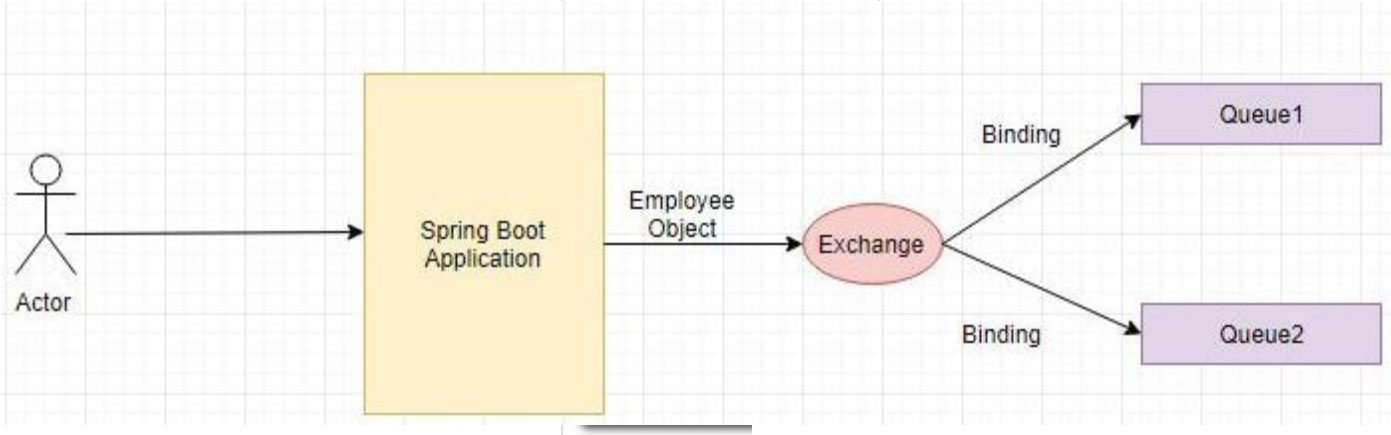
Spring Boot + RabbitMQ Hello World Example

In this post we will integrate Spring Boot and RabbitMQ instance.

In a previous post (/misc/rabbitmq-hello-world) we had seen how to get RabbitMQ up and running.



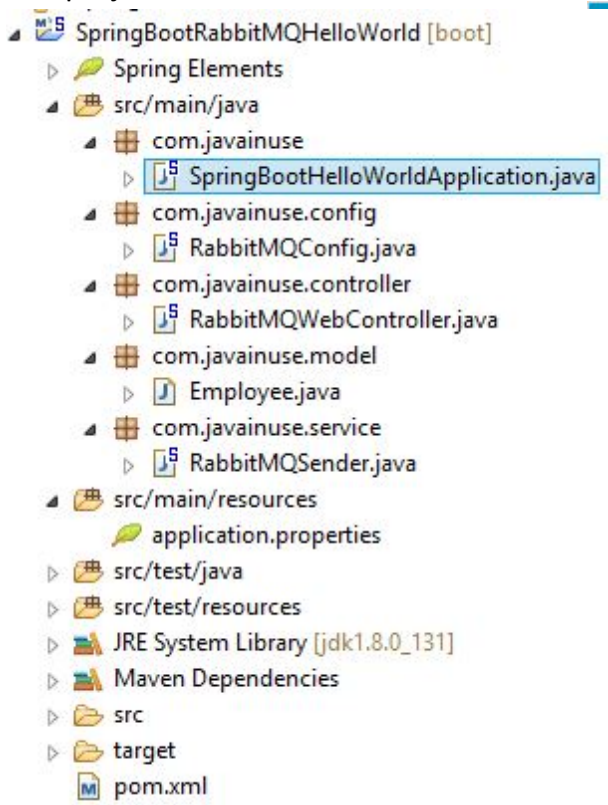
Video

This tutorial is explained in the below Youtube Video.

Spring Boot + RabbitMQ Integration Simple Example

Lets Begin-

The project will be as follows-



Define the pom.xml as follows- Add the **spring-boot-starter-amqp** dependency.

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apac
        <modelVersion>4.0.0</modelVersion>
        <groupId>com.javainuse</groupId>
        <artifactId>SpringBootRabbitMQHelloWorld</artifactId>
        <version>0.0.1-SNAPSHOT</version>

        <packaging>jar</packaging>


        <parent>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-parent</artifactId>
                <version>1.5.3.RELEASE</version>
                <relativePath />
        </parent>

        <properties>
                <project.build.sourceEncoding>UTF-8</project.build.sourceEncodi
                <project.reporting.outputEncoding>UTF-8</project.reporting.outp
                <java.version>1.8</java.version>
        </properties>

        <dependencies>
                <dependency>
                        <groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-amqp</artifactId>
                </dependency>
                <dependency>
                        <groupId>org.json</groupId>
                        <artifactId>json</artifactId>
                </dependency>
                <dependency>
                        <groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-web</artifactId>
                </dependency>
        </dependencies>

        <build>
                <plugins>
```

```
                        <plugin>
                                <groupId>org.springframework.boot</groupId>
                                <artifactId>spring-boot-maven-plugin</artifactI
                        </plugin>
                </plugins>
        </build>

</project>
```


Define the domain class Employee as follows-

```
package com.javainuse.model;

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

@JsonIdentityInfo(generator = ObjectIdGenerators.IntSequenceGenerator.class, pr
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.javainuse.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("")
	String queueName;

	@Value("")
	String exchange;

	@Value("")
	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();
        }


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


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

```
package com.javainuse.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.javainuse.model.Employee;
import com.javainuse.service.RabbitMQSender;

@RestController
@RequestMapping(value = "/javainuse-rabbitmq/")
public class RabbitMQWebController {

	@Autowired
	RabbitMQSender rabbitMQSender;

	@GetMapping(value = "/producer")
	public String producer(@RequestParam("empName") String empName,@Request

	Employee emp=new Employee();
	emp.setEmpId(empId);
	emp.setEmpName(empName);
		rabbitMQSender.send(emp);

		return "Message sent to the RabbitMQ JavaInUse 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.javainuse.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.javainuse.model.Employee;

@Service
public class RabbitMQSender {

	@Autowired
	private AmqpTemplate rabbitTemplate;

	@Value("")
	private String exchange;

	@Value("")
	private String routingkey;

	public void send(Employee company) {
		rabbitTemplate.convertAndSend(exchange, routingkey, company);
		System.out.println("Send msg = " + company);

	}
}
```


Next define th following properties in application.properties-

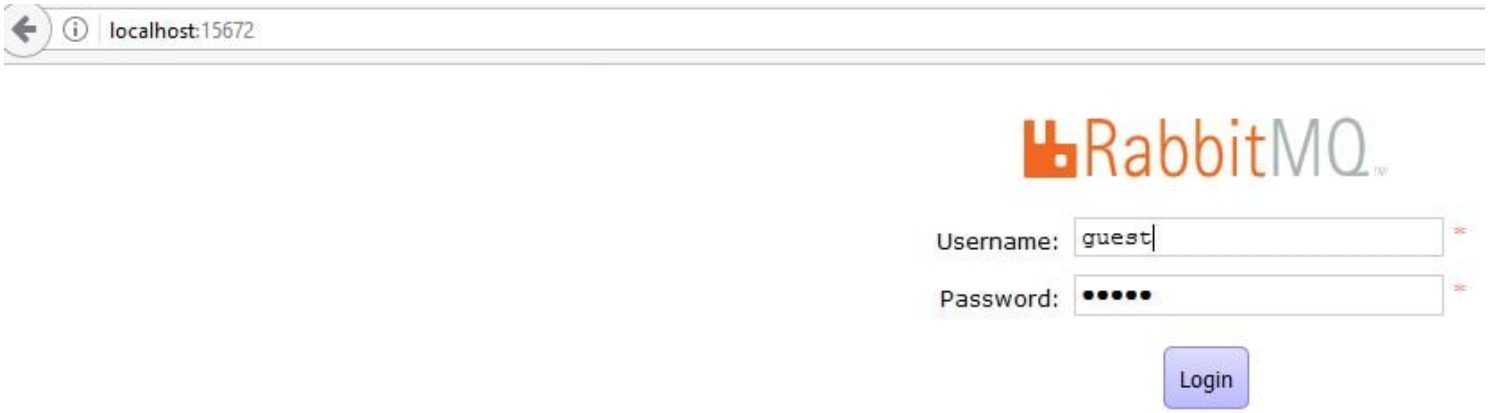
```
spring.rabbitmq.host=localhost
spring.rabbitmq.port=5672
spring.rabbitmq.username=guest
spring.rabbitmq.password=guest
javainuse.rabbitmq.exchange=javainuse.exchange
javainuse.rabbitmq.queue=javainuse.queue
javainuse.rabbitmq.routingkey=javainuse.routingkey
```

Finally Define the Spring Boot Class with @SpringBootApplication annotation

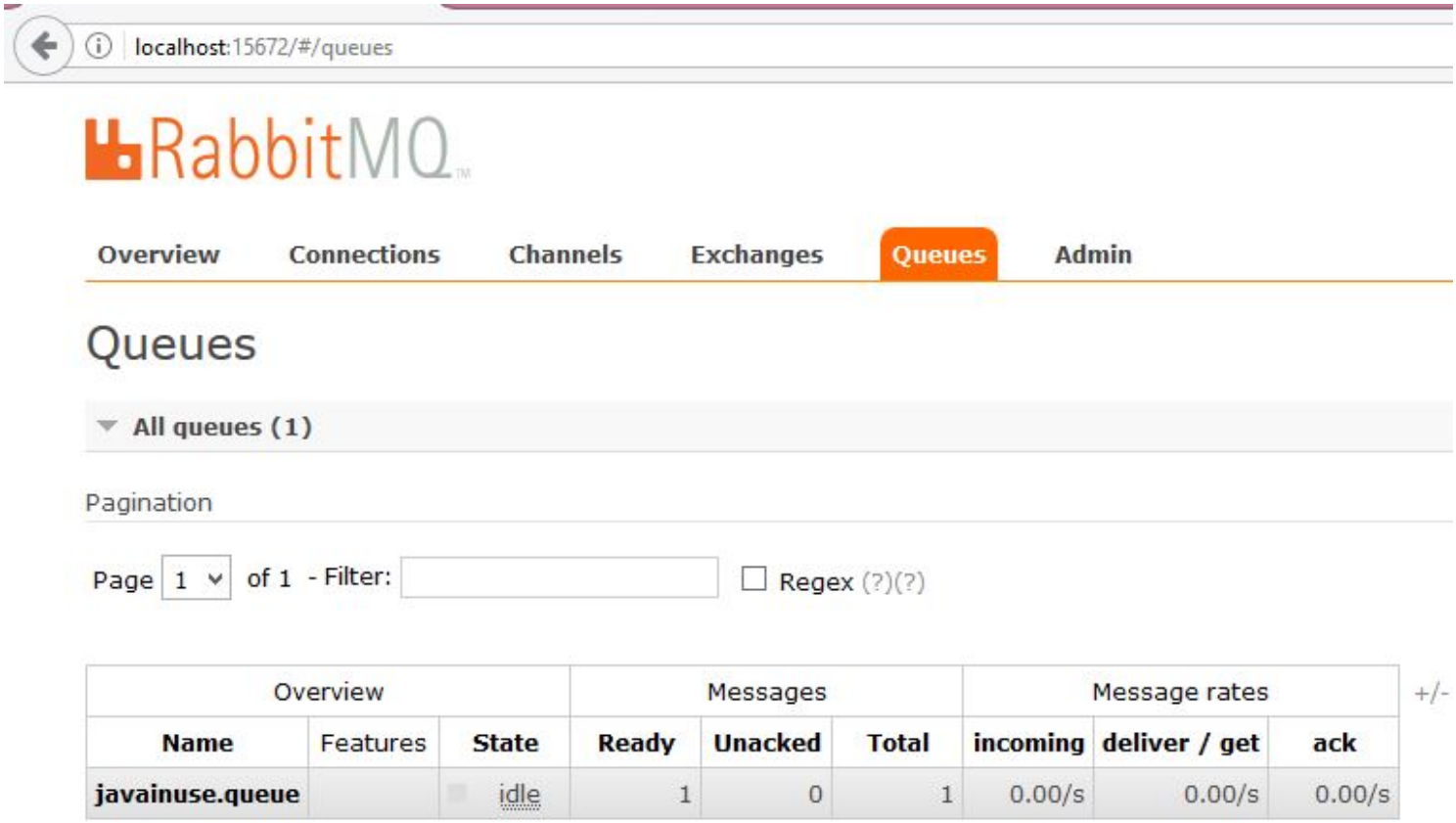
```
package com.javainuse;

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.
	}
}
```

We are done with the required Java code. Now lets start RabbitMQ. As we had explained in detail in the Getting started with RabbitMQ (/misc/rabbitmq-hello-world) perform the steps to start the RabbitMQ. Next start the Spring Boot Application by running it as a Java Application. Hit the url as follows-
http://localhost:8080/javainuse-rabbitmq/producer?empName=emp1&empId=emp001

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



We can see in the Queues section, a queue name java gets created and it has one message.



Download Source Code

Download it -

Spring Boot + RabbitMQ Hello World Example (/zip/spring/boot/SpringBootRabbitMQHelloWorld.rar)

Popular Posts

- File Transfer Using Java DSL Apache Camel (/camel/camel_dsl)
- Apache Camel Java DSL + Spring Integration Hello World Example (/camel/camelSpring)
- EIP patterns using Apache Camel (/camel/camel_EIP)
- Spring Cloud- Netflix Eureka + Ribbon Simple Example (/spring/spring_ribbon)
- Spring Cloud- Netflix Hystrix Circuit Breaker Simple Example (/spring/spring_hystrix_circuitbreaker)
- Spring Boot + Swagger Example Hello World Example (/spring/boot_swagger)
- Spring Boot Batch Simple example (/spring/bootbatch)
- Spring Boot + Apache Kafka Example (/spring/spring-boot-apache-kafka-hello-world)
- Spring Boot Admin Simple Example (/spring/boot-admin)
- Spring Boot Security - Introduction to OAuth (/spring/spring-boot-oauth-introduction)
- Spring Boot OAuth2 Part 1 - Getting The Authorization Code (/spring/spring-boot-oauth-authorization-code)
- Spring Boot OAuth2 Part 2 - Getting The Access Token And Using it to Fetch Data. (/spring/spring-boot-oauth-access-token)
- JBoss Drools Hello World-Stateful Knowledge Session using KieSession (/drools_hello_kie)
- Understand Drools Stateful vs Stateless Knowledge Session (/drools_states)
- JBoss Drools- Understanding Drools Decision Table using Simple Example (/drools/drools_decision)

See Also

- Spring Boot Interview Questions (/spring/SpringBootInterviewQuestions)
- Spring Batch Interview Questions (/spring/sprbatch_interview)
- Spring AOP Interview Questions (/spring/spring-AOP-interview-quesions)
- Angular 2 Interview Questions (/angular/ang2_intvw)
- Apache Camel Interview Questions (/camel/Apache_Camel_Questions)
- JBoss Fuse Interview Questions (/camel/JBoss_Fuse_Questions)
- Drools Interview Questions (/drools/drools_intvw)
- Java 8 Interview Questions (/java/java8_intvw)
- Spring Cloud Interview Questions (/spring/spring-cloud-interview-questions)
- Microservices Interview Questions (/spring/microservices-interview-quesions)
- Java HashMap and ConcurrentHashMap Interview Questions (/java/java_map_intvw)
- Mule ESB frequently asked interview questions (/misc/muleintvw)
- Apache Kafka Interview Questions (/misc/apache-kafka-interview-questions)
- Tosca Testing Tool Interview Questions (/misc/tosca-testing-tool-interview-questions)
- Top Maven Build Tool Interview Questions (/misc/maven-interview-questions)

- Top Gradle Build Tool Interview Questions (/misc/gradle-interview-questions)