Petals, JBI & JOnES

Gaël Blondelle
CTO and co-founder

Context

JBI: Java™ Business Integration
- The Java™ standard for integration solutions assembly

Petals
- ObjectWeb project to implement a highly distributable JBI compliant container
- Led by eBM WebSourcing and Fossil E-Commerce

JOnES
- French R&D funded project (RNTL)
- Supports the development of a JBI container in ObjectWeb code base
Agenda

Introduction to JBI

Petals

JOnES

The three JBI focus

An environment for the management of integration components
  ▪ To assemble JBI compliant components

Loosely coupled messaging
  ▪ Based on WSDL description and XML document payload

Web Service support and interaction
  ▪ SOAP support is mandatory
JBI is a container of containers (*)

JBI: a standard infrastructure to create service-oriented architectures
Enables the creation of custom integration solutions by just assembling JBI compliant components

JBI components are as diverse as
- An XSLT transformation engine
- A Web Service Binding Component with eventual support for WS Security or WS Reliable Messaging
- A J2EE application server like JOnAS, in order to implement common integration services as EJBs
- A BPEL orchestration engine

JBI container manages
- The installation of JBI components
- The deployment of artifacts to the JBI component (XSLT stylesheet, BPEL process definition, …)

(*)See Ron Ten-Hove’s blog. Ron is JSR 208 specification lead

JBI architecture

Two types of JBI components:
- **Service Engines**: provide and consume business logic and transformation services
- **Binding Components**: provide connectivity to services external to a JBI installation
JBI Messaging

JBI defines a model and the APIs for messaging between JBI components.

Loosely coupled
- Several types of component address

All types of interaction
- One-Way
- Reliable One-Way
- Request Response
- Request Optional-Response

Messaging API
- Components send messages to the JBI container
- Components read messages from the JBI container

JBI and JMS

<table>
<thead>
<tr>
<th>JMS</th>
<th>JBI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Both provide a messaging API</td>
<td></td>
</tr>
<tr>
<td>Message exchange paradigm</td>
<td></td>
</tr>
<tr>
<td>Implementations can provide reliable communication and persistent store for both</td>
<td></td>
</tr>
<tr>
<td>Pulling and Listener</td>
<td>Only pulling in JBI 1.0</td>
</tr>
<tr>
<td>Queues and Topics can be part of a transaction</td>
<td>Transaction support is optional</td>
</tr>
<tr>
<td>Explicit addressing of queues and topics</td>
<td>Implicit addressing</td>
</tr>
</tbody>
</table>
Explaining JBI Addressing

An interface is a group of operations.

A service implements an interface represented by qualified name and its WSDL definition.

A service has one or more endpoints accessible by different bindings.

Endpoints can be:
- external for services accessible via a binding component
- internal for services provided by a service engine

JBI and Web Services

The JBI specification relies on WSDL definitions:
- Message Exchange patterns are those of WSDL 2.0 spec.
- WSDL 2.0 or WSDL 1.0 must be supported for service interface descriptions.
- JBI normalized messages follows Web Services document oriented exchange paradigm.

JBI implementations must support a SOAP 1.1 Binding Component:
- Conforming to the WS-I Basic Profile.
Development
- Develop JBI specific classes and message processors \(\Rightarrow\) the core of the component

Packaging
- Package component classes and component description (an XML document known as the installation descriptor) into a JAR archive

Installation
- Installing it into the JBI environment thanks to JMX based administrative tools

Deployment
- JBI component can be seen as a container (ex: BPEL engine).
- Services (ex: BPEL) must be deployed into the component thanks to JBI administrative tools

Activation
- After deployment, services are accessible through the JBI environment as Endpoints
JBI component packaging

Write an XML descriptor for the component: the deployment descriptor

Package the descriptor with component logic in a JAR archive

Install the component in JBI environment

Ajouter un exemple jbi.xml

Component Installation Package (JAR Archive)

Component Descriptor (jbi.xml)

Component Classes (JAR archive)

<?xml version="1.0" encoding="utf-8"?>
<jbi version="1.0" xmlns="http://java.sun.com/xml/ns/jbi">
  <identification>
    <name>xslt-engine-1</name>
    <description>An example service engine</description>
  </identification>
  <component-class-name description="foo">
    org.objectweb.engine.XsltEngine
  </component-class-name>
  <component-class-path>
    <path-element>XsltEngine.jar</path-element>
  </component-class-path>
  <bootstrap-class-name>
    org.objectweb.engine.XsltEngineBootstrap
  </bootstrap-class-name>
  <bootstrap-class-path>
    <path-element>XsltEngine.jar</path-element>
  </bootstrap-class-path>
  <shared-library>util</shared-library>
</component>
</jbi>
Deployment

Provide services to JBI environment through the JBI component

Service (component artifact) is described by XML descriptor and packaged as “Service Unit” (SU)

Services that have to be deployed into different components to produce a new application can be grouped into a “Service Assembly” (SA).

The SA contains an XML descriptor to describe how each SU is deployed on its target component.

```
<?xml version="1.0" encoding="UTF-8"?>
<jbi version="1.0" xmlns="http://java.sun.com/xml/ns/jbi">
  <service-assembly>
    <identification>
      <name>ordering-deployment-1</name>
      <description>An example deployment of two service units</description>
    </identification>
    <service-unit>
      <identification>
        <name>xslt-artifacts</name>
        <description>Ordering xslt artifacts</description>
      </identification>
      <target>
        <artifacts-zip>xslt-artifacts.zip</artifacts-zip>
        <component-name>xslt-engine-1</component-name>
      </target>
    </service-unit>
    <service-unit>
      <identification>
        <name>bpel-artifacts</name>
        <description>Ordering bpel artifacts</description>
      </identification>
      <target>
        <artifacts-zip>bpel-artifacts.zip</artifacts-zip>
        <component-name>bpel-engine-1</component-name>
      </target>
    </service-unit>
  </service-assembly>
</jbi>
```
Components and artifacts are managed with the same unified lifecycle

Lifecycle management is accessible by admin tools via standard JMX MBean defined by the specification

**Bootstrap**
- Used by the JBI engine during component installation
- Any component specific behavior that is needed at (un)install time (ex: creation of database) can be added to this class

**ServiceUnitManager**
- Called by the JBI engine during Service Unit (component artifact) deployment
- Manages the life cycle (init, start, stop or shutdown) of component artifacts
- Optional, required for artifacts deployment
JBI Component main classes

Component
- The base class for a JBI component.

ComponentLifeCycle
- Manages the component lifecycle (init, start, stop or shutdown)

MessageProcessor
- Not required by JBI, but good design pattern
- Used to process Normalized Messages sent to the JBI Component
- Used to link the component main classes and business (or technical) logic associated with the component
Message Processing

Sequence

Message Processing

Message Processing

Message Processing

Message Processing

JBI classes

Agenda

Introducing JBI

Petals

JOnES
Petals

Project created in June 2005 by two ObjectWeb members
- EBM WebSourcing (France)
- Fossil E-Commerce (Brazil)

Vision
- Implement a JBI container as a basis for extensible integration solutions
- Focus on distribution to enable multiple JBI containers to be managed and operated as one unique platform
- Provides basic pre-packaged solutions for common integration needs

A distributed container

JSR 208 promotes a vision of a centralized JBI container
- Running in one JVM
- In this vision, two JBI containers can be connected via specialized Binding Component
- Distribution is considered as future work

Petals promotes a vision of a distributed JBI environment
- Several JBI containers are considered as one unified container
- Central administration and replication of administrative data
- Inter-container communications based on robust message based middleware from ObjectWeb: JORAM
Distributed JBI containers form a unified integration solution

---

**Tooling**

Development and deployment tools
- Will be based on the SOA Tools Project
- eBM WebSourcing is committer of the STP project to
  - Work on the link between SCA and JBI
  - Write deployment plugins for Petals

Monitoring and Operation tools
- Mandatory for industry adoption (ability to run)
- Monitoring will be built on top of Fractal mixing capabilities for flow interception
- Objective: Be able to manage more than a hundred of nodes
Petals Architecture

**Provide preconfigured packages**

Process orchestration solution
- BPEL engine
- XSLT and Xpath transformation engine
- Content based routing
- Interfaces with J2EE container (JOnAS)
- Interfaces with Workflows Engines for user interaction (Bonita and Shark)

B2B integration solution
- Specific Binding Components
  - Odette FTP via integration of ObjectWeb odettej
  - Integrate ebXML binding
  - Integrate AS1, AS2 binding
Status

Project created during Summer 2005
First code contribution from eBM WebSourcing in 08/05
Rebuild of the architecture to use the Fractal component model and migration to Maven build in 12/05
Focus on JBI standard compliance
  - Delivery of a JBI compliant container expected in March 2006
Next step is to focus on distributed JBI in the scope of the JOnES project

Agenda

Introducing JBI

Petals

JOnES
Project objectives

Deliver two full releases of the project
- October 2006
- October 2007

Integrate key ObjectWeb technologies in the core of the container
- Fractal as the core component architecture framework
- Dream and JORAM for robust asynchronous messaging support
- GOTM for transactional support

Promote “JBization” of ObjectWeb components
- XQuare for transformation
- A BPEL engine to be defined

Demonstrate usability of the platform and its efficiency according to real use cases
- Information System backbone
- Collaborative environment

---

Project objectives

- Two years duration, for a total workload of 20 men year
- Partners (by alphabetical order)
  - eBM WebSourcing
  - ENSTIMAC
  - France Télécom, Division R&D
  - INRIA ObjectWeb
  - INRIA Futurs
  - INRIA Rhône-Alpes
  - OpenWide
  - Scalagent Distributed Technologies
Jones contributions in Petals Architecture

Requirements and architecture to be defined by H12006
First prototype and demos: end 2006
Second iteration in 2007 based on the new requirements learned from the demos
**JBI added value**

JBI value increases with the number of JBI compliant components available

- Some open source project have already been “JBIzed” by ServiceMix team

The partners of JOnES project are committed to

- “JBIze” relevant ObjectWeb projects
- Contribute to other open source projects “JBIzation” when relevant

A catalog of ObjectWeb JBI compliant components will be created on the Petals Web site

---

**Petals added value**

Implements JBI, the Java integration standard

Implementation based on proven ObjectWeb technologies

Addresses some industry concerns like clustering and monitoring

Provides packaged integration solutions for the most widely deployed use cases

- Process orchestration
- B2B exchanges

Extensible by nature to create customized integration solutions
Thank you for your attention

EBM WebSourcing

Open Service Oriented Factory

EBM WebSourcing is a start-up dedicated to provide e-Business advantage to his client thanks to a complete Process Driven Service Oriented approach

Funded in 2004 in Toulouse

An Integration Service Provider whose collaborative tools are natively Internet enabled and provided in an Application Service Provider (ASP) mode.

A corporate member of the ObjectWeb consortium, and a leader of the PETALS Service Platform a highly distributed open source JBI container hosted by ObjectWeb.

Provide consulting and training on Service Oriented Architecture (SOA) and open source strategy
EBM WebSourcing

Provides companies with a Service Oriented framework enabling secure collaborative business in ASP mode in order to create dynamic virtual organizations that optimize B2B collaborative partnerships.

Enable SMEs to group themselves in virtual organizations (clusters) and get a common, productive and affordable collaboration system.

EBM WebSourcing team is specialized in EIS (Enterprise Information Systems) integration and business processes management.