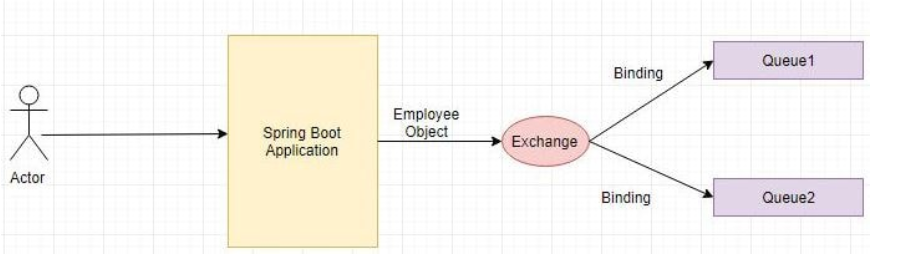
**SpringBoot with RabbitMQ Demo**



Define the **pom.xml** and add the **spring-boot-starter-amqp** dependency.

**<dependency>**

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

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

**</dependency>**

<dependency>

**<groupId>org.json</groupId>**

**<artifactId>json</artifactId>**

</dependency>

Define the domain class Employee as follows-

**package** com.cloud.model;

**import** com.fasterxml.jackson.annotation.JsonIdentityInfo;

**import** com.fasterxml.jackson.annotation.ObjectIdGenerators;

@JsonIdentityInfo(generator = ObjectIdGenerators.IntSequenceGenerator.**class**, property = "@id", scope = Employee.**class**)

**public** **class** Employee {

**private** String empName;

**private** String empId;

**public** String getEmpName() {

**return** empName;

}

**public** **void** setEmpName(String empName) {

**this**.empName = empName;

}

**public** String getEmpId() {

**return** empId;

}

**public** **void** setEmpId(String empId) {

**this**.empId = empId;

}

@Override

**public** String toString() {

**return** "Employee [empName=" + empName + ", empId=" + empId + "]";

}

}

Next define the configuration as follows-

**package** com.cloud.config;

**import** org.springframework.amqp.core.AmqpTemplate;

**import** org.springframework.amqp.core.Binding;

**import** org.springframework.amqp.core.BindingBuilder;

**import** org.springframework.amqp.core.DirectExchange;

**import** org.springframework.amqp.core.Queue;

**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("${springboot.rabbitmq.queue}")

String queueName;

@Value("${springboot.rabbitmq.exchange}")

String exchange;

@Value("${springboot.rabbitmq.routingkey}")

**private** String routingkey;

@Bean

Queue queue() {

**return** **new** Queue(queueName, **false**);

}

@Bean

DirectExchange exchange() {

**return** **new** DirectExchange(exchange);

}

@Bean

Binding binding(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 = **new** RabbitTemplate(connectionFactory);

rabbitTemplate.setMessageConverter(jsonMessageConverter());

**return** rabbitTemplate;

}

}

Define the Controller to expose a GET Request API as follows-

**package** com.cloud.controller;

**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;

**import** com.cloud.model.Employee;

**import** com.cloud.service.RabbitMQSender;

@RestController

@RequestMapping(value = "/springboot-rabbitmq/")

**public** **class** RabbitMQWebController {

@Autowired

RabbitMQSender;

@GetMapping(value = "/producer")

**public** String producer(@RequestParam("empName") String empName,@RequestParam("empId") String empId) {

Employee emp=**new** Employee();

emp.setEmpId(empId);

emp.setEmpName(empName);

rabbitMQSender.send(emp);

**return** "Message sent to the RabbitMQ Successfully";

}

}

Define the RabbitMQSender class which sends the message to the RabbitMQ using AmqpTemplate. We use the exchange and the exchange key.  
Exchanges are message routing agents, defined per virtual host within RabbitMQ. An exchange is responsible for the routing of the messages to the different queues. An exchange accepts messages from the producer application and routes them to message queues with help of header attributes, bindings, and routing keys.  
We will use a direct exchange instead. The routing algorithm behind a direct exchange is simple - a message goes to the queues whose binding key exactly matches the routing key of the message.

**package** com.cloud.service;

**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;

**import** com.cloud.model.Employee;

@Service

**public** **class** RabbitMQSender {

@Autowired

**private** AmqpTemplate;

@Value("${springboot.rabbitmq.exchange}")

**private** String exchange;

@Value("${springboot.rabbitmq.routingkey}")

**private** String routingkey;

String kafkaTopic = "java\_in\_use\_topic";

**public** **void** send(Employee company) {

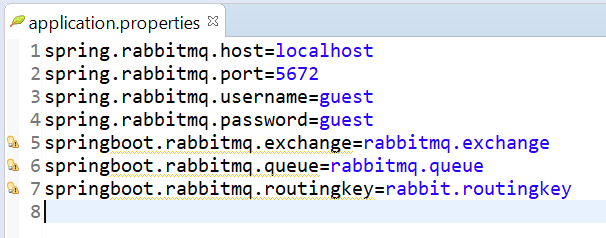
amqpTemplate.convertAndSend(exchange, routingkey, company);

System.***out***.println("Send msg = " + company);

}

}

Next define the following properties in application.properties-



Finally Define the Spring Boot Class with @SpringBootApplication annotation

package com.cloud;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SpringBootHelloWorldApplication {

public static void main(String[] args) {

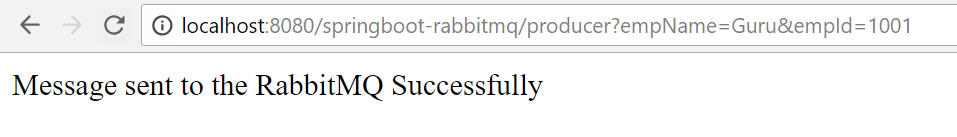
SpringApplication.run(

new Object[] { SpringBootHelloWorldApplication.class }, args);

}

}

Next start the Spring Boot Application by running it as a Java Application. Hit the url as follows-



his will trigger the message to be sent to the rabbitmq queue.  
Next go to the RabbitMQ console-**http://localhost:15672/**

