# 08-rabbitmq-消息路由-spring

## 先决条件

本教程假定RabbitMQ已在标准端口（5672）上的localhost上安装并运行。如果使用不同的主机，端口或凭据，连接设置将需要调整。

## 消息路由

在上一个教程中，我们构建了一个简单的扇出交换。我们能够向许多接收器广播消息。

在本教程中，我们将为其添加一个功能 - 我们将只能订阅一部分消息。例如，我们将能够仅将消息指向感兴趣的某些颜色（“橙色”，“黑色”，“绿色”），同时仍然能够在控制台上打印所有消息

## 绑定

在之前的例子中，我们已经创建了绑定。您可以在我们的Tut3Config文件中回忆一下这样的代码：

|  |
| --- |
| @Bean  public Binding binding1(FanoutExchange fanout,  Queue autoDeleteQueue1) {  return BindingBuilder.bind(autoDeleteQueue1).to(fanout);  } |

绑定是交换器和队列之间的关系。这可以简单地读为：队列对来自此交换器的消息感兴趣。

绑定可以占用一个额外的路由密钥参数。Spring-amqp使用流畅的API来使这种关系非常清楚。我们将交换器和队列传入到BindingBuilder，并将队列“路由密钥”绑定到“交换机”，如下所示：

|  |
| --- |
| @Bean  public Binding binding1a(DirectExchange direct,  Queue autoDeleteQueue1) {  return BindingBuilder.bind(autoDeleteQueue1)  .to(direct)  .with("orange");  } |

绑定密钥的含义取决于交换类型。扇出交交换器，这是我们以前使用的，简单地忽略它的值。

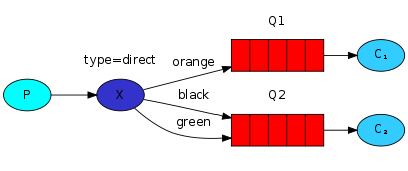
## 直接交换

我们从上一个教程的消息系统向所有消费者广播所有消息。我们希望将其扩展为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以仅接收关键错误的程序，而不会浪费警告或信息日志消息上的磁盘空间。

我们正在使用一个扇出的交换器，它给我们不是很大的灵活性 - 它只能无意识地广播。

我们将使用直接交换。直接交换背后的路由算法很简单 - 消息传递到绑定密钥与消息的路由密钥完全匹配的队列 。

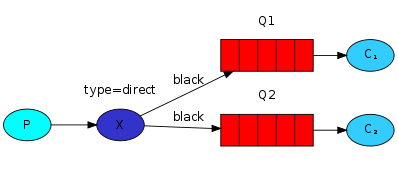
为了说明，请考虑以下设置：



在这个设置中，我们可以看到直接交换X与两个队列绑定。第一个队列与绑定键橙色绑定，第二个队列有两个绑定，一个绑定键为黑色，另一个绑定为绿色。

在这样的设置中，发布到具有路由密钥橙色的交换机的消息将被路由到队列Q1。具有黑色或绿色路由选择密钥的消息将转到Q2。所有其他消息将被丢弃。

## 多重绑定



使用相同的绑定键绑定多个队列是完全合法的。在我们的例子中，我们可以在X和Q1之间添加绑定键黑色的绑定。在这种情况下，直接交换将表现得像扇出，并将消息广播到所有匹配的队列。具有路由密钥黑色的消息将传送到Q1和Q2。

## 发布消息

我们的路由系统将使用此模型。而不是扇出，我们将发送消息到直接交换。我们将提供颜色作为路由密钥。这样接收程序将能够选择要接收（或订阅）的颜色。我们首先关注发送消息。

一如以往，我们在Tut4Config中做一些spring启动配置：和往常一样，我们需要先建立一个交换器：

|  |
| --- |
| @Bean  public FanoutExchange fanout() {  return new FanoutExchange("tut. direct");  } |

我们准备发送消息，我们准备发送消息。如图所示，颜色可以是“橙色”，“黑色”或“绿色”之一。

## 订阅

接收消息将像上一个教程一样工作，除了一个例外 - 我们将为每个我们感兴趣的颜色创建一个新的绑定。

|  |
| --- |
| @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");  } |

## 把它们放在一起



如以前的教程一样，为本教程创建一个名为“tut4”的新包，并创建Tut4Config类。Tut4Config.java类的代码：

|  |
| --- |
| **package** com.example.rabbitmq.tut4;  **import** org.springframework.amqp.core.\*; **import** org.springframework.context.annotation.Bean; **import** org.springframework.context.annotation.Configuration; **import** org.springframework.context.annotation.Profile;  */\*\*  \* Author: 王俊超  \* Date: 2017-06-17 20:12  \* All Rights Reserved !!!  \*/* @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 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();  } } |

我们的发送者类的代码是：

|  |
| --- |
| **package** com.example.rabbitmq.tut4;  **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;  */\*\*  \* Author: 王俊超  \* Date: 2017-06-17 20:15  \* All Rights Reserved !!!  \*/* **public class** Tut4Sender {  @Autowired  **private** RabbitTemplate **template**;   @Autowired  **private** DirectExchange **direct**;   **private int index**;   **private int count**;   **private final** String[] **keys** = {**"orange"**, **"black"**, **"green"**};   */\*\*  \* 定期向三个不同的路由中发送消息  \*/* @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的代码是：

|  |
| --- |
| **package** com.example.rabbitmq.tut4;  **import** org.springframework.amqp.rabbit.annotation.RabbitListener; **import** org.springframework.util.StopWatch;  */\*\*  \* Author: 王俊超  \* Date: 2017-06-17 20:15  \* All Rights Reserved !!!  \*/* **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**);  }  }  } } |

## 运行

先运行接收者，需要添加运行参数

--spring.profiles.active=routing,receiver --tutorial.client.duration=60000

再运行发送者，需要添加运行参数

--spring.profiles.active=routing,sender --tutorial.client.duration=60000