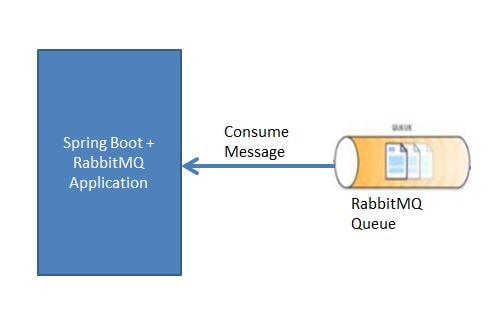
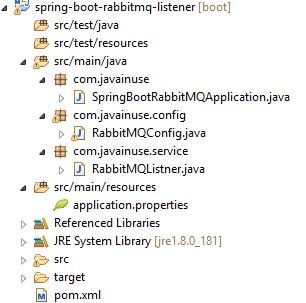
**Spring Boot + RabbitMQ Tutorial - Configure Listeners to consume messages**

In a previous tutorial we had implemented a [Spring Boot + RabbitMQ example to send publish message to RabbitMQ Queue](https://www.javainuse.com/misc/rabbitmq-hello-world). In this tutorial we will be implementing a Spring Boot + RabbitMQ example to consume message from a RabbitMQ Queue. In next tutorial we will be [exploring the various RabbitMQ Exchange types and implementing them using Spring Boot.](https://www.javainuse.com/messaging/rabbitmq/exchange)  


## Let's Begin

The Maven project will be as follows-  
  
The pom.xml will have the following dependencies-

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.javainuse</groupId>

<artifactId>spring-boot</artifactId>

<version>0.0.1-SNAPSHOT</version>

<parent>

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

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

<version>2.1.1.RELEASE</version>

<relativePath /> <!-- lookup parent from repository -->

</parent>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>

<java.version>1.8</java.version>

</properties>

<dependencies>

**<dependency>**

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

**<artifactId>spring-boot-starter-amqp</artifactId>**

**</dependency>**

</dependencies>

</project>

We will first be creating a listener class which implements the AMQP MessageListener interface. This class is responsible for getting the message from the RabbitMQ queue.

package com.javainuse.service;

import org.springframework.amqp.core.Message;

import org.springframework.amqp.core.MessageListener;

import org.springframework.stereotype.Service;

@Service

public class RabbitMQListner **implements MessageListener** {

public void onMessage(Message message) {

System.out.println("Consuming Message - " + new String(message.getBody()));

}

}

Next we will be creating the Spring Boot Configuration class for RabbitMQ. We will

* Create Queue bean using which we create a RabbitMQ named javainuse.input-queue. This will be a non durable queue. Do not misunderstand a non durable queue to be a temporary queue. Durability property is related to how long a message will be stored in the queue. For example for in RabbitMQ restart messages in non durable queue will be lost while those in durable queue will not be lost.
* Create MessageListenerContainer where we configure the RabbitMQConnections. Spring MessageListenerContainer is a replacement for a Message-Driven EJB. A connection is set up with the AMQ topic/queue, it gets messages from that topic/queue and feeds them to your MessageListener. We will be making use of the default connectionfactory. If we do not wish to use the default connectionfactory we can can create our own CachingConnectionFactory and use it.

package com.javainuse.config;

import org.springframework.amqp.core.Queue;

import org.springframework.amqp.rabbit.connection.CachingConnectionFactory;

import org.springframework.amqp.rabbit.connection.ConnectionFactory;

import org.springframework.amqp.rabbit.listener.MessageListenerContainer;

import org.springframework.amqp.rabbit.listener.SimpleMessageListenerContainer;

import org.springframework.beans.factory.annotation.Value;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import com.javainuse.service.RabbitMQListner;

@Configuration

public class RabbitMQConfig {

@Value("${javainuse.rabbitmq.queue}")

String queueName;

@Value("${spring.rabbitmq.username}")

String username;

@Value("${spring.rabbitmq.password}")

private String password;

@Bean

Queue queue() {

return new Queue(queueName, false);

}

**//create MessageListenerContainer using default connection factory**

@Bean

MessageListenerContainer messageListenerContainer(ConnectionFactory connectionFactory ) {

SimpleMessageListenerContainer simpleMessageListenerContainer = new SimpleMessageListenerContainer();

simpleMessageListenerContainer.setConnectionFactory(connectionFactory);

simpleMessageListenerContainer.setQueues(queue());

simpleMessageListenerContainer.setMessageListener(new RabbitMQListner());

return simpleMessageListenerContainer;

}

**//create custom connection factory**

/\*@Bean

ConnectionFactory connectionFactory() {

CachingConnectionFactory cachingConnectionFactory = new CachingConnectionFactory("localhost");

cachingConnectionFactory.setUsername(username);

cachingConnectionFactory.setUsername(password);

return cachingConnectionFactory;

}\*/

**//create MessageListenerContainer using custom connection factory**

/\*@Bean

MessageListenerContainer messageListenerContainer() {

SimpleMessageListenerContainer simpleMessageListenerContainer = new SimpleMessageListenerContainer();

simpleMessageListenerContainer.setConnectionFactory(connectionFactory());

simpleMessageListenerContainer.setQueues(queue());

simpleMessageListenerContainer.setMessageListener(new RabbitMQListner());

return simpleMessageListenerContainer;

}\*/

}

In the application.properties define the following-

spring.rabbitmq.host=localhost

spring.rabbitmq.port=5672

spring.rabbitmq.username=guest

spring.rabbitmq.password=guest

javainuse.rabbitmq.queue=javainuse.input-queue

Finally create the bootstrap class which makes use of the SpringBootApplication Annotation

package com.javainuse;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

**@SpringBootApplication**

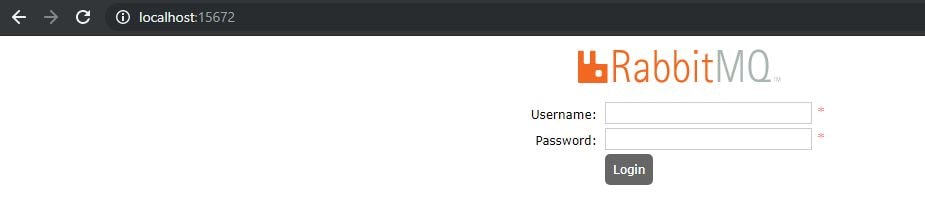
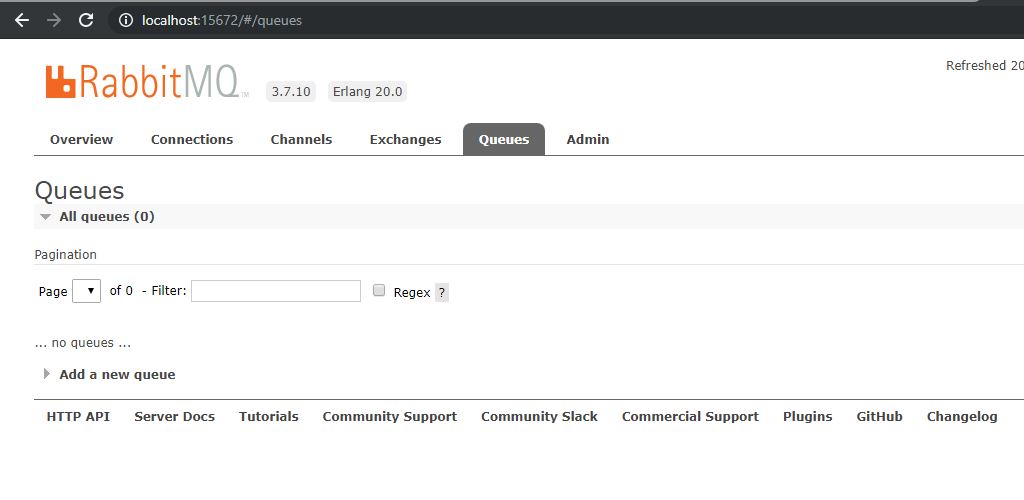
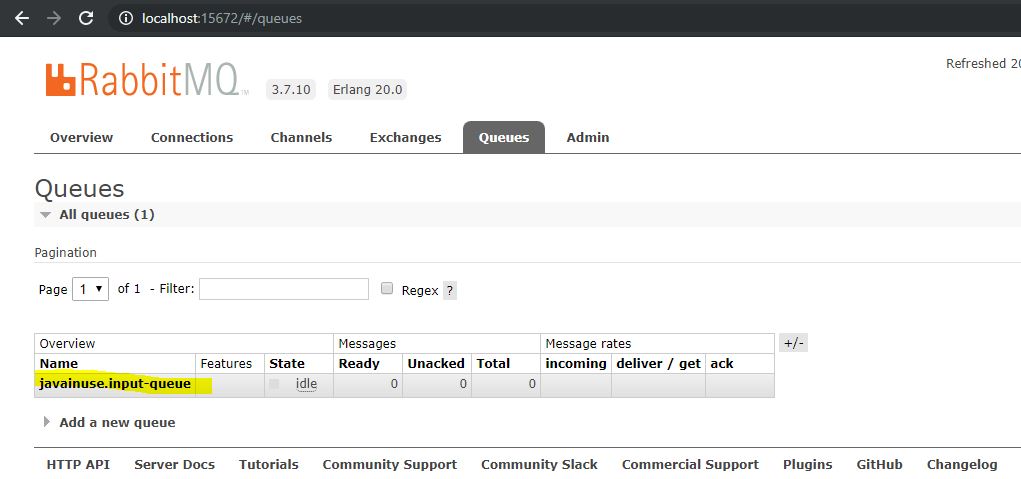
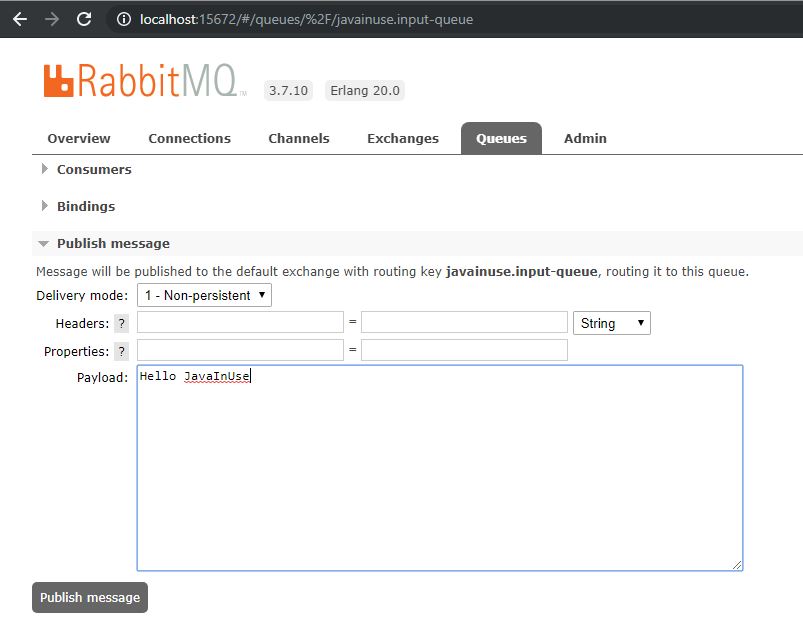
public class SpringBootRabbitMQApplication {

public static void main(String[] args) {

SpringApplication.run(SpringBootRabbitMQApplication.class, args);

}

}

In a [previous tutorial we have shown how to install RabbitMQ and get started.](https://www.javainuse.com/misc/rabbitmq-hello-world) Once you have followed implemented this tutorial go to localhost:15672  
  
Use the username and password as guest.If we now go to the queues section, currently there are no queues  
  
Start the Spring Boot Application. If we now again go to the RabbitMQ Management Console Queues section we can see that a queue named javainuse.input-queue has been created.  
  
Now select the queue and publish a message.  
  
If we now go to the eclipse console we can see that our application has consumed the RabbitMQ message and printed its content.  
