Apache ServiceMix

Presented by :
Srinivas Vothukudi Govindaraj
University of Colorado Boulder
03/22/2011

Agenda

- Pre Requisites 
- Terms and Definitions 
- Architecture 
- Use case 
- Demo 
- References 

- Questions 
- Support 

Pre Requisites

- Service Oriented Architecture (SOA)
- What is Enterprise Service Bus (ESB)
- What is Java Business Integration (JBI)
- Spring Framework
- Maven
- SOAP

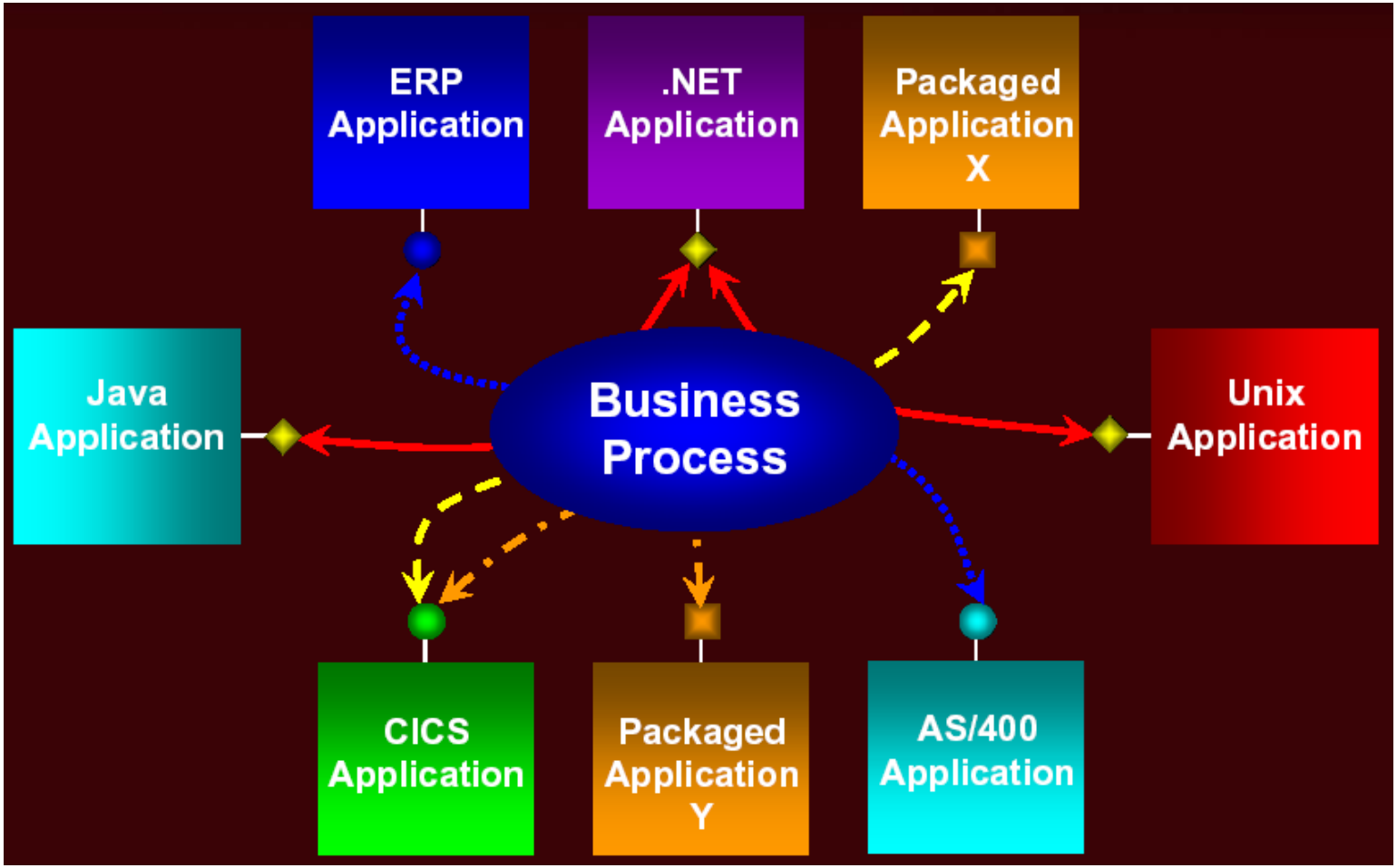
Terms & Definitions

- Enterprise Service Bus (ESB)
- Java Business Integration(JBI)
- Normalized Message Router (NMR)
- Binding Component (BC)
- Service Engine (SE)
- Message Exchange Patterns(MEP)
- Service Unit (SU)
- Service Assembly (SA)

A bit about Service Oriented Architecture:

- SOA is an underlying computer systems structure that supports the connection between various applications and the sharing of data
- Service-oriented architecture (SOA) helps the business processes be better, easier to change and cheaper to create
- Building business processes is faster and cheaper because the existing services can more easily be reused, applications can expose their services in a standard way
- The next slide shows the illustration of Service Oriented Architecture

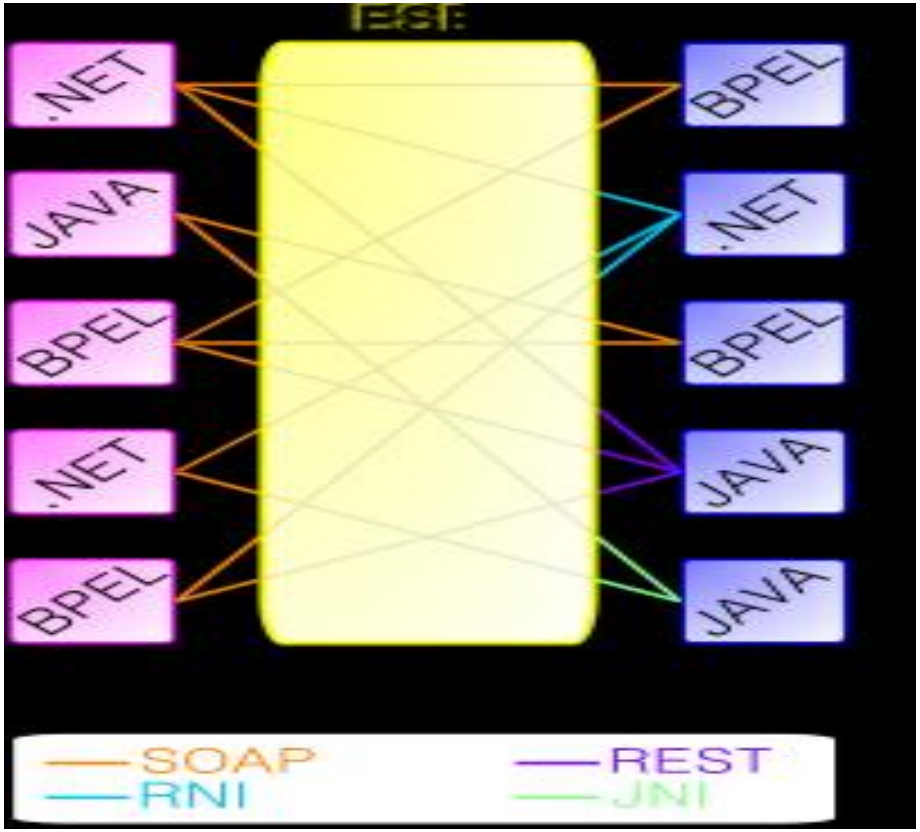
Service Oriented Architecture(II):



Enterprise Service Bus (I)

- Middleware architecture which helps in integrating the business functions and specific applications
- Identifies messages and routes them between applications and services
- Provides an abstraction for endpoints
- Enables loose coupling and easy connection between services
- Based on pluggable components

Enterprise Service Bus (II)



Source: http://en.wikipedia.org/wiki/File:ESB.svg

Java Business Integration

- Java standard for structuring business integration systems along SOA lines
- Defines an environment for plug-in components that interact using a services model based directly on WSDL 2.0
- Viewed as Container of Containers
- Integration model based on WSDL
- JBI components can function as a service provider or a service consumer or both

JBI Messaging Architecture(I)

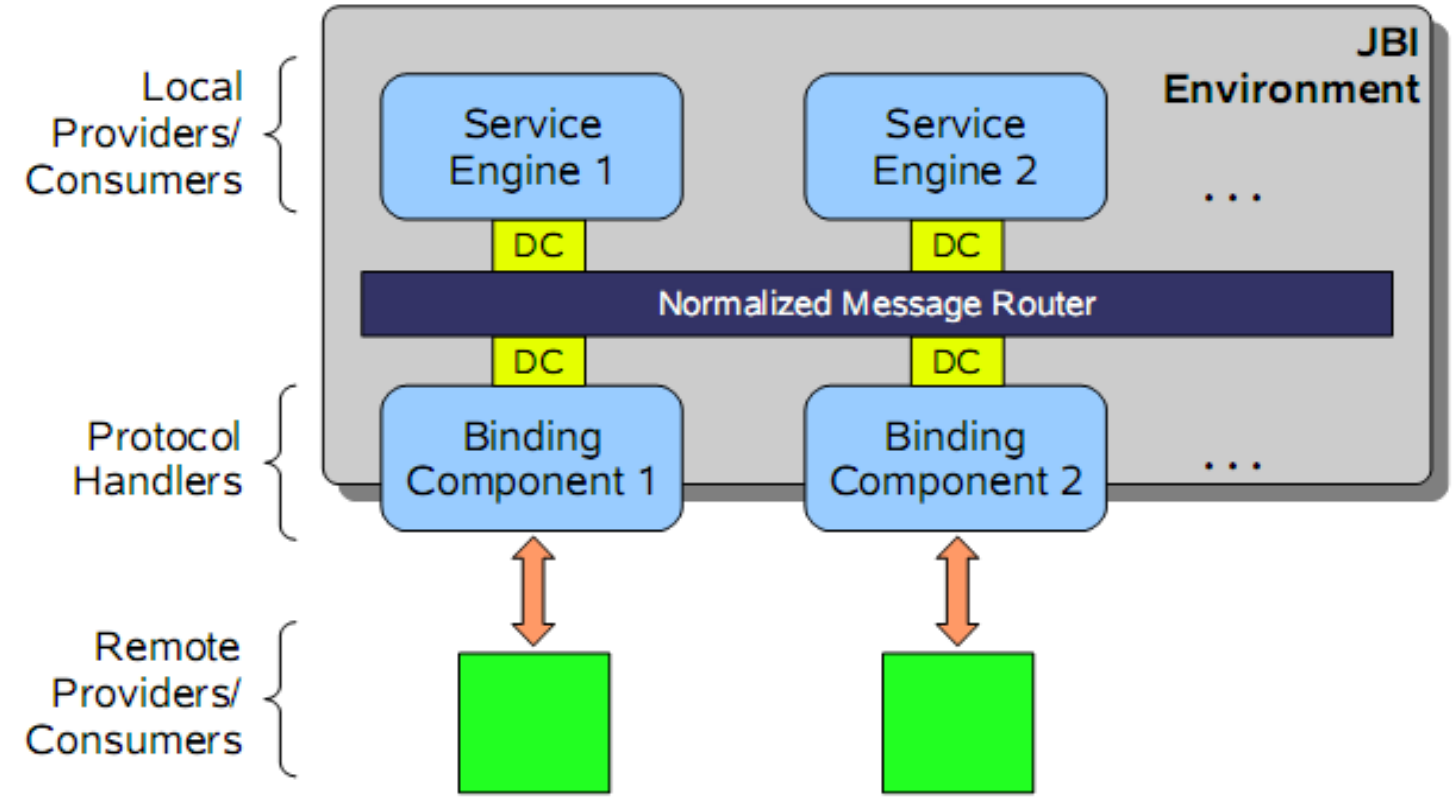


Figure 1: JBI Messaging Architecture

Source: JBI Components: Part 1 (Theory) by Ron Ten-Hove, Sun Microsystems, 2006

JBI Messaging Architecture(II)

- Consists of Binding Components , Service Engines and a Normalized Message Router
- Binding Component provides connectivity for applications that are external to the JBI
- Service Engine provide business logic and transformation services
- The separation of business and processing logic from communication logic makes the implementation of components much less complex

JBI Messaging Architecture(II)

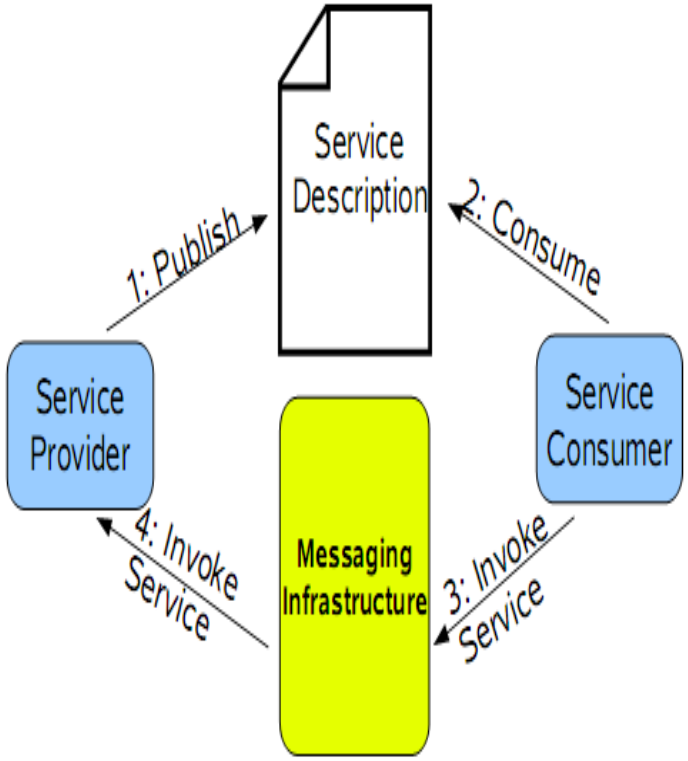
- Normalized Message Router is a light weight messaging infrastructure
- NMR helps in mediated message exchange between components
- NMR routes normalized messages between service providers and consumers
- A normalized message consists of two parts: the message content (payload) and the message metadata
- A message can be routed through several JBI components depending on what processing is needed

JBI Messaging Architecture(III)

- JBI component interact with each other via a process of message exchange, which is fully described WSDL documents, as published by the service provider
- Each operation that a service provider makes available has a particular message exchange pattern(MEP) associated with it
- An MEP defines the sequence, direction, and cardinality of all message exchanges that occur in the course of invoking and performing the operation
- JBI defines four MEPs: in-only, robust in-only, in-out, and in-optional-out

JBI Messaging Architecture(IV)

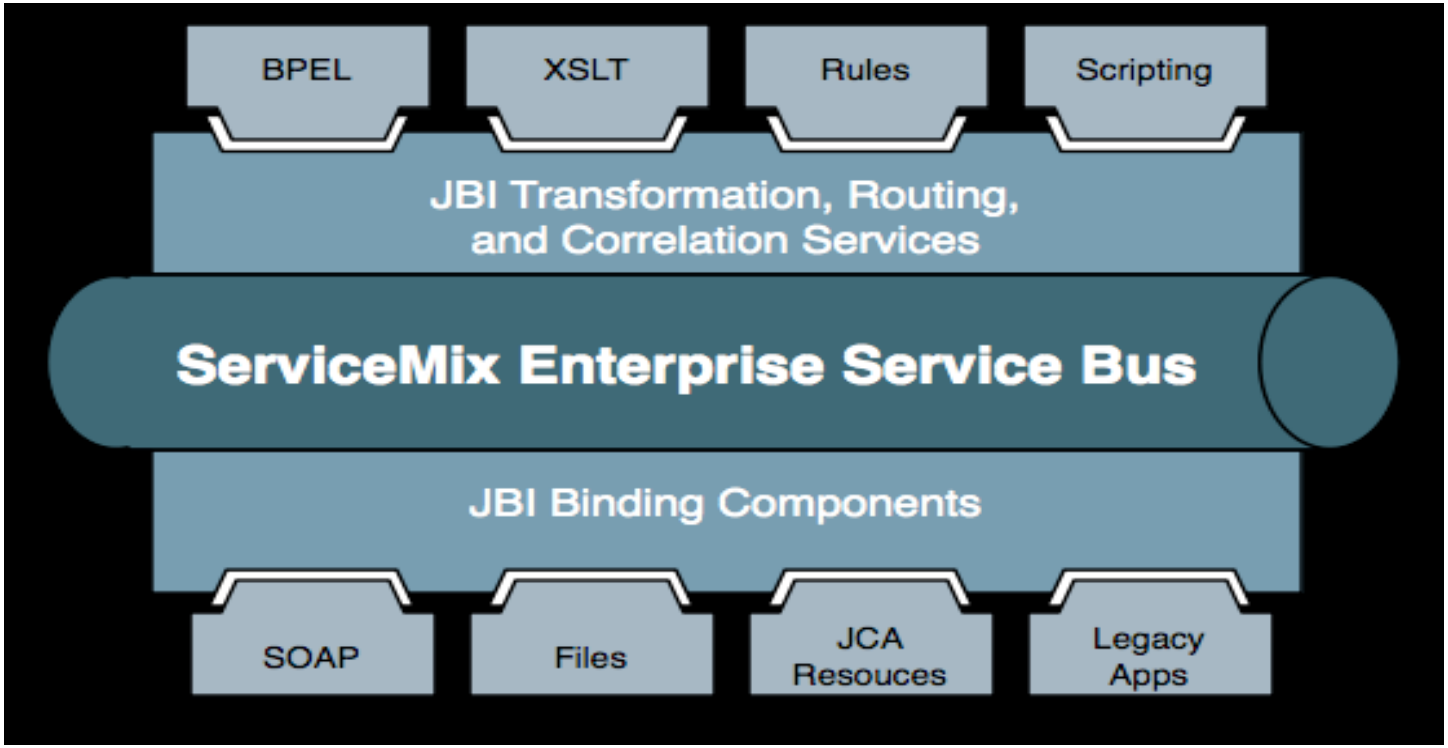
- Service descriptions in JBI are made using the web services description language (WSDL).
- These descriptions are published by service providers and read by service consumers
- Service consumer sends an appropriate request message, via a messaging infrastructure (in this case JBI), to the service provider.



ServiceMix

- Open Source Enterprise Service Bus
- Built on Java Business Integration specification
- Lightweight , easily embedded and has integrated Spring Support
- Combines functionality of Service Oriented Architecture and Event Driven Architecture
- Provides clustering, high availability, remote access, failover
- Provides extensive JMX support for management

ServiceMix Architecture



Source: http://servicemix.apache.org/home.data/ServiceMix3.png

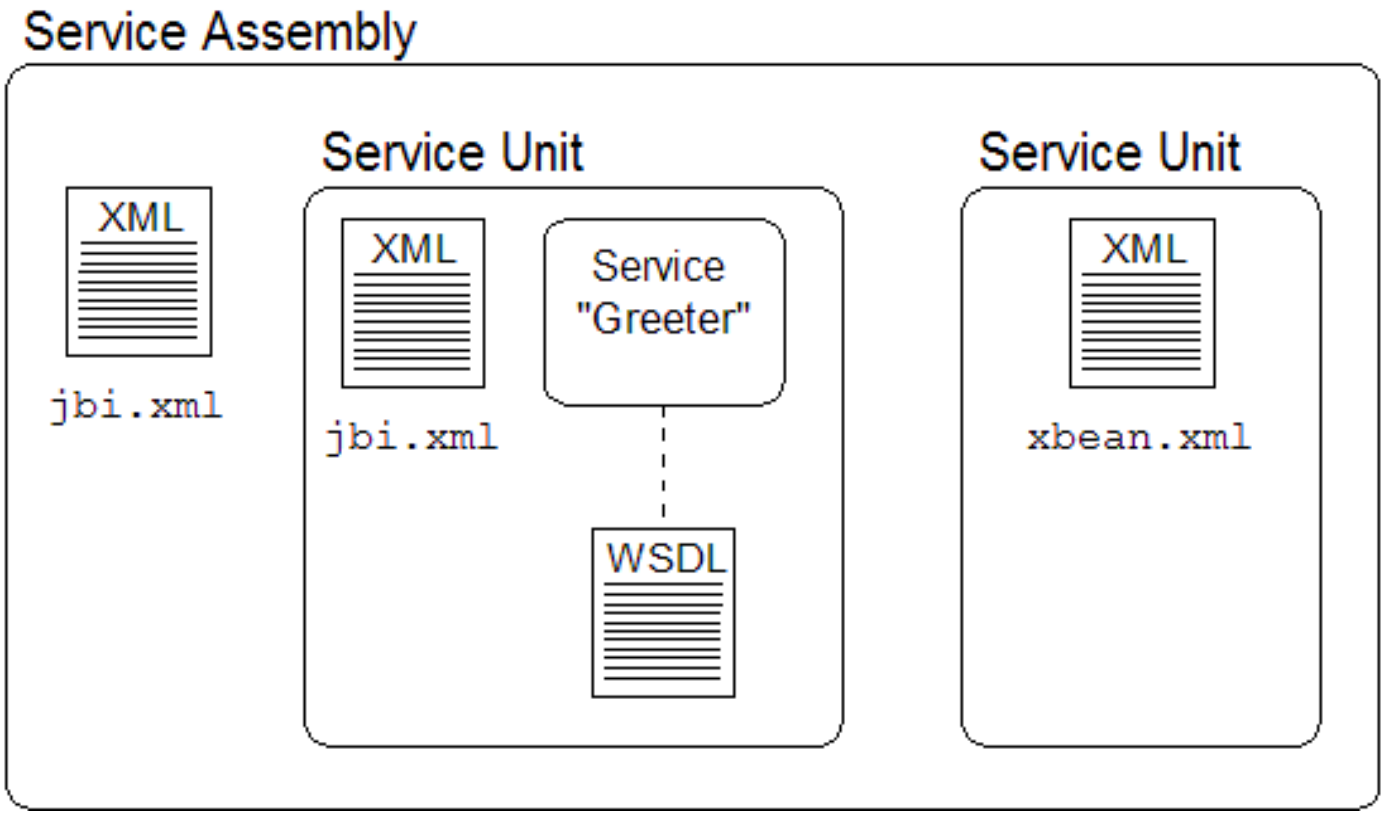
ServiceMix JBI Components

- servicemix-bean
- servicemix-camel
- servicemix-cxf-bc
- servicemix-cxf-se
- servicemix-eip
- servicemix-file
- servicemix-ftp
- servicemix-http
- servicemix-jms
- servicemix-mail
- servicemix-quartz
- servicemix-saxon

Terms used for packaging:

- Service Unit(SU) contains one or more components of same implementation type/binding type
- Service Units are useful when you want to distribute/load balance deployment of your application
- Service Assembly(SA) is a collection of one or more service units
- In order to deploy a service into a JBI container, it is necessary to package the files into a service assembly file

Depiction of Service Assembly:



- Source: http://documentation.progress.com/output/Iona/celtix/1.1/getting_started/N049B4680.04CE80E8.html

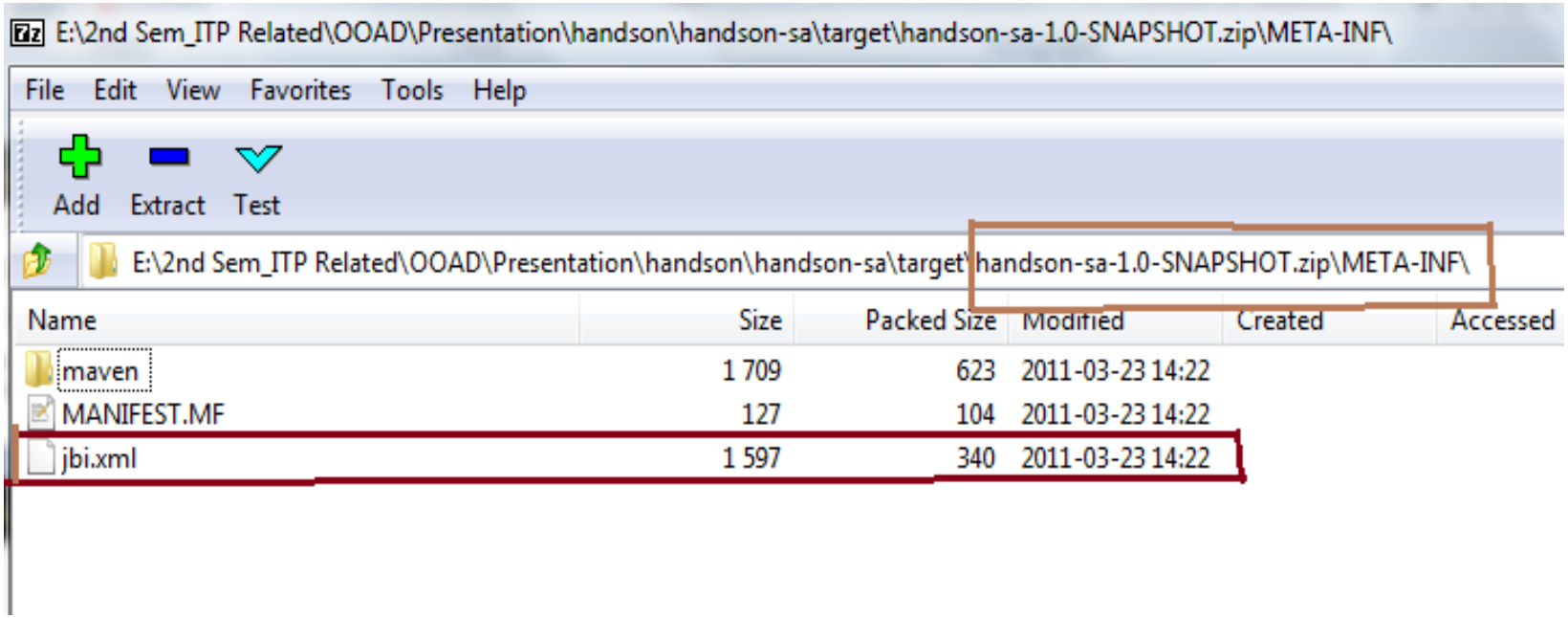
Service Assembly Deployment Descriptor:

- As in the previous slide, SA consists of a jbi.xml which contains sequence of service unit descriptions.
- For each service unit, the descriptor specifies the constituent files and indicates which target JBI component the service should be deployed into.
- These jbi.xml's get generated only after the generation of Service Assembly and Service Units.
- These jbi.xml's are present in the META-INF folder.
- The next slide shows jbi.xml for SA having two service units namely eip-bc and http-bc.

Sample jbi.xml of a Service Assembly:

```
<?xml version="1.0" encoding="UTF-8"?>
<jbi xmlns="http://java.sun.com/xml/ns/jbi" version="1.0">
 <service-assembly>
  <identification>
   <name>handson-sa</name>
   <description>Apache ServiceMix :: Service Assembly</description>
  </identification>
  <service-unit>
   <identification>
    <name>http-bc-su</name>
    <description>http-bc-su</description>
   </identification>
   <target>
    <artifacts-zip>http-bc-su-1.0-SNAPSHOT.zip</artifacts-zip>
    <component-name>servicemix-http</component-name>
   </target>
  </service-unit>
  <service-unit>
   <identification>
    <name>eip-su</name>
    <description>eip-su</description>
   </identification>
   <target>
    <artifacts-zip>eip-su-1.0-SNAPSHOT.zip</artifacts-zip>
    <component-name>servicemix-eip</component-name>
   </target>
  </service-unit>
 </service-assembly>
</jbi>
```

SnapShot of sample jbi.xml (II)



- As shown, the handson-sa.zip contains a META-INF folder.
- In this folder, there is jbi.xml describing the components used for the handson service assembly.

Use Case

- User sends a SOAP/HTTP request

- User expects the soap message to be transformed according to the style sheet

- System dumps result in a configured directory

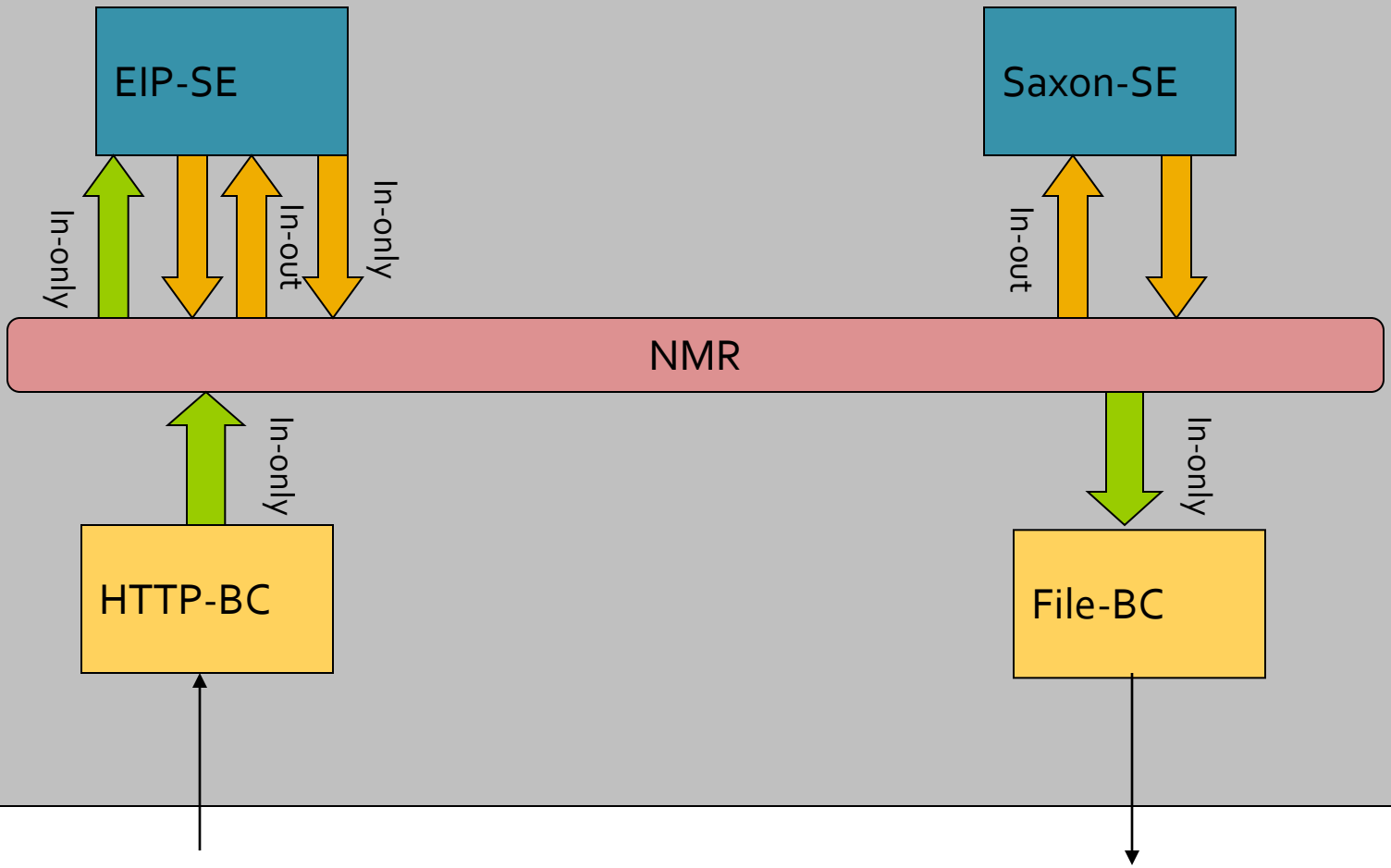
Scenario

- Send SOAP request to the HTTP BC
- HTTP-BC sends IN-ONLY message to EIP(Enterprise Integration Pattern) Service Engine component
- EIP-SE converts the IN-Only to IN-Out MEP(Message Exchange Patterns)
- The IN-Out message is sent to Saxon Component

Cont'd…

- The Saxon-SE transforms the message and sends the response as a OUT message to the EIP component
- The EIP receives the response and sends it as a IN-Only message to the File component
- The response is dumped in form of an xml file at a configured directory
- The following slides shows the maven commands for generating the binding components

JBI Enviornment



Implementation (I)

- Create a parent POM
- Create SU for HTTP-BC
- Create SU for FILE-BC
- Create SU for EIP-SE
- Create SU for SAXON-SE
- Add a dummy style sheet to the Saxon-SU
- Create a SA for the SU's

Implementation (II)

- Configure the endpoints in the SU's
- Add the dependencies in the SA
- Build the SA
- Deploy the SA in Servicemix
- Send a SOAP/HTTP request provided by Servicemix
- Check for response in the configured directory

Steps(I):Create a parent POM

- <?xml version="1.0" encoding="UTF-8"?>

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
    http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.csci.ooad.handson</groupId>
  <artifactId>handson-example</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>pom</packaging>
  <name>Simple example</name>
  <url>http://www.ooad.com</url>
</project>
```

Steps (II):Create SU for components

- For HTTP-BC SU:
- mvn archetype:create -DarchetypeArtifactId=**servicemix-http-consumer-service-unit** -DarchetypeGroupId=org.apache.servicemix.tooling -DartifactId=http-bc-su -DgroupId=com.csci.ooad.handson

- For File-BC SU
- mvn archetype:create -DarchetypeArtifactId=**servicemix-file-sender-service-unit** -DarchetypeGroupId=org.apache.servicemix.tooling -DartifactId=file-bc-su -DgroupId=com.csci.ooad.handson

- For EIP-SE SU
- mvn archetype:create -DarchetypeArtifactId=**servicemix-eip-service-unit** -DarchetypeGroupId=org.apache.servicemix.tooling -DartifactId=eip-su -DgroupId=com.csci.ooad.handson

Cont'd…

- For SAXON-SE SU
- mvn archetype:create -DarchetypeArtifactId=**servicemix-saxon-xslt-service-unit** -DarchetypeGroupId=org.apache.servicemix.tooling -DartifactId=saxon-su -DgroupId=com.csci.ooad.handson

- For Service Assembly
- mvn archetype:create -DarchetypeArtifactId=**servicemix-service-assembly** -DarchetypeGroupId=org.apache.servicemix.tooling -DartifactId=handson-sa -DgroupId=com.csci.ooad.handson

Steps(III): Updating the poms

- Open the pom.xml for all the Service Units and make sure the component version in the dependency section are same
- Also the jbi-maven-plugin version to 4.0 should be same across all pom's

Steps(IV): Configure Endpoints For HTTP-SU

- The binding component HTTP-BC receives HTTP request configured at a port, extracts the soap message and sends it to the target service i.e. to the EIP-SU

- Open the xbean.xml file in src/resources folder and do the following changes:
  - Provide a name for the service and endpoint
  - Provide the values for target Service and target endpoint.

  The message is present here

- Change the IN-Out Mep to IN-Only Mep

Cont'd…

- <beans xmlns:http="http://servicemix.apache.org/http/1.0"
  xmlns:handson="http://servicemix.apache.org/handson"
  xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://servicemix.apache.org/http/1.0
  http://servicemix.apache.org/schema/servicemix-http-3.2.2.xsd
  http://www.springframework.org/schema/beans
  http://www.springframework.org/schema/beans/spring-beans-2.0.xsd">

<http:endpoint service="**handson:httpService**"
endpoint="soap"
targetService ="handson:pipeline"
targetEndpoint="endpoint"
role="consumer"
locationURI="http://localhost:8192/example/"
defaultMep="http://www.w3.org/2004/08/wsdl/in-only"
soap="true" />
</beans>

Steps(V): Configure Endpoints For EIP-SU

- EIP component converts the In-Only MEP from the HTTP-BC to an IN-Out MEP and send it as an In-out to Saxon-SE
- The response from the Saxon-SE is sent to the file sender as an InOnly message
- Open the xbean.xml file in src/resources folder and do the following changes:
  - Provide a name for the service and endpoint
  - Provide the value for target Service in the transformer section. This is the service which will receive the message as an IN-Out MEP
  - Provide the value for target Service in the target section. This is the service which will retrieve the response of the transformer component as an IN-Only MEP

.

Cont'd…

```
<beans xmlns:eip="http://servicemix.apache.org/eip/1.0"

    xmlns:handson="http://servicemix.apache.org/handson"

    xmlns="http://www.springframework.org/schema/beans"

    xmlns:xsi="http://http://www.w3.org/2001/XMLSchema-instance"

    xsi:schemaLocation="http://servicemix.apache.org/eip/1.0
      http://servicemix.apache.org/schema/servicemix-eip-3.2.2.xsd
    http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-
      beans-2.0.xsd">

 <eip:pipeline service="handson:pipeline" endpoint="endpoint">

   <eip:transformer>

    <eip:exchange-target service="handson:xslt" />
   </eip:transformer>
   <eip:target>
    <eip:exchange-target service="handson:fileSender" />
   </eip:target>
 </eip:pipeline>

</beans>
```

Steps(VI):Configure Endpoint for Saxon-SU

- Saxon component receives an xml message as an Inout MEP sends the transformed response as the Out message

- Open the xbean.xml file in src/resources folder and do the following changes:
  - Provide a name for the service and endpoint
  - Provide the location to an xslt stylesheet. This style sheet will be used to transform the incoming xml

Cont'd…

- <beans xmlns:saxon="http://servicemix.apache.org/saxon/1.0"
  xmlns:handson="http://servicemix.apache.org/handson"
  xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://servicemix.apache.org/saxon/1.0
  http://servicemix.apache.org/schema/servicemix-saxon-3.2.2.xsd
  http://www.springframework.org/schema/beans
  http://www.springframework.org/schema/beans/spring-beans-2.0.xsd">

```
<!-- START SNIPPET: xslt -->
<saxon:xslt service="handson:xslt" endpoint="endpoint"
        resource="classpath:transform.xsl" />
<!-- END SNIPPET: xslt -->
```

Steps(VII):Configure Endpoint for File-BC SU

- The file component receives the transformed content from the EIP-su and dumps it into a configured directory.

- Open the xbean.xml file in src/resources folder and do the following changes:
  - Provide a name for the service and endpoint.
  - Provide the name of the directory where the file would be dumped.

Cont'd…

- <beans xmlns:file="http://servicemix.apache.org/file/1.0"
  xmlns:handson="http://servicemix.apache.org/handson"
  xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://servicemix.apache.org/file/1.0
  http://servicemix.apache.org/schema/servicemix-file-3.2.2.xsd
  http://www.springframework.org/schema/beans
  http://www.springframework.org/schema/beans/spring-beans-2.0.xsd">
  <file:sender **service="handson:fileSender"**
  endpoint="endpoint"
  directory="file:D:/hands onnew/">
  </file:sender>
  </beans>

Steps(VIII):Configure the SA

- Edit the pom.xml of the service assembly and add the following to the dependencies section. This indicates the SU's that are part of the SA

```
<dependency>
        <groupId>com. csci.ooad.handson</groupId>
        <artifactId>http-bc-su</artifactId>
        <version>1.0-SNAPSHOT</version>
</dependency>
<dependency>
        <groupId>com. csci.ooad.handson</groupId>
        <artifactId>eip-su</artifactId>
        <version>1.0-SNAPSHOT</version>
</dependency>
<dependency>
        <groupId>com. csci.ooad.handson</groupId>
        <artifactId>saxon-su</artifactId>
        <version>1.0-SNAPSHOT</version>
</dependency>
<dependency>
        <groupId>com. csci.ooad.handson</groupId>
        <artifactId>file-bc-su</artifactId>
        <version>1.0-SNAPSHOT</version>
    </dependency>
```

Steps(IX):Build and deploy

- Go to each of the folders where the pom exists
- Run the command "mvn clean install" on each of the folders
- If the build is successful there should be a handson-sa.jar created in the SA folder
- Start Servicemix by executing the servicemix.bat present in bin folder of the installed servicemix
- Copy the jar/zip and place it in the hotdeploy folder of servicemix

Steps(X):Build and deploy

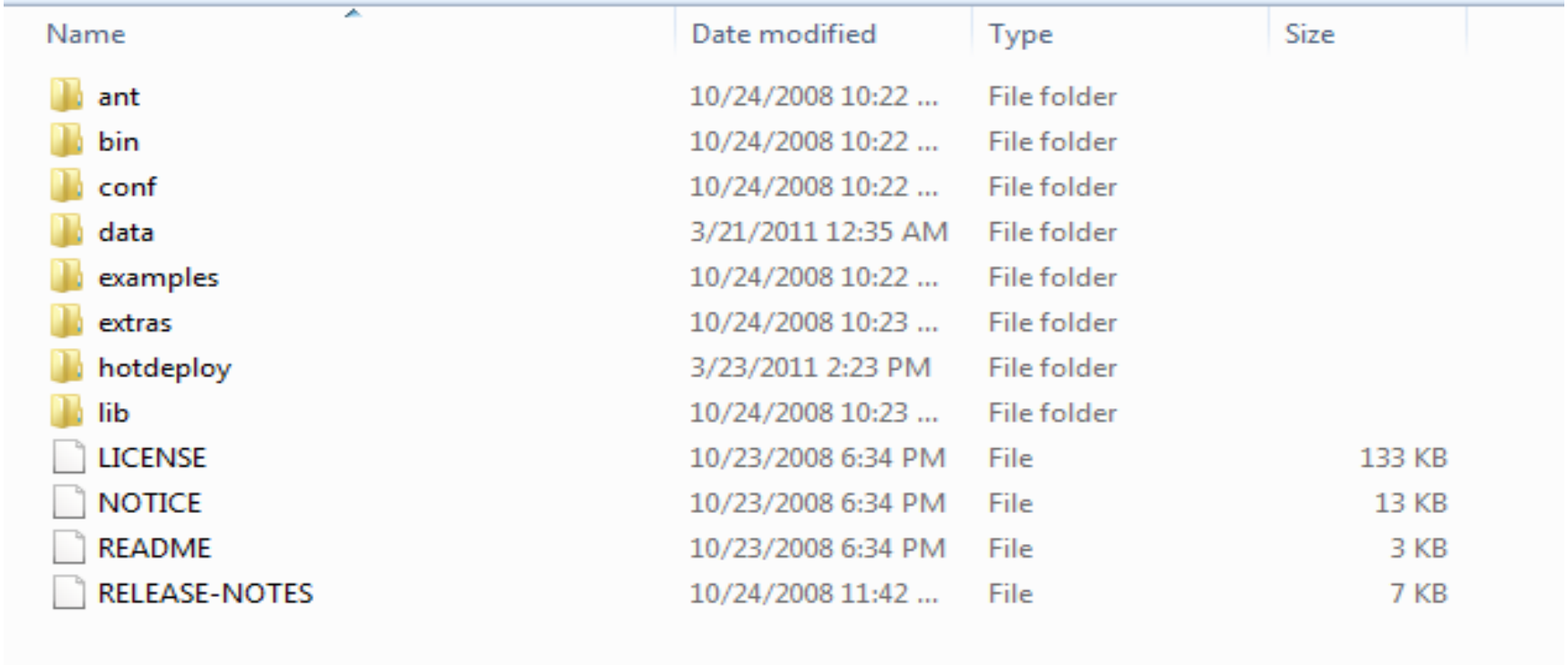
- Send a SOAP request using the attached client.html and check the presence of the file in the output directory, D:/handsonnew/

Demo

- The handson service assembly component gets deployed in servicemix
- A transformed xml will be present in D:/handsonnew/

SnapShot(I)

- The directory structure of service mix



Name	Date modified	Type	Size
ant	10/24/2008 10:22 ...	File folder	
bin	10/24/2008 10:22 ...	File folder	
conf	10/24/2008 10:22 ...	File folder	
data	3/21/2011 12:35 AM	File folder	
examples	10/24/2008 10:22 ...	File folder	
extras	10/24/2008 10:23 ...	File folder	
hotdeploy	3/23/2011 2:23 PM	File folder	
lib	10/24/2008 10:23 ...	File folder	
LICENSE	10/23/2008 6:34 PM	File	133 KB
NOTICE	10/23/2008 6:34 PM	File	13 KB
README	10/23/2008 6:34 PM	File	3 KB
RELEASE-NOTES	10/24/2008 11:42 ...	File	7 KB

SnapShot(II)

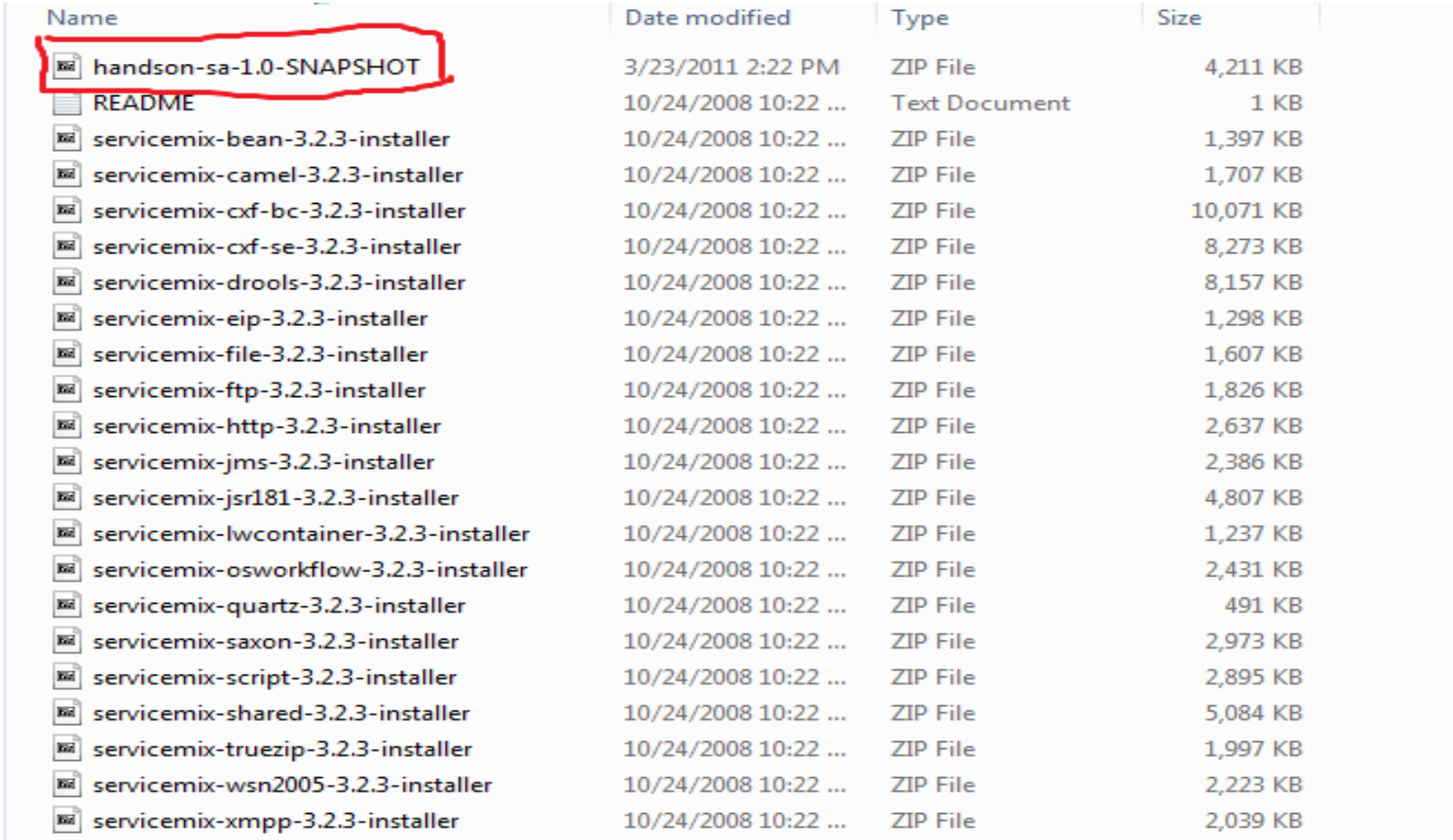
- Servicemix service started by executing the servicemix.bat present in bin folder

```
Starting Apache ServiceMix ESB: 3.2.3

Loading Apache ServiceMix from servicemix.xml on the CLASSPATH
INFO  - ConnectorServerFactoryBean      - JMX connector available at: service:jmx:rmi:///jndi/rmi://localhos
INFO  - JBIContainer                    - ServiceMix 3.2.3 JBI Container (ServiceMix) is starting
INFO  - JBIContainer                    - For help or more information please see: http://servicemix.apache.
INFO  - JCAFlow                         - ActiveMQResourceAdapter server url was null.  Setting it to: tcp:/
INFO  - ComponentMBeanImpl              - Initializing component: #SubscriptionManager#
INFO  - jetty                           - Logging to org.apache.servicemix.http.jetty.JCLLogger@2b735f84 via
jetty.JCLLogger
INFO  - DeploymentService               - Restoring service assemblies
INFO  - ComponentMBeanImpl              - Setting running state for Component: servicemix-bean to Started
INFO  - ComponentMBeanImpl              - Initializing component: servicemix-bean
INFO  - ComponentMBeanImpl              - Setting running state for Component: servicemix-camel to Started
INFO  - ComponentMBeanImpl              - Initializing component: servicemix-camel
INFO  - ComponentMBeanImpl              - Setting running state for Component: servicemix-cxf-bc to Started
INFO  - ComponentMBeanImpl              - Initializing component: servicemix-cxf-bc
1 name = cxf.xml
1 name = cxf.xml
1 name = cxf.xml
Mar 23, 2011 1:58:16 PM org.apache.cxf.bus.spring.BusApplicationContext getConfigResources
INFO: No cxf.xml configuration file detected, relying on defaults.
INFO  - ComponentMBeanImpl              - Setting running state for Component: servicemix-cxf-se to Started
INFO  - ComponentMBeanImpl              - Initializing component: servicemix-cxf-se
1 name = cxf.xml
1 name = cxf.xml
1 name = cxf.xml
Mar 23, 2011 1:58:17 PM org.apache.cxf.bus.spring.BusApplicationContext getConfigResources
INFO: No cxf.xml configuration file detected, relying on defaults.
INFO  - ComponentMBeanImpl              - Setting running state for Component: servicemix-drools to Started
INFO  - ComponentMBeanImpl              - Initializing component: servicemix-drools
INFO  - ComponentMBeanImpl              - Setting running state for Component: servicemix-eip to Started
```

SnapShot(III)

- The contents of hotdeploy folder , having the deployed handson.-sa zip.



Name	Date modified	Type	Size
handson-sa-1.0-SNAPSHOT	3/23/2011 2:22 PM	ZIP File	4,211 KB
README	10/24/2008 10:22 ...	Text Document	1 KB
servicemix-bean-3.2.3-installer	10/24/2008 10:22 ...	ZIP File	1,397 KB
servicemix-camel-3.2.3-installer	10/24/2008 10:22 ...	ZIP File	1,707 KB
servicemix-cxf-bc-3.2.3-installer	10/24/2008 10:22 ...	ZIP File	10,071 KB
servicemix-cxf-se-3.2.3-installer	10/24/2008 10:22 ...	ZIP File	8,273 KB
servicemix-drools-3.2.3-installer	10/24/2008 10:22 ...	ZIP File	8,157 KB
servicemix-eip-3.2.3-installer	10/24/2008 10:22 ...	ZIP File	1,298 KB
servicemix-file-3.2.3-installer	10/24/2008 10:22 ...	ZIP File	1,607 KB
servicemix-ftp-3.2.3-installer	10/24/2008 10:22 ...	ZIP File	1,826 KB
servicemix-http-3.2.3-installer	10/24/2008 10:22 ...	ZIP File	2,637 KB
servicemix-jms-3.2.3-installer	10/24/2008 10:22 ...	ZIP File	2,386 KB
servicemix-jsr181-3.2.3-installer	10/24/2008 10:22 ...	ZIP File	4,807 KB
servicemix-lwcontainer-3.2.3-installer	10/24/2008 10:22 ...	ZIP File	1,237 KB
servicemix-osworkflow-3.2.3-installer	10/24/2008 10:22 ...	ZIP File	2,431 KB
servicemix-quartz-3.2.3-installer	10/24/2008 10:22 ...	ZIP File	491 KB
servicemix-saxon-3.2.3-installer	10/24/2008 10:22 ...	ZIP File	2,973 KB
servicemix-script-3.2.3-installer	10/24/2008 10:22 ...	ZIP File	2,895 KB
servicemix-shared-3.2.3-installer	10/24/2008 10:22 ...	ZIP File	5,084 KB
servicemix-truezip-3.2.3-installer	10/24/2008 10:22 ...	ZIP File	1,997 KB
servicemix-wsn2005-3.2.3-installer	10/24/2008 10:22 ...	ZIP File	2,223 KB
servicemix-xmpp-3.2.3-installer	10/24/2008 10:22 ...	ZIP File	2,039 KB

SnapShot(IV)

- Once the request is sent check for status as 202 which means message has been sent successfully

ServiceMix WSDL-First Example

Welcome to the WSDL-First example for ServiceMix

Perform a POST into the HTTP binding. This requires JavaScript.

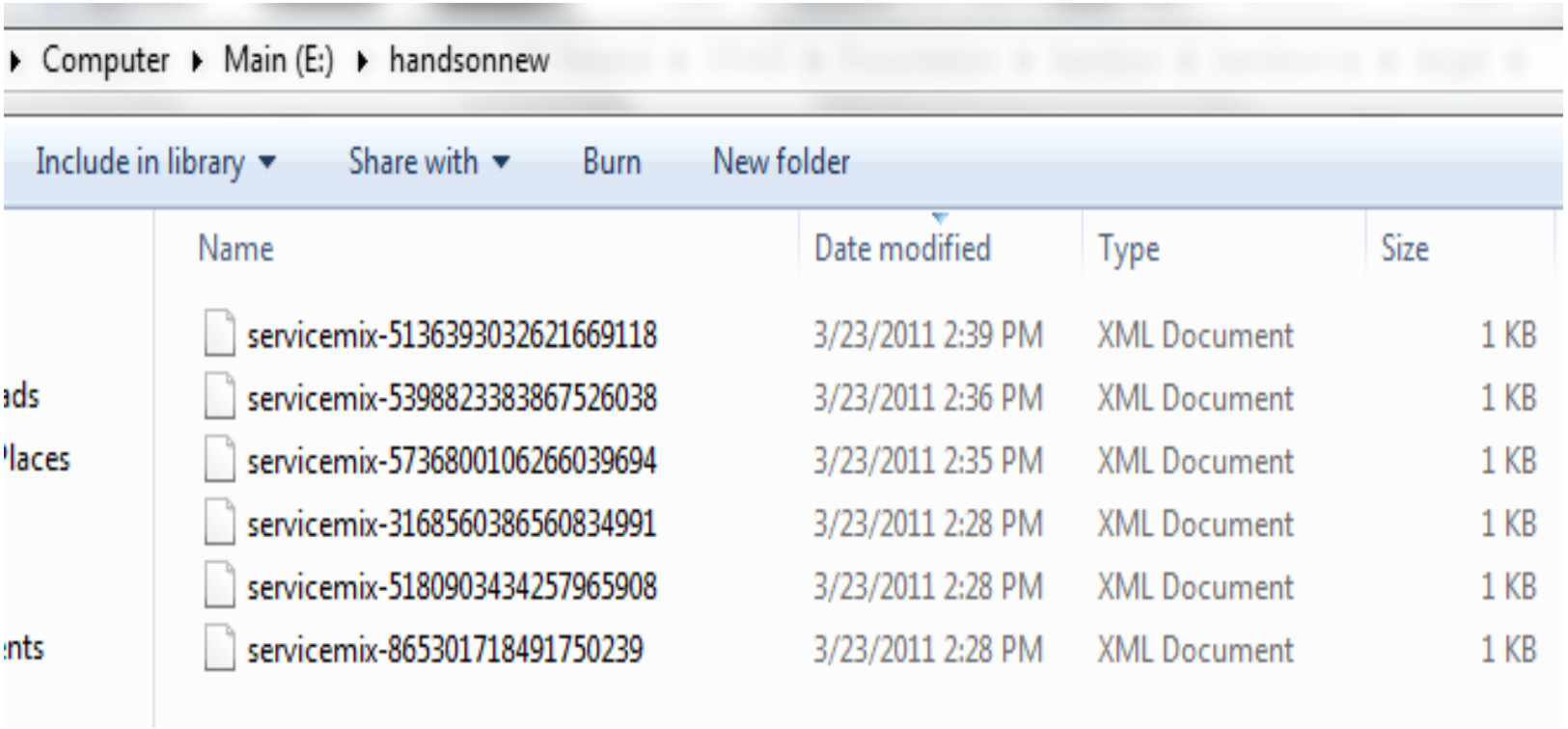
Target: http://localhost:8192/example/

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
              xmlns:tns="http://servicemix.apache.org/samples/wsdl-
first/types">
  <env:Body>
    <tns:GetPerson>
      <tns:personId>world</tns:personId>
    </tns:GetPerson>
  </env:Body>
</env:Envelope>
```

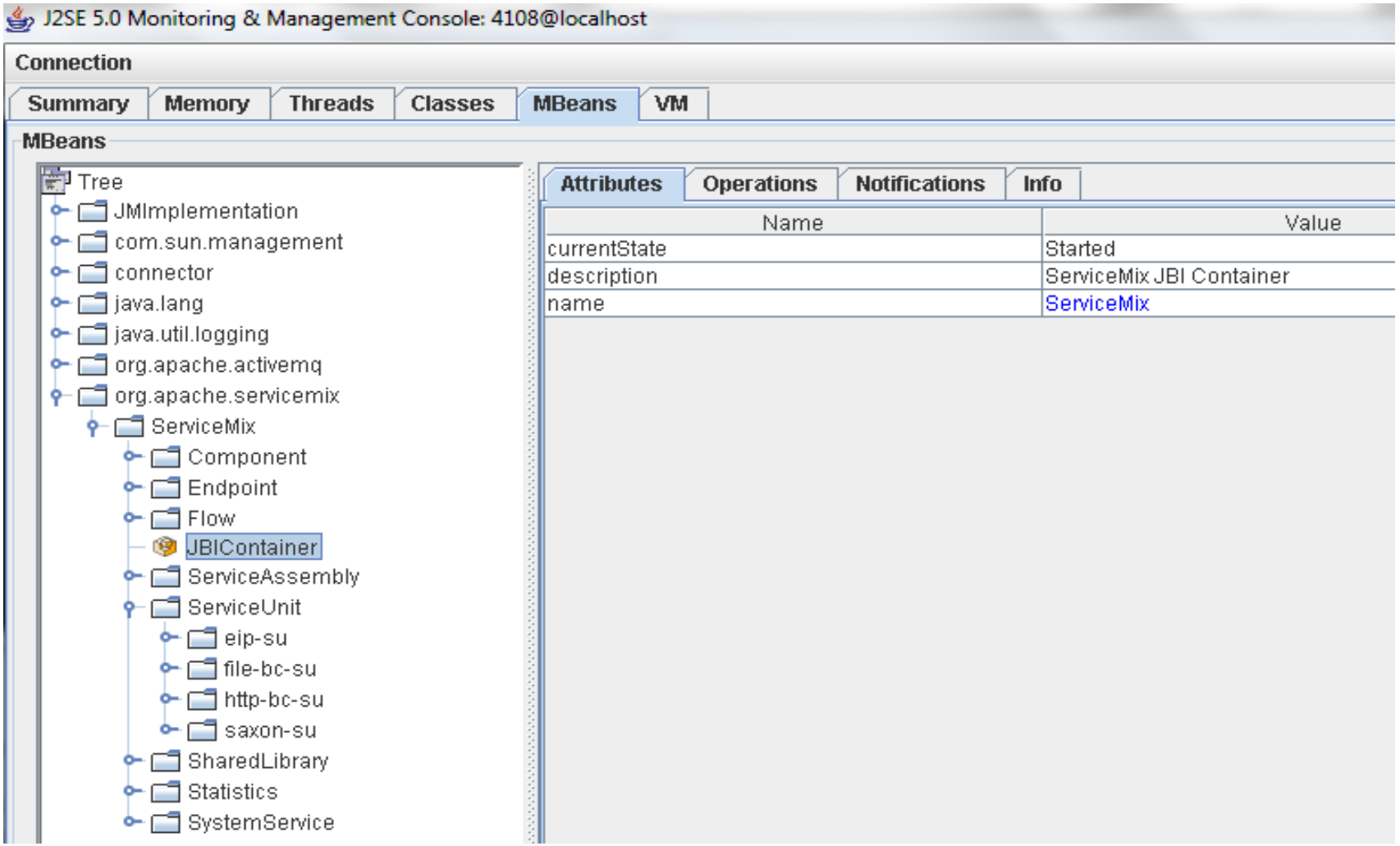
```
STATUS: 202
```

SnapShot(V)

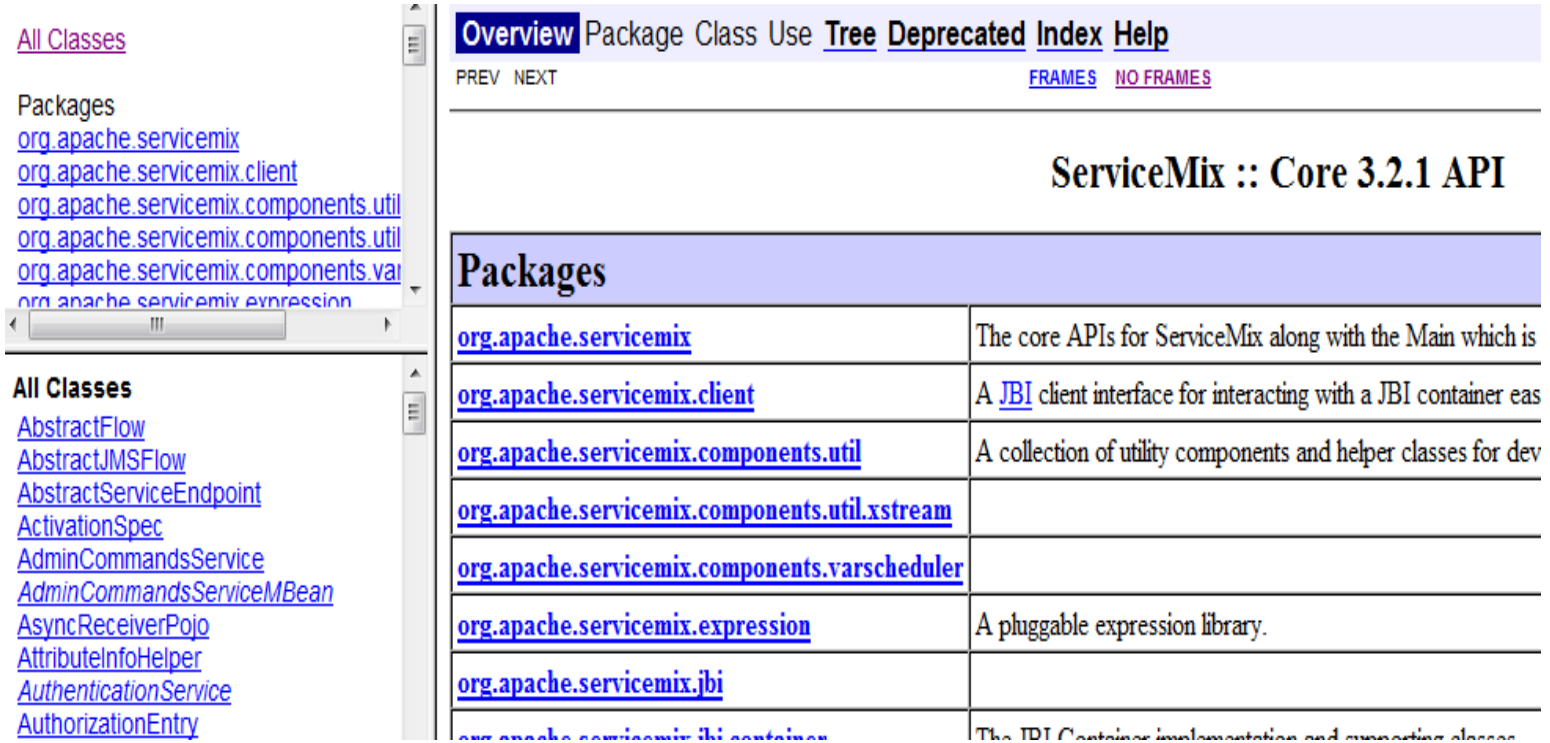
- The final transformed xml's present in the target directory



SnapShot(VI): Managed Through JConsole



ServiceMix API's



- http://servicemix.apache.org/dist/servicemix-3.2.1/site/parent/core/servicemix-core/apidocs/

Benefits (I)

- Nice Abstraction
- Components are reusable
- Loose coupling between the components
- Based on service contracts

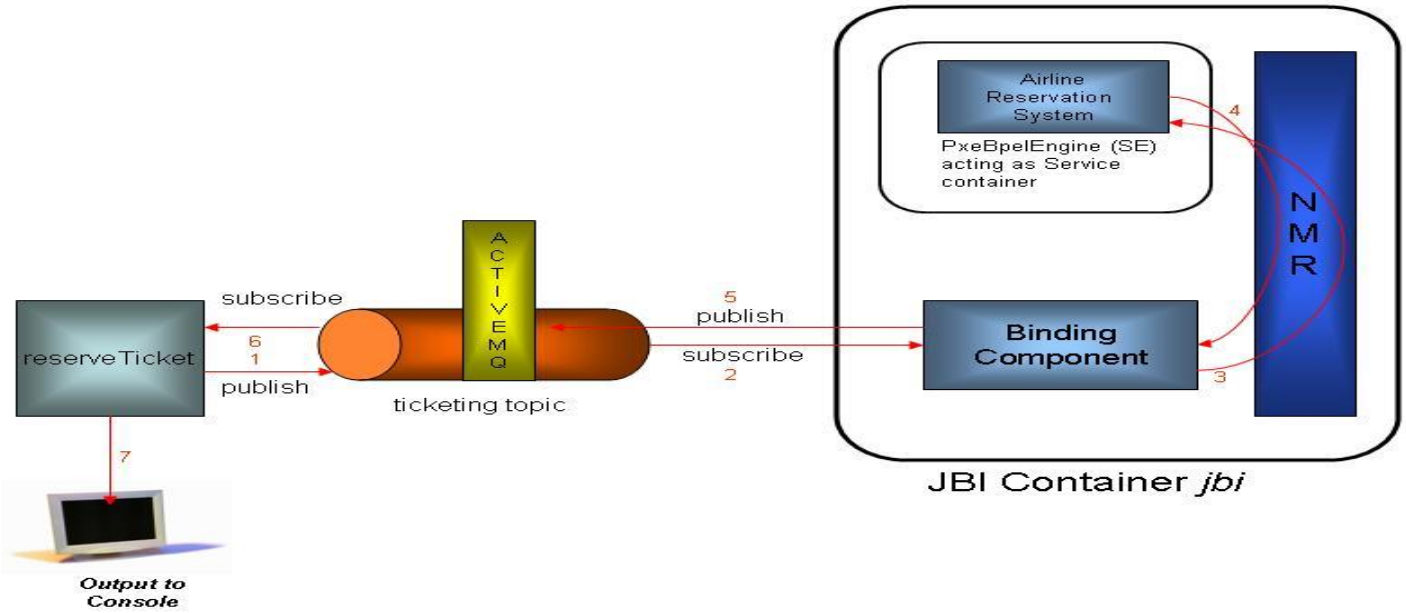
Benefits (II)

- Adapts to changing user requirements
- Scales to wide spread enterprise development
- Supports High Availability and Clustering
- As the configuration involves using xml's ,helps in easy management of modules

Applications(I)

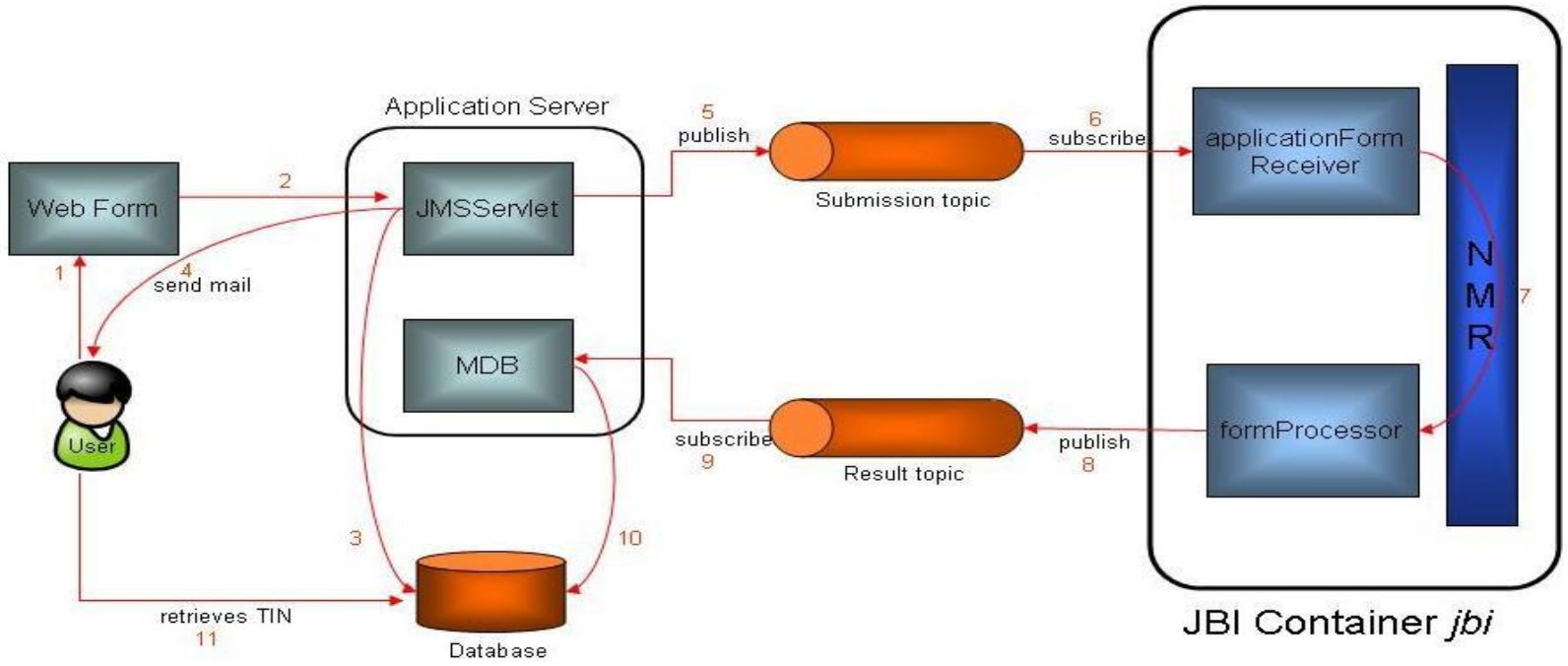
- Used in Mediation Development (Telecom Industry).
- Service mix to solve the integration of different network elements with the network management software(NMS).
- This way the customer is happy, as he can manage using a single network management software.
- Service mix architecture is scalable and hence a number of network elements can be integrated on to single NMS.

Business case (I): Online Ticket Reservation System



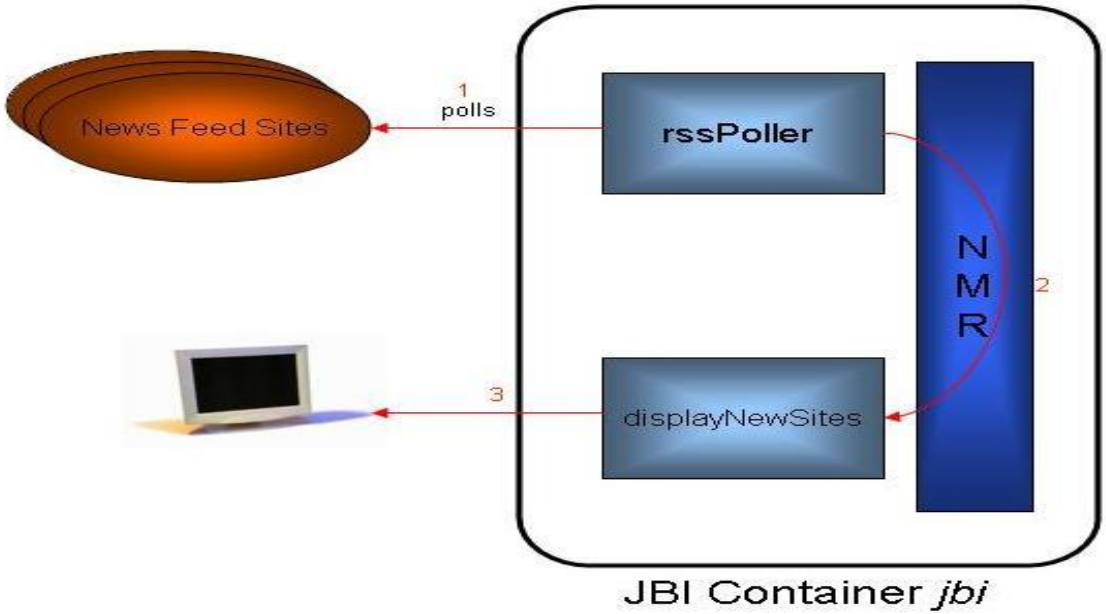
Source: http://servicemix.apache.org/use-cases-whole.html

Business case(II): Online Application for Tax ID No. System



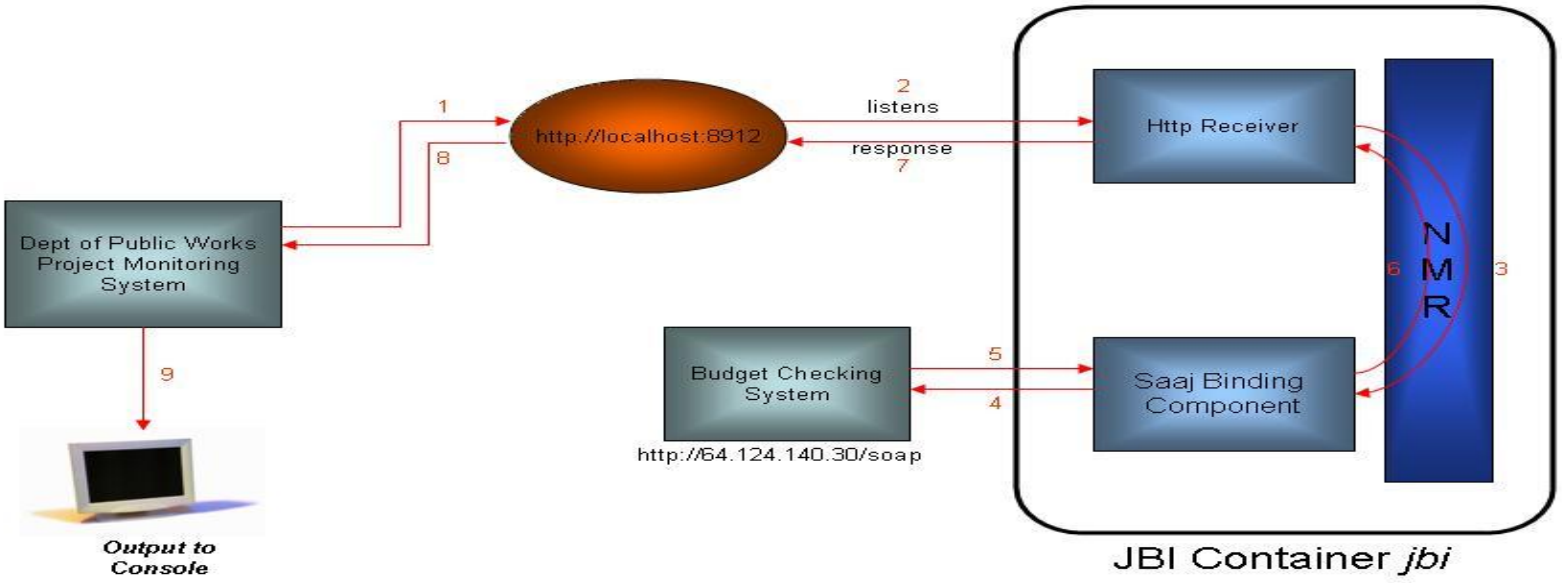
- Source: http://servicemix.apache.org/use-cases-whole.html

Business case(III): News Feed Monitoring System



- Source: http://servicemix.apache.org/use-cases-whole.html

Business case(IV): Department of Public Works Project Monitoring System



- Source: http://servicemix.apache.org/use-cases-whole.html

Different vendors of ESB apart from Apache Service mix

Summary

- Enterprise Service Bus (ESB)
- Java Business Integration (JBI)
- JBI Messaging Architecture
- Apache ServiceMix
- Sample use case

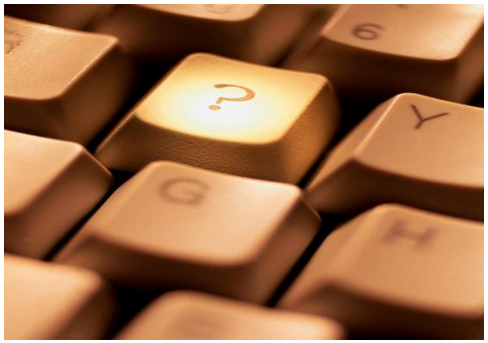
References

- Open Source ESB
  - http://servicemix.apache.org/home.html
- Apache Camel
  - http://camel.apache.org/architecture.html
- Service mix components
  - http://servicemix.apache.org/users-guide.html
- Java Business Integration
  - JBI Components: Part 1 (Theory), *Ron Ten-Hove, Sun Microsystems,2006*

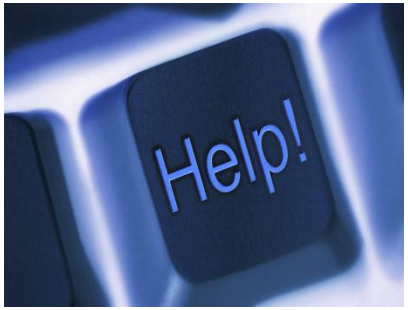
Credits

- The problem with Business Integration were adapted from Service Oriented Architecture with JBI and ServiceMix by Bruce Snyder
- Service-Oriented Architecture:What Next?, *David Chappell Chappell & Associates, www.davidchappell.com* Based on Notes by Brand Niemann, April 8, 2004

Questions



Support



srg06264@colorado.edu

Thank You