# 07-rabbitmq-消息路由

## 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日志记录系统。我们能够向许多接收器广播日志消息。

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，我们将能够仅将关键错误消息指向日志文件（以节省磁盘空间），同时仍然可以在控制台上打印所有日志消息。

## 绑定

在之前的例子中，我们已经创建了绑定。你可能会记得如下代码：

|  |
| --- |
| channel.queueBind(queueName, EXCHANGE\_NAME, ""); |

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。为了避免与basic\_publish参数混淆，我们将其称为 绑定密钥。这是我们如何使用键创建绑定：

|  |
| --- |
| channel.queueBind(queueName, EXCHANGE\_NAME, "black"); |

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。

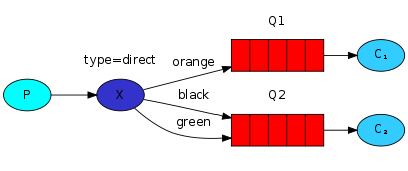
## 直接交换

我们上一个教程的日志系统向所有消费者广播所有消息。我们希望扩展它以允许基于其严重性过滤消息。例如，我们可能需要一个将日志消息写入磁盘的程序，只接收严重错误，而不会在警告或信息日志消息上浪费磁盘空间。

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 。

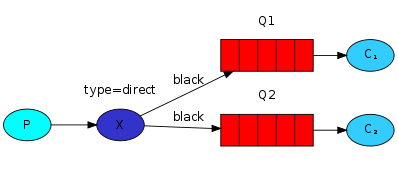
为了说明，请考虑以下设置：



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。

## 发出日志

我们将使用此模型进行日志记录系统。而不是扇出，我们将发送消息到直接交换的交换器。我们将提供日志严重性作为路由密钥。这样接收程序将能够选择要接收的严重性。我们首先关注发射日志。

和往常一样，我们需要先建立一个交换器：

|  |
| --- |
| channel.exchangeDeclare(EXCHANGE\_NAME, "direct"); |

我们准备发送消息：

|  |
| --- |
| channel.basicPublish(EXCHANGE\_NAME, severity, null, message.getBytes()); |

为了简化事情，我们假设“严重性”可以是“信息”，“警告”，“错误”之一。

|  |
| --- |
| String queueName = channel.queueDeclare().getQueue();  for(String severity : argv){  channel.queueBind(queueName, EXCHANGE\_NAME, severity);  } |

## 把它们放在一起



EmitLogDirect.java类的代码：

|  |
| --- |
| **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 19:39  \* All Rights Reserved !!!  \*/* **public class** EmitLogDirect {  **private static final** String ***EXCHANGE\_NAME*** = **"direct\_logs"**;   **public static void** main(String[] argv) **throws** Exception {   ConnectionFactory factory = **new** ConnectionFactory();  factory.setHost(**"localhost"**);  Connection connection = factory.newConnection();  Channel channel = connection.createChannel();   *// 采用直接绑定* channel.exchangeDeclare(***EXCHANGE\_NAME***, BuiltinExchangeType.***DIRECT***);   String severity = *getSeverity*(argv);  String message = *getMessage*(argv);   channel.basicPublish(***EXCHANGE\_NAME***, severity, **null**, message.getBytes(**"UTF-8"**));  System.***out***.println(**" [x] Sent '"** + severity + **"':'"** + message + **"'"**);   channel.close();  connection.close();  }   **private static** String getSeverity(String[] strings) {  **if** (strings.**length** < **1**) {  **return "info"**;  }  **return** strings[**0**];  }   **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();  } } |

ReceiveLogsDirect.java的代码：

|  |
| --- |
| **package** com.example.rabbitmq;  **import** com.rabbitmq.client.\*;  **import** java.io.IOException;  */\*\*  \* Author: 王俊超  \* Date: 2017-06-17 19:42  \* All Rights Reserved !!!  \*/* **public class** ReceiveLogsDirect {  **private static final** String ***EXCHANGE\_NAME*** = **"direct\_logs"**;   **public static void** main(String[] argv) **throws** Exception {  ConnectionFactory factory = **new** ConnectionFactory();  factory.setHost(**"localhost"**);  Connection connection = factory.newConnection();  Channel channel = connection.createChannel();   *// 采用直接绑定* channel.exchangeDeclare(***EXCHANGE\_NAME***, BuiltinExchangeType.***DIRECT***);  String queueName = channel.queueDeclare().getQueue();   **if** (argv.**length** < **1**) {  System.***err***.println(**"Usage: ReceiveLogsDirect [info] [warning] [error]"**);  System.*exit*(**1**);  }   **for** (String severity : argv) {  channel.queueBind(queueName, ***EXCHANGE\_NAME***, severity);  }   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

如果您只想将“警告”和“错误”（而不是“信息”）保存到文件中，需要ReceiveLogsDirect类，并且添加运行参数：

|  |
| --- |
| warning error > logs\_from\_rabbit.log |

如果您想在屏幕上看到所有的日志信息，需要需要ReceiveLogsDirect类，并且添加运行参数：

|  |
| --- |
| info warning error |

而且，例如，要发出错误日志消息，运行EmitLogDirect类，并且添加运行参数：

|  |
| --- |
| error "Run. Run. Or it will explode." |