SpringBootAndRabbitMQ

搭建生产者工程

实现步骤：
1.创建生产者SpringBoot工程
2.引入依赖坐标
3.编写yml配置，基本信息配置
4.定义交换机，队列以及绑定关系的配置类
5.注入RabbitTemplate，调用方法，完成消息发送

创建工程
项目名字： producer-springboot

添加依赖

```
<!--
1. 父工程依赖
-->
<parent>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-parent</artifactId>
<version>2.3.6.RELEASE</version>
</parent>


<dependencies>


<!--2. rabbitmq-->
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-amqp</artifactId>
</dependency>


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

</dependencies>
```

创建application.yml连接参数等配置文件；

```
spring:
rabbitmq:
host: 192.168.6.100
port: 5672
username: admin
password: 123456
virtual-host: /
```

创建配置类

```
@Configuration
public class RabbitMQConfig {
public static final String EXCHANGE_NAME = "boot_topic_exchange";
public static final String QUEUE_NAME = "boot_queue";

// 1 交换机
@Bean("bootExchange")
public Exchange bootExchange(){
return
ExchangeBuilder.topicExchange(EXCHANGE_NAME).durable(true).build();
}
//2.Queue 队列
@Bean("bootQueue")
public Queue bootQueue(){
return QueueBuilder.durable(QUEUE_NAME).build();
}
//3. 队列和交互机绑定关系 Binding
/*
1. 知道哪个队列
2. 知道哪个交换机
3. routing key
noargs()：表示不指定参数
*/
@Bean
public Binding bindQueueExchange(@Qualifier("bootQueue") Queue
queue,
@Qualifier("bootExchange") Exchange exchange){
return BindingBuilder.bind(queue).to(exchange).with("boot.#").noargs();
}
}
```

创建入口类

```
@SpringBootApplication
public class ProducerApplication {
public static void main(String[] args) {
SpringApplication.run(ProducerApplication.class);
}
}
```

发送消息

```
@SpringBootTest
@RunWith(SpringRunner.class)
public class ProducerTest {

@Autowired
private RabbitTemplate rabbitTemplate;

/**
* 第一个参数：交换机名字
* 第二个参数：routingKey
* 第三个参数：发送的消息
*/
@Test
public void testSend(){
rabbitTemplate.convertAndSend(RabbitMQConfig.EXCHANGE_NAME,"boo
t.haha","mq hello");
}
}
```

运行测试类,发送消息

搭建消费者工程

实现步骤
1.创建消费者SpringBoot工程
2.引入start,依赖坐标
3.编写yml配置,基本信息配置
4.定义监听类,使用@RabbitListener注解完成队列监听。

创建工程
创建项目 consumer-springboot

添加依赖
修改pom.xml文件内容为如下：

```
<parent>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-parent</artifactId>
<version>2.3.6.RELEASE</version>
</parent>
<dependencies>
<!--RabbitMQ 启动依赖-->
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-amqp</artifactId>
</dependency>
</dependencies>
```

配置整合
创建application.yml连接参数等配置文件；

```
spring:
rabbitmq:
host: 192.168.6.100
port: 5672
username: admin
password: 123456
virtual-host: /
```

消息监听器
队列监听器
创建 com.atguigu.listener.RabbimtMQListener

```
package com.atguigu.listener;

import org.springframework.amqp.core.Message;
import org.springframework.amqp.rabbit.annotation.RabbitListener;
import org.springframework.stereotype.Component;

@Component
public class RabbimtMQListener {
@RabbitListener(queues = "boot_queue")
public void listenerQueue(Message message){
System.out.println(new String(message.getBody()));
}
}
```

创建入口类

```
package com.atguigu;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class ConsumerApplication {
public static void main(String[] args) {
SpringApplication.run(ConsumerApplication.class, args);
}
}
```

运行applicaion

小结：

SpringBoot提供了快速整合RabbitMQ的方式
基本信息在yml中配置,队列交互机以及绑定关系在配置类中使用Bean的方式配置
生产端直接注入RabbitTemplate完成消息发送
消费端直接使用@RabbitListener完成消息接收

作者：@mg736115 | 来自：知犀思维导图