Content Based Routing

By: Kurt Stam (kurt.stam@jboss.com)

JBoss ESB JBoss Labs Home Page: http://labs.jboss.com/portal/jbossesb

JBoss ESB Developer Community Forums:
http://www.jboss.com/index.html?module=bb&op=viewforum&f=220

1. Content Based Message Routing

The Content Based Router (CBR) in the JBossESB can be used to route message to the next destination based on the content of the message. By default the ESB will use JBossRules as it's evaluation engine, however this is left configurable. In the jbossesb-properties.xml there is property called 'org.jboss.soa.esb.routing.cbrClass' in the 'messagerouting' section:

```
<properties name="messagerouting">
  <property name="org.jboss.soa.esb.routing.cbrClass"
            value="org.jboss.internal.soa.esb.services.routing.cbr.JBossRulesRouter"/>
</properties>
```

which can be used to plug in another evaluation engine.

2.0 JBossRules Based Router

By default the Content Based router uses JBossRules. Read on to find how to bring up CBR, how to give it a specific rule set, and how to send a message to the CBR.

2.1 Create a Content-Based Router

To bring up a Content-based router you need to bring up the CBR-service. Currently this is done by bringing up the CbrListener. To do this you need to add an xml-chapter to your deployment-config.xml that looks like the following example which uses JMS as transport protocol:

```
<?xml version = "1.0" encoding = "UTF-8"?>
<jbossesb
xmlns="http://anonsvn.labs.jboss.com/labs/jbossesb/trunk/product/etc/schemas/xml/jb
ossesb-1.0.xsd"
parameterReloadSecs="10">

      <providers>
          <jms-provider name="localhost"
                        connection-factory="ConnectionFactory"
                        jndi-context-
factory="org.jnp.interfaces.NamingContextFactory"
                        jndi-URL="localhost" >

              <jms-bus busid="QueueA">
                  <jms-message-filter
                      dest-type="QUEUE"
                      dest-name="queue/A"
                      selector="service='CBRouting-SerializableOrXml'"
                   />
              </jms-bus>
          </jms-provider>
      </providers>
      <services>
        <service
            category="MessageRouting"
            name="ContentBasedRoutingService"
            description="CBR Listener"
            is-gateway="false"
            service-class="org.jboss.soa.esb.listeners.message.CbrListener">
            <listeners>
                  <jms-listener name="XPathContentBasedRouter"
                              busidref="QueueA"
                              maxThreads="1">
                       <property name="ruleSet" value="JBossESBRules.drl"/>
                   </jms-listener>
             </listeners>
        </service>
   </services>
</jbossesb>
```

The parameters are identical to the JMS Listener:

1. category: The category of this service, which will be used at registration time.
2. name: The name of this service, which will be at registration time. Note that both the service-category and service-name are needed to get a handle to this service.

3. description: An optional element to describe what this service does. This description is supposed to be human readable.
4. service-class: The class name of the actual listener.
5. Is-gateway: false

Cbr specific properties to the (jms-)listener:

1. ruleSet: The set of rules that should be used to evaluate the content. Only 1 ruleSet can be given for each CBR. Just define more router if you need composite CBR routing.
2. ruleLanguage: In JBossRules you can define a custom (business) language. This (optional) field can be used to pass in that custom language to go with the defined ruleSet.

2.2 Create a ruleSet

A rule set can be created using the JBossIDE which includes a plug-in for JbossRules. Figure 1 shows a screen shot the plug-in.

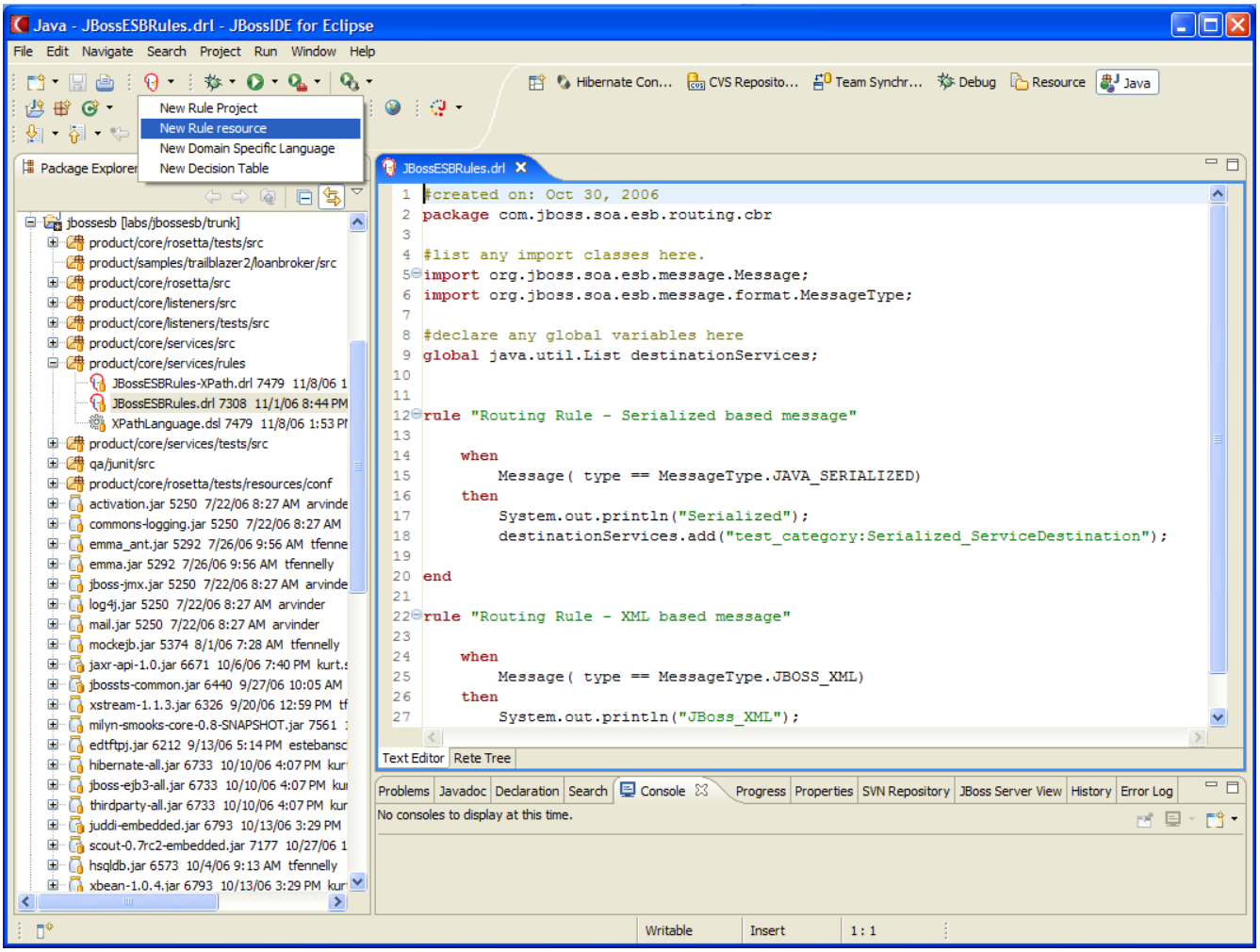

Figure 1. JBoss Rules IDE. Create a new ruleSet.

To add a new rule simple click on “New Rule resource”. Make sure your rules import at least

```
import org.jboss.soa.esb.message.Message;
```

The message will be asserted into the working memory of the rules engine. You can use sub objects of message in your evaluation. This is demonstrated in the JBossESBRules.drl ruleSet where the rule checks if the type is XML or Serializable. Secondly you have to make sure to define:

```
global java.util.List destinationServices;
```

The destinationServices List is returned and should contain one or more destination services. The format of the destination service should be:
<service-category>:<service-name>

If you save your rule into the product/core/services/rules package it will be jarred up into the jbossesb-rules.jar. However as long as you your rules on the classpath somewhere JBossRules should find it.

2.3 Use the XPath Custom Rule Language

For XML-based messages it is convenient to do XPath based evaluation. For this we ship a custom rule language defined in the XpathLanuage.dsl. To use it you need to reference it in your ruleSet with:

```
expander XPathLanguage.dsl
```

and you need to pass it in the name of this file in the attributes like:

```
ruleSet="JBossESBRules-XPath.drl"
ruleLanguage="XPathLanguage.drl"/>


Currently the XPath Language makes sure the message is of the type JBOSS_XML and it defines
```

1. xpathMatch "<element>": yields true if is an element by this name is matched.
2. xpathEquals "<element>", "<value>"): yields true if the element is found and it's value equals the value.

You can define your own Custom Rule Language.

2.4 Use the Content Based Router

Now that we have a CBR up and running and listening we can send it messages. To to this you need to add it in as an action somewhere in your action-pipeline. For example you can add this following fragment:

```
<action name="TestDefaultRouteAction"
    class="org.jboss.soa.esb.actions.CbrProxyAction"
    service-category="MessageRouting"
    service-name="ContentBasedRoutingService"
    process="route" />
```

Where

1. class: the name of the action class.
2. service-category: the service category of the service that needs to be called (the CBR)
3. service-name: the name of the category of the service that needs to be called (the CBR)

4. process: the name of the method on the CBR-action, this can be any of
    1. routeAndDeliver: route and deliver the message to the destination(s).
    2. route: route the message and return a message with a list of destinations attached to it.
    3. deliver: just deliver. A message that was routed before will be delivered to its destination(s)

The separation of route and deliver allows for other actions (like message transformation) to be inserted between routing and delivering.