1) Intall RabbitMQ on Windows

a) Download and Install Erlang https://www.erlang.org/downloads

b) Download and Install RabbitMQ https://www.rabbitmq.com/install-windows.html#installer

2) Set system path to C:\Program Files\RabbitMQ Server\rabbitmq_server-3.11.3\sbin

3) Stop rabbitmq process if its running in windows services

3) Start rabbitmq server

```
rabbitmq-server start
```

4) Set up the RabbitMQ plugin by using below command, to use the RabbitMQ Management Console from Web Browser.

```
rabbitmq-plugins.bat enable rabbitmq_management
```

5) On browser open http://localhost:15672, to see login page. Use guest as username and password.

----

6) Run RabbitMQ using docker

a) Pull RabbitMQ 3-management docker image -

```
docker pull rabbitmq:3-management
```

b) Run the docker image which will map port 15672 and the message broker port 5672

```
docker run --rm -it -p 15672:15672 -p 5672:5672 rabbitmq:3-management
```

----

7) Run RabbitMQ using docker-compose

a) Create docker-compose.yml file

```
version: "3.2"
services:
    rabbitmq:
        image: rabbitmq:3-management-alpine
        container_name: 'rabbitmq'
        ports:
            - 5672:5672
            - 15672:15672
        volumes:
            - ~/.docker-conf/rabbitmq/data/:/var/lib/rabbitmq/
            - ~/.docker-conf/rabbitmq/log/:/var/log/rabbitmq
        networks:
            - rabbitmq_go_net
networks:
    rabbitmq_go_net:
        driver: bridge
```

----

8) Produce/Consume RabbitMQ messages using spring boot application

a) Producer App

i) Add dependency

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-amqp</artifactId>
</dependency>
```

ii) Add following properties in application.properties

```
spring.rabbitmq.host=localhost
spring.rabbitmq.port=5672

spring.rabbitmq.username=guest
spring.rabbitmq.password=guest

rabbitmq.exchange=ssce.exchange
rabbitmq.queue=ssce.queue
rabbitmq.routingkey=ssce.routingkey
```

iii) Create RabbitMQ configuration file (RabbitMQConfig.java)

```
package com.ssce.SpringBootRabbitMQExample.config;

import org.springframework.amqp.core.*;
import org.springframework.amqp.rabbit.connection.ConnectionFactory;
import org.springframework.amqp.rabbit.core.RabbitTemplate;
import org.springframework.amqp.support.converter.Jackson2JsonMessageConverter;
import org.springframework.amqp.support.converter.MessageConverter;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
public class RabbitMQConfig {

   @Value("${rabbitmq.queue}")
   String queueName;

   @Value("${rabbitmq.exchange}")
   String exchange;

   @Value("${rabbitmq.routingkey}")
   private String routingkey;

   @Bean
   Queue queue() {
      return new Queue(queueName, false);
   }

   @Bean
   DirectExchange exchange() {
      return new DirectExchange(exchange);
   }

   @Bean
   Binding binding(Queue queue, DirectExchange exchange) {
      return BindingBuilder.bind(queue).to(exchange).with(routingkey);
   }

   @Bean
   public MessageConverter jsonMessageConverter() {
      return new Jackson2JsonMessageConverter();
   }

   public AmqpTemplate rabbitTemplate(ConnectionFactory connectionFactory) {
      final RabbitTemplate rabbitTemplate = new RabbitTemplate(connectionFactory);
      rabbitTemplate.setMessageConverter(jsonMessageConverter());
      return rabbitTemplate;
   }
}
```

iii) Create service class that producesmessages (RabbitMQSender.java)

```
package com.ssce.SpringBootRabbitMQExample.service;

import com.ssce.SpringBootRabbitMQExample.model.Employee;
import org.springframework.amqp.core.AmqpTemplate;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Service;

@Service
public class RabbitMQSender {

    @Autowired
    private AmqpTemplate amqpTemplate;

    @Value("${rabbitmq.exchange}")
    private String exchange;

    @Value("${rabbitmq.routingkey}")
    private String routingkey;

    public void send(Employee employee) {
        amqpTemplate.convertAndSend(exchange, routingkey, employee);
        System.out.println("Send msg = " + employee);
    }
}
```

iv) Create REST controller class (RabbitMQController.java)

```
package com.ssce.SpringBootRabbitMQExample.controller;

import com.ssce.SpringBootRabbitMQExample.model.Employee;
import com.ssce.SpringBootRabbitMQExample.service.RabbitMQSender;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping(value = "/ssce")
public class RabbitMQWebController {

    @Autowired
    private RabbitMQSender rabbitMQSender;

   //localhost:8080/ssce/rabbitmq/send/message?empName=Rakesh Sharma&empId=1001
    @GetMapping(value = "/rabbitmq/send/message")
    public String produce(@RequestParam("empName") String empName,
                                            @RequestParam("empId") String empId) {

        System.out.println("Inside controller");
        Employee emp=new Employee();
        emp.setEmpId(empId);
        emp.setEmpName(empName);
        rabbitMQSender.send(emp);

        return "Message has been sent to the RabbitMQ techgeeknext successfully";
    }

}
```

b) Consumer App

i) Add dependency

```
<dependency>

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

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

</dependency>
```

ii) Add following properties in application.properties

```
spring.rabbitmq.host=localhost
spring.rabbitmq.port=5672

spring.rabbitmq.username=guest
spring.rabbitmq.password=guest

rabbitmq.exchange=ssce.exchange
rabbitmq.queue=ssce.queue
rabbitmq.routingkey=ssce.routingkey
```

iii) Create RabbitMQ configuration file (RabbitMQConfig.java)

```
package com.ssce.SpringBootRabbitMQReceiver.config;

import org.springframework.amqp.core.*;
import org.springframework.amqp.rabbit.connection.ConnectionFactory;
import org.springframework.amqp.rabbit.core.RabbitTemplate;
import org.springframework.amqp.support.converter.Jackson2JsonMessageConverter;
import org.springframework.amqp.support.converter.MessageConverter;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
public class RabbitMQConfig {

    @Value("${rabbitmq.queue}")
    String queueName;

    @Value("${rabbitmq.exchange}")
    String exchange;

    @Value("${rabbitmq.routingkey}")
    private String routingkey;

    @Bean
    Queue queue() {
        return new Queue(queueName, false);
    }

    @Bean
    DirectExchange exchange() {
        return new DirectExchange(exchange);
    }

    @Bean
    Binding binding(Queue queue, DirectExchange exchange) {
        return BindingBuilder.bind(queue).to(exchange).with(routingkey);
    }
    @Bean
    public MessageConverter jsonMessageConverter() {
        return new Jackson2JsonMessageConverter();
    }

    public AmqpTemplate rabbitTemplate(ConnectionFactory connectionFactory) {
        final RabbitTemplate rabbitTemplate = new RabbitTemplate(connectionFactory);
        rabbitTemplate.setMessageConverter(jsonMessageConverter());
        return rabbitTemplate;
    }
}
```

iv) Create receiver listener service component

```
package com.ssce.SpringBootRabbitMQReceiver.config;


import com.ssce.SpringBootRabbitMQReceiver.model.Employee;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.amqp.rabbit.annotation.RabbitListener;
import org.springframework.amqp.rabbit.annotation.RabbitListenerConfigurer;
import org.springframework.amqp.rabbit.listener.RabbitListenerEndpointRegistrar;
import org.springframework.stereotype.Component;

@Component
public class RabbitMqReceiver implements RabbitListenerConfigurer {
    private static final Logger logger = LoggerFactory.getLogger(RabbitMqReceiver.class);

    @Override
    public void configureRabbitListeners(RabbitListenerEndpointRegistrar
                                                        rabbitListenerEndpointRegistrar) {
    }

    @RabbitListener(queues = "${rabbitmq.queue}")
    public void receivedMessage(Employee employee) {

        logger.info("User Details Received is.. " + employee);
    }
}
```