**Spring JMS**

As usual, Spring and Spring Boot make things quite simple for us to implement allowing the basic code to be written quickly and without loads of code.

JMS and message queues, in general, bring some certain advantages over using RESTful services such as:

* **Redundancy**. A message must confirm that it has completed its transaction and that it can now be removed from the queue, but if the transaction fails it can be reprocessed. The messages can also be stored in a database allowing them to continue later on even if the server stops.
* **Asynchronous messaging**. As the process time of the message cannot be guaranteed, the client that sent the message can carry on asynchronously to the completion of the transaction. Due to this, the queue should be used to write data (POST if you're thinking in a RESTful mindset).
* **Loose coupling**. The services do not interact directly and only know where the message queue is, where one service sends messages and the other receives them.

To create JMS application using SpringBoot-

Step 1)Create a spring boot project and add following dependency in pom.xml-

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-activemq</artifactId>

</dependency>

<dependency>

<groupId>org.apache.activemq</groupId>

<artifactId>activemq-broker</artifactId>

</dependency>

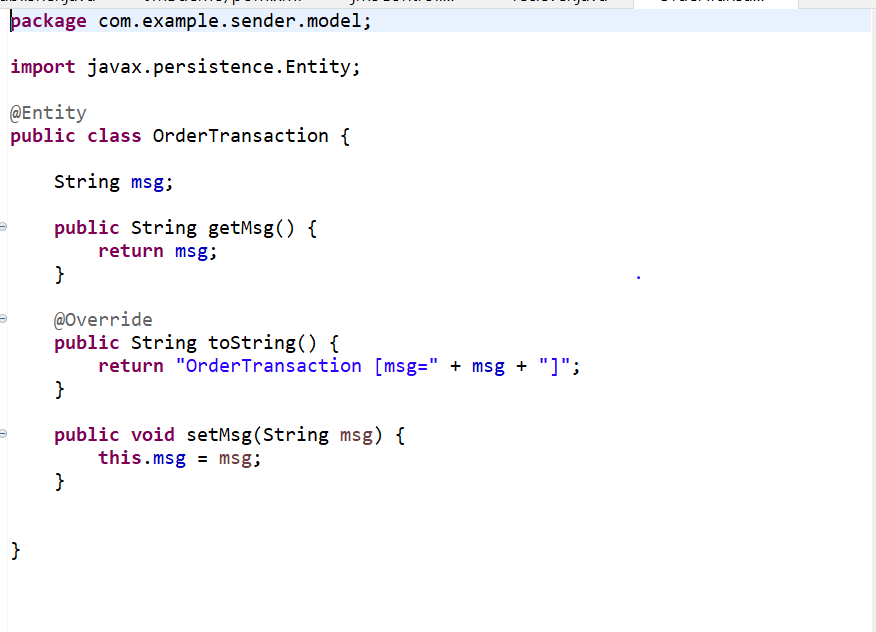
<dependency>

<groupId>com.fasterxml.jackson.core</groupId>

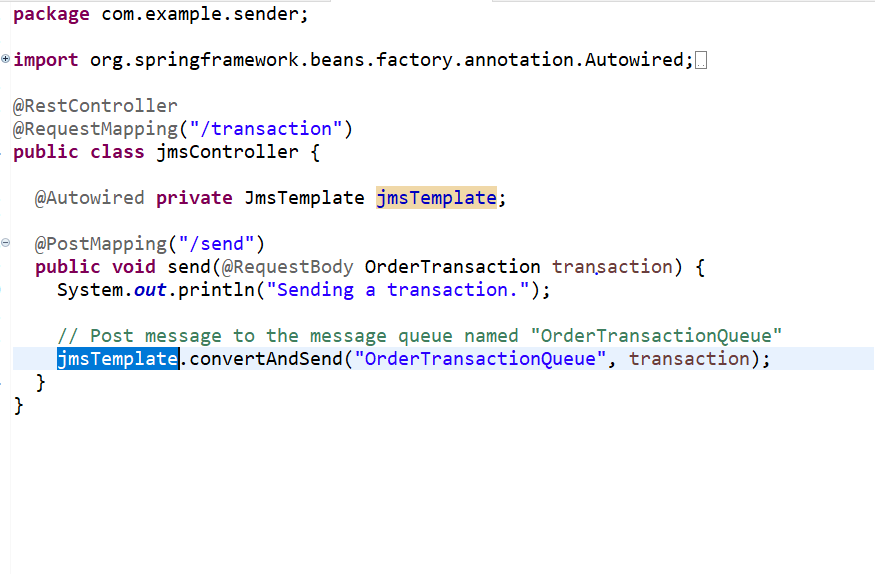
<artifactId>jackson-databind</artifactId>

</dependency>

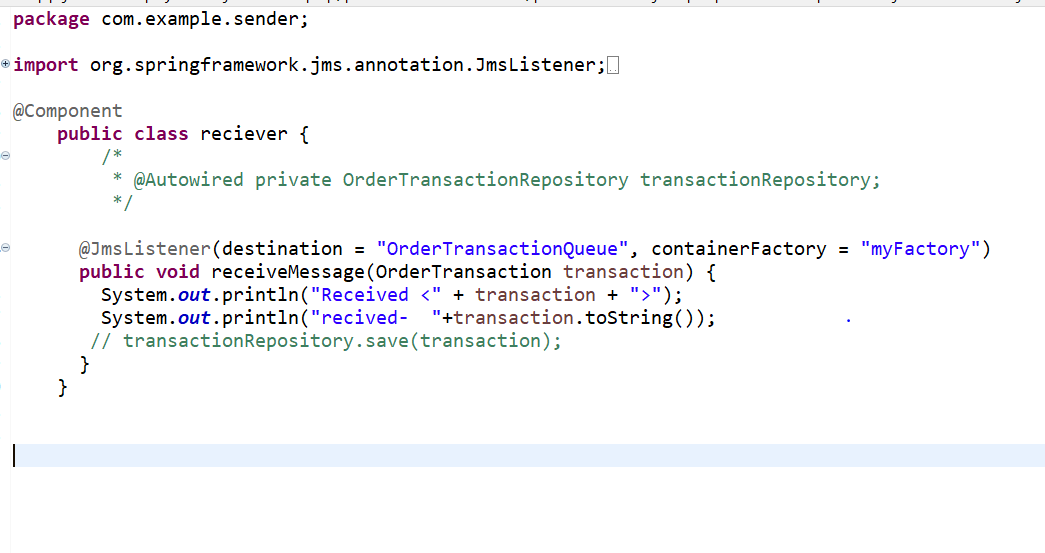
Step 2)Create a model class-



Step 3)Sending message- Create a class in project,here we will receive message by using *jmsTemplate*-



Step 4-Recieving message-create class in project and then add annotation @JmsListener to receive messages-



Step 5)Configure main class-

* add annotation @EnableJms  which triggers the discovery of methods marked with the @JmsListener and creates the listeners themselves behind the scenes
* Define myFactory bean-

@Bean

**public** JmsListenerContainerFactory<?> myFactory(

ConnectionFactory connectionFactory,

DefaultJmsListenerContainerFactoryConfigurer configurer) {

DefaultJmsListenerContainerFactory factory = **new**

DefaultJmsListenerContainerFactory();

configurer.configure(factory, connectionFactory);

**return** factory;

}

* Define jacksonJmsMessageConverter bean-

@Bean

**public** MessageConverter jacksonJmsMessageConverter() {

MappingJackson2MessageConverter converter = **new** MappingJackson2MessageConverter();

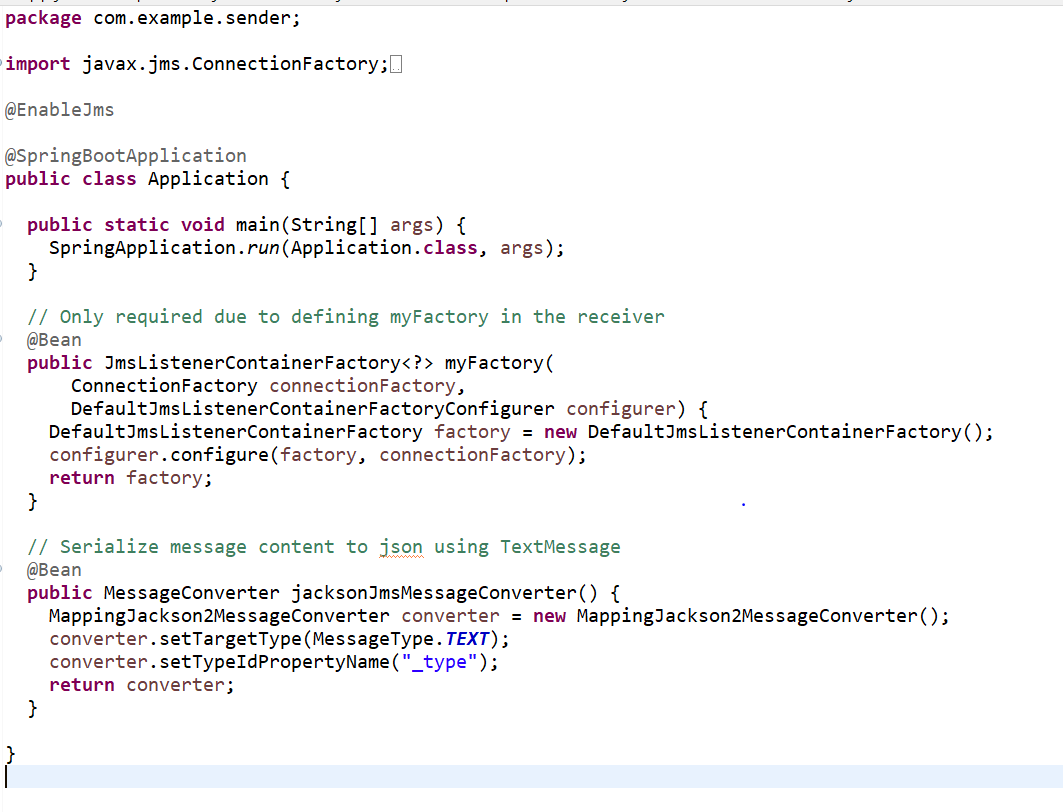
converter.setTargetType(MessageType.***TEXT***);

converter.setTypeIdPropertyName("\_type");

**return** converter;

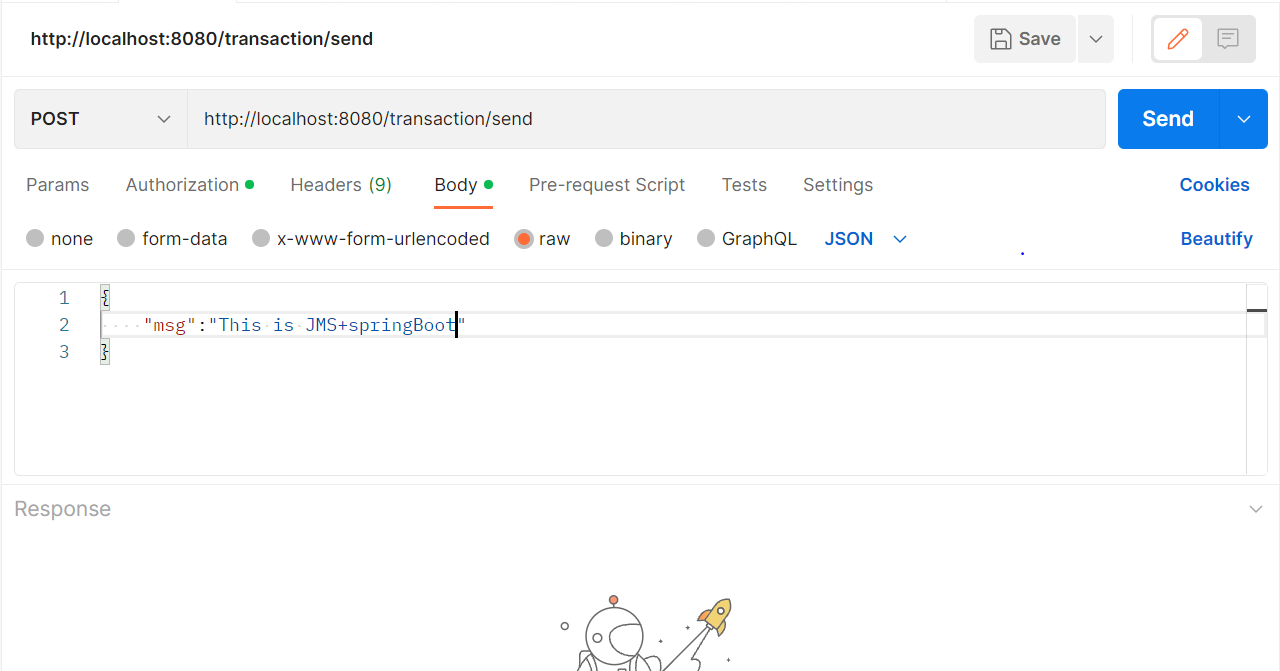
}

Main class should look something like this-

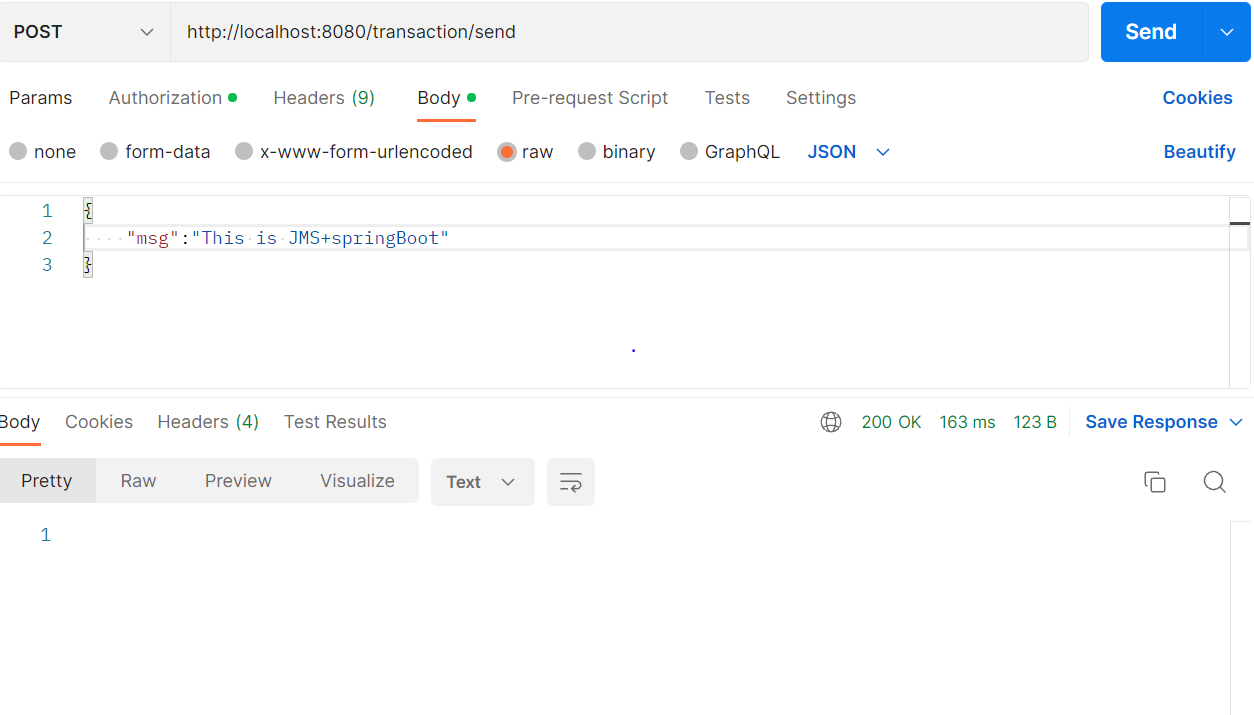


Step 6-Now run the application and test it via post man

Step 7-Open post man and hit api- /send-



You must get 200 response-



Step 8-Verify same response by sys out from receiver class-

