Using the Java Messaging System

Table of Contents

Overview

Celtix provides a transport plug-in that enables endpoints to use Java Messaging System (JMS) queues and topics. Celtix's JMS transport plug-in uses the Java Naming and Directory Interface (JNDI) to locate and obtain references to the JMS provider that brokers for the JMS destinations. Once Celtix has established a connection to a JMS provider, Celtix supports the passing of messages packaged as either a JMS `ObjectMessage` or a JMS `TextMessage`.

Message Formatting

The JMS transport takes the payload formatting and packages it into either a JMS `ObjectMessage` or a `TextMessage`. When a message is packaged as an `ObjectMessage` the message information, including any format-specific information, is serialized into a `byte[]` and placed into the JMS message body. When a message is packaged as a `TextMessage`, the message information, including any format-specific information, is converted into a string and placed into the JMS message body.

When a message sent by Celtix is received by a JMS application, the JMS application is responsible for understanding how to interpret the message and the formatting information. For example, if the Celtix contract specifies that the binding used for a JMS endpoint is SOAP, and the messages are packaged as `TextMessage`, the receiving JMS application will get a text message containing all of the SOAP envelope information.

Namespace

The WSDL extensions for defining a JMS endpoint are defined in the namespace `http://celtix.objectweb.org/transports/jms`. In order to use the JMS extensions you will need to add the line shown in Text 1 to the `definitions` element of your contract.

```
xmlns:jms="http://celtix.objectweb.org/transports/jms"
```

Text 1: JMS Extension Namespace

Basic Endpoint Configuration

JMS endpoints need to know certian basic information about how to establish a connection to the proper

destination. This information is provided using the `jms.address` element and its child the `jms:JMSNamingProperties` element. The `jms:address` element's attributes specify the information needed to identify the JMS broker and the destination. The `jms:JMSNamingProperties` element specifies the Java properties used to connect to the JNDI service.

The address Element

The basic configuration for a JMS endpoint is done by using a `jms:address` element in your service's `port` element. The `jms:address` element uses the attributes described in Table 1 to configure the connection to the JMS broker.

Attribute	Description
`destinationStyle`	Specifies if the JMS destination is a JMS queue or a JMS topic.
`jndiConnectionFactoryName`	Specifies the JNDI name bound to the JMS connection factory to use when connecting to the JMS destination.
`jndiDestinationName`	Specifies the JNDI name bound to the JMS destination to which requests are sent.
`jndiReplyDestinationName`	Specifies the JNDI name bound to the JMS destinations where replies are sent. This attribute allows you to use a user defined destination for replies. For more details see Using a Named Reply Destination.
`connectionUserName`	Specifies the username to use when connecting to a JMS broker.
`connectionPassword`	Specifies the password to use when connecting to a JMS broker.

Table 1: JMS Endpoint Attributes

The JMSNamingProperties Element

To increase interoperability with JMS and JNDI providers, the `jms:address` element has a child element, `jms:JMSNamingProperties`, that allows you to specify the values used to populate the properties used when connecting to the JNDI provider. The `jms:JMSNamingProperties` element has two attributes: `name` and `value`. The `name` attribute specifies the name of the property to set. The `value` attribute specifies the value for the specified property.

The following is a list of common JNDI properties that can be set:

- `java.naming.factory.initial`
- `java.naming.provider.url`
- `java.naming.factory.object`
- `java.naming.factory.state`
- `java.naming.factory.url.pkgs`
- `java.naming.dns.url`
- `java.naming.authoritative`

- java.naming.batchsize
- java.naming.referral
- java.naming.security.protocol
- java.naming.security.authentication
- java.naming.security.principal
- java.naming.security.credentials
- java.naming.language
- java.naming.applet

For more details on what information to use in these attributes, check your JNDI provider's documentation and consult the Java API reference material.

Using a Named Reply Destination

By default Celtix endpoints using JMS create a temporary queue for sending replies back and forth. You can change this behavior by setting the `jndiReplyDestinationName` attribute in the endpoint's contract. A Celtix client endpoint will listen for replies on the specified destination and it will specify the value of the attribute in the `ReplyTo` field of all outgoing requests. A Celtix service endpoint will use the value of the `jndiReplyDestinationName` attribute as the location for placing replies if there is no destination specified in the request's `ReplyTo` field.

Example

Text 2 shows an example of an Celtix JMS port specification.

```
<service name="JMSService">
 <port binding="tns:Greeter_SOAPBinding" name="SoapPort">
  <jms:address jndiConnectionFactoryName="ConnectionFactory"
               jndiDestinationName="dynamicQueues/test.Celtix.jmstransport">
   <jms:JMSNamingProperty name="java.naming.factory.initial"
                          value="org.activemq.jndi.ActiveMQInitialContextFactory" />
   <jms:JMSNamingProperty name="java.naming.provider.url"
                          value="tcp://localhost:61616" />
  </jms:address>
 </port>
</service>
```

Text 2: Celtix JMS Port

Consumer Endpoint Configuration

JMS consumer endpoints have two behaviors that are configurable in the contract:

- the type of message being sent.
- the amount of time the consumer will wait for a response.

The consumer endpoint's behaviors are configured using the optional `jms:client` element. The `jms:client` element is a child of the WSDL `port` element and has two attributes:

`messageType`	Specifies how the message data will be packaged as a JMS message. `text` specifies that the data will be packaged as a `TextMessage`. `binary` specifies that the data will be packaged as an `ObjectMessage`.
`timeout`	Specifies the amount of time, in milliseconds, that the endpoint will wait for a reply before timing out.

Service Endpoint Configuration

JMS service endpoints have a number of behaviors that are configurable in the contract. These include if the service uses durable subscriptions, if the service uses local JMS transactions, and the message selectors used by the endpoint.

Service endpoint behaviors are configured using the optional `jms:server` element. The `jms:server` element is a child of the WSDL `port` element and has the following attributes:

`useMessageIDAsCorrealationID`	Specifies whether JMS will use the message ID to correlate messages. The default is `false`.
`durableSubscriberName`	Specifies the name used to register a durable subscription. See Setting up Durable Subscriptions.
`messageSelector`	Specifies the string value of a message selector to use. See Using Message Selectors.
`transactional`	Specifies whether the local JMS broker will create transactions around message processing. The default is false. See Using Reliable Messaging.

Setting up Durable Subscriptions

If you want to configure your Celtix service to use durable subscriptions, you can set the optional `durableSubscriberName` attribute. The value of the attribute is the name used to register the durable subscription.

Using Message Selectors

If you want to configure your Celtix service to use a JMS message selector, you can set the optional `messageSelector` attribute. The value of the attribute is the string value of the selector. For more information on the syntax used to specify message selectors, see the JMS 1.1 specification.

Using Reliable Messaging

If you want your Celtix service to use the local JMS broker's transaction capabilities, you can set the `transactional` attribute to `true`.

When the `transactional` attribute is set, a Celtix service endpoint's JMS transport layer will begin a transaction when it pulls a request from the queue. The endpoint will then process the request and send the response back to the JMS transport layer. Once the JMS transport layer has successfully placed the response on the response queue, the transport layer will commit the transaction. So, if the Celtix service endpoint crashes while processing a request or the transport layer is unable to send the response, the JMS broker will hold the request in the queue until it is successfully processed.

In cases where Celtix is acting as a router between JMS and another transport, setting the `transactional`

attribute will ensure that the message is delivered to the second server. The JMS portion of the router will not commit the message until the message has been successfully consumed by the outbound transport layer. If an exception is thrown during the consumption of the message, the JMS transport will rollback the message, pull it from the queue again, and attempt to resend it.