JMS Event Service

Proof Of Concept

- This is a functional proof of concept, please feel free to try it out and also post comments, suggestions, etc.

Information

The JMS event service is meant to be a replacement for the current Event Service which uses the database to send messages between nodes in the cluster. It allows for a standard message sending methodology and does not touch the database. Events are also transactional with JMS.

JMS Information

- Java Message Service Tutorial
- ActiveMQ

JMS Event Service

- Goals
  - Standard message types (JMS Message)
  - Better cluster wide events with lower database load and more flexibility
  - More flexible message data (JMS supports all kinds of data being attached to messages)
  - Easy interception of events (via standard JMS listeners)
  - Easy handling of custom event queues
  - Encourage wider usage of events in Sakai
- Allows JMS and event handling in Sakai
  - Lei (a student worker here at CARET) and I have created a general JMS service for Sakai and an implementation of the Sakai event tracking
    service to go along with it
  - Our current JMS implementation uses ActiveMQ and auto discovery to send messages to other Sakai servers running in the cluster
- Uses some sakai.properties for config (but works with defaults)
  - Requires users to disable the legacy EventTrackingService
  - Also requires allowing the JMS api into shared (already there in trunk)
- Benefits in a little more detail
  - Reduce the load that the legacy cluster event service puts on the DB
  - Able to use JMS messages
  - Easy to define standard listeners
    - Allows you to add a listener or listeners that can log to a db on a different machine or wherever else you want without impacting the production database instance
  - ability to set up your own message destinations
  - ability to send just about anything in your message (JMS allows full objects and all kinds of crazy stuff)
  - transactional message processing
  - ability to control where the messages go (local only, cluster, etc.)
- Code https://source.sakaiproject.org/contrib/messageservice/trunk
- JIRA SAK issue for tracking this: http://jira.sakaiproject.org/jira/browse/SAK-11021

Deploying and Running The JMS Event Service in sakai_2-5-x

Getting the source and building it

- ```
  svn co https://source.sakaiproject.org/svn/sakai/branches/sakai_2-5-x
  cd sakai_2-5-x
  https://source.sakaiproject.org/contrib/messageservice/trunk messageservice
  mvn clean install sakai:deploy
  ```

Sakai Properties

Property	Default Value	Comments
jms.useJMX	false	If you have tomcat configured to allow JMX access, I would recommend to enable this setting as it greatly helps during the development process
  ```

jms.brokerURL	tcp://localhost:61616	-
jms.brokerDataPath	sakai-home location	This is either in "tomcat/sakai" or the location defined by JAVA_OPTS "-Dsakai.home=/path/to/sakai-home/"

Troubleshooting

- If tomcat doesn't start completely and the last message in catalina.out is the following:
  - ```
    WARN: brokerName not set (2007-12-11 20:22:26,149 main_org.apache.activemq.transport.discovery.multicast.MulticastDiscoveryAgent)
    ```

  ... then, you probably have some port/transport security service in place such at iptables. The issue is that activeMQ uses UDP on port 6155 for its multicast discovery. This webpage has some information regarding this topic.
- So in my case, I had to add the following line to my iptables
  - ```
    -A RH-Firewall-1-INPUT -m state --state NEW -m udp -p udp -s 192.168.1.0/24 --dport 6155 -j ACCEPT
    ```