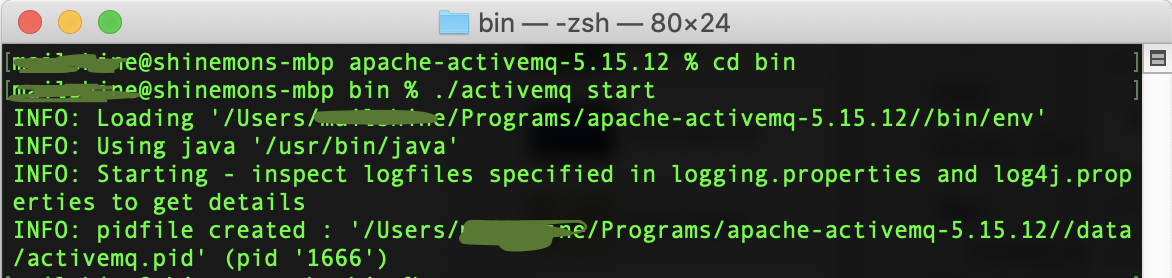
**ActiveMQ — Getting Started with SpringBoot.**

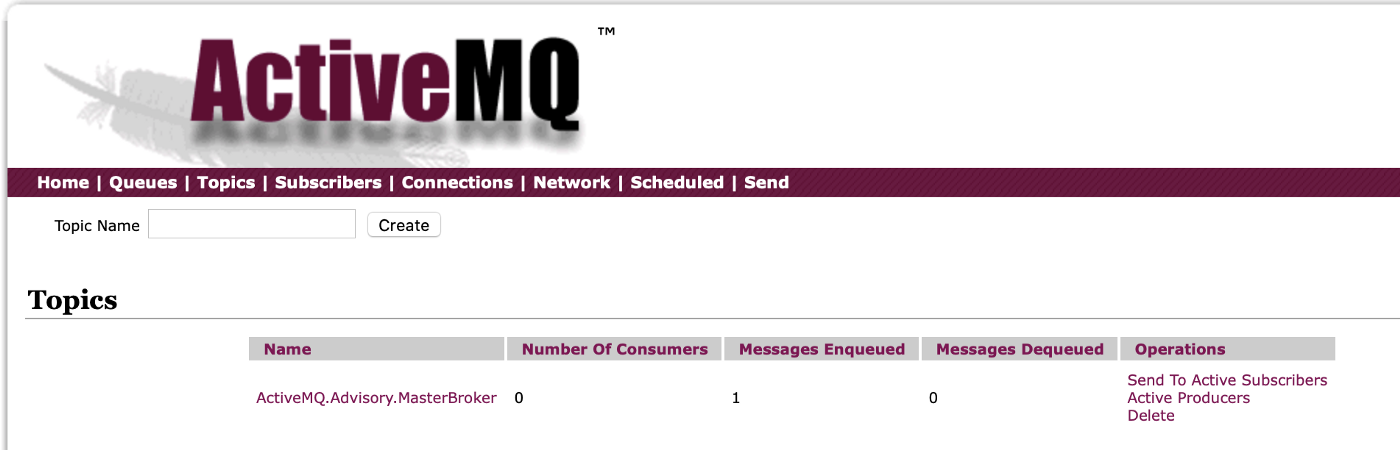
In this post, we will attempt to set up a Standalone ActiveMQ Broker/Server instead of the embedded SpringBoot ActiveMQ broker. And establish a Message Producer to the Queue, and the Message consumer.

**Standalone ActiveMQ:**

1. Download ActiveMQ binary from [here](http://activemq.apache.org/download.html).
2. Unzip your bundle, and Open Terminal.
3. Set the Terminal path to ActiveMQ -> bin
4. Write command **./activemq start**



5. Open [***http://localhost:8161/admin/***](http://localhost:8161/admin/) to confirm the ActiveMQ running.



**Configure Message Producer:**

We will use Java SpringBoot framework, with activemq starter dependency.

<**dependency**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-starter-activemq</**artifactId**>  
</**dependency**>

ActiveMQ Producer Config:

@Configuration  
**public class** ActiveMQConfig {  
  
 @Value(**"${active-mq.broker-url}"**)  
 **private** String **brokerUrl**;  
  
 @Bean  
 **public** ConnectionFactory connectionFactory(){  
 ActiveMQConnectionFactory activeMQConnectionFactory = **new** ActiveMQConnectionFactory();  
 activeMQConnectionFactory.setBrokerURL(**brokerUrl**);  
 **return** activeMQConnectionFactory;  
 }  
  
 @Bean  
 **public** JmsTemplate jmsTemplate(){  
 JmsTemplate jmsTemplate = **new** JmsTemplate();  
 jmsTemplate.setConnectionFactory(connectionFactory());  
 jmsTemplate.setPubSubDomain(**true**); *// enable for Pub Sub to topic. Not Required for Queue.* **return** jmsTemplate;  
 }  
}

**ActiveMQConnectionFactory** will establish connection with our ActiveMQ broker, this can be used for both Queue Connection and Topic Connection. As this implements javax.jms.ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory.

**JmsTemplate** will be used for sending messages to the Topic/Queue. JmsTemplate.**send()** can be used for sending text messages. For sending custom messages we can make use of JmsTemplate.**convertAndSend()**.

Adding **JmsProducer** to send our message.

@Component  
@Slf4j  
**public class** JmsProducer {  
  
 @Autowired  
 JmsTemplate **jmsTemplate**;  
  
 @Value(**"${active-mq.topic}"**)  
 **private** String **topic**;  
  
 **public void** sendMessage(Employee message){  
 **try**{  
 ***log****.info(****"Attempting Send message to Topic: "****+* ***topic****);*  
 **jmsTemplate**.convertAndSend(**topic**, message);  
 } **catch**(Exception e){  
 ***log****.error(****"Recieved Exception during send Message: "****, e);*  
 }  
 }  
}

Now we will expose an api to produce message:

@RestController  
@Slf4j  
**public class** ProduceMessageController {  
  
 @Autowired  
 JmsProducer **jmsProducer**;  
  
 @PostMapping(value=**"/api/employee"**)  
 **public** Employee sendMessage(@RequestBody Employee employee){  
 **jmsProducer**.sendMessage(employee);  
 **return** employee;  
 }  
}

This should successfully publish our messages to the ActiveMQ Topic.

**Configure Message Consumer:**

We will create Bean of DefaultJmsListenerContainerFactory, where we use the ActiveMQConnectionFactory.

@Bean  
**public** ConnectionFactory connectionFactory(){  
 ActiveMQConnectionFactory activeMQConnectionFactory = **new** ActiveMQConnectionFactory();  
 activeMQConnectionFactory.setBrokerURL(**brokerUrl**);  
 activeMQConnectionFactory.setTrustedPackages(Arrays.*asList*(**"com.mailshine.springbootstandaloneactivemq"**));  
 **return** activeMQConnectionFactory;  
}@Bean  
**public** DefaultJmsListenerContainerFactory jmsListenerContainerFactory(){  
 DefaultJmsListenerContainerFactory factory = **new** DefaultJmsListenerContainerFactory();  
 factory.setConnectionFactory(connectionFactory());  
 factory.setPubSubDomain(**true**);  
 **return** factory;  
}

Now will add the Listener:

@Component  
@Slf4j  
**public class** JmsConsumer **implements** MessageListener {  
  
  
 @Override  
 @JmsListener(destination = **"${active-mq.topic}"**)  
 **public void** onMessage(Message message) {  
 **try**{  
 ObjectMessage objectMessage = (ObjectMessage)message;  
 Employee employee = (Employee)objectMessage.getObject();  
 *//do additional processing* ***log***.info(**"Received Message: "**+ employee.toString());  
 } **catch**(Exception e) {  
 ***log***.error(**"Received Exception : "**+ e);  
 }  
  
 }  
}

Start the Spring Boot App.

Attempting: curl -X POST “<http://localhost:8080/api/employee>" -H “accept: \*/\*” -H “Content-Type: application/json” -d “{ \”employeeFirstName\”: \”shine\”, \”employeeFullName\”: \”shine m test\”, \”employeeId\”: \”129\”, \”employeeLastName\”: \”test\”, \”employeeMiddleName\”: \”m\”}”

Or Post call to <http://localhost:8080/api/employee>

The message should be published successfully to the ActiveMQ Topic.

The message should also be successfully subscribed by the consumer from the Topic.