Service Oriented Integration With Apache ServiceMix

Bruce Snyder
bsnyder@apache.org
October 2008
Keystone, Colorado
Agenda

- Enterprise Service Bus
- Java Business Integration
- Apache ServiceMix ESB
- ServiceMix 4
Why it's bad when home owners change their minds about the bathroom’s location late in a building project.
Integration is Messy!
INTEGRATION

Just because you can, doesn’t mean you should.
What is an ESB?
"An Enterprise Service Bus (ESB) is a new architecture that exploits Web services, messaging middleware, intelligent routing, and transformation. ESBs act as a lightweight, ubiquitous integration backbone through which software services and application components flow."

(Gartner)
What is an ESB?

An ESB acts as a shared messaging layer for connecting applications and other services throughout an enterprise computing infrastructure. It supplements its core asynchronous messaging backbone with intelligent transformation and routing to ensure messages are passed reliably. Services participate in the ESB using either web services messaging standards or JMS (LooselyCoupled.com)
What is an ESB?

An ESB is an open standards, message-based, distributed, integration solution that provides routing, invocation, and mediation services to facilitate the interactions of disparate distributed information technology resources (applications, services, information, platforms) in a reliable manner.

(Brenda Michelson, Elemental Links)
Do I need an ESB?

ESB-oriented architecture: The wrong approach to adopting SOA

Use a Planning Process

ESB Planning Process

1. Your business purpose
2. Your functional requirements
3. Your architectural decisions
4. Your criteria for evaluating ESBs
What is JBI?
What is JBI?

*JBI defines an architecture that allows the construction of integration systems from plug-in components, that interoperate through the method of mediated message exchange.*

(JBI 1.0 Spec)
Java Business Integration

Normalized Message Router

JBI Environment
JBI Normalized Message

- **Normalized Message**
  - `key1=value1`
  - `key2=value2`
  - `xml v
    new
  cdx`
  - **Message Properties**
  - **Message Payload**
  - **Attachments**
  - `010101 0001101 0111000 1010101 0101010 0101010`
JBI Packaging

Service Assembly (.jar/.zip)

- META-INF/jbi.xml

Service Unit (.jar/.zip)

- META-INF/jbi.xml
- dependencies

<dependencies>
Apache ServiceMix

http://servicemix.apache.org/
Apache ServiceMix Architecture

Apache ServiceMix

BPEL XSLT Rules Scripting

Normalized Message Router

Standardized interfaces for Service Engines

SOAP File JCA Resources Legacy Adapters

Apache ActiveMQ

Installation Deployment JMX-based mgmt app
Control Monitoring Mgmt Console
Service Deploy & Debug Dependency Management Tooling
ServiceMix Features

- Supports many protocols
  - File, FTP, HTTP/S, JMS, SMTP, SOAP, TCP, XMPP

- Supports many engines
  - Apache Camel, Apache CXF, Apache ODE, Drools, OS Workflow, POJOs, Quartz, Scripting, Saxon XQuery and XSLT, WS-Notification

- Supports Security
  - JAAS, WS-Security

- Web Container/App Server Integration
  - Geronimo, JBoss, Jetty, Tomcat, Weblogic, Websphere
Apache Software Foundation
Message Routing

![Diagram of Message Routing](image)
Message Routing
Example
Configuration

Spring

<xml />

Colorado Software Summit: October 19 – 24, 2008
Bruce Snyder — Service Oriented Integration With Apache ServiceMix
The File Poller

```xml
<beans xmlns:file='http://servicemix.apache.org/file/1.0'
      xmlns:test='urn:test'>
  <file:poller service='test:file'
               endpoint='endpoint'
               targetService='test:wiretapIn'
               file='~/Users/bsnyder/smxdropbox'>
  </file:poller>
</beans>
```
<beans xmlns:eip="http://servicemix.apache.org/eip/1.0"
      xmlns:test="urn:test">
  <eip:wire-tap service="test:wiretapIn" endpoint="endpoint">
    <eip:target>
      <eip:exchange-target service="test:cbr" />
    </eip:target>
    <eip:inListener>
      <eip:exchange-target service="test:logger" />
    </eip:inListener>
  </eip:wire-tap>
</beans>
The Logger

```java
public class MyLogger extends RouteBuilder {
    public void configure() {
        from("jbi:service:urn:test:logger").
        process(new Processor() {
            public void process(Exchange exchange) {
                Message in = exchange.getIn();
                in.setBody(in.getBody(String.class) +
                            "<foo>xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx</foo>" );
            }
        })
        to("log:demo");
    }
}
```
The Logger

```xml
<beans xmlns="http://www.springframework.org/schema/beans"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.springframework.org/schema/beans
                          http://www.springframework.org/schema/beans/spring-beans-2.0.xsd
                          http://activemq.apache.org/camel/schema/spring
                          http://activemq.apache.org/camel/schema/spring/camel-spring.xsd">

  <camelContext id="camel"
                xmlns="http://activemq.apache.org/camel/schema/spring">
    <package>com.mycompany</package>
  </camelContext>

</beans>
```
The Content-Based Router

```xml
<beans xmlns:eip="http://servicemix.apache.org/eip/1.0"
       xmlns:test="urn:test">
  <eip:content-based-router service="test:cbr"
               endpoint="endpoint">
    <eip:rules>
      <eip:routing-rule>
        <eip:predicate>
          <eip:xpath-predicate
              xpath="/message/cheese/text() = 'gouda'" />
        </eip:predicate>
        <eip:target>
          <eip:exchange-target service="test:queue1" />
        </eip:target>
      </eip:routing-rule>
    </eip:rules>
  </eip:content-based-router>
</beans>
```
The Content-Based Router

...<eip:routing-rule>
  <eip:_predicate>
    <eip:xpath-predicate
      xpath="/message/cheese/text() = 'swiss'" />
  </eip:_predicate>
  <eip:target>
    <eip:exchange-target service="test:queue2" />
  </eip:target>
</eip:routing-rule>
</eip:rules>
</eip:content-based-router>
The JMS Sender

```xml
<beans xmlns:jms="http://servicemix.apache.org/jms/1.0"
       xmlns:test="urn:test"
       xmlns:amq="http://activemq.org/config/1.0">

    <jms:endpoint service="test:queue1"
                  endpoint="myProvider"
                  role="provider"
                  destinationStyle="queue"
                  jmsProviderDestinationName="queue1"
                  connectionFactory="#connectionFactory"/>

    <jms:endpoint service="test:queue2"
                  endpoint="myProvider"
                  role="provider"
                  destinationStyle="queue"
                  jmsProviderDestinationName="queue2"
                  connectionFactory="#connectionFactory"/>

</beans>
```
The JMS Sender

\[
\text{...}
\]

\[
<jms:endpoint service="test:queue3"
    endpoint="myProvider"
    role="provider"
    destinationStyle="queue"
    jmsProviderDestinationName="queue3"
    connectionFactory="#connectionFactory"/>
\]

\[
<amq:connectionFactory id="connectionFactory"
    brokerURL="tcp://localhost:61616" />
\]

\[
</beans>
\]
Example
JBI Packaging

Service Assembly (.jar/.zip)

Service Assembly:
1) Maven project (pom.xml)
2) Included Maven projects (pom.xml)

Service Unit (.jar/.zip)

Service Unit:
1) Maven project (pom.xml)
2) Spring XML (xbean.xml)
DEMO

- ServiceMix demo
Distribution of ServiceMix Containers
Eclipse IDE Tooling For ServiceMix

- Eclipse SOA Tooling Platform (STP) Project
  - http://eclipse.org/stp

- IONA FUSE Eclipse Tooling
  - http://open.iona.com/wiki/display/ProdInfo/FUSE+Eclipse+Tools

- Spagic
  - http://spagic.com/

- Sopera
Eclipse Tooling

Eclipse SOA Tools Platform (STP) Project
Lessons Learned
From JBI 1.0

✓ Normalized Exchange and Messages
✓ Normalized Message Router
  ✓ The only way to plug in third party components

○ XML normalization can be problematic
○ Packaging/classloaders not very flexible
○ Creating components is not easy
○ Not always the best fit for a given API
ServiceMix 4.0

Building Blocks

- Runtime: OSGi (Apache Felix)
  - JBI support still intact
  - NMR is an OSGi service
- Message Broker: Apache ActiveMQ
- Web Services: Apache CXF
- Routing Engine: Apache Camel
ServiceMix 4 Architecture

- Spring
- Web
- REST / JSR 311
- EJB
- ServiceMix Runtime
- OSGi
- JBI 1.0
- JBI 2.0
- Camel
- JAX-WS
Add the following to the SMX_HOME/etc/org.ops4j.pax.url.mvn.cfg file:

```xml
org.ops4j.pax.url.mvn.repositories=file:${user.home}/.m2/
people.apache.org/repo/m2-snapshot-repository@snapshots@noreleases
```

(you can do this while the ServiceMix Kernel is running)
servicemix> features addUrl mvn:org.apache.servicemix.nmr/apache-servicemix-nmr/1.0.0-m3-SNAPSHOT/xml/features
servicemix> features addUrl mvn:org.apache.servicemix.features/apache-servicemix/4.0-m2-SNAPSHOT/xml/features
servicemix> features addUrl mvn:org.apache.servicemix.camel/org.apache.servicemix.camel.features/4.0-m2-SNAPSHOT/xml/features
servicemix> features list

<table>
<thead>
<tr>
<th>State</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>[uninstalled]</td>
<td>camel-core</td>
</tr>
<tr>
<td>[uninstalled]</td>
<td>camel-spring</td>
</tr>
<tr>
<td>[uninstalled]</td>
<td>camel-osgi</td>
</tr>
<tr>
<td>[uninstalled]</td>
<td>camel-cxf</td>
</tr>
<tr>
<td>[uninstalled]</td>
<td>camel-http</td>
</tr>
<tr>
<td>[uninstalled]</td>
<td>camel-mina</td>
</tr>
<tr>
<td>[uninstalled]</td>
<td>camel-jetty</td>
</tr>
<tr>
<td>[uninstalled]</td>
<td>camel-jms</td>
</tr>
<tr>
<td>[uninstalled]</td>
<td>camel-amqp</td>
</tr>
<tr>
<td>[uninstalled]</td>
<td>camel-atom</td>
</tr>
<tr>
<td>[uninstalled]</td>
<td>camel-bam</td>
</tr>
<tr>
<td>[uninstalled]</td>
<td>camel-csv</td>
</tr>
<tr>
<td>[uninstalled]</td>
<td>camel-flatpack</td>
</tr>
<tr>
<td>[uninstalled]</td>
<td>camel-ftp</td>
</tr>
<tr>
<td>[uninstalled]</td>
<td>camel-groovy</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
servicemix> features install camel

servicemix> osgi list

START LEVEL 100

<table>
<thead>
<tr>
<th>ID</th>
<th>State</th>
<th>Level</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>[Active]</td>
<td>0</td>
<td>System Bundle (1.0.4)</td>
</tr>
<tr>
<td>1</td>
<td>[Active]</td>
<td>100</td>
<td>Apache ServiceMix Kernel :: GShell Core (1.0.0)</td>
</tr>
<tr>
<td>2</td>
<td>[Active]</td>
<td>50</td>
<td>Apache ServiceMix Bundles: aopalliance-1.0 (1.0.0.1)</td>
</tr>
<tr>
<td>3</td>
<td>[Active]</td>
<td>50</td>
<td>spring-osgi-io (1.1.0)</td>
</tr>
<tr>
<td>4</td>
<td>[Active]</td>
<td>50</td>
<td>Apache ServiceMix Bundles: cglib-2.1_3 (2.1.0.3_1)</td>
</tr>
<tr>
<td>5</td>
<td>[Active]</td>
<td>50</td>
<td>spring-osgi-core (1.1.0)</td>
</tr>
<tr>
<td>6</td>
<td>[Active]</td>
<td>50</td>
<td>Apache ServiceMix Kernel :: GShell OSGi Commands (1.0.0)</td>
</tr>
<tr>
<td>7</td>
<td>[Active]</td>
<td>50</td>
<td>spring-core (2.5.5)</td>
</tr>
<tr>
<td>8</td>
<td>[Active]</td>
<td>50</td>
<td>Apache ServiceMix Kernel :: JAAS Keystore (1.0.0)</td>
</tr>
<tr>
<td>9</td>
<td>[Active]</td>
<td>50</td>
<td>spring-context (2.5.5)</td>
</tr>
<tr>
<td>10</td>
<td>[Active]</td>
<td>50</td>
<td>Apache Felix Bundle Repository (1.0.2)</td>
</tr>
<tr>
<td>11</td>
<td>[Active]</td>
<td>50</td>
<td>Apache ServiceMix Kernel :: GShell OBR Commands (1.0.0)</td>
</tr>
<tr>
<td>12</td>
<td>[Active]</td>
<td>50</td>
<td>Apache ServiceMix Kernel :: Spring Deploier (1.0.0)</td>
</tr>
<tr>
<td>13</td>
<td>[Active]</td>
<td>50</td>
<td>Apache ServiceMix Bundles: ant-1.7.0 (1.7.0.1)</td>
</tr>
<tr>
<td>14</td>
<td>[Active]</td>
<td>50</td>
<td>spring-osgi-extender (1.1.0)</td>
</tr>
<tr>
<td>15</td>
<td>[Active]</td>
<td>50</td>
<td>spring-beans (2.5.5)</td>
</tr>
<tr>
<td>16</td>
<td>[Active]</td>
<td>50</td>
<td>Apache ServiceMix Bundles: mina-1.1.6 (1.1.6.1)</td>
</tr>
<tr>
<td>17</td>
<td>[Active]</td>
<td>50</td>
<td>Apache ServiceMix Kernel :: GShell Features (1.0.0)</td>
</tr>
<tr>
<td>18</td>
<td>[Active]</td>
<td>50</td>
<td>Apache ServiceMix Kernel :: GShell Log Commands (1.0.0)</td>
</tr>
<tr>
<td>19</td>
<td>[Active]</td>
<td>50</td>
<td>spring-aop (2.5.5)</td>
</tr>
<tr>
<td>20</td>
<td>[Active]</td>
<td>50</td>
<td>Apache ServiceMix Kernel :: JAAS Config (1.0.0)</td>
</tr>
<tr>
<td>21</td>
<td>[Active]</td>
<td>20</td>
<td>OPS4J - Pax Logging API (1.1.1)</td>
</tr>
<tr>
<td>22</td>
<td>[Active]</td>
<td>20</td>
<td>OPS4J - Pax Logging Service (1.1.1)</td>
</tr>
<tr>
<td>23</td>
<td>[Active]</td>
<td>15</td>
<td>Apache ServiceMix Kernel :: File Monitor (1.0.0)</td>
</tr>
<tr>
<td>24</td>
<td>[Active]</td>
<td>10</td>
<td>geronimo-servlet_2.5_spec (1.1.2)</td>
</tr>
<tr>
<td>25</td>
<td>[Active]</td>
<td>10</td>
<td>Apache Felix Preferences Service (1.0.2)</td>
</tr>
<tr>
<td>26</td>
<td>[Active]</td>
<td>10</td>
<td>Apache Felix Configuration Admin Service (1.0.0)</td>
</tr>
<tr>
<td>27</td>
<td>[Active]</td>
<td>10</td>
<td>OSGi R4 Compendium Bundle (4)</td>
</tr>
<tr>
<td>28</td>
<td>[Active]</td>
<td>5</td>
<td>OPS4J Pax Url - mvn: (0.3.2)</td>
</tr>
<tr>
<td>29</td>
<td>[Active]</td>
<td>50</td>
<td>spring-tx (2.5.5)</td>
</tr>
<tr>
<td>32</td>
<td>[Active]</td>
<td>50</td>
<td>Apache ServiceMix Specs :: STAX API 1.0 (1.1.0)</td>
</tr>
<tr>
<td>33</td>
<td>[Active]</td>
<td>50</td>
<td>geronimo-activation_1.1_spec (1.0.2)</td>
</tr>
<tr>
<td>34</td>
<td>[Active]</td>
<td>50</td>
<td>Apache ServiceMix Specs :: JAXB API 2.1 (1.1.0)</td>
</tr>
<tr>
<td>35</td>
<td>[Active]</td>
<td>50</td>
<td>Apache ServiceMix Bundles: jaxb-impl-2.1.6 (2.1.6.1)</td>
</tr>
<tr>
<td>36</td>
<td>[Active]</td>
<td>50</td>
<td>camel-core (1.4.0)</td>
</tr>
<tr>
<td>37</td>
<td>[Active]</td>
<td>50</td>
<td>camel-spring (1.4.0)</td>
</tr>
<tr>
<td>38</td>
<td>[Active]</td>
<td>50</td>
<td>camel-osgi (1.4.0)</td>
</tr>
</tbody>
</table>
What is Apache Camel?

http://activemq.apache.org/camel/
What is EIP?
Example Pattern: Content Based Router

```java
RouteBuilder builder = new RouteBuilder() {
    public void configure() {
        from("seda:a").choice()
            .when(header("foo").isEqualTo("bar")).to("seda:b")
            .when(header("foo").isEqualTo("cheese")).to("seda:c")
            .otherwise().to("seda:d");
    }
};
```
Example Pattern:
Content Based Router

```xml
<camelContext id="buildSimpleRouteWithChoice"
  xmlns="http://activemq.apache.org/camel/schema/spring">
  <route>
    <from uri="seda:a"/>
    <choice>
      <when>
        <predicate>
          <header name="foo"/>
          <isEqualTo value="bar"/>
        </predicate>
        <to uri="seda:b"/>
      </when>
      <when>
        <predicate>
          <header name="foo"/>
          <isEqualTo value="cheese"/>
        </predicate>
        <to uri="seda:c"/>
      </when>
      <otherwise><to uri="seda:d"/></otherwise>
    </choice>
  </route>
</camelContext>
```
Camel Makes Routing Much Easier!

```java
from("http://localhost:8080/requests/").
    tryBlock().
    to("activemq:queue:requests").
    setOutBody(constant("<ack/>")).
    handle(Throwable.class).
    setFaultBody(constant("<nack/>"));

from("activemq:queue:requests?transacted=true").
    process(requestTransformer).
    to("http://host:8080/Request").
    filter(xpath("//nack")).
    process(nackTransformer).
    to("jdbc:store");

from("http://localhost:8080/responses/").
    tryBlock().
    to("activemq:queue:responses").
    setOutBody(constant("<ack/>")).
    handle(Throwable.class).
    setFaultBody(constant("<nack/>"));

from("activemq:queue:responses?transacted=true").
    process(responseTransformer).
    to("jdbc:store");

from("http://localhost:8080/pull/").
    to("jdbc:load");
```
Thank You for Attending!

Questions?