Sakai MessageService

Tiny Link

Info

The Sakai MessageService integrates JMS with Sakai and exposes the JMS API as a service. ActiveMQ is used as the JMS implementation.
Moire info about this related to events here: JMS Event Service

News

- 2008-02-13 : Switch to full spring configuration.
  - org.sakaiproject.messageservice.JmsConnectionFactory accessible via Spring declaration.
  - Retained org.sakaiproject.messageservice.api.MessageService for direct JMS implementation.
  - Changed ActiveMQ to version 5.1 to fix 'hanging' broker and client during shutdown problem.
- 2008-02-04 : Applied some Bugfixes to the messageservice based cluster service. This should be ready for initial testing now
- 2008-01-29 : Extended MessageService API to allow to create a connection with username and password
- 2008-01-26 :
  - Extended MessageService API by adding "public Connection getConnection()"
  - Created MessageService sakai_2-4-x branch, which uses ActiveMQ v4.1.1
- 2008-01-23 :
  - Finished first round of converting Sakai Cluster Service to use the Message Service. Performed initial testing in a two node setup.
  - Requested Sakai Foundation JIRA Branch for Sakai Cluster Service using Message Service: http://bugs.sakaiproject.org/jira/browse/SAK-12797

Source Code

- https://source.sakaiproject.org/contrib/messageservice

Documentation JMS v1.1

- Make sure that you look at the **v1.1** and NOT v1.0.2 of the JMS API
  - J2EE 1.4 Tutorial
    - Chapter 33 : JMS Tutorial
  - JavaTM 2 Platform Enterprise Edition, v 1.4 API Specification
    - JMS v1.1 javax.jms

Usage

The MessageService API is very simple and exposes the following methods:

```
/*
 * Creates a JMS connection with the default user identity. The connection is
 * created in stopped mode. No messages will be delivered until the
 * Connection.start method is explicitly called.
 *
 * @return a newly created JMS connection. May return null
 *
 */
 public Connection createConnection();

/*
 * Creates a JMS connection with the specified user identity. The connection is
 * created in stopped mode. No messages will be delivered until the
 * Connection.start method is explicitly called.
 *
 * @parameter userName - the caller's user name
 * @parameter password - the caller's password
 *
 * @return a newly created JMS connection. May return null
 */
 public Connection createConnection(String userName, String password);

/*
 * Gets the JMS Connection with the specified user identity, configured in components.xml
 * The connection is returned in start mode.
 *
 * @return the initially created JMS Connection object. May return null
 *
 */
 public Connection getConnection();
```

Usage with Spring (Sakai 2.5)

```
<!-- definition of a message producer -->
<bean id="org.sakaiproject.messageservice.test.MessageProducer"
  class="org.sakaiproject.messageservice.test.MessageProducerImpl">
    <property name="connectionFactory"
      ref="org.sakaiproject.messageservice.JmsConnectionFactory" />
    <property name="queueName" value="test.destination" />
</bean>

<!-- definition of listner container -->
<bean id="listenerContainer"
  class="org.springframework.jms.listener.DefaultMessageListenerContainer"
  depends-on="org.sakaiproject.messageservice.JmsConnectionFactory">
    <property name="concurrentConsumers" value="1" />
    <property name="maxConcurrentConsumers" value="1" />
    <property name="connectionFactory"
      ref="org.sakaiproject.messageservice.JmsConnectionFactory" />
    <property name="destinationName" value="test.destination" />
    <!-- set pubSubDomain to false to create a queue instead of a Publish/Subscribe topic -->
    <property name="pubSubDomain" value="true"/>
    <property name="messageListener">
      <bean class="org.sakaiproject.messageservice.test.MessageListenerImpl"/>
    </property>
</bean>
```

MessageService (JMS) Best Practices

- JMS Connection Object
    - Usually there is no need to create more than one JMS Connection object, and the ActiveMQ documentation and AQM Forum posts recommend to create/use only one connection object per server. Thus, in Sakai, we should use only one connection by calling getConnection(). In rare cases, it's justified to create a second connection.
- JMS Session
    - I asked about JMS session usage in the ActiveMQ forum, and here is the thread:
        - http://www.nabble.com/Using-one-session-per-message-to15114117s2354.html

Configuring

- changing the persistence adapter
  The persistence adapter can be changed in sakai.properties or local.properties.
  property name:
  targetBeanName@org.sakaiproject.messageservice.activemq.PersistenceAdapter
  default value:
  org.sakaiproject.messageservice.activemq.AMQPersistenceAdapter
  possible persistence adapters:
  org.sakaiproject.messageservice.activemq.AMQPersistenceAdapter
  org.sakaiproject.messageservice.activemq.JournalPersistenceAdapter
  org.sakaiproject.messageservice.activemq.JdbcPersistenceAdapter

- enable clustered broker
  By default the broker is configured to run in standalone mode. To enable a clustered environment and auto discovery the following change have to be applied to activemq-beans.xml

```
<bean id="org.sakaiproject.messageservice.JmsConnectionFactory"
  class="org.apache.activemq.ActiveMQConnectionFactory">
  <property name="brokerURL" value="discovery:(multicast://sakai)"/>
</bean>

<bean id="org.sakaiproject.messageservice.activemq.TransportConnector"
  class="org.apache.activemq.broker.TransportConnector"
  lazy-init="true">
  <property name="name" value="default" />
  <property name="uri" ref="org.sakaiproject.messageservice.api.BrokerUrl" />
  <property name="discoveryUri" ref="org.sakaiproject.messageservice.api.DiscoveryUrl"/>
</bean>

<bean id="org.sakaiproject.messageservice.activemq.NetworkConnectorListFactory"
  class="org.springframework.beans.factory.config.BeanReferenceFactoryBean"
  lazy-init="true">
  <property name="targetBeanName"
    value="org.sakaiproject.messageservice.activemq.DiscoveryNetworkConnectorList" />
</bean>
```

- changing other Broker properties

Property	Default	Comments
persistent@org.sakaiproject.messageservice.broker.BrokerService	true	
useJmx@org.sakaiproject.messageservice.broker.BrokerService	true	
targetBeanName@org.sakaiproject.messageservice.api.BrokerUrlStringFactory	org.sakaiproject.messageservice.api.BrokerUrlString (tcp://localhost:61616?wireFormat.maxInactivityDuration=0)	other valid bean name, that defines a url string

- On each application node, you need to enable the following ports and protocols in your host based firewall
  - TCP port 61616
  - UDP port 6155
  - e.g. Linux iptables

```
-A RH-Firewall-1-INPUT -m state --state NEW -m tcp -p tcp -s 192.168.1.0/24 --dport 61616 -j ACCEPT
-A RH-Firewall-1-INPUT -m state --state NEW -m udp -p udp -s 192.168.1.0/24 --dport 6155 -j ACCEPT
```

Building The MessageService sakai_2-4-x Branch (Maven 1)

- svn co https://source.sakaiproject.org/contrib/messageservice/branches/sakai_2-4-x messageservice-2-4-x
- cd messageservice-sakai_2-4-x
- maven sakai

Using MessageService in existing Sakai services and tools

- event
- cluster
- chat