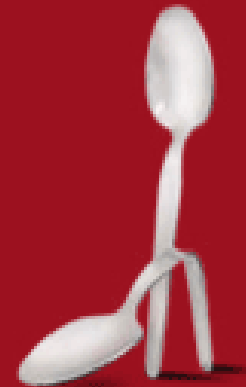


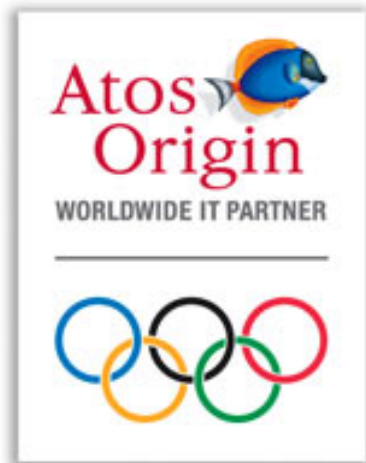
**IMPROVE YOUR  
PATTERNS AT  
JAVAPOLIS 2005.**

**METROPOLIS ANTWERP,  
DECEMBER 12 UNTIL 16.**





# Reliable open source ESB with Mule



Jos Dirksen  
Software Architect  
Atos Origin





• *Exploring the concepts of Mule*

**DEMO**

• Integrating Mule with JMS and Axis

DEMO

• Mule in relation to: JBI, BPEL, JMX...

• Questions and Olympic tickets





# What is Mule?

- Open Source Enterprise Service Bus (ESB):
  - Routing, transformation, security, WS support, logging.
- Out of the box supported transports:
  - JMS, VM, JDBC, TCP, UDP, Multicast, HTTP, Servlet, Email, File, XMPP, FTP, EJB, VFS and more...
- Support for other interesting technologies:
  - BPEL, JBI, Axis, Spring, JSR-223 Scripting, JBoss

*“Mule is a light-weight messaging framework”*





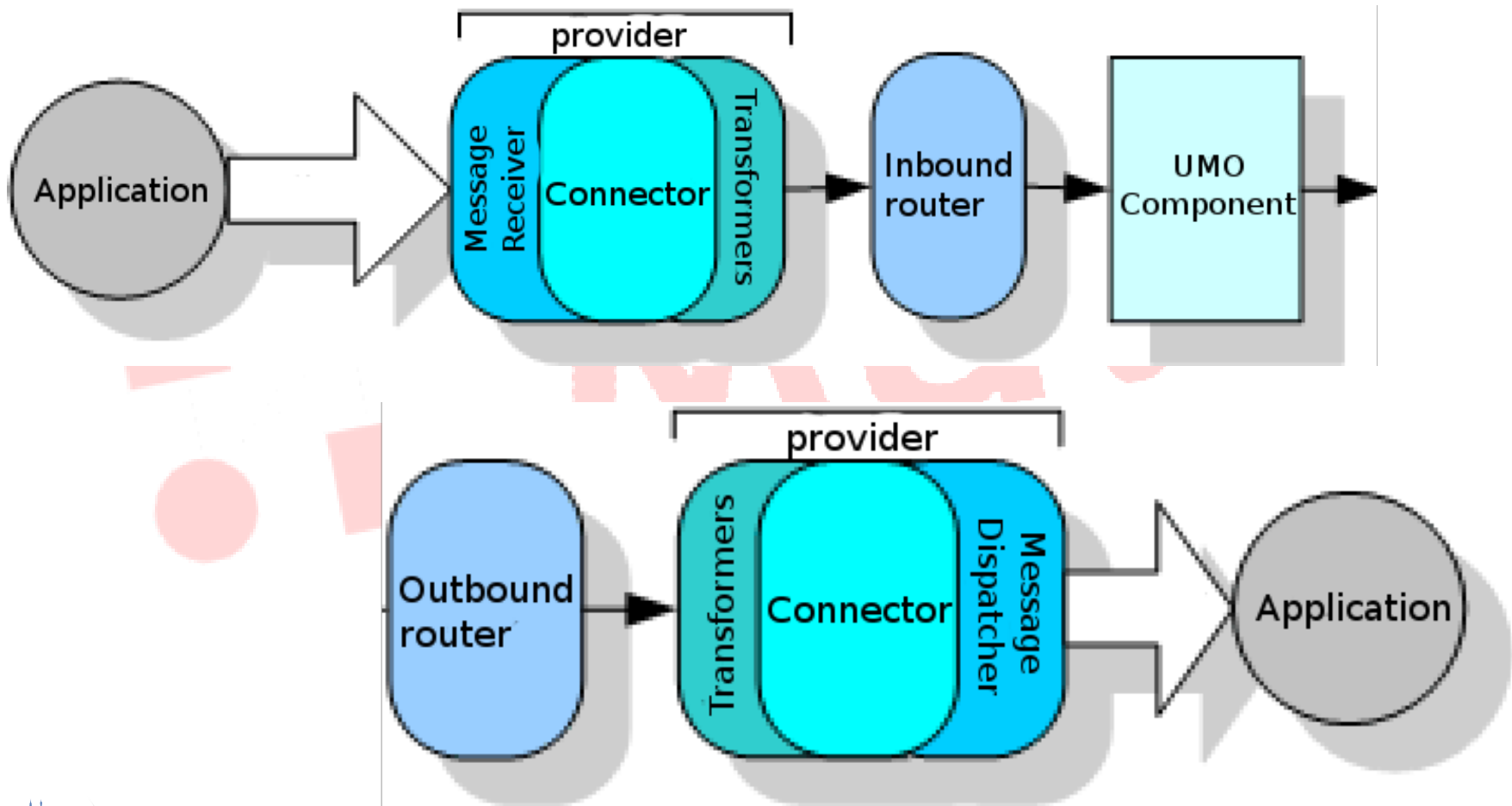
# The mule project

- 10+ developers who support it
- Active user community: mailing lists, wiki
- Used by companies such as: HP, Sony, Deutsche Bank, CitiBank and Atos Origin
- Commercial support available from SymphonySoft

More information: <http://mule.codehaus.org>



# Mule architecture





- Components
  - POJO based, contains the business logic.
  - Defined in the mule configuration file.
- Endpoints
  - Used to connect components and external systems together.
  - `jms://queue:reader.in?connector=jmsAMQConnector`

```
<mule-descriptor name="Component name"
  inboundEndpoint="jms://in.queue"
  outboundEndpoint="vm://internal.queue"
  implementation="org.jp.Concrete"/>
```





- **Transformers**

- Convert data from one format to the other.

```
inboundEndpoint="file:///C:/temp?transformers=XmlToDom"
```

- **Message routers**

- Control how events are received and sent by the system.
- Inbound: SelectiveConsumer, Aggregator
- Outbound: FilteringOutboundRouter, RecipientList

```
<router className="outbound.FilteringOutboundRouter">
  <endpoint address="jms://queue:error.queue"/>
  <filter expression="error='true'" className="JXPathFilter"/>
</router>
```



# Mule and Spring (1)

- Supports multiple IoC containers:
  - Spring, PicoContainer, NanoContainer and Plexus, HiveMind

```
<container-context
  className='org.mule.extras.spring.SpringContainerContext'
  name="spring">
  <properties>
    <!-- point to the location of the configuration file(s)
    where we configure our spring beans -->
    <property name='configFile'
              value='mule/component-config.xml' />
  </properties>
</container-context>
```





# Mule and Spring (2)

- We can now let Spring manage our components.

```
<beans>
  <bean id="component1" class="test.SimpleComponent1">
    <property name="timeout" value="1000"/>
  </bean>
  <bean id="component2" class="test.SimpleComponent2">
    <property name="foo" ref="component1"/>
  </bean>
</beans>
```

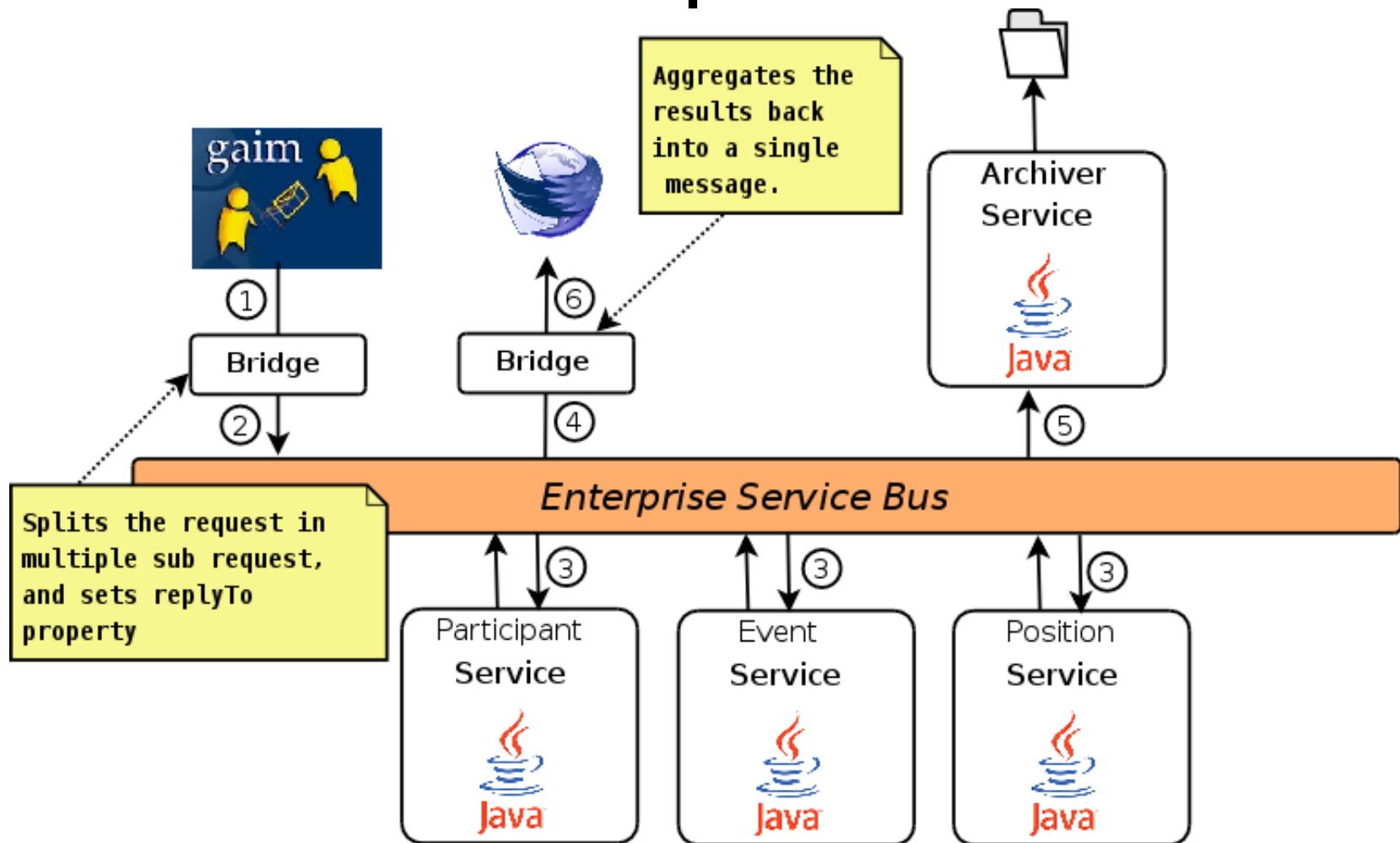
- And use them directly in the mule configuration.

```
<mule-descriptor name="Component name"
  inboundEndpoint="jms://in.queue"
  outboundEndpoint="vm://internal.queue"
  implementation="component1"/>
```





# Demo: basic setup





# End of introduction

- Mule has excellent spring integration
- POJO based
- Important concepts used are:
  - Components
  - Endpoints
  - Transformers
  - Routers

Mule



- Exploring the concepts of Mule DEMO
- *Integrating Mule with JMS and Axis* DEMO

**ActiveMQ**

axis



# Mule with ActiveMQ and Axis



- ActiveMQ

- Open source JMS Provider
- Persistency to filesystem, JDBC
- Clustering
- Supports various transports: VM, TCP, UDP, SSL

- Axis

- De facto web services library
- Provides a Web service layer around our components
- Automatic serialization / deserialization of java beans





# Configuring ActiveMQ

```
<broker>
  <connector>
    <tcpServerTransport uri="tcp://localhost:61616"
                        backlog="1000"
                        useAsyncSend="true"/>
  </connector>
  <persistence>
    <jdbcPersistence dataSourceRef="derby-ds"/>
  </persistence>
</broker>
```



# Configuring ActiveMQ for Mule

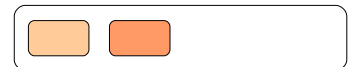
- To use this a connection factory is required

```
<bean id="activeMqConnectionFactory"
      class="org.activemq.ActiveMQConnectionFactory">
  <property name="brokerURL"
            value="tcp://localhost:62001"/>
</bean>                                {Configured in spring}

<connector name="jmsConnector"
           className="org.mule.providers.jms.JmsConnector">
  <properties>
    <property name="specification" value="1.1"/>
    <container-property name="connectionFactory"
                       reference="activeMqConnectionFactory"/>
  </properties>
</connector>                            {configured in mule}
```

- We can now use it directly from mule

```
inboundEndpoint="jms://in.queue"
```





# Using Axis from Mule

- Invoking webservices from Axis

```
outboundEndpoint=
  "axis:http://localhost:81/services/myComp?method=hello"
  "axis:jms://ws.queue/myComp?method=hello"
  "axis:vm://echo?method=hello"
  "axis:xmpp://mule1:mule@jabber.org.au/axis?method=echo"
```

- Configuring Mule Components As Axis Services

```
<mule-descriptor
  name="myComp"
  implementation="myImplementation"
  inboundEndpoint="axis:http://localhost:81/services">
</mule-descriptor>
```

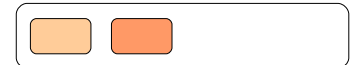
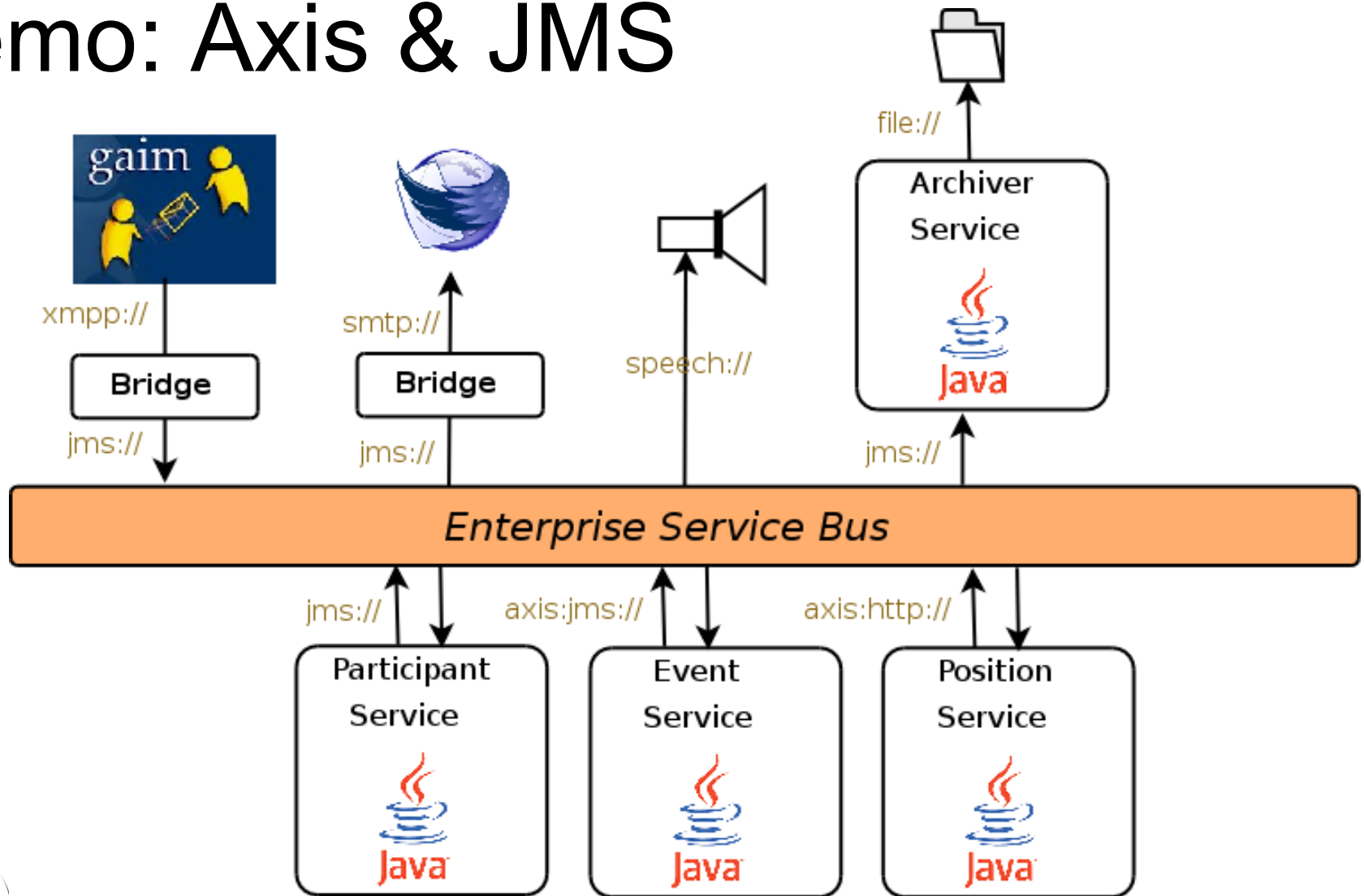


# Controlling Axis settings

```
<mule-descriptor
  name="myComp"
  implementation="myImplementation"
  inboundEndpoint="axis:http://localhost:81/services">
<properties>
  <property name="style" value="Wrapped"/>
  <property name="use" value="Literal"/>
  <map name="axisOptions">
    <property name="allowedMethods"
      value="echo,getDate"/>
  </map>
  <list name="serviceInterfaces">
    <entry value="org.mule.EchoService"/>
    <entry value="org.mule.DateService"/>
  </list>
</properties>
</mule-descriptor>
```

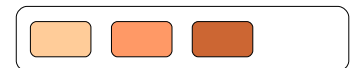


# Demo: Axis & JMS





- Exploring the concepts of Mule DEMO
- Integrating Mule with JMS and Axis DEMO
- *Mule in relation to: JBI, BPEL, JMX...*





# JMX

- Java Management Extensions

```
<agent name="JMX" className="...gement.agents.JmxAgent">  
<agent name="JdmkAgent" className="...gement.agents.JdmkAgent">  
  <properties>  
    <property name="jmxAdaptorUrl"  
      value="http://localhost:9998"/>  
  </properties>  
</agent>
```

- Stop and start the Mule instance.
- Stop, start, pause and resume components.
- Statistics: received events, endpoint routing stats etc.



# JBI and Mule

- Mule is not a JBI container
  - A sub project called Mule JBI is working on this
- Mule can already work with JBI
  - Integrating with Servicemix
  - Transports, components and transformers can be used
- Main differences
  - JBI focusses on XML as message data
  - In Mule components are not linked to any mule API
  - Mule is the ‘Spring Framework’ of integration



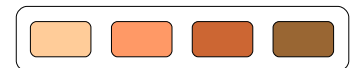
# BPEL and other technologies

- BPEL support using PXE (<http://pxe.fivesight.com>):
  - Start BPEL applications from a mule endpoint
- Quartz:
  - `quartz://service?repeatInterval=500`
- EJB:
  - `ejb://localhost:1099/AService?method=aMethod`
- Security: uses the Spring / Acegi security framework
- Config Graph: Creates graph from mule configuration files.
- Integration with JBoss and Geronimo:
  - Mule JCA connector
  - Through JBossMQ or ActiveMQ



# Questions ?

- Exploring the concepts of Mule DEMO
- Integrating Mule with JMS and Axis DEMO
- Mule in relation to: JBI, BPEL, JMX...
- *Questions and Olympic tickets*





**IMPROVE YOUR  
PATTERNS AT  
JAVAPOLIS 2005.**

**METROPOLIS ANTWERP,  
DECEMBER 12 UNTIL 16.**

