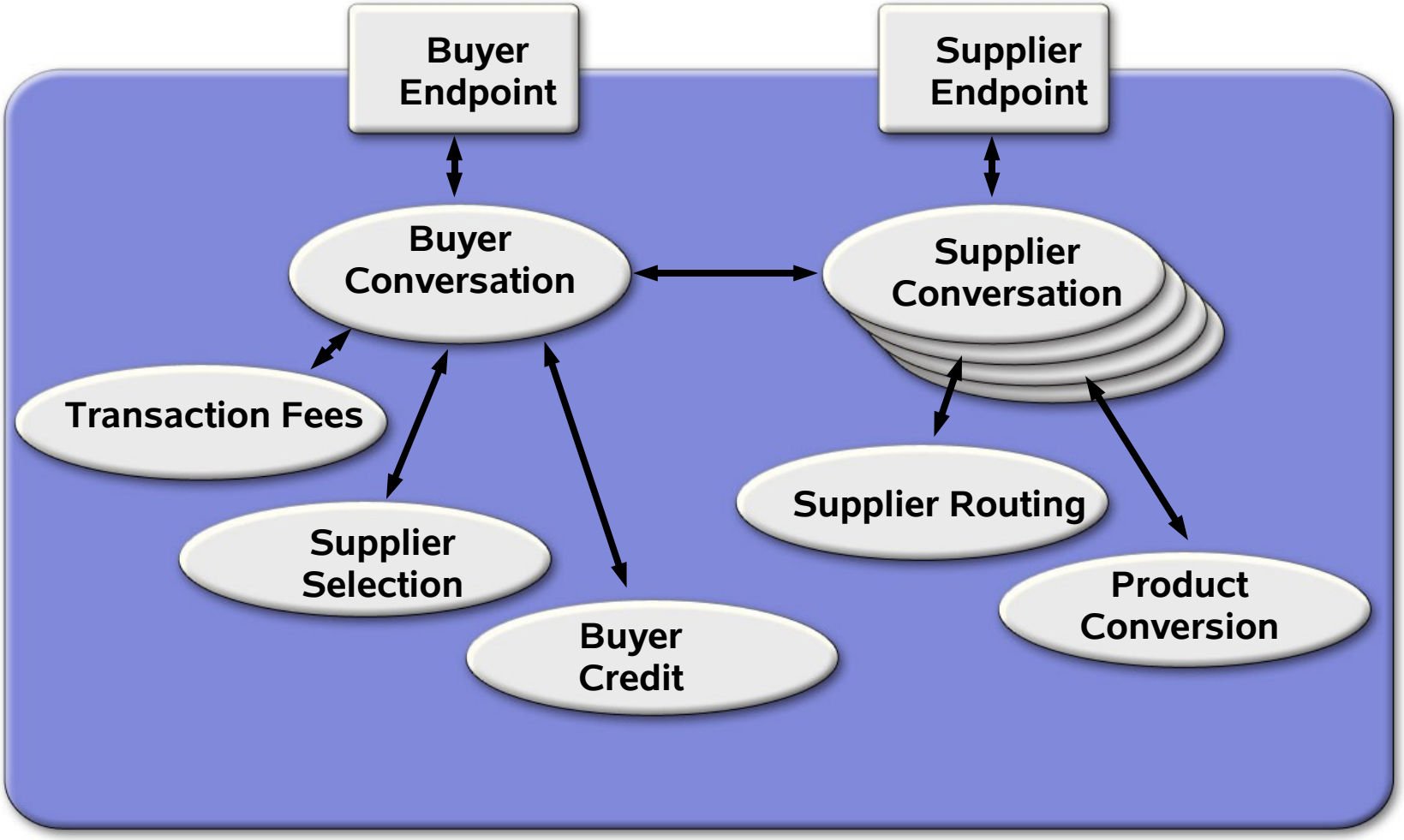
How Are Services Implemented?

- Enterprise JavaBeans™ (EJB™) technology
- BPEL
- XSLT
- SQL
- Business rules
- Mainframe transaction
- EDI transform
- Humans (yes, really!)
- …

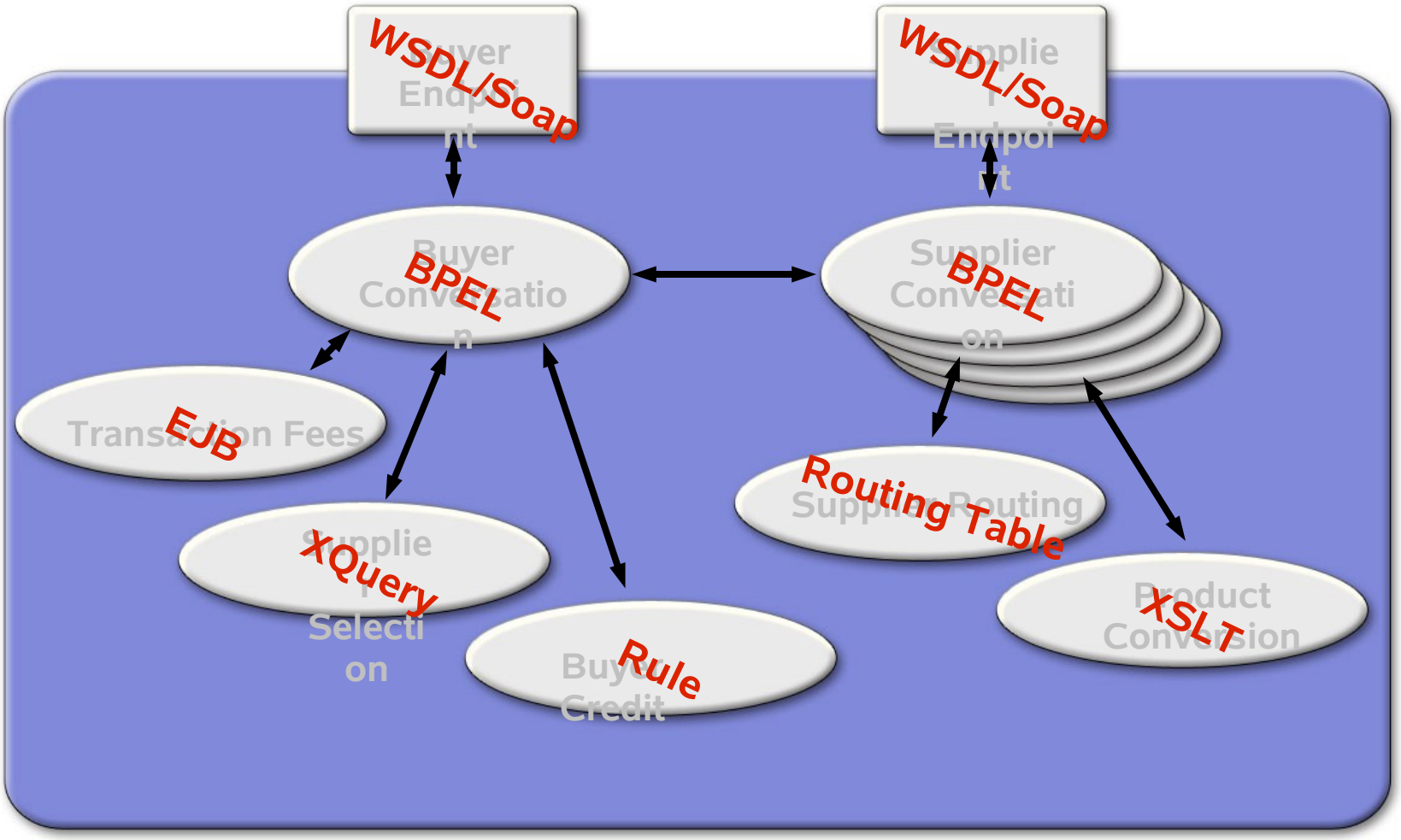
Example: Purchase Service



Purchase Service Functions



Purchase Service Functions

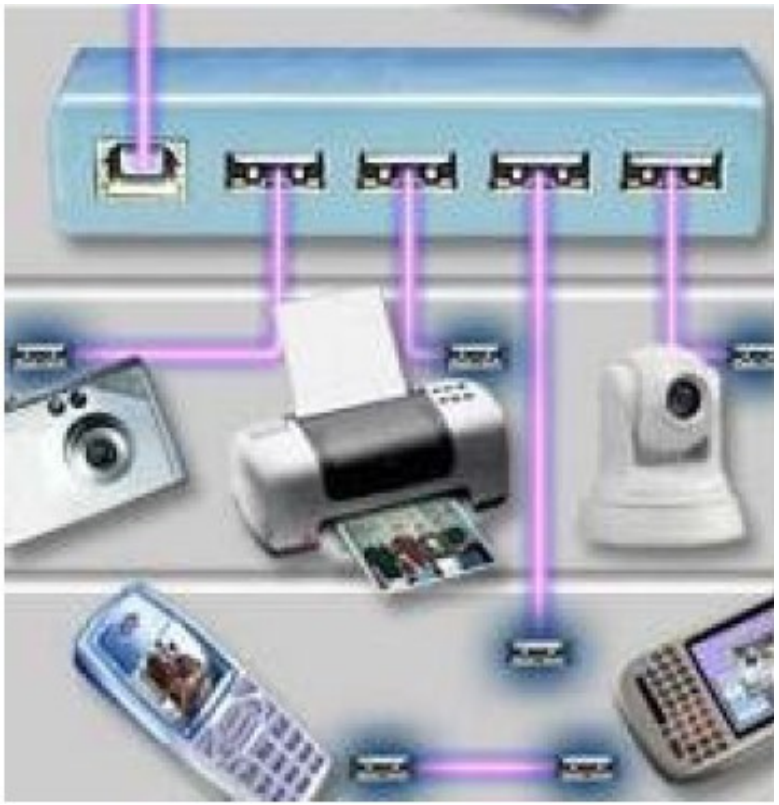


Service Oriented Architecture (SOA)

- 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?

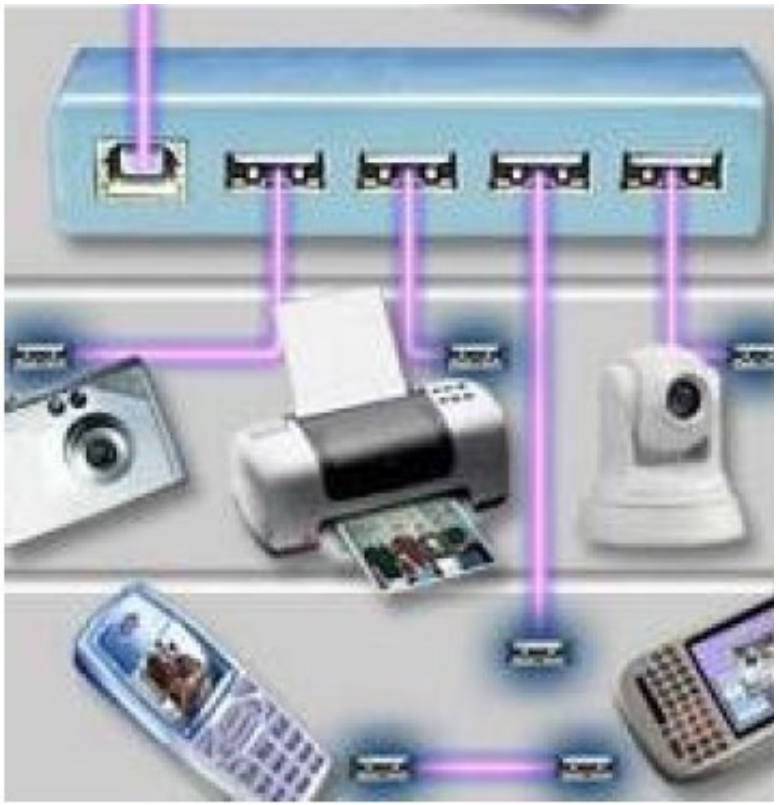


- JBI provides a standard application integration framework
- JBI to Application integration is what Java EE is to Enterprise application

Why JBI?

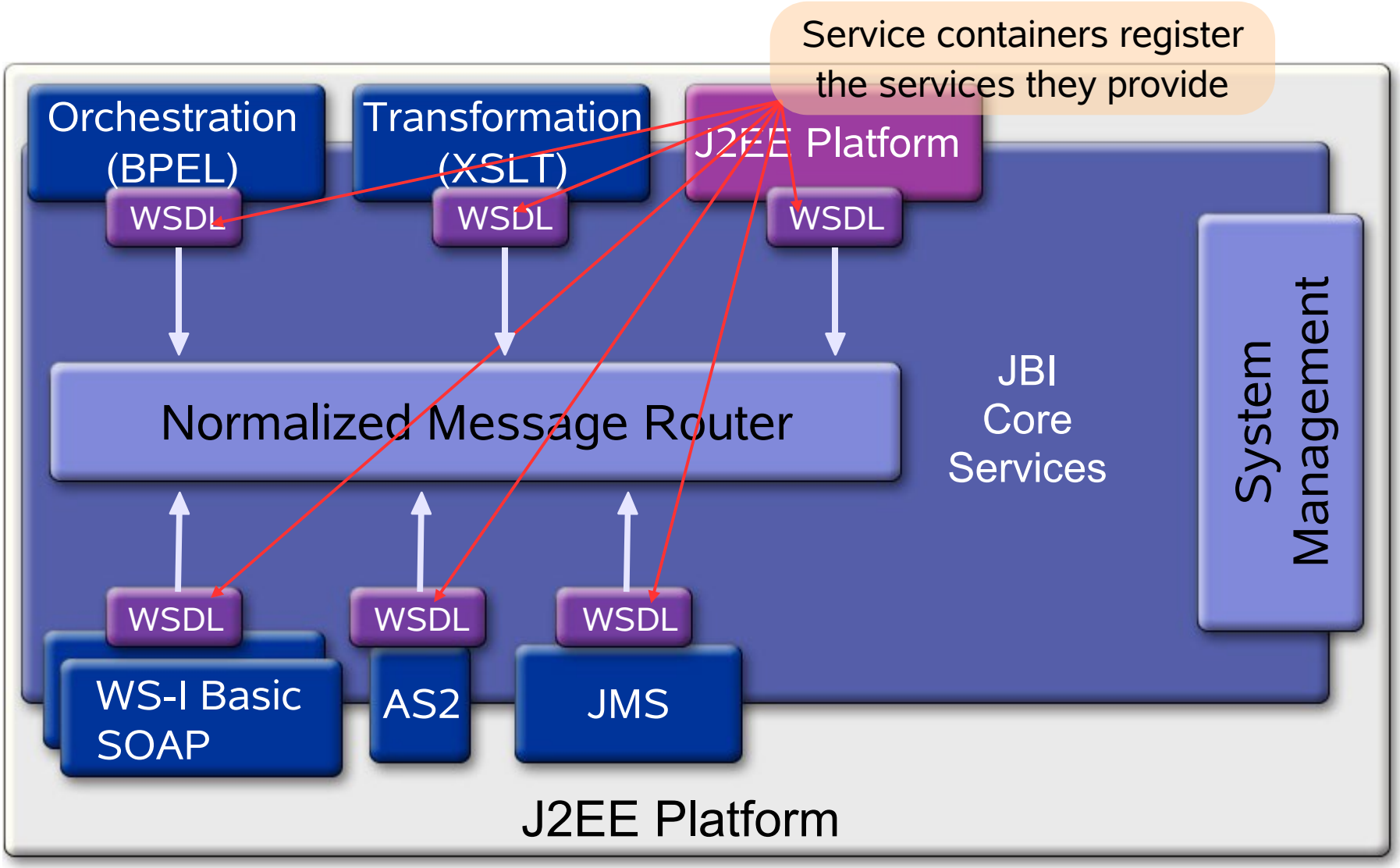
- Point-to-point integration model is not scalable and hard to maintain
- The traditional EAI model has its problems
  - > Proprietary integration server
  - > Vendor lock-in
  - > High barrier for entry for small, independent, innovative ISV's providing best-of-breed solutions
- There is a need for an open standard framework for application integration

What Is JBI?

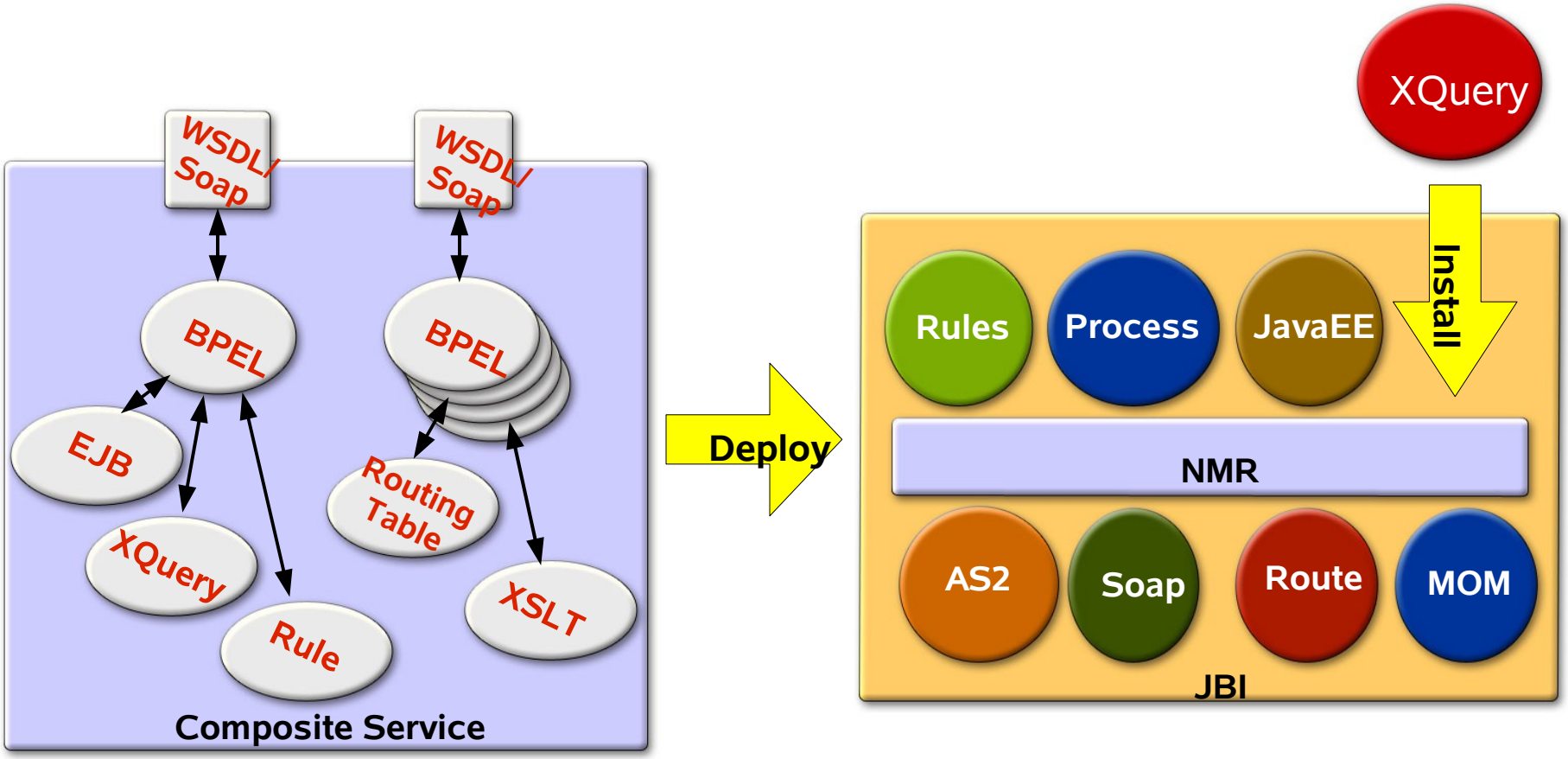


- Standard “meta-container” for integrating “service containers”
  - Service containers can host any services (service units)
    - Business logic service
    - System services
  - Service can be located locally or remotely
- Plug-in architecture
  - Service Engines (SE) – Local service or consumer
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

Service Provider Self-Description



Java Business Integration (JSR 208)

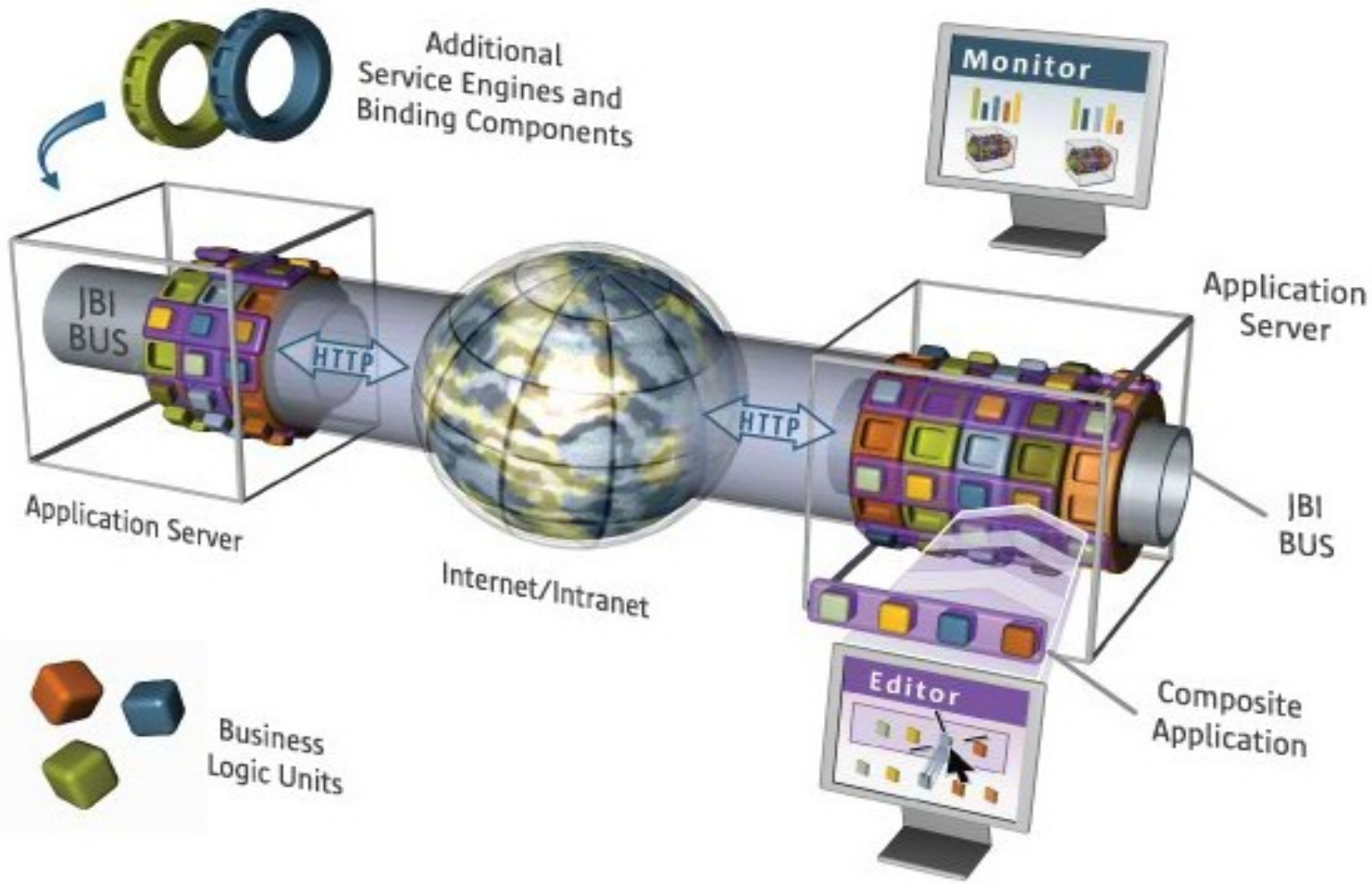


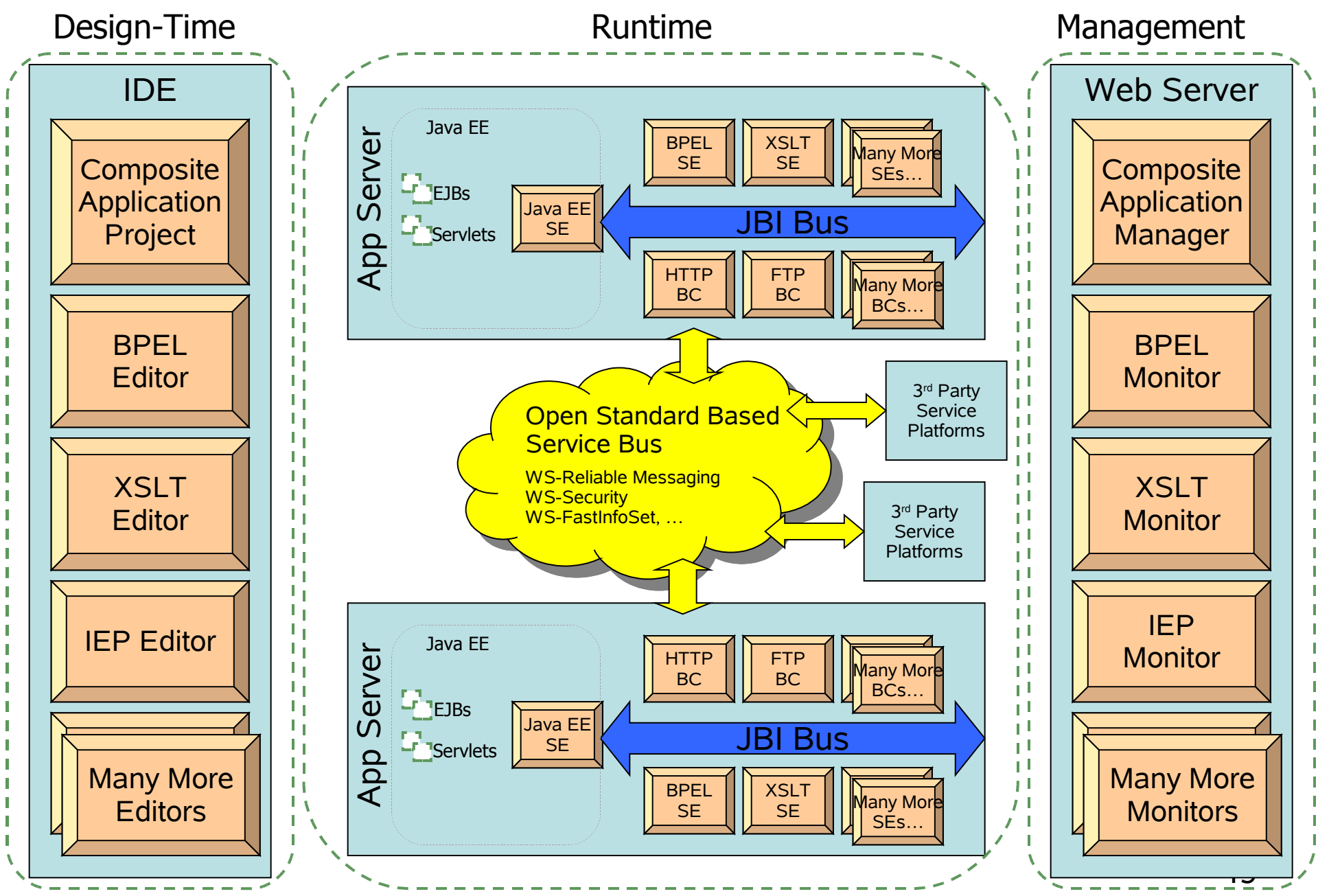
Open ESB

What is Project Open ESB?

- Project Open ESB 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

Open ESB Architecture



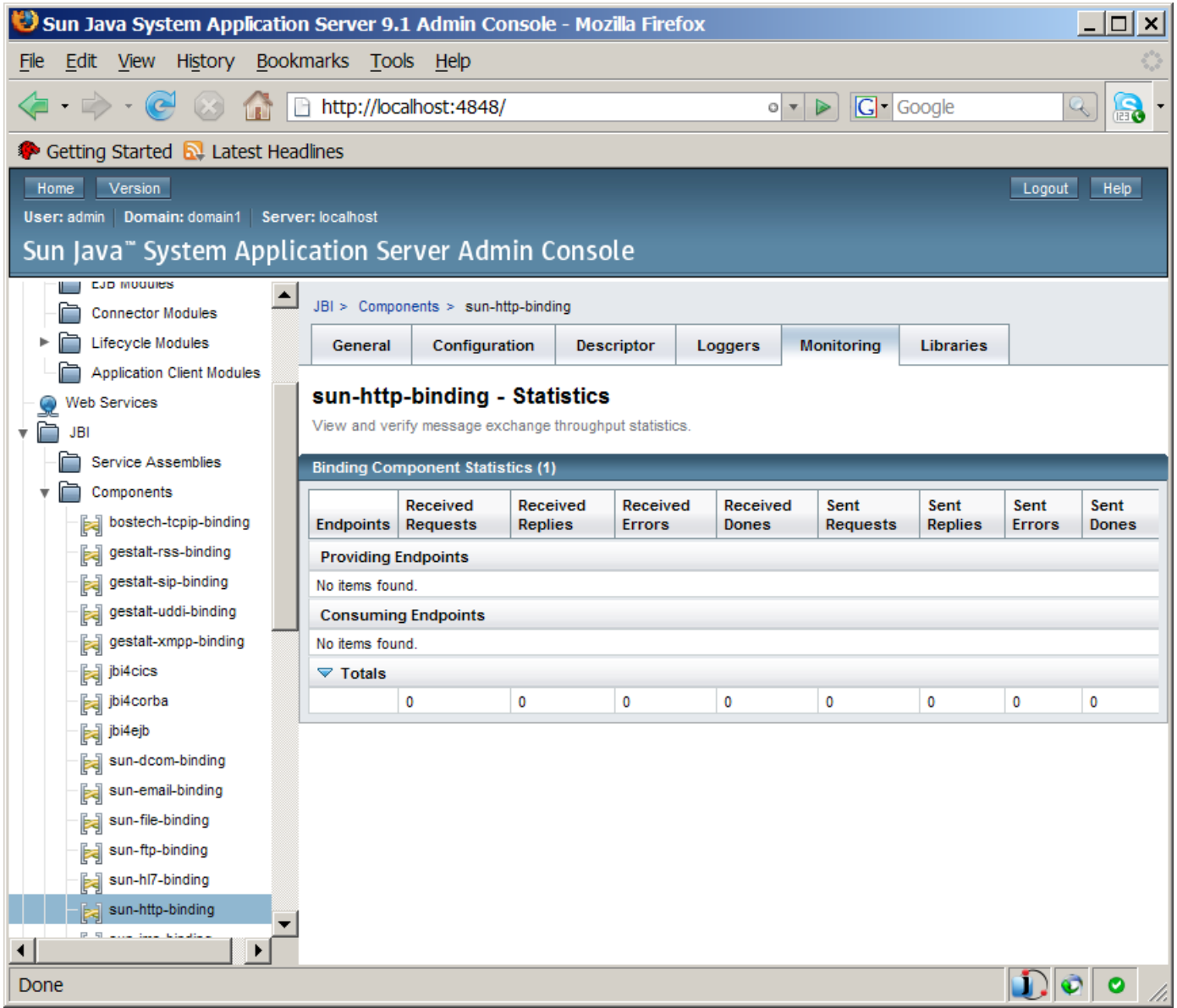

Design-Time
Runtime
Management
IDE
Composite Application Project
BPEL Editor
XSLT Editor
IEP Editor
Many More Editors
App Server
Java EE
EJBs
Servlets
Java EE SE
BPEL SE
XSLT SE
Many More SEs…
JBI Bus
HTTP BC
FTP BC
Many More BCs…
Open Standard Based Service Bus
WS-Reliable Messaging
WS-Security
WS-FastInfoSet, …
3rd Party Service Platforms
3rd Party Service Platforms
App Server
Java EE
EJBs
Servlets
Java EE SE
HTTP BC
FTP BC
Many More BCs…
JBI Bus
BPEL SE
XSLT SE
Many More SEs…
Web Server
Composite Application Manager
BPEL Monitor
XSLT Monitor
IEP Monitor
Many More Monitors

JBI and GlassFish

JBI Support in GlassFish

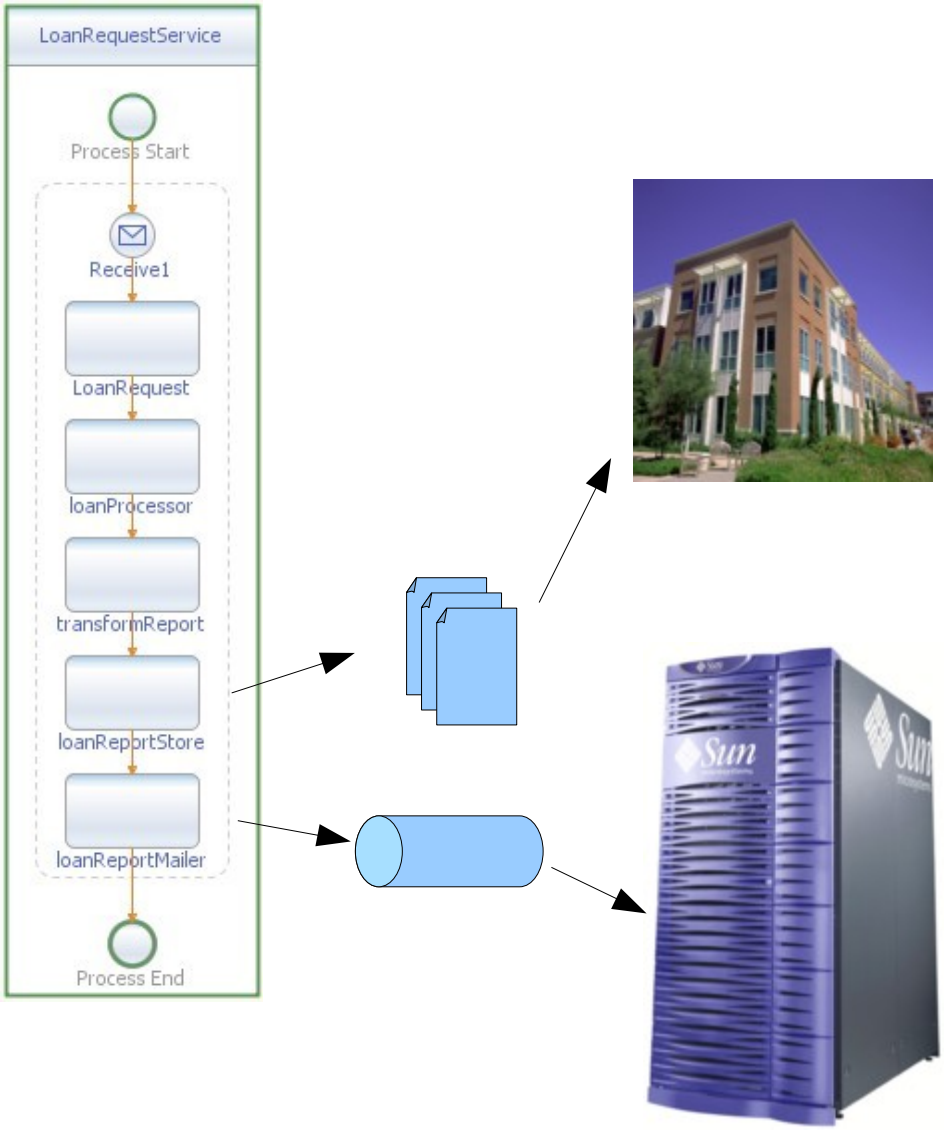
- A JBI runtime has been integrated with GlassFish V2
- GlassFish admin console now supports JBI
- Java EE Service Engine act as the bridge between Java EE applications and JBI
- A Java EE application archive (ear/war/jar) can be packaged in a JBI composite application
- JBI runtime has been enhanced 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



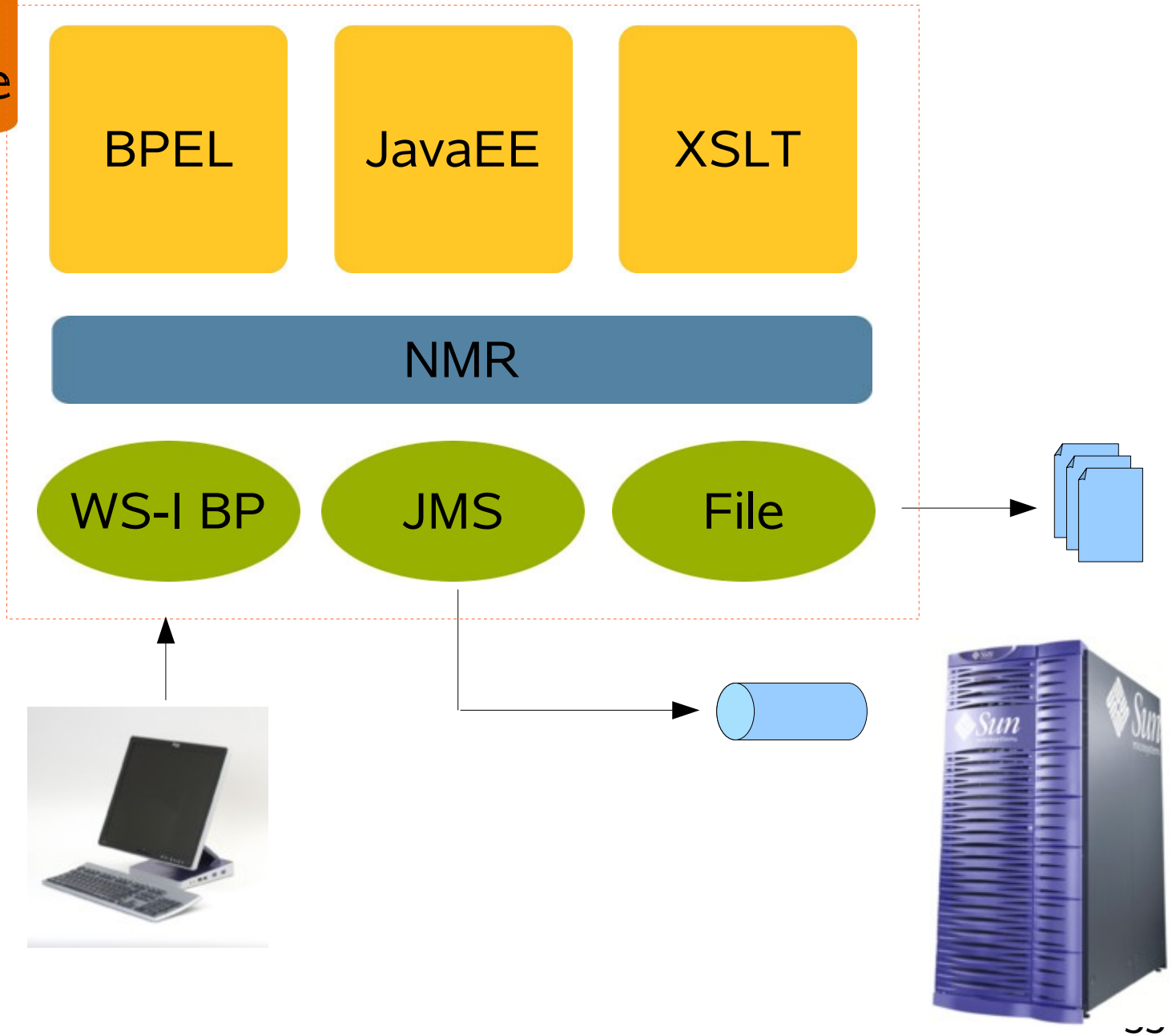
Usage Scenario

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



JBI-based Infrastructure

BPEL

Loan Request Process

JavaEE

Loan Processor EJB

XSLT

Transform Report

NMR

WS-I BP

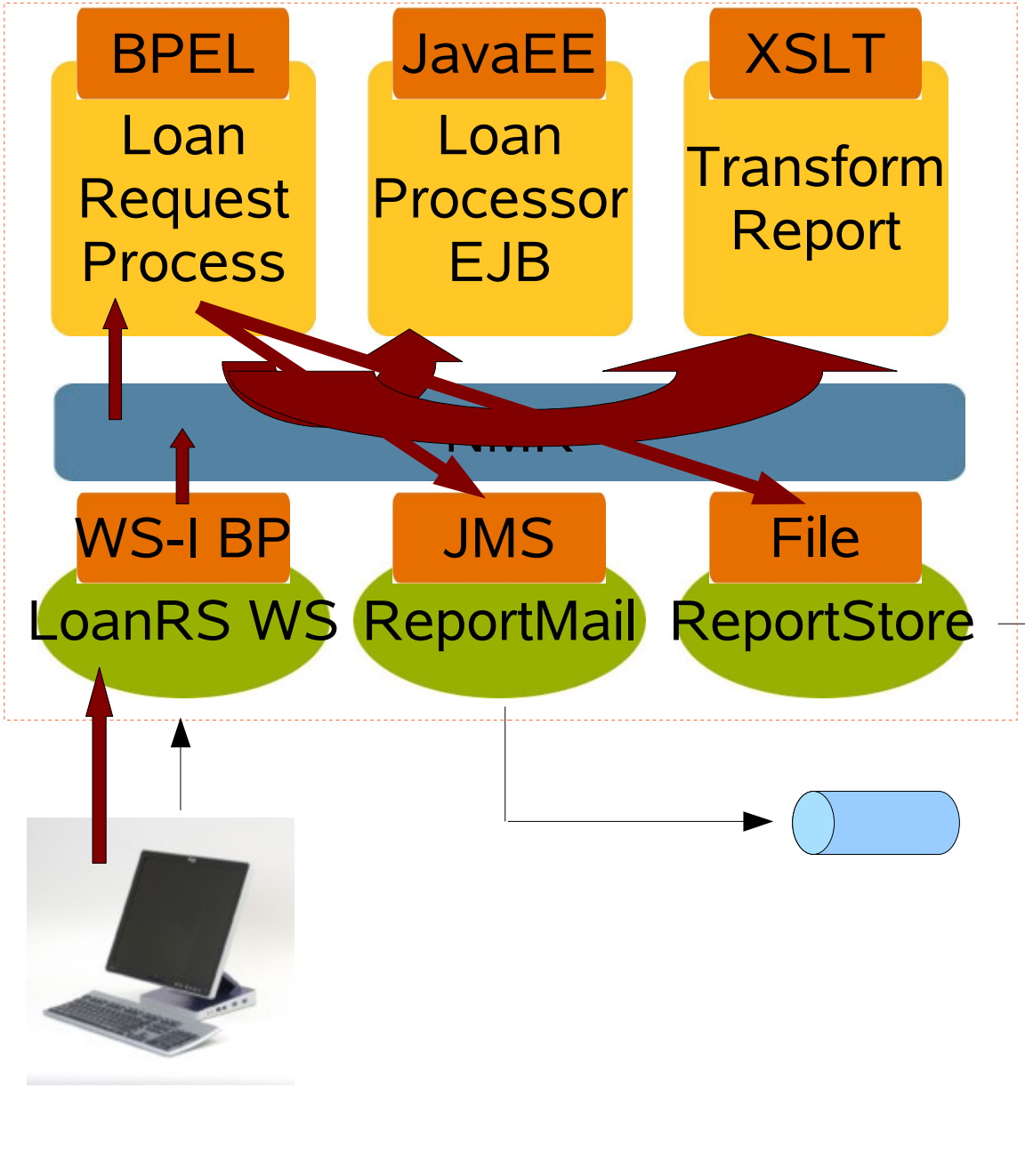
LoanRS WS

JMS

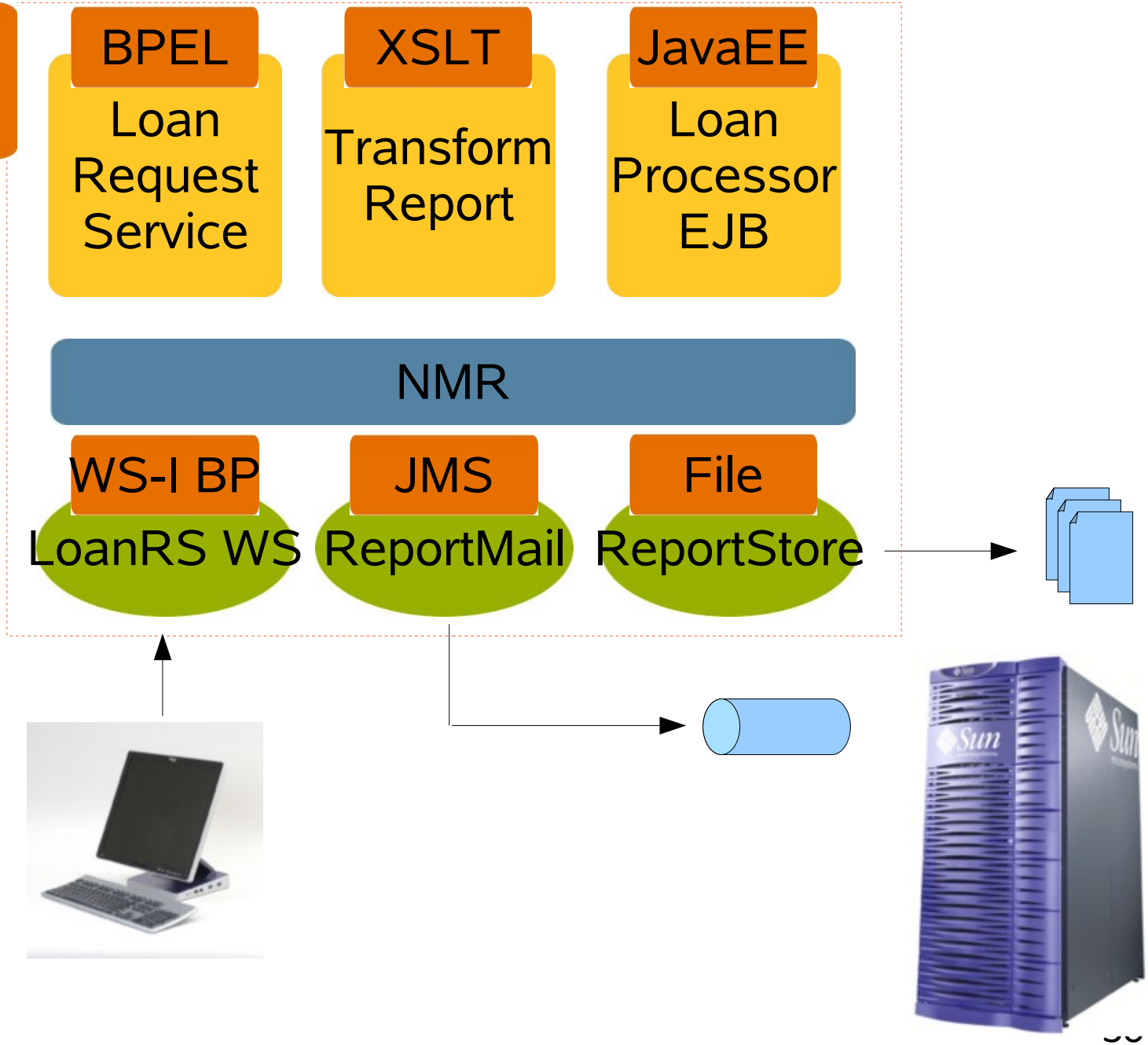
ReportMail

File

ReportStore


BPEL
Loan Request Process
JavaEE
Loan Processor EJB
XSLT
Transform Report
WS-I BP
LoanRS WS
JMS
ReportMail
File
ReportStore

Architecture Refactoring



BPEL

Loan Request Service

XSLT

Transform Report

RulesEngine

Loan Processor

JavaEE

ReportStore

NMR

JMS

LoanRS Q

WS-I BP

LoanRS WS

JMS

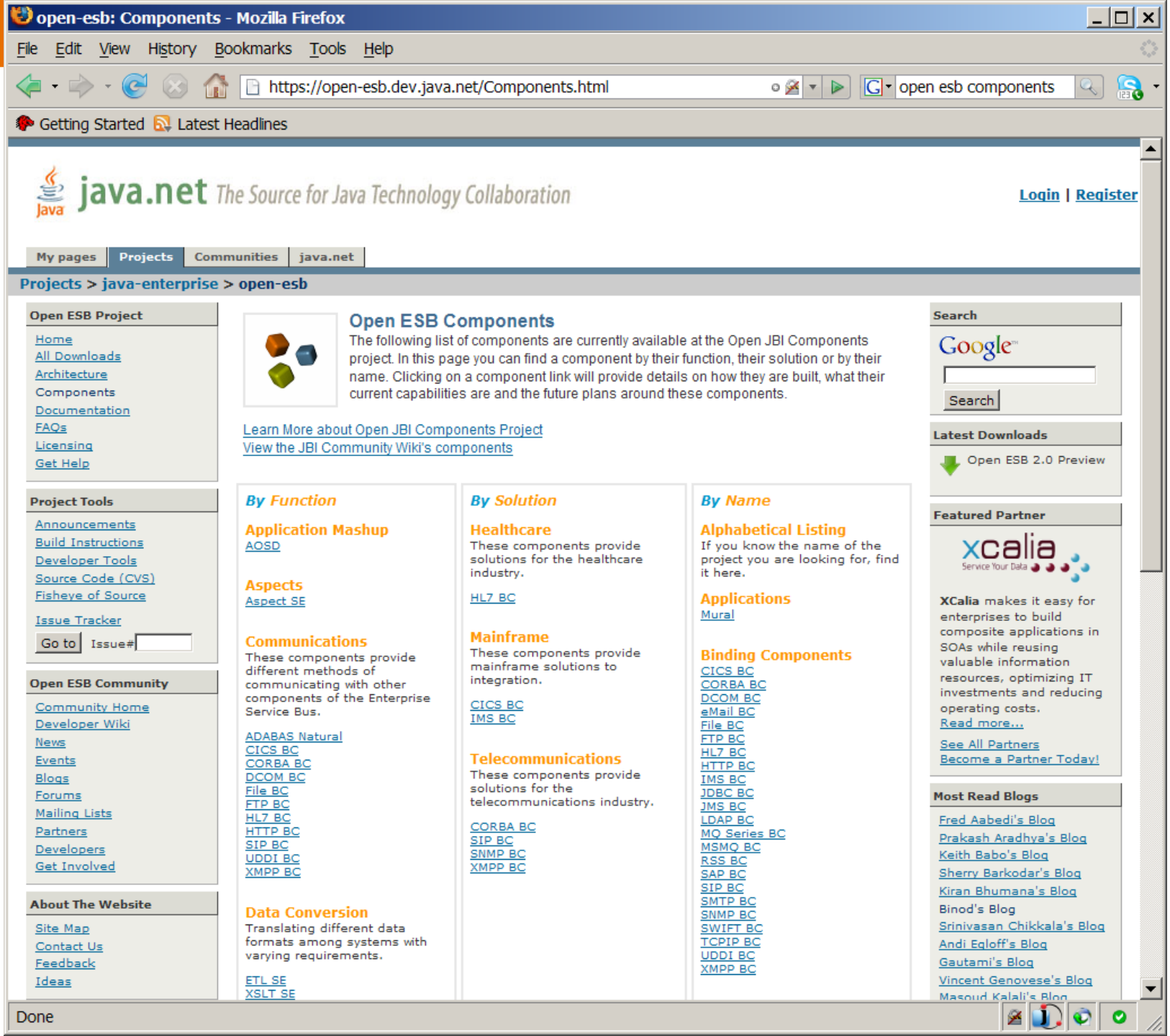
ReportMail

File

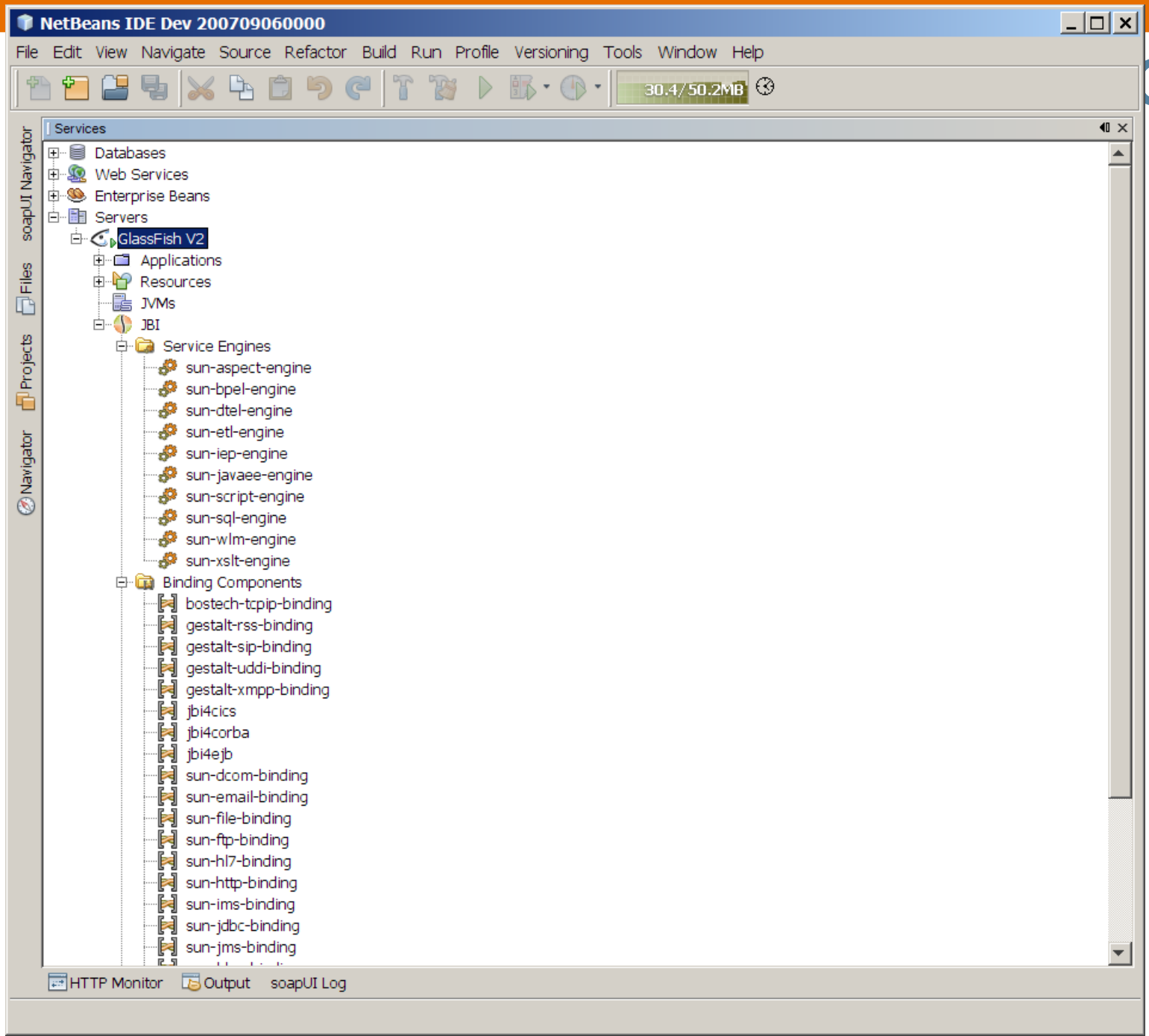
Service Engines (SE) & Binding Components (BC)

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 - Mozilla Firefox
File Edit View History Bookmarks Tools Help
https://open-esb.dev.java.net/Components.html
open esb components
Getting Started Latest Headlines
java.net The Source for Java Technology Collaboration
Login | Register
My pages Projects Communities java.net
Projects > java-enterprise > open-esb
Open ESB Project
Home
All Downloads
Architecture
Components
Documentation
FAQs
Licensing
Get Help
Project Tools
Announcements
Build Instructions
Developer Tools
Source Code (CVS)
Fisheye of Source
Issue Tracker
Go to Issue#
Open ESB Community
Community Home
Developer Wiki
News
Events
Blogs
Forums
Mailing Lists
Partners
Developers
Get Involved
About The Website
Site Map
Contact Us
Feedback
Ideas
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
AOSD
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
HL7 BC
HTTP BC
SIP BC
UDDI BC
XMPP BC
Data Conversion
Translating different data formats among systems with varying requirements.
ETL SE
XSLT SE
By Solution
Healthcare
These components provide solutions for the healthcare industry.
HL7 BC
Mainframe
These components provide mainframe solutions to integration.
CICS BC
IMS BC
Telecommunications
These components provide solutions for the telecommunications industry.
CORBA BC
SIP BC
SNMP 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
HL7 BC
HTTP BC
IMS BC
JDBC BC
JMS BC
LDAP BC
MQ Series BC
MSMQ BC
RSS BC
SAP BC
SIP BC
SMTP BC
SNMP BC
SWIFT BC
TCPIP BC
UDDI BC
XMPP BC
Search
Google
Search
Latest Downloads
Open ESB 2.0 Preview
Featured Partner
xcalia
Service Your Data
XCalia makes it easy for enterprises to build composite applications in SOAs while reusing valuable information resources, optimizing IT investments and reducing operating costs.
Read more...
See All Partners
Become a Partner Today!
Most Read Blogs
Fred Aabedi's Blog
Prakash Aradhya's Blog
Keith Babo's Blog
Sherry Barkodar's Blog
Kiran Bhumana's Blog
Binod's Blog
Srinivasan Chikkala's Blog
Andi Egloff's Blog
Gautami's Blog
Vincent Genovese's Blog
Done

C's

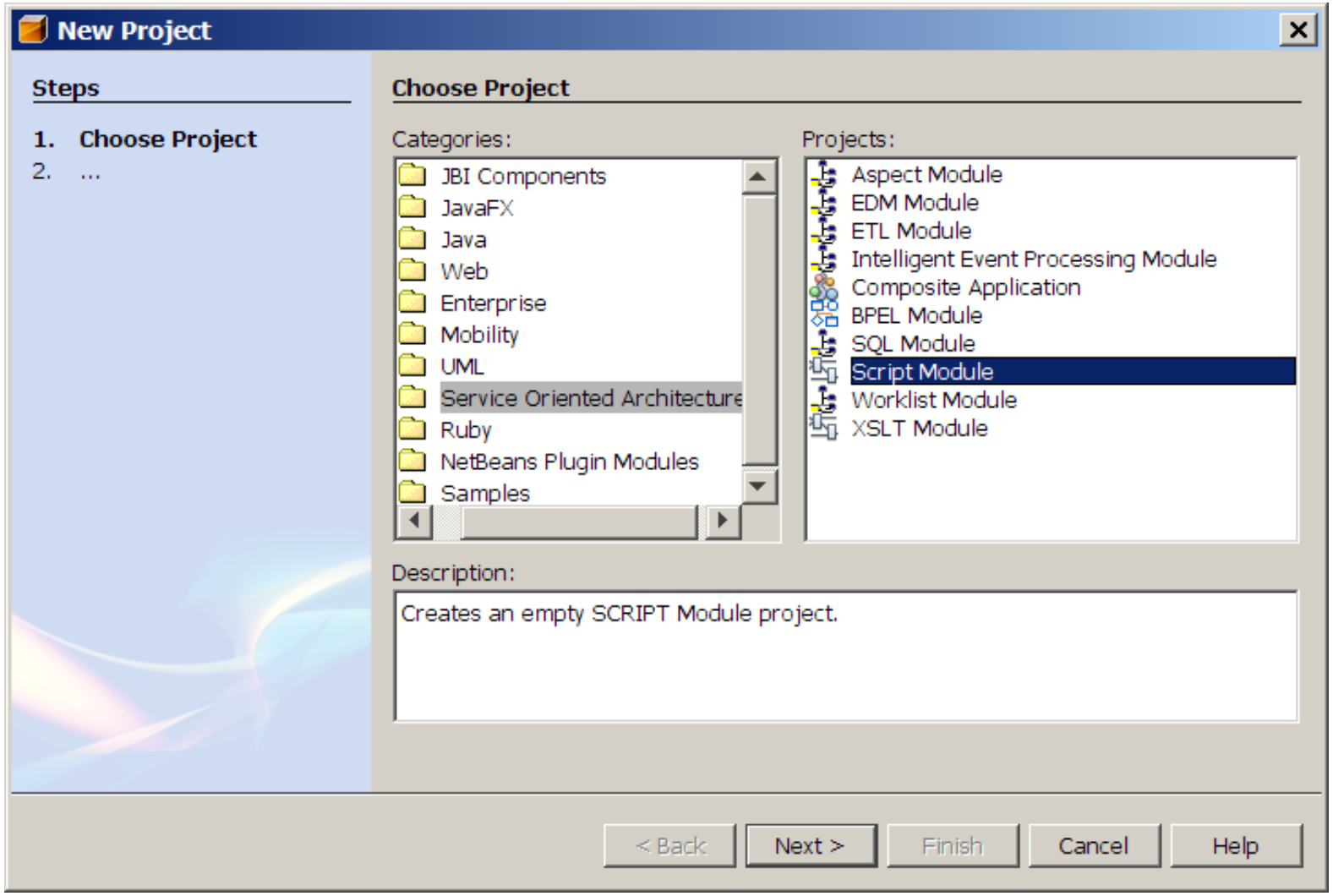

NetBeans IDE Dev 200709060000
File Edit View Navigate Source Refactor Build Run Profile Versioning Tools Window Help
30.4/50.2MB
Services
Databases
Web Services
Enterprise Beans
Servers
GlassFish V2
Applications
Resources
JVMs
JBI
Service Engines
sun-aspect-engine
sun-bpel-engine
sun-dtel-engine
sun-etl-engine
sun-iep-engine
sun-javaee-engine
sun-script-engine
sun-sql-engine
sun-wlm-engine
sun-xslt-engine
Binding Components
bostech-tcpip-binding
gestalt-rss-binding
gestalt-sip-binding
gestalt-uddi-binding
gestalt-xmpp-binding
jbi4cics
jbi4corba
jbi4ejb
sun-dcom-binding
sun-email-binding
sun-file-binding
sun-ftp-binding
sun-hl7-binding
sun-http-binding
sun-ims-binding
sun-jdbc-binding
sun-jms-binding
soapUI Navigator
Files
Projects
Navigator
HTTP Monitor
Output
soapUI Log

NetBeans Support of Open ESB

Types of SOA "NetBeans" Projects

- When creating a composite application, you typically use the following types of SOA "NetBeans" projects:
  - BPEL Module project (NetBeans 6.0)
  - XSLT Module project (NetBeans 6.0)
  - SQL Module project (NetBeans 6.0)
  - Composite Application project (NetBeans 6.0)
  - IEP Module project (OpenESB package)
  - Worklist Module project (OpenESB package)
  - ETL (Extract, Transform, and Load) (OpenESB package)
  - EDM (Enterprise Data Mashup) (OpenESB package)
  - And more

Types of SOA "NetBeans" Projects



BPEL Module Project

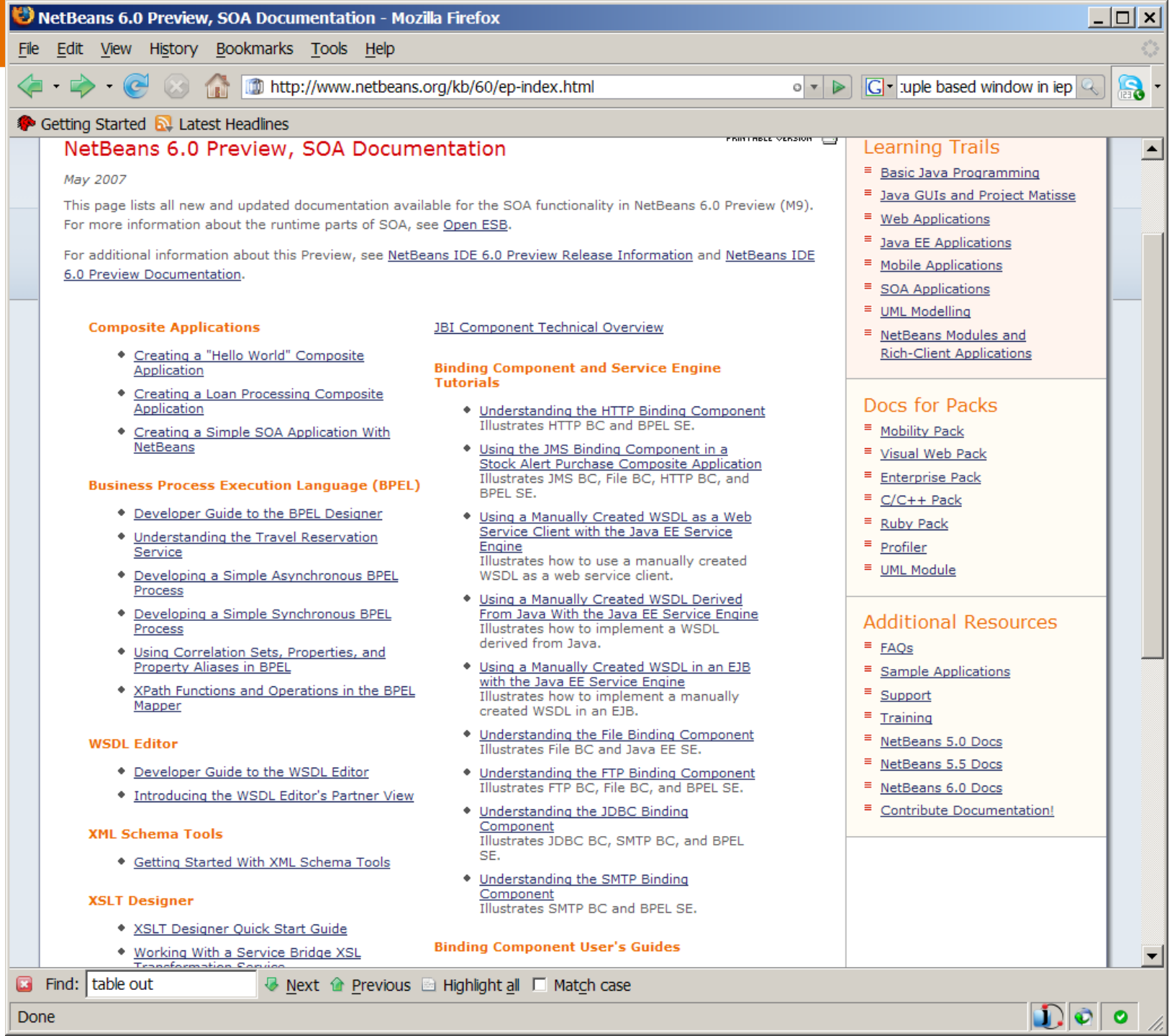
- BPEL Module project is a group of source files which includes
  - XML Schema (*.xsd) files
  - WSDL files
  - BPEL files
- Within a BPEL Module project, you can author a business process compliant with the WS-BPEL 2.0 language specification.
- Will be added to a Composite application as a JBI module

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.

Composite Application Project

- 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 - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://www.netbeans.org/kb/60/ep-index.html
Getting Started Latest Headlines
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.
Binding Component User's Guides
Learning Trails
Basic Java Programming
Java GUIs and Project Matisse
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/

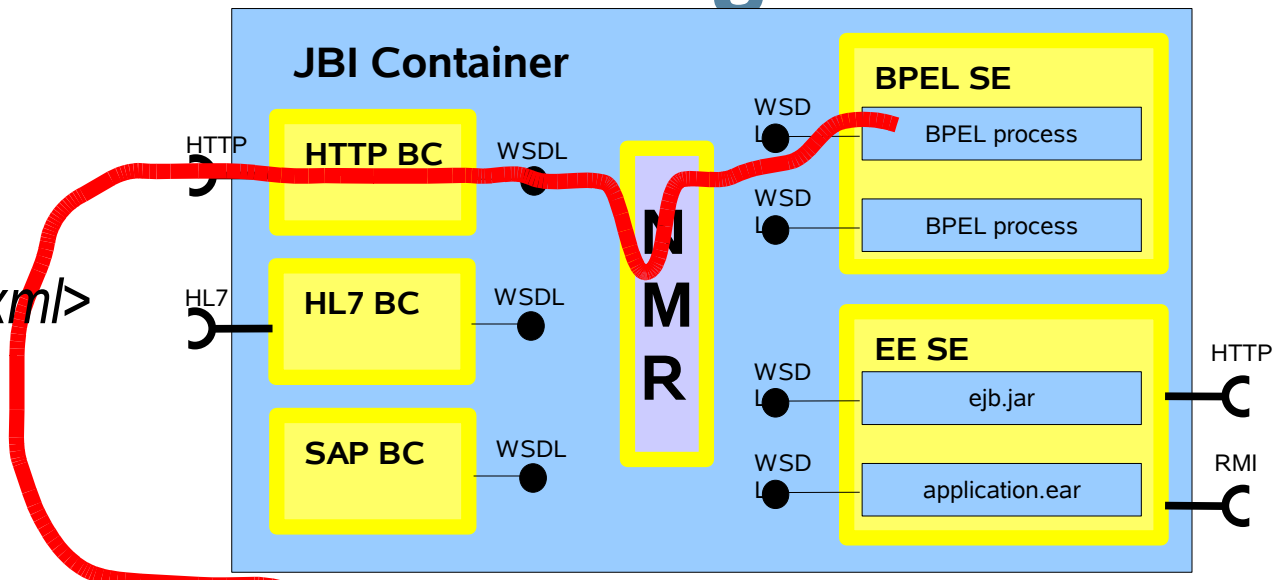
Java EE SE

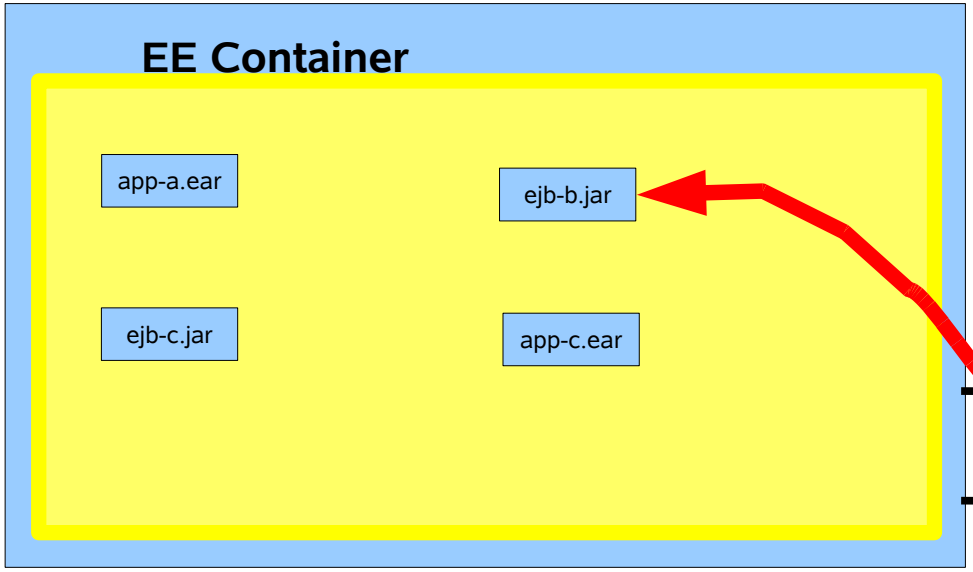
JavaEE SE

- 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 throughHTTP BC

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

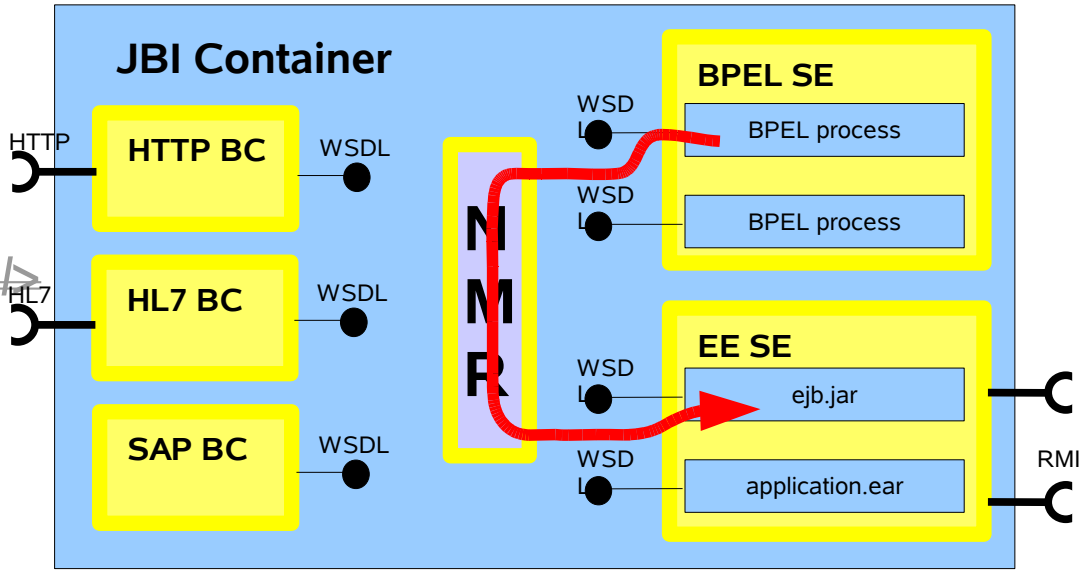




- Network layer
- SOAP/HTTP
- JAXWS
- *Unmarshal <xml>*
- *Create DOM*
- JAXB
- WS.helloWorld(name)

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)



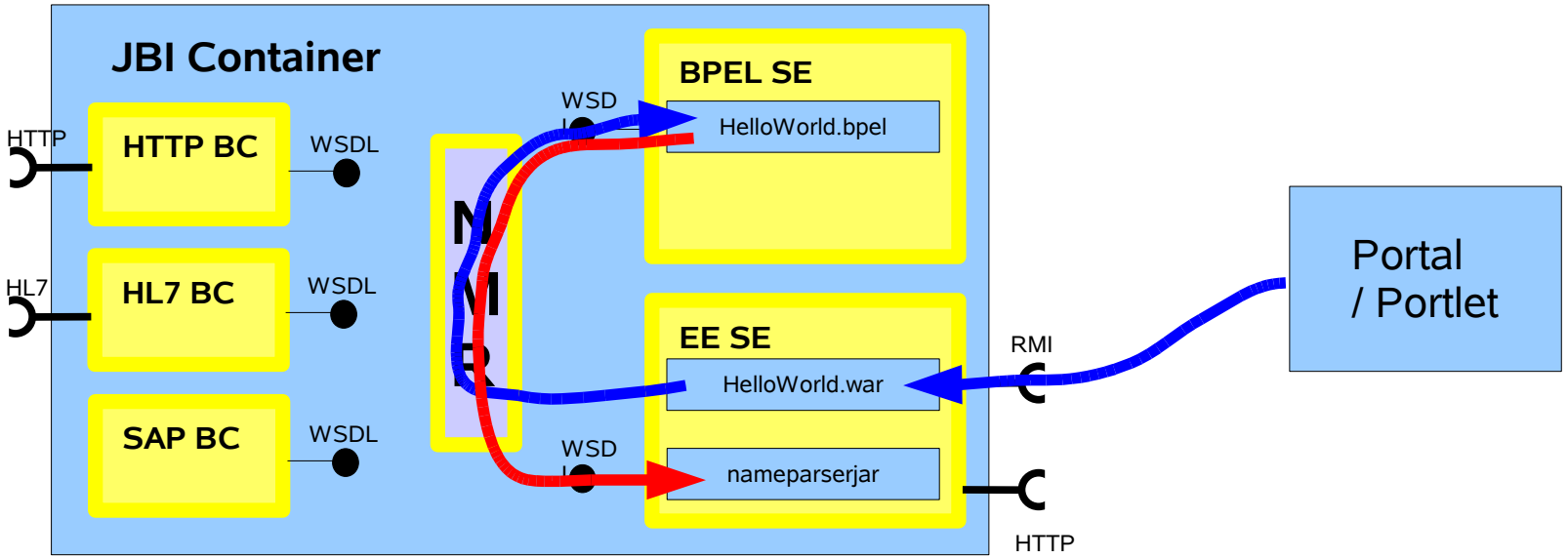
Advantages:

- Performance
- Transaction propagation
- Security context propagation

Likewise: EJB to BPEL

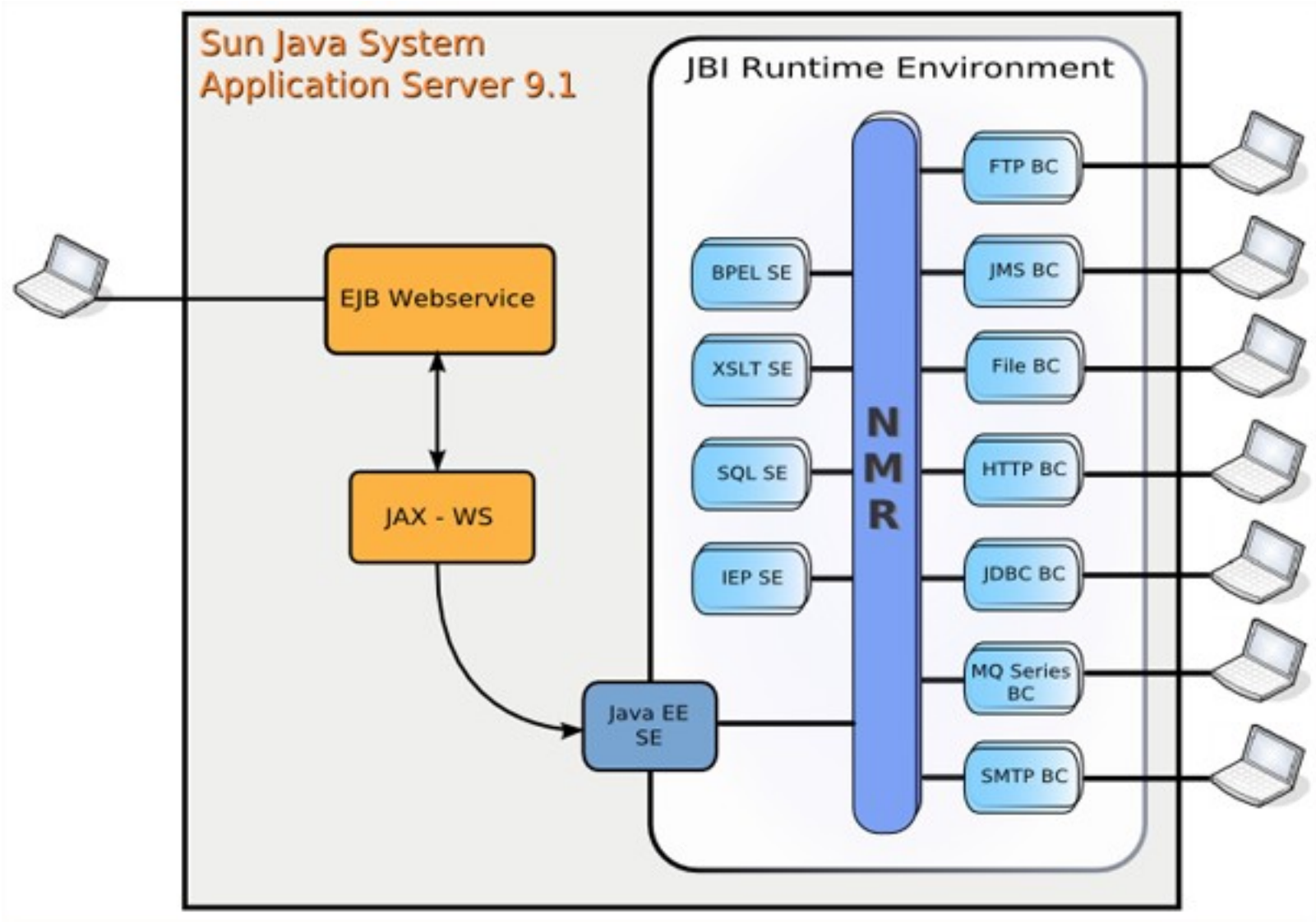
Scenario: Portal + EE + BPEL

- Portlet gets name, invokes WAR which calls BPEL to orchestrate process
- BPEL activity requires complex business logic
  - > executes faster in EJB right

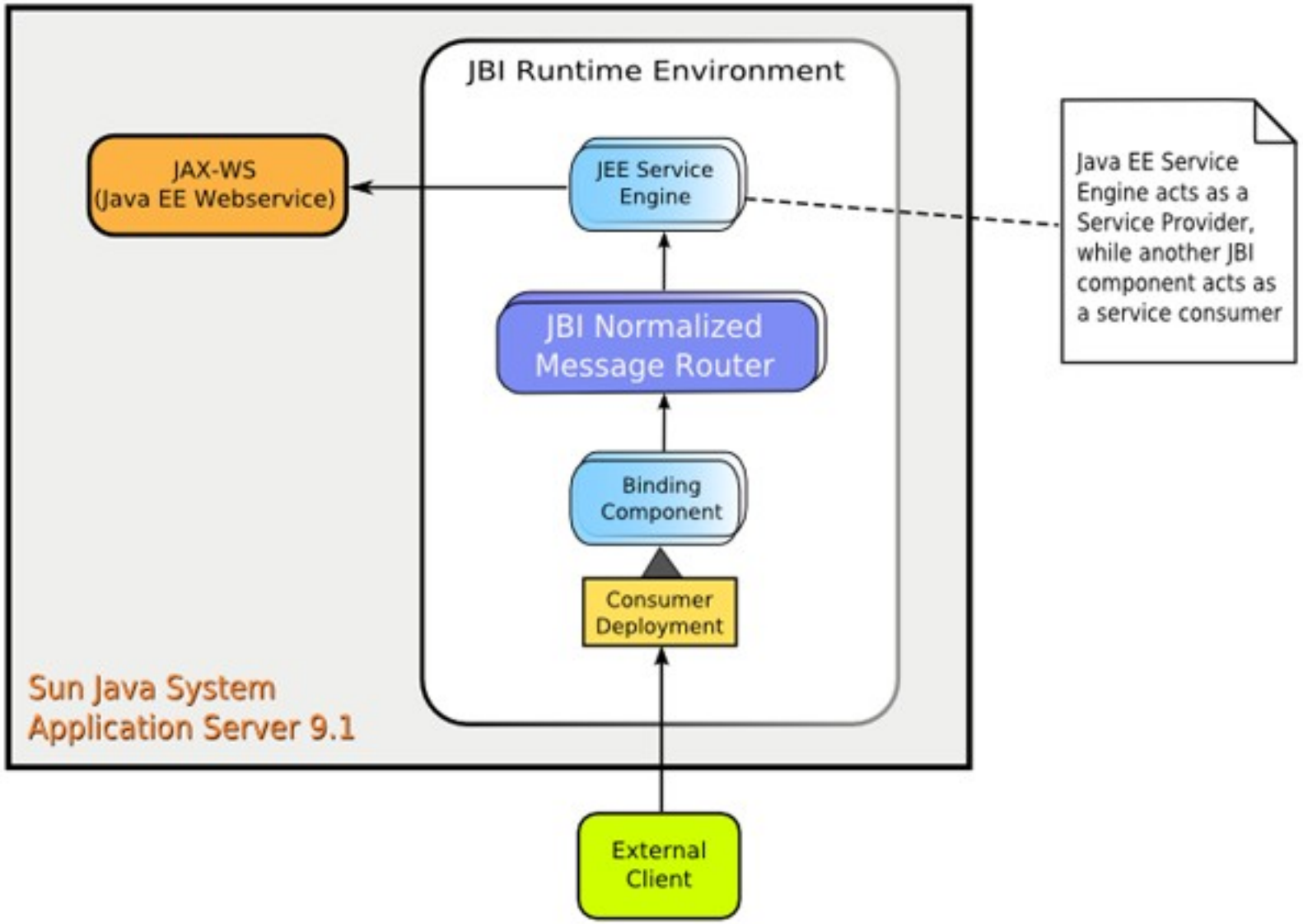


Java EE Service Engine: Functions as Bridge between App Server and JBI Runtime Env.

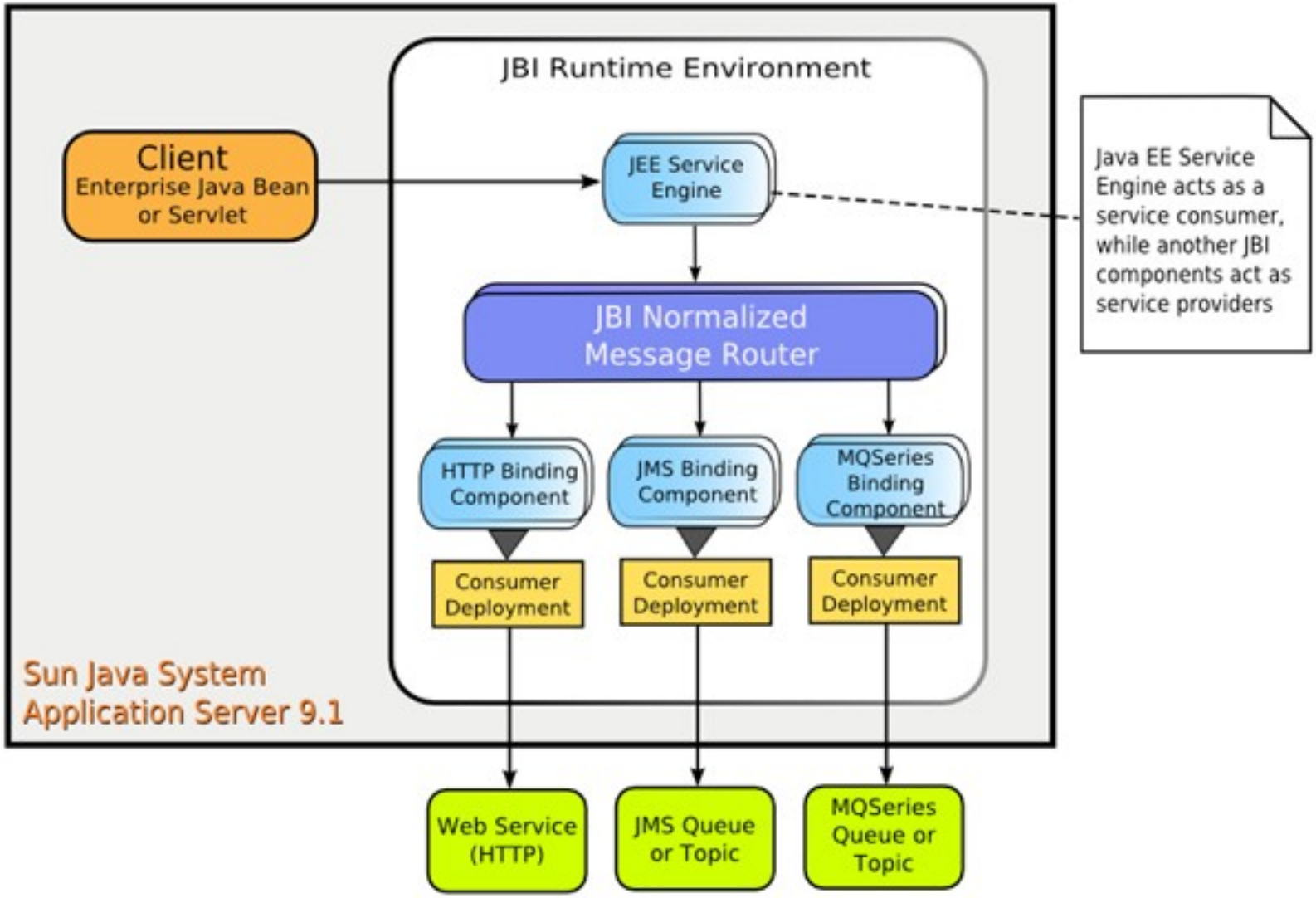
Java EE S.E. As a Bridge



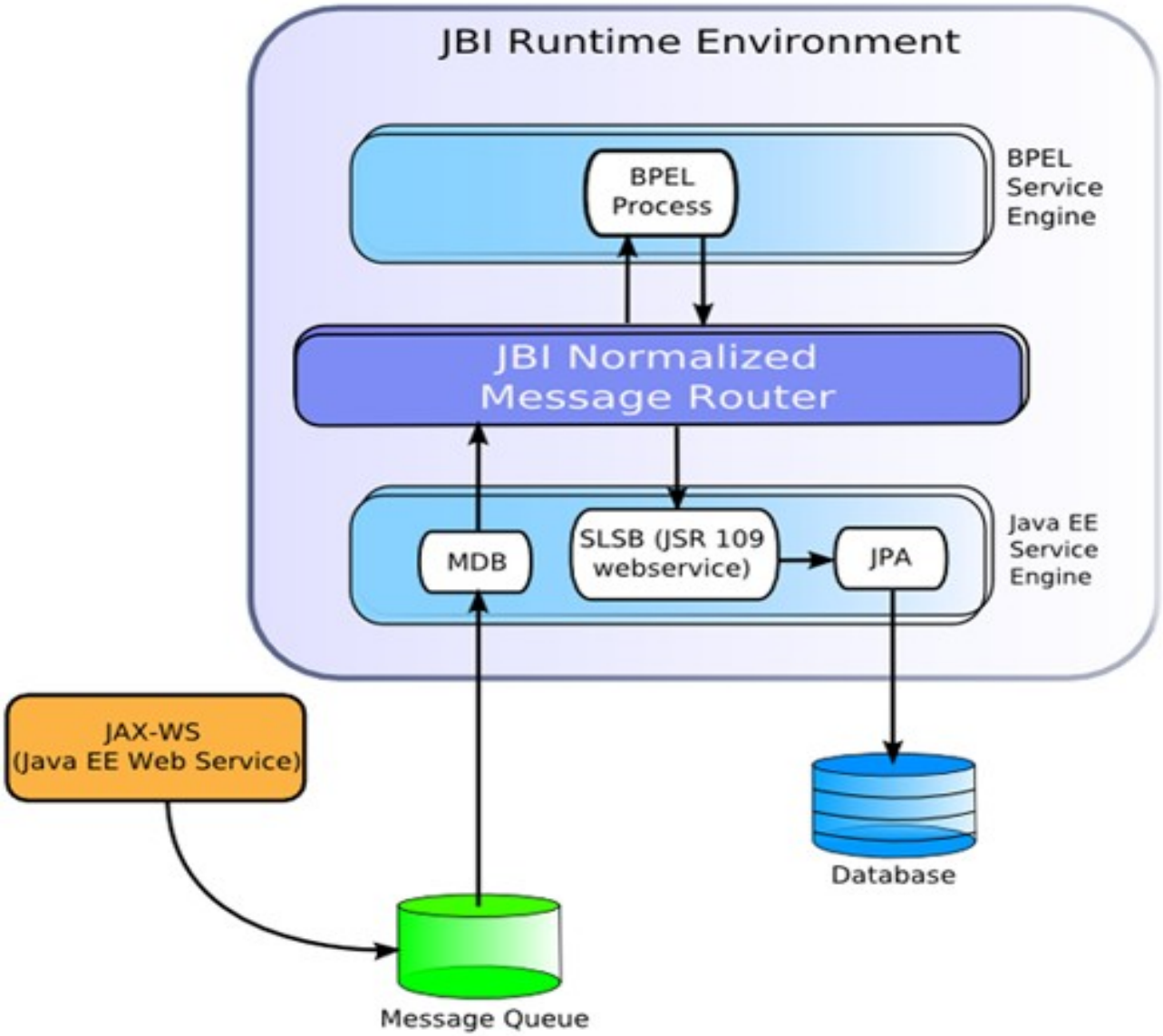
Java EE S.E. as a Service Provider



Java EE S.E. as a Service Consumer



Java EE S.E. Use Case



Java EE S.E. Use Cases

- BPEL Service Engine calling an Enterprise Java Bean web service
- Message Driven Bean or Servlet calling a BPEL Process
- Enterprise Java Bean web service called through a JMS transport using the JMS Binding Component.
- Java EE components calling web services using the FTP Binding Component
- Java EE components making web service calls through SMTP transport using the SMTP Binding Component

Demo:
Building "Loan Processing" Composite Application

www.javapassion.com/handsonlabs/wscompositeapps/#Exercise_2
www.javapassion.com/handsonlabs/wscompositeapps/#Exercise_3

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