将 Sun 的 Open Message Queue 与 Spring 集成

刘岩

Email:suhuanzheng7784877@163.com

1. 前言

基于 JMS 标准的消息中间件实现的产品有很多 ,JBossMQ、ActiveMQ、OpenMQ、OpenJMS 等等，最常用的还是 apache 的 ActiveMQ。有时也使用 Sun 的 OpenMQ。在官网 http://mq.java.net/处可以下载。Open Message Queue 是 Sun Java System Message Queue 的一个开源版本。Open message queue 是一个企业级，可升级，非常成熟的消息服务器。它为面向消息的系统集成提供一套完整的 JMS(Java Message Service)实现。由于 Open MQ 源自 Sun 的 Java Message Queue，所以其具有 Java System Message Queue 拥有的所有特性，功能和性能。

2. 环境配置

下载后将相关的 jar 拷贝到项目的 classpath 下面。笔者在此为了安全起见，引入了很多 jar 包，将语言包都引入了。各位读者可以因地制宜。

以下是引入 jar 包的列表

```
lib/openmqjar/common-message.jar
lib/openmqjar/fscontext.jar
lib/openmqjar/grizzly.jar
lib/openmqjar/imq_de.jar
lib/openmqjar/imq_es.jar
lib/openmqjar/imq_fr.jar
lib/openmqjar/imq_it.jar
lib/openmqjar/imq_ja.jar
lib/openmqjar/imq_ko.jar
lib/openmqjar/imq_pt_BR.jar
lib/openmqjar/imq_zh_CN.jar
lib/openmqjar/imq_zh_TW.jar
lib/openmqjar/imq.jar
lib/openmqjar/imqadmin.jar
```

```
lib/openmqjar/imqbridgemgr.jar
lib/openmqjar/imqbroker.jar
lib/openmqjar/imqjmsbridge.jar
lib/openmqjar/imqjmsra.rar
lib/openmqjar/imqjmx_de.jar
lib/openmqjar/imqjmx_es.jar
lib/openmqjar/imqjmx_fr.jar
lib/openmqjar/imqjmx_it.jar
lib/openmqjar/imqjmx_ja.jar
lib/openmqjar/imqjmx_ko.jar
lib/openmqjar/imqjmx_pt_BR.jar
lib/openmqjar/imqjmx_zh_CN.jar
lib/openmqjar/imqjmx_zh_TW.jar
lib/openmqjar/imqjmx.jar
lib/openmqjar/imql10n_server_de.jar
lib/openmqjar/imql10n_server_es.jar
lib/openmqjar/imql10n_server_fr.jar
lib/openmqjar/imql10n_server_it.jar
lib/openmqjar/imql10n_server_ja.jar
lib/openmqjar/imql10n_server_ko.jar
lib/openmqjar/imql10n_server_pt_BR.jar
lib/openmqjar/imql10n_server_zh_CN.jar
lib/openmqjar/imql10n_server_zh_TW.jar
lib/openmqjar/imqservlet.jar
lib/openmqjar/imqstomp.jar
lib/openmqjar/imqutil.jar
lib/openmqjar/imqxm.jar
lib/openmqjar/jaxm-api.jar
lib/openmqjar/jhall.jar
lib/openmqjar/jms.jar
lib/openmqjar/jta.jar
lib/openmqjar/protobuf-2.3.0.jar
```

3. 之后项目加入 Spring 的相关 jar 包。

增加 Spring 配置文件内容如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:context="http://www.springframework.org/schema/context"
    xsi:schemaLocation="http://www.springframework.org/schema/bean
s

http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
```

```
        http://www.springframework.org/schema/context

http://www.springframework.org/schema/context/spring-context-2.5.x
sd"
    default-autowire="byName">


    <!--消息连接工厂-->

    <bean id="connectionfactoryfactory"
        class="message.listener.OpenMqConnectionFactory">
        <property name="properties">
            <props>
                <prop key="imqAddressList">127.0.0.1:7676</prop>
                <prop key="imqDefaultUsername">admin</prop>
                <prop key="imqDefaultPassword">admin</prop>
                <prop key="imqReconnectEnabled">true</prop>
                <prop key="imqReconnectAttempts">3</prop>
                <prop key="imqReconnectInterval">5000</prop>
                <prop key="imqAddressListBehavior">RANDOM</prop>
            </props>
        </property>
    </bean>

    <bean id="mqConnectionFactory"
factory-bean="connectionfactoryfactory"
        factory-method="createConnectionFactory" />


    <!--设置广发消息目的-->

    <bean id="updateLocalRouteMap" class="com.sun.messaging.Topic">
        <constructor-arg type="java.lang.String" value="mytopic" />
    </bean>

    <bean id="jmsTemplate"
class="org.springframework.jms.core.JmsTemplate">
        <property name="connectionFactory" ref="mqConnectionFactory"
/>
        <property name="defaultDestination"
ref="updateLocalRouteMap" />
        <property name="receiveTimeout" value="20000" />
    </bean>


    <!--消息监听器-->
```

```
    <bean id="messageListener1"

    class="org.springframework.jms.listener.adapter.MessageListene
rAdapter">
        <constructor-arg>
            <bean
                class="message.listener.JMSMessageListener" />
        </constructor-arg>
        <property name="defaultListenerMethod" value="receive" />
        <property name="messageConverter">
            <null />
        </property>
    </bean>


    <!—实际的消息监消费者配置-->
    <bean id="consumercontainer"

    class="org.springframework.jms.listener.DefaultMessageListener
Container">
        <property name="connectionFactory" ref="mqConnectionFactory"
/>
        <property name="destination" ref="updateLocalRouteMap" />
        <property name="messageListener" ref="messageListener1" />
        <property name="transactionTimeout" value="180000" />
        <property name="receiveTimeout" value="180000" />
        <property name="sessionTransacted" value="true" />
    </bean>
</beans>
```

4. 消息监听器

类代码如下

```
/**
 * JMS消息消费者。
 * 接收JMS消息后获得router想要的消息后，调用router接口更新本地缓存
 * @author liuyan
 *
 */
public class JMSMessageListener implements MessageListener {

    private Logger log =
```

```
Logger.getLogger(JMSMessageListener.class.getName());

    /**
     * 接收JMS消息后的业务处理
     */
    public void onMessage(Message message) {


        log.info("接收消息……");
        byte[] byteMessage = JMSByteConverterUtil
                .ConverterMessageToBttes(message);
        try {


            log.info("将转型成实体对象……");

            //.....................................................


            }

        } catch (InvalidProtocolBufferException e) {
            log.error("JMS异常" + e.getMessage());
            e.printStackTrace();
        } catch (Exception e) {
            log.error("其他异常" + e.getMessage());
            e.printStackTrace();
        }
    }
}
```

因为一些原因此处就不给出完整代码了~~~反正是获取一个字节流后，转成对象，直接从对象中获取想要的信息。转成对象的辅助类如下

```
/**
 * 对获得的消息对象进行转型
 * @author liuyan
 */
public class JMSByteConverterUtil {

    private static Logger log =
```

```
Logger.getLogger(JMSMessageListener.class
            .getName());

    /**
     * 对获得的消息对象进行转型
     * @param message
     * @return
     */
    public static byte[] ConverterMessageToBttes(Message message) {

        if (message == null) {
            log.error("消息对象为空……");
            return null;
        } else if (message instanceof BytesMessage) {

            log.debug("消息强制转型BytesMessage");
            BytesMessage bytesMessage = (BytesMessage) message;

            byte[] messageBytes;
            try {

                log.debug("建立空的消息二进制数组");
                messageBytes = new byte[(int)
bytesMessage.getBodyLength()];

                log.debug("往二进制数组中写进二进制信息");
                bytesMessage.readBytes(messageBytes);

                log.debug("messageBytes.length=" +
messageBytes.length);
                return messageBytes;

            } catch (JMSException e) {
                log.error("JMS错误：" + e.getMessage());
                e.printStackTrace();
                return null;
            }
```

```
        }else{


            log.error("消息对象不能正确转型");

            return null;
        }

    }
}
```

5. 启动消息监听器

开启 OpenMQ 的服务，启动{OpenMQ_HOME}\mq\bin\下的 imqcmd.exe 命令

启动消息消费者很简单，代码如下

```
public class MessageConsumer {

    /**
     * @param args
     */
    public static void main(String[] args) {
        ApplicationContext applicationContext = new
ClassPathXmlApplicationContext(
                new String[]
{ "classpath:/spring/applicationContext-openmq-jms.xml" });

        System.out.println(applicationContext.getId());

    }
}
```

6. 消息发送者

启动消息消费者服务后，写一个测试类测试一下消息的，代码如下

```
public class MessageSender {

    /**
     * @param args
     * @throws JMSException
     */
    public static void main(String[] args) throws JMSException {
        ConnectionFactory myConnFactory;
        myConnFactory = new com.sun.messaging.ConnectionFactory();
```

```
    myConnFactory.setProperty(ConnectionConfiguration.imqAddressLi
st,
                "mq://127.0.0.1:7676");

    myConnFactory.setProperty(ConnectionConfiguration.imqReconnect
Enabled,
                "true");
        Connection myConn = myConnFactory.createConnection();
        myConn.start();
        // Step 4:
        // Create a session within the connection.
        Session mySess = myConn.createSession(false,
Session.AUTO_ACKNOWLEDGE);
        Topic myTopic = new
com.sun.messaging.Topic("testmq");// .Queue("testmq");
        MessageProducer myMsgProducer =
mySess.createProducer(myTopic);

        ObjectMessage objectMessage = mySess.createObjectMessage();

        RouterMessageBean routerMessageBean = new
RouterMessageBean();
        routerMessageBean.setDbName("mysql-test");
        routerMessageBean.setUserName("liuyan");
        routerMessageBean.setMaster(null);
        routerMessageBean.setSlave(null);

        objectMessage.setObject(routerMessageBean);

        BytesMessage bytesMessage = mySess.createBytesMessage();

        bytesMessage.writeUTF("the message is 消息内容！");



        myMsgProducer.send(bytesMessage);

        System.out.println("测试发送JMS消息");


        mySess.close();
        myConn.close();

    }
}
```