**声明：此文档为个人翻译官方文档，未经允许请勿外传。**

**翻译者：杨德军**

Routing

**(using the spring-amqp)**

**前置条件**

本教程假定RabbitMQ在默认端口（5672）上的本地主机上安装并运行。 如果您使用不同的host, port or credentials，则connections设置需要进行调整。

**从哪里获得帮助**

如果您在阅读本教程时遇到困难，可以通过邮件列表与我们联系[contact us](https://groups.google.com/forum/#!forum/rabbitmq-users)。

在[previous tutorial](http://www.rabbitmq.com/tutorials/tutorial-three-spring-amqp.html)中，我们构建了一个简单的fanout exchange。 我们能够将messages broadcast给许多receivers。

在本教程中，我们将添加一个功能 - 我们将只能订阅一部分消息。 例如，我们将只能将消息导向某些感兴趣的颜色（“橙色”，“黑色”，“绿色”），同时仍然能够在控制台上打印所有消息。

Bindings

在前面的例子中，我们已经创建了bindings。 你可以在我们的Tut3Config文件中找到这个代码：

@Bean

public Binding binding1(FanoutExchange fanout,

Queue autoDeleteQueue1) {

return BindingBuilder.bind(autoDeleteQueue1).to(fanout);

}

binding是exchange和queue之间的关系。 这可以简单地理解为：queue对来自该exchange的messages感兴趣。

Bindings可以采用额外的routingKey参数。 Spring-amqp使用流畅的API来使这种关系变得非常清晰。 我们通过exchange并将queue放入BindingBuilder中，并简单地将queue “bind”到exchange “使用routing key”，如下所示：

@Bean

public Binding binding1a(DirectExchange direct,

Queue autoDeleteQueue1) {

return BindingBuilder.bind(autoDeleteQueue1)

.to(direct)

.with("orange");

}

binding key的含义取决于exchange type。 我们之前使用的fanout exchanges简单地忽略了它的值。

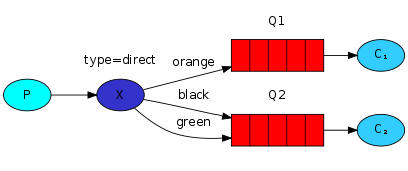
Direct exchange

我们来自上一个教程的messaging system将所有messages broadcasts给所有consumers。 我们希望扩展该功能以允许基于其颜色类型过滤消息。 例如，我们可能需要一个将日志消息写入磁盘的程序，以仅接收严重错误，而不会在警告或信息日志消息中浪费磁盘空间。

我们正在使用一个fanout exchange，这没有给我们很大的灵活性 - 它只能进行盲目的broadcasting。

我们将使用direct exchange。 direct exchange背后的routing algorithm很简单 - 消息进入队列，谁binding key与message的routing key完全匹配。

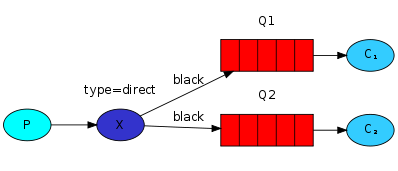
为了说明这一点，请考虑以下设置：



在这个设置中，我们可以看到有两个queues bound的direct exchange X. 第一个queue用binding key orange绑定，第二个队列有两个绑定，一个绑定键为black，另一个为green。

在这种设置中，使用路由键orange 发布到交换机的消息将被路由到队列Q1。 带有black 或green 路由键的消息将进入Q2。 所有其他消息将被丢弃。

Multiple bindings



使用相同的binding key绑定多个队列是完全合法的。 在我们的例子中，我们可以使用key black添加X和Q1之间的绑定。 在这种情况下，direct exchange就像fanout 一样，并将消息broadcast到所有匹配的queues。 带有routing key black的message将delivered到Q1和Q2。

Publishing messages

我们将使用这个模型用于我们的routing system。 我们将发送消息给direct exchange，而不是fanout。 我们将提供颜色作为routing key。 这样receiving program将能够选择想要receive（或subscribe）的颜色。 我们先关注发送消息。

与往常一样，我们在Tut4Config中执行一些spring boot配置：

@Bean

public FanoutExchange fanout() {

return new FanoutExchange("tut.fanout");

}

我们准备发送消息。 如图所示，颜色可以是'orange', 'black', or 'green'之一。

Subscribing

Receiving messages的方式与上一个教程中的一样，只有一个例外 - 我们将为每个我们感兴趣的颜色创建一个新的binding。这也将被编写到进入Tut4Config中

@Bean

public DirectExchange direct() {

return new DirectExchange("tut.direct");

}

...

@Bean

public Binding binding1a(DirectExchange direct,

Queue autoDeleteQueue1) {

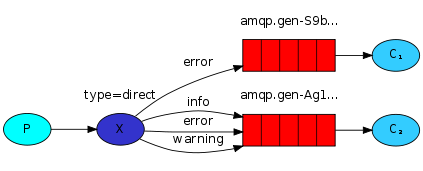
return BindingBuilder.bind(autoDeleteQueue1)

.to(direct)

.with("orange");

}

Putting it all together



和前面的教程一样，为本教程创建一个名为“tut4”的package，并创建Tut4Config类。 Tut4Config.java类的代码如下：

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

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.context.annotation.Profile;

@Profile({"tut4","routing"})

@Configuration

public class Tut4Config {

@Bean

public DirectExchange direct() {

return new DirectExchange("tut.direct");

}

@Profile("receiver")

private static class ReceiverConfig {

@Bean

public Queue autoDeleteQueue1() {

return new AnonymousQueue();

}

@Bean

public Queue autoDeleteQueue2() {

return new AnonymousQueue();

}

@Bean

public Binding binding1a(DirectExchange direct,

Queue autoDeleteQueue1) {

return BindingBuilder.bind(autoDeleteQueue1)

.to(direct)

.with("orange");

}

@Bean

public Binding binding1b(DirectExchange direct,

Queue autoDeleteQueue1) {

return BindingBuilder.bind(autoDeleteQueue1)

.to(direct)

.with("black");

}

@Bean

public Binding binding2a(DirectExchange direct,

Queue autoDeleteQueue2) {

return BindingBuilder.bind(autoDeleteQueue2)

.to(direct)

.with("green");

}

@Bean

public Binding binding2b(DirectExchange direct,

Queue autoDeleteQueue2) {

return BindingBuilder.bind(autoDeleteQueue2)

.to(direct)

.with("black");

}

@Bean

public Tut4Receiver receiver() {

return new Tut4Receiver();

}

}

@Profile("sender")

@Bean

public Tut4Sender sender() {

return new Tut4Sender();

}

}

sender 类如下:

import org.springframework.amqp.core.DirectExchange;

import org.springframework.amqp.rabbit.core.RabbitTemplate;

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

import org.springframework.scheduling.annotation.Scheduled;

public class Tut4Sender {

@Autowired

private RabbitTemplate template;

@Autowired

private DirectExchange direct;

private int index;

private int count;

private final String[] keys = {"orange", "black", "green"};

@Scheduled(fixedDelay = 1000, initialDelay = 500)

public void send() {

StringBuilder builder = new StringBuilder("Hello to ");

if (++this.index == 3) {

this.index = 0;

}

String key = keys[this.index];

builder.append(key).append(' ');

builder.append(Integer.toString(++this.count));

String message = builder.toString();

template.convertAndSend(direct.getName(), key, message);

System.out.println(" [x] Sent '" + message + "'");

}

}

Tut4Receiver.java 如下:

import org.springframework.amqp.rabbit.annotation.RabbitListener;

import org.springframework.util.StopWatch;

public class Tut4Receiver {

@RabbitListener(queues = "#{autoDeleteQueue1.name}")

public void receive1(String in) throws InterruptedException {

receive(in, 1);

}

@RabbitListener(queues = "#{autoDeleteQueue2.name}")

public void receive2(String in) throws InterruptedException {

receive(in, 2);

}

public void receive(String in, int receiver) throws InterruptedException {

StopWatch watch = new StopWatch();

watch.start();

System.out.println("instance " + receiver + " [x] Received '" + in + "'");

doWork(in);

watch.stop();

System.out.println("instance " + receiver + " [x] Done in " +

watch.getTotalTimeSeconds() + "s");

}

private void doWork(String in) throws InterruptedException {

for (char ch : in.toCharArray()) {

if (ch == '.') {

Thread.sleep(1000);

}

}

}

}

像平常一样编译, 参见maven编译[tutorial one](http://www.rabbitmq.com/tutorials/tutorial-one-spring-amqp.html)，并执行jar中的选项。

mvn clean package

在一个终端窗口中，您可以运行：

java -jar target/rabbit-tutorials-1.7.1.RELEASE.jar

--spring.profiles.active=routing,receiver

--tutorial.client.duration=60000

并在另一个终端窗口中运行sender

java -jar target/rabbit-tutorials-1.7.1.RELEASE.jar

--spring.profiles.active=routing,sender

--tutorial.client.duration=60000

[Tut4Receiver.java source](https://github.com/rabbitmq/rabbitmq-tutorials/blob/master/spring-amqp/src/main/java/org/springframework/amqp/tutorials/tut4/Tut4Receiver.java) and [Tut4Sender.java source](https://github.com/rabbitmq/rabbitmq-tutorials/blob/master/spring-amqp/src/main/java/org/springframework/amqp/tutorials/tut4/Tut4Sender.java)的完整源代码。 该配置位于 [Tut4Config.java source](https://github.com/rabbitmq/rabbitmq-tutorials/blob/master/spring-amqp/src/main/java/org/springframework/amqp/tutorials/tut4/Tut4Config.java)中。

转到[tutorial 5](http://www.rabbitmq.com/tutorials/tutorial-five-spring-amqp.html)以了解如何根据pattern来listen messages。