# 09-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。

## 主题

在上一个教程中，我们改进了我们的日志记录系统。而不是使用只能够进行虚拟广播的扇出交换机，我们使用直接交换器，并且有可能选择性地接收日志。

虽然使用直接交换器改进了我们的系统，但它仍然有限制 - 它不能基于多个标准进行路由选择。

在我们的日志记录系统中，我们可能不仅要根据严重性订阅日志，还可以基于发出日志的源进行订阅。您可能会从syslog unix工具中了解这一概念，该工具根据严重性（信息/警告/错误...）和设施（auth / cron / kern ...）路由日志。

这将给我们很大的灵活性 - 我们可能想监听来自“cron”的重要错误，也可以监听“kern”的所有日志。

要在我们的日志记录系统中实现，我们需要了解一个更为复杂的主题交换。

## 主题交换

发送到主题交换器的消息不能有任意的 routing\_key - 它必须是由点分隔的单词列表。这些词可以是任何东西，但通常它们指定与消息相关联的一些功能。几个有效的路由密钥示例：“stock.usd.nyse”，“nyse.vmw”，“quick.orange.rabbit”。路由密钥中可以有任意多的单词，最多可达255个字节。

绑定键也必须是相同的形式。主题交换器背后的逻辑类似于直接的一个 - 使用特定路由密钥发送的消息将被传递到与匹配的绑定密钥绑定的所有队列。但是，绑定密钥有两个重要的特殊情况：

\*（星）可以替代一个字。

#（哈希）可以替换零个或多个单词。

在一个例子中最简单的解释一下：



在这个例子中，我们将发送所有描述动物的消息。消息将使用由三个字（两个点）组成的路由密钥发送。路由密钥中的第一个字将描述速度，第二个颜色和第三个种类：“<speed>.<color>.<species>”。

我们创建了三个绑定：Q1绑定键“\* .orange.\*”和Q2与“\*.\*.rabbit”和“lazy.#” 绑定。

这些绑定可以总结为：

Q1对所有的橙色动物感兴趣。

Q2想听听有关兔子的一切，以及关于懒惰动物的一切。

将路由密钥设置为“quick.orange.rabbit ”的消息将传递给两个队列。消息“lazy.orange.elephant ”也会发送到他们两个队列。另一方面，“quick.orange.fox”只会转到第一个队列，而“lazy.brown.fox”只能到第二个。“lazy.pink.rabbit”将只被传递到第二个队列一次，即使它匹配两个绑定。“quick.brown.fox”不匹配任何绑定，所以它将被丢弃。

如果我们违反合同并发送一个或四个字的消息，如“orange”或“quick.orange.male.rabbit”，会发生什么？那么这些消息将不会匹配任何绑定，并将丢失。

另一方面，“ lazy.orange.male.rabbit ”即使它有四个字，将匹配上一个绑定，并将被传递到第二个队列。

## 主题交换

主题交换是强大的，可以像其他交换器一样行事。

当队列用“#”（哈希）绑定键绑定时，它将接收所有消息，而不管路由密钥如扇出交换。

当特殊字符“\*”（星号）和“#”（哈希）不用于绑定时，主题交换将表现得像一个直接交换。

## 把它们放在一起

我们将在我们的日志记录系统中使用主题交换。我们将从一个工作假设开始，日志的路由密钥将有两个单词：“<facility>.<severity>”。

代码与上一个教程几乎相同，EmitLogTopic.java的代码：

|  |
| --- |
| **package** com.example.rabbitmq;  **import** com.rabbitmq.client.BuiltinExchangeType; **import** com.rabbitmq.client.Channel; **import** com.rabbitmq.client.Connection; **import** com.rabbitmq.client.ConnectionFactory;  */\*\*  \* Author: 王俊超  \* Date: 2017-06-17 20:35  \* All Rights Reserved !!!  \*/* **public class** EmitLogTopic {  **private static final** String ***EXCHANGE\_NAME*** = **"topic\_logs"**;   **public static void** main(String[] argv) **throws** Exception {  Connection connection = **null**;  Channel channel = **null**;   ConnectionFactory factory = **new** ConnectionFactory();  factory.setHost(**"localhost"**);   connection = factory.newConnection();  channel = connection.createChannel();  *// 创建一个主题通道* channel.exchangeDeclare(***EXCHANGE\_NAME***, BuiltinExchangeType.***TOPIC***);  String routingKey = *getRouting*(argv);  String message = getMessage(argv);  *// 使用路由向通道中发送消息* channel.basicPublish(***EXCHANGE\_NAME***, routingKey, **null**, message.getBytes(**"UTF-8"**));  System.***out***.println(**" [x] Sent '"** + routingKey + **"':'"** + message + **"'"**);  connection.close();  }   */\*\*  \* 获取中路由字符串  \** ***@param strings*** *\** ***@return*** *\*/* **private static** String getRouting(String[] strings) {  **if** (strings.**length** < **1**) {  **return "anonymous.info"**;  }  **return** strings[**0**];  }   */\*\*  \* 获取消息  \*  \** ***@param strings*** *\** ***@return*** *\*/* **private static** String getMessage(String[] strings) {  **if** (strings.**length** < **2**) {  **return "Hello World!"**;  }  **return** *joinStrings*(strings, **" "**, **1**);  }   **private static** String joinStrings(String[] strings, String delimiter, **int** startIndex) {  **int** length = strings.**length**;  **if** (length == **0**) {  **return ""**;  }  **if** (length < startIndex) {  **return ""**;  }  StringBuilder words = **new** StringBuilder(strings[startIndex]);  **for** (**int** i = startIndex + **1**; i < length; i++) {  words.append(delimiter).append(strings[i]);  }  **return** words.toString();  } } |

ReceiveLogsTopic.java的代码：

|  |
| --- |
| **package** com.example.rabbitmq;  **import** com.rabbitmq.client.\*;  **import** java.io.IOException;  */\*\*  \* Author: 王俊超  \* Date: 2017-06-17 20:39  \* All Rights Reserved !!!  \*/* **public class** ReceiveLogsTopic {  **private static final** String ***EXCHANGE\_NAME*** = **"topic\_logs"**;   **public static void** main(String[] argv) **throws** Exception {  ConnectionFactory factory = **new** ConnectionFactory();  factory.setHost(**"localhost"**);  Connection connection = factory.newConnection();  Channel channel = connection.createChannel();   channel.exchangeDeclare(***EXCHANGE\_NAME***, BuiltinExchangeType.***TOPIC***);  String queueName = channel.queueDeclare().getQueue();   **if** (argv.**length** < **1**) {  System.***err***.println(**"Usage: ReceiveLogsTopic [binding\_key]..."**);  System.*exit*(**1**);  }   *// 通过参数绑定不同的路由* **for** (String bindingKey : argv) {  channel.queueBind(queueName, ***EXCHANGE\_NAME***, bindingKey);  }   System.***out***.println(**" [\*] Waiting for messages. To exit press CTRL+C"**);   Consumer consumer = **new** DefaultConsumer(channel) {  @Override  **public void** handleDelivery(String consumerTag, Envelope envelope,  AMQP.BasicProperties properties, **byte**[] body) **throws** IOException {  String message = **new** String(body, **"UTF-8"**);  System.***out***.println(**" [x] Received '"** + envelope.getRoutingKey() + **"':'"** + message + **"'"**);  }  };  channel.basicConsume(queueName, **true**, consumer);  } } |

## 运行

接收所有日志，运行接收者，添加“#”参数。

接收所有“kern”日志，添加“kern.\*”参数。

如果您只想听到关于“critical”日志的信息，添加“\* .critical”参数：

绑定多个消息路由，可以添加参数“kern.\*”，“\* .critical”。可以运行发送者并添加运行参数“kern.critical”“A critical kernel error”进行测试。

注意使用时的中文引号需要替换成英文引号。