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`.

Basic Endpoint Configuration

JMS endpoints need to know certain 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.

WSDL 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

The address Element

The basic configuration for a JMS endpoint is done by using a `jms:address` element as the child of 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. `jms:JMSNamingProperties` element can also be used for specification of provider specific properties.

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 specify the type of messages they use. JMS consumer endpoint can use either a JMS `ObjectMessage` or a JMS `TextMessage`. When using an `ObjectMessage` the consumer endpoint uses a `byte[]` as the method for storing data into and retrieving data from the JMS message body. When messages are sent, the message data, including any formating information, is packaged into a `byte[]` and placed into the JMS message body before it is placed on the wire. When messages are received, the consumer endpoint will attempt to unmarshall the data stored in the JMS body as if it were packed in a `byte[]`.

When using a `TextMessage`, the consumer endpoint uses a string as the method for storing and retrieving data from the JMS message body. When messages are sent, the message information, including any format-specific information, is converted into a string and placed into the JMS message body. When messages are received the consumer endpoint will attempt to unmashall the data stored in the JMS message body as if it were packed into a string.

When a native JMS applications interact with Celtix consumers, the JMS application is responsible for interpreting 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.

Consumer endpoint can be configured in one of two ways:

- Celtix configuration
- WSDL file

The recommended method is to place the consumer endpoint specific information into the Celtix configuration file for the endpoint.

Using Celtix Configuration

The Celtix JMS endpoint configuration properties are specified under the namespace http://celtix.objectweb.org/transports/jms. In order to use the JMS configuration properties you will need to add the line shown in Text 3 to the `beans` element of your configuration.

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

Text 3: JMS Properties Namespace

Consumer endpoint configuration is specified using the `org.objectweb.celtix.bus.transports.jms.jms_client_config.spring.JMSClientConfigBean` class for the configuration bean. Using this configuration bean, you specify the message type supported by the consumer endpoint using the `jmsClient` property. It has a single value, `jms:client`, that has a single attribute:

`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`.

Text 4 shows a Celtix configuration entry for configuring a JMS consumer endpoint.

```
<beans xmlns:ct="http://celtix.objectweb.org/configuration/types"
       xmlns:jms="http://celtix.objectweb.org/transports/jms">
...
  <bean id="celtix.{http://celtix.objectweb.org/jms_conf_test}
HelloWorldQueueBinMsgService/HelloWorldQueueBinMsgPort.jms-client"
        class="org.objectweb.celtix.bus.transports.jms.jms_client_config.spring.JMSC
lientConfigBean">
    <property name="jmsClient">
      <value>
        <jms:client messageType="binary" />
      </value>
    </property>
  </bean>
  ...
</beans>
```

Text 4: Configuration for a JMS consumer endpoint

In addition to specifying the `jmsClient` property, you can also specify the contact information used by the consumer for contacting a service endpoint. This is done by adding the `jmsAddress` property to the consumer endpoint's configuration bean.
For more information on using Celtix configuration see the Celtix Configuration Guide.

Using WSDL

The type of messages accepted by a JMS consumer endpoint is configured using the optional `jms:client` element. The `jms:client` element is a child of the WSDL `port` element and has one attribute:

`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`.

Service Endpoint Configuration

JMS service endpoints have a number of behaviors that are configurable in the contract. These include:

- how messages are correlated
- the use of durable subscriptions
- if the service uses local JMS transactions
- the message selectors used by the endpoint

Service endpoints can be configure in one of two ways:

- Celtix configuration
- WSDL file

Using Celtix Configuration

The Celtix JMS endpoint configuration properties are specified under the namespace http://celtix.objectweb.org/transports/jms. In order to use the JMS configuration properties you will need to add the line shown in Text 5 to the `beans` element of your configuration.

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

Text 5: JMS Properties Namespace

Service endpoint configuration is specified using the `org.objectweb.celtix.bus.transports.jms.jms_server_config.spring.JMSServerConfigBean` class for the configuration bean. Using this configuration bean, you specify the service endpoint's behaviors using the `jmsServer` property. It has a single value, `jms:server`, that has a the following attributes:

`useMessageIDAsCorrealationID`	Specifies whether the JMS broker will use the message ID to correlate messages. The default is `false`.
`durableSubscriberName`	Specifies the name used to register a durable subscription.
`messageSelector`	Specifies the string value of a message selector to use. For more information on the syntax used to specify message selectors, see the JMS 1.1 specification.
`transactional`	Specifies whether the local JMS broker will create transactions around message processing. The default is `false`. Currently, this is not supported by the runtime.

Text 6 shows a Celtix configuration entry for configuring a JMS service endpoint.

```
<beans xmlns:ct="http://celtix.objectweb.org/configuration/types"
       xmlns:jms="http://celtix.objectweb.org/transports/jms">
...
  <bean id="celtix.{http://celtix.objectweb.org/jms_conf_test}
HelloWorldQueueBinMsgService/HelloWorldQueueBinMsgPort.jms-server"
        class="org.objectweb.celtix.bus.transports.jms.jms_server_config.spring.JMSS
erverConfigBean">
    <property name="jmsServer">
      <value>
        <jms:server messageSelector="pickMe"
                    useMessageIDAsCorrelationID="true"
                    transactional="false"
                    durableSubscriberName="CeltixSubscriber" />
      </value>
    </property>
  </bean>
  ...
</beans>
```

Text 6: Configuration for a JMS service endpoint

In addition to specifying the `jmsServer` property, you can also specify the contact information of the service endpoint. This is done by adding the `jmsAddress` property to the service endpoint's configuration bean. For more information on using Celtix configuration see the Celtix Configuration Guide.

Using WSDL

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.
`messageSelector`	Specifies the string value of a message selector to use. For more information on the syntax used to specify message selectors, see the JMS 1.1 specification.
`transactional`	Specifies whether the local JMS broker will create transactions around message processing. The default is `false`. Currently, this is not supported by the runtime.

Using the JMS Context

The Celtix context mechanism can be used to inspect a number of the properties associated with a JMS message. The context mechanism can also be used to overide some of the JMS endpoint's configuration.

Inspecting JMS Properties

Once a message has been successfully retrieved from the JMS transport you can inspect the the JMS properties listed in Table 1 using the response context in a consumer endpoint or the request context in a service endpoint.

Property Name	Property Type
JMSCorrelationID	`string`
JMSDeliveryMode	`int`
JMSExpiration	`long`
JMSMessageID	`string`
JMSPriority	`int`
JMSRedelivered	`boolean`
JMSTimeStamp	`long`
JMSType	`string`
TimeToLive	`long`
JMSClientRecieveTimeOut	`long`

Table 1.: JMS Properties

In addition, you can inspect any optional properties stored in the JMS header using the `JMSPropertyType`. Optional properties are stored as name/value pairs.

Setting JMS Properties

Using the request context in a consumer endpoint or the response context in a service endpoint, you can set the following properties:

- JMSCorrelationID
- JMSDeliveryMode
- TimeToLive
- JMSClientRecieveTimeOut
- optional JMS properties